Microsoft Power Apps documentation

Discover how to make the most of Power Apps with online training courses, docs, and videos covering product capabilities and how-to articles. Learn how to quickly build low-code apps that modernize processes and solve tough business challenges in your organization using Power Apps.



GET STARTED
What is Power Apps?



WHAT'S NEW
What's new in Power Apps?



TRAINING
Power Apps training



HOW-TO GUIDE
Troubleshoot

Create and design

Try/Buy

- Try Power Apps
- Buy Power Apps

Create apps

- Choose an app type
- Use the Plan designer
- Create a canvas app
- Create a model-driven app
- Create a code app

AI copilot capabilities

- Get started with AI copilot
- Use the Plan designer
- Overview of the new experience (preview)
- Build apps through conversation
- Add a Copilot control to a canvas app (preview)

Create and manage data

Use solutions

- What are solutions?

Integrate with other products

What is Microsoft Dataverse?

Create and use tables and metadata

Apply business logic

Create and use dataflows

Implement healthy application lifecycle management (ALM) using solutions

Deploy solutions using pipelines

Visualize data with Power BI

Get real-time insights with Azure Synapse Link

Add and use apps in Teams

Best practices guidance

Power Apps coding guidelines

Collaborate and co-develop apps

Administer and extend

Administer Power Apps

Use the Power Platform admin center

Understand Power Apps licensing

Understand security and governance

Extend the data platform

Dataverse developer guide

Work with data using code

Apply business logic using code

Create client applications

Extend model-driven apps

Model-driven apps developer guide

Apply business logic using client scripting

Use code components in model-driven apps

Extend canvas apps

Canvas apps developer guide

Create and use custom connectors

Develop offline-capable canvas apps

Create and use canvas components

Use

Use apps on mobile devices

Use apps in web browsers

Understand basic navigation

Use apps in Teams

Embed a canvas app as a tab

Install and use Power Apps mobile

Use Power Apps for Windows

Troubleshoot issues

Search records

View reports

Embed a canvas app as a personal app

Embed a model-driven app as a tab

Embed a model-driven app as a personal app

Related content

Power Platform

One-stop learning resource for Power Platform applications and features, with admin, developer, and guidance...

Power Automate

Create automated workflows between your favorite apps and services to synchronize files, get notifications, collect...

Power BI

Turn your unrelated sources of data into coherent, visually immersive, and interactive insights.

Power Pages

Design, host, and administer secure, modern, and low-code business websites.

Dynamics 365

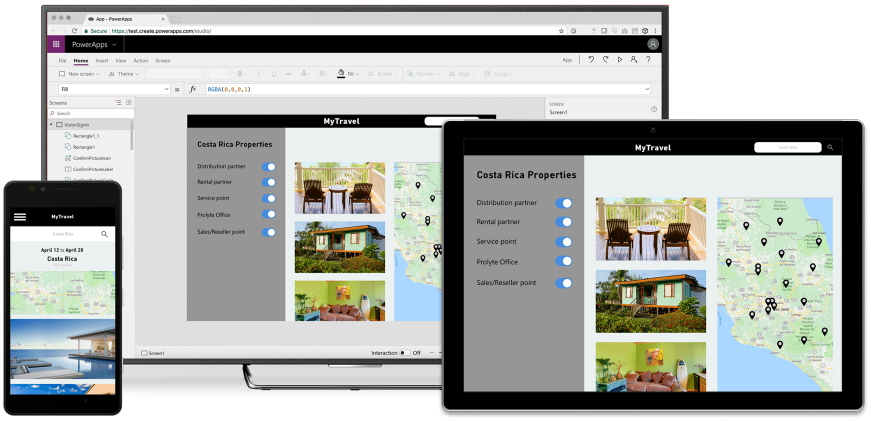
Empower your organization with the next generation of CRM and ERP applications.

Microsoft Azure

Learn how this ever-expanding set of cloud computing services can help your organization meet its busines...

What is Power Apps?

Power Apps is a suite of apps, services, and connectors, as well as a data platform, that provides a rapid development environment to build custom apps for your business needs. With Power Apps, you can quickly build custom business apps that connect to your data stored either in the underlying data platform (Microsoft Dataverse) or in many online and on-premises data sources (such as SharePoint, Microsoft 365, Dynamics 365, SQL Server, and so on).



Apps built with Power Apps provide rich business logic and workflow capabilities to transform your manual business operations into digital, automated processes. What's more, apps built with Power Apps have a responsive design and can run seamlessly in browser and on mobile devices (phone or tablet). Power Apps "democratizes" the business-app-building experience by enabling users to create feature-rich, custom business apps without writing code.

Power Apps also provides an extensible platform that lets pro developers programmatically interact with data and metadata, apply business logic, create custom connectors, and integrate with external data.

Power Apps and Microsoft Copilot let you describe the app you want and AI designs it. Make edits, add automation, and write code—all with Copilot in Power Apps.

For more information:

- Try Power Apps for free
- Watch videos on the Power Apps channel on YouTube

Power Apps for app makers and creators

> **Note**

Try the new Power Apps experience—an AI-native solution designed to simplify app creation. Effortlessly automate workflows, connect data sources, and speed up development, all without needing advanced coding skills. For more information, see Overview of the New Power Apps experience (preview).

With Power Apps, you can create two types of apps: **canvas** and **model-driven**. For more information, see Overview of creating apps in Power Apps.

To create an app, start with make.powerapps.com.

- **Plans** in Power Apps is a copilot-first development tool that lets you quickly create comprehensive business solutions. Describe your business use case in natural language and add relevant images, like business process flows or screenshots of legacy apps. For more information, see Overview of plans.
- **Power Apps Studio** is the app designer you use to build canvas apps. The app designer makes creating apps feel more like building a slide deck in Microsoft PowerPoint. For more information, see Generate an app from data.
- **App designer** for model-driven apps lets you define the sitemap and add components to build a model-driven app. For more information, see Design model-driven apps using app designer.

Ready to convert your ideas into an app? Start here: Planning a Power Apps project.

Power Apps for app users

Run apps that you created, or that someone else created and shared with you, in a browser or on mobile devices (phone or tablet). For more information, see:

- Find and run apps
- Run apps on Power Apps mobile

Power Apps for admins

Power Apps administrators can use the **Power Platform admin center** (admin.powerplatform.microsoft.com) to create and manage environments, view Dataverse analytics, and get real-time, self-help recommendations and support for Power Apps and Power Automate. For more information, see Administer Power Platform.

Power Apps for developers

Developers are app makers who write code to extend business app creation and customization. Use code to create data and metadata, apply server-side logic with Azure functions, plug-ins, and workflow extensions, apply client-side logic with JavaScript, integrate with external data by using virtual tables and webhooks, build custom connectors, and embed apps into your website experiences to create integrated solutions. For more information, see:

- Dataverse developer
- Model-driven apps developer
- Canvas apps developer

Power Apps and Dynamics 365

Dynamics 365 apps (such as Dynamics 365 Sales, Dynamics 365 Customer Service, and Dynamics 365 Marketing) also use the underlying Dataverse platform that Power Apps uses to store and secure data. This feature enables you to build apps by using Power Apps and Dataverse directly against your core business data already used within Dynamics 365, without the need for integration. For more information, see Dynamics 365 and Dataverse.

Try Power Apps for free

You can build Power Apps for free. Sign in to Power Apps. For more information, see Sign in to Power Apps for the first time. Initially, you have access to the default environment. You can build in your own developer environment with Dataverse by signing up for the Power Apps Developer Plan. For more information, see About the Power Apps Developer Plan.

You need a license to play the apps you make with Power Apps. You can play Power Apps for free by signing up for a 30-day trial.

Request a Power Apps license

If you're a Power Apps user and want to run apps, request a Power Apps license while running the app. For more information, see Request a Power Apps license.

Purchase Power Apps

To purchase Power Apps, go to Purchase Power Apps for detailed information.

Power Apps US Government plans

Power Apps US Government offers several plans designed for US government organizations to address the unique and evolving requirements of the United States public sector. The Power Apps GCC environment complies with federal requirements for cloud services, including FedRAMP High, DoD DISA IL2, and requirements for criminal justice systems (CJI data types). For more information, see Power Apps US Government.

Last updated on 11/18/2025

Explore Power Apps for free for 30 days

Get instant access to all Power Apps capabilities with a free 30-day trial. Whether you're new to Power Apps or want to explore premium features, this trial unlocks everything you need to build powerful business applications.

If you don't have a license for Power Apps, the trial plan gives temporary access to these activities:

- Extend the capabilities of Office 365 (SharePoint Online, Teams, Excel, and more)
- Create and run canvas apps that connect to Microsoft Dataverse and over 200 other data sources, including premium connectors and on-premises data.
- Create and run model-driven apps
- Automate workflows with Power Automate
- Create and manage environments and Dataverse databases

If you have a Power Apps license or a license through Office 365, you already have access to some of these features. However, the trial license temporarily expands your access to include all features in the previous list. Learn what capabilities each type of license offers on the pricing page.

> **Note**
>
> - If you're an administrator, see Purchase Power Apps for your organization or Power Apps in your organization Q&A.
> - Starting January 2020, self-service purchase, subscription, and license management capabilities for Power Platform products (Power BI, Power Apps, and Power Automate) are available for commercial cloud customers in the United States. For more information, including steps to enable or disable self-service purchasing in your organization, see the Self-service purchase FAQ.

Need an account?

If you don't have a work or school account, create a free account with Microsoft Entra and become a low-code developer with Power Apps. To keep building apps after the trial period, opt in to the free Power Apps Developer Plan.

Ready to get started?

1. **Start your free trial now →** - Go to PowerApps.com and select **Start free**.

2. Enter your email. If your email isn't a work or school account, you're prompted to create a free Microsoft Entra account by using this sign-up experience.

If you want to evaluate Power Apps integrations with Microsoft 365, sign up for an Office 365 trial account.

> **Note**
>
> If you don't have an organization account or aren't part of a tenant or organization account, you can establish a developer's account through the Microsoft Developer Program.

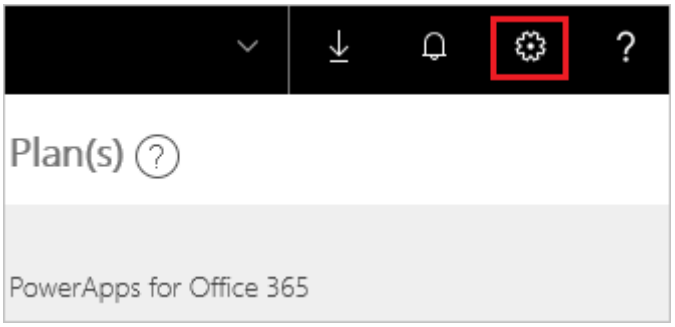
Identify your current license

Compare license options: Learn what capabilities each type of license offers on the pricing page to choose the best plan for your needs.

1. Sign in to Power Apps using your work or school credentials.

   > **Important**
   >
   > You can't sign in by using a personal email address, such as one that ends in outlook.com, hotmail.com, or gmail.com. For more information, see What email address can I use? later in this article.

2. Select the gear icon in the upper-right corner, and then select **Plan(s)**.



Upgrade an existing license

To start a free 30-day trial of Power Apps per user plan from an existing license, sign in to Power Apps. Then, try any feature that requires a premium Power Apps license. For example, select the gear icon near the upper-right corner, select **Admin center**, and select **New**

environment near the upper-right corner. Follow the prompts to complete the sign-up process.

Get a license from scratch

Whether or not you have a license for Power Apps, start a free 30-day trial per user plan by following these steps:

1. Open the Power Apps site, and then select **Try free**.

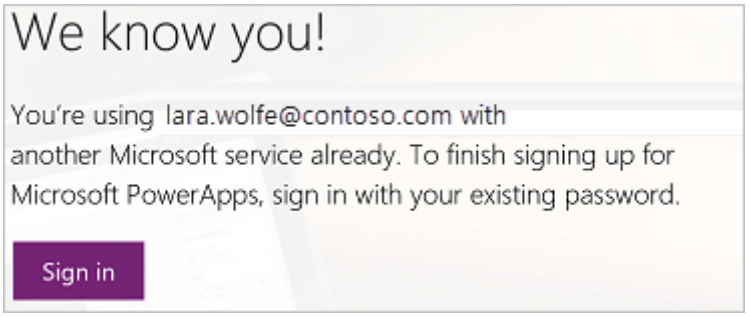
   If you're using a phone, select the menu in the top-right corner, and then select **Try free**.

2. In the middle of the screen, select **Pricing**, and then select **Start free trial**.

3. In the dialog box, enter your work or school email address, and then select **Submit**.

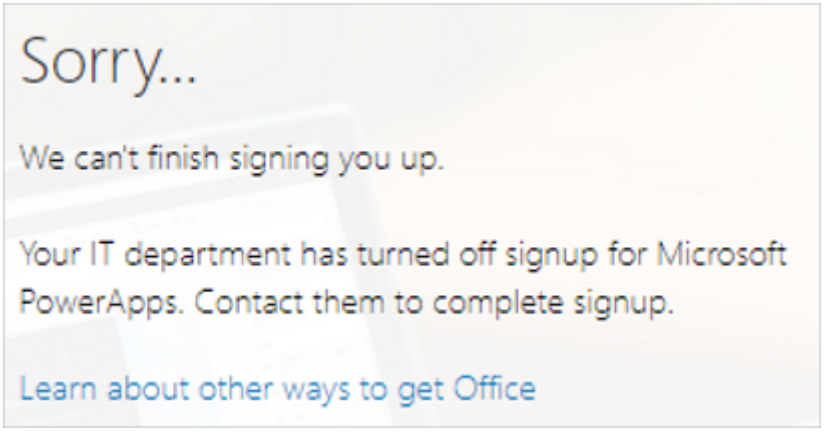
   > **Important**
   >
   > For more information, see What email address can I use? later in this article.

4. If a dialog box indicates that Power Apps recognizes your organizational credentials, follow the prompts to finish signing in.

   Otherwise, follow the prompts to check your email, verify your email address, provide more information if necessary, and then select **Start**.

   You see the following message if your admin disables free trials.



> **Note**
>
> For more information, including the steps to enable or disable trial licenses in your organization, see Block trial licenses commands.

FAQ

What email address can I use?

You can use a work or school email address, backed by Microsoft Entra ID, to sign up for a trial license. If you use a different kind of address, you might experience one of the symptoms in this table.

Expand table

Symptom / error message	Cause and workaround

Personal email addresses (for example, `@gmail.com` email address)
 	
You receive a message like the following during signup:
 	
`You entered a personal email address: Please enter your work email address so we can securely store your company's data.`
 	
or
 	
`That looks like a personal email address. Enter your work address so we can connect you with others in your`
	Power Apps doesn't support email addresses provided by consumer email services or telecommunications providers.
	
	To complete signup, try again using an email address assigned by your work or school.

Symptom / error message	Cause and workaround
`company. And don’t worry. We won’t share your address with anyone.`	

.gov or .mil addresses
 	
You receive a message like the following during signup:
 	
`Power Apps unavailable: Power Apps is not available for users with .gov or .mil email addresses at this time. Use another work email address or check back later.`
 	
or
 	
`We can't finish signing you up. It looks like Microsoft Power Apps isn't currently available for your work or school.`
	Power Apps doesn't support .gov or .mil addresses at this time.

Email address is not an Office 365 ID You receive a message like the following during signup: `We can't find you at contoso.com. Do you use a different ID at work or school? Try signing in with that, and if it doesn't work, contact your IT department.`	Your organization signs in to Office 365 and other Microsoft services with IDs other than email addresses. For example, your email address might be Nancy.Smith@contoso.com, but your ID is nancys@contoso.com. To complete signup, use the ID that your organization has assigned to you for signing in to Office 365 or other Microsoft services. If you don't know what this is, contact your IT administrator.

Does the cloud environment I use Power Apps in allow the use of trial licenses?

Power Apps trial licenses are available only in the Azure commercial cloud. They aren't available in Azure Government or Microsoft Azure China clouds.

What happens when my trial expires?

You're prompted to request an extension of the trial or purchase a plan 30 days after the trial starts. You can extend the trial two more times (30 days each)—with a maximum trial period of 90 days. You can find details about all plans on the pricing page.

If you don't extend the trial or purchase a plan but you still have another kind of license, you can use all the features of Power Apps that your other license provides. Any data in Dataverse remains unchanged, and any app or flow that uses Dataverse continues to run if your license supports them. If you try to use premium Power Apps features but your existing license doesn't support them (for example, modify a schema or entities in Dataverse), you are prompted to purchase a plan.

More questions?

Visit the Power Apps community.

Last updated on 01/13/2026

Request a Power Apps license for yourself

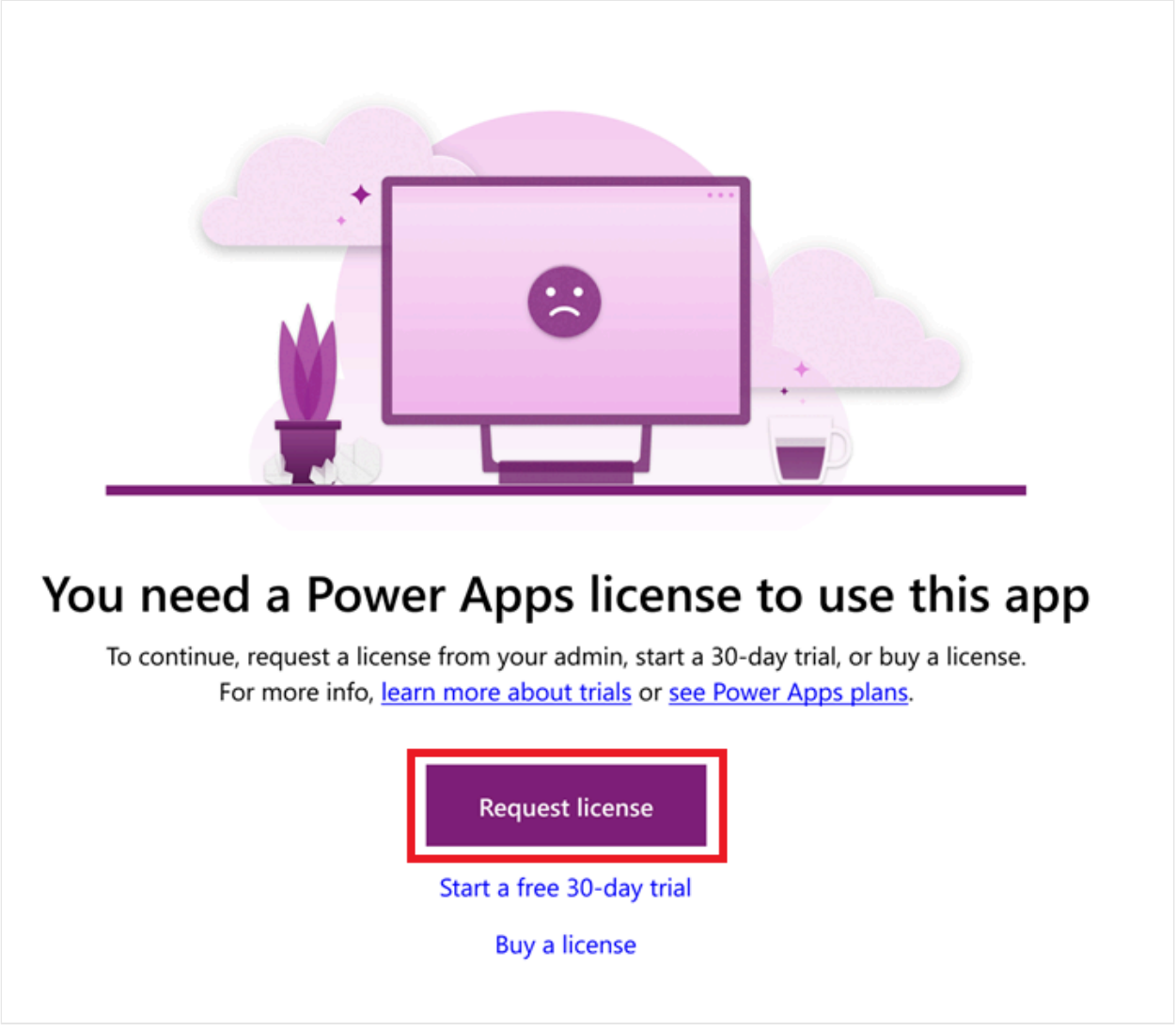
Article • 05/08/2023

You can request a Power Apps license from your admin directly using Power Apps.

Request a license

If you don't have a license and are trying to open apps made with Power Apps, you'll see the options in the following image. Select **Request license** to submit a license request to your admin.

> **Note**
>
> The options in the following image may differ depending on whether you've exhausted any Power Apps trials.



Request a license through a trial

If you're already in an active Power Apps trial, you can request a license by selecting **Request license** near the top of the page.



Wait for a response from your admin

After you request a license, you'll see a banner at the top of the page that confirms that your request was sent, and that it might take a few days for your admin to respond to the request.

If you're using a Power Apps trial, you can continue to use the trial while you wait for your request to be approved.

After your admin has reviewed your request, you'll receive an email notification indicating whether the request was approved.

Feedback

Was this page helpful? Yes No

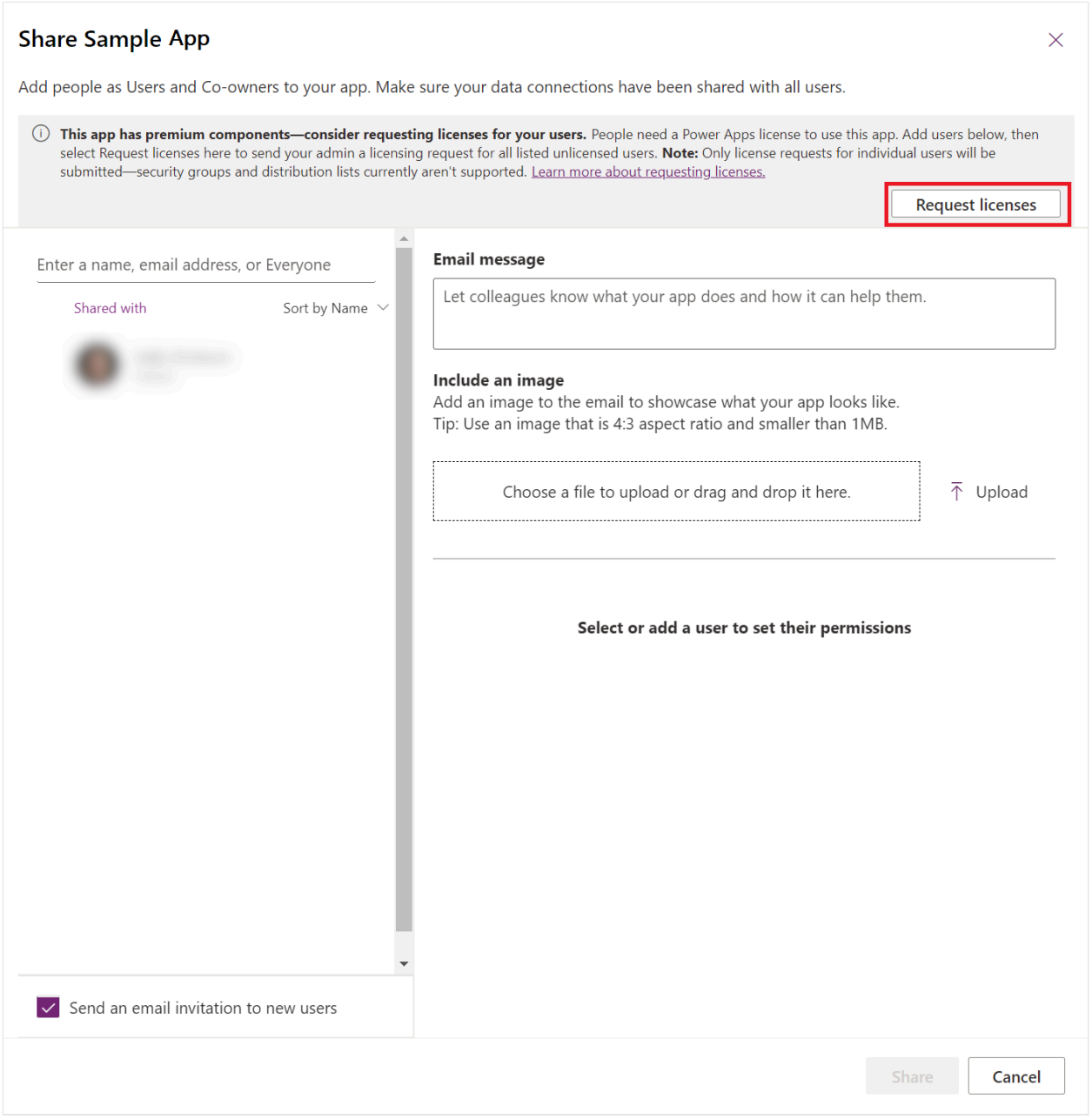
Provide product feedback

Request Power Apps licenses for your app users

Article • 06/30/2023

As a Power Apps maker, you can create an app that requires a license for use. For example, if you create an app with premium components or connectors, users must have an appropriate Power Apps license to use it.

When you share a canvas app or model-driven app that contains premium components, select **Request licenses**. The request is submitted to your admin for approval. You can't request licenses for security groups or distribution lists.



After your admin has reviewed your request, you'll receive an email notification indicating whether the request was approved or denied.

Feedback

Was this page helpful? Yes No

Provide product feedback

Overview of creating apps in Power Apps

Power Apps is a high-productivity development platform for business apps. It has these major components:

- Plans
- Copilot
- Canvas apps
- Model-driven apps
- Microsoft Dataverse

Let's learn more about each of these components.

> **Note**
>
> Try the new Power Apps experience—an AI-native solution designed to simplify app creation. Effortlessly automate workflows, connect data sources, and speed up development, all without needing advanced coding skills. For more information, see Overview of the New Power Apps experience (preview)

Plans

Plans in Power Apps is a copilot-first development tool that helps you rapidly build end-to-end business solutions. Simply describe your business scenario in natural language and include relevant visuals, such as process flows or screenshots of existing systems. Plans then generates a complete Power Platform solution tailored to your requirements, including Microsoft Dataverse tables, canvas apps, model-driven apps, Power Pages sites, Power Automate flows, and Copilot Studio agents.

Create apps using Copilot

Copilot in Power Apps lets you to quickly create apps by simply describing your business requirements in natural language. Copilot automatically generates both the app and its data model—no coding needed. Users can interact with your app using everyday language, making the experience intuitive and engaging. For more information, see Copilot and AI features in Power Apps

Canvas apps

Canvas apps start with your user experience. You craft a highly tailored interface with the power of a blank canvas and connect it to your choice of more than 200 data sources. You can build canvas apps for web, mobile, and tablet applications.

Canvas apps give you the flexibility to arrange the user experience and interface the way you want it. Let your creativity and business sense guide how you want your apps to look and feel.

You can start to build your app from Microsoft tools where your data lives, such as:

- From a SharePoint list
- From a Power BI dashboard

Creating a canvas app is easy. With Power Apps, you can find or create your app in several ways:

- From a Dataverse source
- From data in a SharePoint List
- From Excel data
- From a blank canvas
- Via AppSource

In addition to creating apps in Power Apps, you can also create apps directly within Microsoft Teams by using Power Apps. For more information, see Create apps in Microsoft Teams by using Power Apps.

Model-driven apps

Model-driven apps start with your data model. You build up from the shape of your core business data and processes in the Dataverse to model forms, views, and other components. Model-driven apps automatically generate a great UI that's responsive across devices.

When you create a model-driven app, you can use all the power of the Dataverse to rapidly configure your forms, business rules, and process flows. You create a model-driven app from the Power Apps site.

As part of the custom page public preview, you can add the new page type based on canvas into a new or existing model-driven app. This addition brings the power and flexibility of a canvas experience within the model-driven app. Custom pages allow

- Full control over page layout
- Out of box controls and custom components
- Power Apps connectors
- Low-code authoring

Getting started with model-driven apps is simple. You can begin with these topics:

- Create an app
- Create and design forms
- Create or edit views
- Create or edit a system chart
- Create or edit dashboards
- Add security
- Add business logic
- Converging model and canvas using custom page (preview)

Dataverse

Dataverse is the data platform that comes with Power Apps and allows you to store and model business data. It's the platform on which Dynamics 365 apps (such as Dynamics 365 Sales, Customer Service, Field Service, Marketing, and Project Service Automation) are built. If you're a Dynamics 365 customer, your data is already in Dataverse.

Dataverse allows you to securely store and manage data within a set of standard and custom tables. You can add columns to those tables when you need them.

Start working with Dataverse by reading these topics:

- Create a custom table
- Manage columns
- Create custom choices
- Create a business rule

Planning a Power Apps app or project

It's easy to begin building your first app. Simply sign in to Power Apps. For more information, see Sign in to Power Apps for the first time. If you're interested in exploring even more functionality, such as creating environments, we have a 30-day trial or Developer Plan.

If you're new to Power Apps and want to learn about how to convert your ideas into a fully working solution, start with Planning a Power Apps project.

Last updated on 11/18/2025

Sign in to Power Apps

Sign in to Power Apps at https://make.powerapps.com to create, edit, or play an app.

This article shows you how to access Power Apps for the first time, choose the right environment, and navigate different authentication options including Azure B2B and Microsoft Account sign-in.

To create canvas apps, you need the environment maker predefined security role.

If you don't have a work or school account, you can sign up for a trial account. For information, see Explore Power Apps for free for 30 days.

What you'll learn:

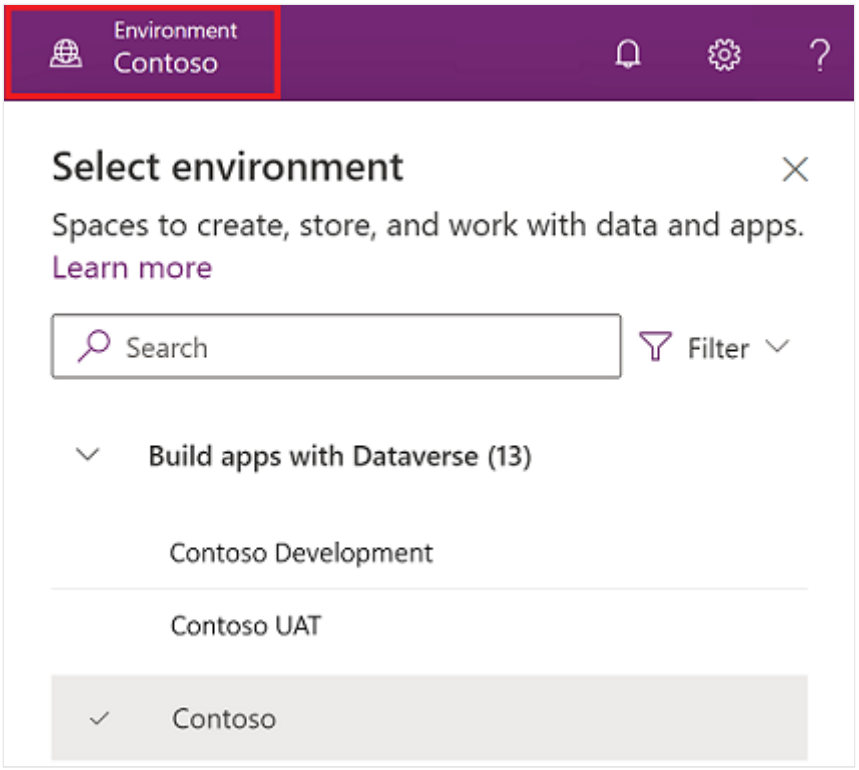
- How to choose the right environment for your work
- Different sign-in options including Azure B2B and Microsoft Account
- How to access environments and create your first app

For more information about the Power Apps home page, see Get started with Power Apps.

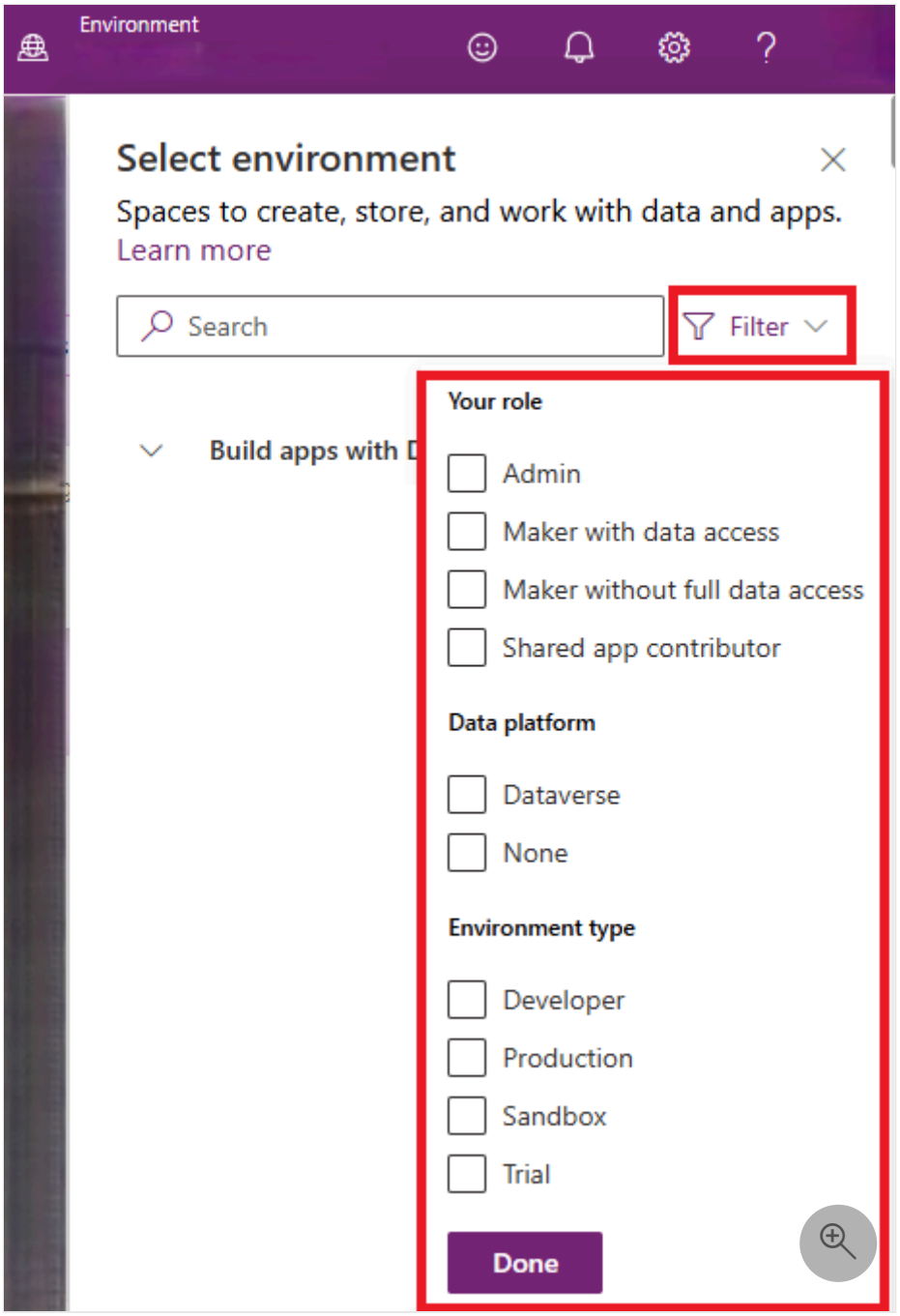
Choose an environment

When you create an app, a flow, a data connection, or a table in Microsoft Dataverse, most work in Power Apps happens in a specific environment. Environments set boundaries between types of work. For example, an organization can use separate environments for different departments. Many organizations use environments to separate apps in development from those ready for use. You might have access to multiple environments or just one. If you have the right permissions, you can create your own environment.

To check which environment you're in, find the environment switcher near the right side of the header.



With the environment selector, environments are grouped into two types: **Build apps with Dataverse** and **Other environments**. Select **Filter** to filter the list of environments by your role, data platform (Dataverse or none), and environment type, like production or sandbox.



Environments where you have system administrator or system customizer security role membership appear under **Build apps with Dataverse**. The **Other environments** list shows environments where you have only environment maker or editing privileges to at least one canvas app in the environment.

> **Tip**
>
> Hover over an environment in the list to see its details.

Filter environments by role

Expand table

Filter role	Power Platform role or description
Admin	System administrator Environment admin
Maker with data access	System administrator System customizer
Maker without full data access	Environment maker (with or without Dataverse)
Shared app contributor	User without a maker-level security role assigned but with edit permission to at least one canvas app in the environment

Important

- To see the environment list in the environment switcher in Power Apps, you need the Environment Maker, System Customizer, or System Administrator security role in the environment. For information about predefined security roles, see Predefined security roles in the Microsoft Power Platform admin guide.
- Make sure you're in the right environment before you create an app, a flow, or a similar component. You can't easily move components from one environment to another.

Note

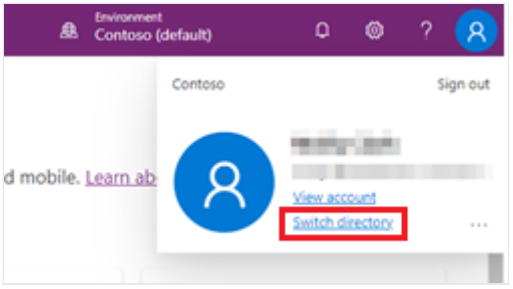
- Every member in an organization can use the default environment. Like any environment, users can see apps where they have enough privileges to use an app.
- All users with the Environment Maker security role in an environment can see all model-driven apps in that environment, including the default environment. For more information, see Model-driven app privileges to view and access apps.
- When you create an app in one environment, you can't see it from another environment. People who want to run your app need access to the environment where you created it.

For more information about environments, see Environments overview.

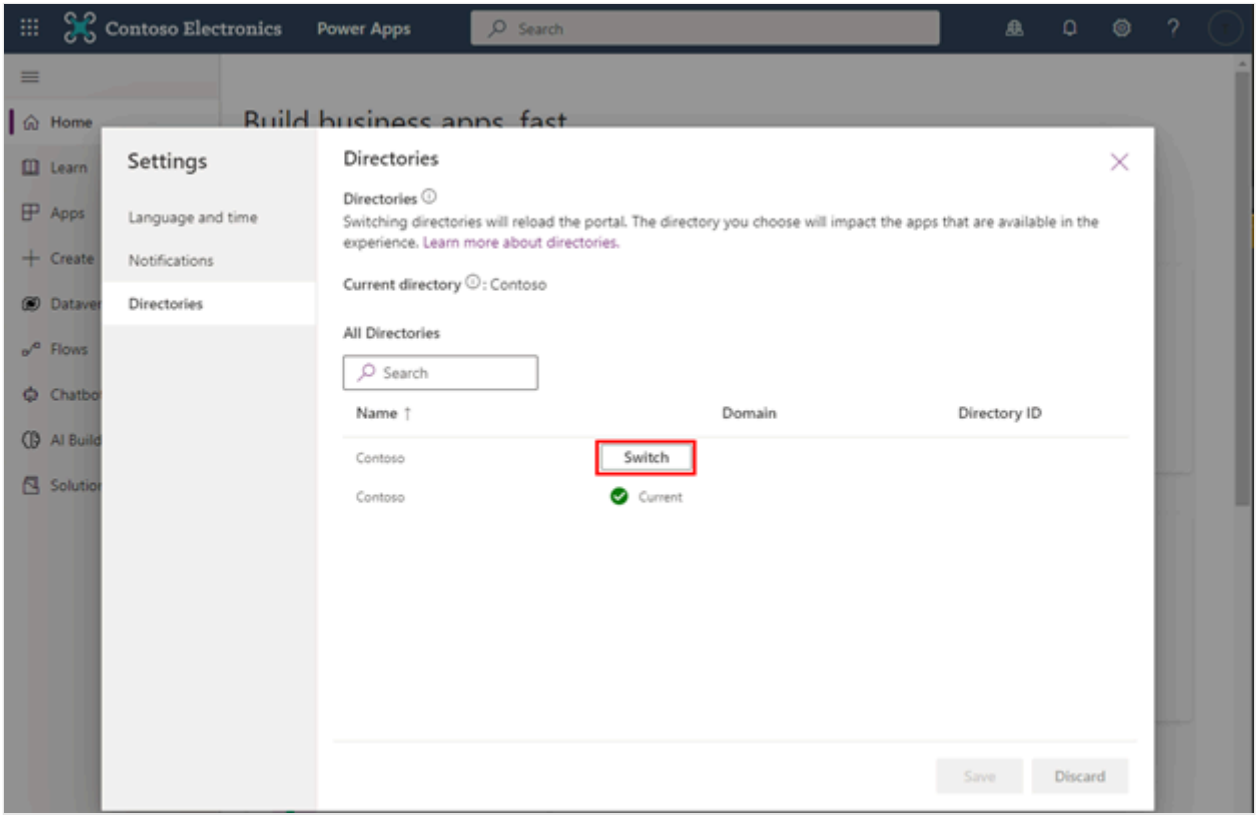
Sign in by using Azure B2B collaboration

> **Note**
>
> - A **resource tenant** is the Microsoft Entra tenant where an app is expected to exist, and where the user is expected to create and edit the app. For Azure B2B makers, this tenant is different from the tenant their account resides.
> - A **home tenant** is where the user's account resides and authenticates against.
> - To create and edit apps in a resource tenant, an admin must follow these steps to give Azure B2B users the prerequisite privileges to build apps.

When a user signs into Power Apps, they sign into their **home tenant** - the Microsoft Entra tenant where their credentials are provisioned. After signing in, the user can change the directory they intend to build apps in by selecting the **Switch directory** link as shown here:



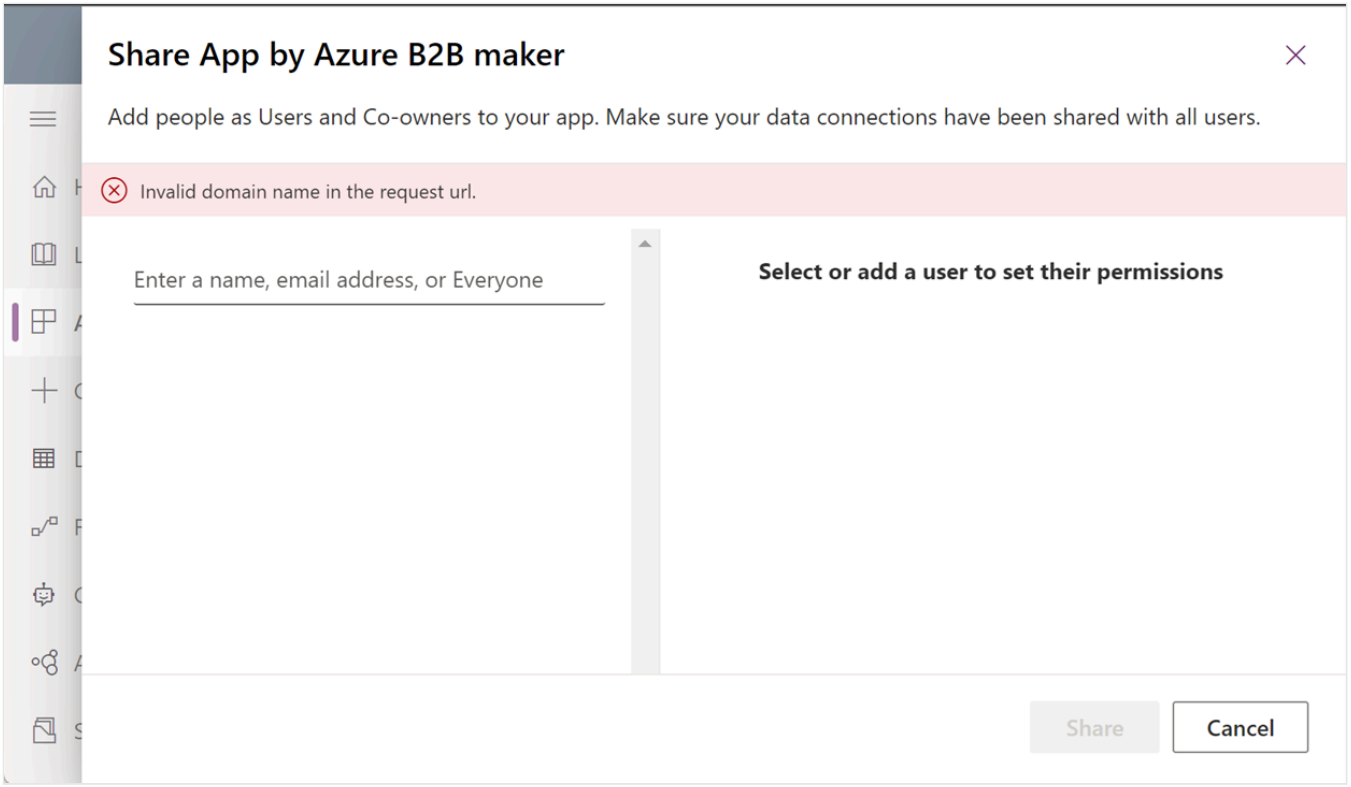
The switch directory link opens Power Apps settings with a **Directories** tab that lists all the Microsoft Entra tenants where the user is a member or an Azure B2B guest. Selecting **Switch** signs out of the current tenant and then signs into the selected tenant in Power Apps.



Frequently asked questions

As an Azure B2B maker, why can't I share apps?

The Power Apps sharing experience requires users to have permissions in the tenant when the Azure Microsoft Entra ID external collaboration settings lets guest user access to "(most inclusive)". Azure B2B makers without enough privileges see the following error dialog when sharing.



Can I work in Power Apps in both my home tenant and a resource tenant?

- Yes, but only in separate browser sessions. For example, Microsoft Edge can be open in its standard mode, and you can start a separate session in a new **InPrivate** window.
- If you open multiple tabs in the same browser session, you can only access resources in the most recently signed-in tenant.

Can I sign in to Power Apps, in a resource tenant with GDAP?

No, https://make.powerapps.com doesn't recognize Granular delegated admin privileges (GDAP), but the Power Platform admin center does.

Sign in by using Microsoft Account (preview)

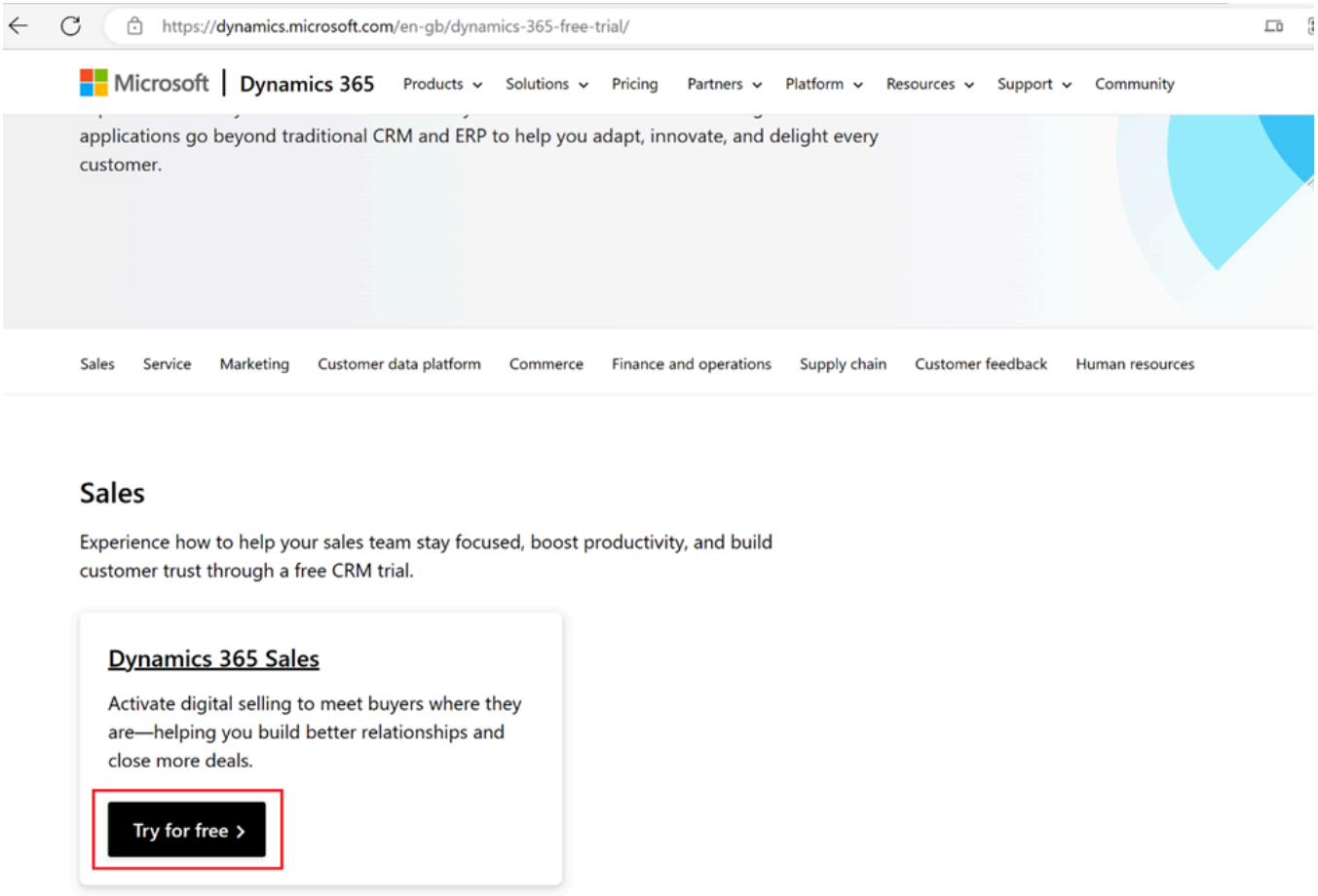
Anyone can build Power Apps, even if they don't have a work or school account. Sign up for a Dynamics 365 Sales trial by using a Microsoft Account and then use that account to sign in to Power Apps.

During the 30-day trial period, you can make and play with as many apps as you like. You can also build apps with Dataverse or other premium connectors. However, if you plan to run these apps beyond the trial period or share them with others, you need to create a (free) Microsoft Entra account.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Where can I get a license with Power Apps use rights for my Microsoft Account?

The Dynamics 365 Sales free trial includes Power Apps use rights, along with other Dynamics and Power Platform use rights. You can sign up here: Dynamics 365 free trial.



> **Note**
>
> Not all Power Apps free licenses support Microsoft Account sign-up. The list of free licenses that allow free sign-up is expected to grow over time.

How do I create model-driven apps and use Dataverse?

Select or create an environment with Dataverse. You can create Developer environments with Dataverse for free.

1. Sign in to Power Apps.
2. Select the environment picker.
3. In the **Need your own environment?** prompt, select **Try it now** to create an environment with Dataverse.

How do I access Power Platform admin center?

Power Platform admin center (https://admin.powerplatform.microsoft.com) doesn't support Microsoft Account sign-in. However, your Microsoft Account is associated with a Microsoft Entra tenant and you can provision a Microsoft Entra based identity to use in the Power Platform admin center.

1. Follow the steps under **How do I create a Microsoft Entra identity with the same administrative privileges as my Microsoft Account?**
2. Using the newly created Microsoft Entra identity, sign in to https://admin.powerplatform.microsoft.com.

How do I create a Microsoft Entra identity with the same administrative privileges as my Microsoft Account?

1. Sign in to https://portal.azure.com.
2. Search for **Microsoft Entra ID**.
3. Add a new user.
4. Associate or add an Azure subscription to your Microsoft Entra tenant. For administrative purposes, assign the same admin role as your Microsoft Account has in Microsoft Entra ID, such as Power Platform or Dynamics 365 admin.

The free license assigned to my Microsoft Account expired, how can I access Power Apps?

If your Microsoft Account is associated with a Microsoft Entra tenant, you can sign in to Power Apps and edit your apps. However, without an active license you can't play the apps.

Today, Microsoft accounts are bound to limited trial periods. You can't sign up for the same free license twice with the same account. Also, Microsoft accounts can't purchase Power Apps plans.

You might proceed with building and running apps by creating a Microsoft Entra identity in the same Microsoft Entra tenant your Microsoft Account is associated with and sign up for a Power Apps Developer plan by using that identity. With your Microsoft Account, you need to share your apps with your Microsoft Entra identity to proceed with editing and playing them.

To create a Microsoft Entra identity with administrative rights, follow the steps under "How do I create a Microsoft Entra identity with the same administrative privileges as my Microsoft Account?".

Can I use my Microsoft Account with the Power Platform PowerShell cmdlets?

No. In the meantime, follow the steps described in the previous questions. Add a new Microsoft Entra identity to the Microsoft Entra tenant associated with your Microsoft Account to use the Power Platform PowerShell cmdlets.

Can I share my apps with other users?

Yes, but you need to add users to the Microsoft Entra tenant associated with your Microsoft Account, assign a license to these users, and then share your app with these users. As an admin, you can assign free licenses to users but these licenses expire.

1. To create a Microsoft Entra identity with administrative rights, follow the steps under **How do I create a Microsoft Entra identity with the same administrative privileges as my Microsoft Account?**.
2. Add each user you want to share an app with to Microsoft Entra ID.
   a. Sign in to https://portal.azure.com.
   b. Search for **Microsoft Entra ID**.
   c. Add a new user or Add a guest user. You can add users that have either a Microsoft Account or Microsoft Entra identity.
3. With your Microsoft Entra admin identity, either purchase Power Apps plans. As an admin, you receive 25 Power Apps per user trials. All of these trials expire.
4. With your Microsoft Entra admin identity, assign licenses to the users you added to your Microsoft Entra tenant.
5. Sign in to Power Apps to proceed with sharing your apps to users.

Can I run my apps in Power Apps mobile?

Yes. You must launch an app from an app link. Without launching an app directly, Power Apps mobile doesn't support Microsoft Account sign in out of the context of launching an app.

You can get the app link for an app by going to Power Apps > **Apps** > (select an app) > **Details** > See the **Web link** property.

Next steps

Build an app

What is Microsoft Dataverse?

Last updated on 01/21/2026

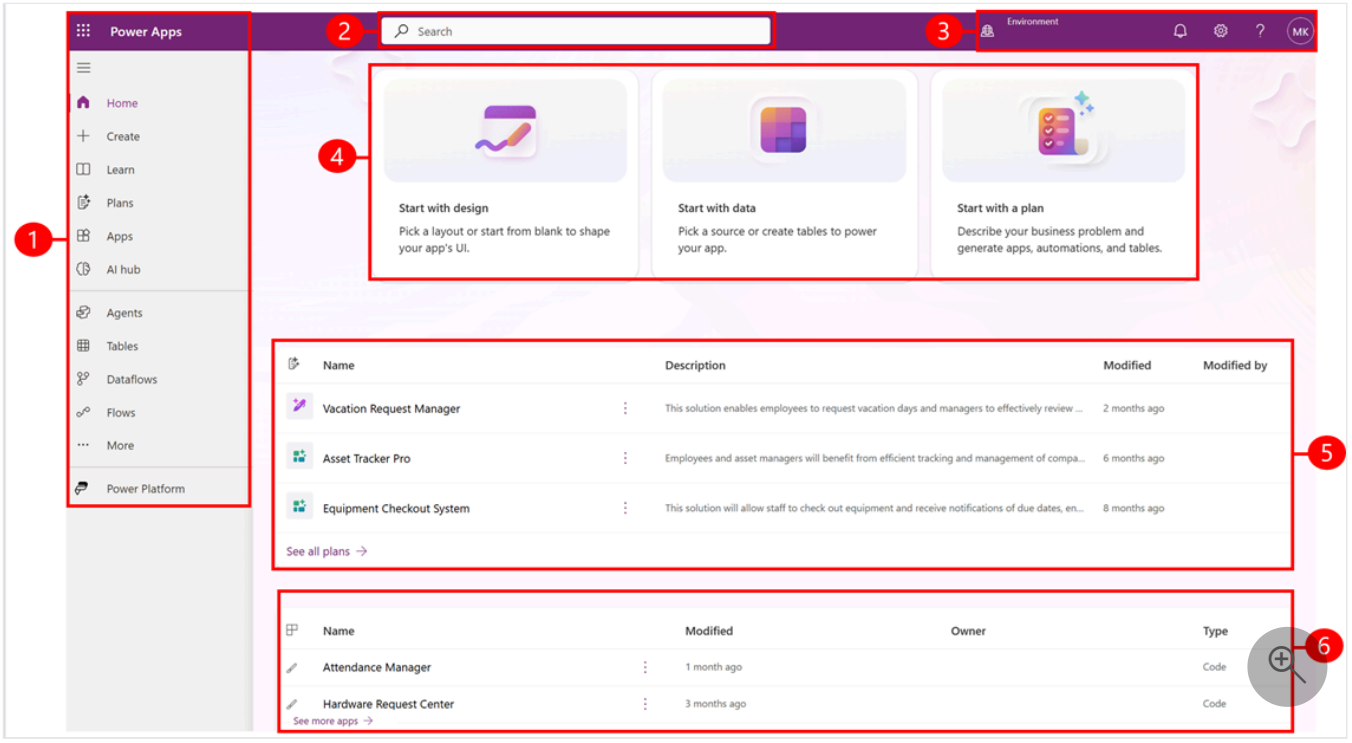
Get started with Power Apps

Summarize this article for me

Get started with Power Apps to build your first app quickly and efficiently. The Power Apps home page is your starting point for creating apps, managing existing ones, and accessing powerful features that streamline your workflow.

What you'll accomplish: By the end of this article, you'll learn how to navigate the interface, create your first app, and access Copilot AI features to accelerate development.

Select the appropriate tab in the following section to learn more.



Legend:

1. Left navigation pane
2. Search
3. Environment information and settings
4. Build apps
5. View and edit plans
6. View and edit apps

> **Note**
>
> Try the new Power Apps experience—an AI-native solution designed to simplify app creation. Effortlessly automate workflows, connect data sources, and speed up

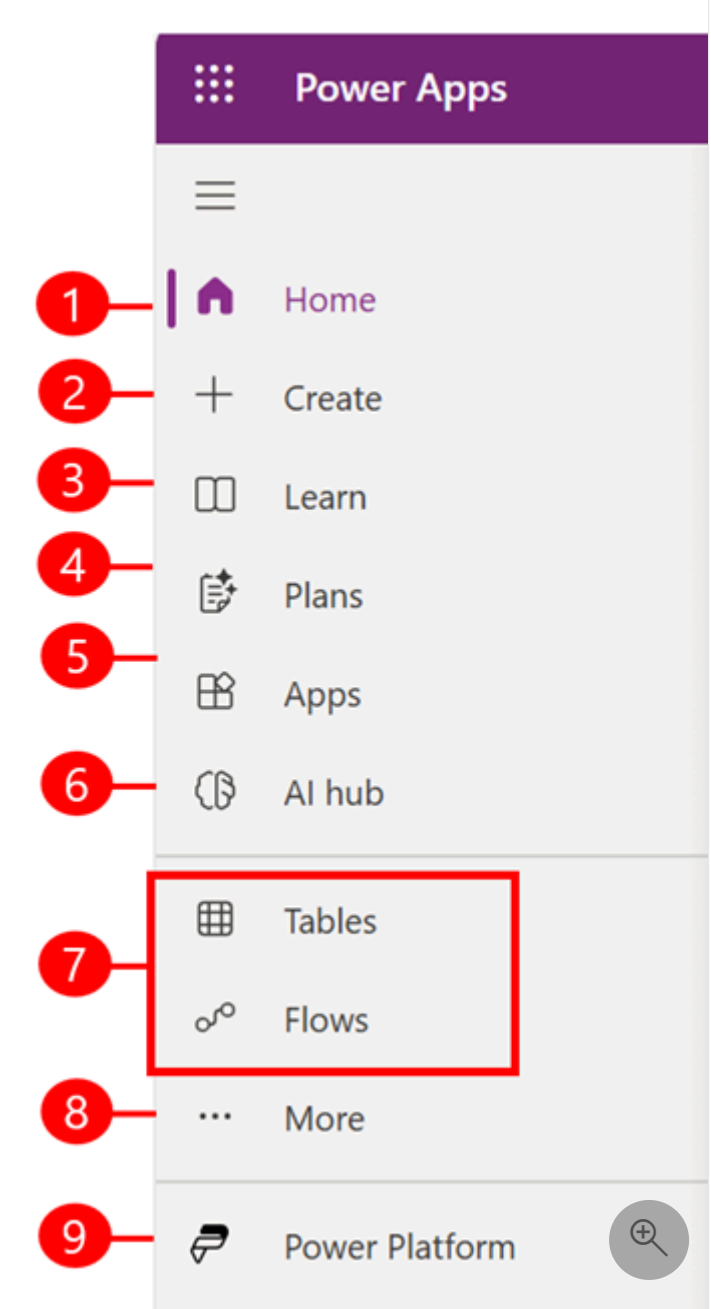
development, all without needing advanced coding skills. For more information, see Overview of the New Power Apps experience (preview).

Quick start options

- **New to Power Apps?** Start with our 5-minute canvas app tutorial
- **Have existing data?** Create an app from SharePoint or Excel
- **Need help choosing?** Review which app type is right for you

1 – Left navigation pane

Your main control center. When you sign in to the Power Apps home page, the left navigation pane gives you quick access to everything you need. Here's what each option does:



Legend:

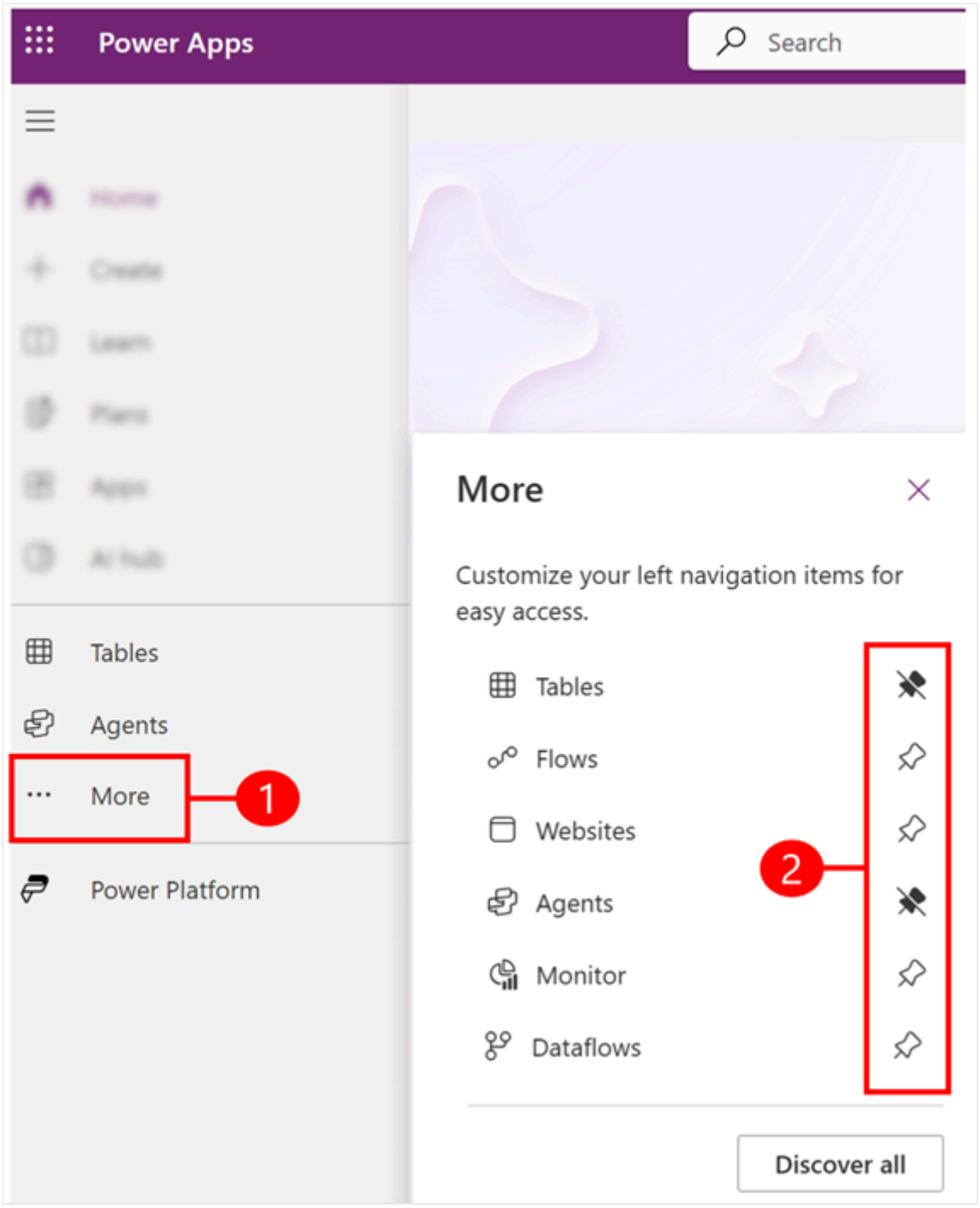
1. **Home**: Takes you to the Power Apps home page.
2. **Create**: Create apps such as canvas apps, model-driven apps, chatbots, and AI models.
3. **Learn**: The learn hub lets you explore documents, training material, get help from the Power Apps community, and other resources that help you to create and build Power Apps.
4. **Plans**: View and edit your plans. For more information, see Overview of Plan designer.
5. **Apps**: If you created an app or someone else created one and shared it with you, you can play or edit it. You can also filter the list of apps based on criteria such as whether you opened it recently.
6. **AI hub**: Create and use AI models that optimize your business processes. For more information, see Overview of AI Builder.
7. **Your most used pages such as Tables, Connections, Flows**: When you first sign in, the top three pages that you use most are pinned to the left navigation pane. Use the **More** option to unpin any of these pages and pin something else.
8. **More**: Pin your most used items to the left navigation such as tables, flows, and more.
9. **Power Platform**: Explore other Power Platform products or service.

Pin and unpin

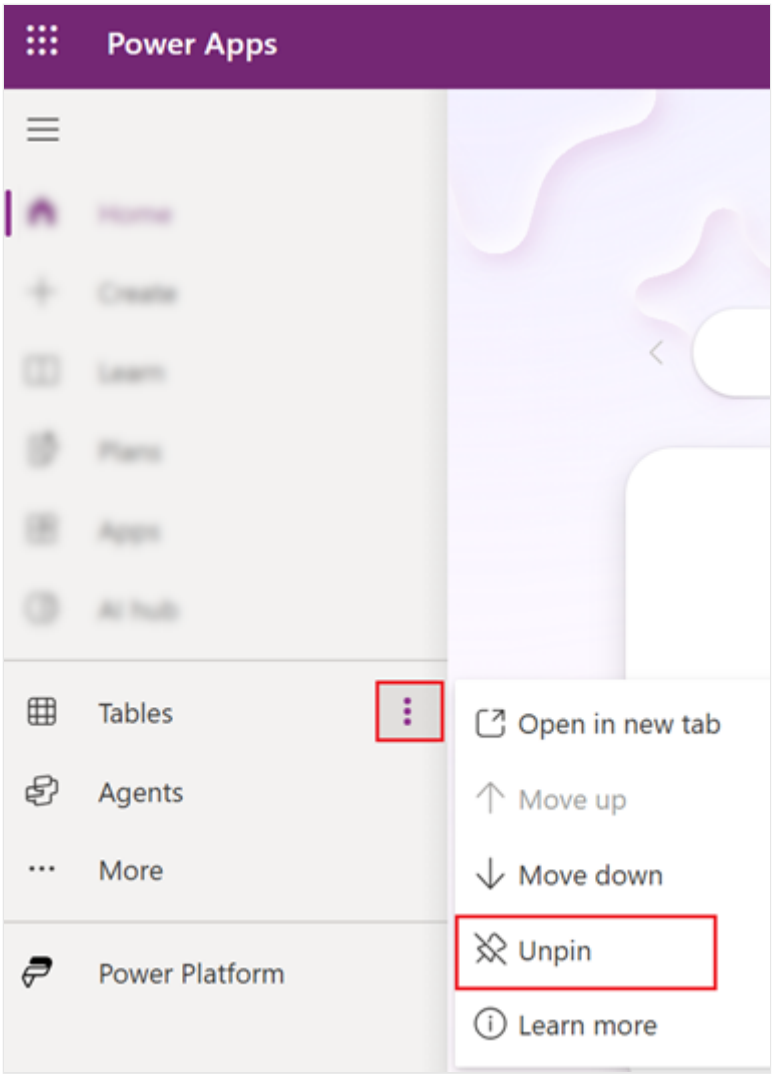
Pin your most used pages in the navigation pane so you can quickly access features that you use frequently. Links to other pages are available through the **More** link. When you pin an item, it appears in the middle section above **More**.

When you sign in, the system automatically pins your most and recently used pages on the left navigation pane. However, you can pin and unpin pages and customize it to your preference.

To pin or unpin a page from the left navigation pane, select **More**, and then select the  pin or  unpin button.



You can also unpin an item by selecting the more button and then select **Unpin**.



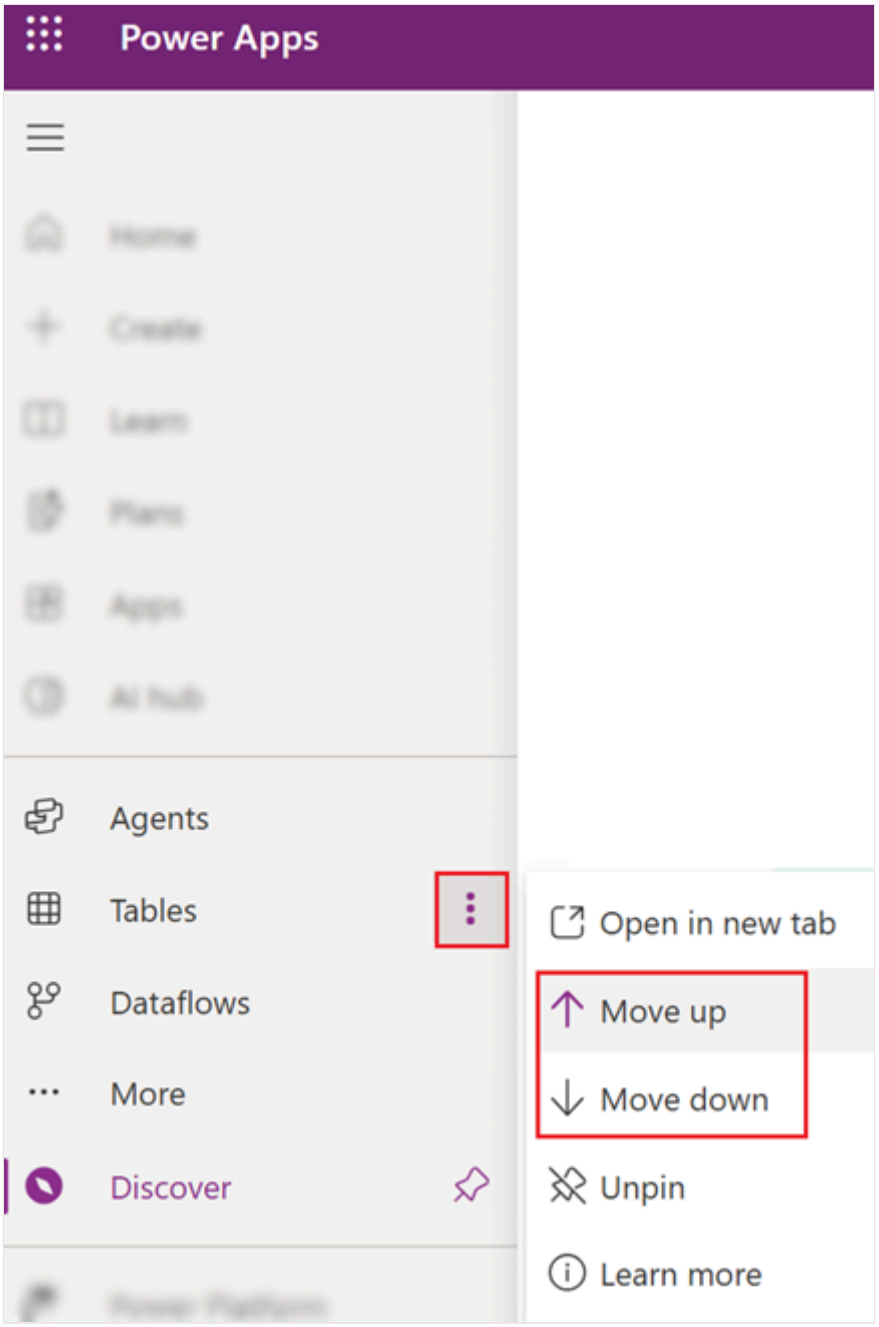
Discover all

To see all your Power Apps, from the left navigation pane, select **More** > **Discover all**. To keep it pinned, select the pin button.

Move up or move down

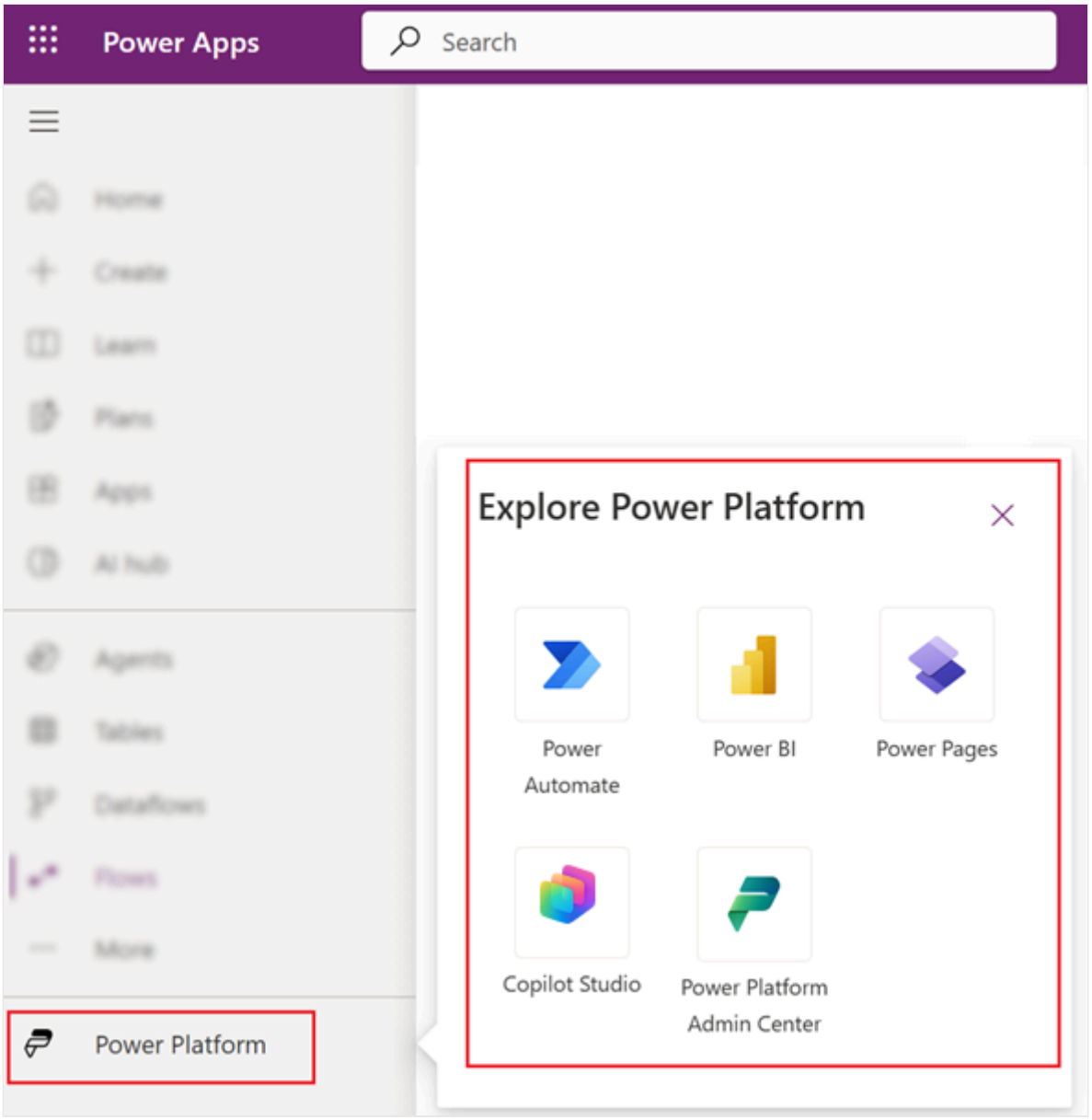
When you have a few pages pinned in the navigation pane, you can start moving them up or down.

To move a page up or down, select the more button next to the page that you want to move, and then select **Move up** or **Move down**.



Power Platform

From the left navigation pane, select **Power Platform** to access Power Platform services such as Power Platform Admin Center, Power BI, and Power Automate.



2 – Search

Find anything fast. The search bar helps you:

- Locate your apps by typing any part of the name
- Discover new app templates
- Access help documentation

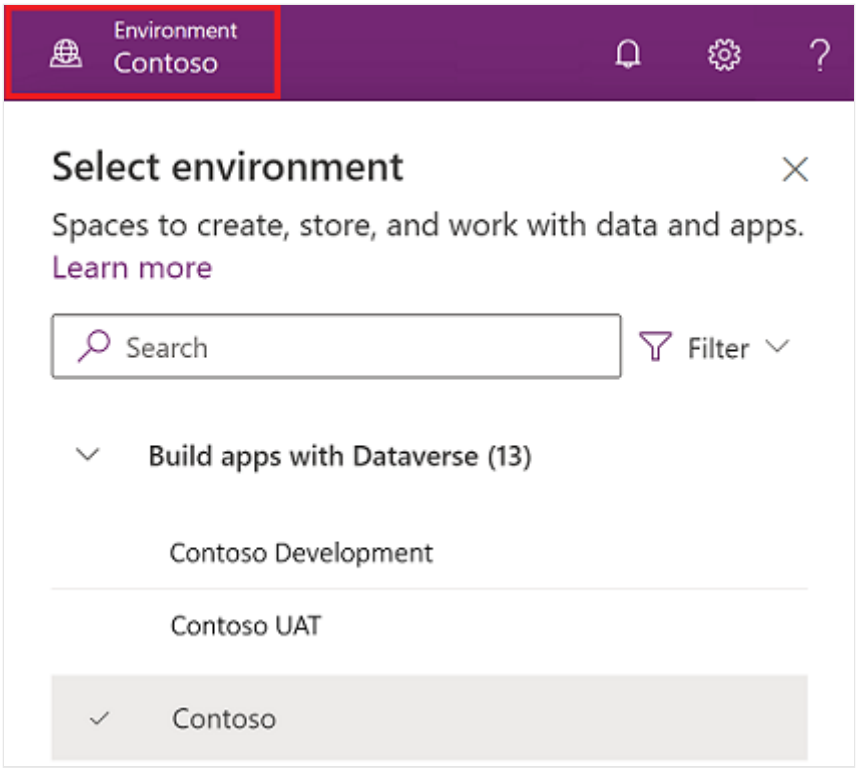
Quick tip: Start typing in the search bar, and then select the banner icon next to your app to open or edit it.

3 – Environment information and settings

View your environment information and settings.

Choose an environment

For best results, start by ensuring that the home page is set to the right environment. For more information, see Choose an environment.



Manage environments by using Power Platform admin center.

Notifications

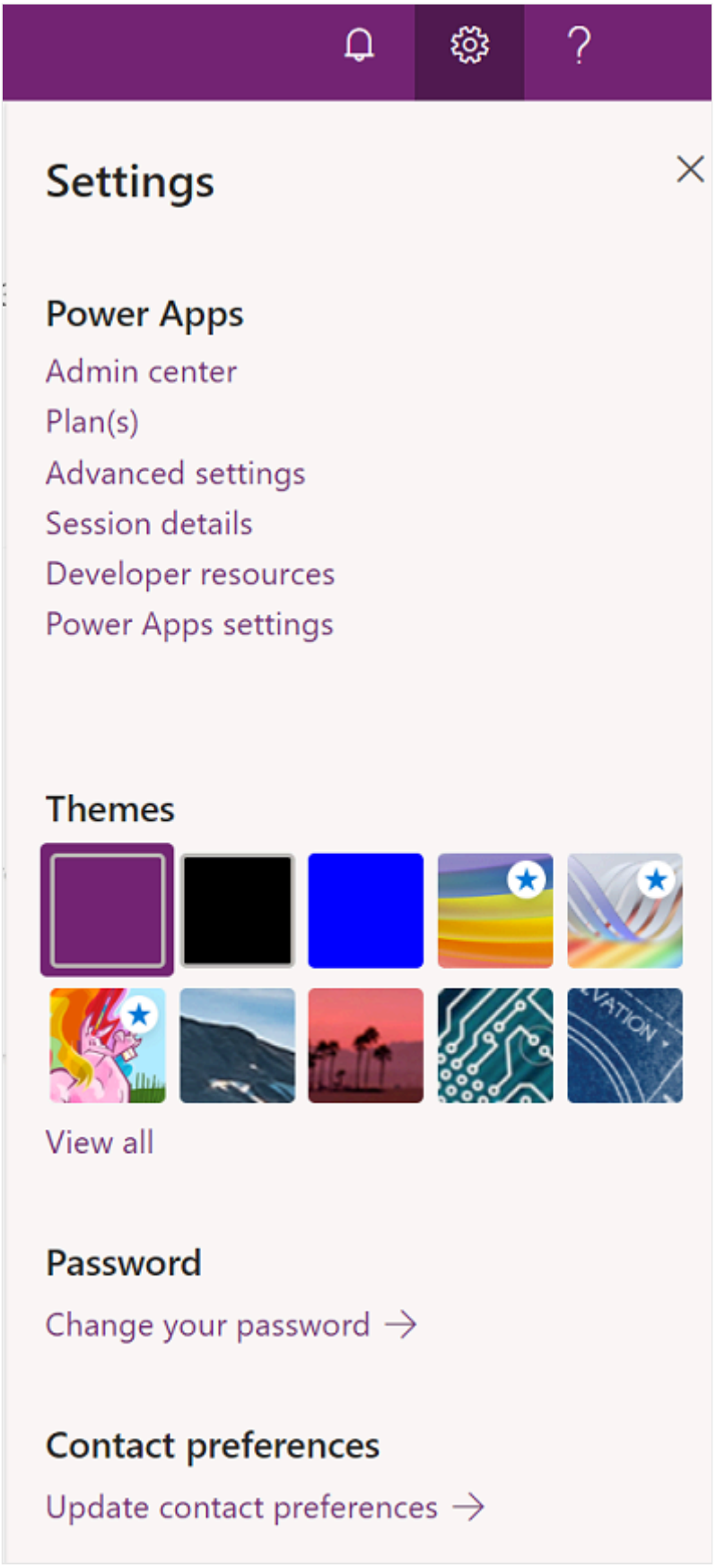
Select the bell icon to see notifications. The notification center stores notifications until you dismiss them or when they expire.

Settings

Select the gear icon to perform tasks such as connecting to data sources, identify your Power Apps licenses, and open the page where you can perform administrative tasks.

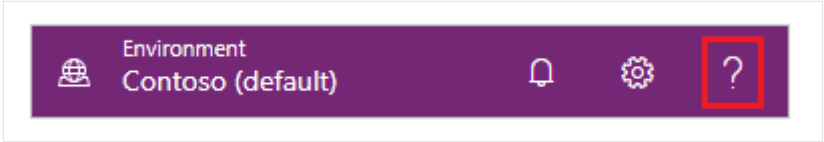
- **Admin center**: Opens another window and goes to the Power Platform admin center.
- **Plans(s)**: View your licenses. For more information, see Licensing overview.
- **Advanced settings**: Manage your admin settings. For more information, see Manage Microsoft Dataverse settings.
- **Session details**: View your Power Apps session details such as session ID, tenant ID, environment ID, and more.

- **Developer resources**: Access developer resources.
- **Power Apps settings**: View or update your language and time settings, notifications, or access directories.
- **Themes**: From the list of themes, select a theme for your organization.
- **Password**: Change your password.
- **Contact preferences**: Update your contact information.



Help

In the header, select the question mark icon to find more information about Power Apps.



You find links to documentation, Power Apps training on Microsoft Learn, the Power Apps Community where you can share information with users in other organizations, and the Power Apps blog where the newest features are announced.

4 – Build apps

Ready to build your first app? Choose the option that best fits your needs:

- **Start with a design →**: Pick a layout or start with blank canvas app and design your app.
- **Start with data →**: Create a new table in Dataverse, or connect to an existing datasource, such as Dataverse, SharePoint, Excel, SQL, or upload a file.
- **Start with a plan →**: Use plans to describe your business problem and generate apps, tables, and more.

You can also create app using the **Create** menu on the left navigation pane. Review the following articles, to learn more about creating apps:

- Create a blank canvas app from scratch.
- Create a canvas app using Microsoft Dataverse
- create a canvas app based on Excel data.
- Create a canvas app with data from a list

> ⓘ Note
>
> If you're trying to create an app by using Dataverse but the option is unavailable, review the permissions table in this article.

Permissions for Dataverse

Can't create a Dataverse app? This table shows what happens based on your permissions and helps you understand your next steps.

Expand table

Maker has access to a developer environment	Maker has permission to create a developer environment	Maker has access to another Dataverse environment with required permissions	Action
Yes	n/a	n/a	Switch to the developer environment that appears first in the alphabetical list
No	Yes	n/a	Create a developer environment
No	No	Yes	Switch to another environment using the environment picker
No	No	No	Contact your admin

5 - View and edit plans

Select a plan to view and edit a plan. You can also, use the **Plans** menu in the left navigation pane to access your plans.

6 – View and edit apps

Manage your apps effectively:

- **Access your apps:** Use the **Apps** menu in the left navigation pane to find apps you own or apps shared with you
- **Find apps quickly:** Filter and sort apps to locate what you need
- **Take action:** Hover over any app name to **Edit**, **Play**, or access **Commands** for sharing, viewing details, and settings

Ready to customize? Learn advanced app editing techniques →

Last updated on 02/24/2026

Copilot and AI features in Power Apps documentation

Learn about Copilot and generative AI features in Microsoft Power Apps.

Get started

GET STARTED

Copilot in Power Apps overview

Copilot admin settings

FAQ about using AI responsibly

Enable or disable Copilot features

Improve Copilot responses from Microsoft Dataverse

Create and use plans

OVERVIEW

Overview and prerequisites

FAQ for plans

HOW-TO GUIDE

Create a plan

Create a plan from a solution

Create and edit apps with Copilot

HOW-TO GUIDE

Build apps through conversation with Copilot

Create a canvas app description with Copilot (preview)

Create a model-driven app description with Copilot

Use field suggestions by Copilot

Create a generative page in a model-driven app (preview)

Use Copilot to create and edit Power Fx formulas

Rename controls in canvas apps with Copilot (preview)

Add a Copilot control to a canvas app (preview)

Add a custom Copilot to a canvas app (preview)

Implement agent supervision and automation

HOW-TO GUIDE

Supervise agents using agent feed (preview)

Add agents to be supervised in app (preview)

Add an agent response control (preview)

Build an agent to automate your business process

Add agents to your model-driven app

Explore Copilot chat features

HOW-TO GUIDE

Add Microsoft 365 Copilot in model-driven apps (preview)

Use Microsoft 365 Copilot in model-driven apps (preview)

Customize Microsoft 365 Copilot with an agent (preview)

Streamline data entry

HOW-TO GUIDE

Get assistance filling forms

Use smart paste

Use Copilot in the email rich text editor

Explore and visualize data

HOW-TO GUIDE

Explore data in a view

Visualize data in a view

See row summaries in a form and view

Use prompt columns

Summarize and provide contextual insight

HOW-TO GUIDE

Use Copilot case summary

Use timeline highlights powered by generative AI

Create a custom row summary

Integrate custom AI experiences

CONCEPT

Use APIs to create custom AI powered experiences (preview)

Copilot in Power Apps overview

Copilot in Power Apps lets you quickly build apps using natural language. Describe your business needs, and the service generates an app and data model for you—no coding required. People interact with your app by asking questions or giving commands in everyday language, making the experience more intuitive and engaging. This article gives an overview of Copilot in Power Apps, including prerequisites, availability, and how to manage Copilot features in your environment. Learn about Copilot and generative AI features in Microsoft Power Apps.

Prerequisites

Each Copilot feature can have different requirements. Check the documentation for each feature to learn what you need.

> **Important**
>
> - This capability is powered by **Azure OpenAI Service**.
> - Some Copilot features are in preview.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to **supplemental terms of use**, and are available before an official release so customers can get early access and provide feedback.

Availability

- This capability might not be available in your region yet. Learn more in Explore Copilot features by geography and languages.
- This capability might be subject to usage limits or capacity throttling.
- Understand the capabilities and limitations of AI-powered and Copilot features in Power Apps. Learn more in FAQ about using AI responsibly in Power Apps.

Enable or disable Copilot features

Generally available Copilot features are on by default and can't be turned off except by Microsoft Support. To turn them off, a tenant admin needs to contact Support.

Turn off Copilot preview features in Power Apps

Copilot preview features are on by default, but your admin can turn them off for an environment or a tenant.

Turn off Copilot preview features for an environment

1. Sign in to the Power Platform admin center.
2. In the navigation pane, select **Environments**, and then select an environment.
3. On the command bar, select **Settings** > **Features**.
4. Set the **Copilot** toggle to **Off**.

Turn off Copilot preview features for your tenant

1. Sign in to the Power Platform admin center.
2. In the navigation pane, select **Settings** to open the **Tenant settings** page.
3. Select **Copilot in Power Apps (preview)**, and then set the toggle to **Off**.
4. Select **Save**.

> **Note**
>
> Turning off Copilot for your tenant turns off Copilot for makers only. It doesn't turn off the Copilot control for canvas apps or Copilot for model-driven apps.

Related information

- Copilot features in Power Apps
- FAQ about using AI responsibly in Power Apps
- Language availability for Power Platform
- Geographical availability for Power Platform

Note: The author created this article with assistance from AI. Learn more

---

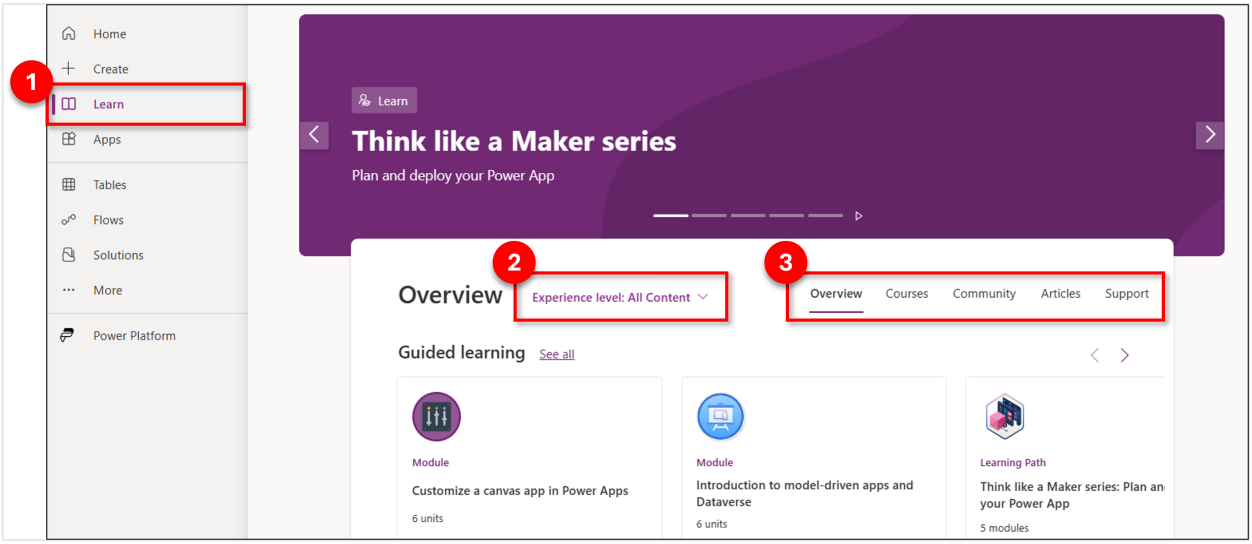
Last updated on 01/21/2026

Use the Learn hub

Article • 09/27/2023

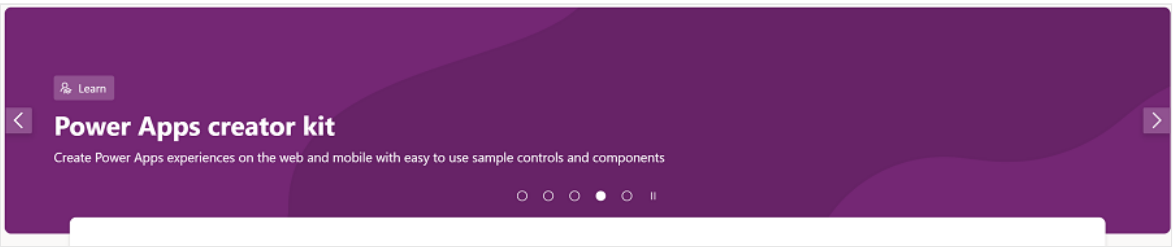
The Learn hub gives you easy access to resources that help you as you're learning to build Power Apps, include training material, product documentation, announcements and event notices, and help from the Power Apps community.

1. To access the Learn hub, go to the Power Apps home page and select **Learn** in the left navigation pane.
2. Use **Experience level** to select the content level that best suits your needs.
3. Select a tab to access the information that you're looking for.



The Learn hub includes the following sections:

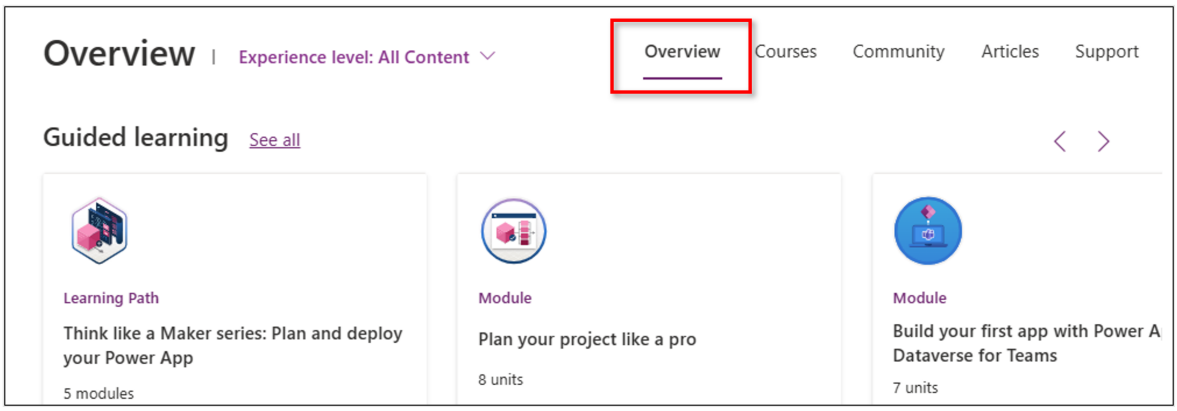
- The banner at the top showcases the latest announcements, trending posts, and information about upcoming Microsoft events.



- The **Experience Level** menu provides learning content based on the experience level that you select. For instance, if you are a developer seeking information about extensibility points in Power Apps, choose **Advanced** to find relevant help resources.



- The **Overview** tab lists featured training paths and modules, posts from the community, help articles, and training videos on YouTube.



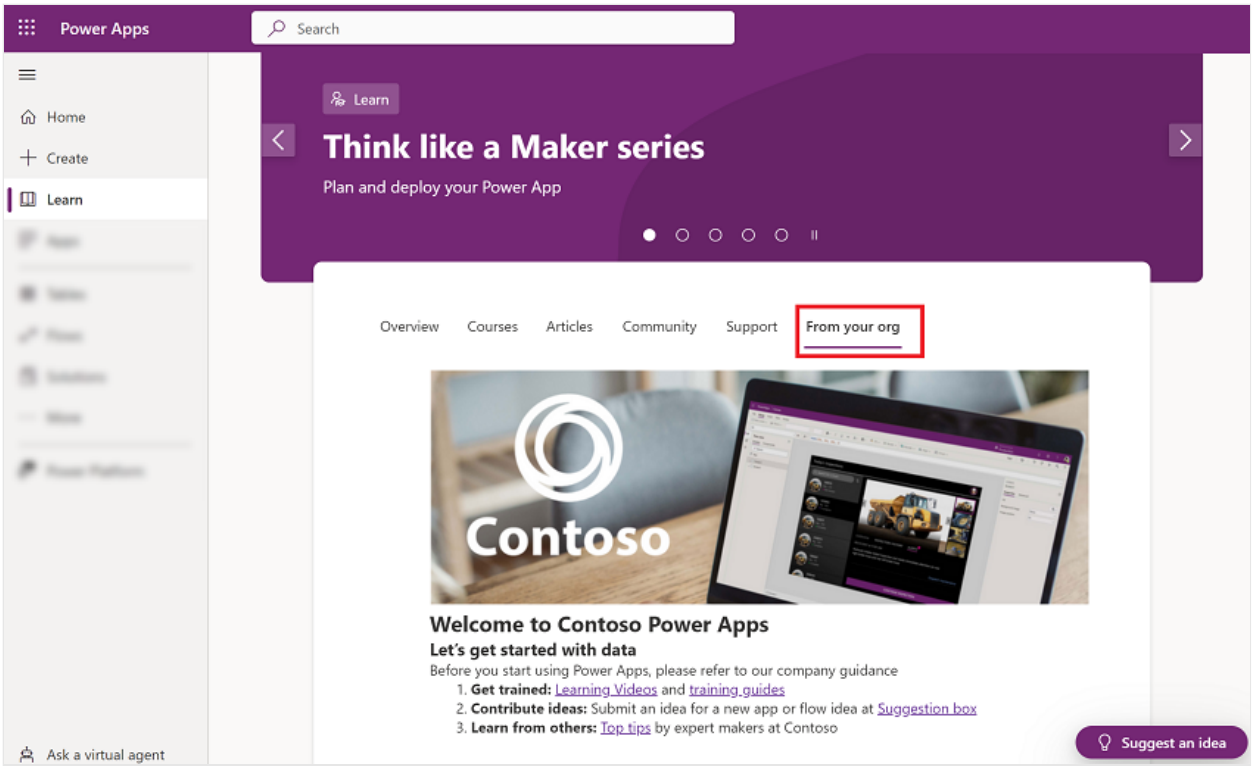
- The **Courses** tab provides guided training such as training paths, modules, and training events.
- The **Articles** tab lists relevant posts in public blogs.
- The **Community** tab provides access to content from the Power Apps community, including user groups, networking events, the LinkedIn Power Apps community, and more.
- The **Support** tab is where you can find help articles, report a problem, and submit an idea.

From your org (preview)

In Managed Environments, admins can provide customized welcome content to help makers get started with Power Apps. Learn about customizing maker welcome content (preview). If your organization has created custom welcome content, you see it when you sign in to Power Apps as a maker.

You can select **Don't show this again here** if you prefer not to see the custom content when you sign in.

To get back to it later, open the Learn hub and select **From your org**.


Power Apps
Search
Home
Create
Learn
Ask a virtual agent
Learn
Think like a Maker series
Plan and deploy your Power App
Overview
Courses
Articles
Community
Support
From your org
Contoso
Welcome to Contoso Power Apps
Let's get started with data
Before you start using Power Apps, please refer to our company guidance
1. Get trained: Learning Videos and training guides
2. Contribute ideas: Submit an idea for a new app or flow idea at Suggestion box
3. Learn from others: Top tips by expert makers at Contoso
Suggest an idea

What's new in Power Apps?

Article • 08/14/2024

This topic provides resources for you to learn about the new features that have been recently released, features that will be releasing over the next few months, and known limitations and resolutions.

Weekly releases

For information about the new features, fixes, and improvements released in the past few weeks, see Released versions for Microsoft Power Apps.

To learn more about how to browse information in the weekly release notes, read this blog post.

> **Note**
>
> Releases are rolled out over several days. New or updated functionality might not appear immediately.

Release plan

For information about new features being released over the next few months that you can use for planning, see Release Planner.

Monthly releases for model-driven apps

Power Platform admins can define the release cadence for features in model-driven apps.

Expand table

Channel	Frequency	Notes
Semi-annual	Twice per year	Rollout follows weekly releases starting in early April and early October. Default release cadence within Power Apps and Dynamics 365.
Monthly	Once per month	Rollout follows the weekly release schedule starting with the first weekly release for the month.

For more information about release channel options and monthly channel, go to Release channel overview.

Known limitations

For information about known limitations, see Common issues and resolutions.

See also

Power Platform release plans

Feedback

Was this page helpful? Yes No

Provide product feedback

Get your developer environment (preview)

Article • 04/18/2023

[This topic is pre-release documentation and is subject to change.]

By default, Microsoft Power Platform app makers can create up to three developer type environments. Developer environments are special environments intended only for use by the owner. Use these environments to build, test, and perform application lifecycle management (ALM) with Power Apps, Power Automate, and Microsoft Dataverse.

> **ⓘ Important**
>
> This is a preview feature.

Developer type environments allow you to:

- Create apps and flows without writing code, with full-featured Power Apps and Power Automate development tools. Easily share and collaborate on these solutions with others.
- Use a fully managed, scalable data platform with Dataverse, including support for common business application actions. Use out-of-the-box common tables or build your own data schema.
- Export the solutions you create in your developer environment and then import them into other Power Platform environments so users can test and use them.

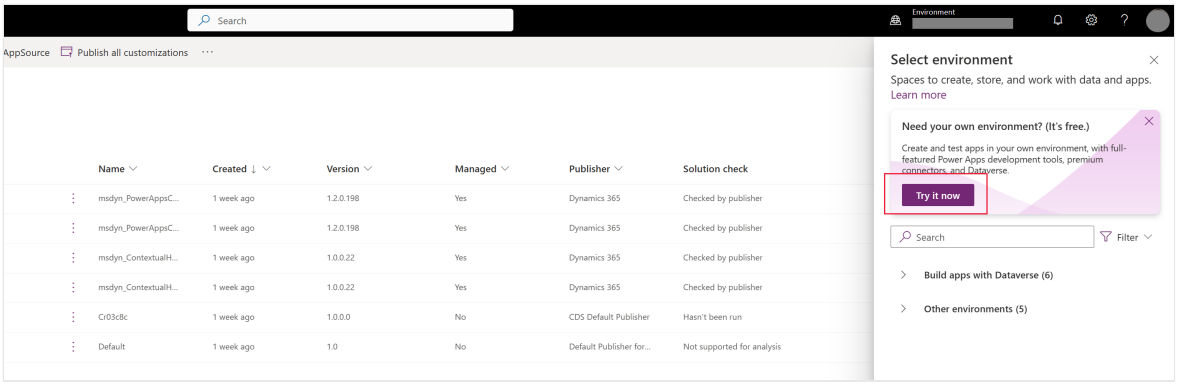
Prerequisites

To create a developer environment, you need an active Power Platform plan that includes Dataverse. More information: Licensing overview for Microsoft Power Platform

Create your own developer type environment

1. Sign in to Power Apps.

2. Select the environment picker > **Try it now**.



3. Follow the instructions on your screen to create your environment.

Create your own environment when you sign up for the Power Apps Developer Plan

The Power Apps Developer Plan gives you up to three free development type environments to build and test with Power Apps, Power Automate, and Microsoft Dataverse. The developer environments maintained under this plan will be available as long as you actively use the Power Apps Developer Plan.

More information: Create a developer environment with the Power Apps Developer Plan

See also

Environments overview

Power Apps preview program

Article • 05/15/2024

Microsoft Power Apps updates the platform and its capabilities every few days or weeks. The Power Apps preview program is a way to get early access to features and updates before they become available in other regions (where customer production apps are deployed).

With the Power Apps preview program, you can:

- **Try out, learn, and dogfood upcoming features**: Many features are rolled out first in the preview for a few days to get feedback. By participating in the preview program, you can learn about new features sooner and provide feedback. Also, you can quickly take advantage of new features as soon as they reach regions where their production apps are created.
- **Enable business continuity by ensuring current apps continue to work** with the upcoming updates (vNext) of Power Apps.

What in Power Apps is available for preview?

To access the preview features, you need to be in a preview environment or using a preview URL. We have preview features for the following areas across Power Apps.

Canvas apps

For preview, experimental, and retired features in canvas apps, use Power Apps Studio to turn the features On or Off while editing your app.

For early access to upcoming changes in canvas apps, use the following environments:

- Preview environments - Upcoming changes are typically rolled out to preview environments 1-2 weeks before they're available to all other regions.
- Sandbox environments - To access preview version of Power Apps on sandbox environments, use make.preview.powerapps.com.

Model-driven apps and app management

You can create, manage, and share apps using Power Apps (make.powerapps.com). To use the preview features, we recommend that you are in a preview or sandbox environment and not in a production environment. Then, go to the preview version of Power Apps (make.preview.powerapps.com).

How to get early access to the upcoming updates?

> **Note**
>
> **Preview (United States)** has been replaced with the **Early release cycle** option. Following the instructions below, you can create preview, or *early release cycle* environments in the United States region and other regions that may be closer to your location.

For Power Platform, all the apps, flows, and related resources are stored in an environment. Early access to all preview features is available when an admin creates an environment that supports the **Get new features early** option. A subset of the regions currently have this capability and the environment must be created in one of those regions. For more information, see Early release cycle environments.

All the apps and other resources created in this environment are on the vNext version of the platform.

How to learn about the latest updates?

You can get aware of the new features, which are available for preview at What's new in Power Apps. The features that are only available in the preview have a 'Preview' tag.

Key scenarios to test with the preview program

1. **Validate your production apps with the upcoming Power Apps updates (vNext)**

   You might like to verify your production apps, to be working fine with the next upcoming updates on Power Apps. You can copy the apps from a production environment to an environment in First Release and play the apps to test out the scenarios. Note, all the other necessary resources like CustomAPI, Power Automate, etc., need to be moved along with it. This should just create another copy of these apps and required resources. You can start testing out the newer updates not just for playing an app, but also while editing and managing the apps.

2. **Trying out the new features available in preview**

   We launch many new features initially in the **Preview (United States)** region. You can try out the new features before they become available in the rest of the

regions (which might impact your production environment).

How to provide feedback to the product team?

You can provide feedback on the Power Apps forum and/or contact support.

What are the known issues and limitations?

1. **Clients not available in preview**

   There are certain features, and services available in preview:

   Expand table

Create an app	Playing an app	Others	Available in preview?
Web studio	Web player	Flow, Connectors and Gateways	Yes
Desktop version of studio	Power Apps mobile for iOS, Android, and Windows		No

2. **Using apps created in preview environments in production environments**

   Power Apps doesn't support opening apps saved in preview only versions of Power Apps in production environments. Most versions of Power Apps eventually move from preview into production, but how and when this happens is influenced by many factors so it shouldn't be relied on. We recommend you use production environments to create or edit any app intended for use in a production environment.

Feedback

Was this page helpful? Yes No

Provide product feedback

System requirements, limits, and configuration values for Power Apps

This article contains information on supported device platforms, web browser requirements, limits, and configuration values for Power Apps.

Supported platforms for running apps by using the Power Apps mobile app

> **Note**
>
> - To ensure an optimal user experience, test your app on the specific devices you plan to use for production deployment.
> - Performance testing and results can vary between device types such as mobile, desktop, and laptops. Factors include processing power, memory capacity, network connectivity, app complexity, and other apps running in parallel on a device. Make sure the test results meet your business requirements before you roll out the solutions in production.

Expand table

Platform	Version
iOS	The latest version of iOS is always the recommended version to run Power Apps mobile. The previous version is the minimum required.
Android	The latest version of Android is always the recommended version to run Power Apps mobile. The previous three versions are the minimum required to run Power Apps mobile.
Windows	Windows 10 version 17763.0 or later to run Power Apps for Windows.

> **Note**
>
> New major versions of iOS and Android release each year. When a new version releases, if you're using the oldest previously supported version, you have 60 days to update your device to at least the new minimum supported version to continue to run Power Apps mobile.

Supported browsers for running Power Apps

Expand table

Browser	Supported versions	App type
Google Chrome	Latest three major releases	Model-driven apps, canvas apps, app and component designers[1].
Microsoft Edge	Latest three major releases	Model-driven apps, canvas apps, app and component designers[1].
Mozilla Firefox	Latest three major releases	Model-driven apps and canvas apps.
Apple Safari	13 and later	Model-driven apps and canvas apps.

[1]App and component designers include Power Apps studio, model-driven app designer, and model-driven custom page designer.

Supported operating systems for browsers running Power Apps

Expand table

Operating system	Supported versions	App type
Windows	Windows 10 or later	Model-driven apps, canvas apps, app and component designers[1].
macOS	10.13 or later	Model-driven apps, canvas apps, app and component designers[1].
iOS	iOS 13 or later	Model-driven apps[2] and canvas apps.
Android	10 or later	Model-driven apps[2] and canvas apps.

[1]App and component designers include Power Apps studio, model-driven app designer, and model-driven custom page designer.

[2]Using the web browser on a phone to run a model-driven app isn't supported; use the Power Apps mobile app.

For classic web application system requirements, see Web application requirements.

> **Note**
>
> Canvas apps running on Windows platform must use the legacy Microsoft Dataverse connector. A warning is displayed for apps that still use the legacy connector, but using it for Windows platform is supported.

Request limits

These limits apply to each single outgoing request:

Expand table

Name	Limit
Timeout	180 seconds
Retry attempts	4

> **Note**
>
> The retry value might vary. For certain error conditions, retrying isn't necessary.

IP addresses

Requests from Power Apps use IP addresses that depend on the region of the environment that the app is in. Microsoft doesn't publish fully qualified domain names for Power Apps scenarios.

Calls made from an API connected through an app (for example, the SQL API or the SharePoint API) come from these IP addresses.

Required services

This list identifies all services to which Power Apps communicates and their usages. Your network must **not** block these services.

Expand table

Domains	Protocols	Uses
api.bap.microsoft.com *.api.bap.microsoft.com	https	Environment permissions management
management.azure.com	https	Power Apps Management Service
msmanaged-na.azure-apim.net	https	Runtime of Connectors/Apis
login.microsoft.com login.windows.net login.microsoftonline.com secure.aadcdn.microsoftonline-p.com *.odc.officeapps.live.com	https	Microsoft Authentication Library
graph.microsoft.com graph.windows.net	https	Azure Graph - For getting user info (for example, profile photo)
*.azure-apim.net	https	API Hubs - Different subdomains for each locale
*.azure-apihub.net	https	API Hubs - Different subdomains for each locale
*.powerapps.com	https	create.powerapps.com, content.powerapps.com, apps.powerapps.com, make.powerapps.com, *gateway.prod.island.powerapps.com, *gateway.prod.cm.powerapps.com, *harvest.preview.powerapps.com, and *harvest.powerapps.com
*.gateway.prod.island.powerapps.com *.harvest.preview.powerapps.com *.harvest.powerapps.com	WSS	Required for communication with and startup of canvas apps and real-time updates with Coding Agent.
*.azureedge.net	https	create.powerapps.com, content.powerapps.com, and make.powerapps.com (Optional) We highly recommend that you use the wildcard listed under Domains. You can download the complete list if you want to allow specific domain names instead of using *.azureedge.net. However, this list is subject to change.
*.azurefd.net	https	create.powerapps.com, content.powerapps.com, and make.powerapps.com

Domains	Protocols	Uses
*.ces.microsoftcloud.com config.centro.core.microsoft admin.microsoft.com petrol.office.microsoft.com oness.microsoft.com	https	Access to net promoter score (NPS) and surveys.
*.blob.core.windows.net	https	Blob storage (Optional) We highly recommend that you use the wildcard listed under Domains. You can download the complete list if you want to allow specific domain names instead of using *.blob.core.windows.net. However, this list is subject to change.
*.flow.microsoft.com *.powerautomate.com	https	create.powerapps.com, content.powerapps.com, and make.powerapps.com
http://*.crm#.dynamics.com and https://*.crm#.dynamics.com	https	Required for environments access. Includes integration and static Content Delivery Network (CDN) content endpoints. Replace # in http://*.crm#.dynamics.com and https://*.crm#.dynamics.com with your region's number: • Asia/Pacific: 5 • Canada: 3 • Europe, Africa, and Middle East: 15 and 4 • France: 12 • Germany: 16 • India: 8 • Japan: 7 • North America: no number • Oceania: 6 • Singapore: 20 • South Africa: 14 • South America: 2 • Switzerland: 17 • UAE: 15 • United Kingdom: 11 • Dynamics 365 US Government: 9
browser.pipe.aria.microsoft.com	https	Rest of the World telemetry endpoint for

Domains	Protocols	Uses
		model-driven apps
*.events.data.microsoft.com	https	Telemetry endpoint for Power Apps
localhost	http	Power Apps Mobile
127.0.0.1	http You might need to explicitly list the port (5040 and up) for localhost/127.0.0.1.	Power Apps Mobile
ecs.office.com	https	Retrieve feature flags for Power Apps
augloop.svc.cloud.microsoft *.augloop.svc.cloud.microsoft	WSS	Power Apps Studio Copilot
config.edge.skype.com	https	Retrieve feature flags for Power Apps (backup)
api.powerplatform.com *.powerplatform.com *.api.powerplatformusercontent.com *.powerplatformusercontent.com	https	Required for Power Platform API connectivity used internally by Microsoft products, and Power Platform programmability and extensibility.
*.sharepointonline.com	https	Retrieve assets for presenting the header that appears at the top of app playing experiences
ris.api.iris.microsoft.com eudb.ris.api.iris.microsoft.com	https	Record user action in response to Power Apps in-app campaigns
arc.msn.com arc-emea.msn.com	https	Record user viewing of Power Apps in-app campaigns
*.hubblecontent.osi.office.net hubble.officeapps.live.com res.cdn.office.net	https	Provides stock images to use in your app
dc.services.visualstudio.com	https	Application Insights endpoint used for custom telemetry in canvas app
*.ocv.microsoft.com	https	Allows users to provide Copilot feedback data in Power Apps
js.monitor.azure.com	https	This is used by the office header for reporting telemetry

[1] Replaces domain name `gov.content.powerapps.us` used before July 2022.
[2] Replaces domain name `high.content.powerapps.us` used before July 2022.
[3] Replaces domain name `content.appsplatform.us` used before July 2022.

> **Important**
>
> - If you're using a VPN, it must be configured to exclude localhost from tunneling for Power Apps Mobile.
> - If you're a US Government customer or a customer in China, unblock additional service URLs. For more information:
>
> **US Government**: Power Apps US Government service URLs, Power Automate US Government service URLs
>
> **China**: Power Platform and Dynamics 365 apps - operated by 21Vianet in China.

Deprecated endpoints

This section lists endpoints that Microsoft no longer supports. Instead, use the new endpoints listed in the table. Update any bookmarks that use the deprecated endpoint to the new endpoint.

Expand table

Deprecated endpoint	New endpoint
web.powerapps.com/apps/{yourAppGuid}	apps.powerapps.com/play/{yourAppGuid}
web.powerapps.com/apps/{yourAppGuid}/open	apps.powerapps.com/play/{yourAppGuid}

Embedding limits for Power Apps

Canvas app embedding

Power Apps doesn't support the nested embedding of canvas apps in native desktop, mobile, or other non-browser clients.

The following table shows some of the examples where embedding a canvas app is and isn't supported:

Expand table

Power Apps embedding scenario	Supported clients	Unsupported clients
A canvas app embedded in a SharePoint page.	Web, for example, Sharepoint.com	• Teams desktop • Teams mobile
A canvas app embedded in a SharePoint page that is added as a tab in a Microsoft Teams channel.	Web, for example. • Sharepoint.com • Teams.microsoft.com	• Teams desktop • Teams mobile
A canvas app used as a custom form in the SharePoint page.	Web, for example, Sharepoint.com	• Teams desktop • Teams mobile
A canvas app used as a custom form in the SharePoint page that is added to a Teams team.	Web, for example. • Sharepoint.com • Teams.microsoft.com	• Teams desktop • Teams mobile • SharePoint mobile
A Power BI report that is added to Teams, or a SharePoint site.	Web, for example, Teams.microsoft.com	• Teams desktop • Teams mobile • SharePoint mobile
A canvas app is embedded in a third party client.	Third party websites in a browser	Third party native applications
A canvas app in Unified Service Desk	None	Unified Service Desk Desktop app
A canvas app embedded in Power BI	Web, for example, app.powerbi.com	Power BI Desktop Client

Model-driven app embedding

Power Apps doesn't support embedding a model-driven app or page within an IFrame in another application.

Proxies

Power Apps doesn't support running with a proxy enabled. This limitation can cause unpredictable behavior. If you encounter problems, disable the proxy and then try again.

- Some proxies, such as Zscaler and Blue Coat, remove headers from Power Apps requests. Power Apps relies on these headers, like CORS or authentication headers, to load the app.

- Some proxies, such as Microsoft Defender for Cloud Apps and McAfee, might intercept and change the URL of an app or embedded app. For example, if there's a Dynamics 365 app that runs under domain **org.crm.dynamics.com** or a canvas app that runs under domain **apps.powerapps.com**, the platform doesn't support a proxy that changes these domains to a custom domain such as **mycustomdomain.com**. This limitation can cause unpredictable behavior when the platform tries to retrieve tokens that are necessary to run the app.

Local network access and permission problems

In 2025, many browsers introduced additional security controls to prevent cross-origin attacks and unauthorized access to private network resources. These changes went into effect for Edge with their December 4, 2025 release.

Some proxies are configured by default to host resources in a way that appears to the end-user's browser as a local network resource. This configuration now results in those requests being blocked. This behavior might appear to end-users as unexpected permission prompts, silent failures, or generic messages such as "This app stopped working. Try refreshing your browser".

> **Note**
>
> Microsoft Support can't resolve these networking problems. Enterprise admins might be able to mitigate this problem by updating browser policies such as LocalNetworkAccessAllowedForUrls for Microsoft Edge and Google Chrome or by updating proxy configurations according to their vendor's guidelines.

Data types size limits

For canvas app data type limits, you can find information about size limits on text, hyperlinks, images, and media in Data types in Power Apps.

For Microsoft Dataverse data type size limits, you can find information on column types, such as text and image columns, in Types of columns.

Power Apps per app plan

For more information, see Power Apps per app plan in the Power Platform admin guide.

See also

System requirements, limits, and configuration values for Power Pages

Last updated on 01/14/2026

How do I find and run apps?

Article • 10/23/2024

No need to manually work on a task to get your work done. With Power Apps, you can create and use a canvas app or model-driven app to get your work done quickly. Once you create an app, you can share the app with your organization.

Canvas apps give you much control over the user experience of the app. An app maker can use templates to create a canvas app, or they can configure the app any way they like. A canvas app, like its name, is a blank canvas, so you can create an app to fit the needs of users in your organization.

Because canvas apps can essentially be anything, there isn't much help documentation on how to use an app once it's created. If you need help with using a canvas app, contact your app maker or Power Apps admin.

Model-driven apps use the Unified Interface that provides a responsive and accessible design. You can run the app on a web browser or on popular mobile devices. The app has multiple components including dashboards, forms, views, charts, and business processes that together help make the app easy to use.

This section provides information about how to find and run canvas and model-driven apps from various places. It also provides information on how to navigate and work within a model-driven app: Use model-driven apps.

What's required to run apps?

Ensure that you're using a supported web browser and operating system. For more information, see System requirements, limits, and configuration values for Power Apps

Additionally, you need a Power Apps license to run an app. If you don't have a license, you can request one by following instructions here: Request a Power Apps license

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Run an app in a web browser

Article • 06/05/2024

When you create an app, or someone shares an app with you, you can run that app on the Power Apps mobile app or in a web browser. In this topic, you'll learn how to run a canvas or model-driven app in a web browser from the Microsoft 365 apps page.

For full functionality and optimized experience on a tablet device, we strongly recommend that you use the Power Apps mobile app. If you don't have the Power Apps mobile app installed, you can still use the web browser on your tablet, as long as your device has sufficiently high screen resolution. For more information: Required privileges and supported devices.

> **Note**
>
> Using the web browser on your phone to run your model-driven apps isn't supported; you must use the **Power Apps mobile app**.

To follow this quickstart, you need:

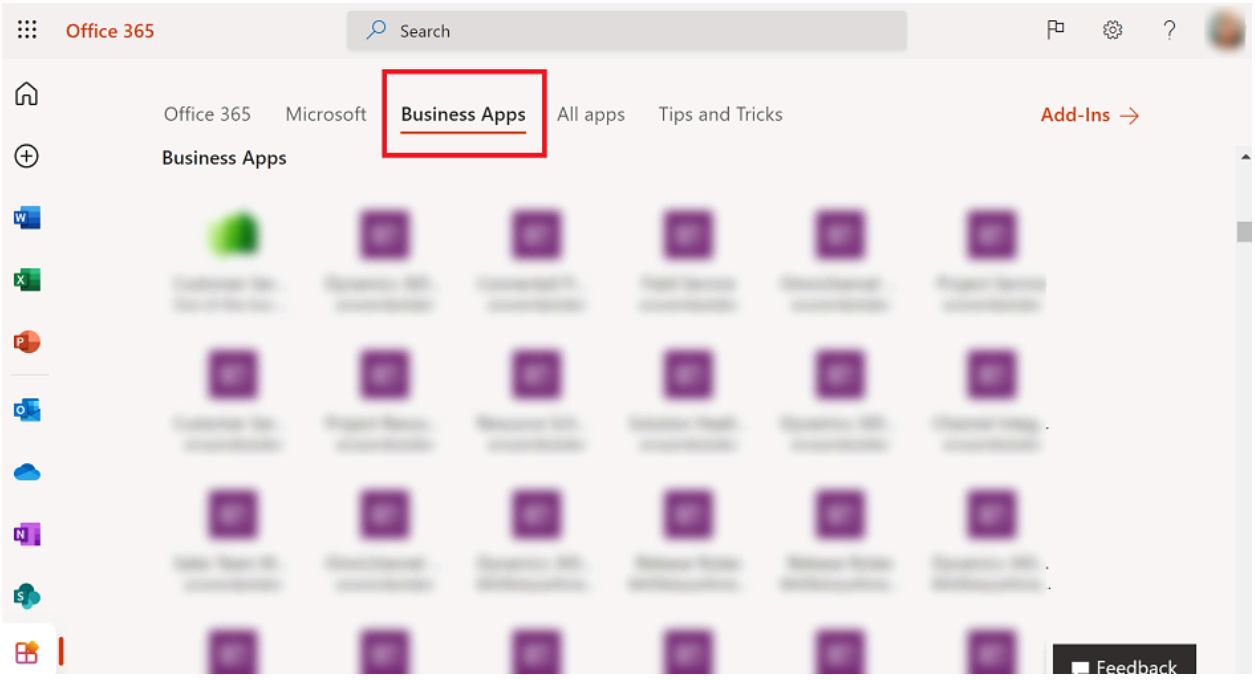
- A Power Apps license. This is available with a Power Apps plan, such as the Power Apps per user plan, or Power Apps trial, or any of the Microsoft Office 365 or Dynamics 365 plans that include Power Apps.
- Access to an app that you built or that someone else built and shared with you.
- Access to a supported web browser and operating system. For more information: System requirements, limits, and configuration values for Power Apps

Sign in to Microsoft 365 apps page

Sign in to the Microsoft 365 apps page.

Find an app on the Home page

The Home page may show several types of business apps.



The list of Office apps is a limited, curated list based on the following criteria:

- Apps that you launched in the last week.
- Apps that you created.
- Apps your admin marked as **Featured** in your tenant.
- First-party, model-driven apps such as Dynamics 365 Sales, Dynamics 365 Customer Service, and Dynamics 365 Field Service that you can access and are hosted in production or default environments.

> **Note**
>
> Marking an app as **Featured** is the only way to guarantee inclusion in the list.

Known limitations

- When a new app is shared with users, the app needs to meet the criteria listed above. It can take up to 24 hours for the app to appear in the list of **Office** apps.
- Some model-driven apps aren't recording the last launched time. Until this issue is resolved, we recommend using featured apps. Featured apps only appear to users who have access to the apps. Featured apps don't appear to all users.
- It's possible that users will be able to see model-driven apps that they used to be able to access, but can no longer. There's also a chance that users won't be able to see model-driven apps that they have access to.
- It can take up to a week for users to start seeing the new URL after an org URL update.

- In the case of model-driven apps, changes made only to the icon may not be reflected. Until this issue is resolved, changing another property, like description, causes the icon to be updated.

What if I can't find an app?

Apps that haven't been used in the last week won't appear in the list of **Office** apps. If you can't find an app, then you need to use the web link to open the app directly in your browser.

1. To find the web link for an app, sign in to Power Platform admin center or Power Apps.
2. Find the app that you're looking for.
3. Across the app name, select (...) **More Commands** > **Details**.
4. From the **Details** tab copy the **Web link**.
5. Paste the web link in a browser window to launch the app.

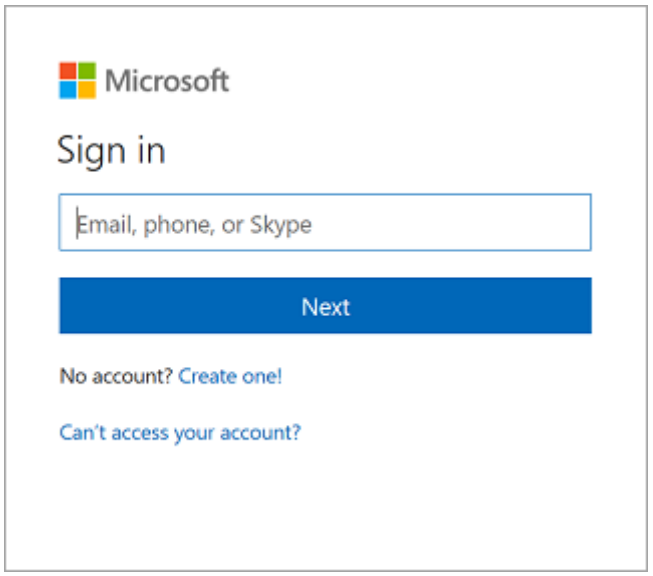
The Microsoft 365 apps page has a different URL per cloud.

Expand table

#	Cloud	Microsoft 365 app discovery URL
1	Commercial	https://www.office.com/apps
2	Government Community Cloud (GCC)	https://www.office.com/apps
3	Government Community Cloud (GCC High)	https://www.office365.us/apps
4	Department of Defense	https://www.ohome.apps.mil/apps
5	Office 365 operated by 21Vianet	There isn't a corresponding https://www.office.com/apps experience for this cloud. However, the Microsoft 365 app launcher experience throughout Microsoft 365 experiences includes Power Apps and Dynamics apps such as new word document in https://portal.partner.microsoftonline.cn/NewSpoDocument?docType=Word

Run an app from a URL

You can save an app's URL as a bookmark in your browser and run it by selecting the bookmark, or you can send a URL as a link through email. If someone else created an app and shared it with you in an email, you can run the app by selecting the link in the email. When running an app using a URL, you may be prompted to sign in using your Azure Active Directory credentials.



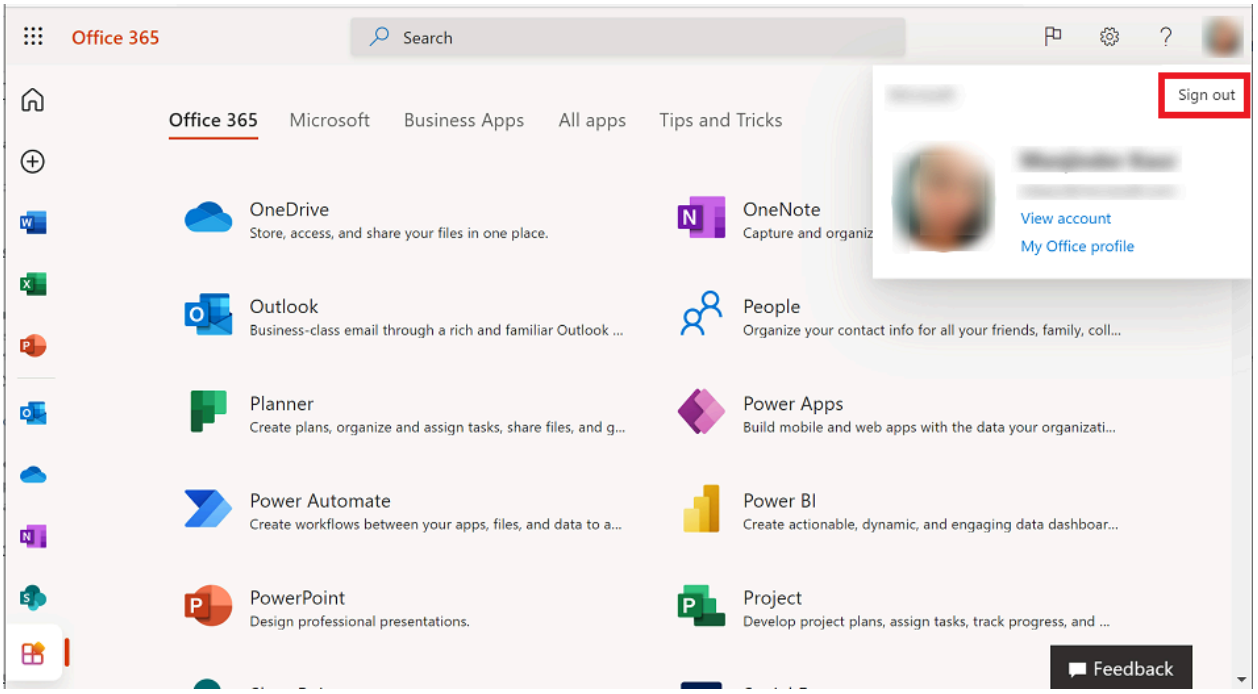
Connect to data

If an app requires a connection to a data source or permission to use the device's capabilities (such as the camera or location services), you must give consent before you can use the app. Typically, you're prompted only the first time.



Sign out

To sign out of the Microsoft 365 apps page, select your profile image and then select **Sign out**.



Next steps

In this topic, you learned how to run a canvas or model-driven app in a web browser. To learn how to:

- run a model-driven apps or canvas apps on a mobile device, see Run model-driven apps and canvas apps on Power Apps mobile
- use a model-driven app, see Use model-driven apps
- mark an app as featured, see Set-AdminPowerAppsFeatured

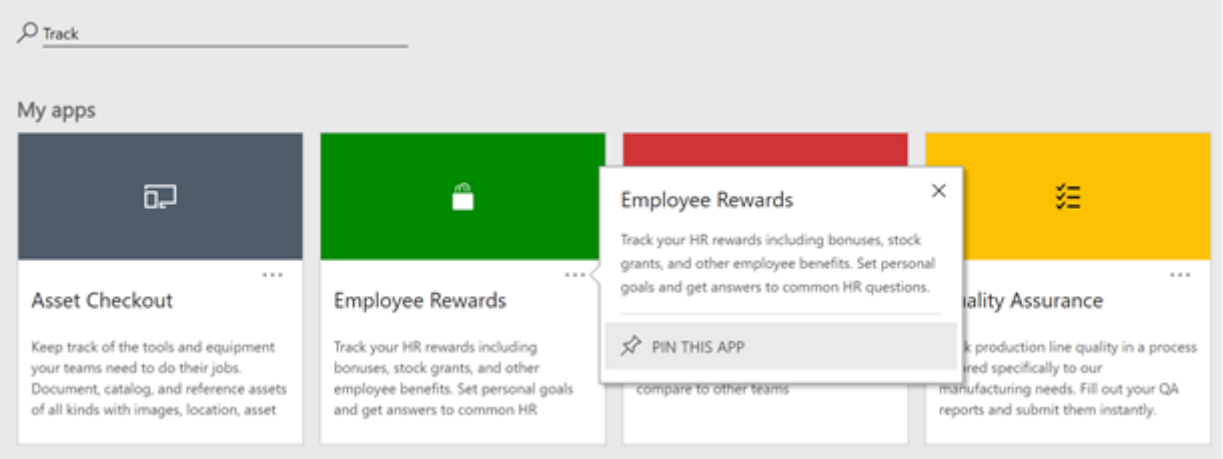
Feedback

Was this page helpful? Yes No

Provide product feedback

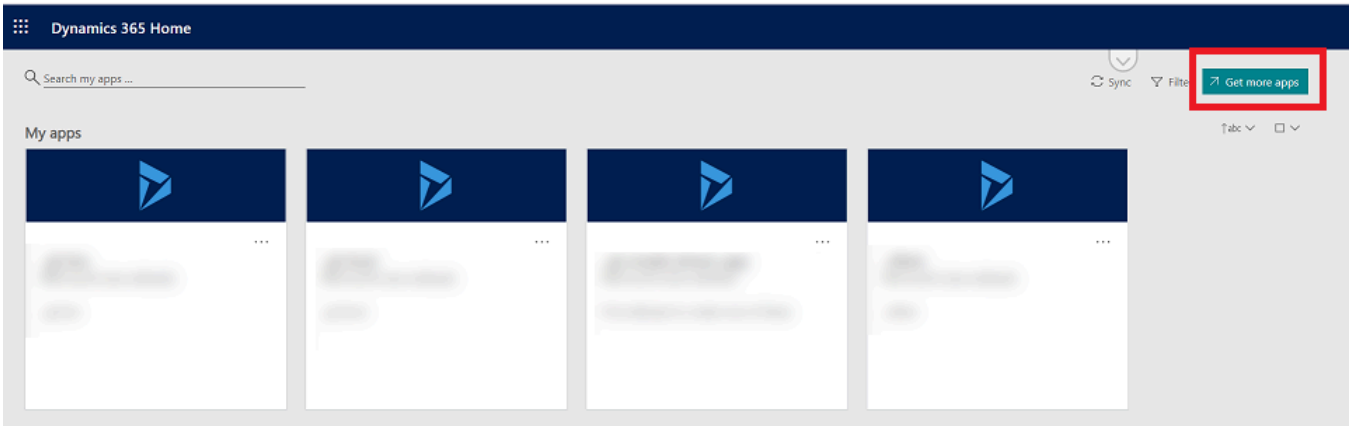
Discover apps via Marketplace

The apps you build and publish with Power Apps are discovered and used in Dynamics 365 on the web. It displays all of your apps--the apps that you've chosen to use (that is, by launching from a share email or opening from Marketplace) or that an admin has provisioned for you. This includes all of your apps from Power Apps, and Dynamics 365 applications from Microsoft. You can search for apps, filter by environment, and pin the apps you use most frequently to the top of the page.

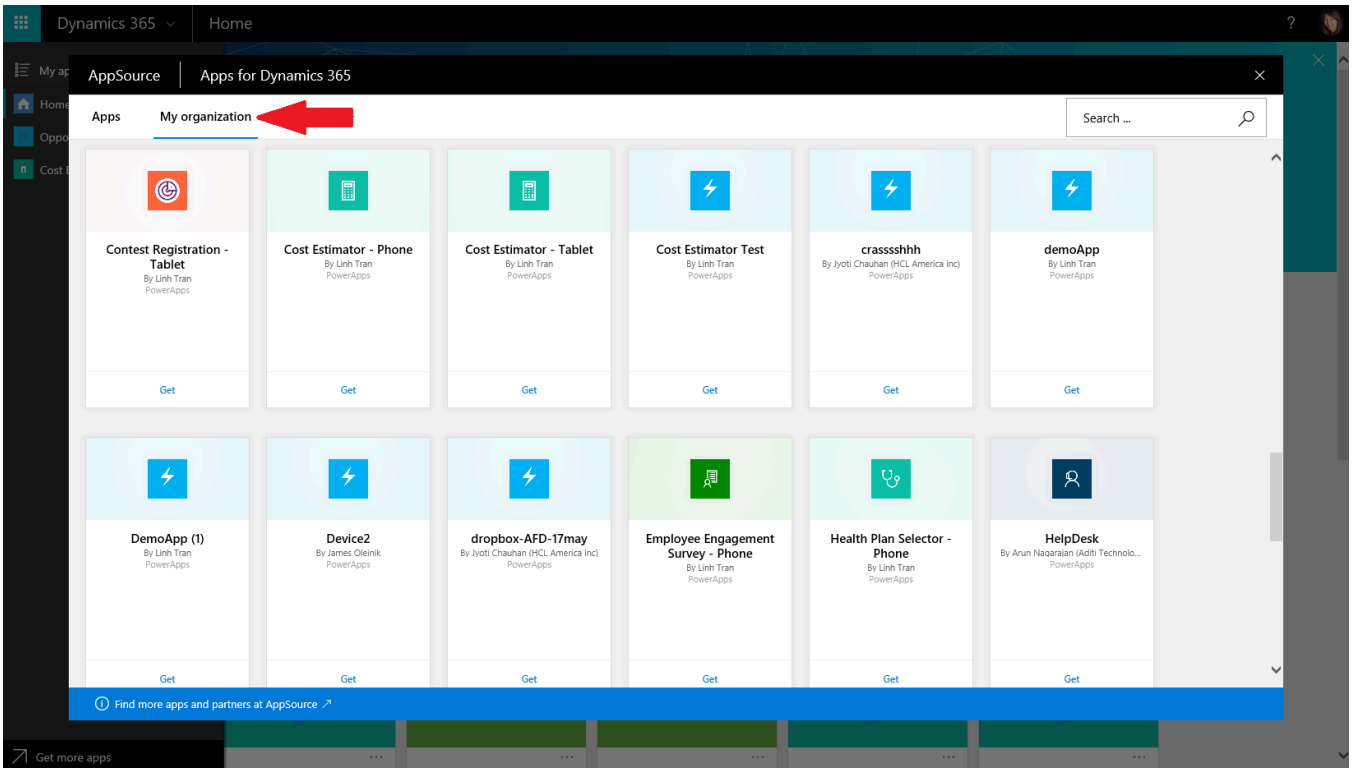


Find apps via the Marketplace organization gallery

Microsoft Marketplace is now embedded in the Microsoft 365 apps (https://www.office.com/apps) home page and throughout the common navigation. It now includes a private gallery of apps available to you within your company. Select **Get more apps** from the home page or task pane, and navigate to the **My Organization** tab to see apps that are available to you.



Apps built with Power Apps and shared with a security group or the entire company with **User** permission appear here, instead of cluttering your home page by default. When you get an app from Marketplace, it will then appear on the home page and be easily accessible throughout the experience.



Last updated on 01/16/2026

Use Microsoft 365 Copilot in canvas apps (preview)

Microsoft 365 Copilot for apps is a next-generation AI assistant that helps you gain insights into the data in your apps through conversations in natural language. Microsoft 365 Copilot boosts your productivity through AI-powered insights and navigation assistance.

> **Important**
>
> - This feature is in preview.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use. They're available before an official release so that customers can get early access and provide feedback.
> - This feature is in the process of rolling out and might not be available in your region yet.

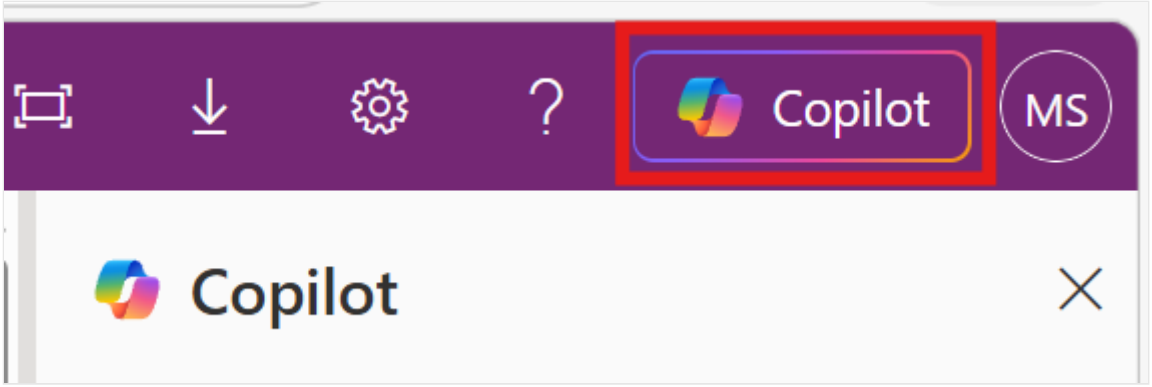
Prerequisites

An administrator must enable Microsoft 365 Copilot in your application before it becomes visible in your app. For more information, see Add Microsoft 365 Copilot for app users.

> **Note**
>
> - To use the Microsoft 365 Copilot feature in your app, you must have a Microsoft 365 Copilot license.
> - You must have access to an early release cycle environment to use this feature.

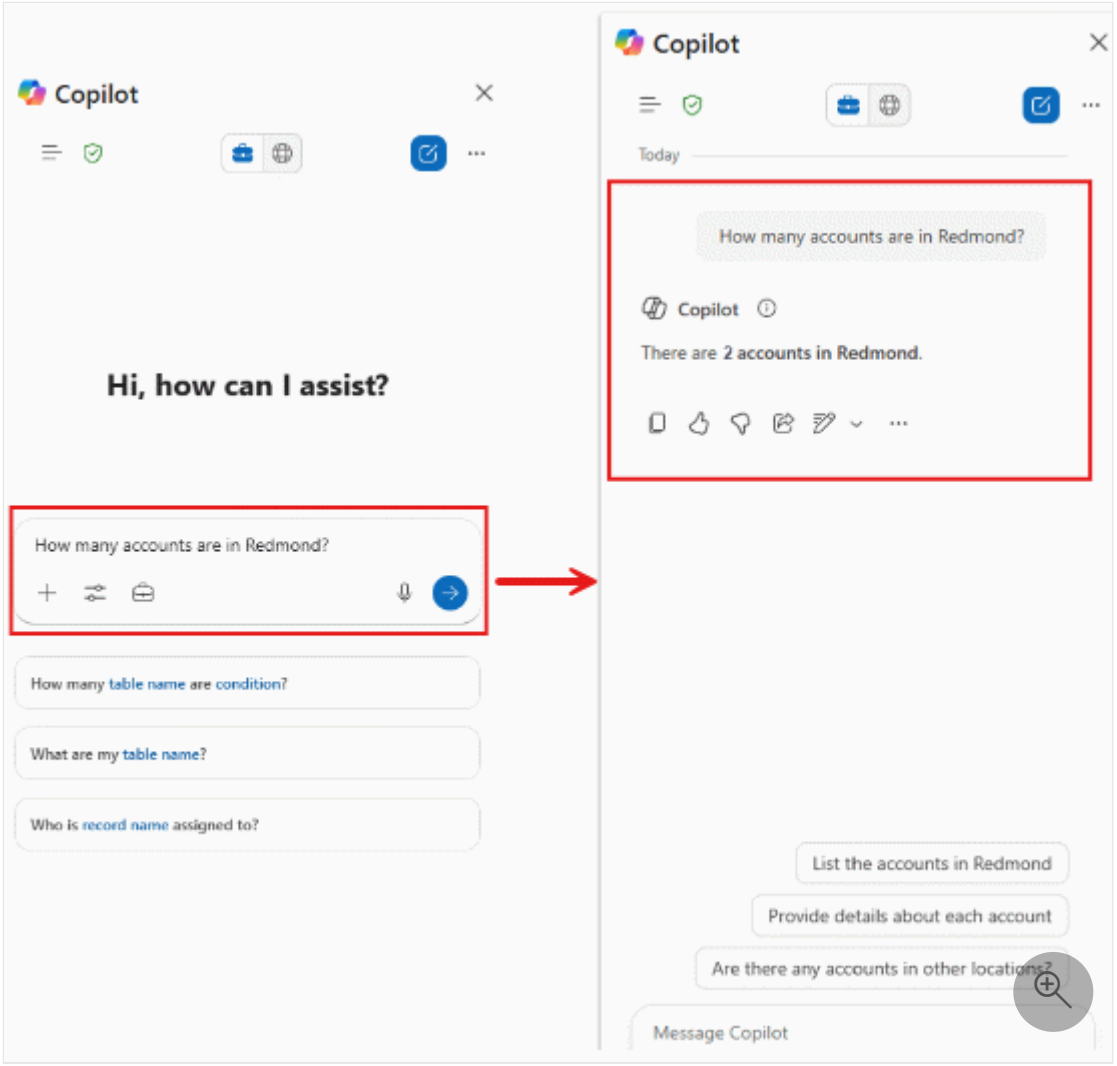
Copilot pane

After Microsoft 365 Copilot is enabled, access it through the Copilot button near the upper-right corner of the page.



Use Microsoft 365 Copilot to ask questions

Microsoft 365 Copilot in apps answers questions about the Dataverse tables or SharePoint list data in the app.



Share your feedback

Help us improve Copilot responses by using the feedback controls on each response. Provide feedback for every response:

- If a response is high quality and helpful, select the thumbs up button.
- If a response is incorrect, incomplete, or not helpful, select the thumbs down button.

When you provide feedback, share as much information as possible. For example, include what you expected to see, what was missing or incorrect, and any relevant context. The more detail you share, the better we can understand your feedback and continuously improve responses.

Copilot

Today

Show me all the accounts in Redmond.

Copilot

Here are **all the accounts in Redmond** that I found in your app. There are **3 accounts total** that have their city set to *Redmond*:

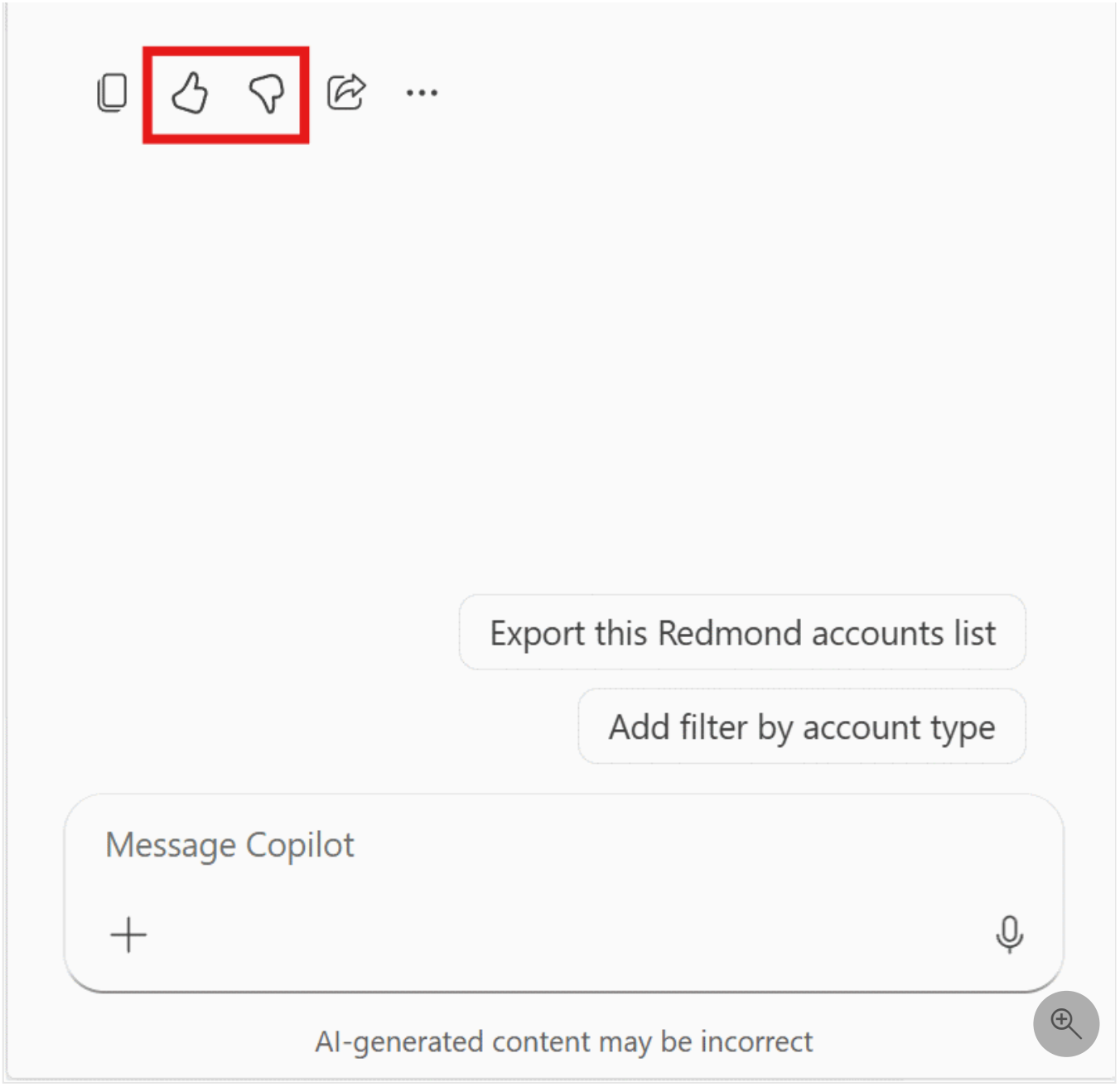
Accounts in Redmond

Account Name	City
City Power & Light (sample)	Redmond
Contoso Pharmaceuticals (sample)	Redmond
A. Datum Corporation (sample)	Redmond

If you'd like, I can:

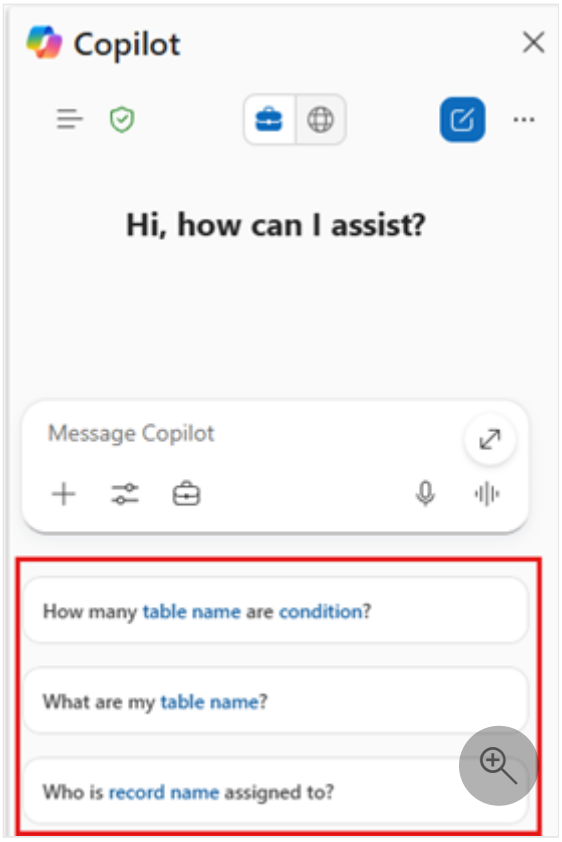
- Open any of these accounts
- Export this list
- Add additional filters (state, owner, account type, etc.)
- Turn this into a saved view for quick access

Just let me know how you want to use this.



Microsoft 365 Copilot suggested questions

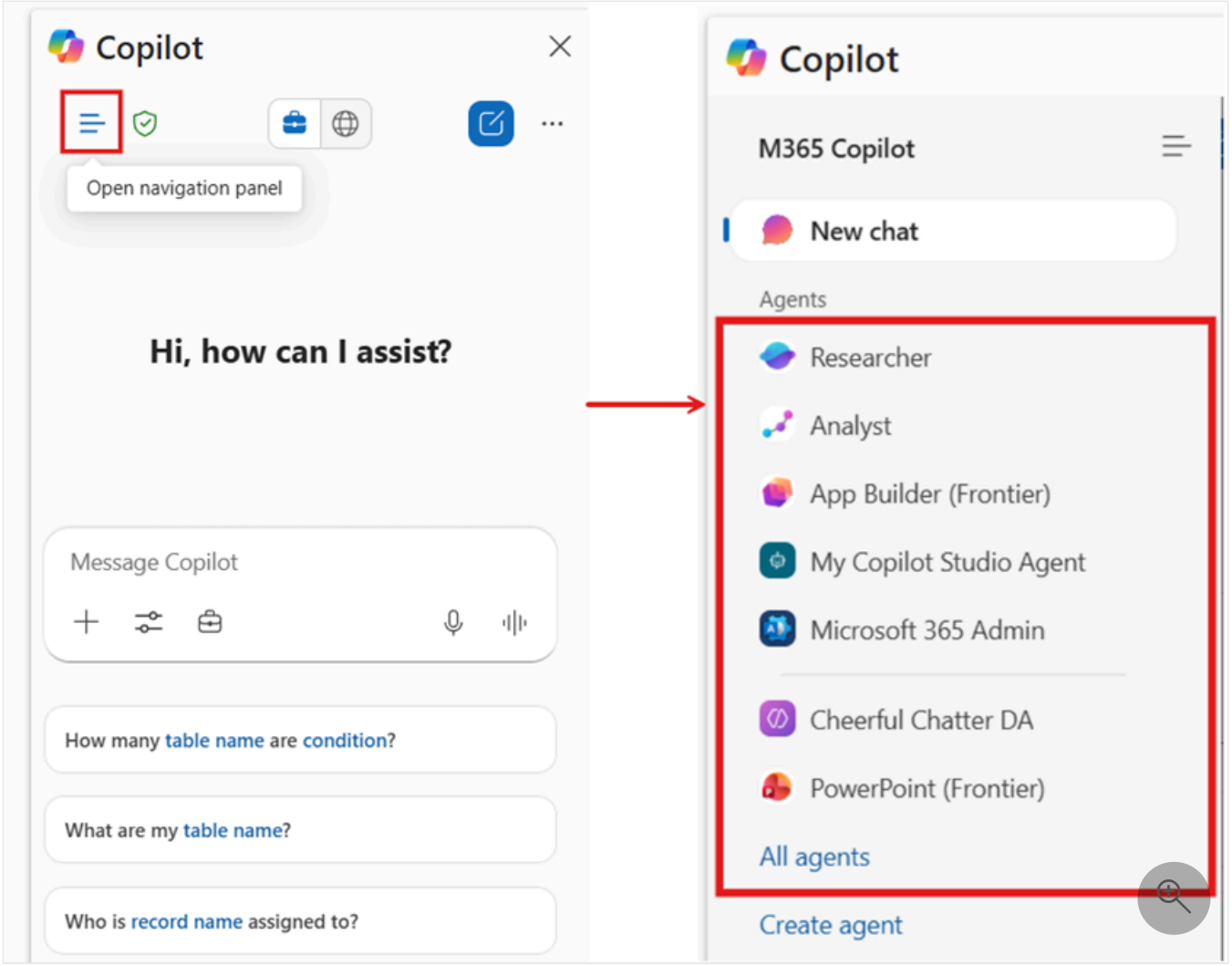
To help you get started, Microsoft 365 Copilot suggests questions to ask. Many suggested questions have placeholders you replace with appropriate text.



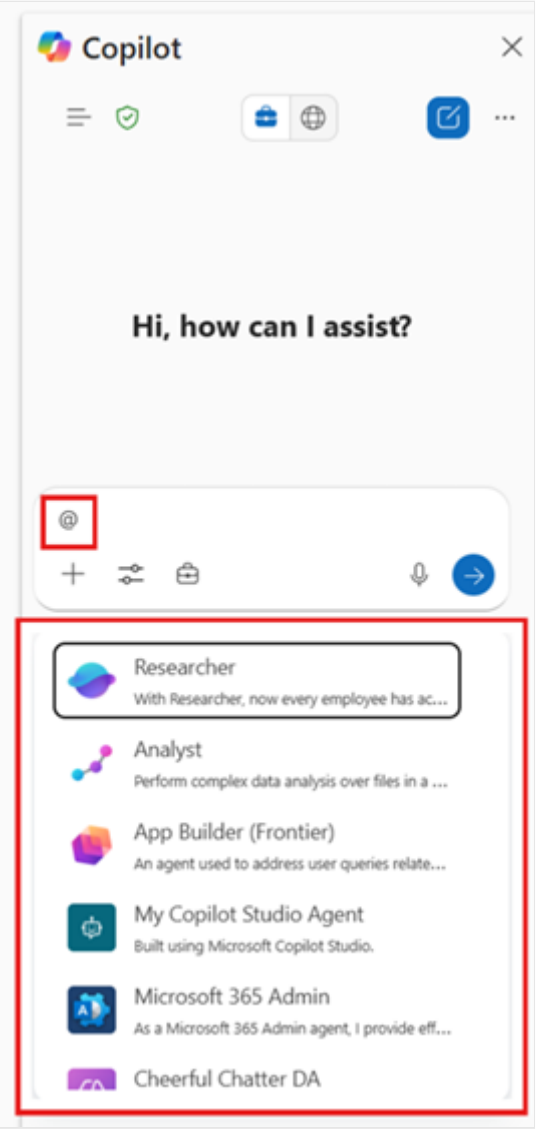
Use agents in Microsoft 365 Copilot

> **ⓘ Important**
>
> When you explicitly select an agent in Microsoft 365 Copilot in apps, the agent no longer answers questions about the Dataverse table or SharePoint list data in the app unless you explicitly configure your agent to do so. However, the agent uses information from the chat history. To get the side chat to answer questions about the Dataverse table or SharePoint list data in the app again, you need to remove the explicit agent selection.

Microsoft 365 Copilot in apps lets you use any agent available in Microsoft 365 Copilot right from the side pane. Once an agent is available within Microsoft 365 Copilot, you can interact with your agent by either choosing it from the navigation panel or @ mentioning it.



One of the benefits of @ mentioning an agent is that you can add or remove it from an ongoing conversation, which lets you have several agents collaborating in one conversation. When you select an agent from the navigation panel, you have a direct conversation only with the agent you selected.



Limitations

- M365 Copilot in canvas apps supports apps that use either SharePoint or Dataverse as a data source but not both within the same app.
- During the initial preview phase, for SharePoint, only applications containing a single SharePoint list will return answers to questions.
- If your canvas app connects to a SharePoint list through an environment variable, Microsoft 365 Copilot responses will not reference data from that SharePoint list.
- Microsoft 365 Copilot for canvas apps allows users to view data by using read-only operations. This capability means that users can only view data that matches their queries and can't make any changes. To make changes, customization with an agent is required.
- Microsoft 365 Copilot for canvas apps isn't available in the Power Apps mobile app.
- As this feature is being gradually deployed, certain settings in the Power Platform Admin Center might not be accessible yet, depending on the geographic location of your tenant.

Related information

- Customize Microsoft 365 Copilot with an agent
- Add Microsoft 365 Copilot for app users in canvas apps (preview)

Last updated on 04/15/2026

Run a canvas app from a SharePoint page

Article • 09/05/2024

You can add a Power Apps canvas app to a SharePoint Online page using the Power Apps web part. Then users can run the app from the SharePoint Online page. Learn more in Office: Use the Power Apps web part.

Feedback

Was this page helpful? Yes No

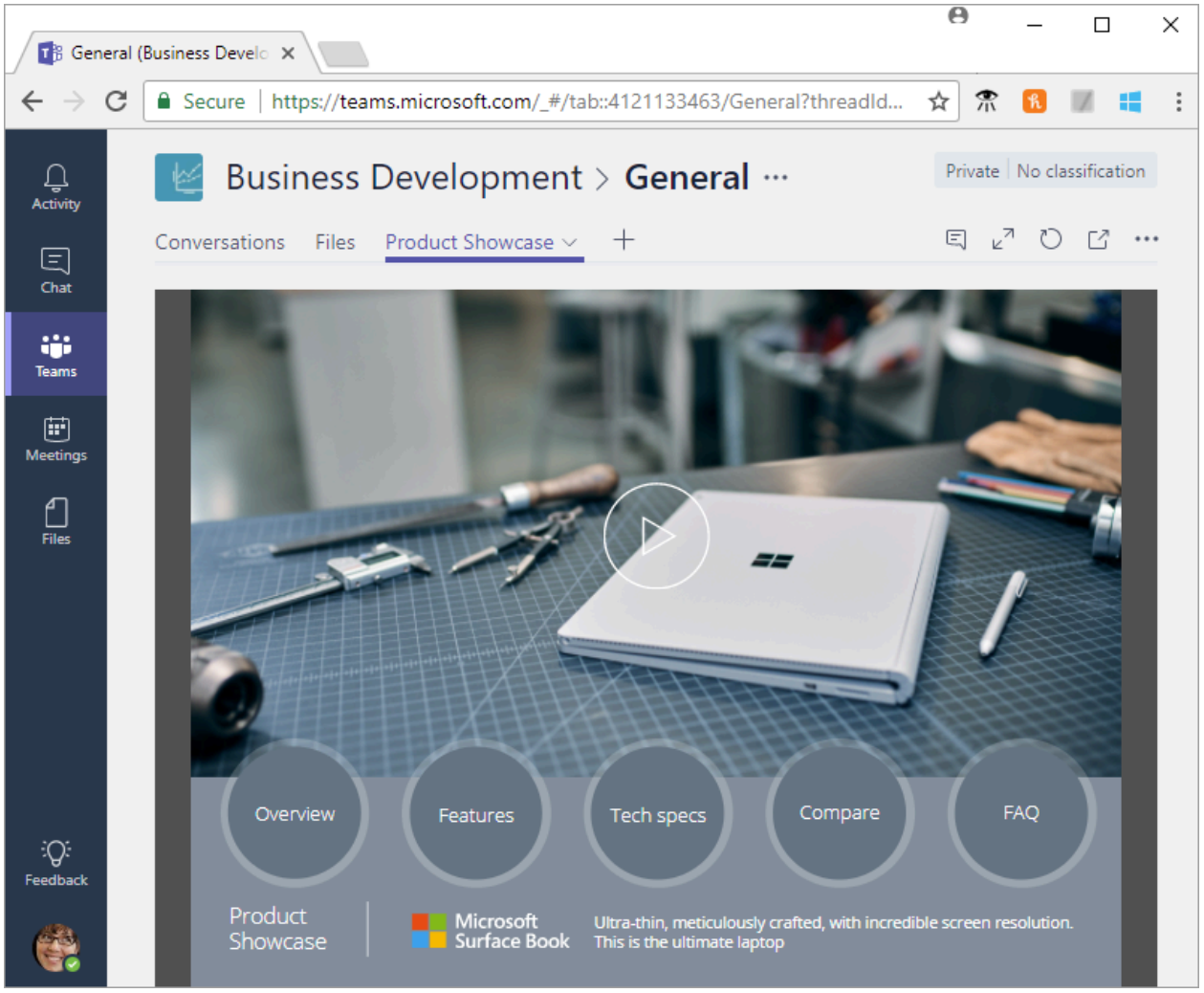
Provide product feedback

Add an app to Microsoft Teams

Article • 02/15/2022

Microsoft Teams is a chat-based collaboration platform built on Office 365 technologies. You can customize the Teams experience by adding Power Apps canvas apps to your channels in Teams.

For detailed information about how to add the Product Showcase sample app to a Teams channel and then open the app from the channel, see Embed a canvas app as a tab app in **Power Apps and Teams integration** section.



Feedback

Was this page helpful? Yes No

Provide product feedback

Use model-driven apps in Power Apps

06/30/2025

A model-driven app has multiple components including dashboards, forms, views, charts, and business processes that make the app easy to use. Model-driven apps can also include AI features that let users to supervise and interact with agents around business processes more efficiently and productively.

This section explains how to navigate in a model-driven app, work with rows, import and export data, and automate processes by using Power Automate.

> **Important**
>
> Because canvas apps provide a lot of control over the user experience and can essentially be anything, there isn't much help documentation on how to *use a canvas app*. If you need help using a canvas app, contact your app maker or Power Apps admin.

Related articles

Enhanced user experience with the Unified Interface
Basic navigation in a model-driven app

See also

Find and run apps
What is a model-driven app?
What is a canvas app?

Modern, refreshed look for model-driven apps

Model-driven apps have updated styling including fonts, colors, borders, and shadows that align with the latest Microsoft Fluent design system. The updated look makes model-driven apps easier to use so that users can accomplish their goals quickly and efficiently. The Fluent design system provides consistency, quality, and Microsoft-wide platform coherence. It also provides a solid foundation for extensibility and allows support for dark mode in the future.

> **Important**
>
> With the **2026 Wave 1** release, all users must use the **New Look**. Makers can't switch a model-driven app back to the classic look. This requirement doesn't include the **Header and navigation refresh (preview)**.
>
> The **New look** toggle was removed for users with the **2025 Wave 1** release.

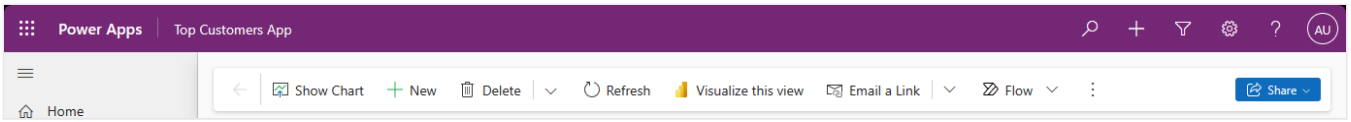
What's included in the modern, refreshed look

Here's what you can expect in the modern, refreshed experience:

- Updated styling in form, view, and dashboard pages, which includes the use of drop shadows and brighter background colors to create an elevated or *floating* appearance. The floating appearance helps to visually separate sections and focuses attention on primary content.
- New Fluent-based controls in forms, business process flows, and dialogs. Dialogs now resize height automatically based on the content.
- A new Power Apps grid in place of the read-only grid in view and standard, dashboard pages.
- A new mechanism for customizing the app header colors to match your personal or organizational branding. To learn more, see Use modern themes.

Command bar

The *floating* command bar aligns with the Microsoft 365 experience, with consistent spacing, rounded corners, and elevation. Notice how the command bar is in a separate section at the top of the page in the following example.

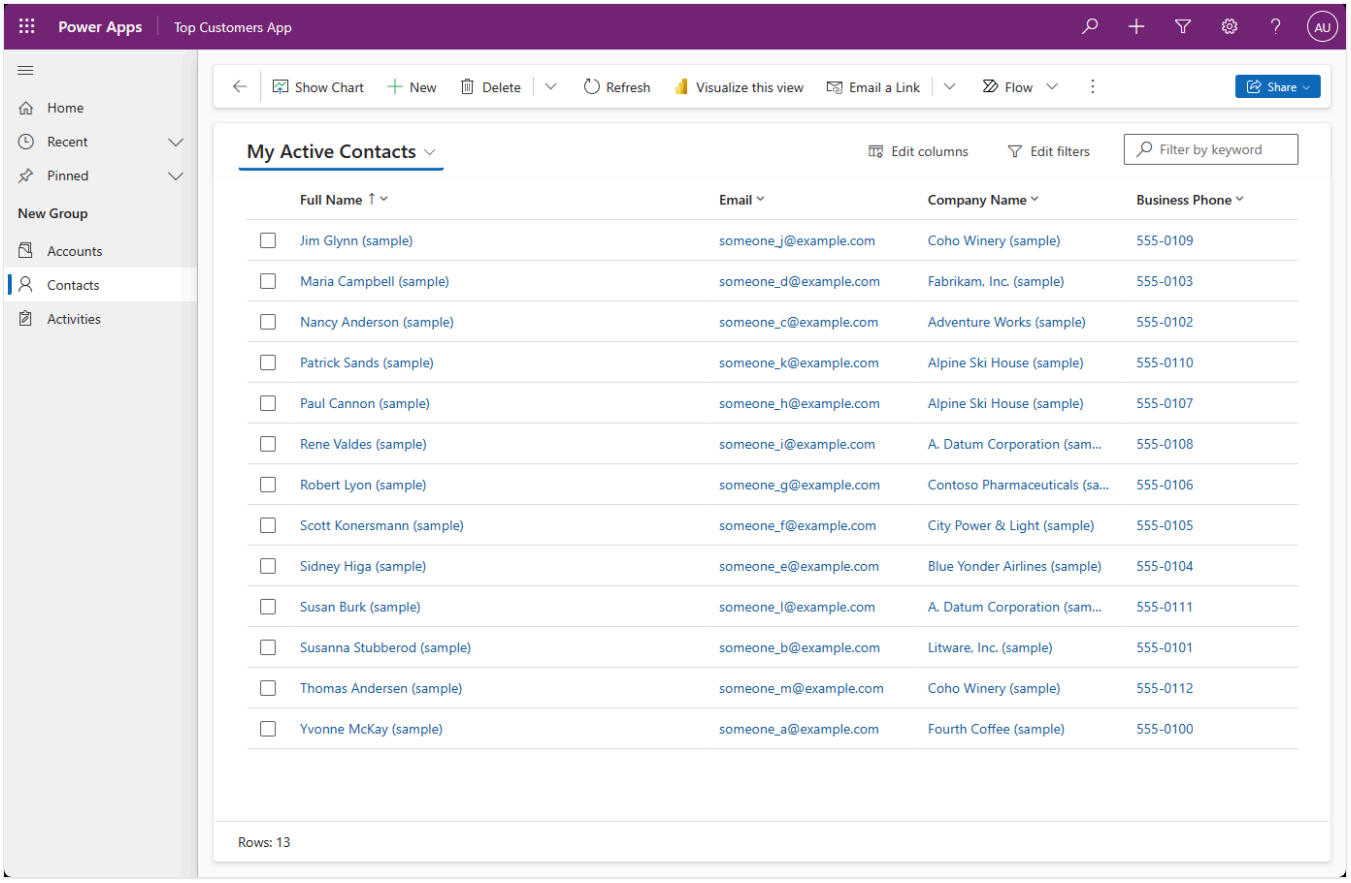


View pages

View pages use the new command bar and updated grid areas that take advantage of the elevation changes to help draw the user's attention.

The most significant change on view pages is the switch from the read-only grid to the Power Apps grid control, which features infinite scrolling for a modern, data browsing experience. This grid also appears in subgrids and associated grids in main forms and dashboards. The Power Apps grid control also supports inline editing by using the **Enable filtering** property. Makers can manually configure their editable grids to use the Power Apps grid control.

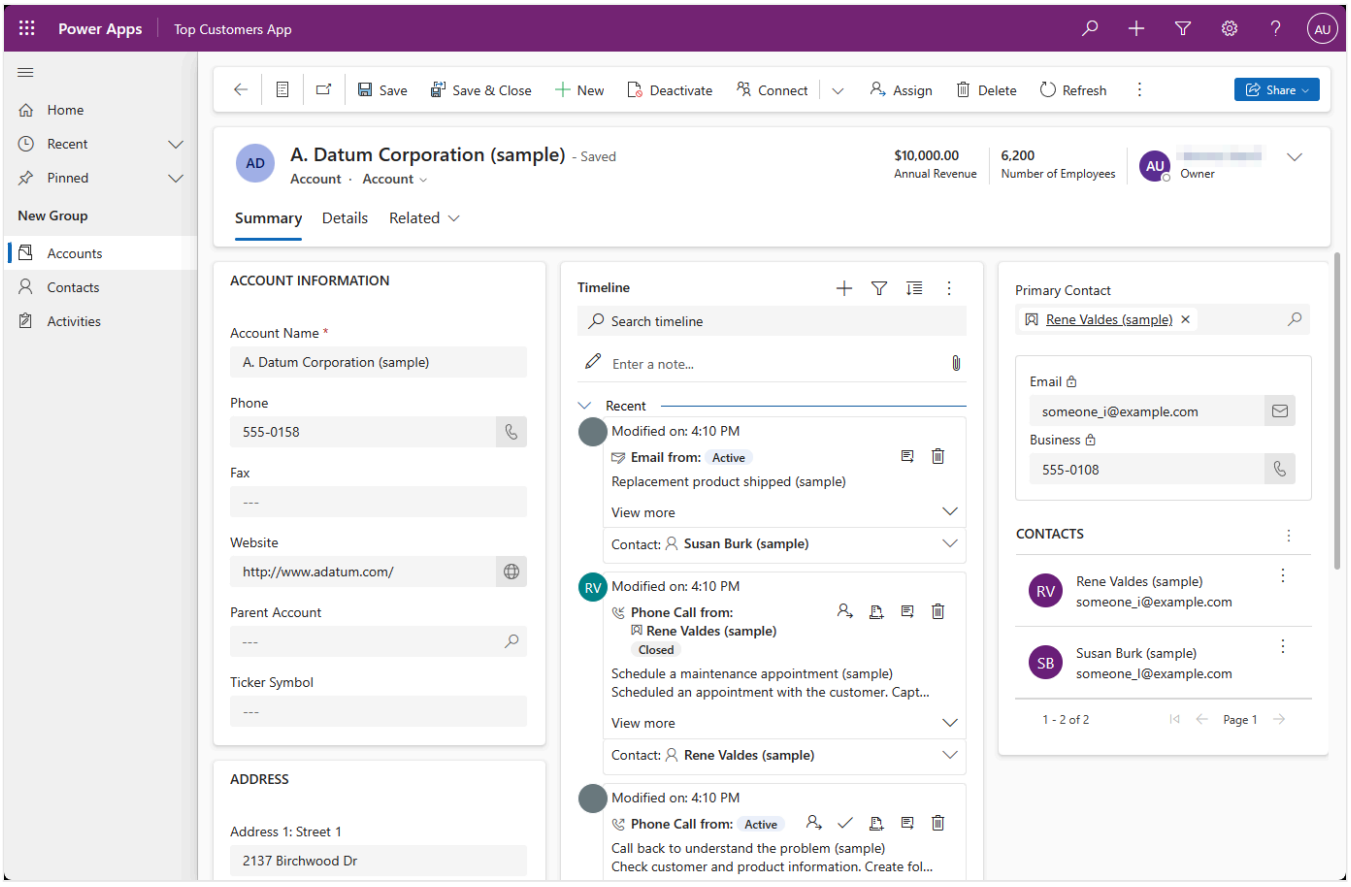
The following example shows a view page with the modern, refreshed look.



Form pages

Form pages use the new command bar with refreshed headers, tabs, sections, and business process flows. Quick views, card forms, headers, sitemaps, and timeline controls also feature updated styling.

The following example shows a form page with the modern, refreshed look.

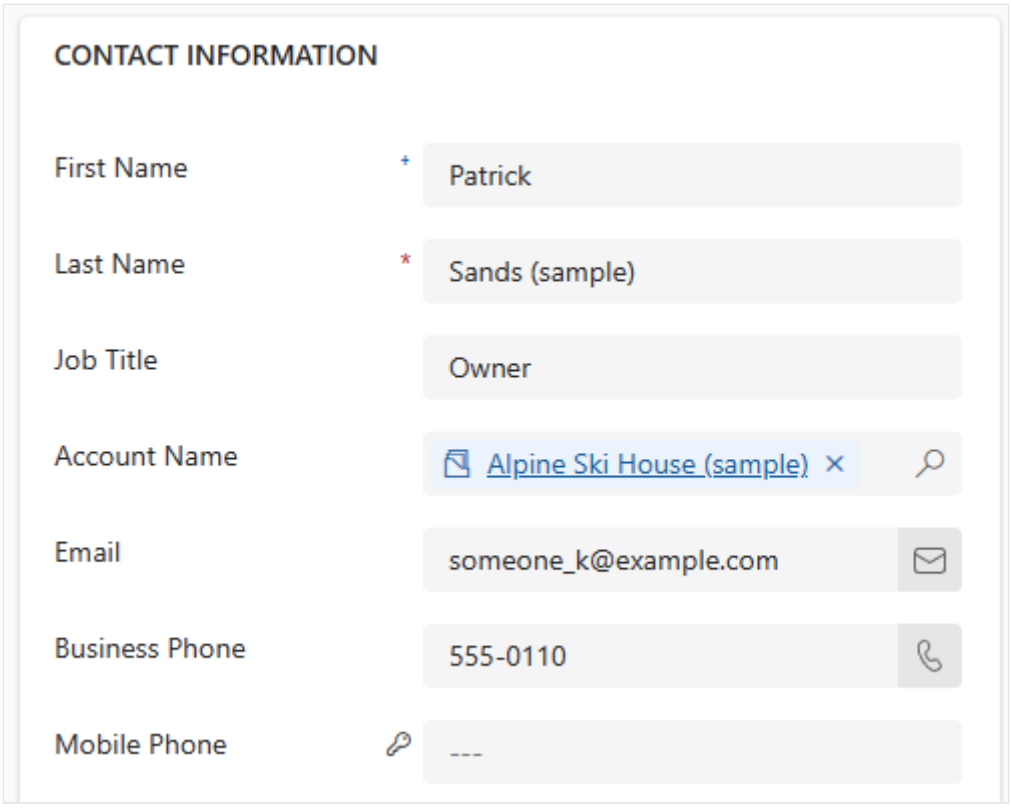


Field controls

Field controls such as text input, action input, lookup, and check box controls are built and designed by using Fluent components. Future updates will modernize more field controls by using Fluent design.

Field sections, which are containers for the fields on a form, have a more streamlined design. Icons are now on the right side of field labels. Some redundant icons were removed for a cleaner layout. Input and error message styling is also refreshed based on Fluent design.

The following example shows a set of fields with the modern, refreshed look.



Dashboard page

The system dashboard page, with independent subgrids and charts, uses the new command bar and has styling similar to the sections in form and view pages. When you turn on the modern experience, or when you use the monthly channel, the system dashboard grids use the new Power Apps grid control. The new grid isn't turned on by default.

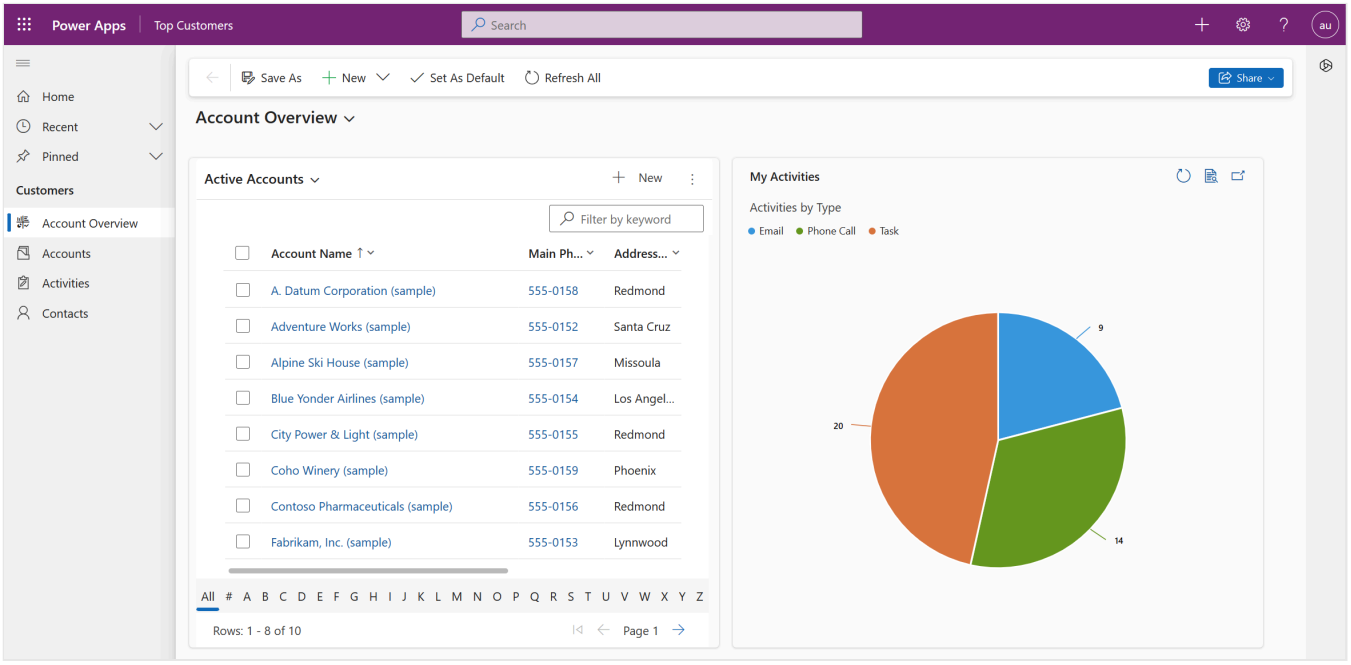


Chart controls

The chart controls on view pages and dashboards are updated to a new color palette for the modern, refreshed look.

If you customize the chart colors, the new look overrides your custom colors. To keep your custom colors, add the **CustomColorOverride** property in the chart .xml file.

XML

```
<Chart CustomColorOverride="true">
```

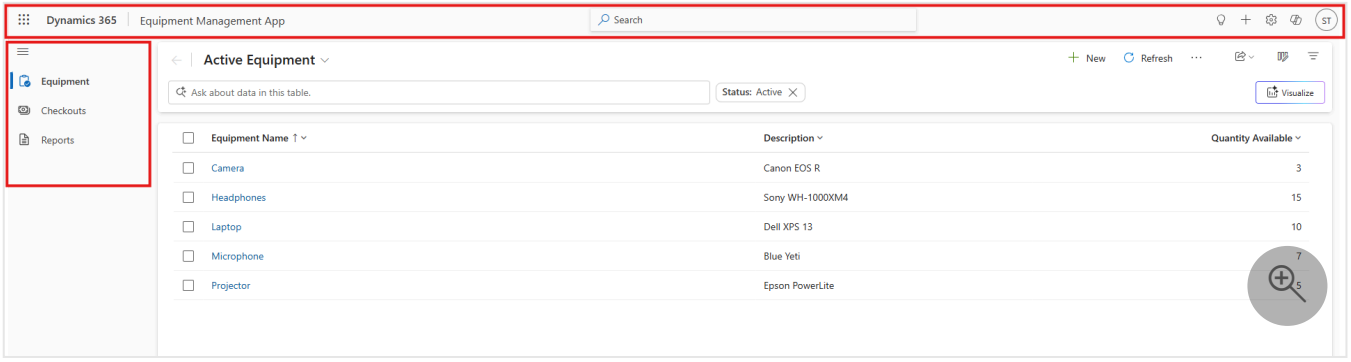
Header and navigation refresh (preview)

[This topic is pre-release documentation and is subject to change.]

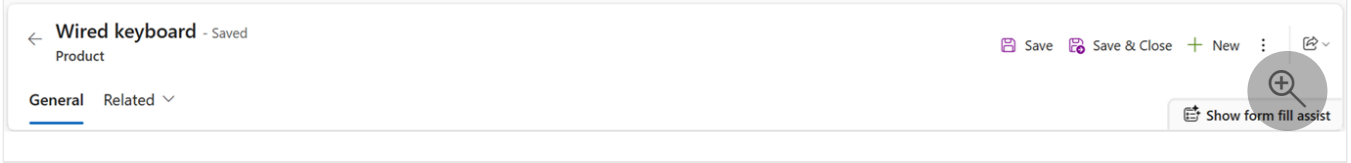
> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

The refreshed header and navigation make apps easier to use with a modern app header, streamlined sitemap, and condensed page headers with compact commands. This experience enhances productivity by reducing time spent navigating pages, finding commands, and learning layouts.

Here's how the new app header and the streamlined sitemap look:

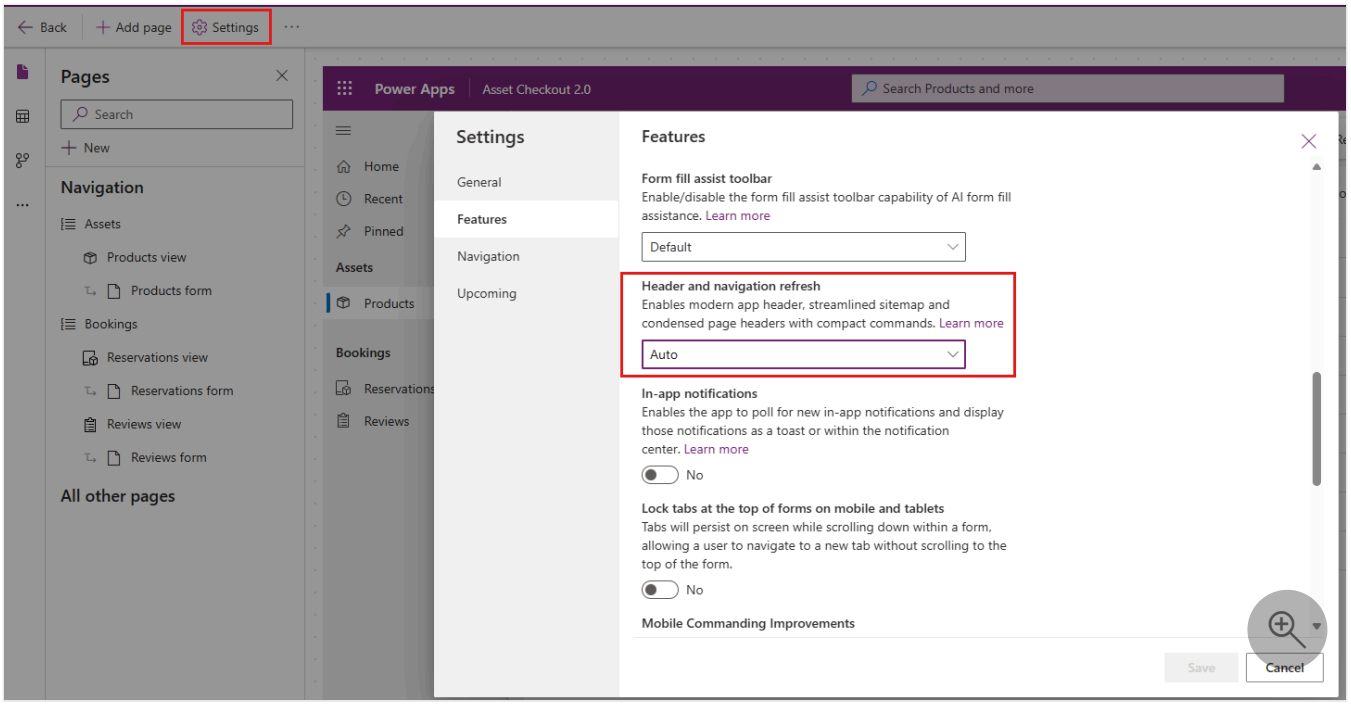


Here's how the new form header looks with more focus on row details:



This update automatically applies to all apps generated by the Plan designer. For other existing apps, you can opt in by following these steps:

1. In the app designer, select **Settings** on the command bar.
2. Select **Features**, and then enable **Header and navigation refresh**.



Learn more in FAQ for header and navigation refresh (preview).

Known limitations

The modern, refreshed look for model-driven apps has some limitations:

Modern, refreshed look outside of browser

The mobile app and mail app don't support the modern, refreshed look. They aren't part of the preview or general availability.

Switching themes or enabling dark mode

Switching themes or enabling dark mode isn't supported at this time.

Honoring classic theming

With the modern, refreshed look, Power Apps is no longer honoring classic theme customizations. You can, however, override the colors for the app header to match your organization branding with the modern, refreshed look. Learn more in Change the color of the

app header. Other theme customization options for the modern, refreshed look aren't available yet.

Custom icons

Only SVG icons are supported. If you use other formats such as PNG, the navigation doesn't display them and a default icon appears instead.

Frequently asked questions (FAQs)

1. Can I use the modern, refreshed look in the Power Apps component framework or in custom code components?

The modern theme in use is passed to Power Apps component framework components, so you can style your components with modern theming.

2. Can I use the modern, refreshed look in custom pages?

You can use modern controls with custom pages and the modern, refreshed look. Currently, custom pages don't use the modern theme.

3. Can I change the color of the app header?

You can change the color of the app header to match your organization while using the modern, refreshed look. To learn more, see Use modern themes. Other theme customization options for the modern, refreshed look aren't available yet.

4. FAQ for header and navigation refresh (preview)

Can I turn on Header and navigation refresh without turning on the New look?

No, you need to turn on New look in addition to turning on Header and navigation refresh to see the refreshed header and navigation experience in the app.

How do I enable the Header and navigation refresh for existing apps?

The feature is available as an opt-in capability for existing apps. You can enable it through the app setting.

Why don't I see the Help icon in the global command bar in the app header?

Help link to documentation is available in the **Settings** pane.

Where is the side pane rail next to the Copilot chat pane?

The side pane switcher becomes available only when there are two or more side panes in the app. When Copilot chat is the only side pane, the switcher is hidden.

Why does the sitemap of an app generated by the Plan designer look different from existing apps' sitemap?

Apps generated by the Plan designer will have a streamlined sitemap. Home, Recent, Pinned and sitemap groups are turned off by default for a simplified flat list of navigation options in the sitemap.

Revert to the classic look

> **Note**
>
> With the **2026 Wave 1** release, the **New Look** is mandatory for all users. Makers will no longer be able to switch a model-driven app to the classic look.

While end users have not been able to switch themselves back to the classic look since the **2025 Wave 1** release, admins can still turn off the new look for all users by updating the **New look for model driven apps** app setting.

1. Open https://make.powerapps.com/.
2. Under **Solutions**, open an existing solution with one or more model-driven apps.
3. Select **Add Existing** > **More** > **Setting**.
4. Search for **New look**.
5. Select **New look for model driven apps**.
6. Select **Add** to add it to the solution.
7. Select **New look for model driven apps** from the solution explorer.
8. Update **Setting Environment Value** to **No**.
9. Add the **apps to the solution** for which you want to turn off new look.
10. Update the **value of the apps** to **No**.
11. Select **Save**.
12. Publish all customizations.

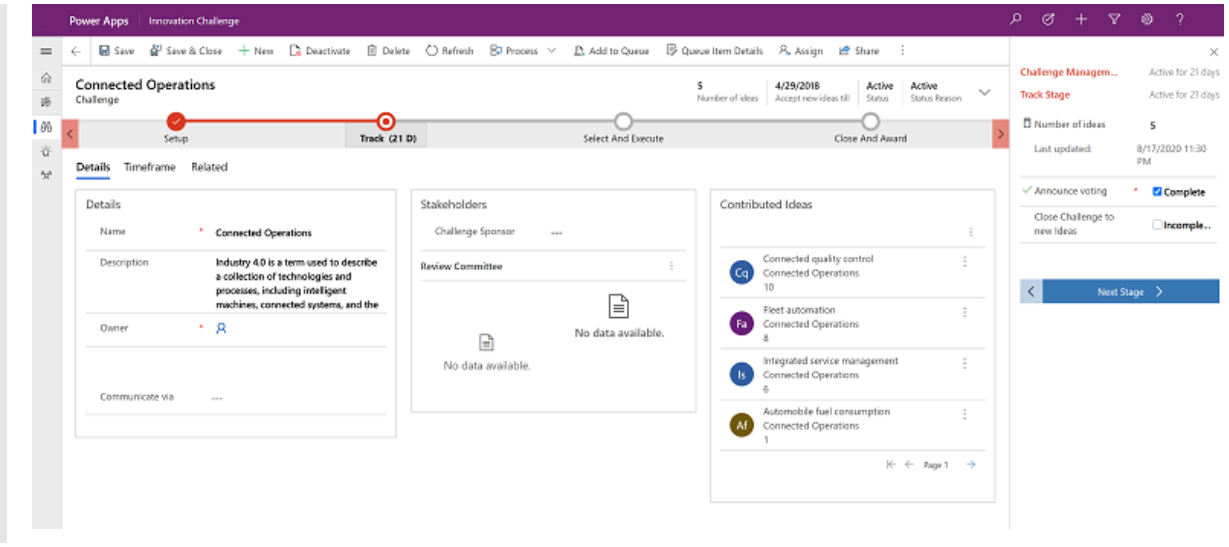
To learn more about this and other app settings, see Manage model-driven app settings in the app designer.

Last updated on 03/24/2026

Enhanced user experience with Unified Interface for model-driven apps

Article • 06/27/2023

Unified Interface for model-driven apps provides a consistent and accessible user experience across devices—whether on a desktop, laptop, tablet, or phone. The apps scale by reflowing the components on the screen. The responsive design adapts to your environment based on screen size, so the more available space that you have, the more information can be displayed.



For an overview of Unified Interface in model-driven apps, watch this video: Introduction to Unified Interface

Reflow

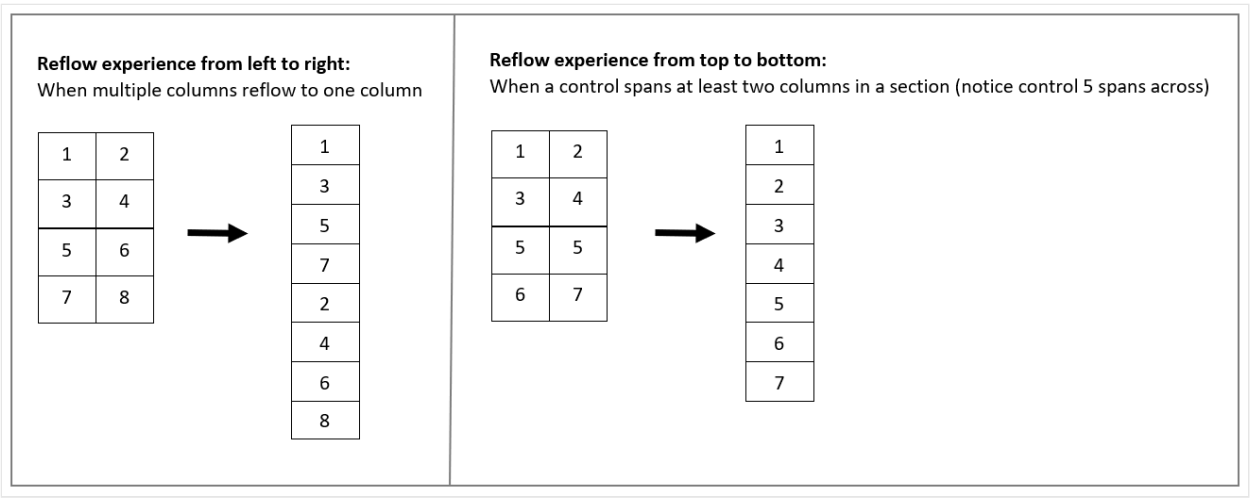
Unified Interface is designed for a modern reflow experience.

For forms, there are pixel breakpoints that cause information on a form to move on the screen based on the space available for the form to display in. This happens when the display space available to the form decreases. For example, if you resize the window that your app is in and decrease the width of the window it will decrease the space available for the form to display in. When the width decreases past certain breakpoints, information that would be hidden will move below where it used to be so that it is either visible or you can vertically scroll down to see it.

When columns in a form section reflows from multiple columns to one column, it reflows from top to bottom. When a control spans at least two columns in a section,

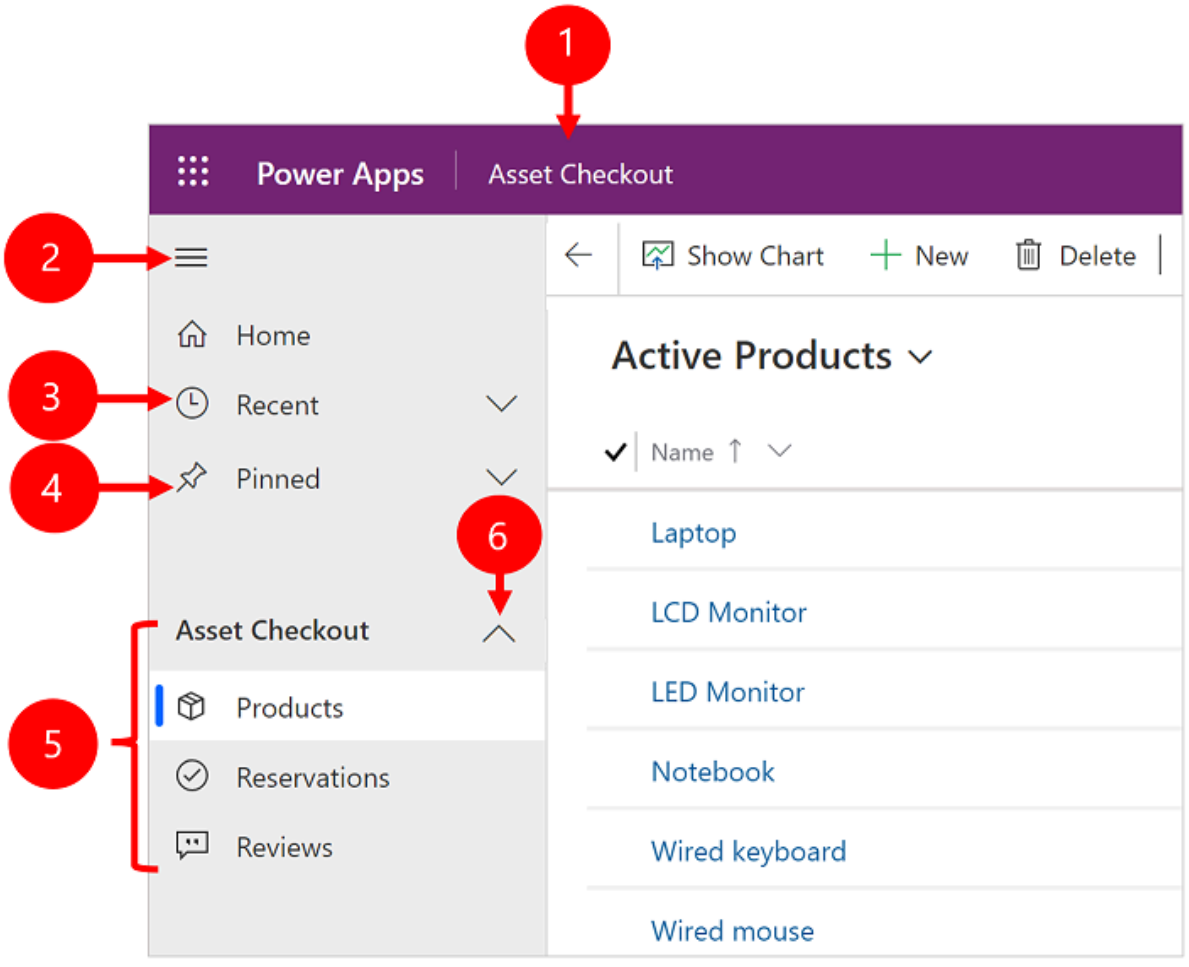
then it reflows from left to right (in left-to-right languages). The difference in behavior is so that tab navigation order is kept after reflow.

In example below, each number corresponds to a control.



Navigation

The menu options let you swiftly navigate the different apps in the system. They provide quick access to recently viewed rows and pinned favorites.



Legend

1. **Click to change app**: Select the current app name to change a differ app.
2. **Collapse/expand button**: Select this to collapse the navigator to allow more room for the main part of the page. If the navigator is already collapsed, select this button to expand it again.
3. **Recent rows**: Expand this entry to view a list of rows you were recently using. Select a row here to open it. Select the push-pin icon next to a row listed here to add it to your favorites (pinned rows).
4. **Favorite rows**: Expand this entry to view and open your favorite (pinned) rows. Use the **Recent rows** list to add rows here. Select the remove-pin icon next to a row listed here to remove it from this list.
5. **Page navigation**: This area lists each entity and dashboard page available for the current work area. Select any entry here to open the named dashboard or list view for that entity.

For more information, see Basic navigation in a model-driven app.

Dashboards and charts

You can access all the system and user dashboards from within your Unified Interface apps. The interactive dashboards are now available for all row types with richer interactive dashboard capabilities. For more information, see Track your progress with dashboards and charts.

Timeline control

The timeline view helps you collaborate with your team by tracking customer communication in a row on a single page in an easy-to-read view. You can see everything from posts and voice attachments, to emails and notes. It provides a quick way to see the entire communication thread. For more information, see Add an appointment, email, phone call, notes or task activity to the timeline.

Business process

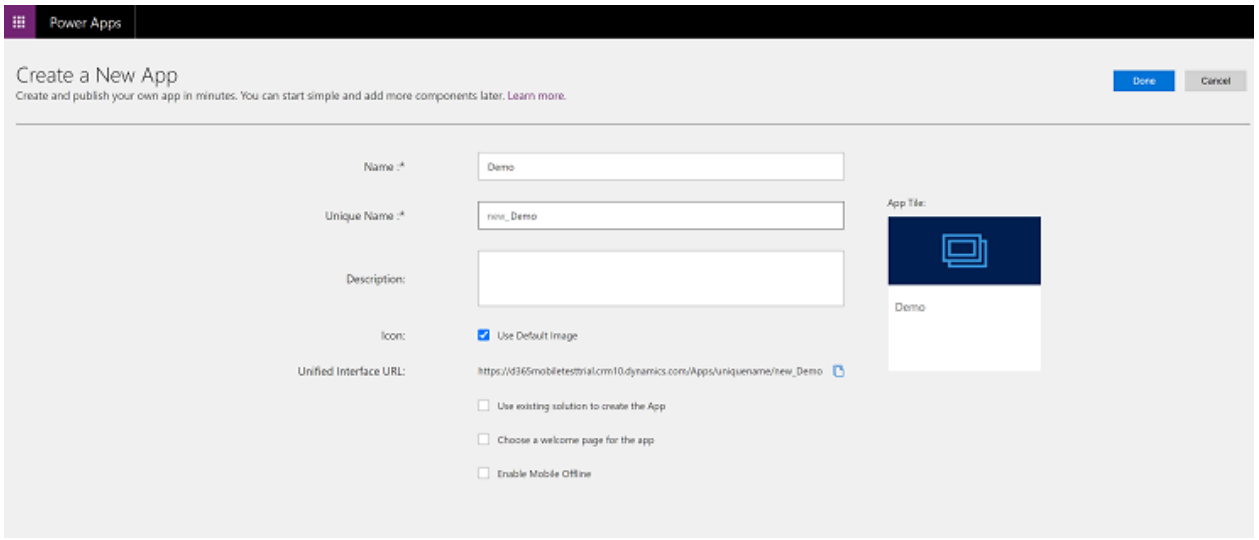
The business process flow has been improved by the docking mechanism. You can dock the business process stage on your screen to help you stay focused on the task at hand in your business process flow. This is especially useful when the stage of the process has complex steps to complete. For more information, see Work with business processes.

Accessibility

The improved accessibility experience lets you use screen readers to translate on-screen information into audible sound and print to a Braille reader so that more people can use the app. For more information, see Use a screen reader.

Create a Unified Interface app

If you have requirements to create your own experience on Unified Interface, you can create a model-driven app using the app designer. See Overview of building model-driven apps.



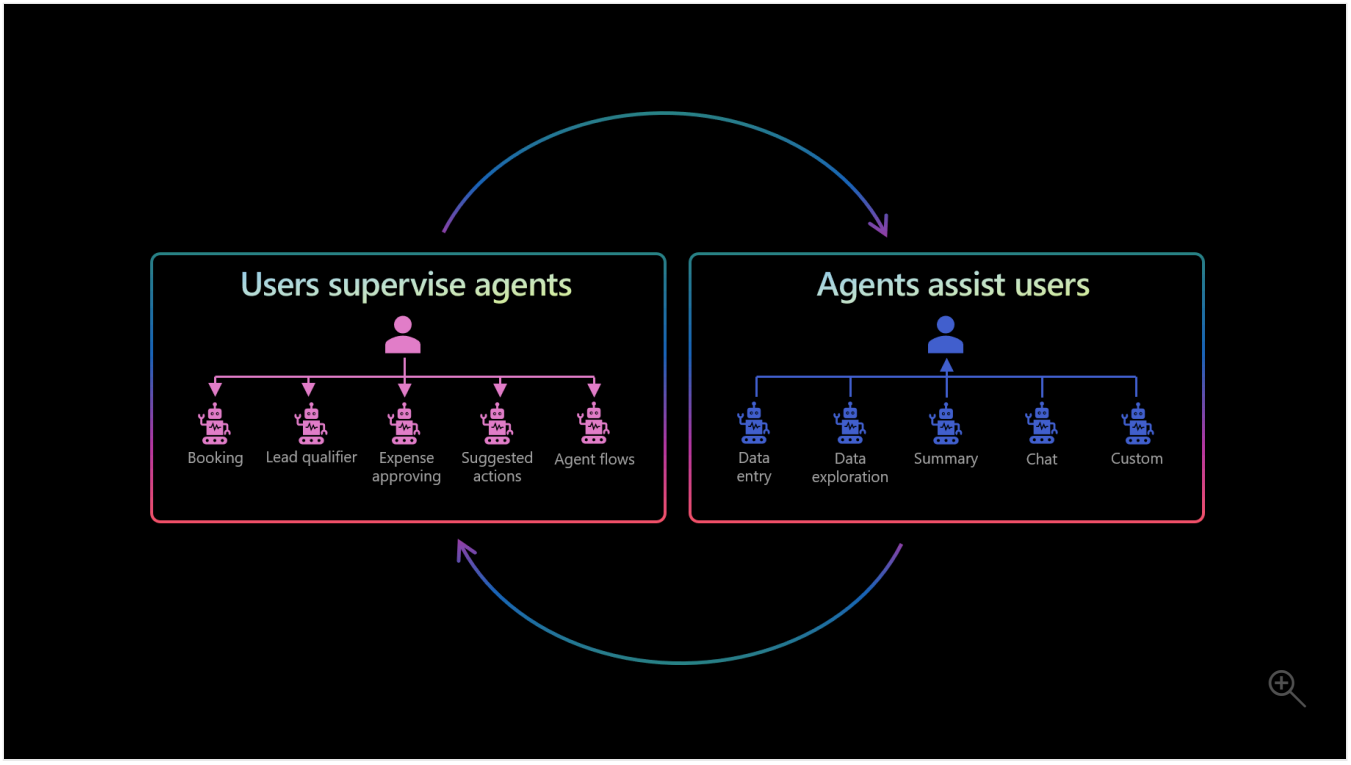
Feedback

Was this page helpful? Yes No

Provide product feedback

AI features overview

AI features in apps combine model-driven apps with intelligent automation and assistance. Business users can supervise their team of AI agents doing work on their behalf, spot-checking accuracy, intervening when they get blocked, and handle exceptions. Users get inline assistance from agents, fill out forms faster, explore data more easily, and understand what changed at a glance. AI in apps lets users supervise and interact with agents around business processes for more efficiency and productivity.



Prerequisites

Copilot needs to be enabled for your tenant and environment, and apps need to have the new look turned on. AI features also have their own enablement settings.

Agent supervision

Agent feed helps business users manage and work with agents directly in their apps. It makes apps a central hub for human-agent collaboration by showing a complete activity feed of what agents have done on their behalf, wherever they are in the app.

- Add agents to be supervised in app
- Supervise agents in app

AI in apps

Built-in skills help you enter, explore, and summarize data faster and better.

Data entry

- Get assistance filling forms
- Smart paste into forms
- Draft richer text

Data exploration

- Explore data in a view
- Generate data visuals for a view

Summary

- Create custom row summaries
- See row summaries in a form and view

Copilot in apps

- Microsoft 365 Copilot in model-driven apps
- Customize Microsoft 365 Copilot in model-driven apps
- Chat with Copilot
- Customize Copilot chat

Custom AI experiences

- Use APIs to create custom AI-powered experiences
- Add an agent response control

Last updated on 04/15/2026

Use Microsoft 365 Copilot in model-driven apps

Microsoft 365 Copilot for apps is a next-generation AI assistant that helps you gain insights into the data in your apps through conversations in natural language. Microsoft 365 Copilot boosts your productivity through AI-powered insights and navigation assistance.

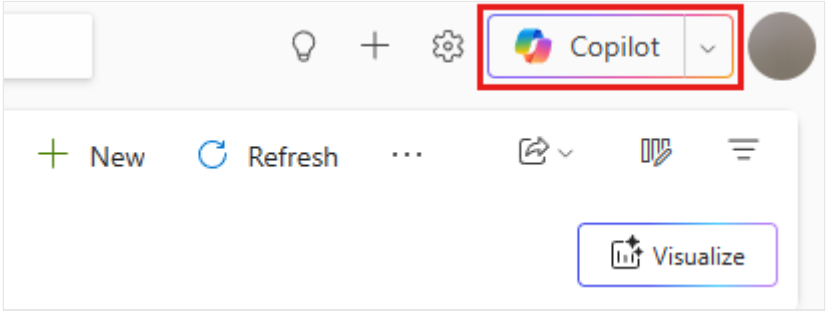
Prerequisites

An administrator must enable Microsoft 365 Copilot in your application before it becomes visible in your app. For more information, see Add Microsoft 365 Copilot for app users.

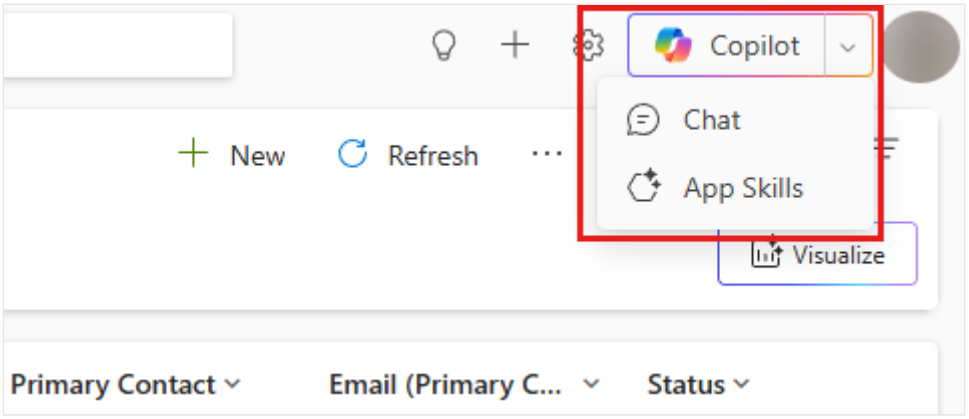
> **Note**
>
> To use the Microsoft 365 Copilot feature in your app, you must have both a Power Apps premium license and a Microsoft 365 Copilot license.

Copilot pane

After Microsoft 365 Copilot is enabled, access it through the Copilot button near the upper-right corner of the page.



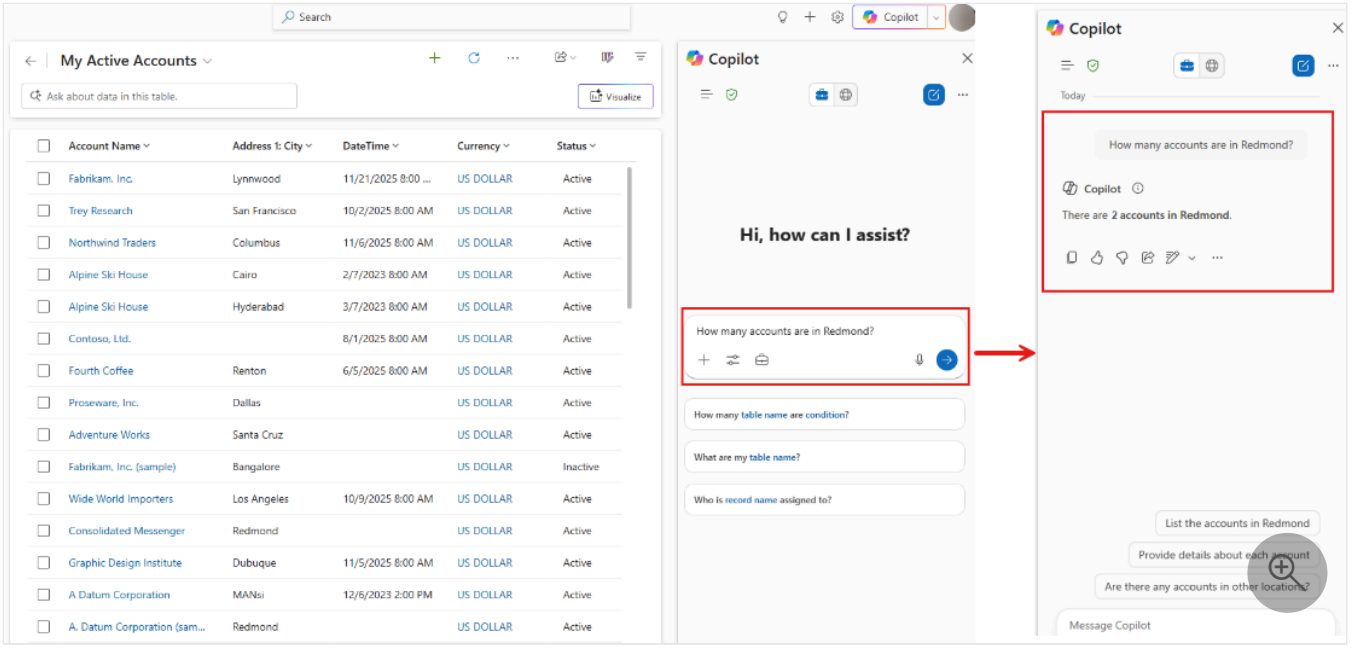
If both Microsoft 365 Copilot and Copilot chat in apps are enabled, you can access and try each option. The **Chat** option opens Microsoft 365 Copilot, and the **App Skills** option opens Copilot chat in apps. These terms align with Microsoft 365 apps for consistency.



Expand or collapse the Microsoft 365 Copilot pane as needed.

Use Microsoft 365 Copilot to ask questions

Microsoft 365 Copilot in apps answers questions about the Dataverse table data in the app.



Share your feedback

Help us improve Copilot responses by using the feedback controls on each response. Provide feedback for every response:

- If a response is high quality and helpful, select the thumbs up button.
- If a response is incorrect, incomplete, or not helpful, select the thumbs down button.

When you provide feedback, share as much information as possible. For example, include what you expected to see, what was missing or incorrect, and any relevant context. The more detail you share, the better we can understand your feedback and continuously improve responses.

Copilot

Today

Show me all the accounts in Redmond.

Copilot

Here are **all the accounts in Redmond** that I found in your app. There are **3 accounts total** that have their city set to *Redmond*:

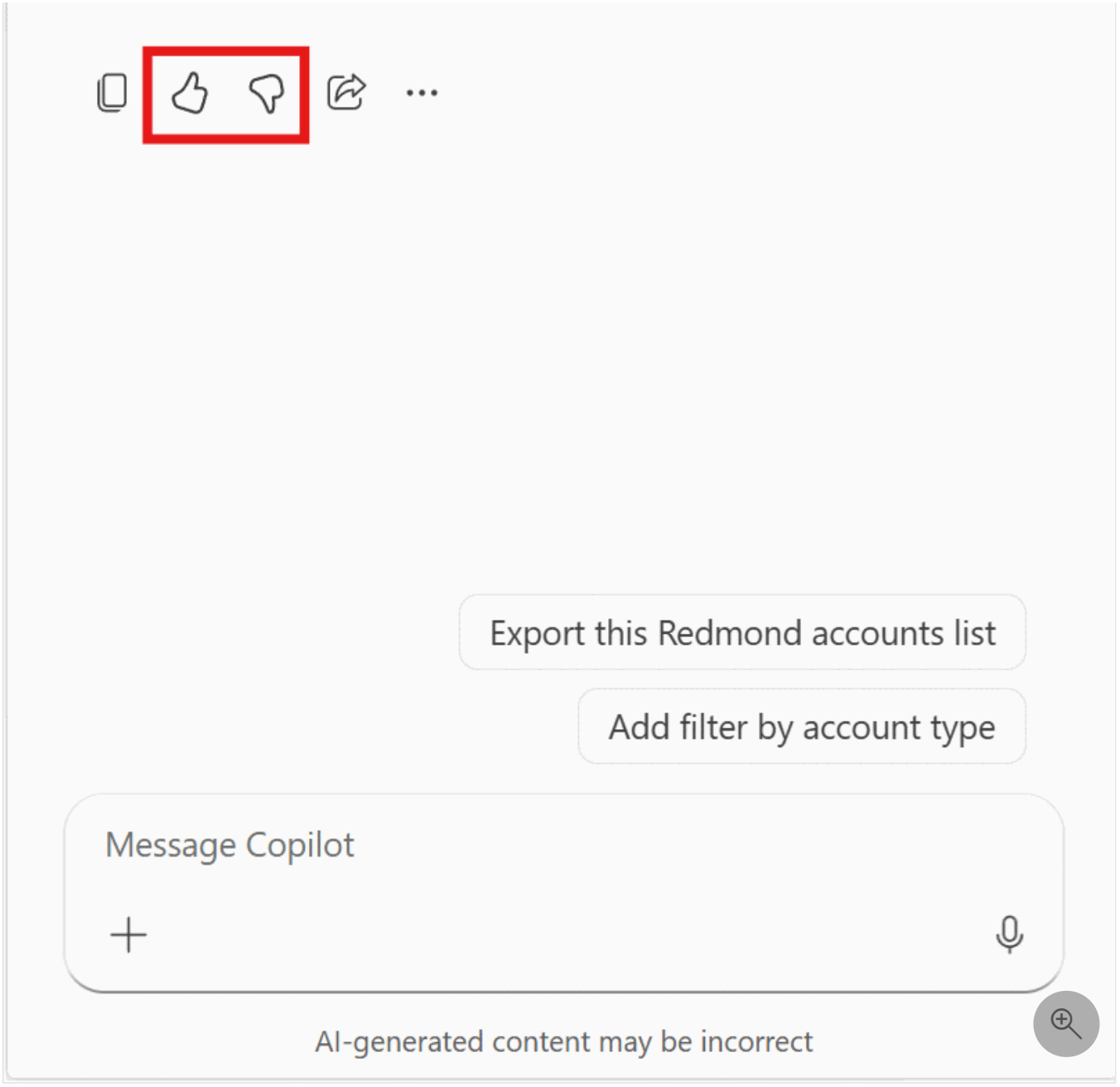
Accounts in Redmond

Account Name	City
City Power & Light (sample)	Redmond
Contoso Pharmaceuticals (sample)	Redmond
A. Datum Corporation (sample)	Redmond

If you'd like, I can:

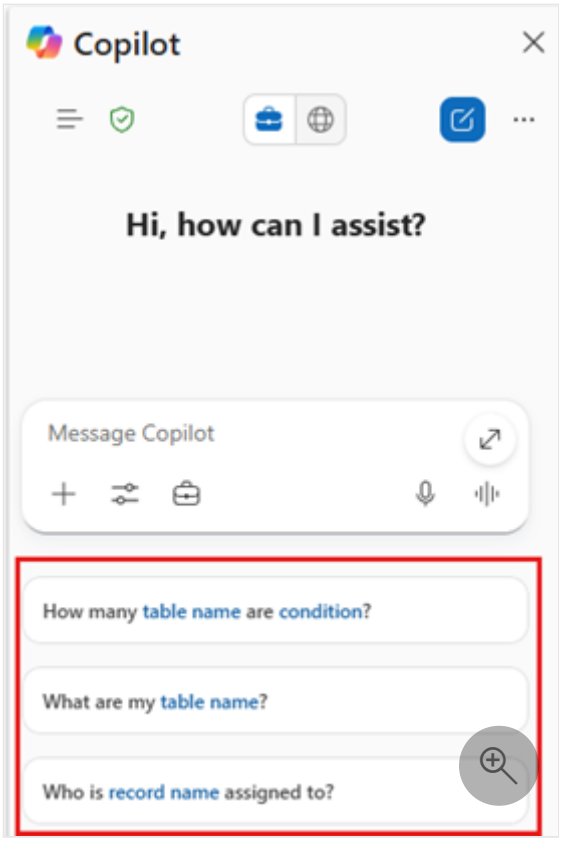
- Open any of these accounts
- Export this list
- Add additional filters (state, owner, account type, etc.)
- Turn this into a saved view for quick access

Just let me know how you want to use this.



Microsoft 365 Copilot suggested questions

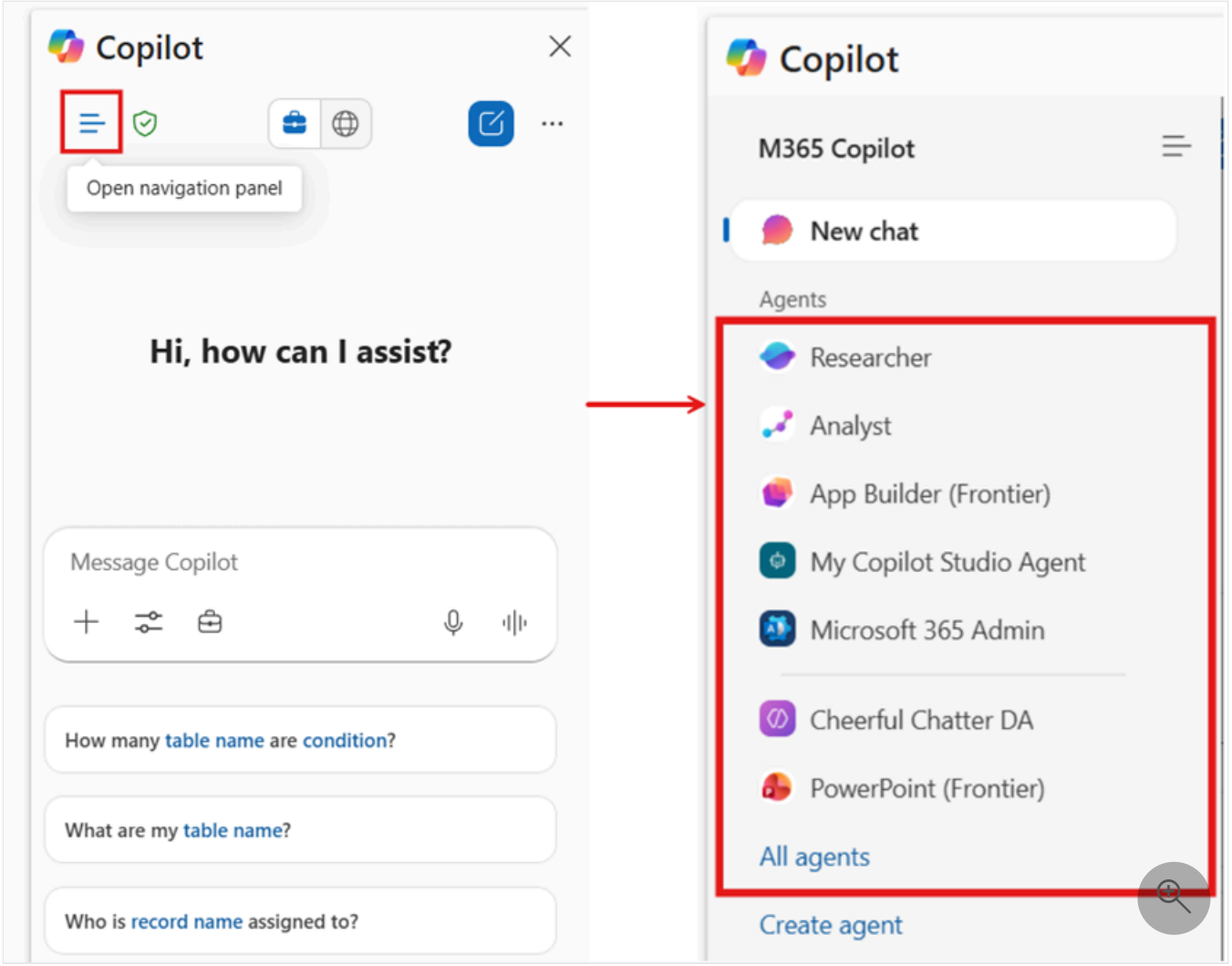
To help you get started, Microsoft 365 Copilot suggests questions to ask. Many suggested questions have placeholders you replace with appropriate text.



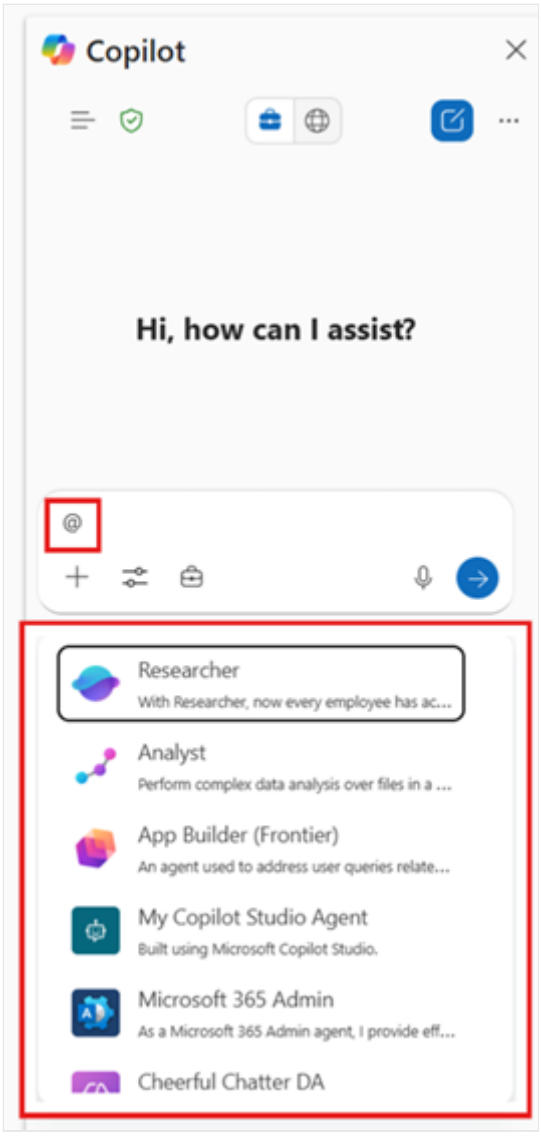
Use agents in Microsoft 365 Copilot

> **Important**
>
> When you explicitly select an agent in Microsoft 365 Copilot in apps, the agent no longer answers questions about the Dataverse table data in the app unless you explicitly configure your agent to do so. However, the agent uses information from the chat history. To get the side chat to answer questions about the Dataverse table data in the app again, you need to remove the explicit agent selection.

Microsoft 365 Copilot in apps lets you use any agent available in Microsoft 365 Copilot right from the side pane. Once an agent is available within Microsoft 365 Copilot, you can interact with your agent by either choosing it from the navigation panel or @ mentioning it.



One of the benefits of @ mentioning an agent is that you can add or remove it from an ongoing conversation, which lets you have several agents collaborating in one conversation. When you select an agent from the navigation panel, you have a direct conversation only with the agent you selected.



Limitations

- Microsoft 365 Copilot for model-driven apps allows users to view data by using read-only operations. This capability means that users can only view data that matches their queries and can't make any changes. To make changes, customization with an agent is required.
- Microsoft 365 Copilot for model-driven apps isn't available in the Power Apps mobile app.
- As this feature is being gradually deployed, certain settings in the Power Platform Admin Center might not be accessible yet, depending on the geographic location of your tenant.

Related information

- Add Microsoft 365 Copilot to model-driven apps
- Customize Microsoft 365 Copilot in model-driven apps

Last updated on 04/15/2026

Supervise agents in model-driven apps with agent feed (preview)

[This article is prerelease documentation and is subject to change.]

> **Important**
>
> Starting May 01, 2026, agent feed will support those agents only that use the Power Apps MCP server to create tasks. Make sure that your agents are properly onboarded to the Power Apps MCP server by this date to continue using the agent feed. If your agents don't use the Power Apps MCP server, the agent feed won't appear in your model-driven app.

Agent feed enables human-agent collaboration in apps by surfacing agent-generated, actionable tasks that users can review, validate, and complete. By using agent feed, apps act as a collaboration surface where users supervise and interact with agent work through a unified, task-based feed.

Prerequisite

The agent feed shows when at least one agent uses the Power Apps MCP server and is supervised in the app. Learn how to Add agents to an app.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is being gradually rolled out across regions and might not be available yet in your region.
> - This feature is only available in English language and replaces older Microsoft Copilot Studio activity-based agent feed version.

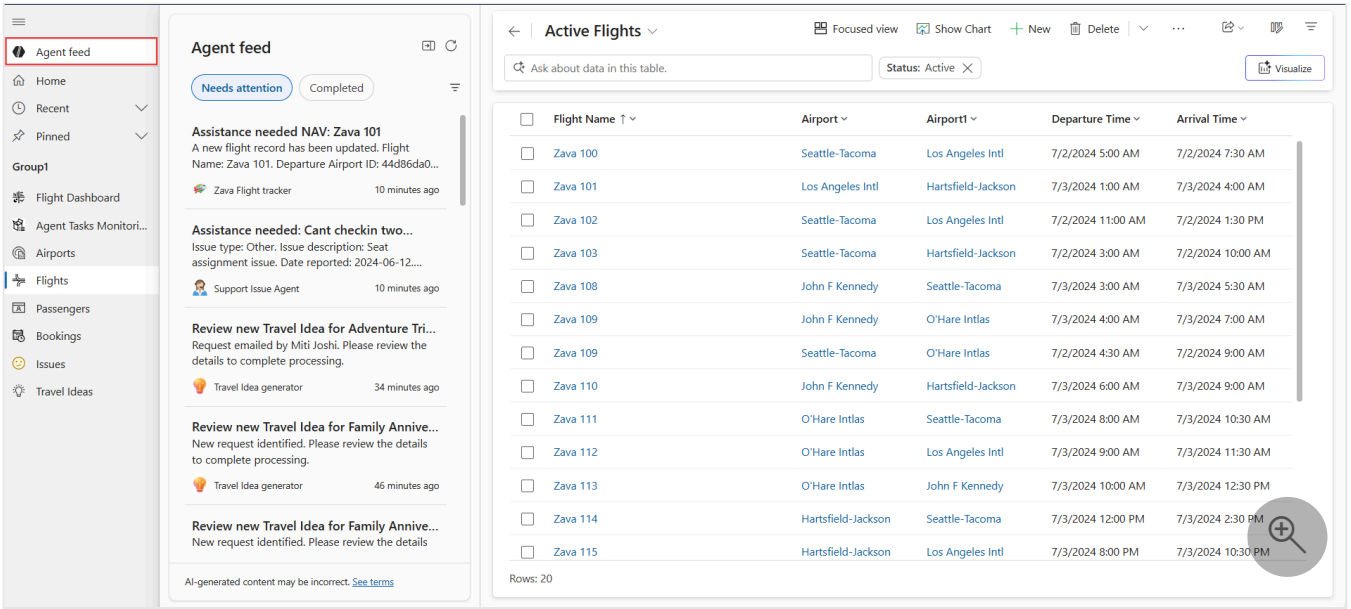
> **Note**
>
> Currently, all users who have access to the Agent Task table can see agent feed items in apps. To prevent unintended exposure, don't configure agents to log tasks targeted at specific users.

Use agent feed to supervise agents

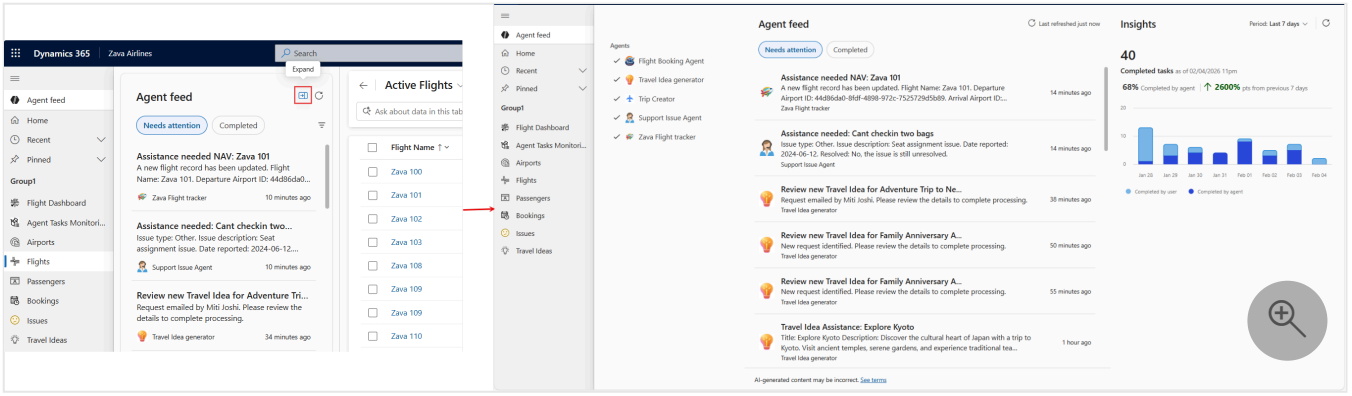
The enhanced agent feed is powered by the autonomous agents using Power Apps MCP server. Power Apps MCP server tools enable following two core human-agent collaboration patterns:

1. Autonomous Dataverse record creation and updates with human-in-the-loop approval and review.
2. Agent requests for human assistance and logging tasks for human review. In agent-enabled apps, user focus shifts from doing the work to supervising and prioritizing agent-driven work. Agents help with automations and organize work, ensuring business experts remain involved in decision-making and critical actions.

To allow users to view an agent's activity in the app, the agent must connect to Power Apps MCP server. The agent feed appears at the top of the site map in an app.

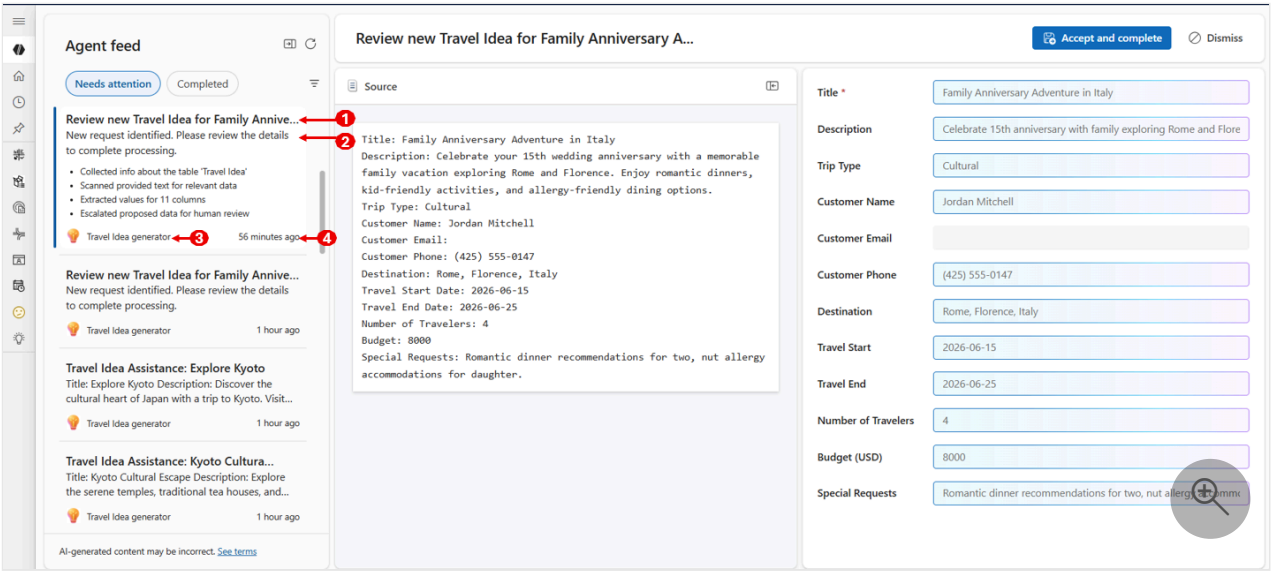


You can view the feed in the side pane or expand it to full screen.



The feed shows a list of agent tasks. Each item can include:

- Title of the task
- Description of the tasks with optional steps
- The agent's name and icon
- A timestamp of when the task was logged

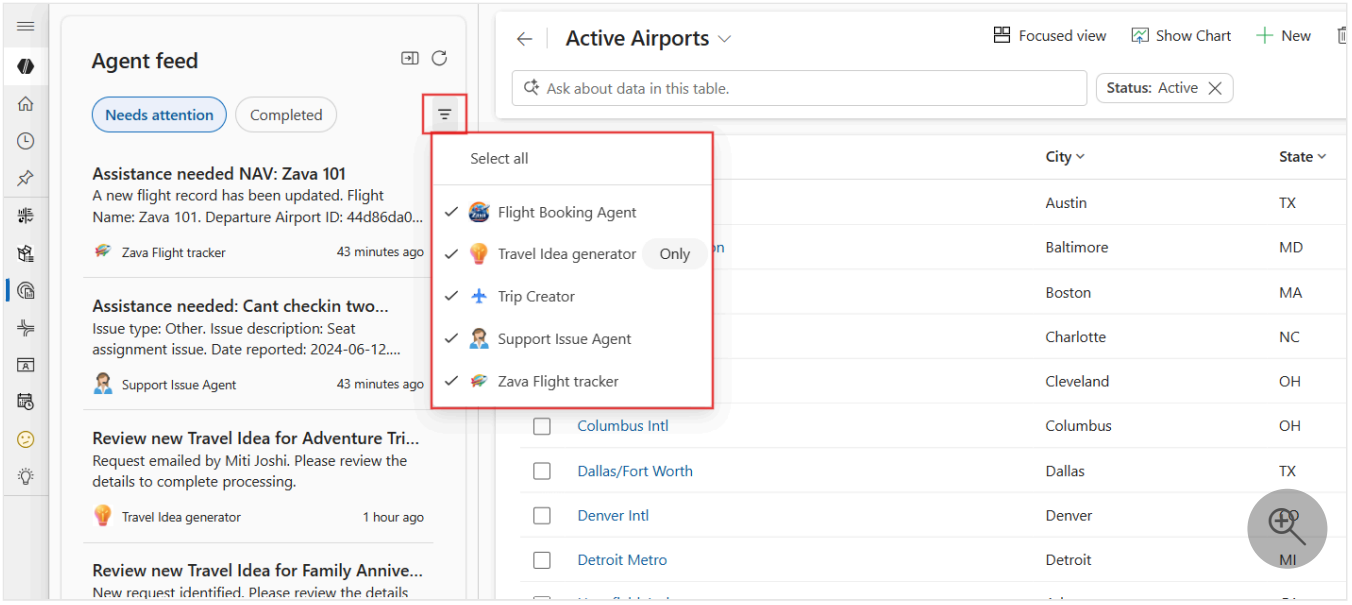


Activity is grouped into two sections:

- **Needs Attention**: Shows tasks that are waiting for your input or need your attention. These tasks are logged by using the `request_assistance` or `invoke_data_entry` MCP tools.
- **Completed**: Displays all other agent feed tasks, including tasks the user completed and tasks the agent completed that are logged for review by using the `request_review` MCP tool.

By default, each section loads 20 agent tasks and lazy loading is enabled as you scroll. The sections are sorted chronologically by the last modification date of each action.

You can filter the feed by agents.

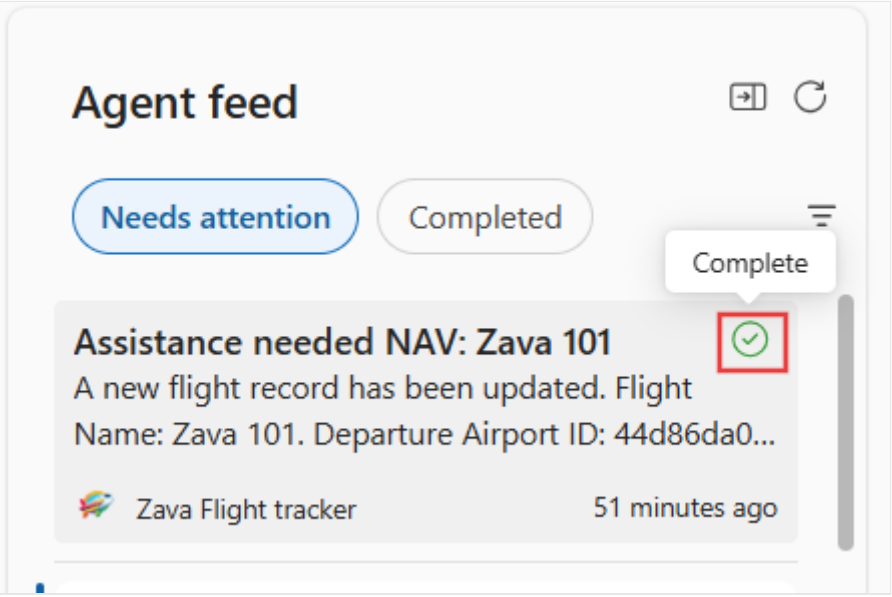


View the agent feed in a side pane to support multitasking or expand it to full screen for focused review.

Take action on agent activity

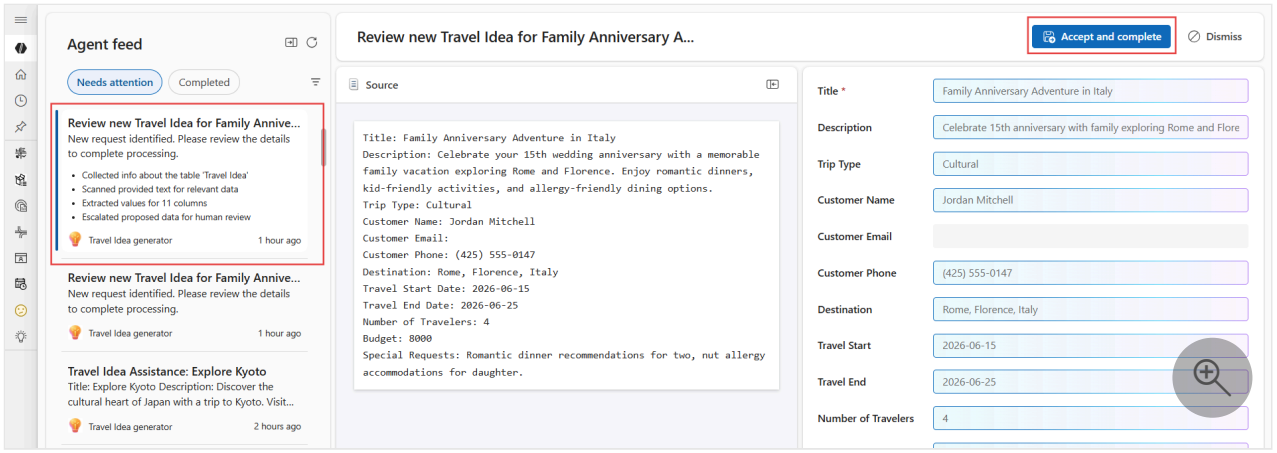
Take these actions on tasks under the **needs attention** tab in agent feed:

- **Complete**: Move a task from the **needs attention** section to **Completed** after completing the task the agent couldn't finish. This action is available for the agent tasks logged by using the **request_assistance** mcp tool.

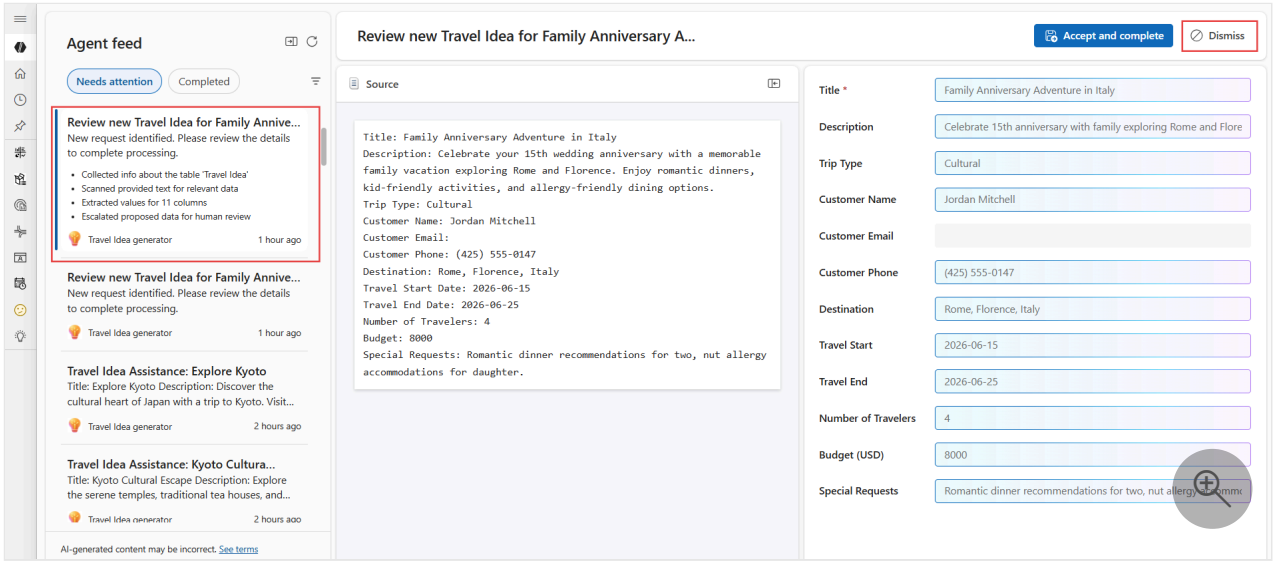


- **Accept and complete**: Move a data entry task from the **Needs attention** section to **Completed** after reviewing and accepting the data entry suggestions from

invoke_data_entry mcp tool. The **Accept and complete** button is available inside the agent pane associated with the agent feed task.

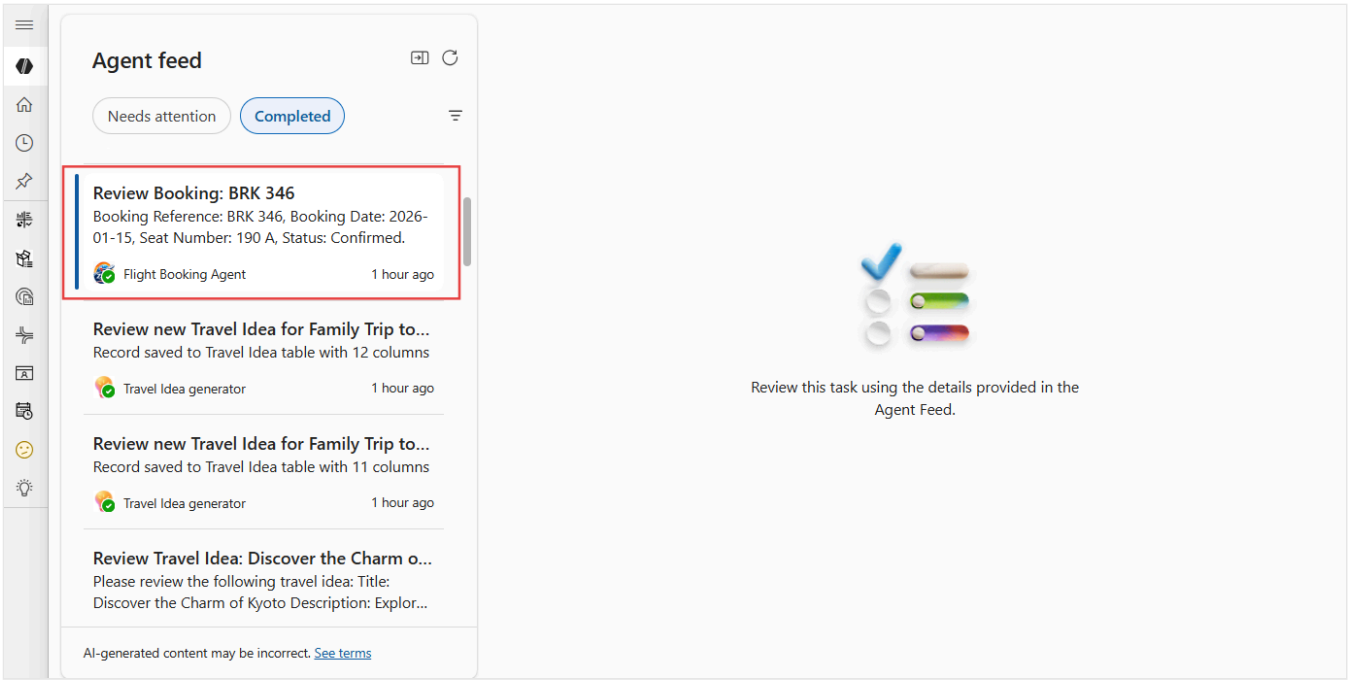


- **Dismiss**: Remove a data entry task logged via **invoke_data_entry** mcp tool from the **Needs attention** section if it's no longer relevant. The **Dismiss** button is available inside the agent pane associated with the agent feed task.



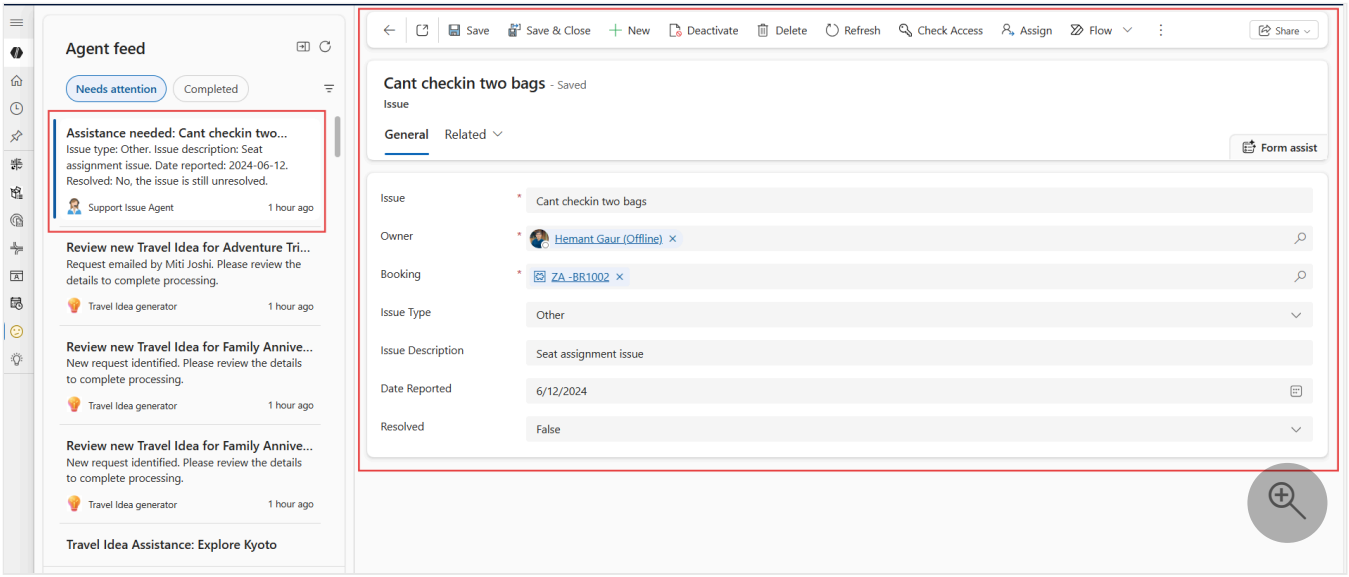
> **Note**
>
> You can perform all actions either from the main screen in the feed or from the details page of a specific action.

- **Request_review tasks**: Tasks logged via request_review mcp tool don't have any actions for the users. They're informational tasks completed autonomously by agent and can be found in completed agent feed tab.



Navigation from agent feed item

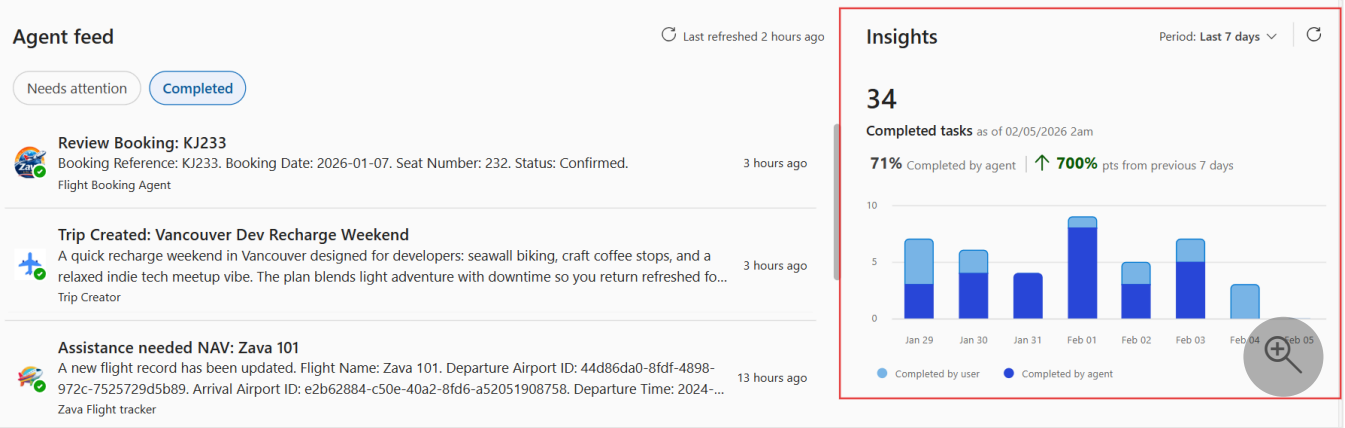
Agent feed tasks generated through the request_assistance or request_review MCP tools can reference a Dataverse record as a related navigation link in the agent description. This reference allows users to easily open and review associated data when interacting with the feed item.



Insights panel

The default insights panel complements the agent feed by giving users a clear, aggregated view of how their agents and users collaborate over time. While the agent feed surfaces individual tasks and actions in the moment, the Insights chart provides a historical, high-level representation of those interactions. This representation helps organizations understand the

true patterns of human–agent collaboration, whether users step in to provide missing context, guide agents through a complex workflow, complete tasks directly, or monitor agent activity. By visualizing these trends over time, organizations gain transparency into how their agentic workflows actually operate in practice, strengthening trust, identifying friction points, and revealing opportunities to improve system reliability and user experience.



The insights chart presents aggregated activity across the last 7, 14, or 30 days. It gives customers a time-based snapshot of collaboration between agents and users. Activity is categorized into two groups:

- Tasks completed by agents
- Tasks completed by users These at-a-glance aggregations offer a concise understanding of collaboration volume and balance over time. They help teams quickly assess how work is distributed across human and agent participants.

Related information

Add agents to an app

Customize Copilot Chat

Last updated on 04/07/2026

Use AI form fill assistance feature in model-driven apps

Form fill assistance suggests AI-generated content for fields, making it easier and faster to enter data in forms. The suggestions are entirely optional and aren't saved until the user explicitly reviews and accepts them.

Prerequisites

- Form fill assistance is available for all apps on the web where the modern, refreshed look is turned on.
- Requires at least one of the AI form fill assistance environment feature settings enabled.
- The **Allow form fill assistance** column property is enabled for the column where form fill assistance occurs. For more information, see Create and edit columns in Dataverse using Power Apps.

Limitations

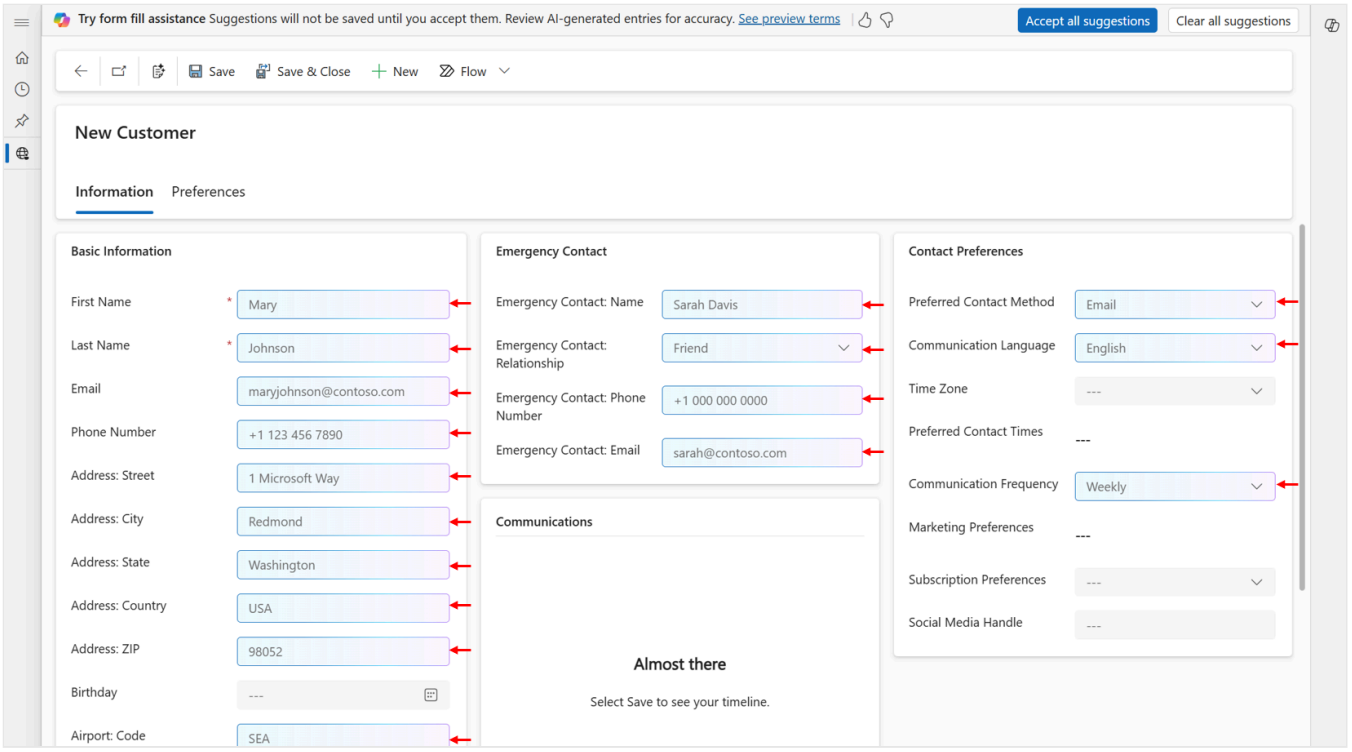
- Inside apps, suggestions are provided for fields only in main forms and quick create forms. In Microsoft 365 Copilot chat, suggestions are provided for fields based on main forms only.
- The currently supported field types inside apps are text, numeric, choice, and date. The currently supported field types inside Microsoft 365 Copilot chat are text, numeric, choice, date, and lookup.
- Fields that have column security aren't currently supported.

Use form fill assistance in apps

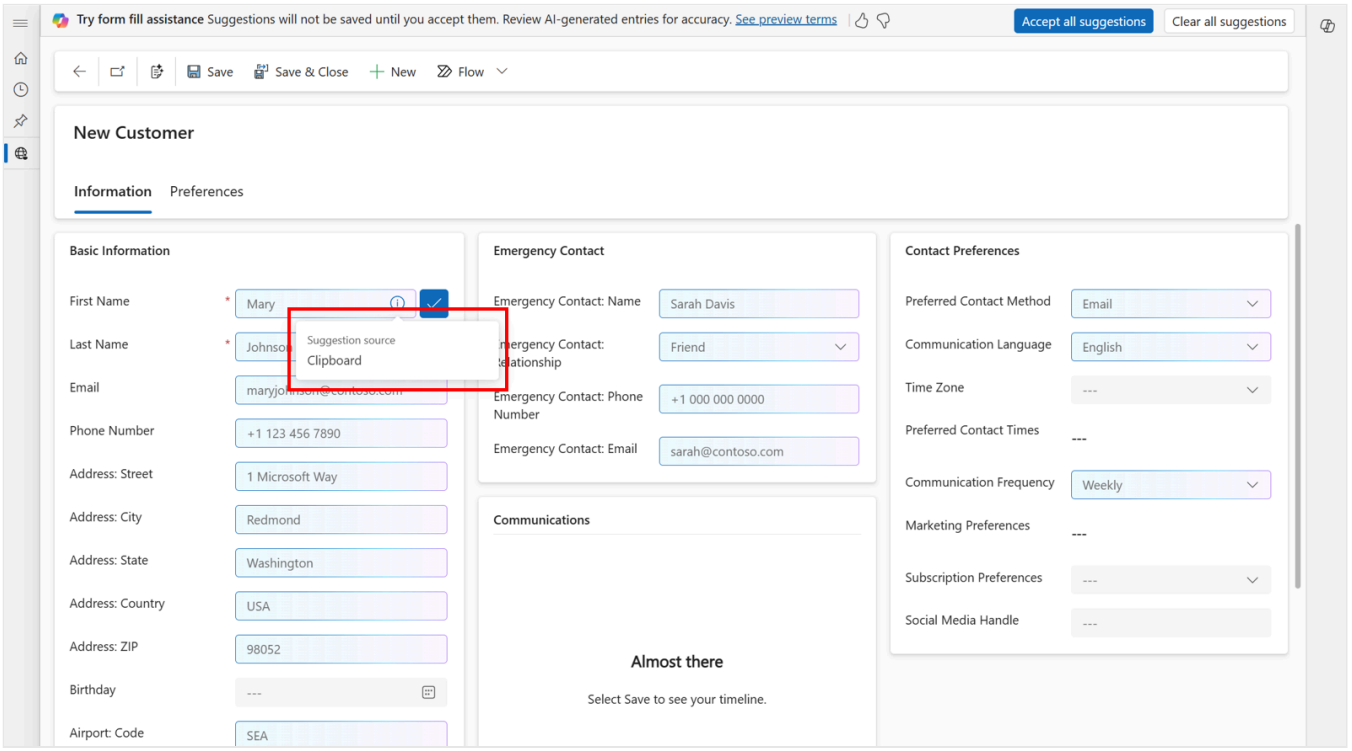
Form fill assistance in apps (Power Apps and Dynamics 365) provides suggestions for blank form fields in three ways:

1. Based on the user's usage of the app, what information is already available in the form, and their frequently used data.
2. Based on the copied text the user provides through the **Smart paste** capability.
3. Based on the file contents the user provides through the **Files** capability.

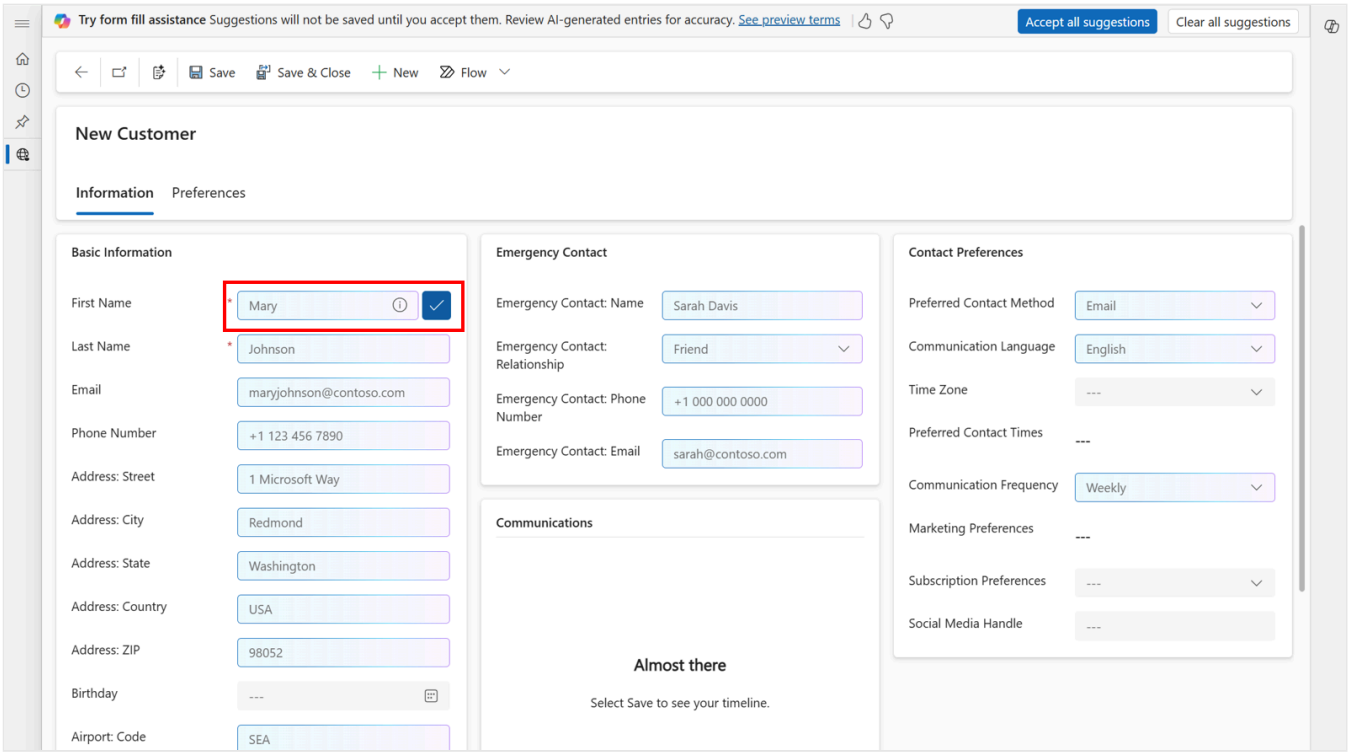
The feature generates suggestions only for the open tab. It presents suggestions inline in the form.



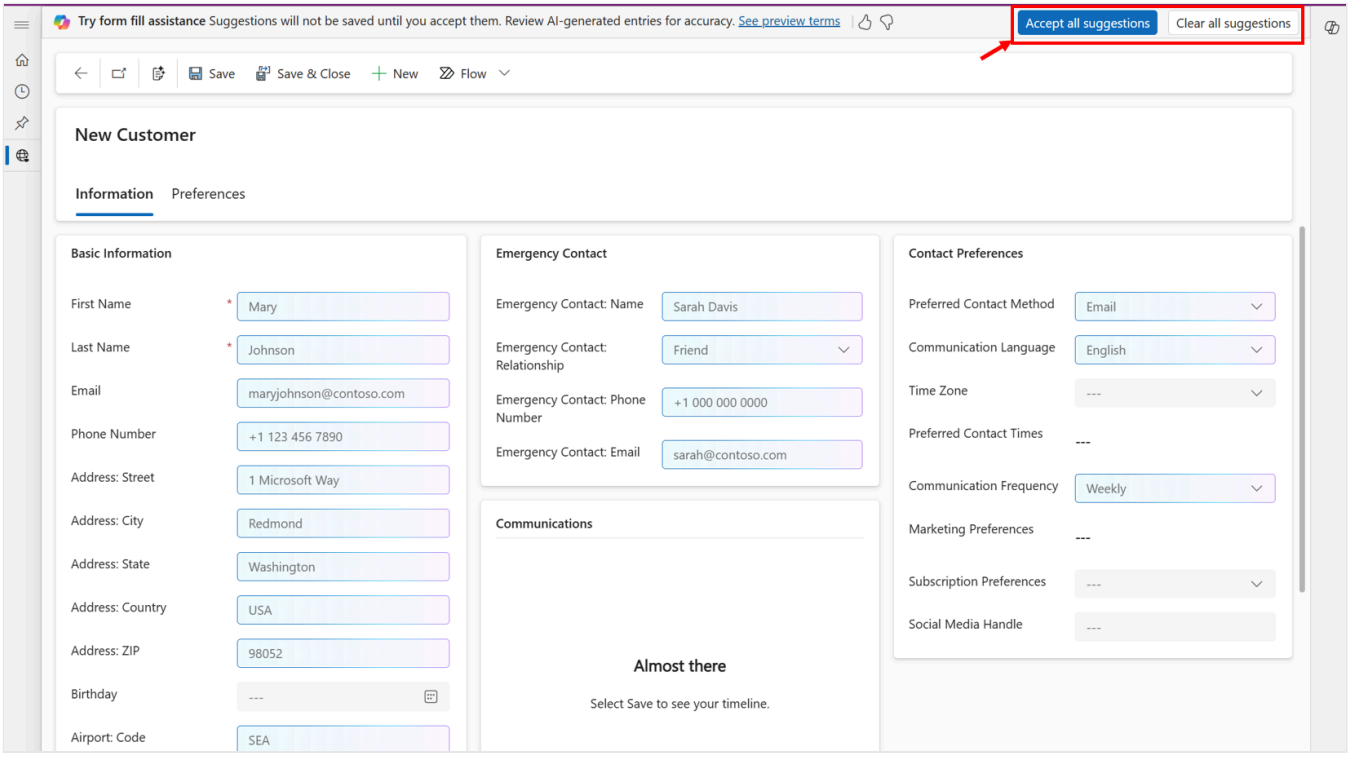
Suggestions are entirely optional and aren't saved in the app until you accept them. To learn more about a suggestion, hover over the field, and specifically hover over the information icon. This icon is a citation, and identifies the source of a specific suggestion, such as "Records you have updated recently" or "Clipboard".



To accept a specific suggestion, hover over the field, and then select **Accept**. Alternatively, select the field, and then press the **Enter** key.



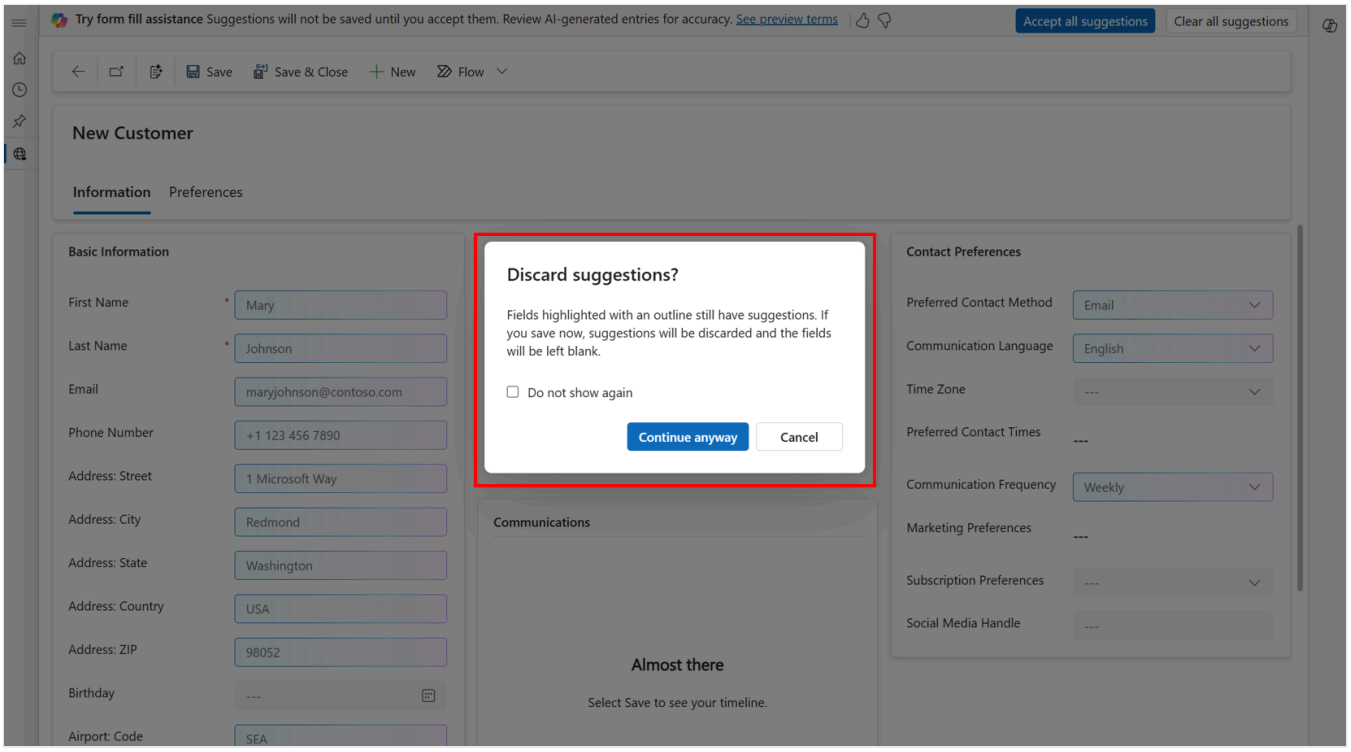
To accept all suggestions on the open tab of the form, select **Accept all suggestions** on the notification bar. To clear all suggestions on the open tab of the form, select **Clear all suggestions**.



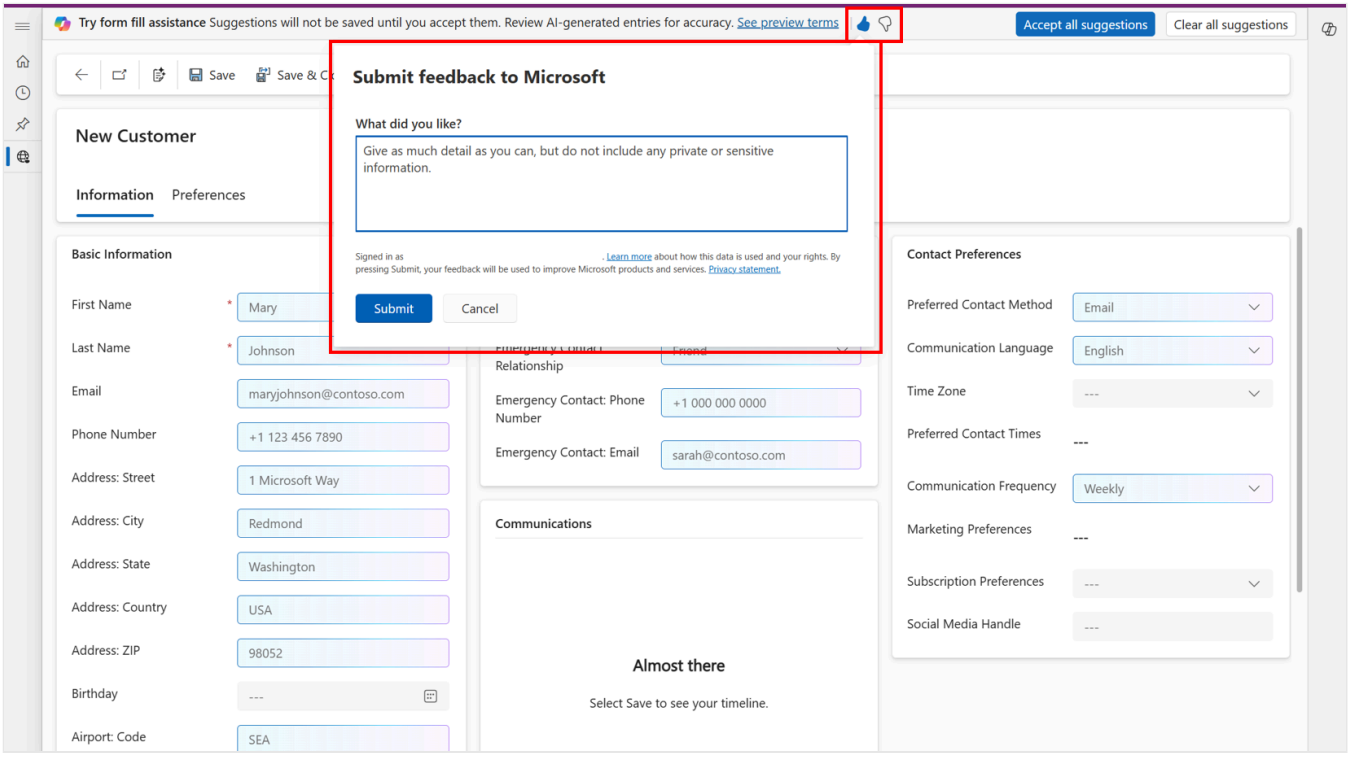
To replace a suggestion with a different value, select the field and start typing to overwrite the suggestion. Alternatively, select the field, press **Backspace** or **Delete**, and then start typing.

If you have unsaved suggestions in a form and try to navigate away, you get a prompt asking if you want to discard those suggestions. The suggestions are saved only when you accept them.

You can continue navigating away or stay on the form to review your pending suggestions. You can also select **Do not show again** to avoid being prompted in the future.



To provide feedback about the feature, select the thumbs up or down button on the notification bar, and optionally provide detailed comments to help improve the feature.



> **Important**
>
> - This feature is generally available in Dynamics 365 apps.

- This feature is in production ready preview in Power Apps.
  - Preview features are available before an official release so that customers can get early access and provide feedback.
- Due to high demand, this feature might be unavailable intermittently. If the feature is unavailable, try again later.

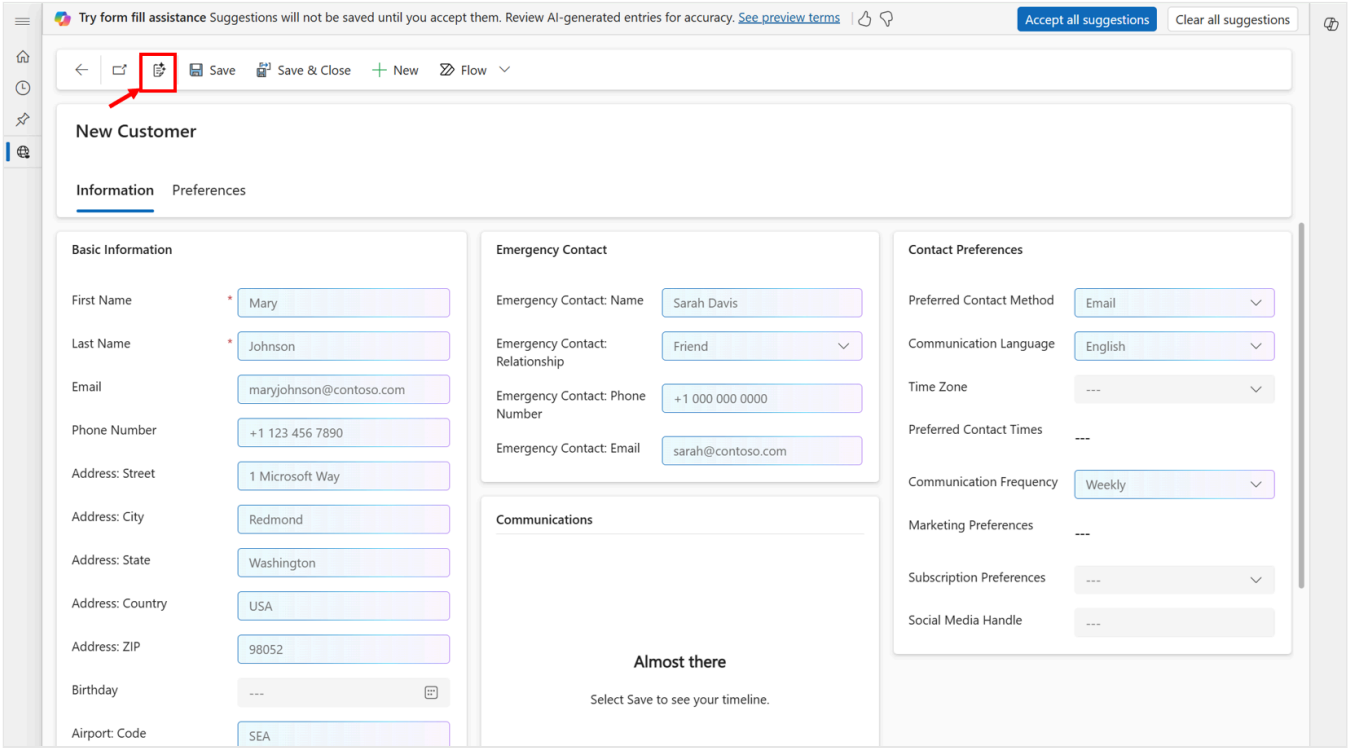
Use smart paste

Smart paste is a capability that analyzes the form and the text or image you copy to your clipboard. It suggests what text could fill specific fields and provides suggestions inline in the form. To use smart paste (preview):

1. Copy the text or image you want to use to fill in the form.
2. Select the **smart paste** icon in the command bar. Alternatively, make sure you don't select any specific fields in the form, and then use the regular paste keyboard shortcut (Ctrl+V or Cmd+V).

To use the smart paste (preview) capability, your admin must enable it. For more information, see Manage feature settings.

You can continue to paste directly, without smart paste (preview), into a specific field by selecting that field first and then pasting.

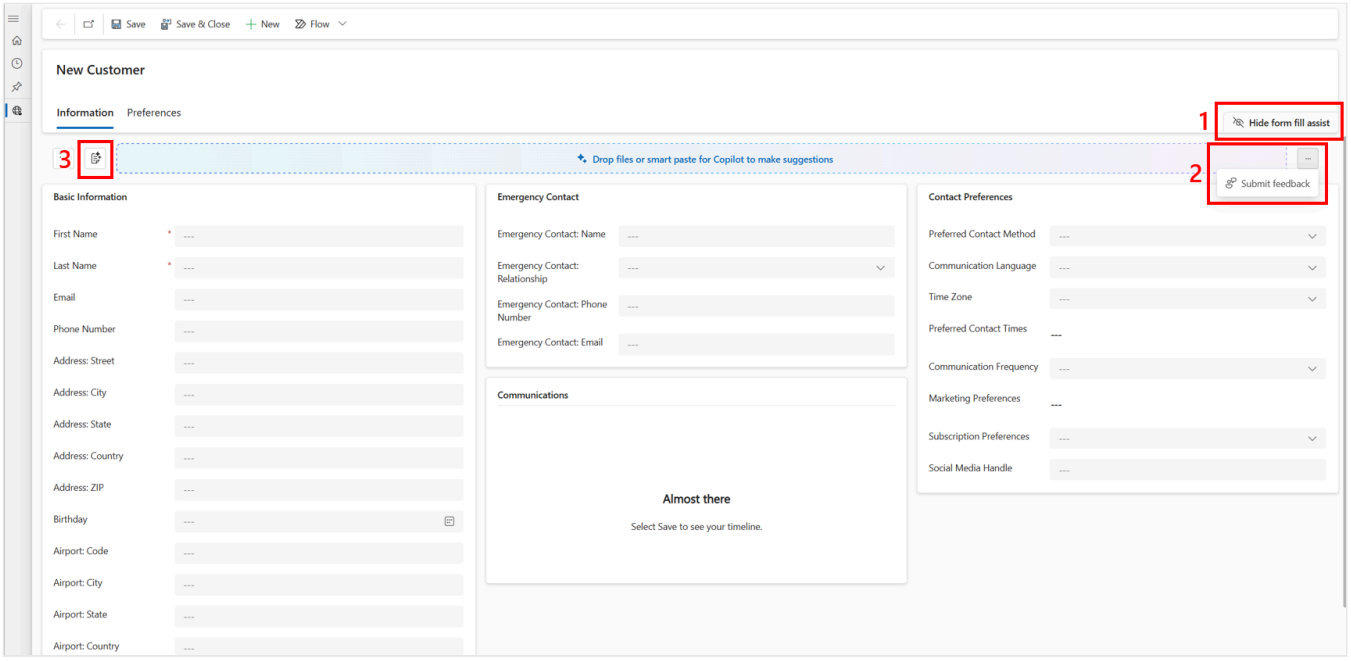


Use toolbar and files

Files is a capability that can reason over the form and the file you provide to suggest what text could fill specific fields. It provides suggestions inline in the form. Supported file types are .txt, .docx, .csv, .pdf, .png, .jpg, .jpeg, and .bmp. Files classified with sensitivity labels aren't supported.

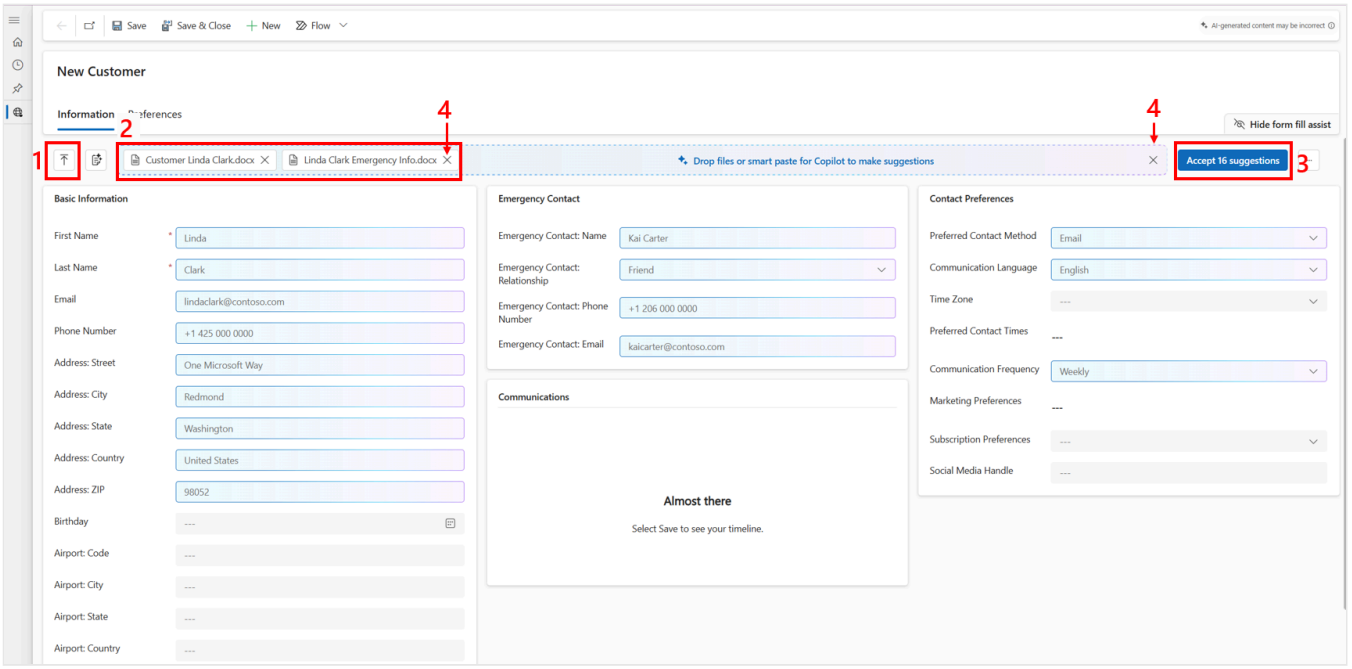
The form fill assist toolbar

1. Show or hide the form fill assist toolbar by using the show/hide button.
2. To provide feedback, select the more actions button (...) and give a compliment, report a problem, or make a suggestion.
3. The smart paste button is now located to the left of the toolbar.



To use files:

1. Use the file selector button to select the file you want to use to fill in the form. Alternatively, drag and drop the file in the toolbar.
2. Hover over the file's source tag to identify which fields are suggested based on the file's contents.
3. To accept all suggestions from the sources listed in the toolbar, select the accept all button located to the right of the toolbar. This button identifies the number of open suggestions in the form that it accepts.
4. To clear suggestions from the file, select the clear button in the tag. If the toolbar lists multiple sources, clear all suggestions by using the clear all button in the toolbar.



Licensing requirements

The form fill assistance requires specific user licenses based on the product. If the user doesn't have the license, the feature is hidden. This license enforcement will gradually rollout following the feature's general availability.

- **Power Apps model-driven app**: The user must have a Power Apps premium license. For details, see the Power Platform License Guide.
- **Dynamics 365 model-driven app**: The user must have a Dynamics 365 enterprise or premium license as outlined in Dynamics 365 License Guide.

Usage might be subject to service limits or throttling. Microsoft reserves the right to update the limits.

Admin control

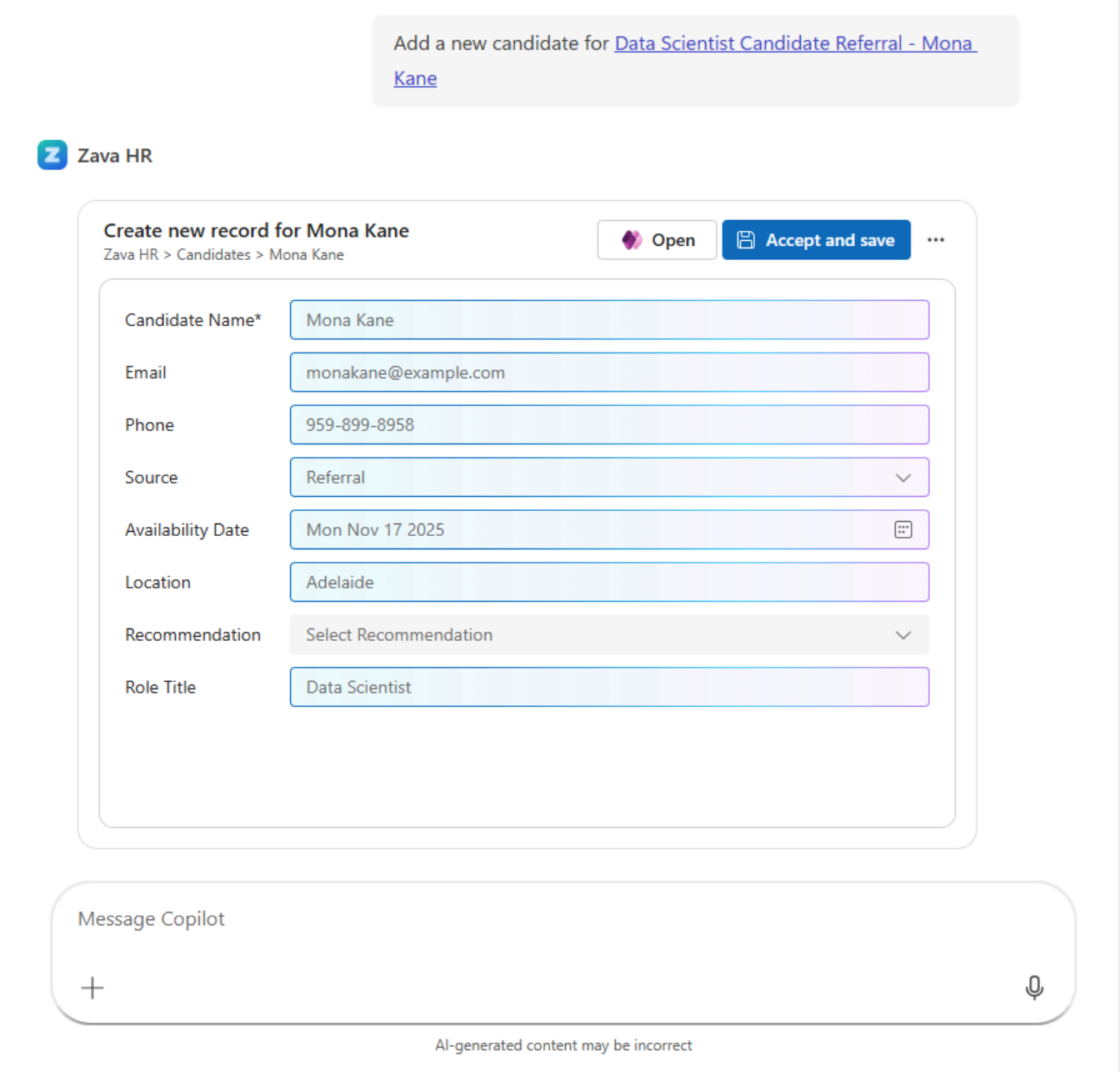
The primary admin control for form fill assistance is moving to the Power Platform admin center under **Copilot** > **Settings** > **Power Apps** > **Data entry**. Learn more in Copilot hub. This is gradually rolling out over the next coming weeks.

Smart paste (`FormPredictSmartPasteEnabled`) or **form fill assist toolbar** (`FormFileUploadEnabled`) app settings previously available in Power Platform admin center **Environment** > **Settings** > **Product** > **Features** page is moving and you can configure this setting at the app level either by using Manage model-driven app settings in the app designer or Updating a setting definition.

Use form fill assistance with agents in Microsoft 365 Copilot (preview)

[This section is prerelease documentation and is subject to change.]

With Microsoft 365 Copilot, you have ability to bring app-based experiences to agents. The app experience can leverage the form fill assistance right in the chat.



> **ⓘ Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are

available before an official release so that customers can get early access and provide feedback.
- This feature is being gradually rolled out across regions and might not be available yet in your region.

> **Note**
>
> To use this capability, you need to set up an apps agent in Microsoft 365 Copilot.

When you interact with agents in Copilot, agents can surface forms from apps directly in the conversation. Form fill assistance helps populate these forms by suggesting values based on relevant context, such as existing data you have access to from emails, chats, or any document.

The suggestions appear inline in the chat-based form and you can review or edit them before submission. All suggestions are optional and aren't saved until you explicitly accept and save the input.

After submission, the data is saved to the connected app and is available, just as if you complete the form directly in the app.

Related information

Manage model-driven app settings in the app designer
Create and edit columns in Dataverse using Power Apps
Responsible AI FAQ about form fill assistance
Complete forms quickly with model-driven apps (video)

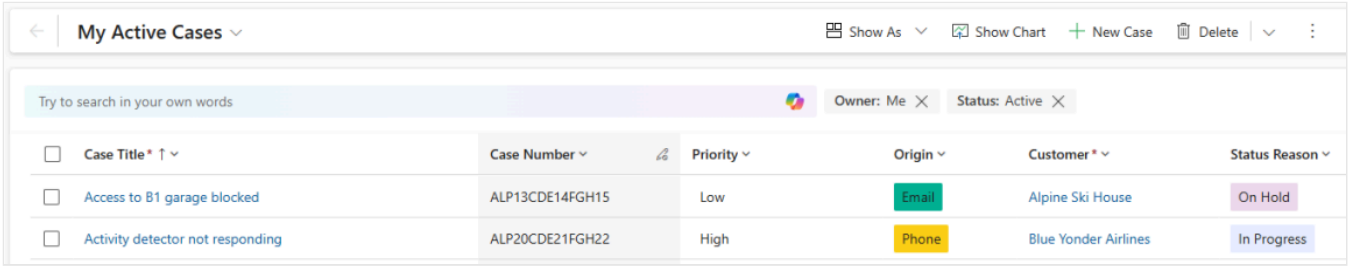
Last updated on 04/21/2026

Find data in a view with AI

Users can quickly find, filter, and sort their data by using natural language, so they don't need to use complicated advanced filters. If your administrator turns on **Natural Language Grid and View Search**, you see the natural language search box.

Natural language search in apps

Use smart grid natural language search to ask data-related questions in natural language. For example, request “cases with high priority with overdue follow-up by date” to filter your view and display only those relevant cases.



Licensing requirements

Natural language grid and view search requires specific user licenses based on the product. If the user doesn't have the license, the feature is hidden. This license enforcement will gradually rollout following the feature's general availability.

- **Power Apps model-driven app**: The user must have a Power Apps premium license. For details, see the Power Platform License Guide.
- **Dynamics 365 model-driven app**: The user must have a Dynamics 365 enterprise or premium license as outlined in Dynamics 365 License Guide.

Usage might be subject to service limits or throttling. Microsoft reserves the right to update the limits.

Admin control

The primary admin control for **Natural Language Grid and View Search** is moving to the Power Platform admin center under **Copilot** > **Settings** > **Power Apps** > **Data exploration**. Learn more in Copilot hub. This is gradually rolling out over the next coming weeks.

Natural language grid and view search app setting (`NLGridSearchSetting`) previously available in Power Platform admin center **Environment** > **Settings** > **Product** > **Features** page is moving and you can configure this setting at the app level either by using Manage model-driven app settings in the app designer or Updating a setting definition.

Explore data with agents in Microsoft 365 Copilot (preview)

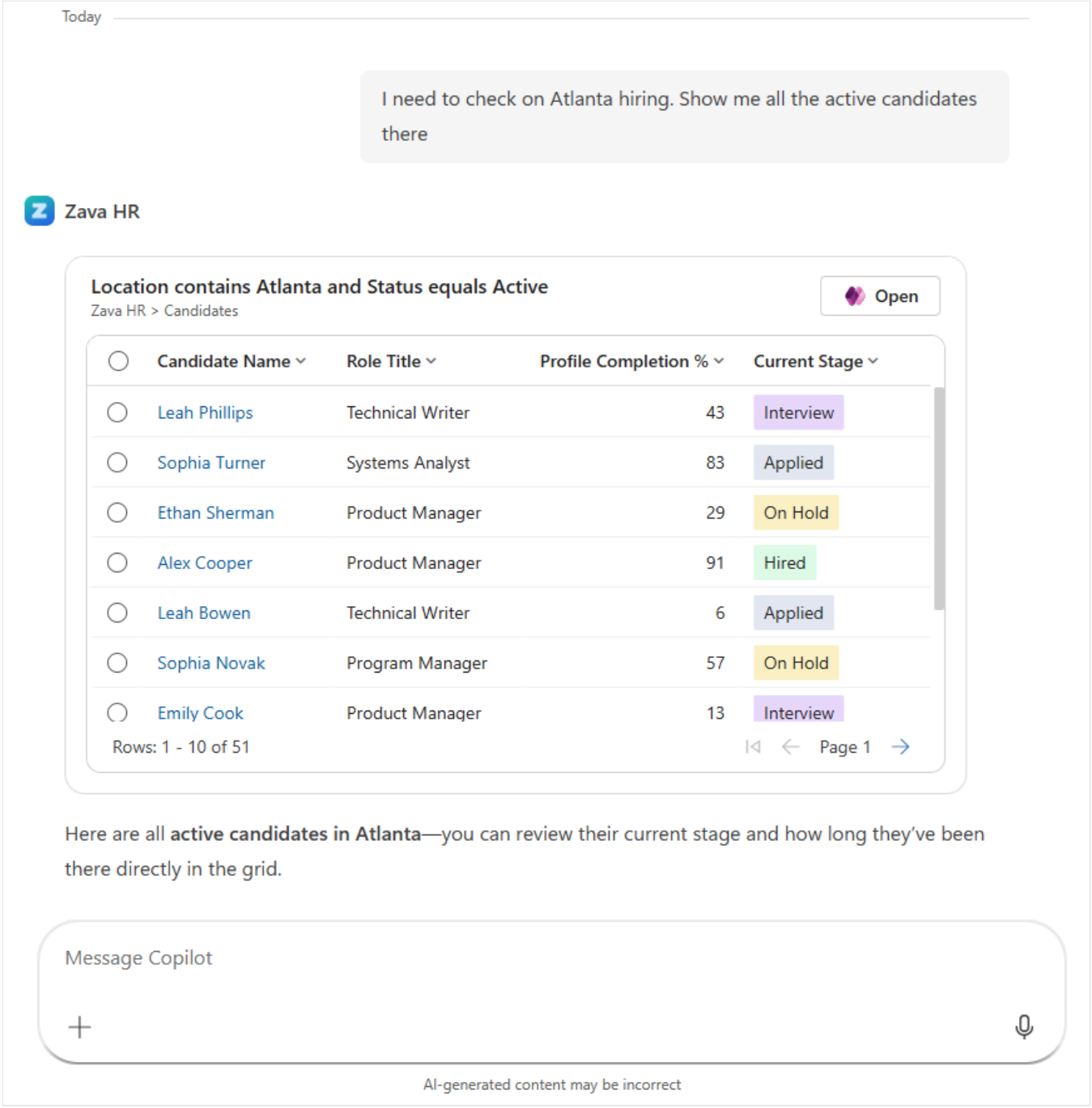
[This section is prerelease documentation and is subject to change.]

With Microsoft 365 Copilot you have the ability to bring app-based experiences to agents, the app experience can also use the AI assisted search in the grid view.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is being gradually rolled out across regions and might not be available yet in your region.

> **Note**
>
> To use this capability, you need to set up an apps agent in Microsoft 365 Copilot.

When users interact with agents in M365 Copilot, the agents can search and retrieve data from apps view directly within the conversation. By using natural language, users can find, filter, and review app data without switching to the app experience. Natural language search interprets the user's request and applies the appropriate filters, sorting, and text search against the underlying view data. The agent returns the results in Copilot chat, presented in a structured, tabular format that reflects the view data. Users can continue to work with the returned results in the conversation, iterating with natural language to refine the output until it reflects the exact results they're looking for.



Supported features

- Filtering of records
- Sorting
- Text search

Known limitations

These capabilities aren't supported:

- Query aggregation
- Query grouping

- Adding columns

Considerations

- After you run a query, check the generated filter tags to make sure that the filter conditions were correctly interpreted from your natural language query. If any part of your query is missing from the filter tags, the results aren't filtered by that condition.
- If search doesn't give the desired results, consider modifying your query by:
  - Referring to data columns by the names shown in the grid header.
  - Separating multiple conditions with commas or periods.
- In model-driven app for grids, if your search string has two words or fewer, the search does a text search. To do a natural language search, use more than two words. To do a text search with more than two words, put the search term in single or double quotes. This behavior doesn't apply to Copilot chat.

Last updated on 04/21/2026

Visualize data in a view with AI

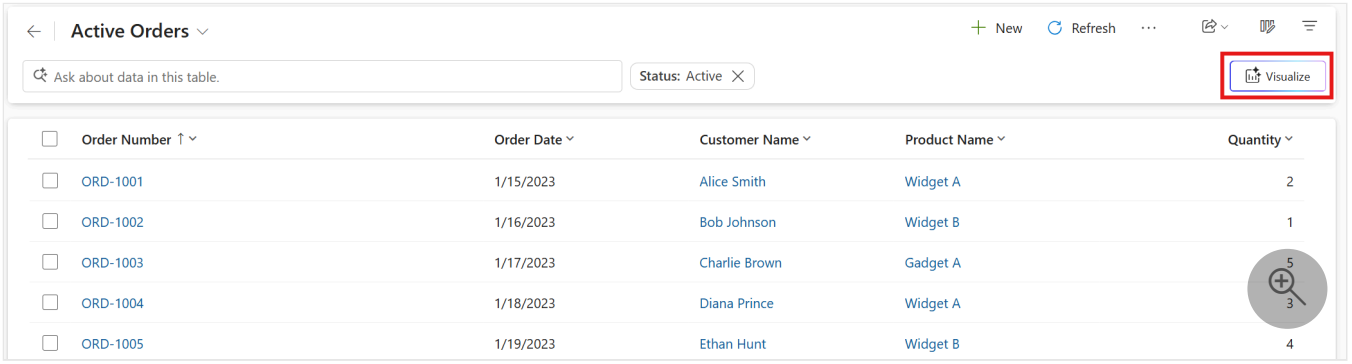
Chart visualization helps you visualize the data in the view as a chart. This visualization makes it easy to understand trends, patterns, and relationships in your data. The visualization is interactive, based on the visible columns in the view, and you can personalize it to support further data exploration.

Prerequisites

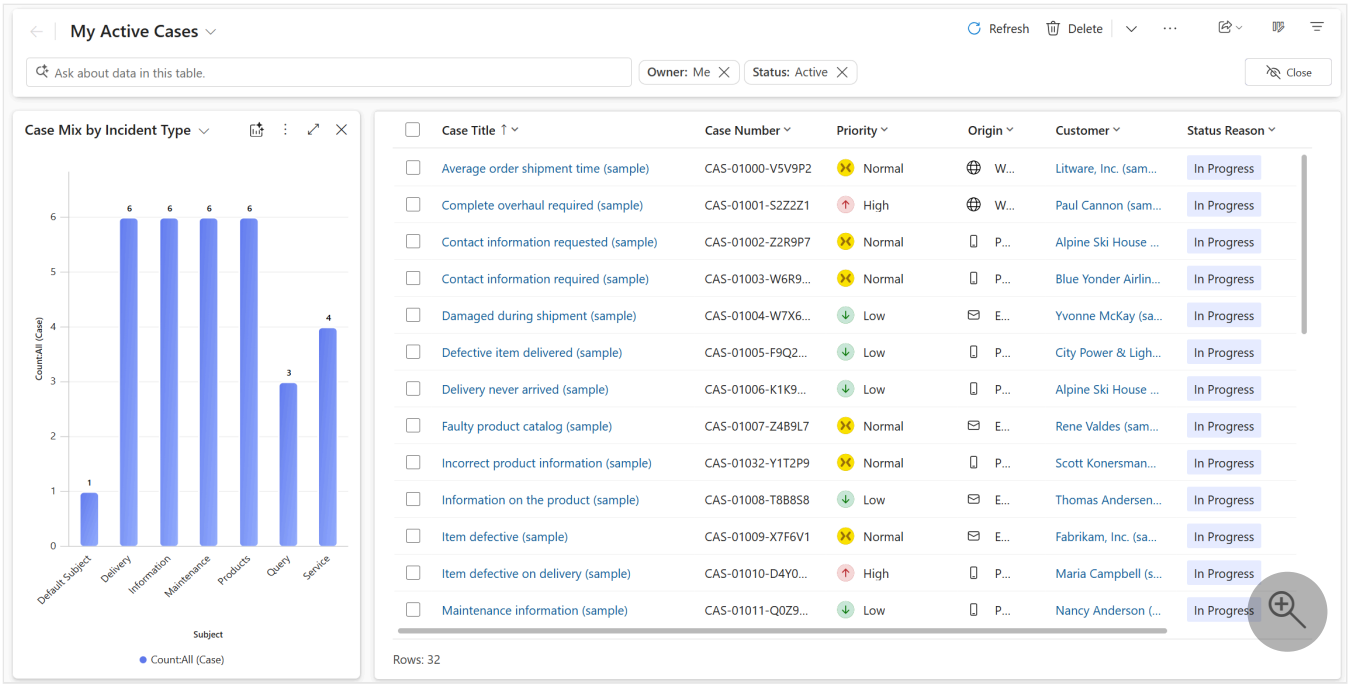
- All apps on the web support chart visualization when the modern, refreshed look is turned on.
- See License requirements and Admin control for additional settings.

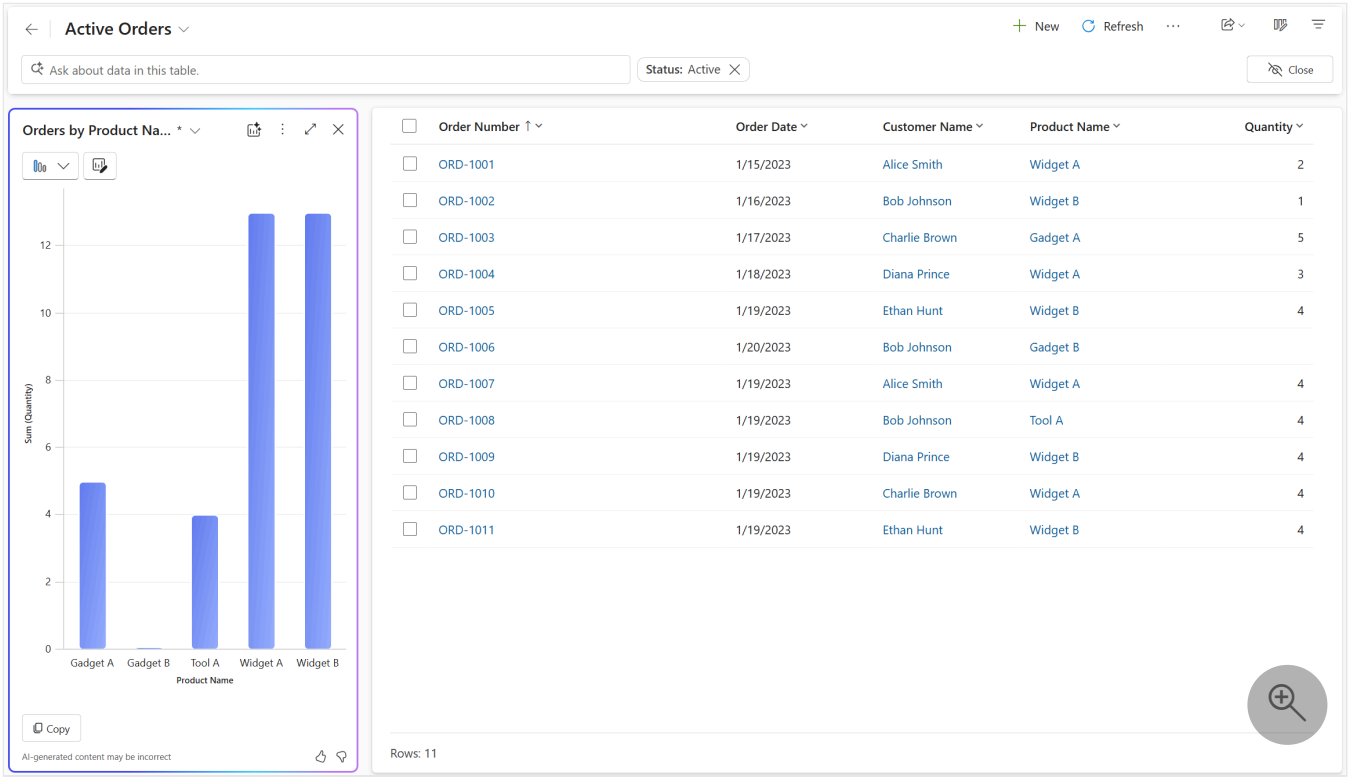
Visualize data in a view

When you navigate to a grid page, you can see the **Visualize** option on top of the page.

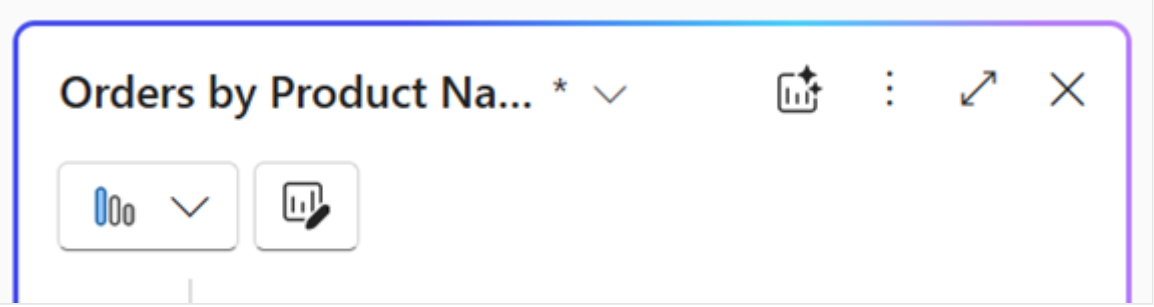


When you select **Visualize**, the chart pane opens next to the grid, complementing the table of rows. If the table has system or personal charts, the default chart for the table appears. If the table doesn't have charts, AI generates a visualization for the view based on the visible columns and filters. The visualization is a chart that aggregates the data in the view.





If chart visualization generates a visualization for the view, you can edit the chart by changing the chart type, the columns that are visualized, or both. If you add or remove columns from the view, chart visualization regenerates the chart based on the updated set of columns for the visualization. If you add or remove filters, chart visualization refreshes the chart without changing the visualization. The asterisk (*) next to the chart name shows that the chart isn't saved yet.



Select the three dots in the upper-right corner of the chart pane, and then select **Save As** to save the chart.

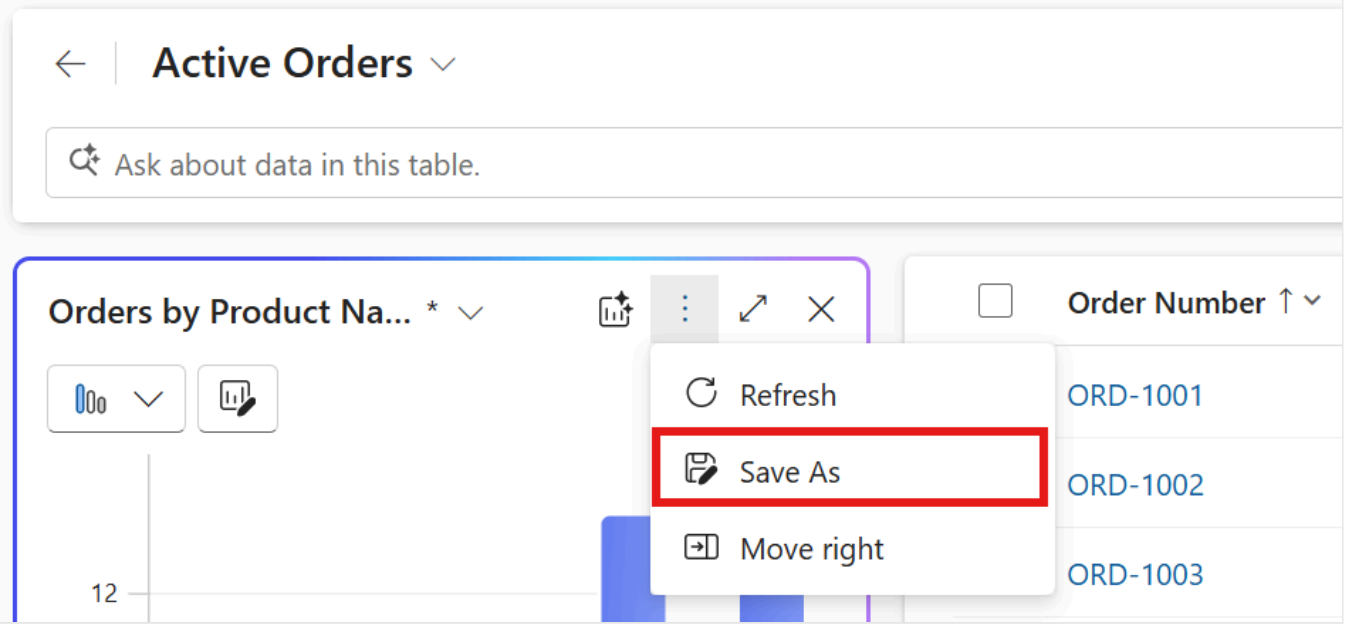
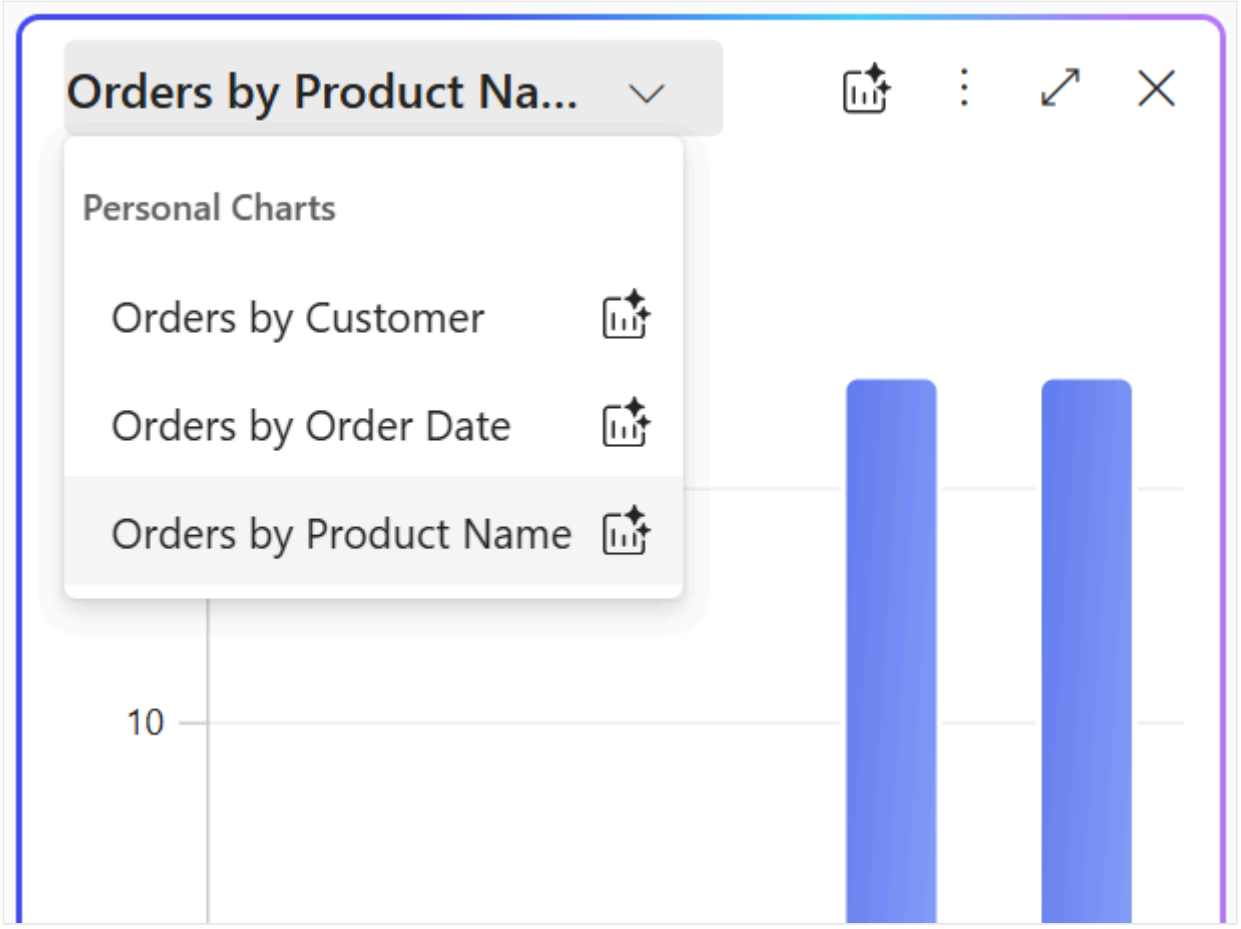
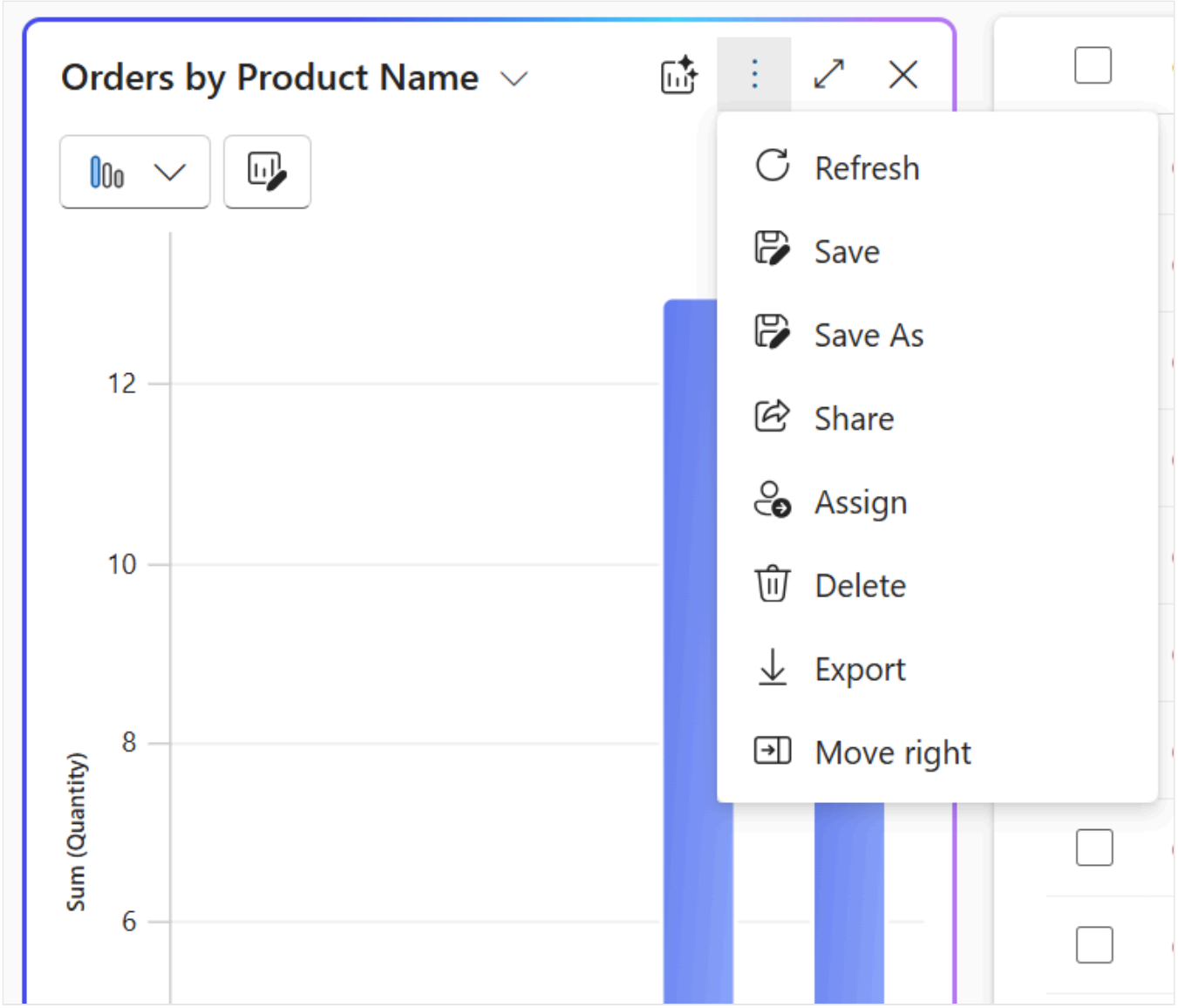


Chart visualization automatically populates the chart name and description, but you can edit them to suit your needs. Select **Save** to save the chart. You access the saved chart by using the chart switcher in the chart pane. The **Visualize** icon next to the chart name indicates that it's an AI-generated chart, helping you distinguish it from other charts.



To manage a personal chart, select the chart in the chart switcher, and then select the three dots in the upper-right corner of the chart pane. Select an option to delete, assign, or share the chart with others in your organization.



Hover over the charts to examine data points or select chart elements to drill into the data. The grid filters rows automatically to match the selected area of the chart. Selecting the same area of the chart again toggles the selection and removes the filter.

> **Note**
>
> Charts created by using the legacy chart designer use the same styling as AI-generated charts. To learn how to turn off the styling update for charts created by using the legacy chart designer, see Manage feature settings.

Visualize data in a view with natural language

Use natural language prompts in the search bar on the grid page to visualize data in the view. Here are example prompts to help you understand the types of requests supported.

- *Visualize data as a bar chart.*

- *Customers by location. Use the prompt [table name] by [column] generically depending on the table you're visualizing.*
- *Trend of orders created during this fiscal year.*
- *Visualize average revenue per month as a column chart.*
- *Visualize high-priority cases as a chart.*
- *Visualize data as a bubble chart.*

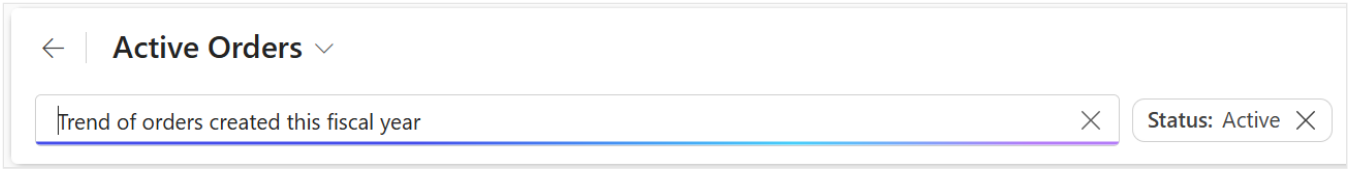
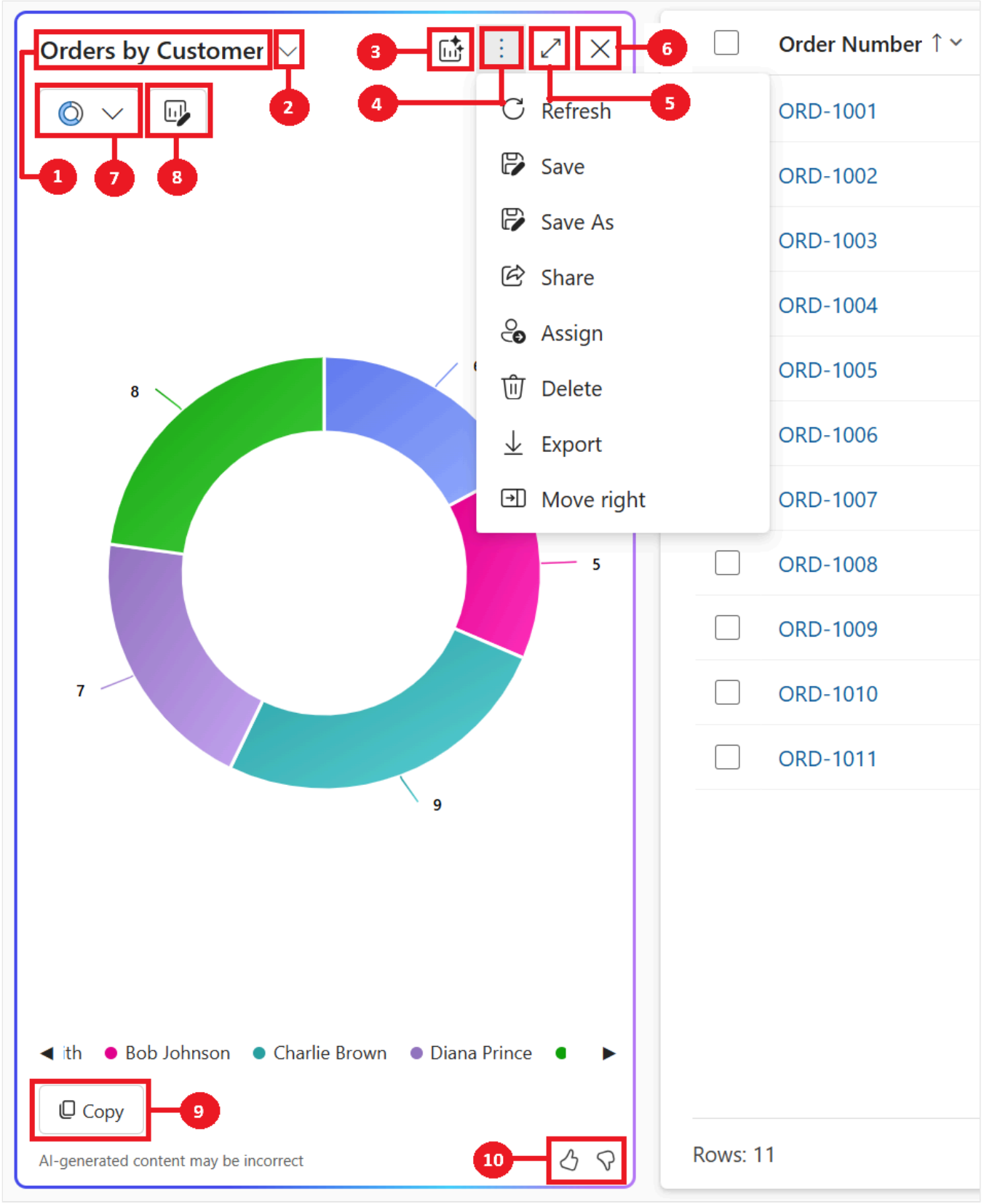


Chart pane



1. **Chart title**: AI-generated title for the chart based on the data and columns visualized.
2. **Chart switcher**: Select and manage saved charts. Use it to switch between different charts, including AI-generated and personal charts, for the current view.
3. **Create new chart with AI**: AI-generated chart for the visualization of this view.
4. **Commands to manage saved chart**: The **Commands to manage saved chart** appear as three dots in the top-right corner of the chart pane. They provide options to perform actions on saved charts.

5. **Expand**: Select **Expand** to expand the chart to full screen for a better view of the chart area.
6. **Close**: Select **Close** to close the chart pane.
7. **Chart selector**: List of 11 supported chart types – Pie, Donut, Bar, Column, Clustered Bar, Stacked Bar, Clustered Column, Stacked Column, Line, Histogram, and Bubble chart.
8. **Edit chart columns**: Customize the chart by selecting which columns from the view are visualized.
9. **Copy**: Select **Copy** to copy the chart as a PNG file to the clipboard.
10. **Feedback**: Select **Thumbs up** or **Thumbs Down** to provide feedback about the feature.

To provide feedback about the feature, select the thumbs up or down button on the chart pane, and provide detailed comments to help improve the feature.

Licensing requirements

The visualize data in a view requires specific user licenses based on the product. If the user doesn't have the license, the feature is hidden. This license enforcement will gradually rollout following the feature's general availability.

- **Power Apps model-driven app**: The user must have a Power Apps premium license. For details, see the Power Platform License Guide.
- **Dynamics 365 model-driven app**: The user must have a Dynamics 365 enterprise or premium license as outlined in Dynamics 365 License Guide.

Usage might be subject to service limits or throttling. Microsoft reserves the right to update the limits.

Admin control

The primary admin control for visualize data in a view is moving to the Power Platform admin center under **Copilot** > **Settings** > **Power Apps** > **Data exploration** > **Chart Visualization**. Learn more in Copilot hub. This is gradually rolling out over the next coming weeks.

Allow AI to generate charts to visualize the data in a view app setting (`NLChartDataVisualizationSetting`) previously available in Power Platform admin center **Environment** > **Settings** > **Product** > **Features** page is moving and you can configure this setting at the app level either by using Manage model-driven app settings in the app designer or Updating a setting definition.

Related information

Manage feature settings

Responsible AI FAQ about Copilot visualizations on a view

Last updated on 04/21/2026

Use row summaries in model-driven apps

The **Row summary** feature in apps provides users with a concise, AI-generated overview of key details about a record. Summaries help users quickly understand essential information without manually scanning fields, related records, or activity timelines. You see summaries as a paragraph or bullet points, so they're easy to read.

Summaries can be accessed in two ways:

- From the **Insights** bar at the top of main forms.
- From view pages by selecting the inline **Summary** action in the grid, which opens the summary for a record in a modal popup.

The feature enhances user efficiency by delivering context-rich insights directly within the form. Users can interact with summaries to copy content, regenerate updated information, and provide feedback on their relevance, improving both usability and accuracy.

Prerequisite

Each of the following settings must be turned on to see the row summaries for apps.

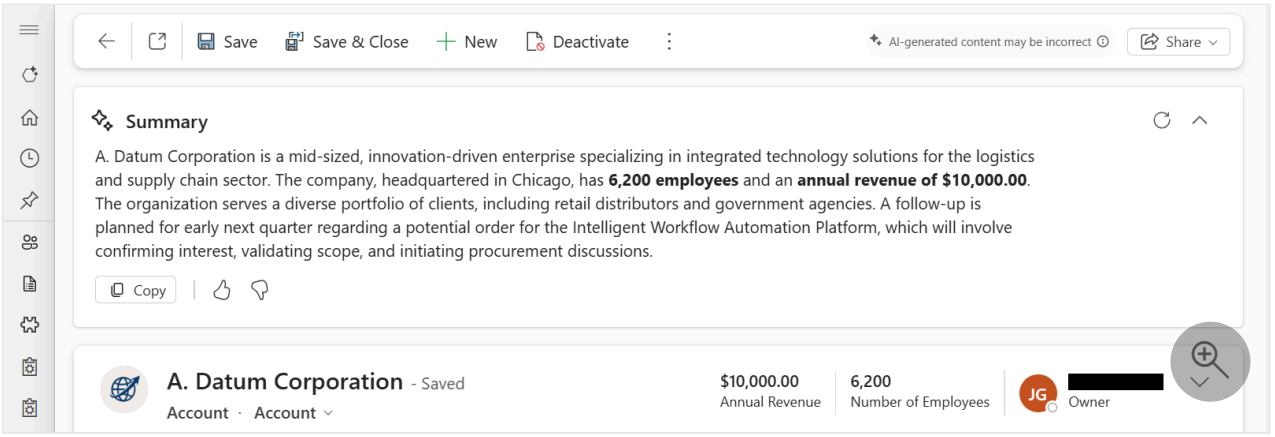
- The AI insight cards toggle is turned on for the Power Platform environment.

Feature details

Accessing record summaries

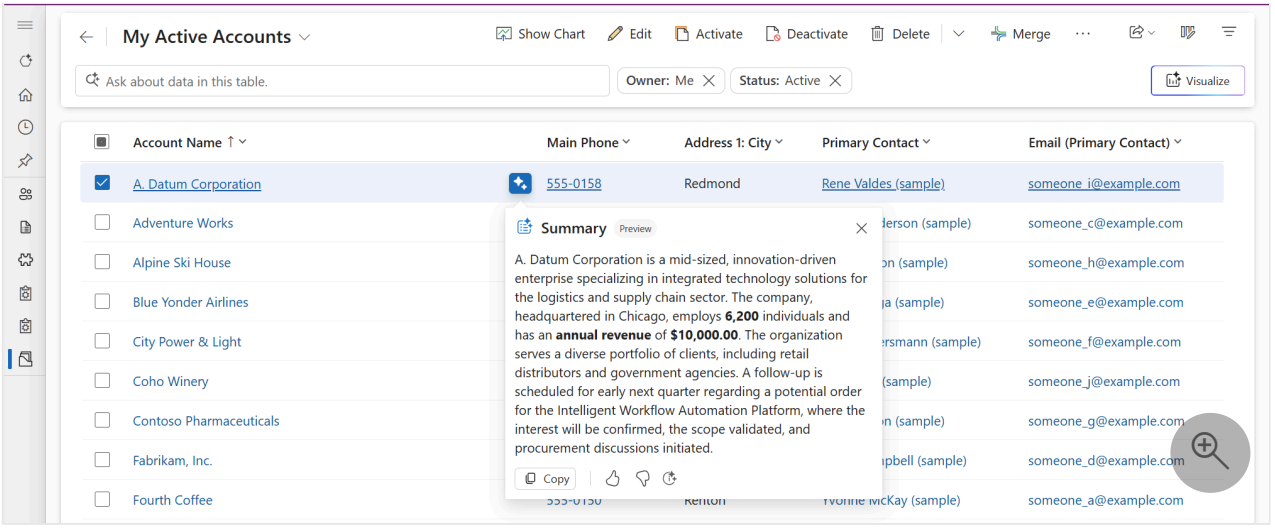
View the record summary from a form or view when a table is configured to display summaries.

- **Forms**: When viewing a record in a main form, the summary appears in the insights bar at the top of the main form.



> **Note**
>
> You see the record summary after you save a new record. If you haven't saved the record, the summary area stays hidden.

- **Views**: When browsing records in a view, select the inline Summary action next to a record to open the summary in a modal popup.

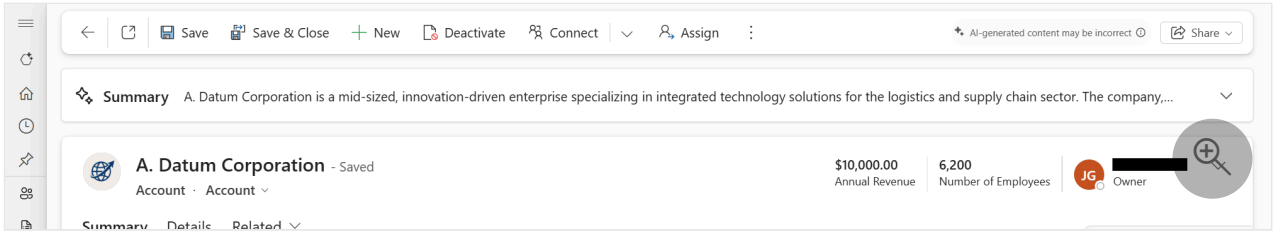


Interacting with summaries

Here are some actions you can take with summaries:

- **Copy**: Select to copy the summary content directly to your clipboard for easy sharing or documentation. The summary copies in markdown format, so when you paste it into other applications, the formatting might not appear exactly as it appears in the app.
- **Feedback**: Use the thumbs up or thumbs down icons to rate the summary's usefulness. Your feedback helps improve future summaries so they better meet your expectations and needs.

- **Refresh** (forms only): Select the **Refresh** button to regenerate the summary so it reflects the latest updates to the record.
- **Expand/Collapse** (forms only): In forms, the insights bar is collapsed by default and shows a one-line peek of the summary. Expand the insights bar to see more details.



Licensing requirements

The row summary feature requires specific user licenses based on the product. If the user doesn't have the license, the feature is hidden. This license enforcement will gradually rollout following the feature's general availability.

- **Power Apps model-driven app**: The user must have a Power Apps premium license. For details, see the Power Platform License Guide.
- **Dynamics 365 model-driven app**: The user must have a Dynamics 365 enterprise or premium license as outlined in Dynamics 365 License Guide.

Usage might be subject to service limits or throttling. Microsoft reserves the right to update the limits.

Admin control

The primary admin control for row summary is moving to the Power Platform admin center under **Copilot** > **Settings** > **Power Apps** > **Summary** > **Row summaries**. Learn more about the Copilot hub here.

The **AI insight cards** app setting (EnableFormInsights) for main forms, previously available in Power Platform admin center **Settings** > **Product** > **Features** is being moved and you can configure this setting at the app level either by using the Model App Designer Setting or Solution Explorer App Settings.

Related information

Manage feature settings
Configuring summaries for tables

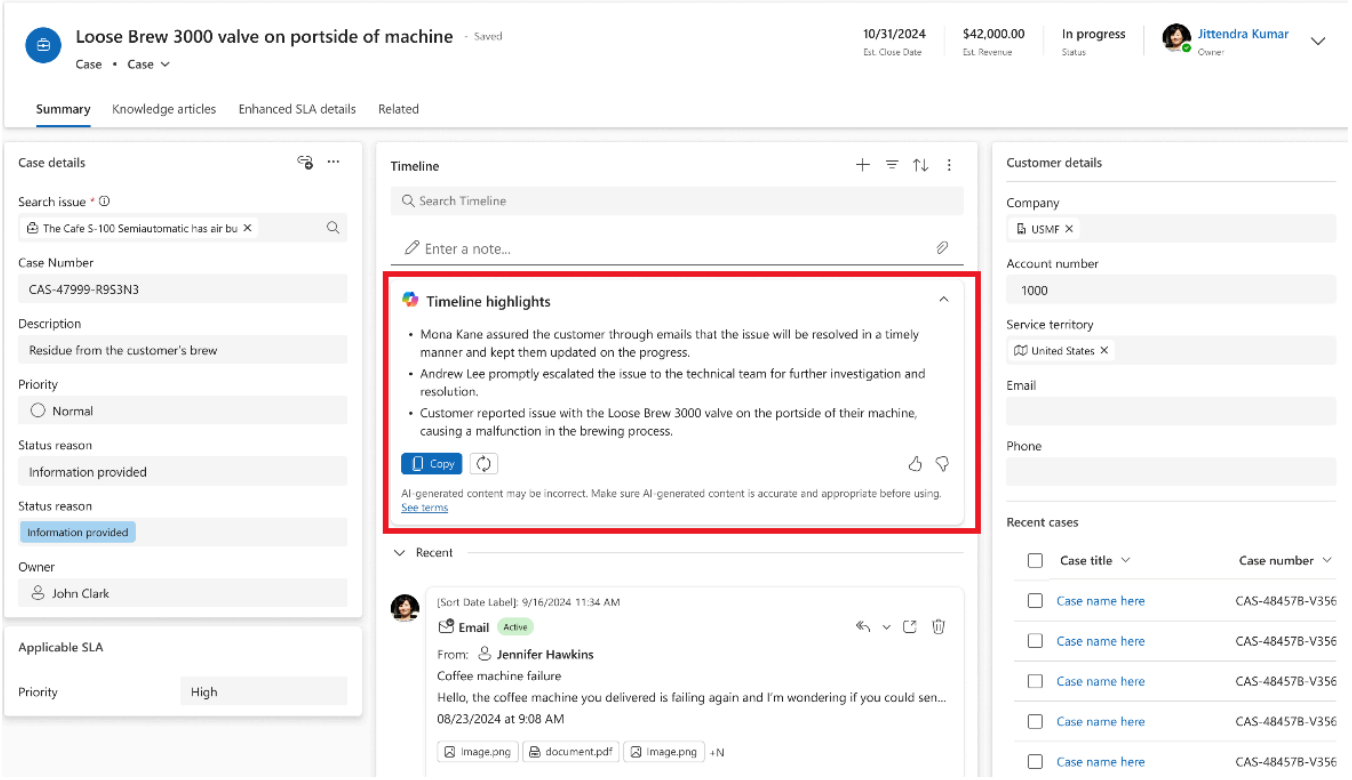
Last updated on 04/21/2026

Use AI-generated timeline highlights

If timeline highlights are enabled on your app, Copilot creates an AI-generated, bulleted list that populates at the top of the timeline. You can use timeline highlights to quickly get caught up on a record's status.

The highlights include a record's main activities, such as details about who you or the previous user interacted with, the reported issue, and actions taken to try to resolve the issue.

You can copy the timeline highlights by selecting **Copy**.



> **Important**
>
> This feature is only available with Power Platform environments with Dataverse that have the **Enable Dynamics 365 apps** setting enabled.

Last updated on 02/04/2026

Use Copilot case summary in model-driven apps

07/08/2025

Copilot case summaries help users quickly understand the context of a case and resolve customer issues more efficiently. The case summary includes key information such as the case title, customer, subject, product, priority, case type, and description.

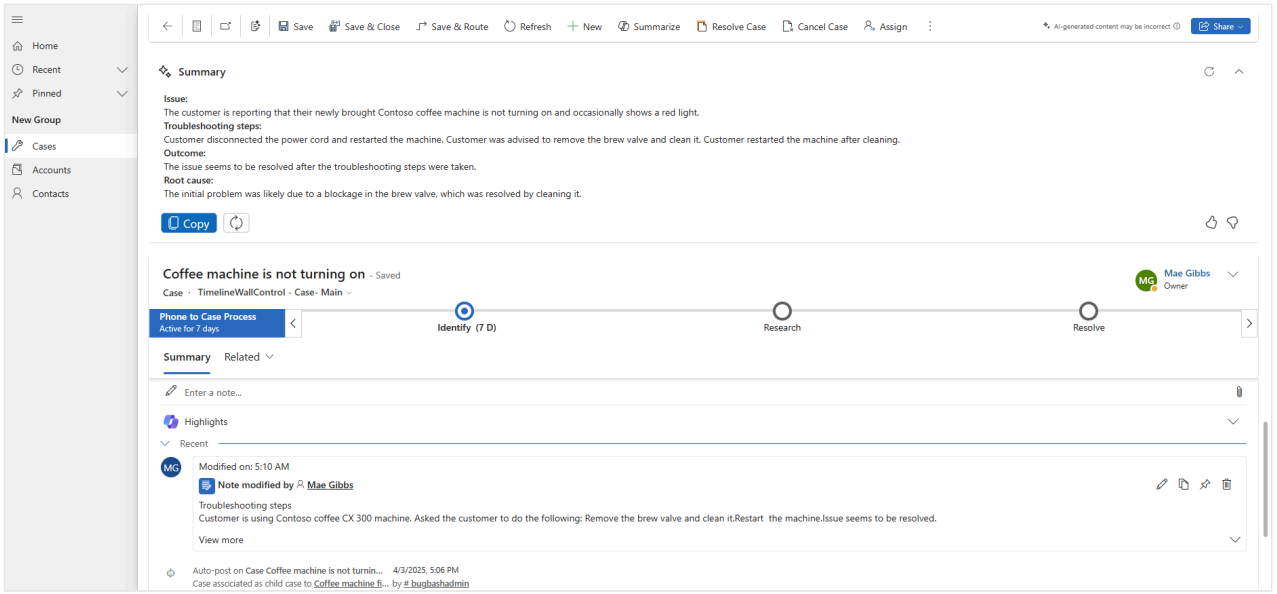
View case summary card

Copilot case summary is enabled by default for all model-driven apps that use the **incident** table. When you open a case record, the case summary card appears. When you open the case or custom record in a new session the summary card is expanded by default until you collapse it for the first time. The card then remains in collapsed state

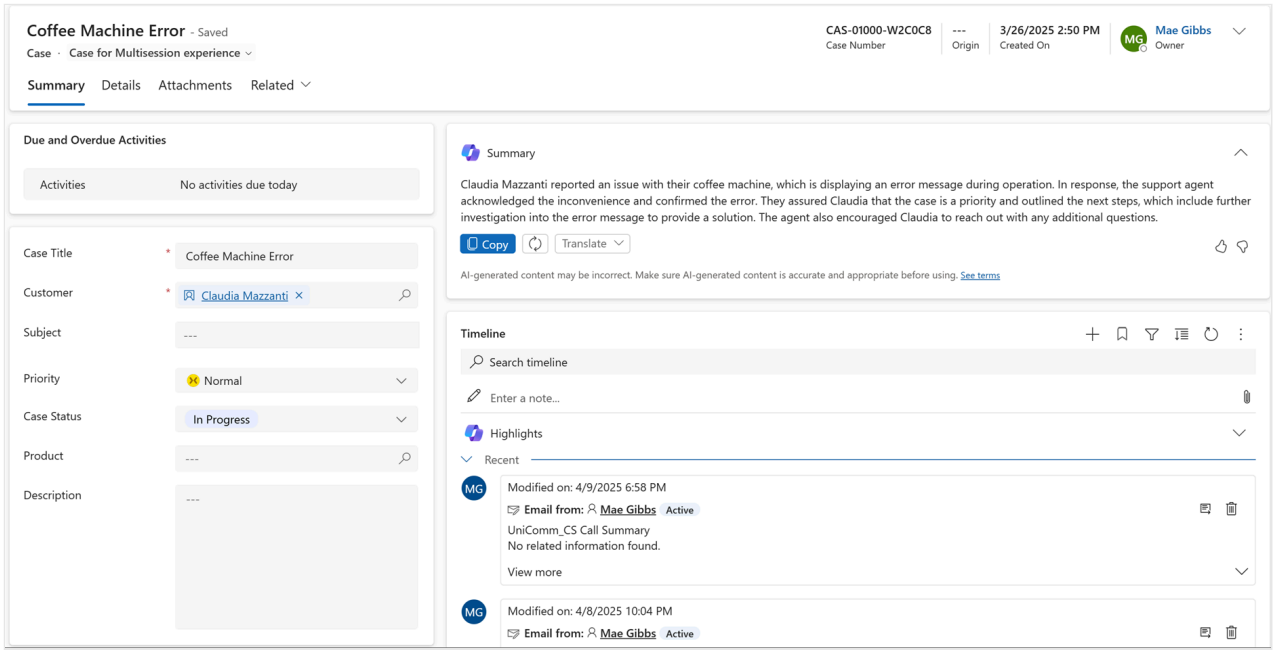
Based on the case form configured, users see the summary either on the top of the case form or within the form.

> ⓘ **Note**
>
> Users won't see case summary cards on case forms
>
> - By default, if their organization has opted out of the automatic enablement of Copilot feature.
> - If case summary is disabled in the experience profile.

- Users can see the case form displayed on the top of all case forms except **Case for Interactive experience**, **Enhanced full case form**, **Case**, and **Case for Multisession experience** forms. Users can copy the summary, refresh it, regenerate the summary, and provide feedback.



- Users can see the case form displayed within the case form for **Case for Interactive experience**, **Enhanced full case form**, **Case**, and **Case for Multisession experience** forms. Users can copy the summary, translate the summary into multiple languages, refresh it, regenerate the summary, and provide feedback.



- If custom record summary is enabled, users see the custom record displayed on top of the form by default.

Supervise agents in model-driven apps with agent feed (preview)

[This article is prerelease documentation and is subject to change.]

> **Important**
>
> Starting May 01, 2026, agent feed will support those agents only that use the Power Apps MCP server to create tasks. Make sure that your agents are properly onboarded to the Power Apps MCP server by this date to continue using the agent feed. If your agents don't use the Power Apps MCP server, the agent feed won't appear in your model-driven app.

Agent feed enables human-agent collaboration in apps by surfacing agent-generated, actionable tasks that users can review, validate, and complete. By using agent feed, apps act as a collaboration surface where users supervise and interact with agent work through a unified, task-based feed.

Prerequisite

The agent feed shows when at least one agent uses the Power Apps MCP server and is supervised in the app. Learn how to Add agents to an app.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is being gradually rolled out across regions and might not be available yet in your region.
> - This feature is only available in English language and replaces older Microsoft Copilot Studio activity-based agent feed version.

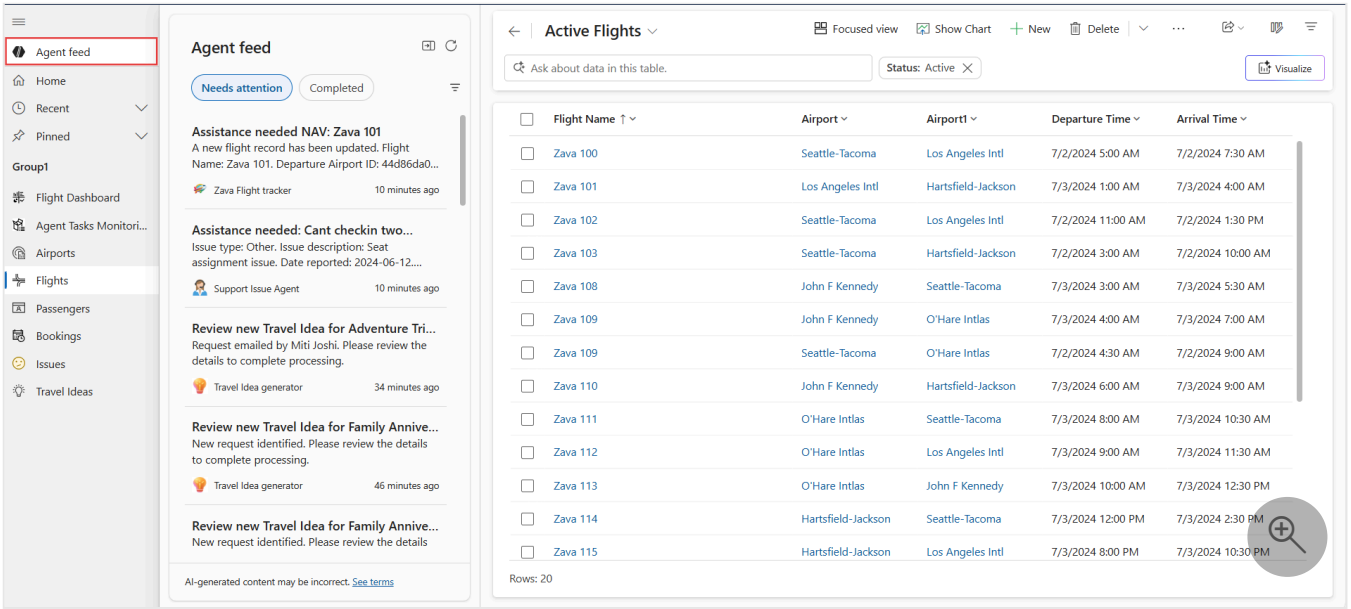
> **Note**
>
> Currently, all users who have access to the Agent Task table can see agent feed items in apps. To prevent unintended exposure, don't configure agents to log tasks targeted at specific users.

Use agent feed to supervise agents

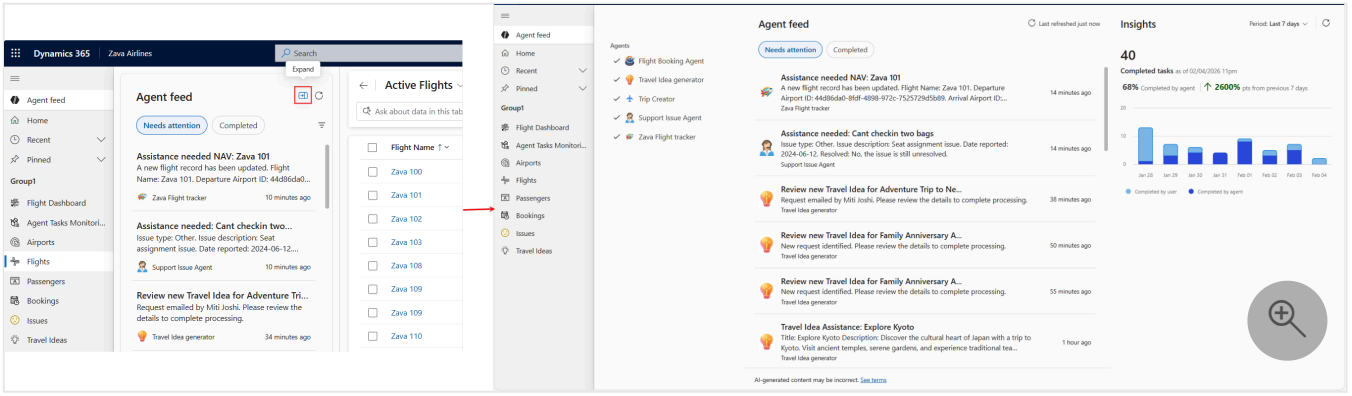
The enhanced agent feed is powered by the autonomous agents using Power Apps MCP server. Power Apps MCP server tools enable following two core human-agent collaboration patterns:

1. Autonomous Dataverse record creation and updates with human-in-the-loop approval and review.
2. Agent requests for human assistance and logging tasks for human review. In agent-enabled apps, user focus shifts from doing the work to supervising and prioritizing agent-driven work. Agents help with automations and organize work, ensuring business experts remain involved in decision-making and critical actions.

To allow users to view an agent's activity in the app, the agent must connect to Power Apps MCP server. The agent feed appears at the top of the site map in an app.

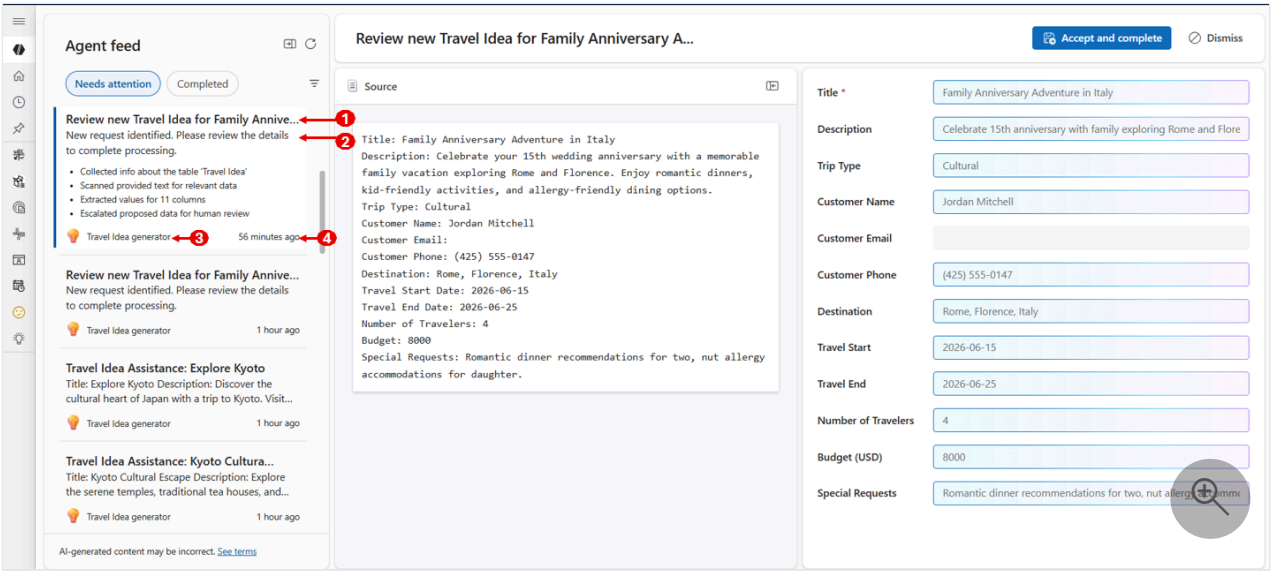


You can view the feed in the side pane or expand it to full screen.



The feed shows a list of agent tasks. Each item can include:

- Title of the task
- Description of the tasks with optional steps
- The agent's name and icon
- A timestamp of when the task was logged

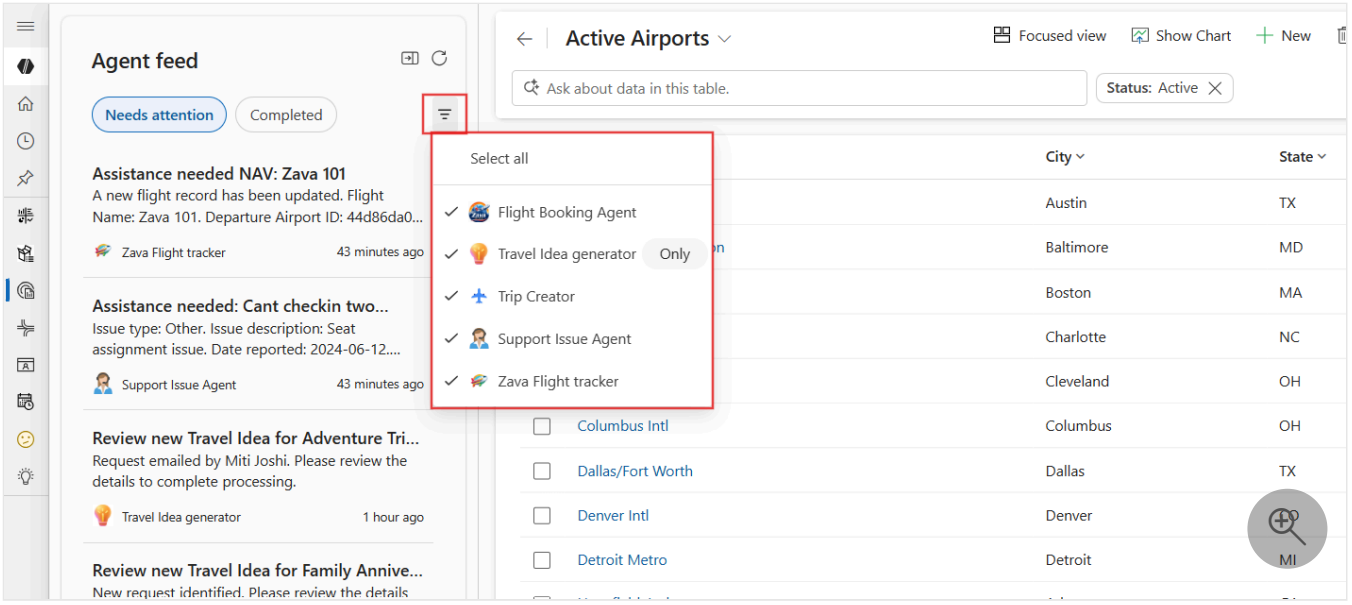


Activity is grouped into two sections:

- **Needs Attention**: Shows tasks that are waiting for your input or need your attention. These tasks are logged by using the `request_assistance` or `invoke_data_entry` MCP tools.
- **Completed**: Displays all other agent feed tasks, including tasks the user completed and tasks the agent completed that are logged for review by using the `request_review` MCP tool.

By default, each section loads 20 agent tasks and lazy loading is enabled as you scroll. The sections are sorted chronologically by the last modification date of each action.

You can filter the feed by agents.

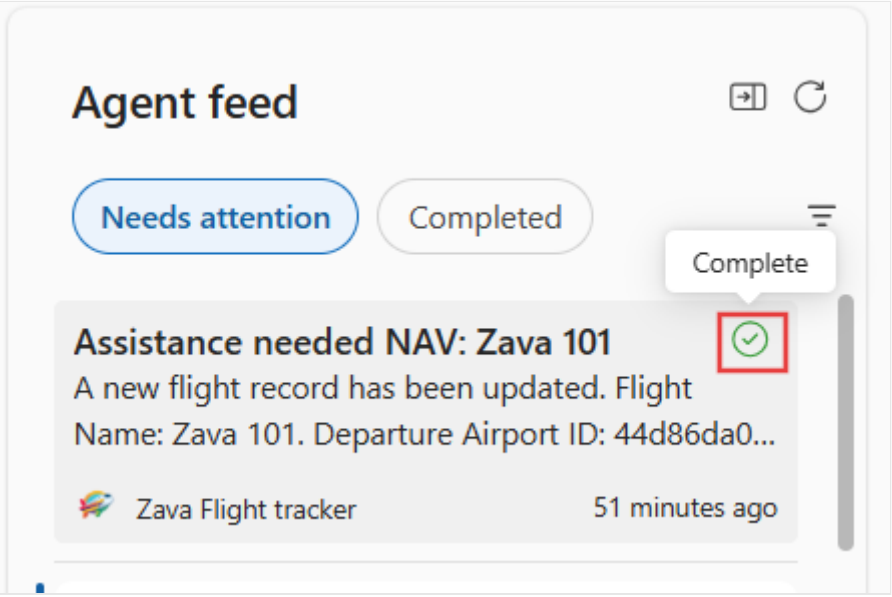


View the agent feed in a side pane to support multitasking or expand it to full screen for focused review.

Take action on agent activity

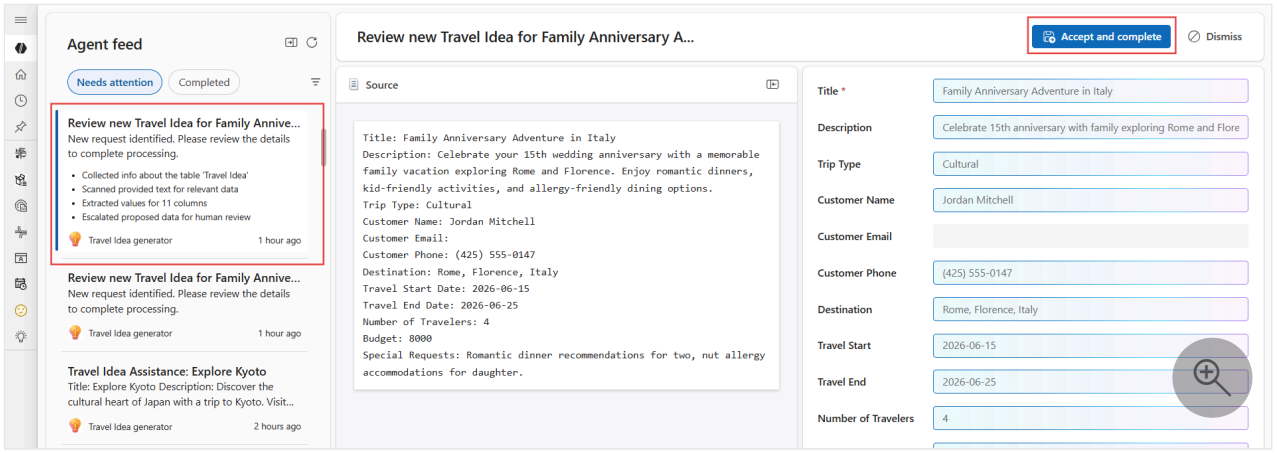
Take these actions on tasks under the **needs attention** tab in agent feed:

- **Complete**: Move a task from the **needs attention** section to **Completed** after completing the task the agent couldn't finish. This action is available for the agent tasks logged by using the **request_assistance** mcp tool.

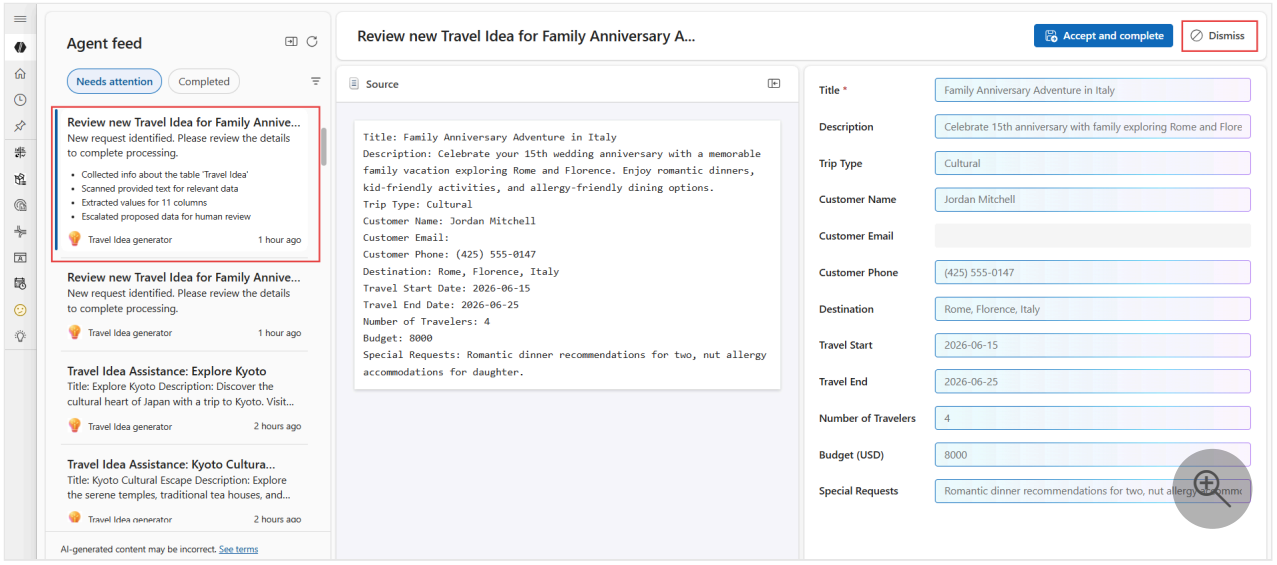


- **Accept and complete**: Move a data entry task from the **Needs attention** section to **Completed** after reviewing and accepting the data entry suggestions from

invoke_data_entry mcp tool. The **Accept and complete** button is available inside the agent pane associated with the agent feed task.

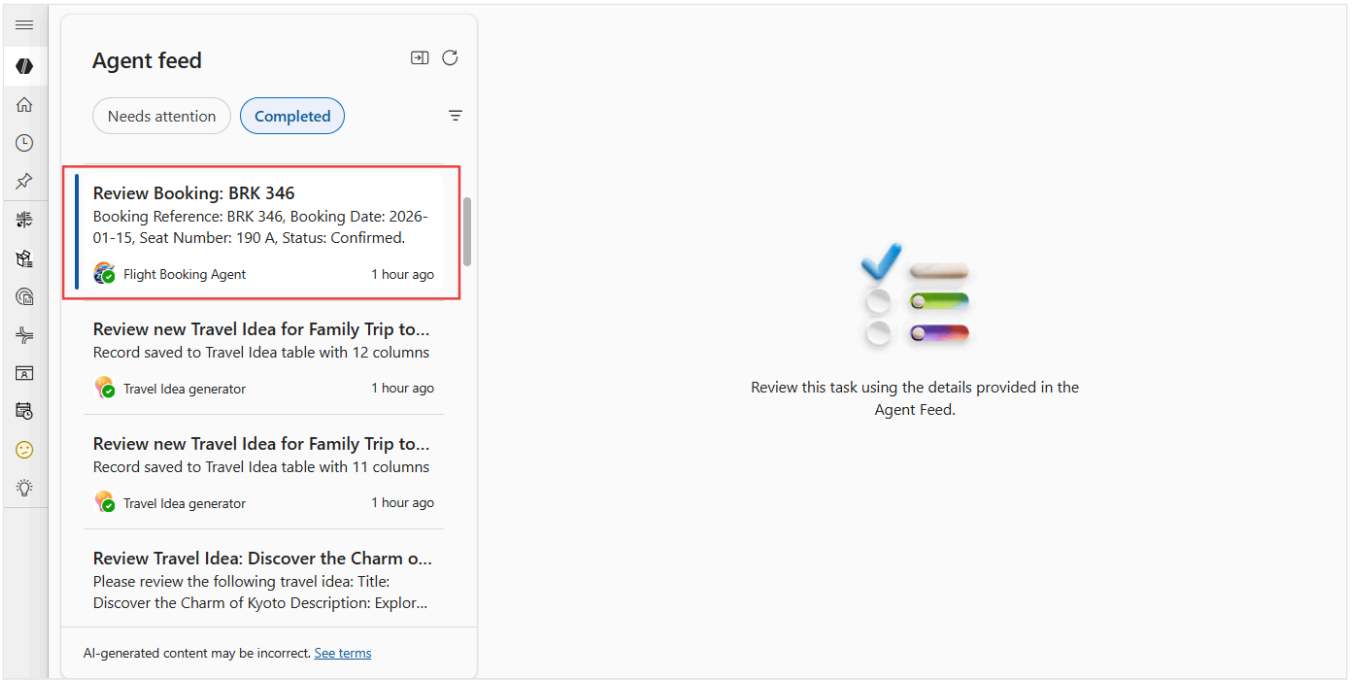


- **Dismiss**: Remove a data entry task logged via **invoke_data_entry** mcp tool from the **Needs attention** section if it's no longer relevant. The **Dismiss** button is available inside the agent pane associated with the agent feed task.



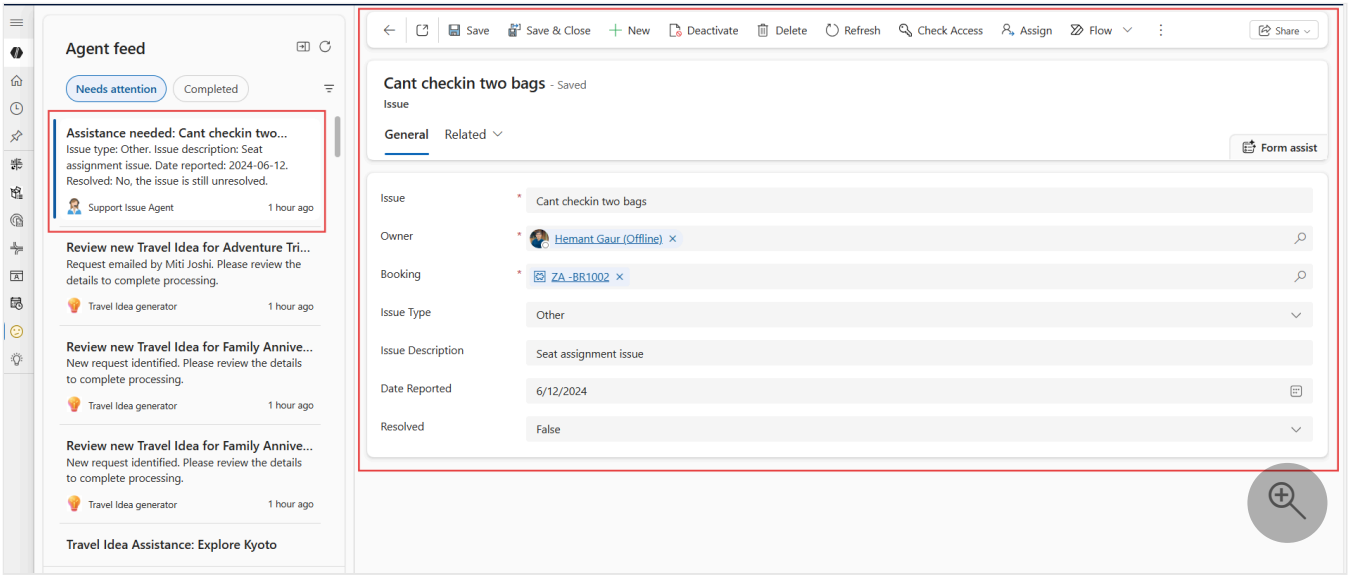
> **Note**
>
> You can perform all actions either from the main screen in the feed or from the details page of a specific action.

- **Request_review tasks**: Tasks logged via request_review mcp tool don't have any actions for the users. They're informational tasks completed autonomously by agent and can be found in completed agent feed tab.



Navigation from agent feed item

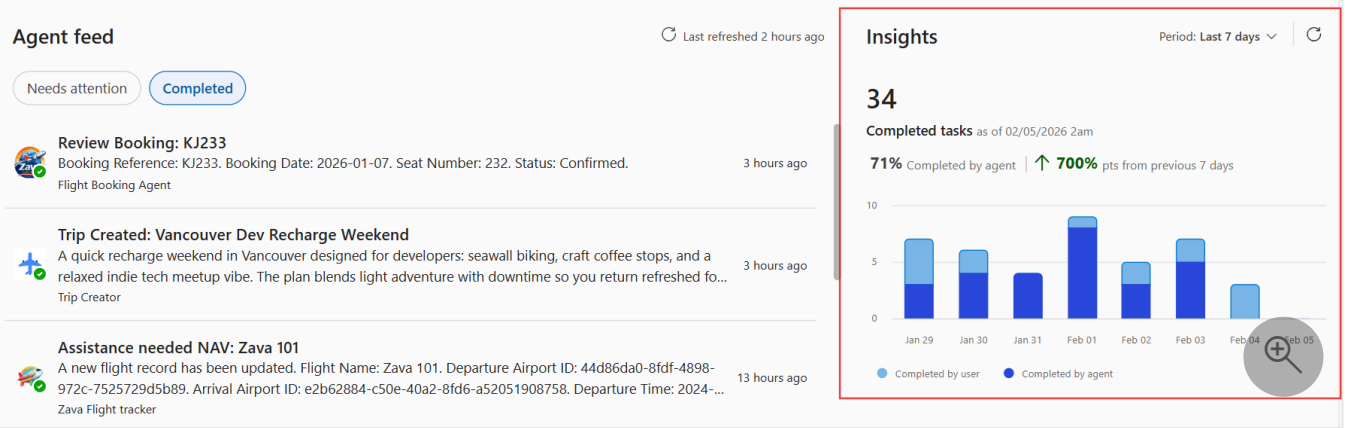
Agent feed tasks generated through the request_assistance or request_review MCP tools can reference a Dataverse record as a related navigation link in the agent description. This reference allows users to easily open and review associated data when interacting with the feed item.



Insights panel

The default insights panel complements the agent feed by giving users a clear, aggregated view of how their agents and users collaborate over time. While the agent feed surfaces individual tasks and actions in the moment, the Insights chart provides a historical, high-level representation of those interactions. This representation helps organizations understand the

true patterns of human–agent collaboration, whether users step in to provide missing context, guide agents through a complex workflow, complete tasks directly, or monitor agent activity. By visualizing these trends over time, organizations gain transparency into how their agentic workflows actually operate in practice, strengthening trust, identifying friction points, and revealing opportunities to improve system reliability and user experience.



The insights chart presents aggregated activity across the last 7, 14, or 30 days. It gives customers a time-based snapshot of collaboration between agents and users. Activity is categorized into two groups:

- Tasks completed by agents
- Tasks completed by users These at-a-glance aggregations offer a concise understanding of collaboration volume and balance over time. They help teams quickly assess how work is distributed across human and agent participants.

Related information

Add agents to an app

Customize Copilot Chat

Last updated on 04/07/2026

Use Copilot chat in model-driven apps

Copilot chat for model-driven apps in Power Apps is a next-generation AI assistant that helps you gain insights into the data in your apps through conversations in natural language. Copilot chat helps you boost your productivity through AI-powered insights and navigation assistance.

> **Important**
>
> - This feature is generally available in Dynamics 365 apps, but remains a preview feature in Power Apps.
> - Preview features aren't meant for production use and might have restricted functionality These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

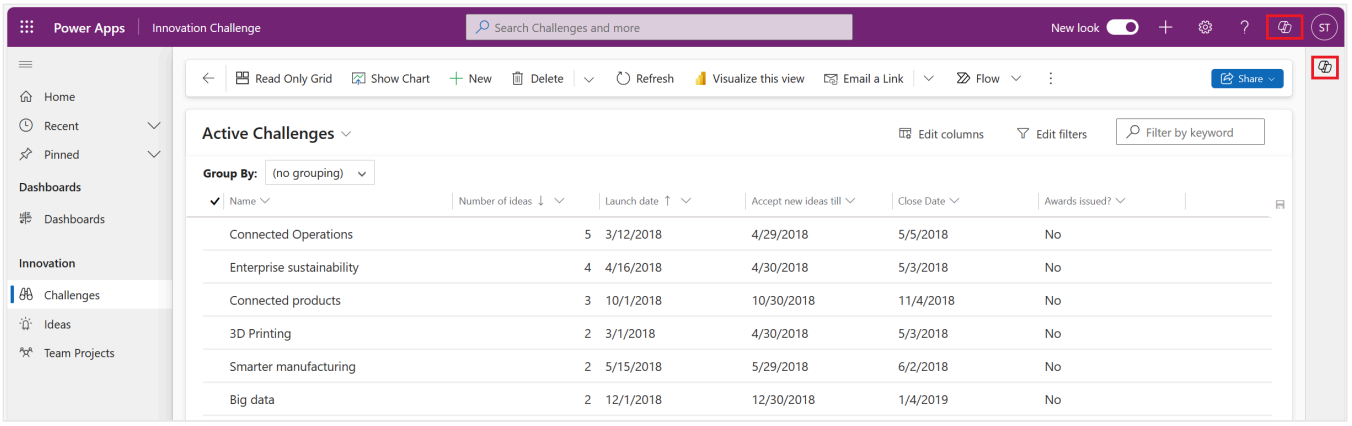
> **Note**
>
> Copilot chat in Power Apps is a preview feature that will be deprecated for model-driven apps in environments not enabled for Dynamics 365 apps, starting January 2026. We recommend switching to Microsoft 365 Copilot chat. During the transition period, makers can enable either one or both chat experiences. For more information, see Deprecation of Copilot chat in model-driven apps

Prerequisites

An administrator must enable Copilot chat in your application before it becomes visible in your app. More information: Add Copilot for app users in model-driven apps.

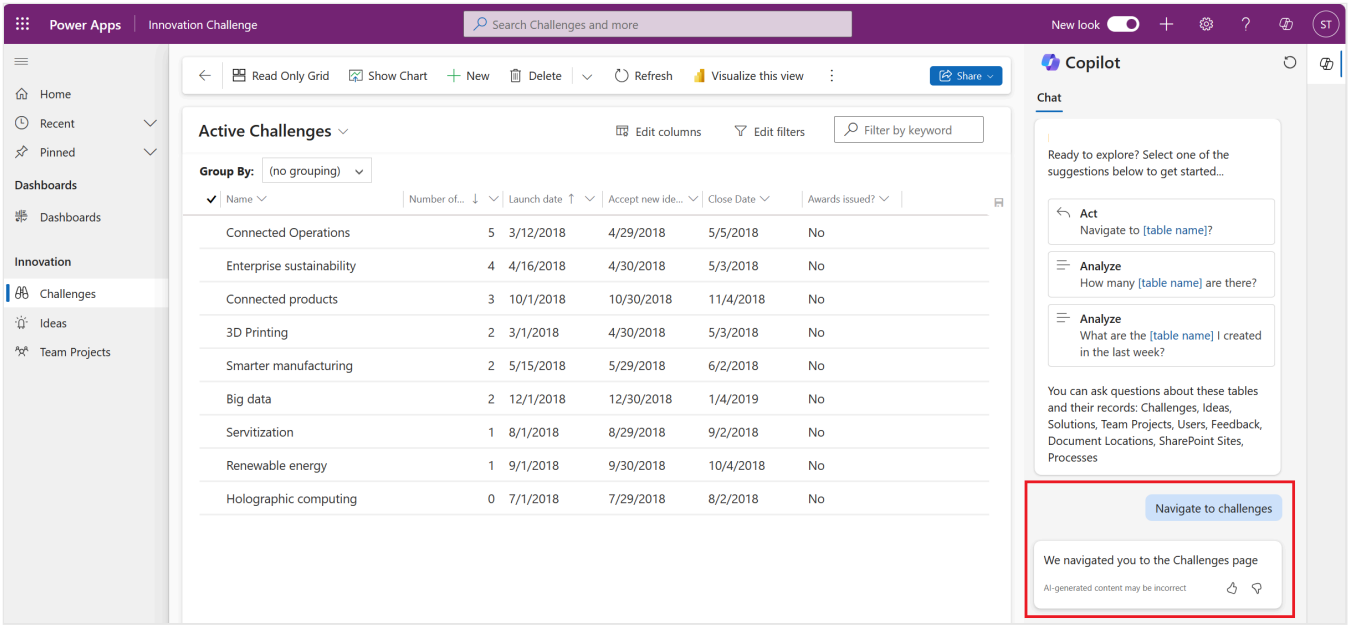
Copilot pane

After Copilot chat is enabled, you can access it through the Copilot icon near the upper-right corner of the page or through the Copilot icon at the right side of the command bar. You can expand or collapse the Copilot pane as you want. By default, the pane is expanded after Copilot chat is enabled.



Use Copilot to ask questions

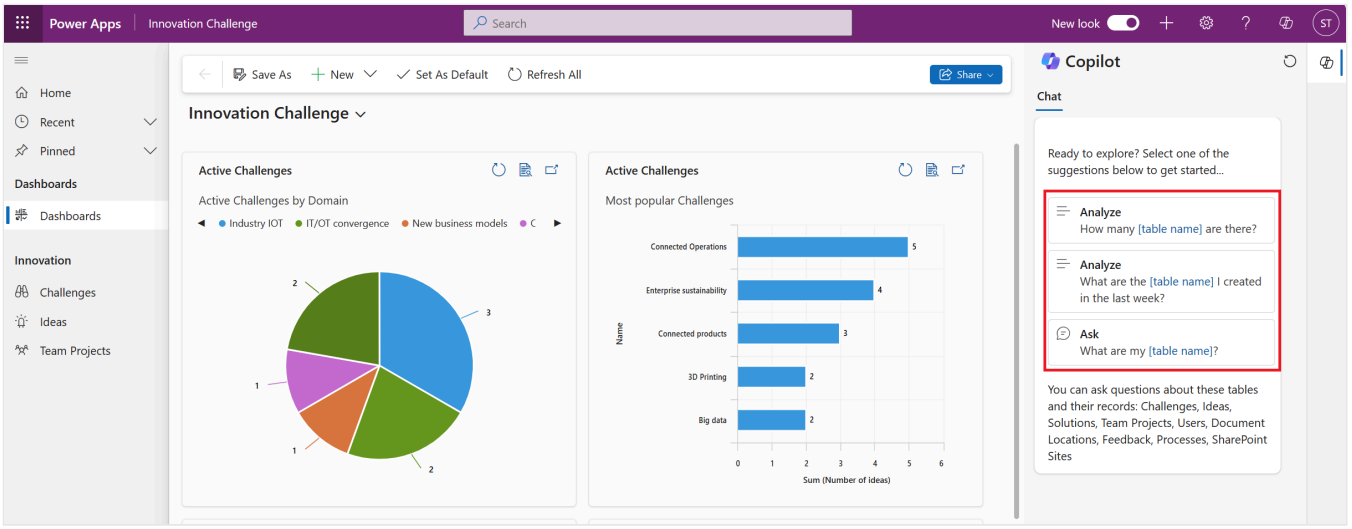
Copilot chat in model-driven apps can answer questions about the Dataverse table data in the app. It can also help you navigate the app. For example, if you enter **Navigate to challenges**, Copilot automatically opens the relevant page in the app.



Copilot chat suggested questions

To help you get started, Copilot chat can suggest questions for you to ask. You can edit and personalize these questions (also known as prompts) based on your app and the type of insight that you need from Copilot.

To view more suggested questions, select **View prompts** in the Copilot pane. Many of the suggested questions have placeholders that you must replace with appropriate text to complete the question.

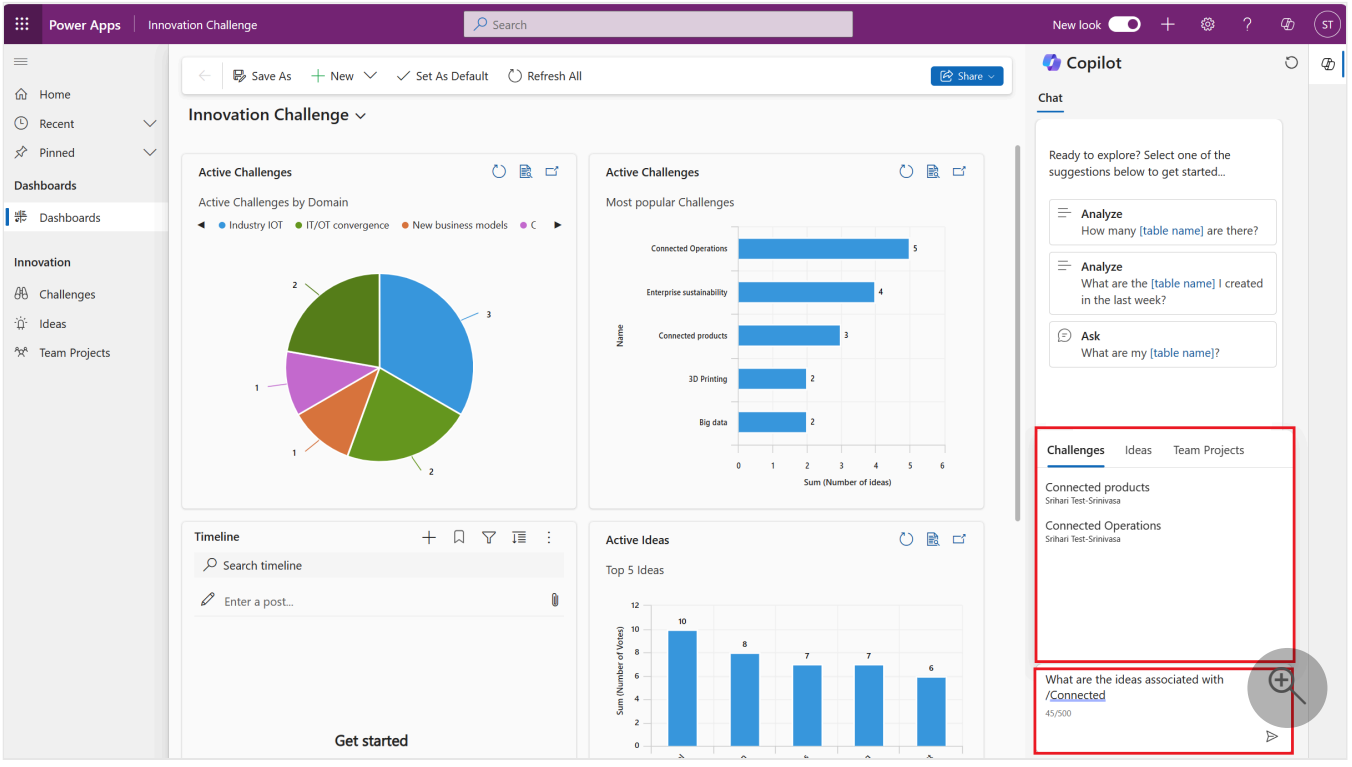


Use the record picker

When you ask a question, you can use the record picker to select the object of your question. The record picker helps Copilot chat understand the context of your question and provide a more accurate response.

Enter a slash (/) to open the record picker, and then enter the name of the record. Select the record that you want, and complete the question to make it meaningful. Finally, submit the question to Copilot to view the response.

For example, you ask Copilot "What are the details for Contoso?" Many accounts might have "Contoso" in their name, such as Contoso, Inc., Contoso Co., and Contoso NW. What you really want are the details for Contoso, Inc. The record picker helps Copilot scope the question to the right record for a more relevant and accurate response.



The record picker is available in Dynamics 365 Sales and Power Apps model-driven apps.

> **Note**
>
> - By using the record picker, you increase the likelihood that Copilot chat can understand your question and therefore respond with an accurate answer.
> - The record picker requires that Dataverse search is enabled and configured for the type-ahead search capability.

Provide feedback

Unless your admin turned off the feedback feature, every Copilot response includes a **Like** button ("thumbs up" icon) and a **Dislike** button ("thumbs down" icon). For each response, you can select one of these icons to provide feedback that helps us improve Copilot chat. Select the "thumbs up" or "thumbs down" icon as appropriate. Optionally, provide feedback in your own words about what you liked or didn't like about the response. When you're done, select **Submit**.

Last updated on 01/08/2026

Basic navigation in a model-driven app

Article • 01/16/2025

This introduction explains how to find and open an app, and how to work with its common user interface elements including lists, forms, and business processes.

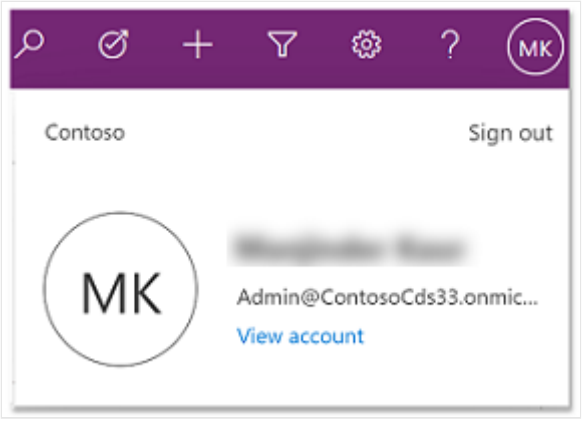
Navigating among apps, areas, and tables

A model-driven app is built out of applications (apps), areas, and tables.

- *Apps* provide a collection of functionalities for accomplishing a specific class of activity, such as managing your accounts and contacts. Use the app-selector menu to navigate between the apps that are available to your organization.
- A *work area* is a subdivision of an app, dedicated to a specific feature. Each work area provides a targeted collection of tables for working in that area. In some cases, the same table appears in more than one area (or even more than one app). The Contact and Account tables, for example, appear in various apps and work areas. Use the work-area menu to navigate between work areas for your current app.
- *Tables* represent a specific type of data, such as contacts or accounts. Tables use a structured data format, which defines the collection of columns available to the table. Each table consists of a collection of individual rows. For example, for the Contact table, each row describes a single person, and each row includes a collection of columns such as first name, last name, and email address. Tables normally present two views: a list view, which is typically a table listing available rows; and a form view, which shows all available data and settings for a single row. Use the side navigator to move between tables in your current work area.

User info and sign out

Select the user menu to view your account information or sign out. Go to your account page by selecting the **View account** hyperlink.

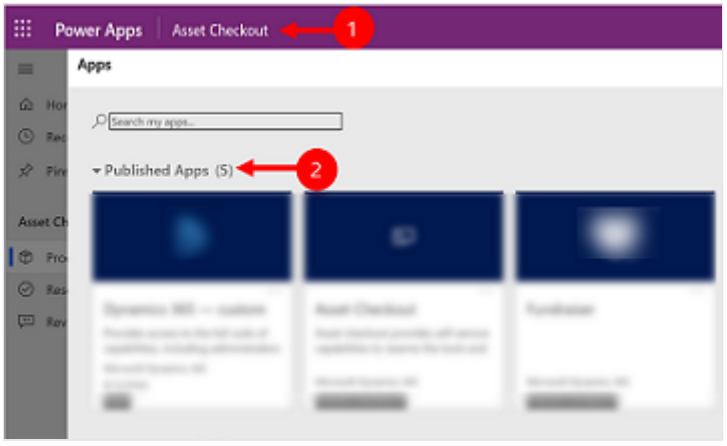


Move between apps

Use the app-selector menu to switch between apps.

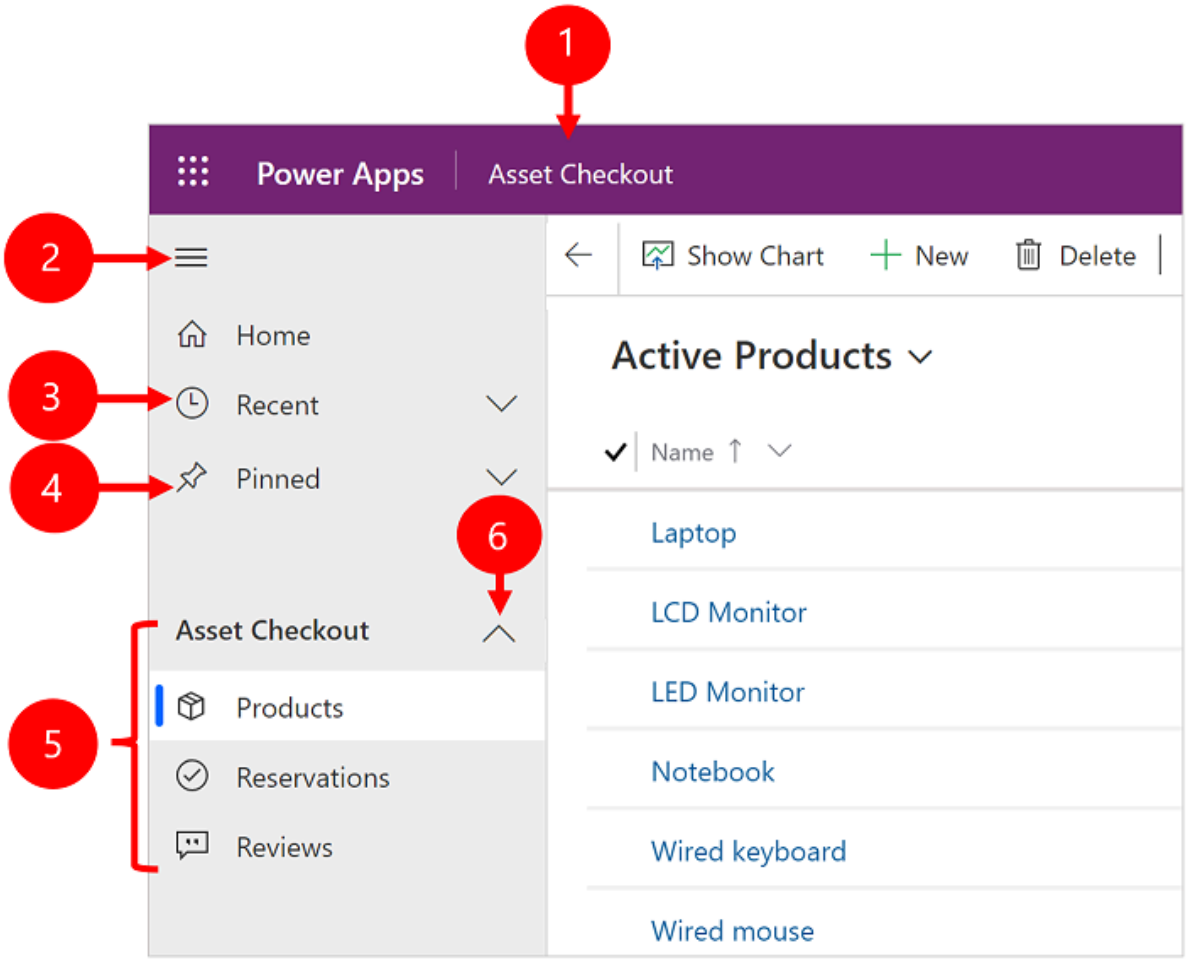
1. Select the current app name.

2. On the Apps page, select an app. You'll only see apps for your organization.



Move between pages and rows

It's easy to get around and get back to your favorite or most-used rows. The following illustration shows the primary navigation elements.



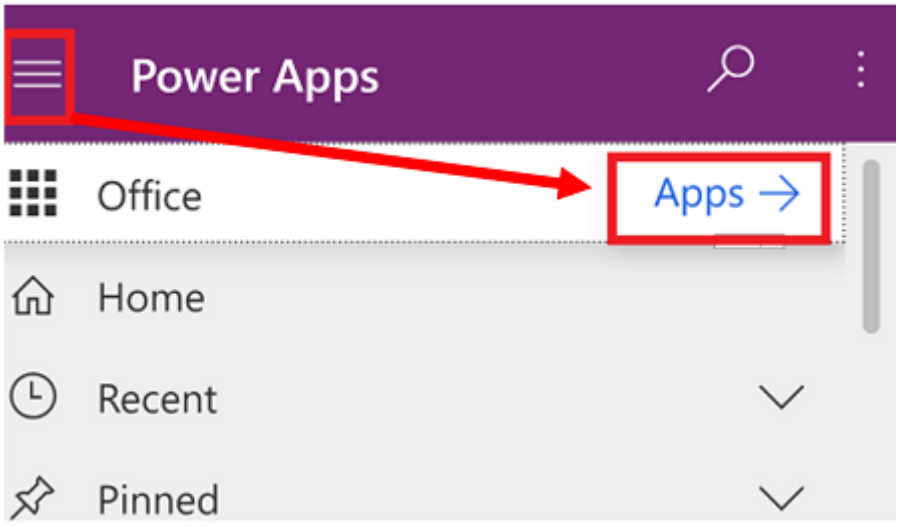
Legend

1. **Click to change app**: Select the current app name to change to a different app.
2. **Collapse/expand site map**: Select this to collapse the navigator to allow more room for the main part of the page. If the navigator is already collapsed, select this button to expand it again.
3. **Recent**: Expand this entry to view a list of rows you were recently using. Select a row here to open it. Select the push-pin icon next to a row listed here to add it to your pinned rows. This is limited to displaying the last 10 rows of pages most recently loaded across apps for the user within this organization.
4. **Pinned**: Expand this entry to view and open your favorite pinned rows. Use the **Recent** list to add rows here. Select the remove-pin icon next to a row listed here to remove it from this list. The number of pinned rows displayed is limited to 25, but no more than 15 rows from a single table are displayed.
5. **Page navigation**: This area lists each table and dashboard page available for the current work area. Select any entry here to open the named dashboard or list view for that table.

> **Note**
>
> When a user sets a dashboard as default (via the **Set as default** action in the command bar), all page navigation to dashboards will take the user to the default dashboard, which may be a different dashboard than the named dashboard the user selected.

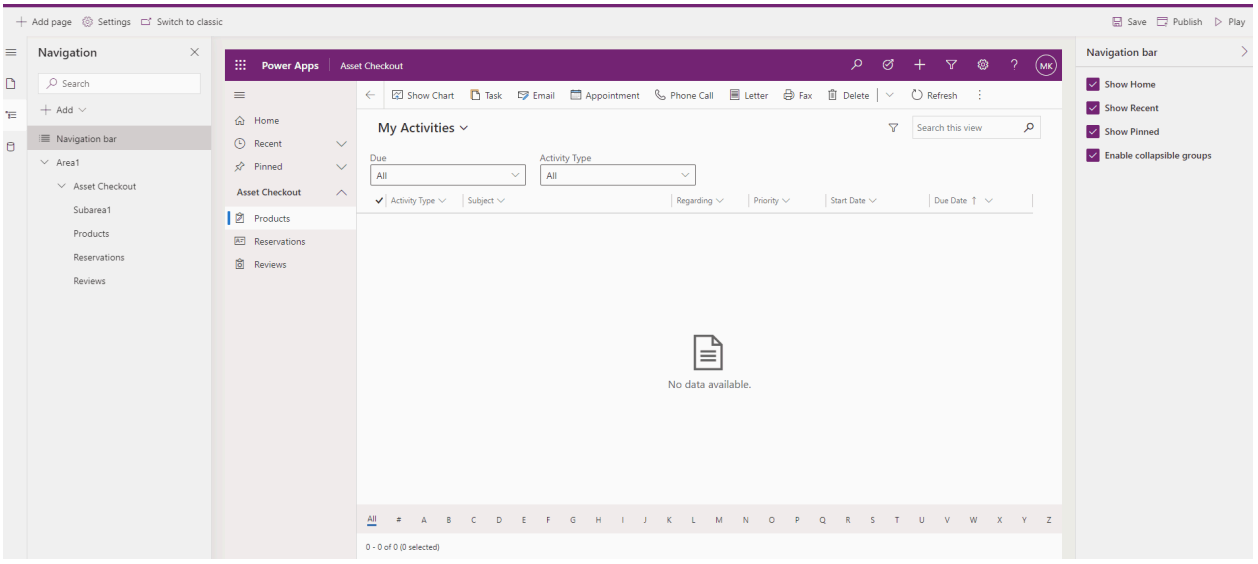
6. **Collapsible groups**: Your app maker can enable collapsible groups. For information on how to customize the navigation bar, see Overview of the model-driven app designer.

If your screen is zoomed in too much or is narrow, then you won't see your current app name which lets you change apps. To access the change app button, select **Site map** > **Apps**.



Hide or show the Home, Pinned, Recent, and collapsible groups

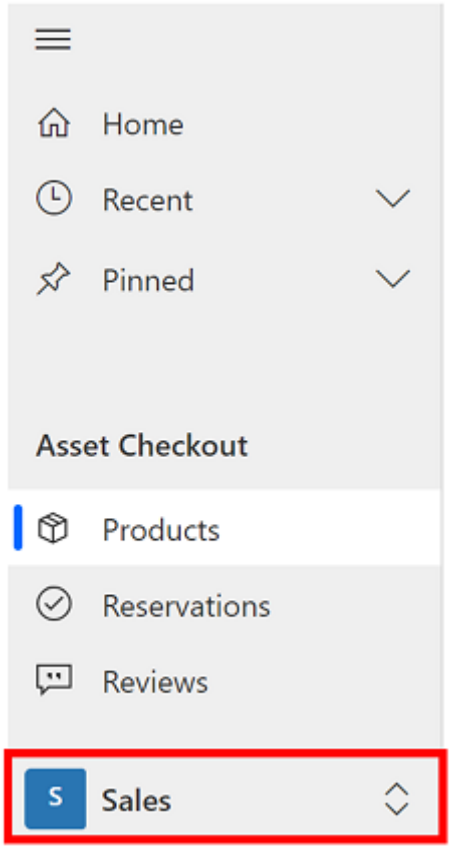
Your site map might look different if it's been customized. Your app maker can customize the navigation bar to show or hide the **Home**, **Recent**, **Pinned** buttons in the site map. An app maker can also make groups collapsible. For information on how to customize the navigation bar, see Overview of the model-driven app designer.



Area switcher

In the left-hand navigation pane, the area switcher appears at the bottom left only when a single model-driven app has more than one area in the sitemap definition.

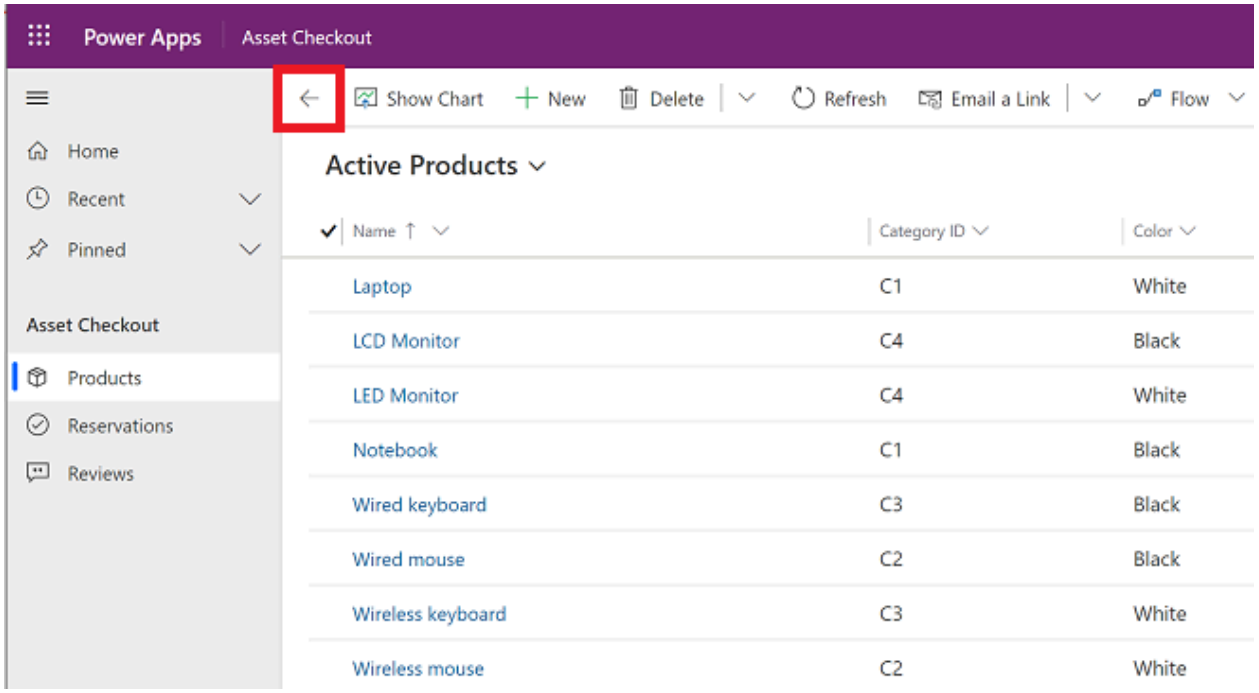
Open this menu to move to another work area. The current work area is named here.



Go back

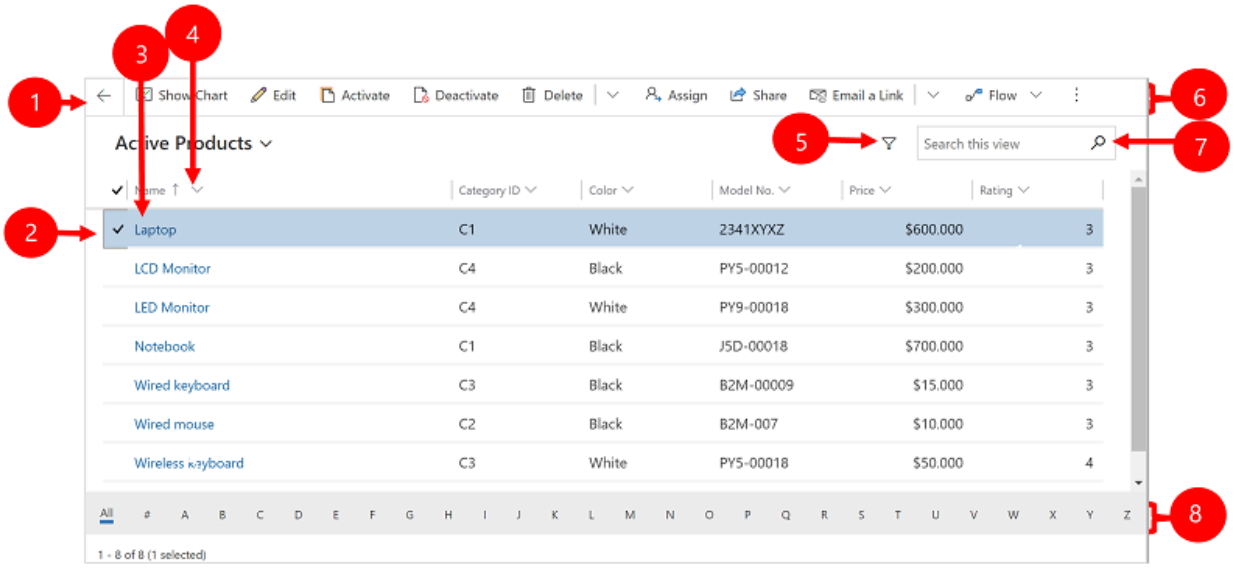
Use the back button for form, view, and dashboard pages on the command bar to go back to the previous page.

To go back to the previous page, select the **Go back** button.



Working with list views

Usually, when you first open a table, you'll see the list view, which shows a list of rows belonging to that table, formatted as a table. For example, if you open the **Products** table, you'll see a list of accounts.



Legend:

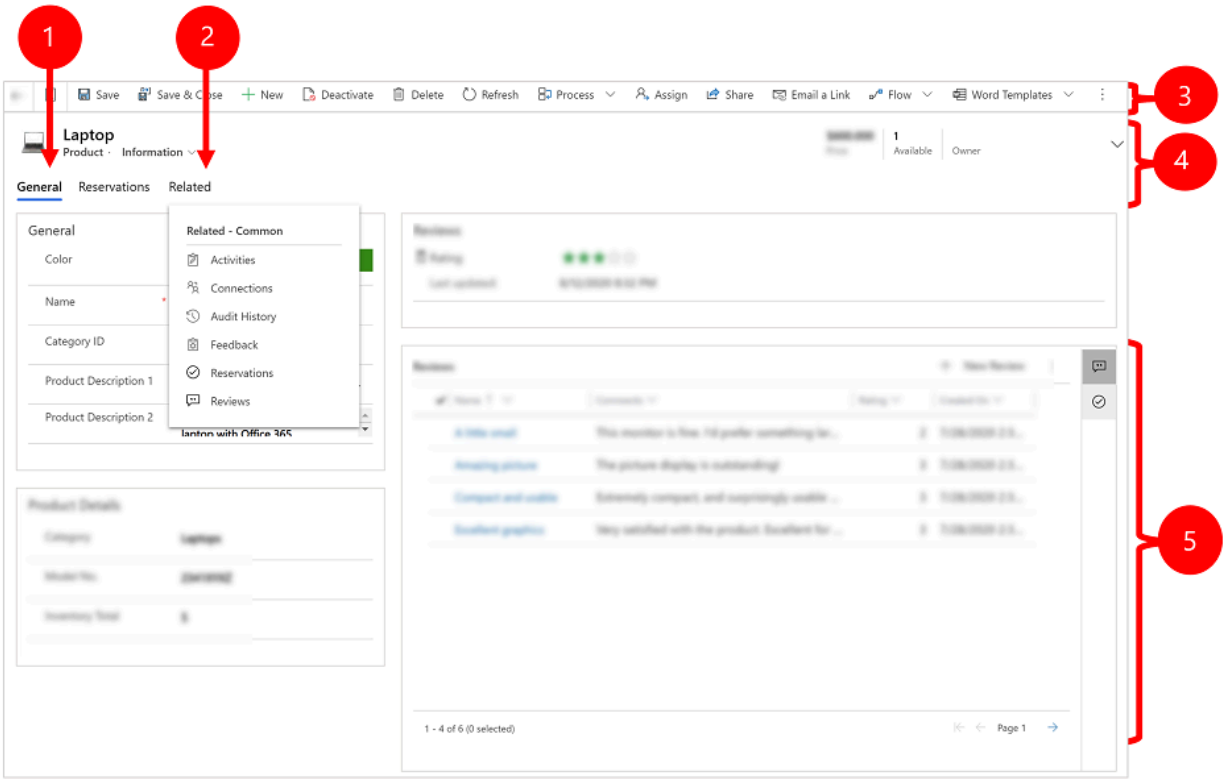
1. **Go back**: Select to go back to the previous page.
2. **Select rows**: Select one or more rows by placing a check mark in this column. Depending on where you're working, you might be able to apply a single operation to all the selected rows at once by using buttons in the command bar.

Note, when you use the **Email a link** command on the command bar you can only select up to 10 rows to send via email.

3. **Open a row**: Select any row in the list to open its row view, which shows all the details about the row. Usually you select from the **Name** column to open a row from the current table. Some tables provide links to rows from related tables in other columns (such as a related contact).
4. **Sort or filter the list**: Select to sort the list by values in that column or filter the list by values in that column. An arrow in the column heading indicates which column is being sorted and in which direction.
5. **Open advanced filtering panel**: See which filters are applied to the current view and add or modify view filters.
6. **Command bar**: Use the commands in the command bar to operate on rows in the list and do related actions. Some commands (such as **Delete**) require that you first select one or more target rows by placing a check mark in the leftmost column, while others operate on the entire list. You can export the list to an Excel workbook (possibly based on a template), open charts and dashboards, and more, depending on the type of rows you're working with.
7. **Search the view**: Enter text in the search column above the list to show only those rows in the current view that contain your text.
8. **Jump bar and paging**: Select a letter to show only those rows whose names start with that letter. If the list contains more rows than can be shown on one page, use the paging arrows at the bottom of the list to move forward and backward through the pages.

Working with row views

Row views show all the details about a single row and sometimes also provide special features for working with it. Usually you'll open a row view by selecting a row that appears in a list view, but you can also open a row view by following a link from a related row.



Legend:

1. **Tabs**: Most row views are divided into tabs. Each tab provides a collection of related columns from the row. When tabs are available, they're listed below the row name. Select any tab name to go to that tab. The current tab is shown underlined.
2. **Related**: Nearly all types of rows show a **Related** tab after you've saved them at least once. This tab is actually a drop-down list that you can use to find other types of rows that use or reference the displayed row. When you choose a table name from the **Related** drop-down list, a new tab named for that table opens, showing a list of all related rows of that type. The **Related** tab remains available, and you can still use it to find other types of rows that reference the current one.
3. **Command bar**: Use the commands in the command bar to operate on the current row or perform a task related to the row. The available commands vary based on the row type, but you can typically use the command bar to save your changes, delete the row, refresh the page, email a link to the row, reassign the row owner, or export the row by using a Word template.
4. **Heading bar**: Some row views display a few especially important columns in the heading bar, opposite the row name. These are typically columns that are fundamental to working with rows of the current type (such as a row name or row owner).
5. **View and edit all column values**: In the main body of the row view, you'll find all of the columns related to the current tab, form view, and row type. Columns marked with a red asterisk are required, and you can't save the row without their having valid values. Columns marked with a blue plus sign are especially important or

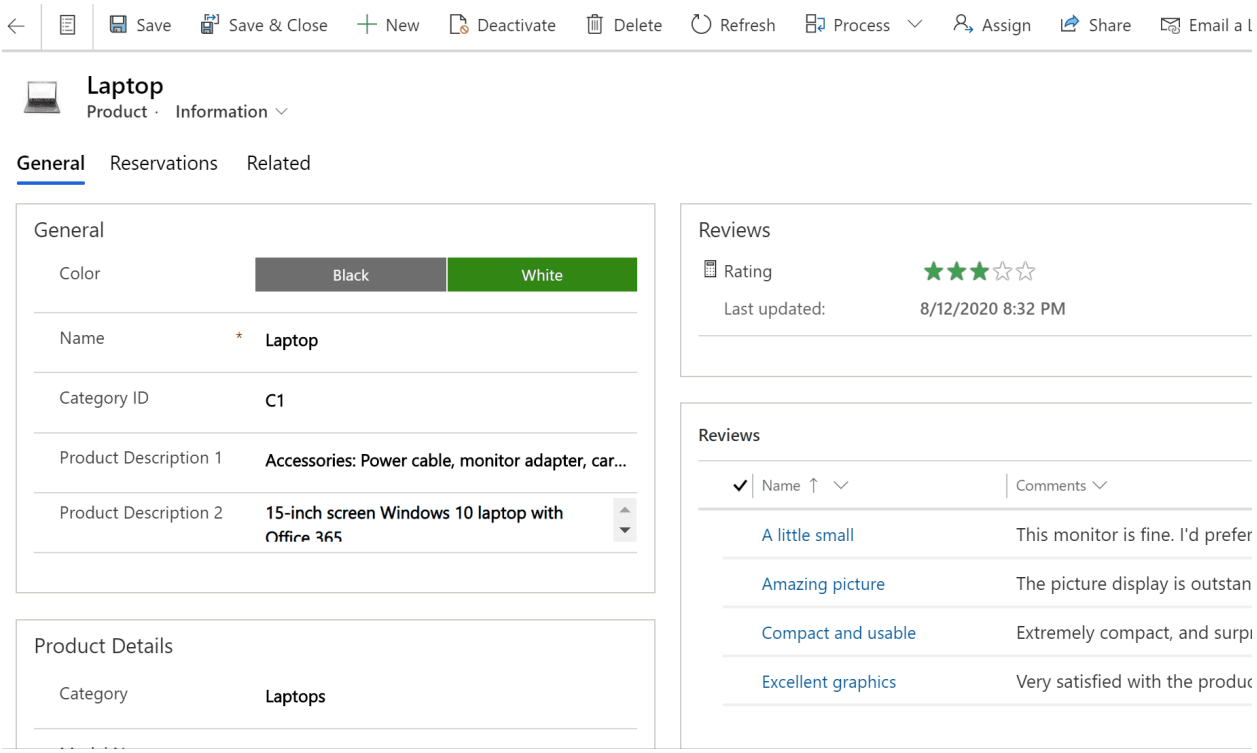
recommended, but aren't strictly required. Columns showing a lock icon are read only and can't be edited.

Row set navigation

Navigate through multiple rows by using preset views and queries. The row-focused navigation improves productivity by allowing users to jump from row to row in the list and easily navigate back without losing their working list.

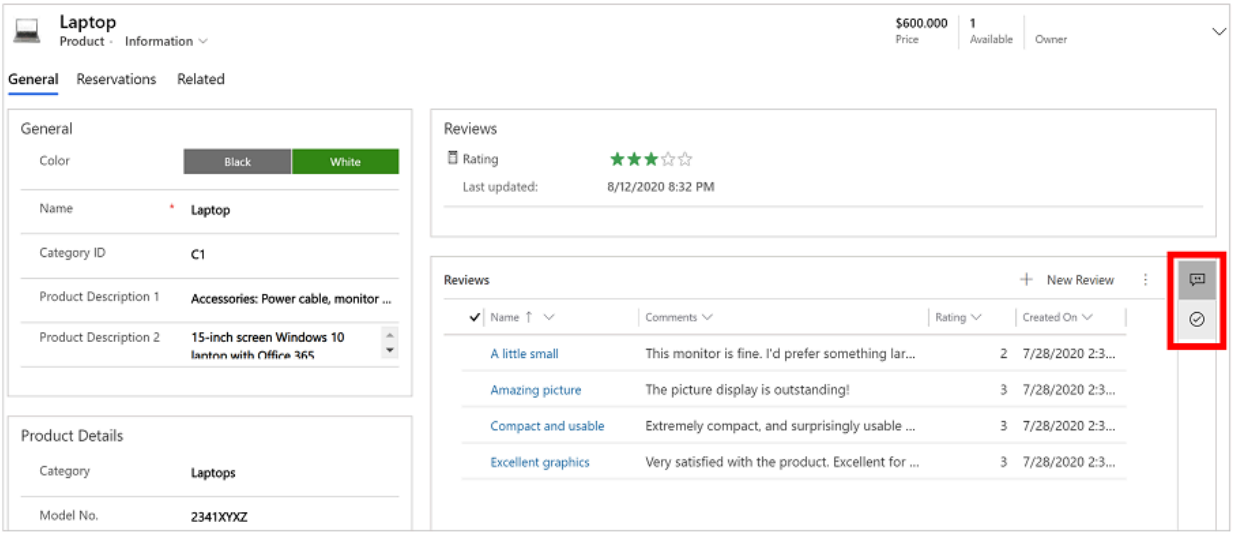
The number of rows that you see in the row set navigation pane is based on the number of rows that your system administrator has defined for the subgrid.

1. To use row set navigation, open a page with a list of rows.
2. Open a row and then select **Open Row Set** and then select a row from the list.



Reference panel

The reference panel is a great way to get work done without moving away from the screen you're on. You can look up other related items—such as reviews or reservations for a product—within the context of the row you're viewing, without having to navigate to other screens.

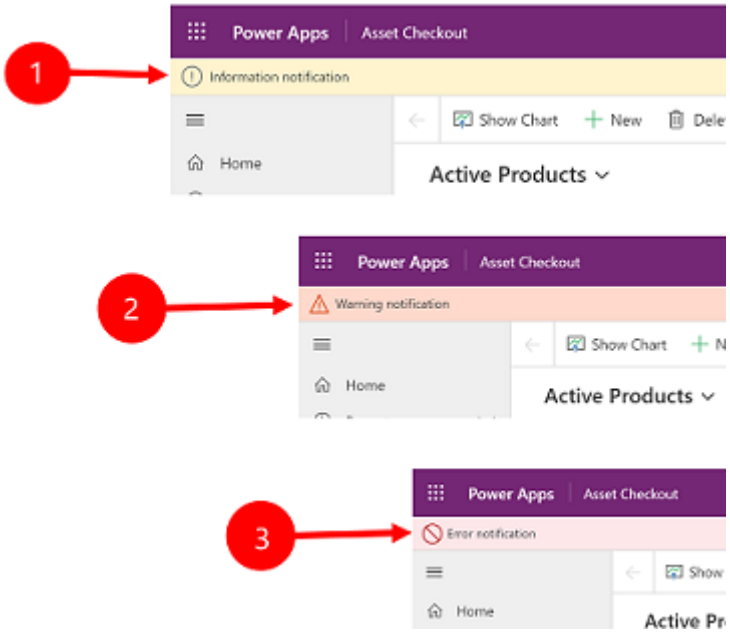


Watch the How to use the Reference panel in the Customer Service Hub video to learn more about the reference panel.

App message bar

The app message bar shows three types of notifications: informational, warning, and error.

When you select the error notification, it will take you to the column on the form where the error occurred.

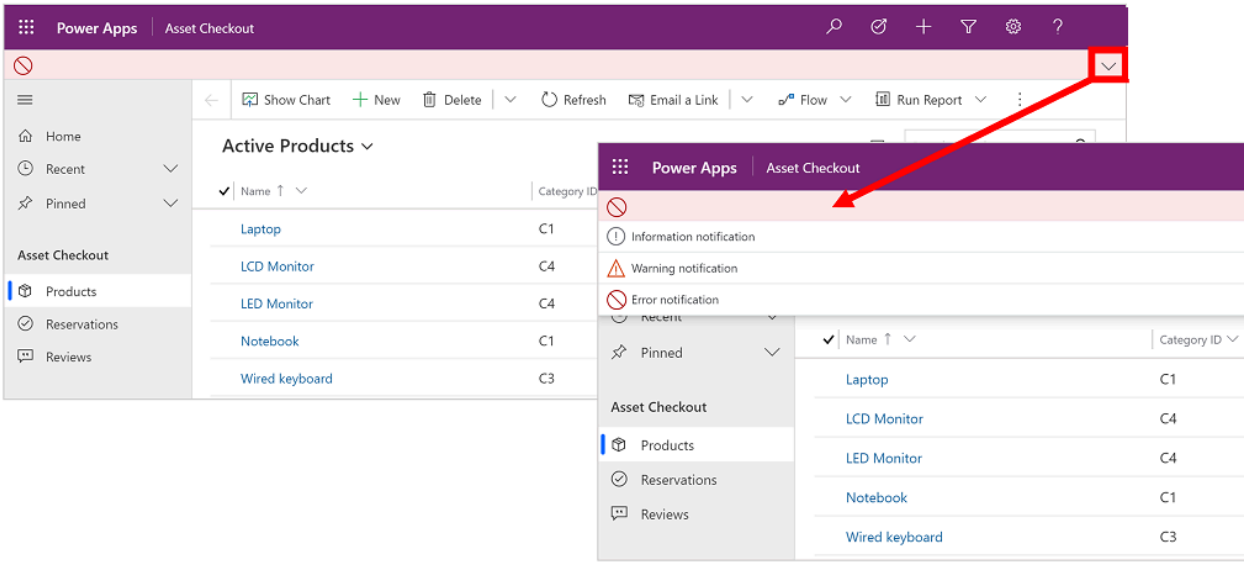


Legend:

1. **Information notifications** ⓘ: The notification is informational.
2. **Warning notifications** ⚠: The notification is a warning.
3. **Error notifications** ⛔: The notification is an error.

Multiple notifications

If there's only one notification, you'll see a single line. If there are multiple notifications, you'll see chevron button. Select the chevron to view each message.



Feedback

Was this page helpful? Yes No

Provide product feedback

In-app notifications in model-driven apps

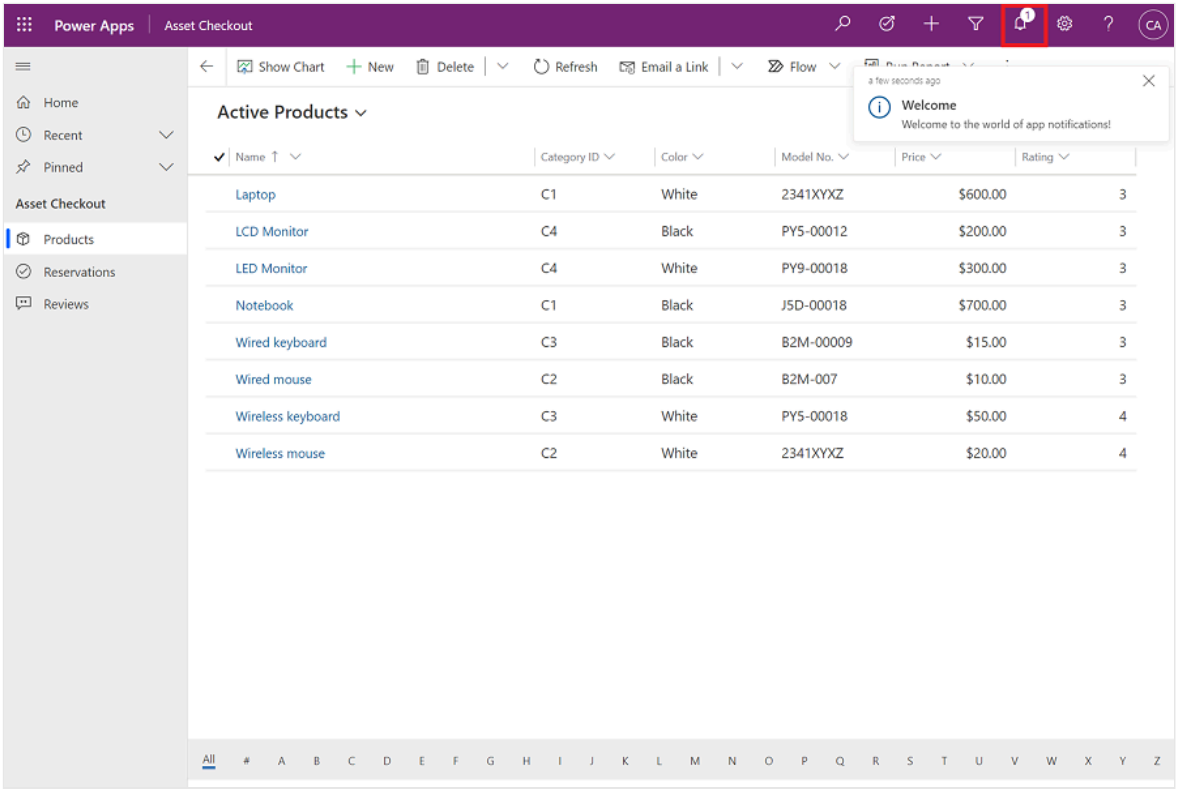
Article • 12/13/2022

In-app notifications allow you to receive notifications in the notification center of your model-driven app. Notifications are stored in the notification center until you dismiss them or when they expire. By default, notifications expire in 14 days but your system administrator can modify the expiration date.

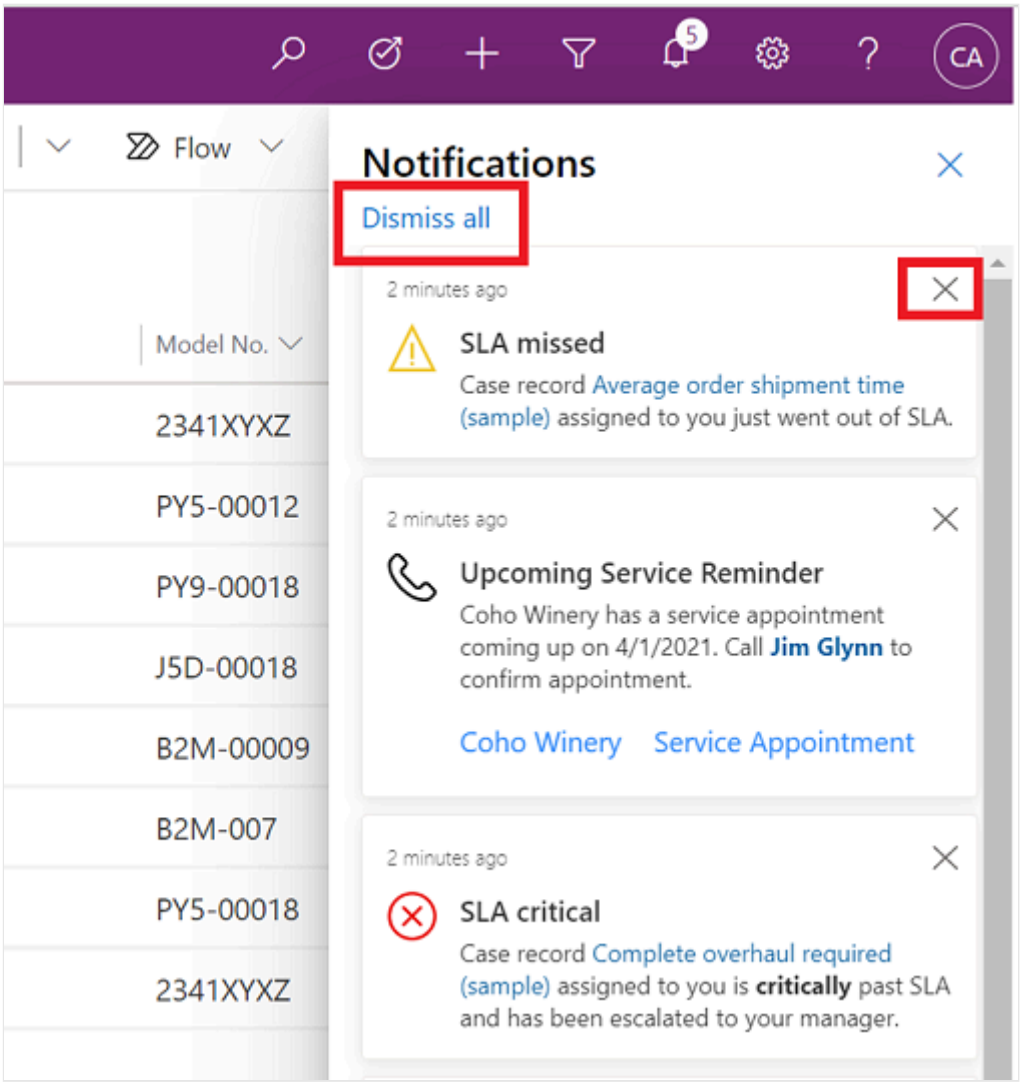
You only see notifications when you're using an app. That's when the system runs a check and displays any new notification. Notifications are at the organization level so you'll see notifications in the app you're using for all apps in your environment.

Notification center

- The notification bell icon shows the count of new notifications. When you open the notification center the count is cleared. To access your notifications, select the bell icon on the nav bar.

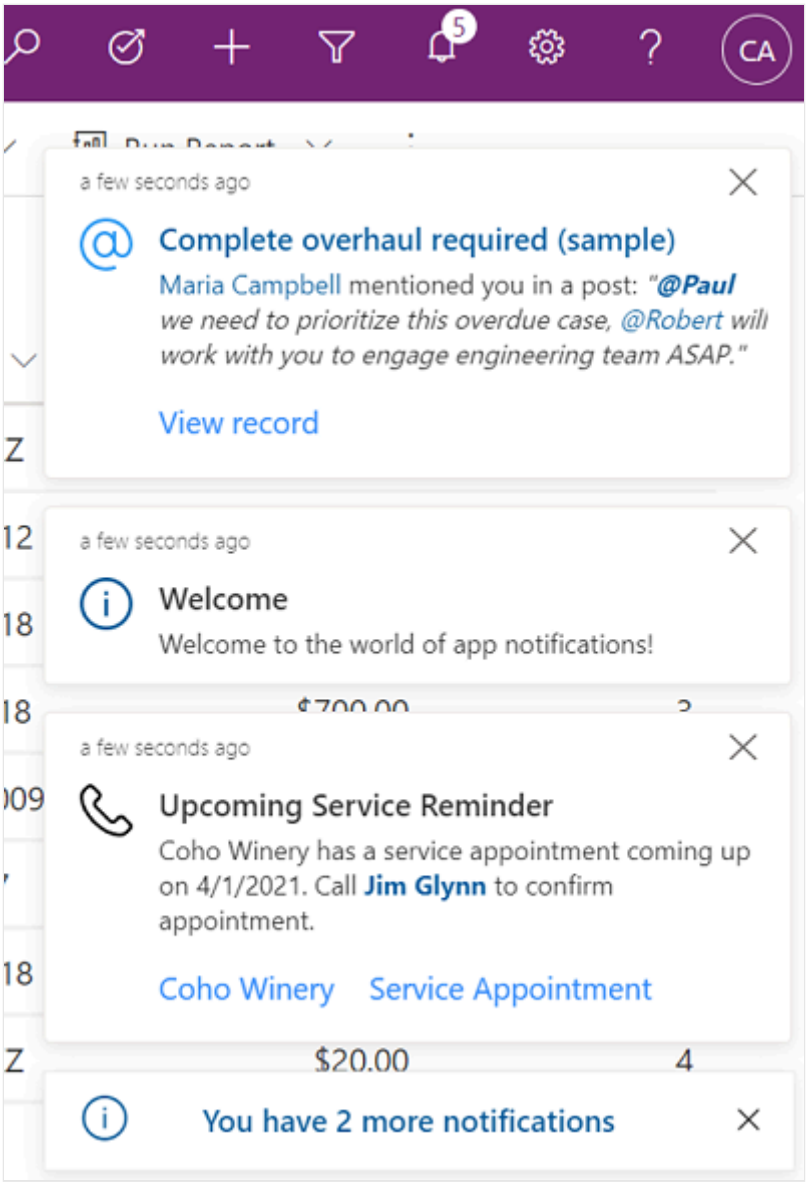


- To dismiss and delete a notification, select the close **X** button on a notification. Or, select **Dismiss all** to dismiss and delete all notifications.



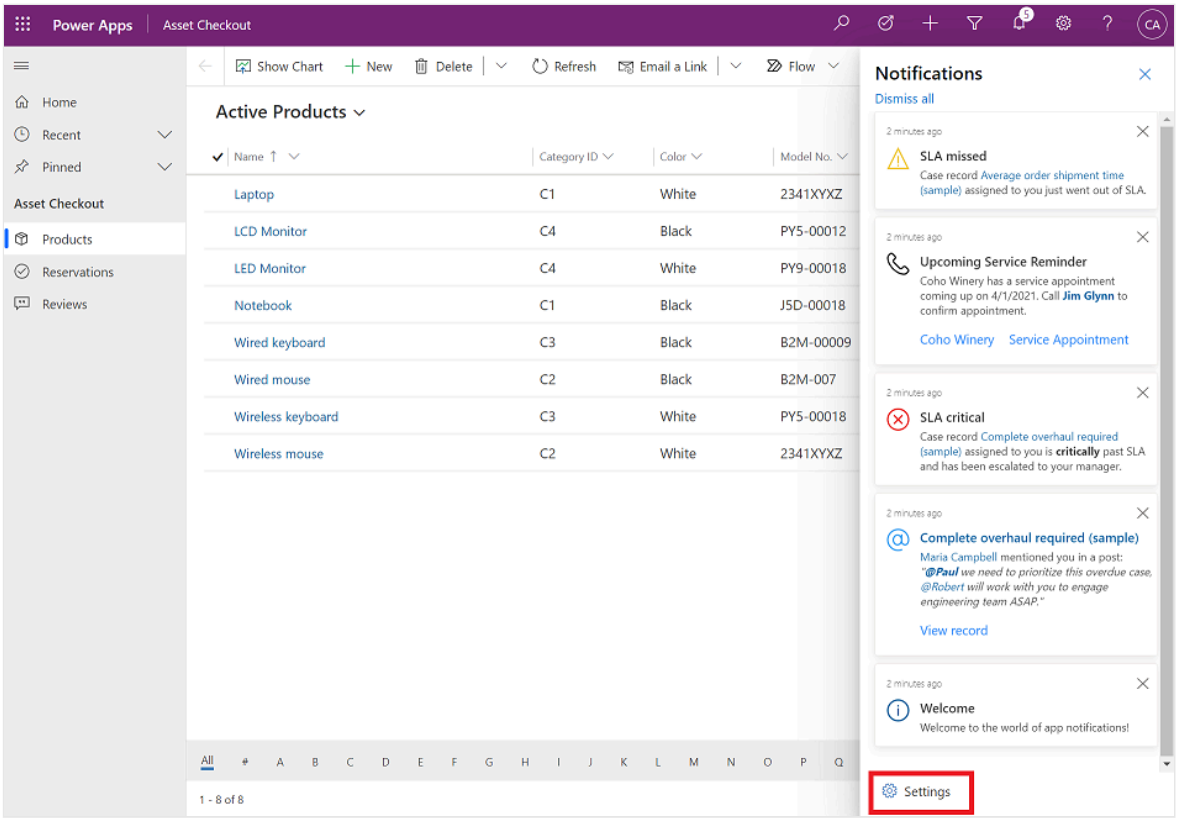
Toast notifications

Toast notifications appear temporarily to the right side of your app. When multiple notifications appear, they are stacked. When there's more than three toast notifications at the same time, you'll get a toast letting you know that there's more notifications. You can use the F2 shortcut key to access the notification toasts.



Your system administrator can enable or disable toast notifications. If toasts are enabled, you can turn them off at a user level. Regardless of the toast being shown, all notifications can be accessed from notification center.

1. To enable or disable toast notifications in the notification center, select **Settings**



2. To enable or disable toast notifications, do one of the following:

   - **To enable toast notifications**: Move the toggle to **On** and then enter how many seconds the toast will show.
   - **To disable toast notifications**: Move the toggle to **Off**.



3. When you're done, select **Save**.

See also

In-app notifications on Power Apps mobile
Send in-app notifications within model-driven apps

Feedback

Was this page helpful? Yes No

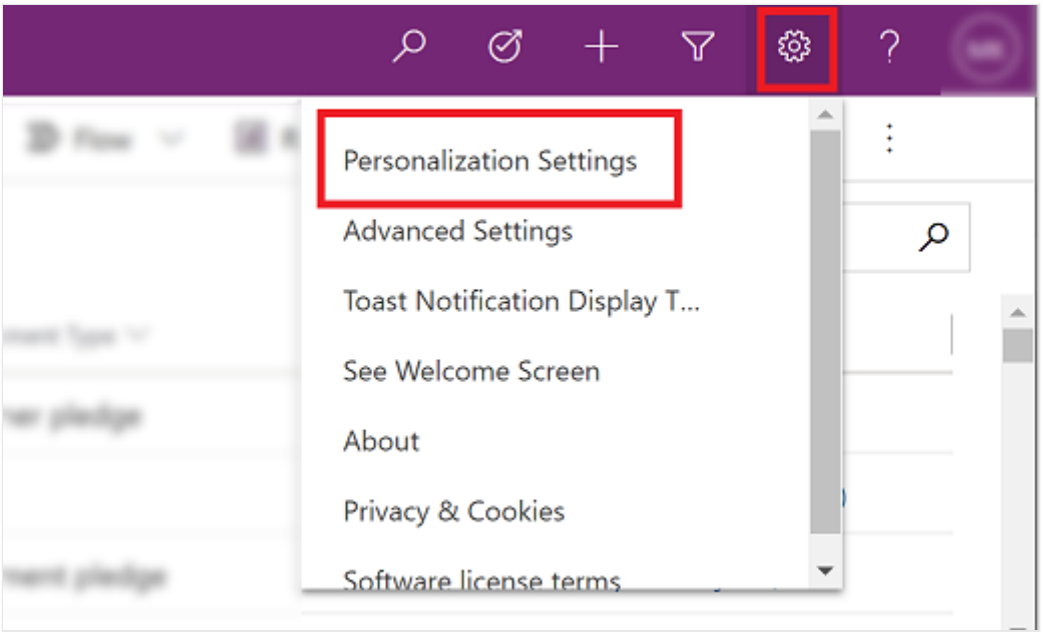
Provide product feedback

View your user information

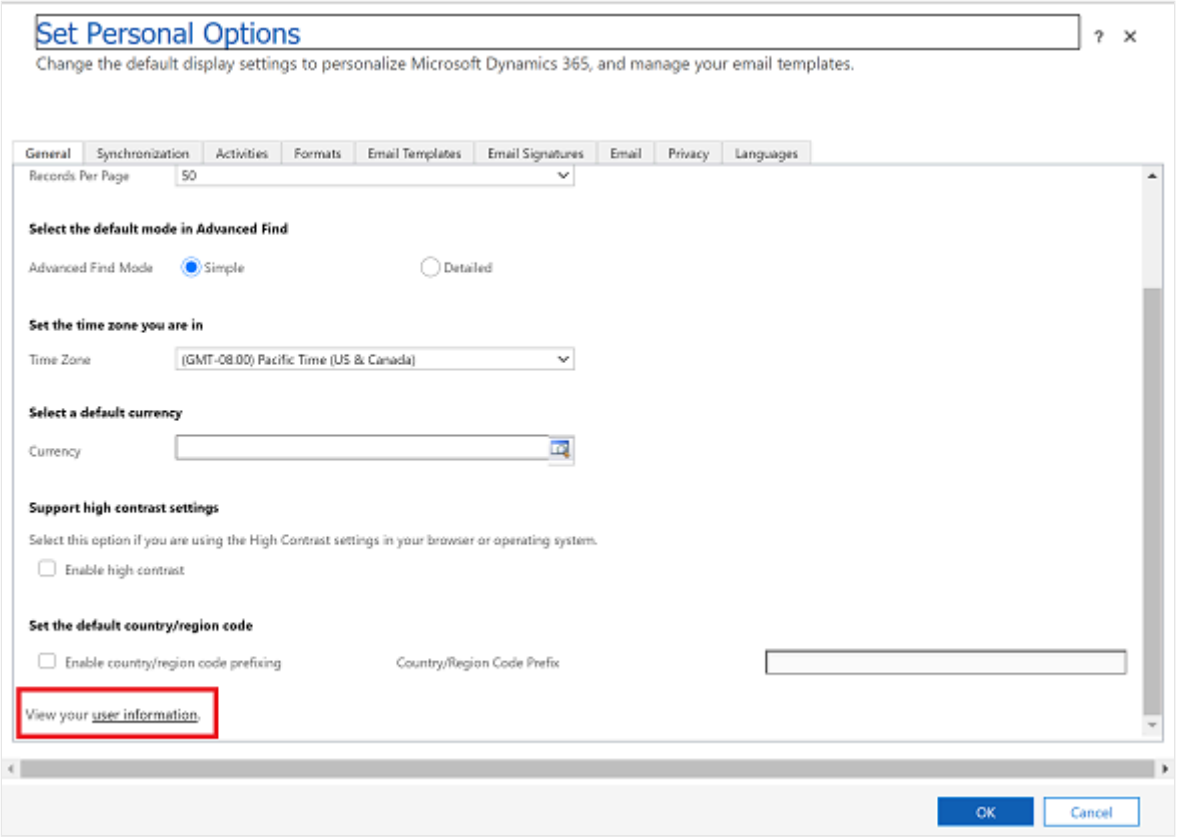
Article • 10/22/2024

The user information page displays useful information about you and your organization. On the page you can view your contact information, your security role, and information about your organization. Depending on your security role, you might be able to make changes to your user profile.

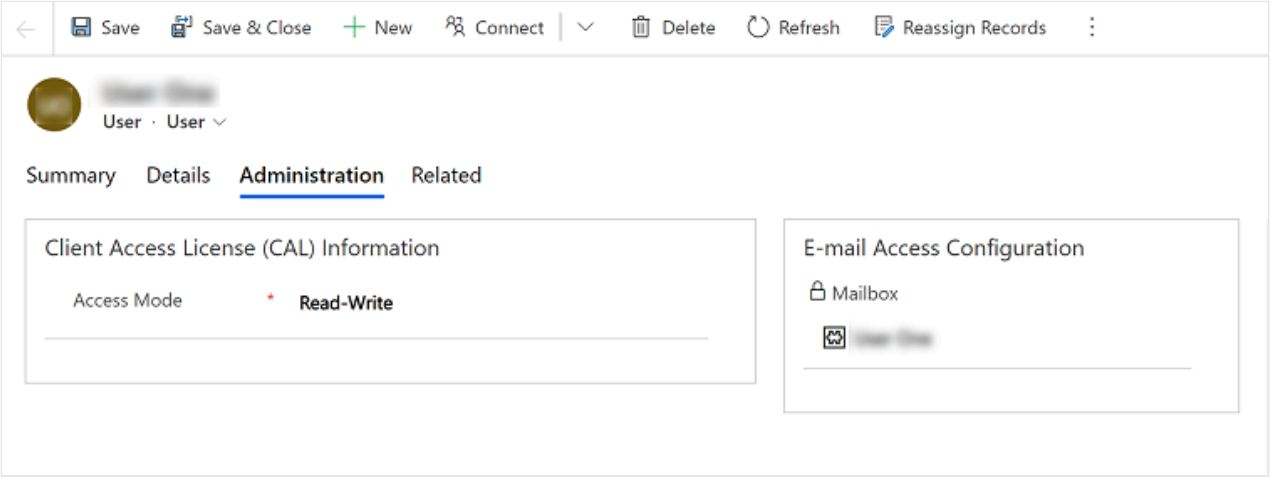
1. Open an app and in the upper-right corner of the screen, select **Settings** > **Personalization Settings**.



2. On the **General** tab, scroll down to the bottom and select the **user information** link.



3. On the user information page, select the different tabs, such as **Summary**, **Details**, or **Administration** to see details about your profile.



Set personal options

Article • 11/27/2023

Change your display settings in an app to suit your preferences. For example, you can choose the page that you want to see as soon as you sign in to the app. You can also personalize many other options such as language, currency, and time zone.

> **Note**
>
> You may need to refresh your app twice for some changed settings to take effect.

To set personal options

1. In the upper-right corner of the screen, select **Settings** > **Personalization Settings**.
2. Fill in the information, as required.
3. When you're done, select **OK**.

General tab options

Expand table

Options	Description
Select your home page and settings for Get Started panes	
Default Pane	Select the default home pane (page) that you want to see when you sign in. By default, the **Default based on user role** option is selected, which shows the default pane based on the app you are using. This option does not apply when you sign in using the Power Apps mobile app.
Default Tab	Select the default tab that you want to see for the selected default pane. For example, select the **Accounts** tab for the **Service** pane.
Set the number of rows shown per page in any list of rows	

Options	Description
Rows Per Page	Select the maximum number of rows you want to see in a list on a page. You can set a value from 25 to 250. A smaller number of rows per page is recommended, as it will help maximize grid performance. Selecting a larger number of rows per page can negatively impact grid performance.
Select the default mode in Advanced Find	
Advanced Find Mode	By default, every time you open the **Advanced Find Mode** dialog box, the query details are hidden. To see the query details every time, select **Detailed**.
Select the default search experience	
Default Search Experience	Select your default search experience. For more information, see Compare search options
Set the time zone you are in	
Time Zone	Select the time zone that you want to display for your region.
Select a default currency	
Currency	Choose the default currency value to use in financial transactions. Select the **Lookup** button to search for a currency.
Support high-contrast settings	
Enable high contrast	Select this check box to enable high-contrast levels in your app display. **Tip:** Set this option if you are using high contrast settings in your browser or operating system.
Set the default country/region code	
Enable country/region code prefixing	Select this check box to enable the default calling code for your region.
Country/Region Code Prefix	Enter the value of your region's calling code in the text box. For example, enter **+1** for the United States.
View your user information	Select this link to view details about you. These details are displayed to your entire organization and include your contact information and security role. For more information, View your user profile.

Synchronization tab options

Expand table

Options	Description
Synchronize Dynamics 365 items with Outlook or Exchange	
filters	Choose the rows to synchronize between Dynamics 365 and Dynamics 365 for Outlook or Exchange (using server-side synchronization).
synchronized columns	View the columns that are synchronized between Dynamics 365 and Dynamics 365 for Outlook so you can see where the data is coming from.
Manage your offline filters and take your information offline in Dynamics 365 for Outlook	
offline filters	Choose a subset of Dynamics 365 data that you want to work with when you go offline with Dynamics 365 for Outlook.

Activities tab options

Expand table

Options	Description
Default view	
Default Calendar	Select the default view for your calendar.
Start your default work hours	Select your default start and end time for your work hours.

Formats tab options

Expand table

Options	Description
Current Formats	

Options	Description
Customize	Choose this option to add a new region and formats.
Format preview	Shows the current region and its formats for **Number**, **Currency**, **Time**, and **Date**.

Email Templates tab options

Expand table

Options	Description
Create and modify personal email templates	
New	Create an email template with custom values (such as a date or signature), so you don't have to enter the same information, repeatedly, when you send an email. On the command bar, select **New** and enter values for the template.

Email Signatures tab options

Expand table

Options	Description
Create and modify personal email signatures	
New	Create an email signature to save time and be consistent in your responses. The owner of an email signature can be a user or a team. On the command bar, select **New** and enter values for the signature.

Email tab options

Expand table

Options	Description
Select if other users can send emails for you	
Allow other users to send email on your behalf.	Select this option to allow other users of Dynamics 365 to send email on your behalf. **Caution**: If you select this option, other users can send email on your behalf with or without your consent. Your email name will appear as the sender.
Select the email messages to track in Dynamics 365	
Track	Select the email messages that you want to automatically track in Dynamics 365. **Options:** - All Email messages - Email messages in response to Dynamics 365 email - Email messages from Dynamics 365 Leads, Contacts and Accounts - Email messages from Dynamics 365 rows that are email enabled
Configure Folder Tracking Rules	Set up folders to automatically track incoming email.
Automatically create rows in Dynamics 365	
Create	Select this option to allow Dynamics 365 to create leads or contacts automatically from the information in tracked email messages.
Show emails as conversation on Timeline	Select this to list emails in a conversation thread, the way many email applications work when viewing emails in a conversation.
View your Mailbox	Select this link to see your mailbox details.

Privacy tab options

> **Note**
>
> This tab isn't available if your system administrator has selected the privacy preference for the entire organization in System Settings. Talk to your administrator.

Expand table

Options	Description
Select your error notification preference	
Options: - Ask me for permission to send an error report to Microsoft. - Automatically send report to Microsoft without asking me for permission. - Never send an error report to Microsoft about Dynamics 365.	Specify what you want Dynamics 365 to do when an error occurs. Based on your selection, the error reports are shared with and used by Microsoft for product improvements. We recommend that you send the error reports to Microsoft so that Microsoft can use the information for product improvements.

Languages tab options

Expand table

Options	Description
Select the language you prefer to see Dynamics 365 displayed in	
Base Language	Shows the base language. The base language is set during the Dynamics 365 installation process. You can't edit this option.
User Interface Language	Select the language that you want to see for labels and dialog boxes in the Dynamics 365 user interface.
Help Language	Select the language for Help. To add an additional language other than the base language, your admin must install the required language packs and enable them. Talk to your administrator.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use a screen reader

Article • 02/15/2022

Screen readers make model-driven apps accessible to people who have low or no vision or might need extra support for a temporary scenario, such as eye fatigue. Commonly used screen readers such as Narrator, JAWS, and NVDA are supported.

- Learn more about working with Microsoft Narrator
- Learn more about working with JAWS
- Learn more about working with NVDA

Basic tasks using a screen reader

Open an app

1. On the navigation bar, use the **Tab** key to move to the app drop-down control and press **Enter** to open the site map.
2. Press the **Tab** key until you hear the name of the application you want to open—for example, "Sales." Press **Enter** to open the app.

Use scan mode in Narrator

You can use scan mode to quickly navigate apps using the arrow keys and common keyboard shortcuts. Quickly jump to headings, links, landmarks, form columns, controls, and tables in this mode. Turn scan mode on and off by pressing **Caps lock+Spacebar**. More information: Using scan mode

Find your way around the app

Navigation bar

When you open an app, a vertical bar with subarea icons is displayed on the left. You can either use the **Tab** key to move through these icons until you hear the name of the subarea you want, such as "Accounts," or you can use the site map control. For example, press the **Tab** key until you hear "Accounts" and then press **Enter** to open the Accounts view.

Grids

Screen readers navigate grids more reliably and consistently, and announce row and column headings, as well as the position within the grid. When you first open a grid, the default tab stop is the view selector.

Whenever you enter a cell within the grid from outside of the grid, Narrator announces the name of the table, the row and column counts, and the position of your cursor within the table.

If your cursor is within the table header, quickly navigate between headers by using the **Tab** key or **Shift+Tab**. Narrator will announce the name of each header as you enter the header cell. It also announces the type of cell (for example, "column header"), the location of the column (for example, "column 1 of 6"), and whether the column is sorted or selected. If you press **Enter** in a table header, it will sort the table by that column. Narrator announces the sort order, and you can press **Enter** again to change the order.

When you move out of the last column in the table, the cursor moves to the second row of the grid, and from this point, you must use the Up and Down arrow keys to navigate between the non-header cells. If you instead press the **Tab** key again, your cursor will move to the next interactive element, typically the table filter list. When moving between non-header row cells, Narrator announces the column name, the location of the column, and text within the cell.

Forms

Multiple navigation modes are available for navigating a form using Narrator, with the most common modes being landmarks, headings, and form columns. To change the navigation mode, press **Caps lock+Up arrow**. Hold the Caps lock key down while pressing the Up arrow key to cycle through the modes until you hear the mode you want to use. Then use Caps lock + the Left/Right arrow keys to navigate through the various items. For example, if you want to go to the Last Name column in the Contact Information section of a contact, do the following:

1. Press and hold the **Caps lock** key and press the **Up arrow** key until you hear "Landmarks."
2. Press and hold the **Caps lock** key and press the **Right arrow** key until you hear "Contact information."
3. Change the mode by pressing and holding the **Caps lock** key and pressing the **Up arrow** key until you hear "Form Columns."
4. Navigate to the Last Name column by using Caps lock + the Left/Right arrow keys until you hear "Last Name." Narrator also announces the control type, value, state,

and any special instructions for the column.

You can also use the Tab key to quickly navigate to interactive elements on the form. Some form columns have an icon that will perform the default action when you press Ctrl+Enter. For example, an email form column might have an envelope icon that opens an email editor.

Dashboards/charts

You can navigate through the dashboard charts using the Tab key and Caps lock + arrow keys. Press the **Tab** key to quickly go to the interactive elements and use Caps lock + an arrow key for navigation of non-interactive elements, such as headings, landmarks, and items.

> **Note**
>
> You must have the latest Windows 10 Update installed to have all of the accessibility features available for charts.

Interactive dashboard streams

You can use the **Tab** key or **Shift+Tab** keys to move between interactive dashboard streams, such as found in the Accounts dashboard, or just change the navigation mode until you hear “Headings” and then use the **Tab** key to quickly move between dashboard streams.

To navigate through each element of a dashboard stream, use the Up/Down arrow keys. Narrator will read the type of control and the title of the control.

Business process flows

You can navigate a business process flow, such as the one found at the top of the Lead form, by pressing the **Tab** key to move forward, and **Shift+Tab** to move backward between the tables. Use the **Enter** key on the **Move to the Left** or **Move to the Right** buttons to display additional tables in the process flow. Narrator reads the table type, stage, status, title, element number out of total elements, and whether it is currently selected.

Dialog boxes

When a dialog box opens, Narrator announces the title. For dialog boxes with input columns, the **Close** button has the default focus, allowing you to close the dialog box by pressing **Enter**. For dialog boxes that require user action, the focus is on the primary action button, such as **Delete** or **OK**.

You can navigate through the controls by using the **Tab** key. The cursor will loop through each element in the dialog box, and you can press **Esc** to close it.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use keyboard shortcuts in Power Apps

Article • 04/12/2023

Keyboard shortcuts give you an alternate way to do common tasks instead of using your mouse or tapping with your finger. They help in seamless navigation of the interface. The following keyboard shortcuts apply across apps.

> **Note**
>
> The keyboard shortcuts described here refer to the United States keyboard layout. Keys on other keyboard layouts might not correspond exactly to the keys on a US keyboard.

Global shortcuts

Expand table

Type	Shortcut key	Description
General	F2	Activates the flyouts including toast notifcations.
Landmark	Ctrl+F6	Navigates you through the landmarks of a page in clockwise order. Remembers the last selected item in each landmark and will navigate to that last item.
Landmark	Ctrl+F6+Shift	Navigates you through the landmarks of a page in counter clockwise order. Remembers the last selected item in each landmark and will navigate to that last item.
General	Ctrl+[or Ctrl+Shift+[	Set the focus on first focusable element of the main page

Form shortcuts

Expand table

Task	Unified Interface
Complete the command for the active option or button	Enter
Cancel a command, or close a selected list or dialog box	Esc

Task	Unified Interface
Save	Ctrl+S
Save and Close	N/A
Cancel edits and close (Close)	Esc
Delete the row (when forms are in edit mode)	N/A
Save and then open a new form (Save and New) (when forms are in edit mode)	N/A
Open the lookup menu with the most recently used items in alphabetical order	Down arrow
Open a lookup drop-down list	Enter
Close a lookup drop-down list	Esc
Open a row found in lookup with forms in edit mode	Enter
View more information when focus is on a lookup row item with a chevron	Right arrow (→) key
Close the more information when focus is on a lookup item with a chevron	Left (←) arrow key
Add an article to an email	N/A
Apply an email template (when editing an email message)	N/A

Form navigation shortcuts

Expand table

Task	Unified Interface
Move to the next option, option group, or column	Tab
Move to the previous option, option group, or column	Shift+Tab
Move between options in an open list, or between options in a group of options	Arrow keys
Go to the Command Bar	N/A
Go to the landmark section of a page	Ctrl+HOME

Email

The following table outlines a list of accessibility shortcuts available when using email. You can access this list while composing email by pressing **Alt+0**.

Expand table

Type	Shortcut key	Description
General	Alt+F11	Toggle full-screen view.
Tab navigation	Alt+Ctrl+0	Go to the editor toolbar.
Tab navigation	Alt+1	Go to the rich-text editor.
Tab navigation	Alt+2	Go to the HTML editor.
Tab navigation	Alt+3	Go to the preview view.
General editor commands	Alt+F10	Navigate to the editor toolbar. Move to the next and previous toolbar group with Tab and Shift+Tab. Move to the next and previous toolbar button with Right Arrow or Left Arrow. Press Space or Enter to activate the toolbar button.
General editor commands	Alt+-	Expand/collapse the toolbar.
General editor commands	See description.	Editor dialog: Inside a dialog, press Tab to navigate to the next dialog element, press Shift+Tab to move to the previous dialog element, press Enter to submit the dialog, press ESC to cancel the dialog. When a dialog has multiple tabs, the tab list can be reached either with Alt+F10 or with Tab, following the dialog tabbing order. With a tab list focused, move to the next and previous tab with Right and Left Arrow, respectively.
General editor commands	See description.	Editor list box: Inside a list box, move to next list item with Tab or Down Arrow. Move to previous list item with Shift+Tab or Up Arrow. Press Space or Enter to select the list option. Press ESC to close the list box.
Base commands	Ctrl+Z	Undo command.

Type	Shortcut key	Description
Base commands	Shift+Ctrl+Z	Redo command.
Base commands	Ctrl+B	Bold command.
Base commands	Ctrl+I	Italic command.
Base commands	Ctrl+U	Underline command.
Base commands	Alt+0	Accessibility help.
Base commands	Esc	Cancel operation.

Editable grids (views)

If your administrator has enabled editable grids (views), you can edit data directly in a grid (view). The following table lists the keyboard shortcuts:

Expand table

Task	Unified Interface
When the focus is on a column header, sort by column	Spacebar to drop down list and then down arrow to select an action, followed by Enter
When the focus is on a column header, open filter dialog	Spacebar to drop down list and then down arrow to select the **Filter by** option, followed by Enter
When the focus is on the cells, move to the next column	Right arrow (→) key
When the focus is on the cells, move to the previous column	Left arrow (←) key
When the focus is on the column header, move to the next column header	Tab
When the focus is on the column header, move to the previous column header	Shift+Tab

Task	Unified Interface
Move to the cell above	Up arrow (↑) key
Move to the cell below	Down arrow (↓) key **Or** Enter, when cell is not in edit mode
Go to edit mode for Text, Number, and Simple columns when the focus is on a column	Type the value directly to overwrite the existing value **Or** Spacebar to continue modifying the existing value **Or** F2 to select the existing value
Go to edit mode for Date and Time columns	Type the value directly **Or** F4 or Alt+↓ to display the date/time picker
Go to edit mode for drop-down (Lookup, Option Set) columns	Spacebar to open the list
Go to edit mode for Two Options columns	Spacebar to switch between the two options **Or** F4 **Or** Alt+↓ to open the list
Move between entries in an open drop-down list	Up/Down (↑/↓) Arrow keys
Select an option in an open drop-down list	Enter
Close an open drop-down list	Esc
Cancel your edits	Esc
Navigate to and open the lookup row	Enter when the focus is on (→) icon
Move to the next page (if any)	Page down key
Move to the previous page (if any)	Page up key
Move to the column header when the focus is in the grid	Shift+Tab
Move to the Save button when the focus is on the column header (if any unsaved data)	Tab
Move to the Refresh button when the focus is on the column header	Tab

Interactive dashboard shortcuts

Interactive dashboards enable new functionality, such as filtering, sorting, and quick actions.

Expand table

Task	Unified Interface
Activate a dashboard element (simulate left-click)	Enter **Or** Spacebar
Move to the next element	Tab
Move to the previous element	Shift+Tab
Move to the next item within a dashboard element	Down (↓) arrow key
Move to the previous item within a dashboard element	Up (↑) arrow key
Close a menu	Escape
Select a check box or radio button	Spacebar
Modify the date range filter	Ctrl+Alt+D
Show the visual filter pane	Ctrl+Alt+V
Move to the global command bar	Ctrl+Alt+A

Global filter shortcuts

Expand table

Task	Unified Interface
Open a menu	Enter **Or** Spacebar **Or** Down (↓) arrow key
Activate a button	Enter **Or** Spacebar
Move to the next element	Tab
Move to the previous element	Shift+Tab
Open a tree view control	Right (→) arrow key
Close a tree view control	Left (←) arrow key
Move to the next tree view node	Up (↑) arrow key
Move to the previous tree view node	Down (↓) arrow key
Move to the first tree view node	Home
Move to the last tree view node	End
Perform the default action for the tree view node	Enter

Task	Unified Interface
Remove a filter	Delete

Dashboard stream control shortcuts

Expand table

Task	Unified Interface
Move to the first command on the active dashboard stream's command bar	Ctrl+Alt+Q
Move to the next element	Tab
Move to the previous element	Shift+Tab
Activate a button	Enter **Or** Spacebar
Activate the **Sort by column** button and open flyout	Enter **Or** Spacebar **Or** Down (↓) arrow key
Move to the next item	Down (↓) arrow key
Move to the previous item	Up (↑) arrow key
Move to the first item	Home
Move to the last item	End
Move to the next stream	Ctrl+F6
Move to the previous stream	Ctrl+Shift+F6
Close a menu	Escape
Change the state of a check box	Spacebar

Common controls

Chart shortcuts

Expand table

Task	Unified Interface
Move to the first element in a chart	Tab

Task	Unified Interface
Move to the next data point	Right (→) arrow key **OR** Down (↓) arrow key
Move to the previous data point	Left (←) arrow key **OR** Up (↑) arrow key
Move to the **See More** button in a Tag chart	Tab
Move back from the **See More** button to the tags in a Tag chart	Shift+Tab
Filter the dashboard when the focus is on a data point in an interactive dashboard	Enter **Or** Spacebar

Date-Time control shortcuts

Expand table

Task	Unified Interface
Open the date picker flyout	Enter **Or** Spacebar **Or** Down (↓) arrow key
Activate the option and close the flyout	Enter **Or** Spacebar
Move to the next item	Down (↓) arrow key
Move to the previous item	Up (↑) arrow key
Move to the first item	Home
Move to the last item	End
Close the date picker flyout	Escape
Move to the next element	Tab
Move to the previous element	Shift+Tab
Select the calendar	Enter **Or** Spacebar
Activate a button	Enter **Or** Spacebar

Search results shortcuts

Expand table

Task	Unified Interface
Move to the next search result	Up (↑) arrow key
Move to the previous search result	Down (↓) arrow key
Move to the list of available views	Right (→) arrow key
Move from the list of available views to search results	Left (←) arrow key
Navigate between buttons	Tab

Navigating submenus

If you use a screen reader, follow these steps to access any of the submenus that are displayed when you hover the mouse over a navigation link.

1. Navigate between the menu options using standard link navigation.
2. Activate the link by selecting Enter or the space key as you usually would. A list of submenus will be displayed.
3. To access these submenu options, use the assistive technology (AT) shortcut key to navigate to the next button. For example, if you use the NVDA screen reader, you would use the “b” shortcut key (NVDA+b). To navigate through all the options, select the key repeatedly.

> **ⓘ Important**
>
> Several areas in the user interface (UI) have auto-collapse built in for menus. This includes the Navigation Bar and Form Navigation. Both expand when selected, but after 30 seconds of inactivity, the items collapse.

Navigating through stages of a process

If you’re in a row—for example, a lead—this section shows you the keyboard shortcuts to navigate through the different stages in the sales process to qualify the lead, and how to make selections in those stages. For example, if you want to move easily from the Qualify stage through to the Close stage, try these keyboard actions:

- After you select a stage, get to the columns by selecting the Down (↓) arrow.
- If a stage is collapsed, select Enter to expand it.

> **Note**
>
> The process name is an icon at the bottom of the page, not the top.

- **Keyboard shortcuts in Windows**: If you are using the Windows operating system, select the following link and choose the version number from the drop-down list. You can view all the shortcuts that are applicable to a Windows environment. See Windows keyboard shortcuts.
- **Keyboard shortcuts in Mac**: If you are using a Mac operating system, select the following link to view all the shortcuts that are applicable to a Mac environment. See Mac keyboard shortcuts.

Feedback

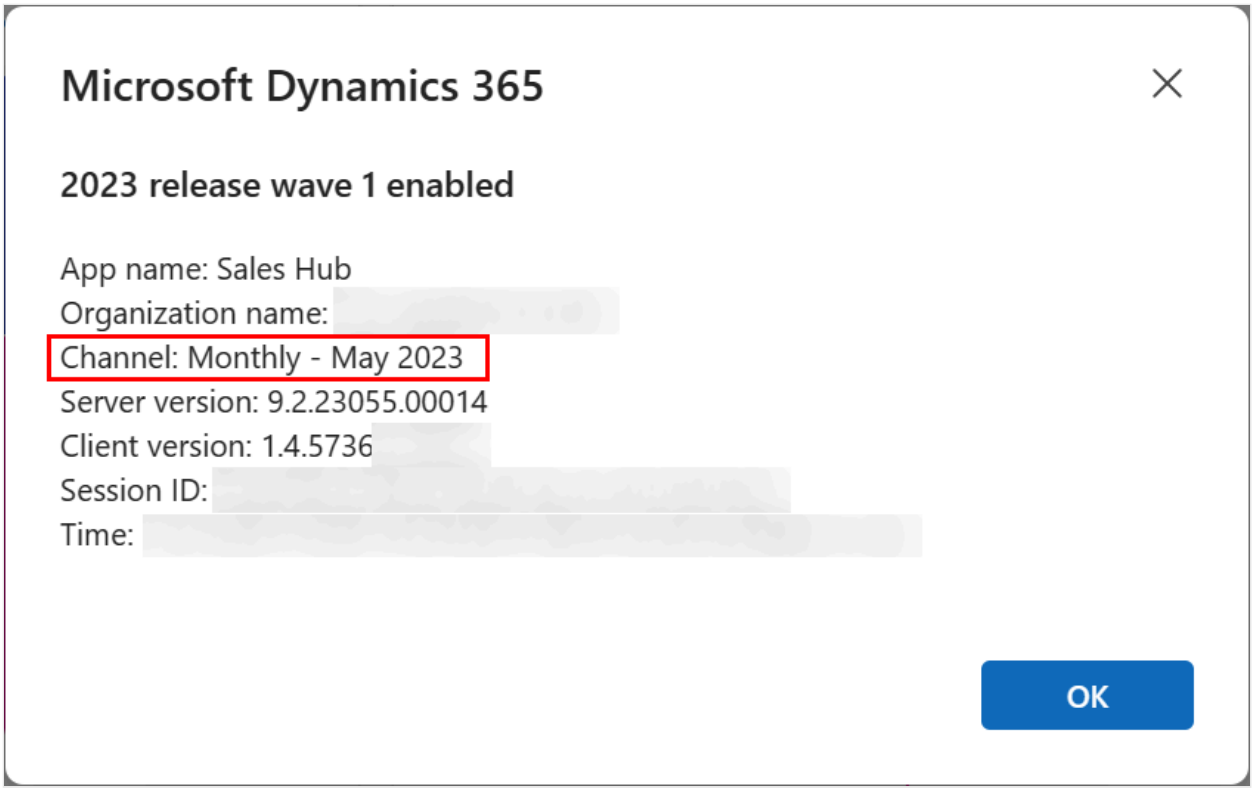
Was this page helpful? Yes No

Provide product feedback

About dialog for model-driven apps

Article • 11/07/2023

This article explains the various parts of the **About** dialog for model-driven apps.



- The **Channel** attribute indicates the feature set being used. This attribute can be set by environment, app, user, or URL parameter so that a suffix indicates the release name of the current session. More information: Changing release channels
- The **semi-annual channel** is the current, twice-a-year release wave that is updated in April and October. The release for the semi-annual channel is shown at the top of the dialog like **2023 release wave 1 enabled**.
- The **monthly channel** is the new, release cadence that happens once per month. The release for the monthly channel is shown after **Channel: Monthly** as a month and year. For example, in the screenshot provided, the monthly channel is May 2023.

See also

- Release Channels for your model driven app
- Changing release channels for model-driven apps

Feedback

Was this page helpful? Yes No

Provide product feedback

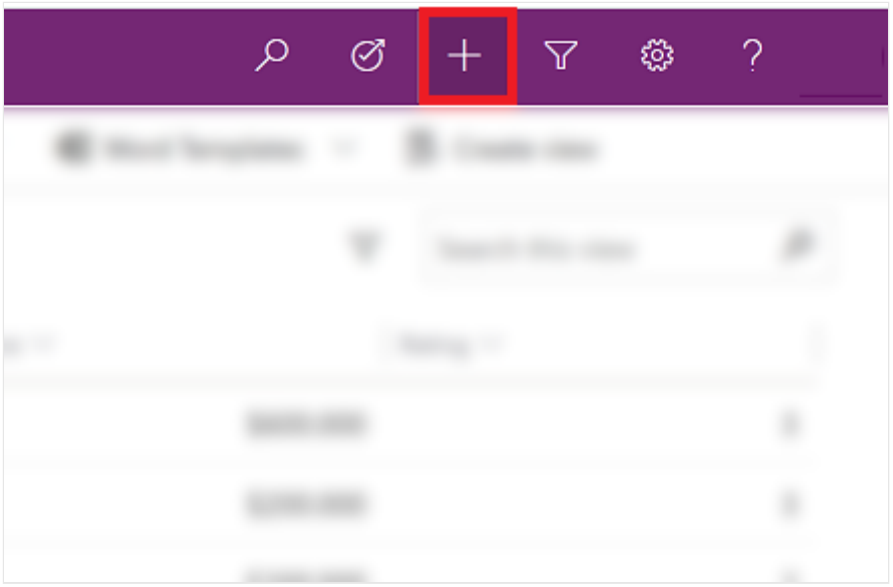
Create a new row

Article • 01/31/2024

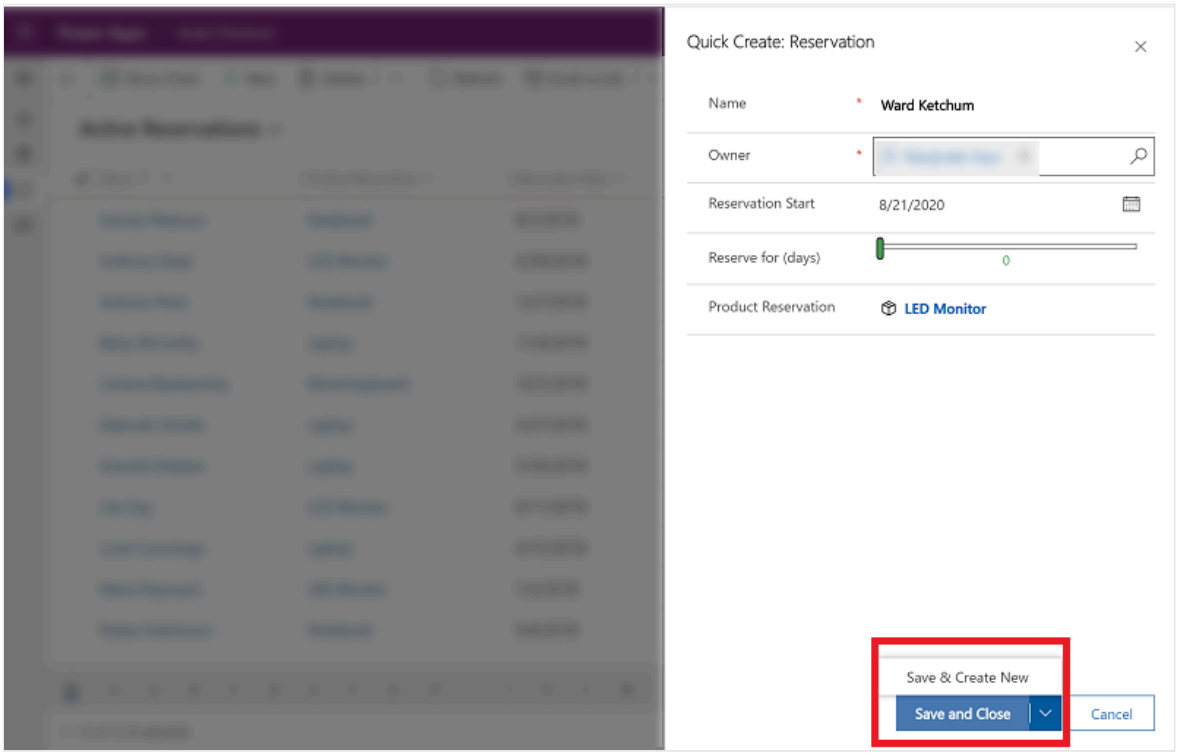
The **New** command or **Quick create** makes it fast and easy to enter almost any type of information into the system. The command is on the navigation bar, so it's available whenever you need to enter new information into the system. You can also save a row and create a new one directly from the **Quick create** form.

> **Note**
>
> The **Quick create** option is only available for rows that are enabled by your admin.

1. On the navigation bar, select the **plus sign** , and then select the item you want.



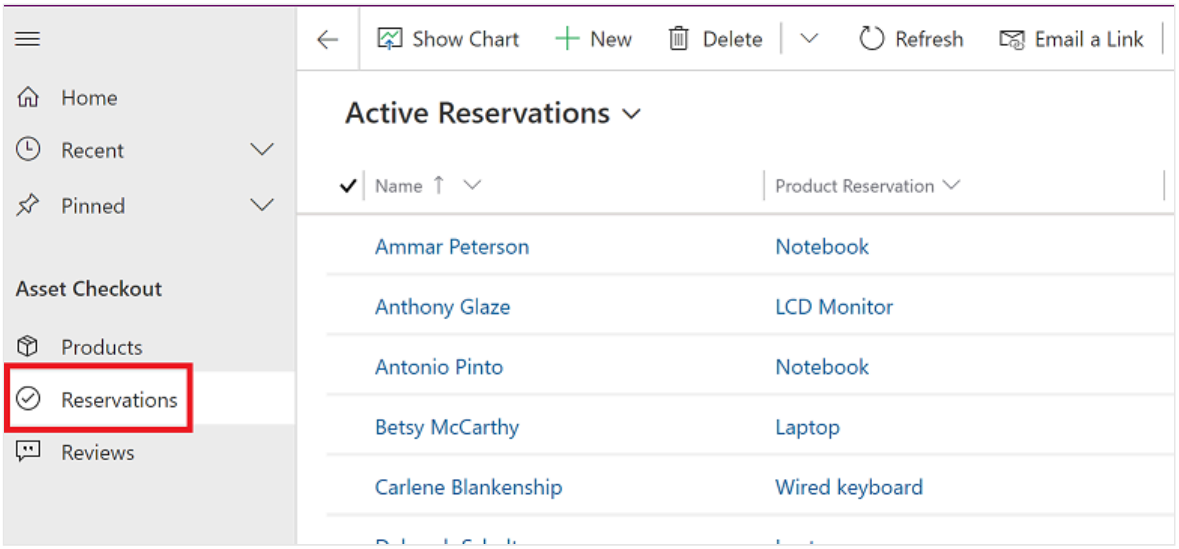
2. Fill in the columns, and then select **Save and Close**. Or, to save and create another row, select the down arrow and then select **Save & Create New**.



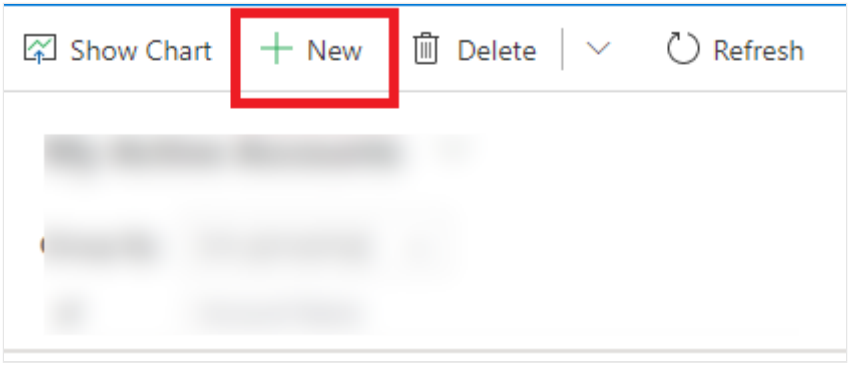
> **Note**
>
> An asterisk * next to the column on the screen means the column is required. If you select **Save and Close** before entering required columns an error message will be displayed or if you have entered information and select **Cancel** a warning will be displayed.
>
> A plus sign + next to the column on the screen means your organization recommends that you fill in the column.
>
> A lock sign next to the column on the screen means this field is read-only and cannot be edited.

Create a new row using the New button

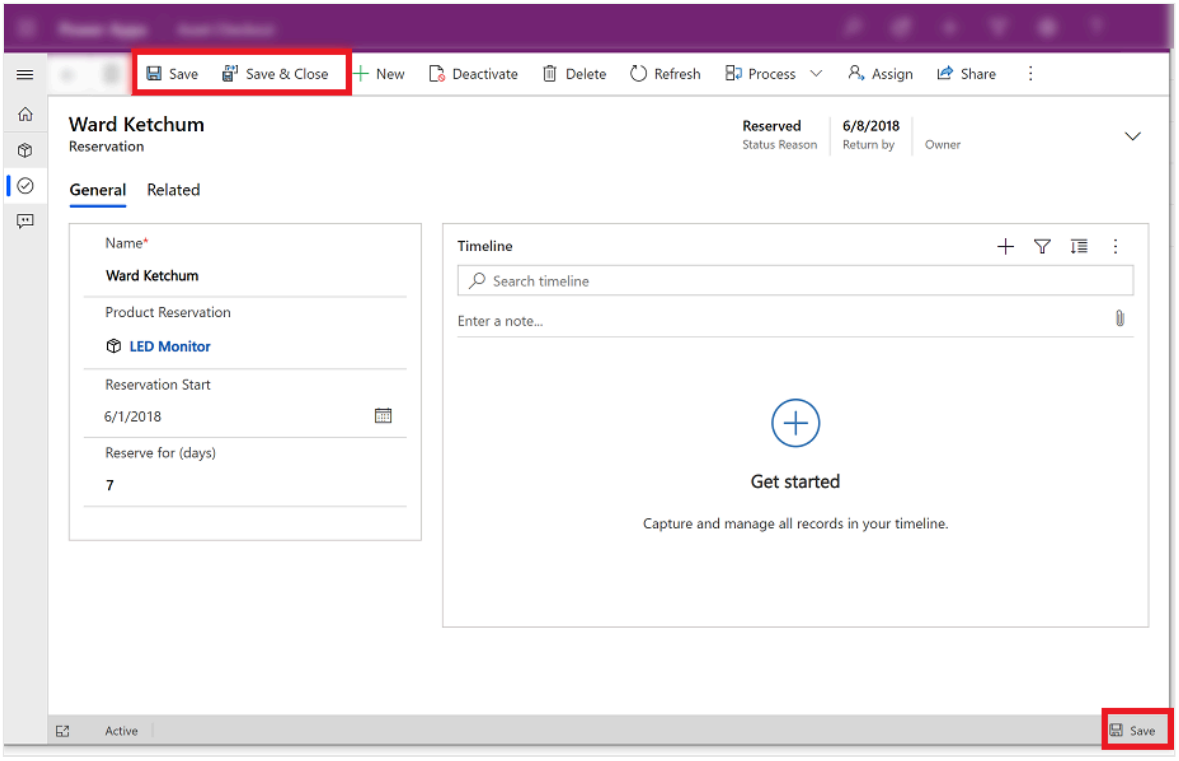
1. From the left navigation pane, select a table type.



2. On the command bar, select **+New**.



3. Fill in the columns and then on the command bar, select **Save** or **Save & Close**. The **Save** option is still available on the bottom right corner.



> **Note**

If you have unsaved changes and try to go to another row, a **Unsaved changes** dialog box will pop-up. If you select **Save and continue**, it will attempt to save your information and open the page you wanted to go to. If you choose to save and continue and there is an error on a column, the dialog will close and you will remain on the page to fix the error before you can navigate away.

Feedback

Was this page helpful? Yes No

Provide product feedback

Edit multiple rows (Bulk edit)

Article • 02/15/2022

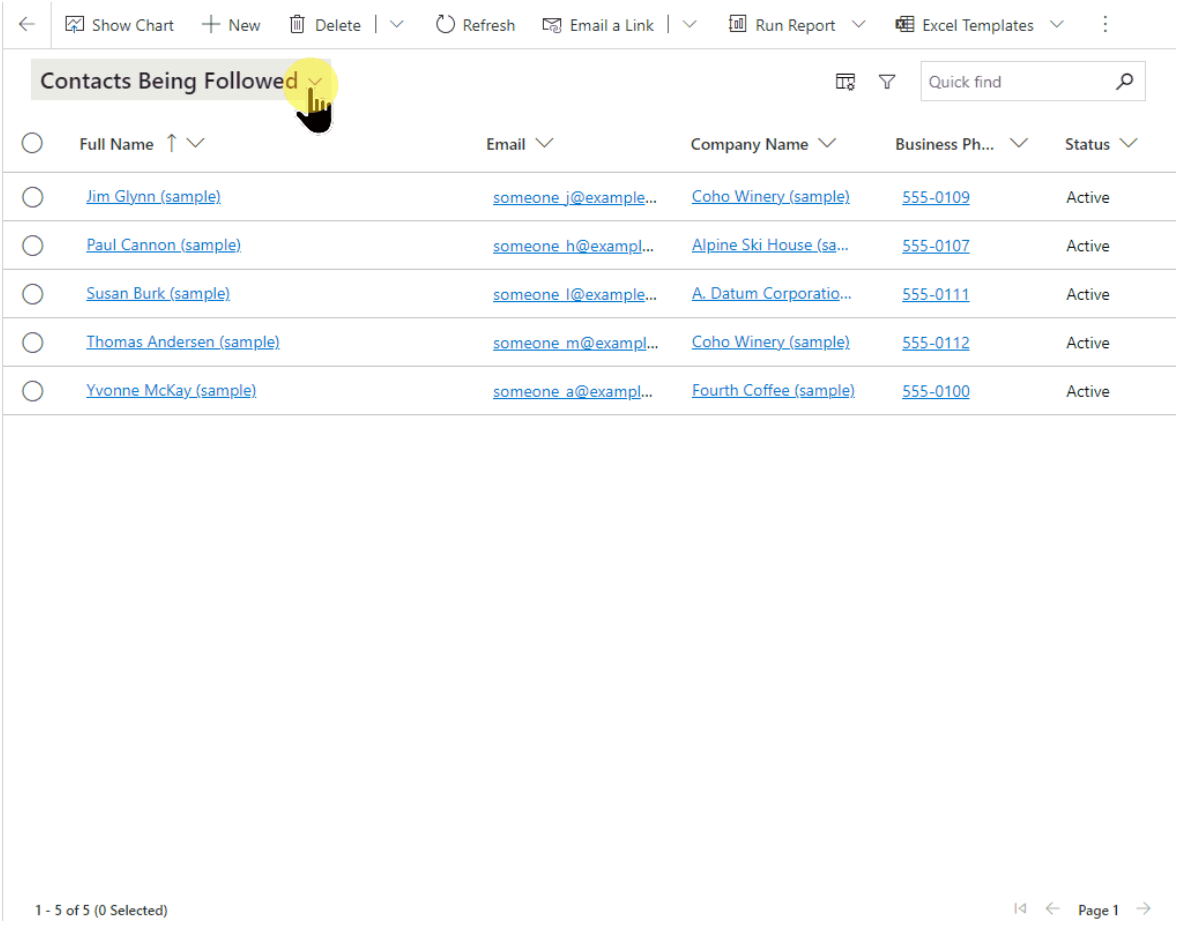
[This topic is pre-release documentation and is subject to change.]

You can edit multiple rows in model-driven apps and quickly update several columns of data across multiple rows in a single action. This is useful in scenarios when you need to update the same column for multiple rows.

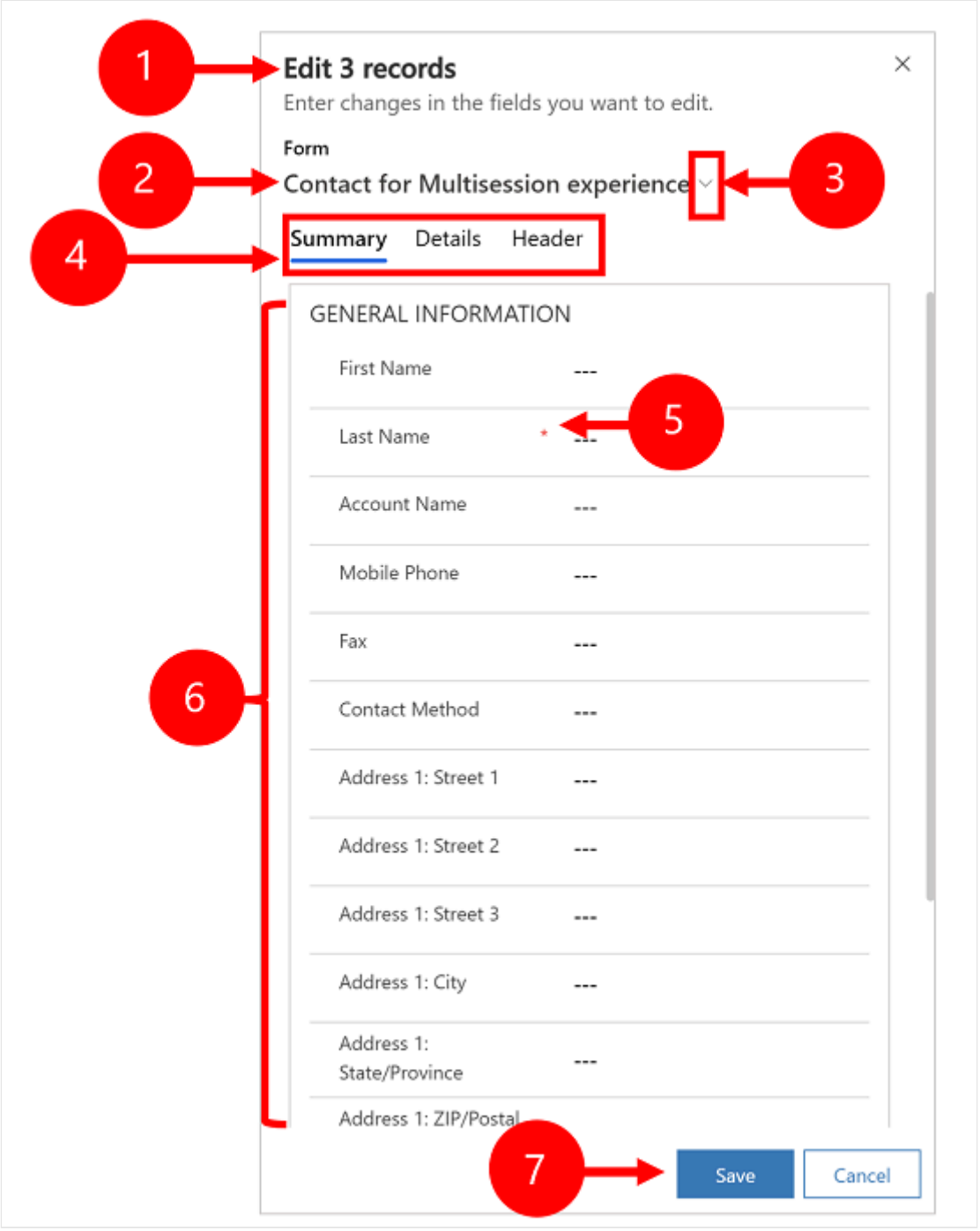
> **Note**
>
> Your administrator must grant you the Bulk Edit miscellaneous privilege to edit multiple rows.

Columns such as timeline wall, quick view forms, reference panel, and etc. aren't available when you're editing multiple rows.

1. To edit multiple rows, go to view page with the list of rows and select two or more rows.
2. On the command bar, select **Edit**.



The **Edit (number of rows) records** dialog opens with your last used form for the table.



Legend

1. Shows the number of rows that you're editing.
2. Shows the form title.
3. Select the chevron icon to switch from the default form to another form.
4. Select a tab to edit the columns on the form. The header column is always listed on the last tab.
5. If you changed the data for a required column, it needs to contain data otherwise you won't be able to save your changes.
6. Shows the form details that can be edited.
7. Save or cancel your changes. Saved changes are saved for all selected rows.

> **Note**
>
> You can't edit multiple rows with Power Apps mobile.

Feedback

Was this page helpful? Yes No

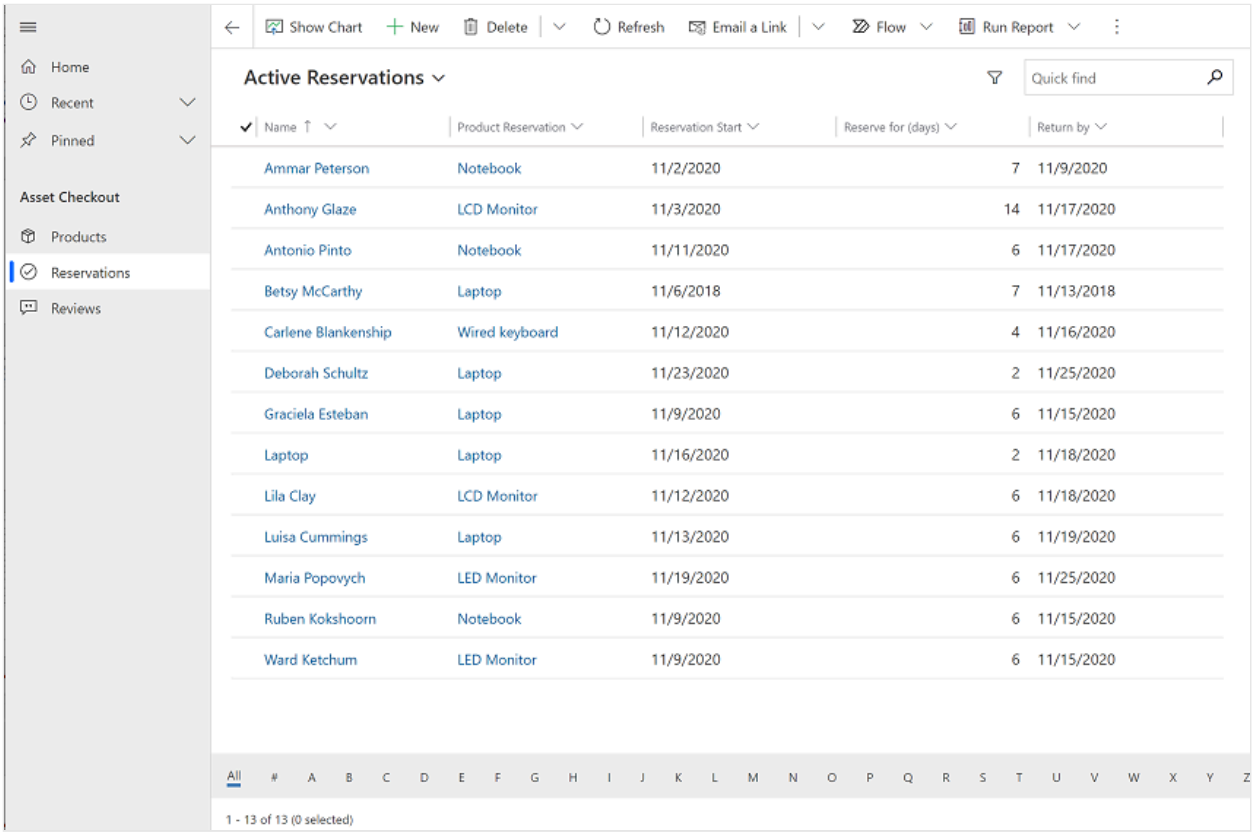
Provide product feedback

Work with rows in the calendar view

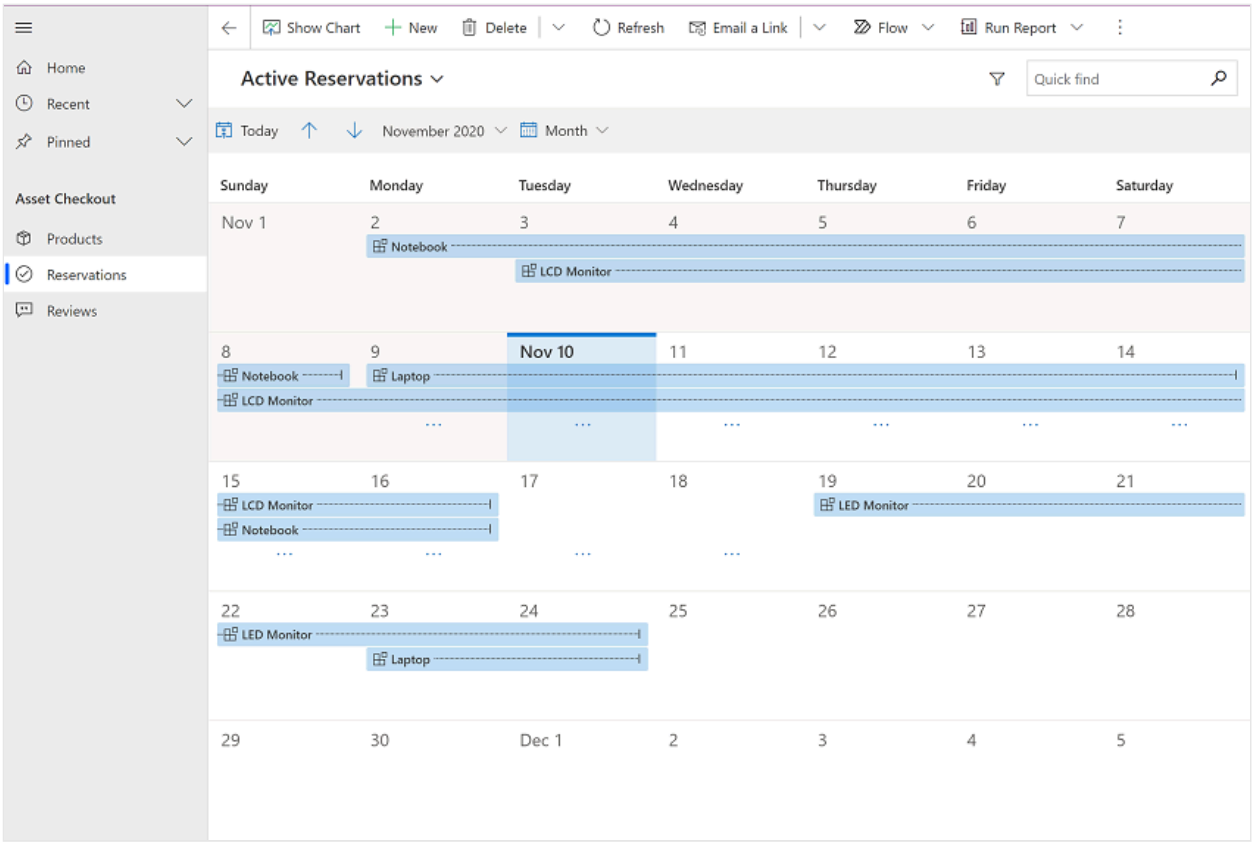
Article • 01/24/2024

The calendar view displays scheduled activities and their associated details. You can view, create, and delete your activities in a day, week, or month view. For the new calendar view to be visible, the **Calendar** control must be added to a table view by a system customizer or a system administrator. For more information, see Add the calendar control to tables. The **Calendar** control can be added to any table.

By default, rows are displayed in a read-only grid.



When the **Calendar** control is added to a table, the default view is changed to the calendar view. Rows associated with the table are displayed in an interactive calendar.

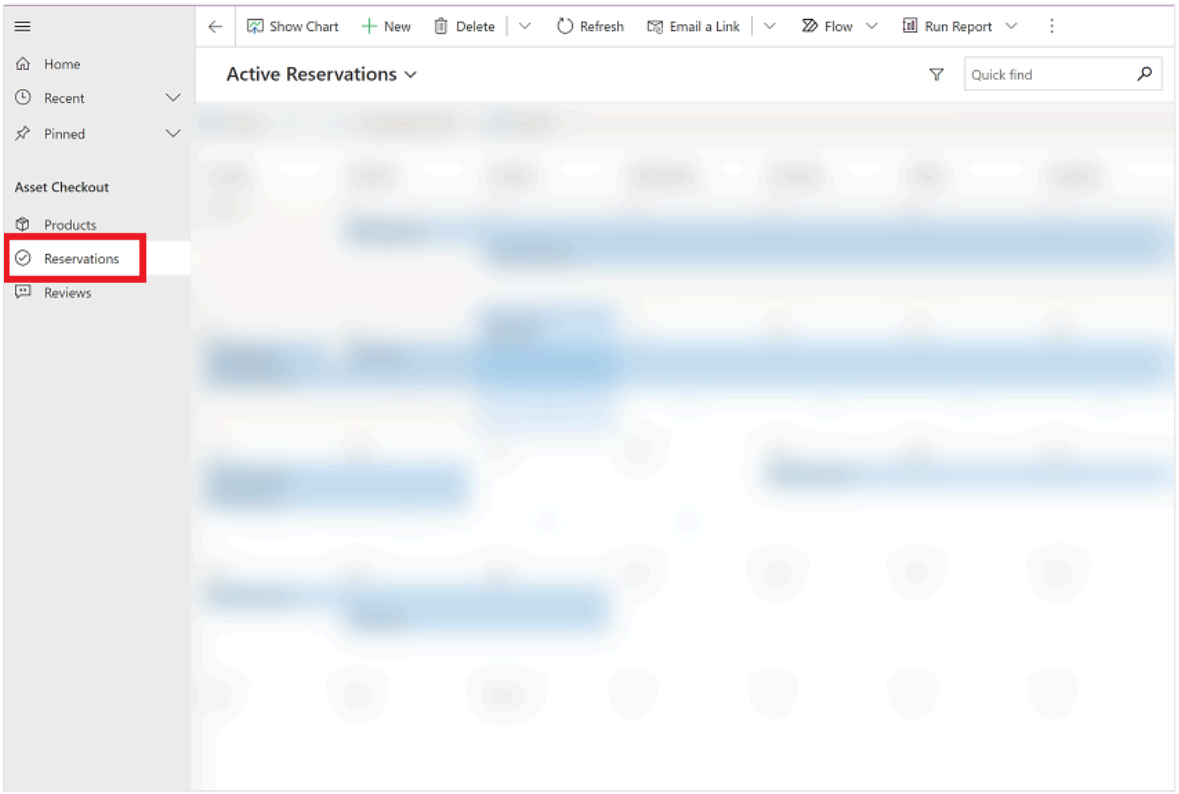


> **Note**
>
> The **Calendar** control is not supported in mobile offline. For more information about mobile offline, see Work offline on your mobile device.

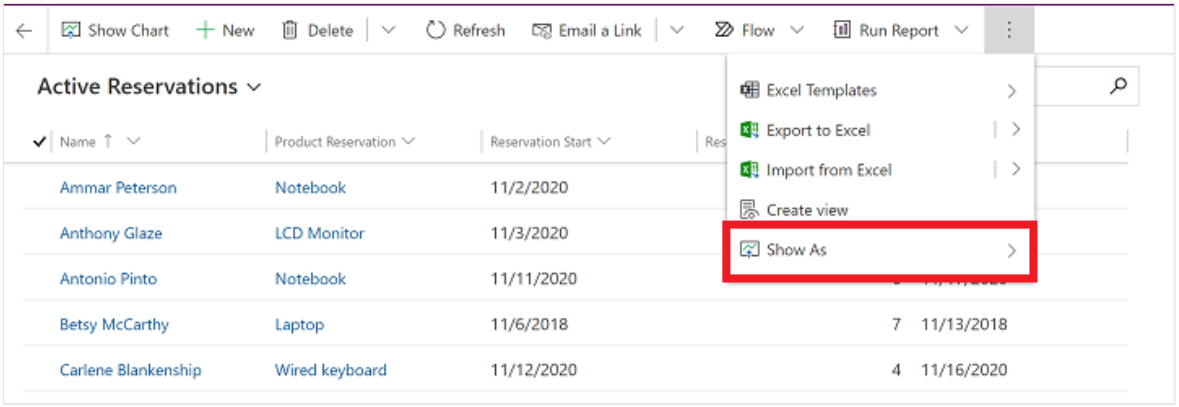
Switch between calendar view and grid view

When the calendar view is turned on, it's the default view for the table. However, you can still switch between the grid view or the calendar view.

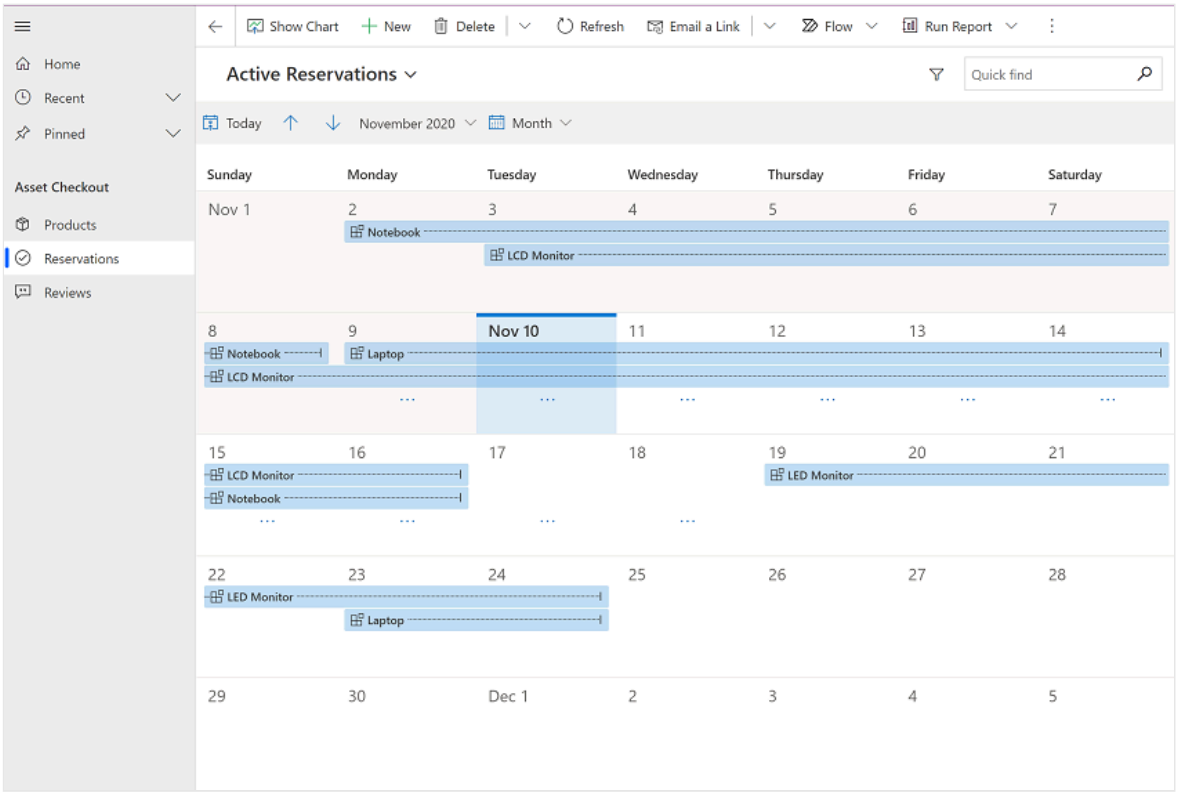
1. To switch between views, on the left navigation pane, select the table that has the calendar view turned on.



2. On the command bar, select the **More commands** icon, and then select **Show As** and choose either **Calendar** or **Read Only Grid**.



When you select the **Calendar** view, activities are displayed in a calendar view.

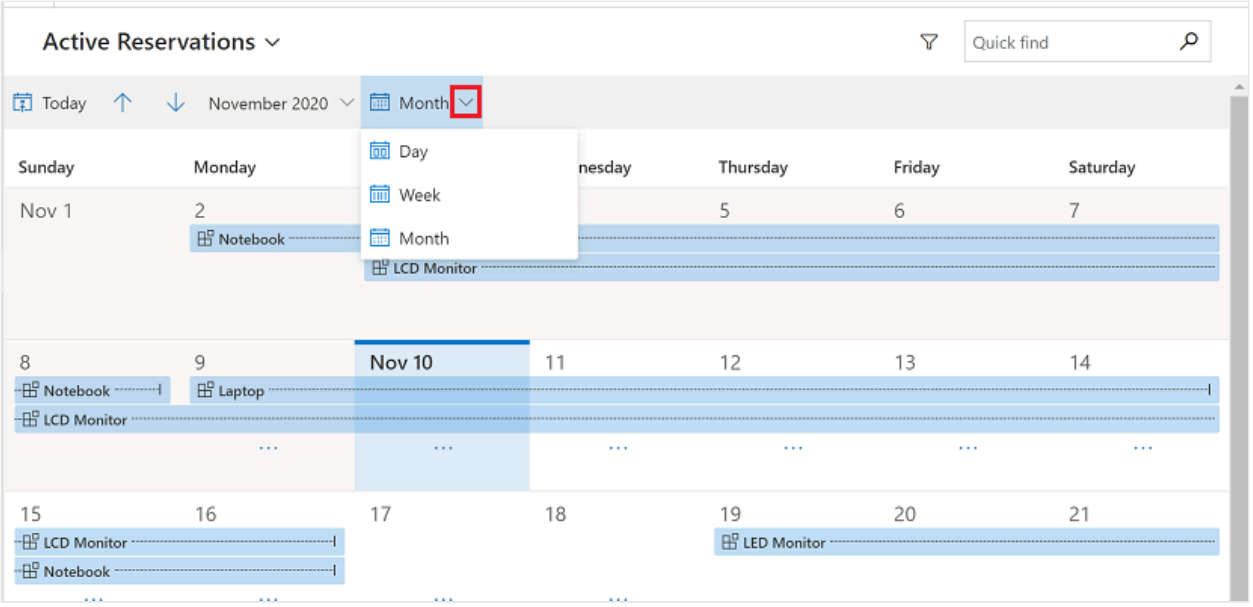


View the calendar by Day, Week, or Month

You can choose from these calendar views:

- **Day**: View the current day.
- **Week**: View the week from Sunday through Saturday, by default. You can change the first day of the week in system settings.
  To view the week numbers, ask your admin to enable displaying the week numbers in the calendar view for your environment. Follow the steps in Customize regional options and select **Show week numbers on Calendar views**.
- **Month**: View the entire month.

To change the calendar format, on the calendar select the drop-down menu and then select **Day**, **Week**, or **Month**.

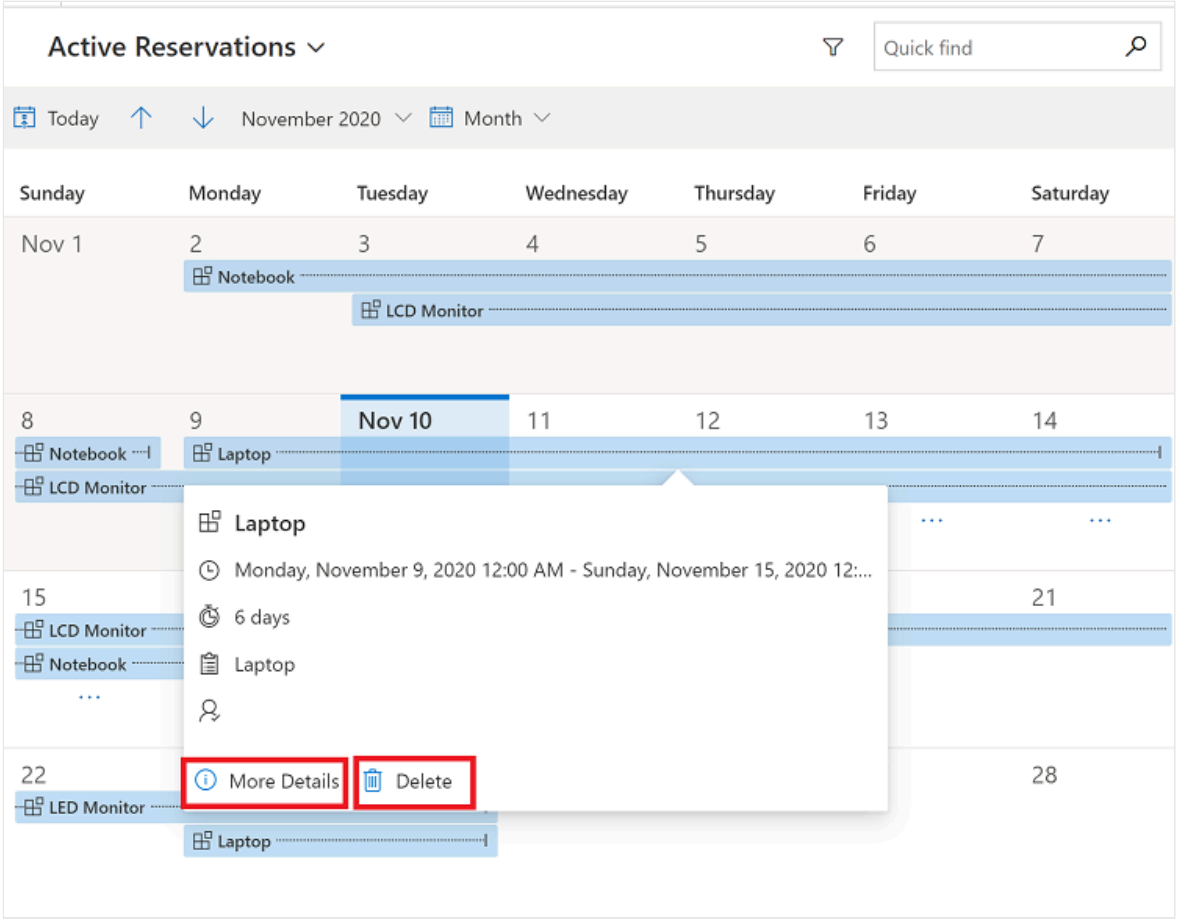


If you want to set a calendar view as the default view, use personalization settings. For more information, see Set personal options.

View details of a calendar item and take action

You can view a summary of a row by selecting it in the calendar. A pop-up window opens and displays the summary information.

- To view complete details about the row, select **More Details** in the pop-up window.
- To delete the row, select **Delete** in the pop-up window.



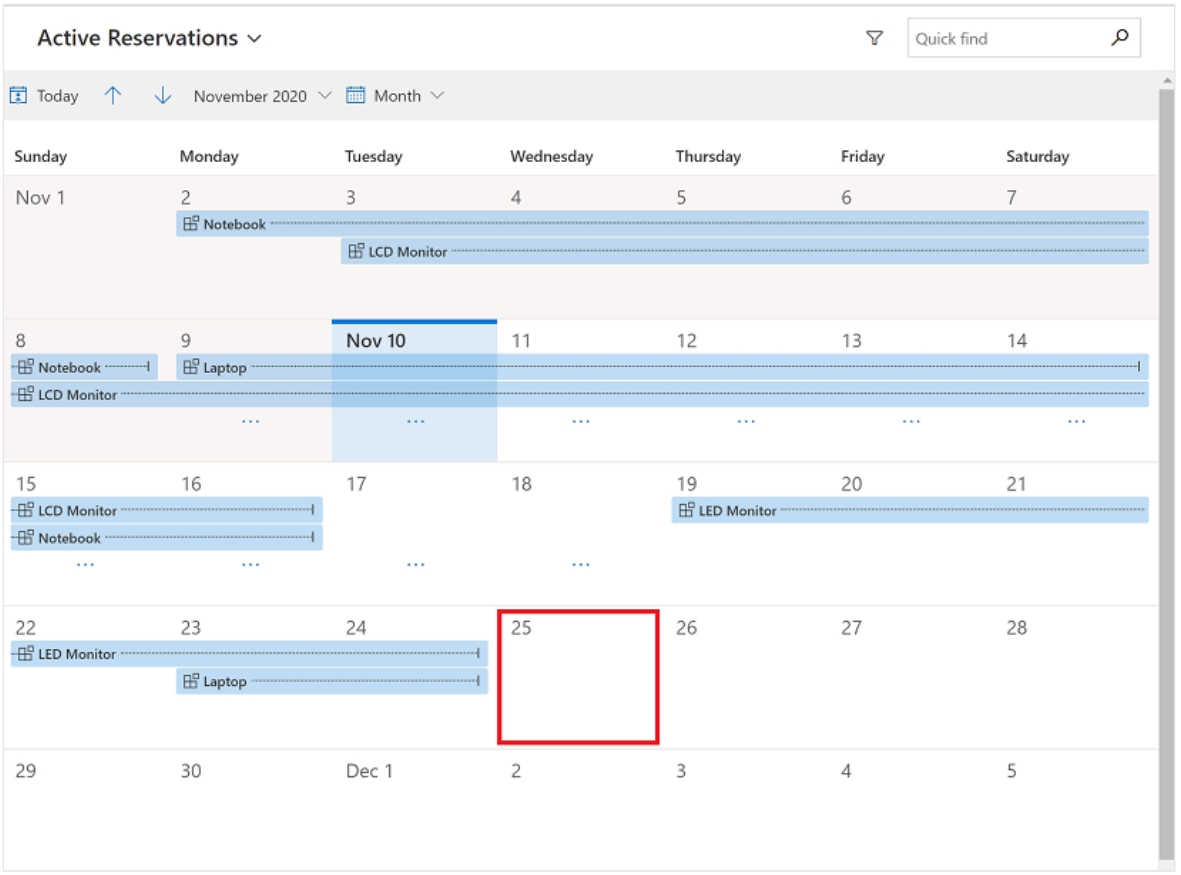
> **Note**
>
> This view is not customizable. If you would like to customize this view, add your suggestion here.

Create a new row from the calendar

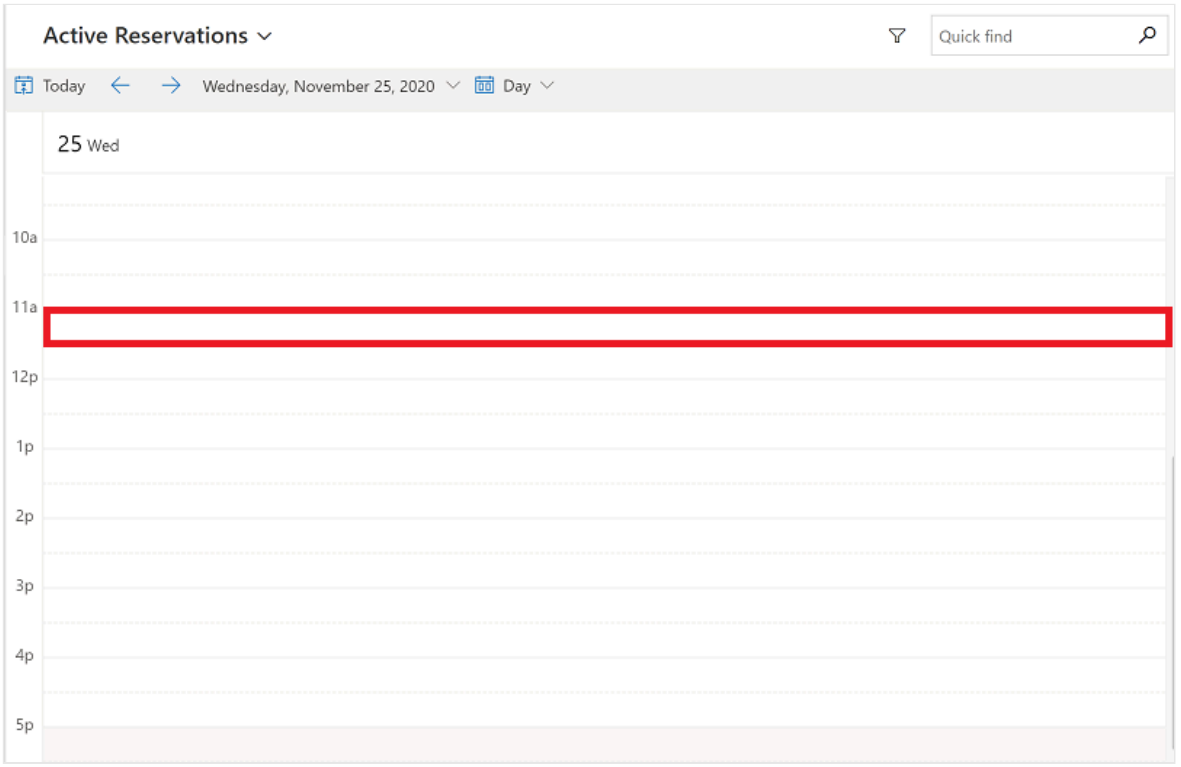
The calendar view allows you to easily create a new row.

1. Depending on your calendar view, create a new row:

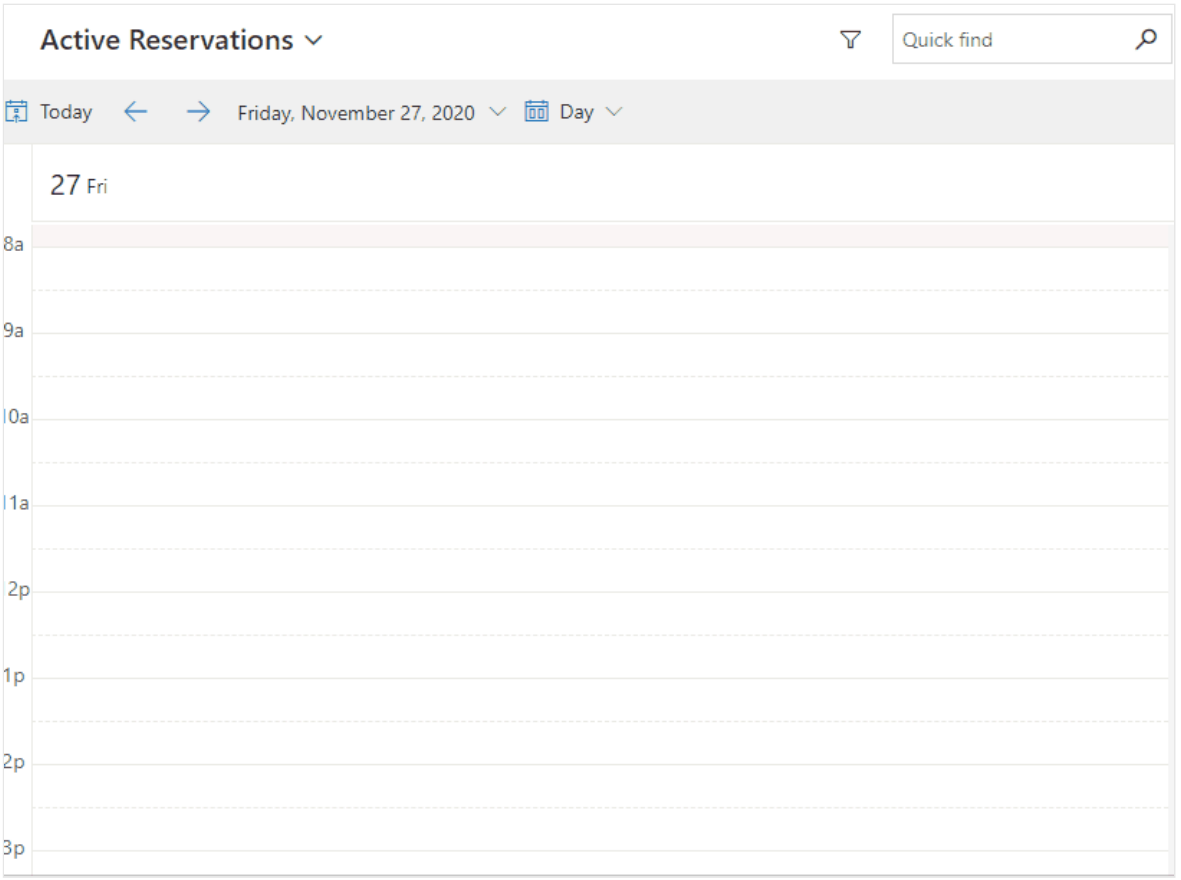
    - On the **Month** view, double-click a date tile, in which you want to create a new row.



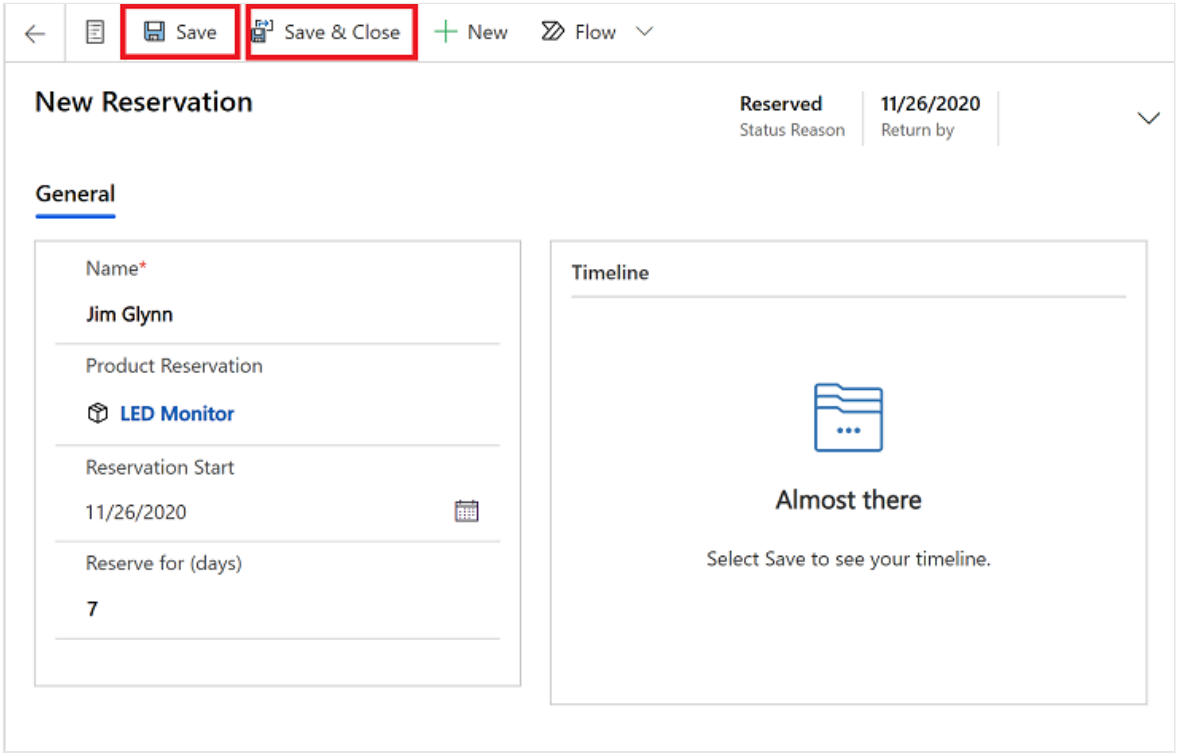
- On the **Day** or **Week** view, select a date and time in which you want to create a row.



- You can also create a new row by selecting and dragging across time slots in **Day** or **Week** view.



2. A new form opens. In this example, a **New Reservation** form is opened. Enter the details, and then select **Save** or **Save & Close**.



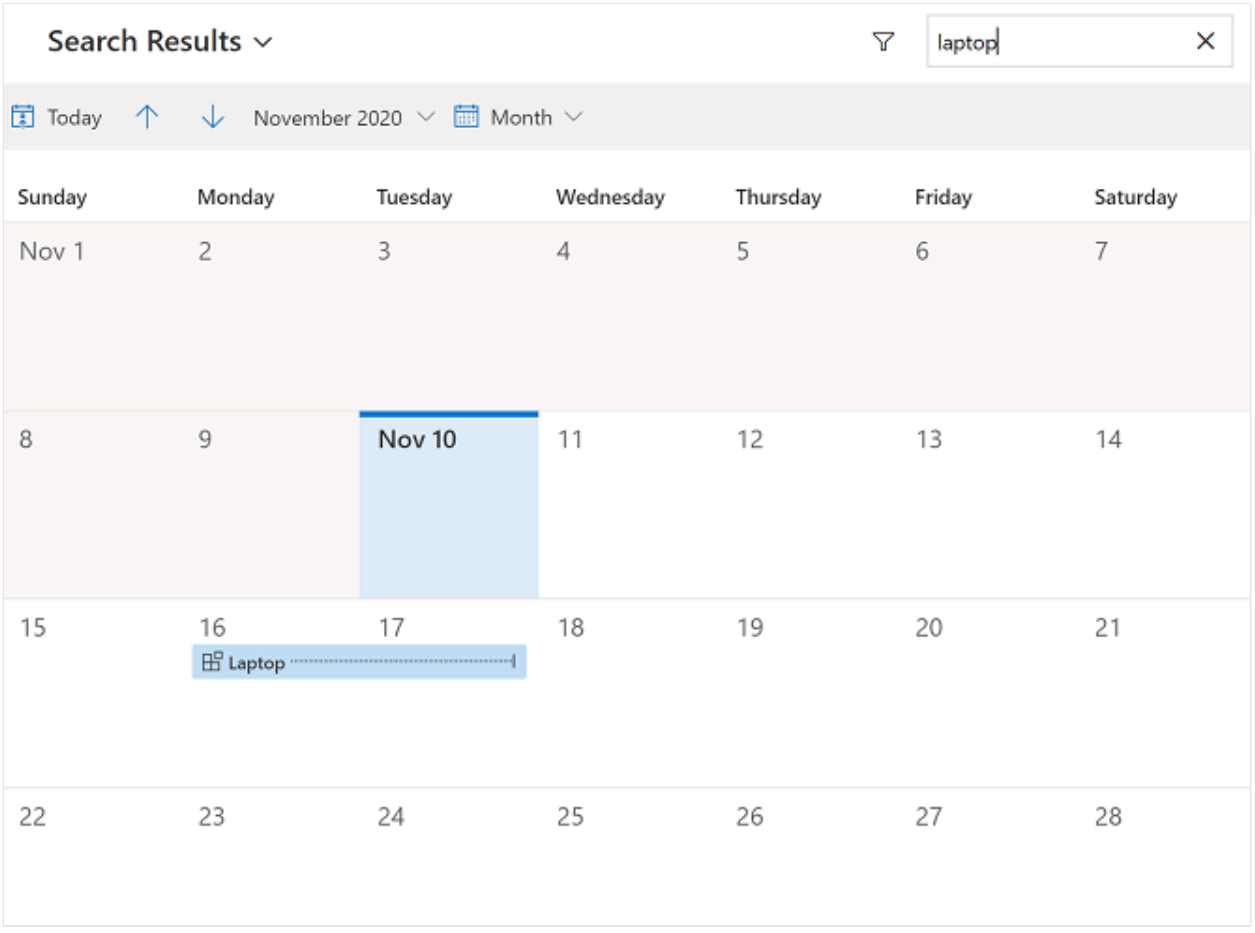
> **Note**
>
> - If your form has start and end times columns, they are filled-in automatically as per the selected date tile or time slots. By default, the

duration is set to 30 minutes.

- When you add the new calendar control on a table other than Activity, the corresponding table row is created when you create a row from the calendar.

Search through rows

You can use the search box to search for data in the view you're in. For example, when you enter the keyword **laptop**, it refreshes the calendar view to show only the rows where the title begins with **laptop**. For more information, see Explore data in a view on a grid page.



See also

Add the calendar control to tables

Feedback

Was this page helpful? Yes No

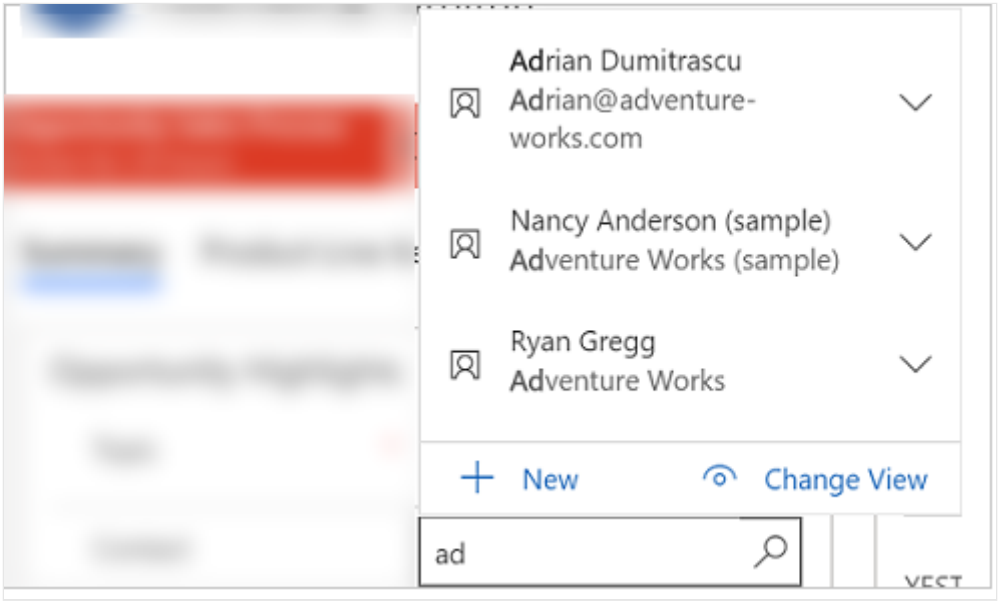
Provide product feedback

Use the lookup column on a row

Article • 04/10/2023

Lookup helps you to choose rows from a related table. When you select a related table and enter search criteria, such as a name or email address, lookup automatically begins to resolve the partial text and displays any matching rows. If no rows are displayed after you have typed the full text of your search criteria, a message is displayed specifying that there are no rows.

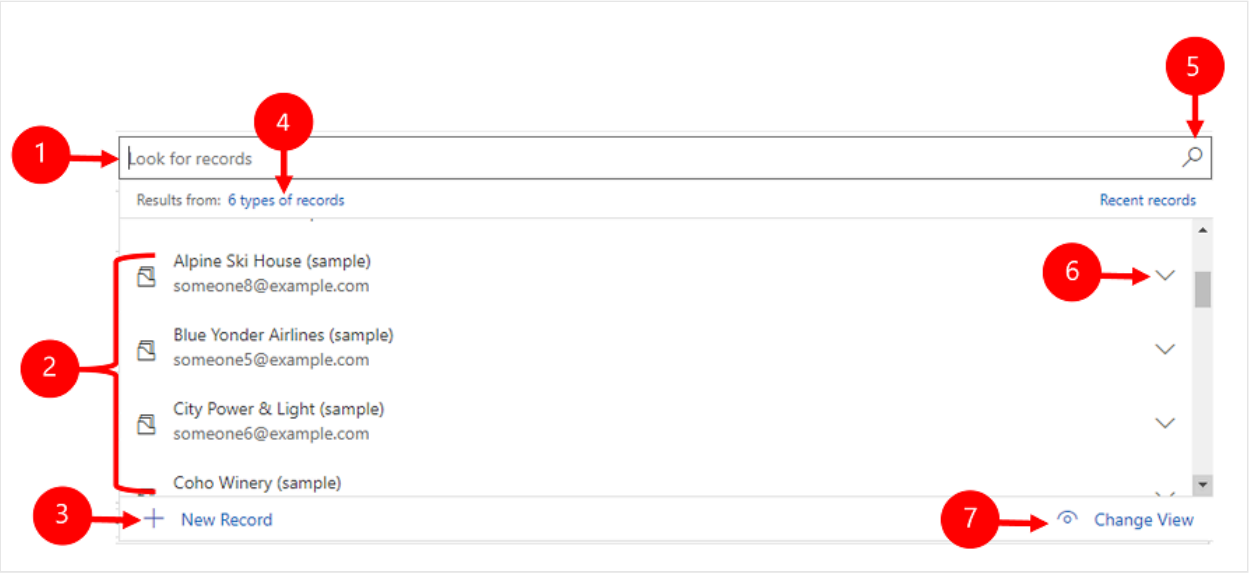
For example, you might search for the name **Adrian Dumitrascu**. When you type **ad**, possible matching rows are automatically populated and displayed.



> **Note**
>
> - An administrator can define the criteria that lookup uses for resolving partial search text.
> - To enable search on multiple columns, see Configure searchable fields.

Also, you can create a new row by selecting the **New** button. You must have sufficient permissions to view the **New** button and create a row. When you select the lookup column, the five most recently used rows are displayed along with five favorite rows. Which rows are displayed depends on your view history and the favorites you've pinned.

For example, if there's only three rows in your history, then the lookup will display those three rows, along with seven of your favorite rows. If you haven't pinned any rows to favorites, then the most recently viewed rows will be displayed.



Legend

1. **Look for records**: Select the textbox and type your search criteria to look for rows.
2. **Recent**: Rows that were recently searched.
3. **New Record**: Create a new row.
4. **Types of records**: Select to filter by a table type such as Contact table or Account table. The number in front of **type of records** represents how many table are available to filter from.
5. **Browse in a lookup column**: To browse a lookup, select the lookup icon (magnifying glass). A full list of items will be shown in the dropdown.
6. **Expand item**: Use the Right and Left arrow keys on your keyboard to expand collapse each item. For more information on keyboard shortcuts, see Use keyboard shortcuts in Power Apps.
7. **Change View**: Select to switch to another view. For more information on views, see Explore data in a view on a grid page.

Types of lookups

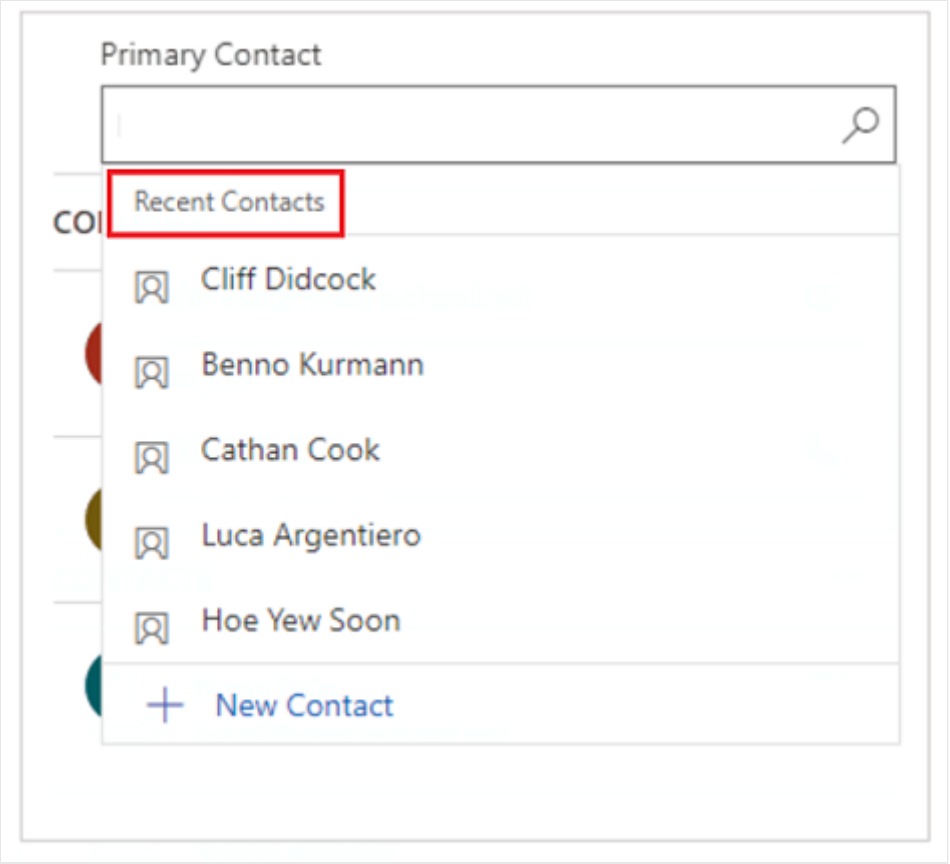
Lookups are classified into the following:

- **Simple lookup:** Select a single row from a single table.
- **PartyList-type lookup:** Use to select multiple rows from multiple tables in a lookup. Use partylist-type columns to select multiple rows. This allows you to add each row by performing a new search, multiple times. Every time you select a row, you will be able to perform a new search for another row. Note, you can't customize the system and disable views for PartyList-type look.
- **Regarding-type lookup:** Use to select a single row from multiple tables in a lookup. Activities must be enabled for a table to use the Regarding lookup. For

more information: Enable activities for a table

Search in a lookup column

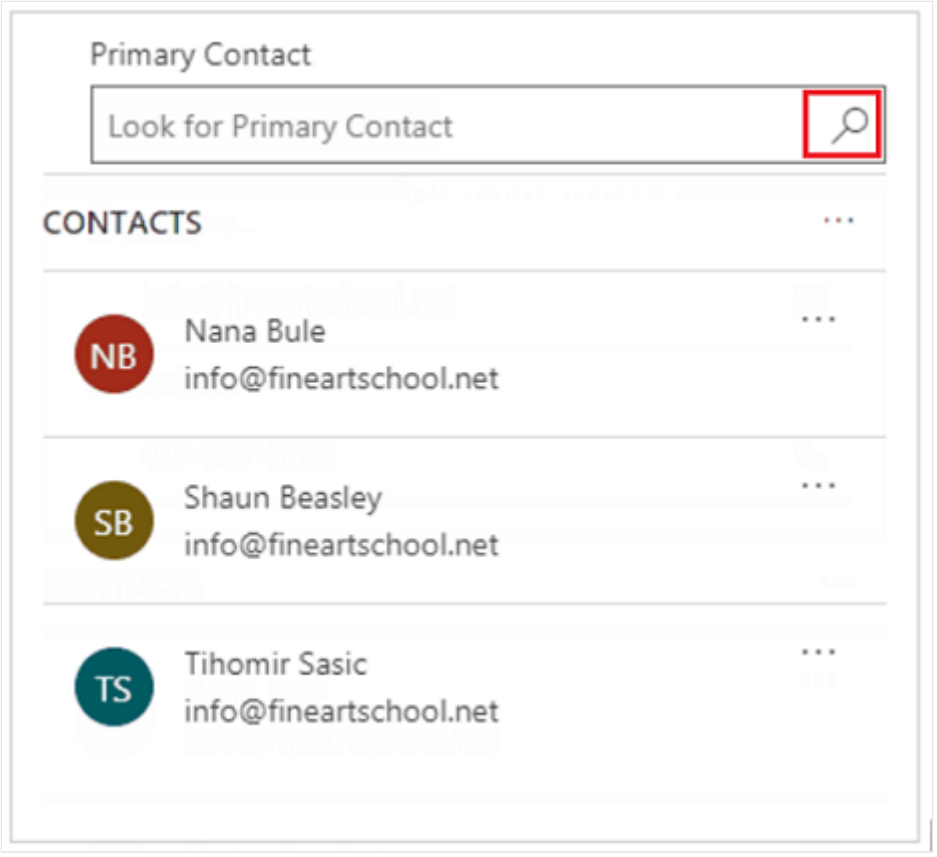
To search a lookup, select the textbox and type your search criteria. If recent rows are enabled for your lookup, your recent rows will be displayed when you select the textbox.



> **Note**
>
> The default search result for lookup search is, **begins with**. This means results include rows that begin with a specific word. For example, if you want to search for **Alpine Ski House**, type **alp** in the search box; if you type **ski**, the row will not show up in the search result.
>
> For a wildcard search use asterisks: For example, type *ski or *ski*.

Browse in a lookup column

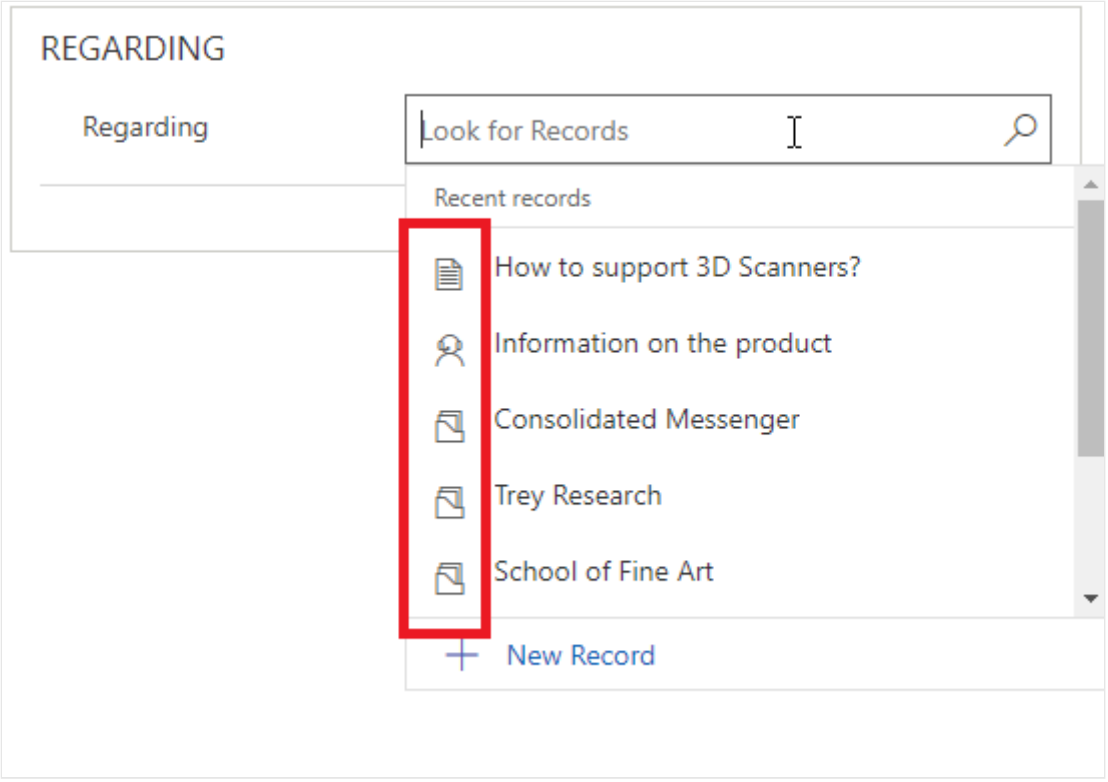
To browse a lookup, select the lookup icon (magnifying glass). A full list of items will be shown in the dropdown.



Most recently used row type images

The most recently used list of rows shows an image to help distinguish between row types.

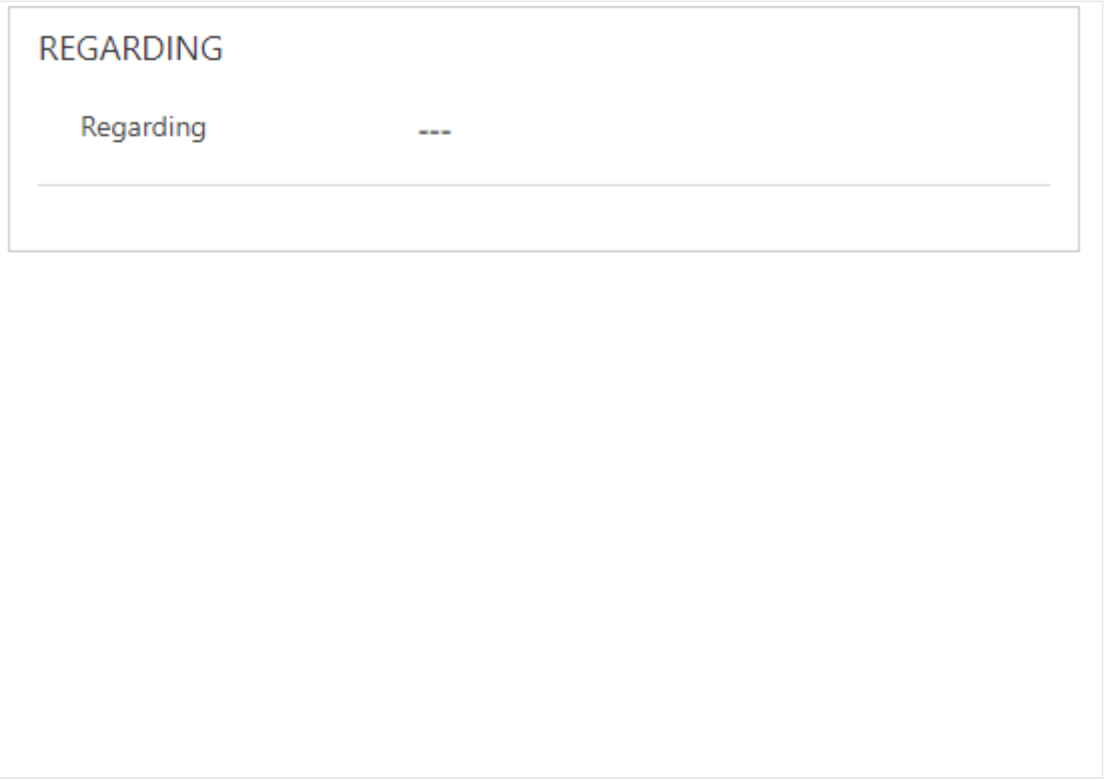
> **Note**
>
> Recent rows are not filtered by search term, selected view or related rows.



Row type selection list

When results span multiple row types, you can see how many types of rows there are and select them from the list. The **Lookup Row** option is not available in Unified Interface. Instead, use the following to search:

- To look for rows, enter text in the search box.
- To search by table type select **types of rows**.
- Select **Change View** to select a view.



Create a new row if you don’t find an existing row

If you do not find a row, select **New** in the lookup area to create a new row. When the row is created it will show up in the lookup.

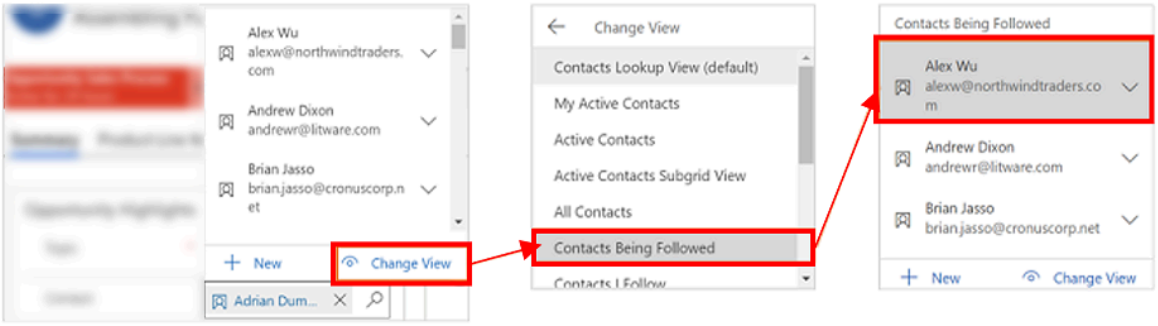
Replace the selected lookup item

You can replace an existing row while using simple and regarding-type lookups. Search for a row and then select the row to replace it with a new row.

Change a view in a lookup column

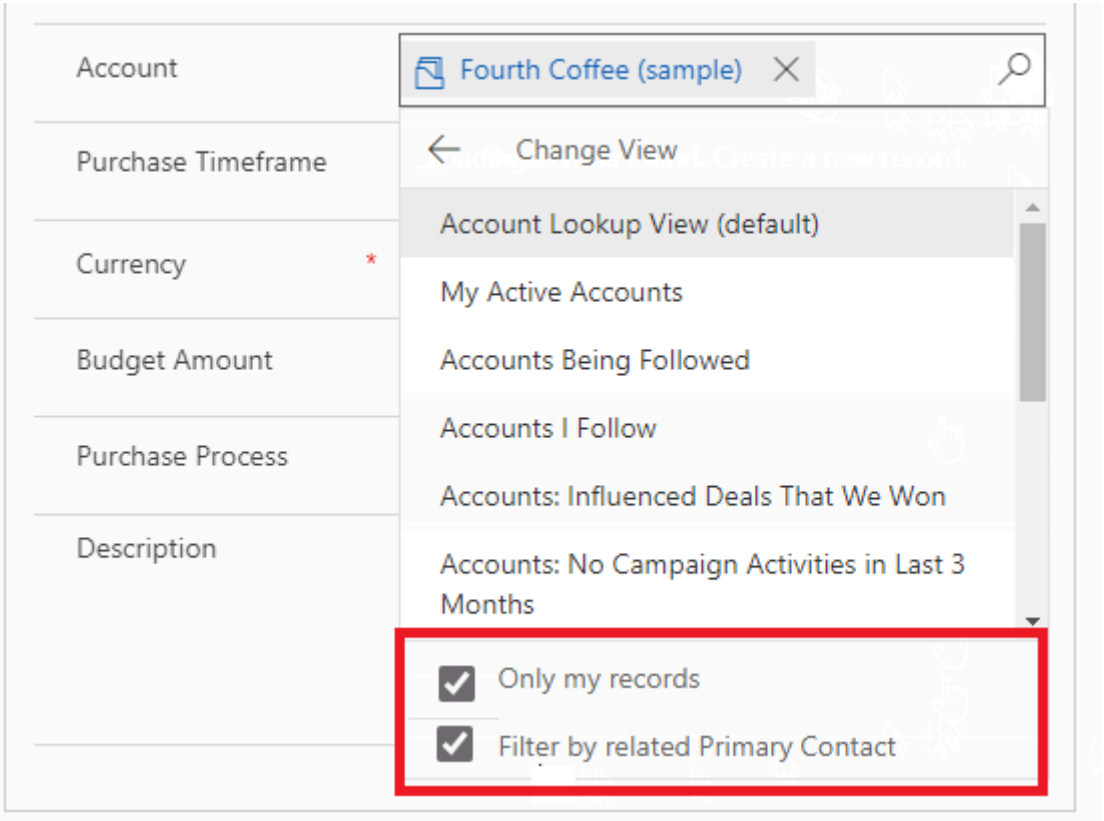
Selecting **Change View** lets you determine:

- How you want to view rows such as **Contacts Being Followed**, **Contacts Lookup View**, or **Active Contacts**.
- What you want to view in the rows, such as name, email, or telephone number. For example, if you want to view only the contacts that you follow, select **Change View** > **Contacts being followed**. Only the contacts that you are following will be displayed, as illustrated here.



Filter by, Only my rows or Filter by related primary contact

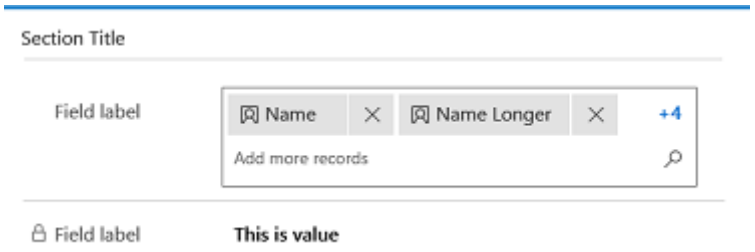
To apply extra filters, in the **Change View** menu, select **Only my rows** or **Filter by related Primary Contact**.



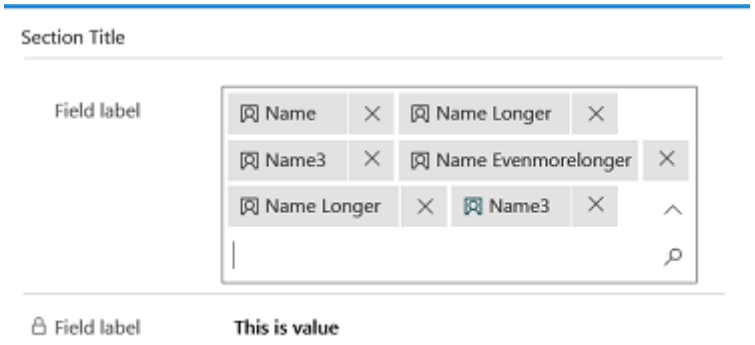
Choose from multiple rows

When lookup has more rows in a column than can fit in the available display area, the display area is collapsed—that is, the rows that do fit the display area are shown next to the number of rows that are not shown. To view all rows, select the number. The following images show the difference between collapsed and non-collapsed columns.

Collapsed:



Non-collapsed:



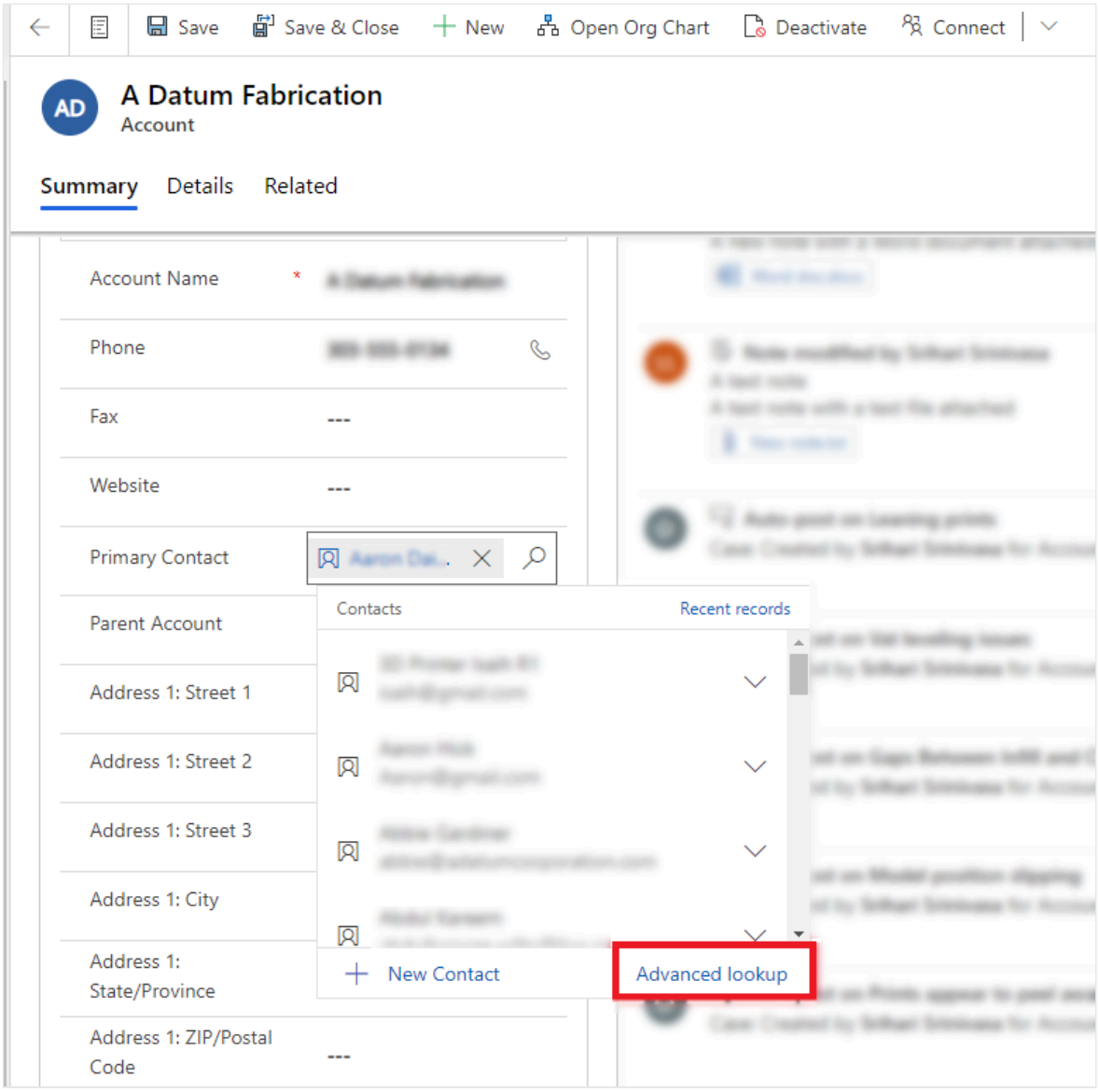
Advanced lookup

The **Advanced lookup** feature is available for early access in 2021 release wave 2. To use **Advanced lookup** your administrator needs to enable early access. For more information, see Manage behavior settings.

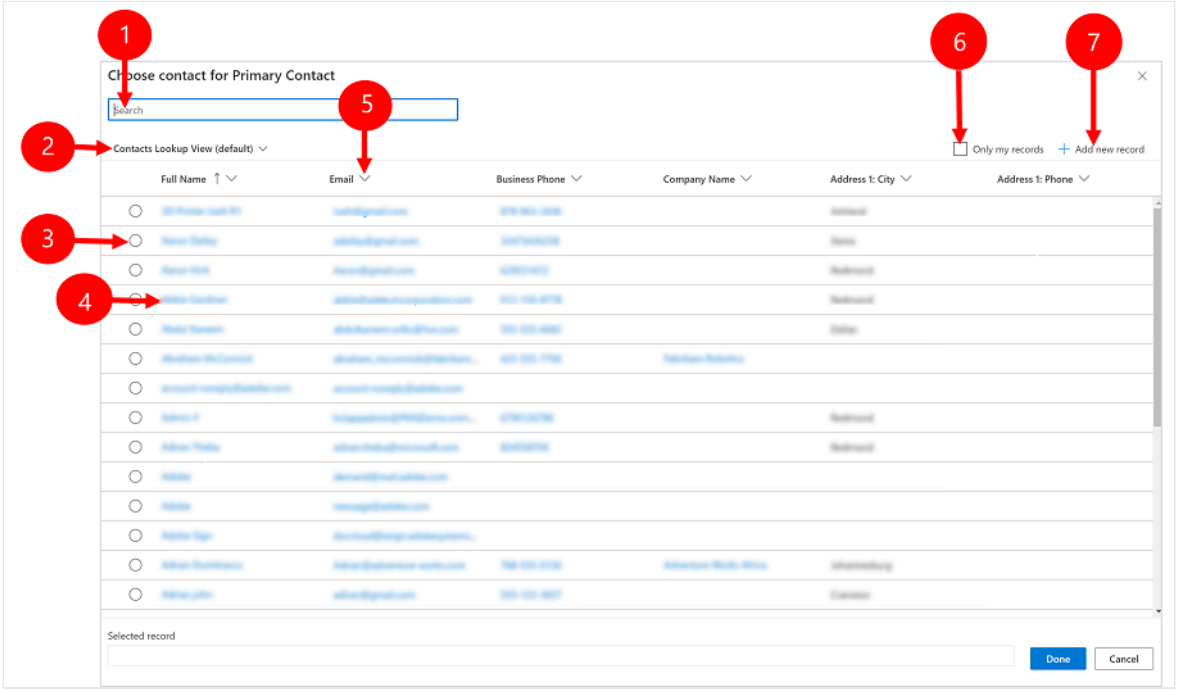
Once enabled **Advanced lookup** is available on any lookup column that's on a quick create form, edit form, or dialog box. With **Advanced lookup** you can easily sort rows and switch between tables or views in a lookup column.

Advanced lookup supports customizations on lookup control through Client APIs such as *addCustomView*, *addCustomFilter*, *addPreSearch* and *removePreSearch*.

1. To use **Advanced lookup**, select the lookup icon (magnifying glass) on any lookup column and then select **Advanced lookup**.



2. The **Advanced lookup** gird opens with a list of rows.



Legend

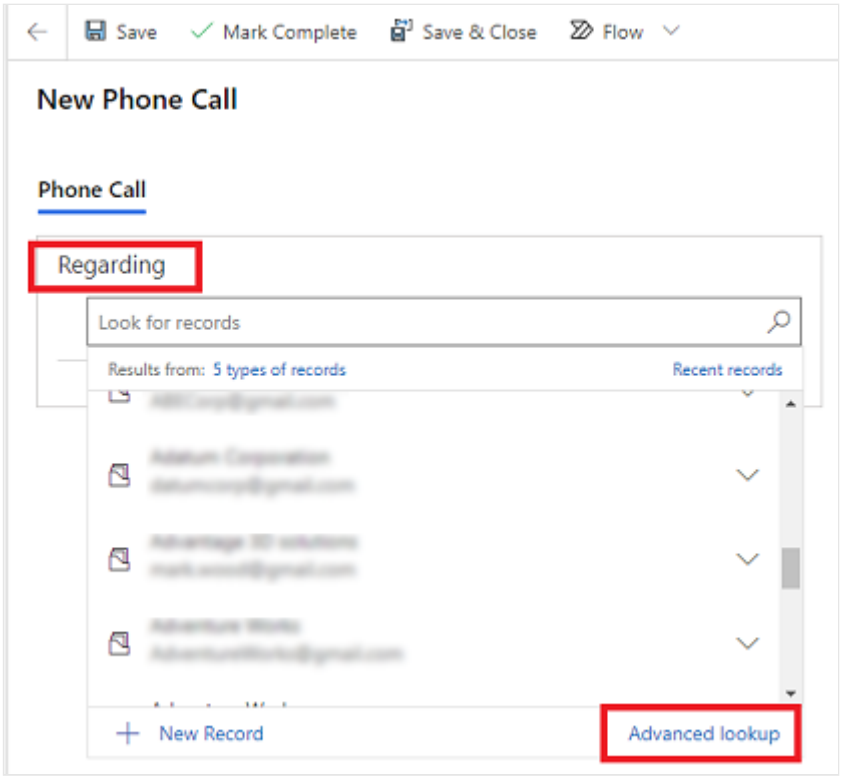
1. **Search**: Enter search text to find the row that you're looking for. When you changes views the search text is retained.
2. **Change views**: Select to change views.
3. **Select row**: Select the radio button to choose a row for the lookup and then select **Done**. The selected row will appear at the bottom under **Selected record**.
4. **Open row**: Select to open the row and make edits and then save your changes. Any saved changes are reflected in the advanced lookup grid.
5. **Sort**: Select the arrow at the top of any column to sort in ascending or descending order.
6. **Only my records**: Select to only show rows that you own.
7. **Add new record**: Opens the quick create form where you can add a new row for the table. When you save the new row, the advanced lookup grid will refresh and show the newly added row.

> ⚠ **Warning**
>
> Advanced lookup isn't available with offline scenarios. The **Advanced lookup** button is replaced by a **Change view** button when the user is offline.

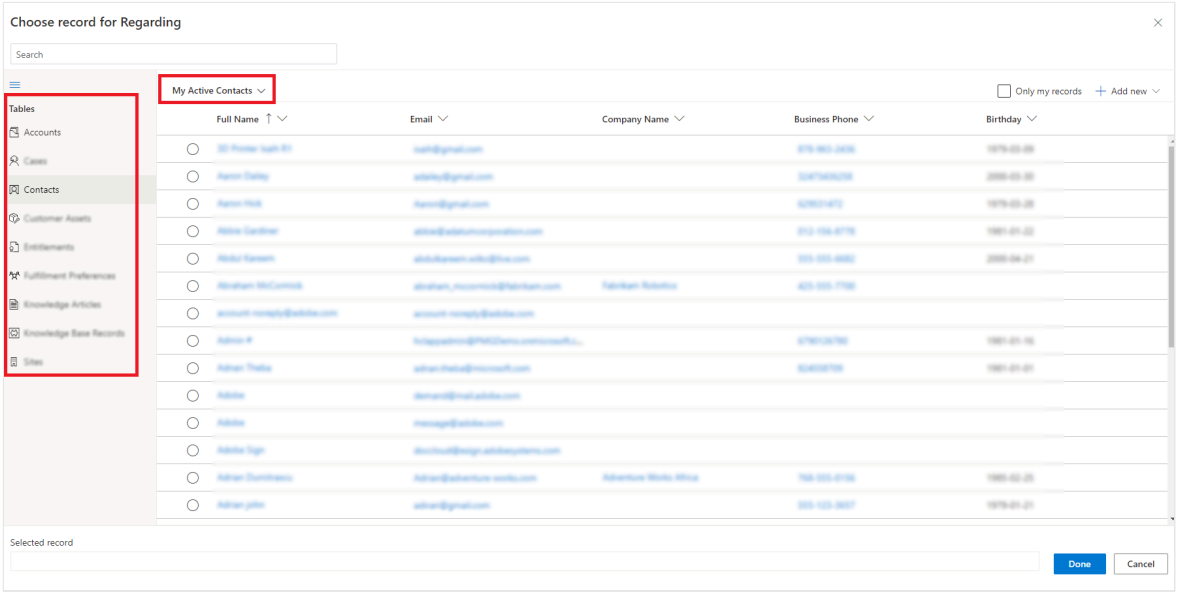
Regarding advanced lookup

In a **Regarding** lookup, you can switch between tables and their views

1. In a **Regarding** lookup column, select the lookup icon (magnifying glass) then select **Advanced lookup**.



2. On the advanced lookup grid, select a table from the list to see rows for the selected table. You can also select the current view to change views.

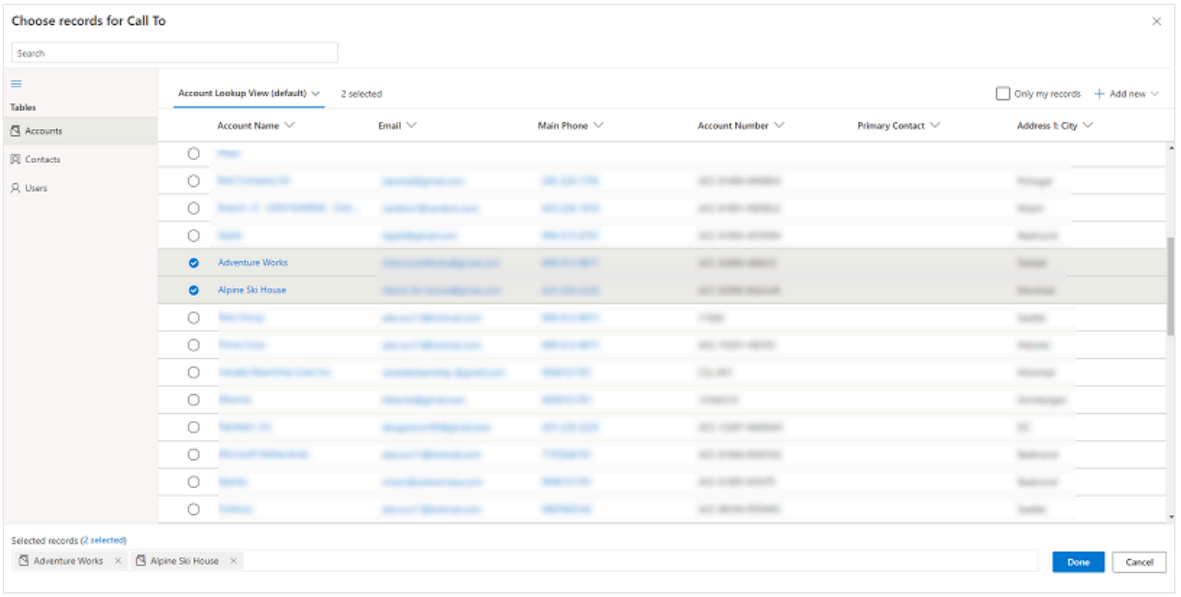


Party list advanced lookup

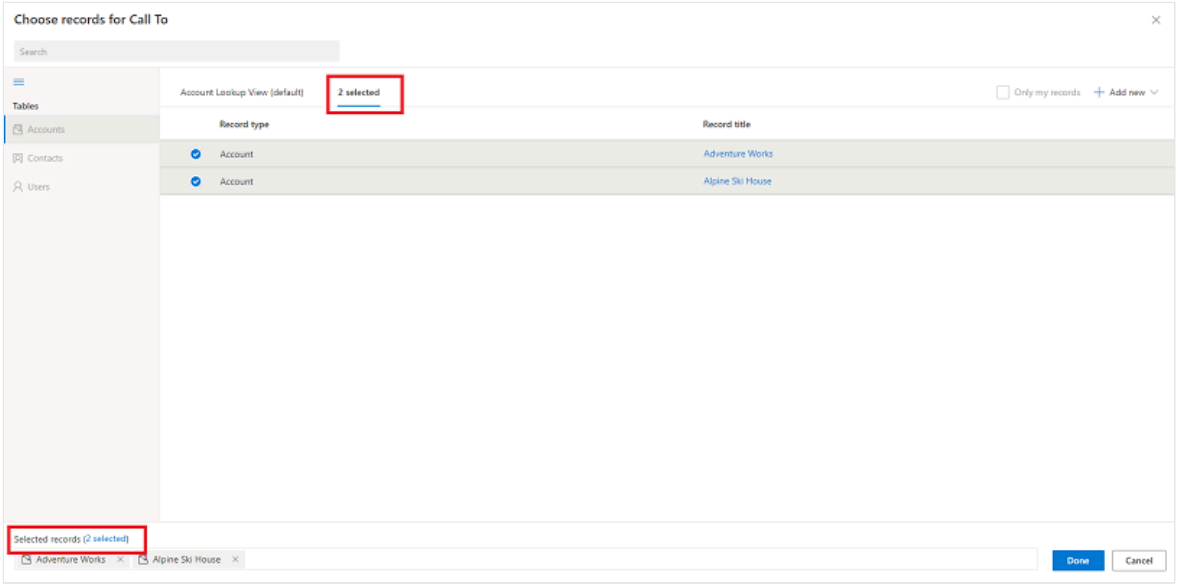
For lookup columns that is a party list, you can select multiple rows from different tables such as the **Call To** field on a phone call activity.

1. Go to a lookup column with party list and select the lookup icon (magnifying glass) then select **Advanced lookup**.

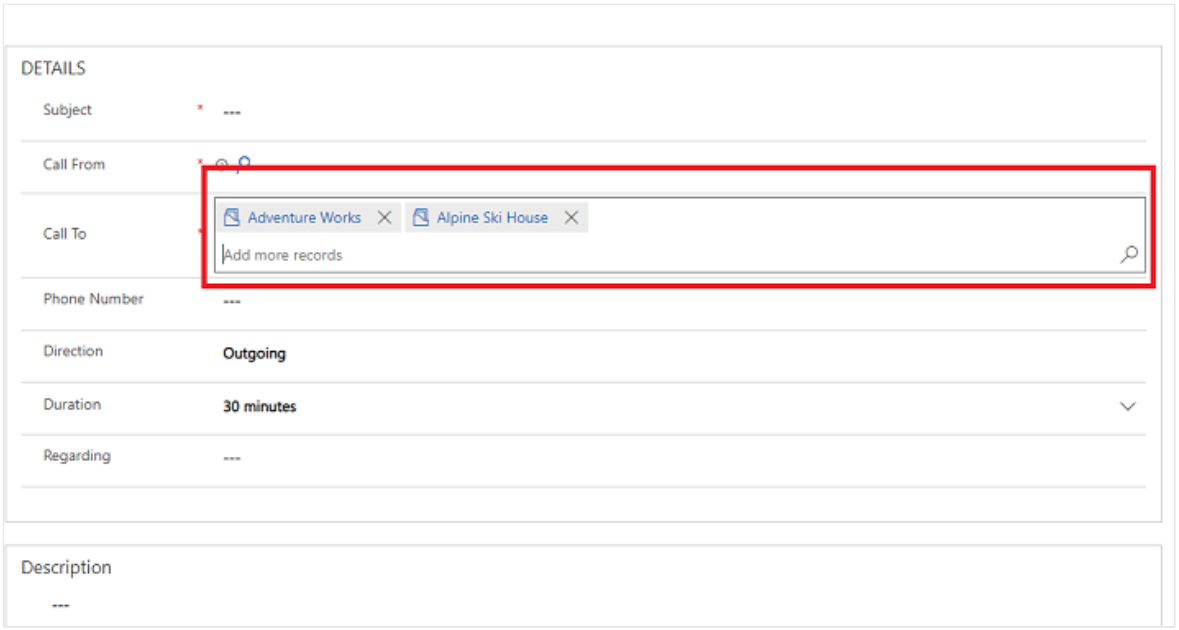
2. On the advanced lookup grid, select from multiple rows from different tables.



3. Select the **(number of rows) selected** tab or the link at the bottom to review and edit your choices and then select **Done**.



4. The selected rows appear in the lookup.



Feedback

Was this page helpful? Yes No

Provide product feedback

View the profile card for a contact or user

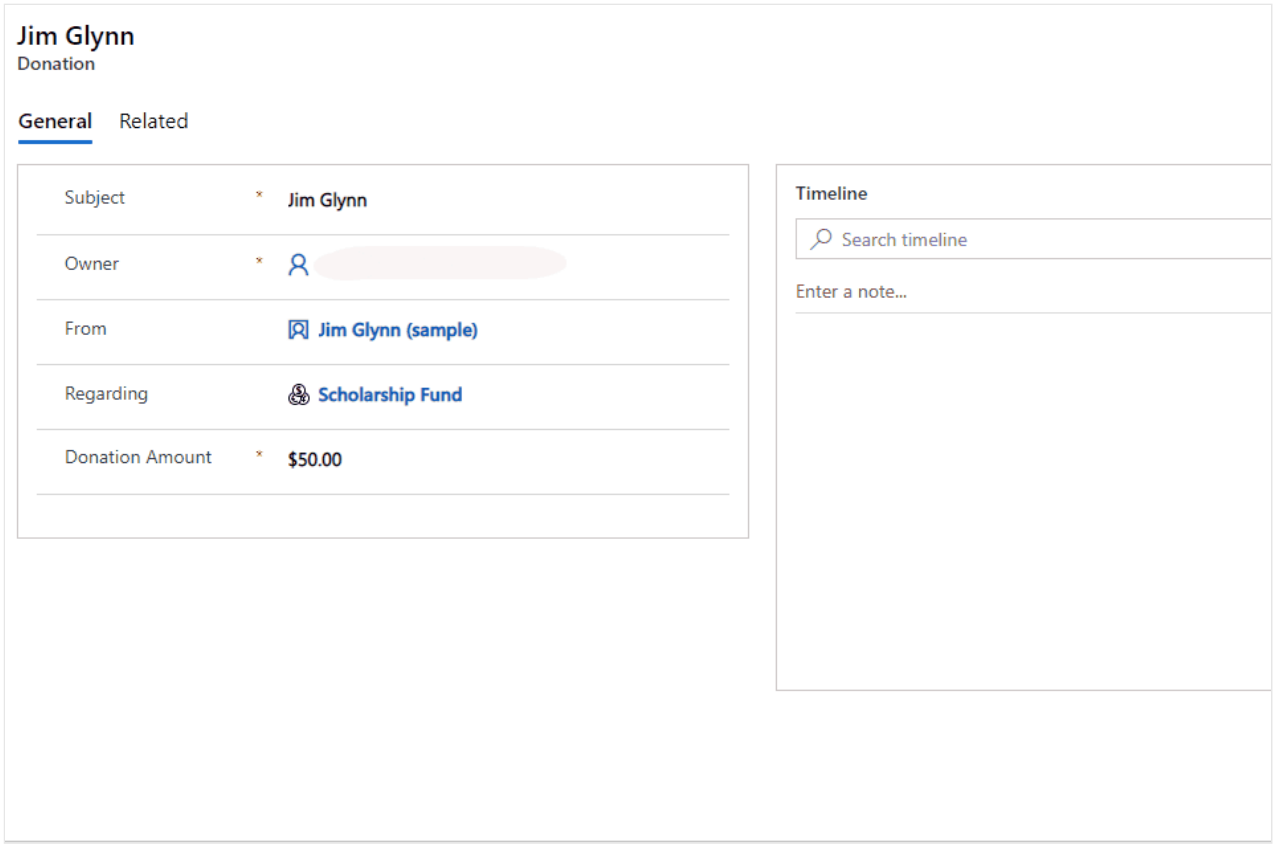
When you select someone's name or picture in Outlook or other Office apps, you can find information related to them on their profile card. The profile card is also sometimes referred to as contact card or people card. In model-driven Power Apps, profile cards can be seen on contact and user rows. You can only view them when they are in a lookup.

For more information about profile cards, see Profile cards in Office 365.

> **Note**
>
> - Profile cards are available on contacts and users in any Unified Interface app. For information, see **Enable viewing profile cards (for admins)**.
> - The profile card in Microsoft Dataverse is not displayed if multi-factor authentication is turned on for Office Delve service in Azure Active Directory.
> - Profile cards are not available when the signed-in user is a Microsoft Entra ID guest user.

View a contact's profile

1. Open a row and go to a lookup column that has a contact or owner name.
2. Hover over the name to view the profile card.



FAQs

Where can I see profile cards in Dynamics 365?

Profile cards can be seen on contact and user rows. You can only view them when they are in a lookup.

Where is information shown in the profile card coming from?

The information shown on the contact profile card is fetched from Dataverse (and not Microsoft Exchange).

The information shown on the user profile card is fetched from Office 365 (Azure Active Directory). For more information, see Profile cards in Office 365 (admin section) .

How can I customize the columns shown on the profile card?

Currently, the list of columns displayed on the profile card is not open for customization.

Why is the Start chat option on the profile card disabled (greyed out)?

The **Start chat** and the **Send Email** options on the profile card will open your default instant message and email apps. The **Start chat** option is enabled if the person you are trying to contact is in the same Azure Active Directory environment as you or is a federated contact.

Why isn't the profile card opening for a user?

If profile cards are not appearing when hovering over any user's name, the signed-in user might be a Microsoft Entra ID guest user. Profile cards aren't supported for signed-in guest users.

Last updated on 04/02/2026

Assign or share a row

Article • 09/26/2022

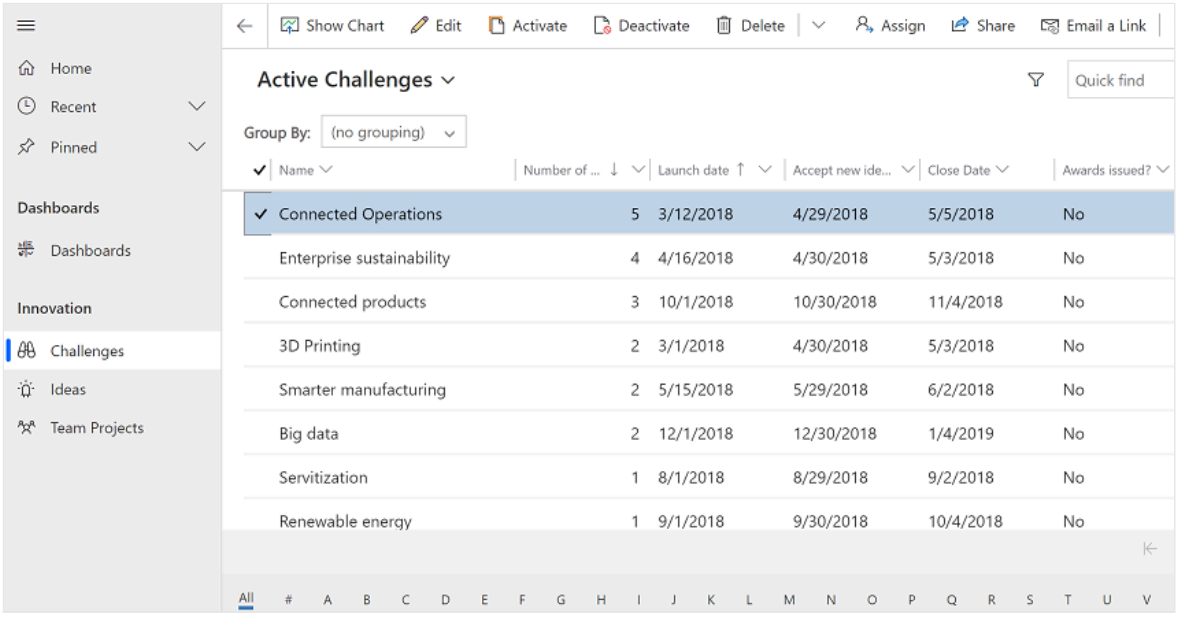
When you create a row, you are the owner of the row. If you would like another person in your organization to take ownership of the row, then you can assign the row to that person. You can assign a row to a user or team. You can also assign a row to yourself that another user owns but you need to have system administrator privilege to do this.

If you want to keep ownership of the row but let someone else work on it with you, then use the **Share** option to share the row. For more information, see Share a row.

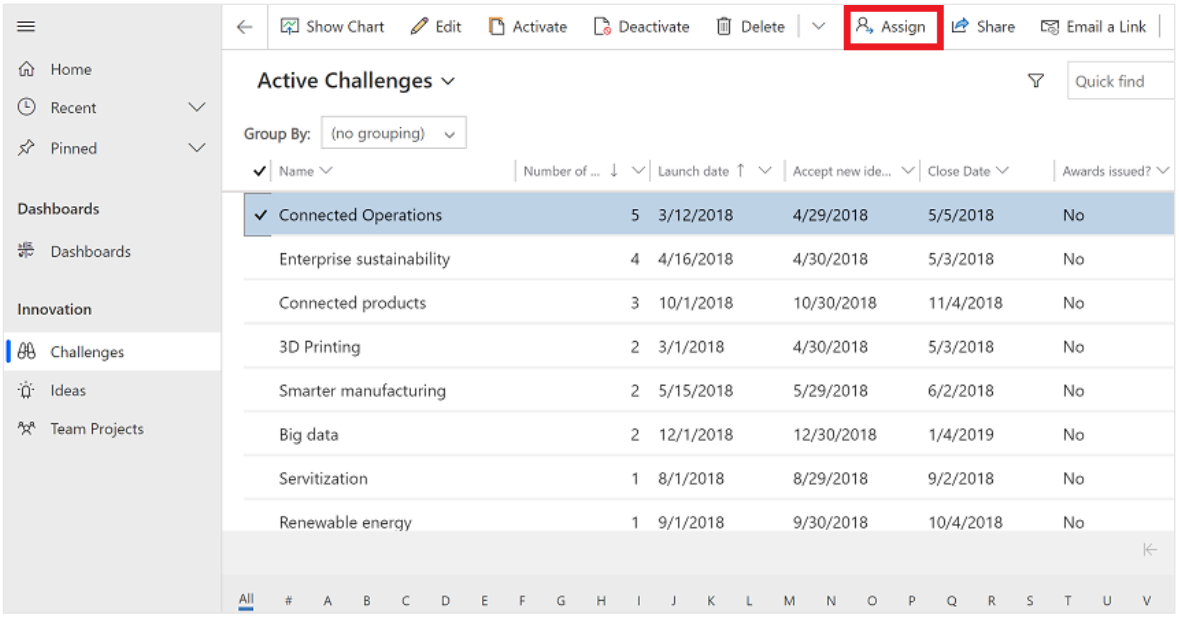
For more information on how privileges and access works, see How access to a row is determined.

Assign a row to a user or team

1. From a list of rows, select the row that you want to assign to someone else. You can select multiple rows.

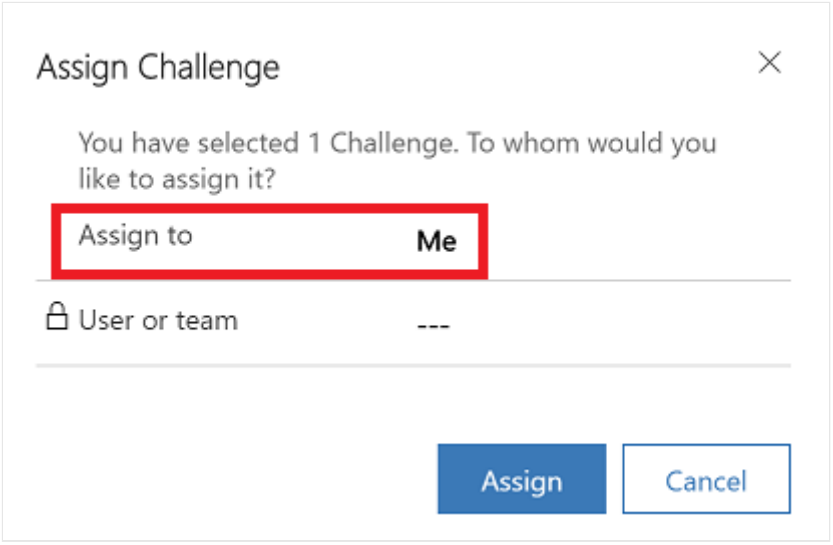


2. On the command bar, select **Assign**.

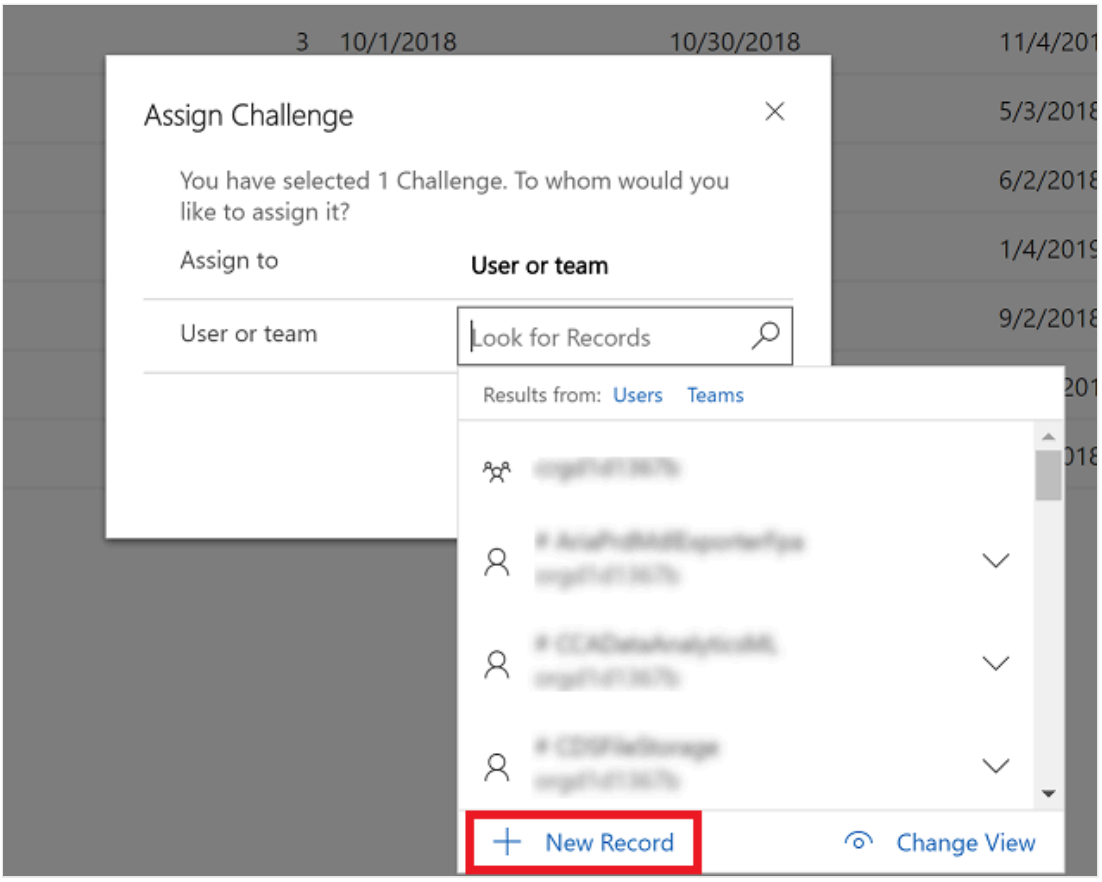


3. In the assign dialog box, select the **Assign to** column, and choose one of the following options:

   - Select **Me** to assign the row to yourself and them select **Assign**. Remember only a system administrator can assign a row that belongs to someone else to themself.



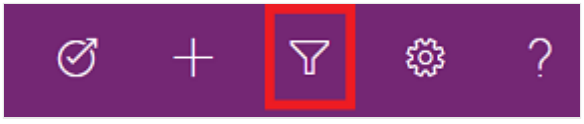
   - Select **User or Team** and then enter the name of the user or team or use the lookup to find them. Or, select **New Row** to create a new user or team row. When you're done select **Assign**.



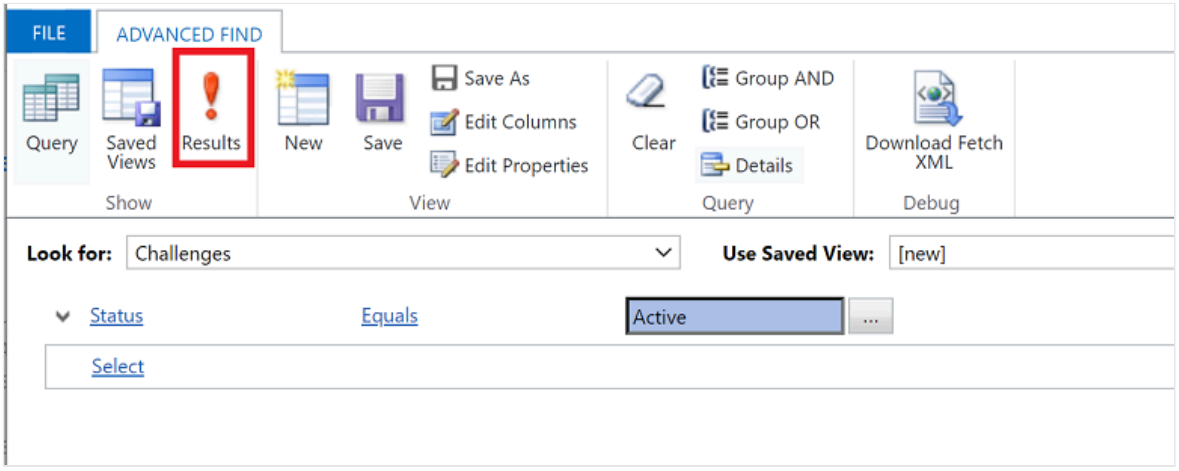
Use advanced find to reassign rows

Use advanced find to search for rows and then reassign them to someone else. For more information on advanced find, see Create, edit, or save an Advanced Find search.

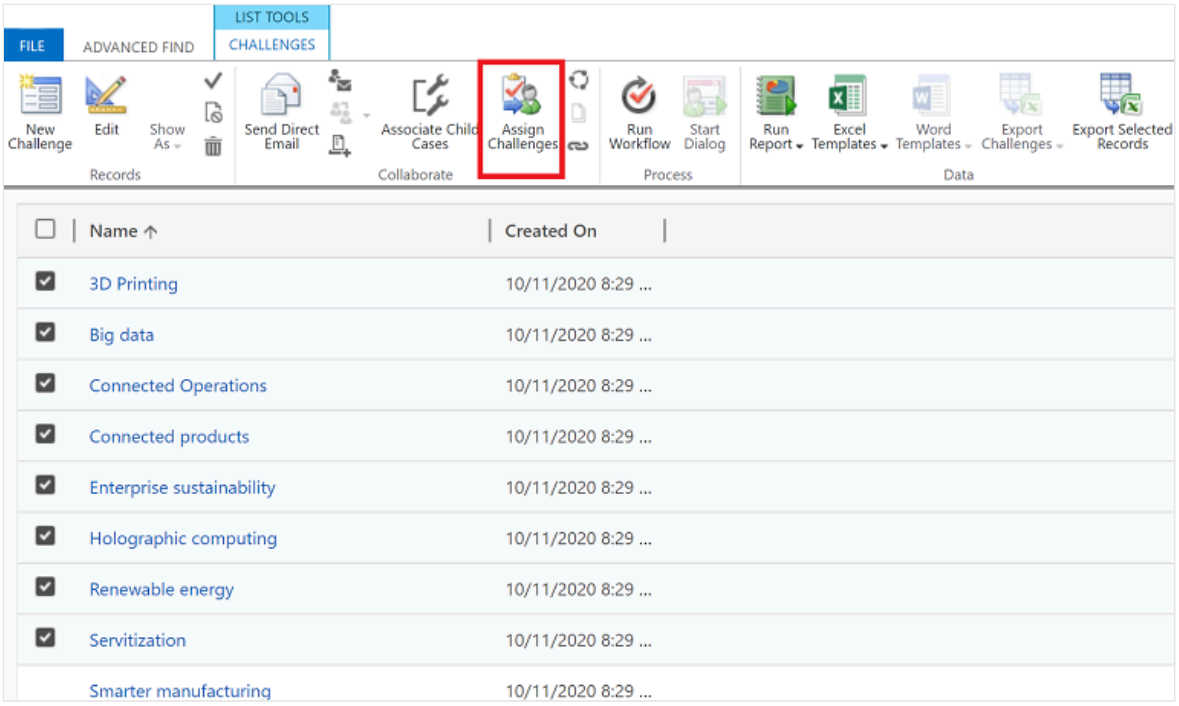
1. On the command bar, select **Advanced Find**.



2. Use the advanced find search to find rows that you want to assign to someone else. For example, to look for active row types that are **Challenges**, in **Look for:** enter **Challenges** and status equals **Active**. Then select **Results** to run the query.



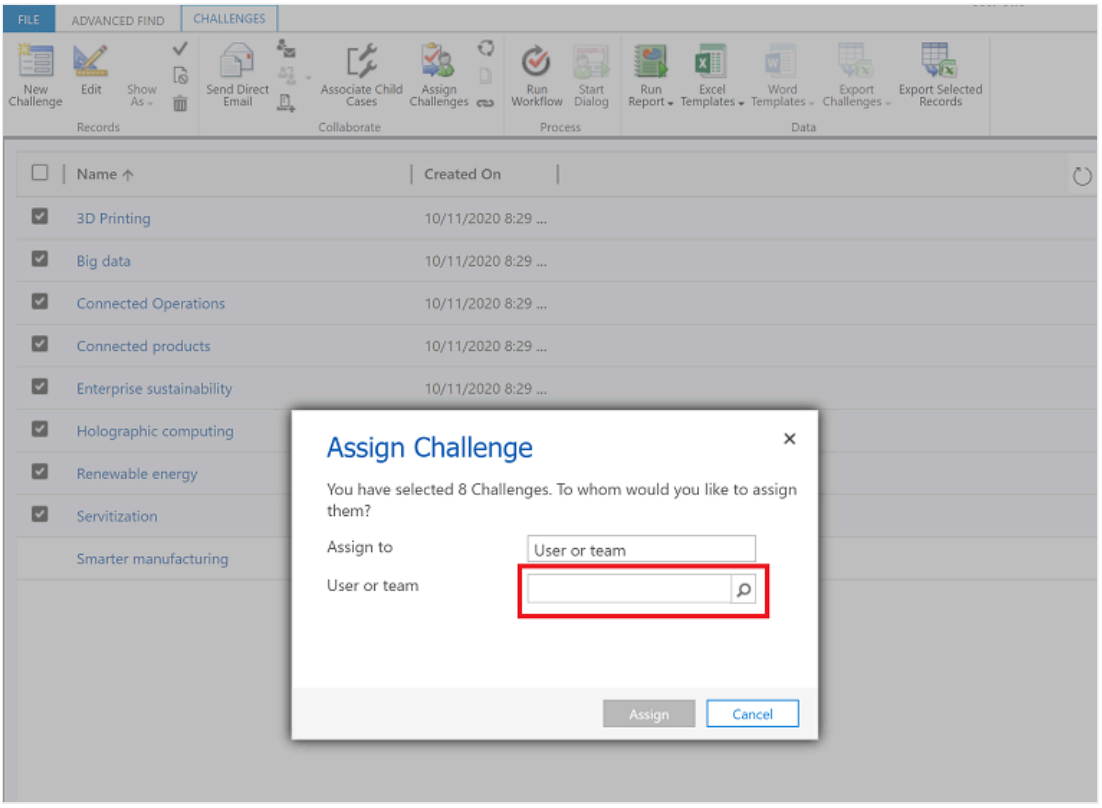
3. Select the rows that you want to assign and then select **Assign (page name)**



4. In the assign dialog box, select the **Assign to** column and choose one of the following options:

   - Select **Me** to assign the row to yourself and then select **Assign**. Remember only a system administrator can assign a row that belongs to someone else to themself.

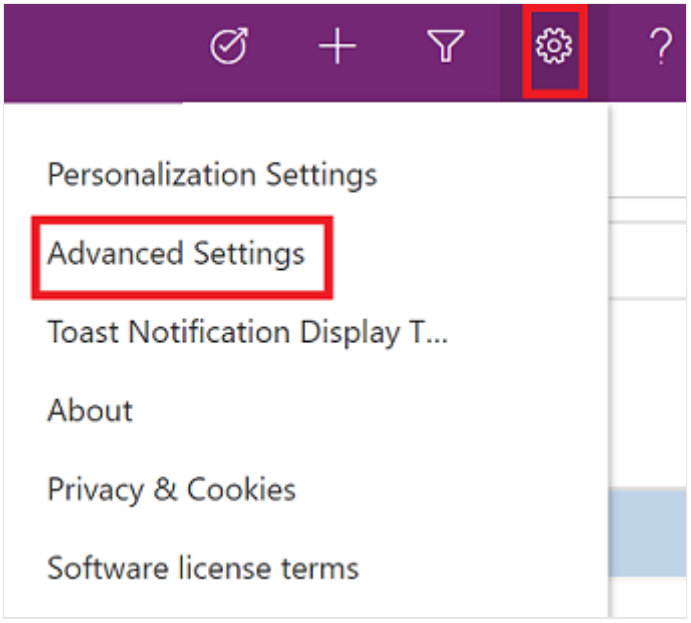
   - Select **User or Team** and then enter the name of the user or team or use the lookup to find them. Or, select **New Row** to create a new user or team row. When you're done select **Assign**.



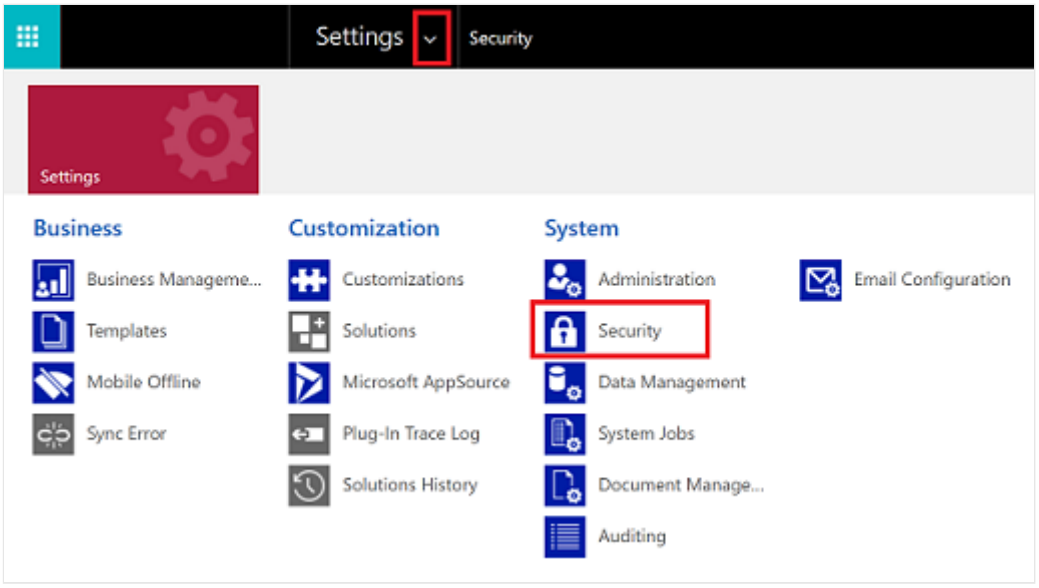
Reassign all rows (for admins)

When you have a user that leaves your organization or ownership needs to change from one user to another the rows can be reassigned by an administrator.

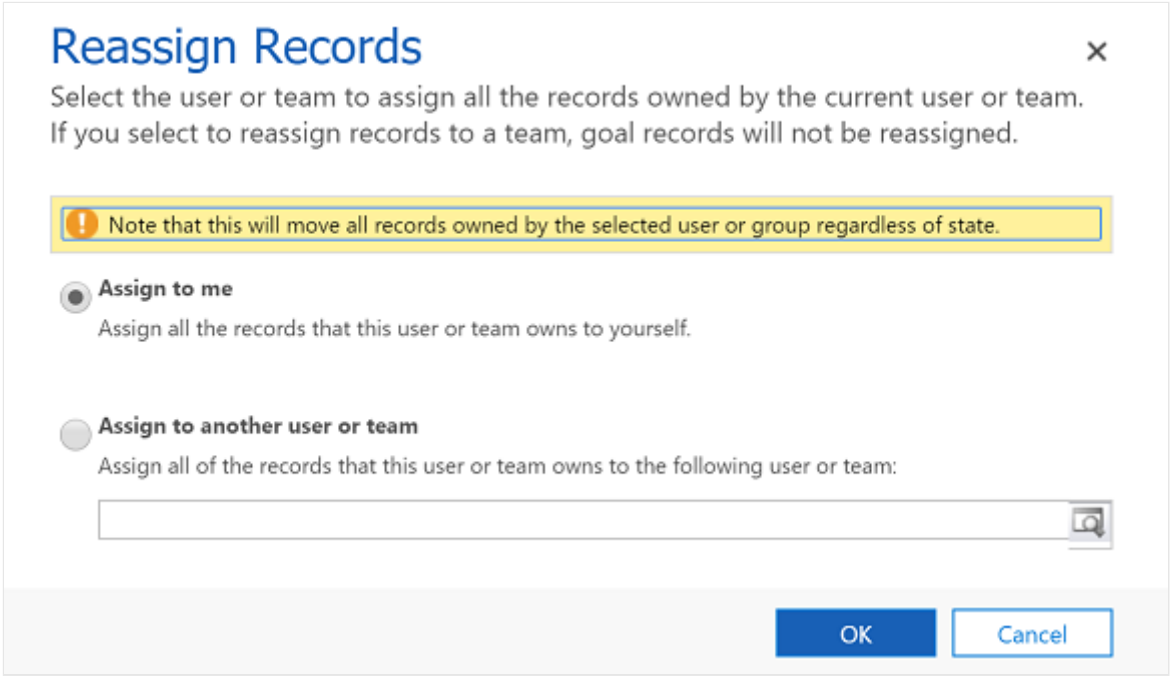
1. Go to **Settings** > **Advanced Settings**.



2. From the top menu navigate from **Settings** > **Security**.



3. Under **Security** select **Users**.
4. From the list of users, select a user name to open the user's profile.
5. On the command bar, select **REASSIGN ROWS**.
6. On the **Reassign Rows** dialog box choose how to want to reassign all the rows and then select **OK**.



> ⓘ Note
>
> - The **Reassign Rows** option will reassign all rows regardless of their status. Inactive and active rows will be reassigned to the other user or team. This will also deactivate all activated processes including business rules and workflows

when the row is reassigned to another user or team. The new owner must activate the processes that was deactivated when the row is reassigned.

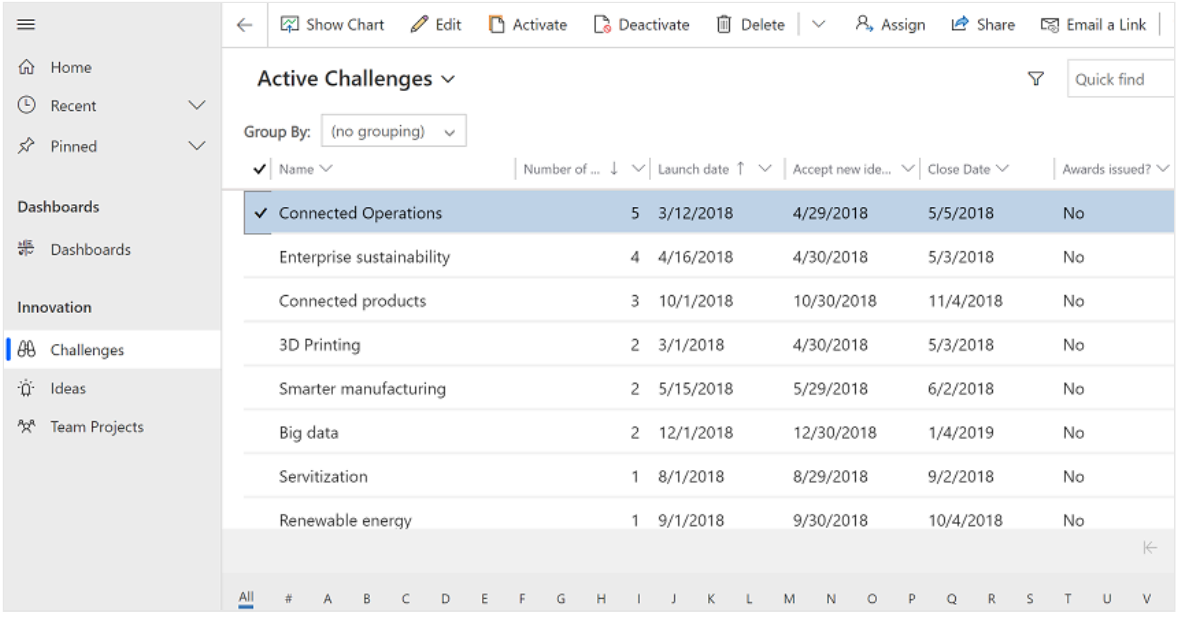
- When there is a large amount rows to reassign, the system may take a while to process.
- If there is an issue during the reassignment process such as the user that the rows are being reassigned to doesn't have the required privileges then the **Reassign Rows** process will stop. The rows that are processed before the issue will be updated and saved. For the rows that were not saved, you will need to reassign the rows again using the **Reassign Rows** option.

Share a row with someone else

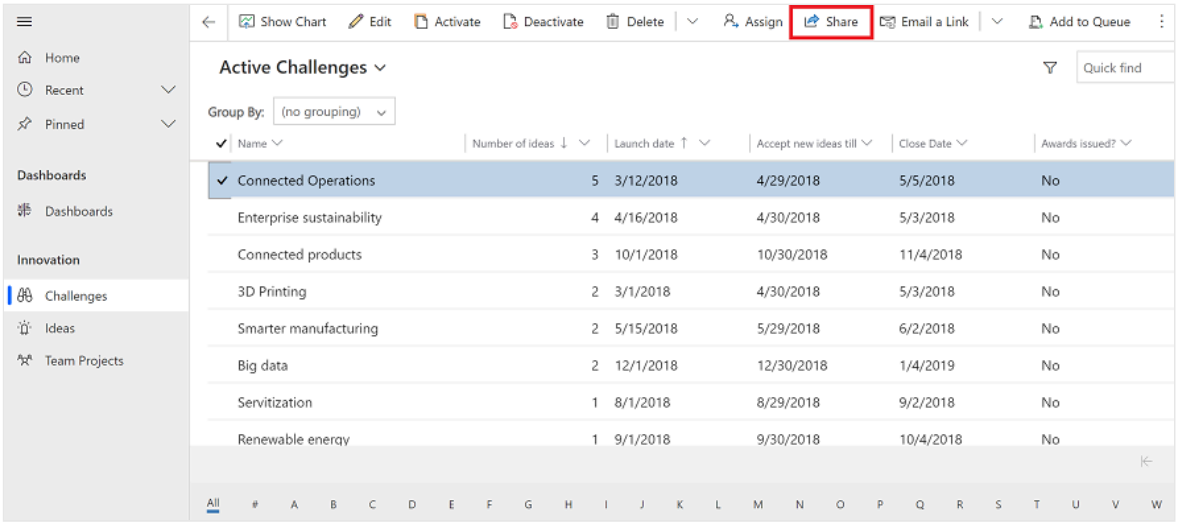
If you want to keep ownership of a row but let someone else work on the row with you then use the share option.

> **Note**
>
> For users on early access the sharing feature has been updated for 2021 release wave 2. For more information, see Share a row with someone else.

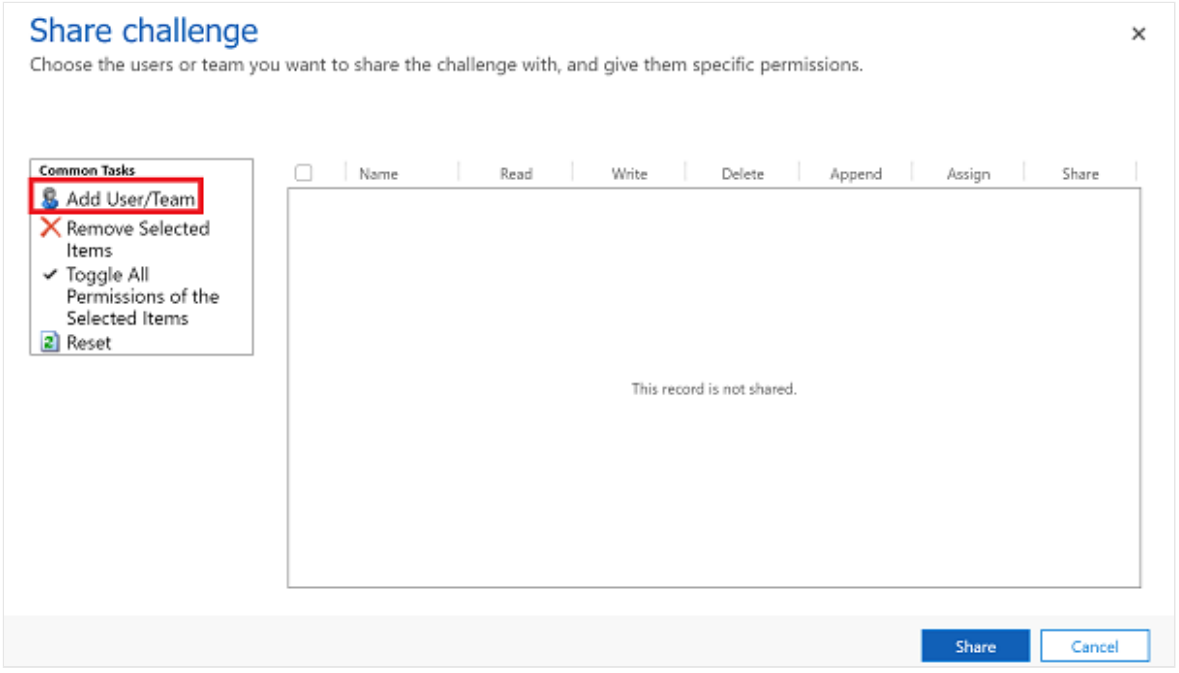
1. From a list of rows, select the row that you want to share with someone else. You can select multiple rows.



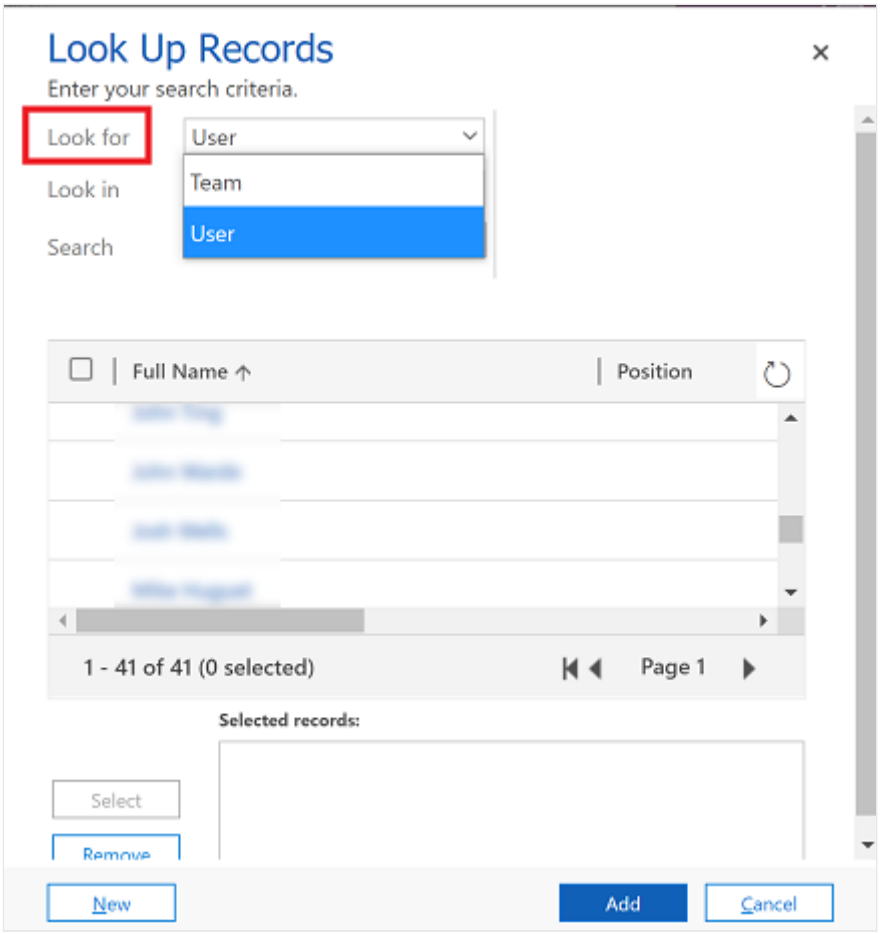
2. On the command bar, select **Share**.



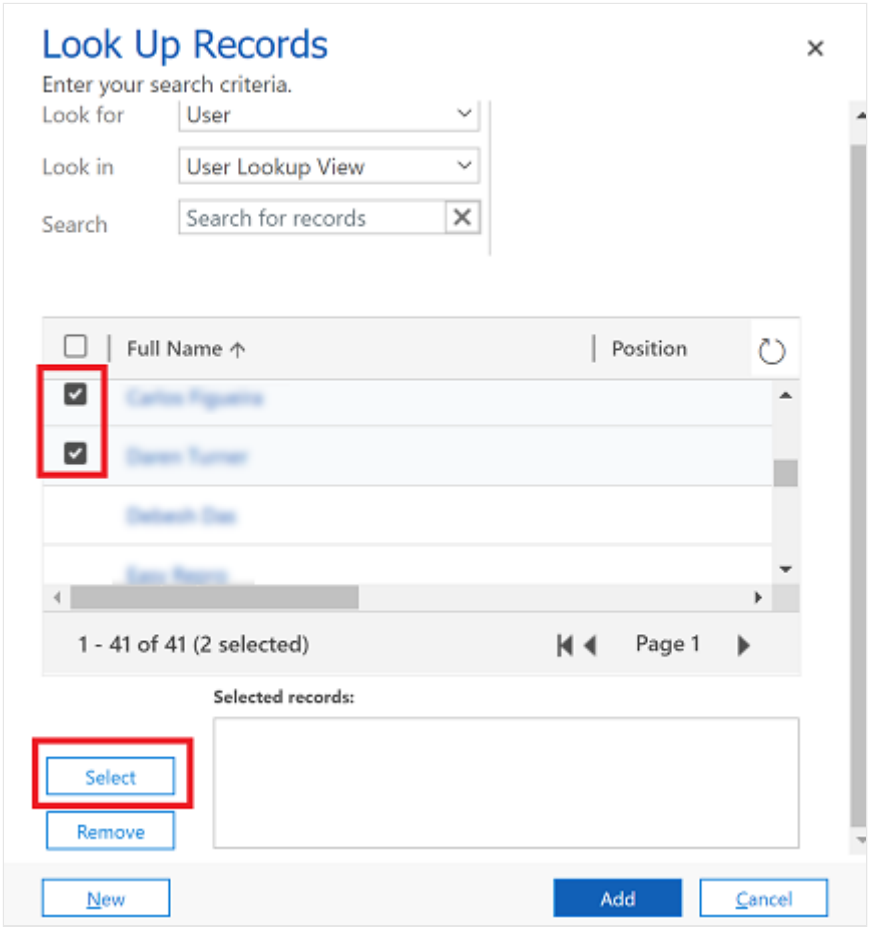
3. On the share dialog box, select **Add User/Team**.



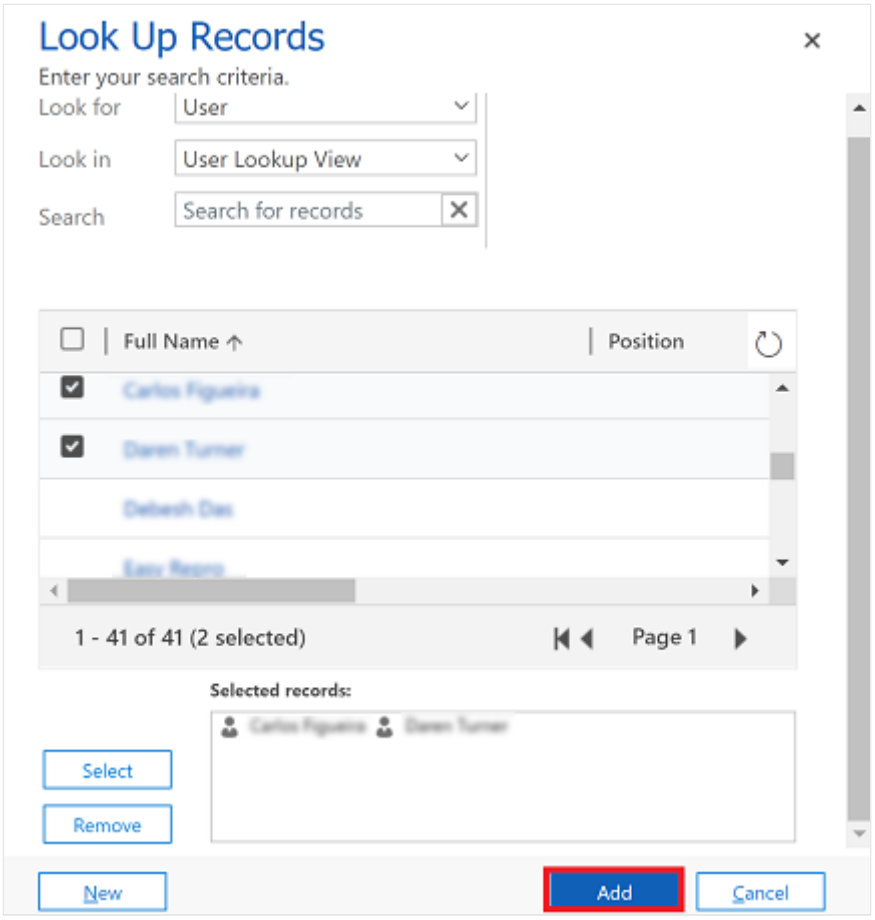
4. On the **Look up Rows** dialog box, select the **Look for** menu and choose **User** or **Team**.



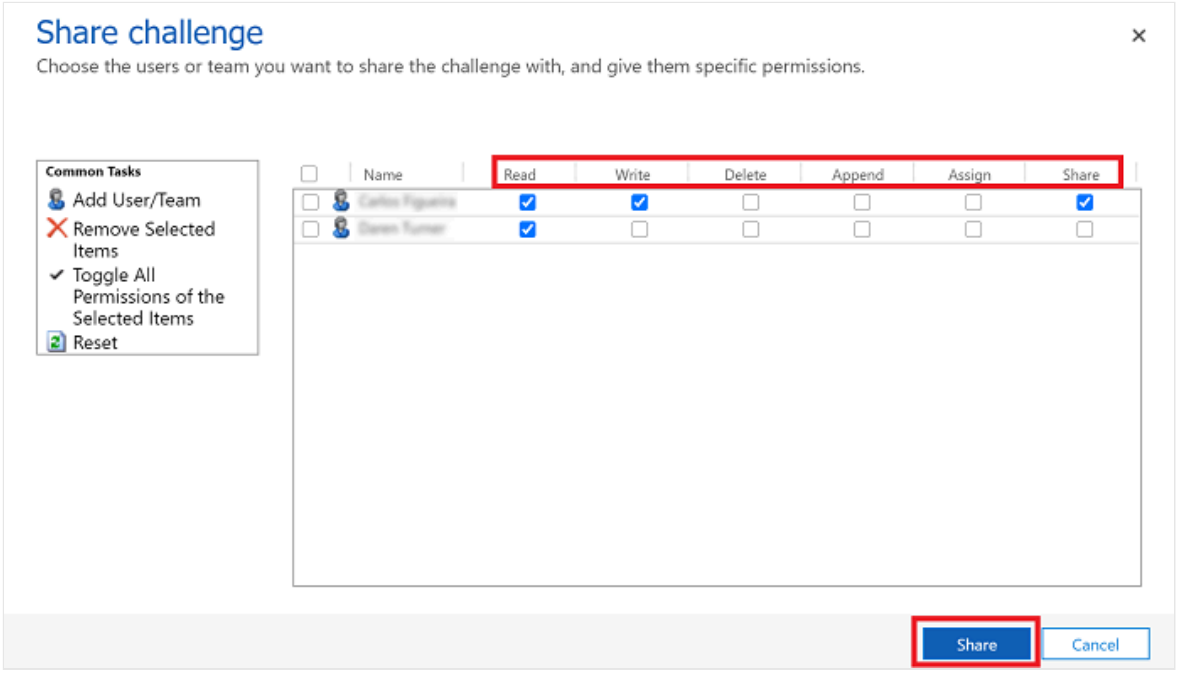
5. Use the lookup to find the user or team and select the name and then choose **Select**.



6. Select **Add**.



7. On the share dialog box, select the type of permissions the user or team has for the row. When you're done, select **Share**



Remove someone from a shared row

When you are the assigned owner of a row, you can remove another user the row is shared with.

1. From a list of rows that you own, select the row that you want to remove someone from the shared row.
2. On the command bar, select **Share**.
3. On the share dialog box, select the user or team who you want to remove sharing from.
4. Select the **Remove Selected items**.
5. Select the **Share** button.

Feedback

Was this page helpful? Yes No

Provide product feedback

Share rows with a user or team

Article • 02/15/2022

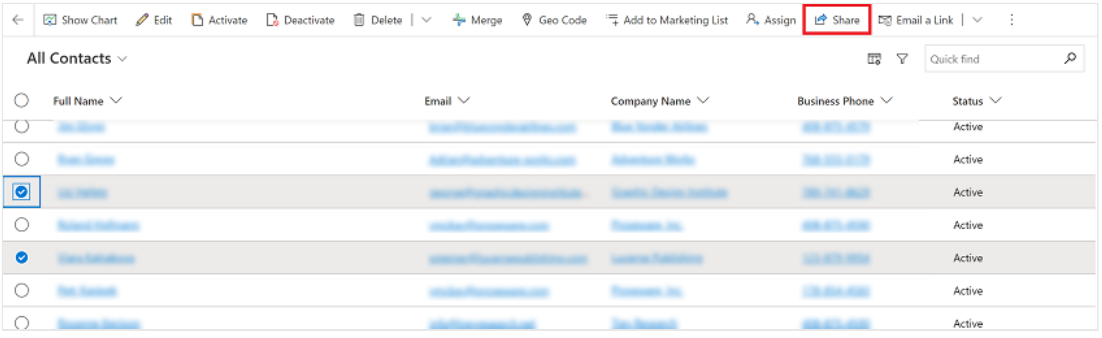
Keep ownership of a row and use the share feature to share the row with another user or team.

> **Note**
>
> - This feature isn't supported on Power Apps mobile or when you're using the app in offline mode with no internet connection.
> - Any user that you share a row with needs to have basic level access that is granted by a system administrator. If you cannot select permissions such as read or write then contact your system administrator and check if the user or team has the required access.

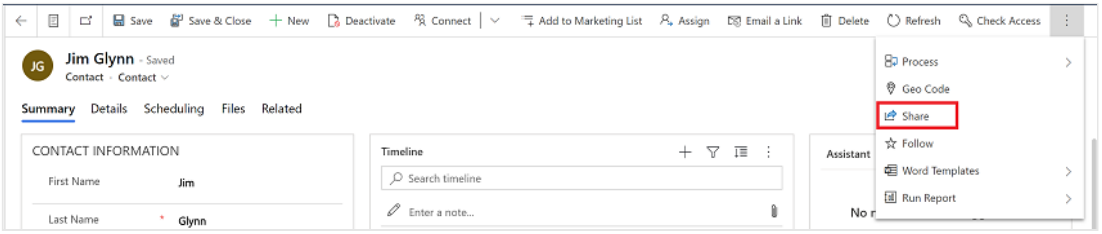
Share rows

1. Share a row from a view page or when you open a row.

   - **Share from the view page**: Select one or more rows on a view page and then select **Share** on the command bar.



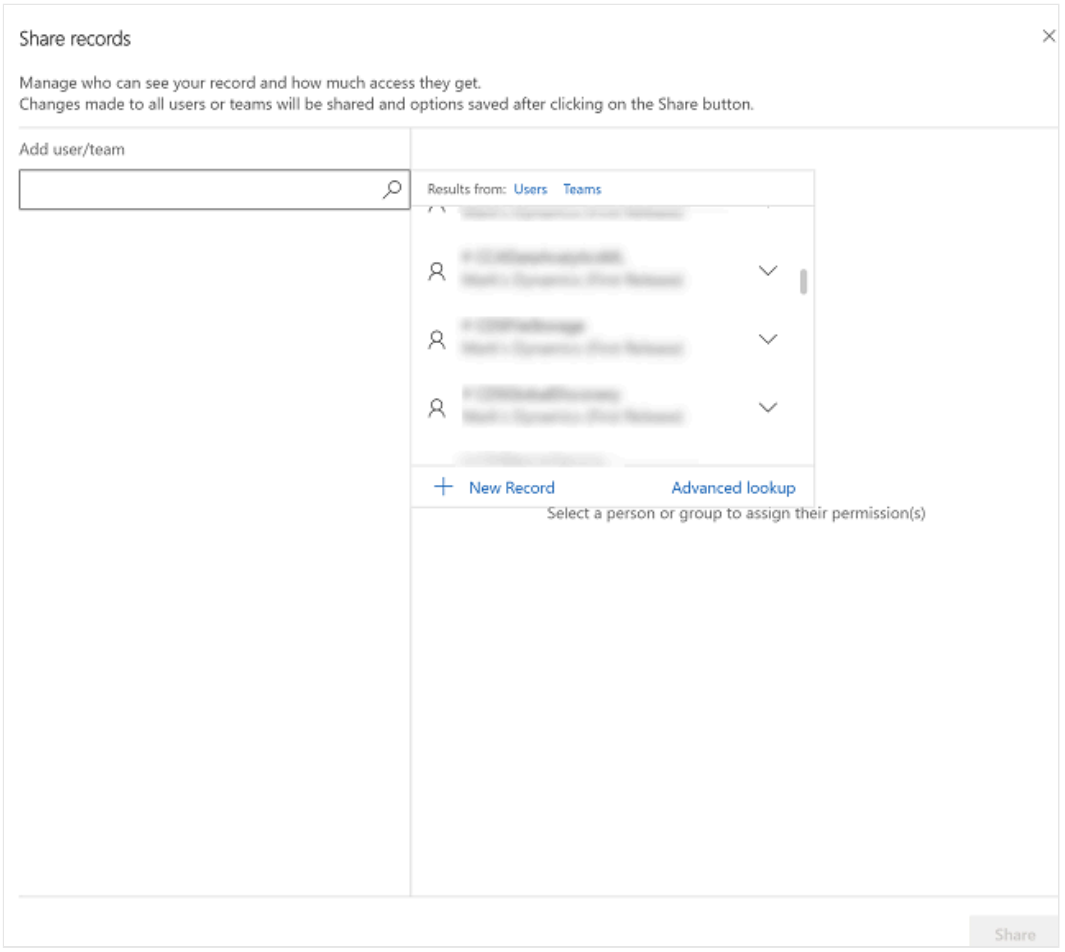
   - **Open a row and share**: Open a row and then select **Share** on the command bar. If you don't see the share option, then select the **More commands** ellipsis and then select **Share**.



2. On the share dialog box, do one of the following:

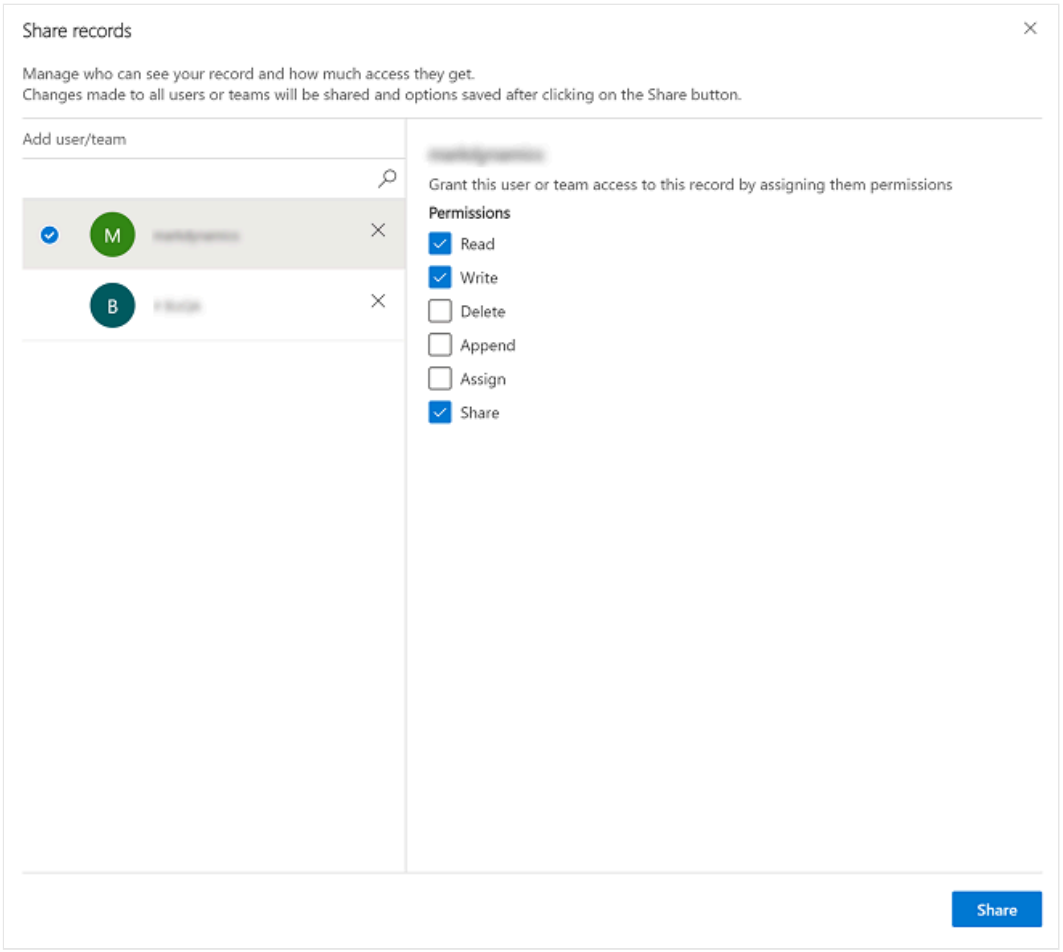
   - **First time sharing a row**

     a. Select the lookup column under **Add user/team**. Enter the name of the user or team in the lookup column. The lookup will start displaying users or teams based on what you typed, or you can select the lookup icon (magnifying glass) and an alphabetical list of names appears that you can select to assign access. You can also search for a user that previously had share rights to manage or change access privileges.



     b. Select a user or team one at a time, and then grant them the appropriate permissions to the row.

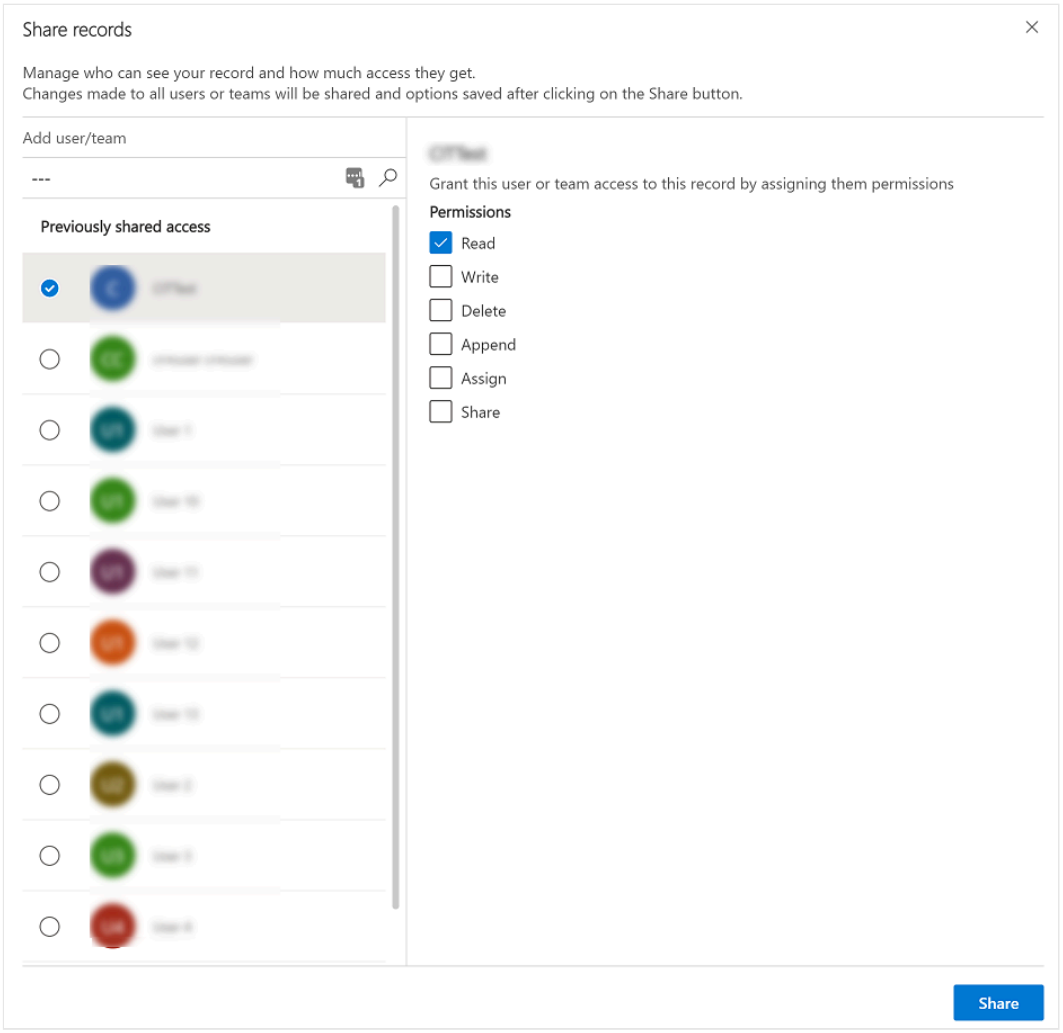
        If check boxes are disabled it means a user or team does not have basic privileges to the row. For more information on user access, see Check your user access to a row.



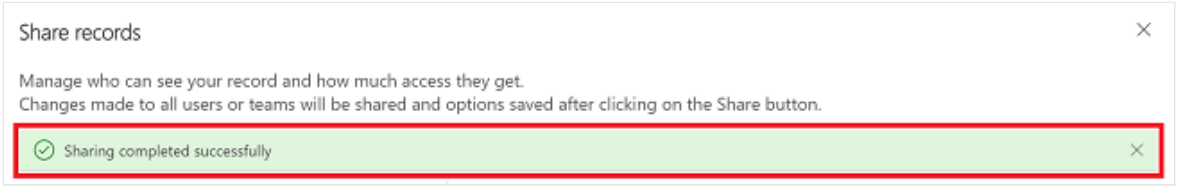
- **Previously shared access to a row**

  a. Under **Previously shared access** select a user or team one at a time and then grant them the appropriate permissions to the row.

  If check boxes are disabled it means a user or team does not have basic privileges to the row. For more information on user access, see Check your user access to a row.



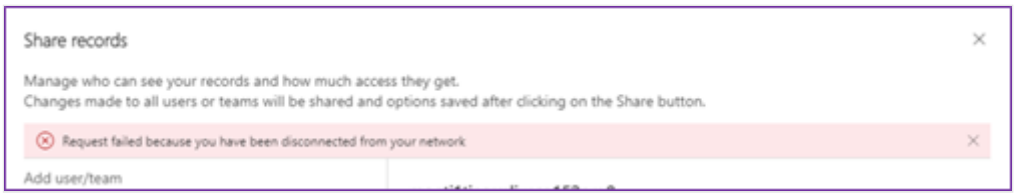
3. Select **Share** to save your changes.



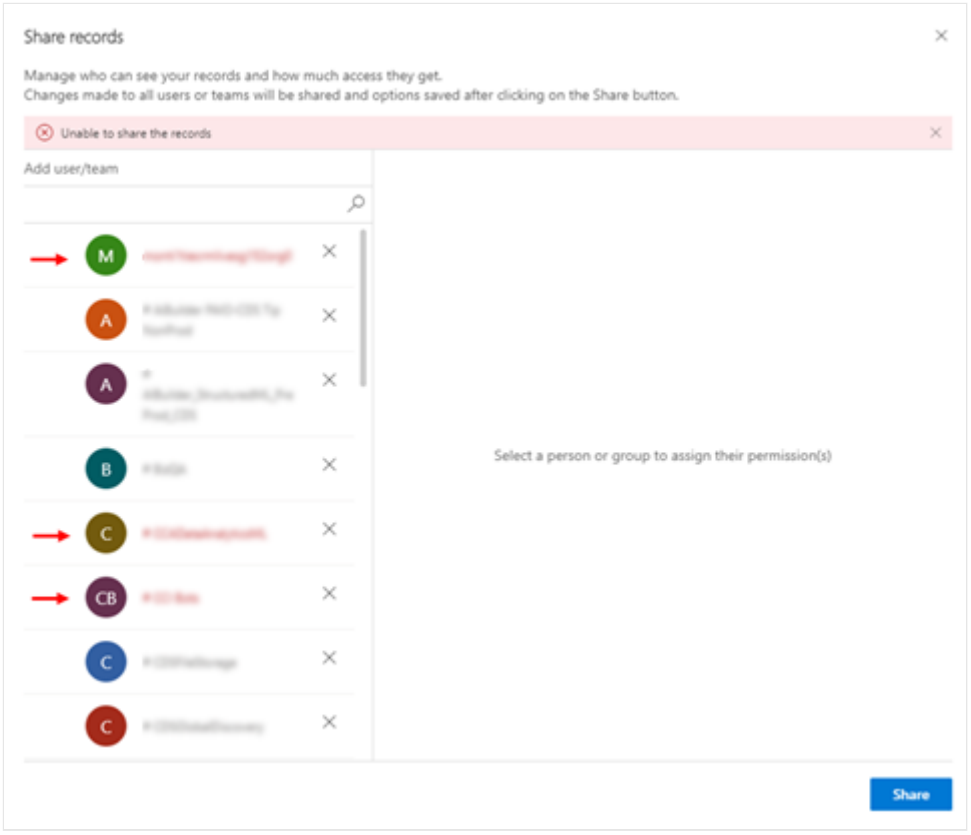
> **Note**
>
> If you add additional users or teams after saving share access and the check box is disabled but has a value, this means that your administrator has changed the basic privileges for the user, and they no longer have basic access rights, however, the share permissions is retained and isn't reset when the administrator changes the base privileges. Contact your system administrator to update a user's access if you want to change or update the disabled options for sharing a row.

Issues and errors

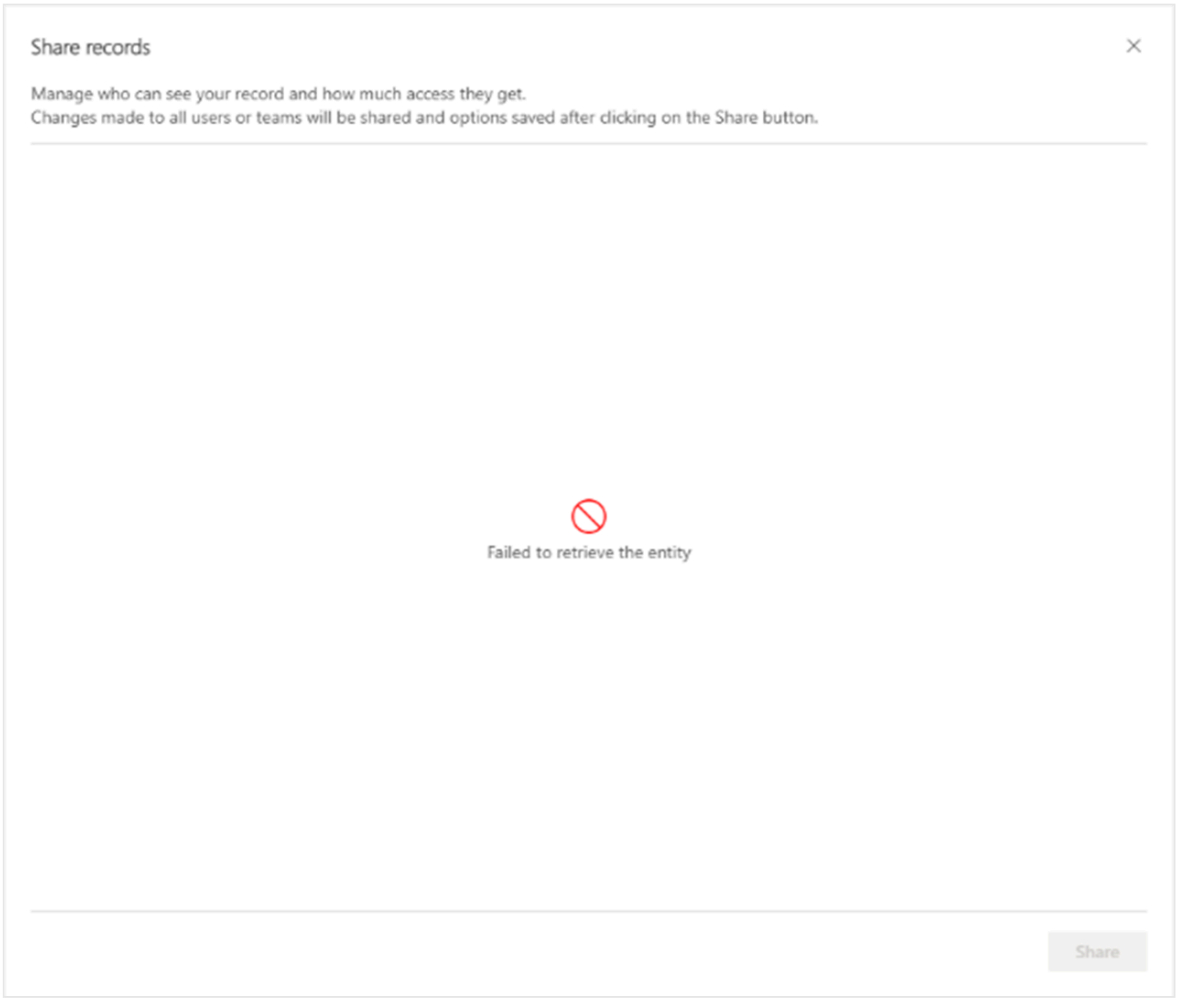
1. If you're disconnected from the internet when sharing a row, then you won't be able to set or update options for users or teams. Sharing won't be available.

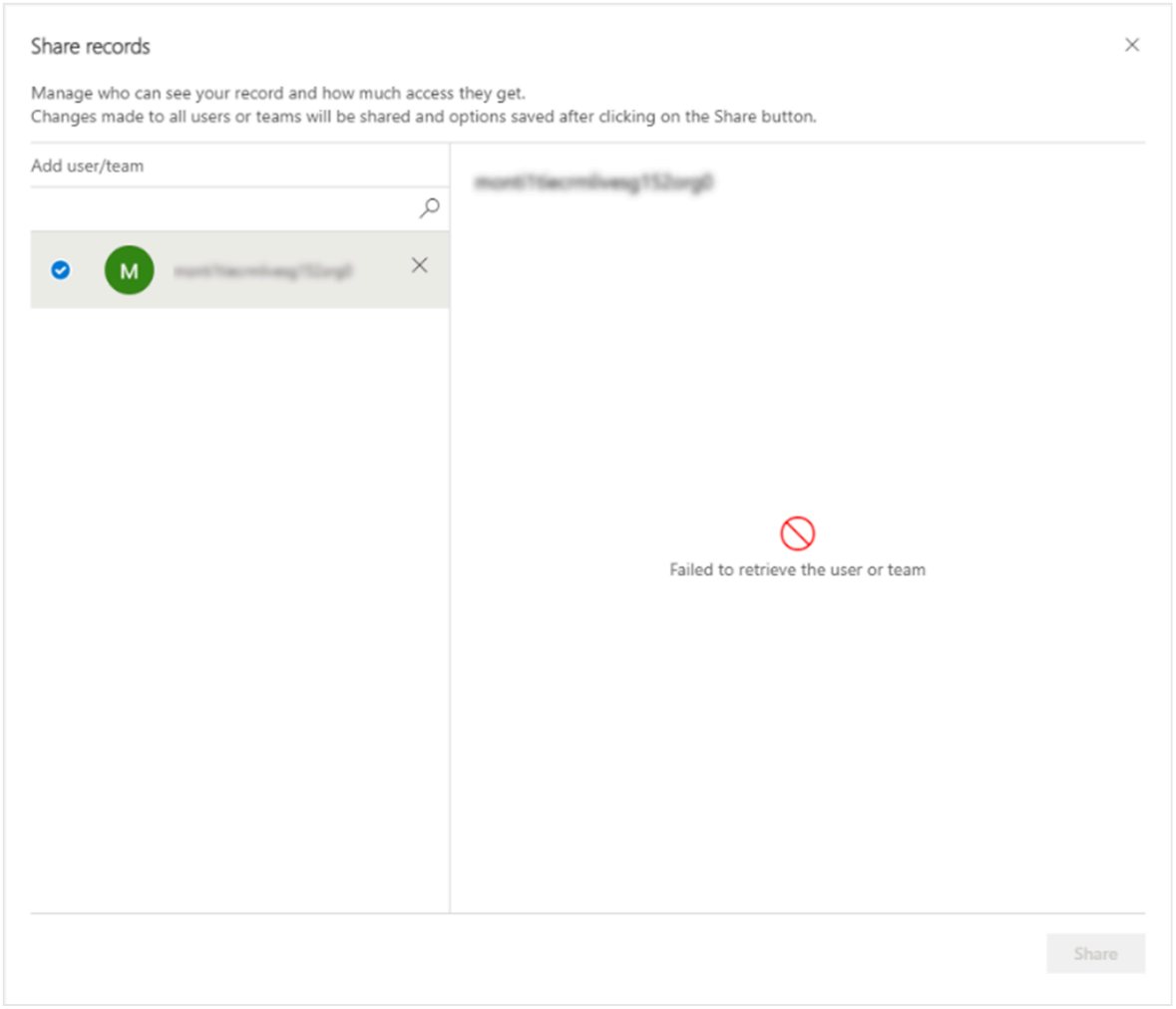


2. If you get a generic error while sharing, this means that one or more of the rows can't be shared. The users will be hi-lighted in red. Select **Share** in the dialog to attempt another share with those users or close the dialog and retry. This usually happens if you are sharing lots of rows at once.



3. You may see an error when the system is unable to retrieve the users to assign share rights or sharing fails due to access related issues.


Share records
Manage who can see your record and how much access they get.
Changes made to all users or teams will be shared and options saved after clicking on the Share button.
Failed to retrieve the entity
Share


Share records
Manage who can see your record and how much access they get.
Changes made to all users or teams will be shared and options saved after clicking on the Share button.
Add user/team
M
Failed to retrieve the user or team
Share

Feedback

Was this page helpful? Yes No

Provide product feedback

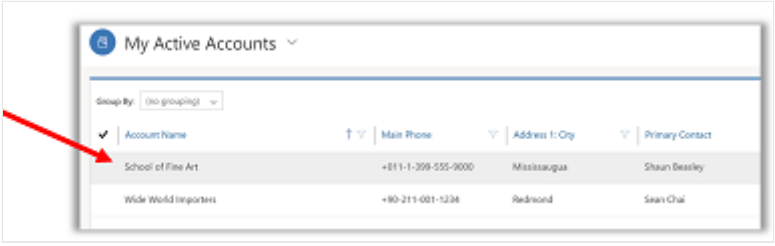
Share rows using Access Team

Article • 08/09/2023

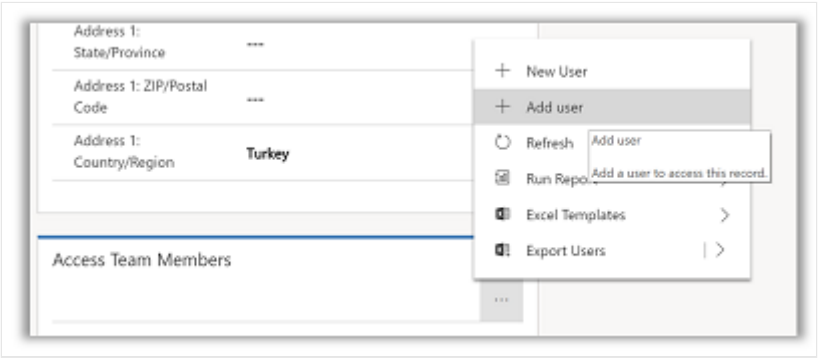
Access Team grant access to rows via sharing rows. Access Team members have privileges defined by their individual security roles and by roles from the teams they are members of.

> **Note**
>
> Before you can share rows using Access Team, your admin will need to set-up an Access Team template. For more information, see About team templates.

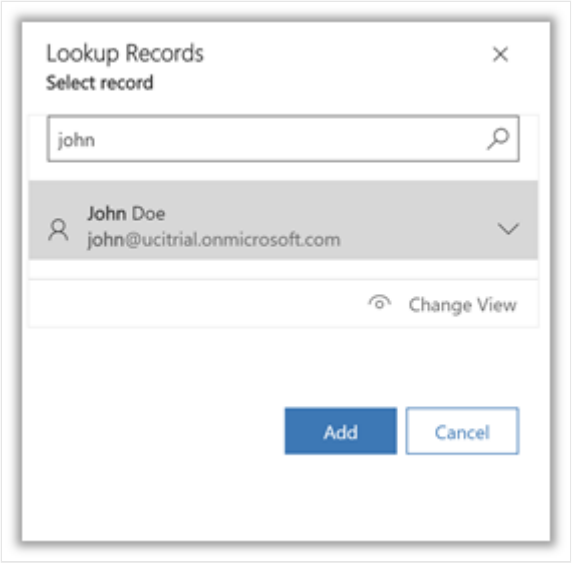
1. To give a user permission to access a row, from the site map, select the row type. For example, **Accounts**.
2. From the list of rows, open the row that you want to provide another user access to.



3. In the **Access Team Members** section, select **More Commands** (...) > **Add User**.



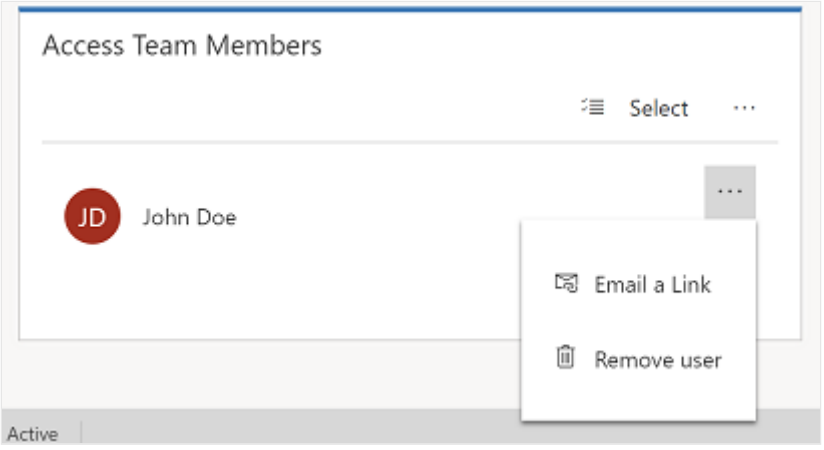
4. In the search box, type in the user name to find the user and then select **Add**.



Remove a user from Access Teams

You can easily remove a user's access to a row just as easily as you added them.

1. Open the row that you want to remove the user from.
2. Under **Access Team Members** subgrid, select **Remove user**.



Feedback

Was this page helpful? Yes No

Provide product feedback

Check user access to a row

To perform an action on a row, a user needs to have the required privilege assigned through a security role or the user must be a member of a team that has a security role with assigned privileges.

To perform actions on a row, the system checks your privileges and if the privilege check passes, then the system performs an access check. The access check verifies that you have the required rights to perform actions on a row such as read, write, create, delete, append, append to, share, and assign.

You can have access rights to perform an action on a row through ownership, role access, shared access, or hierarchy access.

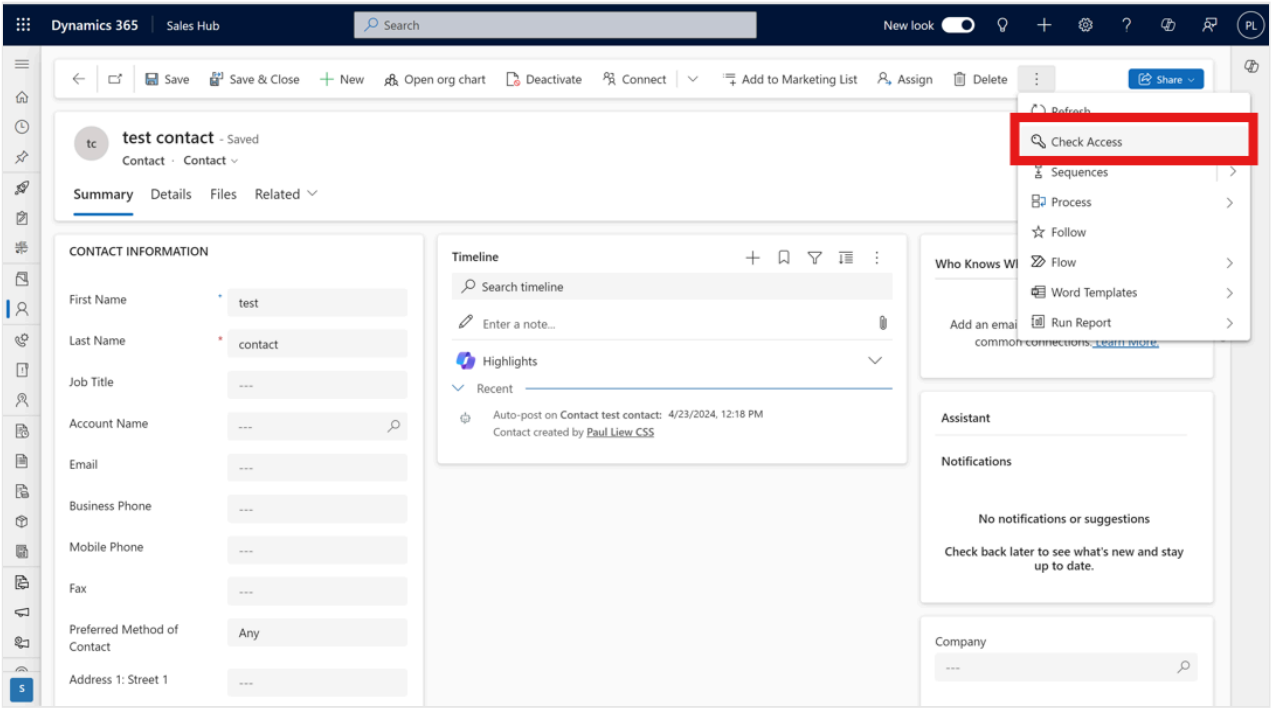
Expand table

Access type	Description
Ownership	User owns the row or belongs to a team that owns the row.
Role access	User has access to perform an action on a row because of their security role.
Shared access	The row is shared with a user, team, or organization by a user that has appropriate share rights.
Hierarchy access	Hierarchy access only takes place if hierarchy security management is turned on for the organization and the table. The user also needs to be a manager.

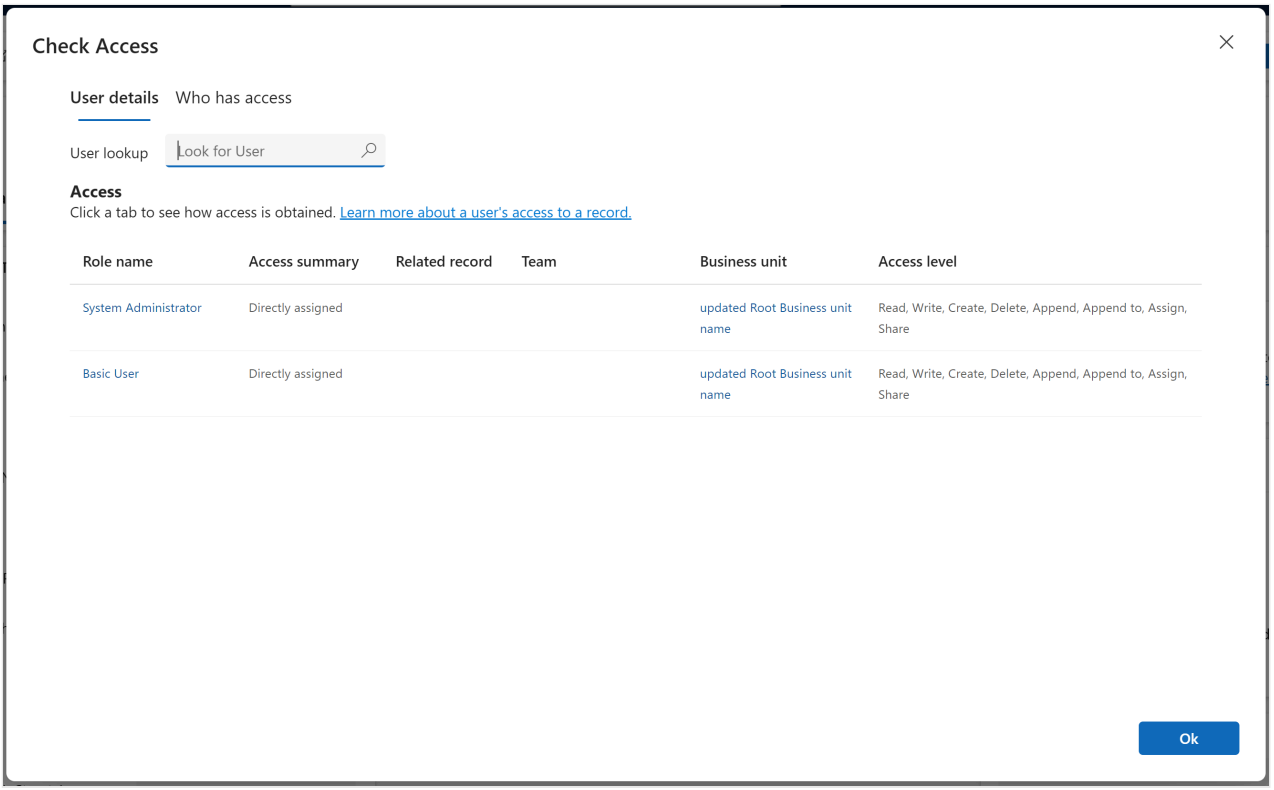
For more information on how privileges and access checks work, go to How access to a row is determined.

Check your access to a row

1. Select a row.
2. Select **Check Access** on the command bar.



3. The **Check Access** window displays your access information.



Contact your administrator if you don't have the required access. Only an administrator can edit your security role and privileges. To find your administrator, go to Find your administrator or support person.

Check another user's access to a row

If you're an administrator, you can check the access another user has to a row.

1. Select a row.
2. Select **Check Access** on the command bar.
3. In the **User Lookup** field, select or enter a user name to search for the user.
4. The **Check Access** window displays the user's access information.

Check all users who have access to a row

The admin, owner of the record, or users who have access to a row can share the row with other users for collaboration.

To see who has access to a row and their respective access level, complete the following steps.

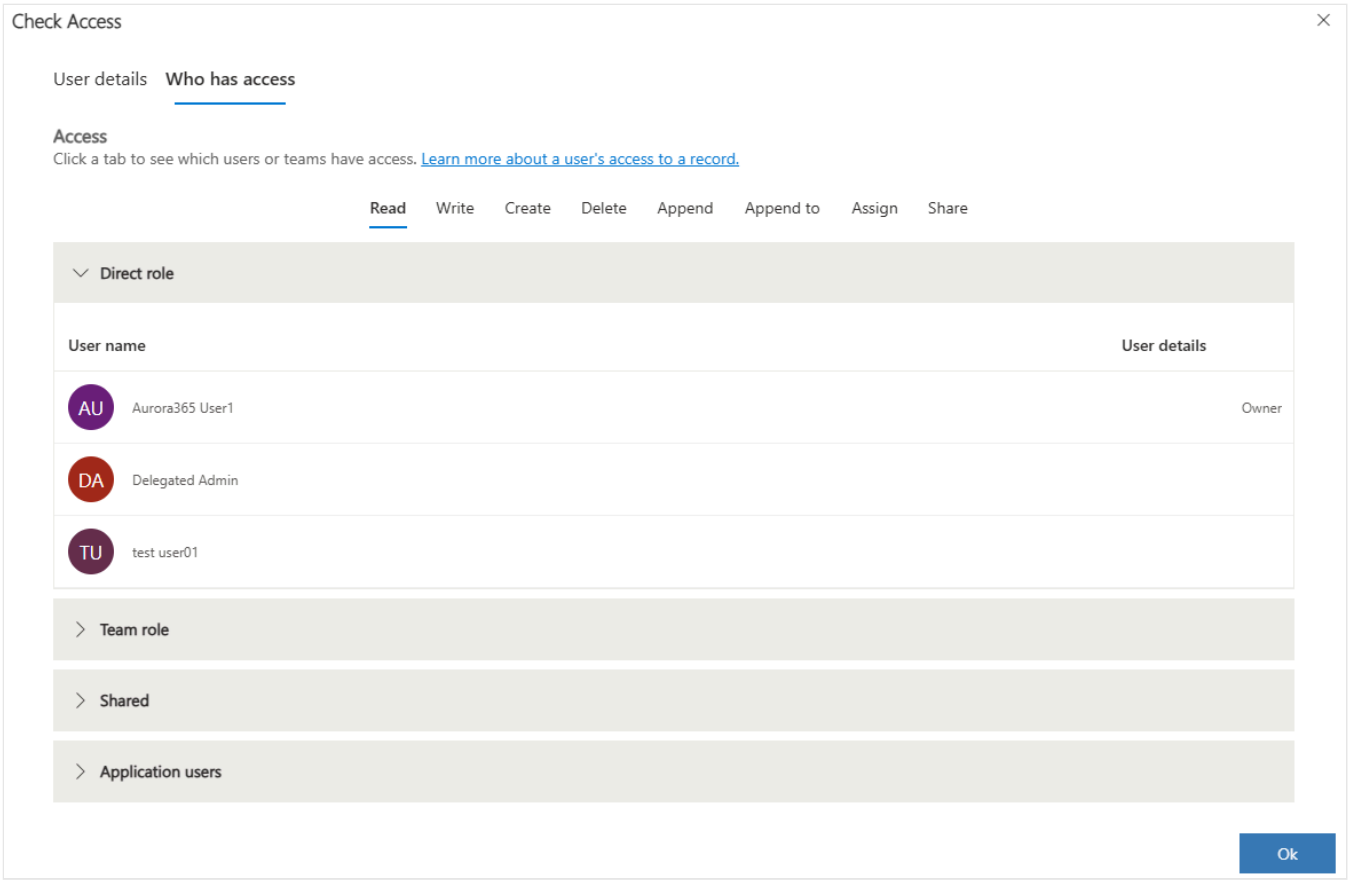
> ⓘ **Note**
>
> There are two database settings for each environment that enable the **Who has access** feature. System adminstrators can Install the OrganizationSettingsEditor tool and set the following options to true:
>
> - **IsAccessCheckerAllUsersEnabled**: This allows the admin to see who has access to the row.
> - **IsAccessCheckerNonAdminAllUsersEnabled**: This allows the admin, owner of the record, and users who have access to the row to see who has access.

1. Select a row.
2. Select **Check Access** on the command bar.
3. In the **Check Access** window, select **Who has access**.

   > ⓘ **Note**
   >
   > The manager list from heirarchy and position security is not shown under **Who has access** as this list can be long.
   >
   > If you select a user who doesn't have access to the record, the page returns with a blank screen.

Check who has access results

The user's record privileges are displayed. You can tab to each of these privileges to see who has this privilege on the record.



A user can have access to a record through one or more of these categories.

- **Direct role** - The record access is granted to a security role that is assigned to the user. Learn more at Role access.
- **Team role** - The record access is granted to a security role that is assigned to the team, and the user is a member of the team. Learn more at Role access.
- **Shared** - The record access is shared with the user or team who the user is a member of. Learn more at Shared access.
- **Application users** - Dataverse's application users who have access to the record because they're assigned to a security role that has privilege to the record.

Last updated on 01/08/2026

Add a connection role to link rows to each other

Article • 08/09/2023

Connections enable you to easily associate users, contacts, quotes, sales orders, and many other table rows with each other. The rows in the association can be assigned particular roles that help define the purpose of the relationship.

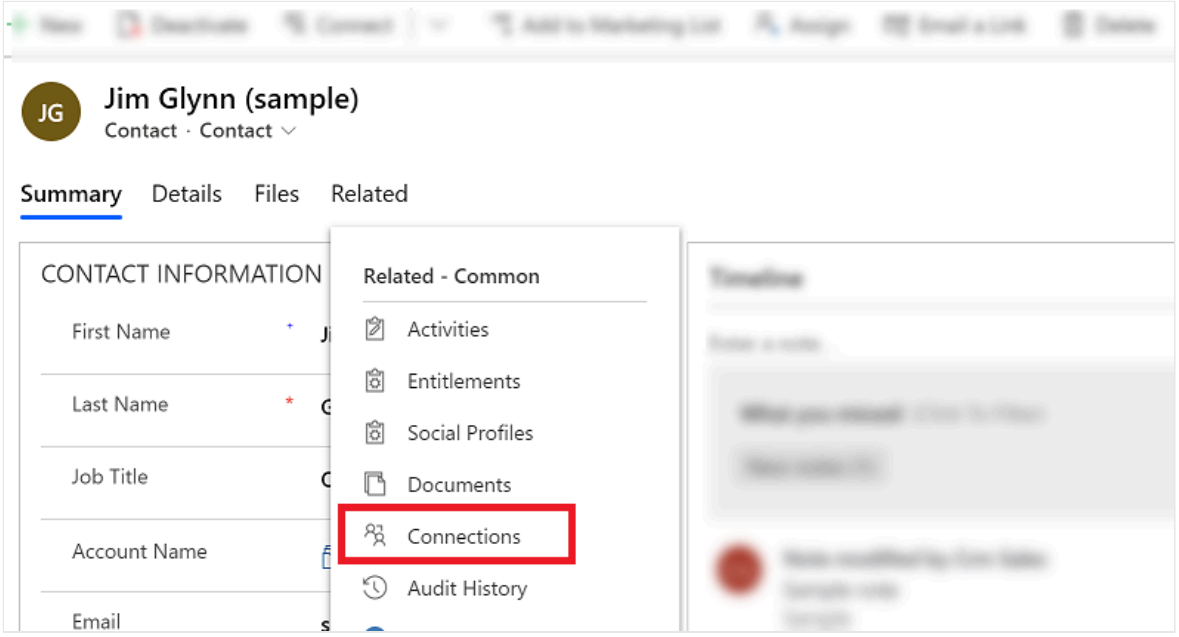
It's a quick way to create multiple connections and roles for a particular row. For example, a contact may have many relationships with other contacts, accounts, or contracts. In each relationship, a contact may play a different role.

Connection roles are directly associated to a connection. To use a connection role, you must first add a connection to your row.

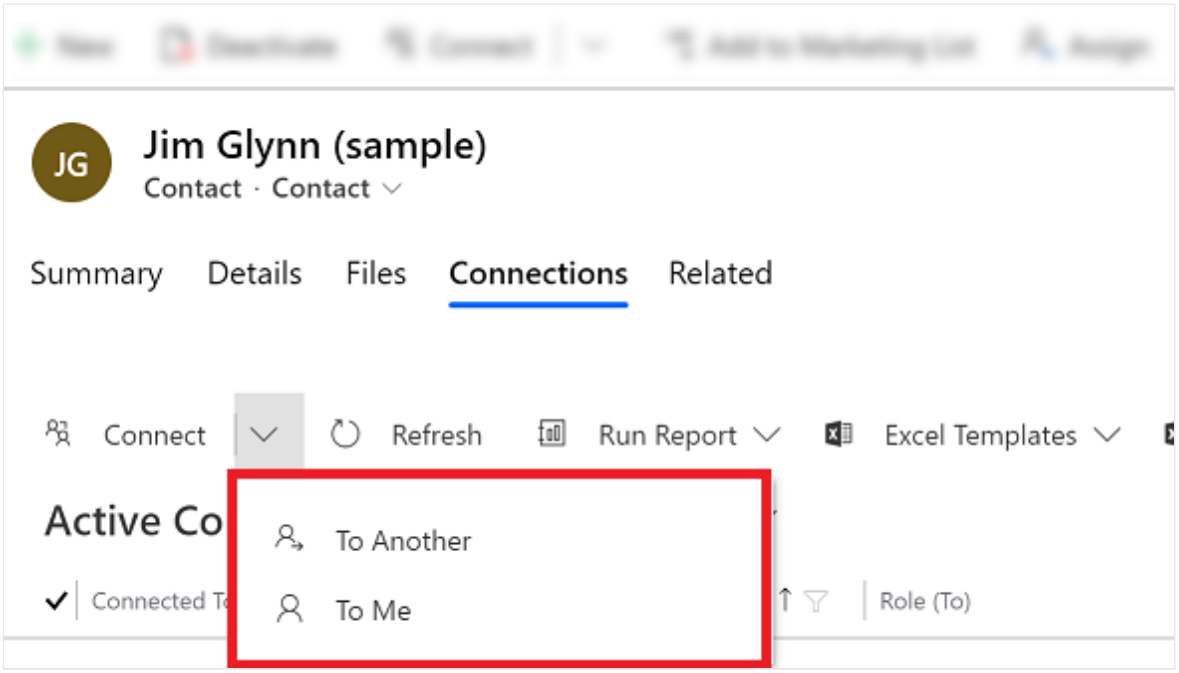
Before you can add connection roles, it needs to be enabled by your admin. For more information see, Configure connection roles.

1. To add or manage connections, select the row you want to manage like an opportunity.

2. Select the **Related** tab and then select **Connection**. This opens the connection grid with the list of connections for the row.

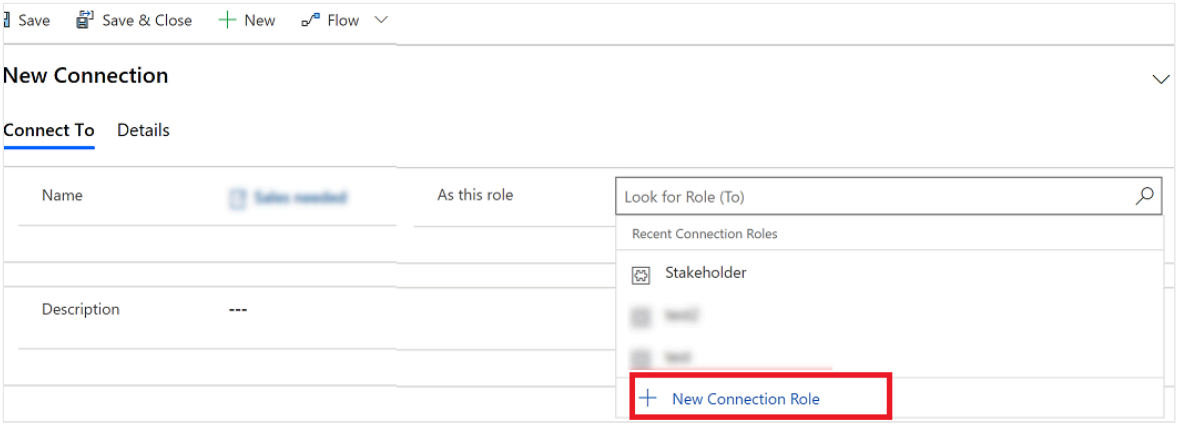


3. Select **Connect** and then choose **To Another** or **To Me**.



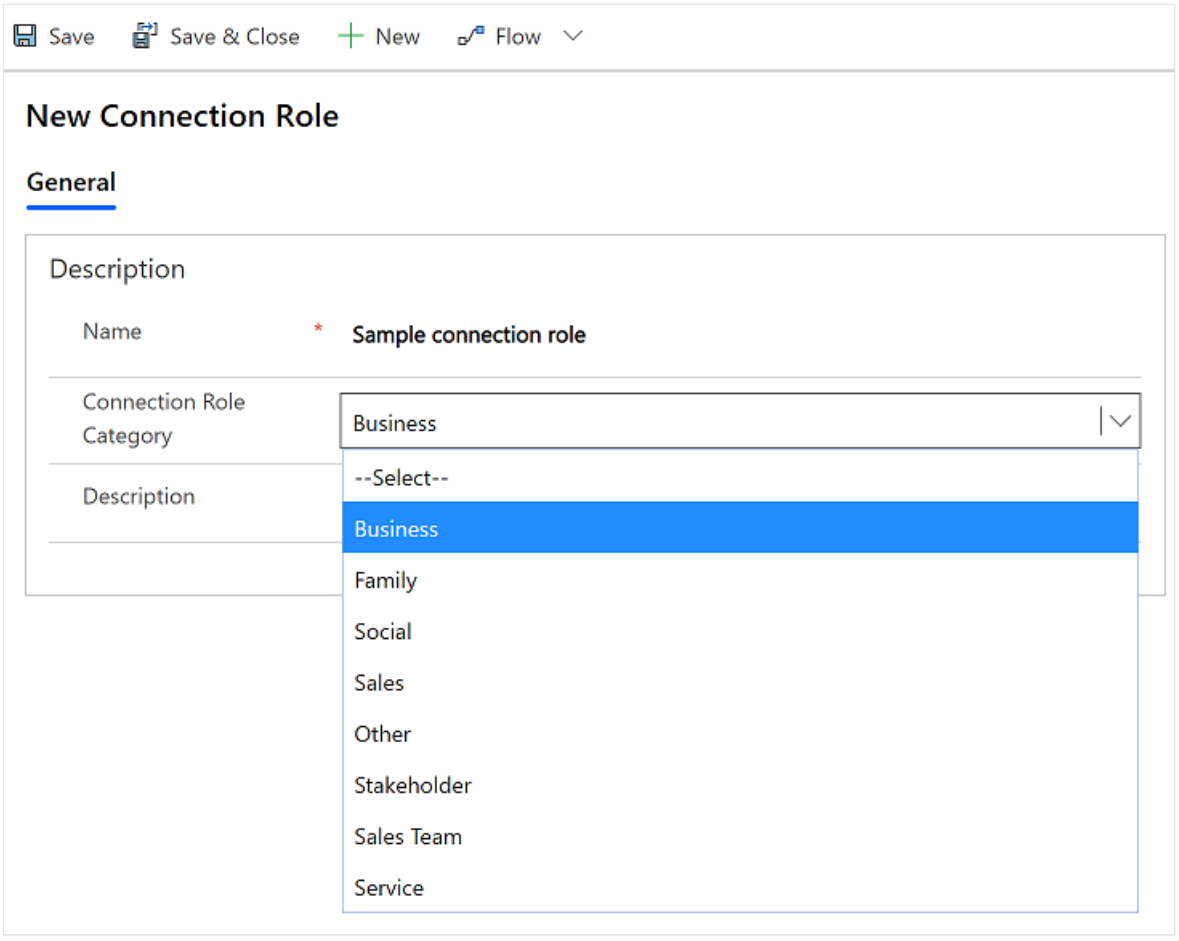
4. On the **Name** column, enter or find the name of the row for the connection.

5. On the **As this Role** column, select the lookup icon and then choose **New Connection Role**. Or, use the search to find an existing role that you want to associate to the connection and then select **Save**.



> **Note**
>
> If you have entered information before creating a new connection role, a warning dialog will be displayed asking if you would like to cancel and continue working on the connection or to go ahead and leave the current row you are working on.

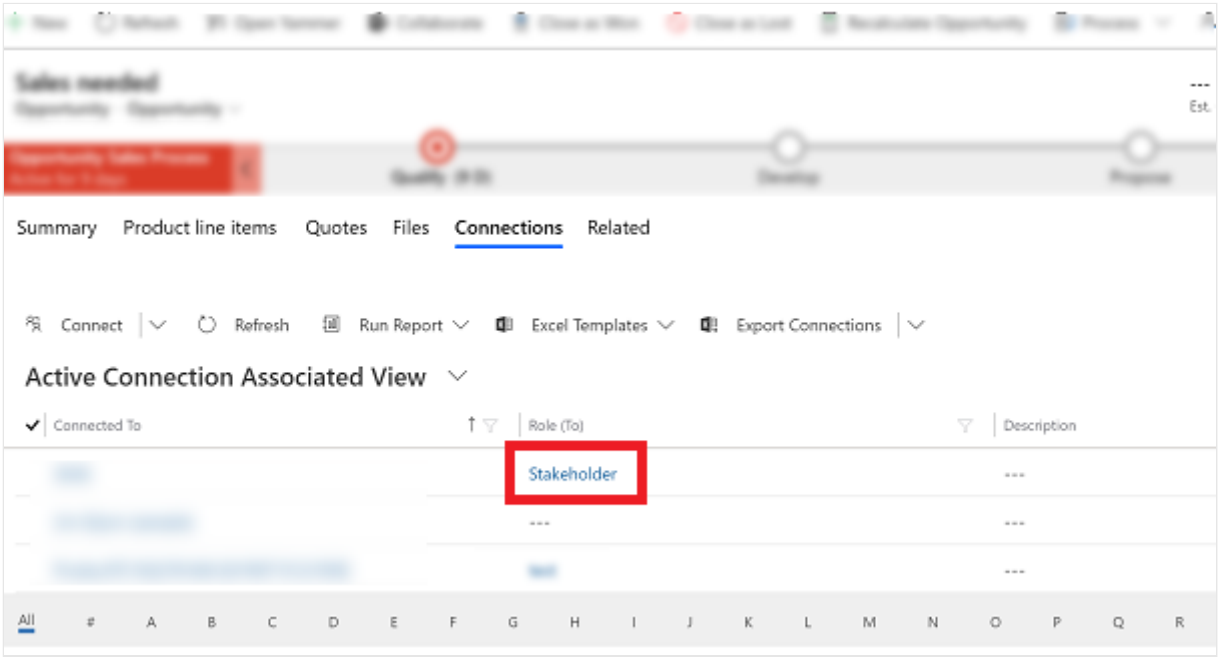
6. To create a new connection role, on the **New Connection Role** screen enter a name and then choose a **Connection Role Category**.



7. When you are done, select **Save & Close**.

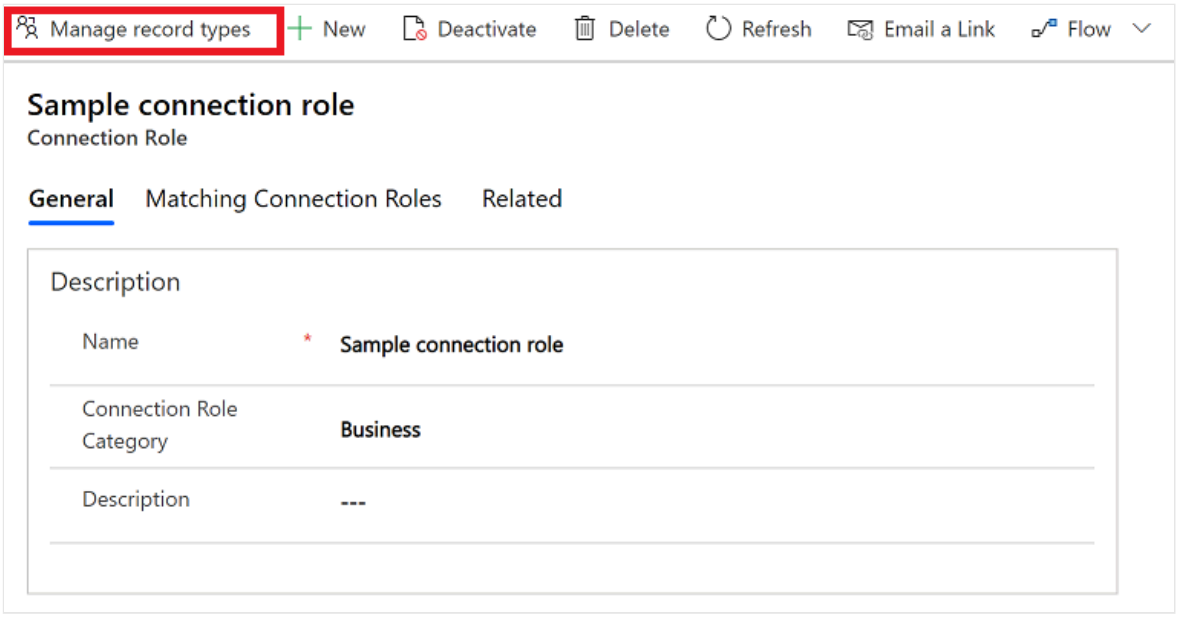
Manage Connection Roles

To manage a connection role, select the connection role from a connection table. This opens the connection role table row. You can edit the name, select a connection role category, and add a description.

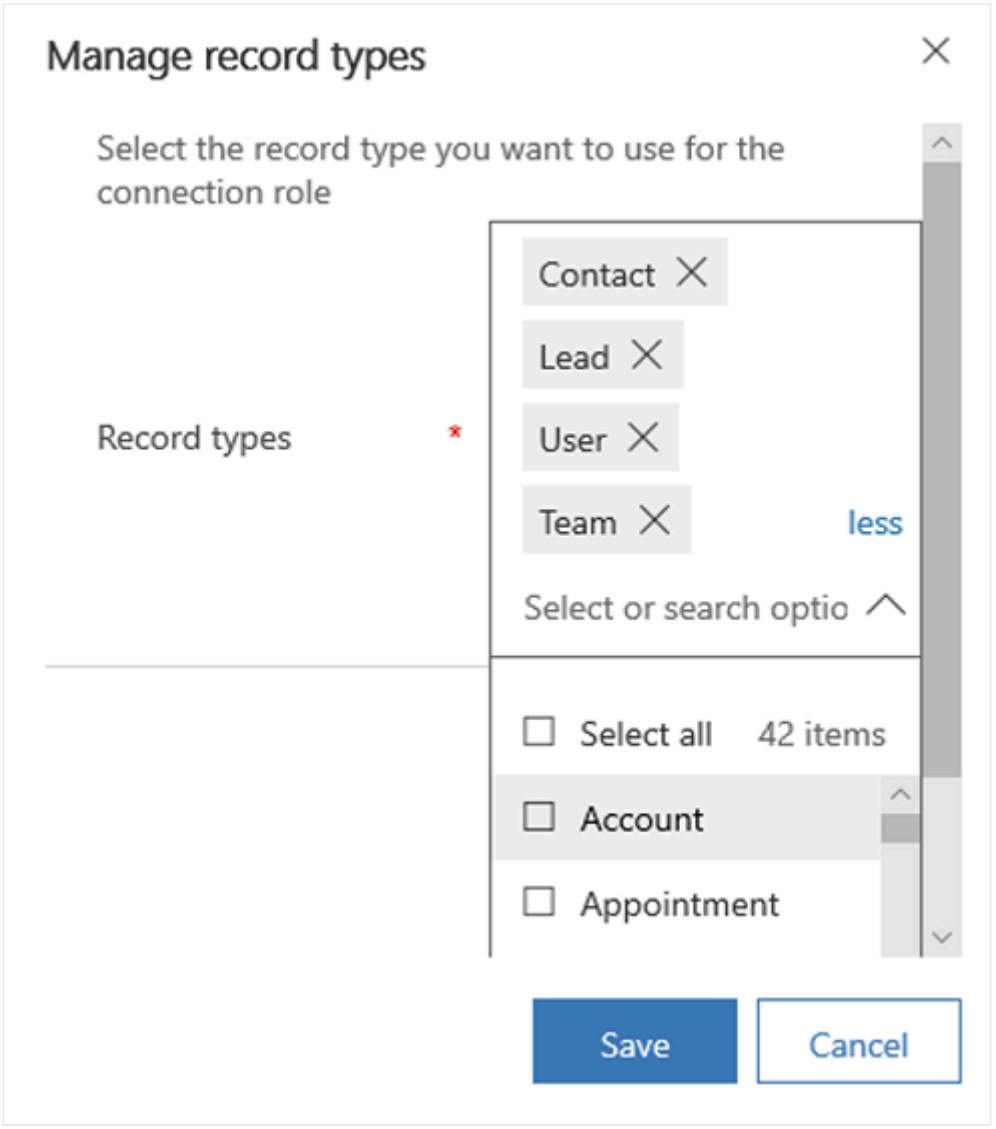


You can also manage the connection role types that you want to associate with the connection role.

1. Open the connection role and then select **Manage Row Type** on the command.



2. This opens a list of connection role types that you can add or remove for this connection role.



Feedback

Was this page helpful? Yes No

Provide product feedback

Deactivate or activate an account or contact

Article • 12/10/2024

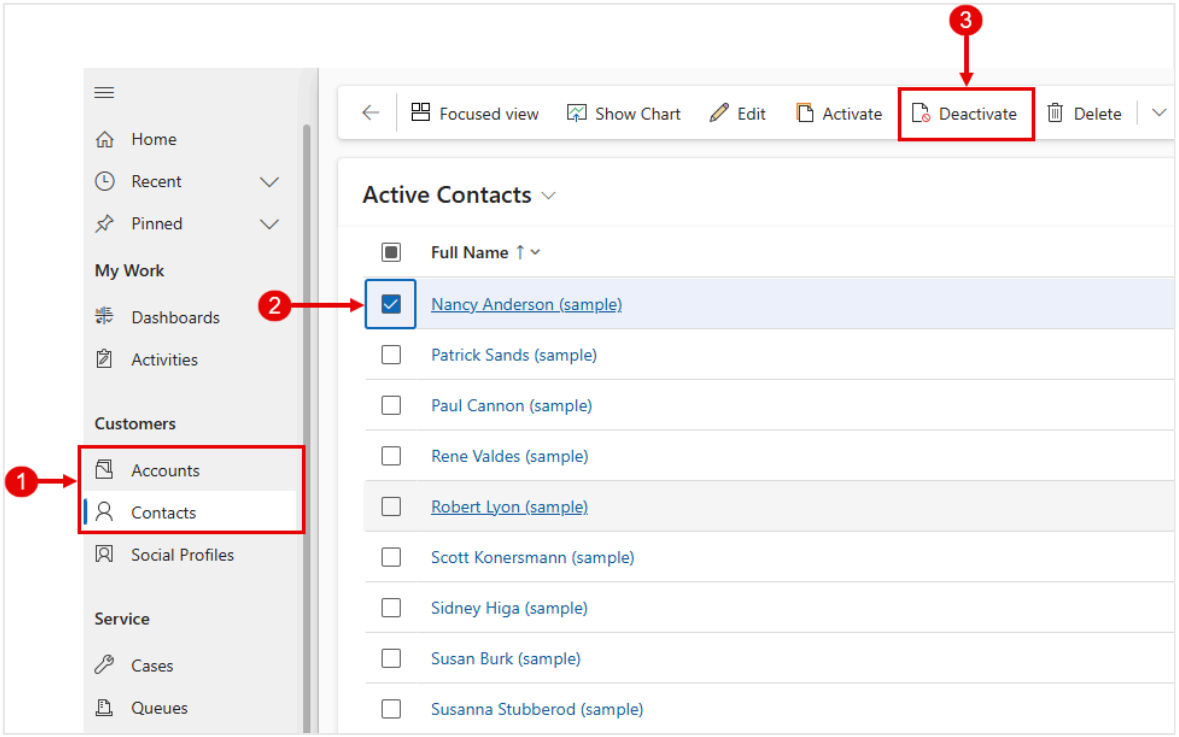
In a model-driven app, you can deactivate an account or contact rather than deleting it. This ensures the integrity of the audit trail associated with that row.

A deactivated account or contact becomes inactive, which means it cannot be edited or used in establishing new relationships with other rows. However, all relationships created with the deactivated item are still available.

If later you need to reactivate a deactivated account, it's easy to do so.

Deactivate an account or contact

1. From the menu on the left, go to **Accounts** or **Contacts**.
2. Select the active account or contact that you want to deactivate, on the command bar select **Deactivate**, and then confirm the deactivation.



Activate an account or contact

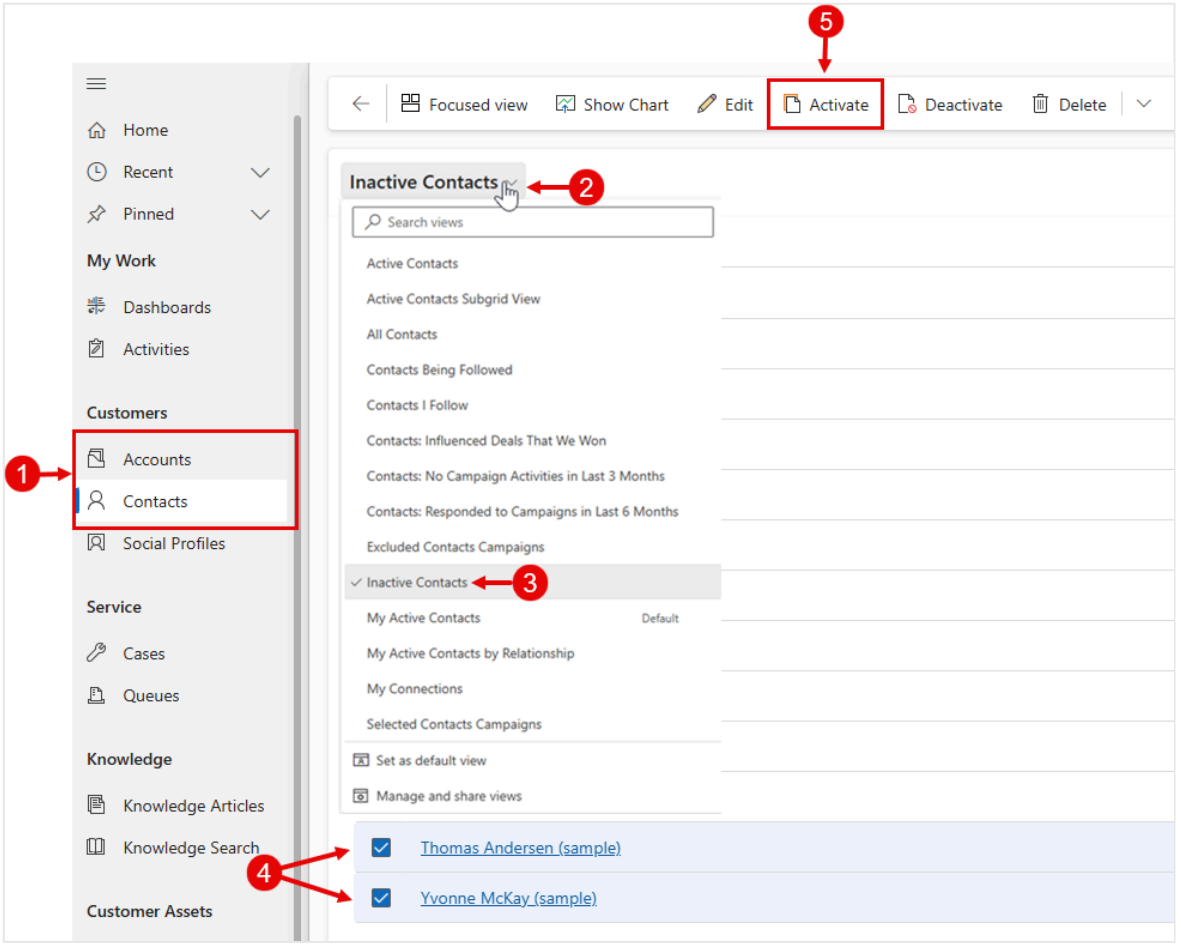
1. From the menu on the left, go to **Accounts** or **Contacts**.

2. Go to the **System Views** list.

3. Select **Inactive Accounts** or **Inactive Contacts**.

4. Select the inactive accounts or contacts you want to activate.

5. Select **Activate**, and then confirm the activation.



Feedback

Was this page helpful? Yes No

Provide product feedback

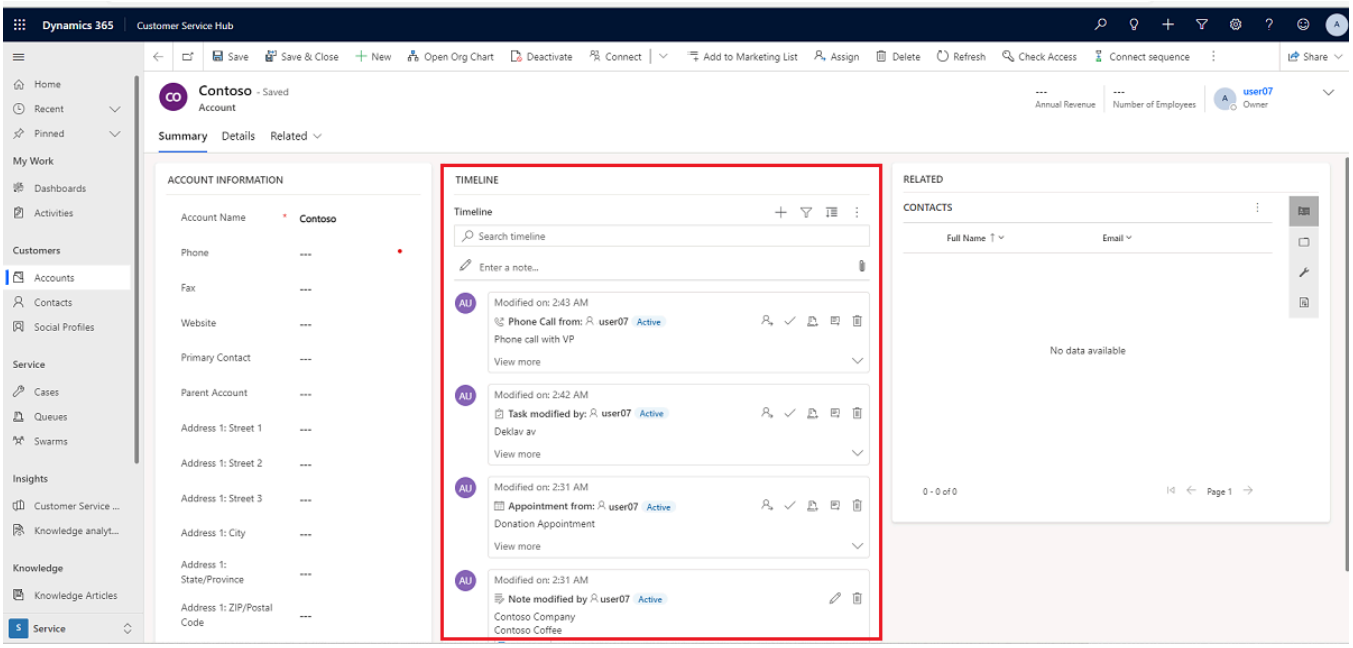
Use timeline

Article • 05/29/2025

Timeline is a modernized and optimized control that helps provide you with at-a-glance understandability when working with your customers. You can use the timeline to view information that is connected to a row in a table and occurs over time in a single stream. Some information that can be logged into the timeline includes notes and posts, in addition to displaying activities such as emails, appointments, phone calls, and tasks.

Timeline allows you to:

- Access the command bar to perform common actions quickly
- Filter and view important notes, posts, and activities using multiple filter options, and keep the filtered settings across activities
- Quickly view and manage email conversation threads with ease
- Keep detailed customer records by attaching important files and images to posts



Timeline functionality

The timeline section on the table form is intended to centralize access to information and streamline effort to accomplish actions. The following section is an overview of how the timeline functionality works in each section on the table form.

When enabled by your system administrator, you can use the icons that are displayed on the top-right navigation in the timeline for quick access to do the following actions:

- Create a timeline row
- Work with Filters

- Work with row
- Explore more commands
- Search row
- Add attachments

The timeline dashboard displays only the entities that are configured for the signed-in user. So, what you see on the timeline may differ from what another user sees. The signed-in user sees any rows they followed if those rows are configured to display on the dashboard. For example, if only email and appointments are configured to display, then the signed-in user only sees email that relates to them, which can be email they sent, email sent to them, and email they follow. The same would be true for appointments in this scenario. For more information on setting up timeline, see Set up the timeline control.

> **Note**
>
> The Notes feature isn't available from the timeline.

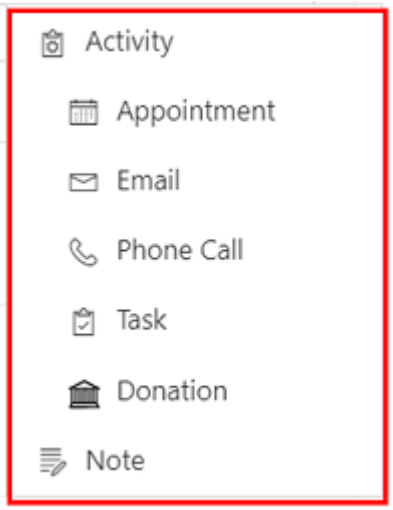
Create a timeline row

When enabled by your system administrator, you can use the **Create a timeline record** feature to quickly create appointments, tasks, send emails, and make notes and posts and associate them with other rows.

1. Select **Create a timeline record** icon.



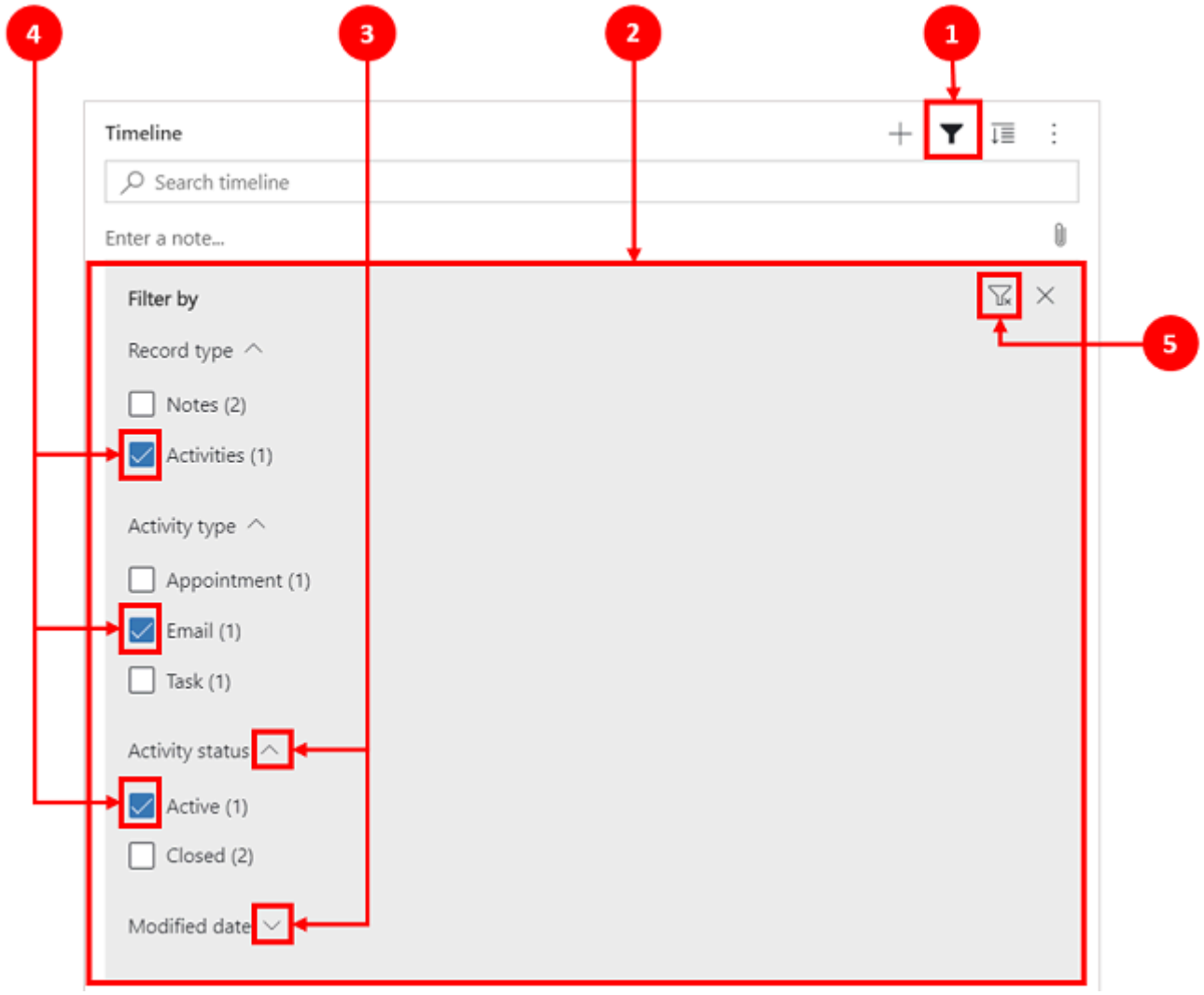
2. Select the type of row you want to create from the display options available.



For information on timeline row configuration, Set up the timeline control.

Work with filters

When enabled by your system administrator, the timeline **Filter** further streamlines your productivity by allowing you to customize your experience and view data that's relevant to you.



Legend:

1. The timeline **Filter** icon is located in the top-right navigation on the timeline.
2. The filter pane displays when the filter icon is selected.
3. Use the carats to expand or minimize selections.
4. When enabled by your system administrator, row and activity types are displayed in the filter pane. Row and activity types can be filtered by selecting the corresponding box next to the item.
5. The icon removes and clears all filter selections from the filter pane.

Filter rows on timeline

Filters are valuable for sorting data. You can easily filter through rows and activity types using multiple options to quickly see what matters to you. The filter is available for the activities, notes, posts, and custom entities that are present in timeline.

Filter icon display

- When the **Filter** icon displays as a transparent , it means the filter pane is empty and nothing has been applied.
- When the **Filter** icon displays as dark , it means filters are applied.

Filter options

The following category and subcategory options are available on the filer menu:

Expand table

Category	Subcategory
Row type	• Notes • Posts • Activities
Activity type	• Appointment • Campaign Activity • Campaign Response • Email • Fax • Case Resolution • Letter • Opportunity Close • Order Close • Phone Call • Quote Close • Recurring Appointment • Social Activity • Task • Project Service Approval • Booking Alert • Conversation • Session • Customer Voice survey invite • Customer Voice survey response • Custom activities (as configured by your system administrator)
Activity status	• Active • Overdue

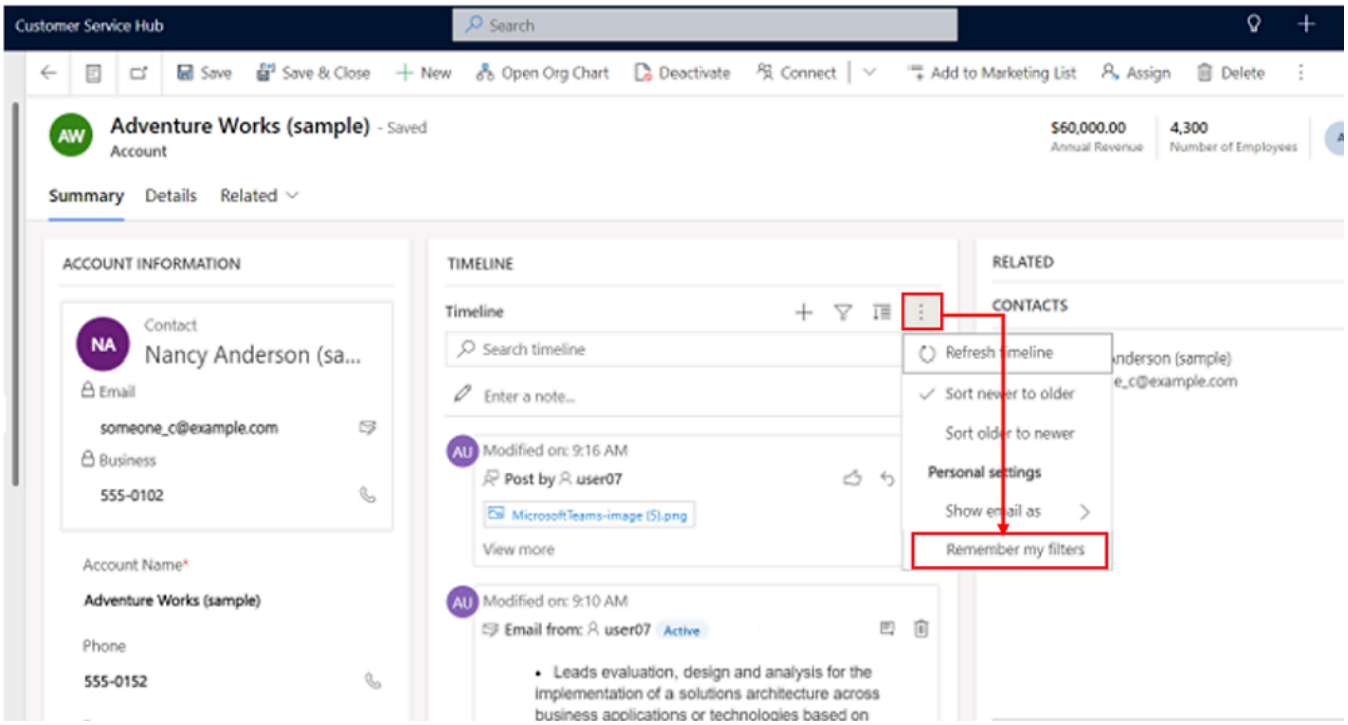
Category	Subcategory
	• Closed
Activity due date (active)	• Next 30 days • Next seven days • Next 24 hours • Last 24 hours • Last seven days • Last 30 days
Posts by	• Auto post • Users
Modified date	• Last 24 hours • Last seven days • Last 30 days

Save filter settings

You can choose to save your filter settings so that you don't have to set them again each time you work with activities in the timeline.

To save your filter settings, after configuring the filters you want, select the ellipses, and then select **Remember my filters**.

> ⓘ **Note**
>
> If you have a system administrator or basic user role, Remember my filters works by default. If you don't have either of these roles, you must have Create/Read/Write privileges to be added to the msdyn_customcontrolextendedsettings table for the setting to work.

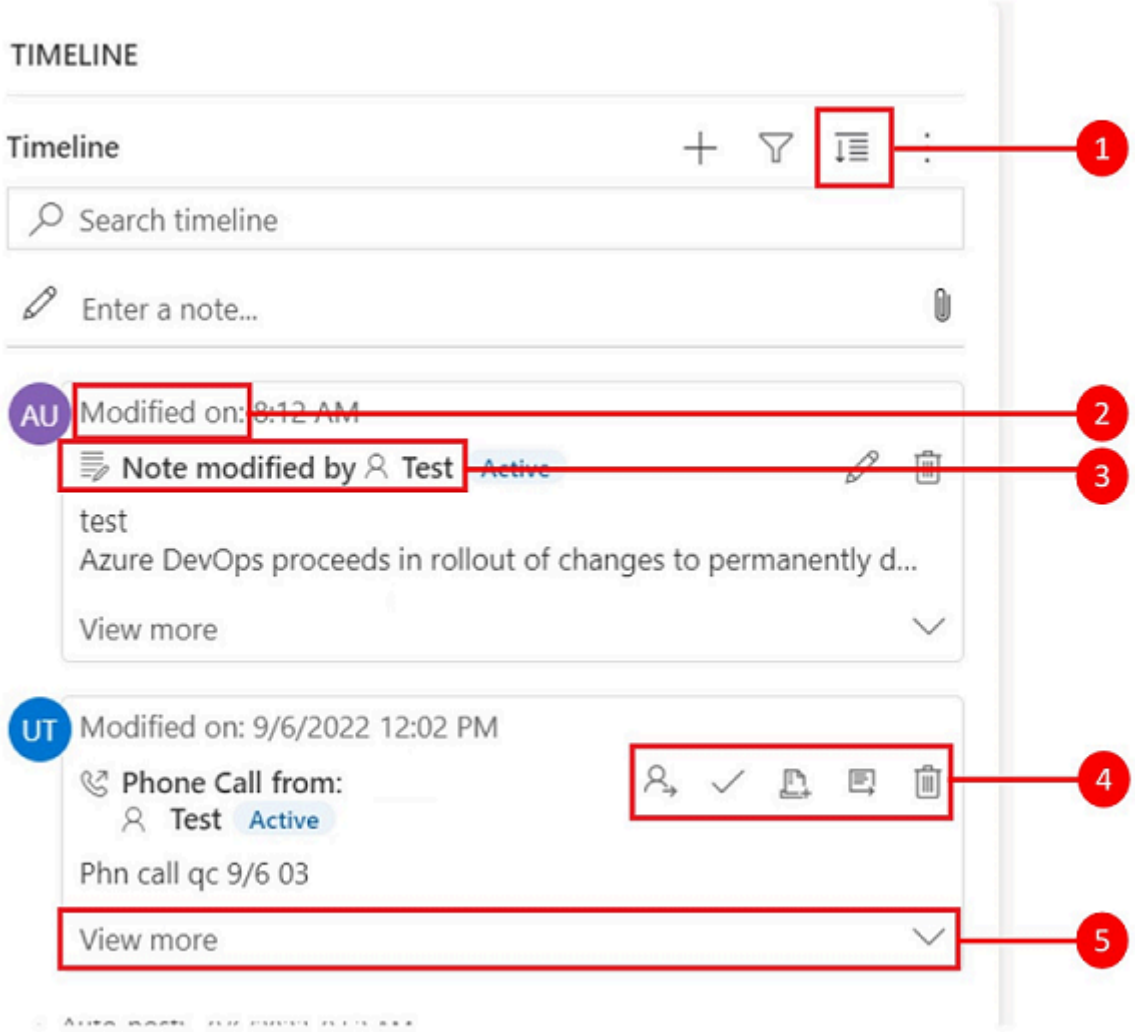


Work with rows

When working with rows, you can view key information at a glance or expand rows to view more details. When you hover over a row, actions display that you can use.

Expand all rows

When enabled by your system administrator, you can view timeline rows quickly in a collapsed format or expand a row to see more details for ease of use.

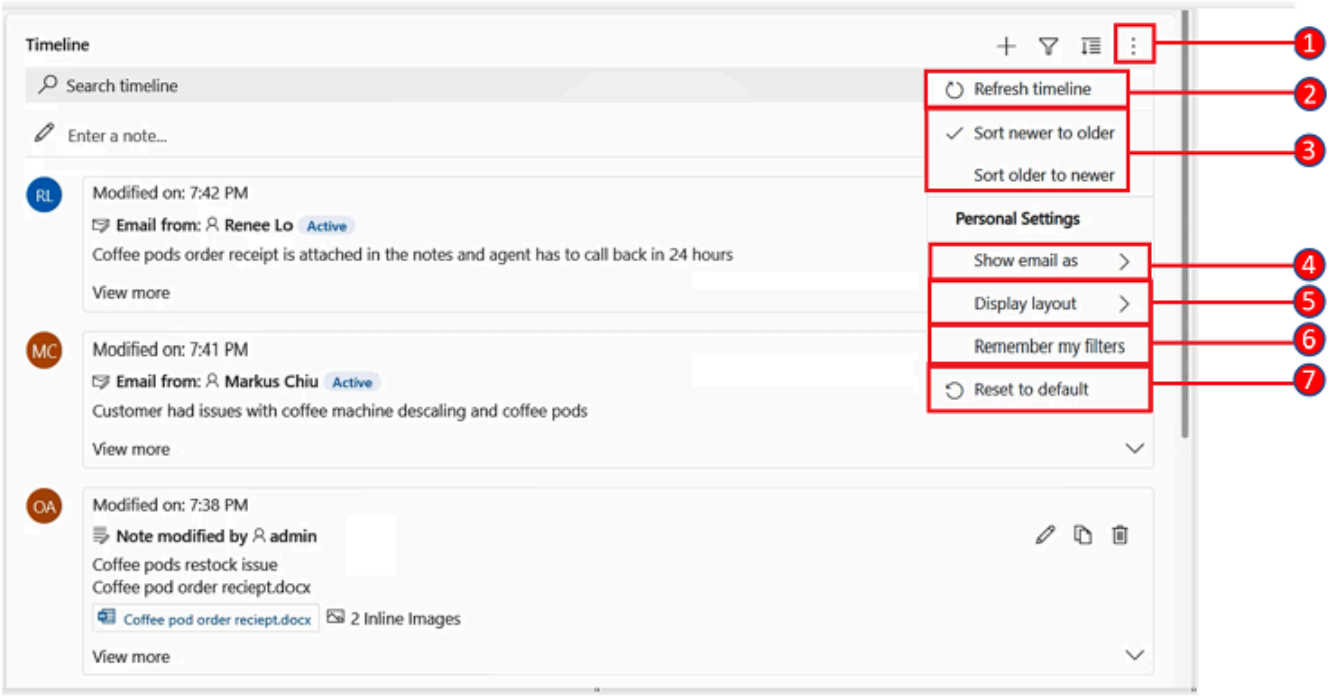


Legend

1. The **Expand all records** icon is used to expand all rows listed on the timeline.
2. The **Modified on** timestamp displays the time the record was last updated.
3. Each row has a list of commands relevant to their row type that displays in the top right corner for ease of use.
4. Row backgrounds change to grey when you hover over it, and it becomes transparent again when you move the mouse away.
5. The carats in the bottom-right corner of each row allow you to expand (^) or minimize (v) the display views.

More commands

The **More commands** option contains refresh, sort, and email functionalities.



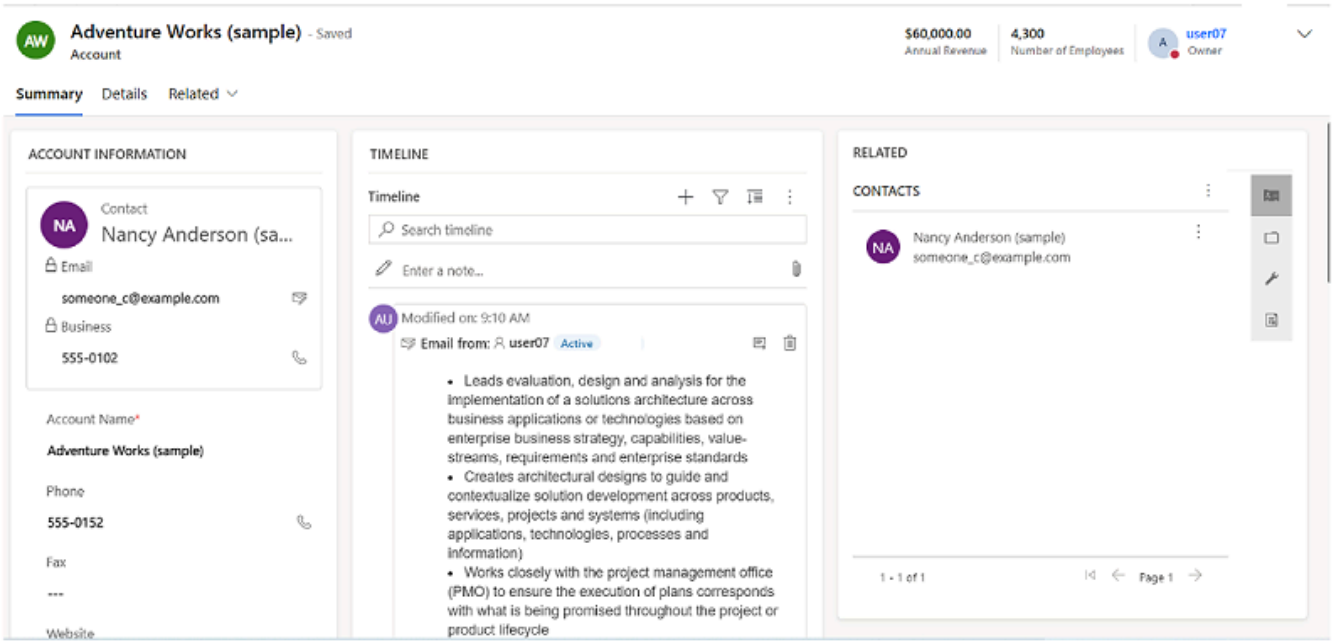
Legend

1. The **More commands** icon, when expanded, provides more functionality options.
2. The **Refresh timeline** command updates row details in the timeline.
3. The **Sort newer to older** or **Sort older to newer** command is used to sort rows.
4. The **Show email as** command is used to enable or disable threaded email views.
5. The **Display layout** command allows you to choose from two different views: **Roomy** or **Cozy**. **Roomy** (set by default) shows you more information, such as profile details, and adds space between the rows. **Cozy** hides some of the details, such as icons, and compacts the space between rows.
6. The **Remember my filters** command allows your filters to persist when you close the record so that they continue to be applied across sessions.
7. The **Reset to default** command reverts all of the command settings you changed back to the default settings.

View email inline in timeline

If preferred, you can view email messages inline on the timeline rather than open them as separate pages.

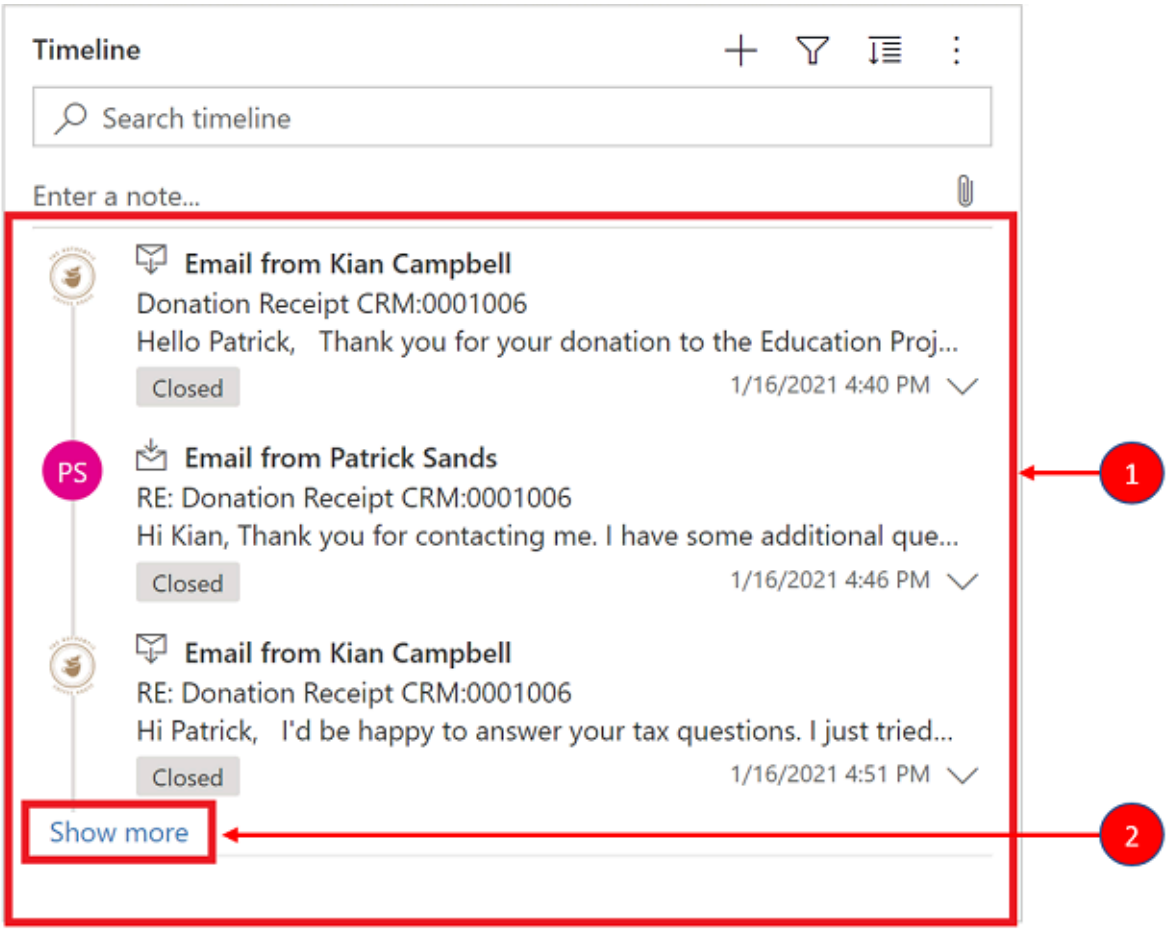
To view an email directly in the timeline, on the row, select **View more**. The email is displayed on the same form. When you're finished viewing the email, you can collapse the screen and return to the previous view of the form.



Threaded email views in timeline

Use the threaded email view option to save valuable space when you have emails with multiple replies. This view consolidates email threads when collapsed.

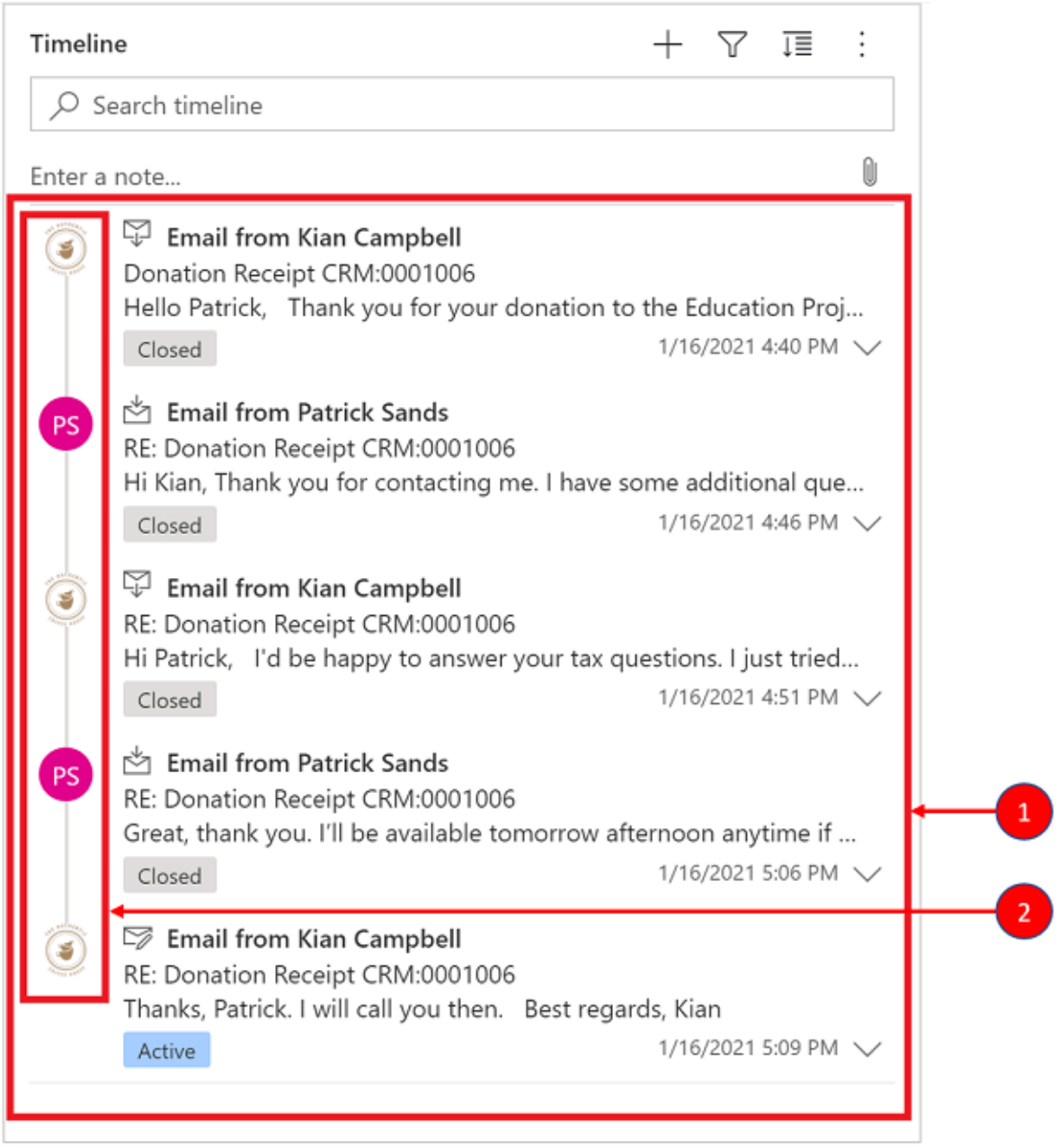
Collapsed threaded email view



Legend

1. When enabled using the **Always show email as conversations** option under **More commands** ⋮, emails with multiple responses are grouped together into threaded emails.
2. The **Show more** link appears in the bottom-left corner of the last visible email in the thread to view the complete string of email responses.

Expanded threaded email view



Legend

1. When expanded, the most recent 10 email responses in the thread display. To display more, select **Show more** to expand the view again.
2. Emails in the same thread are displayed as being linked together.

Enable threaded email views in timeline

Enabling the threaded email view is done via the timeline configuration in Power Apps. Threaded email views must be done in this setting for the specific table, in the specific form, for each specific timeline.

Threaded email views can be enabled in timeline via two options:

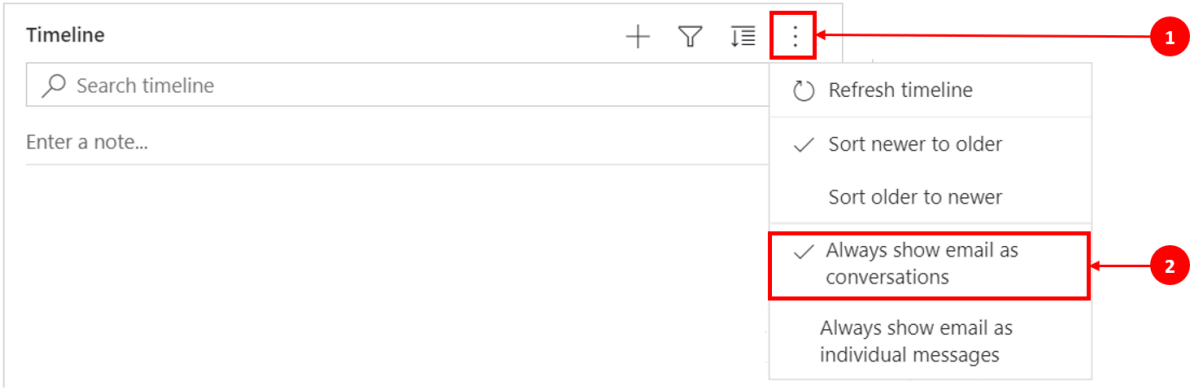
Option 1:

You can enable the threaded email view using the **More Commands** ⋮ feature, located on the right-hand side of the timeline.

Enable threaded email through the More Commands option

Steps:

1. Use the **More commands** ⋮ icon on timeline that displays a flyout menu.
2. Select **Always show email as conversations** to thread and group all emails in that specific timeline.



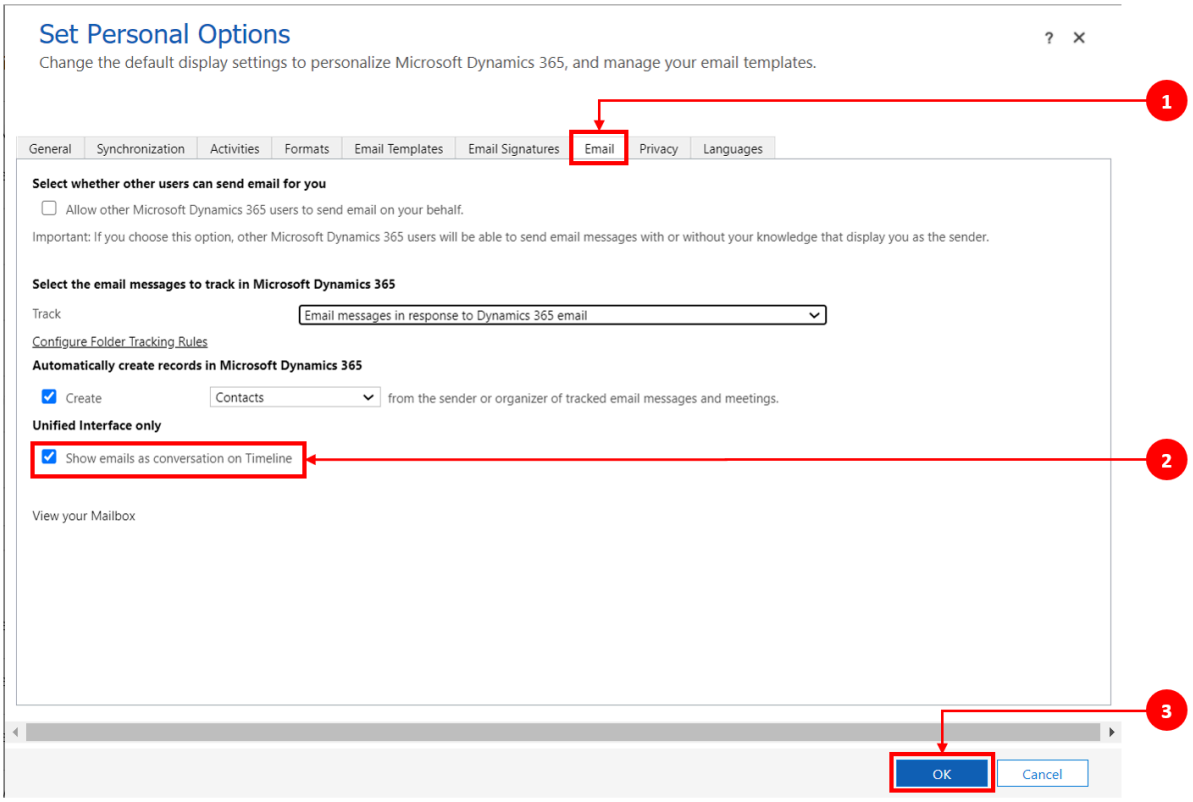
Option 2:

Enable threaded email through Personal Options setting

To enable threaded email via this option, go to the command bar and select **Settings**, then **Personalization Settings**. The **Set Personal Options** dialog box is displayed so you can use the following steps to enable the threaded email view on timeline.

Steps:

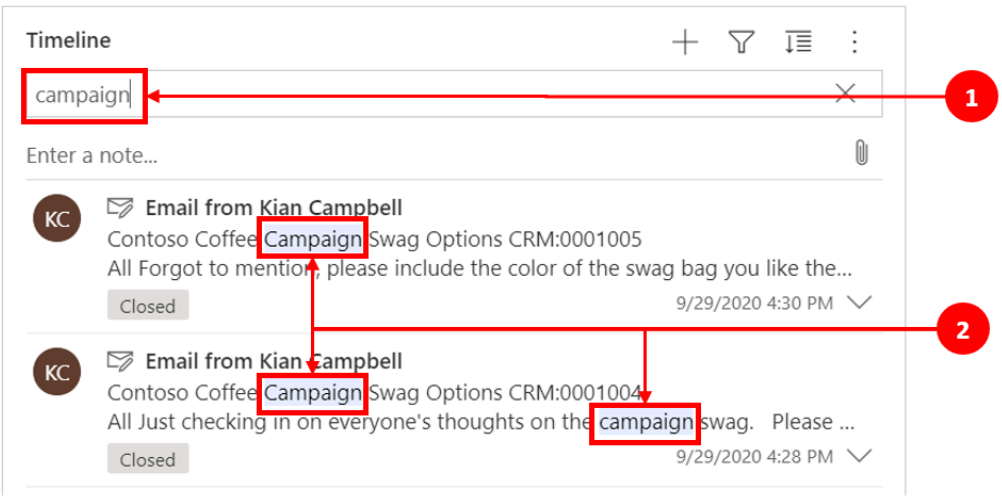
1. Select the **Email** tab.
2. Select the **Show emails as conversation on Timeline** check box.
3. Select **OK**.



> **Note**
>
> The **emails as conversation** view is based on the user's preference. Personal settings are tied to the user, not the form, which means when you enable the threaded email view, the setting is applied to all of your timeline views.

Search rows on timeline

When enabled by your system administrator, you can easily search for rows in the timeline. When you search for a phrase in the timeline, it searches in the title, subject of the row, body, or description fields of the row and displays the row for you.



Legend

1. The **Search** bar allows you to quickly find rows using keywords or phrases that the search feature retrieves and displays below the search bar.
2. Keywords that are used are highlighted.

Work with notes and posts

Notes allow you to record richly formatted information and images for reference, summary, or investigation. You can easily add a mention to another user or a reference to a timeline row. The mention capability is available in timeline only when rich text notes and rich text posts are enabled.

When you type the @ symbol, the results you see are matches with the first name, last name, or email address of system users starting with your search string. When you mouse over the profile image, you can view the user's detailed profile card and online presence status. More information: View the profile card for a contact or user.

When you type the # symbol, the results you see are matches with the account and contact name table rows starting with your search string.

Add a note or post to a row on timeline

When enabled by your system administrator, you can either add a note or a post to a row on timeline. Each note and post has a maximum size limit of 100,000 characters.

You can also enter a note and add it to a row, but you can only add one attachment per note. You can only add an attachment if notes are enabled.

Note display view

When enabled by your system administrator, the **Note** field is displayed under the **Search** feature on timeline.



Post display view

When enabled by your system administrator, the **Post** field is also displayed under the **Search** feature on timeline.

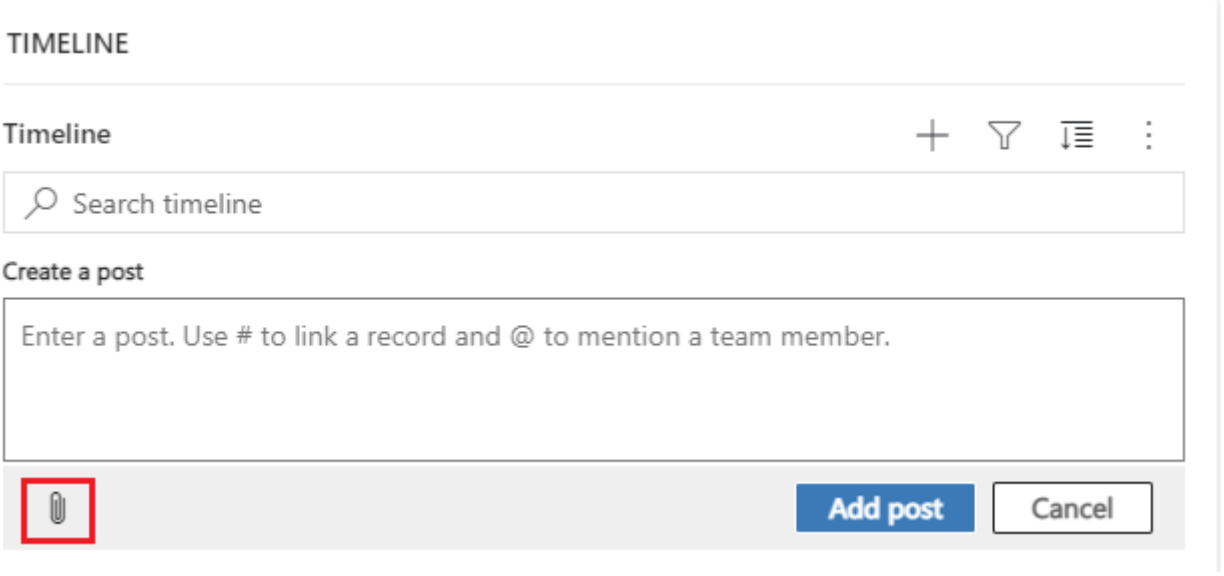


Add an attachment to a post

You can add an attachment to a post, such an image or a document. The supported file formats are as follows: .JPG, .PNG, and .PDF.

To add an attachment to a post:

1. In the **Create a post** section in the timeline, select the **Attach** icon in the lower-left corner.

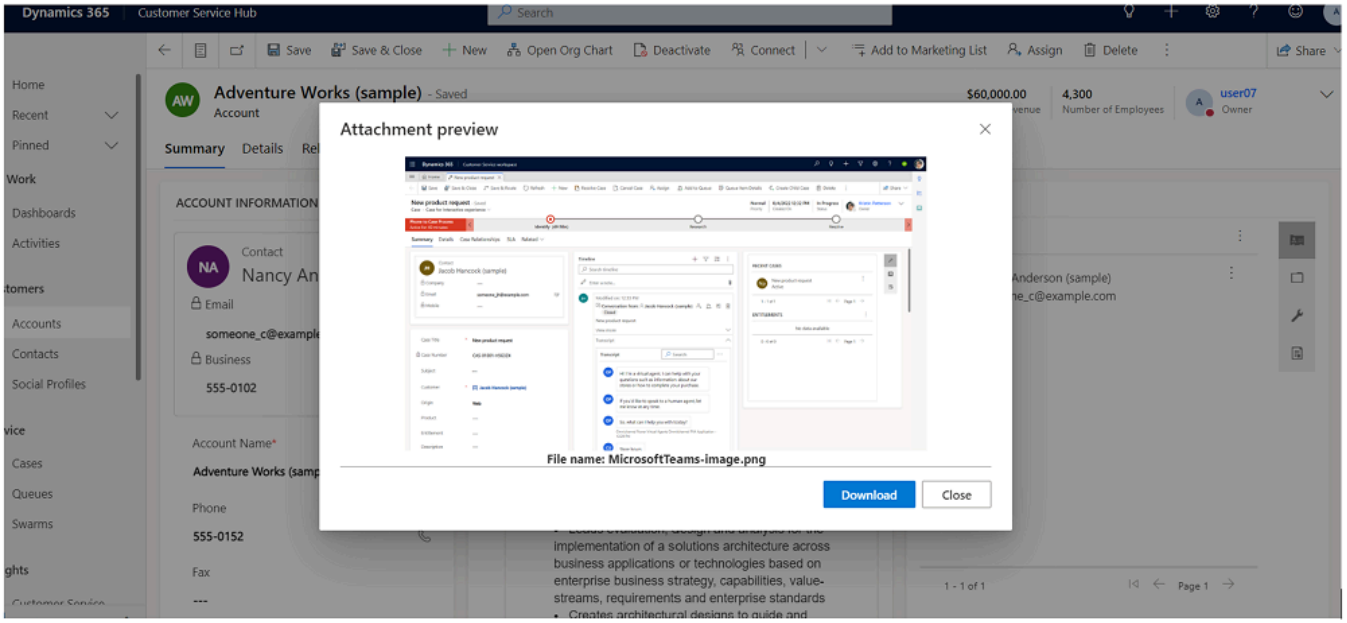


2. Browse to the file you want to attach to the post, and then select it.

3. Finish the edits you want to make to your post, and then in the timeline, select **Add post**.

Preview an attachment in the timeline

You can view previews of attachments in the timeline if they're in the supported file format (.JPG, .PNG., or .PDF). Any attachment that's viewable inline on the timeline can be downloaded, whether or not a preview is available.



To preview an attachment in the timeline, do either of the following steps:

In edit mode:

1. Right-click the attachment, and then select **Preview**. The **Attachment preview** dialog opens with the attachment displayed. You can view and download the attachment from the dialog.
2. When you're finished viewing the attachment, select **Close** to return to the note or post.

OR

In read mode:

1. Select the attachment. The **Attachment preview** dialog opens with the attachment displayed. You can view and download the attachment from the dialog.
2. When you're finished viewing or downloading the attachment, select **Close** to return to the note or post.

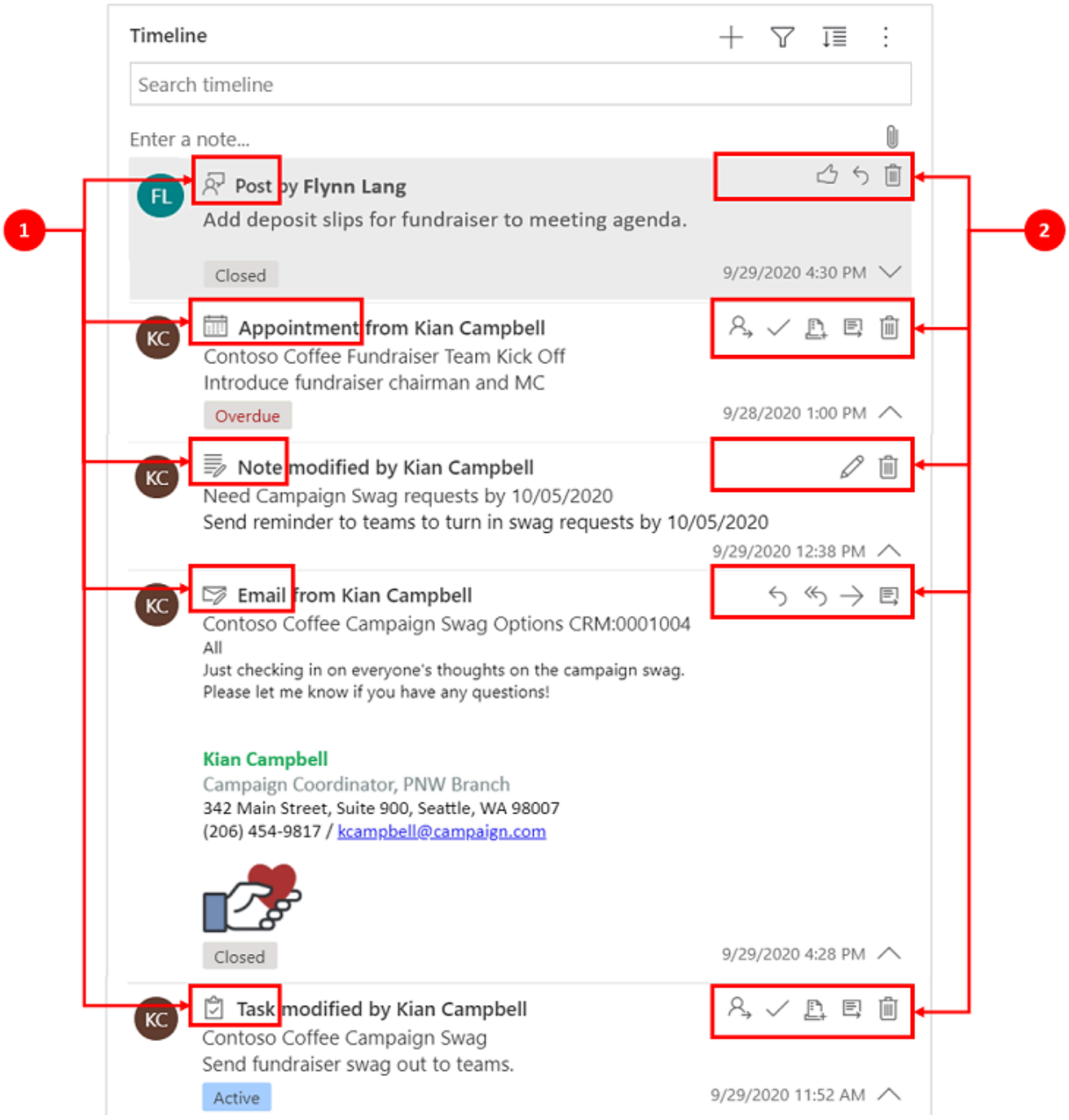
> ⓘ **Note**
>
> The use of the **Attachment preview** dialog to view the file can be skipped in the browser by using **Ctrl + Left click (of mouse)**. By doing so, the attachment is directly downloaded without previewing it.

Access the command bar to perform actions

When you hover your cursor on or expand an activity, post, or note, the command bar appears in the top-right corner of the row with the specific options that support that activity, such as

assigning it to others, adding it to a queue, converting it to a case, liking or replying to a post, editing a note, or deleting an activity. However, not all rows can perform the same actions.

Command actions by row

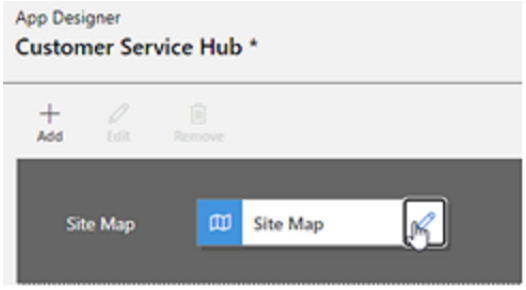


Legend

1. Only the actions associated with that row are displayed in the top-right command bar displayed in the row.
2. Each row displays a different set of associated actions. Not all actions work on all rows.

The following table is an overview of the command bar icons and the actions they support:

Expand table

Icon	Name	Description
	Assign	Lets you assign a task.
	Close Activity	Lets you close the activity row.
	Add to Queue	Lets you add a task to a queue.
	Open Row	Lets you open a row.
	Delete	Lets you delete the row.
	Reply	Lets you reply to an email. **Note**: This command is only available for Email.
	Reply All	Lets you reply to all for an email. **Note**: This command is only available on Email.
	Forward	Lets you forward an email. **Note**: This command is only available on Email.
	Like this post	Lets you like a post. **Note**: This command is only available on Posts.
	Edit this note	Lets you edit a note. **Note**: This command is only available on Notes.

The following table is an overview of the default command bars for frequently used activities, notes, and posts:

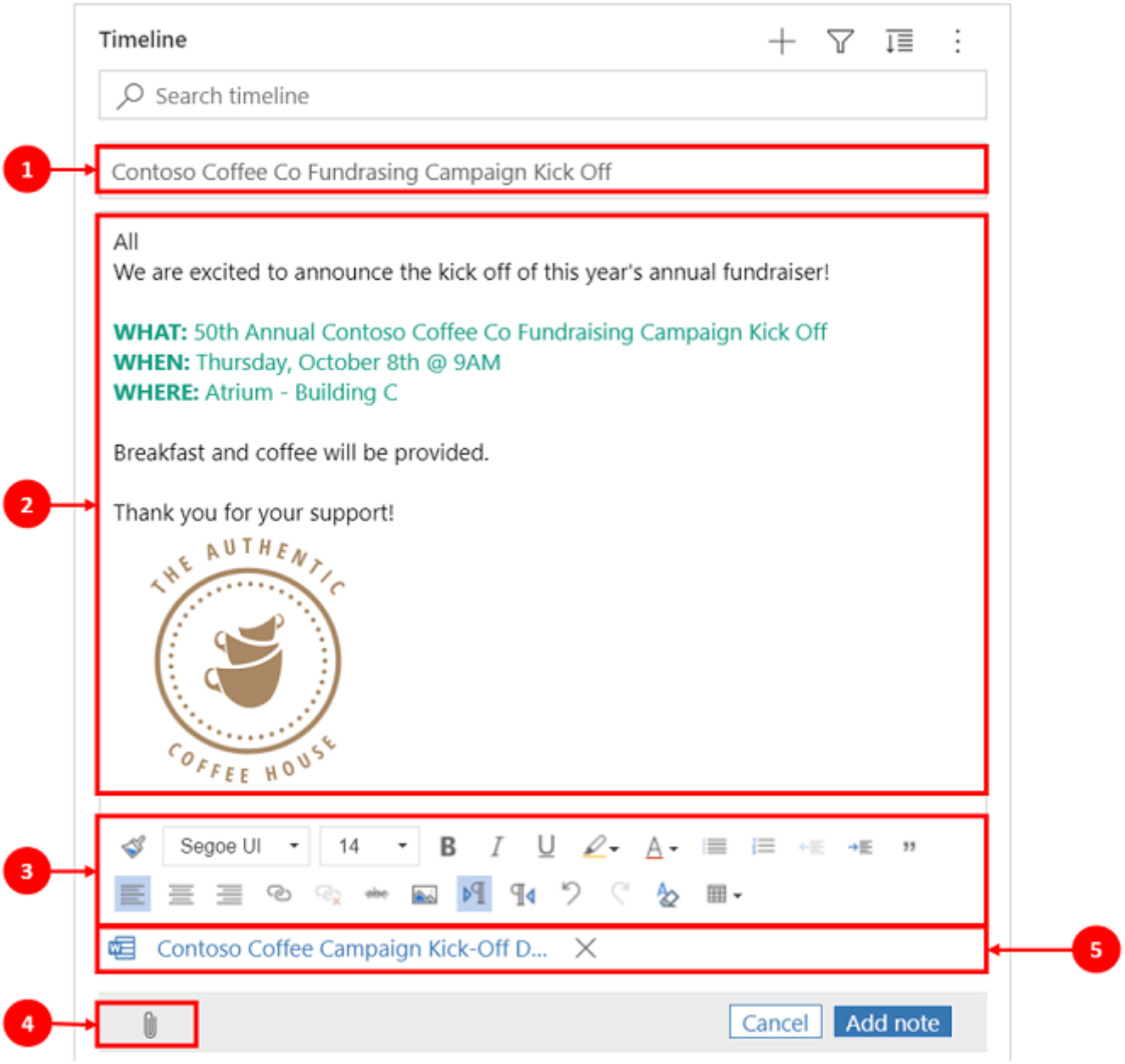
Expand table

Activity	Command bar icons	Notes
Email (draft)	Open Row, Delete	
Email	Reply, Reply All, Forward, Open Row	
Appointment	Assign, Close Activity, Add to Queue, Open Row, Delete	

Activity	Command bar icons	Notes
Appointment (closed)	Assign, Add to Queue, Open Row, Delete	
Task	Assign, Close Activity, Add to Queue, Open Row, Delete	
Task (closed)	Assign, Add to Queue, Open Row, Delete	
Phone Call	Assign, Add to Queue, Open Row, Delete	
Note	Edit, Delete	
Post (user)	Like, Reply, Delete	Only the author of the post has the ability to delete it. The author always has the ability to delete regardless of user privilege.
Autopost	Like, Reply, Delete	

Work with notes

The **Note** message box has a rich text editor that enables you to use rich text and create well-formatted content.



Legend

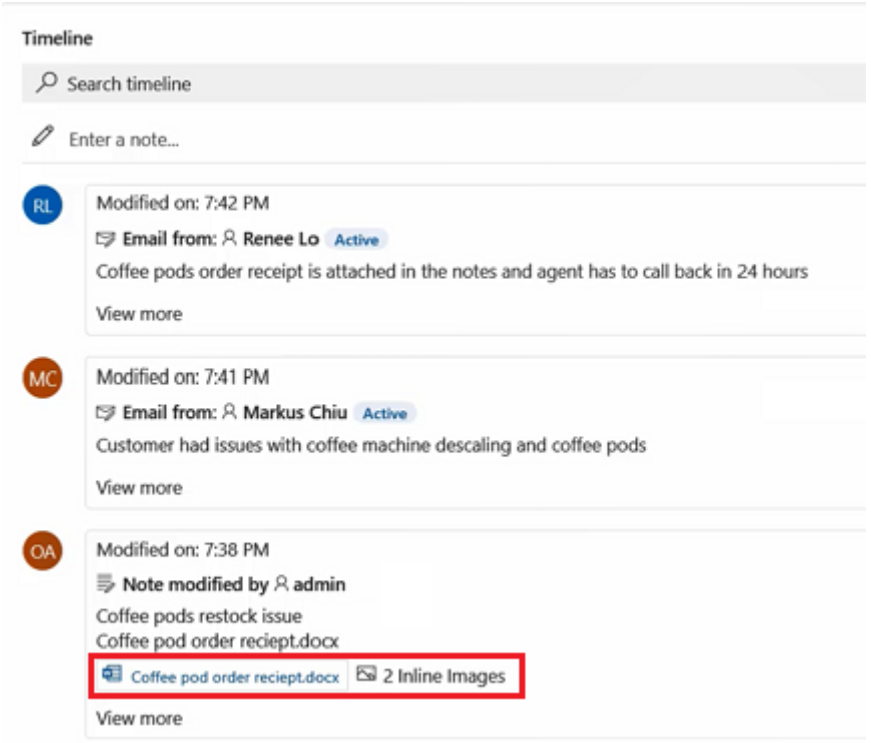
1. Note title field for adding headings and subjects.
2. Notes have a rich text editor so you can format and structure your messages.
3. The rich text editing tool expands to a full menu for quick access to formatting features.
4. The paperclip is used to attach files to your note.
5. Attached files are displayed at the bottom of the note.

> **Note**
>
> - Only **Notes** have rich text capability, and they're only available in the timeline.
> - You can only format the message body of a note; not the title.
> - The rich text editor is available only for Unified Interface.

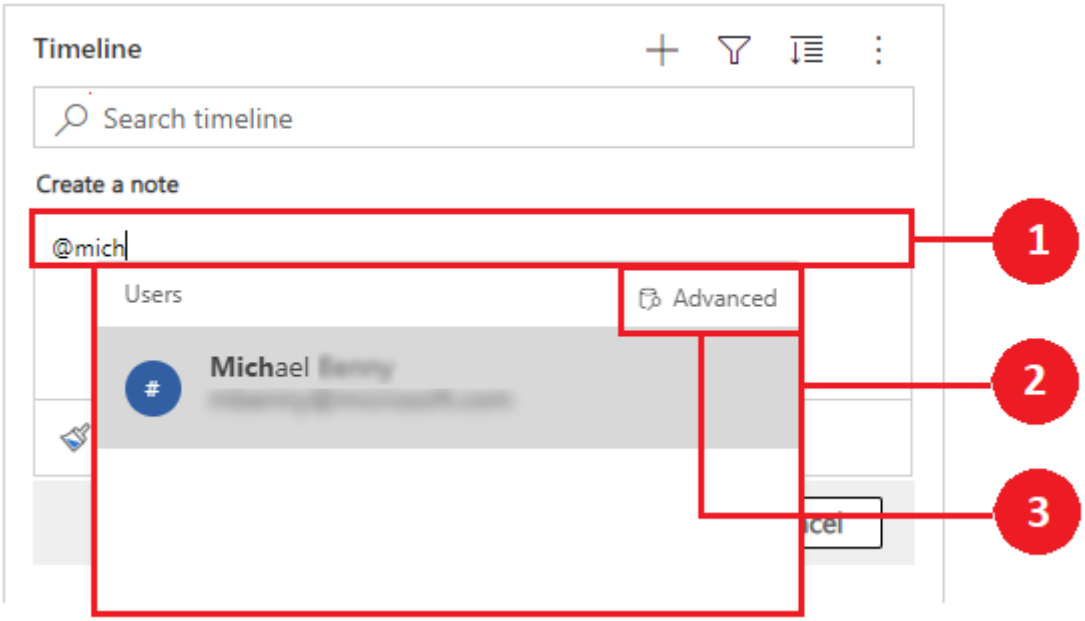
- If you create a rich text editor note in Unified Interface and then view the note in the web client, it displays in HTML syntax.

View whether a note has attachments and inline files in the timeline

If you attach a file to a note or include an inline image in the note, the timeline shows that the note has an attachment or inline files. If you attach a file or include multiple files inline, the note shows the number of files using the first two file names and a number to represent the remaining count of and the file type.



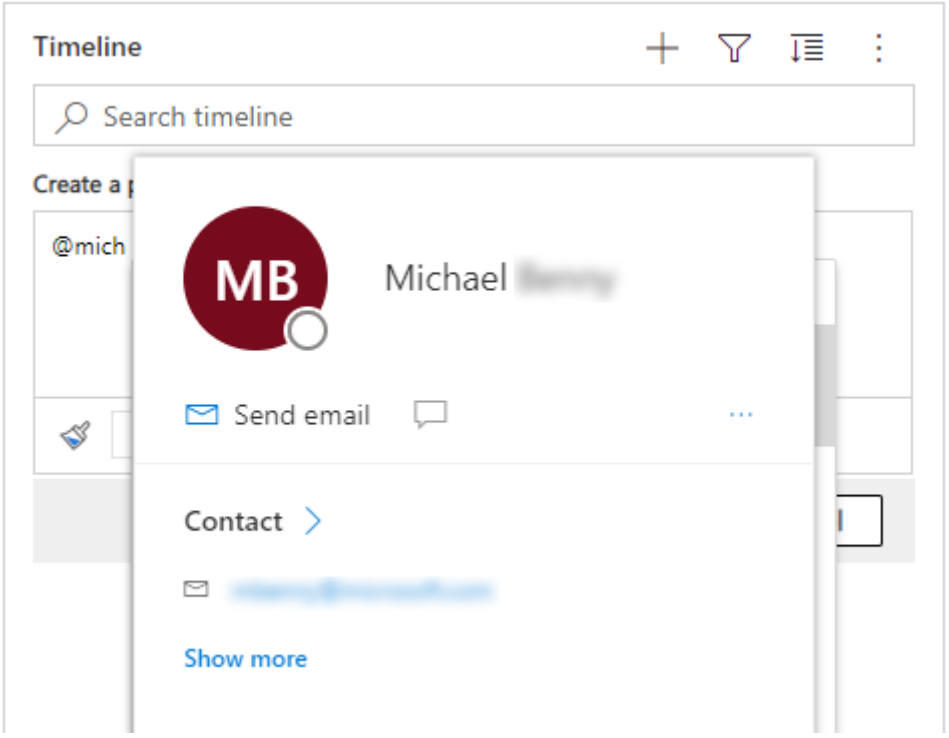
Add a mention or reference a team member in a note



Legend

1. Typing the @ sign in the **Note** field displays a flyout menu with a list of users that have been recently used.
2. If the user row isn't displayed, typing the name after the @ sign refreshes the rows in the flyout menu using the **Quick Find** search engine that's operating in the background.
3. Switch views using **Advanced** to display the **Lookup Records** pane. In the **Lookup Records** pane, you can switch between the **Recent records** and **All records** view to search rows.

When you mouse over the profile image, you can view the user's detailed profile card and online presence status.



> **Note**
>
> You can mention only a system user in a note. For example, a team member within your organization.

Add an attachment to a note in timeline

When enabled by your system administrator, you can add an attachment, such as a file, to a note in timeline to share with users. Any attachment that is viewable inline on the timeline can be downloaded, whether or not a preview is available.

> **Note**
>
> - The list of allowed attachments is configured by your system administrator.
> - On iOS and Android you can only select and attach a picture or video.

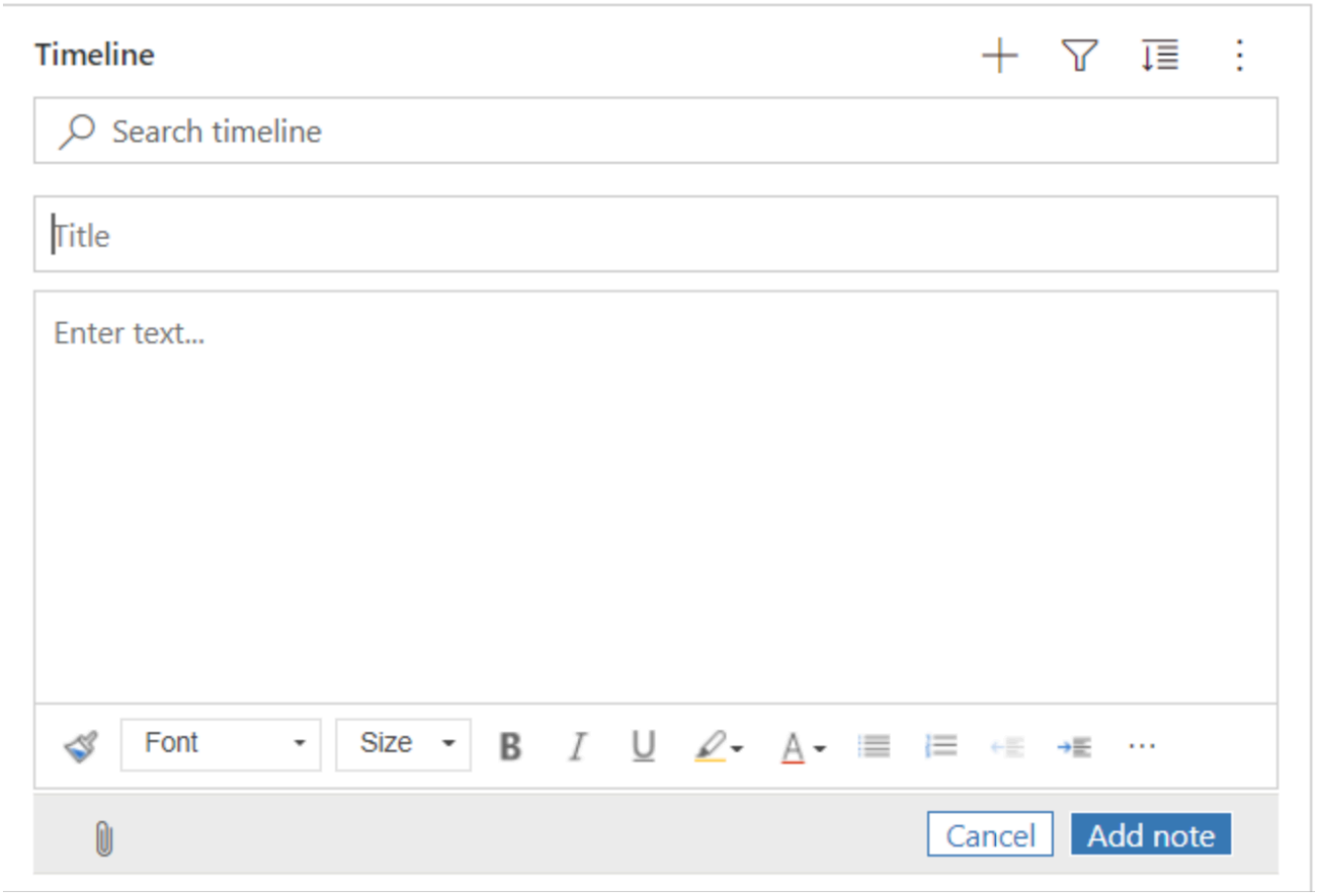
Select the paperclip icon to attach a file to a note.



Edit Note

When you attach a file to your note, the **Edit note** screen displays, and attachments show at the bottom of the **Edit note** screen.

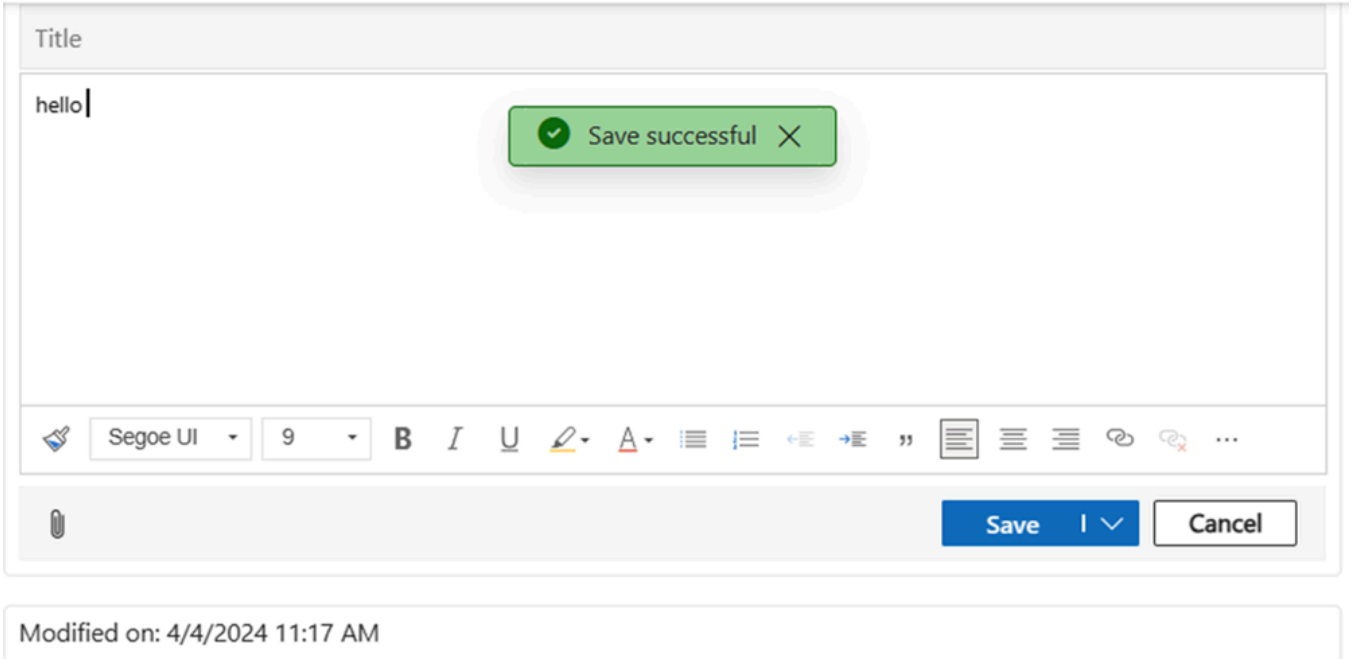
> **Important**
>
> You can only add one file attachment per note.



Quickly save notes

The Quick save feature allows you to save your notes faster by selecting **Ctrl**+**S** on your keyboard at any point when you're typing a note. When you use Quick save, you don't have to navigate to **Save** or **Save and Close** on the command bar.

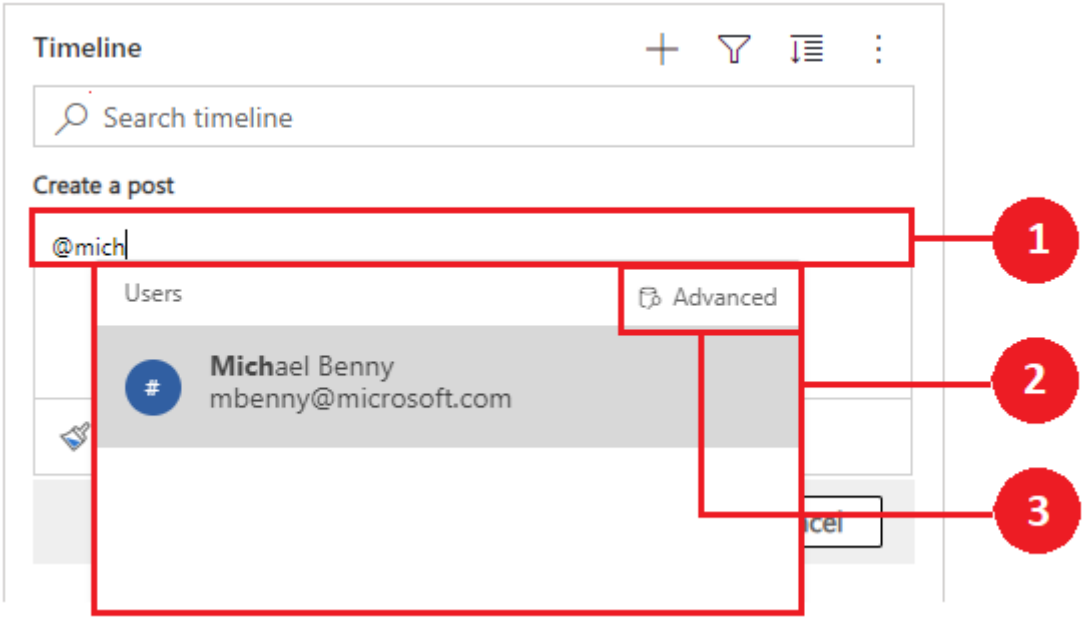
When you use **Ctrl**+**S**, a **Save successful** confirmation message is displayed in the note to let you know your note saved on the timeline.



Work with posts

When enabled by your system administrator, you can either add a post to a row or mention to another user or timeline.

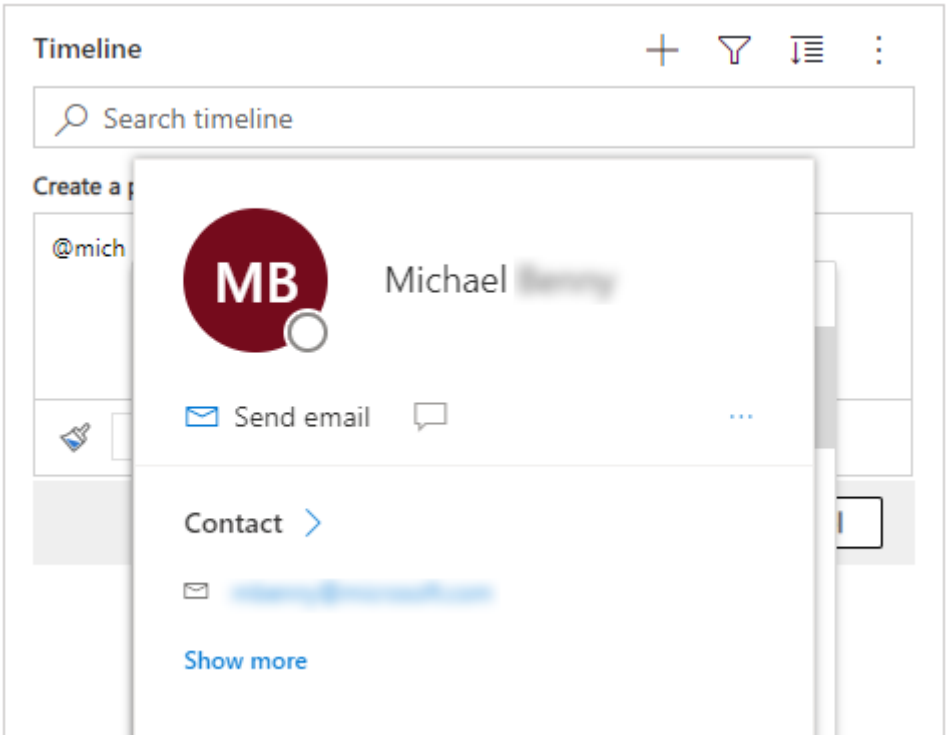
Add a mention or reference a team member in a post



Legend

1. Typing the @ sign in the **Post** field displays a flyout menu with a list of users that have been recently used.
2. If the user row isn't displayed, typing the name after the @ sign refreshes the rows in the flyout menu using the **Quick Find** search engine that's operating in the background.
3. Switch views using **Advanced** to display the **Lookup Records** pane. In the **Lookup Records** pane, you can switch between the **Recent records** and **All records** view to search rows.

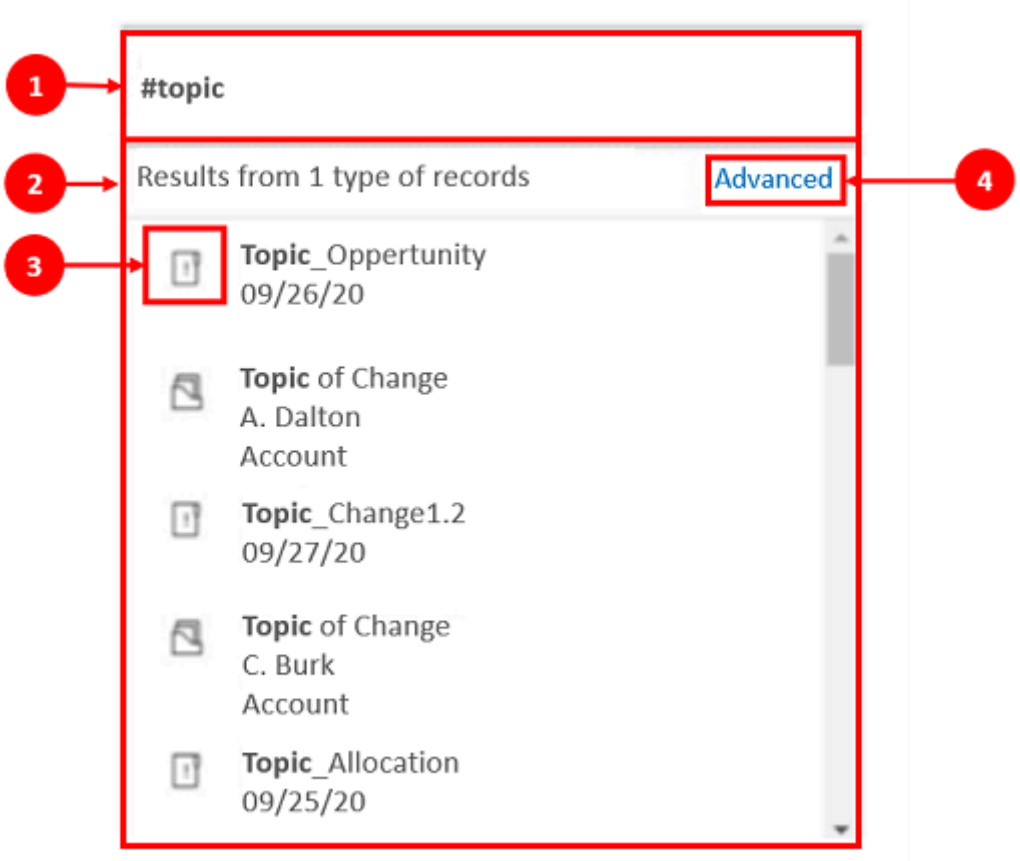
When you mouse over the profile image, you can view the user's detailed profile card and online presence status.



> **Note**
>
> You can mention only a system user, i.e. a team member within your organization, in a post.

Add references in a post in timeline rows

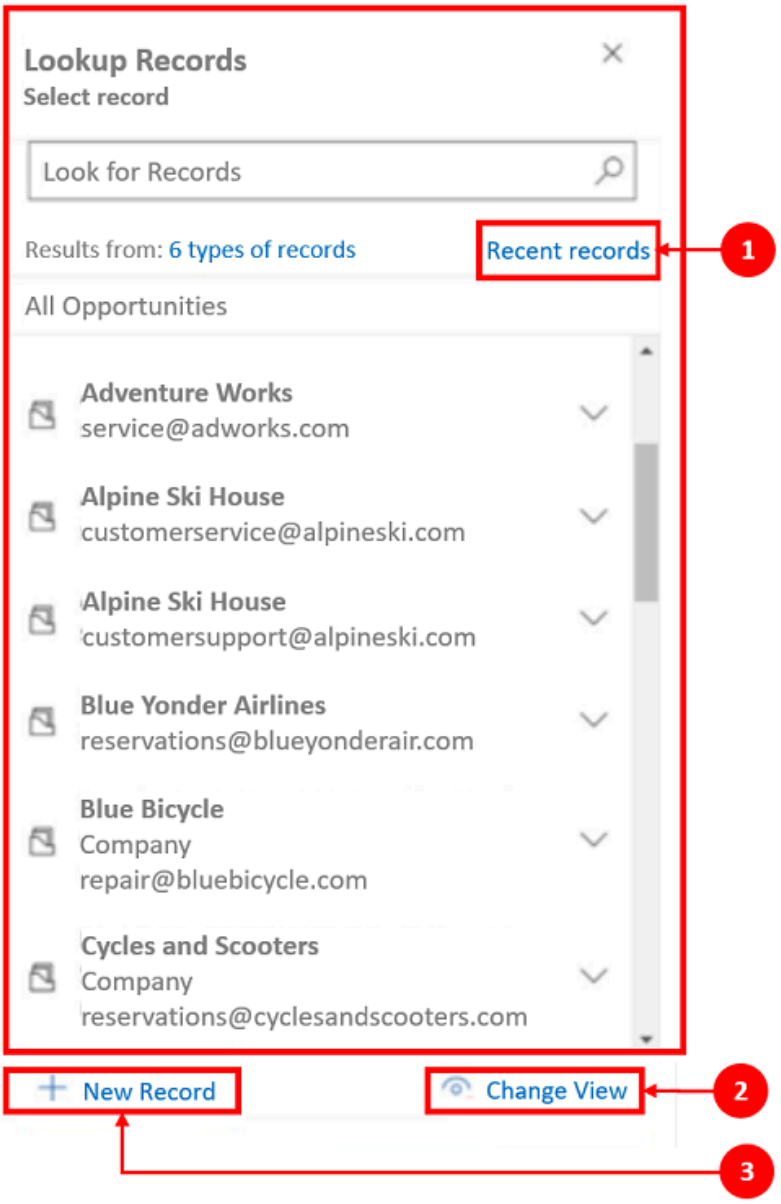
To add a reference such as links in a post to a timeline row, select the + icon, select **Post**, and then complete these steps:



Legend

1. Links can be added to a row by typing # which displays a flyout menu with a list of rows recently accessed. Type a keyword after the # tag and the display refreshes in the flyout menu via the **Quick Find** search engine that's operating in the background.
2. Rows displayed can be of any row type, such as accounts, contacts, or leads.
3. The icons beside the row names in the flyout menu indicate the row types, which help you determine which row to select.
4. **Advanced** provides more options when a row can't be located by allowing the user to switch to a different view or create a new row.

Look up rows in posts in timeline



Legend

1. The **Lookup Records** pane provides users with options to switch between the **Recent records** and **All records** view to find a row.
2. **Change View** to see rows from a different view.
3. **New Record** allows you to create a new row if you can't find the row you need.

For more information on notes and post functionality and configuration, see: Set up the timeline control

Use the rich text editor toolbar

The rich text editor toolbar provides features and functionality that allows you to work with rich text format in notes.

Formatting options

The following table describes the different formatting features and functionality options that are available in the rich text editor that you can use in notes.

> **Note**
>
> You can access your browser's context menu by selecting **Ctrl + right-click**. This setting is useful if you need to use your browser's built-in spellchecker. Otherwise, you can right-click to provide contextual formatting for any element you are using.

Expand table

Icon	Name	Shortcut key	Description
	Format Painter	Ctrl+Shift+C, Ctrl+Shift+V	Apply the look of a particular section to another section.
	Font	Ctrl+Shift+F	Select a font of your choice. The default font is Segoe UI. **Note**: When you select any formatted content, the font name for that content displays. If your selection contains multiple fonts, the topmost font name of your selection is displayed.
	Font size	Ctrl+Shift+P	Change the size of your text. The default size is 9 pt. **Note**: When you select any formatted content, the font size for that content displays. If your selection contains multiple font sizes, the topmost font name of your selection is displayed.
	Bold	Ctrl+B	Make your text bold.
	Italic	Ctrl+I	Italicize your text.
	Underline	Ctrl+U	Underline your text.
	Text Highlight Color		Make your text stand out by highlighting it in a bright color.
	Font Color		Change the color of your text.
	Bullets		Create a bulleted list.
	Numbering		Create a numbered list.
	Decrease Indent		Move your paragraph closer to the margin.

Icon	Name	Shortcut key	Description
	Increase Indent		Move your paragraph farther away from the margin.
	Block Quote		Apply a block-level quotation format in your content.
	Align Left	Ctrl+L	Align your content with the left margin. (Commonly used for body text to make it easier to read.)
	Align Center	Ctrl+E	Center your content on the page. (Commonly used for a formal appearance.)
	Align Right	Ctrl+R	Align content to the right on the page. (Commonly used for a formal appearance.)
	Link		Create a link in your document for quick access to web pages and files. Pasted or typed URL text is converted into a link. For example, "http://myexample.com" becomes "http://myexample.com ". In the **Link** dialog box, choose the type of link you'd like to insert. The **Link Info** tab allows you to choose the link type and set the link protocol and URL. The **Target** tab is only available for the URL link type. It specifies the location where the link opens after you select it.
	Unlink		Delete a link in your email or document. When you place the cursor on a link, the **Unlink** button on the toolbar becomes active. Select the button to remove the link and make it plain text.
	Superscript		Type small letters just above the line of text.
	Subscript		Type small letters just below the line of text.
	Strikethrough		Cross out text by drawing a line through it.
	Insert Image		You can insert an image by directly copying and pasting it inline in the editor, dragging and dropping it from your desktop or local folder directly into the editor, or by typing a URL. The following formats are supported: .PNG, .JPG., or .GIF. To insert an image inline in your article: 1. Drag and drop the image or copy and paste it directly into the article.

Icon	Name	Shortcut key	Description
			2. Drag any corner of the image to resize it. To insert an image using a URL or navigating to the local image: 1. Choose Insert Image. 2. In the **Image** property dialog, choose from the following options: • Select **Browse** to navigate to the image on your computer. • Or specify the web address of the image, and properties to define how the image appears in the email or article. **Note:** ◦ If the image is located on the external server, use the full absolute path. ◦ If the image is located on a local server, you can use a relative path. ◦ If you want the image to be linked to a target, add a URL for the image. ◦ You can also specify if you want the targeted page to open in a new window, topmost window, same window, or parent window.
	Left to Right		Change the text to left-to-right for content such as a paragraph, header, table, or list. Commonly used for bi-directional language content. This setting is the default setting.
	Right to Left		Change the text to right-to-left for content such as a paragraph, header, table, or list. Commonly used for bi-directional language content. The default setting is left-to-right.
	Undo Typing		Undo changes you made to the content.
	Redo Typing		Redo changes you made to the content.
	Clear All Formatting		Remove all formatting from a selection of text, leaving only the normal, unformatted text.
	Add a Table		Add a table to your content. After adding a table, you can do any of the following: • Resize table columns by clicking and dragging your mouse to resize to the columns to the desired width.

Icon	Name	Shortcut key	Description
			• Select one or several cells within a table and apply specific formatting, add links to the selection, or cut, copy, or paste entire rows or columns. • Right-click to access the properties. This setting supports features such as cell type, width and height, word wrapping, alignment, merging, and splitting cells horizontally and vertically, inserting or deleting rows and columns, row and column span, and cell and border color.
•••	Expand Toolbar		Appears when the toolbar is collapsed. Select to expand the toolbar and make all options visible.
	Copy to Clipboard		Copies entire text in a note to the clipboard so that you can paste the contents in another location.

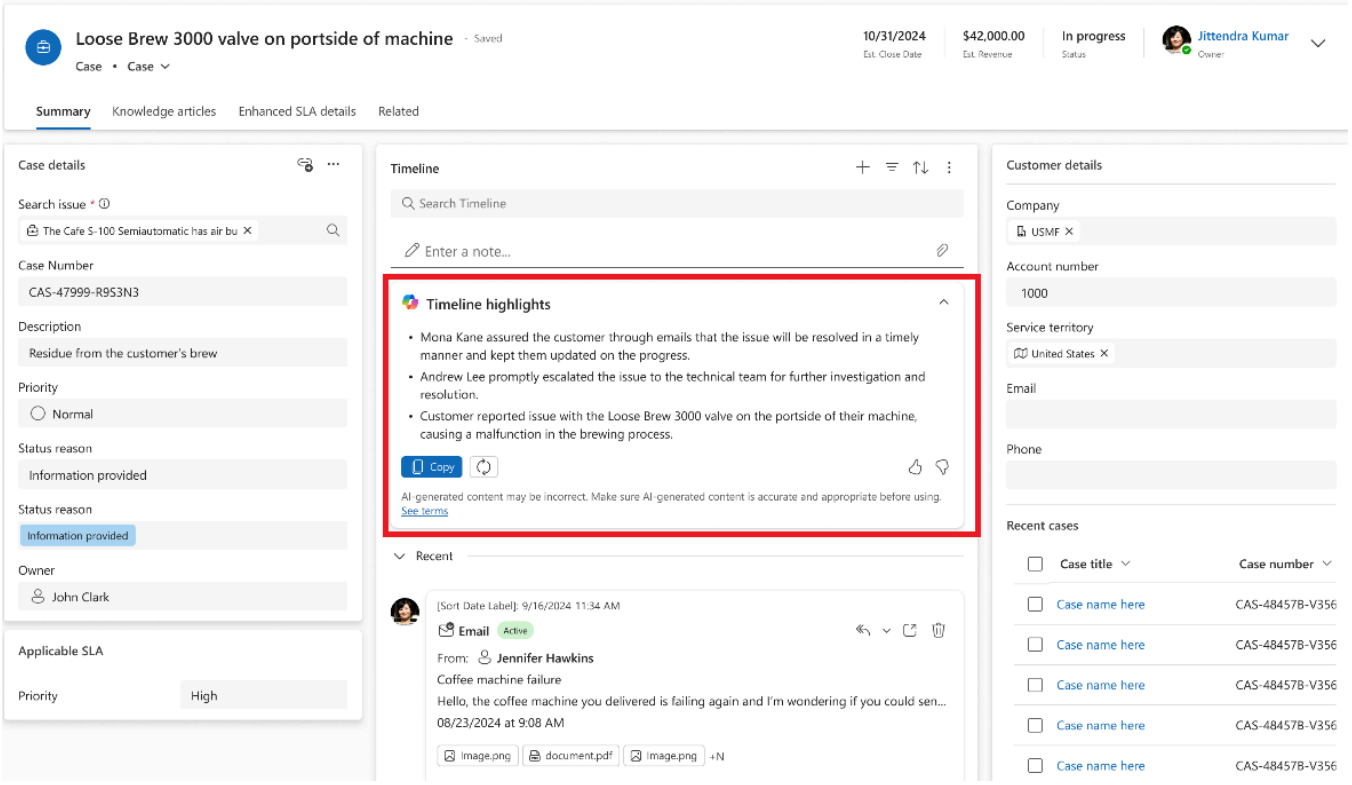
The rich text editor is enabled by default. Your system administrator can enable or disable the editor for you.

Use timeline highlights powered by generative AI

If timeline highlights are enabled on your app, Copilot creates an AI-generated, bulleted list that populates at the top of the timeline. You can use timeline highlights to quickly get caught up on a record's status.

The highlights include a record's main activities, such as details about who you or the previous user interacted with, the reported issue, and actions taken to try to resolve the issue.

You can copy the timeline highlights by selecting **Copy**.

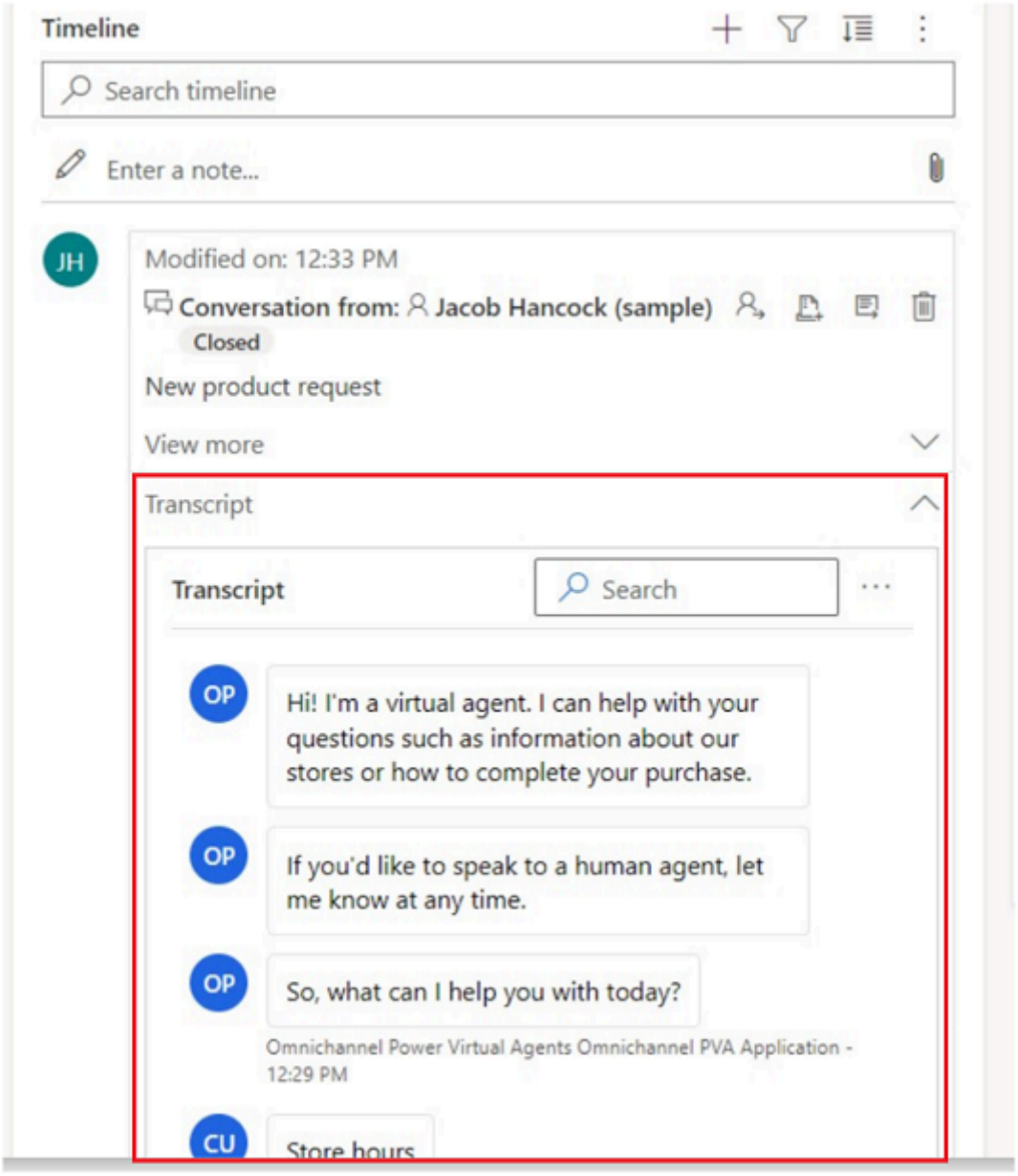


> **Important**
>
> This feature is only available with Power Platform environments with Dataverse that have the **Enable Dynamics 365 apps** setting enabled.

View a conversation transcript

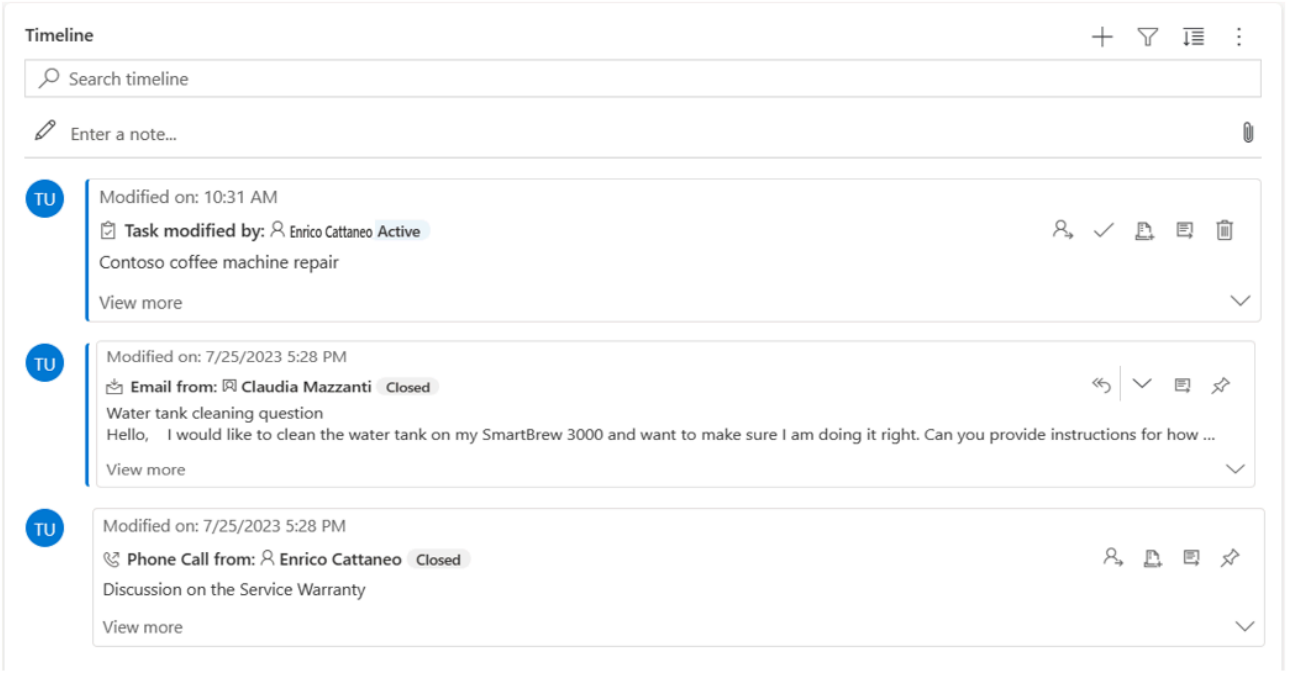
In the timeline, you can view the transcripts of completed chats you had in conversation channels. To view these transcripts, you must have prvReadmsdyn_transcript permission. More information: Roles and permissions

To view a conversation transcript, select the **View more** link in the row for which you want to view the transcript. The row expands and the entire transcript is displayed inline.



View what you missed in timeline

When you access the timeline after being away, activities that you missed display at the top of the timeline, denoted with a blue vertical line next to them. You can quickly open the activities or scroll past them.



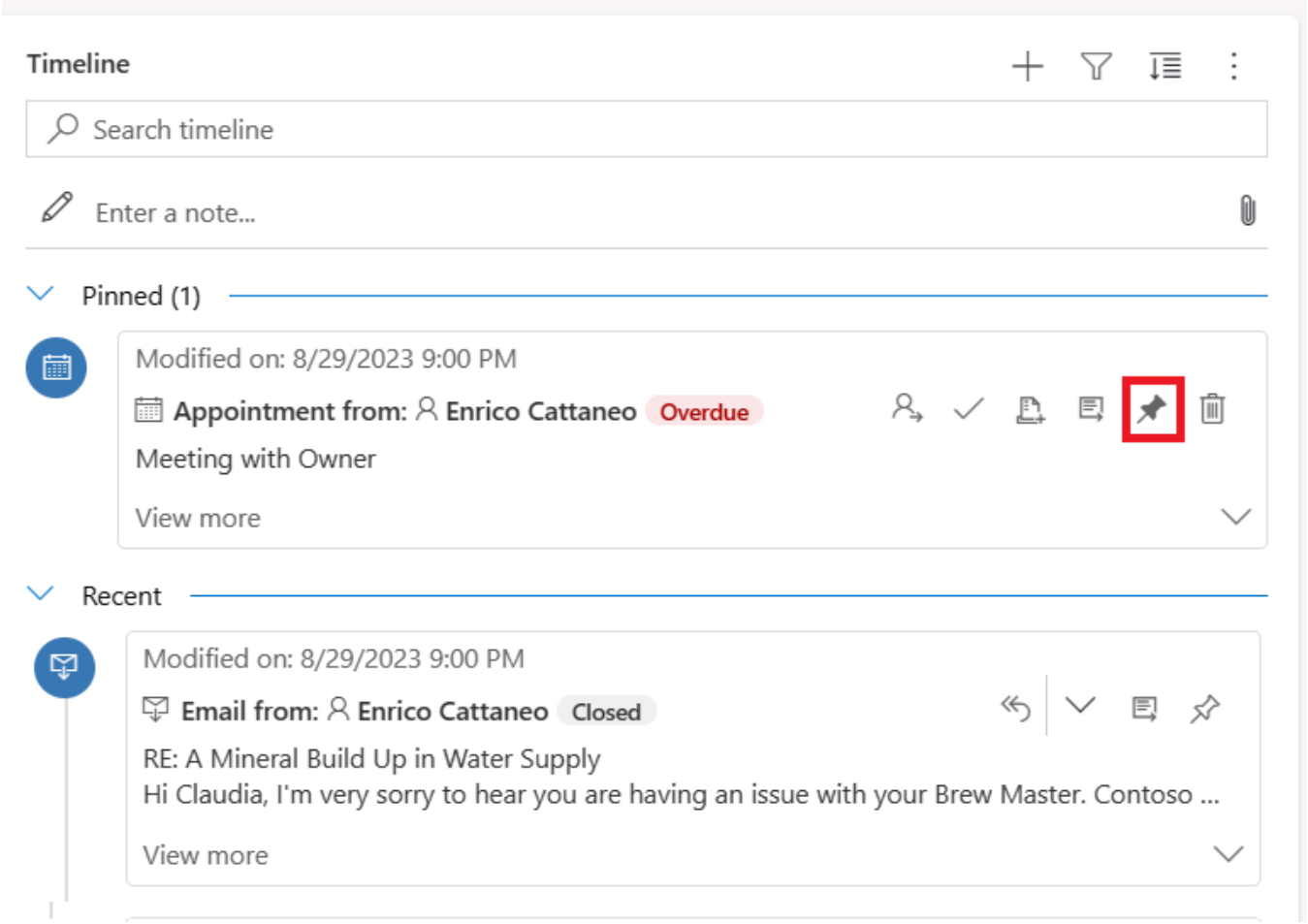
Pin or unpin records in timeline

If your administrator has enabled the pin setting and you have write/delete access, you can quickly access records, such as activities, by keeping them at the top of the timeline. Pinning a record eliminates the need to scroll through every record to find the one you need.

> **Note**
>
> If you have a system administrator or basic user role, the pin and unpin settings work by default. If you don't have either of these roles, you must have Create/Read/Write privileges to be added to the msdyn_customcontrolextendedsettings table for these settings to work.

By default, you can pin notes. The activities that you can pin depend on how your administrator has set up the feature. More information: Let users pin and unpin activities to the top of timeline.

Pin a record by selecting the pin icon next to it. This setting moves it to the top of the timeline in the **Pinned** section, where it can be quickly accessed. The Pinned section also lists the number of records that are pinned. You can pin a maximum of 15 records. Records that are pinned will stay at the top of the timeline for a year unless you unpin them.



To unpin a record, select the pin icon next to it. Unpinning it moves it out of the **Pinned** section and back to where it originally appeared in the list of records in the timeline.

Bookmarks

Bookmarks are a quick way for the user to set, and switch between, different sets of filters. Bookmarks are enabled by default. If necessary, you can disable them from the Advanced section.

> **Note**
>
> Bookmarks feature replaces **Remember my filters** button. Use the steps in the following section to set up your user filters.

Create a bookmark

1. Open the filter pane, and select the filters that will be associated with the new bookmark. You may have no filters selected.

2. Select the bookmark icon in timeline's action bar.

3. Select **Add bookmark**.
4. Enter a bookmark name. Optionally, set it as a default bookmark.
5. Select **Save**.

Selecting the bookmark icon again will now display a flyout of your created bookmarks. You can apply a bookmark by selecting it from this list, or edit it using the **More options** button on the right side of the bookmark.

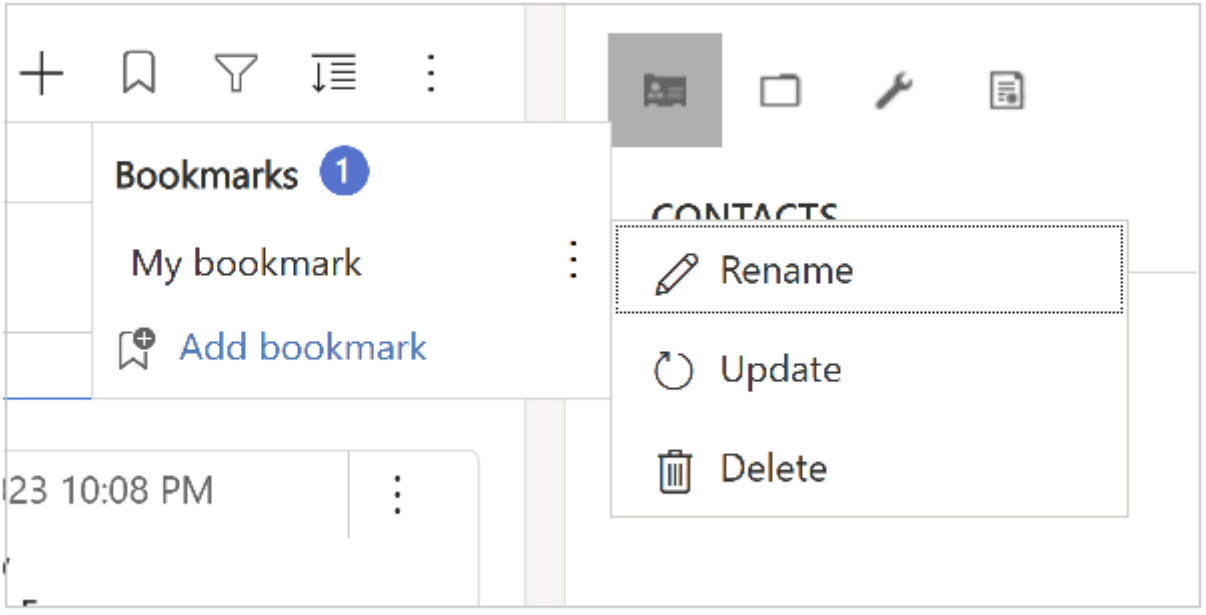
If a bookmark is selected, its filters are applied to the timeline, and is indicated by a filled bookmark icon in the action bar. Hover over the icon to see the name of the bookmark, or select the icon. The selected bookmark has a grey background color.

> **Note**
>
> You can have a maximum of 15 bookmarks.

Edit a bookmark

Any bookmark can be renamed, updated, or deleted from the **More options** menu.

> **Note**
>
> Selecting **Delete** directly deletes the bookmark.



Updating a bookmark will change its filters to the currently selected filters.

You may also set this bookmark as a default for the current form or all forms.

Set a default bookmark

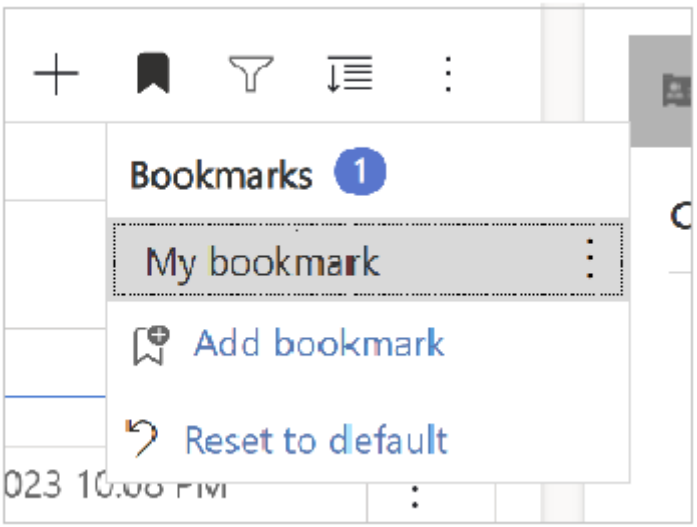
When creating or updating a bookmark, you may select the **Set as default bookmark** checkbox, with the option to apply this change for the current form or globally.

A default bookmark has a star icon ☆ next to its name, and is selected automatically when the timeline loads, overriding any admin filters set for timeline.

> **Note**
>
> You may set a bookmark as a form default for multiple forms; however, it can't be both a form default and a global default. A form-default bookmark will override a global one.

Reset to the default view

If a nondefault bookmark is selected, or if other filters are applied, a reset to default button appears in the bookmark flyout. This applies the **default bookmark** to timeline, if one exists, otherwise **clearing all filters**.



Bookmark filters can't be applied

If a bookmark's filters can't be applied (for example, a bookmark that filters for notes while timeline has notes disabled), then an exclamation mark icon is displayed next to its name.

See also

Set up timeline control

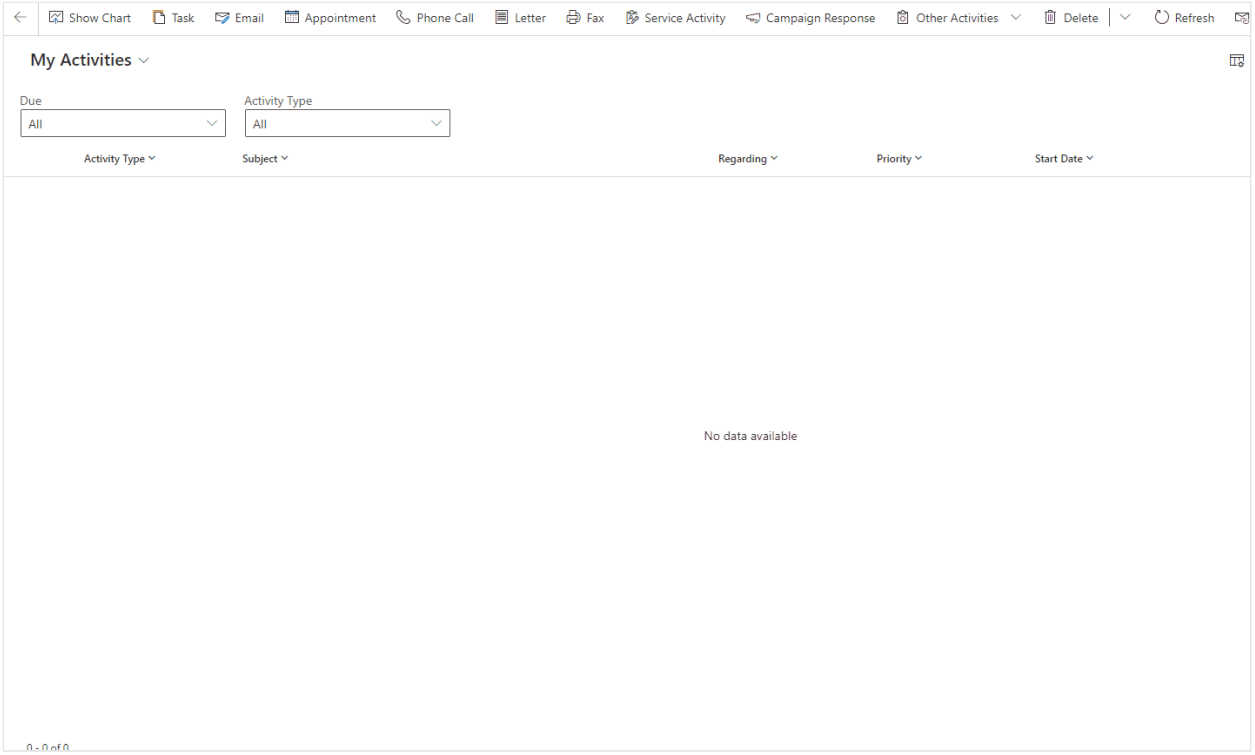
FAQs for timeline control

FAQs about Activities and the Timeline Wall

Create and join a Teams meeting from an appointment

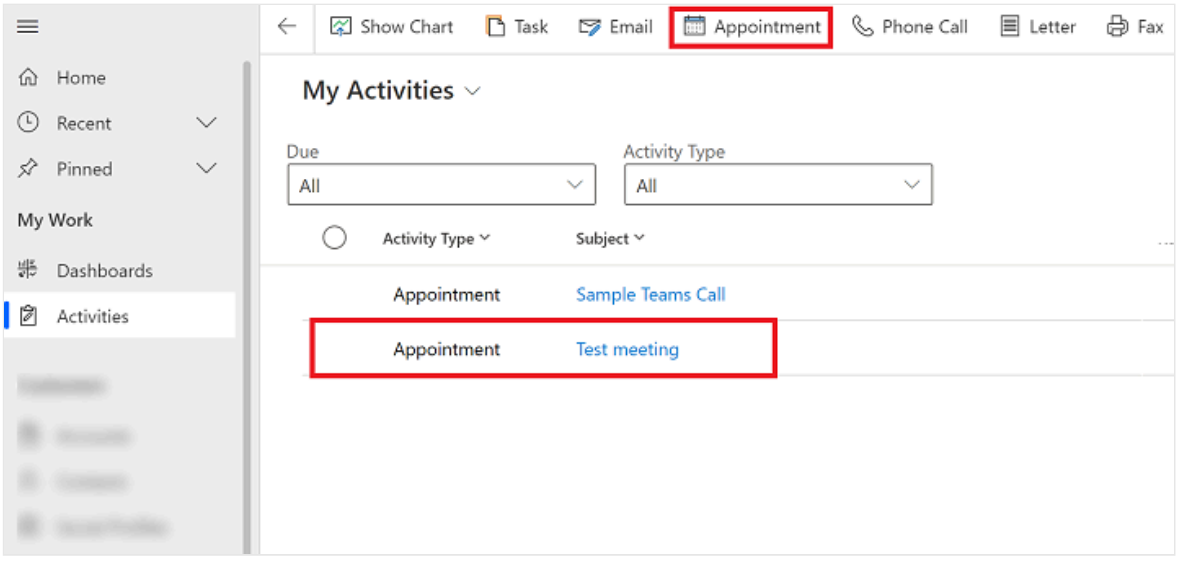
Article • 10/25/2022

Now you can create and join a Teams meeting from an appointment in your model-driven app.



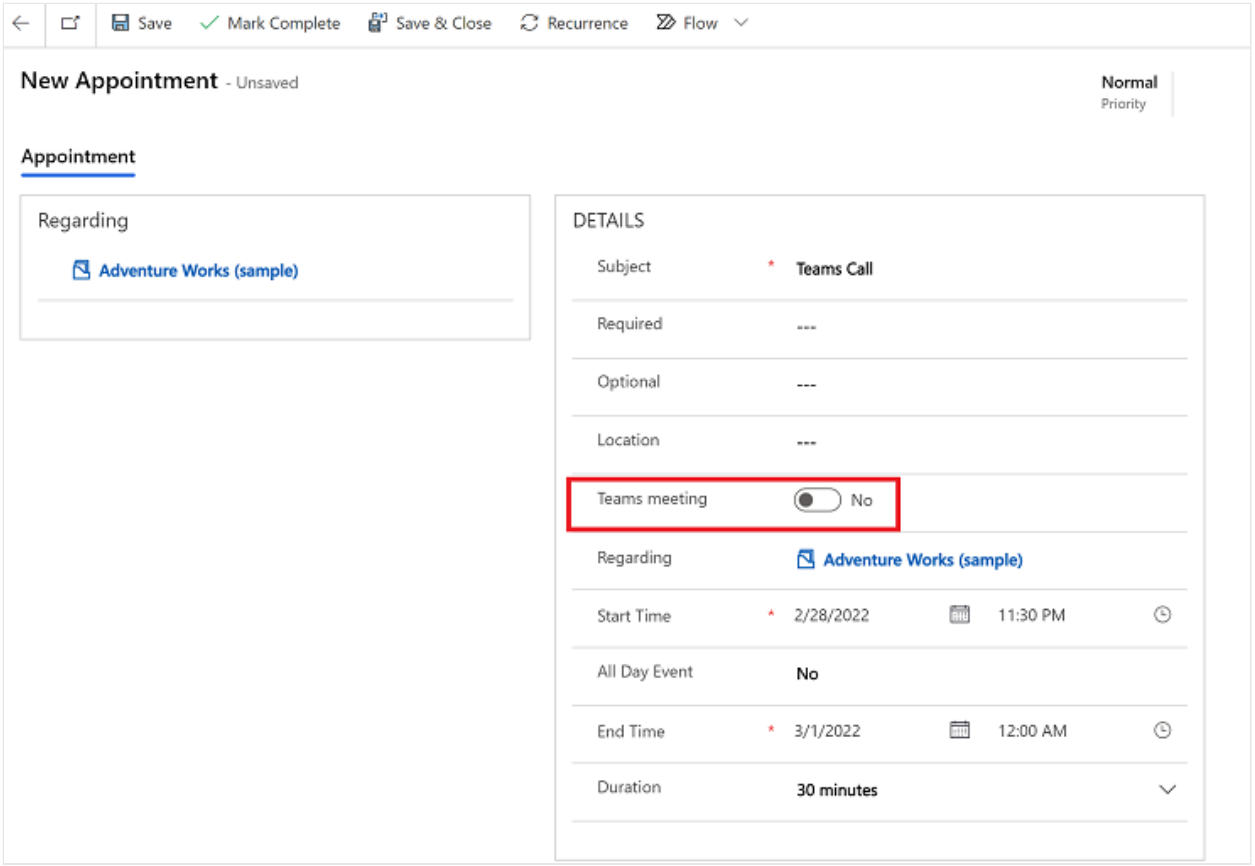
Add a Teams meeting to an appointment

1. From the left navigation pane, select **Activities**.
2. From the list of activities, select an existing appointment or select **Appointment** on the command bar to create a new one.



3. Enter the required information and in the **Details** section, set the **Teams meeting** toggle to, **Yes**.

> **Important**
>
> For the join link to be created successfully, the user enabling the meeting for Teams needs to be the meeting organizer. The organizer field is not exposed by default on the appointment form but the value defaults to the user who creates the meeting.



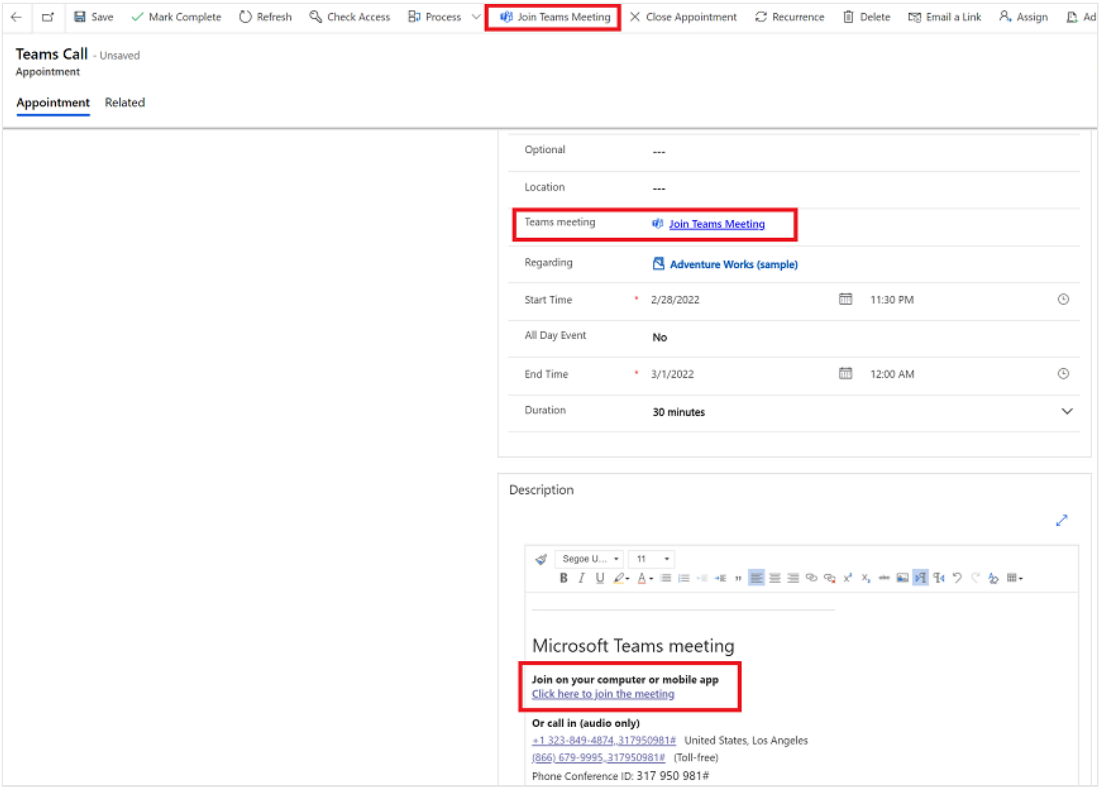
4. When you're done, select **Save**.

> **Note**

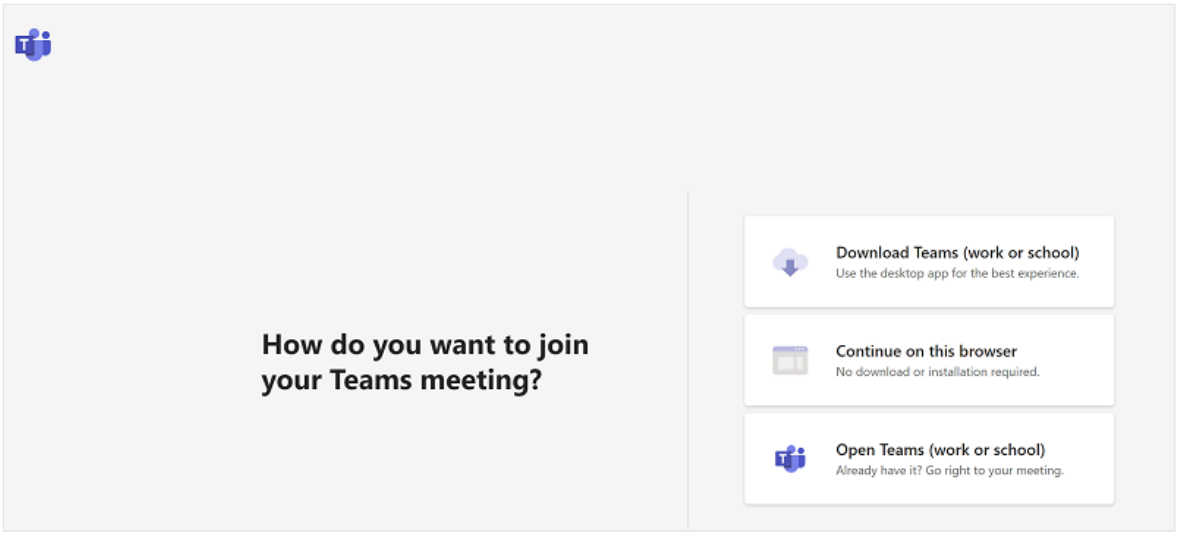
It may take a couple of minutes for the meeting to sync with Outlook and for the meeting link to appear in the appoitnment.

Join a Teams meeting from an appointment

1. Open an appointment activity.
2. Choose one of the following ways to join the meeting:
    - On the command bar select, **Join Teams Meeting**.
    - From the **Details** section, select the **Join Teams Meeting** link.
    - Scroll down the **Description** section of the appointment and select, **Click here to join the meeting**.



3. Follow the instructions on your screen and choose how you want to join the Teams meeting.



> **Note**
>
> For the meeting to synchronize from Dynamics 365 to the user's calendar in Microsoft Outlook and Teams, the user needs to be configured for appointment, contact, and task synchronization.

Feedback

Was this page helpful? Yes No

Provide product feedback

Known issues about activities

Summarize this article for me

Learn about known issues that may occur with activities in model-driven apps.

- On appointment grids, all-day appointments may show incorrect values for **Start Time** and **End Time**. This issue is limited to the grid control showing appointment records, as forms and calendar control show correct **Start Time** and **End Time** for all-day appointments.

  To fix this issue, use the calendar control as the default control to view appointments.

- The process that checks for conflicts for the people and resources on an appointment is triggered only when a valid Dynamics 365 solution is installed on the environment, as this requires a few scheduling-related tables to be a part of the environment. Appointments created in an environment without a relevant Dynamics 365 solution installed have an **Open** status, rather than the **Scheduled** status. For more information on the scheduling tables and the license required, see Restricted tables for create, update, and delete operations.

- When you use the calendar control view for appointments and navigate through time periods (like moving from Day to Week view or using the navigation arrows), selecting **Edit filters** and then **Apply** can result in duplicate Start Time and End Time filter conditions. This duplication causes appointments not to display in the view. To resolve this issue, select **Edit filters**, manually remove one set of the duplicated Start Time and End Time filters, and then select **Apply**. Alternatively, work in the grid view instead of the calendar view.

- Error message `Unable to find one-to-many relationship, entity: [entity name], referencing entity: activityparty`

  This transient error appears when saving records with relationships to activity parties. The root cause of the issue is that schema information about the relationship is not loaded when you save the record. The work around is to try to save the record again because the schema information will be loaded eventually. If this error occurs due to custom scripts on the page, the script developer should catch the error and retry after a short delay.

Last updated on 02/16/2026

Understand the email experience

Article • 02/15/2022

Having the ability to communicate effectively with customers can lead to increased sales and customer satisfaction. The email capability in customer engagement apps empowers users with the ability to respond and provide timely customer interaction effortlessly through quick email access with enhanced features and functionality. You can compose, reply, and review email, as well as quickly attach and send images that add value and support to customer communication using streamlined functionality. You can also ensure interactions between you and your customers are consistent and professional by using email templates with rich text and images.

You can work in either the **navigate-to** email or the **contextual** email experience. The difference between the navigate-to vs. contextual email experience is where the email displays on your screen. Both experiences use the same **Enhanced Email** form, which is enabled by default out-of-the-box (OOTB).

> **Important**
>
> - **Contextual** email is also referred to as enhanced email. Please be aware that there isn't any special functionality associated with the contextual email experience, other than it is a pop-up experience.

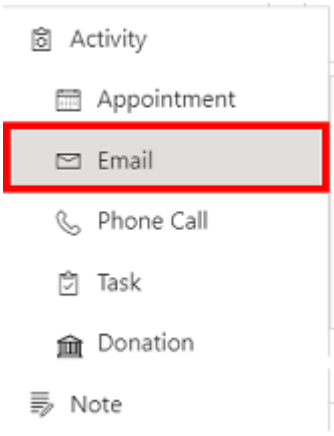
- **Navigate-to email experience**: Always displays in a new screen.
- **Contextual email experience**: Can only be initiated from the timeline, displays email in a pop-up screen, and allows you to:
  - Move to different pages without losing email content.
  - Minimize the email window to return to the records you were working on.
  - Simultaneously view, compose, and reply to five emails in pop-up windows when enabled.

The following highlights the difference between the "navigate-to" and contextual email experience in the timeline based on how your system administrator has configured your email:

1. Select **Create a timeline record**.

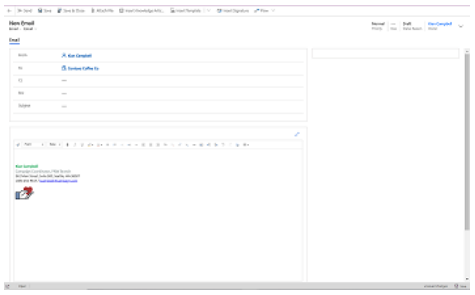
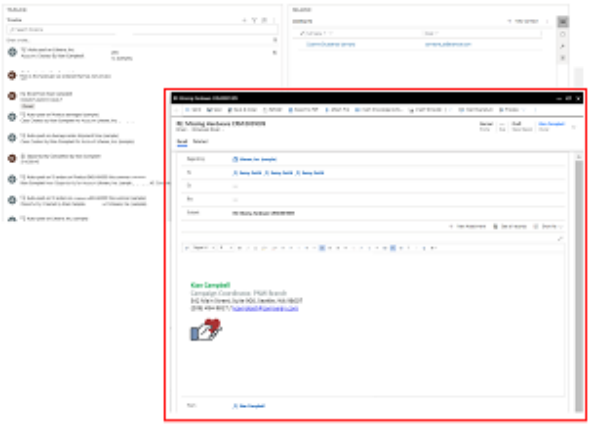


2. Select **Email**.



Depending on the experience you are using, your email will display as follows:

Expand table

Navigate-to email experience	-	Contextual email experience
		
The "navigate-to" email will display in a new screen.		The contextual email will display in a pop-up window on your current screen.

ⓘ Important

- The pop-up window only opens when the height and width of your screen size is at least 400 x 650 pixels or greater. If lower, "navigate-to" email will be the default experience.
- The contextual email experience is available only for timeline.

ⓘ Note

- If you use customized forms in email, you will still be able to see and use them. The **Enhanced Email** form showcases various email capabilities

and functionality.

For more information, see View, compose and respond to email.

See also

Set up enhanced email
Insert an email template

Feedback

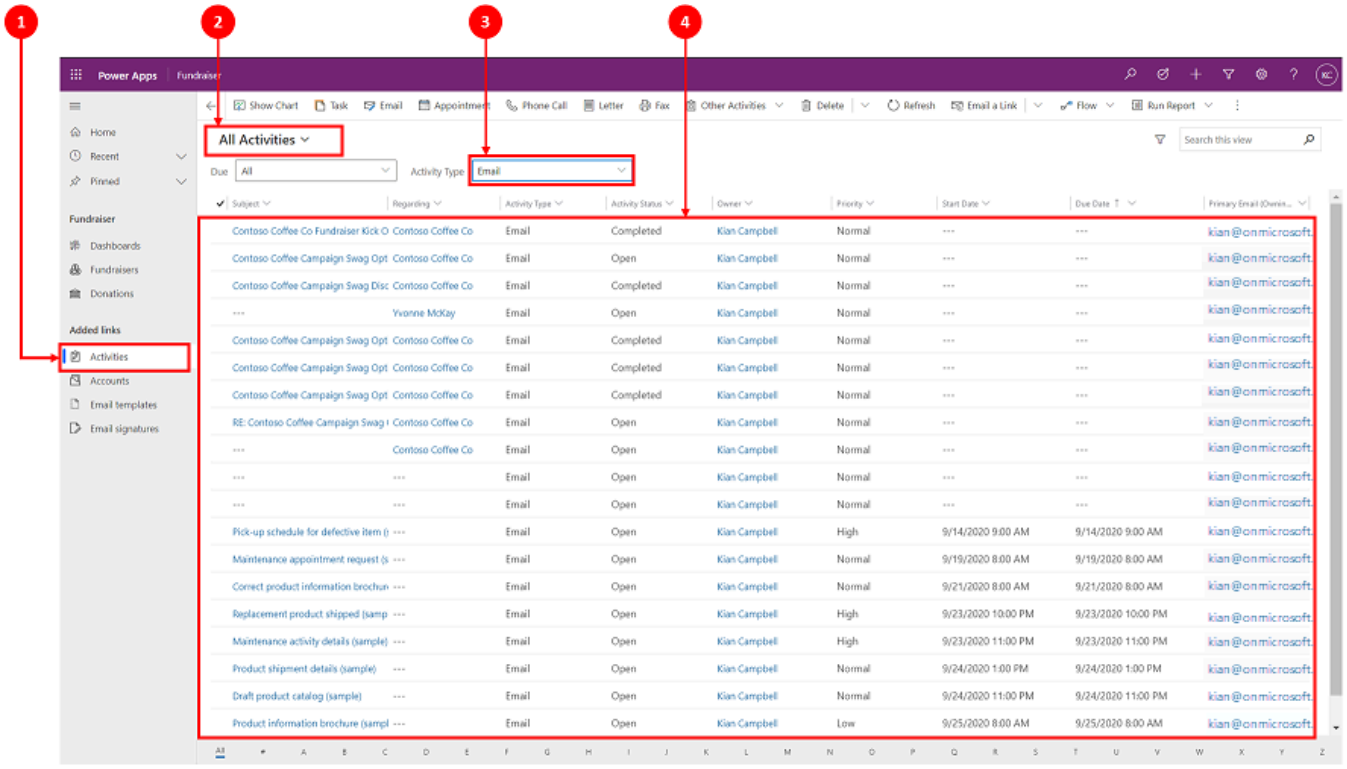
Was this page helpful? Yes No

Provide product feedback

View, compose and respond to email

Users can view, create, and send email faster with streamlined access that helps users compose, reply, and send emails using rich text editing and drag-and-drop attachment features and functionalities to embed images in emails.

Access and view emails



Legend

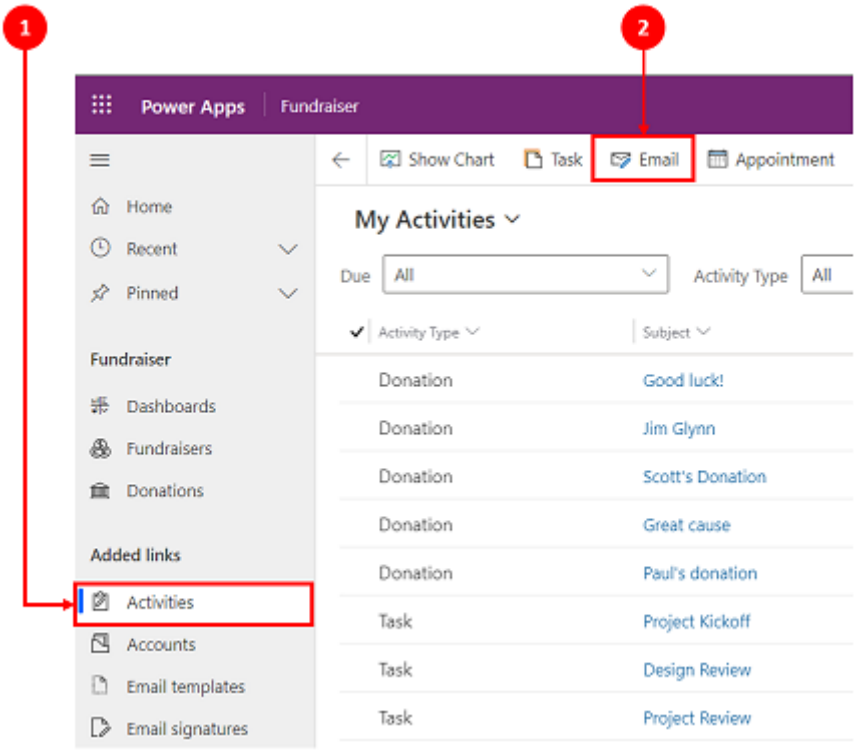
1. Select **Activities**
2. Select **All Activities**
3. Under **Activity Type** select **Email**. The email list appears in the screen.

Compose an email

Email capabilities, like rich-text editing and drag-and-drop or copy-and-paste images enriches customer interactions when composing email.

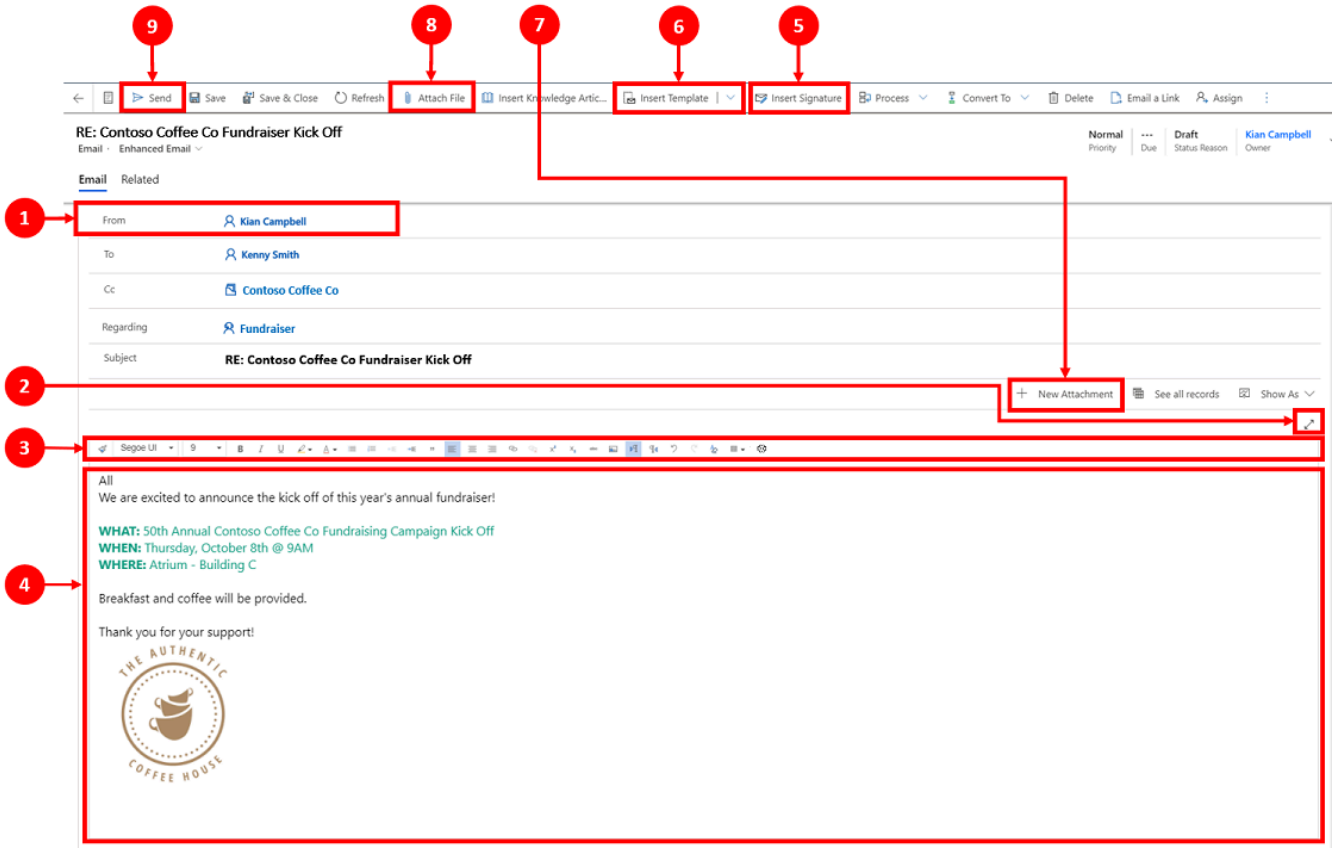
When enabled by your system administrator, you can access email from the top navigation bar.

Compose an email



1. Select **Activities**
2. On the command bar, select **Email**. An email then opens in a new window where you can begin composing.

Anatomy of an email form



Legend

1. **From**. The name displayed in the **From** field is automatically populated based on the user who is currently signed in.
2. **Expand**. The **Expand** icon allows you to compose your email in a full-screen view and minimize when done.
3. **Rich text editor**. This tool bar helps you format emails. The editor is displayed as a single line that can be expanded to view the full list of editing features by default. For more information, see Use the rich text editor toolbar in email and Accessibility shortcuts for email. **User Personalization**. The icon allows you to set the default font and font size for your emails. Once set, the **Font** and **Font Size** display these values by default.
4. **Body**. The body is where you compose and/or reply to an email.
5. **Insert Signature**. Use this command to personalize your message.
6. **Insert Template**. This function is used to apply an email template.
   For more information, see Insert an email template.

   > **Note**
   >
   > The **To** box must contain a recipient to select a template.

7. **New Attachment**. This command is used to add a file to your email.
8. **Attach File**. This command uses the 'plus' icon functionality to add attachments.

   > **Note**
   >
   > After you save your email, you can use Attach file and New Attachments interchangeably.

9. **Send**. Select this icon to **Send** your email.

   > **Important**
   >
   > - The **From** and **To** fields are automatically populated based on the user and the account and contact of the original record.

View email sentiment (preview)

[This article is prerelease documentation and is subject to change.]

When you get an incoming email from a customer, the customer sentiment card is displayed on the email form.

When enabled by your system administrator, the customer sentiment card displays the following features:

- **Current sentiment**: The customer's current sentiment. A sentiment intensity indicator in the corresponding colors is also displayed.
- **Sentiment timeline**: A visual representation that shows how the customer sentiment in the emails received changes across a period.
  The timeline displays indicators for up to six customer emails, including the current email. Customer service representatives can select historical sentiment and then select **View email**. The system shows the relevant email for that period and sentiment in a new tab.
- The **Inbox** view also displays the sentiment with the sentiment intensity indicator.

> ⓘ **Note**
>
> For emails received before the feature is enabled, the system doesn't display the sentiment.

> ⓘ **Important**
>
> - This is a production-ready preview feature.
> - Production-ready previews are subject to supplemental terms of use.

Reply to an email

The way you can reply to an email depends on where you are in your app.

When enabled by your system administrator, the email option displays in the top navigation bar and in the activities command bar.

> ⓘ **Note**
>
> The best performance is achieved when the HTML content size is 1 MB or less. When your HTML content size exceeds 1 MB, you might notice slower response times for loading and editing content. By default, image content is referenced from the content HTML but isn't

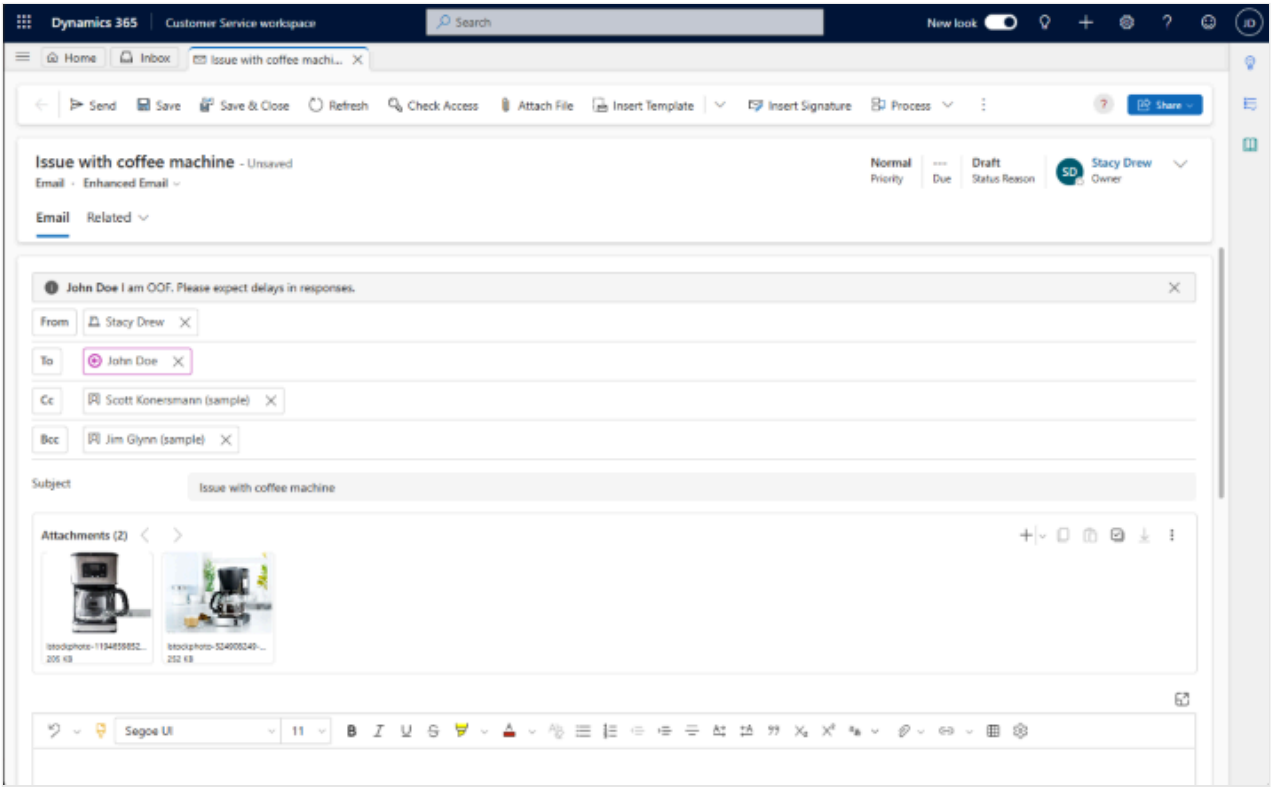
stored as part of the HTML content, so in the default configuration, images don't negatively impact performance.

Enhanced recipient handling

Users can perform the following actions only if the administrator added the **Email Recipient control** component to an email form:

- Drag and drop recipient names across **To**, **CC**, and **BCC** fields.
- View the recipient's email address along with their names.
- View the recipients' presence status and out-of-office messages.
- Resolve an unresolved email address faster. In the email editor, if there's an unresolved email address, the application displays a banner message and the **Review** option to map the email. Select the button to view and map an unresolved email address to an existing record.

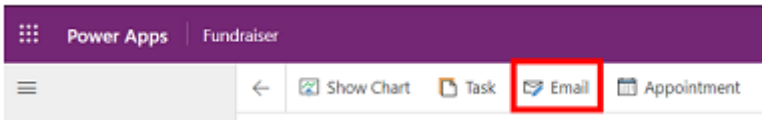
  > **Note**
  >
  > You can't resolve an unresolved email address for a read-only email.



Access email

When you access email, there are a couple of options you can use.

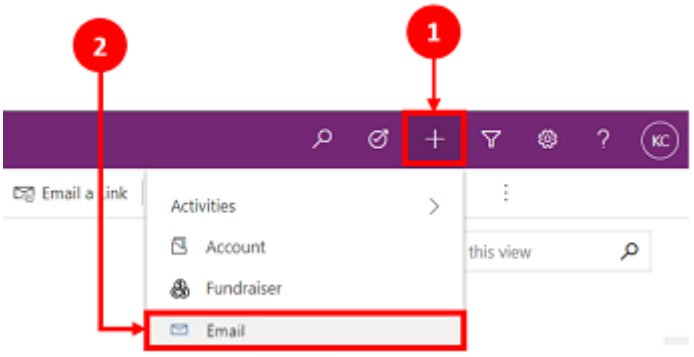
- **Option 1:**



  - Select **Email** on the command bar.

> ⓘ **Note**
>
> Email option will only appear in the drop-down list and the command bar when enabled by your system administrator.

- **Option 2:**



1. From the navigation bar, select the plus icon.
2. Select **Email** from the drop-down menu.

Set the default font and font size

You can set the default font and font size for your email text. Perform the following steps:

- In the email editor, in the Rich text editor tool bar, select .
- In **User Personalization** specify the **Font** and **Font Size**.
- Select **Ok**. The application defaults the values specified in the Font and Font Size fields of the rich text editor toolbar.

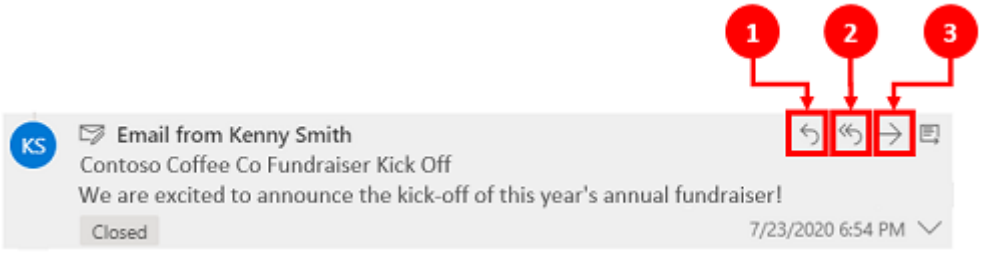
If your administrator set a default font and font size, the application displays the set values in the Font and Font Size fields of the rich text editor toolbar.

Reply to email in timeline

When you reply to an email in the timeline, the command bar in the top-right corner provides you with reply options. When you select an option, your email automatically is set and opens in

that state ready for you to begin your message.

You can use the following command options when working with emails in the timeline.



1. **Reply**. Use this command to reply directly the sender of the email you received.
2. **Reply All**. Use this command to reply to everyone on an email you received.
3. **Forward**. Use this command to forward the email to someone else.

Download email

Select **Download** to download emails as an .eml file.

Last updated on 02/04/2026

Attach a file to email

Article • 02/01/2024

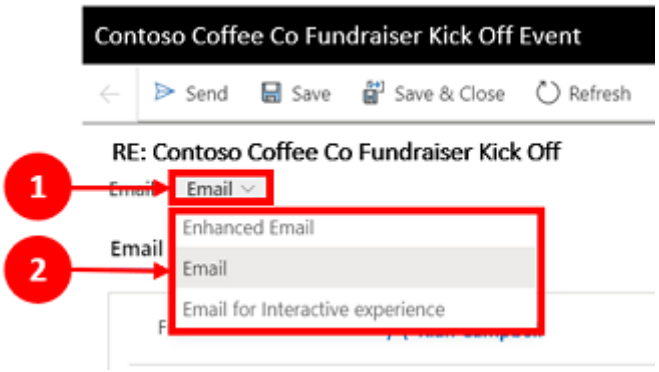
The option to attach files to an email is a useful feature when you need to share additional information. Use the **Attach file** button on the command menu or the **New Attachments** button in the email body to attach files to the email.

> **Important**
>
> - The default file size limit for an email file attachment is 5 MB. The size limitations on email file attachments can be increased by your system administrators.
> - You can't attach files to sent email messages after they have been attempted for delivery through server-side synchronization.

Work with attachments

You can attach an unlimited number of files to an email; however, there are file size limitations that are managed by your system administrator.

Enhanced email is the default email form, however you can switch forms by doing the following:



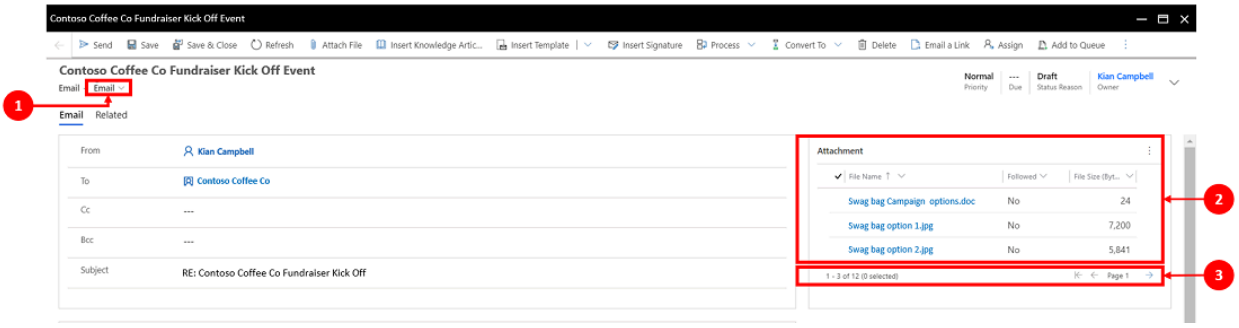
1. Under the email subject name, select on the caret (v) next to **Email**
2. A dropdown appears with a list of email form options you can choose from.

File attachments display differently, depending on the email form you're working in. You can switch email forms without losing information.

If you specify "attach" or "attachments" in the subject or the body of the email and you select **Send** without attaching a file, the application displays a warning message reminding you to attach a file.

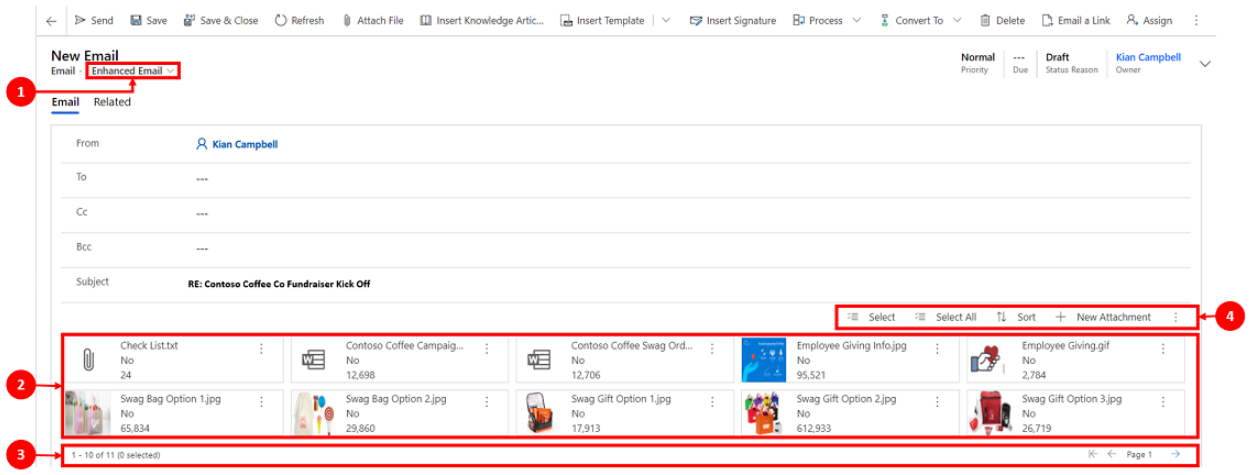
> **ⓘ Important**
>
> - The warning message reminding you to attach a file is an early access feature.
> - Early access features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Email form file attachment



1. **Email**. This display confirms which email form you're working in.
2. **Attachment**. This section displays a list of file attachments in a vertical view and only displays the first three attachments.
3. **File count**. This displays when you have more files than what can be viewed and allows you to see the total number of attached files on the left side under the last attachment with the option to page over to see more files.

Enhanced email form file attachment



1. **Enhanced Email**. This display confirms you're working in the **Enhanced email** form.
2. **Attached files**. This section displays all your email attachments in a tile view above your message horizontally. The defaulted setting for the maximum number of files that can display is five attachments before displaying the page forward functionality.
3. **File count**. This displays when you have files that occupy more than two rows and allows you to see the total number of attached files on the left side under the last attachment with the option to page over to see more files.
4. **Attachment Commands**. This allows you to use the select, select all, sort, delete, and download functionality when working with your file attachments.

Work with the attachment command bar

When you've attached one or more files to a message, you can manage them using the file attachment commands.

Select commands

Select commands allow you to select individual files or multiple files at once.



Legend

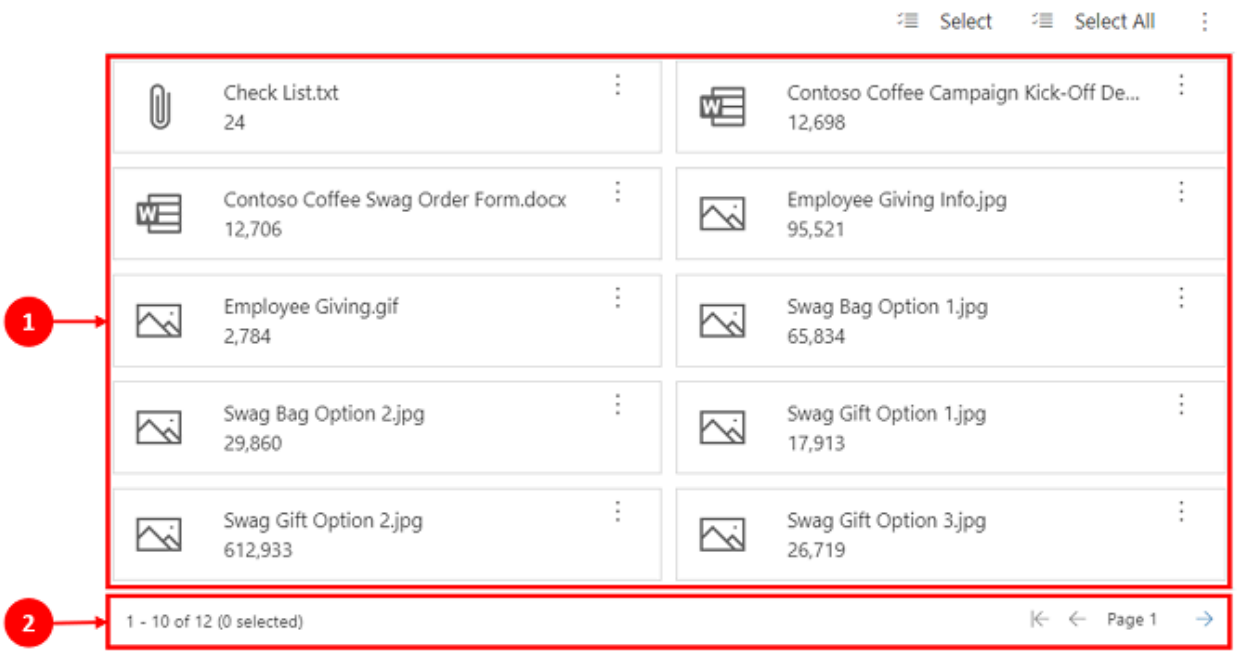
1. **Select All**. This command is located on the right, above your attached files and allows you to select all file attachments at once.

   > **Note**
   >
   > **Select All** only selects files that are visible. If you have multiple pages of file attachments, you will need to go to that page and select those files independently.

2. **Tile Color Transparency**. The background color of selected file attachments changes to light grey to identify which files have been selected.

3. **Check box** You can also verify which files have been selected by viewing whether the checkbox in the top-right corner of the file attachment is selected.

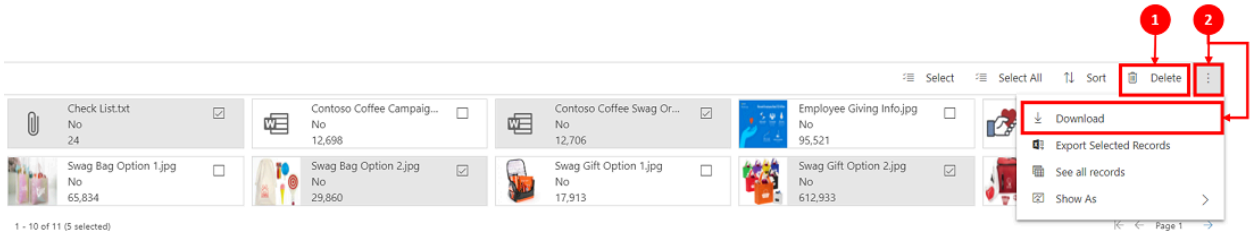
Formatting of file attachments in email can be realigned based on the device being used and format it's being worked in.



1. **Format Display** Regardless of format display, file attachments will only display the number of files that are enabled by your system administrator.
2. **Page view**. The page view displays the total number of email attachments and always is displayed under the file attachments.

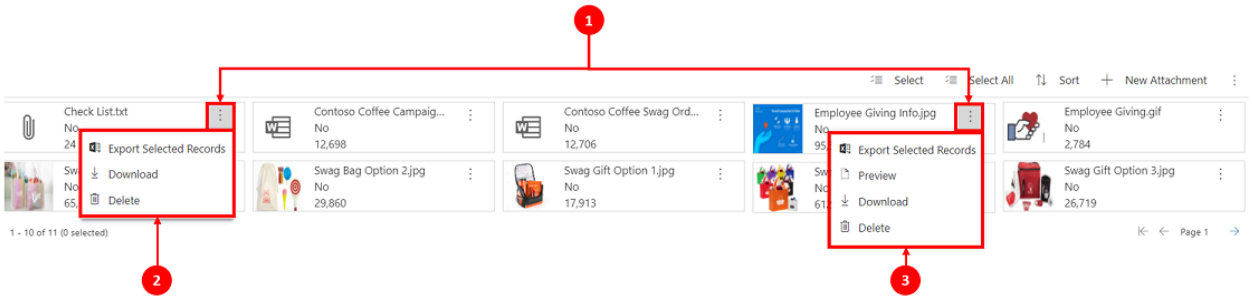
Delete and download commands

These are two of the most important commands when working with images.



1. **Delete**. This command works with the **Select** commands. You must first select a file before you can delete it. A confirmation request appears, requiring you to confirm you want to delete the file(s) you've selected.
2. **Download**. This command also works with the **Select** commands and your browser download settings, allowing you to download an individual file or multiple files at once.

Preview image attachments



1. **More options**. This command provides a list of actions for that file type.
2. **Non-preview image**. Only non-preview image types allow you to either **export selected records**, **download**, or **delete** the attached file.
3. **Preview image types**. Allow you to **export selected record**, **preview**, **download**, or **delete** the attached file. When you select the **preview** option, a pop-up window appears, displaying the image with the image name listed below.

Supported email file attachments

The following is a list of file image types that are supported:

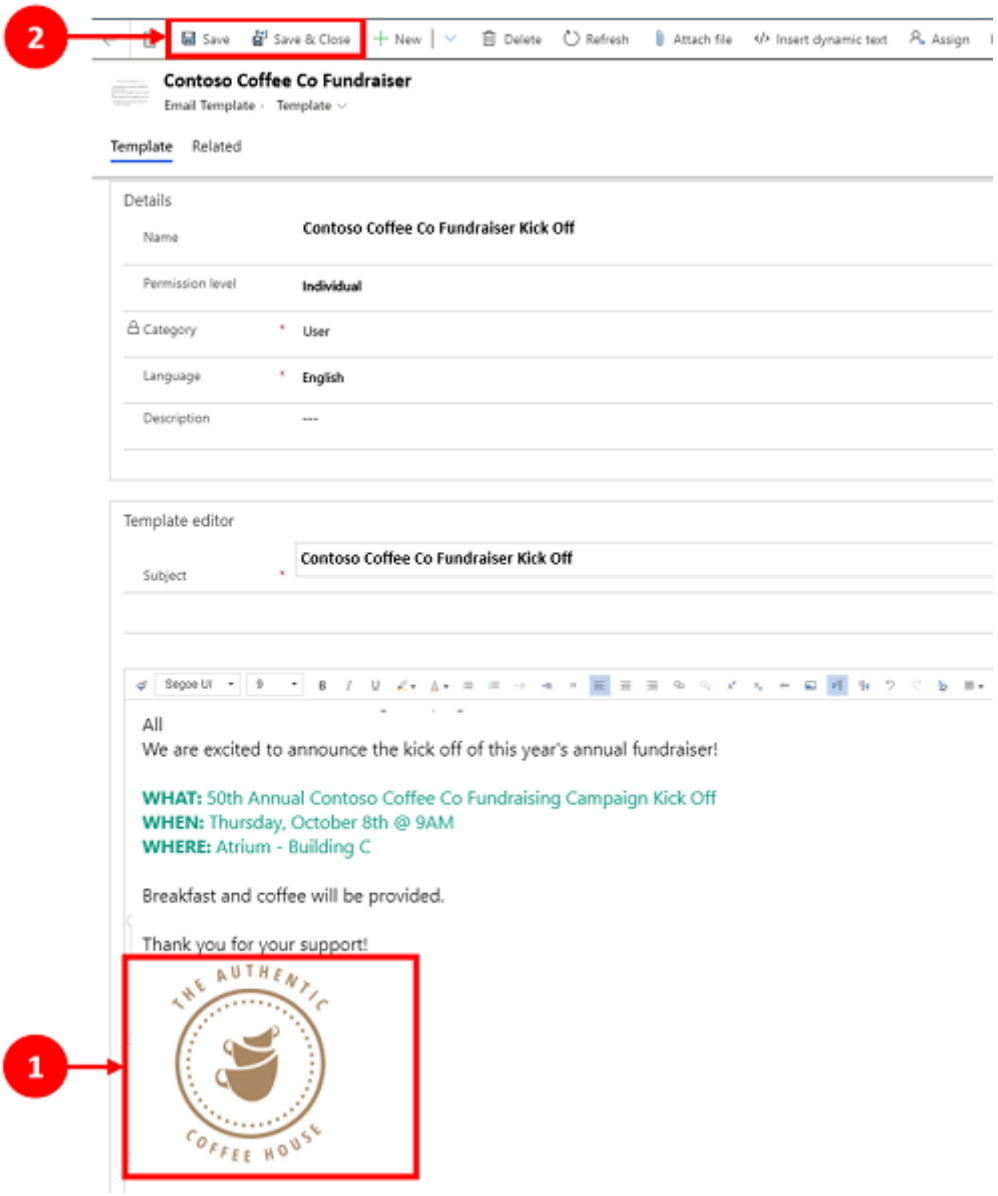
Expand table

image type	supports preview	supported inline (in editor)
BMP.bmp	Yes	Yes
GIF.gif	Yes	Yes
ICO.ico	Yes	Yes
JPG.jpg	Yes	Yes
PNG.png	Yes	Yes
SVG.svg	Yes	Yes
PDF.pdf	Yes (preview not supported on IE)	No

View and insert images in email

The use of visual aids helps users convey a tremendous amount of information with just a glance. Having the ability to view and insert inline images in email using the drag-and-drop or copy-and-paste functionality is a powerful tool for users. Additionally, you can browse to a file or reference an online URL using the image toolbar button.

You can drag-and-drop or copy-and-paste images into the email.



1. **Email images** You can adjust the image size.

2. **Save**. When you complete your edits, select **Save** or **Save & Close**.

> **Note**
>
> There's a 1 MB file size limitation when adding an inline image to an email for email templates. (For new or replied emails, there isn't a limit to the email size (it becomes part of the total email size).

Feedback

Was this page helpful? Yes No

Provide product feedback

Use the rich text editor toolbar in email

Article • 10/03/2022

The rich text editor toolbar provides features and functionality that allows you to work with rich text formats in emails to better communicate with your customers.

Formatting options

The following table describes the different formatting features and functionality options available in the rich text editor that you can use in the email form.

> **Note**
>
> You can access your browser's context menu by pressing **Ctrl** + right-click. This is useful if you need to use your browser's built-in spell checker. Otherwise, you can use the right-click to provide contextual formatting for any element you might be using.

Expand table

Icon	Name	Shortcut key	Description
	Format Painter	Ctrl+Shift+C, Ctrl+Shift+V	Apply the look of a particular section to another section.
	Font	Ctrl+Shift+F	Select your desired font. The default font is Segoe UI. **Note**: When you select any formatted content, the font name for that content displays. If your selection contains multiple fonts, the topmost font name of your selection is displayed.
	Font size	Ctrl+Shift+P	Change the size of your text. The default size is 9. **Note**: When you select any formatted content, the font size for that content displays. If your selection contains multiple font sizes, the topmost font name of your selection is displayed.
	Bold	Ctrl+B	Make your text bold.
	Italic	Ctrl+I	Italicize your text.

Icon	Name	Shortcut key	Description
	Underline	Ctrl+U	Underline your text.
	Text Highlight Color		Make your text stand out by highlighting it in a bright color.
	Font Color		Change the color of your text.
	Bullets		Create a bulleted list.
	Numbering		Create a numbered list.
	Decrease Indent		Move your paragraph closer to the margin.
	Increase Indent		Move your paragraph farther away from the margin.
	Block Quote		Apply a block-level quotation format in your content.
	Align Left	Ctrl+L	Align your content with the left margin. (Commonly used for body text to make it easier to read.)
	Align Center	Ctrl+E	Center your content on the page. (Commonly used for a formal appearance.)
	Align Right	Ctrl+R	Center your content on the page. (Commonly used for a formal appearance.)
	Link		Create a link in your document for quick access to web pages and files. Pasted or typed URL text is converted into a link. For example, "http://myexample.com" will become "http://myexample.com ". In the **Link** dialog box, choose the type of link you'd like to insert. The **Link Info** tab allows you to choose the link type as well as set the link protocol and URL. The **Target** tab is only available for the URL link type. It specifies the location where the link will open after you select it.
	Unlink		Delete a link in your email or document. When you place the cursor on a link, the **Unlink**

Icon	Name	Shortcut key	Description
			button on the toolbar becomes active. Select the button to remove the link and make it plain text.
	Superscript		Type very small letters just above the line of text.
	Subscript		Type very small letters just below the line of text.
	Strikethrough		Cross out text by drawing a line through it.
	Insert Image		You can insert an image by directly copying and pasting it inline in the editor, dragging and dropping it from your desktop or local folder directly into the editor, or by typing a URL. The following formats are supported: .PNG, .JPG., or .GIF. To insert an image inline in your article: 1. Drag and drop the image or copy and paste it directly into the article. 2. Drag any corner of the image to resize it. To insert an image using a URL or navigating to the local image: 1. Choose Insert Image. 2. In the **Image** property dialog, choose from the following options: • Select **Browse** to navigate to the image on your computer. • Or specify the web address of the image, and properties to define how the image will appear in the email or article. **Note:** ◦ If the image is located on the external server, use the full absolute path. ◦ If the image is located on a local server, you can use a relative path. ◦ If you want the image to be linked to a target, add a URL for the image. ◦ You can also specify if you want the targeted page to open in a new window, topmost window, same window, or parent window.
	Left to Right		Change the text to left-to-right for content such as a paragraph, header, table, or list. Commonly used for bi-directional language content. This is the default setting.

Icon	Name	Shortcut key	Description
	Right to Left		Change the text to right-to-left for content such as a paragraph, header, table, or list. Commonly used for bi-directional language content. The default setting is left-to-right.
	Undo Typing		Undo changes you made to the content.
	Redo Typing		Redo changes you made to the content.
	Clear All Formatting		Remove all formatting from a selection of text, leaving only the normal, unformatted text.
	Add a Table		Add a table to your content. After adding a table, you can do any of the following: • Resize table columns by clicking and dragging your mouse to resize to the columns to the desired width. • Select one or several cells within a table and apply specific formatting, add links to the selection, or cut, copy, or paste entire rows or columns. • Right-click to access the properties. This supports features such as cell type, width and height, word wrapping, alignment, merging and splitting cells horizontally and vertically, inserting or deleting rows and columns, row and column span, and cell and border color.
	User personalization		Select the default Font Name and Font Size. After they're added, the **Font** and **Font Size** reflect these values as default font and font size.
	Expand Toolbar		Displays when the toolbar is collapsed and not all options appear. Click to expand the toolbar and make all options visible.

> **Note**
>
> You can access your browser's context menu by pressing **Ctrl** + right-click. This is useful if you need to use your browser's built-in spell checker. Otherwise, you can

use the right-click to provide contextual formatting for any element you might be using.

Feedback

Was this page helpful? Yes No

Provide product feedback

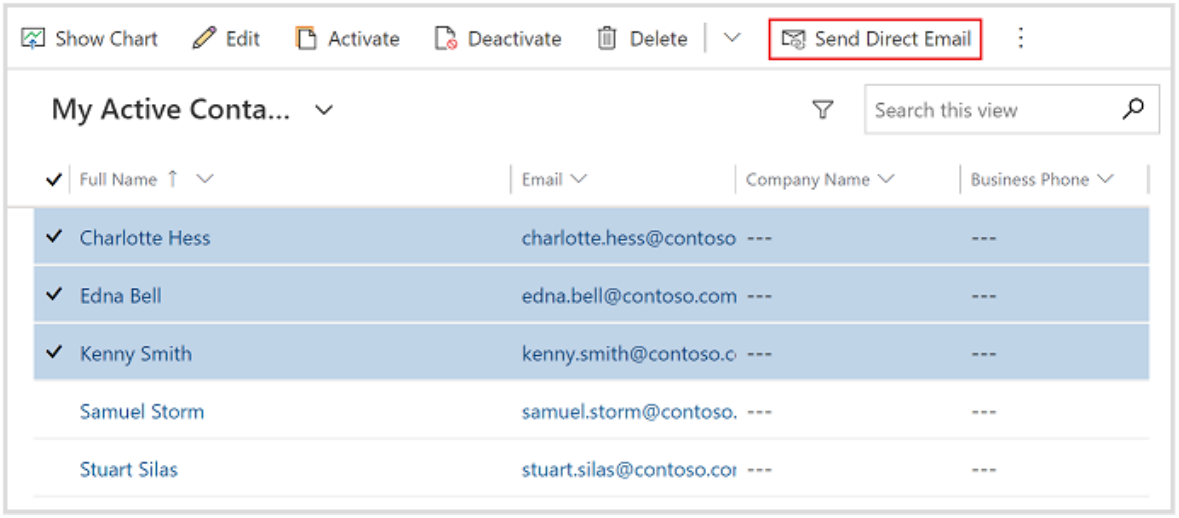
Send email to multiple recipients

Article • 02/15/2022

You can send an email to multiple recipients by using email templates. This is known as *direct* or *bulk* emailing. By default, the direct email feature is disabled. An administrator must enable the direct email feature by selecting **Yes** for **Enable Send Direct Email Action in Unified Interface for Send Email enabled tables** on the **Email** tab in the **System Settings** dialog box. More information: System Settings Email tab

To send email to multiple recipients

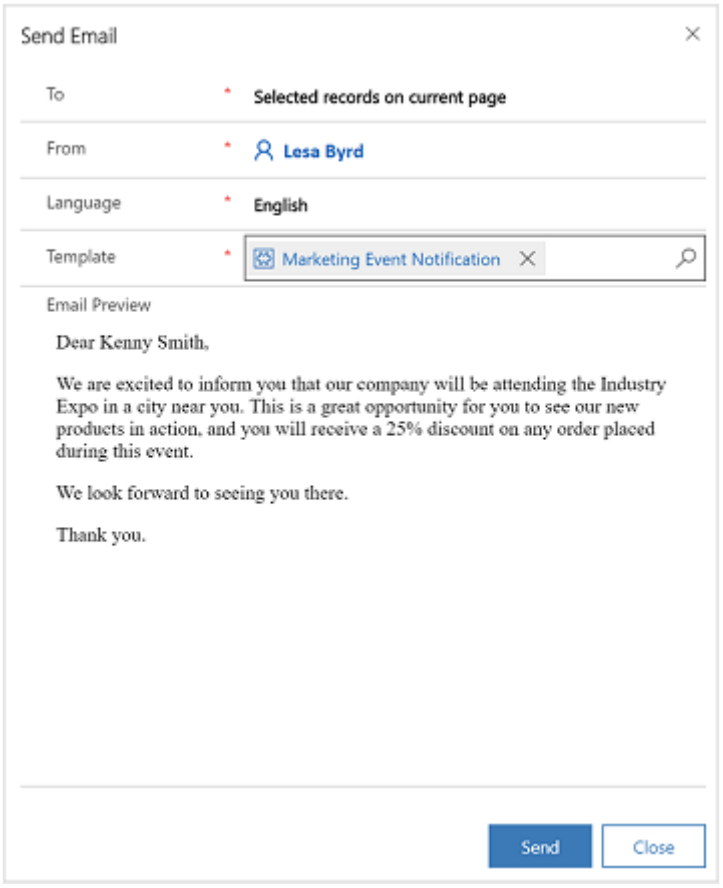
1. In the site map for the model-driven app, select a table to which you want to send email. For example, **Contacts**.
2. In the list of rows, select the contacts you want to send an email to.
3. On the command bar, select **Send Direct Email**.



4. In the **Send Email** pane, select an email template from the **Template** list.

> **Note**
>
> - If you have multiple rows that span multiple pages, you can select one of following options from the **To** list:
>   - **All rows on current page**: Sends the email to all rows displayed on the current page.
>   - **All rows on all pages**: Sends the email to all the stored rows.
> - If a few rows don't have an email address or they have an invalid email address, those rows will be skipped from sending the email.

5. Select **Send**.

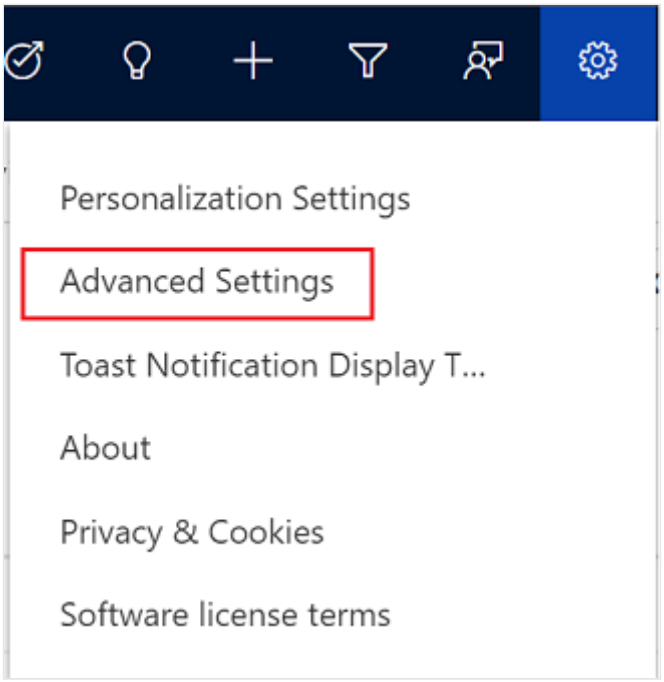


Check the status of bulk email action

You can check the status of bulk email action on the **System Jobs** page. You can see whether the bulk email job has failed or succeeded. If a job has failed, you can open the failed job to see its details.

To check the bulk email status

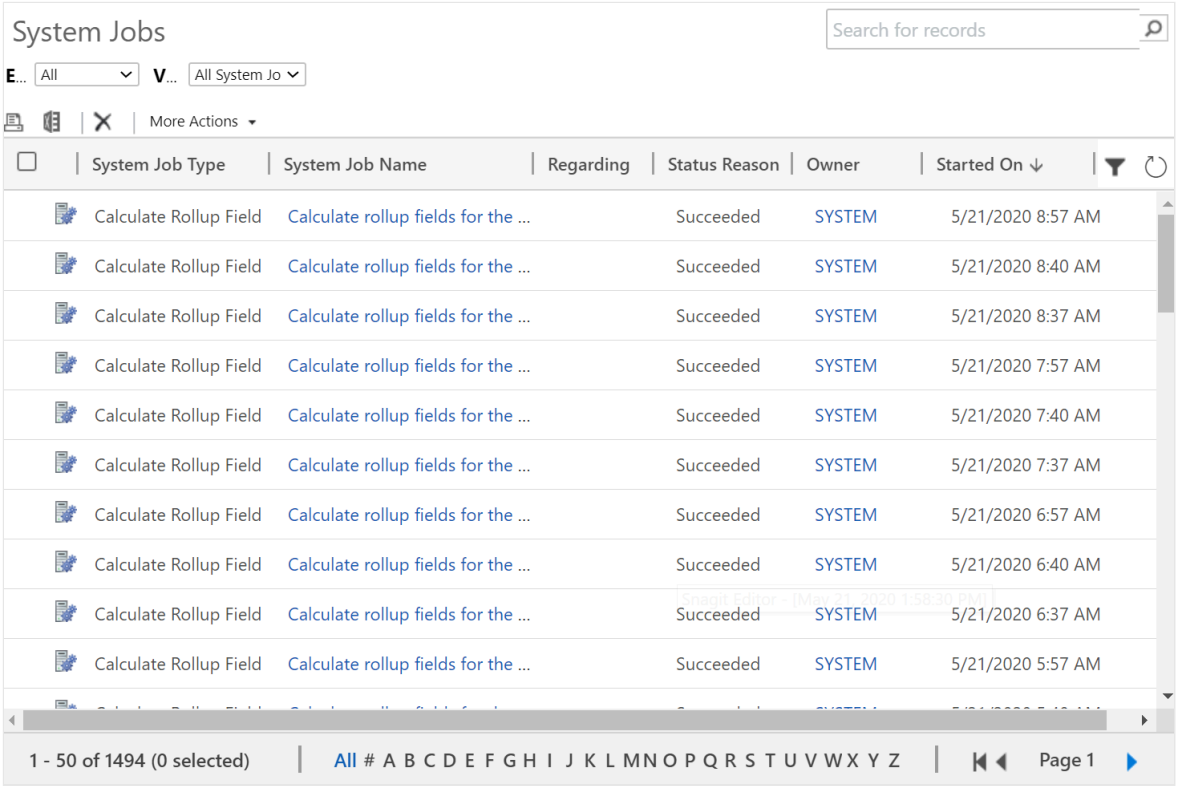
1. In your app, select the **Settings** icon, and then select **Advanced Settings**.



The **Business Management** page opens in a new browser tab.

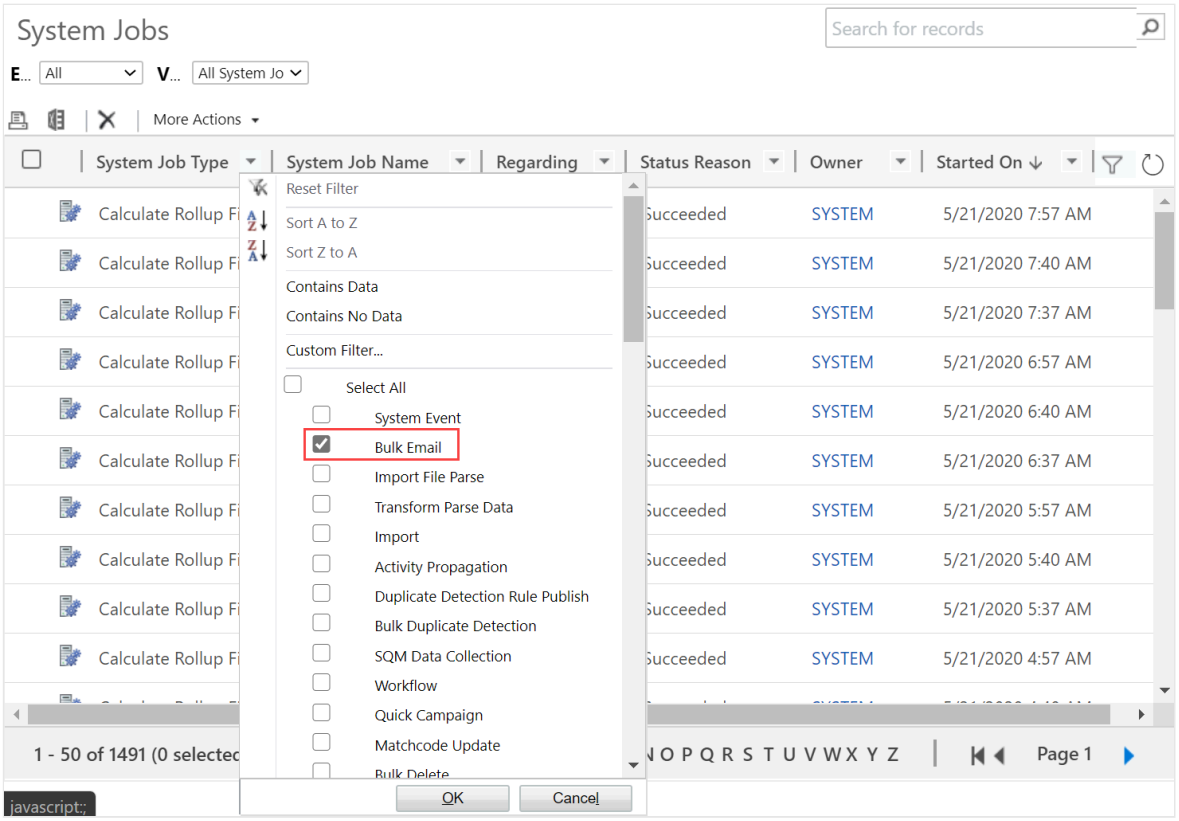
2. On the navigation bar, select **Settings**, and then under **System**, select **System Jobs**.

   A list of system jobs is displayed.

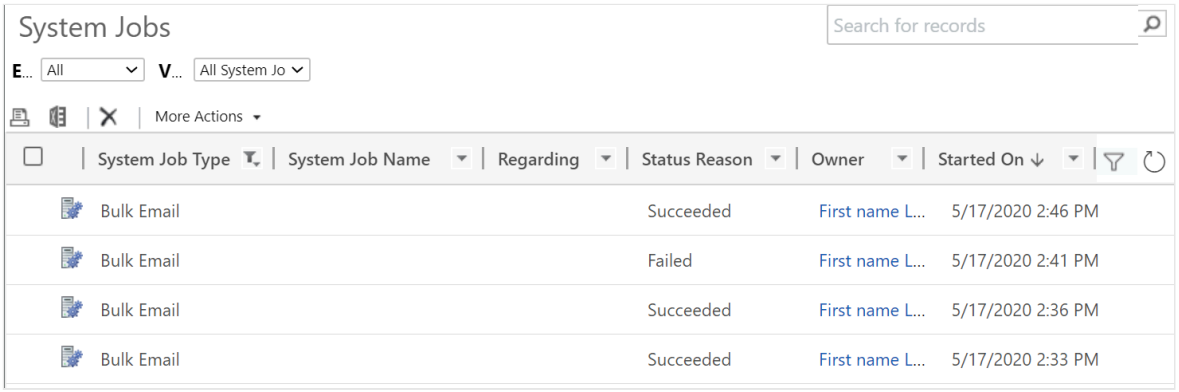


3. In the grid header, select **Filter** .

4. In the **System Job Type** column header, select the down arrow, select **Bulk Email**, and then select **OK**.



Bulk email jobs are displayed with their corresponding status.



5. Double-click the failed job to see its details.

See also

System Settings Email tab

Feedback

Was this page helpful? Yes No

Provide product feedback

Resolve an unresolved email recipient

Article • 02/15/2022

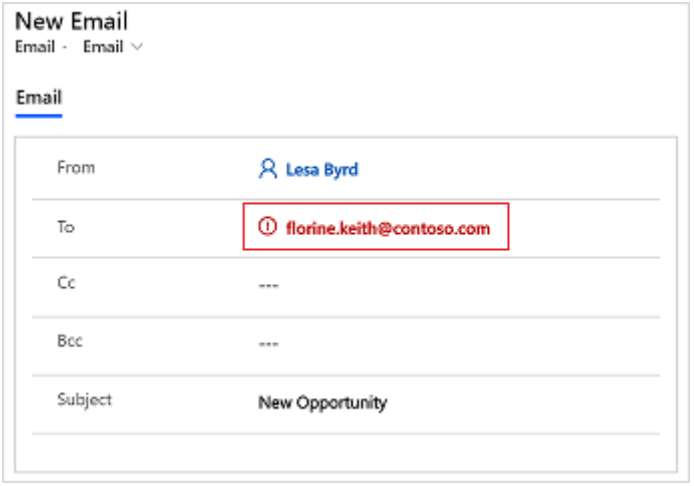
An *unresolved email recipient* is one whose email address isn't associated with any table rows in Microsoft Dataverse. By default, you can't send an email to an unresolved email recipient. If you enter an unresolved email recipient address, the email address is deleted as soon as you move the focus out of the **To**, **Cc**, or **Bcc** column. An administrator must enable the unresolved email recipient feature by selecting **Yes** for **Allow messages with unresolved email recipients to be sent** on the **Email** tab in the **System Settings** dialog box. More information: System Settings Email tab

After the feature is enabled, you can add an unresolved email recipient in the **To**, **Cc**, or **Bcc** column. The entered email address is shown in red. You can then select the email address and associate it with a table row in Dataverse without navigating away from the email form.

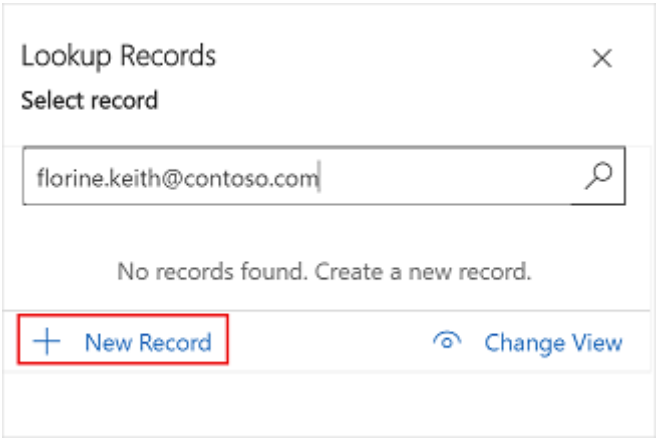
If you receive an email that has email addresses that aren't associated with a table row, the email addresses are shown in red. You can then individually select the email addresses and associate them with a table row. You can then send emails to the newly added email addresses.

To resolve an unresolved email recipient

1. Open the email editor and select the unresolved email recipient.

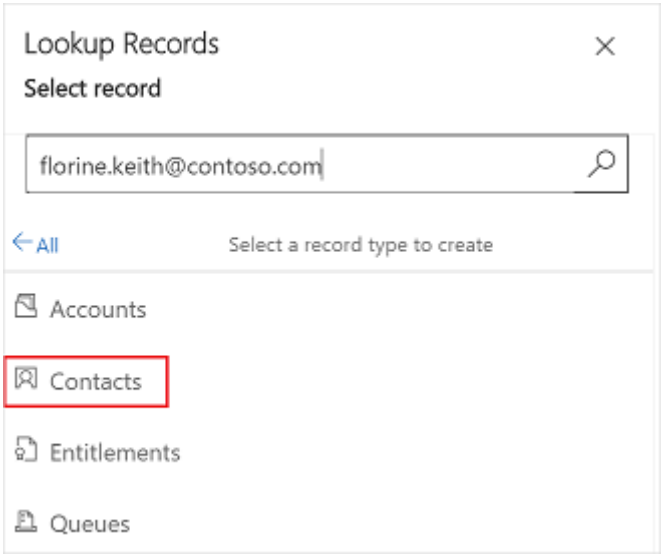


2. In the **Lookup Rows** pane, select **New Row**.

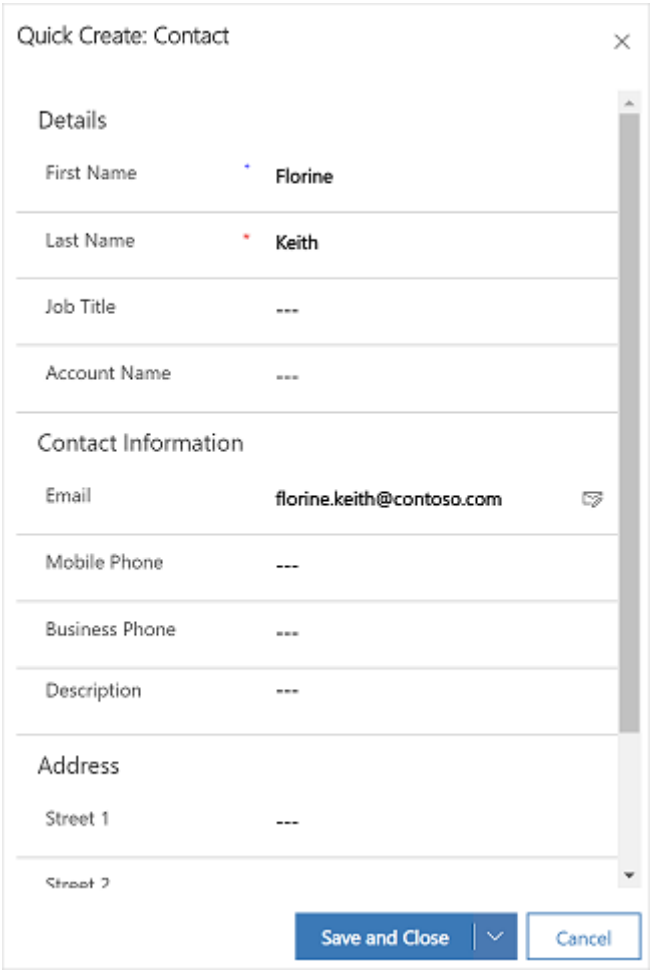


> **Note**
>
> If rows are displayed in search results, you can resolve the email to one of the existing rows.

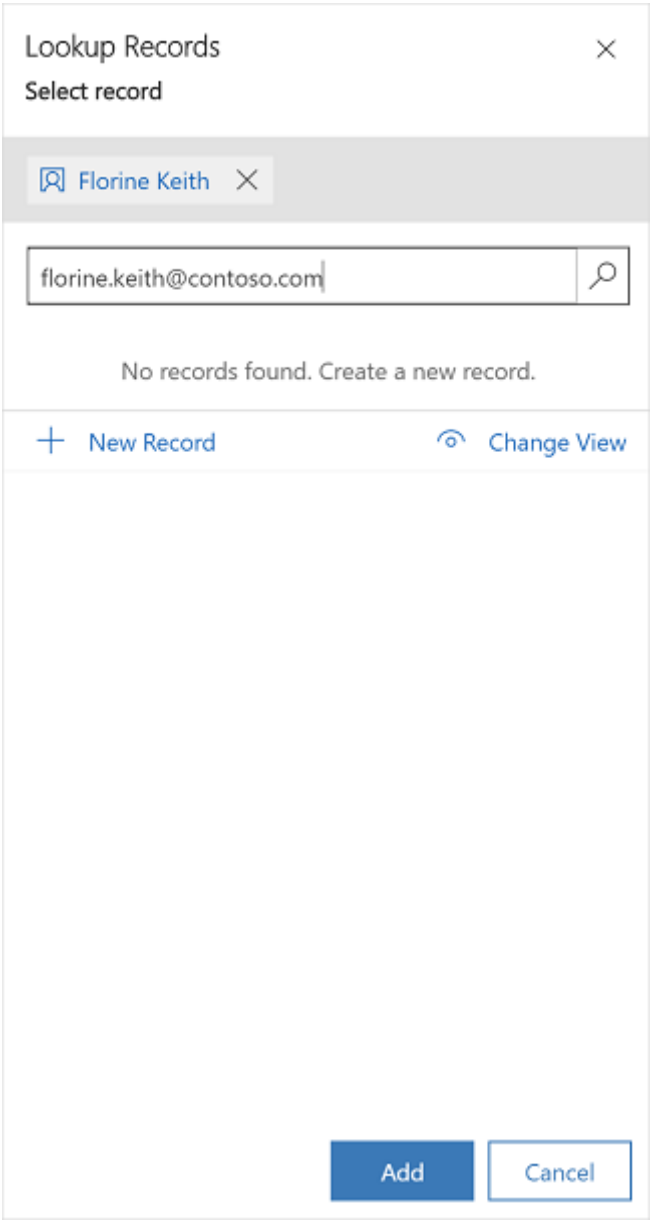
3. Select the row type to create. For example, **Contacts**.



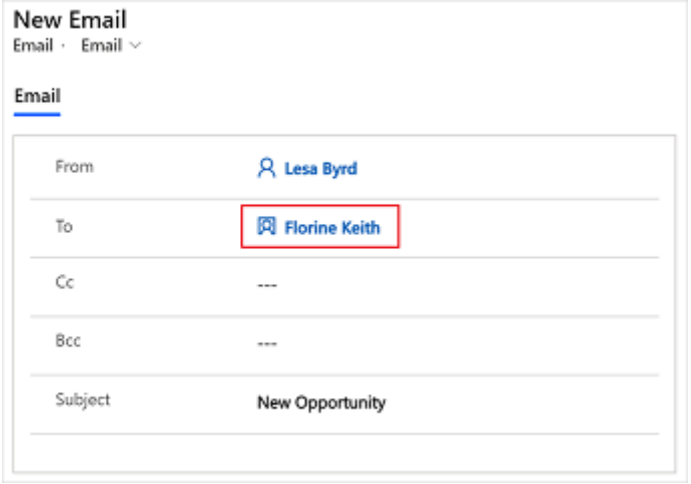
4. In the **Quick Create: Contact** pane, enter the required details, and select **Save and Close**.



5. The contact is created and selected in the **Lookup Rows** pane. Select **Add**.



6. The unresolved email recipient is resolved and shown in the **To** column in the email editor.



See also

Allow unresolved email recipients

Feedback

Was this page helpful? Yes No

Provide product feedback

Email FAQs and known issues

Article • 02/15/2022

Find answers to common questions regarding the email experience features and functionality, configurations, and other key concepts.

Email content is lost after entering text in the body of the message

Issue: Recent content added to the email may get lost if you enter text in the body of an email and immediately select **Save**, **Save and Close**, or **Send**.

Resolution: To avoid this issue, wait a few seconds before you select **Save**, **Save and Close**, or **Send** after you enter text in the body of the email.

The From column is read only or lookup isn't working

Issue: The **From** column in an email form is read-only or you can't look up rows and filter results by **User** or **Queue**.

Resolution: This happens when customization specifically have been applied on this column by your system administrator. To fix the issue, open a browser window and run the following two commands replacing the **environment URL** with the URL of your environment.

- (replace with your environment URL)/api/data/v9.1/RemoveActiveCustomizations(SolutionComponentName='AttributeLookupValue',ComponentId=(25E9AF0C-2341-DB11-898A-0007E9E17EBD))
- (replace with your environment URL)/api/data/v9.1/RemoveActiveCustomizations(SolutionComponentName='AttributeLookupValue',ComponentId=(26E9AF0C-2341-DB11-898A-0007E9E17EBD))

To find the URL, in the address bar the first part of the URL that starts with **https://** and ends with **.com** is your environment URL. For more information on how to find the environment URL, see Get the environment UR.

I've enabled an email form but it doesn't display.

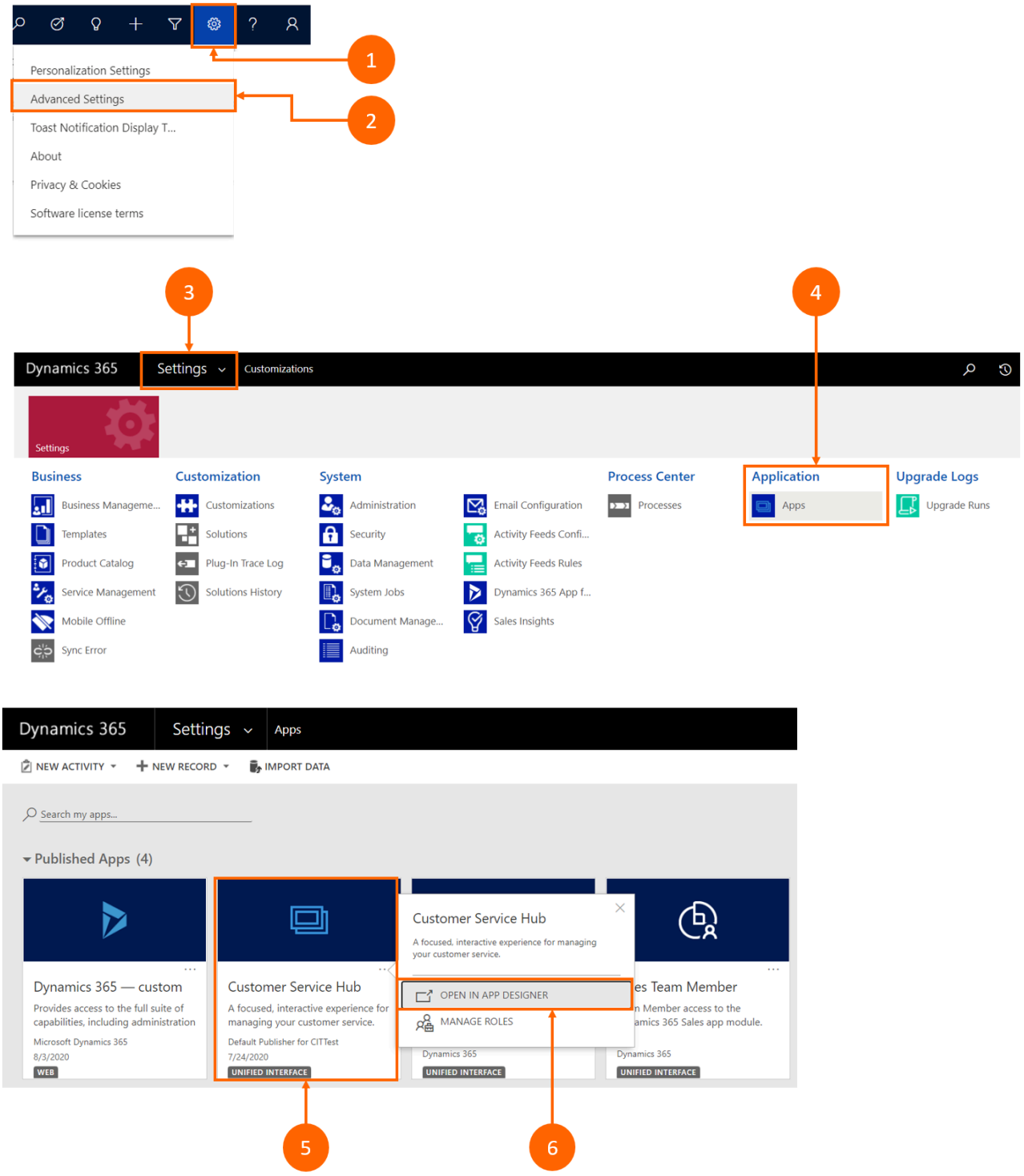
If you've enabled an email form but it is doesn't display, you'll need to enable it in the **App Designer** by doing the following:

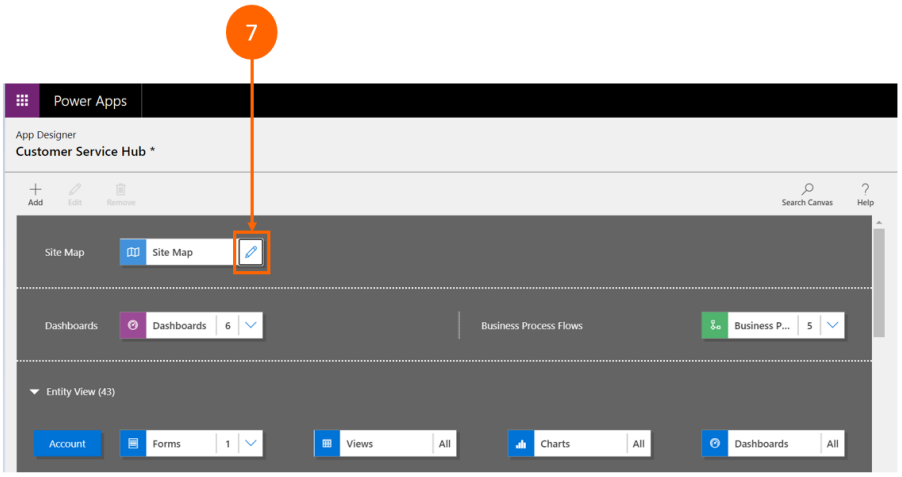
1. Go to **Settings** > **Advanced Settings**.
2. Select **Settings**.
3. In **Application** settings, select **Apps*.
4. Select ⋮ **More Options** on **Customer Service Hub**.
5. On the **App Designer** drop-down menu, select **Open**.
6. On the site map, select the edit feature.
7. Under the **Email** form selector, select the drop-down menu to display the **Email** forms you have enabled.

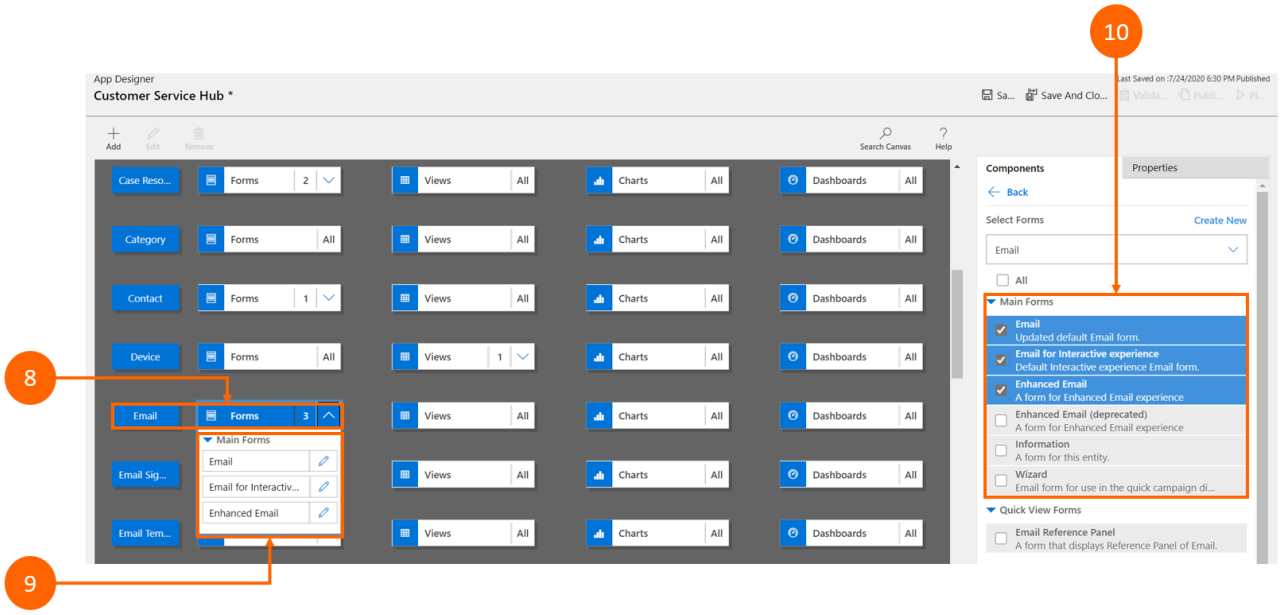
If the form isn't displayed in the list, it hasn't been enabled.

8. To enable a form, go to **Main Forms** and select the form.

> **Note**
>
> You can't change the display order under **Main Form**. You must go back to the **Form Order** to change the display order.







9. Add the email template and signature link to the site navigation by going to the site map and doing the following:

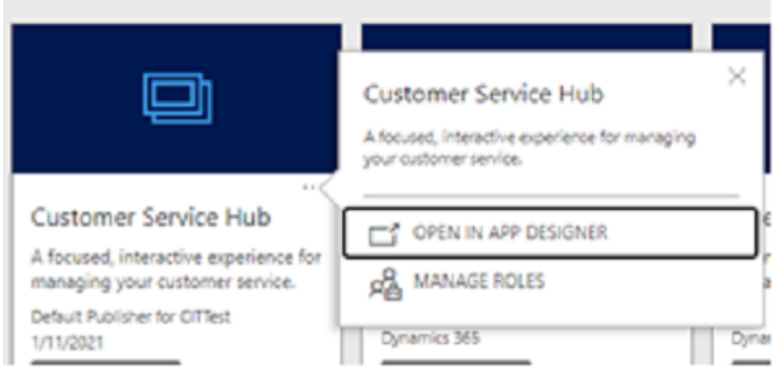
   a. Go to **Settings** > **Advanced Settings**.

   b. Select **Settings**.

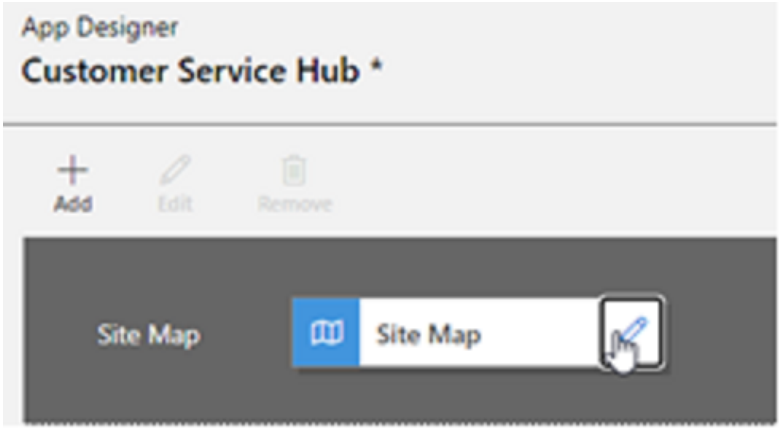
   c. In **Application settings**, select **App*.

   d. Select **More Options** on the **Customer Service Hub** tile.

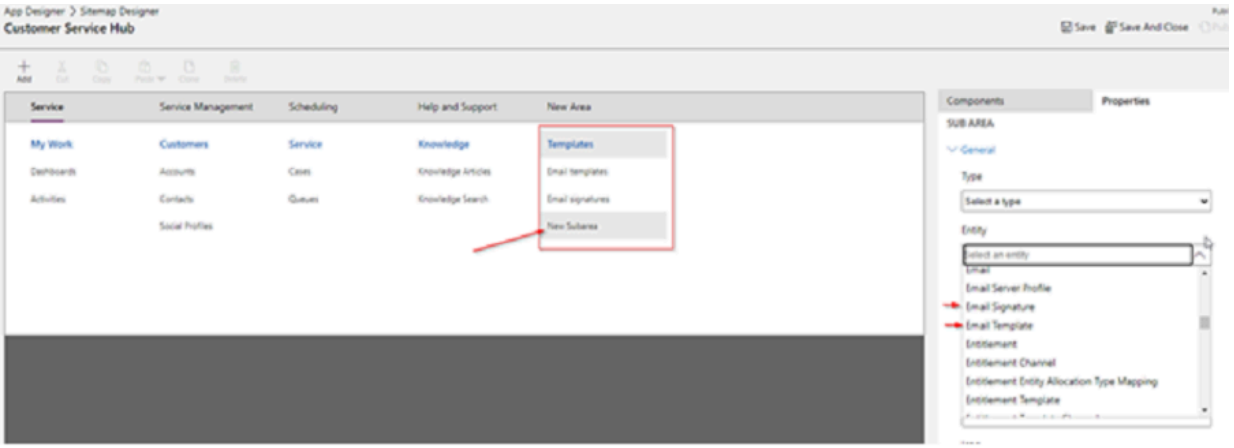
   e. Select **OPEN IN APP DESIGNER**.



f. Select the pencil icon to edit.



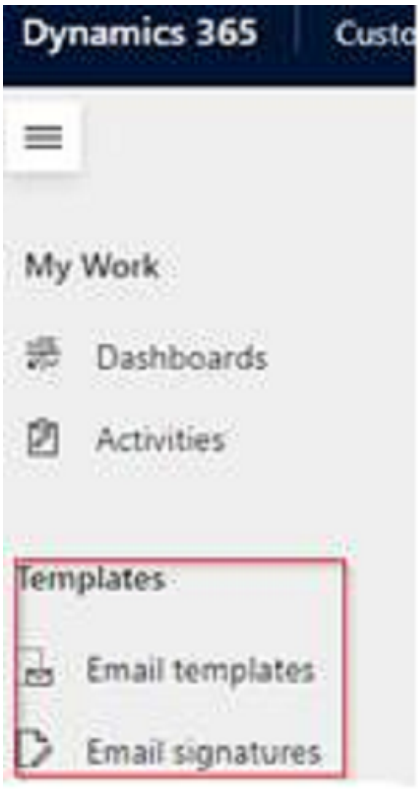
g. When the site map opens, create a new group for templates by selecting **New subarea**.



h. Select **email templates** or **email signature**.

i. Add a title. (This is the name that will show instead of **New subarea** when saved.)

j. Save and publish. A link is added to the site map.



Feedback

Was this page helpful? Yes No

Provide product feedback

Create email templates

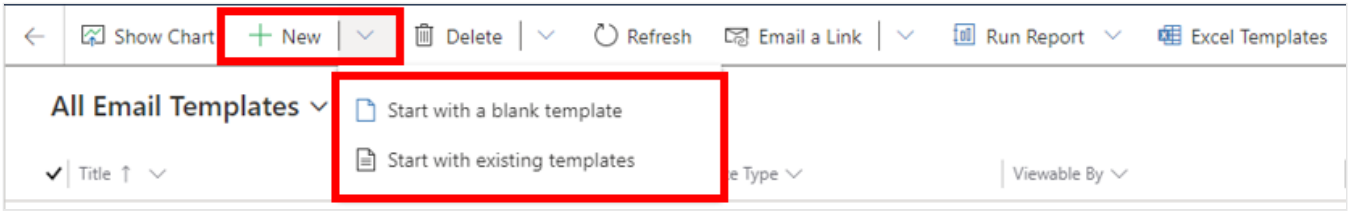
06/03/2025

> **Important**
>
> - As of May 23, 2025, the following functionalities will no longer be available:
>   - The enhanced email template and legacy rich text editor.
>   - The drag and drop functionality of the enhanced email template editor.
> - The modern rich text editor supports HTML formatting. Any existing templates created using the enhanced email template editor will now open in the modern rich text editor. If you any see any formatting issues, we recommend that you update or recreate the template using the modern rich text editor.

Email templates are a fast and easy way to send consistent, professional, preformatted email messages to your customers.

You can create email templates by using one of the following options:

- **Start with a blank template**: Create a template from scratch.
- **Start with an existing template** : Create a template by choosing an existing template.



Prerequisites

When you need to create an email template based on a table, make sure that you have the necessary permissions on the related tables, such as the Account table for the email template based on a case.

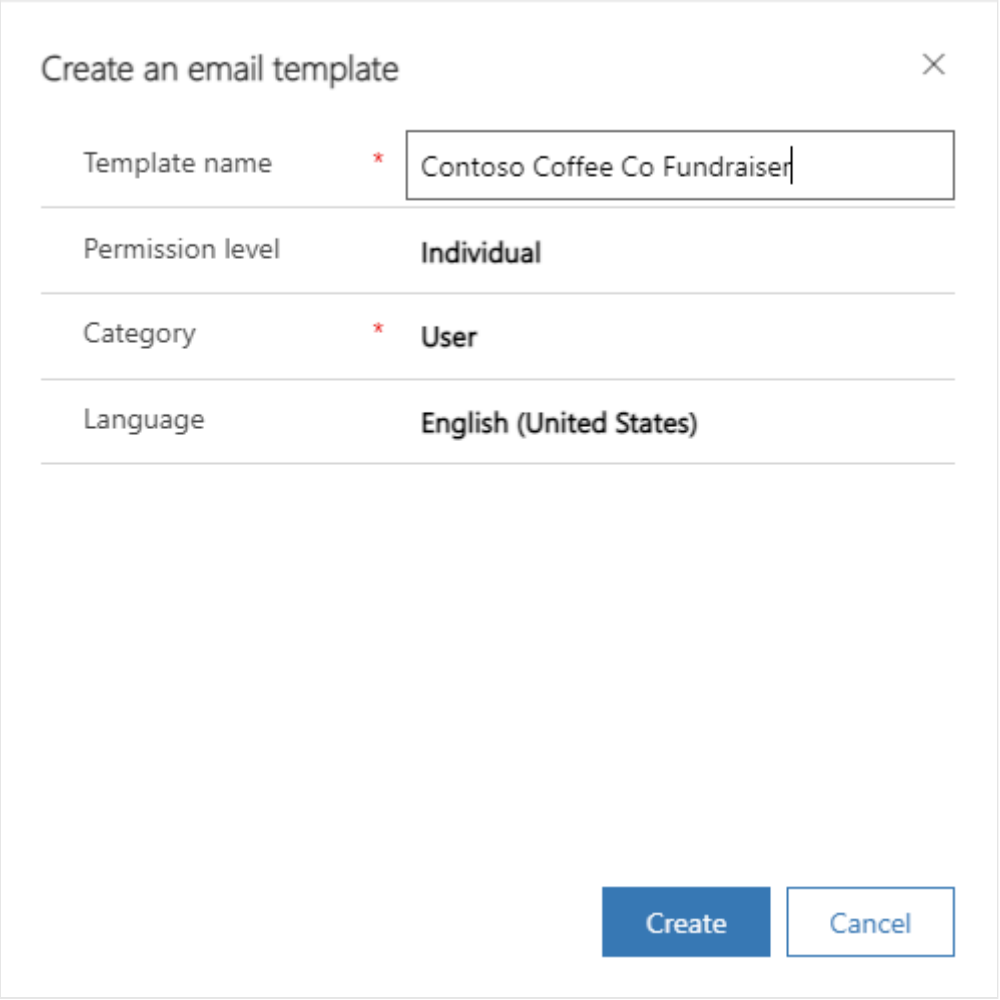
Start with a blank template

When you start with a blank template, nothing is provided for you. You must add everything you want customers to see.

1. On the **Email Templates** command bar, select **New**, and then select **Start with a blank template**.

2. Enter the following details:

   - **Template name**: Give your email template a detailed name to help you identify it later.
   - **Permission level**: Select from **Organization** or **Individual**. More information: Permission level for email templates
   - **Category**: The default value is **User**. Categories determine which dynamic text fields are available for use in your template.
   - **Language**: Display installed language packs. **Language** also helps to categorize your templates.
   - **Create**: Opens one of two editors where you can build your template.



3. Select **Create**. The template editor is displayed.

Permission level for email templates

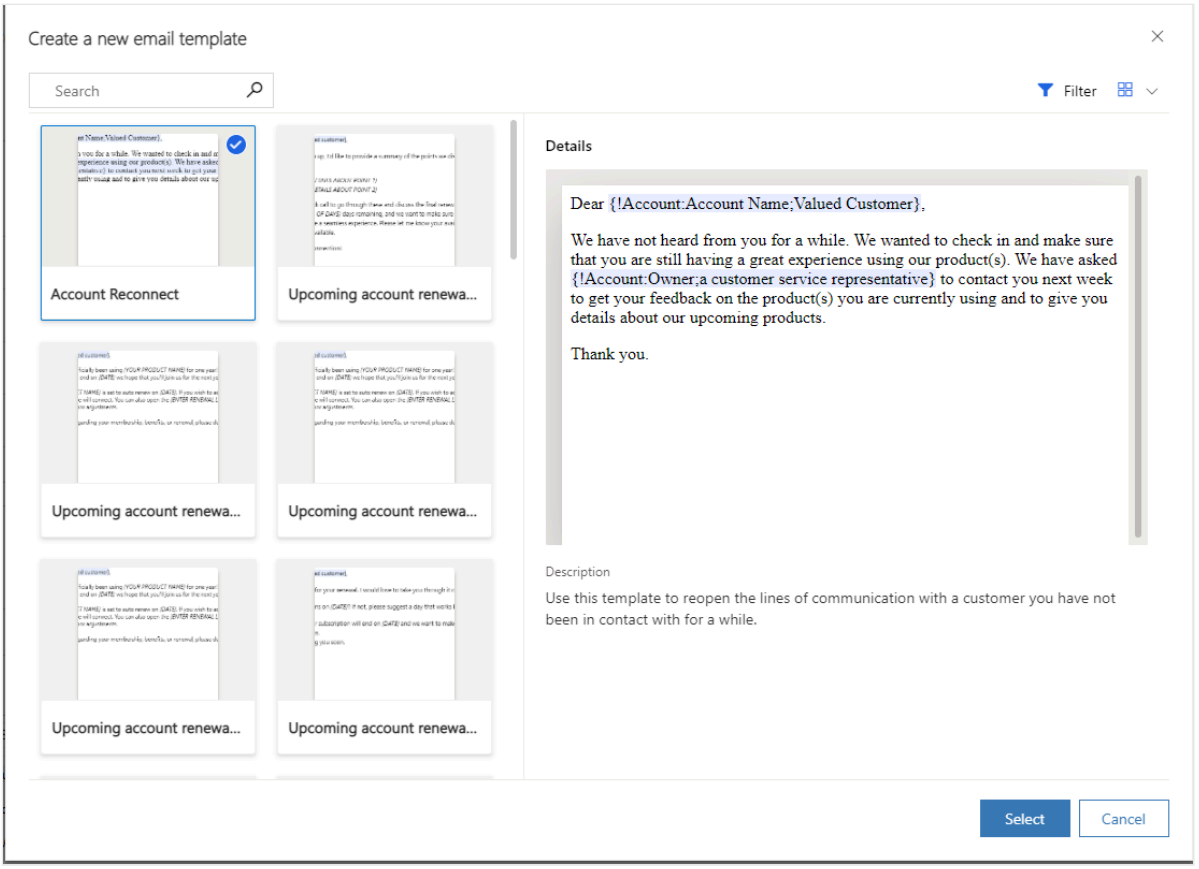
Selecting **Organization** permission level for email templates allows you to create templates that everyone in the organization can use. **Individual** permission level is for personal use.

> **Note**
>
> For **Individual** permission templates, these templates do not show up on the list of email templates unless your are the owner of the templates or the owner has shared the template with you.
>
> For **Organization** permission templates, these templates show up for all users in the organization.

Start with existing templates

Start with an existing template to create an email template faster and with less effort. The template is pre-populated with data. All you need to do is customize it for your needs.

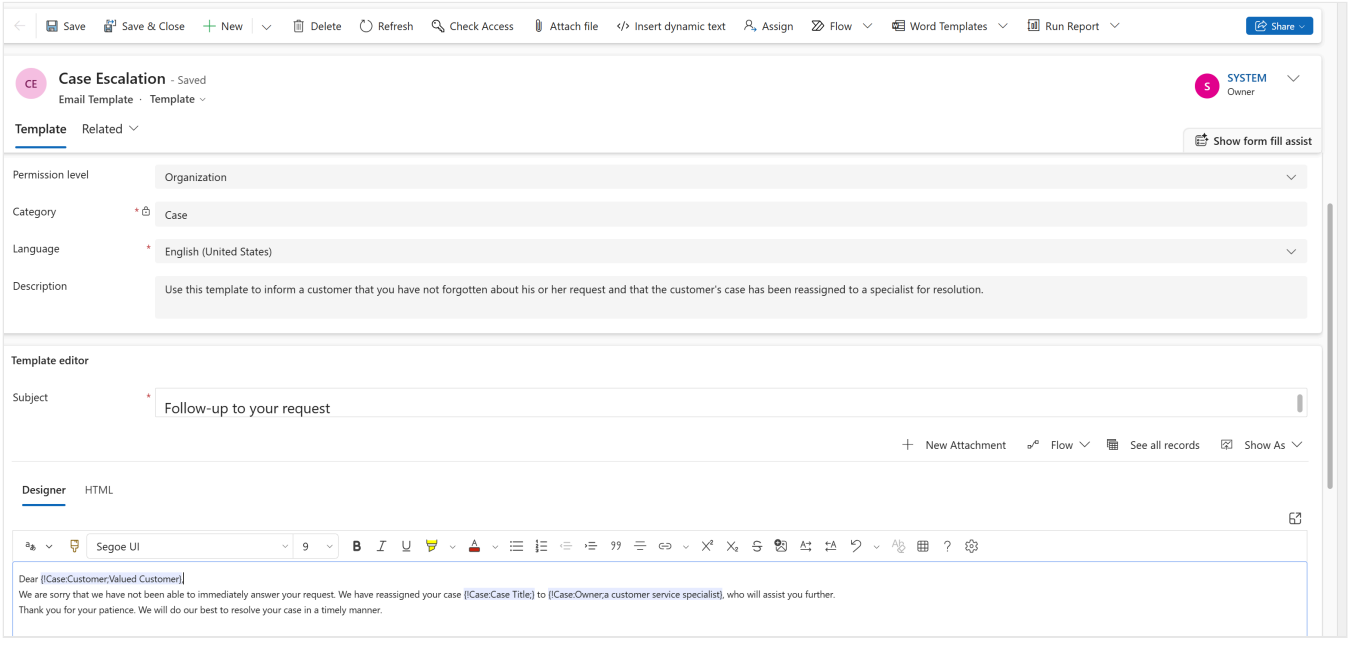
1. On the **Email Templates** command bar, select **New**, and then select **Start with existing templates**.
2. Enter the following details:
    - **Search**: Search for the name of a template to start from. Search doesn't support regular expressions.
    - **Browse** out-of-the-box templates, global and entity-specific, listed in alphabetical order. Global templates are shown as the type User. If you've created custom email templates, they're also listed.
    - **Details**. Preview the templates so that you can pick the one that best meets your needs. You can modify it as needed later.



3. The template editor page is displayed.

Email template editor

You can use the email template editor to to create a new email template or edit an existing template.



- **Designer** tab: Use this tab to draft and apply standard formatting to your email template. You can also add dynamic text, images, and attachments. You can add Copilot prompts to

your email template for a consistent experience. Learn more in Add Copilot prompts to email template.

> **Note**
>
> Across all the model-driven apps, the email template editor supports upto 1048576 characters.

- **HTML** tab: Use this tab to view and edit the HTML code of your email template. You can add HTML code to customize your templates with elements such as layouts, images, and buttons.
- **Template** tab: Specify or change the template details and the email subject.
- **Attachments** tab: Add attachments to your template if needed.

See also

Personalize content with Insert dynamic text
Understand the email experience

Personalize emails with dynamic text

Article • 03/05/2025

Generic greetings like "Dear customer" make customers less likely to engage with your emails. Personalize your outreach instead with dynamic text in your email templates.

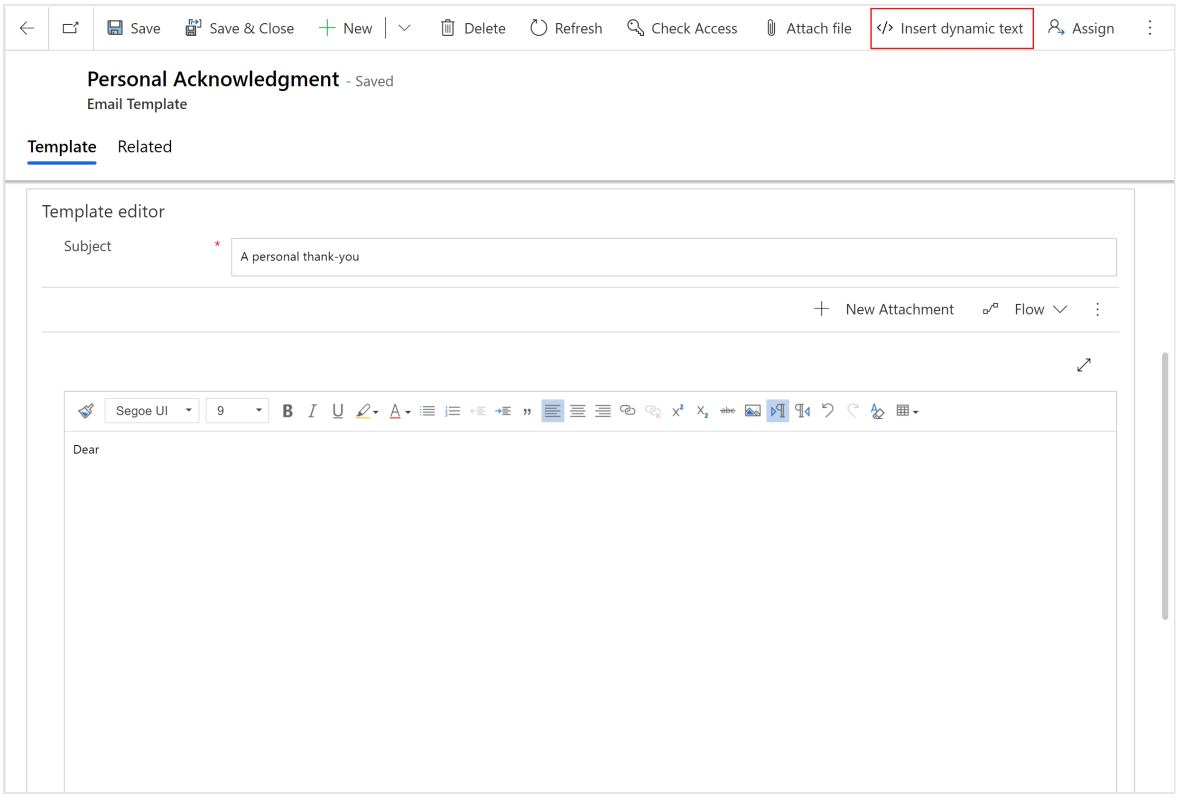
Dynamic text replaces a placeholder entity with the value of the entity when you send the email. For example, your email template might contain "Dear `{!User:First Name;}`" but your customer sees "Dear Sal".

Insert dynamic text

You can insert dynamic text in existing email templates or templates that you create.

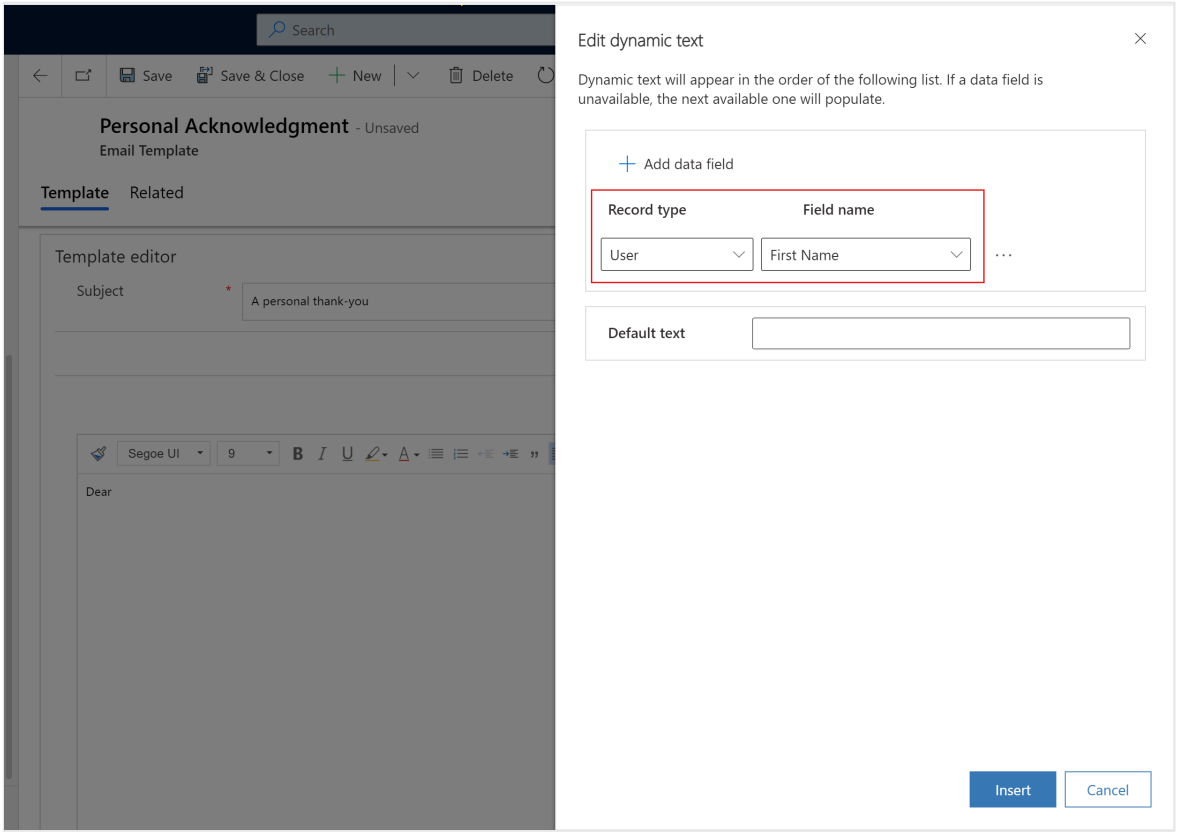
> **Note**
>
> Dynamic text in a hyperlink or an anchor tag isn't supported.

1. Select **Email Template** to edit an existing template or create a new email template.
2. On the template editor, place your cursor where you want the personalized content to appear.
3. Select **Insert dynamic text**.



4. Select the **Record type** and **Field name**.

   The category that you specified when you created the email template determines the record types you can choose. The record type you choose determines the fields you can select.



Insert custom fields

You can insert fields from custom entities by typing them directly in your email template.

Use the following syntax:

Expand table

Field Type	Syntax
• Single line of text • Currency • Multiple lines of text • Decimal number • Floating number	`{!EntityLogicalName:FieldLogicalName;}`

Field Type	Syntax
• Lookup • Multi-select option set • Option set	`{!EntityLogicalName:FieldLogicalName/@name;}`
Date	`{!EntityLogicalName:FieldLogicalName/@date;}`
Time	`{!EntityLogicalName:FieldLogicalName/@time;}`

For example, let's say you want to insert a custom field, *Customer ID*, linked to the record type *User*. Type the following placeholder in your template: `{{!User:CustomerId;}}`.

If you want to insert a custom field, *ModifiedOn*, linked to the record type *User*. Type the following placeholder in your template: `{{!User:ModifiedOn/@date;}}`.

See also

How to create an email template in model-driven apps
Enable the enhanced email template editor page

Feedback

Was this page helpful? Yes No

Provide product feedback

Insert an email template

Article • 04/14/2025

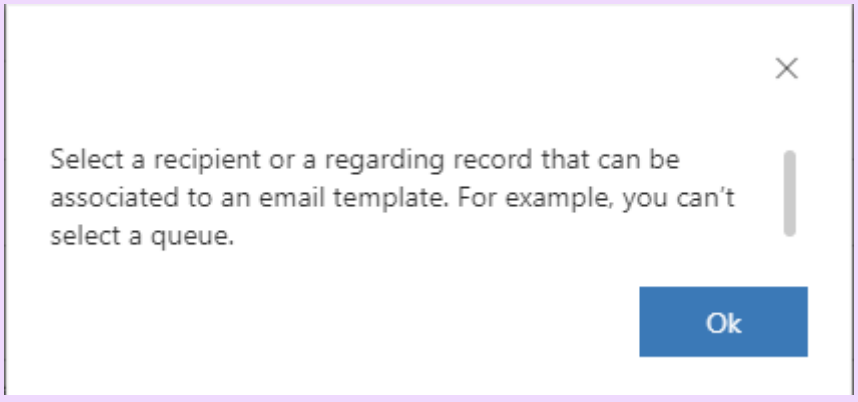
If you previously created an email template, you can easily use it as the basis for your email by inserting it into the email.

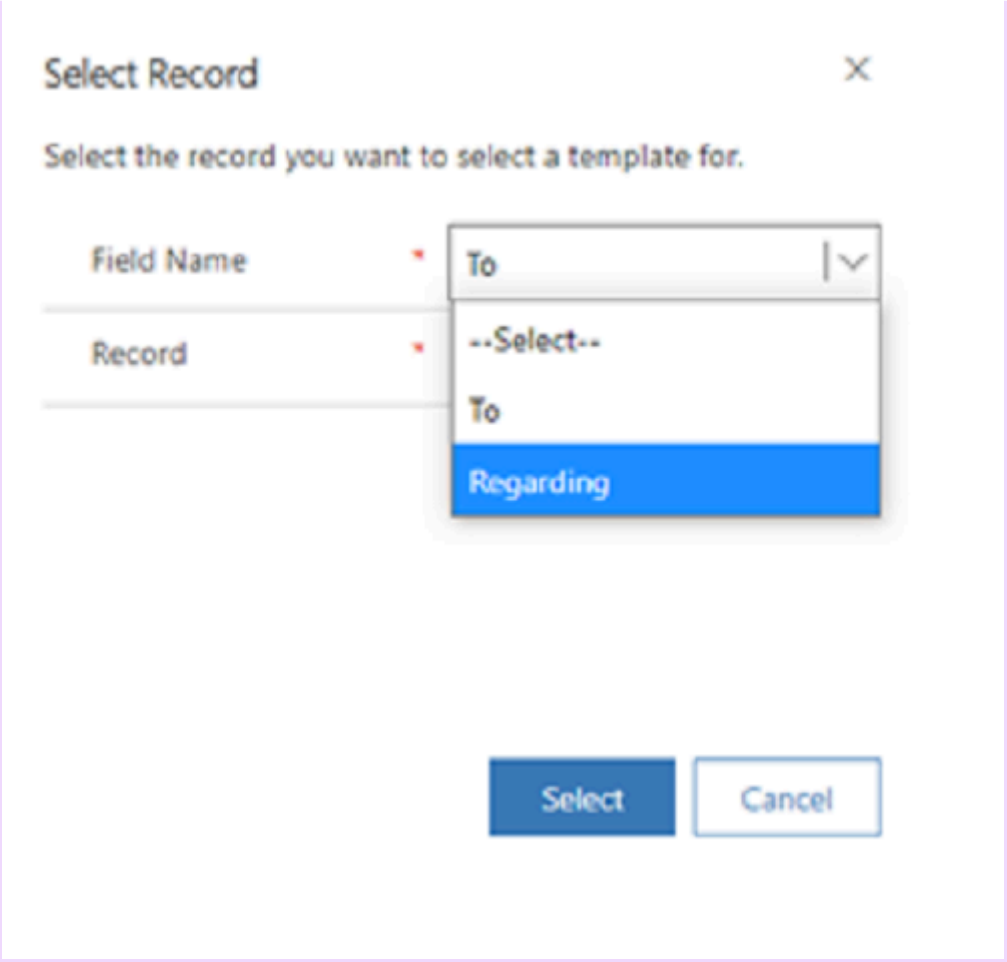


You can access the email template you created from the command bar by selecting **Insert Template**.

Insert an email template overview

After you select **Insert Template**, a window appears displaying a list of existing **Email templates** that are available to use.

> **Note**
>
> There is an intermediary window that might appear if:
>
> - You don't have either a Recipient or Regarding field for your email.
>
> 
>
>
> - You have both a Recipient and Regarding field. You must select one.



This window doesn't appear if your administrator has set **Skip Select Record dialog** in **Advanced Settings** to **Yes**. The selection of one of these fields determines which template types are shown to a user in the template selection window:

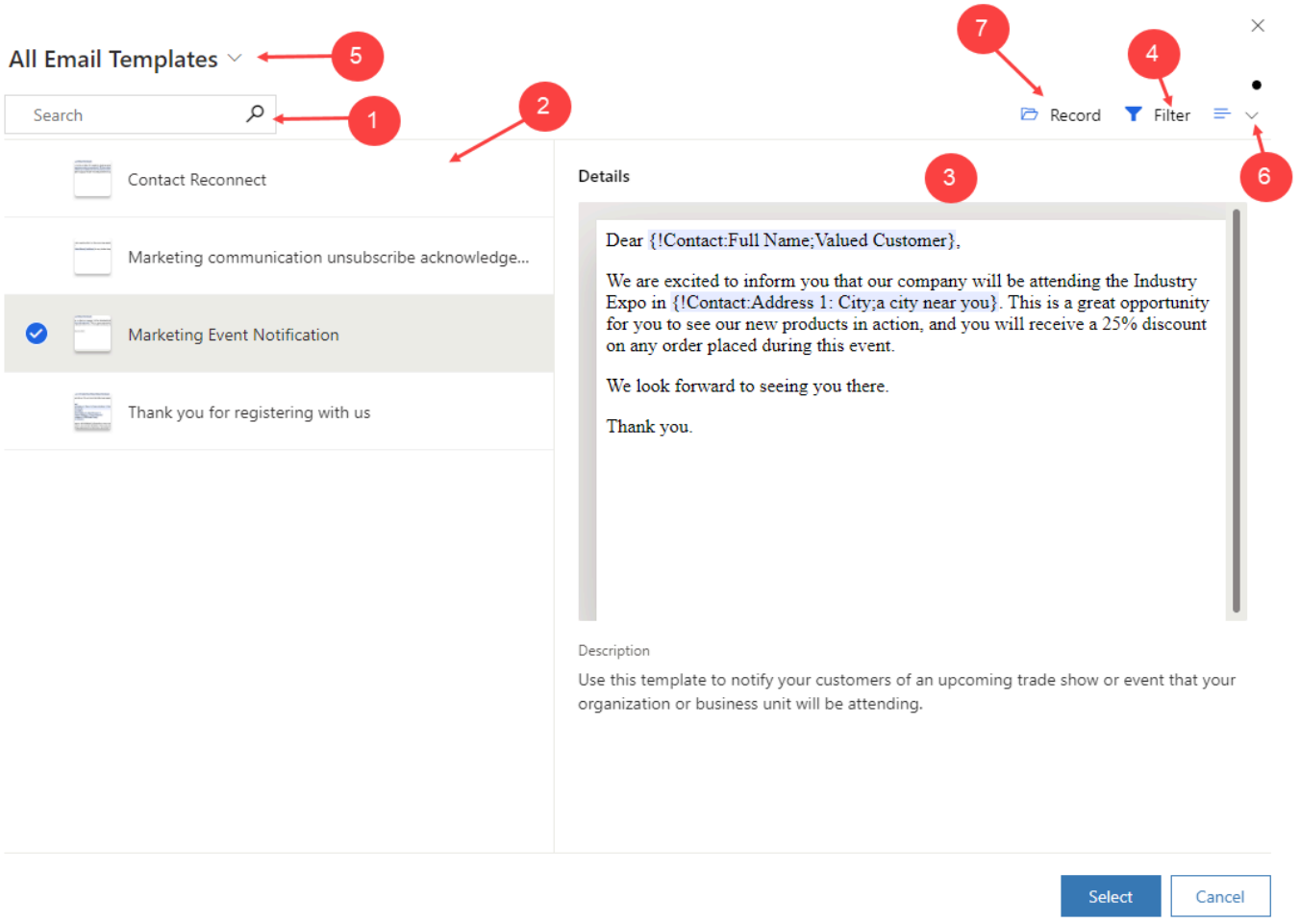
- Recipient (TO): user (global) and contact templates are displayed.
- Regarding: user (global) and templates for the regarding entity are shown.

Enhanced email template selection dialog

The application displays the enhanced email template selection window only if your admin has configured the enhanced insert template dialog option. You can perform the following actions in the enhanced email template selection window:

- Switch between email template views.
- Switch among list, tile, and grid views.
- Search for templates based on their title, subject, description, or content of the template.
- By default, you can filter templates based on standard and custom attributes, and language, if your Administrator has enabled **Language** in the **Template Gallery Filter Form**. See: Add the language filter in the email template selection view
- Additionally, if your administrator has set **Enable Email Template views** in **Advanced Settings** to **Yes**, you can filter templates based on the email template view filters.

- Zoom in to view templates on a full screen. The application provides an option to navigate between templates.



Legend

1. **Search**. Use search to find a template. You can search for templates based on name, subject, description, or content. Search doesn't support regular expressions.
2. **All templates**. Choose and browse from the templates displayed based on TO or Regarding, the filters applied, or a combination of both. You can change the view to display the templates in a grid, list, or tile view. Hover over the template tile to see the zoom option. Select the tile to open the template on a full screen.
3. **Preview**. When you select an email template, a preview of the template is displayed here. The preview shows you the content so you can pick the template that best meets your needs. After inserting an email template, you can modify the content as needed in the email editor.
4. **Filter**. You can filter templates based on standard or custom attributes.
5. **View**. Use the view to switch between email template views. Copilot can recommend the relevant email template for your email. The **Copilot recommended templates** view displays five relevant email templates with prompts that Copilot recommends.
6. Use the dropdown list to switch between grid, tile, or list views. Based on your admin's settings, the application displays a specific view. More information: Customize the email.

7. **Record**. Specify the **Field Name** and **Record**. The template types shown to a user is based on what you've selected for these fields.

    - Recipient (TO): user (global) and contact templates are displayed.
    - Regarding: User (global) and templates for the regarding entity are shown. By default, **Field Name** is set to **Regarding**.

> ⓘ Note
>
> - Bookable Resource Booking isn't supported.
> - You can see the **Record** tab and switch between email template views only if your administrator has enabled the **Enable Email Template views** and **Skip Select Record dialog**. More information: Customize the enhanced email template selection view
> - The enhanced email template selection view might vary based on the customizations your administrator has configured on the **Template Gallery Filter Form**, **Email Template Sorted View**, and **Template Gallery Properties Form**.
> - If you try to insert an email template on a device with a smaller screen size, you'll only see an option to search and select a template only.

Customize your email template selector filters

With the enhanced email template selector dialog, you can switch between multiple email template views. The filters set for the email template views are available on the template selector, and lets you to view only the relevant email templates. You can modify the filters set for the view, but you can't save the modified filter conditions.

For example, you have a custom view, Agent email template view, with filters set to display only the templates that have "case" in the title.

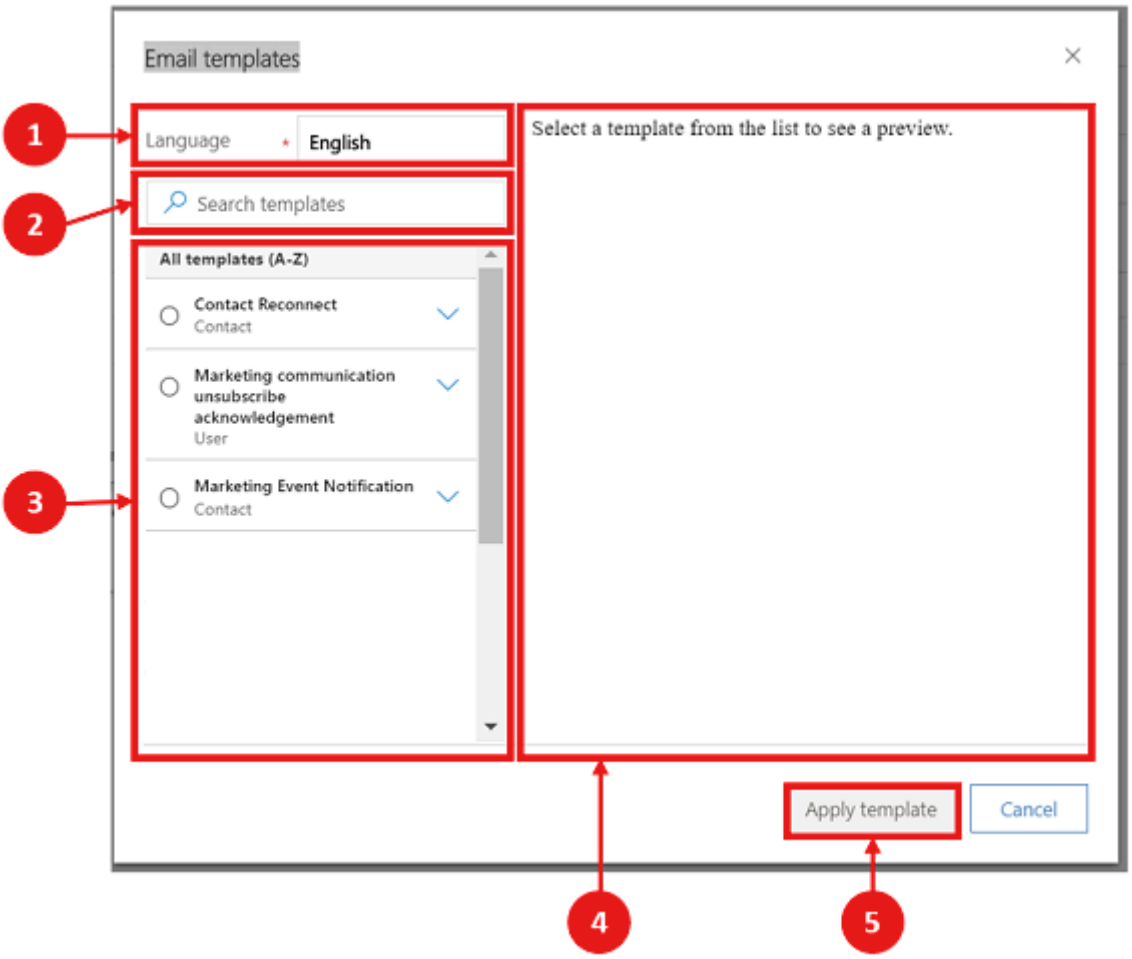
When you access the enhanced email template selector and choose Agent email template view, the application displays only the templates with "case" in the title.

> ⓘ Note
>
> The email templates that are displayed are filtered based on the specified email template view filters and record.

Email template selection dialog

You can perform the following actions on the email template selection window:

- Select a language.
- Search for templates by title.
- Modify the content for a selected template.



Legend

1. **Language**. Templates are loaded according to the selected language.
2. **Search**. Use search to find a template. Search doesn't support regular expressions and only works when using the name of that specific template.
3. **All templates**. All existing templates that have been created are displayed in this window, which you can browse and choose from.
4. **Preview**. When you select an email template, a preview of the template is displayed here. The preview shows you the content so you can pick the template that best meets your needs. After inserting an email template, you can modify the content as needed.
5. **Apply template** to insert the content to your email.

> ⓘ **Note**

If you try to insert an email template on a device with a smaller screen size, you'll only see an option to search and select a template.

See also

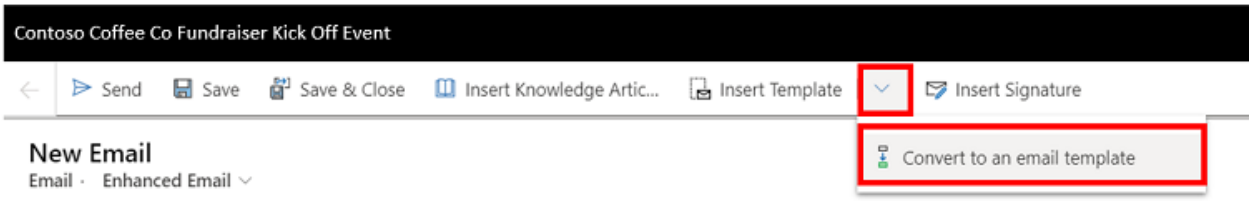
Set up enhanced email
Understand the email experience
Configure the enhanced insert email template

Convert an email to a template

Article • 02/17/2025

When you've composed an email that you'd like to reuse with other customers, you can convert it to an email template and save it for future use.

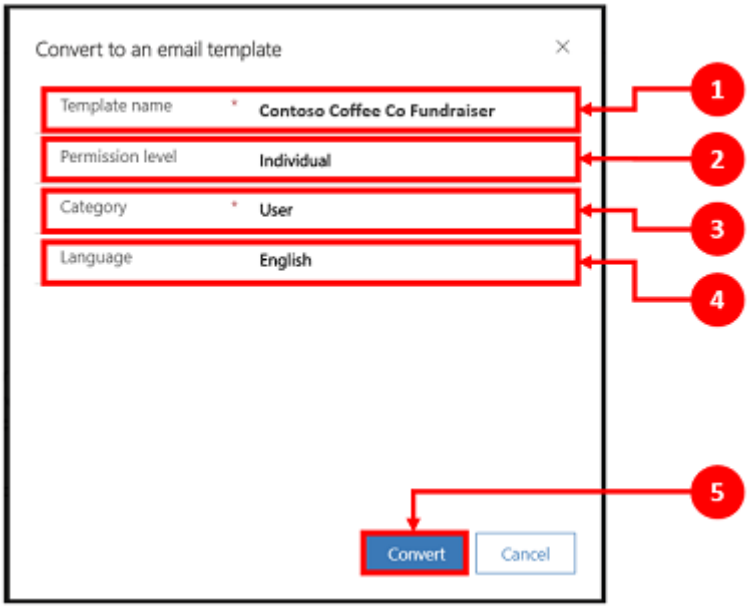
Convert an email into a template



To save an email you've composed and formatted an email message as a template, select the downward-facing caret (v) next to **Insert Template**, and then select **Convert to an email template** from the drop-down list.

Convert an email to a template overview

The following window will display with a list you can customize.



Legend

1. **Template name**. Assign your email template a detailed name you want to save for future use.
2. **Permission level**. Select from **Organization** or **Individual**. More information: Permission level for email templates

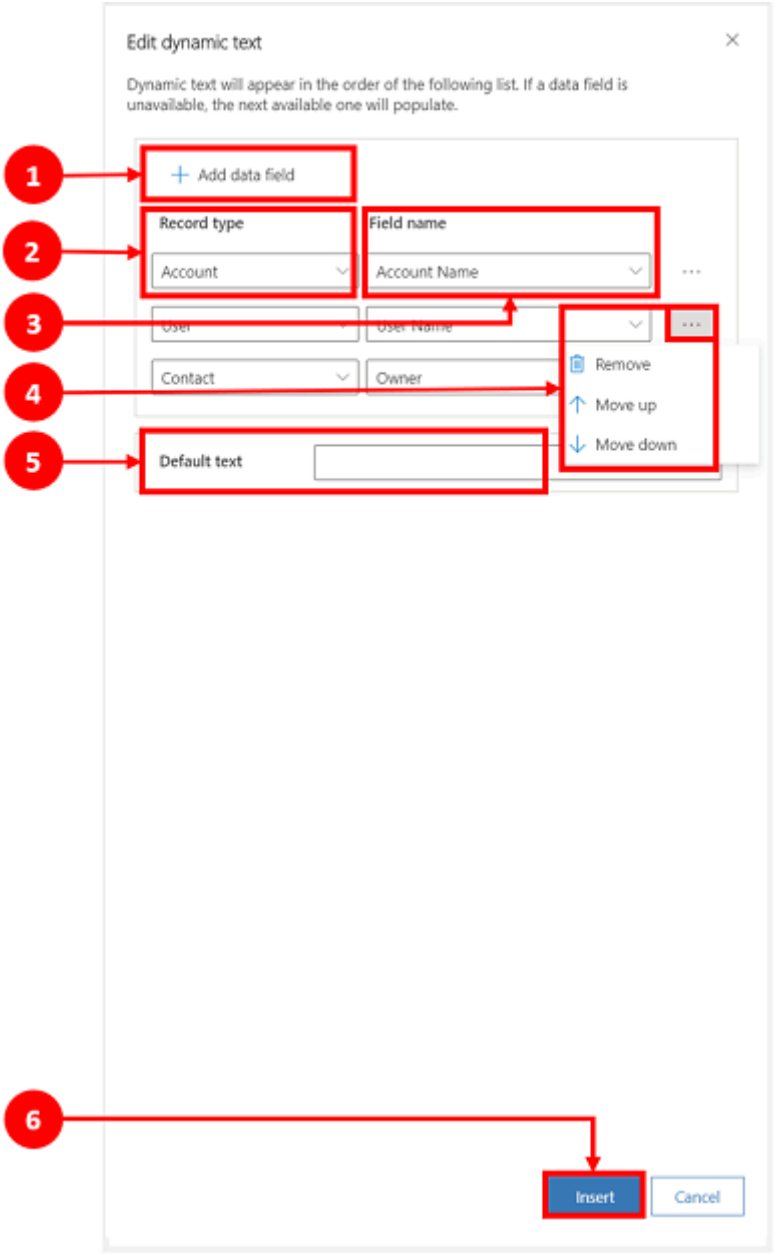
3. **Category**. The default for this field is set to **User**.
4. **Language**. You can display your installed language packs in this field. **Language** also helps to categorize your templates.
5. **Convert**. Once an email is formatted, you can convert it to an email template.

View an email template

To view a complete list of all the email templates you've created:

1. Go to the **Templates** section on the left-hand pane.
2. Select **Email templates** A list of **All Email Templates** is displayed, which you can edit and update.

Insert dynamic text in an email template form



Legend

1. **Add data fields**. When you add data fields, the template adds **Record type** and **Field name** fields below. You can add as many data fields you want.
2. **Record type**. You can then customize the record types. Default is set to account.
3. **Field name**. You can customize the field names. Default is set to account name.
4. **More Commands**. The **More commands** ⋮ icon allows you to edit the order of the record types using the **Move Up** or **Move down** functionality to align records in a customized order.
5. **Default text**. Allows you to select the style of text for the subject and body of your email template. If no text is selected, it will display using the default text.
6. **Insert**. When done this command inserts and saves your form.

> **Note**
>
> If you click in an area on the form that does not support dynamic text, it displays **Unable to insert**.

For more information on creating email templates and configurations, see Create templates for email.

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a signature for emails

Article • 04/10/2023

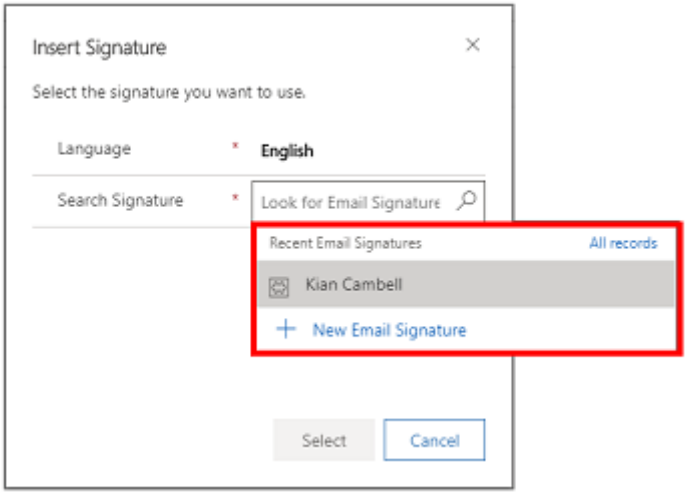
Email signatures are an important and valuable tool for users. Email signatures are like electronic business cards you include when you send an email, and can promote professional branding for your company.

Add text to an email signature

Adding an email signature is easy and can be done at any point while you are composing an email. You can either add an email signature with static or dynamic text. Dynamic placeholders are used to define dynamic text. The placeholders are replaced with the corresponding values of the email's sender, specified in the **From** field.

Add an email signature

- On the command bar, select **Insert Signature**.
- From the drop-down list, select an existing signature or select **New Email Signature**.



Based on the email's sender specified in the **From** field, one of the following actions occur:

- If the sender is a user, the dynamic placeholders defined in the email signature template are replaced with the values corresponding to the user.
- If the sender is set to a queue and the queue's owner is a user, the dynamic placeholders defined in the email signature template are replaced with the values corresponding to the queue's owner.
- If the sender is set to a queue and the queue's owner is a team, the dynamic placeholders are replaced with the values corresponding to the team's

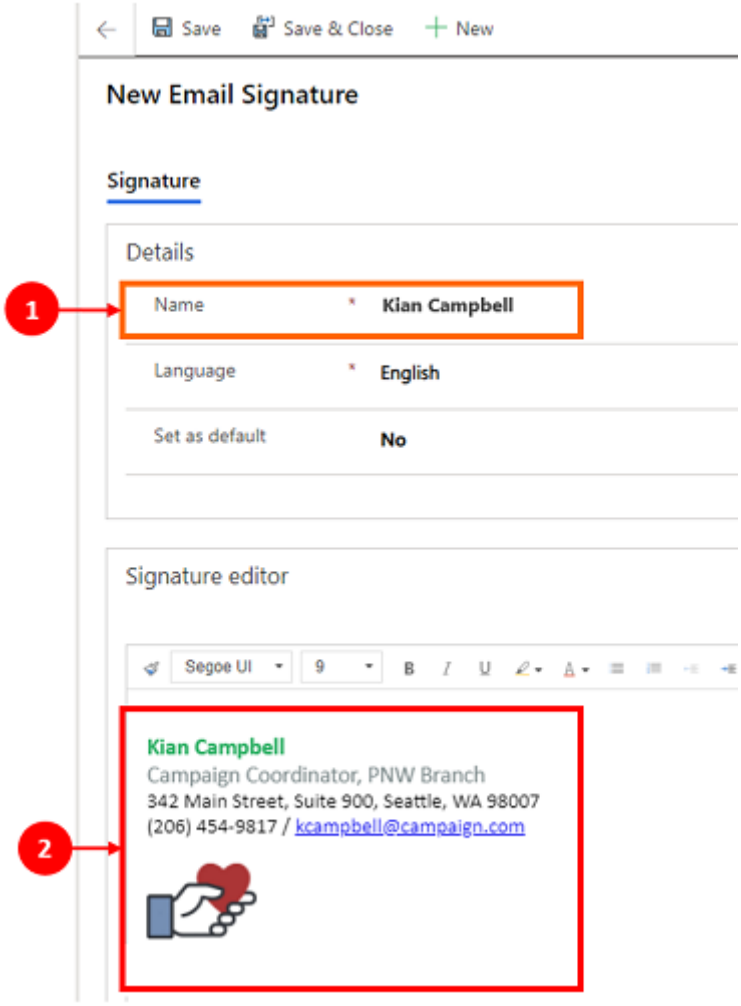
administrator.

Create an email signature

You can quickly create an email signature at any point while you are composing an email.

Create a new email signature

You can add a new email signature by:



1. **Details** in this section, enter the name of the email signature.
2. **Signature editor**. This allows you to create your email signature and **Save** when finished.

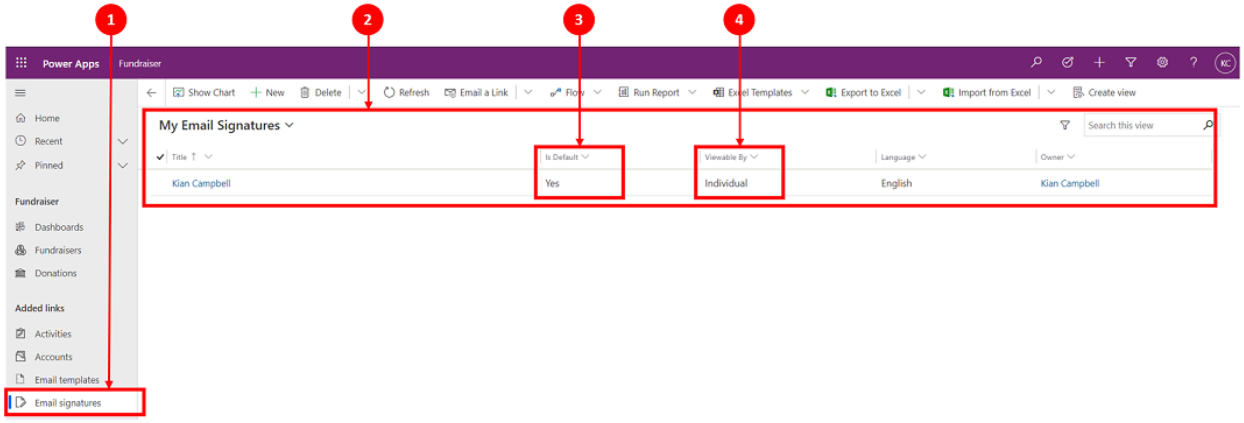
> ⓘ **Note**
>
> - You can include inline images, such as a business logo, in email signatures, as long as they are under 1 MB in size. However, the inline

image might not render properly if you open the email in the Outlook Desktop app.

- You can add only those fields as dynamic placeholders that are linked to the **Record type** user.
- Signatures can also be created and used for queues.

Manage email signature lists

You can view and manage email signatures you have created to make edits, change default, and who can view your signatures.

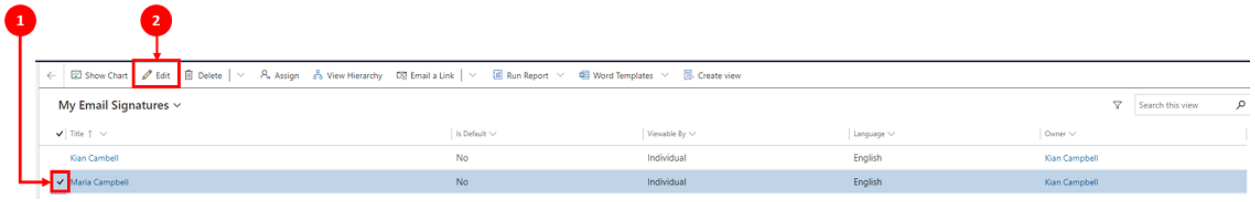


Legend

1. Under **Templates** select **Email signatures**.
2. **My Email Signatures** list displays, showing all your email signatures you can edit and update.
3. You can select a default signature for email. Only one signature per user can be set as **Is Default**.
4. Email signatures can only be **Viewable By** the individual user.

Change the default email signature setting

While you can create multiple email signatures depending on your needs, only one can be set as your default email.



To change your default email signature setting:

1. Select the email signature you want to set as default. This will change the command bar to display **Edit**.
2. Select **Edit**. A new page opens, where you can update the email default setting. While you are on this page, you can also delete one or multiple email signatures just by selecting them.

Feedback

Was this page helpful? Yes No

Provide product feedback

Configure the enhanced insert email template selection dialog

Article • 04/10/2023

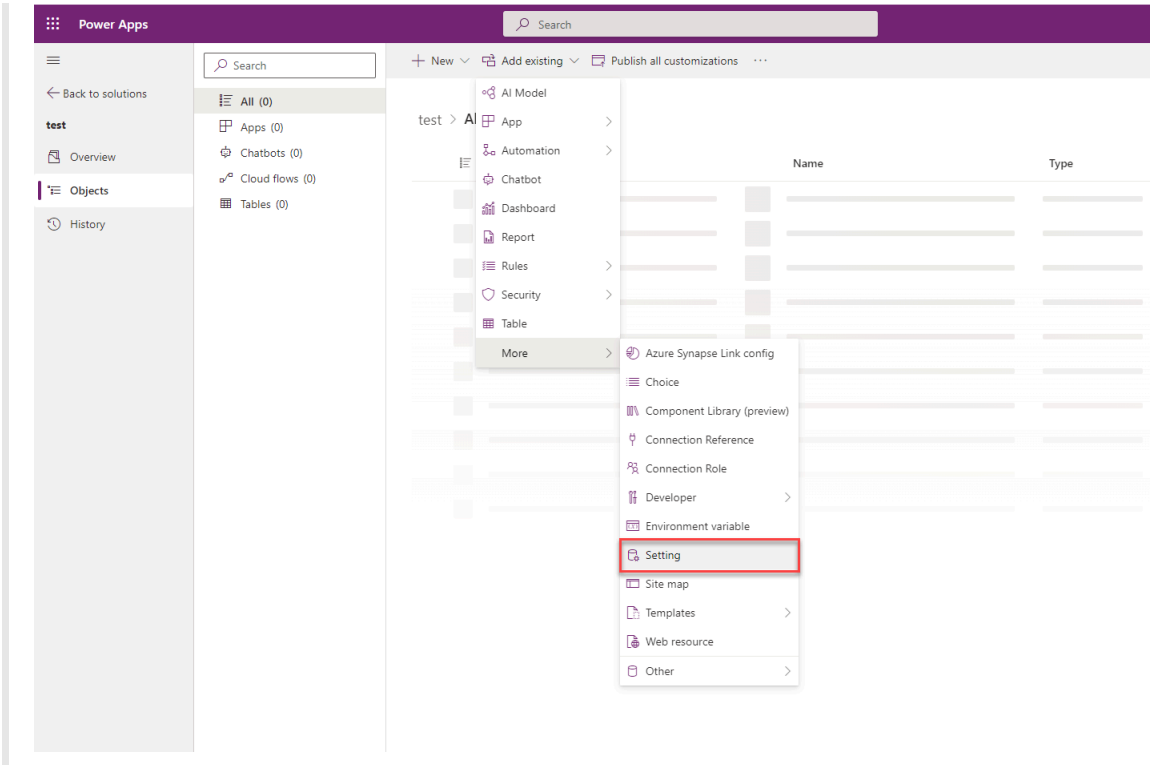
Turn on the enhanced email template selection experience for apps across your organization. If you need to, you can turn off the enhanced email template selection dialog for a specific app. And then, the users of the app see the default email template editor page.

Enable the enhanced email selection dialog

1. In Power Apps, select the environment the environment that contains your solution.
2. Select **Solutions**, and then select the solution in which you want to turn on the enhanced template editing experience.

   > **Note**
   >
   > Don't select the default solution to configure the template.

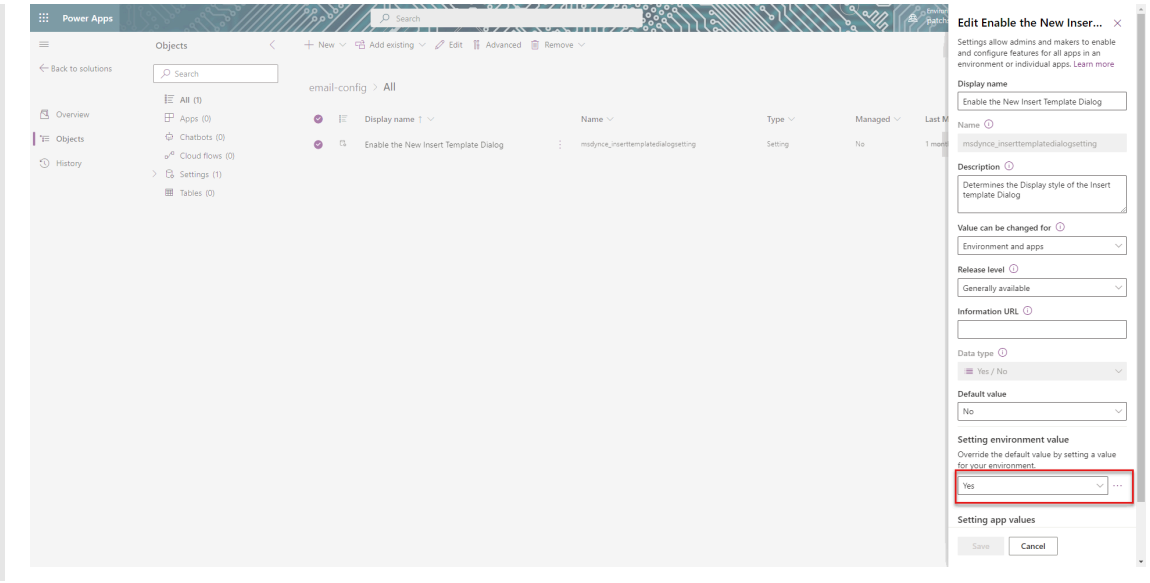
3. Select **Add Existing** > **More** > **Setting**.



4. On the **Add existing Setting Definition** pane, select the **Enable the New Insert Template Dialog** option and then select **Next**.

5. Select **Add** on the **Selected Setting Definition**. The **Enable the New Insert Template Dialog** option is added to your solution. Select **Edit**.

6. Set the **Setting environment value** option to **Yes** on the **Edit Enable the New Insert Template Dialog** pane.



7. Select **Publish All Customizations**.

Disable the enhanced email selection dialog for an app

For an app to display the default email template selection dialog, you must disable the enhanced email template selection option. To disable the option for a specific app, you must add the app to the Solution in which you've added the email template selection option. Perform the following steps:

1. Go to **Power Apps** instance.
2. Select **Solutions**.
3. On the **Solutions** page, select the Solution in which you've added the email template selection option.
4. Go to **Add Existing** > **App** > **Model-driven app**> **Add existing model-driven apps** pane. Select the app for which you want to disable the enhanced insert email template selection dialog. The app is added to the solution.
5. Select the **Enable the New Insert Template Dialog** option in the solution.
6. On the **Edit Enable the New Insert Template Dialog**, in the **Setting app value** section, the selected app is displayed.

7. Select **New app value** for the app, and select **No** for the specified app.
8. Select **Save** and **Publish All Customizations**.

See also

How to create an email template in model-driven apps
Customize an email template using the template editor

Feedback

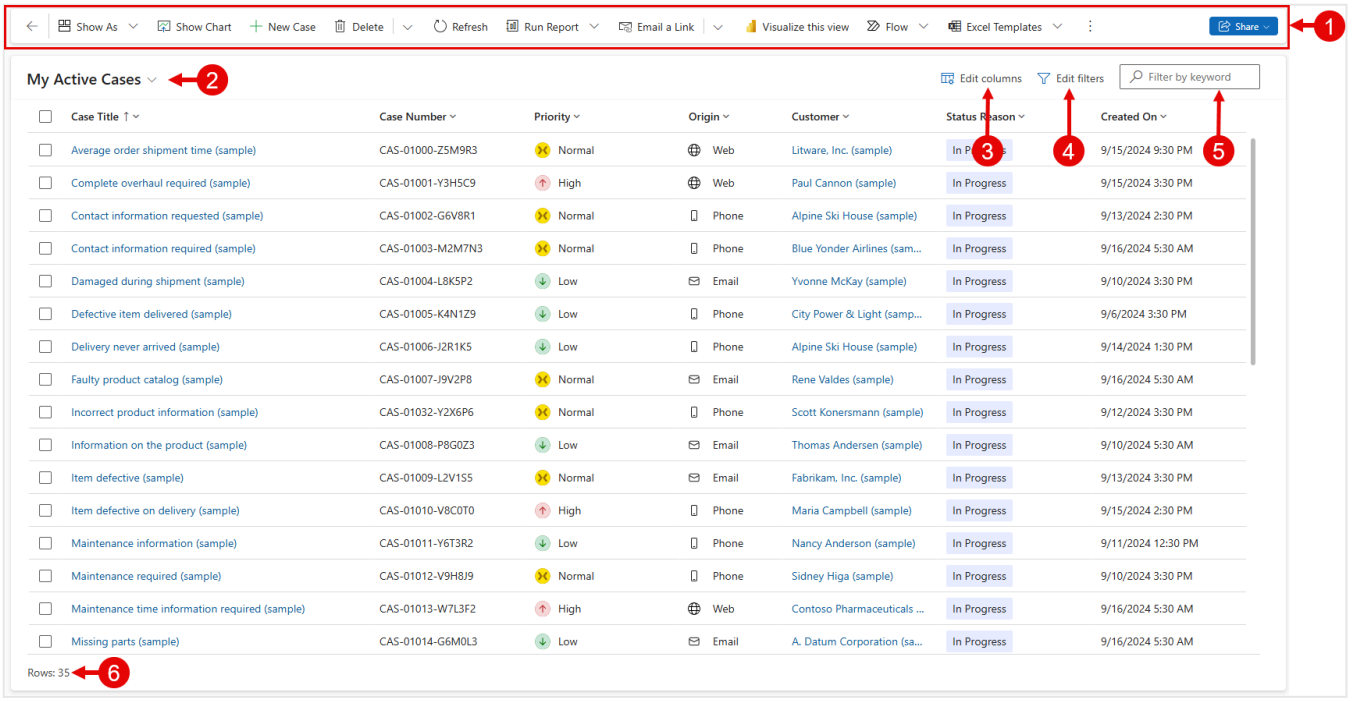
Was this page helpful? Yes No

Provide product feedback

Explore data on a grid page

06/30/2025

In model-driven apps, the default control that's used to show data in tabular form is the read-only grid. When a subarea contains a table, the subarea's default layout in the app sitemap is called the grid page.



Legend:

1. **Command bar**: The commands available for the page
2. **View selector**: Select to change views
3. **Edit columns**: Select to add, remove, or reorder columns
4. **Edit filters**: Select to filter the view with advanced queries
5. **Filter by keyword** or **Quick find**: Show only the rows in the current view that match your search criteria
6. **Row counter**: Shows total number of rows

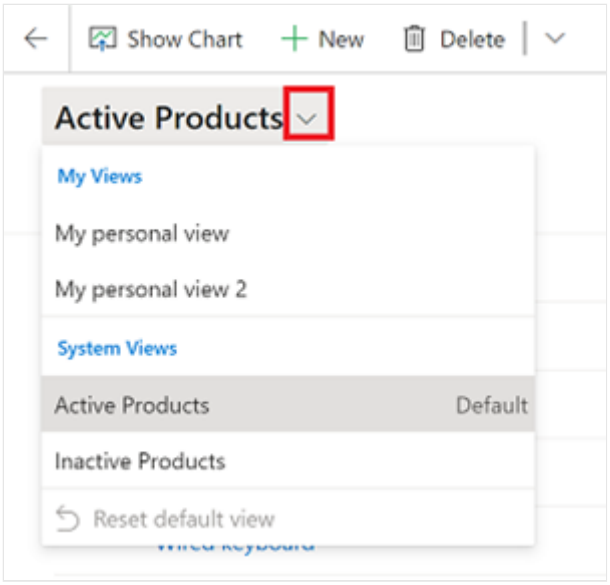
Command bar

The command bar provides quick access to various commands and actions within an application.

View selector

The view selector lists the views that are available for the table. By default, the list is divided into two groups:

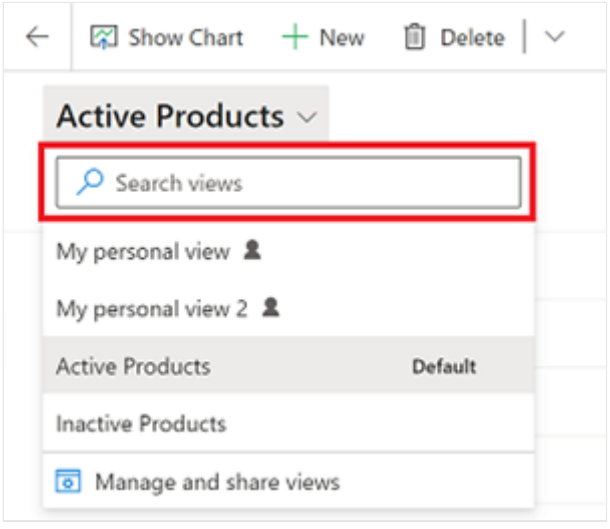
- **My Views**: Personal views that you created
- **System Views**: Views that your administrator created and shared with you



Personal views are listed before system views by default. Both lists are sorted alphabetically. You can change the order the views are listed in.

> **Note**
>
> If your administrator has turned on the modern advanced find experience, the list of views isn't divided. To easily tell which kind of view you're looking at, personal views have a user icon. You can also hover over the information icon to see the view type.

To quickly find a view in a long list, use the search box at the top of the view selector.



Every table in your app has a default view that's set by your administrator. The **Default** label in the view selector identifies which view is the default for the table.

Column editor

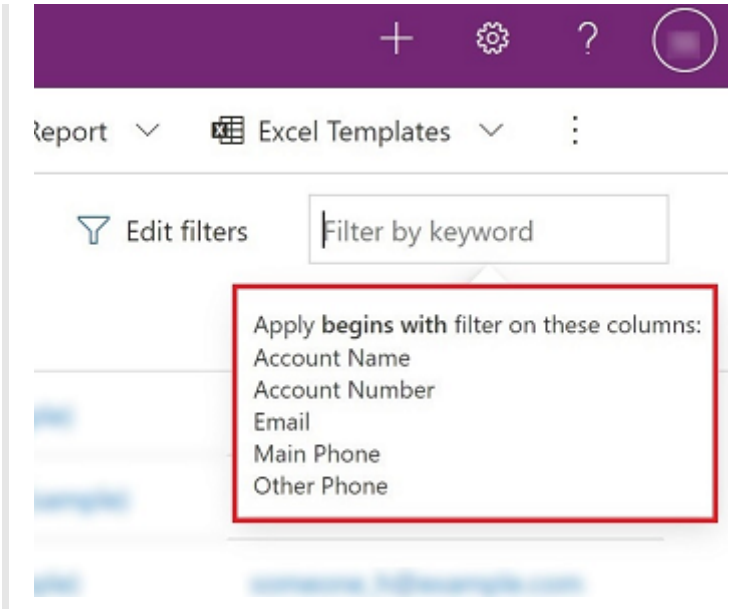
Use the column editor to add, remove, or reorder columns.

Filter editor

Use the filter editor to see or modify the query that selected the records in the view.

Grid search

Grid search enables you to filter the current view based on a keyword. When you choose the search box, an information tooltip appears to indicate the list of columns on which a begins-with match is applied.



The search box can offer two different experiences, depending on how your administrator has configured search.

> **Note**
>
> If your app is configured to be offline by default, the search box will utilize **Quick find** regardless of the grid search configuration in place.

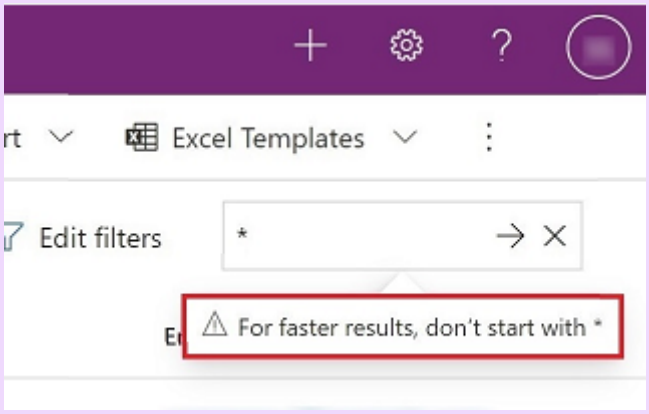
Filter by keyword

Filter by keyword is the default experience. When you enter text in the **Filter by keyword** box, the system filters the current view. It displays the results in the same columns that are used in the current view.

Quick find

If your administrator has turned on quick find, then the search box shows **Quick find**. When you enter text in the **Quick find** box, the system filters the table's quick find view. It displays the results in the columns of the quick find view rather than the current view.

> **Note**
>
> Grid search keywords that begin with an asterisk (*) will display a warning informing you that the search might be slower. If your administrator has turned on the Prevent slow keyword filter for quick find terms on view pages setting, then you cannot perform a search on keywords that begin with an asterisk (*).



Find and visualize data

Find data with AI and visualize data with AI using the natural language text input box.

Row counter

Row counter displays the total number of rows and indicates infinite scrolling on the page. This is the default view available.

Record counter

Record counter shows the position of the current page of records within the total number of records in the view. This is typically shown as a record counter, such as "Records 21-40 of 100".

To view the record counter, your administrator must enable the settings.

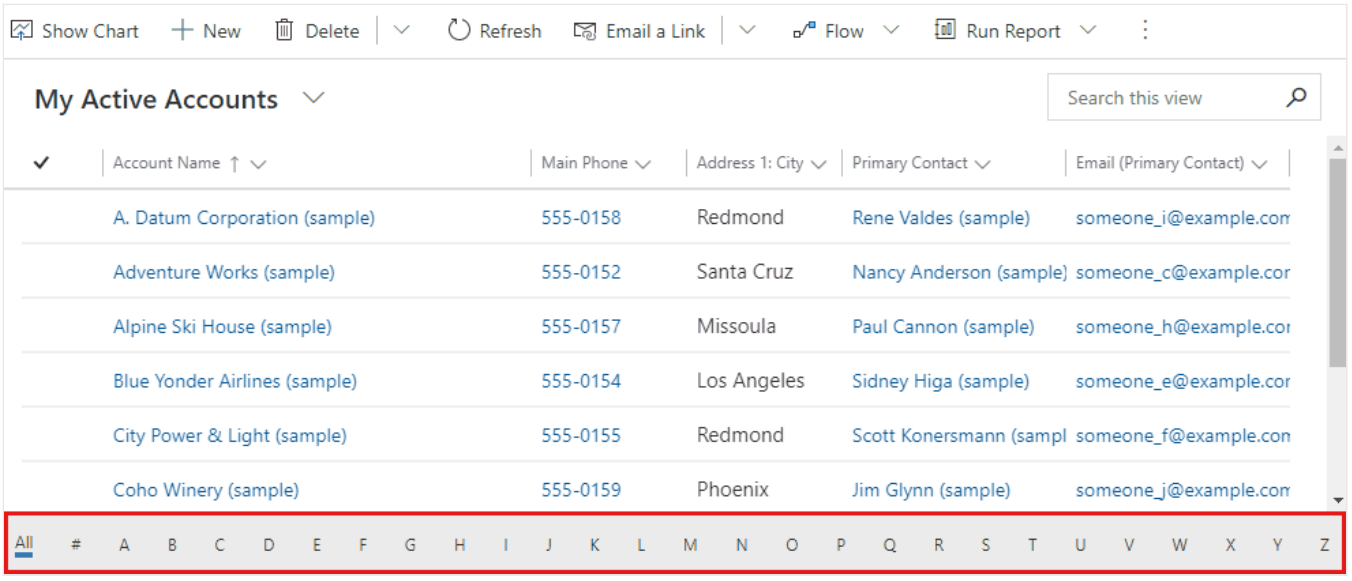
Page navigation

Page navigation allows you to move forward or backward by page in the set of records shown in the view and see the position of each page within the total dataset. To view the page navigation, your administrator must enable the settings.

Jump bar

The jump bar is a quick way to filter out records based on letters of the alphabet. To see only records that start with the letter *S*, for instance, select **S** in the jump bar.

The jump bar acts on the first sorted column. If the sort order hasn't changed, the jump bar uses the primary column. To view the jump bar, your administrator must enable the settings.



Column header actions

Dropdown column headers provide various options to let you modify the data shown in the grid.

Sorting

You can modify the order of records in the grid using the sorting actions in the dropdown column headers. By default, selecting to sort by a column replaces any existing sorting criteria on the grid.

> **Note**
>
> Multi-column sorting can be accomplished by holding the **Shift** key when you select a sort on a different column. This adds the sort condition to the existing sorting criteria on the grid, instead of replacing the sorting criteria.

Filtering

To focus on records that meet particular criteria, you can filter a view by one or more columns. The type of data in the column determines the available filter options.

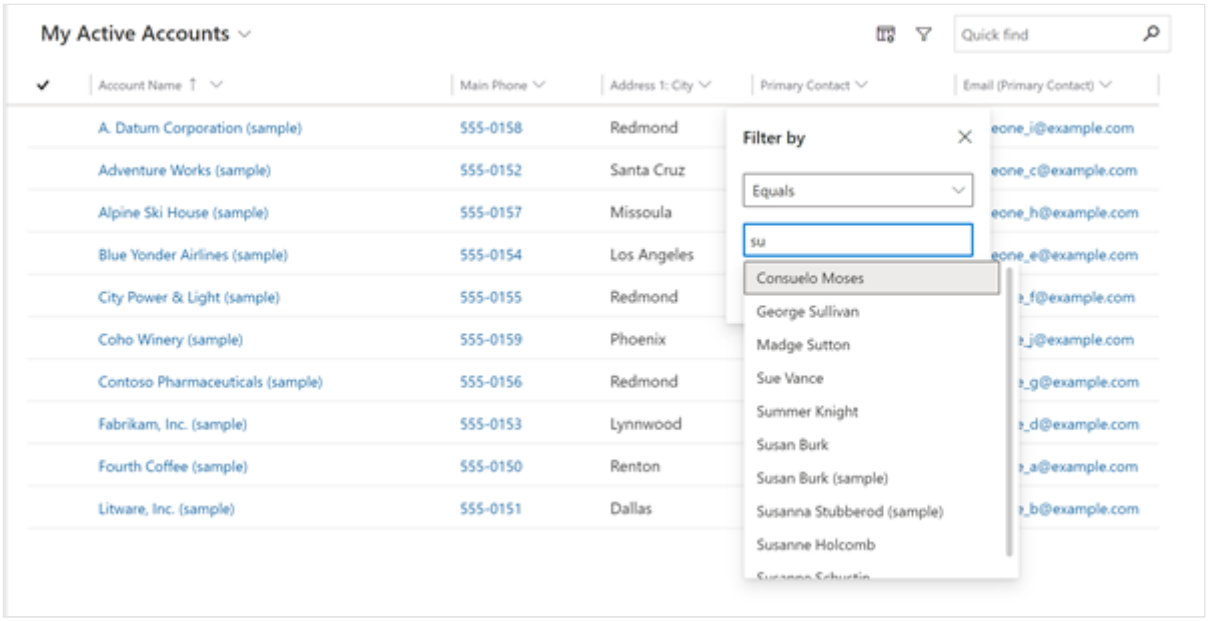
> **Important**
>
> In Unified Interface, grids don't pre-populate column filters based on the current view definition.

> **Note**
>
> Filtering isn't supported on virtual entities. The grid filter's look-up list is limited to 100 results.

Filtering on a lookup column

When you filter data on a lookup column, you can select from a list of values, rather than typing. For example, to filter a view by **Primary Contact**, you can select a name from the list that appears when you select the filter text box.

If you start typing, the filter suggests matching results to help you complete the search. The results include all rows that aren't otherwise filtered out.

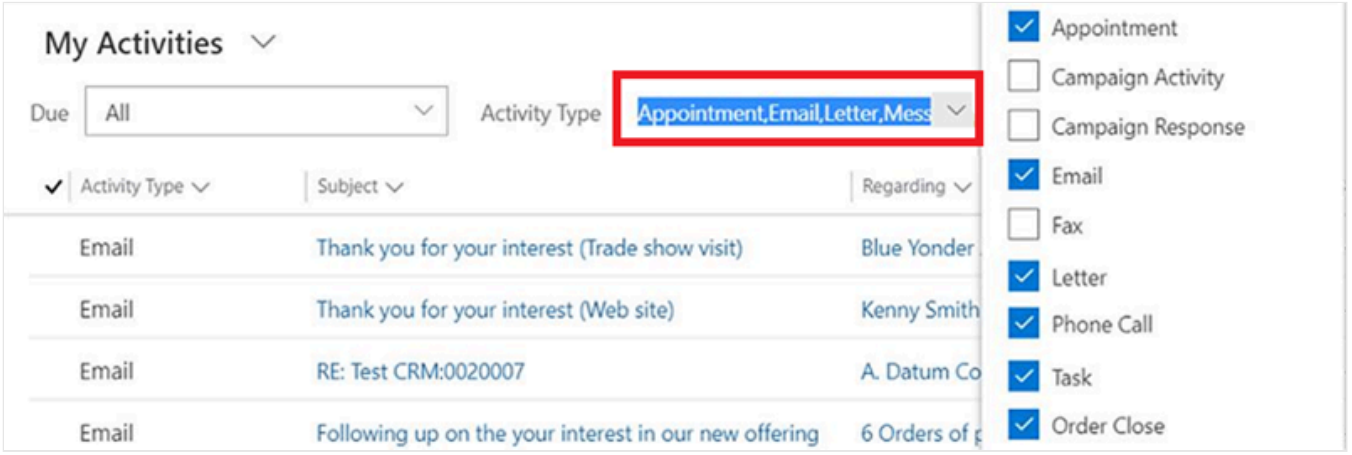


Filtering on a date column

When you filter on a date column, you can select from many different ways to specify a date or a time frame. For example, you can search **On or after** an exact date, **Next fiscal year**, or **Last week**.

Filtering activities

Use the activity type filter to show only the activities you're interested in, such as appointments or email.



Grouping

When configured, the dropdown column headers surface a **Group by** option. This allows you to group the data in the grid by values in the selected group.

- Grouping is available for most kinds of columns. This notably doesn't include lookup columns.

- When grouping on date fields, you need to choose a timeframe to on which to group. This includes day, week, month, quarter, and year.
- You can only group by one column at a time; selecting **Group by** on a different column replaces any existing grouping criteria.
- You can remove the grouping by selecting **Ungroup** from the same dropdown column header to remove the grouping.

> ⓘ **Note**
>
> Grouping criteria isn't saved to a view. Grouping is not supported on nested grids or when paging on the grid is allowed.

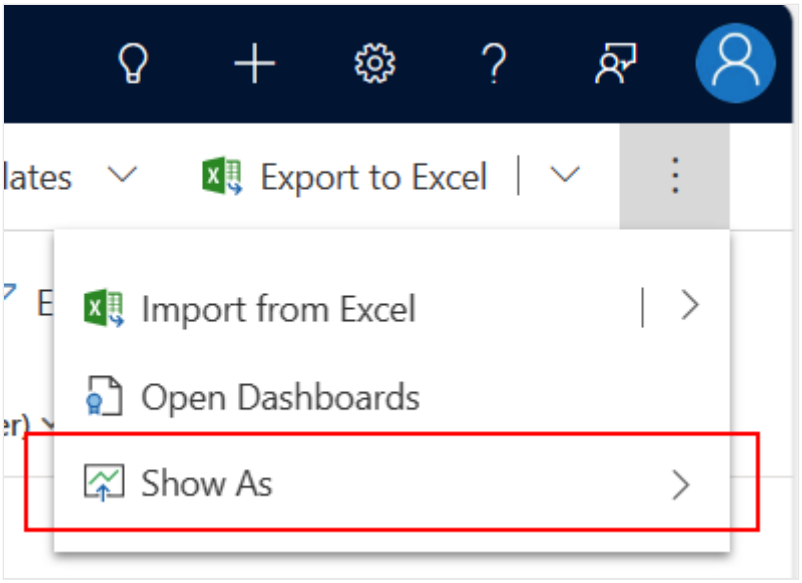
Aggregation

When configured, the dropdown column headers surface a **Totals** option in numeric columns. This allows you to surface an aggregate calculation in the grid footer for the data in this column. You can choose between showing a sum, the maximum value, the minimum value, or the average value. If the data in the grid has been grouped, the aggregate is calculated and displayed for each group.

> ⓘ **Note**
>
> Aggregations aren't saved to a view.

Show as

While the default read-only grid is initially used to show data, if multiple dataset controls are configured for the table view, you can switch between these controls using the **Show as** menu in the command bar, which is typically found under the **More commands** action. Once you've switched to a different display of the data, that control is remembered for future sessions on that device.


lates
Export to Excel
Import from Excel
Open Dashboards
Show As

Create and manage personal views on a grid page

Article • 03/08/2024

In model-driven apps, views define the data that's displayed in tabular form on a grid page. System views are views that are created by your administrator and shared with you. You typically can't modify them. However, you can create and manage personal views. Personal views are available to you only unless you share them.

A view consists of the following elements:

- The columns to display
- Default sorting options
- Filters to show rows that match specific criteria, if needed

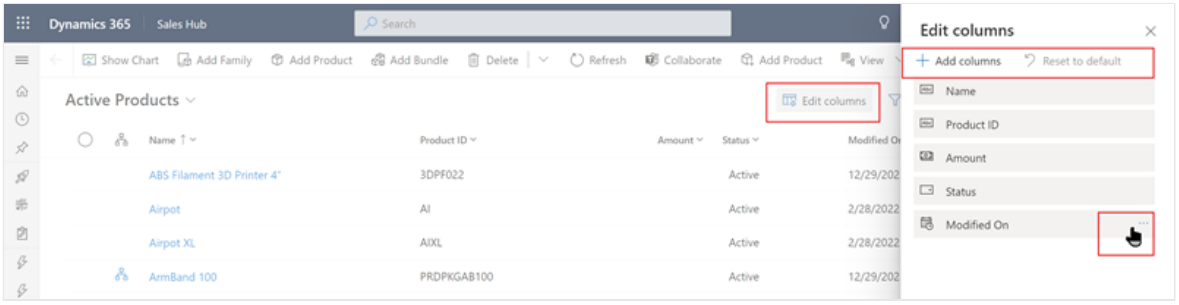
> **Important**
>
> To ensure proper functionality of all views, it's recommended to create all views with the View designer in the maker portal or the Edit filters panel in the view page. Custom views imported from solutions can include invalid fetchXML and might not be supported.

For more information, see Understand model-driven app views.

Select columns in the column editor

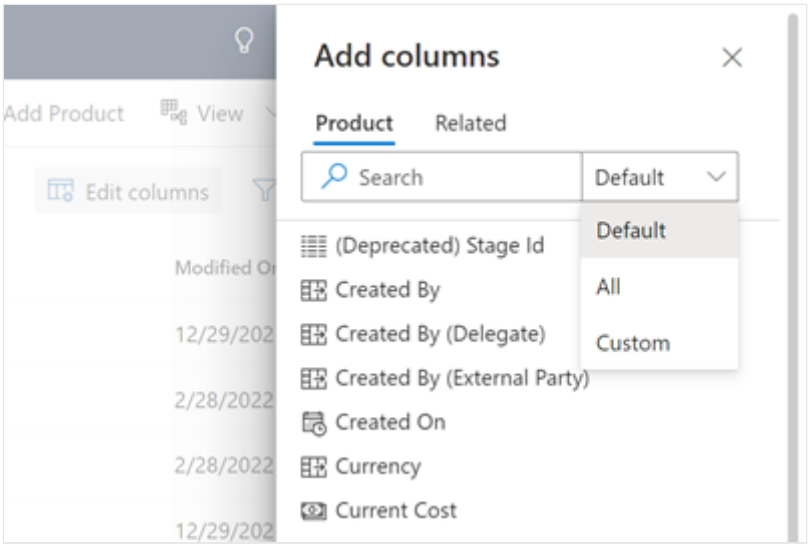
1. Select **Edit columns** to remove, reorder, or add columns to get a clearer view of your data.

2. Select **Apply** to see your changes on the grid page.

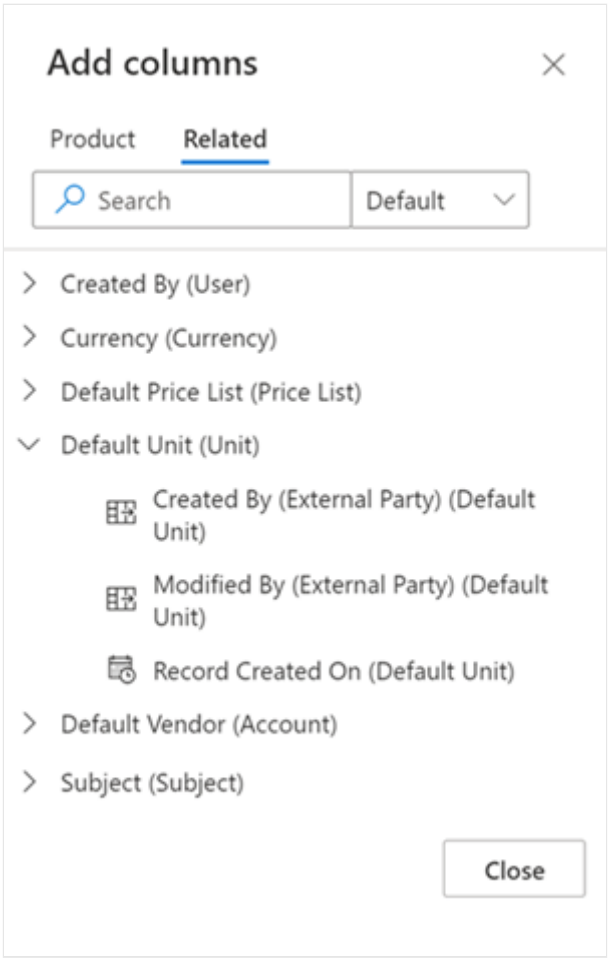


The following actions are available in the column editor:

- To remove a column, hover over the column name, select **...**, and then select **Remove**.
- To change the order of the columns, select **...**, and then select **Move up** or **Move down**. Or, drag the column to the desired position in the list.
- To add columns to the view, select **Add columns**, and then do one or more of the following:
  - Select a column from the default set of system columns in the table.
  - Select **Default**, then select **All** or **Custom**, and then select from the set of columns in the table.
  - Type a column name in the **Search** box to find a specific column, and then select it.



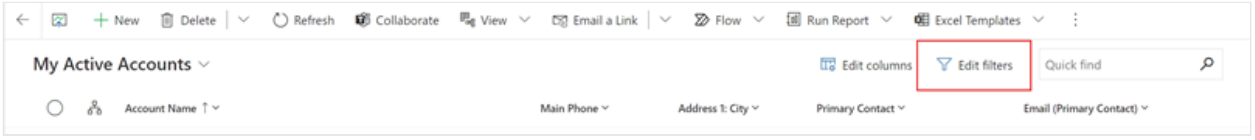
  - To add columns from other tables, select the **Related** tab. Expand the table that contains the column you want to add, and then select the column. When you're done adding columns, select **Close**.



To undo changes to the view's columns at any time, select **Reset to default**.

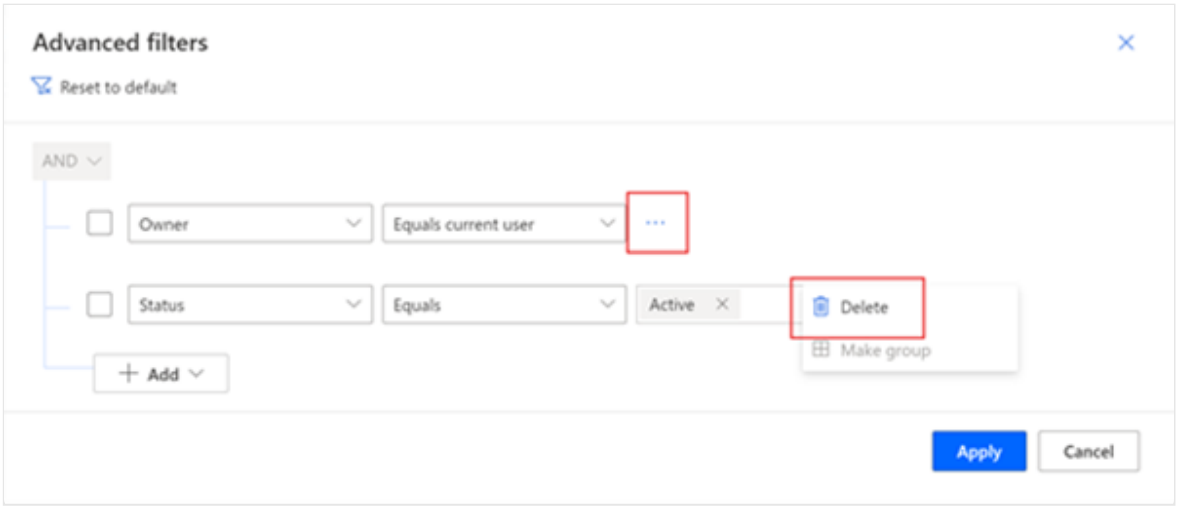
Select filter conditions in the filter editor

Use the filter editor to view or change the query that underlies the data in the current view.



Edit or delete a condition

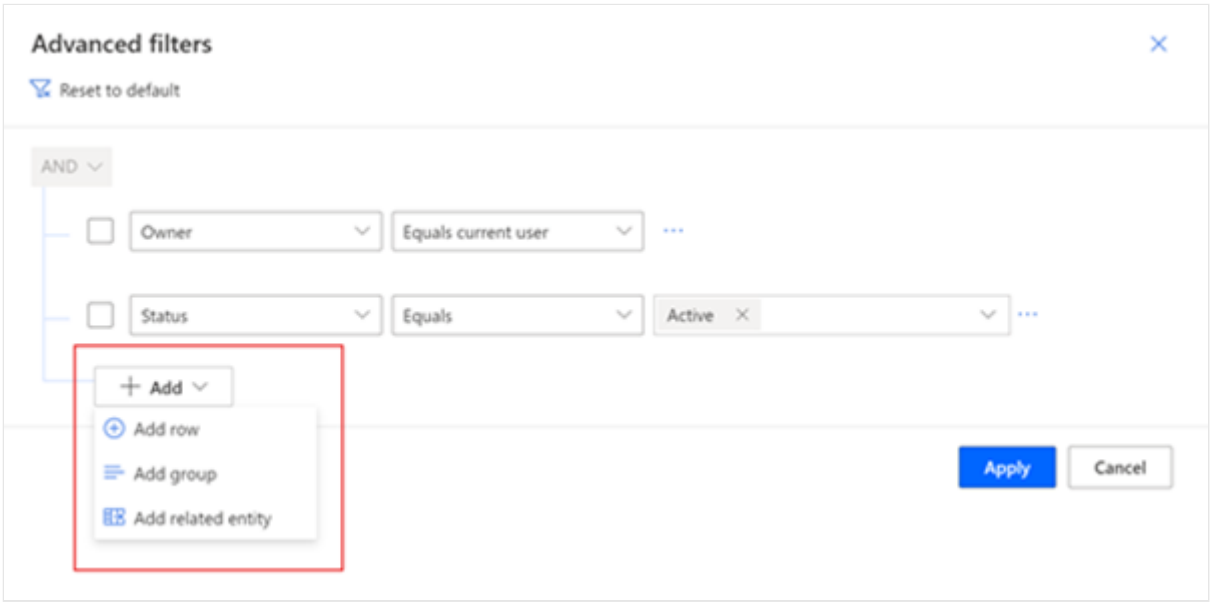
- To change a condition, select the column, operator, or value and edit as needed.
- To remove a condition, select More commands (...) next to the row, and then select **Delete**.



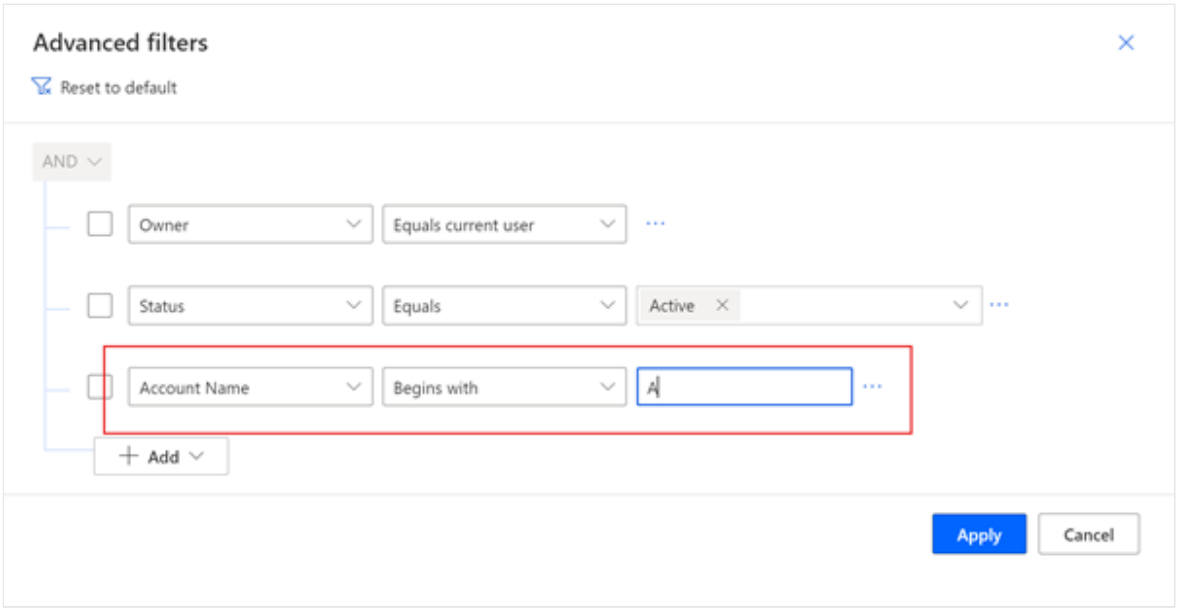
Add conditions

1. To add one or more conditions, select **Add**.
2. When you're finished editing the view filters, select **Apply** to see your changes on the grid page.

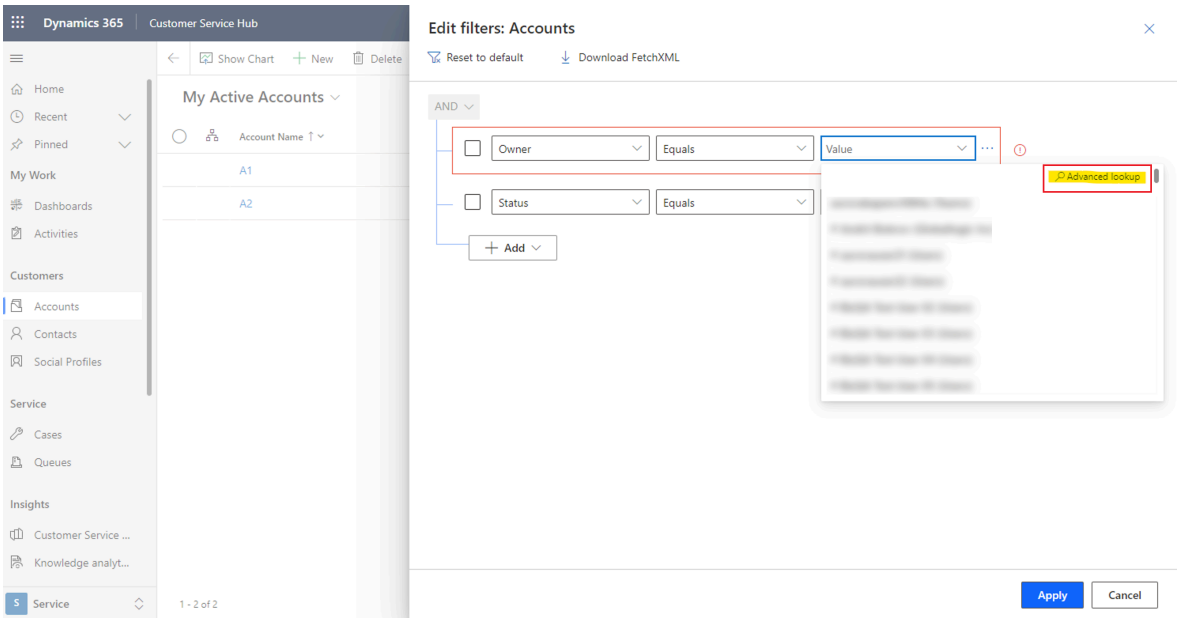
The following options are available when you add conditions in the filter editor:

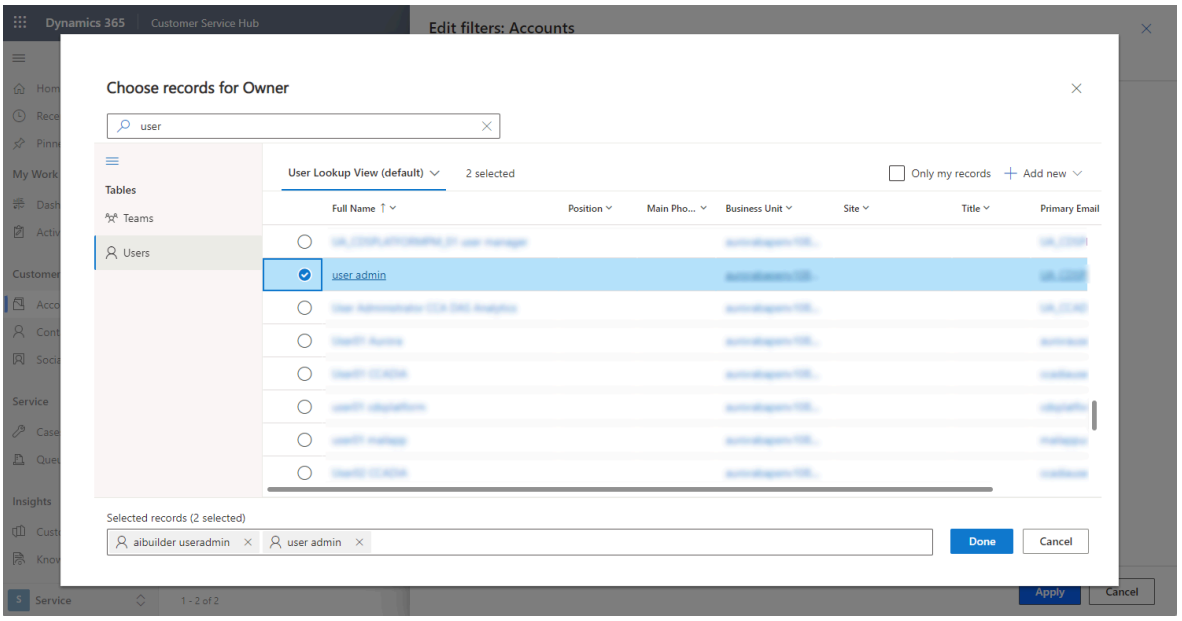


- To add a condition, select **Add row**, and then select a column, an operator, and a value.

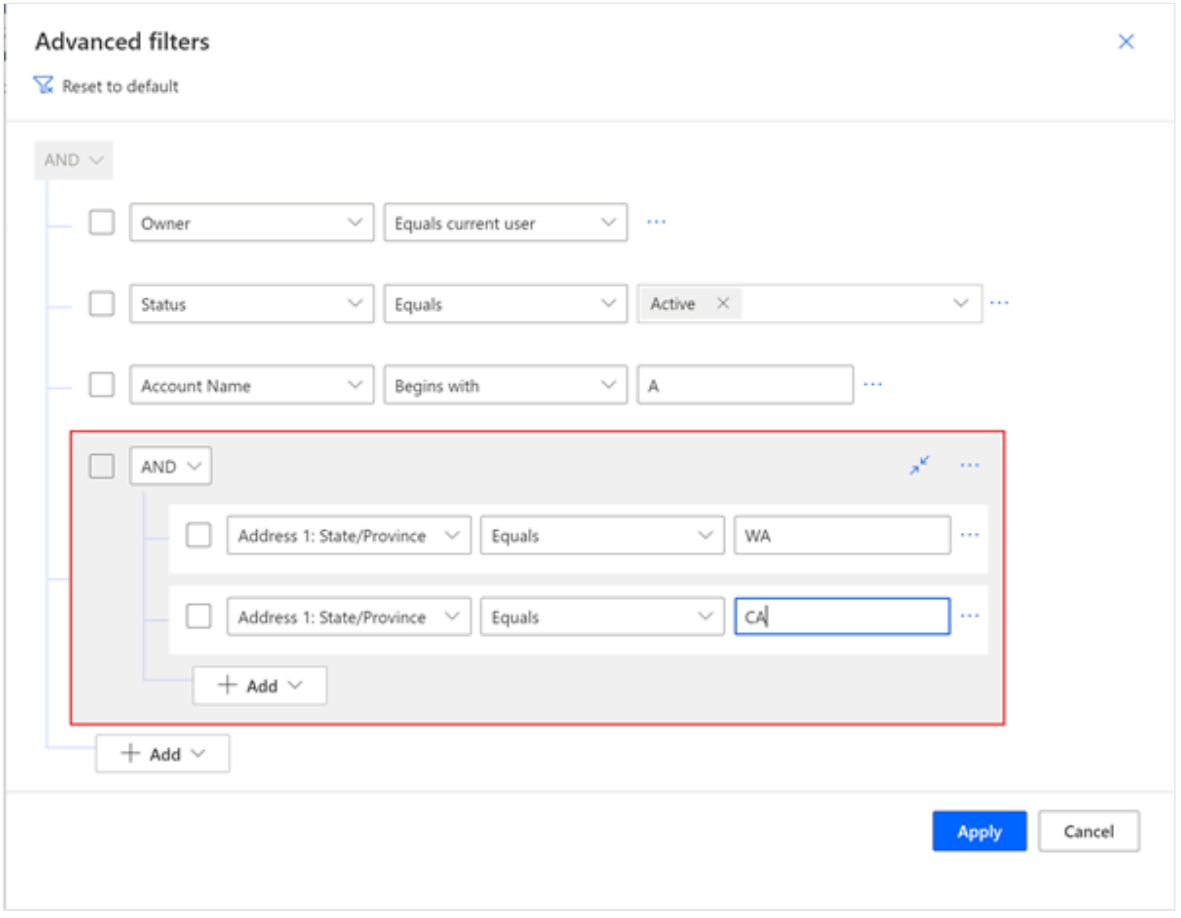


If you're adding a condition to a column with lookups, you can look up the record in the **Value** field. In the lookup drop-down menu, you can choose **Advanced lookup** to see more fields and search for records within a specific view.



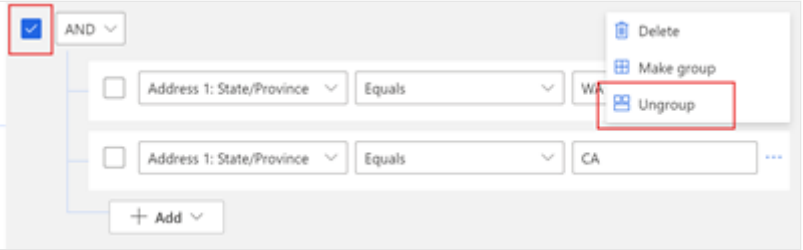


- To add a group of conditions, select **Add group**. You can create multiple subconditions and group them using AND and OR operators to create an expression.

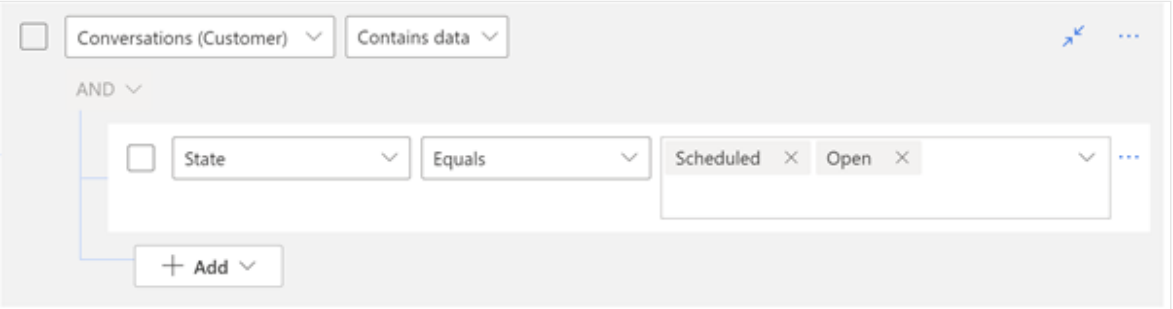


- To add a single condition to a group, select the condition checkbox, then select **...**, and then select **Make group**.

- To separate a group of conditions, select the group's checkbox, then select **...**, and then select **Ungroup**.



- To add a condition based on a column in a related table, select **Add related entity**, and then select a table. **Contains data** is the only conditional operator that can be used with a related table in a view filter. Then, select a column in the related table, an operator, and a value.



To undo changes to the view filter at any time, select **Reset to default**.

> **Tip**
>
> If you have the security privilege **Enable Download Fetch XML**, found under Miscellaneous Privileges section of the **Customization** tab within the security role definition page, you can download the FetchXML expression for a view. Contact your administrator if you need this capability.

Sort the data

Select a column heading, and then select how the view should sort data in the column. You can sort on one column at a time.

Sorting options depend on the type of data. Alphanumeric columns can be sorted in A to Z (ascending) or Z to A (descending) order. Date columns can be sorted in older to newer (ascending) or newer to older (descending) order. An arrow to the right of the column name indicates the view is sorted on that column. The direction the arrow is pointing indicates whether the column is sorted in ascending or descending order.

Edit column width

If your administrator has turned on modern advanced find, you can drag the vertical column separator bar to change the **minimum** width of a column. When you change the width of a column, as asterisk appears beside the view name to indicate that you edited the view and haven't saved it. See Save your edits to as a new view to learn more about different options to save a view.

> **Tip**
>
> Since the column width edits change the **minimum** width of the column, you might see column widths getting adjusted after the view is saved, to fill the available horizontal screen space.

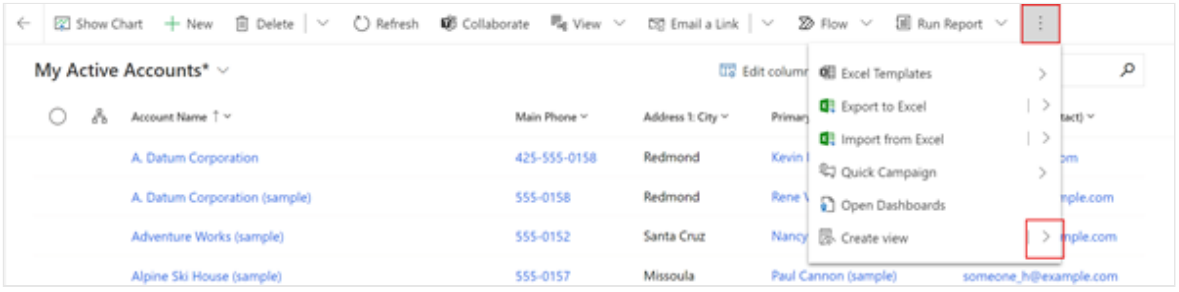
Save your view

An asterisk beside the view name indicates you edited the view and haven't saved it. If you switch to another view or leave the page, the view will revert to its original definition. To keep your changes and add the new view to the list of views available to you, be sure to save it.

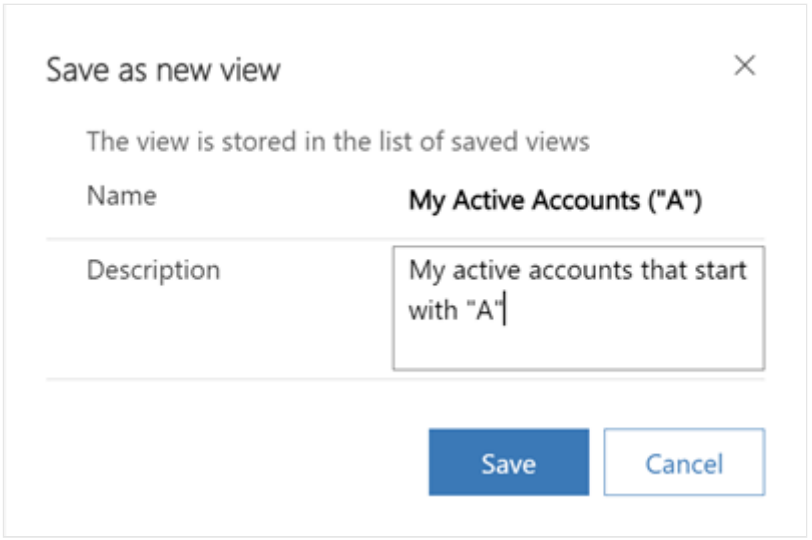
Save your edits as a new view

To keep the original view, save your edited version as a new view with a different name. If you made changes to a view you don't have *Write* permission to, whether it's a system view or a personal view, then you can only save it as a new view.

1. On the command bar, select **More commands** (⋮) > **Create view** > **Save as new view**.



2. Enter a new name for the view, and a description if you like, and then select **Save**.

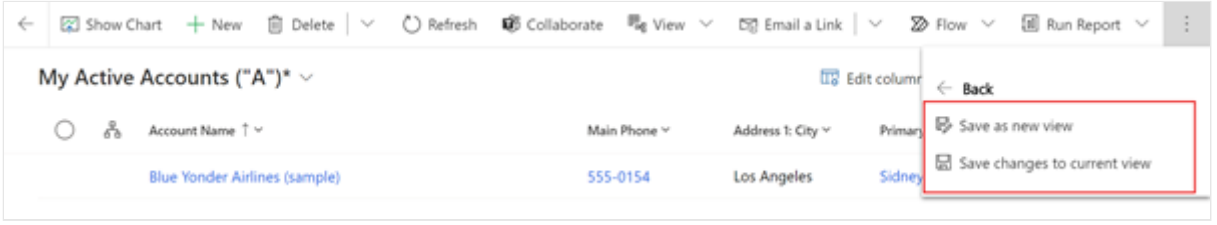


> **Note**
>
> If you want to keep the original view, be sure to give your edited view a different name. If you don't, your changes will overwrite the existing view without confirmation, even though you selected to save it as a new view.

Change the definition of an existing view

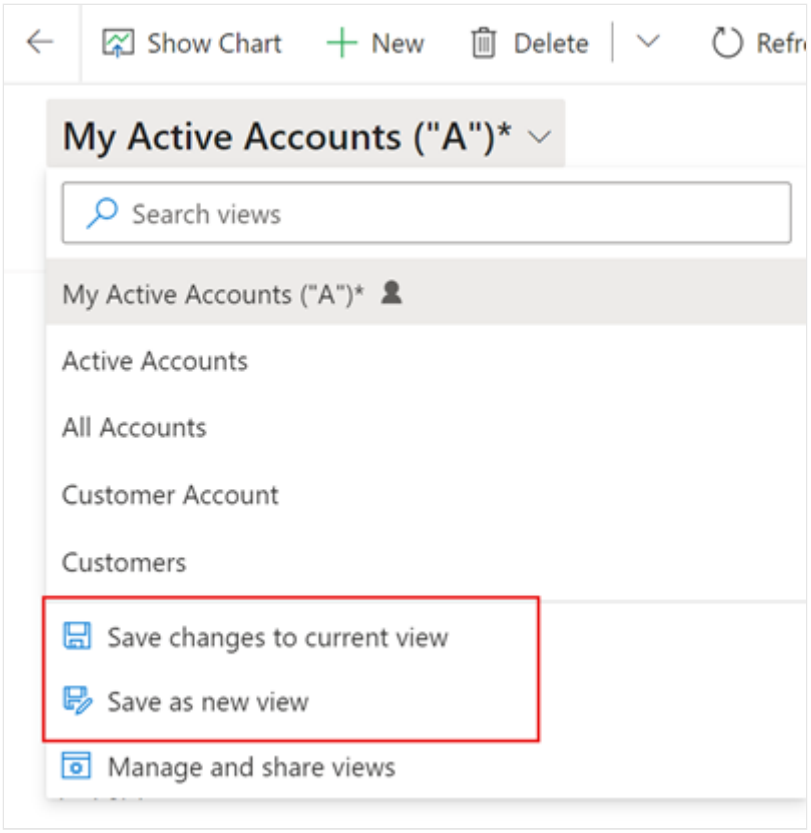
If you don't want to keep the original view, you can replace it with your edited version. You'll need *Write* permission to the view to overwrite the original definition with your changes.

On the command bar, select **More commands** (⋮) > **Create view** > **Save changes to current view**.



Save changes to views in modern advanced find

If your administrator has turned on modern advanced find, you can create and manage views directly from the view selector.



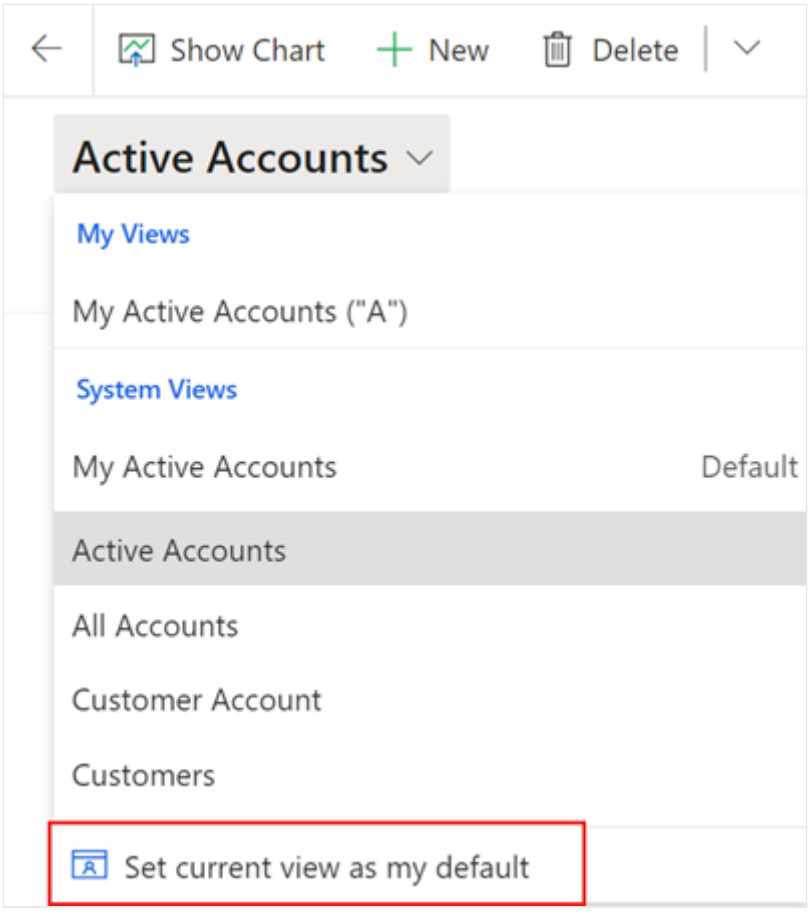
Change the default view

The default view for a table is indicated by the **Default** label in the view selector. Your administrator sets a default view for each table in your app. However, you can set a different view as your personal default.

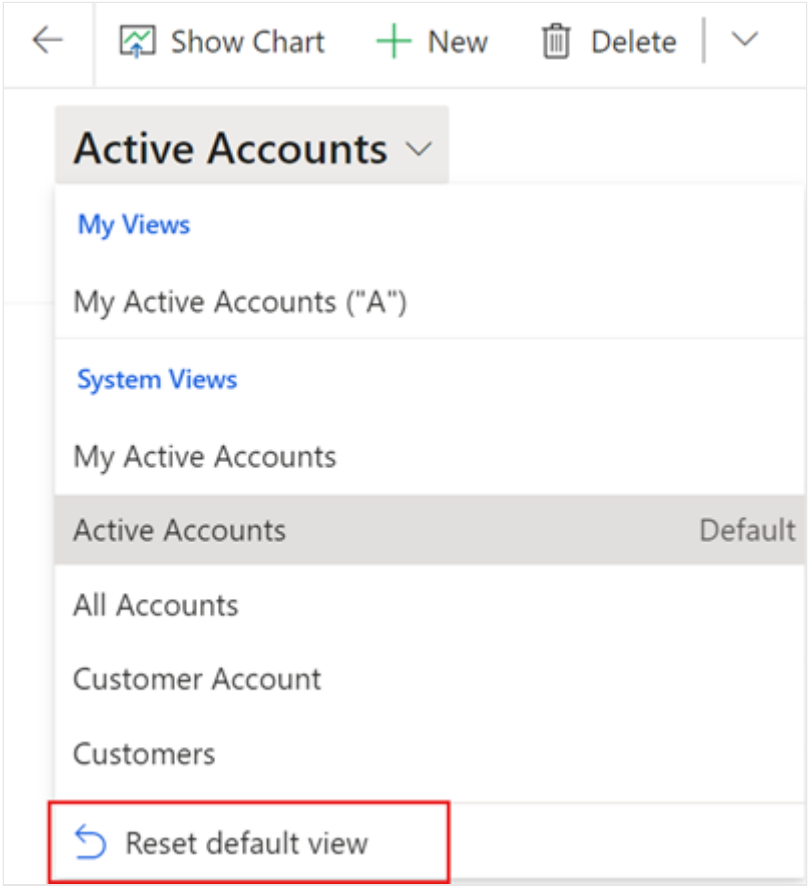
In the following example, the administrator set the view *My Active Accounts* as the default for the Accounts table. To change your default view to *Active Accounts*:

1. In the view selector, select **Active Accounts** to apply the view to the grid page.

2. Open the view selector again and select **Set current view as my default**.



To change your personal default view back to the view that was set by your administrator, select **Reset default view** in the view selector.

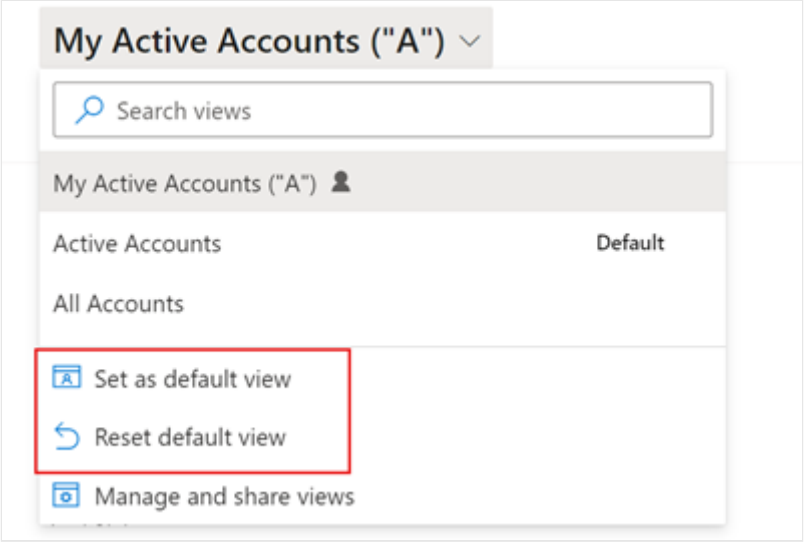


Note

If you don't see **Reset default view**, it means the current default view for the table is the one set by the administrator.

Change the default view in modern advanced find

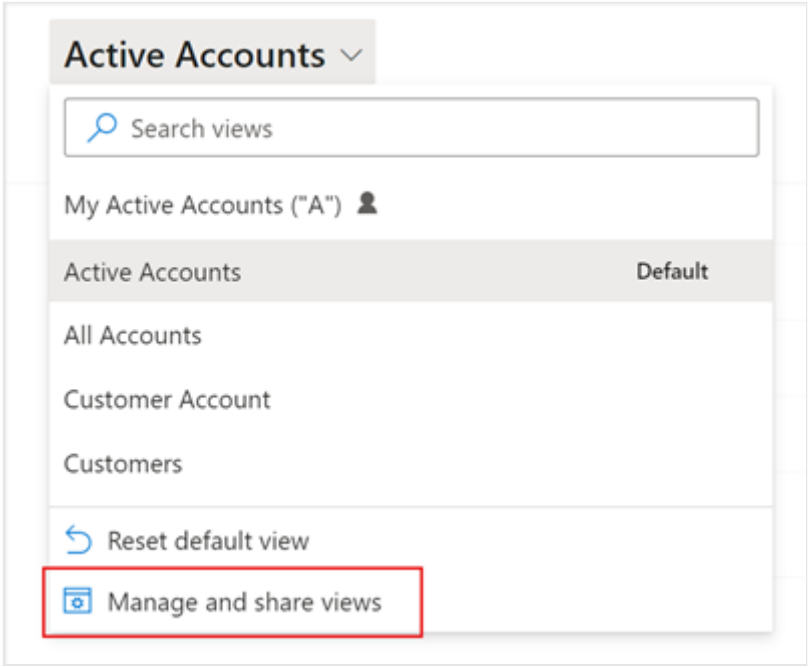
If your administrator has turned on modern advanced find, you can change and reset the default view directly from the view selector.



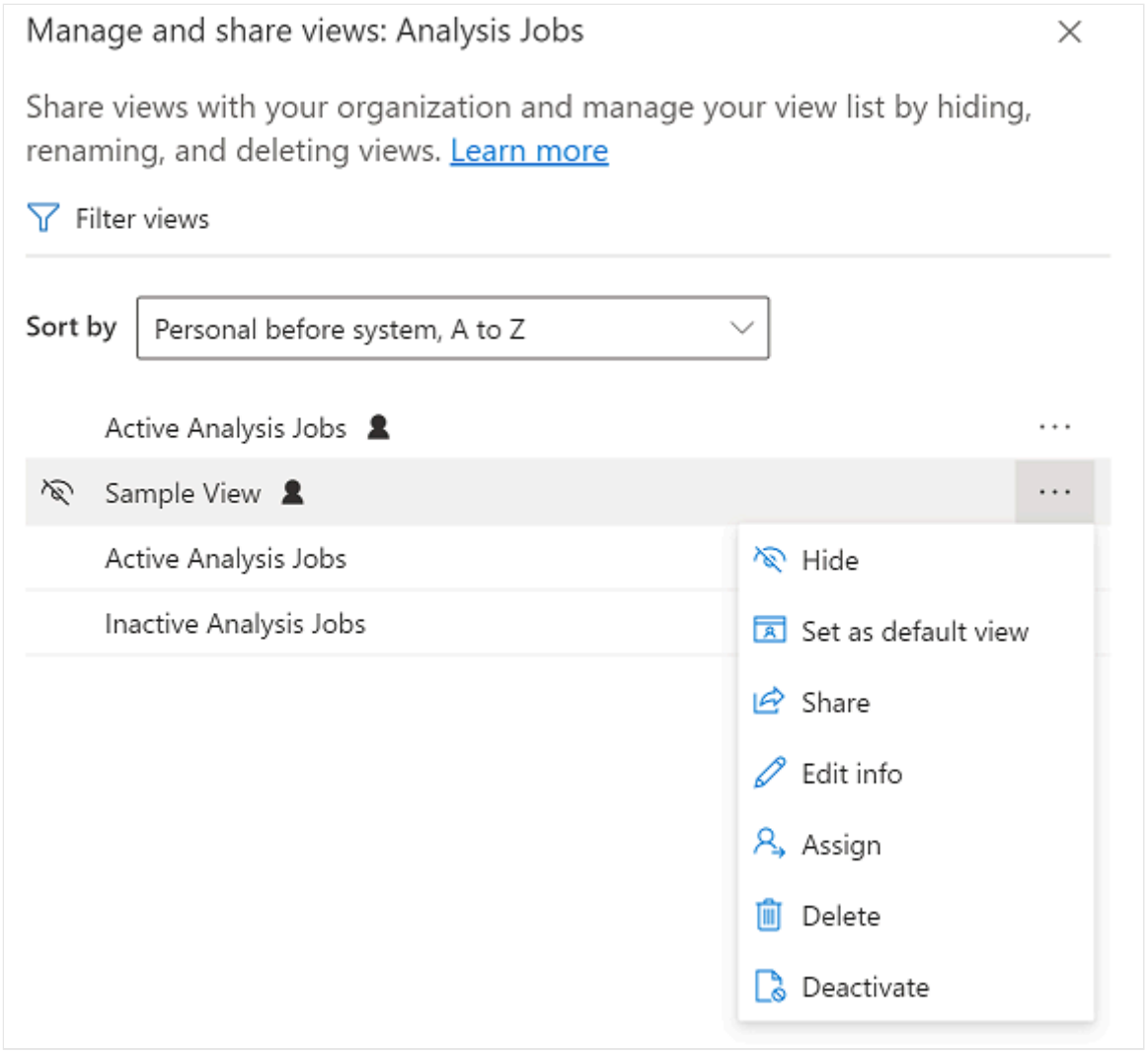
Manage and share personal views

If your administrator has turned on modern advanced find, you can share, hide, rename, and delete views.

1. In the view selector, select **Manage and share views**.



2. Select View commands (...) and then choose your option.

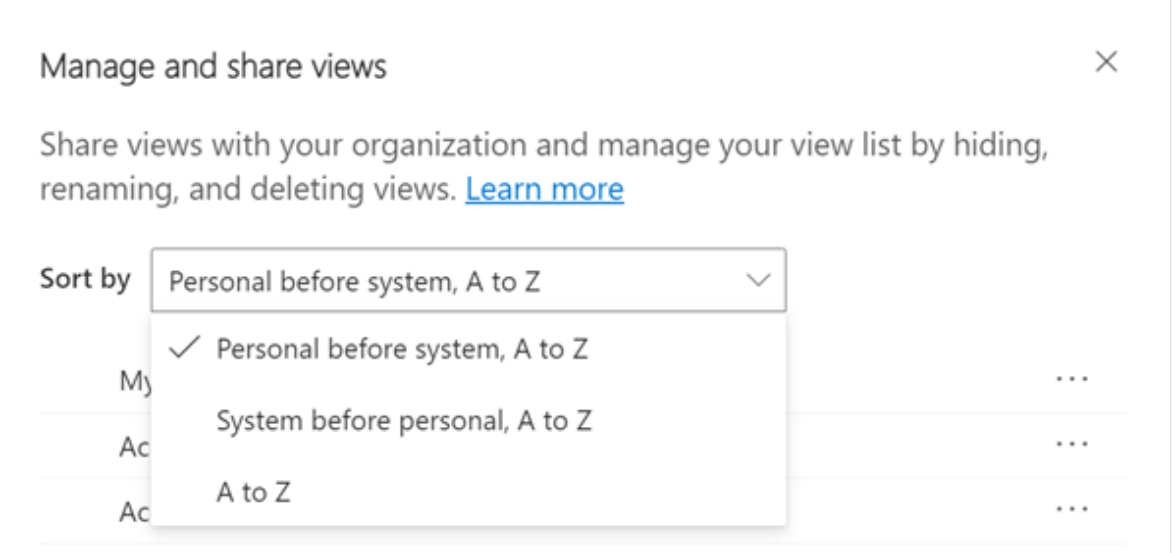


Change the order of views in the view selector

By default, the view selector lists views in two groups, personal views and system views, each ordered alphabetically. You can change the sort order within groups, or remove the grouping entirely and order the entire list alphabetically.

1. In the view selector, select **Manage and share views**.

2. Change the view selector sort order, and then select **Apply**.



The following sorting options are available:

- **Personal before system, A to Z**: This is the default. Both groups are ordered alphabetically. Personal views appear above system views.
- **System before personal, A to Z**: Both groups are ordered alphabetically. System views appear above personal views.
- **A to Z**: The grouping is removed. All views (system and personal) are presented in alphabetical order.

> **Note**
>
> If your administrator has turned on the modern advanced find experience, the list of views isn't divided. To easily tell which kind of view you're looking at, personal views have a user icon. You can also hover over the information icon to see the view type.

Hide views

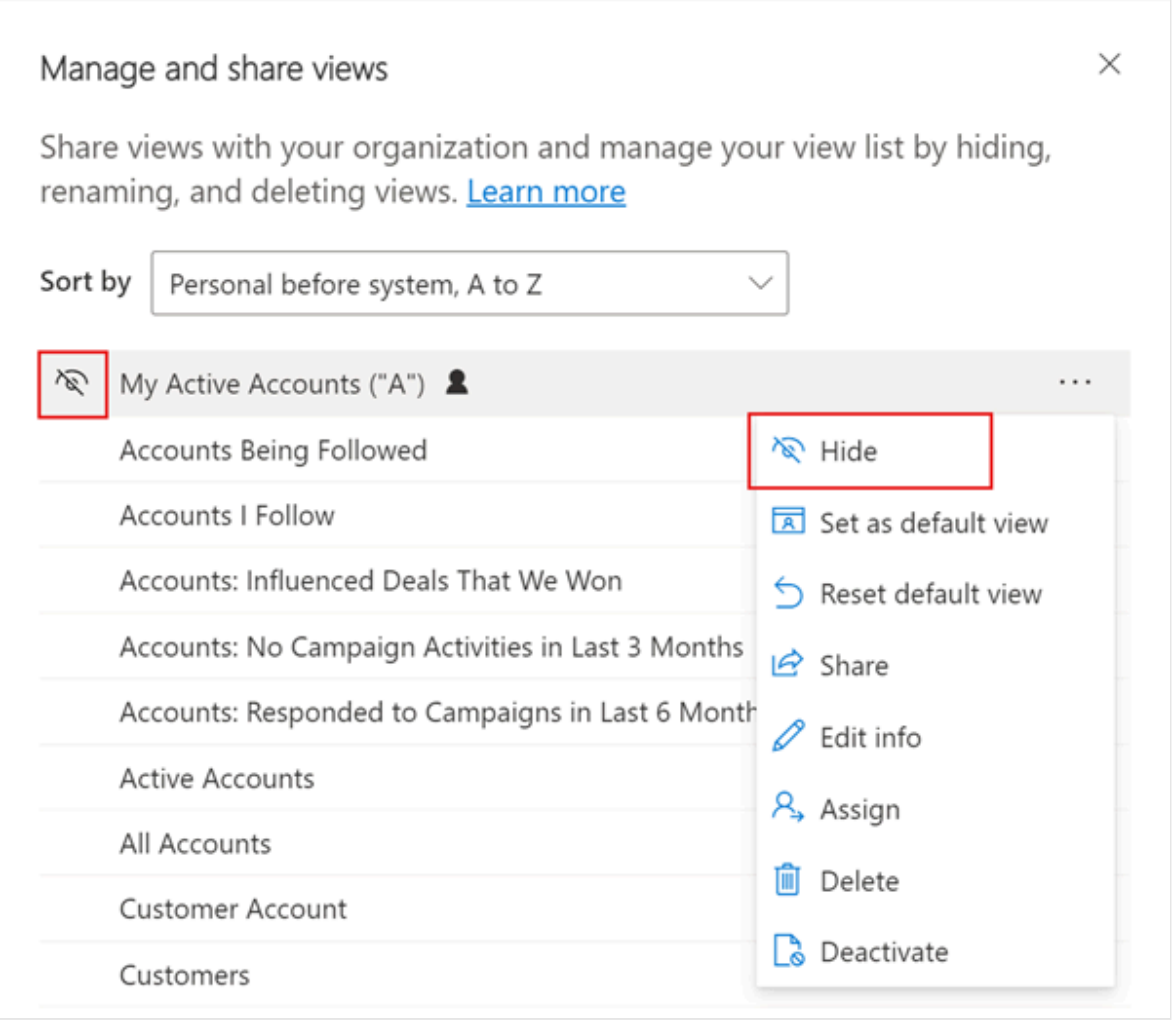
You can hide views in the view selector to personalize the list and reduce clutter. You should keep some things in mind:

- When you hide a view, it's hidden from that table's views in all model-driven apps on all devices.
- If the view is shared with you and other team members, hiding it only hides the view for you. It's not hidden for everyone else who has access to the view.
- You can hide your personal views by default. You can hide system views only if the administrator has allowed it.

- You can't hide a table's default view. If you want to hide the default view, set another view as the default first.
- You can't set a hidden view as the default view for a table.

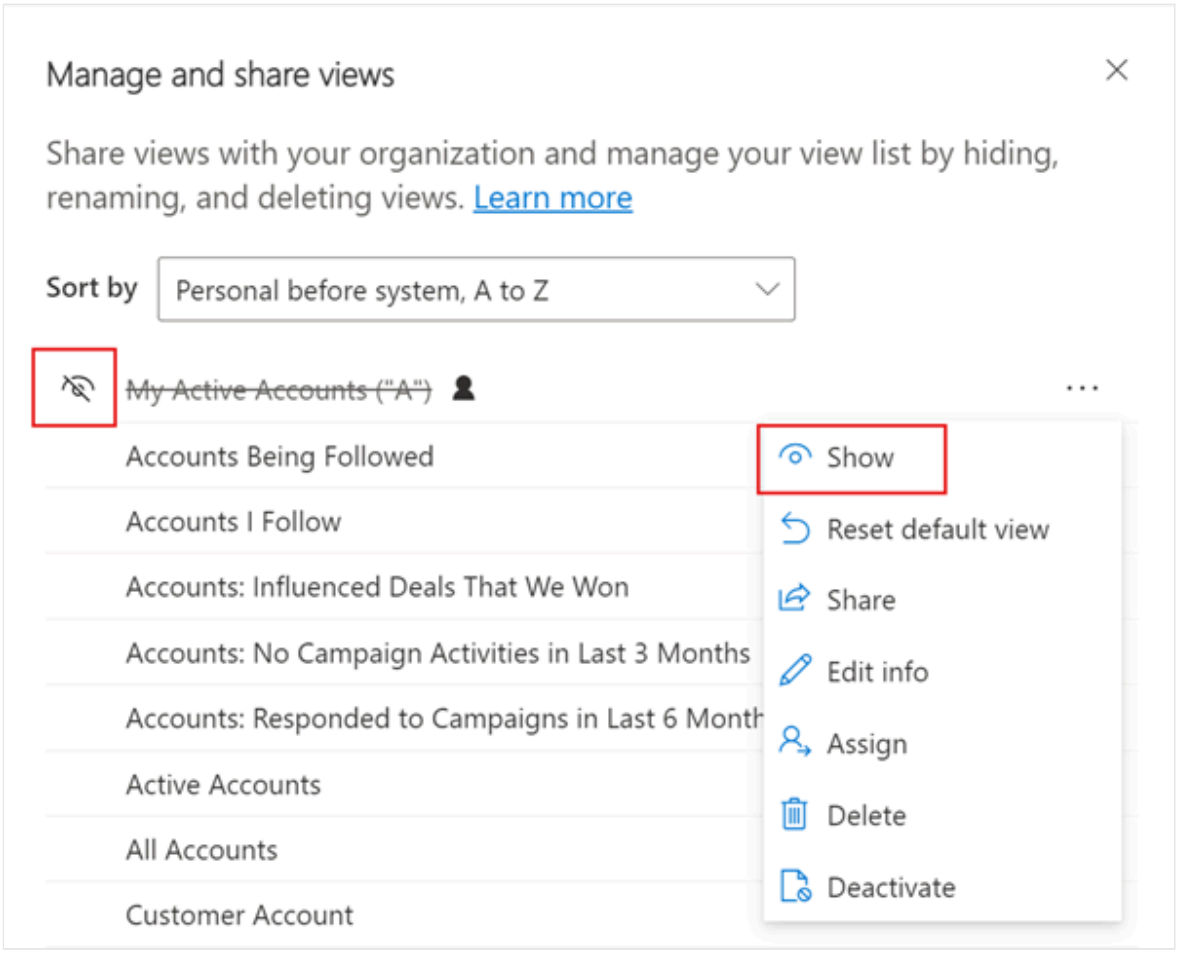
To hide a view:

1. In the view selector, select **Manage and share views**.
2. Hover over the view and select **Hide**. Or, select **View commands** (...) to the right of the view, and then select **Hide**.
3. Select **Apply**.



To restore a hidden view:

1. In the view selector, select **Manage and share views**.
2. Hover over the view and select **Show**. Or, select **View commands** (...) to the right of the view, and then select **Show**.
3. Select **Apply**.



Other view management tasks

- **Set as default**: Makes a view the default view for that table across all model-driven apps in the environment. This option doesn't appear on a hidden view or on a view that's already the default.
- **Reset default**: Changes the default view of a table back to what the administrator set, across all model-driven apps in the environment.
- **Share**: Shares the view with other members of your organization. This option appears only on personal views that you have *Share* permission to. For more information, see Share rows with a user or team.
- **Edit info**: Change the view name and description. This option appears only on personal views that you have *Write* permission to.
- **Assign**: Assigns ownership of the view to another user or team in your organization. This option appears only on personal views that you own.
- **Delete**: Deletes the view. If you delete a personal view, the view is deleted for everyone who has access to it. This option appears only on personal views you have *Delete* permission to.
- **Deactivate/Activate**: Deactivates or activates the view. If you deactivate or activate a personal view, the view becomes inactive or active for everyone who has access to it. This option appears only on personal views you have *Write* permission to.

- **Download Fetch XML**: Download the FetchXML expression for the view. This capability is enabled by an administrator with the security privilege **Enable Download Fetch XML**, found under Miscellaneous Privileges section of the **Customization** tab within the security role definition page.

Feedback

Was this page helpful? Yes No

Provide product feedback

FAQs and known issues about grids and views

Article • 10/14/2022

[This topic is pre-release documentation and is subject to change.]

How many levels of related table columns can I add to a view using the column editor?

You can add one level of related table columns using the column editor (**Edit columns**) on a grid page.

How many levels of related table columns can I add to nested conditions in a view using the filter editor?

You can add up to 10 levels of related table conditions using the filter editor (**Edit filters**) on a grid page.

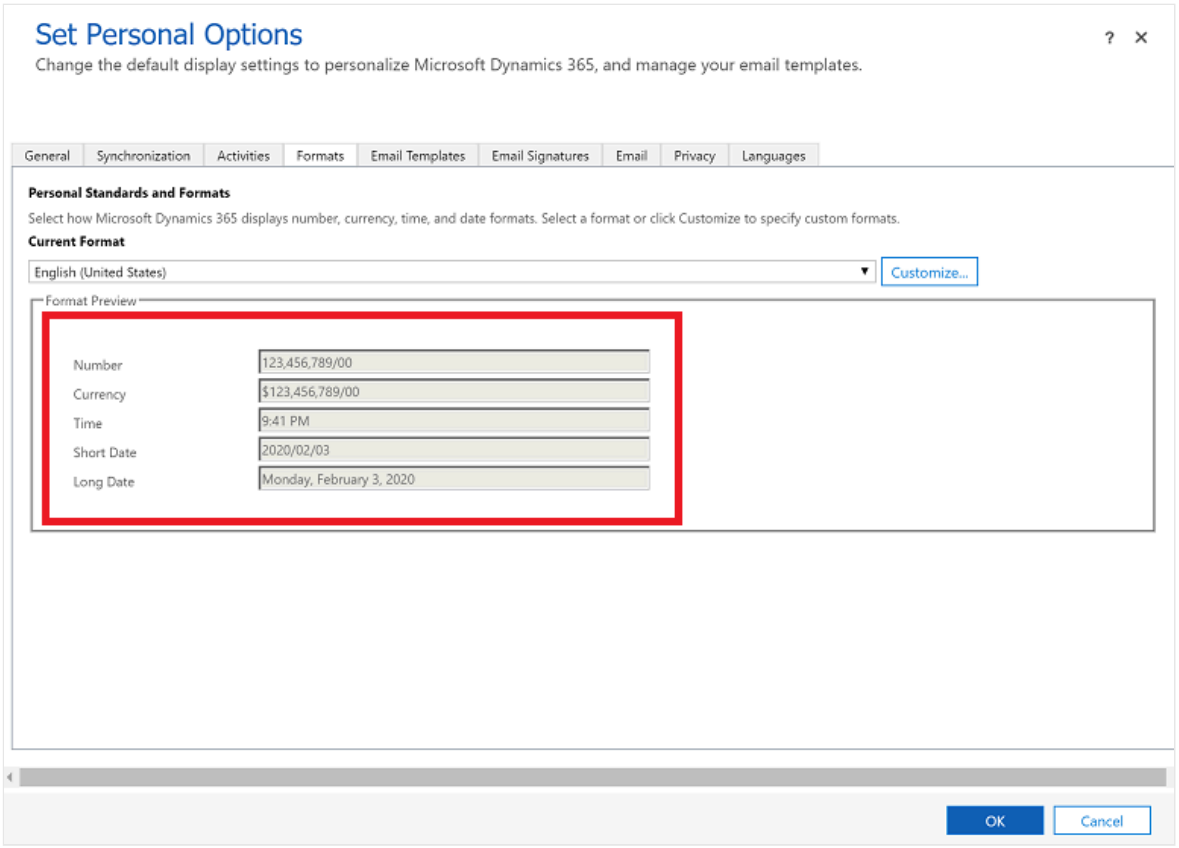
Known issues

- If you change the default display format for number, currency, time, or date and then filter data on a grid, the filter won't show your selected display format. The filters are still displayed in the system default format. In some cases, filtering might not work at all.

  To fix the issue, set the display format for number, currency, time, and date back to the default setting.

  1. In the upper-right corner, select the gear icon  , and then select **Personalization Settings**.

  2. On the **Formats** tab, change the number, currency, time, and date value back to the default setting.



We're working to correct this issue. Check here for updates.

- The advanced filter option on the Queues table doesn't work and displays this error message: *We are unable to display the filter conditions for this view.*
- Public views that include **Contains data** or **Does not contain data** filters don't appear in the list of saved views in legacy advanced find.
- When you open the **Edit filters** panel on certain Activity tables, you will see a condition filtering tables that are not valid for Unified Client. These conditions may not be a part of the FetchXML of the view.
- When you choose the **Search for rows in a table using advanced filters** option or the **Advanced find** button when working in a **Dynamics 365 - Custom** app with a modern, advanced find experience enabled, you might see a loading spinner appear indefinitely.

See also

Explore data in a view on a grid page
Create and manage personal views on a grid page
Advanced find

Feedback

Was this page helpful? Yes No

Provide product feedback

View and manage records in focused view

09/15/2025

Focused view allows you to view and manage account, contact, lead, opportunity, and other records and their associated activities on one page. This view makes it easier for you to stay focused on your tasks and reduces the need to navigate between multiple screens. It also allows you to better organize and prioritize your tasks, thus saving time and effort.

Focused view is now available for all entities, including custom entities. However, your administrator must enable it for you to access it. More information: Enable focused view and set it as the default view.

> **Note**
>
> - Focused view is similar to the sales accelerator workspace view. **Learn how to prioritize your sales pipeline through work lists**.
> - To view the latest updates and features of focused view, you must opt in for early access updates. Contact your administrator to enable early access updates for you. More information: **Opt in to early access updates**.

Open the focused view

Focused view is a centralized place to view all your open records that have activities that are due or pending, without having to navigate through multiple screens. By default, the lead entity grid opens in focused view. However, an administrator must enable and set other entities such as Account, Contact, Opportunity, and custom as default. More information: Enable focused view and set it as the default view.

From the sitemap, select an entity such as lead, account, contact, or opportunity. The entity grid opens in the focused view.

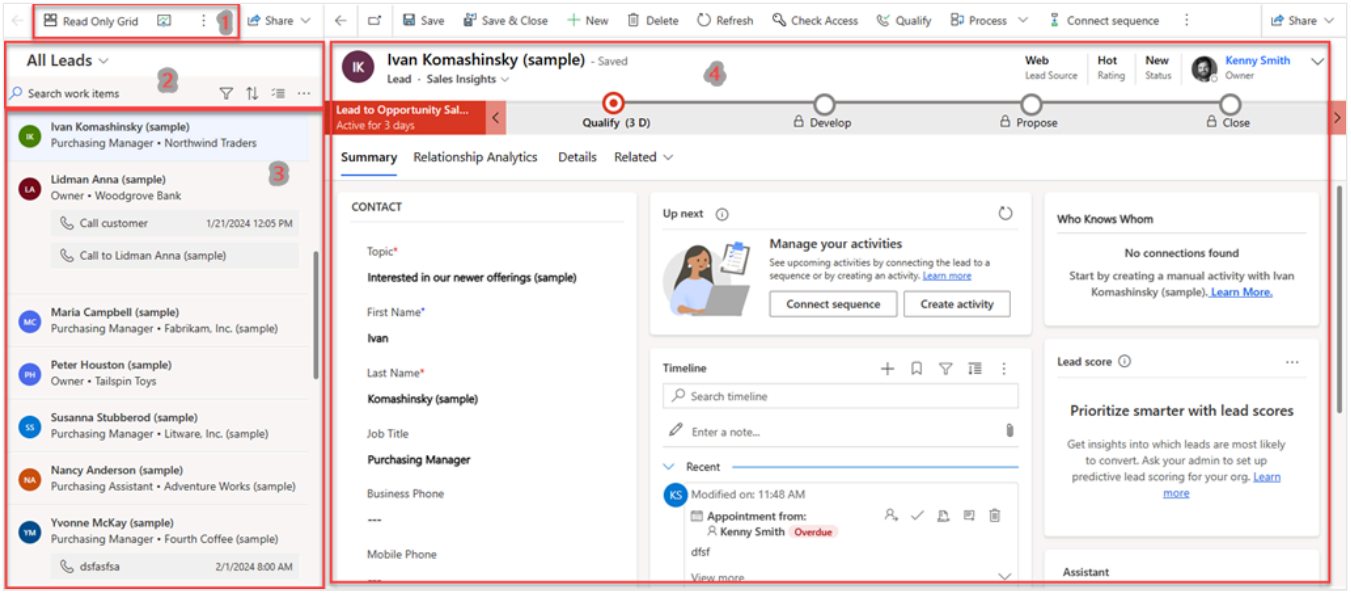
If you open an entity grid in the focused or list view and then close it within the same view, the application will remember to open the grid in the same view when you access it the next time. For example, if you open the Account entity grid in the focused view and then close it, the next time you open the Account entity grid, it will open in the focused view.

> **Note**

Focused view is now available for the **Activities** entity. However, you must opt in for early access features.

Understand the focused view

The following image is an example of focused view for the Lead entity:



Focused view is categorized into the following sections.

1. **Switch view and command bar**:

   - **Switching a View**: This action allows you to change from the focused view to the list view. To do this, select the **Read Only Grid** option. The **Read Only Grid** option is only available if you have one control for the entity. If you have multiple controls, the **Show As** option is displayed instead. Select **Show As** and then choose an option to change to the required view.
     If you have opted in for early access updates, the **Show As** option is displayed as first option on the command bar.
   - **Command bar**: This feature contains actions that help you perform bulk actions on multiple records. To use it, select **More options**. You can then choose the records that you want to work with and perform the required actions.

2. **Work list actions**:

   - **Search, filter, and sort**: These options are displayed at the top of the work list item view. You can search, filter, and sort the records that you want to view in the list to quickly identify only the records that you want to work with. You can also switch between focused view and list view. The attributes available in the view are displayed as filters and sort. These attributes include fields added through

workspace customizations in the sales accelerator, and any new fields added to the work list card through card customization.

> ⓘ **Note**
> - The following attributes types are not supported in filter and sort—Uniqueidentifier, related entities, and advanced filter that are created through workspace customizations.
> - Filters aren't supported on mobile devices.

For lead and opportunity entities, filters might display deprecated attribute options. To remove these filters, go to work list card customization and remove the **Predictive scoring** field.

If you want to add more filters to the view, you can create an advanced filter.

- **Advanced filter**: Create an advanced filter option using the expression builder to refine the content displayed in the work list. You can have only one advanced filter defined for an entity in focused view. Select **Edit filters** to open the expression builder. In the expression builder, create the conditions as required and then select **Apply**. When the advanced filter is applied, the filter icon is filled with a blue color, and the advanced filter tag is displayed under the filter options.

- **Select multiple records**: To perform bulk actions on multiple records, select the **Select multiple records** option. You can choose the records that you want to work with and then perform the required actions. On the command bar, select **More options** to perform the bulk actions.

- **Refresh and Settings**: Select **More options** to perform list refresh and configure the work list card through settings.
  - **Refresh**: Select **Refresh** to refresh the list of records.
  - **Customize work list card**: Select **Settings** to customize the work list card. More information: Customize the work list card

3. **Records list**: All records for the selected entity, including records with no associated activities, are displayed in a card format. You can perform and complete the activities, whether manual or automated, on these cards from the current date and from previous dates. Each card displays information such as the name of the record, the primary contact name, and the next best action.

   - **Next best actions**: These include the top two activities with the latest due date. To complete the next best action, hover over the action and then select the action icon.

- **No action items**: When no action items are available for the record, the record displays information such as the name and contact details.
- **Predictive scoring**: When predictive scoring is enabled in your organization, the list also displays the score for that lead or opportunity at the upper-right corner of the card. Learn more about prioritizing leads through scores and prioritizing opportunities through scores.

4. **Record details**: Select a record from the work list to view its details on the same page. When you select an entity record in the detail view page, the selected entity record is displayed inline on the same page, keeping the work list visible. For example, selecting a contact record within an opportunity record keeps the work list visible while displaying the contact details.
   However, when you perform certain actions such as creating a new record, composing an email, or opening an organization chart, the view opens in full page and hides the work list. Select the **Back** button to return to the actual work list record with the work list visible.
   When records are updated in the view, work list gets auto-refreshed if the entity on the work list matches the entity form of the record. After the refresh, if the current form record exists in the work list, the same record will remain selected else, the first record in the work list is selected.

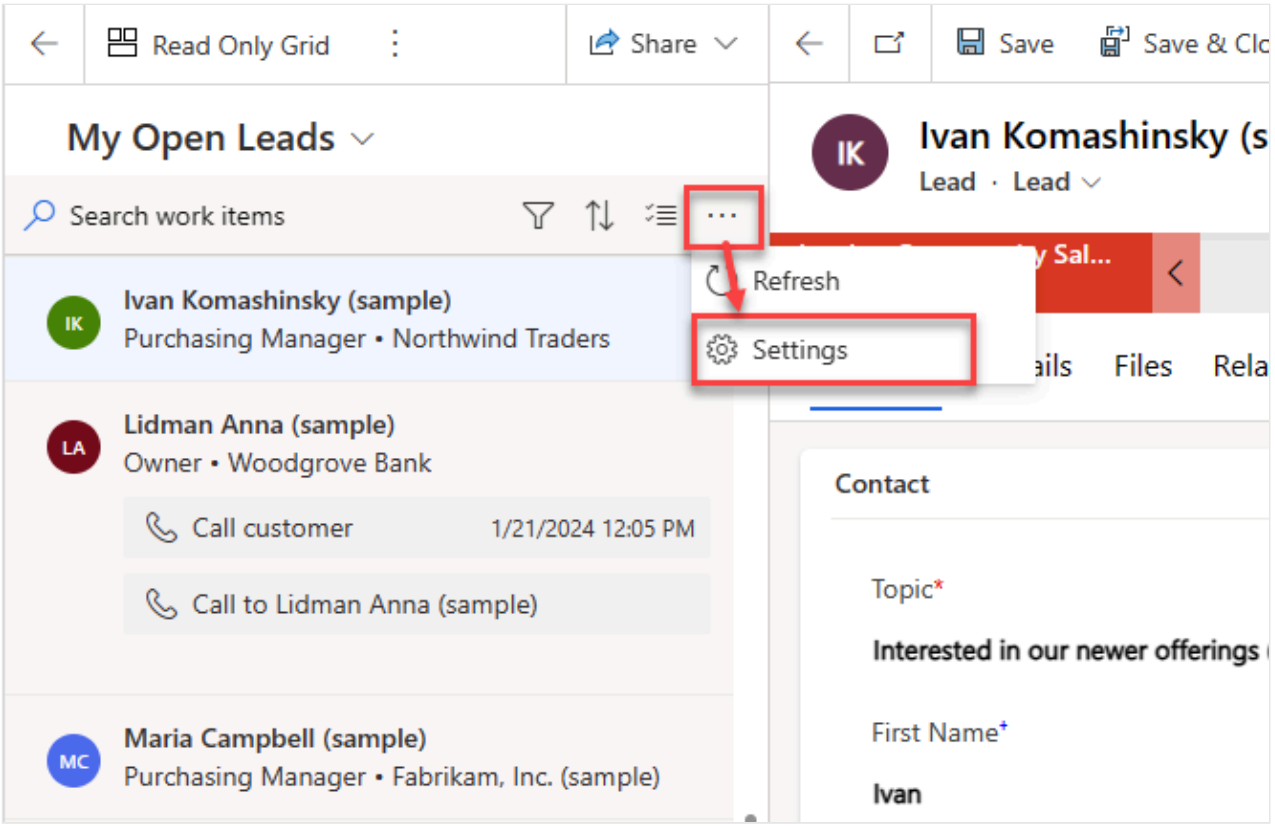
Customize the work list card

You can customize the work list card to display the information that you want to see. When sales accelerator is enabled, the customizations you make here are also reflected in the sales accelerator work list items. More information: Customize the appearance of work items in sales accelerator

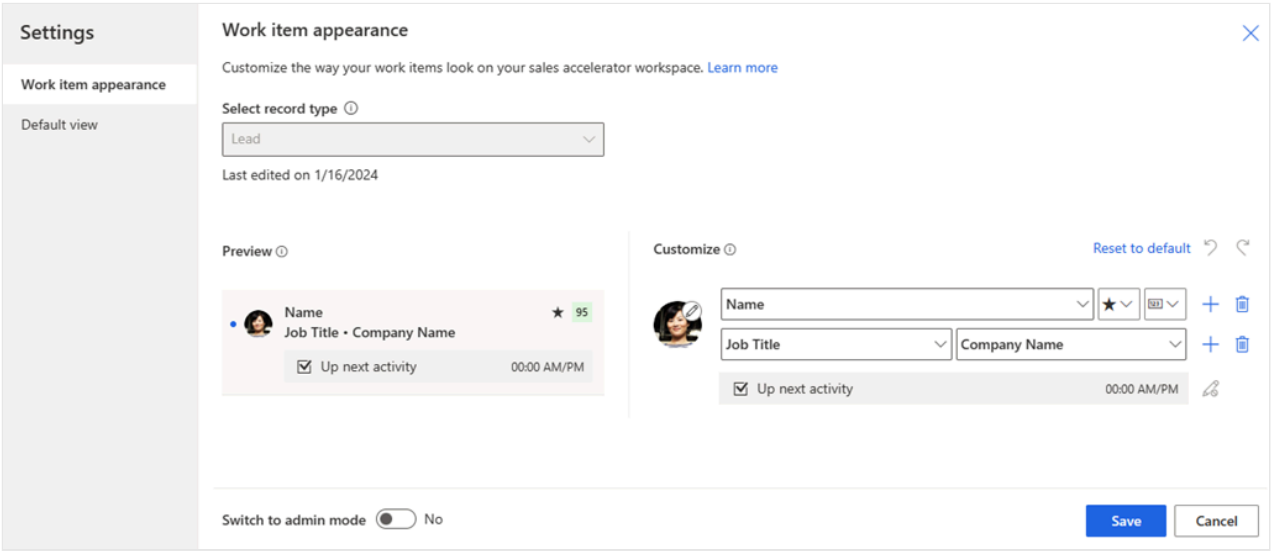
The customization that you make to the work list card is specific to the entity type. For example, if you customize the work list card for the Lead entity, the customization is applied only to the Lead entity. The customization isn't applied to the Account, Contact, Opportunity, or other entities. These customizations also affect the sales accelerator workspace view.

> **Note**
>
> You need to have read permissions for the **Work list user setting** entity to see the **Settings** icon and customize the work list card. Contact your administrator to provide the required permissions.

1. In the focused view page, on the work list pane, select **More options** and then select **Settings**.



2. In the **Settings** dialog box, select **Work item appearance**.



The following are the considerations to keep in mind when customizing the work list card in focused view:

- The **Switch to admin mode** and **Lock customization** options are available only for administrators.
- If the customization is disabled, contact your administrator to turn off the customization lock.
- When admin mode is turned on, you can customize work items for all users within your organization. When turned off, you can customize work items only for yourself. If admin mode is turned off, contact your administrator to turn it on. More information: Lock or unlock customization for other user roles

- You can't customize the action items in the work list card. Also, in preview mode, the **More options** icon isn't visible to select action items that are configured in the sales accelerator workspace view.

3. Under the **Customize** section, use the following options to customize:

   - To change the icon that appears on a work item, select the pencil icon, and then choose one of the following options:
     - **Record image**: A photo of the contact, if one is available.
     - **Record initials**: The contact's first and last initials.
     - **Record type**: The icon associated with the record type.
     - **Activity type**: The icon associated with the current activity on the record.

   - To change the content of work items, select the fields or attributes to display:
     - To change an existing attribute, select a different one in its place from the list.
     - To add an attribute in the same row, hover over an existing attribute and select **More options** > **Add a field**. You can show up to three fields in a row.
     - To add an icon in the same row, hover over an existing attribute and select **More options** > **Add an icon**. You can show up to three icons in a row.
     - To add a row, select the add icon (+). You can show up to four rows of information in a work item.
     - To delete a field or an icon, hover over it and select **More options** > **Remove**.

     > ⓘ Note
     >
     > Currently, the **Follow indicator** option doesn't work in focused view.

   - To remove all customizations and restore work items to their original appearance, select **Reset to default**.

   The **Preview** section displays a preview of the work list card based on the customizations that you make.

4. Select **Save**.

Auto-refresh records

Records in the work list can be refreshed automatically when they are updated. However, an administrator must enable this feature and configure the time interval for the records to refresh. For more information on how to configure this feature, see Automatically refresh records.

Increase the number of records displayed in focused view

You can configure the number of records that are displayed per page in the focused view. By default, 50 records are displayed, but you can increase this number up to a maximum of 250. To do so, go to your personalization settings, and under the **General** tab configure the **Set the number of rows shown per page in any list of rows** option to increase the number of records displayed per page in focused view. More information: Set personal options

Related information

Enable focused view and set it as default
Prioritize sales pipeline with work lists
Work with opportunities in the Kanban view
Customize work items
Focused view FAQs

Visualize data in a view with Power BI service

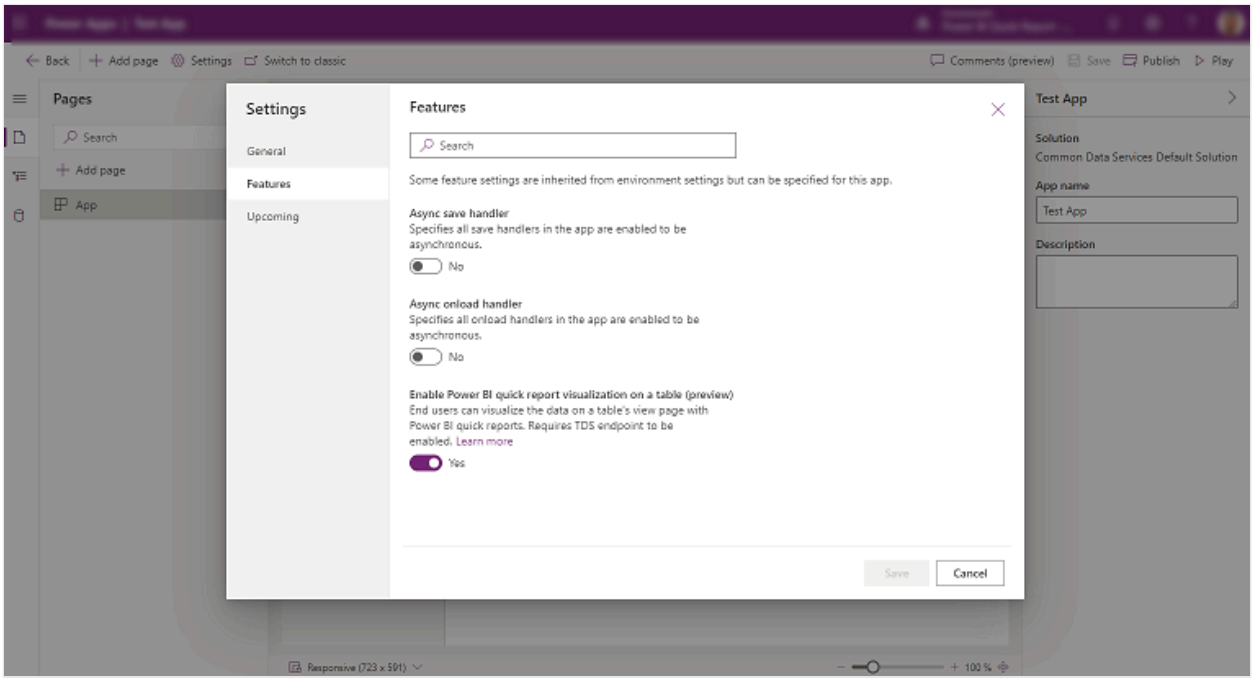
Article • 03/10/2025

Create reports in the Power BI service using the data that's in a view of your model-driven app. Power BI automatically generates the visuals for you, so you can start exploring the data with just a few clicks.

If you're new to Power BI reports and need more information, see Reports in Power BI.

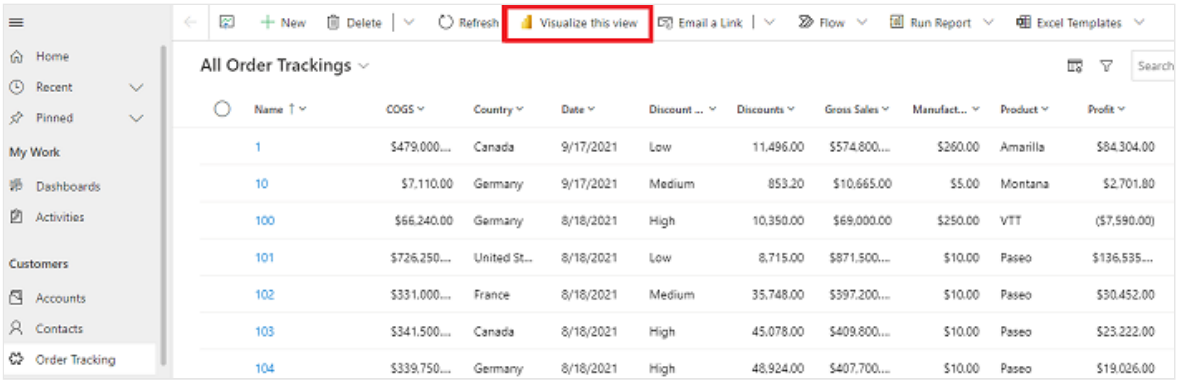
If you're using Power BI Desktop, you can also create a report using data from Dataverse in Power BI Desktop.

> **Note**
>
> Your administrator needs to turn on the **Enable Power BI quick report visualization on a table** feature setting, to use the capabilities that are covered in this topic. For more information, see Manage feature settings.

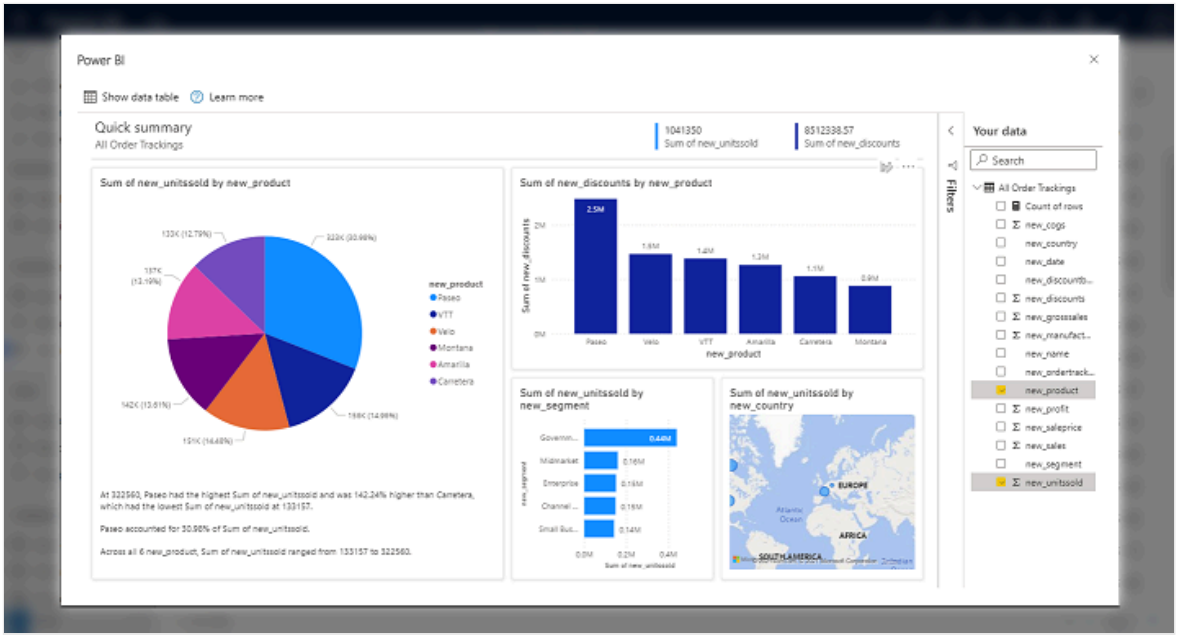


Visualize in Power BI

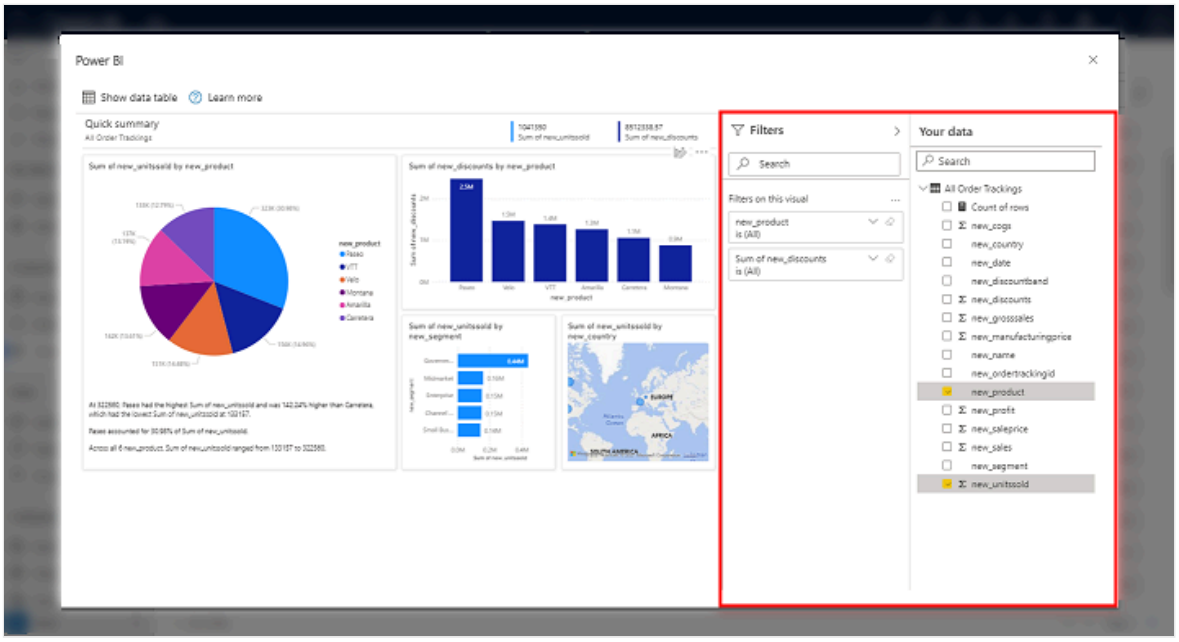
1. In your model-driven app, select a table and then on the command bar, select **Visualize this view**.



2. Power BI-generates visuals of the data in your view, based on the view definition. That is, the filters acting on the view are automatically applied to the Power BI visualization. A subset of columns that are part of the view are used to auto-generate the Power BI visuals.



The full set of view columns are available in the Power BI report to be used to change the data you see in the report.



For more information on how you can interact with this visualization, see Visualize your data quickly from Power Apps and Dynamics 365 apps (preview).

Licensing considerations

Anyone can use the **Visualize this view** feature to explore the data in a view, but to use the full edit experience, publish reports, access reports that others have published, or delete reports, you need a Power BI Pro license. If you don't currently have a Pro license, you can buy a Power BI Pro license or start a free trial.

Known issues and limitations

- The first time you visualize the data in a view, creating the dataset and report can take some time. We're working to improve this performance experience.
- This feature isn't supported for the **Activities** table due to a limitation in generating charts.
- This feature isn't supported for guest users in a tenant.
- If you're building your app for mobile, Power BI dashboards aren't a recommended choice to visualize data. Power BI dashboards are suited for larger screens, and the small screens and form factor of mobile devices can cause usability issues with scroll bars, filters, and more.

Feedback

Was this page helpful? Yes No

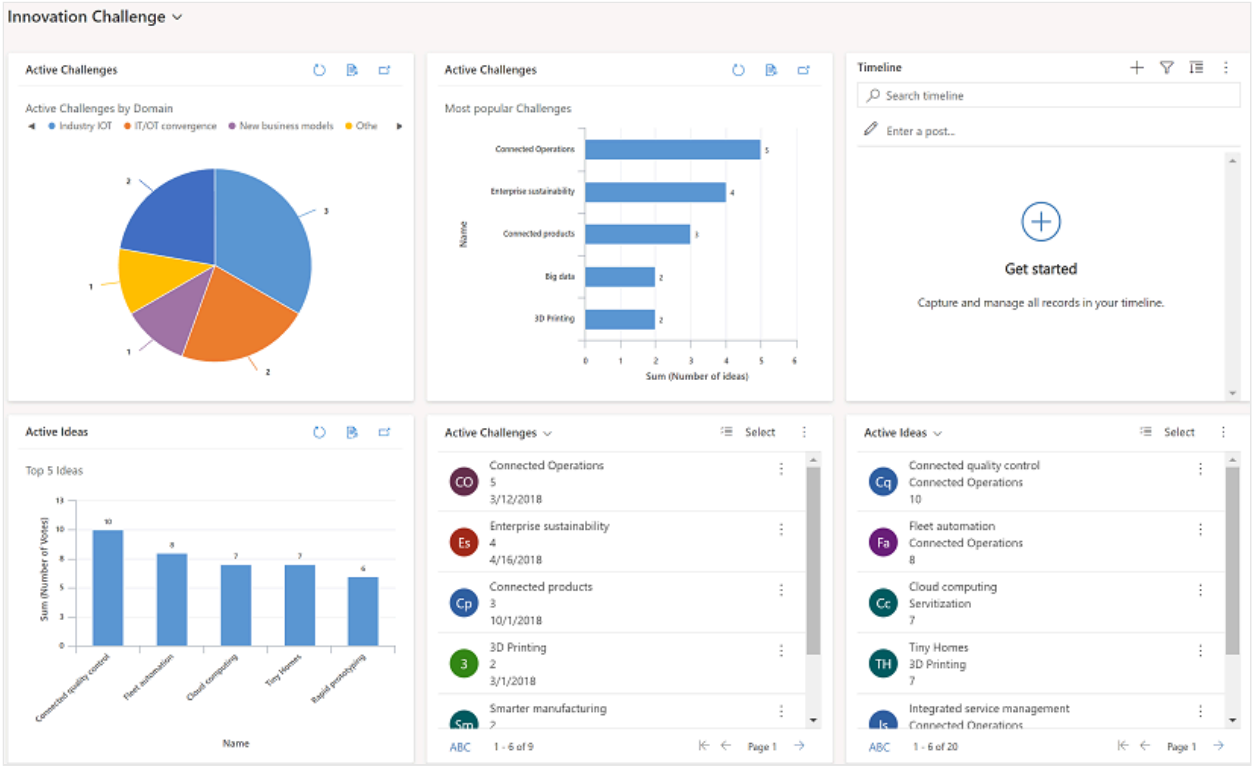
Provide product feedback

Track your progress with dashboards and charts

Article • 01/16/2025

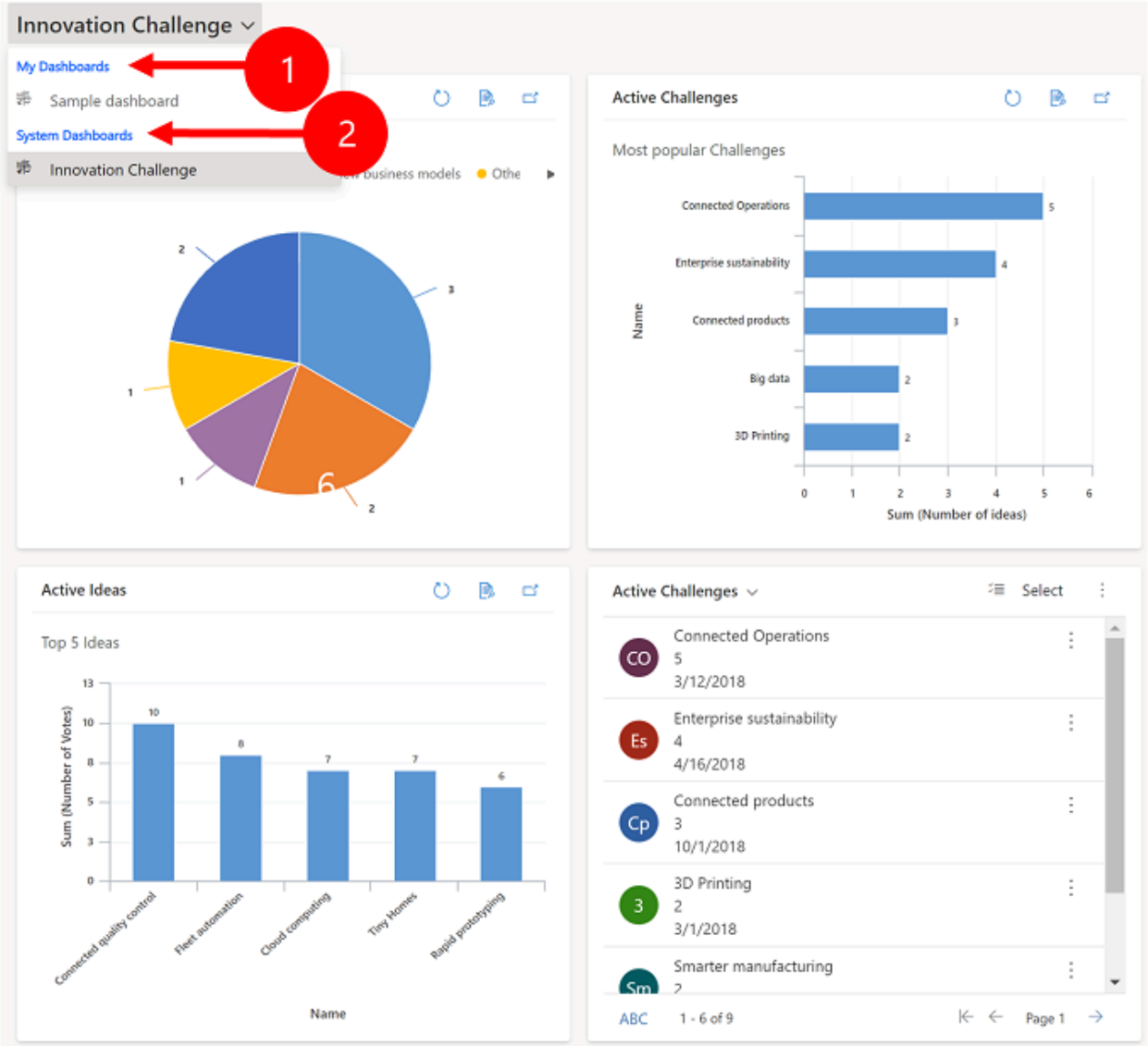
A dashboard is made up of a collection of easy-to-read charts and graphs that provide visual representation of filtered app data. The interactive charts and graphs on a dashboard display important data that gives you a 360-degree view of your business and helps you to understand key performance indicators (KPI).

Dashboards are available for all row types. Charts are available on the Dashboard page, Grid page, Associated Home page, and Edit Form page.



View dashboards

There are two types of dashboards, **System Dashboards** and **My Dashboards**.



Legend:

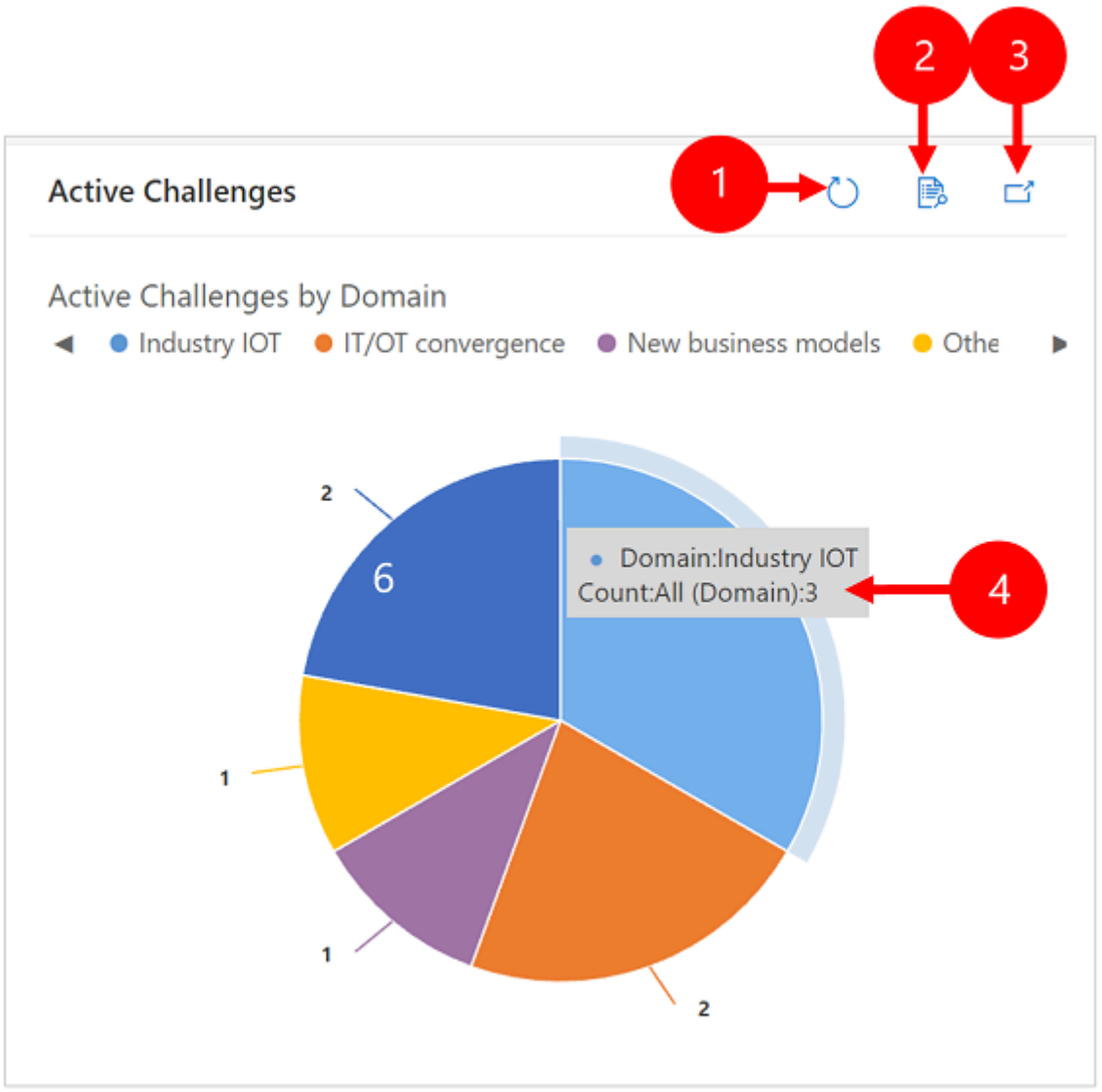
1. **My Dashboards**: Personal dashboards that can only be seen by the user who creates them. The user can also share their personal dashboard with other users if the user has sharing privileges for their security role.

2. **System Dashboards**: Created by your administrator. By default, these dashboards are visible to all users.

> ⓘ Note
>
> - When a personal dashboard is shared, the owner of the personal dashboard needs to explicitly share any contained, personal charts or personal views because they have separate permissions.
> - When a user sets a dashboard as default (via the **Set as default** action in the command bar), all page navigation to dashboards will take the user to the

default dashboard, which may be a different dashboard than the named dashboard the user selected.

Expand, refresh, and view rows

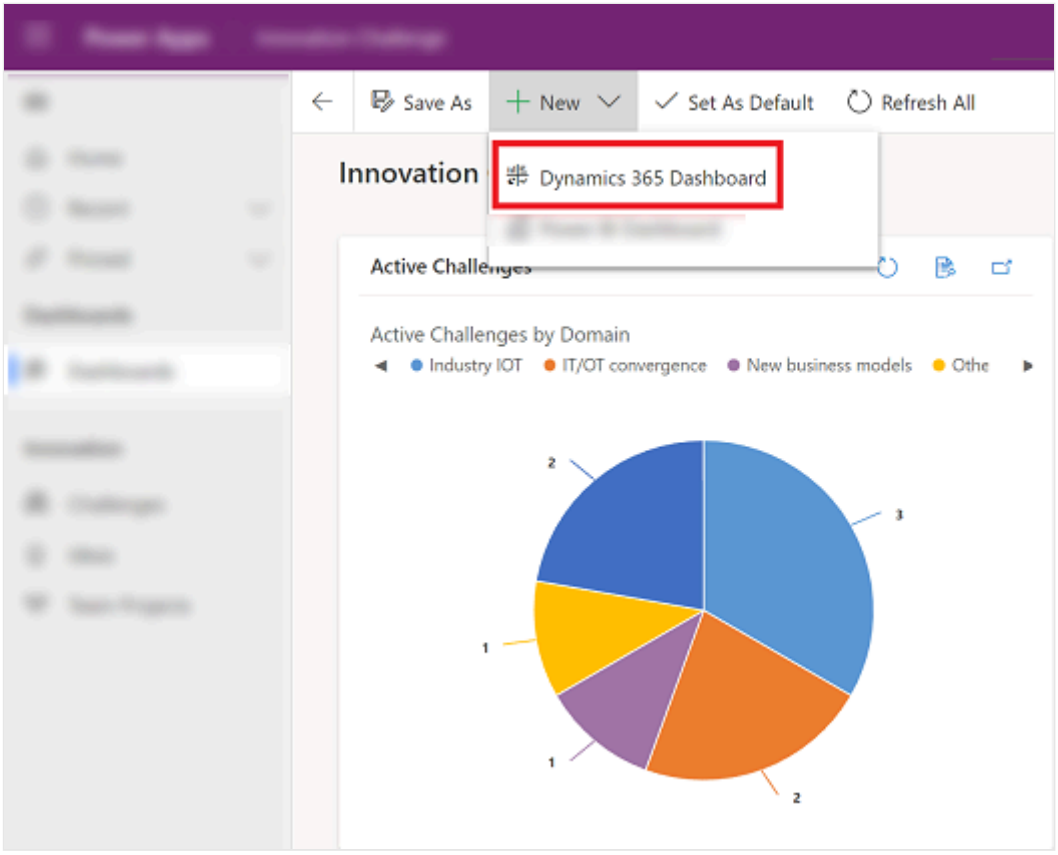


Legend:

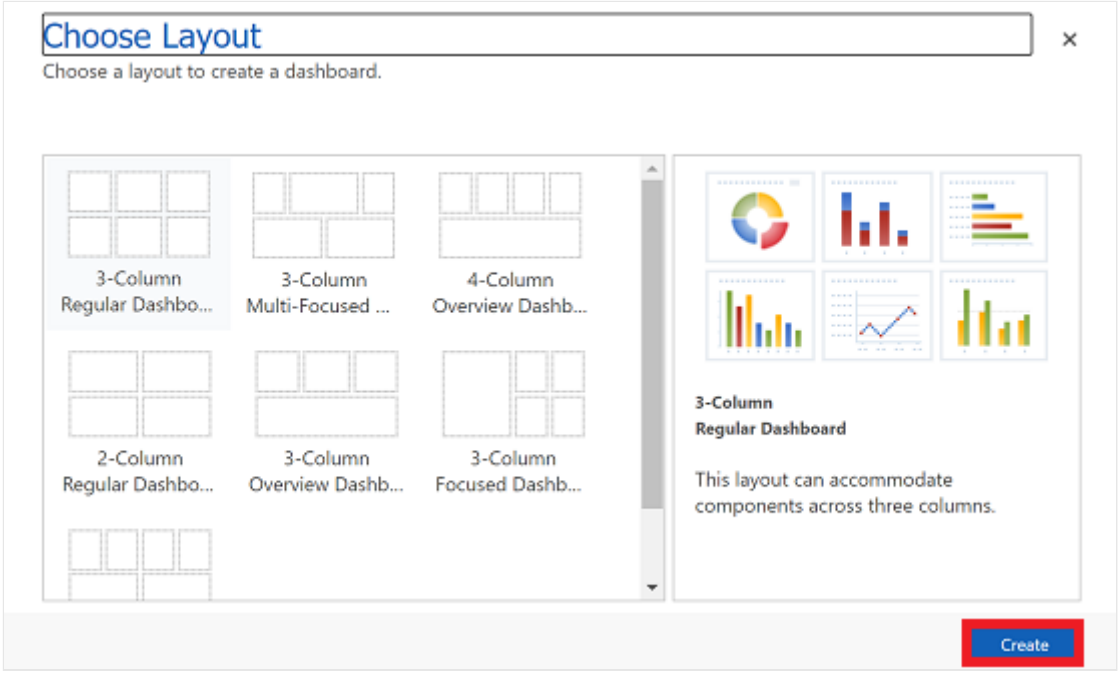
1. **Refresh**: Refreshes the rows in the chart.
2. **View Records**: See the chart and the underlying rows together. This opens the chart for the corresponding rows.
3. **Expand Chart**: Shows the chart in full screen mode. The chart is still interactive in expanded mode.
4. **Tool tips**: Hover your mouse over the chart to see a tooltip that provides quick information about that area of the chart.

Create a dashboard with charts

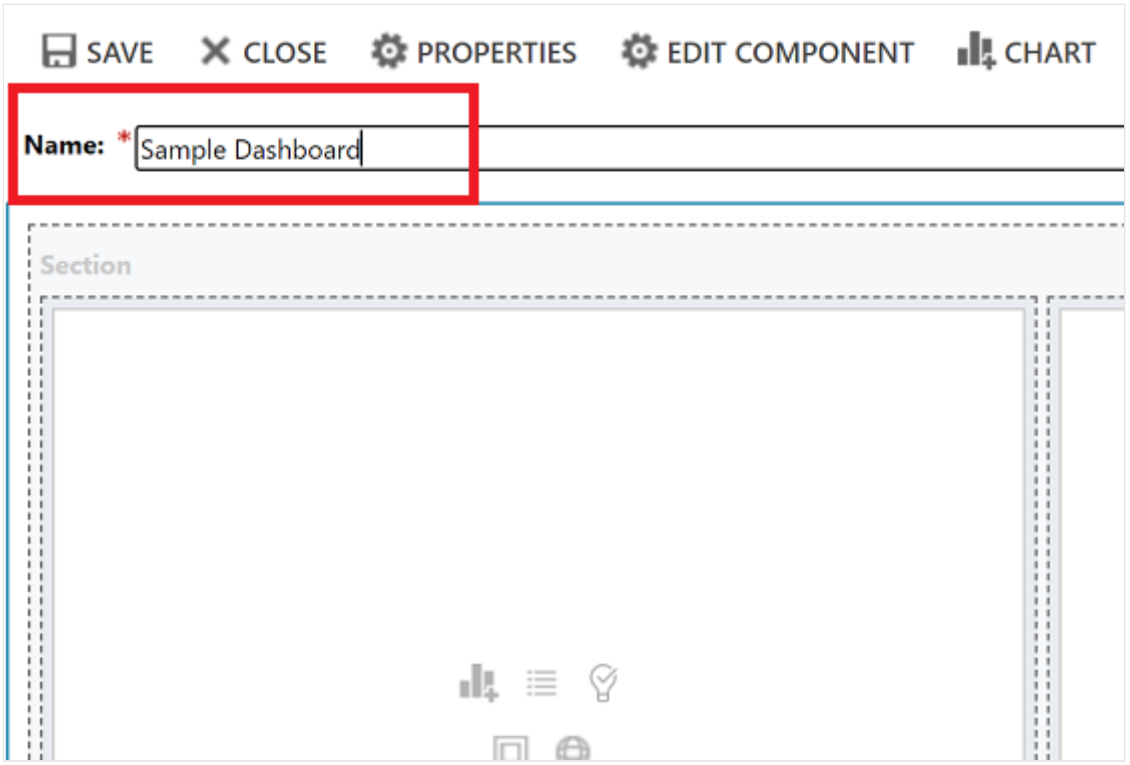
1. Go to **Dashboards** and then on the command bar select **New** > **Dynamics 365 Dashboard**.



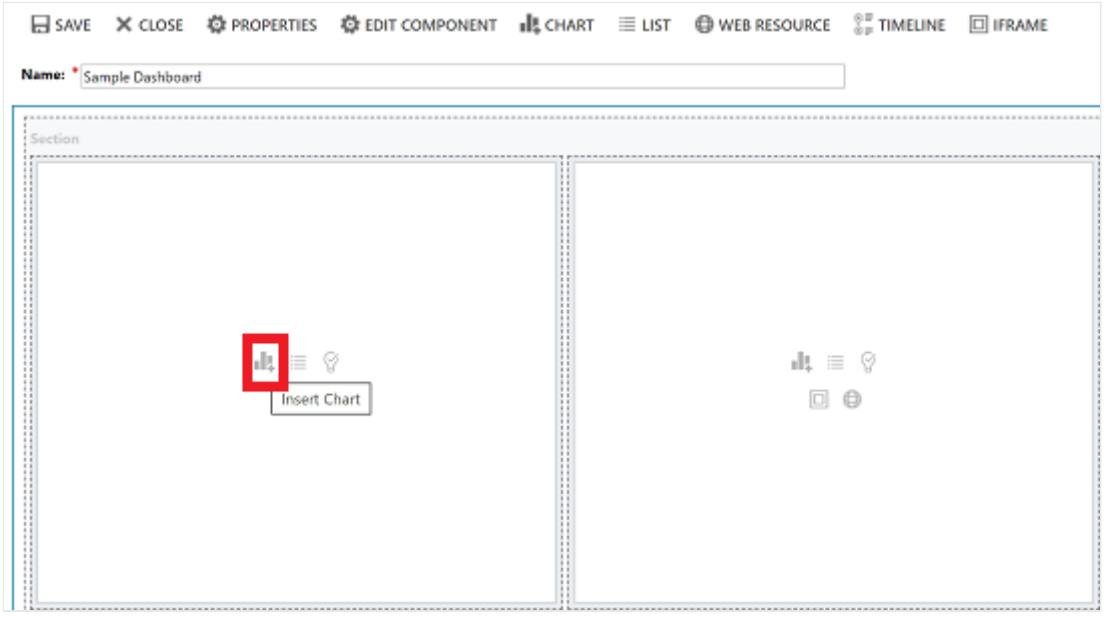
2. Choose a layout and then select **Create**. This is how your dashboard displays the charts, graphics, and numbers.



3. Type a name for the dashboard.



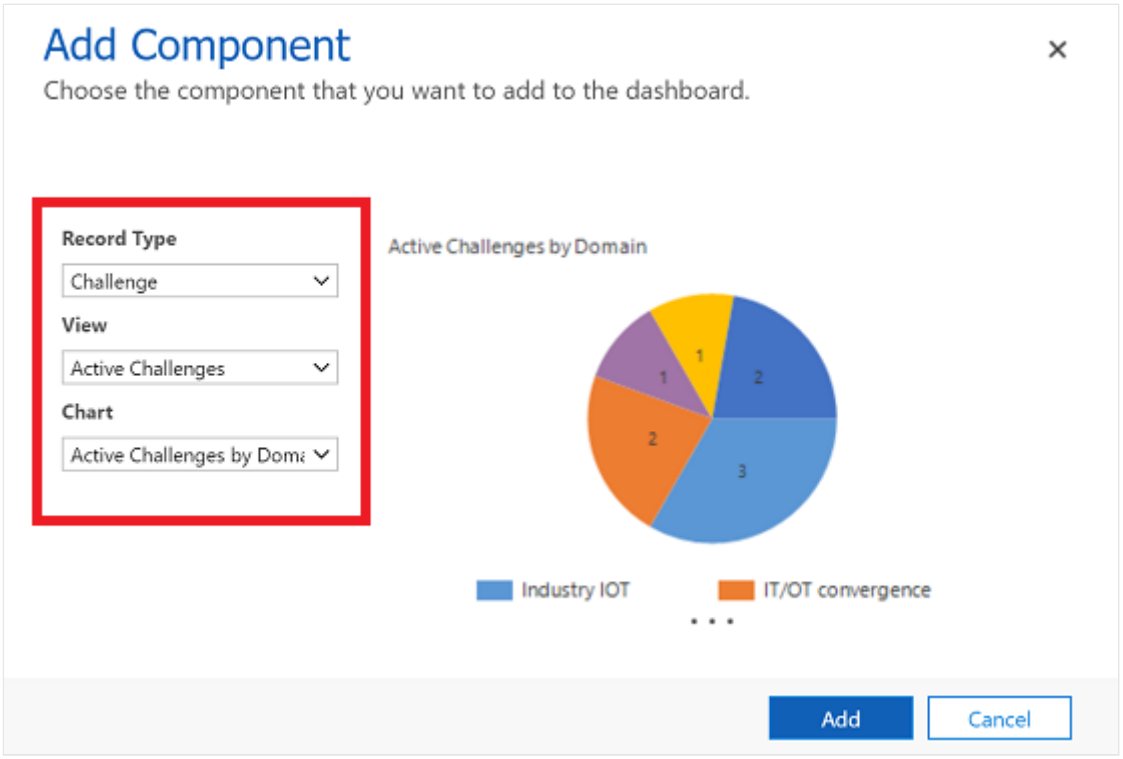
4. Add what you want to each area of your dashboard. For example, let's add a chart.



5. On the **Add Component** dialog box, select your options.

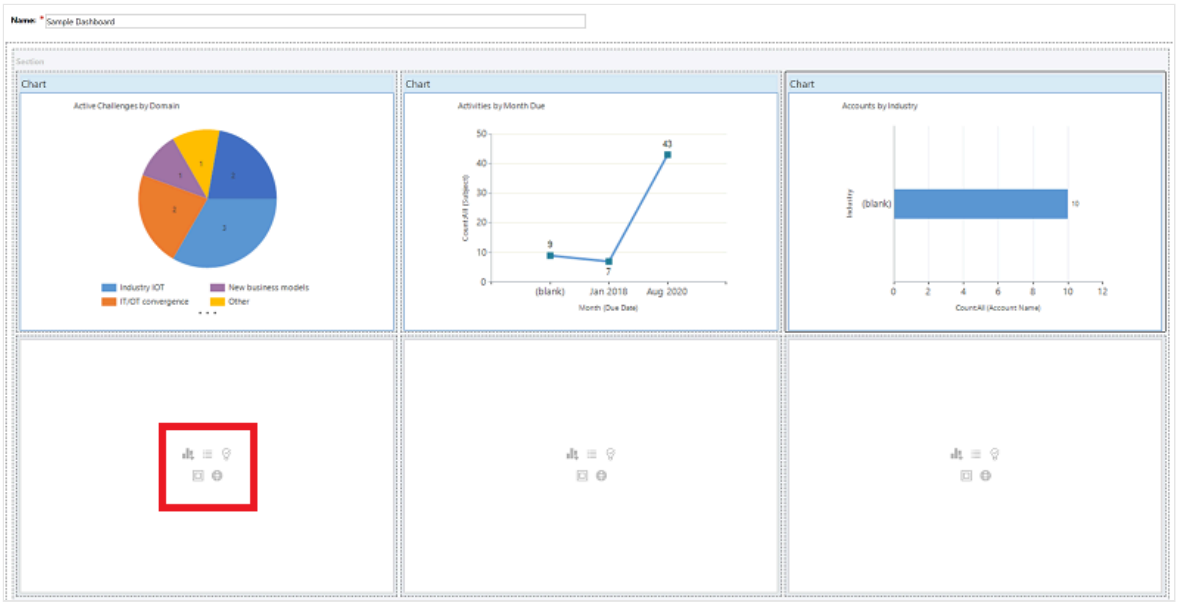
Expand table

	Options
Row Type	Select a row.
View	Select a view for the row.
Chart	Select the type of chart that you want use.



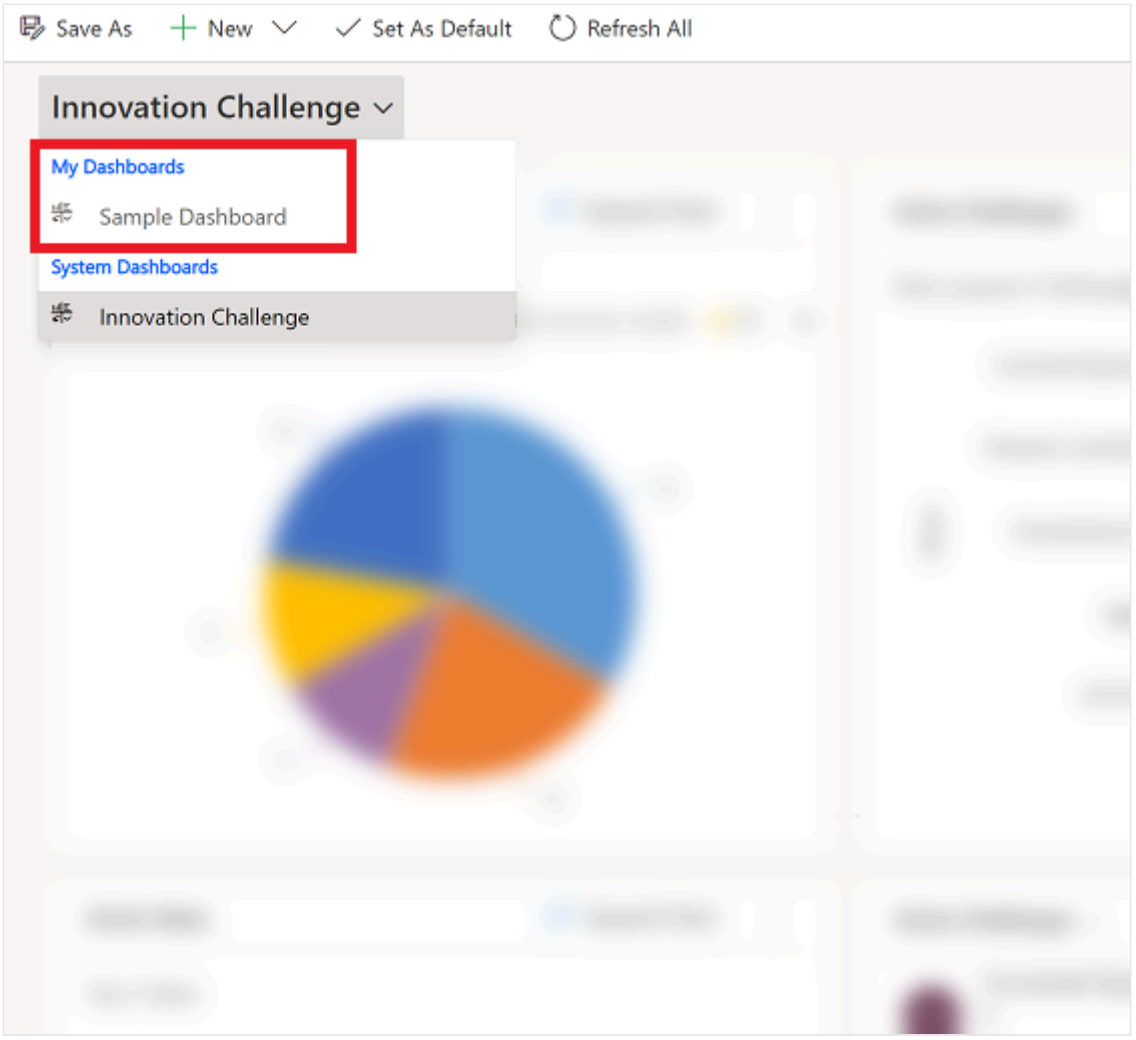
6. When you're done adding each component, select **Add**.

7. Continue adding components to the dashboard. You can add up to six components.



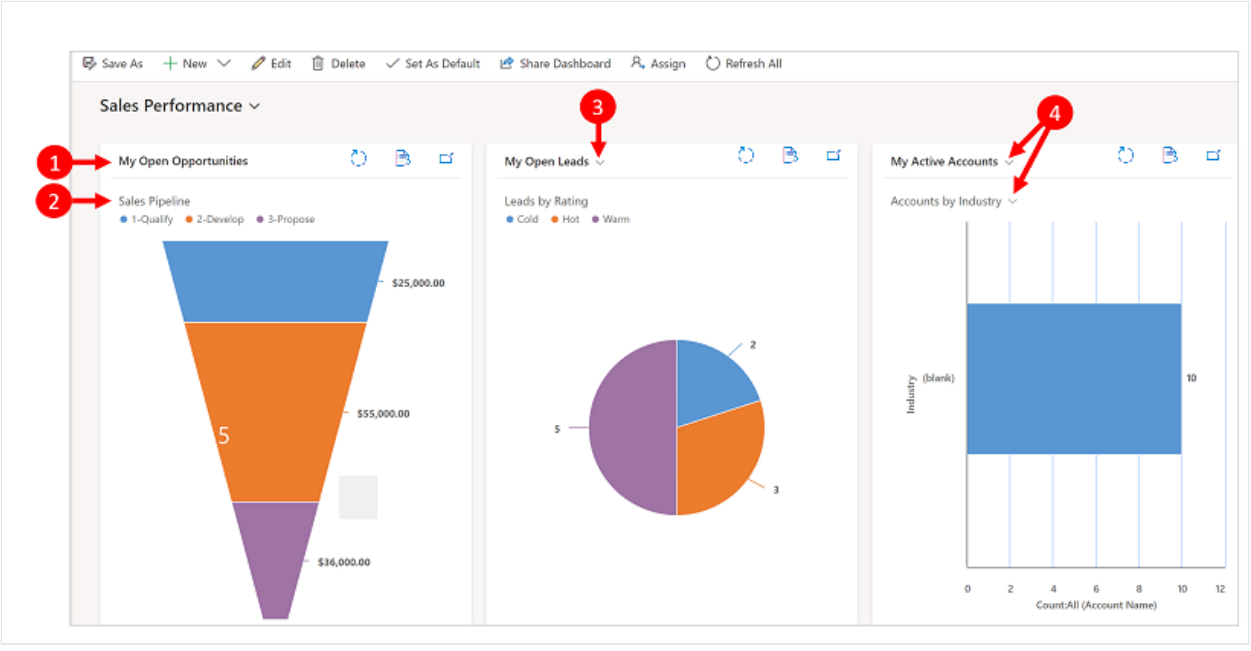
8. When you're done, on the command bar select **Save**.

9. The dashboard that you created will appear in the drop-down menu under **My Dashboards**.



Edit the properties of a chart

When you add a chart on a dashboard, you can edit the chart properties and modify the chart view and display options. The following screenshot shows how a chart will display based on the chart properties that you selected.

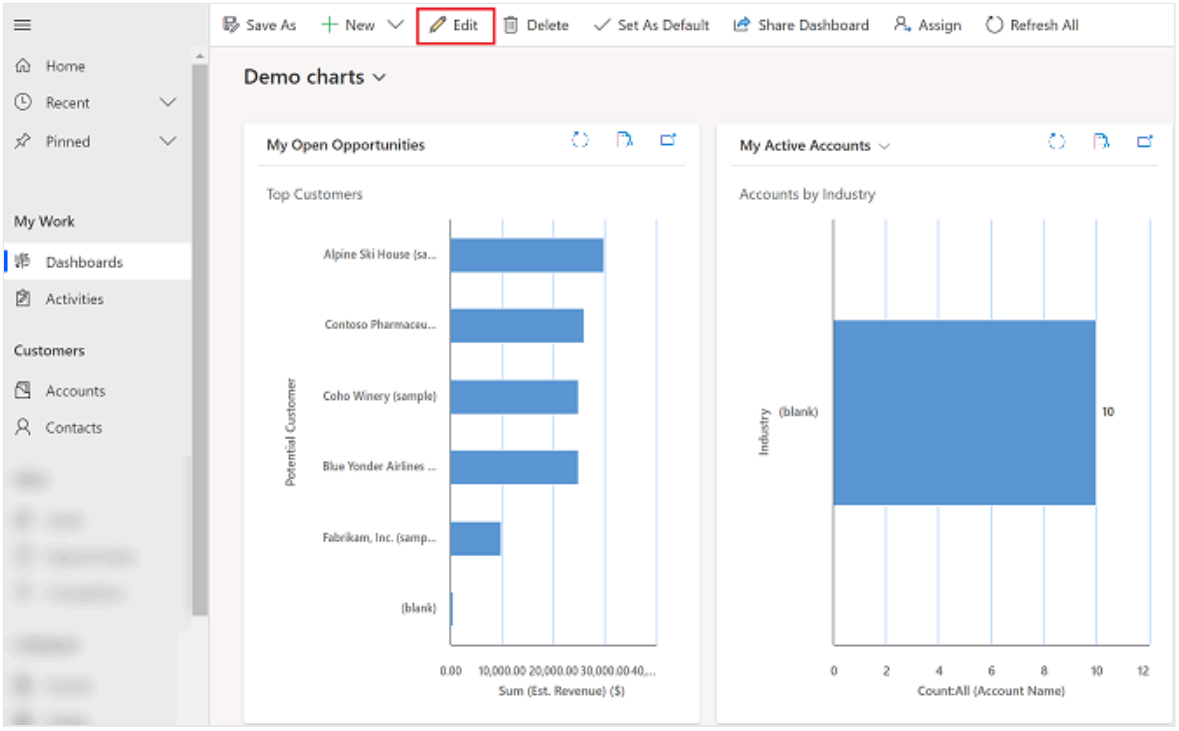


Legend:

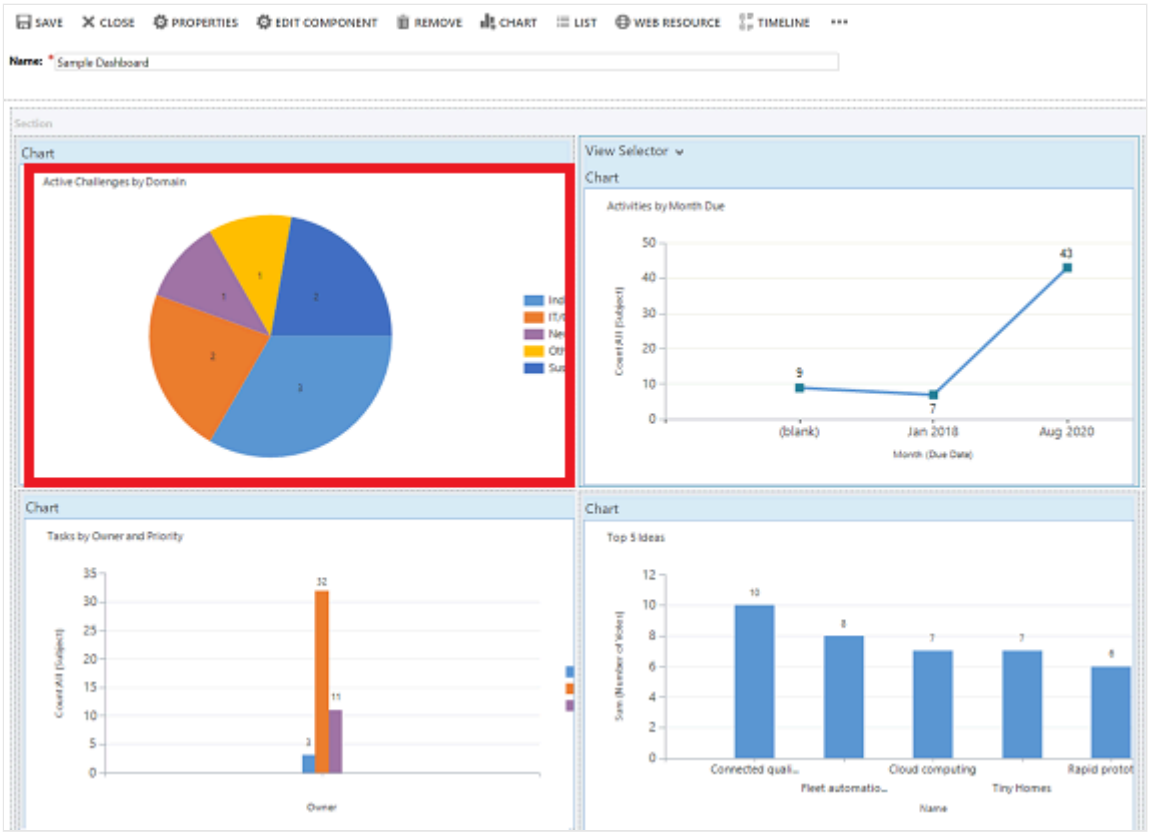
1. View title with **View Selector** turned off. You can't toggle between different views. You only see the title.
2. Chart title with **Display Chart Selection** turned off. You can't toggle between different charts for the table. You only see the title.
3. Only **View Selector** is turned on. You can toggle to a different view and the chart will render based on the underlying data for the selected view.
4. Both **View Selector** and **Display Chart Selection** is turned on. You can toggle to a different view and chart for the table.

To edit the chart properties, you need to edit the dashboard that the chart is in. If you're in the process of adding a new chart on a dashboard, then you're already in edit mode and can skip to step 2.

1. Go to the dashboard that the chart is in and on the command bar, select **Edit**.

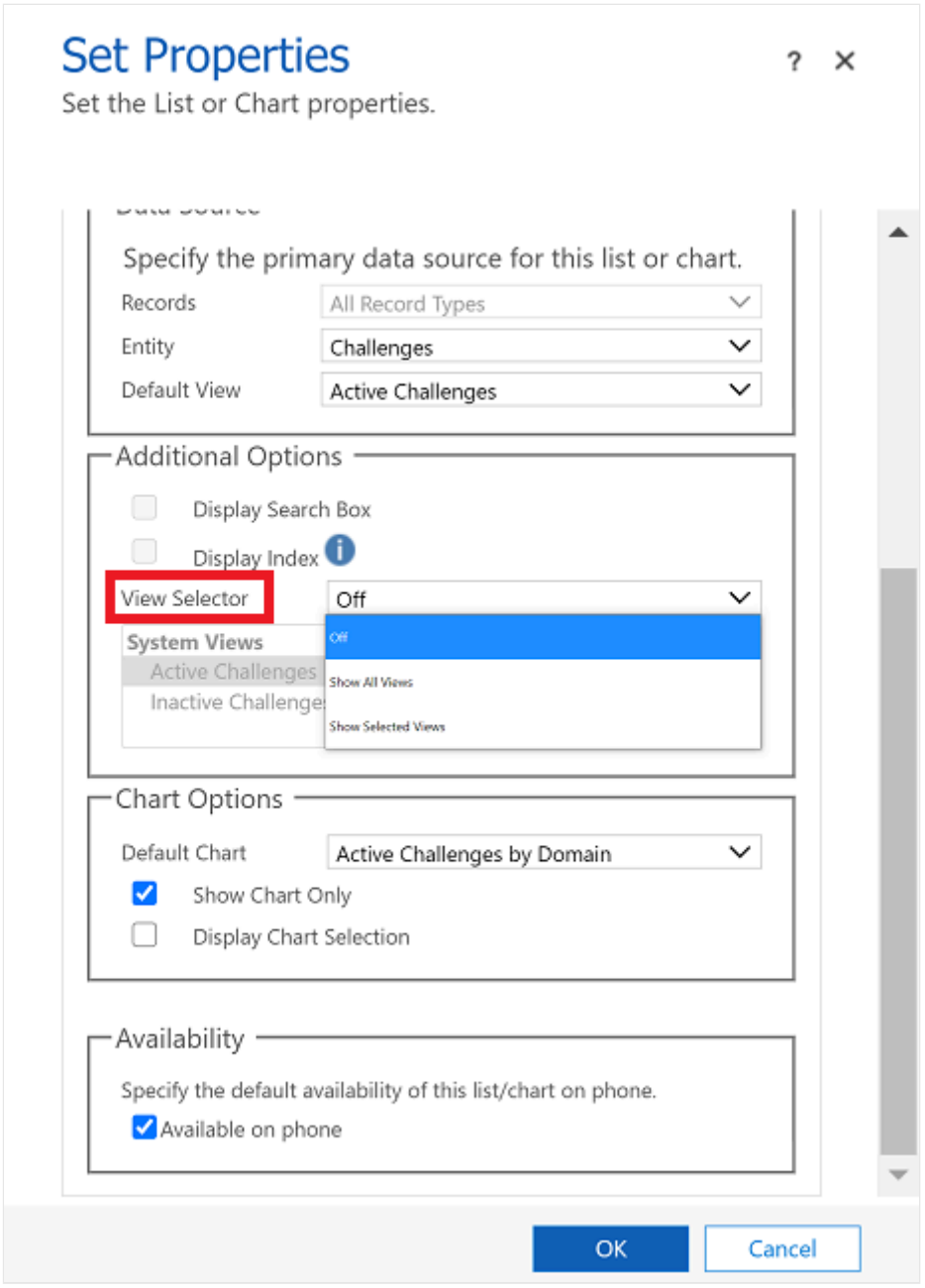


2. Double-select the chart that you want to edit.

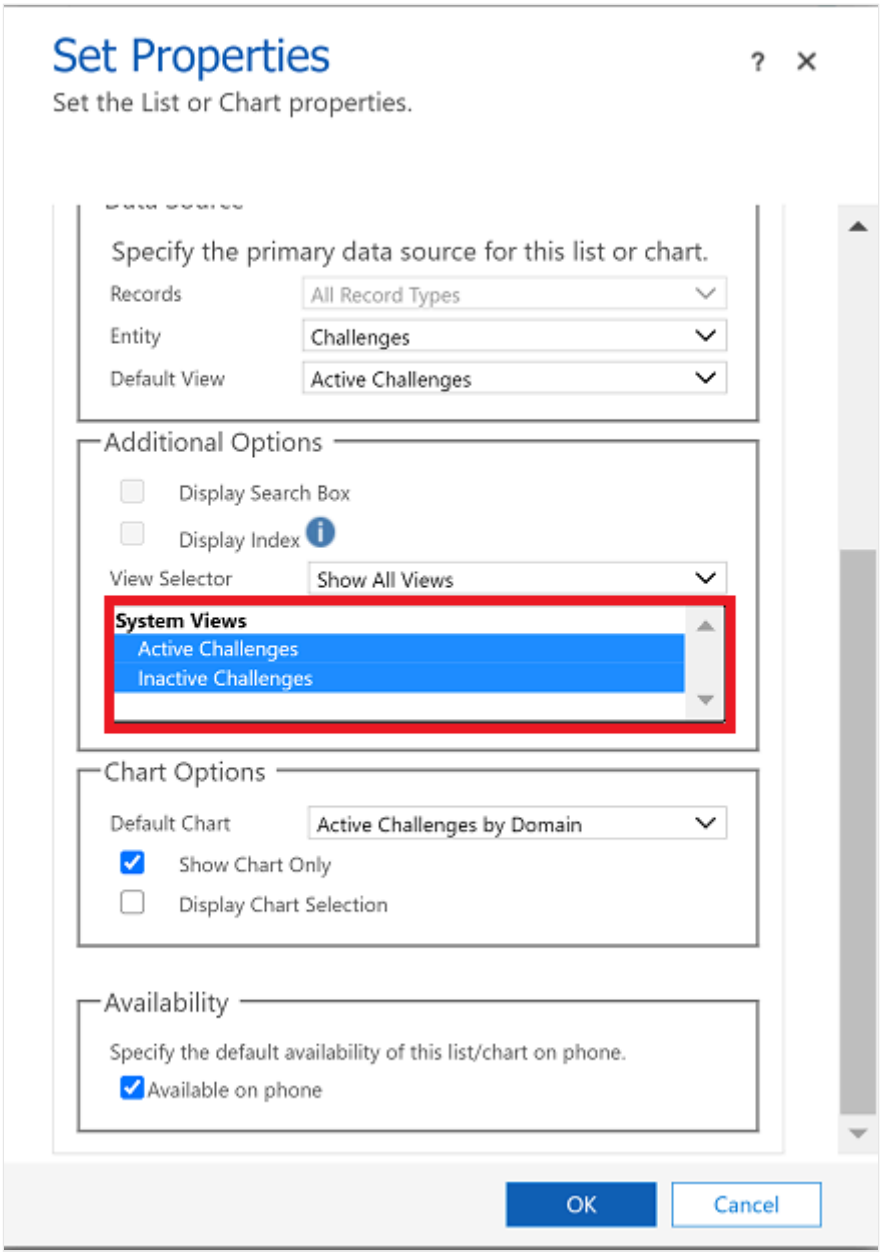


3. On the **Set Properties** dialog box, select a **View Selector** option.

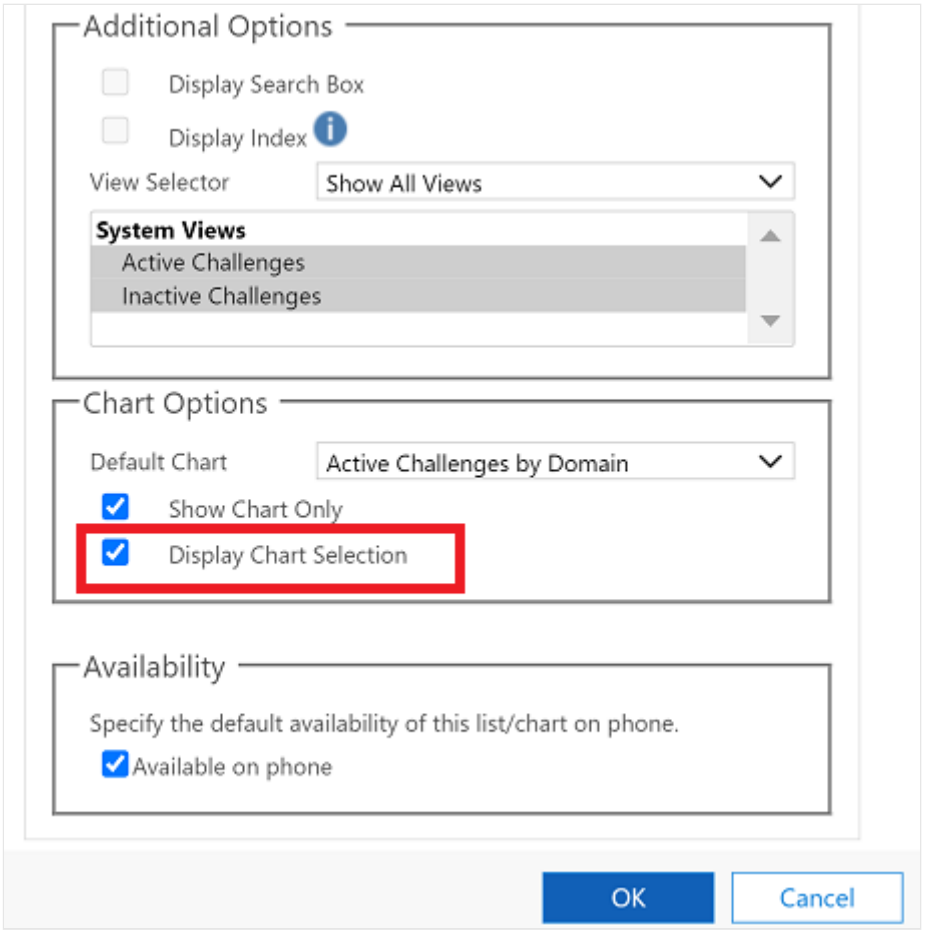
   - **Off**: Don't show view selector, show the view title only.
   - **Show All Views**: Show the view selector on chart and display all views in the view selector.
   - **Show Selected Views**: Show the view selector on chart and display only the selected views in the view selector.



4. Once **View Selector** is turned on, you can select different views to see how the chart renders with different data associated with the view.



5. Select the **Display Chart Selection** check box to show the chart selector. When chart selector is enabled, you can toggle to a different chart for the table and see different visualizations of the data. If the box is unchecked, then you only see the chart title.



6. When you're done, select **OK**.

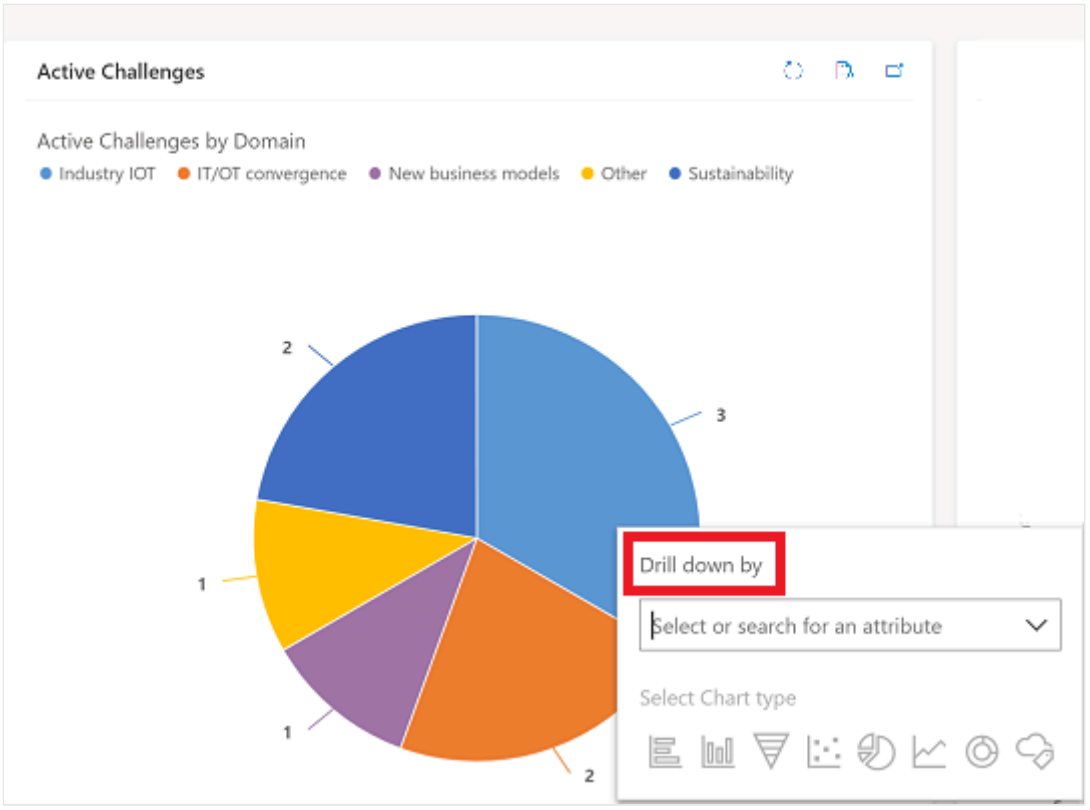
Drill-down in a chart

You can drill down into a segment on a chart and filter the data for that segment. For example, you can drill down on one segment of a chart and change the chart type as you drill down.

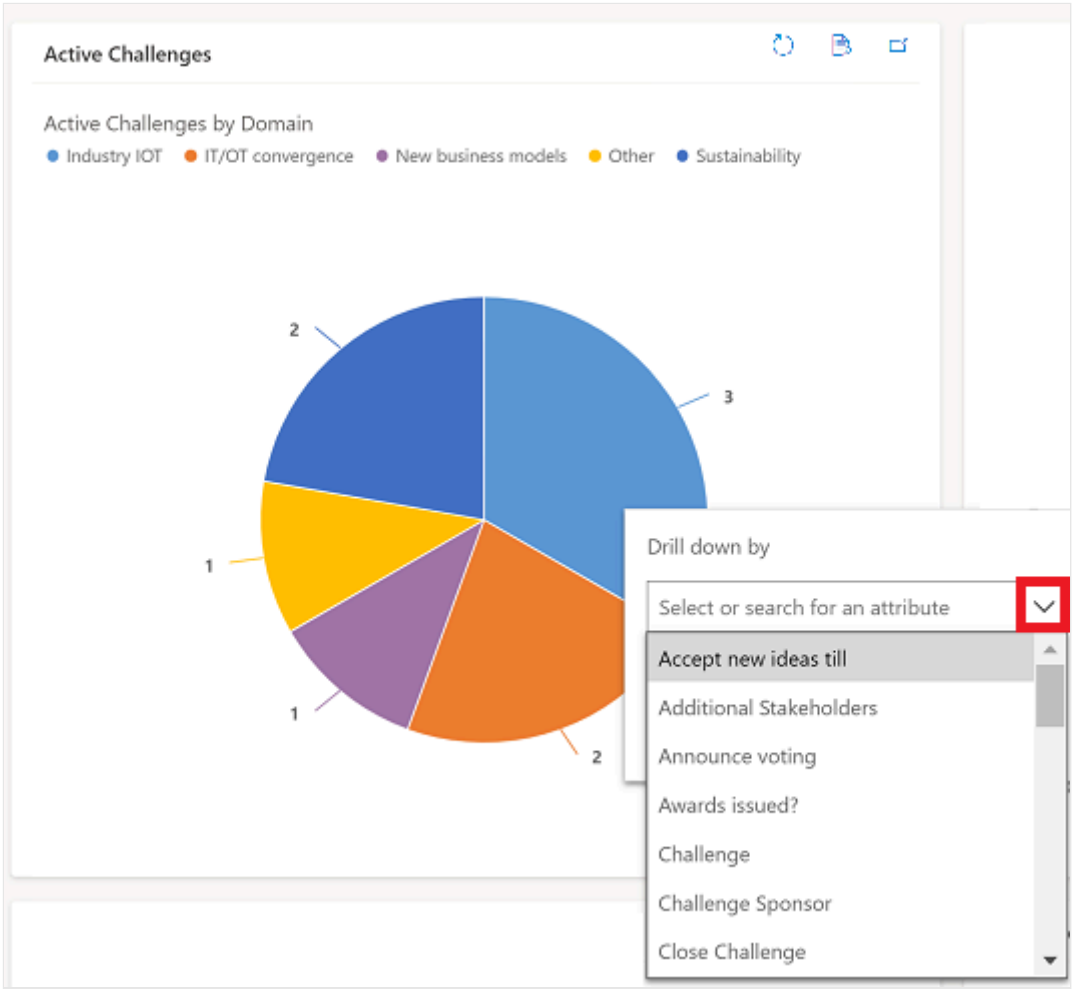
The chart drill-down feature is only available on the browser. It isn't available on the mobile app for phones or tablets.

In the following example, let's further drill-down into the chart.

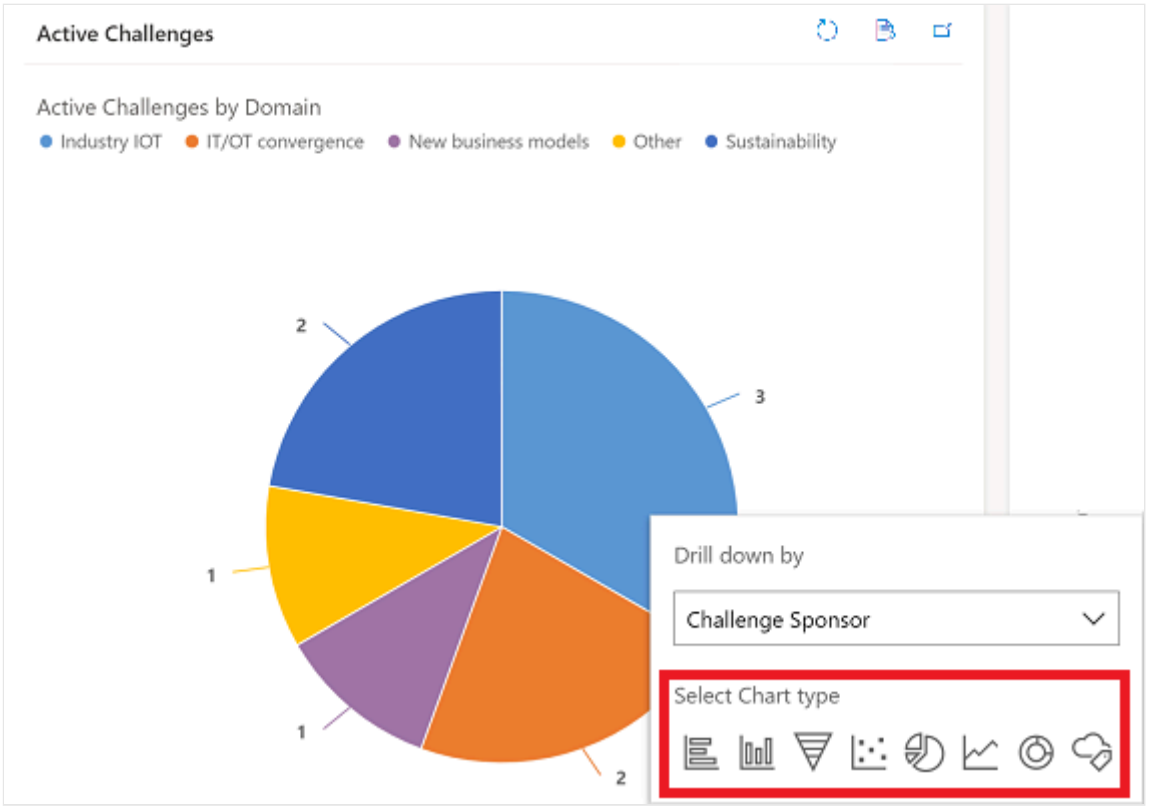
1. Select a data point on the chart to open the **Drill down by** dialog box.



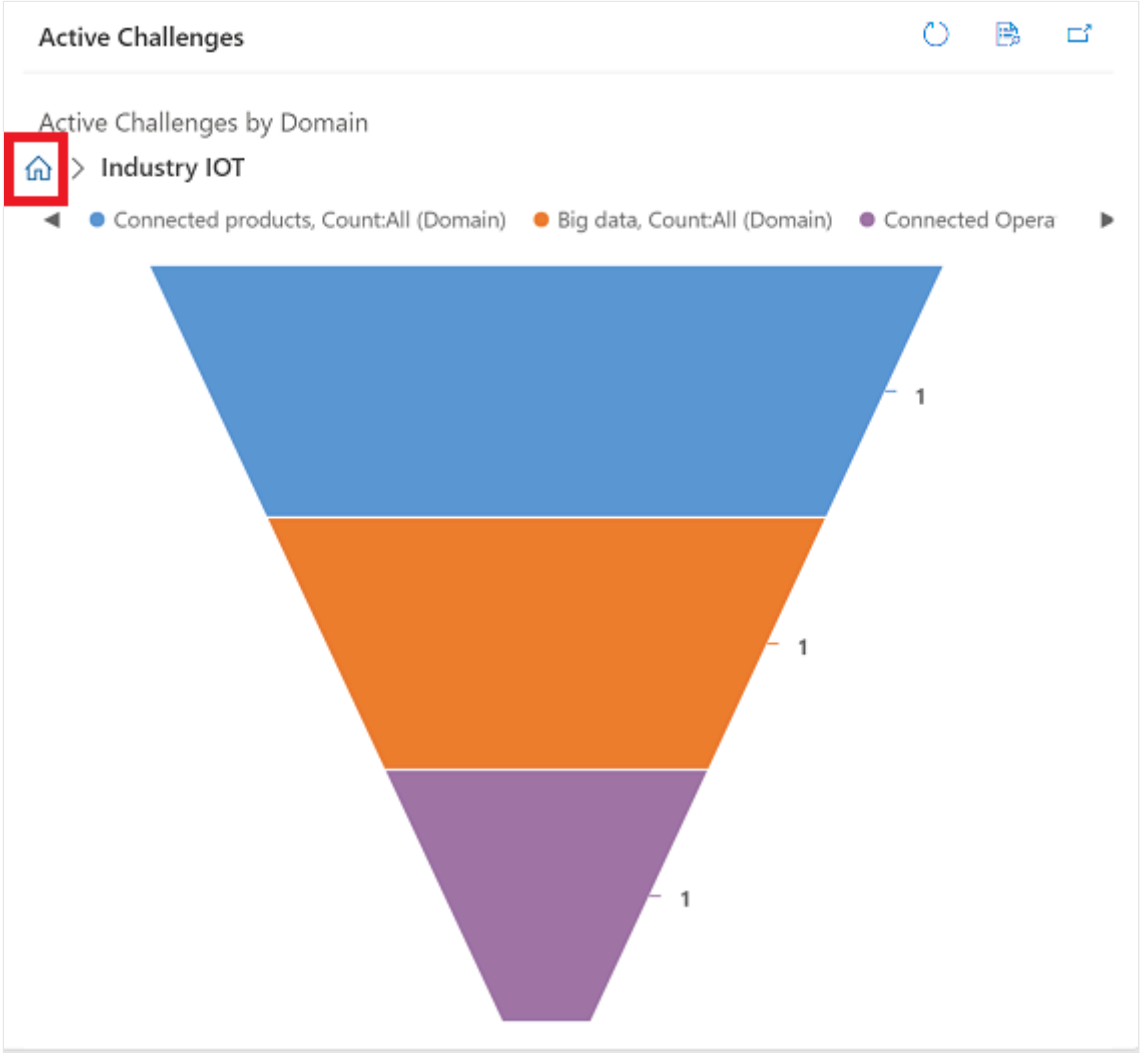
2. Select the drop-down menu and choose how you want to group the data.



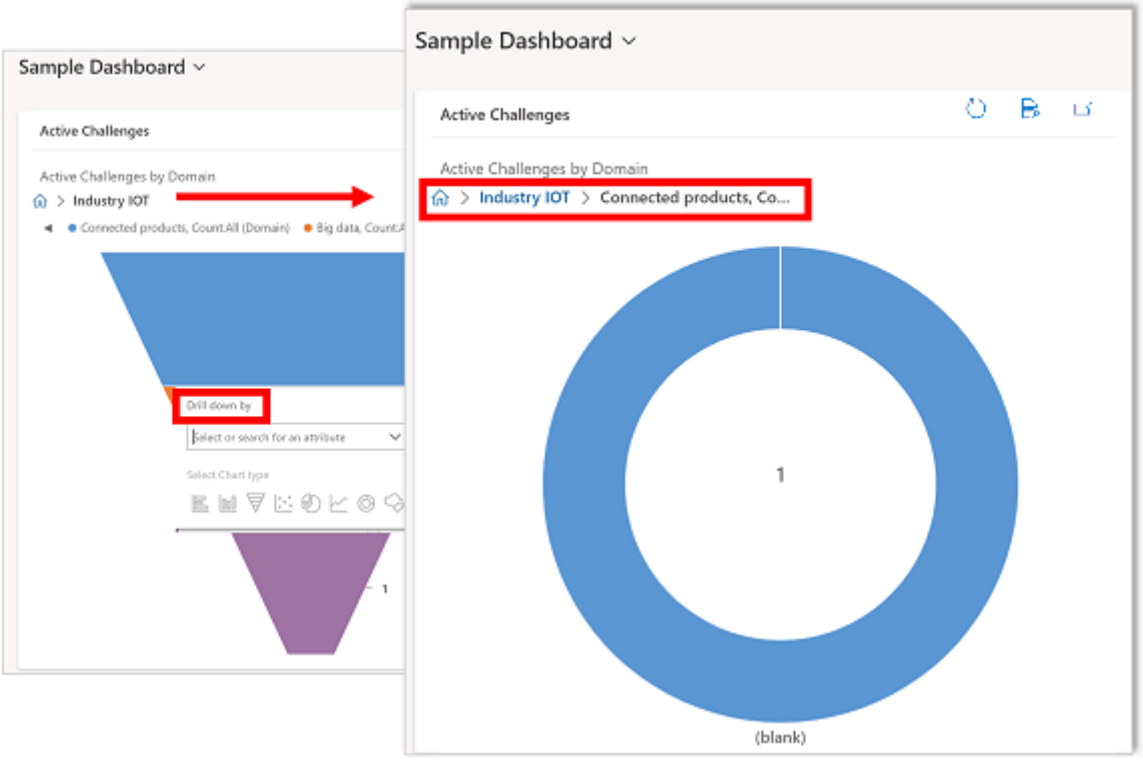
3. Select the type of chart that you want to see the drill-down data in.



4. The chart with the drilled-down data will show. The breadcrumbs at the top show the filter criteria of the drill-down data. You can select the breadcrumbs to go back home or to the previous screen.



You can continue drilling further into the chart data. As you drill down deeper, the breadcrumbs will expand and you can go back to the previous screen or select the Home button to go back to the Home screen.



Known issues

- In the chart designer, adding an order on certain calculated columns isn't supported and will cause an error. The calculated columns causing this are using other calculated columns, a related table column, or a local column on the table.
- Chart displays zero values when there's no data. With Unified Interface for model-driven apps you can still select a chart with zero value and drill down in to the data. More information: Chart displays zero values when there's no data
- By design, charts in rollup column can't be displayed because rollup columns contain an aggregate value computed over the rows related to a specified row.
- Chart drill-down functionality isn't available on charts in a subgrid or an associated grid.
- When you add a list control to a dashboard and switch the control to the **Power Apps Read-Only grid** control, you'll see a blank tile on the dashboard because the **Power Apps Read-Only grid** control isn't supported on dashboards. We recommend that you use **Power Apps grid** control.

Feedback

Was this page helpful? Yes No

Provide product feedback

View charts on a page or in a row

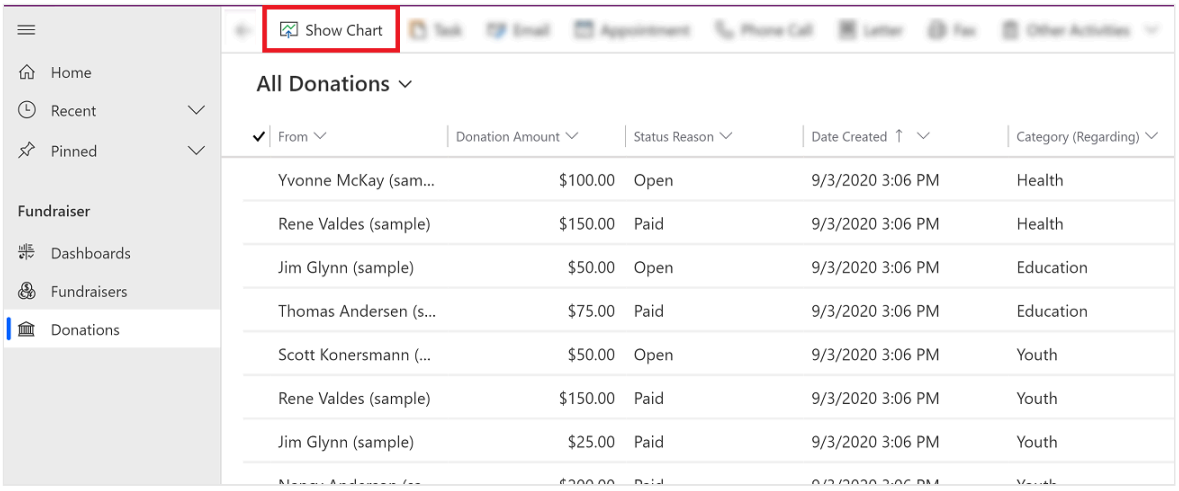
Article • 10/14/2022

You can view charts on a page that has a list of rows. You can also open a row and view a chart in the row.

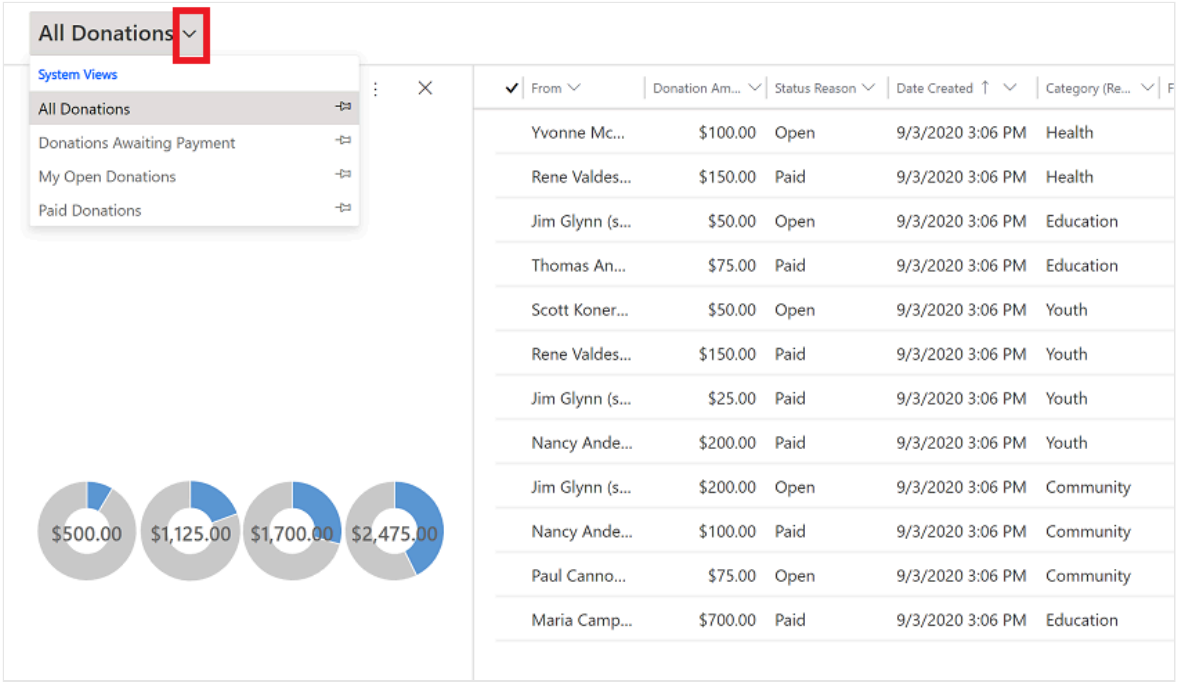
Only an administrator can add charts to a row. For information, go to Add a chart to a form.

View charts on a page

1. On the left navigation pane, select a page.
2. On the command bar, select **Show Chart**.

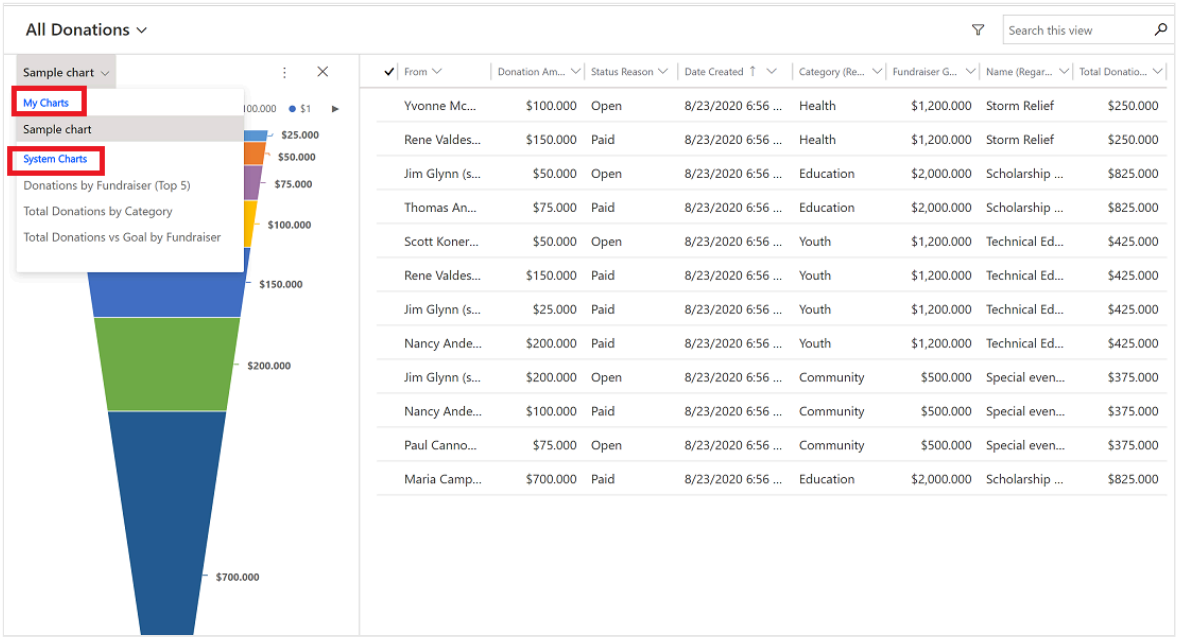


3. Select the drop-down menu to choose a view.



4. Select the chart view drop-down menu to choose between a system or personal view.

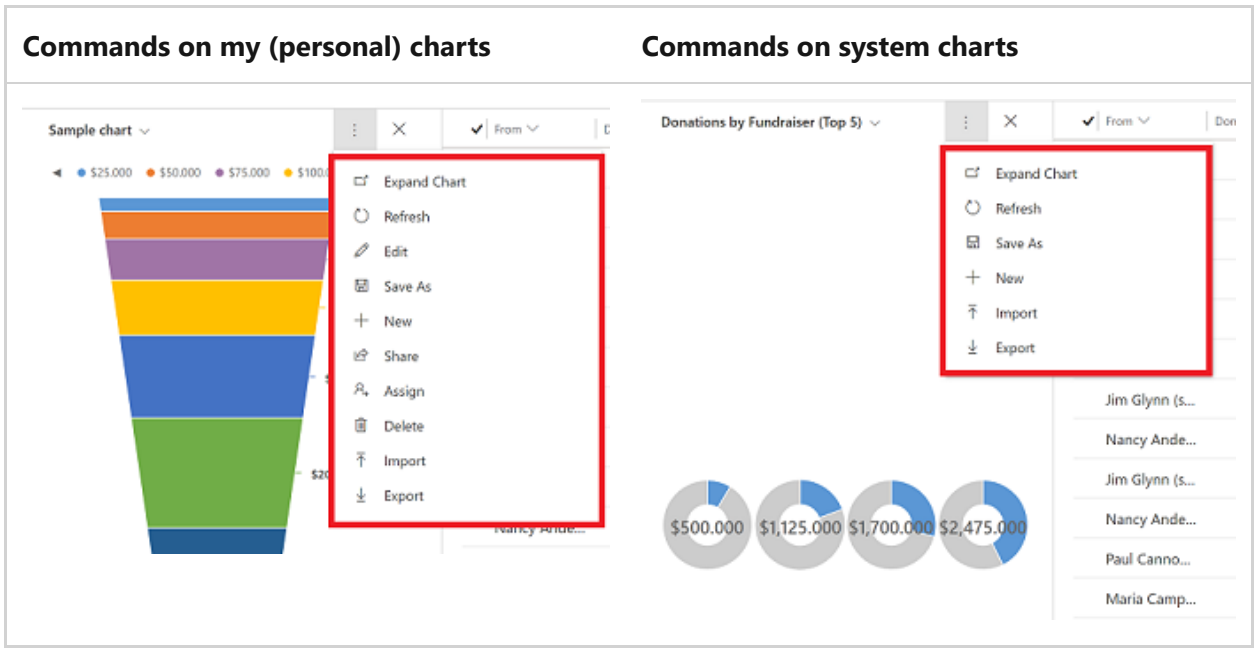
    - **My Charts**: Personal charts that can only be seen by the user who created the chart. The user can also share their personal chart with other users if the user has sharing privileges for their security role.
    - **System Charts**: Created by your administrator. By default, these charts are visible to all users.



Commands on a chart view

Use the **More Commands** button ⋮ to access commands such as create a new chart, import, export, and more. The chart commands options are different for system charts and personal charts.

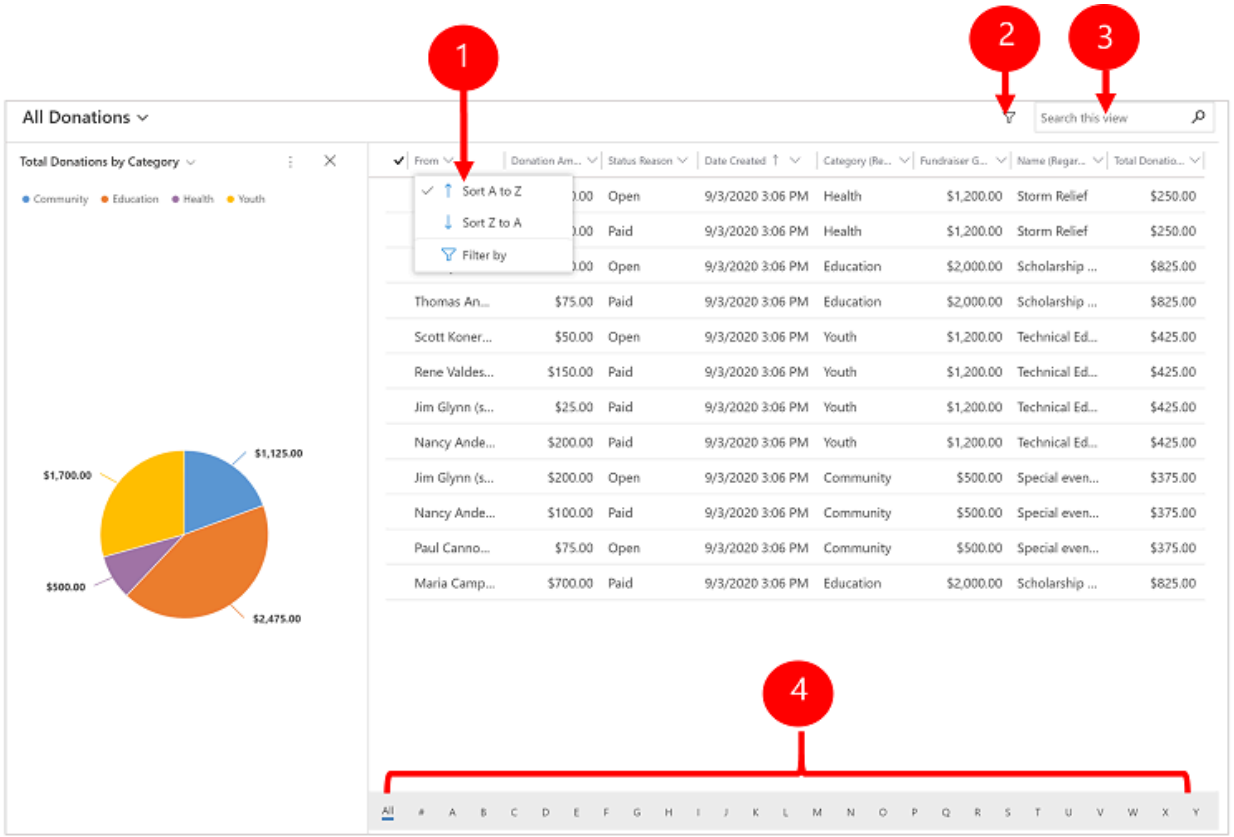
Expand table



Grid filter on charts

Grid home page provides multiple filter options for a user to filter the grid data such as quick find, advanced find, grid column filter, and jump bar. For more information, go to How to use grid filters.

When a chart is shown together with grid, the grid filter option that you select will filter both chart and grid.

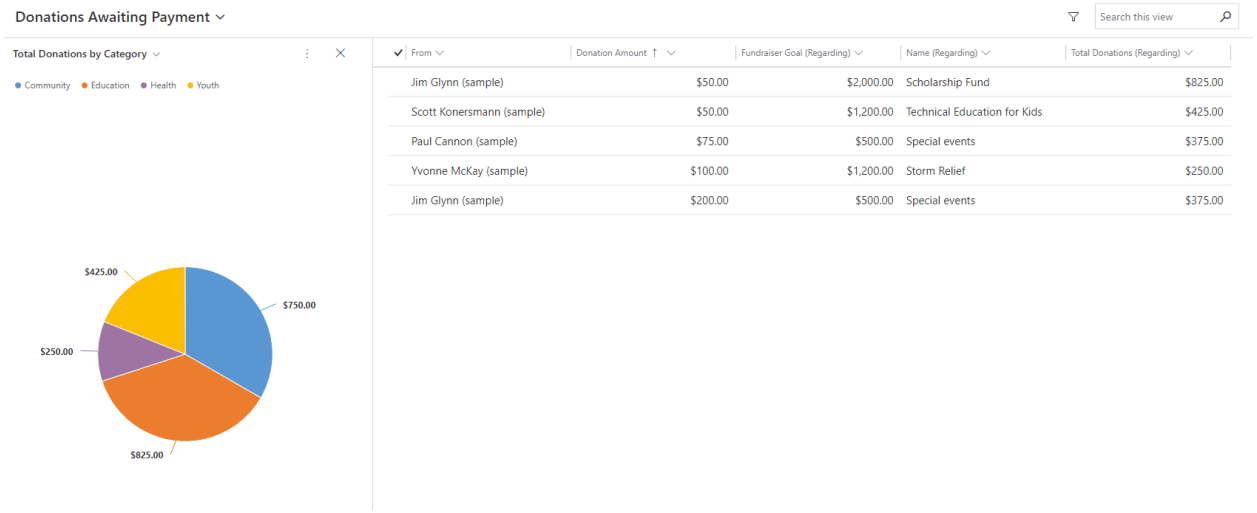


Legend:

1. **Sort**: Sorts rows. Use **Filter by** to filter both grid and chart data based on the filter criteria.
2. **Open advanced filtering panel**: See which filters were applied to the current view. For more information, go to Create and manage personal views on a grid page.
3. **Search this view**: Search for data in the view you're in.
4. **Jump bar**: The page jump bar uses the first sorted column. If no change has been made to the sort order, the jump bar uses the primary column.

Drill-down in the chart data

You can drill down into a segment on a chart and filter the data for that segment. When a chart is shown together with grid, the drill-down option that you select will filter both chart and grid as shown in this illustration.



> **Note**
>
> Chart drill-down functionality is not available on charts in a subgrid or an associated grid.

For more information on how to use the drill-down feature, go to Drill-down in a chart.

Feedback

Was this page helpful? Yes No

Provide product feedback

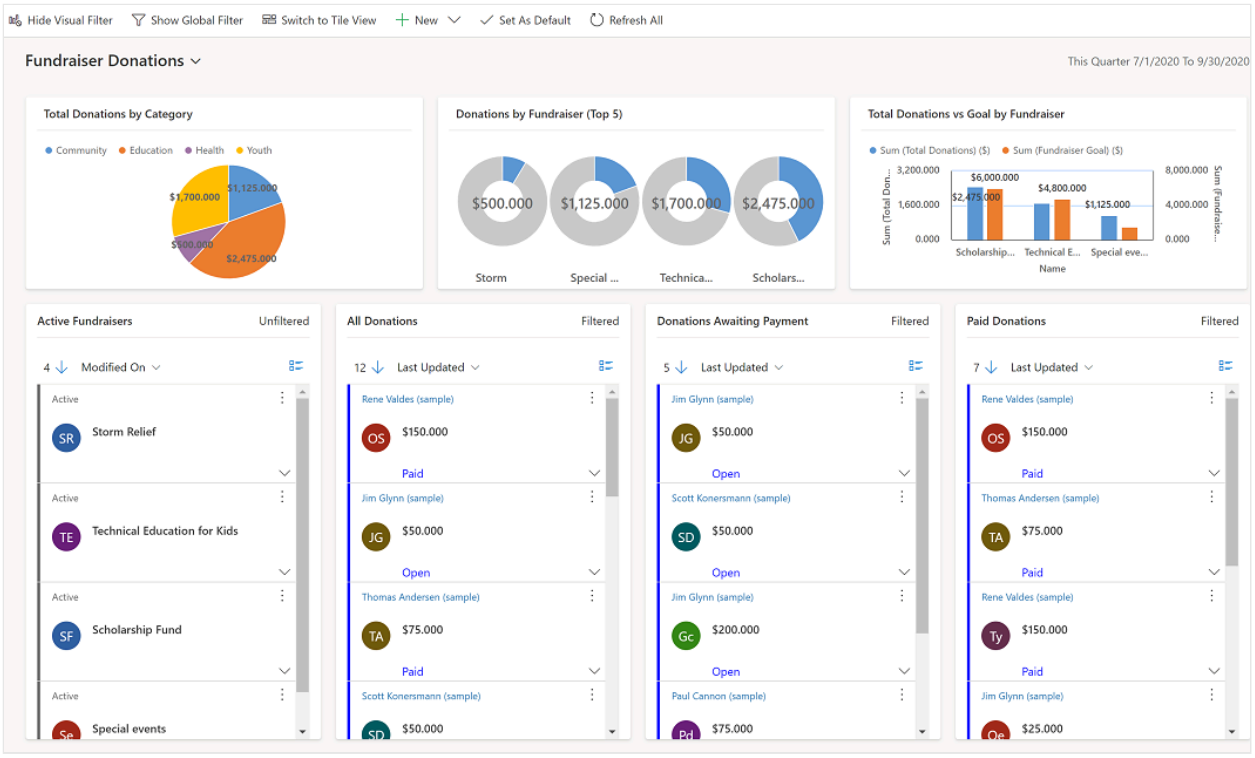
Use interactive dashboards to filter data

Article • 02/15/2022

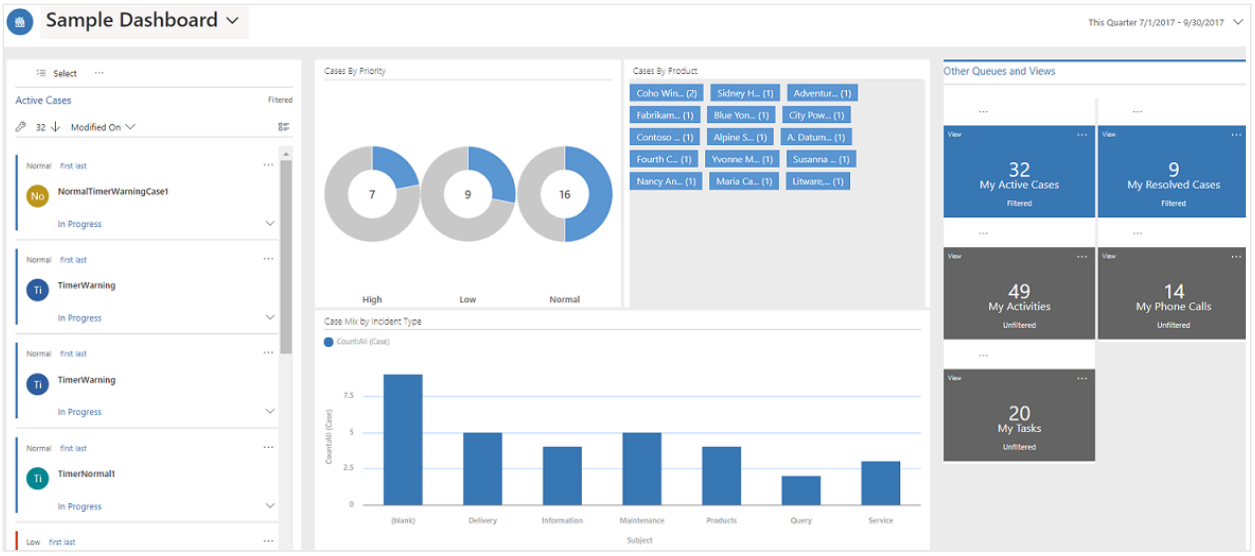
Use interactive dashboards to filter and view your app data. Charts on an interactive dashboard serve as a simple filtering source. You can select different data points on a chart to filter data based on the filter criteria.

Interactive dashboards are system dashboards that only an admin can create. To learn more about how to create and configure interactive dashboards, go to Create and configure model-driven app interactive experience dashboards.

Interactive dashboards come in two forms: multi-stream and single-stream. In addition, multi-stream dashboards can be home page or table-specific dashboards. The multi-stream dashboards display data in real time over multiple data streams.



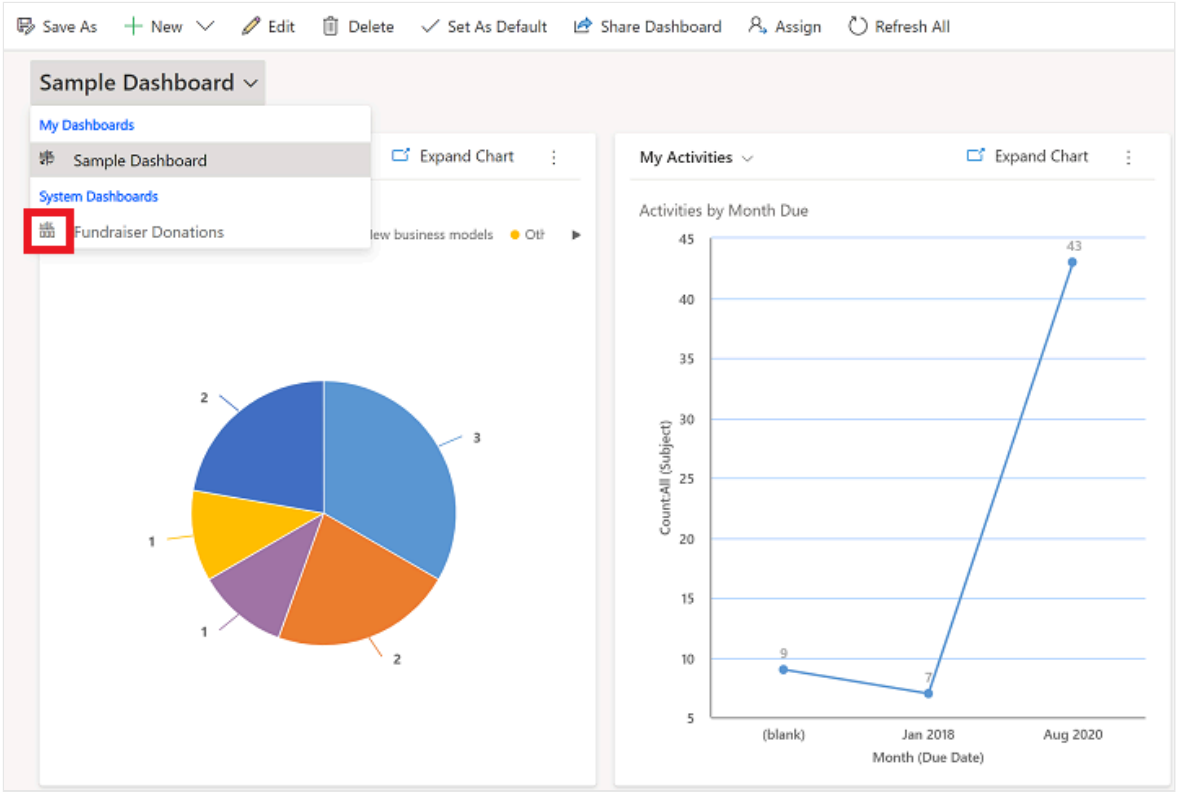
The single-stream dashboards display real-time data over one stream based on a table view. Visual filters are already applied. The tiles are positioned on the right side of the dashboards and are always shown.



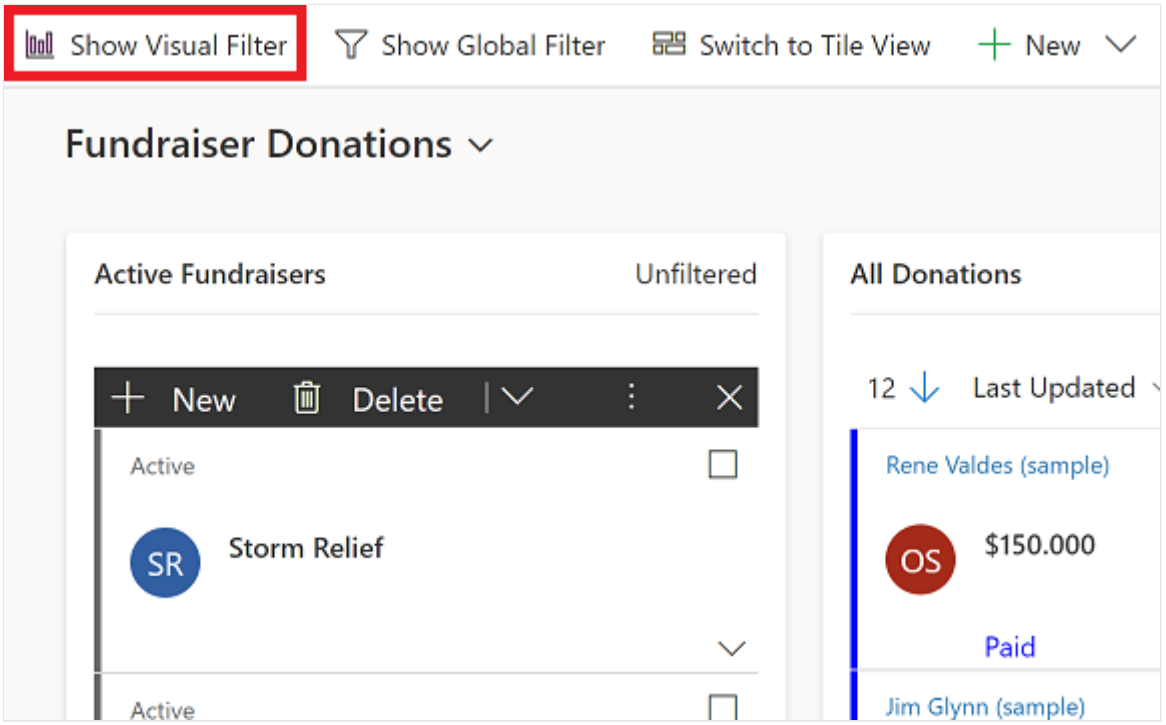
Use visual filters

Use the visual filter option to drill down into the data on the charts. This displays information that interests you most.

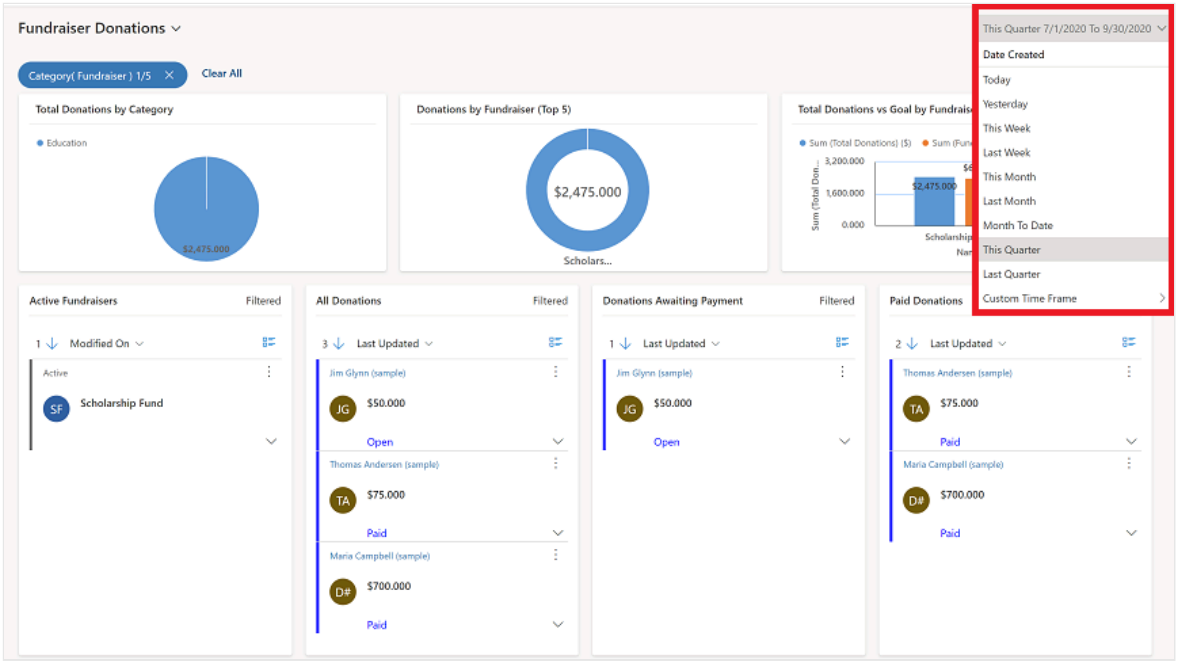
1. To view an interactive dashboard, go to the list of dashboards and find the one that has the interactive icon  next to the dashboard name.



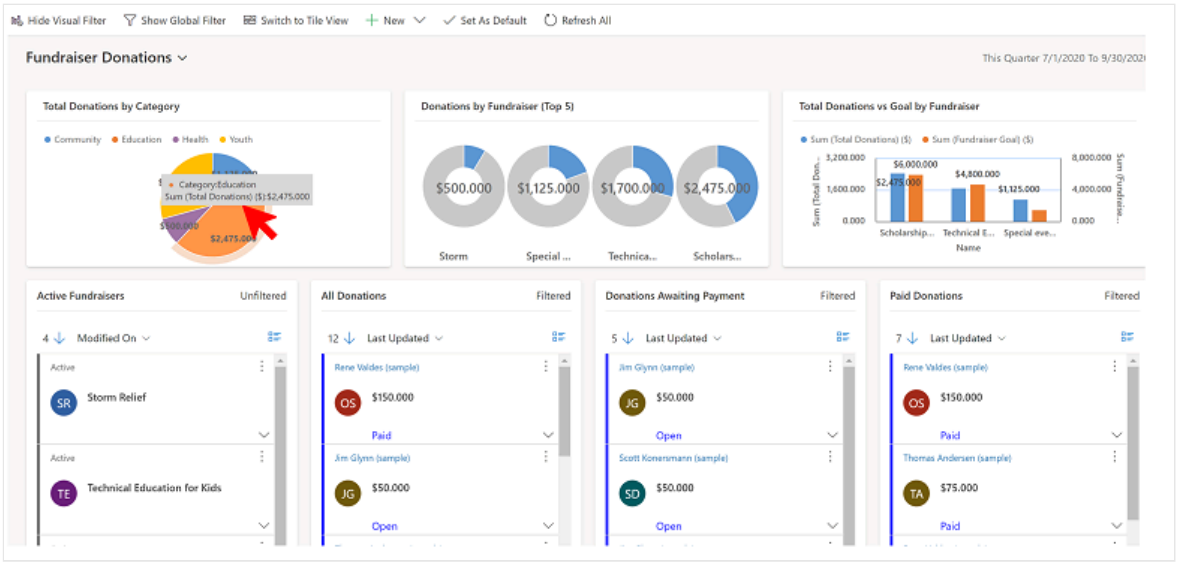
2. On the command bar, select **Show Visual Filter**.



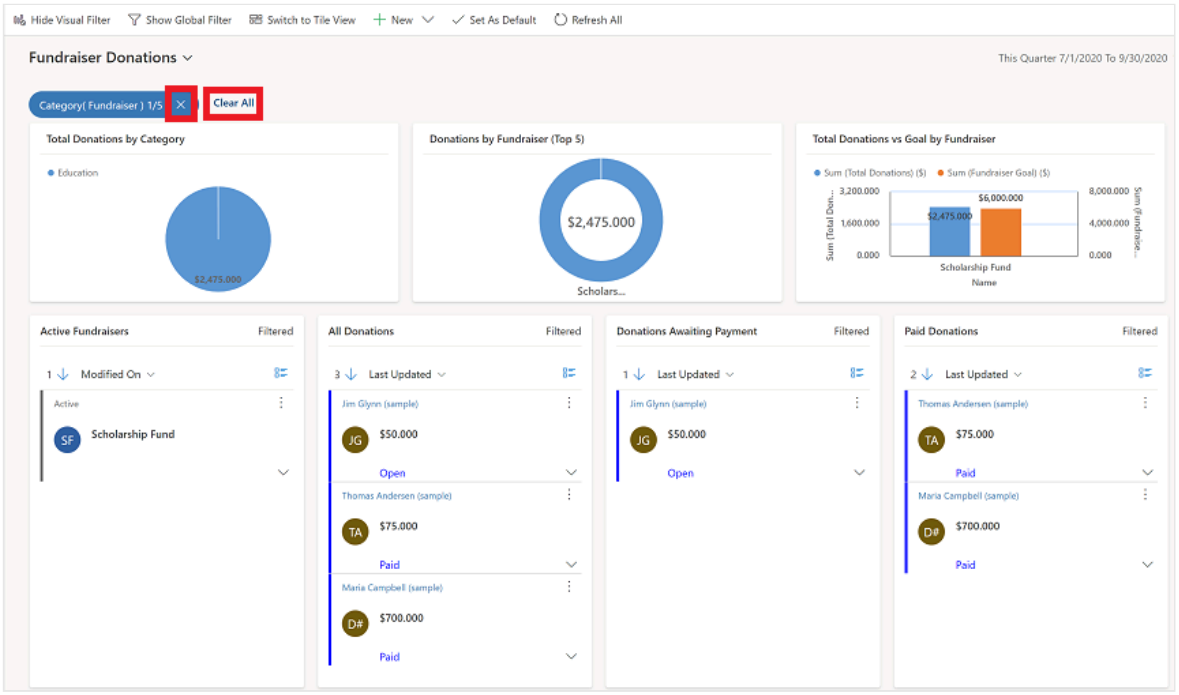
3. Select a drill-down option.



4. Select an area on one of the charts or graphs to apply a filter.



5. To clear the filter, select **X** next to the filter or to clear all filters, select **Clear All**.



Feedback

Was this page helpful? Yes No

Provide product feedback

Add or edit Power BI visualizations on your dashboard

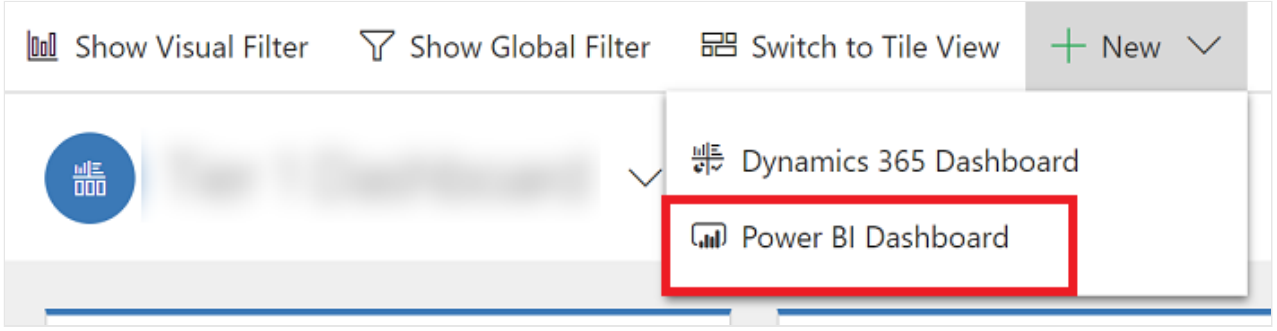
Create rich, interactive reports and real-time visualizations with Power BI dashboards and tiles that you add to your personal dashboards.

> **Note**
>
> To add Power BI visualizations to personal dashboards in your model-driven app, you must:
>
> - Enable Power BI visualizations for your organization in **Advanced Settings** > **System** > **Administration** > **Reports** > **Embed Power BI visuals**.
> - Have a Power BI account and have access to at least one Power BI dashboard.
> - Enable third-party cookies in your browser, otherwise authentication to Power BI will fail.

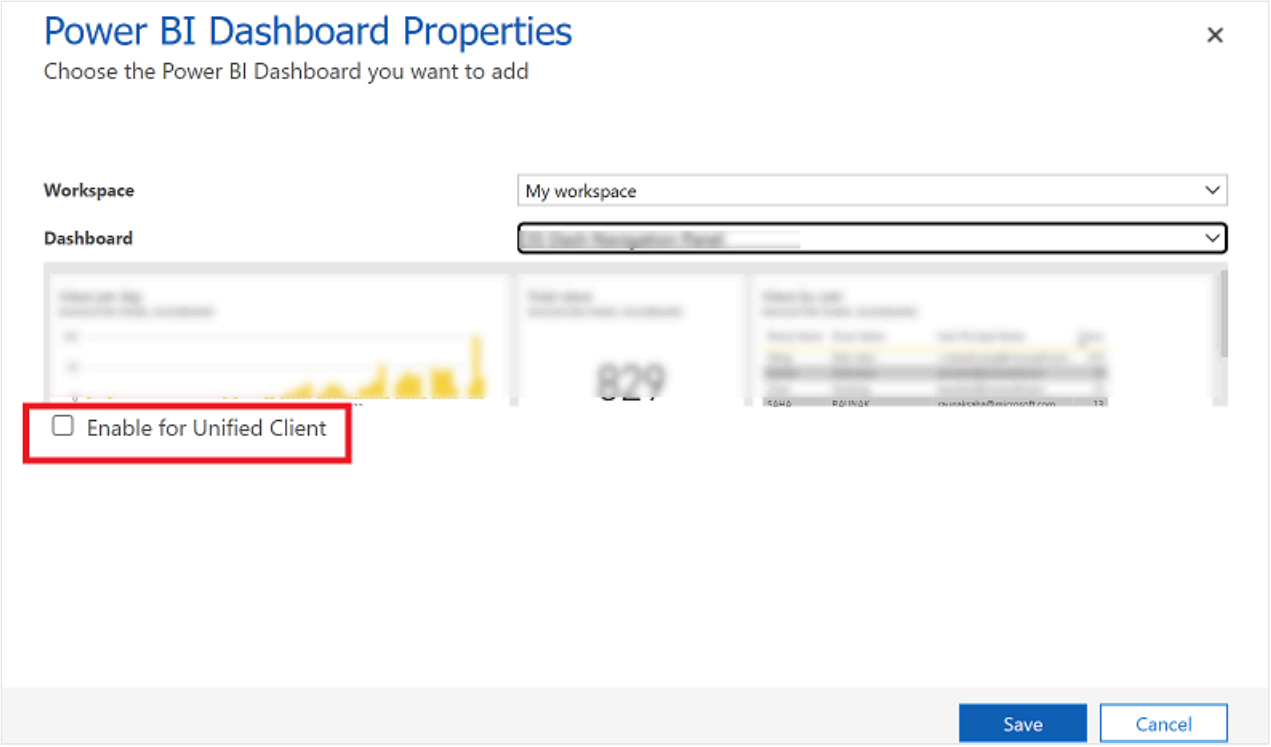
Create a personal Power BI dashboard

Follow these steps to add a Power BI dashboard to your model-driven app. If you are connecting to the Power BI service, you need an account and to have selected your Microsoft Dataverse instance as a data source. For more information about registering and connecting data sources, see Microsoft Power BI.

1. Open your app and go to **Dashboards**.
2. Select **New** and then select **Power BI Dashboard**.



3. In the **Power BI Dashboard Properties** dialog select the workspace and then select the Power BI dashboard that you want to embed in your dashboard. Select **Enable for Unified Client** if you want to make the dashboard available for Dynamics 365 for tablets and Dynamics 365 for phones.

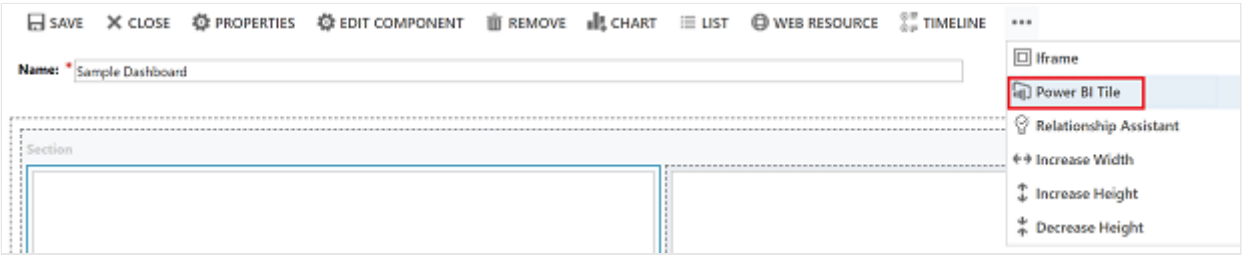


4. Select **Save** to save your dashboard.

Embed Power BI tiles on your personal dashboard

Follow these steps to add one or more Power BI tiles to your personal dashboard. If you are connecting to the Power BI service, you need an account and to have selected your Dataverse instance as a data source. For more information about registering and connecting data sources, see Microsoft Power BI.

1. Open your app and go to **Dashboards**.
2. Select an existing personal dashboard or select **New** to create one.
3. On the dashboard, select an area where you want the tile to appear, and then select **Power BI Tile** on the toolbar.



4. In the **Power BI Tile** dialog, select the workspace and then select the Power BI tile that you want to display on your dashboard. Select **Enable for mobile** if you want to make the tile available for Dynamics 365 for tablets and Dynamics 365 for phones.

Select another area of the dashboard and repeat this step to add another Power BI tile, or other component, such as a chart or list, to your dashboard.

5. Select **Save** to save your dashboard.

> **Note**
>
> If a user clicks on the title of an embedded Power BI dashboard, a modal popup will be triggered, showing the dashboard along with **Refresh** and **Open in Power BI** actions.

Share a personal dashboard that contains Power BI visualizations

To share your personal dashboard that contains Power BI visualizations you must configure sharing in both Dataverse and Power BI, and the user or group must have the same credentials and appropriate level of access in both services. To share your personal dashboard in your app go to, **Dashboards**. In the list of dashboards, select the personal dashboard you want, and then select **SHARE DASHBOARD**. For more information about sharing a dashboard in Power BI, see Power BI: Share a dashboard with colleagues and others.

Privacy notice

By enabling the embedding of Power BI tiles and dashboards, when a user embeds a Power BI tile or dashboard, that user's Microsoft Entra ID authorization token for Microsoft Dataverse is used to authenticate with the Power BI service with an implicit grant, providing a seamless "single-sign on" experience for the end user.

An administrator can disable embedding of Power BI tiles and dashboards at any time to stop use of the Dynamics 365 authorization token for authenticating with Power BI service. Any existing tiles or dashboards will stop rendering for the end user.

The Azure component or service that is involved with embedding of Power BI tiles is detailed in the following section.

Note: For more information about additional Azure service offerings, see the Microsoft Azure Trust Center.

Microsoft Entra ID

This service provides the authentication token exchanged with Power BI service for API and UI authentication.

Last updated on 03/13/2026

Collaborate with others in model-driven apps

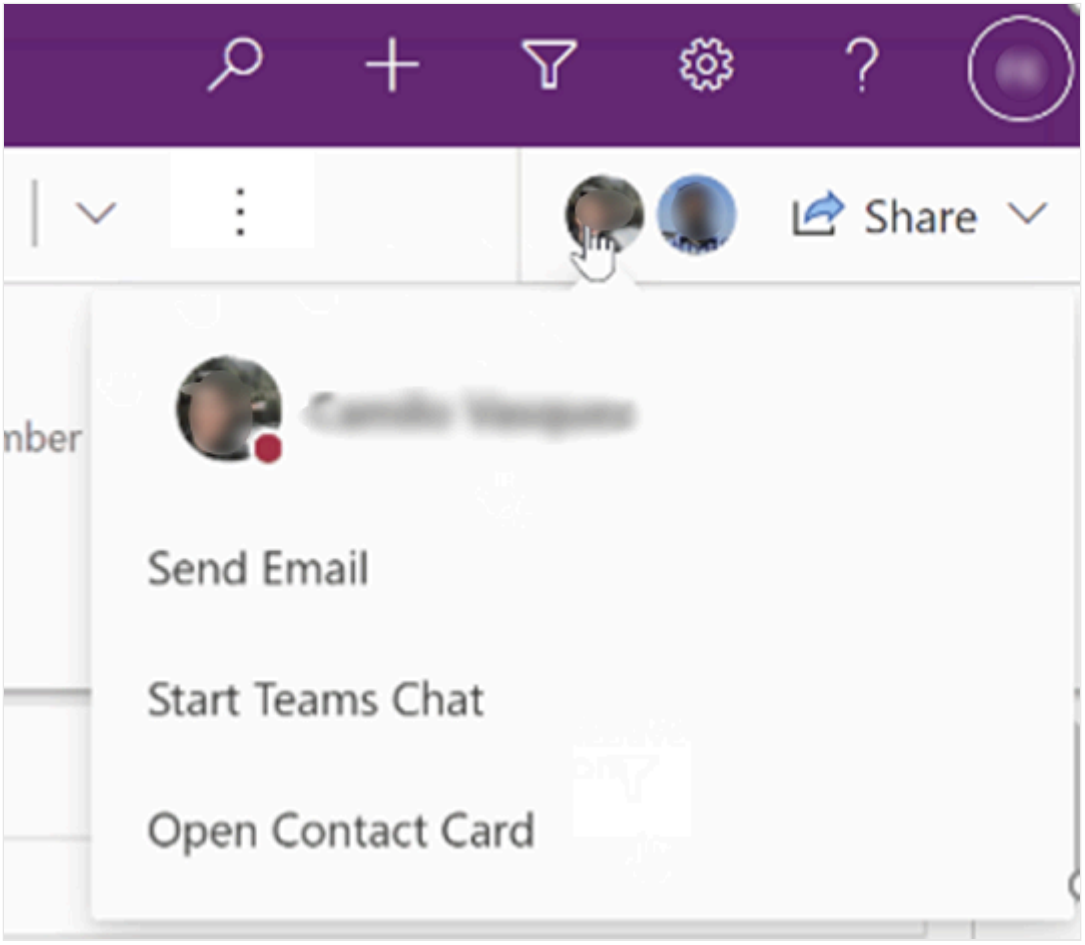
Article • 10/10/2023

Collaborate with colleagues and share links in model-driven Power Apps the same way do you as in Microsoft Office.

Multiplayer apps

When you're working on a record, you can also see other users who are working on the same record.

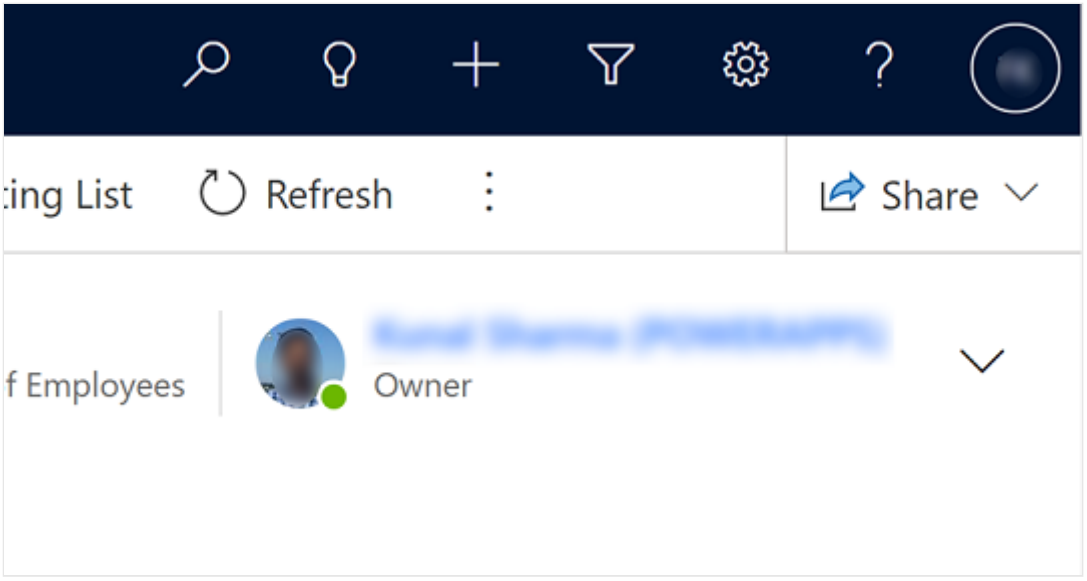
You can also select a user's picture to see their online status, send them an email, or start a Teams chat.



This feature uses the Azure Fluid Relay service, which must be available in your region. For more information, see Azure Fluid Relay service availability. Because this feature uses the Azure Fluid Relay service, users must be able to access fluidrelay.azure.com.

View a user's status and picture

App users, such as the record owner, as well as the users appearing in grids and lookups are displayed with their picture and online status. This feature is now available on all model-driven apps and cannot be disabled.



> **Note**
>
> This feature will be extended to user lookups and grid user columns.

Share

You can now easily share records with your colleagues using the new **Share** button. Select **Copy link** or **Email link** in order to send a record link to your colleagues.

> **Note**
>
> If you have the share privilege on a record and you email a record link to a colleague, the colleague will automatically get read access to the record.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use OneDrive for Business

Article • 02/15/2022

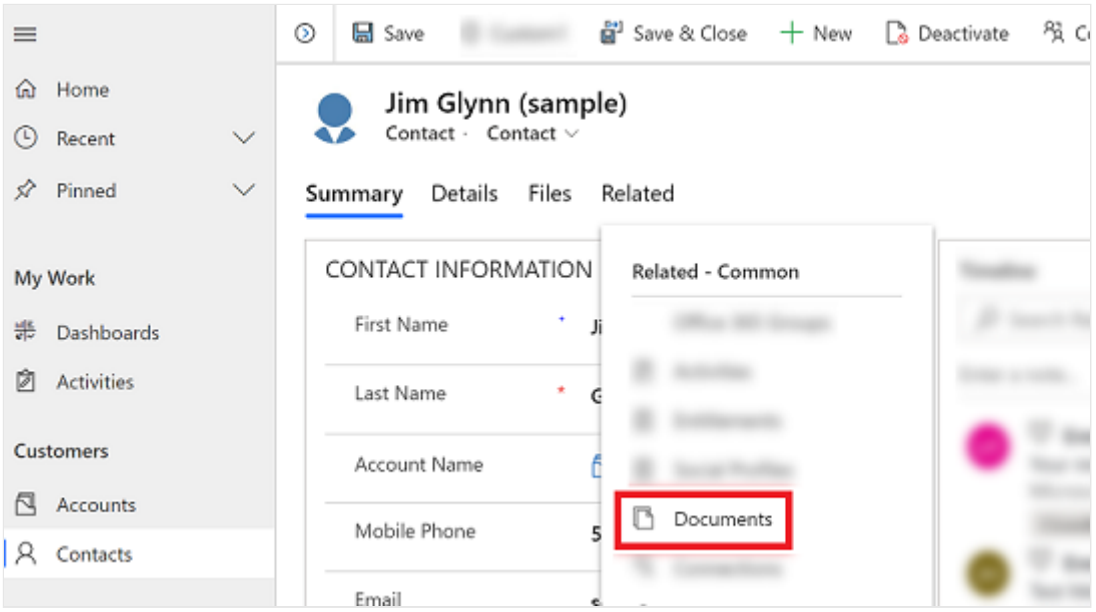
Create and manage private documents from within Microsoft Dataverse apps by using OneDrive for Business. More information: What is OneDrive for Business?

Before you can use OneDrive for Business, it must be enabled by your system administrator. More information:

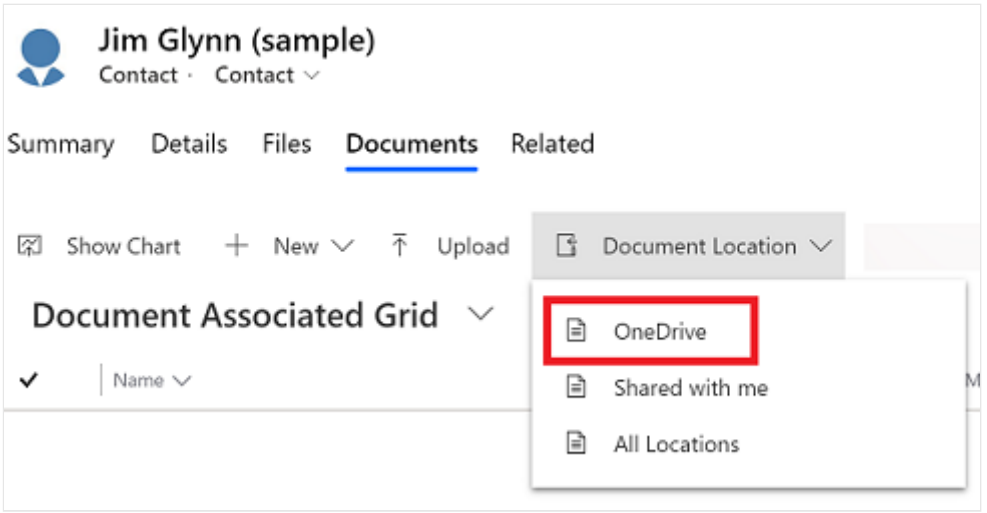
- Find your administrator or support person
- Enable OneDrive for Business

The first time you view your documents

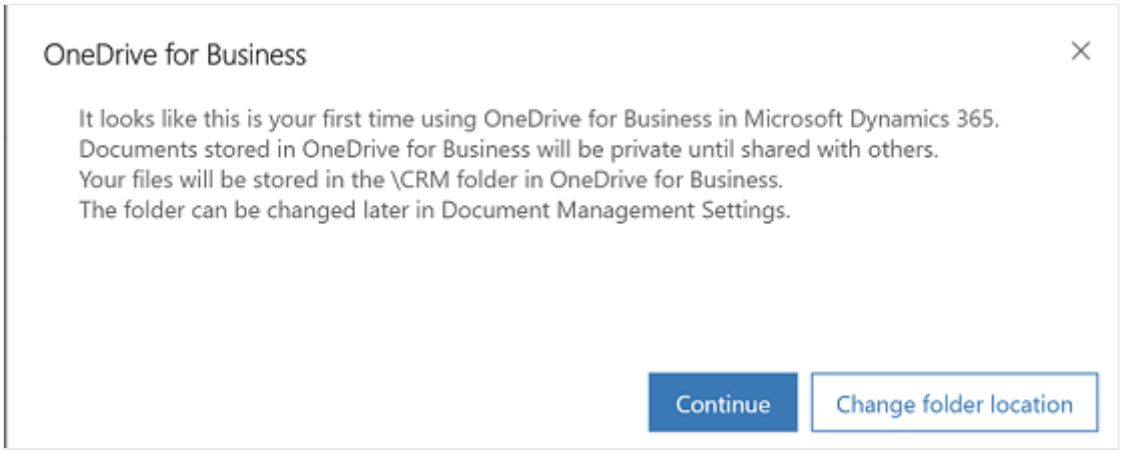
1. Open a row and go to the **Document Associated Grid** view. For example, open a contact row.
2. On the open row, select the **Related** tab, and then select **Documents**.



3. Select **Document Location** > **OneDrive**.



4. After OneDrive for Business is enabled, you'll see the following dialog box when you go to the **Documents** tab to view documents in Dataverse and upload a file to OneDrive, or when you attempt to create a new document or folder.

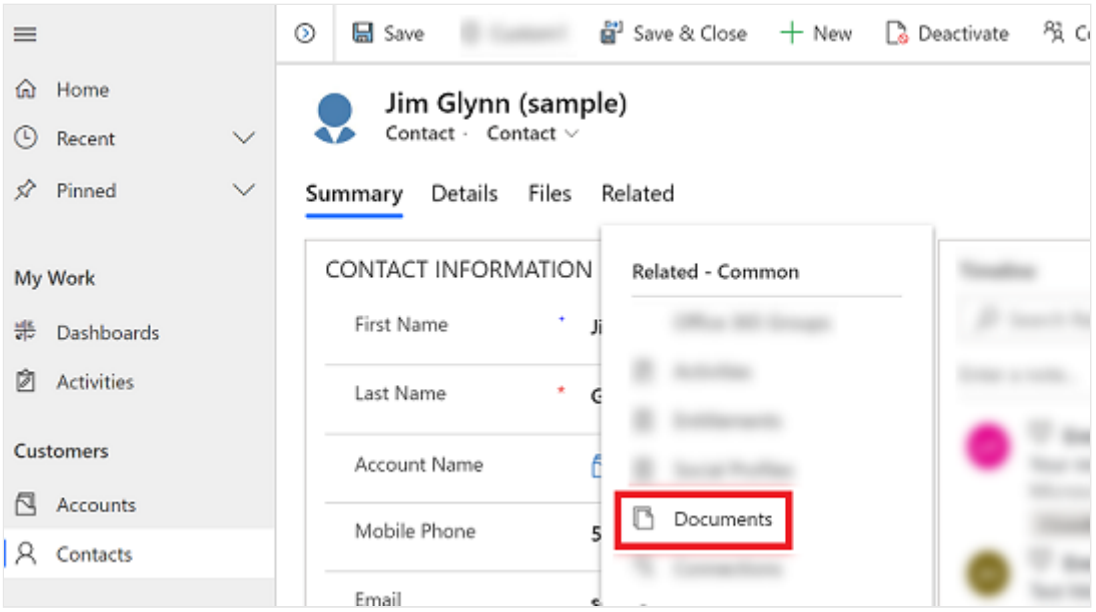


5. Select **Change folder location** to pick a new location to store OneDrive documents, or select **Continue** to accept the default folder location.

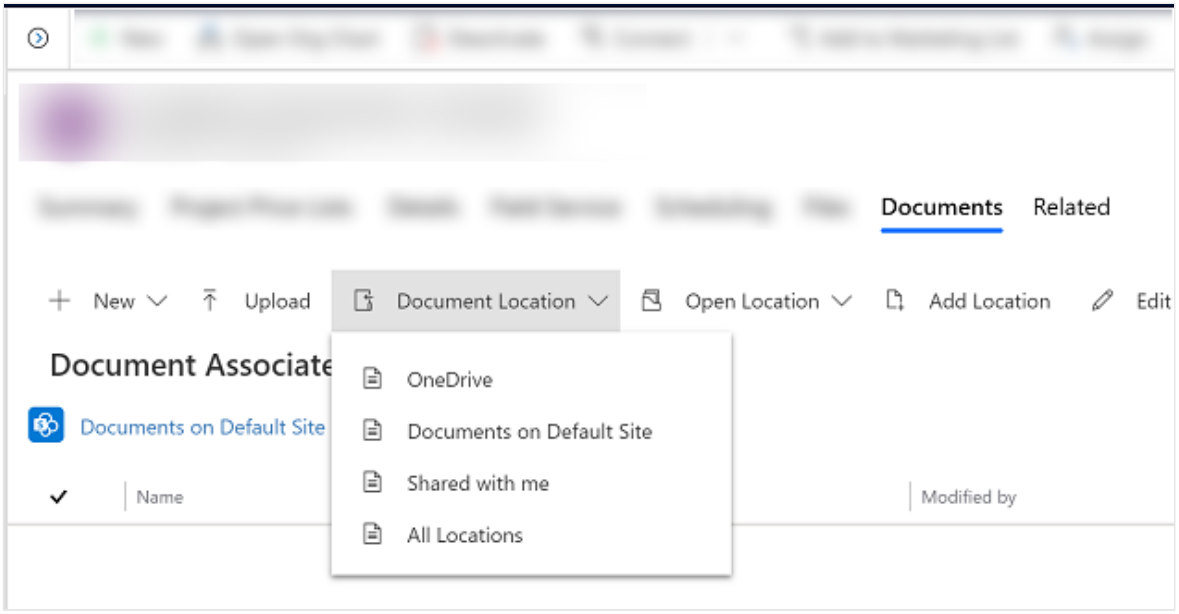
View existing OneDrive documents

1. Open a row and go to the **Document Associated Grid** view. For example, open a contact row.

2. On the open row, select the **Related** tab, and then select **Documents**.



3. Select **Document Location** to filter the document list.



4. Select a location as described in the following table.

Expand table

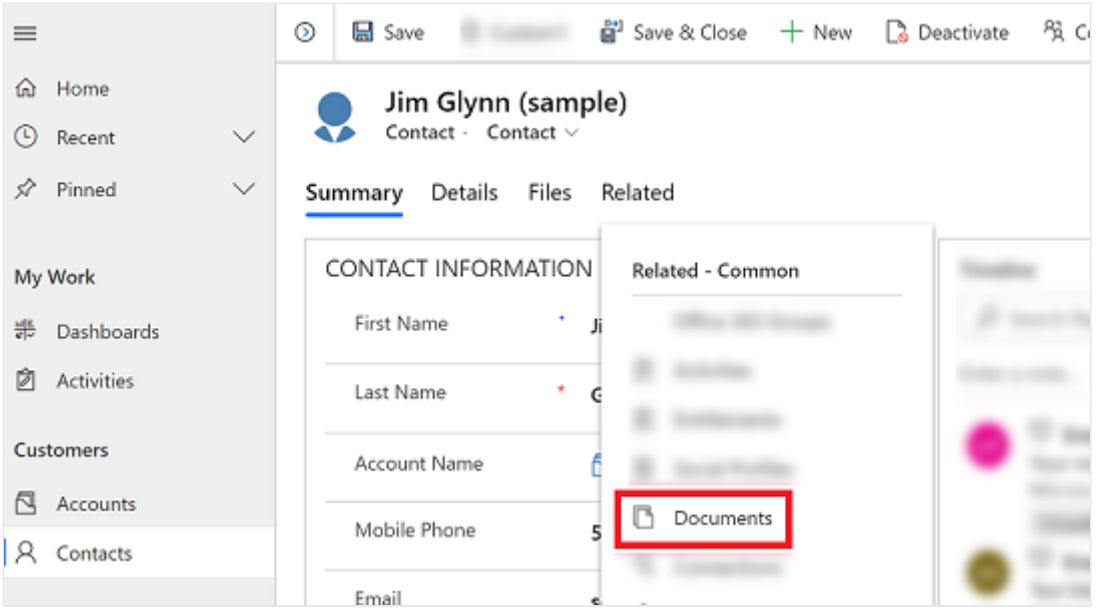
Document Location	Description
OneDrive	Documents stored in OneDrive for Business
Documents on Default Site	Documents stored in your default SharePoint site
Shared with me	Documents that others shared with you that are associated with this row
All Locations	All document locations associated with this row

5. After you select a location, you'll see the documents saved in that location.

Create a new document and save it to OneDrive

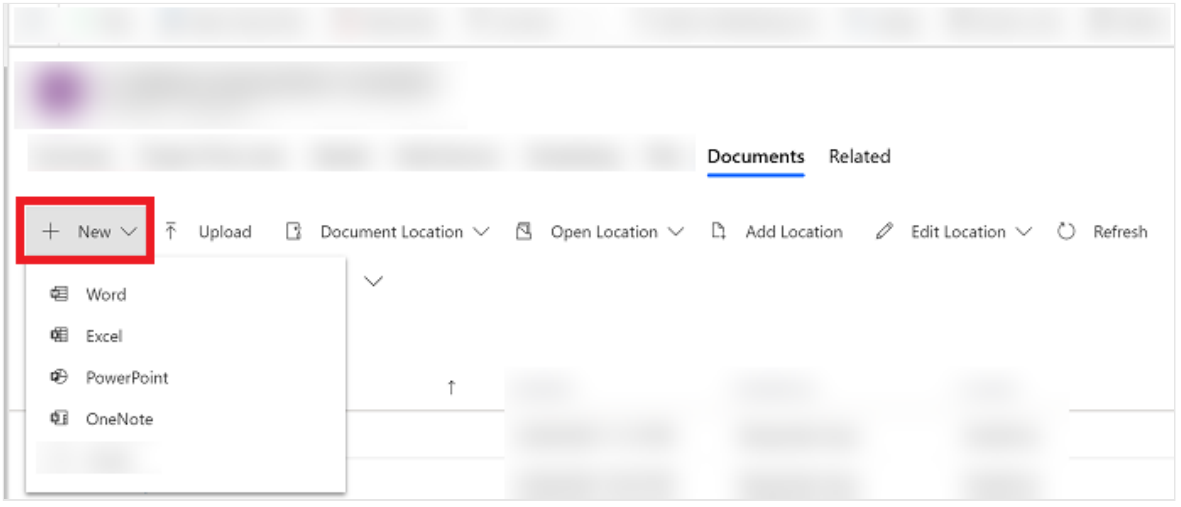
1. Open a row and go to the **Document Associated Grid** view. For example, open a contact row.

2. On the open row, select the **Related** tab, and then select **Documents**.



3. Select **Document Location**, and change the location to **OneDrive**.

4. Select **New**, and then choose a document type such as PowerPoint or Word.



5. Enter a document name, and then select **Save**.

Things to consider

Be aware of the following regarding OneDrive for Business in Dataverse:

- OneDrive storage folders are created in the user's current Dataverse language. If the language changes, new folders will be created in the new language. Old folders will remain in the previous language.
- There might be a delay between when the documents are shared in OneDrive and when they're available to other users.

Feedback

Was this page helpful? Yes No

Provide product feedback

Take notes by using OneNote

Article • 02/15/2022

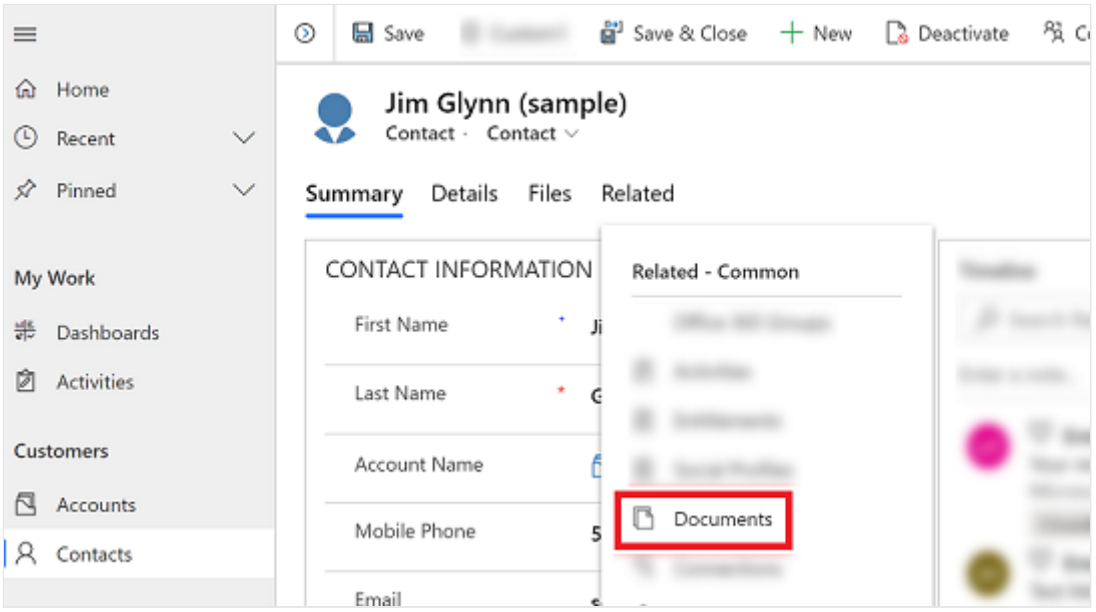
Use OneNote to take or review notes, ideas, plans, and research from the **Document Associated Grid** view of a row in Microsoft Dataverse.

Before you can use OneNote, it must be enabled by your system administrator. More information:

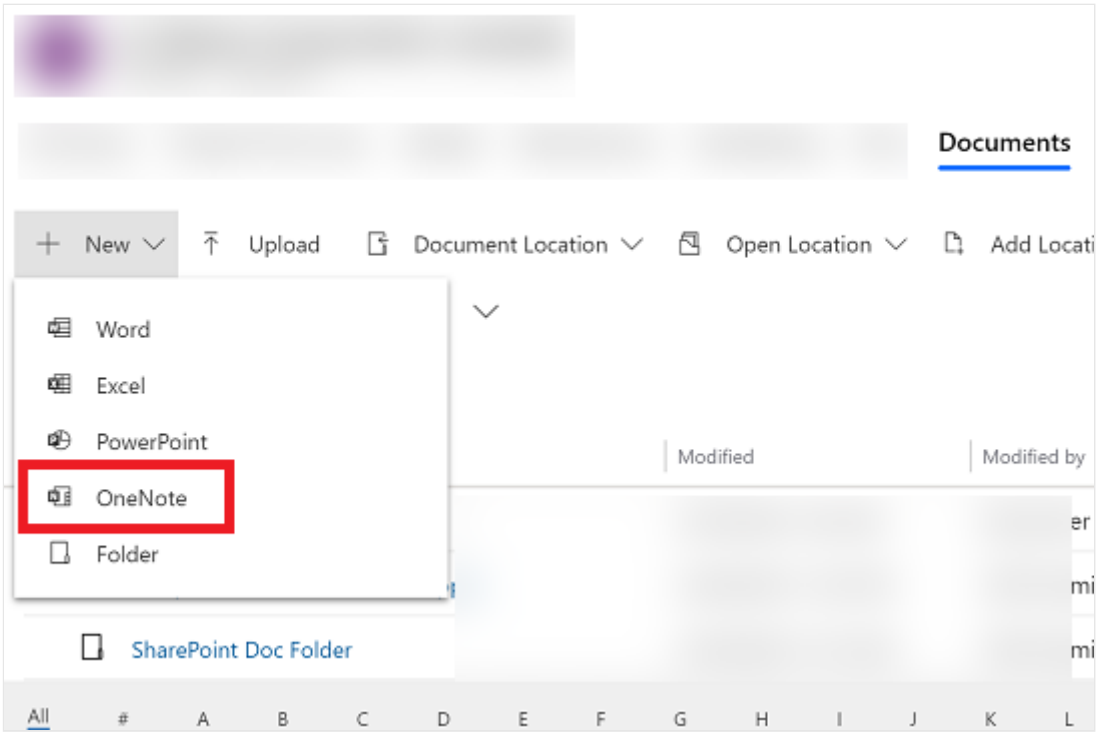
- Find your administrator or support person
- Set up OneNote integration

Start a new OneNote notebook

1. Open a row and go to the **Document Associated Grid** view. For example, open a contact row.
2. On the open row, select the **Related** tab, and then select **Documents**.



3. Select **Document Location**, and select the location where you want to save the notebook. For more information, see View existing OneDrive documents.
4. Select **New**, and then select **OneNote**.

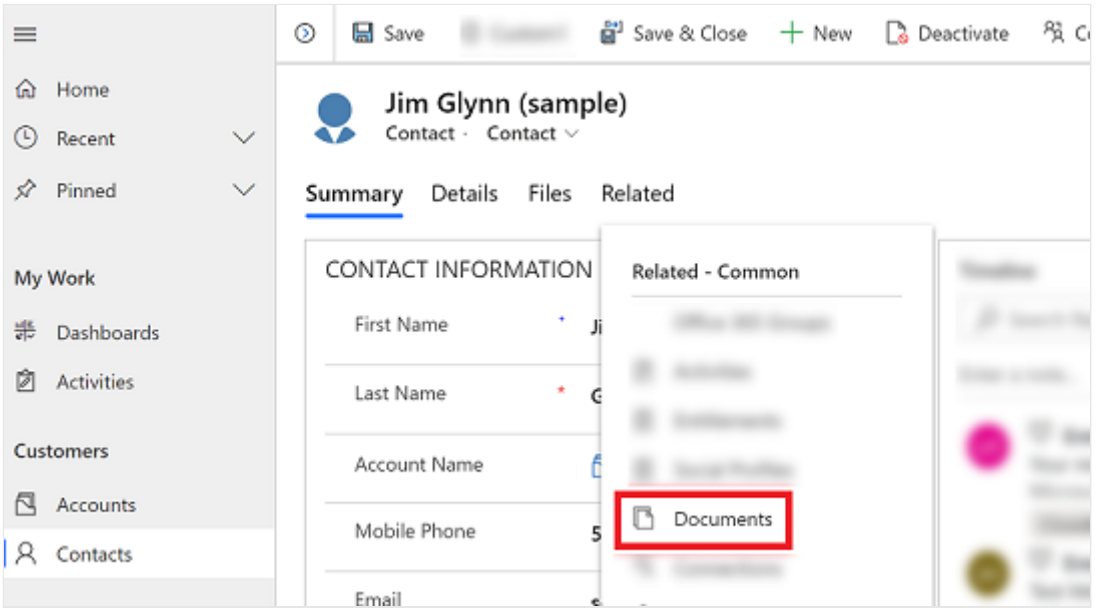


5. Enter a name for the notebook, and then select **Save**. A new notebook is created and opened.

Open an existing OneNote notebook

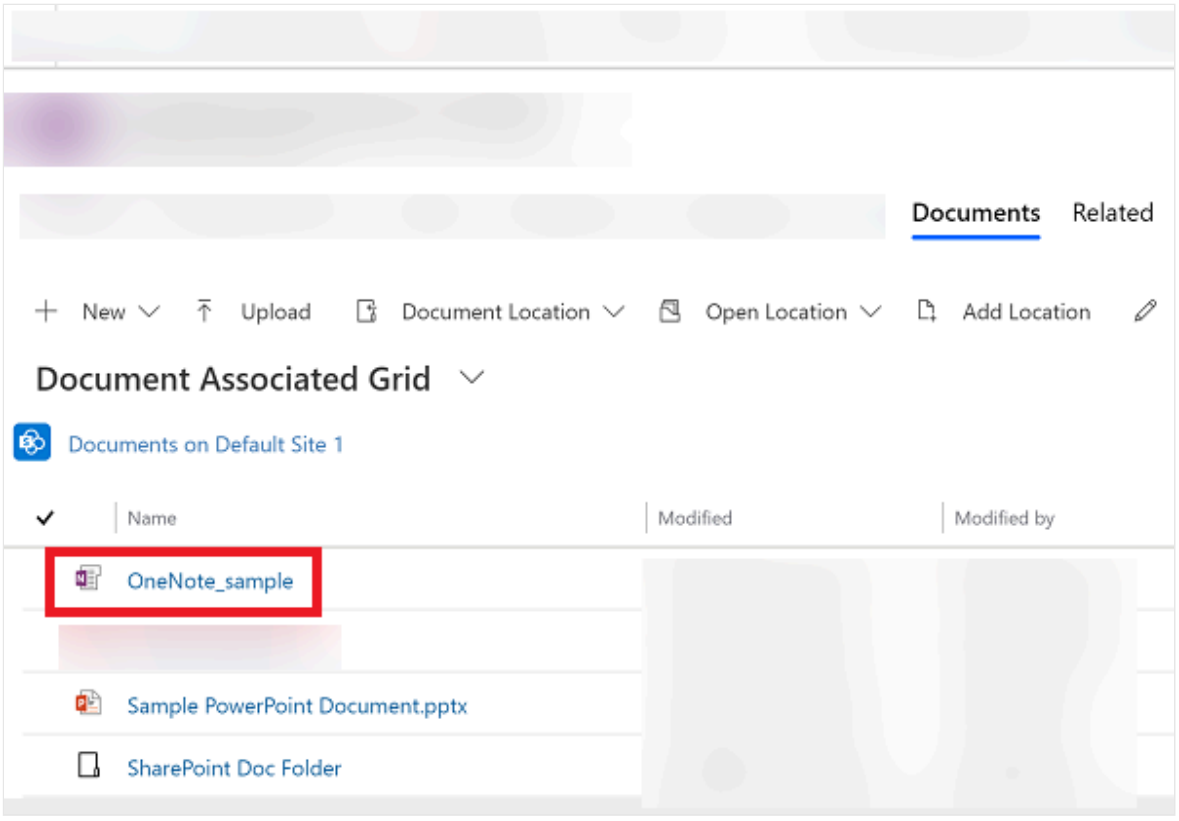
1. Open a row and go to the **Document Associated Grid** view. For example, open a contact row.

2. On the open row, select the **Related** tab, and then select **Documents**.



3. Select **Document Location**, and select the location of your notebook. For example, choose **OneDrive** if your notebook is stored on OneDrive. More information: View existing OneDrive documents

4. Select your notebook to start adding notes.



Feedback

Was this page helpful? Yes No

Provide product feedback

Collaborate using SharePoint

07/15/2025

With Microsoft Dataverse, you can store your documents on SharePoint and manage them from within your app. The documents that you create in your app are stored on SharePoint, and are automatically synced to your desktop and mobile devices.

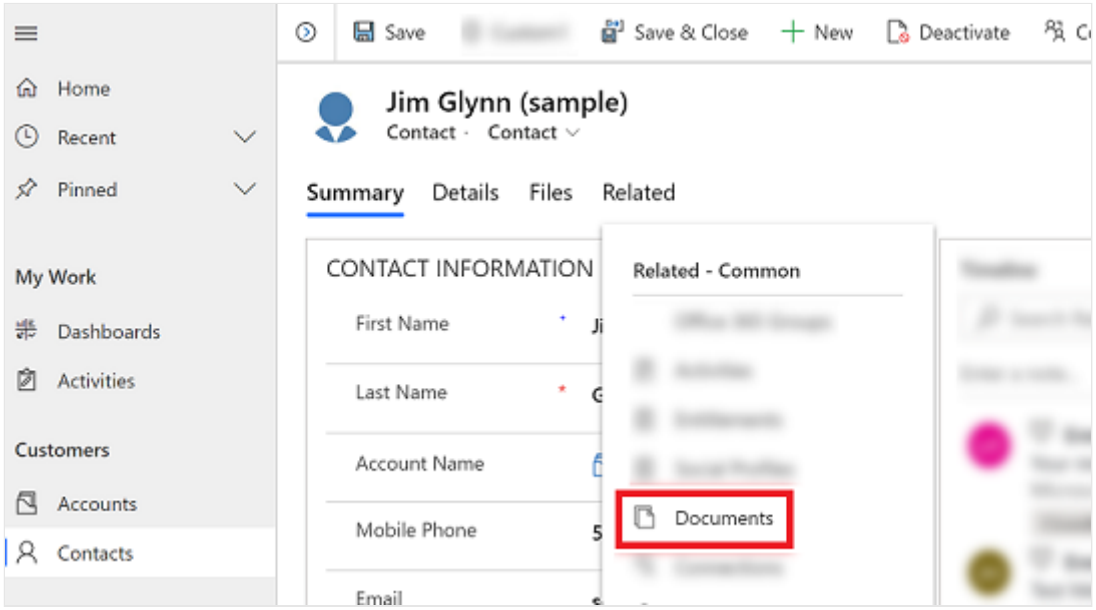
Before you can use SharePoint to store documents, it must be enabled by your system administrator. More information:

- Find your administrator or support person
- Manage your documents using SharePoint

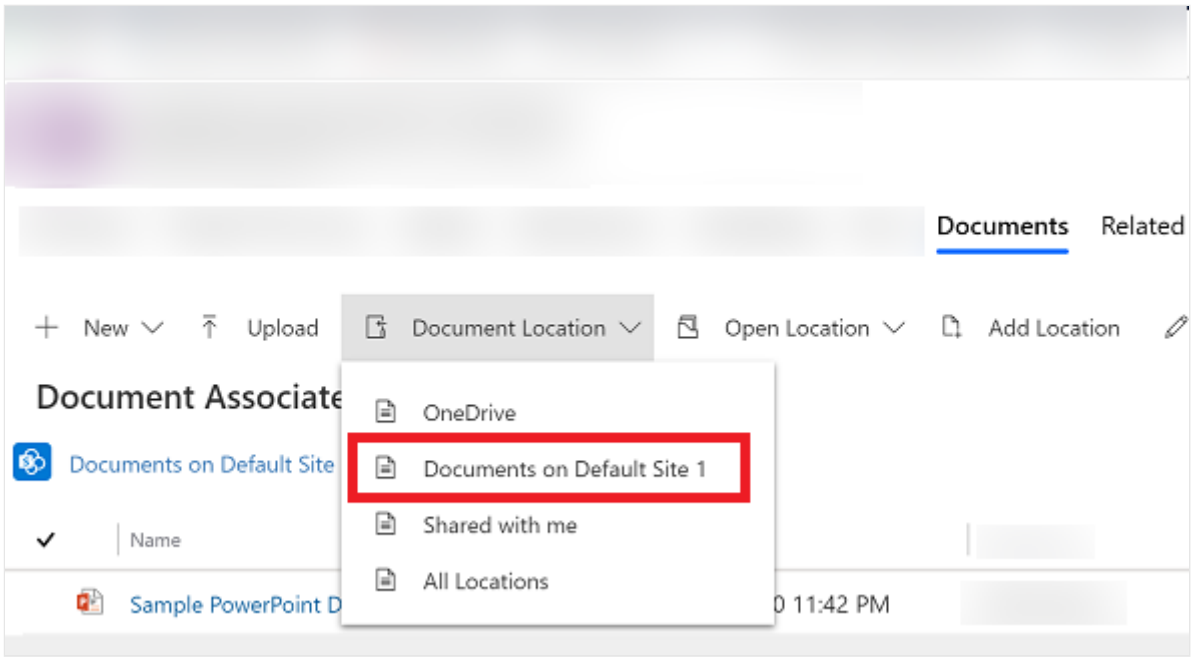
> **Important**
>
> - When you delete a SharePoint managed document using a Power Apps or Dynamics 365 customer engagement app, the app permanently deletes the document from both the app and the SharePoint site.
> - When you merge records, the app doesn't merge SharePoint document libraries. The app deactivates the subordinate record, but its SharePoint folder and documents remain accessible unless you delete them manually.

Where do you access the documents from?

1. For row types that support document management, open the row, select the **Related** tab, and then select **Documents**.



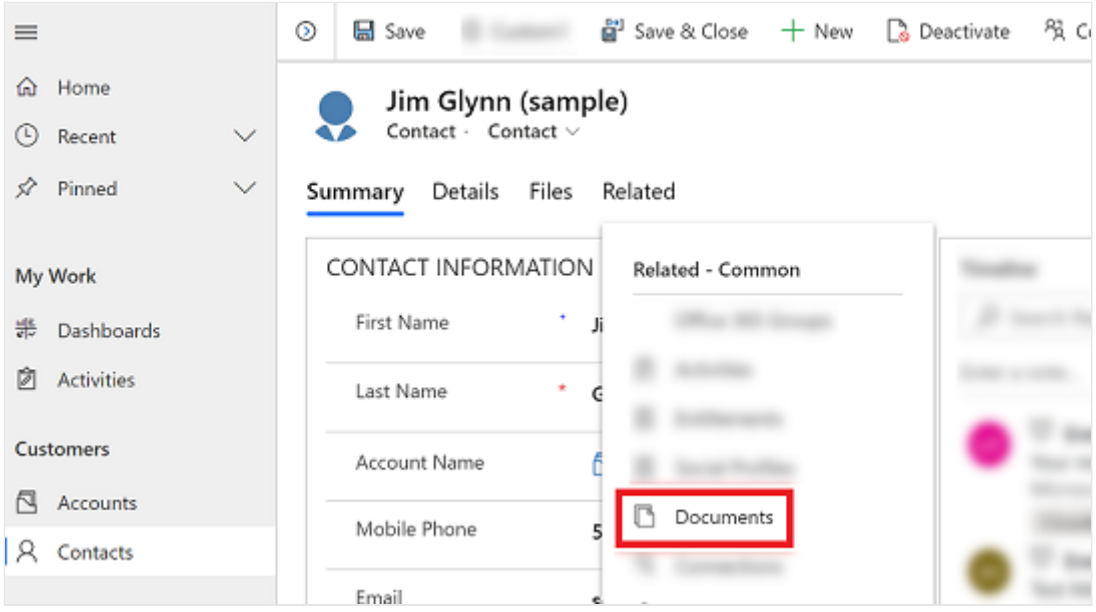
2. Select **Document Location** > **Documents on Default Site 1**. When SharePoint is enabled, the location is set to **Documents on Default Site 1** by default.



Create a new document and save it to SharePoint

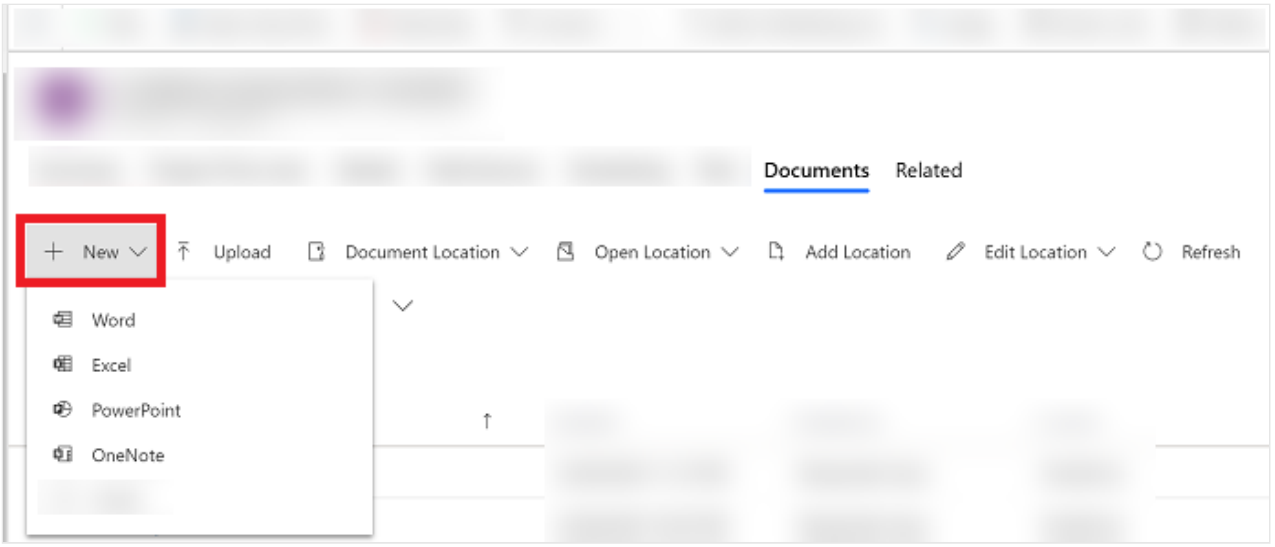
1. Open a row and go to the **Document Associated Grid** view. For example, open a contact row.

2. On the open row, select the **Related** tab, and then select **Documents**.



3. Select **Document Location**, and change the location to **Documents on Default Site 1**.

4. Select **New**, and then choose a document type such as Word, Excel, or PowerPoint.

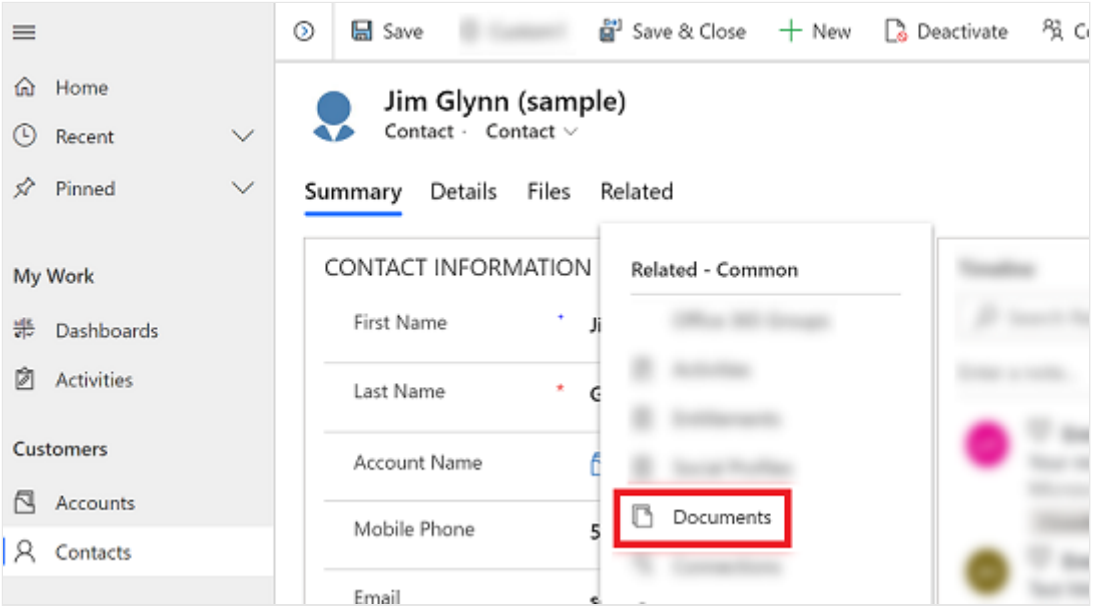


5. Enter a document name, and then select **Save**.

Create a new folder in the default SharePoint site location

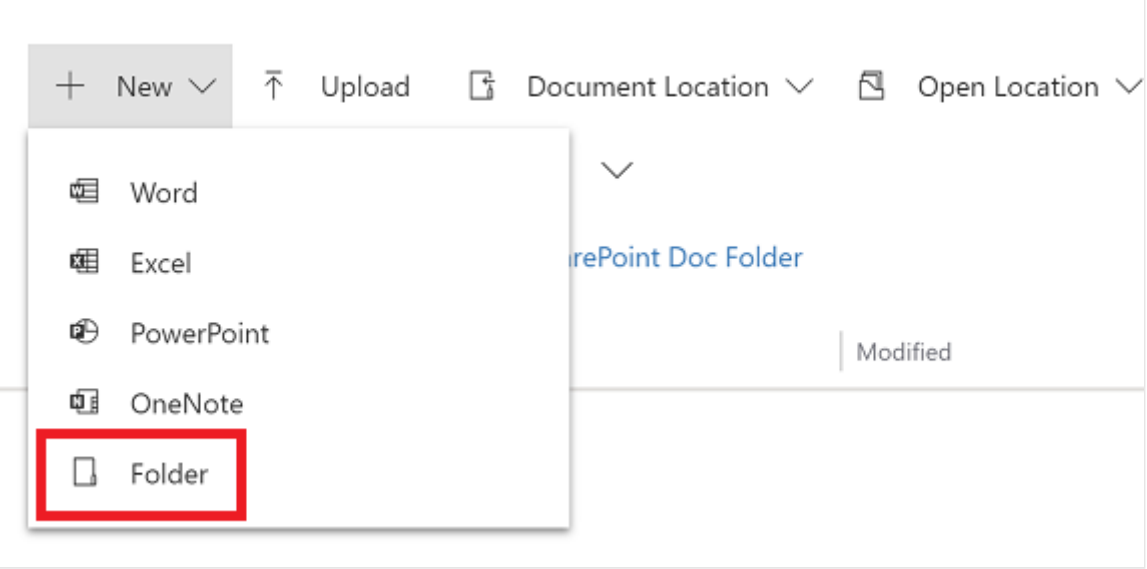
1. Open a row and go to the **Document Associated Grid** view. For example, open a contact row.

2. On the open row, select the **Related** tab, and then select **Documents**.



3. Select **Document Location**, and change the location to **Documents on Default Site 1**.

4. Select **New**, and then choose **Folder**.

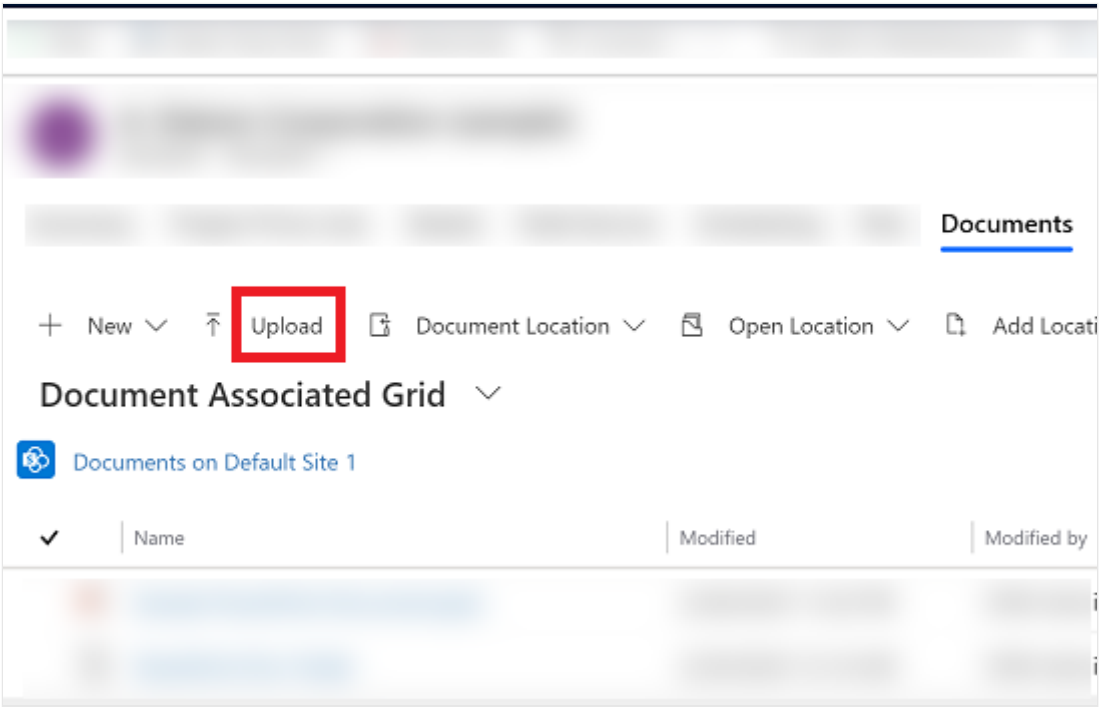


5. Enter a folder name, and then select **Save**.

Upload an existing document to SharePoint from your app

1. Go to the row you want to create the document for, select the **Related** tab, and then select **Documents**.

2. Select **Upload**.



3. Choose the file you want to upload. You can choose only one file at a time.

   The document is created in the current document location you're in.

> **Note**
>
> You can upload a file of up to 50 MB. If your internet connection is slow, you might get an error while uploading large files.

4. If files with the same name exist in SharePoint, select whether you want to overwrite the files.

5. Select **OK**.

Manage SharePoint locations

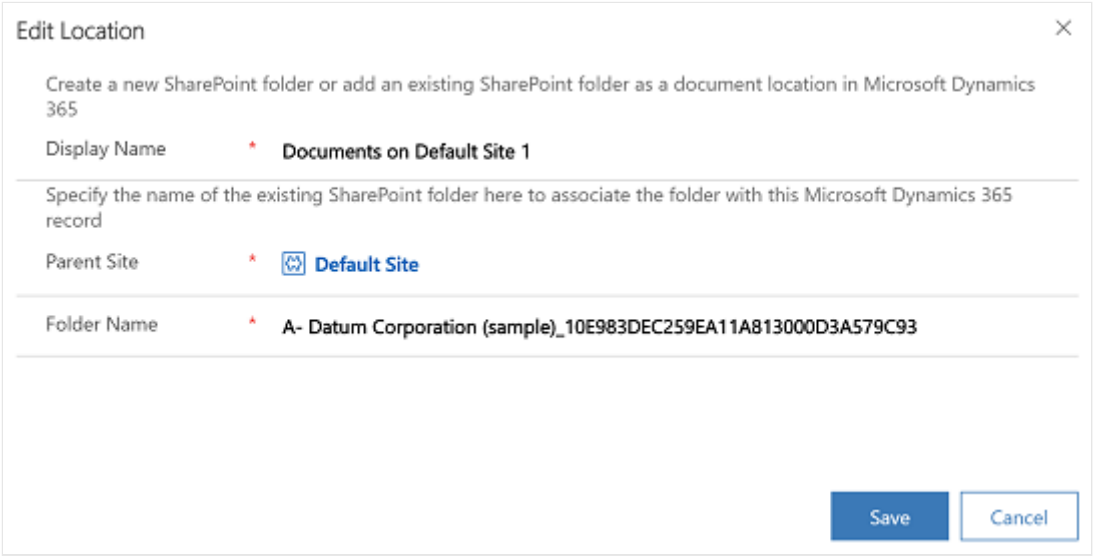
You can create new or edit existing SharePoint locations from your app in Dataverse.

Edit a location

1. Open a row, select the **Related** tab, and then select **Documents**.

2. Select **Edit Location**, and then select a SharePoint site location.

   The **Edit Location** dialog box appears.



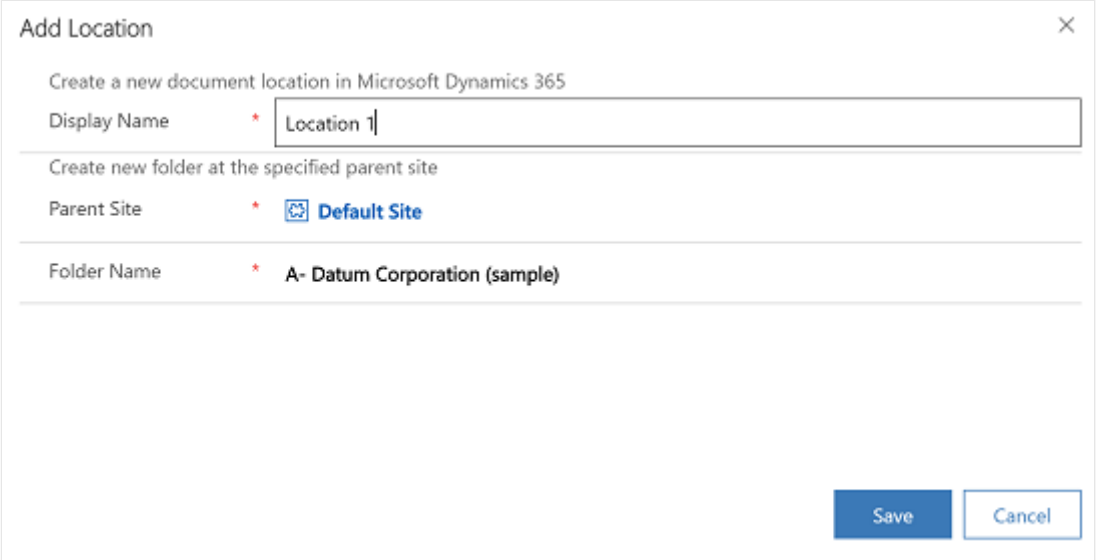
3. The display name, parent site, and folder name are automatically populated. Enter details about the new location, and then select **Save**.

Add a new location

1. Open a row, select the **Related** tab, and then select **Documents**.

2. Select **Add Location**.

The **Add Location** dialog box appears.



3. The display name, parent site, and folder name are automatically populated. Change the details if necessary, and then select **Save**.

Files tab FAQ

Why was the location to access documents moved?

- We moved the command to make documents easier to find with fewer clicks.

Has the Documents tab gone away?

- No, it hasn't gone away. Users can still access the documents associated with the row in question the old way, simply by selecting the **Related** menu and then the **Documents** link.

With the change, will subfolders in SharePoint still be created automatically?

- Yes. The behavior is similar to that of the **Documents** link under the **Related** menu. When a user selects the **Files** tab for the first time, the corresponding SharePoint subfolder is created by the system.

Is there a way to add the Files tab to other tables, or remove it?

- Yes. To add or remove the **Files** tab, follow the steps in this article: Add the SharePoint documents tab to the main form for a table

Use @mention to collaborate with your team using Notes

Article • 03/21/2025

Collaborate effortlessly by mentioning your coworkers in Notes on the timeline.

The administrator must enable the **@mention people when in rich text editor to add and notify user settings** in Power Platform admin center and enable in-app notifications for the app for this capability to be available.

> **ⓘ Important**
>
> Users must have the Read privilege for the **msdyn_postconfig** entity, as well as the Share privilege for the entity for which they use @mentions (for example, account, contact, case, and so forth).

@mention a user in Notes

1. Open a row and go to the activity timeline on the page.
2. Add a note by selecting *Enter a note*.
3. In the **Description** field, enter @ and the first few letters of the person's first name to see a list of options.
4. Select the name you want.

   If the user doesn't already have permission to view the row and you have the Share privilege for the row, you're asked to provide basic read permission for the row you're adding a note on.

Notifications

1. The notification bell icon indicates a new notification when you're @mentioned in a note.
2. Open the notification center and select the notification to navigate to the record on which you were mentioned.

Feedback

Was this page helpful? Yes No

Provide product feedback

Compare search options in Microsoft Dataverse

Article • 04/05/2025

There are three ways to search rows in Dataverse:

- Dataverse search
- Quick Find (single-table or multi-table)
- Advanced Find

> **Note**
>
> Multi-table Quick Find is also called Categorized Search. For a given query, the search results from the following three options might be different due to the search cope and search behavior.

The following table provides a brief comparison of the three options.

Expand table

Functionality	Dataverse search	Quick Find	Advanced Find
Enabled by default	"On" state for all new production environments. "Default" state for all other environments. Learn more about how Dataverse search works.	Yes, for the table grid. No, for multiple-table quick find (categorized search). An administrator must first disable Dataverse search before multiple-table grid find can be enabled.	Yes
Single-table search scope	Not available in a table grid. You can filter the search results by a table on the results page.	Available in a table grid.	Available in a table grid.
Multi-table search scope	There's no maximum limit on the number of tables you can search.	Searches up to 10 tables, grouped by a table.	Multi-table search not available.

Functionality	Dataverse search	Quick Find	Advanced Find
Search behavior	Finds matches to any word in the search term in any column in the table.	Finds matches to all words in the search term in one column in a table; however, the words can be matched in any order in the column.	Query builder where you can define search criteria for the selected row type. Can also be used to prepare data for export to Office Excel so that you analyze, summarize, or aggregate data, or create PivotTables to view your data from different perspectives.
Searchable columns	Text columns like Single Line of Text, Multiple Lines of Text, Lookups, and Option Sets. Doesn't support searching in columns of Numeric or Date data type.	All searchable columns.	All searchable columns.
Search results	Returns the search results in order of their relevance, in a single list.	For single-table, returns the search results in a table grid. For multi-table, returns the search results grouped by categories, such as accounts, contacts, or leads.	Returns search results of the selected row type with the columns you have specified, in the sort order you have configured.
Wildcards (*)	Trailing wildcard supported for word completion.	Leading wildcard supported. Trailing wildcard added by default.	Leading wildcard supported.

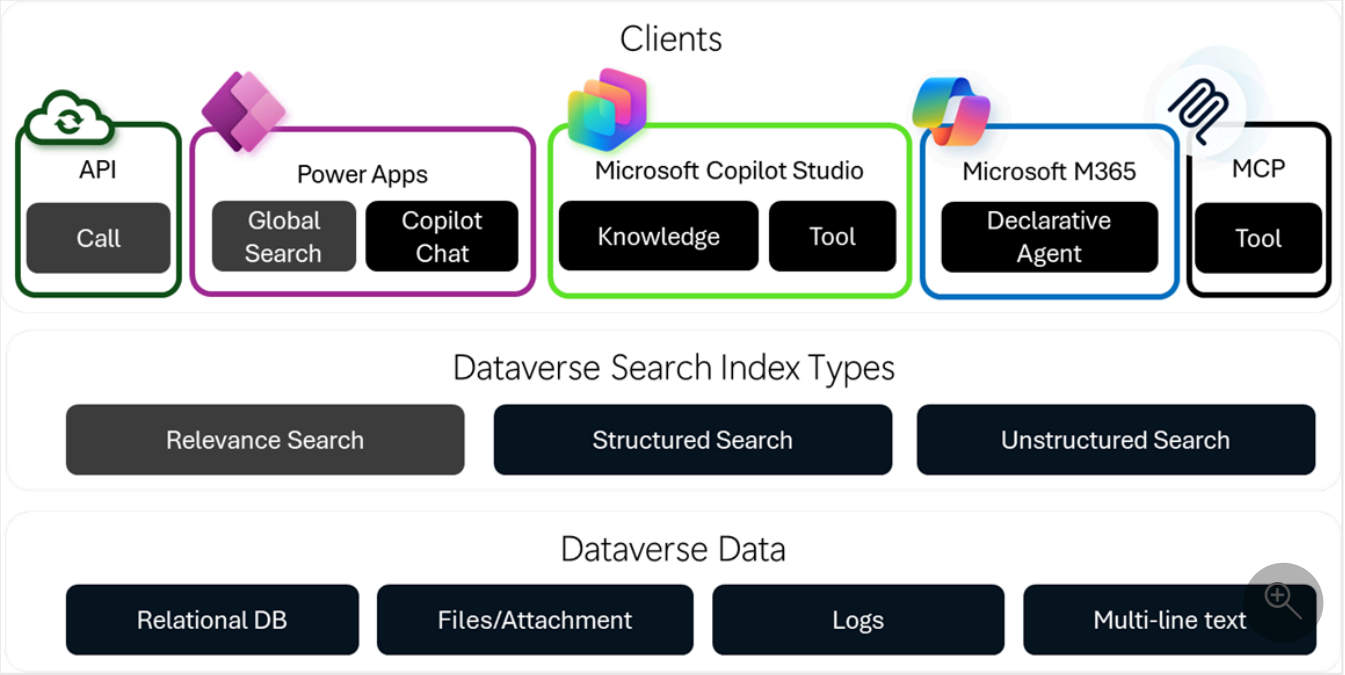
Feedback

Was this page helpful? Yes No

Provide product feedback

What is Dataverse search?

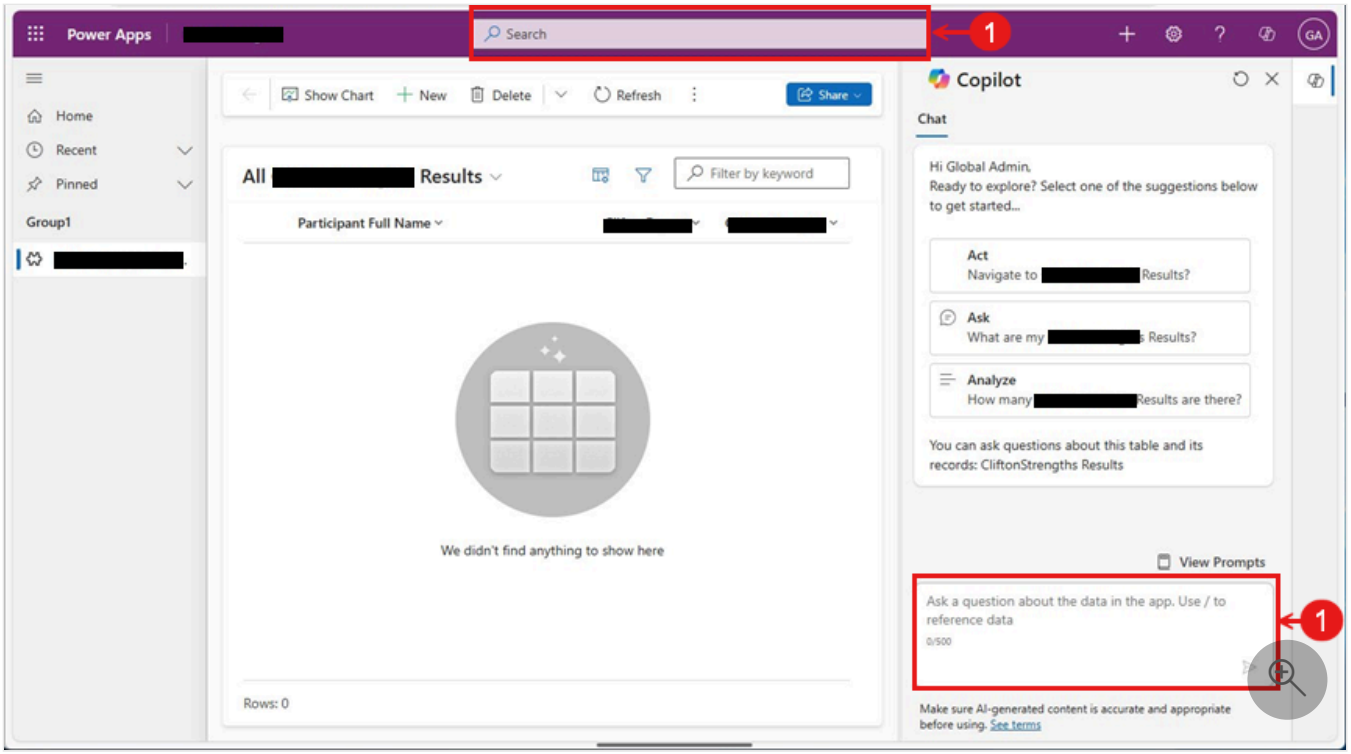
Dataverse search is the foundation that enables Copilot and in-app search experiences to understand and process business data. It turns raw information into actionable intelligence.

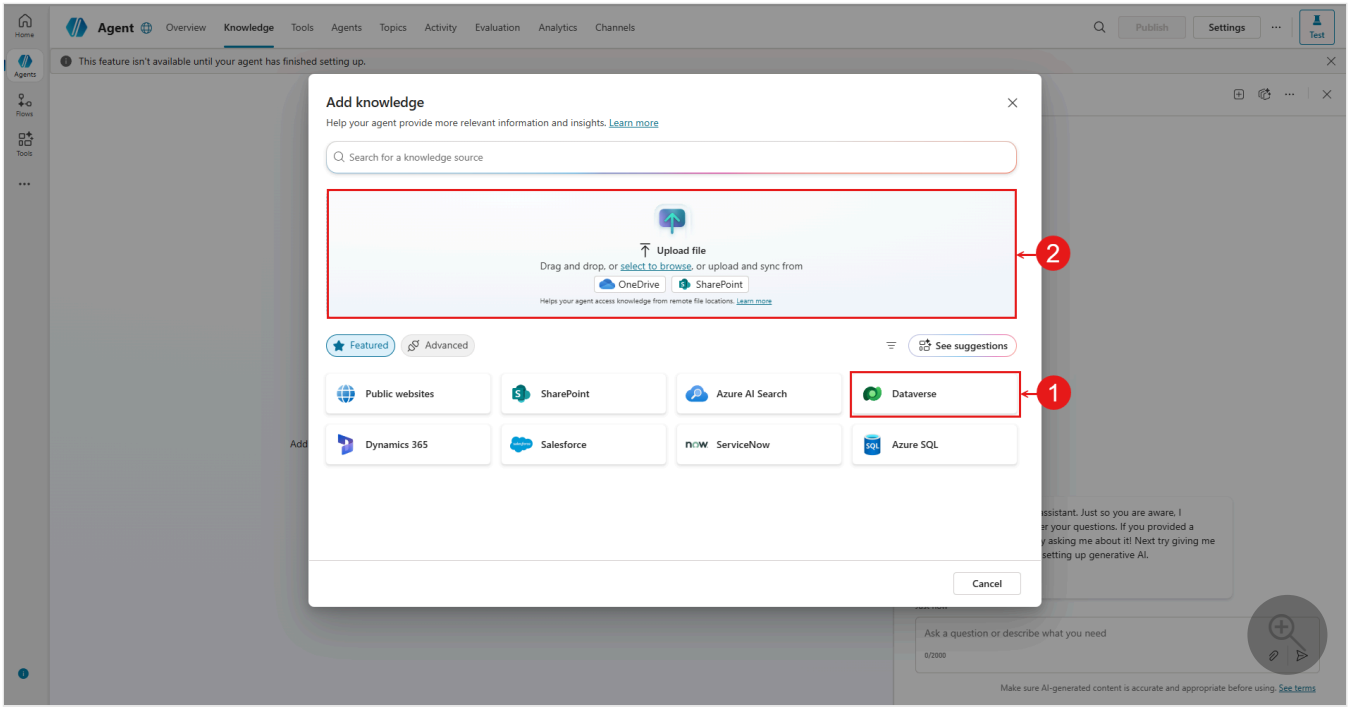


Dataverse search powers global search experiences, API calls, and agents built with Copilot Studio, including those running on Microsoft 365 Copilot and MCP tools. Beyond helping users of model-driven apps quickly, find what they need, Dataverse search enables rich search and AI-driven experiences across products that use Dataverse as a data source.

What type of data does Dataverse search index?

Dataverse search includes structured data (tables) and unstructured data (files).





Legend:

1. **Structured data**: Supports multiple experiences, including global search, Copilot for model-driven apps, and Copilot Studio agents that use Dataverse knowledge from tables.
2. **Unstructured data**: Supports experiences like Copilot Studio agents that use Dataverse knowledge from files.

Dataverse search includes three index types:

- **Relevance search**: Powers global search and SearchQuery API.
- **Structured index**: Enables Copilot and generative AI experiences.
- **Unstructured index**: Enables Copilot and generative AI experiences.

The structured and unstructured indexes enable Copilot and generative AI experiences in scenarios such as agents built with Copilot Studio, Copilot in Dynamics 365, declarative agents, app Copilot, Copilot chat, and Dataverse MCP.

What's the value of indexed data?

Dataverse search uses indexed Dataverse data to help systems perform more efficient searches. Indexed Dataverse data is enterprise content that's organized in a specific structure, like a data catalog. When an agent or system uses the indexed data structure, search performance increases based on a specific input.

Experiences enabled by Dataverse search

Dataverse search lets you use multiple features in Power Platform, including Copilot indexes. Here are the experiences:

Microsoft Copilot Studio Agents

- Add files as a knowledge source
- Add Dataverse as a knowledge source
- Add SharePoint as a knowledge source
- Add OneDrive as a knowledge source
- Agent Copilot

Dynamics 365 Copilots

- Sales Copilot
- Customer Service Copilot
- Contact Center Copilot
- Field Service Copilot

Power Apps: Model-driven apps

- Copilot for app users in model-driven apps
- Dataverse search

Experiences across the Power Platform

- In Power Apps or Power Automate
- In Microsoft Copilot Studio
- Power Apps or Power Automate—custom AI prompts

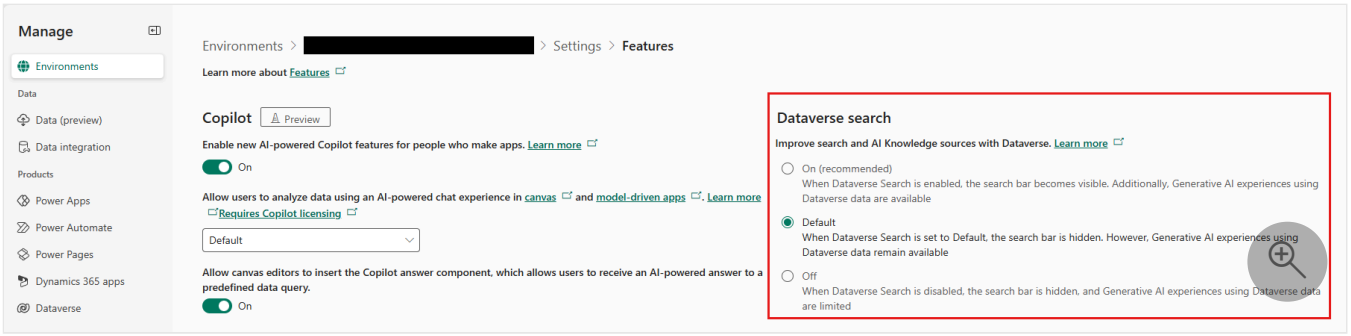
Benefits of Dataverse search

1. **Integration with AI and Copilot experiences:** By using natural language, you can have a conversation with your data and identify themes, patterns, and insights. This feature uses AI technology to interpret natural language, such as misspellings, common abbreviations, and synonyms, to deliver quality results.
2. **Unified search:** Dataverse search lets you quickly find content in model-driven apps and other products that use Dataverse as a data source. It provides a better user experience than quick find, where all words in the search term must be found in one column.
3. **Efficient relevance-based results:** Intelligent ranking algorithms return the most relevant records first, with performance that's superior to categorized search. This feature reduces time spent searching and increases accuracy.
4. **Smart fuzzy search:** Dataverse search handles variations in spelling and terminology, so it doesn't depend on exact keyword matches.
5. **Security and compliance:** Dataverse search respects Dataverse security roles and permissions. Users can only see search results for records that they have access to.
6. **Scalability and performance:** Dataverse search is optimized for large datasets and supports multiple data types, such as Choice and Lookup.
7. **Search across documents in Microsoft Dataverse:** Dataverse search includes search results for text in documents that are stored in Dataverse, such as PDF, Microsoft Office documents, HTML, XML, ZIP, EML, plain text, and JSON file formats. It also searches text in notes and attachments.

> **Note**
>
> Global search supports up to 2 MB of file search.

Dataverse search implications

Dataverse search is an opt-out feature. The feature is **On** for all new production environments and set to **Default** for all other environment types. Find this setting in Power Platform admin center by going to **Manage** > **Environment** > **Settings** > **Product** > **Features** > **Dataverse Search**.



In Power Platform admin center, admins can use the Dataverse Search setting to manage search. The selected Dataverse search state affects how to use Dataverse data across the enabled experiences for the entire organization. The following tables show how each Dataverse search setting affects global search experiences (including SearchQuery API) and Generative AI experiences, and how admins can use this setting to manage it.

What Dataverse search setting means for global search

When you set Dataverse search to **On**, the **Global Search** box appears at the top of every page in your model-driven app. This search box is the default global search experience for all model-driven apps. Learn more in Search for records by using Dataverse search.

Expand table

Dataverse search set to On	Dataverse search set to Default	Dataverse Search set to Off
Search bar is visible. (Quick Find isn't visible or accessible)	Search bar isn't visible. (Quick Find is an alternative)	Search bar isn't visible. (Quick Find is an alternative.)
Dataverse search is used for all production environments	Dataverse search isn't available for Global search in any environment	Dataverse search isn't available for Global search in any environment
Dataverse data is automatically indexed. Data is searchable for global search	Dataverse data isn't indexed. Data isn't searchable for global search	Dataverse data isn't indexed. Data isn't searchable for global search

What Dataverse search means for generative AI enabled experiences

Some generative AI experiences use Dataverse search data, such as Copilot chat in model-driven applications. You can access Copilot chat through the Copilot icon in the right navigation bar in a model-driven app. You can open or minimize the Copilot chat pane as needed.

- **Power Apps**: Copilot chat for model-driven apps in Power Apps is a next-generation AI assistant that helps app users get insights about the data in their apps through conversation in natural language.
- **Copilot Studio**: Integrating Dataverse tables as your knowledge source allows you to ground your agent in the data contained in your tables. This integration can also happen by using unstructured data as knowledge or Dataverse tools.

> **Note**
>
> An enabled generative AI feature's availability isn't managed by Dataverse search itself and is enabled through its own feature setting.

Expand table

Dataverse search set to On	**Dataverse search set to Default**	**Dataverse search set to Off**
All production environments use Dataverse search	Sandbox, Trial, Developer, Dataverse for Teams environments use Dataverse search for generative AI experiences	No environment uses Dataverse search
Dataverse data is automatically indexed and becomes searchable	Dataverse data (tables and files) is indexed when triggered and becomes searchable	Dataverse data isn't indexed nor searchable

What actions can makers or admins take to manage Dataverse search efficiently?

Depending on the experience that uses Dataverse search and its usage, consumption size can increase. The following recommendations help you optimize storage consumption for efficient and value-added output, from granular and less impactful actions (data level) to broader and more impactful actions (environment level).

> **Note**
>
> While one action helps optimize the amount of indexed data within a specific feature, other enabled features might trigger index provisioning. For more impactful approaches, go through all the steps, in order of recommendation, or consider turning off features in the Power Platform admin center.

Data level

To manage the search index at the data level, review each table and each column to confirm that search is needed. If not, take the following actions:

Global Search: Break down tables into multiple columns

Why this helps?

Indexed data includes structures like inverted indexes, metadata, and pointers to enable fast search and retrieval. These components add overhead beyond the raw data and can significantly increase storage requirements compared to the original dataset.

How to implement?

- Ensure the data is consistently formatted across records to increase repetitiveness of words and reduce the number of indexes.
- By breaking down the data across multiple columns, you can be more specific about which data to search over.

Global Search: Only select columns that need to be searchable

Why this helps?

Dataverse search has default selected columns. By default, certain columns like Primary Name and ID are indexed for all tables. These columns are part of the 50 fields indexed by default and don't count toward the limit for every table. Enable columns for Dataverse search only if a Quick Find view is set as the default view for the table. For more information, see Managing Dataverse search.

How to unselect tables that aren't used so that they're not indexed or searchable

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**.
3. Choose the solution you want to modify, and then select **Edit** in the command bar.
4. On the **Objects** page, in the navigation pane, select **Overview**.
5. In the Dataverse search pane, select **Manage search index**.
6. Review the tables that are indexed.
7. Unselect the tables, and then select **Save**.

How to unselect columns that aren't used so that they're not indexed or searchable

1. Sign in to Power Apps.
2. On the left navigation pane, select **Tables**.
3. On the **Tables** page, select the table you want to edit.
4. In the **Data experiences** pane, select **Views**.
5. From the list of views, select the Quick Find view type. For example, select **Quick Find Active Accounts**.
6. Edit **View columns**.
7. Go to the columns you want to remove, and select **Remove** in the dropdown menu.
8. Select **Save and Publish** to publish the changes to the view.

Copilot Studio: Ensure specific Dataverse tables or files are added to Copilot Studio agent's knowledge

Why this helps?

When you add a Dataverse tool or knowledge, such as Dataverse MCP, Dataverse table, or file to a Copilot Studio agent, the process automatically indexes all the underlying Dataverse data for efficient, semantically relevant searches. Reducing knowledge and tools to only the necessary content helps indexing processing and storage consumption and increases search quality.

How to remove unnecessary content or imagery, or tables from files

- **File Upload, OneDrive, and SharePoint upload**: Remove files that aren't needed for search from the agent's knowledge base. You might want to consider removing pages, sheets, or any accessory data point from the files themselves.
- **OneDrive and SharePoint upload**: Select only the files or folders you need instead of selecting nested folders.

How to remove Dataverse tables or files from Copilot studio agent’s knowledge

In the **Overview** or **Knowledge** tabs on a Copilot Studio agent, go to the **Knowledge** section and select the dot menu for each of the Dataverse-enabled knowledge sources, and then select **Delete**. Use the same instructions for the following knowledge sources:

- Dataverse Tables
- Files you upload

- OneDrive files
- SharePoint files (from upload)

> **Note**
>
> The same process applies to Dataverse MCP tools, but you complete it in the Tools section (instead of the Knowledge section).

App or agent level

To manage the search index at the app or agent level, review the purpose and usage of each application and agent to confirm search is needed. If not, take the following actions:

Copilot for Power Apps: Specify which applications you'd like Copilot to be enabled for

Why this helps?

Copilot in model-driven apps uses AI to interpret natural language queries and generates suggestions or actions based on the app's underlying Dataverse data. In some situations, applications don't require enhanced search through Copilot, so managing Copilot at an app level can help control consumption.

How to turn off Copilot feature for each model-driven application

1. In the app designer, open the model-driven app for editing.
2. Select **Settings** from the command bar.
3. On the **Settings** screen, select **Upcoming**.
4. Select the **Copilot control** setting, and set it to **Off**.
5. Select **Save and Publish** to apply your changes.

Copilot for Dynamics 365 Apps: Specify which applications you'd like Copilot to be enabled for

Why this helps?

Just like model-driven applications in Power Apps, Copilot in model-driven apps uses AI to interpret natural language queries and generate suggestions or actions based on the app's

underlying Dataverse data. In some situations, applications don't require that level of enhanced search through Copilot. Managing Copilot at an app level can help with consumption control.

How to turn off Copilot feature for each Dynamics 365 application

1. In the Sales Hub app, change the area by using the option in the lower-left corner of the page.
2. Select **App Settings** to open sales setting.
3. Select **Copilot** > **Edit settings** > **Advanced settings**, which opens a new Copilot in Dynamics 365 Sales page.
4. Select **Individual apps**.
5. Select desired app and turn the setting to **Off**.

How to turn off Copilot feature for all Dynamics 365 application

1. In the Sales Hub app, change the area by using the option in the lower-left corner of the page.
2. Select **App Settings** to open sales setting.
3. Select **Copilot** > **Edit settings** > **Advanced settings**, which opens a new Copilot in Dynamics 365 Sales page.
4. Select **All Apps** and turn the setting to **Off**.

The other way to turn off Copilot feature for all apps is as follows:

1. In the Sales Hub app, change the area by using the option in the lower-left corner of the page.
2. Select **App Settings** to open sales settings.
3. Select **Copilot** > **Edit settings**.
4. Select **Off** from the Copilot dropdown, which opens a dialog box.
5. Select **Continue**, and then save your changes.

Environment level

To manage the search index at the environment level, admins can review the purpose and usage of each application and agent, as well as global search to confirm that search is needed. If not, admins can take the following actions:

Global search is disabled while Copilot experiences are enabled

Why this helps?

When Dataverse search is set to **Default**, the global search experience isn't enabled on the navigation of the model-driven apps or Dynamics 365. This setting means that users can't query global search, and data marked as "searchable" for the exclusive purpose of global search isn't indexed. However, Dataverse data might still be indexed for other experiences, such as Copilot in model-driven applications or Copilot Studio agents.

How to change Dataverse search to Default value

1. In Power Platform admin center, go to **Manage** > **Environments**.
2. Select the environment, and then select **Settings**.
3. Select **Product** > **Features**.
4. Change **Dataverse search** to **Default**.
5. Select **Save**.

> **Note**
>
> For organizations that want to search in these applications and don't need relevance-based search, they can enable Quick Find.

Copilot for Power Apps and Dynamics 365 apps is disabled while global search is anabled

Why this helps?

Copilot in Power Apps uses Dataverse indexed data to quickly retrieve and interpret relevant records when responding to natural language queries within the model-driven app. The semantic indexes enable efficient search across tables and relationships, allowing Copilot to provide accurate suggestions and automate actions based on the underlying data context. For organizations that don't want Copilot to be used with their model-driven applications, admins can turn off the feature at an environment level.

How to turn off the Copilot setting in Power Platform admin center

1. In Power Platform admin center, go to **Manage** > **Environments**.
2. Select the environment, and then select **Settings**.
3. Select **Product** > **Features**.
4. Turn the toggle and the set the dropdown to **Off**:

Dataverse search isn't needed for the environment

Why this helps?

At any time, you can manually set Dataverse to **Off**. If you set Dataverse search to "Off" for the environment, all users of features, apps, or agents of the environment can't use the search capability in the Power Apps navigation bar or any generative AI experience that relies on Dataverse, like uploaded files or using OneDrive or SharePoint files (via upload) in Microsoft Copilot Studio agents, among other experiences.

How to turn off Dataverse search for the environment

1. In Power Platform admin center, go to **Manage** > **Environments**.
2. Select the environment, and then select **Settings**.
3. Select **Product** > **Features**.
4. Change **Dataverse search** to **Off**. This change opens a dialog box. Acknowledge the impact of turning off Dataverse search, write the name of the environment, and select **Turn Off**:
5. Select **Save**.

> **Note**
>
> - Turning off Dataverse search deprovisions and removes the index within a period of 12 hours. If you turn on Dataverse search after it's been off for 12 hours, it provisions a fresh index that needs to go through a full sync. Syncing might take up to an hour or more for average size organizations, and a couple of days for large organizations. Be sure to consider these implications when you turn off Dataverse search temporarily. Configure Dataverse search for your environment
> - Index removal (or provisioning) can take multiple days to be completed, depending on the amount of Dataverse search consumption. For example, an organization with 10 GB of indexed data might take one day to clean up all indexes, while an organization with 500 GB might take multiple days to see it reflected in Dataverse search reporting. Give a few days or a week before submitting a support ticket, to ensure a complete removal of Dataverse search indexed data.

What happens when you turn off Dataverse search?

Expand table

Feature	Maker experience	End user experience
Microsoft Copilot Studio Agent – Add Knowledge	You can't upload files or select Dataverse tables. The agent doesn't provide results that rely on this indexed data until Dataverse search is enabled for the environment. Ask the environment's admin to enable Dataverse search.	The agent doesn't provide results that rely on this indexed data until Dataverse search is enabled for the environment. The agent uses the fallback answer by default.
Microsoft Copilot Studio Agent – Using Copilot Chat	The agent doesn't provide results that rely on this indexed data until Dataverse search is enabled for the environment. Ask your environment admin to enable Dataverse search.	The agent doesn't provide results that rely on this indexed data until Dataverse search is enabled for the environment. By default, the agent uses a fallback answer.
Model Driven Applications – Dataverse search	The search bar isn't visible in model-driven apps.	The search bar isn't visible in model-driven apps.

See also

Search for tables and rows by using Dataverse search
Configure facets and filters
Frequently asked questions about Dataverse search
Developer's guide: Search for Dataverse records using the API
Compare search options in Microsoft Dataverse

Last updated on 01/13/2026

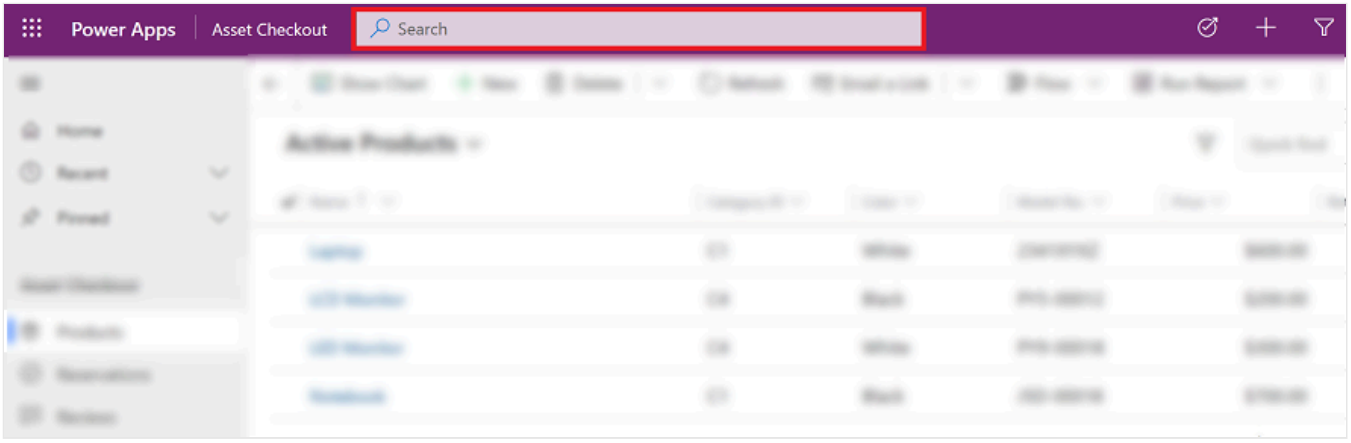
Search for records by using Dataverse search

06/26/2025

When Dataverse search is "**On**", the search box appears at the top of every page in your app and is the default global search experience for all model-driven apps. Dataverse search also lets you use certain generative AI experiences. You can't switch to quick find search, also known as categorized search. Start a new search to quickly find the information you need.

Turn on Dataverse search to give users a better search experience in model-driven apps. Environment admins can manage their environments by selecting **Off** to opt out of this feature.

[!Note] If Dataverse search is set to "**Default**", the search box isn't visible for searching content. Generative AI experiences are still available.

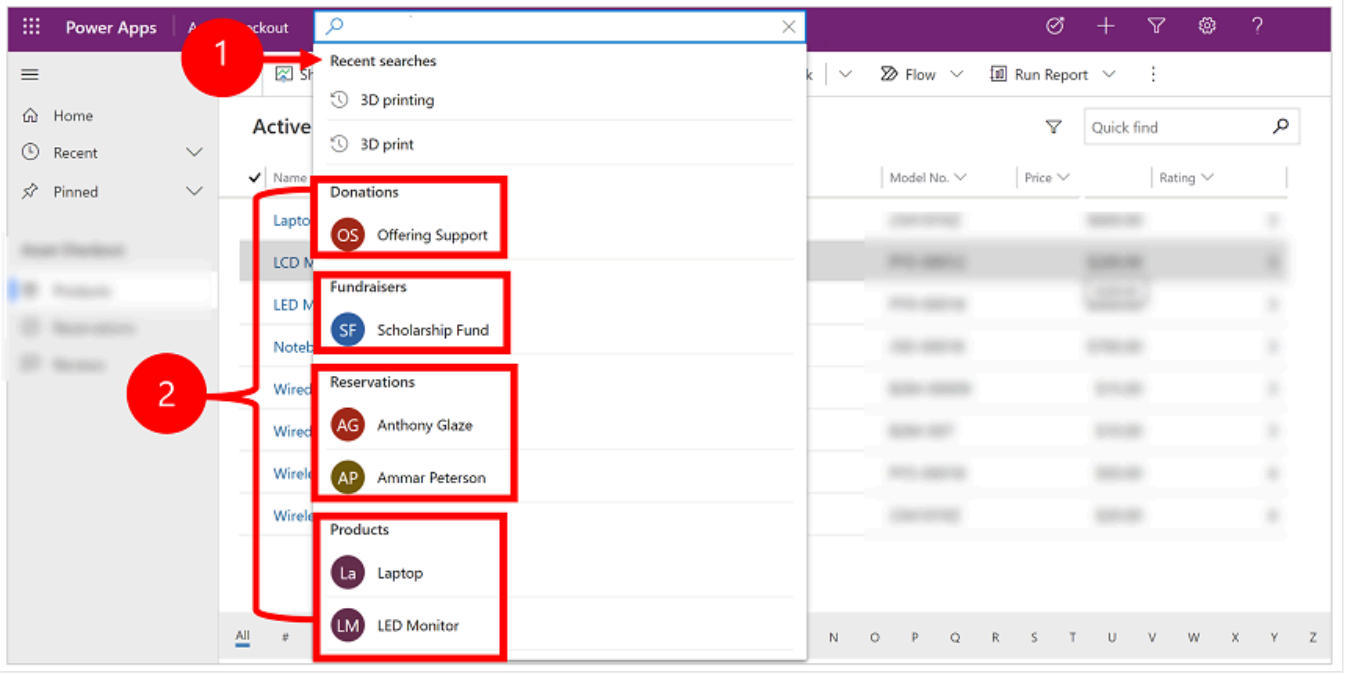


See recent rows and searches

See your recent searches and recently accessed rows when you select the search box. Before you start typing in the search box, you'll see information pop-up that helps you complete your search.

Up to three recent search terms appear at the top, based on the last three search queries that you viewed the results for. These recent search terms are personalized and based on your device and browser.

The recent search terms are followed by up to seven of the rows you recently accessed. If you frequently access a few rows, you can quickly get to them from here. Recently accessed rows are independent of tables that have been indexed for Dataverse search, because no search has been performed yet. The rows are also grouped by table type, which lets you quickly scan the list.



Legend

1. **Recent searches**: Shows your recent searches.
2. **Recently accessed rows**: Shows rows that you recently accessed, grouped by table type.

Inline suggestions

> **Note**
>
> If you're using custom analyzers for searching records, the records don't show in the inline suggestions.

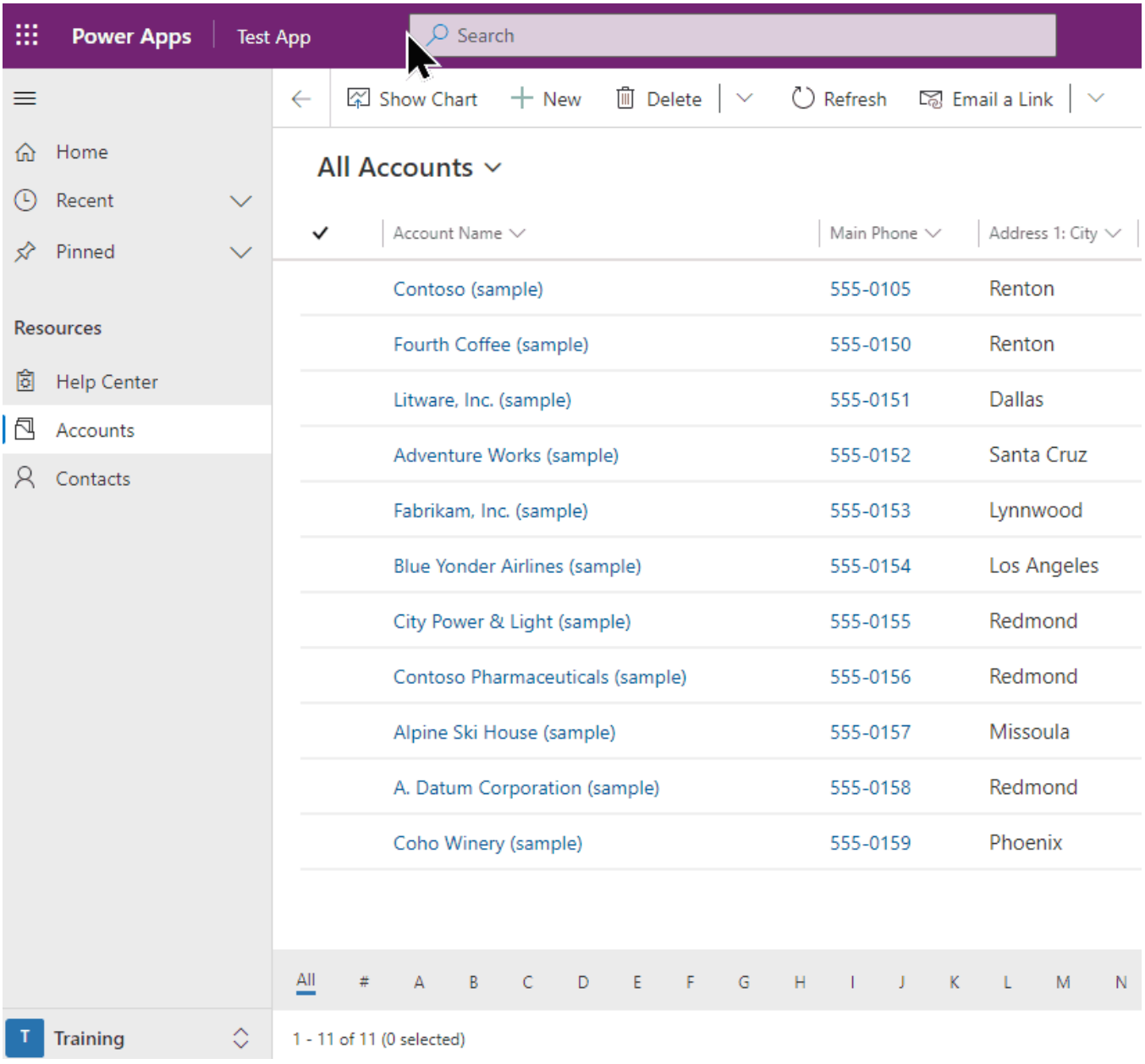
As you start entering text, you'll see suggested results that help minimize keystrokes and simplify page navigation. Suggested results are based on a search performed on all searchable columns of a table, which is enabled for Dataverse search and included in your model-driven app. For more information, see Add or edit model-driven app components in the Power Apps app designer.

Suggestions are shown when three or more characters are entered in the search box, and are based on two types of matching:

- **Word completion**: Rows where the primary field contains a word that begins with the search term. For example, entering **work** will show the account **AdventureWorks** and the contact **John Worker**, among other results.
- **Fuzzy search**: Terms that are misspelled by one character are matched. For example, entering **winry** will show the account **Coho Winery**, among other results.

Suggested rows display a secondary field alongside the primary field to help differentiate rows with the same primary field value. The secondary field is selected based on the table's quick find view:

- If the search text appears in a field within the quick find view, that field is used as the secondary field.
- If no match is found, the first non-primary field in the quick find view that contains data for the row is displayed instead.



With suggestions, you can access your information quickly by using minimal keystrokes even when the search term is misspelled by a character. Text that's highlighted in bold in the suggestions shows the term that was matched.

Search results page

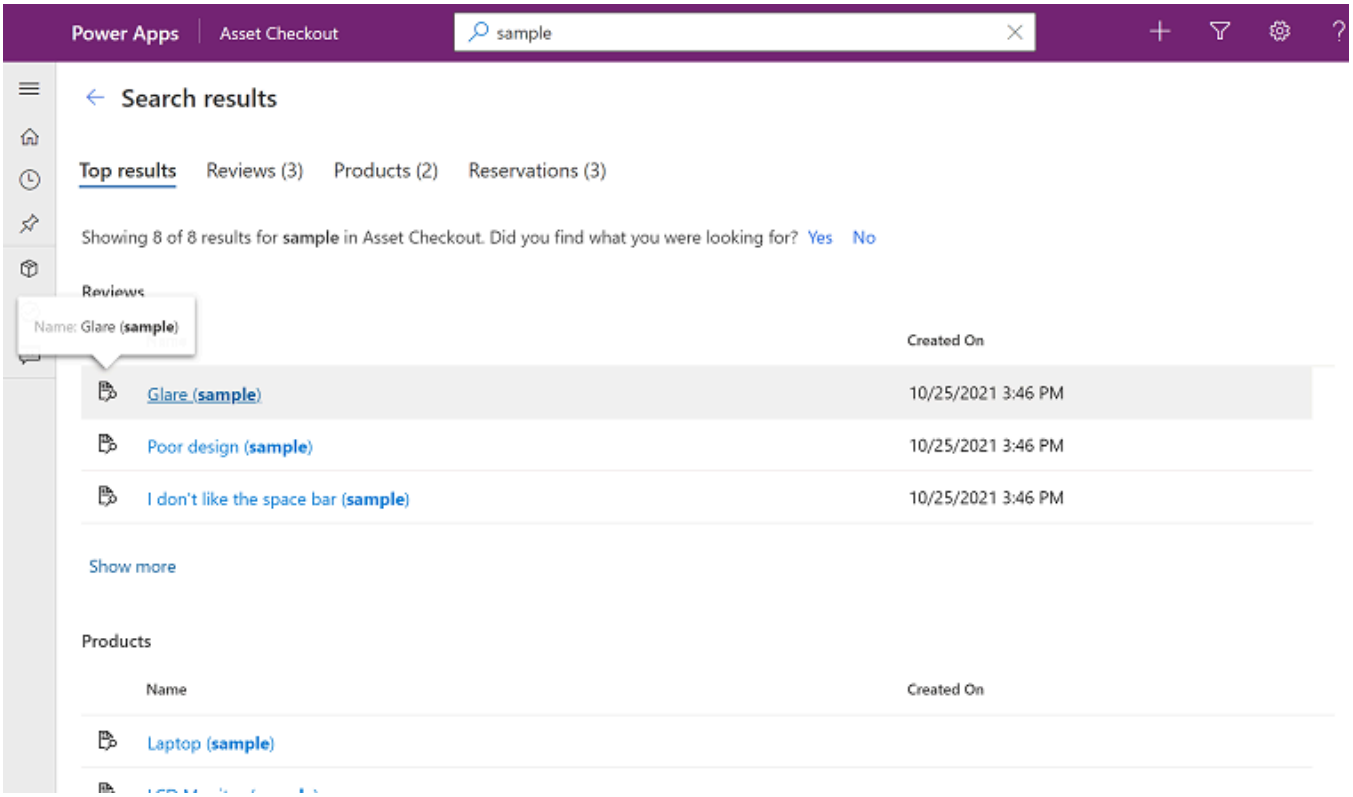
View the full results for a search by pressing **Enter** on your keyboard or selecting **Show more results**. Dataverse search will append "*" to the search if two or less characters are entered.

Search results are ranked based on relevance and grouped by tables. The list of tables that include rows that match the search term are displayed horizontally as tabs along the top of the screen.

When a search query contains multiple words, it's automatically split into individual search terms. Additionally, our fuzzy search feature expands results by including related words such as synonyms, acronyms, and abbreviations. For example, searching for "Robert Clark" might return results for "Bob Clark" due to fuzzy matching and "John Clark" because one of the search terms matches. To ensure an exact match, enclose the entire query in double quotes (for example, "Robert Clark").

To understand why a row appears in your search results, hover over the icon at the beginning of each row to see the text that matched your search.

> **Note**
>
> Rows returned in the search grid can't be sorted.



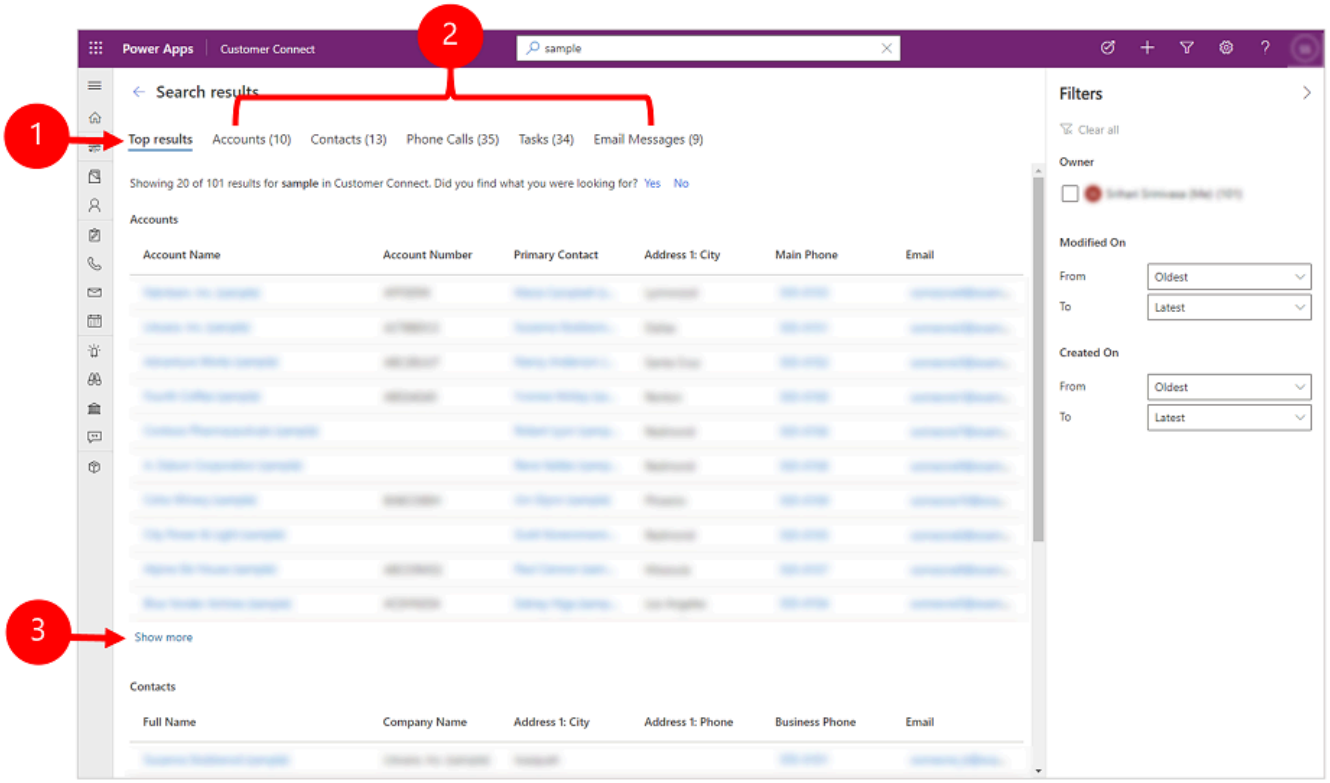
Top results tab

The **Top results** tab displays the top 20 results for the search term, with results grouped by table type. Each group shows results for a table in a grid that has up to six columns. The

columns are the first six columns of the table's **View columns** quick find view.

> **Note**
>
> - The primary column of a table is always used as the first column in the results on the **Top results** tab.
> - For notes and attachment tables, two additional columns show information about the related row for that note or attachment.
> - Party list columns on activity tables—such as **To**, **CC**, and attendees—can't be searched on or shown, and will be blank.

In the following image, the top search results from three tables are displayed.



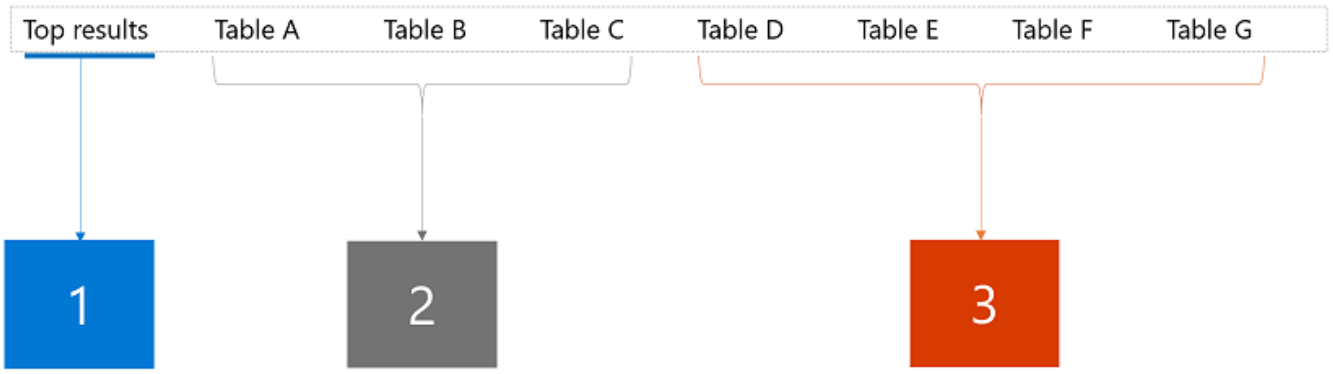
Legend:

1. **Top results**: This tab shows the top 20 results for your search term.
2. **Rows grouped by table type**: To narrow your search results to a specific table, select the table-specific tab.
3. **Show more**: When you select **Show more** at the bottom of a group, you switch to the table-specific tab.

Table-specific tabs

Tabs for the specific tables that contain search results are displayed horizontally. The exact order of the tables from left to right (in a left-to-right environment) is based on the relevance of the results. You can collapse the filter panel or hover over the list of tabs to scroll horizontally.

The tables that include the rows in the top 20 results are shown in the first three tabs from left to right, from most to least relevant. The tables that include the results outside of the top 20 are displayed in descending order.



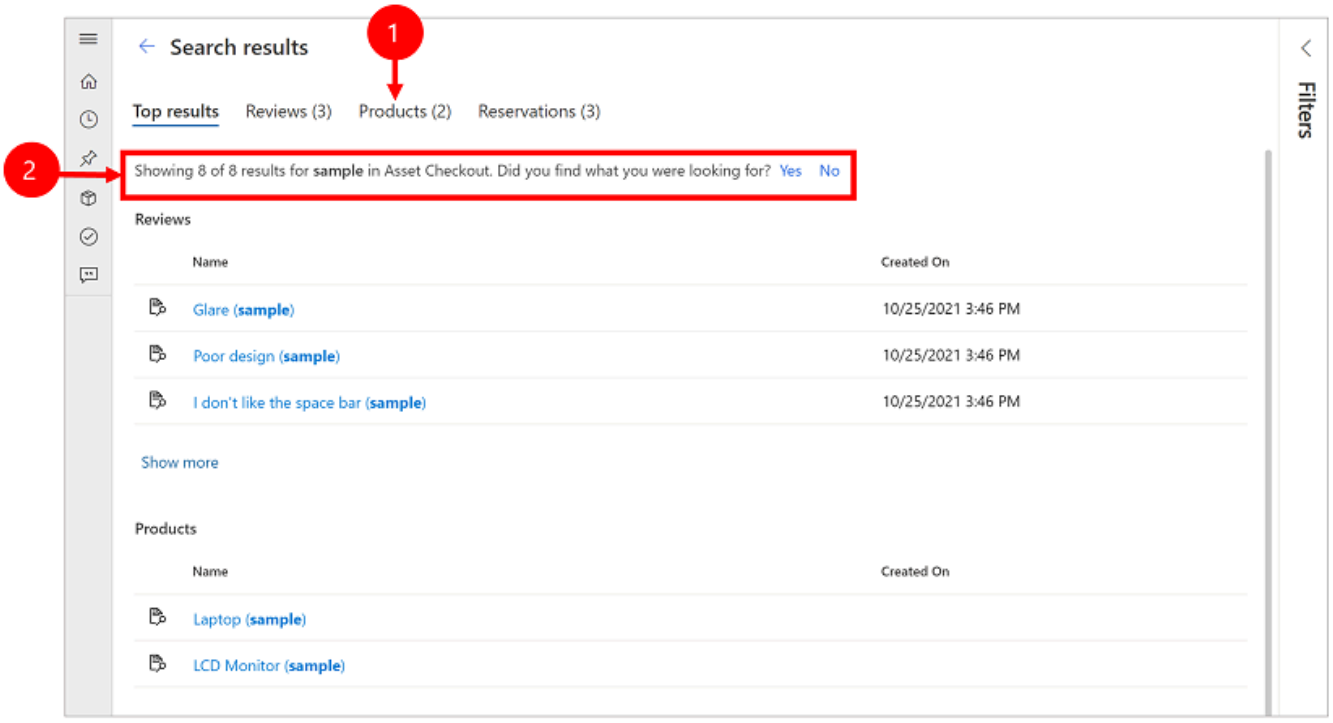
Legend:

1 Shows the top 20 results from three different tables.

2 Shows the three tables that are part of the top results, ordered by relevance.

3 Shows the rest of the matched table types, ordered by the number of records that match the search.

Select a tab to drill down into a specific table and view more information about the rows in the results.



Legend:

1. Shows the number of results for this specific table.
2. Shows the columns that were searched for the table.

Each table-specific tab displays more information than the **Top results** tab along two different dimensions:

- If the quick find view for the table has more than six **View Column**, then all columns are displayed on the table-specific tab, compared to the maximum of six columns shown on the **Top results** tab.
- All matching rows for the table are accessible in the table-specific tab as an infinitely scrollable list.

Filter results by using facets

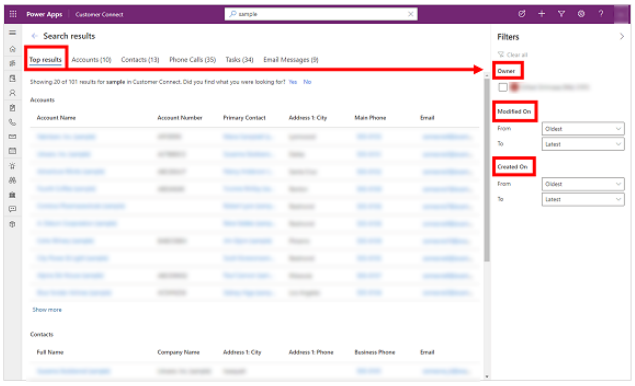
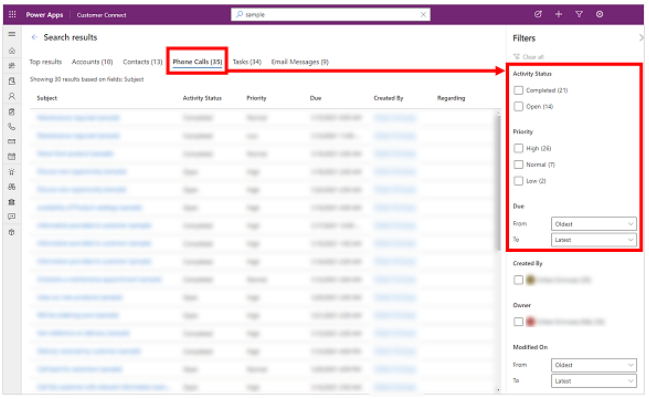
On the search results page, you can use facets and filters to refine and explore your search results. Facets are available on the **Filter** panel, which you can expand or collapse.

The **Top results** tab shows search results across different row types, such as a contact versus a phone call. Thus, the **Top results** tab only shows the three global facets that apply to all row types:

- Owner
- Modified On
- Created On

On a table-specific tab, facets that apply to that table type are displayed above the three global facets.

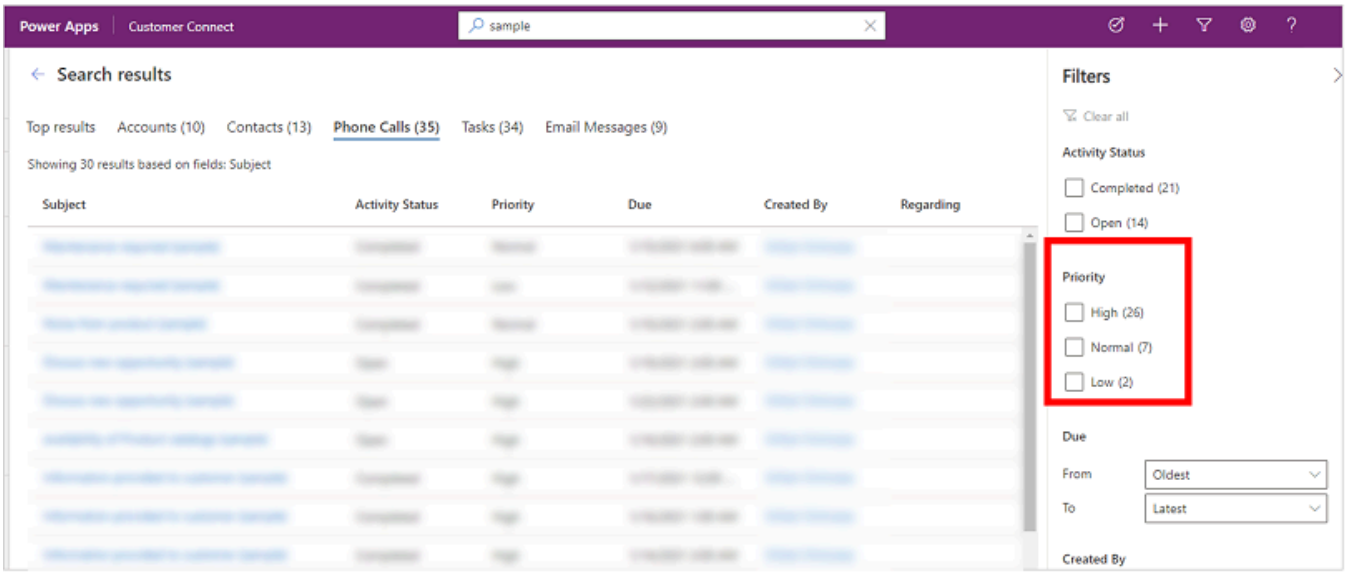
Expand table

The Top results tab shows only global facets	Other tabs show facets that are specific to that table type
	

You can personalize facets for a table by using the **Set Personal Options** dialog box. More information: Configure facets and filters

Text-based facets with discrete values

All **Lookup** and **Choice** data types are text-based facets. For example, the text-based facet **Priority** consists of a list of column values (choices) and their corresponding counts.

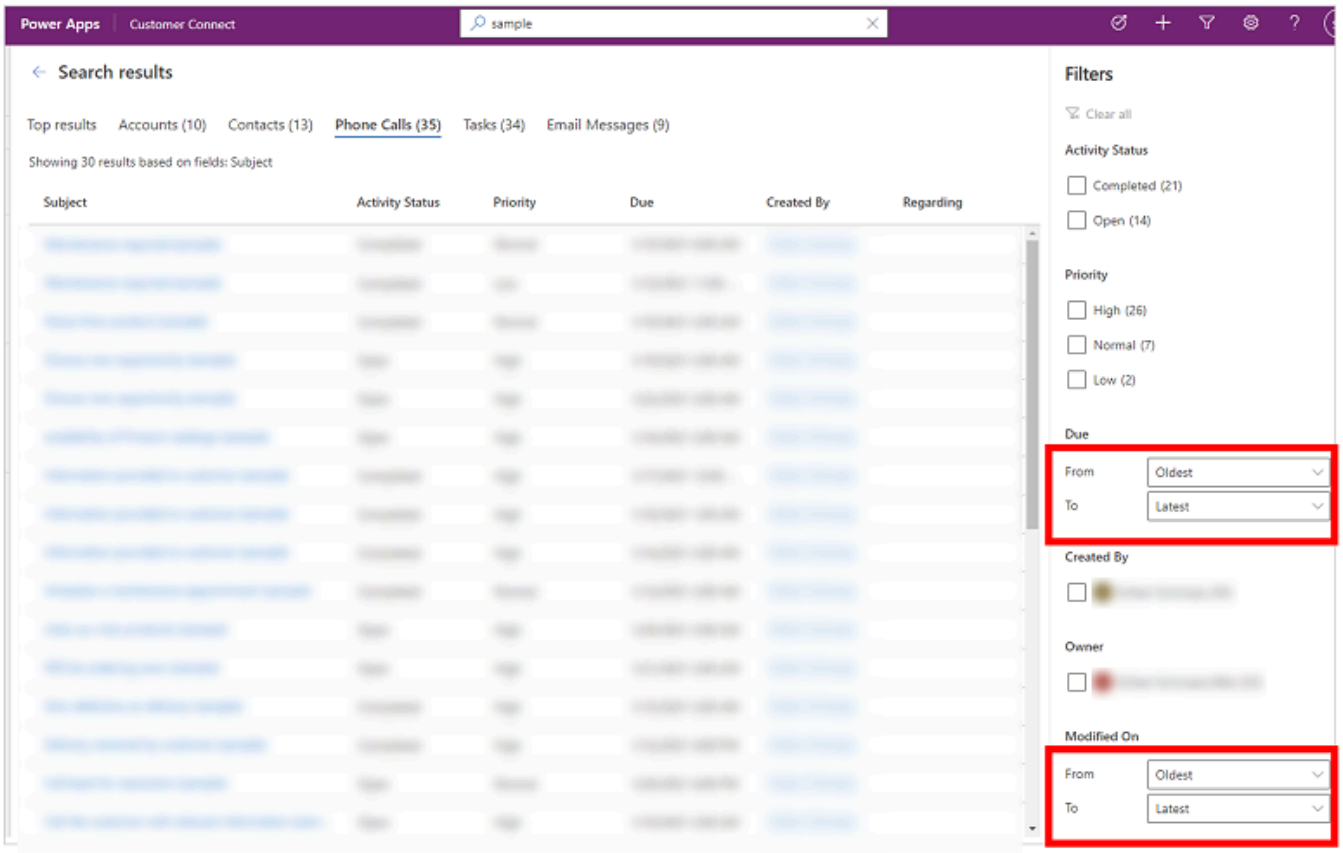


Filters in these facets are sorted in descending order by count. By default, the top four-facet values are displayed. When there are more than four-facet values, you can select **Show more** to expand the list and see up to 15 results.

When you select a facet value, you filter search results to show only rows where the column includes the value that you selected.

Date and time facets

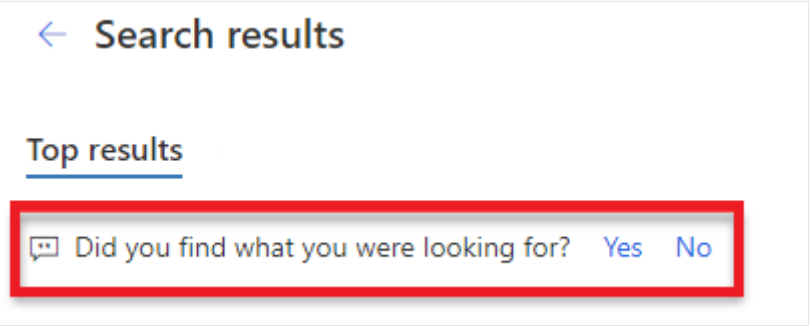
The date and time facets let you filter and see search results for a specific period. Use the **From** and **To** drop-down lists to specify a time period.



Feedback link

On the search results page, the answer to **Did you find what you were looking for?** is collected in our product telemetry as a binary feedback. Search parameters—like the query text that you entered in the search box—isn't collected, irrespective of your response to the question. We only use **Yes** or **No** response statistics to help us understand the usefulness of the search experience.

Currently there's no option to disable the feedback question prompt.



Understand search results

Dataverse search has a powerful set of capabilities to help you put Microsoft Dataverse at your fingertips. Here are some pointers to help you understand what results you can expect in different scenarios.

> **Note**
>
> - Dataverse search is text-based, and can search only on columns of type **Single line of text**, **Multiple lines of text**, **Choice**, or **Lookup**. Columns of type **Numeric** and **Date** aren't searched on.
> - Dataverse search enables you to search for rows that are shared with you and rows that you own. Hierarchical security models aren't supported. This means that even if you see a row in Dataverse because you have access to it through hierarchical security, you won't see that row matched by Dataverse search unless you own it or it was shared with you.
> - Dataverse search supports column level security. This means that a user with access to a restricted column can search for the column and will see it in the search results. A user that doesn't have permission to the restricted column won't be able to search or see results for the column.

> **Tip**
>
> The order of tables in the **Top results** tab and in the horizontal list of tabs is based on the ranking and relevance of search results for that search term. You can make results from a particular table appear at the top by including the table name in the search term. For example, searching for **account fabrikam** would, in most cases, rank result records that have the term **fabrikam** of type **account** higher than result records that have the term **fabrikam** of type other than **account**.

- **Boosted search results**
  Use important attributes as keywords to "boost" your search for information. Entering a table type to boost search results is an easy way to indicate which results you prefer. For example, searching for **Susan Adatum** indicates that you're looking for a contact **Susan** related to the account **Adatum**.

- **Natural language understanding**
  Search has a better understanding of terms that are used to indicated intent. Some examples include:
  - Searching for **open phone calls** will show phone call activity records that have the status **open**.
  - Searching for **contact missing email** will show contacts who have an empty email address field, rather than treating **missing** as a keyword.

- **Spelling correction**
  Search forgives common spelling mistakes. For example, when you search for **william conatc**, search results will include the contact record with the name **William**, even though the word **contact** was misspelled.

- **Synonyms, acronyms, and abbreviations**
  With support for common abbreviations and synonyms, you can search for your information the way you remember it, not necessarily the way it's stored. Some examples include:
  - Common name variations like **Bob/Robert** are matched with either search term.
  - A search term that uses an abbreviation like **active accts** will be interpreted semantically as "active accounts."

- **The power of common knowledge**
  Search can understand popular locations, date, time, holidays, and numbers. For example, a search for **account in WA** will generate results for accounts located in Washington state.

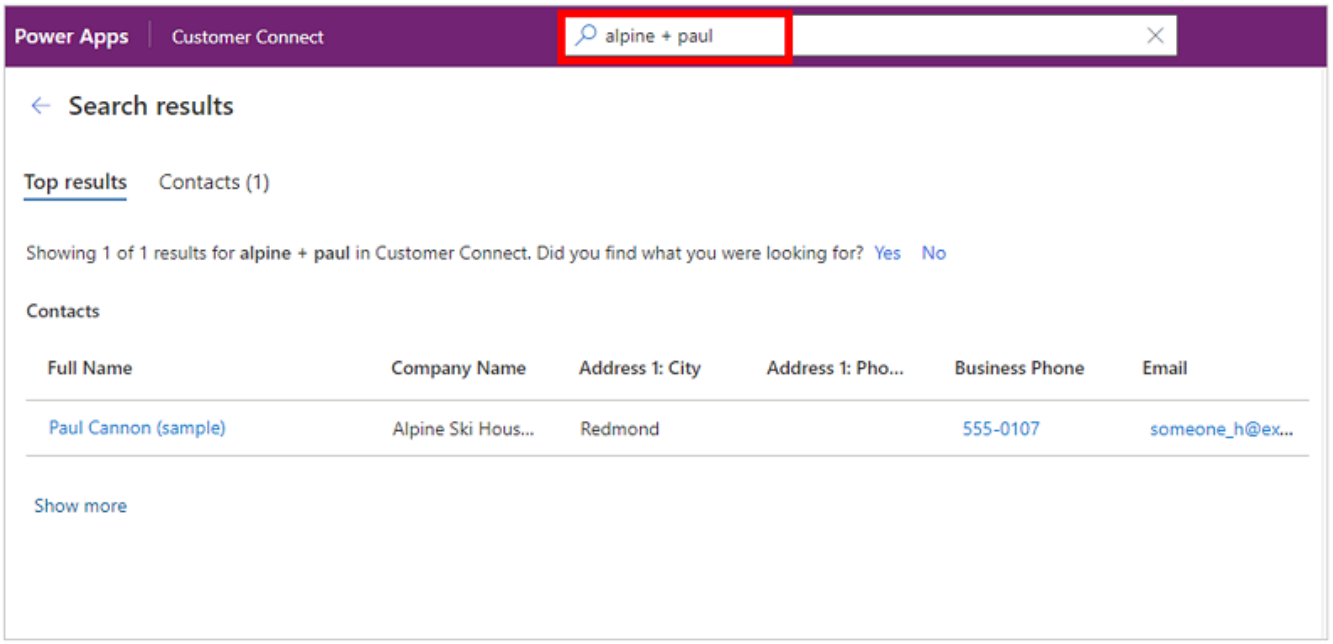
> ⓘ **Note**
>
> - All the preceding capabilities are part of intelligent search, which is only available for public cloud environments that have English as the base language.
> - To get these results, the preceding examples assume that specific fields for each table were indexed.

Working with operators

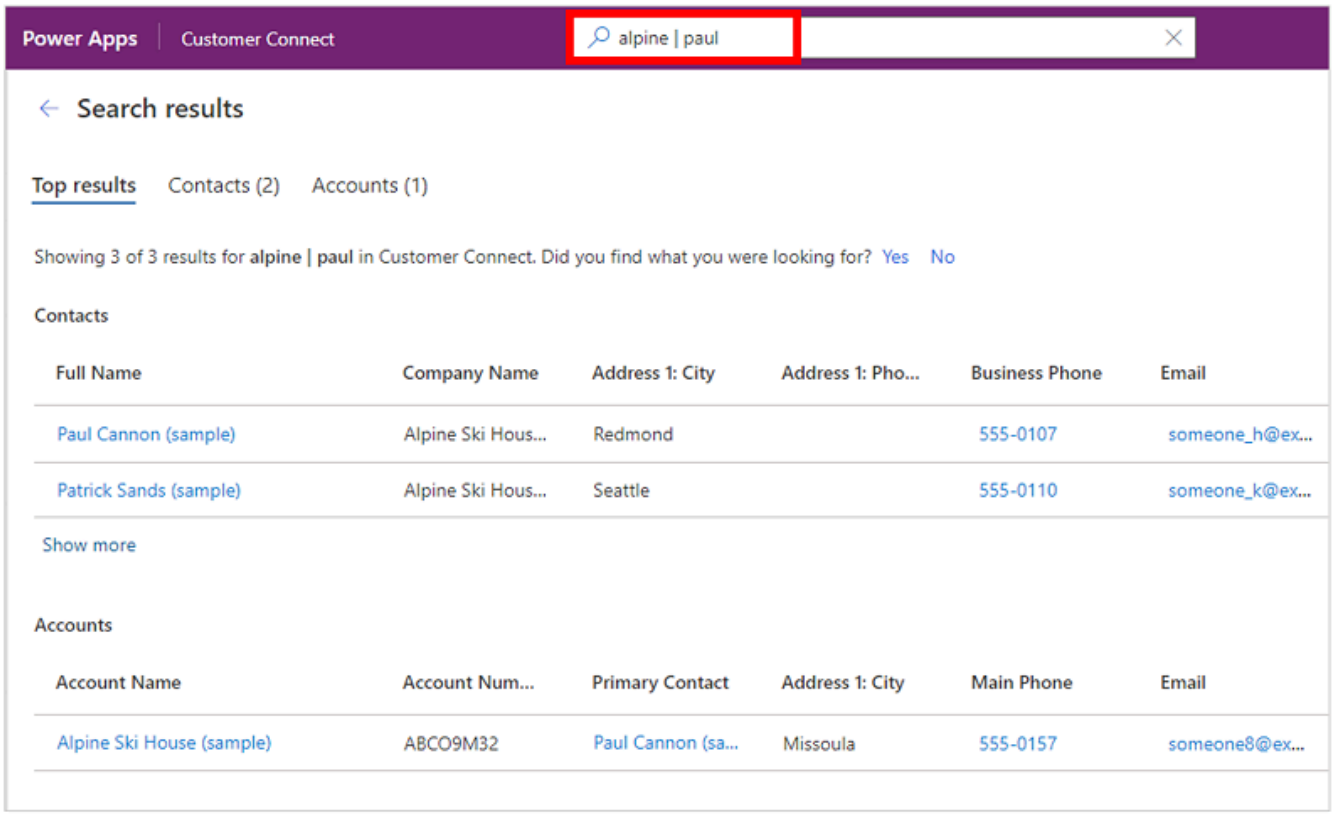
Operators help you craft precise search queries that can tailor your results according to specific conditions. This section describes some of the supported operators.

Boolean operators

Search on a combination of terms using the + operator, which performs an **AND** search and shows results containing all the individual terms separated by +. For example, **alpine + paul** returns the contact record **Paul Cannon** related to the account record **Alpine Ski House**.

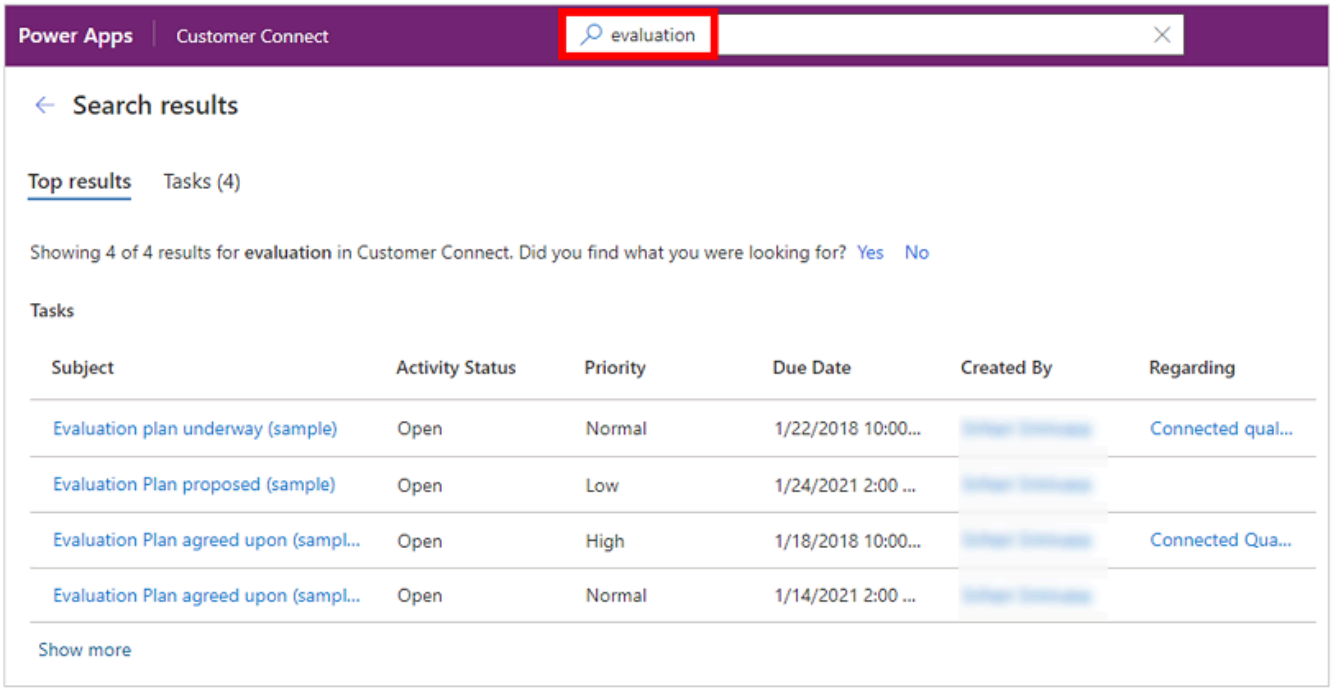


Similarly, the **OR** operator can be indicated by separating terms with a vertical bar (|). For example, searching for **alpine | paul** returns records with reference to either term.

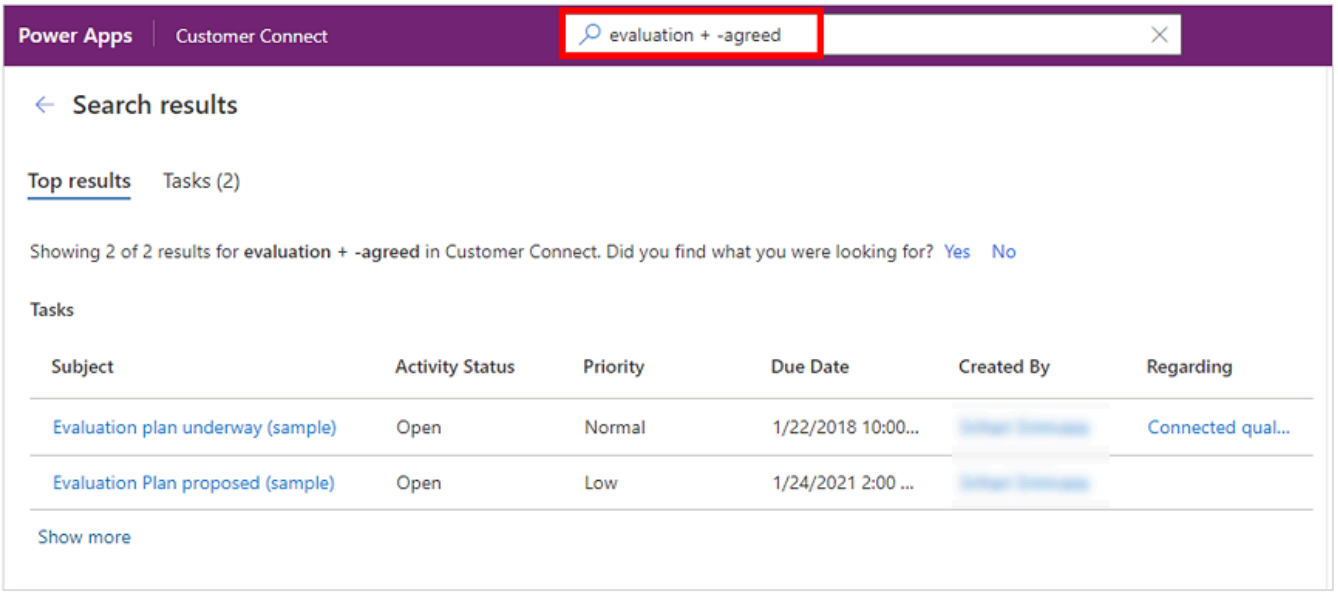


Use the **NOT** operator with a hyphen (-) before a keyword to indicate that it should be excluded from results.

For example, when you search for **evaluation**, all rows matching the term **evaluation** will appear in results.

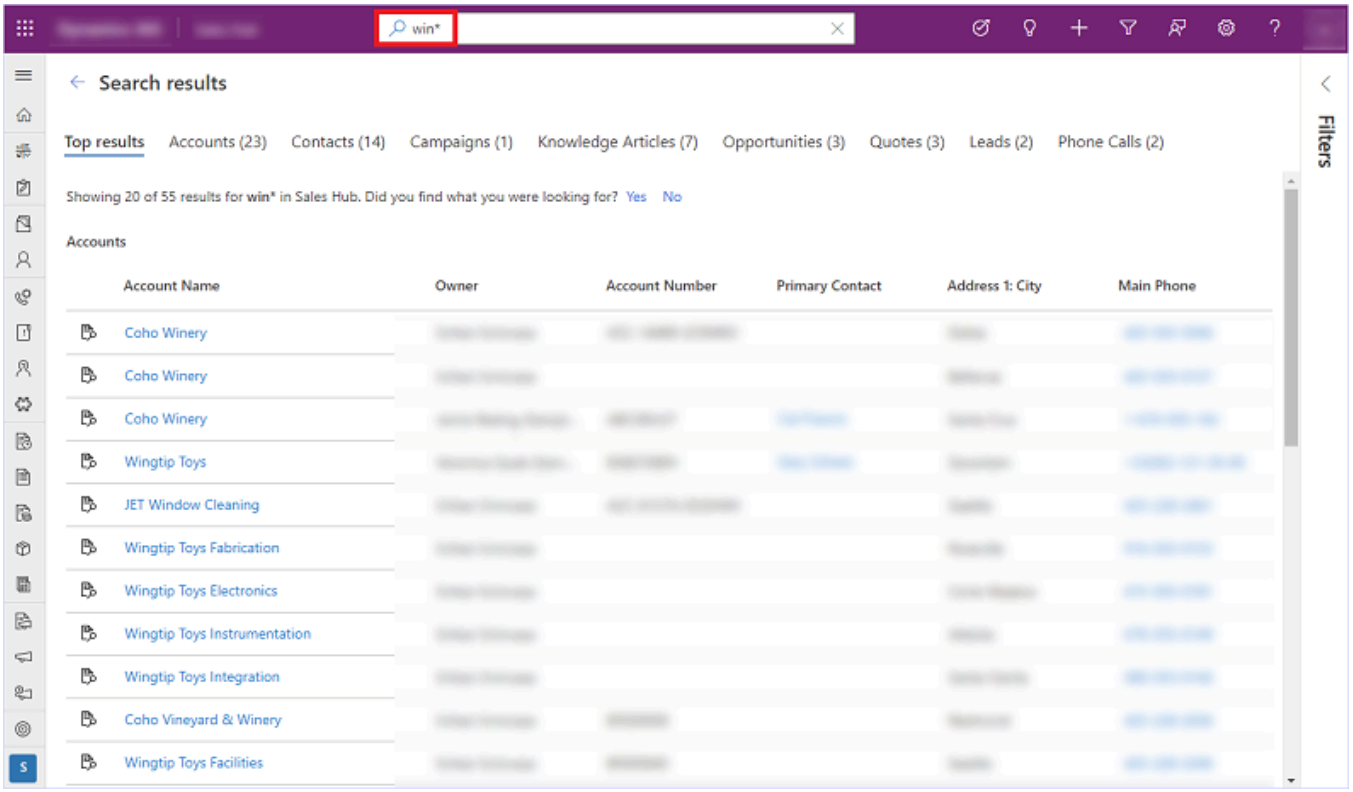


However, searching for **evaluation + -agreed** will match all rows that contain the term **evaluation** but not the term **agreed**.



Wildcards

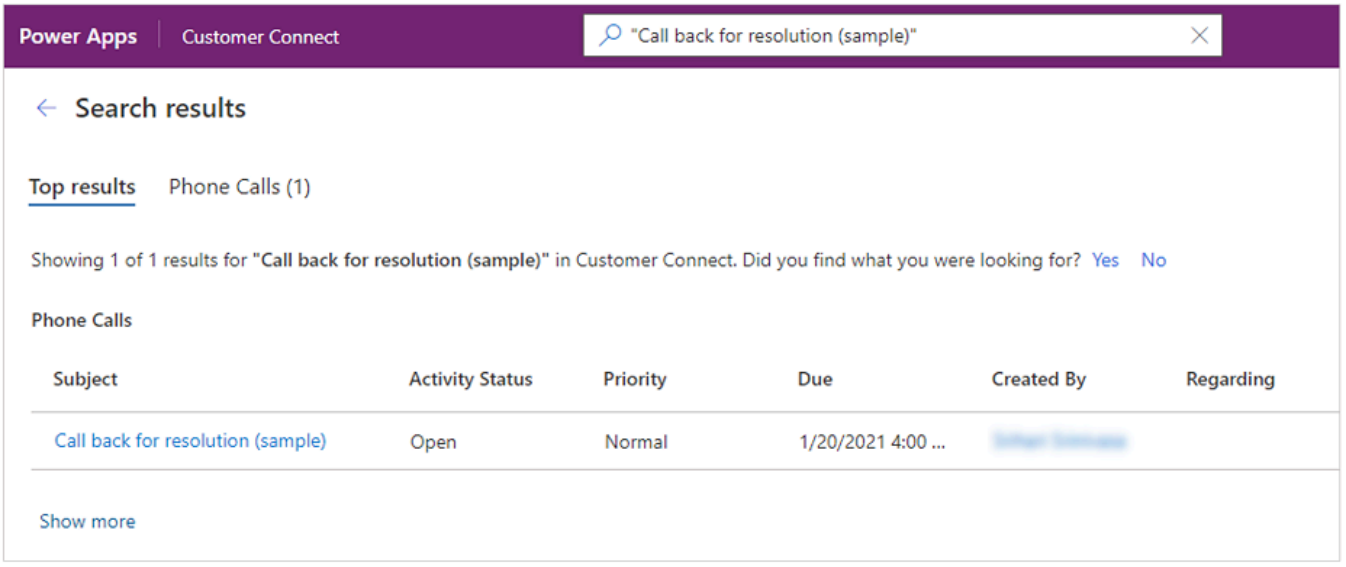
Use wildcards for operators such as **begins-with** and **ends-with**. Using an asterisk (*) at the beginning or end of a keyword makes the asterisk a placeholder for zero or more characters. For example, searching on ***win** will show results for all records associated with a keyword that ends with **win**.



Exact matches

Use double quotation marks around a keyword to perform an exact match. Search ignores common used words such as **the**, **an**, and **a**. To improve matching, use double quotation marks

and force an exact match for your search. For example, searching on **"Call back for resolution (sample)"** will match the exact keywords.



Quick actions

The Dataverse search experience brings some of the most frequently used actions closer to search results, to help you complete your task without having to navigate to the record page in model-driven apps.

Quick actions are a small set of commands specific to a table. You can see quick actions when you're interacting with search in model-driven apps running on a web browser. Some of the commonly used tables are configured to show a set of commands to help you complete your task without losing context.

Expand table

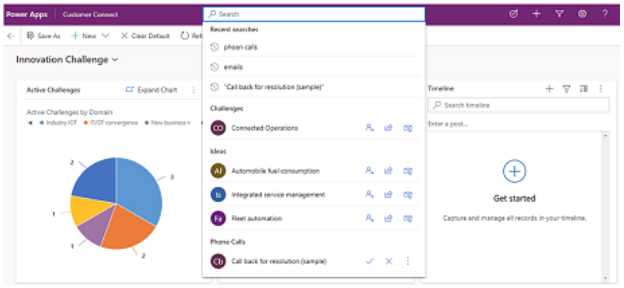
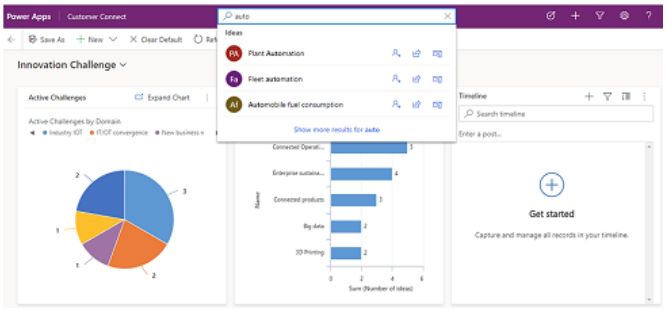
Table	Quick actions
Account	Assign, Share, Email a link
Contact	Assign, Share, Email a link
Appointment	Mark complete, Cancel, Set Regarding, Assign, Email a link
Task	Mark complete, Cancel, Set Regarding, Assign, Email a link
Phone Call	Mark complete, Cancel, Set Regarding, Assign, Email a link
Email	Cancel, Set Regarding, Email a link

For more information on how to configure quick actions, see Configure Dataverse search to improve search results and performance.

Quick actions and suggested results

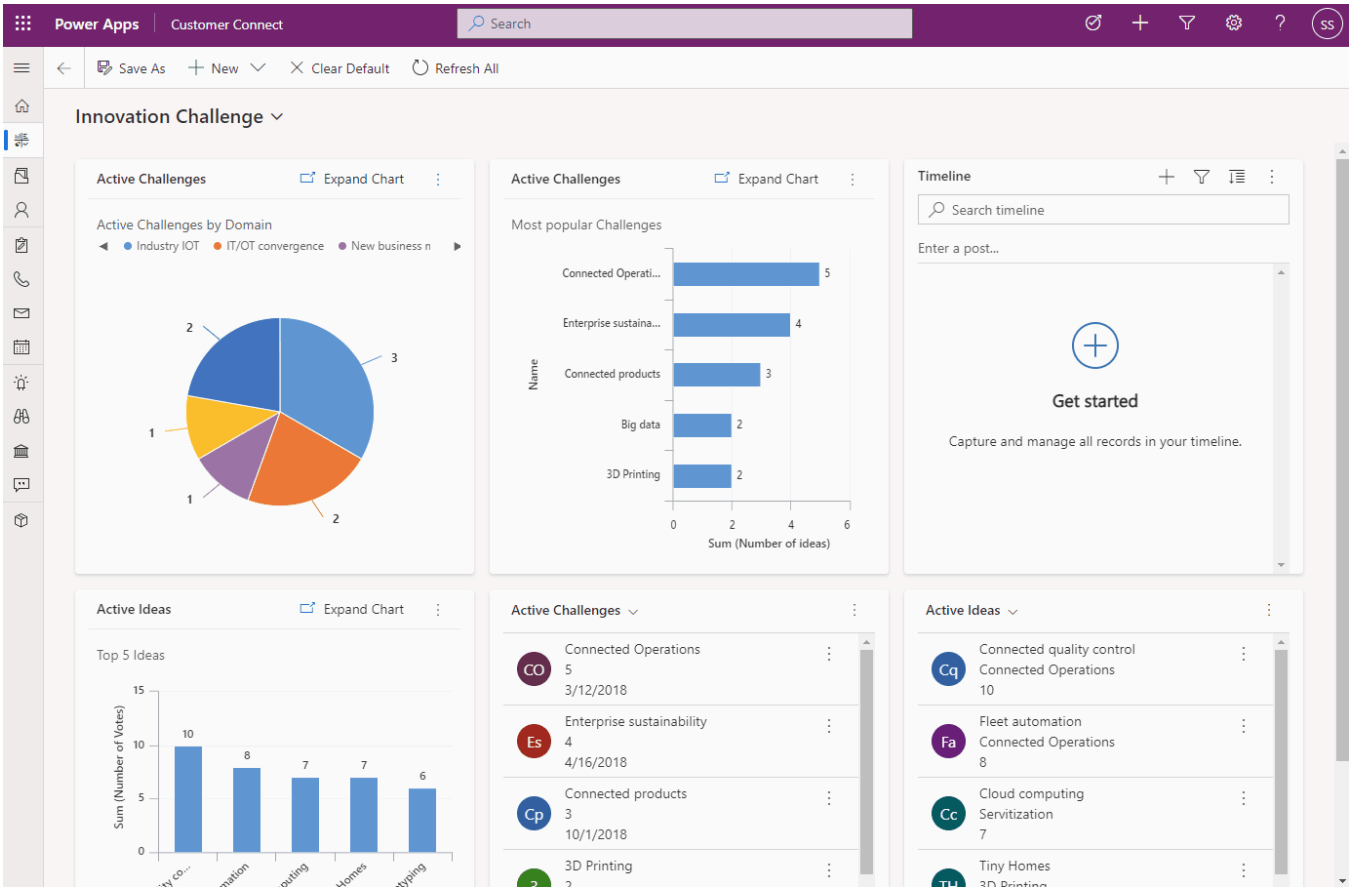
As you interact with search, you'll find quick actions next to the most recently accessed records and the suggested results that appear as you type.

Expand table

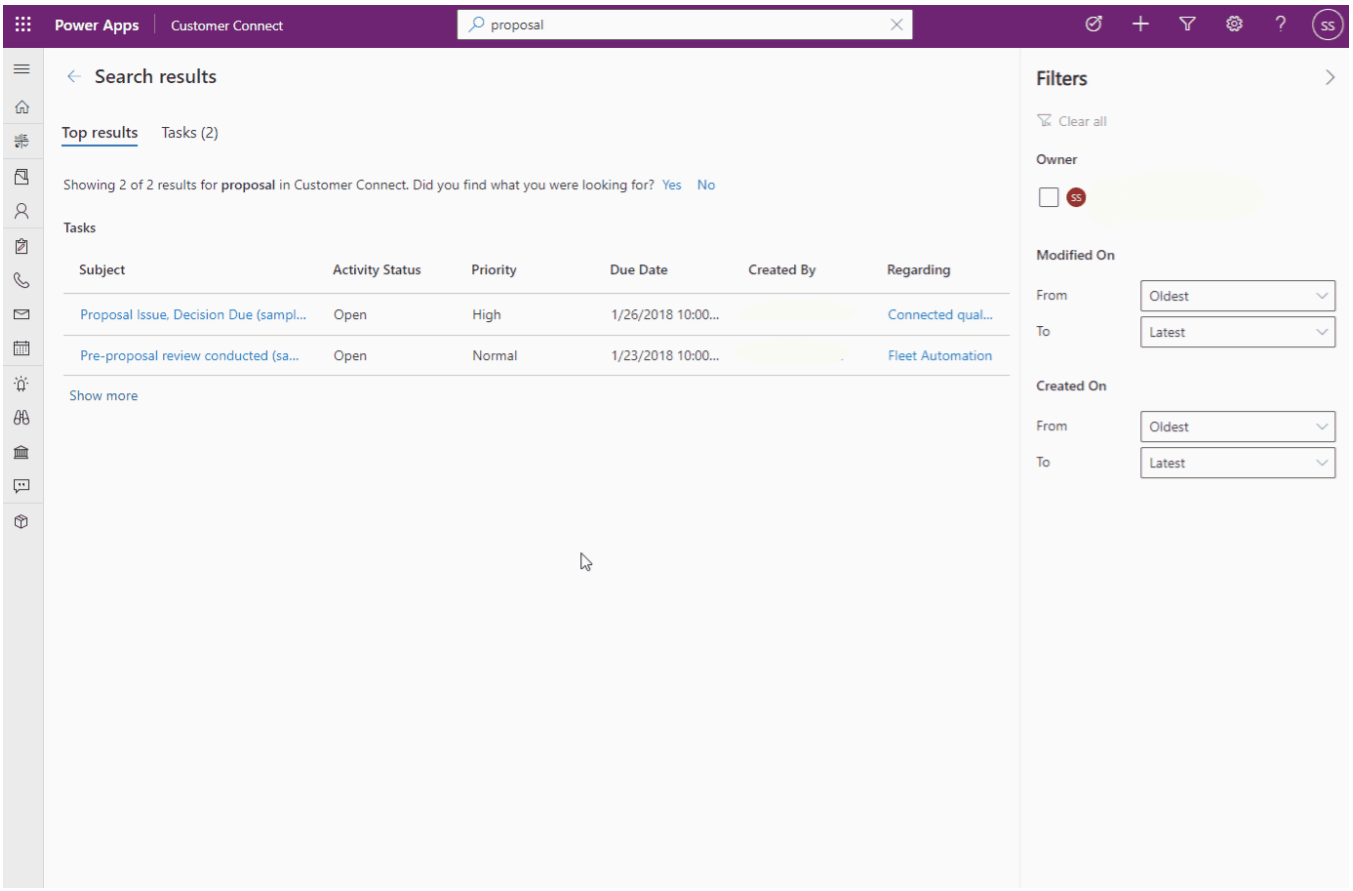
Quick actions for recently accessed rows	Quick actions for suggested results as you type
	

Quick actions in results pages

As you type, quick actions will appear for suggested results.



On the results page, hover over the results to see quick actions available for the row.



Use keyboard shortcuts

The following keyboard shortcuts apply to grids. For more information, see:

- Navigation
- Editing
- Selection

See also

What is Dataverse search?
Configure facets and filters
Frequently asked questions about Dataverse search

Configure facets and filters

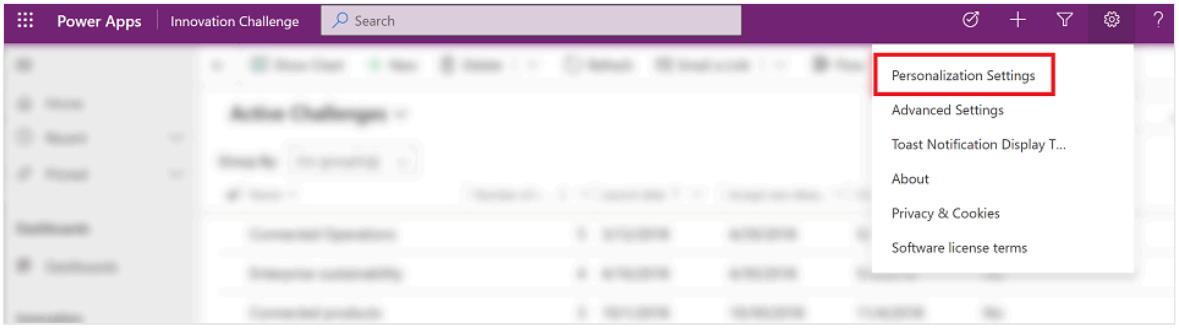
Article • 02/15/2022

With Microsoft Dataverse, you can refine your search results by using facets and filters. Facets and filters let you drill into and explore the results of your current search without having to repeatedly refine your search terms.

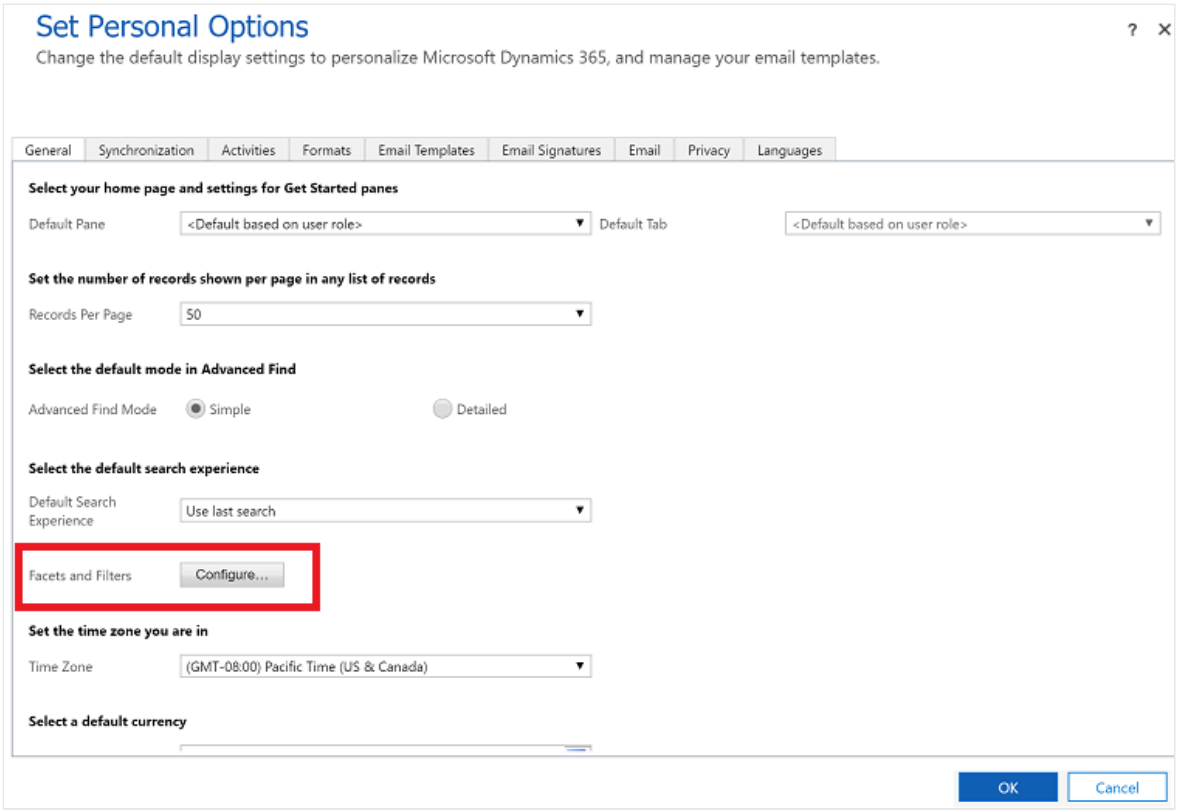
In the **Set Personal Options** dialog box, you can personalize the facets for a table.

> **Note**
>
> Your admin can use the quick find view to define which fields appear as default facets when you use Dataverse search. The first four view columns that include data types other than **Single line of text** or **Multiple lines of text** are displayed as default facets in the result. You can override this setting in your personalization settings. At any time, you can select up to four fields as facets.

1. In the upper-right corner, select **Settings**, and then select **Personalization Settings**.



2. On the **General** tab, in the **Select the default search experience** section, for **Facets and Filters**, select **Configure**.



3. In the **Configure Facets and Filters** dialog box, specify the facets you'd like to see for a table:

    - In the **Select Table** drop-down list, select the table you want to configure facets for. This drop-down list contains only the tables that are enabled for Dataverse search.

    - For the selected table, select up to four facet columns. By default, the first four "facet-able" columns in the quick find view for the selected table are selected in the list. At any time, you can only have four columns selected as facets.

You can update multiple tables at a time. When you select **OK**, the changes for all tables that you've configured are saved. To revert to the default behavior for a table that you previously configured, select **Default**.

> ⓘ **Note**
>
> If a system customizer deletes a column or makes it no longer searchable, and you've saved a facet for that column, it will no longer show up as a facet. You'll only see the columns that exist in the default solution and have been configured as searchable by your system customizer.

See also

What is Dataverse search?
Search for tables and rows by using Dataverse search
Frequently asked questions about Dataverse search

Feedback

Was this page helpful? Yes No

Provide product feedback

Frequently asked questions about Dataverse search

What is Dataverse search?

Dataverse search is the foundation that enables Copilot and in-app search experiences to understand and process business data. It helps organizations turn raw information into meaningful insights. For more information about Dataverse search and its enabled experiences, see What is Dataverse search.

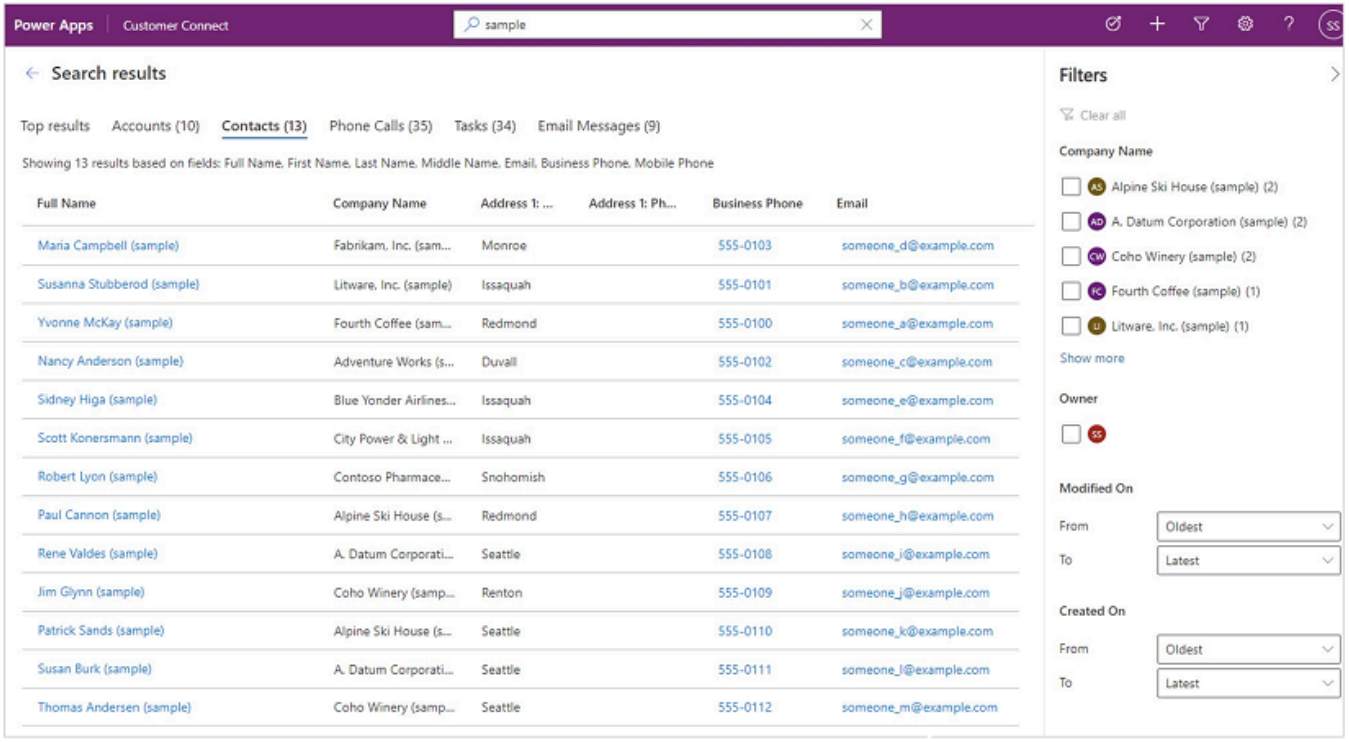
What is the difference between "On" and "Default" state for Dataverse search?

The state you select for Dataverse search affects the ability to use Dataverse data across the enabled experiences for your organization. For a better understanding of what each Dataverse search state means, see:

- What Dataverse search setting means for global search
- What Dataverse search means for generative AI enabled experiences

What is the scope of content searched by Dataverse search using Global Search?

Any file or Dataverse knowledge added to Agents or model-driven apps defines the scope of content that's searched.



What are the column types that Dataverse search can search by using Global Search?

The **Find Columns** on a **Quick Find View** define the searchable fields in the Dataverse search index. Text fields such as Single Line of Text and Multiple Lines of Text, Lookups, and Option Sets are searchable. **Find Columns** of all other data types such as Integer and Double are ignored. For more information, see Select searchable fields and filters for each table.

Why don't I see Global Search results from a table that I enabled for Dataverse search?

If a table isn't part of the model-driven app, it isn't included in search results. Use the Power Apps app designer to verify that the table is included in that app's components. Make sure that the table has a default Quick Find View created and defined. A default Quick Find View is created with a table, but if you remove it, you need to select the Quick Find View you want and set as the default for your table. For more information, see Add or edit model-driven app components.

Can I configure quick actions to show or hide certain commands?

Yes, you can configure quick actions by using version 9.2.21034.00126 or later. Quick actions are a subset of a table's grid-level command set. You can configure them by using ribbon rules. For more information on how to configure quick actions, see Configure Dataverse search to improve search results and performance.

Why are results that appear in suggestions sometimes not seen on the results page?

Suggestions are quick results based on a search performed on the primary column of a table. This feature is enabled for Dataverse search in model-driven apps. Learn more in Inline suggestions.

When you go to the results page, the search terms are treated as the complete search query. The search performs many more types of matching to display a more comprehensive set of results.

Can I configure the order of tables appearing in the Global Search results page?

The order of tables in the **Top results** tab and in the horizontal list of tabs is based on the ranking and relevance of search results for that search term. You can make results from a particular table appear at the top by including the table name in the search term. For example, searching for **account fabrikam** usually ranks result records that have the term **fabrikam** of type **account** higher than result records that have the term **fabrikam** of a type other than **account**.

Can I see search results from SharePoint files and documents through Global Search?

Currently, Dataverse search searches your data in Microsoft Dataverse only. It doesn't search SharePoint files and documents, including the names of the files and the content in the files. It also doesn't search objects of **File** data type in Dataverse.

Why am I unable to view information for party list fields like To, From, and CC in full results?

Party list fields are special fields. Dataverse search doesn't support them, and it doesn't include them in the search results page.

Why don't returns support HTML formatting for memo data types?

Dataverse search doesn't return HTML formatting for memo types to optimize the UI experience.

Why aren't columns enabled for Dataverse search after adding them to a quick find view, to use them in Global Search?

Columns are enabled for Dataverse search only if a quick find view is set as the default view. For more information on how to set a default view, see Specify a default view for a table.

Why doesn't search on the OwnerID attribute work when I enable search on it?

The search and suggest operations don't have access to data from the Owner column. For more information, see Types of columns.

Why doesn't search on the RegardingObjectId attribute work when I enable search on it?

Search doesn't support polymorphic lookup attributes. The RegardingObjectId attribute in activity tables like email and task is a polymorphic lookup attribute.

How is the Dataverse search API throttled?

When you use the Dataverse search API, throttling limits restrict you to one request per second for each user. Additionally, each organization can make up to 150 requests per minute.

What attribute types does Dataverse search indexing support?

- BigInt
- Boolean
- Customer

- DateTime
- Decimal
- Double
- EntityName
- Integer
- Lookup
- Memo
- Money
- Owner
- Picklist
- State
- Status
- String
- Uniqueidentifier
- Virtual (only for MultiSelectPicklistType and FileType)

What attribute types are eligible for facet list fields?

- Lookup
- DateTime
- Money
- Picklist
- Integer
- Customer
- Decimal
- MultiSelectPicklist
- State
- Status

How can I use the Dataverse search API?

See Developer's guide: Search for Dataverse records using the API.

Does Dataverse search support US Government clouds?

Dataverse search strives to maintain functional parity between our commercially available services and those available through our US Government clouds. It's available in US

Government Community Cloud (GCC), US GCC High, and Department of Defense (DoD).

How is Dataverse search reported?

In addition to the database and file storage, Dataverse search includes the indexes that power different experiences. These indexes support search and generative AI across structured or tabular, and unstructured data stored in Dataverse, including files. Dataverse search storage appears at the environment level in the `DataverseSearch` table, previously called `RelevanceSearch`.

Learn more in Dataverse capacity-based storage.

What actions can admins take to manage Dataverse search?

Depending on the experience that uses Dataverse search and its usage, the consumption size might increase. To learn more about managing Dataverse search, see What is Dataverse search.

What happens if I turn off Dataverse search?

If you set Dataverse search to **Off** for the environment, you can't use the search capability in the Power Apps navigation bar or any generative AI experience that relies on Dataverse. This limitation includes experiences like uploaded files or using OneDrive or SharePoint files (via upload) in Microsoft Copilot Studio agents, among other experiences. For details about these limitations, see What is Dataverse search.

See also

What is Dataverse search?
Search for tables and rows using Dataverse search
Configure facets and filters

Last updated on 01/13/2026

Use quick find search

Article • 10/22/2024

Quickly search across multiple tables at the same time and find the information that you're looking for. When you enter a search term the system finds matches to all words in the search term in one column within a table. The words can be matched in any order in the column.

With categorized search, you can search for rows that begin with a specific word or use a wildcard character.

- **Begins with**: Results include rows that begin with a specific word. For example, if you want to search for "Alpine Ski House," type **alp** in the search box; if you type **ski**, the row won't show up.
- **Wildcard**: For example, *ski or *ski*.

> **Note**
>
> Using a wildcard at the beginning of your quick find (single or multiple-table) search query might result in slower performance.

Single table quick find

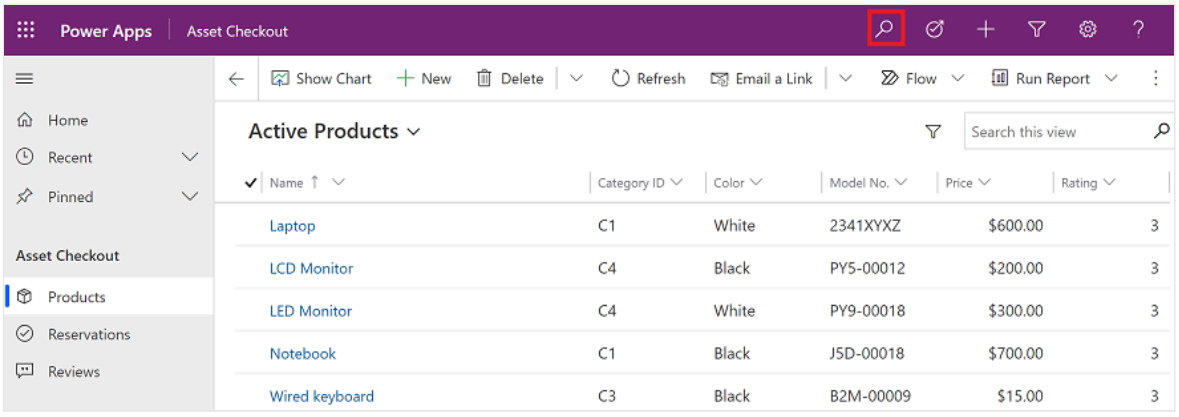
Single table quick find is also called *grid search*. For more information, see Grid search.

Multiple-table quick find (categorized search)

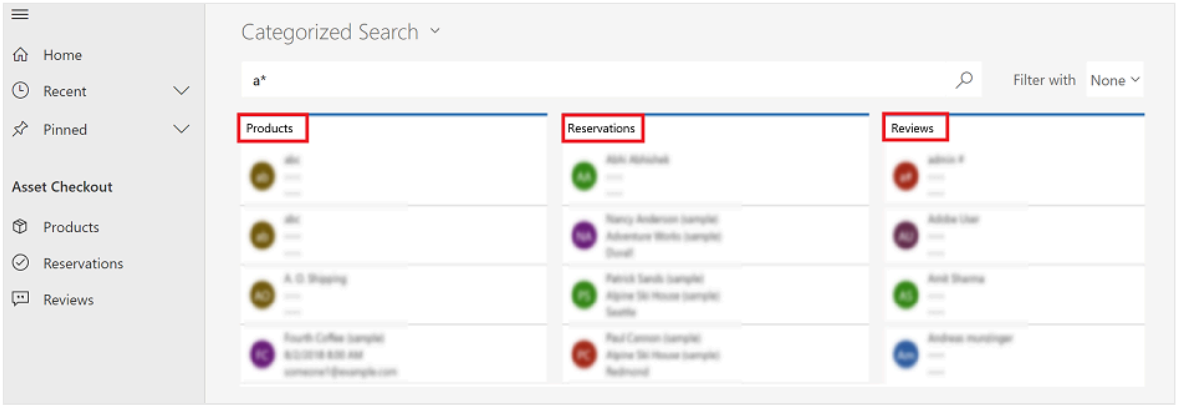
Multiple-table quick find searches up to 10 tables and returns the search results grouped by table.

You can view a maximum of 80 rows per table with multiple-table quick find. You might need to refine your query, if the row that you're looking for is outside the maximum range.

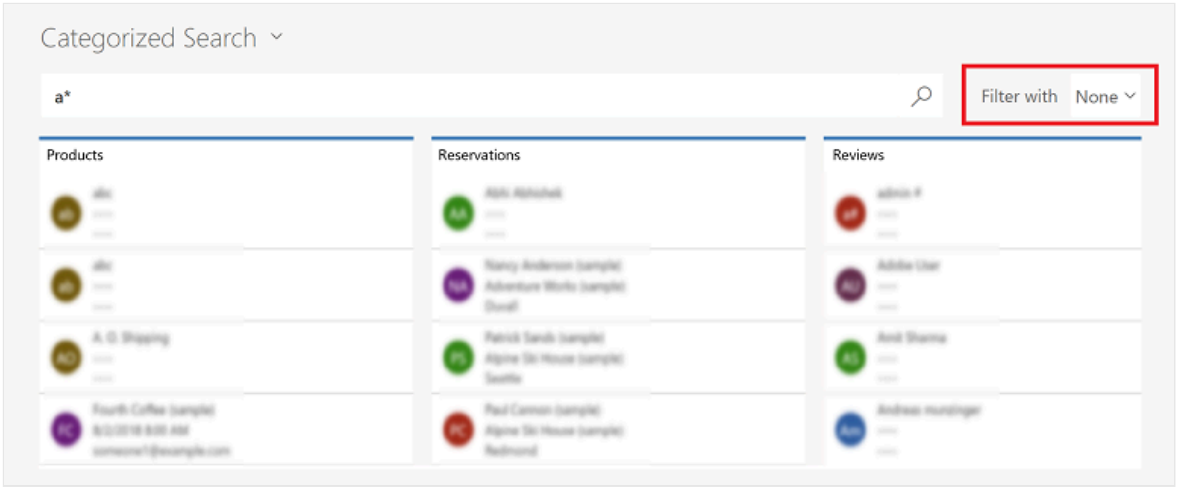
1. To start a categorized search, on the command bar, select **Search**.



2. Type your search words in the search box, and then select **Search**. The search results are grouped by table such as Products, Reservations, and Reviews.



3. Select a table from the **Filter with** list to filter results by table type. To search across all tables, select **None**.



Feedback

Was this page helpful? Yes No

Provide product feedback

Advanced find in model-driven apps

Article • 01/16/2024

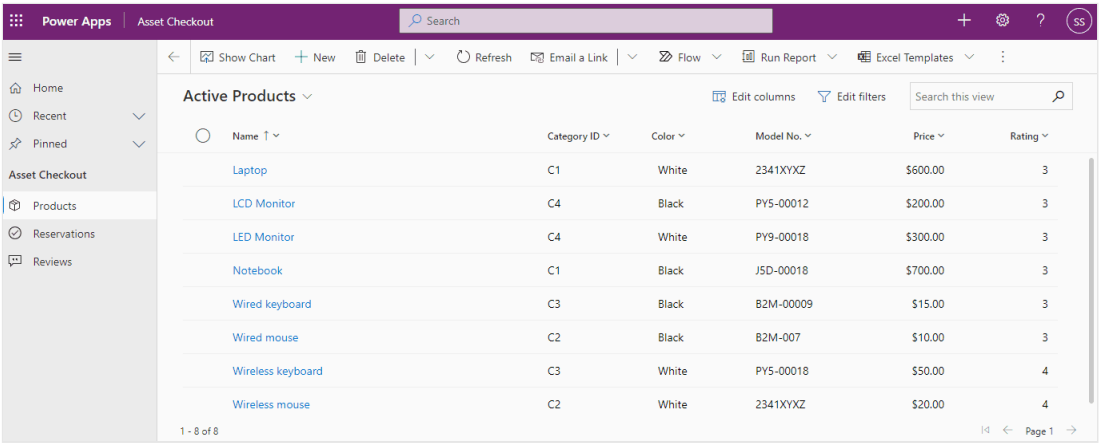
The topic covers the various different ways to use the advanced find experience.

Search for rows and create personal views with modern advanced find

When your administrator enables the modern advanced find experience, it lets you search for rows and create personal views in your app. Depending on how search is setup for your app, the search experience might look a little different. Your administrator can turn on Dataverse search or disable it.

1. To start a search, choose one of the following:

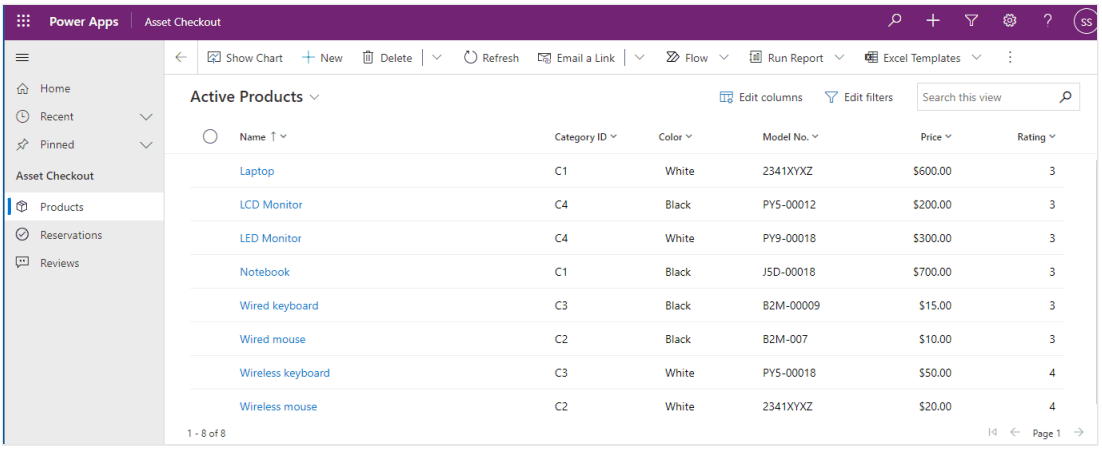
   - **Dataverse search enabled**: If your administrator has enabled Dataverse search, you will see a search bar at the top. Select the search box and then select **Search for rows in a table using advanced filters** to see the list of tables to choose from.



> **Tip**
>
> **Search for rows in a table using advanced filters** option appears at the bottom of the search drop-down menu, only when you have not entered a search term. Once you start typing, you will see results for the search term, with an option to see more search results.

   - **Dataverse search disabled**: If your administrator has not enabled Dataverse search, then you will see the **Search** button and the **Advanced Find** button

on the command bar. Select the **Advanced Find** button, to see the list of tables to choose from.



2. Select a table and choose **Continue** to access the table. The tables in the filter must meet the following requirements:

   - Part of the model-driven app
   - Enabled for Unified Interface
   - Valid for advanced find

3. You will be navigated to the table's grid page, with the default view for the table selected and the rows displayed in a read-only grid. The **Edit filters** panel will be opened, so you can filter the data on that table. For more information, see create and manage personal views for that table.

For more information on creating and managing system views, see Create and edit public or system views.

Create, edit, or save a view using legacy advanced find

If your administrator has not enabled the modern advanced find experience, you can find the rows you want by using the legacy Advanced Find search option. You can also use Advanced Find to prepare data for export to Office Excel so that you analyze, summarize, or aggregate data, or create PivotTables to view your data from different perspectives.

> **Note**

To use the modern advanced find in model-driven apps, see Create and manage personal views on a grid page .

1. On the command bar, select **Advanced Find** button  .
2. Specify what to search for.
    a. In the **Look for** list, select a row type.
    b. Choose **Select** to define search criteria: column (for example, Account Name or City), the query relational operator (part of an expression such as "is equal to" or "contains" that defines how a specified column should be compared with a value.), and the values to locate (for example, "Seattle" or "Email").

    You can select columns from the current row type, or from related rows. For example, an account might have many related contact rows.

    At the bottom of the **Select** list, the **Related** section shows related row types. For example, most row types have a related Notes row type. To select columns from a related row type, select column and a new **Select** link appears.
3. Specify the columns to include in the search results.
    a. Select **Edit Columns**, and then select **Add Columns**.
    b. Select the row type that includes the columns you want to add.
    c. Select the columns you want to add, and then select **OK**.
4. Specify the sort order.
    a. Select **Edit Columns**.
    b. Select **Configure Sorting**.
    c. Specify the column to sort on, specify the sort order, and then select **OK**.
    d. Select **OK**.
5. Select **Results**.

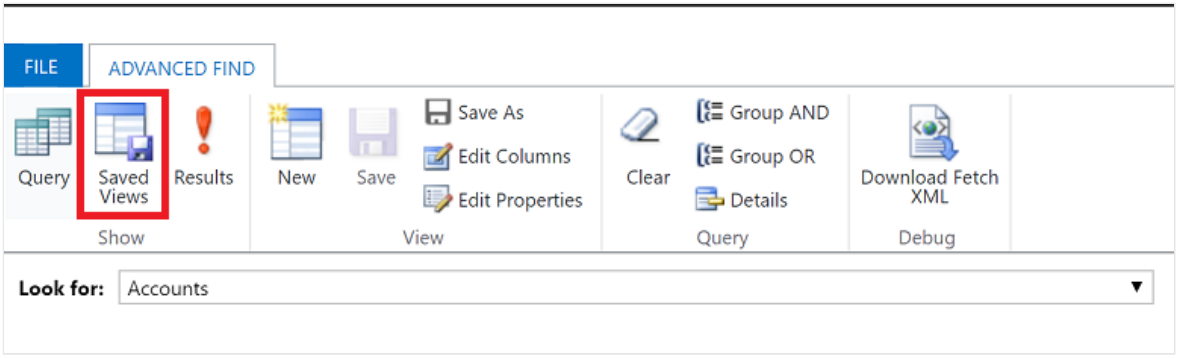
To save the results, select **Save As** and name the personalized view.

Share a personal view

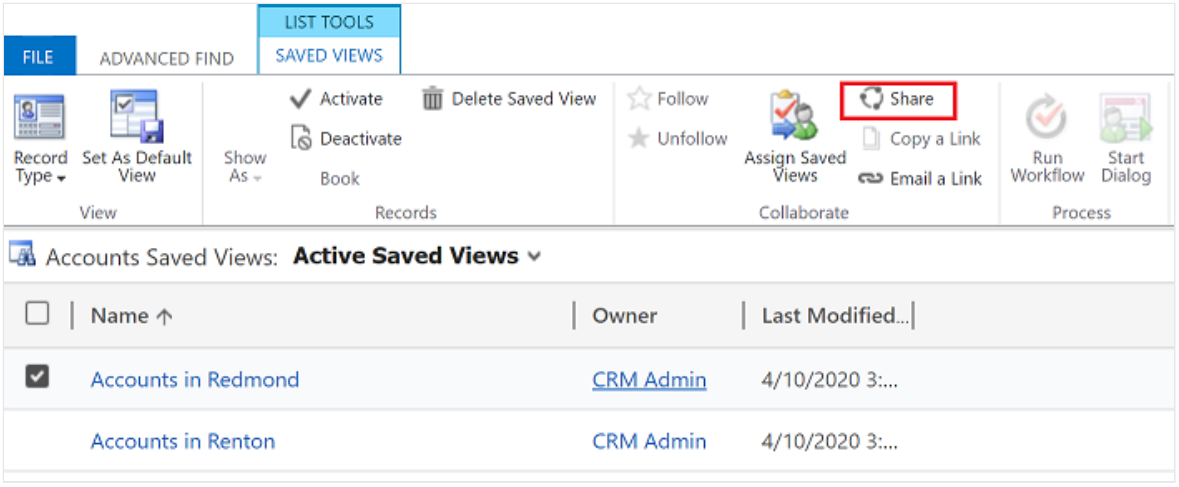
Before you can share a personal view, it must be saved.

1. On the command bar, select the **Advanced Find** button.

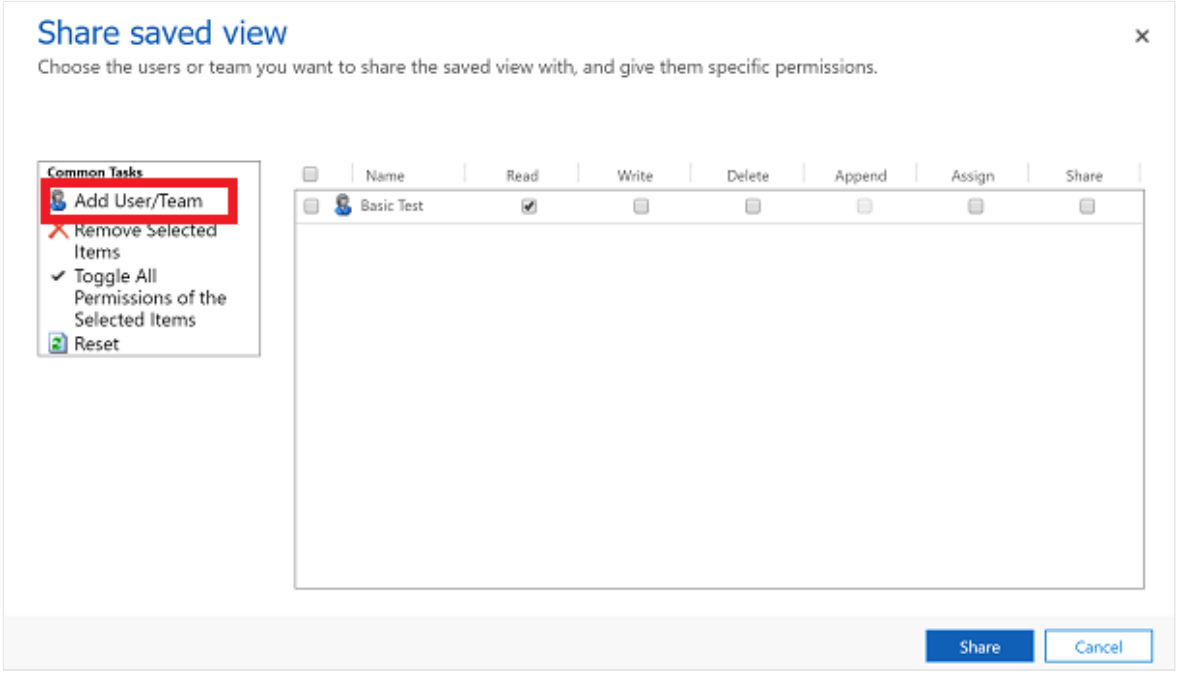
2. On the Advanced Find tab, select **Saved Views**.



3. From the list of saved views, select a view and then select **Share**.

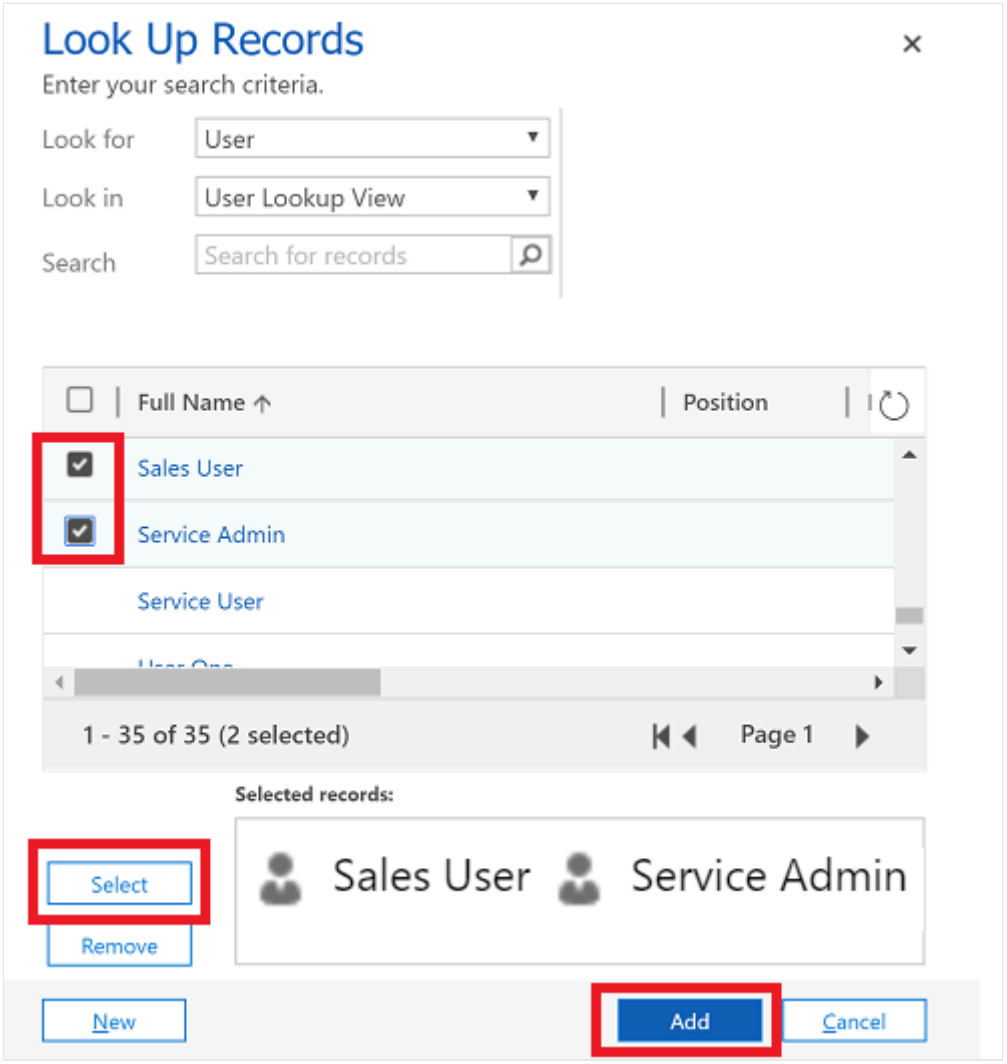


4. In the **Share saved view** screen, select **Add User/Team**.

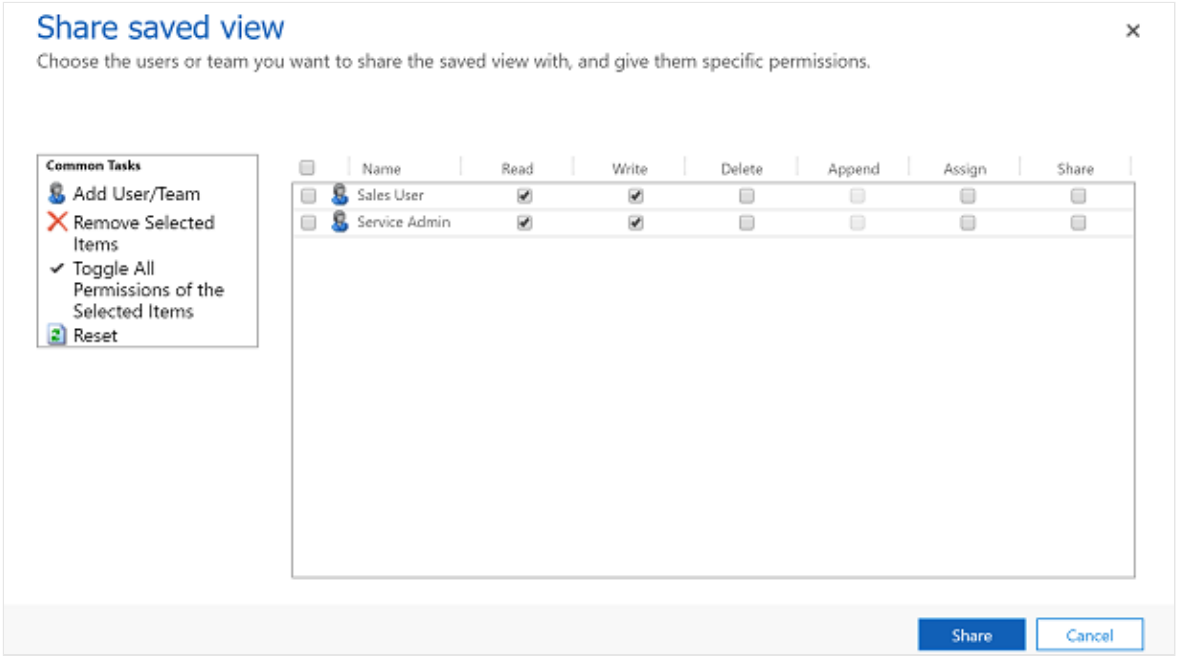


5. Use the look up to find the users or team that you want to share the view with.

6. When you find the user or team, select their name and then choose **Select** and then **Add**.

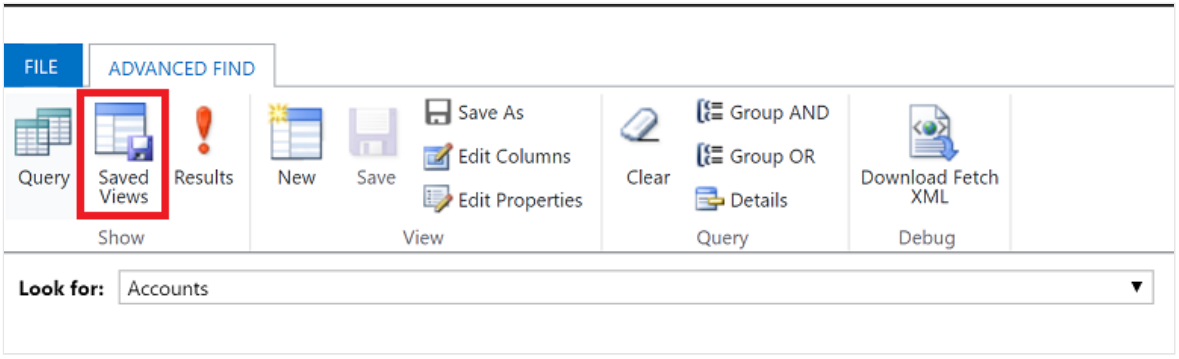


7. Select each user's permission and then select **Share**.

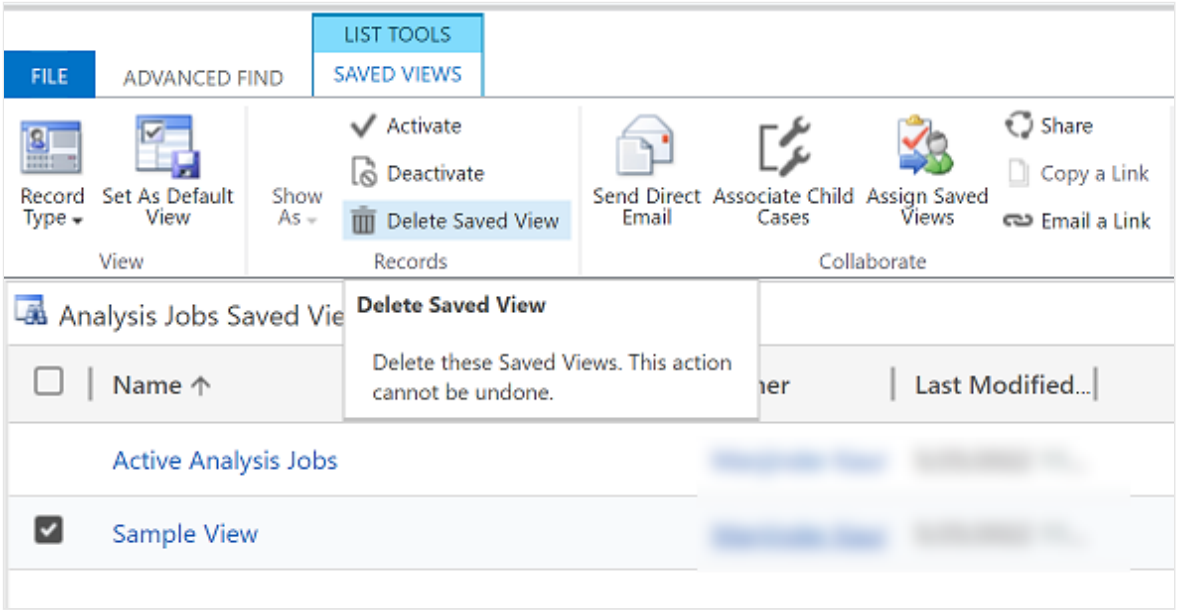


Delete a personal view

1. On the command bar, select the **Advanced Find** button.
2. On the Advanced Find tab, select **Saved Views**.



3. From the list of saved views, select the view that you want to delete and then select **Delete Saved View**.



Feedback

Was this page helpful? Yes No

Provide product feedback

How to import data

10/22/2025

Import data stored somewhere else into your model-driven app using the import feature in Power Apps.

Every table requires columns that must exist in your input file. You should download an Excel template, add your data, and then import the file to your app. The template saves time and effort. Don't add or modify columns in the template to avoid issues during the import.

Before you import a file, make sure the column headings match the column names in your app. During the import process, the system tries to map the column names from your input file to the column names in the app. If they don't match, then you have to manually map the fields or you might run into issues during the import process.

If you're an advanced user and know the required columns for a given Dataverse table, define your own Excel, CSV, or XML source file and then follow the steps in this article on how to import data to your app. These file formats are supported:

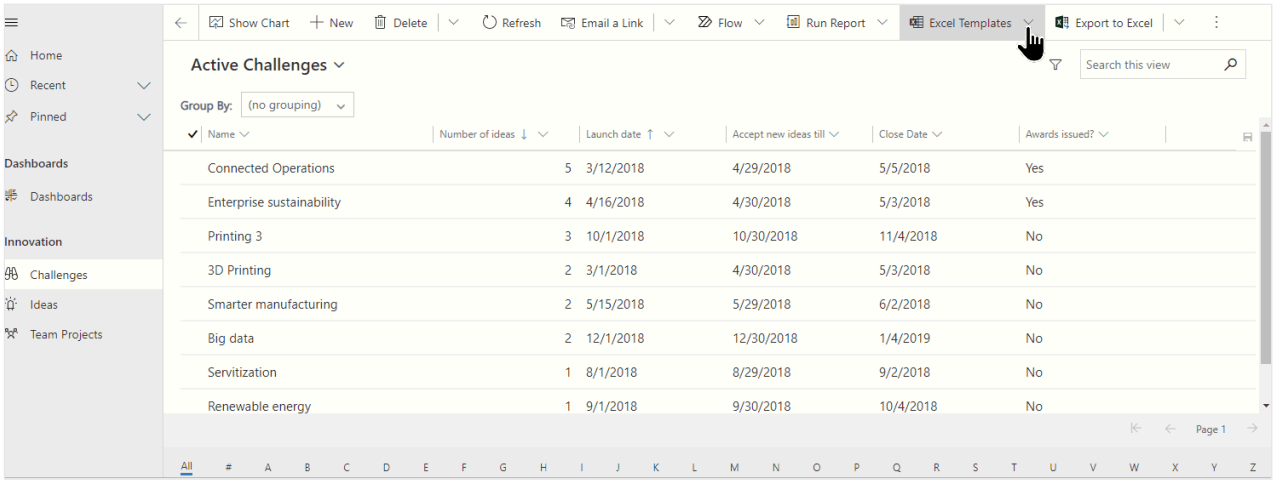
- Excel workbook (.xlsx)
- Comma-separated values (.csv)
- XML Spreadsheet 2003 (.xml)

The maximum file size allowed for .zip files is 32 MB. For the other file formats, the maximum file size allowed is 8 MB.

Download an Excel template

To avoid mapping issues, you should use an Excel template that you can download from your app. Once the template is downloaded, add your data and then import the file back to your app. Remember don't add or modify columns in the template to avoid issues during the import process.

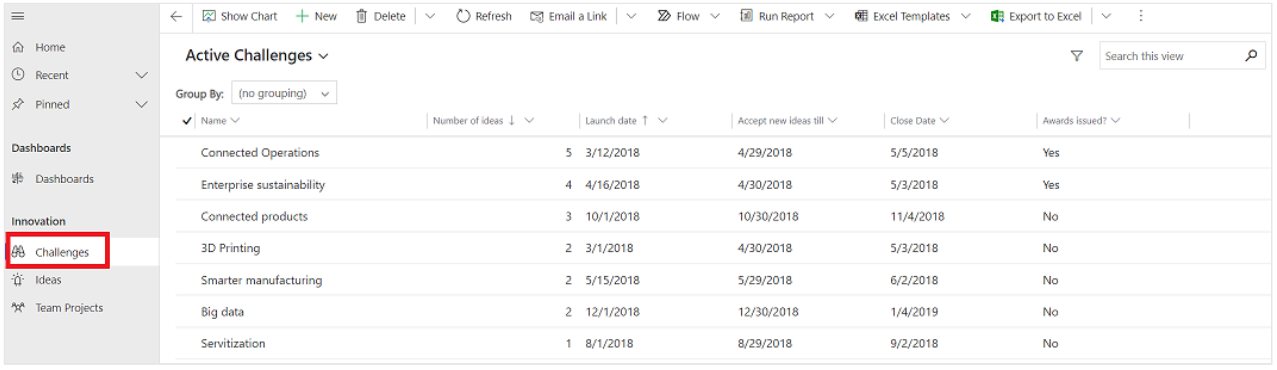
1. Open your app and from the left nav select a table.
2. On the command bar, select **Excel Templates** > **Download Template**.
3. Select the table type and view to download or select **Edit Columns** and choose the rows to include in the template.
4. Select **Download**.



Import your data

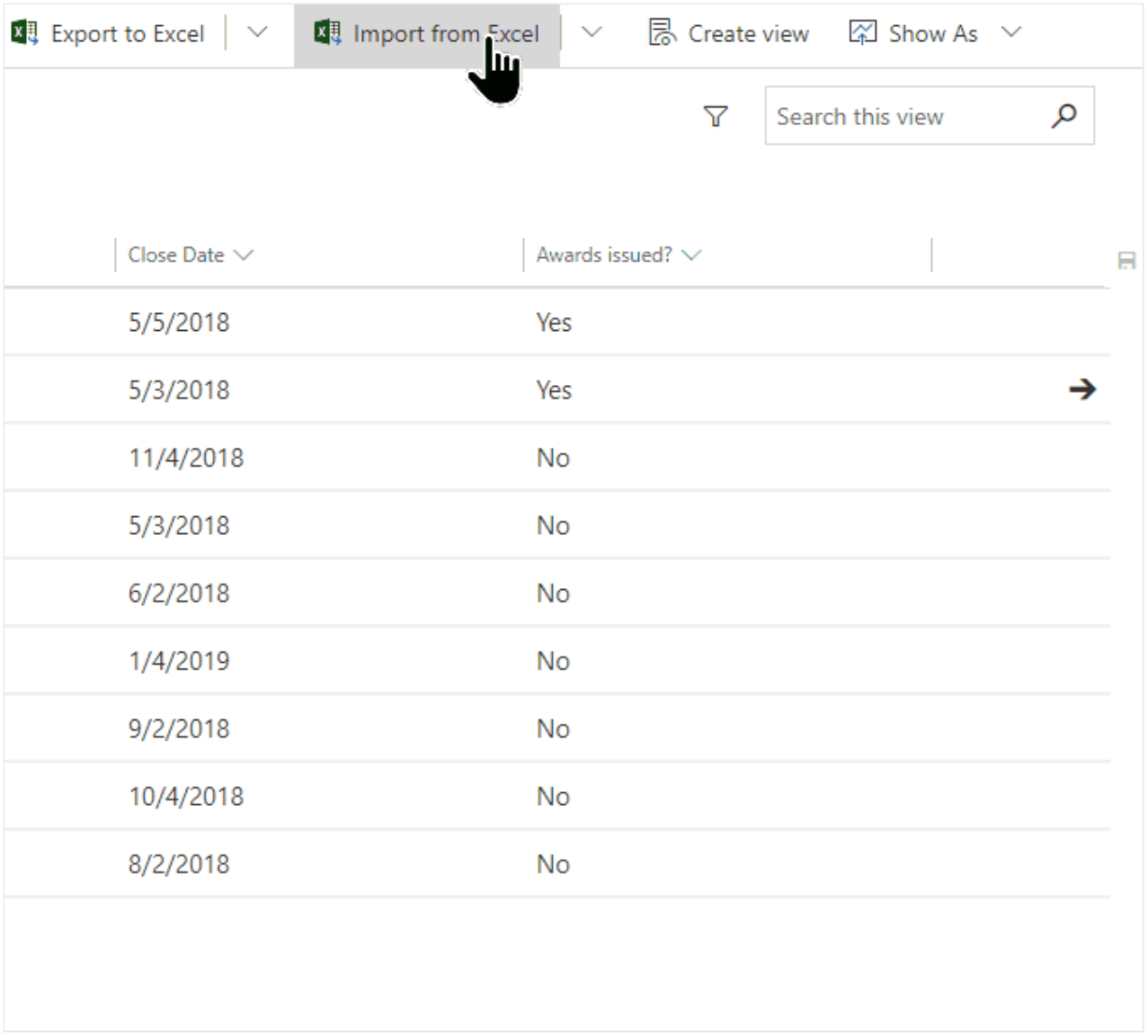
Use the template that you downloaded in the previous step (modified with your data) and import the file to your app.

1. Open an app and from the left nav select a table.



2. On the command bar, select the file type to import from:

   - **Import from Excel**: Select **Import from Excel** if you're using an Excel template.
   - **Import from CSV**
   - **Import from XML**



3. Select **Choose File** and browse to the folder where the file is saved. Select the file, select **Open**, and then select **Next**.

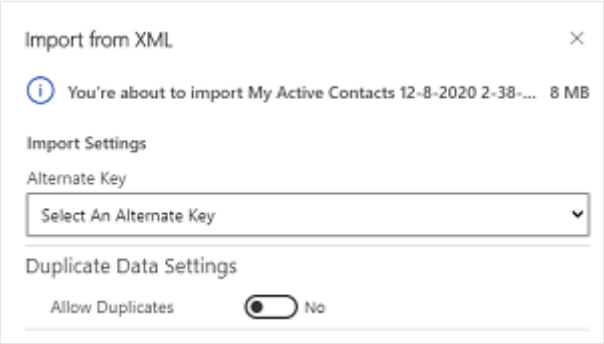
   > **Tip**
   >
   > You can only import one file at a time. To bring in more files, run the wizard again.

4. Select whether to **Allow Duplicates** or not. More information, see Set up duplicate detection rules to keep your data clean.

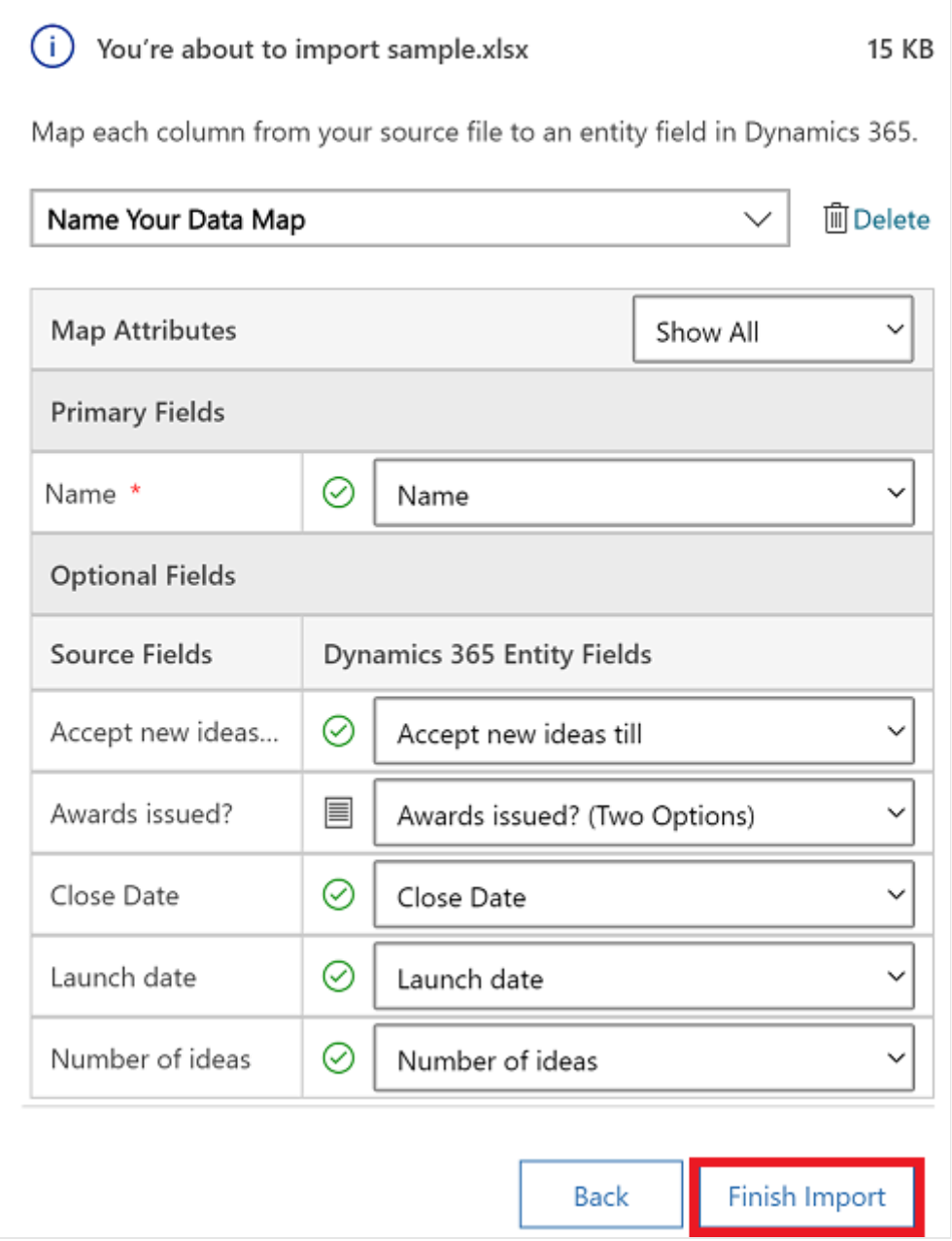
5. For CSV and XML files (skip this step if you're importing an Excel file):

   - For a CSV file: Select the drop-down list and select the data delimiter and field delimiter that's used for the CSV file.

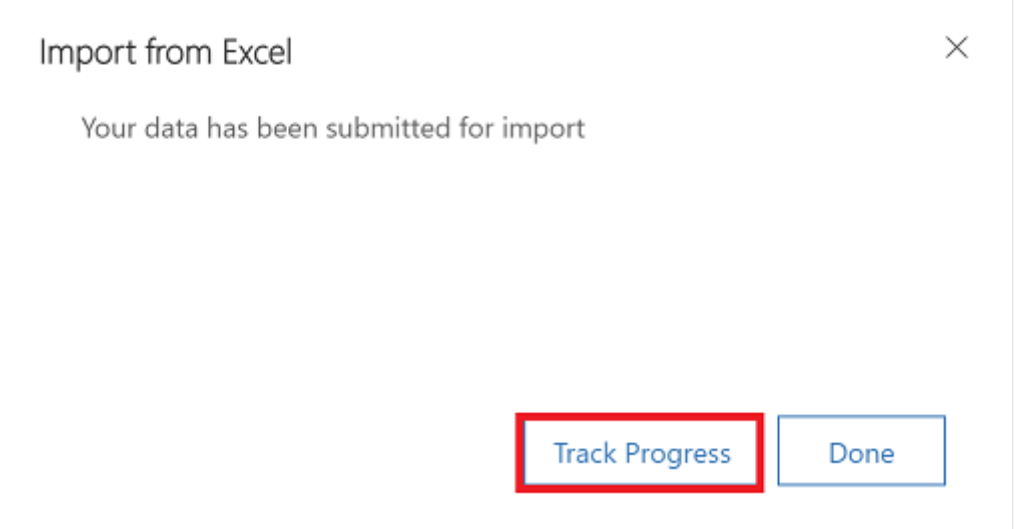
   - For CSV or XML file: If you have an alternate key defined, select it from the alternate Key drop-down list. The alternate key is used to uniquely identify and update rows during import. More information: Define alternate keys to reference rows.



6. Select Review Mapping and verify the columns (fields) are mapped correctly. If everything looks good, select **Finish Import**.

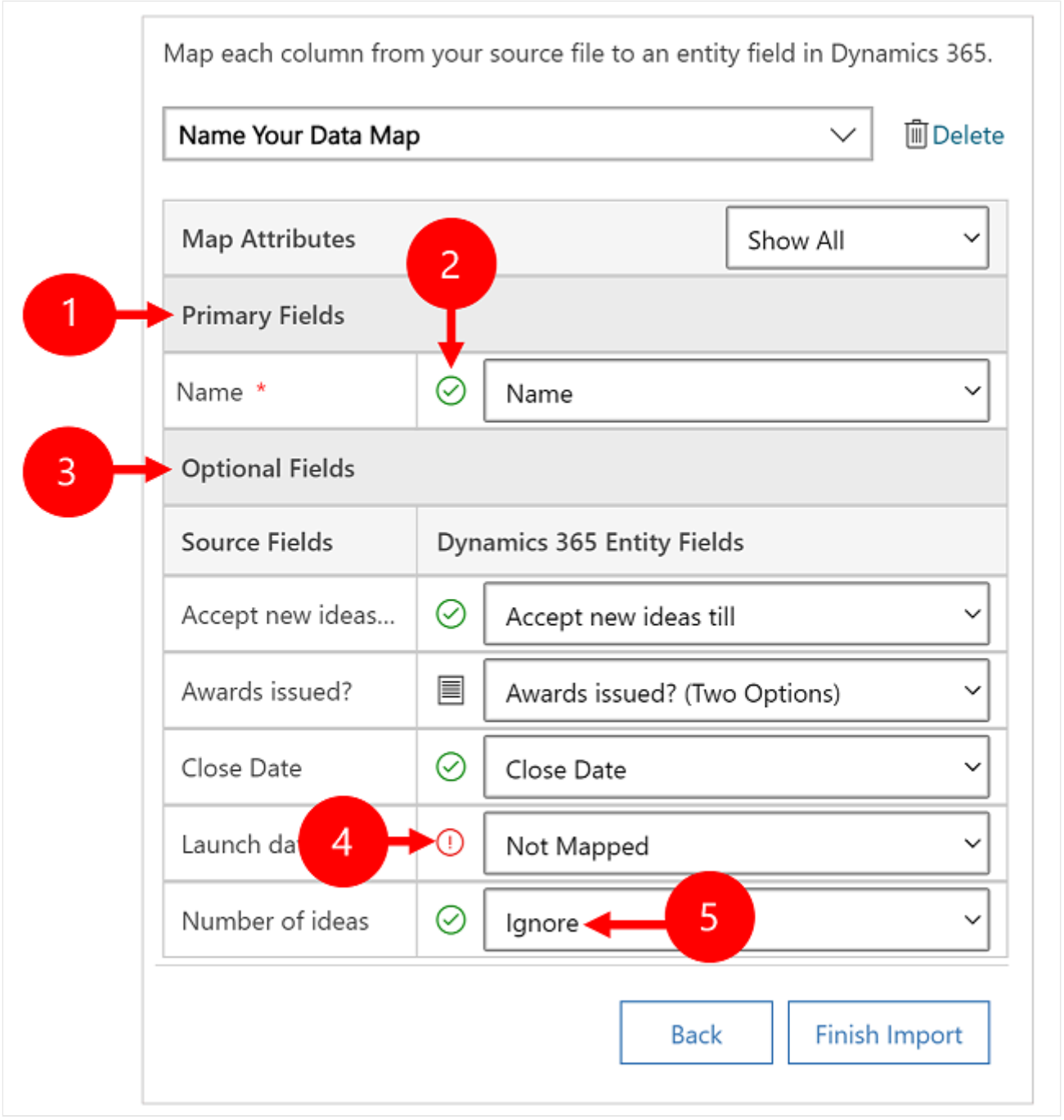


7. Select **Track Progress** to see the progress of the import.



Review mapping

When you import a file, it's important to review the column headings and verify that they match the columns (fields) in your app.



Legend:

1. **Primary Fields**: Shows all the required columns for the table that must be mapped for the data to be imported successfully. If the column headings of your source file match the column display names, these columns are automatically mapped.

2. **Mapped columns**: Correctly mapped columns are shown with a green check mark.

3. **Optional Fields**: These are optional column headings in your source file. If the column headings match the column display names, the columns are automatically selected in the corresponding drop-down lists.

4. **Unmatched columns**: If the column headings don't match, the unmapped columns are shown with a red exclamation point. To map the column correctly, select a column to map

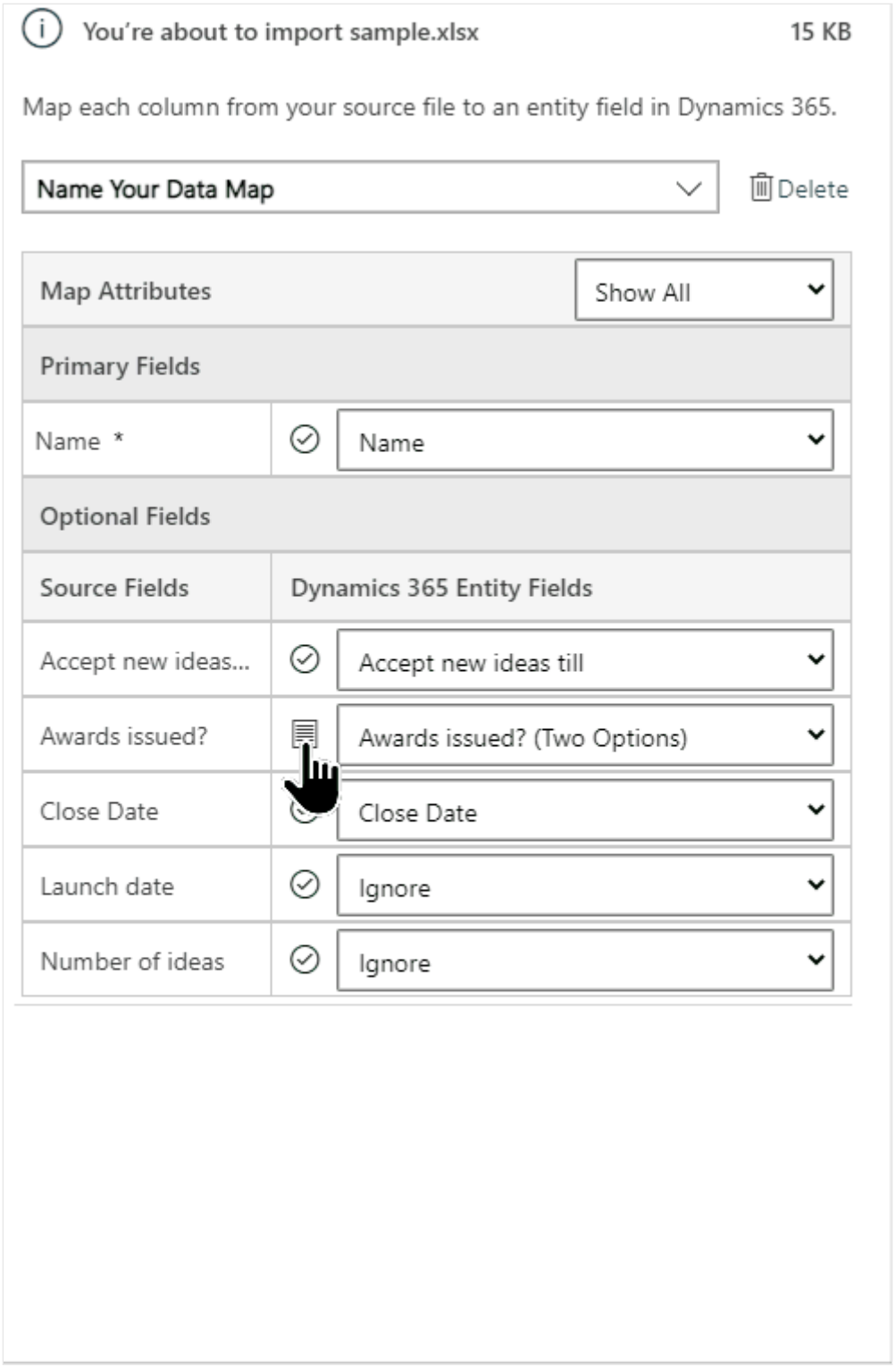
to the unmapped column heading of your file.

5. **Ignore** (For **Optional Fields** only): Choose **Ignore** from the drop-down list. Data from ignored columns aren't imported into your app.

Option set

If any column in your source file includes a fixed set of values, you must map the column to a column of type **Option Set**. A column of this type has values such as **Yes** and **No** or **Low** and **High**.

To do this, select the button next to the option set column, select the values, and then select **OK**.



The option values drop-down list combines the values available in the incoming file with those already in your app. For example:

- **Values in import file**: Low, High
- **Values already in your app**: Cold, Warm, Hot
- **Resulting target values**: Cold, Warm, Hot, Low, High

After import, the import wizard will add all mapped values in your app, but will drop unmapped values from the import file that aren't yet in your app. For example, you could map the "Low" source value to the "Cold" target value, but map the "High" source value to the (new) "High" target value. Based on these mappings, the import wizard creates "High" as a target

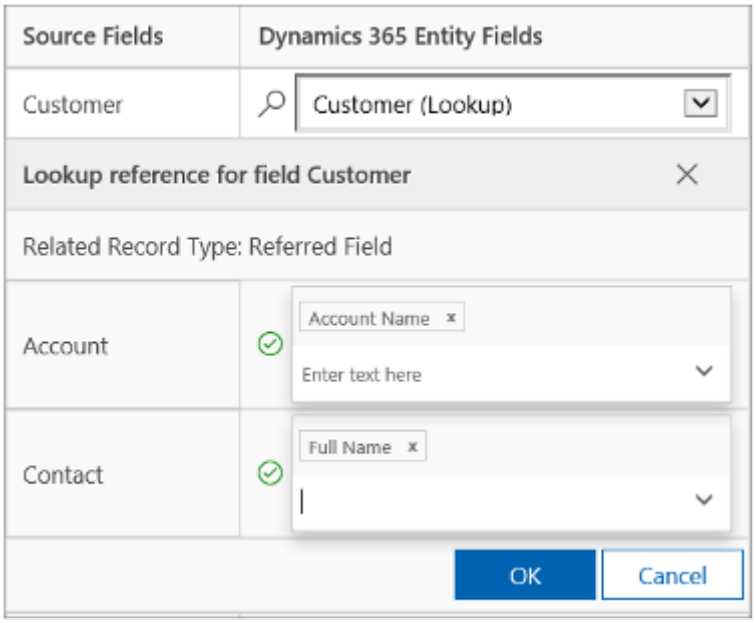
value in your app. It doesn't create "Low" as a target value in your app because you didn't map any source to this target value.

> **Note**
>
> You can map a column in your source file to a "Two Options" or "Multiselect Option Set" column. When mapping to a "Multiselect Option Set," if your source file includes values not available in your app, new values may be created in Dataverse if the import mapping is set to **NotMapped**. If the mapping is set to **Ignore**, the record can be successfully imported. And, for "Multi-Select Option Set" fields, the values are not imported, and no new values are created in the target environment.

Map lookup columns

If some data in your source file references other existing rows in your app, you must map the column in the source file to a lookup column in your app.

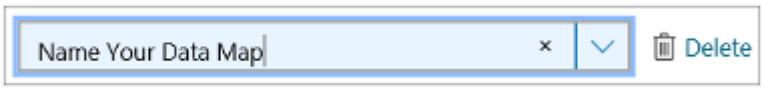
For example, you might want to import a file named Leads.csv, which contains customer rows. The **Customer** column in Leads.csv contains the associated account or contact data. To map this, select the **Lookup Reference** button next to the lookup column. The **Lookup Reference** section opens and lists the tables related to the current table.



For each table, select the columns to search during import to retain the relationships between the rows, and then select **OK**.

Save mapping settings

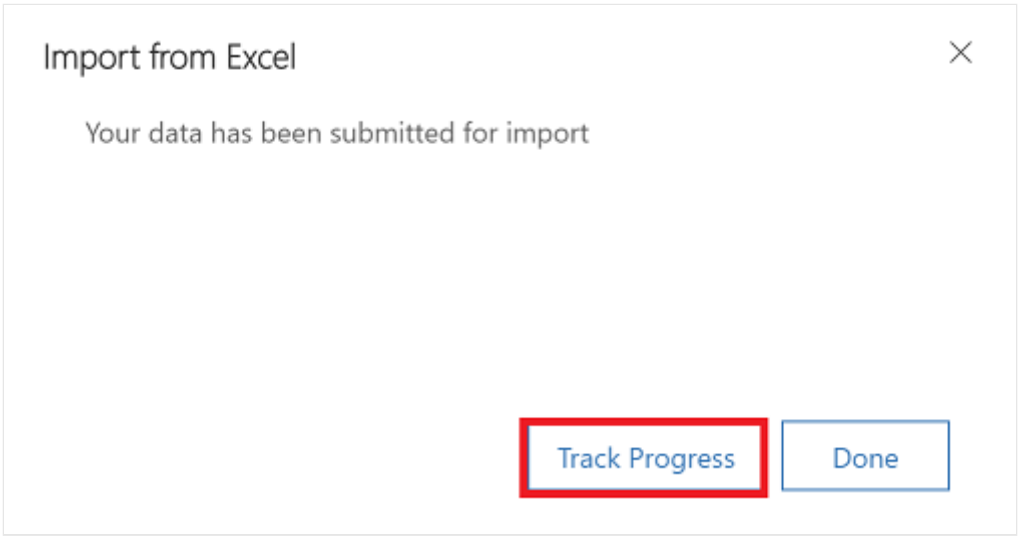
To save the mapping settings for next time, enter a name in the **Name your data map** box. This way, the next time you import a similar set of data, you'll be able to use the same mapping.



Open the import log

After you initiate an import, you can monitor the status of the import job and view results in the import log.

In the last step of the import wizard, select **Track Progress** to view the progress of the import job.



If you missed this step, append the following code to your app URL: `&newWindow=true&pagetype=entitylist&etn=importfile&viewid=a2927069-e4dc-4e8f-8fb1-d2f6c40c427e&viewType=1039`.

For example, if your app URL is `https://org12345.crm.dynamics.com/main.aspx?appid=00001111-aaaa-2222-bbbb-3333cccc4444`, then the URL to view the import log is `https://org12345.crm.dynamics.com/main.aspx?appid=00001111-aaaa-2222-bbbb-3333cccc4444&newWindow=true&pagetype=entitylist&etn=importfile&viewid=a2927069-e4dc-4e8f-8fb1-d2f6c40c427e&viewType=1039`.

View the status of an import, and correct errors

After the import is complete, you can view the records that were successfully imported, failed to import, or were partially imported.

1. Open the import log.

Delete imported records

You can delete the import file and imported records if you want to reimport the data or you don't need the data for some reason.

1. Open the import log.
2. Select the import file that you want to delete, followed by one of the following actions:
    - **Delete import source file**. This option deletes the import log file and its details.
    - **Delete imported records**. This option deletes all records that were imported from the selected file.
    - **Delete All**. This option deletes the import file along with the records that were imported from the import file.

Limitations

- Excel Import isn't supported with Virtual tables or Elastic tables
- Excel import updates fields from the primary table but ignores fields from related tables.
- Excel import jobs run in parallel only when the file size is under 1 MB. Files larger than 1 MB are processed sequentially to avoid resource exhaustion.

> **Note**
>
> To enable parallel processing, keep each file under 1 MB. For `.zip` files, the decompressed size must also be under 1 MB. Note that files with special characters may have a calculated size that exceeds 1 MB even if the original file appears smaller.

See also

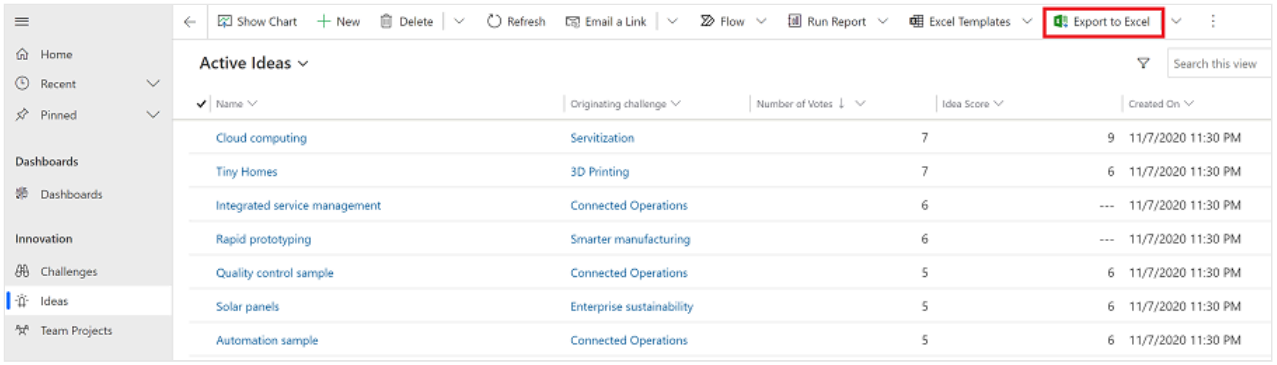
Download a template for data import

Export data to Excel

08/05/2025

Export data from your model-driven app to Excel. The allowed data export size depends on the number of rows, columns, lookup fields, and lookup relationship types. The export query might time out depending on the combination of these factors.

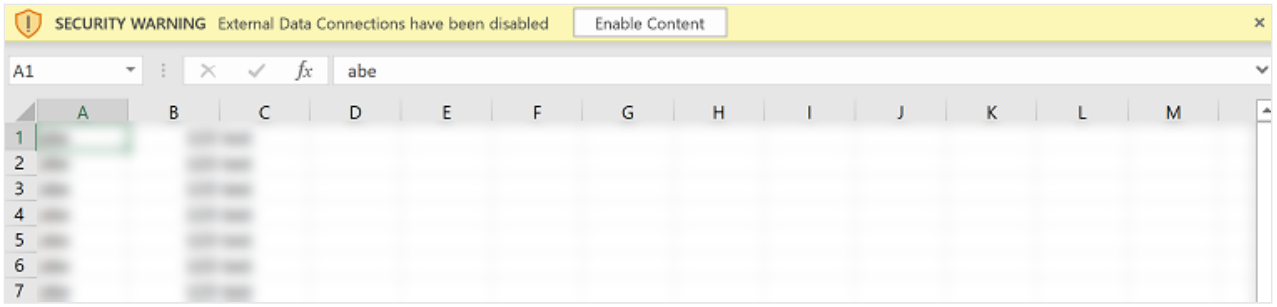
1. From the left nav, select a table that you want to export data from.
2. On the command bar, select **Export to Excel**.



3. When the download is complete, navigate to the location of the downloaded file.

> **Note**
>
> When you download a worksheet, it should automatically save to your computer. However, if it doesn't then make sure that you save it before you open and edit it. Otherwise, you might get this error message: **Excel cannot open or save any more documents because there is not enough available memory or disk space.**
>
> Follow these steps to fix the issue:
>
> a. Open Excel and go to **File** > **Options** > **Trust Center Settings Center Settings** > **Protected View**.
>
> b. In **Protected View**, clear all three items.
>
> c. Select **OK** > **OK**.
>
> We still strongly recommend that you save and then open the data file rather than disabling protected view, which might put your computer at risk.

4. Open the saved Excel file.
5. If you see the security warning **External Data Connections have been disabled**, select **Enable Content**.



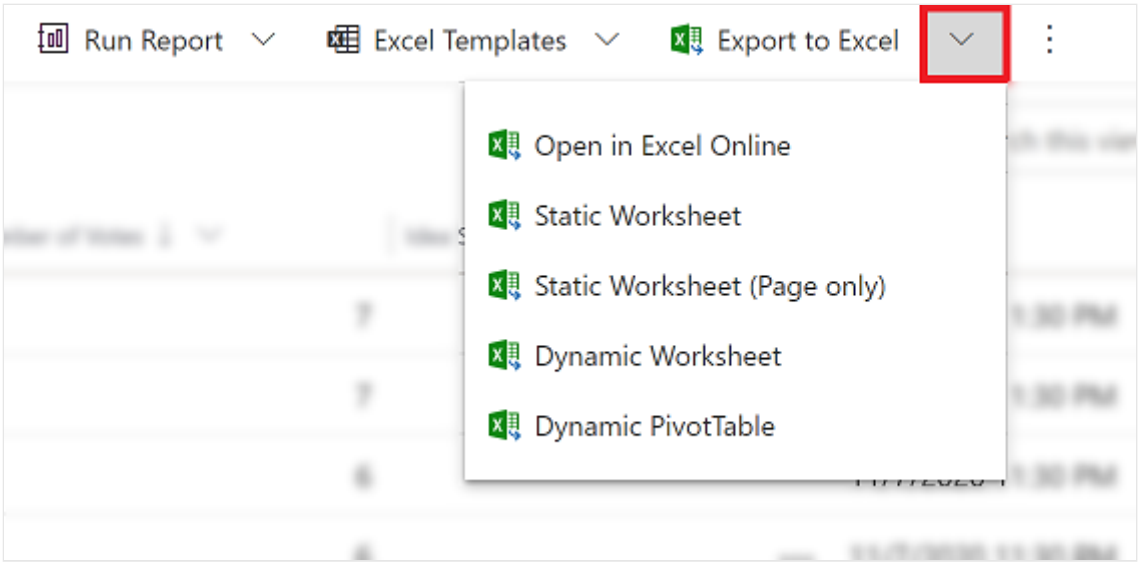
The data is exported in the same format that you see in your app. Text remains as text, numbers remain as numbers, and dates remain as dates. However, when you export data from your app to Excel, some cell formats might change. The following table summarizes how you see the data in apps and how the cell format changes when you export the data to Excel.

Expand table

Data format in model-driven apps	Cell format in Excel
Text, Ticker Symbol, Phone, Options set, and Look Up	Shows as Text and option set becomes drop-down list
Email, URL	Shows as General
Number	Shows as Number without group separator
Currency	Shows as Number and doesn't include dollar sign ($)
Date only, Date and Time	Shows as Date only
Calculated and Roll-up columns	Editable in Excel but can’t be imported back to Power Apps
Secured columns	Editable in Excel but can’t be imported back to Power Apps

Export options

There are other export options such as Excel Online, Static Worksheet, Dynamics Worksheet, or Dynamics PivotTable.



The following table summarizes the different options. Pick the one that works best for you.

Expand table

Task	Learn more
Do an *ad-hoc* or *what if* analysis without modifying app data. Or, quick bulk edits to multiple rows.	Open app data in Excel Online
Get a snapshot of the data at the current date and time or if you want to share the data with others.	Export to an Excel static worksheet
Get the most up-to-date information and be able to refresh it in Excel and match what you see in your app at any time.	Export to an Excel dynamic worksheet
View your app data in a pivot table.	Export to an Excel PivotTable

Known issues

Exporting to Excel doesn't support virtual tables.

- If Export to Excel times out, try these options:
  - Export smaller batches by adjusting the filter criteria.
  - Reduce lookup fields that increase query complexity when retrieving data from related tables.
  - Reduce the number of columns.

- You can export your app data to Excel (.xlsx format), use the file as a template to add more data, and import it back into your app. If you add or modify columns in the Excel file and import it back into your app, the import fails because the added or modified columns don’t map to tables in your app. When you import an Excel file into your app,

make sure that columns and tables are mapped correctly to avoid import failures. For more information on mapping, see Review mapping.

- If you're using Excel 2010, you might get this error message when you export data from the Accounts area: `The file is corrupt and cannot be opened.`
  The error message occurs due to a setting in Excel. To fix the issue:

  1. Open Excel 2010 and go to **File** > **Options** > **Trust Center** > **Trust Center settings**.
  2. Select **Protected view** and then clear the check boxes for the first two options.
  3. Select **OK** and then close the **Options** dialog box.

See also

Troubleshoot export to excel

Open your app data in Excel Online

Article • 10/23/2024

Open your app data in Excel Online and make quick edits or do an ad-hoc analysis. You can make changes to your app data in Excel Online and then save the updated information back to your app.

This feature requires that you have an Office 365 subscription or a subscription to an online service such as SharePoint Online or Exchange Online and a Microsoft account.

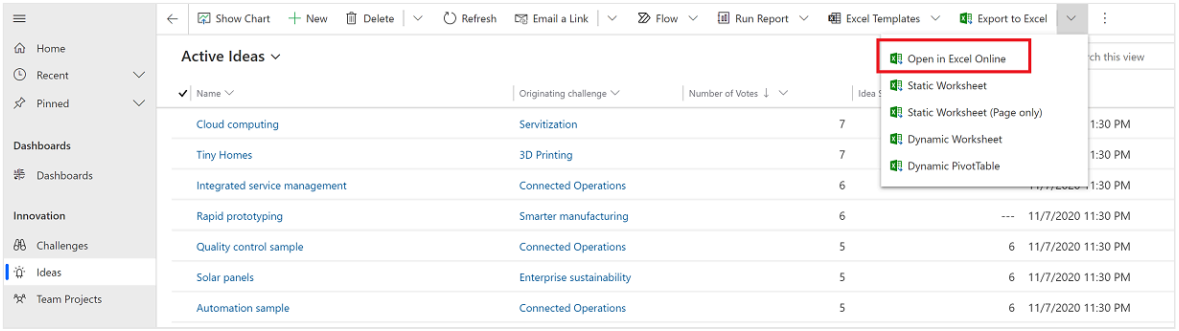
A few things to note:

- The data for an *ad-hoc* analysis with Excel Online is stored temporarily. Any additions, such as charts, graphs, calculations, columns, and colors won't be saved back to the app from the ad-hoc analysis in Excel Online. Keep the existing format of the Excel cells to prevent problems when you save the file. The file import might fail if you make numerous changes.
- Updated data in an app isn't immediately reflected in Excel Online if the same view was opened in the last two minutes in Excel Online. After that time frame, any updated data should show in Excel Online.
- The option to open data in Excel Online isn't available for all tables. If you don't see the option for a table, then it's not available for that table.

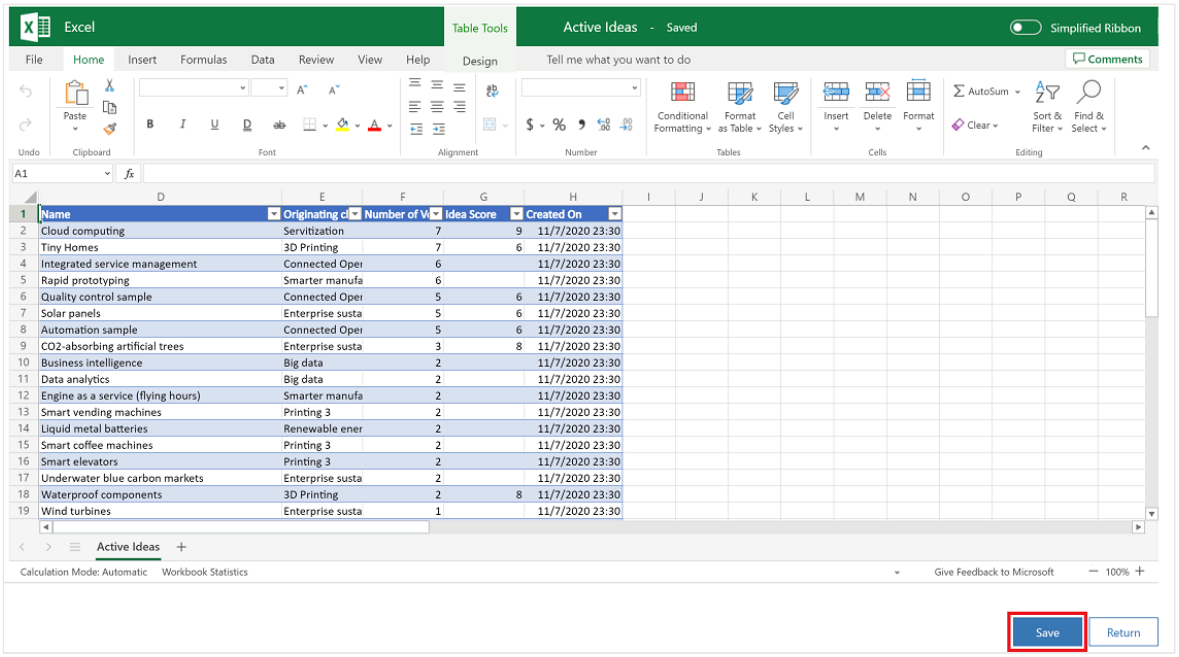
Export to Excel Online

1. On the left nav, select a table.

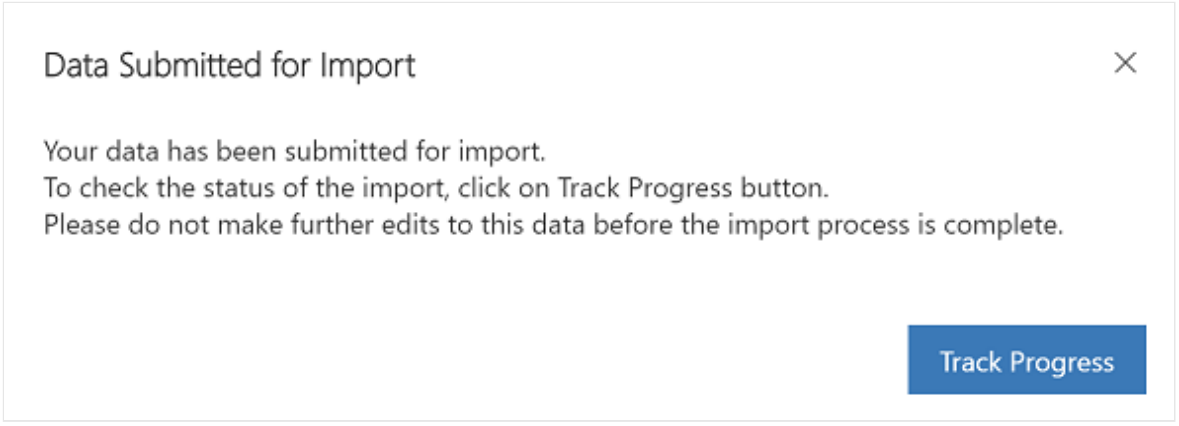
2. On the command bar, select the **Export to Excel** menu and then select **Open in Excel Online**.



3. Make your edits in the Excel Online file and when you're done select **Save**.



4. Your changes are submitted for import. To track the progress of the import, select **Track Progress**.



> **Note**
>
> By design, you can't do a File > **Save As** in Excel Online. If you do, you'll get a **Can't Save Workbook** error message.

Feedback

Was this page helpful? Yes No

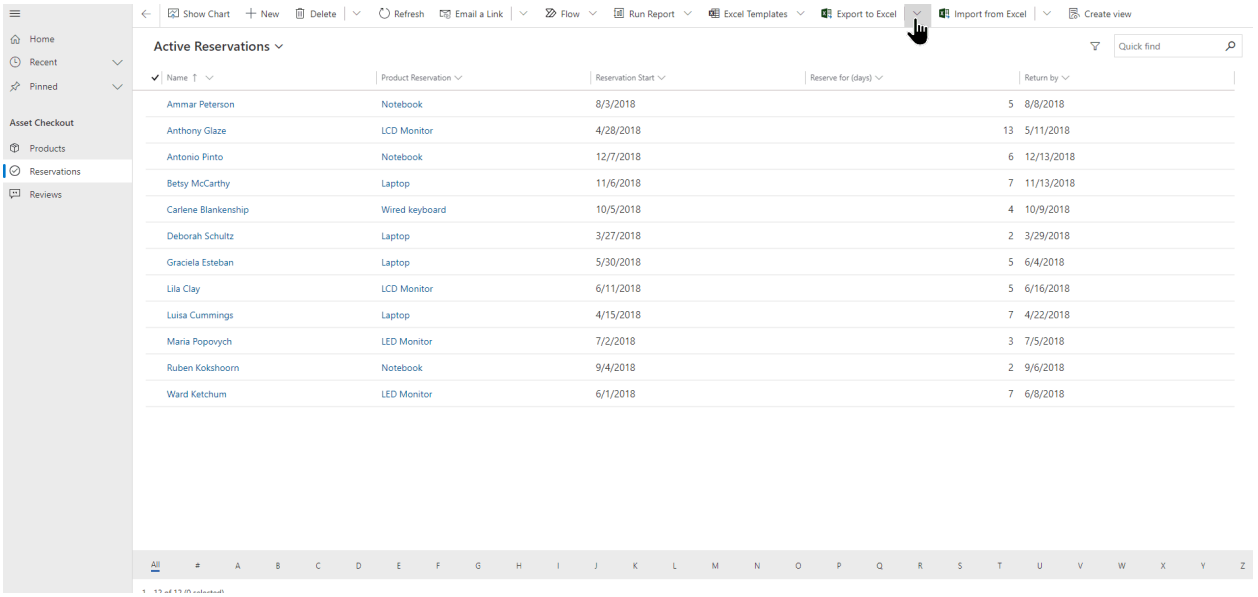
Provide product feedback

Export data to an Excel dynamic worksheet

Article • 10/22/2024

Export your app data to an Excel dynamic worksheet to get the most up-to-date information from your app. When you export your data to an Excel dynamic worksheet and then make changes to your data in the app, you can use the refresh feature in Excel to get the latest information from the app.

In the example below, we downloaded an Excel dynamic worksheet then went back to the app and changed the **Reserve for days** column from **5** days to **10**. Then opened the Excel worksheet and refreshed the data to get the latest information from the app.



A few things to note:

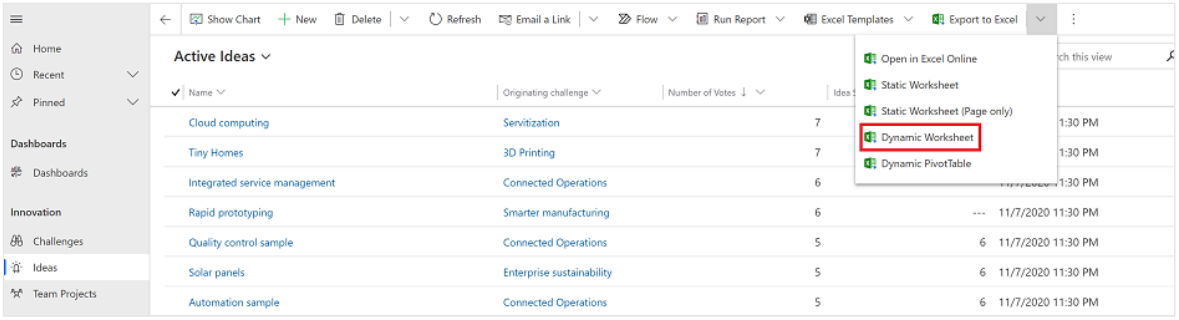
- You can export up to 100,000 rows at a time. You can't export data to a dynamic worksheet in Excel for all tables. If you don't see the option for a table, then it's not available for that table.
- You can email a dynamic Excel file or store it as a shared file if the recipients are in the same domain as you. When recipients open the dynamic file, they only see the data that they have permission to view.
- Some system views can be exported only to a static Excel worksheet.
- Currency values are exported to Excel as numbers. To format the data as currency after you have completed the export, see Format numbers as currency.
- The date and time values that you see in the app show up only as Date when you export the file to Excel, but the cell actually shows both the date and time.
- If you're an app maker, you can use the Microsoft Power Apps Excel Add-in to download your app data and make edit in Excel and then save the data back to

your app. For more information, see Open table data in Excel.

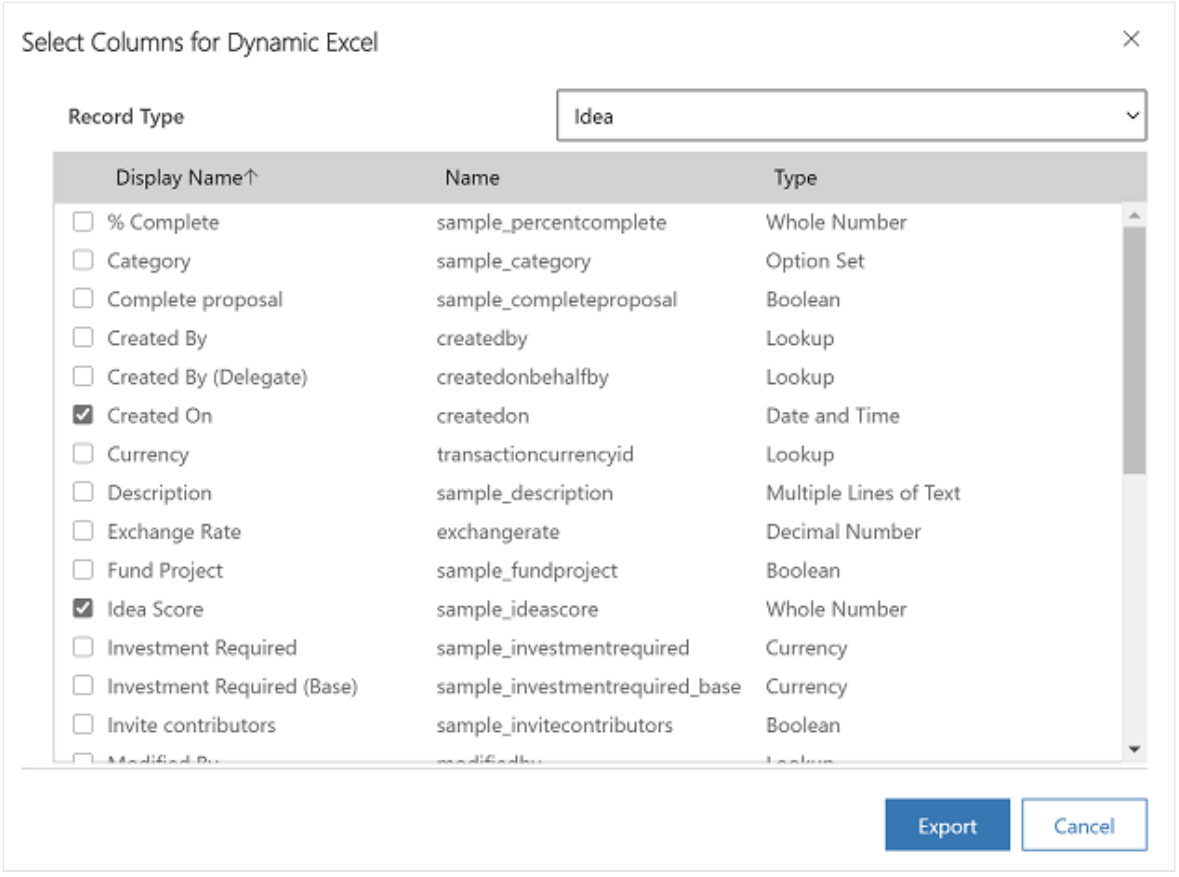
Export a dynamic worksheet

1. On the left nav, select a table.

2. On the command bar, select the **Export to Excel** menu and then select **Dynamic Worksheet**.



3. Select the columns to export and then select **Export**.



4. When the download is complete, navigate to the location of the downloaded file.

> **Note**

When you download a worksheet it should automatically save to your computer. However, if it doesn't then make sure that you save it before you open and edit it. Otherwise, you might get this error message: **Excel cannot open or save any more documents because there is not enough available memory or disk space.**

To fix the issue:

a. Open Excel and go to **File** > **Options** > **Trust Center Settings Center Settings** > **Protected View**.

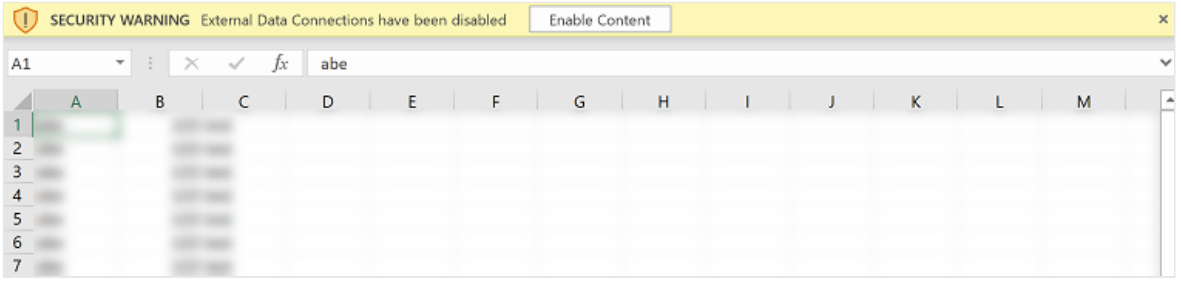
b. In **Protected View**, clear all three items.

c. Select **OK** > **OK**.

We still strongly recommend that you save and then open the data file rather than disabling protected view, which might put your computer at risk.

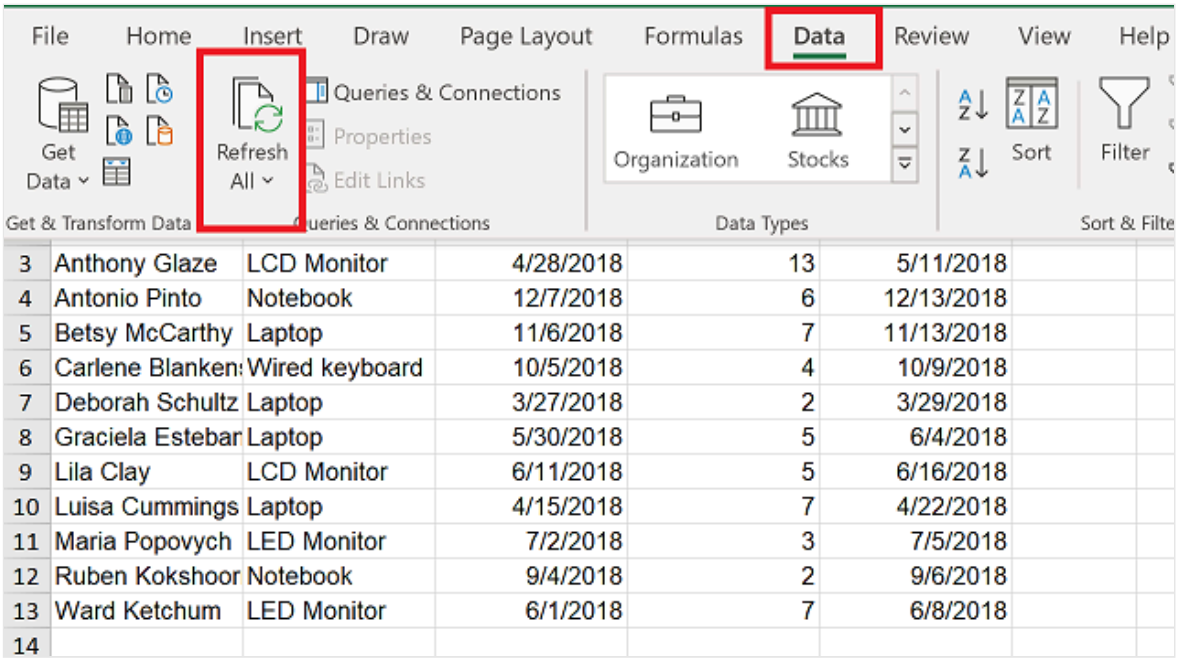
5. Open the saved Excel file.

6. If you see the security warning **External Data Connections have been disabled**, select **Enable Content**.



7. Go back to your app and update your app data.

8. To see your updates in the Excel dynamics worksheet, in Excel go to **Data** > **Refresh All**. If this doesn't work, see Refresh All doesn't work.



3	Anthony Glaze	LCD Monitor	4/28/2018	13	5/11/2018	
4	Antonio Pinto	Notebook	12/7/2018	6	12/13/2018	
5	Betsy McCarthy	Laptop	11/6/2018	7	11/13/2018	
6	Carlene Blanken	Wired keyboard	10/5/2018	4	10/9/2018	
7	Deborah Schultz	Laptop	3/27/2018	2	3/29/2018	
8	Graciela Estebar	Laptop	5/30/2018	5	6/4/2018	
9	Lila Clay	LCD Monitor	6/11/2018	5	6/16/2018	
10	Luisa Cummings	Laptop	4/15/2018	7	4/22/2018	
11	Maria Popovych	LED Monitor	7/2/2018	3	7/5/2018	
12	Ruben Kokshoor	Notebook	9/4/2018	2	9/6/2018	
13	Ward Ketchum	LED Monitor	6/1/2018	7	6/8/2018	
14						

Refresh All doesn't work

If selecting **Data** > **Refresh All** causes data to disappear with a blank workbook after export to a dynamic worksheet, see Troubleshoot: Refresh All doesn't work after exporting app data to a dynamic worksheet.

Feedback

Was this page helpful? Yes No

Provide product feedback

Export to an Excel static worksheet

Article • 02/15/2022

When you want to present information about the data in your app to an individual who doesn't have access to the app, or you have data that doesn't change often, consider exporting the app data to an Excel static worksheet.

There are two types of static worksheets that you can export.

- **Static worksheet**: Exports all the pages in the current view.
- **Static worksheet (Page only)**: Exports the data from the current page only.

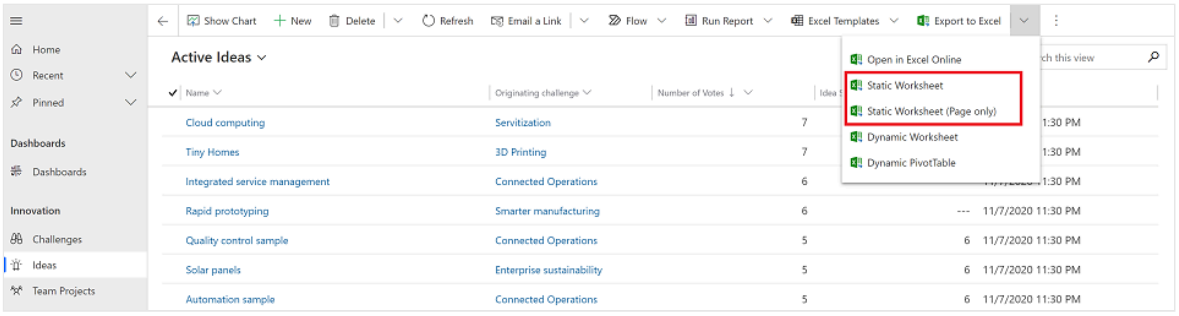
A few things to note:

- You may have the option to export data to an Excel static worksheet in all row types. However, in some cases the format might be old, or the data might not be filtered by what you see in the app.
- You can export up to 100,000 rows at a time. By default, a model-driven app lists up to 50 rows per page.
- Once imported, you can email the static worksheet to anyone or store it in a shared file. Anyone who opens the file will see all the data in the file.
- You can't change the columns for a system view. You must either customize the view, which requires the System Administrator or System Customizer security role, or use Advanced Find to create your own view based on the current view.
- Currency values are exported to Excel as numbers. After you have completed the export, see Format numbers as currency.
- The date and time values that you see in the app show up only as Date when you export the file to Excel, but the cell actually shows both the date and time.

Export a static worksheet

1. To export data, on the left nav, select a table.

2. On the command bar select the **Export to Excel**, and then choose **Static worksheet** or **Static worksheet (Page only)**.

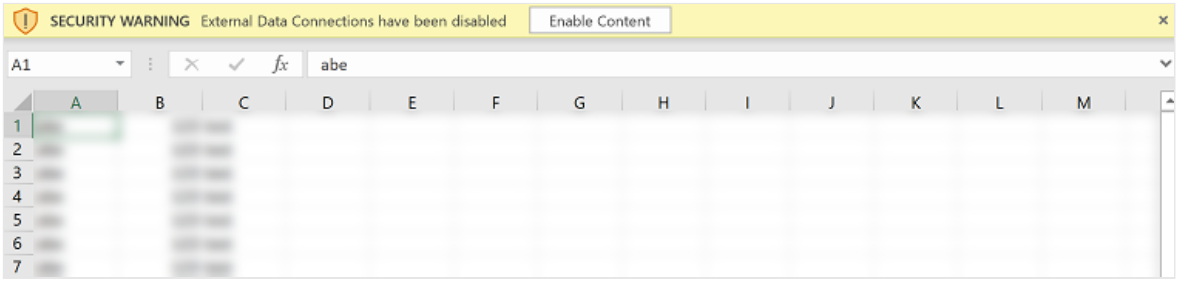


3. when the download is complete, navigate to the location of the downloaded file.

> **Note**
>
> When you download a worksheet it should automatically save to your computer. However, if it doesn't then make sure that you save it before you open and edit it. Otherwise, you might get this error message: **Excel cannot open or save any more documents because there is not enough available memory or disk space.**
>
> To fix the issue:
>
> a. Open Excel and go to **File** > **Options** > **Trust Center Settings Center Settings** > **Protected View**.
>
> b. In **Protected View**, clear all three items.
>
> c. Select **OK** > **OK**.
>
> We still strongly recommend that you save and then open the data file rather than disabling protected view, which might put your computer at risk.

4. Open the saved Excel file.

5. If you see the security warning **External Data Connections have been disabled**, select **Enable Content**.



By default, an exported worksheet includes the columns that are displayed in the list, using the same column order, sorting, and column widths.

Feedback

Was this page helpful? Yes No

Provide product feedback

Export to an Excel PivotTable

Article • 10/26/2022

You can export app data to an Office Excel PivotTable to see patterns and trends in data. An Excel PivotTable is a great way to summarize, analyze, explore, and present your app data. You can export up to 100,000 rows at a time.

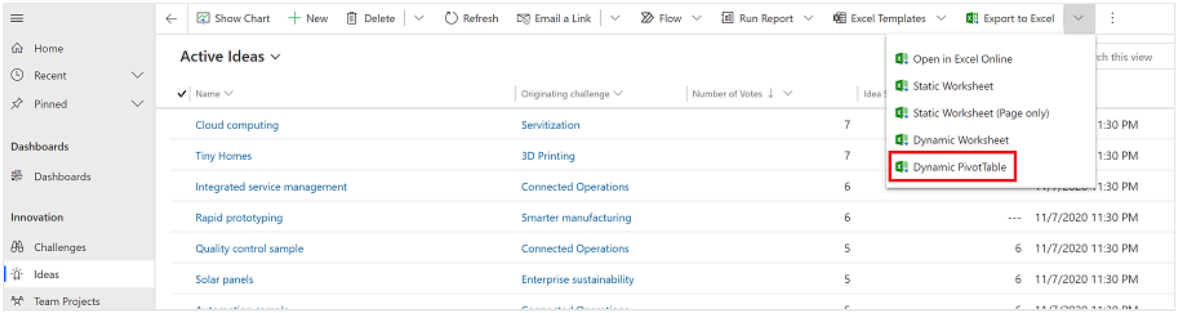
A few things to note:

- The option to export data to an Excel PivotTable isn't available in all row types. If you don't see the option, it's not available for that row.
- Currency values are exported to Excel as numbers. After you complete the export, see the Excel Help topic "Display numbers as currency" to format the data as currency.
- The date and time values that you see in the app show up only as Date when you export the file to Excel, but the cell actually shows both the date and time.

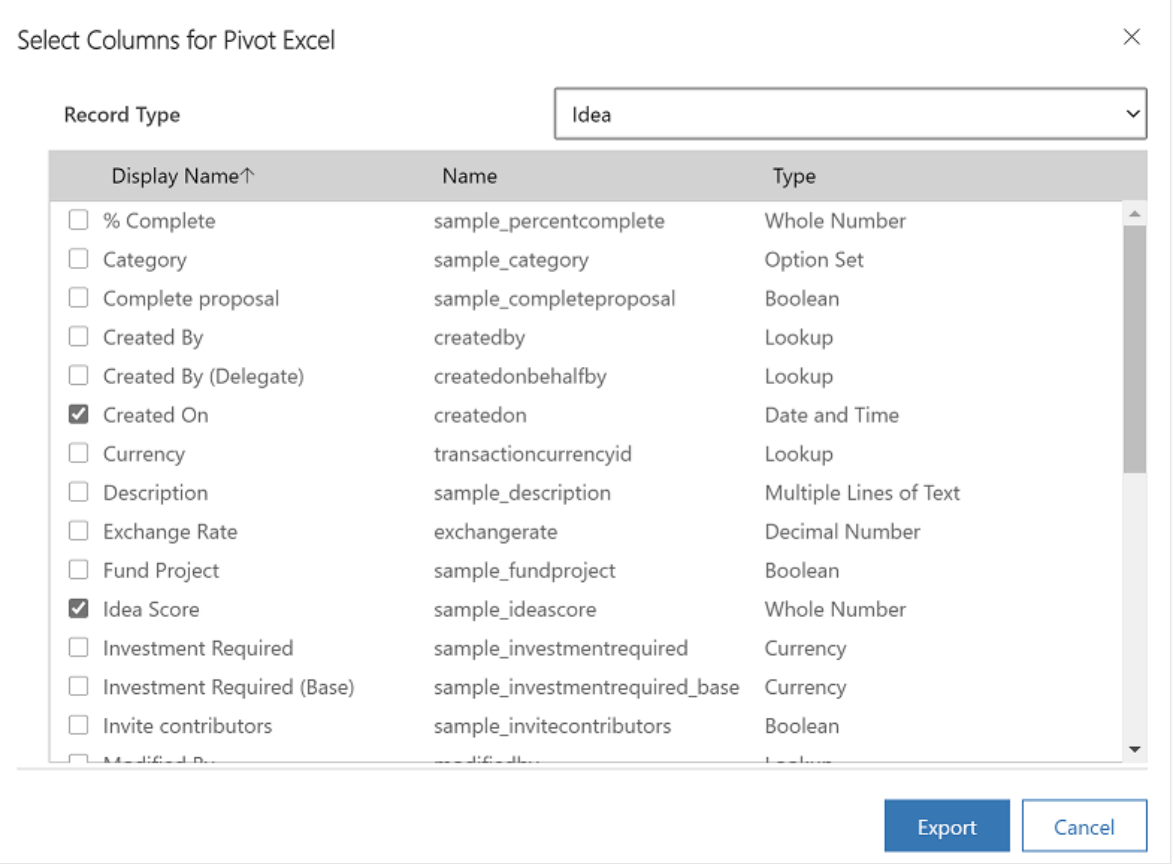
Export data to a PivotTable

1. On the left nav, select a table.

2. On the command bar select the **Export to Excel** menu, and then select **Dynamic PivotTable**.



3. In the **Select Columns for Pivot Excel** dialog box, select the column settings and then select **Export**.



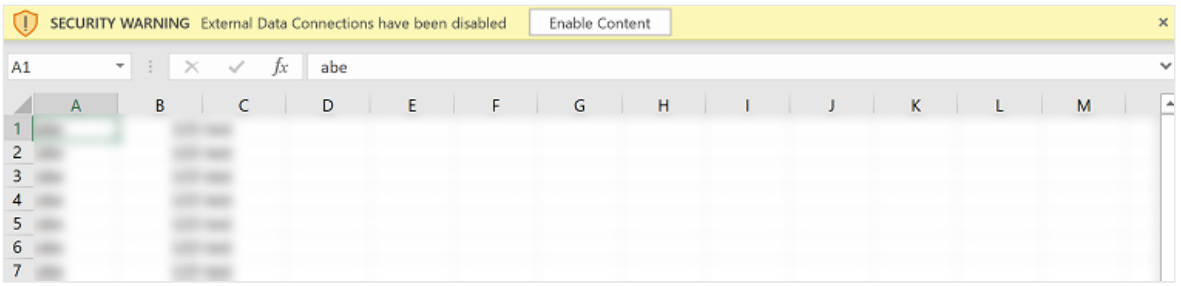
By default, the **PivotTable Column List** includes only columns that are displayed in the **Select Columns for Pivot Excel** list.

4. When the download is complete, navigate to the location of the downloaded file.

> **Note**
>
> When you download a worksheet it should automatically save to your computer. However, if it doesn't then make sure that you save it before you open and edit it. Otherwise, you might get this error message: **Excel cannot open or save any more documents because there is not enough available memory or disk space.**
>
> To fix the issue:
>
> a. Open Excel and go to **File** > **Options** > **Trust Center Settings Center Settings** > **Protected View**.
>
> b. In **Protected View**, clear all three items.
>
> c. Select **OK** > **OK**.
>
> We still strongly recommend that you save and then open the data file rather than disabling protected view, which might put your computer at risk.

5. Open the saved Excel file.

6. If you see the security warning **External Data Connections have been disabled**, select **Enable Content**.



> **Note**
>
> If don't see any data and you get an error, see Refresh all doesn't work.

7. To add a PivotTable, on the **Insert** tab, select **PivotTable**.

> **Note**
>
> You must complete this step to insert the PivotTable. The PivotTable is not created automatically.

8. Drag the columns from the PivotTable Column List to the PivotTable. For more information, see Overview of PivotTables and PivotCharts.

Feedback

Was this page helpful? Yes No

Provide product feedback

Merge duplicate rows

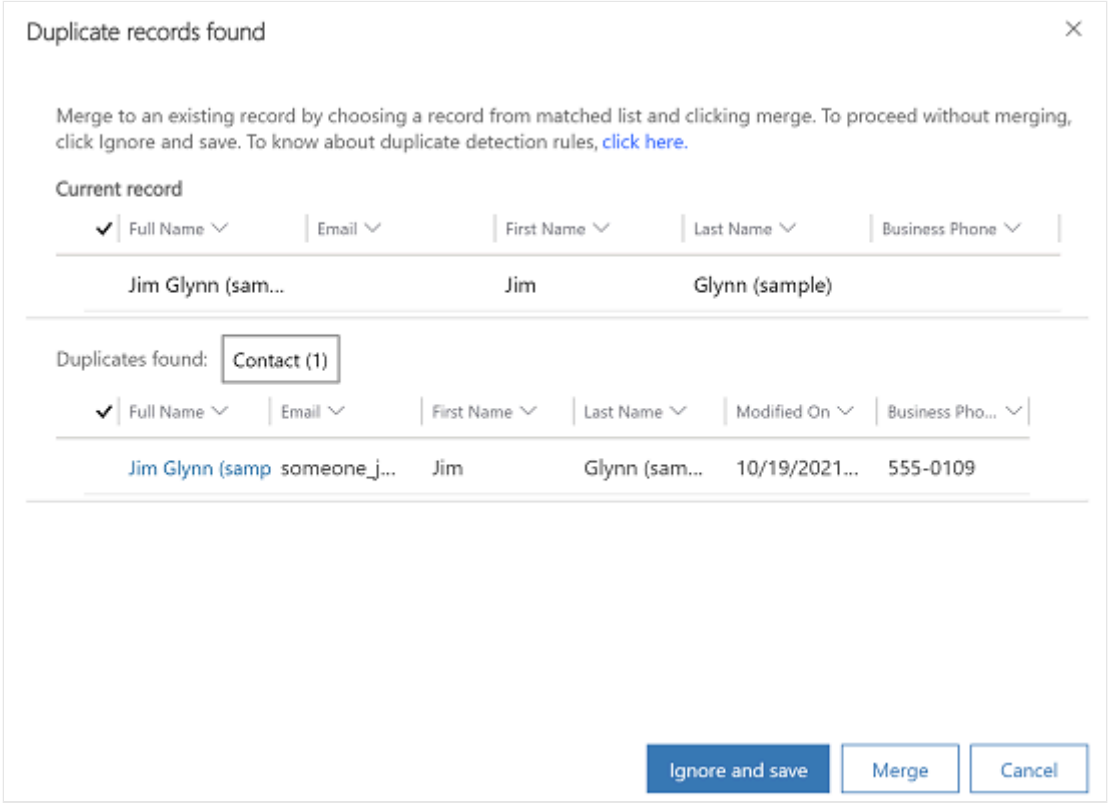
07/22/2025

Duplicate rows can creep into your data when you or others enter data manually or import data in bulk. Microsoft Dataverse helps you address potential duplicates by providing duplicate detection for active rows, such as accounts and contacts. When you merge a row, any related or child rows are also merged. Your administrator may also set up duplicate detection rules for other situations.

There are a few situations when duplicates might be found:

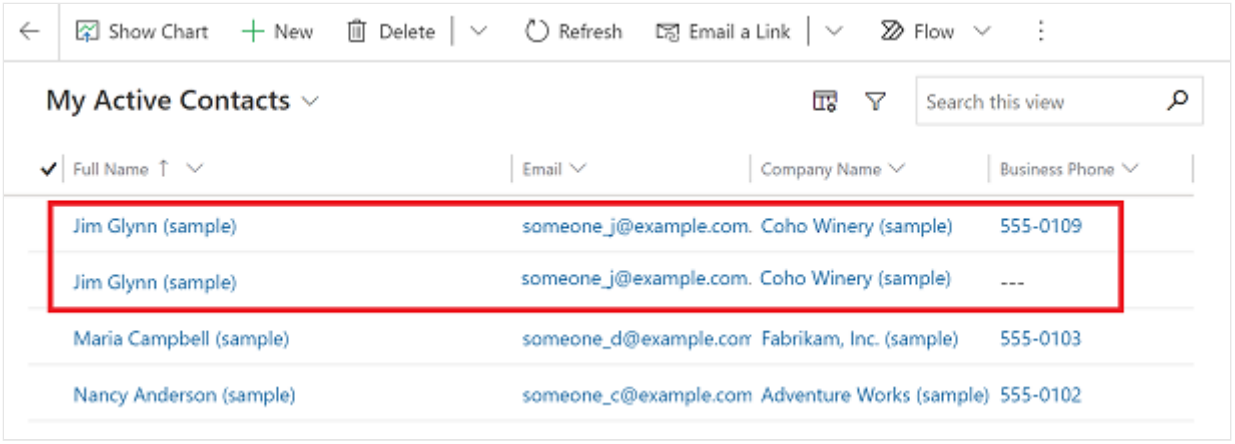
- When a row is created or updated.
- When you're using Dynamics 365 for Outlook and you go from offline to online.
- When you import data using the Import Data wizard.
- Duplicates aren't detected when you merge rows, save an activity as completed, or change the status of a row. For example, when activating or reactivating a row.

For example, let's say you enter a contact row, Jim Glynn, along with a mobile phone number. The duplicate detection rule discovers that you already have a similar row, and displays this dialog box.



You're not sure if this is a different person that has the same name as an existing contact, so you select **Ignore And Save**.

Next, you go to the **My Active Contacts** list and see that now you have two rows with the same name. After reviewing the rows, you determine that they're duplicates that need to be merged.



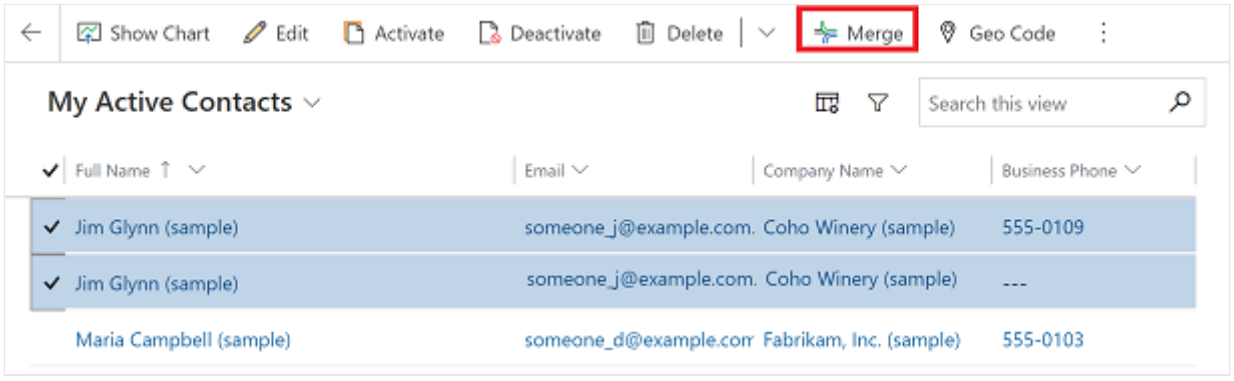
Dataverse includes duplicate detection rules for accounts and contacts. These rules are automatically turned on, so you don't have to do anything to set up duplicate detection for these row types.

> **Note**
>
> If available on your system, you may also be able to check for duplicates of other row types, in addition to contacts and accounts. Check with your system administrator. Find your administrator or support person

> **Note**
>
> To modify the columns that are visible for the current record in the duplicate detection dialog, administrators can modify the **Advanced Find View** for that entity.

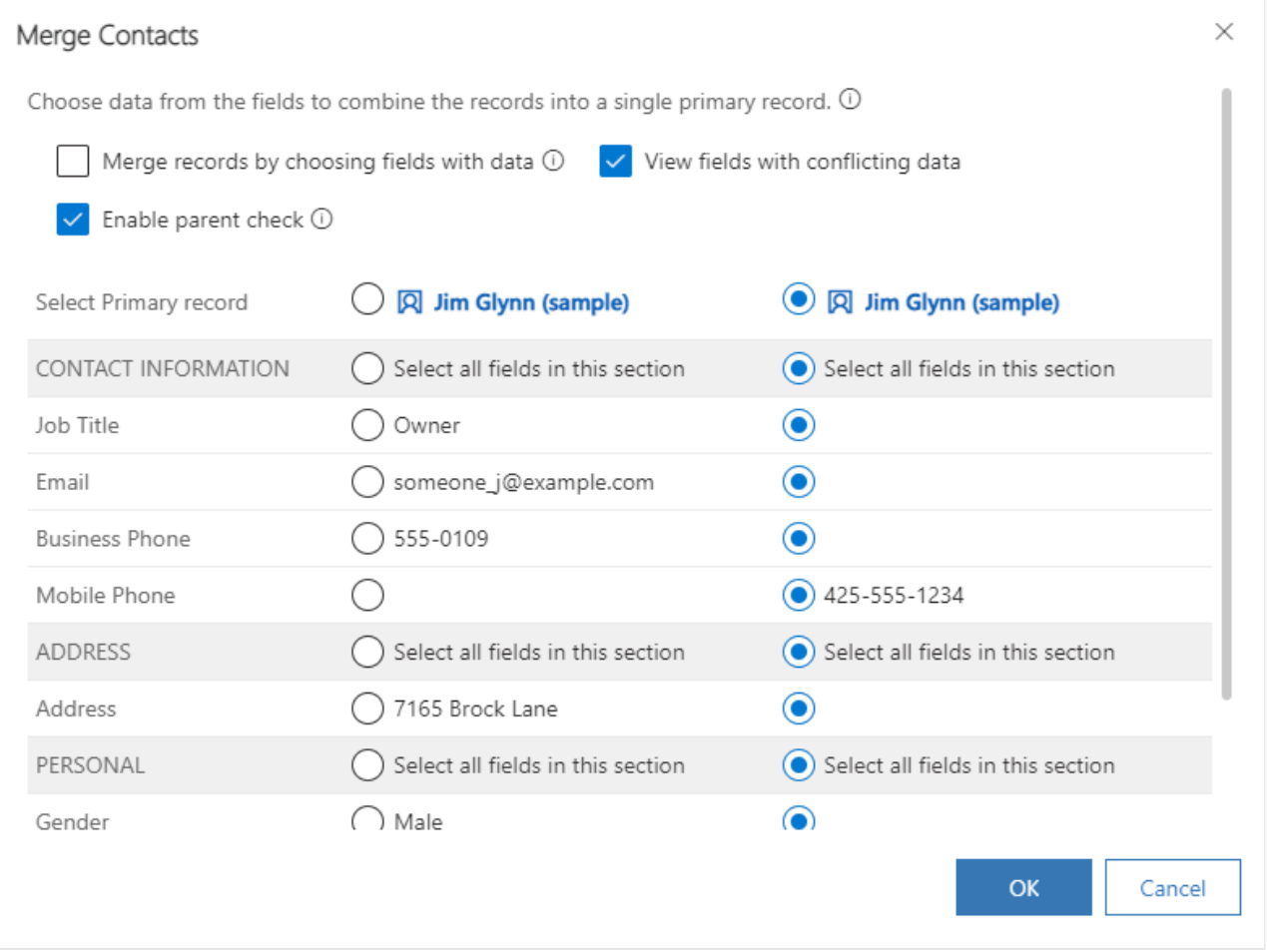
How to merge duplicate rows

1. Select the duplicate rows from the view page in your model-driven app, and then select **Merge** in the command bar at the top of the page.



2. In the **Merge Contacts** dialog box, select the primary row (the one you want to keep) and then select any columns in the new row that you want to merge into the primary row. Data in these columns may override the existing data in the primary row. Select **OK**.

> **Note**
>
> If you run into issues during the merge process, uncheck **Enable parent check**. If files are deleted during the merge process, you can skip the parent check. This allows your merge to continue even if someone deletes a row from the set while the job is running in the background. For more information, see Asynchronous processing of cascading transactions.



Limitations

- You can merge rows only from Account, Contact, and Lead tables. Other out-of-box Dataverse tables and custom tables aren't supported.
- You can only merge two records at a time.
- File and image columns can't be previewed. They show as base64 text. Go to the original records to view the file or image.

- If a column or control for the column matches any of the following conditions, it doesn't appear in the merge dialog:
  - The column or containing section is hidden by default in column or section properties. Hidden columns and sections can be forced to appear using Client API functions, but they still aren't be eligible for merging.
  - The column's `ValidForUpdate` property is `False`.
  - The column is **Choice** or **Choices** type and it has either a parent choice or child choice column.
  - The column is `parentaccountid` on the Account table.
  - The column is `parentcustomerid` on the Contact table.
  - The column is **Owner** type (example: `ownerid`).
  - The control for the column does not have a class property.
  - The control for the column is **Quick Form Collection Control** or **Reference Panel Quick Form Collection Control**.
- Merging records with a SharePoint documents library merges only the records, not the documents libraries. Learn more in Collaborate using SharePoint.

Work with reports

Article • 02/27/2024

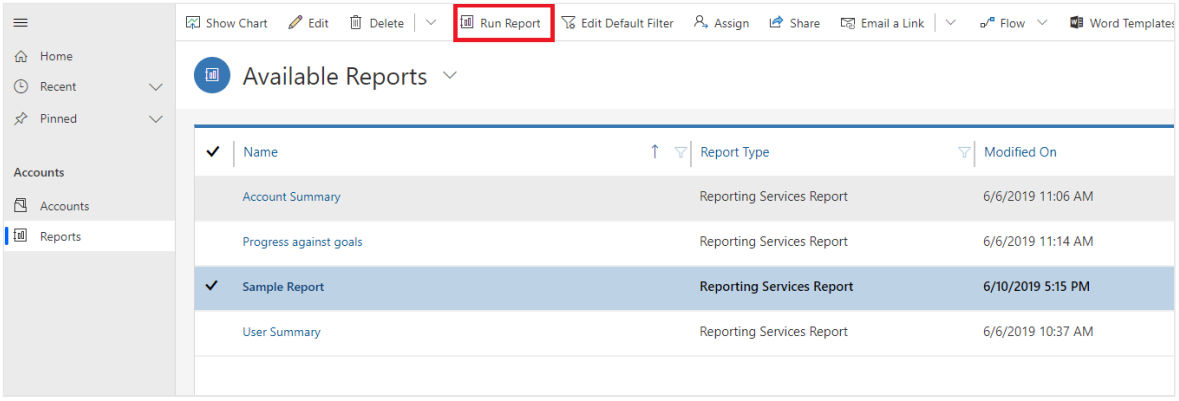
Reports help you monitor your progress towards your business goals by helping you see how you're doing. You can also track trends - which can give you an advantage over your competitors.

The reporting features need to be enabled for your model-driven app for you to see the reports area on the left navigation pane. For more information, see Add reporting features to your model-driven app.

Run a report

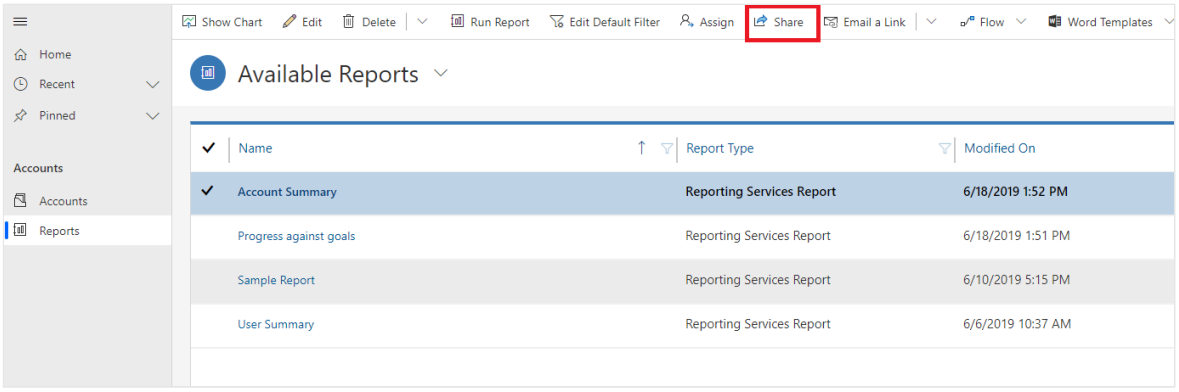
1. From the left navigation pane, select the reports area.
2. Choose the report you want > **Run Report**.

> **Note**
>
> In **Report Viewer** dialog box, you can leave the search criteria as is, or change it as needed.



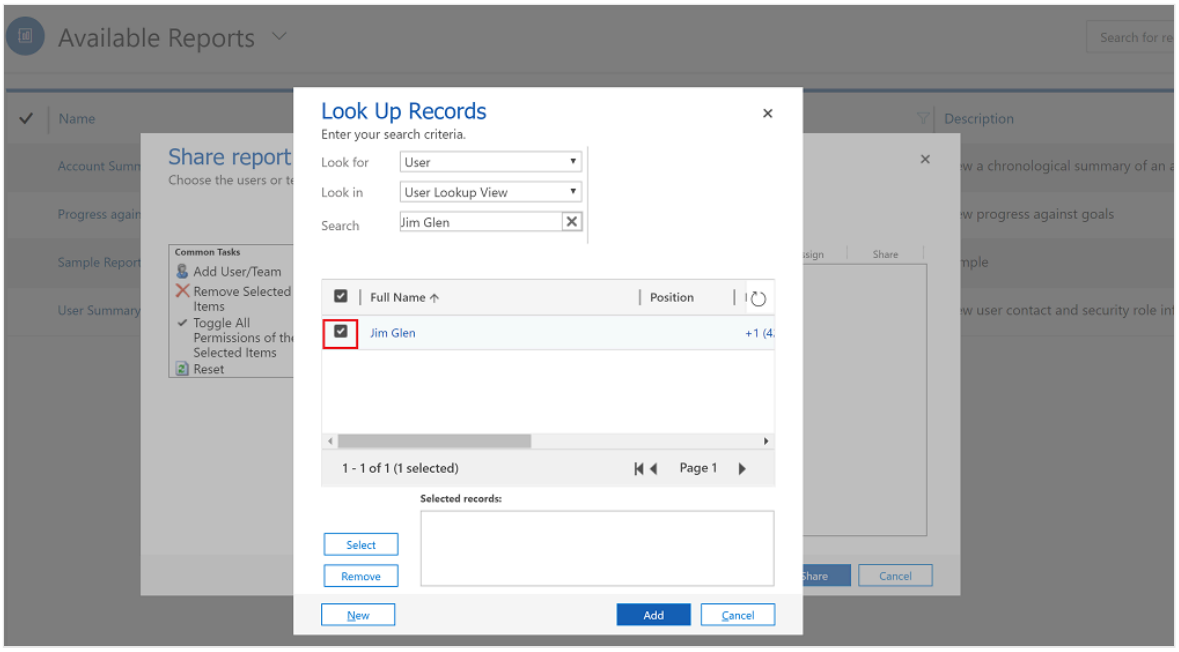
Share a report with other users or teams

1. From the left navigation pane, select the reports area.
2. In the list of reports, select the report you want to share.
3. On the command bar, select **Share**.

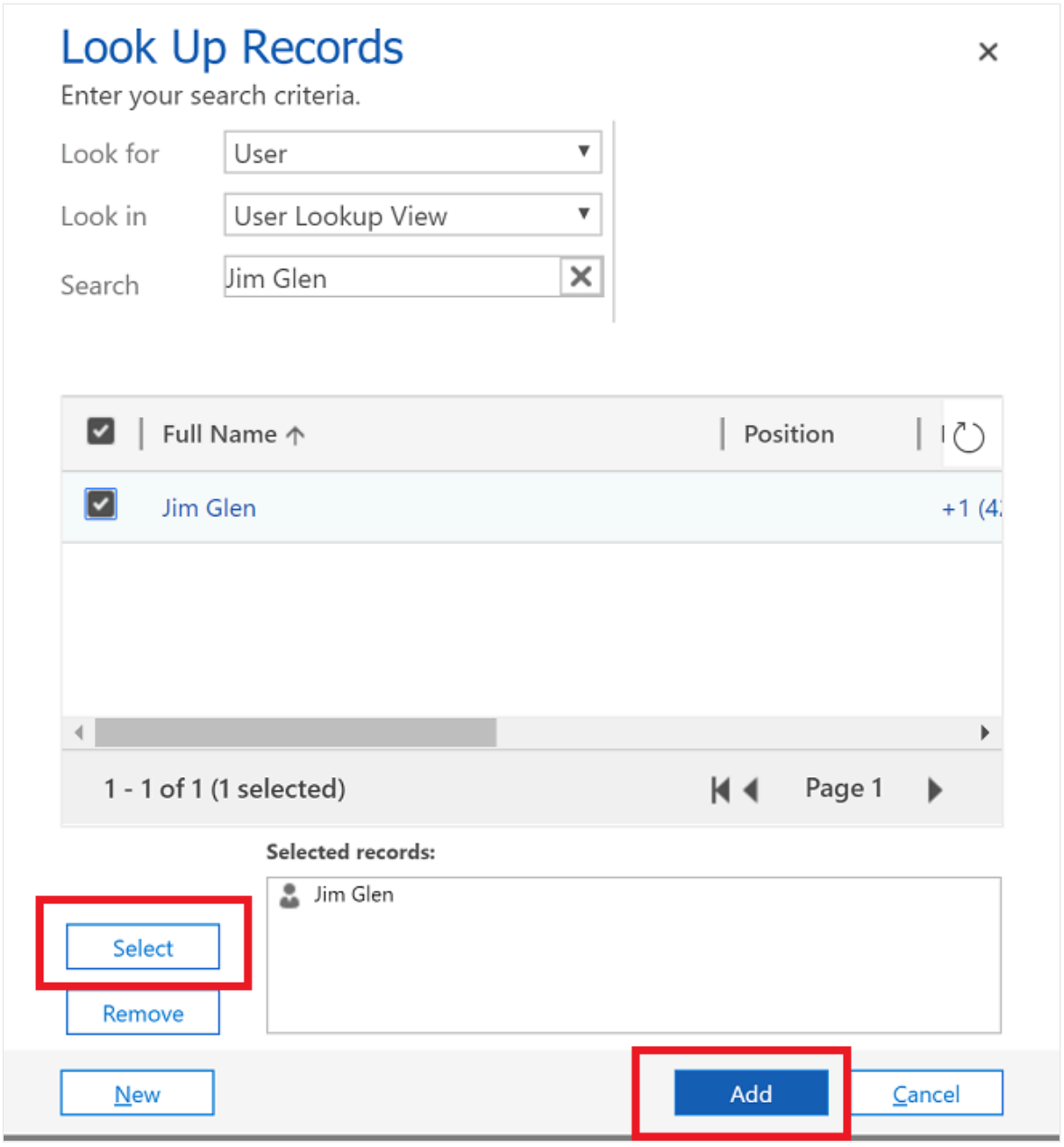


4. On the **Share Report** dialog box, select **Add User/Team**.

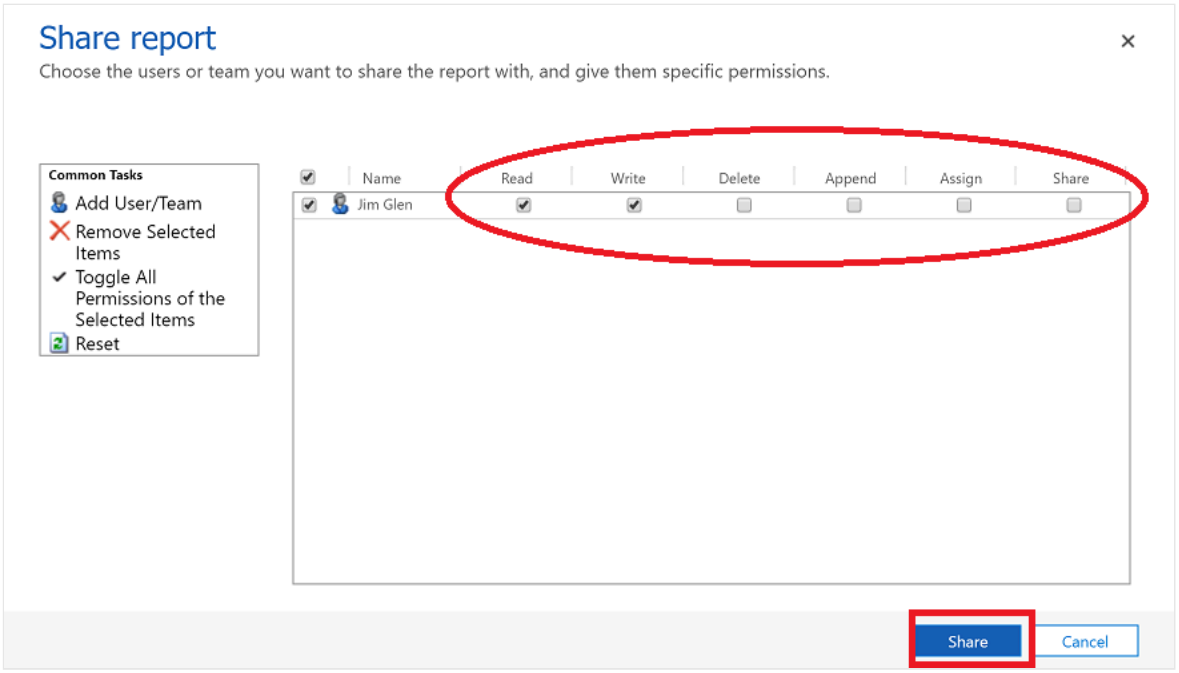
5. In the **Look Up Rows** dialog box, find the users or team row that you want to share the report with, and select the check box next to the row.



6. Choose **Select** to add the user or team row to the **Selected rows** box and then select **Add**.



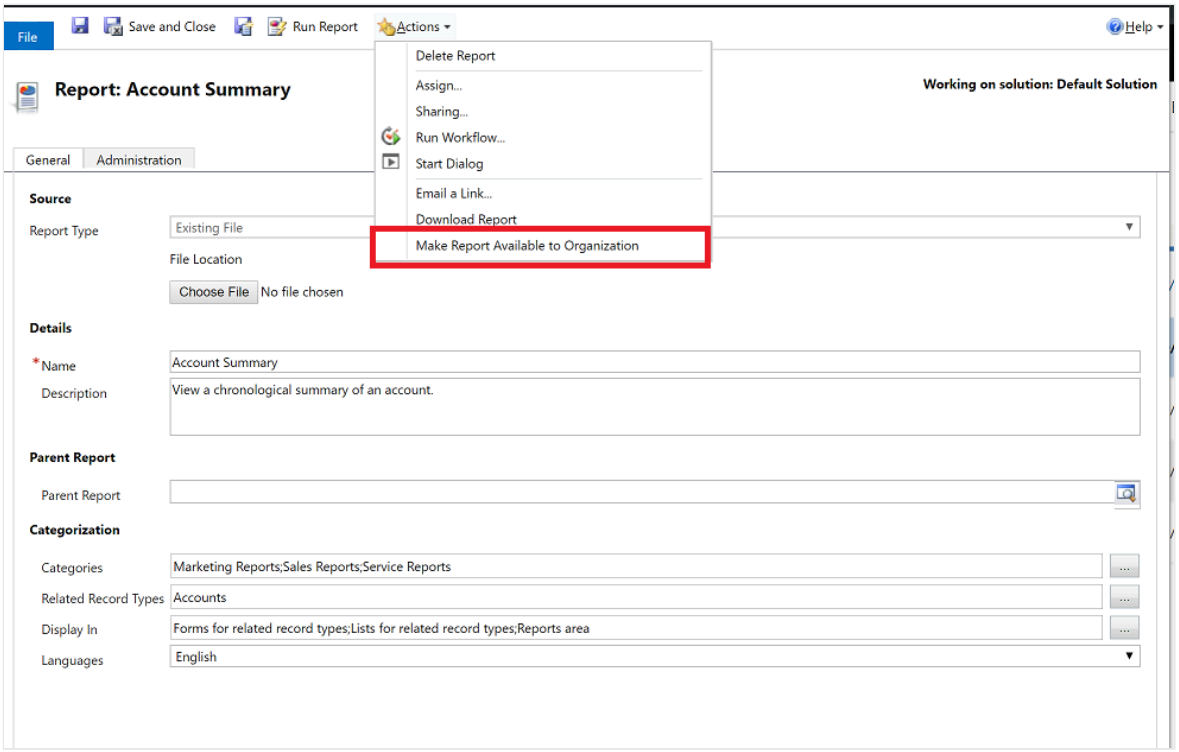
7. In the **Share Report** dialog box, select the type of share access that you want. The available permissions are: Read, Write, Delete, Append, Assign, or Share. This will add the user or team row to the **Selected rows** box.



Share a report with your organization (for admins)

If the report would be useful for all users, make it available to the organization.

1. From the left navigation pane, select the reports area.
2. In the list of reports, select the report you want to share.
3. On the command bar, select **Edit**.
4. On the **Actions** menu, select **Make Report Available to Organization**.



Download a report

1. From the left navigation pane, select the reports area.
2. In the list of reports, select the report you want to share.
3. On the command bar, select **Edit**.
4. On the **Actions** menu, select **Download Report**.
   The RDL file contains the fetchXML that the report is based on.
5. Open the report once the download is complete.

Known issue

Reports rely on a hybrid experience called out in Enable hybrid experience which are not supported in mobile or Teams.

See Also

Create a report using the Report Wizard

Add a existing report

Edit report filter

Troubleshoot problems with data not displaying in a report

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a report using the Report Wizard

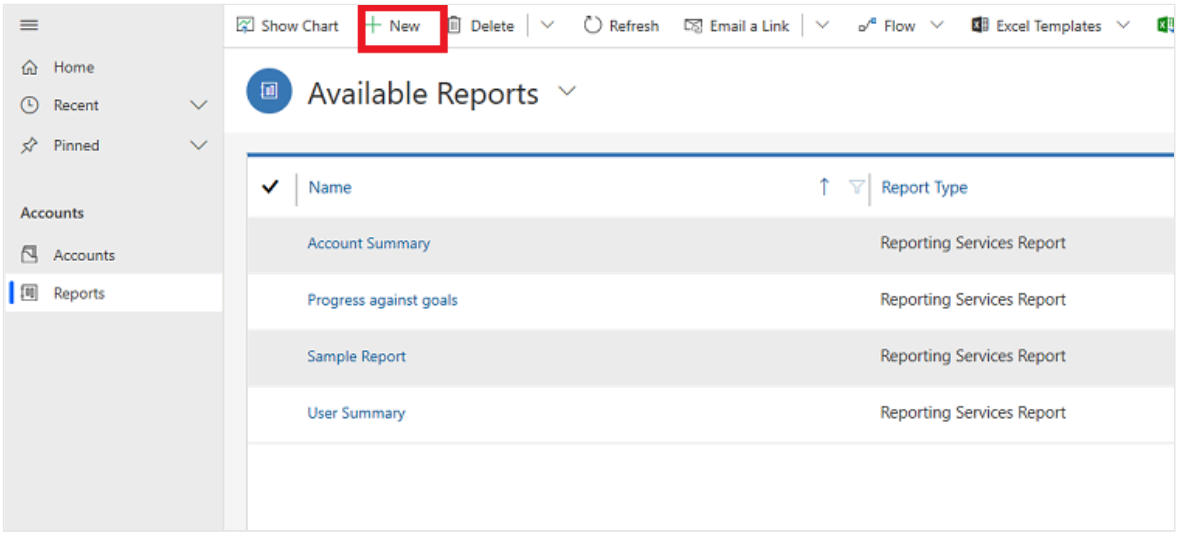
Article • 11/17/2023

Use the Report Wizard to create reports with charts and tables that allow you to easily analyze your data.

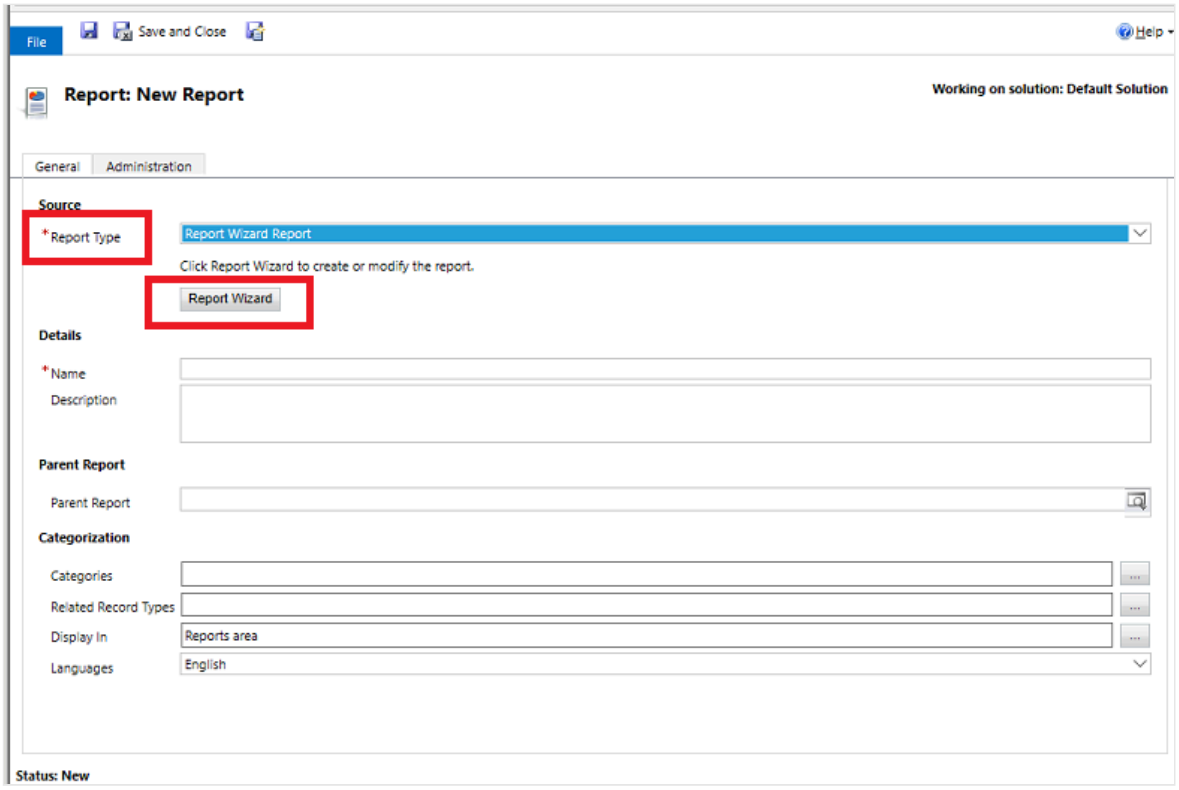
All reports that are created using the Report Wizard are Fetch-based reports. All reports generated with the Report Wizard print in landscape mode.

Create a new report

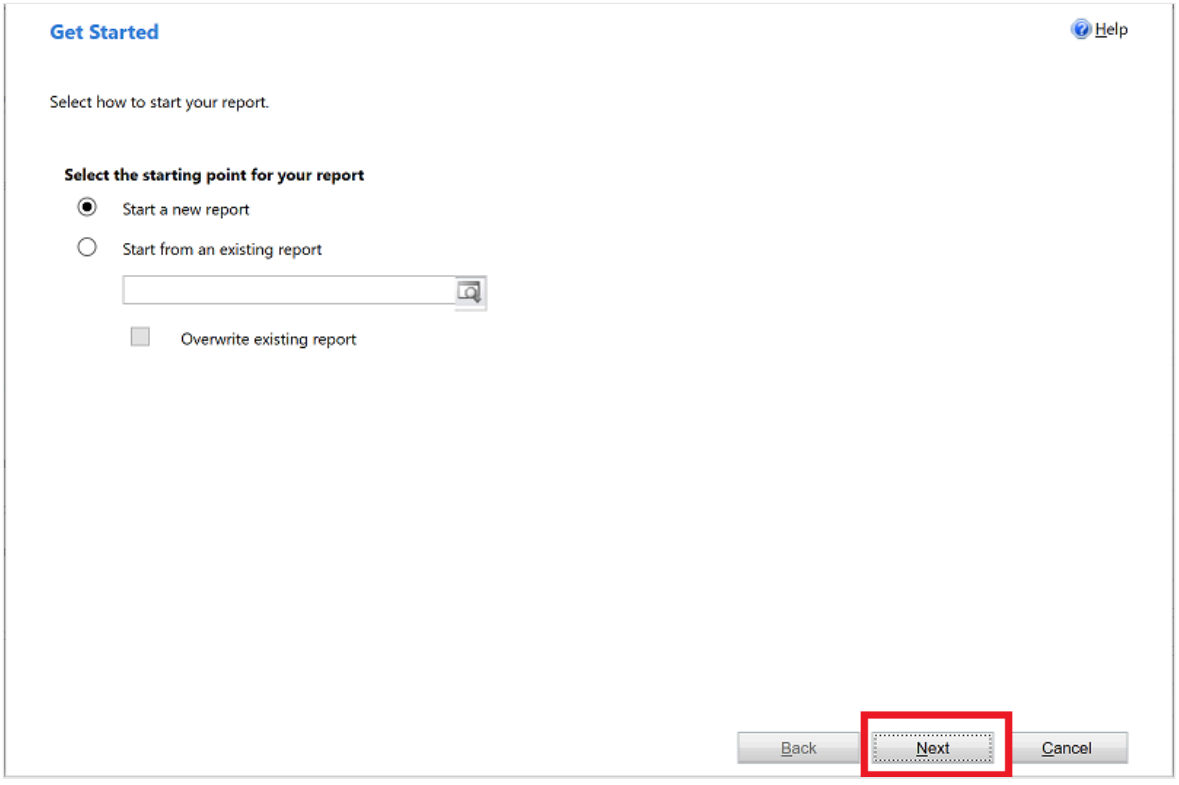
1. From the left navigation pane, select the reports area.
2. On the command bar, select **New**.



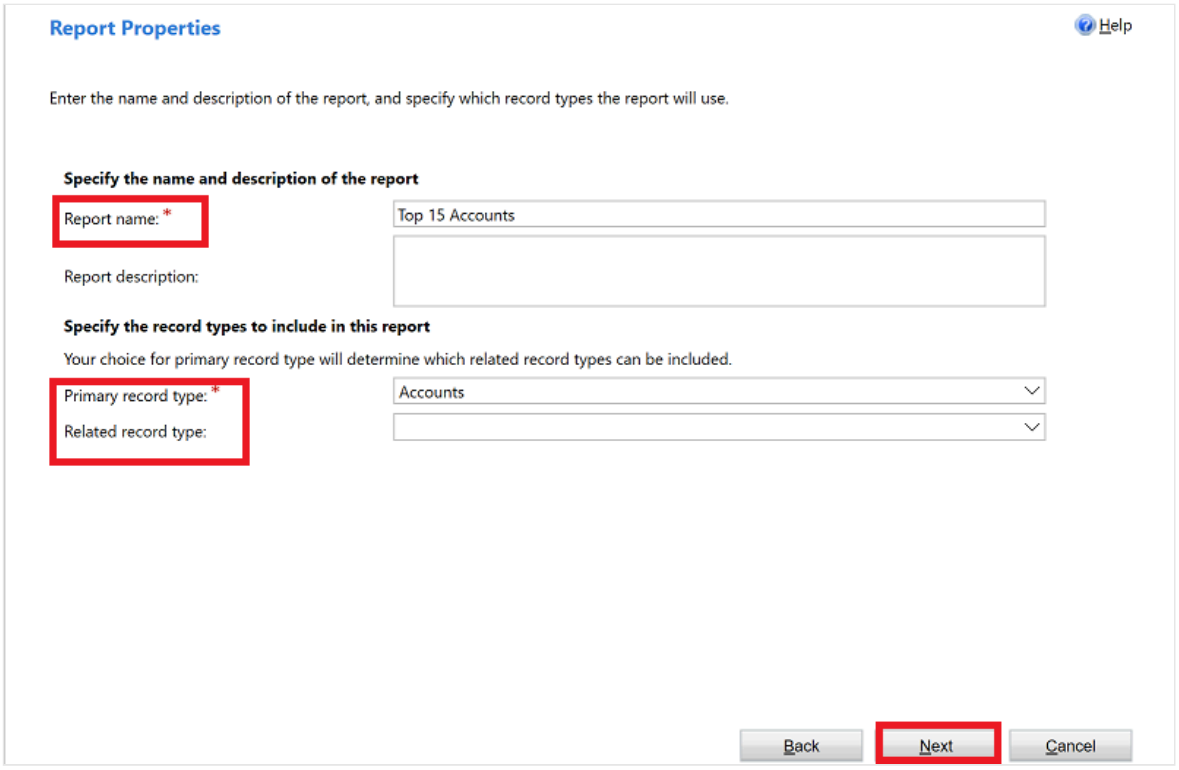
3. A **Report:New Report** screen appears. For **Report Type** leave the default selection to, **Report Wizard Report** and select the **Report Wizard** button.



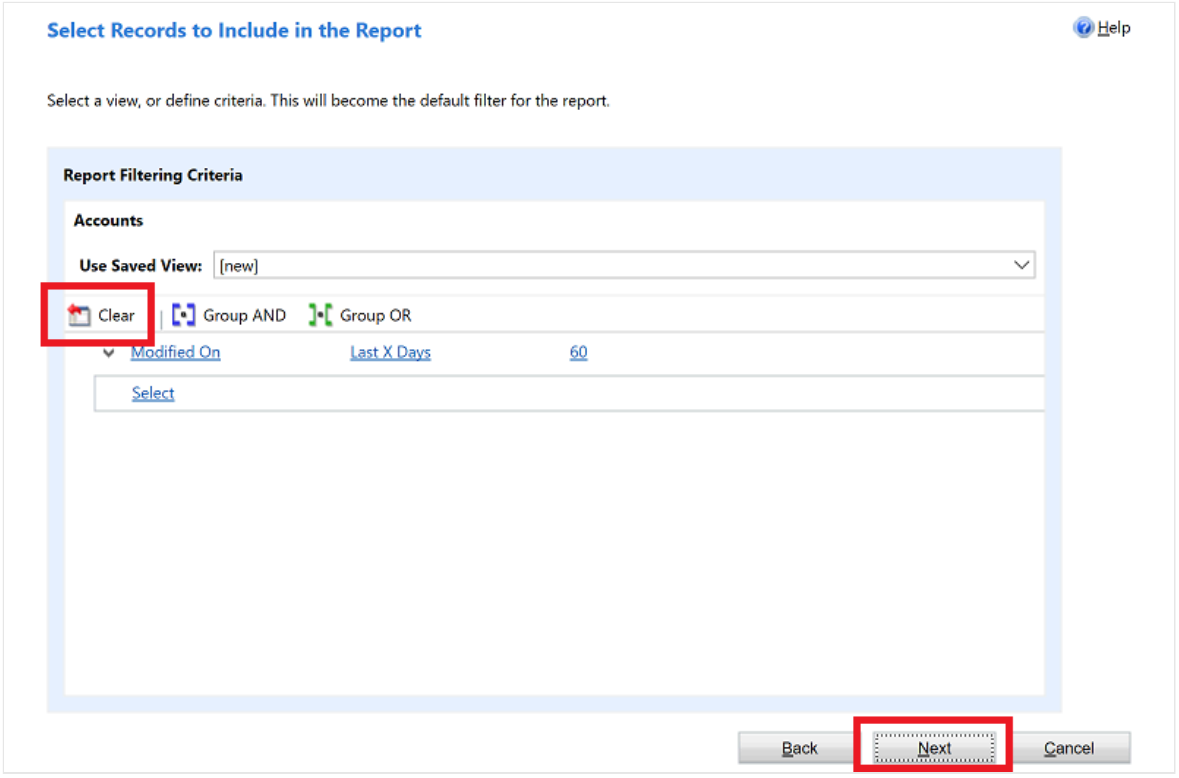
4. In the next screen, leave the default selections and then select **Next**.



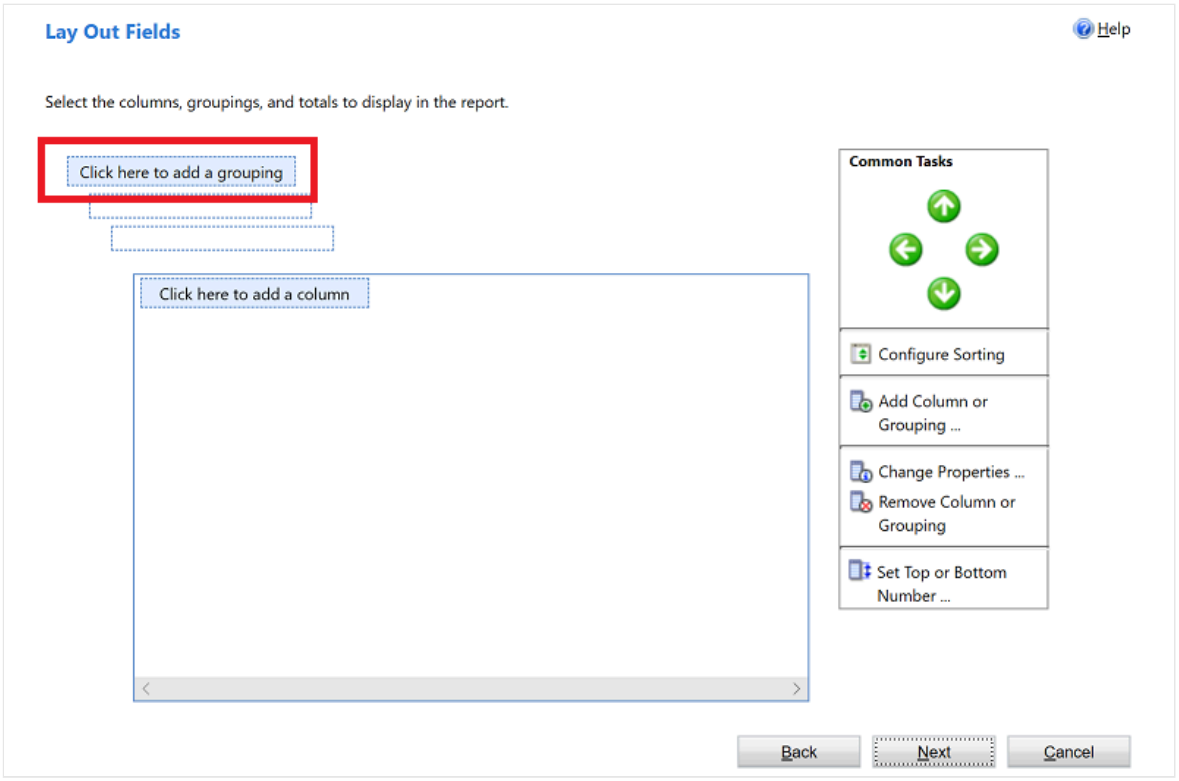
5. On the **Report Properties** screen, enter a name for the report and then choose the row to include in the report and then select **Next**.



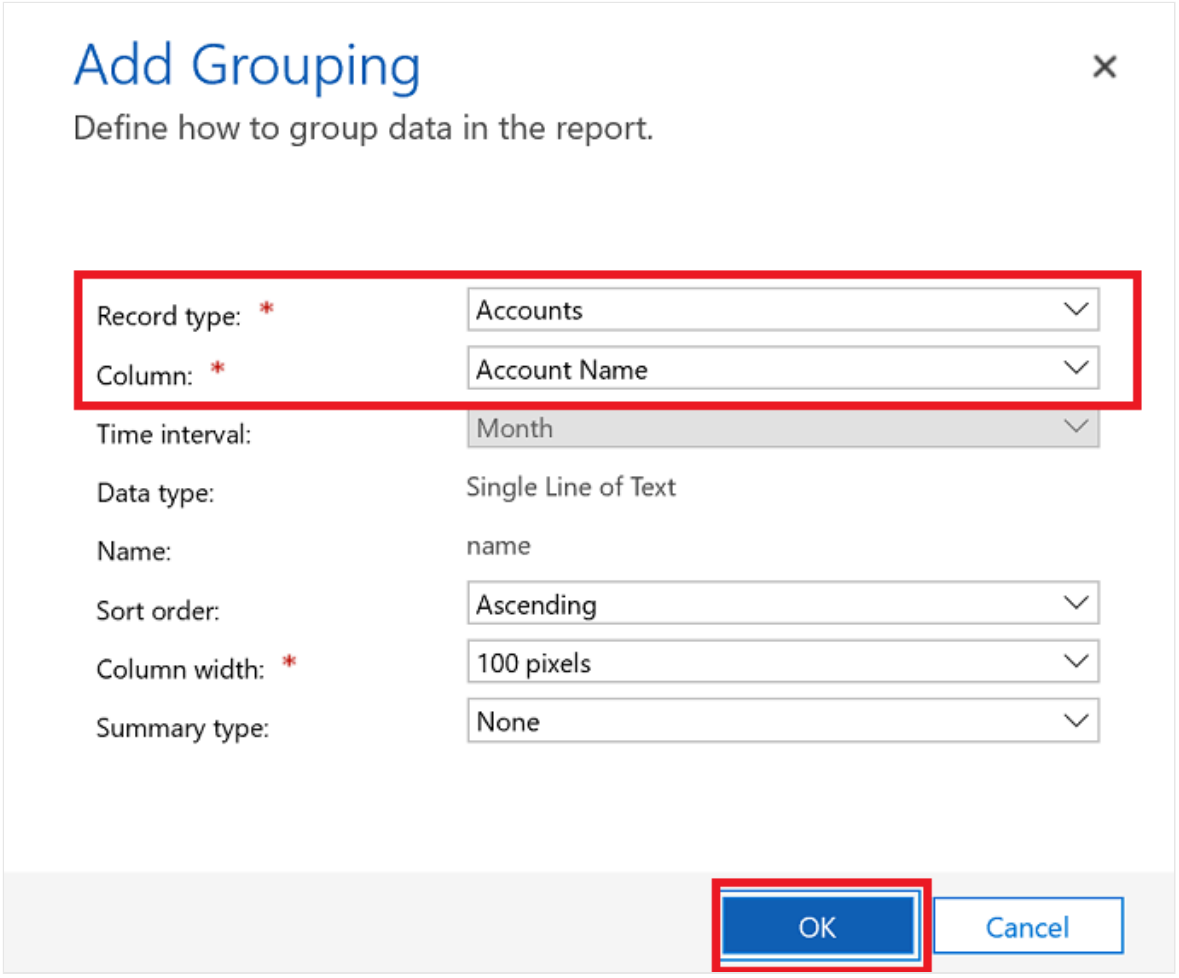
6. On the **Select Rows to Include in the Report** screen, choose the filters to determine which rows are included in your report. For example, if you only want to see results for rows modified in the last 60 days, you can set that filter in this screen. If you don't want the data filtered, select **Clear**.



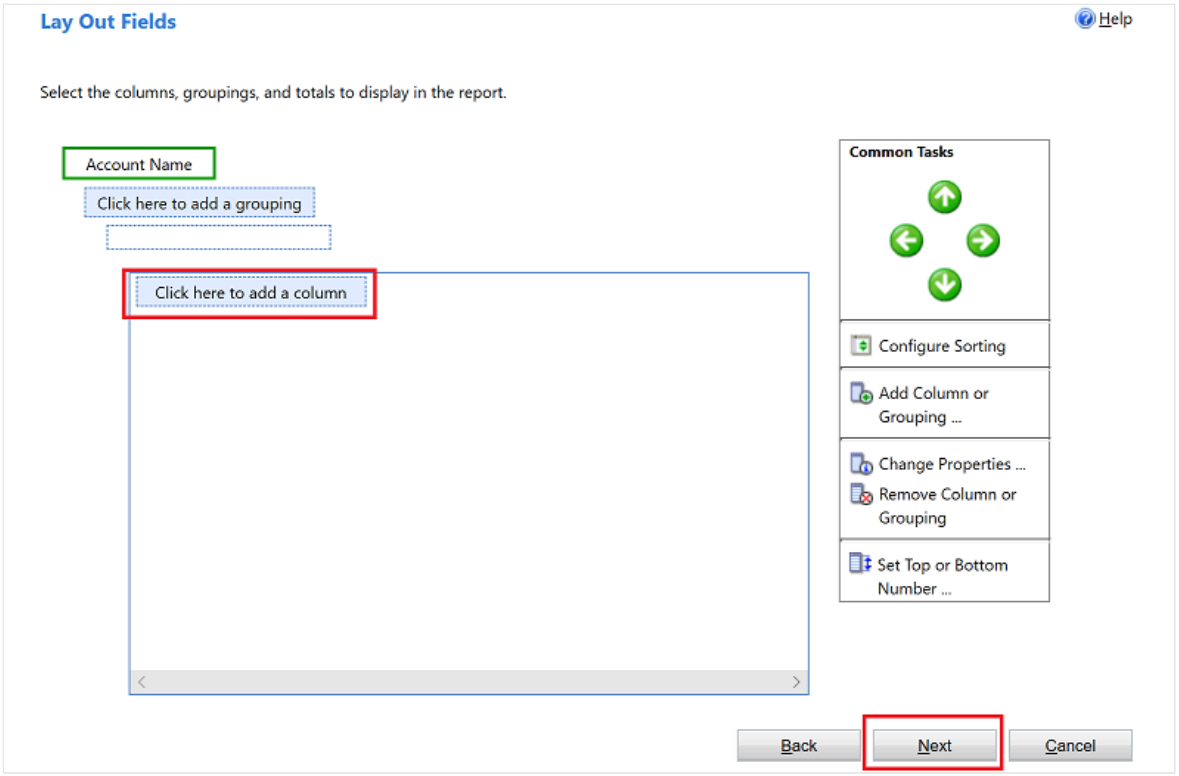
7. On the **Lay out Columns** screen, choose the layout of your report. Select **Click here to add a grouping** and choose how you want your data grouped.



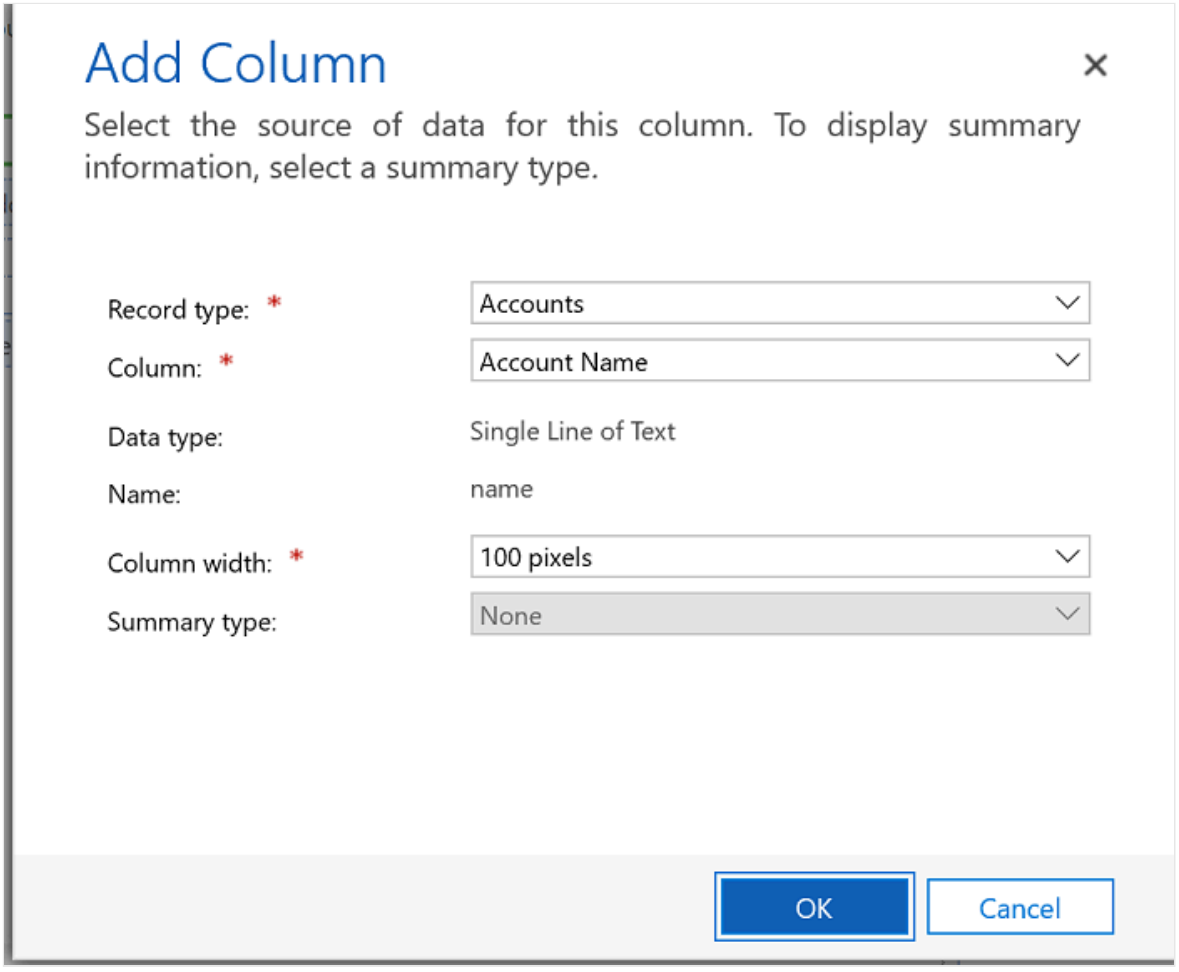
8. Select the **Row type** and the **Column** for the data you want to have grouped in the report. When you are done with your selections, select **OK**.



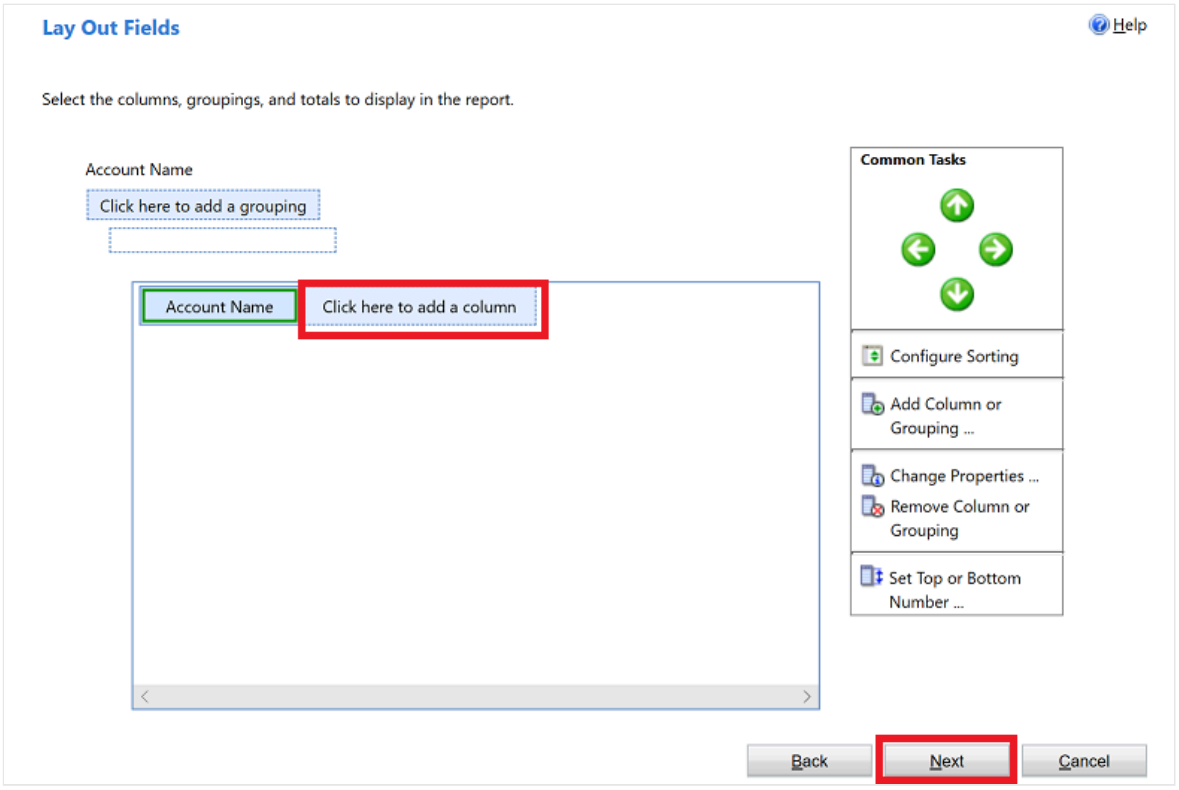
9. Select **Click here to add a column** to columns of data related to the row type you chose in the previous step.



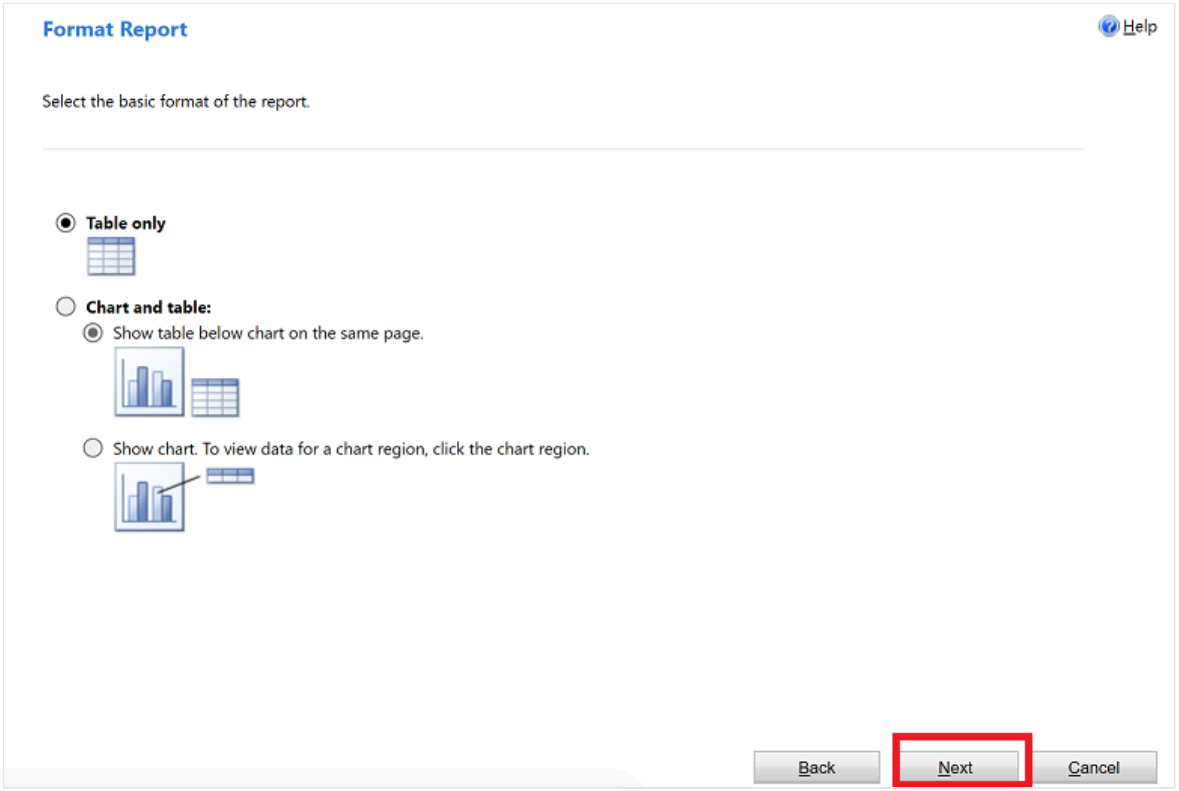
10. On the **Add Column** screen choose the data you want to have displayed for the column and then select **OK**.



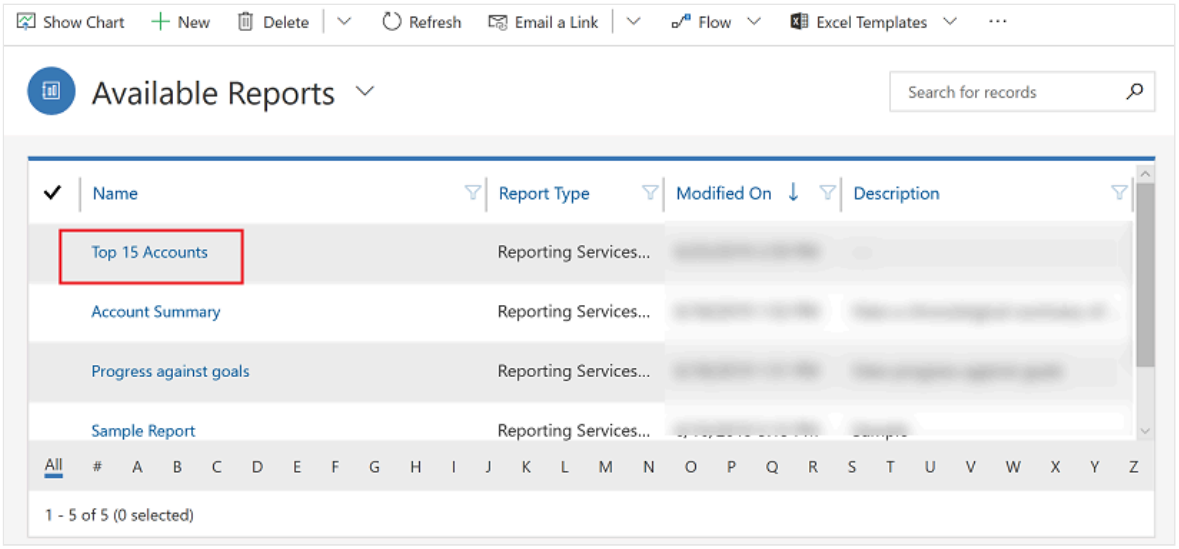
11. Repeat the previous step for any more columns that you want to add. When you are done, on the **Lay Out Columns** screen, select **Next**.



12. On the **Format Report** screen, choose how to format your report and then select **Next**.



13. Review the summary of your report and select **Next** and then select **Finish**. You can now see this report in the list of report in the system.



> **Note**
>
> The reports that are querying notes, may display in HTML-formatted text instead of plain text. This is because the rich-text editor is enabled by default. To workaround this issue, do one of the following:
>
> - Disable the rich-text editor. For more information, see Rich text editor control in Power Apps. When you disable the rich-text editor, new notes that are created will not then have the HTML tags. The existing notes that were created when the rich-text editor was enabled will still show in HTML-formatted text.
> - Update the report definition to support formatted text. For more information, see Importing HTML into a Report.
>
> Note that multi-select picklist fields are not fully supported in the Report Wizard. Multi-select picklist fields can't be added as a grouping or a column to the report.

See Also

Work with reports

Add a existing report

Edit report filter

Troubleshoot problems with data not displaying in a report

Feedback

Was this page helpful? Yes No

Provide product feedback

Add a report from outside Power Apps

Article • 09/08/2022

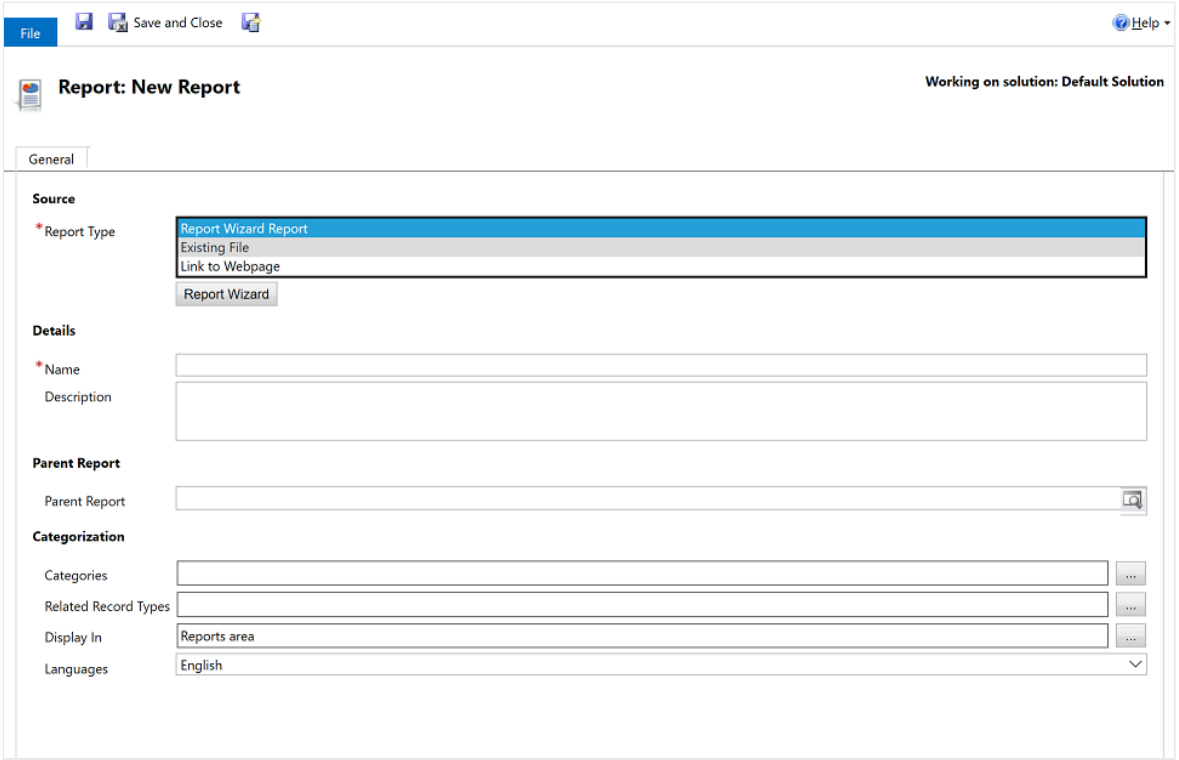
If you've created a custom report outside of the system, you can easily add it to Power Apps.

For information about how to create a custom report, see Reporting and Analytics Guide.

1. From the left navigation pane, select the reports area.
2. On the command bar, select **New**.

   Add a file created in another application

   a. In the **Source** section, in the **Report Type** box, select **Existing File**.



   b. In the **File Location** box, enter the path and file name of the file to add, or choose **Browse** to locate the file.

      You can upload many other file types such as an Excel file but for this to run like a SQL Server Reporting Services report or Report Wizard created report, the file needs to be an .RDL file. For more information, see Report writing environment using SQL Server Data Tools.

      > ⓘ Note

> The maximum size of the report file is determined by the **Maximum file size for attachments** setting in the Email settings environment setting. However, a report definition file (.RDL) can't be more than 4 MB in size, even if the maximum file size for attachments setting is larger than 4 MB.

-OR

Add a link to a webpage

a. In the **Source** section, in the **Report Type** box, select **Link to Webpage**.

b. In the **Webpage URL** box, enter the URL of the webpage.

3. Specify the properties for the report.

   a. In the **Details** section, specify a meaningful name and description for the report.

   b. The **Parent Report** text box displays the parent report of the current report, if one exists.

   c. **Categories**. Choose the **Select or change the values for this column** button, and then specify the categories to include in this report.

   d. **Related Row Types**. To have the report appear in the Reports list on a page for specific row types, choose the **Select or change the values for this column** button, and then select row types.

   e. **Display In**. To specify where reports should be visible, choose the **Select or change the values for this column** button, and then select one or more of the options.

      If no values are selected, the report won't be visible to end users.

4. Choose **Save** or **Save and Close**.

See Also

Work with reports

Create a report using the Report Wizard

Edit report filter

Troubleshoot problems with data not displaying in a report

Feedback

Was this page helpful? Yes No

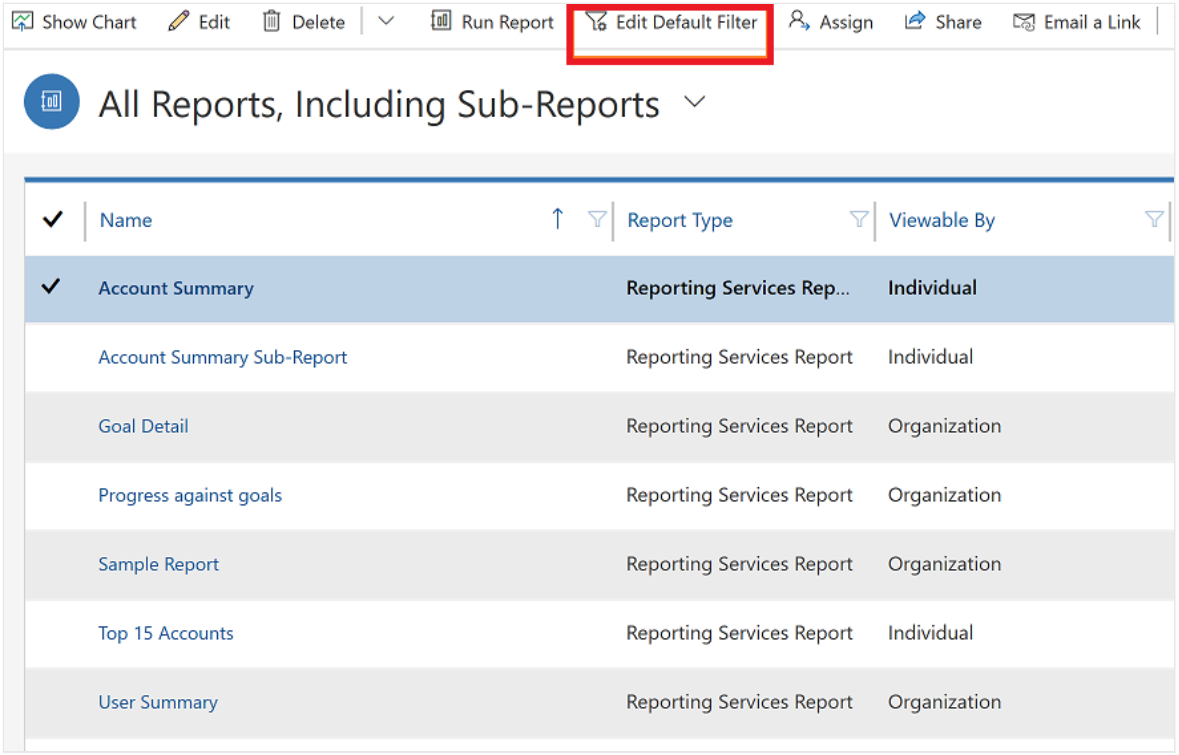
Provide product feedback

Edit the default filter of a report

Article • 12/06/2022

When a report is a SQL Server Reporting Services report, is enabled for prefiltering, and has a default filter, you can change the default filter to display the data you expect to see in the report. This filter is used each time any user runs the report.

1. From the left navigation pane, select the reports area
2. Choose a report and on the command bar, select **Edit Default Filter**.



3. Modify the filter criteria.

   The criteria are grouped by row types that you can use in the filter, such as **Accounts** or **Contacts**.

To edit an existing row

a. Select the query relational operator and select an operator, or Select the underlined value and enter a new value.

b. Select the query relational operator, and select an operator.

To add a criteria row:

a. Select **Select**, and specify the column to filter on.

b. Select the query relational operator, and select an operator.

c. Select **Enter Value**, and enter a value to filter on. For some values, you can select the **Select or change the values for this column** button  to open the **Select Values** dialog box and select the value you want.

To group criteria

You must select two or more rows for the same row type. However, rows with column values from different row types, such as **Account** and **Contact** row types, cannot be grouped.

a. For each row you want to group, in detailed mode, select the **Options menu** button for that row, and then choose **Select Row**.

b. On the Filter toolbar, select **Group AND** or **Group OR**.

c. To remove a row from a group, select the **Options menu** button for that row, and then select **Delete**.

d. To select a group, select the **Options menu** button for that group, and then select **Select Group**.

e. To add a criteria clause to a group, select the **Options menu** button for that group, select **Add Clause**, and then select the column, query relational operator, and value.

f. To unselect a group that has been previously selected, select the **Options menu** button for that group, and then select **Deselect Group**.

g. To ungroup a group, select the **Options menu** button for that group, and then select **Ungroup**.

h. To change a **Group AND** group to a **Group OR** group, or a **Group OR** group to a **Group AND** group, select the **Options menu** button for that group, and then select **Change to OR** or **Change to AND**.

Tip

- To clear all criteria and start over, on the Filter toolbar, select **Clear**, and then select **Confirm**.
- To delete a row, select the **Options menu** button for that row, and then select **Delete**.

4. When you are done, select **Save Default Filter**.

See Also

Work with reports

Create a report using the Report Wizard

Add a existing report

Troubleshoot problems with data not displaying in a report

Feedback

Was this page helpful? Yes No

Provide product feedback

Troubleshoot problems with data not displaying in a report

Article • 02/15/2022

There are several possible reasons why data that you expect to be in a report does not appear:

- **Insufficient security permissions**. If you don't have permission in Microsoft Dataverse to view a row, it will not appear in the report.
- **Data is not entered.** The person entering data may have left columns empty.
- **Data does not match the criteria for the report.** Many reports include a default filter that displays only active rows, or you may have selected criteria that don't have any matching row. Try changing the report filter. For more information, see Edit the default filter of a report
- **You may be viewing a cached copy of the report.** By default, data in Dataverse reports is pulled from the database each time you run a report. However, your system administrator may have changed a report to run from the cache. If data you entered recently is not included in the report, you may have an older version of the report from the cache. To refresh the report, on the Report toolbar, select the **Refresh** button.
- **You may not have permission to read rows in a sub-report.** If you do not have permission to read row types that are included in a sub-report, you will get an error message saying that the sub-report could not be displayed.
- **Your Microsoft Internet Explorer privacy settings may block required cookies.** For chart reports, if instead of seeing the chart, you see a red letter X, your privacy settings may be blocking a cookie that is required for the chart control. To fix this problem, in your browser, enable cookies for the server that is running Reporting Services.

See Also

Work with reports

Create a report using the Report Wizard

Add a existing report

Edit report filter

Feedback

Was this page helpful? Yes No

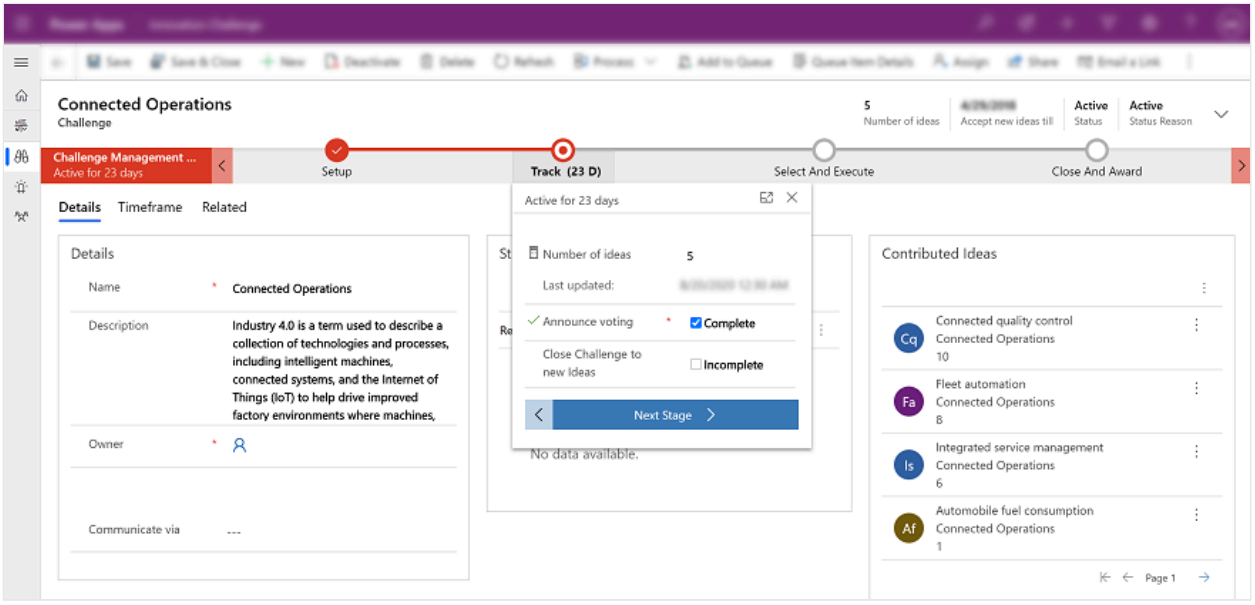
Provide product feedback

Work with business processes

Article • 10/23/2024

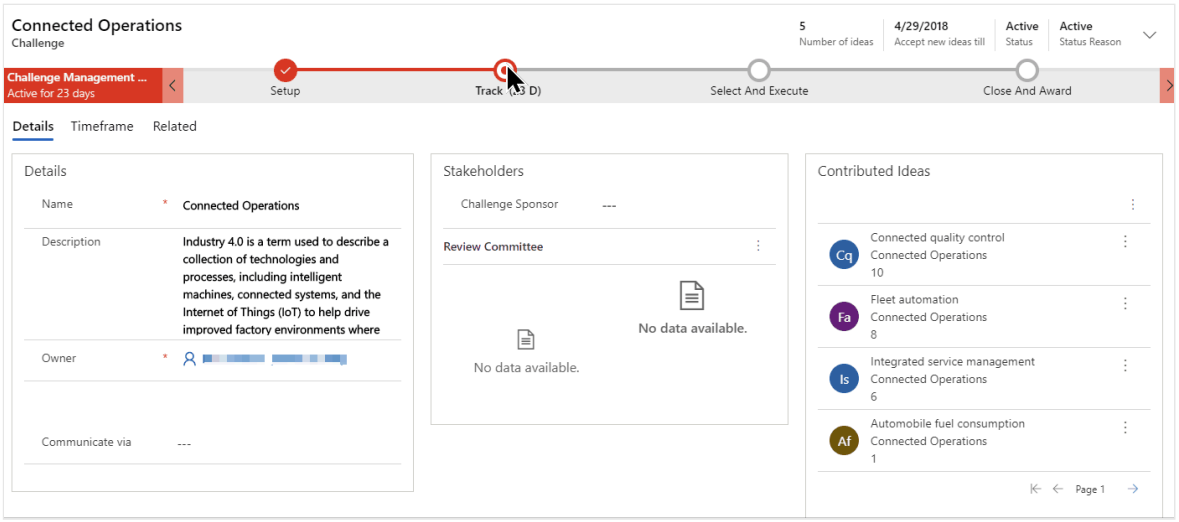
Business processes help everyone follow best practices, even for situations that don't occur often. Where available, business processes provide a step-by-step timeline for the stages of a process at the top of the relevant row. You open the menu for the active stage, enter each column of required and business-critical data, and then select the next stage when you're ready to work with it. Some business processes can be completed in the time it takes to make a single phone call, while others might take several weeks.

As you work on a new customer row, the business process flow helps you see each step that you need to take to create the new row and fill out the required information according to your organizations business process.



The process bar can also be docked to the right side of the screen for easy reference as you work with a customer row.

- On the process bar, select a stage, and then select **Pin the stage flyout** .



What if your business processes look different from these examples?

The system comes with business processes for common tasks. However, most organizations customize these processes to match the way they do things. Or, you might have added processes to the system that have been customized for your industry or business goals. In other words, what you see here may not exactly match the process bars you see on your system. But regardless, the process bars work the same for everyone. You enter data in the columns, and then move the customer to the next stage.

For more information on to how to apply custom business logic with business rules and flows, see Apply custom business logic with business rules and flows in model-driven app.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Power Automate to automate processes

Article • 02/15/2022

Power Automate lets you create automated processes between your favorite apps and services. From within your app, you can run a flow on one or more rows.

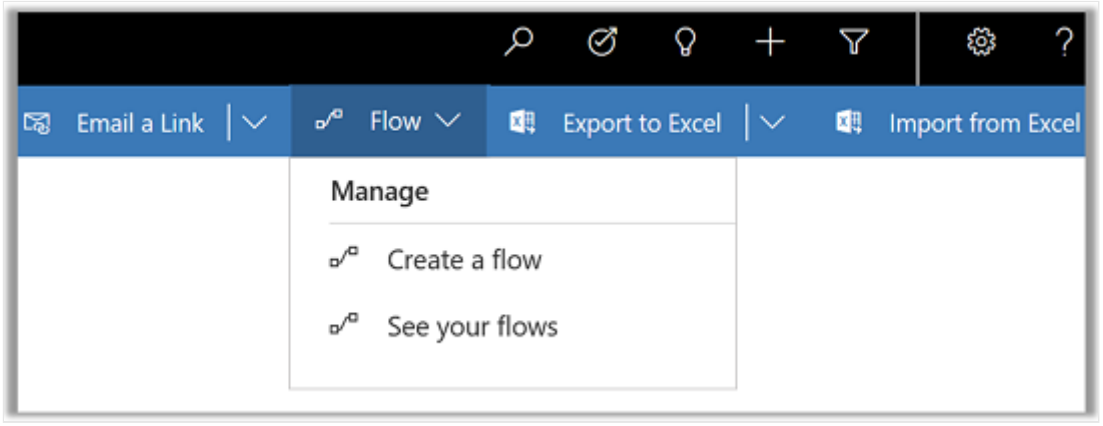
Also, you can open Power Automate from your app to view, edit, or create new flows. More information: Get started with Power Automate

Manage your flows

Connect with Power Automate from your app to manage your flows.

> **Note**
>
> To manage and run Power Automate from your app, Power Automate must be enabled in System Settings for you organization. More information: Flows in your organization

1. On the command bar, select **Flow**.



2. The following options appear:

   - Select **Create a flow** to open a new browser tab and direct you to the Power Automate site. On the Power Automate page, select **Continue** to use an existing template or **X** to close, choose, or create a different one.
   - Select **See your flows** to open a new browser tab and direct you to the Power Automate site where any flows in the Flow environment are displayed.

- Select **Login to flow** to connect with Power Automate and display available flows for you to run. This only appears if you aren't currently signed in to Power Automate.

For information about creating a flow, see Get started with Power Automate

Feedback

Was this page helpful? Yes No

Provide product feedback

Find it with unified search

Article • 09/26/2024

Unified search helps you quickly discover apps in Power Apps, flows in Power Automate, templates, and external web content. From a single entry point, you can find:

Expand table

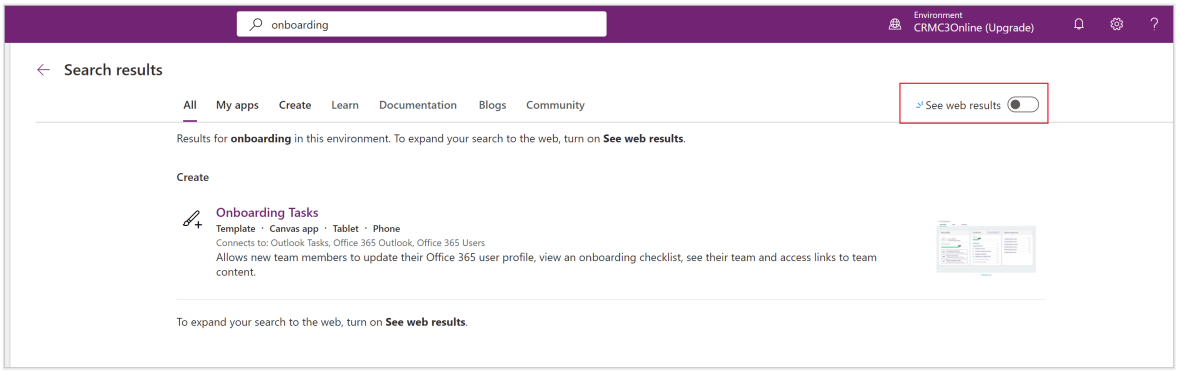
Item	Content type
(Power Apps) Apps that have been built and published	Environment
(Power Automate) Flows that have been built and published	Environment
Templates	Environment
Documentation and learn content	External web
Community content	External web
Blog posts	External web

> **Note**
>
> To open the Power Automate documentation, go to the Power Automate Overview.

Use search

By default, search displays items only from your environment, which are apps, flows, and templates.

1. To display suggestions, type in the **Search** box.
    - For Power Apps, suggestions include apps and templates.
    - For Power Automate, suggestions include flows and templates.
2. To go directly to that item, select any suggestion.
3. (Optional) To expand your search to external web content, select **See web results**.



> **⚠ Warning**
>
> Don't send personal data or confidential/proprietary information as part of your search when you choose this option.

To prevent users from using the **See web results** option to perform external web content searches for an environment, go to Manage search providers.

Manage search providers

The Microsoft.PowerApps.Administration.PowerShell module includes the cmdlets that members of either the Power Platform administrator or Dynamics 365 administrator security groups can use to modify the search providers. More information: Get started using the Power Apps admin module.

By default, all search providers are enabled, except for the docs search provider `PowerPlatform.Search.disableDocsSearch`. The following search providers can be enabled or disabled.

Expand table

Search provider namespace	Default	Description
`PowerPlatform.Search.disableDocsSearch`	True	Setting the provider namespace to True disables search for the provider. When this provider is disabled, users in the environment will see a message that Microsoft Learn and Documentation search categories have been turned off by the administrator in the search results page. When the provider is enabled, users' keywords might be sent outside their

Search provider namespace	Default	Description
		company, region, or cloud. Make sure keywords don't contain sensitive or confidential information.
`PowerPlatform.Search.disableCommunitySearch`	False	Setting the provider namespace to True disables search for the provider. When this provider is disabled, users in the environment will see a message that Community and Blog search categories have been turned off by the administrator in the search results page.

To return the current settings including which search providers are enabled or disabled, run this cmdlet: `Get-TenantSettings`

Disable a search provider

Members of the Power Platform admin role can disable or enable a search provider by specifying the search provider namespace when running the Set-TenantSettings cmdlet.

For example, to disable the Microsoft Learn and Documentation search provider, run this cmdlet:

PowerShell

```
$requestBody = @{PowerPlatform.Search.disableDocsSearch = $true}
Set-TenantSettings -RequestBody $requestBody
```

To enable the Microsoft Learn and Documentation search provider, run this cmdlet:

PowerShell

```
$requestBody = @{PowerPlatform.Search.disableDocsSearch = $false}
Set-TenantSettings -RequestBody $requestBody
```

Alternatively, run:

PowerShell

```
$settings = Get-TenantSettings
$settings.PowerPlatform.Search.disableDocsSearch = $false
Set-TenantSettings -RequestBody $settings
```

See also

Overview of creating apps in Power Apps

Feedback

Was this page helpful? Yes No

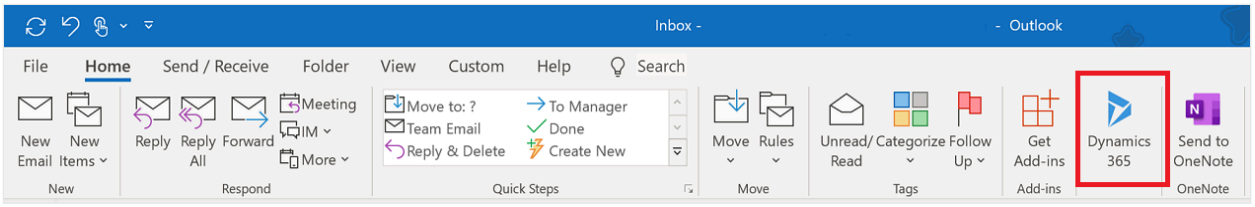
Provide product feedback

Use Dynamics 365 App for Outlook

Article • 02/15/2022

Use Dynamics 365 App for Outlook to manage your app data while using Outlook on the desktop, web, or mobile. You can view information about an email or appointment or link it to a row in your app.

When Dynamics 365 App for Outlook is installed, in the Outlook ribbon select **Dynamics 365** to open the **Dynamics 365** pane next to a selected Outlook email message, or when you're composing an email message or setting up a meeting or appointment.



What Dynamics 365 App for Outlook offers?

With Dynamics 365 App for Outlook, you can:

- Track your emails and appointments in Outlook and associate them to the right record in your app.
- Synchronize contacts and related information so that Exchange and your app are always up to date.
- View information about Dataverse contacts and accounts while you're working in Outlook. You can view this information in the context of an email message, meeting, or appointment. For example, view phone numbers, company name, last and next activities, and recent rows.

For information on using Dynamics 365 App for Outlook, see Basic navigation in App for Outlook.

See also

Customize Dynamics 365 App for Outlook

Feedback

Was this page helpful? Yes No

Provide product feedback

Overview of the new Power Apps vibe experience (preview)

[This article is prerelease documentation and is subject to change.]

The new Power Apps vibe experience is an AI-native platform that builds solutions to business problems, including requirements, data, and full apps with generated code. You can automate workflows, integrate data sources seamlessly, and accelerate app development without extensive coding knowledge.

Why choose Power Apps vibe?

The vibe experience transforms how you build business applications by combining three key capabilities in a single, integrated workspace:

- **Rapid prototyping**: Go from idea to working app in minutes, not weeks. Simply describe your business needs in natural language, and AI generates a complete solution including data models, business logic, and user interface.
- **Unified development**: Instead of switching between separate tools for planning, data modeling, and app building, work with everything in one place. Make changes to your plan, data, or app, and see updates automatically reflected across all components.
- **AI-powered assistance**: Get intelligent suggestions, code generation, and guided problem-solving throughout the development process. The AI understands your business context and helps you make better architectural decisions.

Ideal for: Business analysts, citizen developers, and professional developers who want to build line-of-business applications quickly without getting bogged down in technical implementation details.

Common use cases: Employee onboarding apps, inventory management systems, project tracking tools, customer feedback applications, and departmental workflow solutions.

Watch this video to see the vibe experience in action. The video demonstrates how to create a complete business application from a simple text description:

https://learn-video.azurefd.net/vod/player?id=085271a3-bec6-4740-89df-044f984466d2&locale=en-us&embedUrl=%2Fpower-apps%2Fvibe%2Foverview

After watching the video, explore the detailed walkthrough in Create apps, data, and plans together to understand the full development process.

> **Important**
>
> - This feature is in preview.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Prerequisites

- A tenant admin must enable Copilot for the tenant. To enable Copilot, admins should:

  1. Sign in to the Power Platform admin center.
  2. In the navigation pane, go to **Settings** to open the **Tenant settings** page.
  3. Set the toggle for **Copilot in Power Apps (preview)** to **On**.
  4. Select **Save** to apply the changes.

- **Optional but recommended**: For higher quality apps, enable **Anthropic models** by going to Power Platform Admin Center > **Environments** > **[Name of Environment]** > **Settings** > **Features** > **Enable External Models**.

  For more information, see Tenant settings.

- This capability isn't available in a default environment.

- This capability is currently available only in the US, Australia, Asia, and India regions, and only in English. For more information, see Explore Copilot features by geography and languages.

Troubleshooting connectivity issues

If you're experiencing connection errors or can't access environments:

- **Verify environment region**: Ensure you have access to an environment located in the US, Australia, Asia, or India region. You can check your environment region in the Power Platform admin center:

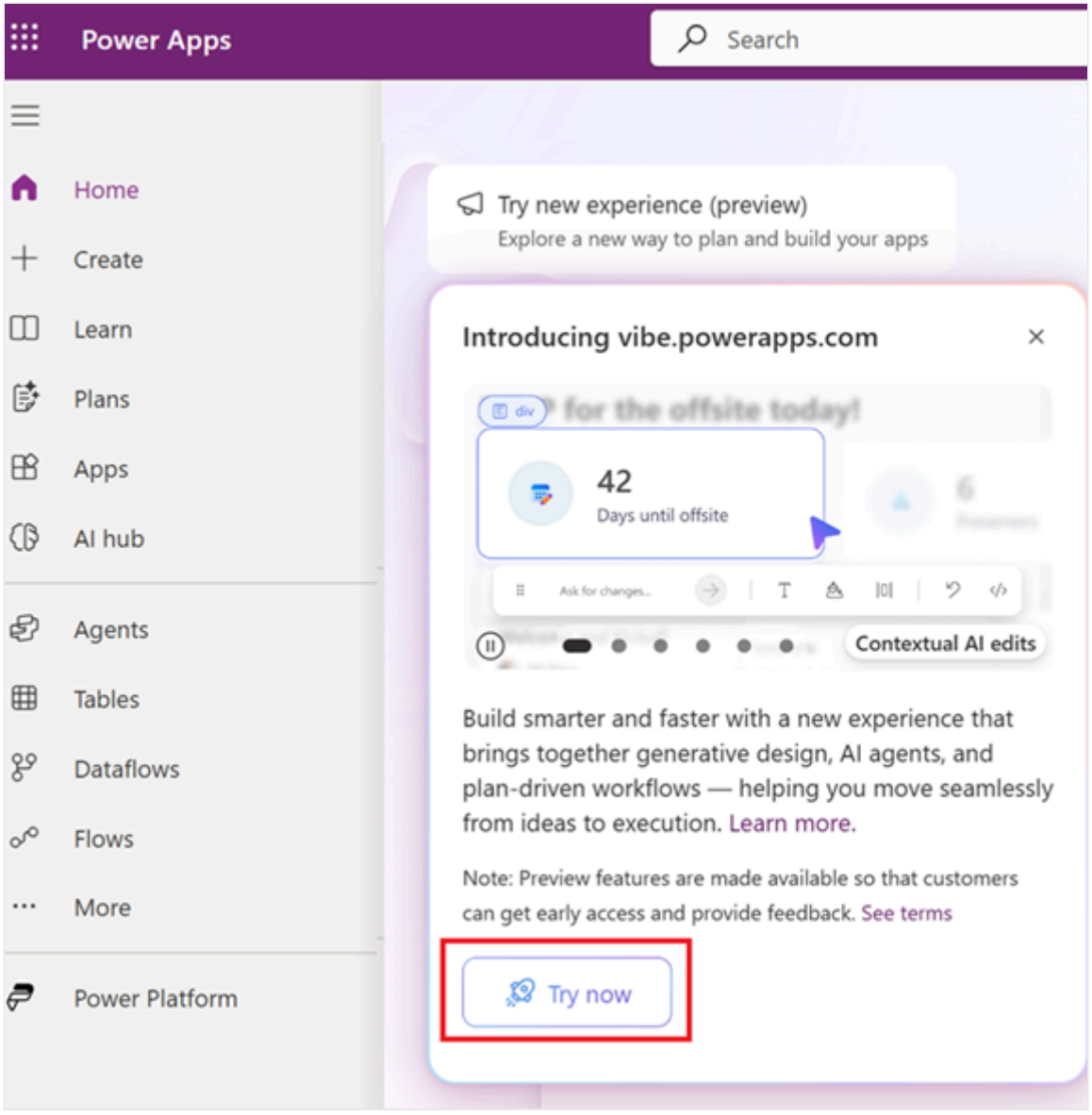
1. Sign in to the Power Platform admin center.
2. In the navigation pane, select **Manage** > **Environments**.
3. Locate the **Region** column to see where your environments are hosted.

For additional support with connectivity issues, contact Power Platform support.

Access Power Apps vibe

You can access the new vibe experience in one of two ways:

- Go to https://vibe.powerapps.com and sign in to start using the new experience.
- Alternatively, sign in to Power Apps. At the top, select **Try new experience (Preview)** to enable it.

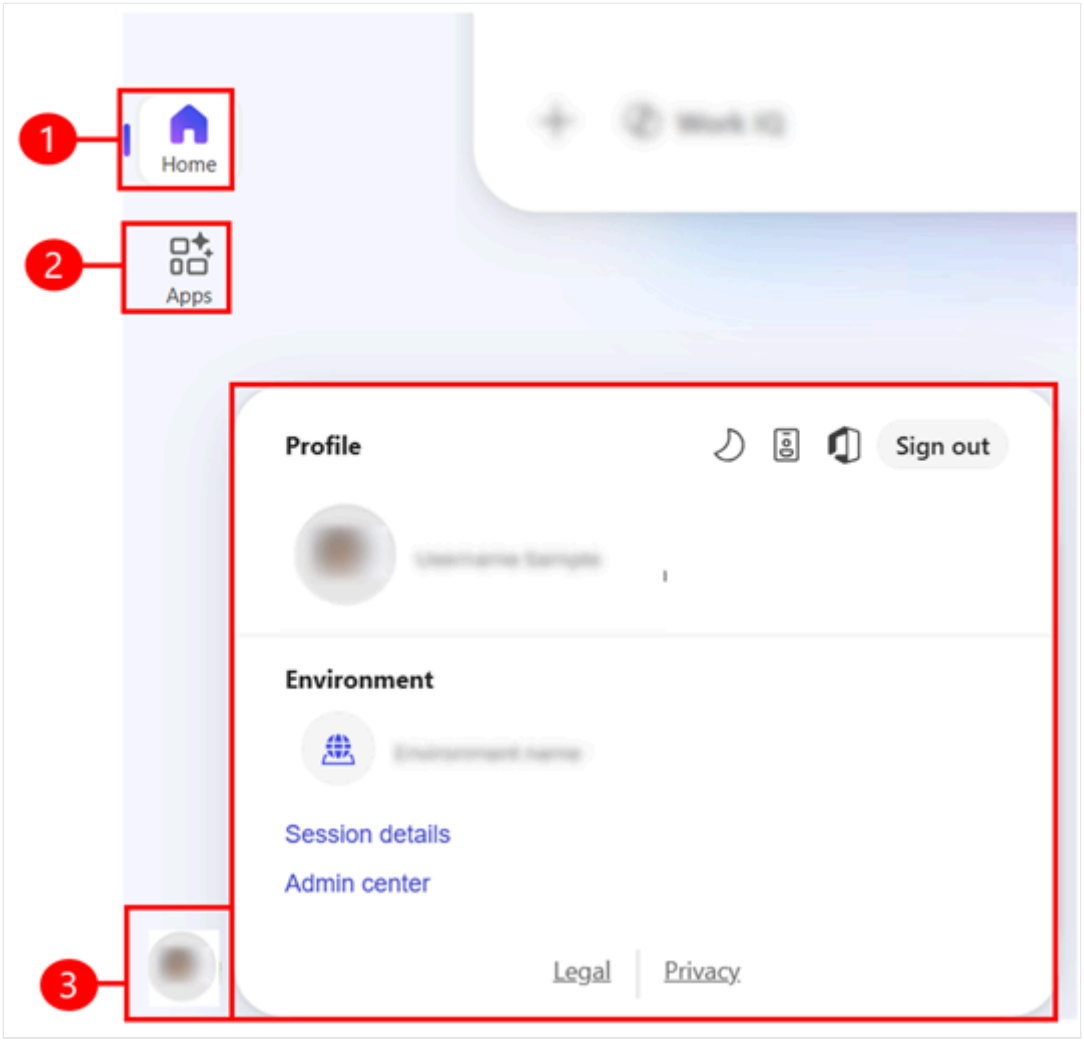


> **Note**

If you access the new experience through Power Apps, you need to enable it each time you sign in.

Navigation

The left navigation shows the following items:

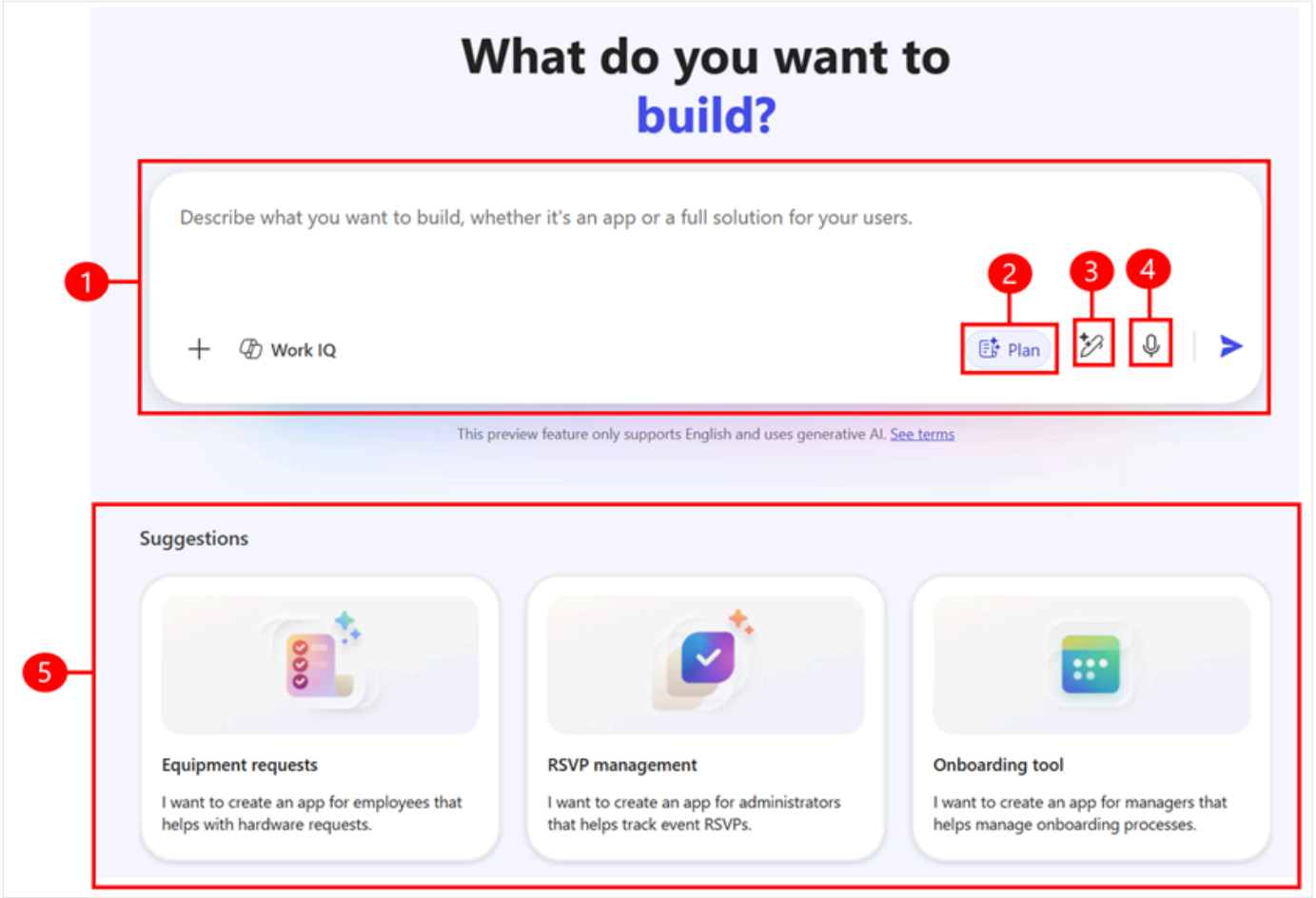


Legend:

1. **Home**: Returns you to the main landing page.
2. **Apps**: Open, play, or edit apps you created or that others shared with you.
3. **Profile**: Access the environment picker, session details, notifications, and settings.

Enter your prompt to build apps

Describe your idea using natural language.



Legend:

1. **Enter prompt**: Type a description of what you want to build.
2. **Plan**: Use this mode to review and discuss a plan with agenta before making changes.
3. **Enhance prompt** (Optional): Add more detail to your prompt before generating your app.
4. **Start dictation**: Start speech to text and speak into a microphone to convert your voice into text.
5. **Suggestions**: Want to experiment? Select one of the suggestions to explore the new app-building experience.

Next steps

- Create apps, data, and plans together (preview)
- FAQ for the Power Apps vibe experience (preview)

Last updated on 04/21/2026

Create apps, data, and plans together using vibe (preview)

[This article is prerelease documentation and is subject to change.]

The new vibe experience in Power Apps combines plans, data models, and apps in a single design surface. This integrated approach minimizes distractions and keeps you focused on building solutions efficiently.

Key capabilities

- **Design everything together**: Build apps, plans, and data models together in a single, cohesive process.
- **Build faster with AI**: Quickly move from ideas to working prototypes with conversational chat.
- **Instant app creation**: Generate modern web apps directly from your requirements and data with one click.

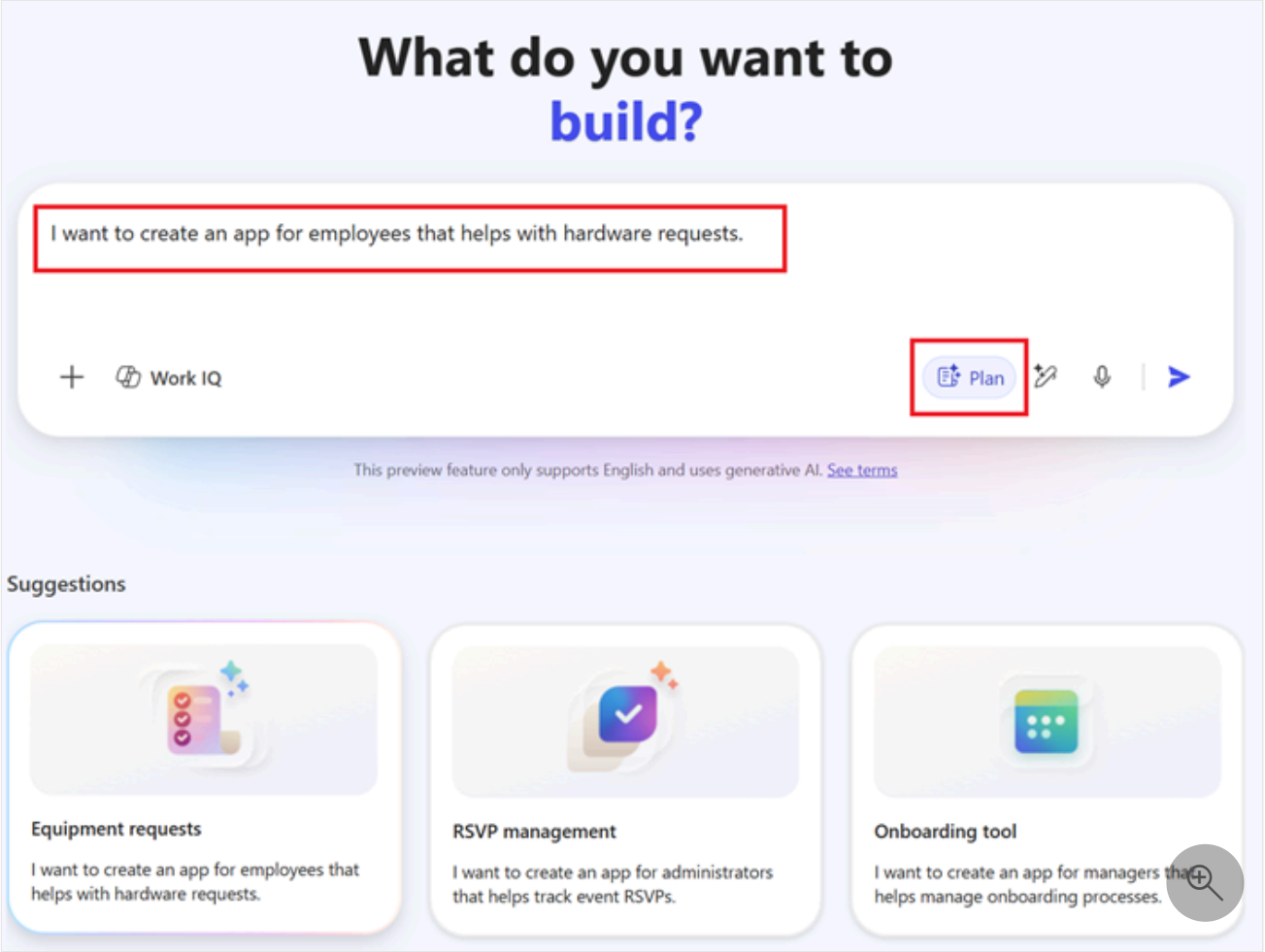
> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Create an app

1. Sign in to https://vibe.powerapps.com.
2. Enter your prompt in the text box. **Plan** mode is the default mode, which means it starts with a plan that includes potential clarifying questions before building the app. You can also select **Start dictation** and use your voice to turn speech into text.

   > **Tip**

To skip planning and start creating apps, select **Plan** again to turn it off.



3. In **Plan** mode, the agent shows you a proposed plan to review before generating the app. It might also ask follow-up questions to clarify your requirements.

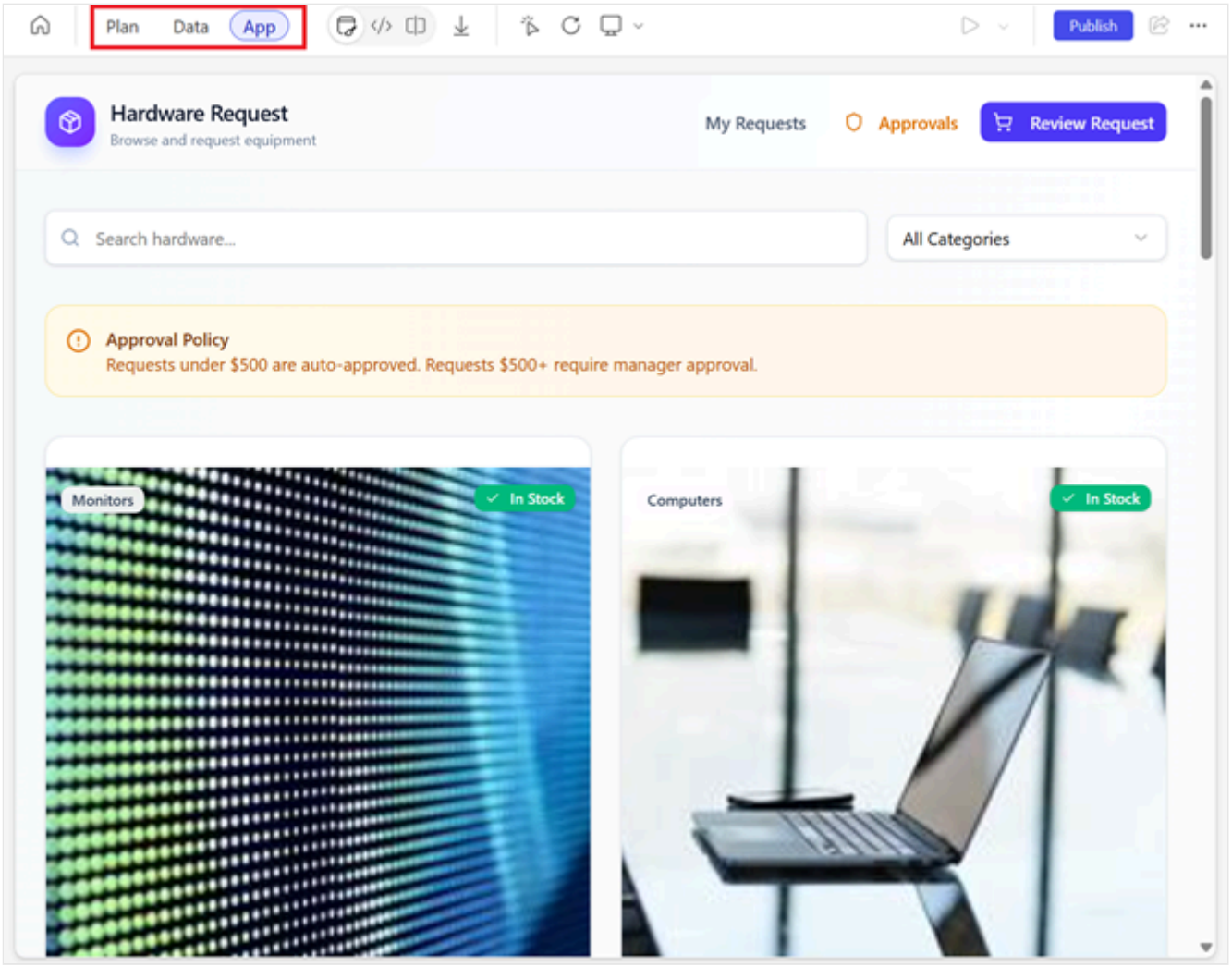
4. When the plan looks good, select **Accept this plan and create app** > **Submit** to start generating your app.

5. The agent starts working and loads assets. When it's finished, you have a generated app, data model (if applicable), and a plan (if applicable). You can then make any necessary edits.

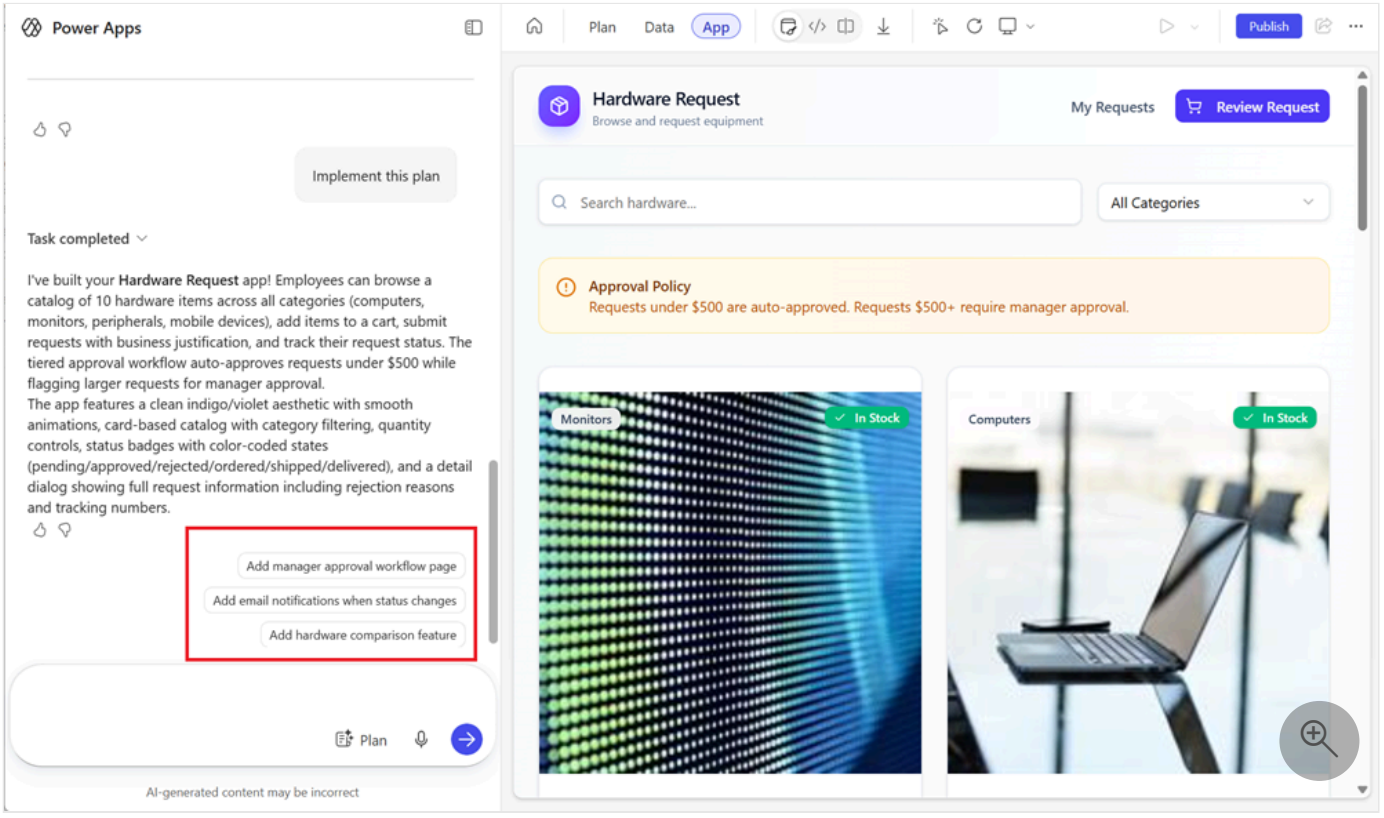
6. The first save stores the app, tables, and plan together in your selected solution, while your data remains in draft tables (in memory). If you don't see a **Save** button in the top right of the screen, your app is autosaved automatically.

Review and refine your generated app and data model

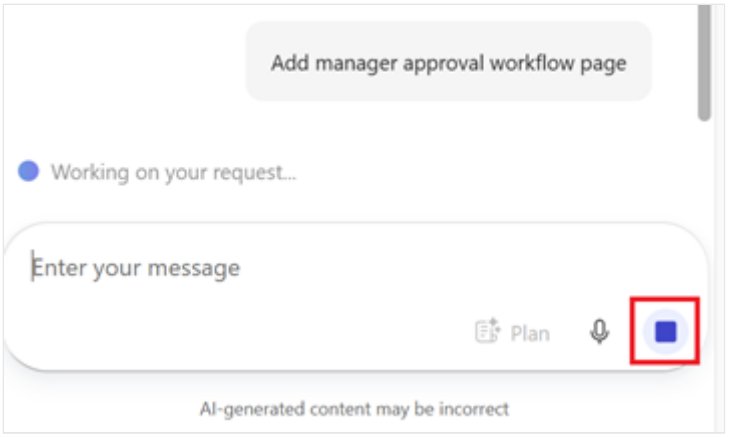
The new workspace uses a unified chat experience that keeps context as you switch between **Plan**, **Data**, and **App**. It speeds up development by letting you preview the app visually, inspect the code, and make changes without leaving the workspace.



A list of next actions appears above the chat area to suggest relevant next steps.

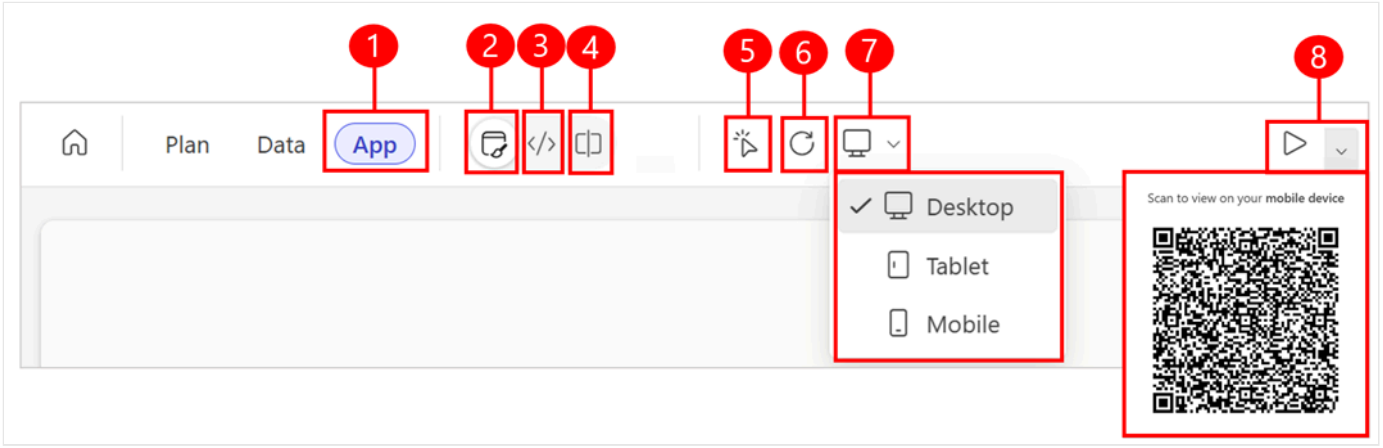


If you submit a request and then decide to cancel it, select the stop button.



Focus area

Each focus area displays its specific tools above the work surface. For instance, when you select the **App** focus area, you see these options:



Legend:

1. **App**: When selected, this focus area displays menu options specific to the app.
2. **Preview**: View a live preview of the app.
3. **Code**: Access and review the app's code.
4. **Split**: See both the code and the app preview side by side.
5. **Toggle inline edits**: Select any area in the app and edit its text directly.
6. **Refresh preview**: Reload the app preview to see the latest changes.
7. **Select preview screen size**: Choose **Desktop**, **Tablet**, or **Mobile** to see how your app renders across screen sizes and orientations.
8. **Play**: Run the app to test behavior and interactions. Use the drop-down to show a QR code, then scan it to open the app on your mobile device in a browser.

Edit the generated app

When generation finishes, the visual preview appears in focus on the work surface.

You can interact with the app directly, using the navigation items and buttons to see how they function and respond.

To inspect the code, switch to **Code** view or **Split** view. Expand the **src** folder to see subfolders and files, and select any file to view the generated code.

When you modify the app, plan, or data, the agent analyzes your changes to see if updates are needed elsewhere in the solution. If so, the agent automatically applies updates and summarizes them in the chat.

Edit the app using chat

Use the chat input on the work surface to ask questions or request changes to the generated app. For example: "What does this button do?" or "Change theme to blue."

- **Plan mode**: To discuss potential changes with the agent before applying them, select **Plan** in the chat box. The agent proposes a plan for your review. Once you accept it, the agent executes the plan and makes the necessary edits. When the edits are complete, **Plan** mode turns off automatically.
- **Questions**: If you ask a question, the agent provides an explanatory response.
- **Change requests**: If you request a change, the agent evaluates the request, outlines an implementation plan, and then applies it.
- **Unsupported changes**: If the agent can't implement the change directly, you receive step-by-step guidance or instructions for retrying the request.

Edit the app using inline actions

Use inline actions to edit the style of your app.

1. Select the option to toggle inline edits.
2. Select the item you want to edit. The element is outlined and labeled with the corresponding code object so you know exactly what you're editing.

   Depending on the element type, a properties pane or inline controls appear for typography, styling, or layout. Adjust values like font, color, or spacing directly in the pane.

   > **Tip**

If multiple elements share the same style, a single change applies across the group. For example, changing a button color inside a card updates all buttons in that card group.

Inline edits update the associated code directly without involving the AI agent.

You can also open inline chat on a selected element and request a change like **Make this button primary blue**. The element's context is passed automatically, so you don't need to describe it in the prompt.

3. When you're done, review the result in the visual preview. If needed, switch to **Code** view or **Split** view to inspect the generated files.

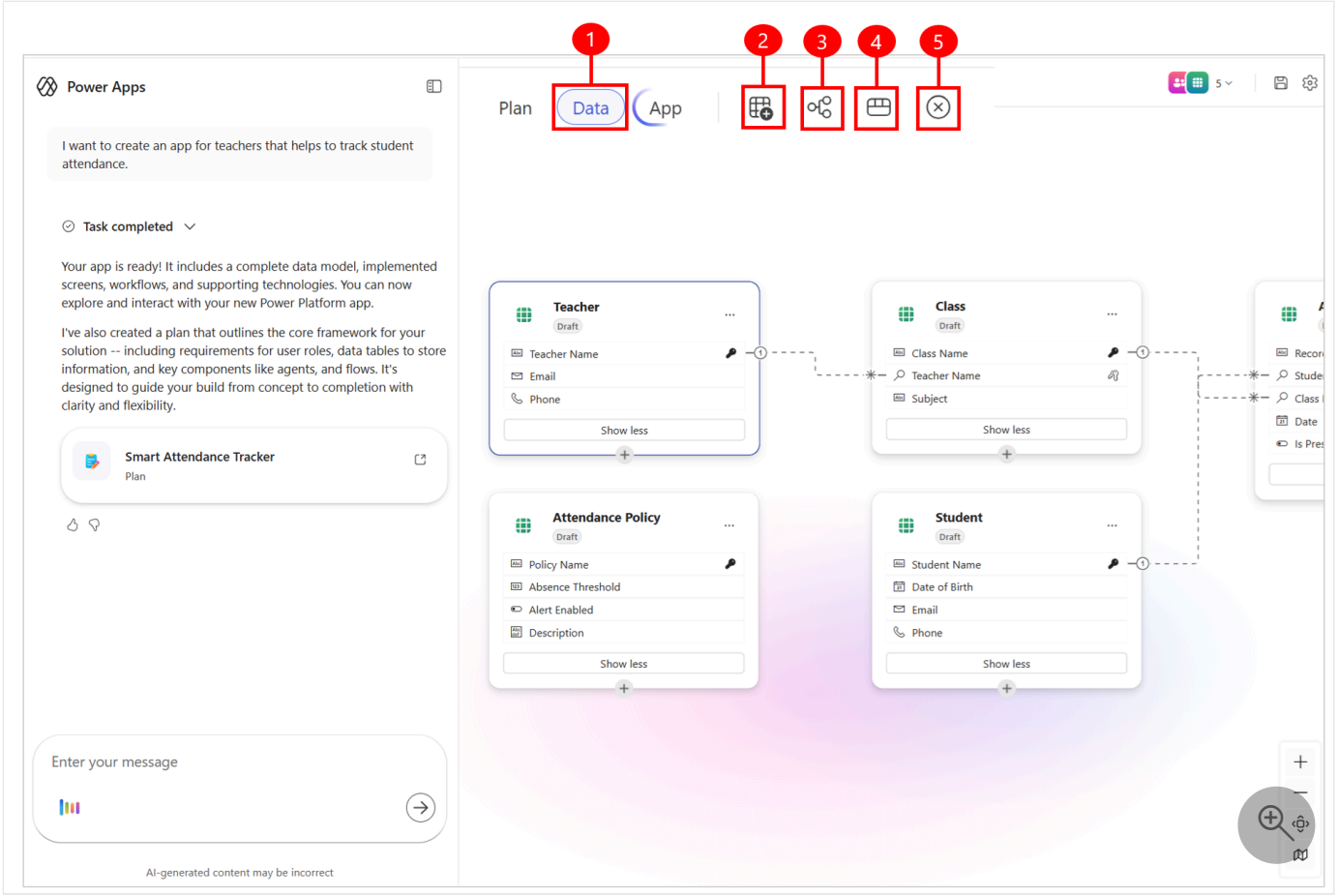
Keyboard shortcuts for inline actions

Use keyboard shortcuts to open and close inline edit actions while editing your app.

- **Hold and release Alt** to toggle the inline edit gesture.
- **Press Esc** to dismiss the inline edit gesture.

Edit the generated data model

Select **Data** to review and refine your data model. The data view displays all proposed tables, their relationships, and sample data to help you understand the schema and how the generated app uses the data. The tables exist in memory only and aren't published to a data source until you explicitly publish them. You can modify them until they meet your business needs. For more information, see Create and edit tables using Power Apps.



Legend

1. **Data**: Edit the data model directly in this view.
2. **Add table**: Add a draft or existing table.
3. **Create relationships**: Creates a table relationship between two tables.
4. **Data view**: View the data workspace experience where you create tables, configure table relationships, and can view a diagram of your data.
5. **Remove**: Deletes the table and all associated table rows.

Edit data using chat

Use the chat to ask questions or request changes to the proposed fields, tables, and relationships.

You can also ask for more sample data to enrich your dataset for the app. The agent analyzes your requests and makes the necessary updates.

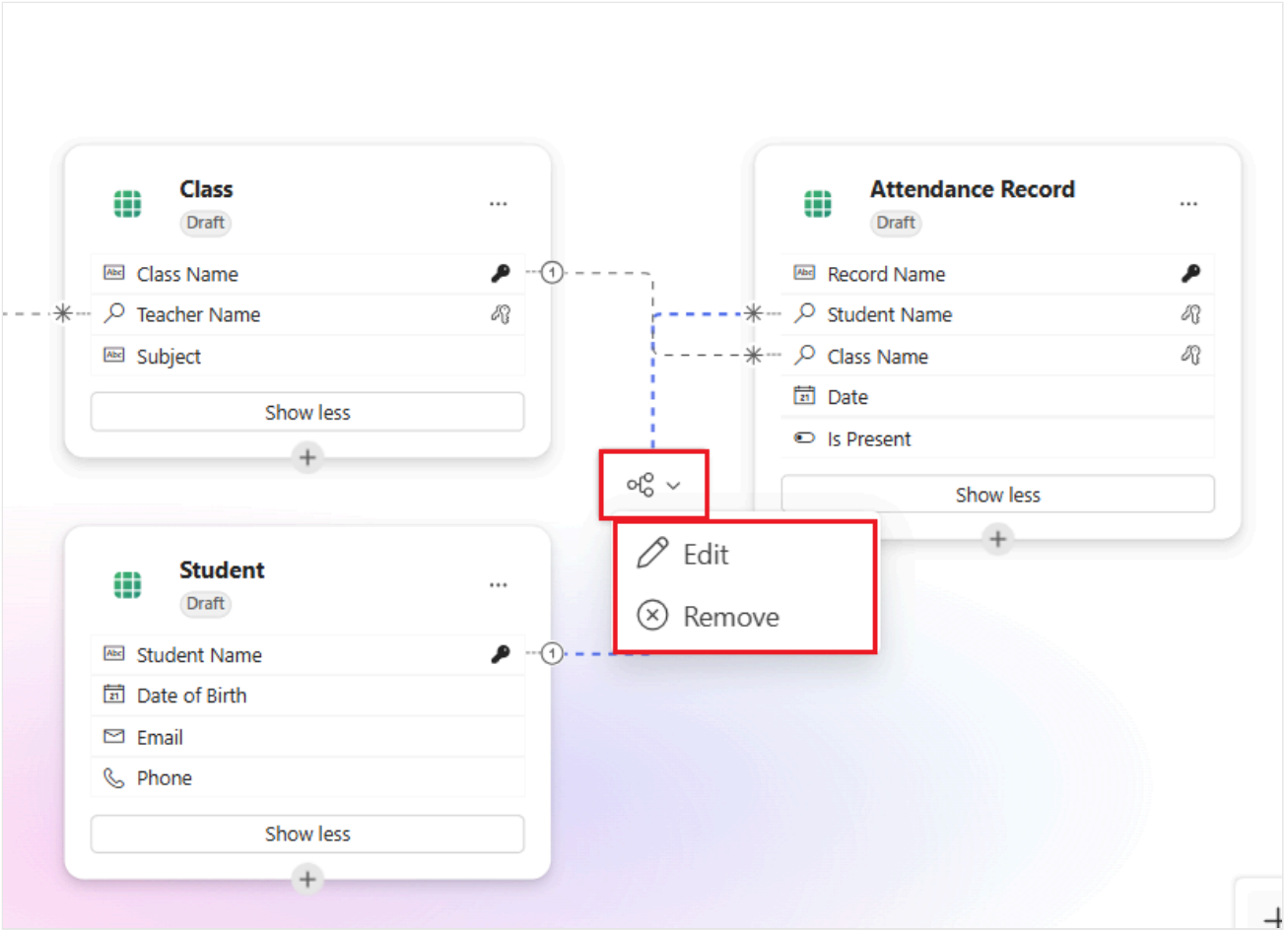
Some actions might not be supported through agent chat. For more information, see the limitations section in the article.

Edit data using inline actions

Add new or existing tables, or make targeted changes to relationships, tables, and sample data directly within the data workspace.

1. To edit data, select the relationship line or the ellipsis (...) next to the table you want to modify.

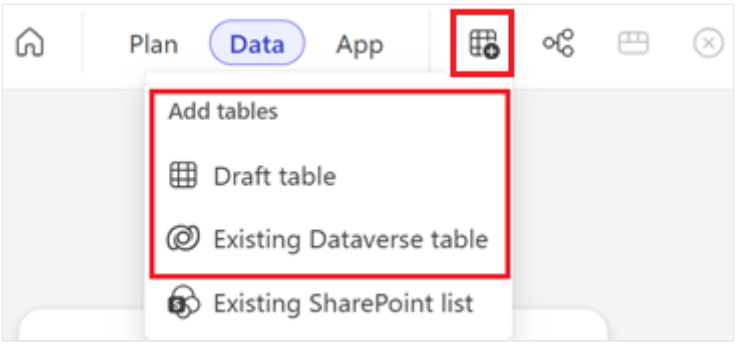
2. Select **Remove** to remove relationships or tables from the data model, or make specific edits as needed.



Add a data table

1. Select **Add table** in the command bar to include a new or existing table in your data model.

2. Choose **Draft table** to create a new in-memory table from scratch, or select **Existing Dataverse table** to add data you already have.



> **Note**
>
> You can also add SharePoint data. For more information, Add a SharePoint list in Power Apps vibe.

Add and edit relationship

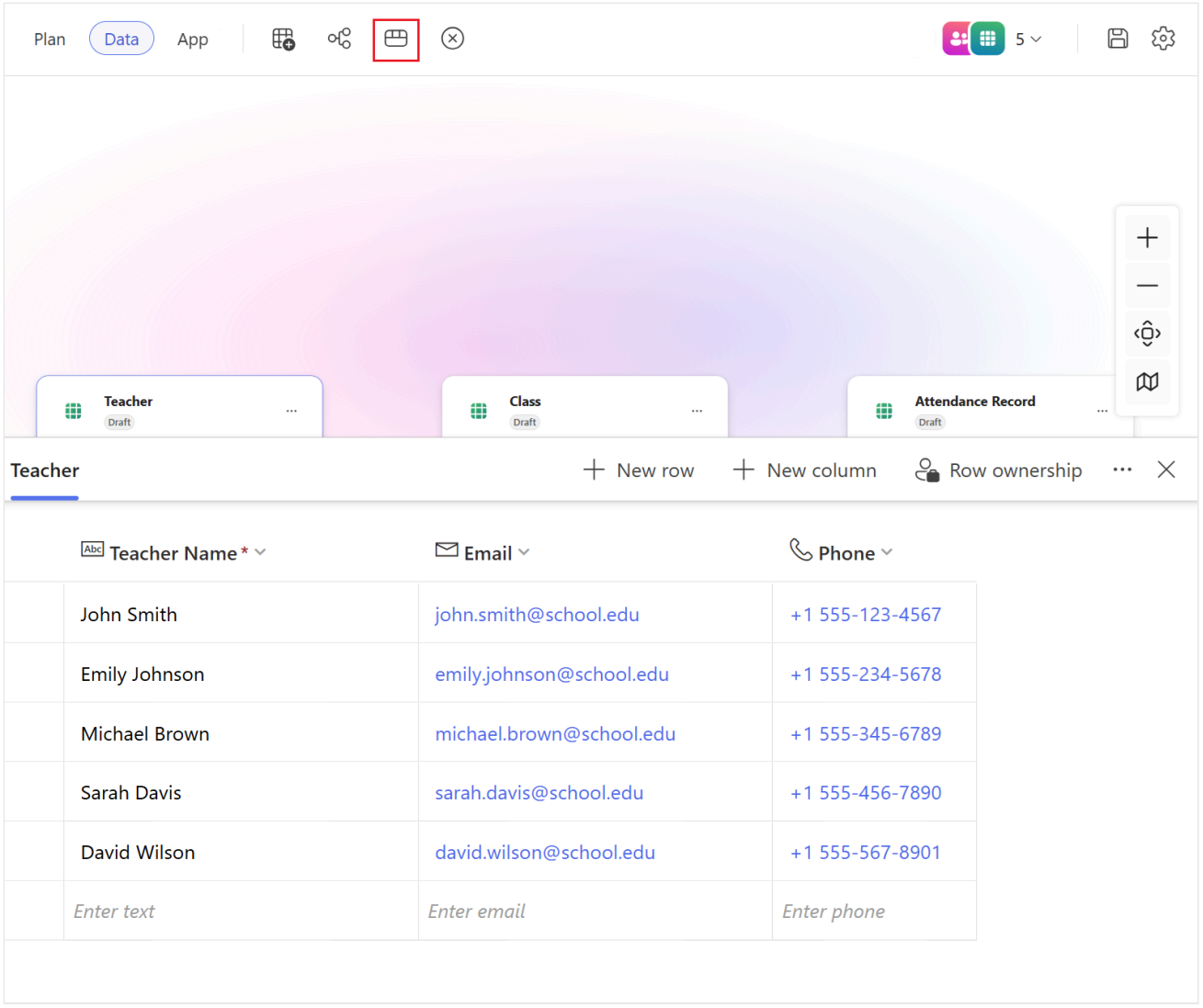
1. Select **Add relationship** in the command bar to add a new relationship between two tables.
2. To edit a relationship, select the relationship line between two tables and select **Edit** from the dropdown menu. Change the relationship type or add a lookup column. You can also perform advanced edits like changing the logical names used in the relationship.
3. Some relationship configurations currently aren't supported. For more information, see Known limitations.

Remove a data table

1. Select the table you want to remove.
2. Select **Remove** in the command bar.
    - If you remove a draft table, you delete it along with all its rows.
    - If you remove an existing Dataverse table, the table and its data remain in the original source; you only remove the reference from your solution.

View and edit data in a table

1. Select **View data** from the command bar.

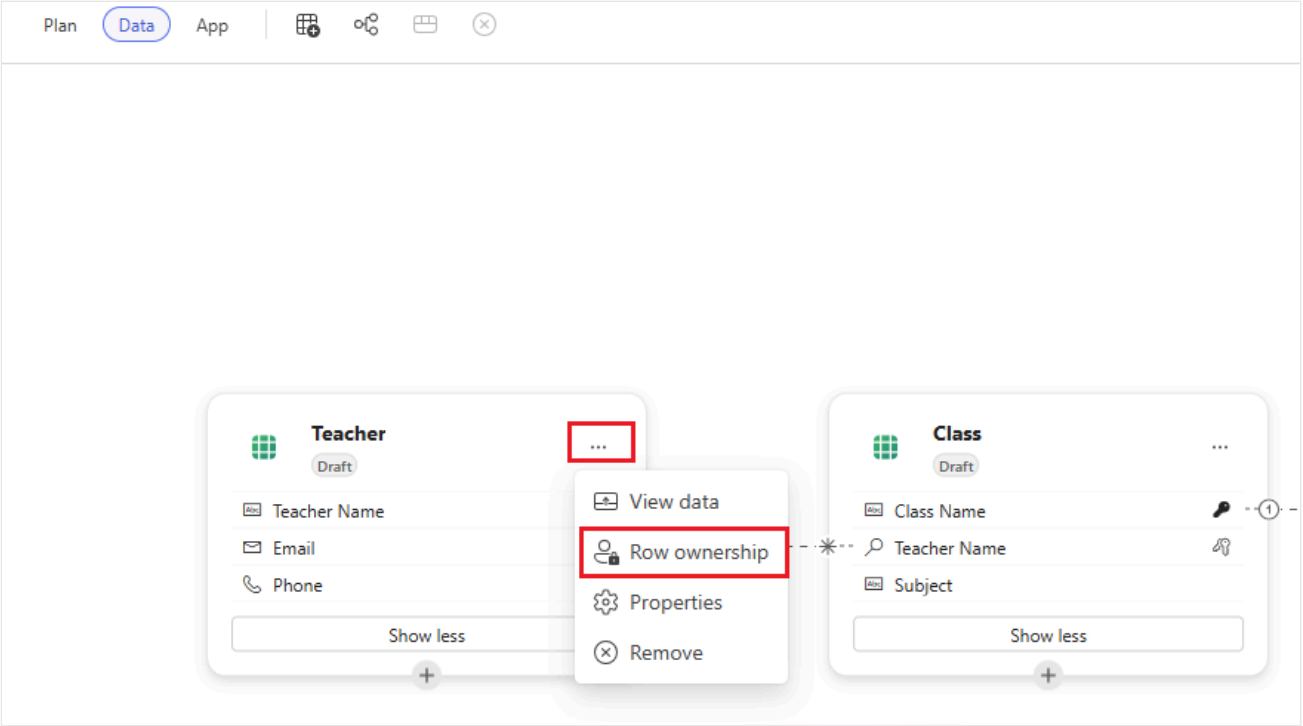


2. In this view, you can add, remove, or modify columns and rows as needed.

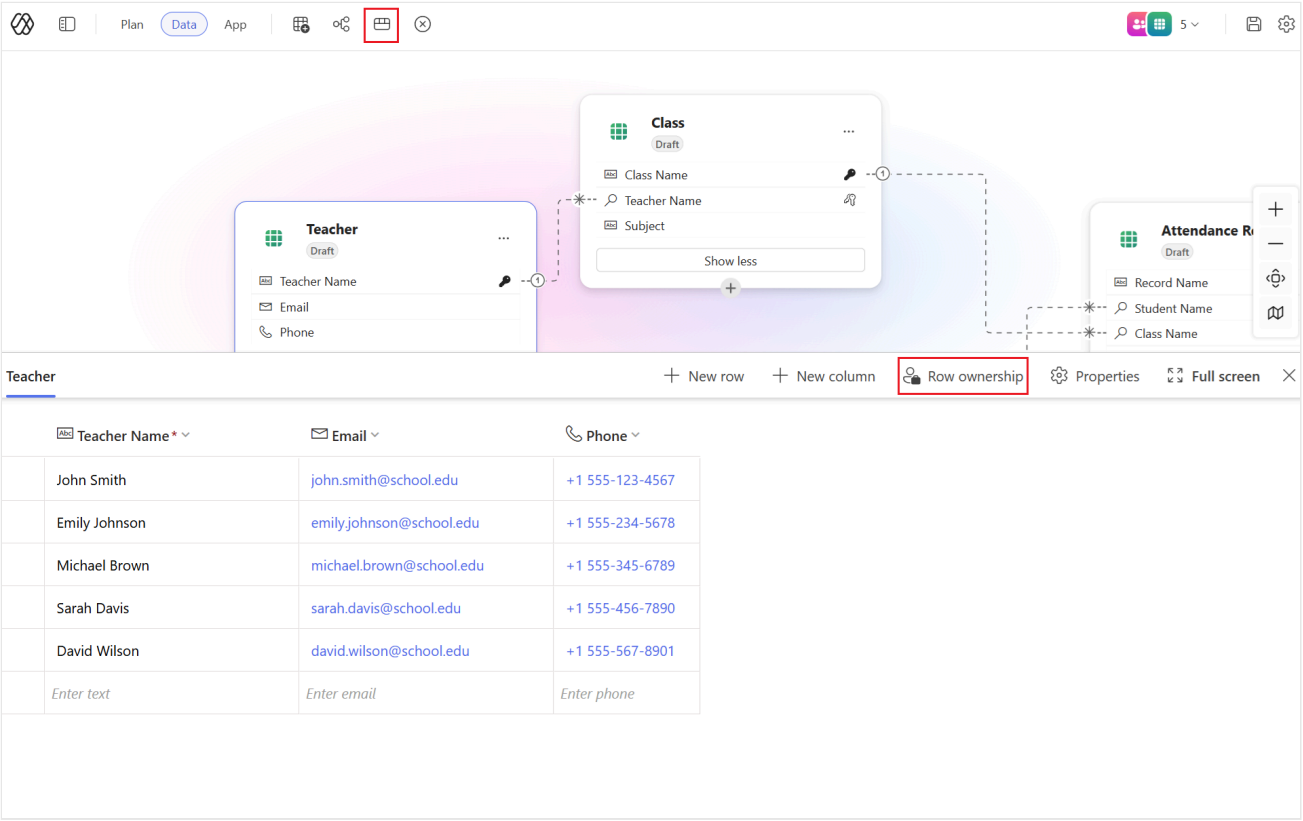
Row ownership

To change the ownership type to **User**, **Team**, or **Organization**, use one of the following methods:

1. Select **View options**, and then select **Row ownership**.

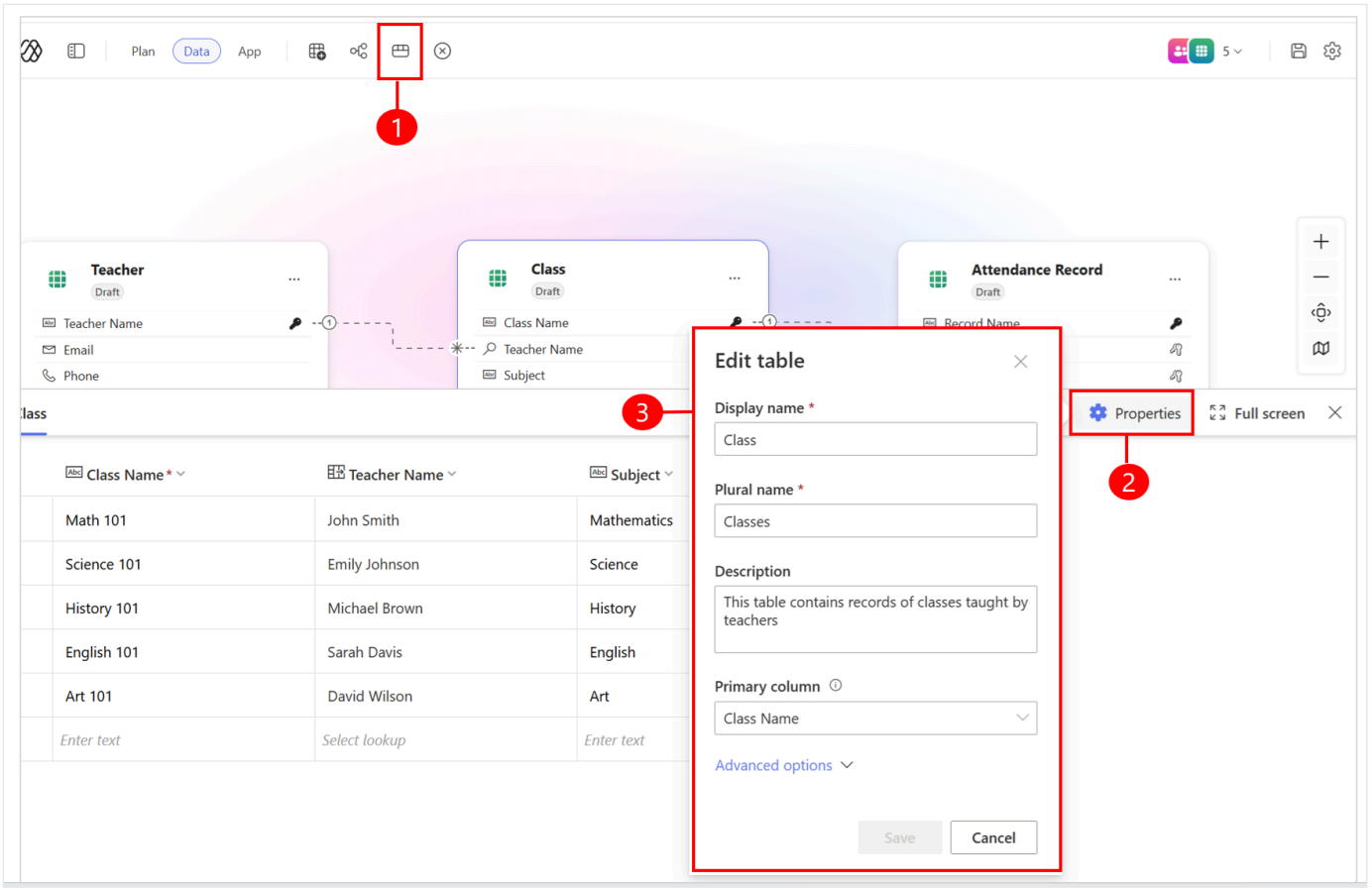


2. Select **View data** on the command bar, select a row, and then select **Row ownership**.



Edit the data table properties

1. Select the table you want to edit. Then, select **View data** in the command bar.
2. Select **Properties** to update the table's display name, plural name, description, and other details.



Application lifecycle management (ALM) capabilities

The following ALM features are supported:

- **Save**: For new apps created after April 9, 2026, AutoSave is turned on by default. You don't need to manually select save. For apps created before this date, manual saving is still required. To save your progress, select **Save** in the command bar at any time. This action saves all edits to every solution object in your preferred solution.

- **Publish data**: To publish draft tables, select **Publish draft tables** from the data card's context menu when you're ready to make a table available for production. You can choose Dataverse as the target data source. Once published, any apps using these draft tables automatically update to use the published versions.

- **Publishing app**: When you're ready to release your app for production, select **Publish** in the app's command bar. If your app uses any draft tables, you're prompted to publish those tables as well.

- **Share the app**: To share your app with others, select **Share** and add the accounts of users you want to grant access to. You can also share with Microsoft Entra ID security-enabled groups. Distribution lists and nonsecurity Microsoft 365 Groups aren't supported.

> **Note**
>
> Group sharing only works in production environments. Developer environments don't support group-level permissions.

Choose a data source and publish

1. When you're ready to publish, select **Publish**.
2. Your Dataverse tables are published in the same environment.

Known limitations

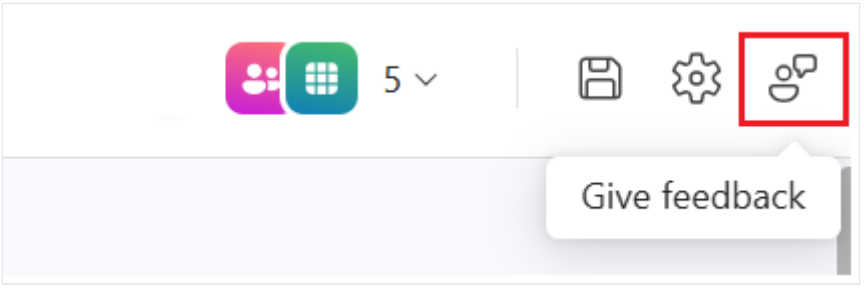
The new Power Apps experience has the following limitations:

- You can't access or edit apps created in the new authoring experience outside of the new authoring experience.
- If you export an app and edit it outside the new authoring experience (for example, in VS Code), redeploying via PAC CLI creates a new app, and it disconnects from the original plan.
- Currently, you can only have one app per plan; multiple apps within a single plan aren't supported.
- You can't open preexisting plans in the new experience.
- Canvas and model-driven apps aren't supported; you can't recommend, open, or edit these app types in the new authoring experience.
- Existing tables aren't automatically recommended during data model proposal; you can manually add existing tables to your plan.
- Editing existing Dataverse tables via chat isn't currently supported; you can make changes to these tables through the data editor manually.
- Direct code edits in **Code** view or **Split** view aren't supported; you can't modify code directly in these views.

Share your feedback

- **Feedback on generated content**: Use the thumbs up or thumbs down icons in the chat pane to rate the generated content. For more details, select **Tell us why** to provide specific feedback.

- **Feedback on the overall experience**: Select **Give feedback** in the command bar to share your thoughts about the overall experience. Choose a feedback category and enter your comments in the provided field.



See also

FAQ for the Power Apps vibe experience (preview)

Last updated on 04/21/2026

Add a SharePoint list in Power Apps vibe (preview)

[This article is prerelease documentation and is subject to change.]

Connect your SharePoint lists directly to Power Apps vibe and start building apps with your existing data in minutes. No migration required.

Build a functional app using your SharePoint list data while keeping your current workflows intact. This approach is ideal for scenarios where you want to:

- Use existing SharePoint lists.
- Build apps quickly without data migration.
- Maintain your current SharePoint-based workflows.

> **Important**
>
> SharePoint lists are currently read-only in Power Apps vibe. To modify list items, you need to make changes directly in SharePoint.

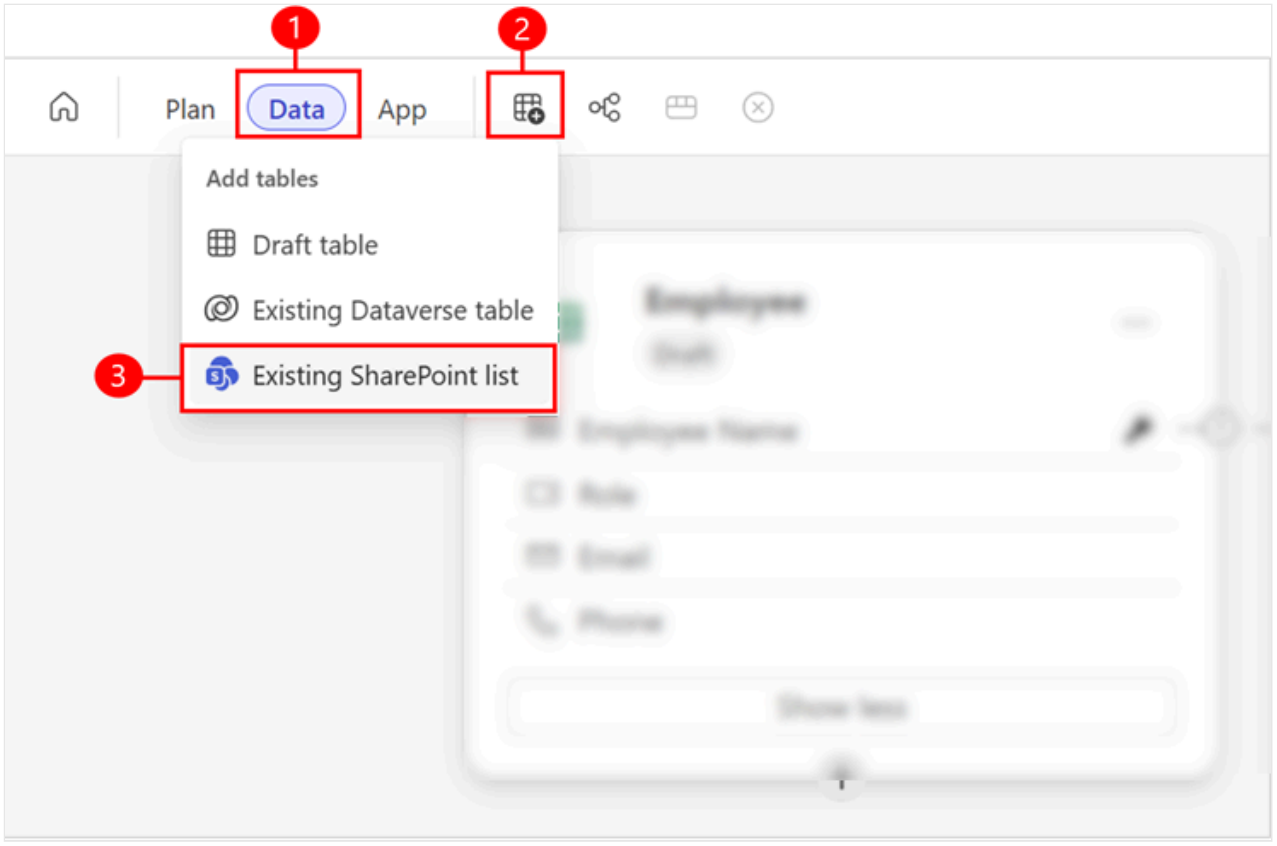
Prerequisites

You need access to the SharePoint site that contains the list you want to use.

Add an existing SharePoint list

Select an existing SharePoint list to use with your app. Vibe doesn't suggest or automatically identify available lists.

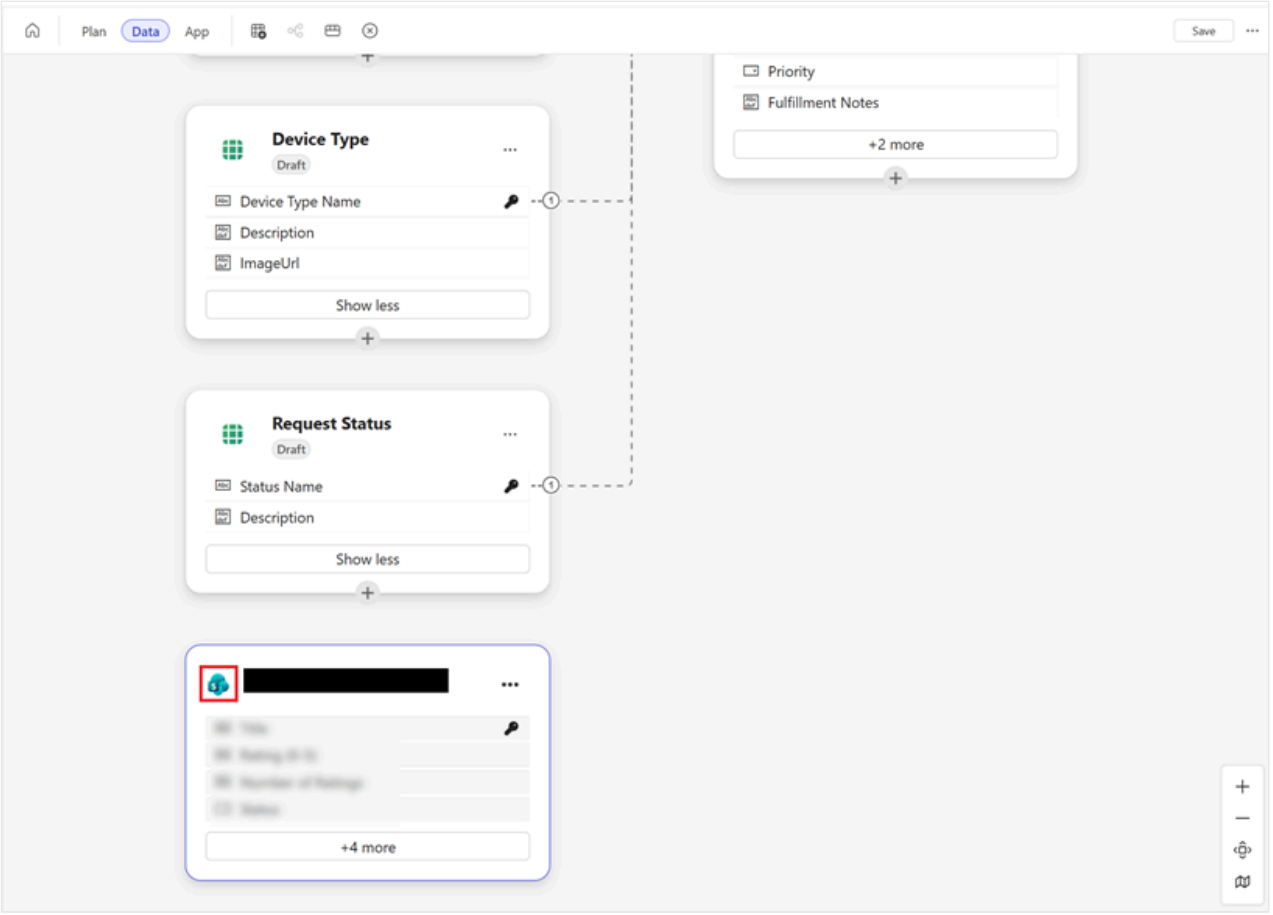
1. Sign in to https://vibe.powerapps.com.
2. Open an app for editing, and then select **Data** > **Add table** > **Existing SharePoint list**.



3. Enter the SharePoint site URL, and then select **Connect**. Alternatively, select a site from **Recent sites**.

4. Under **Select a list**, choose the list you want to add, and then select **Connect**.

5. Vibe connects directly to your SharePoint list. After the list is added, it appears in the data workspace with a SharePoint icon and its columns.



Limitations

- **Location**,**ManagedMetadata**, **Calculated**, **ExternalData**, and **TaskOutcome** columns aren't currently supported and are converted to text columns.

Last updated on 04/20/2026

Released versions - new Power Apps experience (preview)

This capability is in preview with the following limitations:

- Copilot must be enabled in the tenant and environment.
- This capability isn't available in a default environment.
- This capability is available only in the US region and in English only.

Latest version availability

Expand table

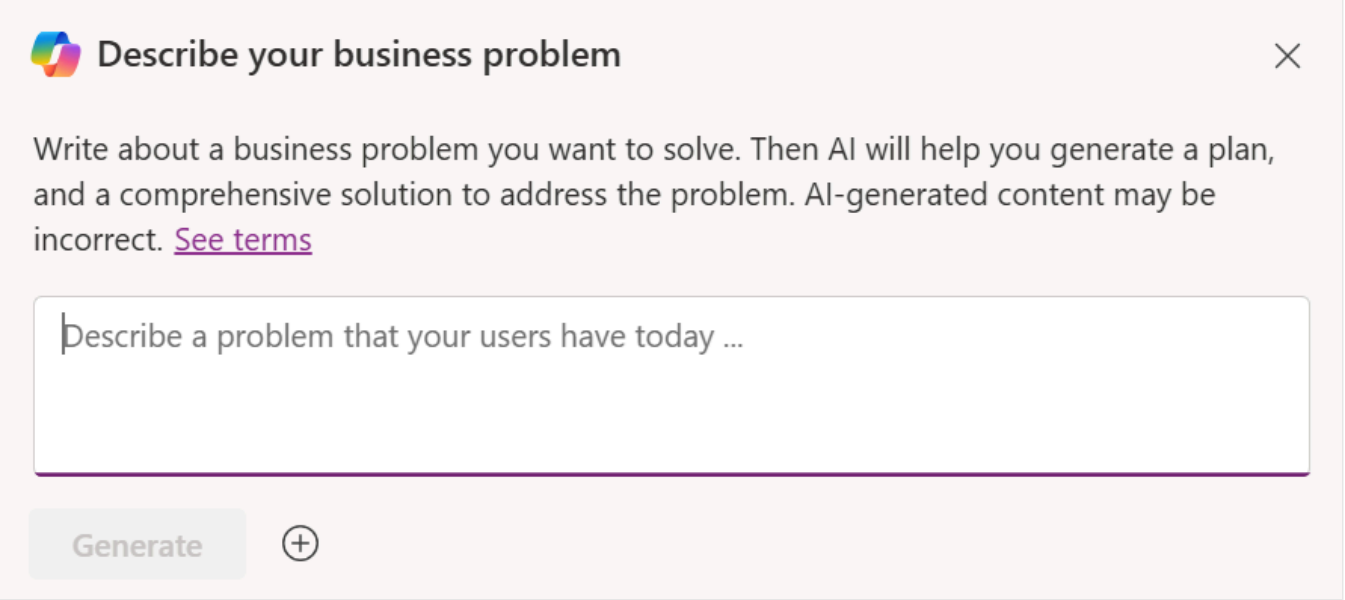
Month	Released version
December 2025	251218
January 2026	260130
February 2026	260227
March 2026	260331
April 2026	260430

Last updated on 04/30/2026

Overview of plans

Summarize this article for me

Plans in Power Apps is a copilot-first development tool that lets you quickly create comprehensive business solutions. Describe your business use case in natural language and add relevant images, like business process flows or screenshots of legacy apps. Plans generates a complete Power Platform solution tailored to your needs. It includes Microsoft Dataverse tables, canvas apps, model-driven apps, Power Pages sites, Power Automate flows, and Copilot Studio agents.



Prerequisites

> **Important**
>
> For Government Community Cloud users, plans is supported only in the Government Community Cloud (GCC). For more information, see Plans in Sovereign Clouds

To use plans in Power Apps, follow these steps:

1. Environment requirements

1. **Dataverse database required:** Make sure your environment includes a Microsoft Dataverse database.
2. **Copilot features enabled:** Plans is generally available and enabled by default. It can't be turned off except through Microsoft Support. A tenant administrator must contact Support to request that it be turned off.
3. **Environment must be in an eligible locale**

2. What if you don’t meet the prerequisites?

1. **Developer Environment Option**: If your current environment doesn’t meet the requirements but you can use developer environments, you're automatically routed to your own developer environment. There, you have permission to create Dataverse tables. Learn more in Create a developer environment.
2. **No Developer Environment Available:** If you can’t use or create a developer environment, you won’t be able to create plans in your current environment. In this case, switch to an environment where you have permission to create tables and start building your plan there.

3. Who can access plans?

The following security roles can see the entry point for plans on the Power Apps home page:

- System admin
- System customizer
- Environment maker: Are redirected to their own developer environment when they save tables.
- For custom security roles: You need the following privileges for each operation in plans:

Expand table

Operation in plans	Table	Privilege
Create a plan	Plan	Read
	Publisher	Read
	Entity	Read
	Attribute	Read
	Solution	Read
	Relationship	Read
	Entity	Create
Save tables in plan	Entity	Read
	User settings	Read
	Async operations	Read
	Entity	Append to

Operation in plans	Table	Privilege
	Entity	Create
Share a plan	Plan	Share

Availability

Plans is generally available and enabled by default. Check if the feature is available in your region. Learn more in Explore Copilot features by geography and languages.

Next steps

Create a plan

Last updated on 02/24/2026

Create a plan

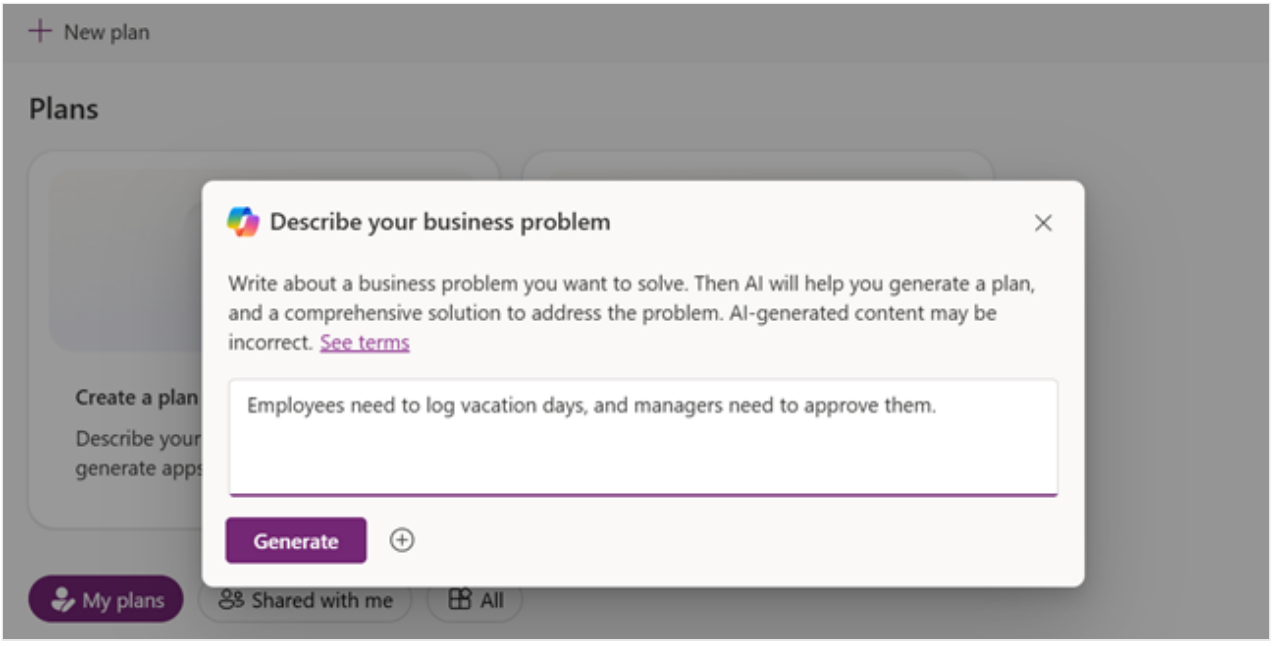
Summarize this article for me

Use plans to create user roles, data tables, and apps that address your business needs.

This article uses a sample scenario to show how plans works by building a solution for managing paid time off (PTO) requests for employees and managers.

1. Sign in to Power Apps.
2. On the Home screen, select **Start with a plan**
3. Select **Create a plan**
4. Enter your prompt in the text box, like the example. Add context such as process diagrams, data models, or screenshots of legacy apps. When you're done, select **Generate**.

Copilot prompt
`Employees need to log vacation days, and managers need to approve them.`



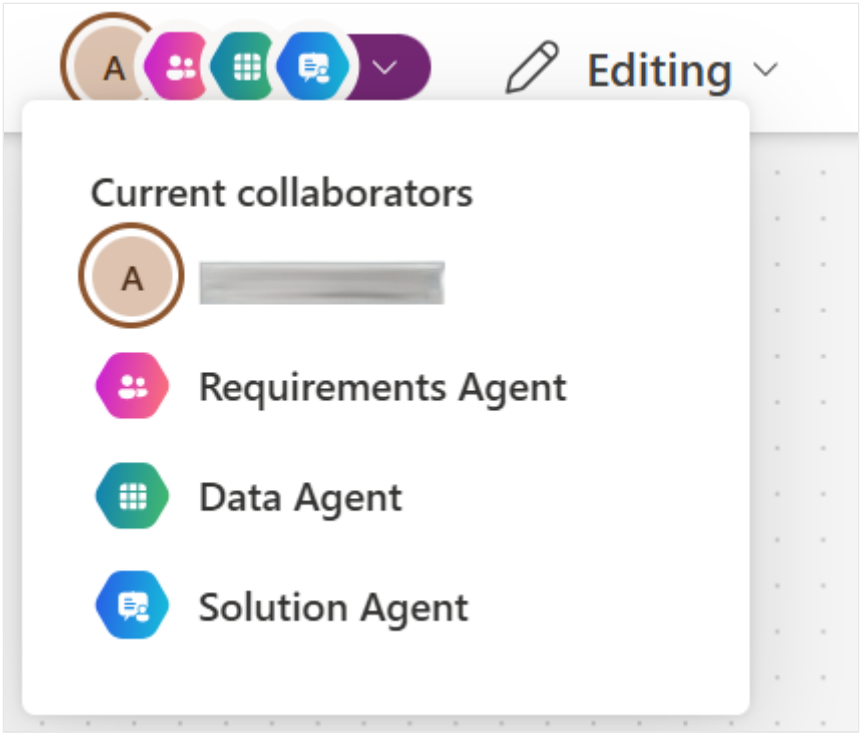
Copilot opens the plan and analyzes your business scenario based on the description.

Plan agents and copresence

When working within a plan, you can view the presence status of all collaborators, including plan agents. Plan agents are AI assistants that review your business problem and assist in generating a tailored plan to address your needs.

- The **Requirement Agent** examines the business problem and uses AI to generate user requirements. This step is first step in the plan.
- The **Data Agent** suggests a set of tables for storing business information, complete with recommended columns, data types, and relationships.
- The **Solution Agent** evaluates the needs, processes, and data to create a solution that includes technologies such as apps, flows, and AI agents to address your business problem.

As the plan builds, review and edit the outputs generated by the agents.



Copresence

Plans lets multiple authors use the same plan at the same time. The first author who enters the plan gets editing rights, and any author who joins later can only view the plan in read-only mode. You see other authors in the command bar, along with the agents in the plan. If you have read-only permissions, you can get editing rights after the author with editing permission exits the plan or after a period of inactivity. Plans supports up to 100 authors for copresence, but only one author can edit at a time. Multi-author coauthoring isn't supported.

> ⓘ **Important**
>
> Multiple users can't edit the plan at the same time.

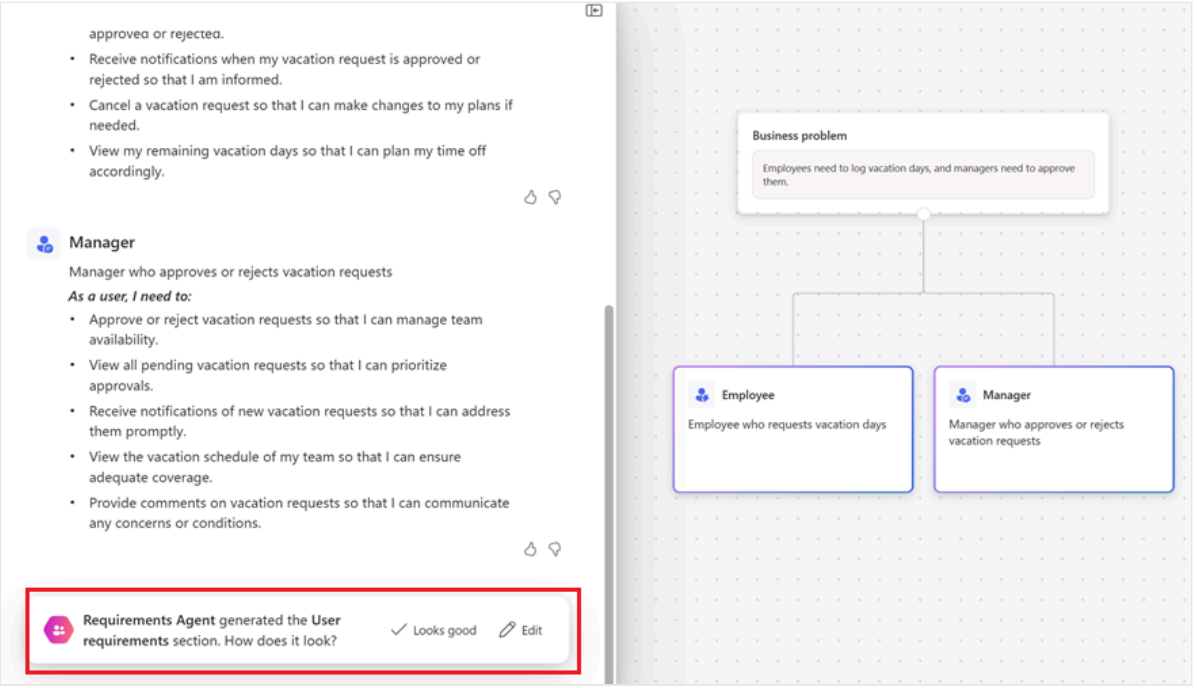
Generate user requirements

When a plan starts, the **Requirements Agent** identifies user needs based on your description.

In this scenario, two user roles are generated: the employee and the manager. Each user role includes a description and user needs. The user roles and needs appear in a bullet list and on the right side is a visual diagram.

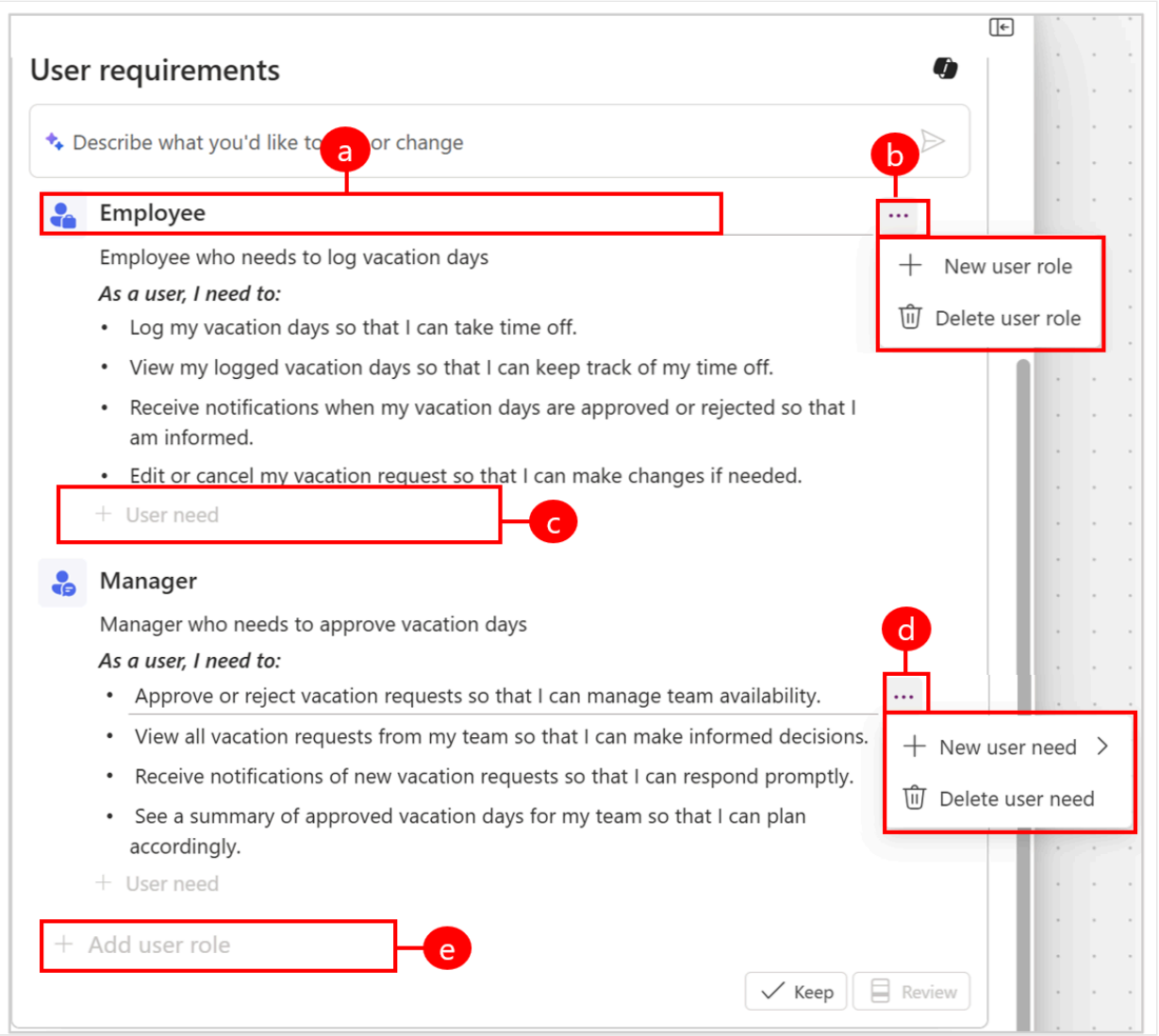
1. Review the user roles and needs. Then, choose one of the following options:

    - Select **Looks good** to move to the next step and generate a data model.
    - Select **Edit** to make edits inline or add new user roles or needs.



Making edits

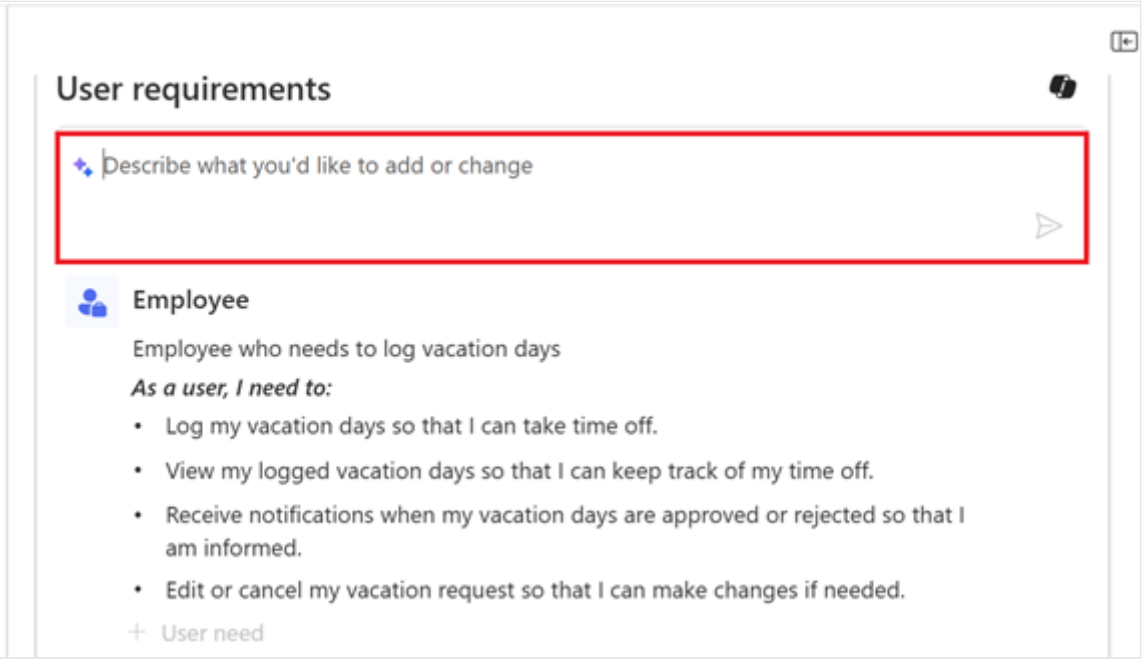
a. Inline editing provides these options:



Legend:

a. Select and modify a user role. b. Add a new user role or delete the selected user role.
c. Add a new user need.
d. Add a new user need either at top of selected user need or after it, or delete the selected user need.
e. Add a new user role.

b. Alternatively, use **Copilot** to provide feedback on the generated user needs.

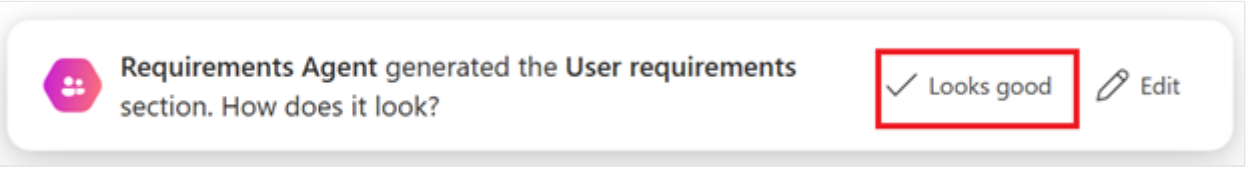


To make changes, provide a brief description of what you want to modify. Here are examples of what to ask Copilot to do:

- Add a user role for HR admin to monitor PTO across teams to manage payroll.
- Add a user need for employees to view PTO blackout dates.
- Remove the user need for managers for viewing vacation history of team members.

Select **Keep** or **Review** to finalize your edits.

2. When you're done, select **Looks good** to proceed to the next step and generate data tables.

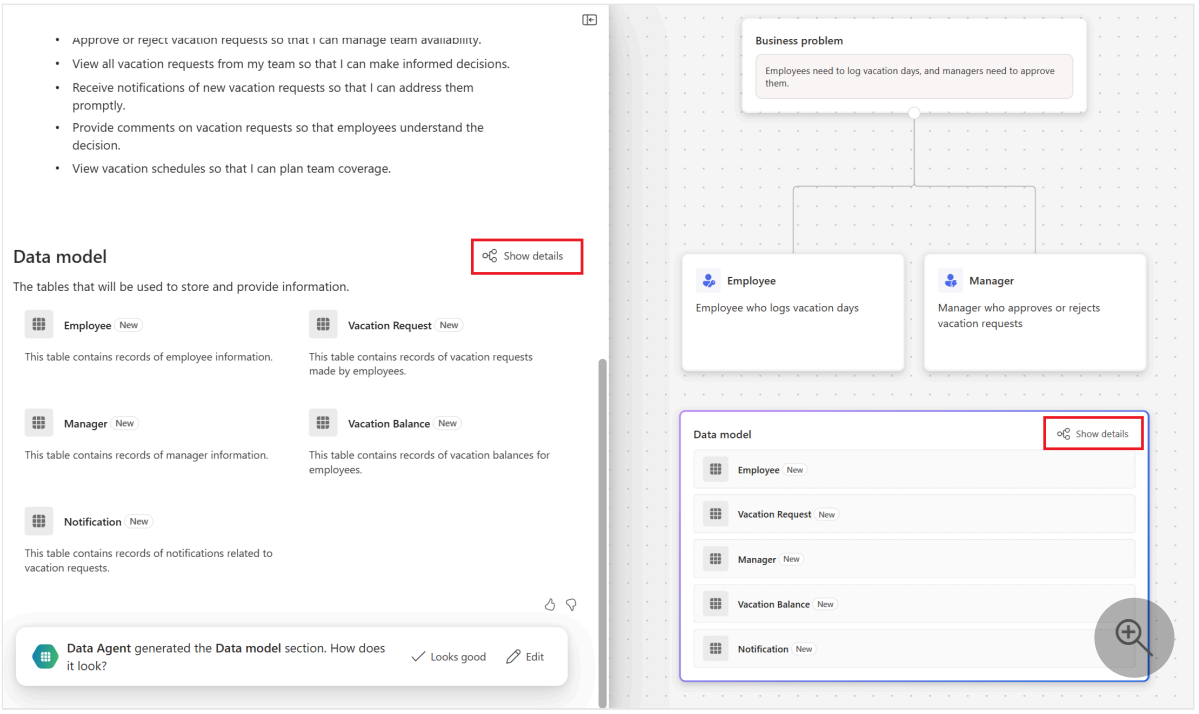


Generate data model

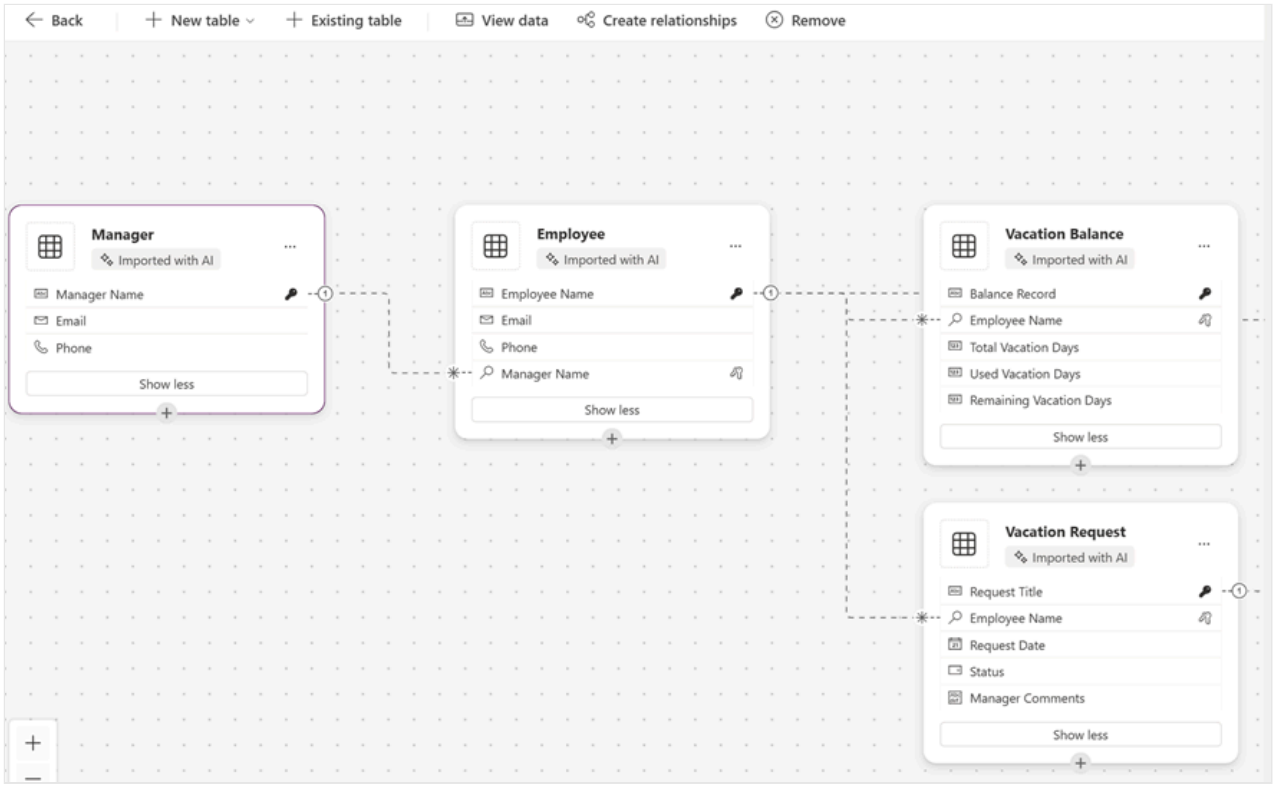
Next, the **Data Agent** suggests tables to store your business data.

1. Suggested data tables appear in the **Data model** section of the plan.

   - Select **Edit** and use Copilot to describe what you want to change or add.
   - Select **Show details** to view the data in a diagram, and edit it.



2. If you select, **Show details** it opens the data workspace. Use the data workspace to create tables, set up relationships, and view a data diagram. For more information, see Create and edit tables using Power Apps.

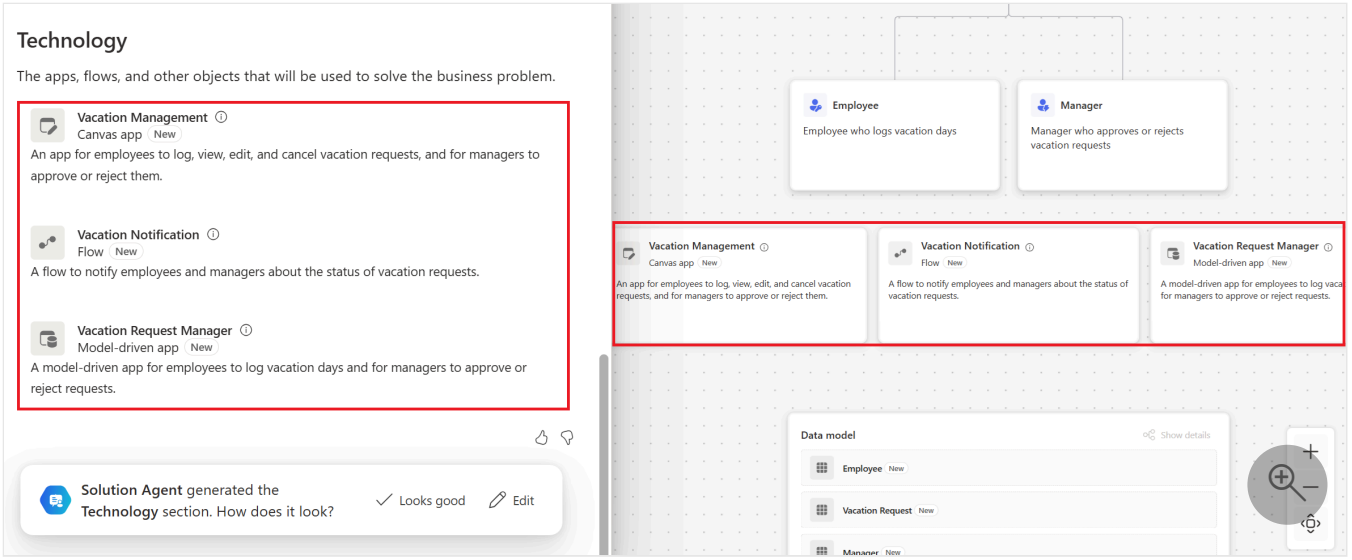


3. When you're done, select **Back**.

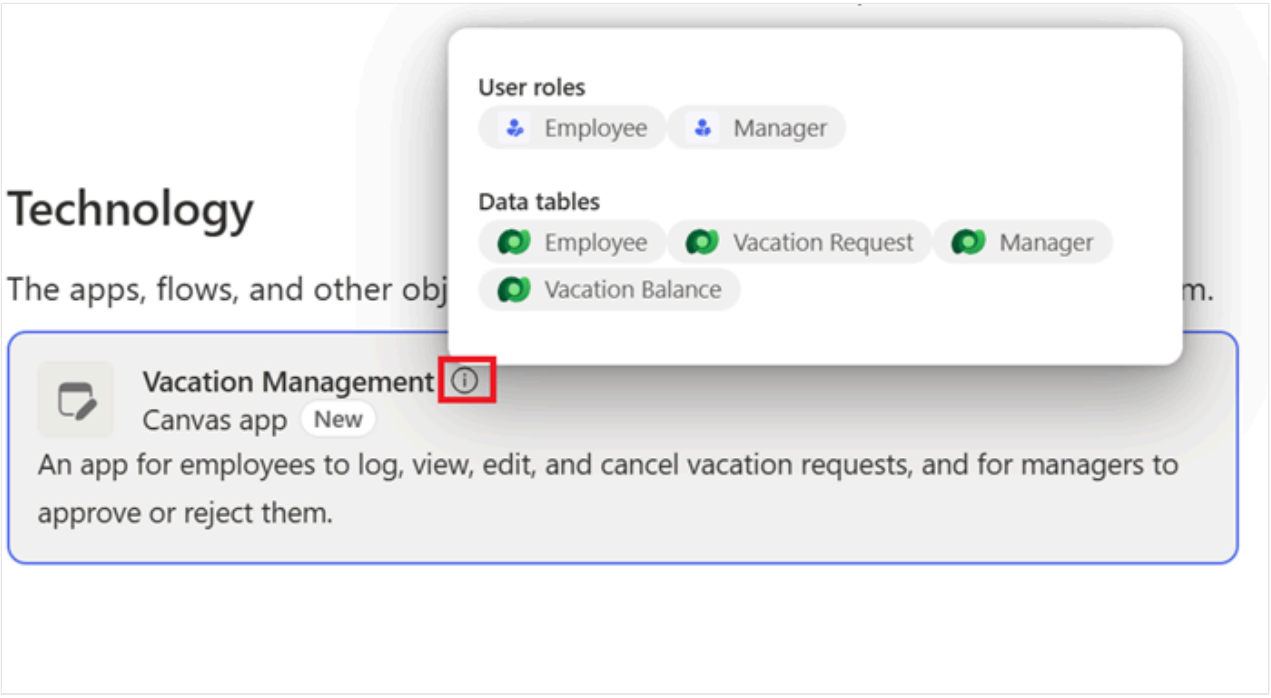
4. When you're ready to generate the technology proposal, select **Looks good** to continue.

Generate technology proposal

The **Solution Agent** analyzes the plan and proposes technologies tailored to solve your business problem. In this scenario, a canvas app, a model-driven app, and a Power Automate flow are created.



1. To view details of the proposed technologies, hover over the **information icon**. The pop-up info card shows the user roles being met and the data tables being used by the proposed technology.



2. Select **Looks good** to accept the proposed technologies, or select **Edit** to use Copilot and describe what you want to change.

Proposed technologies

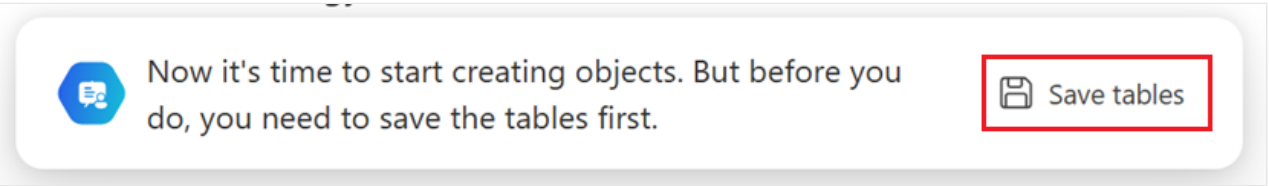
Plans suggests technologies from the following list:

- Apps like canvas apps or model-driven apps
- Power Pages websites
- Power Automate flows
- Power BI Report
- Copilot Studio Agents

Save tables

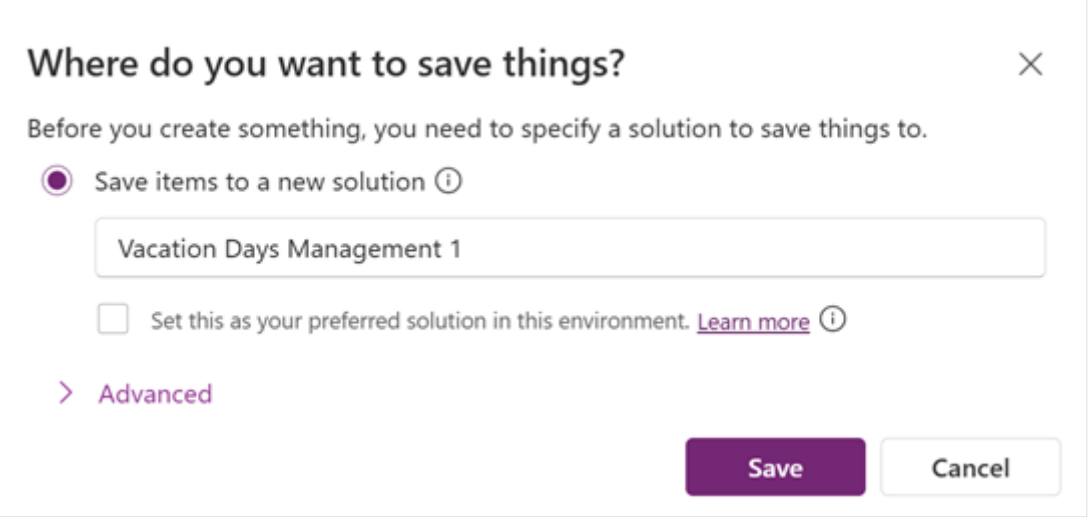
To create objects such as apps, flows, sites, and agents, you need to save your plan and tables.

1. Select **Save tables**.



2. Enter a solution name.

3. Choose a publisher or select an existing solution to contain the items generated from the plan, and then select **Save**.



The plan is stored in a solution. This action enables the **Objects** view in the plan, so you can switch between the plan and the solution view without leaving the plan. Learn more about solutions in Solution view.

> **Note**
>
> - The solution name must only contain letters, numbers, and underscores. For more information, see Create a solution.

- Anyone with the required plan privileges can access your plan.

You're now ready to build your solutions.

Your tables have been saved. To create and customize an object, just hover over it and select **Create**.

Share a plan

Share your plan with other makers to collaborate by assigning read or edit access.

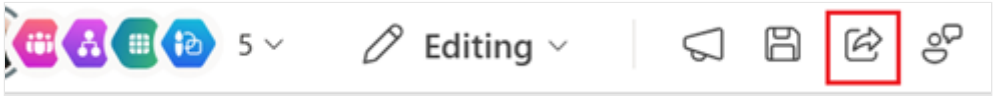
Plan collaborators need sufficient privileges to open and work with a shared plan. If your admin enabled plan level security roles in Power Platform Admin Center, makers with the System Admin security role can directly give collaborators plan-level privileges when sharing the plan.

For example, User 1 doesn't have enough privileges to see or edit plans. When the maker shares a plan with User 1, a custom security role is automatically created and assigned, which then gives them the needed plan privileges. The auto-generated security role appears in Power Platform Admin Center with the naming convention **Plan + Plan name + Role name + Plan ID**. To change assigned security roles, unshare and reshare the plan to assign the right security role under **Manage Access**.

If plan level security roles isn't enabled in Power Platform Admin Center, makers need to manually give collaborators the needed privileges so they can access the plan.

Follow these steps to share your plan:

1. Open your plan and then select **Share** in the top right corner.



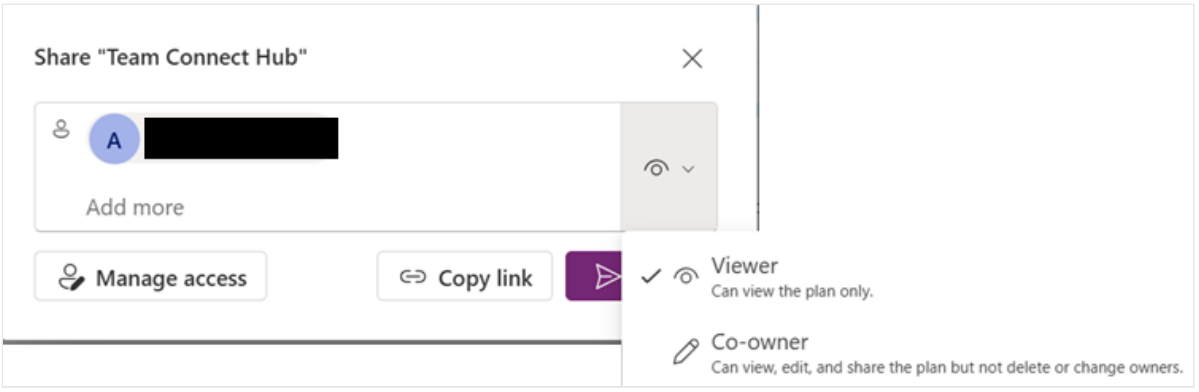
2. Select **Share**.

3. In the **Share** dialog box, type the names or alias of the user or security groups.

4. Select the down arrow to choose permissions for the plan. Options include:

    - **Viewer**: Can only view the plan.

- **Co-Owner**: Can use, edit, and share the plan but not delete or change owners.



5. Select **Manage access** to see access details, including current users and co-owners. You can also edit user access and assign security roles here.

6. When you're done, select **Share**.

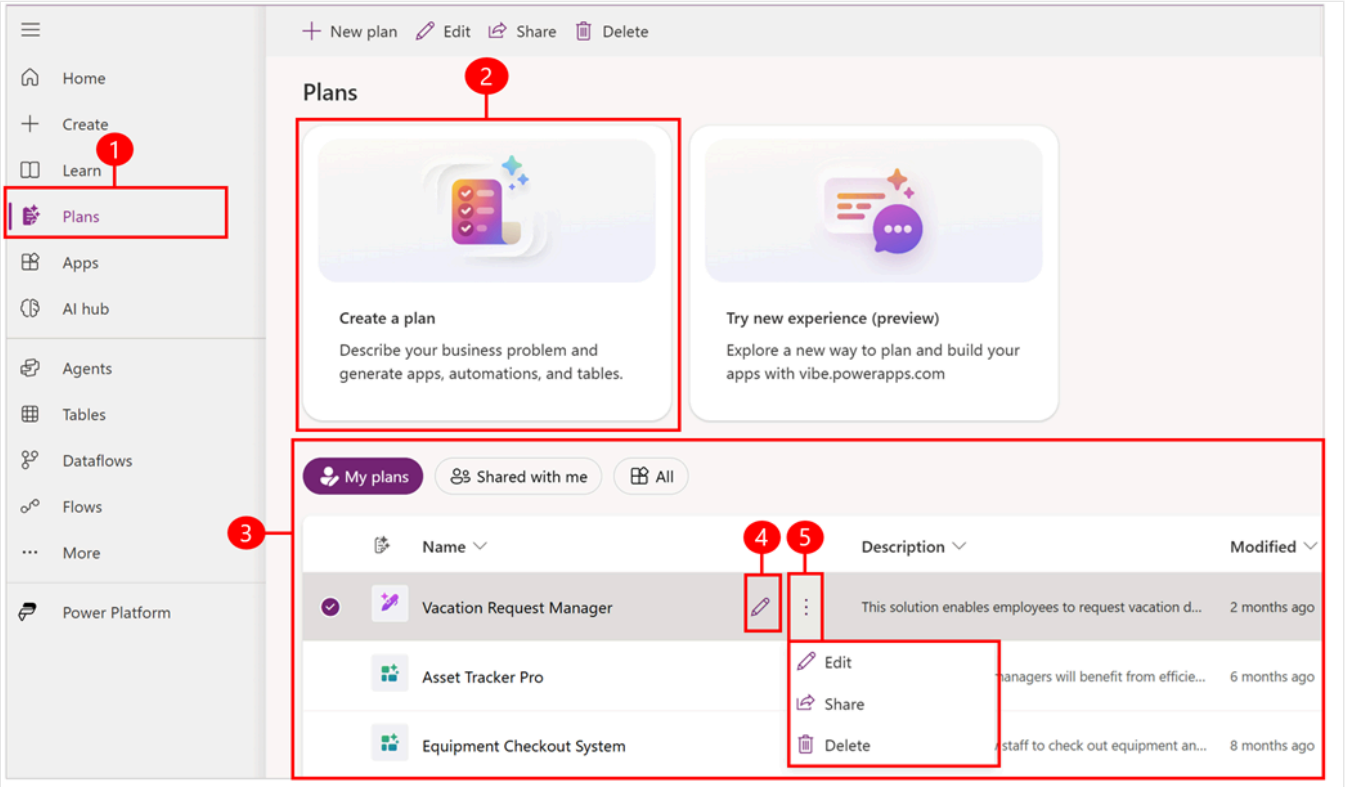
Export a plan to a PDF

The export to PDF feature lets you share plans with stakeholders. It includes all content from the document side of the plan, but not the diagram side. Use this feature to share user requirements, data models, and proposed data tables with leaders who don't have access to Power Apps.

- To export a plan to PDF open a plan and then select **Share** > **Export as a PDF** in the top right corner.

View and edit plans

Use the **Plans** menu in the left navigation pane to access and edit your plans.



Legend:

1. **Plans**: Access your plans.
2. **Create a plan**: Create a new plan.
3. **Your plans**: View your plans.
4. **Edit**: Edit the selected plan.
5. **Edit, share, or Delete**: Edit, share, or delete the selected plan.

Next steps

Build a solution from a plan

Last updated on 02/18/2026

Generate process diagrams

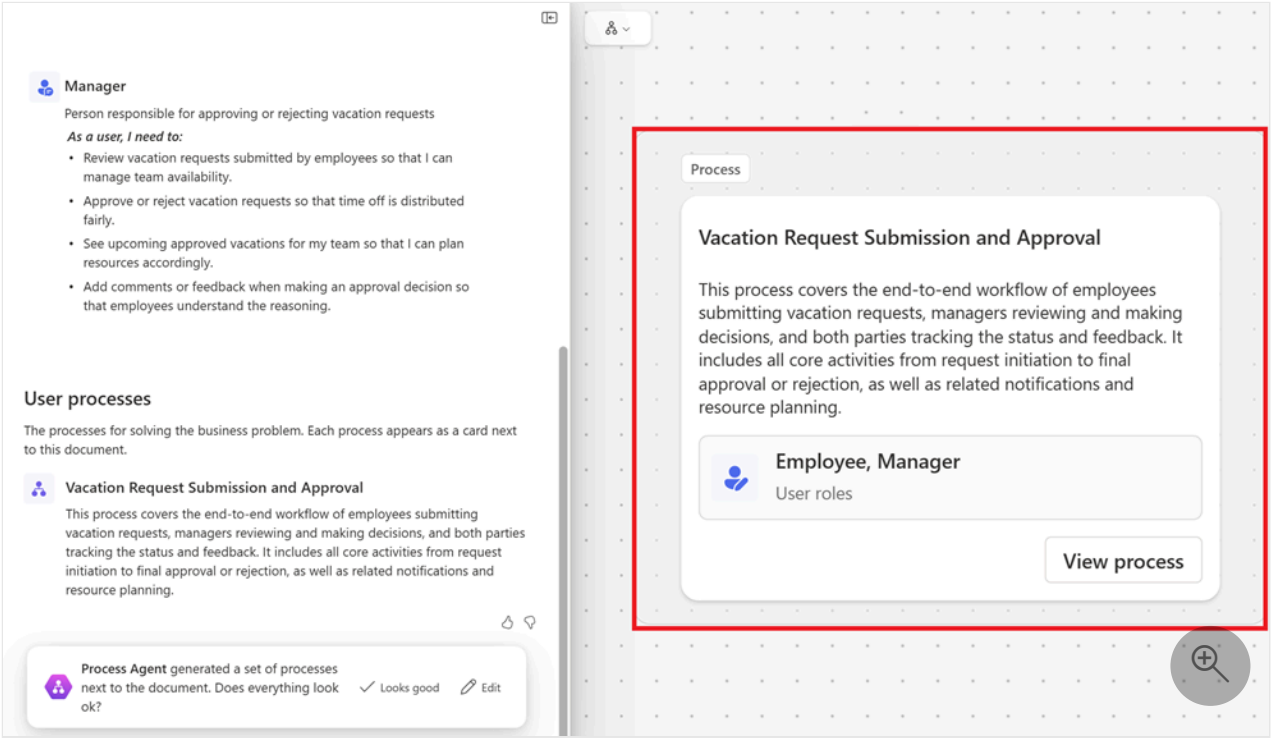
Transform complex workflows into clear, visual process diagrams with plans in Power Apps. Whether you're managing multi-step approvals or mapping customer journeys, these AI-generated diagrams help your team communicate better and work more efficiently on complex projects.

Optionally, include images of existing process diagrams as additional resources during the creation experience to enhance clarity. But if you add predefined processes, Copilot might just replicate what you provide, which can limit creativity and cause duplication or misinterpretation.

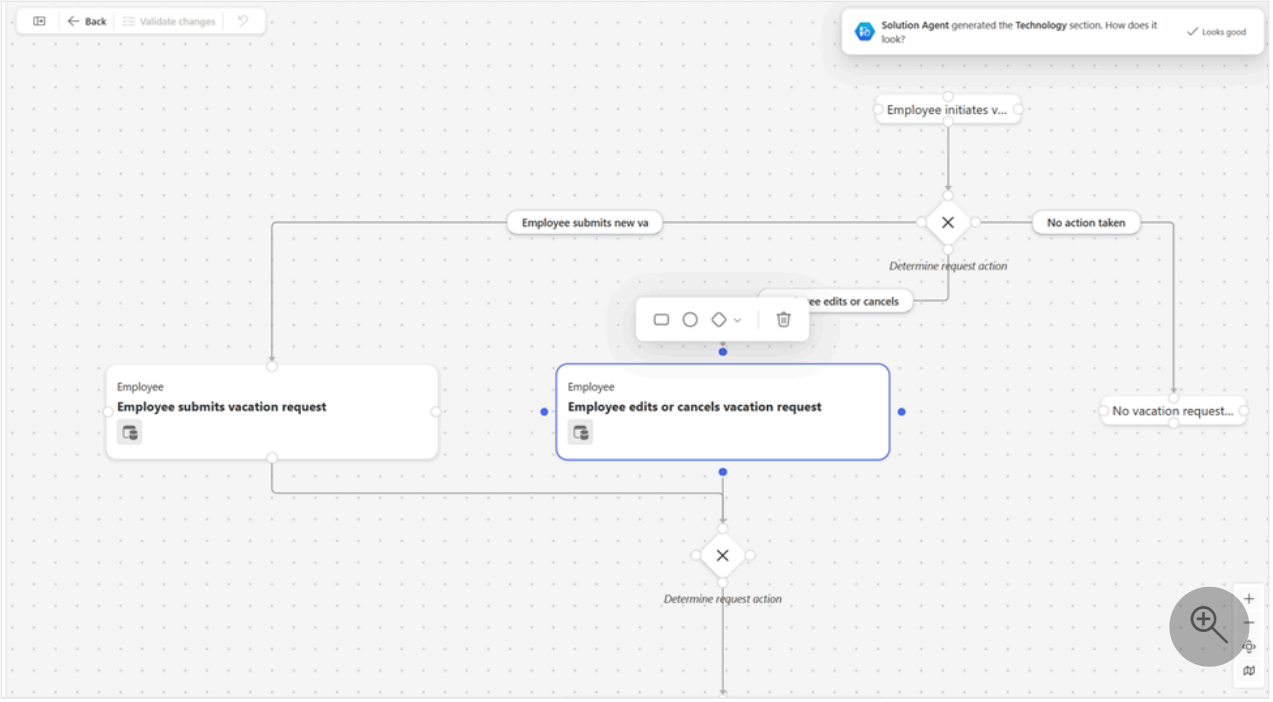
The process diagram appears on the right as you create a plan. The diagram generates after you create user requirements. If you prefer to see a plan overview instead, you can toggle off the process diagram.

Processes use a two level hierarchy.

- **Process stages**: An overview of the processes needed to address the business problem and user stories.



- **Process maps**: Select **View process** on a process tile to see detailed steps, events, and decision points for developing a process. Each activity shows the user role responsible for the action.



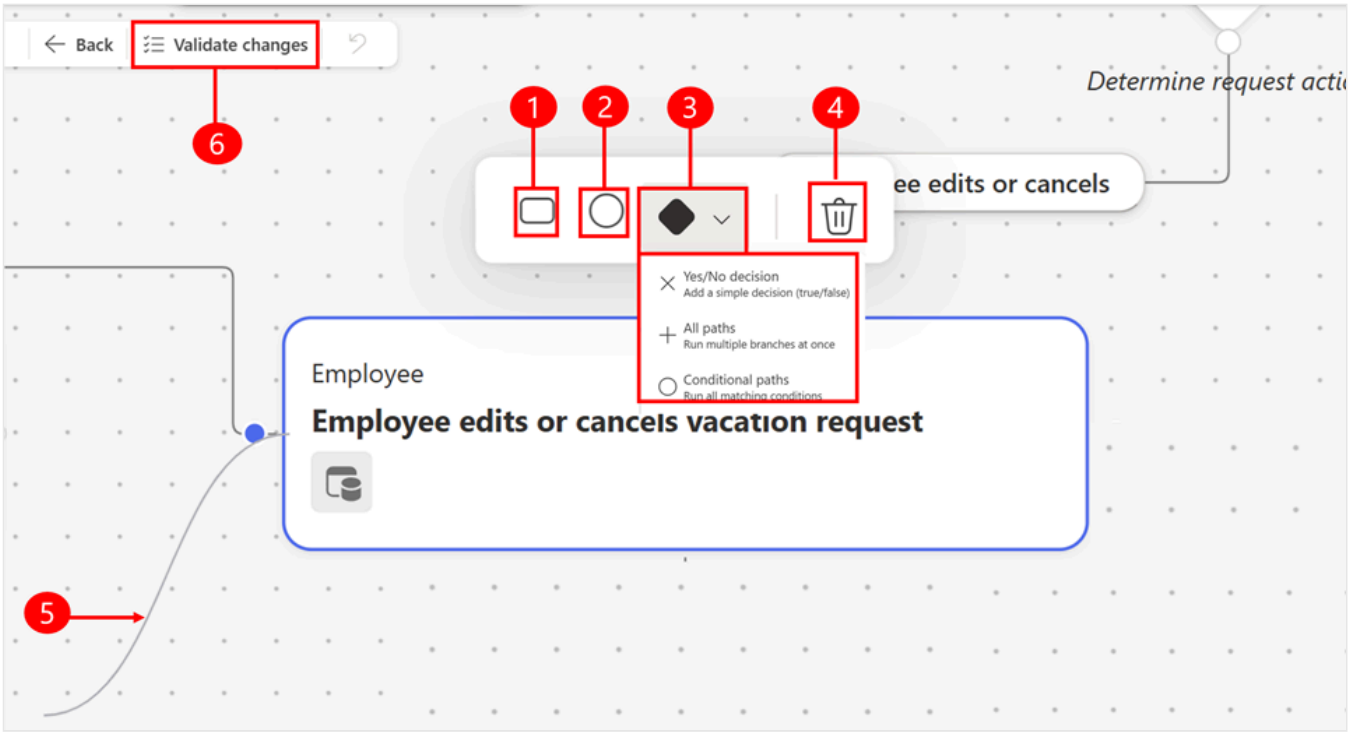
Understand a process

Process diagrams use three main types of nodes to represent different workflow elements:

1. **Events** control the flow of your proces:
    - **Start**: Triggers the beginning of your workflow
    - **Intermediate**: Marks a checkpoint during the process
    - **End**: Indicates successful completion
2. **Gateways** handle decision-making:
    - **Exclusive**: Creates a single path based on conditions (if/then logic)
3. **Activities** represent actionable work:
    - **Task**: Specific actions performed by users or systems

Edit a process

The process is an AI-first experience and might not always produce the results you expect. Use natural language to describe the changes you want, or manually edit the process. The AI then reassesses your changes and the overall process to generate a new process map.



Legend:

1. **Add step**: Add a new step.
2. **Add event**: Add a new event.
3. **Add decision**: Add a yes/no decision, run multiple branches at once, or run all matching conditions.
4. **Delete**: Delete a node. Remove connecting lines by selecting the backspace key.
5. **Connect nodes**: Add connecting edges between nodes by selecting a node anchor and dragging it to another node.
6. **Validate changes**: Confirm your changes. Changes aren't finalized or saved with the plan until you validate them. The process experience in plans is an AI-first experience, even when you make manual edits. When you validate changes, the AI reviews and merges them with the existing process. As a best practice, batch three to five manual changes at a time, then validate them. If you want to make more than five changes, try the natural language update first to save time and improve the accuracy of the AI-generated content.

Known limitations

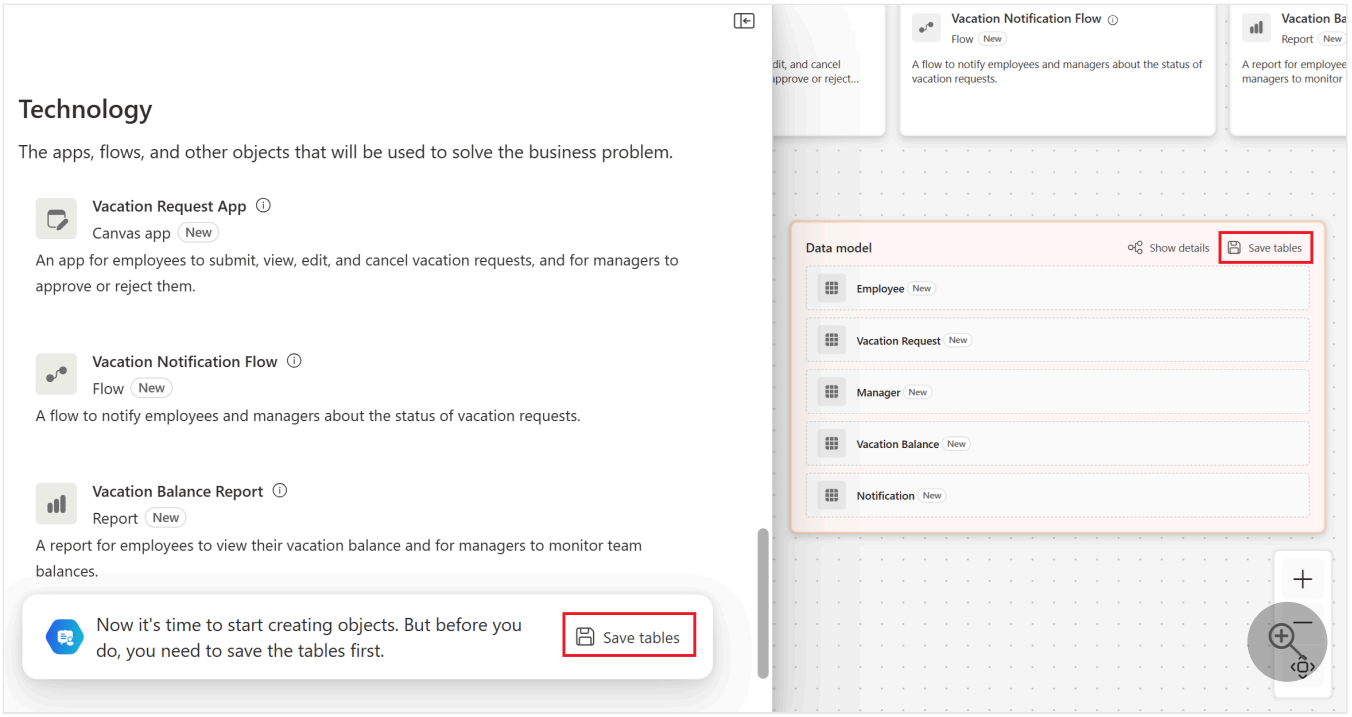
- You can't add events manually.
- The process doesn't directly affect technology proposals.
- You can validate only one set of changes at a time.
- You can't change node types. You can only delete and re-add them.

Last updated on 01/14/2026

Build a solution from a plan

Build a solution based on the objects plan's suggests for your business problem.

To follow this article, create and save a plan. Learn more in Create a plan.

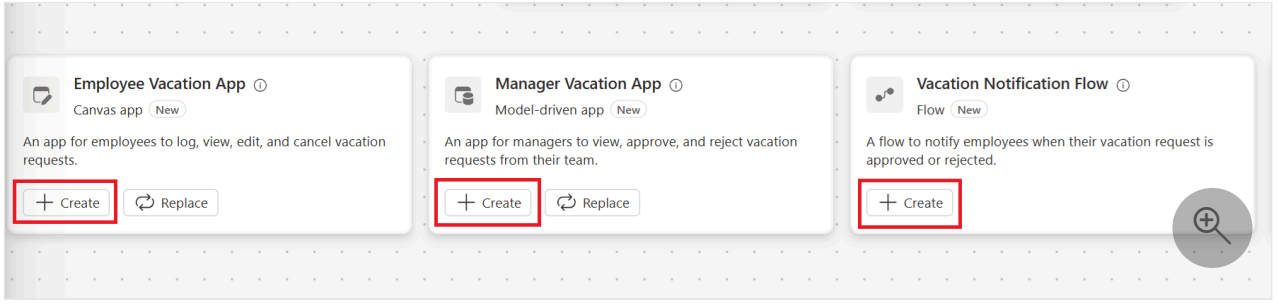


After you save a plan in a solution, create apps like canvas apps, model-driven apps, Power Automate flows, and Copilot Studio agents based on the solution proposal from a plan.



Create objects

1. Select **Create**.

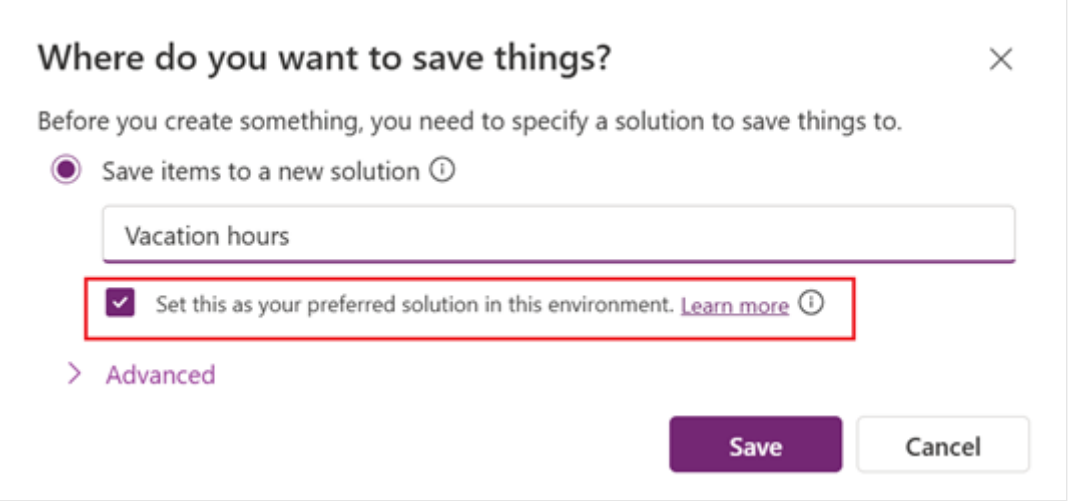


2. When you select **Create** on one of these tiles, these objects are created:

   - **Canvas app**: Opens Power Apps Studio with a data-connected app that starts for you. The app has responsive screens to view and edit data for each recommended

table, and a welcome screen to navigate the app. Learn more about customizing screens in Add and navigate screens. Before you use the app, save and publish it.

- **Model-driven app**: Opens the modern app designer with data tables already added. Before you use the app, save and publish it.
- **Power Pages site**: Opens Power Pages design studio with the initial setup, including layout design, domain configuration, and pages tailored to your goals. Use the studio to develop your website. Learn more in Overview of designing and building sites. To create a Power Pages site, you need the System administrator role in the environment and permissions to register an app in Microsoft Entra.
  - Save and publish the site before you use it. Save the site to the same solution as the plan for healthy application lifecycle management (ALM).
  - When you save a plan in a new solution, make sure to select the **Set this as your preferred solution in this environment** option to save the subsequent items created in this solution by default.



- **Power Automate cloud flow**: Opens the Power Automate using natural language to flow experience with a prefilled prompt based on the business problem, user story, flow description, and data sources. To use the flow, save and publish it.

> **Note**
>
> If you're using the preview experience to create flows, use a preview environment for consistent results. Learn more in Early release cycle environments.

- **Copilot Studio agents**: Opens Microsoft Copilot Studio, where you finish creating your agent and then publish it. An agent is an AI-powered assistant that simplifies

workflows, boosts productivity, and automates repetitive tasks. Plans includes agents as part of its recommended technologies. The plan prefills the agent with the following information:

- **Name**: Name of the agent.
- **Description**: Purpose of the agent.
- **Instructions**: Set of actions for the agent to take.
- **Knowledge**: All tables created by the plan are added to the agent as knowledge sources.

Review the instructions and update the agent as needed for your scenario. For example, add a trigger or an action to the agent. Test the agent to make sure it works as expected before you publish it.

Learn more about Microsoft Copilot Studio.

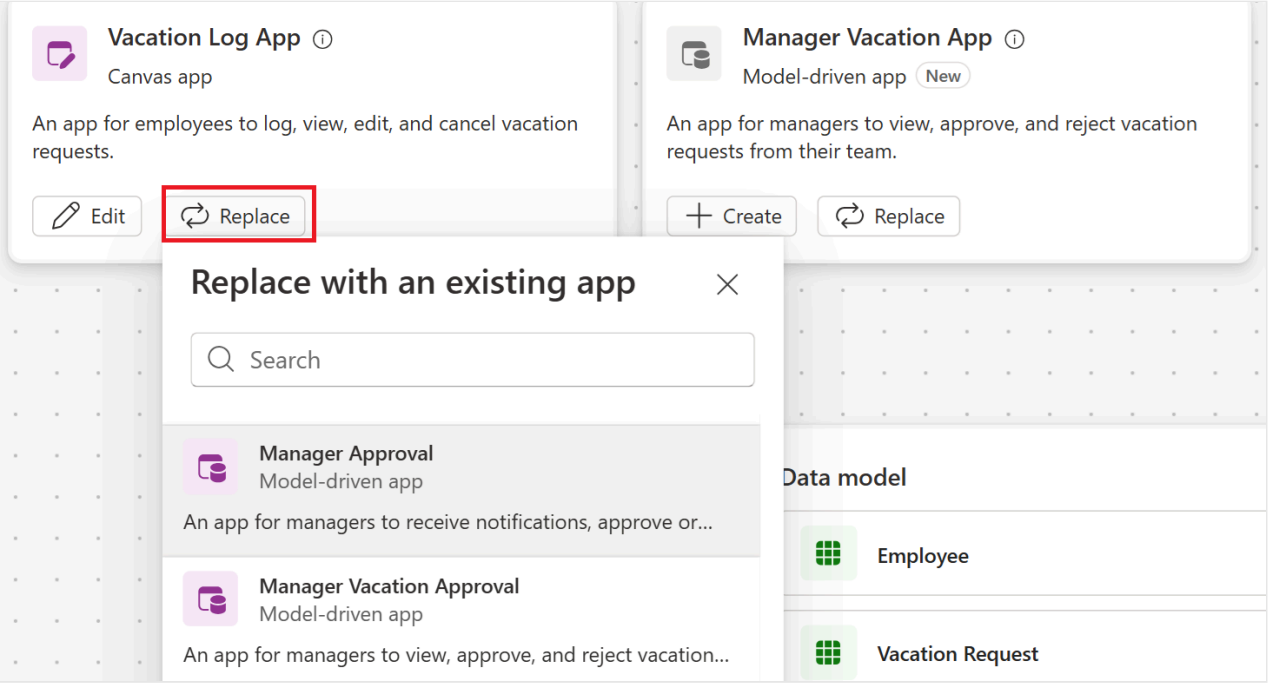
- Agent flows overview
- Add actions to custom agents
- Test your agent
- Key concepts – Publish and deploy your agent
- Key concepts - Analytics

- **Power BI dashboards**: Plans can recommend Power BI dashboards, but connecting a Power BI report with plan context is not currently supported.

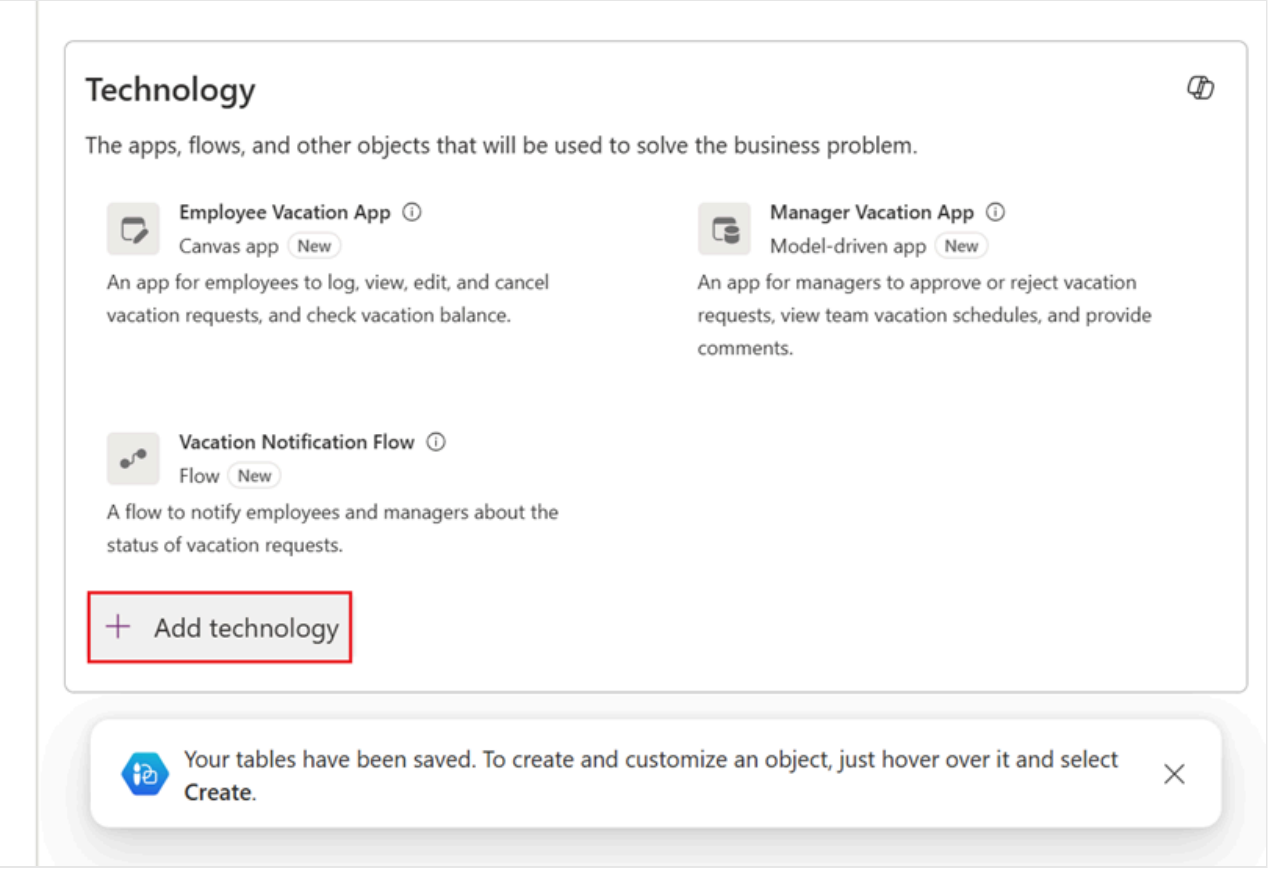
Use existing apps

When apps are proposes apps, replace a proposed app with an existing app or add an existing app to the plan. This approach helps if you already developed apps and want to use them in a new business solution. It saves time, reduces redundancy, and keeps your solutions consistent.

- To replace an app proposed in the plan, select **Replace** on the app tile you want to replace, and then select an app.



- To add an existing app that isn't related to the proposed scenarios, select **Add technology**, and then select an app.



Existing apps you add to the plan aren't included in a new solution. Use the solution view to see all solutions related to the plan.

Next steps

Manage your solution

Last updated on 12/10/2025

Manage your solution

10/02/2025

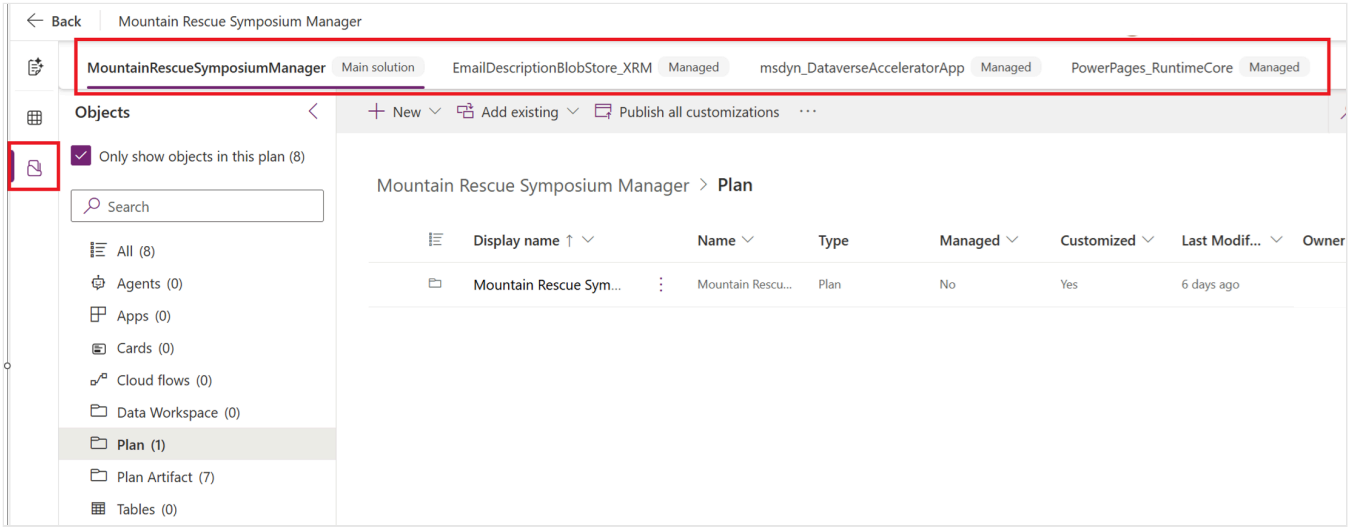
Business solutions built with plans promote healthy application lifecycle management (ALM) practices by using Power Platform resources. Use advanced ALM capabilities to deploy artifacts to target environments through managed solutions by packaging all components of a plan into a single solution artifact.

When you create a plan, the plan is stored in the main solution, which enables the **Objects** view in plans. Makers can switch between the plan and the solution view without leaving the plan.

A plan might be associated with multiple solutions. For example, adding an existing Microsoft Dataverse table to the plan's data model links the solutions associated with the Dataverse table to the plan.

The solutions associated with the plan display as tabs across the top when you select the **Objects** view. Select a tab to view objects in the solution. If there are more than five solutions, an overflow menu shows the remaining solutions. All solution objects are displayed, not just the ones included in the plan.

Learn more about solutions in Solution view.

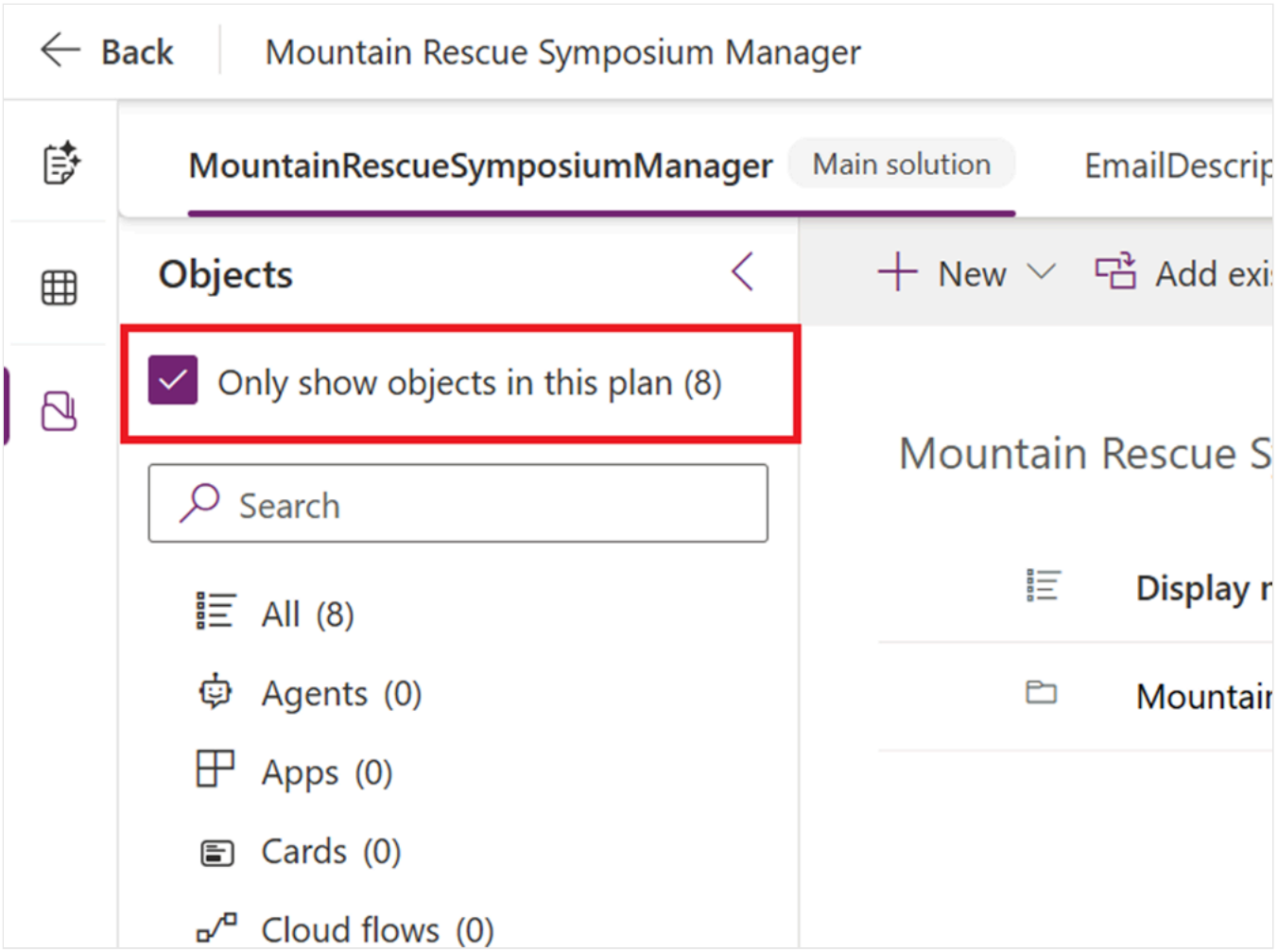


Show objects in this plan

Each plan has a main solution, but other solutions are associated with the plan when you add existing tables or apps from them. These associated solutions might include many objects beyond the ones you add to the plan.

To track which objects from the associated solutions are tied to the plan, select **Only show objects in this plan**. This option lets you view components explicitly designated as part of the

plan.



When you clear **Only show objects in this plan**, the complete contents of these solutions are shown. It helps you understand the broader context, overall composition, and potential dependencies.

Known issues

- When the **Only show objects in this plan** filter is on, some object categories with zero items still appear in the left tree view.
- Some existing canvas apps added to the plan might not filter properly with the **Only show objects in this plan** filter. This issue sometimes affects canvas apps that aren't connected to Dataverse.
- Intermittent caching issues occur. If associated solutions don't show up in the **Objects** view, resave the plan and refresh the **Objects** view to fix the issue.

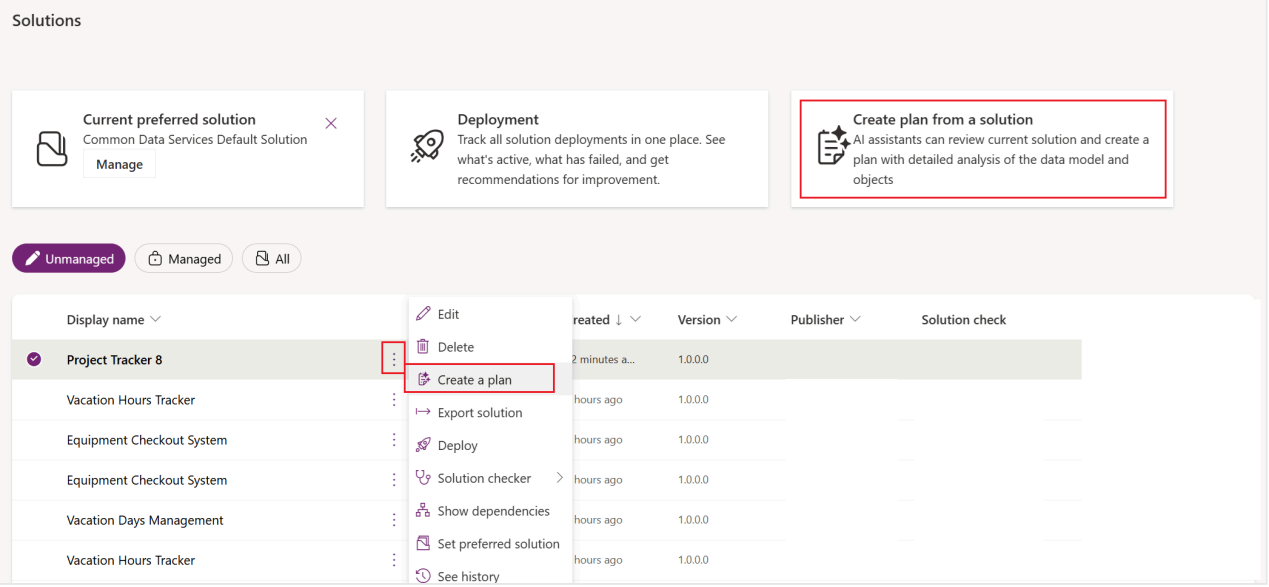
Related information

- Create a plan
- Create a plan from a solution

Create a plan from a solution

Learn how to create a plan for your existing solution. Plans generates a detailed document that describes your solution. The plan covers the business problem, user requirements like user roles and stories, the data model, and technologies like apps, flows, and sites. This feature saves time when you're trying to understand a solution's content and helps makers improve an existing solution.

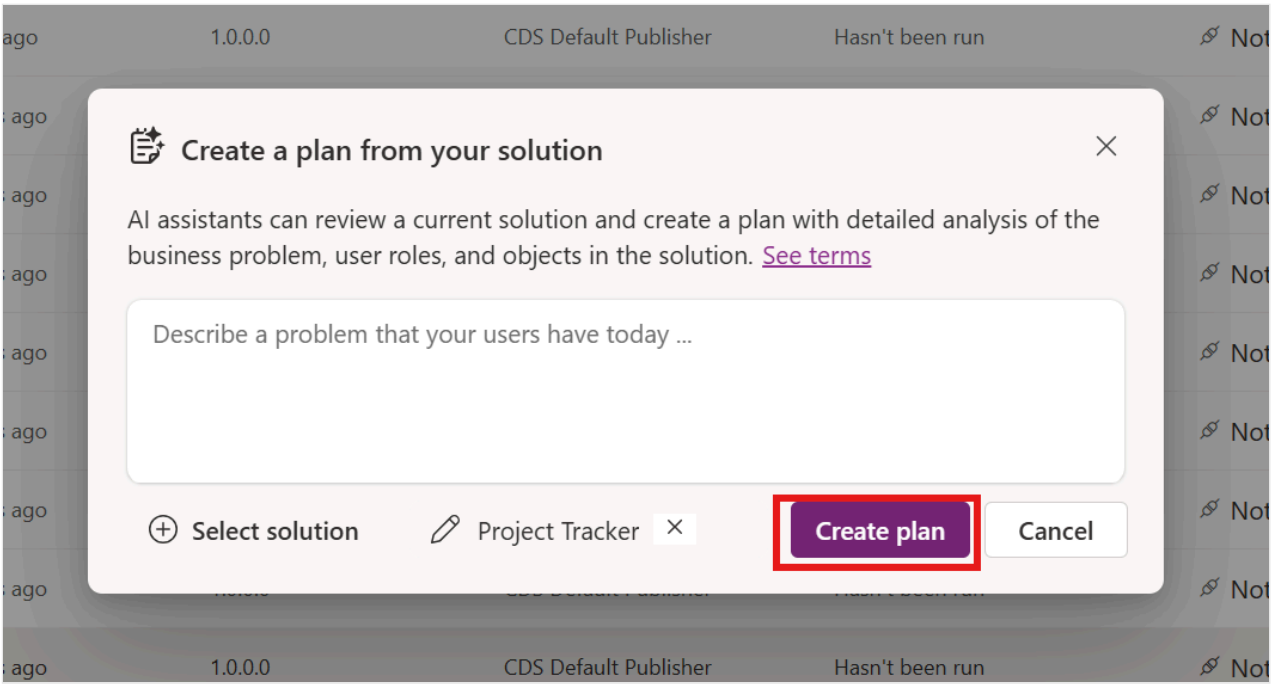
1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**.
3. Select **Create plan from a solution**. Or, from the list of solutions, select a solution, and then select **Commands** > **Create a plan**.



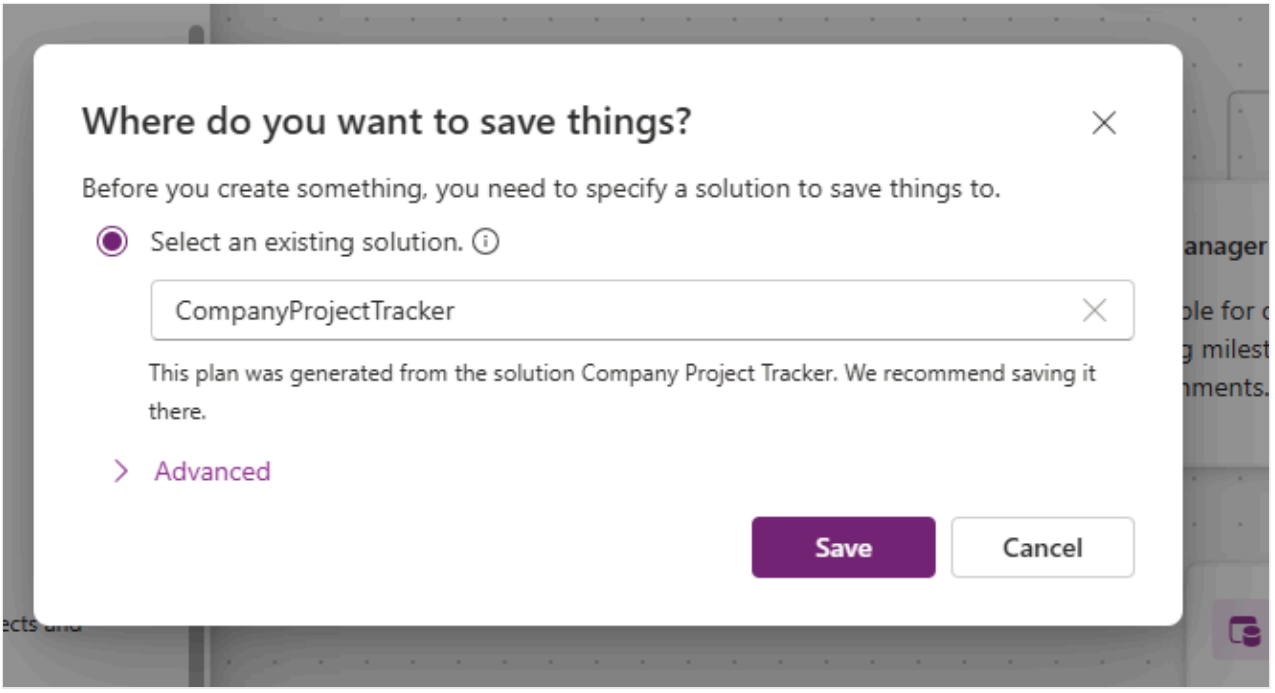
4. Select **Select solution** and then choose a solution.

> **Note**
>
> The solution must have at least one app and one associated table.

5. Select **Create plan**.



6. When the plan is created, save it. For an unmanaged solution, the plan is saved in the same solution. For a managed solution, the plan is saved in a new unmanaged solution by default.



Known limitations

- A solution needs at least one app and one associated table to create a plan.
- It's not recommended to create a plan from a default solution.

Related information

Create a plan

Last updated on 11/05/2025

Power Apps canvas apps documentation

Build business apps with canvas apps in Microsoft Power Apps. Connect to various data sources and create custom applications without writing code in traditional programming languages like C#.

Get started

OVERVIEW

What are canvas apps?

System and configuration requirements

Planning a Power Apps project

TRAINING

Get started with canvas apps in Power Apps

Create a canvas app

HOW-TO GUIDE

Create a blank canvas app

Create an app from conversation

Create a canvas app using Dataverse

TRAINING

Create canvas apps in Power Apps

Design the interface

HOW-TO GUIDE

Add and configure controls

Add and navigate screens

Add forms

TRAINING

Use the UI and controls in canvas apps

Customize

TUTORIAL

Galleries

Forms

Cards

Accessibility

Understand building blocks

CONCEPT

Connectors

Delegation

Data sources

Tables and records

Manage

HOW-TO GUIDE

Save and publish

Share an app

Edit an app

Licensing

Application Lifecycle Management (ALM)

CONCEPT

Create a canvas app within a Dataverse solution

Reuse canvas app components

ALM with Microsoft Power Platform

Integrate with other websites

Test, debug and optimize

HOW-TO GUIDE

Test using Test Studio

Debug using Live monitor

Analyze app telemetry using Application Insights

Performance tips

Reference and advanced learning

REFERENCE

Controls and properties reference

Formula reference

TRAINING

Use advanced data options and connectors

Master advanced techniques

What are canvas apps?

Canvas apps are custom business applications you can create in Microsoft Power Apps without writing code. Think of a canvas as a blank design surface where you can drag and drop components to build exactly the user interface you need. Whether you're creating a simple data entry form or a complex business workflow, canvas apps give you the flexibility to design solutions that work the way your business works.

With canvas apps, you can:

- Connect to data from hundreds of sources, including Microsoft 365, Dataverse, SharePoint, and other data sources
- Create responsive designs that work seamlessly across browsers and mobile devices
- Use AI-powered Copilot to build apps through natural language conversations
- Share your apps securely within your organization
- Embed apps directly in SharePoint, Power BI, and Microsoft Teams

This overview helps you understand how to get started building canvas apps and guide you through the key concepts for creating effective business solutions.

Build an app

The following articles provide guidance on building apps:

- Create a plan
- Create a blank canvas app from scratch
- Build apps through conversation with Copilot
- Create a canvas app using Microsoft Dataverse
- Create a canvas app with data from a list
- Create a canvas app based on Excel data

For more information on where makers can create apps, see Get started with Power Apps.

After you generate an app automatically, customize its default appearance and behavior based on your users' workflows. For example, change which types of data appear, how they're sorted, or whether users specify a number by typing it or adjusting a slider. Add and customize screens, galleries, forms, and other controls.

After you generate and customize an app automatically, create an app from scratch based on Dataverse, Excel, or another data source. By working from the ground up, you get flexibility in app design, flow, and controls, and you can use a larger variety of data sources.

If you're new to Power Apps and want to turn your ideas into a working solution, start with Planning a Power Apps project.

Connect to data

Canvas apps can connect to a wide variety of data sources. Choose the data source that best fits your scenario:

- **Microsoft Dataverse** – Use Dataverse as your primary data store for business applications with built-in security, logic, and data management.
- **SharePoint** – Build apps on top of SharePoint lists to extend your existing collaboration workflows.
- **Excel** – Create apps based on Excel workbooks stored in cloud storage like OneDrive or SharePoint.
- **SQL Server and Azure SQL** – Connect directly to SQL databases for enterprise data scenarios.
- **Other connectors** – Browse hundreds of available connectors for services like Dynamics 365, Salesforce, and more.

For guidance on linking Power Apps data to Microsoft Fabric, see Configure your environment and link to Microsoft Fabric.

Share and run an app

When you finish the app and save it to the cloud, share it with others in your organization. Specify which users or groups can run the app, and whether they can also customize and share it with more people in the organization.

Run your own apps, and any apps shared with you, on Windows, in a web browser, or on an iOS or Android device.

Licensing and trials

To get started with Power Apps, you need the right license for your scenario:

- **Free trial** – If you don't have a license for Power Apps, you can sign up for a free trial. Trial licenses provide access to most Power Apps features for a limited time.
- **Trial extension** – If your trial is expiring, contact your admin or visit the Power Platform admin center to explore extension options.
- **Developer environment** – If you don't have an organization account, establish a developer's account through the Microsoft Developer Program.
- **Licensing overview** – For detailed plan comparisons, see the Power Apps pricing page.

Troubleshoot common issues

If you encounter issues while building or running canvas apps, the following resources can help:

- **Known issues for canvas apps** – Solutions for common errors and known limitations.
- **Performance optimization** – Tips for improving app load times and responsiveness, including slow site performance.
- **Formula reference** – Complete reference for Power Fx formulas and functions used in canvas apps.
- **Coding guidelines** – Best practices for maintaining clean, performant canvas app code.
- **Dataverse troubleshooting** – Resolve issues with Dataverse connections, including mailbox configuration and solution imports.

If your issue isn't resolved through documentation, you can create a support request through the Power Platform admin center.

Learn more

- Want to turn your ideas into an app? Start with Planning a Power Apps project.
- Explore step-by-step, conceptual, and reference articles in the navigation pane on the left.
- Check out the webinars and video gallery to learn how to use Power Apps features and functions.

Share your experience

- Read and post in the Power Apps Community, where anyone who uses Power Apps can post a question for others to answer. Before you post a question, search the community to see if your question is already answered.
- Submit an idea to improve Power Apps in the Power Apps Ideas Forum.

- If you're a Power Apps admin for your organization, open a support ticket in the Power Platform admin center.

Last updated on 04/16/2026

Keyboard shortcuts for canvas apps

> **Note**
>
> Shortcuts can vary based on keyboard layout.

File

Expand table

Shortcut	Action
Ctrl+Shift+S (or Alt+P)	Save the app with a different name
Ctrl+S	Save the app with the same name or for the first time
Ctrl+Shift+P	Save the app and activate the publish dialog
F12	Download the app file (.msapp)

Ribbon

Expand table

Shortcut	Action
Enter	Run the selected command
Tab	Move between commands on the selected tab, then to the next tab
Alt+I	Select the **Insert** tab

Editing

Expand table

Shortcut	Action
Ctrl+A	Select all
Ctrl+X	Cut
Ctrl+C	Copy

Shortcut	Action
Ctrl+V	Paste
Ctrl+Z	Undo
Ctrl+Y	Redo
Ctrl+=, Ctrl+Shift+=, or Ctrl+Mouse wheel up	Zoom in
Ctrl+-, Ctrl+Shift+-, or Ctrl+Mouse wheel down	Zoom out
Ctrl+0	Fit canvas to page
`Enter` or `Shift` + `Enter`	Select a suggestion in a formula without adding a new line (Enhanced formula bar shortcuts is turned off)
`Enter`	Select a suggestion in a formula and create a new line (Enhanced formula bar shortcuts must be turned on)
`Shift` + `Enter`	Create a new line in a formula without keeping the suggestion (Enhanced formula bar shortcuts must be turned on)

> **Important**
>
> To use keyboard shortcuts for the formula bar, turn on **Enhanced formula bar shortcuts**. In Power Apps, go to Settings, and then select **Power App settings**. On the **Editing** tab, turn on **Enhanced formula bar shortcuts**.

Preview

Expand table

Shortcut	Action
F5	Open Preview mode
Esc	Close Preview mode, a dialog box, or a flyout pane

Canvas

Expand table

Shortcut	Action
`Tab`	Select the next control
`Shift` + `F11`	Set focus to the inline action bar
Hold down `Ctrl` while selecting or hold down `Shift` while selecting	Select multiple objects at once
`Right arrow`	Move the selected control to the right
`Left arrow`	Move the selected control to the left
`Up arrow`	Move the selected control up
`Down arrow`	Move the selected control down

Tree view

> **Note**
>
> These shortcuts work when the **Tree view** pane has focus.

Expand table

Shortcut	Action
F2	Rename a control
Esc	Cancel renaming a control
Ctrl+G	Group or ungroup controls
Ctrl+]	Bring a control forward
Ctrl+[	Send a control backward
Ctrl+Shift+]	Bring to front
Ctrl+Shift+[	Send to back

Resize

Expand table

Shortcut	Action
Shift+Left arrow	Decrease width
Ctrl+Shift+Left arrow	Decrease width slightly
Shift+Down arrow	Decrease height
Ctrl+Shift+Down arrow	Decrease height slightly
Shift+Right arrow	Increase width
Ctrl+Shift+Right arrow	Increase width slightly
Shift+Up arrow	Increase height
Ctrl+Shift+Up arrow	Increase height slightly

Text format

Expand table

Shortcut	Action
Ctrl+B	Cycle through levels of bold
Ctrl+I	Turn italics on or off
Ctrl+U	Add or remove underline

Alternate behavior

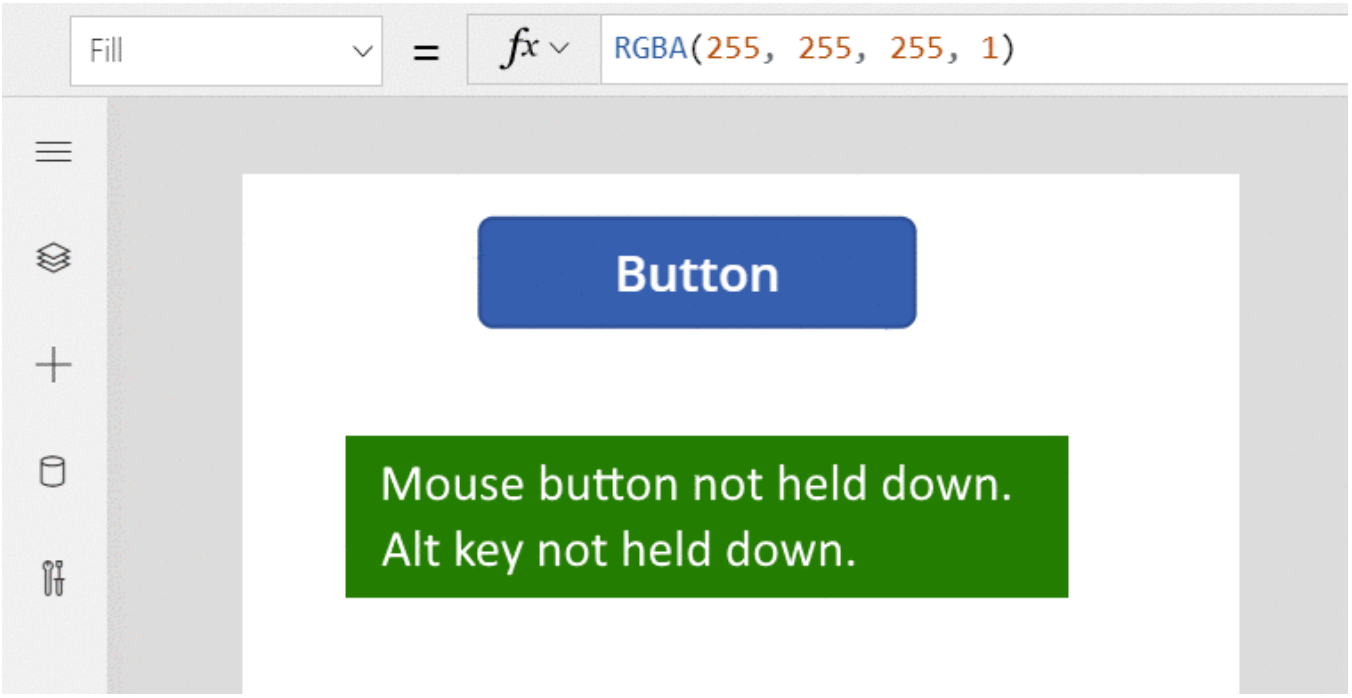
Expand table

Shortcut	Action
Alt or Ctrl+Shift	1. Before selecting a control, hide design elements so that you can interact with the controls as the app's user would. 2. After initiating a resize or reposition of a control, holding down these keys overrides any snap points.

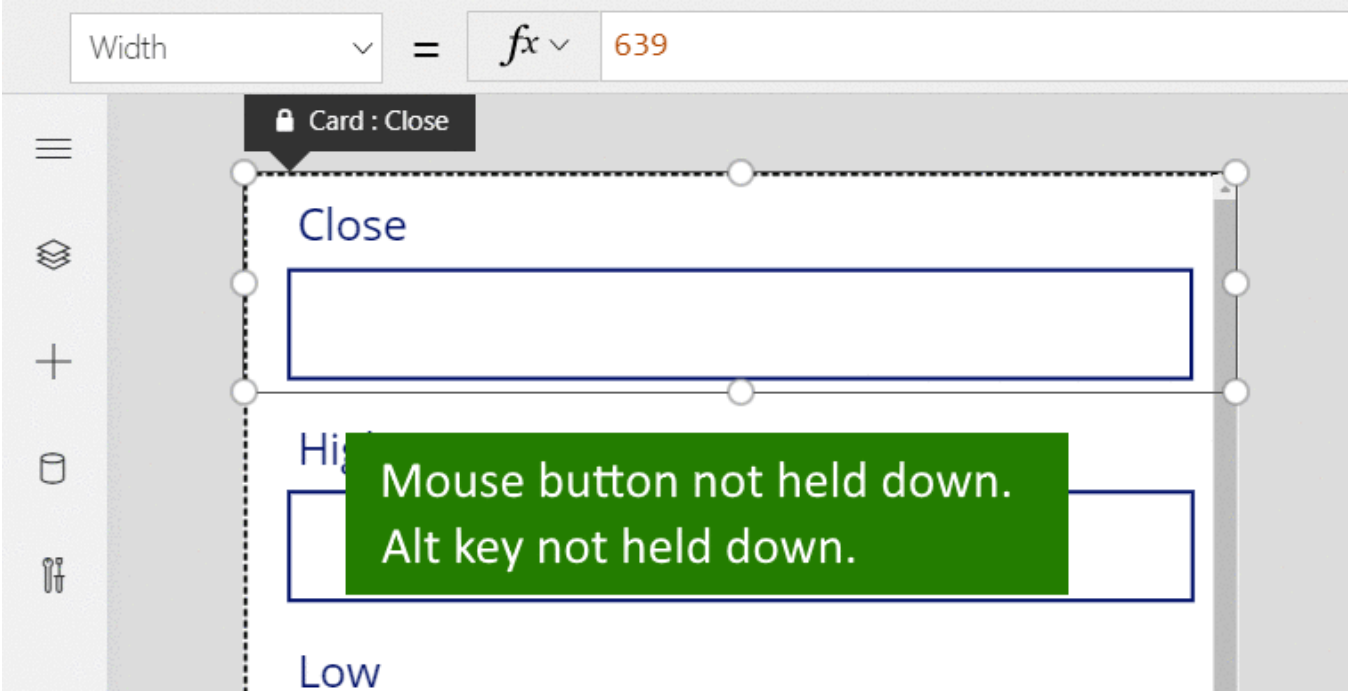
Like an Excel spreadsheet, canvas apps are live and running even when you're editing them. You don't need to switch to preview mode to test your app, so editing and testing are much faster. But this feature creates a challenge: How do you select a button control to resize it instead of selecting it to run the app logic?

When you're in design mode, selecting an object lets you edit it. You can move, resize, change properties, and configure the object. To override this default behavior, hold down the Alt or Ctrl+Shift keys *before* you select the object. This action treats the selection as if a user of the app selected it.

In the following animation, a button control is first selected for editing. Adorners appear around the control, and the formula bar shows the **OnSelect** property, ready to edit. After you release the button, hold down the Alt key and select the button control again. This time, the **OnSelect** property runs, and the notification appears, just like when a user selects the button in a running app.



You can also use the Alt key *after* you select a control to override snap points when moving or resizing. The next animation shows how to resize a data card in an **Edit form** control. At first, resizing is limited to specific snap points. Later, *without releasing the mouse button*, hold down the Alt key. This action overrides the snap points, so you can set any width with the mouse.



Other

Expand table

Shortcut	Action
F1	Open documentation
Ctrl+F6	Go to the next landmark. It navigates clockwise through the landmarks of a page (for example, headers, navigation bars, main content). Adding the Shift key (Ctrl+Shift+F6) moves through them in counterclockwise order.
Ctrl+Shift+F6	Go to the previous landmark. It navigates in counterclockwise order through the landmarks of a page (for example, headers, navigation bars, main content).
Shift+F10	Open a shortcut menu, for example, in **Tree view**

Last updated on 01/23/2026

Create a blank canvas app from scratch

Summarize this article for me

Power Apps canvas apps give you the flexibility to design custom business apps tailored to your specific needs. When you create a blank canvas app, you start with an empty design surface that you can customize with controls, data connections, and business logic.

Prerequisites

Ensure you have the Environment Maker security role, either directly or through a Microsoft Dataverse team that is part of the **AAD Security Group** category. Custom security roles aren't currently supported for Canvas App maker scenarios.

Create a blank canvas app

1. Sign in to Power Apps.
2. In the left navigation pane, select **Create** > **Start from blank**.
3. Choose an app size (format) such as responsive, tablet, or phone.

When you create the app, it opens in Power Apps Studio where you can start building and customizing it.

Next steps

Configure the app's functionality and, depending on your business scenario, add the necessary connections and data sources.

- Add a connection
- Add and configure controls
- Save and publish an app

Last updated on 02/24/2026

Create and edit canvas apps with AI code generation tools (preview)

[This article is prerelease documentation and is subject to change.]

This article describes how to use AI code generation tools, such as GitHub Copilot CLI and Claude Code, to create and edit canvas apps in Power Apps. By using this approach, you can integrate advanced code generation capabilities directly into your development workflow. You can create new canvas apps or iterate on existing ones by using natural language instructions.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Using AI code generation tools with canvas apps provides an alternative development approach that complements the UI-based experience in Power Apps (make.powerapps.com).

What you can do with code generation tools

- *Create new canvas apps* by using plain-language requirements.
- *Update existing canvas apps* by requesting changes or enhancements through your AI tool.
- *Work locally* by using your preferred IDE and development tools, syncing with live coauthoring sessions.

How it works

1. You describe what you want to build in natural language, for example, "Create a canvas app for tracking expense reports with an approval workflow."
2. The AI code generation tool uses installed canvas app skills to discover available controls, connectors, and data sources. It asks clarifying questions about your requirements.
3. The tool generates `.pa.yaml` files that define your app screens, controls, and Power Fx formulas.
4. The tool validates the generated code by using the canvas app authoring MCP server and fixes any errors.

5. Your canvas app syncs with Power Apps Studio through the coauthoring session.

Prerequisites

Before you start, make sure you have the required software and permissions described in this section.

Software requirements

Expand table

Component	Minimum version	More information
.NET SDK	10.0 or later	Download .NET SDK
GitHub Copilot CLI, Claude Code, or other code generation tool	Latest	GitHub Copilot CLI or Claude Code

Additional requirements

- A Power Platform environment with a canvas app.
- Power Apps Studio open with **coauthoring enabled** for your canvas app.
  - To enable coauthoring, open your app in Power Apps Studio, go to **Settings** > **Updates**, and turn on **Coauthoring**.

Install the canvas apps plugin

To install the canvas apps MCP plugin, run the following commands in either Copilot CLI, or Claude Code:

1. Add the Power Platform Skills marketplace plugin:

```shell
/plugin marketplace add microsoft/power-platform-skills
```

2. Install the canvas apps plugin:

```shell
/plugin install canvas-apps@power-platform-skills
```

After you install the plugin, you need to configure the canvas app authoring MCP server before you can create or edit apps. You can configure the server by either of the following options:

- Run the `/configure-canvas-mcp` command to set up the MCP server connection.
- Describe what you want to build - the tool detects that the MCP server needs to be configured and guides you through setup.

Configure the canvas app authoring MCP server

The canvas apps plugin uses an MCP (Model Context Protocol) server to communicate with Power Apps. Before creating or editing apps, you need to configure this server connection:

1. Open your canvas app in Power Apps Studio and ensure **coauthoring** is enabled.
2. Copy the Power Apps Studio URL from your browser.
3. Run `/configure-canvas-mcp` in your AI tool, or ask the tool to configure the MCP server.
4. Provide the Designer URL when prompted. The tool extracts your environment ID, app ID, and cluster information automatically.

By using the MCP server, your AI tool can list and describe available controls, discover APIs and data sources, validate app YAML, and sync app state from live coauthoring sessions.

The canvas-apps plugin repository includes control documentation, design guidance, technical reference, and workflow instructions necessary to create code adhering to canvas app requirements. Refer to the repository for information on accessing and using these resources with your preferred tool.

Skills overview

The canvas apps plugin provides these skills for working with canvas apps.

Expand table

Skill	Command	Description
Generate canvas app	`/generate-canvas-app`	Create a new canvas app from a natural language description
Edit canvas app	`/edit-canvas-app`	Edit an existing canvas app from a natural language description
Configure Canvas MCP	`/configure-canvas-mcp`	Register the canvas app authoring MCP server with your AI tool

These skills enable you to describe what you want to build and have the AI tool generate complete `.pa.yaml` files for your canvas app, validate them against the Canvas Authoring server, and sync changes with your live Power Apps Studio session.

Create a new canvas app

Follow this workflow when building a new app from scratch.

1. Start a conversation with your AI tool. Describe what you want to create, including the layout and what data you want to include (which Dataverse tables, or connectors to use). Be as specific as you want - the more vague you are with the request, the more details the agent tries to fill in itself. You can also attach or provide an image or other materials to help guide the visuals, theming, and layout. For example:

    - "Create a canvas app for tracking inventory with a searchable list and detail view"
    - "Build a canvas app for employee onboarding with a multi-step form"
    - "Make a canvas app dashboard showing sales metrics with charts and KPIs"
    - "Create a canvas app for field inspections with photo capture"

2. Answer clarifying questions. The AI tool discovers available controls and data sources using the MCP server and asks questions to understand your requirements. Be specific about business needs and data requirements and mention any specific UI components or layout preferences.

3. Review code and validate. The AI tool generates `.pa.yaml` files for each screen and validates them using the canvas app authoring MCP server. The tool fixes any validation errors automatically. Your changes sync with the live coauthoring session in Power Apps Studio.

4. Test and iterate. Open your canvas app in Power Apps Studio to preview and test. If you need to make changes, return to your AI tool and describe the updates using natural language.

Edit an existing canvas app

Use this workflow to update an app that already exists in your coauthoring session.

1. Sync the existing app. In your AI code generation tool, ask to edit your canvas app. The tool syncs the current app state from the coauthoring session, pulling all existing screens and controls into local `.pa.yaml` files. For example, "I want to edit my expense tracking canvas app."

2. Describe your updates. Tell the AI tool what changes you want to make. For example:

    - "Add a filter to show only pending expenses"
    - "Change the home screen layout to use a card-based grid"
    - "Add a new screen for viewing expense history with charts"
    - "Update the form to include a dropdown for expense categories"

3. Review, validate, test, and iterate. The AI tool generates updated `.pa.yaml` files based on your requested changes and validates them with the Canvas Authoring server. Follow the same review and test process described in the "Create a new canvas app" section. Continue iterating with natural language instructions until the app meets your requirements.

Troubleshooting

App doesn't reflect changes

If your changes don't appear in Power Apps Studio:

1. Verify the connection to the canvas app authoring MCP server by asking the tool to list available controls.
2. Ensure coauthoring is enabled in your Power Apps Studio session (**Settings** > **Updates** > **Coauthoring**).
3. Check that the MCP server is configured with the correct environment ID and app ID by running `/configure-canvas-mcp` again.

MCP server connection problems

If the canvas app authoring MCP server isn't responding:

1. Verify that the .NET 10 SDK is installed by running `dotnet --version` in your terminal.
2. Ensure your Power Apps Studio session is still active and coauthoring is enabled.
3. Re-run `/configure-canvas-mcp` with a fresh Designer URL.

Revert to a working version

If recent changes break your app or make problems worse, ask the AI tool to roll back to a previous working version:

"The recent changes broke the app. Please revert to the last working version."

The AI tool then:

1. Syncs the current state from the coauthoring session
2. Identifies the changes that you made
3. Restores the previous working code
4. Validates and resyncs the stable version

Best practices

- Start simple. Begin with a basic version of your app and iterate to add complexity.
- Test frequently. Preview your app in Power Apps Studio after each significant change.
- Be specific. Provide detailed requirements to get better initial results.
- Use existing patterns. Reference similar apps or UI patterns when describing your requirements.
- Validate generated code. Always review the generated `.pa.yaml` files to ensure they meet your organization standards and compliance requirements.
- Bold design choices. Don't settle for generic layouts - describe the visual style and aesthetic direction you want.

> **Important**
>
> While AI code generation tools make a best-effort attempt to generate complete, production-ready code with accessibility and security best practices, you're ultimately responsible for validating the code. Ensure the generated code meets your organization standards, policies, and compliance requirements.

Related content

- Create a canvas app from scratch
- Power Apps canvas app overview

Last updated on 04/06/2026

Build apps through conversation with Copilot

Create apps without writing code or designing screens. By using Copilot in Power Apps, you use natural language to describe what you want your app to do, and AI does the rest.

Copilot is a feature in Power Apps that helps you build apps with AI assistance. From the Power Apps home screen, select **Start with data**, and then select **Create new data** to open the table editor with Copilot. Enter the type of information you want to collect, track, or show in your app, and Copilot generates one or more Microsoft Dataverse tables that you use to build your canvas app.

Prerequisites

- Check the prerequisites and region availability in Copilot in Power Apps overview (preview).
- Depending on where your environment is hosted, you might need to enable data movement across regions. For more information, see Copilots and generative AI features that are available when you enable data movement across regions. You need access to a Microsoft Dataverse database in your environment to use this feature. For more information, see Add a Microsoft Dataverse database.
- Check if this feature is available in your region. For more information, see Explore Copilot features by geography and languages.
- Assign the system customizer security role in the environment.

Create an app by using Copilot

Let's create an app to track housekeeping tasks for a hotel so you can see how Copilot works.

1. Sign in to Power Apps.
2. On the home screen, select **Start with data**.
3. Select **Create new data**.

   The **Create new tables** workspace opens with a blank table and the Copilot panel on the right side.

4. In the Copilot panel text box, enter your prompt, like the example prompt in the following section.

   Copilot prompt

   ```
   Create tables to track hotel housekeeping tasks including room numbers, task types, staff assignments, and task status
   ```

5. Select **Submit** or press **Enter**.

Copilot creates one or more Dataverse tables with data that includes typical hotel housekeeping tasks.

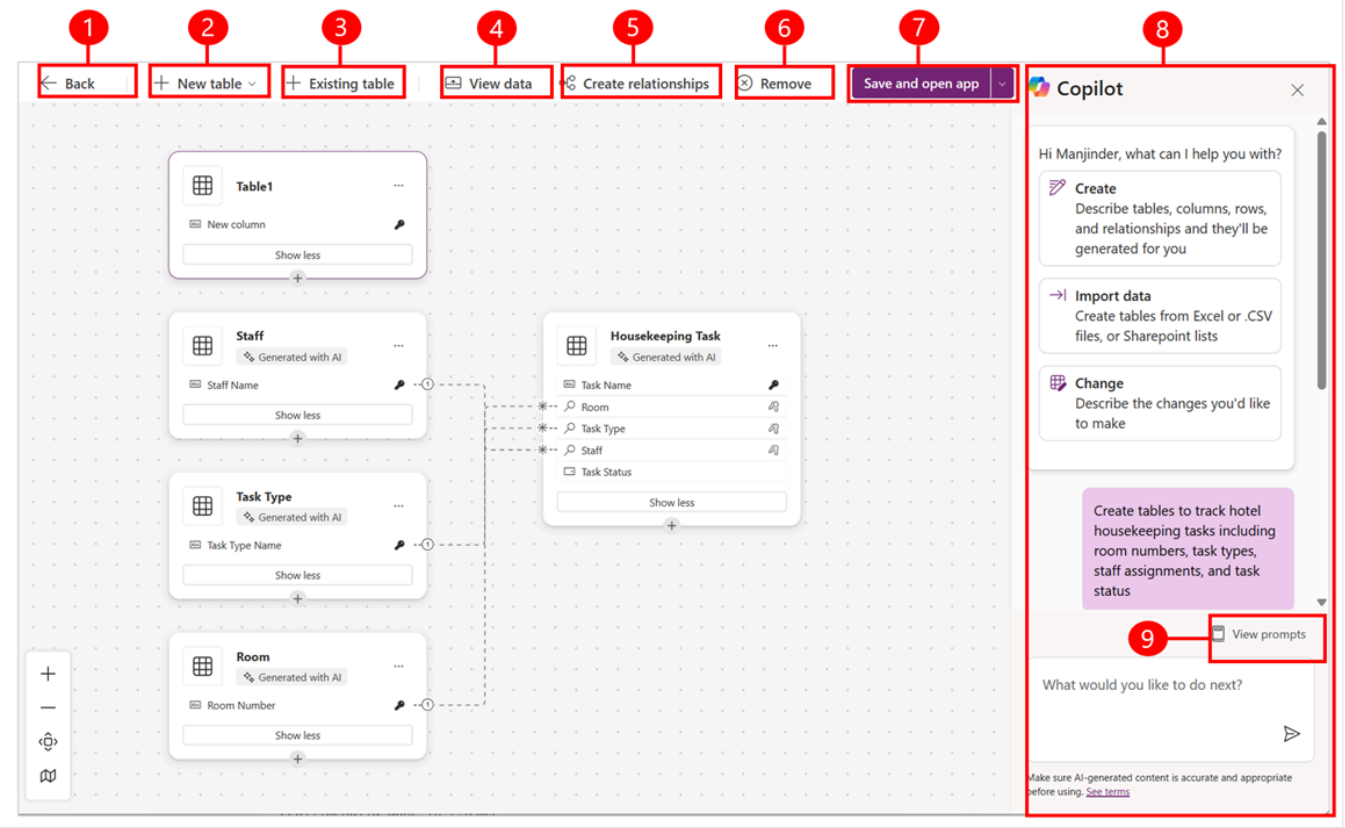
> **Important**
>
> If you don't have the right permissions or access to Dataverse in the environment you're working in, an alert asks you to create the app in your own environment. Make sure you can create the table and app in your environment to continue. If you don't have a personal developer environment, a new one is created for you automatically. For more information, see Get your developer environment (preview).

Review the table

Copilot generates tables and relationships based on your description. Review them and make any changes you need before you create your app.

Review the tables for your app

Copilot shows you the tables and relationships it generates based on your description in the **Create new tables** workspace.



The workspace includes the following options:

Expand table

Option	Description
1. **Back**	Return to the previous screen.
2. **New table**	Add a new table to your workspace.
3. **Existing table**	Add an existing Dataverse table.
4. **View data**	View the data in your tables.
5. **Create relationships**	Create relationships between tables.
6. **Remove**	Remove a table from the workspace.
7. **Save and open app**	Creates draft tables and relationships, then opens your app for editing.
8. **Copilot panel**	Ask Copilot to create tables, import data, or make changes using natural language.
9. **View prompts**	See examples of what you can ask Copilot to do.

Use Copilot to make changes

To change something, enter a brief description of the change in the Copilot panel. Copilot makes the change for you.

The Copilot panel provides three options to help you get started:

- **Create**: Describe tables, columns, rows, and relationships and they'll be generated for you.
- **Import data**: Create tables from Excel or .CSV files, or SharePoint lists.
- **Change**: Describe the changes you'd like to make.

For example, ask Copilot to add columns to track cleaning start and end time.

1. In the Copilot text box, enter **Add columns to track start and end time**.

   Copilot adds two new columns called **Start Time** and **End Time**.

2. Continue editing the table as needed. For example, add room status, change room types, or set a priority level for each room.

3. When you're ready to create your app, select **Save and exit**.

Related information

- Copilot in Power Apps overview (preview)
- Add a Copilot control to a canvas app (preview)
- Build apps through conversation (video)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 04/10/2026

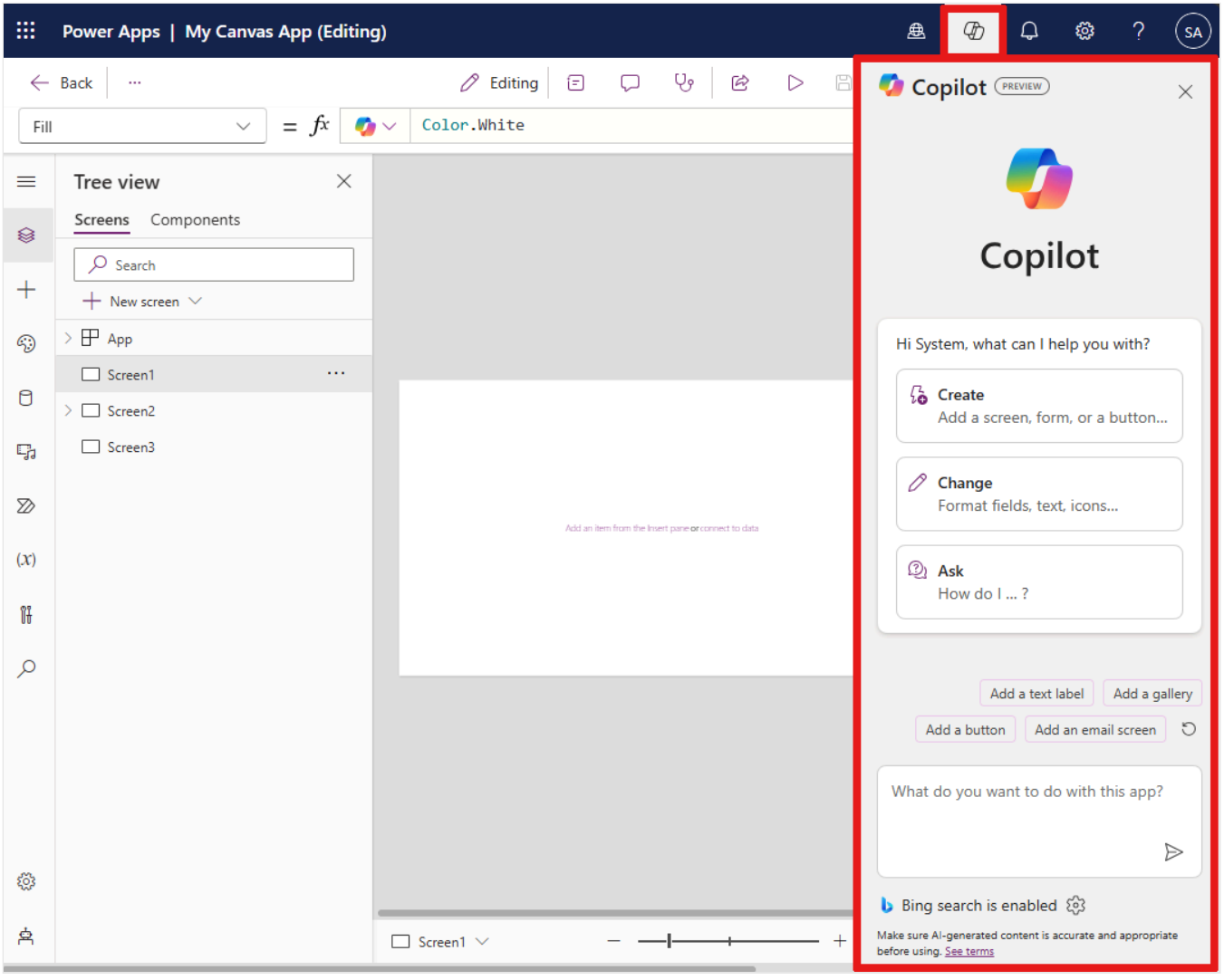
Edit your app with Copilot in Power Apps Studio (preview)

[This article is prerelease documentation and is subject to change.]

> **Important**
>
> Starting January 6, 2026, this feature will be removed. To continue creating apps with Copilot, use Plans.

Edit your apps without writing code or designing screens. With Copilot in Power Apps, you can use natural language to describe the changes you want to make, and AI does the rest.

Copilot is a feature in Power Apps that helps you edit your apps with AI assistance. You can access Copilot from Power Apps Studio when you edit a canvas app. Tell Copilot what kind of changes you want to make in the Copilot conversation pane, such as adding a screen, configuring navigation, styling a control, or editing multiple controls at a time.



> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

> **Note**
>
> This capability:
>
> - Is powered by Azure OpenAI Service
> - Might be subject to usage limits or capacity throttling

Prerequisites

Ensure you meet the prerequisites and region availability in Copilot in Power Apps overview (preview).

Supported languages

The following table lists the languages that Copilot understands.

You can track future language availability in the Copilot product report.

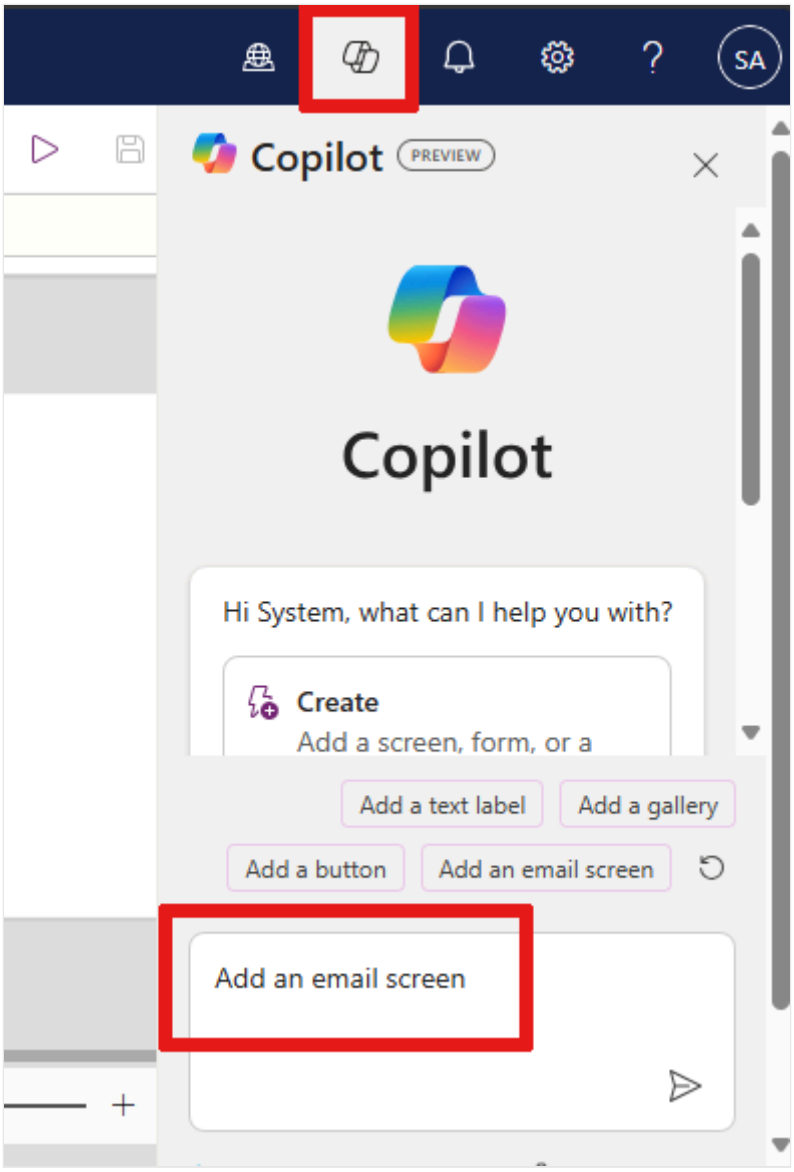
Expand table

Name	Language Code
English	en-US
Chinese (simplified)	zh-Hans
Czech	cs-CZ
Danish	da-DK
Dutch	nl-NL
Finnish	fi-FI

Name	Language Code
French	fr-FR
German	de-DE
Greek	el-GR
Italian	it-IT
Japanese	ja-JP
Korean	ko-KR
Norwegian (Bokmål)	nb-NO
Polish	pl-PL
Portuguese	pt-BR
Russian	ru-RU
Spanish (Traditional Sort)	es-ES
Swedish	sv-SE
Thai	th-TH
Turkish	tr-TR

Use Copilot to edit your app

1. Sign in to Power Apps and open a canvas app for editing.
2. In Power Apps Studio, in the upper-right corner of the page, select **Copilot**.
3. In the Copilot text box, describe a change you want to make, such as *Add an email screen*.



What you can do with Copilot in Power Apps

Copilot in Power Apps allows you to perform the following tasks:

- Add a new screen using a template. Learn more in Add a screen.
- Modify the properties of some controls, including:
  - Screen
  - Horizontal and vertical containers
  - Gallery
  - Edit form
  - Button
  - Text label
  - Text input

Sample commands you can try

When you open the Copilot pane, you can select from three preset prompts that demonstrate Copilot's capabilities: **Create**, **Change**, and **Ask**. You can also try entering the following commands in the Copilot text box.

> ⓘ **Note**
>
> Some Copilot requests are built into preview features, which only work when **Try the new data experience** toggle is turned on from your Power Apps home page. Learn more in Build apps with Copilot in Power Apps (preview).

Scenario: Add a new screen using a template

- Add a new screen
- Add a new email screen
- Add a new screen with header, body, and footer

Scenario: Add/edit/style a control

- Add a new button
- Change the selected button to have width 100
- Add a new icon
- Add a new text label
- Add a submit button and a cancel button to the form

Scenario: Bulk editing

- Change all buttons to gray
- Change all labels in the selected container to red

Scenario: Working with containers

- Add a button to the selected container

Scenario: Templatized formulas

- When the user selects `Button1`, show `Screen2`

Scenario: Modern theming

- Change my app to deep forest green

Related information

Editing your app with Copilot in Power Apps (video)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 11/11/2025

Use field suggestions by Copilot

Copilot is an AI feature in Microsoft Power Apps that helps you select the best fields to display when you link a data source to a control in a canvas app. Instead of using the default fields that Power Apps selects, you can view up to 10 suggestions from Copilot. The field suggestions are based on the data schema and the context of the app. You can review the suggestions and make adjustments as needed, saving time and improving the quality of the app.

Prerequisites

Ensure you meet the prerequisites and region availability in Copilot in Power Apps overview (preview).

Use field suggestions

Field suggestions by Copilot work when you bind a data source to one of the following controls:

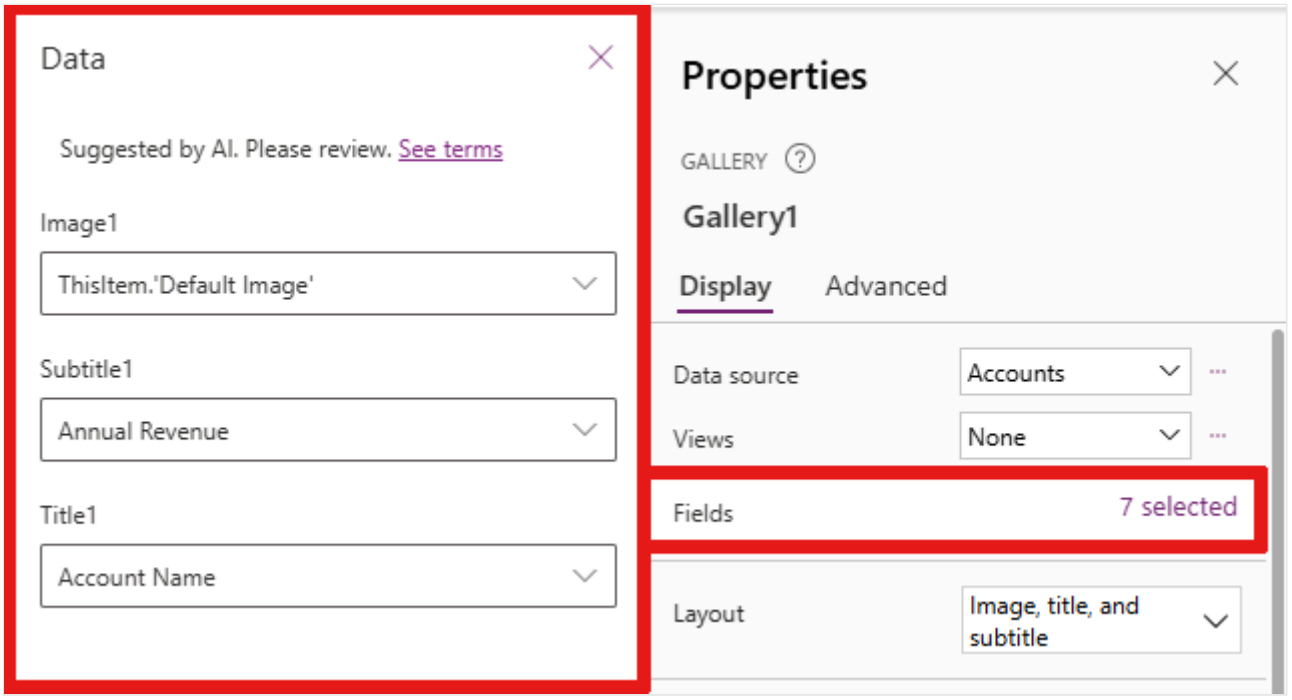
- Gallery
- Form (modern)
- Table (modern)
- Edit form (classic)
- Display form (classic)
- Data table (classic)

You can bind these controls to one of the following data sources:

- Dataverse
- SQL Server
- SharePoint list

When you select a data source for a control, Copilot analyzes the data schema and recommends up to 10 fields that are relevant and meaningful for your app. If you have more than 10 required fields, Copilot combines the AI suggestions with the required fields so that you don't get an error when you submit a form.

You can view the suggestions in the **Fields** pane and preview how they look in your app. You can also adjust the order, remove fields, and add more fields from the data source.



When you're done, apply the changes and view the results in your app.

Related information

- FAQs for field suggestions by Copilot

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 01/21/2026

Create a canvas app using Microsoft Dataverse

Summarize this article for me

Microsoft Dataverse provides a secure, cloud-based platform for storing and managing business data. With Power Apps, you can create canvas apps that connect directly to Dataverse to display, edit, and manage your organization's data without complex data connections.

This article shows you how to build canvas apps using Dataverse tables. You'll learn three different approaches: generating apps from existing data, creating new tables and apps together, and building custom apps from scratch. Whether you're working with standard tables or custom organizational data, these methods help you create functional business apps quickly.

Prerequisites

- Switch to an environment where a database is created with sample data. If you have an appropriate license, you can create an environment to meet this need.
- Ensure you have the Environment Maker security role, either directly or through a Dataverse team that is part of the **AAD Security Group** category. Custom security roles aren't currently supported for Canvas App maker scenarios.

Create an app

1. Sign in to Power Apps.
2. From the home screen, choose one of the following options based on how you want to create your app using Dataverse:

Expand table

Create options using Dataverse	Benefits	Navigation
A single-page gallery app	Use your existing business data in Dataverse and create a lightweight responsive app.	Select **Start with data** > **Dataverse**. Select a table and select **Create app**.
Create new data and build an app	Set up structured tables in Dataverse to organize data and build secure, scalable apps.	Select **Start with data** > **Create new data**. Use the table designer or Copilot to create tables, then select **Save and exit**.

Create options using Dataverse	Benefits	Navigation
Blank app that uses data from Dataverse	Use your existing business data in Dataverse to tailor your app with flexibility, free from the limitations of predesigned templates.	In the left navigation pane, select **Create** > **Start with blank canvas** > **Phone size**.

> **Note**
>
> The Power Apps home screen now surfaces a **vibe.powerapps.com** prompt, introducing a new experience that brings together generative design, AI agents, and plan-driven workflows. You can try this experience to go from idea to a working app faster.

3. The app is created and Power Apps Studio opens, where you continue building the app.

Start with data using Dataverse

The **Start with data** experience lets you quickly connect to an existing Dataverse table and generate a canvas app. You can also create new tables directly from this experience.

Connect to an existing Dataverse table

1. Sign in to Power Apps.
2. From the home screen, select **Start with data**.
3. On the **Create an app** page, select **Dataverse**.
4. Select an existing Dataverse table and select **Create app**.

Power Apps Studio opens with a gallery app connected to your selected table.

Create new data

If you don't have an existing table, you can create one directly from the **Start with data** experience.

1. From the home screen, select **Start with data**.
2. Select **Create new data**.

3. In the **Create new tables** designer, you can:

    - Add a new table by selecting **+ New table**.
    - Add an existing table by selecting **+ Existing table**.
    - Use **Copilot** to describe your tables, columns, rows, and relationships, and it generates them for you.
    - Import data from Excel or .CSV files, or SharePoint lists by selecting **Import data** in the Copilot pane.

4. When you finish defining your tables, select **Save and exit** to generate the app.

Add a Dataverse table in a blank app

To help you understand how to create an app with data from Dataverse, start by creating a blank app and then adding data from Dataverse.

1. Sign in to Power Apps.

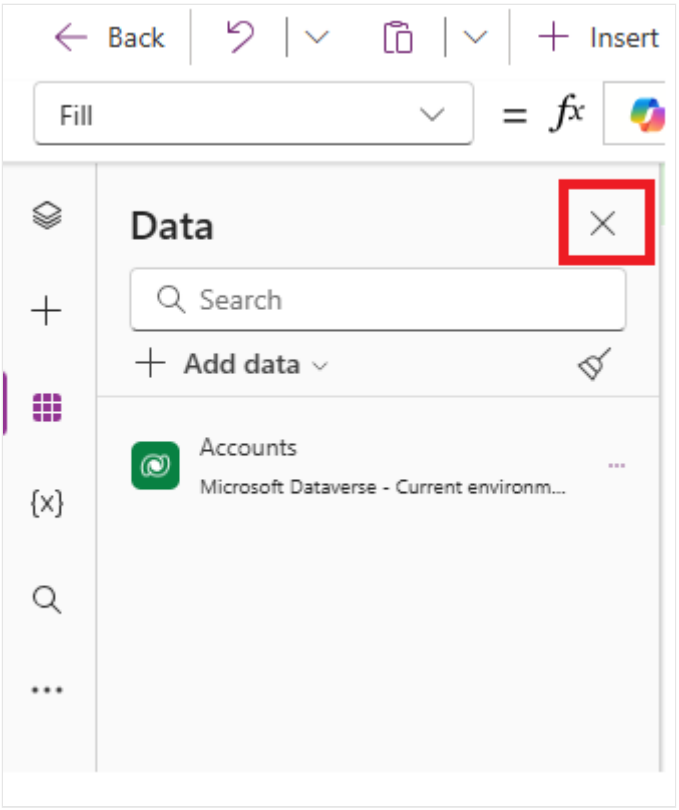
2. In the left navigation pane, select **Create** > **Start with blank canvas** > **Phone size**.

3. When Power Apps Studio opens, go to the app authoring menu and select **Data**.

    > **Note**
    >
    > If this is your first time connecting to Dataverse, you're prompted to create a connection. Select **Create** to establish the connection.

4. Select **Add data**, enter **Accounts** in the search box, and select it.

5. Close the **Data** pane by selecting the close icon in the upper right corner.



Add a list screen

1. On the command bar, select **New screen** > **List**.

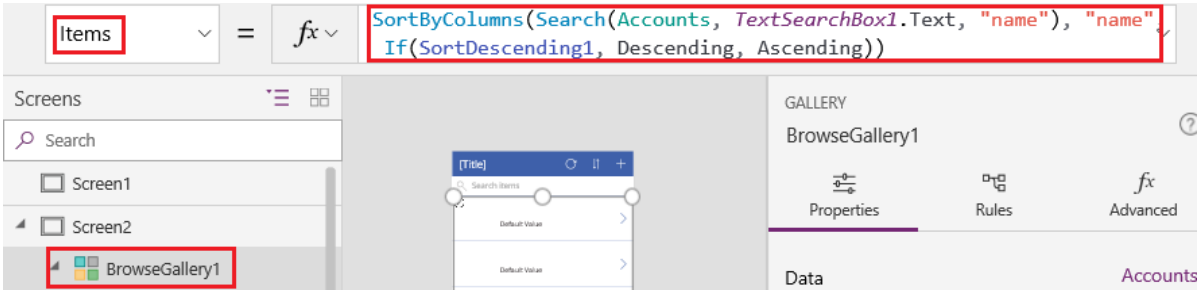
2. In the left navigation bar, select **BrowseGallery1**, and then set the value of the **Items** property to this formula:

   ```
   SortByColumns(Search(Accounts, TextSearchBox1.Text, "name"), "name",
   If(SortDescending1, SortOrder.Descending, SortOrder.Ascending))
   ```

   This formula specifies that:

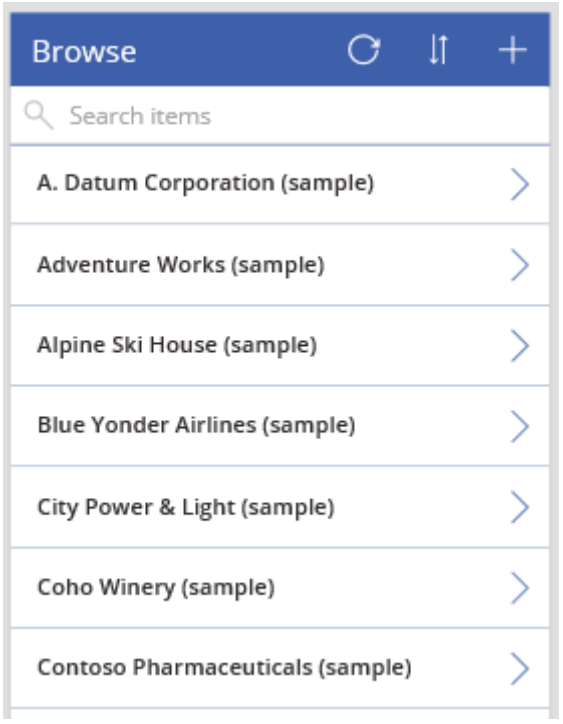
   - The gallery shows data from the **Accounts** table.
   - The data is sorted in ascending order until a user selects the sort button to toggle the sort order.
   - If a user types or pastes one or more characters into the search bar (**TextSearchBox1**), the list shows only those accounts for which the **name** field contains the characters that the user specified.

     You can use these and many other functions to specify how your app appears and behaves.



3. In the gallery's **Properties** pane, set the **Layout** to **Title**.

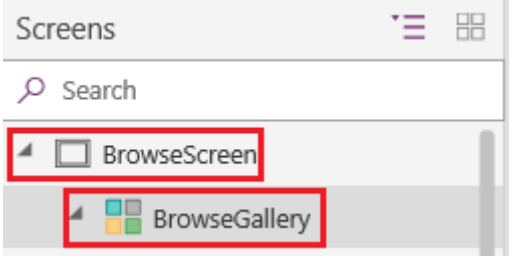
4. Edit the **Title** text property and rename it to **Browse**. For more information, see Customize a gallery.



5. In the left app authoring pane, hover over **Screen1**, select the ellipsis icon (...), and then select **Delete**.

6. In the left app authoring pane, hover over **Screen2**, select the ellipsis icon (...), and then select **Rename**.

7. Type or paste **BrowseScreen**, and then rename the gallery in that screen as **BrowseGallery**.



Add a form screen

1. Repeat the first step of the previous procedure, but add a **Form** screen instead of a **List** screen.

2. Set the form's **DataSource** property to **Accounts** and its **Item** property to **BrowseGallery.Selected** as the **Advanced** tab of the right-hand pane shows.

3. On the **Properties** tab of the right-hand pane, select **Edit Fields** to open the **Fields** pane.

4. Select **Add field**, and then select the checkboxes for these fields:

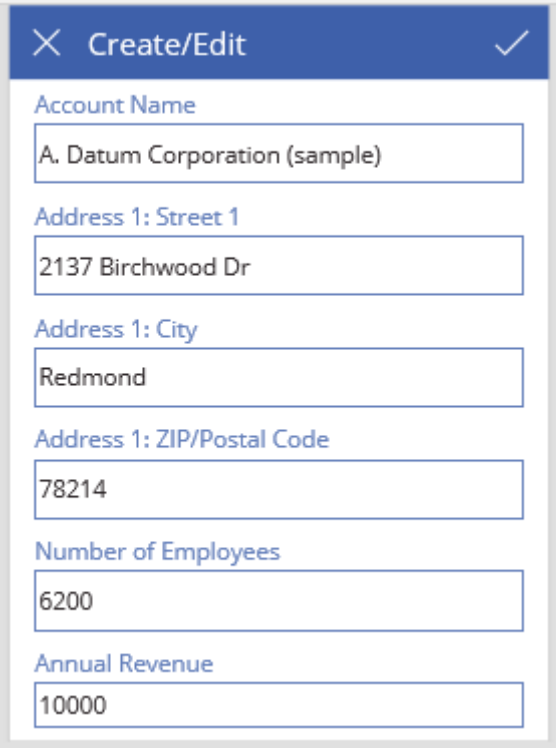
   - **Account Name**
   - **Address 1: Street 1**
   - **Address 1: City**
   - **Address 1: ZIP/Postal code**
   - **Number of Employees**
   - **Annual Revenue**

   > **Note**
   >
   > Outside of this scenario, you can create a custom field by selecting **New field**, providing the required information, and then selecting **Done**. For more information, see Create a column.
   >
   > 
   >

5. Select **Add**.

6. Set the title bar's **Text** property to show **Create/Edit**.

   The screen reflects your changes.



7. Rename this screen **FormScreen**.

Configure icons

1. On the **BrowseScreen**, set the **OnSelect** property of the circular icon near the top of the screen to this formula:

   ```
   Refresh(Accounts)
   ```

2. Set the **OnSelect** property of the plus icon to this formula:

   ```
   NewForm(EditForm1); Navigate(FormScreen, ScreenTransition.None)
   ```

3. Set the **OnSelect** property of the first arrow pointing to the right to this formula:

   ```
   EditForm(EditForm1); Navigate(FormScreen, ScreenTransition.None)
   ```

4. On the **FormScreen**, set the **OnSelect** property of the cancel icon to this formula:

```
ResetForm(EditForm1);Navigate(BrowseScreen, ScreenTransition.None)
```



5. Set the **OnSelect** property of the checkmark icon to this formula:

```
SubmitForm(EditForm1); Navigate(BrowseScreen, ScreenTransition.None)
```



6. On the **Insert** tab, select **Icons**, and then select the **Trash** icon.

7. Set the **Trash** icon's **Color** property to **White** and its **OnSelect** property to this formula:

```
Remove(Accounts, BrowseGallery.Selected); Navigate(BrowseScreen,
ScreenTransition.None)
```



Test the app

1. On the actions menu, select **Preview the app**. For more information, see Preview an app.

2. Toggle the list between ascending and descending sort orders, and filter the list by one or more characters in the account name.

3. Add an account, edit the account that you added, start to update the account, but cancel your changes, and then delete the account.

See also

- Overview of the new Power Apps vibe experience (preview)
- Customize a gallery
- Create a column

Last updated on 02/24/2026

Integrate SharePoint Online into Power Apps overview

This article guides you through creating a list from SharePoint or Microsoft Lists, integrating that list into Power Apps, and customizing the list in your app.

When you add data to an app, you can choose a SharePoint list or Microsoft Lists as a source. Alternatively, you can create an app based on a SharePoint list through the **integrate** menu in Microsoft Lists. Once your list is in your app, you can filter it and customize the list.

> **Note**
>
> When you create or view a list in SharePoint, you're automatically redirected to Microsoft Lists. You can always find the list in both Microsoft Lists and SharePoint. Learn more in What is a list in Microsoft 365?

Prerequisites

- A subscription to Microsoft 365 and Microsoft Power Platform.
- A basic familiarity with SharePoint Online and Power Apps:
  - Get started with SharePoint
  - What is Power Apps?

Create a list

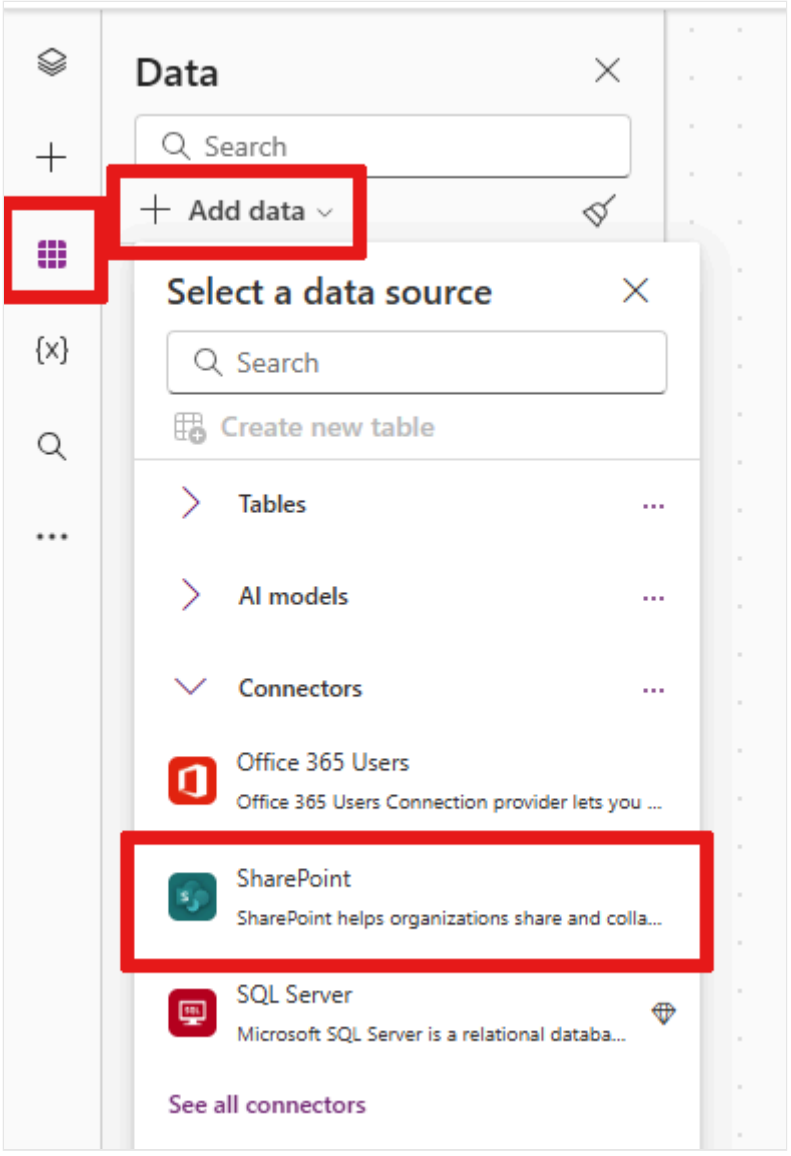
You can create a new SharePoint list through the Microsoft Lists app in Create a list.

Integrate a list

Once you have a list, you can integrate it into Power Apps or create an app based on a SharePoint list or Microsoft Lists.

Connect with a list in Power Apps

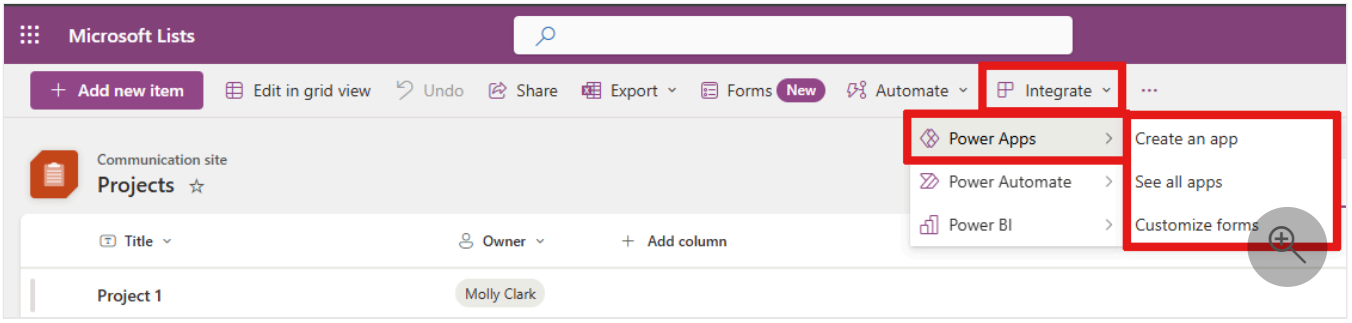
One way to work with a list in your app is to connect to SharePoint through the **Data** menu of an app to import a list. The list serves as a data source in Power Apps.



Learn more in Connect to SharePoint from a canvas app.

Create an app based on a list

When you view a SharePoint list, you automatically go to Microsoft Lists where you can integrate a list with Power Apps:



Learn more in Create a canvas app with data from Microsoft Lists.

Customize a list or form

When you import a list into Power Apps as a data source, you can use it to display a form on a screen. You can customize the appearance of your list in your app, including the formatting and behavior for the list.

Customization includes how the list displays the:

- Addition or removal of fields
- Control type and data type for the fields
- Layout orientation and number of columns
- Formatting like alignment, color, and borders
- Interactive features such as **OnFailure** or **OnSuccess**

Learn how to customize a list in Customize a form for a SharePoint list.

Learn more about form properties in Understand SharePoint forms integration.

Filter a list in Power Apps

You can apply formulas to interface elements of your app, such as filters. For a list, add a filter formula to a vertical gallery in your app screen by adding the formula to the formula bar of an *Items* property.

Learn more in Filter, Search, and LookUp functions.

Example of a search and filter function

Put this example formula on a gallery for the *Items* property. The formula searches for the status in a column and filters for the *Active* status. The list then only shows the list items with an *Active* status.

powerfx

```
SortByColumns(Filter('Issue Tracking', 'Issue Status'.Value = "Active",
StartsWith(Title, TextSearchBox1.Text)), "Title", If(SortDescending1, Descending,
Ascending))
```

The formula contains the following functions:

- SortByColumns sorts a table based on one or more columns.
- Filter finds the records in a table according to a formula that you specify.
- StartsWith tests whether one text string begins with another text string.
- If returns a value depending on whether a condition is true or false.

Learn more in Formula reference - canvas apps.

Linking lists with a lookup column

You can link two lists where one list uses one of its columns as a *lookup* column. A lookup column is the point where the list connects to another list.

Learn more in How to link lists from Microsoft Lists using a lookup column in Power Apps.

If you want your lookup column to be a dropdown list with options, use the Choices function. The choices function returns a table of the possible values for a lookup column.

For example, you can add a choices function to a dropdown control in your canvas app by adding this formula to your *Items* property of the control. You include your list name and the name of the lookup column in the formula.

PowerApps Formula

```
Choices([@'Your list name'].Lookup_x0020_column)
```

Related information

- Using the SharePoint connector with canvas apps
- Customize permissions for a SharePoint list or library
- Move SharePoint Custom Forms with Power Apps (white paper)
- Create a report quickly from a SharePoint list or library
- Embed a report web part in SharePoint Online

Last updated on 01/13/2026

Create a canvas app with data from a list

Summarize this article for me

Transform your Microsoft Lists or SharePoint data into a powerful mobile and web app in minutes—no coding required. This article shows you how to create a fully functional canvas app that connects directly to your existing lists, giving you and your team an intuitive way to view, edit, and manage data on any device.

You can also connect to on-premises SharePoint lists through a data gateway.

You can create an app by using a list in SharePoint or Lists in two ways:

- Sign in to Power Apps and then create an app by connecting to a list in SharePoint from Power Apps.
- Create an app directly from a list in SharePoint or Lists.

When you create the app, Power Apps Studio opens and shows an app with three screens:

- **Browse screen**: Scroll through all items in the list.
- **Details screen**: Show all information about a single item in the list.
- **Edit screen**: Create an item or update information about an existing item.

Watch this video to learn how to create a canvas app from a list.

https://learn-video.azurefd.net/vod/player?id=de3b0fbc-f5b6-4958-855e-109f9c3692ac&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fapp-from-sharepoint

> **Note**
>
> When you create or view a list in SharePoint, you're automatically redirected to Microsoft Lists. The list is always available in both Microsoft Lists and SharePoint. Learn more in What is a list in Microsoft 365?

Prerequisites

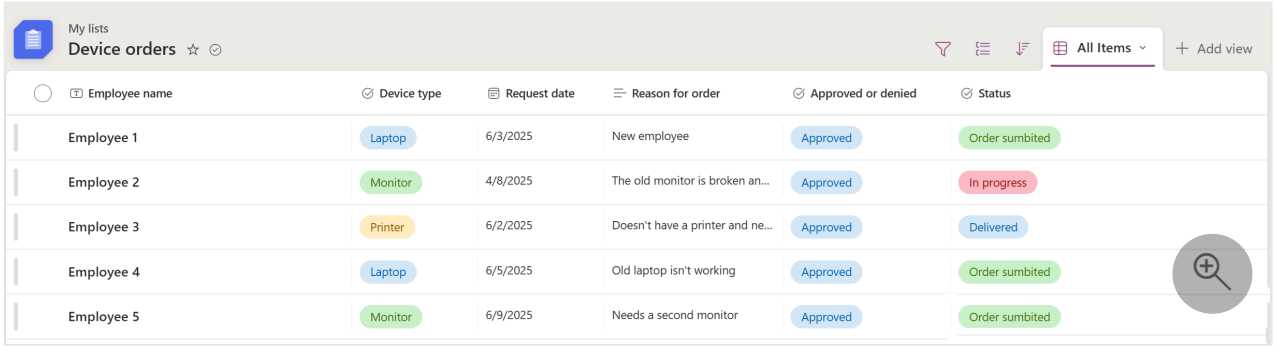
Use SharePoint and Power Apps with a subscription to Microsoft 365 and Microsoft Power Platform.

Create a list

Before you create an app from a list, create a list.

For this example, create a list to track device orders for your company. Include the following columns with sample data:

1. Employee name
2. Device type
3. Request date
4. Reason for the order
5. Approved or denied
6. Status



Use a list in SharePoint or Lists to create an app

Now, use the list that you created to build an app from the list.

1. Sign in to Power Apps.
2. From the home screen, select **Start with data**.
3. On the **Create an app** page, select **SharePoint**.
4. Enter the SharePoint URL, and then select **Connect**. Or, select a recent site.
5. Select a list or select the sample list you created earlier, and then select **Create app**.

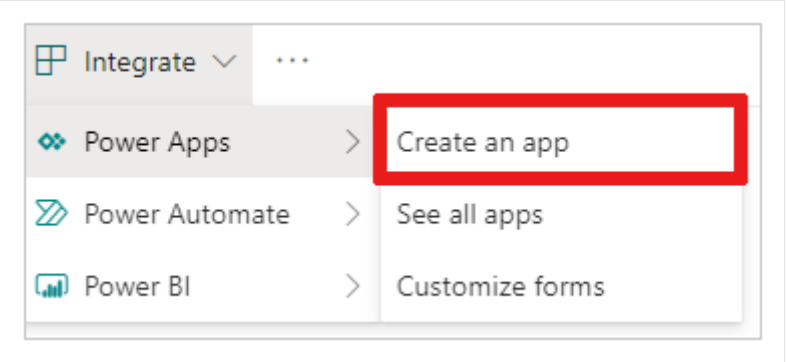
   The app opens in Power Apps Studio, where you design, build, and manage the app. For more information, see Understand Power Apps Studio.
6. Select the **Preview the app** icon to see how the app works.
7. Save and publish your app.

When you add or edit information in the app, the information in SharePoint or Lists also updates.

Create an app directly from a list in SharePoint or Lists

Before you create the app, make sure you create a list.

1. Sign in to SharePoint: `https://yourorganizationname.sharepoint.com`
2. Open a list in either SharePoint or Lists. If you open the list from SharePoint, you're taken to Lists to view the list.
3. Select **Integrate** > **Power Apps** > **Create an app**.



4. Select the **Preview the app** icon to see how the app works.
5. Save and publish your app.

   When you add or edit information in the app, the information in SharePoint or Lists also updates.

Manage your app

After you create an app, you can play, save, share, and publish it.

Learn more about these app actions in the Power Apps Studio interface.

Related information

Move SharePoint Custom Forms with Power Apps (white paper)

Last updated on 02/24/2026

Link lists using a lookup column in Power Apps

Article • 03/07/2025

This tutorial shows you how to connect two lists with a lookup column in a canvas app in Power Apps.

SharePoint provides two types of lookup columns:

- **Lookup**: Links to another list. For example, an **Orders** list might have a lookup column that links to customers in a **Customer** list.
- **Choice**: Selecting a column displays a menu of items you can choose.

> **Note**
>
> When you create or view a list in SharePoint, you're automatically redirected to Microsoft Lists. The list can always be found in both Microsoft Lists and SharePoint. Learn more in What is a list in Microsoft 365?

Prerequisites

- A SharePoint license or a Microsoft 365 license
- A SharePoint site

  To create a SharePoint site, see Guided walkthrough: Creating an organization site.

Why use a lookup column?

Data in an enterprise is large and complex. Data in one list often relates to data in another list. Lookup columns are the primary way such business data comes together.

For example, you might have an **Orders** list which has a lookup column that links to a **Customers** list, to show which customer placed the order. The lookup column in the **Orders** list lets you get other data from the **Customers** list as well.

You might also use a lookup column to connect the **Orders** list to a **Product** list, and bring in information you need about the product ordered, such as product pictures, specifications, or manufacturer details.

What are choice columns used for?

Choice columns are used for short lists. Instead of creating a separate list, you include the list values in a small menu. This menu appears when you select a **Choice** column, and then you can select one of the values from the menu.

Examples include data like Customer Status Code, Product Availability, State Codes—any fixed, relatively short list.

The choice data can stay as a separate list, if you use a **Lookup** column to link them. However, a **Choice** column implementation is easier and quicker than a **Lookup** column implementation.

Learn more in Integrate SharePoint Online into Power Apps overview.

Create the lists

In this tutorial, you link two lists together, **Assets** and **RepairShop**. The **Assets** list is used to track hardware equipment in a team. Since hardware breaks from time to time, we use the **RepairShop** list to track the local shops which can fix it.

The lookup column used in this example

The **RepairShop** list uses the *ContactEmail* column to identify the shop. This list is defined first so that each row in the **Assets** list points to something.

The **Assets** list has two lookup columns:

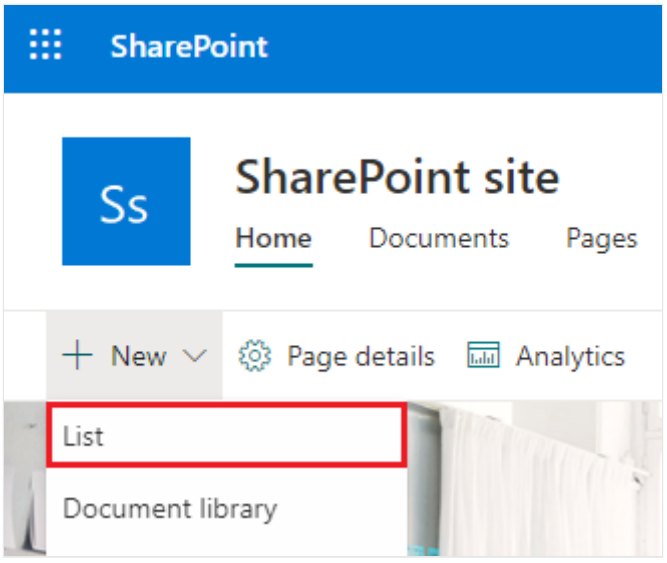
- *RepairShop*, of type **Lookup**, uses email addresses to point to entries in the **RepairShop** list.
- *AssetType*, of type **Choice**, lists the asset types for hardware.

You can define more columns, depending on the information you need to track.

Define the RepairShop list and add data

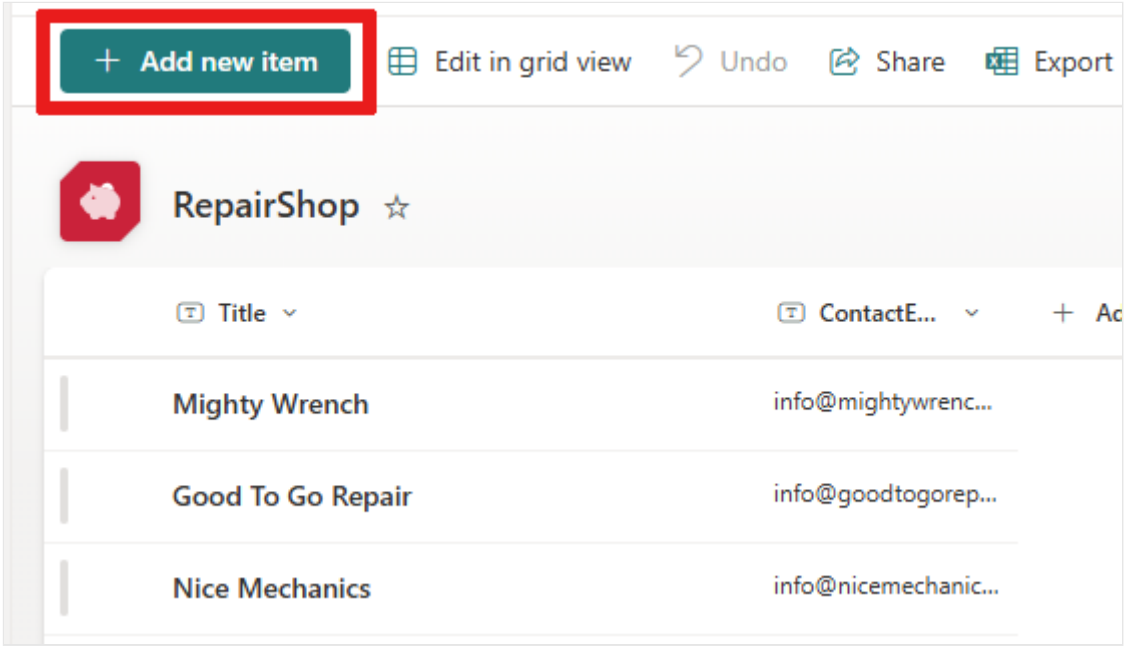
Create this list first, so when you add data to the **Assets** list, **RepairShop** entries are available from the *Assets.RepairShop* lookup column.

1. On any SharePoint site, create a new **RepairShop** list from a **Blank list** template.



2. Select **Add a column** of type **Text** and name it *ContactEmail*, then choose **Save**.

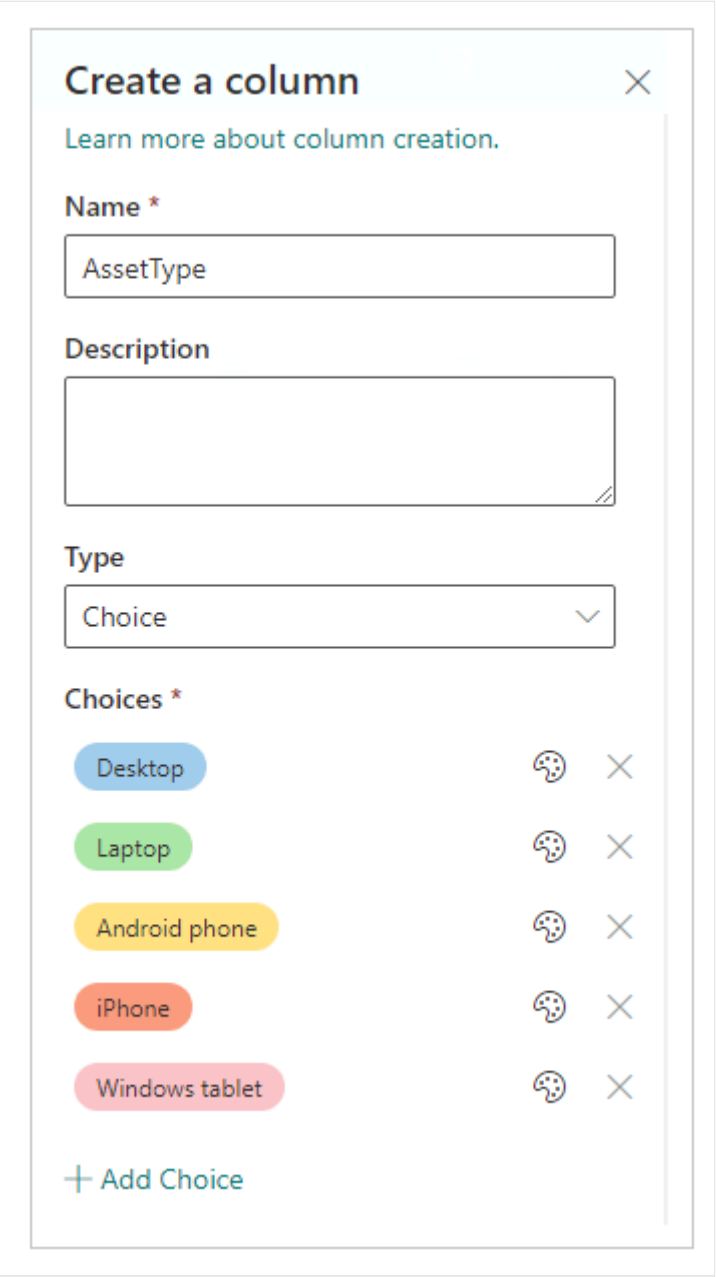
3. Select **+ Add new item** to enter at least three rows with different *ContactEmail* sample values. When an asset needs to be repaired, you choose one of these repair shops.



Define the Assets list

1. On the same SharePoint site, create a new **Assets** list from a **Blank list** template.

2. Select **+ Add column** and choose the **Choice** column type. Select **Next** and name it *AssetType*. Fill in the *Choice* values under the **Choices** section with sample assets such as *Desktop*, *Laptop*, *Android phone*, *iPhone*, and *Windows tablet*.



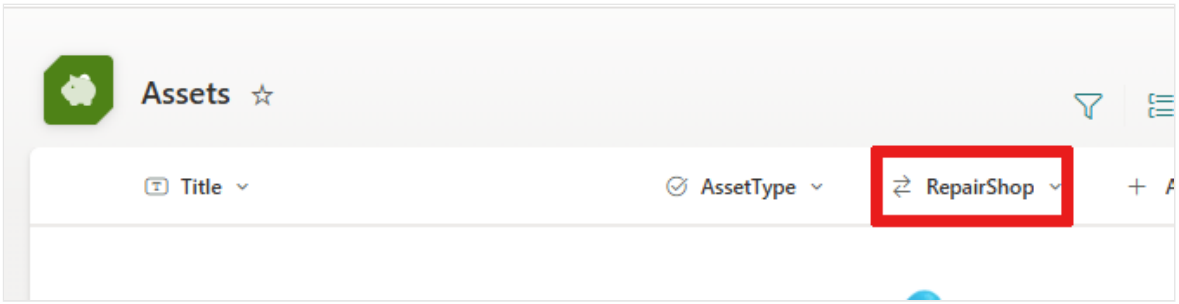
3. Select **Save**.

4. Select + **Add column** and choose the **Lookup** column type, then select **Next**.

5. In the **Create a column** pane, add RepairShop to the **Name** field. Go to **Select a list as a source** and choose **RepairShop**. Go to **Select a column from the list above** and choose **ContactEmail**.

6. Select **Save**.

   You see the column *RepairShop* has a two-arrows icon that indicates it's a lookup type column.



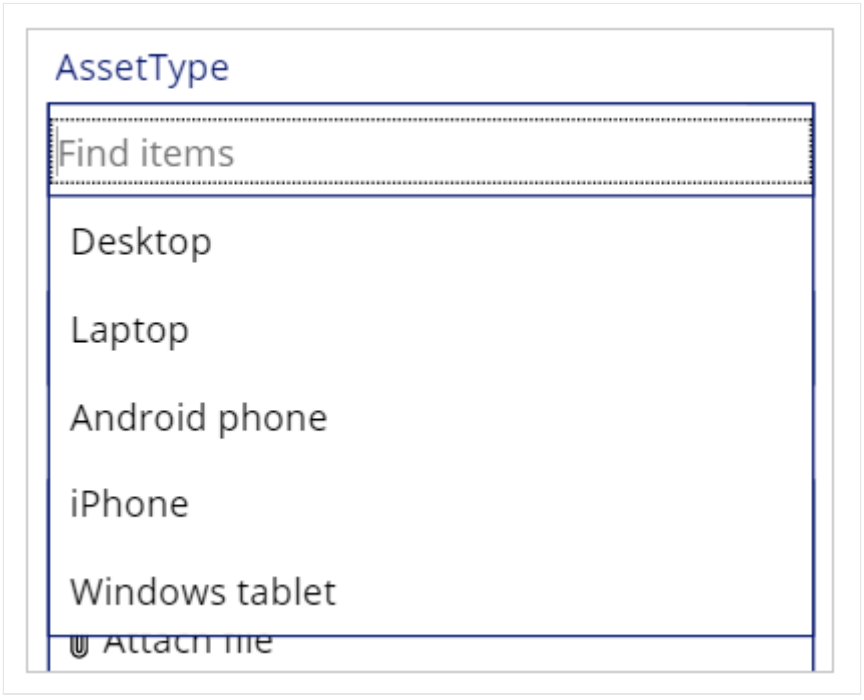
Create an app from the Assets list

Create a canvas app from the Assets list created earlier.

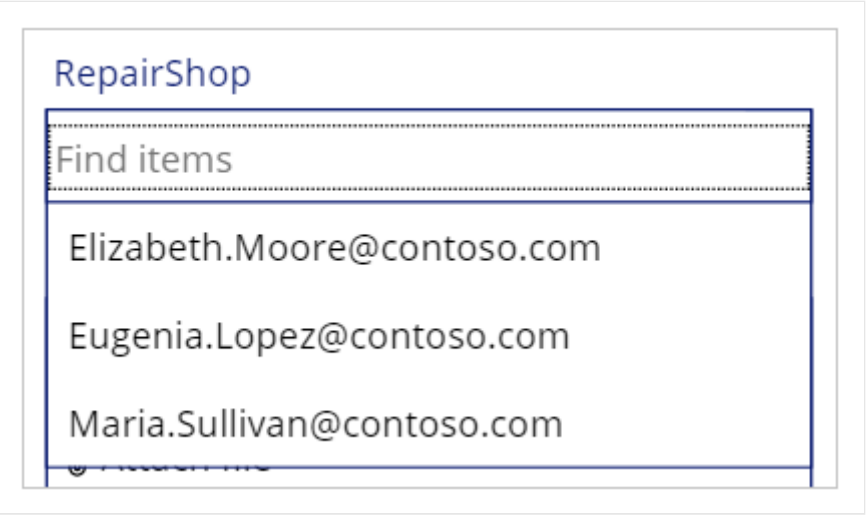
Add data to the Assets list

Let's preview the app and add items to your **Assets** list.

1. Press F5 or select Preview (▷).
2. Select the + symbol in the upper right corner of your app to add an entry.
3. Enter a **Title** for this asset, for example *Micah's Laptop*.
4. Select the **AssetType** dropdown arrow. The values displayed are values you entered when you created this column. Choose one of the entries.



5. Select the **RepairShop** dropdown arrow. Choose one of the entries.



6. Select the check mark to save the new entry.

7. (Optional) Add more items to the list.

8. Press `Esc` to return to the default workspace.

9. Save and publish the app.

Related information

- Configure Drop down control in Power Apps in your list.
- Generate an app by using a Microsoft Dataverse database
- Create an app from scratch using a Dataverse database
- Move SharePoint Custom Forms with Power Apps (white paper)

Feedback

Was this page helpful? Yes No

Provide product feedback

Understand SharePoint forms integration

You can customize a form for a SharePoint list when you integrate a list into Power Apps. In this article, you learn how these forms work and how you can customize them.

When you customize a form for a list, you see the default generated form works for all operations, like creating, showing, or editing an item. This functionality uses generated formulas and the **SharePointIntegration** control.

> **Note**
>
> When you create or view a list in SharePoint, you're automatically redirected to Microsoft Lists. You can always find the list in both Microsoft Lists and SharePoint. For more information, see What is a list in Microsoft 365?

Understand the default generated form

The default generated form consists of the following controls and their corresponding defaults. Formulas are autogenerated within the defaults.

- **FormScreen1**: This screen contains the form.
- **SharePointForm1**: This form can create, show, or edit a list item.
  - **Data Source**: The list for which the form is customized. For example, a SharePoint list acts as a data source.
  - **Item**: A selected item from a list. This item is set to `First()` item in the list for your convenience, when working in Power Apps Studio.

```
Power Fx
If( IsBlank(SharePointIntegration.Selected) ||
IsEmpty(SharePointIntegration.Selected),
    First('*YourListName*'),
SharePointIntegration.Selected )
```

> **Tip**
>
> This formula pattern uses `... SharePointDatasourceName.Selected` and works for the **Item** property of a form. See the Common issues with the

SharePointIntegration object for a formula pattern to set the value of a SharePoint record.

- **OnSuccess**: Once the item is created or saved successfully, the form resets and SharePoint hides the form.

```
Power Fx
ResetForm(SharePointForm1); RequestHide()
```

- **SharePointIntegration**: Communicates user actions between SharePoint and Power Apps.

Expand table

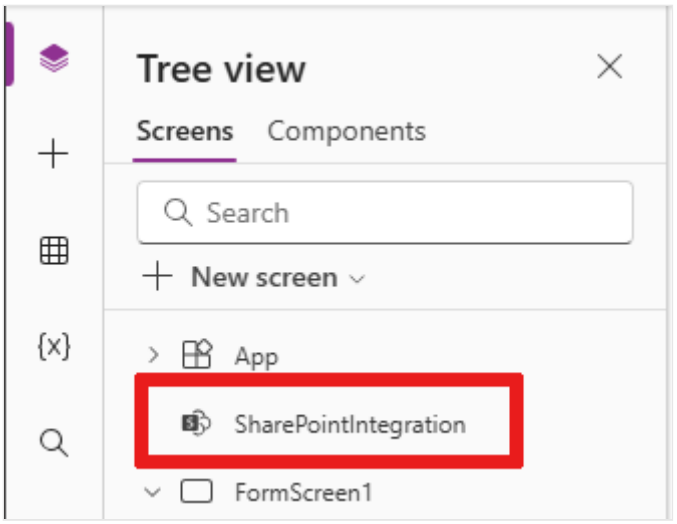
Property	Description	Example
Data Source	The list for which the form is customized.	*YourListName*
OnNew	Sets **SharePointForm1** in new mode.	*NewForm(SharePointForm1)*
OnView	Sets **SharePointForm1** in view mode.	*ViewForm(SharePointForm1)*
OnEdit	Sets **SharePointForm1** in edit mode.	*EditForm(SharePointForm1)*
OnSave	Submits the changes to **SharePointForm1**. The **SharePointForm1.OnSuccess** formula executes when you successfully submit a form.	*SubmitForm(SharePointForm1)*
OnCancel	Resets the changes to **SharePointForm1**. SharePoint always hides the form when a user selects **Cancel** in SharePoint.	*ResetForm(SharePointForm1)*

These defaults ensure that the form works when running within SharePoint. The defaults change the Power Apps form mode as the user interacts with it in SharePoint, and they ensure that the changes are submitted to SharePoint.

> **Note**
>
> While you can manually copy a form from one environment to another, there's currently no automated method in Power Apps to copy a form from one environment to another.

Understand the SharePointIntegration control

The **SharePointIntegration** control communicates user actions between SharePoint and Power Apps.



> **Note**
>
> You can access properties for the **SharePointIntegration** control only when the form runs in SharePoint, not when you're customizing the form in Power Apps Studio. These properties might not be available in **OnStart** or **OnVisible**.

The **SharePointIntegration** control has these properties:

Expand table

Property	Effect or description
Selected	The selected item from the list.
OnNew	Activates when a user selects the **New** button or opens the **Create item** form in SharePoint.
OnView	Activates when a user selects an item or opens the **Item detail** form in SharePoint.
OnEdit	Activates when a user selects the **Edit all** button or opens the **Edit item** form in SharePoint.
OnSave	Activates when a user selects the **Save** button in SharePoint.
OnCancel	Activates when a user selects the **Cancel** button in SharePoint.
SelectedListItemID	Item ID for the selected item in a list.
Data Source	The list that contains the record that the form shows, edits, or creates. If you change this property, the **Selected** and **SelectedItemID** properties might stop working.

Customize the default form

You can change the formulas to further customize the forms.

Consider these tactics when you customize forms:

- Use the **OnSave** formula of the `SharePointIntegration` control to customize the effect of a user selecting **Save** in SharePoint. If you have multiple forms, make sure to submit the changes only for the form currently being used.

  > **Tip**
  >
  > Set different values for a variable in the **OnNew**, **OnView**, and **OnEdit** formulas. You can use this variable in the **OnSave** formula to determine which form is being used.

- Include **RequestHide()** in the **OnSuccess** formula of all your forms. Without this function, SharePoint doesn't know when to hide the form. Avoid running important code *after* calling **RequestHide()**. Code should run while the form is still visible and able to run logic.

- You can't control the hiding of a form when a user selects **Cancel** in SharePoint. Always reset your forms in the **OnCancel** formula of the `SharePointIntegration` control.

- Properties for the `SharePointIntegration` control might not be available in **OnStart** or **OnVisible**. These events execute only once while the list is loaded. You can use **OnNew**, **OnView**, or **OnEdit** to run logic before the form is shown to the user.

Common issues with the SharePointIntegration object

- When you set the value of `SharepointIntegration.Selected` to a collection on the **OnView** property, it doesn't show the latest value. To fix this problem, use `SharepointIntegration.SelectedListItemID`, and then do a lookup on the table to get the selectedRecord.

  For example, for **OnView** property:

  - Instead of:

    Power Fx

    ```
    Set( selectedItem,
    SharePointIntegration.Selected );
    ```

- Use:

```
Power Fx
Set( selectedLookupItem,
LookUp( YourSharepointIntegrationObject,
ID=SharePointIntegration.SelectedListItemID ) );
```

- Collection variables aren't reset when you close the Power Apps form. The state persists for the entire session. If you need to reset the variables, clear them in the **OnView** property of the `SharePointIntegration` object.
- Don't use imperative functions such as **Launch()** in the `SharePointIntegration` properties (such as **OnNew** and **OnView**). This use can cause unexpected behavior since the `SharePointIntegration` lifecycle events (such as selections changing) can trigger in the background even when the form isn't visible.
-

Understand user access to customized forms

You can't manually share a SharePoint form customized with Power Apps. Instead, any user who has at least **Read** or **Restricted View** access to the linked SharePoint list inherits access to the form. Users who have only been given permissions to specific items in the list don't have access to the customized form.

Related information

- Move SharePoint Custom Forms with Power Apps (white paper)
- EditForm, NewForm, SubmitForm, ResetForm, and ViewForm functions
- RequestHide function

Last updated on 11/14/2025

Integrate SharePoint Online into Power Apps overview

This article guides you through creating a list from SharePoint or Microsoft Lists, integrating that list into Power Apps, and customizing the list in your app.

When you add data to an app, you can choose a SharePoint list or Microsoft Lists as a source. Alternatively, you can create an app based on a SharePoint list through the **integrate** menu in Microsoft Lists. Once your list is in your app, you can filter it and customize the list.

> **Note**
>
> When you create or view a list in SharePoint, you're automatically redirected to Microsoft Lists. You can always find the list in both Microsoft Lists and SharePoint. Learn more in What is a list in Microsoft 365?

Prerequisites

- A subscription to Microsoft 365 and Microsoft Power Platform.
- A basic familiarity with SharePoint Online and Power Apps:
  - Get started with SharePoint
  - What is Power Apps?

Create a list

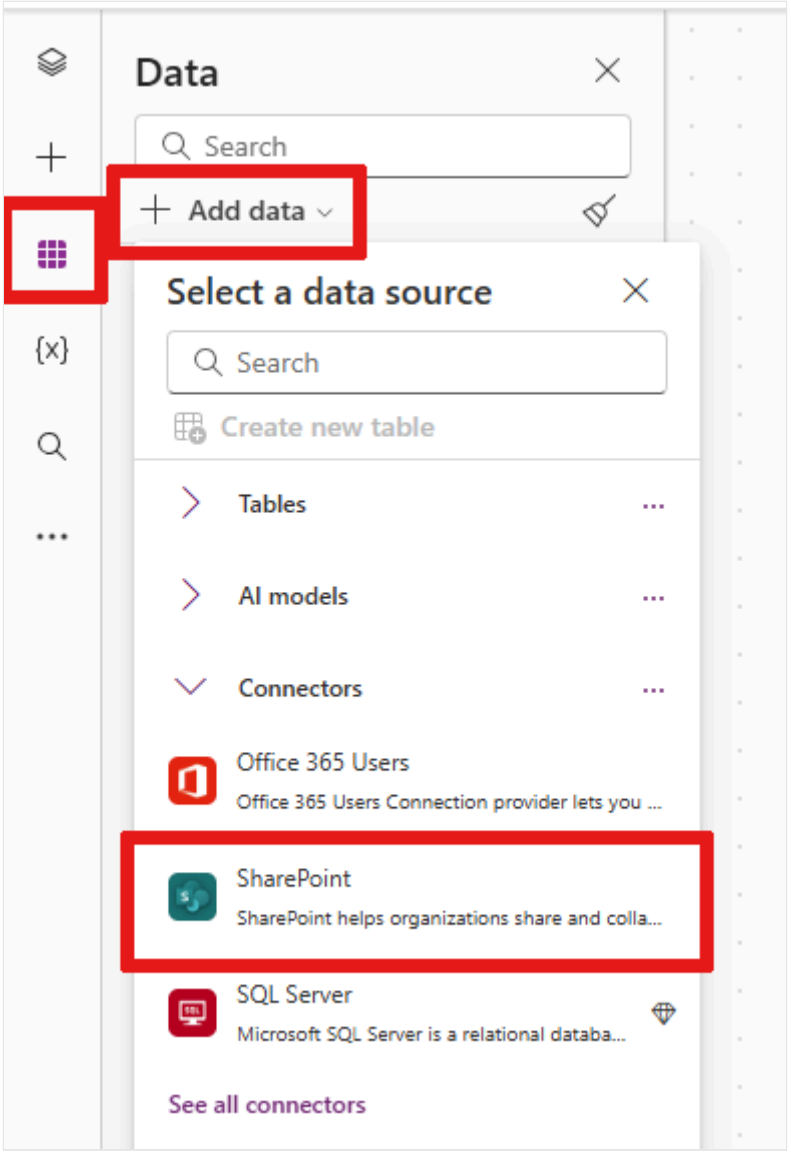
You can create a new SharePoint list through the Microsoft Lists app in Create a list.

Integrate a list

Once you have a list, you can integrate it into Power Apps or create an app based on a SharePoint list or Microsoft Lists.

Connect with a list in Power Apps

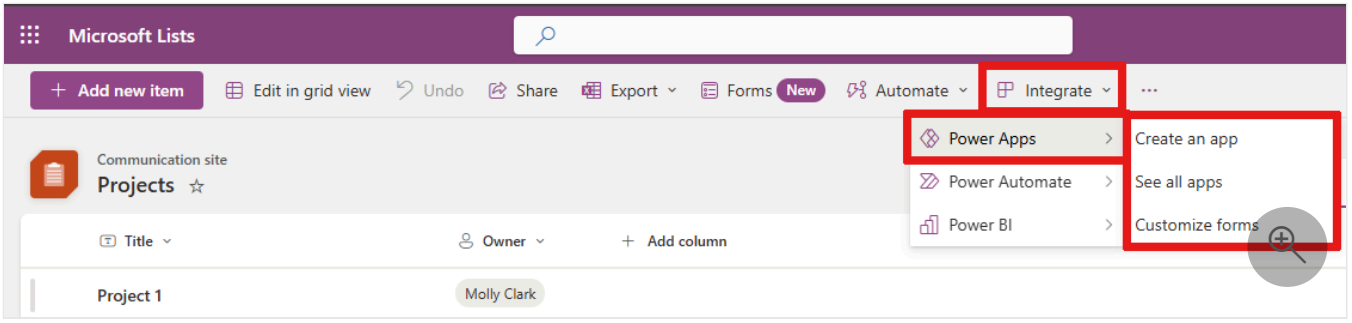
One way to work with a list in your app is to connect to SharePoint through the **Data** menu of an app to import a list. The list serves as a data source in Power Apps.



Learn more in Connect to SharePoint from a canvas app.

Create an app based on a list

When you view a SharePoint list, you automatically go to Microsoft Lists where you can integrate a list with Power Apps:



Learn more in Create a canvas app with data from Microsoft Lists.

Customize a list or form

When you import a list into Power Apps as a data source, you can use it to display a form on a screen. You can customize the appearance of your list in your app, including the formatting and behavior for the list.

Customization includes how the list displays the:

- Addition or removal of fields
- Control type and data type for the fields
- Layout orientation and number of columns
- Formatting like alignment, color, and borders
- Interactive features such as **OnFailure** or **OnSuccess**

Learn how to customize a list in Customize a form for a SharePoint list.

Learn more about form properties in Understand SharePoint forms integration.

Filter a list in Power Apps

You can apply formulas to interface elements of your app, such as filters. For a list, add a filter formula to a vertical gallery in your app screen by adding the formula to the formula bar of an *Items* property.

Learn more in Filter, Search, and LookUp functions.

Example of a search and filter function

Put this example formula on a gallery for the *Items* property. The formula searches for the status in a column and filters for the *Active* status. The list then only shows the list items with an *Active* status.

powerfx

```
SortByColumns(Filter('Issue Tracking', 'Issue Status'.Value = "Active",
StartsWith(Title, TextSearchBox1.Text)), "Title", If(SortDescending1, Descending,
Ascending))
```

The formula contains the following functions:

- SortByColumns sorts a table based on one or more columns.
- Filter finds the records in a table according to a formula that you specify.
- StartsWith tests whether one text string begins with another text string.
- If returns a value depending on whether a condition is true or false.

Learn more in Formula reference - canvas apps.

Linking lists with a lookup column

You can link two lists where one list uses one of its columns as a *lookup* column. A lookup column is the point where the list connects to another list.

Learn more in How to link lists from Microsoft Lists using a lookup column in Power Apps.

If you want your lookup column to be a dropdown list with options, use the Choices function. The choices function returns a table of the possible values for a lookup column.

For example, you can add a choices function to a dropdown control in your canvas app by adding this formula to your *Items* property of the control. You include your list name and the name of the lookup column in the formula.

PowerApps Formula

```
Choices([@'Your list name'].Lookup_x0020_column)
```

Related information

- Using the SharePoint connector with canvas apps
- Customize permissions for a SharePoint list or library
- Move SharePoint Custom Forms with Power Apps (white paper)
- Create a report quickly from a SharePoint list or library
- Embed a report web part in SharePoint Online

Last updated on 01/13/2026

Create a canvas app based on Excel data

Summarize this article for me

This article shows you how to create a canvas app using Excel data as your data source. Whether you have existing Excel spreadsheets with business data or want to leverage Excel's familiar structure, Power Apps provides flexible options to transform your data into powerful applications.

You'll learn three different approaches to work with Excel data:

Upload Excel files to create Dataverse tables for enhanced data management and security Connect to external Excel files stored in cloud services for quick app creation Start with a blank canvas app and add Excel data for complete customization control Before you begin, ensure your Excel data is properly formatted. This preparation step helps Power Apps read and interact with your data more effectively. For more information, see Formatted as a table in Excel.

To create an app by using Excel data, select one of the following options in Power Apps.

Expand table

Create options by using Excel	Benefits	Navigation
Upload an Excel or .CSV file to Power Apps	Uploaded Excel data generates a Dataverse table. With Dataverse's standard and custom tables, you can securely store your data in the cloud. These tables enable you to define your organization's data in a way that is tailored to your business needs, making it easier to use within your apps.	Select **Start with Data** > **Upload file**.
Connect to an external Excel file and use it to create a canvas app. Your data stays where it is.	Using the Excel connector in Power Apps provides a straightforward and fast method to create and deploy apps with data stored in Excel. It's especially advantageous for makers that already use Excel and want to enhance its functionality with minimal effort. Makers benefit from cloud access, easy data manipulation, rapid prototyping, and no need for a complex database setup.	Select **Start with Data** > **Excel Online (Business)**.
Create a blank canvas app and then add Excel data	You have complete flexibility, control, and customization. It allows you to design an app that meets your exact specifications without being limited by predesigned templates, providing greater creative freedom and adaptability to suit your unique needs.	On the left navigation pane, select **Create** > **Create from blank**. Then select the app size.

Upload an Excel or CSV file to Power Apps

When you upload your Excel data to Power Apps, it doesn't just stay as an Excel file. Instead, Power Apps converts it into a Dataverse table. This conversion makes it easier to integrate and manage your data within Power Apps, and you get more powerful features compared to using Excel alone. For more information, see Why use Dataverse?

1. Sign in to Power Apps.
2. On the home screen, select **Start with data**.
3. On the **Create an app** page, select **Upload file**.
4. Select **Select from device** and go to the location where your Excel file is saved. Select the file to upload it.

   > **Note**
   >
   > The maximum file size limit is 5 GB.

5. When Power Apps creates the table, select a column name or the table name to edit the properties to suit your needs. If you change column data types and there are values in cells that don't match the selected data type, Power Apps removes those values when it generates the table. For more information, see Create and edit tables using Power Apps.
6. Select **Row ownership** and choose how you want to manage row ownership.
7. When you're done, select **Save and open app**. Power Apps uploads the first 20 rows of data so you can start reviewing the data in your app. The system uploads the remaining data in the background.

Known issues

- The current data upload process doesn't take into account the environment data format setting.

Connect to an external Excel file from Power Apps

Store the Excel file in a cloud storage service like Dropbox, Google Drive, OneDrive, or OneDrive for Business because Power Apps can only connect to Excel files that are stored in the cloud.

Power Apps has an Excel connector that you can use to access your Excel data. The Excel Online (Business) connector offers flexibility by accessing a wide variety of cloud-based file storage services.

1. Sign in to Power Apps.
2. On the home screen, select **Start with data**.
3. On the **Create an app**, select **Excel Online (Business)**.
4. Only one connection appears at a time. To select a different connection, select the **...** button to switch connections or add a new connection.
5. Enter the file location and select the table.
6. When you're done, select **Create app**.

For more information about how to share Excel data, see Sharing Excel tables.

Create a blank canvas app and add Excel data

Create a blank app and then use Excel data that's formatted as a table as the source of the app's information.

Follow the example below to create an app with two screens where users can view (browse) the data from an Excel file on one screen, and on another screen, they can interact with (add, modify, or remove) that data. The data interaction features, such as creating, updating, and deleting records, are built into the second screen of the app.

Prerequisites

1. Copy this data and paste it into an Excel file.

Expand table

StartDay	StartTime	Volunteer	Backup
Saturday	10am-noon	Vasquez	Kumashiro
Saturday	noon-2pm	Ice	Singhal
Saturday	2pm-4pm	Myk	Mueller
Sunday	10am-noon	Li	Adams
Sunday	noon-2pm	Singh	Morgan

StartDay	StartTime	Volunteer	Backup
Sunday	2pm-4pm	Batye	Nguyen

2. Format the table in Excel and name it **Schedule** so that Power Apps can parse the information.

3. Save the file as **eventsignup.xlsx**, close it, and then upload it to a cloud-storage account such as OneDrive.

> **Important**
>
> You can use your own Excel file and review this example for general concepts only. However, the data in the Excel file must be formatted as a table.

Create a blank app and connect to data

1. Sign in to Power Apps.

2. On the left navigation pane, select **Create** > **Start from blank**.

3. Select the **Phone size** layout.

   The app opens in Power Apps Studio where you can add data and start building the app.

4. In the middle of the screen, select **Connect to data**.

5. In the **Data** pane, select **Add data**. Select the connection for your cloud-storage account if it appears. Otherwise, follow these steps to add a connection such as OneDrive:
   a. In the search box, enter **OneDrive** and select it.
   b. Select **Add a connection**.
   c. On the connection pane, select **Connect**.
   d. If prompted, provide your credentials for that account.

6. Under **Choose an Excel file**, find and then select the **eventsignup.xlsx** that you saved earlier.

7. Under **Choose a table**, select the check box for **Schedule**, and then select **Connect**.

8. In the upper-right corner of the **Data** pane, close it by selecting the close icon (**X**).

Create the view screen

1. On the command bar, select **New screen** > **List**.

A screen is added with several default controls, such as a search box and a **Gallery** control. The gallery covers the entire screen under the search box.

2. At the top of the new screen, select the **[Title]** label control and rename it to **View records**.

3. In the **Tree view**, select **BrowseGallery1**.

4. In the gallery's **Properties** pane, set the **Layout** to **Title, subtitle, and body**.

5. In the formula bar, replace **CustomGallerySample** with **Schedule**, and replace both instances of **SampleText** with **Volunteer**.

6. On the right edge of the formula bar, select the **Expand formula bar** down arrow, and then select **Format text**.

   The formula matches this example:

   powerfx

   ```
   SortByColumns(
       Search(
           Schedule,
           TextSearchBox1.Text,
           "Volunteer"
       ),
       "Volunteer",
       If(
           SortDescending1,
           SortOrder.Descending,
           SortOrder.Ascending
       )
   )
   ```

7. In the **Properties** pane, select **Edit** next to the **Fields** label.

8. In the **Title2** box, select **Volunteer**, in the **Subtitle2** box, select **StartDay**, and in the **Body1** box, select **StartTime**.

9. In the upper-right corner of the **Data** pane, close it by selecting the close icon (X).

   Users can sort and filter the gallery by volunteer name based on the **SortByColumns** and **Search** functions in that formula.

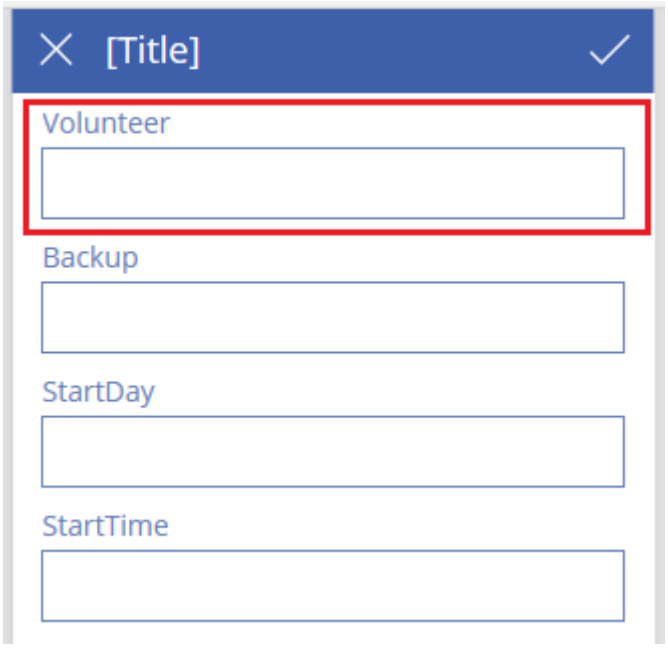
   - If a user types at least one letter in the search box, the gallery shows only those records for which the **Volunteer** field contains the text that the user typed.

- If a user selects the sort button (between the refresh button and the plus button in the title bar), the gallery shows the records in ascending or descending order (depending on how many times the user selects the button) based on the **Volunteer** field.

For more information about these and other functions, see the formula reference.

Create the change screen

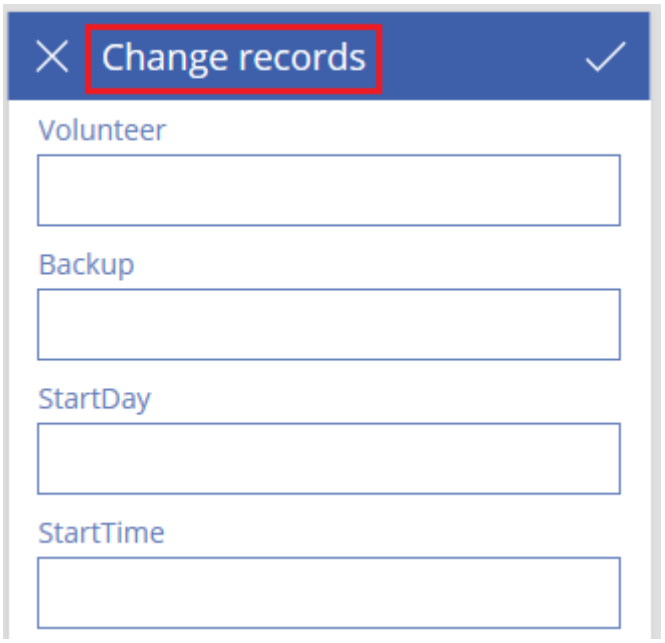
1. On the command bar, select **New screen** > **Form**.
2. In the **Tree view**, select **EditForm1**.
3. On the **Properties** pane, select the down arrow next to **Data source**, and then select **Schedule** in the list that appears.
4. Under the data source that you specified, select **Edit fields**.
5. In the **Fields** pane, select **Add field**, select the check box for each field, and then select **Add**.
6. Select the arrow next to the name of each field to collapse it, and then drag the **Volunteer** field up so that it appears at the top of the list of fields.



7. In the upper-right corner of the **Fields** pane, close it by selecting the close icon (X).
8. Set the **Item** property of the form to this expression in the formula bar:

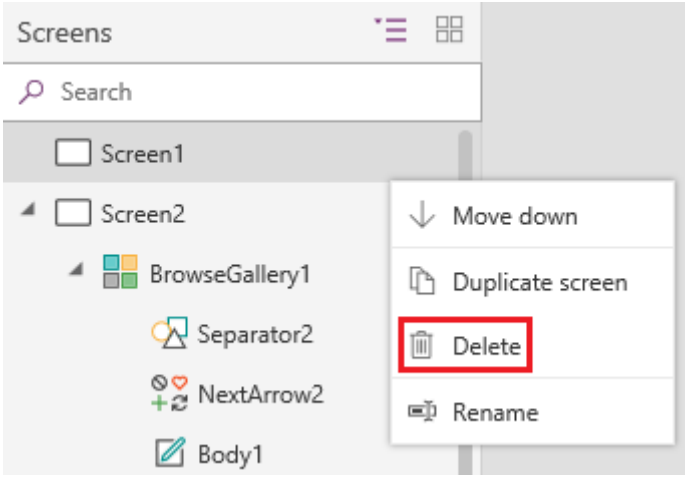
```
BrowseGallery1.Selected
```

9. At the top of the screen, select the **Label** control, and then replace **[Title]** with **Change records**.



Delete and rename screens

1. In the **Tree view**, select the ellipsis (...) for **Screen1**, and then select **Delete**.



2. Select the ellipsis (...) for **Screen2**, select **Rename**, and then type or paste **ViewScreen**.

3. Select the ellipsis (...) for **Screen3**, select **Rename**, and then type or paste **ChangeScreen**.

Configure icons on the view screen

1. Near the top of the **ViewScreen**, select the circular arrow icon.



2. Set the **OnSelect** property for that icon to this formula:

```
Refresh(Schedule)
```

When the user selects this icon, the data from **Schedule** refreshes from the Excel file.

For more information about this and other functions, see formula reference.

3. In the upper-right corner of the **ViewScreen**, select the plus icon.

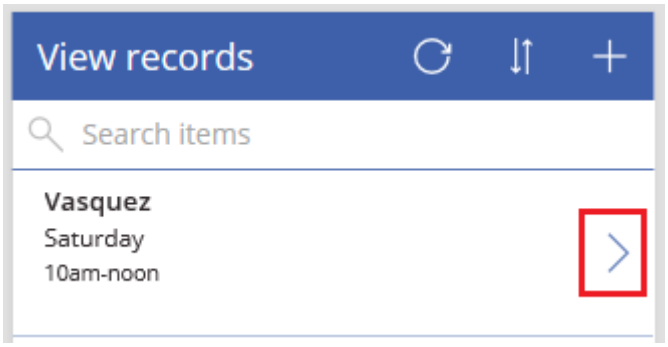


4. Set the **OnSelect** property for that icon to this formula:

```
NewForm(EditForm1);Navigate(ChangeScreen,ScreenTransition.None)
```

When the user selects this icon, **ChangeScreen** appears with each field empty, so the user can create a record more easily.

5. Select the right-pointing arrow for the first record in the gallery.



6. Set the **OnSelect** property for the arrow to this formula:

```
EditForm(EditForm1); Navigate(ChangeScreen, ScreenTransition.None)
```

When the user selects this icon, **ChangeScreen** appears with each field showing the data for the selected record, so the user can edit or delete the record more easily.

Configure icons on the change screen

1. On **ChangeScreen**, select the "X" icon in the upper left corner.



2. Set the **OnSelect** property for that icon to this formula:

```
ResetForm(EditForm1);Navigate(ViewScreen, ScreenTransition.None)
```

When the user selects this icon, the app discards any changes made in this screen and opens the view screen.

3. In the upper-right corner, select the checkmark icon.



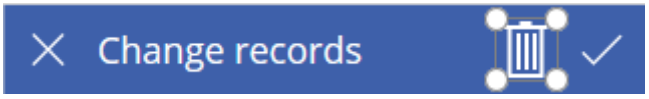
4. Set the **OnSelect** property for the checkmark to this formula:

```
SubmitForm(EditForm1); Navigate(ViewScreen, ScreenTransition.None)
```

When the user selects this icon, the app saves any changes made in this screen and opens the view screen.

5. On the **Insert** tab, select **Icons**, and then select the **Trash** icon.

6. Set the new icon's **Color** property to **White** and move the new icon so it appears next to the checkmark icon.



7. Set the **Visible** property for the trash icon to this formula:

```
EditForm1.Mode = FormMode.Edit
```

This icon appears only when the form is in **Edit** mode, not in **New** mode.

8. Set the **OnSelect** property for the trash icon to this formula:

```
Remove(Schedule, BrowseGallery1.Selected); Navigate(ViewScreen,
ScreenTransition.None)
```

When the user selects this icon, the app deletes the selected record from the data source and opens the view screen.

Test the app

1. Select the **ViewScreen**, and then preview the app by pressing `F5` or select **Preview**.



2. Type or paste one or more letters in the search box to filter the list based on the volunteer's name.

3. Select the sort icon one or more times to show the data in ascending or descending order based on the volunteer's name.
4. Add a record.
5. Update the record that you added, and then save the changes.
6. Update the record that you added, and then cancel the changes.
7. Delete the record that you added.
8. Close Preview mode by pressing Esc or selecting the close icon in the upper-right corner.

Next steps

- Press Ctrl+S to save your app in the cloud so you can run it from other devices.
- Share the app so other people can run it.

Last updated on 02/24/2026

Create a canvas app from a template

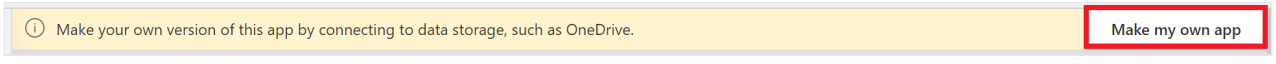
Creating a canvas app by using templates in Power Apps is a streamlined and efficient way to develop applications tailored to your needs. By using templates, you can quickly create a mobile app with three screens or choose from various templates designed for real-world scenarios with fictitious data. These templates come with prebuilt layouts, controls, and sample data, so you can focus on customizing the app to meet your specific requirements.

Create a three-screen mobile app

1. Sign in to Power Apps.
2. From the home screen, select the **Start with an app template** tile.
3. If your data is stored in Dataverse, select **From Dataverse**. Or choose from one of the external data sources:
    - **From SharePoint**
    - **From Excel**
    - **From SQL**
4. Select the data that appears in your app. When ready, select **Create app**.
5. The app opens in Power Apps Studio, where you can customize it.
6. Preview the app by pressing F5 or selecting the play button near the upper-right corner.
7. When ready, save and publish the app.

Use other templates to create an app

1. Sign in to Power Apps.
2. On the home screen, select the **Start from template** tile.
3. Select a template from the list of **other app templates**.
4. Update the **App name**, and then select **Next**.
5. Select **Make my own app** to connect the app to data storage, such as OneDrive.



6. After you select or connect to storage from the available choices, select **Done**.

The app opens in Power Apps Studio where you can customize it.

1. Preview the app by pressing F5 or selecting the play button near the upper-right corner.

When you're ready, save and publish the app.

See also

Share a canvas app

Last updated on 01/13/2026

Microsoft Power Apps documentation for canvas apps

Learn how to build and design canvas apps in Microsoft Power Apps.

Get started

OVERVIEW

Understand Power Apps Studio

Understand data sources and connections

Build responsive apps

Understand coauthoring and copresence

TRAINING

Get started with Power Apps canvas apps

Connect to data

CONCEPT

Overview of connectors

Connect to Microsoft Dataverse

HOW-TO GUIDE

Add data connections

Manage connections

Add screens, controls, and data elements

HOW-TO GUIDE

Add and navigate screens

Add and configure controls

Add a Copilot control (preview)

Create and edit tables using the table designer

Work with a gallery

HOW-TO GUIDE

Add a gallery

Add variable-height gallery

Show, sort, and filter data in a gallery

Best practices and recommendations

Add and configure forms

CONCEPT

Understand canvas-app forms

Understand form layout

Understand data cards in canvas apps

HOW-TO GUIDE

Show, edit, or add a record

Display a full-sized image

Add a list, chart, or multimedia

HOW-TO GUIDE

Add a list box, a drop-down list, a combo box, or radio buttons

Create dependent drop-down lists

Add a chart

Add multimedia

Work with controls

CONCEPT

Overview of modern controls and themes

Core properties

Add mixed reality controls

Add geospatial controls

HOW-TO GUIDE

Modern controls and properties

Rename controls (preview)

Use modern themes

Configure app functionality

CONCEPT

Get started with formulas

Understand behavior formulas

Formula reference

HOW-TO GUIDE

Use Copilot to create and edit Power Fx formulas

Build an AI agent to automate your business process (preview)

Create and update a collection

Overview of connectors for canvas apps

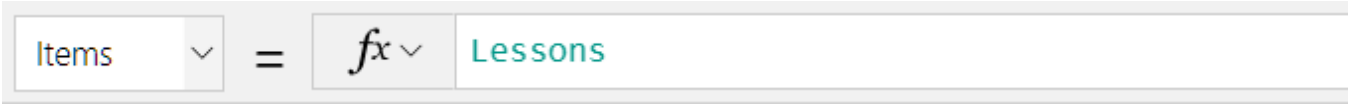
Data is at the core of most apps, including apps you build in Power Apps. You store data in a *data source*, and you bring that data into your app by creating a *connection*. The connection uses a specific *connector* to talk to the data source. Power Apps has connectors for many popular services and on-premises data sources, including SharePoint, SQL Server, Office 365, Salesforce, and Twitter. To get started adding data to a canvas app, see Add a data connection in Power Apps.

A connector might provide **tables** of data or **actions**. Some connectors provide only tables, some provide only actions, and some provide both. Also your connector might be either a standard or custom connector.

> **Note**
>
> Keep the number of connectors in a canvas app to a maximum of 10, and connection references to no more than 20. Going beyond these limits can lead to longer loading times for users when launching the app and can cause problems when saving the app.

Tables

If your connector provides tables, add your data source, and then select the table in the data source that you want to manage. Power Apps retrieves table data into your app and updates data in your data source automatically. For example, add a data source that has a table named **Lessons**, and then set the **Items** property of a control, such as a gallery or a form, to this value in the formula bar:



Specify the data that your app retrieves by customizing the **Items** property of the control that shows your data. Continuing the previous example, sort or filter the data in the **Lessons** table by using that name as an argument for the **Search** and **SortByColumn** functions. In this graphic, the formula set for the **Items** property specifies that the data is sorted and filtered based on the text in **TextSearchBox1**.



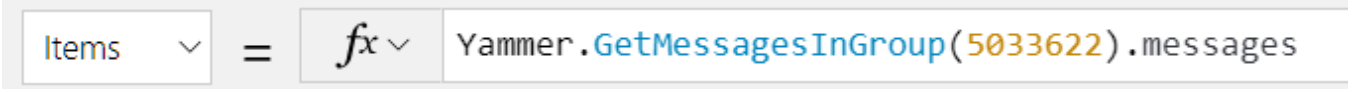
For more information about customizing your formula with tables, see these articles:

Understand data sources in Power Apps
Generate an app from Excel data
Create an app from scratch
Understand tables and records in Power Apps

> **Note**
>
> To connect to data in an Excel workbook, host it in a cloud storage service like OneDrive. For more information, see Connect to cloud-storage from Power Apps.

Actions

If your connector provides actions, select your data source as you did before. Instead of selecting a table as the next step, manually connect a control to an action by editing the **Items** property of the control that shows your data. The formula you set for the **Items** property specifies the action that retrieves data. For example, the app doesn't retrieve any data if you connect to Yammer and then set the **Items** property to the name of the data source. To populate a control with data, specify an action such as **GetMessagesInGroup(5033622).messages**.



To handle custom data updates for action connectors, build a formula that includes the **Patch** function. In the formula, identify the action and the fields that bind to the action.

> **Note**
>
> For action-based connectors, galleries and other controls don't automatically page in more data like they do for tabular connectors. For example, if you bind a tabular data source to a gallery, it retrieves the first set or page of records (for example, 100 records), and then it pages in more data as the control requests it. For an action-based connector, it retrieves a "page" of data, but if the data requested exceeds the page size, the control doesn't automatically get the next page.

For more information about how to customize your formula for custom updates, see these articles:

Patch
Collect
Update

Dynamic schema is a common type of result for action based connectors. Dynamic schema refers to the possibility that the same action might return a table with different columns depending on how it's called. Conditions that might cause the columns in the table to differ include input parameters, the user or role executing the action, and the group in which the user is working, among others. For example, SQL Server stored procedures might return different columns if run with different inputs, or an Azure DevOps instance might use custom fields that aren't available by default.

> **Note**
>
> The connector documentation shows dynamic schema results with this message **"The outputs of this operation are dynamic."** as the return value.

For more information about how to work with dynamic schema in Power Apps, see Working with Dynamic values for an overview and Connect to Azure DevOps from Power Apps for a detailed example.

Popular connectors

This table links to more information about popular connectors. For a complete list, see All connectors.

Expand table

Microsoft Dataverse	Cloud storage **
Dynamics AX	Excel
Microsoft Translator	Office 365 Outlook
Office 365 Users	Oracle
Power BI	SharePoint
SQL Server	Twitter

** Applies to Azure Blob, Box, Dropbox, Google Drive, and OneDrive.

Standard and custom connectors

Power Apps provides *standard* connectors for many commonly used data sources. If Power Apps has a standard connector for the type of data source you want to use, use that connector. To connect to other types of data sources, like a service you built, see Register and use custom connectors.

All standard connectors

Standard connectors don't require special licensing. For more information, see Power Apps plans.

Ask questions about a specific connector in the Power Apps forums, and suggest connectors you want to add or other improvements in Power Apps Ideas.

Security and types of authentication

As you author your app and create a connection to a data source, you might see that your choice of connector offers different ways to authenticate. For instance, the SQL Server connector allows you to use Microsoft Entra Integrated, SQL Server Authentication, and Windows Authentication. Each type of authentication has different levels of security. Understand what information and rights you share with users who use your application. The primary example in this article is SQL Server. However, the principles apply to all types of connections.

> **Note**
>
> - For detailed information about security considerations when using a relational database server, such as Microsoft SQL Server or Oracle, as the data source for an app, see Use Microsoft SQL Server securely with Power Apps.
> - Power Apps doesn't support **External member** identities. For more information, see Properties of a Microsoft Entra B2B collaboration user.
> - Gateway selection isn't supported for custom connectors that use the anonymous authentication type.

Microsoft Entra ID

This authentication is a secure type of connection. For example, SharePoint uses this type of authentication. SQL Server also allows for this type of authentication. When you connect, the Microsoft Entra service identifies you separately to SharePoint on your behalf. You don't have to supply a username or password. As an author, you can create and work with the data source

by using your credentials. When you publish your application, your application user signs in by using their credentials. If the data is appropriately secured on a back-end, your users can only see what they're authorized to see based on their credentials. This type of security allows you to change rights for specific application users on the back-end data source after the application is published. For instance you can grant access, deny access, or refine what a user or set of users can see all on the back-end data source.

Open-standard authorization (OAuth)

This type of connection is also secure. For example, Twitter uses this type of authentication. When you connect, **you must supply your user name and password.** As an author, you can create and work with the data source by using your credentials. When you publish your application and your application user signs in, they must also supply their credentials. Therefore this type of connection is secure as your users must use their own credentials to access the data source service.

Shared connections and secure implicit connections

In a shared connection, the Power Apps author provides the user name and password for the connection when creating the data source in the application. The connection authentication to the data source is then **implicitly shared** with end users. Once the application is published, the connection is also published and available to your users.

Before January 2024, your end users could take the connection that you shared with them and create separate new applications. Your users can't see the user name or password, but the connection is available to them. However, **after January 2024, all newly created shared connections are secured.** To secure existing apps, republish them. The connection is no longer shared with end users. The published Power App talks to a connection proxy. The connection proxy only talks to the specific Power App for which it's linked. The connection proxy limits the actions that are sent to the connections to the ones in the Power App **{Get, Put/Patch, Delete}** for a given data source. If you have an app using the connections published before January 2024, republish your application and unshare any connections with end users that shouldn't have them.

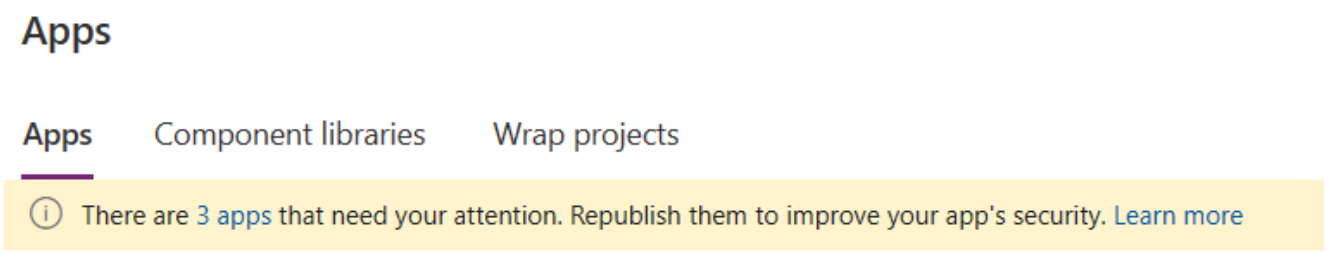
In SQL Server, an example of this type of connection is **SQL Server Authentication**. Many other database data sources provide a similar capability. When you publish your application, your users don't need to supply a unique user name and password.

> ⓘ **Note**

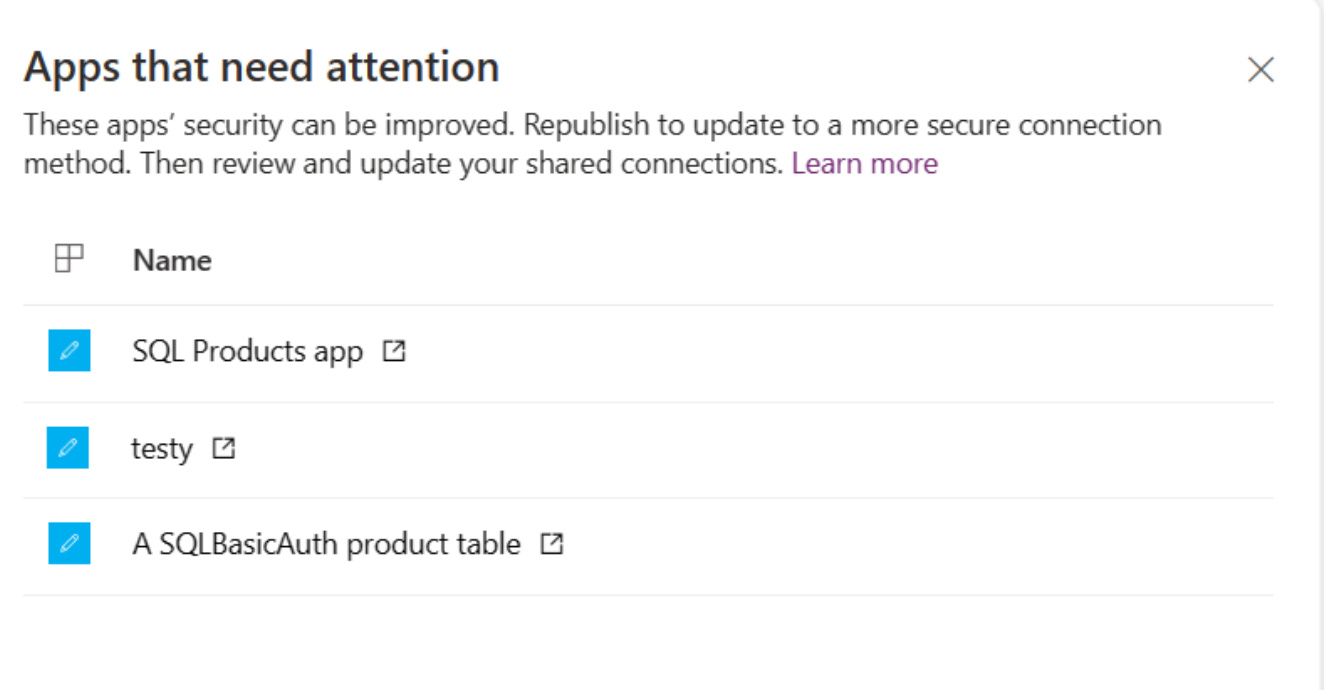
Your end-users might encounter the error message **You do not have correct permissions to use this connection** in the consent dialog. Two situations might cause this error. First, the application might have a shared implicit connection that **isn't** a secure implicit connection. Sharing the connection with the end user resolves this issue but isn't recommended because all shared connections should be secure implicit connections. The author should convert all connections in the application to be secure implicit connections to resolve this issue. Second, the connection might already be a secure implicit connection. Republishing might resolve this issue. If it doesn't, then a product bug should be filed.

Notification to update your apps (secure implicit connections)

If you have applications that might be upgraded to use this feature, you see a message on the **Apps** page. It indicates the number of apps that need your attention.



Select the link and it opens a side panel that lists all of the apps that need attention.



Select the **open** icon to the right of the app name to open and republish it. Continue with the following directions.

Enable secure implicit connections for an existing app

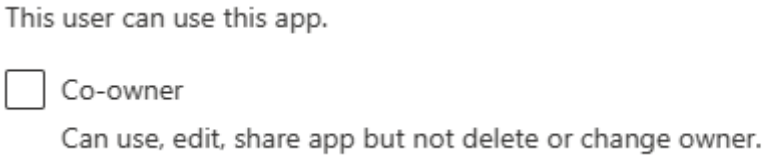
Open an existing app open for editing with implicitly shared connections already published:

1. On the command bar, select **Settings** and search for "**Secure**".
2. Update the feature switch appropriately to enable secure implicit connections.
3. Save and publish the app.

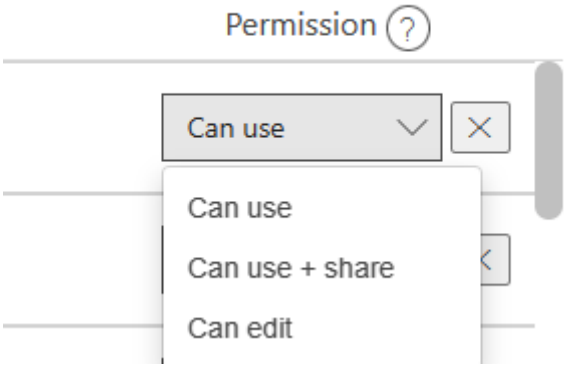
Unsharing

Once you publish the app, follow these steps to verify that sharing works correctly:

- Check if connections are shared with co-owners. If you don't want an end-user to get a connection, then uncheck the **Co-owner** checkbox.



- To verify the feature works correctly, share the app with a different user who isn't an owner. Once you share the app, check the **Connections** list in the **Dataverse** tab in Power Apps for that user. Verify that the user doesn't have a connection available.
- Open the **Sharing** panel to change the end-user's right to the connection. Choose the **X** to remove the user's access to the connection.



Use apps with a new secure implicit connection

When you republish and share your app, end-users don't have access to the connection but work with the hidden proxy connection. Users can't create a new app based on your original connection.

Limitations

1. All types of implicitly shared connections work, such as action and tabular.
2. Server and database names are hidden in network traces but visible in the consent dialog. Column names aren't hidden.
3. For tabular connectors, the feature only limits CRUD actions such as Get, Post, Put, or Delete. If you have permissions to **Put**, then you have access to **Post**.
4. Action based connectors limit based on the specific API being used in the application.
5. Warnings are still enabled in sharing. The warning around implicitly shared connections still warns while in preview. However, your connection with this feature is secure – despite the warning.
6. Publishing to an entire tenant, as opposed to specific groups or individuals isn't supported.
7. There's a known issue when importing an implicitly shared secure connection via a connection reference. The security isn't set properly in the target environment.
8. There's a known issue importing a solution using a service principal, causing import failure. A workaround is to share the connection with the service principal.

Windows Authentication

This type of connection isn't secure because it doesn't rely on end-user authentication. Use Windows authentication when you need to connect to a data source that is **on-premises**. An example of this type of connection is to an on-premises server that has a SQL Server. The connection must go through a gateway. Since it goes through a gateway, the connector has access to all of the data on that data source. As a result, any information that you can access with the Windows credentials you supply is available to the connector. When you publish the application, you also publish the connection and make it available to your users. This behavior means that your end users can create applications by using this same connection and access the data on that machine. Connections to the data source are also **Implicitly Shared** with users that the app is shared with. This type of connection might be valid when your data source only lives on an on-premises server and the data on that source is freely shareable.

Data sources in solutions

Solutions help with application lifecycle management and offer other ways to manage the lifecycle of **data sources**. If a canvas app is in a solution, you can create connection references and environment variables to store information about the data sources. This setup makes it easy to change or reconnect data sources when you move solutions to different environments.

Rename data sources in apps

Learn how to rename data sources in an app, and understand the difference between tabular and action-based data sources. For more information, see Rename Power Apps action-based data sources.

Connection consent dialog

When users open an app that uses connectors for the first time, they see a connection consent dialog for the following purposes.

1. To inform users about the data sources accessed by the app.
2. To outline the actions a connector might or might not perform in an app. For example, for apps using the **Office 365 Users** connector:

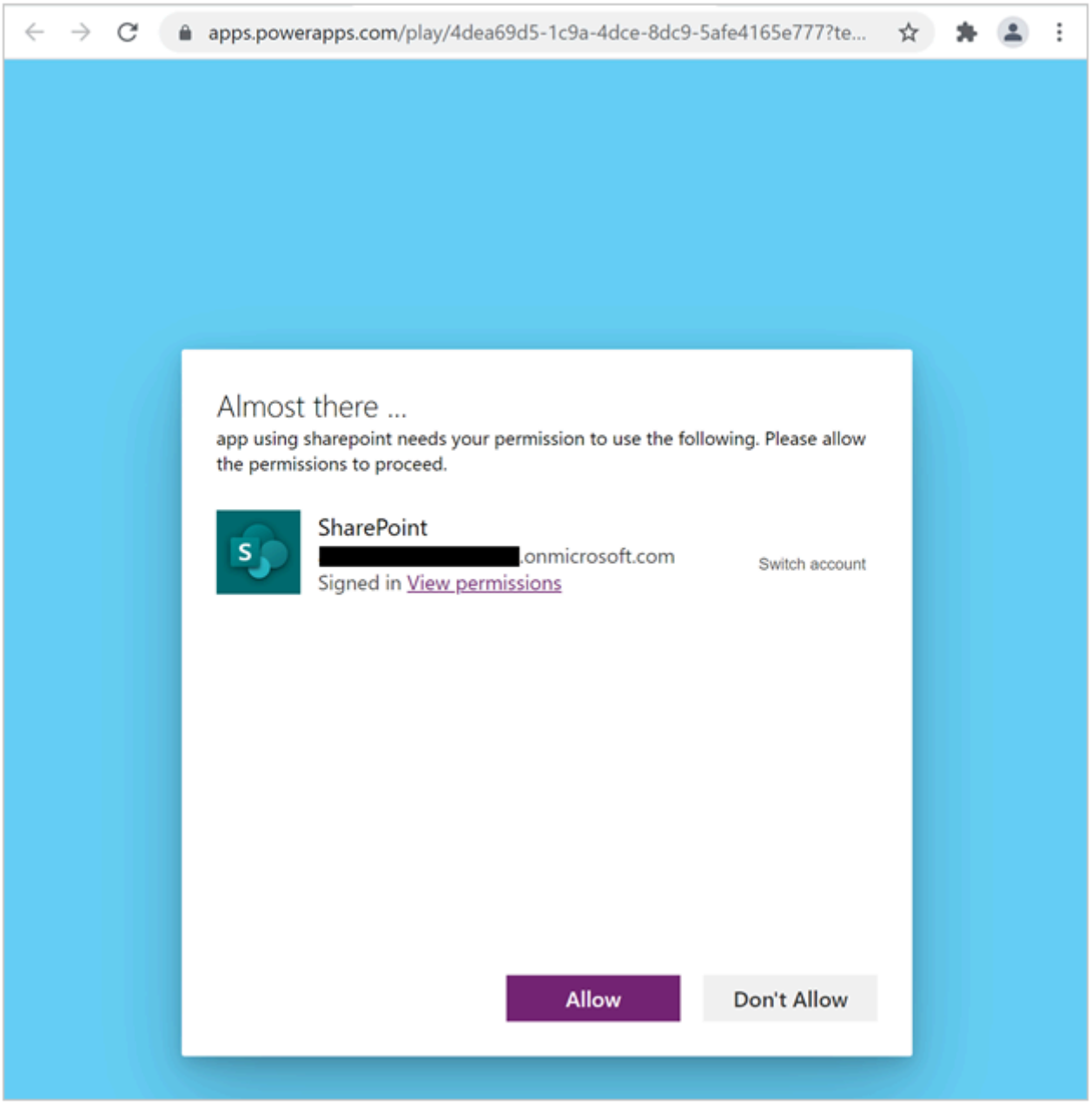
- This app is able to:
  - Read your full user profile
  - Read the full profile of all users
- The app can't:
  - Modify or delete any user-profile information

1. To capture end-user consent to connect to the data sources that the app uses.
2. To facilitate manual end-user authentication, when needed.

For some connections, Power Platform can automatically authenticate a user to access a data source. However, if the automatic sign-in fails, this dialog prompts users to fix a connection by manually signing in. Power Platform can only attempt automatic sign-in for a connection when a data source preauthorizes Microsoft's Azure API connections service principal, granting it permission to perform single sign-on for a user when a connection is created. For more information on single sign-on, see What is single sign-on (SSO)?

For model-driven apps that use custom pages, when there are multiple custom pages in an app, the consent dialog asks for data permissions for all of the connectors in all the custom pages even if users don't open them.

The following image is an example of the connection consent dialog for an app connecting to a SharePoint site.



For select connectors, admins can suppress this dialog and consent on behalf of end users to connect to a data source. The following table explains which types of connectors the consent dialog might be suppressed for an app.

> **Note**
>
> If an admin suppresses the consent dialog but the platform can't perform single sign-on for an end-user, the dialog is presented to the user when they launch the app.

Expand table

Connector type	Consent dialog suppressible?	Reference
Microsoft connectors that support single sign-on (such as SharePoint, Office 365 users)	Yes	Power Apps admin cmdlet
Connector accessing a non-Microsoft, partner service, such as Salesforce	No	Not applicable
Custom connectors using OAuth with Microsoft Entra ID as the identity provider. These custom connectors are built by organizations, and are only accessible by the users within the organization (for example, built by Contoso for only Contoso users)	Yes	Manage Connections

Microsoft Power Platform can only suppress the consent dialog for connections to data sources where all the following conditions are true:

1. The data source doesn't require an explicit consent UI.
2. The data source preauthorizes Microsoft's Azure API connections service principal to enable single sign-on.
3. An admin configures an app to suppress the consent for the preceding connections.

Microsoft's Azure API connections service principal preauthorization exists for Microsoft's first-party data sources. Custom applications registered in a Microsoft Entra tenant that are used by custom connectors might configure this preauthorization. An admin manages consent suppression on a per-app basis (as opposed to connector basis), so suppression is managed at the most granular app experience level. This level of granularity prevents consent suppression for an organization's "approved apps" from inadvertently suppressing consent for apps that aren't approved or reviewed.

Last updated on 01/13/2026

Working with dynamic values

06/21/2025

When you use actions in Power Apps, you can encounter dynamic return values or input values for some actions. Previously, Power Apps ignored dynamic input fields, and they weren't visible in Power Fx expressions. Now, you work directly with these fields. Previously, when a return type was dynamic, Power Apps returned a Boolean value. Now, it returns a dynamic value instead.

> **Note**
>
> If your Power Fx expressions rely on a Boolean return value from these functions, rewrite the formula and explicitly cast the dynamic value to a Boolean. Certain functions, like 'IfError,' don't fully support dynamic values yet. If your expression uses one of these functions, see the note at the end of this article for workarounds.

Passing in dynamic values as parameters

Certain actions necessitate using a dynamic value as a parameter. If you have a Power Fx record, convert it to a dynamic value to pass it to the action.

In the following example, the merge action on a Dataverse **Account** table requires several dynamic arguments. To prepare, define three variables to hold the TargetObject, SubordinateObject, and UpdateContextObject. Start by assigning the text string **Microsoft.Dynamics.CRM.account** to a variable, which you reuse throughout the example.

Power Fx

```
Set (OdataType, “Microsoft.Dynamics.CRM.account”);
```

Then assign TargetObject a Power Fx record with the properties name, accountid, and @odata.type. Similarly, assign Power Fx records to the Subordinate and UpdateContext objects.

Power Fx

```
Set (TargetObject, {name: "Test 2", accountid: "145dc2ba-85a2-ed11-aado-
0022482d76a5", '@odata.type': OdataType});
Set (SubordinateObject, {name: FirstRecord.’Account name’, accountid:
FirstRecord.Account, ‘@odata.type’ : OdataType });
Set (UpdateContextObject, {telephone1: FirstRecord.’Main Phone’, address1_city:
FirstRecord.’Address 1 : City’, ‘@odata.type’ : OdataType });
```

Next, create three more variables to store the dynamic records after the conversion: TargetDynamicValue, SubordinateDynamicValue, and UpdateContextDynamicValue. To convert, use the ParseJSON(JSON()) function on the original variables. This action transforms the Power Fx records into dynamic values.

Power Fx

```
Set (TargetDynamicValue, ParseJSON(JSON(TargetObject)));
Set (SubordinateDynamicValue, ParseJSON(JSON(SubordinateObject)));
Set (UpdateContextDynamicValue, ParseJSON(JSON(UpdateContextObject)));
```

Finally, call the merge action and pass in the necessary parameters for both dynamic and specific types:

Power Fx

```
Environment.Merge({Target: TargetDynamicValue, Subordinate:
SubordinateDynamicValue, UpdateContent: UpdateContextDynamicValue,
PerformParentingChecks: false  });
```

Using dynamic values returned via an action

If an **Action** based connector returns an object, you can access its properties directly, even if they don't have a type. But if you want to use a property for something specific in Power Apps, like labeling, cast it first.

In this example, the `httpRequest` function returns a dynamic value that's already cast as a Boolean.

Power Fx

```
Set (response, Office365Groups.HttpRequest("/v1.0/me", "GET", ""));
```

One of the properties in the response is displayName. It can be accessed, and cast, with a Power Fx expression like the following:

Power Fx

```
Text(response.displayName)
```

Cast to the object **Text** to use it in a Power Apps label control.

Working with Dynamic fields

Action responses now capture dynamic output, and you can use the method described above to access these properties. You can also work with dynamic input fields.

Consider the `GetMessageDetails` action in Microsoft Teams, which has a dynamic input `body` parameter. Previously, you couldn't view or specify this parameter. With the recent update, set a variable called `body` with the appropriate Power Fx record structure.

Power Fx

```
Set ( body, ParseJSON(JSON( {recipient: { groupID: “7f733b36-7c7f-4f4c-9699-
0a7b7a2b3897”, channelID: “19: 085d522328fb4a439220641006f7f25@thread.tacv2”}})));
```

Then, we can call the GetMessageDetails action and assign the response to the teamsResponse variable.

Power Fx

```
Set (teamsResponse, MicrosoftTeams.GetMessageDetails ( 1661365068558, “channel”,
body ));
```

Converting formulas that return dynamic values that previously returned Boolean

Power Fx takes a limited number of dynamic values so explicit conversion may be necessary for your formula. In particular, if your formula depends on a Boolean response then you need to convert. If you need to simply know if an error exists, you can use the IsError function:

Power Fx

```
If(
  IsError(Office365Outlook.CalendarDeleteItemV2("Calendar", 1)),
  Notify("An Outlook appointment could not be found or could not be deleted")
)
```

To access error information that is exclusively available through IfError, you must transform the dynamic value into a specific type using a conversion function such as Boolean, Text, or Value. These functions will produce an error if they are given one. The following example, illustrates this:

Power Fx

```
With({result: Office365Outlook.CalendarDeleteItemV2("Calendar", 1)},
If( IsError(result),
  IfError(
    Boolean(result),  // any conversion function would do, such as Text, Value, …
    Notify("An Outlook appointment could not be found or could not be deleted: " &
FirstError.Message)
) ) )
```

Manage expired tokens

Article • 04/03/2025

This article explains how tokens are managed to ensure apps continue to function smoothly when credentials change.

API Hub handles connections differently for Power Apps, Logic App, and Power Automate. When you change your credentials, Power Apps automatically reauthenticates using a new token, while Logic Apps and Power Automate might run into issues because they use stored tokens that can become stale.

Manage tokens in Power Apps

When a connection is invoked from Power Apps, API Hub uses the OBO (on-behalf-of) token sent in the 'invoke' request from Power Apps. API Hub exchanges the OBO token for a token to the target service. The OBO token is generated and validated during each Power Apps session before invoking the connector.

For example, if a user **changes their password**, all tokens are revoked. When Power Apps directly invokes the connection, it uses the new token in the 'invoke' request to proceed. The request to the connector succeeds without the user having to reauthenticate the connection.

Manage tokens in Logic Apps and Power Automate

When a connection is invoked from Logic Apps or Power Automate, API Hub uses the stored token in the connection to exchange it for a token to the target service.

For example, **changing passwords or enforcing an MFA policy** revokes all tokens. A Logic Apps or Power Automate SSO (single sign-on) connection call fails during token exchange because the tokens aren't marked as broken, and the token stored in the connection is stale.

Refresh tokens in Power Apps, Logic Apps, and Power Automate

Power Apps automatically renews or reauthenticates any connection that is broken. However, connections aren't marked as broken when tokens are revoked. They're

marked as broken when the token exchange on API Hub fails.

Therefore, Logic Apps and Power Automate must first attempt to run the connector with expired tokens. This action causes the connection to be marked as broken. Once Power Apps encounters a broken connection, it refreshes its local cache of connections, and the connection is repaired.

Feedback

Was this page helpful? Yes No

Provide product feedback

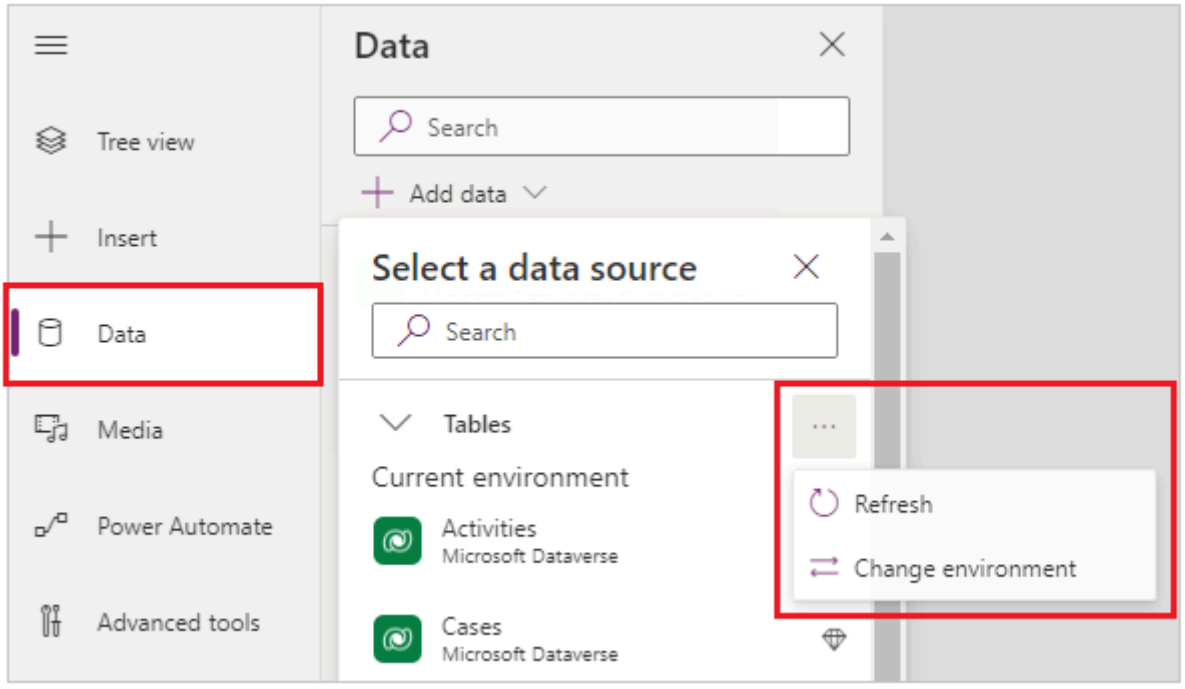
Connect to Microsoft Dataverse

06/21/2025

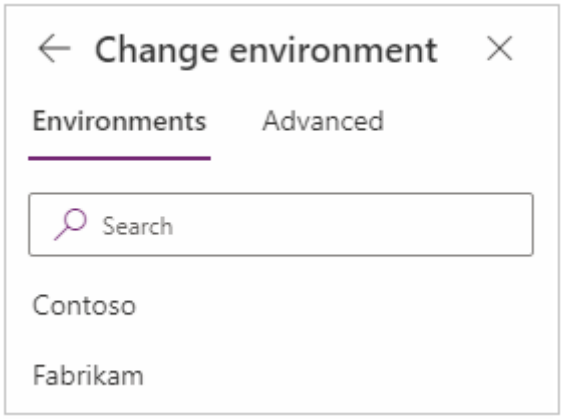
Securely store your business data in Dataverse and build rich apps in Power Apps so users can manage that data. You can also integrate that data into solutions that include Power Automate, Power BI, and data from Dynamics 365.

By default, the app connects to the current environment for Dataverse tables. If your app moves to another environment, the connector connects to data in the new environment. This behavior works well for an app that uses a single environment or follows an application lifecycle management (ALM) process for moving from development, to test, and then to production.

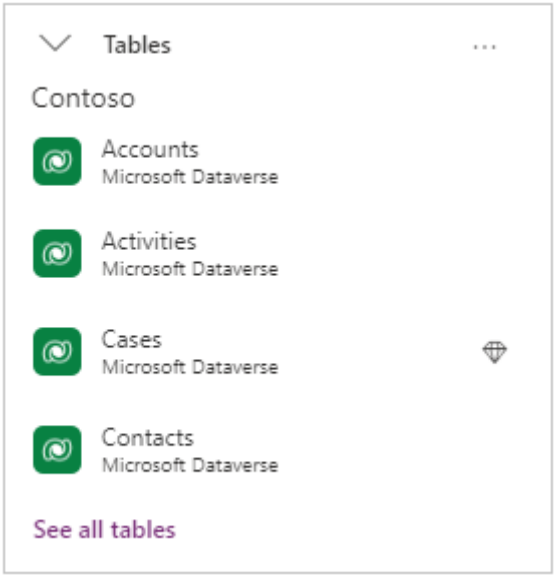
When you add data from Dataverse, change the environment, and then select one or more tables. By default, the app connects to data in the current environment.



If you select **Change environment**, specify a different environment to pull data from instead of, or in addition to, the current environment.



The name of the selected environment appears under the tables list.



Visibility and access

When you select **Change environment**, you see a list of environments. Even if you see an environment in the list, the security roles in the environment control what you can do there. For example, if you don't have read privileges, you can't see the tables and records in the environment.

> **Note**
>
> Connections listed in the app details pane outside of the app designer show connections that need user consent. Because native Dataverse connections used in the app don't need additional consent, a native connection isn't in that list.

Power Apps data type mappings

The Microsoft Dataverse connector is more robust than the Dynamics 365 connector and is approaching feature parity. The following table lists the data types in Power Apps and how they map to data types in Dataverse.

Expand table

Power Apps	Microsoft Dataverse
Choice	Choice, Yes/No
DateTime	Date Time, Date and Time, Date Only
Image	Image
Number	Floating Point Number, Currency, Decimal Number, Duration, Language, TimeZone, Whole Number
Text	Email, Multiline Text, Phone, Text, Text Area, Ticker Symbol, URL
Guid	Unique Identifier

Power Apps delegable functions and operations for Dataverse

These Power Apps operations, for a given data type, can be delegated to Dataverse for processing instead of processing locally within Power Apps.

Expand table

Item	Number [1]	Text [2]	Choice	DateTime [3]	Guid
<, <=, >, >=	Yes	Yes	No	Yes	-
=, <>	Yes	Yes	Yes	Yes	Yes
And/Or/Not	Yes	Yes	Yes	Yes	Yes
CountRows [4] [5], CountIf [6]	Yes	Yes	Yes	Yes	Yes
Filter	Yes	Yes	Yes	Yes	Yes
First [7]	Yes	Yes	Yes	Yes	Yes
In (membership) [8]	Yes	Yes	Yes	Yes	Yes
In (substring)	-	Yes	-	-	-

Item	Number [1]	Text [2]	Choice	DateTime [3]	Guid
IsBlank [9]	Yes	Yes	No	Yes	Yes
Lookup	Yes	Yes	Yes	Yes	Yes
Search	No	Yes	No	No	-
Sort	Yes	Yes	Yes	Yes	-
SortByColumns	Yes	Yes	Yes	Yes	-
StartsWith	-	Yes	-	-	-
Sum, Min, Max, Avg [6]	Yes	-	-	No	-
UpdateIf/RemoveIf [10]	Yes	-	-	No	-

Notes

1. Numeric with arithmetic expressions (for example, `Filter(table, field + 10 > 100)`) aren't delegable. Language and TimeZone aren't delegable. Casting a column to a number isn't supported. If a value appears as a number in Power Apps but the backend data source isn't a simple number, such as currency, then it isn't delegated.
2. Doesn't support Trim[Ends] or Len. Supports other functions like Left, Mid, Right, Upper, Lower, Replace, and Substitute. Also, casting such as Text(column) isn't supported for delegation.
3. DateTime is delegable except for DateTime functions Now() and Today().
4. CountRows on Dataverse uses a cached value. For non-cached values where the record count is under 50,000 records, use `CountIf(table, True)`.
5. For CountRows, ensure that users have appropriate permissions to get totals for the table.
6. All aggregate functions are limited to a collection of 50,000 rows. If needed, use the Filter function to select 50,000 rows. Aggregate functions aren't supported on views.
7. The FirstN function isn't supported.
8. The `In` operator is subject to the 15-table query limit of Dataverse.
9. Supports comparisons. For example, `Filter(TableName, MyCol = Blank())`.
10. UpdateIf and RemoveIf work locally but simulate delegation to a limit of 500 or 2,000 records. They successively bring down records beyond the nondelegation 500 or 2,000 record limit. Records that meet the If condition are collected. Generally, a maximum of 500 or 2,000 records are collected separately and then changed per execution. However, more records can be updated if the existing local data cache is large because the function can access more records for evaluation.

Call Dataverse actions directly in Power Fx

As part of the Power Fx language, you can now directly invoke a Dataverse action within a formula. Both unbound and bound actions are supported. Add a Power Fx `Environment` language object to your app to use Dataverse actions.

You can work with dynamic fields for both inputs and outputs. For inputs, many Dataverse actions require a dynamic value as an argument. Pass these arguments by using ParseJSON to convert a Power Fx record into a dynamic value. For outputs, if an action returns dynamic values, just use dot notation to access object properties. Cast specific values for use in Power Apps, such as in a label.

Before this feature, you often used Power Automate to call Dataverse directly. Calling Dataverse from Power Fx gives you significant performance benefits and is easier to use, so use this approach for direct transactional reads and updates. If your app uses Power Automate to call Dataverse actions, you see a banner suggesting you use this direct action approach instead.

Working with dynamic fields isn't limited to Dataverse. This feature works with all types of connectors and provides basic ad hoc dynamic schema support.

> ⓘ **Note**
>
> 1. DV actions aren't fully supported in Power Fx commanding (for any action call with parameters).
> 2. Direct references to an entity or entity collections aren't supported.
> 3. For parameters of object type that are nested two or more levels deep, Power Apps treats the second-level attributes as required.

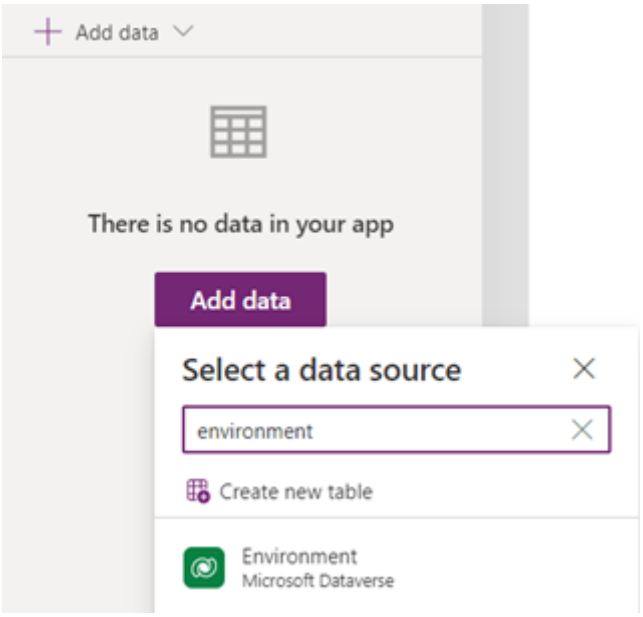
Enable access to Microsoft Dataverse actions

For new apps, this feature is automatically enabled. For apps you created earlier, enable access to Dataverse actions.

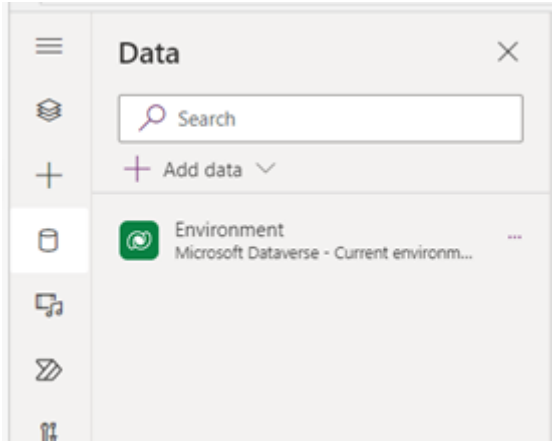
For older apps, open your canvas app for editing. Go to **Settings** > **Upcoming features** > **Retired**, and enable Dataverse actions.

Add the Power Fx Environment language object to your app

To use Dataverse actions in your Power Fx formulas, select **Add data**, search for **Environment**, and add it to your app.

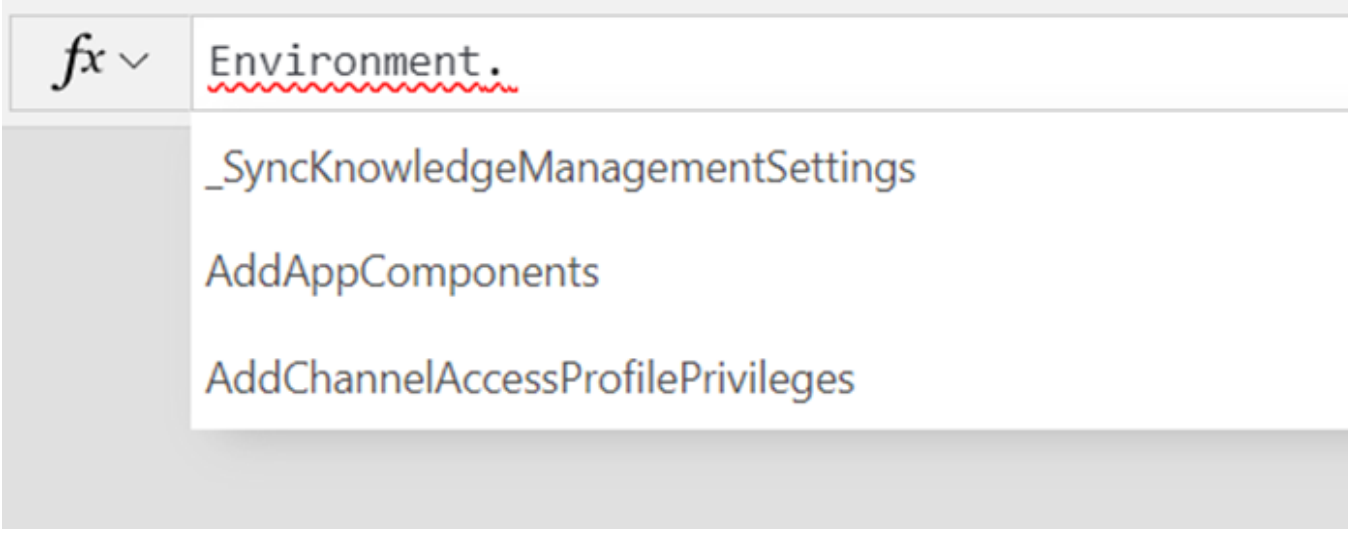


This step adds the Power Fx `Environment` language object to your app.

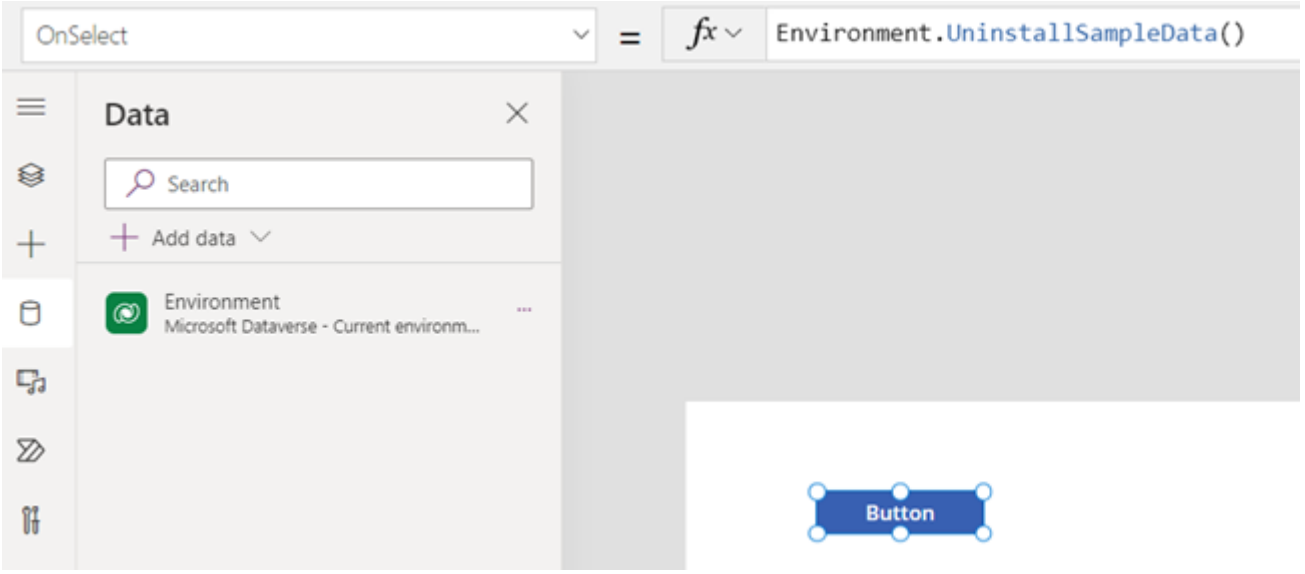


Access Dataverse actions

After you add the Power Fx `Environment` object to your app, access Dataverse actions by adding `Environment` to your formula and then using dot notation for the actions.



Unbound Dataverse actions are at the same level as tables and need the parent scope of the **Environment** language object. All actions in your environment are available—both system and custom. Both bound and unbound actions are available. The two-level call limit is removed.



For more details on how to use Dataverse actions in your formulas, see Working with dynamic values.

Pass entity type arguments for bound and unbound actions

To pass entity type arguments for Dataverse actions, set the entity type argument value to a variable. Make sure to fill in any missing values, such as ***activityId**. This step is important for entities that don't have defined types in the swagger.

Power Fx

```
Set(MyArgVar, {
  name: First(systemUser).name,
```

```
    Id: First(systemUser).Id
    ... })
```

Rename, refresh, and use actions in other environments

To rename an Environment, select the ellipses and then select "Rename". If you add a new Dataverse action in Dataverse and want Power Apps to see it, select "Refresh". To use an action in a different environment, change the environment, then search for 'Environment', select it, and add it to your app.

Microsoft Dataverse and the improve data source experience

Article • 01/24/2024

Overview

In May 2021, many features that were previously optional are now the default behavior for all new apps. This article provides guidance to users who developed an app before the introduction of new features, on how to upgrade their app to leverage the latest functionalities.

Native Dataverse

If you created a canvas app with either the Dataverse or the Dynamics 365 connector before November 2019, then you might not have the benefits of using the current native connection experience for the Dataverse.

If you created your app prior to November 2019 and you aren't using a native Dataverse connection, then in Settings in the Upcoming features tab, in the 'Retired' tab you'll find the **Improve data source experience and Microsoft Dataverse views** option along with the **Relational data, choices, and other new features for Microsoft Dataverse**.

How do I upgrade?

Upgrade your app by inspecting the settings of the features and then by following the directions below:

Relational data, option sets and other new features for Microsoft Dataverse is Off

Check *Retired* section under *Upcoming features*. If set to *Off*, continue with the following instructions as a first step in the conversion.

If you don't see **Relational data, option sets and other new features for Microsoft Dataverse** in *General settings*, or if it's already *On*, skip the following steps and continue to the next section.

- **Step 1**: Turn **Use display names** feature **On**:

1. Turn **Use display names** feature *On*.
2. Wait for the health monitor to finish analyzing your app.
3. Save, close, and reopen your app.
4. Resolve all formula errors.
5. Save, close, and reopen your app.

Possible errors and suggestions:

It's possible that some of the newly shown display names may conflict with the display names for other tables, fields, or controls. For example, you may have a control and a field with the same name. You can change the name of the control with a unique value to fix.

For any field and table display name conflict, you may see a formula that expects a table but resolves to a locally scoped field name.

Use the square bracket with an @ symbol to indicate a global scope so it resolves to the table; for example, **[@tableName]**.

- **Step 2**: Turn **Relational data, option sets and other new features for Microsoft Dataverse** and **Use GUID data types instead of strings** features **On**:

  1. Turn **Relational data, option sets and other new features for Microsoft Dataverse** feature *On*.
  2. Turn **Use GUID data types instead of strings** feature *On*.
  3. Wait for the health monitor to finish analyzing your app.
  4. Resolve all formula errors.
  5. Save, close, and reopen your app.

  Possible errors and suggestions:

  It's possible to have errors at this stage if you're using a choice field or hard-coded GUID text values.

  - *Choice values*: If you're using choice field with a text identifier for the choice value, use the dot notation instead to reference the choice value. For example, change `Patch(Accounts, OptionSet1 = "12345")` to `Patch(Accounts, OptionSet.Item1)` where `Item1` corresponds to the `12345` value.
    More information: Detailed Examples.
  - *GUIDs*: If you're using a static GUID string such as `015e45e1044e49f388115be07f2ee116`, convert it to a function that returns a GUID object; for example `GUID("015e45e1044e49f388115be07f2ee116")`.

- *Lookups*: If you're using Lookup functions to get first-level lookup values such as `Lookup(Contacts, 'contactID' = ThisItem.ContactID")`, consider using `ThisItem.PrimaryContacts` (where PrimaryContacts is the name of the table) instead.

Improve data source experience and Microsoft Dataverse views is Off

Use the following instruction to turn **Improve data source experience and Microsoft Dataverse views** feature *On*:

1. Remove your existing Dataverse data source connections.
2. Turn *On* the **Improve data source experience and Microsoft Dataverse views** feature.
3. Add the Dataverse connection using the new data source selection experience.
4. Save your application.

> ⓘ **Note**
>
> If your application is extremely large, adding your data source connections back may take a while. Don't close the application during this process.

Converting canvas apps with the Dynamics 365 connector

To convert your app that uses the Dynamics 365 connector, you need to remove and add the connections to your data sources. Use the steps below to convert your connections to your data sources.

1. Ensure the **Improve data source experience and Microsoft Dataverse views** feature is turned *On*.
2. Remove your existing Dynamics 365 data source connections.
3. Add the connections to your data sources to the Dataverse using the new data source selection experience.

   > ⓘ **Note**

- If you have connections to other environments (other than current), select the *Table* category and then the *More* (...) option to change the environment. You can then select a table from a different environment to add to your application. Cross-tenant connections don't work with the improved native connector. You'll need to use data integration to access data cross-tenant.
- You need to meet one of the following requirements to be able to see an environment that you want to add the connection to:
  - You're the **Owner** of the app, or the app is shared with you as a **Co-owner**.
  - You're a member of at least one of these security roles: **Environment Admin**, **Environment Maker**, or **System Administrator**. For more information about the security roles in an environment, go to Configure user security to resources in an environment.

4. Save your application.

Possible errors and suggestions:

It's possible to have errors as you convert if: you aren't using Display Names, if you're using GUID strings, or if you're using a choice.

- If control name conflicts, change the name of the control to be different and unique.
- For field and table display name conflicts, you might see a formula that is expecting a table but is resolving to a more locally scoped field name. Use the square bracket with an @ symbol to indicate a global scope so it resolves to the table; for example, **[@tableName]**.
- *Choice values*: If you're using a choice field with a text identifier for the choice value, use the dot notation instead to reference the choice value. For example, change `Patch(Accounts, OptionSet1 = "12345")` to `Patch(Accounts, OptionSet.Item1)` where `Item1` corresponds to the `12345` value.
  More information: Detailed Examples.
- *GUIDs*: If you're using a static GUID string such as `015e45e1044e49f388115be07f2ee116`, convert it to a function that returns a GUID object; for example `GUID("015e45e1044e49f388115be07f2ee116")`.
- *Lookups*: If you're using Lookup functions to get first-level lookup values such as `Lookup(Contacts, 'contactID' = ThisItem.ContactID")`, consider using

`ThisItem.PrimaryContacts` (where PrimaryContacts is the name of the table) instead.
- For any Polymorphic references, refer to the Detailed Examples section below.

Detailed Examples

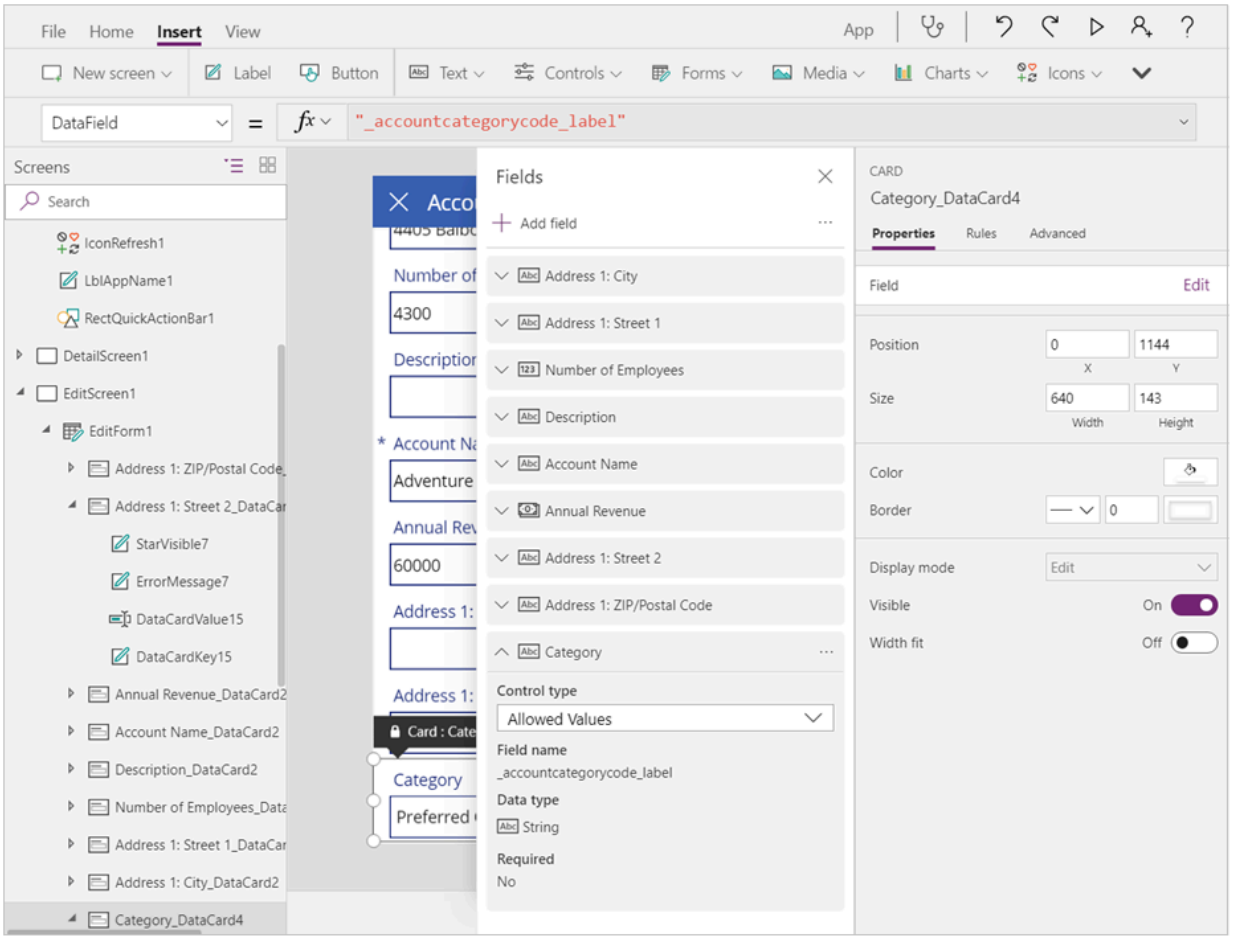
Converting your app to use the new **Option sets** and **Two options** data types with supporting controls can be challenging while upgrading an app to use the new *Improve data source experience and Microsoft Dataverse views* feature.

Choices

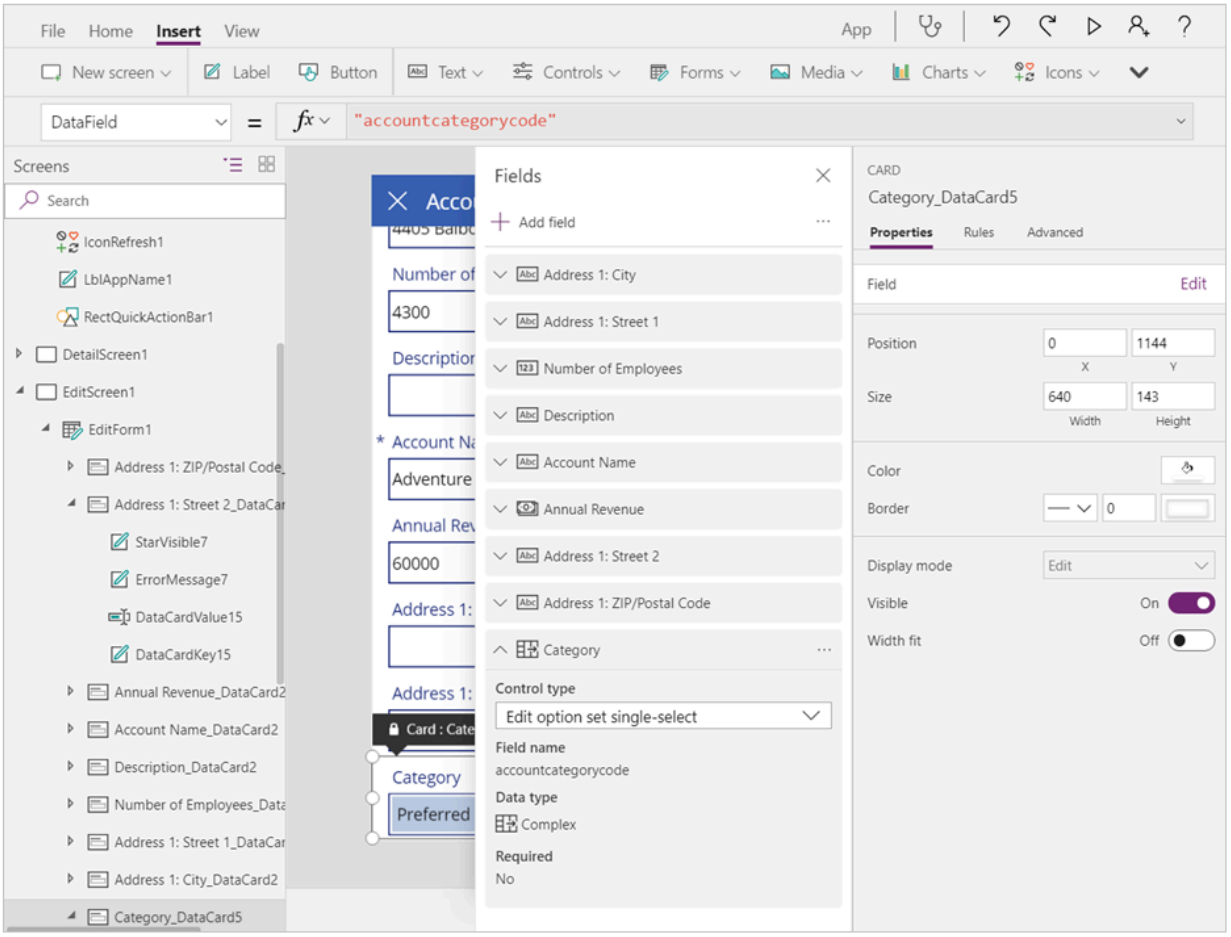
Separate `_myfield` and `_myfield_label` fields were used for choice earlier. Now, there's a single `myfield` that can be used both for locale-independent comparisons and to obtain the locale-specific label.

Removing and adding choice Data cards

We recommend that youremove existing data cards and add them back to work with your choice. For example, if you're working with the Account table and the Category choice, you see that the *DataField* property of the data card was set to `_accountcategorycode_label`. In the field list you can see that the data card has a type of *String*:



With the new *Improved data source experience and Microsoft Dataverse views* feature, you no longer see `_accountcategorycode_label`. It's replaced by `accountcategorycode`. Your card is now be marked as **custom** and you'll see errors. Remove the old data card and add the *Choice* back. The new data card is *Choice* aware.



Editing the Choice Filter expressions to use new syntax

Previously, if you wanted to use a Choice value in a Filter expression you would need to use the *Value* field. For example:

Power Fx

```
Filter(Account,'Category Value' = "1")
```

You'll need to edit this formula. Choice text identifer is no longer used for the value. This expression should be updated to:

Power Fx

```
Filter(Account, Category= ‘Category (Accounts)’.’Preferred Customer’)
```

'Category(Accounts)' is the name of enum used in the Category field of the Accounts table. This is a local choice. You can read more about local and global choices here: Global choices.

Editing choices patch statements to use new syntax

Following is an example of earlier Patch statement for choice:

```
Power Fx

Patch( Accounts, First(Accounts), { ‘Category Value’: 1 } ) )
```

You'll need to update your statements to follow this form:

```
Power Fx

Patch( Accounts, First(Accounts), { Category: ‘Category
(Accounts)’.’Preferred Customer’ } )
```

Choice disambiguation

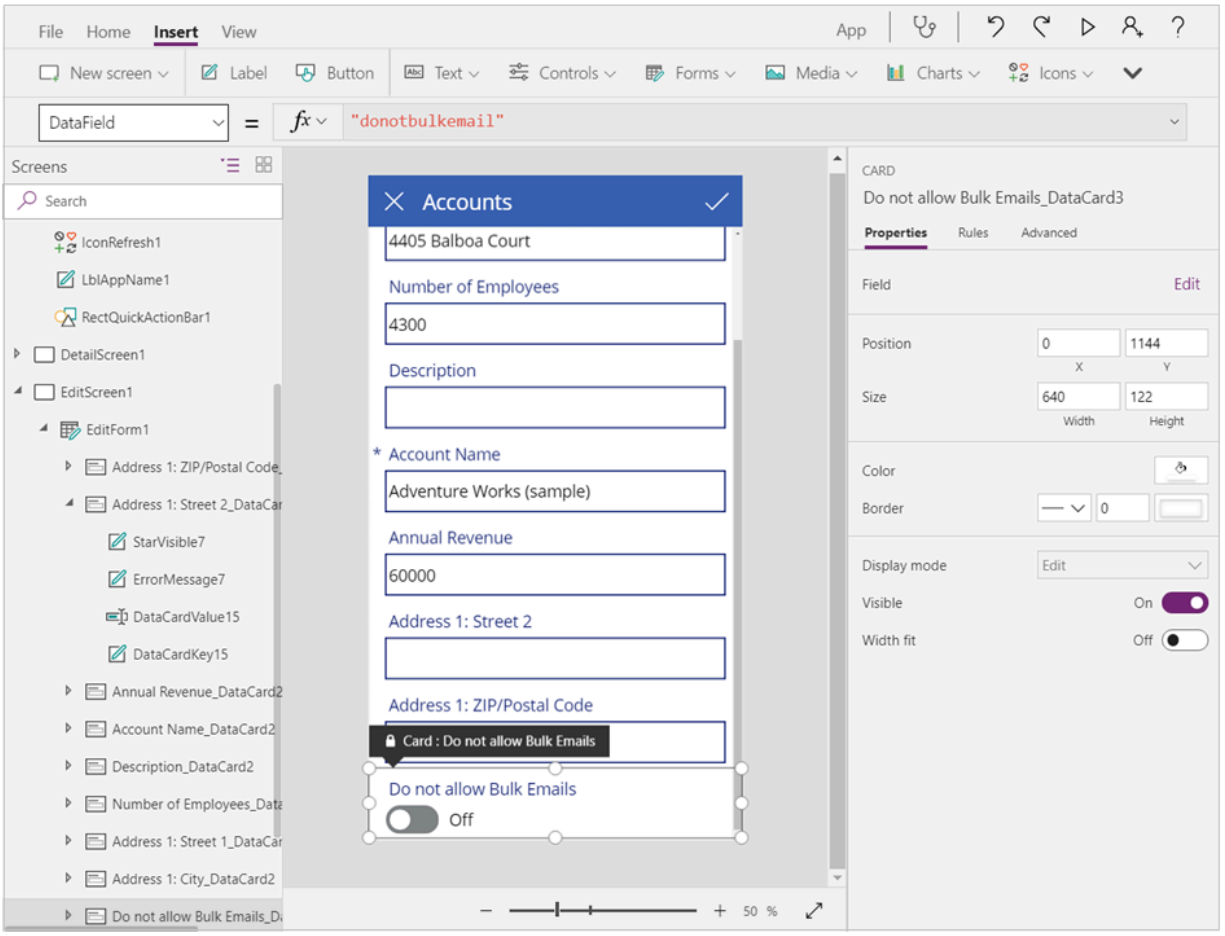
If the display name of an choice **field** and the name of the choice are the same, you'll need to disambiguate the formula. To continue using the Accounts Category Code example, the @ implies to use the choice, not the field.

```
Power Fx

Filter(Accounts, 'Category Code' = [@’Category Code’].'Preferred Customer')
```

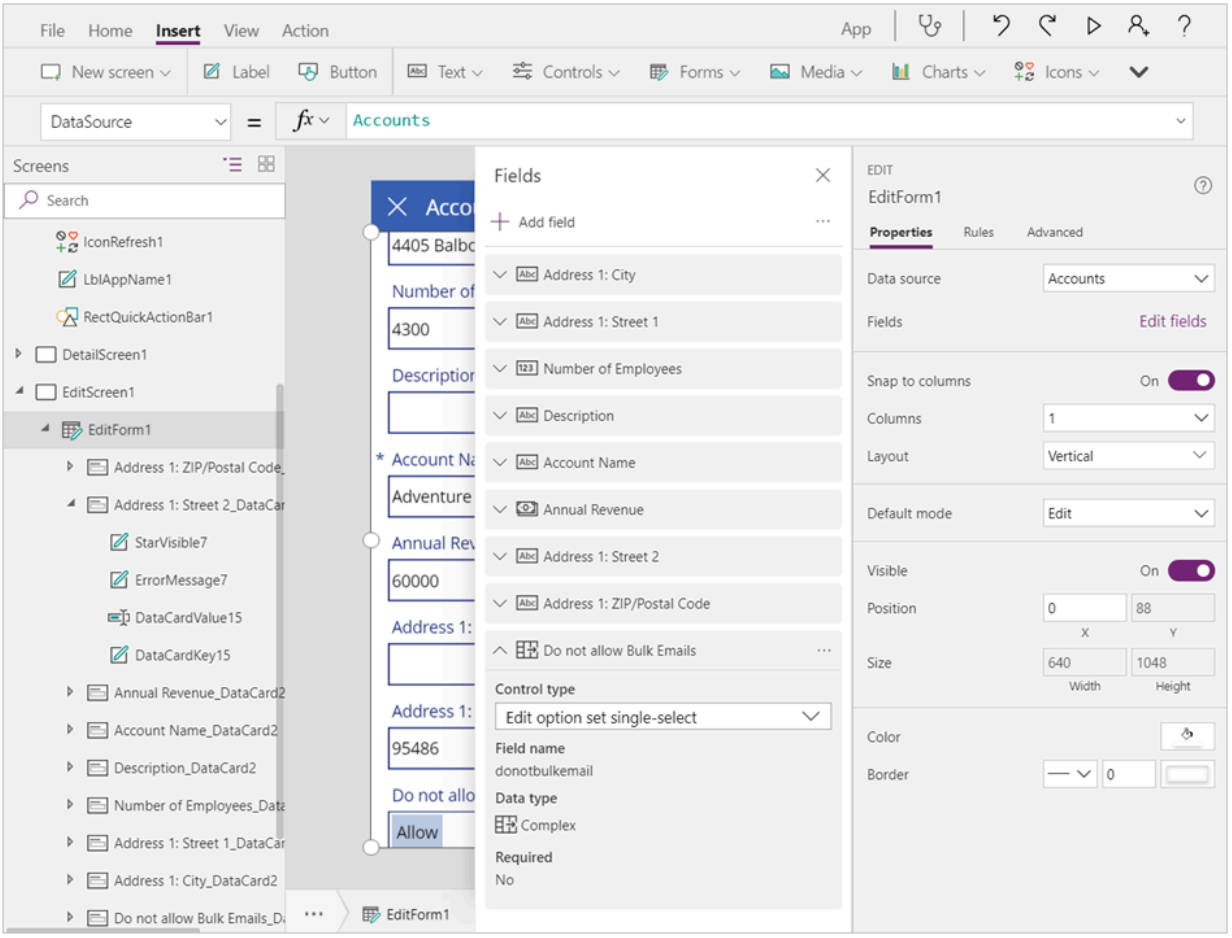
Two Options

Removing and adding Yes/No Data cards

Remove existing data cards and add them back to work with your Yes/No. The data types were earlier recognized as simple boolean - such as true/on and false/off with no labels:



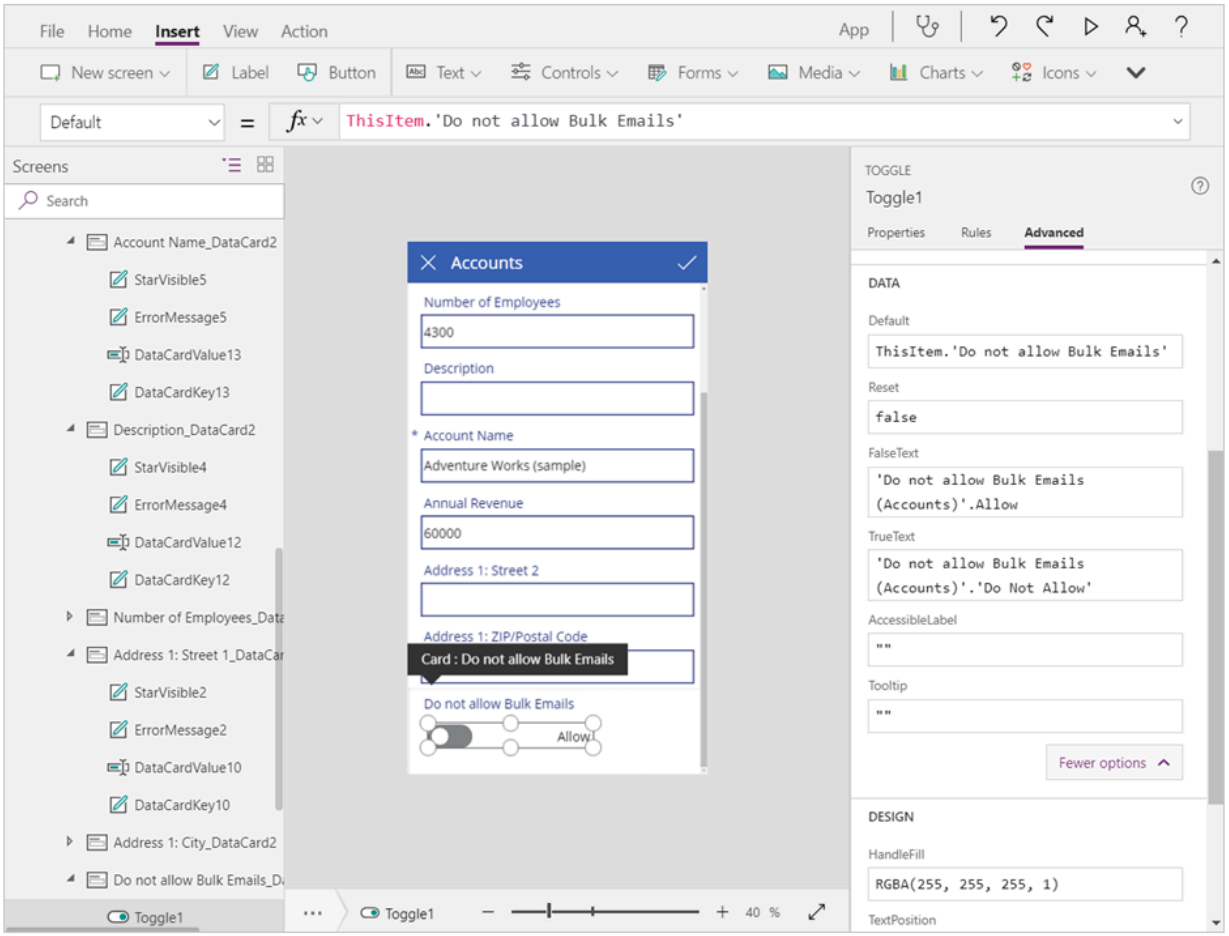
With the new *Improved data source experience and Microsoft Dataverse views* feature, your card will now be marked as **custom** and you'll see errors. Remove the old data card and add the choice back. You'll see an edit control with two options by default after you add.



If you prefer the toggle switch for your boolean field, you can unlock the data card and replace the control in the data card with a toggle instead. You'll also need to set these properties on the Toggle.

Power Fx

```
Toggle1.Default = ThisItem.’Do not allow Bulk Emails’
Toggle1.TrueText = ‘Do not allow Bulk Emails (Accounts)’.’Do Not Allow’
Toggle1.FalseText = ‘Do not allow Bulk Emails (Accounts)’.Allow
DataCard.Value = If( Toggle1.Value,
    ‘Do not allow Bulk Emails (Accounts)’.’Do Not Allow’,
    ‘Do not allow Bulk Emails (Accounts)’.Allow )
```



Refining Two Option Patch statements

Using the Patch function with Two option should work 'as is'. It supports direct use of true and false, similar to Boolean. The only difference is being if you had put the value in a Label control earlier that showed true and false, it will now show the Two option labels instead.

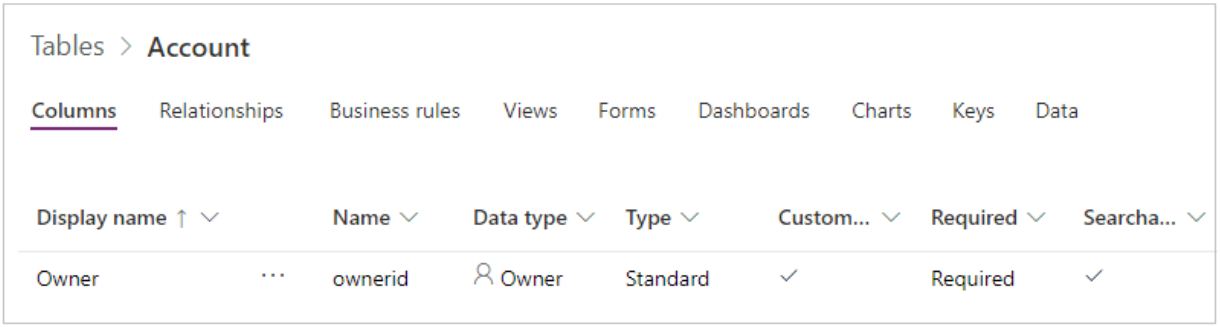
Polymorphic lookups

Following guidelines help to upgrade your application if it referenced polymorphic fields. Polymorphic lookups, from the same field, support references to a restricted set of multiple tables. Similar to references in other languages, a record reference is a pointer to a specific record in a specific table. A record reference carries the table information allowing it to point to a record in several other tables, which differs from a normal lookup that can only point to records in one table.

Access, Set, and Filter on the Owner field of a record

For instance, the Owner field in a table can refer to a record in the Users table or the Teams table. The same lookup field in different records could refer to records in

different tables.



Polymorphic with Filter and Patch

Record references can be used just like a full record:

```
Power Fx
```

```
Filter( Accounts, Owner = First( Teams ) )
Patch( Accounts, First( Accounts ), { Owner: First( Users ) })
```

Polymorphic with a Gallery displaying Owner name

Since a reference can point to different tables, you must be specific. You can't use **ThisItem.Owner.Name**, as the name field in the **Team** table is **Team Name**, and the name field in the **User** table is **Full Name**. Power Apps doesn't know which type of lookup you're referring to, until you run the app.

To fix this issue:

1. Add the data sources for the entity types that Owner could be; in current example, Users and Teams).
2. Use more functions to make your intent clear.

There are two new functions you can make use of:

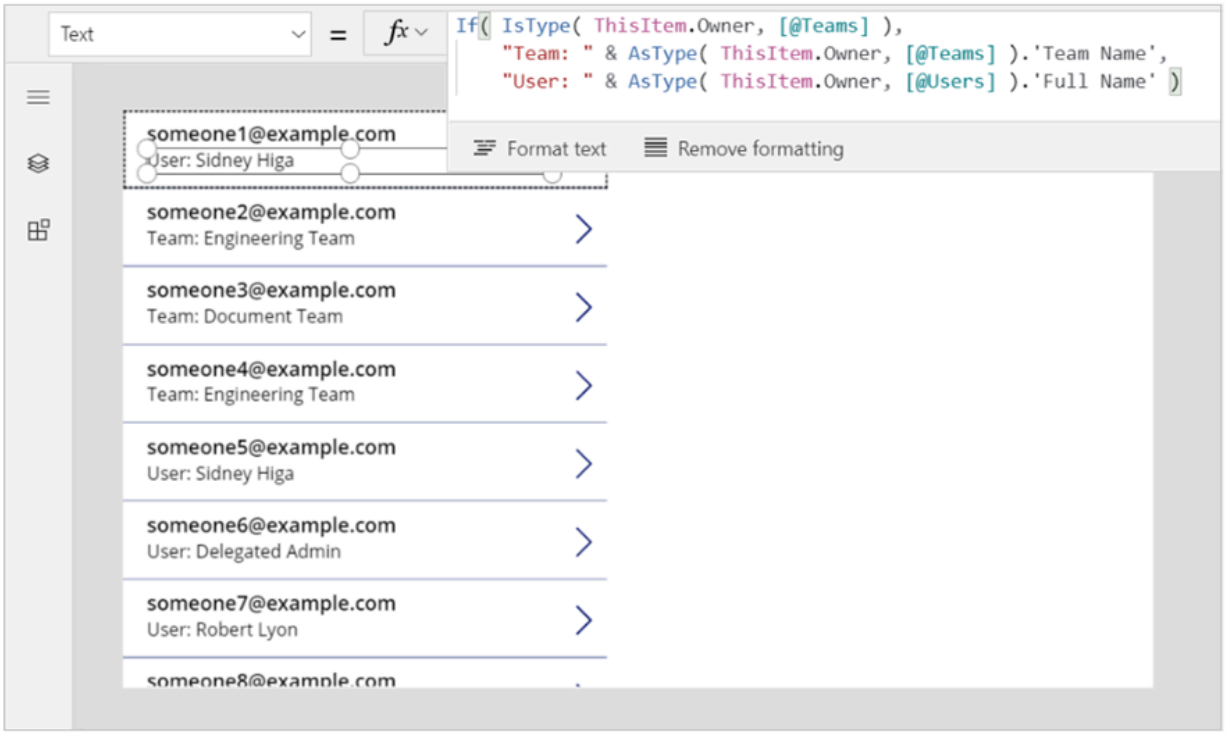
- IsType – Checks if a record reference is of a particular table type.
- AsType – Casts a record reference to a particular table type.

With these functions, you can write a formula that displays the name of the Owner taken from two differently named fields, based on the table type of the Owner:

```
Power Fx
```

```
If( IsType( ThisItem.Owner,  [@Teams]),
    AsType( ThisItem.Owner, [@Teams]).'Team Name',
```

```
    AsType( ThisItem.Owner, [@Users]).'Full Name' )
```



Global disambiguation operator for `[@Teams]` and `[@Users]` is used to ensure that you reference the global table type. Though in this case it's not necessary, it's a recommended to always be clear. One-to-many relationships often conflict in the gallery's record scope, and this practice avoids that confusion.

Access and set the Company Name field (a Customer data type) of the Contacts table

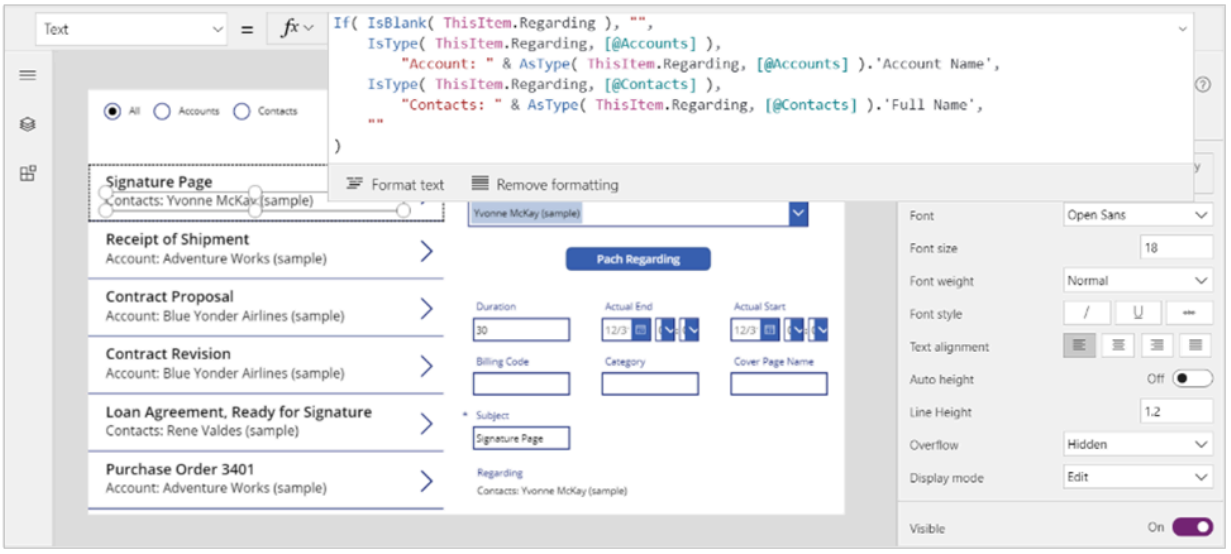
Customer lookup field is another polymorphic lookup that's similar to Owner. You can only have one Owner field per table. But a table can include zero, one, or more Customer lookup fields. The Contacts system table includes the Company Name field, which is a Customer lookup field. Read show the fields of a customer for more details.

Access and set the Regarding field of activity tables such as Faxes, Phone Calls, Email Messages

Polymorphic lookups aren't limited to Accounts and Contacts. The list of tables is extensible with custom tables. For example, the Faxes table has a polymorphic Regarding lookup field, which can refer to Accounts, Contacts, and other tables. If you have a gallery with data source set to Faxes, you can use the following formula to display the name associated with the Regarding lookup field.

```
Power Fx
```

```
If( IsBlank( ThisItem.Regarding ), "",
   IsType( ThisItem.Regarding, [@Accounts] ),
       "Account: " & AsType( ThisItem.Regarding, [@Accounts] ).'Account
Name',
   IsType( ThisItem.Regarding, [@Contacts] ),
       "Contacts: " & AsType( ThisItem.Regarding, [@Contacts] ).'Full Name',
   "" )
```



Read Regarding lookup fields and Regarding relationships for more details.

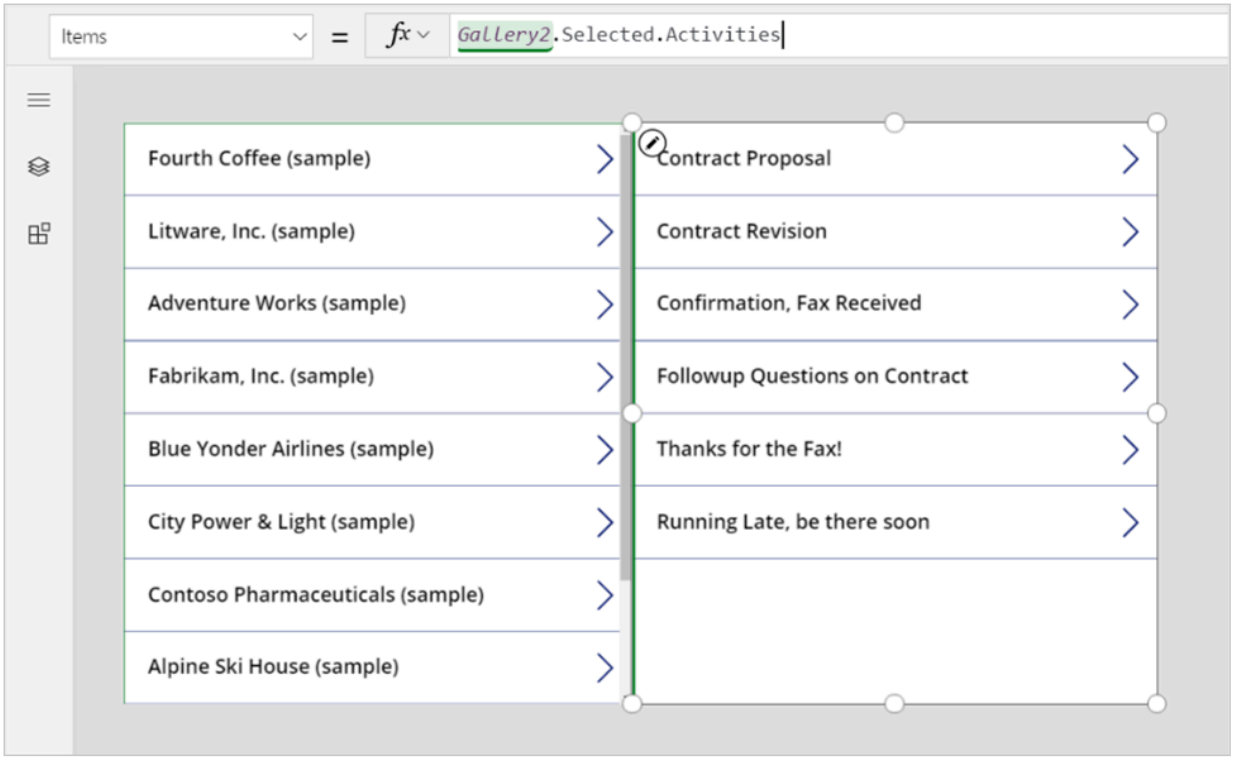
Access the list of all Activities for a record

In Dataverse, tables such as Faxes, Tasks, Emails, Notes, Phone Calls, Letters, and Chats are designated as activities. You can also create your own custom activity tables.

You can show activities of a specific type (such as Faxes or Taxes), or all activities associated with a table such as account. Add the Activities table and other individual tables whose data you plan to display in the canvas app.

Each time you add a record to (for example the Tasks table), a record in the Activity table with the fields common across all activity tables is created. Read activity table for more details.

The following example shows that as you select an Account, all the Activities associated with that account will be displayed:

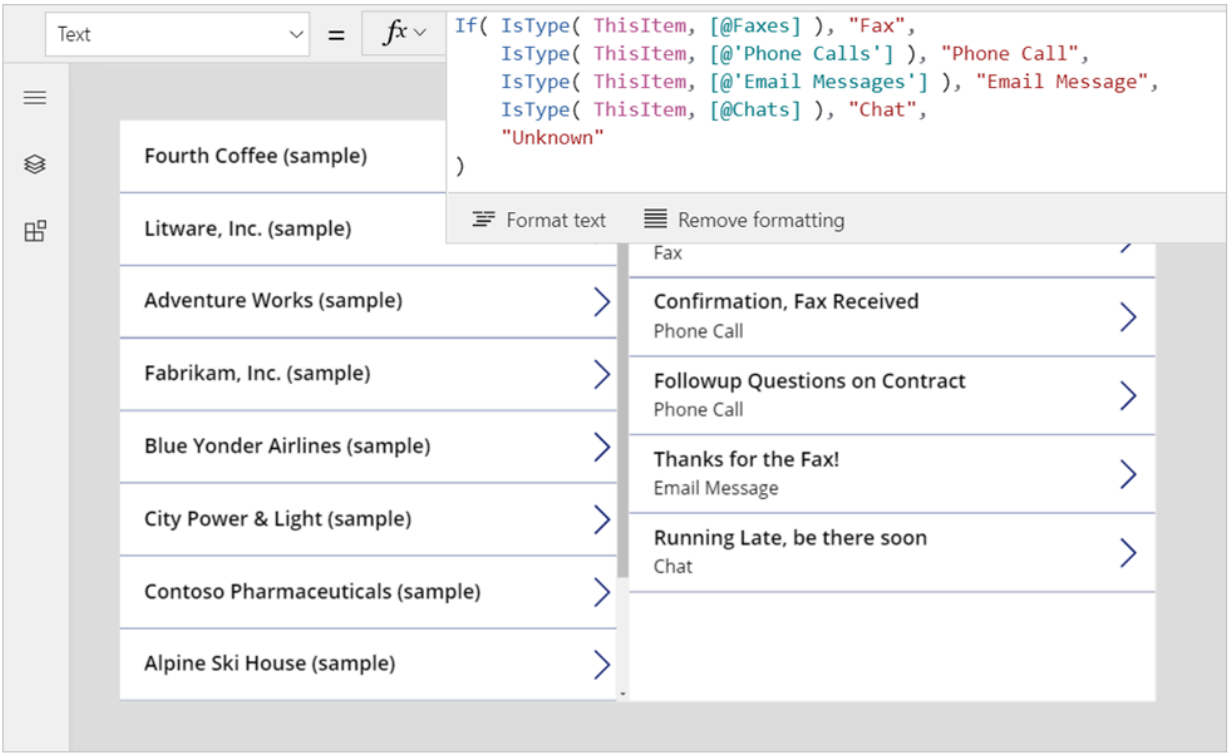


The records are being displayed from the Activity table. But you can still use the IsType function to identify which kind of activity they are. Again, before you use IsType with a table type, you must add the necessary data source.

By using this formula, you can show the record type in a label control within the gallery:

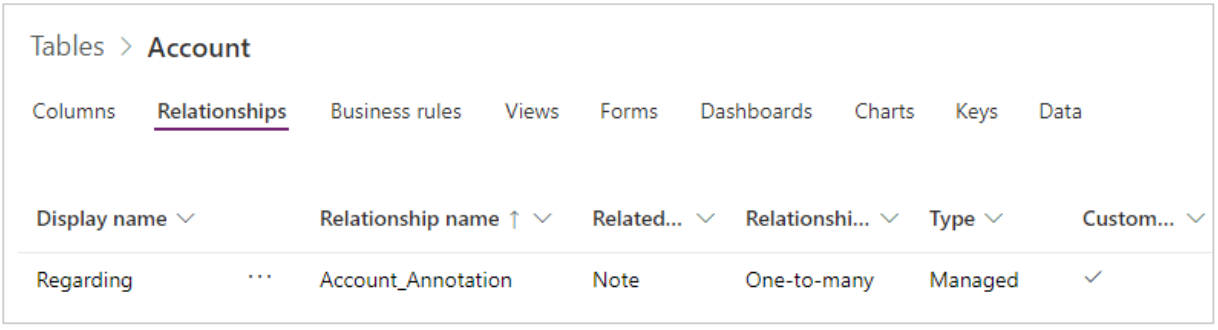
Power Fx

```
If( IsType( ThisItem, [@Faxes] ), "Fax",
   IsType( ThisItem, [@'Phone Calls'] ), "Phone Call",
   IsType( ThisItem, [@'Email Messages'] ), "Email Message",
   IsType( ThisItem, [@Chats] ), "Chat",
   "Unknown")
```



Access the list of Notes for a record

When you create a table, you can enable attachments. If you select the check box for enabling attachments, you create a Regarding relationship with the Notes table, as this graphic shows for the Accounts table:



Filtering

You can't read or filter based on the Regarding field. However, the reverse Notes one-to-many relationship is available. To list all the Notes associated to an Account table, you can use the following formula:

Power Fx

```
First( Accounts ).Notes
```

Patch

You can't set the Notes field on a table by using Patch. To add a record to a table's Notes table, you can use the Relate function. Create the note first, as in this example:

Power Fx

```
Relate( ThisItem.Notes, Patch( Notes, Defaults( Notes ), { Title: "A new
note", isdocument:'Is Document (Notes)'.No } ) )
```

Next steps

- Formula reference
- Controls reference

See also

What is Dataverse?

Feedback

Was this page helpful? Yes No

Provide product feedback

Connect to cloud-storage from Power Apps

Article • 12/16/2022

Power Apps offers several cloud-storage connections. Using any of these connections, you can store an Excel file and use the information in it throughout your app. These connections include:

Box	Dropbox	Google Drive	OneDrive	OneDrive for Business
				

> **Note**
>
> To learn about using Azure Blob Storage with Power Apps, go to **Connect to Azure Blob Storage from Power Apps**.

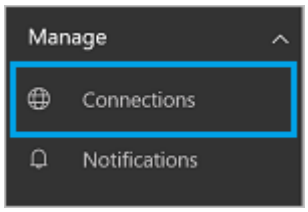
Prerequisites

- Access to Power Apps
- Add the connection
- Create an app from a template, from data, or from scratch
- An Excel file with the data formatted as a table:

    1. Open the Excel file, and then select any cell in the data that you want to use.
    2. On the **Insert** tab, select **Table**.
    3. In the **Save as Table** dialog box, select the **My table has headers** checkbox, and then select **OK**.
    4. Save your changes.

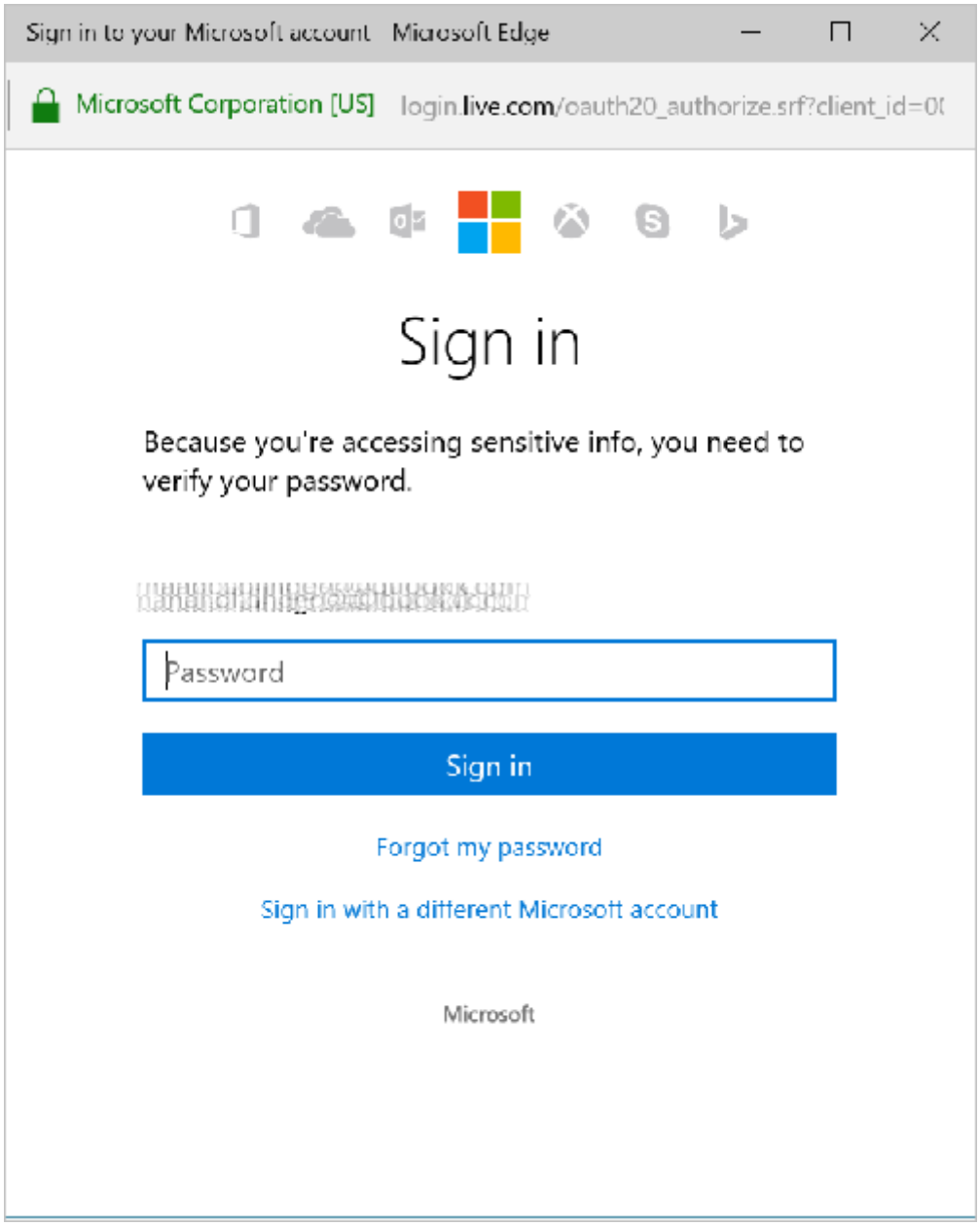
Connect to the cloud storage connection

1. At powerapps.com, expand **Manage**, and select **Connections**:



2. Select **New connection**, and select your cloud storage connection. For example, select **OneDrive**.

3. You are prompted for the user name and password of your cloud storage account. Enter them, and then select **Sign in**:

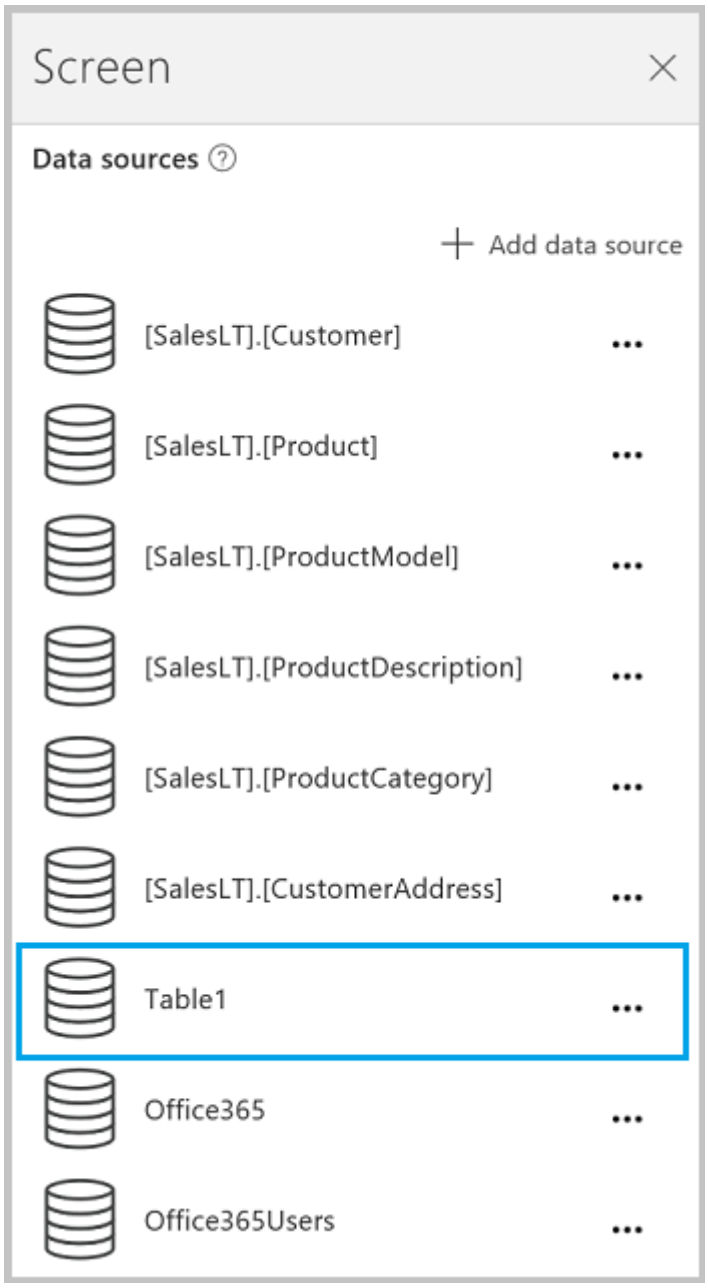


   Once you are signed in, this connection is ready to be used within your apps.

4. In your app, click or tap **Data sources** on the **View** tab of the ribbon. In the right-hand pane, click or tap **Add a data source**, click or tap your cloud-storage connection, and then choose the Excel table.

5. Select **Connect**.

   The table is listed as a data source:



> **Note**
>
> Remember, the Excel data must be formatted as a table.

Using the Excel data in your app

1. On the **Insert** tab, select **Gallery**, and then select a **With text** gallery control.

2. Set the **Items** property of the gallery to your Excel table. For example, if your Excel table is named **Table1**, then set it to Table1:



The gallery is automatically updated with information from your Excel table.

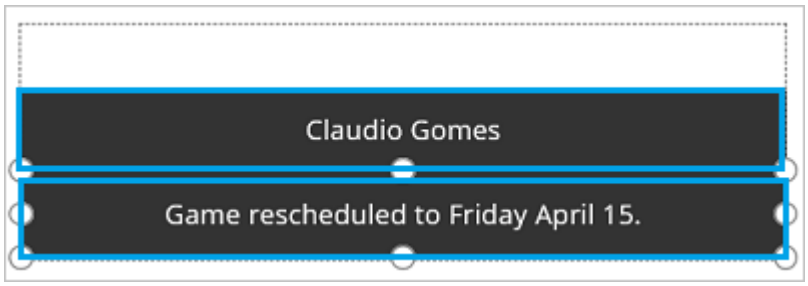
3. In the gallery, select the second or third **Label** control. By default, you see the **Text** property of the second and third labels is automatically set to `ThisItem.something`. You can set these labels to any column in your table.

   In the following example, the second label is set to `ThisItem.Name` and the third label is set to `ThisItem.Notes`:





   Sample output:



> **Note**
>
> The first box is actually an image control. If you don't have an image in your Excel table, then you can delete the image control, and add a label in its place. **Add and configure controls** is a good resource.

Understand tables and records provides more details and some examples.

Sharing your app

You can share your app, your resources such as connectors, and your data with others in your organization.

If you're sharing a folder in Dropbox, the shared folder must be attached to the user's Dropbox account.

There are certain limitations with connectors involving Excel files.

Known limitations

- If **Data type unsupported** or **Not formatted as a table** appears when you try to use an Excel connection in your app, format the data as a table.
- If your Excel data includes a calculated column, you can't use it to build an app, and you can’t add that data to an existing app.
- Power Apps mobile app for Windows platform doesn't support **Dropbox** connector. A pop-up dialog will show the following message in this situation:
  `Upgrade browser to use Dropbox`
  When this happens, consider using web player on Windows platform.

Sharing Excel tables

To share data in an Excel file:

- In OneDrive for Business, share the file itself.
- In OneDrive, share the folder that contains the file, and specify file paths, not URLs, for any media.
- In Dropbox or Google Drive, share either the file or the folder.

Helpful links

See all the available connections.
Learn how to add connections and add a data source to your apps.
Understand tables and records with tabular data sources.
Some additional gallery resources include Show a list of items and Show images and text in a gallery.

Connect to Azure Blob Storage from Power Apps

Article • 12/16/2022

Power Apps can connect to Azure Blob Storage. You can upload files such as Word, Excel, or multimedia images, audio or video using the Azure Blob Storage connector for Power Apps.

When you design a canvas app that connects to Azure Blob Storage, the app uses the blob storage account name and key to connect. After you share the app with others, users can use the connection configured inside the app to upload files to Azure Blob Storage without the need to share blob storage name and keys with the app users.

In this article, you'll learn how to create a sample canvas app that connects to Azure Blob Storage, and add controls to the app that allow you to upload different types of files to the connected blob storage.

> ⓘ **Note**
>
> To learn more about other types of cloud storage options with Power Apps (such as OneDrive, OneDrive for Business, Google Drive, Dropbox, or Box), go to **Connect to cloud storage from Power Apps**.

Prerequisites

Before you begin, create and configure a BlockBlobStorage account. You can also use legacy BlobStorage account, though not recommended. More information: Types of storage accounts in Azure Blob Storage

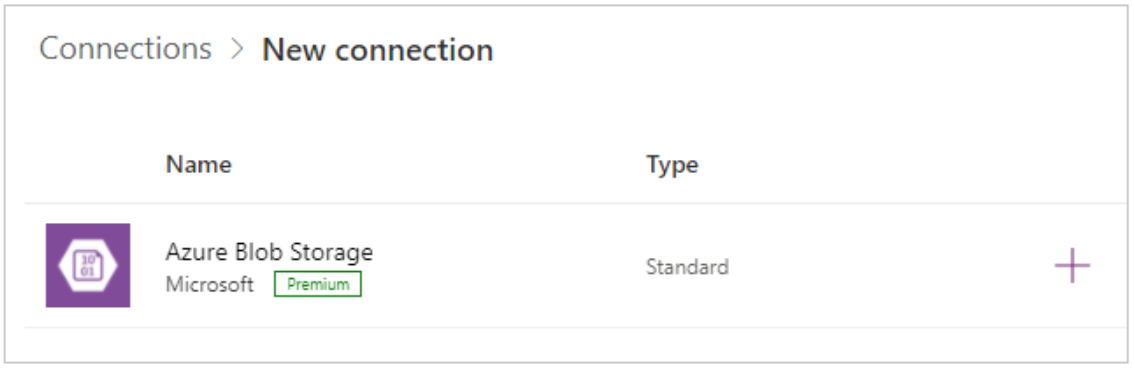
Create Azure Blob Storage connection

Power Apps requires a connection to Azure Blob Storage to be created for the app to connect to the storage.

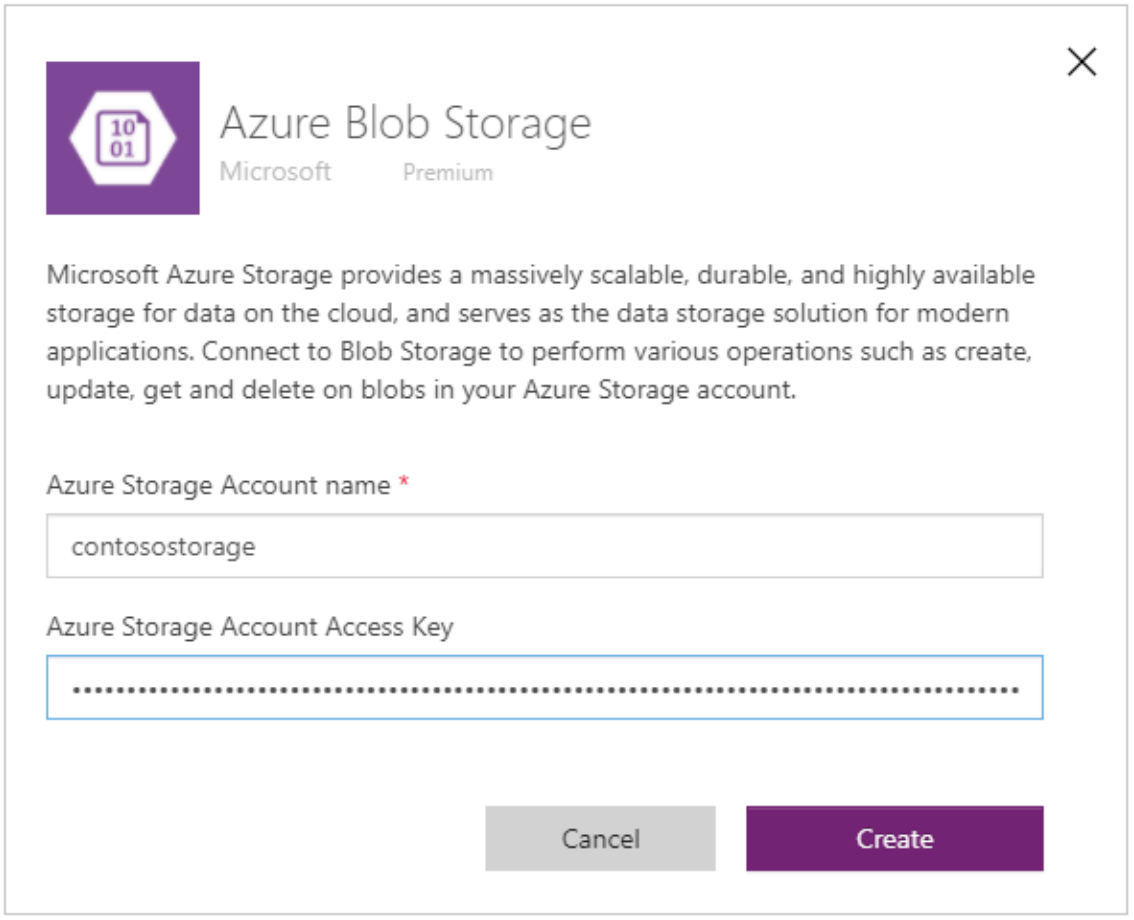
To create the Azure Blob Storage connection:

1. Sign in to Power Apps.
2. On the left-pane, expand **Data**.

3. Select **Connections**.
4. Select **New connection**.
5. Select **Azure Blob Storage**.



6. Copy and paste the account name, and access key.



   For more information about how to copy account name and access key, go to View account access keys in Azure.

7. Select **Create**.

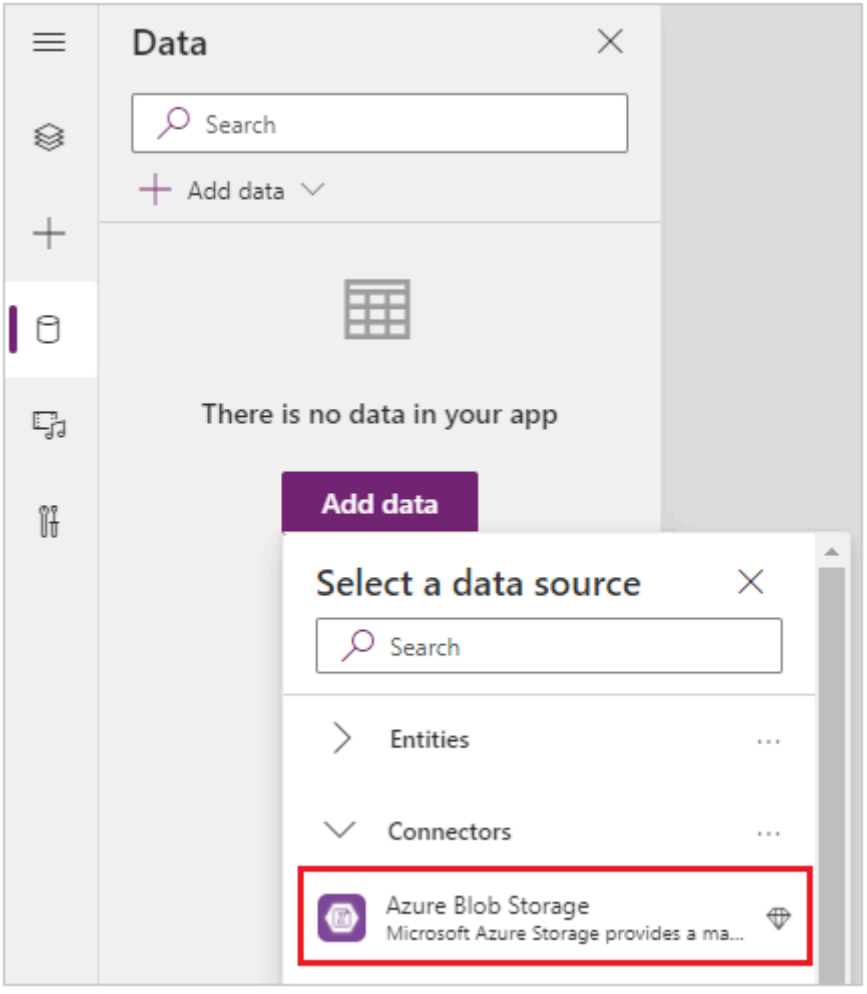
Your connection to the Azure Blob Storage is now configured and ready to use with canvas apps.

Create canvas app with Azure Blob Storage connection

Now that you have the connection with Azure Blob Storage created, let's create a canvas app that connects to this storage.

> **Note**
>
> In this section, you'll create a sample app with sample controls, functionality and layout design. Depending on your business requirement, you can create the app with a different structure, or customize differently.

1. Create a blank canvas app with the name "Sample app for Azure Blob Storage" and **Phone** layout.
2. Inside Power Apps Studio, on the left-pane, select .
3. Select **Add data**.
4. From the list of connectors, select **Azure Blob Storage**.

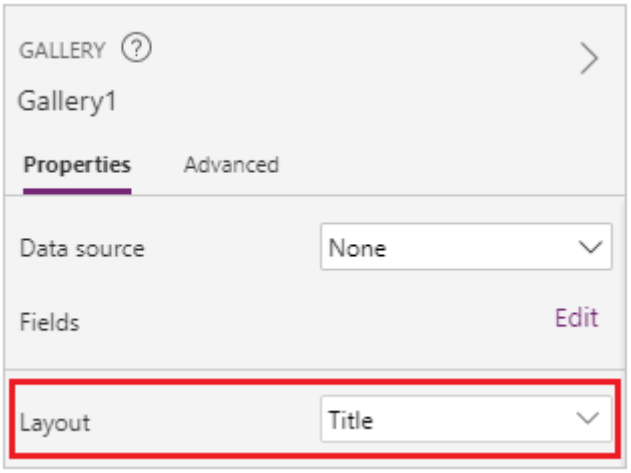


View containers and files

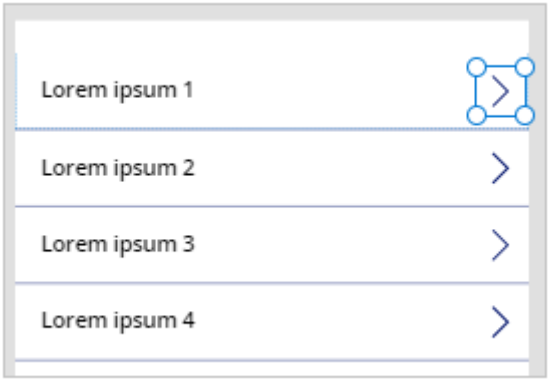
Now that you have the app connected to Azure Blob Storage, let's add galleries to see containers and files within the containers from the connected storage.

1. Select **Insert** -> **Gallery** -> **Blank vertical**.

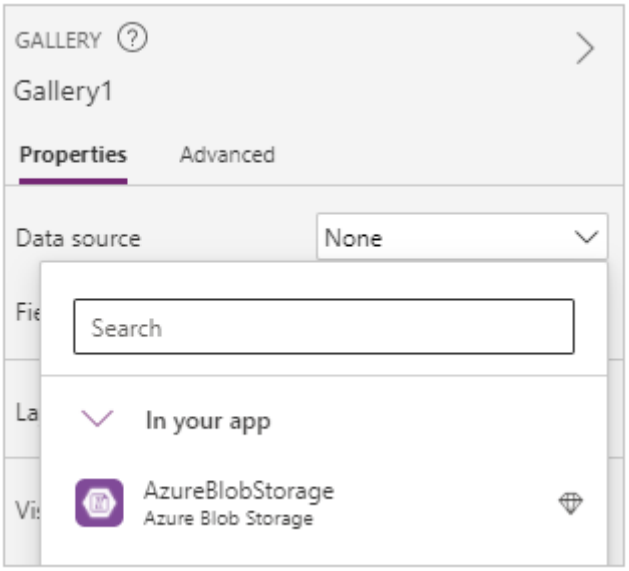
2. From the right-side of the screen, on the property pane, select the layout drop-down and choose **Title**.



3. Select first > inside the gallery, and delete it.

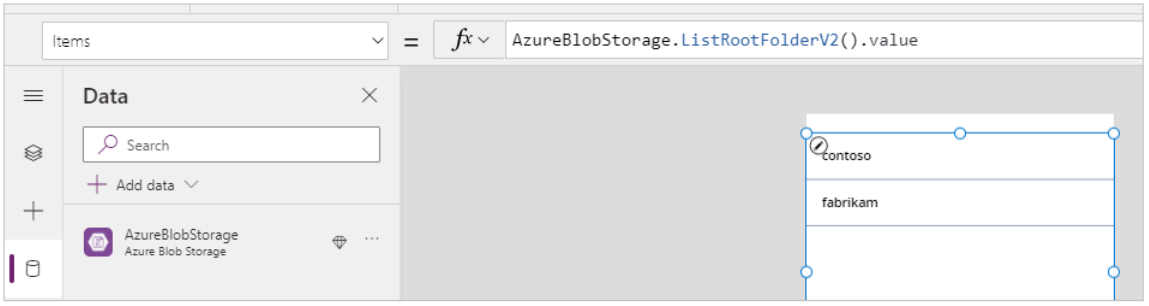


4. From the right-side of the screen, on the property pane, select the drop-down for data source, and choose **Azure Blob Storage**.



5. Set the **Items** property of the gallery to:

```
AzureBlobStorage.ListRootFolderV2().value
```



This operation lists blobs in the Azure Blob Storage root folder. More information: List blobs in root folder

6. Select **Insert** -> **Gallery** -> **Blank vertical** to add another blank vertical gallery.

7. Move the gallery below the gallery you added earlier that shows the list of containers.

8. From the right-side of the screen, on the property pane, select the layout drop-down and choose **Title, subtitle, and body**.

9. Select first > inside the gallery, and delete it.

10. From the right-side of the screen, on the property pane, select the drop-down for data source, and choose **Azure Blob Storage**.

11. Set the **Items** property of the gallery to:

Power Fx

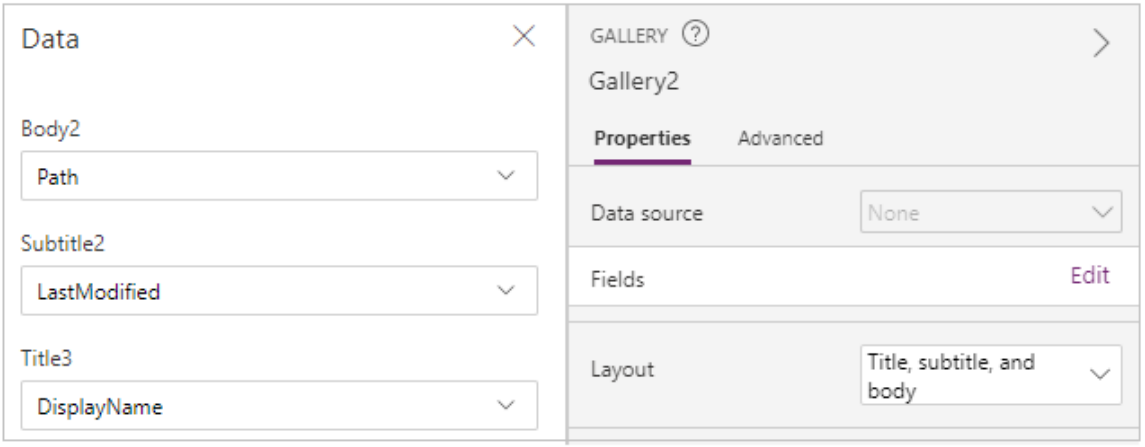
```
AzureBlobStorage.ListFolderV2(Gallery1.Selected.Id).value
```

This operation lists blobs in a container. More information: List blobs

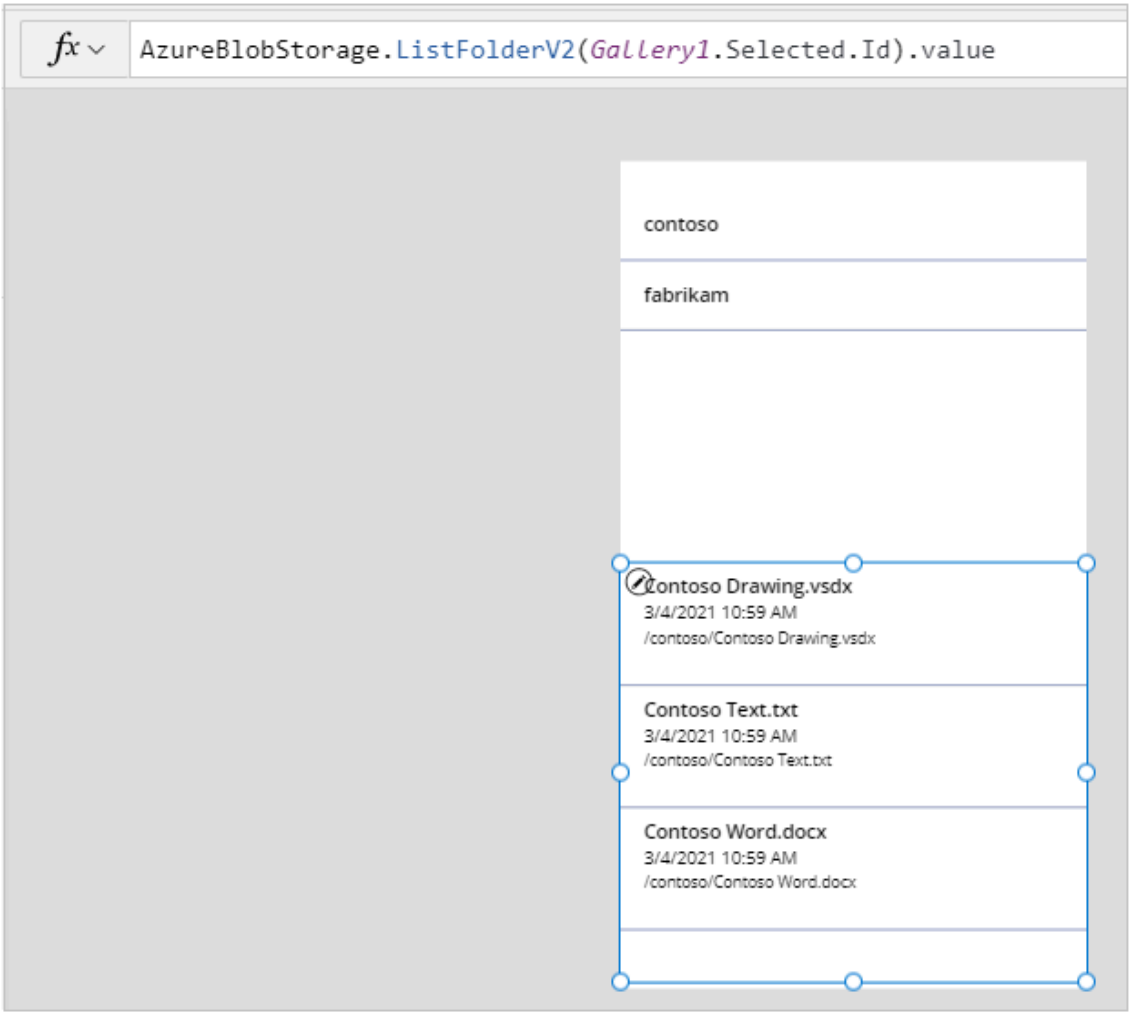
> **Note**
>
> *Gallery1* in this formula is the reference to the gallery added earlier that lists all containers in the storage account. Update the formula with the gallery name if different.

12. From the right-side of the screen, on the property pane, select **Edit** for **Fields**.

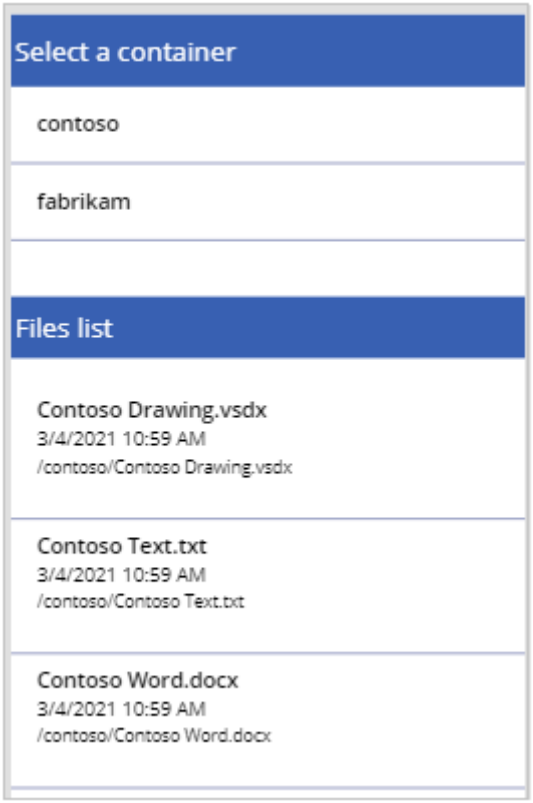
13. Change the selected fields for the gallery title as **DisplayName**, subtitle as **LastModified**, and body as **Path**.



The gallery now shows the list of files from the container selected using the gallery on the top.



14. Select **Insert** -> **Text label**.
15. Place the label on top of the app screen.
16. Set the label's **Text** property as "Select a container".
17. Use the properties pane on the right-side of the screen and choose the label text color, size, and label text background color of your choice.
18. Select **Insert** -> **Text label**.
19. Place the label above the gallery with the list of files.
20. Set the label's **Text** property as "Files list".



Upload files to Azure Blob Storage

With the app design so far, you're able to select a container and then list the files from the container.

Let's configure the app with controls and logic to allow upload of files to the connected Azure Blob Storage.

1. Select **Insert** -> **Media** -> **Add picture** to add the ability to select files to upload.
2. Resize the **Add picture** control and place it on the bottom-left of the app screen.
3. Set the **Text** property of the control to "Select a file to upload".
4. Select **Insert** -> **Button**.
5. Place the button on bottom-right side of the app screen.
6. Set the **Text** property of the button to "Upload".
7. Select **Insert** -> **Text input**.
8. Place the text input control above the **Upload** button.
9. Set the **Default** property of the button to "Enter File Name".
10. Set the **OnSelect** property of the button to:

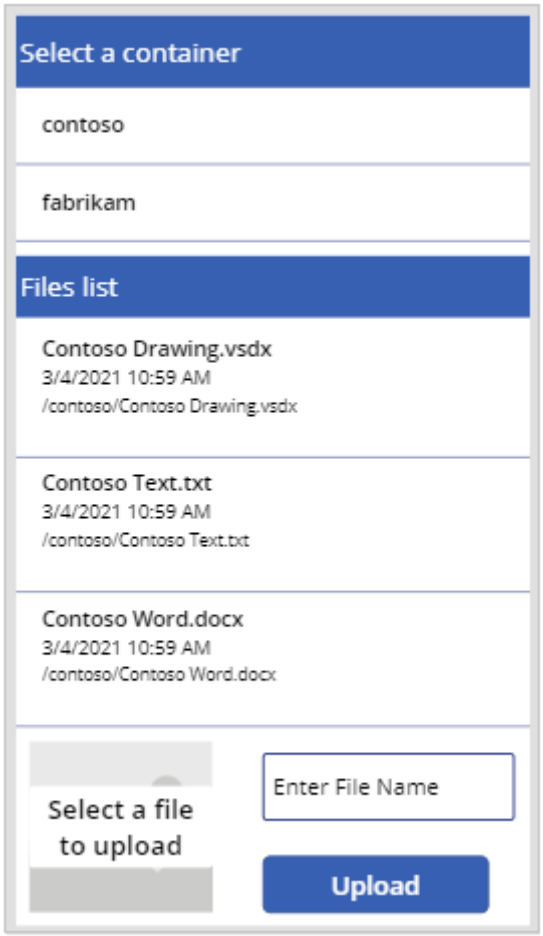
Power Fx

```
AzureBlobStorage.CreateFile(Gallery1.Selected.Name,TextInput1.Text,
UploadedImage1.Image)
```

This operation uploads a blob to Azure Blob Storage. More information: Create blob

> **Note**
>
> *Gallery1* in this formula is the reference to the gallery added earlier that lists all containers in the storage account. The file will be uploaded to the selected container in gallery 1. *TextInput1* and *uploadImage1* reference the text input and upload image controls. Update the formula with the control names if different.

The app controls look like this in the sample app now.



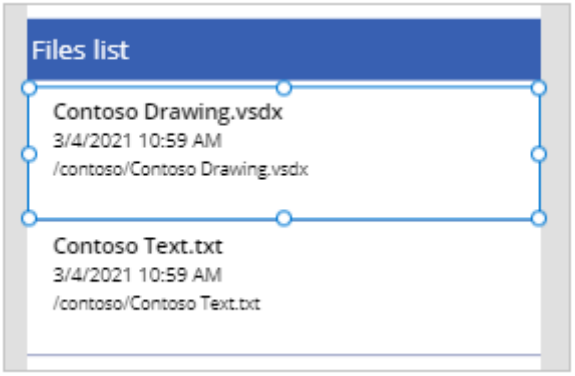
> **Tip**

Ensure you select **All files** when using the upload option to ensure all file types are visible in the file explorer dialog box.

Download files from Azure Blob Storage

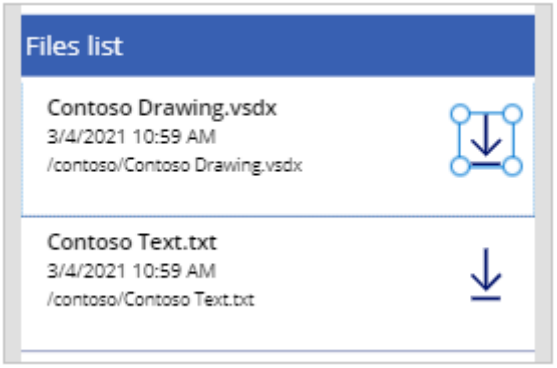
So far you've added the ability to view containers, files from the selected container, and the option to upload files to the storage. Now, let's understand how to work with the download capability with the connected storage.

1. Select the first row in the gallery with the list of files from a container.



2. Select **Insert** -> **Icons** -> **Download**. This adds the download icon for all rows in the gallery.

3. Move the first download icon towards the right side inside the gallery on the app screen. This also moves the rest of the icons for the next rows in the gallery.



4. Set the **OnSelect** property of the download icon to:

```
Power Fx

Launch(AzureBlobStorage.CreateShareLinkByPath(ThisItem.Path).WebUrl)
```

This operation creates a SAS link for a blob using the path. More information: Create SAS URI by path

> **Important**
>
> SAS URI created using **CreateShareLinkByPath** has a default expiry of 24 hours. If you have a business requirement to expire the URI in a shorter or different time, consider updates to this formula. For example, the below sample expires the URI in 1 hour using Now() and DateAdd() functions.

Power Fx

```
Launch(AzureBlobStorage.CreateShareLinkByPath(ThisItem.Path,
{ExpiryTime:DateAdd( Now(),1)}).WebUrl)
```

> **Tip**
>
> For more information about configuring Azure Blob Storage for public anonymous access, and different public access levels, go to Configure anonymous public read access for containers and blobs.

The app now has the ability to allow you to download the files.

Test, save, publish and share the app

Play the app to test, and verify the app works as expected. After the testing, ensure you save and publish the app before you close Power Apps Studio. And then, you can share the app with others within your organization, or guests outside your organization.

Optional customizations

In this section, you'll learn about optional and additional customizations that you can consider for your app.

Media type

You can use **Media type**, or **Path** fields for the gallery to optionally display the image content in the respective controls. For example, PDF viewer for PDF files, Image for images, or Audio/video for audio/video files.

For example, to filter the files with the file extension type of **.pdf**, use the following sample formula.

```
Power Fx
```

```
If(".pdf" in Gallery2.Selected.Path,
AzureBlobStorage.GetFileContent(Gallery2.Selected.Id))
```

Similarly, you can use different file extension types, or media types to additionally customize added controls.

Refresh galleries connected to Azure Blob Storage

The Azure Blob Storage connection doesn't refresh data inside the galleries automatically when the data is updated. If you have more than one container, you can select the other container and then, select the previously selected container back to refresh the connected gallery to display the changes.

Another method that can be considered is to use a collection for the first gallery, and then, use the function ClearCollect to refresh the collection.

For example, the following formulas allow you to update the collection for the top containers list in the first gallery, and update the second gallery when the upload button is selected, or when the screen appears (OnVisible screen property).

1. Set the **Items** property of the first gallery for the list of containers to "TopLevelList".
2. Append to the upload button **OnSelect** property:

   ```
   Power Fx
   ```

   ```
   ClearCollect(TopLevelList, AzureBlobStorage.ListRootFolderV2().value)
   ```

3. Add to the screen **OnVisible** property:

   ```
   Power Fx
   ```

   ```
   ClearCollect(TopLevelList, AzureBlobStorage.ListRootFolderV2().value)
   ```

Limitations

You can't use Microsoft Excel as the data source when the file is stored in Azure Blob Storage. To use Excel as the data source, use the other cloud storage connectors (such

as OneDrive, OneDrive for Business, Google Drive, Dropbox, or Box). More information: Connect to cloud storage from Power Apps

Next steps

Design the app interface

See also

Connect to cloud storage from Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Connect to Azure DevOps from Power Apps

06/21/2025

The Power Apps connector for Azure DevOps lets you work with your Azure DevOps instance. View Azure DevOps queries, select work items by type, and view or edit details—all from inside a canvas app connected to Azure DevOps.

> **Tip**
>
> For a complete list of all actions, see **Azure DevOps connector actions**.

This article guides you through building a canvas app that connects to Azure DevOps to get a list of queries and interact with work items in your project.

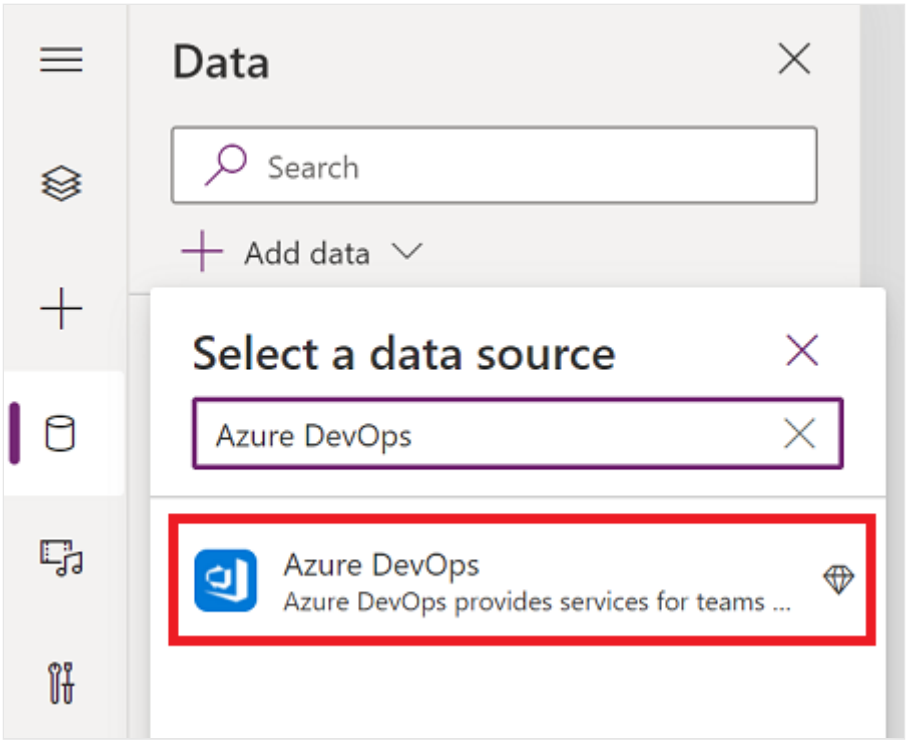
Prerequisites

You need the following:

- A Power Apps license. If you don't have one, use a 30-day trial, or sign up for a developer plan for non-production use.
- If you're new to Power Apps, learn the basics by generating an app, then customize the app's controls, gallery, forms, and cards.
- A blank canvas app to connect to Azure DevOps.
- To create the app in this article, you need an Azure DevOps instance with an organization, a project, and a shared query that has a few sample work items available for editing.
- The Azure DevOps instance must let **Third-party application access via OAuth**. For more information, see Manage access policies for Azure DevOps.

Step 1 - Add Azure DevOps data source

To connect to Azure DevOps, edit the blank canvas app, and add the **Azure DevOps** data source.



If you don't have an Azure DevOps connection, select **Connect**, follow the prompts to enter your details, and allow the app to connect.

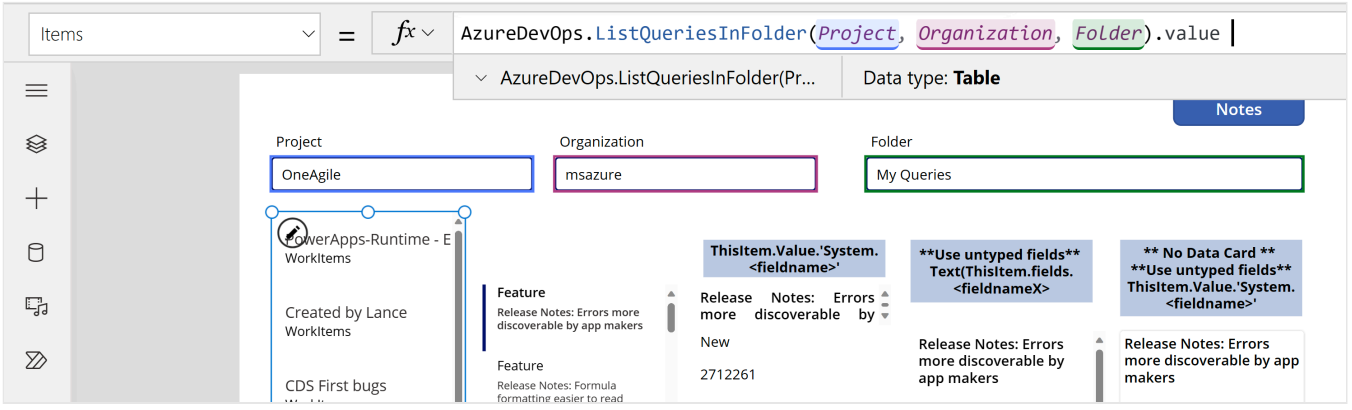
Step 2 - List shared queries

In this section, you use the ListQueriesInFolder action for the Azure DevOps connector to list the available queries.

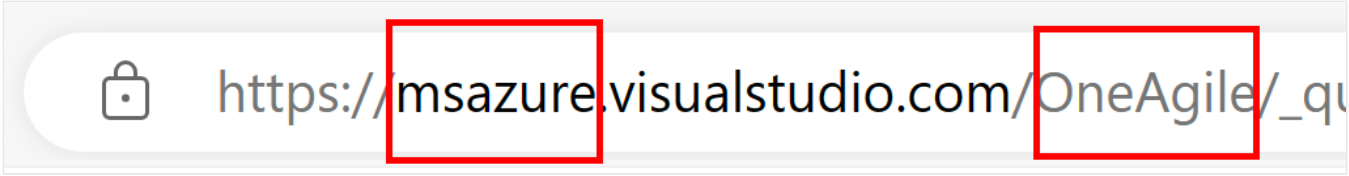
1. In the left pane, select **Insert** > **Layout** > **Blank vertical gallery**.

2. Enter the following formula for the **Items** property of the gallery. Replace the example parameter values with your own values.

```
Power Fx

AzureDevOps.ListQueriesInFolder("Project", "Organization", "Folder").value
```



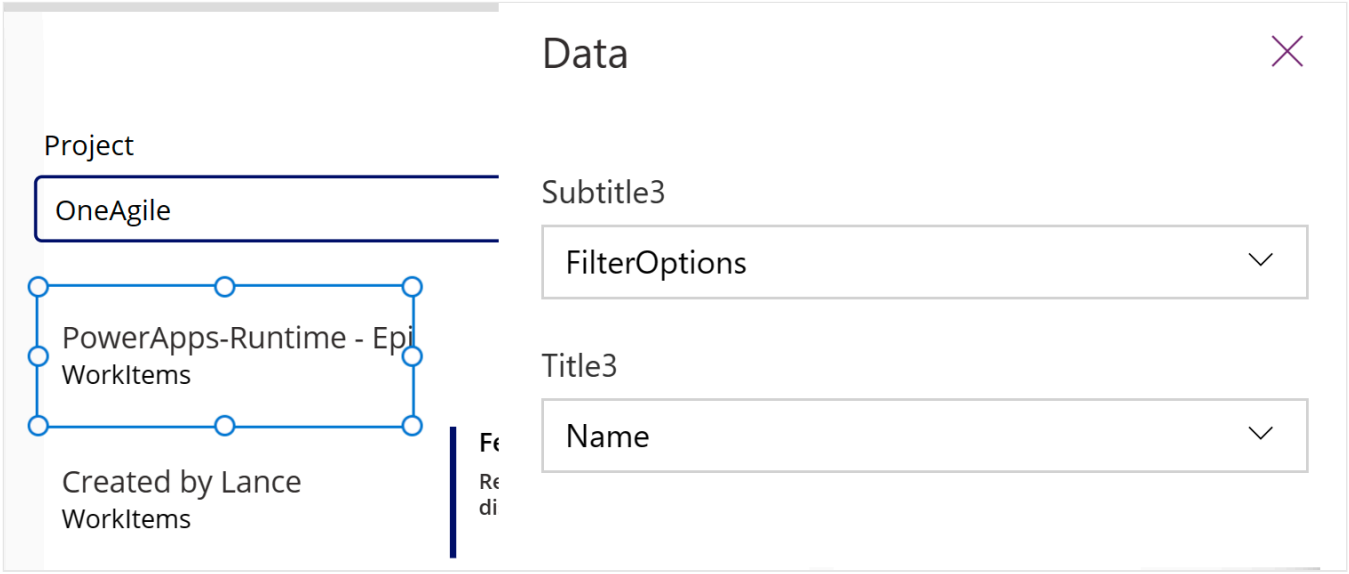
The example uses the variables "Project", "Organization", and "Folder". The actual values are in the text boxes below the formula bar (highlighted). You find your Project and Organization values in the URL used to connect to Azure DevOps. The Folder is usually "Shared Queries" or "My Queries".



If you get the following error in the above formula, enable third-party app access using OAuth in your Azure DevOps organization, and try again.

"AzureDevOps.ListQueriesInFolder failed:{"status":401,"message":"TF400813:The user 'GUID' isn't authorized to access this resource."}"

1. Set the **Layout** for the gallery to **Title and subtitle**.
2. Choose the fields appropriate for Azure DevOps as **Name** and **FolderOptions** for the title and subtitles.



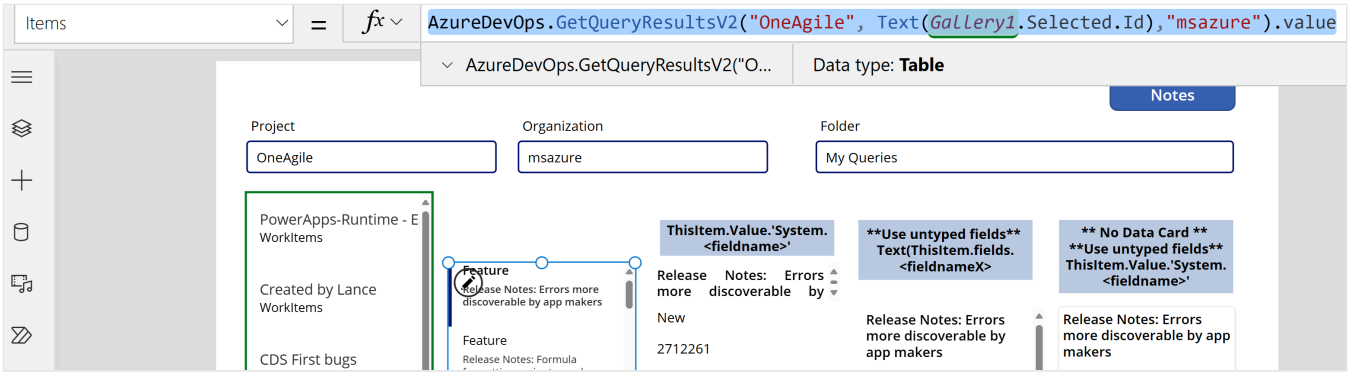
Step 3 - List work items

Use the GetQueryResultsV2 action for the Azure DevOps connector to list all work items for the selected query. This action binds the gallery to the data source.

1. Insert another blank vertical gallery and place it next to the existing gallery.
2. Enter the following formula for the **Items** property of the gallery. Replace the example parameter values with your project and organization names.

Power Fx

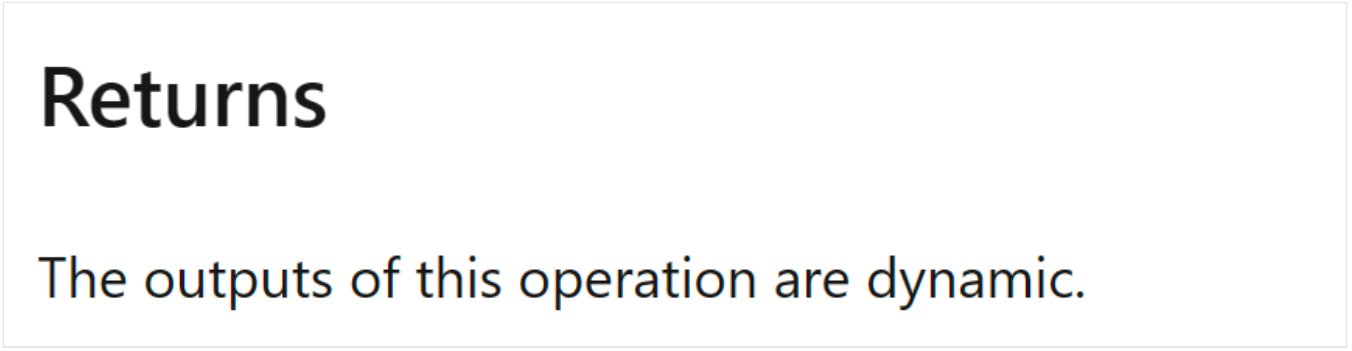
```
AzureDevOps.GetQueryResultsV2("Project", Text(Gallery1.Selected.Id),
"Organization").value
```



This formula uses the GetQueryResultsV2 action with the project name, query ID, and organization name. The query ID in this example (`Gallery2.Selected.Id`) refers to the query selected from the list of queries available through the gallery added earlier. Replace the gallery name as needed.

Add dynamic return values to your gallery

The result of **GetQueryResultsV2** is dynamic, so the values are dynamic as well.



But you can access some of the values. Azure DevOps returns a basic set of values for all typed items. Select the data card in the gallery and insert two text labels. Set the text property of the labels as follows:

Power Fx

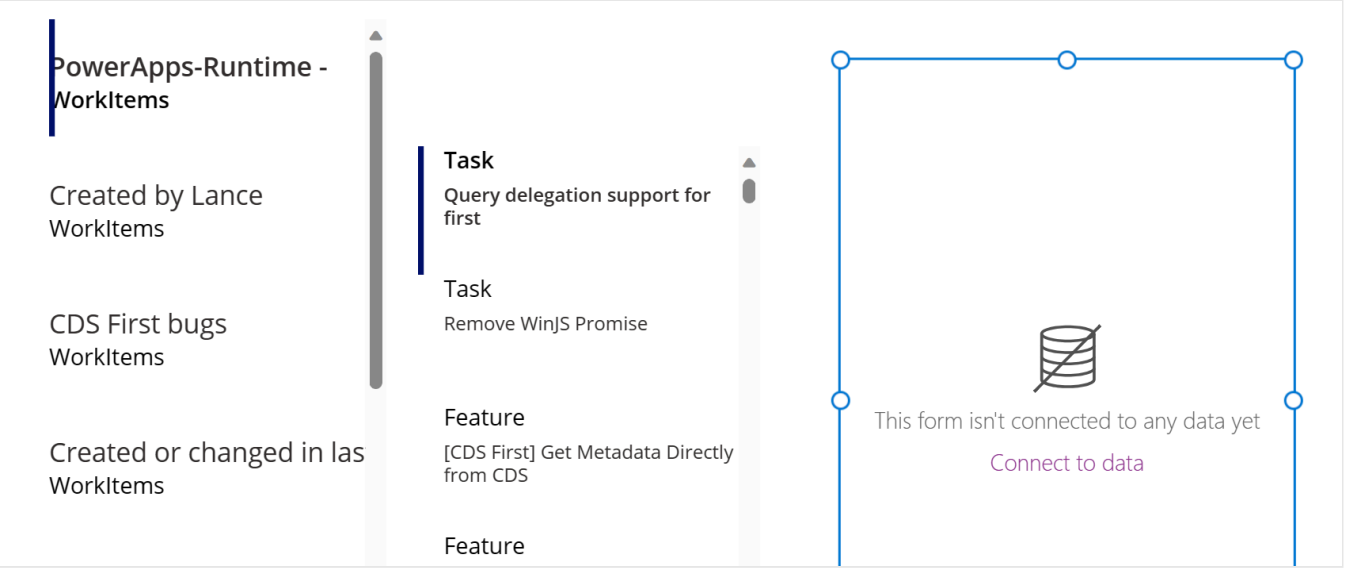
```
ThisItem.Value.'System.WorkItemType'
ThisItem.Value.'System.Title'
```

Step 4 - Display work items

The app shows a list of all queries and the list of work items for the selected query. Now, add an edit form to display data.

1. Move both galleries to the left side of the screen to make room for the edit form.

2. Add **Edit form** to the screen and move it to the right side of the galleries.

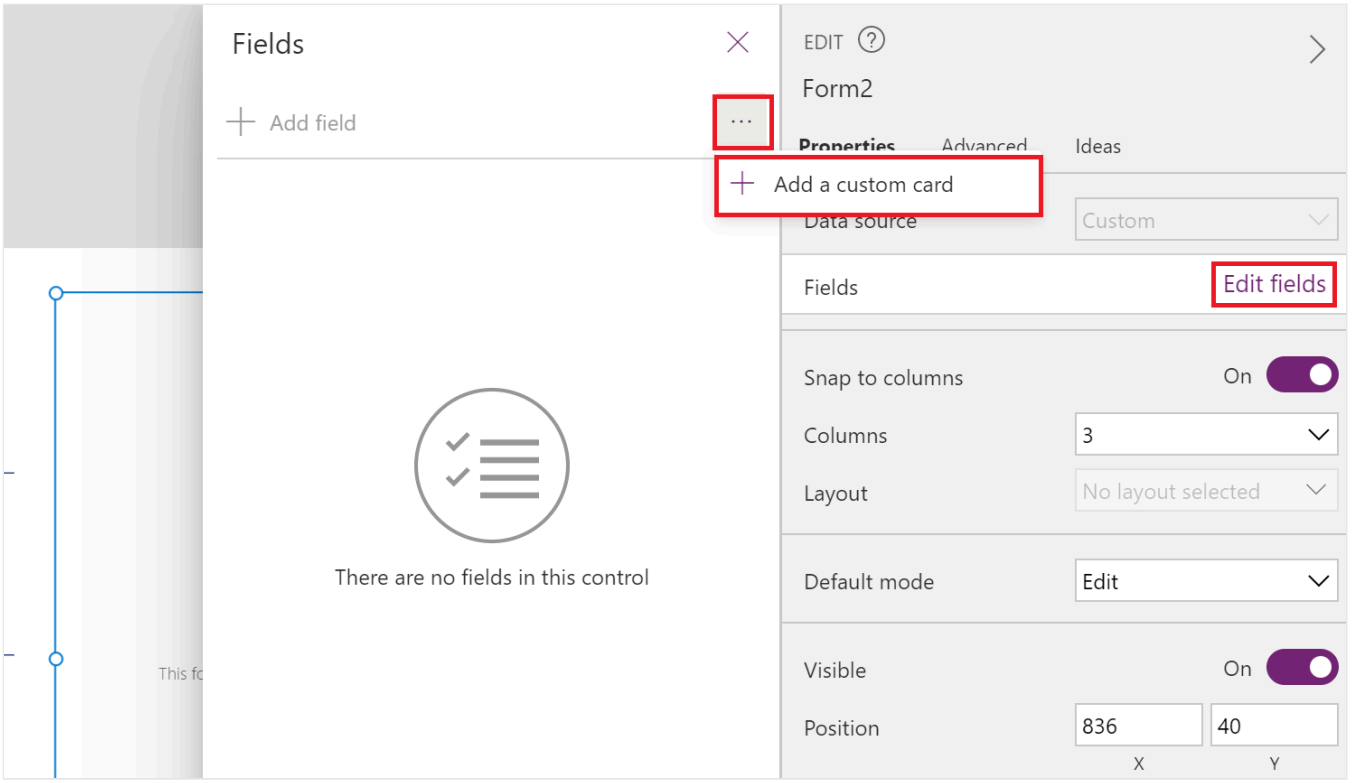


1. Set the **DataSource** property of the edit form to `AzureDevOps.GetQueryResultsV2("Project", Text(Gallery1.Selected.Id),"Organization").value`. Replace "Project" and "Organization" with your project and organization names.

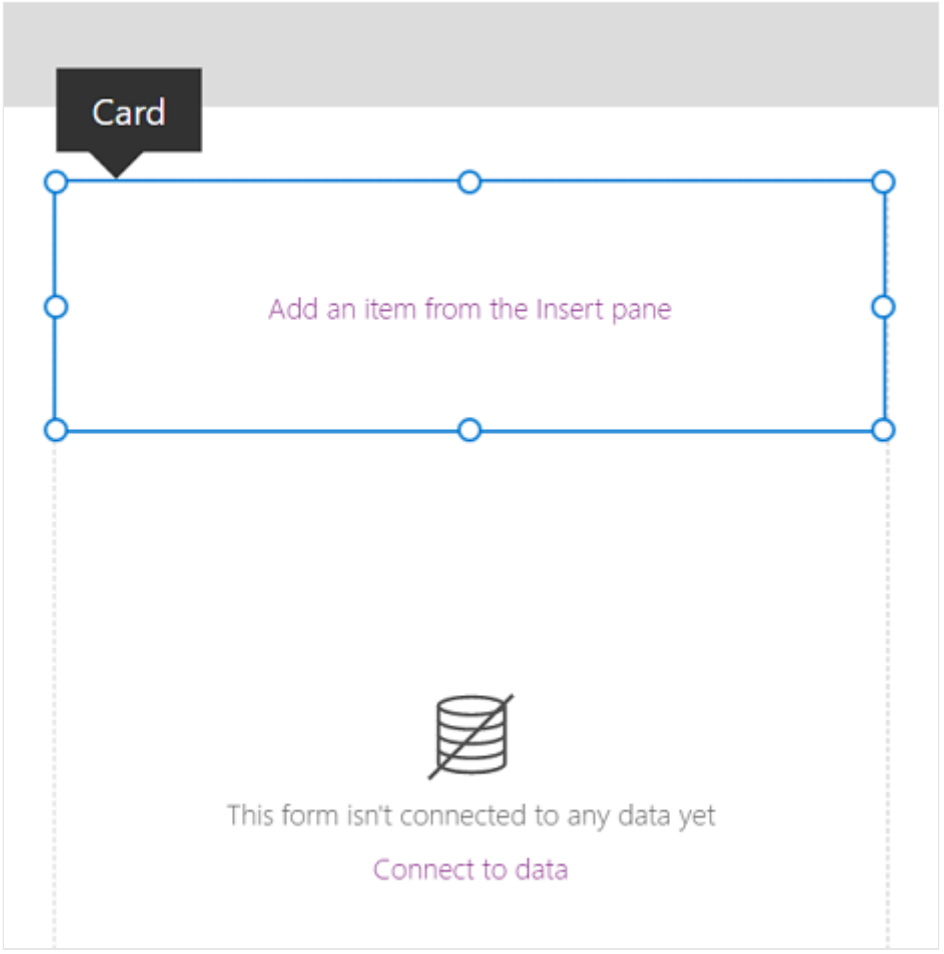
2. Set the **Item** property of the edit form to `Gallery2.Selected`.

This formula sets the **Item** property for the edit form to the work item that is selected.

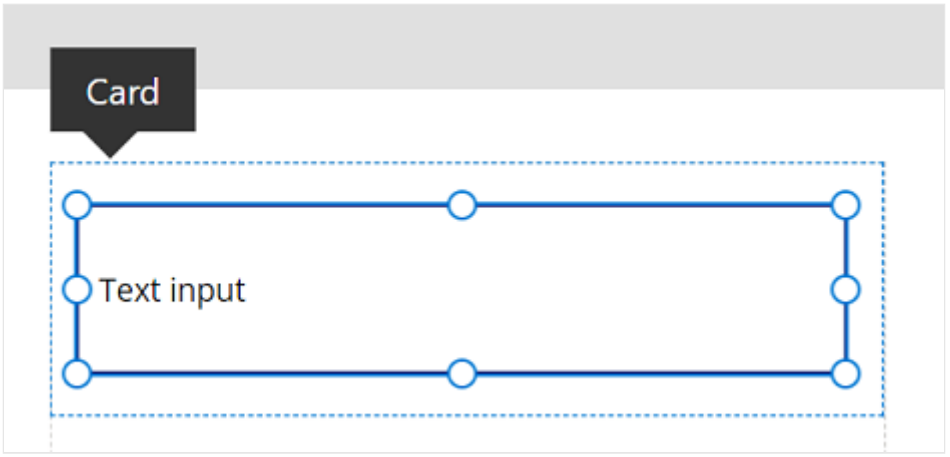
1. Select **...** (ellipsis) > **Add a custom card**.



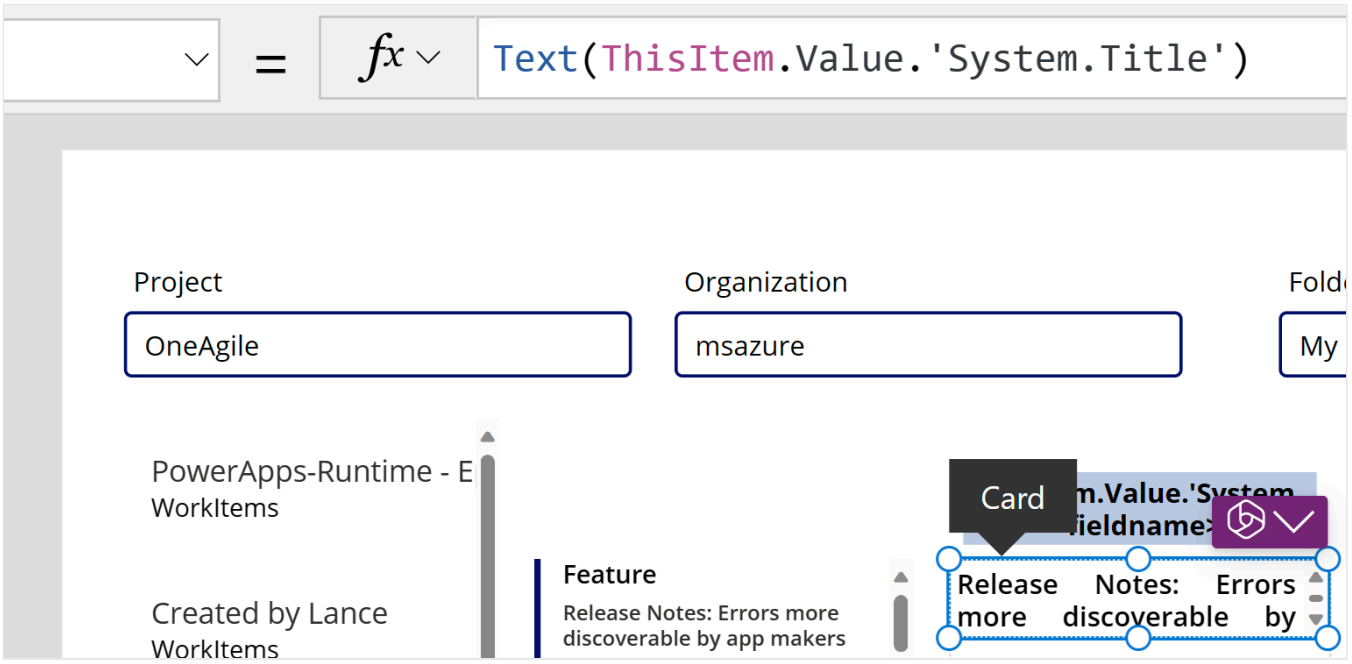
1. Move the data card to the top of the edit form.



1. Increase the size of the text input control.



1. Set the **Default** property of the text input control to `Text(ThisItem.Value.'System.Title')`. The Text function returns the value as text.



This formula sets the default text inside the text input control to the **Title** field from the selected Azure DevOps work item.

> **Tip**
>
> If your Azure DevOps project uses the **Description** field with HTML or rich text, use the Rich text editor input control instead of the Text input or label controls. The **Rich text editor** control helps display the description as rich text instead of HTML code.

1. Repeat the previous steps to add another custom card, with a text input control inside with the **Default** property set to `Text(ThisItem.Value.'System.State')`.

This formula sets the default text inside the text input control to the **State** field from the selected Azure DevOps work item.

1. Rearrange the data cards inside the edit form to create space for the save icon.

Add dynamic return values to your forms

So far, you use the Edit form, which simplifies data access by providing a common DataSource and Item property for all data cards in the form. To access dynamic values, set **both** the data source and the item properties as follows (replace Organization and Project with your values):

```
Power Fx

AzureDevOps.GetWorkItemDetails(Gallery2.Selected.Value.'System.Id',Organization,
Project, Gallery2.Selected.Value.'System.WorkItemType')
```

When you pass the text property "WorkItemType", such as "Feature", you can switch between items like Features and Work Items. Because the set of fields for these items differs, the return type from this call is dynamic.

Access specific values using the common method Text(ThisItem.Value.'System.Id'). Alternatively, access them through the more general dynamic response using Text(ThisItem.fields.System_Id). These dynamic value names aren't typically documented. To find the correct names for these fields, including non-standard fields, open the monitor tool and examine the data response for the GetWorkItemDetails call. See the image below for guidance.

If you aren't using an Edit form but are using a container, retrieve these values with a formula like the one below, which gets information from a custom team field.

```
Power Fx

Text(AzureDevOps.GetWorkItemDetails(Gallery2.Selected.Value.'System.Id',Organizati
on, Project,
Gallery2.Selected.Value.'System.WorkItemType').fields.One_custom_CustomField1)
```

GetWorkItemDetails

Details Formula Request Response


```
"propertyName": "Item",
"nodeId": 11,
"formulaData": {…
},
"data": {
  },
  "response": {
    "duration": 362.8,
    "body": {
      "id": 1770396,
      "rev": 34,
      "url": "https://dev.azure.com/msazure/_apis/wit/workItems/1770396",
      "fields": {
        "System_Id": 1770396,
        "System_AreaId": 197452,
        "System_AreaPath": "OneAgile\\PowerApps\\Runtime",
        "System_TeamProject": "OneAgile",
        "System_NodeName": "Runtime",
        "System_AreaLevel1": "OneAgile",
        "System_AreaLevel2": "PowerApps",
        "System_AreaLevel3": "Runtime",
        "System_Rev": 34,
        "System_AuthorizedDate": "2020-05-15T16:36:14.233Z",
        "System_RevisedDate": "9999-01-01T00:00:00Z",
        "System_IterationId": 232968,
        "System_IterationPath": "OneAgile\\PowerApps\\MakerX\\2020\\20.8",
        "System_IterationLevel1": "OneAgile",
        "System_IterationLevel2": "PowerApps",
```


Updating values in Azure DevOps

To update a value in Azure DevOps, use the `UpdateWorkItem` function in the `OnSelect` property of a button.

Power Fx

```
AzureDevOps.UpdateWorkItem(
    Gallery2.Selected.Value.'System.Id',
    Organization,
    {
        description: "This is a new description",
        dynamicFields: ParseJSON(JSON({'Custom Field 1': "This is new custom text"
}))
    }
)
```

The formula adds new sample text, but you can also use a Power Fx expression.

Make sure the formula uses lowercase for built-in field names. For example, when you refer to the Description field, use `description: "This is a new description"` instead of `Description: "This is a new description"`. Incorrect casing can result in the error "400 Required parameter missing for requested operation: 'UpdateWorkItem'". For custom or dynamic values, use the display name as shown in the UI, such as 'Custom Field 1'. This naming convention is specific to Azure DevOps and might differ from other services.

Next steps

Run the app. Select a query from the list of queries. Then, choose a work item you want to update the title or description of. Make a change, and then select the save button. The app saves your changes to the Azure DevOps work item. Switch to another query, and then switch back to see the changes in the app.

You can also customize the app further or create an app with more data cards on forms. Use a display form instead of an edit form to show data in different data cards. When you use a display form, use the Text label control to show text. When you use rich text or HTML format (like the **Description** field in Azure DevOps), use the HTML text control. For more information about customizing the app, see controls, gallery, forms, and cards.

See also

Working with dynamic schema data sources in Power Apps (experimental)

Connect to Dynamics 365 from Power Apps

Article • 12/16/2022

If you want to connect to Dynamics 365 Business Central, Dynamics 365 Customer Insights, or Dynamics 365 Finance and Operations apps, choose the relevant connector.

> **ⓘ Important**
>
> The Dynamics 365 connector is deprecated. For more information, see **Dynamics 365 connector deprecation announcement**.
>
> - Use the **Microsoft Dataverse connector** when connecting to the Microsoft Dataverse used for Microsoft Dynamics 365 Sales, Microsoft Dynamics 365 Customer Service, Microsoft Dynamics 365 Field Service, Microsoft Dynamics 365 Marketing, and Microsoft Dynamics 365 Project Service Automation.
> - To convert your app that uses the deprecated Dynamics 365 connector, go to **Converting canvas apps with the Dynamics 365 connector**.

See also

Connecting to Your Business Central Data to Build a Business App Using Power Apps
Microsoft Power Platform integration with Finance and Operations

Connect to Dynamics AX from Power Apps

Article • 12/16/2022



Use this connection to easily build apps that read, update, and delete data from these versions of Dynamics AX:

• Dynamics AX Update1 + hotfix-KB article

• Dynamics AX Update 2 and later

For information about how to create an app, see Generate an app automatically or Create an app from scratch. These topics are written for Excel, but the same principles apply to Dynamics AX.

For information about how to add data from Dynamics AX to an existing app, see Add a data connection.

See the list of available connections, and learn how to manage connections in Power Apps.

Next steps

- Learn how to show data from a data source.
- Learn how to view details and create or update records.

Create a canvas app based on Excel data

Summarize this article for me

This article shows you how to create a canvas app using Excel data as your data source. Whether you have existing Excel spreadsheets with business data or want to leverage Excel's familiar structure, Power Apps provides flexible options to transform your data into powerful applications.

You'll learn three different approaches to work with Excel data:

Upload Excel files to create Dataverse tables for enhanced data management and security Connect to external Excel files stored in cloud services for quick app creation Start with a blank canvas app and add Excel data for complete customization control Before you begin, ensure your Excel data is properly formatted. This preparation step helps Power Apps read and interact with your data more effectively. For more information, see Formatted as a table in Excel.

To create an app by using Excel data, select one of the following options in Power Apps.

Expand table

Create options by using Excel	Benefits	Navigation
Upload an Excel or .CSV file to Power Apps	Uploaded Excel data generates a Dataverse table. With Dataverse's standard and custom tables, you can securely store your data in the cloud. These tables enable you to define your organization's data in a way that is tailored to your business needs, making it easier to use within your apps.	Select **Start with Data** > **Upload file**.
Connect to an external Excel file and use it to create a canvas app. Your data stays where it is.	Using the Excel connector in Power Apps provides a straightforward and fast method to create and deploy apps with data stored in Excel. It's especially advantageous for makers that already use Excel and want to enhance its functionality with minimal effort. Makers benefit from cloud access, easy data manipulation, rapid prototyping, and no need for a complex database setup.	Select **Start with Data** > **Excel Online (Business)**.
Create a blank canvas app and then add Excel data	You have complete flexibility, control, and customization. It allows you to design an app that meets your exact specifications without being limited by predesigned templates, providing greater creative freedom and adaptability to suit your unique needs.	On the left navigation pane, select **Create** > **Create from blank**. Then select the app size.

Upload an Excel or CSV file to Power Apps

When you upload your Excel data to Power Apps, it doesn't just stay as an Excel file. Instead, Power Apps converts it into a Dataverse table. This conversion makes it easier to integrate and manage your data within Power Apps, and you get more powerful features compared to using Excel alone. For more information, see Why use Dataverse?

1. Sign in to Power Apps.
2. On the home screen, select **Start with data**.
3. On the **Create an app** page, select **Upload file**.
4. Select **Select from device** and go to the location where your Excel file is saved. Select the file to upload it.

   > **Note**
   >
   > The maximum file size limit is 5 GB.

5. When Power Apps creates the table, select a column name or the table name to edit the properties to suit your needs. If you change column data types and there are values in cells that don't match the selected data type, Power Apps removes those values when it generates the table. For more information, see Create and edit tables using Power Apps.
6. Select **Row ownership** and choose how you want to manage row ownership.
7. When you're done, select **Save and open app**. Power Apps uploads the first 20 rows of data so you can start reviewing the data in your app. The system uploads the remaining data in the background.

Known issues

- The current data upload process doesn't take into account the environment data format setting.

Connect to an external Excel file from Power Apps

Store the Excel file in a cloud storage service like Dropbox, Google Drive, OneDrive, or OneDrive for Business because Power Apps can only connect to Excel files that are stored in the cloud.

Power Apps has an Excel connector that you can use to access your Excel data. The Excel Online (Business) connector offers flexibility by accessing a wide variety of cloud-based file storage services.

1. Sign in to Power Apps.
2. On the home screen, select **Start with data**.
3. On the **Create an app**, select **Excel Online (Business)**.
4. Only one connection appears at a time. To select a different connection, select the **...** button to switch connections or add a new connection.
5. Enter the file location and select the table.
6. When you're done, select **Create app**.

For more information about how to share Excel data, see Sharing Excel tables.

Create a blank canvas app and add Excel data

Create a blank app and then use Excel data that's formatted as a table as the source of the app's information.

Follow the example below to create an app with two screens where users can view (browse) the data from an Excel file on one screen, and on another screen, they can interact with (add, modify, or remove) that data. The data interaction features, such as creating, updating, and deleting records, are built into the second screen of the app.

Prerequisites

1. Copy this data and paste it into an Excel file.

Expand table

StartDay	StartTime	Volunteer	Backup
Saturday	10am-noon	Vasquez	Kumashiro
Saturday	noon-2pm	Ice	Singhal
Saturday	2pm-4pm	Myk	Mueller
Sunday	10am-noon	Li	Adams
Sunday	noon-2pm	Singh	Morgan

StartDay	StartTime	Volunteer	Backup
Sunday	2pm-4pm	Batye	Nguyen

2. Format the table in Excel and name it **Schedule** so that Power Apps can parse the information.

3. Save the file as **eventsignup.xlsx**, close it, and then upload it to a cloud-storage account such as OneDrive.

> **Important**
>
> You can use your own Excel file and review this example for general concepts only. However, the data in the Excel file must be formatted as a table.

Create a blank app and connect to data

1. Sign in to Power Apps.

2. On the left navigation pane, select **Create** > **Start from blank**.

3. Select the **Phone size** layout.

   The app opens in Power Apps Studio where you can add data and start building the app.

4. In the middle of the screen, select **Connect to data**.

5. In the **Data** pane, select **Add data**. Select the connection for your cloud-storage account if it appears. Otherwise, follow these steps to add a connection such as OneDrive:
   a. In the search box, enter **OneDrive** and select it.
   b. Select **Add a connection**.
   c. On the connection pane, select **Connect**.
   d. If prompted, provide your credentials for that account.

6. Under **Choose an Excel file**, find and then select the **eventsignup.xlsx** that you saved earlier.

7. Under **Choose a table**, select the check box for **Schedule**, and then select **Connect**.

8. In the upper-right corner of the **Data** pane, close it by selecting the close icon (**X**).

Create the view screen

1. On the command bar, select **New screen** > **List**.

A screen is added with several default controls, such as a search box and a **Gallery** control. The gallery covers the entire screen under the search box.

2. At the top of the new screen, select the **[Title]** label control and rename it to **View records**.

3. In the **Tree view**, select **BrowseGallery1**.

4. In the gallery's **Properties** pane, set the **Layout** to **Title, subtitle, and body**.

5. In the formula bar, replace **CustomGallerySample** with **Schedule**, and replace both instances of **SampleText** with **Volunteer**.

6. On the right edge of the formula bar, select the **Expand formula bar** down arrow, and then select **Format text**.

   The formula matches this example:

   powerfx

   ```
   SortByColumns(
       Search(
           Schedule,
           TextSearchBox1.Text,
           "Volunteer"
       ),
       "Volunteer",
       If(
           SortDescending1,
           SortOrder.Descending,
           SortOrder.Ascending
       )
   )
   ```

7. In the **Properties** pane, select **Edit** next to the **Fields** label.

8. In the **Title2** box, select **Volunteer**, in the **Subtitle2** box, select **StartDay**, and in the **Body1** box, select **StartTime**.

9. In the upper-right corner of the **Data** pane, close it by selecting the close icon (X).

   Users can sort and filter the gallery by volunteer name based on the **SortByColumns** and **Search** functions in that formula.

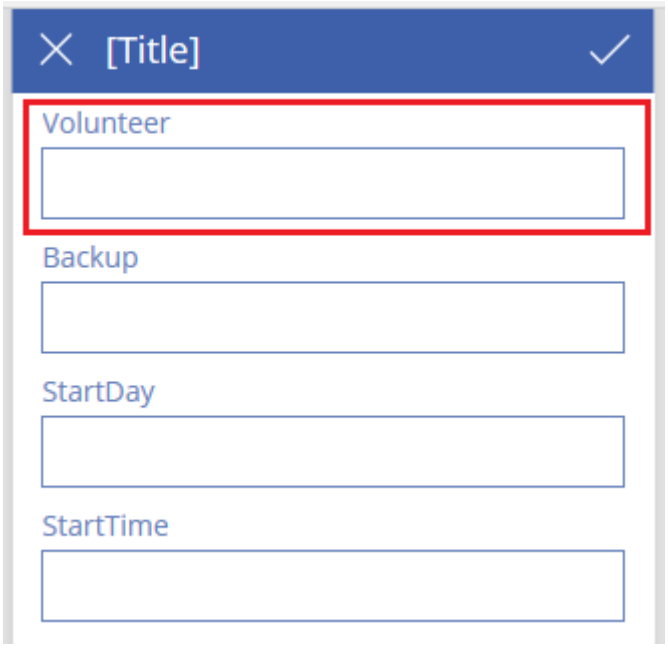
   - If a user types at least one letter in the search box, the gallery shows only those records for which the **Volunteer** field contains the text that the user typed.

- If a user selects the sort button (between the refresh button and the plus button in the title bar), the gallery shows the records in ascending or descending order (depending on how many times the user selects the button) based on the **Volunteer** field.

For more information about these and other functions, see the formula reference.

Create the change screen

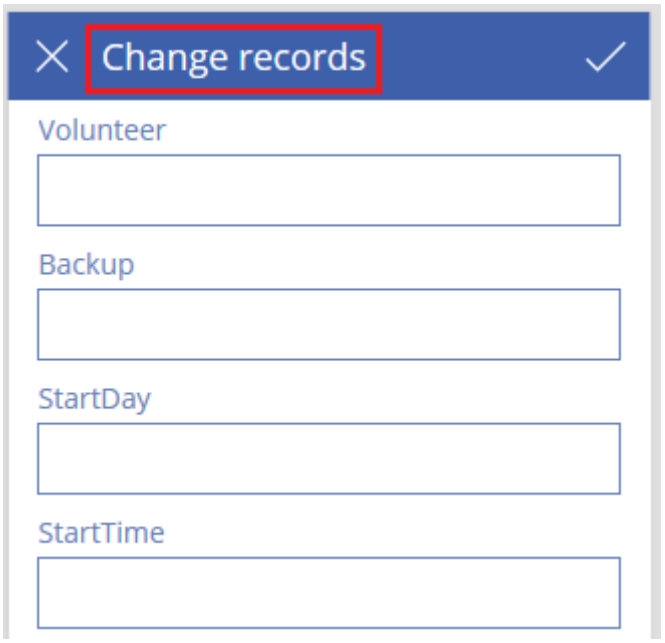
1. On the command bar, select **New screen** > **Form**.
2. In the **Tree view**, select **EditForm1**.
3. On the **Properties** pane, select the down arrow next to **Data source**, and then select **Schedule** in the list that appears.
4. Under the data source that you specified, select **Edit fields**.
5. In the **Fields** pane, select **Add field**, select the check box for each field, and then select **Add**.
6. Select the arrow next to the name of each field to collapse it, and then drag the **Volunteer** field up so that it appears at the top of the list of fields.



7. In the upper-right corner of the **Fields** pane, close it by selecting the close icon (X).
8. Set the **Item** property of the form to this expression in the formula bar:

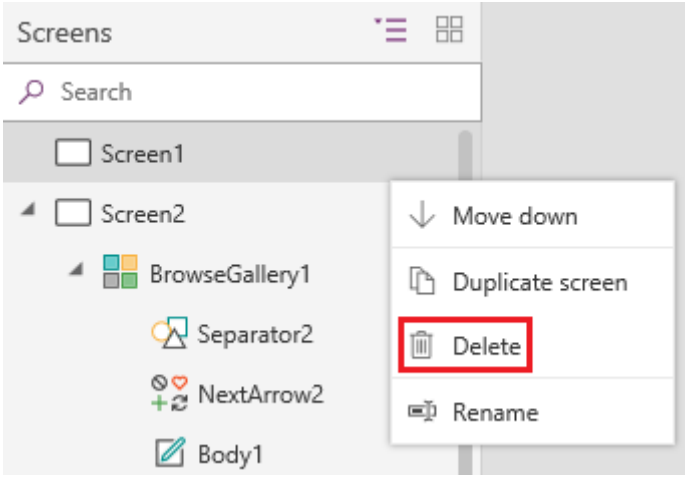
```
BrowseGallery1.Selected
```

9. At the top of the screen, select the **Label** control, and then replace **[Title]** with **Change records**.



Delete and rename screens

1. In the **Tree view**, select the ellipsis (...) for **Screen1**, and then select **Delete**.



2. Select the ellipsis (...) for **Screen2**, select **Rename**, and then type or paste **ViewScreen**.

3. Select the ellipsis (...) for **Screen3**, select **Rename**, and then type or paste **ChangeScreen**.

Configure icons on the view screen

1. Near the top of the **ViewScreen**, select the circular arrow icon.



2. Set the **OnSelect** property for that icon to this formula:

```
Refresh(Schedule)
```

When the user selects this icon, the data from **Schedule** refreshes from the Excel file.

For more information about this and other functions, see formula reference.

3. In the upper-right corner of the **ViewScreen**, select the plus icon.

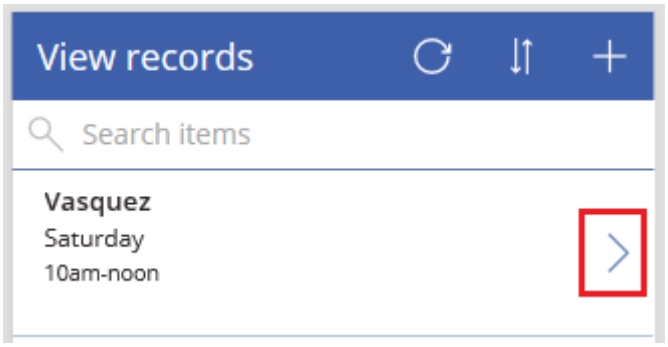


4. Set the **OnSelect** property for that icon to this formula:

```
NewForm(EditForm1);Navigate(ChangeScreen,ScreenTransition.None)
```

When the user selects this icon, **ChangeScreen** appears with each field empty, so the user can create a record more easily.

5. Select the right-pointing arrow for the first record in the gallery.



6. Set the **OnSelect** property for the arrow to this formula:

```
EditForm(EditForm1); Navigate(ChangeScreen, ScreenTransition.None)
```

When the user selects this icon, **ChangeScreen** appears with each field showing the data for the selected record, so the user can edit or delete the record more easily.

Configure icons on the change screen

1. On **ChangeScreen**, select the "X" icon in the upper left corner.



2. Set the **OnSelect** property for that icon to this formula:

```
ResetForm(EditForm1);Navigate(ViewScreen, ScreenTransition.None)
```

When the user selects this icon, the app discards any changes made in this screen and opens the view screen.

3. In the upper-right corner, select the checkmark icon.



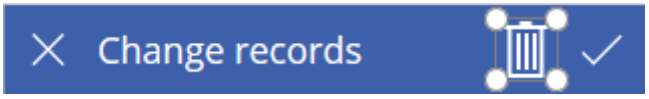
4. Set the **OnSelect** property for the checkmark to this formula:

```
SubmitForm(EditForm1); Navigate(ViewScreen, ScreenTransition.None)
```

When the user selects this icon, the app saves any changes made in this screen and opens the view screen.

5. On the **Insert** tab, select **Icons**, and then select the **Trash** icon.

6. Set the new icon's **Color** property to **White** and move the new icon so it appears next to the checkmark icon.



7. Set the **Visible** property for the trash icon to this formula:

```
EditForm1.Mode = FormMode.Edit
```

This icon appears only when the form is in **Edit** mode, not in **New** mode.

8. Set the **OnSelect** property for the trash icon to this formula:

```
Remove(Schedule, BrowseGallery1.Selected); Navigate(ViewScreen,
ScreenTransition.None)
```

When the user selects this icon, the app deletes the selected record from the data source and opens the view screen.

Test the app

1. Select the **ViewScreen**, and then preview the app by pressing `F5` or select **Preview**.



2. Type or paste one or more letters in the search box to filter the list based on the volunteer's name.

3. Select the sort icon one or more times to show the data in ascending or descending order based on the volunteer's name.
4. Add a record.
5. Update the record that you added, and then save the changes.
6. Update the record that you added, and then cancel the changes.
7. Delete the record that you added.
8. Close Preview mode by pressing Esc or selecting the close icon in the upper-right corner.

Next steps

- Press Ctrl+S to save your app in the cloud so you can run it from other devices.
- Share the app so other people can run it.

Last updated on 02/24/2026

Connect to Microsoft Translator from Power Apps

Article • 12/16/2022



Add the Microsoft Translator connector to display translated text in a **Label** control in your app. For example, you can create an input text box that asks the user to enter some text to translate. In another label, you can display the translated text.

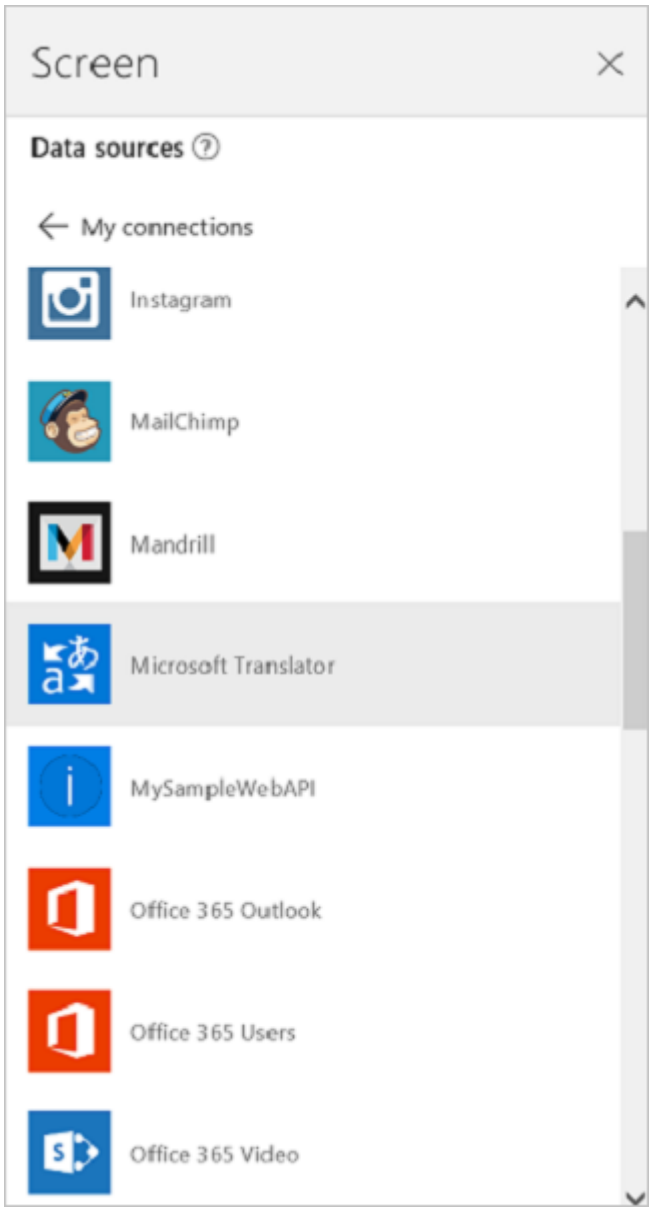
This topic shows you how to create the Microsoft Translator connection, use the Microsoft Translator connection in an app, and lists the available functions.

Prerequisites

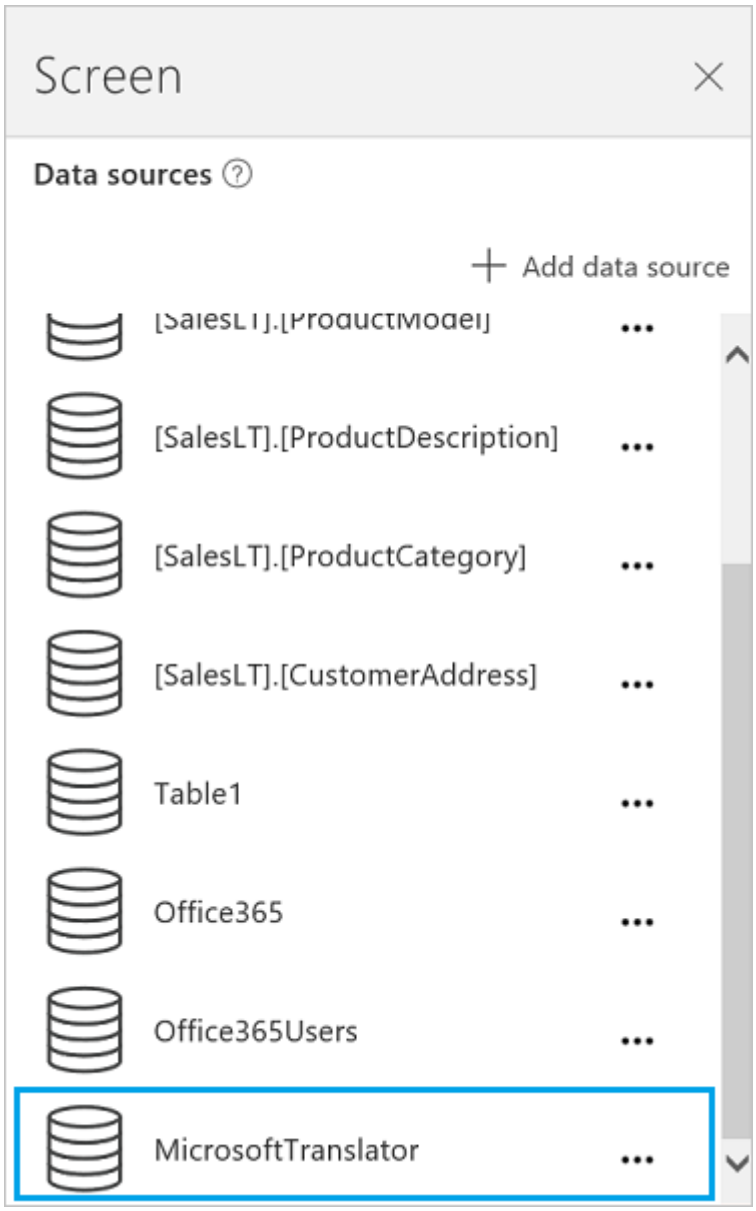
- Access to Power Apps
- Add the connection
- Create an app from a template, from data, or from scratch

Connect to Microsoft Translator

1. Create a blank canvas app.
2. In the left-hand pane, select **Data** tab, and then select **Add data source**.
3. Select **New connection**, and then select **Microsoft Translator**:



4. Select **Connect**. Your connection appears under **Data sources**:



Use the Microsoft Translator connection in your app

Translate text

1. From the **Tree View** on the left-side of the screen, select **Insert**, and then choose the **Text input** control. Rename the text input control to **Source**.

2. Add a **Drop down** list (**Insert** menu > **Controls**), rename it to **TargetLang**, and move it below **Source**.

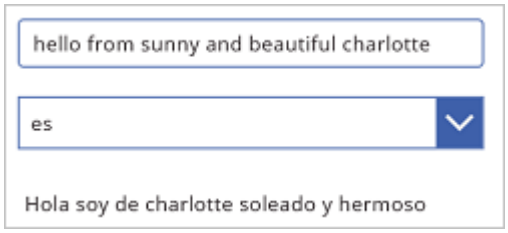
3. Set the Items property of **TargetLang** to the following formula:

```
MicrosoftTranslator.Languages()
```

4. Add a label, move it below **TargetLang**, and set its **Text** property to the following formula:

   ```
   MicrosoftTranslator.Translate(Source.Text, TargetLang.Selected.Value)
   ```

5. Type some text into **Source**, and select a language in **TargetLang**. The label shows the text that you entered in the language you chose:



Speak translated text

If you haven't already, follow the steps in the previous section to translate some text. These next steps use the same controls.

1. Set the **Items** property of the **TargetLang** drop-down list to the following formula:

   ```
   MicrosoftTranslator.SpeechLanguages()
   ```

2. Rename the second label (not the **Source** box) to **Target**.

3. Add an **Audio** control (**Insert** menu > **Media**), and set its **Media** property to the following formula:

   ```
   MicrosoftTranslator.TextToSpeech(Target.Text, TargetLang.Selected.Value)
   ```

4. Press F5, or select the Preview button (). Type some text into **Source**, select a language in **TargetLang**, and then select the play button in the audio control.

   The app plays an audio version of the text that you entered in the language you chose.

5. Press Esc to return to the default workspace.

Detect the source language

These next steps use the same **Source** text input and **Target** text controls. You can create new controls if you prefer, just update the names in the formula.

1. Select the **Target** text control, and set the **Text** property to the following formula:

`MicrosoftTranslator.Detect(Source.Text).Name`

2. Type some text into **Source**.

   The label shows you the language of the text that you typed. For example, the label shows **French** if you type **bonjour**, or **Italian** if you type **ciao**.

View the available functions

This connection includes the following functions:

Function Name	Description
Languages	Retrieves all languages that Microsoft Translator supports
Translate	Translates text to a specified language using Microsoft Translator
Detect	Detects source language of given text
SpeechLanguages	Retrieves the languages available for speech synthesis
TextToSpeech	Converts a given text into speech as an audio stream in wave format

Languages

Get languages: Retrieves all languages that Microsoft Translator supports

Input properties

None.

Output properties

Property Name	Data Type	Required	Description
Code	string	No	
Name	string	No	

Translate

Translate text: Translates text to a specified language using Microsoft Translator

Input properties

Name	Data Type	Required	Description
query	string	yes	Text to translate
languageTo	string	yes	Target language code (example: 'fr')
languageFrom	string	no	Source language (if not provided, Microsoft Translator will try to auto-detect) (example: en)
category	string	no	Translation category (default: 'general')

Output properties

None.

Detect

Detect language: Detects source language of given text

Input properties

Name	Data Type	Required	Description
query	string	yes	Text whose language will be identified

Output properties

Property Name	Data Type	Required	Description
Code	string	No	
Name	string	No	

SpeechLanguages

Get speech languages: Retrieves the languages available for speech synthesis

Input properties

None.

Output properties

Property Name	Data Type	Required	Description
Code	string	No	
Name	string	No	

TextToSpeech

Text to speech: Converts a given text into speech as an audio stream in wave format

Input properties

Name	Data Type	Required	Description
query	string	yes	Text to convert
language	string	yes	Language code to generate speech (example: 'en-us')

Output properties

None.

Helpful links

See all the available connections.
Learn how to add connections to your apps.

Connect to Office 365 Outlook from Power Apps

Article • 12/16/2022



If you connect to Office 365 Outlook, you can show, send, delete, and reply to email messages, in addition to other tasks.

You can add controls to perform these functions in your app. For example, you can add **Text input** controls to ask for the recipient, the subject, and the body of the email, and add a **Button** control to send the email.

This article shows you how to add Office 365 Outlook as a connection, add Office 365 Outlook as a data source to your app, and use this data in different controls.

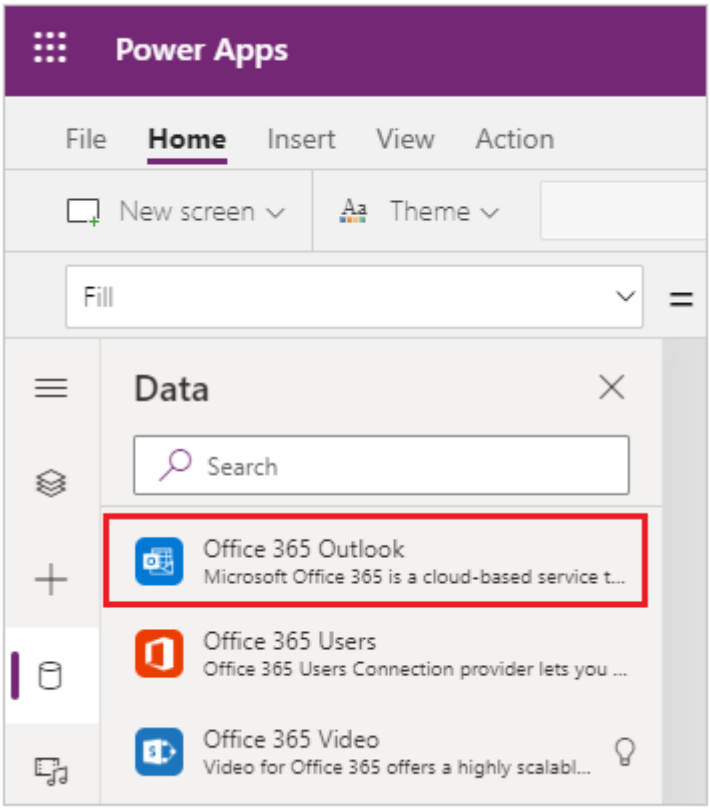
> **Important**
>
> As of this writing, the calendar operation doesn't support recurring events.

Prerequisites

- Access to Power Apps
- Add the connection
- Create an app from a template, from data, or from scratch

Connect to Office 365 Outlook

1. Add a data connection and select **Office 365 Outlook**:



2. Select **Connect**, and if prompted to sign in, enter your work account.

The Office 365 Outlook connection has been created and added to your app. Now, it's ready to be used.

Show messages

1. On the **Insert** menu, select **Gallery**, and then select a **Blank vertical** control.

2. Set its Items property to the following formula:

   ```
   Office365Outlook.GetEmails({fetchOnlyUnread:false})
   ```

   After changing the settings, change **Layout** to **Title, Subtitle, Body**.

   The gallery control is automatically populated with some of your emails.

3. In the gallery, set the **Text** property of the first label to `ThisItem.From`. Set the second label to `ThisItem.Subject`. Set the third label to `ThisItem.BodyPreview`. You can also resize the labels.

   The gallery control is automatically populated with the new properties.

4. This function has several optional parameters available. Set the gallery's **Items** property to one of the following formulas:

`Office365Outlook.GetEmails({fetchOnlyUnread:false})`

`Office365Outlook.GetEmails({fetchOnlyUnread:false, top:2})`

`Office365Outlook.GetEmails({folderPath:"Sent Items", fetchOnlyUnread:false, top:2})`

`Office365Outlook.GetEmails({folderPath:"Sent Items", fetchOnlyUnread:false, top:2, searchQuery:"powerapps"})`

`Office365Outlook.GetEmails({folderPath:"Deleted Items", fetchOnlyUnread:false, top:2, skip:3})`

Send a message

1. On the **Insert** menu, select **Text**, and then select **Text input**.

2. Repeat the previous step 2 more times so that you have three boxes, and then arrange them in a column:



3. Rename the controls to:

   - **inputTo**
   - **inputSubject**
   - **inputBody**

4. On the **Insert** menu, select **Controls**, and then select **Button**. Set its **OnSelect** property to the following formula:

   `Office365Outlook.SendEmail(inputTo.Text, inputSubject.Text, inputBody.Text)`

5. Move the button so that it appears under all the other controls, and set its **Text** property to "**Send email**".

6. Press F5, or select the Preview button (▷). Type in a valid email address in **inputTo**, and type whatever you want in the other two **Text input** controls.

7. Select **Send email** to send the message. Press Esc to return to the default workspace.

Send a message with an attachment

You can, for example, create an app in which the user takes pictures by using the device's camera and then sends them as attachments. Users can also attach many other kinds of files to an email app.

To add an attachment to a message, follow the steps in the previous section, but add a parameter to specify an attachment (when you set the **OnSelect** property of the button). This parameter is structured as a table in which you specify up to three properties for each attachment:

- Name
- ContentBytes
- @odata.type

> ⓘ **Note**
>
> You can specify the @odata.type property for only one attachment, and you can set it to an empty string.

In this example, a photo will be sent as **file1.jpg**:

```
Office365Outlook.SendEmail(inputTo.Text, inputSubject.Text, inputBody.Text, {Attachments:Table({Name:"file1.jpg", ContentBytes:Camera1.Photo, '@odata.type':""})})
```

In this example, an audio file will be sent in addition to the photo:

```
Office365Outlook.SendEmail(inputTo.Text, inputSubject.Text, inputBody.Text, {Attachments:Table({Name:"file1.jpg", ContentBytes:Camera1.Photo, '@odata.type':""}, {Name:"AudioFile", ContentBytes:microphone1.audio })})
```

Delete a message

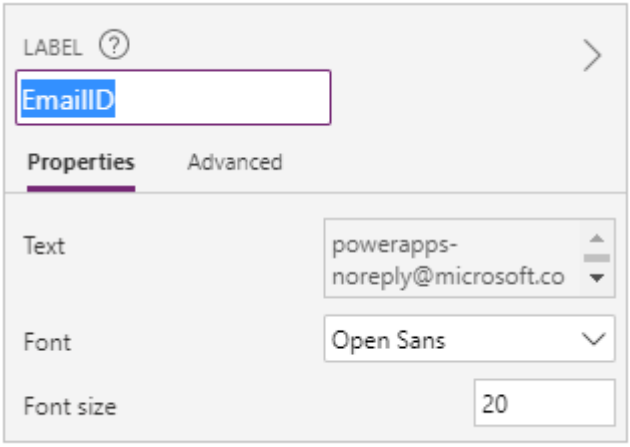
1. On the **Insert** menu, select **Gallery**, and then select a **Text gallery** control.
2. Set its **Items** property to the following formula:

   ```
   Office365Outlook.GetEmails({fetchOnlyUnread:false})
   ```

The gallery control is automatically populated with some of your emails.

3. In the gallery, set the **Text** property of the first label to `ThisItem.Id`. Set the second label to `ThisItem.Subject`. Set the third label to `ThisItem.Body`.

4. Select the first label in the gallery, and rename it to **EmailID**:



5. Select the third label in the gallery, and add a **Button** (**Insert** menu). Set the button's **OnSelect** property to the following formula:

   ```
   Office365Outlook.DeleteEmail(EmailID.Text)
   ```

6. Press F5, or select the Preview button (▷). Select one of the emails in your gallery, and select the button.

   > **Note**
   >
   > This deletes the selected email from your inbox. So, choose wisely.

7. Press Esc to return to the default workspace.

Mark a message as read

This section uses the same controls as Delete a message.

1. Set the button's **OnSelect** property to the following formula:

   ```
   Office365Outlook.MarkAsRead(EmailID.Text)
   ```

2. Press F5, or select the Preview button (▷). Select one of the unread emails, and then select the button.

3. Press Esc to return to the default workspace.

See also

- For a list of all functions and their parameters, see the Office 365 Outlook reference.
- See all the available connections.
- Learn how to manage your connections.

Connect to Office 365 Users connection from Power Apps

Article • 12/16/2022



Office 365 Users lets you access user profiles in your organization using your Office 365 account. You can perform various actions such as get your profile, a user's profile, a user's manager or direct reports.

You can display this information in a label on your app. You can display one function, multiple functions, or even combine different functions. For example, you can create an expression that combines the User Name and Phone Number, and then display this information in your app.

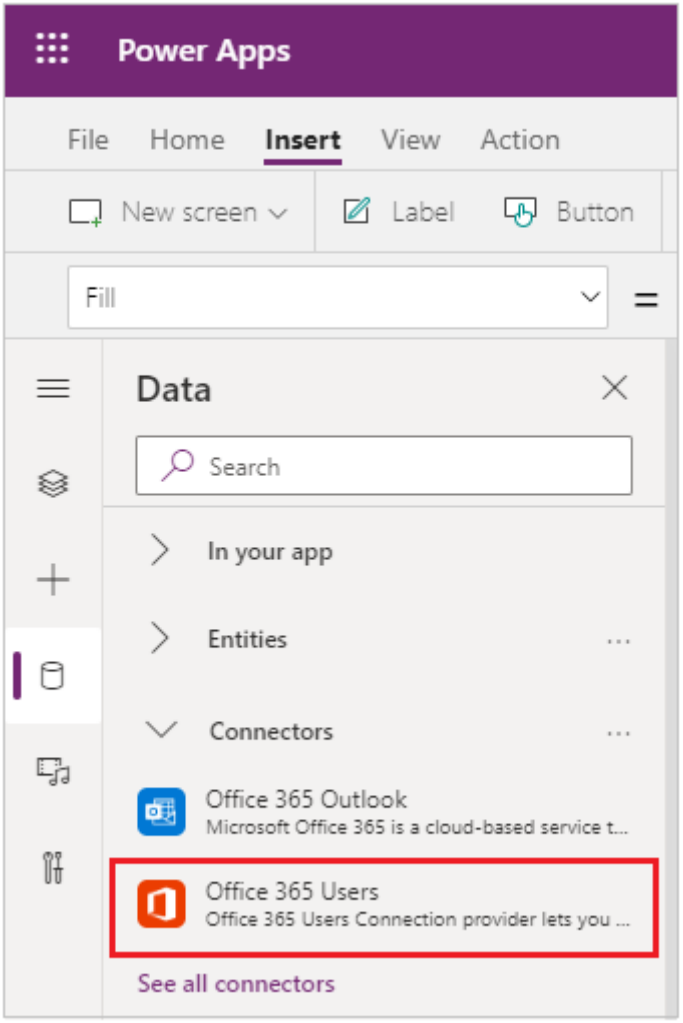
This article shows you how to add Office 365 Users as a connection, add Office 365 Users as a data source to your app, and use table data in a gallery control.

Prerequisites

- Access to Power Apps
- Add the connection
- Create an app from a template, from data, or from scratch

Add a connection

1. Add a data connection and select **Office 365 Users**:



2. Select **Connect**, and if prompted to sign in, enter your work account.

The Office 365 Users connection has been created and added to your app. Now, it's ready to be used.

Use the connection in your app

Show information about the current user

1. On the **Insert** menu, select **Label**

2. In the function bar, set its **Text** property to any of the following formulas:

```
Office365Users.MyProfile().City
Office365Users.MyProfile().CompanyName
Office365Users.MyProfile().Country
Office365Users.MyProfile().Department
Office365Users.MyProfile().DisplayName
Office365Users.MyProfile().GivenName
```

`Office365Users.MyProfile().Id`

`Office365Users.MyProfile().JobTitle`

`Office365Users.MyProfile().Mail`

`Office365Users.MyProfile().MailNickname`

`Office365Users.MyProfile().mobilePhone`

`Office365Users.MyProfile().OfficeLocation`

`Office365Users.MyProfile().PostalCode`

`Office365Users.MyProfile().Surname`

`Office365Users.MyProfile().TelephoneNumber`

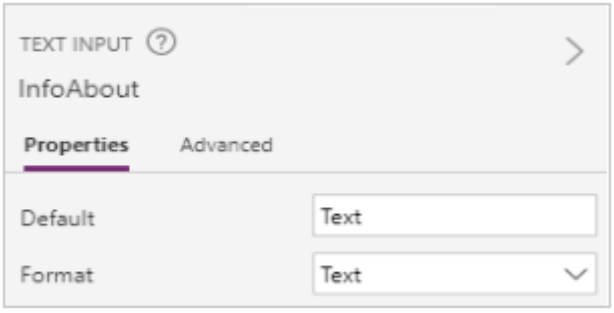
`Office365Users.MyProfile().UserPrincipalName`

`Office365Users.MyProfile().AccountEnabled`

The label shows the information that you entered about the current user.

Show information about another user

1. On the **Insert** menu, select **Text**, and then select **Text input**. Rename it **InfoAbout**:



2. In **InfoAbout**, type or paste an email address of a user in your organization. For example, type in *yourName@yourCompany.com*.

3. Add a **Label** (**Insert** menu), and set its **Text** property to any of the following formulas:

   - To show information about another user:

     `Office365Users.UserProfile(InfoAbout.Text).City`

     `Office365Users.UserProfile(InfoAbout.Text).CompanyName`

     `Office365Users.UserProfile(InfoAbout.Text).Country`

     `Office365Users.UserProfile(InfoAbout.Text).Department`

     `Office365Users.UserProfile(InfoAbout.Text).DisplayName`

     `Office365Users.UserProfile(InfoAbout.Text).GivenName`

     `Office365Users.UserProfile(InfoAbout.Text).Id`

`Office365Users.UserProfile(InfoAbout.Text).JobTitle`

`Office365Users.UserProfile(InfoAbout.Text).Mail`

`Office365Users.UserProfile(InfoAbout.Text).MailNickname`

`Office365Users.UserProfile(InfoAbout.Text).mobilePhone`

`Office365Users.UserProfile(InfoAbout.Text).OfficeLocation`

`Office365Users.UserProfile(InfoAbout.Text).PostalCode`

`Office365Users.UserProfile(InfoAbout.Text).Surname`

`Office365Users.UserProfile(InfoAbout.Text).TelephoneNumber`

`Office365Users.UserProfile(InfoAbout.Text).UserPrincipalName`

`Office365Users.UserProfile(InfoAbout.Text).AccountEnabled`

`Office365Users.UserProfile(InfoAbout.Text).BusinessPhones`

- To show information about another user's manager:

`Office365Users.Manager(InfoAbout.Text).City`

`Office365Users.Manager(InfoAbout.Text).CompanyName`

`Office365Users.Manager(InfoAbout.Text).Country`

`Office365Users.Manager(InfoAbout.Text).Department`

`Office365Users.Manager(InfoAbout.Text).DisplayName`

`Office365Users.Manager(InfoAbout.Text).GivenName`

`Office365Users.Manager(InfoAbout.Text).Id`

`Office365Users.Manager(InfoAbout.Text).JobTitle`

`Office365Users.Manager(InfoAbout.Text).Mail`

`Office365Users.Manager(InfoAbout.Text).MailNickname`

`Office365Users.Manager(InfoAbout.Text).mobilePhone`

`Office365Users.Manager(InfoAbout.Text).OfficeLocation`

`Office365Users.Manager(InfoAbout.Text).PostalCode`

`Office365Users.Manager(InfoAbout.Text).Surname`

`Office365Users.Manager(InfoAbout.Text).TelephoneNumber`

`Office365Users.Manager(InfoAbout.Text).UserPrincipalName`

`Office365Users.Manager(InfoAbout.Text).AccountEnabled`

`Office365Users.Manager(InfoAbout.Text).BusinessPhones`

The label shows the information that you entered about the user you specified or that user's manager.

> ⓘ **Note**

If you're developing an app based on a table in the Microsoft Dataverse, you can specify a user based on ID instead of email address.

For example, you can create an app automatically, add a screen that contains a **Label** control, and set the control's **Text** property to this formula: **Office365Users.UserProfile(BrowseGallery1.Selected.CreatedByUser).DisplayName**

If you create a contact and select that contact in the browse screen of the app, the **Label** control will show your display name.

Show the direct reports of another user

1. Add a **Text input** control (**Insert** menu > **Text**), and rename it **InfoAbout**.
2. In **InfoAbout**, enter the email address of a user in your organization. For example, enter *yourManagersName@yourCompany.com*
3. Add a **With text** gallery (**Insert** menu > **Gallery**), and set its **Items** property to the following formula:

   ```
   Office365Users.DirectReports(InfoAbout.Text)
   ```

   The gallery shows information about the direct reports of the user you entered.

   With the gallery selected, the right-hand pane shows options for that gallery.
4. In the second list, select **JobTitle**. In the third list, select **DisplayName**. The gallery is updated to show these values.

> ⓘ **Note**
>
> The first box is actually an image control. If you don't have an image, you can delete the image control, and add a label in its place. Add and configure controls is a good resource.

Search for users

1. Add a **Text input** control (**Insert** menu > **Text input**).
2. Select **Insert** > **Vertical gallery**.
3. Change layout of the vertical gallery to **Title, subtitle, and body**.

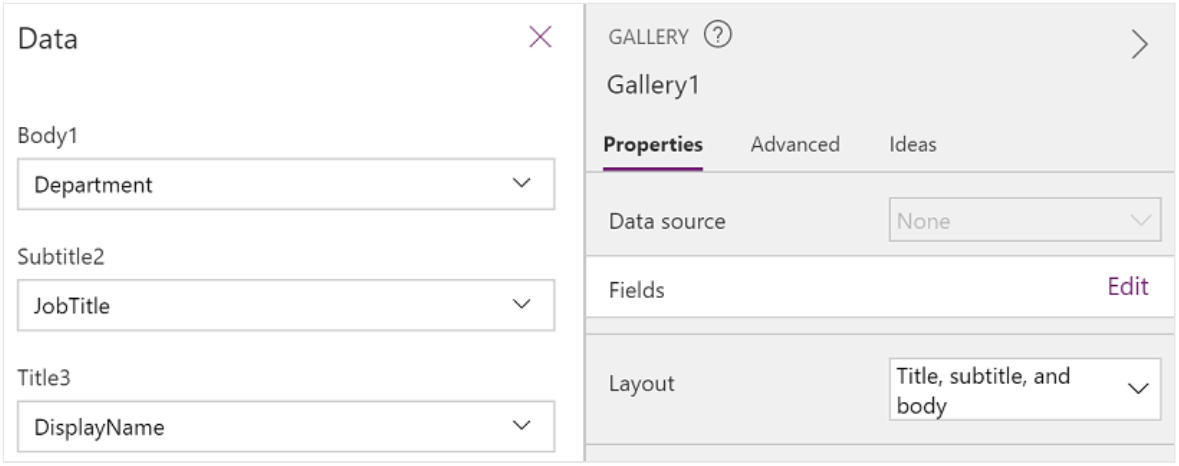
4. Select the arrow icon from first row inside the gallery, and delete it.

5. Update the vertical gallery formula to the following.

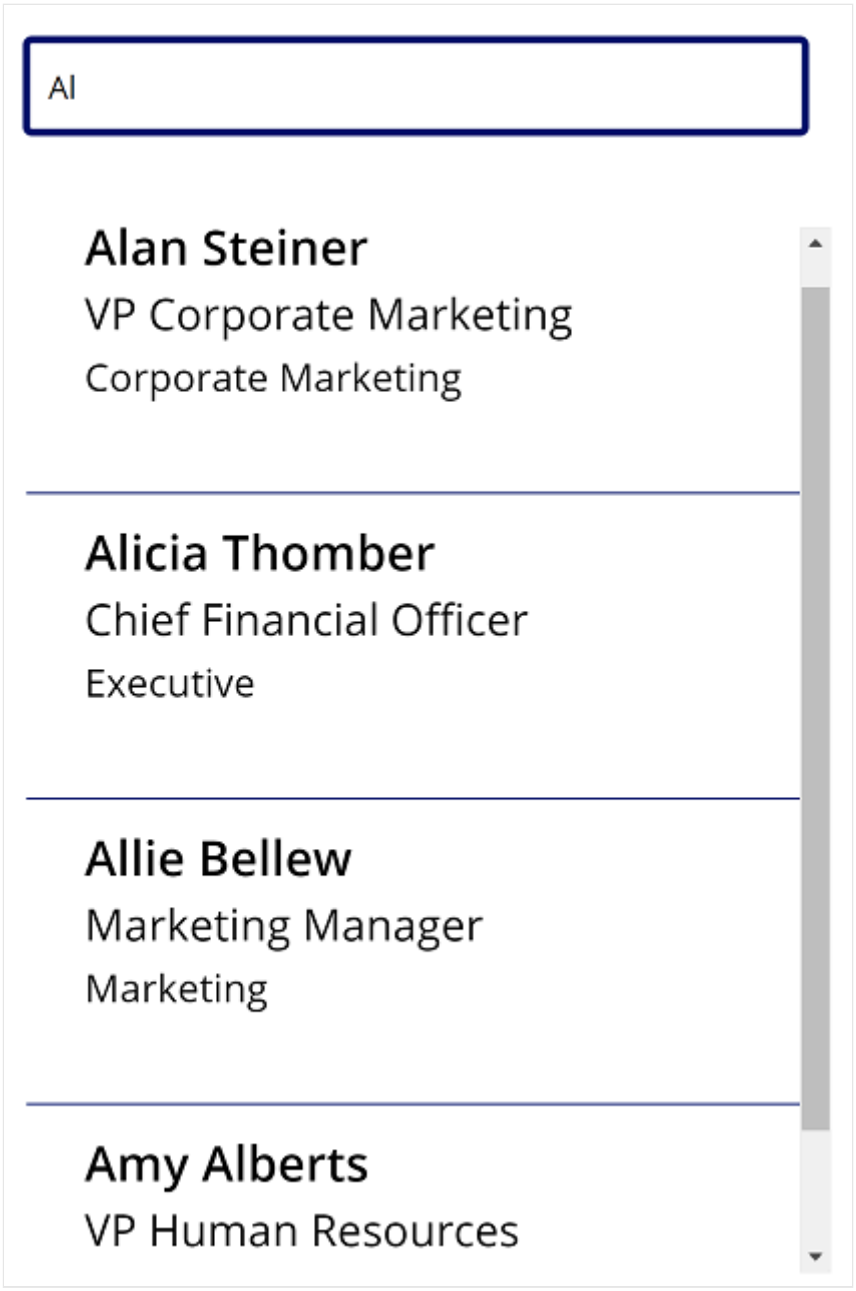
```
Power Fx

Office365Users.SearchUserV2({searchTerm:TextInput1.Text,top:5}).value
```

   In the formula above, the search term references text entered in the text input control ("TextInput1" in this example); and the search results are limited to top 5.

6. Update vertical gallery fields to show **DisplayName** in gallery title, **JobTitle** in subtitle, and **Department** in body.



7. Play the app, and enter text inside the text input box. As you enter text, the results change based on the input string changes.



View the available functions

This connection includes the following functions:

Expand table

Function Name	Description
DirectReports	Returns the direct reports for the specified user.
Manager	Retrieves user profile for the manager of the specified user.
MyProfile	Retrieves the profile for the current user.
SearchUser	Retrieves search results of user profiles.

Function Name	Description
UserProfile	Retrieves a specific user profile.

MyProfile

Get my profile: Retrieves the profile for the current user.

Input properties

None.

Output properties

Expand table

Property Name	Type	Description
City	string	City of user.
CompanyName	string	Company of user.
Country	string	Country of user.
Department	string	Department of user.
DisplayName	string	Display name of user.
GivenName	string	Given name of user.
Id	string	User id.
JobTitle	string	Job title of user.
Mail	string	Email id of user.
MailNickname	string	Nickname of user.
mobilePhone	string	Mobile phone of user.
OfficeLocation	string	Office location of user.
PostalCode	string	Postal code of user.
Surname	string	Surname of user.
TelephoneNumber	string	Telephone number of user.

Property Name	Type	Description
UserPrincipalName	string	User Principal Name.
AccountEnabled	boolean	Account enabled flag.
BusinessPhones	string	Phone numbers of user's company.

UserProfile

Get user profile: Retrieves a specific user profile.

Input properties

Expand table

Name	Data Type	Required	Description
Id	string	yes	User principal name or email id.

Output properties

Expand table

Property Name	Type	Description
City	string	City of user.
CompanyName	string	Company of user.
Country	string	Country of user.
Department	string	Department of user.
DisplayName	string	Display name of user.
GivenName	string	Given name of user.
Id	string	User id.
JobTitle	string	Job title of user.
Mail	string	Email id of user.
MailNickname	string	Nickname of user.
Surname	string	Surname of user.

Property Name	Type	Description
TelephoneNumber	string	Telephone number of user.
UserPrincipalName	string	User Principal Name.
AccountEnabled	boolean	Account enabled flag.
BusinessPhones	string	Phone numbers of user's company.

Manager

Get manager: Retrieves user profile for the manager of the specified user.

Input properties

Expand table

Name	Data Type	Required	Description
Id	string	yes	User principal name or email id.

Output properties

Expand table

Property Name	Type	Description
City	string	City of user.
CompanyName	string	Company of user.
Country	string	Country of user.
Department	string	Department of user.
DisplayName	string	Display name of user.
GivenName	string	Given name of user.
Id	string	User id.
JobTitle	string	Job title of user.
Mail	string	Email id of user.
MailNickname	string	Nickname of user.

Property Name	Type	Description
mobilePhone	string	Mobile phone of user.
OfficeLocation	string	Office location of user.
PostalCode	string	Postal code of user.
Surname	string	Surname of user.
TelephoneNumber	string	Telephone number of user.
UserPrincipalName	string	User Principal Name.
AccountEnabled	boolean	Account enabled flag.
BusinessPhones	string	Phone numbers of user's company.

DirectReports

Get direct reports: Get direct reports.

Input properties

Expand table

Name	Data Type	Required	Description
Id	string	yes	User principal name or email id.

Output properties

Expand table

Property Name	Type	Description
City	string	City of user.
CompanyName	string	Company of user.
Country	string	Country of user.
Department	string	Department of user.
DisplayName	string	Display name of user.
GivenName	string	Given name of user.

Property Name	Type	Description
Id	string	User id.
JobTitle	string	Job title of user.
Mail	string	Email id of user.
MailNickname	string	Nickname of user.
mobilePhone	string	Mobile phone of user.
OfficeLocation	string	Office location of user.
PostalCode	string	Postal code of user.
Surname	string	Surname of user.
TelephoneNumber	string	Telephone number of user.
UserPrincipalName	string	User Principal Name.
AccountEnabled	boolean	Account enabled flag.
BusinessPhones	string	Phone numbers of user's company.

SearchUser

Search for users: Retrieves search results of user profiles.

Input properties

Expand table

Name	Data Type	Required	Description
searchTerm	string	no	Search string. Applies to: display name, given name, surname, mail, mail nickname, and user principal name.

Output properties

Expand table

Property Name	Type	Description
City	string	City of user.

Property Name	Type	Description
CompanyName	string	Company of user.
Country	string	Country of user.
Department	string	Department of user.
DisplayName	string	Display name of user.
GivenName	string	Given name of user.
Id	string	User id.
JobTitle	string	Job title of user.
Mail	string	Email id of user.
MailNickname	string	Nickname of user.
mobilePhone	string	Mobile phone of user.
OfficeLocation	string	Office location of user.
PostalCode	string	Postal code of user.
Surname	string	Surname of user.
TelephoneNumber	string	Telephone number of user.
UserPrincipalName	string	User Principal Name.
AccountEnabled	boolean	Account enabled flag.
BusinessPhones	string	Phone numbers of user's company.

See also

- See all the available connections.
- Learn how to add connections to your apps.

Feedback

Was this page helpful? Yes No

Provide product feedback

Connect to an Oracle database from Power Apps

09/04/2025

List tables, and create, read, update, and delete table rows in an Oracle database after you create a connection and build an app in Power Apps. The Oracle Database connection supports full delegation of filtering, sorting, and other functions but not triggers or stored procedures.

Prerequisites

- Oracle 9 and later
- Oracle Data Access Client (ODAC) software 11.2 or later
- Installation of an on-premises data gateway
- Installation of the Oracle client SDK

Install an on-premises data gateway

To install a gateway, follow the steps in this tutorial.

An on-premises data gateway acts as a bridge, providing quick and secure data transfer between on-premises data (data that isn't in the cloud) and the Power BI, Power Automate, Logic Apps, and Power Apps services. You can use the same gateway with multiple services and multiple data sources. For more information, see Understand gateways.

Install Oracle client

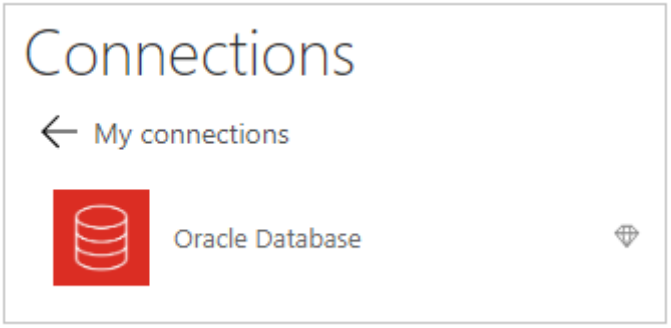
On the same computer as the on-premises data gateway, install ODAC 11.2 or later. For example, you can install 64-bit ODAC 12c Release 4 (12.1.0.2.4) for Windows x64 or choose a version that you prefer. You must choose a windows installer version since xcopy version does not work with on-premises data gateway. Otherwise, an error will appear if you try to create or use the connection, as the list of known issues describes.

Create an app from a table in an Oracle database

1. Go to Power Apps.
2. Select **Create** from the left-pane.
3. Under **Start from**, select **More data sources**.

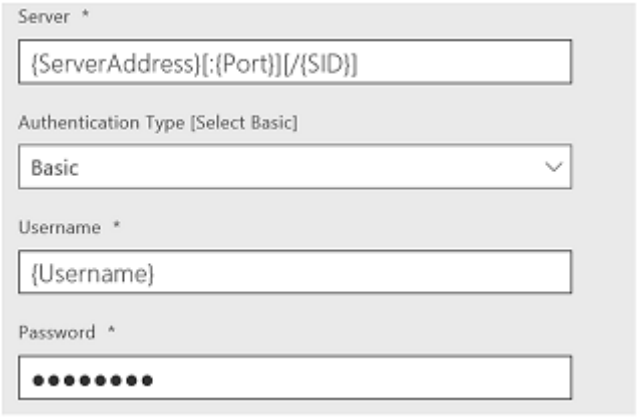
4. Select **New connection**.

5. In the list of connections, select **Oracle Database**.



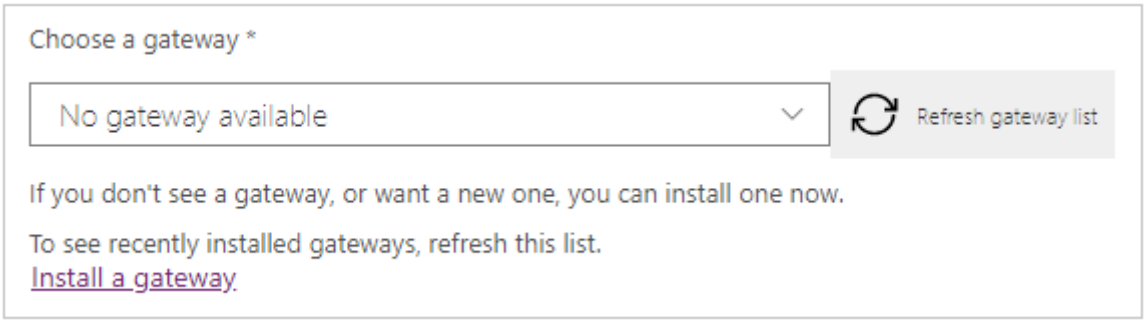
6. Specify the name of an Oracle server, a username, and a password.

   Specify a server in this format if an SID is required:
   `ServerName/SID`



7. Select the gateway that you want to use, or install one.

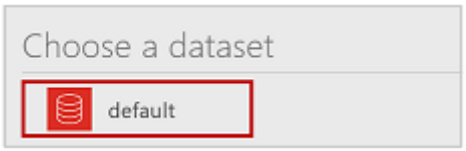
   If your gateway doesn't appear after you install it, select **Refresh gateway list**.



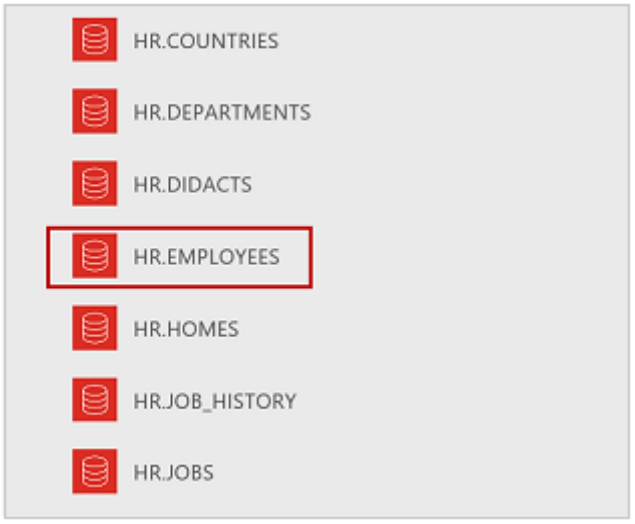
8. Select **Create** to create the connection.



9. Select **default** dataset.



10. In the list of tables, select the table that you want to use.

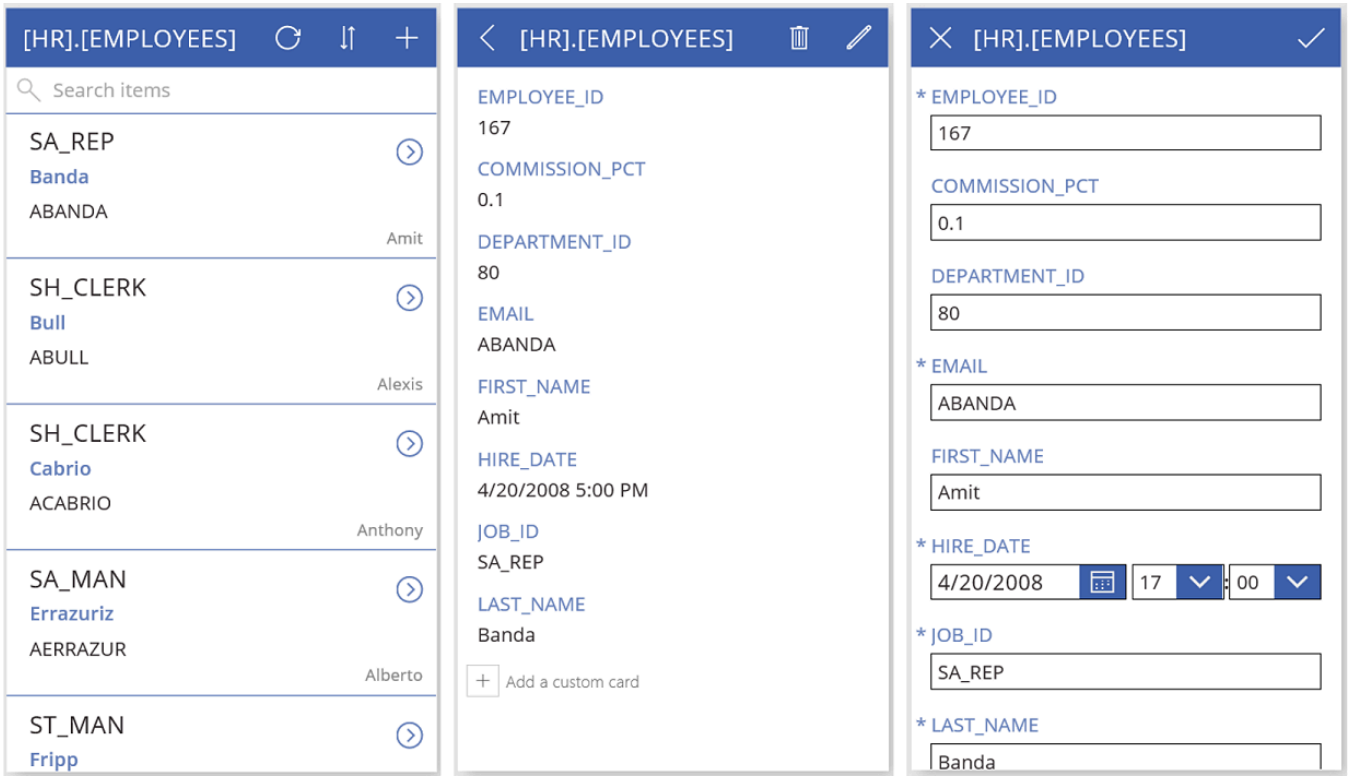


11. Select **Connect** to create the app.



Power Apps creates an app that has three screens and shows data from the table that you selected:

- **BrowseScreen1**, which lists all entries in the table.
- **DetailScreen1**, which provides more info about a single entry.
- **EditScreen1**, in which users can update an entry or create an entry.



Next steps

- To save the app that you've generated, press Ctrl-S.
- To customize **BrowseScreen1** (which appears by default), see Customize a layout.
- To customize **DetailsScreen1** or **EditScreen1**, see Customize a form.

Known issues, tips, and troubleshooting

1. Can't reach the Gateway.

   This error appears if the on-premises data gateway can't connect to the cloud. To check the status of your gateway, sign in to powerapps.microsoft.com, select **Gateways**, and then select the gateway that you want to use.

   Make sure that your gateway is running and can connect to the Internet. Avoid installing the gateway on a computer that may be turned off or asleep. Also try restarting the on-premises data gateway service (PBIEgwService).

2. System.Data.OracleClient requires Oracle client software version 8.1.7 or greater.

   This error appears if the Oracle client SDK isn't installed on the same computer as the on-premises data gateway. To resolve this issue, install the official provider.

3. Table '[Tablename]' doesn't define any key columns.

This error appears if you're connecting to a table that doesn't have a primary key, which the Oracle Database connection requires.

4. Stored procedures, tables with composite keys, and nested object types in tables aren't directly supported in Power Apps. However, stored procedures using Power Automate are supported.

5. Schema name dependency across environments.

   When connecting Power Apps to Oracle on-premises data sources, the schema prefix is treated as part of the table name. This means that the schema name must be exactly the same in every environment such as development, preproduction, and production. If you use different Oracle usernames or schemas in each environment, your app may not be able to find the tables during deployment—even if you update the connection references correctly. This will result in an **Entity Not Found** error.

   This limitation is by design: Power Apps does not currently support dynamic schema name substitution using environment variables. To prevent deployment failures, you should use the same Oracle schema name in all environments. Alternatively, you can consider solutions like Power Automate flows or Dataverse virtual entities, which provide more flexibility for managing environment-specific configurations.

Connect to Power BI from Power Apps

Article • 12/16/2022



Power BI is a suite of business analytics tools to analyze data and share insights. Monitor your business and get answers quickly with rich dashboards available on every device. In your app, you can check the status of the data alerts that you have set up in the Power BI service. For more information on data alerts in Power BI, head to the documentation page.

This topic shows you how to use the Power BI connection in an app, and lists the available functions.

> ⓘ **Note**
>
> The Power BI connection is not **delegable**.

Prerequisites

- Sign up
- Add the Power BI connection
- Create an app from a template, from data, or from scratch

Use the Power BI connection in your app

List the alerts that you've set up in the Power BI service

1. On the **Insert** menu, select **Gallery**, and add any of the **Text galleries**.
2. To show the current user's alerts, set the Items property of the gallery to the following formula:

   ```
   PowerBI.GetAlerts()
   ```

The gallery will update with the list of alerts. For each alert, you will receive the alert name, the ID number of the alert, and the ID of the group workspace in which the alert

was configured. You will need the alert ID to get further information about the alert.

View the status of an alert

To view the status of the alert, call the CheckAlertStatus function with the alert ID obtained from the step above.

The alert ID can be passed in either as a literal string (e.g. "1234") or as a reference to a gallery section populated using the GetAlerts() call (e.g. Gallery1.Selected.alertId)

To proceed, add a label, and then set its Text property to one of these formulas:

- `PowerBI.CheckAlertStatus( /* alert ID that you received from GetAlert */ ).alertTitle`
- `PowerBI.CheckAlertStatus( /* alert ID that you received from GetAlert */ ).currentTileValue`
- `PowerBI.CheckAlertStatus( /* alert ID that you received from GetAlert */ ).alertThreshold`
- `PowerBI.CheckAlertStatus( /* alert ID that you received from GetAlert */ ).isAlertTriggered`

The label will update with the current status of the alert.

View the available functions

This connection includes the following functions:

Function Name	Description
GetAlerts	List the alerts that you have set up in the Power BI service
CheckAlertStatus	Check the status of a particular alert

GetAlerts

List the alerts that you have set up in the Power BI service.

Input properties

None.

Output properties

Property Name	Data Type	Required	Description
value	array	No	An array of the data alerts that you have set up in the Power BI service. Each element in the array will include: • alertTitle: the title of the alert • alertId: the ID of the alert • groupId: the ID of the group that the alert was created in

CheckAlertStatus

Check the status of an alert.

> ⓘ Note
>
> Requests to this endpoint will be throttled on a per-alert basis if called too frequently.

Input properties

Property Name	Data Type	Required	Description
alertId	integer	Yes	The ID of the alert, as returned by GetAlerts

Output properties

Property Name	Data Type	Required	Description
tileValue	number	No	The value of the tile when the alert was triggered
tileUrl	string	No	URL for the tile that has the alert
alertTitle	string	No	Name of the alert
isAlertTriggered	boolean	No	Whether the alert is currently triggered
alertThreshold	number	No	The threshold at which the alarm is triggered

Helpful links

See all the available connections.

Learn how to add connections to your apps.

Connect to SharePoint from a canvas app

Article • 03/14/2025



There are different ways to connect to SharePoint in a canvas app:

- Connect to a SharePoint site to generate an app automatically from a list.
- Create a SharePoint connection before you add data to a new or existing app.

> **Note**
>
> When you create or view a list in SharePoint, you're automatically redirected to Microsoft Lists. The list can always be found in both Microsoft Lists and SharePoint. Learn more in What is a list in Microsoft 365?.

Connect SharePoint to an app by creating an app

If you have a SharePoint site with a SharePoint list, you can create an app in Power Apps through the list menu in Microsoft Lists.

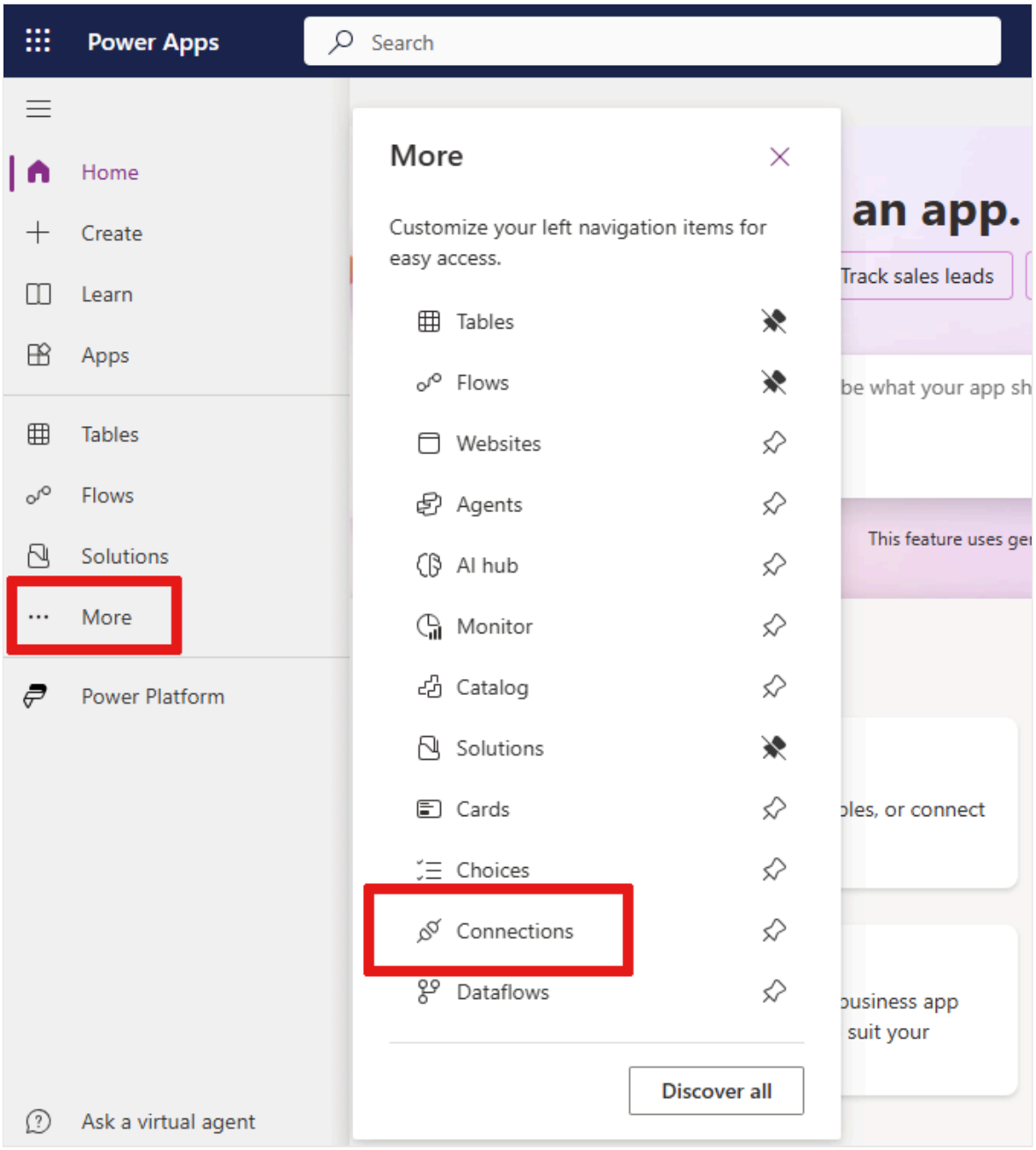
The **Integrate** > **Power Apps** > **Create an app** menu in Microsoft Lists takes you to Power Apps and builds an app using your list as a data source.

Learn more in Create a canvas app with data from Microsoft Lists.

Create a SharePoint connection

If you build a new app or have an existing app, you can connect to SharePoint through the **Data** menu in Power Apps.

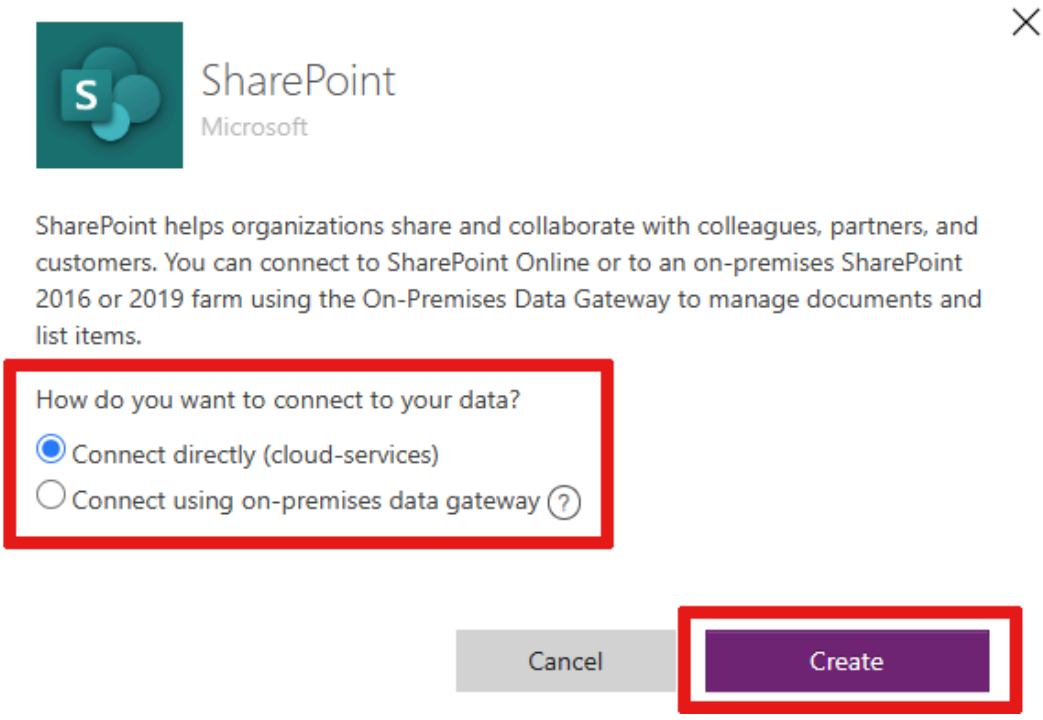
1. Sign in to Power Apps.
2. Select **More** > **Connections** in the left navigation bar.



3. Select + **New connection** and choose **SharePoint**. A popup appears.

4. Choose either the cloud-services or on-premises option:

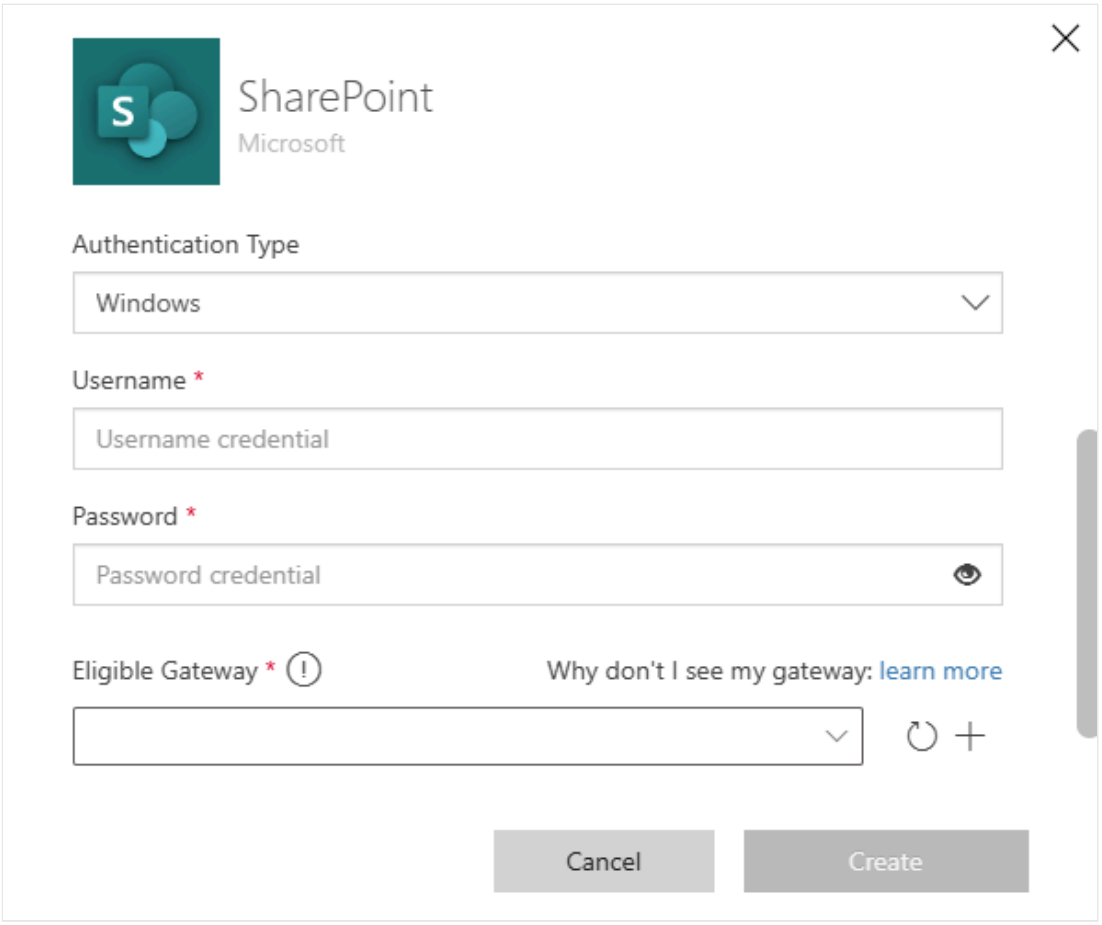
    - To connect to SharePoint Online, select **Connect directly (cloud services)** and then **Create**. Provide credentials (if prompted).



The connection is created, and you can add data to an existing app or build an app from scratch.

- To connect to an on-premises site, select **Connect using on-premises data gateway**.

  Choose **Windows** as the authentication type and add your credentials. If your credentials include a domain name, specify it as `domain\alias`.



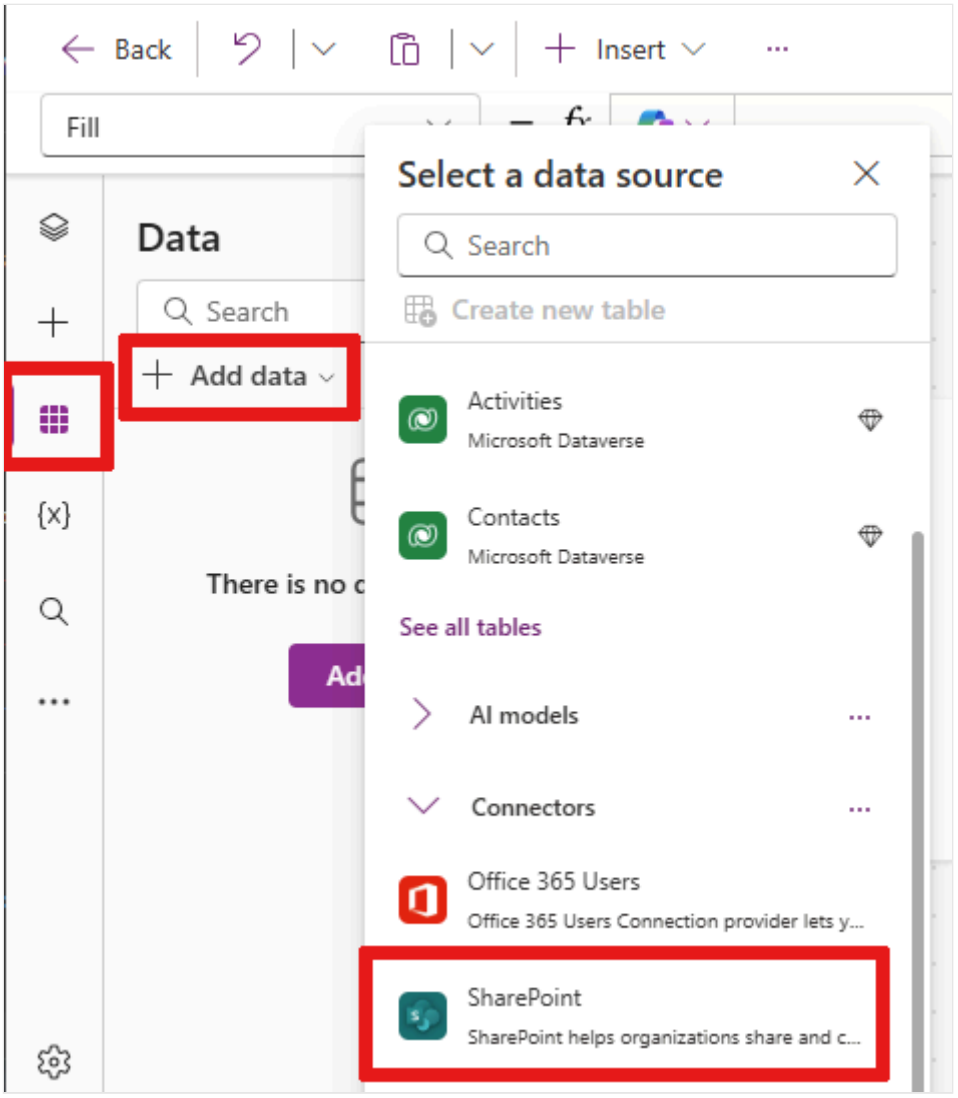
Under **Choose a gateway**, select the gateway that you want to use, and then select **Create**.

> **Note**
>
> If you don't have an on-premises data gateway installed, install one, and then select the refresh icon for a list of gateways.

The connection is created. You can add data to an existing app or build an app from scratch.

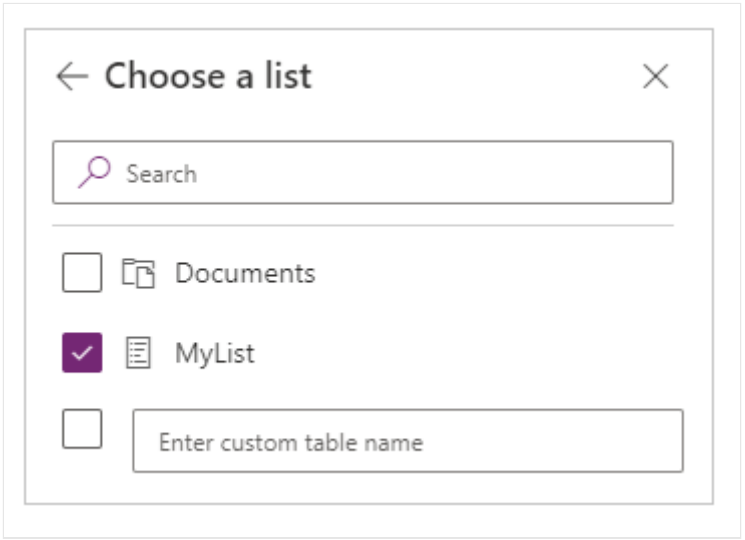
Add data to an existing app

1. In Power Apps Studio, open the app that you want to update, select **Data** on the left-pane.
2. Select **Add data** > **Connectors** > **SharePoint**.



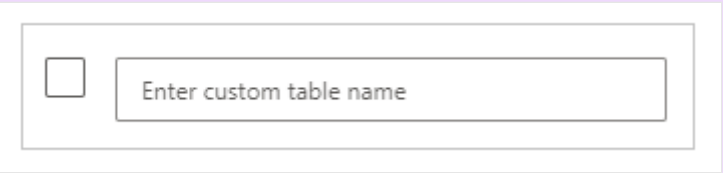
3. A side panel opens. Under **Connect to a SharePoint site**, select an entry in the **Recent sites** list (or add your own URL), and then select **Connect**.

4. Under **Choose a list**, select the check box for one or more lists, then select **Connect**:



> ⓘ Note

Power Apps supports lists, not template-based lists. If you don't see your list, type the list name in **Enter custom table name** box.



You see a confirmation banner that your data source was added to your app.

Power Apps data type mappings

Expand table

Power Apps	SharePoint
`Boolean`	Yes/No (checkbox)
`DateTime`	Date and Time
`Image`	Hyperlink or Picture, Image
`Complex`	Choice, Lookup, Person, Group, Task Outcome, External Data, Managed Metadata
`Number`	Number, Currency, ID
`Text`	Single line of text, Multiple lines of text, Calculated

> **Note**
>
> SharePoint types that map to Power Apps as complex often have subfields that map to basic types such as text and number.

Power Apps delegable functions and operations for SharePoint

The following Power Apps operations, for a given data type, may be delegated to SharePoint for processing (rather than processing locally within Power Apps).

Expand table

Operation/Function [1]	Number	Text	Boolean	DateTime	Complex [2]
`<, <=,<>, >, >=`	Yes [3]	No	No	Yes	Yes
`=`	Yes	Yes	Yes	Yes	Yes
`Filter`	Yes	Yes	Yes	Yes	Yes
`IsBlank`	-	No [4]	-	-	No
`Lookup`	Yes	Yes	Yes	Yes	Yes
`Sort`	Yes	Yes	Yes	Yes	No
`SortByColumns`	Yes	Yes	Yes	Yes	No
`StartsWith`	-	Yes	-	-	Yes
`UpdateIf/RemoveIf` [5]	Yes	No	-	-	No

Notes

1. Expressions that are joined with **And** or **Or** are delegable to SharePoint. **Not** won't delegate. SharePoint system fields don't generally delegate. These fields include:

- Identifier
- IsFolder
- Thumbnail
- Link
- Name
- FilenameWithExtension
- Path
- FullPath
- ModerationStatus
- ModerationComment
- ContentType
- IsCheckedOut
- VersionNumber
- TriggerWindowStartToken
- TriggerWindowEndToken

2. SharePoint supports delegation of complex types by deferring the decision for delegation to the subfield involved. Check the type of the subfield being used on the complex type and then check this table for delegation capabilities. Only Email and DisplayName are delegable in the Person data type.

3. SharePoint ID fields are shown as a number field in Power Apps. However, the underlying type is actually Text. SharePoint only supports the equal ('=') operation for delegation on an ID field. Relational operations such as `<, <=,<>, >, >=` won't work on a SharePoint ID field.
4. A formula such as Filter(..., IsBlank(CustomerId)) won't delegate to SharePoint. However, that formula is semantically close to Filter(..., CustomerId = Blank()), will delegate to SharePoint. These formulas aren't equivalent because the second formula won't treat the empty string ("") as empty. However, the second formula might work for your purposes. On SharePoint, this approach works for the 'equals' operator ("=") but not the operator for 'not equals' ("<>").
5. UpdateIf and RemoveIf work locally but simulate delegation to a limit of 500/2000 records. They successively bring down records beyond the non-delegation 500/2000 record limit. Records that meet the If condition are collected. Generally, a maximum of 500/2000 records are collected separately and then changed per execution. However, more records may be updated if the existing local data cache is large as the function may have access to more records for evaluation.

Related information

- Learn more about working with lists in Power Apps in Integrate SharePoint Online into Power Apps overview
- Move SharePoint Custom Forms with Power Apps (white paper)

Feedback

Was this page helpful? Yes No

Provide product feedback

Connect to SQL Server from Power Apps overview

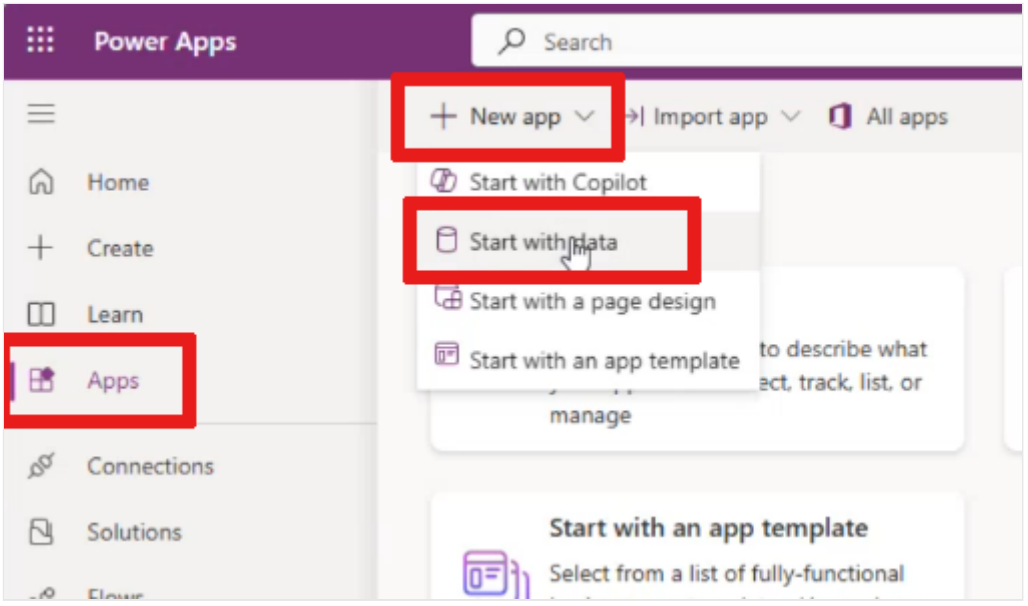
Article • 03/14/2025

There are different ways to connect to data in SQL Server in your app, using Power Fx formulas. You can access data directly or use a view or stored procedures to create, update, or delete data in your app.

Prerequisites

To access data directly, you can create a *Start with data* app for your SQL Server data. This method lets you get a basic, working app that you can modify with views and stored procedures.

1. Sign in to Power Apps and go to the **Apps** page.
2. Select **New app** > **Start with data**.



3. From the **Start with data** page, choose **Connect external data**.
4. Under **Choose a data set to start** select **From SQL**.

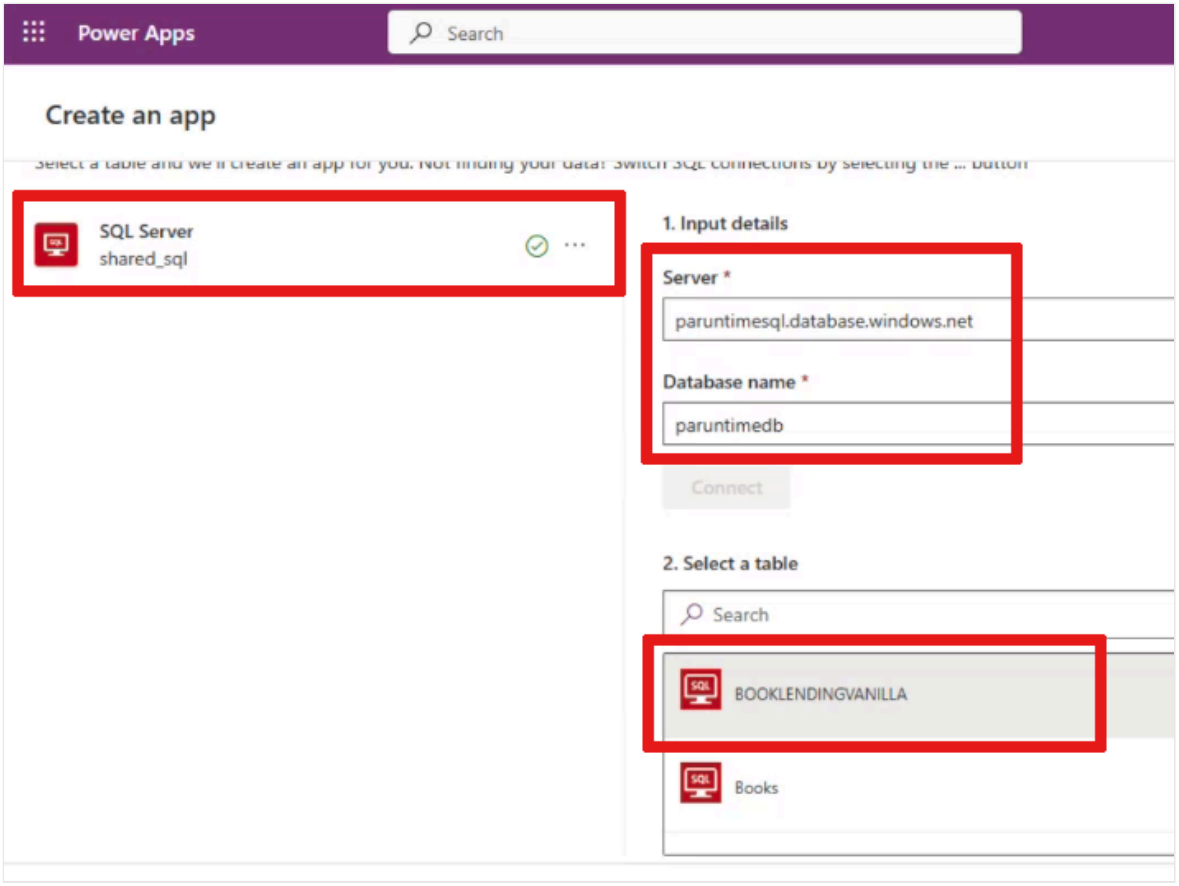
   If you have an existing SQL Server connection, it loads.

> **Note**

If you don't already have a SQL Server connection, you're prompted to create one.

5. Select your SQL connection.

6. Enter your **Server** name and **Database name**, then choose **Connect**. Choose a table from the list of tables that appear.



> **Note**
>
> Only one connection is shown at a time. To use a different connection, select the ... overflow menu on your SQL connection, then find a new one or create a new SQL connection.

7. Select **Create app**.

Access data

Once your app is connected to SQL Server, you can access data: Access data in SQL Server.

View results

To view the results of your SQL queries, see: View results in SQL Server.

Power Apps data type mappings

Expand table

SQL Server	Power Apps
`bigint`, `decimal`, `int`, `money`, `numeric`, `smallint`, `smallmoney`, `tinyint`, `float`, `real`	Number
`char`, `nchar`, `varchar`, `nvarchar`, `text`, `ntext`	Text
`bit`	Boolean
`date`, `datetime`, `datetime2`, `smalldatetime`, `datetimeoffset`, `time`	DateTime
`uniqueidentifier`	Guid

> **Note**
>
> If a SQL Server data type doesn't appear in the previous table, that data type isn't supported. Unsupported data types include the following examples: `binary()`, `varbinary()`, `image`, `cursor`, `rowversion`, `hierarchyid`, `sql_variant xml`, Spatial Geometry Types, Spatial Geography Types, and table. Also, 'tinyint' and 'smallint' are not supported as primary keys.

Power Apps functions and operations delegable to SQL Server

Power Apps can delegate Power Fx query operations (for example, Filter) to the back-end data source for processing. Delegation allows the back-end data service (for example, SQL Server) to do the filtering work so that only a few records must be transferred to your device.

This table shows which operations are delegable to SQL Server by data type. Expressions joined with `And`, `Or`, and `Not` are delegable.

Expand table

Operation or function	Number	Text	Boolean	DateTime	Guid
`*, +, -, /`	Yes	-	-	No	-
`<`, `<=`, `>`, `>=`	Yes	No	No	Yes	-
`=`, `<>`	Yes	Yes	Yes	Yes	Yes
`Average`	Yes	-	-	-	-
`EndsWith`	-	Yes [1]	-	-	-
`Filter`	Yes	Yes	Yes	Yes [2]	Yes
`In` (substring)	-	Yes [3]	-	-	-
`IsBlank` [4]	No	No	No	No	No
`Len` (length)	-	Yes [5]	-	-	-
`Lookup`	Yes	Yes	Yes	Yes	Yes
`Max`	Yes	-	-	No	-
`Min`	Yes	-	-	No	-
`Search`	No	Yes	No	No	-
`Sort`	Yes	Yes	Yes	Yes	-
`SortByColumns`	Yes	Yes	Yes	Yes	-
`StartsWith`	-	Yes [6]	-	-	-
`Sum`	Yes	-	-	-	-
`UpdateIf, RemoveIf` [7]	Yes	Yes	Yes	Yes	Yes

Notes

1. Supported for `(EndsWith(\<column>, "string value"))` but not for `(EndsWith("string value", \<column>))`. If a char(10) column has a value of "hello", `EndsWith(\<column>, "llo")` returns false, by design. A char(10) column has 10 characters.

2. Direct date filters don't work for SQL Server with an on premise Data Gateway. However, you can create a calculated column that works. For example, you can

create:
`ALTER TABLE myTable ADD DateAsInt AS (YEAR([date]) * 10000 + MONTH([date]) * 100 + DAY([date]))`
And then filter on the calculated number column.

3. Supported for `("string value" in \<column>)`, but not for `(\<column> in "string value")`.

4. An expression such as `Filter('[dbo].[MyOrders]', !IsBlank(CustomerId))` won't delegate to the server. However, you can use an expression such as `Filter('[dbo].[MyOrders]', CustomerId <> Blank())`, which does delegate to the server and is semantically close. The difference is that the second expression won't treat the empty string ("") as empty. Although the expressions aren't equivalent, the latter might work for your purposes. You can't use this method for the Guid data type.

5. Power Apps delegates the `Len` function, but the behavior might not work as you expect. In SQL Server, a char(10) column with value "hello" always has a length of 10. However, Power Apps treats that string as having a length of 5, which may cause discrepancies and confusion. Don't use `char`/`nchar` on SQL server, but use `varchar`/`nvarchar` instead.

6. Supported for `(StartsWith(\<column>, "string value"))`, but not for `(StartsWith("string value", \<column>))`

7. UpdateIf and RemoveIf work locally but simulate delegation to a limit of 500/2000 records. They successively bring down records beyond the non-delegation 500/2000 record limit. Records that meet the If condition are collected. Generally, a maximum of 500/2000 records are collected separately and then changed per execution. However, more records may be updated if the existing local data cache is large as the function may have access to more records for evaluation.

Related information

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Microsoft SQL Server securely with Power Apps

Article • 05/21/2025

There are different ways to connect and authenticate to SQL Server with Power Apps. This article outlines concepts that can be helpful in making a choice about how to connect to SQL Server with a security approach that matches requirements for your app.

> **Important**
>
> The secure implicit connections feature was released in January 2024. Microsoft strongly encourages all apps currently using implicit connections to convert to secure implicit connections and to revoke connections shared with end users.

Difference between explicit, implicit, and secure implicit connections

A connection to SQL Server is created whenever you create an app using Power Apps connecting to SQL Server. When such apps are published and shared with others, both the app and the connection are deployed to those users. In other words, the app and the connection—both are visible to users the apps is shared with.

The authentication method used for such connections can be **explicit** or **implicit**. We can also say such connection is shared explicitly or implicitly.

- An **explicitly shared connection** means that the end user of the application must authenticate to SQL Server with their own explicit credentials. Usually this authentication happens behind the scenes as part of Microsoft Entra or Windows authentication handshake. The user doesn't even notice when the authentication takes place.
- An **implicitly shared connection** means that the user implicitly uses the credentials of the account that the app maker used to connect and authenticate to the data source during while creating the app. The end user's credentials are **not** used to authenticate. Each time the end user runs the app, they're using the credentials the author created the app with.
- A **secure implicitly shared connection** refers to a scenario where the end user of the app implicitly uses the credentials of the account that the app maker used to connect and authenticate to the data source while creating the app. This means that the end user's own credentials are not used to authenticate. Instead, when the user runs the app, they are using the credentials that the author of the app created it with. It is important to note

that the end user is not provided with direct access to the connection, and the app only allows access to a limited set of actions and tables.

The following four connection authentication types can be used with SQL Server for Power Apps:

Expand table

Authentication Type	Power Apps connection method
Microsoft Entra Integrated	Explicit
SQL Server Authentication	Implicit / Secure Implicit
Windows Authentication	Implicit / Secure Implicit
Windows Authentication (nonshared)	Explicit

Implicit connection sharing risks

All new applications automatically use the new secure implicit connections. However, with apps that use the older 'implicit connections', both the app and its connections are deployed to end users, it means that **end users can author new applications based on those connections**.

When an author uses secure implicit connections, it means that no connection is shared and no end user receives the connection object. This eliminates the risk of an end-user author reusing a connection to create a new app. Instead, the app works with a proxy connection that is aware of the app and only communicates with that specific app. The proxy connection allows limited actions (create, read, update, delete) and access to specific tables in the app that are defined when the app is published. Therefore, only authorized actions and access are granted to the end user.

The older style simple implicit connection actually distributes a connection object to the end user. For example, if you create an app that filters out the data you don't want users to see. But, the filtered out data is present in the database. But you're relying on the filter you configured to ensure the end users won't see certain data.

Again, with older style simple implicit connections, after you deploy the app, end users can use the connection deployed with your app in any new apps they create. In the new apps, users can see the data you filtered out in your application. It's important to use the new secure implicit connections.

> ⓘ **Important**

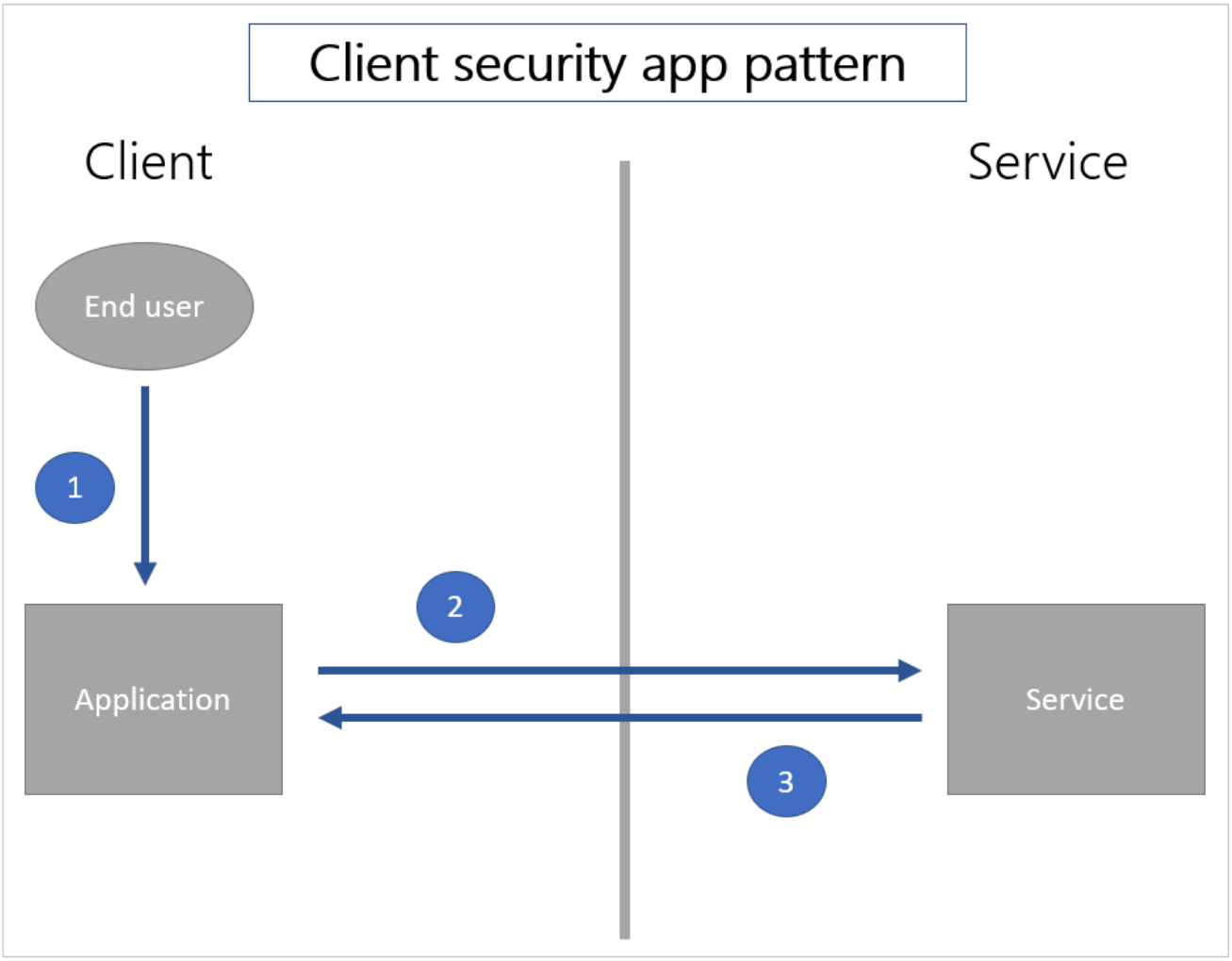
Once an older implicitly shared connection is deployed to end users, the restrictions you may have put in the app you shared (such as filters or read-only access) are no longer valid for new apps end users create. The end users will have whatever rights the authentication allows as part of implicitly shared connection. Therefore, when you convert an app to use secure implicit connections, you must **also** revoke the connections you shared with your app. Admins can get a report of apps with implicitly shared connections with the COE toolkit.

Client and server security

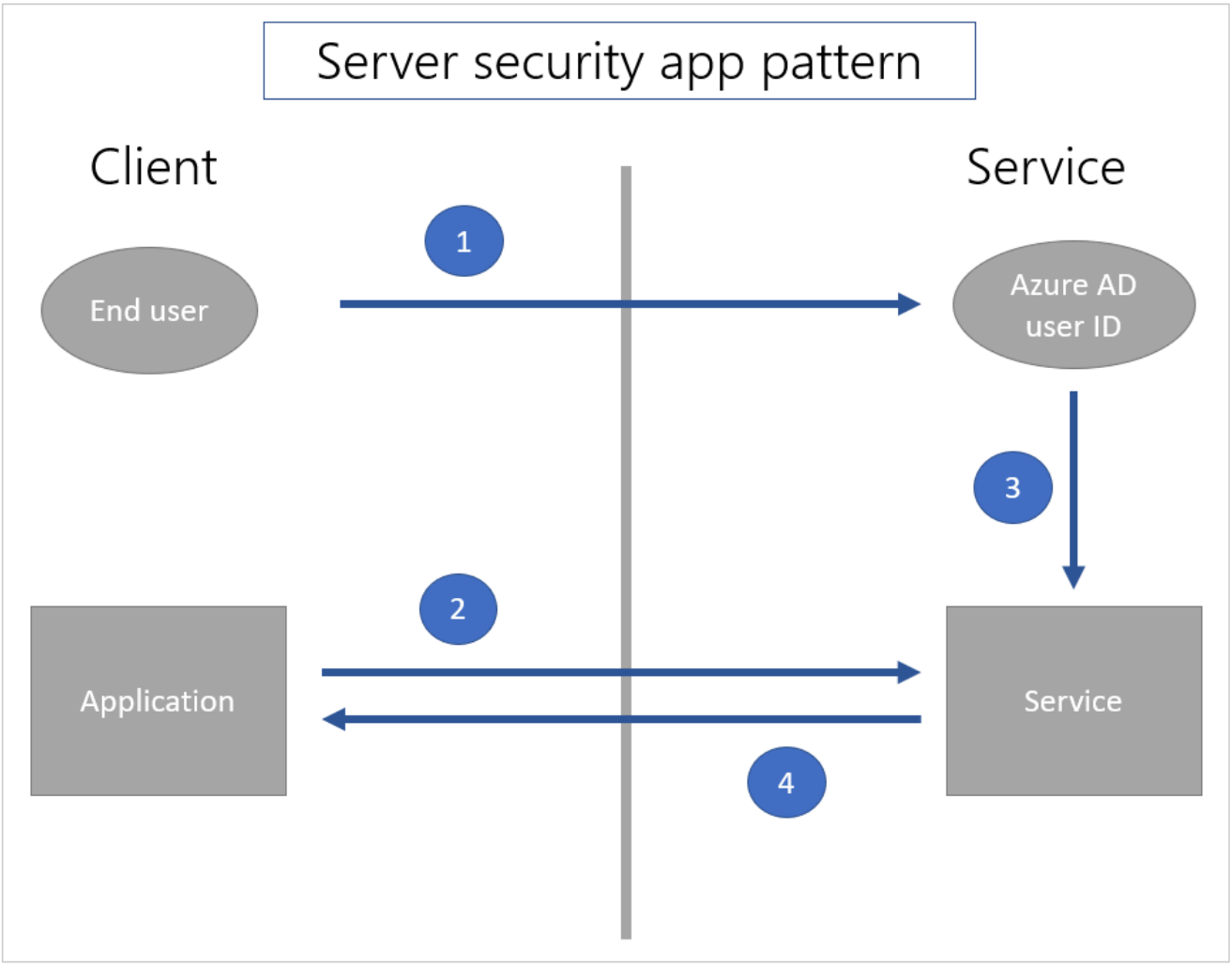
You can't rely on the security of data through filtering or other client-side operations to be secure. Applications that require secure filtering of data must ensure that both the user identification and filtering happens on the server.

Use services such as Microsoft Entra ID instead of relying on the filters designed within the apps when it comes to user identity and security. This configuration ensures server-side filters work as expected.

The following illustrations explain how the security patterns within the apps differ between client-side and server-side security models.



In a client security app pattern, [1] the user only authenticates to the application on the client side. Then [2] the application requests information of the service, and [3] the service returns the information solely based on the data request.



In a server-side security pattern, [1] the user first authenticates to the service so the user is known to the service. Then, [2] when a call is made from the application, the service [3] uses the known identity of the current user to filter the data appropriately and [4] returns the data.

The implicit departmental sharing scenarios described above is combination of these two patterns. The user must log in to the Power Apps service using Microsoft Entra credentials. This behavior is the server security app pattern. The user is known using the Microsoft Entra identity on the service. Hence, the app is restricted to the set of users to which Power Apps has formally shared the application.

However, the implicit shared connection to SQL Server is the client security app pattern. SQL Server only knows that a specific user name and password is used. Any client-side filtering, for instance, can be bypassed with a new application using the same user name and password.

To securely filter data on the server side, use built-in security features in SQL Server such as row level security for rows, and the deny permissions to specific objects (such as columns) to specific users. This approach uses the Microsoft Entra user identity to filter the data on the server.

Some existing corporate services have used an approach where the user identity is captured in a business data layer much in the same way that Microsoft Dataverse does. In this case, the

business layer may or may not use SQL Server's row level security and deny features directly. If it doesn't, it's often the case that the security is enabled using stored procedures or views.

The business layer (on the server side) uses a known user Microsoft Entra identity to invoke a stored procedure as a SQL Server principal and filter the data. However, Power Apps doesn't currently connect to stored procedures. A business layer may also invoke a view that uses the Microsoft Entra identity as a SQL Server principal. In this case, use Power Apps to connect to the views so that the data is filtered on the server-side. Exposing only views to users may need Power Automate flows for updates.

See also

Overview of connectors for canvas apps

Access data in SQL Server

06/21/2025

Add a data source

Direct access:

If you select the *Start with data* option when you create an app, the **Items** property of your gallery uses a Power Fx formula with a data source name that points directly to your database table.

For example, if you have a `BOOKLENDING` table, you see this formula:

Power Fx

```
Search([@'[dbo].[BOOKLENDING]'], SearchInput1.Text, author,
author,book_name,category)
```

Views and stored procedures:

A common professional data access pattern is to use views and then stored procedures for create, update, and delete instead of allowing direct access. If you want to use views or stored procedures, change the example formula. Similarly, the form for the record doesn't use the built-in direct approach of the `SubmitForm()` formula.

Triggers:

One database pattern is to use triggers on tables. If a table has a trigger, you can't use the direct pattern `Submit()` for create, update, and delete. `Submit()` conflicts with SQL trigger handling and the built-in Power Apps behavior, which use the same output parameter.

You can directly access the table for queries, but to handle `Create`, `Update`, or `Delete`, call a stored procedure.

> ⓘ **Note**
>
> The SQL Server connector, like all connectors that work with relational data, assumes that tables have a primary key. A primary key is critical for finding specific records to update. If a SQL Server table doesn't have a primary key, the data is read-only. If you have access and edit rights to the SQL Server table, consider adding an auto-generated key.

Use a view

A *view* is a saved query that shows as a single table of data.

Views appear in the list of tables you can select when you add a data source. Views only support queries—not updates. To update data, use a stored procedure.

If you create a table with the `Start with data` option, you get screens and formulas that show records in a gallery and form. You see formulas and functionality for creating, editing, and deleting records. But if you use a view, you only see a display screen for the gallery and form.

You might want autogenerated screens from `Start with data` for views.

To use this autogenerated option:

1. Choose `Start with data` with a *basic table*.
2. Delete and replace the table data source.

Example:

For example, if you have a `BOOKLENDINGVIEW` table and add it as a data source for Power Apps, the formula can be as simple as:

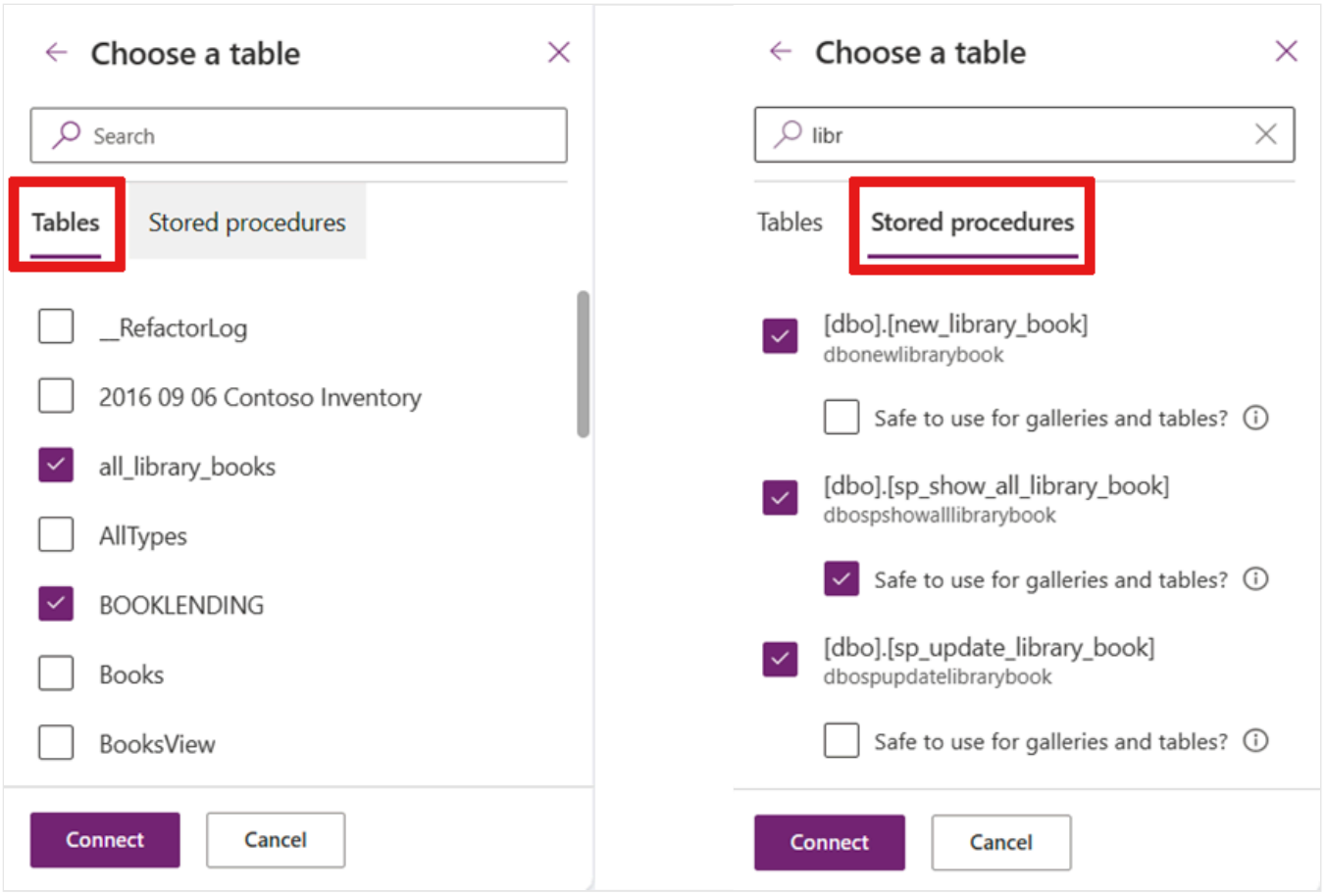
```
Power Fx

BOOKLENDINGVIEW
```

You can also replace other create, update, and delete formulas with a view data source and calls to stored procedures.

Use stored procedures

When you add a SQL Server connection to your app, you can add stored procedures and call them directly in Power Fx.

> **Note**
>
> This feature also works with secure, implicit connections.



After you select a stored procedure, a child node appears, and you can designate the stored procedure as **Safe to use for galleries and tables**.

A stored procedure is *safe* if it doesn't perform any action that might be unwanted in certain scenarios. For example, if a stored procedure collects all accounts from a given city and then sends them an email, you might not always want emails sent every time the stored procedure is called. In that case, don't mark the stored procedure as safe.

Check a stored procedure as safe only if:

1. There are **no side effects** to calling this procedure on demand.

   You can call the procedure multiple times or whenever Power Apps refreshes the control. When you use it with an **Items** property of a gallery or table, Power Apps calls the stored procedure whenever the system determines a refresh is needed. You can't control when the stored procedure is called.

2. The stored procedure returns a modest amount of data.

   Action calls, like stored procedures, don't have a limit on the number of rows retrieved. They aren't automatically paged in 100-record increments like tabular data sources such as tables or views.

   If the stored procedure returns too much data (many thousands of records), your app might slow down or crash. For performance reasons, bring in fewer than 2,000 records.

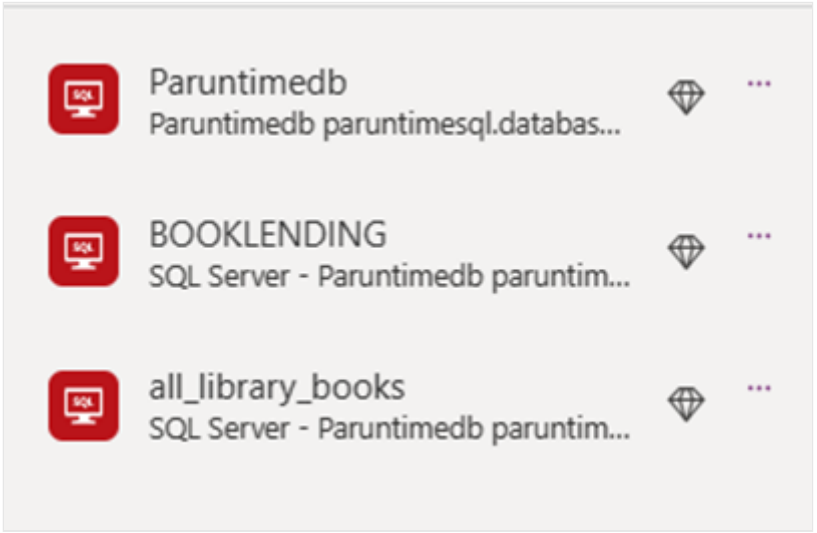
If you check a stored procedure as safe, you can assign it as an **Items** property in galleries or tables in your app.

> **Important**
>
> The schema of the return values of the stored procedure should be *static*, so the values don't change from call to call. For example, if a stored procedure returns two tables, it *always* returns two tables. You can work with either specific or dynamic results.
>
> The structure of the results also needs to be static. For example, if the schema of the results is *dynamic*, then results are dynamic and you must provide a specific type to use them in Power Apps. For more information, see dynamic results.

SQL namespace prepended to stored procedure name

The SQL Server namespace name, where you store the procedure, is added to the start of the stored procedure name. For example, all stored procedures in the '**DBO**' SQL Server namespace have '**dbo**' at the start of the name.

For example, when you add a stored procedure, you might see more than one data source in your project.



Calling a stored procedure

To use a stored procedure in Power Apps, add the connector name before the stored procedure name, such as `Paruntimedb.dbonewlibrarybook`.

> **Note**

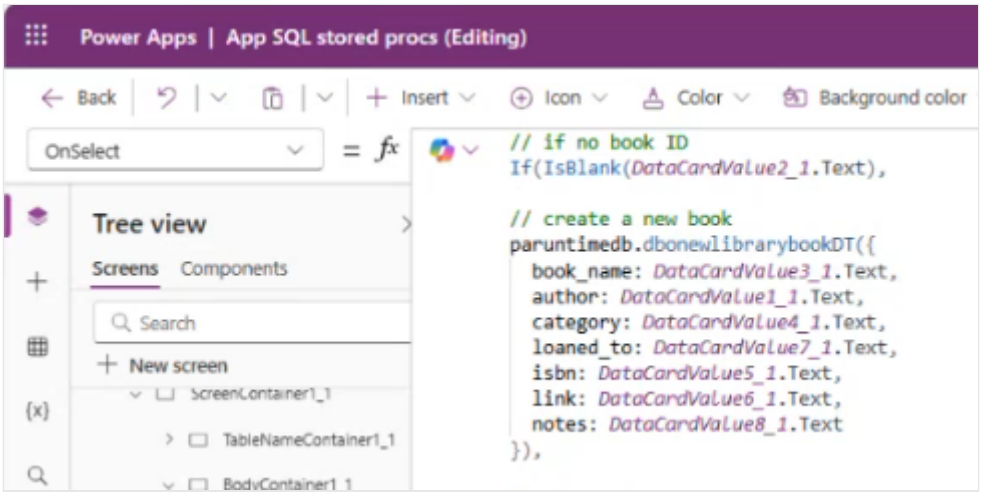
When Power Apps brings in the stored procedure, it combines the namespace and procedure name so that `dbo.newlibrarybook` becomes `dbonewlibrarybook`.

Arguments are passed as a Power Apps record with named value pairs:

```
Power Fx

<datasourceName>.<StoredprocedureName>({<paramName1: value, paramName2: value, ...
>})
```

> **Tip**
>
> Convert values if needed as you pass them into your stored procedure, since you're reading from a text value in Power Apps. For example, if you're updating an integer in SQL, convert the text in the field using `Value()`.

Here's an example of what stored procedures could look like when assigning them to an `OnSelect` property.



Variables and all stored procedures

Access a stored procedure for the **Items** property of a gallery after you declare it safe for the UI. Reference the data source name and the name of the stored procedure followed by `ResultSets`. Access multiple results by referencing the set of tables returned, such as Table 1, Table 2, and so on.

For example, a stored procedure from the table `Paruntimedb` with the name `dbo.spo_show_all_library_books()` looks like this:

```
Power Fx
```

```
Paruntimedb.dbospshowalllibrarybooks().ResultSets.Table1
```

This query populates the gallery with records. However, stored procedures are *action* behaviors on the tabular model. `Refresh()` only works with tabular data sources and doesn't work with stored procedures. Refresh the gallery when a record is created, updated, or deleted.

> **Note**
>
> When you use a `Submit()` on a form for a tabular data source, it effectively calls `Refresh()` under the hood and updates the gallery.

Use a variable to populate and refresh the gallery

Use a variable in the `OnVisible` property for the screen, and set the stored procedure to the variable.

Power Fx

```
Set(SP_Books, Paruntimedb.dbospshowalllibrarybooks().ResultSets.Table1);
```

You can then set the `Items` property of the gallery to the variable name.

Power Fx

```
SP_Books
```

After you create, update, or delete a record with a call to the stored procedure, set the variable again to refresh the gallery.

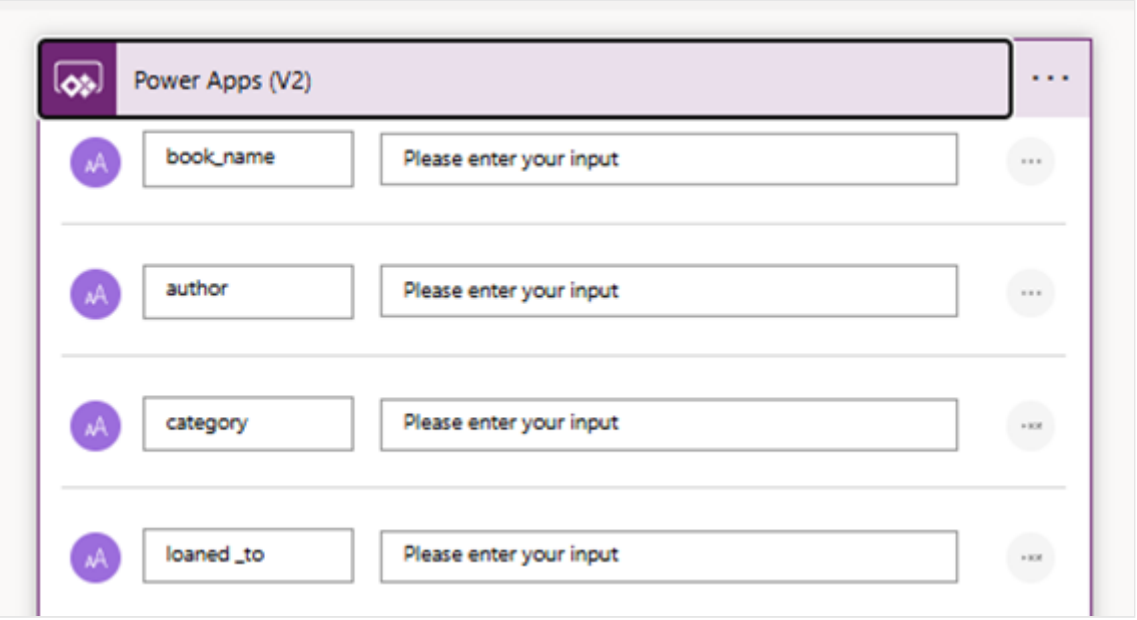
Power Fx

```
Paruntimedb.dbonewlibrarybook({
  book_name: DataCardValue3_2.Text,
  author: DataCardValue1_2.Text,
    ...
});
Set(SP_Books, Paruntimedb.dbospshowalllibrarybooks().ResultSets.Table1);
```

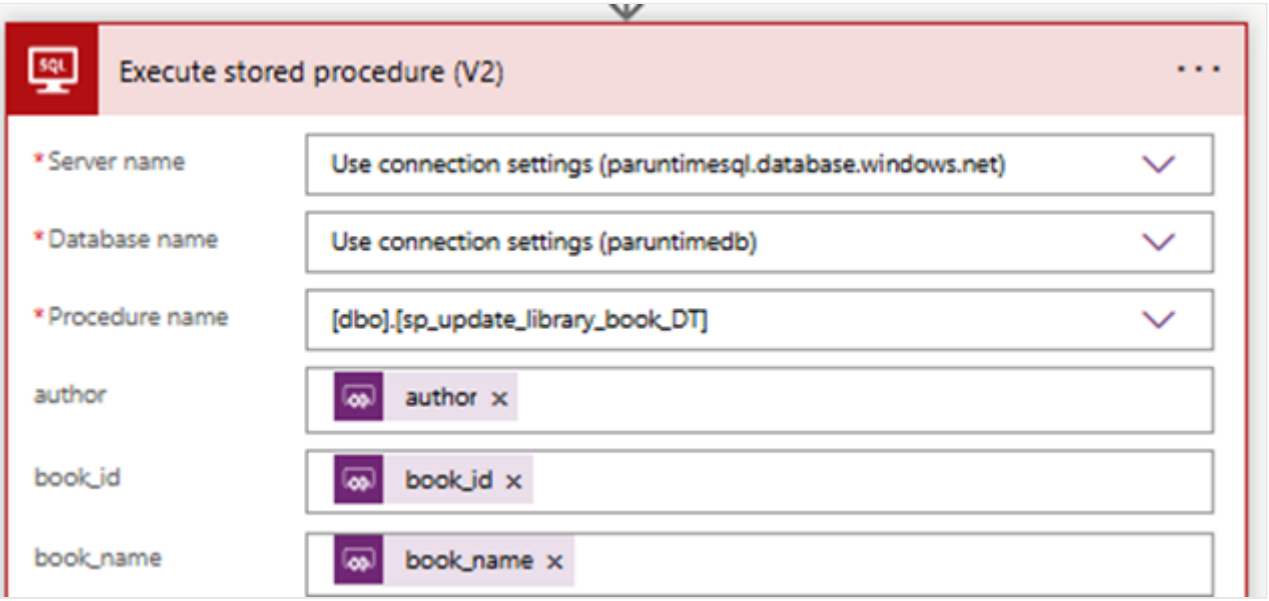
Use Power Automate to call stored procedures

Power Automate handles asynchronous actions best. Call stored procedures as part of a series of calls in a business process.

To call Power Automate and then call stored procedures, create input variables as part of your flow.



Then pass your input variables into the call of your stored procedure.



Add this Power Automate flow to your app and call it. Pass optional arguments as a record "{ ... }". The following example includes all optional arguments.

```
Addanewbook.Run(
{
text:   If(IsBlank(DataCardValue3_6.Text), "", DataCardValue3_6.Text),  // book_name
text_1: If(IsBlank(DataCardValue1_6.Text), "", DataCardValue1_6.Text),  // author
text_2: If(IsBlank(DataCardValue4_6.Text), "", DataCardValue4_6.Text),  // category
text_3: If(IsBlank(DataCardValue7_6.Text), "", DataCardValue7_6.Text),  // loaned_to
```

View results in SQL Server

06/21/2025

If you use a direct table access pattern or a view, the query result binds to the control or table. Power Fx automatically lets your app page data into the gallery or table. However, stored procedures can return a query result, a return code, or values from `Out` parameters.

To use these different result types in your app, follow these patterns.

Formulas for different controls

Here are typical formulas for views and stored procedures:

Expand table

Control	Property	Formula	Description
Gallery or Table	Items	`DataSource`	You can further refine the table or view data source with a Filter and a StartsWith. The other generated query clauses are appended to the existing query.
Form	DataSource	`DataSource`	The table or view data source.
Submit button on a form	OnSelect	`DataSource.dboSPName({ args}); Refresh ('DataSource')`	The first `DataSource` in this formula is the stored procedure data source, which holds your stored procedure. The `DataSource` in the refresh formula is the view data source.
Delete button on a form	OnSelect	`SP DataSource.dboSPName({ args}); Refresh ('View DataSource')`	The first `DataSource` in this formula is the stored procedure data source, which holds your stored procedure. The `DataSource` in the refresh formula is the view data source.

Return code

Use this return code to get the result of a return statement.

Power Fx

```
<datasourceName>.<StoredprocedureName>({<paramName1: value, paramName2: value, ... >}).ReturnCode
```

Output parameters

Use the parameter name as it appears in the JSON payload.

```
Power Fx

<datasourceName>.<StoredprocedureName>({<paramName1: value, paramName2: value, ...
>}).OutputParameters.<parameterName>
```

Result Sets

You can use other tables by their name, like `Table1`, `Table2`, or `Table3`.

```
Power Fx

<datasourceName>.<StoredprocedureName>({<paramName1: value, paramName2: value, ...
>}).ResultSets.Table1
```

Dynamic results

Some complicated stored procedures return dynamic results. This result is common for stored procedures that use temporary tables. Power Apps can't easily determine the results ahead of time. So, the return is marked as *dynamic* and you can't access these results directly. First, provide a type.

You can access the data with the following data access example pattern.

Data access example

1. Pull the results into a variable named `MyDynamicValue`.
2. Pull `Table1` from that variable and put it into a variable named `table1`.

   > **Tip**
   >
   > This step isn't strictly necessary. It's useful, though, to put all the results in a variable and then pull out the parts you need later.

3. Iterate through `table1` and extract the JSON elements as named value pairs.
4. Match the names with those returned in the JSON payload.

5. To validate, open Power Apps monitor and look at the body section of the data node for a record.

Power Fx

```
Set(
    <MyDynamicValue>, // pull results into variable
    <datasourceName>.<StoredprocedureName>(
      { <paramName1>: "someString" }
    ).ResultSets
);
Set(
    table1, // put Table1 into table1
    <MyDynamicValue>.Table1
);
Set(
    TypedTable,
    ForAll(
        table1, // extract JSON from table1
        {
            BookID: Value(ThisRecord.BookID),
            BookName: Text(ThisRecord.BookName)
        }
    )
);
```

Connect to Twitter from Power Apps

Article • 12/16/2022



Twitter lets you post tweets and get tweets, timeline, friends, and followers from your Twitter account.

You can display this information in a label on your app. For example, you can add an input text box, ask the user to enter in some Tweet text, and then add a button that "posts" the tweet. You can use similar methods to get a tweet or search for a tweet, and then display the text in a label or gallery control in your app.

This topic shows you how to create the Twitter connection, use the Twitter connection in an app, and lists the available functions.

Prerequisites

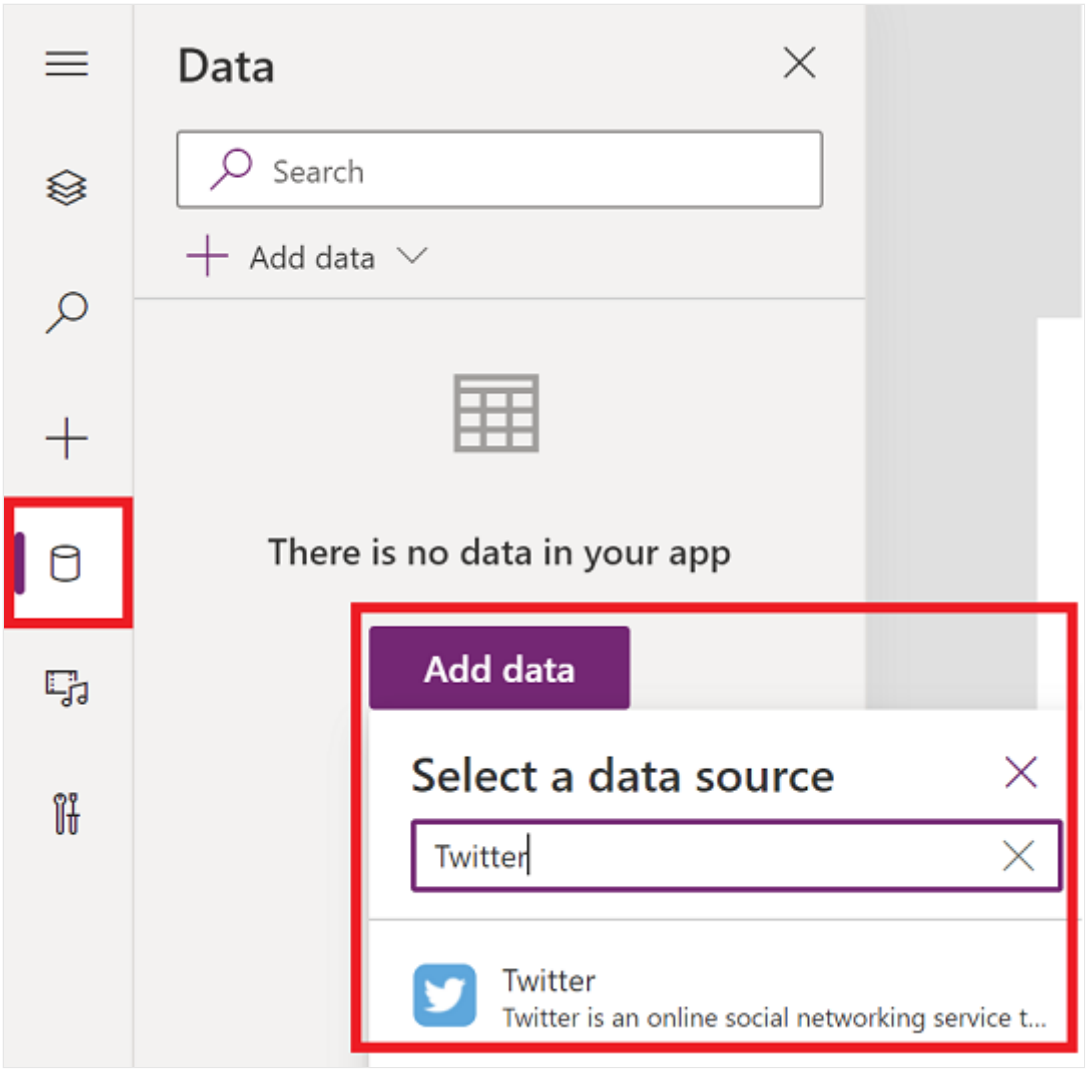
- Access to Power Apps
- Add the connection
- Create an app from a template, from data, or from scratch

Connect to Twitter

1. Start by creating a blank canvas app.

2. In the left-hand pane, select **Data** > **Add data**.

3. Select **New connection**, and then select **Twitter**:

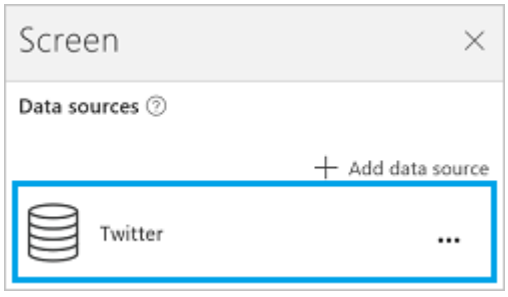


4. Select from the default shared application, or choose to bring your own application (along with Consumer Key/Consumer Secret).

5. Select **Connect**.

6. Choose , enter your Twitter sign in credentials, and then select **Authorize app**.

7. Select **Add Data Source**. Your connection appears under **Data sources**:



The Twitter connection has been created, and added to your app. Now, it's ready to be used.

Use the Twitter connection in your app

Show a timeline

1. On the **Insert** menu, select **Gallery**, and add any of the **With text** galleries.

2. Let's show some timelines:

   - To show the current user's timeline, set the **Items** property of the gallery to the following formulas:

     `Twitter.HomeTimeline().TweetText`

     `Twitter.HomeTimeline({maxResults:3}).TweetText`

   - To show another user's timeline, set the **Items** property of the gallery to the following formula:

     `Twitter.UserTimeline( *TwitterHandle* ).TweetText`

     Enter a Twitter handle in double quotation marks or an equivalent value. For example, enter `"satyanadella"` or `"powerapps"` directly in the formula expression.

   - Add a text input control named **Tweep**, and set its Default property to `Tweep.Text`. In the Tweep text box, type in a Twitter handle such as `satyanadella` (without quotation marks and without the @ symbol).

     In the gallery control, set the Items property to the following formula:

     `Twitter.UserTimeline(Tweep.Text, {maxResults:5}).TweetText`

     The gallery control automatically shows the tweets of the Twitter handler you type in.

     > **Tip**
     >
     > Some of these formulas use the **maxResults** argument to show the *x* number of most recent tweets in a timeline.

3. Set the gallery's **Items** property to `Twitter.HomeTimeline()`.

   With the gallery selected, the right-hand pane shows options for that gallery.

4. Select **TweetText** in the first list, select **TweetedBy** in the second list, and select **CreatedAt** in the third list.

   The gallery now shows the values of the properties you chose.

Show followers

1. Using a **With text** gallery, let's show some followers:

    - To show the current user's followers, set the **Items** property of the gallery to the following formula:

      `Twitter.MyFollowers()`

      `Twitter.MyFollowers({maxResults:3})`

    - To show another user's followers, set the **Items** property of the gallery to the following formula:

      `Twitter.Followers( *TwitterHandle* )`

      Enter a Twitter handle in double quotation marks or an equivalent value. For example, enter `"satyanadella"` or `"powerapps"` directly in the formula expression.

    - Add a text input control named **Tweep**, and set its Default property to `Tweep.Text`. In the Tweep text box, type in a Twitter handle such as `satyanadella` (without quotation marks and without the @ symbol).

      In the gallery control, set the Items property to the following formula:

      `Twitter.Followers(Tweep.Text, {maxResults:5})`

      The gallery control automatically shows who is following the Twitter handle you type in.

      > **Tip**
      >
      > Some of these formulas use the **maxResults** argument to show the *x* number of most recent tweets in a timeline.

2. Set the gallery's **Items** property to `Twitter.MyFollowers()`.

   With the gallery selected, the right-hand pane shows options for that gallery.

3. Select **UserName** in the second list, and select **FullName** in the third list.

   The gallery now shows the values of the properties you chose.

Show followed users

1. Using a **With text** gallery, let's show some followed users:

    - To show which users the current user is following, set the **Items** property of the gallery to the following formula:

      `Twitter.MyFollowing()`

      `Twitter.MyFollowing({maxResults:3})`

    - To show which users another user is following, set the **Items** property of the gallery to the following formula:

      `Twitter.Following( *TwitterHandle* )`

      Enter a Twitter handle in double quotation marks or an equivalent value. For example, enter `"satyanadella"` or `"powerapps"` directly in the formula expression.

    - Add a text input control named **Tweep**, and set its Default property to `Tweep.Text`. In the Tweep text box, type in a Twitter handle such as `satyanadella` (without quotation marks and without the @ symbol).

      In the gallery control, set the Items property to the following formula:

      `Twitter.Following(Tweep.Text, {maxResults:5})`

      The gallery control automatically shows the other handles you are following.

      With the gallery selected, the right-hand pane shows options for that gallery.

2. Select **Description** in the **Body1** list, **UserName** in the **Heading1** list, and **FullName** in the **Subtitle1** list.

    The gallery now shows the values of the properties you chose.

Show information about a user

Add a label, and then set its **Text** property to one of these formulas:

- `twitter.User( *TwitterHandle* ).Description`
- `twitter.User( *TwitterHandle* ).FullName`
- `twitter.User( *TwitterHandle* ).Location`
- `twitter.User( *TwitterHandle* ).UserName`
- `twitter.User( *TwitterHandle* ).FollowersCount`
- `twitter.User( *TwitterHandle* ).FriendsCount`

- `twitter.User( *TwitterHandle* ).Id`
- `twitter.User( *TwitterHandle* ).StatusesCount`

Enter a Twitter handle in double quotation marks or an equivalent value. For example, enter `"satyanadella"` or `"powerapps"` directly in the formula expression.

Or, you can use an input text control to type in a Twitter handle, just as we have throughout this topic.

Search tweets

1. Using a **With text** gallery, set its **Items** property to the following formula:

   `Twitter.SearchTweet( *SearchTerm* ).TweetText`

   Enter a *SearchTerm* in double quotation marks or by referring to an equivalent value. For example, enter `"PowerApps"` or `"microsoft"` directly in the formula.

   Or, you can use an **Input text** control to specify a search term, just as we have throughout this topic.

   > **Tip**
   >
   > Show the first five results by using maxResults:

   `Twitter.SearchTweet(SearchTerm.Text, {maxResults:5}).TweetText`

2. Set the gallery's **Items** property to `Twitter.SearchTweet(SearchTerm.Text, {maxResults:5})`.

   With the gallery selected, the right-hand pane shows options for that gallery.

3. Select **TweetText** in the first list, **TweetedBy** in the second list, and **CreatedAt** in the third list.

   The gallery now shows the values of the properties you chose.

Send a tweet

1. Add a text input control, and then rename it **MyTweet**.
2. Add a button, and then set its **OnSelect** property to the following formula:
   `Twitter.Tweet("",{tweetText: MyTweet.Text})`
   Example: `Twitter.Tweet("",{tweetText:"Sample tweet!"})`

3. Press F5, or select the Preview button (). Type some text into **MyTweet**, and then select the button to tweet the text that you entered.
4. Press Esc to return to the default workspace.

View the available functions

This connection includes the following functions:

Function Name	Description
UserTimeline	Retrieves a collection of the most recent tweets posted by the specified user
HomeTimeline	Retrieves the most recent tweets and re-tweets posted me and my followers
SearchTweet	Retrieves a collection of relevant tweets matching a specified query
Followers	Retrieves users following the specified user
MyFollowers	Retrieves users who are following me
Following	Retrieves users who the specified user is following
MyFollowing	Retrieves users that I am following
User	Retrieves details about the specified user (example: user name, description, followers count, etc.)
Tweet	Tweet
OnNewTweet	Triggers a workflow when a new tweet is posted which matches your search query

UserTimeline

Get user timeline: Retrieves a collection of the most recent tweets posted by the specified user

Input properties

Name	Data Type	Required	Description
userName	string	yes	Twitter handle
maxResults	integer	no	Maximum number of tweets to retrieve, e.g. {maxResults:5}

Output properties

Property Name	Data Type	Required	Description
TweetText	string	Yes	
TweetId	string	No	
CreatedAt	string	No	
RetweetCount	integer	Yes	
TweetedBy	string	Yes	
MediaUrls	array	No	

HomeTimeline

Get home timeline: Retrieves the most recent tweets and re-tweets posted me and my followers

Input properties

Name	Data Type	Required	Description
maxResults	integer	no	Maximum number of tweets to retrieve, e.g. {maxResults:5}

Output properties

Property Name	Data Type	Required	Description
TweetText	string	Yes	
TweetId	string	No	
CreatedAt	string	No	
RetweetCount	integer	Yes	
TweetedBy	string	Yes	
MediaUrls	array	No	

SearchTweet

Search tweet: Retrieves a collection of relevant tweets matching a specified query

Input properties

Name	Data Type	Required	Description
searchQuery	string	yes	Query text (you may use any Twitter supported query operators: https://www.twitter.com/search)
maxResults	integer	no	Maximum number of tweets to retrieve, e.g. {maxResults:5}

Output properties

Property Name	Data Type	Required	Description
TweetText	string	Yes	
TweetId	string	No	
CreatedAt	string	No	
RetweetCount	integer	Yes	
TweetedBy	string	Yes	
MediaUrls	array	No	

Followers

Get followers: Retrieves users following the specified user

Input properties

Name	Data Type	Required	Description
userName	string	yes	Twitter handle of the user
maxResults	integer	no	Maximum number of users to retrieve, e.g. {maxResults:5}

Output properties

Property Name	Data Type	Required	Description
FullName	string	Yes	
Location	string	Yes	

Property Name	Data Type	Required	Description
Id	integer	No	
UserName	string	Yes	
FollowersCount	integer	No	
Description	string	Yes	
StatusesCount	integer	No	
FriendsCount	integer	No	

MyFollowers

Get my followers: Retrieves users who are following me

Input properties

Name	Data Type	Required	Description
maxResults	integer	no	Maximum number of users to retrieve, e.g. {maxResults:5}

Output properties

Property Name	Data Type	Required	Description
FullName	string	Yes	
Location	string	Yes	
Id	integer	No	
UserName	string	Yes	
FollowersCount	integer	No	
Description	string	Yes	
StatusesCount	integer	No	
FriendsCount	integer	No	

Following

Get following: Retrieves users who the specified user is following

Input properties

Name	Data Type	Required	Description
userName	string	yes	Twitter handle of the user
maxResults	integer	no	Maximum number of users to retrieve, e.g. {maxResults:5}

Output properties

Property Name	Data Type	Required	Description
FullName	string	Yes	
Location	string	Yes	
Id	integer	No	
UserName	string	Yes	
FollowersCount	integer	No	
Description	string	Yes	
StatusesCount	integer	No	
FriendsCount	integer	No	

MyFollowing

Get my following: Retrieves users that I am following

Input properties

Name	Data Type	Required	Description
maxResults	integer	no	Maximum number of users to retrieve, e.g. {maxResults:5}

Output properties

Property Name	Data Type	Required	Description

Property Name	Data Type	Required	Description
FullName	string	Yes	
Location	string	Yes	
Id	integer	No	
UserName	string	Yes	
FollowersCount	integer	No	
Description	string	Yes	
StatusesCount	integer	No	
FriendsCount	integer	No	

User

Get user: Retrieves details about the specified user (example: user name, description, followers count, etc.)

Input properties

Name	Data Type	Required	Description
userName	string	yes	Twitter handle of the user

Output properties

Property Name	Data Type	Required	Description
FullName	string	Yes	
Location	string	Yes	
Id	integer	No	
UserName	string	Yes	
FollowersCount	integer	No	
Description	string	Yes	
StatusesCount	integer	No	

Property Name	Data Type	Required	Description
FriendsCount	integer	No	

Tweet

Post a new tweet: Tweet

Input properties

Name	Data Type	Required	Description
tweetText	string	no	Text to be posted e.g. {tweetText:"hello"}
body	string	no	Media to be posted

Output properties

Property Name	Data Type	Required	Description
TweetId	string	Yes	

OnNewTweet

When a new tweet appears: Triggers a workflow when a new tweet is posted which matches your search query

Input properties

Name	Data Type	Required	Description
searchQuery	string	yes	Query text (you may use any Twitter supported query operators: https://www.twitter.com/search)

Output properties

Property Name	Data Type	Required	Description
value	array	No	

Helpful links

See all the available connections.

Learn how to add connections to your apps.

Add data connections to canvas apps

Article • 10/04/2023

In Power Apps, add a data connection to an existing canvas app or to an app that you're building from scratch. Your app can connect to SharePoint, Microsoft Dataverse, Salesforce, OneDrive, or many other data sources.

Your next step after this article is to display and manage data from that data source in your app, as in these examples:

- Connect to OneDrive, and manage data in an Excel workbook in your app.
- Connect to Twilio, and send an SMS message from your app.
- Connect to Dataverse, and update a table from your app.
- Connect to SQL Server, and update a table from your app.

Prerequisites

Sign up for Power Apps, and then sign in by providing the same credentials that you used to sign up.

Add data source

1. Create a blank canvas app.
2. From the app authoring menu in Power Apps Studio, select **Data** > **Add data**.
3. Expand **Connectors**, and if the list of connections includes the one that you want, select it to add it to the app. Otherwise, skip to the next step. To connect to a table in Microsoft Dataverse instead, choose a table from the list of tables.
4. To create a new connection, search for or select a connector such as SharePoint.
5. Select **Add a connection** to both create the connection and add it to your app.

   Some connectors, such as **Office 365 Outlook**, require no additional steps, and you can show data from them immediately. Other connectors prompt you to provide credentials, specify a particular set of data, or perform other steps. For example, SharePoint and SQL Server require additional information before you can use them. With Dataverse, you can change the environment before you select a table.

Identify or change a data source

If you're updating an app, you might need to identify or change the source of data that appears in a gallery, a form, or another control. For example, you might need to identify a data source as you update an app that someone else created or that you created long ago.

1. Select the control, such as a gallery, for which you want to identify or change the data source.

   The name of the data source appears on the **Properties** tab of the right-hand pane.

2. To show more information about the data source or to change it, select the down arrow next to its name.

   More information about the current data source appears, and you can select or create another source.

Next steps

- To show and update data in a source such as Excel, SharePoint, Dataverse, or SQL Server, add a gallery, and add a form.
- For data in other sources, use connector-specific functions such as those for Office 365 Outlook, Twitter, and Microsoft Translator.

Feedback

Was this page helpful? Yes No

Provide product feedback

Manage connections in canvas apps

08/27/2025

In powerapps.com, create a connection to one or more data sources, delete a connection, or update its credentials.

Your canvas app's data connection can connect to SharePoint, SQL Server, Office 365, OneDrive for Business, Salesforce, Excel, and many other data sources.

Your next step after this article is to display and manage data from the data source in your app, as in these examples:

- Connect to OneDrive for Business, and manage data in an Excel workbook in your app.
- Update a list on a SharePoint site.
- Connect to SQL Server, and update a table from your app.
- Send email in Office 365.
- Send a tweet.
- Connect to Twilio, and send an SMS message from your app.

Prerequisites

1. Sign up for Power Apps.
2. Sign in to make.powerapps.com using the same credentials that you used to sign up.

Background on data connections

Most canvas apps use external information called **Data Sources** that is stored in cloud services. A common example is a table in an Excel file stored in OneDrive for Business. Apps are able to access these data sources by using **Connections**.

The most common type of data source is the table, which you can use to retrieve and store information. You can use connections to data sources to read and write data in Microsoft Excel workbooks, Microsoft Lists, SQL tables, and many other formats, which can be stored in cloud services like OneDrive for Business, DropBox, and SQL Server.

There are other kinds of data sources that aren't tables, such as email, calendars, twitter, and notifications.

Using the **Gallery**, **Display form**, and **Edit form** controls, it's easy to create an app that reads and writes data from a data source. To get started, read the article Understand data forms.

In addition to creating and managing connections in powerapps.com, you also create connections when you do these tasks:

- Automatically generate an app from data, such as a list created using Microsoft Lists.
- Update an existing app, or create one from scratch as add a connection describes.
- Open an app that another user created and shared with you.

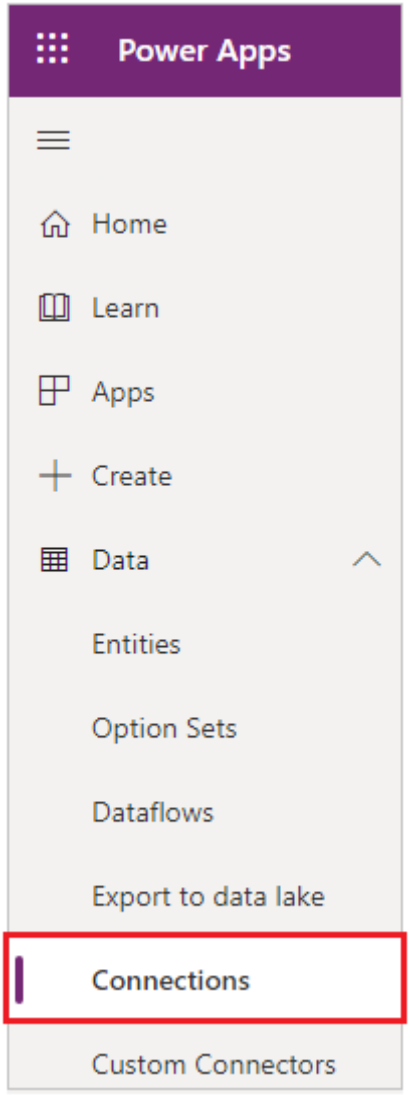
How data connections are used when running the app

When a canvas app runs, Power Apps attempts to locate and reuse an existing connection. If no suitable connection is found, a new one is created for the user based on the connection information stored in the app. This newly created connection may differ from the one originally used during app authoring. This behavior may differ from how other products in the Power Platform handle connections.

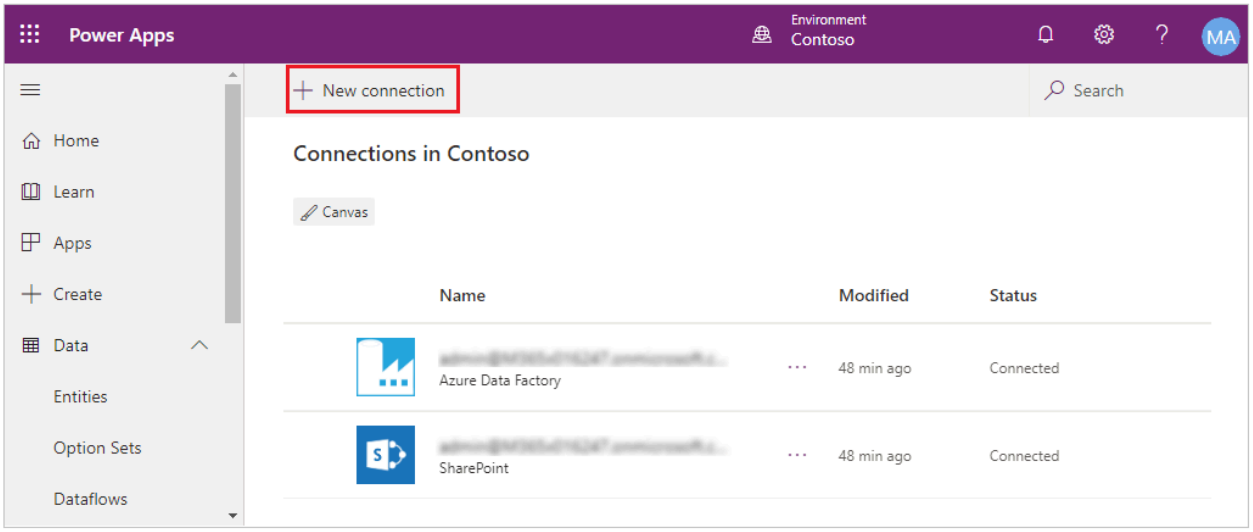
> **Note**
>
> If you want to use Power Apps Studio instead, open the **File** menu, and then click or tap **Connections**, powerapps.com opens so that you can create and manage connections there.

Create a new connection

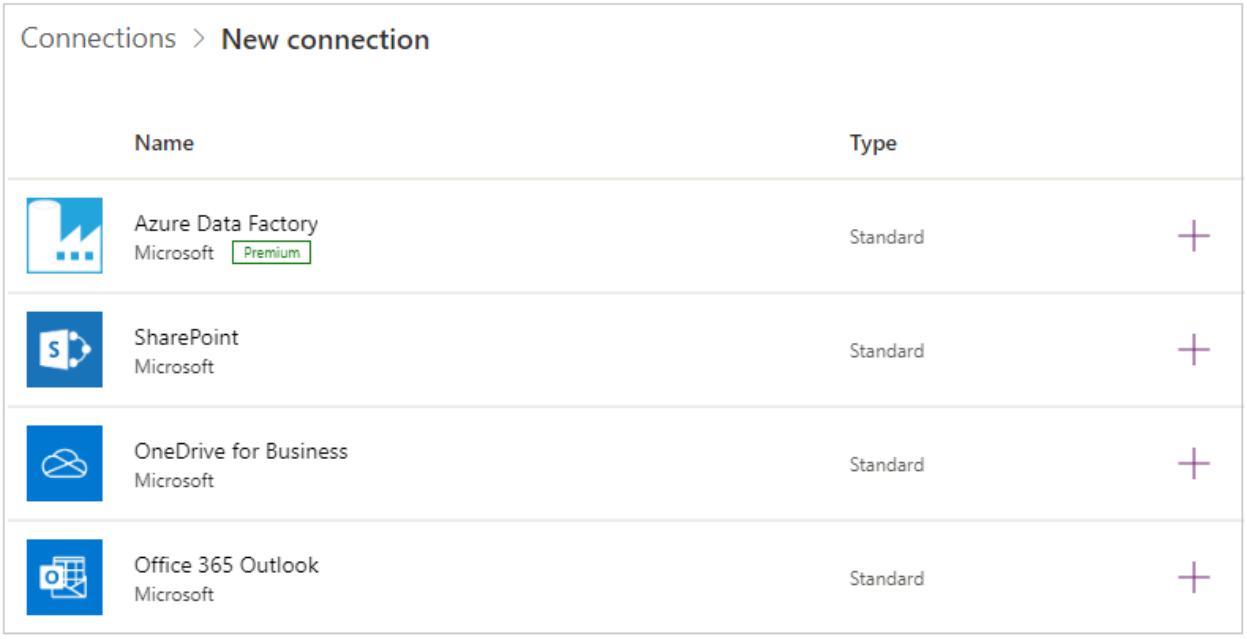
1. If you haven't already done so, log in to make.powerapps.com.
2. In the left navigation, expand **Data** and select **Connections**.



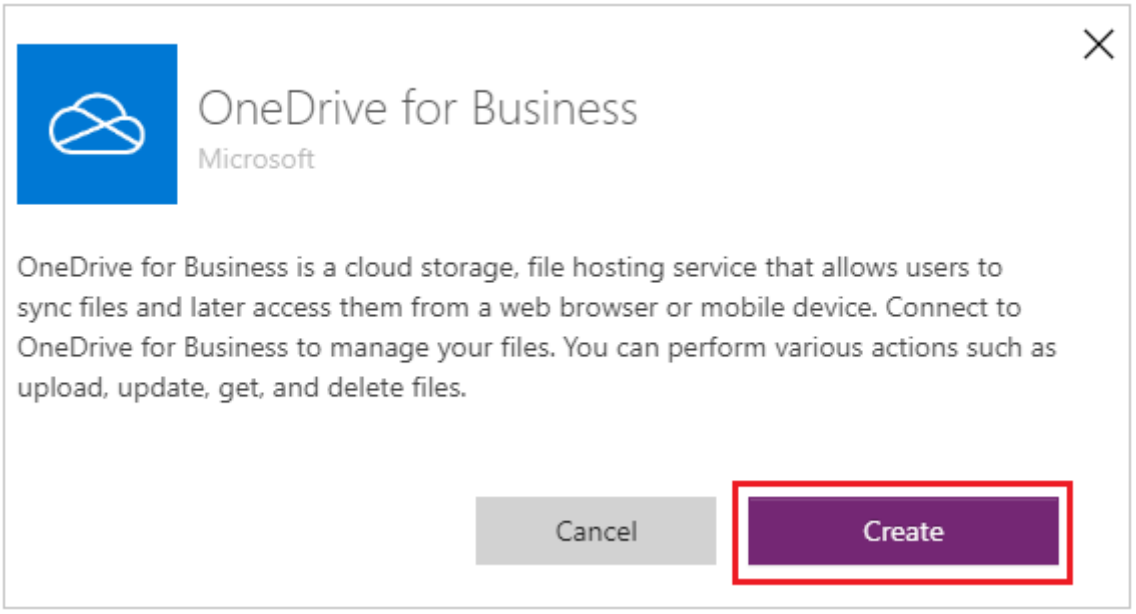
3. Select **New connection**.



4. Select a connector in the list that appears, and then follow the prompts.



5. Select the **Create** button.



6. Follow the prompts. Some connectors prompt you to provide credentials, specify a particular set of data, or perform other steps. Others such as **Microsoft Translator**, don't.

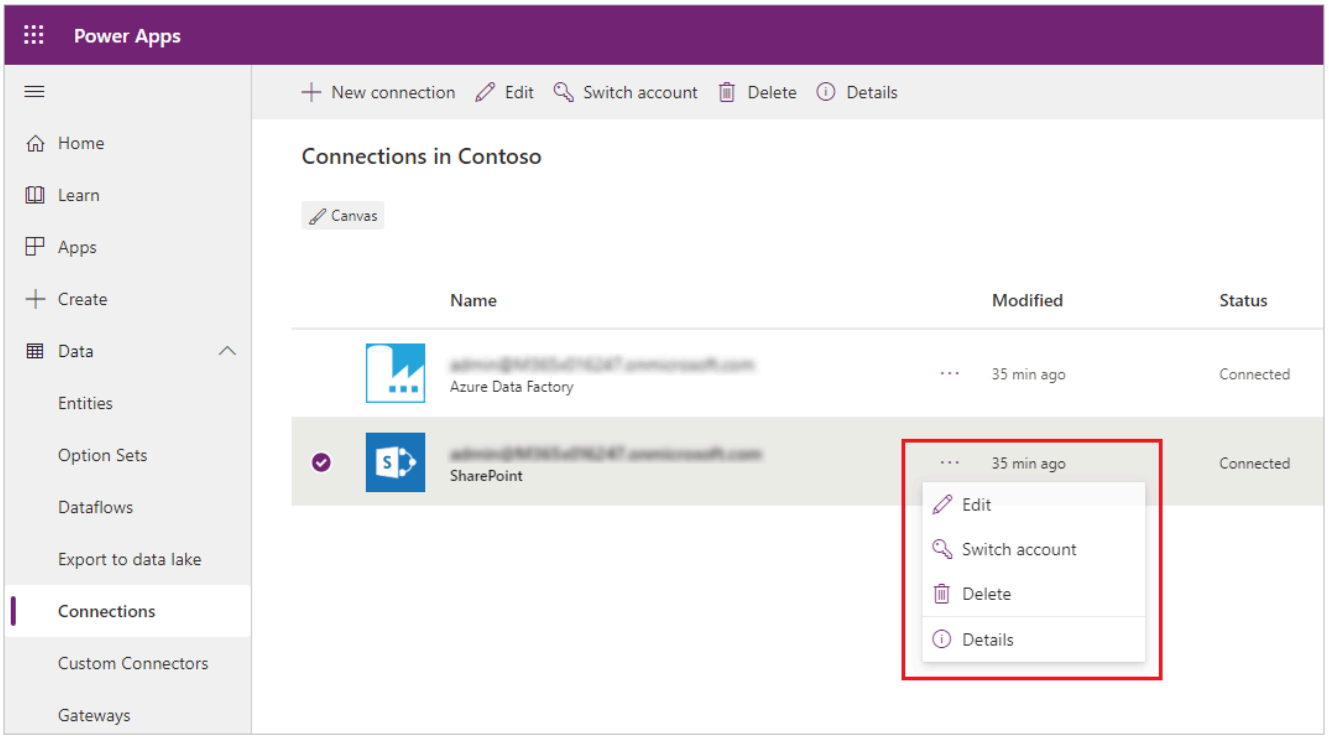
   For example, these connectors require additional information before you can use them.

   - SharePoint
   - SQL Server

The new connector appears under **Connections**, and you can add it to an app.

Update or delete a connection

In the list of connections, find the connection that you want to update or delete, and then select the ellipsis (...) on the right of the connection.



- To update the credentials for a connection, select the key icon, and then provide credentials for that connection.
- To delete the connection, select delete.
- Select the information icon to see the connection details.

Consent dialog fine-grained permissions

The consent dialog presents **fine-grained** permissions to end users. Instead of asking the user to give permissions to all actions a connector can perform, the consent dialog lists the specific permissions that the app uses. The operations that an app uses are captured and stored in the app metadata when app is saved. For example, if an app is published with the specific Read action, then it will initially just request permission for the Read action. If then the author subsequently adds **Create**, **Update**, and **Delete** record actions then the consent dialog will be presented to the user again for the aggregated permissions of **Read**, **Create**, **Update**, and **Delete**. If the author subsequently removes the **Delete** records action, then the consent dialog isn't presented again. The permissions continue with the maximum set of actions that have ever been used in the app at any point. If you wish to publish an app that only shows reduced permissions, the app must be republished under a different name.

The exception to this rule is for actions used in a Power Automate Flow that is embedded in a Power App. In this case **all** the actions are always shown for the actions used by a Power Automate flow.

Manage the consent dialog appearance for custom connectors using Microsoft Entra ID OAuth

By default, when end-users launch canvas apps they're presented a connection consent dialog before they're able to access the app experience for the first time. It's possible for admins to suppress this consent dialog for select connectors:

- Microsoft First Party connectors (like SharePoint, Office 365 Users) and
- Custom connectors using either Microsoft Entra ID OAuth or NoAuth (No Authentication)

Suppress consent dialog for apps that use custom connectors using Microsoft Entra ID OAuth

To suppress consent dialog for apps created using Power Apps that connect through custom connectors using Microsoft Entra ID OAuth/NoAuth, follow the below steps.

Step 1. Provision Microsoft's Azure API connections service principal in your Microsoft Entra tenant

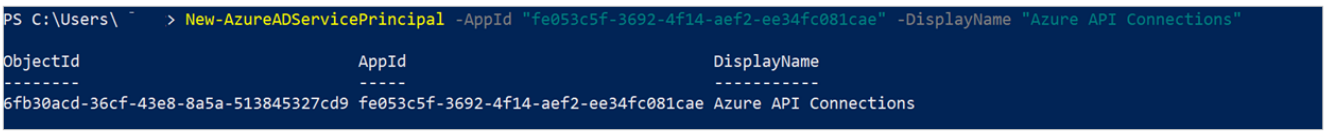
Microsoft's Azure API connectors service is used by all Power Apps using connectors. Provisioning this service in your tenant is a prerequisite for your custom applications, and custom connectors to pre-authorize this service to exercise single-sign-on capabilities with your custom applications and allow Power Apps to suppress the consent dialog.

A tenant admin must run the following PowerShell commands:

Powershell

```
Connect-MgGraph -Scope Application.ReadWrite.All -TenantId <target tenant id>
New-MgServicePrincipal -AppId "fe053c5f-3692-4f14-aef2-ee34fc081cae" -DisplayName
"Azure API Connections"
```

Example successful output:

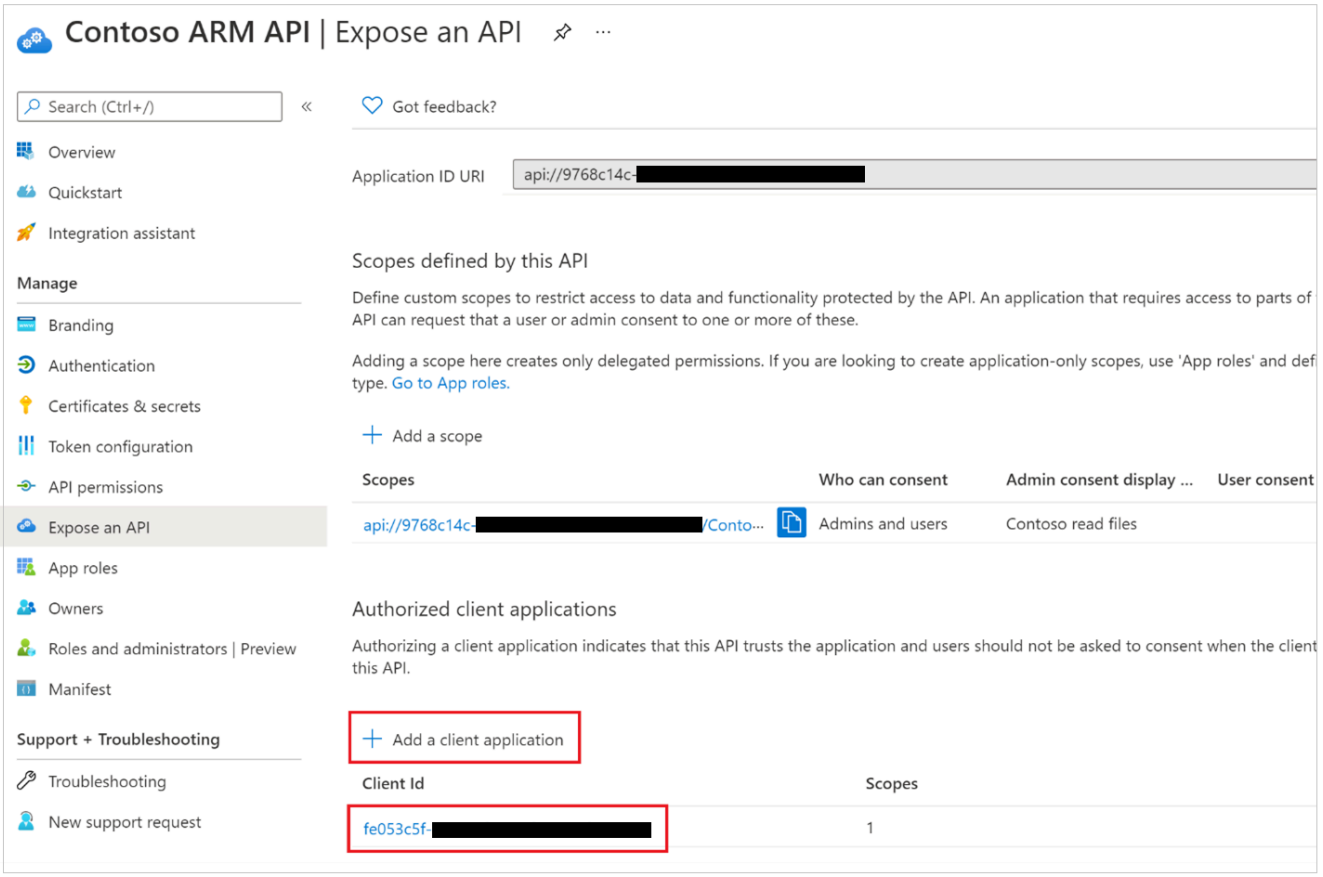


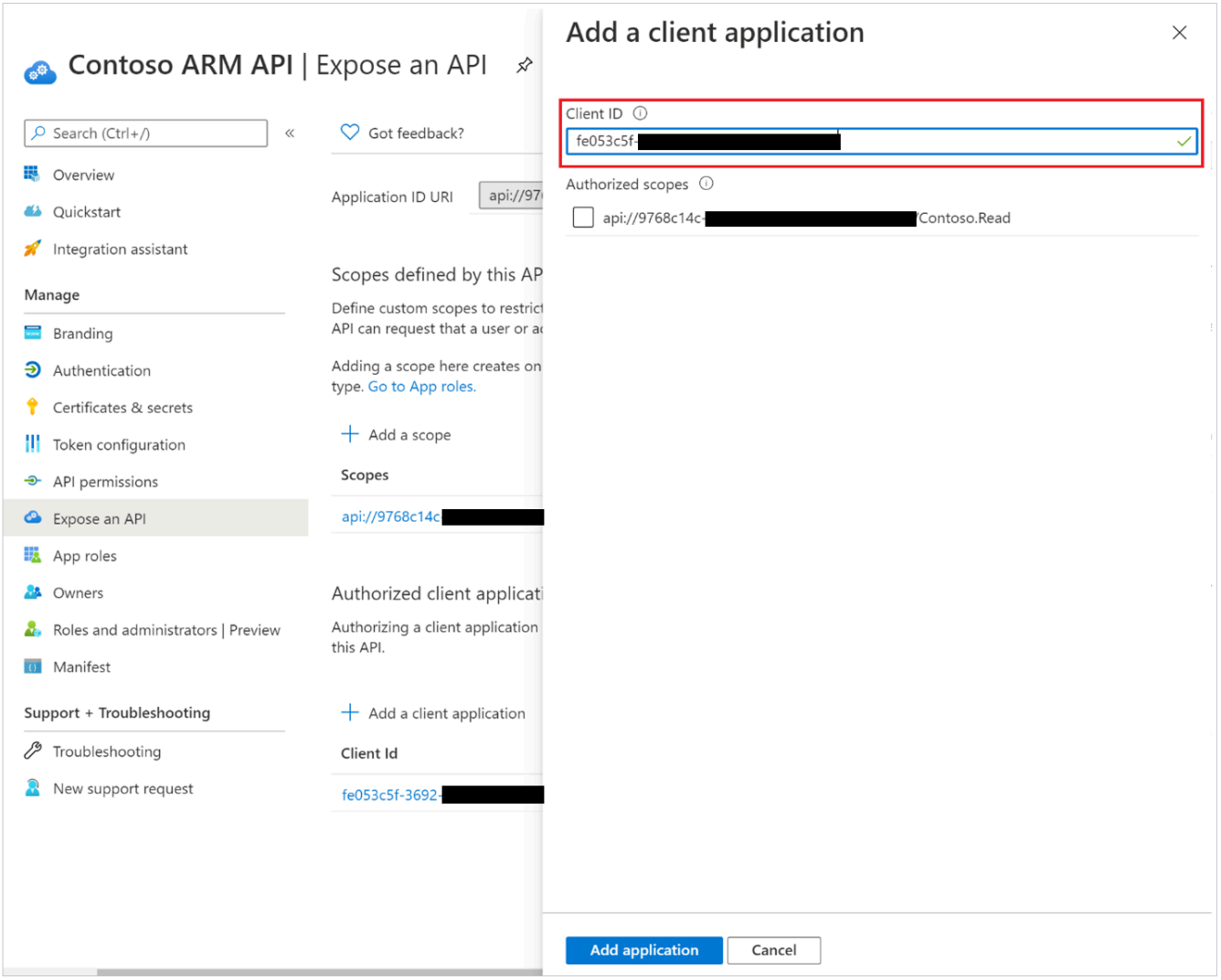
Step 2. Pre-authorize Microsoft's Azure API connections service principal in your Microsoft Entra app

For each custom connector where consent is expected to be suppressed, authorize "Microsoft's Azure API Connections" service principal to one of the scopes defined in your app.

The owner of the Microsoft Entra custom application used by a custom connector must add the app ID "fe053c5f-3692-4f14-aef2-ee34fc081cae" to one of the application scopes. Any scope can be created and used for single-sign-on to succeed.

To set the scope using Azure portal, go to Azure portal > Microsoft Entra ID > App Registrations > Select the relevant app > Expose an API > Add a client application > Add the app ID "fe053c5f-3692-4f14-aef2-ee34fc081cae" to one of the application scopes.





Step 3. Grant admin consent the client third-party Microsoft Entra app

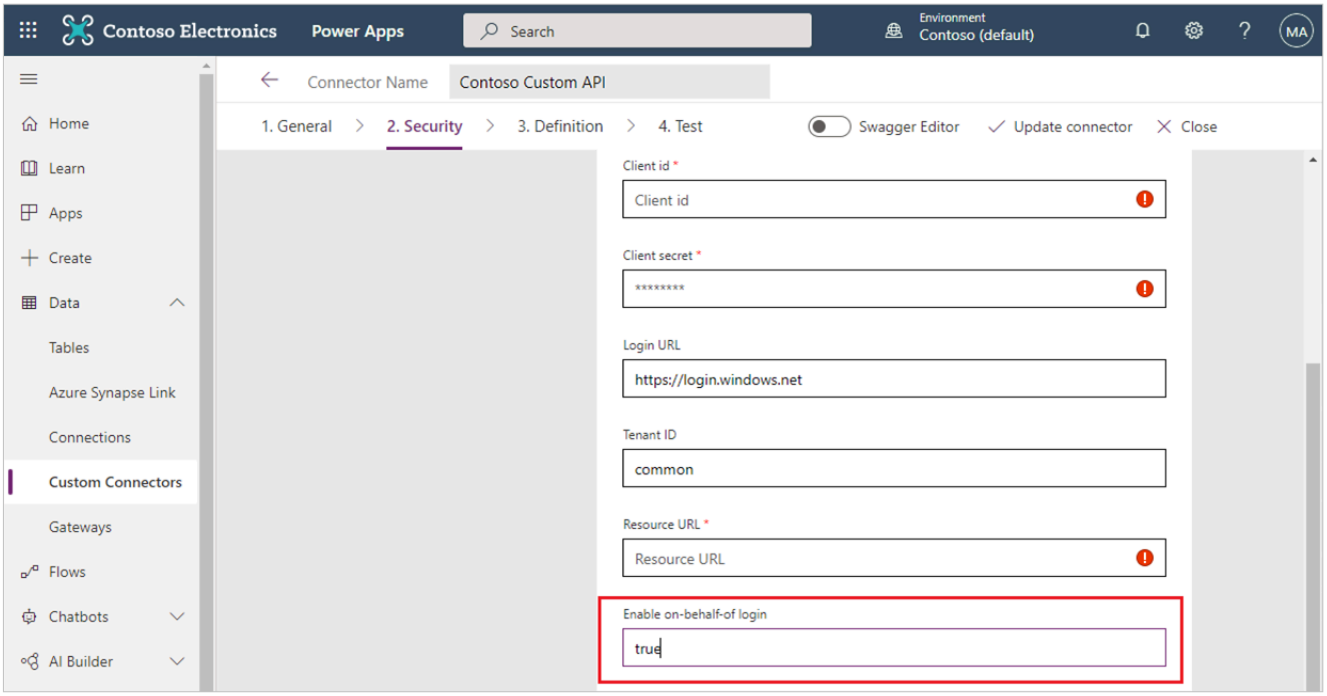
For each custom connector using OAuth/NoAuth where consent is expected to be suppressed, an admin must use Microsoft Entra's grant tenant-wide admin consent to an application.

> **Note**
>
> Admins have granular control on which custom applications, and the corresponding custom connector consent may be suppressed.

Step 4. Update custom connector in Power Platform to attempt single-sign-on

For each custom connector using OAuth/NoAuth where consent is expected to be suppressed, a user with edit permissions on the custom connector must change the "Enable on-behalf-of login" value to "true".

The owner of the custom connector must choose to edit the connector, go to the **Security** section, and change the value in **Enable on-behalf-of login** from "false" to "true".



Step 5. Admin configures consent bypass for the app

In addition to the admin consent granted on a custom application in Microsoft Entra ID, which is used by a custom connector, an admin must also configure an app to bypass consent. For each app where consent is expected to be bypassed an admin must run the following command:

Powershell

```
Set-AdminPowerAppApisToBypassConsent -AppName <Power Apps app id>
```

Remove consent suppression for apps that use custom connectors using Microsoft Entra ID OAuth/NoAuth

To remove consent suppression for a custom connector, an admin must perform at least one of the following actions:

1. Remove the tenant-wide admin consent grant to the application in Azure: Microsoft Entra's grant tenant-wide admin consent to an application.
2. Use the following Power Apps admin cmdlet to disable Power Apps' attempt to suppress the consent dialog. Clear-AdminPowerAppApisToBypassConsent

Rename Power Apps action-based data sources

Article • 12/16/2022

A change in the display name of a connector in Power Apps may cause errors in formulas used by the app. This problem takes extra efforts to update existing formulas with manual intervention, and name changes.

With the ability to rename action-based data sources, you can save the extra efforts required to fix the formulas affected by the connector name change.

Action-based and tabular data sources

Before you start using this feature, it's important to understand the difference between action-based and tabular data sources.

Action-based data sources use the data source, or the service name in the formula. Tabular data sources use the name of the table instead.

For example, Azure DevOps is action-based connector, while Excel is tabular data source.

For example, when you reference an Azure DevOps instance, the formula uses Azure DevOps as the data source name



When using Excel, the data source name in formula uses the table name (instead of the connector, or the service name):



How are data source names generated?

Data source names are generated from the display name of the connection they're based on. The first instance of a data source name in an app is typically the exact name

of the connection. For instance, if you use “AzureDevOps” data source, the name of the connection will be “AzureDevOps”. And, if you add another “AzureDevOps” data source to the same application, the second data source will be named “AzureDevOps_1”.

Renamed connectors

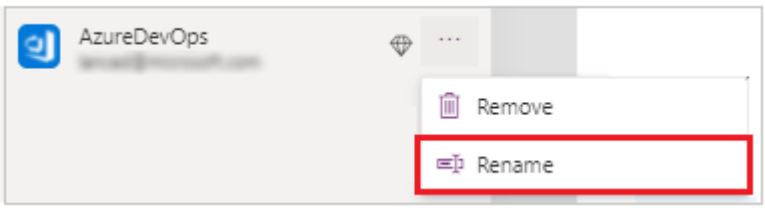
Occasionally, a connector goes through a change in the display name. For example, from “OldConnector” to “NewConnector”.

Existing apps will continue to work even though your data source names say “OldConnector”. This behavior is because your data sources still point (under the covers) to the correct connector type.

However, if you drop your existing data source in the app, and add it again, the new data source will be named “NewConnector”. Your formulas will still reference “OldConnector” though, causing formula errors wherever this data source name exists.

Fixing broken formula references to data source names

Using the new “Rename” feature, you can rename your action-based connectors and fix up your formulas automatically.



To fix this kind of problem, rename the data source from the new display name back to the old display name.

For example, rename the data source display name from “New Connector” back to “Old Connector”. Once you rename, all your formula references will reconnect and work as before. Your application will now work correctly. You can save the app changes and publish the new version.

Optional – use new data source name (Recommended)

After your data source name and formulas are in sync, you can also rename your data source again to bring it in line with the new connector name.

For example, you can rename your data source to be “NewConnector”. This rename updates all formula references to use the new name. We recommended this approach because if you ever have to drop your data source again, then readding the data source will be based on the newer connector display name. This step will help you avoid future issues.

This approach works because the data source display name, and the formula referenced data sources are in sync. In other words, exactly same display name. Renaming data sources in the formulas can’t happen before you rename your data source back because the data source display name, and the formula referenced data sources aren’t the same.

Limitations

This rename capability is limited to action-based connectors. It doesn’t work for tabular data sources. Tabular data sources use the name of the tables they’re pointing to (as opposed to the general name of the service hosting them).

See also

- Overview of connectors for canvas apps
- List of all Power Apps connectors

Query limitations: Delegation and query limits

Power Apps works best with a back-end data source when a Power Fx query fully translates into an equivalent query that runs on the data source. Power Apps sends a query the data source understands, the data source runs the query, and Power Apps gets the results. For example, the data source filters the data and only returns the rows that meet the filter criteria. When this process happens, Power Apps **delegates** the query to the data source.

But Power Fx queries can't always translate into equivalent queries on every data source. For example, Dataverse supports more query features than Excel. Dataverse supports the 'in' (membership) query operator, but Excel doesn't. A query is **nondelegable** if it uses a feature the data source doesn't support. If any part of a query expression is nondelegable, Power Apps doesn't delegate any part of the query.

When a query is nondelegable, Power Apps gets the first 500 records from the data source and then runs the actions in the query. You can increase this limit to 2,000 records. Changing the limit **Power Apps limits the result size to 500 records to keep your app performing well.** Larger result sets can cause performance problems for your app and Power Apps.

But this limitation can be a problem because the query might return incorrect results if the data source has more than 500 or 2,000 records. For example, if your data source has 10 million records and your query needs to work on the last part of the data, like family names that start with 'Z', and your query uses a nondelegable operator like distinct, you only get the first 500 or 2,000 records. So, you get incorrect results.

Create your Power Fx queries by using the delegable tables for your data source. Only use query functions that can be delegated. It's the only way to keep your app performing well and to make sure users get all the information they need.

Pay attention to delegation warnings that show where delegation isn't possible. If you work with small data sets (fewer than 500 records), you can use any data source and formula because the app processes data locally if the formula isn't delegated.

> ⓘ **Note**
>
> Delegation warnings help you manage your app so it returns correct results. If the data in your data source exceeds 500 records and a function isn't delegated, Power Fx marks the formula with a blue underline.

Delegable data sources

Delegation works only with certain tabular data sources. If a data source supports delegation, its connector documentation explains that support. These popular tabular data sources support delegation:

- Power Apps delegable functions and operations for Microsoft Dataverse
- Power Apps delegable functions and operations for SharePoint
- Power Apps delegable functions and operations for SQL Server
- Power Apps delegable functions and operations for Salesforce

Imported Excel workbooks (by using the **Add static data to your app** data source), collections, and tables stored in context variables don't need delegation. This data is already in memory, so you use the full Power Apps language.

Delegable functions

Use only formulas that can be delegated. This article lists formula elements that can be delegated. Every data source is different, and not all data sources support all these elements. Check for delegation warnings in your formula.

Filter functions

You can delegate **Filter**, **Search**, **First**, and **LookUp**.

Within the **Filter** and **LookUp** functions, use these functions with columns of the table to select the appropriate records:

- **And** (including **&&**), **Or** (including **||**), **Not** (including **!**)
- **In** > [!NOTE]

  > In is only delegated for columns on the base data source. For example, if the data source is the **Accounts** table, `Filter(Accounts, Name in ["name1", "name2"])` delegates to the data source for evaluation. But `Filter(Accounts, PrimaryContact.Fullname in ["name1", "name2"])` isn't delegated because the **Full name** column is on a different table (**PrimaryContact**) than **Accounts**. The expression is evaluated locally.

- =, <>, >=, <=, >, <
- +, -
- TrimEnds

- **IsBlank**
- **StartsWith**, **EndsWith**
- Constant values that are the same across all records, like control properties, and global and context variables.

You can also use parts of your formula that evaluate to a constant value for all records. For example, **Left(Language(), 2)**, **Date(2019, 3, 31)**, and **Today()** don't depend on any columns of the record, so they return the same value for all records. These values are sent to the data source as a constant and don't block delegation.

The previous list doesn't include these notable items:

- **If**
- *, /, **Mod**
- Column casting operations: **Text**, **Value**
- **Concatenate** (including **&**)
- **ExactIn**
- String manipulation functions: **Lower**, **Upper**, **Left**, **Mid**, **Len**, ...
- Signals: **Location**, **Acceleration**, **Compass**, ...
- Volatiles: **Rand**, ...
- Collections

Delegation and collections

When you use `With`, `UpdateContext`, or `Set`, these functions create collections internally. Collections are a static in-memory list of records and can't participate in delegation. You don't see a delegation warning.

Query limitations

Lookup and expand levels

Power Apps lets you use up to two lookup levels. A Power Fx query expression can include a maximum of two lookup functions to maintain performance. When a query expression includes a lookup, Power Apps first queries the base table, then runs a second query to expand the first table with the lookup information. One additional level beyond this is supported as the maximum. For offline scenarios, only one level of lookup expand is supported.

You can expand or join up to 20 entities in a single query. If you need to join more than 20 tables in one query, try creating a view on the data server if possible.

Expression evaluation - property of entity must be on left side 'LHS' of equality operator

Put the property of an entity to be compared on the left hand side (LHS) of an equation. For example, in the following expression, the entity property **'Business unit ID'.Name** is on the LHS, and the expression works:

Power Fx

```
Filter(
        Budgets,
        'Business unit ID'.Name = LookUp(
            Users,
            'Primary Email' = User().Email,
            'Business Unit'
        ).Name,
        DataCardValue37.Selected.'Date Range String'='Date Range String'
    )
```

However, this expression doesn't work:

Power Fx

```
 Filter(
        Budgets,
        LookUp(
            Users,
            'Primary Email' = User().Email,
            'Business Unit'
        ).Name = 'Business unit ID'.Name,
        'Date Range String'=DataCardValue37.Selected.'Date Range String'
    )
```

Sorting functions

Sort and SortByColumns can be delegated.

In **Sort**, the formula can only be the name of a single column and doesn't include other operators or functions.

Aggregate functions

Some aggregate functions can be delegated based on back-end support. Functions like Sum, Average, Min, and Max can be delegated. Counting functions, like CountRows and Count, can also be delegated. However, RemoveIf and UpdateIf have delegation restrictions. Only a

limited number of data sources support delegation for these functions. For more information, see the Delegation list.

Nondelegable functions

All other functions can't delegate. Notable functions include:

- **FirstN**, **Last**, **LastN**
- **Choices**
- **Concat**
- **Collect**, **ClearCollect**
- **GroupBy**, **Ungroup**

Nondelegable limits

Formulas that can't be delegated are processed locally. Local processing lets you use the full Power Apps formula language. But there's a tradeoff: all the data comes to the device first, which can mean getting a large amount of data over the network. This process can take time and make your app seem slow or unresponsive.

To avoid this problem, Power Apps limits the amount of data that can be processed locally to 500 records by default. This limit lets you use small data sets completely and work with large data sets by seeing partial results.

Be careful when using this feature because it can confuse users. For example, if you use the **Filter** function with a selection formula that can't be delegated over a data source with a million records, only the first 500 records are scanned. If the record you want is record 501 or 500,001, **Filter** doesn't find or return it.

Aggregate functions can also be confusing. For example, if you use **Average** over a column in that same million-record data source, **Average** can't be delegated because the expression isn't delegated (see the earlier note). Only the first 500 records are averaged. If you're not careful, a user might think a partial answer is the complete answer.

Changing the limit

The default number of records is 500, but you can change this number for your app:

1. Select **Settings**.
2. Under **General**, change the **Data row limit** setting from 1 to 2,000.

In some cases, 2,000, 1,000, or 1,500 records is enough for your scenario. You can increase this number to fit your needs, but as you do, your app's performance can degrade, especially for wide tables with many columns. It's still best to delegate as much as possible.

To make sure your app scales to large data sets, set this value to 1. Anything that can't be delegated returns a single record, which is easy to detect when testing your app. This helps you avoid surprises when moving a proof-of-concept app to production.

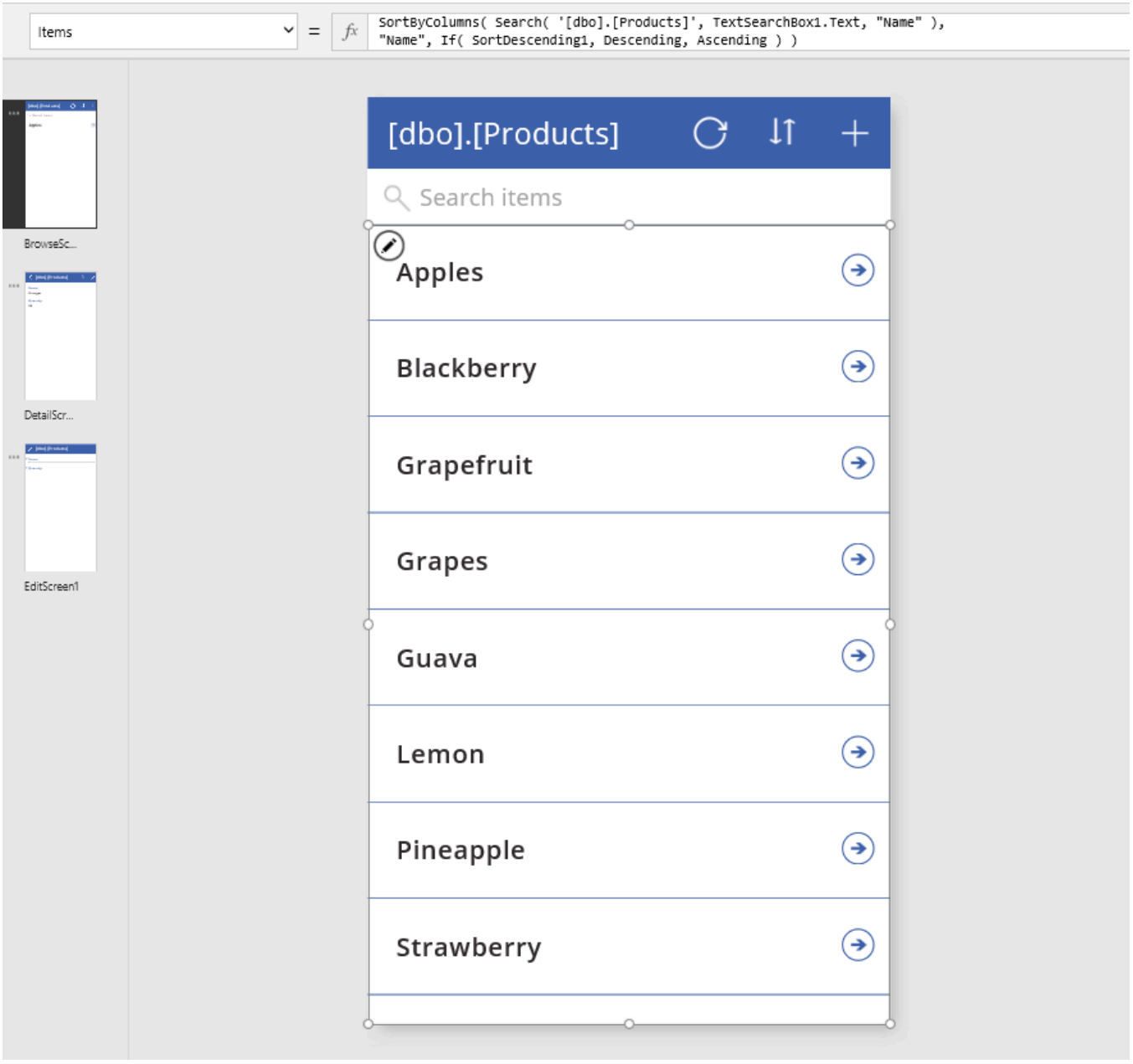
Delegation warnings

Power Apps shows a warning (yellow triangle) when you create a formula that can't be delegated. It's easier to know what is and isn't delegated.

Delegation warnings appear only on formulas that use delegable data sources. If you don't see a warning but think your formula isn't delegated, check your data source type against the list of delegable data sources earlier in this article.

Examples

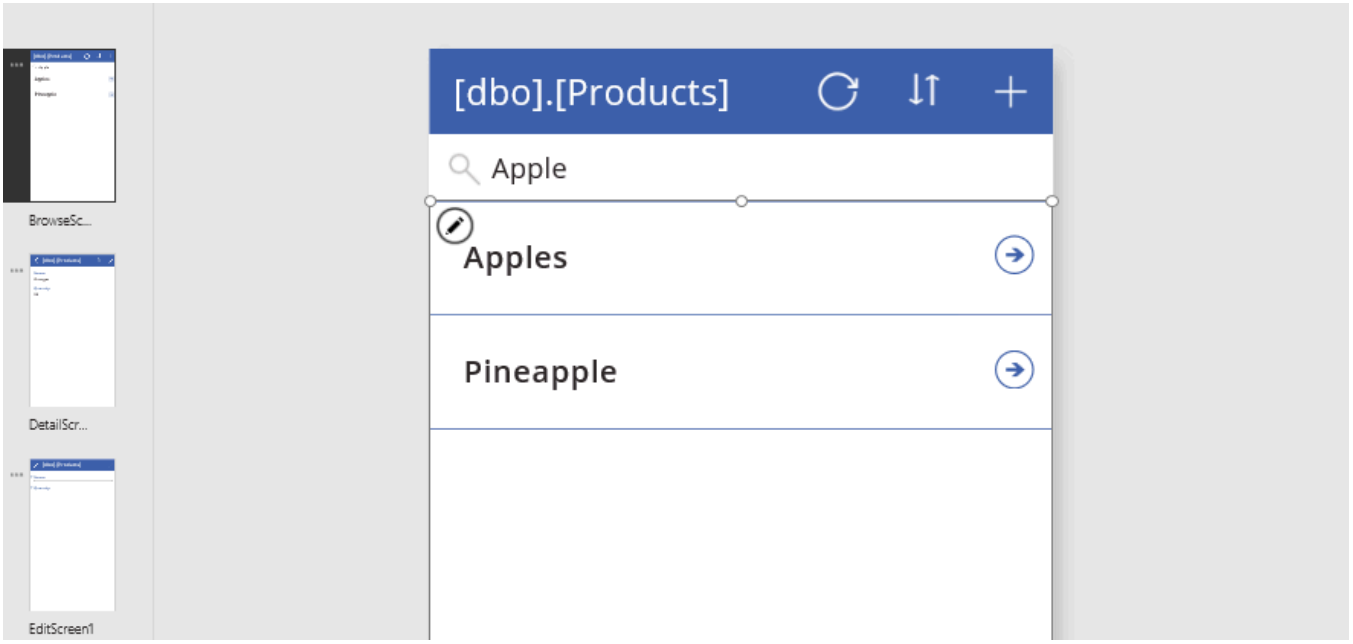
In this example, you automatically generate a three-screen app based on a SQL Server table named **[dbo].[Fruit]**. To learn how to generate the app, apply similar principles from the article about Dataverse to SQL Server.



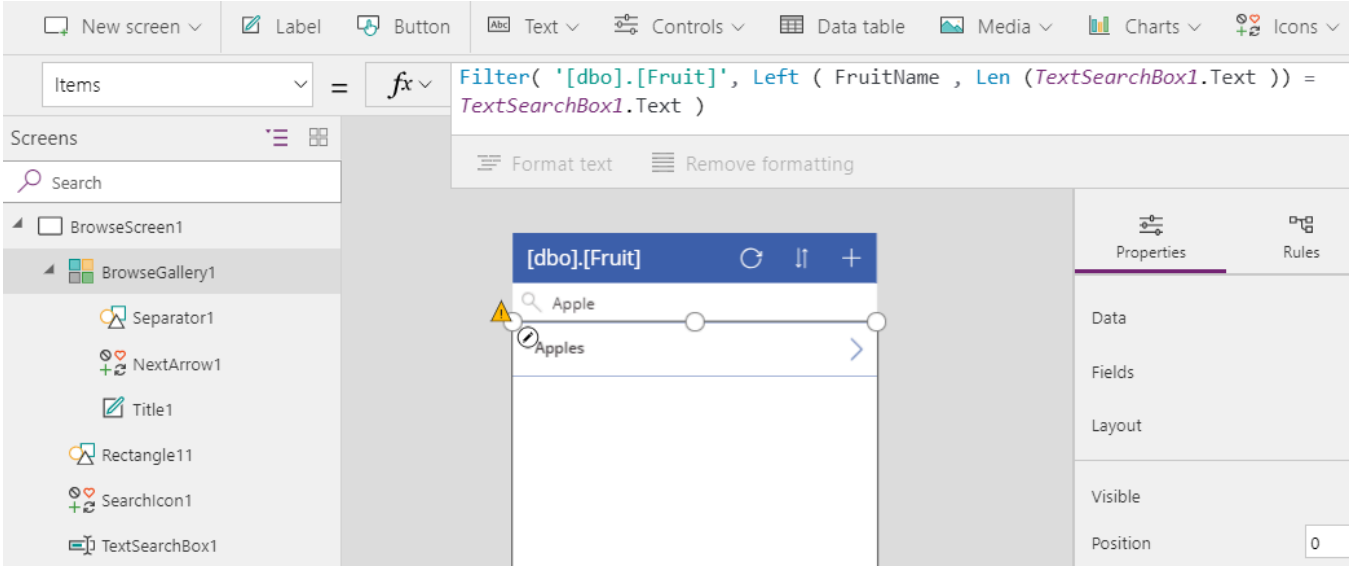
The gallery's **Items** property uses a formula with the **SortByColumns** and **Search** functions, both of which can be delegated.

In the search box, enter "**Apple**".

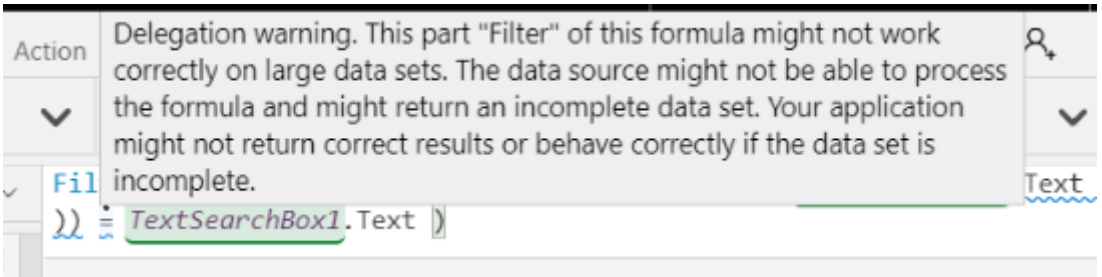
Marching dots briefly appear near the top of the screen as the app communicates with SQL Server to process the search request. All records that match the search criteria appear, even if the data source has millions of records.



The search results include "**Apples**" and "**Pineapple**" because the **Search** function looks everywhere in a text column. To find only records that contain the search term at the start of the fruit's name, use another delegable function, **Filter**, with a more specific search term. For simplicity, remove the **SortByColumns** call.



The new results include "**Apples**" but not "**Pineapple**". A yellow triangle appears next to the gallery and in the screen thumbnail if the left navigation bar shows thumbnails. A blue, wavy line appears under part of the formula. Each of these elements indicates a warning. When you hover over the yellow triangle next to the gallery, this message appears:



SQL Server is a delegable data source, and **Filter** is a delegable function. However, **Mid** and **Len** can't be delegated to any data source.

But it works, doesn't it? Kind of. That's why this is a warning and not a red, wavy squiggle.

- If the table has fewer than 500 records, the formula works perfectly. All records are brought to the device, and **Filter** is applied locally.
- If the table has more than 500 records, the formula doesn't return record 501 or higher, even if it matches the criteria.

See also

- Troubleshoot Power Apps canvas app performance issues
  Performance tips and best practice to use delegation

Last updated on 01/13/2026

Manage an on-premises data gateway in Power Apps

Article • 12/16/2022

Install an on-premises data gateway to transfer data quickly and securely between a canvas app that's built in Power Apps and a data source that isn't in the cloud, such as an on-premises SQL Server database or an on-premises SharePoint site. View all gateways for which you have administrative permissions, and manage permissions and connections for those gateways.

You can connect to on-premises data over the connectors that use data gateway.

Prerequisites

- The user name and password that you used to sign up for Power Apps.
- Administrative permissions on a gateway. (You have these permissions by default for each gateway that you install, and an administrator of another gateway can grant you these permissions for that gateway.)
- A license that supports accessing on-premises data using an on-premises gateway. For more information, see the pricing page.

Install a gateway

To install a gateway, follow the steps in Install an on-premises data gateway. Install the gateway in standard mode because the *on-premises data gateway (personal mode)* is available only for Power BI.

View and manage gateway permissions

1. In the left navigation bar of powerapps.com, click or tap **Gateways**, and then click or tap a gateway.

2. Add a user to a gateway by clicking or tapping **Users**, specifying a user or group, and then specifying a permission level:

    - **Can use**: Users who can create connections on the gateway to use for apps and flows, but cannot share the gateway. Use this permission for users who will run apps but not share them.

- **Can use + share**: Users who can create a connection on the gateway to use for apps and flows, and automatically share the gateway when sharing an app. Use this permission for users who need to share apps with other users or with the organization.
- **Admin**: Administrators who have full control of the gateway, including adding users, setting permissions, creating connections to all available data sources, and deleting the gateway.

For **Can use** and **Can use + share** permission levels, select the data sources that the user can connect to over the gateway.

> ⓘ Note
>
> **Can use** and **Can use + share** does not apply to custom connectors. Only gateway **Admins** can create connections for custom connectors.

View and manage gateway connections

1. In the left navigation bar of powerapps.com, click or tap **Gateways**, and then click or tap a gateway.
2. Click or tap **Connections**, and then click or tap a connection to view its details, edit the settings, or delete it.
3. To share a connection, click or tap **Share**, and then add or remove users.

   > ⓘ Note
   >
   > You can share only some types of connections, such as SQL Server. For more information, see Share app resources.

For more information about how to manage a connection, see Manage your connections.

Troubleshooting and Advanced Configuration

For more information about troubleshooting issues with gateways, see Troubleshoot the on-premises data gateway. For more information about configuration, see Use the on-premises data gateway app.

Next steps

- Install the on-premises data gateway.
- Create an app that connects to an on-premises data source, such as SQL Server or SharePoint.
- Share an app that connects to an on-premises data source.

Feedback

Was this page helpful? Yes No

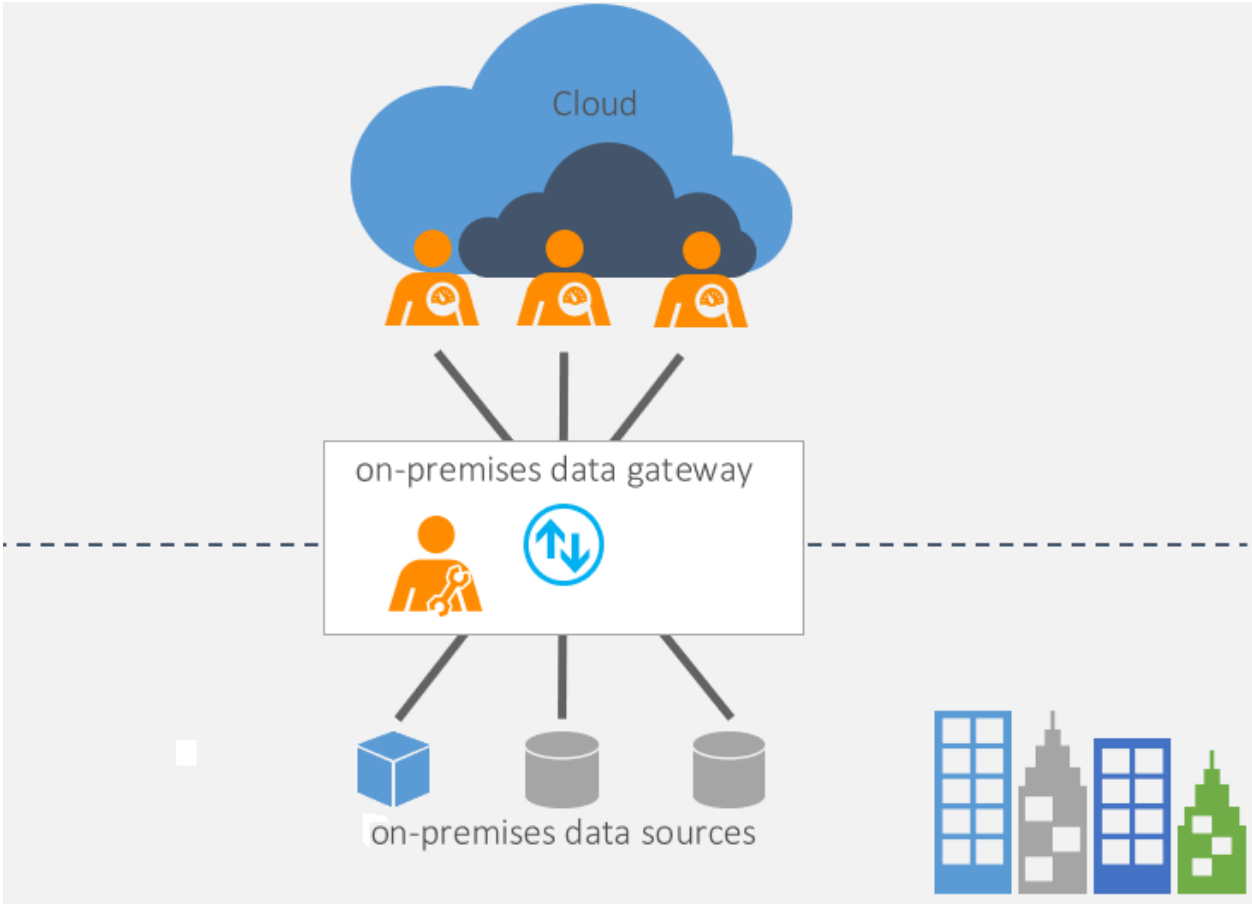
Provide product feedback

What is an on-premises data gateway?

Article • 03/01/2024

The on-premises data gateway acts as a bridge to provide quick and secure data transfer between on-premises data (data that isn't in the cloud) and several Microsoft cloud services. These cloud services include Power BI, Power Apps, Power Automate, Azure Analysis Services, and Azure Logic Apps. By using a gateway, organizations can keep databases and other data sources on their on-premises networks, yet securely use that on-premises data in cloud services.

How the gateway works



For more information on how the gateway works, see On-premises data gateway architecture.

Types of gateways

There are two different types of gateways, each for a different scenario:

- **On-premises data gateway** allows multiple users to connect to multiple on-premises data sources. You can use an on-premises data gateway with all

supported services, with a single gateway installation. This gateway is well-suited to complex scenarios with multiple people accessing multiple data sources.

- **On-premises data gateway (personal mode)** allows one user to connect to sources, and can't be shared with others. An on-premises data gateway (personal mode) can be used only with Power BI. This gateway is well-suited to scenarios where you're the only person who creates reports, and you don't need to share any data sources with others.

Use a gateway

There are four main steps for using a gateway.

1. Download and install the gateway on a local computer.
2. Configure the gateway based on your firewall and other network requirements.
3. Add gateway admins who can also manage and administer other network requirements.
4. Troubleshoot the gateway in case of errors.

End user accessing via a gateway

An end user will use the credentials for the gateway that the Power Apps author used to create the gateway. End users will see a connection in the consent dialog but it will be with the credentials provided by the Power Apps author.

Next steps

- Install the on-premises data gateway

Understand data sources for canvas apps

Article • 03/07/2025

In Power Apps, most canvas apps use external information stored in cloud services called **Data Sources**. A common example is a table in an Excel file stored in OneDrive. Apps access these data sources by using **Connections**.

This article discusses the different kinds of data sources and how to work with table data sources.

It's easy to create an app that does basic reading and writing to a data source. But sometimes you want more control over how data flows in and out of your app. This article describes how the **Patch**, **DataSourceInfo**, **Validate**, and **Errors** functions provide more control.

Kinds of data sources

Data sources can be connected to a cloud service, or they can be local to an app.

Connected data sources

The most common data sources are **tables**, which you can use to retrieve and store information. You can use **connections** to data sources to read and write data in Microsoft Excel workbooks, lists created using Microsoft Lists, SharePoint libraries, SQL tables, and many other formats, which can be stored in cloud services such as OneDrive, DropBox, and SQL Server.

Data sources other than tables include email, calendars, Twitter, and notifications, but this article doesn't discuss these other kinds of data sources.

Local data sources

With **Gallery**, **Display form**, and **Edit form** controls, you can create an app that reads and writes data from a data source. To get started, see Understand data forms.

When you ask Power Apps to create an app from data, these controls are used. Behind the scenes, the app uses an internal table to store and manipulate the data that comes from the data source.

A special type of data source is the Collection, which is local to the app and not backed by a connection to a service in the cloud. Therefore, the information can't be shared across devices for the same user or between users. Collections can be loaded and saved locally.

Kinds of tables

Tables that are internal to an app are fixed values, just as a number or a string is a value. Internal tables aren't stored anywhere, but exist in your app's memory. You can't directly modify the structure and data of a table. What you can do instead is to create a new table through a formula: you use that formula to make a modified copy of the original table.

External tables are stored in a data source for later retrieval and sharing. Power Apps provides "connections" to read and write stored data. Within a connection, you can access multiple tables of information. You can select which tables to use in your app, and each becomes a separate *data source*.

To learn more, see Working with tables about internal tables, but also external tables residing in a cloud service.

Working with tables

You can use table data sources the same way that you use an internal Power Apps table. Just like an internal table, each data source has records, columns, and properties that you can use in formulas. In addition:

- The data source has the same column names and data types as the underlying table in the connection.

  > ⓘ **Note**
  >
  > For SharePoint and Excel data sources that contain column names with spaces, Power Apps replaces the spaces with **"_x0020_"**. For example, **"Column Name"** in SharePoint or Excel appears as **"Column_x0020_Name"** in Power Apps when displayed in the data layout or used in a formula.

- The data source is loaded from the service automatically when the app is loaded. You can force the data to refresh by using the **Refresh** function.

- As users run an app, they can create, modify, and delete records and push those changes back to the underlying table in the service.
  - Records can be created with the **Patch** and **Collect** functions.
  - Records can be modified with the **Patch**, **Update**, and **UpdateIf** functions.
  - Records can be removed with the **Remove** and **RemoveIf** functions.
  - Errors when working with a data source are available through the **Errors** function.

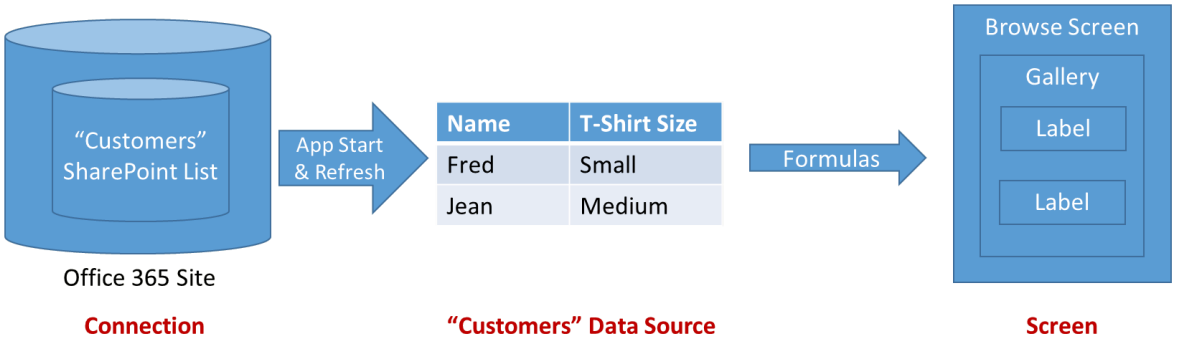
- The **DataSourceInfo**, **Defaults**, and **Validate** functions provide information about the data source that you can use to optimize the user experience.

Creating data sources

Power Apps can't be used to create a connected data source, or modify its structure; the data source must already exist in a service somewhere. For example, to create a table in an Excel workbook stored on OneDrive, you first use Excel Online on OneDrive to create a workbook. Next you create a connection to it from your app.

However, collection data sources *can* be created and modified inside an app, but are only temporary.

Display one or more records



This diagram shows the flow of information when an app reads the information in a data source:

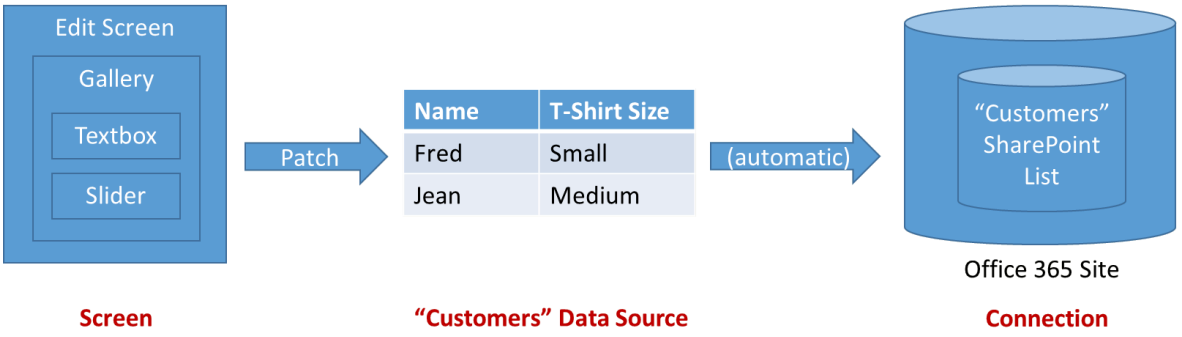
- The information is stored and shared through a storage service (in this case, Microsoft Lists or SharePoint Online).
- A connection makes this information available to the app. The connection takes care of authentication of the user to access the information.
- When the app is started or the **Refresh** function is pressed, information is drawn from the connection into a data source in the app for local use.

- Formulas are used to read the information and expose it in controls that the user can see. You can display the records of a data source by using a gallery on a screen and wiring the **Items** property to the data source: **Gallery.Items** = **DataSource**. You wire controls within the gallery, to the gallery, using the controls' **Default** property.
- The data source is also a table. So you can use **Filter**, **Sort**, **AddColumns**, and other functions to refine and augment the data source before using it as a whole. You can also use the **Lookup**, **First**, **Last**, and other functions to work with individual records.

Modify a record

The arrows in the previous diagram are one way. Changes to a data source aren't pushed back through the same formulas in which the data was retrieved. Instead, new formulas are used. Often a different screen is used for editing a record than for browsing records, especially on a mobile device.

To modify an existing record of a data source, the record must have originally come from the data source. The record might display in a gallery, a context variable, and any number of formulas, but its origin should be traceable back to the data source. Extra information travels with the record that uniquely identifies it, ensuring that you modify the correct record.



This diagram shows the flow of information to update a data source:

- An **Edit form** control provides a container for input cards, which are made up of user input controls such as a text-input control or a slider. The **DataSource** and **Item** properties are used to identify the record to edit.
- Each input card has a **Default** property, which is set to the field of the form's **ThisItem** record. The controls within the input card take their input values from **Default**. Normally, you don't need to modify this default.
- Each input card exposes an **Update** property. This property maps the user's input to a specific field of the record for writing back to the data source. Normally, you don't need to modify this property.

- A button or an image control on the screen enables the user to save changes to the record. The **OnSelect** formula of the control calls the **SubmitForm** function to do this work. **SubmitForm** reads all the **Update** properties of the cards and writes back to the data source.
- Sometimes there are issues. A network connection might be down, or a validation check is made by the service that the app didn't know about. The **Error** and **ErrorKind** properties of the form control makes this information available, so you can display it to the user.

For more fine grained control over the process, you can also use the **Patch** and **Errors** function. The **Edit form** control exposes an **Updates** property so that you can read the values of the fields within the form. You can also use this property to call a custom connector on a connection, completely bypassing the **Patch** and **SubmitForm** functions.

Validation

Before you make a change to a record, the app should do what it can to make sure the change is acceptable. There are two reasons for checking:

- *Immediate feedback to the user*. The best time to fix a problem is right when it happens, when it's fresh in the user's mind. Literally with each touch or keystroke, red text can appear that identifies an issue with their entry.
- *Less network traffic and less user latency*. More issues detected in the app mean fewer conversations over the network to detect and resolve issues. Each conversation takes time during which the user must wait before they can move on.

Power Apps offers two tools for validation:

- The data source can provide information about what is and isn't valid. For example, numbers can have minimum and maximum values, and one or more entries can be required. You can access this information with the **DataSourceInfo** function.
- The **Validate** function uses this same information to check the value of a single column or of an entire record.

Error handling

Now that you validated your record, update that record with **Patch**.

But, there might still be a problem. The network is down, validation of the service failed, or the user doesn't have the right permissions, just to name a few of the possible errors your app might encounter. Your app needs to respond appropriately to error situations, providing feedback to the user and a means for them to make it right.

When errors occur with a data source, your app automatically records the error information and makes it available through the **Errors** function. Errors are associated with the records that had the problems. If the problem is something the user can fix, such as a validation problem, they can resubmit the record, and the errors are cleared.

If an error occurs when a record is created with **Patch** or **Collect**, there's no record to associate any errors with. In this case, *blank* is returned by **Patch** and can be used as the record argument to **Errors**. Creation errors are cleared with the next operation.

The **Errors** function returns a table of error information. This information can include the column information, if the error can be attributed to a particular column. Use column-level error messages in label controls that are close to where the column is located on the edit screen. Use record-level error messages where the **Column** in the error table is *blank*, in a location close to the **Save** button for the entire record.

Working with large data sources

When you create reports from large data sources (perhaps millions of records), you want to minimize network traffic. Let's say you want to report on all **Customers** having a **StatusCode** of *Platinum* in New York City. Your **Customers** table contains millions of records.

You don't want to bring those millions of customers into your app, and then choose the ones you want. The choice should happen inside the cloud service where your table is stored, where you send the chosen records over the network.

Many, but not all, functions that you can use to choose records can be *delegated*, which means that they're run inside the cloud service. Learn more in Delegation.

Collections

Collections are a special type of data source. They're local to the app and not backed by a connection to a service in the cloud. Therefore, the information can't be shared across devices for the same user or between users.

Collections operate like any other data source, with a few exceptions:

- Collections can be created dynamically with the **Collect** function. They don't need to be established ahead of time, as connection-based data sources do.
- The columns of a collection can be modified at any time using the **Collect** function.
- Collections allow duplicate records. More than one copy of the same record can exist in a collection. Functions such as **Remove** operate on the first match they

find, unless the **All** argument is supplied.

- You can use the **SaveData** and **LoadData** functions to save and reload a copy of the collection. The information is stored in a private location that other users, apps, or devices can't access.
- You can use the **Export** and **Import** controls to save and reload a copy of the collection to a file that the user can interact with.

For more information on working with a collection as a data source, see create and update a collection.

Collections are commonly used to hold global state for the app. See working with variables for the options available for managing state.

Feedback

Was this page helpful? Yes No

Provide product feedback

Understand tables and records in canvas apps

In Power Apps, you can create a canvas app that accesses information in Microsoft Excel, SharePoint, SQL Server, and several other sources that store data in records and tables. To work most effectively with this kind of data, review the concepts that underlie these structures.

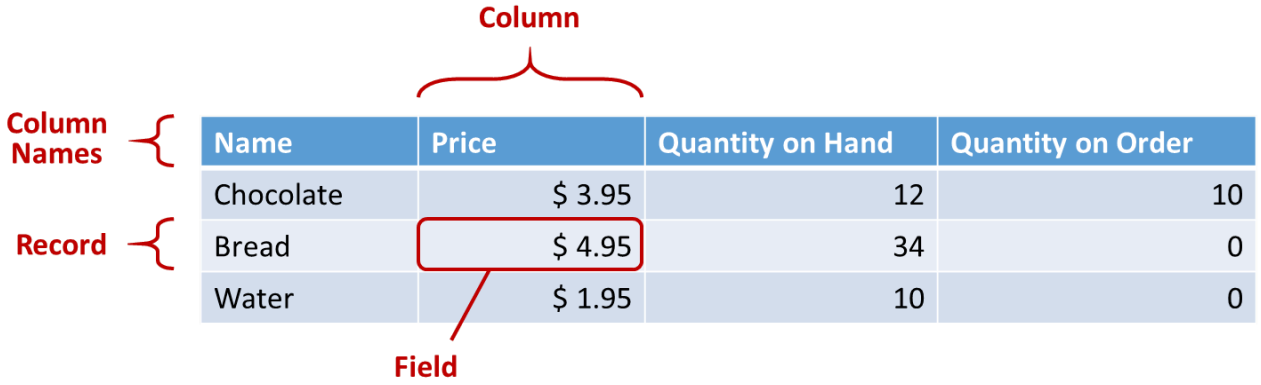
- A record contains one or more categories of information about a person, a place, or a thing. For example, a record might contain the name, the email address, and the phone number of a single customer. Other tools refer to a record as a "row" or an "item."
- A table holds one or more records that contain the same categories of information. For example, a table might contain the names, the email addresses, and the phone numbers of 50 customers.

In your app, use formulas to create, update, and manipulate records and tables. You probably read and write data to an external data source, which is an extended table. In addition, you might create one or more internal tables, which are called collections.

You can build a variety of formulas that take the name of a table as an argument, just as a formula in Excel takes one or more cell references as arguments. Some formulas in Power Apps return a table that reflects the other arguments that you specify. For example, you might create a formula:

- to update a record in a table by specifying that table as one of multiple arguments for the **Patch** function
- to add, remove, and rename columns in a table by specifying that table as an argument for the **AddColumns**, **DropColumns**, or **RenameColumns** function. None of those functions modifies the original table. Instead, the function returns another table based on the other arguments that you specify.

Elements of a table



Records

Each record contains at least one category of information for a person, a place, or a thing. The example above shows a record for each product (**Chocolate**, **Bread**, and **Water**) and a column for each category of information (**Price**, **Quantity on Hand**, and **Quantity on Order**).

In a formula, you can refer to a record by itself, outside of a table's context, by using curly braces. For example, this record **{ Name: "Strawberries", Price: 7.99 }** isn't associated with a table. Field names, such as **Name** and **Price** in that example, aren't enclosed in double quotation marks.

Fields

A field is an individual piece of information in a record. You can visualize this sort of field as a value in a column for a particular record.

Just as with a control, you refer to a field of a record by using the **.** operator on the record. For example, **First(Products).Name** returns the **Name** field for the first record in the **Products** table.

A field can contain another record or table, as the example for the **GroupBy** function shows. You can nest as many levels of records and tables as you want.

Columns

A column refers to the same field for one or more records in a table. In the preceding example, each product has a price field, and all products share the same column for that price. The preceding table has four columns, shown horizontally across the top:

- **Name**
- **Price**
- **Quantity on Hand**
- **Quantity on Order**

The column's name reflects the fields in that column.

All values within a column are of the same data type. In the preceding example, the "Quantity on Hand" column always contains a number and can't contain a string, such as "12 units," for one record. The value of any field can also be *blank*.

You might have referred to columns as "fields" in other tools.

> ⓘ **Note**

For data sources such as SharePoint, Excel, or Power BI tiles that contain column names with spaces, Power Apps replaces the spaces with "_x0020_". For example, "**Column Name**" in SharePoint, Excel, or Power BI tile appears as "**Column_x0020_Name**" in Power Apps when displayed in the data layout or used in a formula.

Table

A table comprises one or more records, each with multiple fields that have consistent names across the records.

You use the table name to refer to any table stored in a data source or a collection. You can also pass the table name to functions that take tables as arguments. A function or formula can also return tables.

As shown in the following example, you can express a table in a formula by using the **Table** function with a set of records, which you express in curly braces:

```
Table( { Value: "Strawberry" }, { Value: "Vanilla" } )
```

You can also define a single-column table by using square brackets. The following expression is an equivalent way to write the preceding expression:

```
[ "Strawberry", "Vanilla" ]
```

Table formulas

In Excel and Power Apps, you use formulas to manipulate numbers and strings of text in similar ways:

- In Excel, type a value, such as **42**, in cell **A1**, and then type a formula, such as **A1+2**, in another cell to show the value of **44**.
- In Power Apps, set the **Default** property of **Slider1** to **42**, and set the **Text** property of a label to **Slider1.Value + 2** to show the value of **44**.

In both cases, the calculated value changes automatically if you change the values of the arguments (for example, the number in cell **A1** or the value of **Slider1**).

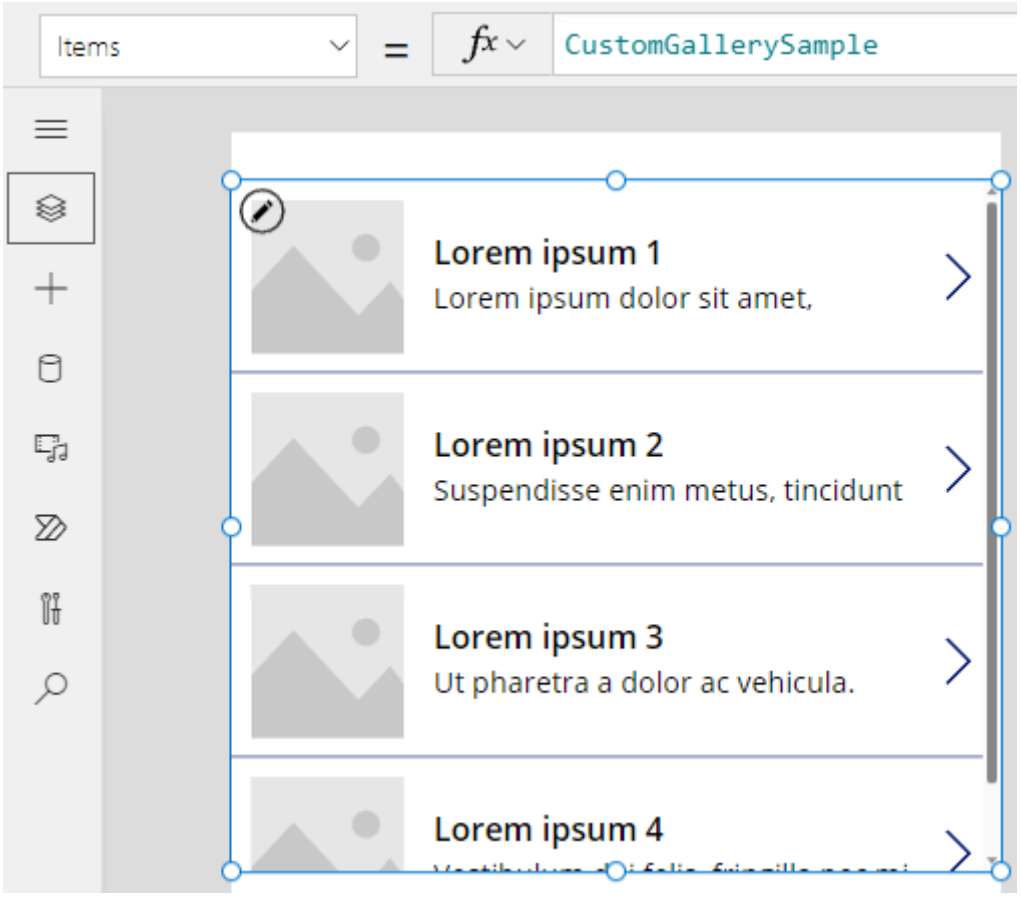
Similarly, you can use formulas to access and manipulate data in tables and records. You can use names of tables as arguments in some formulas, such as **Min(Catalog, Price)** to show the lowest value in the **Price** column of the **Catalog** table. Other formulas provide whole tables as return values, such as **RenameColumns(Catalog, "Price", "Cost")**, which returns all the records from the **Catalog** table but changes the name of the **Price** column to **Cost**.

Just as with numbers, formulas that involve tables and records automatically recalculate as the underlying table or record changes. If the cost of a product in the **Catalog** table is lowered to a value that is less than the previous minimum, the return value of the Min formula automatically changes to match it.

Let's walk through some simple examples.

1. Create a blank app for a phone, and add a vertical Gallery control that contains other controls.

   By default, the screen shows placeholder text from a table named **CustomGallerySample**. The Items property of the screen's Gallery control is automatically set to that table.



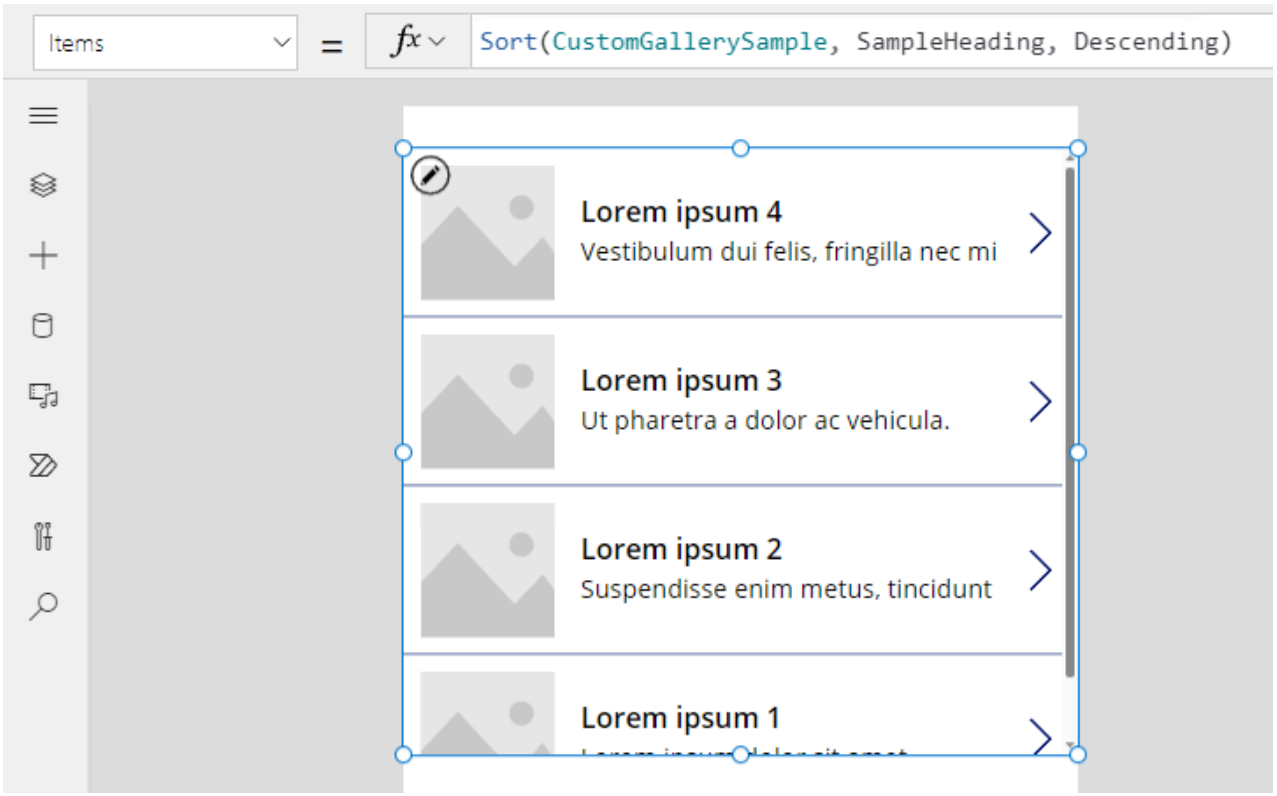
> ⓘ **Note**
>
> Some controls are rearranged and enlarged for illustration purposes.

2. Instead of setting the Items property to the name of a table, set it to a formula that includes the name of the table as an argument, as in this example:

   ```
   Sort(CustomGallerySample, SampleHeading, SortOrder.Descending)
   ```

   This formula incorporates the Sort function, which takes the name of a table as its first argument and the name of a column in that table as its second argument. The function

also supports an optional third argument, which stipulates that you want to sort the data in descending order.



3. Set the **Items** property to a formula that takes the formula from the previous step as an argument and returns a table, as in this example:

```
FirstN(Sort(CustomGallerySample, SampleHeading, SortOrder.Descending), 2)
```

In this formula, you use the **FirstN** function to show a particular number of records in a table. You use the **Sort** function as the first argument to **FirstN** and a number (in this case, **2**) as the second argument, which specifies how many records to show.

The entire formula returns a table that contains the first two records of the **CustomGallerySample** table, sorted by the **SampleHeading** column in descending order.

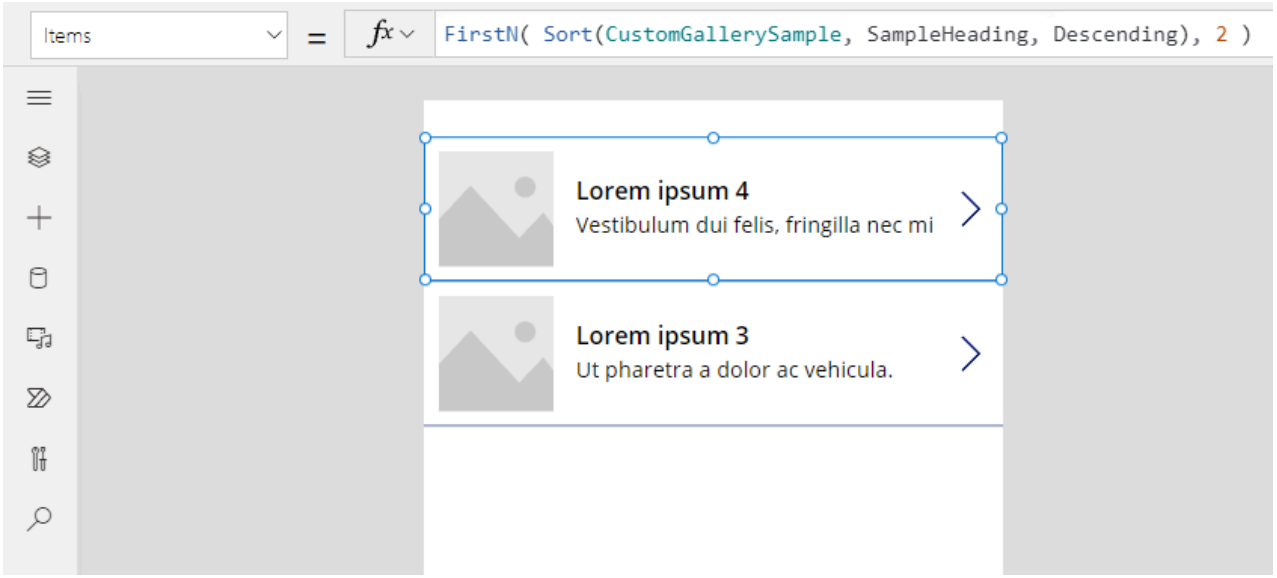


Table functions and control properties

Consider the **Lower** function. If the variable **welcome** contains the text string **"Hello, World"**, the formula **Lower(welcome)** returns **"hello, world"**. This function doesn't change the value in that variable. **Lower** is a pure function because it only processes input and produces output. It has no side effects. All functions in Excel and most functions in Power Apps are pure functions, which means the workbook or the app recalculates automatically.

Power Apps offers a set of functions that operate on tables in the same manner. These functions take tables as input and filter, sort, transform, reduce, and summarize entire tables of data. In fact, **Lower** and many other functions that typically take a single value can also take a single-column table as input.

- **Sort**, **Filter** - Sort and filter records.
- **FirstN**, **LastN** - Return the first N or last N records of the table.
- **Abs**, **Sqrt**, **Round**, **RoundUp**, **RoundDown** - Perform arithmetic operations on each record of a single-column table, resulting in a single-column table of results.
- **Left**, **Mid**, **Right**, **Replace**, **Substitute**, **Trim**, **Lower**, **Upper**, **Proper** - Perform string manipulations on each record of a single-column table, resulting in a single-column table of strings.
- **Len** - For a column of strings, returns a single-column table that contains the length of each string.
- **Concatenate** - Concatenates multiple columns of strings, resulting in a single-column table of strings.
- **AddColumns**, **DropColumns**, **RenameColumns**, **ShowColumns** - Perform column manipulation of the table, resulting in a new table with different columns.
- **Distinct** - Removes duplicate records.
- **Shuffle** - Shuffles records into a random order.
- **HashTags** - Searches for hashtags in a string.
- **Errors** - Provides error information when you work with a data source.

Many of these functions take a single-column table as their input. If an entire table has only one column, you can specify it by name. If a table has multiple columns, you can specify one of those columns by using *Table.Column* syntax. For example, **Products.Name** returns the single-column table of only **Name** values from the **Products** table.

You can completely reshape a table however you want by using the **AddColumns**, **RenameColumns**, **ShowColumns**, or **DropColumns** function. Again, these functions change only their output, not their source.

Properties of controls can also be tables:

- **Items** - Applies to galleries, list boxes, and combo boxes. This property defines the table that the gallery or the list shows.
- **SelectedItems** - Applies to list boxes and combo boxes. This property defines the table of items that the user selected if **SelectMultiple** is enabled.

Behavioral formulas

Some functions are specifically designed to modify data and have side effects. Because these functions aren't pure, you must build them carefully. They can't participate in automatically recalculating values in the app. You can use these functions only within behavioral formulas.

- **Collect**, **Clear**, **ClearCollect** - Creates collections, clears them, and adds data to them.
- **Patch** - Modifies one or more fields in a record.
- **Update**, **UpdateIf** - Updates records that match one or more criteria that you specify.
- **Remove**, **RemoveIf** - Deletes records that match one or more criteria that you specify.

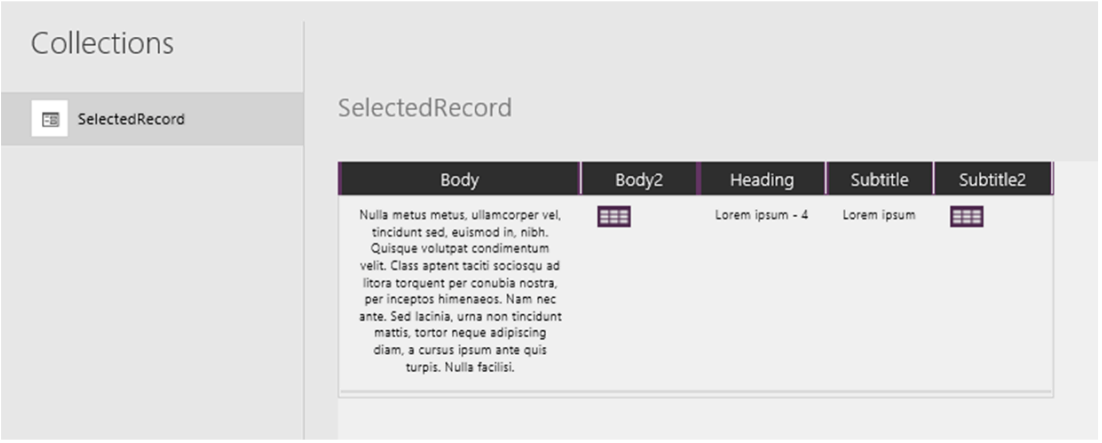
Record formulas

You can also build a formula that calculates data for an individual record, takes an individual record as an argument, and provides an individual record as a return value. Returning to the gallery example, use the **Gallery1.Selected** property to display information from whatever record the user selects in that gallery.

1. Add a **Button**, and set its **OnSelect** property to this formula:
   Collect(SelectedRecord, Gallery1.Selected)

2. Select the button while holding down the Alt key.

3. In the **File** menu, select **Collections.**



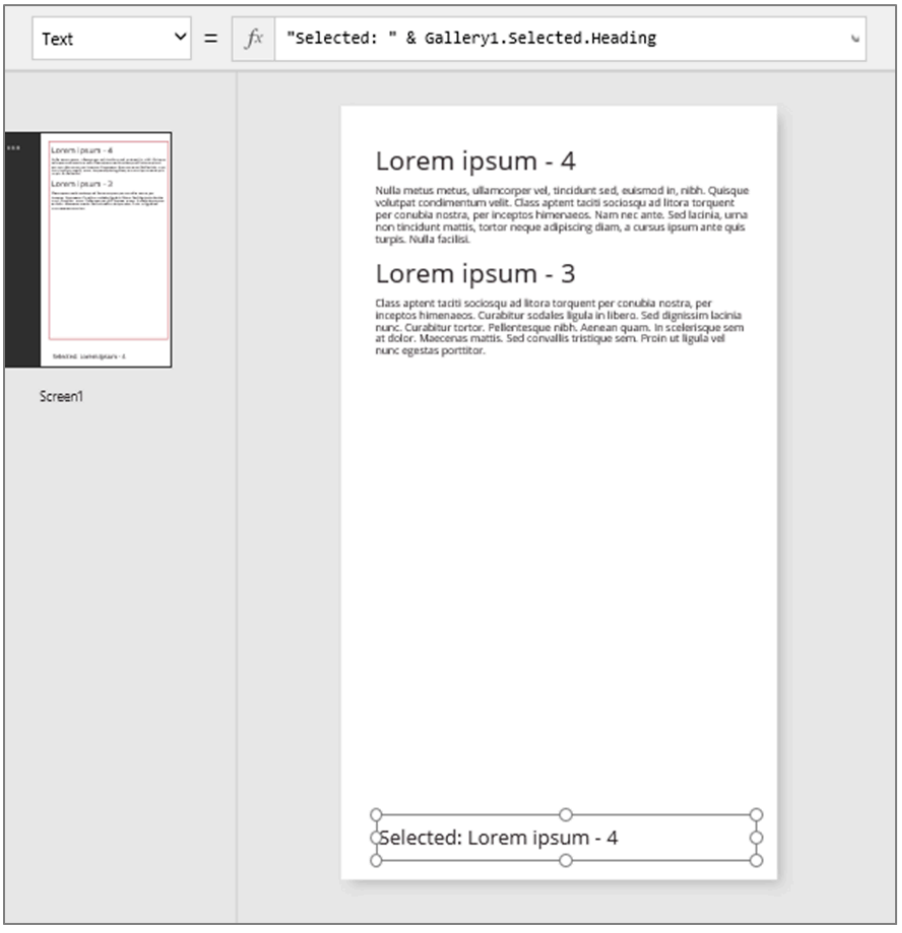
This formula returns a record that includes not only the data from the record that's currently selected in the gallery but also each control in that gallery. For example, the record contains

both a **SampleText** column, which matches the **SampleText** column in the original table, and a **Subtitle1** column, which represents the label that shows the data from that column. Select the table icon in the **Subtitle1** column to drill into that data.

> **Note**
>
> The **Subtitle1** column might be named **Subtitle2** or similar if you add elements other than those that this topic specifies.

Now that you have the selected record, you can extract individual fields from it by using the **.** operator.

1. Add a **Label** control, and then move it under the gallery and the button.
2. Set the label's **Text** property to this expression:
   "Selected: " & Gallery1.Selected.SampleHeading



You took the **Selected** property, which is a record, and extracted the **SampleHeading** property from it.

You can also use a record as a general-purpose container for related named values.

- If you build a formula around the **UpdateContext** and **Navigate** functions, use a record to gather the context variables that you want to update.
- Use the **Updates** property on an **Edit form** control to gather the changes that the user makes in a form.
- Use the **Patch** function to update a data source but also to merge records.

In these cases, the record was never a part of a table.

Record functions and control properties

Functions that return records:

- **FirstN**, **LastN** - Returns the first or last record or records of the table.
- **Lookup** - Returns the first record from a table that matches one or more criteria.
- **Patch** - Updates a data source or merges records.
- **Defaults** - Returns the default values for a data source.

Properties that return records:

- **Selected** - Applies to galleries and list boxes. Returns the currently selected record.
- **Updates** - Applies to galleries. Pulls together all the changes that a user makes in a data-entry form.
- **Update** - Applies to input controls such as text-input controls and sliders. Sets up individual properties for the gallery to pull together.

Record scope

Some functions work by evaluating a formula across all the records of a table, one record at a time. You use the formula's result in different ways for each function:

- **AddColumns** - Use the formula to get the value for the new field you add.
- **Average**, **Max**, **Min**, **Sum**, **StdevP**, **VarP** - Use the formula to get the value you want to aggregate.
- **Filter**, **Lookup** - Use the formula to decide if the record should be part of the output.
- **Concat** - Use the formula to decide which strings to join together.
- **Distinct** - Use the formula to return a value that helps find duplicate records.
- **ForAll** - Use the formula to return any value, possibly with side effects.
- **Sort** - Use the formula to get the value to sort the records by.
- **With** - Use the formula to return any value, possibly with side effects.

Inside these formulas, you can reference the fields of the record you're working on. Each of these functions creates a "record scope" where the formula runs. The fields of the record show

up as top-level identifiers. You can also reference control properties and other values from throughout your app.

For example, consider a table of **Products**:

Product	Quantity Requested	Quantity Available
Widget	6	3
Gadget	10	20
Gizmo	4	11
Apparatus	7	6

To create this example table in your app, insert a button, set its **OnSelect** property to this formula, and then select the button (click it while you hold down the Alt key in Power Apps Studio):

Power Fx

```
Set( Products,
    Table(
        { Product: "Widget",    'Quantity Requested': 6,  'Quantity Available': 3 },
        { Product: "Gadget",    'Quantity Requested': 10, 'Quantity Available': 20
},
        { Product: "Gizmo",     'Quantity Requested': 4,  'Quantity Available': 11
},
        { Product: "Apparatus", 'Quantity Requested': 7,  'Quantity Available': 6 }
    )
)
```

To check if any of these products have more requested than is available, use this formula:

```
Filter( Products, 'Quantity Requested' > 'Quantity Available' )
```

The first argument to **Filter** is the table of records to work on. The second argument is a formula. **Filter** makes a record scope for evaluating this formula. The fields of each record are available, like **Product**, **Quantity Requested**, and **Quantity Available**. The result of the comparison decides if each record should be part of the function's result:

Product	Quantity Requested	Quantity Available
Widget	6	3
Apparatus	7	6

You can add to this example by calculating how much of each product to order:

Power Fx

```
AddColumns(
    Filter( Products, 'Quantity Requested' > 'Quantity Available' ),
    "Quantity To Order", 'Quantity Requested' - 'Quantity Available'
)
```

Here, you add a calculated column to the result. **AddColumns** uses its own record scope to calculate the difference between what you requested and what is available.

Product	Quantity Requested	Quantity Available	Quantity To Order
Widget	6	3	3
Apparatus	7	6	1

Finally, you can reduce the result table to just the columns that you want:

Power Fx

```
ShowColumns(
    AddColumns(
        Filter( Products, 'Quantity Requested' > 'Quantity Available' ),
        "Quantity To Order", 'Quantity Requested' - 'Quantity Available'
    ),
    "Product",
    "Quantity To Order"
)
```

Product	Quantity To Order
Widget	3
Apparatus	1

In the preceding example, you used double quotes (") in some places and single quotes (') in other places. You need single quotes when you reference the value of an object, such as a field or table, and the name of the object contains a space. Use double quotes when you're not referencing the value of an object but instead talking about it. Use double quotes especially in situations where the object doesn't yet exist, as in the case of **AddColumns**.

Disambiguation

Field names you add with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope by using the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table you're operating upon, using this pattern:
  Table[@*FieldName*]

- To access global values, such as data sources, collections, and context variables, use the pattern **[@***ObjectName***]** (without a table designation).

If the table you're operating upon is an expression, such as **Filter(** *Table***, ...)**, you can't use the disambiguation operator. Only the innermost record scope can access fields from this table expression, by not using the disambiguation operator.

For example, imagine having a collection **X**:

Value
1
2

You create this collection by using **ClearCollect(X, [1, 2])**.

And another collection **Y**:

Value
"A"
"B"

You create this collection by using **ClearCollect(Y, ["A", "B"])**.

In addition, define a context variable named **Value** with this formula: **UpdateContext({Value: "!"})**

Let's put it all together. In this context, the following formula:

Power Fx

```
Ungroup(
    ForAll( X,
        ForAll( Y,
            Y[@Value] & Text( X[@Value] ) & [@Value]
        )
    ),
    "Value"
)
```

produces this table:

Value
"A1!"
"A2!"
"B1!"
"B2!"

What's going on here? The outermost **ForAll** function defines a record scope for **X**, allowing access to the **Value** field of each record as it processes. You can access it by simply using the word **Value** or by using **X[@Value]**.

The innermost **ForAll** function defines another record scope for **Y**. Since this table also has a **Value** field defined, using **Value** here refers to the field in **Y**'s record and no longer the one from **X**. To access **X**'s **Value** field, you must use the longer version with the disambiguation operator.

Since **Y** is the innermost record scope, accessing fields of this table doesn't require disambiguation, so you can use this formula with the same result:

Power Fx

```
Ungroup(
    ForAll( X,
        ForAll( Y,
            Value & Text( X[@Value] ) & [@Value]
        )
    ),
    "Value"
)
```

All the **ForAll** record scopes override the global scope. The **Value** context variable you defined isn't available by name without the disambiguation operator. To access this value, use **[@Value]**.

Ungroup flattens the result because nested **ForAll** functions result in a nested result table.

Single-column tables

To operate on a single column from a table, use the **ShowColumns** function as in this example:

Power Fx

```
ShowColumns( Products, "Product" )
```

This formula produces this single-column table:

Product
Widget
Gadget
Gizmo
Apparatus

For a shorter alternative, specify *Table.Column*, which extracts the single-column table of just *Column* from *Table*. For example, this formula produces exactly the same result as using **ShowColumns**.

Power Fx

```
Products.Product
```

Inline records

Express records by using curly braces that contain named field values. For example, you can express the first record in the table at the start of this topic by using this formula:

```
{ Name: "Chocolate", Price: 3.95, 'Quantity on Hand': 12, 'Quantity on Order': 10 }
```

You can also embed formulas within other formulas, as this example shows:

```
{ Name: First(Products).Name, Price: First(Products).Price * 1.095 }
```

You can nest records by nesting curly braces, as this example shows:

```
{ 'Quantity': { 'OnHand': ThisItem.QuantOnHand, 'OnOrder': ThisItem.QuantOnOrder } }
```

Enclose each column name that contains a special character, such as a space or a colon, in single quotes. To use a single quote within a column name, double it.

Note that the value in the **Price** column doesn't include a currency symbol, such as a dollar sign. That formatting is applied when the value is displayed.

Inline tables

Create a table by using the Table function and a set of records. You can express the table at the start of this topic by using this formula:

Power Fx

```
Table(
    { Name: "Chocolate", Price: 3.95, 'Quantity on Hand': 12, 'Quantity on Order':
10 },
    { Name: "Bread", Price: 4.95, 'Quantity on Hand': 34, 'Quantity on Order': 0 },
    { Name: "Water", Price: 4.95, 'Quantity on Hand': 10, 'Quantity on Order': 0 }
)
```

You can also nest tables:

Power Fx

```
Table(
    { Name: "Chocolate",
      'Quantity History': Table( { Quarter: "Q1", OnHand: 10, OnOrder: 10 },
                                 { Quarter: "Q2", OnHand: 18, OnOrder: 0 } )
    }
)
```

Inline value tables

Create single-column tables by specifying values in square brackets. The resulting table has a single column, named **Value**.

For example, `[ 1, 2, 3, 4 ]` is equivalent to `Table( { Value: 1 }, { Value: 2 }, { Value: 3 }, { Value: 4 } )` and returns this table:

Value
1
2
3
4

Last updated on 01/14/2026

Understand record references and polymorphic lookups in canvas apps

Article • 12/16/2022

When you wrote a research paper in school, you probably provided a list of your references at the end. You didn't include a copy of the actual background material you used but rather a web link, book title and author, or other information so that someone could track down the original source. You mixed different kinds of sources in a single list, newspaper articles next to audio recordings, each with their own specific details for a proper citation. For example, Wikipedia articles often include a long list of references.

In canvas apps, you often work with copies of records downloaded from data sources. You use the **LookUp** and **Filter** functions and the **Gallery** control's **Selected** property to identify the specific record that you want. All the records from **Filter** or **Selected** will be of the same table type, so you can use fields with a simple *.Field* notation. These copies often include reference information so you can use the **Patch** function to update the original source.

Canvas apps also support *record references*. Much like a research-paper reference, a record reference refers to a record without including a complete copy of it. Such a reference can refer to a record in any table. Also like research-paper references, you can mix records from different tables in a single column.

Many operations on record references are identical to working with records. You can compare record references to each other and to full records. You can set a record reference's value with the **Patch** function just as you would a lookup with a full record.

There is one important usage difference: you can't directly access the fields of a record reference without first establishing to which table it refers. This is because canvas apps require that all types be known when you write formulas. Because you don't know the type of a record reference until the app is running, you can't use the simple *.Field* notation directly. You must first dynamically determine the table type with the **IsType** function and then use *.Field* notation on the result of the **AsType** function.

Table type refers to the schema of each record in a table. Each table has a unique set of fields with different names and data types. Each record of the table inherits that structure; two records have the same table type if they come from the same table.

> **Note**

You can choose from many different connectors to connect to different types of data sources for canvas apps. However, when working with canvas apps inside Power Apps Studio, **columns** in Microsoft Dataverse are referred to as **fields** similar to all other data sources. **Column** is only used when referring to a column inside Dataverse. More information: Dataverse terminology updates

Polymorphic lookups

Microsoft Dataverse supports relationships between records. Each record in the **Accounts** table has a **Primary Contact** lookup column to a record in the **Contacts** table. The lookup can only refer to a record in **Contacts** and can't refer to a record in, say, the **Teams** table. That last detail is important because you always know what columns will be available for the lookup.

Dataverse also supports polymorphic lookups, which can refer to a record from any table in a set. For example, the **Owner** column can refer to a record in the **Users** table or the **Teams** table. The same lookup column in different records could refer to records in different tables. In this case, you don't always know what columns will be available.

Canvas record references were designed for working with polymorphic lookups in Dataverse. You can also use record references outside of this context, which is how the two concepts differ.

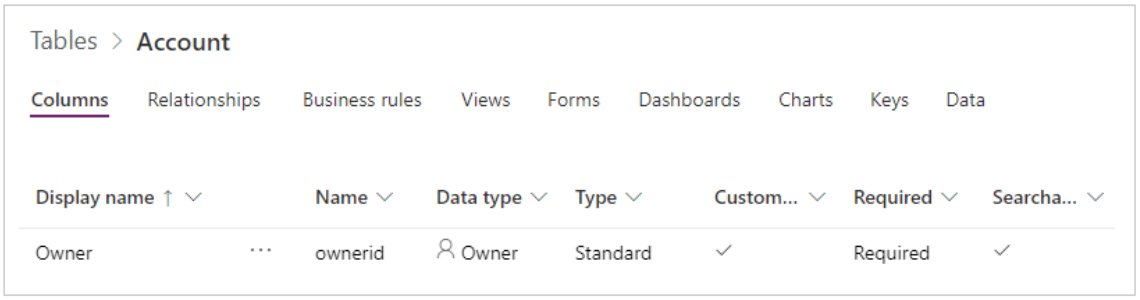
In the next section, you'll start to explore these concepts by working with the **Owner** lookup.

Show the columns of a record owner

Every table in Dataverse includes an **Owner** column. This column can't be removed, you can't add another, and it always requires a value.

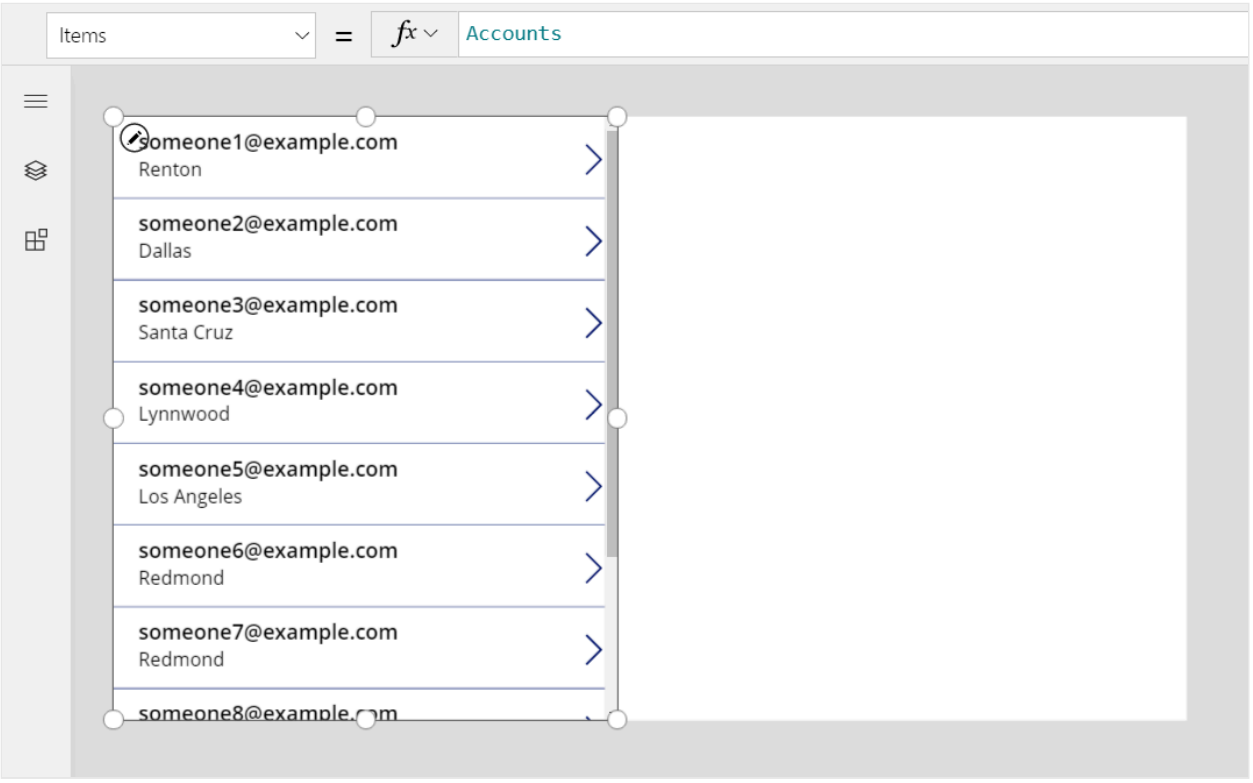
To show that column in the **Account** table:

1. Sign in to Power Apps.
2. In the left pane bar, select **Data** > **Tables**.
3. In the list of tables, select **Account**.
4. In the upper-right corner, open the filter list (which is set to **Default** by default), and then select **All**.
5. Scroll down until the **Owner** column appears.



This lookup column can refer to a record from either the **Teams** table or the **Users** table. Not every record in these tables has permission to be an **Owner**; check the supported roles if you run into a problem.

This graphic shows a simple gallery of **Accounts**, where the **Accounts** table has been added to the app as a data source:

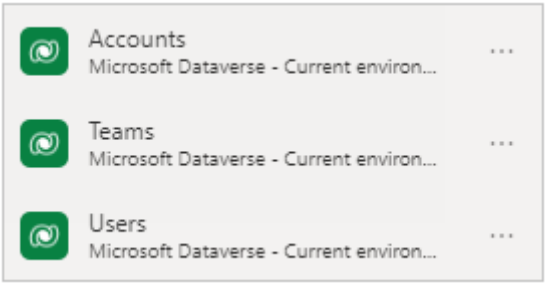


> **Important**
>
> Throughout this topic, the graphics show some names and other values that aren't part of the sample data that ships with Dataverse. The steps accurately demonstrate how to configure controls for a particular result, but your experience will vary based on the data for your organization.

To show the owner of each account in the gallery, you might be tempted to use the formula **ThisItem.Owner.Name**. However, the name field in the **Team** table is **Team Name**, and the name field in the **User** table is **Full Name**. The app can't know which type

of lookup you're working with until you run the app, and it can vary between records in the **Accounts** table.

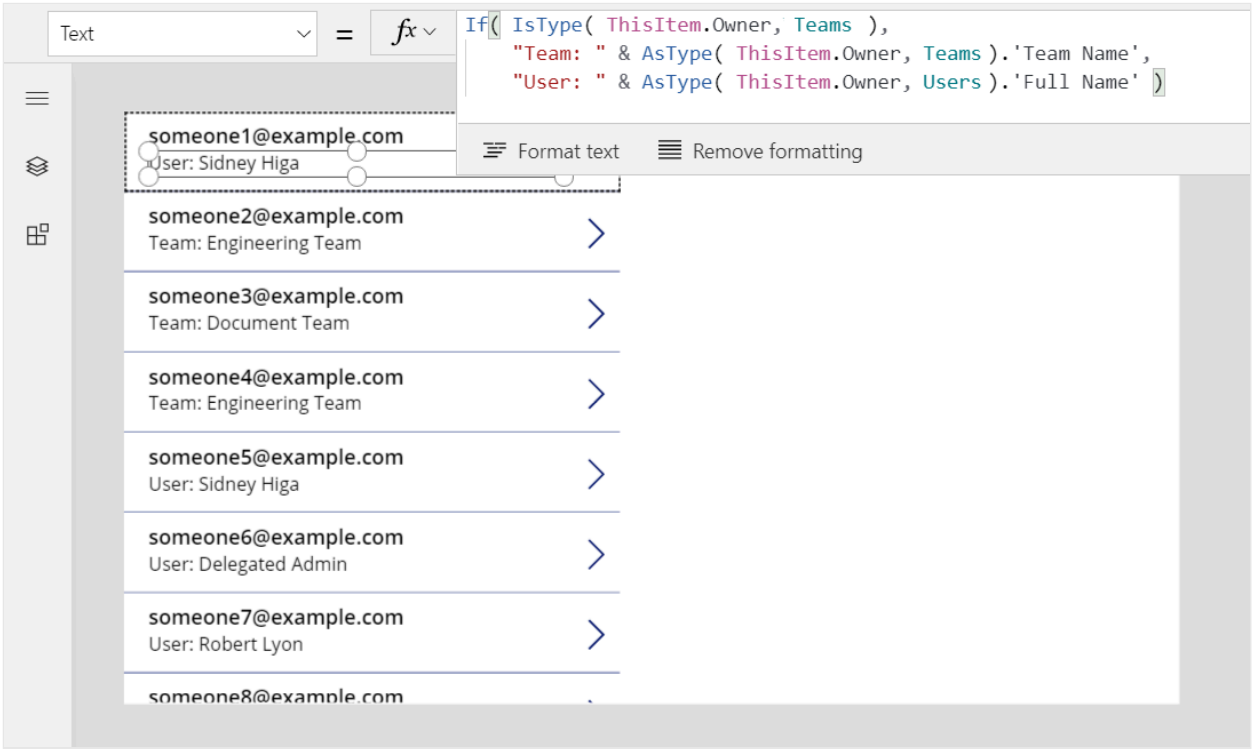
You need a formula that can adapt to this variance. You also need to add the data sources for the table types that **Owner** could be (in this case, **Users** and **Teams**). Add these three data sources to your app:



With these data sources in place, use this formula to display the name of either a user or a team:

Power Fx

```
If( IsType( ThisItem.Owner, Teams ),
    "Team: " & AsType( ThisItem.Owner, Teams ).'Team Name',
    "User: " & AsType( ThisItem.Owner, Users ).'Full Name' )
```

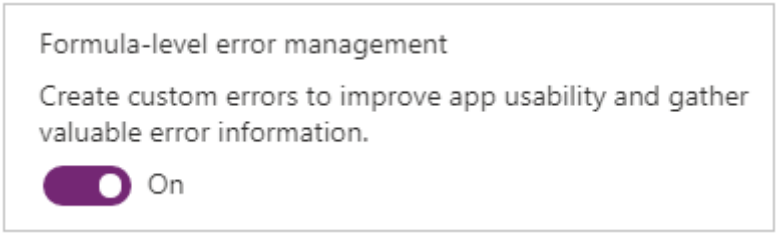


In this formula, the **IsType** function tests the **Owner** field against the **Teams** table. If it's of that table type, the **AsType** function casts it to a **Team** record. At this point, you can access all the fields of the **Teams** table, including **Team Name**, by using the *.Field*

notation. If **IsType** determines that the **Owner** isn't a record in the **Teams** table, that field must be a record in the **Users** table because the **Owner** field is required (can't be *blank*).

To use any fields of a record reference, you must first use the **AsType** function to cast it to a specific table type. You can't access fields directly from the **Owner** field because the system doesn't know what table type you want to use.

The **AsType** function returns an error if the **Owner** field doesn't match the table type being requested, so you can use the **IfError** function to simplify this formula. First, turn on the experimental feature **Formula-level error management**:



Then replace the previous formula with this one:

Power Fx

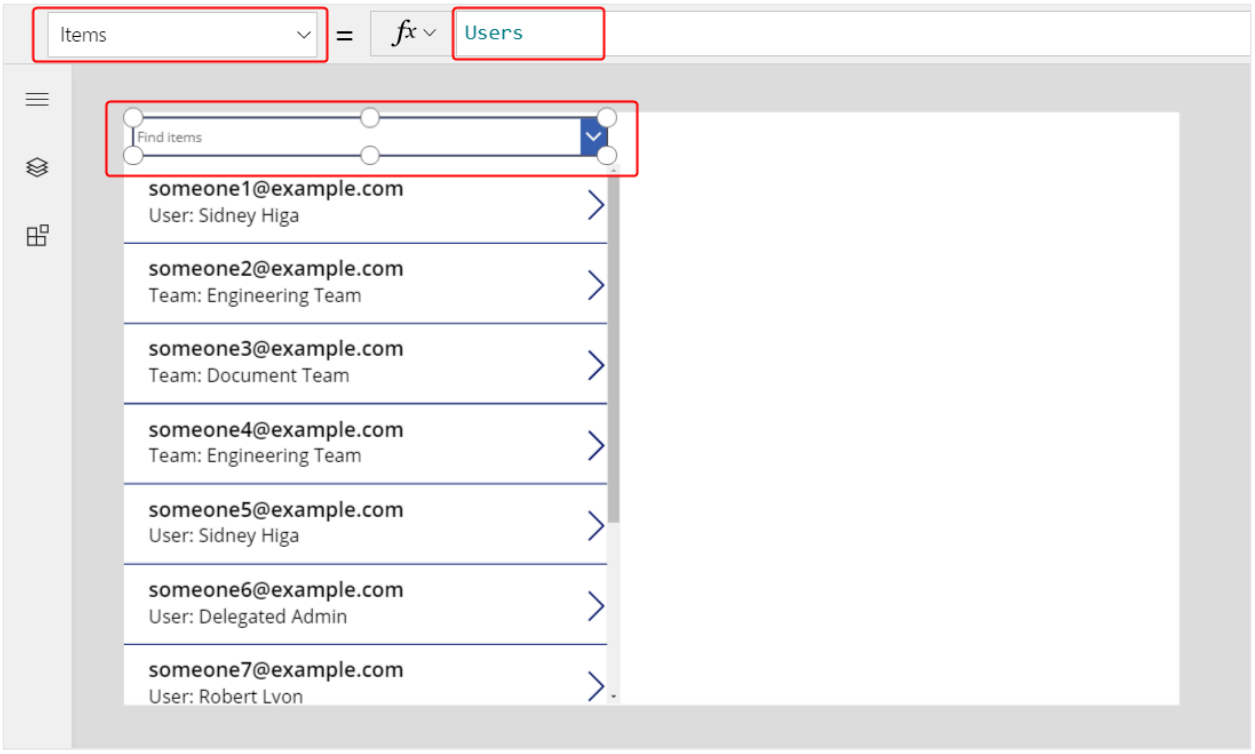
```
IfError(
    "Team: " & AsType( ThisItem.Owner, Teams ).'Team Name',
    "User: " & AsType( ThisItem.Owner, Users ).'Full Name' )
```

Filter based on an owner

Congratulations—you've finished the hardest aspect of working with a record reference. Other use cases are more straightforward because they don't access fields of the record. As a case in point, take filtering, which you'll explore in this section.

Add a **Combo box** control above the gallery, and set these properties of the new control:

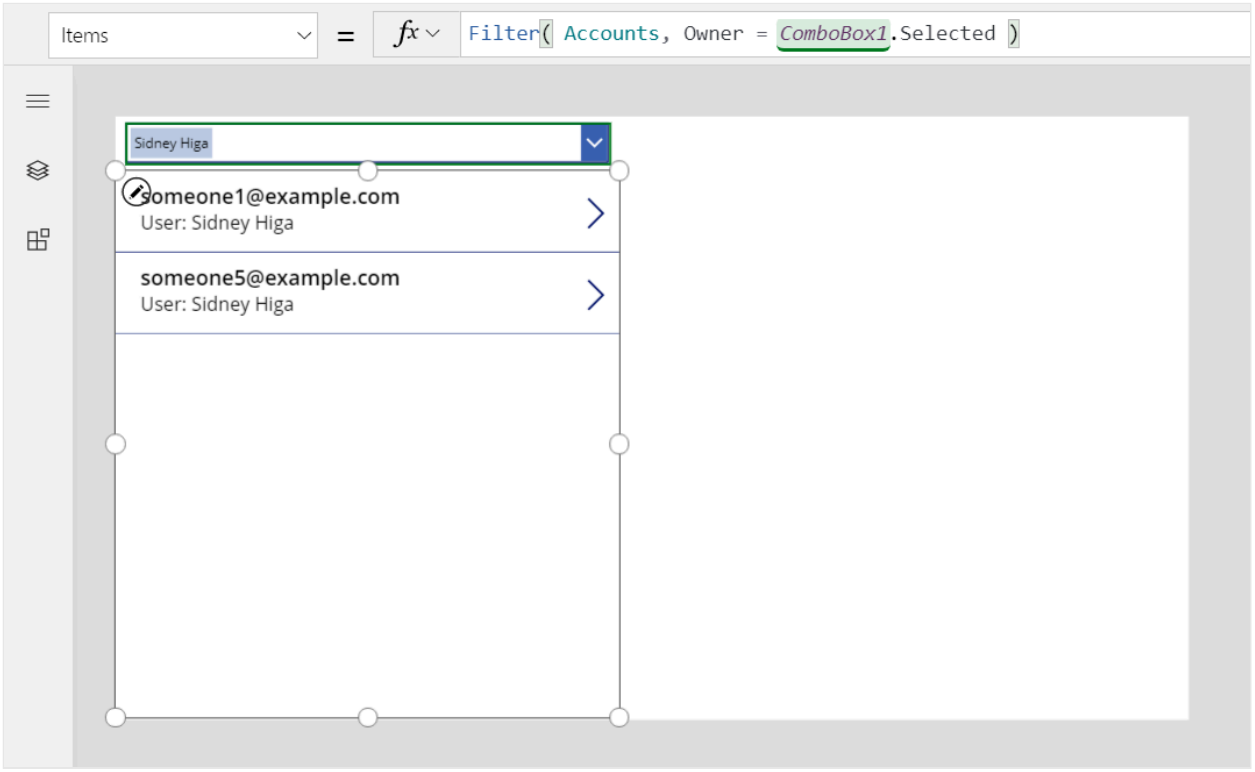
- **Items**: `Users`
- **SelectMultiple**: `false`



To filter the gallery by a specific user selected from this combo box, set the gallery's **Items** property to this formula:

Power Fx

```
Filter( Accounts, Owner = ComboBox1.Selected )
```



Important

The instructions in this topic are accurate if you follow the steps exactly. However, any formula that refers to a control by its name fails if the control has a different name. If you delete and add a control of the same type, the number at the end of the control's name changes. For any formula that shows an error, confirm that it contains the correct names of all controls.

You don't need to use **IsType** or **AsType** because you're comparing record references to other record references or to full records. The app knows the table type of **ComboBox1.Selected** because it's derived from the **Users** table. Accounts for which the owner is a team won't match the filter criterion.

You can get a little fancier by supporting filtering by either a user or a team.

1. Make some space near the top of the screen by resizing the gallery and moving the combo box, insert a **Radio** control above the gallery, and then set these properties for the new control:

   - **Items**: `[ "All", "Users", "Teams" ]`
   - **Layout**: `Layout.Horizontal`

2. For the **Combo box** control, set this property (if the combo box disappears, select **Users** in the radio control):

   - **Visible**: `Radio1.Selected.Value = "Users"`

3. Copy and paste the **Combo box** control, move the copy directly over the original, and then set these properties for the copy:

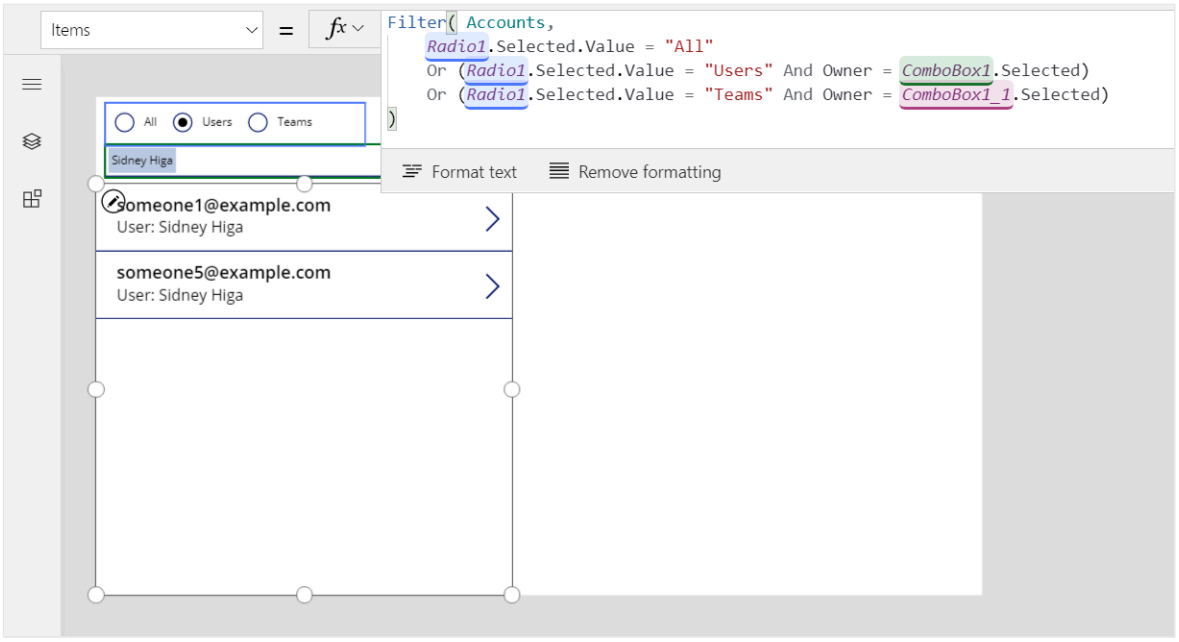
   - **Items**: `Teams`
   - **Visible**: `Radio1.Selected.Value = "Teams"`

   The app will display only one combo box at a time, depending on the state of the radio control. Because they're directly above one another, they'll appear to be the same control that changes its contents.

4. Finally, set the **Items** property of the **Gallery** control to this formula:

   Power Fx

   ```
   Filter( Accounts,
       Radio1.Selected.Value = "All"
       Or (Radio1.Selected.Value = "Users" And Owner = ComboBox1.Selected)
       Or (Radio1.Selected.Value = "Teams" And Owner =
   ComboBox1_1.Selected)
   )
   ```

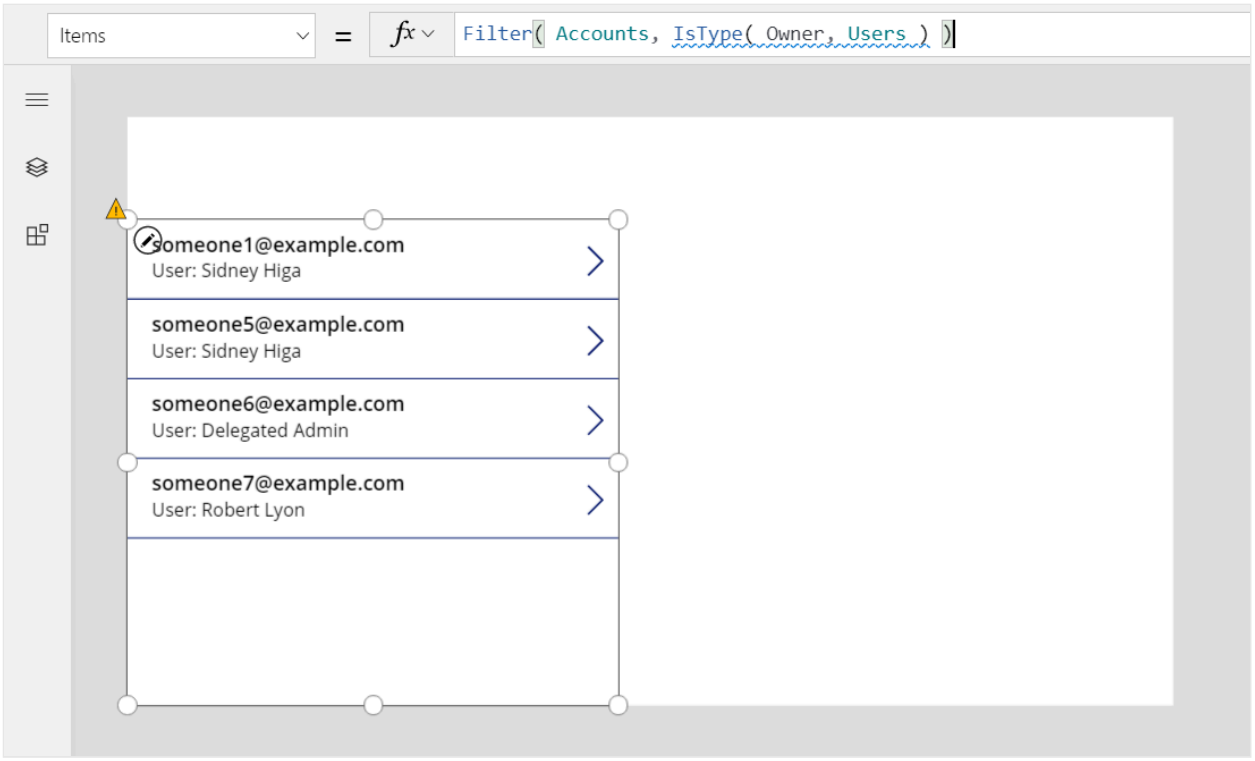


With these changes, you can show all records or filter them based on either a user or a team:



The formula is fully delegable. The portion that's comparing the radio-button values is a constant across all records and is evaluated before the rest of the filter is sent to Dataverse.

If you want to filter on the type of the owner, you can use the **IsType** function, but it's not yet delegable.



Update the owner by using Patch

You can update the **Owner** field in the same manner as any other lookup. To set the currently selected account's owner to the first team:

Power Fx

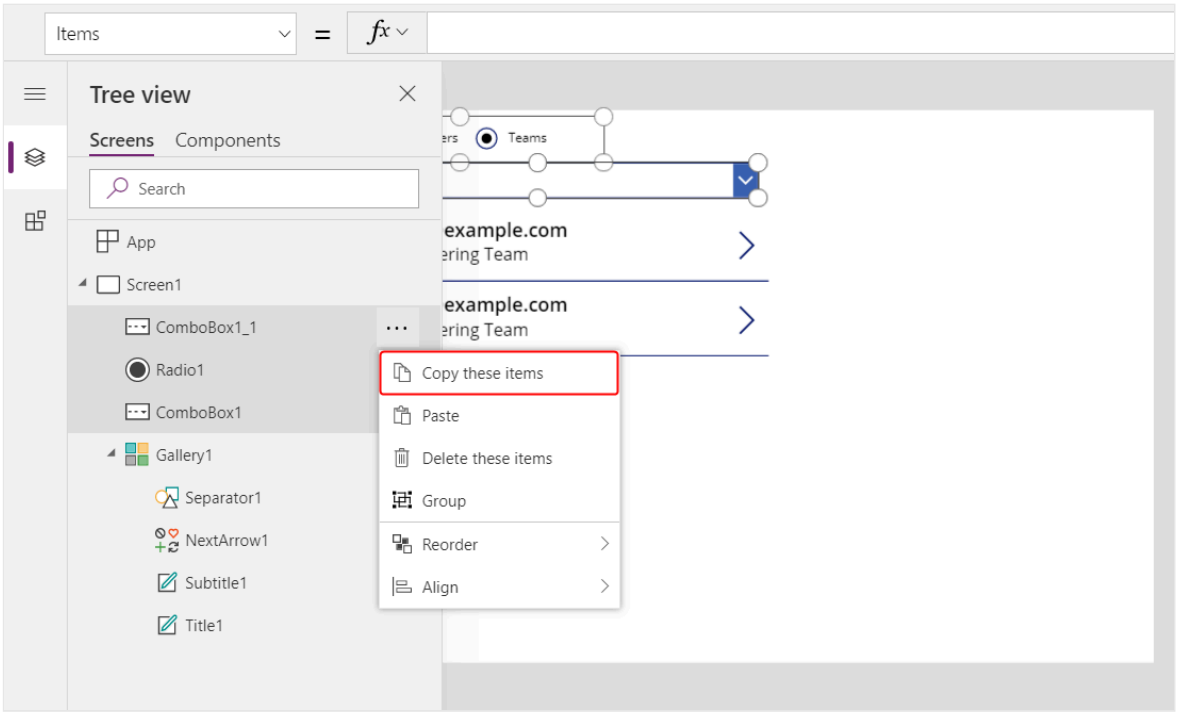
```
Patch( Accounts, Gallery1.Selected, { Owner: First( Teams ) } )
```

This approach doesn't differ from a normal lookup because the app knows the type of **First(Teams)**. If you want the first user instead, replace that portion with **First(Users)**. The **Patch** function knows that the **Owner** field can be set to either of these two table types.

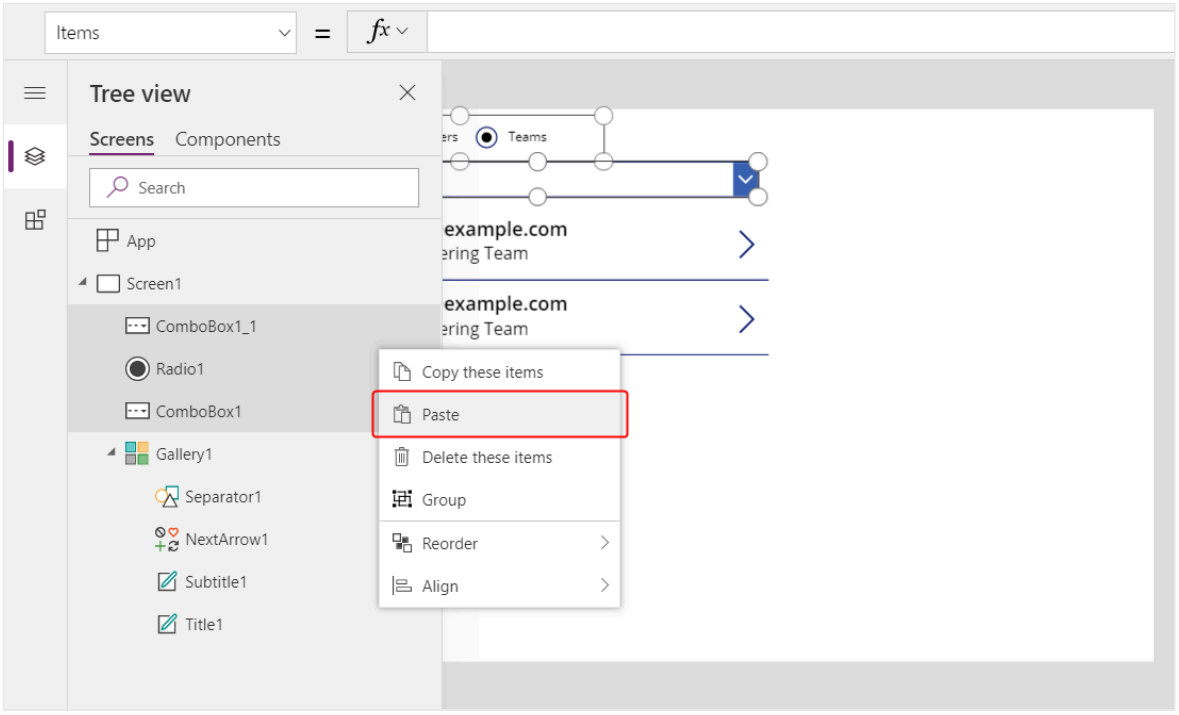
To add this capability to the app:

1. In the **Tree view** pane, select the **Radio** control and the two **Combo box** controls at the same time.

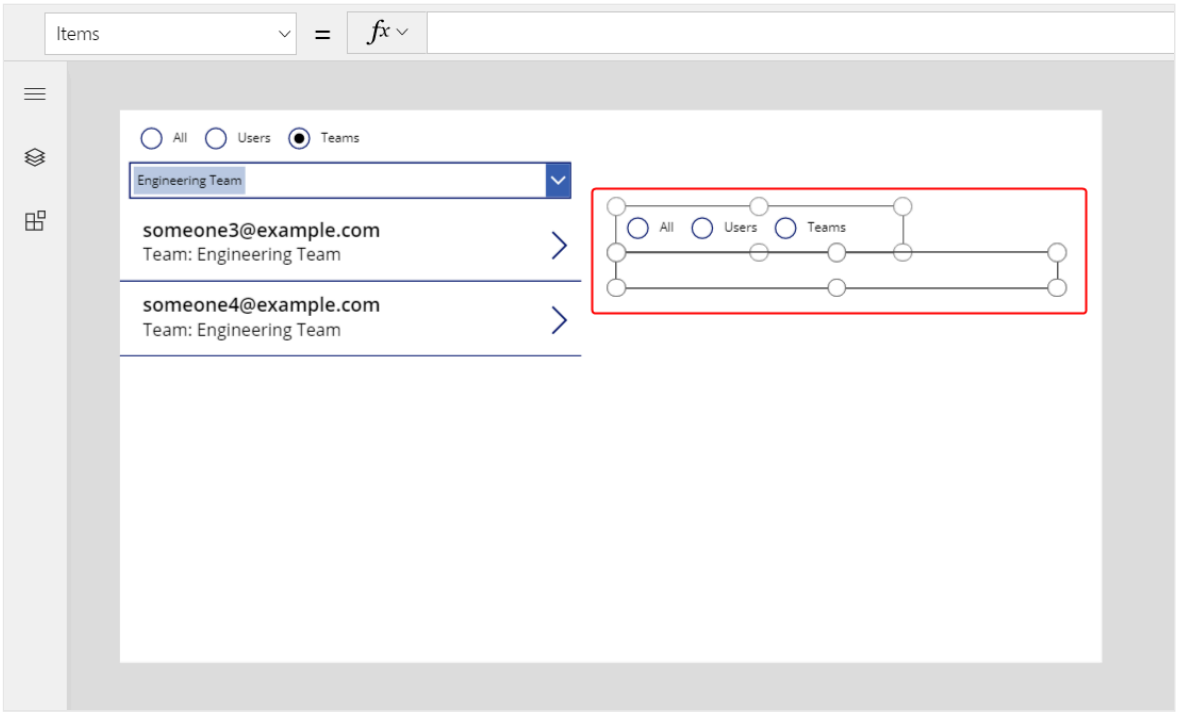
2. On the ellipsis menu, select **Copy these items**.



3. On the same menu, select **Paste**.

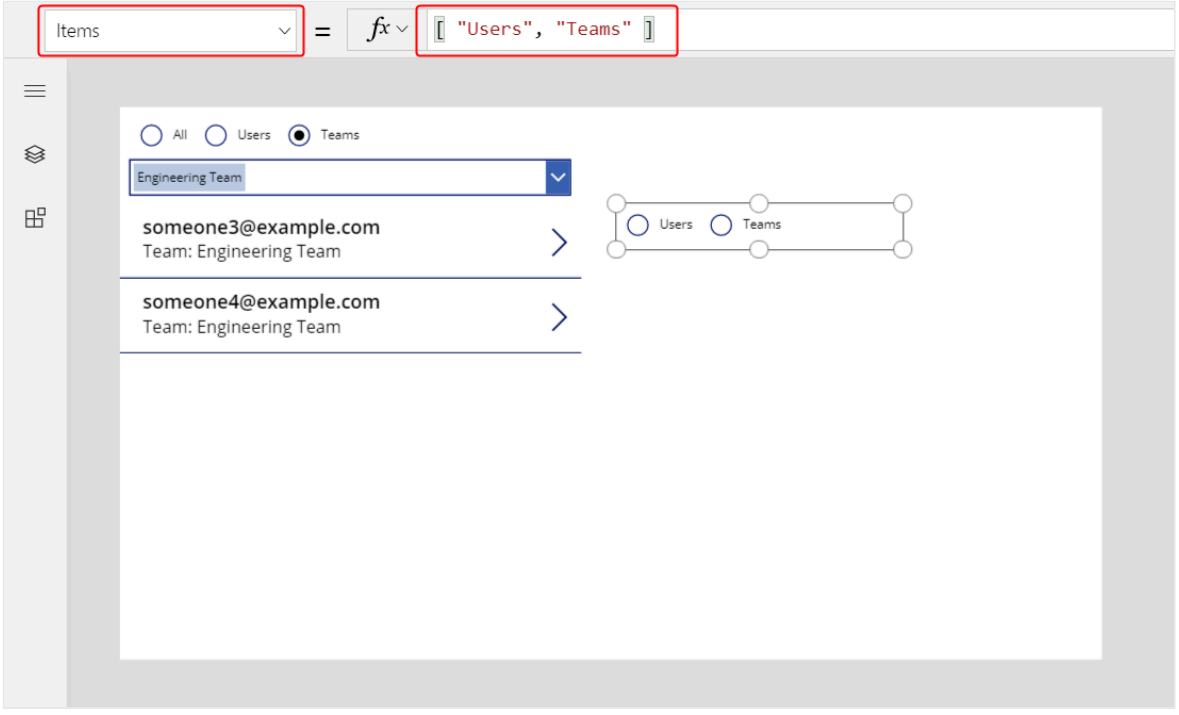


4. Move the copied controls to the right of the gallery.



5. Select the copied **Radio** control, and then change these properties:

   - Items: `[ "Users", "Teams" ]`
   - Default: `If( IsType( Gallery1.Selected.Owner, Users ), "Users", "Teams" )`

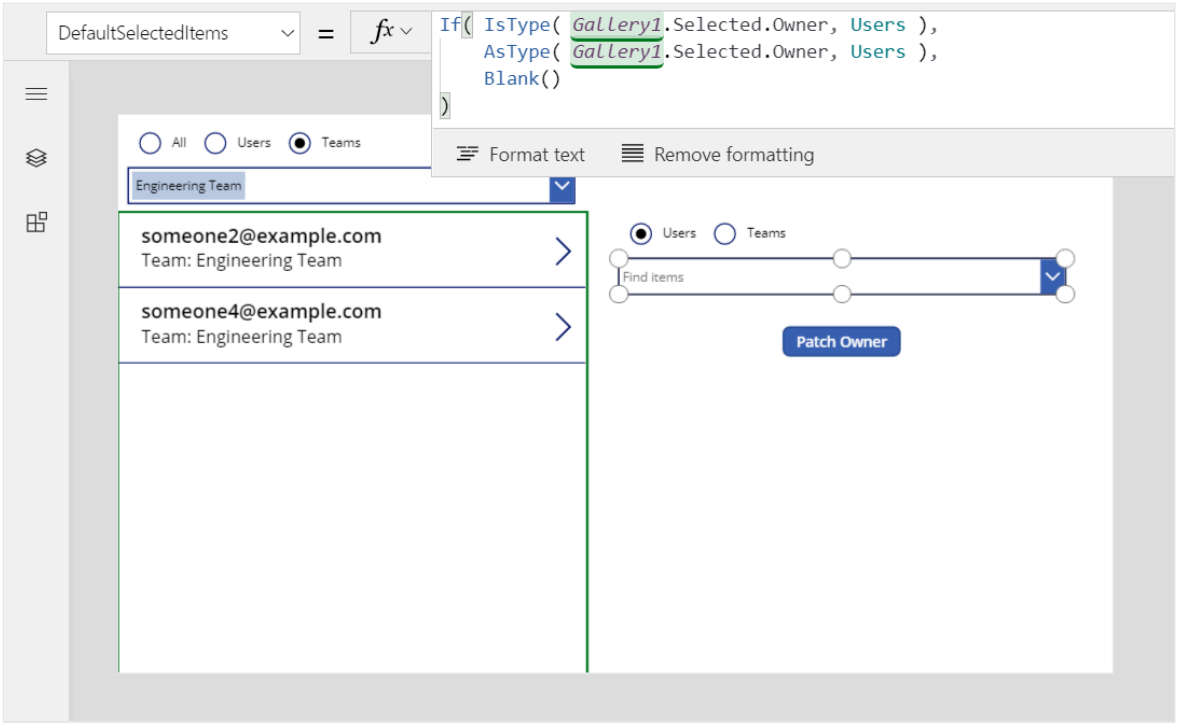


6. In the **Radio** control, select **Users** so that the **Combo box** control that lists users is visible.

7. Select the visible **Combo box** control, and then set the **DefaultSelectedItems** property to this formula:

```
Power Fx
```

```
If( IsType( Gallery1.Selected.Owner, Users ),
    AsType( Gallery1.Selected.Owner, Users ),
    Blank()
)
```



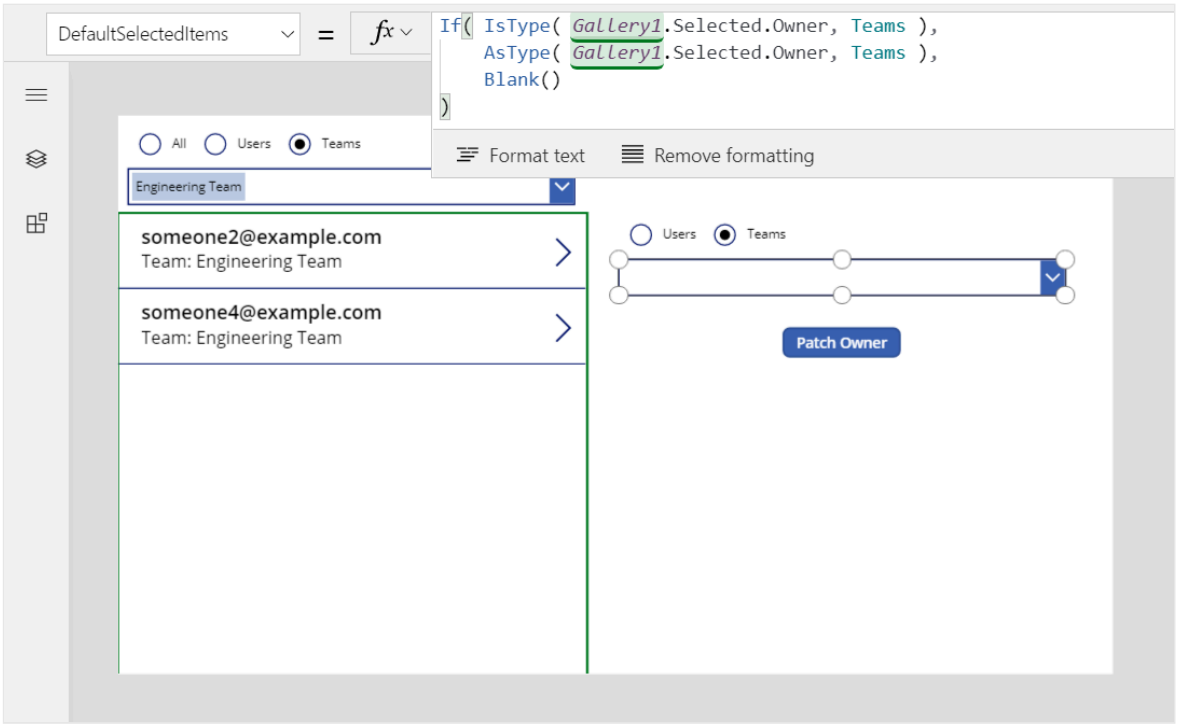
8. In the **Radio** control, select **Teams** so that the **Combo box** control that lists teams is visible.

9. Select the **Radio** control to take selection away from the now-invisible **Combo box** control for users.

10. Select the visible **Combo box** control for teams, and then set its **DefaultSelectedItems** property to this formula:

Power Fx

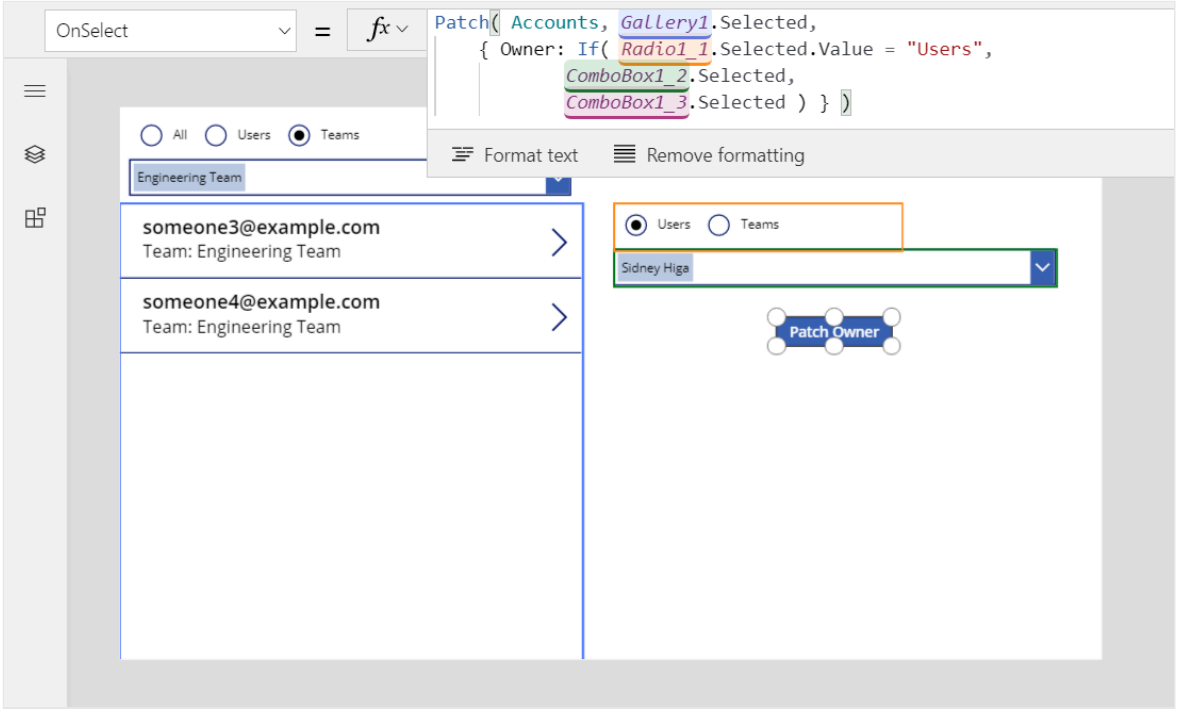
```
If( IsType( Gallery1.Selected.Owner, Teams ),
    AsType( Gallery1.Selected.Owner, Teams ),
    Blank()
)
```



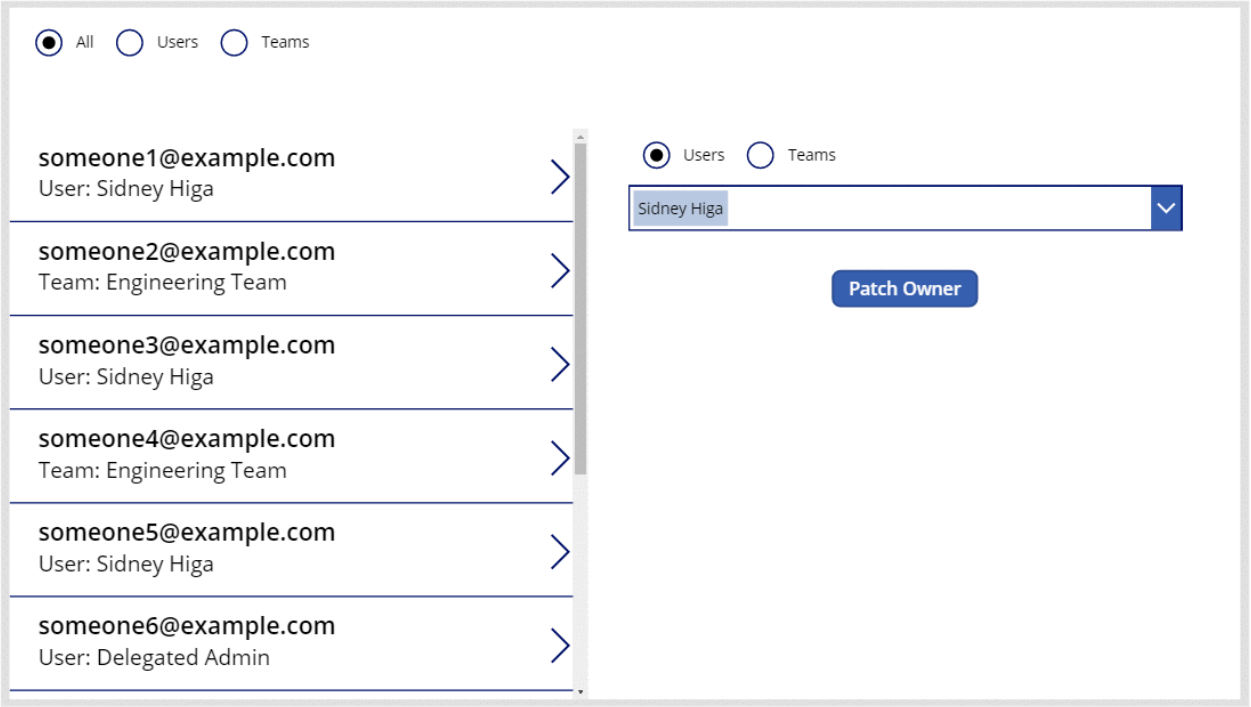
11. Insert a **Button** control, move it under the **Combo box** control, and then set the button's **Text** property to `"Patch Owner"`.

12. Set the **OnSelect** property of the button to this formula:

```
Power Fx

Patch( Accounts, Gallery1.Selected,
    { Owner: If( Radio1_1.Selected.Value = "Users",
                 ComboBox1_2.Selected,
                 ComboBox1_3.Selected ) } )
```



The copied **Radio** and **Combo box** controls show the owner for the currently selected account in the gallery. With the same controls, you can set the owner of the account to any team or user by selecting the button:

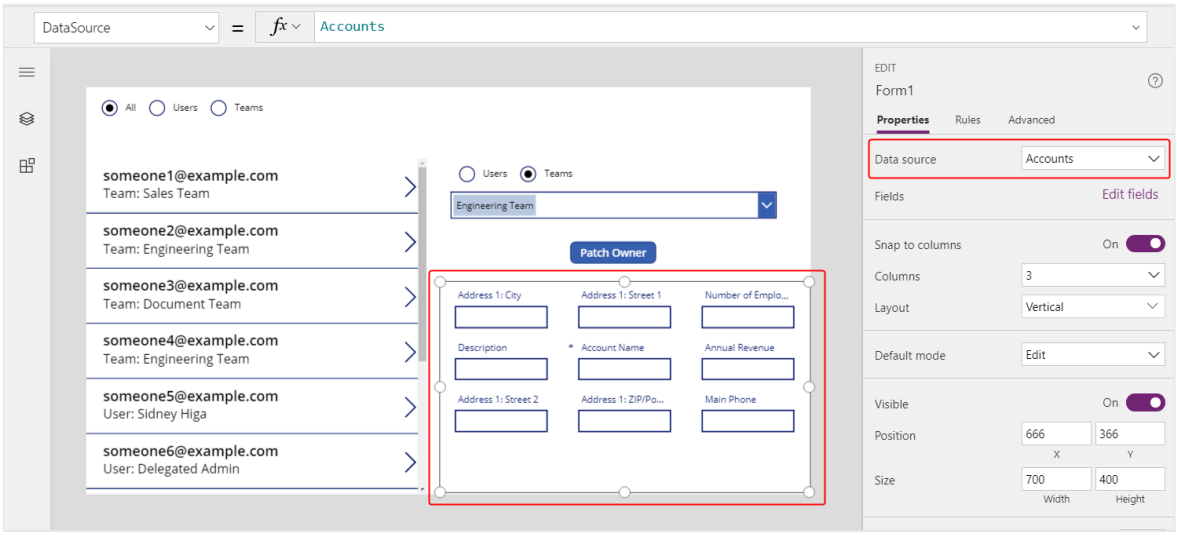


Show the owner by using a form

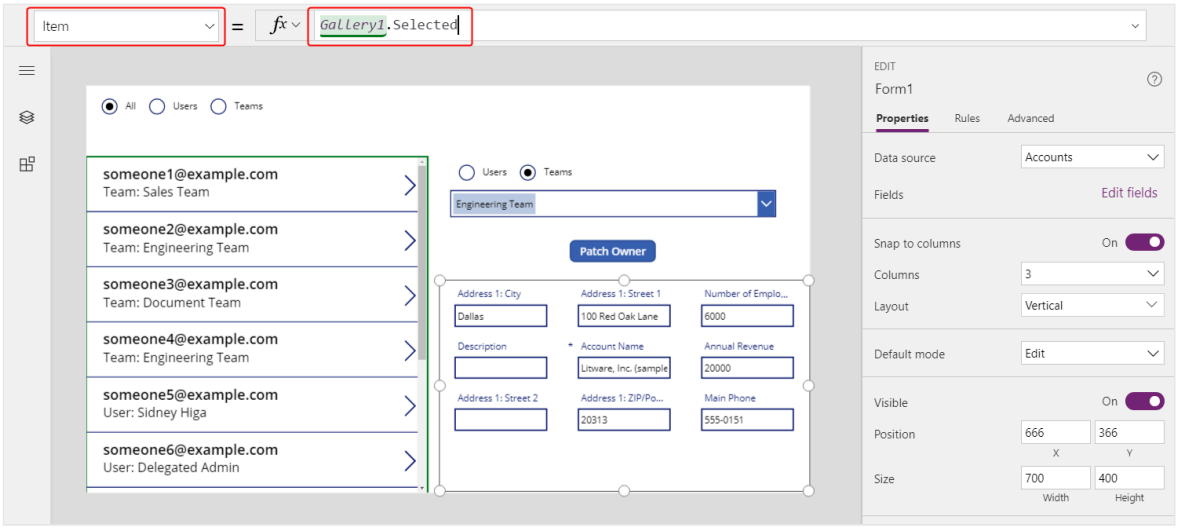
You can show an **Owner** field inside a form by adding a custom card. As of this writing, you can't change the value of the field with a form control.

1. Insert an **Edit form** control, and then resize and move it to the lower-right corner.

2. On the **Properties** tab near the right side of the screen, open the **Data source** list, and then select **Accounts**.

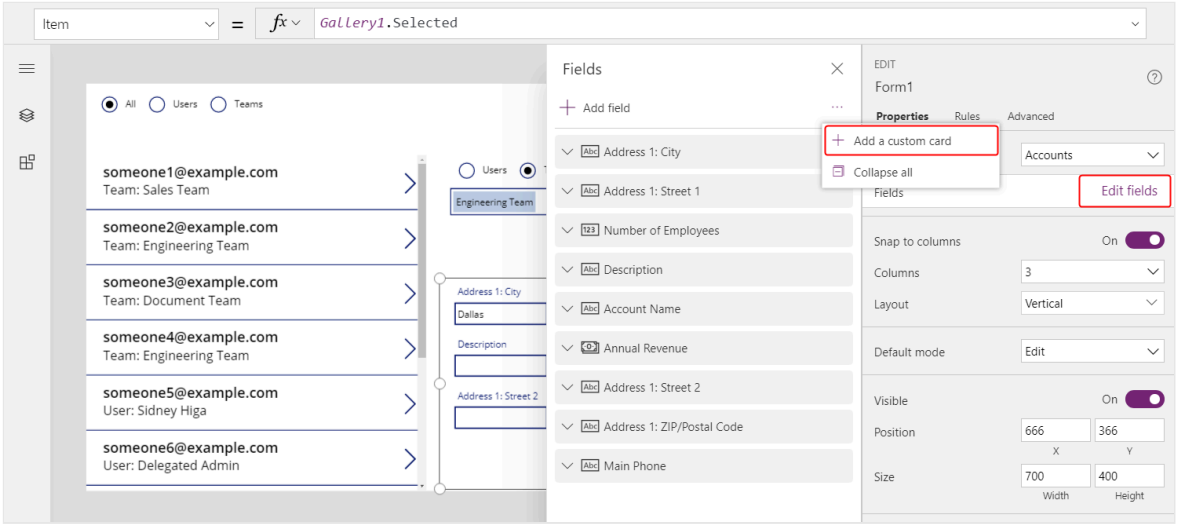


3. Set the form's **Item** property to `Gallery1.Selected`.



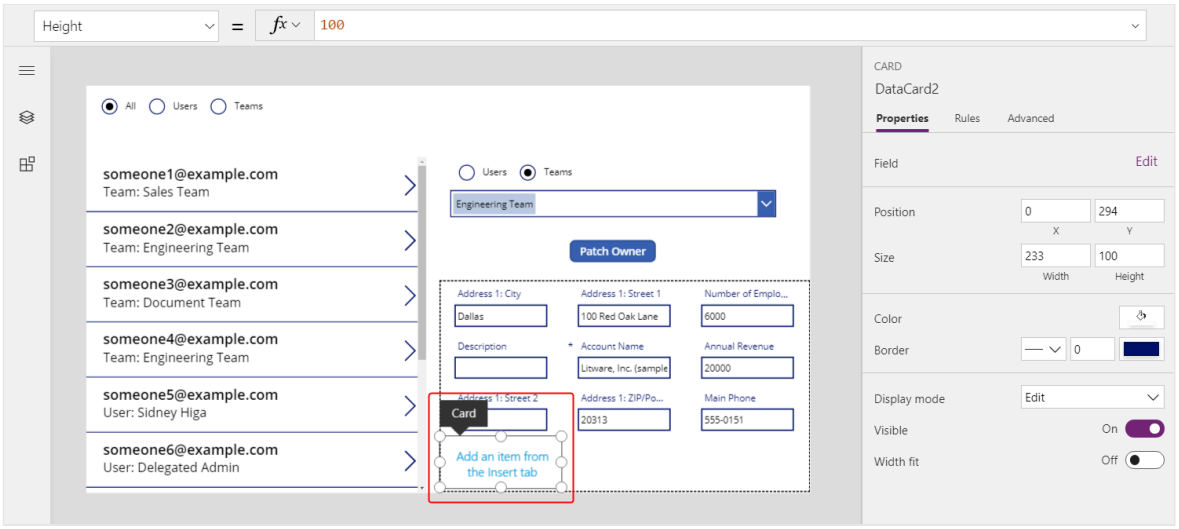
4. On the **Properties** tab near the right side of the screen, select **Edit fields**.

5. In the **Fields** pane, select the ellipsis, and then select **Add a custom card**.



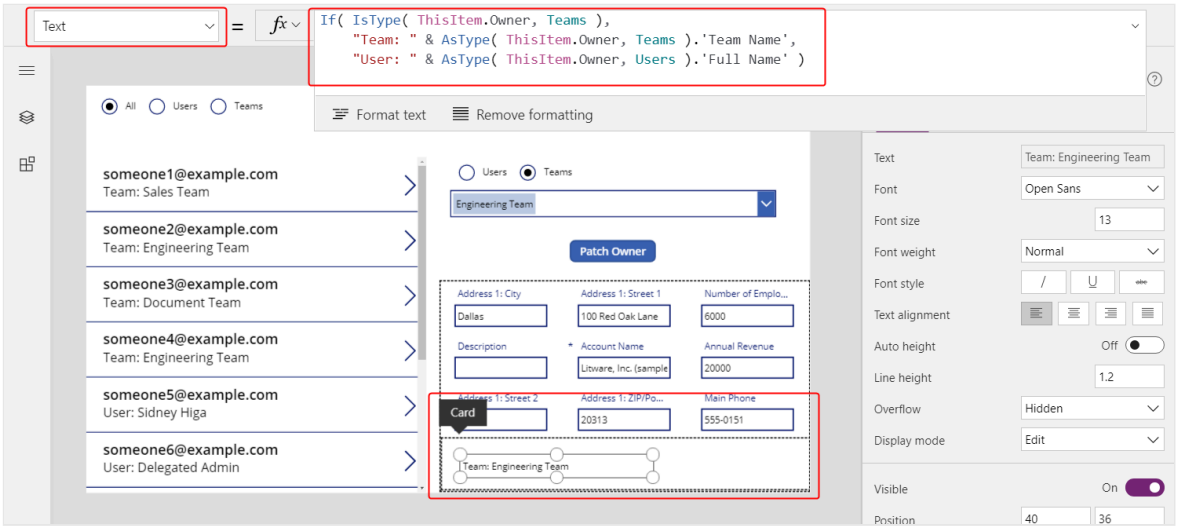
The new card appears at the bottom of the form control.

6. Resize the card as needed to show all the text.

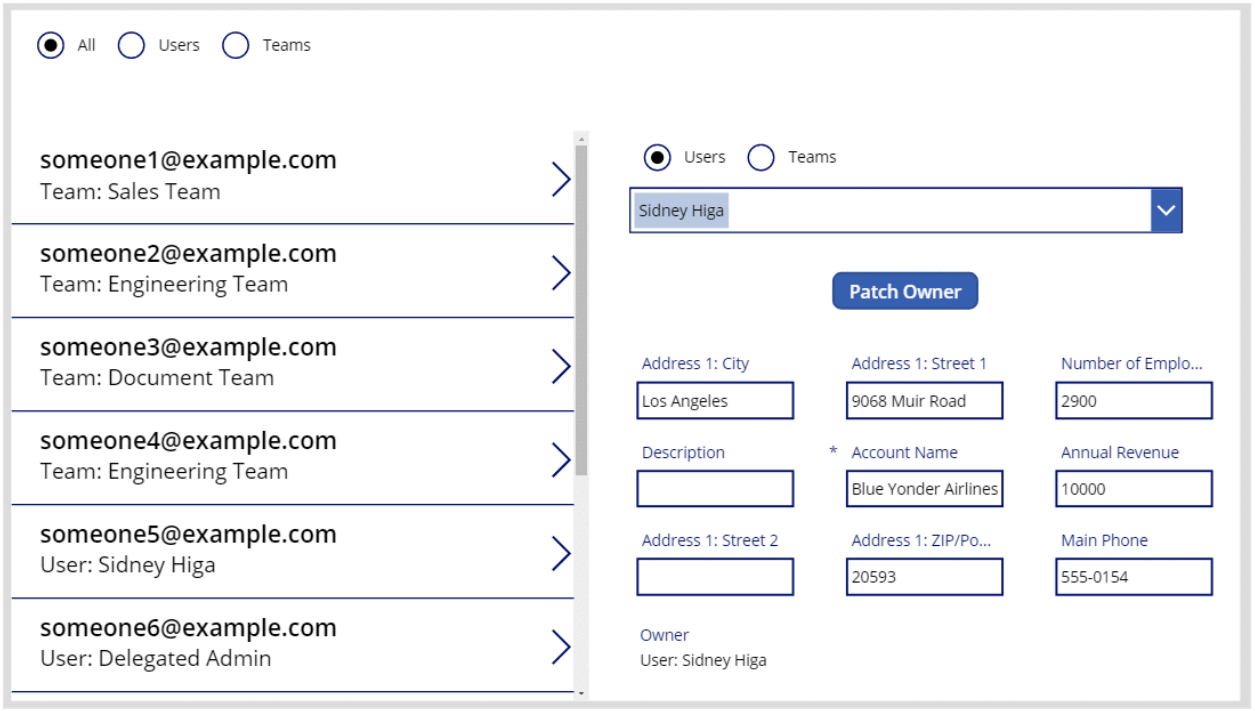


7. Insert a **Label** control into the custom card, and then set the label's **Text** property to the formula that you used in the gallery:

```
Power Fx

If( IsType( ThisItem.Owner, Teams ),
    "Team: " & AsType( ThisItem.Owner, Teams ).'Team Name',
    "User: " & AsType( ThisItem.Owner, Users ).'Full Name' )
```



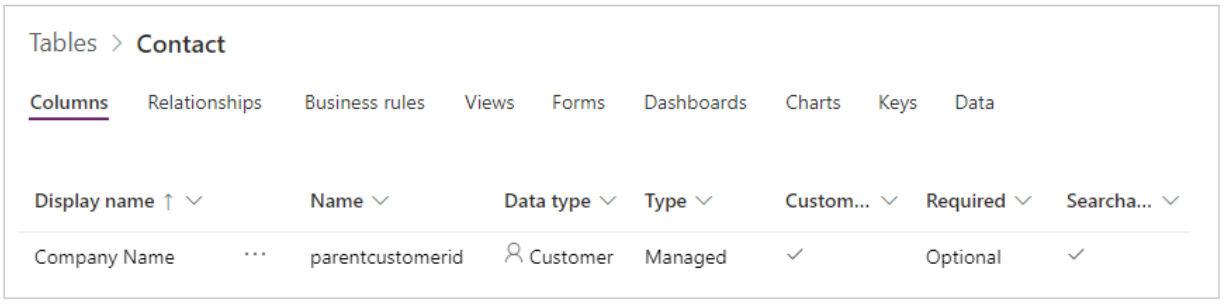
For each selection in the gallery, more fields of the account, including the record's owner, appear in the form. If you change the owner by using the **Patch** button, the form control also shows that change.



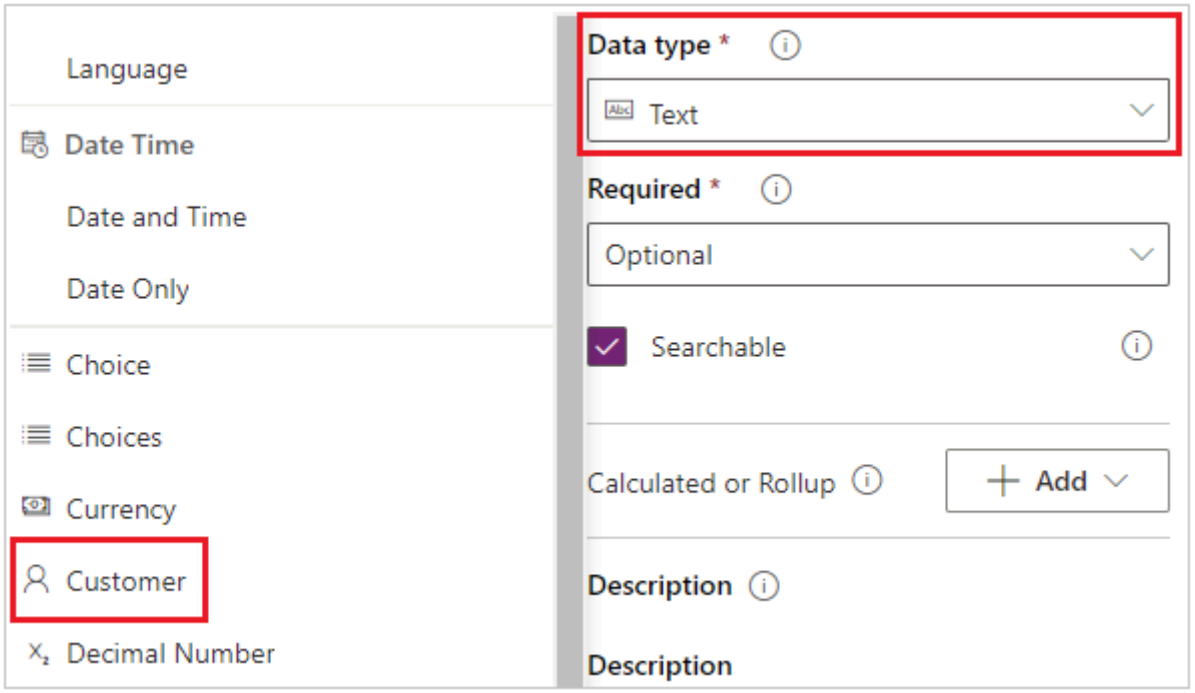
Show the columns of a customer

In Dataverse, the **Customer** lookup column is another polymorphic lookup that's very similar to **Owner**.

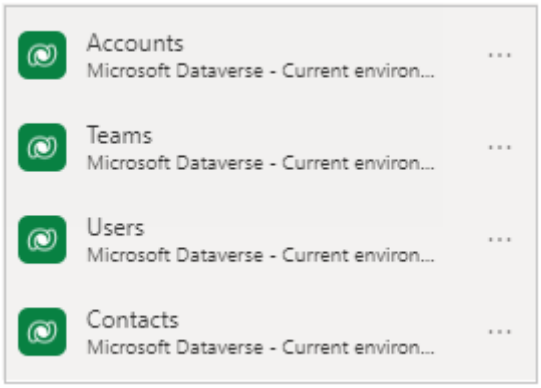
Owner is limited to one per table, but tables can include zero, one, or more **Customer** lookup column. The **Contacts** system table includes the **Company Name** column, which is a **Customer** lookup column.



You can add more **Customer** lookup columns to a table by selecting the **Customer** data type for a new column.



A **Customer** lookup field can refer to a record from either the **Accounts** table or the **Contacts** table. You'll use the **IsType** and **AsType** functions with these tables, so now is a good time to add them as data sources (you can leave **Teams** and **Users** in place).



The treatment of the **Customer** and **Owner** fields is so similar that you can literally copy the app (**File** > **Save as**, and then specify a different name) and make these simple replacements:

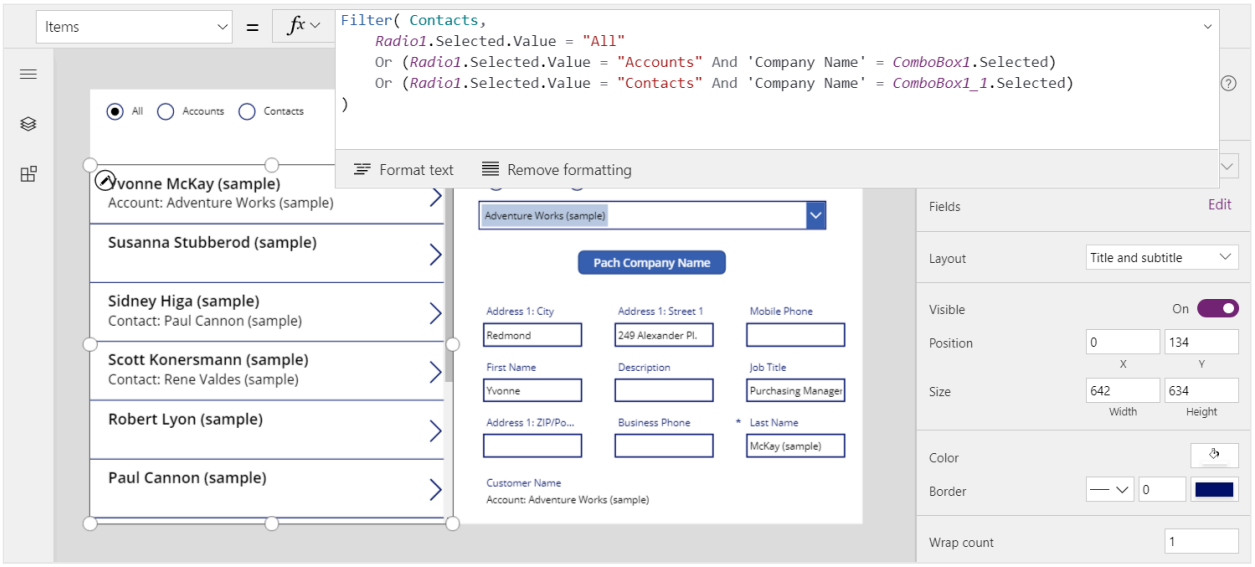
Expand table

Location	Owner **sample**	Customer **sample**
Throughout	**Owner**	**'Customer Name'**
Throughout	**Users**	**Accounts**
Throughout	**Teams**	**Contacts**
Gallery's **Items** property	**Accounts**	**Contacts**
Form's **Items** property	**Accounts**	**Contacts**
The first argument of **Patch** in the button's **OnSelect** property	**Accounts**	**Contacts**
Filter radio's **Items** property	**["All", "Users", "Teams"]**	**["All", "Accounts", "Contacts"]**
Patch radio's **Items** property	**["Users", "Teams"]**	**["Accounts", "Contacts"]**
Combo box's **Visible** property	**"Users"** and **"Teams"**	**"Accounts"** and **"Contacts"**

For example, the new gallery should have this **Items** property:

Power Fx

```
Filter( Contacts,
    Radio1.Selected.Value = "All"
    Or (Radio1.Selected.Value = "Accounts" And 'Company Name' =
ComboBox1.Selected)
    Or (Radio1.Selected.Value = "Contacts" And 'Company Name' =
ComboBox1_1.Selected)
)
```



Two important differences between **Customer** and **Owner** require an update to the formulas inside the gallery and the form:

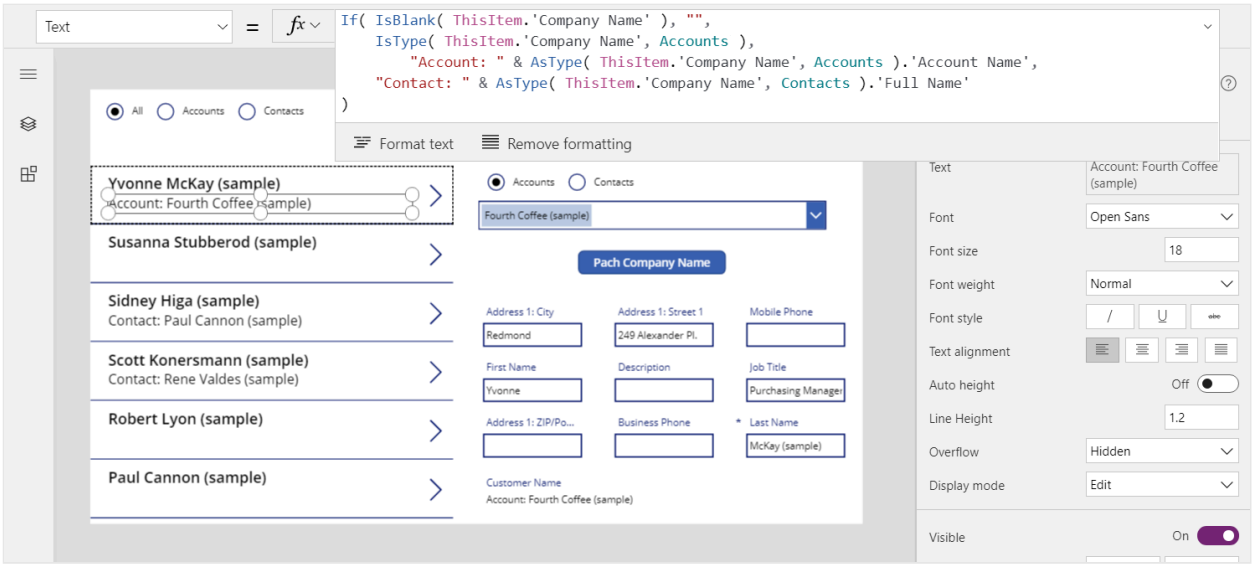
1. One-to-many relationships between **Accounts** and **Contacts** take precedence when you refer to these table types by name. Instead of **Accounts**, use **[@Accounts]**; instead of **Contacts**, use **[@Contacts]**. By using the global disambiguation operator, you ensure that you're referring to the table type in **IsType** and **AsType**. This problem exists only in the record context of the gallery and form controls.

2. The **Owner** field must have a value, but **Customer** fields can be *blank*. To show the correct result without a type name, test for this case with the **IsBlank** function, and show an empty text string instead.

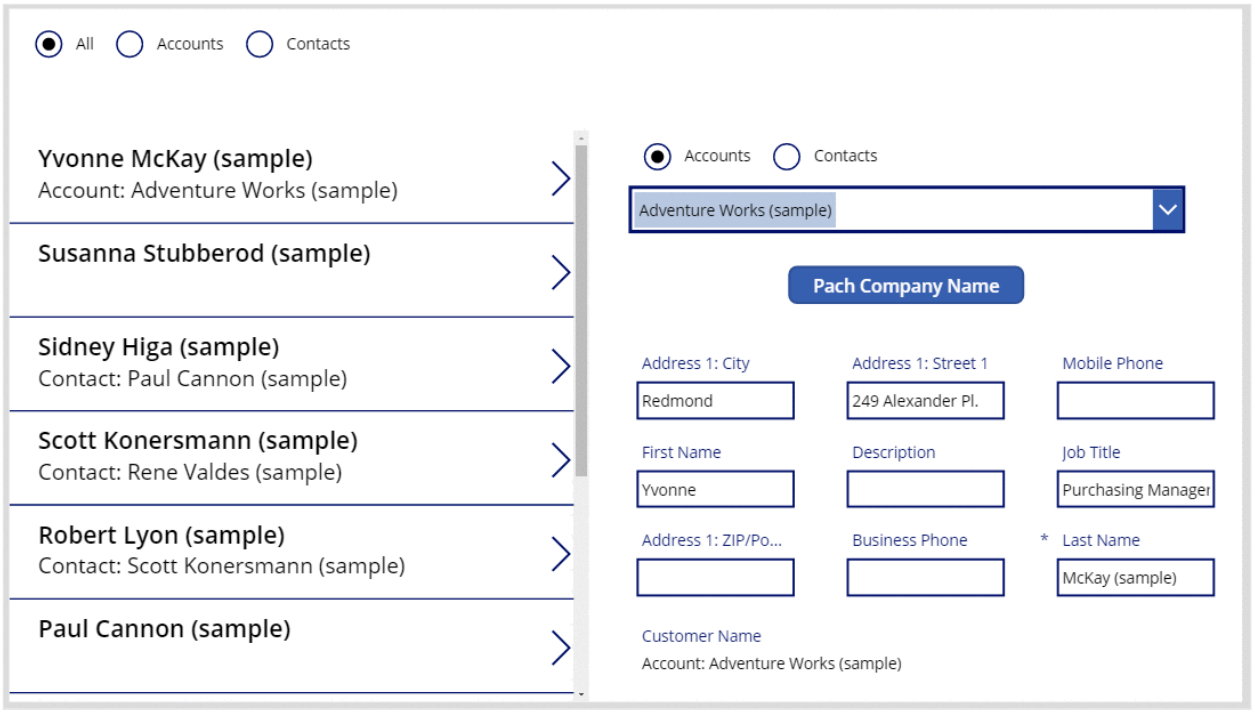
Both of these changes are in the same formula, which appears in the custom card in the form, as well as in the **Text** property of the gallery's label control:

Power Fx

```
If( IsBlank( ThisItem.'Company Name' ), "",
    IsType( ThisItem.'Company Name', Accounts ),
        "Account: " & AsType( ThisItem.'Company Name', Accounts ).'Account
Name',
    "Contact: " & AsType( ThisItem.'Company Name', Contacts ).'Full Name'
)
```



With these changes, you can view and change the **Company Name** field in the **Contacts** table.



Understand Regarding lookup columns

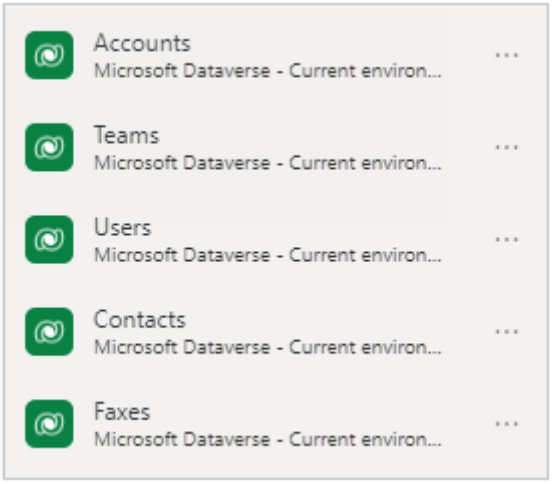
The **Regarding** lookup column differs a little from those that you've already worked with in this topic. You'll start by applying the patterns that this topic described earlier, and then you'll learn other tricks.

You can start simply with the **Faxes** table. This table has a polymorphic **Regarding** lookup column, which can refer to **Accounts**, **Contacts**, and other tables. You can take the app for **Customers** and modify it for **Faxes**.

Expand table

Location	Customer **sample**	Faxes **sample**
Throughout	**'Customer Name'**	**Regarding**
Gallery's **Items** property	**Contacts**	**Faxes**
Form's **Items** property	**Contacts**	**Faxes**
The first argument of **Patch** in the button's **OnSelect** property	**Contacts**	**Faxes**

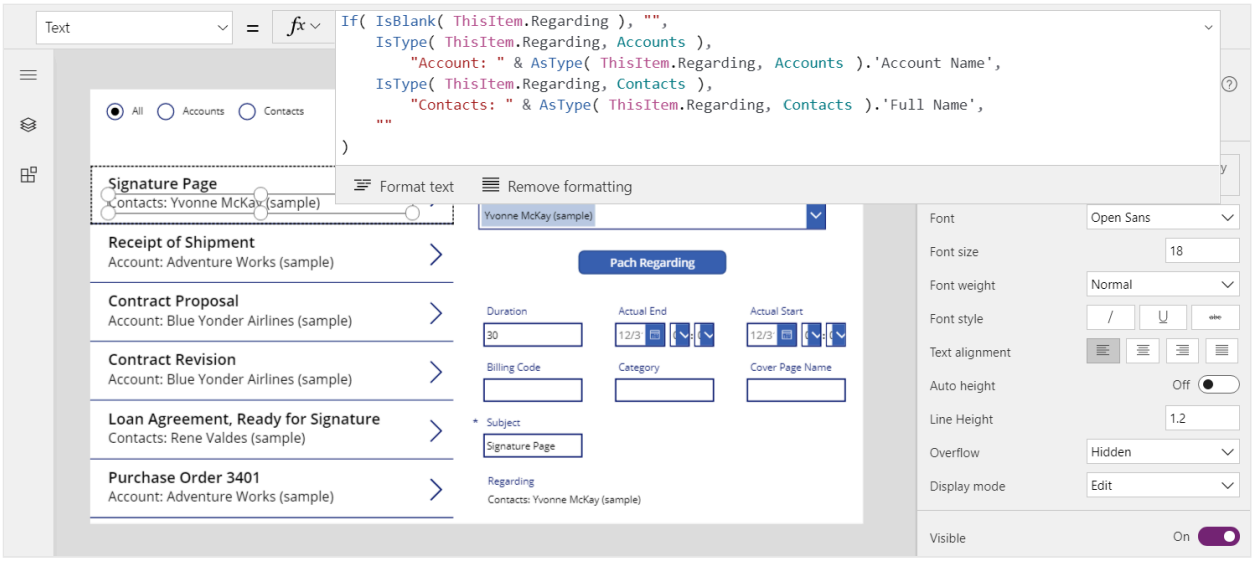
Again, you'll need to add a data source: this time for **Faxes**. On the **View** tab, select **Data sources**:



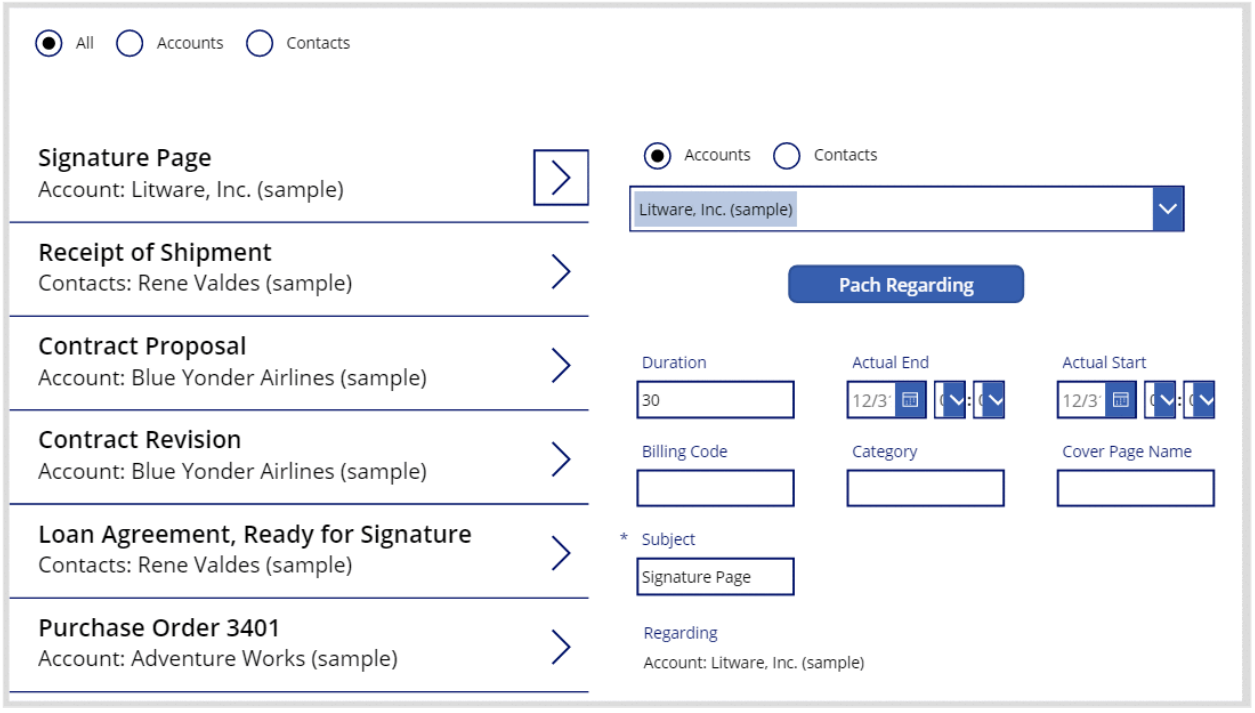
An important difference for **Regarding** is that it isn't limited to **Accounts** and **Contacts**. In fact, the list of tables is extensible with custom tables. Most of the app can accommodate this point without modification, but you must update the formula for the label in the gallery and the form:

Power Fx

```
If( IsBlank( ThisItem.Regarding ), "",
    IsType( ThisItem.Regarding, Accounts ),
        "Account: " & AsType( ThisItem.Regarding, Accounts ).'Account Name',
    IsType( ThisItem.Regarding, Contacts ),
        "Contacts: " & AsType( ThisItem.Regarding, Contacts ).'Full Name',
    ""
)
```



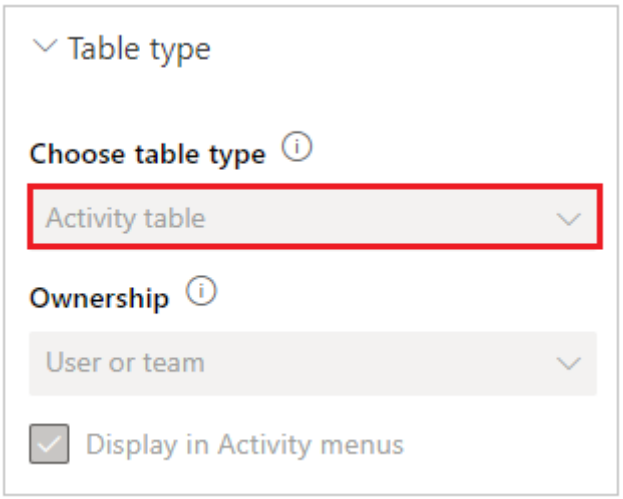
After you make these changes, you work with the **Regarding** lookup just as you did the **Owner** and **Customer** lookups.



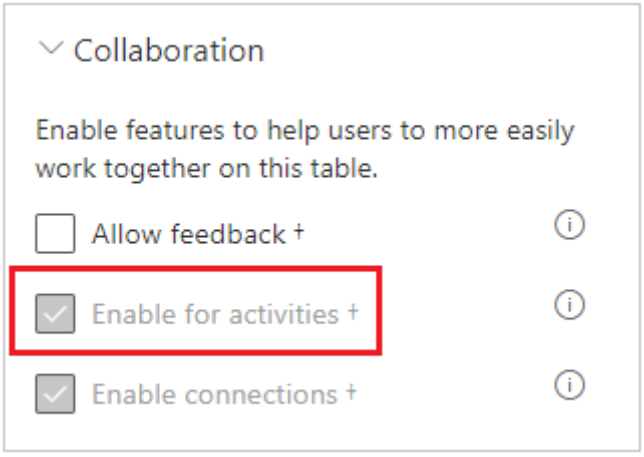
Understand Regarding relationships

Regarding differs from **Owner** and **Customer** because the former involves a many-to-one relationship. By definition, a reverse, one-to-many relationship allows you to write **First(Accounts).Faxes**.

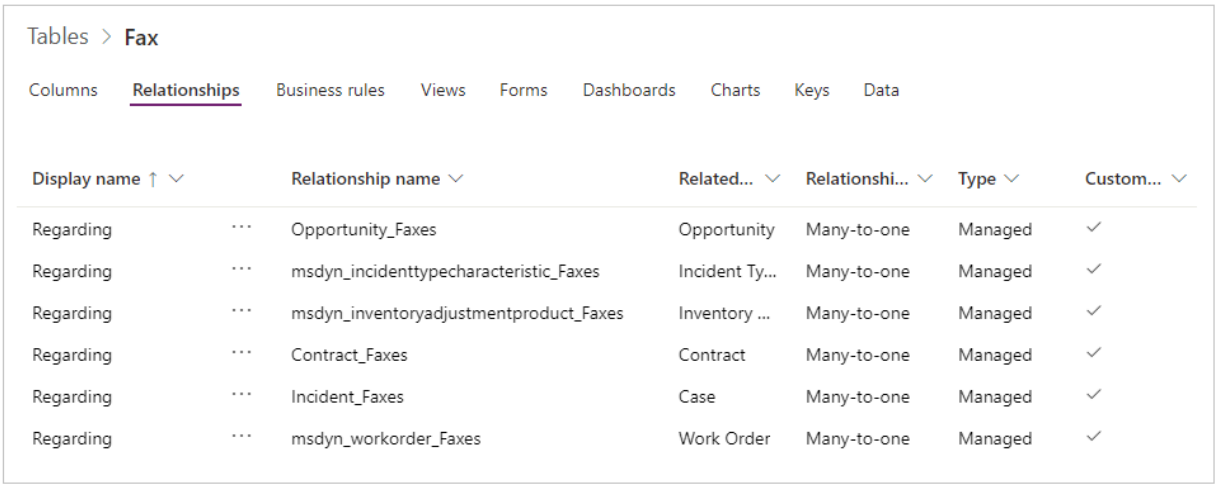
Let's back up and look at the table definitions. In Dataverse, tables such as **Faxes**, **Tasks**, **Emails**, **Notes**, **Phone Calls**, **Letters**, and **Chats** are designated as *activities*. You can also create your own custom activity tables. When you view or create an activity table, its settings appear under **More settings**.



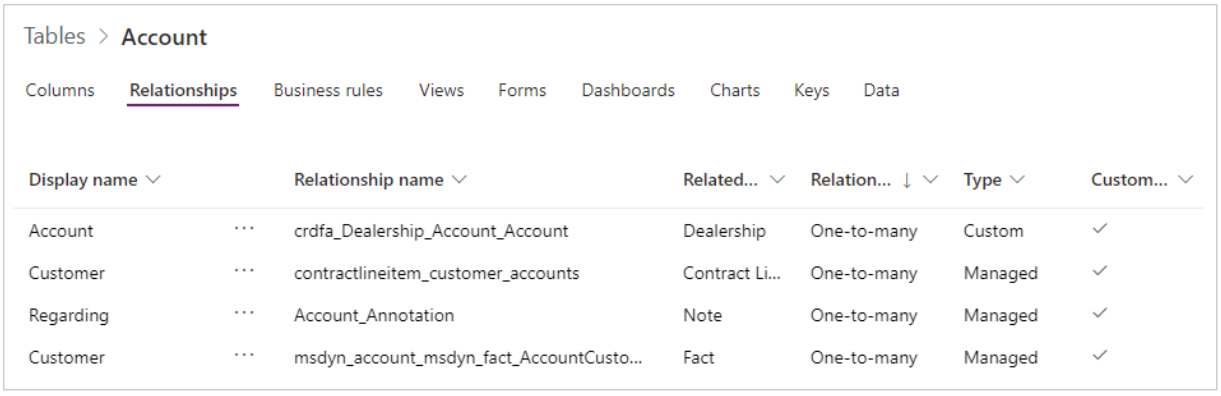
Other tables can be related to an activity table if they're enabled as an *activity task* in the table's settings. **Accounts**, **Contacts**, and many other standard tables are so designated (again, under **More settings**).



All activity tables and activity-task tables have an implied relationship. If you change the filter to **All** at the top of the screen, select the **Faxes** table, and then select the **Relationships** tab, all tables that can be a target of a **Regarding** lookup appear.



If you show the relationships for the **Accounts** table, all the tables that can be a source of a **Regarding** lookup field appear.

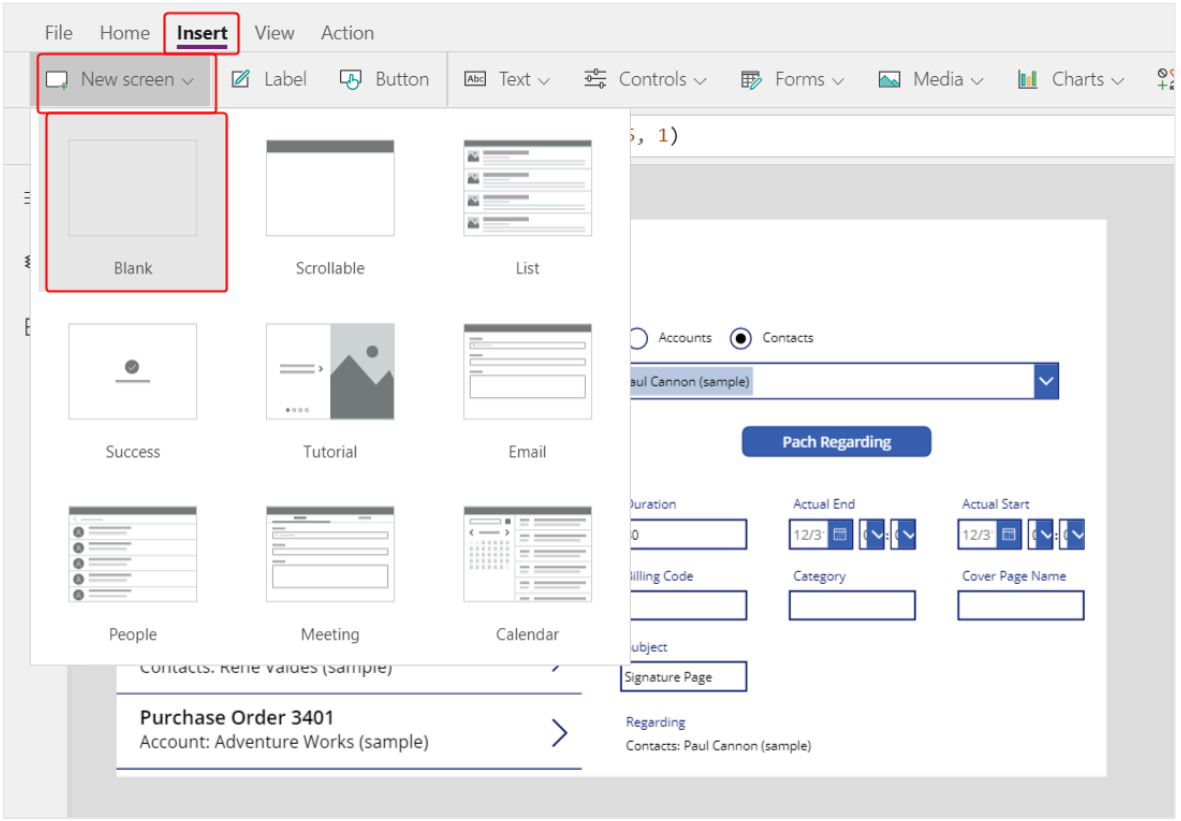


What does it all mean?

- When you write formulas, you must consider that the list of activity tables isn't fixed, and you can create your own. The formula must appropriately handle an activity table that you didn't expect.
- Activity tasks and activities have a one-to-many relationship. You can easily ask for all faxes that relate to an account.

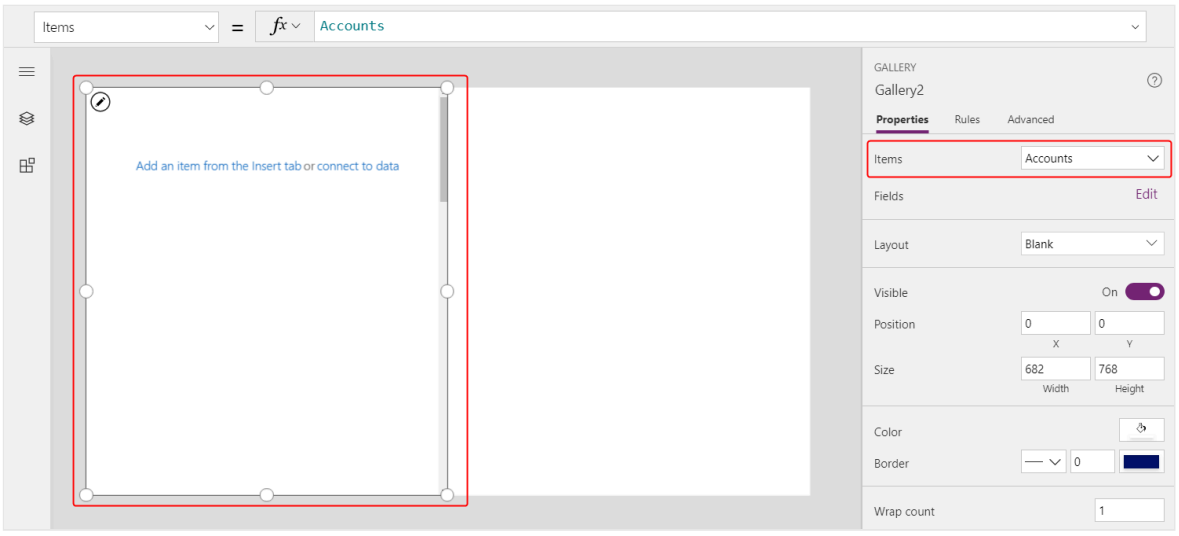
To explore this concept in the app:

1. Add another screen.

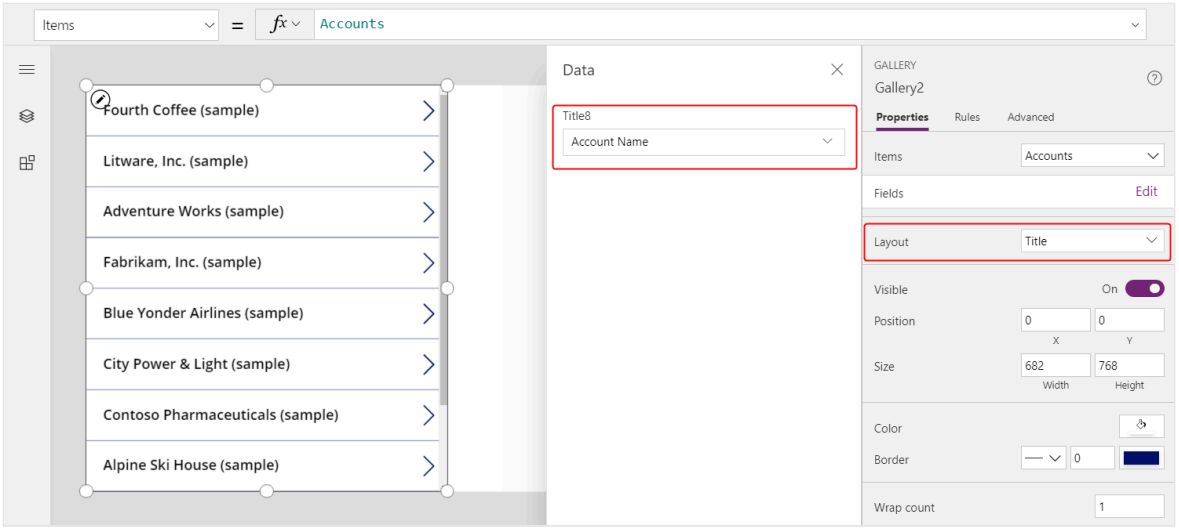


2. Insert a gallery control, resize it, and then move it to the left side of the screen.

3. On the **Properties** tab near the right side of the screen, set the gallery's **Items** to **Accounts**.



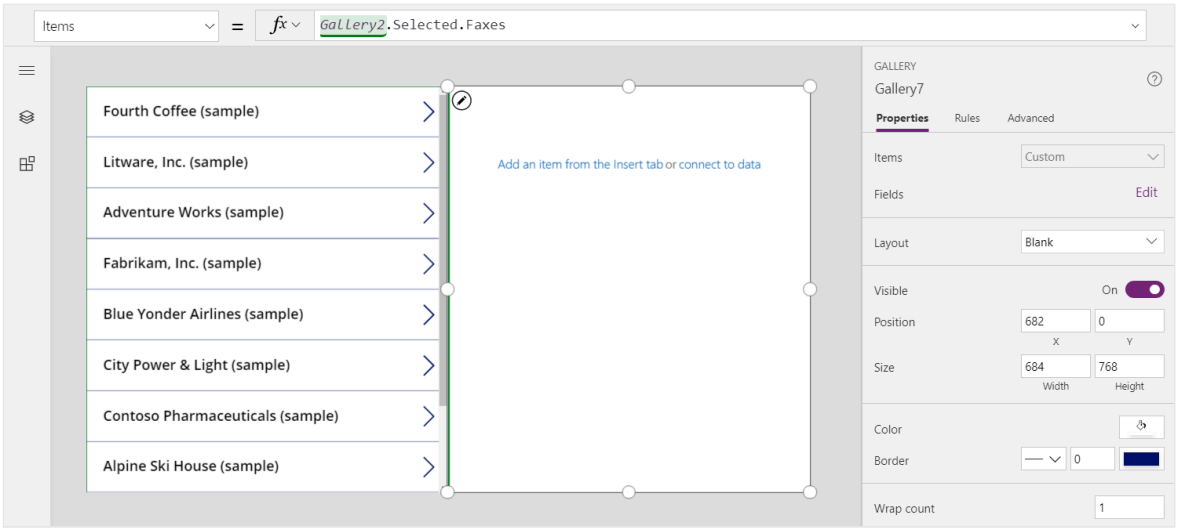
4. Set the gallery's layout to **Title**, and then set the title field to **Account Name**.



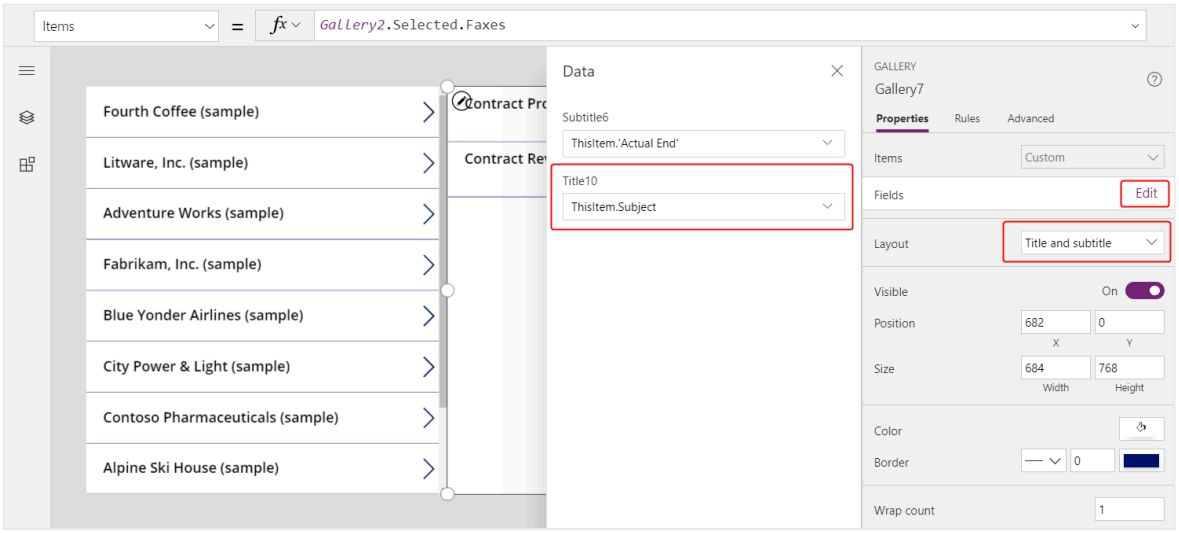
5. Add a second gallery, resize it, and then move it to the right side of the screen.

6. Set the new gallery's **Items** property to `Gallery2.Selected.Faxes`.

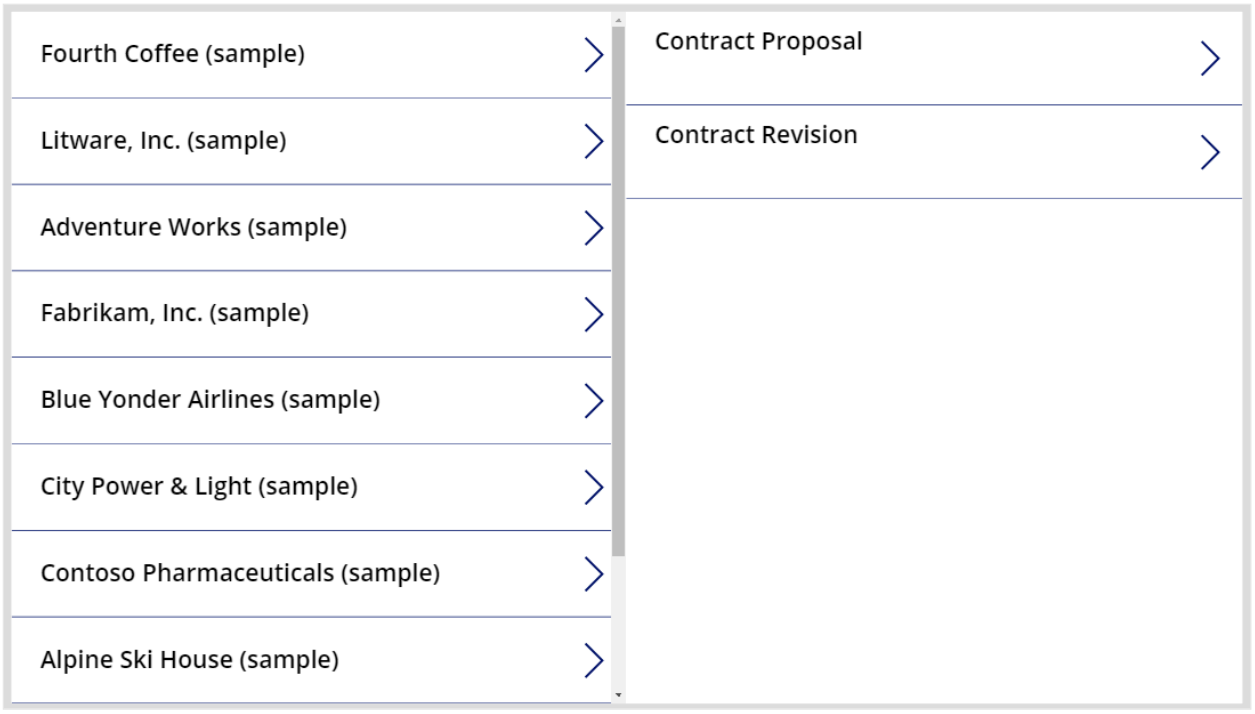
   This step returns the filtered list of faxes for a given account.



7. Set the gallery's layout to **Title and subtitle**, and then set the title field to show the **Subject** field (which might be lowercase **subject**).



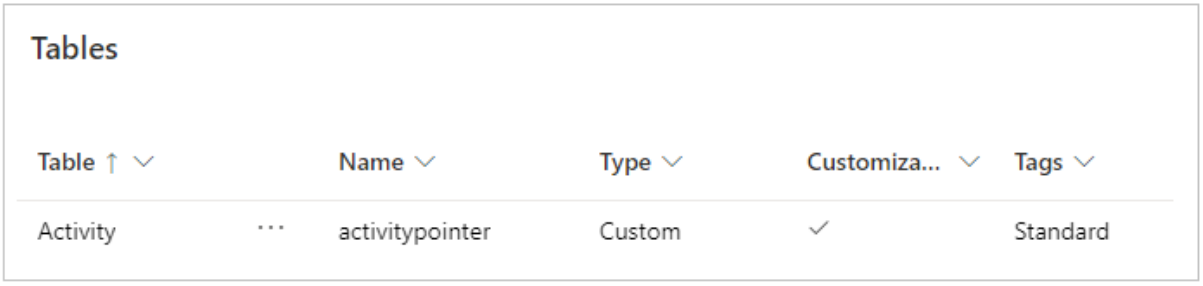
As you select an item in the list of accounts, the list of faxes shows faxes for only that account.



Activity table

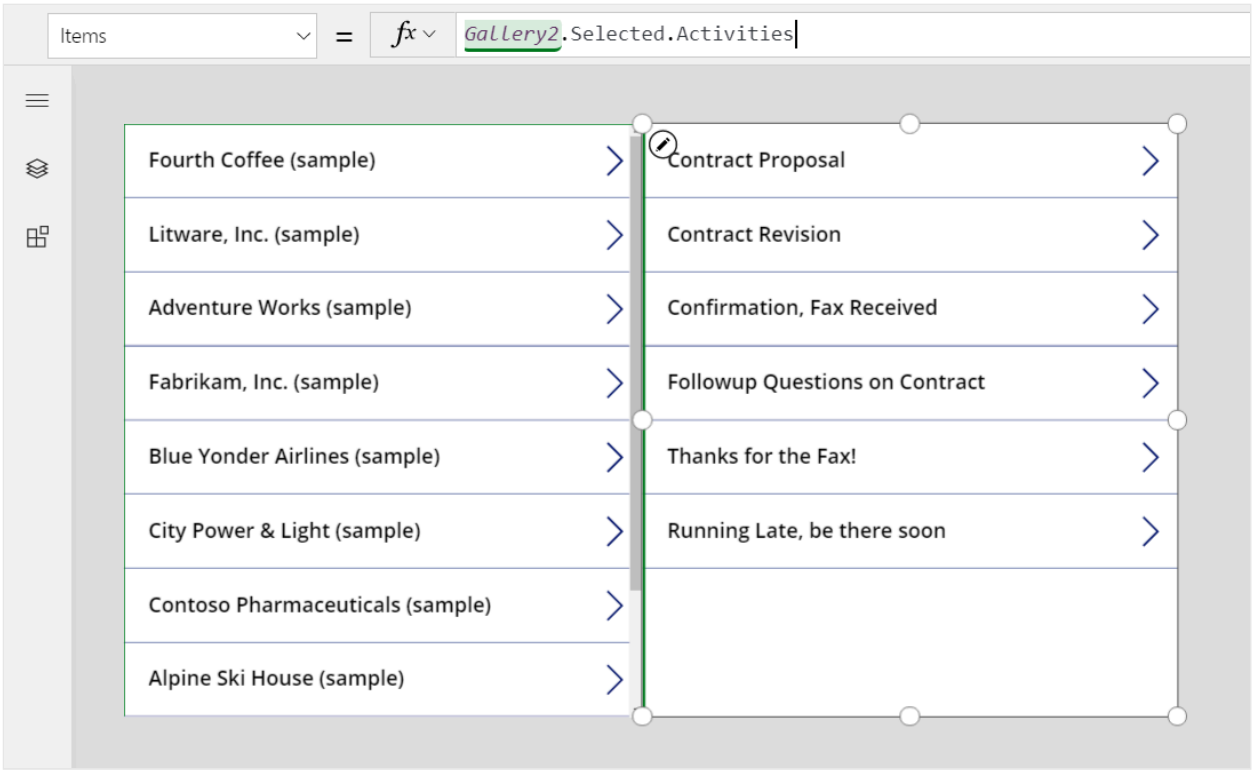
As the previous section describes, you can show all the faxes for an account. However, you can also show all the activities for an account, including faxes, email messages, phone calls, and other interactions.

For the latter scenario, you use the **Activity** table. You can show this table by turning on **All** in the upper-right corner to remove the filter from the list of tables.

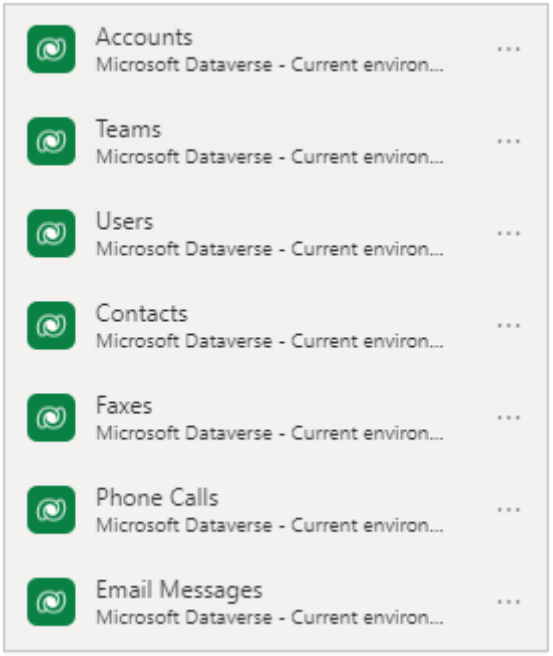


The **Activity** table is special. Whenever you add a record to the **Faxes** table, the system also creates a record in the **Activity** table with the columns that are common across all activity tables. Of those columns, **Subject** is one of the most interesting.

You can show all activities by changing only one line in the previous example. Replace `Gallery2.Selected.Faxes` with `Gallery2.Selected.Activities`.



Records are coming from the **Activity** table, but you can nevertheless use the **IsType** function to identify which kind of activity they are. Again, before you use **IsType** with a table type, you must add the data source.

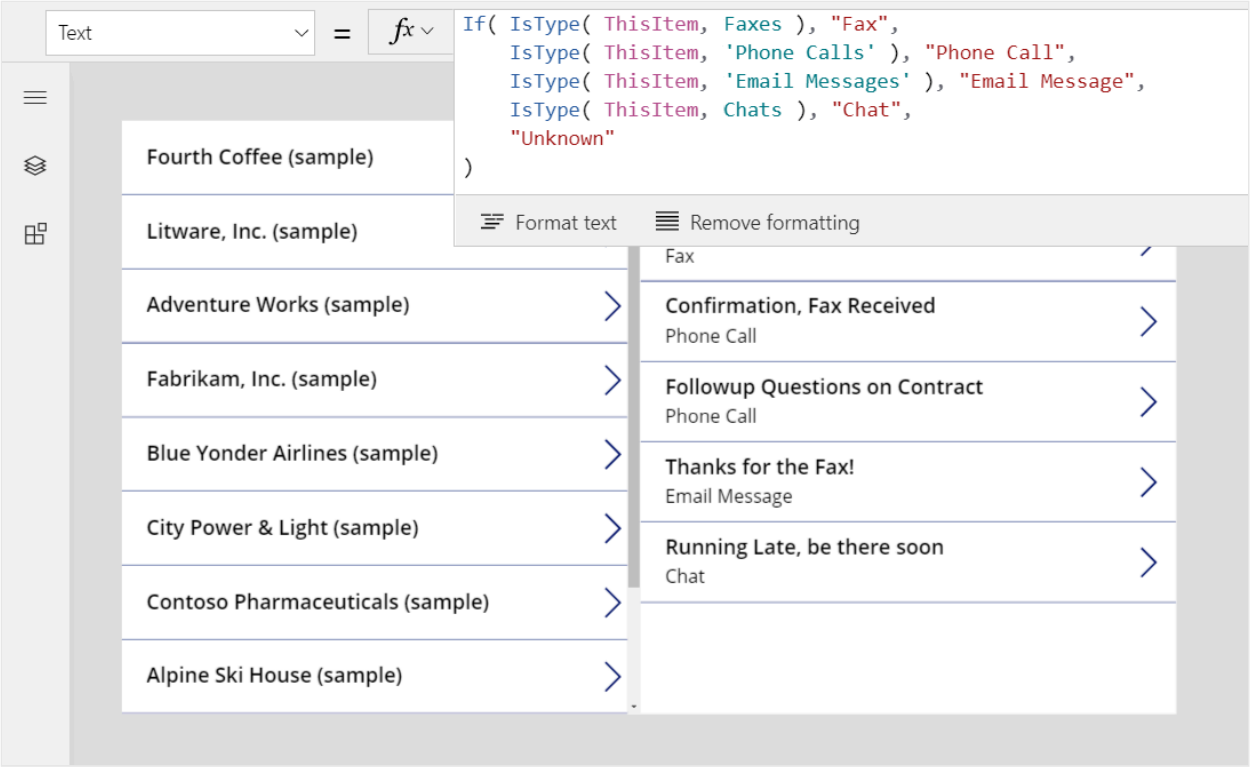


By using this formula, you can show the record type in a label control within the gallery:

Power Fx

```
If( IsType( ThisItem, Faxes] ), "Fax",
    IsType( ThisItem, 'Phone Calls' ), "Phone Call",
    IsType( ThisItem, 'Email Messages' ), "Email Message",
    IsType( ThisItem, Chats ), "Chat",
```

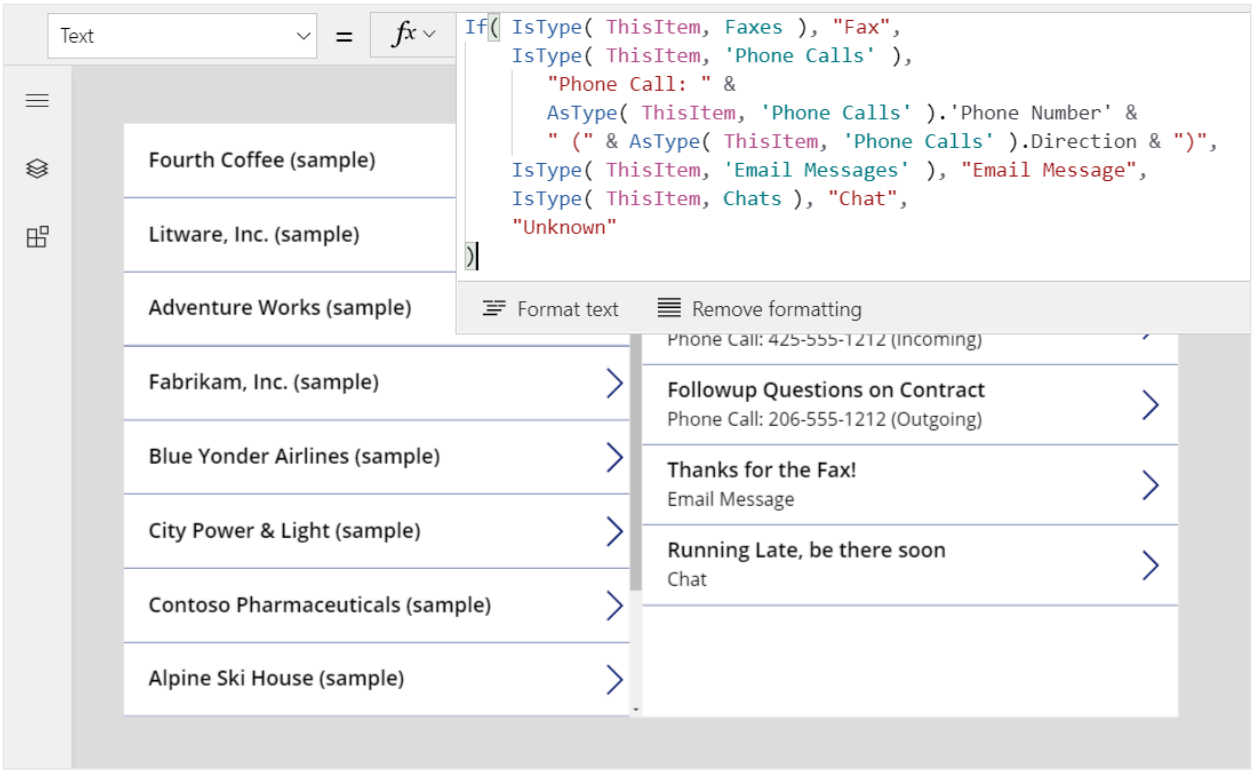
```
    "Unknown"
)
```



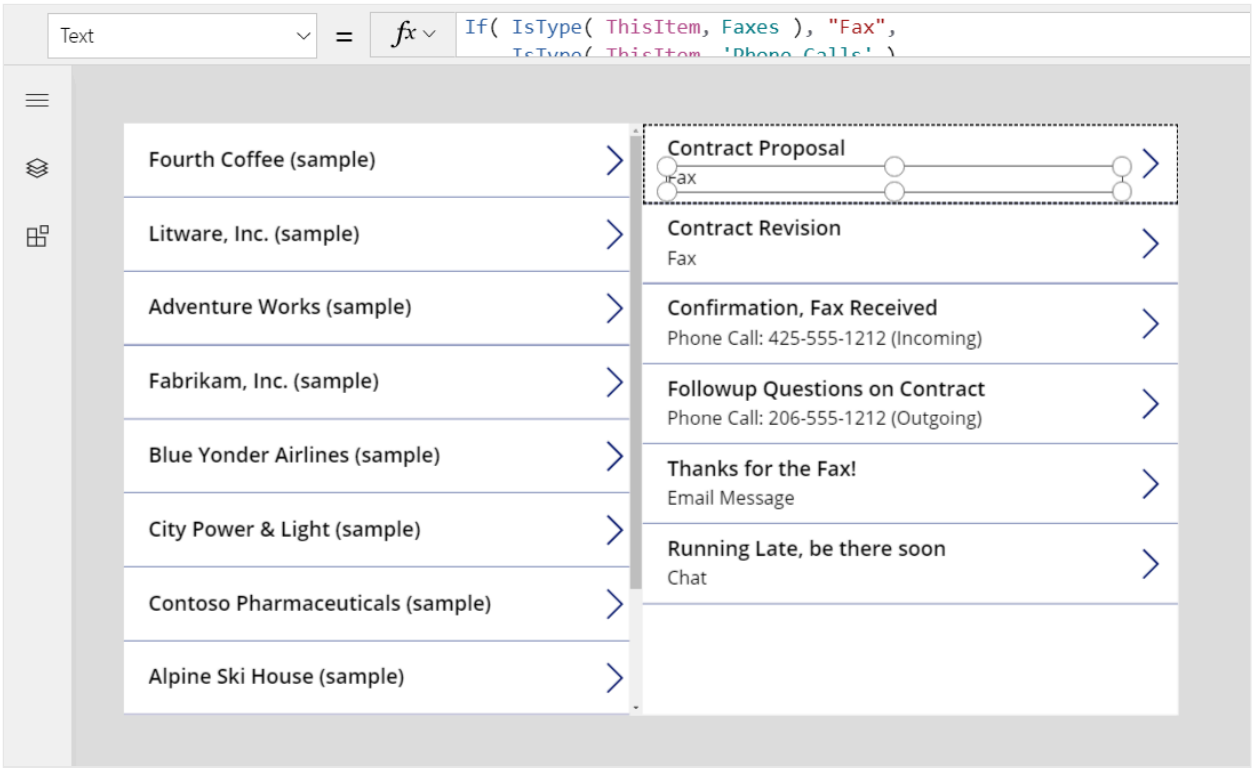
You can also use **AsType** to access the fields of the specific type. For example, this formula determines the type of each activity and, for phone calls, shows the phone number and call direction from the **Phone Numbers** table:

Power Fx

```
If( IsType( ThisItem, Faxes ), "Fax",
    IsType( ThisItem, 'Phone Calls' ),
       "Phone Call: " &
       AsType( ThisItem, 'Phone Calls' ).'Phone Number' &
       " (" & AsType( ThisItem, 'Phone Calls' ).Direction & ")",
    IsType( ThisItem, 'Email Messages' ), "Email Message",
    IsType( ThisItem, Chats ), "Chat",
    "Unknown"
)
```



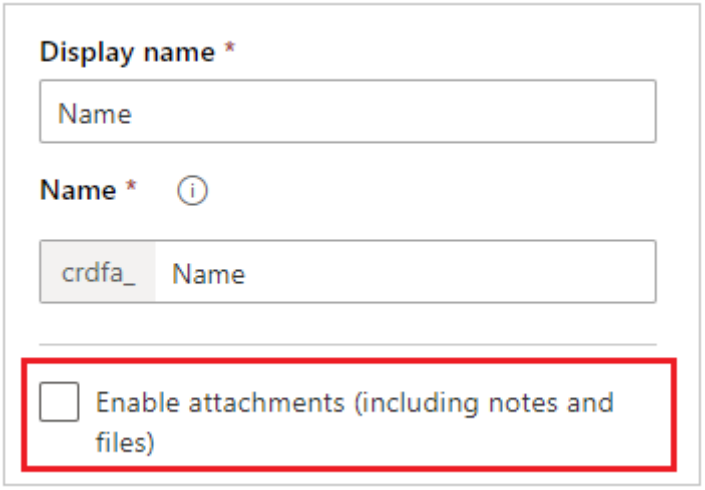
As a result, the app shows a complete list of activities. The **Subject** field appears for all types of activities, whether the formula takes them into account or not. For types of activities that you know about, you can show their type names and type-specific information about each activity.



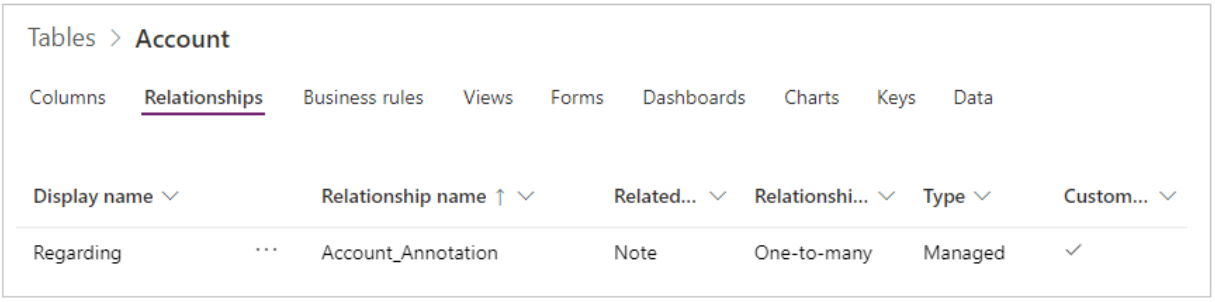
Notes table

So far, all of the **Regarding** examples have been based on activities, but the **Notes** table represents another case.

When you create a table, you can enable attachments.



If you select the check box for enabling attachments, you'll create a **Regarding** relationship with the **Notes** table, as this graphic shows for the **Accounts** table:



Other than this difference, you use the **Regarding** lookup in the same manner in which you use activities. Tables that are enabled for attachments have a one-to-many relationship to **Notes**, as in this example:

```
First( Accounts ).Notes
```

> **Note**
>
> As of this writing, the **Regarding** lookup isn't available for the **Notes** table. You can't read or filter based on the **Regarding** column, and you can't set the column by using **Patch**.
>
> However, the reverse **Notes** one-to-many relationship is available, so you can filter a list of notes for a record that's enabled for attachments. You can also use the Relate function to add a note to a record's **Notes** table, but the note must be created first, as in this example:

```
Relate( ThisItem.Notes, Patch( Notes, Defaults( Notes ), { Title: "A new note" } ) )
```

Activity parties

As of this writing, canvas apps don't support activity parties.

Feedback

Was this page helpful? Yes No

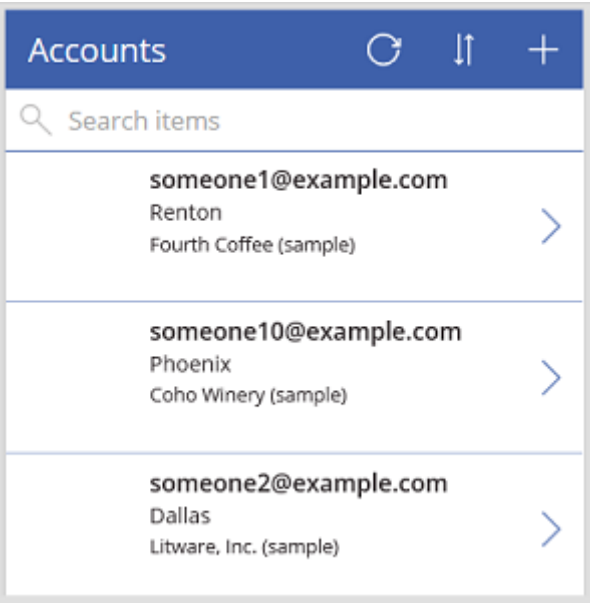
Provide product feedback

Tutorial: Customize a gallery in Power Apps

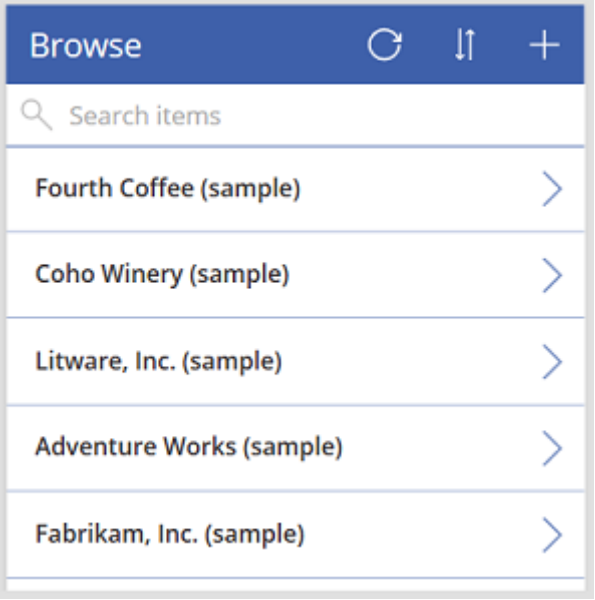
Article • 03/03/2023

In this tutorial, you'll customize a list of records, called a gallery, and make other changes in an app that was generated automatically in Microsoft Power Apps. Users can manage data in the app even if you don't make these changes, but the app will be easier to use if you customize it for your organization's needs.

For example, the gallery for this tutorial matches this graphic by default. The email address is featured more prominently than other types of data, and users can sort and filter the gallery based on text in that address:



However, your users might be more interested in the account name more than the email address, so you'll reconfigure the gallery to highlight, sort, and filter based on the key data for your organization. In addition, you'll change the title of the default screen to differentiate it from the other screens in the app.



You'll also add a scroll bar so that users who don't have touch screens or mouse wheels can browse the entire gallery.

- ✓ Change the layout of the gallery
- ✓ Change the type of data that appears in the gallery
- ✓ Change the columns by which users can sort and search the data
- ✓ Change the screen title
- ✓ Show a scroll bar

This tutorial starts with an app that was generated from a specific data source. However, the same concepts apply to any app that you generate in Power Apps, whether from a list created using Microsoft Lists, a SharePoint library, an Excel table, or some other data source.

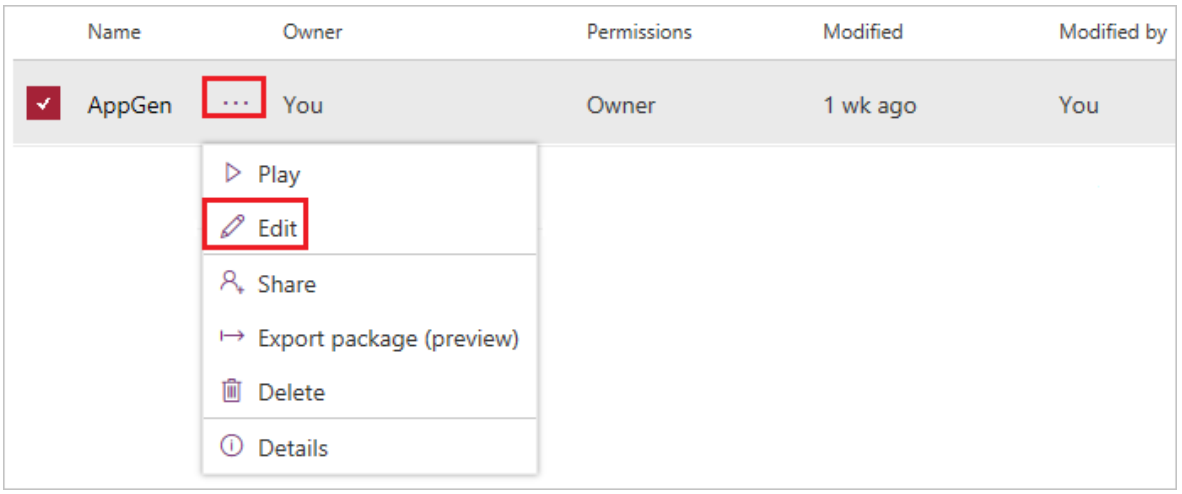
If you're not signed up for Power Apps, sign up for free before you start.

Prerequisites

Generate an app from the **Accounts** table of Microsoft Dataverse.

Open the generated app

1. Sign in to Power Apps, and then select **Apps** near the left edge.
2. Find the app that you generated, select the ellipsis icon (...) for it, and then select **Edit**.

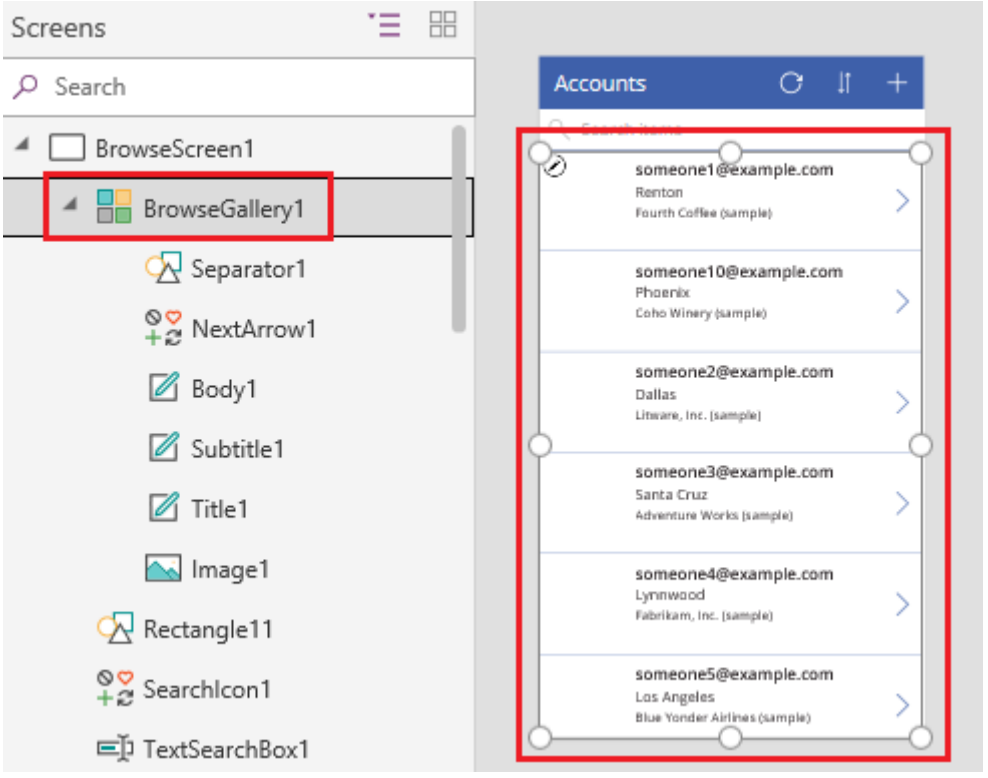


3. If the **Welcome to Power Apps Studio** dialog box appears, select **Skip**.

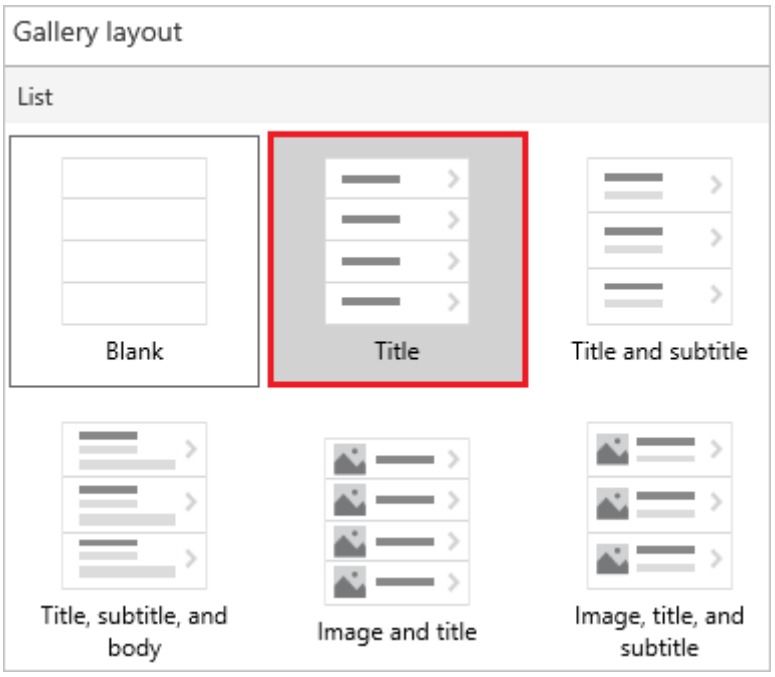
Change the layout

1. In the left navigation pane, select **BrowseGallery1**.

   When the gallery is selected, a selection box with handles surrounds it.



2. On the **Properties** tab of the right-hand pane, open the list of options under **Layout**, and then select the option that shows only a title.



3. Next to **Fields**, select **Edit**, and then select the down arrow for the title box.

   The name of this control will end in a numeral, such as **Title1**, but the numeral might differ based on other actions you might have taken.

4. In the list of options, select **Account name**, and then close the **Data** pane.

   The gallery shows the name of each account.

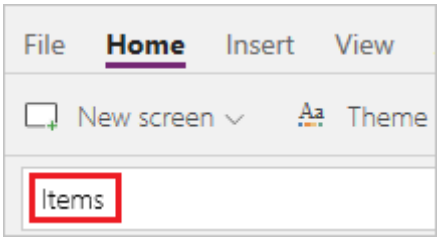


Change sort and search columns

1. Select the gallery as the previous section describes.



2. Near the upper-left corner, confirm that the property list shows **Items**.



   The value of this property appears in the formula bar. You set this property to specify not only the data source for the gallery but also the columns by which users can sort and search the data.

3. Copy this formula, and then paste it in the formula bar.

```
SortByColumns(Search(Accounts, TextSearchBox1.Text, "name"), "name",
If(SortDescending1, Descending, Ascending))
```

   By using this formula, you ensure that:

   - If a user types one or more characters in the search bar, the gallery shows only those account names that contain the text that the user typed.
   - If a user selects the sort icon, the gallery is sorted alphabetically by account name in either ascending or descending order, depending on how many times the user selects the icon.

   For more information about these and other functions, see the formula reference.

Test sorting and searching

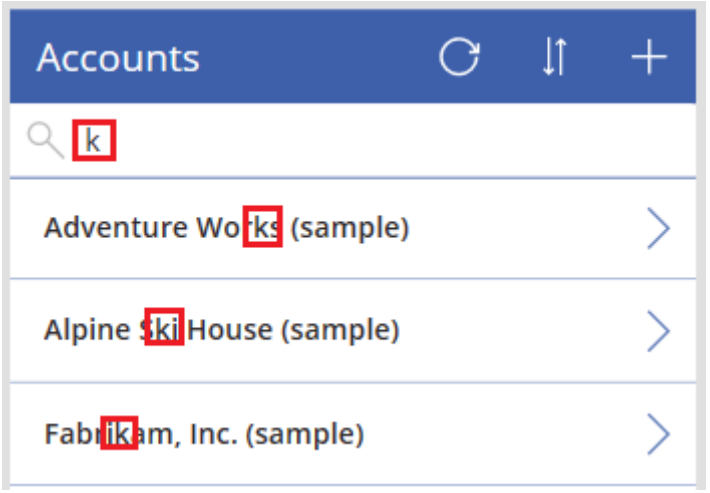
1. Open Preview mode by pressing F5 (or by selecting the play button near the upper-right corner).



2. Near the upper-right corner of the browse screen, select the sort icon one or more times to change the alphabetical sort order between ascending and descending.



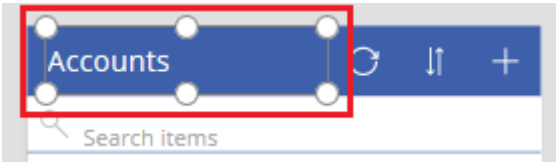
3. In the search box, type **k** to show only those account names that contain the letter that you typed.



4. Remove all text from the search bar, and then close Preview mode by pressing Esc (or by selecting the close icon near the upper-right corner).

Change the screen title

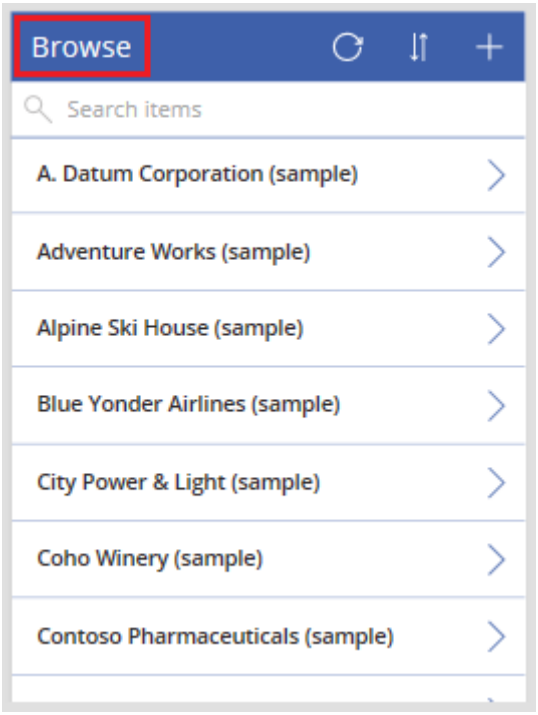
1. Select the title of the screen by clicking or tapping it.



2. Ensure that the property list shows **Text** and then, in the formula bar, replace **Accounts** with **Browse** (retaining the double quotation marks).



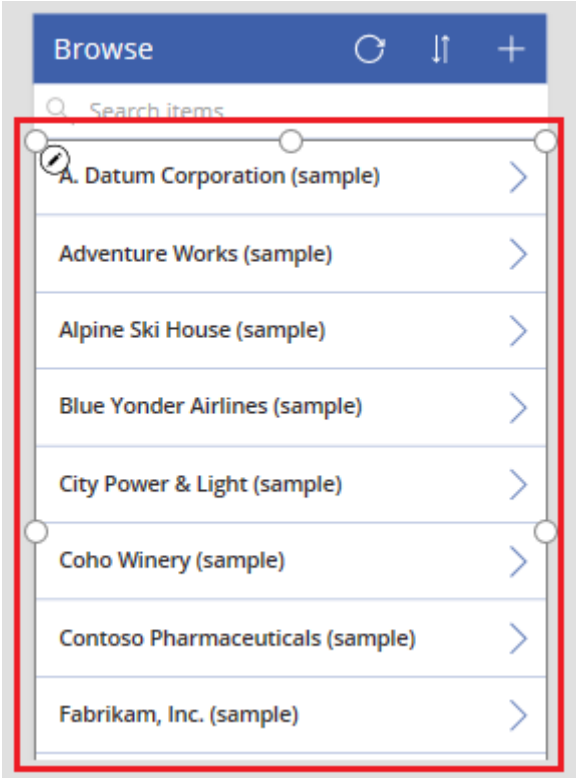
The screen reflects your change.



Show a scrollbar

If your users might have neither touch screens nor mouse wheels, configure the gallery to show a scrollbar when the user hovers over it with the mouse. That way, users can show all accounts even if the screen can't show them all at once.

1. Select the gallery as the first procedure describes.



2. Set the gallery's **Show scrollbar** property to **true**.

Next steps

In this tutorial, you've customized the gallery and made other changes to the default screen for browsing records in a generated app. You can also customize the default screens for displaying details and creating or updating accounts. As the browse screen contains a gallery, the other two screens in the app contain forms. You can change, for example, which types of data the forms show and in which order.

Feedback

Was this page helpful? Yes No

Provide product feedback

Customize forms in canvas apps

Article • 03/03/2023

In a canvas app, customize a **Display form** control and an **Edit form** control so that they show the data that matters most and in the most intuitive order to help users easily understand and update the data.

Each form comprises one or more cards, each of which shows data from a particular column in the data source. By following the steps in this topic, you can specify which cards appear in a form and move cards up and down within a form.

If you're unfamiliar with canvas-pps, see What are canvas apps?.

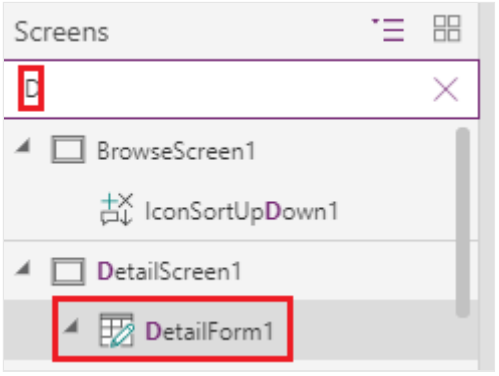
Prerequisites

Generate an app from the Microsoft Dataverse, and then customize the gallery in that app.

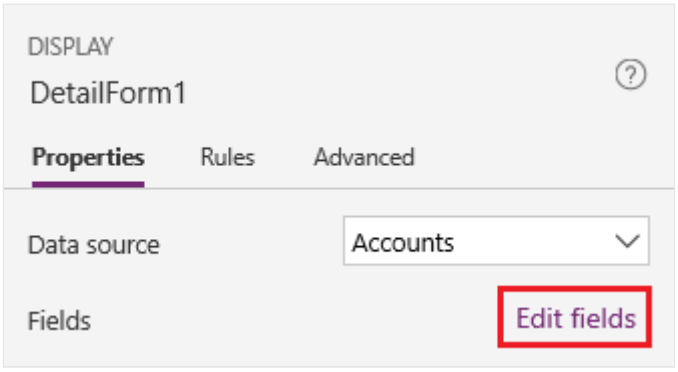
Show and hide cards

1. Sign in to Power Apps, and then open the app that you generated and customized.

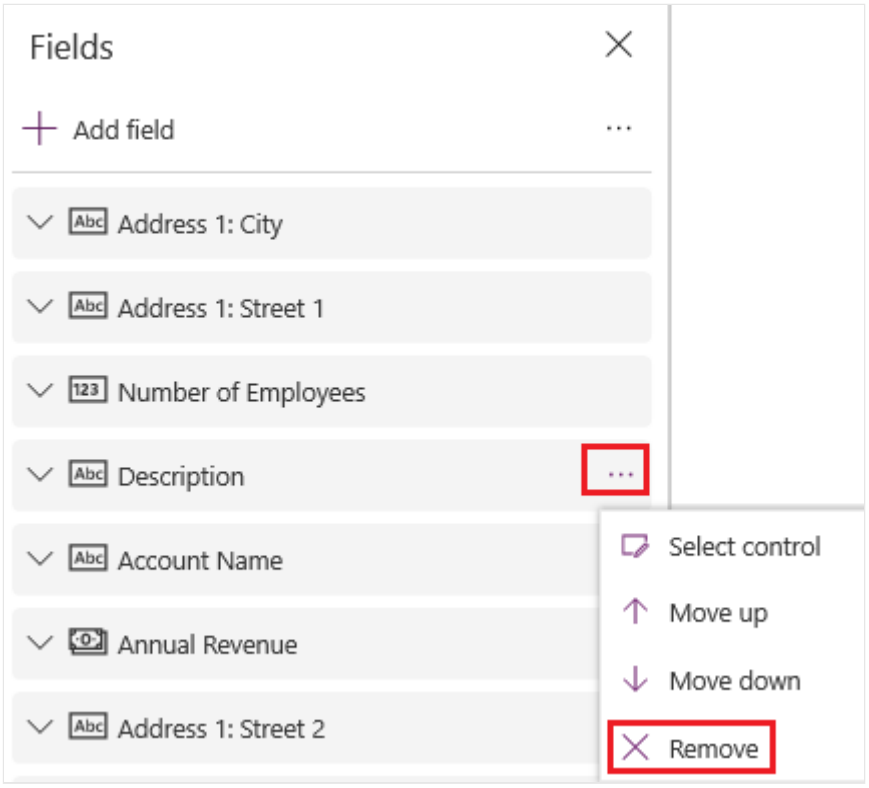
2. In the left navigation bar, type or paste **D** in the search bar to filter the list of elements, and then select **DetailForm1**.



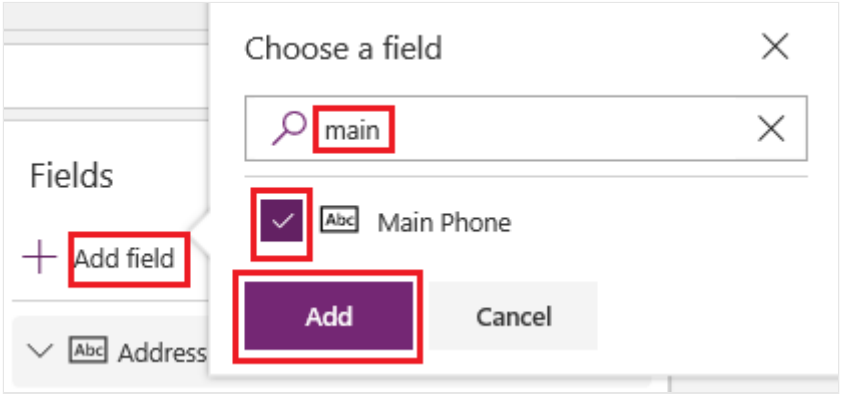
3. On the **Properties** tab of the right-hand pane, select **Edit fields** to open the **Fields** pane.



4. Hide a field, such as **Description**, by hovering over it, selecting the ellipsis (...) that appears, and then selecting **Remove**.



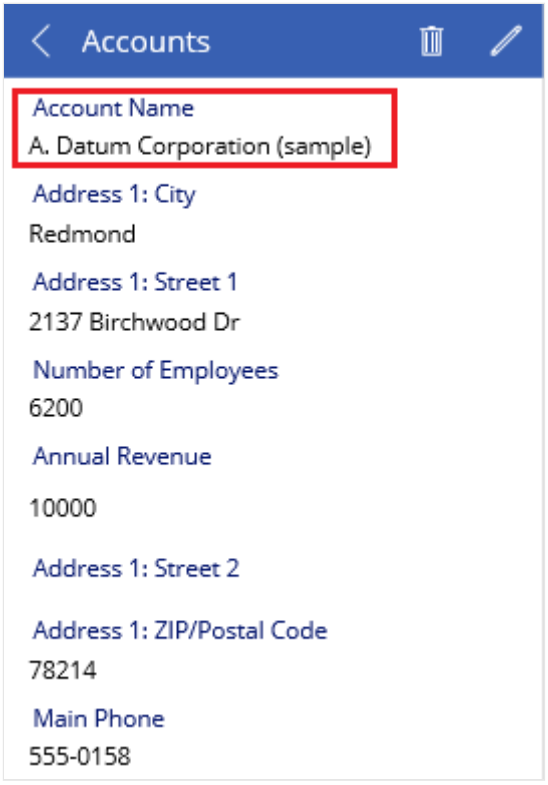
5. Show a field by selecting **Add field**, typing or pasting the first few letters of the field's name in the search box, selecting the field's check box, and then selecting **Add**.



Reorder the cards

1. In the **Fields** pane, drag the **Account Name** field to the top of the list of fields.

   The cards in **DetailForm1** reflect the change.



2. (optional) Reorder the other cards into this sequence:

   - Account Name
   - Number of Employees
   - Annual Revenue
   - Main Phone
   - Address 1: Street 1
   - Address 1: Street 2
   - Address 1: City
   - Address 1: ZIP/Postal Code

3. In the left navigation bar, type or paste **Ed** in the search bar, and then select **EditForm1** to select it.

4. Repeat the steps in the previous procedure and this one so that the fields in **EditForm1** match those in **DetailForm1**.

Run the app

1. In the left navigation bar, type or paste **Br** in the search bar, and then select **BrowseScreen1** to select it.

2. Open Preview mode by pressing F5 (or by selecting the **Preview** icon near the upper-right corner).



3. In the upper-right corner, select the plus icon to add a record in **EditScreen1**.



4. Add whatever data you want, and then select the checkmark icon in the upper-right corner to save your changes and return to **BrowseScreen1**.



5. Select the arrow for the item that you just created to show details about that item in **DetailScreen1**.



6. In the upper-right corner, select the edit icon to update the record in **EditScreen1**.



7. Change the information in one or more fields, and then select the check mark in the upper-right corner to save your changes and return to **DetailScreen1**.



8. Near the upper-right corner, select the trash-can icon to delete the record that you just updated and to return to **BrowseScreen1**.



9. Close Preview mode by pressing Esc (or by selecting the close icon near the upper-left corner).

Next steps

- Save and publish your app.

- Customize a card in your app.

Feedback

Was this page helpful? Yes No

Provide product feedback

Customize a card in a canvas app

Article • 03/03/2023

Perform basic customization (without unlocking a card) by, for example, changing its control. Perform advanced customization by unlocking the card and, for example, adding a control that isn't available for that card by default.

For an overview, see Understand data cards.

Prerequisites

- Learn how to add and configure controls.
- You can review this topic for general concepts only, or you can follow it step by step if you first complete the procedures in these topics:

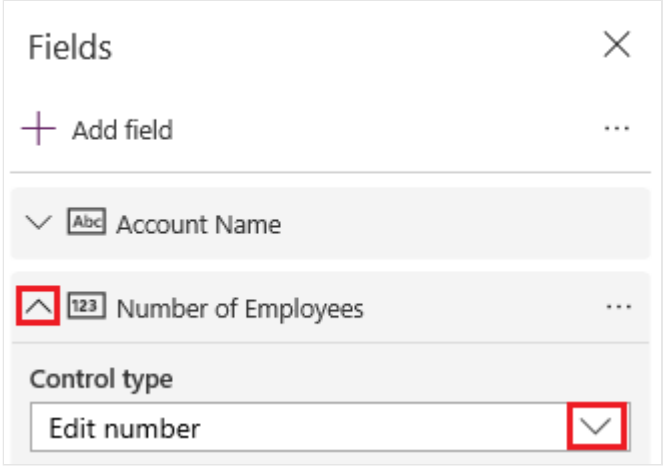
    1. Generate an app.
    2. Customize its gallery.
    3. Customize its forms.

Customize a locked card

In this procedure, you'll replace a **Text-input** control with a **Slider** control without unlocking the card.

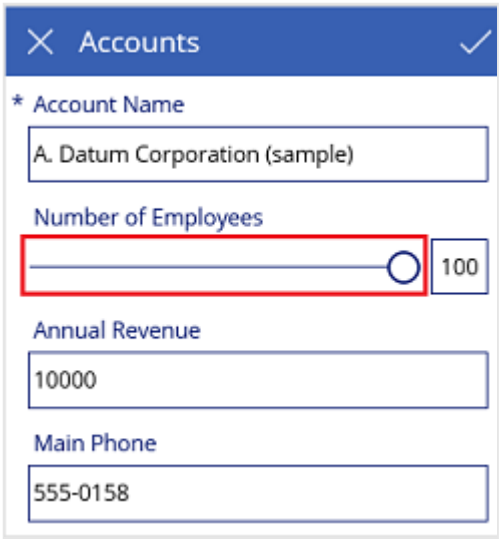
1. In the app that you generated and customized, select **EditForm1** in the left navigation bar, and then select **Edit fields** on the **Properties** tab of the right-hand pane.

2. In the list of fields, select the down arrow for **Number of Employees**, and then open the list under **Control type**.



3. Select **Edit Slider**.

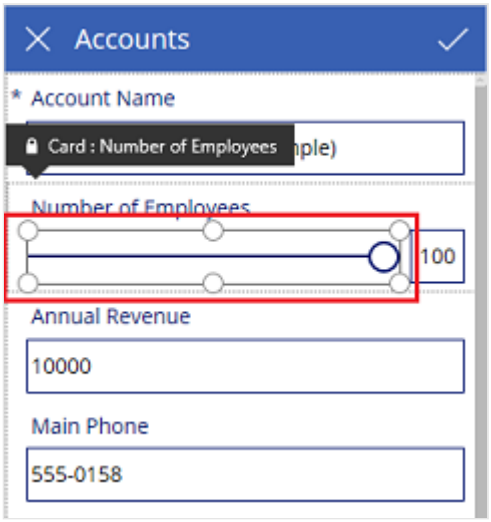
   The screen reflects your change.



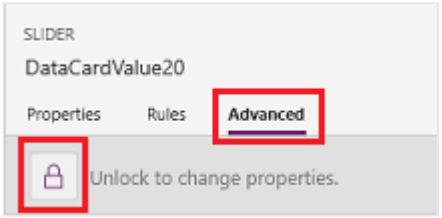
Unlock and customize a card

In this procedure, you'll unlock a card and update the **Max** property of the **Slider** control that you just added.

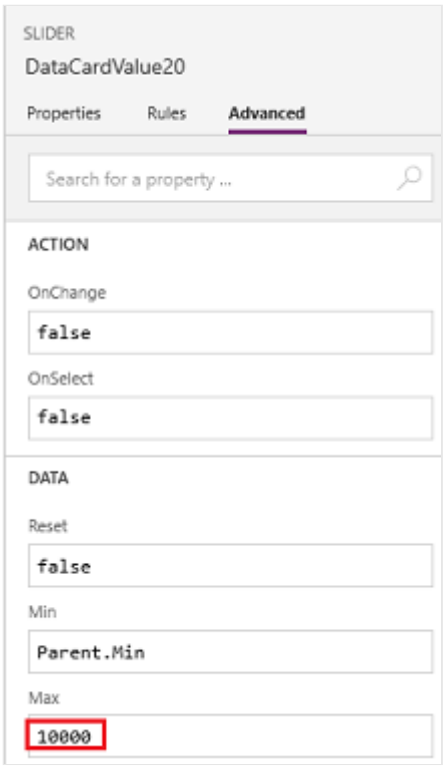
1. In **EditForm1**, select the **Slider** control in the **Number of Employees** card.



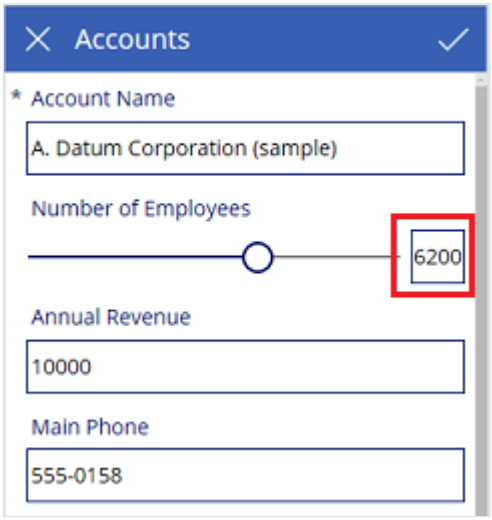
2. On the **Advanced** tab of the right-hand pane, select the lock icon to unlock the card.



3. Set the **Max** property of the **Slider** control to 10,000.



The **Slider** control shows a more accurate value.



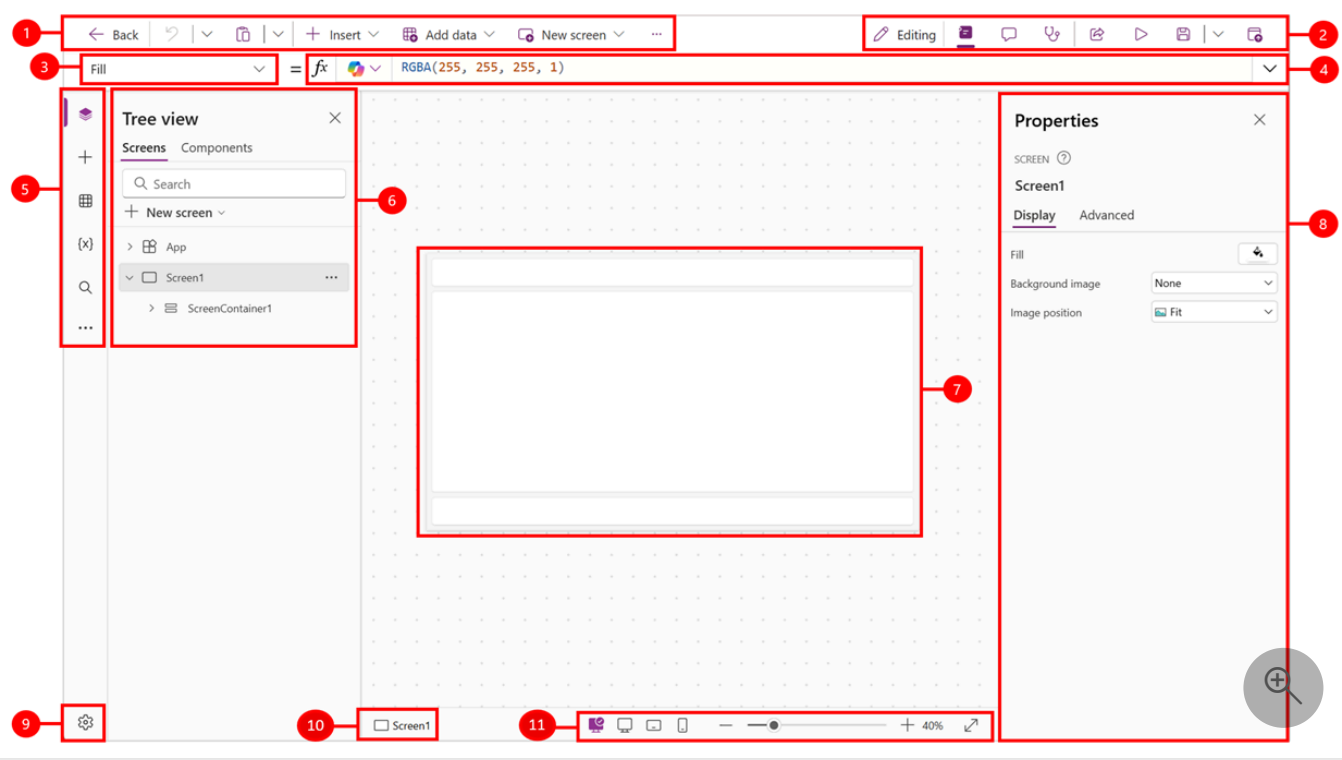
Next steps

Now that you have a basic understanding of how to generate an app and customize a gallery, a form, and a card, you can build your own app from scratch.

Understand Power Apps Studio

Power Apps Studio is the integrated development environment where you design, build, and manage canvas apps. Learn about its key components including the command bar, formula bar, canvas, and authoring tools to create effective business applications.

Overview of Power Apps Studio



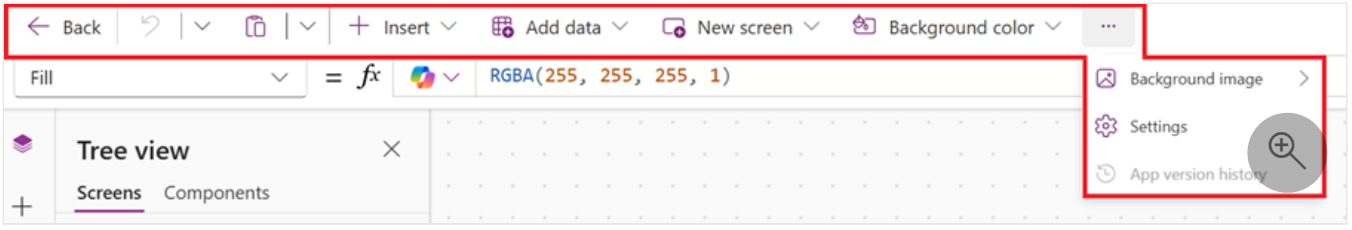
1. Power Apps Studio modern command bar: Dynamic command bar that shows a different set of commands based on the control selected.
2. App actions: Options to view properties, add comments, run the app checker, share, preview, save, or publish the app.
3. Properties list: List of properties for the selected object, such as *fill* or *height*.
4. Formula bar: Compose or edit a formula for the selected property with one or more functions.
5. App authoring menu: Selection pane to switch between data sources and allows you to insert options.
6. App authoring options: This pane reveals the corresponding authoring feature when a feature is selected from the app authoring menu.
7. Canvas/screen: Primary canvas for composing the app structure.

8. Properties pane: Properties list for the selected object in Tree view.
9. Settings: Configure your app in settings.
10. Screen selector: Switch between different screens in your app. Tree view selections display here.
11. Change canvas screen size: Change the display size of the canvas as you author your app.

Let's understand each option in Power Apps Studio in detail.

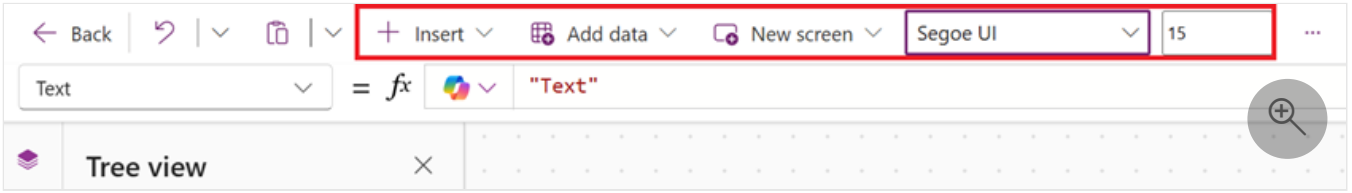
1 – Power Apps Studio modern command bar

The command bar provides Power Apps Studio options. These options are relevant to the current session and app-related settings.



Modern command bar

The modern command bar displays the relevant set of commands depending on the control that you select. For example, if you select an item to *insert* like a text label, you see the modern command bar change to accommodate your selection. The bar now has font type, size, color, and other text label controls.



The command bar changes when you select one of the following controls or objects:

1. App object
2. Screen
3. Button
4. Shape
5. Icon
6. Blank form
7. Form with data
8. Blank gallery

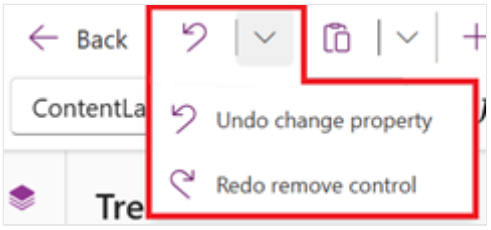
9. Gallery with data
10. Label
11. Text input
12. Date picker

When you select more than one type of control, the command bar shows the common commands between the selected controls.

Back

Takes you back to the build hub, closing the current Power Apps Studio session.

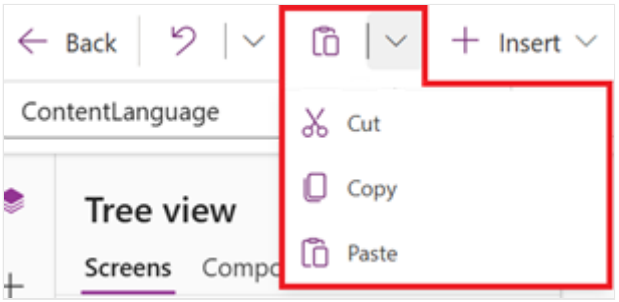
Undo and redo



- **Undo**: Undo the last action.
- **Redo**: Repeat the last action.

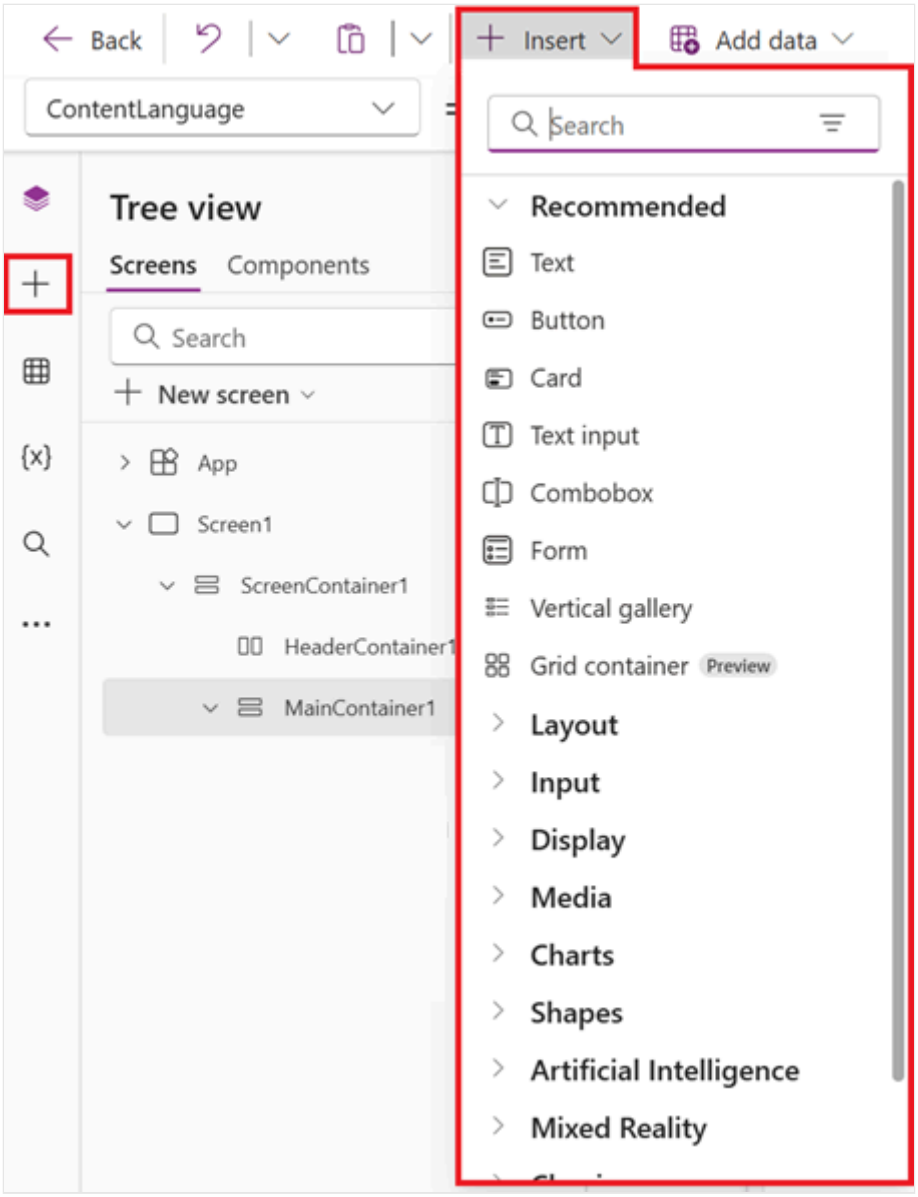
> **Note**
>
> You can't undo or redo data operations, such as insert or delete a data source.

Cut, copy, and paste



- **Cut**: Cut the selection, and store it in the clipboard.
- **Copy**: Copy the selection, and store it in the clipboard.
- **Paste**: Paste the last cut or copied selection from the clipboard.

Insert



You can access the **Insert** menu from the command bar and from the app authoring menu.

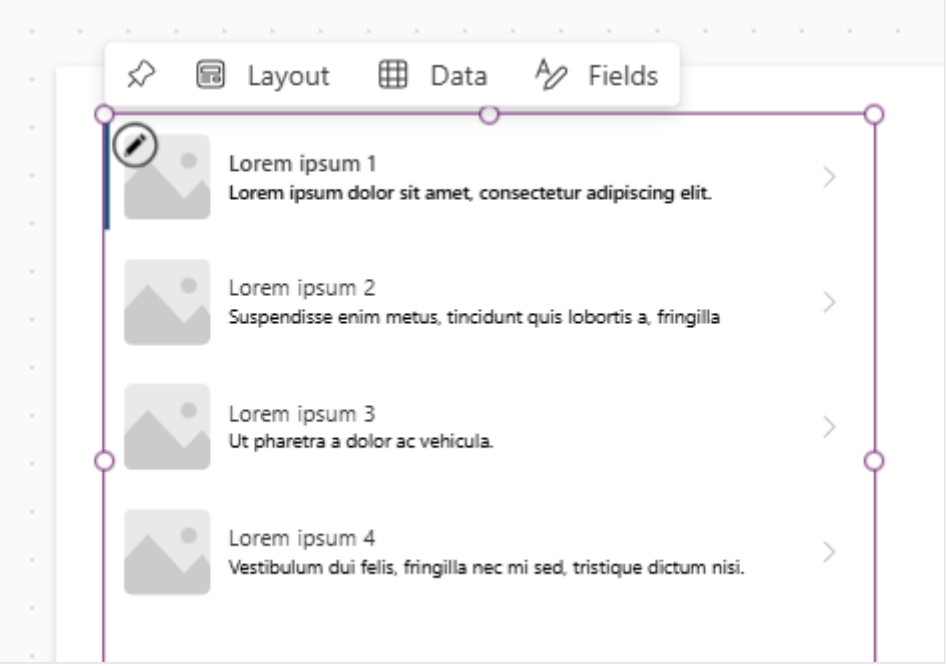
Insert shows all the popular objects or controls that you can add on the selected screen in your canvas app. You can also expand other choices or use the components option to insert controls from a component library.

To insert controls on the canvas, drag the control to the canvas or select the control.

Popular controls to insert

Expand table

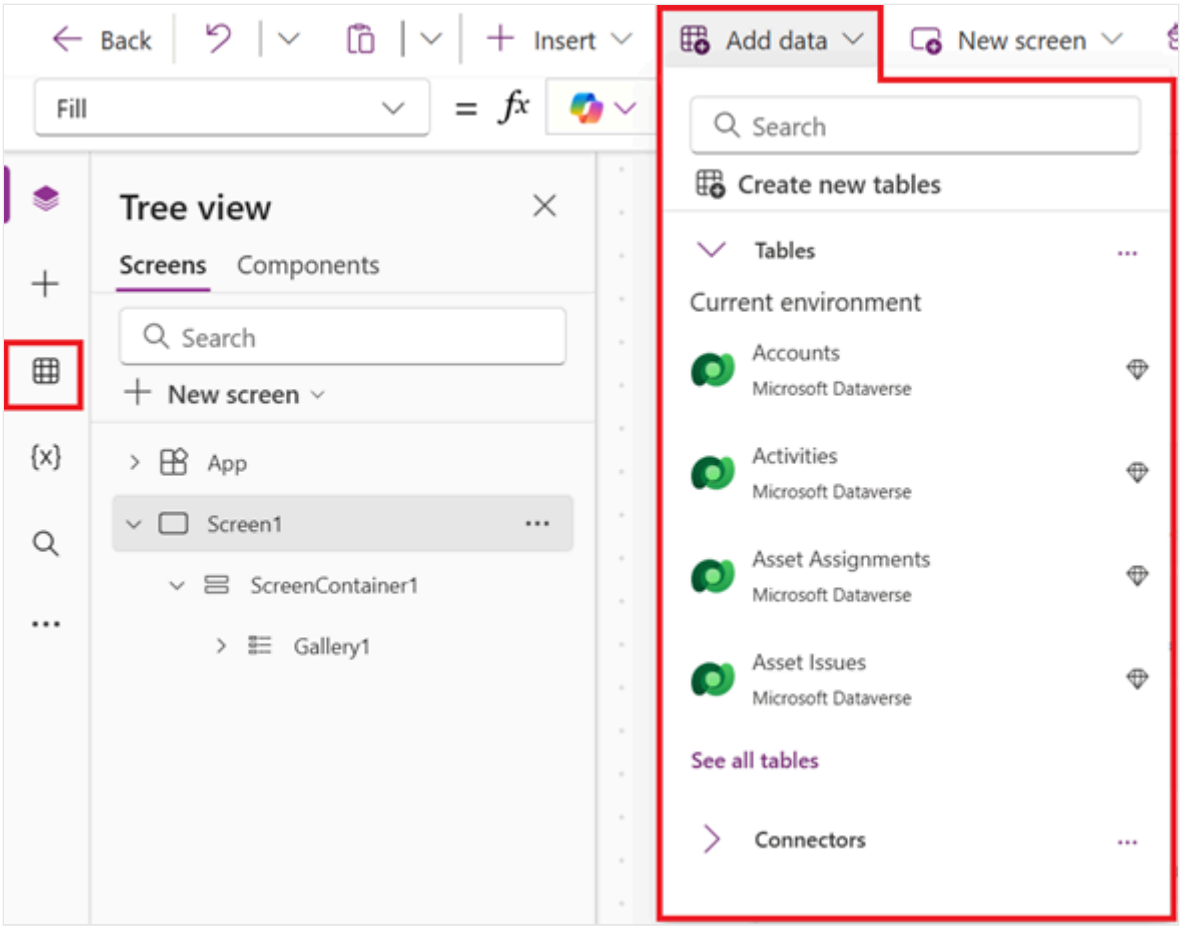
Name	Description
Form	Display, edit, or create a record in a data source.
Input > *Button*	A control that the user can select to interact with the app.

Name	Description
Input > Date picker	A control that the user can select to specify a date.
Input > Text input	A text box that allows user input.
Shapes > Rectangle	A rectangular shape to configure the canvas appearance.
Text	A box that displays data such as text, numbers, dates, or currency.
Vertical gallery	A control that you can add other controls to and display a set of data. For example, list items made up of an image, title, and description is contained within a vertical gallery. 

For more information about the controls you can insert, including their properties and definitions, see Controls and properties in canvas apps.

Add data

Add, refresh, or remove data sources from your canvas app. You can add one or more connections by using data sources.



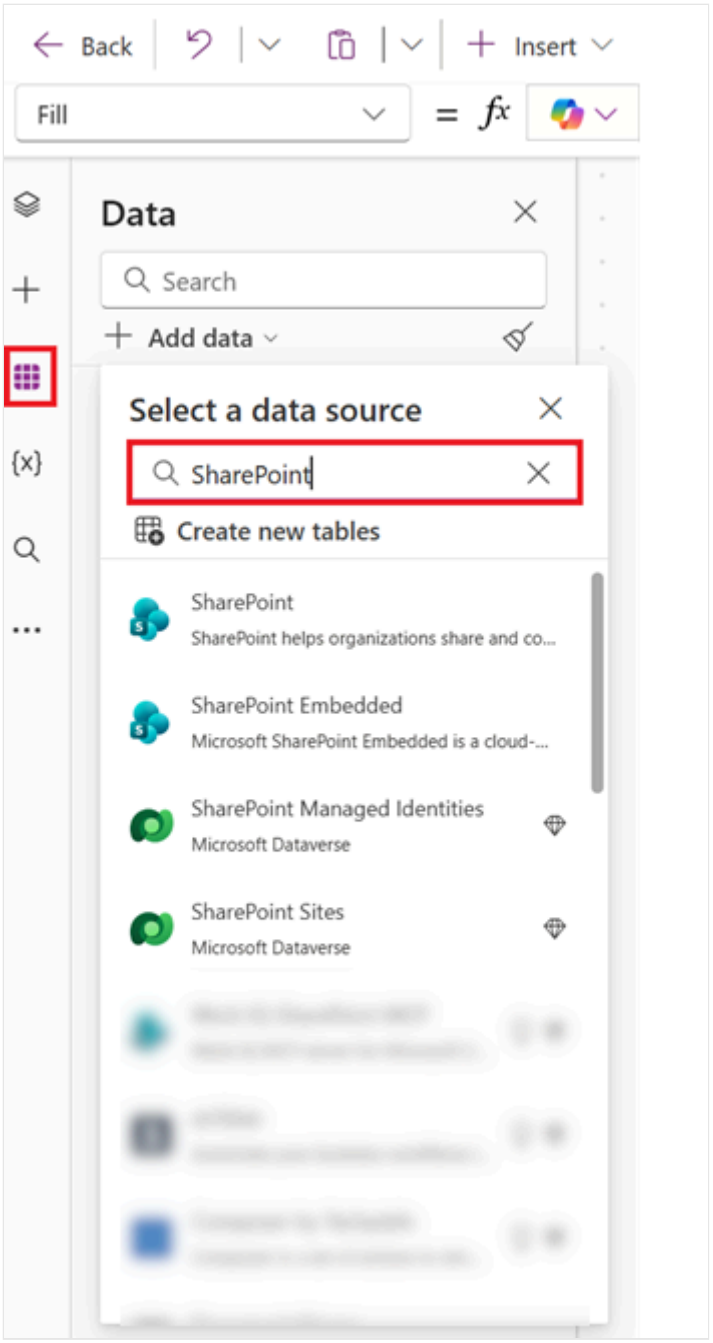
From the **Add data** menu, you can:

- Select any other existing tables from the current environment.
- Search for and select a connector, such as **SharePoint** or **SQL Server**.

Connectors

In addition to data stored within tables, many connectors are available to interact with data in popular software as a service (SaaS), services, and systems.

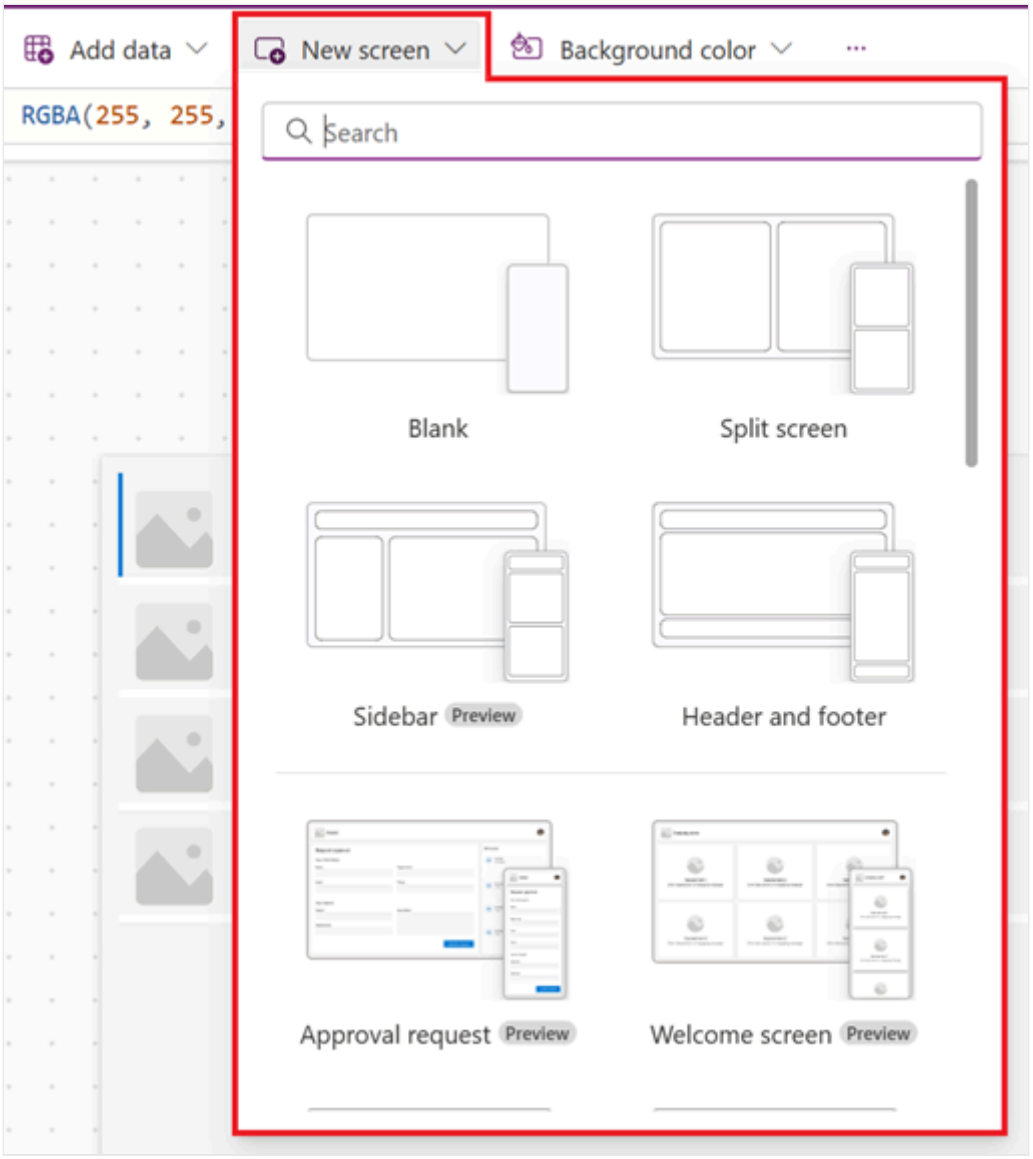
To choose other connectors such as SharePoint, OneDrive, or SQL Server, enter text in the data source search box or select from the list of connectors.



For more information, see List of all Power Apps connectors.

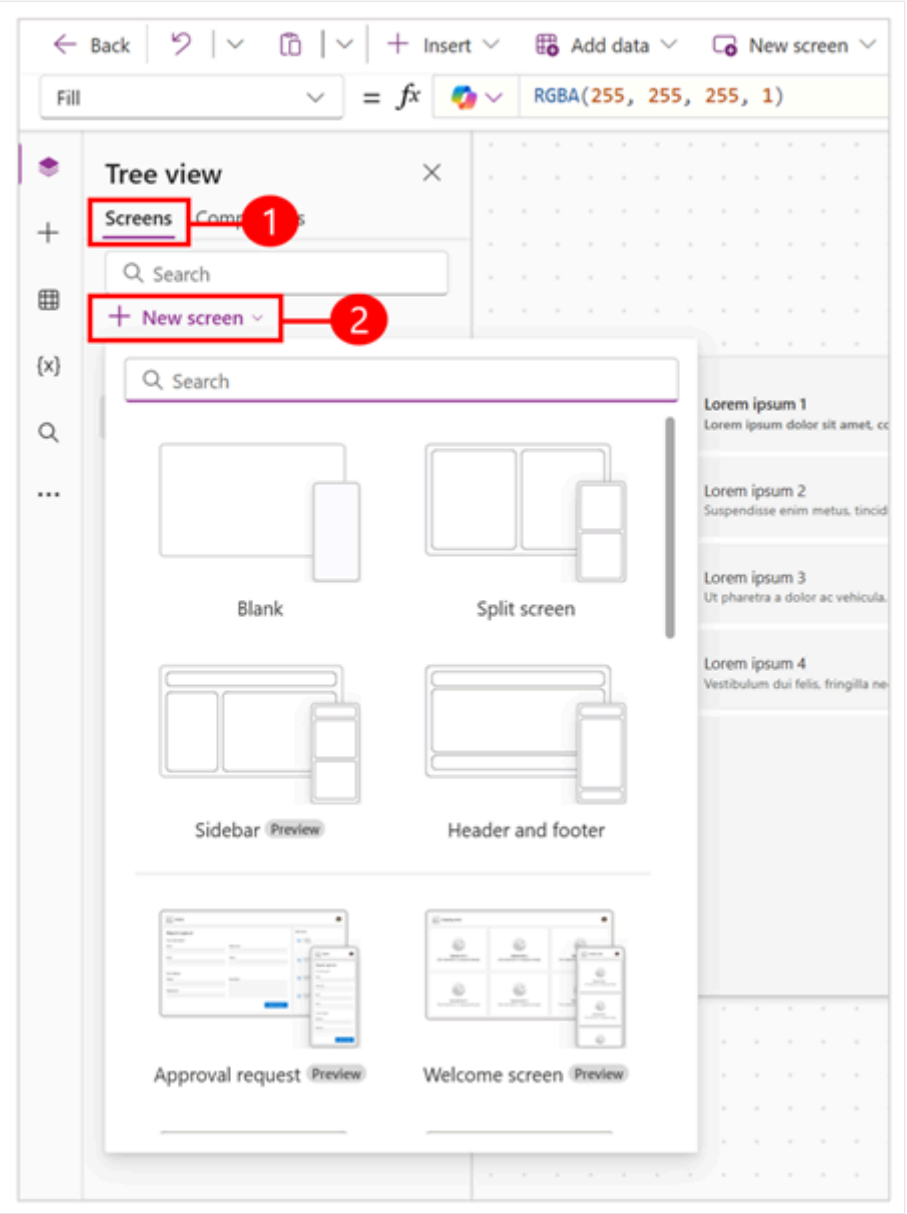
New screen layouts

Add a new screen to your app from the **New screen** option in the command bar.



You can also add a new screen from the tree view.

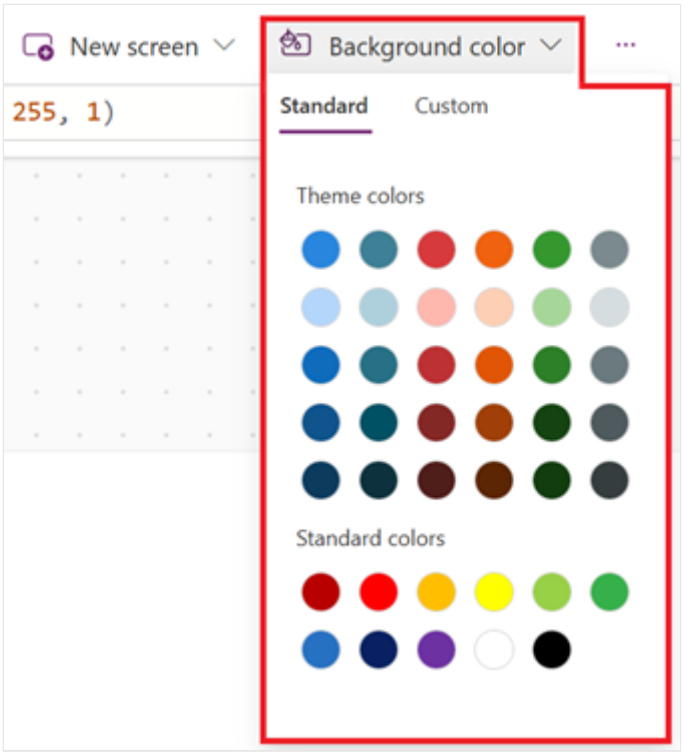
1. From the **Screens** tab, select **+ New screen**.



2. Select from the list of available screen layouts such as **Sidebar**, new layouts in preview such as **Header and form**, or templates like **Email**.

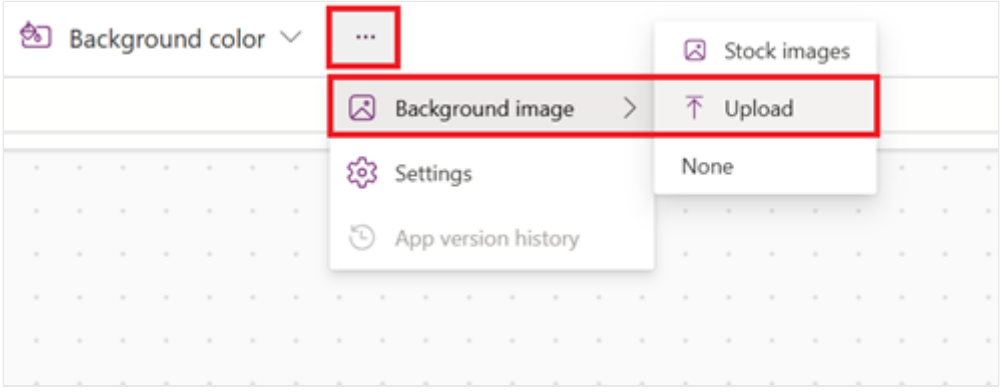
Background color

Select a background color for a screen. You can select from the list of standard colors or select the **Custom** tab and choose your own color.



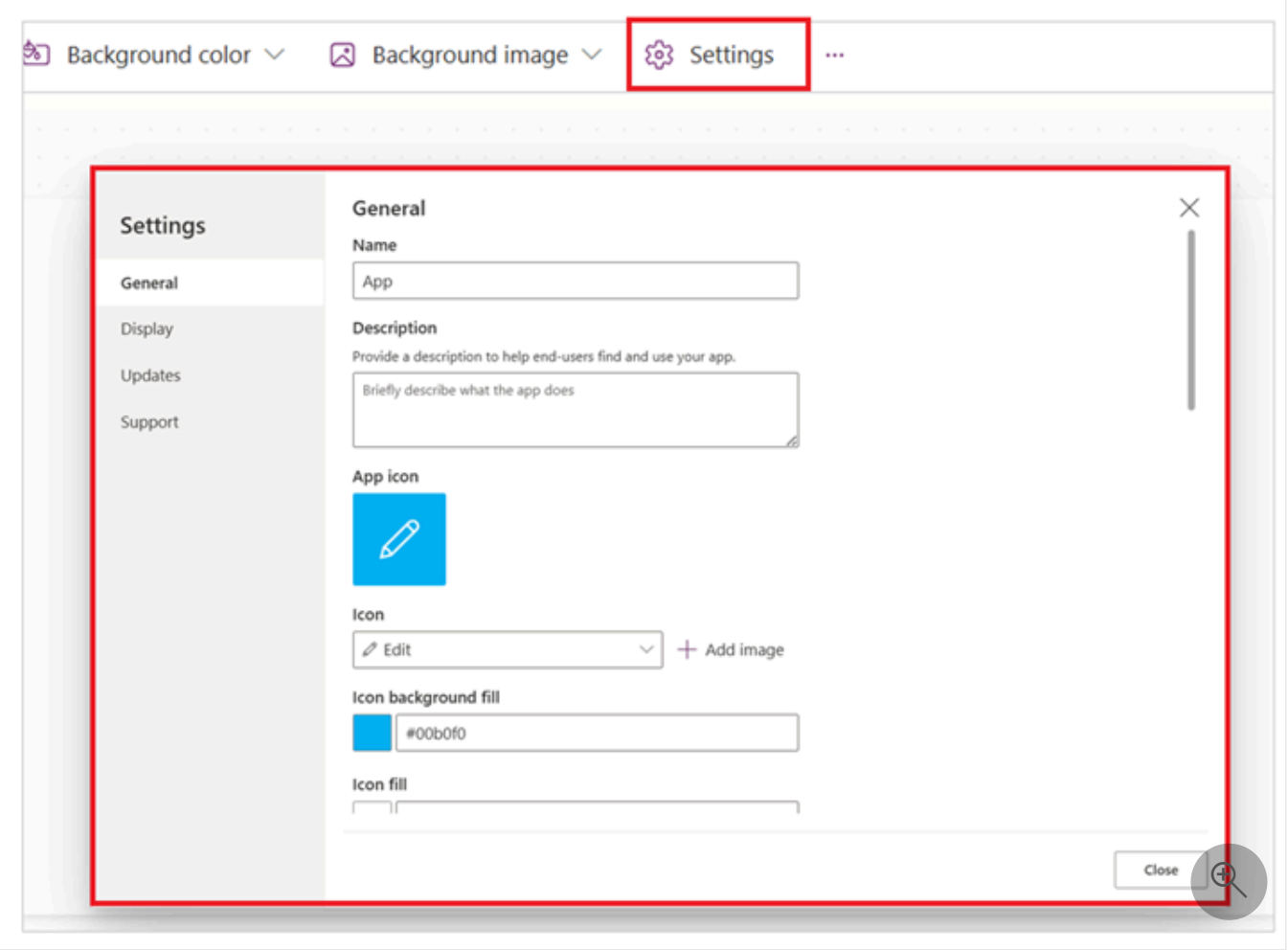
Background image

Select **Upload** to upload images to set as the background image.



Settings

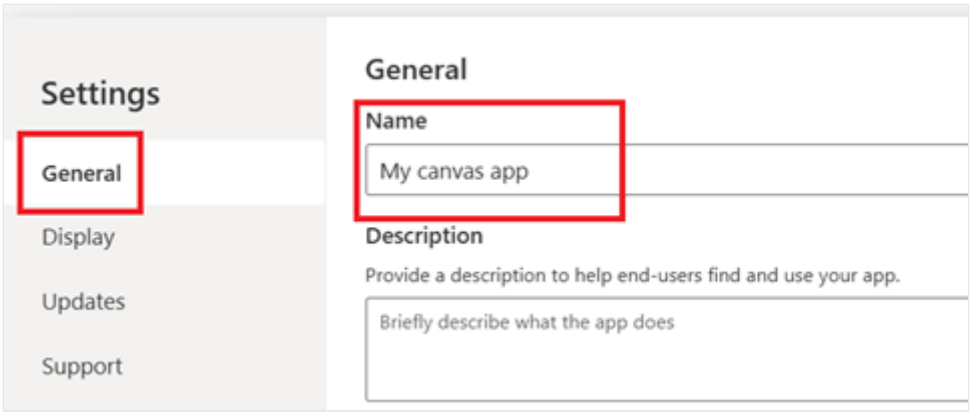
Configure the app's settings from the General, Display, Updates, or Support tab.



General

- Edit the app **Name** and **Description**.
- Add or update the **App icon**. Add a custom icon by selecting **+ Add image**.
- Select an **Icon background fill** or **Icon fill** color.
- Toggle **Auto save** to save every two minutes automatically.
- Configure offline use.
- Set your **Data row limit**.
- Include debug information when you publish.
- Enable autocreation of environment variables.
- Enable the `App.OnStart` property.

To edit the app **Name**, go to the **General** tab in **Settings**.



Display

- Select the **Orientation** and screen **Size**.
- **Scale to fit**: Scales the app to fit available space.
- **Lock aspect ratio**: Locks the height and width ratio.
- **Lock orientation**: Maintains app orientation when device rotates.
- **Show mobile device notifications area**: Displays notifications at the top of the screen.

For more information, see Change screen size and orientation.

Updates

Configure advanced settings for the app that include features under preview, experimental, or retired features.

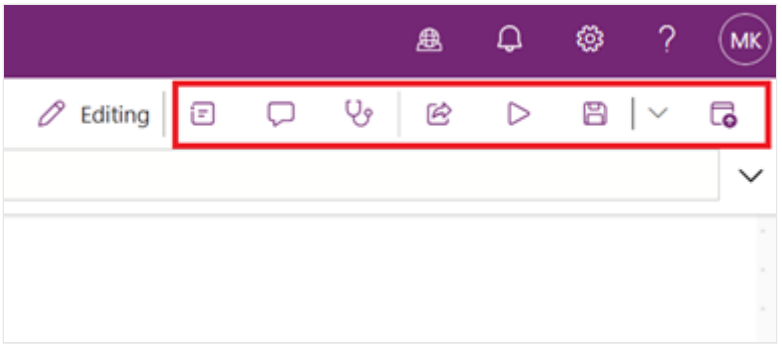
For more information, see Understand experimental, preview, and retired features in Power Apps.

Support

Access current Power Apps Studio information such as environment, authoring version, session ID, and session details. This information is useful for Microsoft Support sessions.

2 – App actions

To perform app-specific actions, use the options such as **Properties**, **Comments**, **App checker**, **Share**, **Preview the app**, **Save**, and **Publish**.



Share

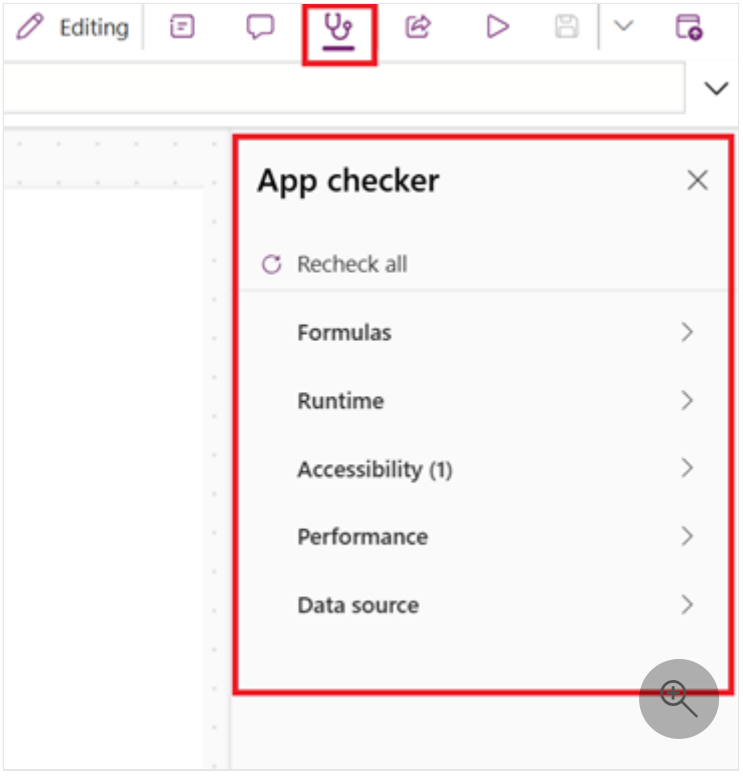
When you select the **Share** app action, a new window opens where you can share the app. You can share with other users or add them as co-owners of your app. For more information, see Share an app.

> **Tip**
>
> You must save the app before you can share it.

App checker

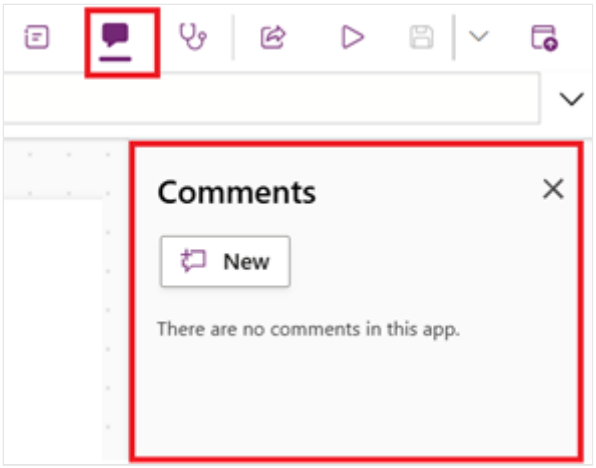
Select **App checker** to run a check.

For more information, see PowerApps checker now includes App checker results for canvas apps in solutions.



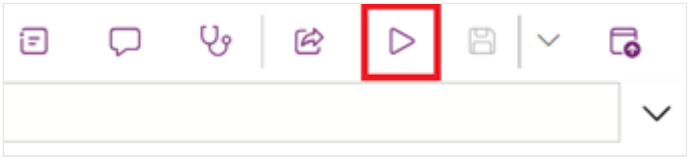
Comments

Comments are notes associated with items in your app. Use comments to help your team review the app and provide feedback, or provide additional information on implementation details in your app.



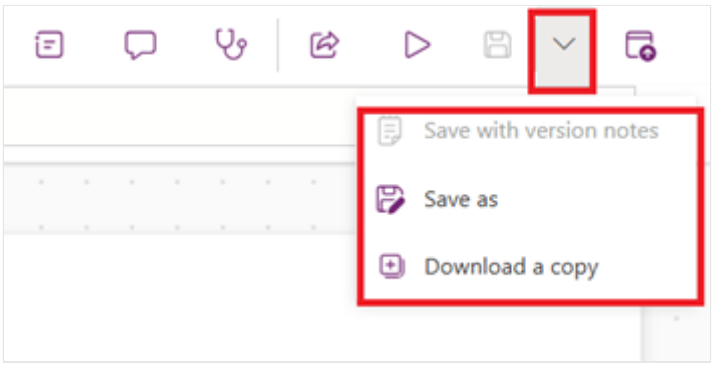
Preview

Select **Preview the app** to go into preview mode. Here you can view and interact with the current version of the app.



Save

You can save your app in different ways, such as **Save**, **Save with version notes**, **Save as**, or **Download a copy**. The save options dropdown is located next to the **Save** app action.



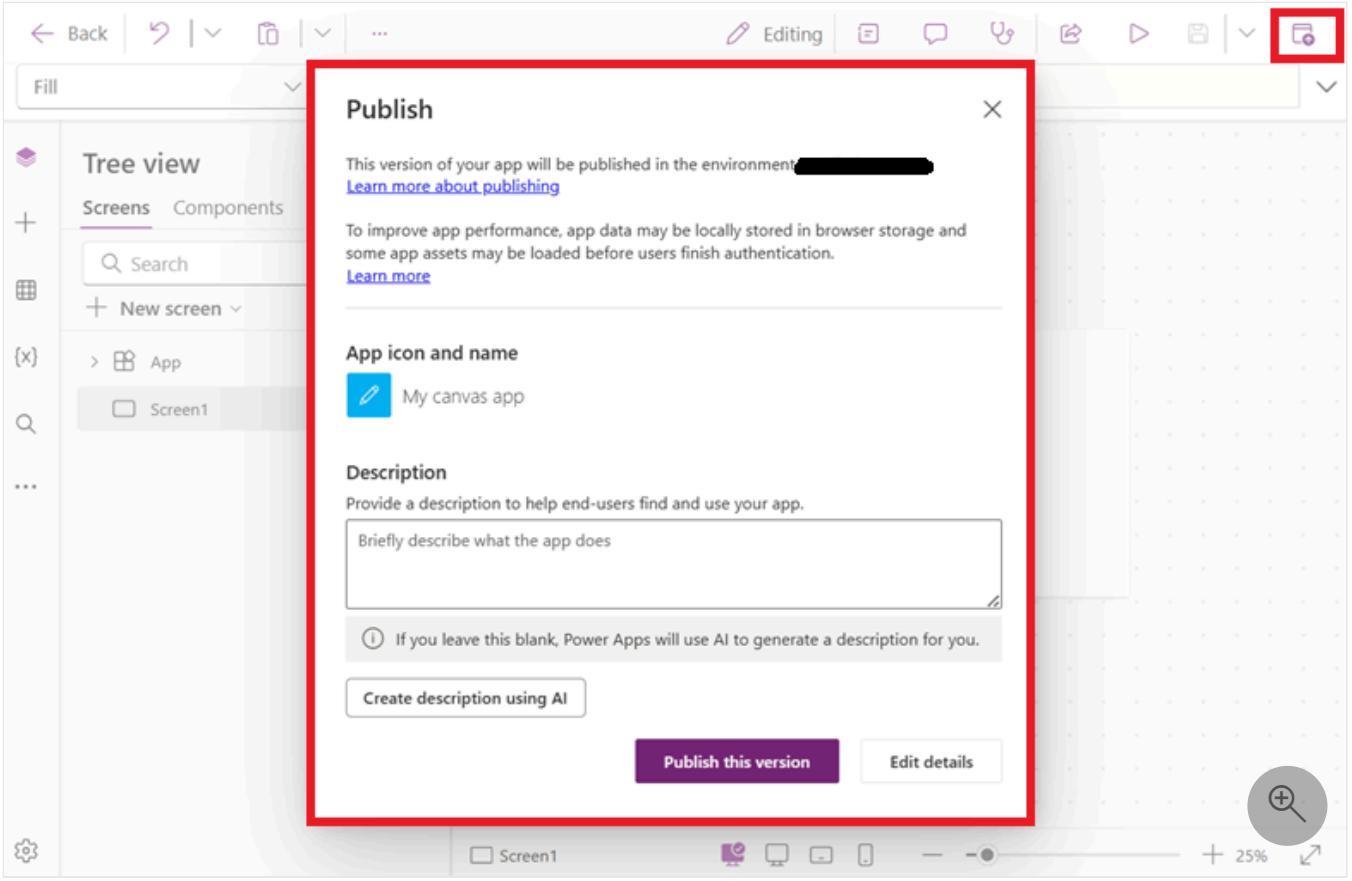
Save options include:

- **Save**: Saves recent and unsaved changes you made to the app. Each time you save changes, a new version is created.

- **Save with version notes**: Save and add notes about your updates.
- **Save as**: Duplicate the app by saving it with a different name.
- **Download a copy**: Download a local copy of the app.

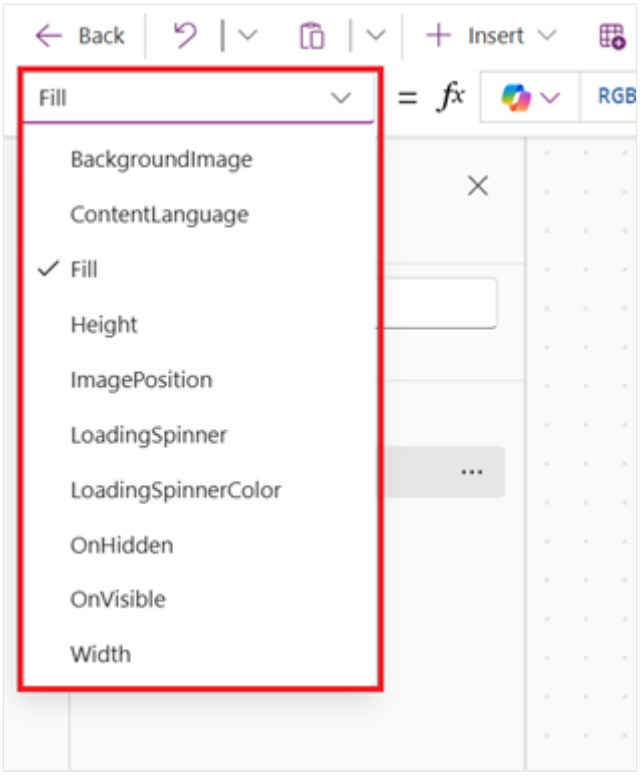
Publish

Select **Publish** to publish the app's current version. For more information, see Save and publish canvas apps.



3 – Properties list

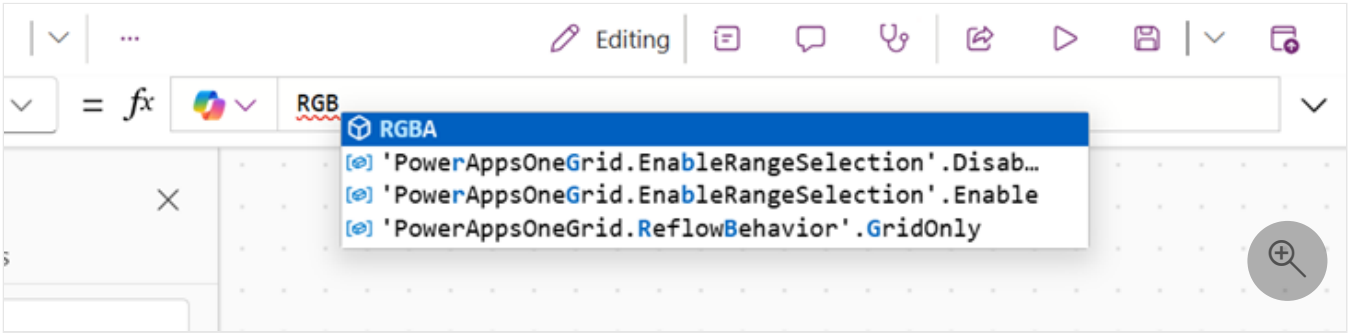
When you select an object in your canvas, you can choose one of the object's properties from its properties list. For a complete list of all possible properties, see All properties.



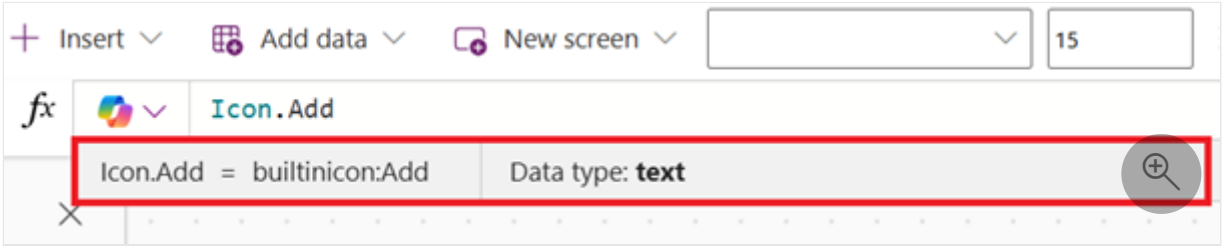
4 – Formula bar

The formula bar lets you add, edit, or remove functions of a property from your selected object. For example, select the app screen to update the background color by using the RGBA function.

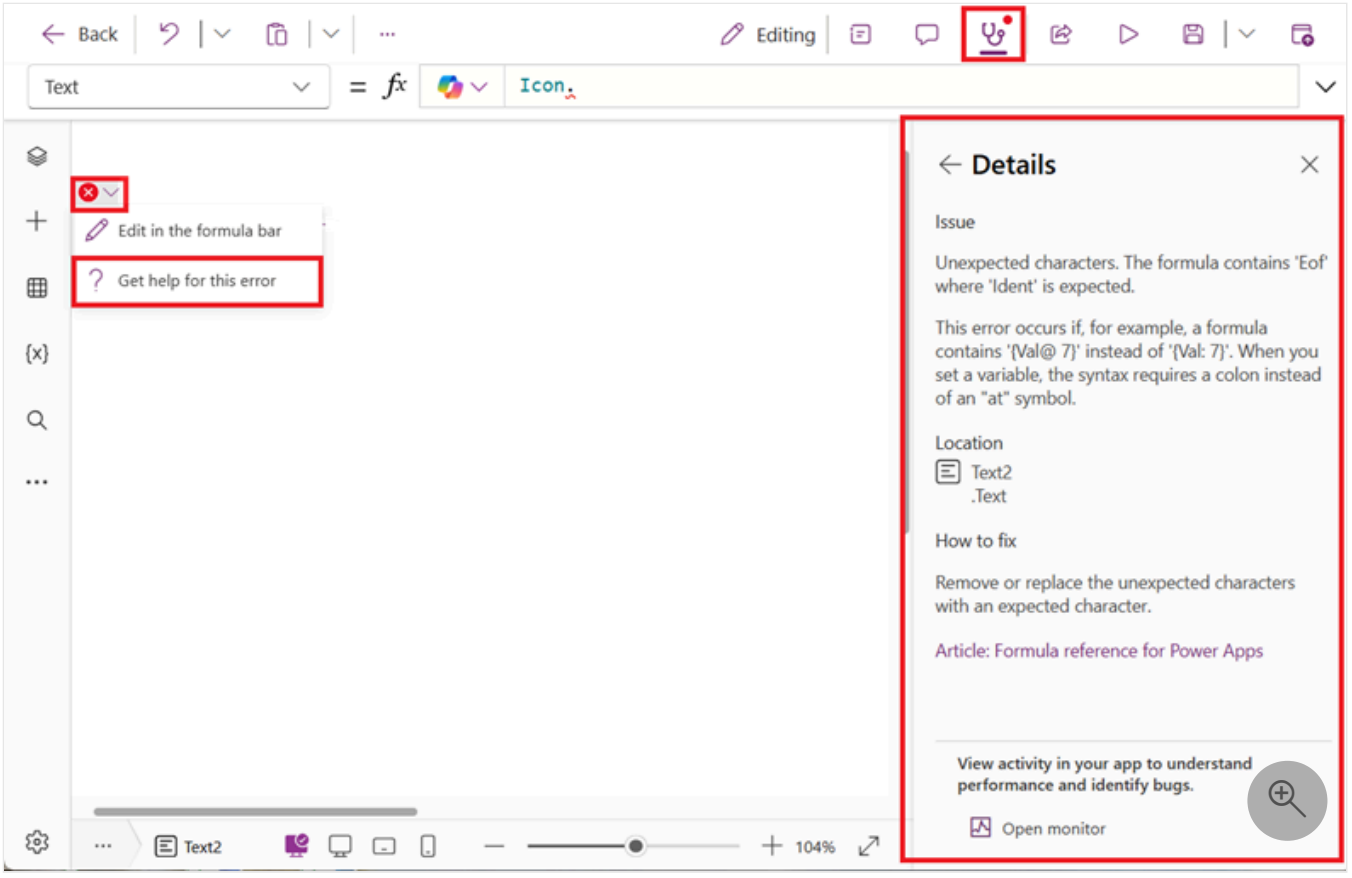
The formula bar is IntelliSense-enabled. When you start entering text that matches one or more functions, the formula bar shows a list of functions.



When you select a function, the formula bar shows inline function help and highlights help text relevant to the cursor position.



If a formula returns an error, you can view the error details in **App checker** or directly from the app view in the canvas.

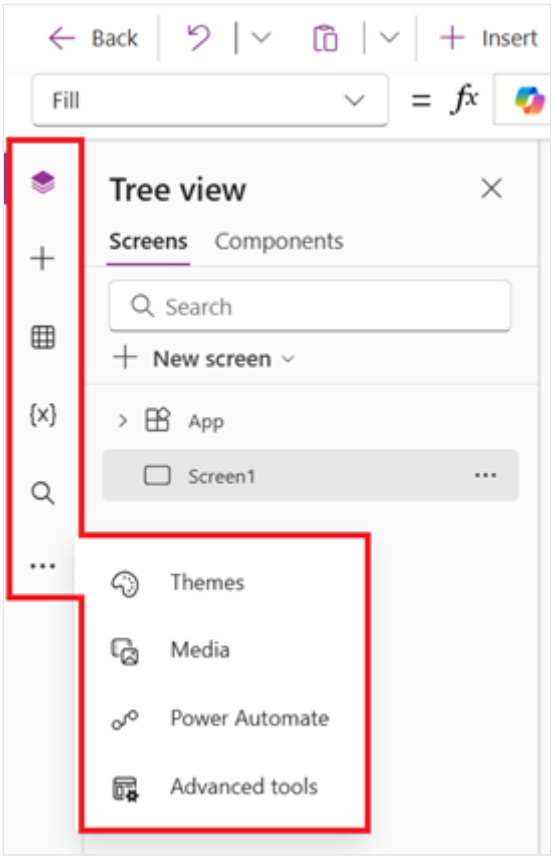


Similarly, you find help when working with complex functions, nested functions, or when correcting a formula syntax.

For a complete list of all canvas app functions, see Formula reference - Power Apps.

5 – App authoring menu

Switch between various authoring options while working with the app.



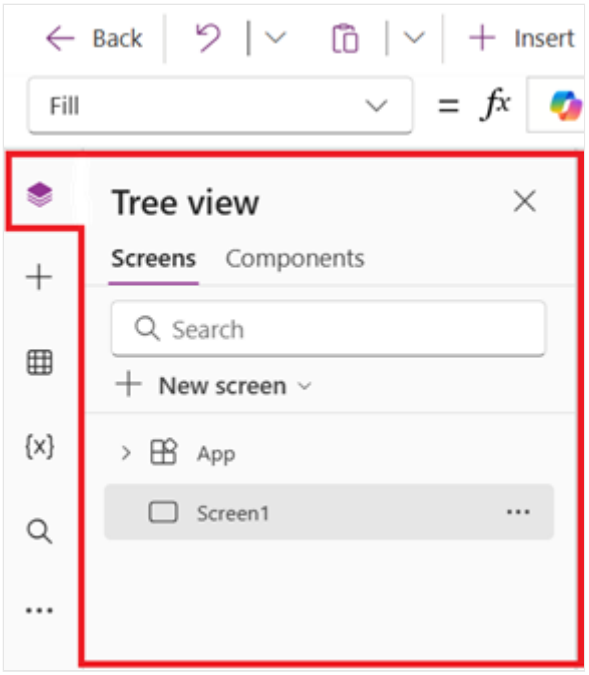
- **Tree view**: Shows a tree view of all screens and controls in the current app.
- Insert: Add different controls to the screen.
- **Data**: Add data and other connectors, such as tables that connect to your app.
- **Variables**: Work with variables and collections while editing your app.
- **Search**: Select to search for media, formulas, text, and more in your app.
- **Themes**: Choose a colored theme for the Power Apps UI.
- **Media**: Insert or remove media from the app.
- **Power Automate**: Add a flow by using the Power Automate pane.
- **Advanced tools**: Access the Monitor and Test tools to debug and test your app.

The options for working with canvas apps change depending on your selection from the authoring menu. For example, when you select the **Tree view** menu option, you see a **Tree view** pane appear.

6 – App authoring options

Tree view

Select the tree view to show the screens available in the app.



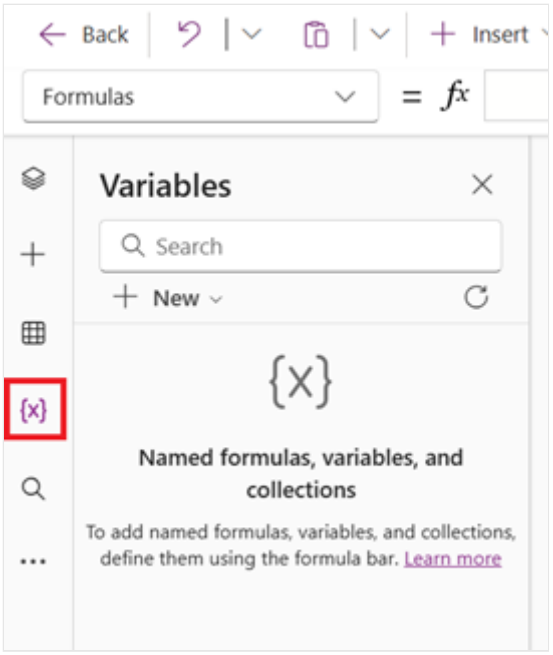
> **Tip**
>
> Select **App** in the tree view to work with app-specific controls. You can change app behavior, such as adding a formula for the *OnStart* event of the app.

Switch to the **Components** tab of **Tree view** to work with component library features. You can add new components or reuse ones from the published component libraries. For more information, see Component library.

For more information on adding components by selecting **+ New component**, see the Insert section.

Variables

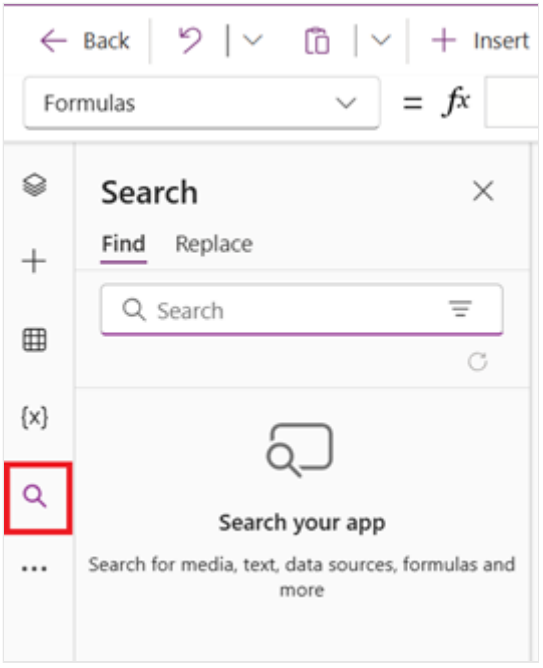
You can save data such as resulting values from a data set into temporary storage by using variables. This section lists variables used by the current app. For more information, see Variables in canvas apps.



A collection is a group of items that are similar, such as products in a product list. For more information, see Collections in canvas apps.

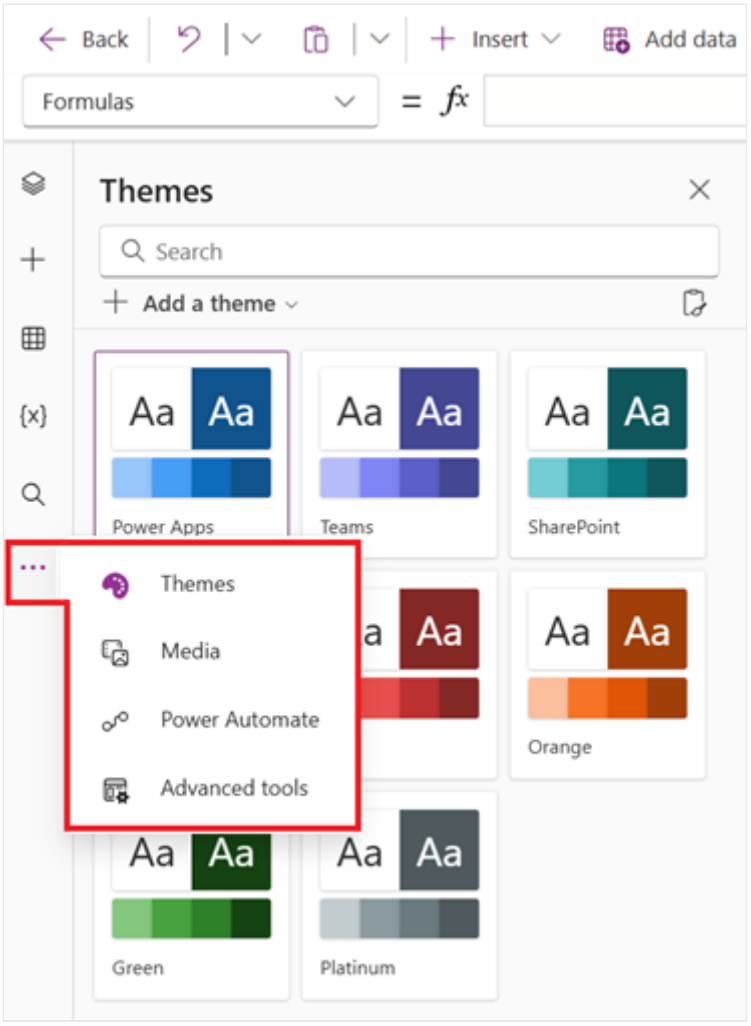
Search

You can find components in your app or find and replace them.



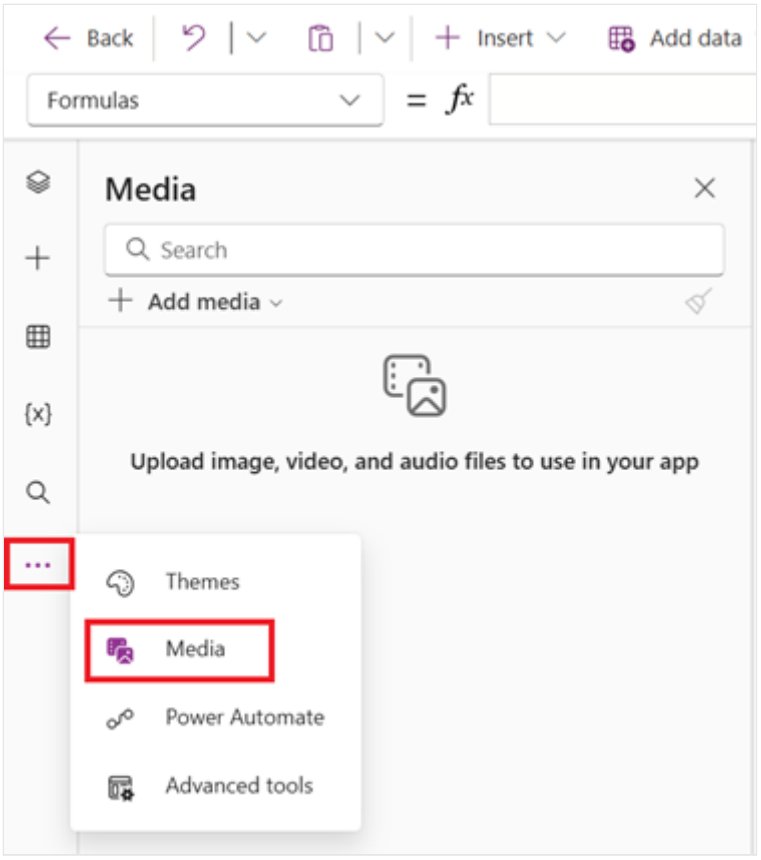
Themes

You can choose a premade colored theme or custom-make your own colored theme by selecting + **Add a theme**.



Media

Select **Media** to add images, video, or audio files to your app. Adding media directly to your app uploads the files to the app and uses the app storage.



Tip

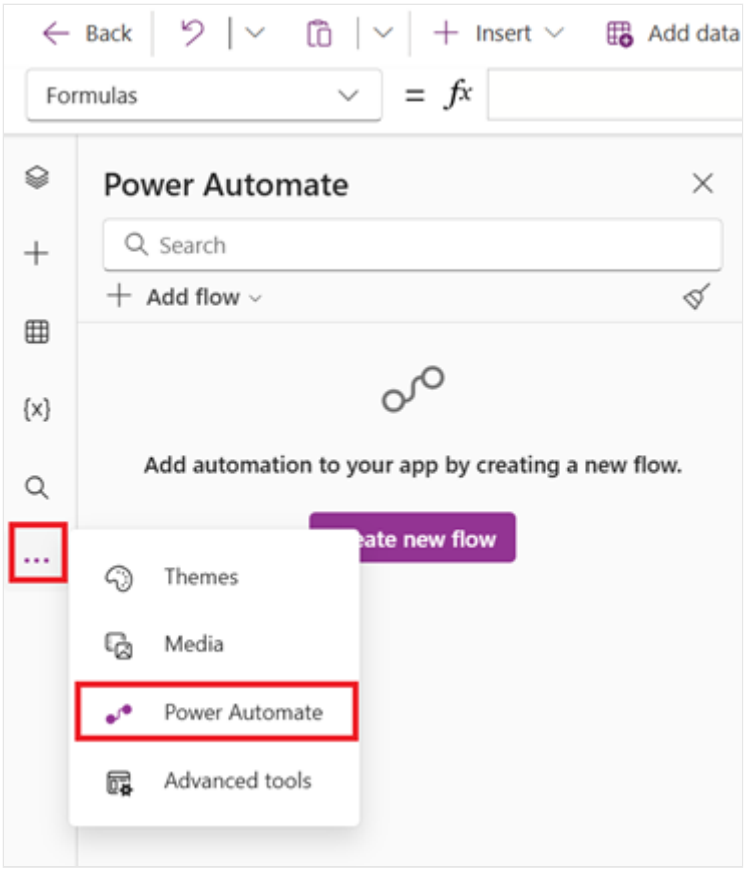
- Each file uploaded to the app as media must be 64 MB or smaller.
- The size of all media files uploaded to an app can't exceed 200 MB.

If you want to reference more media, consider using audio and video controls with URLs, media from Azure Media Services.

For more information, see Using multimedia files in Power Apps.

Power Automate

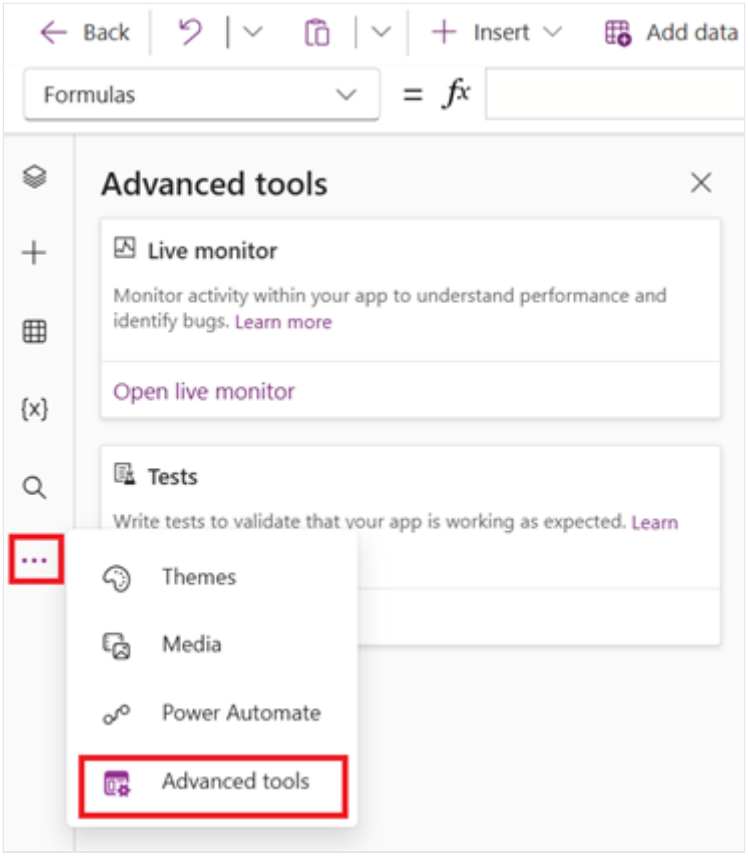
Create a new flow with Power Automate, or select a flow you added previously.



For more information, see Use Power Automate pane.

Advanced tools

Advanced tools include **Monitor** and **Tests**, where you can track and test your app.



7 – Canvas/screen

The canvas shows the currently selected screen in the **Tree view** from the authoring menu.

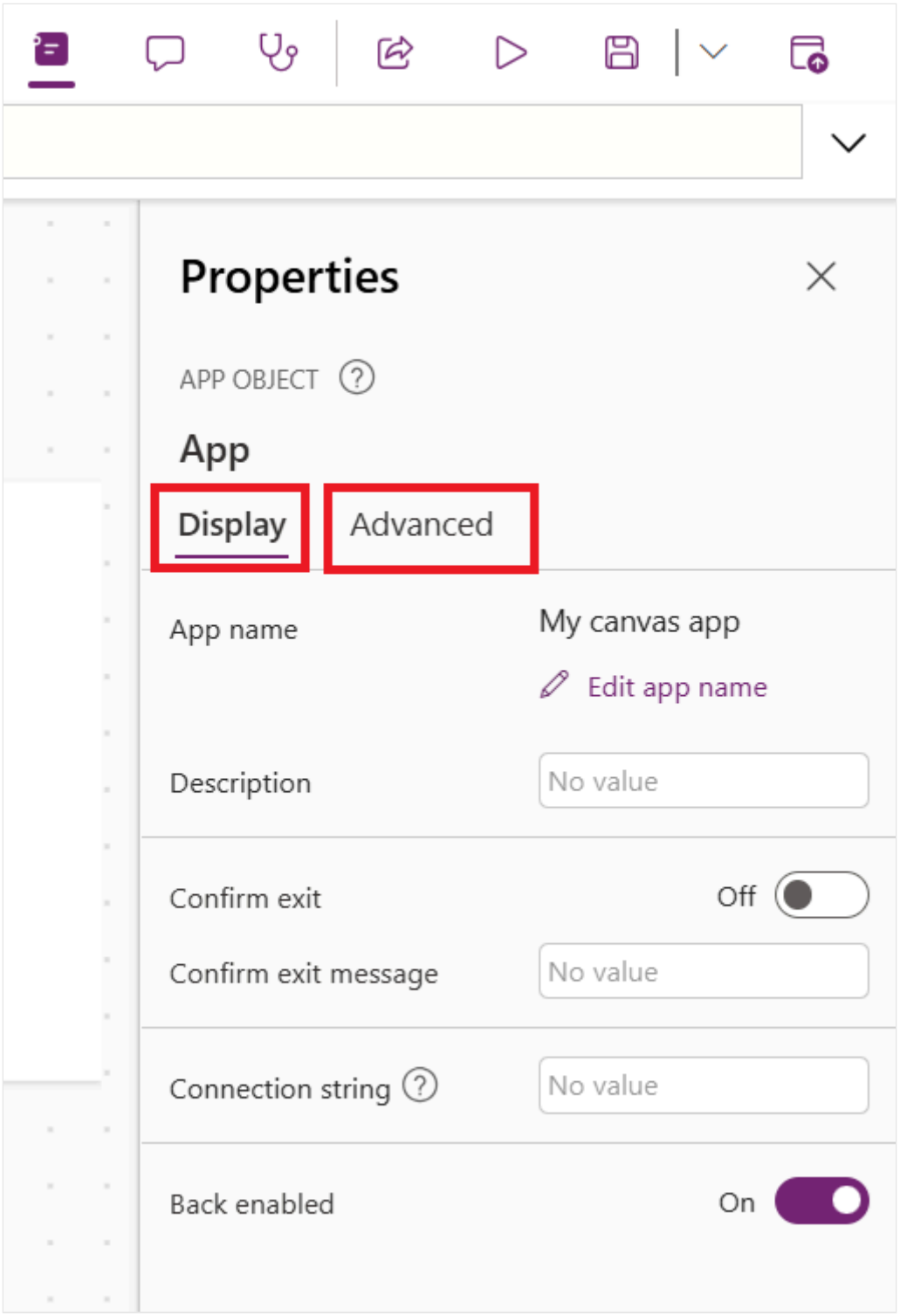
When you edit a canvas app, you can Use inline actions in Power Apps Studio so you don't have to leave your current view. Inline actions make the development process more efficient.

8 – Properties pane

The properties pane shows properties and options for the currently selected object on the canvas.

- The **Display** tab shows generic options such as the name, description, exit or exit message, and more.
- The **Advanced** tab shows options for customization with key value pairs in areas of action, data, and design in your app.

  > **Tip**
  >
  > The advanced properties might be locked for editing, such as when working with data cards. Select **Unlock to change properties** in such situations.



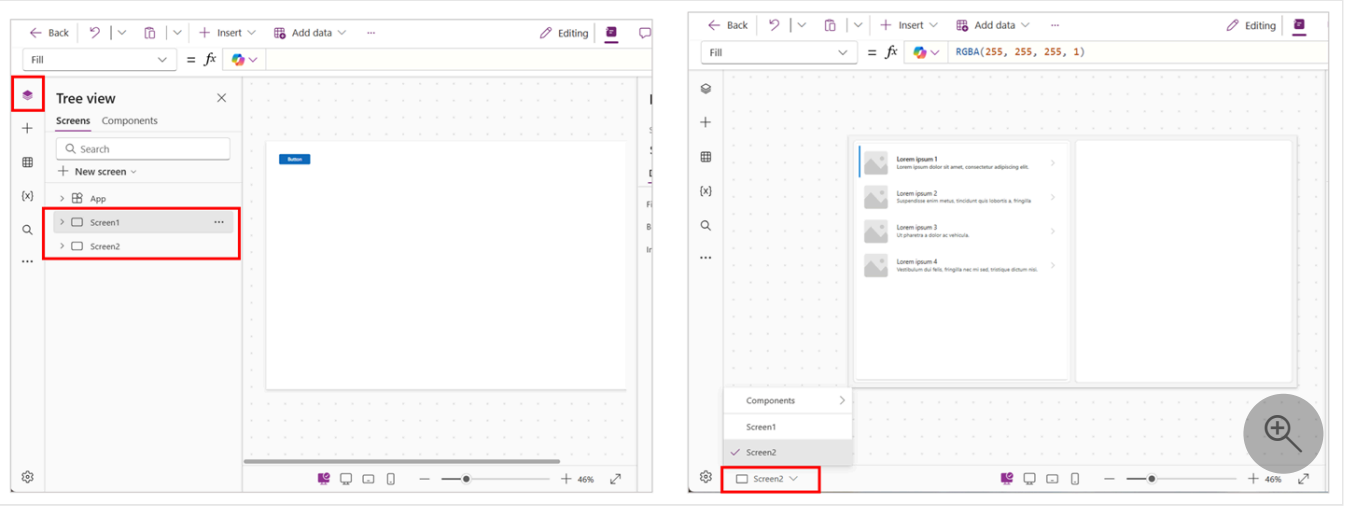
9 – Settings

Access Settings at the bottom of the app authoring menu or from the command bar as previously described.

10 – Screen selector

Use the screen selector to switch between screens when your canvas app has multiple screens. You can also select a screen from the authoring menu by selecting the tree view. If the current

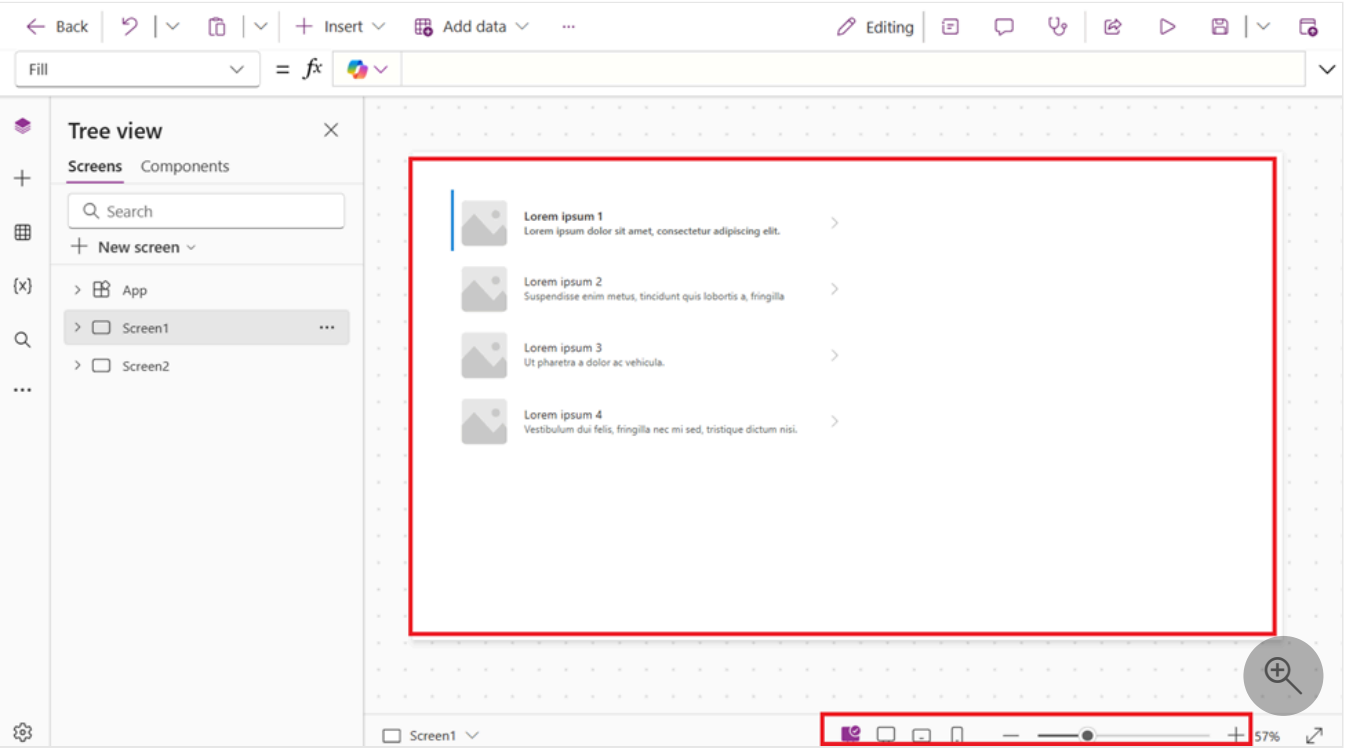
selection is inside a container, or inside an individual cell in a gallery, the selector shows the breadcrumbs for the parent elements at each level.



11 – Change canvas screen size

You can zoom in or out while authoring the canvas app. Select **Ctrl**+**0** (zero) to *fit to window*, which fits the canvas to the current authoring window size.

The following image shows what the canvas looks like when *fit to window*. You can also use the controls to zoom further in or out to view your canvas.



> **Note**
>
> The zoom percentage or screen size you use when authoring a canvas app doesn't affect the aspect ratio you configure for the app. When you preview your app or play a

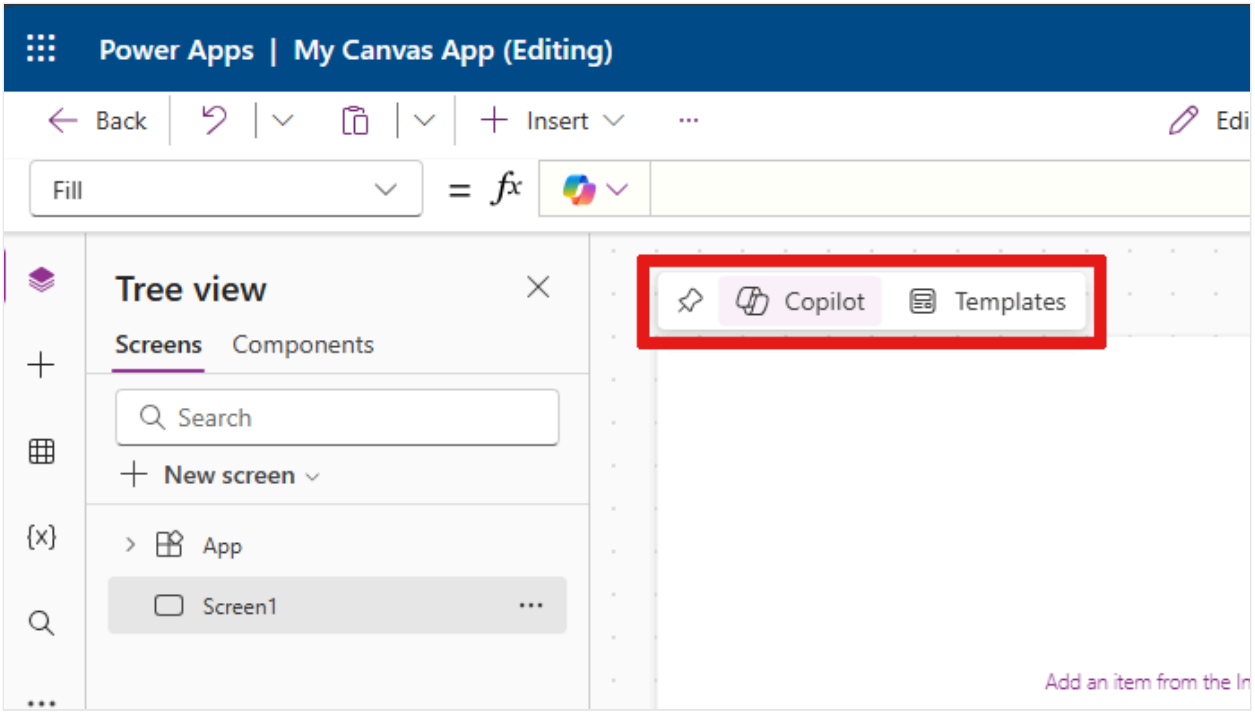
published app, your screen size is temporary.

Last updated on 03/25/2026

Use inline actions in Power Apps Studio

Article • 10/29/2024

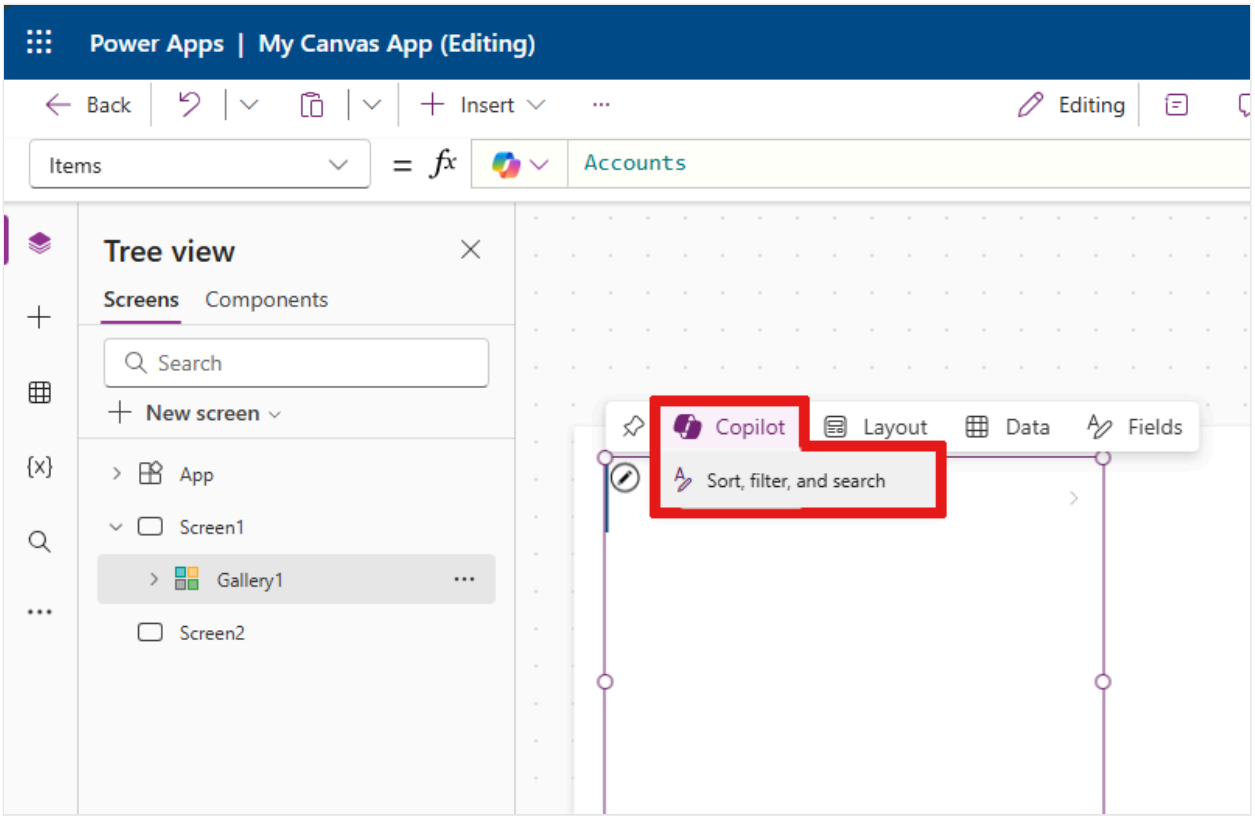
Inline actions are hidden menus that appear when you hover over screen layouts such as a blank screen and gallery or form controls. For example, if you hover over the canvas of a blank screen of your app, you see the **Copilot** and **Templates** inline actions appear.



Use inline actions to update layouts, work with data, collaborate with your colleagues, and use Copilot directly on the canvas for specific controls. This means you can build and modify your app without navigating away from your current view in Power Apps Studio. Inline actions provide a more streamlined and efficient way to work.

Use Copilot inline

Use Copilot inline to help you write Power Fx formulas. Use Copilot to apply conditional formatting, sort and filter your data, and more with the help of Power Apps Ideas. The Copilot button appears where applicable, offering suggestions for supported controls such as **Gallery**, **Data table**, **Text input**, **Drop down**, **Combo box**, and **Text label**. For more information, see Power Apps Ideas.



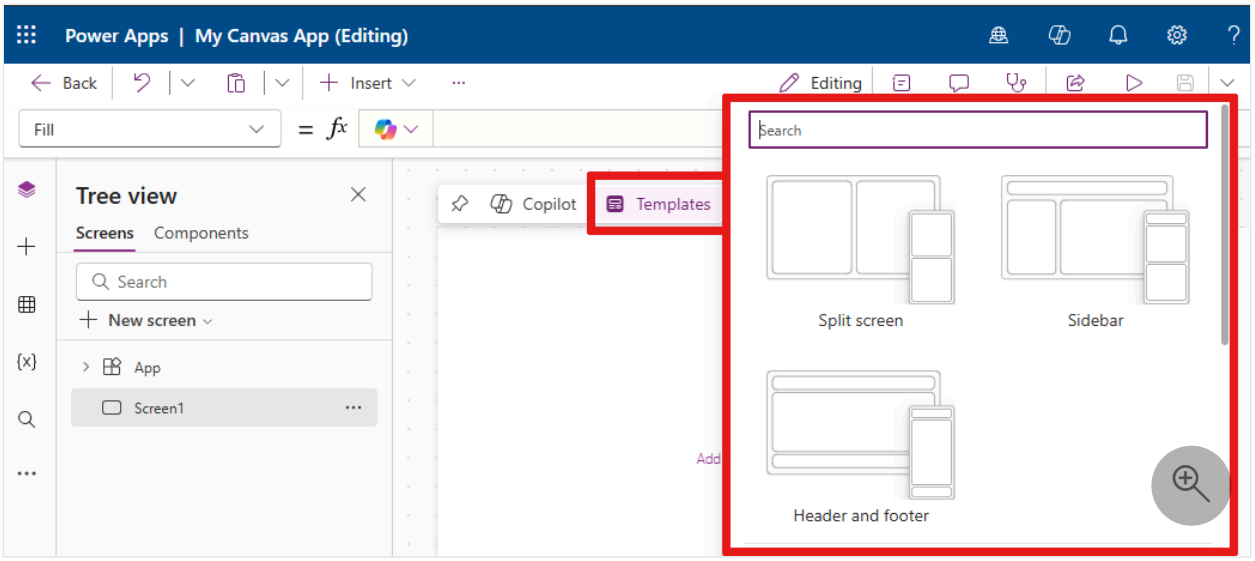
Choose a layout

Inline actions can help you with layout options to create the look you want.

Blank screens

You can use inline actions to apply a layout to blank screens. For example, if you add a new blank screen to your app and hover over the canvas, you can select the inline action **Templates** and then select one of more than one dozen layouts.

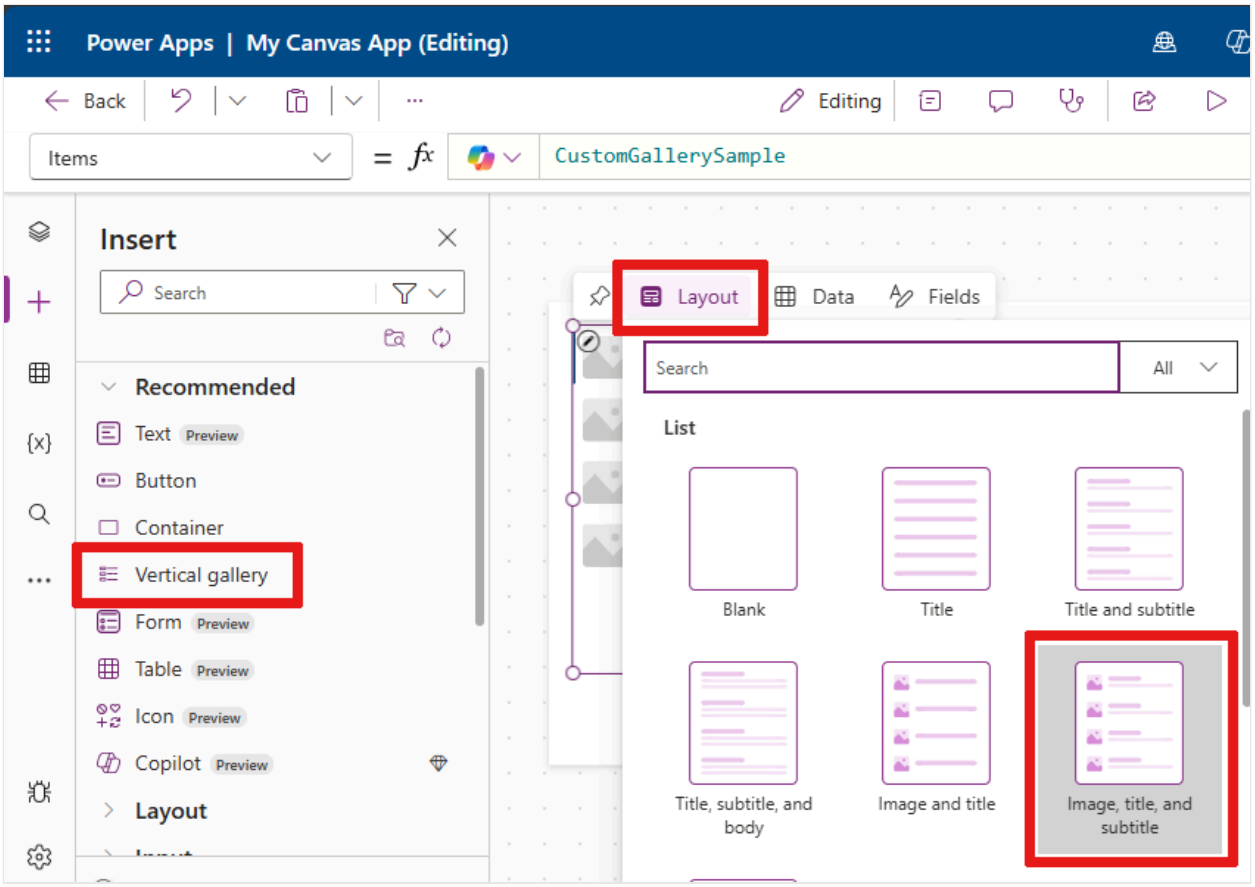
> **Note**
>
> Once you add content to a screen, the **Templates** button is no longer available unless you remove the content.



Gallery control (classic) layouts

You can insert a gallery control, hover over the control, and then use the **Layout** inline action to change the layout for that control. Different layout options appear when you add a **Vertical gallery** or **Horizontal gallery**.

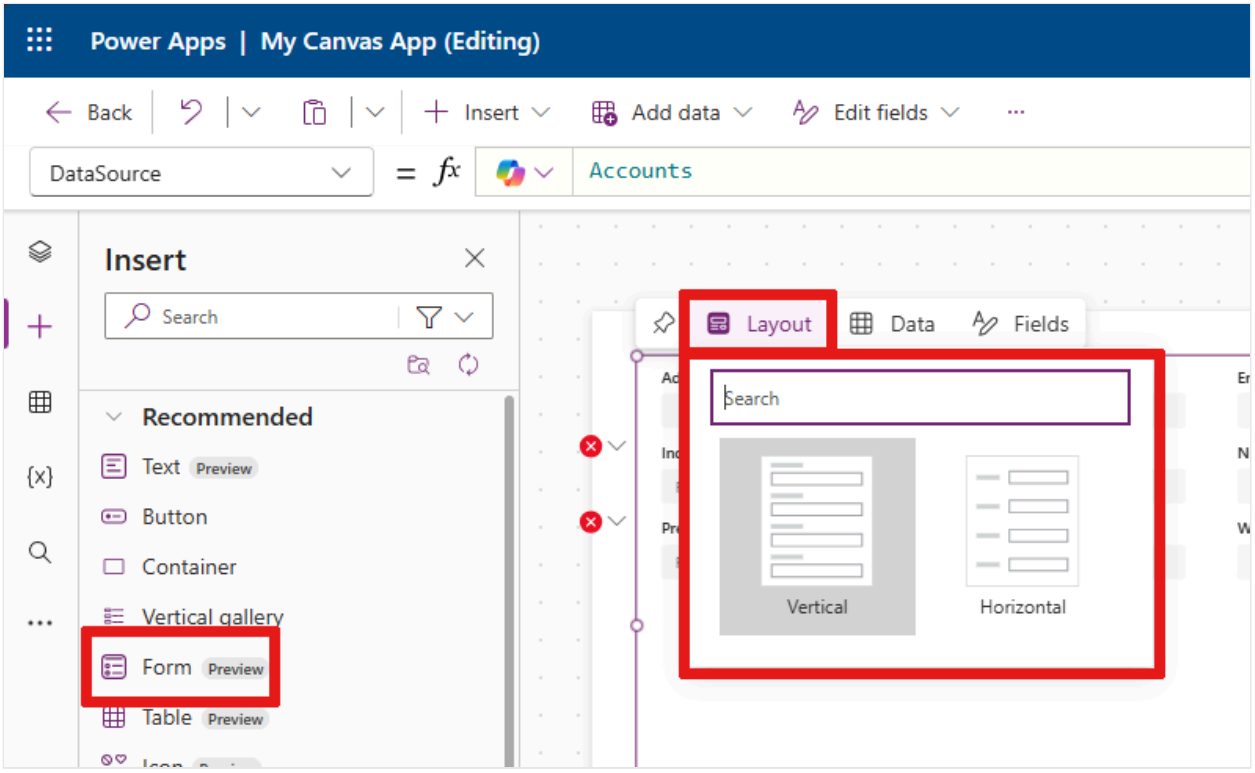
> **Note**
>
> When you change or remove controls after applying a layout, your customizations are removed.



Form control (modern or classic) layouts

When you add a modern or classic **Form** control, you're prompted to add a data source. Once the control is linked to data and added to your screen, you can see the **Layout** inline action appear when hovering over the control.

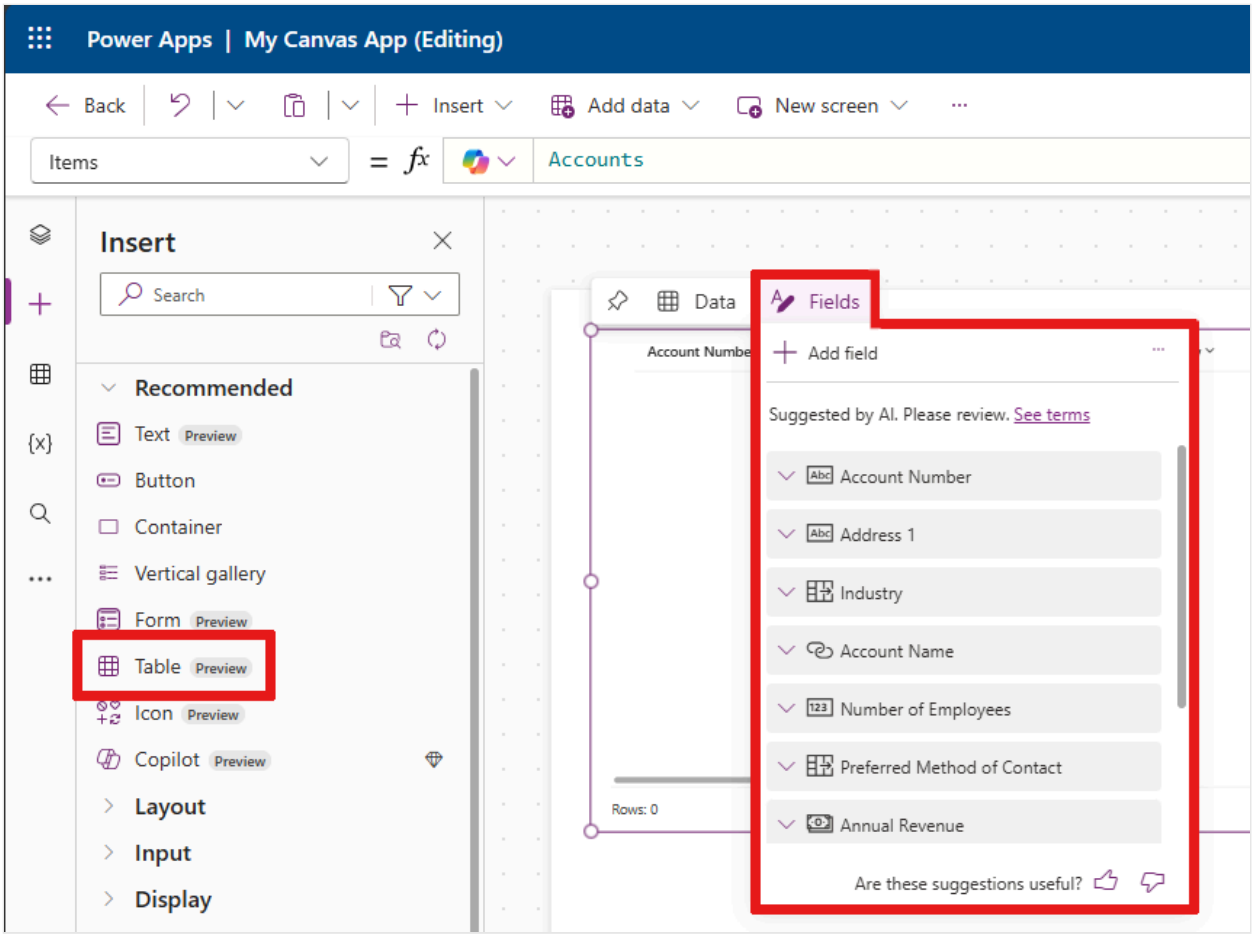
You can then use the **Layout** option to select whether the labels and inputs are arranged vertically or horizontally.



- Select the **Vertical** layout to place the label above the corresponding input control. For example, you might want a title displayed with a description below the title.
- Select the **Horizontal** layout to place the label to the left of the input control. For example, you might want an image on the left side of your list item and a name on the right side of your list item.

Add Data and Fields

When you link any control with data, you see the **Fields** inline action fill with fields from the data. For example, linking a **Table** control with data populates the **Fields** inline action with fields directly from the data such as **Account Number**, **Address 1**, and **Account Name**. You can choose which fields you want to display or add a new field.

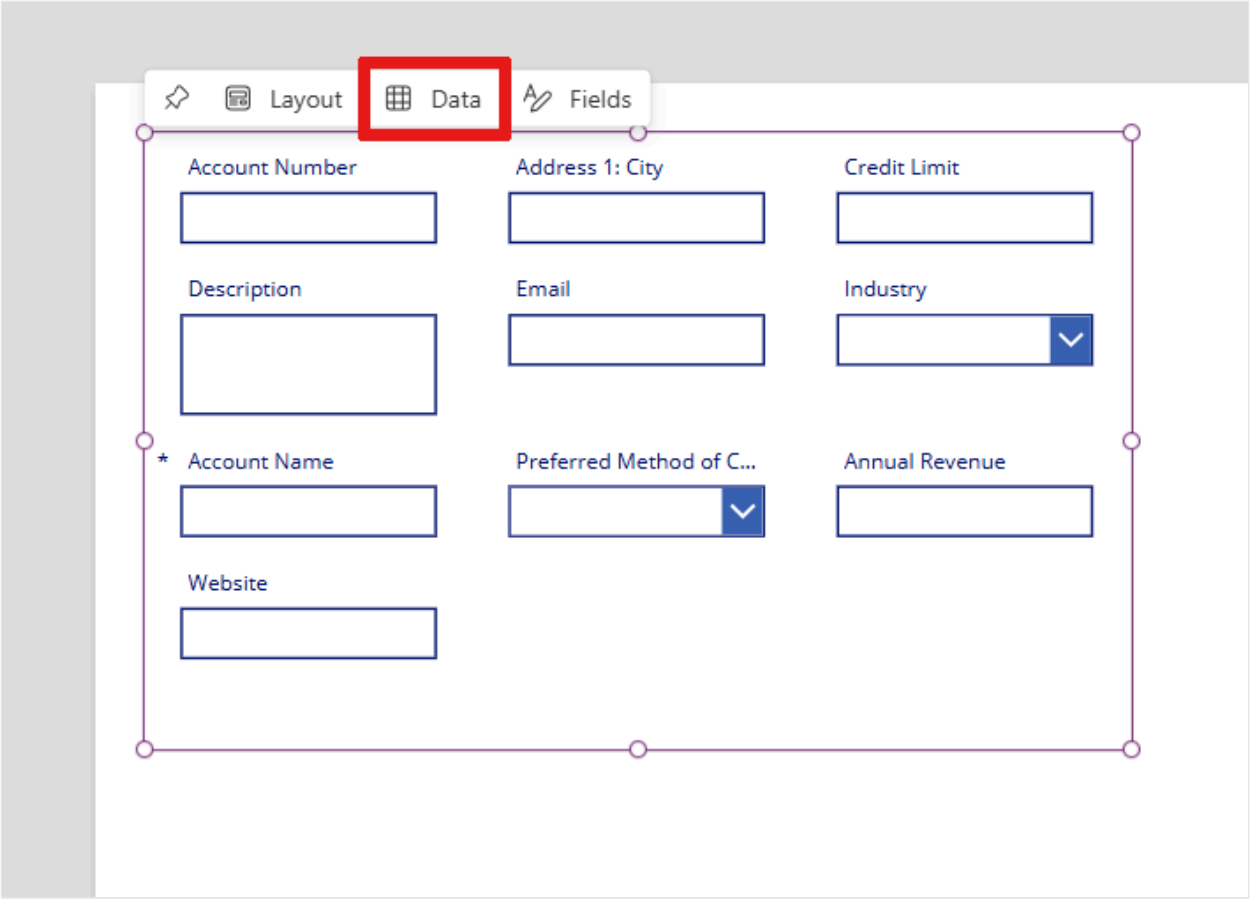


Gallery, Table, and Form

When you add a **Gallery**, **Table**, or **Form** control, you're prompted to add a data source through the **Select a data source** popup. Once you add a data source, the control displays the data. Use the **Fields** inline action to select which fields are shown on the control from the selected data source.

- In a **Table** control, you can drag and drop the fields to change the order that they appear in the table.
- In a **Form** control, if you choose a new data source, you're prompted to replace the data card. When you confirm your selection, it removes any customizations and generates a new form with fields corresponding to the new data source.

This form control has the **Accounts** data set as a data source. If you change that source by selecting the **Data** inline action, all customized fields are removed or replaced with fields from the new data.



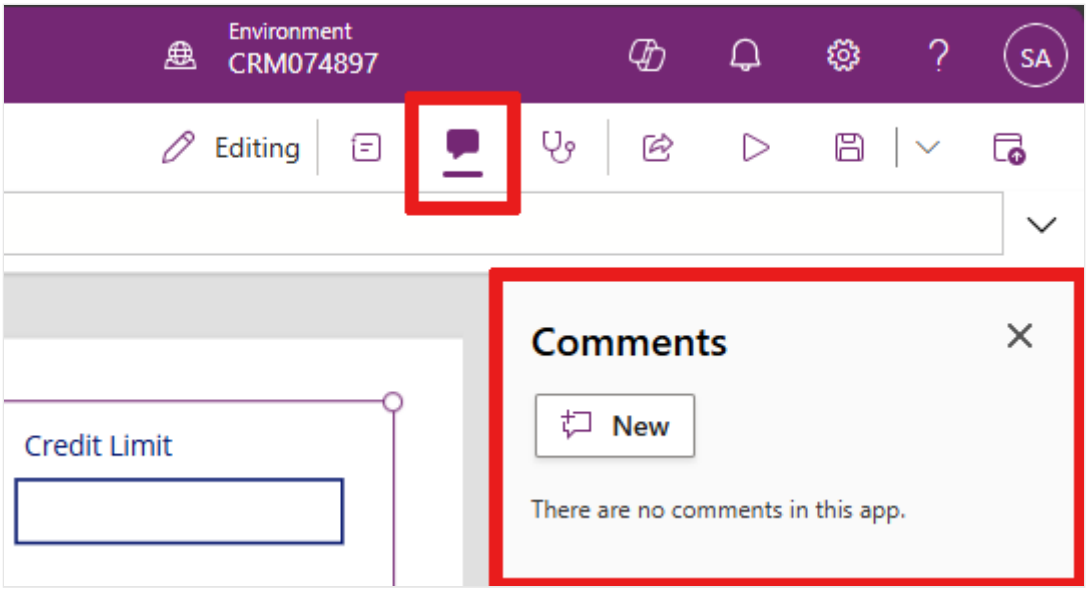
Tip

You can always access the **Data** and **Fields** inline actions to modify data used in the **Form**, **Gallery**, or **Table** control. These inline actions are available when hovering, whether or not you've selected a data source already.

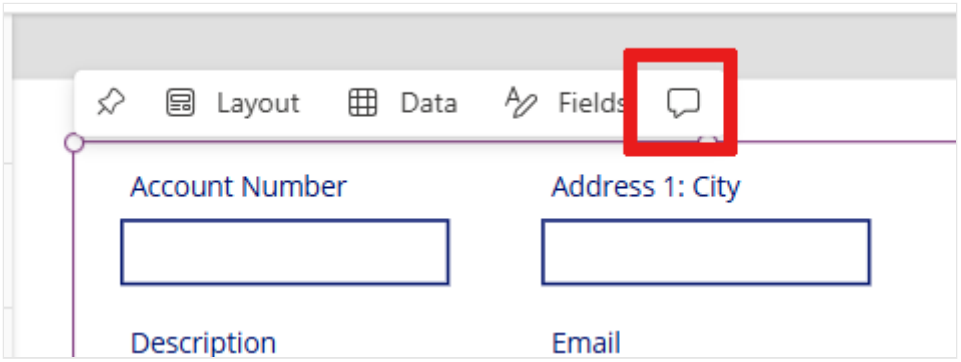
Add comments

Comments allow you to tag peers in your app to ask questions, leave instructions, and propose next steps. You can add and read comments on a screen and any control from inline actions.

To add a comment, select the **Comments** icon in the command bar to show the **Comments** pane. Select the **New** button to start writing a comment.



Once the comment is posted, you see a comment icon in the inline action menu, when you hover over a control.



If you remove all comments from a control, the comment icon in the inline action menu is removed too.

Feedback

Was this page helpful? Yes No

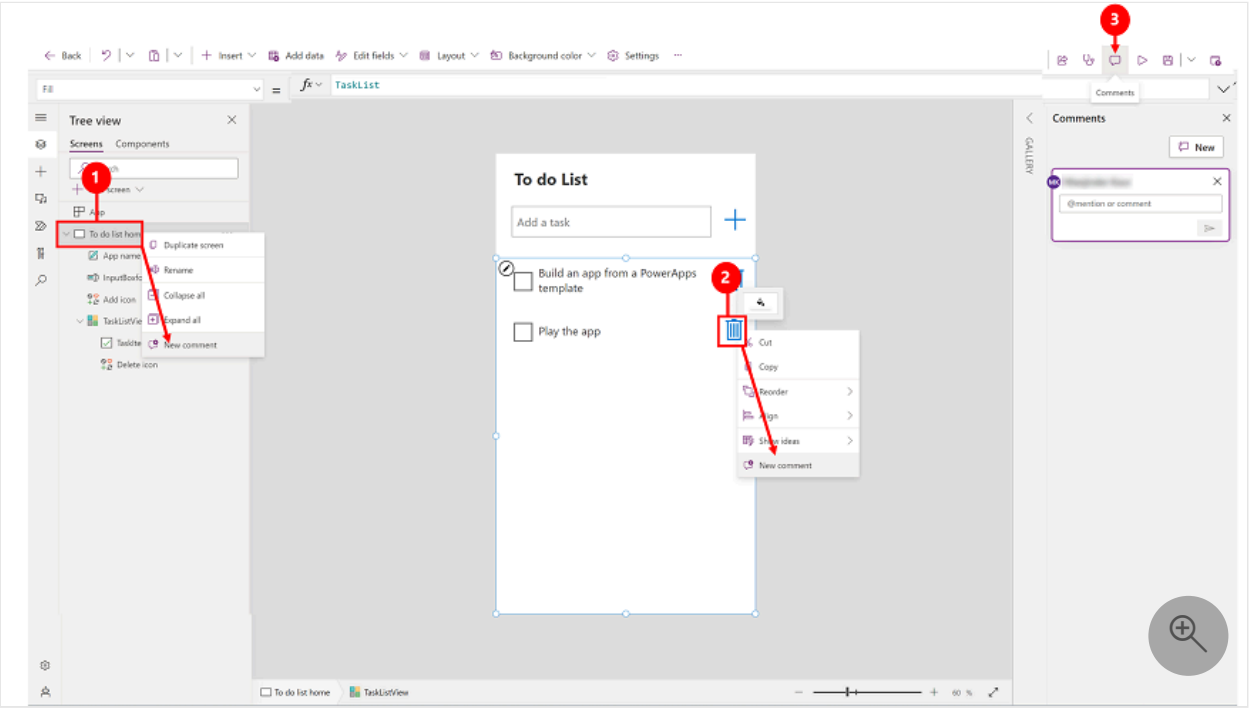
Provide product feedback

Add comments when you're building a canvas app

Article • 03/07/2025

Easily add comments when you're building a canvas app in Power Apps Studio. Use comments to help your team review the app and provide feedback, or provide additional information on implementation details in your app. To allow other makers to add comments, make sure to share the app with them.

You can add comments to different components or pages.



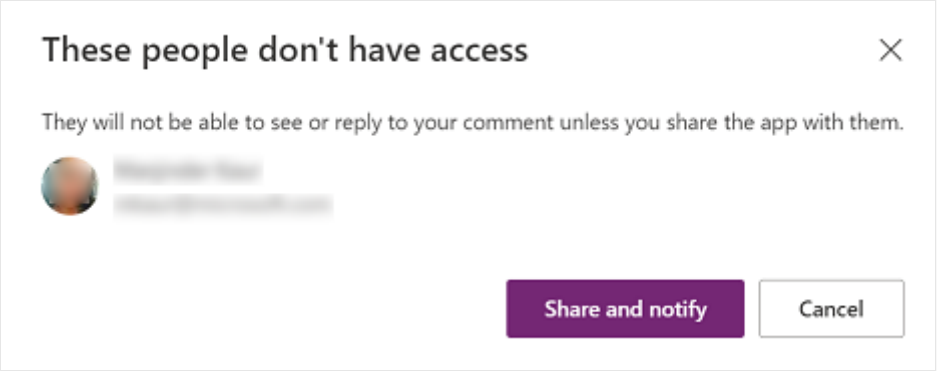
Legend

1. **Tree view**: Select a screen from the tree view and right-click to add a **New comment**.
2. **Component**: In the canvas area, right-click on a component to add a **New comment**.
3. **App actions**: Select **Comments** > **New** and then add your comments.

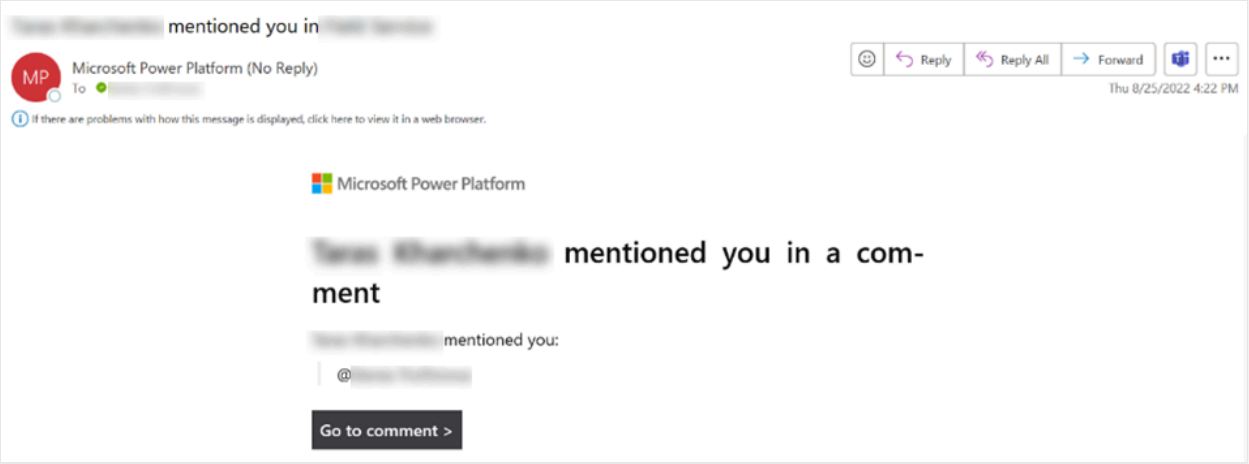
Use @mention in comments to tag someone for feedback

Connect with other makers in your organization as you collaborate on creating your app. You can tag people in the comments by using @+ their name.

If you tag a maker that doesn't have access to your app, then you're prompted to share the app with them.



When someone tags you in a comment, you receive an email that lets you know which app you're tagged in and the person who tagged you. It also includes the comment text and a direct link to that comment.



Edit or delete a comment

You can edit your comments or remove existing comments from appearing in the app.

In the **Comments** pane, next to a comment, select **...**, and then select **Edit comment** or **Delete comment**.

Resolve a comment

Comments are commonly questions, feedback, or ideas that are important context to live alongside your app. But they might not be active discussions anymore. You can resolve a comment or reopen a resolved comment thread to better track the active comments.

1. In the **Comments** pane, next to a comment, select **...**, and then select **Resolve comment**.

2. The comment thread appears as **Resolved**.

3. Select one to remove the comment:

 **Reopen a thread**.

 **Delete a thread**.

Limitations

1. The numbers of users who can access an app in edit mode is limited. If you have an app open and tag someone, the user receives an email notification. The user must wait for you to exit the app in order to see the comment and resolve it.
2. Comments aren't supported when you're working on an app using Git.
3. @mention isn't supported for SharePoint custom Power Apps. Learn more in Customize a form for a SharePoint list.

Feedback

Was this page helpful? Yes No

Provide product feedback

Understand coauthoring and copresence in canvas apps

Summarize this article for me

Collaborate on canvas apps in Power Apps Studio by using copresence and coauthoring. These features let you work with other makers in real time, improving productivity and teamwork.

https://learn-video.azurefd.net/vod/player?id=719b89e6-32ce-498b-9597-600d91019ec4&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fcopresence-power-apps-studio

What you'll learn:

- How to see who's working on your app and work together effectively
- When to use copresence vs. coauthoring for your collaboration needs
- Step-by-step instructions to enable and use these features

Key differences:

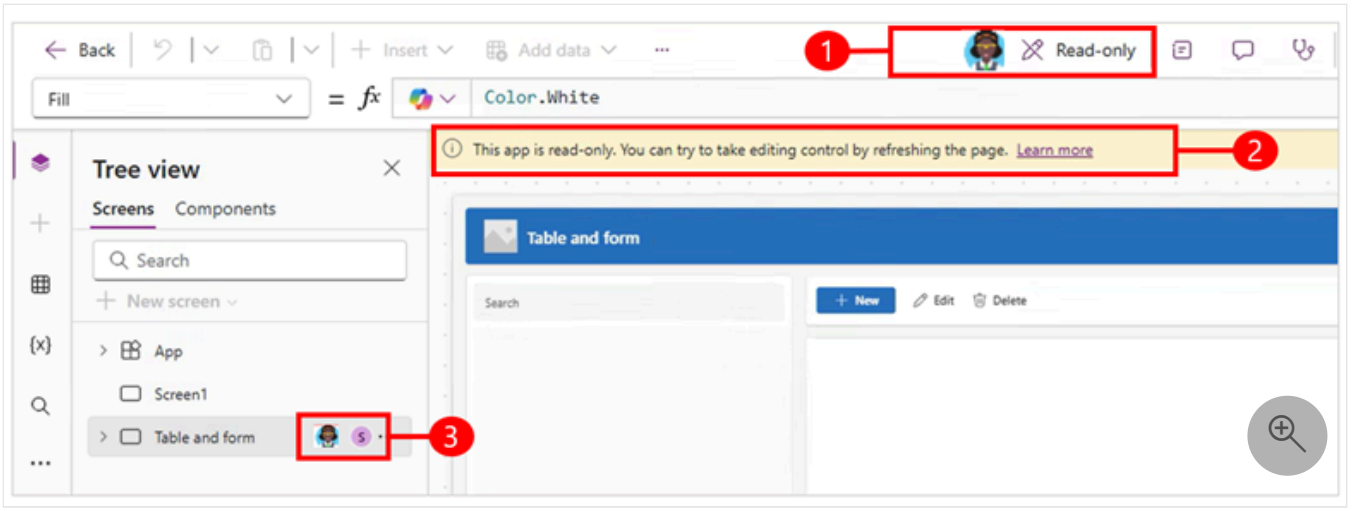
- **Copresence** shows who else is working on the same app, but only one person can edit the app at a time. The first person who opens the app gets editing control.
- **Coauthoring** lets multiple people edit the app at the same time and see each other's changes in real time.

Use copresence to see who's working on the same app

When you open an app for editing in Power Apps Studio, indicators might show that other people are also working on the app. These indicators are part of the copresence feature.

The first person who opens the app gets editing control. If another person opens the app, a notification lets them know that someone else is editing the app, and they're viewing it in read-only mode. In read-only mode, you can't add new screens, edit control properties, or use the command bar. You can save a copy of the app.

The command bar shows the names and icons of other makers who are editing or viewing the app. The left navigation pane shows the app's structure and highlights which part of the app someone else is editing. You also get a notification to refresh the app when someone saves changes to the app.



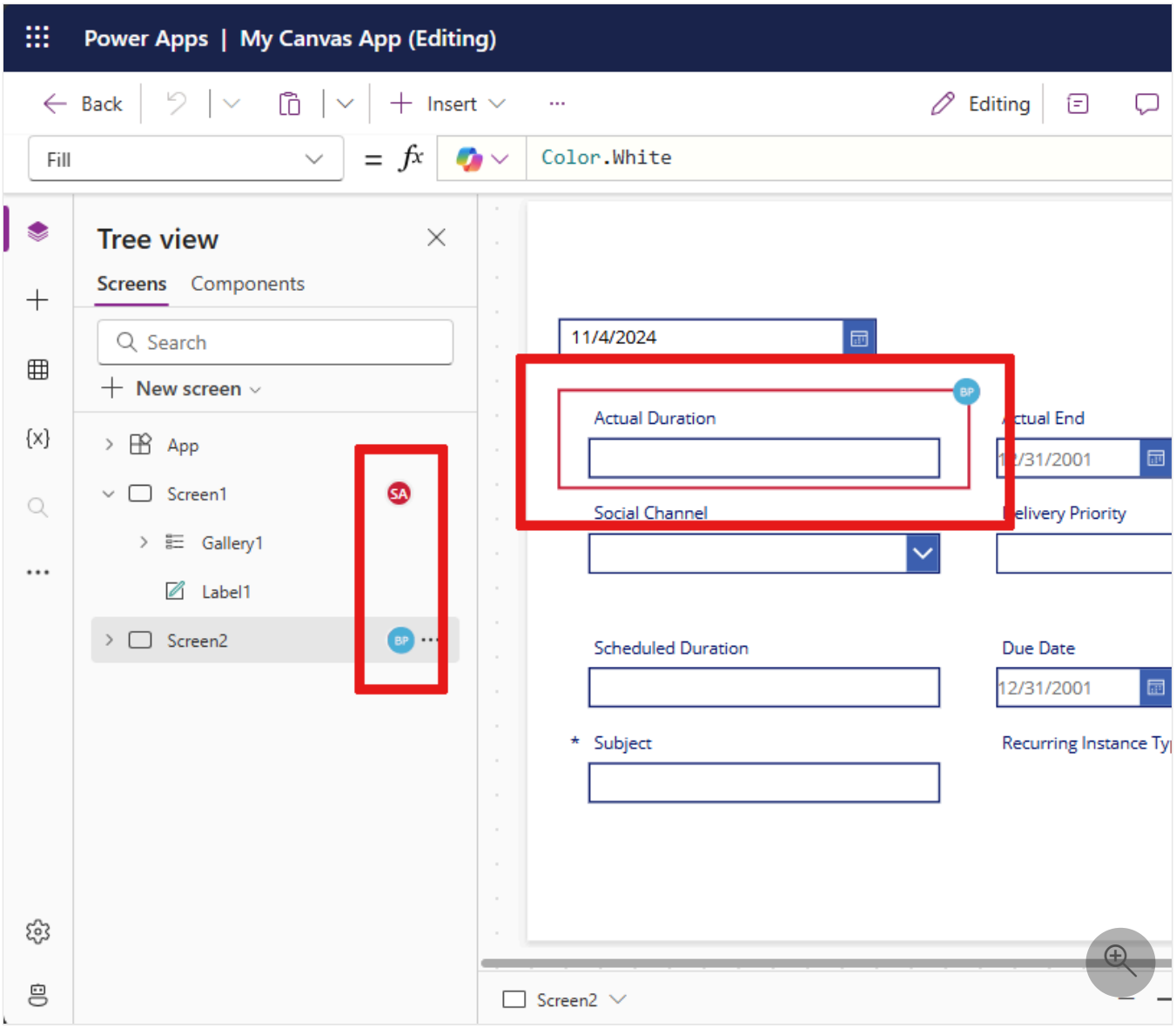
Legend:

1. The command bar shows the names and icons of other makers who are editing or viewing the app.
2. A **Read-only** warning appears if someone already has editing control elsewhere. You can select **Override** to become the main author and regain editing rights.
3. In **Tree view**, you see the app's structure and other people's profile pictures on the part of the app someone else is editing.

> **Tip**
>
> If you're inactive for two hours while editing an app, Power Apps asks if you want to keep editing or switch to read-only mode. If you don't respond, the system puts you in read-only mode so other makers can become the editor.
>
> If autosave is on, your changes are automatically saved. If autosave isn't on, Power Apps notifies you that you're no longer editing and lets you save a copy of your changes.

Use coauthoring to edit the same app together

Coauthoring lets multiple makers edit a canvas app at the same time. When you use coauthoring, you can see where other makers are working on the app and view their changes in real time.



When multiple editors work on the app, their avatars appear.

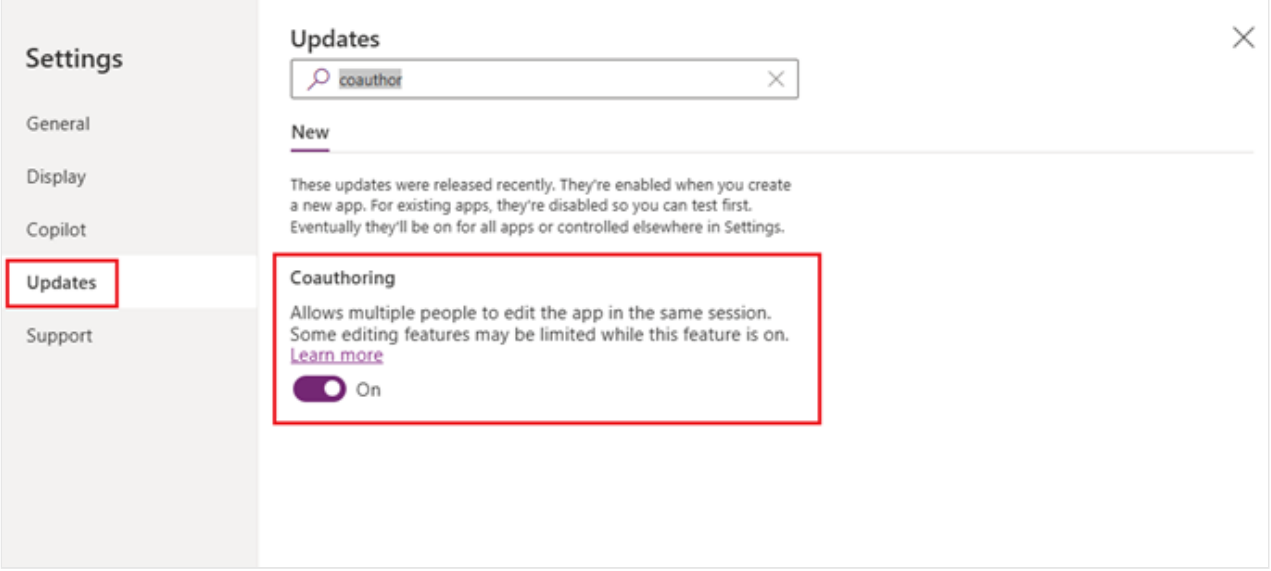
- The left navigation pane shows the app's structure in **Tree view** and shows which part of the app someone else is editing. For example, you might edit **Screen 1**, while someone else edits **Screen 2**.
- The area that another maker edits is highlighted and shows their initials. For example, someone else might edit a part of a form, which is highlighted for the first user.

> **Important**
>
> Multiple makers can select and edit a control at the same time. Be careful not to overwrite each other's changes.

Turn on coauthoring

To use coauthoring, turn it on for each app. When you turn on coauthoring, it overrides the copresence feature.

1. Open your app for editing in Power Apps Studio.
2. Go to **Settings** > **Updates** > **New**.
3. In the search box, type **coauthor**.
4. Turn on the **Coauthoring** toggle.



Limitations of coauthoring

- When more than one maker edits an app, the following actions aren't available:
  - Search
  - Save as
  - Open another or new app
  - Undo and redo
  - Switch authoring versions
- The maximum number of coauthors is 10, either in one session or across a total of 10 tabs, depending on which limit you reach first. Any other coauthors or tabs beyond 10 are in copresence and can't edit the app or view real-time updates.
- The app language is locked to the locale of the first maker who opens the app for editing. Opening the same app in a different locale can lead to errors in formulas.
- Cut isn't available.
- Coauthoring is turned off in the Monitor tool.
- You can encounter a problem with the following actions:

- Renaming a control
- Adding AI Builder components
- Adding geospatial controls
- Running a flow that another coauthor added, without refreshing the app first
- Viewing errors from one author's actions on all other coauthors' screens
- Copying and pasting

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 02/25/2026

Add and configure controls in canvas apps

Article • 10/29/2024

Add a variety of UI elements to your canvas app, and configure aspects of their appearance and behavior directly, from the toolbar, in the **Properties** tab, or in the formula bar. These UI elements are called controls, and the aspects that you configure are called properties.

Prerequisites

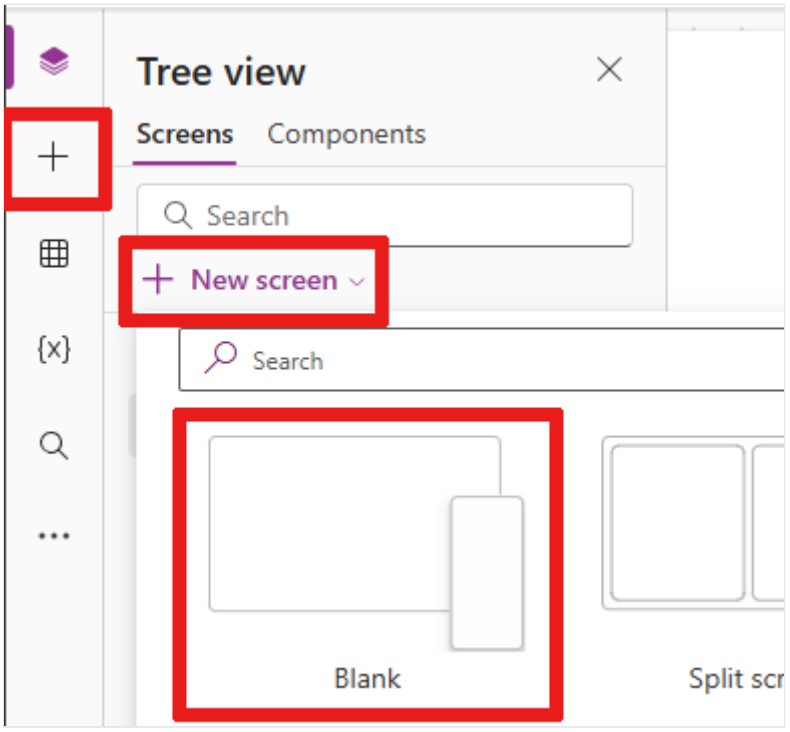
1. If you don't already have a Power Apps license, sign up, and then sign in.
2. Create a blank canvas app.

   You can always take the tour later by selecting the question-mark icon near the upper-right corner of your screen and then selecting **Take the intro tour**.

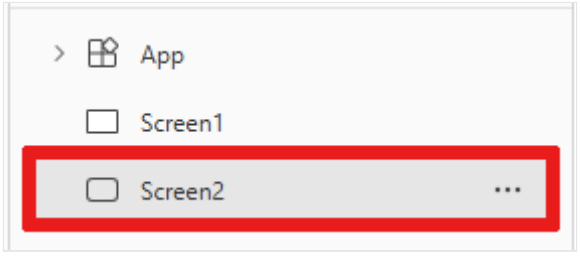
Add and select a control

Let's add a new screen and a button control. However, you can add any type of control.

1. Select **Insert** from the authoring menu, then select **New screen**.
2. Choose **Blank** to add a blank screen to your app. (Screens are a type of control that can contain other types of controls.)

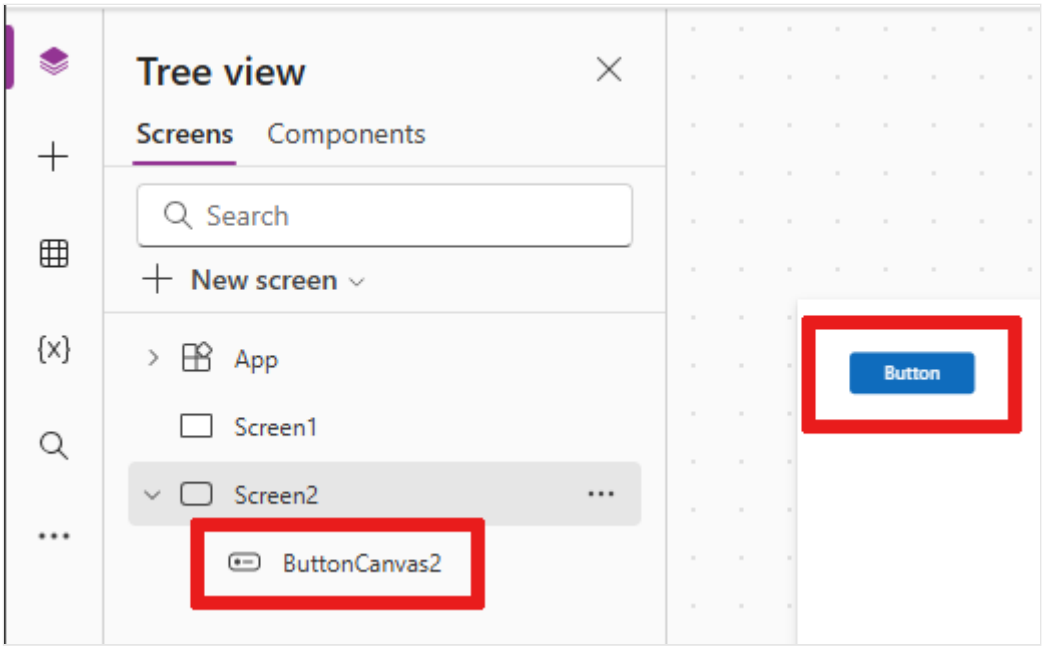


The new screen is named **Screen2** and appears in the **Tree view**. This pane shows a hierarchical list of controls in your app so that you can easily find and select each control.



3. With **Screen2** selected, select **Insert** again.

4. Select the **Button** control. The new control appears under **Screen2** in the hierarchical list of **Tree view**.



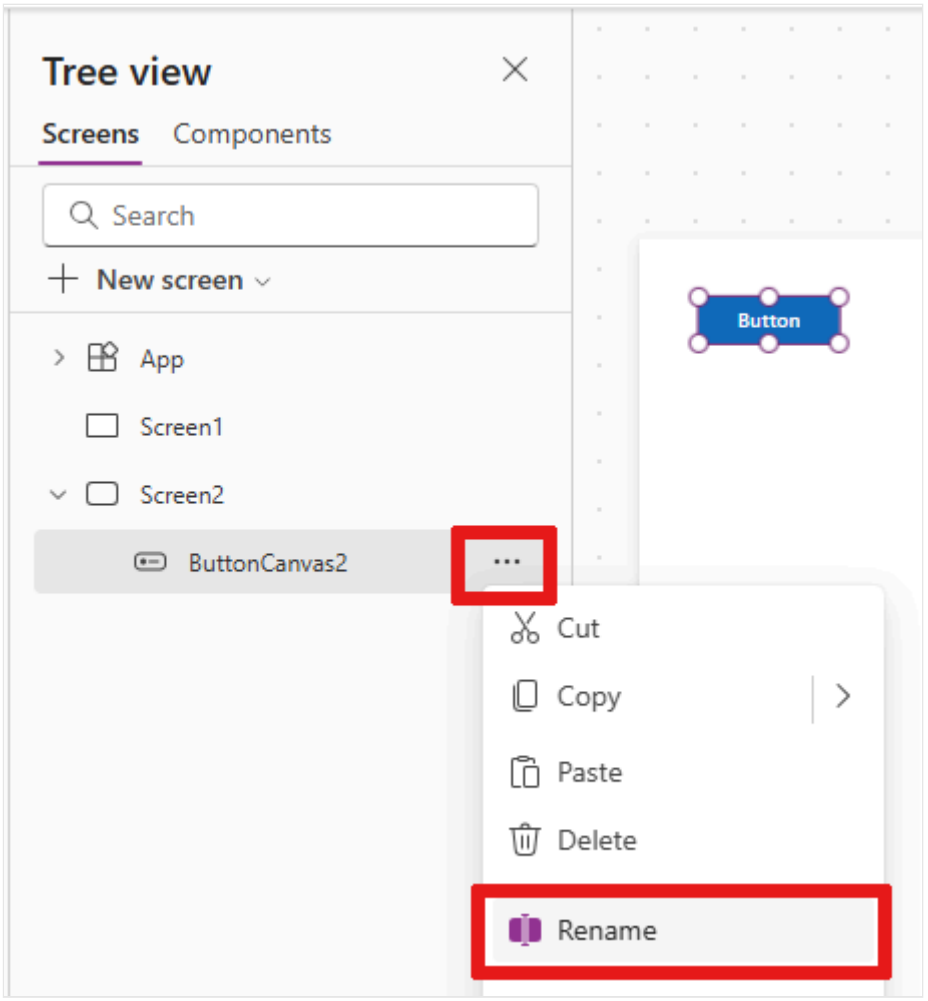
In the screen, a box with six handles surrounds the button by default. These handles surround any control selected and are used to size the control.

5. Select the screen (outside the button) and the box handles disappear. To select the control again, select it directly on the canvas or from the **Tree view** list.

> **Important**
>
> You must always select a control before you can configure it.

Rename a control

1. In the list of controls, hover over the control that you want to rename to reveal the overflow menu (ellipsis).
2. Select the ellipses.
3. Select **Rename**. Add a unique, memorable name to make building your app easier.



Delete a control

You can delete a control either through control in the **Tree view** list or directly from the canvas.

Delete a control from Tree view

1. In the list of controls, hover over the control that you want to delete to reveal the overflow menu (ellipsis).
2. Select the ellipses.
3. Select **Delete**.

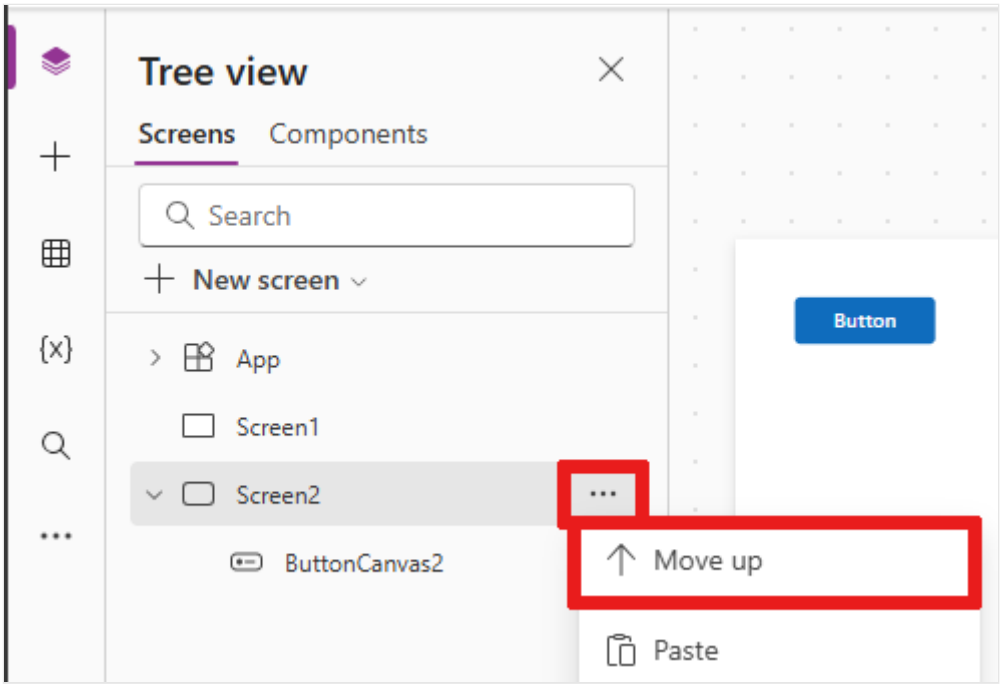
Delete a control from the canvas

Select the control on the canvas, then press the `Delete` key.

Reorder screens

1. In **Tree view**, hover over a screen that you want to move up or down to reveal the overflow menu (ellipses).

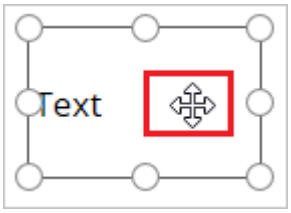
2. Select the ellipsis, then choose **Move up** or **Move down**. If you don't see a **Move down** option, your screen is at the lowest point in the list already.



> [!NOTE]
> When the app is opened, the screen at the top of the list of controls usually appears first. However, you can specify a different screen by setting the OnStart property to a formula that includes the Navigate function.

Move and resize a control

To move a control, select it, hover over its center so that the four-headed arrow appears, and then drag the control to a different location.



To resize a control, select it, hover over any handle in the selection box so that the two-headed arrow appears, and then drag the handle.

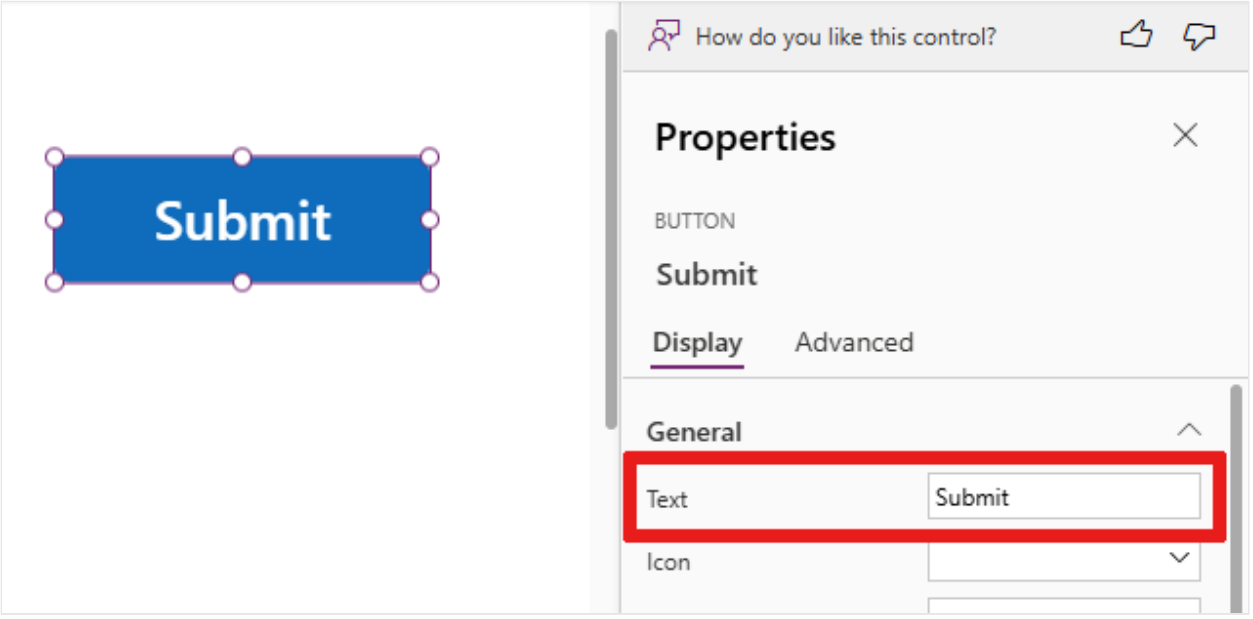


> **Note**
>
> As this topic describes later, you can also move and resize a control by modifying any combination of its X, Y, Height, and Width properties in the formula bar.

Configure a control in the Properties pane

By using the **Properties** pane, specific to the control you select, you can specify a wide variety of options.

For example, if you select a button control, you can replace the value of the text displayed through the control's **Properties** pane.



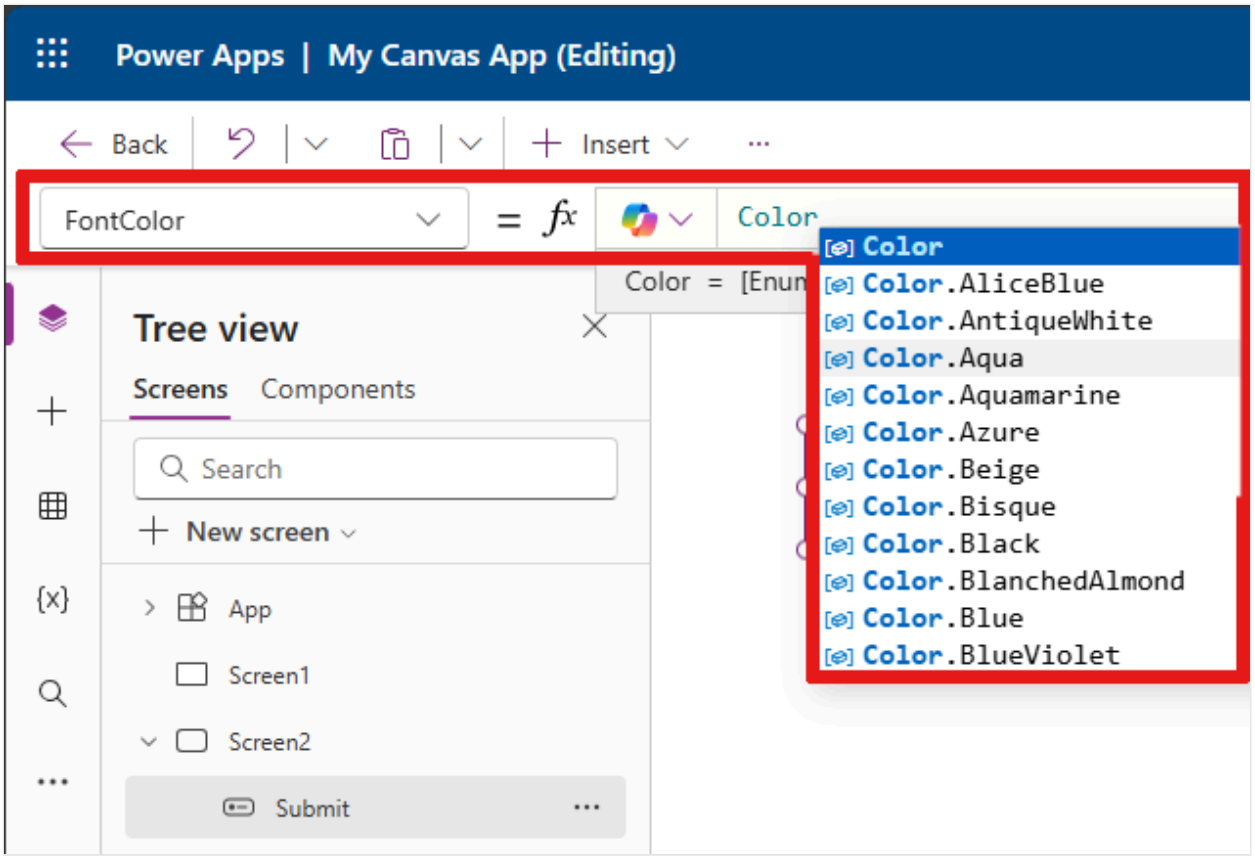
> **Note**
>
> You can alternatively change display text by modifying its Text property in the formula bar.

Configure a control in the formula bar

You can configure a control by selecting a property in the property list and then specifying a value in the formula bar. Autocomplete can be helpful in the formula bar to see values you can use for a control.

Change the font color of a control

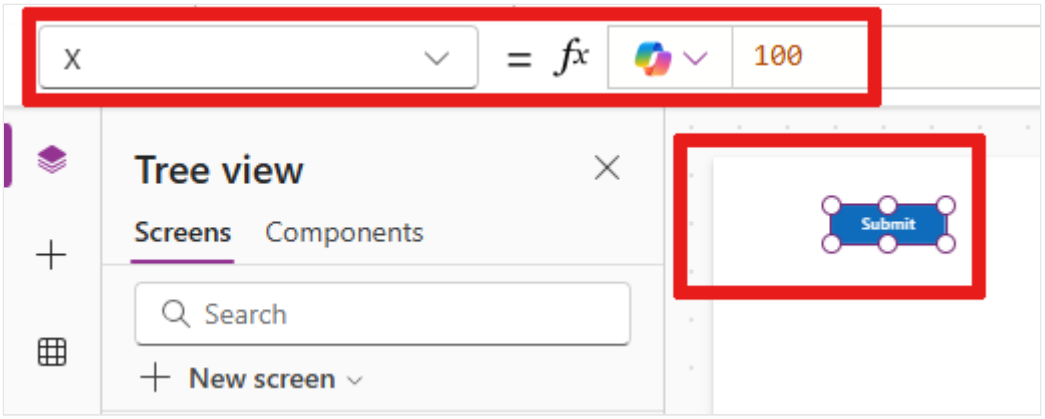
For example, if you want to change the font color for your control's text. You see many color options.



Change the position on the canvas of a control

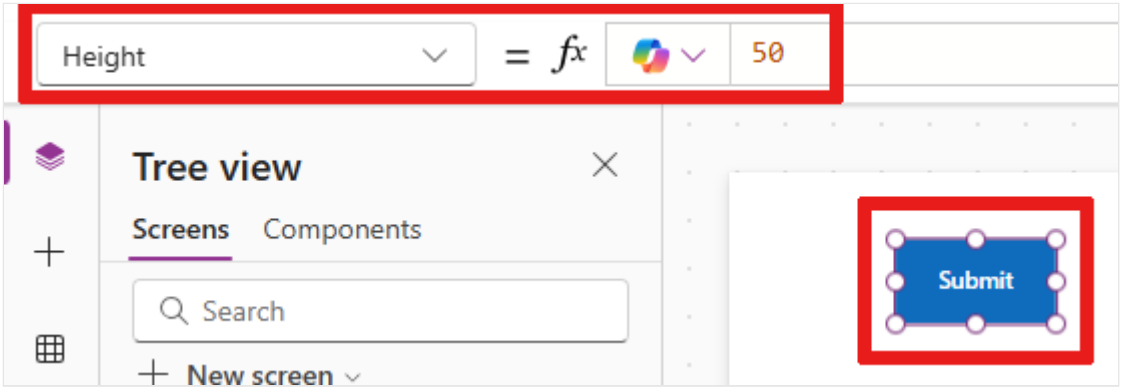
You can set the X and Y coordinates for the position of the control using the formula bar. Setting coordinates in the formula bar gives you greater precision of control placement.

For example, if you select **X** or **Y** in the properties list and specify a number in the formula bar, you see the control move to that coordinate. In this case, the button moves to position 100 on the X axis, assuming the far left side of the canvas is the zero point.



Change the control's size

You can resize a control by selecting **Height** or **Width** in the properties list and then specifying a different number in the formula bar.

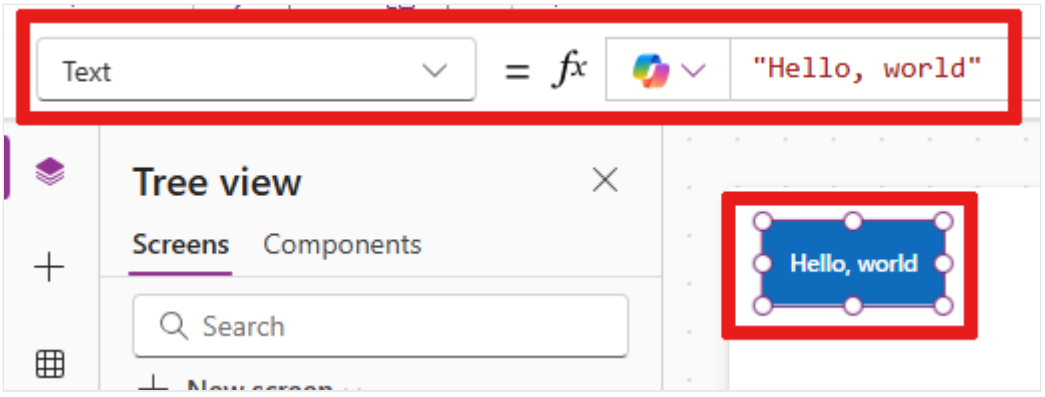


Change the control's display text

You can change a control's display text by selecting **Text** in the properties list and then specifying any combination of a literal string, an expression, or a formula in the formula bar.

Literal string

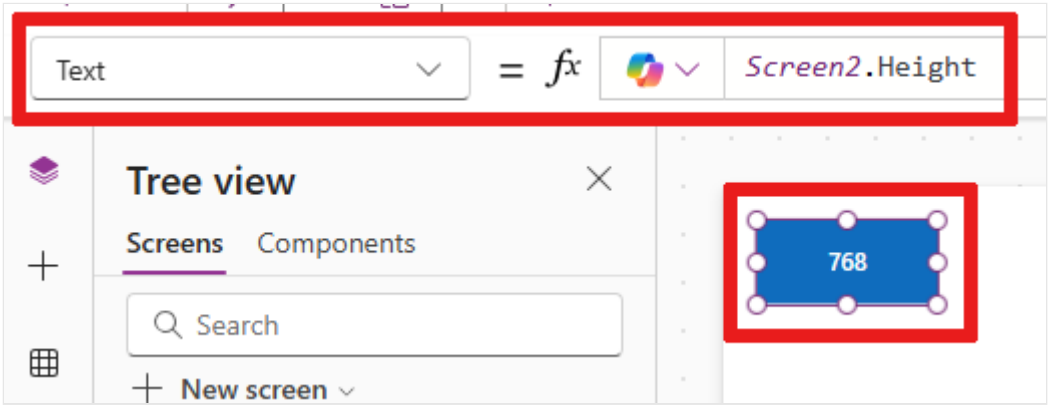
A literal string is surrounded by quotation marks and appears exactly as you type it. **"Hello, world"** is a literal string.



> **Note**
>
> When you add text that is longer than the width of your control, the control doesn't change its width to accomodate the text. You must manually change the control's size to fit the text.

Expression

An expression is often based on a property of another control. **Screen2.Height** is an expression that shows the height of **Screen2** where the display text is shown for a button control.

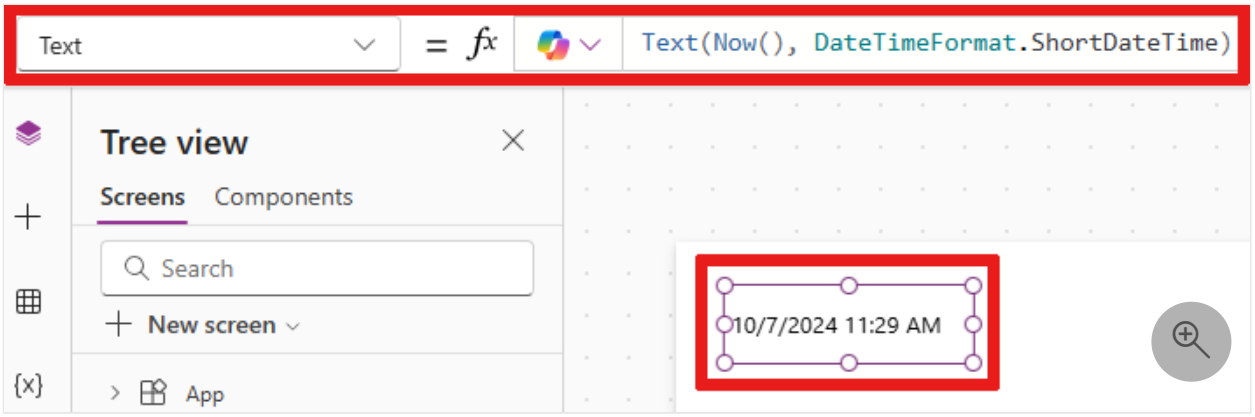


Using an expression, for example, you can set a background image or icon as the fill for your button.

Formula

A formula includes one or more functions. The **Now** function returns the current date and time in your local time zone, and the **Text** function formats values such as dates, times, and currency.

For example, you can display the date and time in a **Label**:



Formulas are usually much more complex than this example so that they can update data, sort it, filter it, and perform other operations. For more information, see the formula reference.

Related information

- Find step-by-step procedures for configuring common controls such as screens, lists, galleries, forms, and charts.
- Find reference information about each type of control in the control reference.

Feedback

Was this page helpful? Yes No

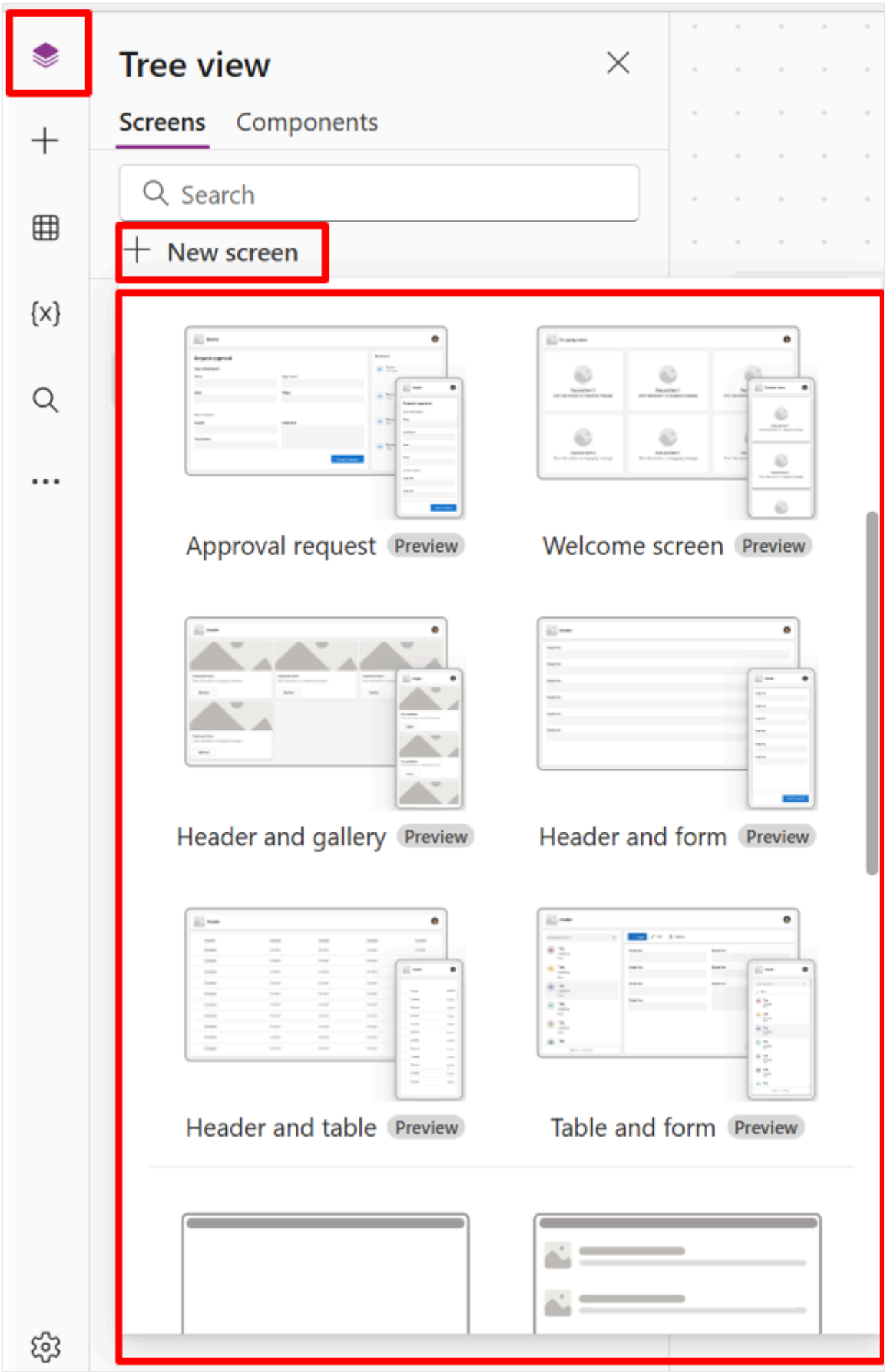
Provide product feedback

Add and navigate screens in canvas apps

Create modern, responsive apps by adding prebuilt screens for common app scenarios. The screens feature responsive containers and modern controls that adapt to different screen sizes. You can also add custom screens with different layouts and controls.

The following new prebuilt screens are available:

- Approval request
- Welcome screen
- Header and gallery
- Header and form
- Header and table



Add a new screen

1. Sign in to Power Apps.
2. Create a canvas app or open one for editing.
3. On the command bar, select **New screen**, and then select a screen layout.
4. Preview the app to determine how it looks on different devices. For more information, see Preview an app.

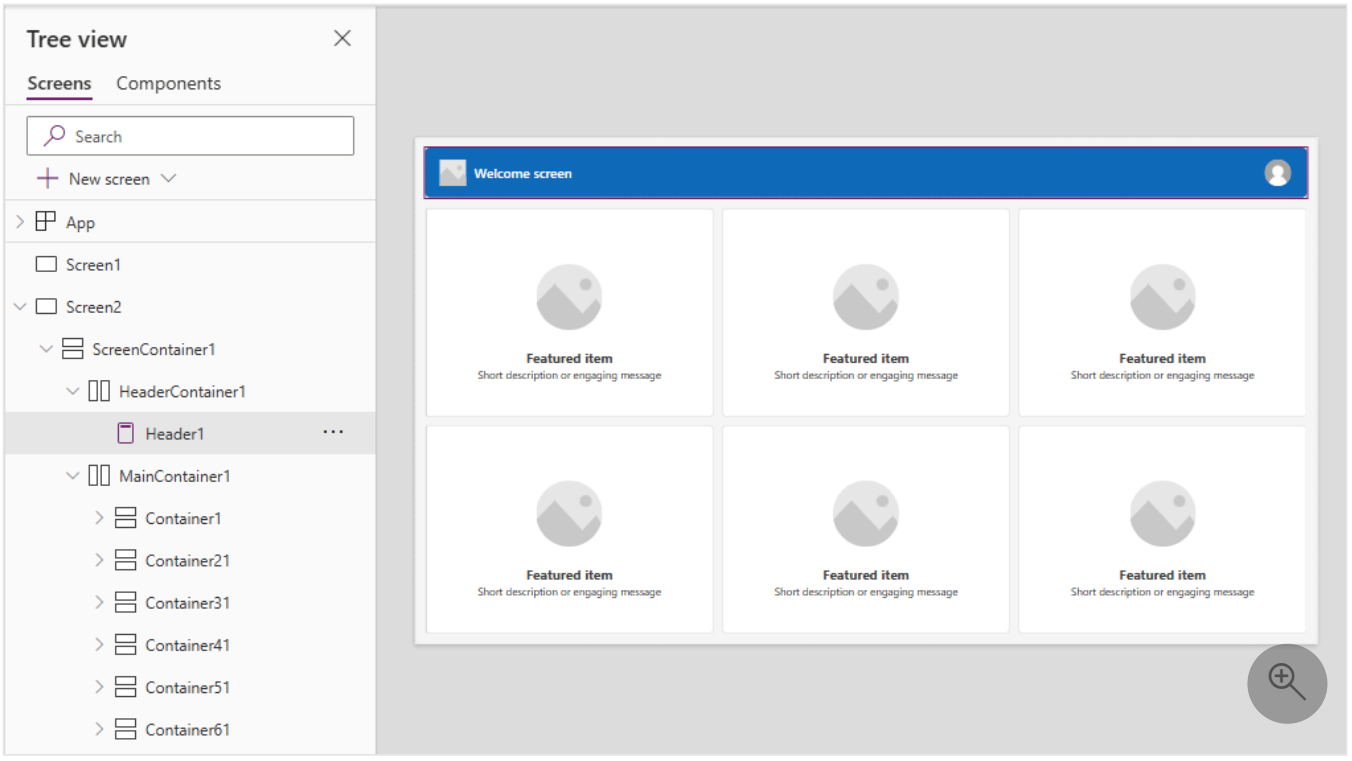
> **Tip**
>
> To make the app adapt to the size of the device display it's running on, turn off the **Scale to fit** option.

Welcome screen

The **Welcome screen** works well as the first screen of an app. You can customize each tile with an image, a title, and a description. Add or remove tiles in the main container to change the number of tiles. Use the tiles to guide users to other parts of the app.

The **Welcome screen** template includes the following controls:

- Screen container
  - Header container
    - Header
  - Main container
    - Container
      - Image container
        - Image
      - Title container
        - Feature item button
        - Description text

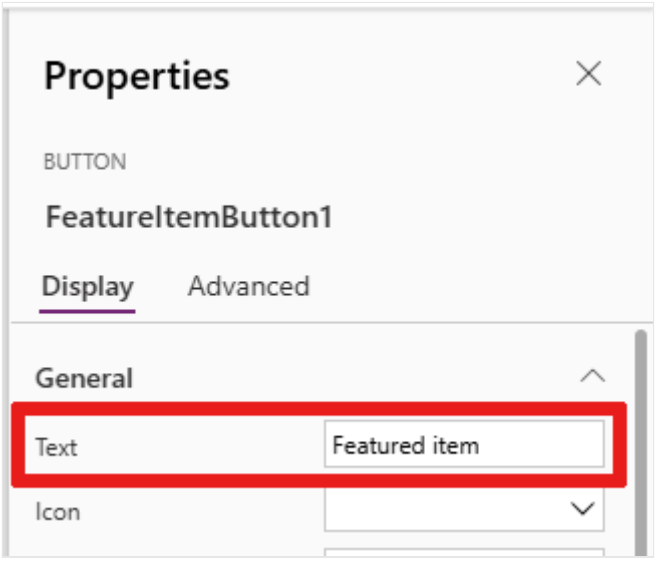


Add and customize the Welcome screen

1. Select **New screen** > **Welcome screen**.

2. To change the image of a tile, select it and then select **Edit**.

3. Select the **Feature Item** button control and add your own text in the Properties pane.



4. Select the **Short description or engaging message** text and add your own.

5. Add and remove tiles if needed.

    - To add tiles in the tree view, copy and paste a **Container** item.

    - To remove a tile in the tree view, right-click a **Container**, and then select **Delete**.

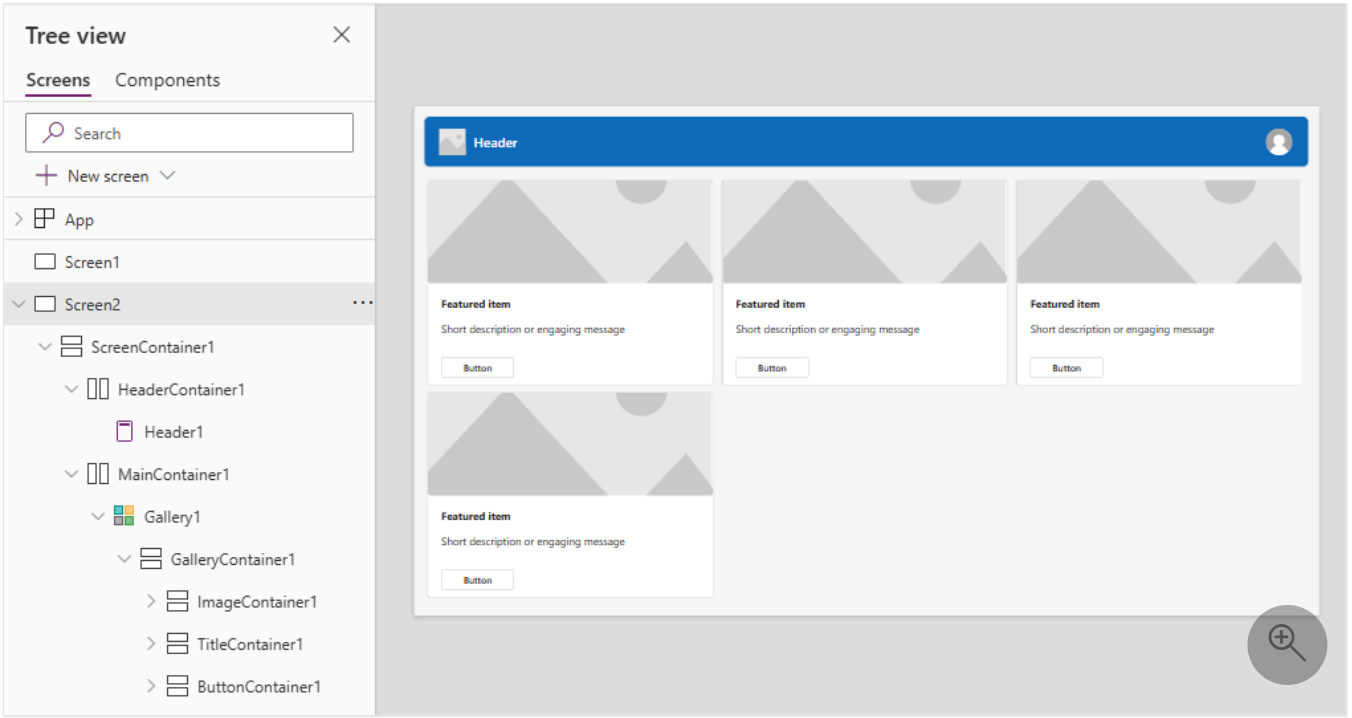
Header and gallery screen

Use the **Header and gallery** screen to show a range of product or service information, like a product catalog. When you connect a gallery control to a data source, a catalog is automatically created with little customization required.

The gallery control in the **Header and gallery** screen is a classic control. When the modern gallery control is released, the **Header and gallery** screen uses it. For more information, see Overview of modern controls and themes in canvas apps.

The **Header and gallery** screen has the following template for controls:

- Screen container
  - Header container
    - Header
  - Main container
    - Gallery
      - Gallery container
        - Image container
          - Image
        - Title container
          - Title text
          - Description text
        - Button container
          - Button



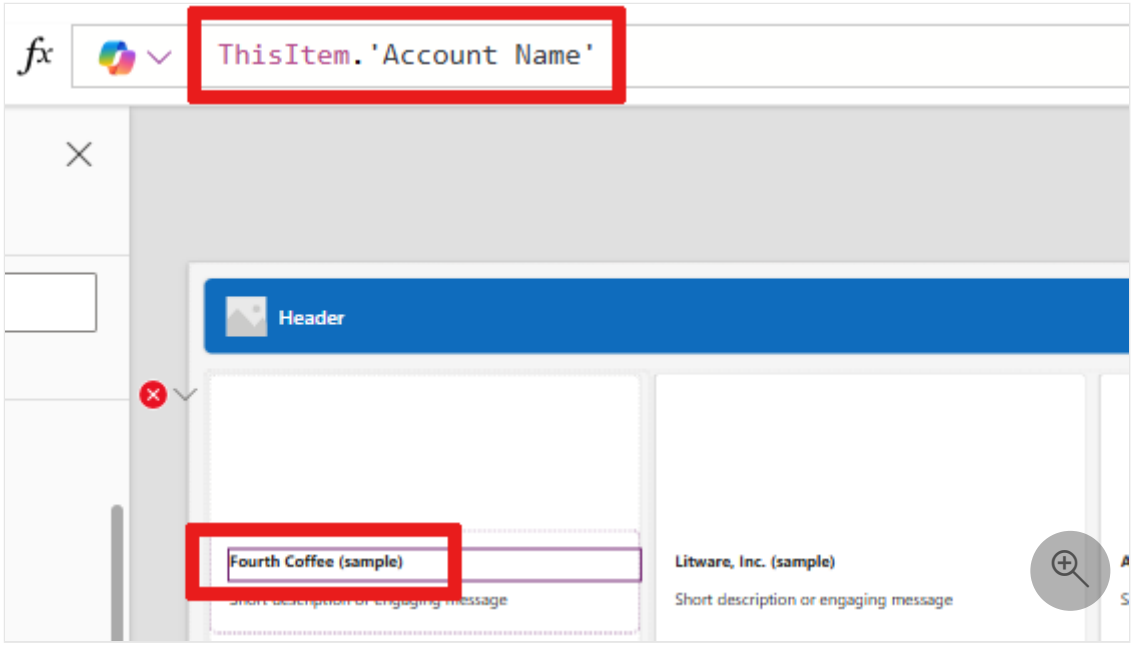
Add and customize the header and gallery screen

1. Select **New Screen** > **Header and gallery**.

2. In the tree view, select **Gallery** and connect it to a data source such as Dataverse.

3. Select specific controls inside the gallery, such as the image, title text, and text description. In the control's properties, use the *ThisItem* syntax to set the desired image, title text, and description.

   For example, to add the **Account name** to the title text control, find it by typing `ThisItem` in the formula bar to see a list of available items to add from your data set.



Approval request screen

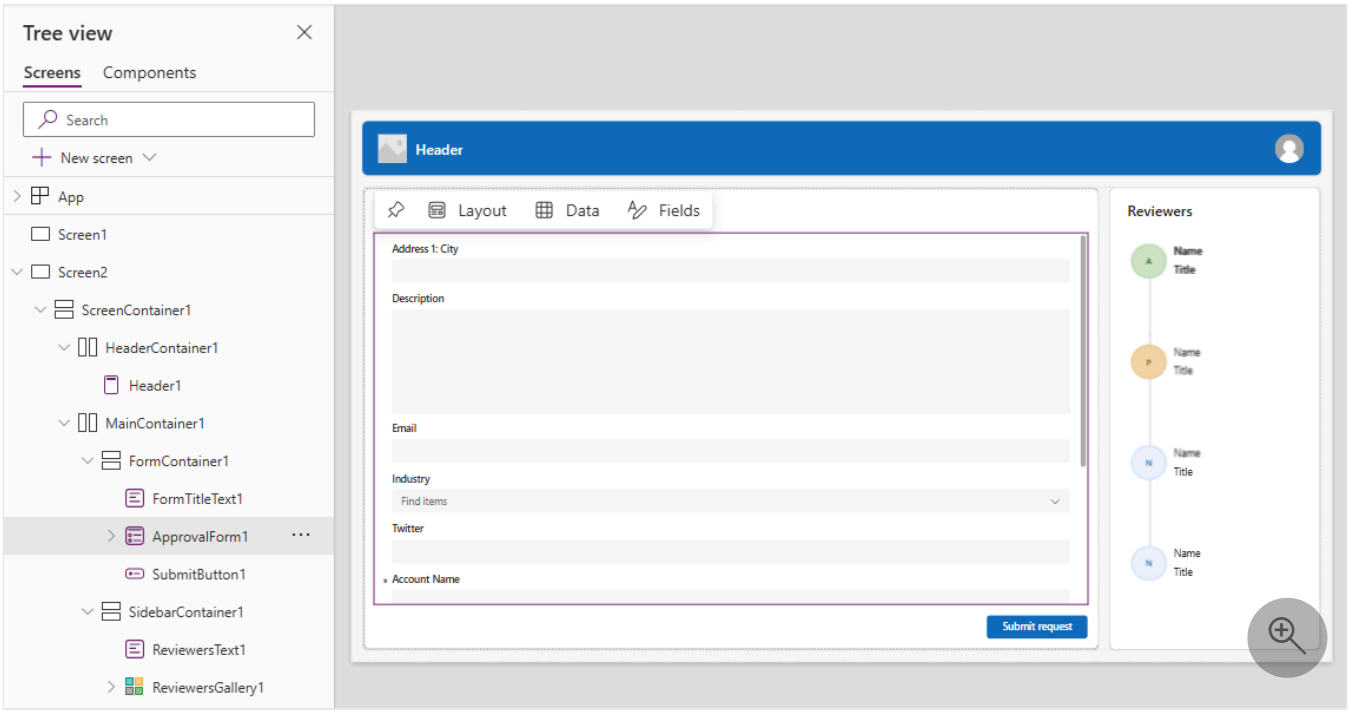
The **Approval request** screen has a header, a form with a submit button, and a gallery with predefined stages. The **Approval request** screen is useful for scenarios where actions are triggered by form submissions, such as submitting an approval request or displaying a workflow process for a business.

> **Tip**
>
> When you add an approval request screen, connect it to a data source. The data source determines the field names in the request form.

The **Approval request** screen has the following template for controls:

- Screen container

- Header container
  - Header
- Main container
  - Form container
    - Form title text
    - Approval form
    - Submit button
  - Sidebar container
    - Reviewers text
    - Reviewers gallery



Add and customize the Approval request screen

1. Select **New Screen** > **Approval request**.

2. In the tree view on **ApprovalForm**, connect to a data source such as Dataverse.

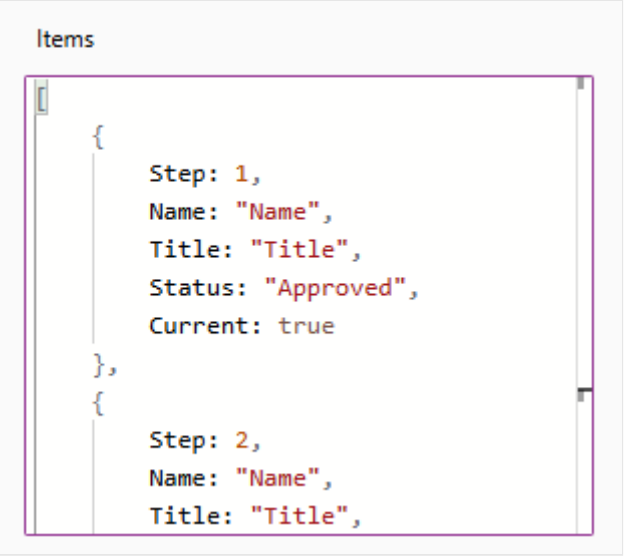
3. Choose a data source.

4. To view the details of the approval stages, select **ReviewersGallery** under the **SidebarContainer** node in the tree view. Then, in the properties pane, select the **Advanced** tab and go to **Items**.

   The approval stages have the following details:

   - **Name**: Name of the stage or approver
   - **Title**: Subtitle of the stage or approver
   - **Status**: Stage status

- **Current**: Whether this stage is the current stage of the approval request



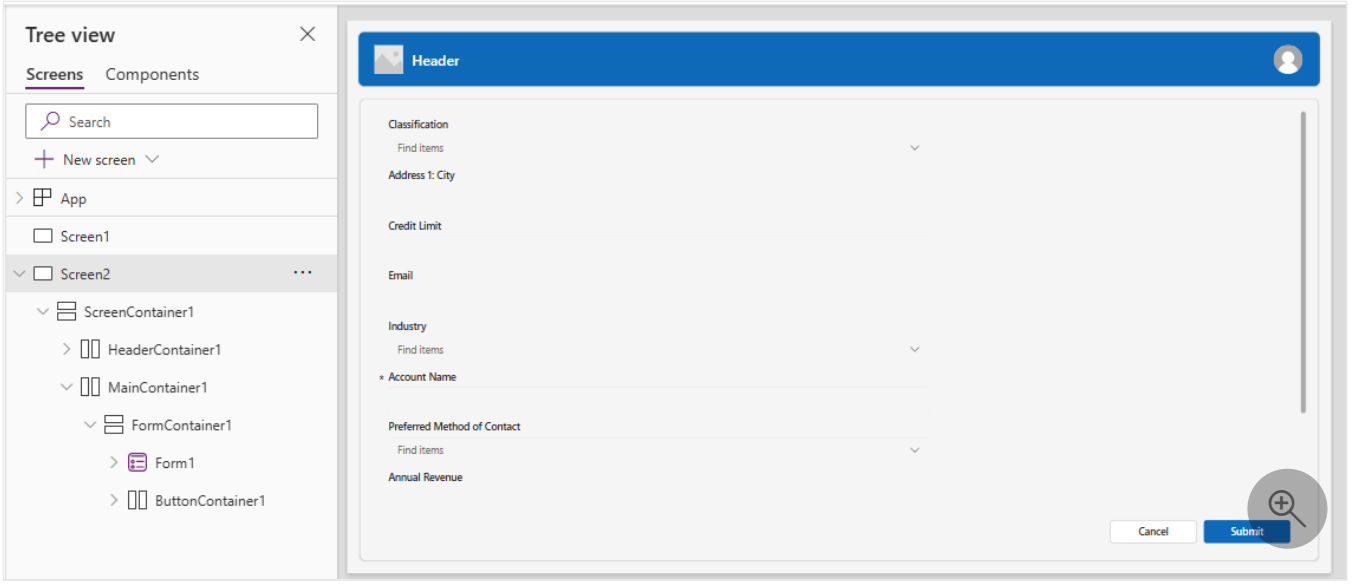
To notify the approver, add a Power Automate approval workflow in the button. For more information, see Create and test an approval workflow with Power Automate.

Header and form

The **Header and form** screen has a header, a form, and two buttons to submit the form or cancel form submission. This screen is great for using a full-screen form.

The **Header and form** screen has the following template for controls:

- Screen container
  - Header container
    - Header
  - Main container
    - Form container
      - Form
    - Button container
      - Cancel button
      - Submit button



Add and customize the Header and form screen

1. Select **New Screen** > **Header and form**.
2. In the tree view, you see **Form** selected and a prompt appears for you to connect the form to a data source.
3. Choose a data source.
4. Optionally, to ensure the best screen responsiveness, select each data card on the form and set its **Width Fit** property to **On** in the **Display** tab of the **Properties** pane.

Header and table

The **Header and table** screen has two controls: a header control and a table control. This template is great for showing a detailed data table on a screen.

The **Header and table** screen has the following controls:

- Header container
  - Header
- Main container
  - Table

Add and customize the Header and table screen

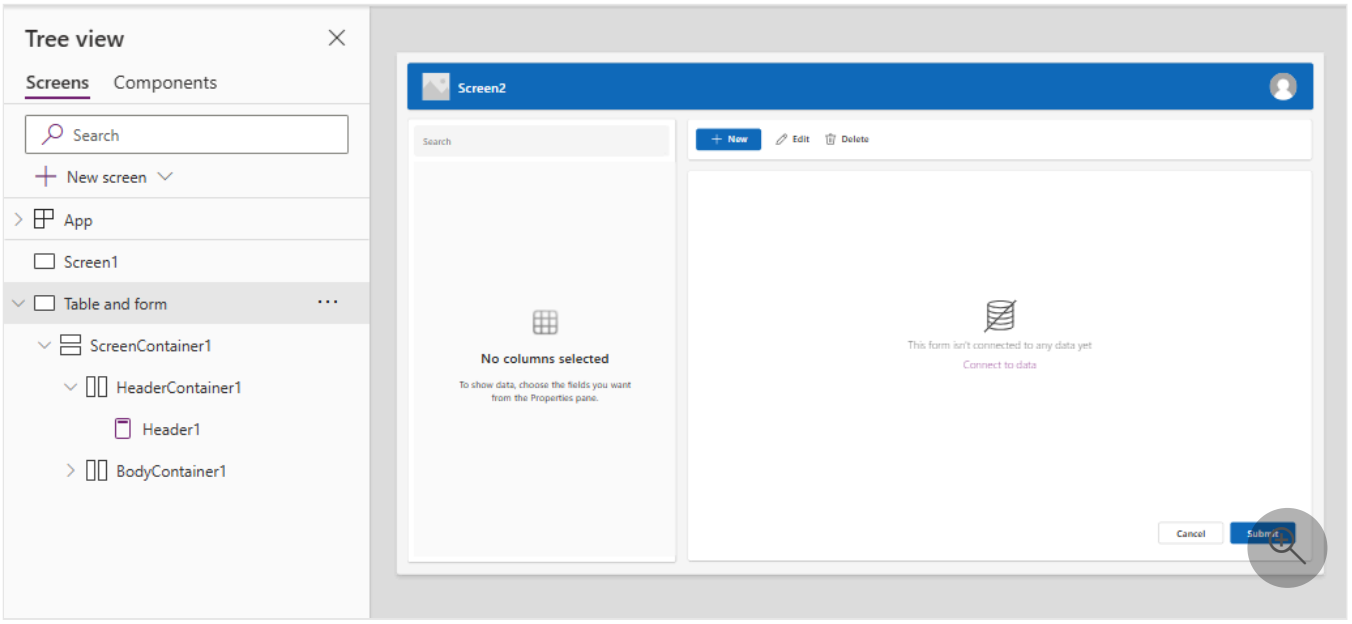
1. Select **New Screen** > **Header and table**.
2. In the tree view, select **Table** and connect it to a data source.

Table and form

The Table and form screen contains two main controls: the modern table control and the modern form control. This template has built-in Power Fx formulas to make data connection and customization easy. This template is ideal for selecting and editing a record by using a form.

The Table and form screen contains the following controls:

- Screen container
  - Header container
    - Header control
  - Body container
    - Text input (Search)
    - Table control
    - Form control
    - Button
      - New
      - Edit
      - Delete
      - Submit
      - Cancel



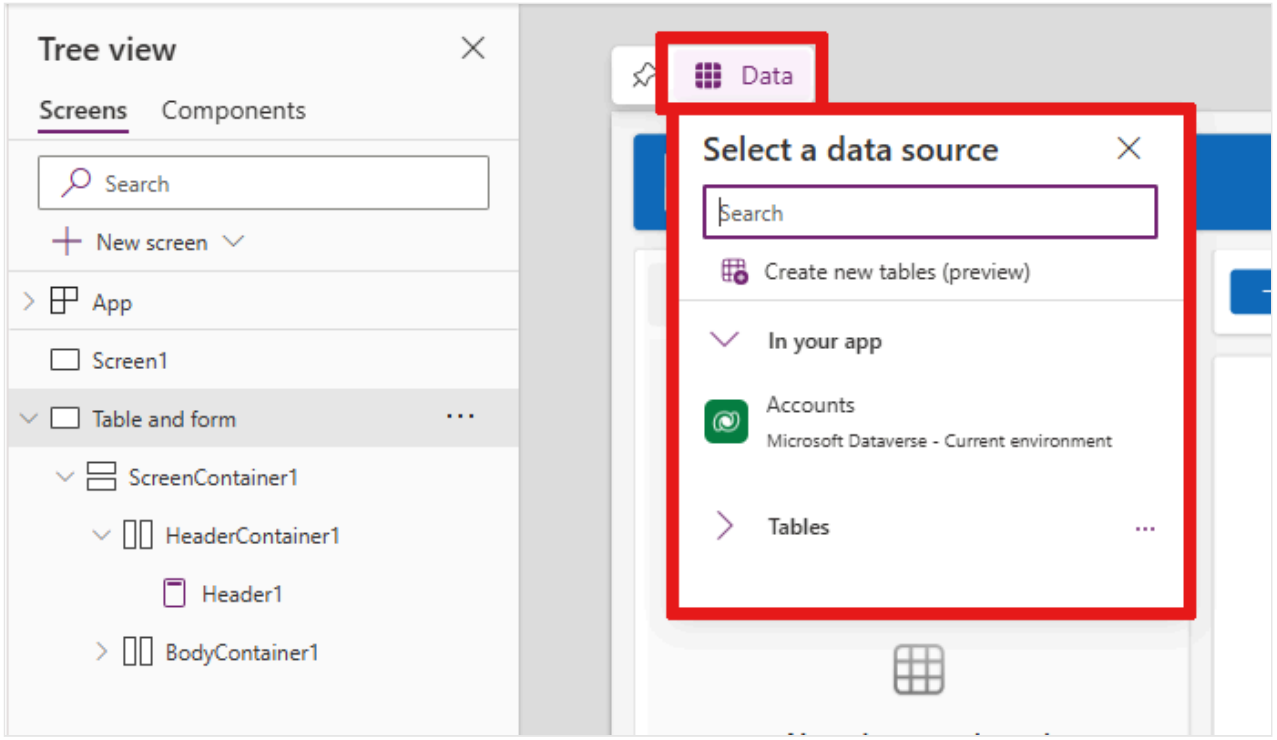
Add and customize Table and form screen

In Power Apps Studio:

1. Select **New Screen** > **Table and form**.

2. Select **Data** in the inline action bar.

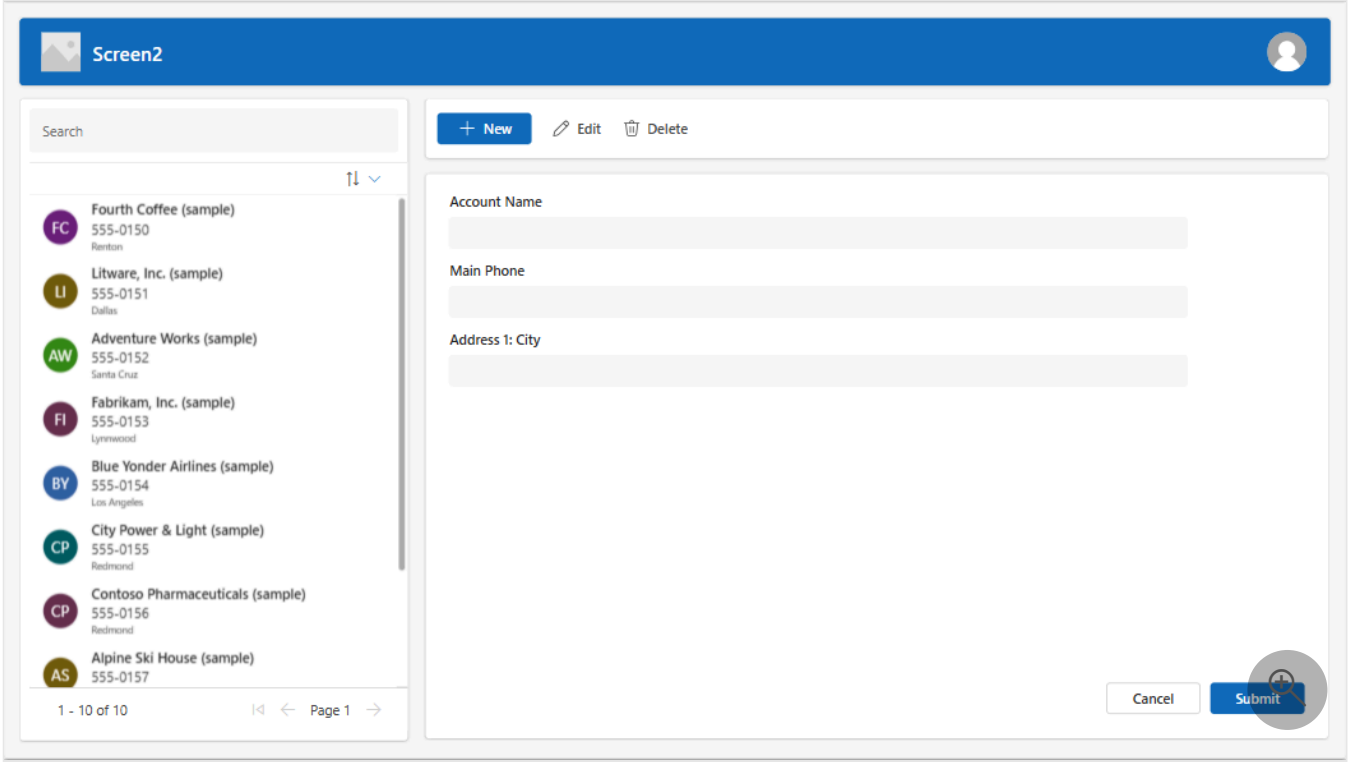
3. Select the data source you want.



When you connect, the table and form controls both connect to the data source.

You can select a record in the table control, and the form shows the details of the selected record.

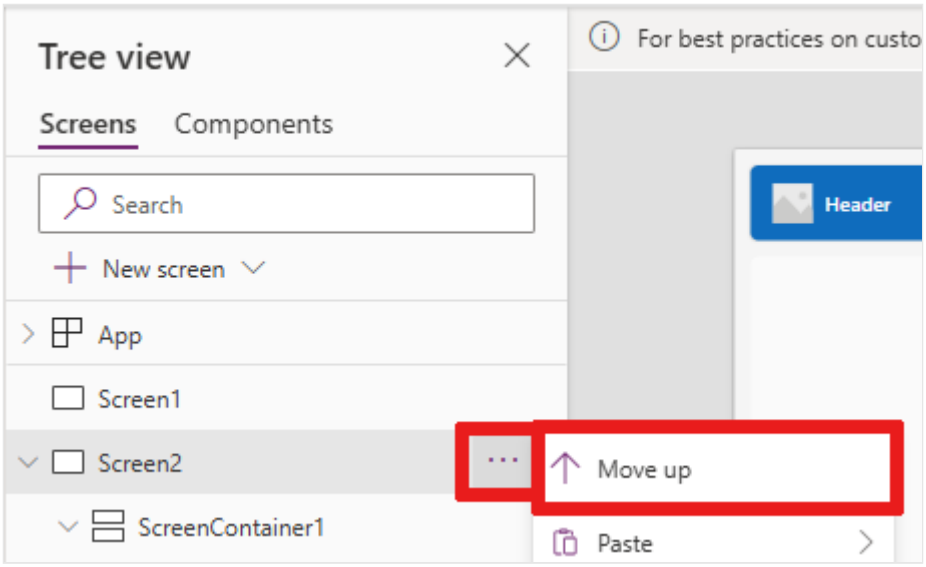
All buttons in this template have prebuilt Power Fx, so you don't need to customize them. When you connect the screen to the selected data source, the screen becomes fully functional.



Reorder screens

When your app has more than one screen, you can change their order in the tree view.

Select the overflow menu of a screen that you want to reorder, and then select **Move up** or **Move down**.



Use the StartScreen property to set the screen that displays first.

Add navigation

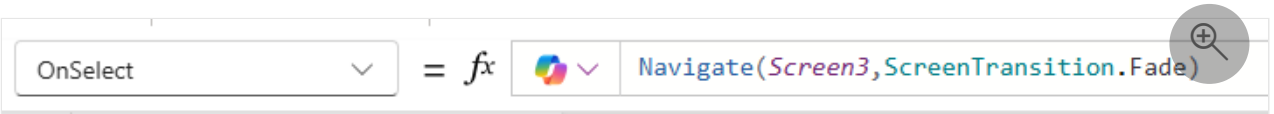
When your app has more than one screen, add navigation so your users can move between them.

1. Select the screen. Select **Insert**. In the search box, type **Next arrow** and then select it.

2. Move the arrow to where you want it to appear on the screen.

3. Select the arrow. Set the OnSelect property to the **Navigate** function. For example, use **Navigate(*Target*, Fade)**.

   Replace *Target* with the name of the screen where you want to navigate.

   In this example, when a user selects the arrow, the target screen fades in.

4. On the *Target* screen, add a **Back arrow** icon the same way. Set its OnSelect property, for example, `Navigate(Target, ScreenTransition.Fade)`.

   Replace *Target* with the name of the screen where you want to return.

Related information

Screen-control reference

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 01/14/2026

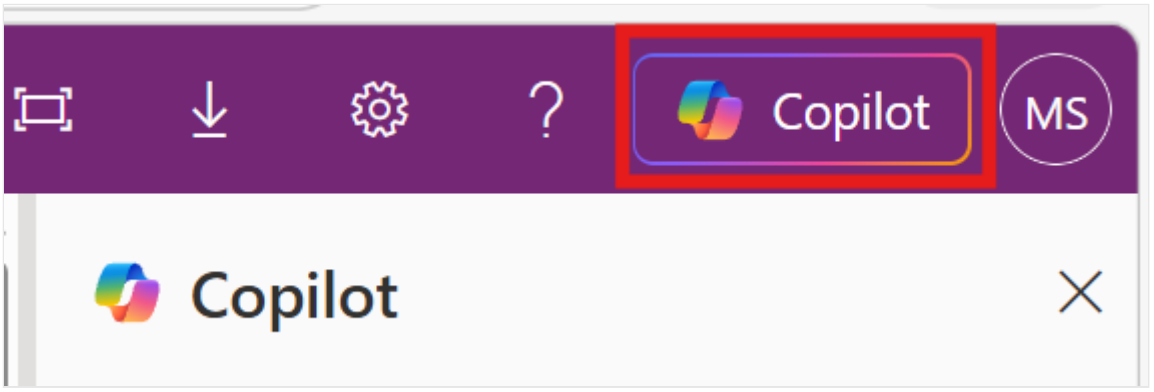
Add Microsoft 365 Copilot for app users in canvas apps (preview)

[This article is prerelease documentation and is subject to change.]

Microsoft 365 Copilot makes it easier for users to work with canvas apps by offering AI-powered insights through natural language conversations. By using this feature, users can quickly find information, navigate apps more easily, and get help to boost their productivity. App makers can enable Microsoft 365 Copilot to give users access to conversational AI that understands app data and provides helpful, contextual answers. Users can ask questions about their Microsoft Dataverse table or SharePoint list data in plain language and receive immediate, relevant responses.

This article shows you how to enable and configure Microsoft 365 Copilot for your canvas apps, both at the environment level and for individual apps.

When enabled, users can open Microsoft 365 Copilot in their canvas app by selecting **Copilot** on the upper-right corner. For more information, see Use Microsoft 365 Copilot in canvas apps.



Important

- This feature is in preview.
- Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use. They're available before an official release so that customers can get early access and provide feedback.
- This feature is in the process of rolling out and might not be available in your region yet.

Prerequisites

To use Microsoft 365 Copilot in canvas apps, make sure the following prerequisites are met:

- You must have access to an early release cycle environment to use this feature.
- Your tenant must be set to allow **Dataverse data available in Microsoft 365 Copilot**. For more information, see Enable Microsoft 365 admin center Copilot Dataverse settings.
- Dataverse Search must be set to **Default** or **On** for the environment. For more information, see Understand what this means for generative AI experiences.
- To use the Microsoft 365 Copilot feature in your app, users must have a Microsoft 365 Copilot license.

> **Note**
>
> - Microsoft 365 Copilot for canvas apps with Dataverse relies on Dataverse Search indexes. Enabling this feature might increase your environment's Dataverse capacity consumption. For more information, see What actions can makers or admins take to manage Dataverse Search efficiently.
> - Sufficient Dataverse database capacity must be available to support indexing, semantic search, and Copilot operations. For more information, see Capacity page details.

Enable Microsoft 365 Copilot for your environment

To manage Microsoft 365 Copilot for canvas apps, start by learning how to manage Microsoft 365 Copilot.

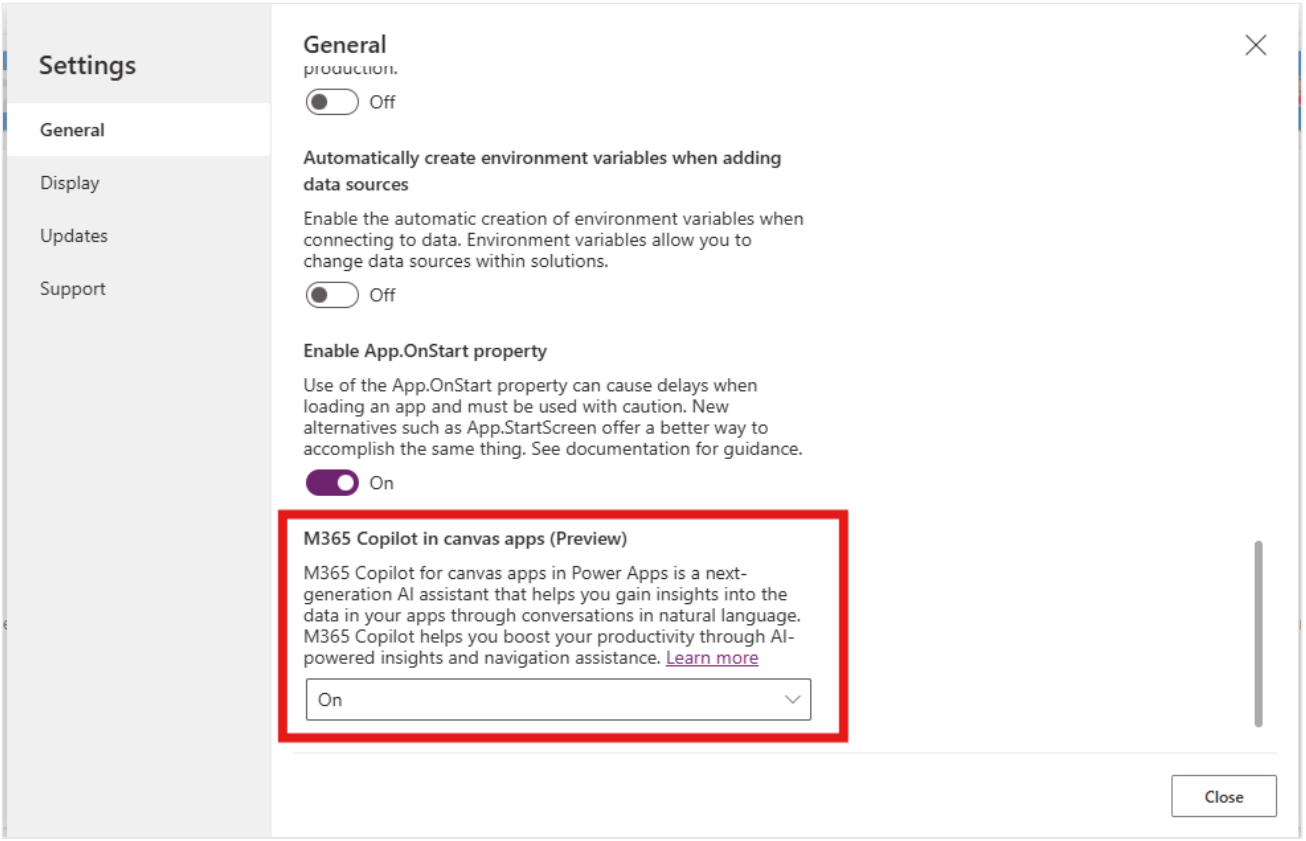
Power Platform administrators can set up and configure the Microsoft 365 Copilot feature for users in their environment.

1. Sign in to the Power Platform admin center.
2. In the navigation pane, go to **Copilot** > **Settings**. Under **Power Apps**, expand **Chat Agent** and choose **M365 Copilot**.
3. Select an environment group or an environment name, and then select **Edit Setting**.
4. Select **On** and then select **Save** to enable Microsoft 365 Copilot for your canvas apps in the selected environment group or environment.

Enable Microsoft 365 Copilot in a canvas app

Makers can enable or disable Microsoft 365 Copilot for a specific canvas app.

1. Sign in to Power Apps.
2. Open a canvas app for editing.
3. Select **Settings** in the command bar.
4. On the left, select **General**.
5. To enable Microsoft 365 Copilot, set **M365 Copilot in canvas apps (Preview)** to **On**. To disable it, set the option to **Off**.



6. Select **Save** and then publish the app for the changes to take effect.

Microsoft 365 Copilot vs. Copilot chat

Microsoft 365 Copilot is replacing Copilot chat in canvas apps.

Known limitations

- Agents you author can't yet use in-app user context to optimize their responses.

- M365 Copilot in canvas apps supports apps that use either SharePoint or Dataverse as a data source but not both within the same app.
- During the initial preview phase, for SharePoint, only applications containing a single SharePoint list will return answers to questions.
- If your canvas app connects to a SharePoint list through an environment variable, Microsoft 365 Copilot won't reference data from that SharePoint list.
- Microsoft 365 Copilot for canvas apps allows users to view data by using read-only operations. This capability means that users can only view data that matches their queries and can't make any changes. To make changes, customization with an agent is required.
- Microsoft 365 Copilot for canvas apps isn't available in the Power Apps mobile app.
- As this feature is being gradually deployed, certain settings in the Power Platform Admin Center might not be accessible yet, depending on the geographic location of your tenant.

Related information

- Customize Microsoft 365 Copilot with an agent
- Use Microsoft 365 Copilot in canvas apps

Last updated on 04/22/2026

Customize Microsoft 365 Copilot with an agent

Microsoft 365 Copilot in Power Apps enables users to interact with custom agents that extend Copilot's capabilities beyond standard functionality. By customizing agents, you can create tailored experiences that address your organization's specific business processes and data requirements. For more information, see Use agents in Microsoft 365 Copilot

This article describes the different types of agents you can build and integrate with Microsoft 365 Copilot in Power Apps. You'll learn about the capabilities and use cases for each agent type, helping you choose the right approach for your customization needs.

You can customize agents in several ways to fit your organization's needs.

Expand table

Agent Type	Description	Common use cases
Declarative Agent	Configured using low-code tools and templates, allowing quick setup for common scenarios within Microsoft 365 Copilot.	Automating routine tasks, answering FAQs, and providing guided workflows.
Custom Engine Agent	Developed with custom logic and integrations, offering advanced capabilities tailored to specific business needs.	Complex business processes, custom data integrations, and specialized automation.
Copilot Studio Agent	Built and managed in Microsoft Copilot Studio, enabling integration with Teams and Microsoft 365 for conversational experiences.	Interactive chatbots, team collaboration, and personalized support within Microsoft 365 apps.

Use the following information to find the best solution for your organization:

- Microsoft 365 Copilot Agents Overview documentation provides guidance on how to choose what type of agent to build.
- Create agents as Declarative Agents for Microsoft 365 Copilot or as Custom engine agents for Microsoft 365.
- Use Microsoft Copilot Studio to connect and configure an agent for Teams and Microsoft 365, making it available in Microsoft 365 Copilot.
- Review guidance on how to Choose between Microsoft 365 Copilot and Copilot Studio to build your agent.

Limitation

- Agents you author can't yet use in-app user context to optimize their responses.

Related information

- Add Microsoft 365 Copilot for app users in model-driven apps
- Use Microsoft 365 Copilot in model-driven apps
- Add Microsoft 365 Copilot for app users in canvas apps (preview)
- Use Microsoft 365 Copilot in canvas apps (preview)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 04/15/2026

Add a Copilot control to a canvas app (preview)

Summarize this article for me

[This article is prerelease documentation and is subject to change.]

The Copilot control is an AI assistant that you can add to your canvas apps. By using Copilot, app users can get insights about the data in your app through natural language conversations with an AI-powered copilot. Makers can add the control to any canvas app and choose which data provide answers.

> **Important**
>
> Starting February 2, 2026, you can't add the Copilot control to new canvas apps. Existing apps with this control will remain functional for a limited time but will eventually no longer be supported. Microsoft 365 Copilot Chat in canvas apps is the new the recommended solution for natural language interactions in canvas apps. To ensure a smooth transition and prevent any disruption, we recommend that you migrate to Microsoft 365 Copilot chat as soon as it becomes available in your environment.

> **Note**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is powered by Azure OpenAI Service.

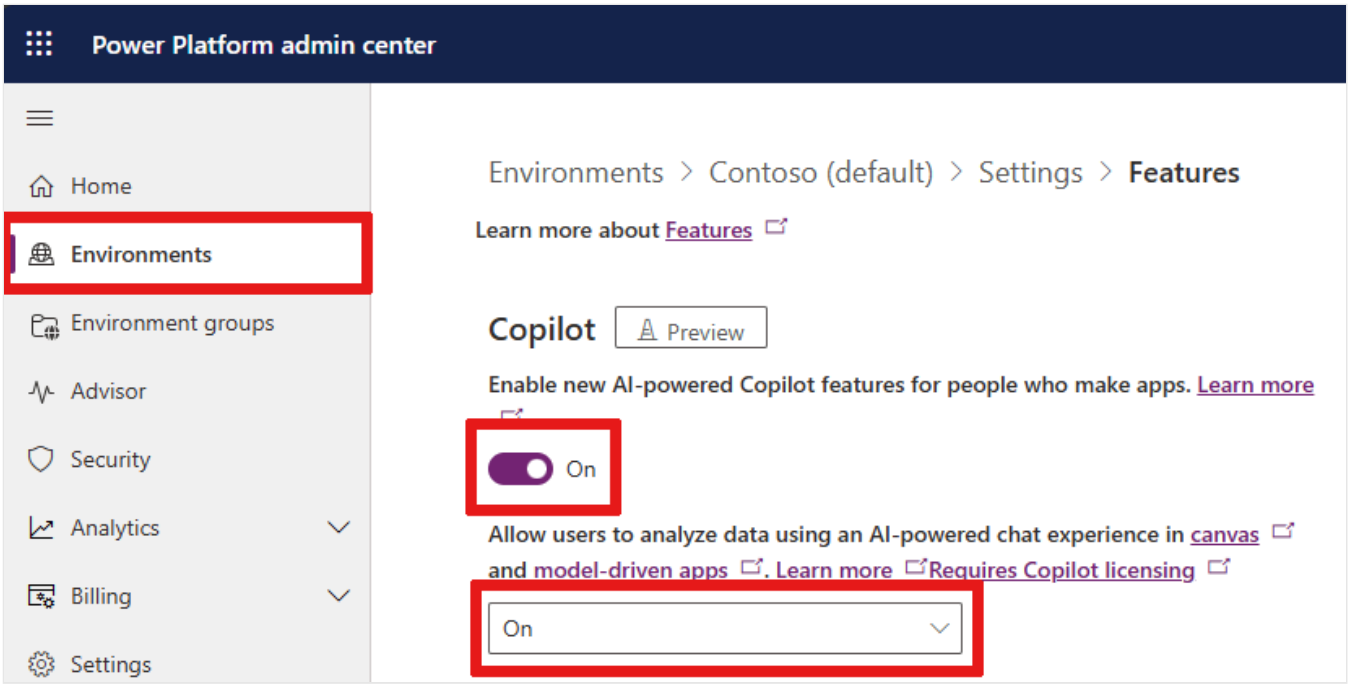
Prerequisites

Ensure you meet the prerequisites and region availability in Copilot in Power Apps overview (preview).

Copilot doesn't work in environments that use a customer-managed key or Customer Lockbox.

Set up Copilot for your environment

To let app users use the Copilot chat experience in a canvas app, a Power Platform admin needs to enable **Allow users to analyze data using an AI-powered chat experience in canvas and model-driven apps** in the Power Platform admin center. Learn more in Manage feature settings.



Set up Copilot for your canvas app

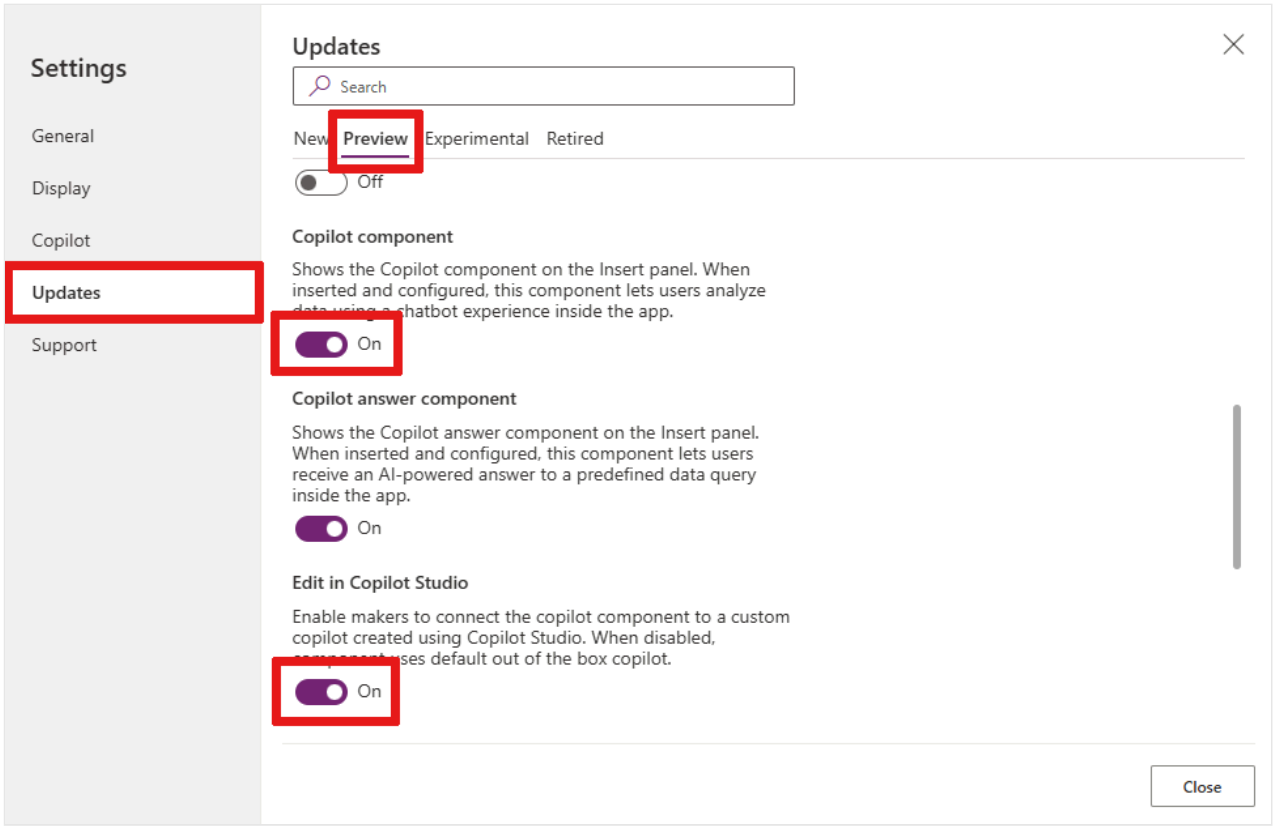
To add a Copilot control to your canvas app, turn on **Copilot component** and **Edit in Copilot Studio** in the app settings in Power Apps.

Copilot component: Turns on the Copilot feature in your canvas app so you can add the Copilot control and connect it to a copilot.

Edit in Copilot Studio: Lets you customize the copilot using Copilot Studio. If you turn off this option, only the default copilot is available.

1. Open your canvas app for editing in Power Apps Studio. On the command bar, select **Settings** > **Updates**.

2. On the **Preview** tab, turn on the **Copilot component** and **Edit in Copilot Studio** settings.

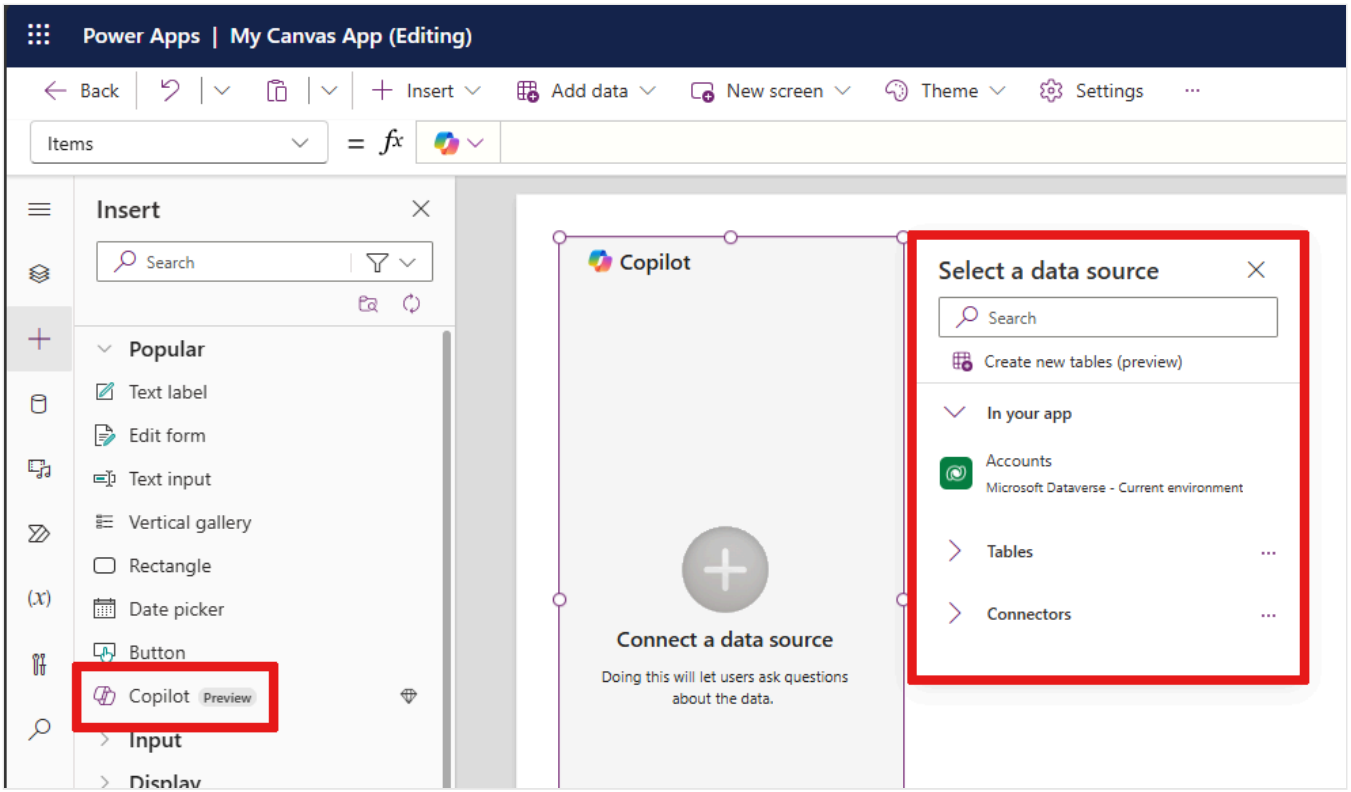


Add the Copilot control to your canvas app

With your canvas app open for editing, and all settings configured:

1. In the app authoring menu, select **Insert**, and then select **Copilot (preview)**.

2. When prompted to add a data source to Copilot, select a Dataverse table as the data source.



> **Note**
>
> The Copilot control only supports Dataverse tables for the data source.

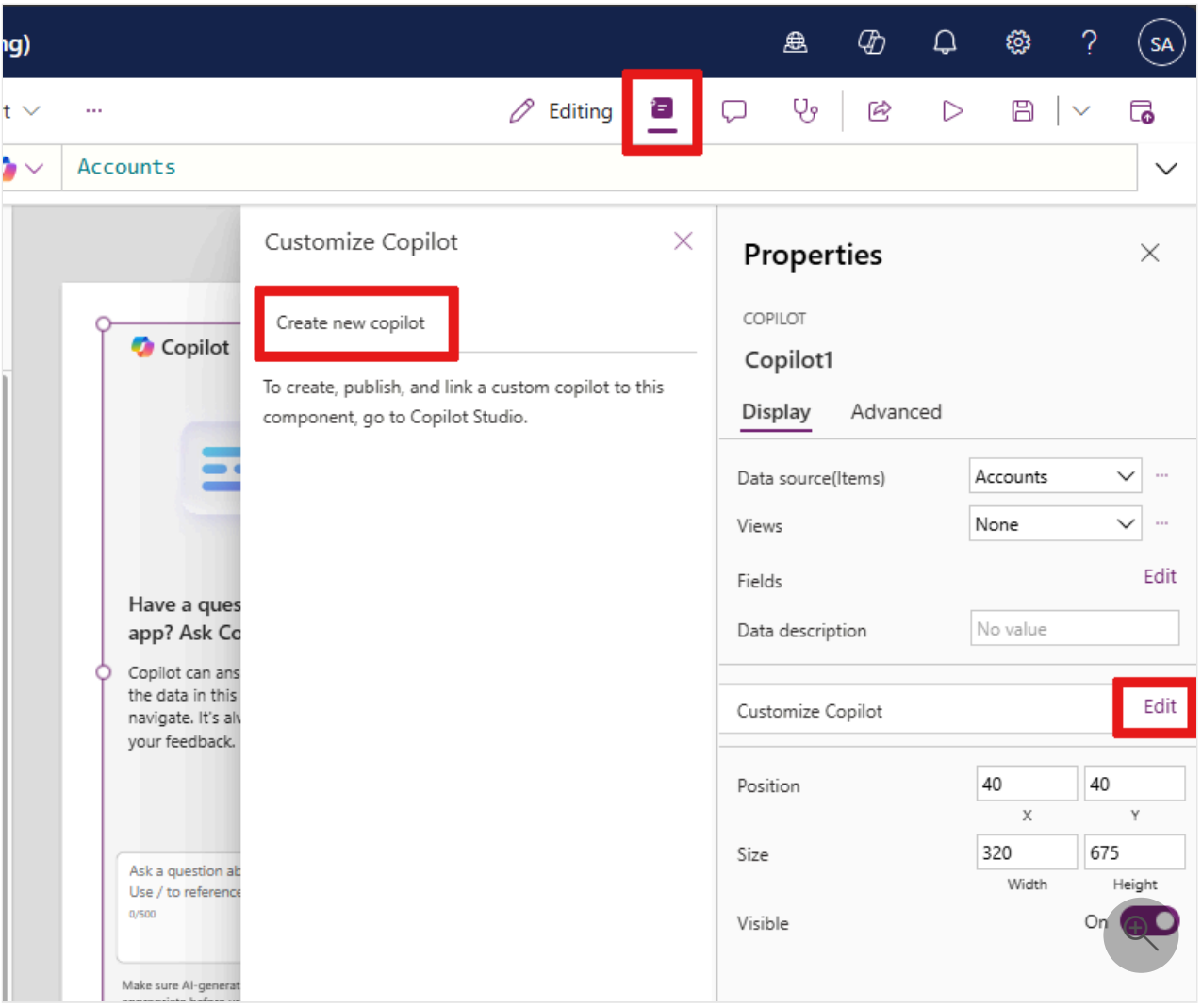
Customize the copilot by using Copilot Studio

Copilot Studio is an app that you use to create and edit copilots for your apps. Define your copilot's articles, actions, and other features. For example, make your copilot respond to specific questions about your app's data or perform actions like opening a screen or sending an email.

Customize your newly connected copilot in Power Apps through the properties menu.

1. Select the Copilot control on your canvas. Select **Edit** next to the **Customize copilot** field in **Properties**.

2. If you don't have a copilot, select **Create new copilot** in the **Customize Copilot** pane. The Copilot control in Power Apps Studio doesn't support enabling an existing Copilot from Copilot Studio.



Copilot Studio opens in a new tab. Any changes you make in Copilot Studio appear in your connected copilot in your canvas app.

Learn more in Quickstart: Create and deploy a copilot.

Collect feedback from app users

Makers and app users provide feedback on copilot responses by selecting the **Like** (thumbs up) or **Dislike** (thumbs down) button for each response. Enter detailed feedback in the text box, and then select **Submit**.

Feedback goes to Microsoft to help improve the Copilot control.

Disallow feedback from app users

If you don't want your app users to provide feedback to Microsoft, turn off the feedback option.

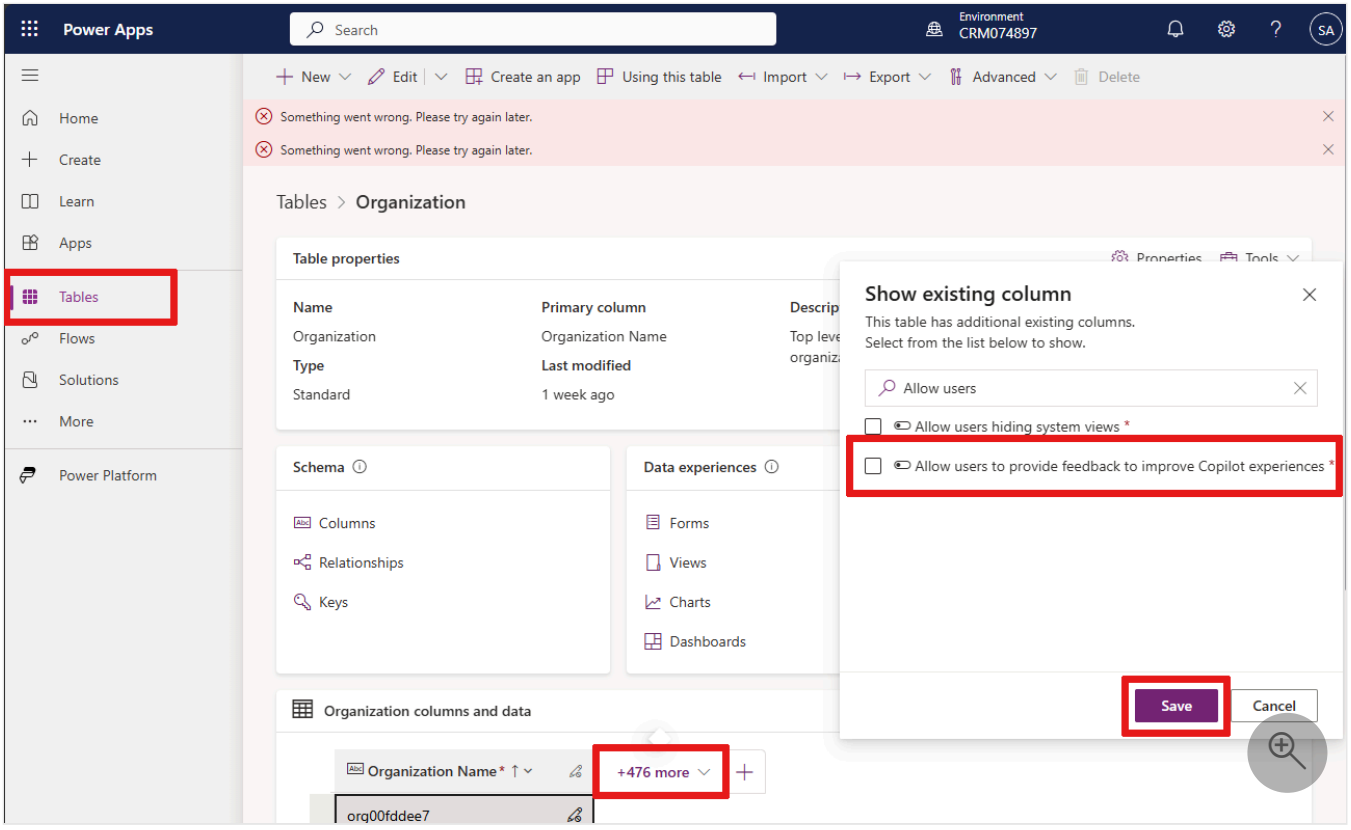
1. Sign in to Power Apps, and select **Tables** from the left navigation pane.

2. Select the **Organization** table from the list.

3. In the **Organization columns and data** section, select the column headers list. The **Show existing column** popup appears.

4. Search for **Allow users to provide feedback for App Copilot**, and make sure the check box is unchecked.

5. Select **Save**.



Related information

- Build apps through conversation (preview)
- Use your prompt in Power Automate
- Add Copilot for app users in model-driven apps
- Power Apps Copilot control (video)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 02/24/2026

Add a custom Copilot to a canvas app (preview)

Summarize this article for me

[This article is prerelease documentation and is subject to change.]

You can integrate a custom Copilot created in Microsoft Copilot Studio and enable it for your canvas app. This lets users interact with Copilot to ask questions about the data in your app. With just a few simple steps, you can embed a custom Copilot across all your canvas app screens without changing the app's design.

> **Important**
>
> Starting February 2, 2026, adding a custom Copilot to new canvas apps will be discontinued. Existing apps using this feature will remain functional for a limited time but will eventually no longer be supported. Microsoft 365 Copilot Chat in canvas apps is the new the recommended solution for natural language interactions in canvas apps. To ensure a smooth transition and prevent any disruption, we recommend that you migrate to Microsoft 365 Copilot chat as soon as it becomes available in your environment.

> **Note**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - App Copilot for canvas apps is currently rolling out and might not yet be available in your region. App Copilot may be available on Power Apps mobile for mobile devices before it becomes available on the web browser during rollout.

Availability

- This capability might not be available in your region yet. Learn more in Explore Copilot features by geography and languages.
- During the rollout, this capability might be accessible on the Power Apps mobile app before it becomes available on web browsers.

- To use copilots in Power Apps, you must enable data movement across regions for generative AI features. This is crucial if your organization and environment are located in different regions. For more information, see Enable copilots and generative AI features.

Prerequisites

Before using this feature, make sure the following prerequisites are met:

- Create a custom Copilot in Copilot Studio and publish it in the same environment as your canvas app.
- Configure the custom Copilot with user authentication in Microsoft Entra ID.
- Give makers and users access to the Copilot within the environment.

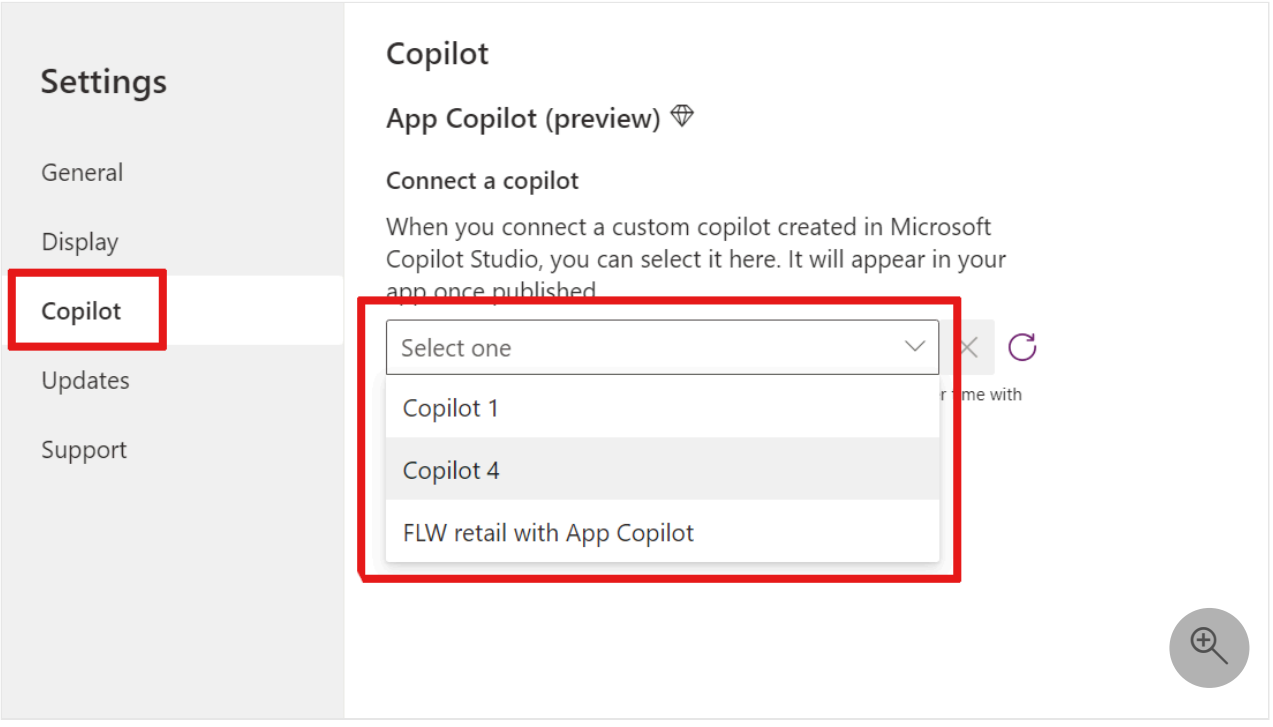
Connect Copilot to your app

Open your canvas app for editing in Power Apps Studio:

1. In the command bar, go to **Settings** > **Updates** > **Preview** tab, and set the toggle for **App Copilot** to **On**.

   You see a **Copilot** tab appear in **Settings**.

2. Select the **Copilot** tab. In the drop-down list under **Connect a copilot**, select a custom copilot published and shared in the same environment as your canvas app.



3. Once you select a Copilot, close the settings dialog box and publish the app. Once the app is published, users are able to use Copilot within it.

Copilot isn't visible when previewing an app in Power Apps Studio. To see Copilot in action, save and publish your app, and then open it using the Power Apps mobile app or a web browser.

> **Tip**
>
> Removing the custom Copilot from the app will disable its functionality.

Use Copilot in your app

- Open your canvas app in a web browser. On the command bar, select **Copilot**.



- In the Power Apps mobile app on iOS or Android devices, select the floating button to access Copilot. You can reposition the button or dismiss it. To use Copilot again after dismissing it, close and reopen the app.

Difference between this feature and the Copilot control

This feature lets you add a custom Copilot created in Microsoft Copilot Studio to a canvas app. It doesn't require any changes to the app's layout. It's available both on the web and natively on mobile devices, and it can connect to existing custom Copilots created in Copilot Studio.

When you add a Copilot control to a canvas app, makers need to update the app's layout to accommodate the control, which is only available on the web. This control supports asking questions about your data or connecting to a new custom copilot from Copilot Studio.

See Also

FAQ for Copilot

Last updated on 02/24/2026

Create and edit tables using the table designer

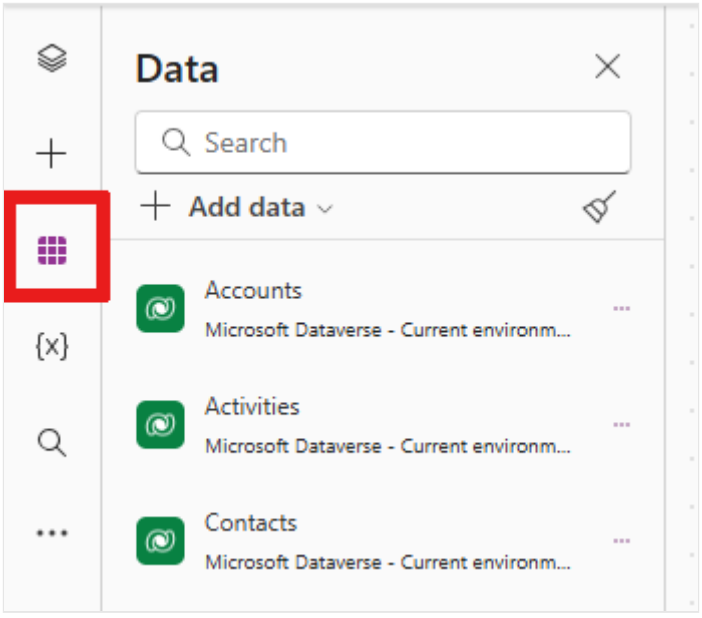
Article • 10/29/2024

Create and edit Dataverse tables in canvas apps using the table designer inside Power Apps Studio. You can also work with the data that's in the table without leaving the app that you're working on.

Create a table

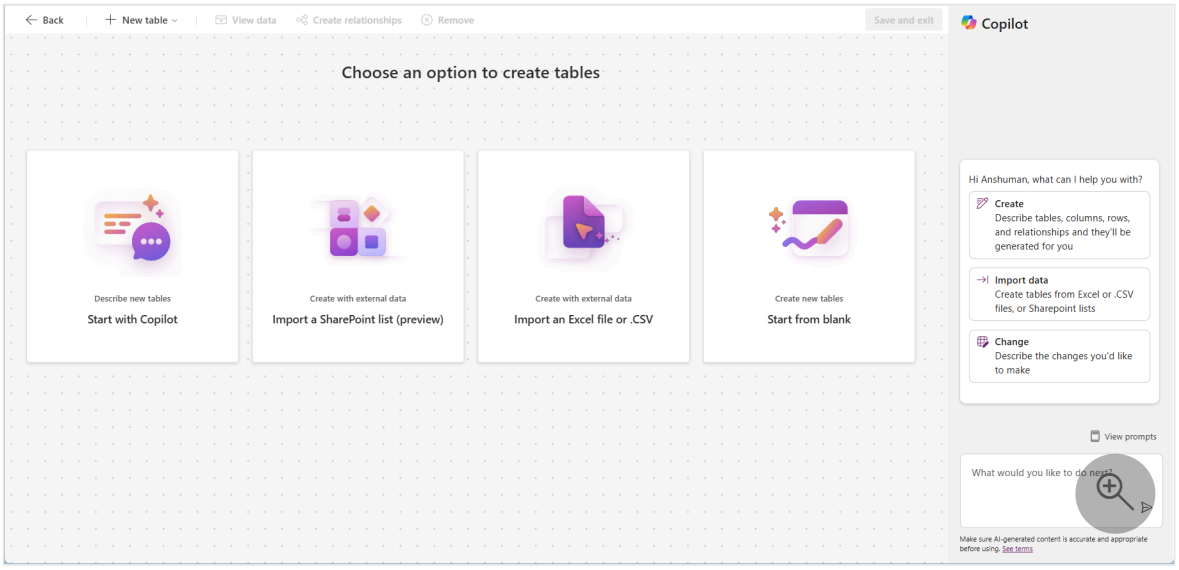
1. Create a new or open an existing app in Power Apps Studio.

2. In the app authoring menu, select **Data**.



3. On the **Data** pane, select **Add Data** > **Create new table**.

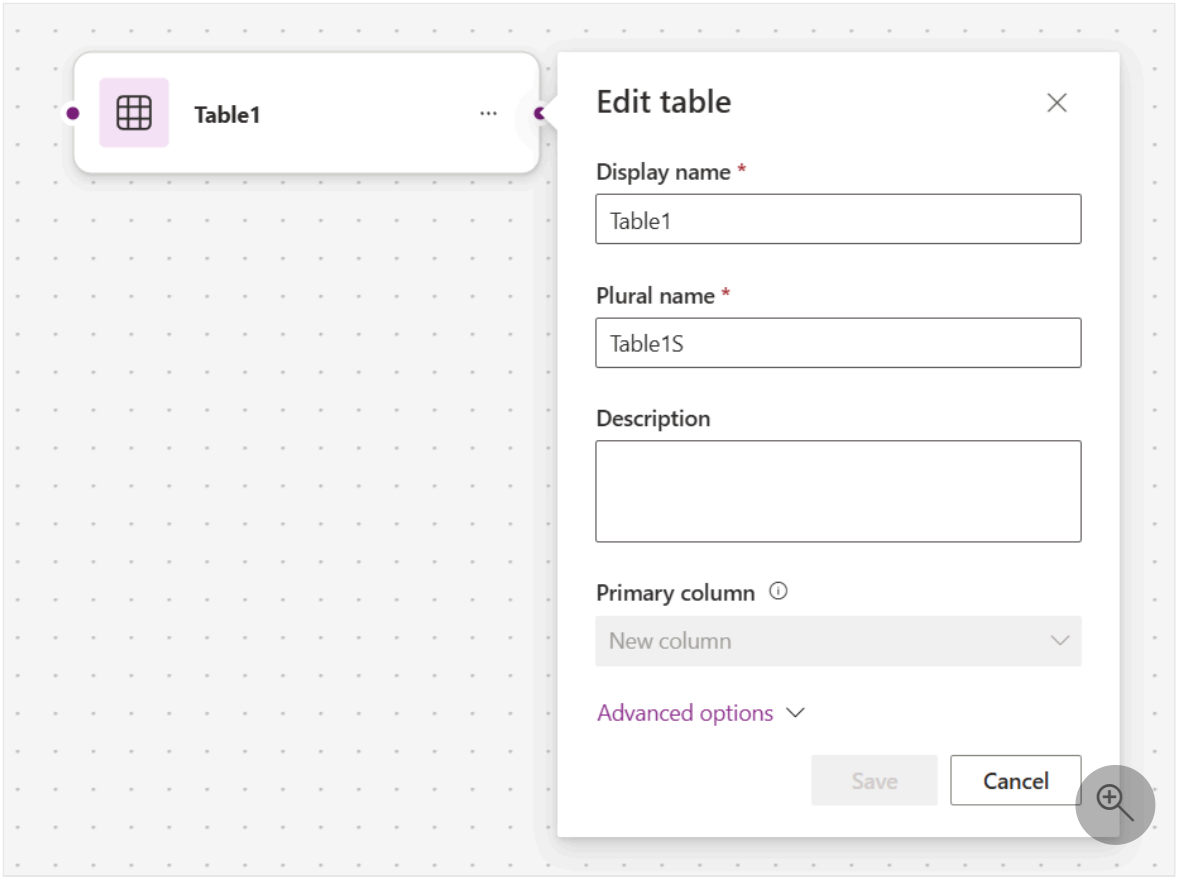
> ⓘ Note
>
> If the **Create new table** button is disabled then review Required permissions and limitations.



4. You see **Choose an option to create tables**. For example, you can choose to **Start from blank** or **Start with Copilot** to create a new table.

5. A new table is created for you in the canvas. Enter a name for the table directly where you see the table name.

   You can also expand the **Properties** pane found in the table's overflow menu (the ellipses) to edit the table properties, such as the plural name for the table. For example, a table name can be Ticket, and the plural table name can be Tickets.



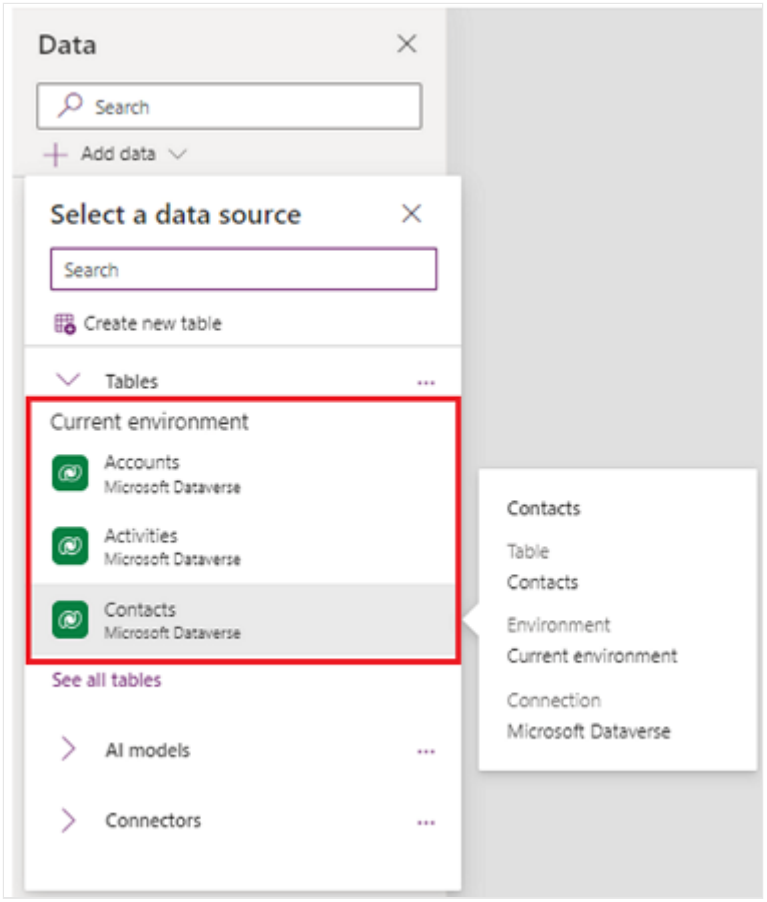
   Can you also add columns or sample data to the table.

6. When you're done editing, select **Save** and exit.

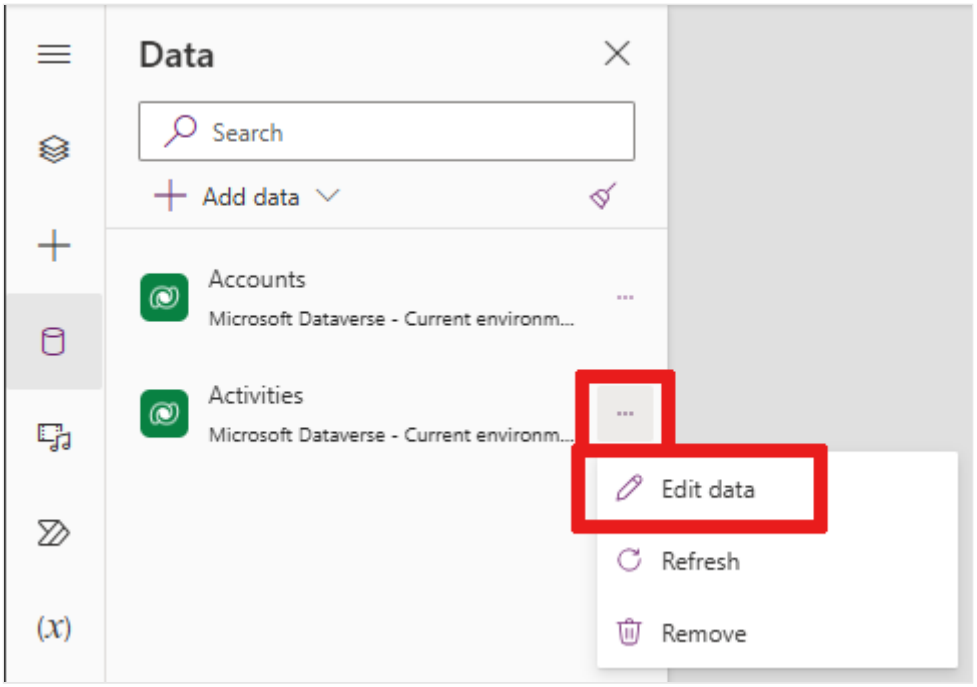
Edit an existing table

Once a Dataverse table is created, you can start editing the table. You can also add existing Dataverse tables to your app and edit those tables using the table designer.

1. From the app authoring menu, select **Data**.
2. On the **Data** pane, select **Add data**.
3. In the **Select a data source** dialog box, from the list of Dataverse tables, select a table to add to your app.



4. On the newly added table select, **More actions** > **Edit data**.



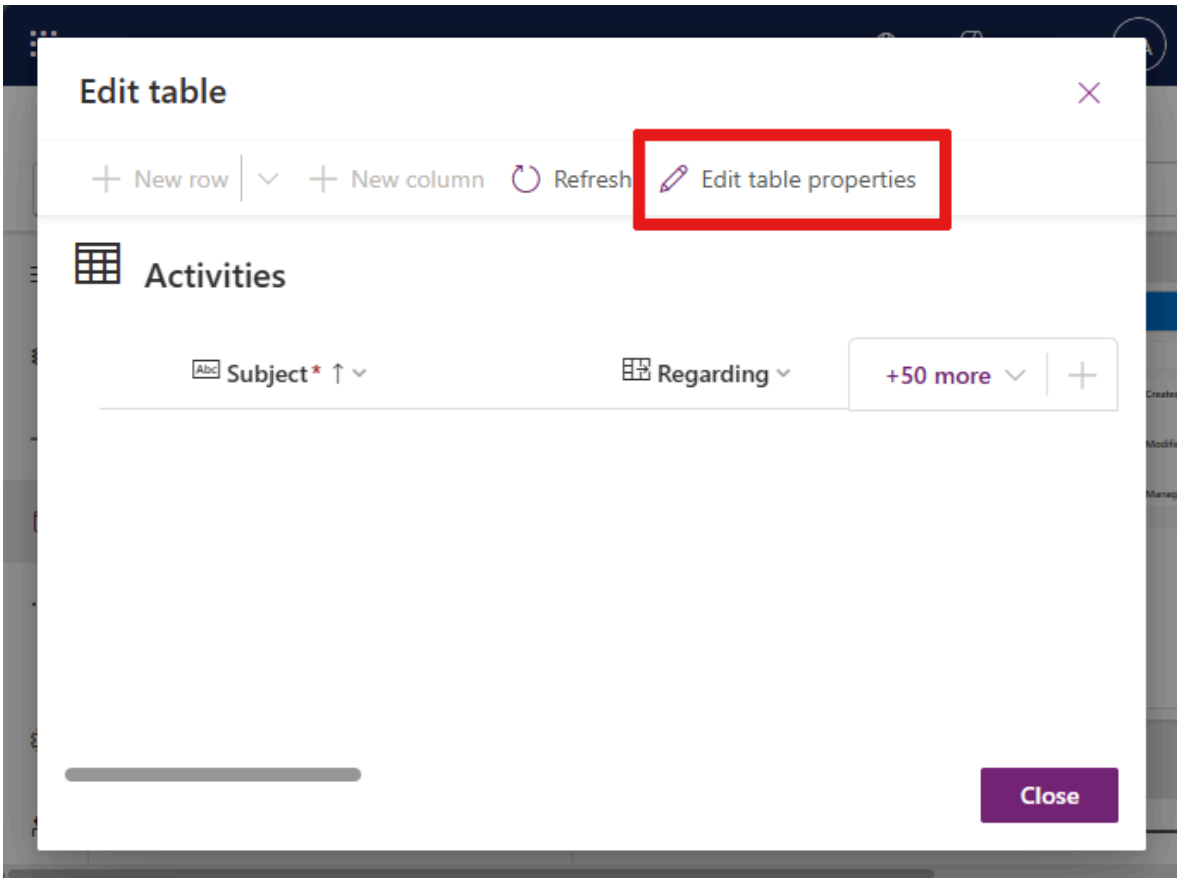
5. The selected table is ready for editing in the table designer.

> **Note**
>
> Some system tables might not be available for editing, or some editing actions may be disabled.

Edit the table properties

After selecting **Edit data**, you see the **Edit table** designer.

1. Select **Edit table properties**.



You can edit the following table properties.

Expand table

Property	Description
Display name	The singular name for the table shown in the app. The display name can be changed later.
Plural display name	The plural name for the table shown in the app. The plural display name can be changed later.
Description	Provide a meaningful description of the purpose of the table.

Select **Enable Attachments** to append notes and files to records for this table.

Select the **Primary column** tab if you want to change its **Display Name** or **Description**. The primary column is used by lookup fields when establishing relationships with other tables.

Advanced options

Select **Advanced options** to see the optional properties you can set. Otherwise some are already set for you.

Expand table

Property	Description
Schema name	By default, the schema name is automatically created for you based on the display name, but you can change it. The schema name can't contain spaces and includes the customization prefix for the Dataverse solution publisher. You can't change the schema name after the table is saved.
Logical name	A unique identifier for the entity. This name can't be changed once the entity is created.
Type	Select the type of table. Use standard for most tables. Activity tables are special tables that can only be owned by a user or team, but can't be owned by an organization. Virtual tables require the table be populated with data from an external source.
Record ownership	Switch the table type to Activity table, so you can create tables that can manage tasks. The type of **Ownership** defines who can perform operations on a record.
Color	You can select your table color using words or hex codes.
Primary row image	You can choose whether to display an image for the table. This image is displayed in Power Apps in some design areas. Notice that the image doesn't appear in apps using the table. To display images in apps, use the image column. Learn more in Image columns.
Apply duplicate detection rules	If duplicate detection is enabled for your organization, enabling allows you to create duplicate detection rules for this table.
Track changes	Enables data synchronization in a performant way by detecting what data changed since the data was initially extracted or last synchronized.
Provide custom help	When selected, set a **Help URL** to control what page users see when they select the help button in the application. Use help to provide guidance specific to your company processes for the table.
Audit changes to its data	When auditing is enabled for your organization, table records can change to be captured over time. When you enable auditing for a table, auditing is also enabled on all its fields. You can select or clear fields that you want to enable auditing on.
Leverage quick create form if available	After you create and publish a Quick Create Form for this table, people can create a new record using the Create button in the navigation pane. Learn more in Create and design model-driven app forms. If enabled for a custom activity table, the custom activity is visible in the group of activity entities when people use the **Create** button in the navigation pane. However, because activities don't support quick create forms, the main form is used when the custom table icon is selected.

Property	Description
Enable long term retention	Table can be included in the configuration of data retention, which moves certain records to long term storage.
Creating a new activity	Associate activities to records for this table.
Doing a mail merge	App users can use this table with mail merge.
Setting up SharePoint document management	After other tasks are performed to enable document management for your organization, enabling this feature allows for this table to participate in integration with SharePoint.
Can have connections	Use the connections feature to show how records for this table have connections to records of other tables that also have connections enabled.
Can have a contact email	Send emails using an email address stored in one of the fields for this table. If a **Single Line of Text** column with format set to email doesn't already exist for this table, a new one is created when you enable sending email.
Have an access team	Create team templates for this table.
Can be linked to feedback	Let app users write feedback for any table record, or rate table records within a defined rating range. Learn more in Configure a table for feedback/ratings.
Appear in search results	Enable so that table records can be included in search results when using an app.
Can be taken offline	Makes data in this table available while the Power Apps application isn't connected to a network.
Can be added to a queue	Use the table with queues. Queues improve routing and sharing of work by making records for this table available in a central place that everyone can access.

Add and edit columns

You can create and edit columns in the table editor.

Learn more in How to create and edit columns.

Editing other table properties

For more advanced table editing scenarios, using the table hub is required. Some examples of these kinds of edits are table relationships, keys, and deleting the table.

Learn more in Create and edit tables using Power Apps.

Add and edit data

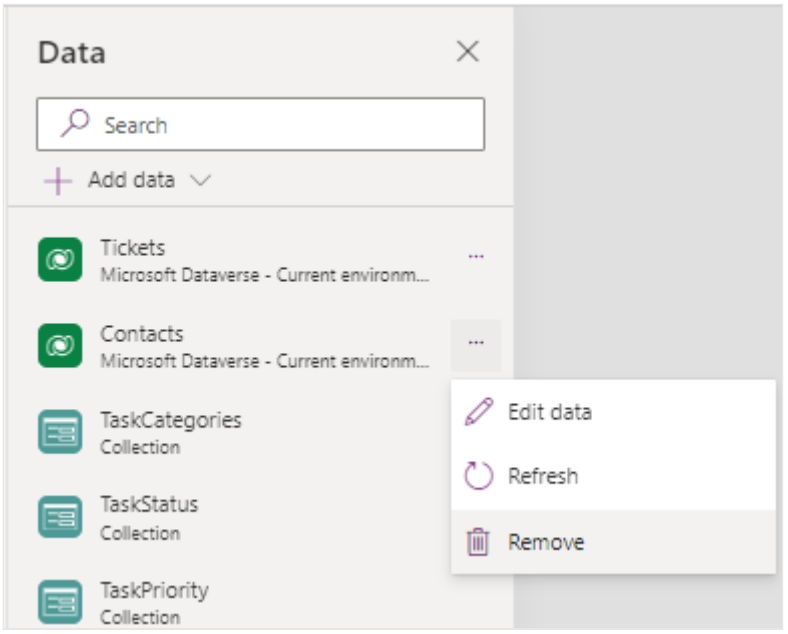
The table designer for canvas apps allows you to add new rows to a Dataverse table or edit rows inside Power Apps Studio using the table designer. You can start typing directly into the grid or select, **New row**.

Learn more in Create your first table.

Remove a table

To remove a table from your app, select **More options** to the right of the table name then select, **Remove**.

> **Note**
>
> When you remove a table, the table is removed and not deleted.



Required permissions and limitations

- To create a table in Dataverse, the environment must have a Dataverse database provisioned. You must have system customizer or system administrator

permissions for the environment where you'd like to build a table. If these conditions aren't met, the **Create New Table** button is disabled.

- You can only create and edit tables in the current Dataverse environment. If you changed your Dataverse environment using the environment picker, then the **Create new table** and **Edit table** options is disabled.

Feedback

Was this page helpful? Yes No

Provide product feedback

Show a list of items in canvas apps

05/31/2025

Show a list of items from any data source by adding a **Gallery** control to your canvas app. This topic uses Excel as the data source. Filter the list by configuring the **Gallery** control to show only those items that match the filter criterion in a **Text input** control.

Prerequisites

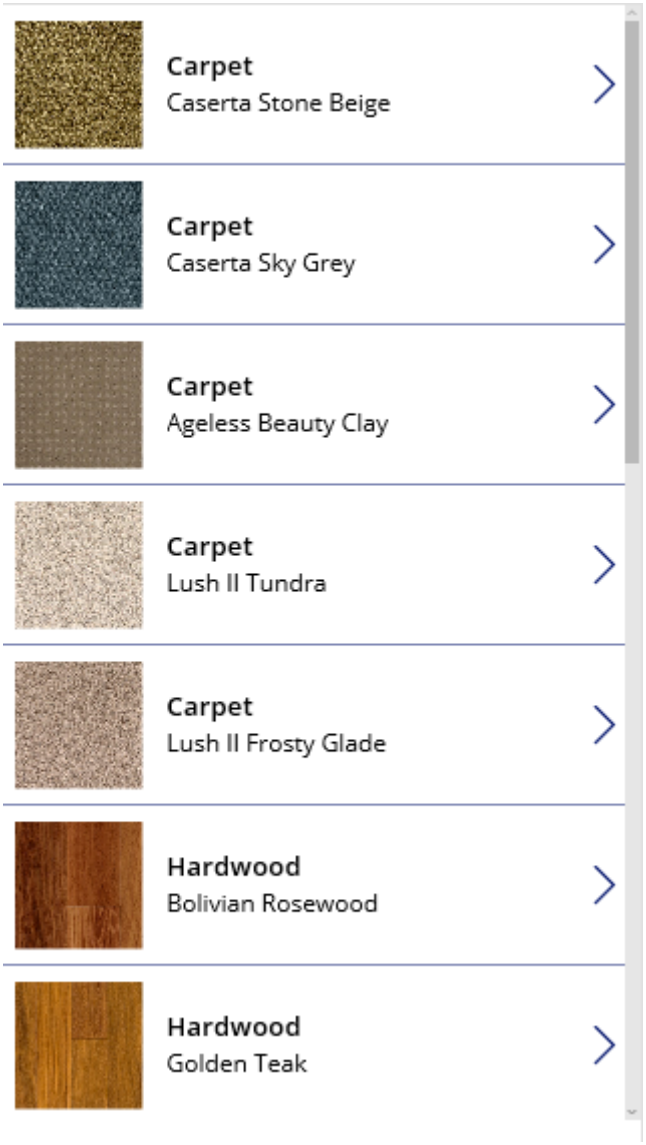
- Learn how to add and configure a control in Power Apps.
- Set up the sample data:
    1. Download this Excel file, which contains sample data for this tutorial.
    2. Upload the Excel file to a cloud-storage account, such as OneDrive for Business.
- Open a blank app:
    1. Create a blank canvas app.
    2. Add a connection to the **FlooringEstimates** table in the Excel file.

Add a gallery to a blank screen

1. On the **Insert** tab, select **Gallery**, and then select **Vertical**.



2. On the **Properties** tab of the right-hand pane, open the **Items** list, and then select **Flooring Estimates**.



3. (optional) In the **Layout** list, select a different option.

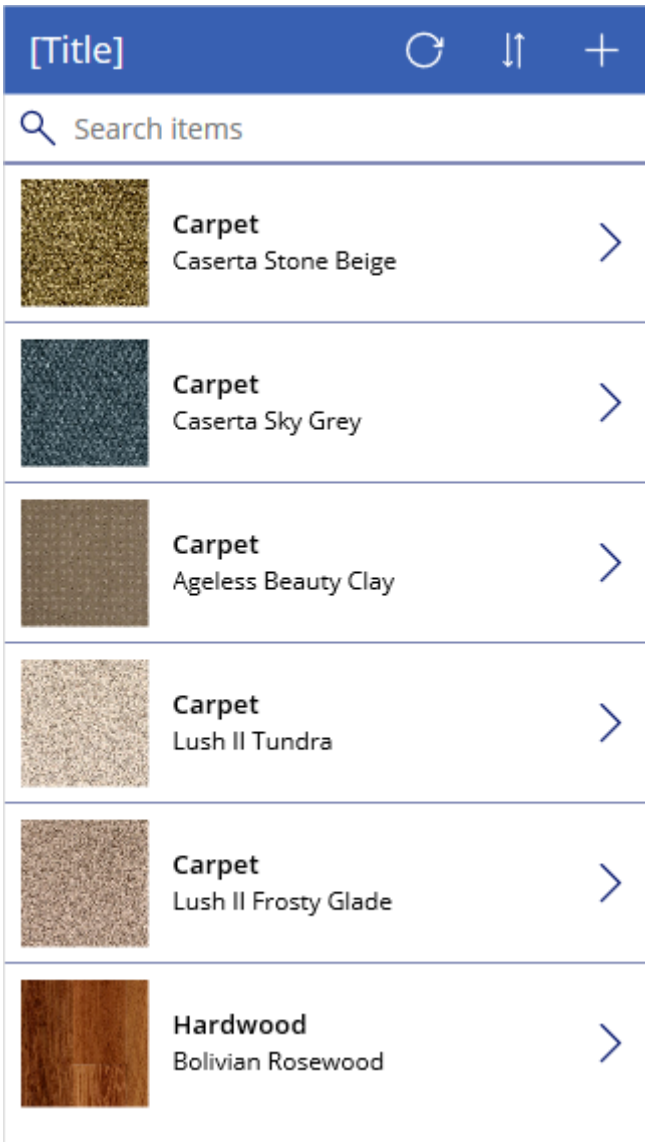
Add a gallery in a screen

1. On the **Home** tab, select **New screen** > **List screen**.

   A screen that contains a **Gallery** control and other controls, such as a search bar, appears.

2. Set the gallery's **Items** property to `FlooringEstimates`.

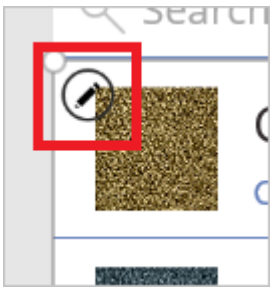
   The **Gallery** control shows the sample data.



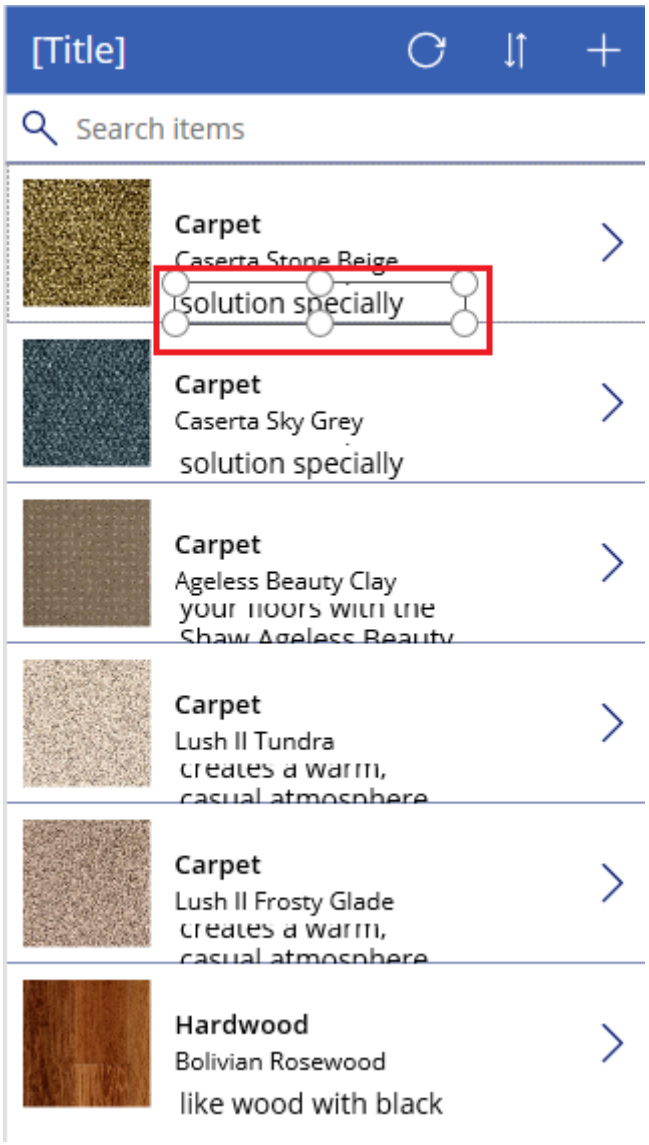
Add a control to the Gallery control

Before you do any other customization, ensure that the layout for your **Gallery** control most closely matches what you want. From there, you can further modify the **Gallery** template, which determines how all data in the **Gallery** control appears.

1. Select the template by clicking or tapping near the bottom of the **Gallery** control and then selecting the pencil icon in its upper-left corner.

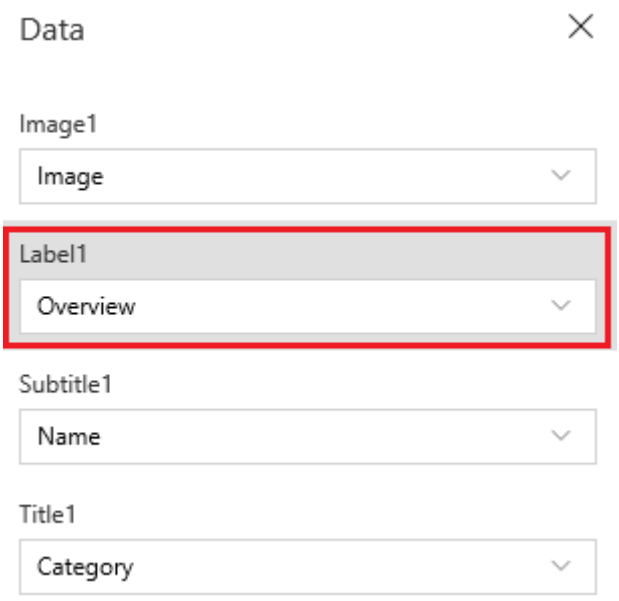


2. With the template still selected, add a **Label** control, and then move and resize it so that it doesn't overlap with other controls in the template.



3. Select the gallery, and then select **Edit** next to **Fields** on the **Properties** tab of the right-hand pane.

4. Select the label that you added earlier in this procedure, and then open the highlighted list in the **Data** pane.



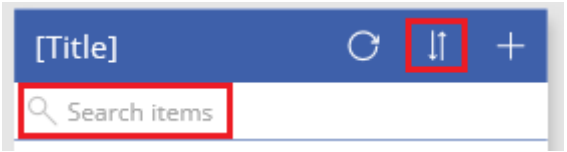
5. In that list, click or tap **Price**.

   The **Gallery** control shows the new values.



Filter and sort a gallery

The **Items** property of a **Gallery** control determines which items it shows. In this procedure, you configure that property so that it also determines which records appear based on filter criteria and in what order.



1. Set the **Items** property of the **Gallery** control to this formula:

```
Power Fx
```

```
Sort(
    Filter(
       FlooringEstimates,
       IsBlank(TextSearchBox1.Text) or TextSearchBox1.Text in Text(Name)
    ),
    Name,
    If(
        SortDescending1,
        SortOrder.Descending,
        SortOrder.Ascending
    )
)
```

For more information about the functions in this formula, see the formula reference.

2. Double-click the search box, and then type part or all of a product name in it.

   Only those items that meet the filter criterion appear.

3. While pressing the Alt key, select the sort icon one or more times to switch the sort order.

   The records toggle between ascending and descending alphabetical order based on the product name.

Highlight the selected item

Set the **Gallery** control's **TemplateFill** property to a formula that's similar to this example, but you can specify different colors if you want:

If(ThisItem.IsSelected, LightCyan, White)

Change the default selection

Set the **Gallery** control's **Default** property to the record that you want to select by default. For example, you can specify the fifth item in the **FlooringEstimates** data source:

Index(FlooringEstimates, 5)

In this example, you specify the first item in the **Hardwood** category of the **FlooringEstimates** data source:

LookUp(FlooringEstimates, Category = "Hardwood")

Next steps

Learn how to work with forms and formulas.

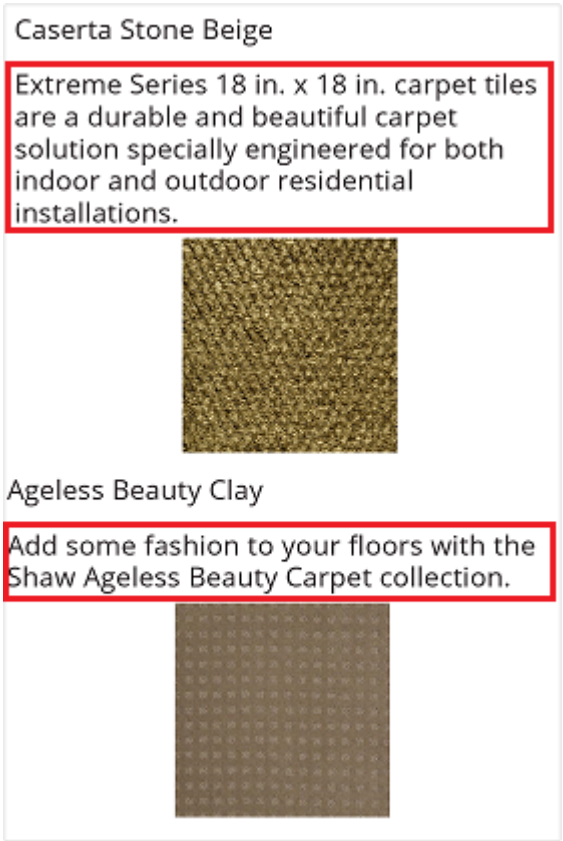
Show items of different heights in canvas apps gallery

Article • 01/19/2023

If different items in your data set contain different amounts of data in the same field, you can completely show items that contain more data without adding empty space after items that contain less data. Add and configure a **Flexible height** gallery control so that you can:

- Configure **Label** controls to expand or shrink based on their contents.
- Position each control so that it automatically appears just under the control above it.

In this tutorial, you show data about flooring products in a **Flexible height** gallery control. The image of each product appears 5 pixels below the overview, whether the overview contains five lines of text or two lines.



Suggested reading

If you've never added controls to a gallery, follow the steps in Show a list of items before you proceed in this topic.

Add data to a blank app

1. Download this Excel file, which contains names, overviews, and links to images of flooring products.

	A	B	C	D	E
1	Name	Category	Price	Image [image]	Overview
2	Lush II Frosty Glade	Carpet	$3.79	https://az787822.vo.msecr	Made with 100% premium nylon fiber, this textured carpet creates a warm, casual atmosphere that invites you to relax and thoroughly enjoy your home.
3	Ageless Beauty Clay	Carpet	$1.98	https://az787822.vo.msecr	Add some fashion to your floors with the Shaw Ageless Beauty Carpet collection.

2. Upload the Excel file to a cloud-storage account such as OneDrive, Dropbox, or Google Drive.

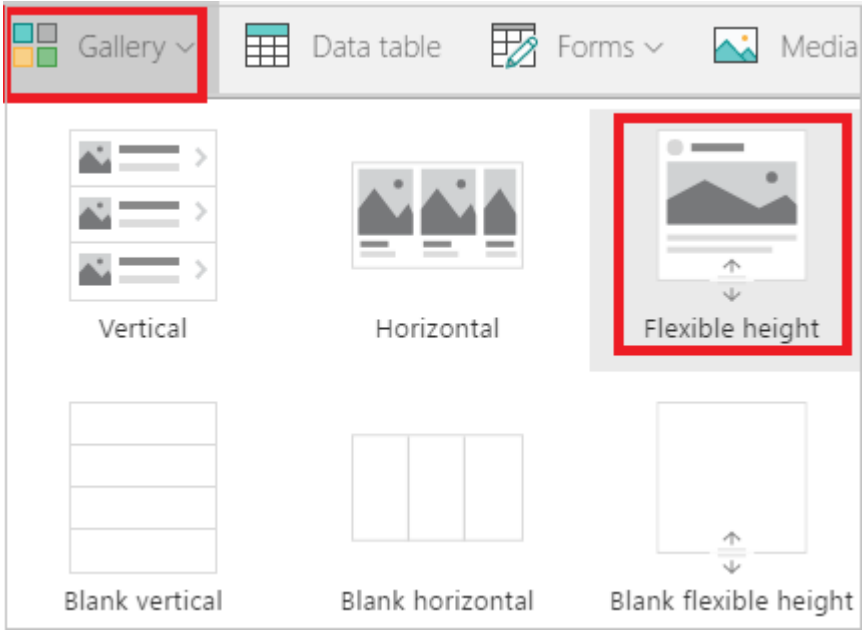
3. Create a blank app with **Phone** layout.

4. Add a connection to the **FlooringEstimates** table in the Excel file.

   For more information, see Add a connection.

Add data to a gallery

1. On the **Insert** tab, click or tap **Gallery**, and then click or tap **Flexible height**.



2. Resize the gallery to take up the entire screen.

3. Set the gallery's **Items** property to **FlooringEstimates**.

Show the product names

1. In the upper-left corner of the gallery, click or tap the pencil icon to select the gallery template.



2. With the gallery template selected, add a **Label** control to it.

3. Set the **Text** property of the **Label** control to this expression: **ThisItem.Name**

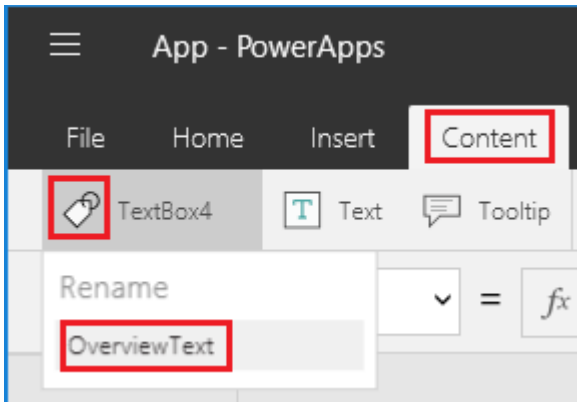


Show the product overviews

1. With the gallery template selected, add another **Label** control, and move it below the first **Label** control.

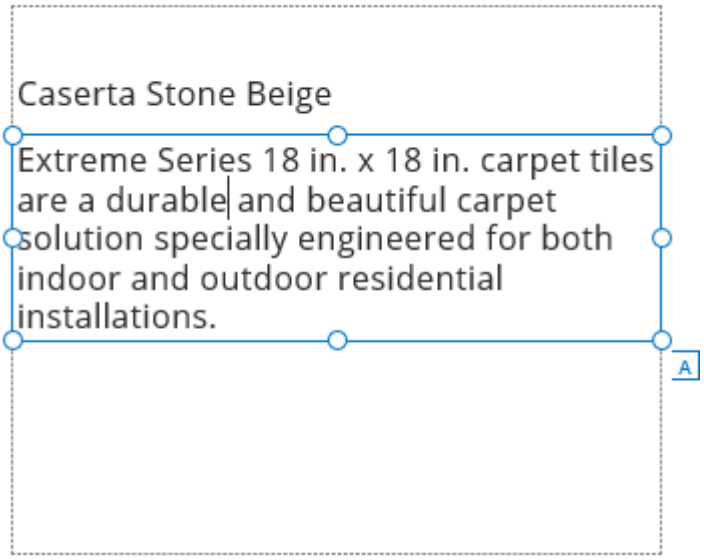
2. Set the **Text** property of the second **Label** control to this expression: **ThisItem.Overview**

3. With the second **Label** control selected, click or tap the name-tag icon on the **Content** tab, and rename the control to **OverviewText**.



4. Set the **AutoHeight** property of the **OverviewText** box to **true**.

   This step ensures that the box will grow or shrink to fit its contents.



Show the product images

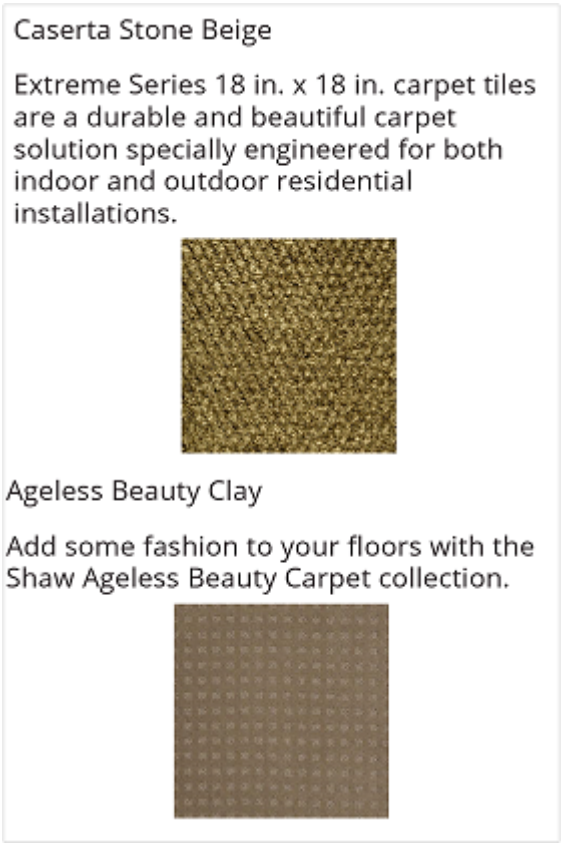
1. Resize the template so that it's twice as tall as it was.

   You can add controls to the template more easily as you build the app, and this change won't affect how the app looks when it runs.

2. With the gallery template selected, add an **Image** control, and move it below the **OverviewText** box.

3. Ensure that the **Image** property of the **Image** control is set to this expression: **ThisItem.Image**

4. Set the **Y** property of the **Image** control based on the position and the size of the **OverviewText** box, as in this expression:
   OverviewText.Y + OverviewText.Height + 5



Apply the same concept if you want to add more controls: set each control's **Y** property based on the **Y** and **Height** properties of the control above it.

Next steps

Learn more about how to work with a gallery control and formulas.

Best practices and recommendations

Article • 04/11/2024

The Gallery is the only control that can create other controls. It has its own scope. These advanced features can lead to unexpected behavior if the **Gallery** isn't configured correctly. This article covers best practices and recommendations when you're working with **Galleries**.

Don't change gallery items from within the gallery

It's easy to create unstable behavior if `OnChange` or `OnSelect` of child controls modifies the **Items** of the parent gallery. For example, a **Text input** in a gallery can have its `OnChange` property set to:

Power Fx

```
Patch(GalleryData, ThisItem, {Name: TextInput.Text})
```

This is usually fine. Most controls will only trigger `OnChange` when users change their value directly. However, these controls can cause issues because they also trigger OnChange when the system changes their value:

- Combo box
- Date picker
- Slider
- Toggle

For example, when `ComboBox.DefaultSelectedItems` changes, it triggers `OnChange`. Consider a **Combo box** in a **Gallery** with `DefaultSelectedItems` set to `First(ThisItem.A)` and its `OnChange` property set to:

Power Fx

```
UpdateIf(GalleryData, Name = ThisItem.Name, {
    A: Table({
        B: First(Self.SelectedItems).B
    })
})
```

This expression updates a record in the gallery's data source that matches the current item's name. The change is to replace the `A` column of the record with a new table.

This causes an infinite loop. When the user changes the value of the **Combo box**, `OnChange` is triggered, which updates `GalleryData`, which changes `DefaultSelectedItems` (because `First(ThisItem.A)` refers to a new table), which triggers `OnChange` again, and so on.

To avoid infinite loops, you can use modern controls or other controls that don't trigger `OnChange` when their data changes.

Slow performance from patching items

Even if you avoid unwanted loops, patching or updating **Gallery** items can be slow if the **Gallery** has many items. The **Gallery** doesn't just update the rows associated with the changed records. In some cases, it reloads all items. This is because of the loosely structured nature of data sources. It can be difficult for the **Gallery** to know whether just a single record changed or the entire data source changed.

Gallery.Selected can change unexpectedly

The **Selected** property of the **Gallery** is a moving target. It can change without user interaction when:

- **Gallery.Default** changes
- **Gallery.Items** changes
- Refreshing or updating data sources

This might not be desirable for your scenario. If you want a stable copy of the item that is selected by the user, consider storing it in a variable. For example, set the **OnSelect** property of the **Gallery** to a global variable `CurrentItem`:

Power Fx

```
Set(CurrentItem, Self.Selected)
```

You can then use `CurrentItem` in other parts of the app to refer to the most recent item selected by the user rather than by the system.

Don't use Gallery.Selected in a child control's event

The **Selected** property of the **Gallery** changes when a user selects an item. However, this event isn't related to events of child controls. Referring to `Gallery.Selected` in a child control's events can lead to unexpected results.

For example, when a user selects a **Checkbox** in a **Gallery**, the following events occur:

- `Checkbox.OnSelect`
- `Checkbox.OnCheck`
- `Gallery.Selected` changes to the newly clicked row

The order of these events isn't fixed. This is a problem if `Checkbox.OnSelect` is set to:

```
Power Fx

Notify(Gallery.Selected.Name)
```

`Gallery.Selected` might still be referring to the previously selected row when `Checkbox.OnSelect` is executed.

To avoid timing issues, don't use `Gallery.Selected` in events. If you must, use `Gallery.OnSelect` to respond to changes to `Gallery.Selected`.

Ensure that the Gallery knows the schema of its items

Setting `Gallery.Items` to a variable or output of a Canvas component can give unexpected results, depending on when the variable is set or changed.

Galleries need to know the schema of its **Items** when the app loads. A schema, also known as shape, is the name and type of columns in a data source. Consider this table:

```
Power Fx

[{A: "abc", B: 123}, {A: "def", B: 456}]
```

Its schema consists of a text column `A` and a number column `B`.

Most of the time, the **Gallery** can guess the schema of its **Items** from the data source and expressions used in the app. But if the **Items** property is set to the output of a **Canvas component** or **Import** control, the **Gallery** can't determine its schema. The output table from these controls might not be available when the app loads and the schema might even change. The **Gallery** won't render any items when it doesn't know its schema.

The same issue might happen when **Items** is set to a variable that's not initialized when the app loads.

As a workaround, you can hint the expected schema to the Gallery with a variable. Set **App.OnStart** to:

```
Power Fx

If(false, Set(GalleryData, [{A: "abc", B: 123}]), Set(GalleryData, []))
```

This lets the system know the schema of the `GalleryData` table. Then, you can use `GalleryData` as the **Items** property of the **Gallery**. Change it to the actual data source when needed.

Don't assume AllItems contains all items of its dataset

The **AllItems** property are the items that are loaded into view in a gallery. It's not all items in **Items**. **AllItems** can change when the user scrolls the gallery to load more items. Typically, this property is used to get values of child controls when the user is interacting with them. Hence, **AllItems** is guaranteed to have loaded that item and it's safe to refer to it. Don't refer to an item in **AllItems** if you're not sure whether the user has seen it.

Similarly, **AllItemsCount** is the number of items that are loaded into view in the gallery. It's not the total number of records in **Items**. To get the total records in **Items**, use `CountRows(<expression used for Items property>)`.

Next steps

Isolate issues in canvas apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Show, edit, or add a record in a canvas app

Article • 01/19/2023

In a canvas app, add and configure a **Display** form control to show all fields in a record. You can also add and configure an **Edit** form control to edit any field in a record, add a record, and save your changes back to a data source.

Prerequisites

- Learn how to add and configure a control in Power Apps.
- Download this Excel file, which contains sample data for this tutorial.
- Upload the Excel file to a cloud-storage account, such as OneDrive for Business.
- Create or open an app for phones, add a connection to the **FlooringEstimates** table in the Excel file.

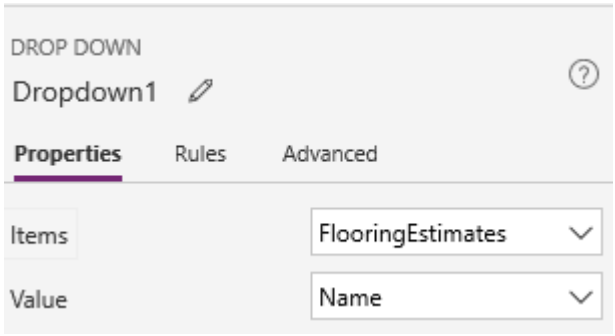
  You can add a form to a tablet app, but it won't match this article because the form will have three columns by default.
- If you open an existing app, add a screen to it.

Add a form, and show data

1. On a blank screen, add a **Drop down** control, and name it **ChooseProduct**.

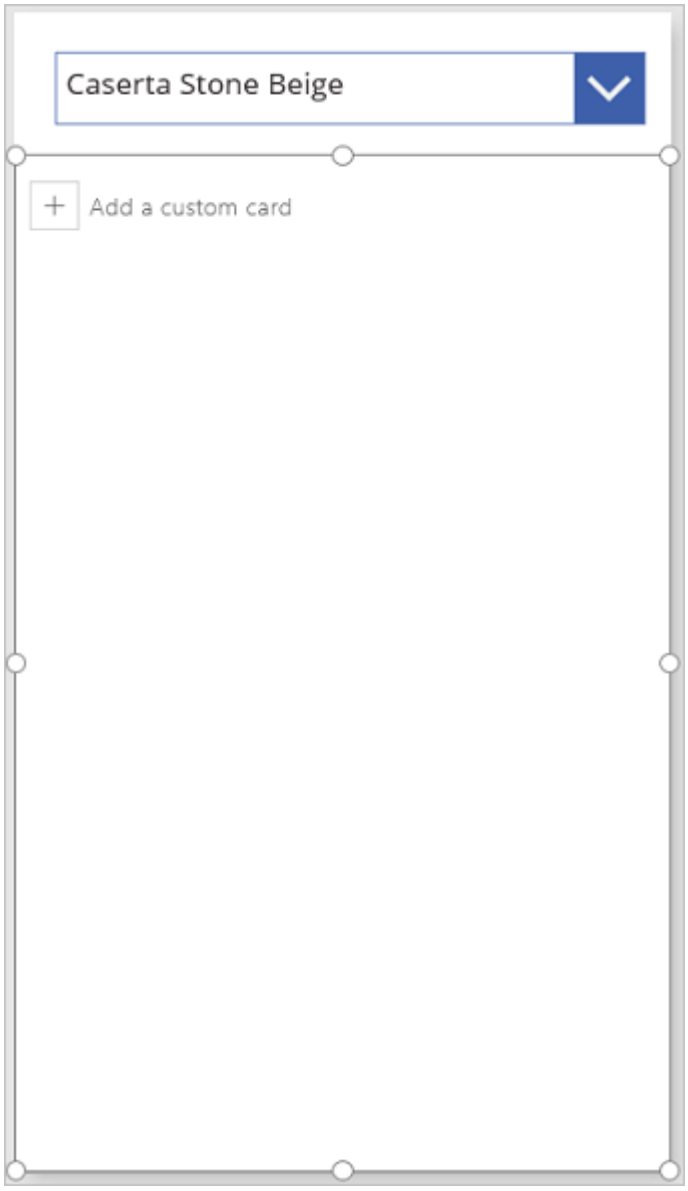
   > **Note**
   >
   > If you're not sure how to add a control, rename it, or set a property, see **Add and configure controls**.

2. On the **Properties** tab of the right-hand pane, set **Items** to `FlooringEstimates` and **Value** to `Name`.



The list shows names of flooring products from the data source.

3. Add an **Edit** form control, move it below **ChooseProduct**, and then resize the form to cover most of the screen.



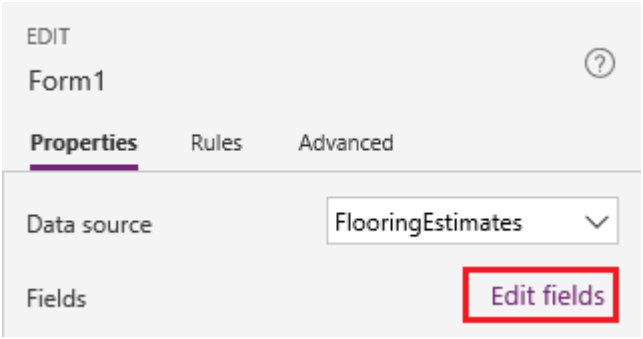
> **Note**
>
> This topic describes the **Edit** form control, but similar principles apply to the **Display** form control.

4. Set the form's DataSource property to **FlooringEstimates** and its Item property to this formula:

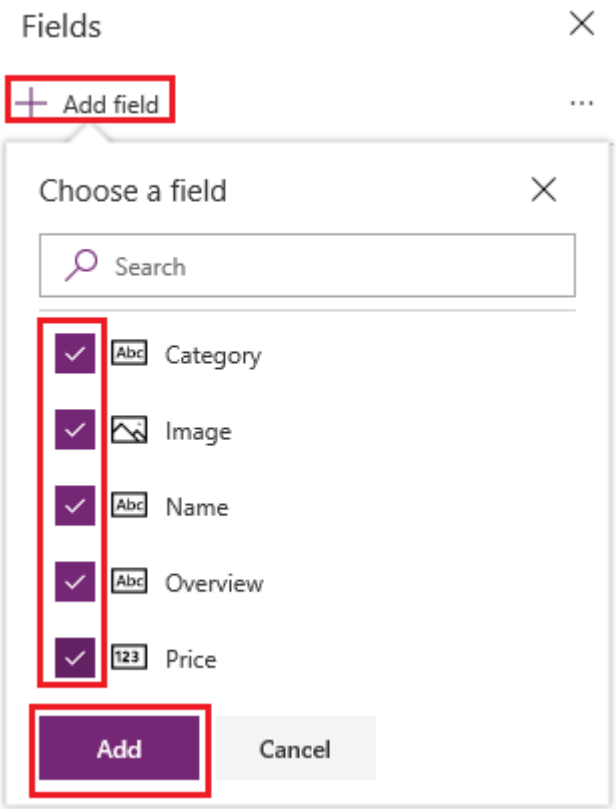
   ```
   ChooseProduct.Selected
   ```

   This formula specifies that, after you finish configuring the form, it will show the record that the user selects in **ChooseProduct**.

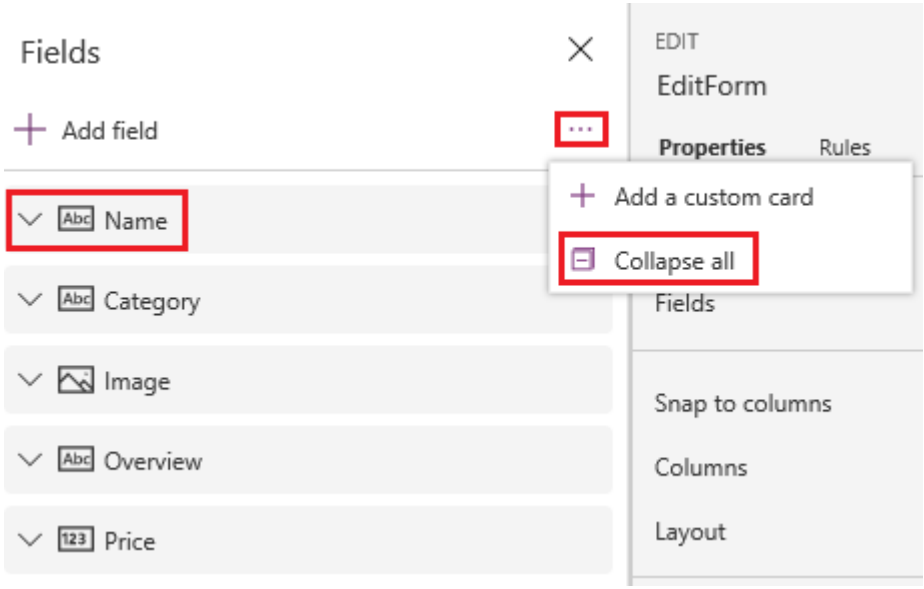
5. On the **Properties** tab of the right-hand pane, select **Edit fields**.



6. In the **Fields** pane, select **Add field**, select the check box for each field, and then select **Add**.



7. Select the ellipsis (...) next to **Add field**, select **Collapse all**, and then drag **Name** to the top of the list.



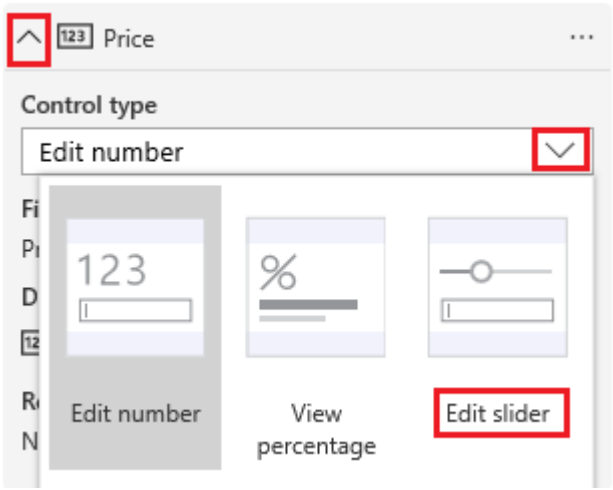
The **Edit** form control reflects your change.



Set the card type for a field

1. In the **Fields** pane, expand the **Price** field by selecting its down arrow.

2. Open the **Control type** list, and then select **Edit slider**.



   In the form, the **Price** field shows a **Slider** control instead of a **Text input** control.

3. (optional) Follow the same process to change the control for the **Overview** field to an **Edit multi-line text** control.

(Edit form only) Save changes

1. Rename the form **EditForm**.

2. Add a **Button** control, and set its **OnSelect** property to this formula:

   ```
   SubmitForm(EditForm)
   ```

3. Press F5 to open Preview, change the name of a product, and then select the button that you created.

   The **SubmitForm** function saves your changes to the data source.

4. (optional) Close Preview by pressing Esc (or by selecting the close icon in the upper-right corner).

Next steps

Learn more about working with forms and formulas.

Understand canvas-app forms

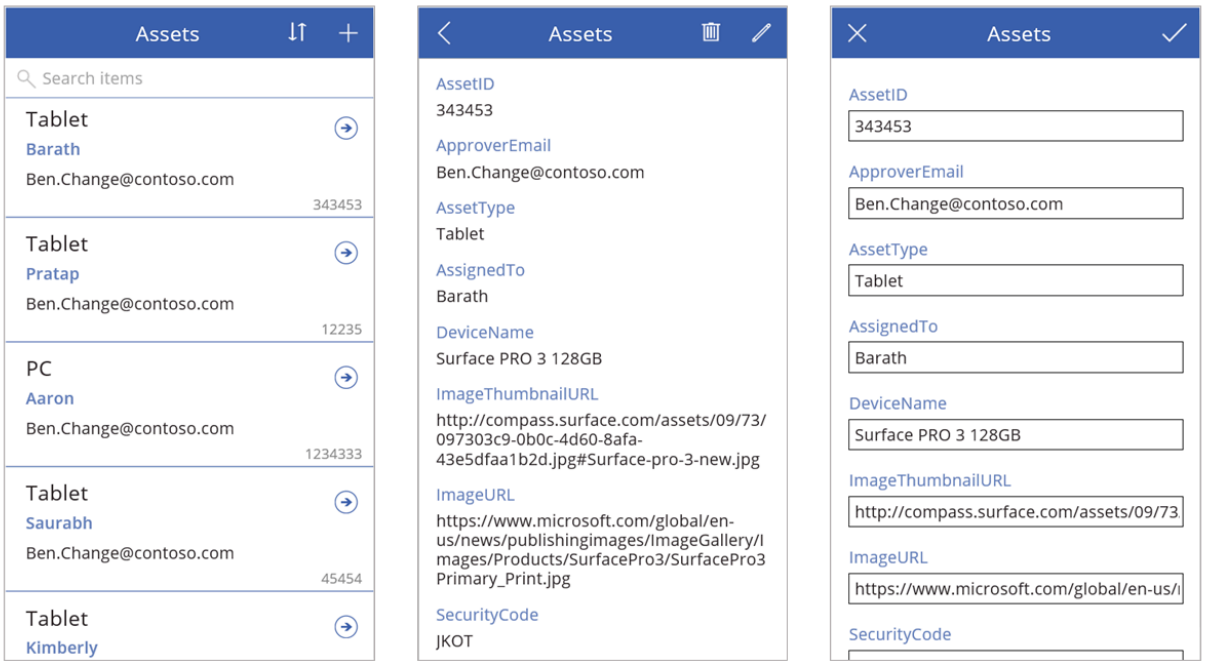
Article • 12/16/2022

Add three types of controls to a canvas app so that the user can browse for a record, display details about that record, and edit or create a record:

Expand table

Activity	Control	Description
Browse for a record	Gallery control	Filter, sort, search, and scroll through records in a data source, and select a specific record. Display only a few fields from each record to show several records at a time, even on a small screen.
Show details of a record	Display form control	For a single record, display many or all fields in that record.
Edit or create a record	Edit form control	Update one or more fields in a single record (or create a record starting with default values), and save those changes back to the underlying data source.

Put each control on a different screen to make them easier to distinguish:



Browse screen
Primary control: Gallery

Details screen
Primary control: Display form

Edit and Create screen
Primary control: Edit form

As this topic describes, combine these controls with formulas to create the overall user experience.

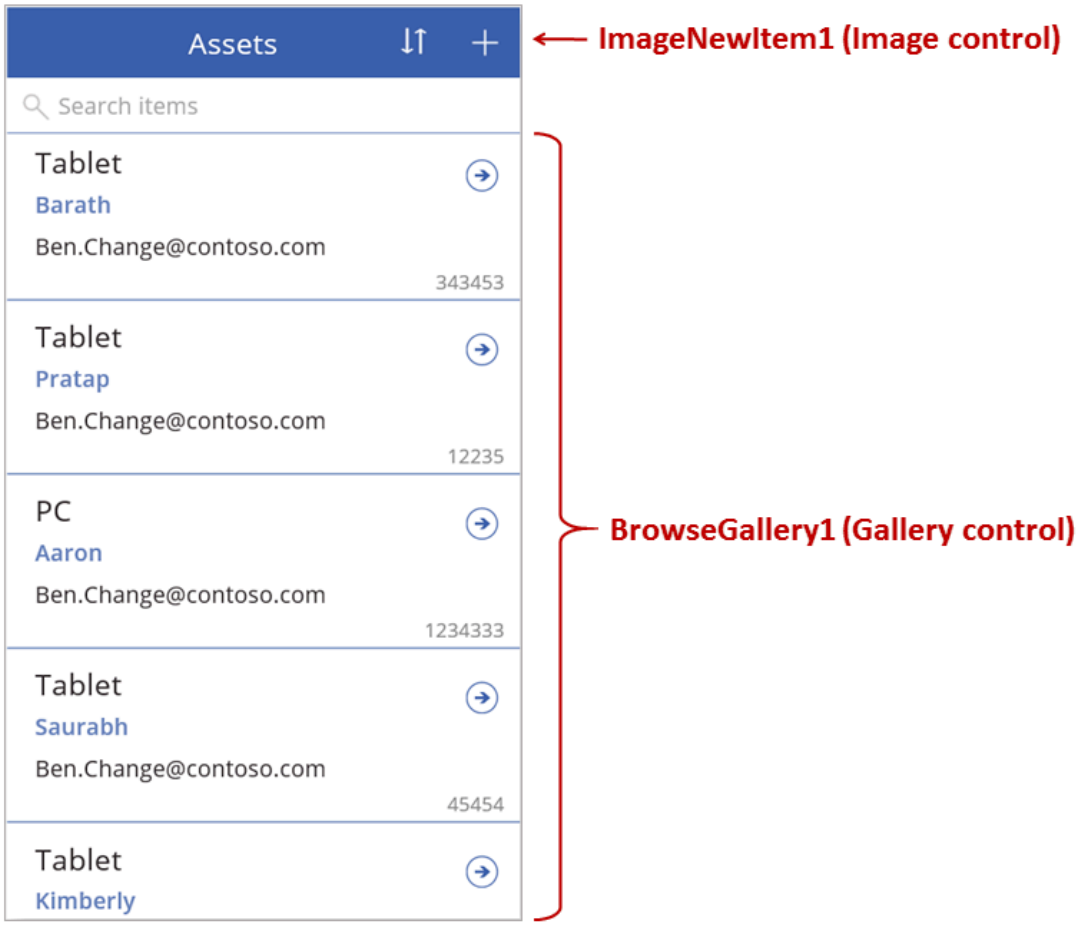
Prerequisites

- Sign up for Power Apps, and then sign in by providing the same credentials that you used to sign up.
- Learn how to configure a control in Power Apps.

Explore a generated app

Power Apps can automatically generate an app based on a data source that you specify. Each app contains three screens with the controls described earlier and formulas that connect them. Run these apps "out of the box," customize them for your specific goals, or examine how they work so that you can learn useful concepts that apply to your own apps. In the following sections, inspect the screens, controls, and formulas that drive a generated app.

Browse screen



This screen features these key formulas:

Expand table

Control	Supported behavior	Formula
BrowseGallery1	Display records from the **Assets** data source.	The **Items** property of the gallery is set to a formula that's based on the **Assets** data source.
ImageNewItem1	Display the **Edit and Create** screen with each field set to a default value, so that the user can easily create a record.	The **OnSelect** property of the image is set to this formula: **NewForm(EditForm1); Navigate(EditScreen1, None)**
NextArrow1 (in the gallery)	Display the **Details** screen to view many or all fields of the currently selected record.	The **OnSelect** property of the arrow is set to this formula: **Navigate(DetailScreen1, None)**

The primary control on this screen, **BrowseGallery1**, covers most of the area of the screen. The user can scroll through the gallery to find a specific record to display more fields or to update.

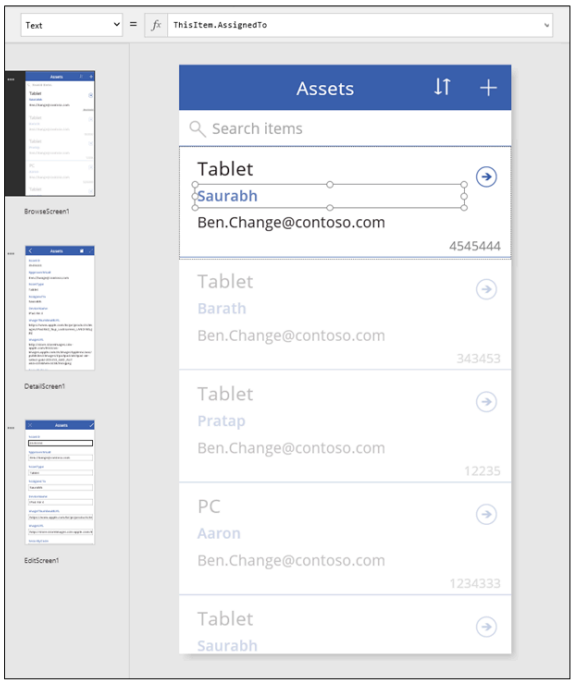
Set the **Items** property of a gallery to show records from a data source in it. For example, set that property to **Assets** to show records from a data source of that name.

> **Note**
>
> In a generated app, **Items** is set to a significantly more complicated formula by default so that the user can sort and search for records. You'll learn how to build that formula later in this topic; the simpler version is enough for now.

Instead of finding a record to display or edit, the user can create a record by selecting the "+" symbol above the gallery. Create this effect by adding an **Image** control, showing a "+" symbol in it, and setting its **OnSelect** property to this formula: **NewForm(EditForm1); Navigate(EditScreen1, None)**

This formula opens the **Edit and Create** screen, which features an **Edit form** control named **EditForm1**. The formula also switches that form into **New** mode, in which the form shows default values from the data source so that the user can easily create a record from scratch.

To examine any control that appears in **BrowseGallery1**, select that control in the first section of that gallery, which serves as a template for all other sections. For example, select the middle **Label** control on the left edge:



Text control selected, used to display a field of the record

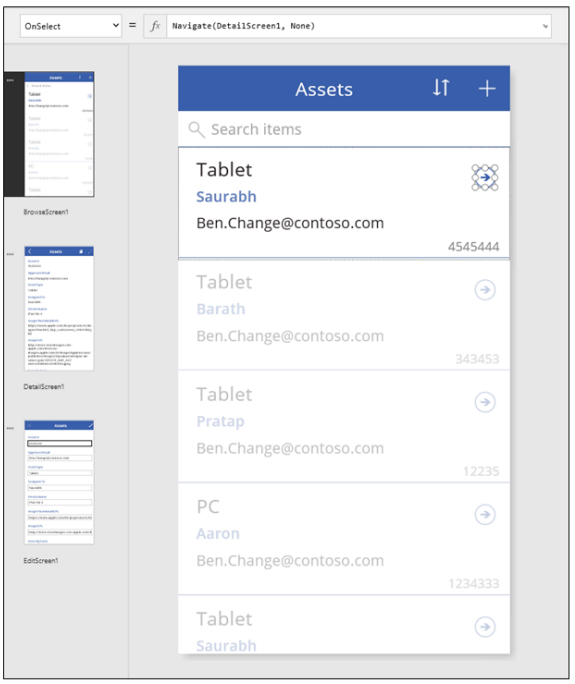


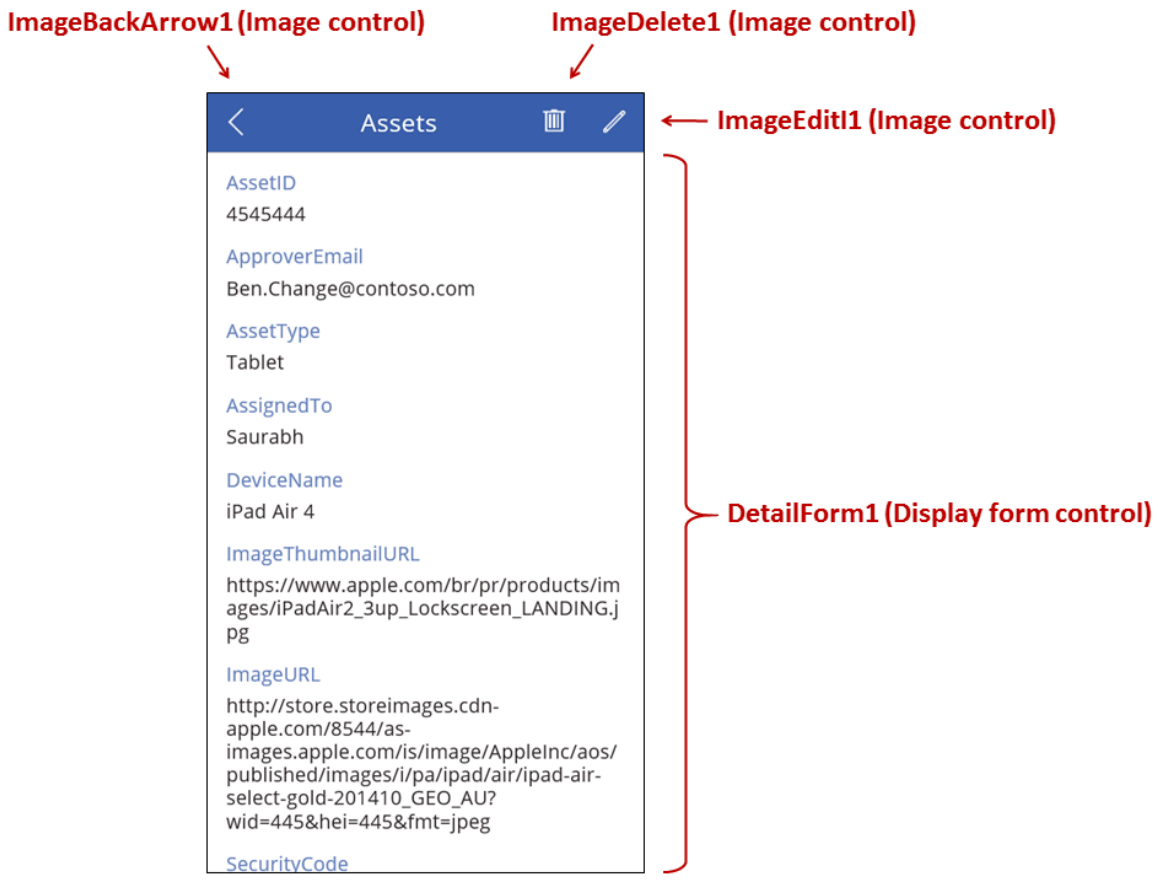
Image control selected, used to change screens

In this example, the control's **Text** property is set to **ThisItem.AssignedTo**, which is a field in the **Assets** data source. The **Text** property of the other three **Label** controls in the gallery are set to similar formulas, and each control shows a different field in the data source.

Select the **Shape** control (the arrow), and confirm that its **OnSelect** property is set to this formula:
Navigate(DetailScreen1, None)

If the user finds a record in **BrowseGallery1**, the user can select the arrow for that record to show more information about it in **DetailScreen1**. By selecting an arrow, the user changes the value of the **Selected** property of **BrowseGallery1**. In this app, that property determines which record appears in not only **DetailScreen1** but also, if the user decides to update the record, the **Edit and Create** screen.

Detail screen



This screen features these key formulas:

Expand table

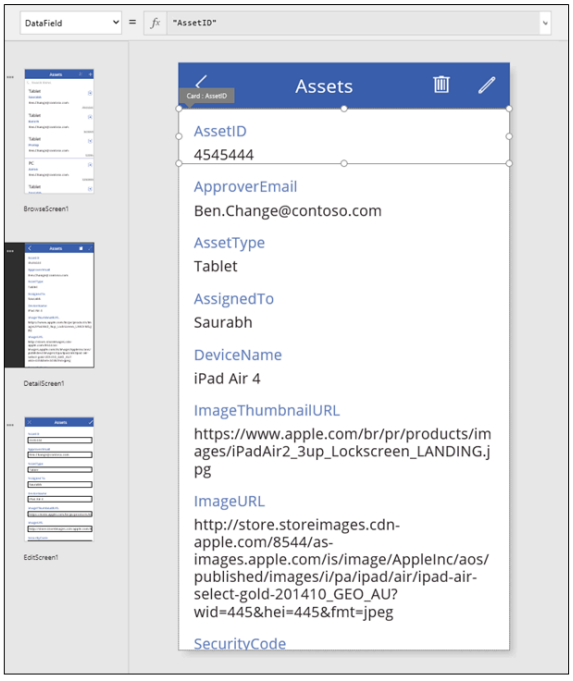
Control	Supported behavior	Formula
DetailForm1	Displays a record in the **Assets** data source	Set the DataSource property to **Assets**.
DetailForm1	Determines which record to display. In a generated app, displays the record that the user selected in the gallery.	Set the Item property of this control to this value: **BrowseGallery1.Selected**
Card controls	In a Display form control, displays a single field in a record.	Set the DataField property to the name of a field, enclosed in double quotation marks (for example, **"Name"**).
ImageBackArrow1	When the user selects this control, opens **BrowseScreen1**.	Set the OnSelect property to this formula: **Back()**
ImageDelete1	When the user selects this control, deletes a record.	Set the OnSelect property to this formula: **Remove(Assets, BrowseGallery1.Selected)**

Control	Supported behavior	Formula
ImageEdit1	When the user selects this control, opens the **Edit and Create** screen to the current record.	Set the OnSelect property to this formula: **Navigate(EditScreen1, None)**

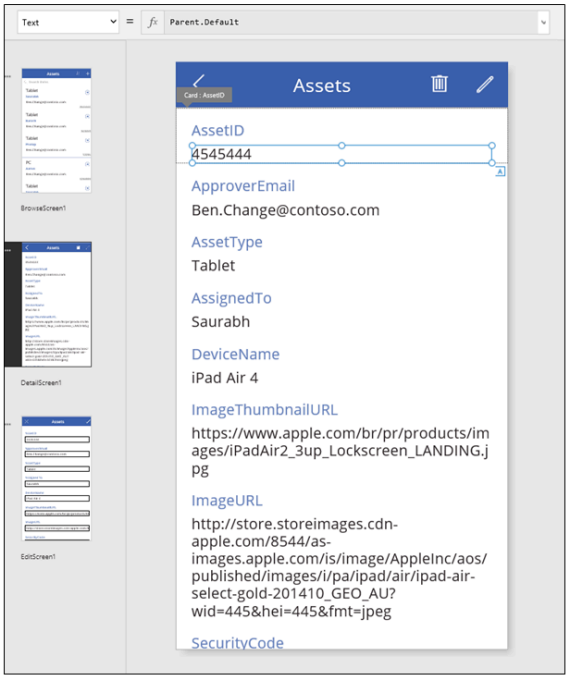
At the top of the screen, three images sit outside of **DetailForm1** and act as buttons, orchestrating between the three screens of the app.

DetailForm1 dominates this screen and displays the record that the user selected in the gallery (because the form's Item property is set to **BrowseGallery1.Selected**). The DataSource property of the form also provides metadata about the data source, such as a user-friendly display name for each field.

DetailForm1 contains several Card controls. You can select either the Card control itself or the control that it contains to discover additional information.



Card selected, used to display a field of the record

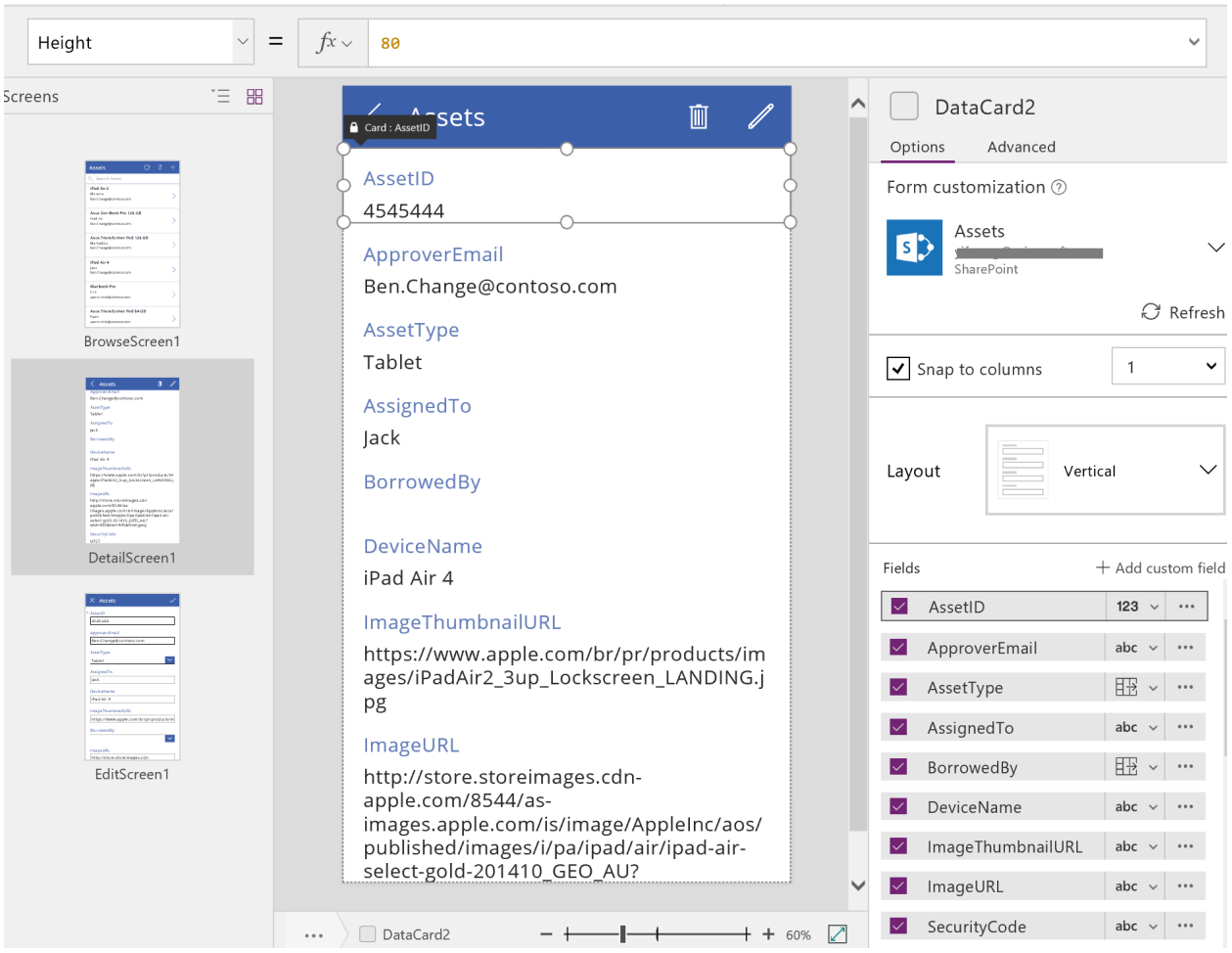


Text control selected, showing the field's value

The DataField property of a Card control determines which field the card displays. In this case, that property is set to **AssetID**. The card contains a Label control for which the Text property is set to **Parent.Default**. This control shows the **Default** value for the card, which is set through the DataField property.

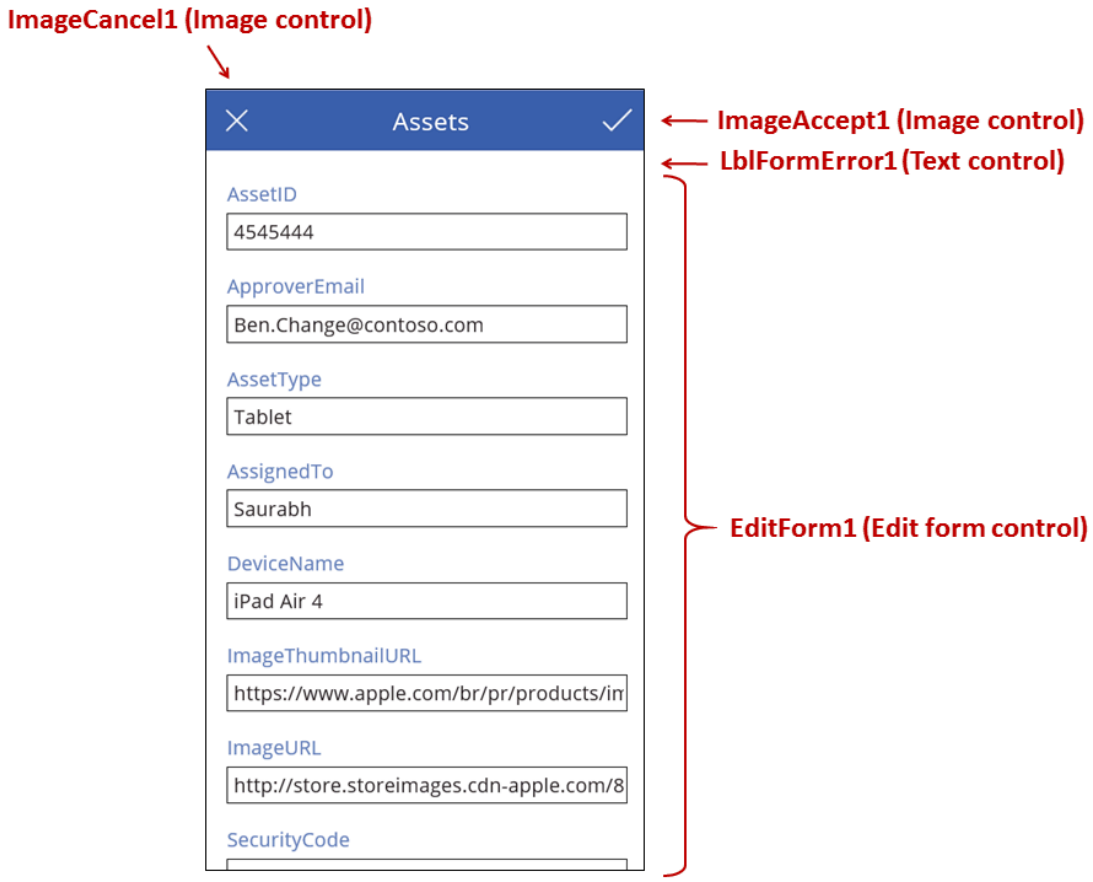
In a generated app, Card controls are locked by default. When a card is locked, you can't modify some properties, such as DataField, and the formula bar is unavailable for those properties. This restriction helps ensure that your customizations don't break the basic

functionality of the generated app. However, you can change some properties of a card and its controls in the right-hand pane:



In the right-hand pane, you can select which fields to display and in which kind of control each field displays.

Edit/Create screen



This screen features these key formulas:

Expand table

Control	Supported behavior	Formula
EditForm1	Displays a record in the **Assets** data source.	Set the DataSource property to **Assets**.
EditForm1	Determines which record to display. In a generated app, displays the record that the user selected in **BrowseScreen1**.	Set the Item property to this value: **BrowseGallery1.Selected**
Card controls	In a Edit form control, provides controls so that the user can edit one or more fields in a record.	Set the DataField property to the name of a field, enclosed in double quotation marks (for example, **"Name"**).
ImageCancel1	When the user selects this control, discards any changes in progress, and opens the **Details** screen.	Set the OnSelect property to this formula: **ResetForm(EditForm1); Back()**
ImageAccept1	When the user selects this control, submits changes to the data source.	Set the OnSelect property to this formula: **SubmitForm(EditForm1)**

Control	Supported behavior	Formula
EditForm1	If changes are accepted, returns to the previous screen.	Set the **OnSuccess** property to this formula: **Back()**
EditForm1	If changes aren't accepted, remain on the current screen so that the user can fix any issues and try to submit again.	Leave the **OnFailure** property blank.
LblFormError1	If changes aren't accepted, shows an error message.	Set the **Text** property to this value: **EditForm1.Error**

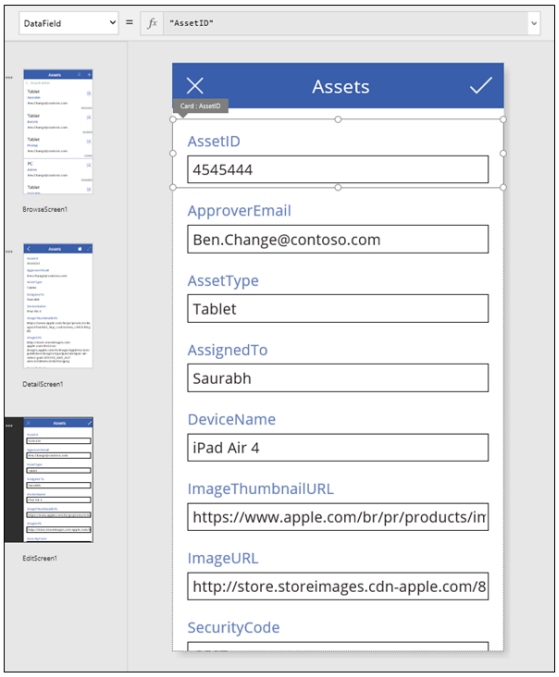
As in the **Details** screen, a form control, named **EditForm1**, dominates the **Edit and Create** screen. In addition, the **Item** property of **EditForm1** is set to **BrowseGallery1.Selected**, so the form displays the record that the user selected in **BrowseScreen1**. While the **Details** screen shows each field as read-only, the user can update the value of one or more fields by using the controls in **EditForm1**. It also uses the **DataSource** property to access metadata about this data source, such as the user-friendly display name for each field and the location where changes should be saved.

If the user selects the "X" icon to cancel an update, the **ResetForm** function discards any unsaved changes, and the **Back** function opens the **Details** screen. Both the **Details** screen and the **Edit and Create** screen show the same record until the user selects a different one on **BrowseScreen1**. The fields in that record remain set to the values that were most recently saved, not any changes that the user made and then abandoned.

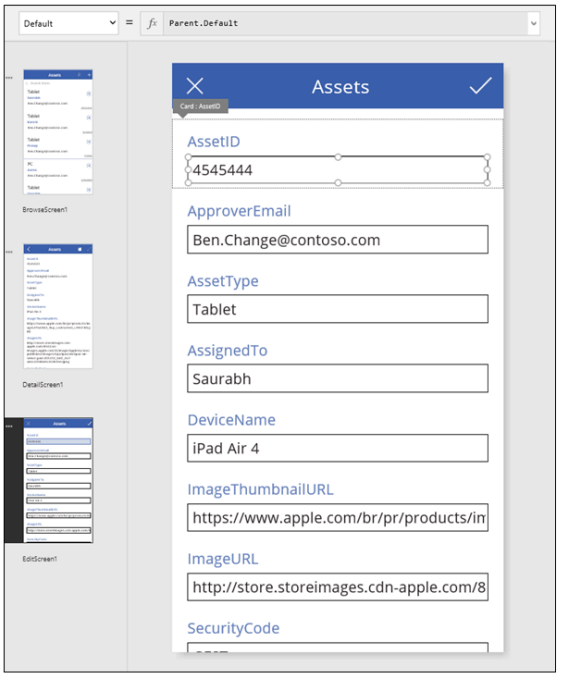
If the user changes one or more values in the form and then selects the "checkmark" icon, the **SubmitForm** function sends the user's changes to the data source.

- If the changes are successfully saved, the form's **OnSuccess** formula runs, and the **Back()** function opens the detail screen to show the updated record.
- If the changes aren't successfully saved, the form's **OnFailure** formula runs, but it doesn't change anything because it's *blank*. The **Edit and Create** screen remains open so that the user can either cancel the changes or fix the error. **LblFormError1** shows a user-friendly error message, to which the form's **Error** property is set.

As with a **Display form** control, an **Edit form** control contains **Card** controls, which contain other controls that show different fields in a record:



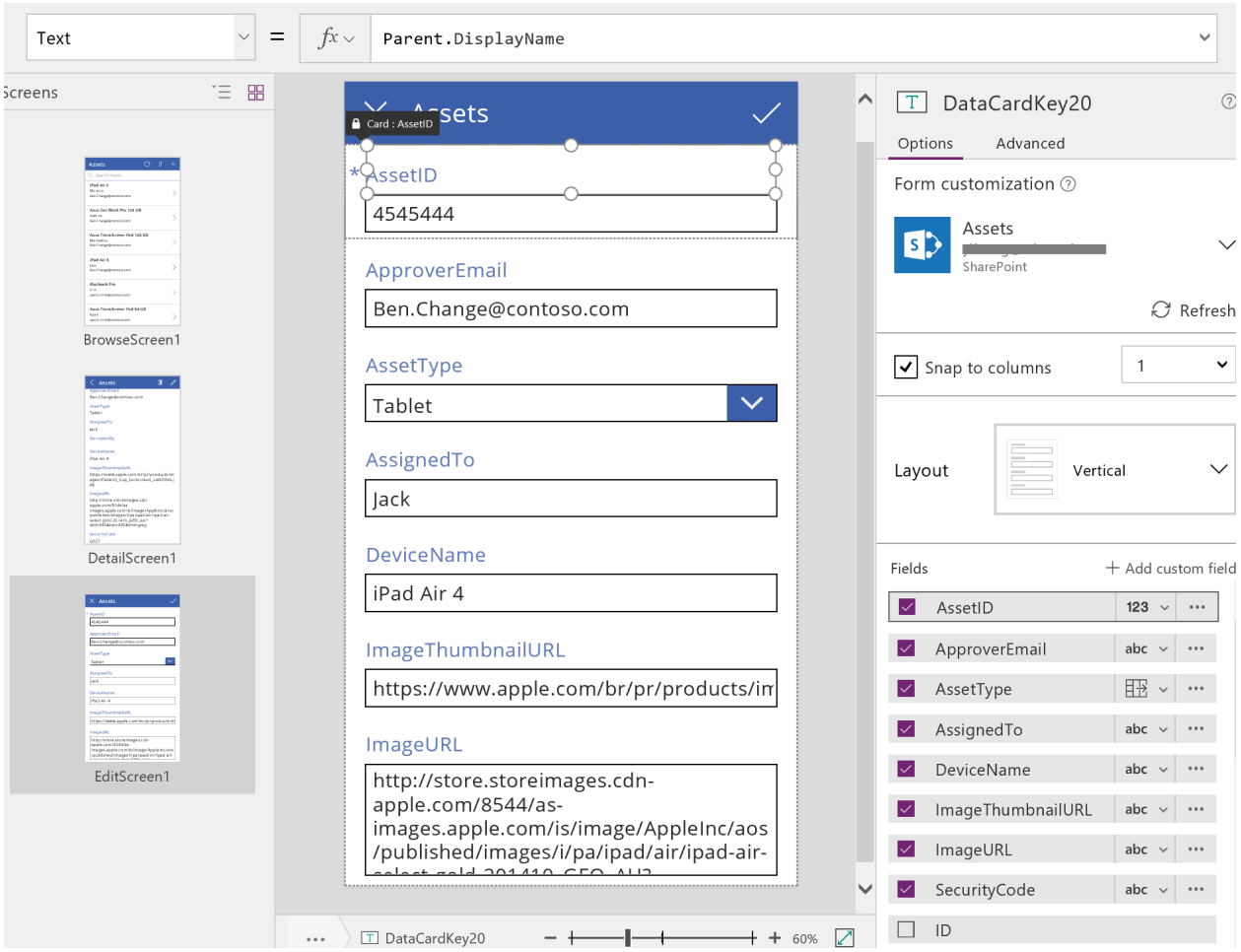
Card selected, used to edit a field of the record



Text input control selected, for editing the field's value

In the previous image, the selected card shows the **AssetID** field and contains a **Text input** control so that the user can edit the value of that field. (In contrast, the detail screen shows the same field in a **Label** control, which is read-only.) The **Text input** control has a **Default** property, which is set to **Parent.Default**. If the user were creating a record instead of editing one, that control would show an initial value that the user can change for the new record.

In the right-hand pane, you can show or hide each card, rearrange them, or configure them to show fields in different types of controls.



Build an app from scratch

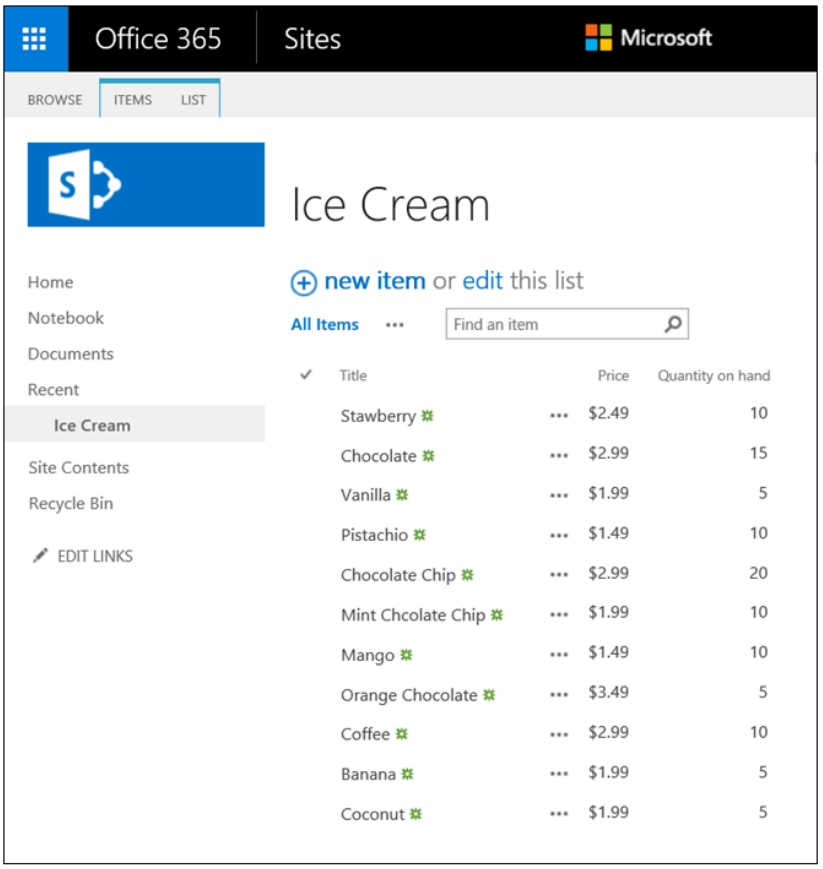
By understanding how Power Apps generates an app, you can build one yourself that uses the same building blocks and formulas discussed earlier in this topic.

Identify test data

To get the most from this topic, start with a data source with which you can experiment. It should contain test data that you can read and update without concern.

> **Note**
>
> If you use a list created using Microsoft Lists, a SharePoint library, or an Excel table that contains column names with spaces as your data source, Power Apps will replace the spaces with "_x0020_". For example, "**Column Name**" in SharePoint or Excel will appear as "**Column_x0020_Name**" in Power Apps when displayed in the data layout or used in a formula.

To follow the rest of this topic exactly, create a list named "Ice Cream" that contains this data:



- Create an app from blank, for phones, and connect it to your data source.

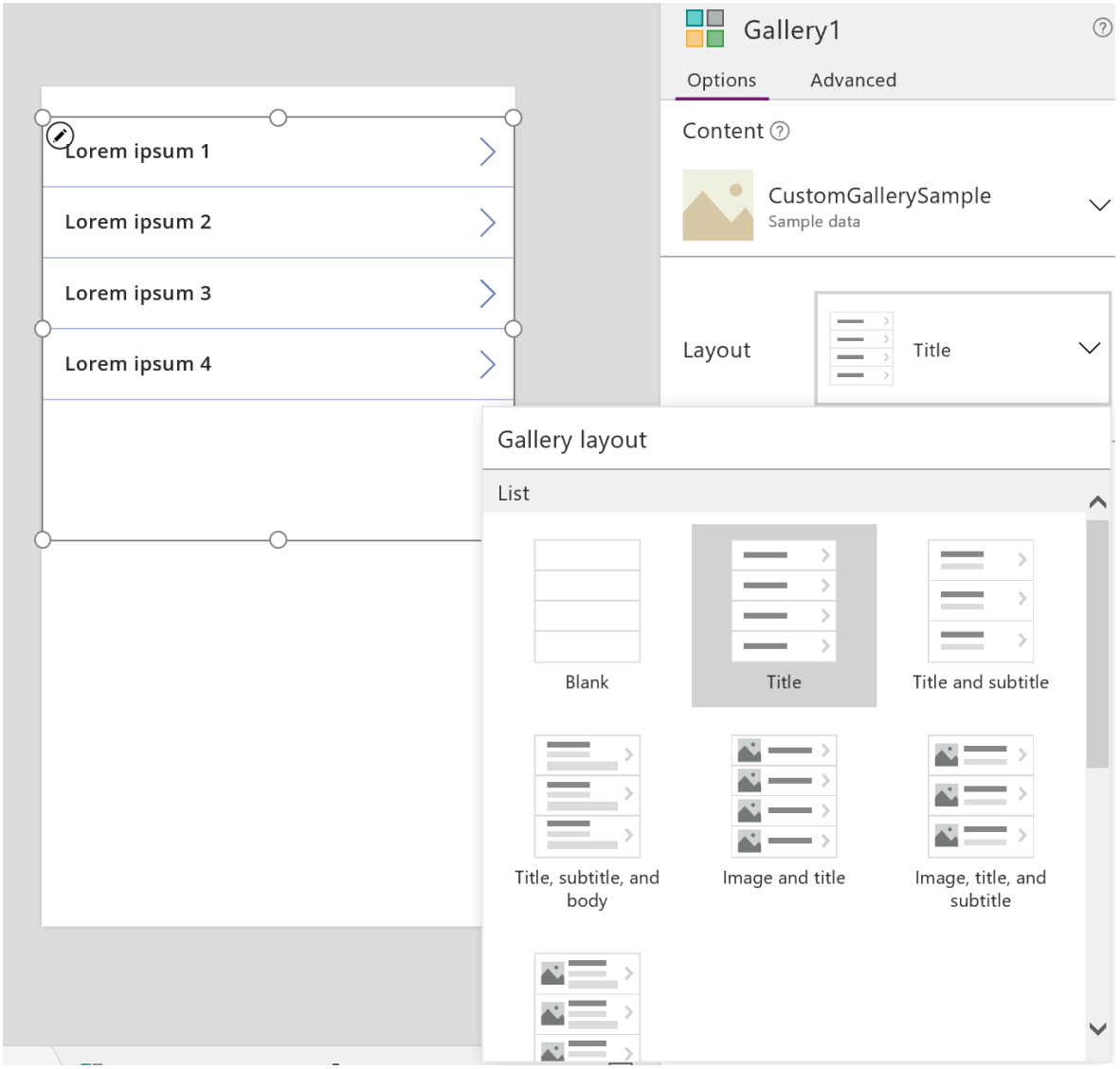
  > **Note**
  >
  > Tablet apps are very similar, but you may want a different screen layout to make the most of the extra screen space.

  The examples in the rest of the topic are based on a data source named **Ice Cream**.

Browse records

Get a quick piece of information from a record by finding it in a gallery on a browse screen.

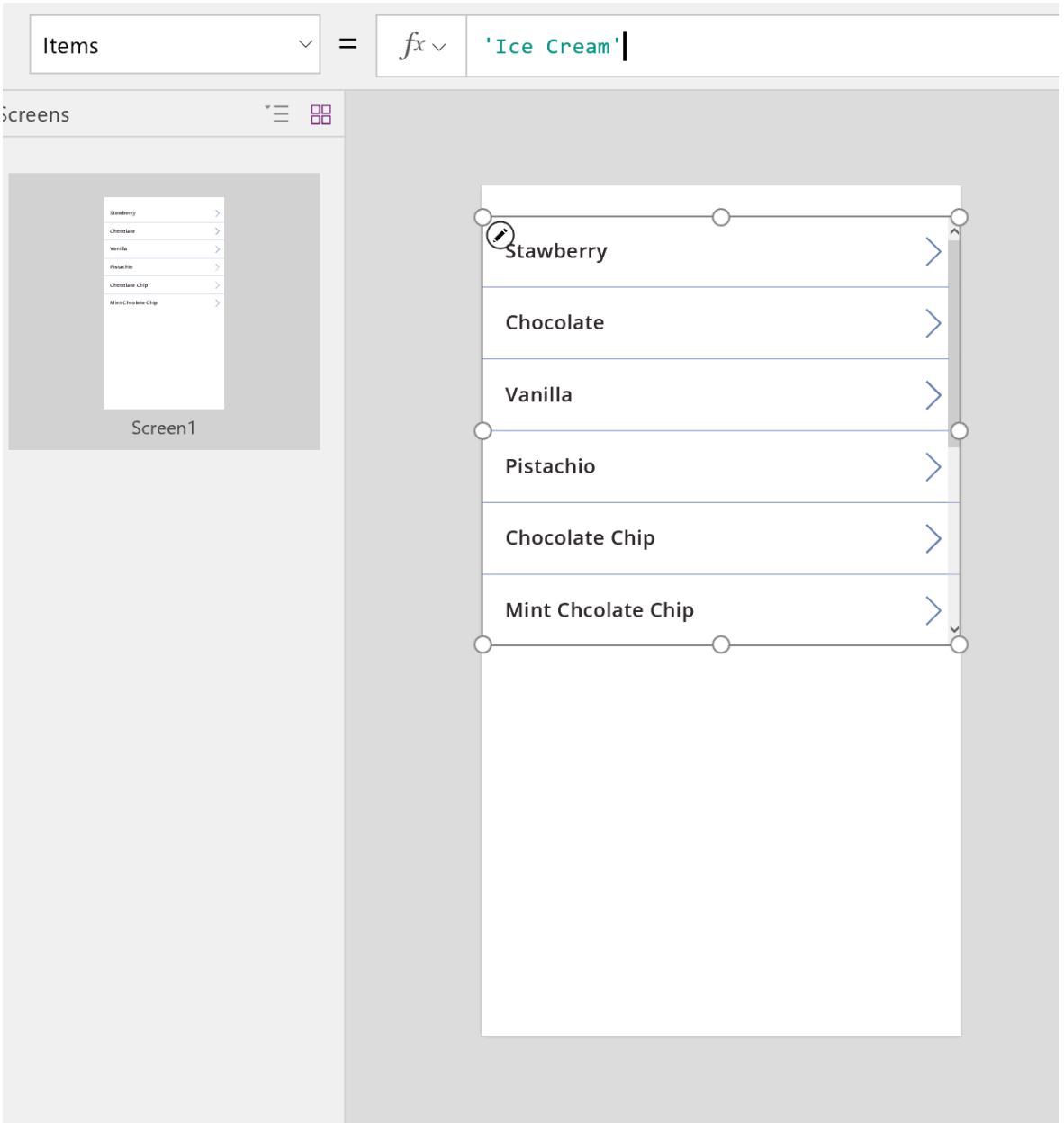
1. Add a **Vertical** gallery, and change the layout to **Title** only.



2. Set the gallery's **Items** property to **Ice Cream**.

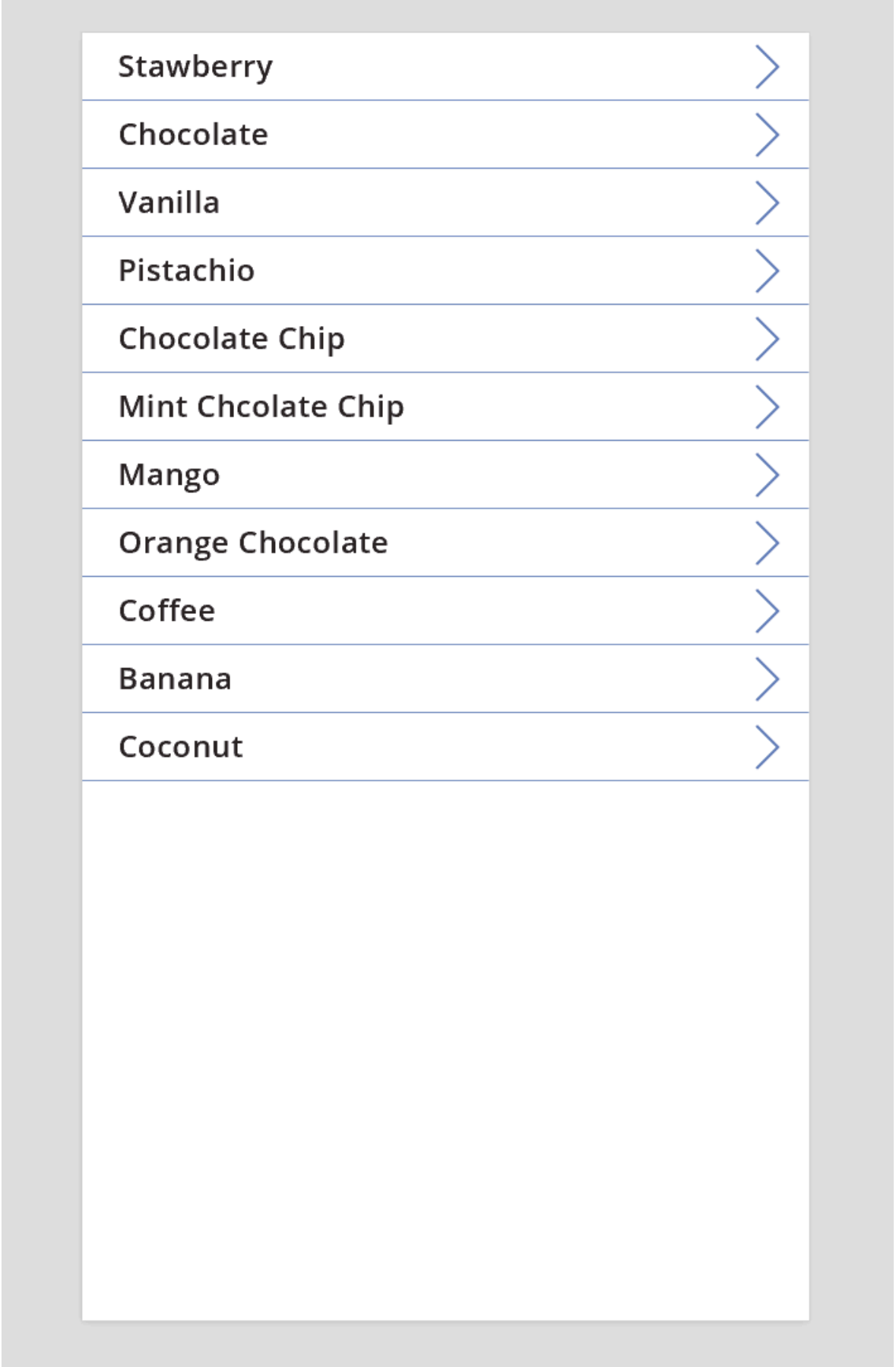
3. Set the **Text** property of the first label in the gallery to **ThisItem.Title** if it's set to something else.

   The label now shows the value in the **Title** field for each record.



4. Resize the gallery to fill the screen, and set its **TemplateSize** property to **60**.

   The screen resembles this example, which shows all records in the data source:



View details

If the gallery doesn't show the information that you want, select the arrow for a record to open the details screen. A Display form control on that screen shows more, possibly

all, fields for the record that you selected.

The **Display form** control uses two properties to display the record:

- **DataSource** property. The name of the data source that holds the record. This property populates the right-hand panel with fields and determines the display name and data type (string, number, date, etc.) of each field.
- **Item** property. The record to display. This property is often connected to the **Selected** property of the **Gallery** control so that the user can select a record in the **Gallery** control and then drill into that record.

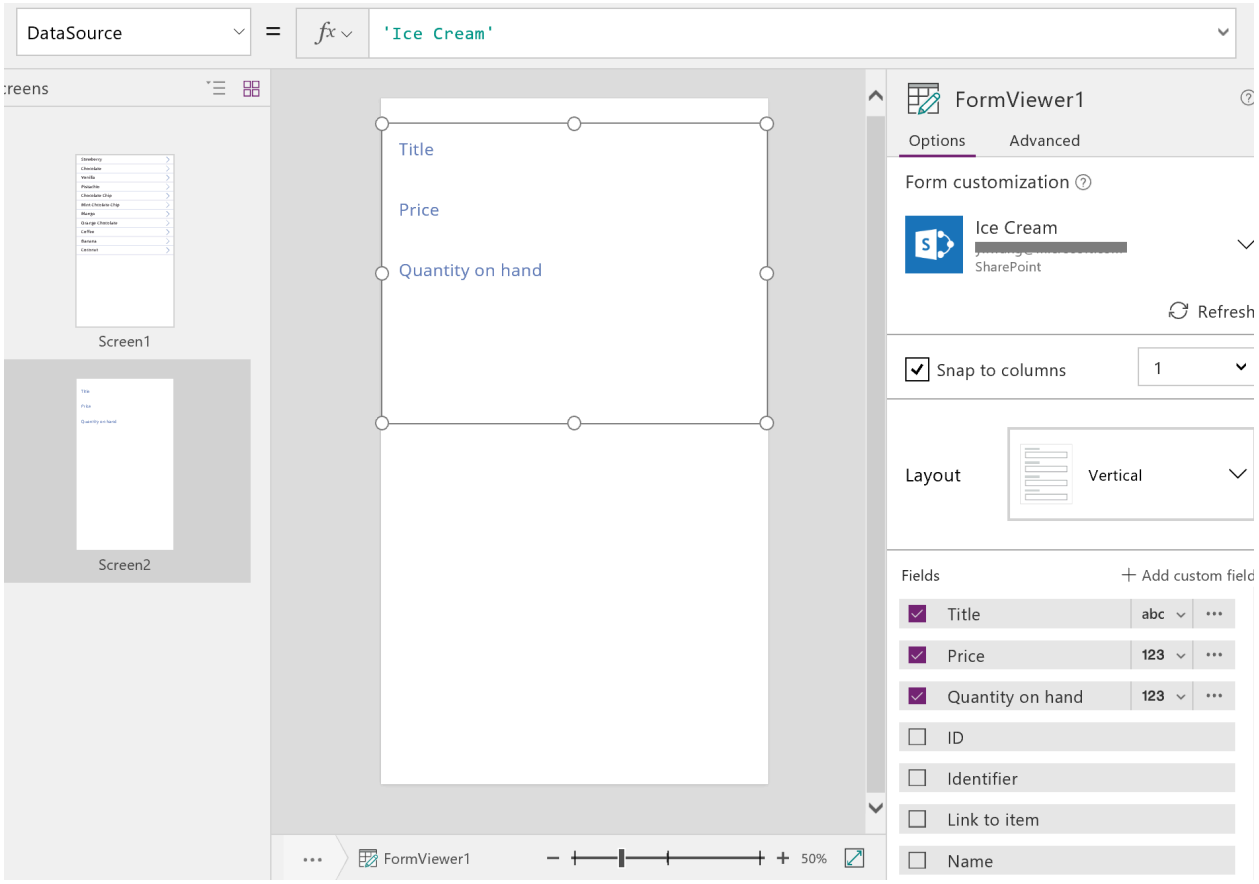
When the **DataSource** property is set, you can add and remove fields through the right-hand pane and change how they're displayed.

On this screen, users can't intentionally or accidentally change any values of the record. The **Display form** control is a read-only control, so it won't modify a record.

To add a **Display form** control:

1. Add a screen, and then add a **Display form** control to it
2. Set the **DataSource** property of the form control to '**Ice Cream**'.

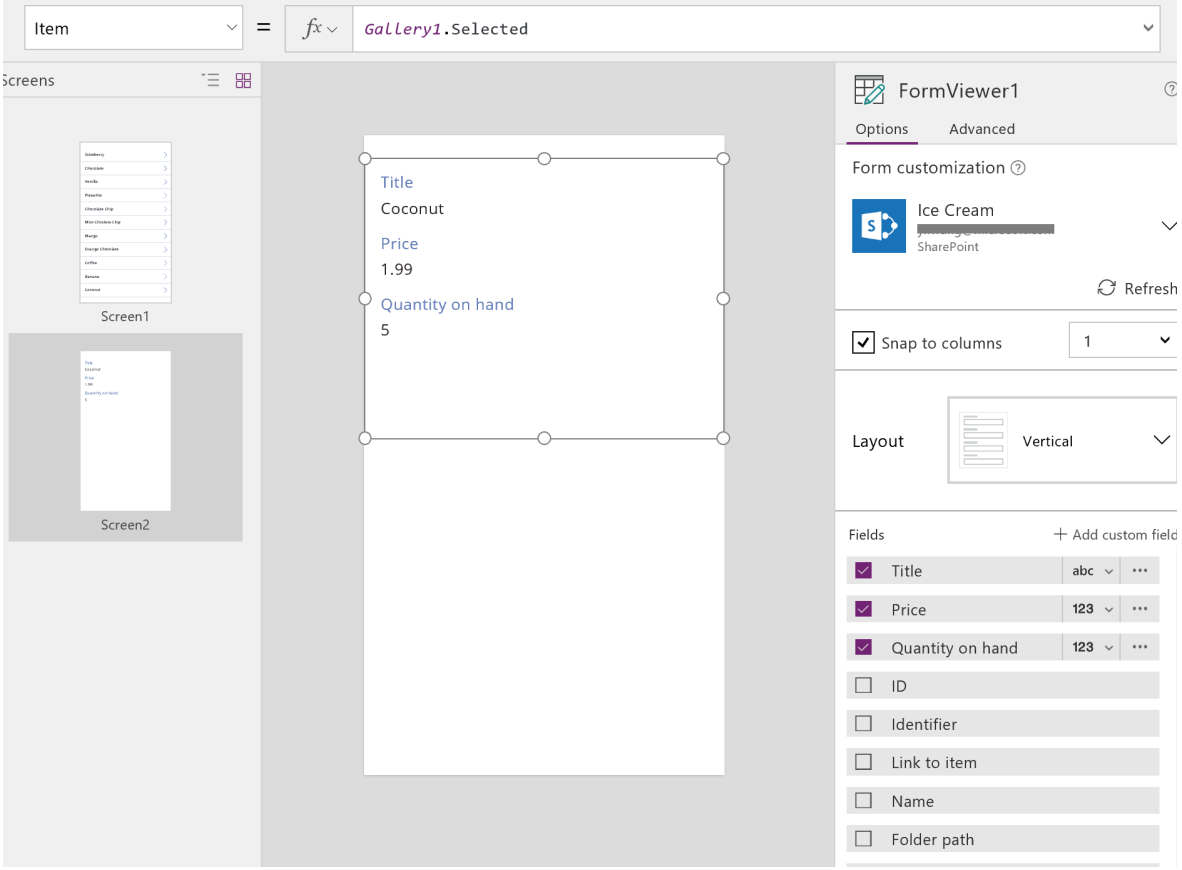
In the right-hand pane, you can select the fields to display on your screen and which type of card to display for each field. As you make changes in the right-hand pane, the **DataField** property on each **Card** control is set to the field that the user will interact with. Your screen should resemble this example:



Finally, we need to connect the **Display form** control to the **Gallery** control so that we can look at details for a specific record. As soon as we complete setting the **Item** property, the first record from the gallery will appear in our form.

- Set the **Item** property of the **Display form** control to **Gallery1.Selected**.

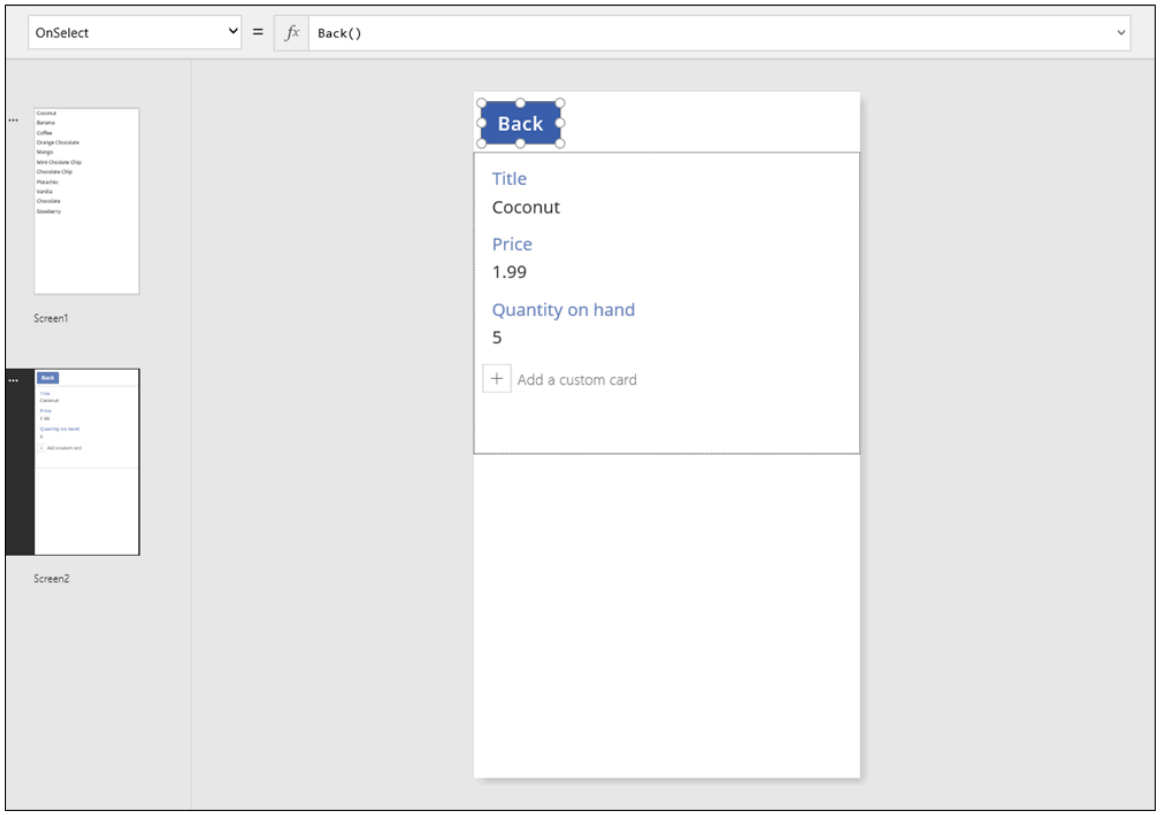
  The details for the selected item appear in the form.



Great! We now turn to navigation: how a user opens the details screen from the gallery screen and opens the gallery screen from the details screen.

- Add a **Button** control to the screen, set its **Text** property to show **Back**, and set its **OnSelect** property to **Back()**.

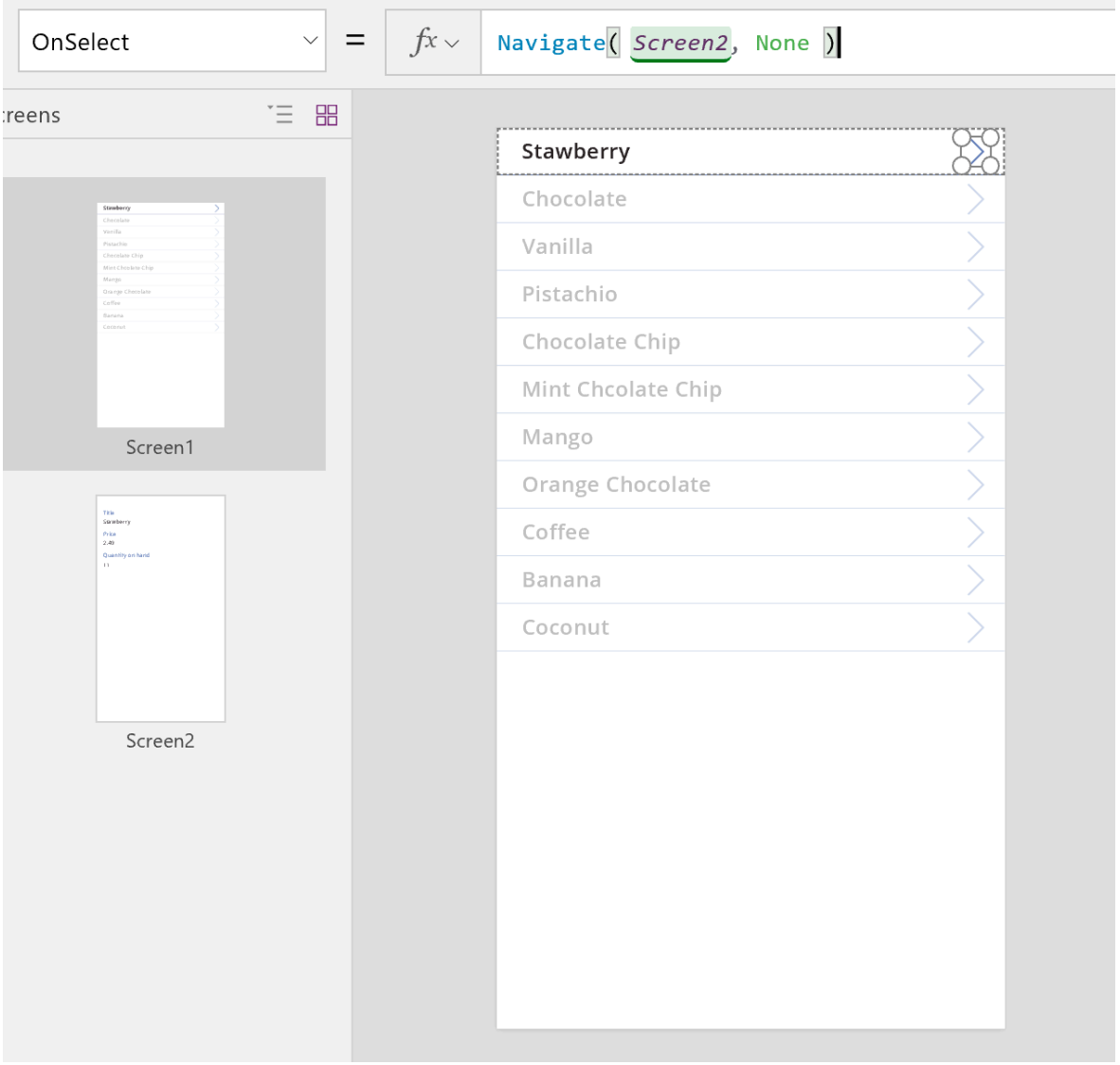
  This formula returns the user back to the gallery when they finish viewing details.



Now, let's return to the **Gallery** control and add some navigation to our detail screen.

1. Switch to the first screen, which is hosting our **Gallery** control, and select the arrow in the first item in the gallery.

2. Set the **OnSelect** property of the shape to this formula:
   Navigate(Screen2, None)



3. Press F5, and then select an arrow in the gallery to show the details for an item.

4. Select the **Back** button to return to the gallery of products, and then press Esc.

Editing details

Finally, our last core activity is changing the contents of a record, which users accomplish in an **Edit form** control.

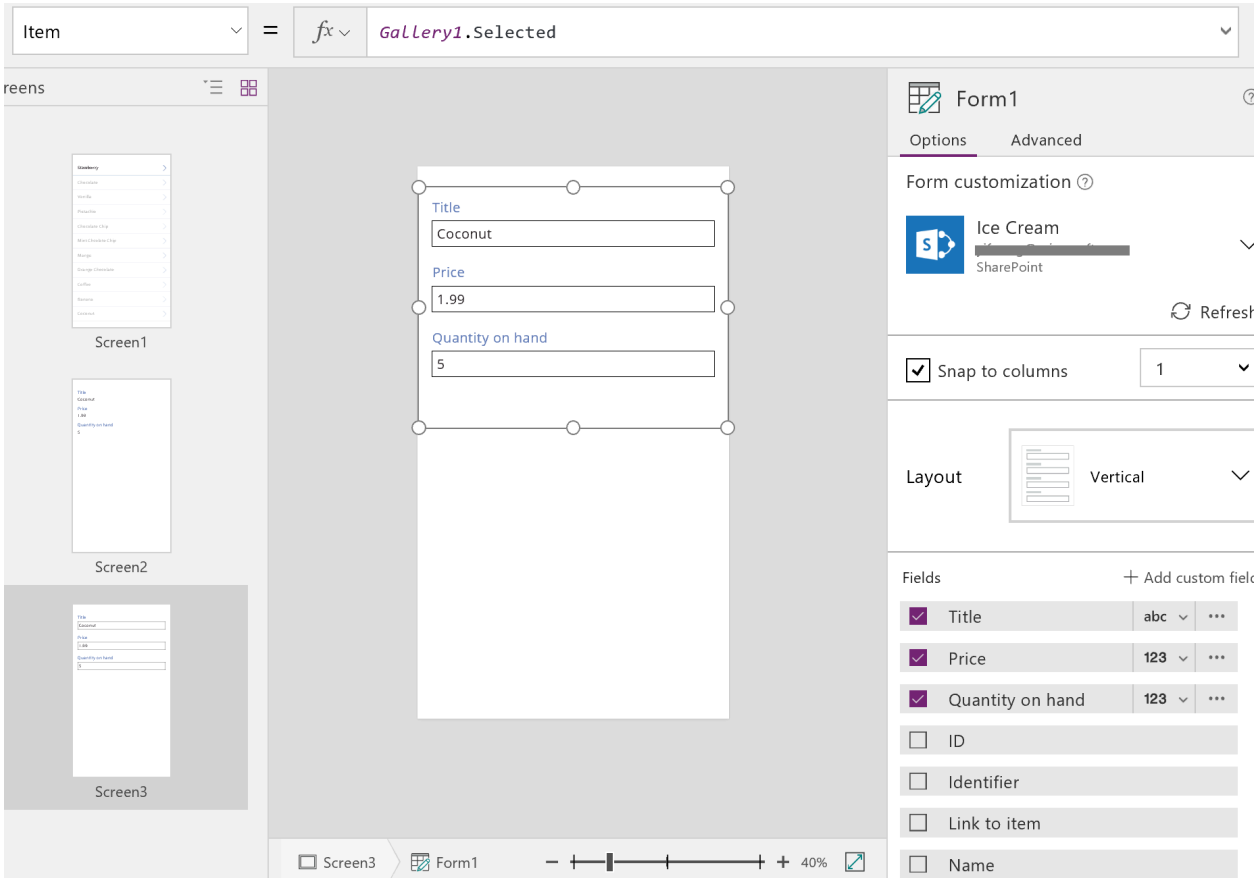
The **Edit form** control uses two properties to display and edit the record:

- **DataSource** property. The name of the data source that holds the record. Just as with the **Display form** control, this property populates the right-hand panel with fields and determines the display name and data type (string, number, date, etc.) for each field. This property also determines whether each field's value is valid before submitting it to the underlying data source.
- **Item** property. The record to edit, which is often connected to the **Selected** property of the **Gallery** control. That way, you can select a record in the **Gallery** control, show it in the details screen, and edit it in the **Edit and Create** screen.

To add an **Edit form** control:

1. Add a screen, add an **Edit form** control, and then set the form's **DataSource** property to '**Ice Cream**'.
2. Set the **Item** property to **Gallery1.Selected**.

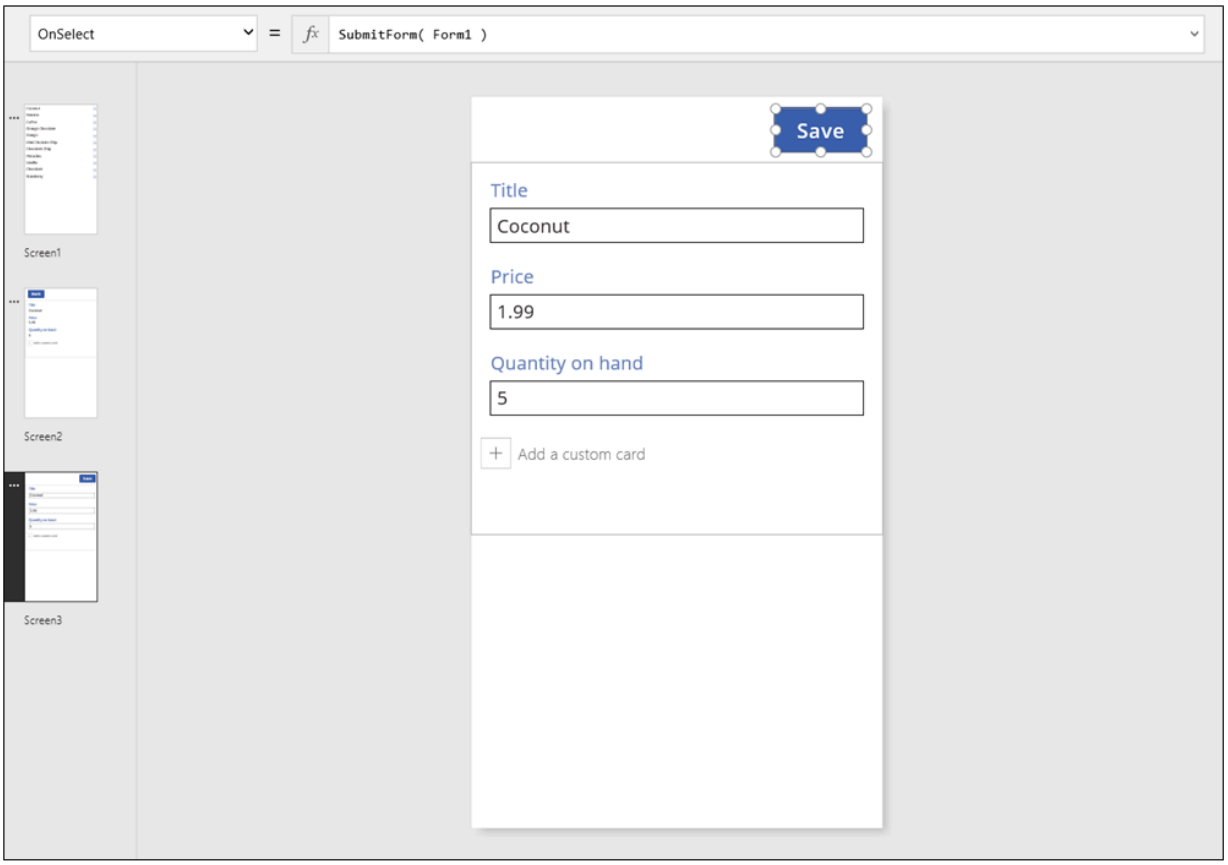
You can now select the fields to display on your screen. You can also select which type of card to display for each field. As you make changes in the right-hand pane, the **DataField** property on each **Card** control is set to the field your user will interact with. Your screen should resemble this example:



These two properties are the same as the properties on the **Display form** control. And with these alone, we can display the details of a record.

The **Edit form** control goes further by offering the **SubmitForm** function to write back changes to the data source. You use this with a button or image control to save a user's changes.

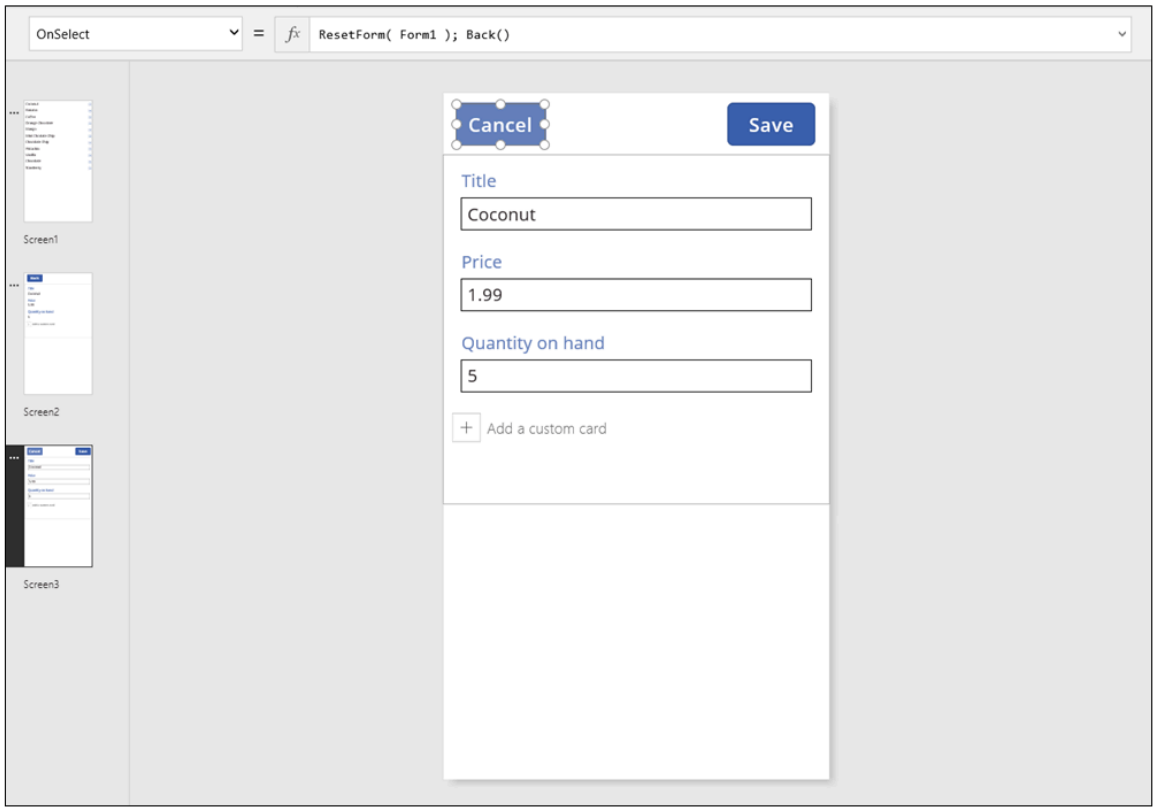
- Add a **Button** control, set its **Text** property to show **Save**, and set its **OnSelect** property to this formula:
  SubmitForm(Form1)



To add navigation to and from this screen:

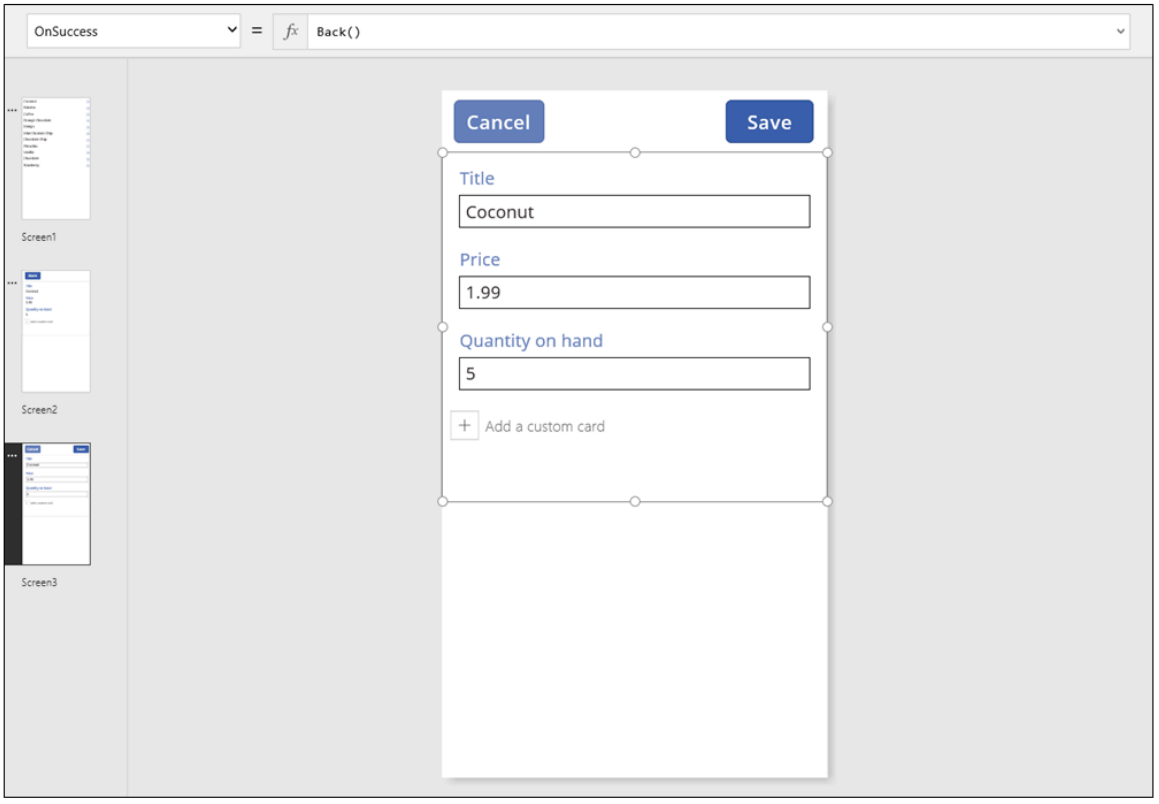
1. Add another **Button** control, set its **Text** property to show **Cancel**, and set its **OnSelect** property to this formula:
   ResetForm(Form1); Back()

   This formula discards any unsaved edits and opens the previous screen.



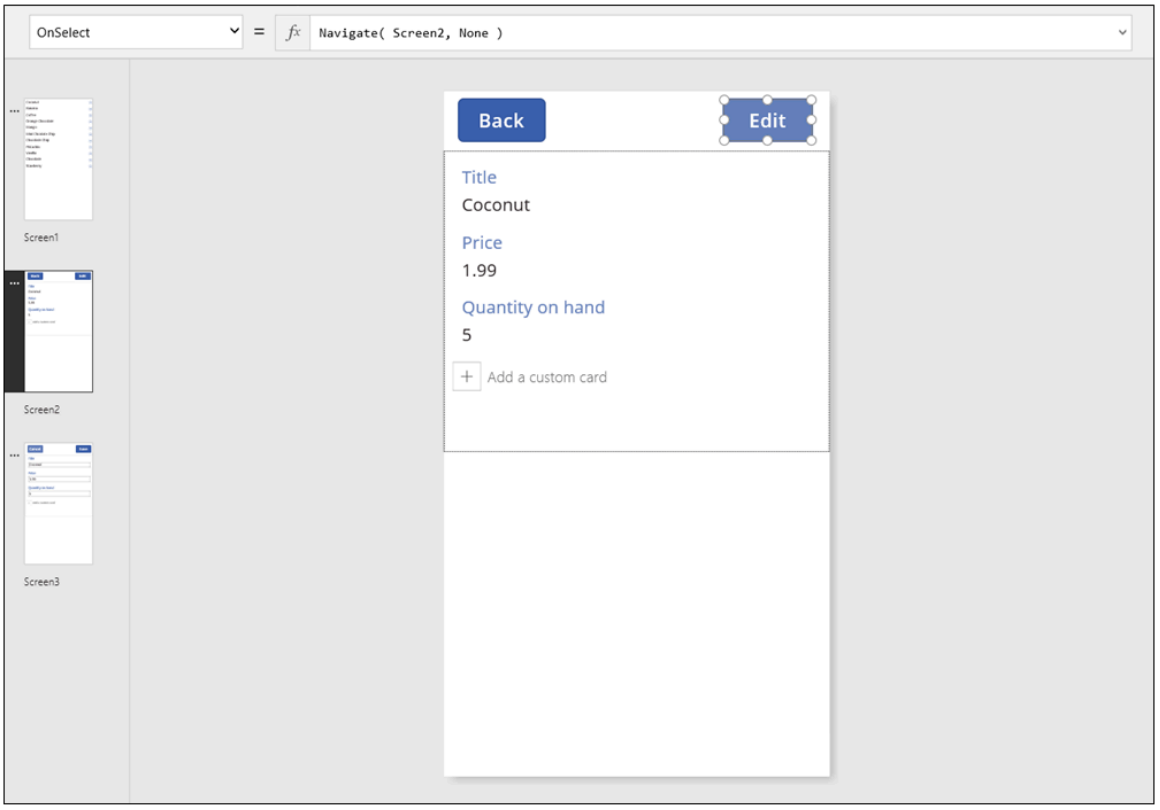
2. Set the **OnSuccess** property of the form to **Back()**.

   When updates are successfully saved, the previous screen (in this case, the details screen) opens automatically.

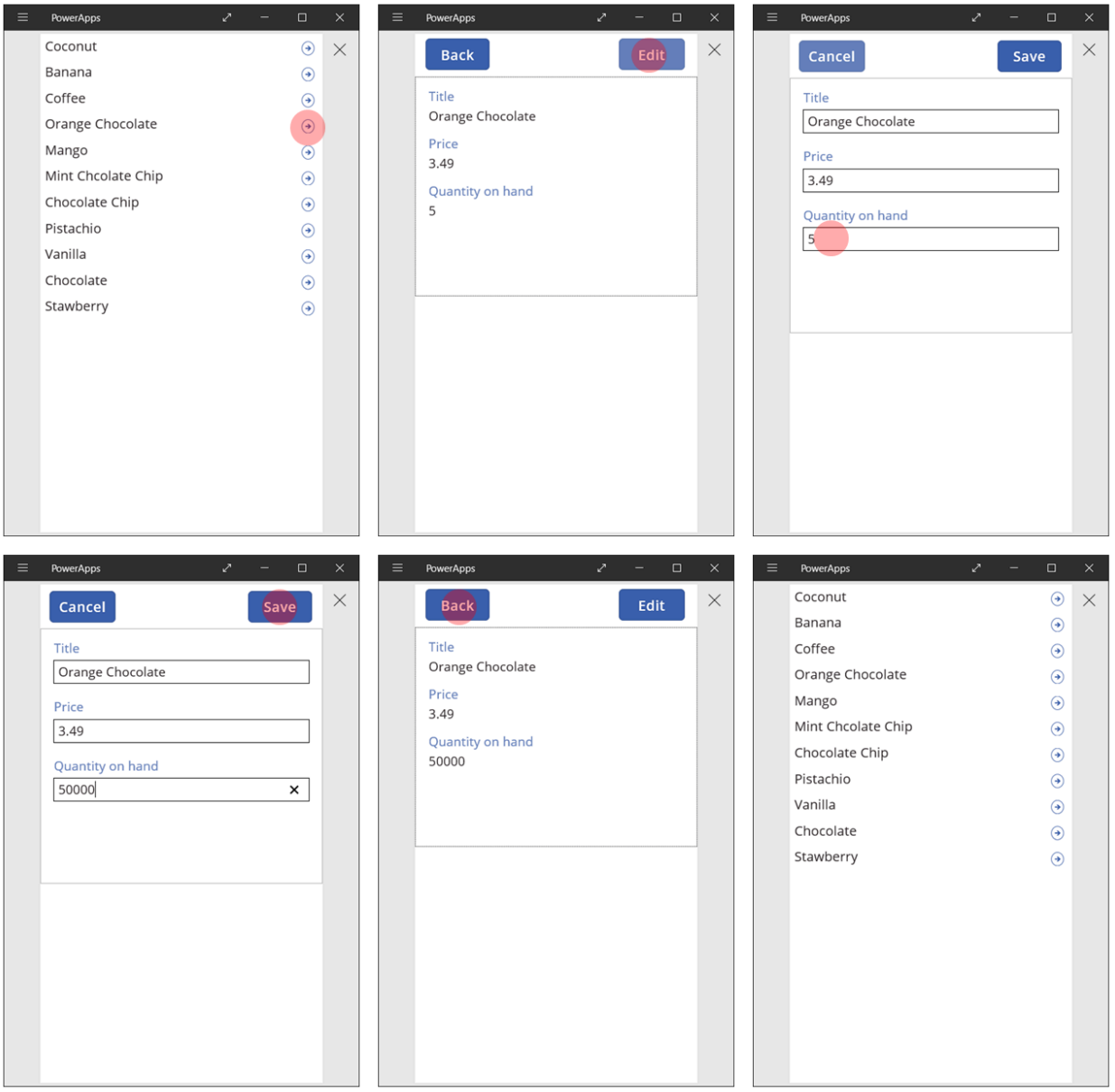


3. On the **Display** screen, add a button, set its **Text** property to show **Edit**, and set its **OnSelect** property to this formula:

Navigate(Screen3, None)



You've built a basic app with three screens for viewing and entering data. To try it out, show the gallery screen, and then press F5 (or select the forward arrow "Preview" button near the upper-left corner of the screen). The pink dot indicates where the user clicks or taps the screen at each step.



Create a record

The user interacts with the same **Edit** form to both update and create records. When the user wants to create a record, the NewForm function switches the form to **New** mode.

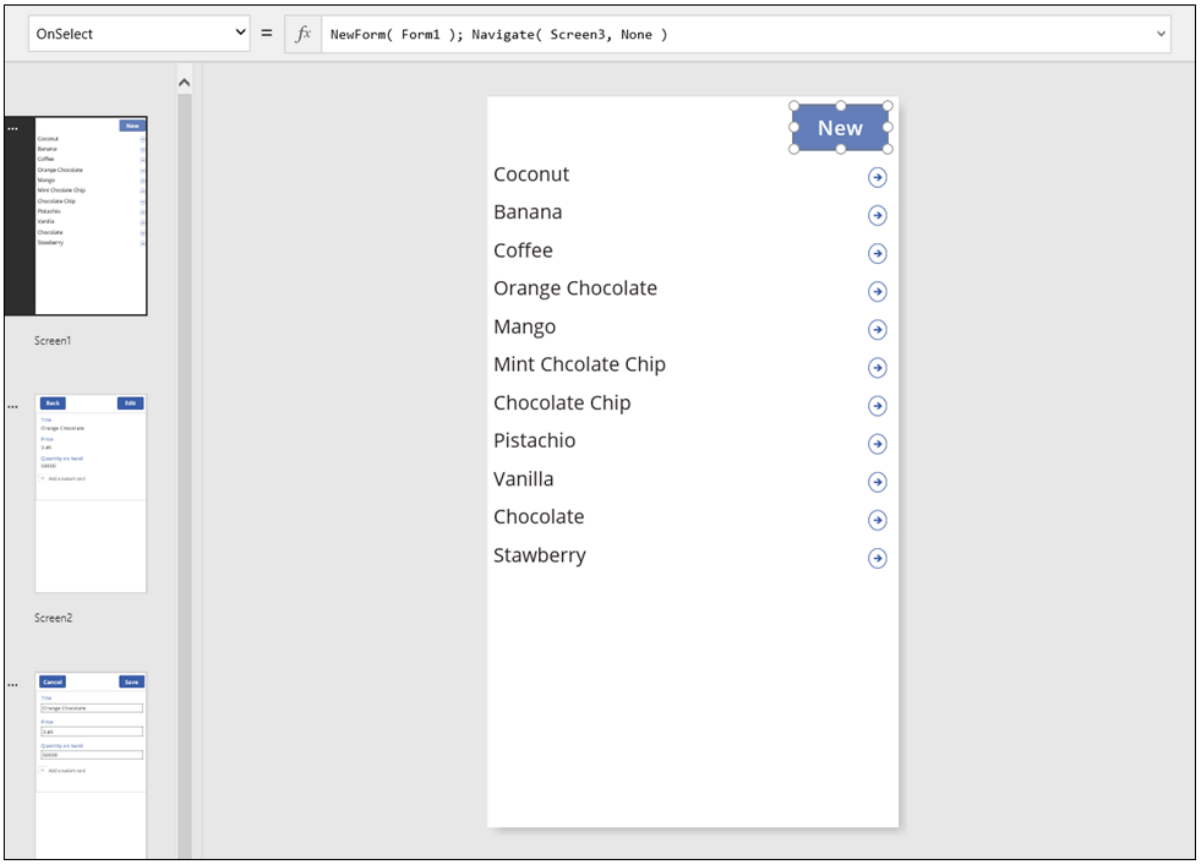
When the form is in **New** mode, the value of each field is set to the defaults of the data source. The record that's provided to the form's Item property is ignored.

When the user is ready to save the new record, SubmitForm runs. After the form is successfully submitted, the form is switched back to **EditMode**.

On the first screen, you'll add a **New** button:

1. On the screen with the gallery, add a Button control.
2. Set the button's Text property to **New** and its OnSelect property to this formula:
   NewForm(Form1); Navigate(Screen3, None)

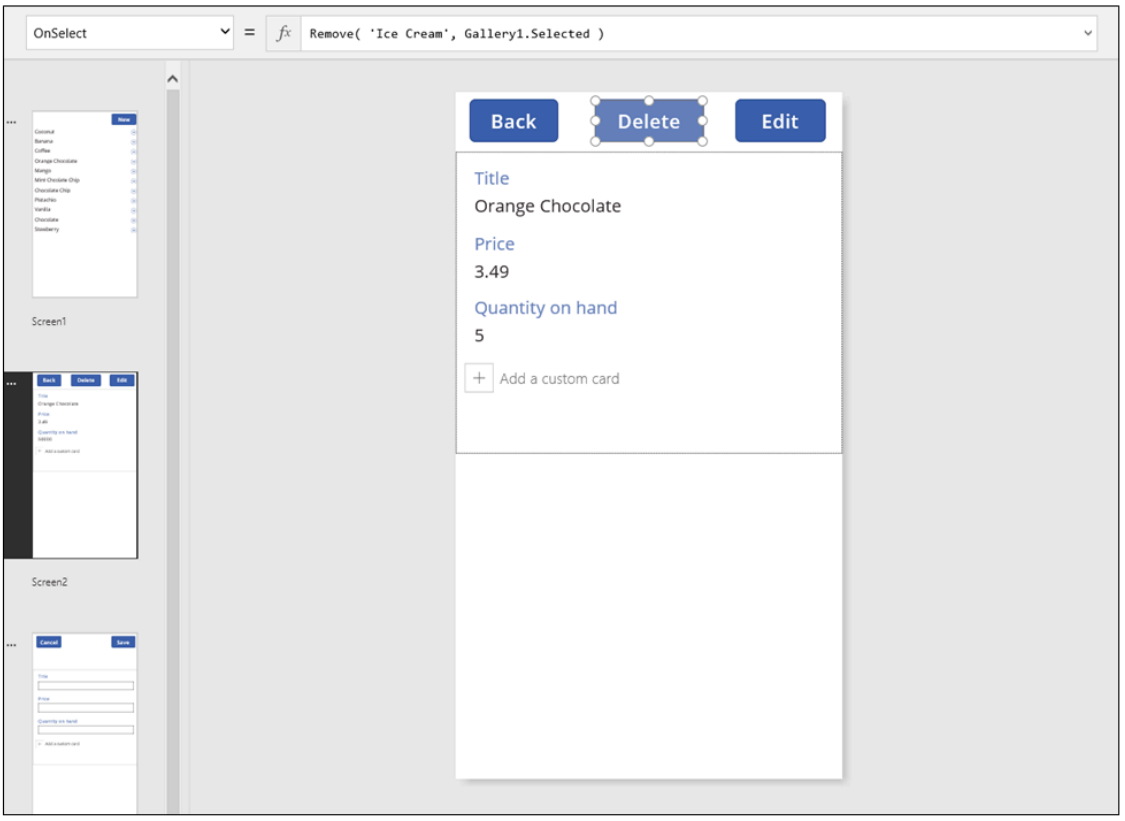
This formula switches the Edit form control on **Screen3** to **New** mode and opens that screen so that the user can fill it in.



When the Edit and Create screen opens, the form is empty, ready for the user to add an item. When the user selects the **Save** button, the SubmitForm function ensures that a record is created instead of being updated. If the user selects the **Cancel** button, the ResetForm function switches the form back to **Edit** mode, and the Back function opens the screen for browsing the gallery.

Delete a record

1. On the **Display** screen, add a button, and set its Text property to show **Delete**..
2. Set the button's OnSelect property to this formula:
   Remove('Ice Cream', Gallery1.Selected); Back()



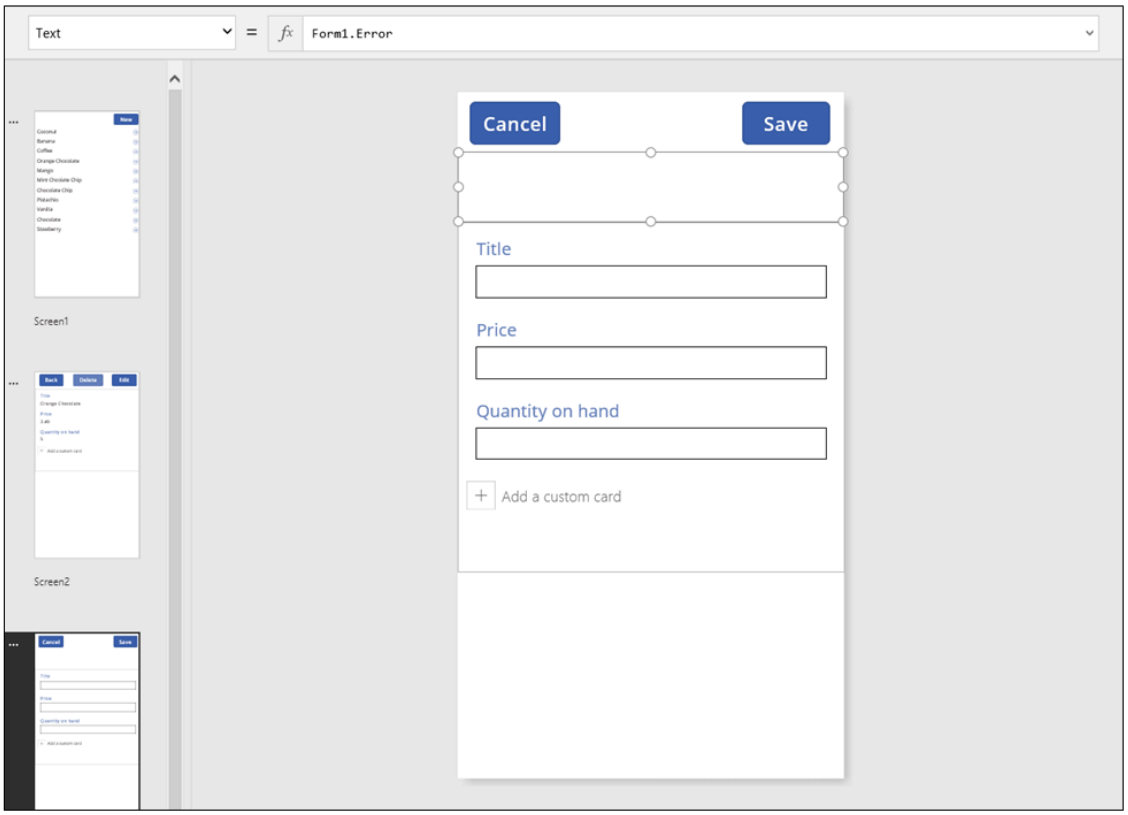
Handling errors

In this app, an error occurs when the value of a field is not valid, a required field is blank, you're disconnected from the network, or any number of other problems pop up.

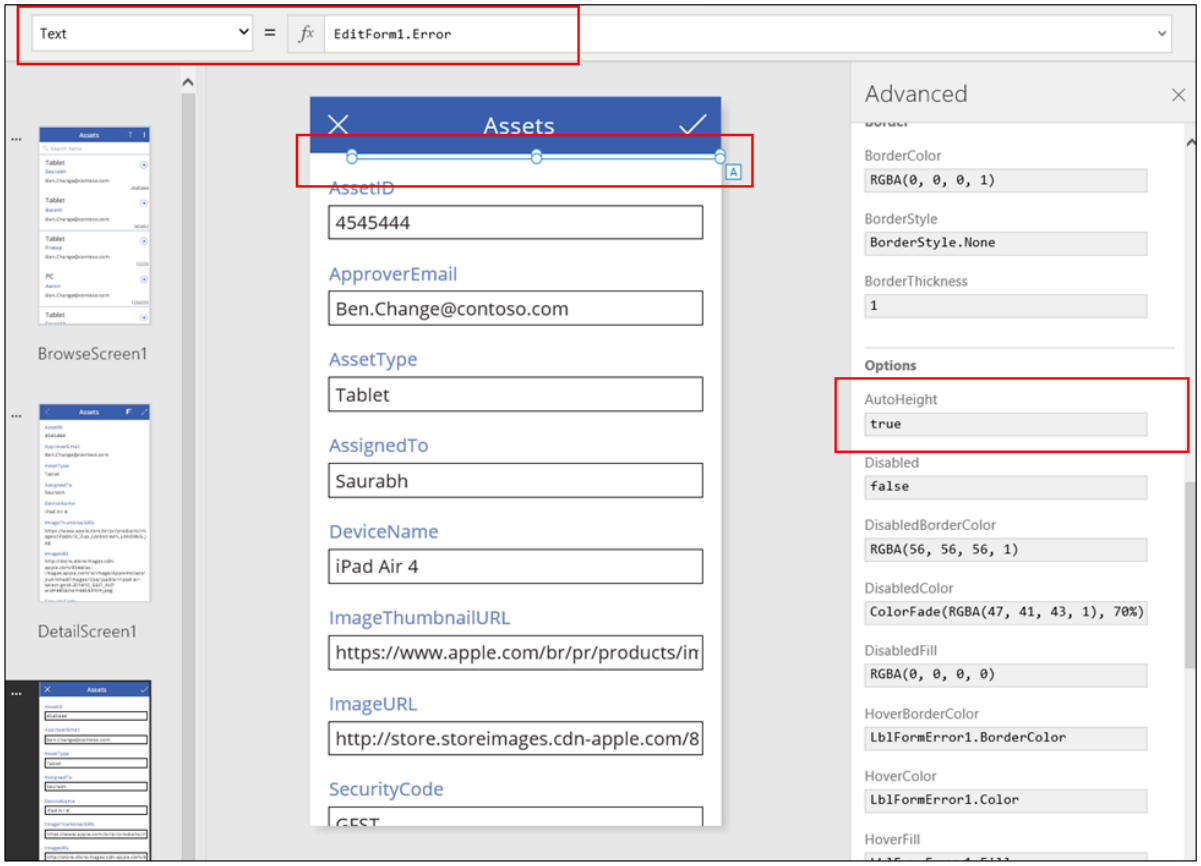
If **SubmitForm** fails for any reason, the **Error** property of the **Edit form** control contains an error message to show the user. With this information, the user should be able to correct the issue and resubmit the change, or they can cancel the update.

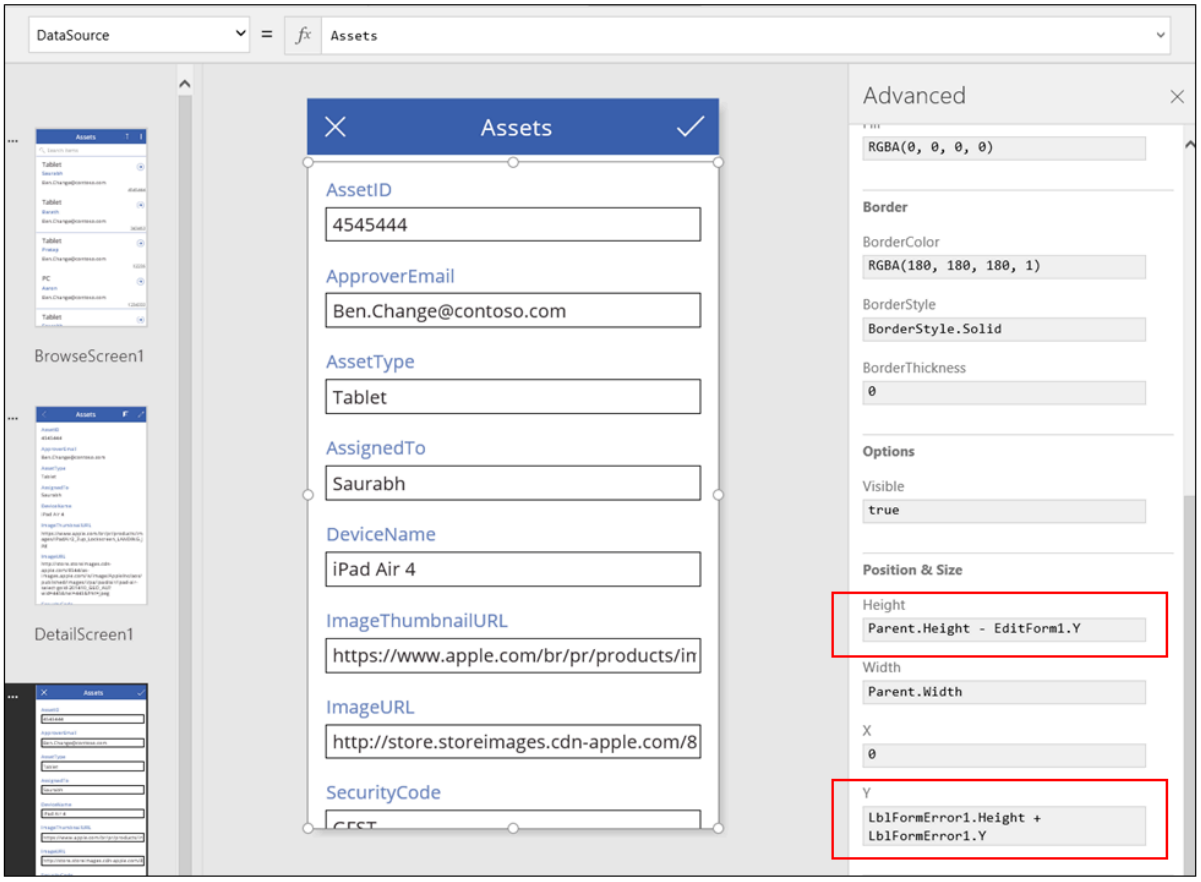
1. On the Edit and Create screen, add a **Label** control, and move it just below the **Save** button. Any error will be easy to see after the user selects this control to save changes.

2. Set the **Text** property of the **Label** control to show **Form1.Error**.



In an app that Power Apps generates from data, the **AutoHeight** property on this control is set to *true* so that no space is consumed if no error occurs. The **Height** and **Y** properties of the **Edit form** control are also adjusted dynamically to account for this control growing when an error occurs. For more details, generate an app from existing data, and inspect these properties. The text-box control for errors is very short when no error has occurred, you may need to open the **Advanced** view (available on the **View** tab) to select this control.



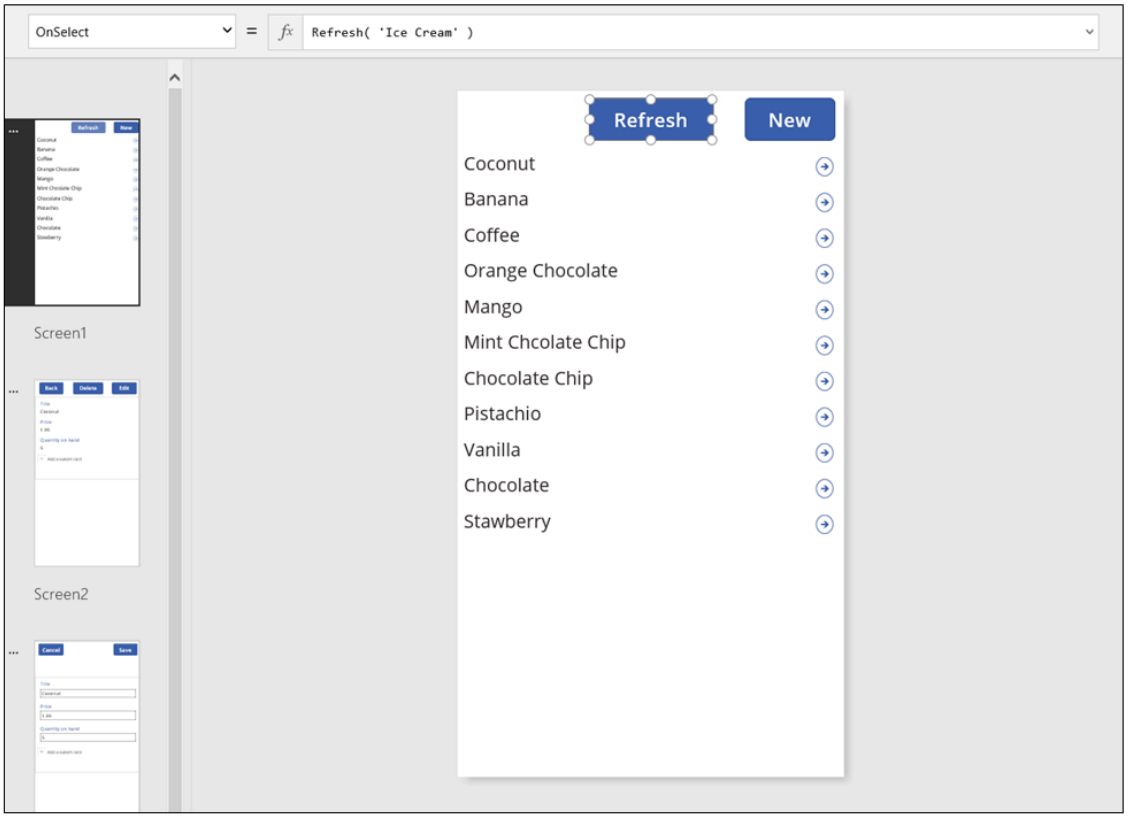


Refresh data

The data source is refreshed whenever the user opens the app, but the user might want to refresh the records in the gallery without closing the app. Add a **Refresh** button so that the user can select it to manually refresh the data:

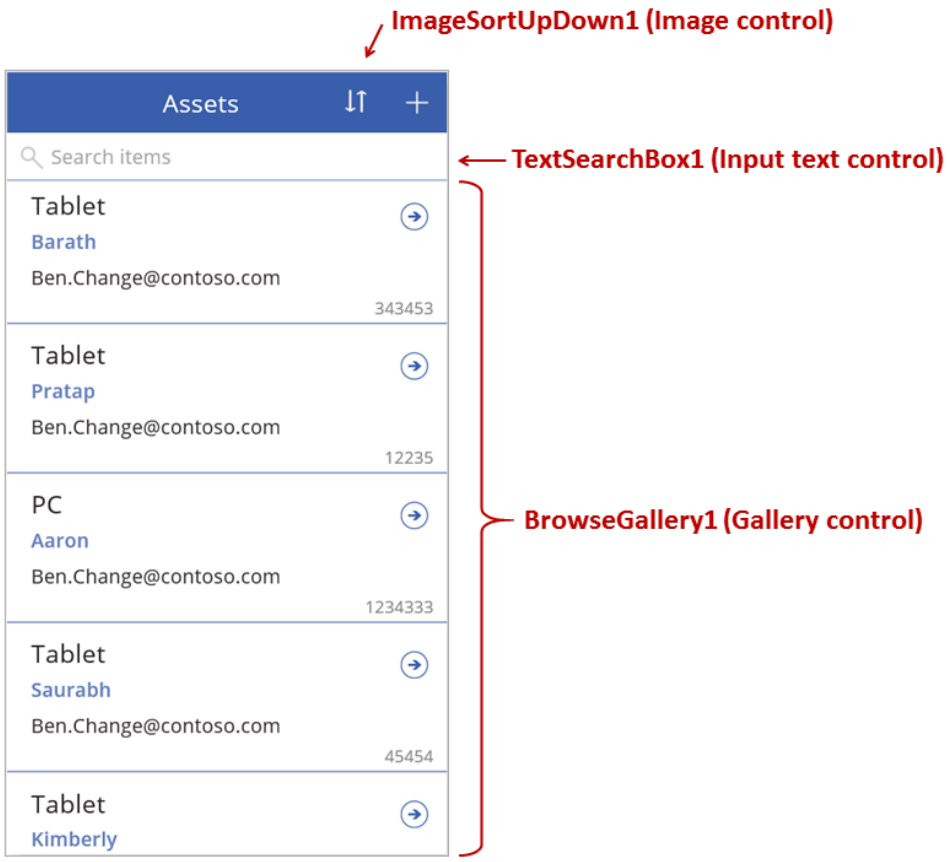
1. On the screen with the **Gallery** control, add a **Button** control and set its **Text** property to show **Refresh**.

2. Set the **OnSelect** property of this control to this formula:
   Refresh('Ice Cream')



Search and sort the gallery

In the app that Power Apps generated from data, we neglected to discuss two controls at the top of the Browse screen. By using these controls, the user can search for one or more records, sort the list of records in ascending or descending order, or both.



When the user selects the sort button, the sort order of the gallery reverses. To create this behavior, we use a *context variable* to track the direction in which the gallery is sorted. When the user selects the button, the variable is updated, and the direction reverses. The OnSelect property of the sort button is set to this formula:
UpdateContext({SortDescending1: !SortDescending1})

The UpdateContext function creates the **SortDescending1** context variable if it doesn't already exist. The function will read the value of the variable and set it to the logical opposite by using the **!** operator. If the value is *true*, it becomes *false*. If the value is *false*, it becomes *true*.

The formula for the Items property of the Gallery control uses this context variable, along with the text in the **TextSearchBox1** control:

Power Fx

```
Sort(
    If( IsBlank(TextSearchBox1.Text),
        Assets,
        Filter( Assets, TextSearchBox1.Text in Text(ApproverEmail) )
    ),
    ApproverEmail,
    If(SortDescending1, Descending, Ascending)
)
```

Let's break this down:

- On the outside, we have the **Sort** function, which takes three arguments: a table, a field on which to sort, and the direction in which to sort.
  - The sort direction is taken from the context variable that toggles when the user selects the **ImageSortUpDown1** control. The *true/false* value is translated to the constants **Descending** and **Ascending**.
  - The field to sort on is fixed to **ApproverEmail**. If you change the fields that appear in the gallery, you'll need to change this argument too.

- On the inside, we have the **Filter** function, which takes a table as an argument and an expression to evaluate for each record.
  - The table is the raw **Assets** data source, which is the starting point before filtering or sorting.
  - The expression searches for an instance of the string in **TextSearchBox1** within the **ApproverEmail** field. Again, if you change the fields that appear in the gallery, you'll also need to update this argument.
  - If **TextSearchBox1** is empty, the user wants to show all records, and the **Filter** function is bypassed.

This is but one example; you can craft your own formula for the **Items** property, depending on the needs of your app, by composing **Filter**, **Sort**, and other functions and operators together.

Screen design

So far, we haven't discussed other ways to distribute controls across screens. That's because you have many options, and the best selection depends on your specific app's needs.

Because real estate on phone screens is so limited, you probably want to browse, display, and edit/create on different screens. In this topic, the **Navigate** and **Back** functions open each screen.

On a tablet, you can browse, display, and edit/create on two or even one screen. For the latter, no **Navigate** or **Back** function would be required.

If the user is working on the same screen, you need to be careful that the user can't change the selection in the **Gallery** and potentially lose edits in the **Edit form** control. To keep the user from selecting a different record when changes to another record haven't been saved yet, set the **Disabled** property of the gallery to this formula:
EditForm.Unsaved

Feedback

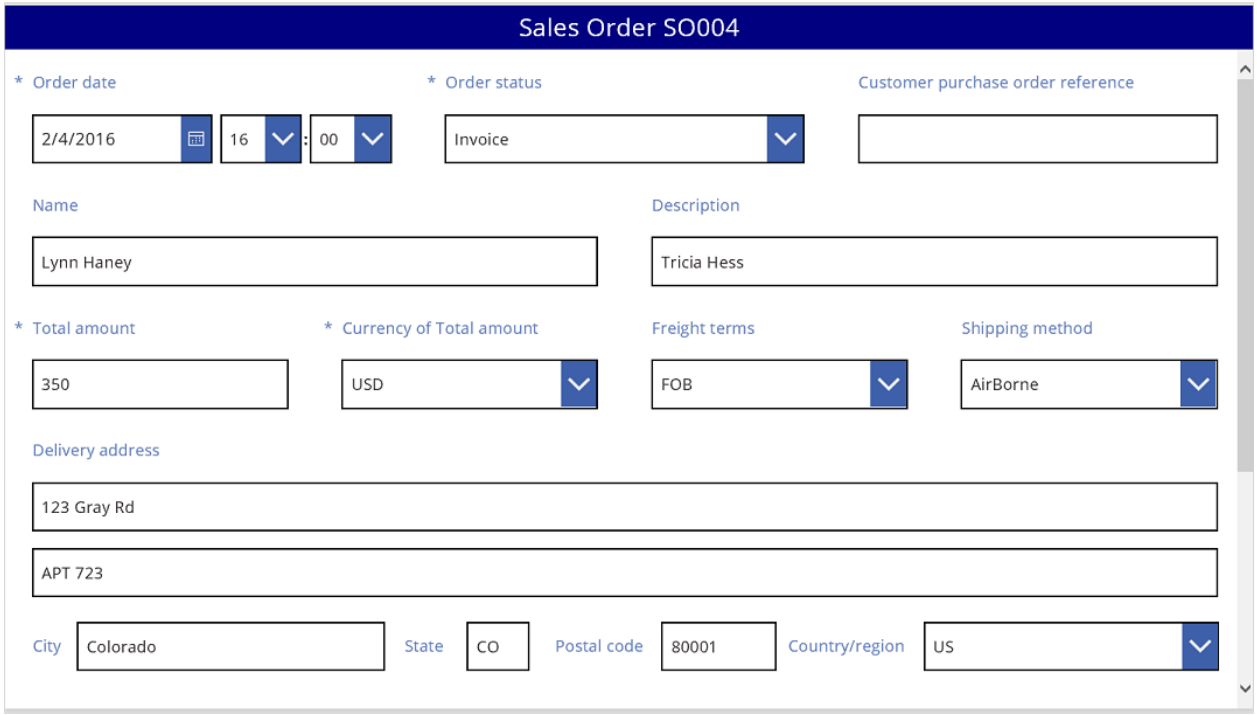
Was this page helpful? Yes No

Provide product feedback

Understand data-form layout for canvas apps

Article • 12/16/2022

Easily create an attractive and efficient form when you build a canvas app in Power Apps. For example, consider this basic form for recording sales orders:



In this tutorial, we'll walk through the steps to create this form. We'll also look at some advanced topics, such as dynamic sizing of fields to fill available space.

Before you start

If you're new to Power Apps (or have only generated apps automatically), you'll want to build an app from scratch before you dive into this article. By building an app from scratch, you'll become familiar with required concepts, such as adding data sources and controls, that are mentioned but not explained in this article.

This article is written as though you have a data source that's named **Sales order** and that contains the fields in the previous graphic. If you have a Power Apps per user, per app, or trial license and system administrator or system customizer permissions, you can create a table in Microsoft Dataverse and add similar fields.

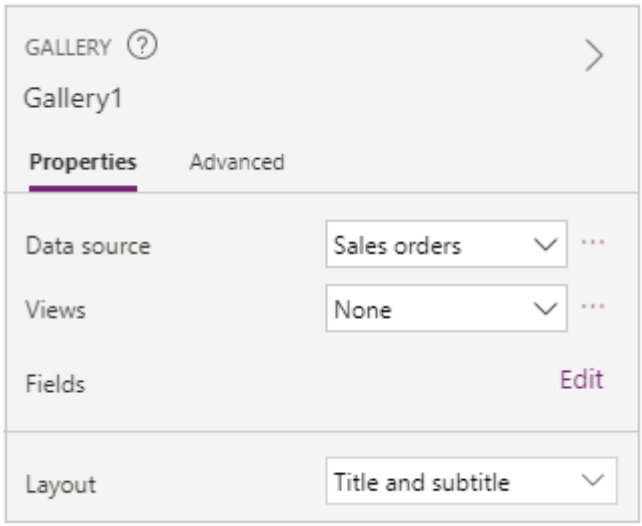
Add a gallery

1. Create a tablet app from scratch, and add your data source.

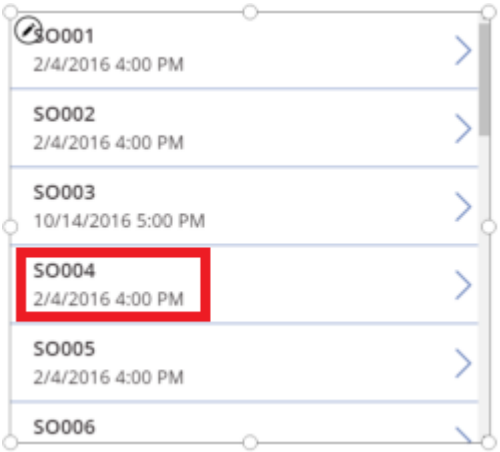
   Everything discussed in this article also applies to phone layouts, but phone apps often have only one vertical column.

2. Add a vertical **Gallery** control, and set its **Items** property to '**Sales order**'.

   (optional) To match the examples in this tutorial, change the gallery's **Layout** to show only **Title and subtitle**.



3. In the gallery, click or tap **SO004**.



   This record will appear in the form that you build by following steps later in this article.

Add a title bar

1. Add a blank screen where you'll put the form.

   Outside of this tutorial, you can put the **Gallery** and **Edit form** controls on the same screen, but you'll have more room to work with if you put them on separate

screens.

2. At the top of the new screen, add a **Label** control, and set its **Text** property to this expression:
   "Sales Order " & Gallery1.Selected.SalesOrderId

   The label shows the sales-order number of the record that you selected in the gallery.

3. (optional) Format the label as follows:

   a. Set its **Align** property to **Center**.

   b. Set its **Size** property to **20**.

   c. Set its **Fill** property to **Navy**.

   d. Set its **Color** property to **White**.

   e. Set its **Width** property to **Parent.Width**.

   f. Set its **X** and **Y** properties to **0**.



Add a form

1. Add an **Edit form** control, and then move and resize it to fill the screen under the label.

   In the next step, you'll connect the form control to the **Sales order** data source by using the right-hand pane, not the formula bar. If you use the formula bar, the form won't show any fields by default. You can always show any fields that you want by selecting one or more check boxes in the right-hand pane.

2. In the right-hand pane, click or tap the down arrow next to **No data source selected**, and then click or tap **Sales order**.

   A default set of fields from the **Sales order** data source will appear in a simple, three-column layout. However, many are blank, and they may take a few moments to settle into their final positions.

3. Set the form's **Item** property to **Gallery1.Selected**.

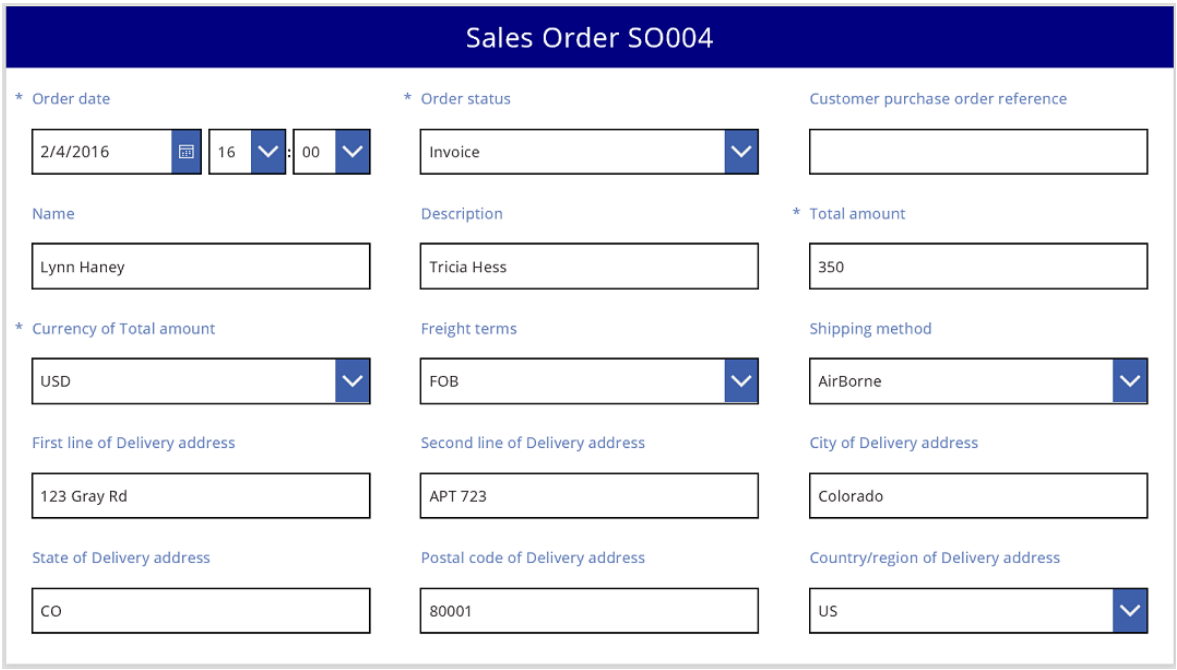
The form shows the record that you selected in the gallery, but the default set of fields might not match what you want in your final product.

4. In the right-hand pane, hide each of these fields by clearing its checkbox:

   - **Sales order ID**
   - **Account**
   - **Sales person**
   - **Account contact**

5. Move the **Order status** field by dragging it to the left and then dropping it on the other side of the **Customer purchase order reference** field.

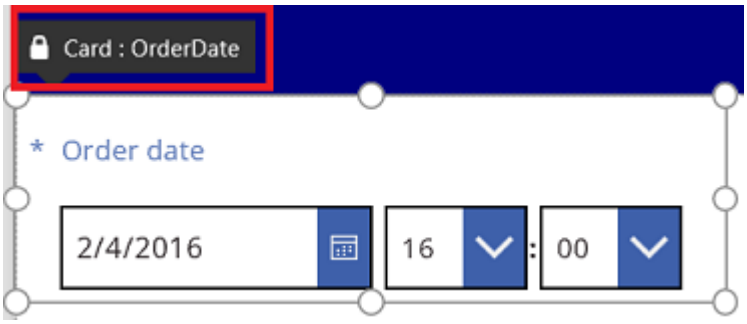
   Your screen should resemble this example:



Select a data card

Each field displayed has a corresponding data card on the form. This card includes a set of controls for the field title, an input box, a star (which appears if the field is required), and a validation error message.

You can also select cards directly on the form. When a card is selected, a black caption appears above it.

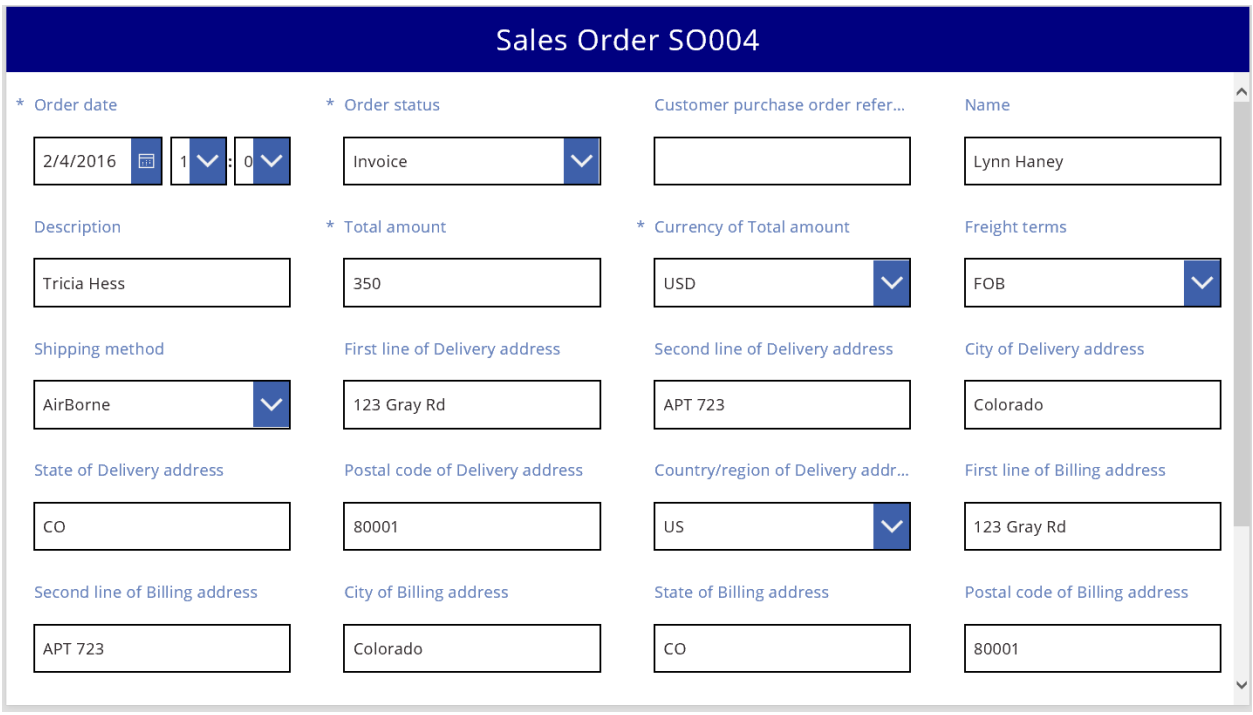


> **Note**
>
> To delete a card (not just hide it), select it, and then press Delete.

Arrange cards in columns

By default, forms in tablet apps have three columns, and in phone apps have one. You can specify not only how many columns a form has but also whether all cards should fit within column borders.

In this graphic, the number of columns in the form was changed from three to four with the **Snap to columns** check box selected. The cards in the form were arranged automatically to fit the new layout.



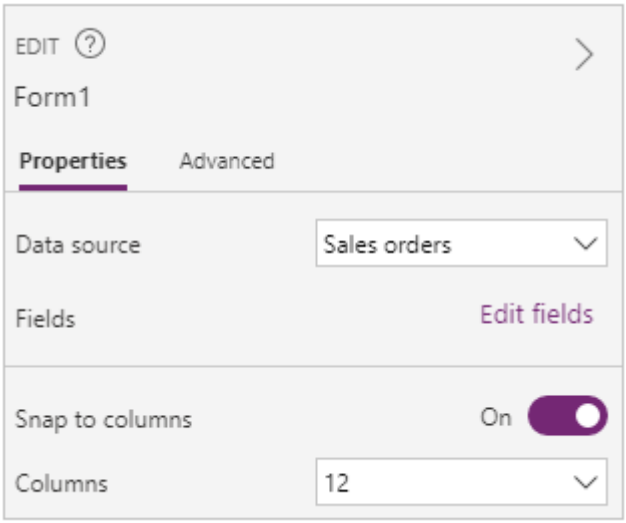
Resize cards across multiple columns

Depending on the data in each card, you might want some cards to fit in a single column and other cards to span multiple columns. If a card contains more data than you

want to show in a single column, you can widen the card by selecting it and then dragging the grab handle on the left or right border of its selection box. As you drag the handle, the card will "snap" to column boundaries.

To make your design more flexible but keep some structure, you can increase the number of columns to 12. With that change, you can easily configure each card to span the entire form, half of the form, one-third, one-quarter, one-sixth, and so forth. Let's see this in action.

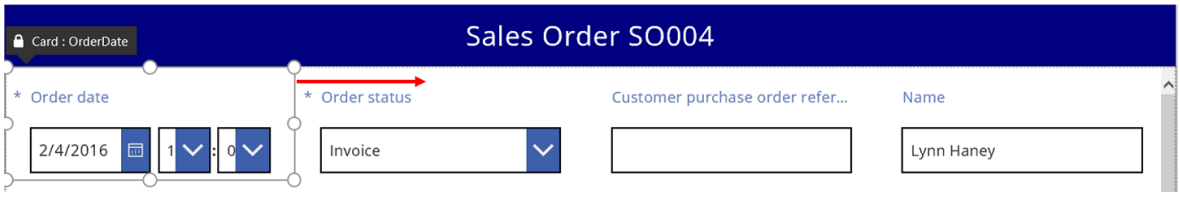
1. In the right-hand pane, set the number of columns in the form to **12**.



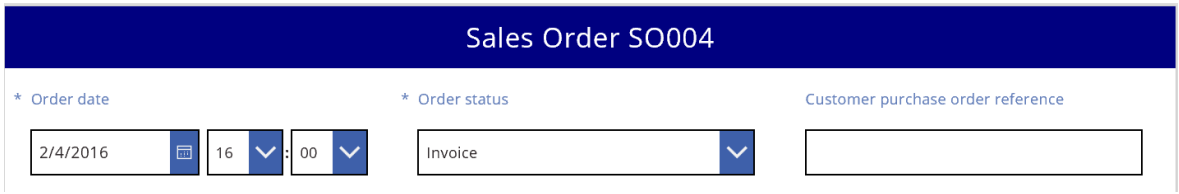
   The form doesn't visibly change, but you have more snap points as you drag the left or right grab handle.

2. Increase the width of the **Order date** card by dragging the grab handle on the right one snap point to the right.

   The card spans four of the form's 12 columns (or 1/3 of the form), instead of only three of the form's 12 columns (or 1/4 of the form). Whenever you increase a card's width by one snap point, the card spans an additional 1/12 of the form.



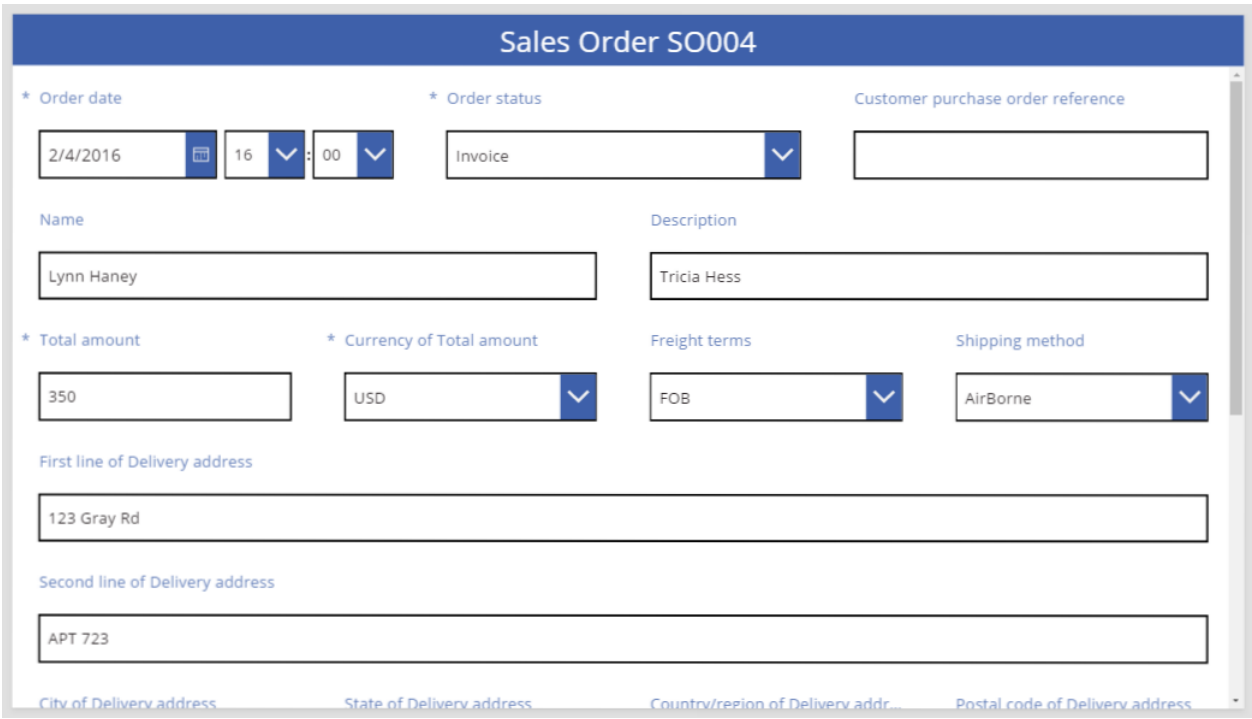
3. Repeat the previous step with the **Order status** and **Customer purchase order reference** cards.



4. Resize the **Name** and **Description** cards to take up six columns (or 1/2) of the form.

5. Make the first two lines of the delivery address stretch entirely across the form:

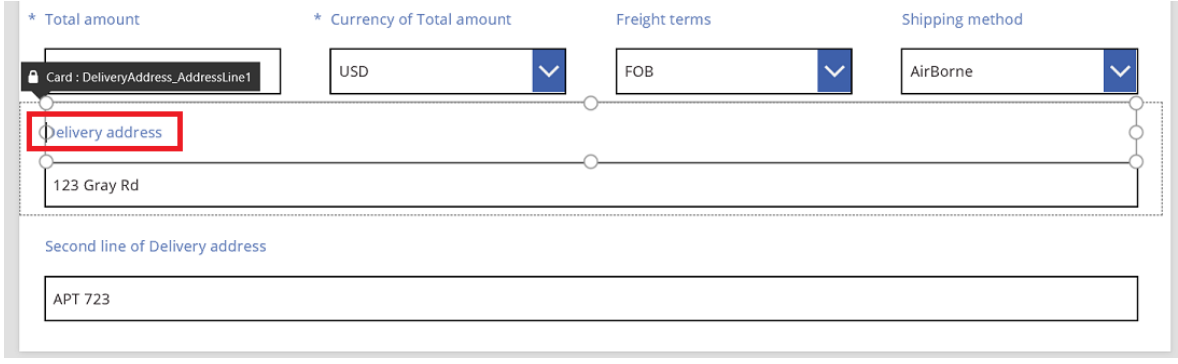
All done. We have our desired form, mixing rows with different numbers of columns:



Manipulate controls in a card

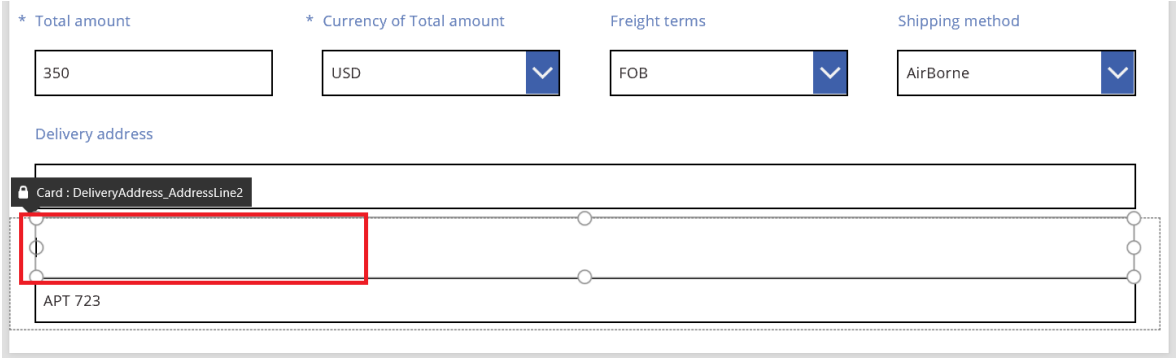
The delivery address includes several pieces of information that we want to visually group together for the user. Each field will remain in its own data card, but we can manipulate the controls within the card to make them fit better together.

1. Select the **First line of Delivery address** card, select the label within that card, and then delete the first three words from the text.



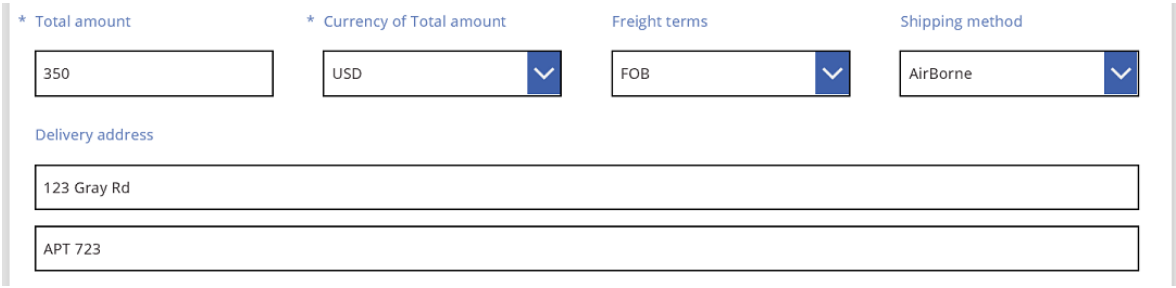
2. Select the **Second line of Delivery address** card, select the label within that card, and then delete all of the text in it.

   It may be tempting to remove the label control and, in many cases, that will work fine. But formulas might depend on that control being present. The safer approach is to remove the text or to set the **Visible** property of the control to **false**.

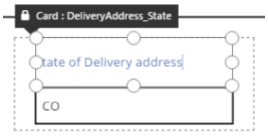


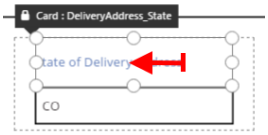
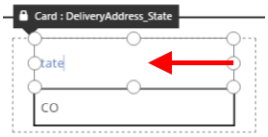
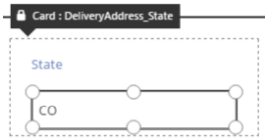
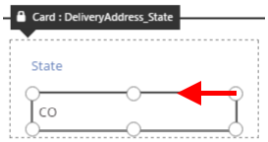
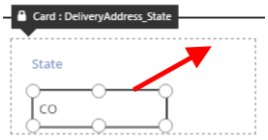
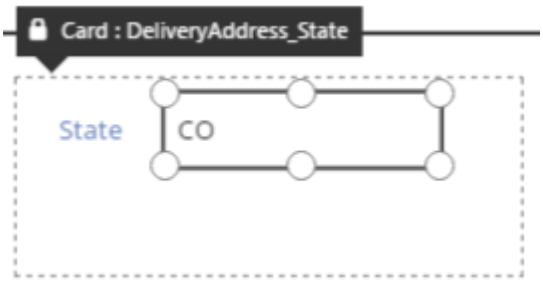
3. In the same card, move the text input box over the label to reduce the space between the first and second lines of the address.

   The height of the card shrinks when its contents take up less space.

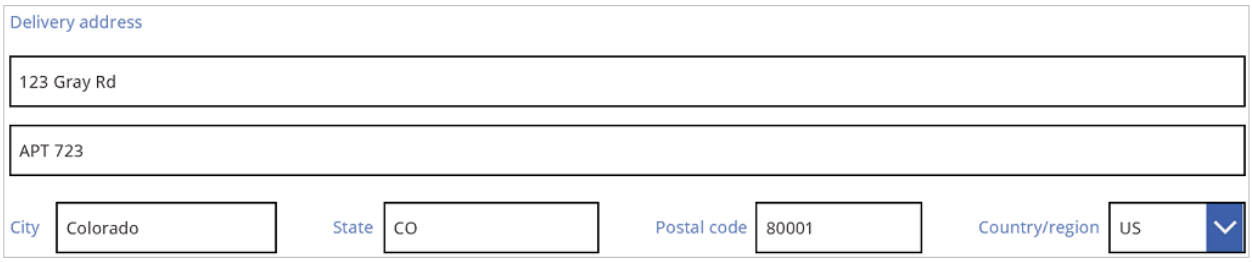


Now let's turn our attention to the third line of the address. Similar to what we just did, let's shorten the text of each label for these cards and arrange the Text input box to be to the right of each label. Here are the steps for the **State** card:

Step	Description	Result
1	Select the **State** card so that grab handles appear around it.	
2	Select the label within this card so that grab handles appear around it.	

Step	Description	Result
3	Place the cursor to the right of the text, and then delete the portion that we don't need.	
4	Using the grab handles on the sides, size the label control to fit the new text size.	
5	Select the text input control within this card.	
6	Using the grab handles on the sides, size the text input control to the size that you want.	
7	Drag up the text input box and to the right of the label control, and then drop the text input box.	
Our modifications to the **State** card are now complete.		

The result for the complete third address line:



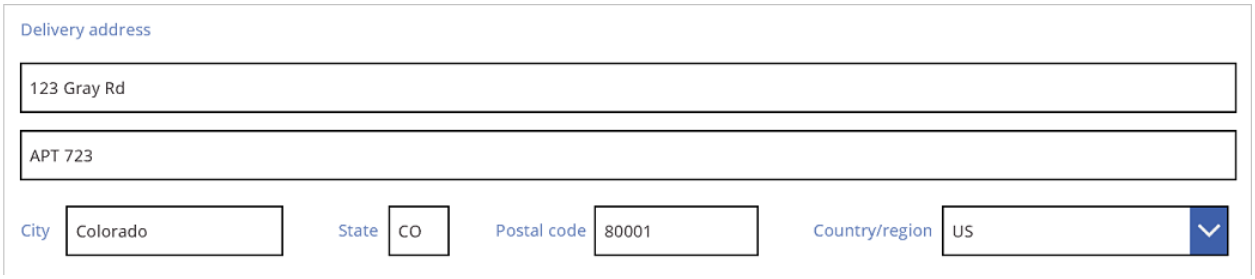
Many of the cards start out with dynamic formulas for their properties. For example, the Text input control that we resized and moved above had a **Width** property based on the width of its parent. When you move or resize a control, these dynamic formulas are replaced with static values. If you want, you can restore the dynamic formulas by using the formula bar.

Turning off Snap to columns

Sometimes you'll want finer control than the standard 12 columns can provide. For these cases, you can turn off **Snap to columns** and then position cards manually. The form will continue snapping to 12 columns, but you can also hold down the Alt or Ctrl+Shift keys after starting a resize or reposition to override the snap points. For more information, see alternate behavior keyboard shortcuts.

In our example, the four components that make up the third line of the address all have exactly the same width. But this may not be the best layout, as city names are longer than state abbreviations, and the Text input box for countries/regions is short because of the length of its label. To optimize this space, turn off **Snap to columns** in the right-hand pane and then hold down the Alt or Ctrl+Shift keys after starting to size and position these cards.

After careful positioning, the result has appropriate sizes for each field and even spacing horizontally between fields:



In summary, what are the differences when **Snap to columns** is on versus off?

Behavior	**Snap to columns On**	**Snap to columns Off**
Resize snaps to	Number of columns you select: 1, 2, 3, 4, 6, or 12	12 columns
Resize snap can be overridden	No	Yes, with Alt or Ctrl+Shift keys after starting the resize
Cards automatically relayout between rows (more on this later)	Yes	No

Set width and height

As with everything in Power Apps, the form's layout is governed by properties on the card controls. As already described, you can change the values of these properties by dragging controls to different locations or dragging grab handles to resize controls. But you'll discover situations in which you'll want to understand and manipulate these properties more precisely, especially when making your forms dynamic with formulas.

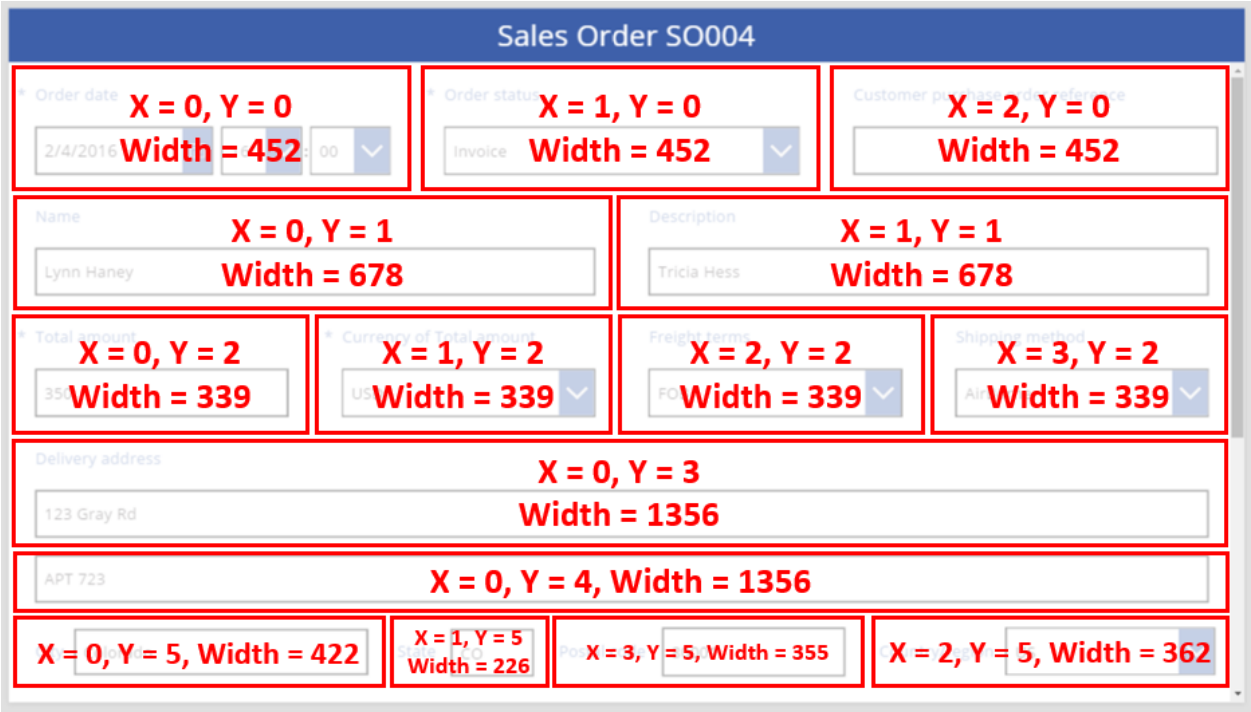
Basic Layout: X, Y, and Width

The **X** and **Y** properties control the position of cards. When we work with controls on the raw canvas, these properties provide an absolute position. In a form, these properties have a different meaning:

- **X**: Order within a row.
- **Y**: Row number.

Similar to controls on the canvas, the **Width** property specifies the minimum width of the card (more on the minimum aspect in a moment).

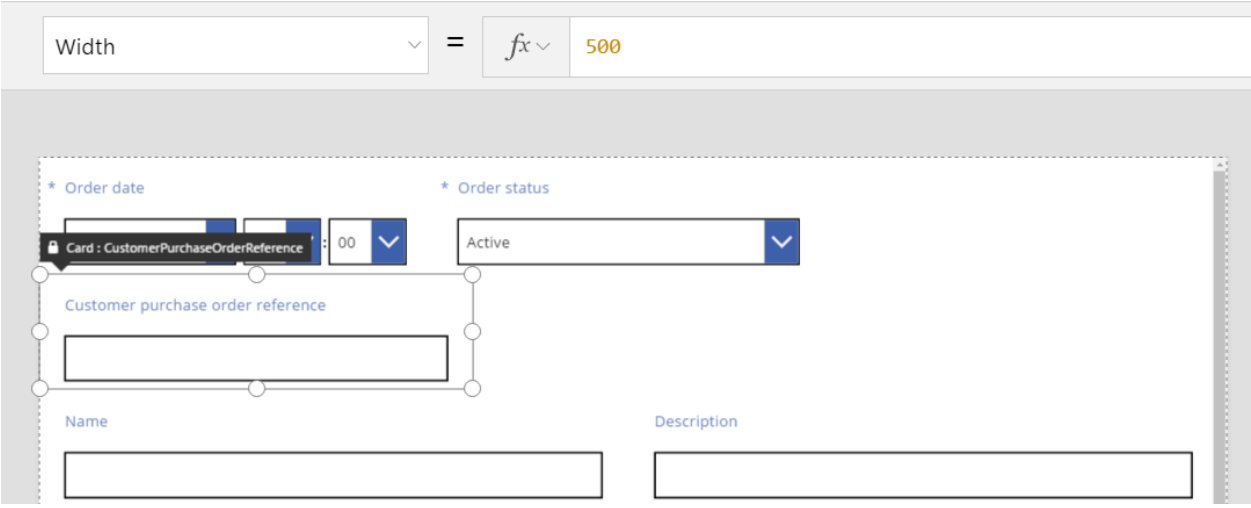
Let's take a look at the **X**, **Y**, and **Width** properties of the cards in our form:



Overflowing rows

What happens if the cards on a row are too wide to fit on that row? Normally you don't need to worry about this possibility. With **Snap to columns** on, these three properties will automatically be adjusted so that everything fits nicely within rows without overflowing.

But with **Snap to columns** turned off or a formula-based **Width** on one or more of your cards, overflowing a row can happen. In this case, the cards will automatically wrap so that, effectively, another row is created. For example, let's manually change the **Width** property of our **Customer purchase order reference** card (first row, third item) to **500**:



The three cards on the top row no longer fit horizontally, and another row has been created to wrap the overflow. The **Y** coordinate for all these cards is still the same at 0, and the **Name** and **Description** cards still have a **Y** of 1. Cards that have different **Y** values aren't merged across rows.

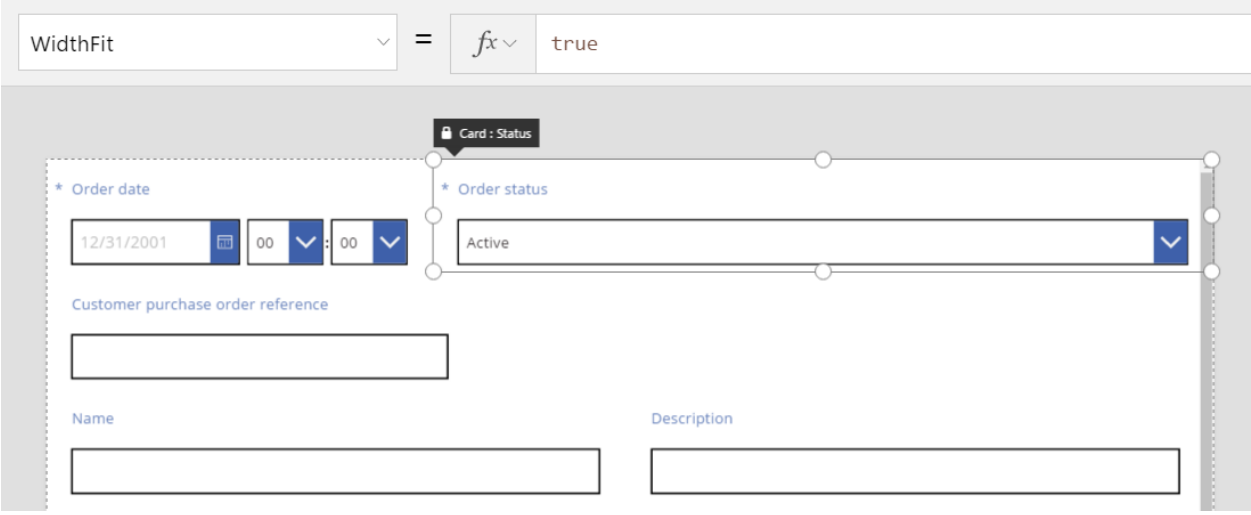
You can use this behavior to create a fully dynamic layout, where cards are placed based on a Z-order, filling across as much as possible before moving to the next row. To achieve this effect, give all the cards the same **Y** value, and use **X** for the order of the cards.

Filling spaces: WidthFit

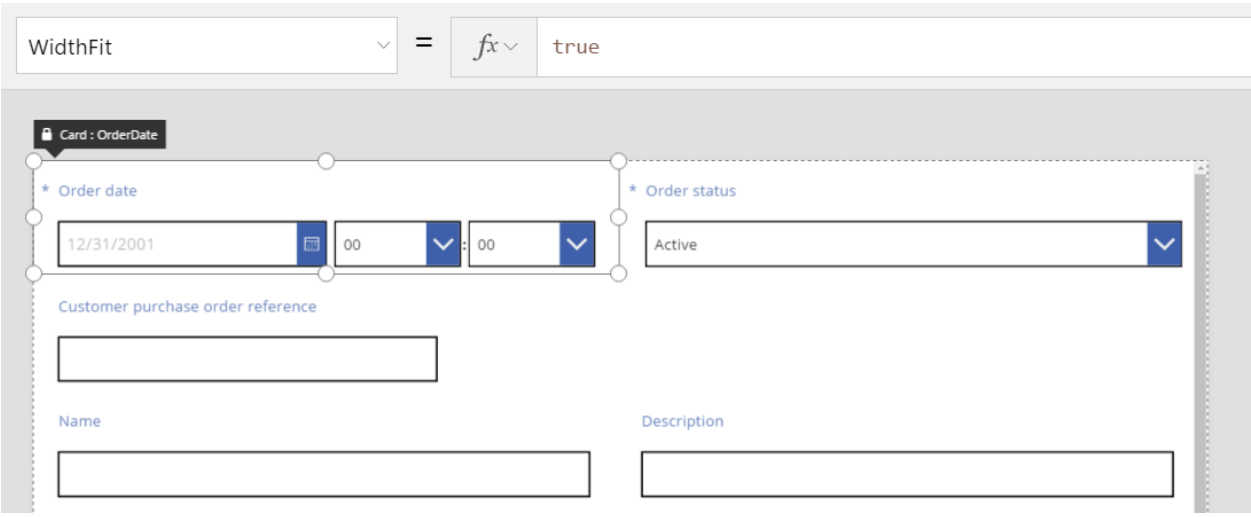
The overflow in the last example created a space after the **Order status** card, which was the second card in the first row. We could manually adjust the **Width** properties of the two remaining cards to fill this space, but this approach is tedious.

As an alternative, use the **WidthFit** property. If this property is set to **true** for one or more cards in a row, any remaining space on the row will be evenly divided between them. This behavior is why we said earlier that the **Width** property of a card is a *minimum*, and what is seen can be wider. This property will never cause a card to shrink, only expand.

If we set **WidthFit** to **true** on the **Order status** card, it fills the available space, while the first card remains unchanged:



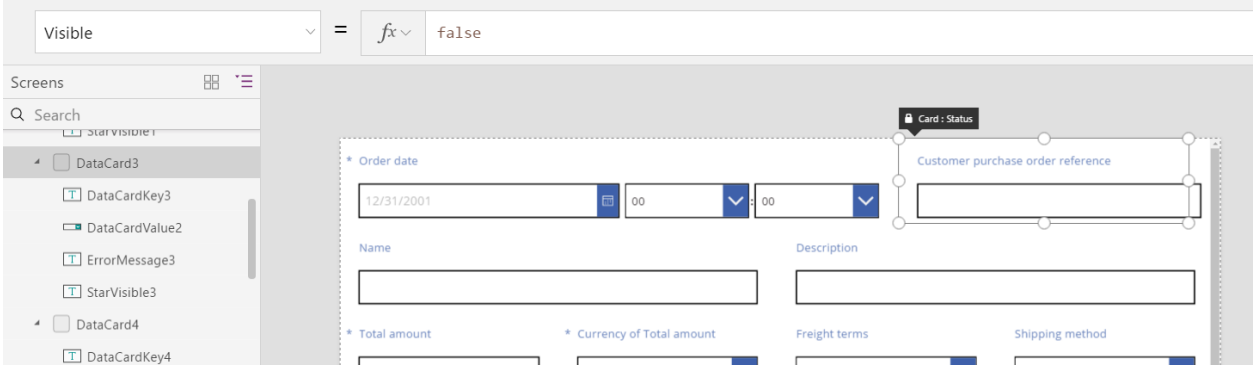
If we also set **WidthFit** to **true** on the **Order date** card, both cards will evenly split the available space:



Grab handles on these cards take into account the extra width provided by **WidthFit**, not the minimum width provided by the **Width** property. It can be confusing to manipulate the **Width** property while **WidthFit** is turned on; you may want to turn it off, make changes to **Width**, and then turn it back on.

When might **WidthFit** be useful? If you have a field that is used only in certain situations, you can set its **Visible** property to **false**, and the other cards on the row will automatically fill the space around it. You might want to use a formula that shows a field only when another field has a particular value.

Here, we'll set the **Visible** property of the **Order status** field to a static **false**:



With the second card effectively removed, the third card can now return to the same row as the first card. The first card still has **WidthFit** set to **true**, so it alone expands to fill the available space.

Because **Order status** is invisible, you can't select it as easily on the canvas. However, you can select any control, visible or not, in the hierarchical list of controls on the left side of the screen.

Height

The **Height** property governs the height of each card. Cards have the equivalent of **WidthFit** for **Height**, and it's always set to **true**. Imagine that a **HeightFit** property exists, but don't look for it in the product because such a property isn't yet exposed.

You can't turn off this behavior, so changing the heights of cards can be challenging. All cards within a row appear to be the same height as the tallest card. You might look at a row like this:

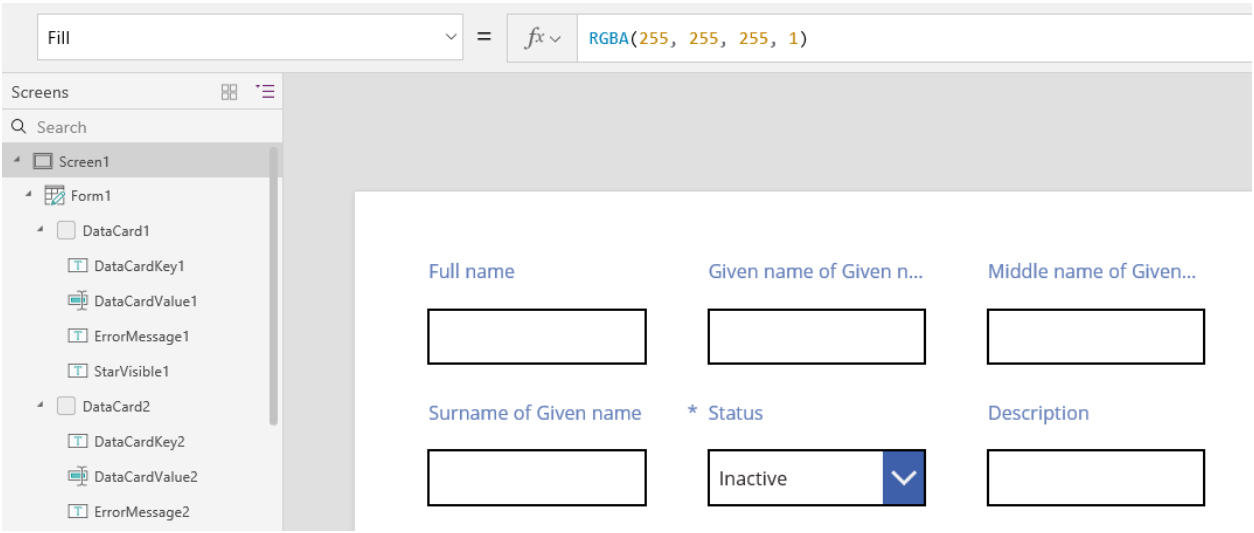


Which card is making the row tall? In the previous graphic, the **Total amount** card is selected and appears tall, but its **Height** property is set to **80** (same as the height of the first row). To reduce the height of a row, you must reduce the **Height** of the tallest card in that row, and you can't identify the tallest card without reviewing the **Height** property of each card.

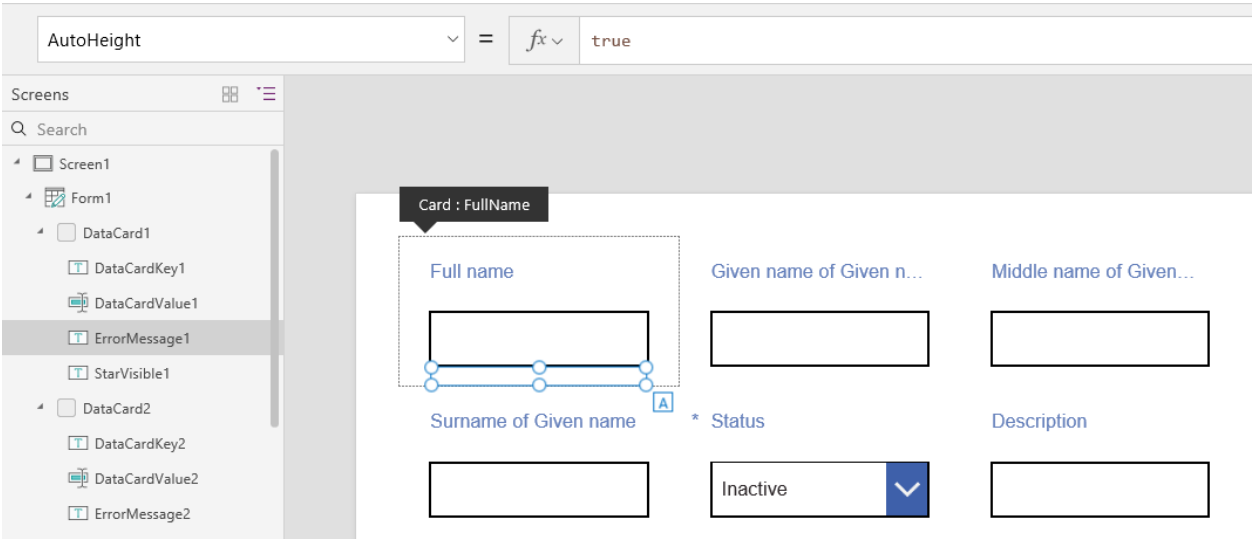
AutoHeight

A card might also be taller than you expect if it contains a control for which the **AutoHeight** property is set to **true**. For example, many cards contain a label that displays an error message if the field's value causes a validation problem.

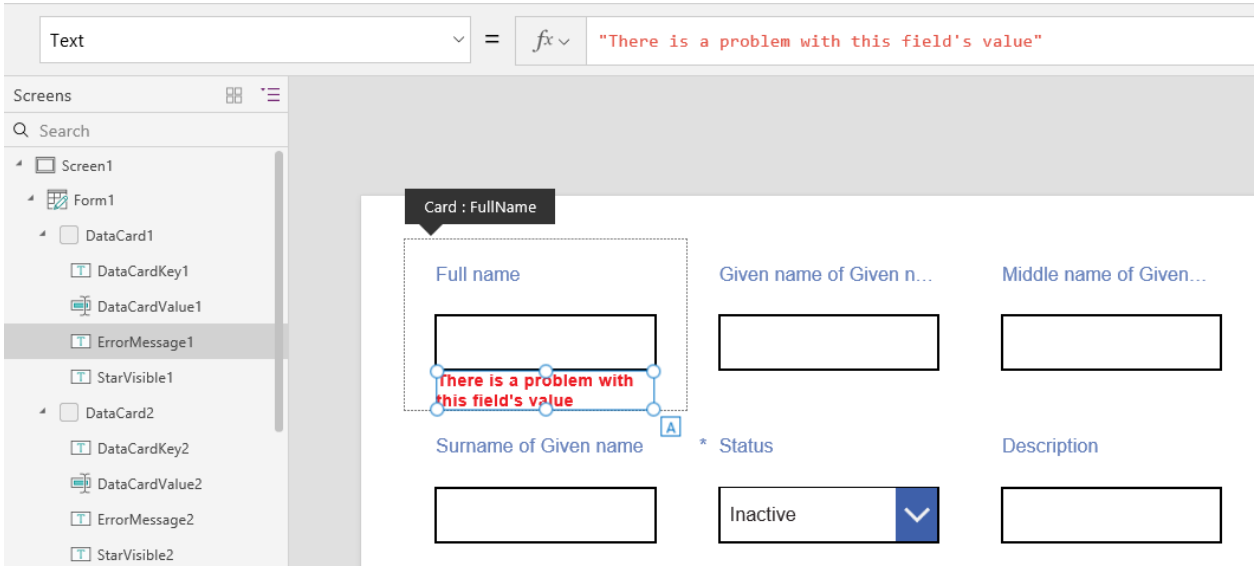
Without any text to display (no error), the label collapses to zero height. If you didn't know any better, you wouldn't know it was there, and that's as it should be:



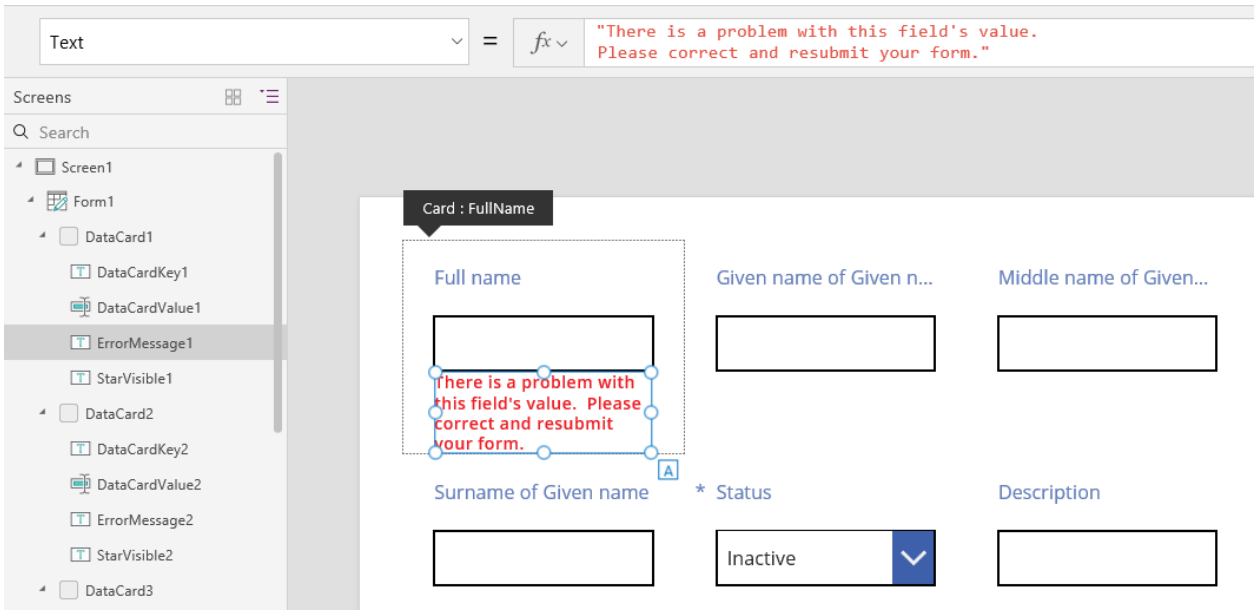
On the left side of the screen, the list of controls shows **ErrorMessage1**, which is our label control. As you update an app, you can select this control to give it some height and show grab handles with which you can position and size the control. The "A" in a blue box indicates that the control has **AutoHeight** set to **true**:



The **Text** property of this control is set to **Parent.Error**, which is used to obtain dynamic error information based on validation rules. For illustration purposes, let's statically set the **Text** property of this control, which will increase its height (and, by extension, the height of the card) to accommodate the length of the text:



Let's make the error message a little longer, and again the control and the card grow to accommodate. The row overall grows in height, keeping vertical alignment between the cards:

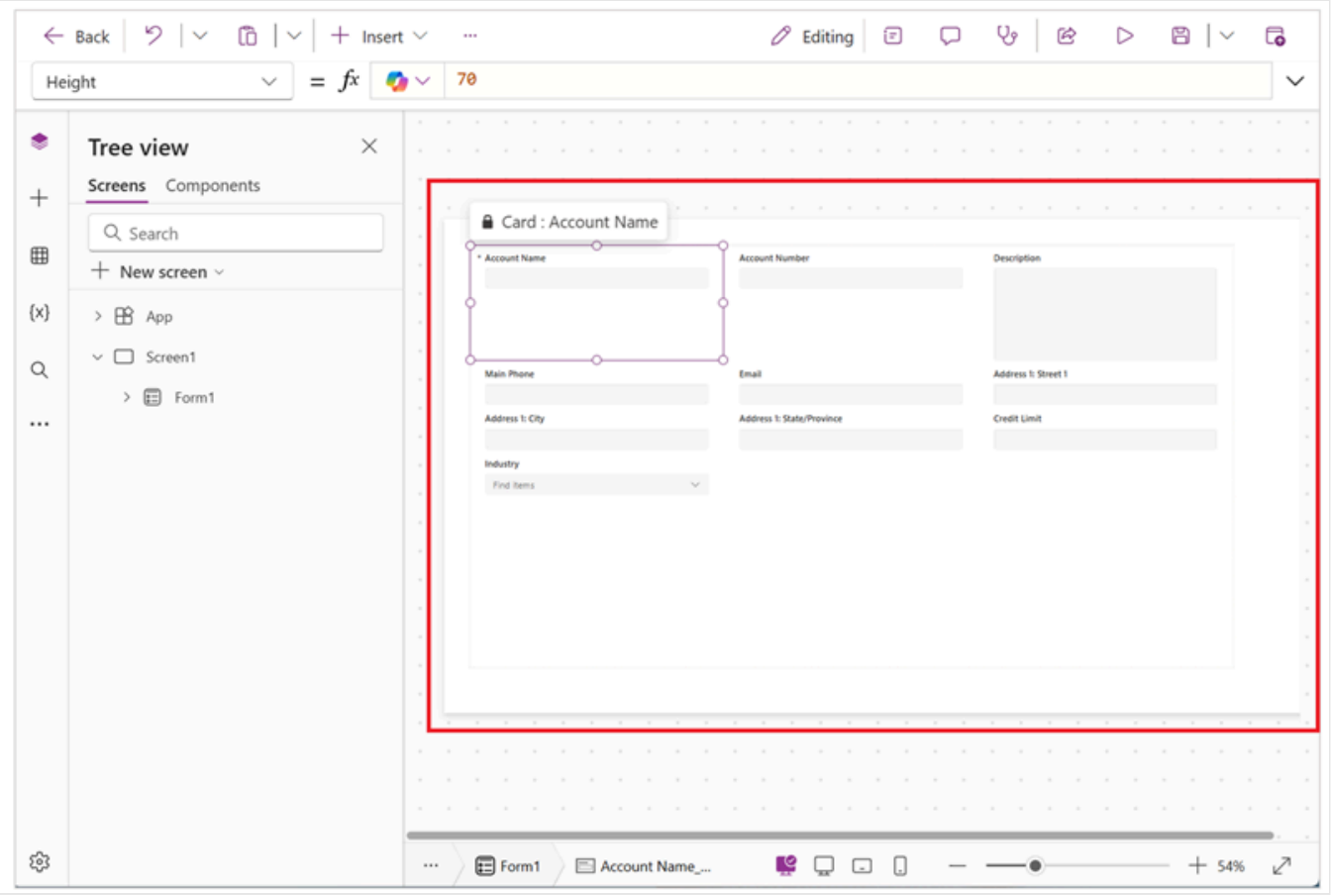


Understand data cards in canvas apps

This article explains what a data card is, how to change the control users enter data into, when to unlock a card, and how the form saves changes back to Dataverse.

Data cards are the building blocks of Power Apps forms. Each data card displays and collects a value for a single field such as Name, Job Title, or Phone Number.

The following screenshot shows a form with a data card for **Account name** selected.



A **Display form** shows one record. **Edit forms** let users update an existing record or create a new one. In either type of form, each data card is connected to exactly one field (one column) in your data source—often a Dataverse table.

A data card is a container that includes the field label, required validation behavior, and the control that users interact with. A data card contains the following:

- **Required indicator (asterisk or star)**: Shows that the field must have a value.
- **Title**: The label shown to users for the field.
- **Data card value**: The input control (for example, a text box or dropdown) where users enter or edit the value

- **Error message**: Displays validation errors, usually after the form is submitted.

Prerequisites

If you're new to forms, start by reading add-form.md and working-with-forms.md.

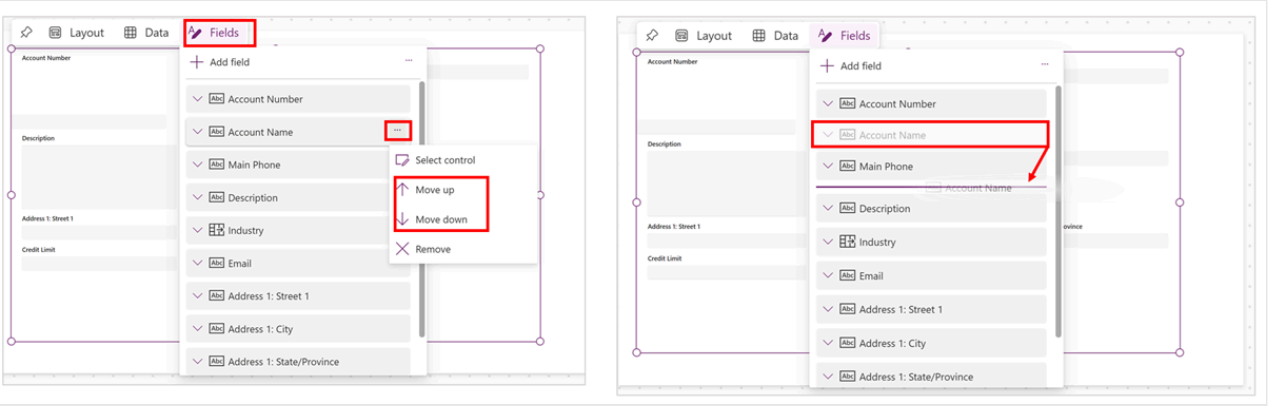
Customize a card

A data card contains the controls that users see - usually a label such as the title, an input control like text input or dropdown, and an error message label. To customize how a field looks, select the control *inside* the card for example, the text input and adjust its size, position, or properties.

Reorder fields on a form

Power Apps offers predefined cards for strings, numbers, and other data types. You edit the form and rearrange fields letting you change the layout without unlocking any cards.

1. Open your app for editing in Power Apps Studio.
2. Select the form then select **Fields**.
3. Select **More actions** and then selet **Move up** or **Move down**. You can also select and then drag fields into the order you want.



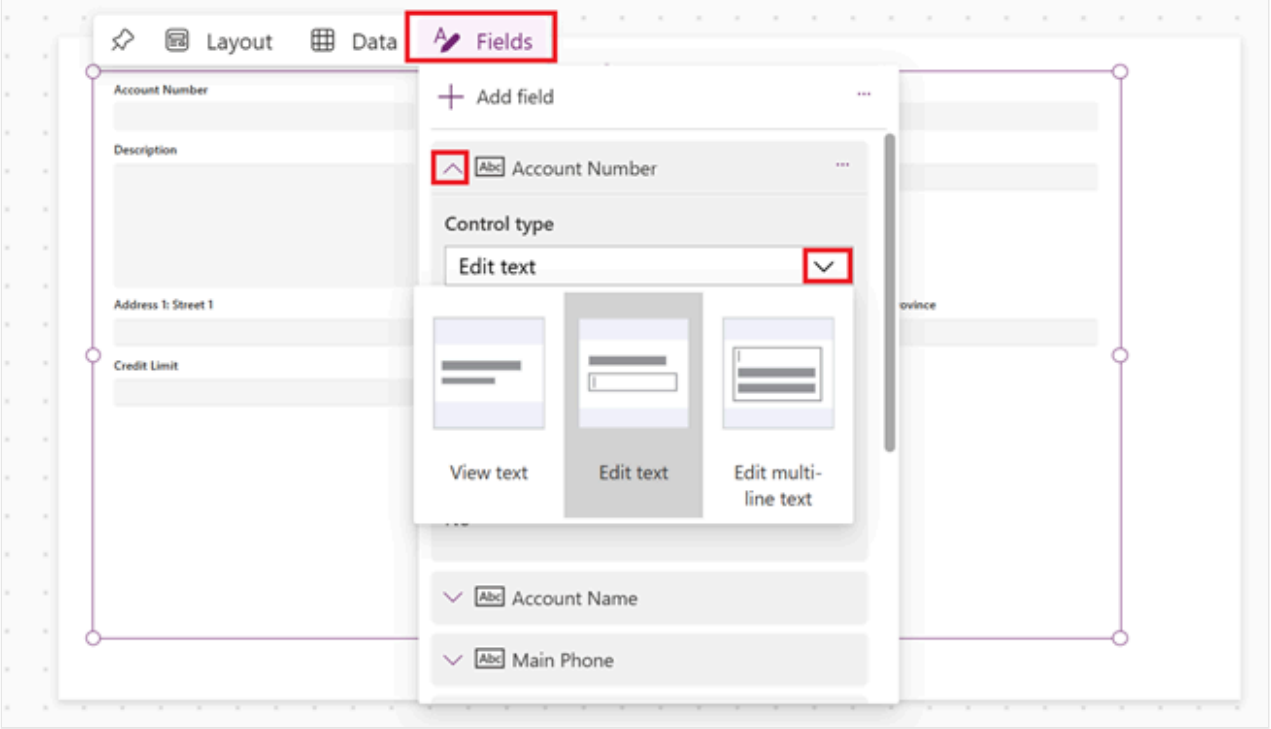
Change a card's control type

You can easly change a card's control type. For exmaple you might want to change a single-line text card to a multiline text card so users have more space to enter information.

1. Open your app for editing in Power Apps Studio.
2. Select the form then select **Fields**.

3. Select **Expand field details** which is the down arrow next to the field that you want to edit.

4. Select the dropdown for **Control type** and then select a different control type. For example, switch from **Edit text** to **Edit multi-line text**.



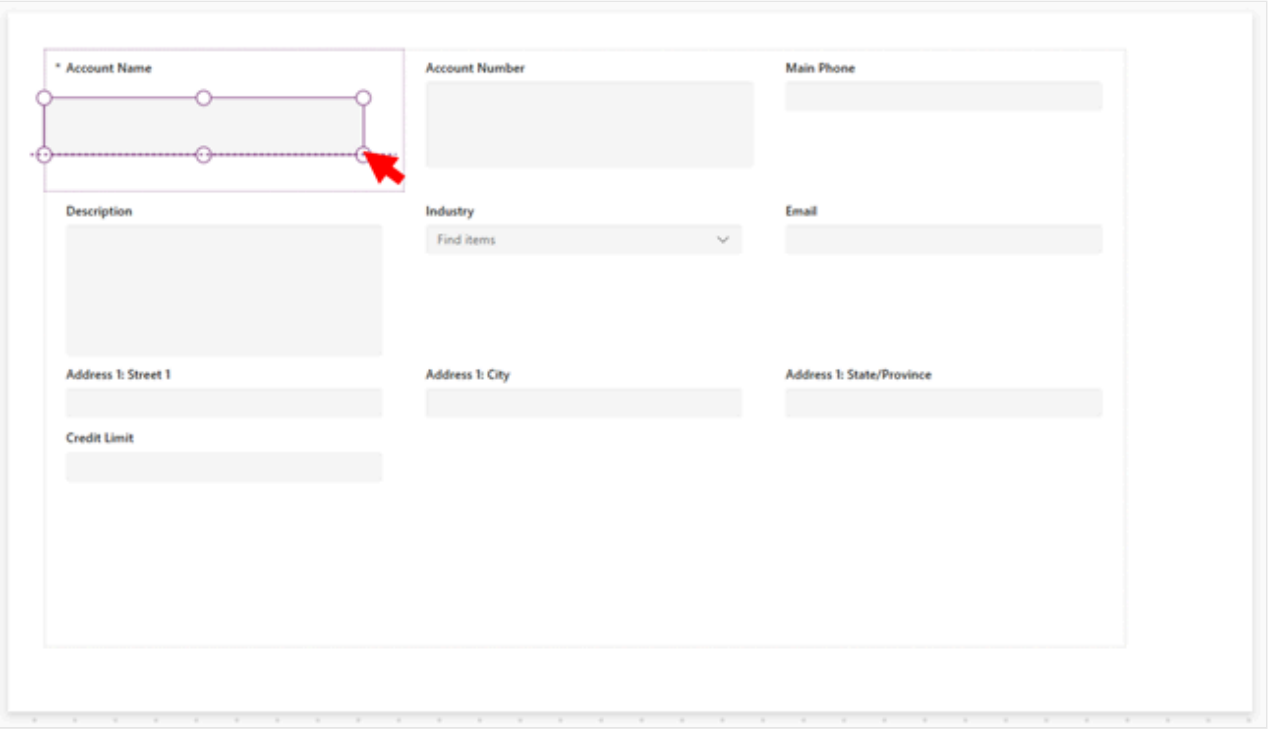
Resize a card

You can move and resize controls in a locked card.

You editing the controls inside a card without changing what field the card connects to.

1. In the form, select the data card that you want to resize such as **Account Name**.

2. Drag to move the text input within the card, and use the handles to resize it. This improves spacing and readability without unlocking the card.

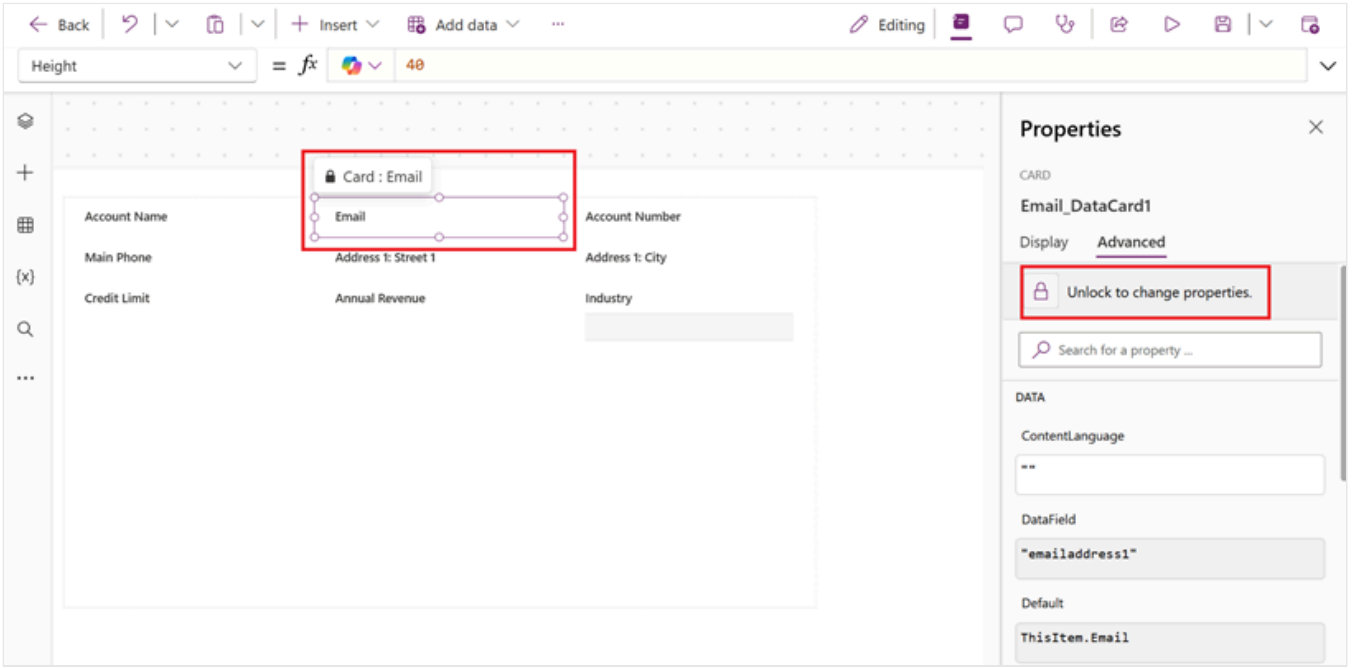


Unlock a card

To make changes like deleting a control or adding a new control requires unlocking the card.

When you add a field to a form, Power Apps creates a data card for you and sets up the basic formulas that connect the card to the data source. By default, Power Apps locks these cards so you don't accidentally break that connection. If you need more control, such as a custom layout, extra controls, or different formulas, you can unlock the card.

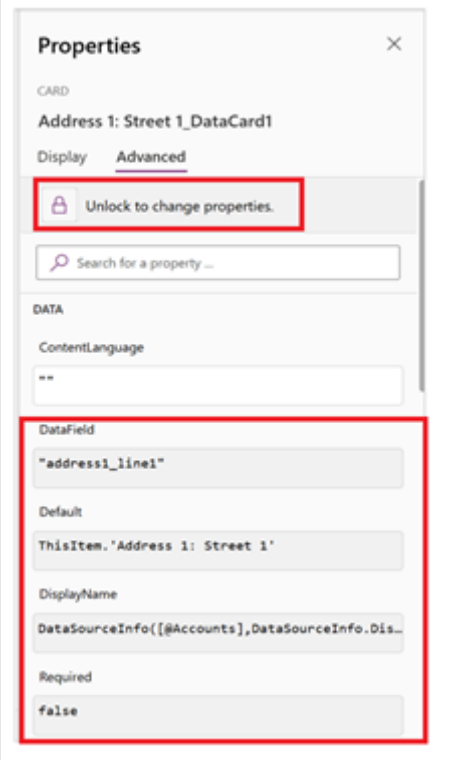
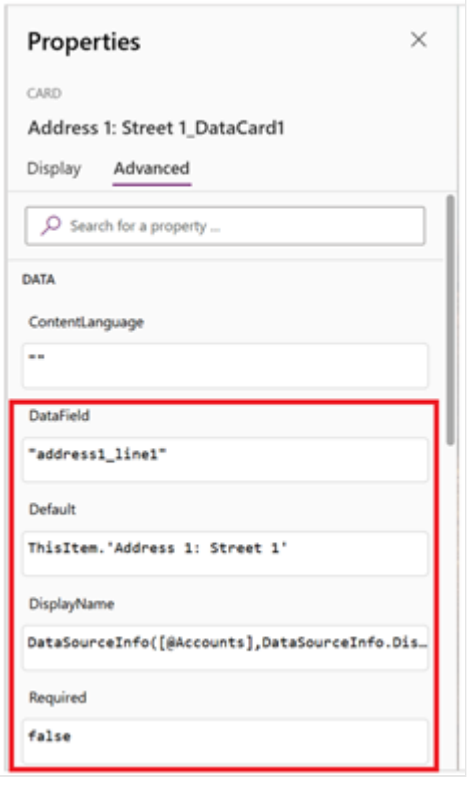
The following screenshot shows a form with a data card that is locked.



The key setting is DataField. It tells Power Apps which field (column) this card is responsible for. When the form submits, Power Apps uses the card's **DataField** value to know what field to update.

1. To unlock a card, select the card
2. In the **Properties** pane, select the **Advanced**.
3. Select lock icon or select **Unlock to change properties** to unlock properties such as **DataField**, **DisplayName**, or **Required**.
4. After you unlock the card, edit the generated formulas and add or remove controls inside the card.

Expand table

Locked properties	Unlocked properties
	

You now take control over this card and can modify it further to fit your needs. But you lose the ability to change the card from one representation to another (for example, single-line text to multi-line text) as you did before. You transformed the predefined card into a "custom card" that you now control.

> **Important**

You can't relock a card after you unlock it. To get a card back to a locked state, remove it, and reinsert it in the right-hand pane.

Interact with a form

After you unlock a card, you can control how values move between the form and the controls inside the card.

A helpful way to think about an Edit form is: **data flows in** to show the current record (defaults), and **data flows out** when you submit the form updates. The card sits in the middle - its properties tell the form which field it represents and what value should be saved.

DataField property

The card's **DataField** property is the "this card edits this field" setting. It helps Power Apps decide what to validate, what value is required, and what field should be updated when you submit the form.

Information flowing in

When a form shows a record, it exposes that record as **ThisItem**. Think of **ThisItem** as the current row you're editing. It contains every field for that record.

In Dataverse-backed forms, the card's **DataField** typically matches the field's logical name. The card's **Default** formula commonly references the current record, for example, **ThisItem.FieldName**, and the input control inside the card usually reads that value through **Parent.Default**. This pattern keeps the input control independent of the data source and lets the card encapsulate how the field value flows in.

Most cards set their **Default** property to the current record's value, like **ThisItem.FieldName**. You can optionally transform that value, for example, format text before it shows in the input.

Inside the card, the input control usually uses **Parent.Default** so it always shows whatever value the card provides. This approach keeps the card "self-contained". If you later change the card's default formula, you don't have to rewrite formulas inside the card.

By default, the data source's metadata sets the **DefaultValue** and **controls/control-card.md** properties based on the **controls/control-card.md** property. You can override these formulas with your own logic by using the **functions/function-datasourceinfo.md** function to integrate the data source's metadata.

Information flowing out

When the user selects **Save**, this typically call **SubmitForm**. The form gathers values from each data card and writes them back to the data source. It uses each card's **DataField** to know *which* field to update.

To save changes from an **Edit form**, trigger **SubmitForm(FormName)** (for example, from a **Save** button). If you want to clear the inputs after a successful submit, follow with **ResetForm(FormName)**. To create a new record instead of editing an existing one, set the form's **DefaultMode** property to **FormMode.New** before the user starts entering data.

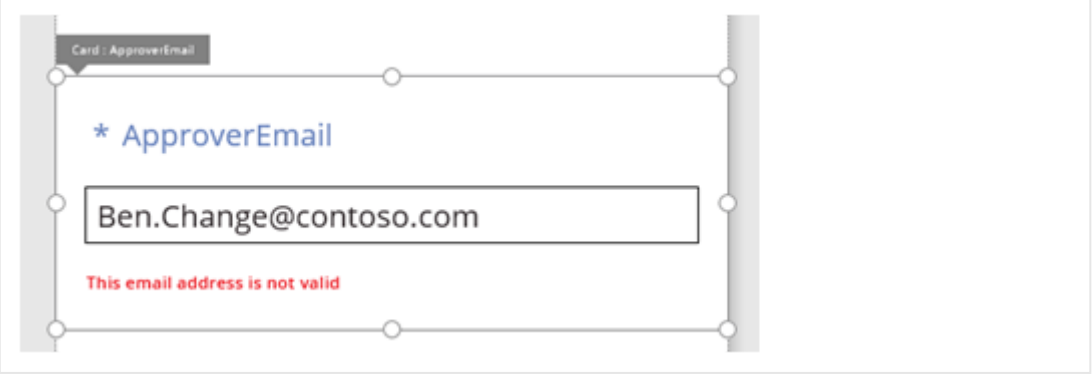
The form also reads each card's **Update** property - this is the value that gets saved for that field. If you need to clean up a value before saving, such as removing extra spaces, converting text to a number, or reversing a formatting change you made in **Default**, **Update** is usually the right place to do it.

Valid is basically "is this field okay to submit?" Power Apps uses the data source rules and the card's **Required** setting to decide this. If the value isn't valid, the card's **Error** property contains a message that you can show to the user, often through the **Error Message** label in the card.

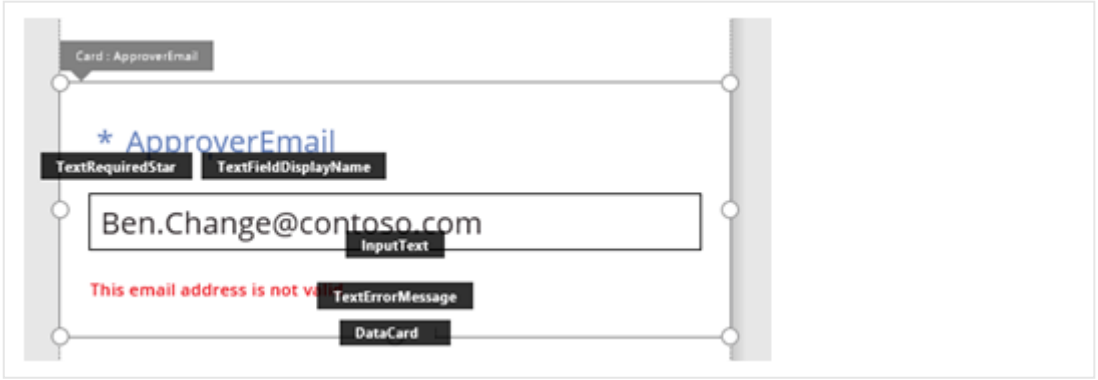
If a card's **DataField** is blank, the card isn't tied to any field - it's just a container you can use for layout. In that case, its **Update** and **Valid** values don't affect what gets saved when you submit the form.

Dissecting an example

Take a closer look at what's inside a typical data-entry card. The following screenshots spread the controls out so you can see each piece clearly.



In the next image, the controls inside the card are labeled so you can match what you see on the canvas to what's in the tree view.



Here are the main controls you usually see inside a card:

Expand table

Name	Type	What it does
TextRequiredStar	**Label** control	Shows a star or asterisk when the field is required.
TextFieldDisplayName	**Label** control	Shows the friendly field name the user sees. This name often differs from the internal schema name.
InputText	Text input control	Shows the current value and lets the user type a new one.
TextErrorMessage	**Label** control	Shows a message if the value can't be submitted, such as missing required data.

These controls usually don't connect to Dataverse directly. Instead, they read simple values from the **parent card** by using **Parent**, and the card handles the connection to the data source. The following formulas are common examples.

Expand table

Control property	Formula	Beginner explanation
TextRequiredStar.Visible	**Parent.Required**	Only show the star when the field is required.

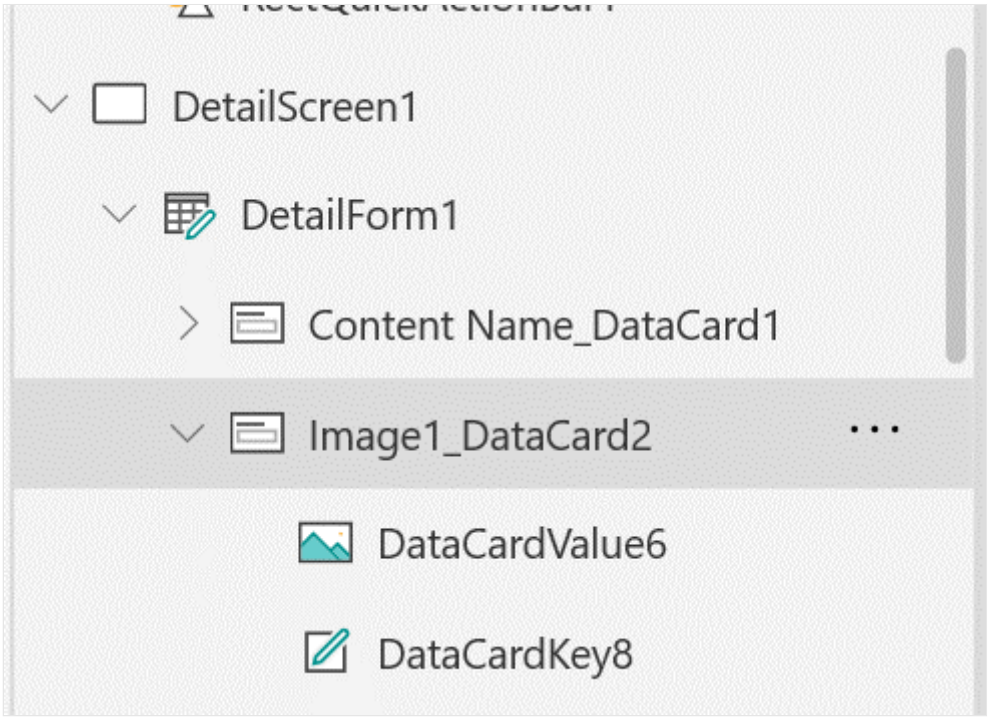
Last updated on 04/14/2026

Display a full-sized image on a canvas app form

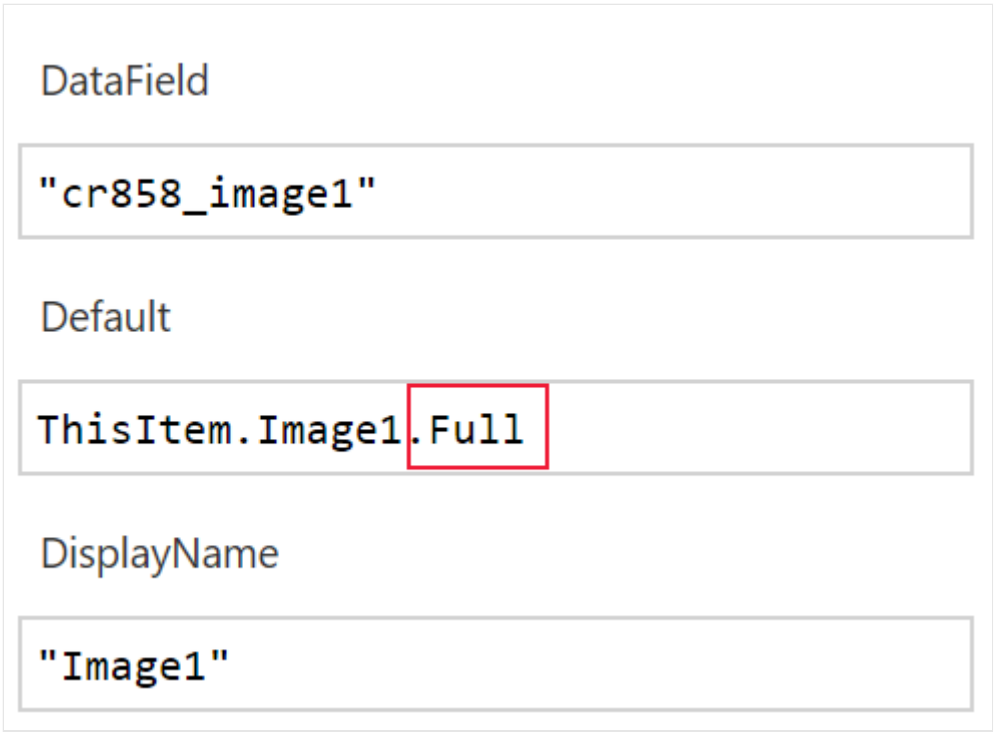
Article • 12/16/2022

By default, when an app user adds an image to display to a form in a canvas app, the image displayed is the thumbnail image. To display a full image for a canvas app, follow these steps:

1. Open the canvas app in PowerApps Studio.
2. Select **Insert**, and then select **Image**.
3. Select the image data card.



4. Under the **Advanced** tab set the **Data** field to the table that contains the image you want to display.
5. Add **.Full** after the value for the image **Default** setting.



6. Select **Save**.

See also

Show, edit, or add a record in a canvas app
Image fields

Add a list box, a drop-down list, a combo box, or radio buttons to a canvas app

Article • 12/16/2022

Show a single column of data (for example, from a multi-column table) in a canvas app so that users can select one or more items in a list.

- Add a list box to allow users to select more than one option.
- Add a drop-down list or combo box to take up less space on a screen.
- Add a set of radio buttons for a particular design effect.

This topic focuses on lists boxes and radio buttons, but the same principles apply to drop-down lists.

Prerequisites

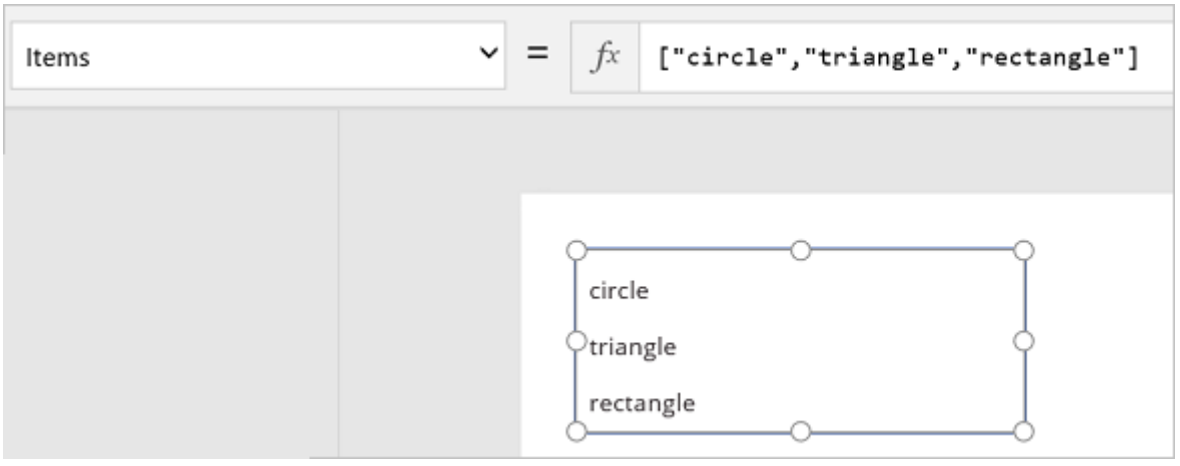
1. Create a blank canvas app.
2. Learn how to add and configure controls.

Create a simple list

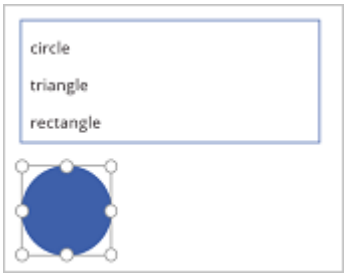
1. Add a **List box** control named **MyListBox**, and set its **Items** property to this expression:

   ```
   ["circle","triangle","rectangle"]
   ```

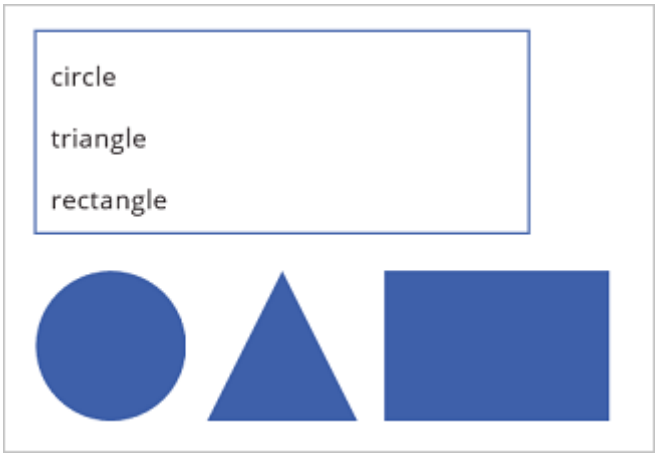
   Your designer looks similar to the following:

2. On the **Insert** tab, select **Icons**, select the circle, and move it under **MyListBox**:



3. Add a triangle and a rectangle, and then arrange the shapes in a row under **MyListBox**:



4. Set the **Visible** property of the following shapes to the following functions:

Expand table

Shape	Set Visible function to
circle	`If("circle" in MyListBox.SelectedItems.Value, true)`
triangle	`If("triangle" in MyListBox.SelectedItems.Value, true)`
rectangle	`If("rectangle" in MyListBox.SelectedItems.Value, true)`

5. While holding down the Alt key, select one or more shapes in **MyListBox**.

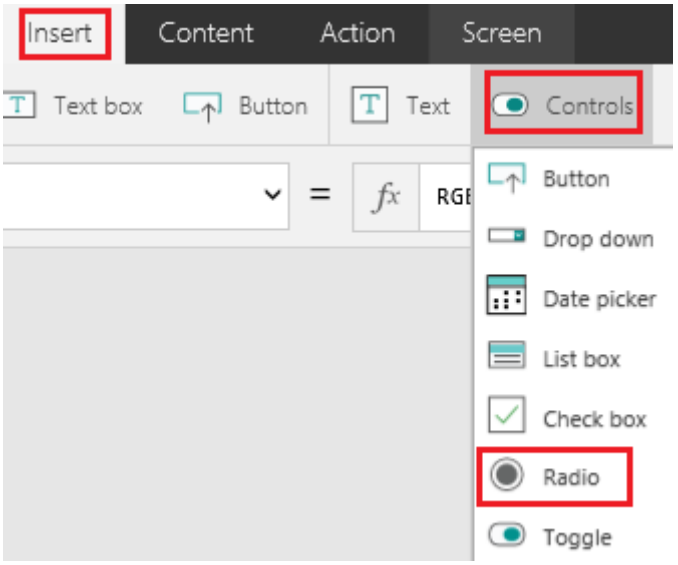
   Only the shape or shapes that you select appear.

In these steps, you used an expression to create a list of items. You can apply this to other elements within your business. For example, you can use a **Drop down** control to display product images, product descriptions, and so on.

Add radio buttons

1. On the **Home** tab, select **New Screen**, and then select **Blank**.

2. On the **Insert** tab, select **Controls**, and then select **Radio**.



3. Rename the **Radio** control to **Choices**, and set its **Items** property to this formula: `["red","green","blue"]`



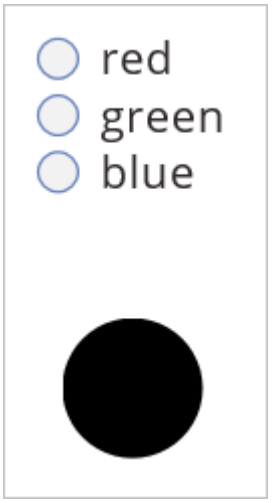
If needed, resize the control to show all the options.

4. On the **Insert** tab, select **Icons**, and then select the circle.

5. Set the **Fill** property of the circle to the following function:
`If(Choices.Selected.Value = "red", Red, Choices.Selected.Value = "green", Green, Choices.Selected.Value = "blue", Blue)`

In this formula, the circle changes its color depending on which radio button you choose.

6. Move the circle under the **Radio** control, as in this example:



7. While holding down the Alt key, select a different radio button to change the color of the circle.

Add an item to an existing list

1. Add a **Button** control and name it "btnReset".

   Don't know how to add, name, and configure a control?

2. Set **OnSelect** property on **btnReset** to this formula:

   ```
   Power Fx
   ClearCollect(MyItems, {value: "circle"},{value: "triangle"},{value: "rectangle"})
   ```

3. Set the **Text** property on **btnReset** to `"Reset"`.

4. Add a **List box** control named **lbItems**, and set its **Items** property to `MyItems`.

5. While holding down the **Alt** key, press the **Reset** button.

   > **Note**
   >
   > The list box should populate with the items from the "MyItems" collection.

6. Arrange the list box and button so they're lined up vertically.



7. Add a **Text Input** control, and name it "txtAdd".

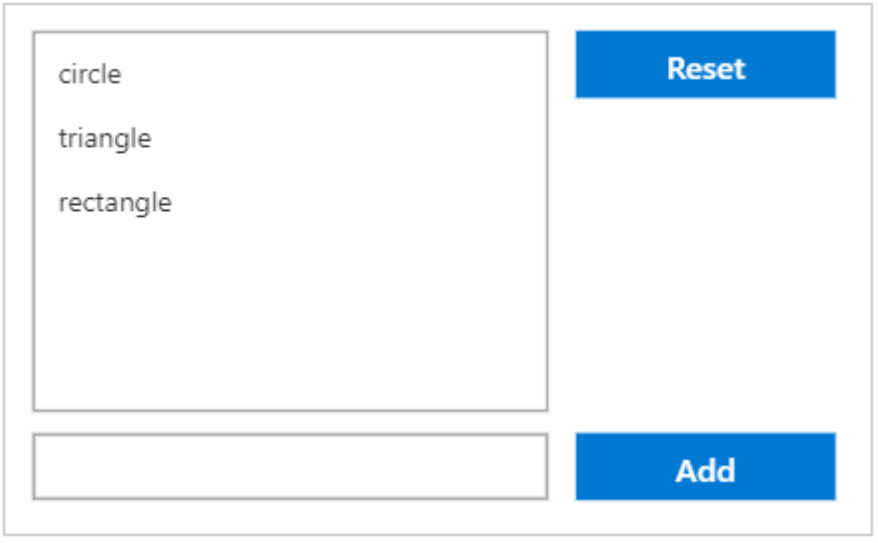
8. Set **Text** property of **txtAdd** to `""`.

9. Add a **Button** control, and name it "btnAdd".
10. Set the **Text** property of **btnAdd** to `"Add"`.
11. Set **OnSelect** property of **btnAdd** to the following formula:

```
Power Fx

Collect(MyItems,{value: txtAdd.Text}); Reset(txtAdd)
```

> **Note**
>
> - The collect function will add the text from the text input as an item in the collection.
> - The reset function will reset the text input back to it's default state.

12. Arrange **txtAdd** and **btnAdd** so they're lined up vertically underneath **lblItems** and **btnReset**.



13. Preview the app by pressing **F5**.
14. Add a text value to **txtAdd** text input control.
15. Press the Add button.

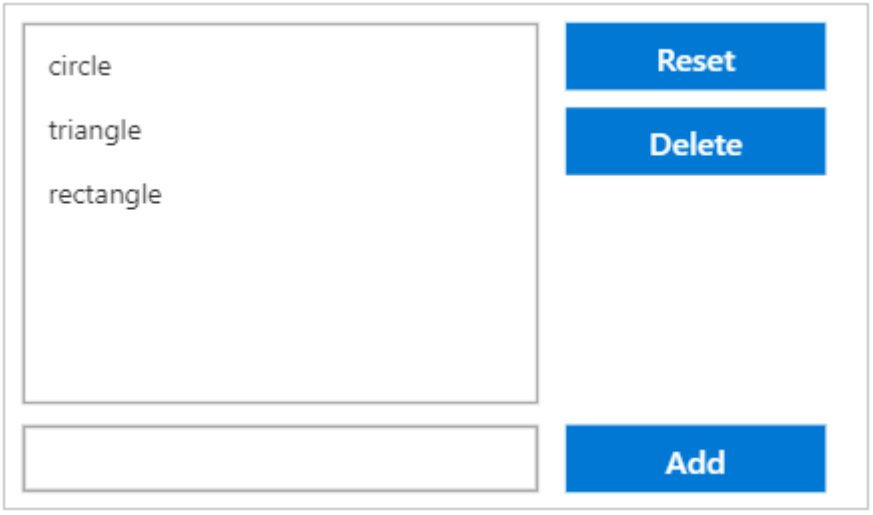
> **Note**
>
> The list box should populate with the items from the MyItems collection.

(Optional) Remove an item from an existing list

1. Add a **Button** control, and name it "btnDelete".
2. Set the **Text** property of **btnDelete** to `"Delete"`.
3. Set **OnSelect** property of **btnDelete** to the following formula:

```
Power Fx

Remove(MyItems, lbItems.Selected)
```

4. Arrange **btnDelete** so it's lined up vertically underneath **btnReset**



5. Preview the app by pressing **F5**.
6. Press the Reset button to reset the list box.
7. Press an item in the list box to select it.
8. Press the Delete button to delete item.

Feedback

Was this page helpful? Yes No

Provide product feedback

Create dependent drop-down lists in a canvas app

Article • 12/16/2022

When you create dependent (or cascading) drop-down lists, users select an option in a list to filter options in another list. Many organizations create dependent lists to help users fill out forms more efficiently. For example, users might select a country or region to filter a list of cities, or users might select a category to show only the codes in that category.

As a best practice, create a data source for the values in the "parent" and "child" lists (for example, countries/regions and cities) that's separate from the data source that users update by using the app. If you take this approach, you can use the same parent and child data in more than one app, and you can update that data without republishing the app or apps that use them. You can accomplish the same outcome by using a collection or static data, but it isn't recommended for enterprise scenarios.

For the scenario in this topic, store employees submit issues to an **Incidents** list through a form. Employees specify not only the location of the store at which the incident occurred but also the department within that location. Not all locations have the same departments, so a **Locations** list ensures that employees can't specify a department for a location that doesn't have that department.

> **Note**
>
> This article uses a list created using Microsoft Lists as the data source. However, all tabular data sources work the same way. Within **Dataverse for Teams**, you'll need to enable **classic controls** to get the classic drop-down to work for this scenario.

Create data sources

A **Locations** list shows the departments at each location.

Location	Department
Eganville	Bakery
Eganville	Deli
Eganville	Produce

Location	Department
Renfrew	Bakery
Renfrew	Deli
Renfrew	Produce
Renfrew	Pharmacy
Renfrew	Floral
Pembroke	Bakery
Pembroke	Deli
Pembroke	Produce
Pembroke	Floral

An **Incidents** list shows contact information and information about each incident. Create the Date column as a **Date** column, but create the other columns as **Single line of text** columns to simplify configuration and avoid delegation warnings in Microsoft Power Apps.

First Name	Last Name	Phone Number	Location	Department	Description	Date
Tonya	Cortez	(206) 555 - 1022	Eganville	Produce	I had a problem with...	2/12/2019
Moses	Laflamme	(425) 555 - 1044	Renfrew	Floral	I experienced an issue...	2/13/2019

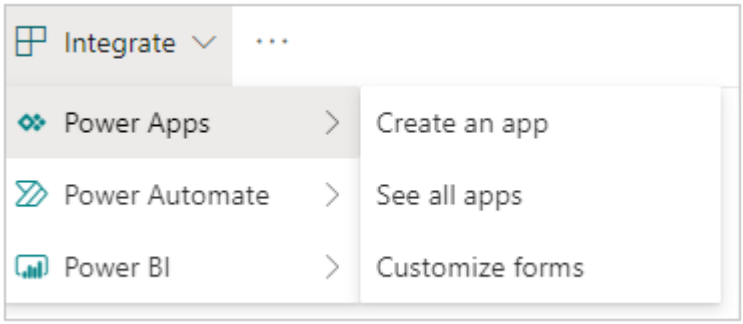
By default, lists include a **Title** column that you can't rename or remove, and it must contain data before you can save an item in the list. To configure the column so that it doesn't require data:

1. Near the upper-right corner, select the gear icon, and then select **List settings**.
2. On the **Settings** page, select **Title** in the list of columns.
3. Under **Require that this column contains information**, select **No**.

After that change, you can ignore the **Title** column, or you can remove it from the default view if at least one other column appears.

Open the form

1. Open the **Incidents** list, and then select **Integrate** -> **Power Apps** -> **Customize forms**.



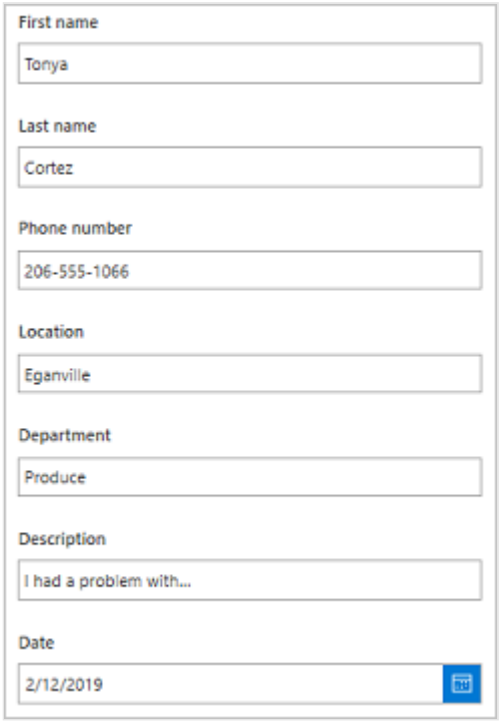
   A browser tab opens with the default form in Power Apps Studio.

2. (optional) In the **Fields** pane, hover over the **Title** field, select the ellipsis (...) that appears, and then select **Remove**.

   If you've closed the **Fields** pane, you can open it again by selecting **SharePointForm1** in the left navigation bar and then selecting **Edit fields** on the **Properties** tab of the right-hand pane.

3. (optional) Repeat the previous step to remove the **Attachments** field from the form.

   The form appears with just the fields that you added.

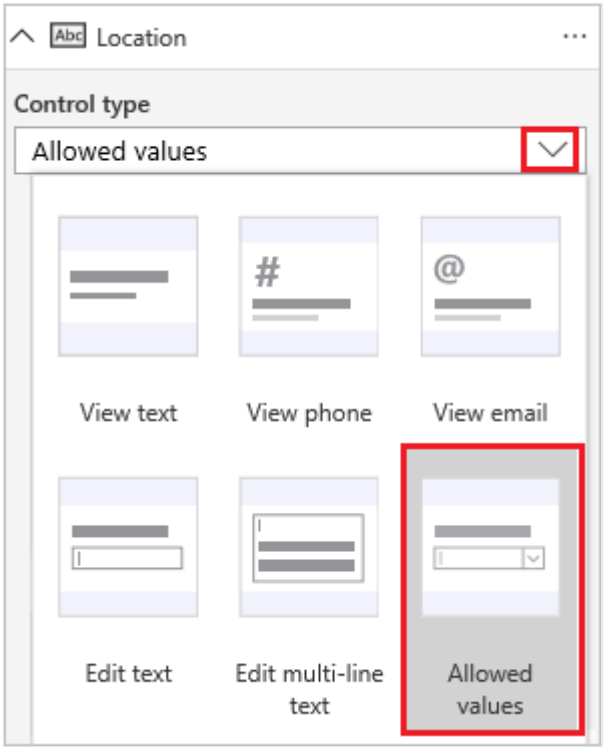


Replace the controls

1. In the **Fields** pane, select the arrow next to **Location**.

   If you've closed the **Fields** pane, you can open it again by selecting **SharePointForm1** in the left navigation bar and then selecting **Edit fields** on the **Properties** tab of the right-hand pane.

2. Open the **Control type** list, and then select **Allowed Values**.



   The input mechanism changes to a **Drop down** control.

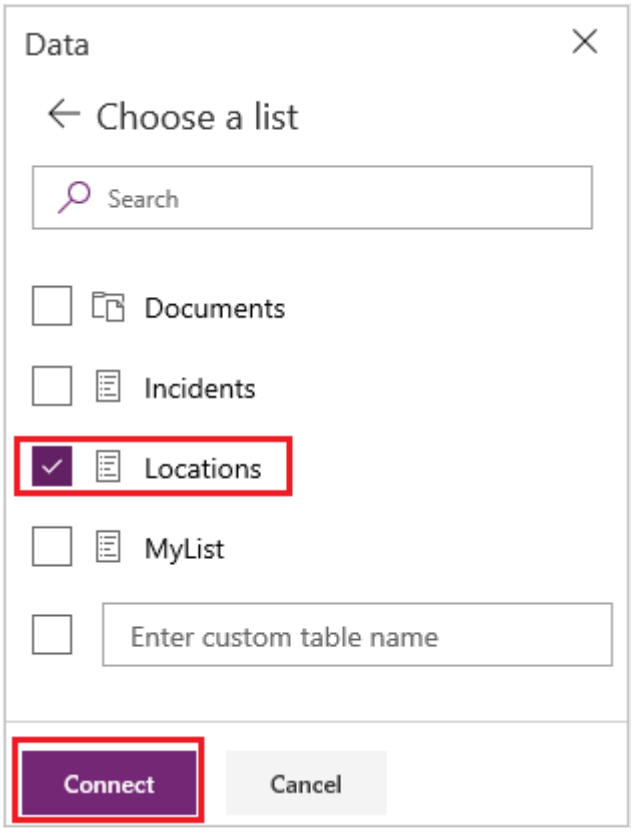
3. Repeat these steps for the **Department** card.

Add the Locations list

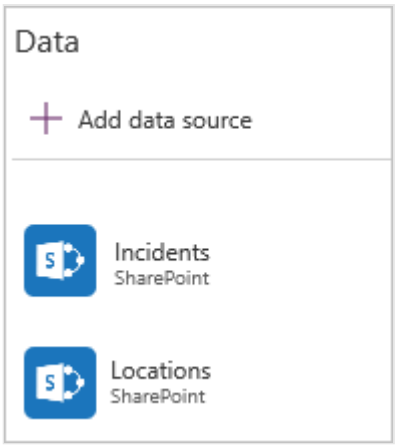
1. Select **View** > **Data Sources** > **Add data source**.

2. Select or create a SharePoint connection, and then specify the site that contains the **Locations** list.

3. Select the check box for that list, and then select **Connect**.



The list of connections shows the **Incidents** list, on which the form is based, and the **Locations** list, which will identify locations and departments in the form.



Unlock the cards

1. Select the **Location** card, select the **Advanced** tab in the right-hand pane, and then select **Unlock to change properties**.

2. Repeat the previous step for the **Department** card.

Rename the controls

If you rename your controls, you can identify them more easily, and the examples are easier to follow. To discover other best practices, review the Coding Standards and

Guidelines whitepaper.

1. In the **Location** card, select the **Drop down** control.
2. Near the top of the right-hand pane, rename the selected control by typing or pasting **ddLocation**.



3. Repeat the previous two steps in the **Department** card to rename the **Drop down** control to **ddDepartment**.

Configure the locations

1. Set the **Items** property of **ddlocation** to this formula:

```
Distinct(Locations, Location)
```

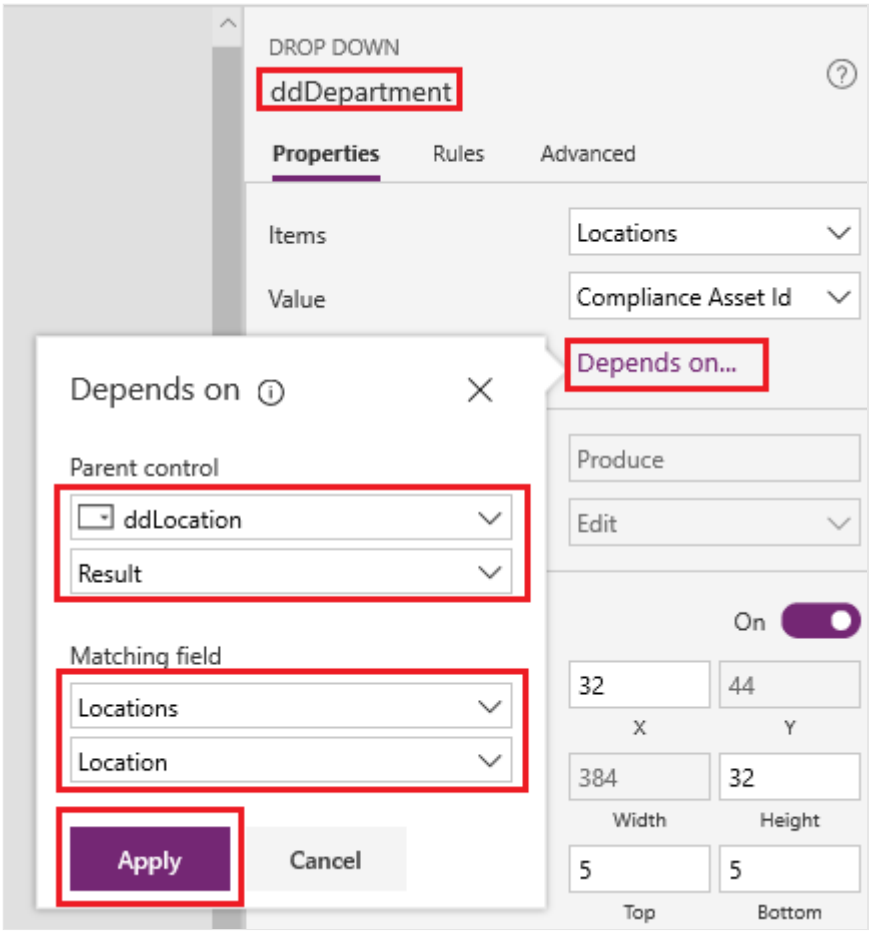
2. (optional) While holding down the Alt key, open **ddLocation**, and confirm that the list shows the three locations.

Configure the departments

1. Select **ddDepartment**, and then, on the **Properties** tab of the right-hand pane, select **Depends on.**
2. Under **Parent control**, ensure that **ddLocation** appears in the upper list and **Result** appears in the lower list.

> **Note**
>
> If you don't want to match on a string but on the actual ID of the row of data, select **ID** Instead of **Result**.

3. Under **Matching field**, select **Locations** in the upper list, select **Location** in the lower list, and then select **Apply**.



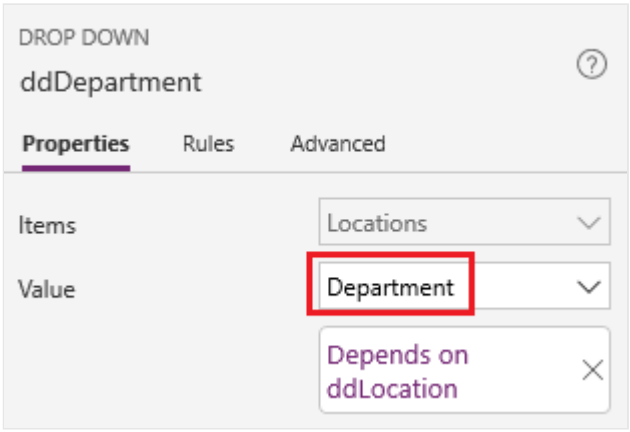
The **Items** property of **ddDepartment** is set to this formula:

```
Filter(Locations, Location = ddLocation.Selected.Result)
```

This formula filters the items in **ddDepartment** based on what the user selects in **ddLocation**. Such a configuration ensures that the "child" list of departments reflects the data for its "parent" location, as the **Locations** list in SharePoint specifies.

4. On the **Properties** tab of the right-hand pane, open the list next to **Value**, and then select **Department**.

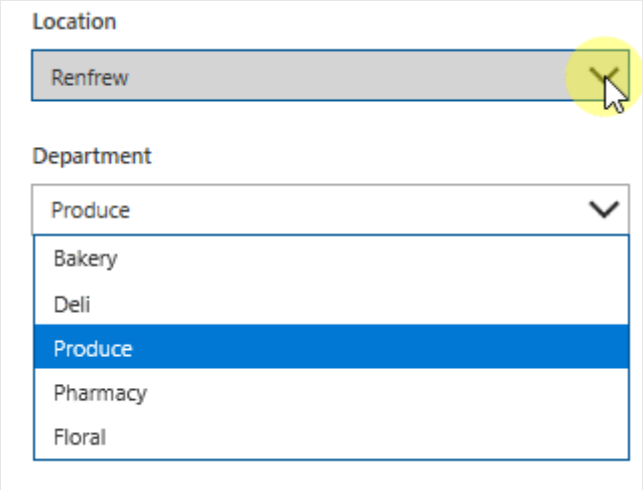
   This step sets the display text to the options from the **Department** column of the **Locations** list in SharePoint.



Test the form

While holding down the Alt key, open the list of locations, select one, open the list of departments, and then select one.

The lists of locations and departments reflects the information in the **Locations** list in SharePoint.



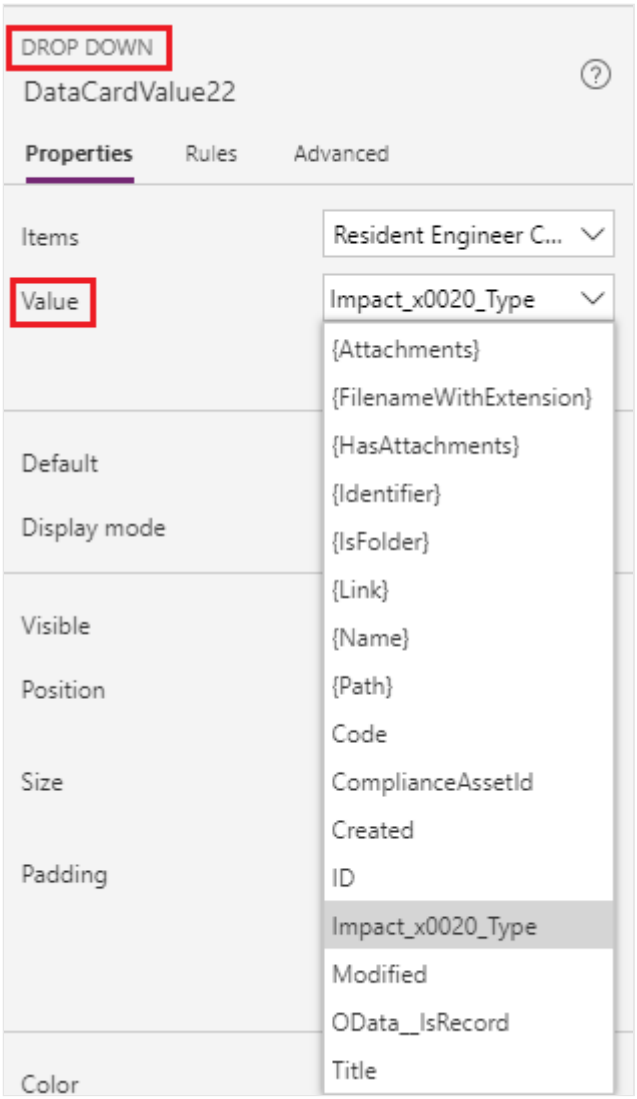
Save and open the form (optional)

1. Open the **File** menu, and then select **Save** > **Publish to SharePoint** > **Publish to SharePoint**.
2. In the upper-left corner, select the back arrow, and then select **Back to SharePoint**.
3. In the command bar, select **New** to open your customized form.

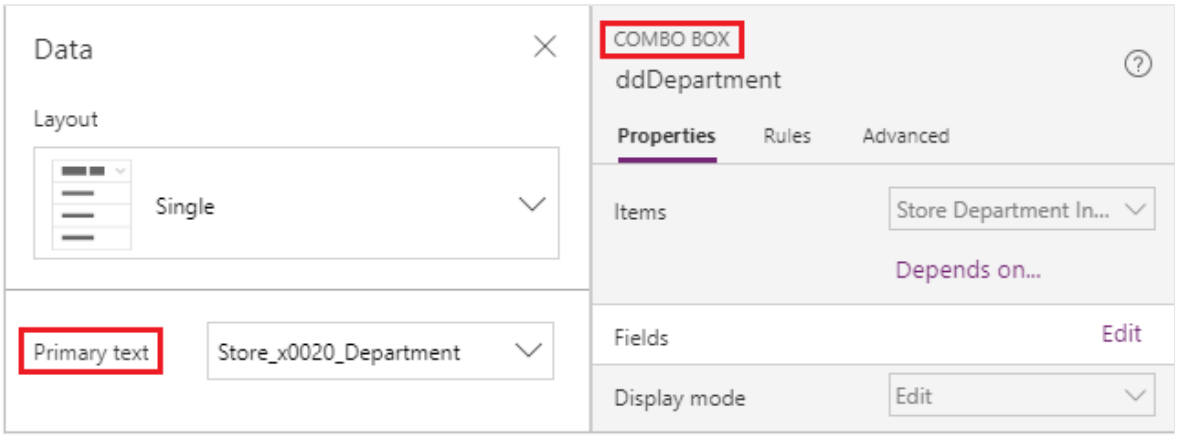
FAQ

I can’t see any data: the sources are all blank or have the wrong data. Confirm whether you're displaying the correct field for your control in either of these ways:

- Select a drop-down list, and then select the **Value** property in the **Properties** tab of the right-hand pane.



- Select a combo box, and then ensure that the primary text is the field that you want to display.



My child drop-down list contains duplicate items. This symptom is likely due to using a **LookUp** column in SharePoint or a **Choices** function in Power Apps. To remove the duplication, wrap a **Distinct** function around the properly returning data. More information: Distinct function.

Known limitations

This configuration is available on **Drop down** controls, as well as **Combo box** and **List box** controls that allow one selection at a time. You can't use the **Depends On** configuration for any of those controls if they allow multiple selections. This approach isn't recommended for working with choices in Microsoft Dataverse.

The **Depends On** configuration doesn't support static data or collections. To configure dependent drop-down lists with these sources, edit the expression directly in the formula bar. In addition, Power Apps doesn't support using two choice fields in SharePoint without any matching table of data, and you can't define **Matching field** within this UI.

Show data in a line, pie, or bar chart in canvas apps

Article • 05/21/2025

Use line charts, pie charts, and bar charts to display your data in a canvas app. When you work with charts, the data that you import should be structured based on these criteria:

- Each series should be in the first row.
- Labels should be in the leftmost column.

For example, your data should look similar to the following:

Expand table

Product	Revenue2012	Revenue2013	Revenue2014
Europa	21000	26000	28000
Ganymede	15000	17000	21000
Callisto	14000	19000	23000

You can create and use these charts within Power Apps. Let's get started.

Prerequisites

- Sign up for Power Apps, and then sign in using the same credentials that you used to sign up.
- Create an app from a template, from data, or from scratch.
- Learn how to configure a control in Power Apps.
- Create your own sample data using the example above and save it in Excel. Follow the steps in this topic to import it directly into your app.

Import the sample data

In these steps, we import the sample data into a collection, named **ProductRevenue**.

1. On the command bar select, **Insert** > **Media** > **Import**.

2. Set the control's **OnSelect** property to the following function:

```
Collect(ProductRevenue, Import1.Data)
```

3. On the app actions menu, select **Preview the app** and then select the **Import Data** button.

4. In the **Open** dialog box, select your Excel file, select **Open**, and then press Esc.

5. On the app authoring menu select, **Variables** > **Collections**.

   The ProductRevenue collection should be listed with the chart data you imported.

   > **Note**
   >
   > The import control is used to import Excel-like data and create the collection. The import control imports data when you are creating your app, and previewing your app. Currently, the import control does not import data when you publish your app.

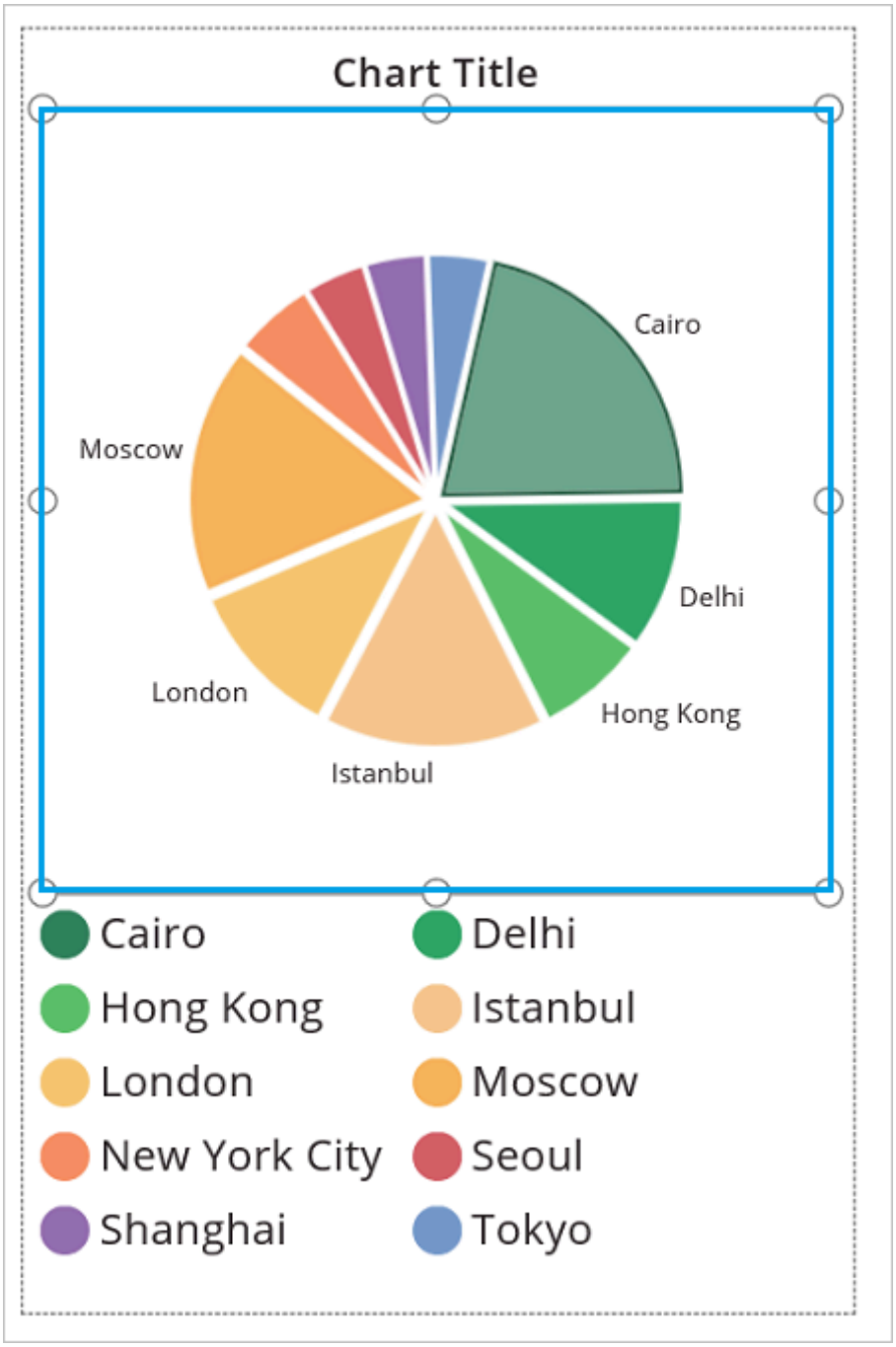
6. Press Esc to return to the default workspace.

Add a pie chart

1. On the command bar selelct, **Insert** > **Charts** > **Pie Chart**.

2. Move the pie chart under the **Import data** button.

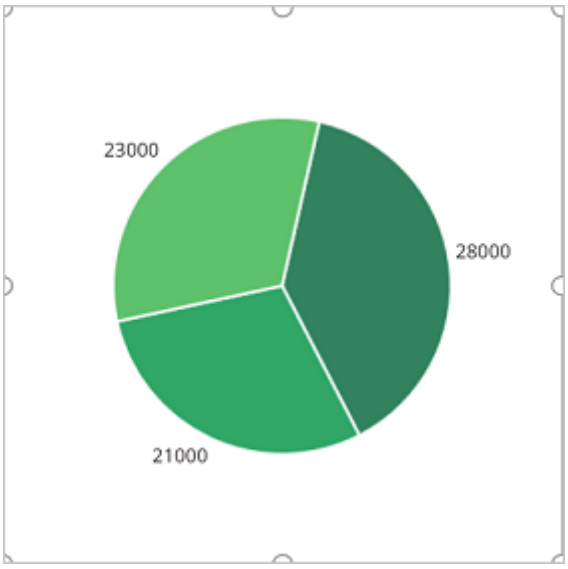
3. In the pie-chart control, select the middle of the pie chart:



4. Set the **Items** property of the pie chart to this expression: `ProductRevenue.Revenue2014`



The pie chart shows the revenue data from 2014.



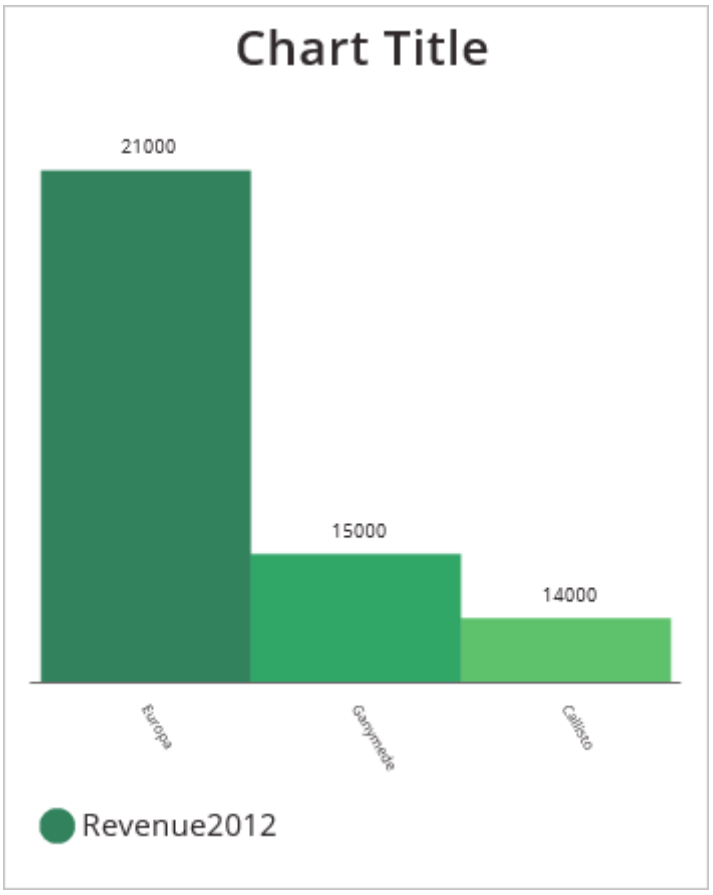
Add a bar chart to display your data

Now, let's use this ProductRevenue collection in a bar chart:

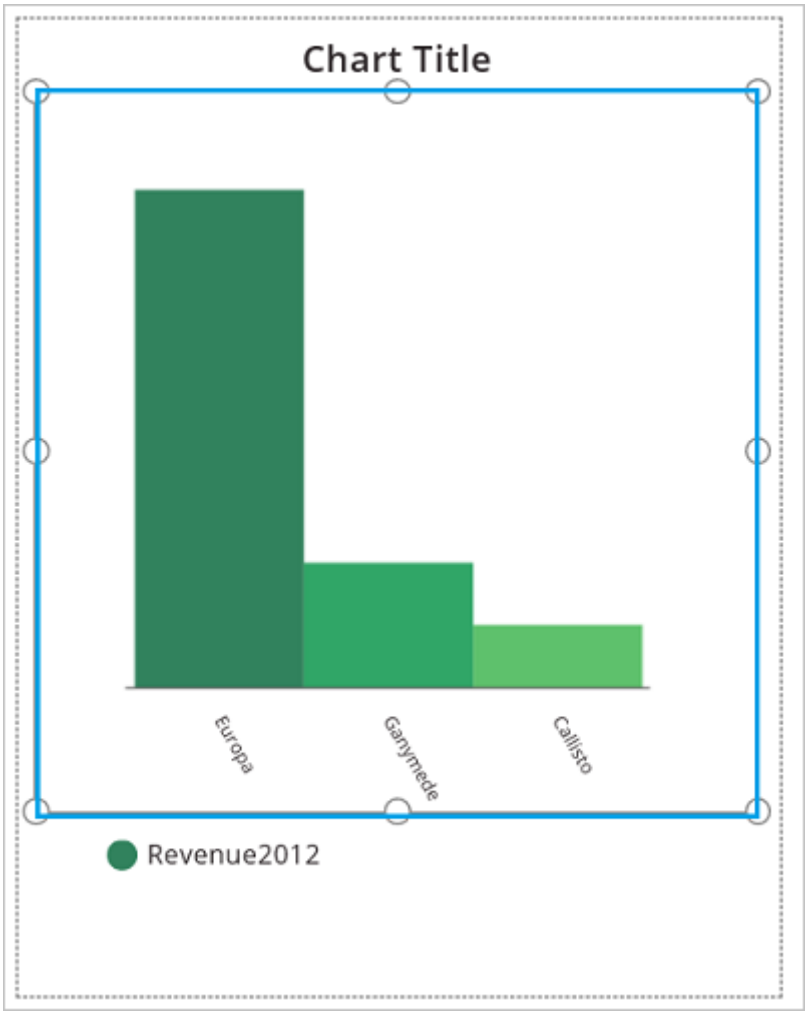
1. On the command bar, select **New screen** > **Blank**.
2. On the command bar, select **Insert** > **Tree view** > **Column Chart**.
3. Select the middle of the column chart. Set the **Items** property of the column chart to `ProductRevenue`:

   The column chart shows the revenue data from 2012:

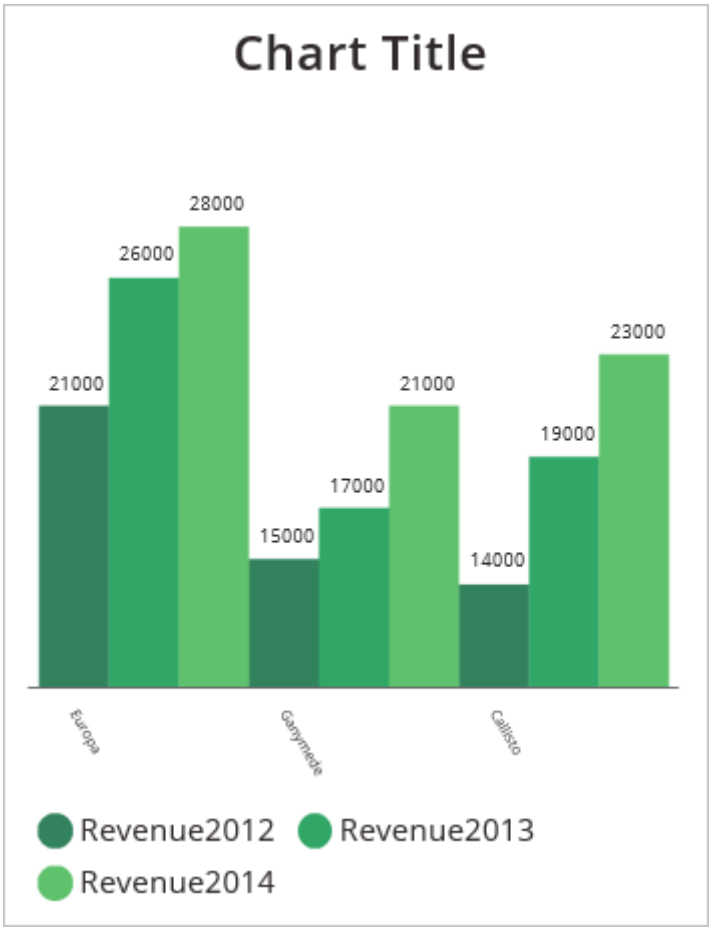


4. In the column chart, select the center square:



5. On the **Chart** tab, select **Number of Series**, and then enter **3** in the formula bar:

The column chart shows revenue data for each product over three years:



Using multimedia files in canvas apps

Summarize this article for me

You can add images, audio, and video files to a canvas app. Add video from streaming services such as Microsoft Clipchamp, Azure Media Services, or third-party streaming services, such as YouTube. Or use input controls such as **Pen Input** to collect signatures.

This article walks you through working with multimedia, streaming, and input control scenarios. The data source used in this article is an Excel file in OneDrive for Business.

Prerequisites

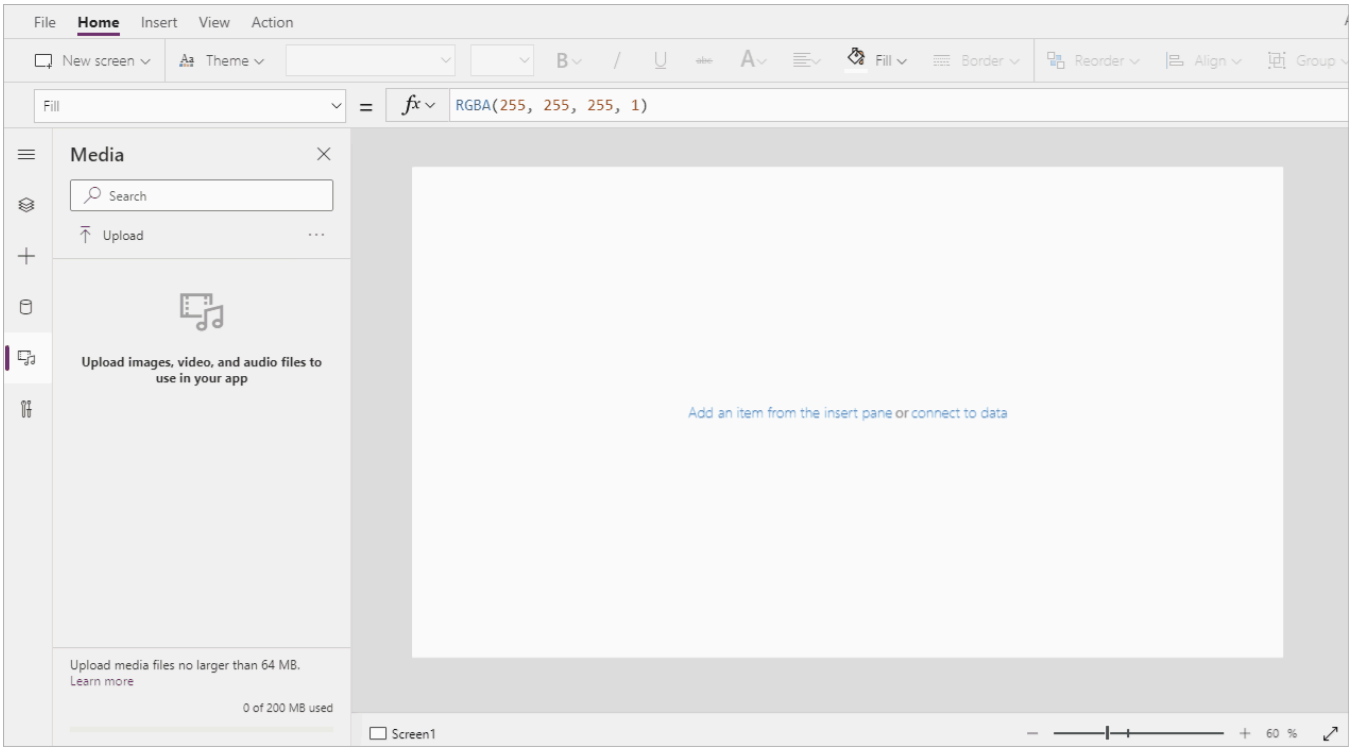
Sign up for Power Apps, and then sign in by using the same credentials that you used to sign up.

Watch this video to learn how to use multimedia files in canvas apps:

https://learn-video.azurefd.net/vod/player?id=d8fdba22-a9cf-4ce4-a77d-7c7be89666bc&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fadd-images-pictures-audio-video

Add media from a file or the cloud

You can choose the kind of media file to add (for example, images, video, or audio). You can add images by using the media pane or by using the images control.

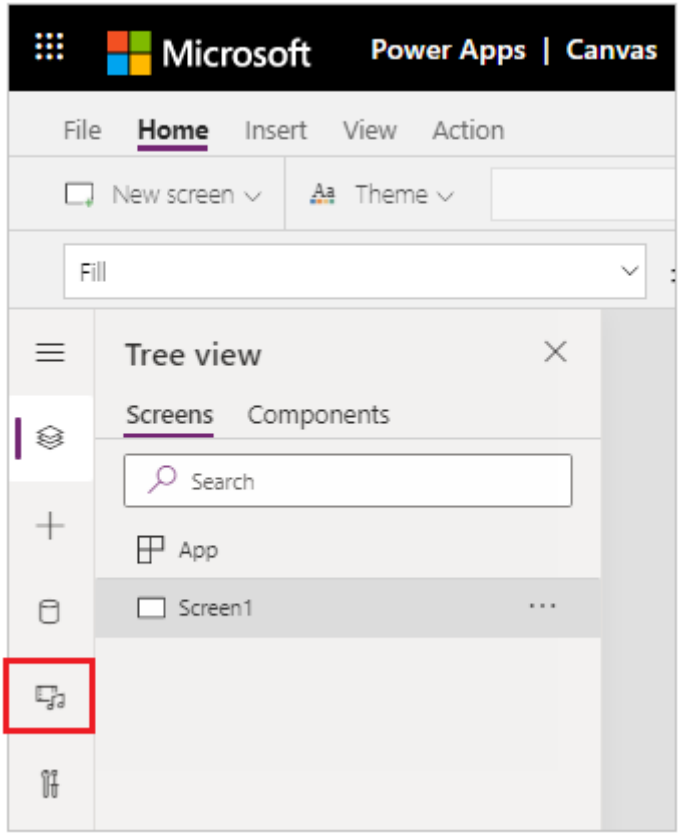


> **Note**
>
> When adding media files by URL,
>
> - Use HTTPS to ensure compatibility with modern browsers.
> - Ensure the URL is accessible anonymously without any authentication. Try to enter the URL in private browsing mode to verify that it's accessible without signing in.

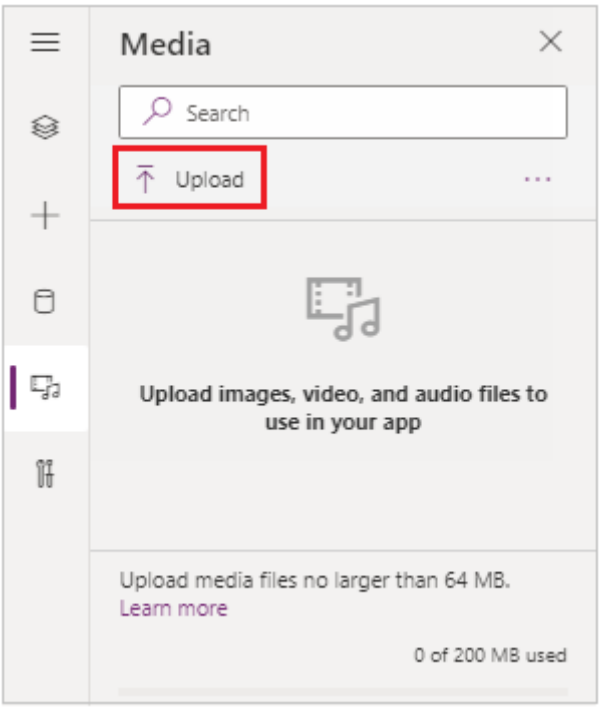
Add images, audio, or video by using the media pane

To add or remove media files in your app by using the **Media** pane:

1. Select **Media** from the left pane.

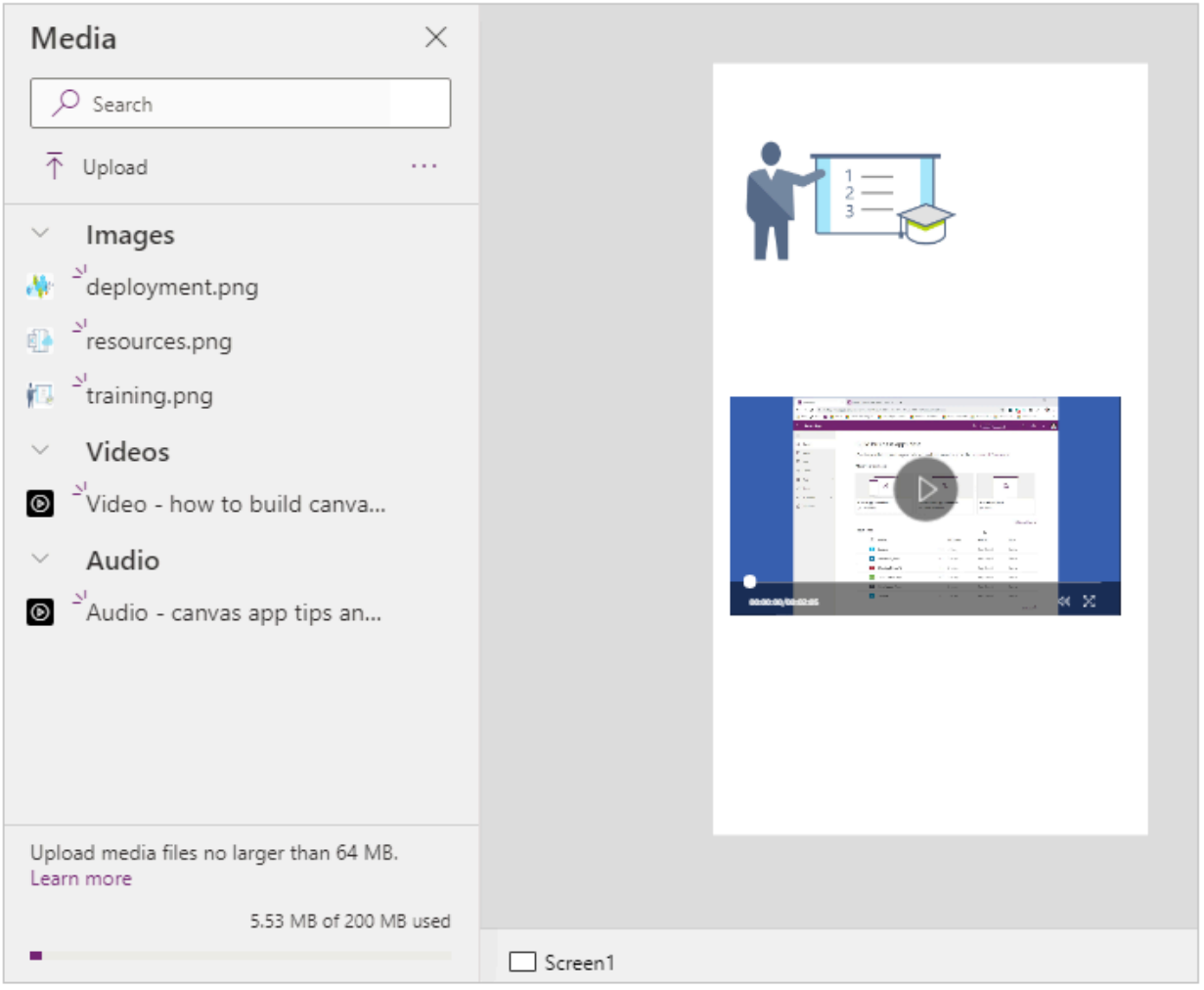


2. Select **Upload** from the media panel.



3. Select the files that you want to add, and then select **Open**.

4. Select the file from the media pane to insert into the screen.



5. Save and publish your app.

6. Share app with others.

Add images, audio, or video by using the controls

To add images, audio, or video by using the Image, Audio or Video controls:

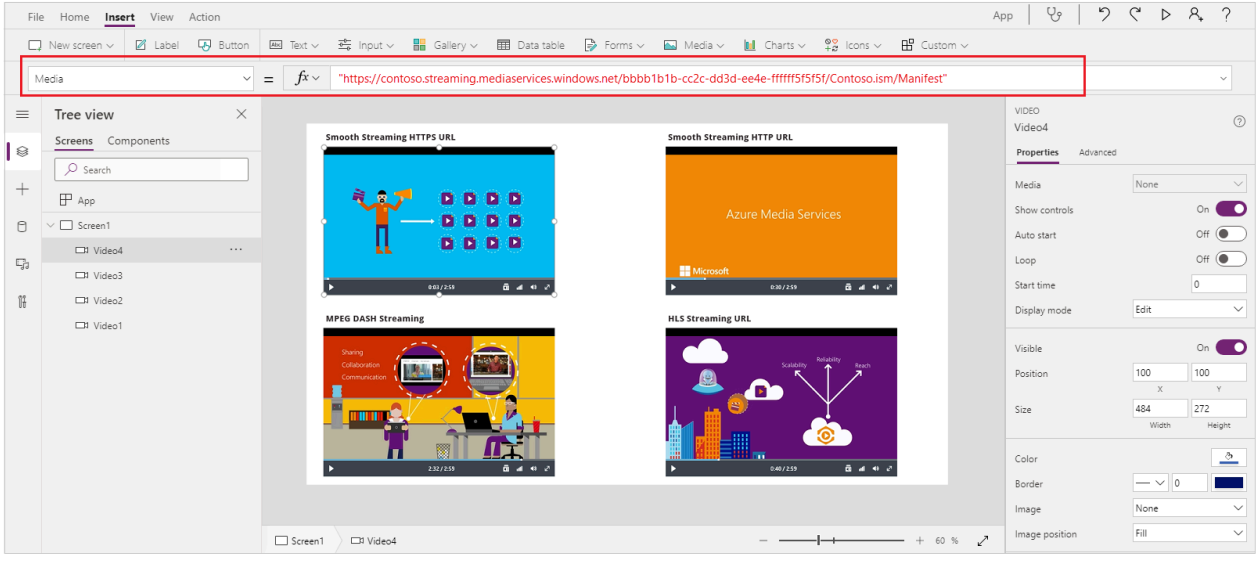
1. Select **Insert** from the top menu.
2. Select the **Media** drop-down.
3. Choose from **Image**, **Audio**, or **Video** controls.

For the **Image** control, update the *Image* property with the image file name, without extension. For the **Audio** or **Video** control, update the *Media* property with the file name, or the URL such as YouTube video URL in double quotation marks.

Add media from Azure Media Services

1. From your Azure Media Services account, upload and publish your video asset from **AMS** > **Settings** > **Assets**.
2. After the video is published, copy its URL.
3. From Power Apps, add the **Video** control from **Insert** > **Media**.
4. Set the **Media** property to the URL that you copied.

   As this graphic shows, you can choose any streaming URL that Azure Media Services supports:



5. Save and publish your app.

6. Share app with others.

Add Microsoft Clipchamp media

For information about how to add a Microsoft Clipchamp video control, see Microsoft Clipchamp video control example.

Add images from the cloud to your app

In this scenario, you save images in a cloud storage account, OneDrive for Business. You use an Excel table to contain the path to the images, and you display the images in a gallery control in your app. In this scenario, you use your own image files stored in OneDrive for Business. You create an Excel table that contains the file paths to those images, and then display the images in a gallery control in your app.

> **Note**
>
> The path to these images in the Excel file must use forward slashes. When Power Apps saves image paths in an Excel table, the path uses backslashes. If you use image paths from such a table, change the paths in the Excel table to use forward slashes instead of backslashes. Otherwise, the images won't display.

1. In OneDrive for Business, create a folder named Assets_images, and upload a few .jpg or .png files to that folder.

2. (Optional) If you used a different folder name, rename the folder to Assets_images.

3. In an Excel spreadsheet, create a one-column table named Images, and then add one row per image. In each row, paste the file path (direct link) to an image in your Assets_images folder.

Images
https://contoso.sharepoint.com/personal/contoso_com/Assets_images/Jacket_10_LightBrown.jpeg
https://contoso.sharepoint.com/personal/contoso_com/Assets_images/Jacket_5_Lilac.jpeg
https://contoso.sharepoint.com/personal/contoso_com/Assets_images/Jacket_7_DARKBLUE.jpeg
https://contoso.sharepoint.com/personal/contoso_com/Assets_images/Jacket_7_DarkGreen.jpeg
https://contoso.sharepoint.com/personal/contoso_com/Assets_images/Jacket_8_PaleMarigold.jpeg

   Tip: To copy the file path from **OneDrive for Business**, select an image file, and then select **Path** (*Copy direct link*) from the details pane on the right side of the screen.

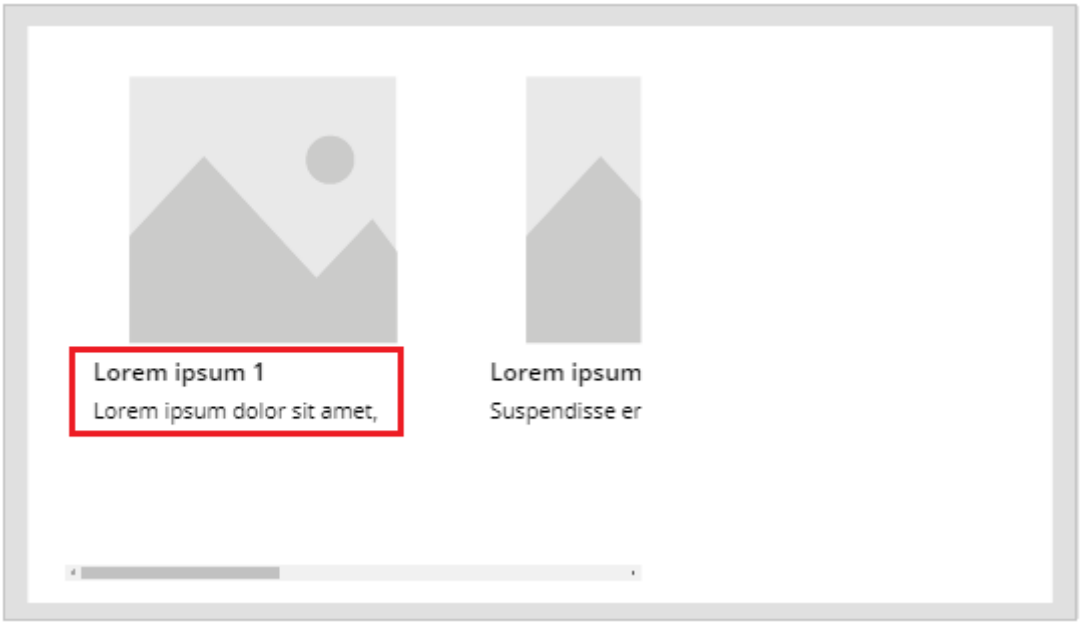
4. Name the table **Jackets**, and name the Excel file **Assets.xlsx**.

5. In your app, add the **Jackets** table as a data source.

6. Optionally, update your app orientation to Landscape.

7. Select **Insert** > **Gallery**, and then select **Horizontal**.

8. Optionally, select the text field and then the heading field under the first image, and delete them to keep only images on the screen.

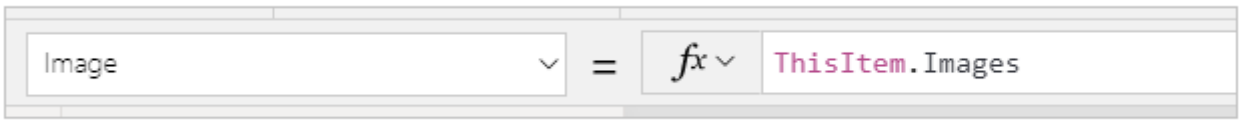


   If you see a formula error, press **Ctrl+Z** to undo the delete. Then, delete the *Subtitle* field and the *Title* field.

9. Set the gallery's **Items** property of the gallery to `Jackets`.



10. Select the first image in the gallery, and set its **Image** property to `ThisItem.Images`:



    The gallery is automatically updated with the images:



When you set the **Items** property, a column named **PowerAppsId** is automatically added to the Excel table.

11. Save and publish your app.

12. Share app with others.

Upload pen drawings to the cloud

In this scenario, you learn how to upload pen drawings to your data source, OneDrive for Business, and examine how the drawings are stored there.

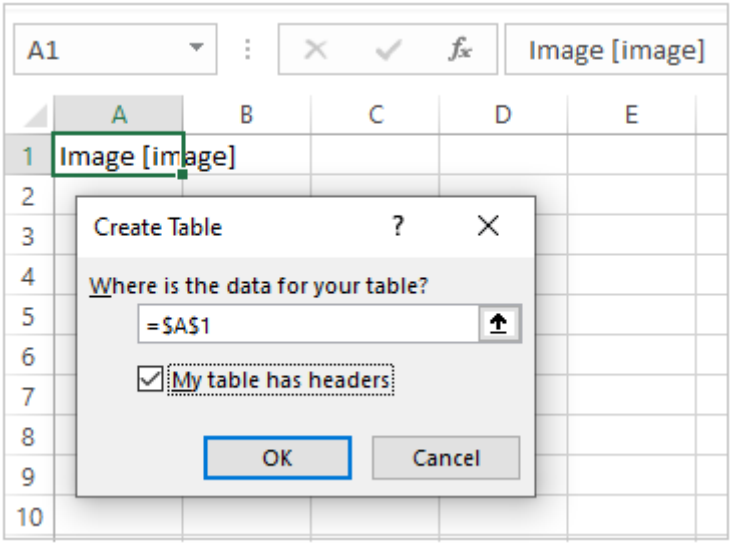
1. In Excel, add **Image [image]** to cell A1.

2. Create a table by using the following steps:

   a. Select cell A1.

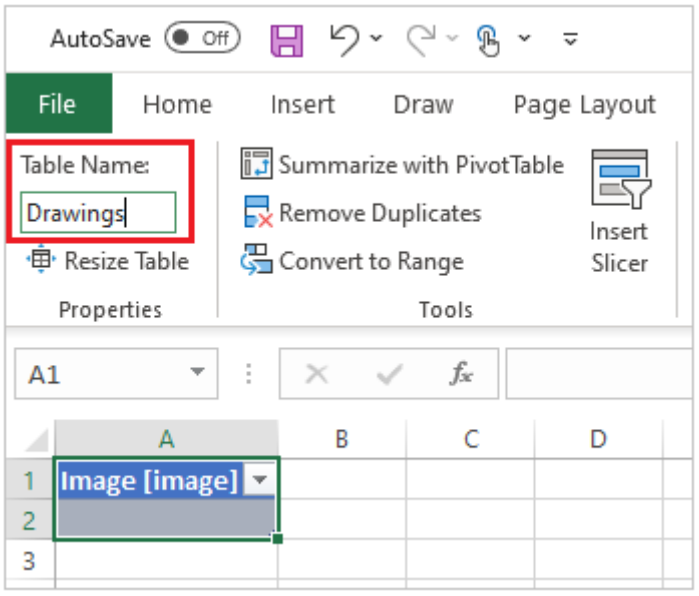
   b. On the **Insert** ribbon, select **Table**.

   c. In the dialog box, select **My table has headers**, and then select **OK**.



Your Excel file is now in a table format. For more information about table formatting in Excel, see Format the data as a table.

d. Name the table **Drawings**:

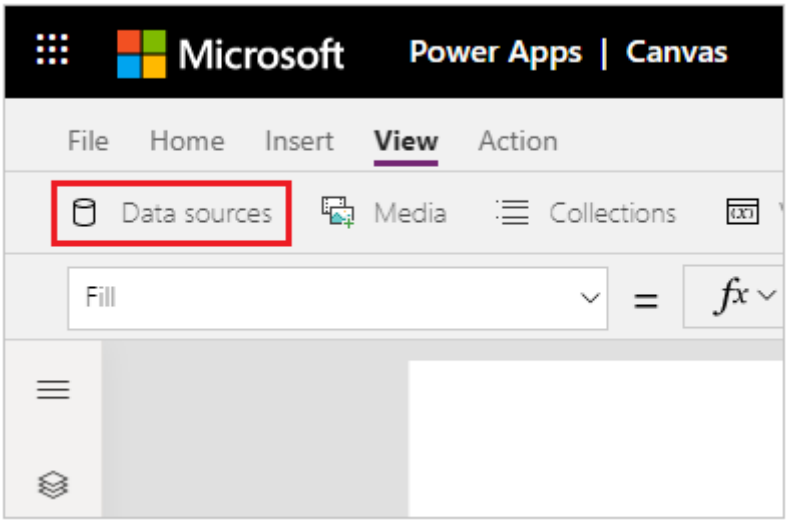


3. Save the Excel file to OneDrive for Business as **SavePen.xlsx**.

4. In Power Apps, create a blank app with Tablet layout.

5. In your app, add the OneDrive for Business account as a data source:

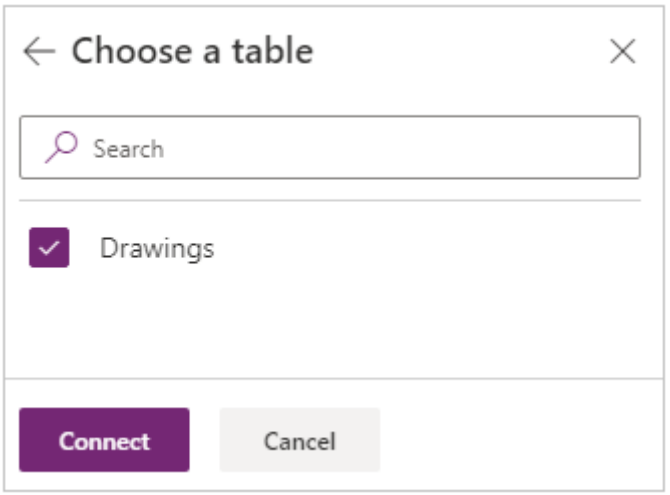
   a. Select **View** menu, and then select **Data sources**.



   b. Select **Add data source**, and then select **OneDrive for Business**.

   c. Select **SavePen.xlsx**.

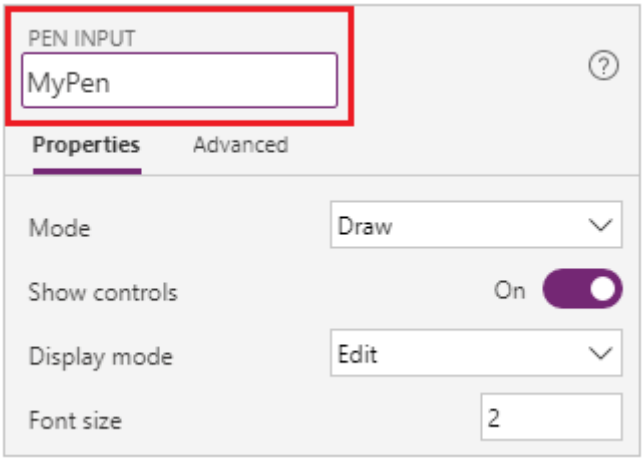
   d. Select the **Drawings** table, and then select **Connect**.



Now, the Drawings table is listed as a data source.

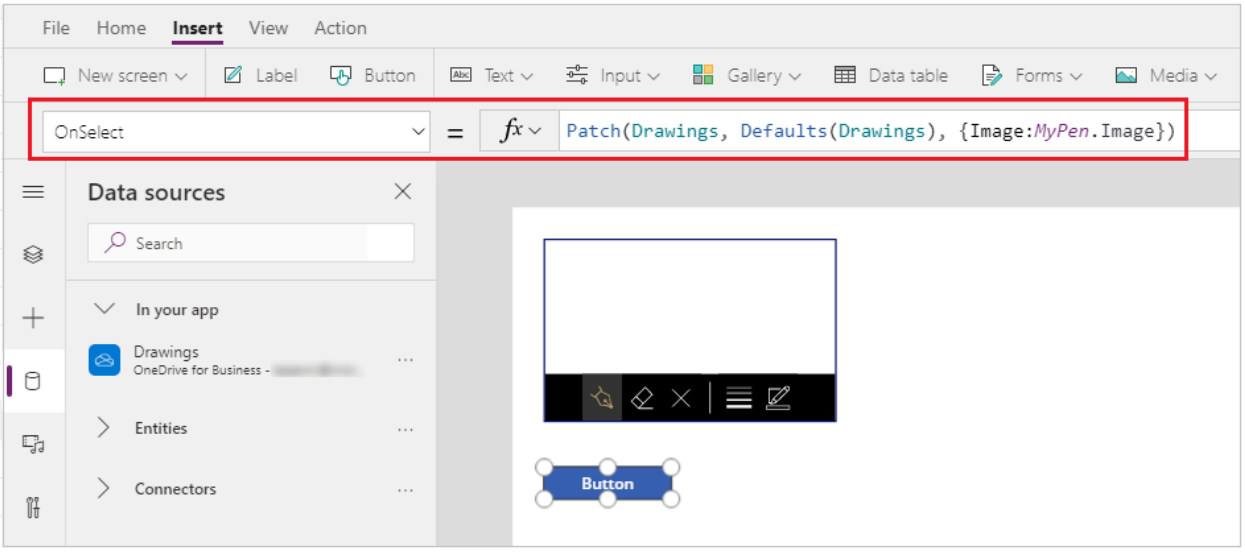
6. Select **Insert** > **Input**, and then select **Pen Input**.

7. Rename the new control **MyPen**:



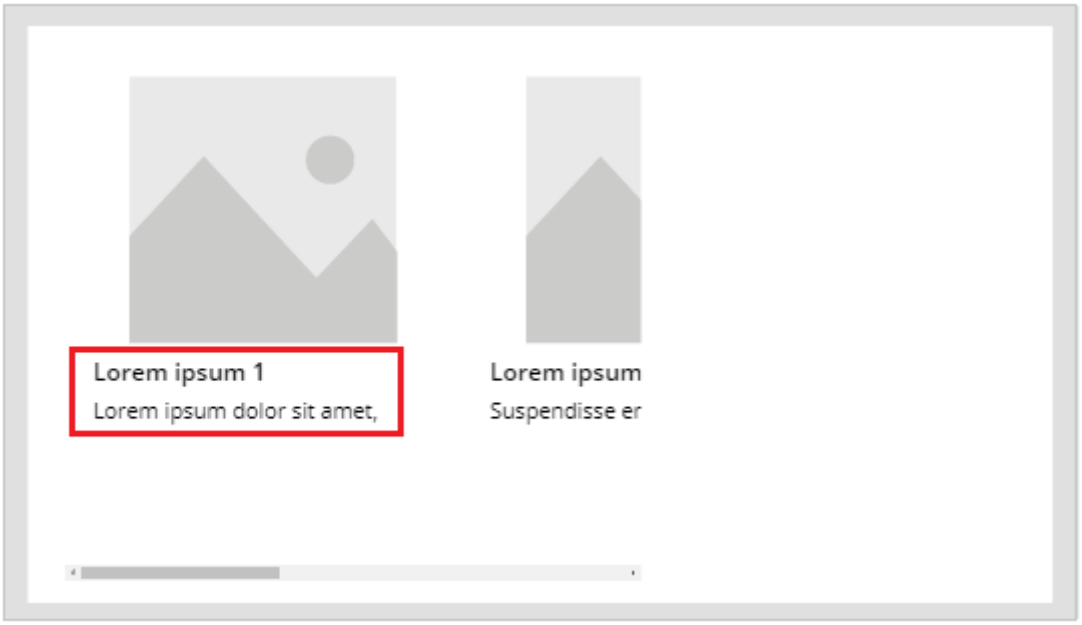
8. On the **Insert** tab, add a **Button** control, and set its **OnSelect** property to this formula:

   Patch(Drawings, Defaults(Drawings), {Image:MyPen.Image})



9. Add a **Horizontal** gallery control (**Insert** tab > **Gallery**).

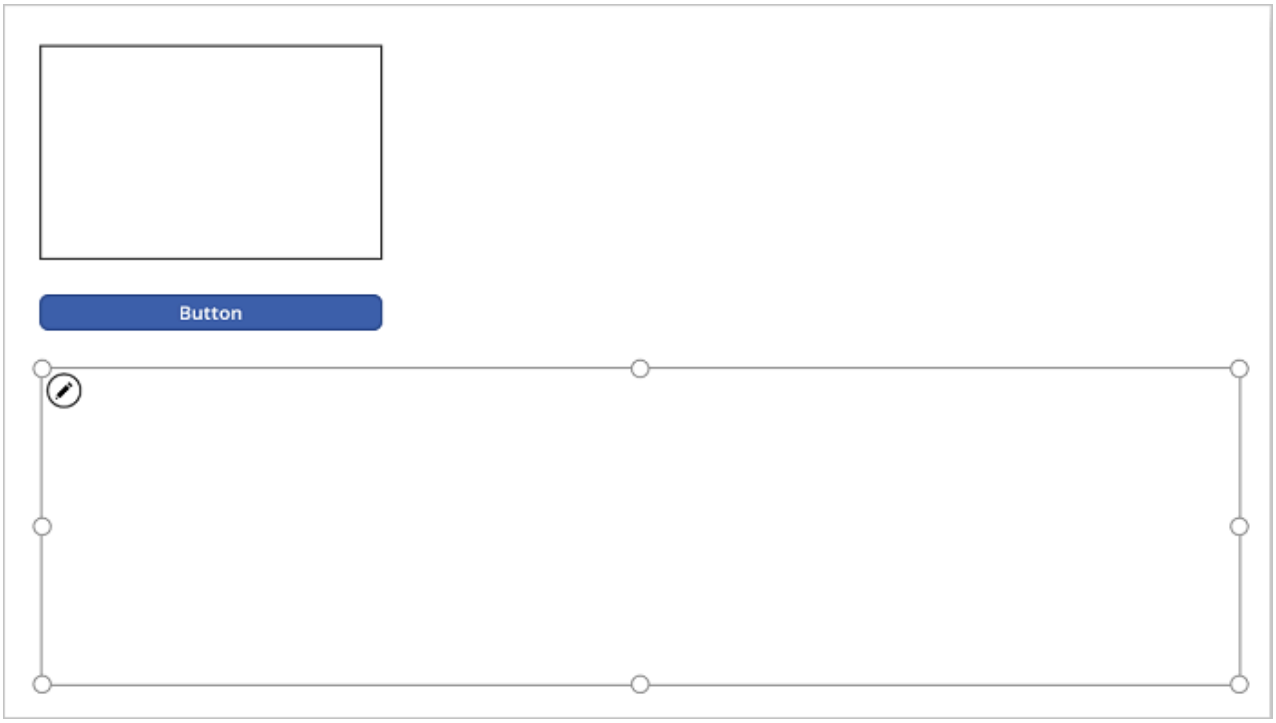
10. Optionally, select the text field and then the heading field under the first image, and delete them to keep only images on the screen.



    If you see a formula error, press **Ctrl+Z** to undo the delete. Then, delete the *Subtitle* field and the *Title* field.

11. Set the gallery **Items** property to `Drawings`. The **Image** property of the gallery control is automatically set to `ThisItem.Image`.

    Arrange the controls so that your screen resembles the following image:



12. Press F5, or select Preview ( ).

13. Draw something in MyPen, and then select the button.

    The first image in the gallery control displays what you drew.

14. Add something else to your drawing, and select the button.

    The second image in the gallery control displays what you drew.

15. Close the preview window by pressing Esc.

    In your cloud storage account, a **SavePen_images** folder is automatically created. This folder contains your saved images with IDs for their file names. To show the folder, you might need to refresh the browser window by, for example, pressing F5.

    > **Note**
    >
    > The folder name might be different if your Excel file name is different. For example, if your file name is Pen.xlsx, the folder name is Pen_images.

    In **SavePen.xlsx**, the **Image** column specifies the path to the new images.

16. Save and publish your app.

17. Share app with others.

Remove unused media

To clean up your app or reduce its size, remove all unused media files by selecting **Media** > **Remove unused media**.



Known limitations

- To enable better performance while loading an app, the following size restrictions apply:
  - The total size of all media files uploaded to an app can't exceed 200 MB.
  - The maximum size of an individual media file in an app can't exceed 64 MB.
- Supported media file types: `.jpg, .jpeg, .gif, .png, .bmp, .tif, .tiff, .svg, .wav, .mp3, .mp4`.
- When connecting your app to cloud-based storage, Cloud-storage known limitations apply.

See also

- Control reference
- Working with formulas

Last updated on 02/27/2026

Use Copilot answer control for canvas apps (preview)

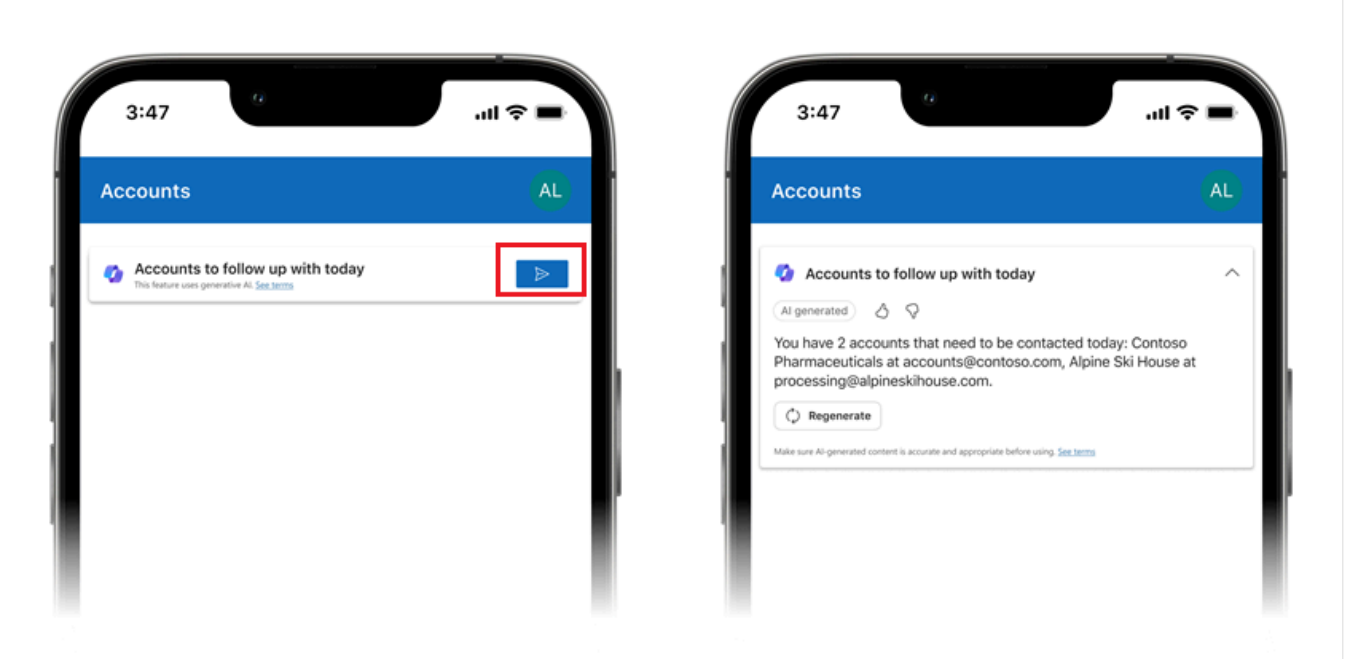
Summarize this article for me

[This article is prerelease documentation and is subject to change.]

Makers can use the Copilot answer control to set predefined questions that end users can use to get AI-generated information with a single click. The control is powered by Microsoft Copilot and is designed for mobile users to provide them with immediate answers to frequently asked questions.

> **Important**
>
> Starting February 2, 2026, this feature will be removed. Existing apps with this control will remain functional for a limited time but will eventually no longer be supported. Microsoft 365 Copilot Chat in canvas apps is the new the recommended solution for natural language interactions in canvas apps. To ensure a smooth transition and prevent any disruption, we recommend that you migrate to Microsoft 365 Copilot chat as soon as it becomes available in your environment.

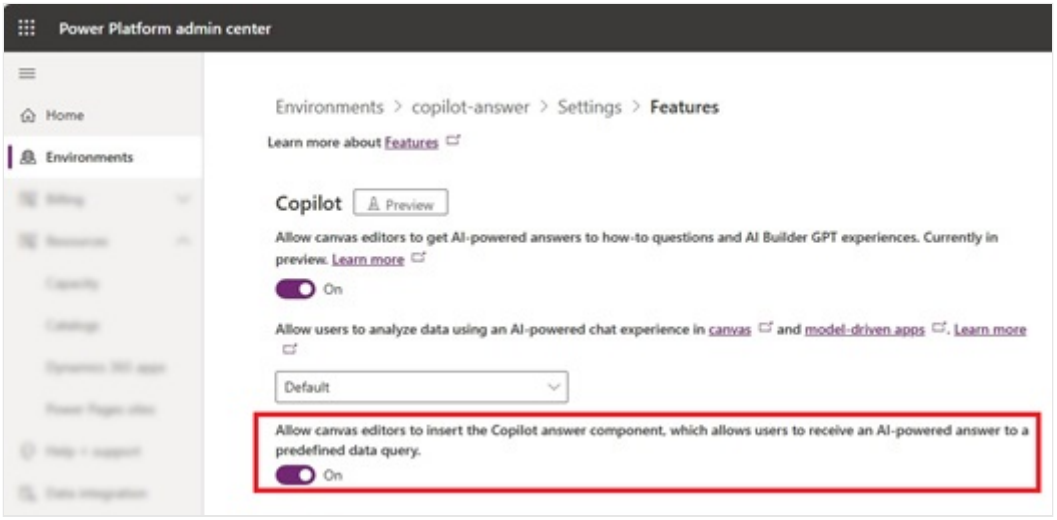
> **Note**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.



Enable Copilot answer control for your environment

To use the Copilot answer control in a canvas app, an admin must first turn on the feature, **Allow canvas editors to insert the Copilot answer component, which allows users to receive an AI-powered answer to a predefined data query** in their environment.

This setting can be found in the Power Platform admin center by going to **Environments** > [select an environment] > **Settings** > **Product** > **Features**. For more information, see Manage feature settings.

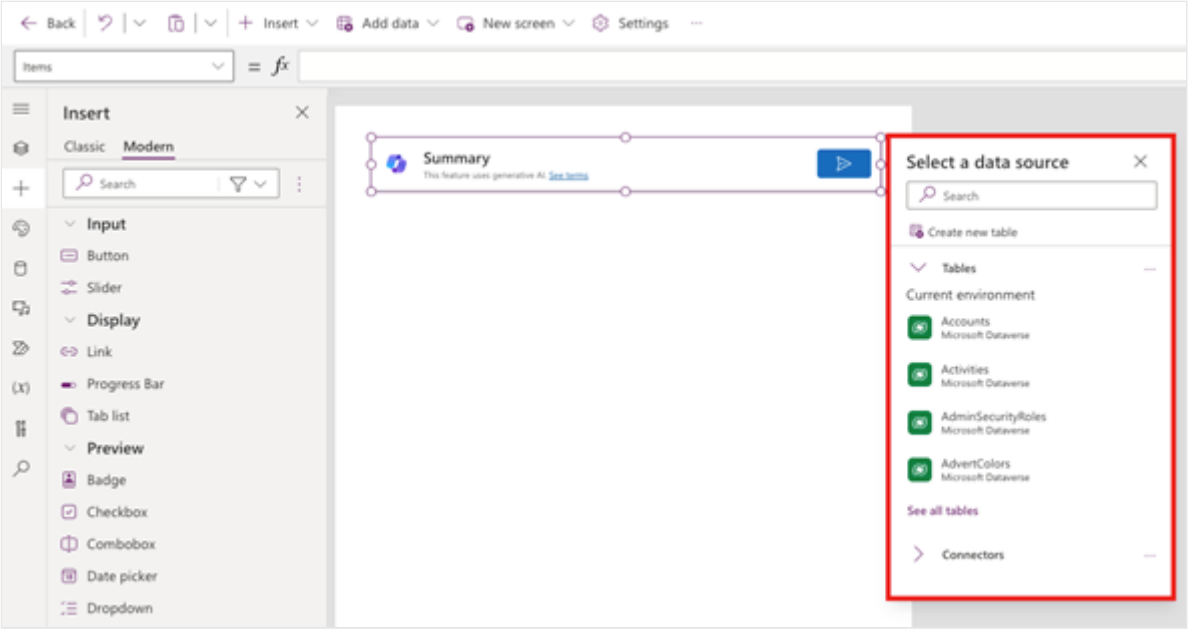


Add Copilot answer control

With your canvas app open for editing in Power Apps Studio:

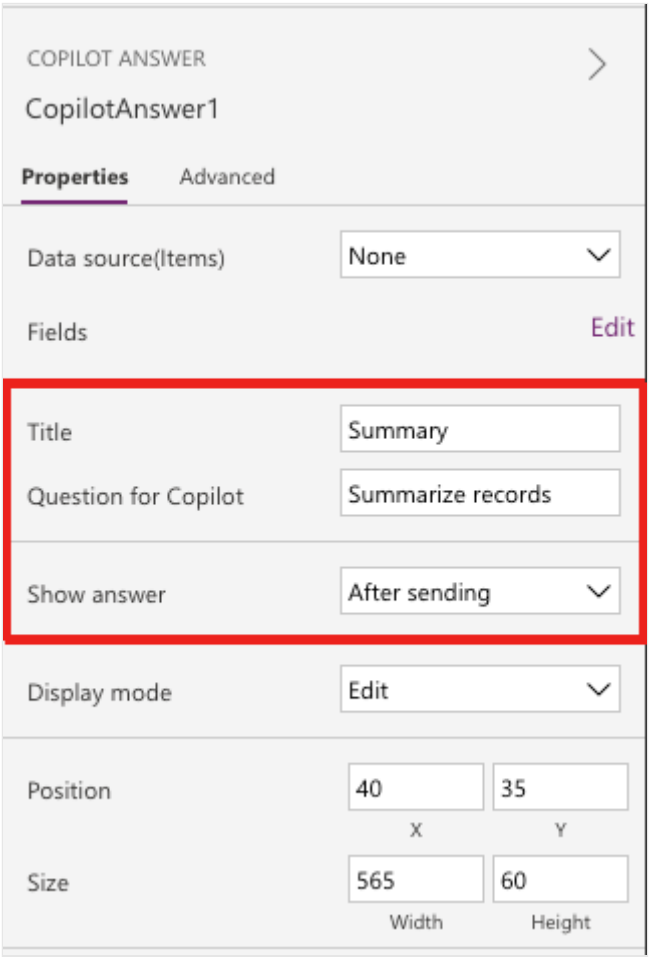
1. On the command bar, select **Insert** and then select **Copilot answer (preview)**

2. When the **Copilot answer (preview)** control is added to the canvas app, select a data source from the pane. Copilot only supports Dataverse tables.



3. On the control properties pane, select the **Properties** tab.

4. In the properties, enter the **Title**, **Questions for Copilot**, and **Show answer** for the question. You can also edit other control properties as needed.



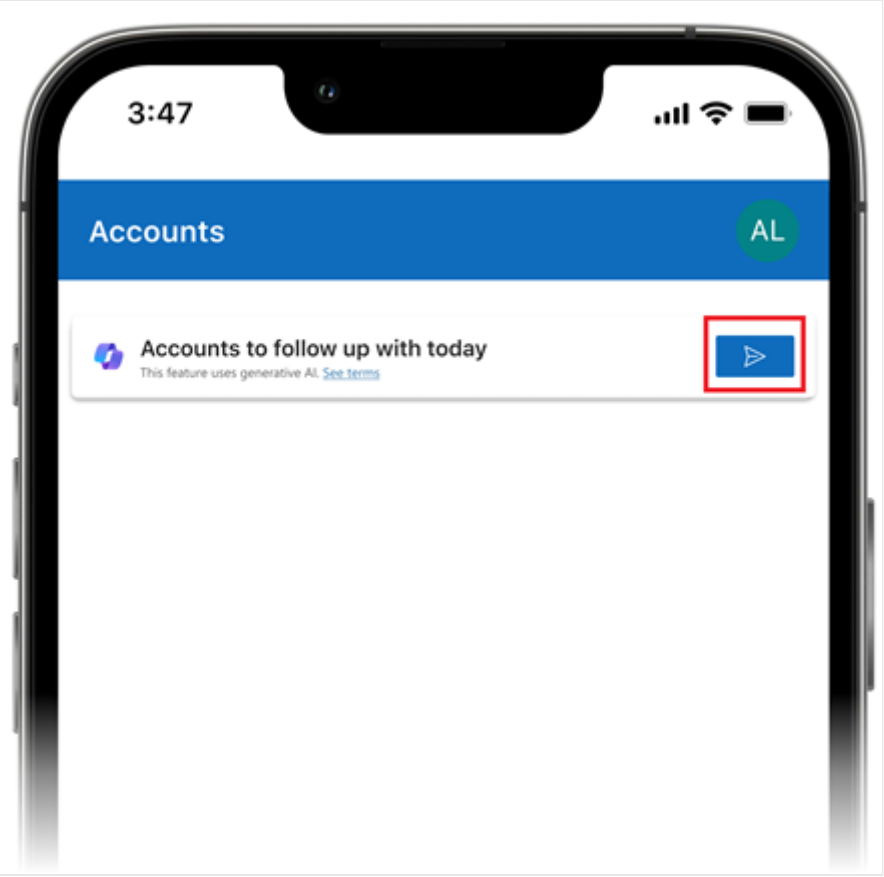
5. When you're done, save and publish your app.

Use Copilot answer control

End users can regenerate responses and provide feedback on the quality of responses.

Follow these steps to see answers generated by the Copilot answer control:

1. Open a canvas app on Power Apps mobile and navigate to the Copilot answer control in your app.
2. On the Copilot answer control select, **Send**.



Last updated on 02/24/2026

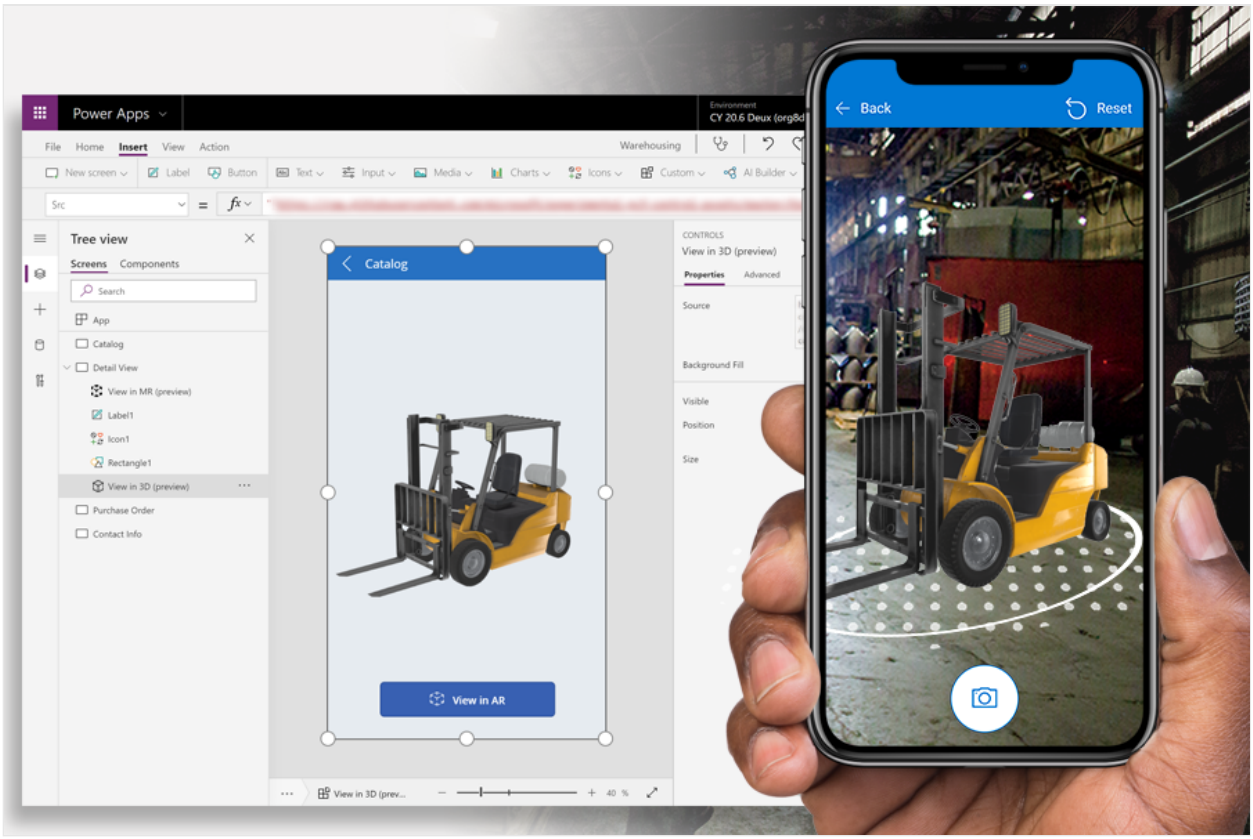
Add mixed reality controls to your canvas app

Article • 09/20/2022

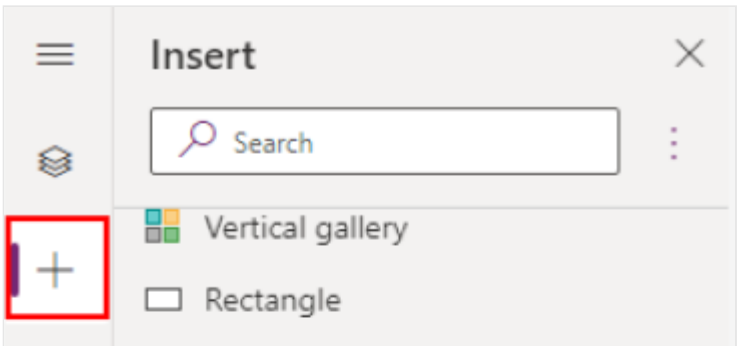
Add mixed reality (MR) controls to your canvas apps to interact with the real world in 3D.

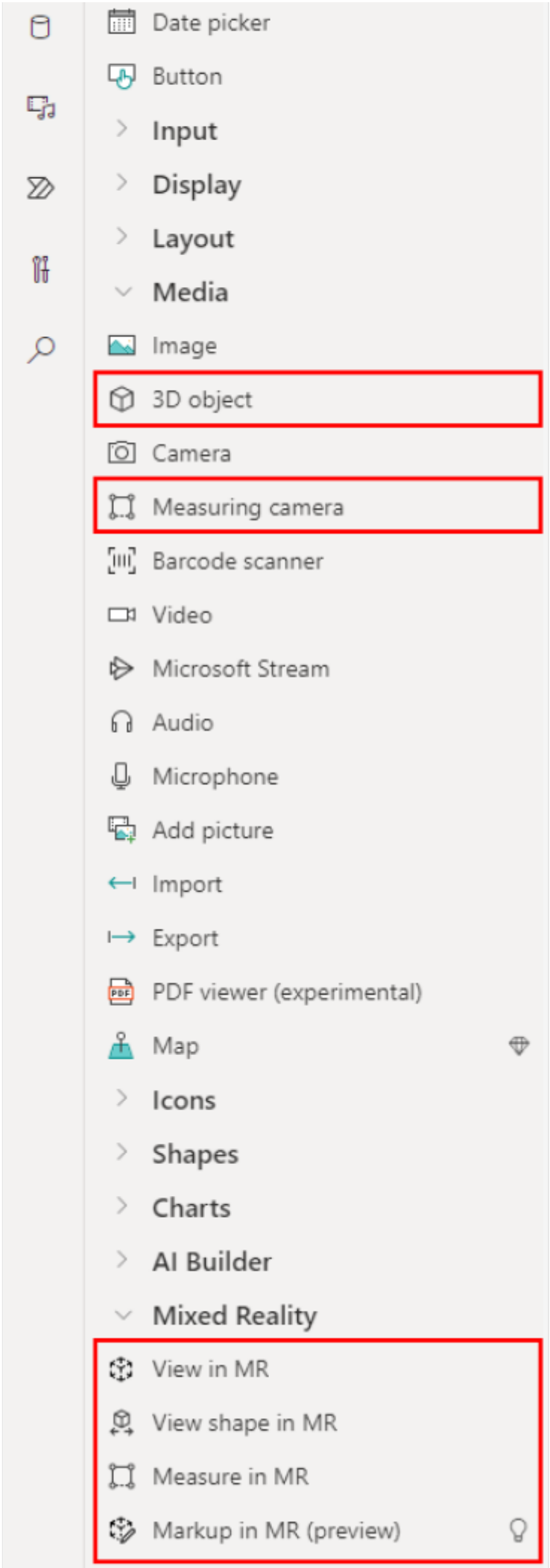
You can use MR controls to:

- View and manipulate 3D content with 3D object
- Overlay 3D content and 2D images on the camera feed with View in mixed reality
- Measure distance, area, and volume with Measure in mixed reality
- Identify spaces and locations by painting 3D lines and arrows with Markup in mixed reality



You'll find the controls on the **Insert** pane, under **Media** and **Mixed Reality**:




Date picker
Button
Input
Display
Layout
Media
Image
3D object
Camera
Measuring camera
Barcode scanner
Video
Microsoft Stream
Audio
Microphone
Add picture
Import
Export
PDF viewer (experimental)
Map
Icons
Shapes
Charts
AI Builder
Mixed Reality
View in MR
View shape in MR
Measure in MR
Markup in MR (preview)

> **Tip**
>
> The MR controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the **Babylon sandbox** should work in Power Apps through this shared MR platform. If your content works in Babylon but not in Power Apps, ask a question in the **Power Apps community forum**.
>
> **Read more about our integration with Babylon**.

Prerequisites

The device you use to create an app with MR controls, such as your PC, doesn't need to be MR-capable. Mobile devices that run the app must have specific hardware and software to use MR controls.

Android devices

Android devices need to have Google Play Services for AR (known more commonly as ARCore) installed. ARCore should be installed automatically on MR-capable devices. If necessary, download Google Play Services for AR from the Google Play Store.

For more information about ARCore and devices that support it, see the list of supported devices on the Google ARCore support site.

The experience is a little different for devices in China, which don't ship with the Google Play Store. ARCore must be downloaded from specific app stores in China.

iOS (Apple) devices

iPhones and iPads with specific hardware can run MR apps using the ARKit framework. For more information about MR and iOS devices that support it, see the list of supported devices on the Apple augmented reality website. (Scroll to the bottom of the page to see the list.)

Mixed reality controls

Start adding the controls to your apps:

- View 3D content with the **3D object** control.

- View images and 3D content in the real world with the **View in mixed reality** control.
- Measure distance, area, and volume with the **Measure in mixed reality** control.
- Create and view 3D shapes with the **View shape in mixed reality** control.
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

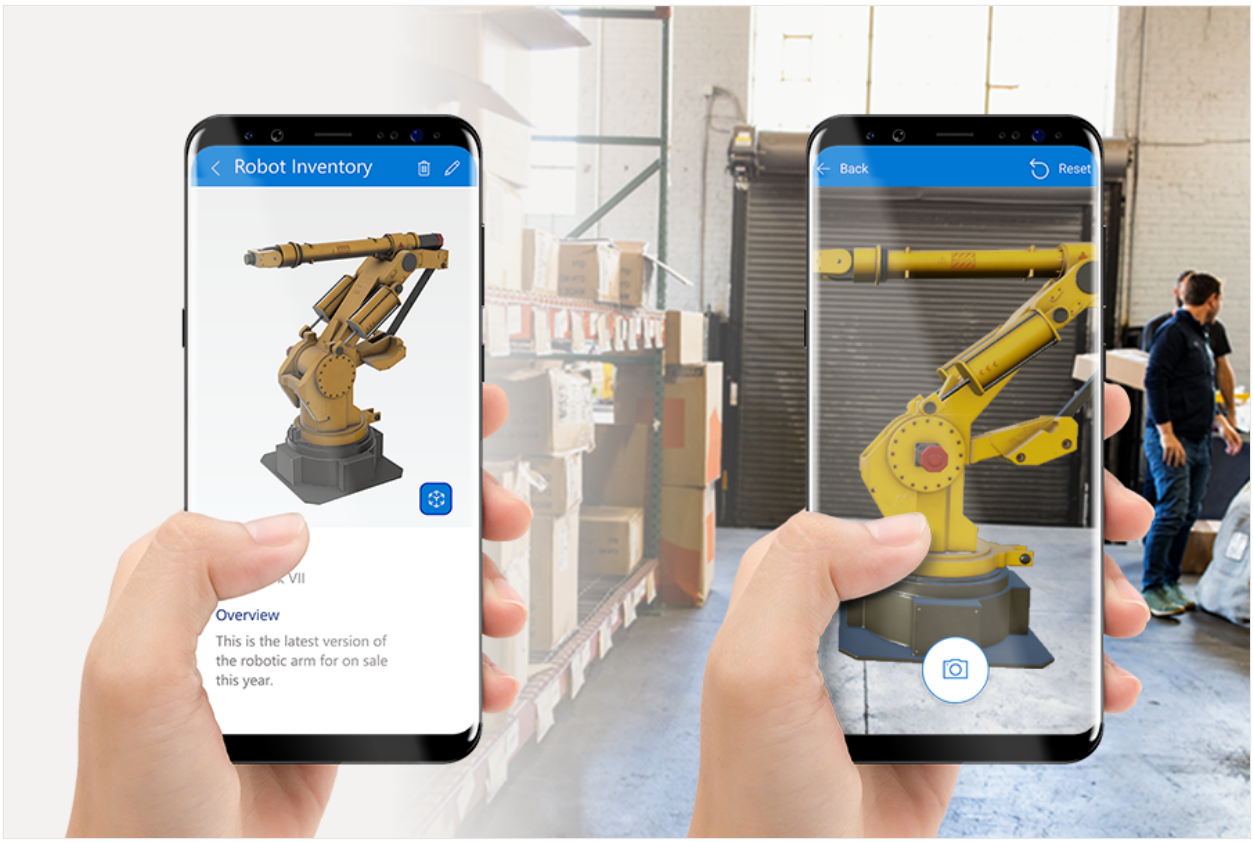
Build a 3D mobile app with Power Apps in mixed reality

Improve sales conversation by placing products directly in your customer's space with Mixed Reality

View 3D content in Power Apps

Article • 05/09/2022

Easily add interactive 3D content to your canvas apps. You can select from a gallery of 3D content then rotate and zoom into the model with simple gestures.



Important

Your 3D content must be in the .glb, .obj, or .stl file formats. You can **convert your existing 3D models into the .glb file format** from a variety of 3D formats.

Tip

The mixed reality (MR) controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the **Babylon sandbox** should work in Power Apps through this shared MR platform. If your content works in Babylon but not in Power Apps, ask a question in the **Power Apps Community Forum**. (Tag it with "mixed reality.")

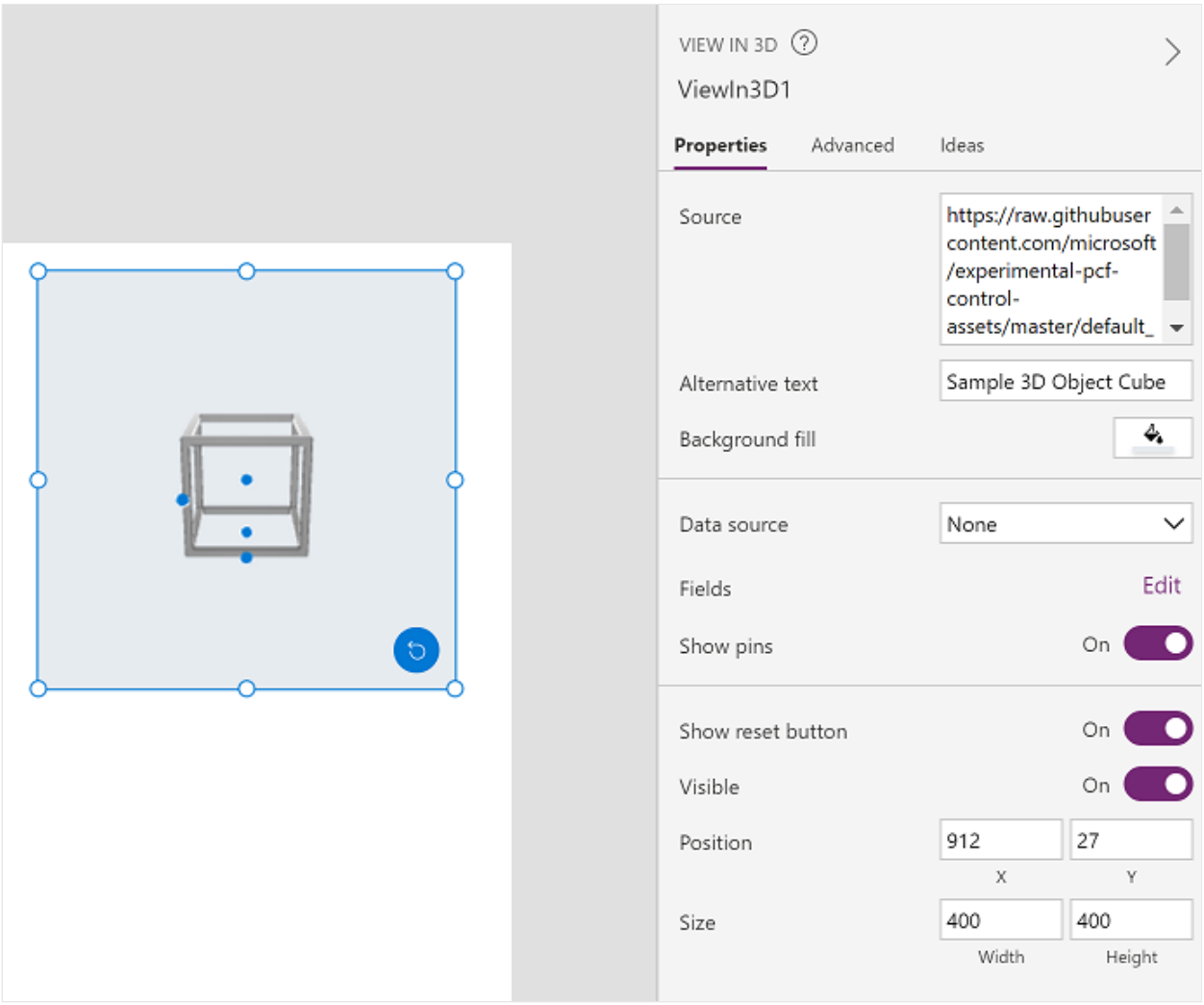
Add 3D object control to an app screen

With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Media**.
2. Select **3D object** to place the control in the app screen, or drag the control to the screen to position it more precisely.

Key properties

Change the 3D content control's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Property	Description	Type	Location
Source	Identifies the object file to display. The **3D object** control supports loading models from various sources. See Loading external 3D models for details.	Not applicable	Properties; Advanced: **Source**
Alternative text	Specifies the text to be displayed if the model can't load or if the user hovers over the model.	String	Properties; Advanced: **AltText**

Property	Description	Type	Location
Background fill	Sets the control's background color.	Color picker	Properties; Advanced: **BackgroundFill** (accepts RGBA or HTML hexadecimal color codes)
Pins(Items)	Shows pins on the model at specific coordinates, provided in a data source (**Items**). If *None*, no pins are shown.	Data table	Properties; Advanced: **Items**
Show pins	Shows the pins described in **Items**.	Boolean	Properties; Advanced: **ShowPins**
MaxPins	Specifies the maximum number of pins that can be shown on the model.	Integer	Advanced
PinsX	Specifies the location of pins on the X axis in 3D coordinate space.	Floating point number	Advanced
PinsY	Specifies the location of pins on the Y axis in 3D coordinate space.	Floating point number	Advanced
PinsZ	Specifies the location of pins on the Z axis in 3D coordinate space.	Floating point number	Advanced
Show reset button	Shows or hides a button that resets the model to its initial state.	Boolean	Properties; Advanced; **ShowReset**
OnModelLoad	Contains behavior formula that runs when a model is loaded.	Event	Advanced
OnChange	Contains behavior formula that runs when any property of the control is changed.	Event	Advanced
OnSelect	Contains behavior formula that runs when the user selects a pin or the control.	Event	Advanced

Additional properties

Property	Description	Type	Location

Property	Description	Type	Location
ContentLanguage	Determines the display language of the control, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Dropdown list	Advanced
Position	Places the upper-left corner of the control at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the control using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
TabIndex	Specifies the order the control is selected if the user navigates the app using the Tab key.	Integer	Advanced
Tooltip	Determines the text to display when the user hovers over a pin.	String	Advanced
Visible	Shows or hides the control.	Boolean	Properties; Advanced: **Visible**

Performance considerations

We recommend that you use one 3D control on a screen for the best user experience. Multiple instances of the **3D object** control on one screen will try to load their 3D models at the same time, which can severely degrade the performance of your app.

Other mixed reality controls

- View 3D content and images in the real world with the **View in mixed reality** control.
- Measure distance, area, and volume with the **Measure in mixed reality** control.
- Create and view predefined 3D shapes with the **View shape in mixed reality** control.
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

Create an app with 3D and mixed reality controls

Load 3D models in canvas apps

Article • 02/25/2023

Load a 3D model in your canvas apps from a variety of sources. You can get models from attachments or media content, a direct URL, or a Base64-encoded URI (uniform resource identifier).

Make sure your 3D models are optimized for use with Power Apps to minimize load times.

Loading 3D models from common connectors

Loading 3D models from attachments or media content depends on how a data connector is supported. To check if a data connector will work the mixed reality controls, add a label control to the canvas app and set the **Text** property to the data source. If the label text starts with `appres://`,then that data connector should work with the **3D object control**.

> **Tip**
>
> You can rename a .glb file extension to .jpg and directly upload it to the app through the Media tab.

Load 3D models from Microsoft Lists

First, create a list in SharePoint and add an entry for each 3D model that you want to have in your app.

1. Create a list using Microsoft Lists.
2. Select the **+ Add column** column heading and then select **Show/hide columns**.
3. Select **Attachments** and then select **Apply**.
4. Add an entry to the list. In the entry form, select **Add attachments** and select your 3D model file.
5. Repeat for each model you want to include in your app.

Then, add a gallery to your app, set its source to the list, add a **3D object** control, and set its source to the gallery.

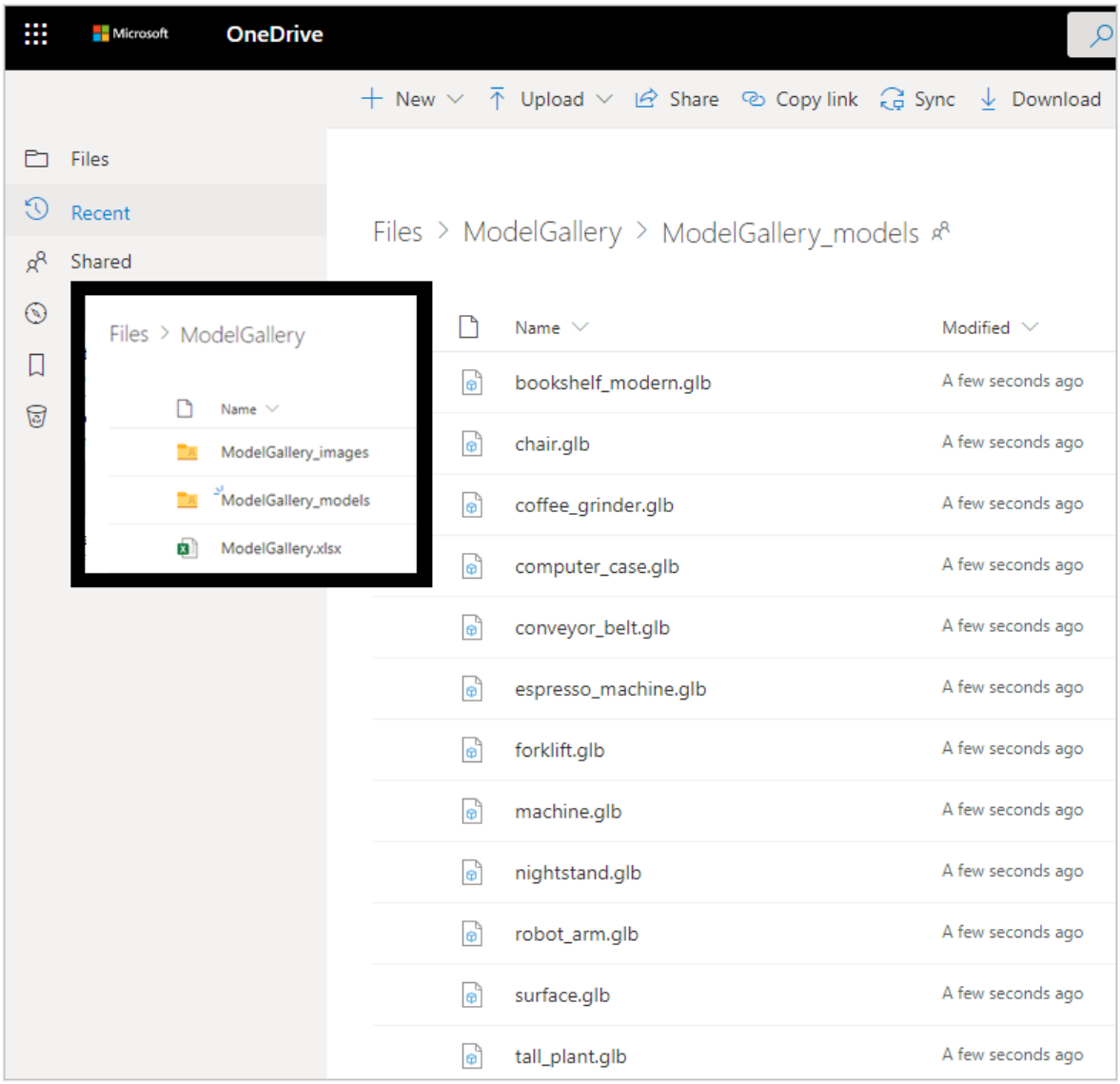
1. Add a gallery in Power Apps Studio.
2. Set the gallery data source to the list.

3. Add the **3D object** control.
4. In the **Advanced** properties tab, set **Source** to **First(Gallery1.Selected.Attachments).Value**.

Load 3D models from an Excel workbook

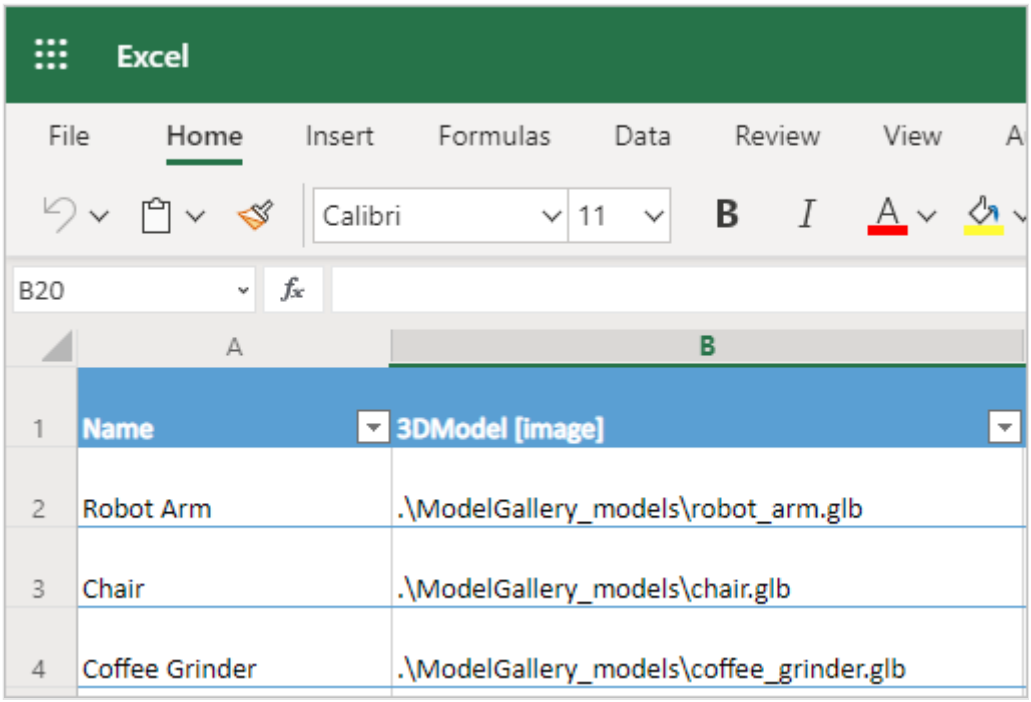
First, create an Excel workbook in OneDrive in the same folder that contains your model files. Add a table with rows for each model that you want to have in your app.

1. Create an Excel workbook and save it in the OneDrive folder that contains your model files.



2. In the workbook, create a table with columns named **3DModel [image]** and **Name**.

3. Add a row for each model you want to show in the app gallery. Enter a label for the model in the **Name** column and the relative file path to the model file in the **3DModel [image]** column.



4. Close the workbook.

Then, add a gallery to your app, set its source to the Excel workbook, add a **3D object** control, and set its source to the gallery.

1. Add a gallery in Power Apps Studio.
2. Use the OneDrive connector to set the gallery data source to the Excel workbook.
3. Add the **3D object** control.
4. In the **Advanced** properties tab, set **Source** to **Gallery1.Selected.'3DModel'**.

Load 3D models from a URL

The **Source** property of the **3D object** control can be the URL of a 3D model file.

The 3D model file must be on a server that doesn't have restrictive cross-origin resource sharing (CORS) settings. The hosting server must permit cross-origin requests from *powerapps.com*. You can use Dropbox or GitHub to host your files and get a CORS-compliant URL.

Host your 3D model files in Dropbox

1. Upload a 3D model file to Dropbox and select **Share**.
2. Generate a public download link. For example, *https://www.dropbox.com/s/rANdoMGeneR4tedLink/my-file.glb?dl=0*.
3. Modify the URL like this: replace **www** with **dl**, and remove **?dl=0** at the end.

You now have a direct-access URL (in our example, *https://dl.dropbox.com/s/rANdoMGeneR4tedLink/my-file.glb*), that you can use as the source of your 3D control.

Host your 3D model files in GitHub

1. Make sure the 3D model file is stored in a public repo.
2. Get the URL of the file. For example, *https://github.com/microsoft/experimental-pcf-control-assets/blob/master/robot_arm.glb*.
3. Modify the URL like this: remove **/blob/**, and replace **https://github.com** with **https://raw.githubusercontent.com**.

You now have a CORS-compliant URL (in our example, *https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/robot_arm.glb*), that you can use as the source of your 3D control.

Load Base64-encoded 3D models

The **Source** property of the **3D object** control can be a Base64-encoded 3D model data URI that is in the format *data:base64,<Base64-encoded content>*.

> **Important**
>
> Your app may take longer to load if you use Base64-encoded models.

You can create a Base64-encoded URI of your model using Microsoft Power Automate or Microsoft Dataverse.

Create a Base64-encoded 3D model with Microsoft Power Automate

Power Automate can convert 3D model files stored in a SharePoint document library to Base64 using the **dataUri(base64(*file content*))** expression.

In the following example, a document library named *3DModelBase64Library* and a list named *3DModelBase64* exist in the same SharePoint site. The list must include a column of type **multiple-line text**.

1. In the document library, create a flow based on the **When a new file is added in SharePoint, complete a custom action** template.

2. Set **Library Name** to *3DModelBase64Library* (the name of the document library in this example).

3. Add a step, **Get file content from SharePoint**.

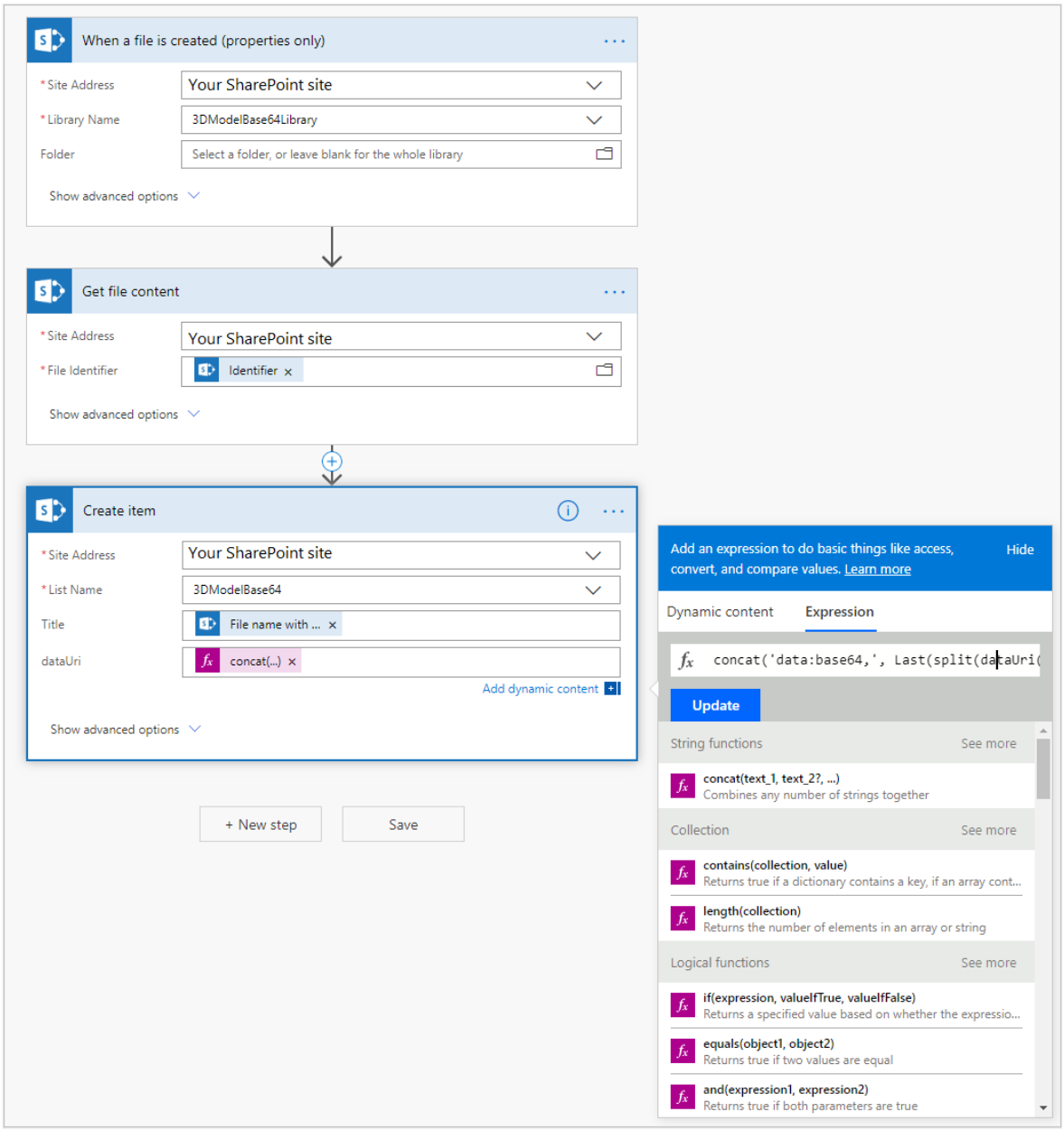
4. Set **File Identifier** to **Identifier**.

5. Add a step, **Create item from SharePoint**.

6. Set **List Name** to *3DModelBase64* (the name of the list in this example) and **Title** to **File Name with extension**.

7. Set **dataUri** to the following expression:

```
concat('data:model/gltf-binary; base64,',
Last(split(dataUri(base64(body('Get_file_content'))), ',')))
```



The flow runs when a file is added to the document library, converting the file to a Base64-encoded data URI.

In Power Apps Studio, connect the **3D object** control to the list using the SharePoint data connector. Set the control's **Source** property to the Base64-encoded data URI.

Create a Base64-encoded 3D model with Microsoft Dataverse

The Note (Annotation) table in Microsoft Dataverse converts any file attached in the **Document** field to Base64.

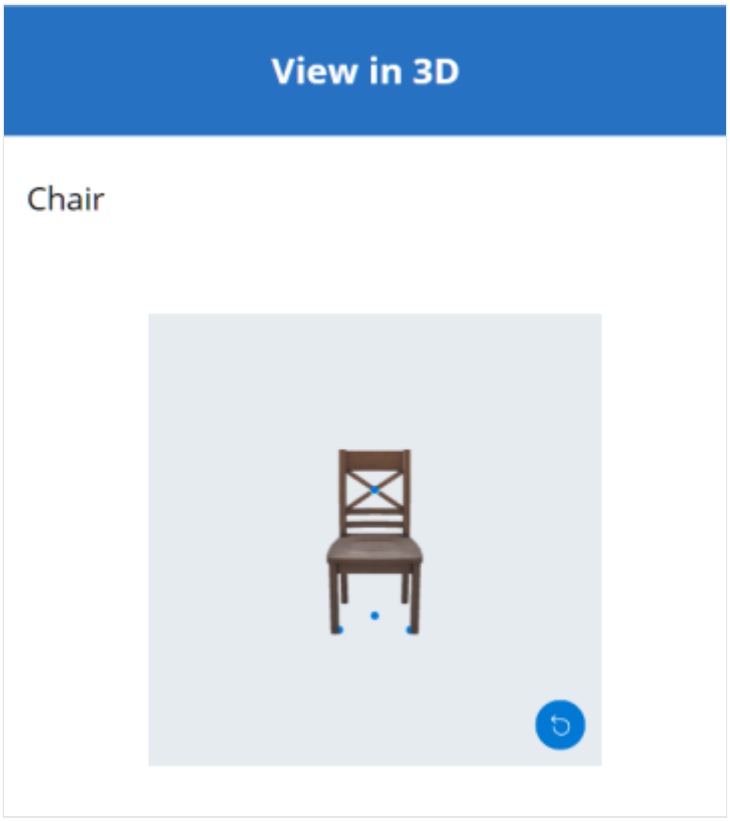
Known constraints when loading 3D models from URLs in canvas apps

- The security architecture of Power Apps requires HTTPS links, not HTTP.
- The server that hosts the model files must not require authentication and must be CORS-compliant.

Add pins to 3D models in Power Apps

Article • 07/15/2022

Add pins (shown as circular markers) to 3D models in your canvas apps. Users can add their own pins and view information about pins they select.



Add predefined pins to a 3D model from an Excel workbook

The easiest way to add pins to a 3D model is to load them from an Excel workbook.

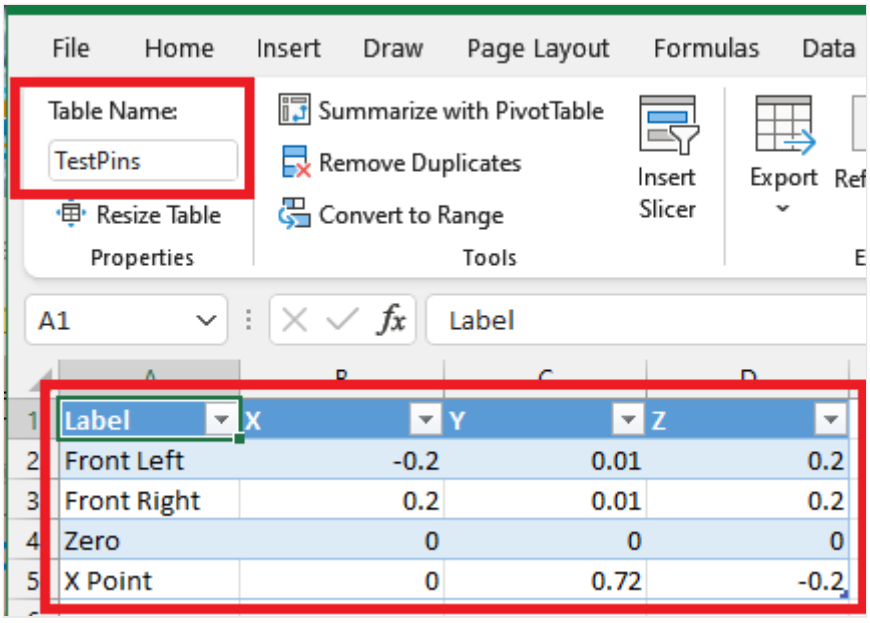
The data source for the pins must be a named table in Excel. The table must have four columns. Each column corresponds to a property of the **3D object** control. Each row is plotted as a pin on the model. The rows determine the placement of the pins in a three-dimensional coordinate system.

> **Important**
>
> The mixed-reality (MR) controls in Power Apps use a left-handed coordinate system.

Expand table

Column	Corresponds to
Label	ItemsLabels
X	PinsX
Y	PinsY
Z	PinsZ

Here's an example of a table named "TestPins" that contains coordinates to place four pins on a 3D model:



To try this example, start by creating a table in Excel.

1. Copy the following sample data and paste it in cell A1 of a new Excel worksheet.

Expand table

Label	X Pins	Y Pins	Z Pins
Front Left	-0.2	0.01	0.2
Front Right	0.2	0.01	0.2
Zero	0	0	0
X Point	0	0.72	-0.2

2. Select one of the pasted cells. On the **Home** tab, select **Format as Table**, choose a style, and then select **OK**.

3. On the **Table Design** tab under **Table Name**, enter a name such as *TestPins*.

4. Save the workbook and close it.

Next, add the table as a data source to display pins on a 3D model.

1. In Power Apps Studio, insert a 3D model or select an existing model.
2. Set the **3D object** control's **Source** property to *https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/chair.glb* (or any other model file of your choosing).

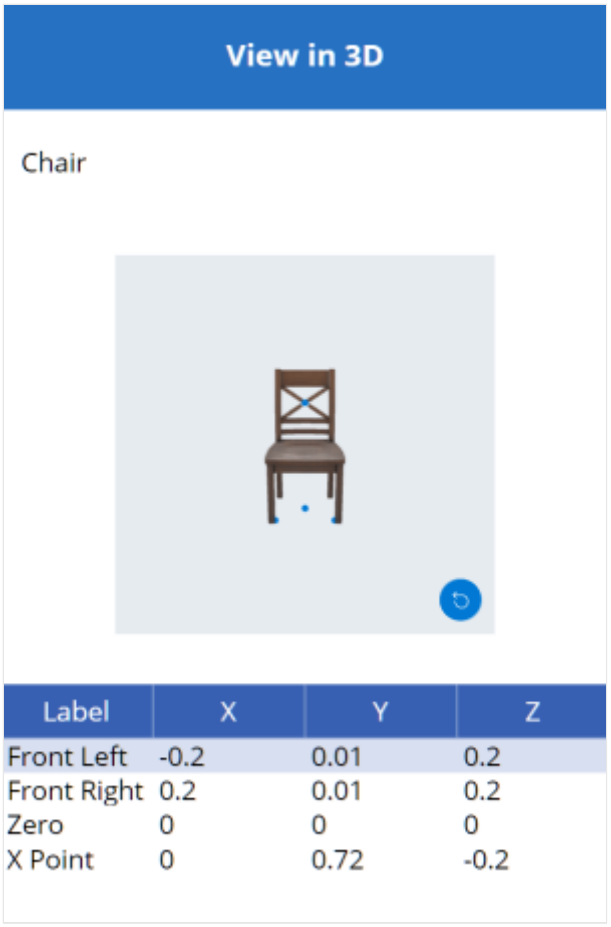
> **Note**
>
> The first instance of the **3D object** control is named *3DObject1* by default and that's the name we'll use in the following examples. Additional instances of the control would be referred to as *ViewIn3D2*, *ViewIn3D3*, and so on.

1. On the model's **Properties** tab, select the **Pins(Items)** dropdown list.
2. Type *excel* in the search box, and then select **Import from Excel**.
3. Locate the Excel workbook you saved earlier and open it.
4. Select the name you gave the table in Excel and then select **Connect**.
5. On the **Advanced** tab, find **PinsX**, **PinsY**, and **PinsZ** and enter the name of the corresponding column in the table. (In this example, enter *Label* in **ItemsLabels**, *X* in **PinsX**, and so on.) Enclose the column name in quotation marks.

Pins appear on the model at the locations described by the coordinates.

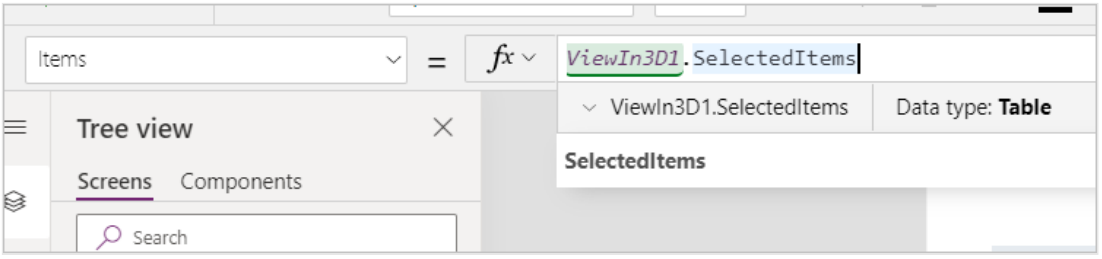
Show the label and coordinates of pins on a 3D model

Use a data table to display the label and coordinates of selected pins in a canvas app.



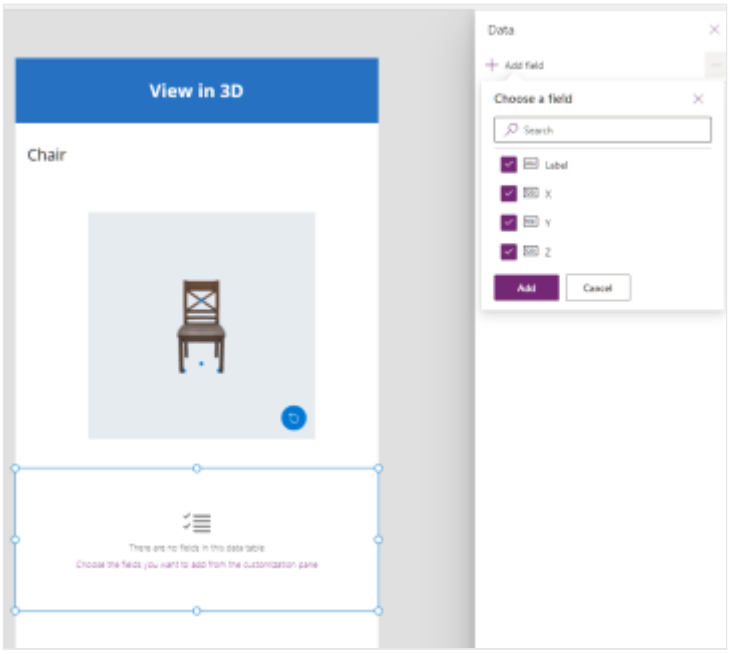
With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Layout**.
2. Select **Data table** to place a table in the app screen, or drag the control to the screen to position it more precisely.
3. Choose to show information about all pins or only the selected pin. With the data table selected:
   - **Show all pins:** In the **Properties** pane, select the **Pins(Items)** dropdown list and select the Excel table you connected to earlier.
   - **Show the selected pin only:** In the Power Apps Studio expression editor, enter *3DObject1.SelectedItems*.



4. On the **Properties** pane, select **Edit fields**, and then select **Add field**.

5. Select the columns to show, and then select **Add**.



Show the label of a selected pin on a 3D model

Use a text box to display just the label of the selected pin in a canvas app.

With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab.
2. Select **Text label** to place a text box in the app screen, or drag the control to the screen to position it more precisely.
3. In the expression editor, enter *3DObject1.Selected.Label*.

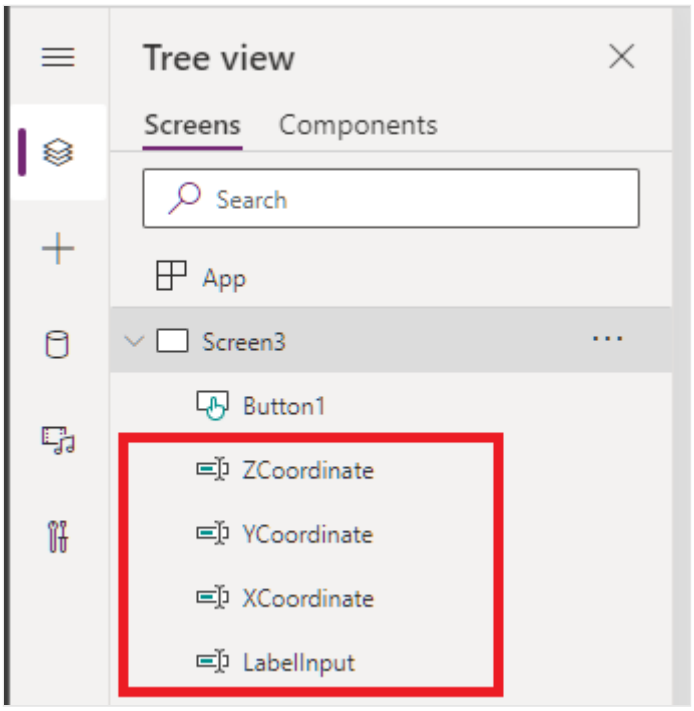
Create a collection to let users add their own pins to a 3D model

If you want to allow users to place pins on a 3D model, use a collection.

With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab.
2. Select **Text input** to place a text entry box in the app screen.
3. Copy the text box and paste it three times to create four text input controls.
4. Set the **Default** property for each control to *X*, *Y*, *Z*, and *Label*, respectively.
5. Open the **Tree view** tab.

6. Rename each control, as appropriate, to *XCoordinate*, *YCoordinate*, *ZCoordinate*, and *LabelInput*.



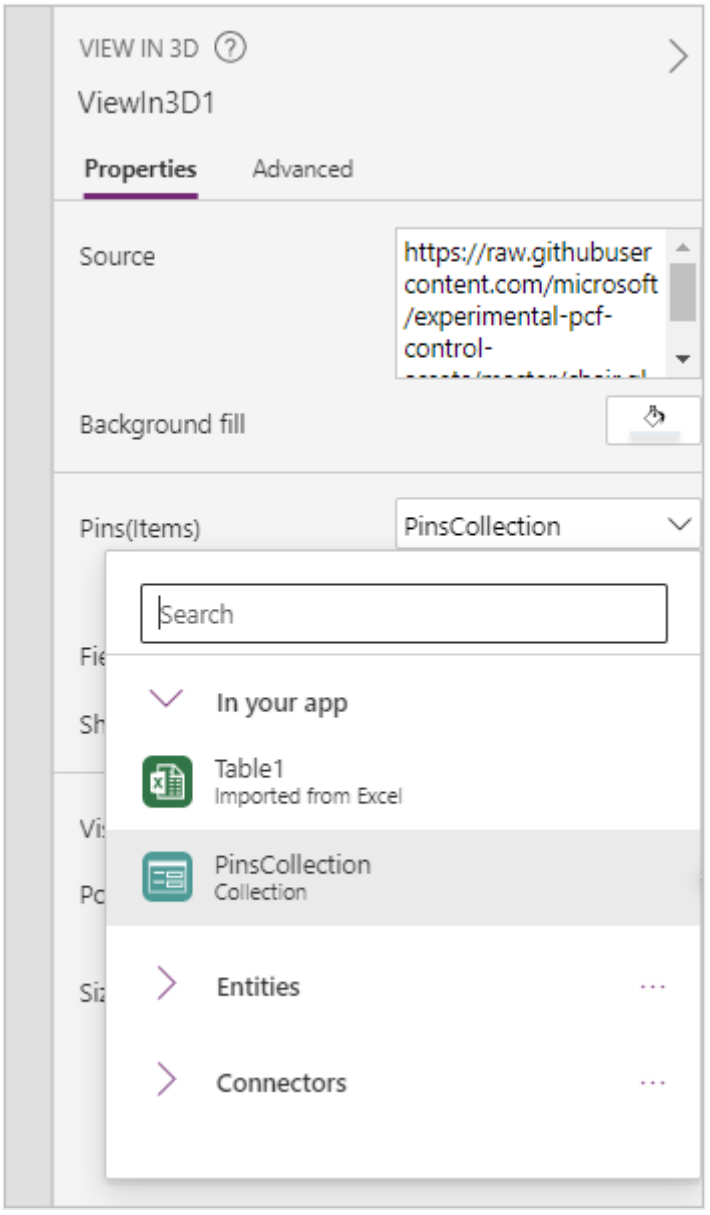
Next, add a button to append the user's data to the pin collection.

1. Open the **Insert** tab.
2. Select **Button** to place a button in the app screen.
3. Set the button's **Text** property to *Add pin*.
4. Enter the following code in the button's **OnSelect** property, exactly as shown:

```
Power Fx

Collect(
PinsCollection,
  {
    X: XCoordinate.Text,
    Y: YCoordinate.Text,
    Z: ZCoordinate.Text,
    Label: LabelInput.Text
  }
)
```

Now you'll change the data source of the pins from the Excel table to the new collection.

1. Select the 3D model.
2. On the **Properties** tab, select the **Pins(Items)** dropdown list, and then select **PinsCollection**.



3. On the **Advanced** tab, find **PinsX**, **PinsY**, and **PinsZ** and set them to *X*, *Y*, and *Z*, respectively.

When the user enters coordinates and a label and selects **Add pin**, a labeled pin is placed on the model at the entered coordinates.

Show information about pins as they're added to a 3D model

Finally, combine the previous examples by displaying information about the pins as the user adds them.

With your app open for editing in Power Apps Studio :

1. Add a data table to the screen.

2. On the **Properties** tab, select the **Pins(Items)** dropdown list and select **PinsCollection**.
3. Select **Edit fields**, and then select **Add field**.
4. Select the columns to show in the table, and then select **Add**.

Other mixed reality controls

- View 3D content in mixed reality with the **View in MR** control.
- Measure distance, area, and volume with the **Measuring Camera** control.
- Create and view predefined 3D shapes with the **View shape in MR** control.
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

Feedback

Was this page helpful? Yes No

Provide product feedback

View predefined 3D shapes in mixed reality

Article • 06/17/2022

Use the **View shape in MR** control in a canvas app to see whether a digital cube of a given size fits in a real-world space. (To compare a 3D model of an object instead, use the View in MR control.)

The **View shape in MR** control adds a button to your app. When a user selects the button, the app overlays a cube on the device camera feed. The cube's dimensions are set as a property on the control when you author the app in Microsoft Power Apps Studio. Users may not change the size of the cube while they use the app.



Tip

The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

Tip

The MR controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the **Babylon sandbox** should work in Power Apps

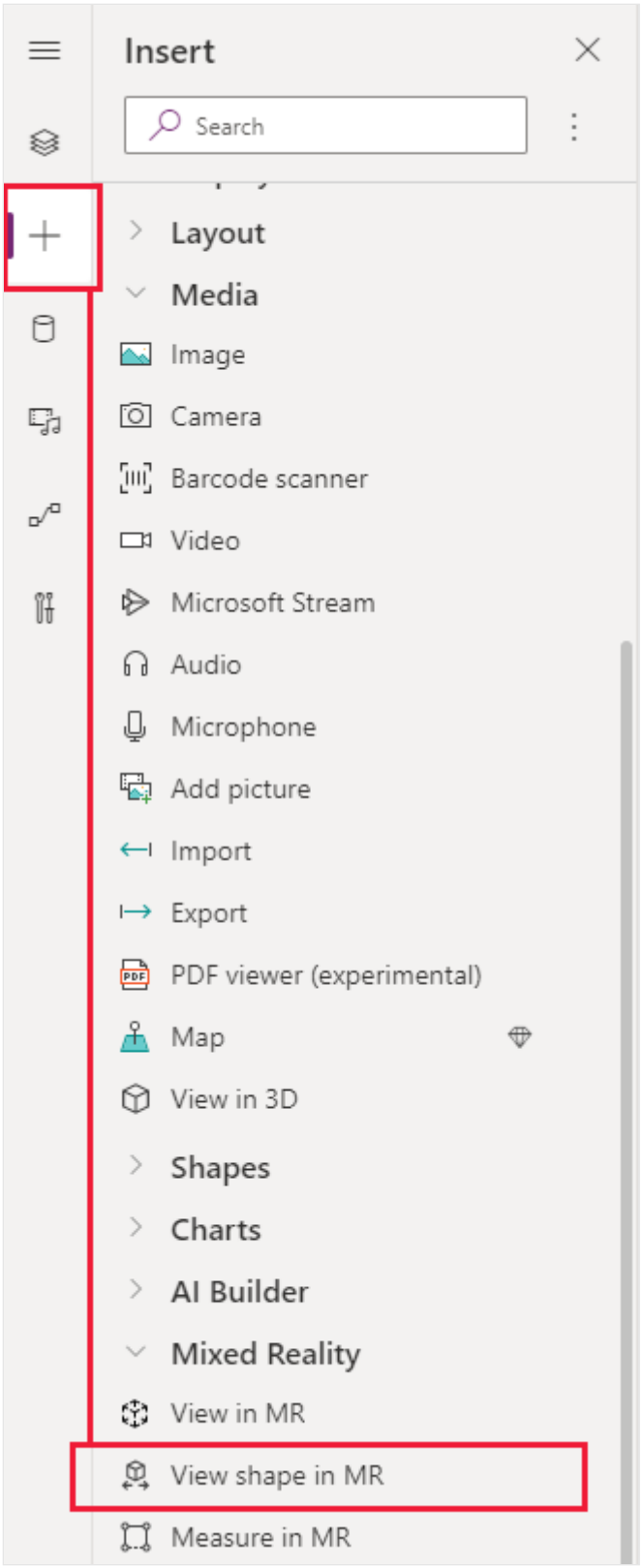
through this shared MR platform. If your content works in Babylon but not in Power Apps, ask a question in the **Power Apps Community Forum**. (Tag it with "mixed reality.")

Add a View shape in MR button to an app screen

With your app open for editing in Power Apps Studio:

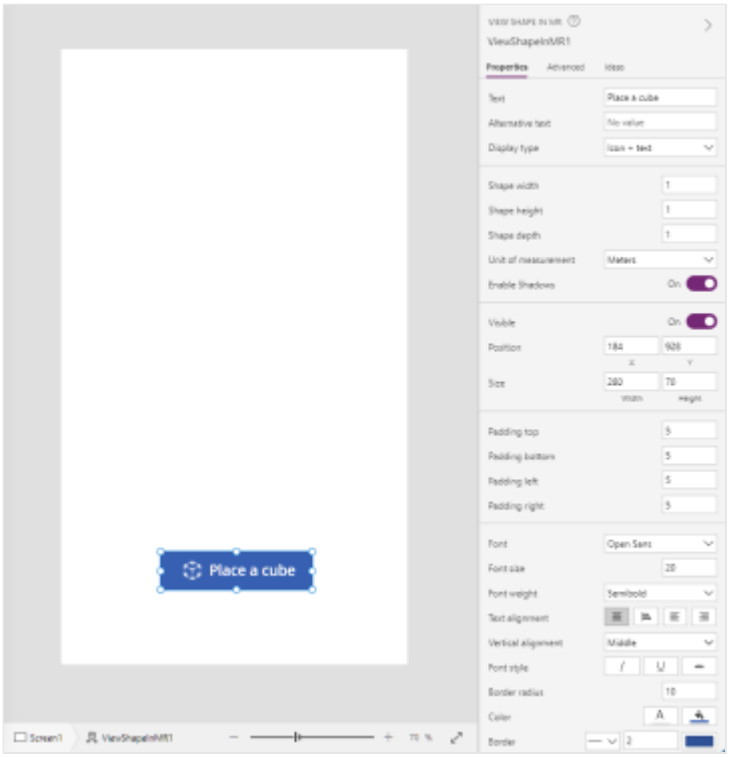
1. Open the **Insert** tab and expand **Mixed Reality**.
2. Select **View shape in MR** to place the control in the app screen, or drag the control to the screen to position it more precisely.

The control is a button that's labeled with an icon of a cube and the text *Place a cube*. Change the label and hide the icon if you like in the **Text** and **Display type** properties.



Properties

Change the **View shape in MR** button's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Property	Description	Type	Location
Text	Sets the button label text.	String	Properties; Advanced: **Text**
Alternative text	Specifies the text to be displayed if the model can't load or if the user hovers over the model.	String	Properties; Advanced: **AltText**
Display type	Determines whether the button label shows an icon of a cube, text, or both.	Drop-down selection	Properties; Advanced: **DisplayType**
Shape width	Sets the width of the cube.	Integer	Properties; Advanced: **ShapeWidth**
Shape height	Sets the height of the cube.	Integer	Properties; Advanced: **ShapeHeight**
Shape depth	Sets the depth of the cube.	Integer	Properties; Advanced: **ShapeDepth**
Unit of measurement	Sets the measurement unit used for the cube's width, height, and depth.	Drop-down selection	Properties; Advanced: **Units**
Enable Shadows	Determines whether shadows are used to enhance the 3D effect when the cube is shown.	Boolean	Properties; Advanced: **EnableShadows**

Property	Description	Type	Location
Enable Marker Detection	Determines whether marker detection will be used to to determine model position and rotation. See Use marker detection with mixed reality controls	Boolean	Properties; Advanced: **EnableMarkerDetection**
Visible	Shows or hides the button.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the button at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the button using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Padding top	Sets the distance between the button label text and the top of the button.	Floating point number	Properties; Advanced: **PaddingTop**
Padding bottom	Sets the distance between the button label text and the bottom of the button.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the button label text and the left edge of the button.	Floating point number	Properties; Advanced: **PaddingLeft**
Padding right	Sets the distance between the button label text and the right edge of the button.	Floating point number	Properties; Advanced: **PaddingRight**
Font	Sets the name of the family of fonts used for the button label text.	Dropdown list	Properties; Advanced: **Font**
Font size	Sets the size of the button label text.	Floating point number	Properties; Advanced: **FontSize**
Font weight	Sets the weight of the button label text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	Dropdown list	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the label text in the button, either *Center*, *Justify*, *Left*, or *Right*.	Not applicable	Properties; Advanced: **TextAlignment**

Property	Description	Type	Location
Vertical alignment	Sets the vertical alignment of the label text in the button, either *Bottom*, *Middle*, or *Top*.	Dropdown list	Properties; Advanced: VerticalAlign
Font style	Sets the style of the button label text, either *Italic*, *Underline*, *Strikethrough*, or none.	Not applicable	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Border radius	Determines the corner radius of the button border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the button label text and the button background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the button border.	Not applicable	Properties; Advanced: BorderStyle, BorderThickness, BorderFillColor
Disabled	Turns off the button but leaves it visible.	Boolean	Properties; Advanced: **Disabled**
Disabled color	Sets the colors of the button label text, the button background, and the button border if DisplayMode is **Disabled**.	Not applicable	Properties; Advanced: **DisabledContentColor**, **DisabledFillColor**, **DisabledBorderColor**
Pressed color	Sets the colors of the button label text, the button background, and the button border when the user selects the button.	Not applicable	Properties; Advanced: **PressedContentColor**, **PressedFillColor**, **PressedBorderColor**
Hover color	Sets the colors of the button label text, the button background, and the button border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverContentColor**, **HoverFillColor**, **HoverBorderColor**
OnMixedRealitySelect	Behavior that is triggered when user selects the button to launch into the mixed reality experience.	Defined action	Advanced

Property	Description	Type	Location
OnChange	Behavior that is triggered when any property on the button is changed. This is a common property across multiple controls.	Defined action	Advanced
Tooltip	Determines the text to display when the user hovers over the button.	String	Advanced
ContentLanguage	Determines the display language of the button label, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the button allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced
TabIndex	Specifies the order the button is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **TabIndex**

Output properties

Your app can make use of more properties when a user interacts with the **View shape in MR** control. These are known as output properties. You can use these properties in other controls or to customize the app experience.

Property	Description	Type
Photos	Collects the photos captured during the mixed reality session. You can upload the mixed-reality photos to OneDrive and show them in a gallery.	Not applicable

Other mixed reality controls

- View 3D content with the **3D object** control.
- View images and 3D content in the real world with the **View in mixed reality** control.
- Measure distance, area, and volume with the **Measure in mixed reality** control.
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

- Create an app to view a shape in mixed reality
- Use marker detection with mixed reality controls

View 3D content or images in mixed reality

Article • 07/15/2022

Use the **View in MR** control to place a 3D model of an object or an image in a real-world space. For instance, preview a painting on a wall before you hang it, or see whether a 3D virtual bookcase will fit where you intend it to go.

What if you don't have a 3D model of the object? Load an image of the object and set its dimensions to the object's size. The control applies the image as the face of a virtual cube of the desired size. Place this stand-in for the 3D model in your space.

The **View in MR** control adds a button to your app. When a user selects the button, the app overlays a selected 3D model (in .glb, .stl, or .obj file formats) or image (in .jpg or .png file formats) on the device camera feed.



You can also take photos using the **View in MR** control and upload them to OneDrive.

> **Important**
>
> Your 3D content must be in the .glb, .stl, or .obj file formats. You can convert your existing 3D models into the .glb file format from a variety of 3D formats. 3D content compressed with Draco compression Draco is not supported.

> **Tip**
>
> - Make sure your 3D models are optimized for use with Power Apps to minimize load times.
> - The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

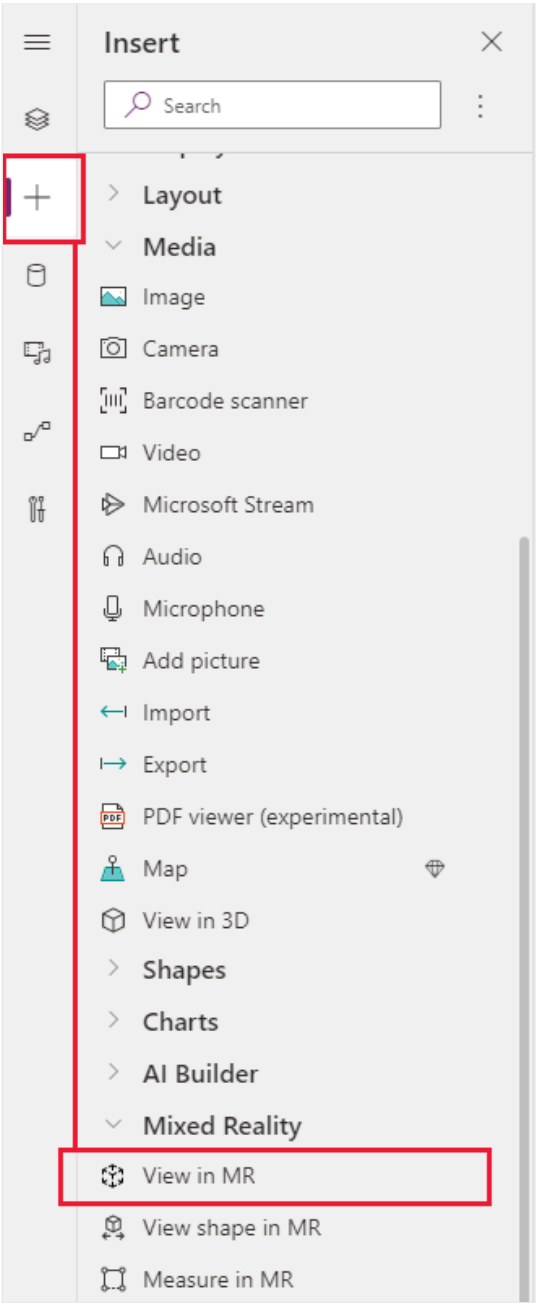
- The MR controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the Babylon sandbox should work in Power Apps through this shared MR platform. If your content works in Babylon but not in Power Apps, ask a question in the Power Apps Community Forum. (Tag it with "mixed reality.")

Add a View in MR button to an app screen

With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Mixed Reality**.
2. Select **View in MR** to place the control in the app screen, or drag the control to the screen to position it more precisely.

The control is a button that's labeled with an icon of a cube and the text *View in MR*. Change the label and hide the icon if you like in the **Text** and **Display type** properties.



View different models in a single session

The View in MR control displays one media file by default. To view different media files selected through a media gallery during the MR experience, use the control's *Items* advanced property. This property creates a table that lists media files the user can choose from in the mixed reality experience.

When you use the *Items* property, you need to set a value for the following additional advanced properties to map the columns of your table:

Expand table

Property	Description
ItemsSource (required)	Specifies which column in *Items* identifies the 3D model or image to show in the mixed reality view. See Connect 3D models to Power Apps to learn more about bringing models into Power Apps.
ItemsLabel	Specifies which column in *Items* contains the label to show in the media gallery. The *Alternative text* property is used for items that don't have a label specified.
ItemsThumbnail	Specifies which column in *Items* identifies the thumbnail to show in the media gallery. A default thumbnail is provided for items that don't have one specified.
ItemsObjectWidth, ItemsObjectHeight, ItemsObjectDepth	Specifies an explicit size for each media item in the table. *Object width*, *Object height*, and *Object depth* are used for items that don't have a specific size.

To test it, let's create a media gallery that shows three different plants in a room setting. Our *Items* table looks like this:

Expand table

label	source	thumbnail
Birds of Paradise	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/bird_of_paradise_blue.glb	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/bird_of_paradise_blue.png
Fiddle Leaf Fig	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/flf_orange.glb	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/flf_orange.png
Small Palm Tree	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/palm_green.glb	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/palm_green.png



With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Mixed Reality**.
2. Select **View in MR** to place the control in the app screen, or drag the control to the screen to position it more precisely.
3. On the **Advanced** properties tab, select the *Items* property and type or paste the following code:

```
Power Fx

Table(
{
label: "Birds of Paradise",
source: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/bird_of_paradise_blue.glb",
thumbnail: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/bird_of_paradise_blue.png"
},
{
label: "Fiddle Leaf Fig",
source: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
```

```
assets/master/flf_orange.glb",
thumbnail: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/flf_orange.png"
},
{
label: "Small Palm Tree",
source: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/palm_green.glb",
thumbnail: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/palm_green.png"
}
)
```

4. Set the *ItemsLabel* property to `"label"`.

5. Set the *ItemsSource* property to `"source"`.

6. Set the *ItemsThumbnail* property to `"thumbnail"`.



> **Note**
>
> The *Items* property is set to your data source which can be a collection, from a connector, or a hardcoded table like in this example. The *ItemsSource*, *ItemsLabel*, and *ItemsThumbnail* properties are set to the column names of that data source within quotation marks.

7. Save and publish the app, then open it on your mobile device to try it out.

How objects are scaled if you resize them

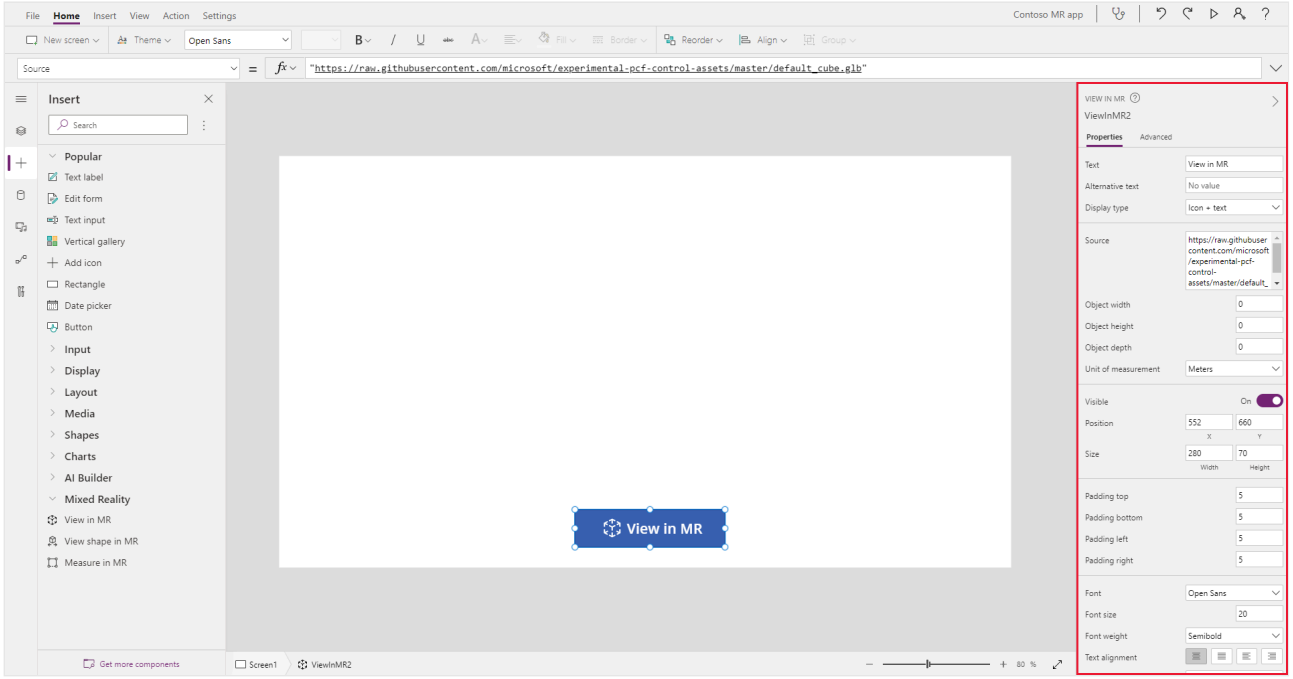
If you don't specify dimensions when you place a model, the **View in MR** control follows the dimensions given in the object file. You can resize the model by entering a value other than 0 in one or more of the control's width, height, and depth properties (**Object width**, **Object height**, and **Object depth**). How the model is scaled depends on how many dimensions you change.

- If all dimensions are *0* (the default), the model's size is left as it's set in the object file.
- If one dimension is set, the model is scaled equally based on the changed dimension, similar to setting "keep aspect ratio" when you resize an image.
- If two dimensions are set, the model scales to the two given dimensions and an average of the first two. For example, say you have a model that is 12 m high, 6 m wide, and 3 m deep. You set the **View in MR** control's properties as follows: **Object height**: **24**, **Object width**: **9**, **Object depth**: **0** (unchanged). The height increases by a factor of 2, and the width increases by a factor of 1.5. The increase in height and width is averaged to find the factor by which the depth should be increased: 2 + 1.5 = 3.5 / 2 = 1.75. The final dimensions of the model are **Object height** 24, **Object width** 9, and **Object depth** 5.25 (3 x 1.75).
- If all three dimensions are set, the model's size is changed to what you specified.

> **Note**
>
> If the measurements don't adhere to the original aspect ratio, the model could be squashed or distorted when it's viewed in MR.

Properties

Change the **View in MR** button's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Expand table

Property	Description	Type	Location
Text	Sets the button label text.	String	Properties; Advanced: **Text**

Property	Description	Type	Location
Alternative text	Specifies the text to be displayed if the model can't load or if the user hovers over the model.	String	Properties; Advanced: **AltText**
Display type	Determines whether the button label shows an icon of a cube, text, or both.	Drop-down selection	Properties; Advanced: **DisplayType**
Source	Identifies the data source (.glb, .stl, or .obj file) to display.	Not applicable	Properties; Advanced: **Source**
Sources List (Items)	Data source (table) that lists multiple media files to view in a single mixed reality session.	Table	Properties; Advanced: **Items**
Object width	Sets the width of the model.	Integer	Properties; Advanced: **ObjectWidth**
Object height	Sets the height of the model.	Integer	Properties; Advanced: **ObjectHeight**
Object depth	Sets the depth of the model.	Integer	Properties; Advanced: **ObjectDepth**
Unit of measurement	Sets the measurement unit used for the model's width, height, and depth.	List	Properties; Advanced: **Units**
Enable shadows	Determines whether shadows are used to enhance the 3D effect when the model is shown.	Boolean	Properties; Advanced: **EnableShadows**
Enable marker detection	Determines whether marker detection will be used to determine model position and rotation. See Use marker detection with mixed reality controls	Boolean	Properties; Advanced: **EnableMarkerDetection**
Visible	Shows or hides the button.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the button at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the button using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Padding top	Sets the distance between the button label text and the top of the button.	Floating point number	Properties; Advanced: **PaddingTop**
Padding bottom	Sets the distance between the button label text and the bottom of the button.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the button label text and the left edge of the button.	Floating point number	Properties; Advanced: **PaddingLeft**
Padding right	Sets the distance between the button label text and the right edge of the button.	Floating point number	Properties; Advanced: **PaddingRight**
Font	Sets the name of the family of fonts used for the button label text.	List	Properties; Advanced: **Font**
Font size	Sets the size of the button label text.	Floating point	Properties; Advanced: **FontSize**

Property	Description	Type	Location
		number	
Font weight	Sets the weight of the button label text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	List	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the label text in the button, either *Center*, *Justify*, *Left*, or *Right*.	Not applicable	Properties; Advanced: **TextAlignment**
Vertical alignment	Sets the vertical alignment of the label text in the button, either *Bottom*, *Middle*, or *Top*.	List	Properties; Advanced: **VerticalAlign**
Font style	Sets the style of the button label text, either *Italic*, *Underline*, *Strikethrough*, or none.	Not applicable	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Border radius	Determines the corner radius of the button border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the button label text and the button background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the button border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderFillColor**
Disabled	Turns off the button but leaves it visible.	Boolean	Properties; Advanced: **Disabled**
Disabled color	Sets the colors of the button label text, the button background, and the button border if DisplayMode is **Disabled**.	Not applicable	Properties; Advanced: **DisabledContentColor**, **DisabledFillColor**, **DisabledBorderColor**
Pressed color	Sets the colors of the button label text, the button background, and the button border when the user selects the button.	Not applicable	Properties; Advanced: **PressedContentColor**, **PressedFillColor**, **PressedBorderColor**
Hover color	Sets the colors of the button label text, the button background, and the button border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverContentColor**, **HoverFillColor**, **HoverBorderColor**
OnMixedRealitySelect	Behavior that is triggered when user selects the button to launch into the mixed reality experience.	Defined action	Advanced
OnChange	Behavior that is triggered when any property on the button is changed.	Defined action	Advanced
Tooltip	Determines the text to display when the user hovers over the button.	String	Advanced
ContentLanguage	Determines the display language of the button label, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the button allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced
TabIndex	Specifies the order the button is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **TabIndex**

Output properties

Your app can make use of more properties when a user interacts with the **View in MR** control. These are known as output properties. You can use output properties in other controls or to customize the app experience.

Expand table

Property	Description	Type
Photos	Collects the photos captured during the mixed reality session. You can upload the mixed-reality photos to OneDrive and show them in a gallery.	Not applicable

Other mixed reality controls

- View 3D content with the **3D object** control.
- Measure distance, area, and volume with the **Measure in mixed reality** control.
- Create and view predefined 3D shapes with the **View shape in mixed reality** control
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

- Create an app with 3D and mixed reality controls
- Use marker detection with mixed reality controls

Feedback

Was this page helpful? Yes No

Provide product feedback

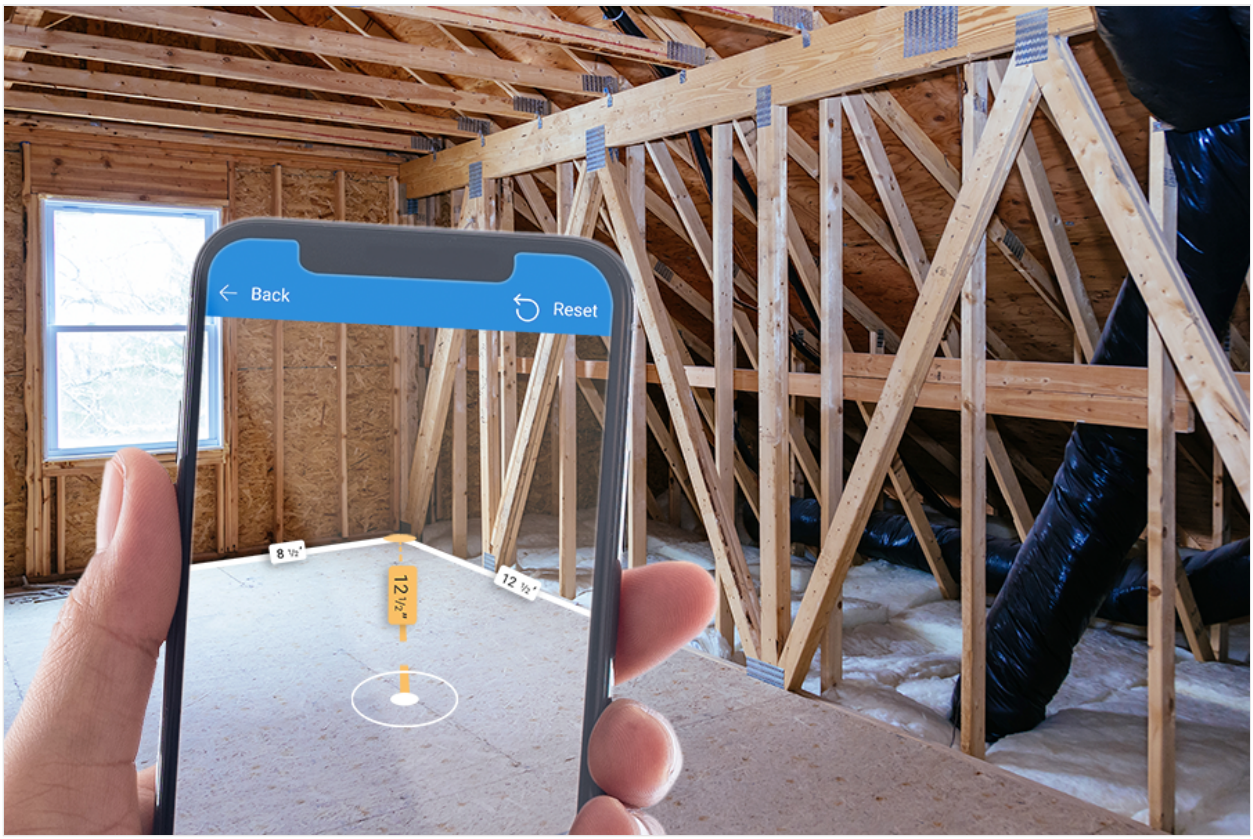
Take measurements in mixed reality

Article • 05/09/2022

Use the **Measuring camera** control in a canvas app to measure distance, area, and volume in the real world. You can also create two- and three-dimensional freeform shapes and see how they fit in a real-world space.

The **Measuring camera** control adds a button to your app. When a user selects the button, the app shows a live feed from the device camera. On-screen instructions direct the user to establish tracking by slowly panning the device from right to left across the surface to be measured. The user identifies a starting point and adds segments as needed until the measurement is complete. Dimensions of the measured space are overlaid on the camera feed.

The user can submit the measurements for capture in the control's **Measurements** output properties. The user can also take screenshots while a space is measured. The screenshots are captured in the control's **Photos** output property.



Tip

The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

Add a Measuring camera button to an app screen

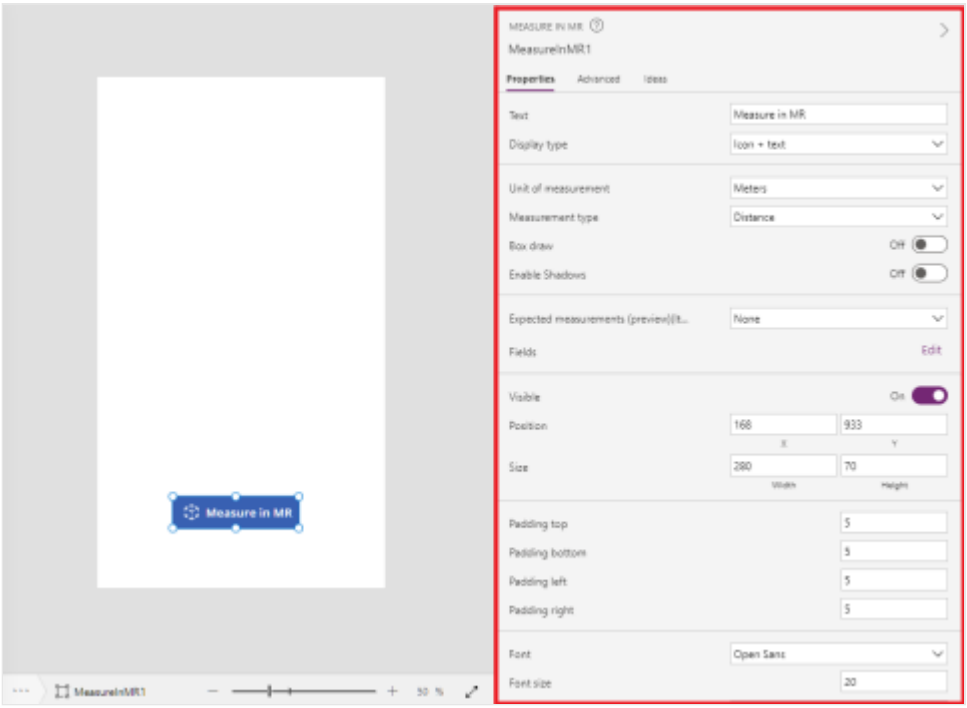
With your app open for editing in Power Apps Studio :

1. Open the **Insert** tab and expand **Media**.
2. Select **Measuring camera** to place the control in the app screen, or drag the control to the screen to position it more precisely.

The control is a button that's labeled with an icon of a cube and the text *Measure*. Change the label and hide the icon if you like in the **Text** and **Display type** properties.

Properties

Change the **Measuring camera** button's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Property	Description	Type	Location
Text	Sets the button label text.	String	Properties; Advanced: **Text**
Display type	Determines whether the button label shows an icon of a cube, text, or both.	Dropdown list	Properties; Advanced: **DisplayType**
Unit of measurement	Sets the unit used for the dimensions.	Dropdown list	Properties; Advanced: **Units**

Property	Description	Type	Location
Measurement type	Determines the kind of measurement the user can make, either *Distance*, *Area*, or *Volume* (area plus height or depth).	Dropdown list	Properties; Advanced: **MeasurementType**
Box draw	Locks captured measurements to rectangular shapes.	Boolean	Properties; Advanced: **BoxDraw**
Enable Shadows	Determines whether shadows are used to enhance the 3D effect when a shape is drawn.	Boolean	Properties; Advanced: **EnableShadows**
Expected measurements (Items)	Identifies a data source (**Items**) in the form of a table in an Excel workbook from which to get the measurements the user should capture.	Not applicable	Properties; Advanced: **Items**
Visible	Shows or hides the button.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the button at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the button using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Padding top	Sets the distance between the button label text and the top of the button.	Floating point number	Properties; Advanced: **PaddingTop**
Padding bottom	Sets the distance between the button label text and the bottom of the button.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the button label text and the left edge of the button.	Floating point number	Properties; Advanced: **PaddingLeft**

Property	Description	Type	Location
Padding right	Sets the distance between the button label text and the right edge of the button.	Floating point number	Properties; Advanced: **PaddingRight**
Font	Sets the name of the family of fonts used for the button label text.	Dropdown list	Properties; Advanced: **Font**
Font size	Sets the size of the button label text.	Floating point number	Properties; Advanced: **FontSize**
Font weight	Sets the weight of the button label text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	Dropdown list	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the label text in the button, either *Center*, *Justify*, *Left*, or *Right*.	Not applicable	Properties; Advanced: **TextAlignment**
Vertical alignment	Sets the vertical alignment of the label text in the button, either *Bottom*, *Middle*, or *Top*.	Dropdown list	Properties; Advanced: **VerticalAlign**
Font style	Sets the style of the button label text, either *Italic*, *Underline*, *Strikethrough*, or none.	Not applicable	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Border radius	Determines the corner radius of the button border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the button label text and the button background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the button border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderFillColor**
Disabled	Turns off the button but leaves it visible.	Boolean	Properties; Advanced: **Disabled**
Disabled color	Sets the colors of the button label text, the button background, and the button border if **DisplayMode** is **Disabled**.	Not applicable	Properties; Advanced: **DisabledContentColor**, **DisabledFillColor**, **DisabledBorderColor**

Property	Description	Type	Location
Pressed color	Sets the colors of the button label text, the button background, and the button border when the user selects the button.	Not applicable	Properties; Advanced: **PressedContentColor**, **PressedFillColor**, **PressedBorderColor**
Hover color	Sets the colors of the button label text, the button background, and the button border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverContentColor**, **HoverFillColor**, **HoverBorderColor**
OnMixedRealitySelect	Contains code that runs when the user exits the measurement screen with new results.	Event	Advanced
OnChange	Contains code that runs when a button property is changed.	Event	Advanced
OnSelect	Contains code that runs when the user selects the button.	Event	Advanced
ItemsLabels	Identifies the column in **Items** that contains the labels for the measurements users should capture.	ColumnName	Advanced
ItemsMeasurementType	(Optional) Identifies the column in **Items** that contains a string (*Distance*, *Area*, *Volume*, or *Freeform*) that specifies the type of measurement users should capture.	ColumnName	Advanced; see **MeasurementType**
ItemsBoxDraw	(Optional) Identifies the column in **Items** that contains a string (*True* or *False*) that locks measurements to rectangular shapes.	ColumnName	Advanced; see **BoxDraw**
Tooltip	Determines the text to display when the user hovers over a visual.	String	Advanced

Property	Description	Type	Location
ContentLanguage	Determines the display language of the control, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced
TabIndex	Specifies the order the control is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **Tab index**

Output properties

Your app can make use of more properties when a user interacts with the **View in MR** control. These are known as output properties. You can use these properties in other controls or to customize the app experience.

Property	Description
Photos	Captures, in a collection, information about photos the user takes while a space is measured. Use the collection to upload mixed-reality photos to OneDrive and show them in a gallery.
Measurements	Captures, in a table, the measured distances, volumes, and areas.
Segments	Captures, in a table, information about segments in a measured object.

The Measurements table has the following columns:

- **Label:** A string that identifies the measurement
- **Id:** A number that uniquely identifies the measurement
- **Units:** A string that identifies the unit of the measurement
- **Height:** A number that represents the height of the measured volume (0 if a 2D area was measured)
- **Length:** A number that represents the total length of the measured perimeter or path
- **BoundingWidth:** A number that represents the minimum width that bounds the shape

- **BoundingDepth:** A number that represents the minimum depth that bounds the shape
- **Area:** A number that represents the estimated area of the measured shape in units squared
- **Volume:** A number that represents the estimated volume of the measured shape in units cubed

The Segments table has the following columns:

- **Length:** A number that represents the total distance of a segment in the specified units
- **DirectionX:** A number that represents the X direction of the segment in world (3D) space
- **DirectionY:** A number that represents the Y direction of the segment in world(3D) space (typically 0)
- **DirectionZ:** A number that represents the Z direction of the segment in world (3D) space

Learn more about how to use output properties of the **Measuring camera** control in your apps: Validate measurements in mixed reality using a spatial test filter

Other mixed reality controls

- View 3D content with the **3D object** control.
- View images and 3D content in the real world with the **View in mixed reality** control.
- Create and view predefined 3D shapes with the **View shape in mixed reality** control
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

Create an app using the Measuring Camera control

Test whether an object will fit in a space using mixed reality

Article • 12/16/2022

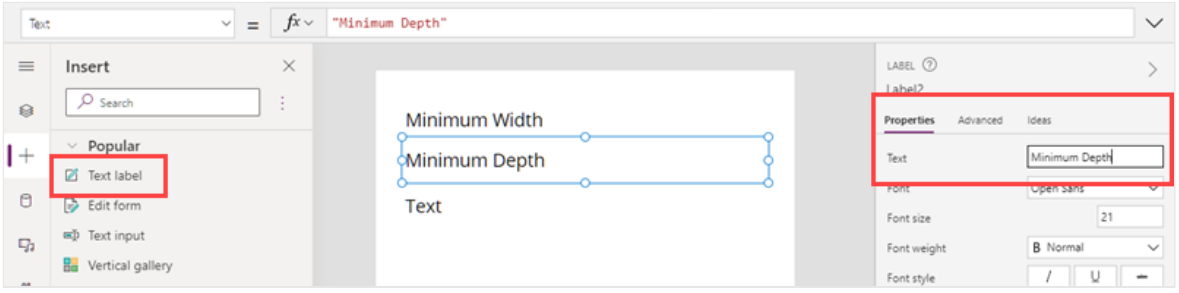
You can use the Measuring Camera control to determine whether an object of a known size will fit in a particular space. In this article, we'll create a canvas app to do just that.

> **Tip**
>
> The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

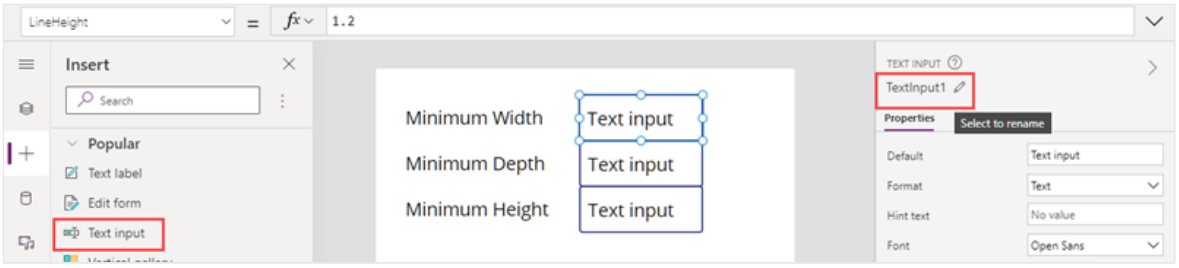
Add minimum dimensions input fields to a canvas app

First, we'll create a blank canvas app. Then we'll add a way for the user to enter the minimum dimensions to test against (that is, the smallest space that will hold the object we want to fit).

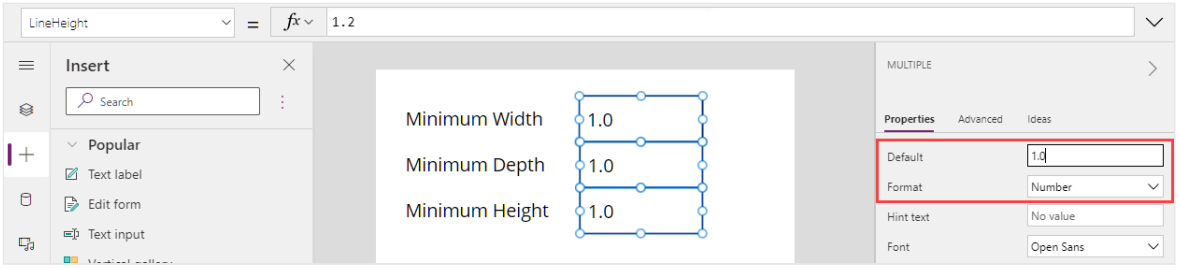
1. Create a blank canvas app.
2. Select the **Insert** tab. Insert and place three **Text label** controls on the screen.
3. Change the **Text** property of the labels to **Minimum Width**, **Minimum Depth**, and **Minimum Height**, respectively.



4. Select the **Insert** tab. Insert three **Text Input** controls, and place them to the right of the three labels.
5. Rename the text input controls to **minWidth**, **minDepth**, and **minHeight**, respectively.



6. Select all three text input controls. Change the group's **Format** property to **Number** and set the **Default** property to **1.0**.



Add and bind the Measuring camera control

Next, we'll add the **Measuring camera** control to allow the user to measure a three-dimensional space. We'll bind the measurements to a data table for later use.

1. Select the **Insert** tab, and expand **Media**.

2. Select **Measuring camera** to place the control in the app screen. Drag the control near the bottom of the screen.

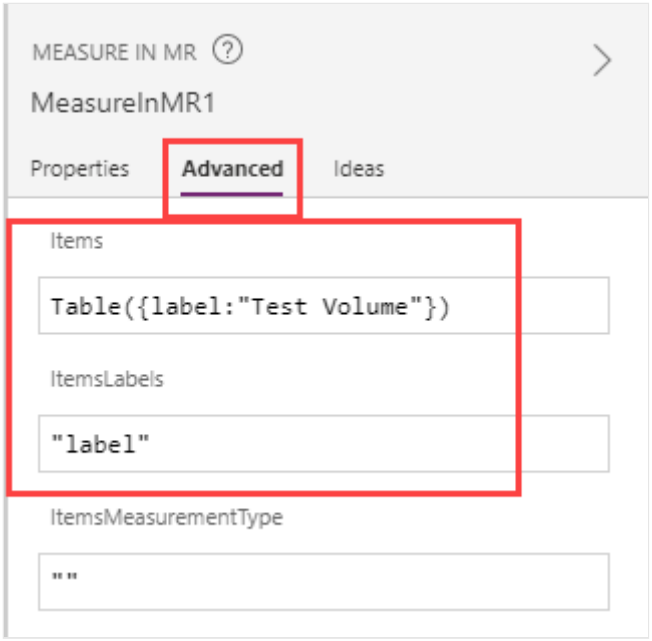
3. Change the control's properties as follows:

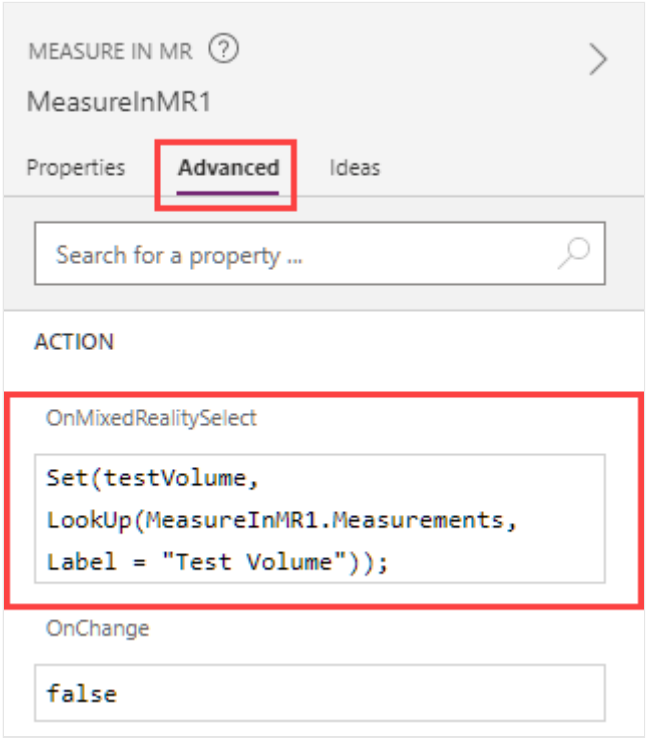
Expand table

Property name	Value	Where
Unit of measurement	Feet or Meters	Select on the **Properties** tab
Measurement type	Volume	Select on the **Properties** tab
Box draw	On	Select on the **Properties** tab
Items	**Table({label:"Test Volume"})**	Enter on the **Advanced** tab
ItemsLabels	**"label"**	Enter on the **Advanced** tab

Property name	Value	Where
OnMixedRealitySelect	**Set(testVolume, LookUp(MeasureInMR1.Measurements, Label = "Test Volume"));**	Enter on the **Advanced** tab

The **Items**, **ItemsLabels**, and **OnMixedRealitySelect** properties of the **Measuring camera** control should look like the following examples when you're done:





The formula in **Items** captures the dimensions of the measured space in a data table named "Test Volume." The formula in **OnMixedRealitySelect** sets a variable named "testVolume" to the values captured in the "Test Volume" table.

Display the results

Finally, we'll add text labels to display the measurements that were taken and the result of the test (that is, whether the object will fit in the measured space).

1. Select the **Insert** tab. Insert and place four **Text label** controls on the screen.

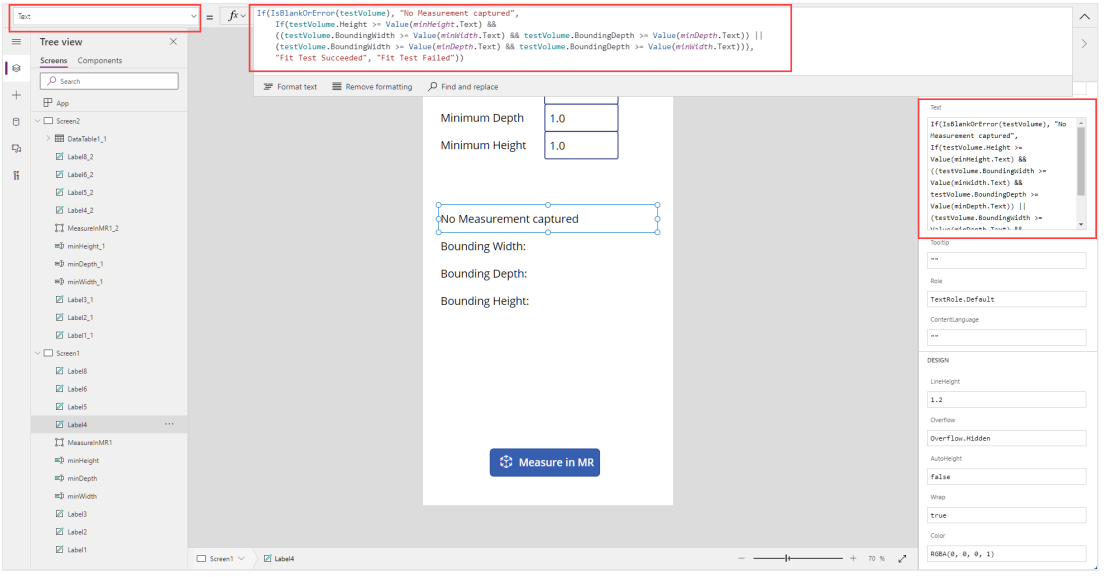
2. Change the **Text** property of the label controls as follows. (Hint: Copy each formula and paste it in the formula bar or on the **Advanced** properties tab, whichever you prefer.)

    - First label:

```
Power Fx
```

```
    If(IsBlankOrError(testVolume), "No Measurement captured",
        If(testVolume.Height >= Value(minHeight.Text) &&
        ((testVolume.BoundingWidth >= Value(minWidth.Text) &&
testVolume.BoundingDepth >= Value(minDepth.Text)) ||
        (testVolume.BoundingWidth >= Value(minDepth.Text) &&
testVolume.BoundingDepth >= Value(minWidth.Text))),
        "Fit Test Succeeded", "Fit Test Failed"))
```

      This formula sets the label text according to whether the test succeeded or failed, based on the minimum dimensions required to hold the object we want to fit. If the test didn't run, either because no measurements were captured or an error occurred, the label text reflects that.



    - Second label:

```
Power Fx
```

```
    Concatenate("Bounding Width: ",
Text(testVolume.BoundingWidth))
```

This formula sets the label text to "Bounding Width:" followed by the width dimension of the measured space.

- Third label:

```
Power Fx

    Concatenate("Bounding Depth: ",
Text(testVolume.BoundingDepth))
```

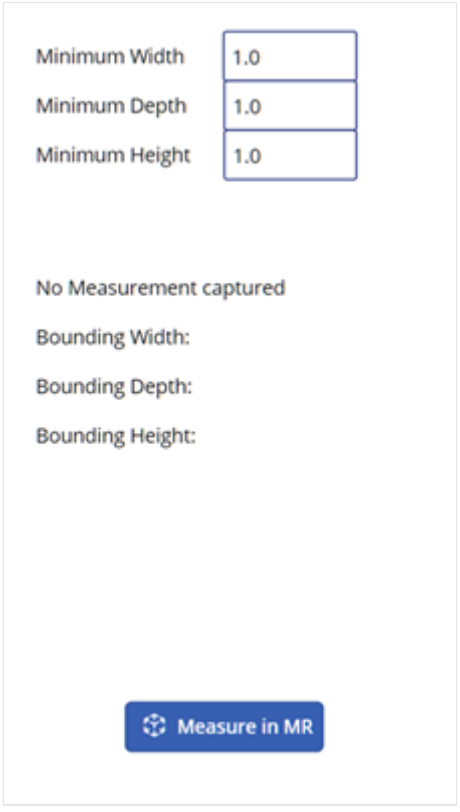
This formula sets the label text to "Bounding Depth: " followed by the depth dimension of the measured space.

- Fourth label:

```
Power Fx

    Concatenate("Bounding Height: ", Text(testVolume.Height))
```

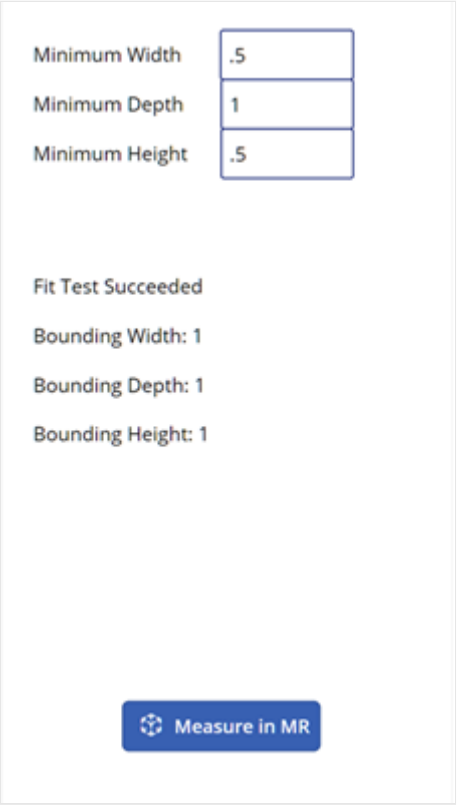
This formula sets the label text to "Bounding Height: " followed by the height dimension of the measured space.

The app screen should now look like this:



Test the app

Select the preview button to run the app in preview mode. Select **Measuring camera** to populate the labels with data. To verify that the label bindings are working, enter new values in the text input fields.



The values for **Bounding Width** and **Bounding Depth** can be swapped when you perform the test.

Save and publish your app. Open it on a mixed-reality–enabled device to test whether an object with the specified dimensions will fit within the bounds of any measurement captured.

Filtering a data source

The app we created tests a single set of dimensions that are specified by the user when the app runs. However, you can extend it to test against any data source by applying the formula as a **Filter** predicate.

For example, let's say that our app contains a reference to a Dataverse table named **Products** that includes three columns: **Width**, **Depth**, and **Height** (corresponding to each product's dimensions). To filter the collection to show only the measurements that would fit in a measured volume, we can apply the following formula.

```
Power Fx
```

```
    If(IsBlankOrError(testVolume), Products,
    Filter(Products, testVolume.Height >= Height &&
        ((testVolume.BoundingWidth >= Width && testVolume.BoundingDepth >=
Depth) ||
        (testVolume.BoundingWidth >= Depth && testVolume.BoundingDepth >=
Width))))
```

See also

- 3D object control
- Measuring Camera control
- View shape in MR control

Feedback

Was this page helpful? Yes No

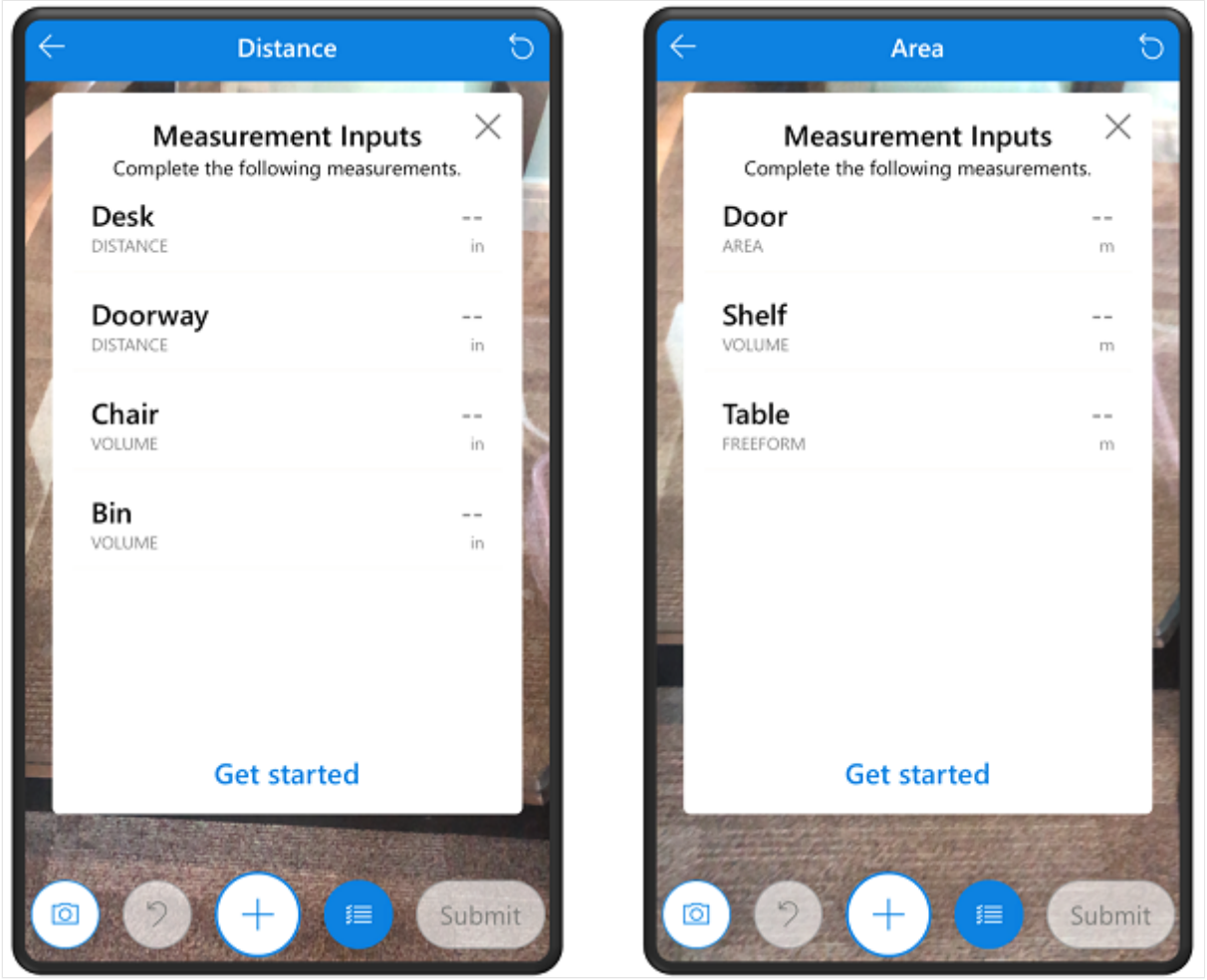
Provide product feedback

Use the Measure in MR control to take measurements

Article • 07/15/2022

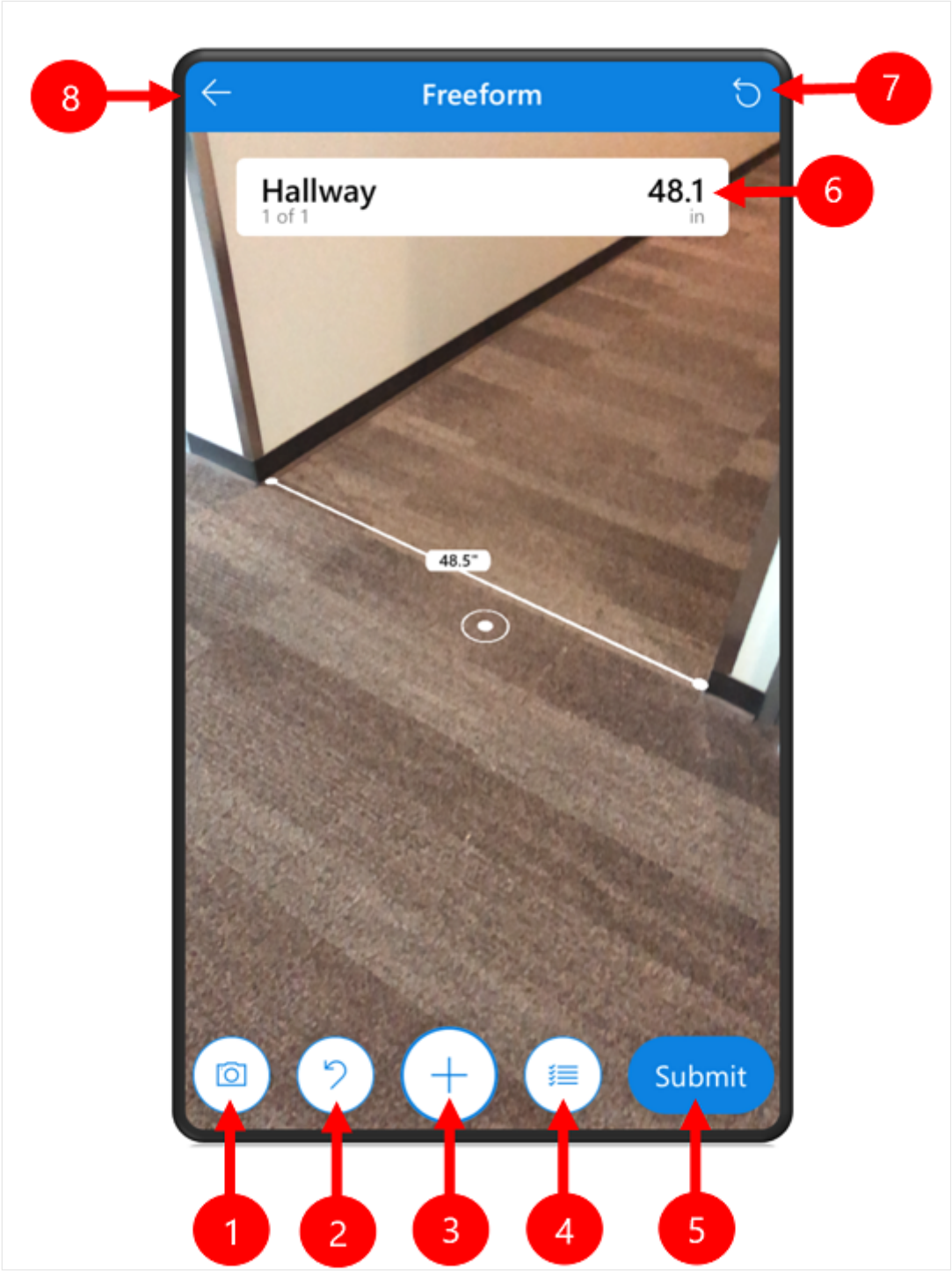
Measure the distance, area, and volume of your physical space using your device camera.

To use the measurement feature, your app maker needs to enable the Measure in MR control for your canvas app. Depending on how your app maker has configured the app, the type of measurements can vary as shown in the example below.



Basic navigation

It's easy to get around and take measurements of your space. The following illustration shows the primary navigation elements.



Legend

1. **Camera**: Take a picture of your current view. The picture can be used by your canvas app but isn't viewable when you're in a mixed reality experience.
2. **Undo**: Rewinds the last measurement action taken and returns to the state before the last add, reset, or delete.
3. **Add**: Select to start a new measurement or add a new measurement point. When you're done measuring the button changes to a checkmark button that you select to complete your measurement.
4. **Overview**: Opens a list with details of the items you have measured.
5. **Submit**: When you're done taking your measurement(s), use the Submit button to close the mixed reality experience and send the measurement(s) to your canvas

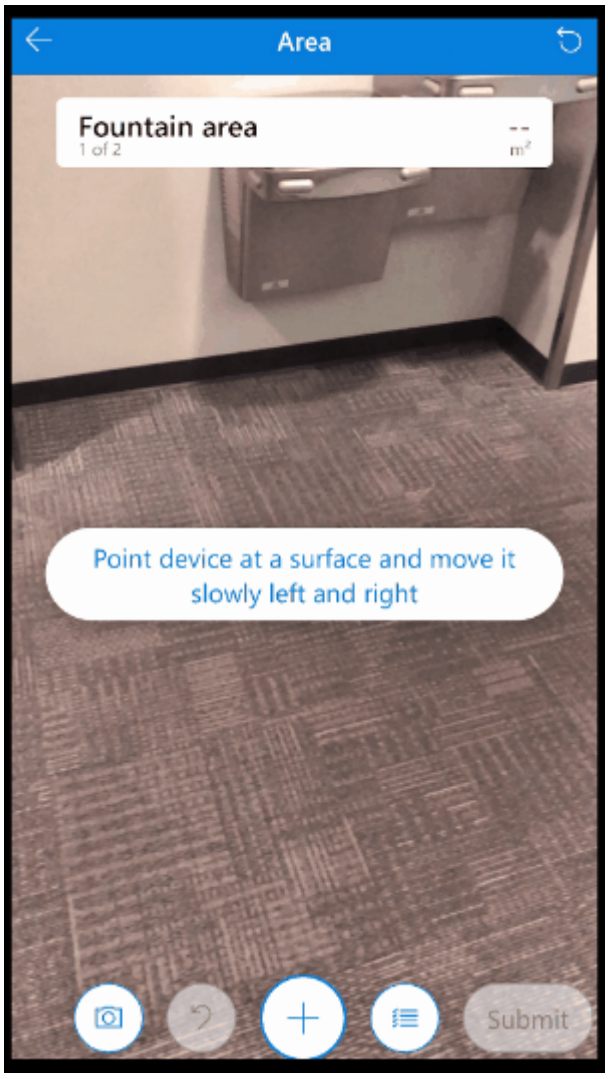
app. The button is grayed out if there are no measurements or when you're in the process of taking measurements.

6. **Overview**: Opens a list with details of the items you have measured.
7. **Reset**: Clear all your measurements at once.
8. **Exit**: Leave the mixed reality experience and return to your canvas app.

Take a measurement

1. Open your canvas app and select the **Measure in MR** button.

2. Follow the instructions on the screen by moving your device until it vibrates, and white dots appear. The white dot with a circle is the starting point of your measurement.

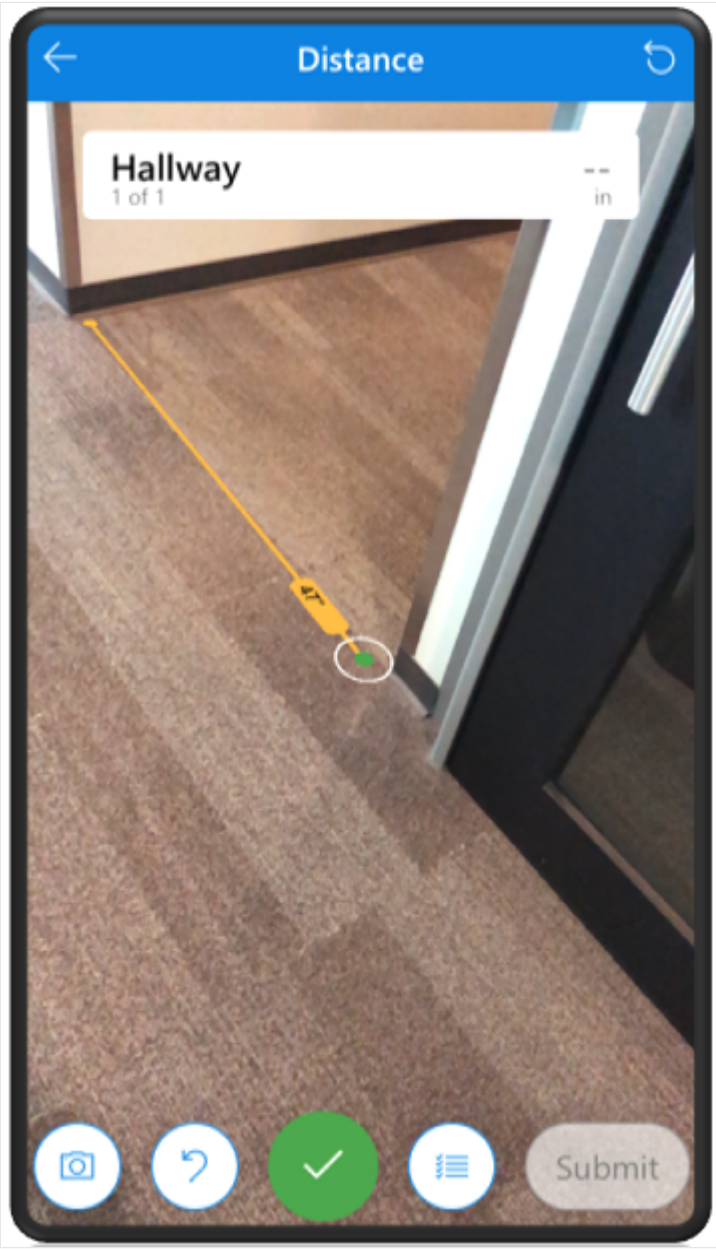


3. Select  to start a new measurement. Then follow these steps depending on the type of measurement you're taking:

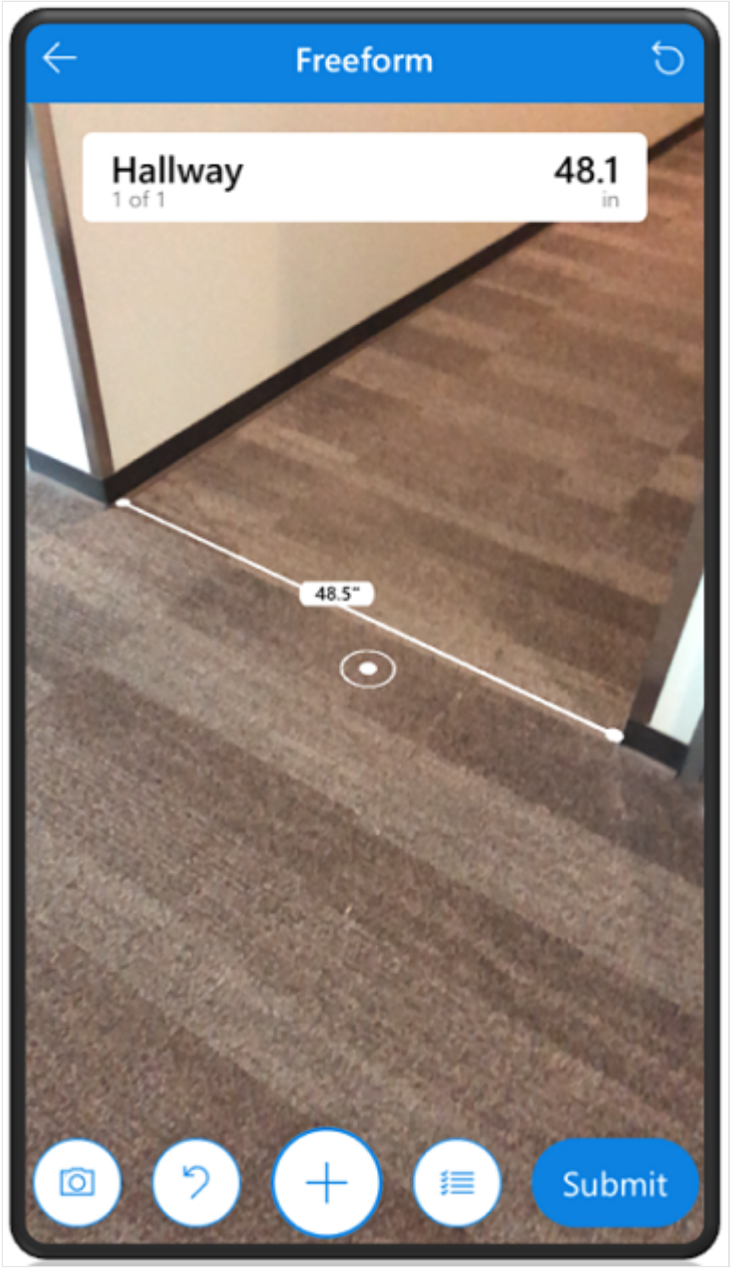
   - **Distance**

a. Select and then slowly move the dot to where you want the measurement to end.

b. Select the checkmark button to finish the measurement.

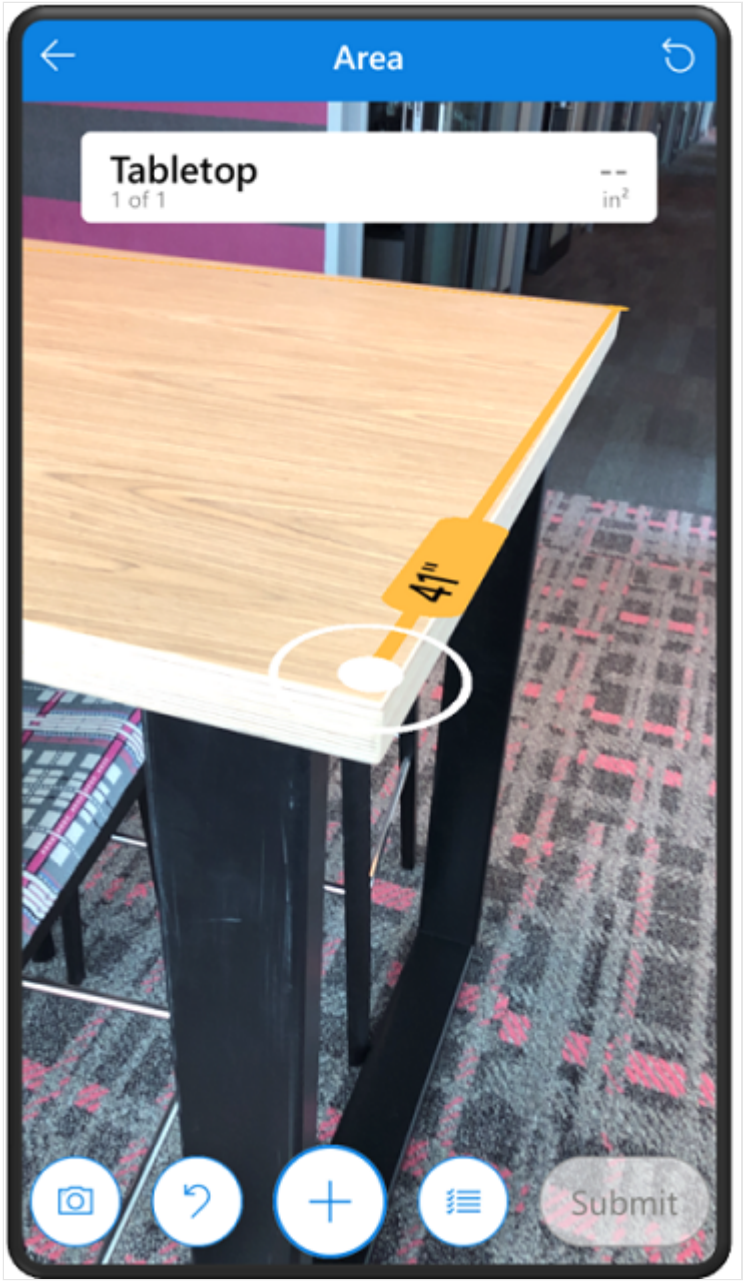


c. Continue taking measurements, or select **Submit** to save your final measurements and return to your canvas app.

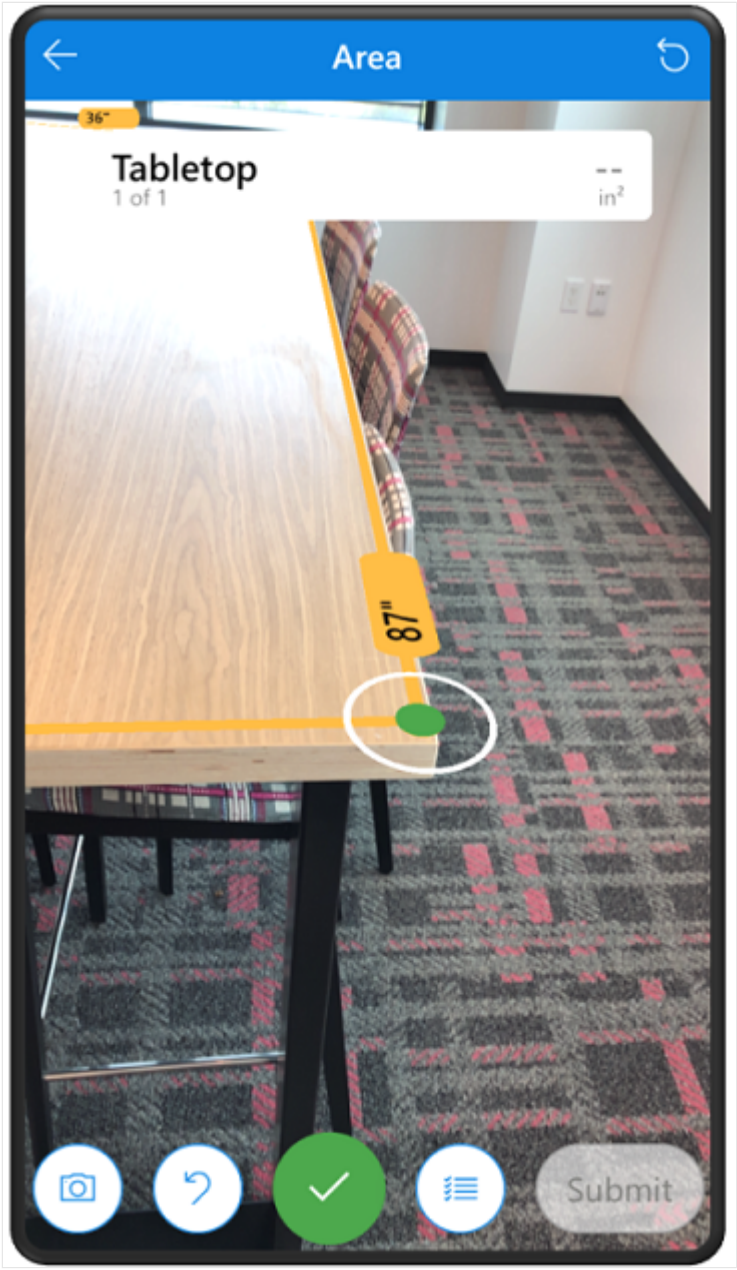


- **Area**

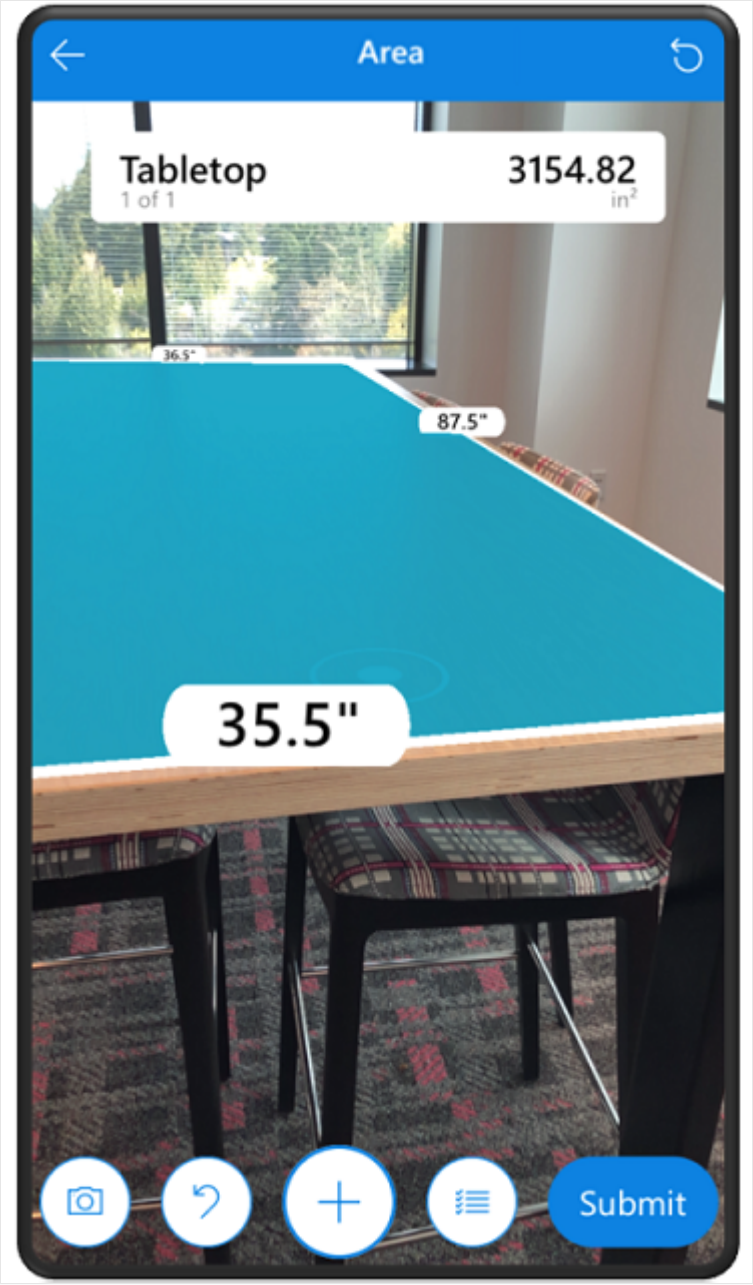
  a. Select  to add points and outline the area that you want to measure.



b. Connect the last dot to the first dot to complete the measurement. When you're done, select the checkmark button.

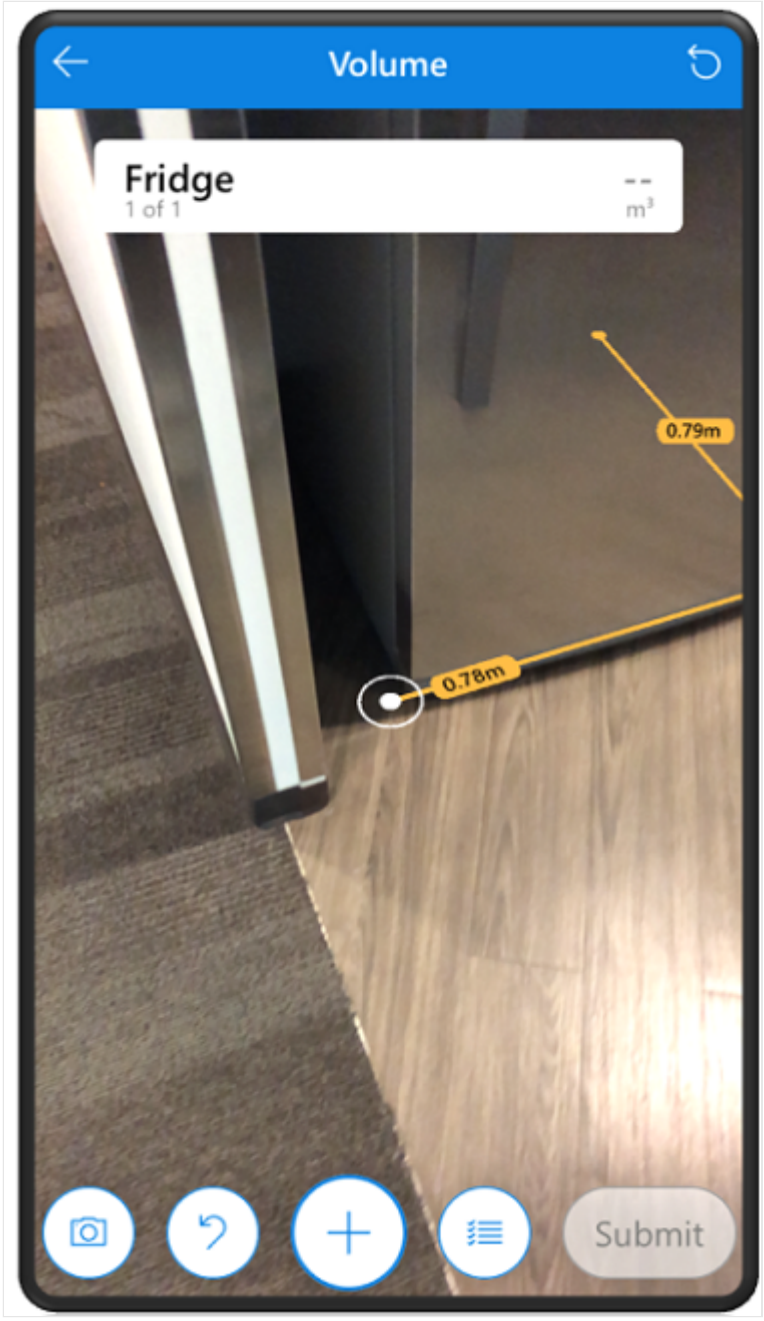


c. Continue taking measurements, or select **Submit** to save your final measurements and return to your canvas app.



- **Volume**

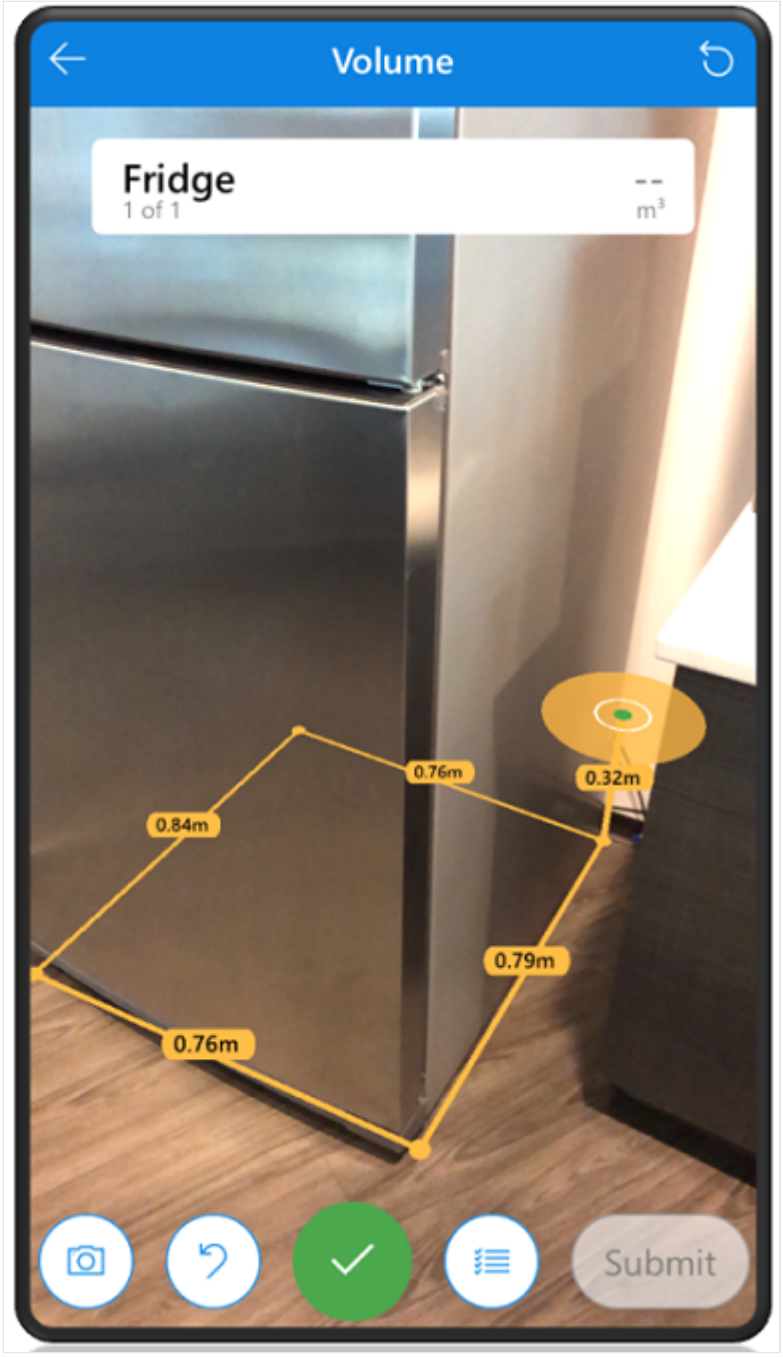
  a. Select  to add points and outline the base of the volume that you want to measure.



b. Connect the last dot to the first dot to complete the measurement.



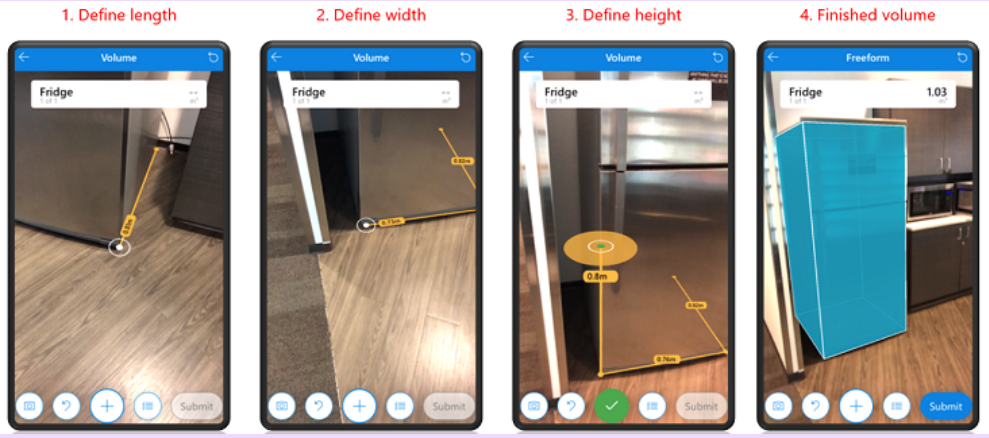
c. Move your device upwards and select the desired height and then select the checkmark button.



d. Continue taking measurements, or select **Submit** to save your final measurements and return to your canvas app.



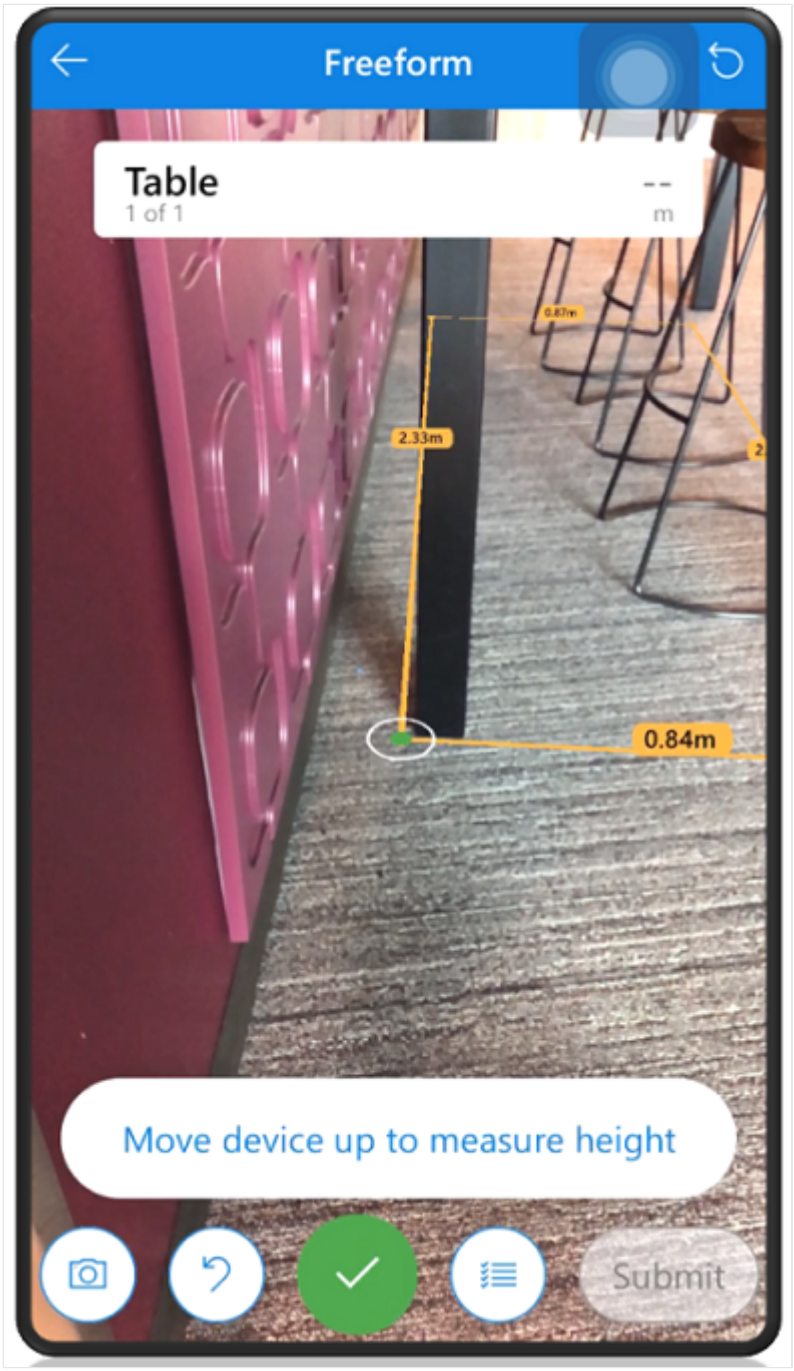
> **Note**
>
> If your app has **Box Draw** enabled then you can only measure volume by a rectangular prism. To take this measurement select four points; three to draw the base and one for the height.



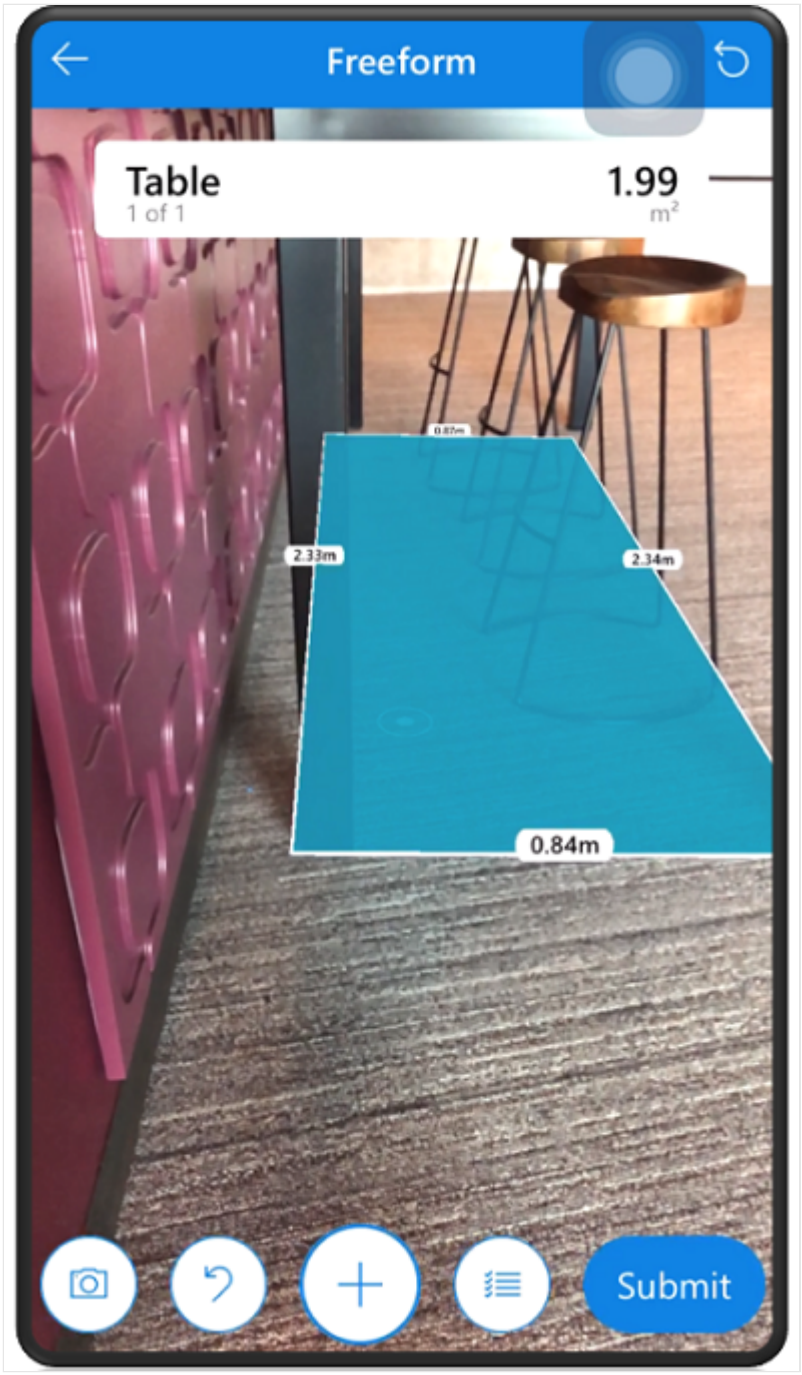
- **Freeform**

  a. Select  to measure distance, multi-segment, area, and volume all together.

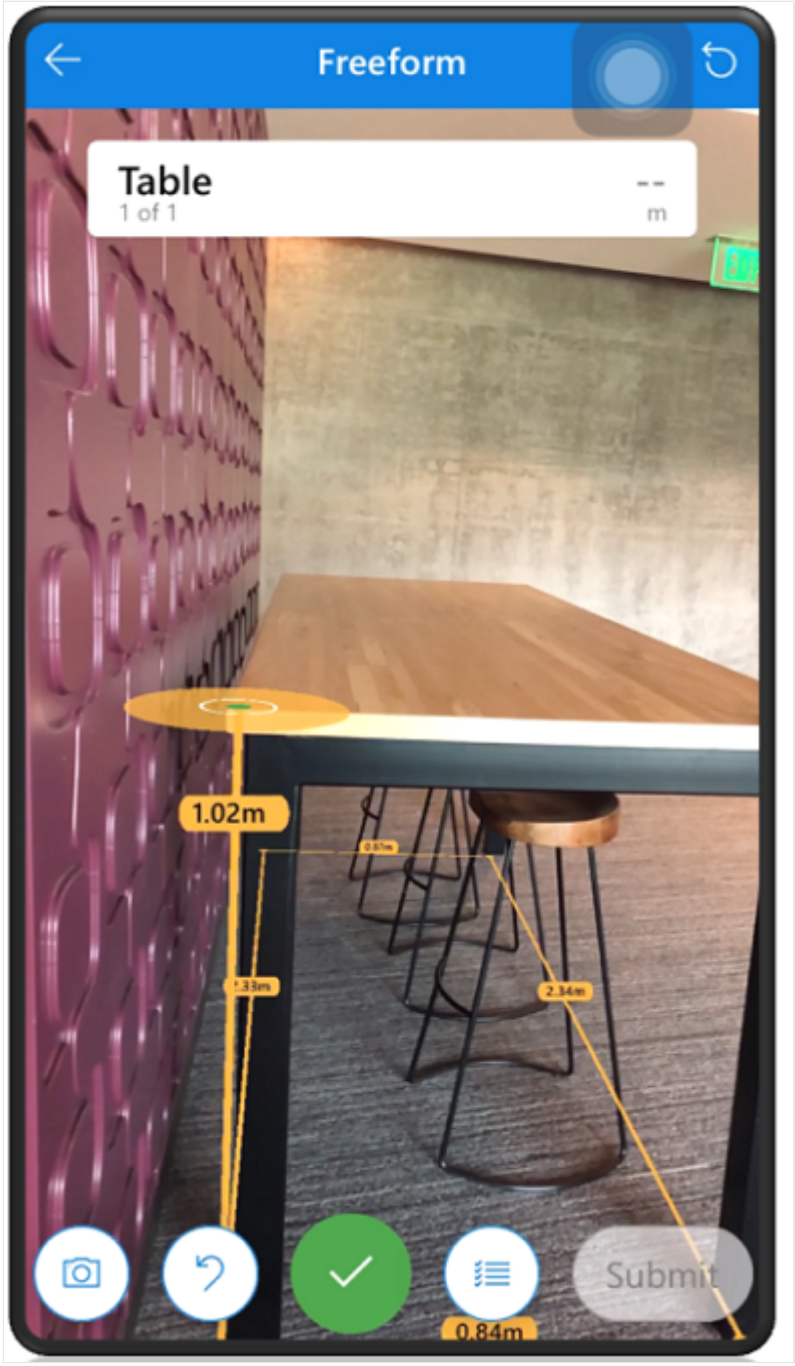
  b. Connect the last dot to the first dot to complete the measurement.



c. Keep the dot at the last point and tap the green checkmark to finish your current measurement. Continue taking other measurements, or if you are done, select **Submit** to return to your canvas app.



The same rectangle can be part of a volume measurement. Instead of selecting the last point you added, rotate your phone upwards to select a height and then tap the green checkmark. Continue taking other measurements, or if you are done, select **Submit** to return to your canvas app.

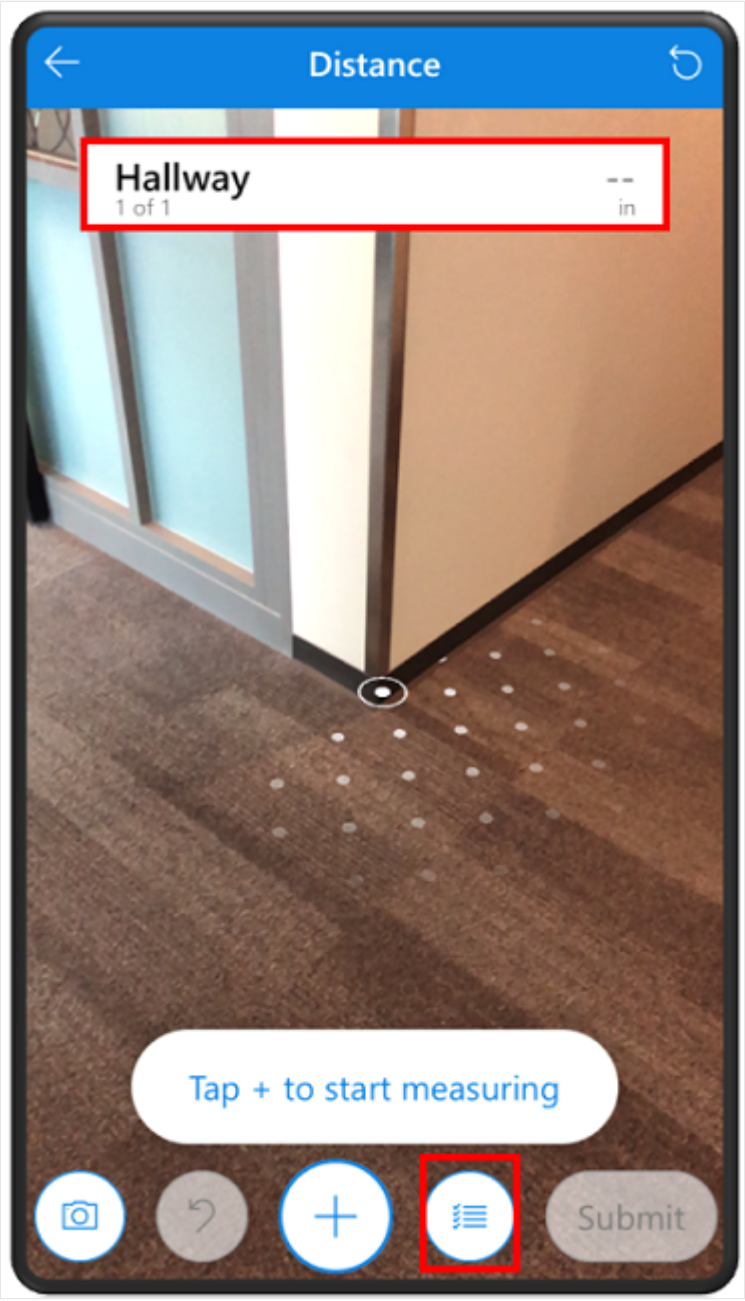


> ⓘ **Note**
>
> If your app has **Box Draw** enabled, then in freeform you can only measure in a rectangular area and volume.

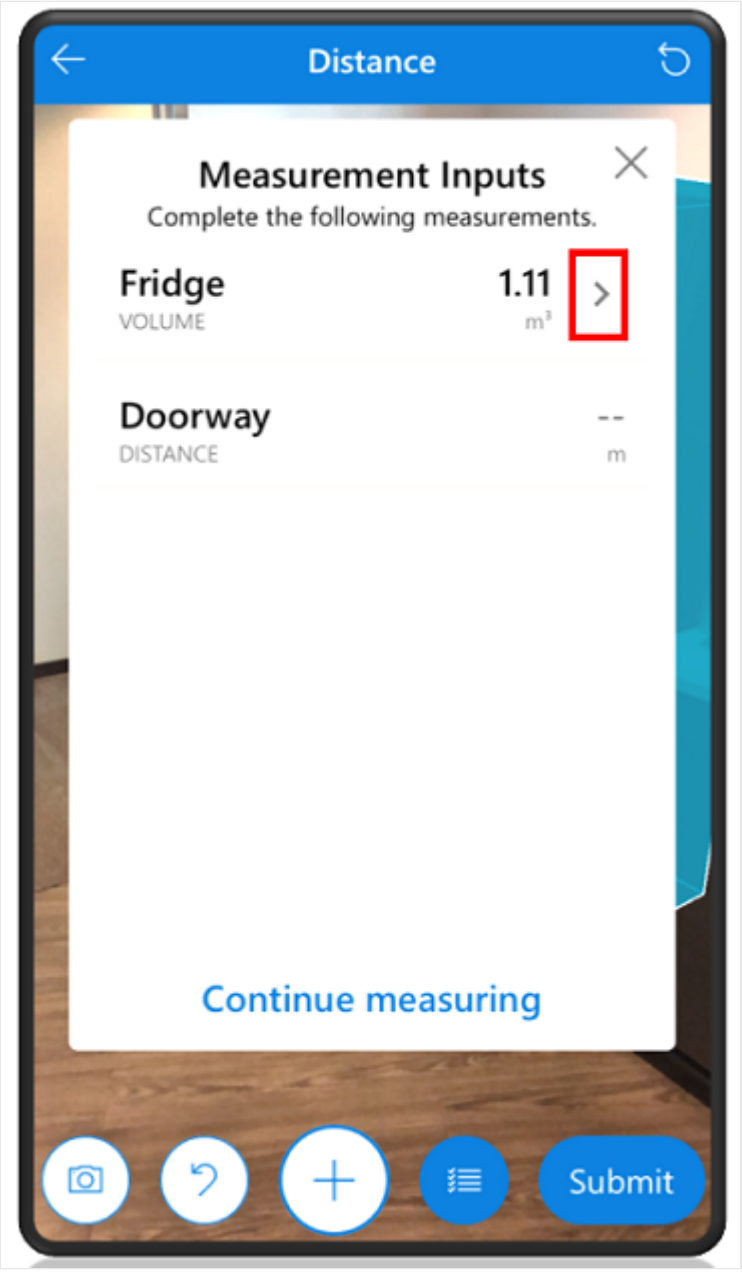
track of your measurements

When you take measurements, you can always go back and review your measurements.

1. At the top select the measurement overview or select the overview button.

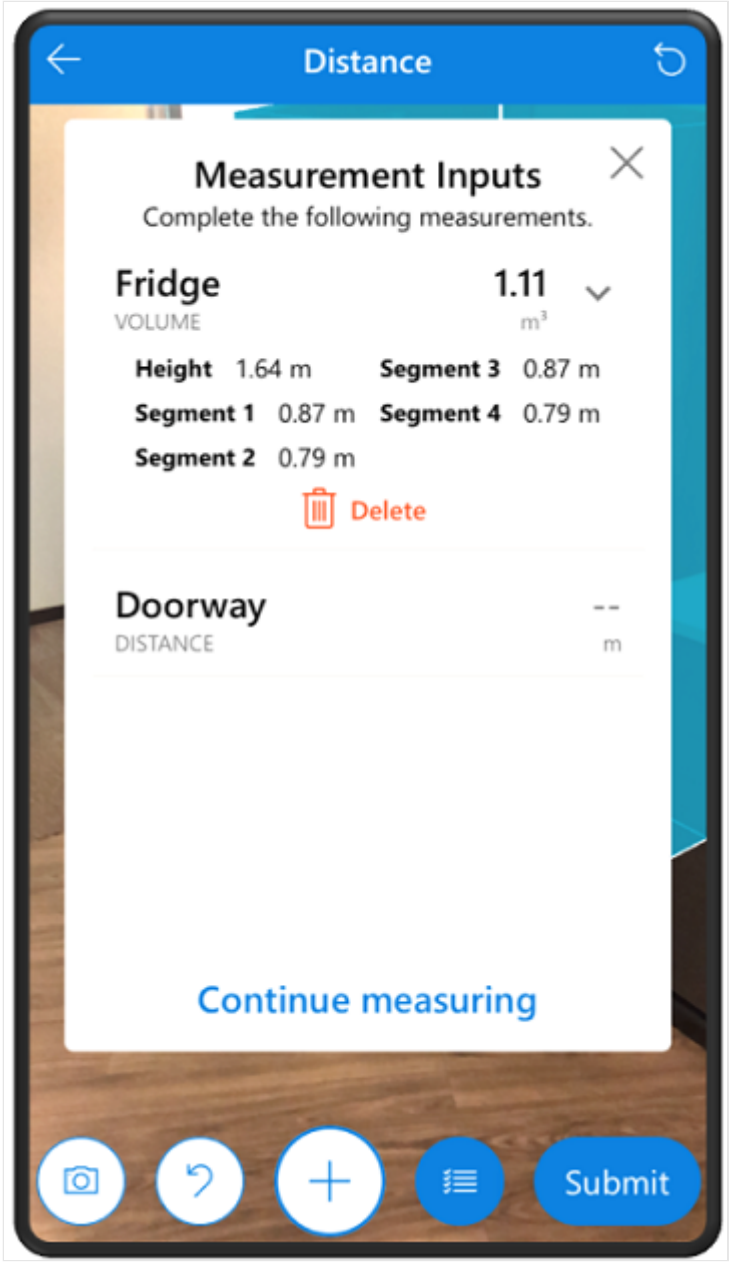


2. If your app maker has given you specific measurements to take, you can see their types here, beneath their label. Select the arrow next to measurement to view more details.



3. On the **Measurement Inputs** screen, you can do the following:

- Select an incompleted measurement to measure it immediately.
- Select the delete button to clear or remove a measurement.
- Select the close button or overview button to continue measuring where you left off.


Distance
Measurement Inputs
Complete the following measurements.
Fridge
VOLUME
1.11
m³
Height 1.64 m
Segment 1 0.87 m
Segment 2 0.79 m
Segment 3 0.87 m
Segment 4 0.79 m
Delete
Doorway
DISTANCE
--
m
Continue measuring
Submit

Take and upload mixed-reality photos

Article • 08/05/2022

In this article, we'll create an app that can take photos of a mixed-reality session and upload them to a folder on OneDrive. We'll use the **View in MR** control in this example, but the **View shape in MR** and **Measuring camera** controls would work as well.

We'll cover the following tasks:

- Adding a **3D object** control to view and manipulate a sample 3D object
- Connecting the **3D object** control to a **View in MR** control to view the 3D object in the real world
- Adding a gallery control to view photos taken with the **View in MR** control
- Uploading the photos to OneDrive with a Microsoft Power Automate flow
- Uploading photos captured in mixed-reality to Dataverse

Prerequisites

- Create a blank canvas app.
- Create a folder called **MRPhotos** on OneDrive. You'll use this folder to store your uploaded photos.

> **Tip**
>
> The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

Add a button to take a photo of a 3D object in mixed reality

This example has three parts. First, we'll add a button that lets users take a photo of a 3D object in a mixed reality experience.

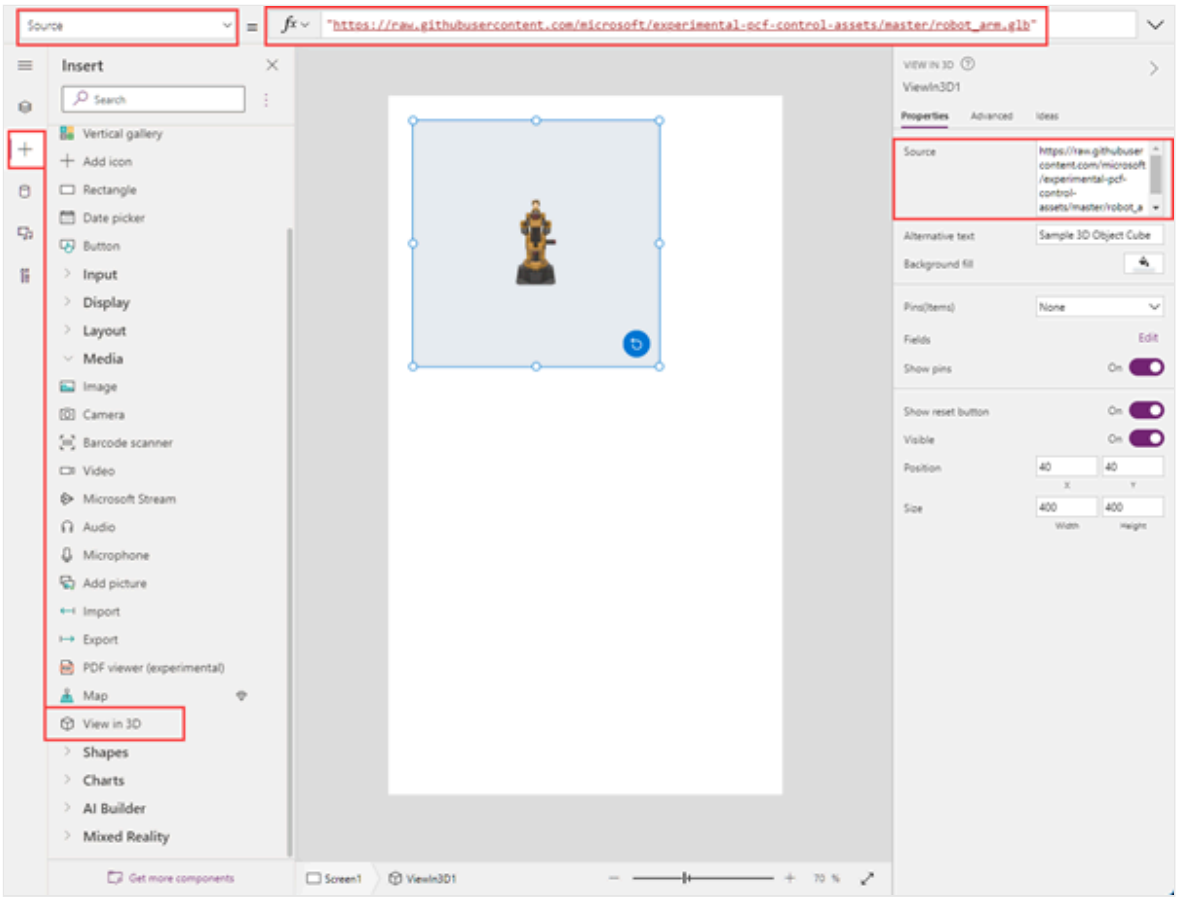
Insert a 3D object control

With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Media**.

2. Select **3D object** to place a 3D object on the app screen. Drag the control to the screen to position it more precisely.

   The control comes with a transparent cube shape. If you like, change the control's **Source** property to load a different 3D model. In this example, we'll use the URL *https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/robot_arm.glb*.

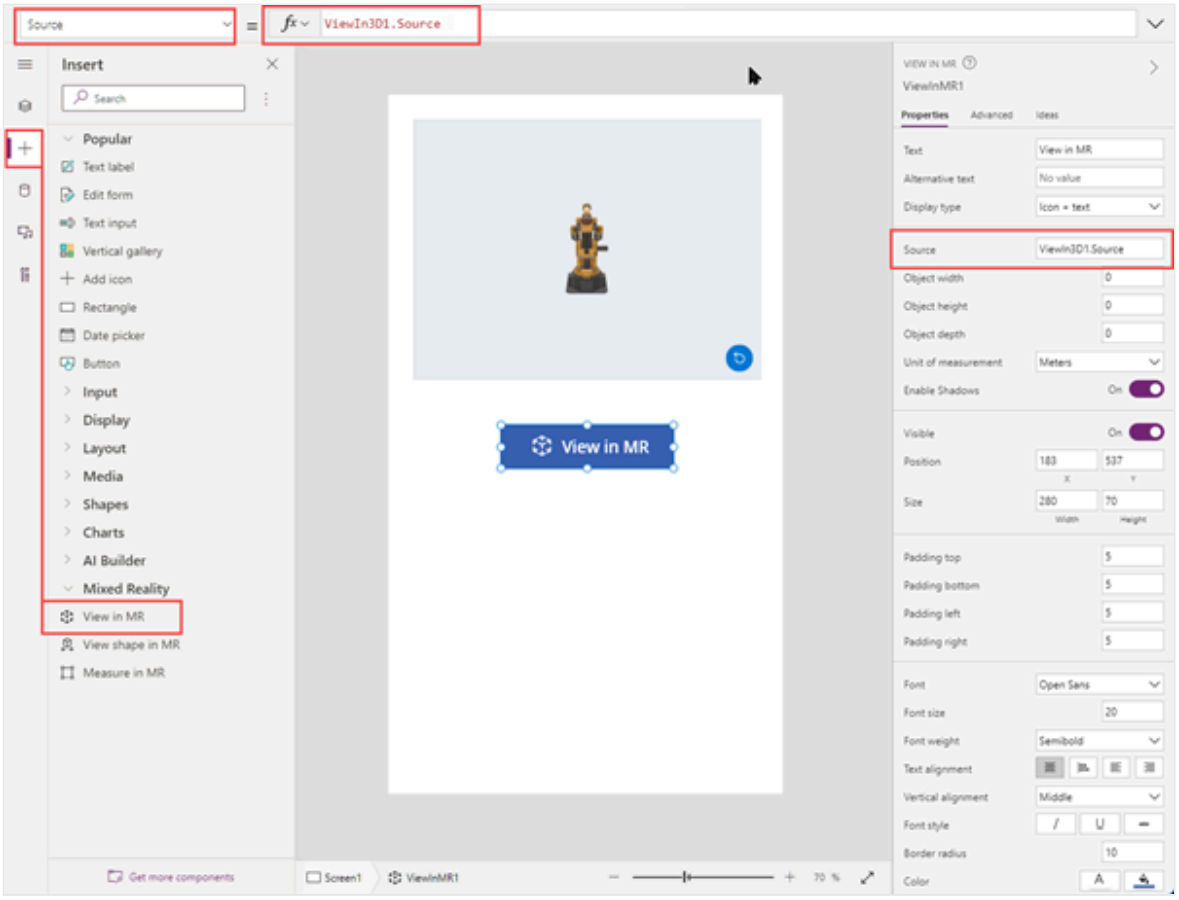


Insert and connect a View in MR control

1. Open the **Insert** tab and expand **Mixed Reality**.

2. Select **View in MR** to place the control on the app screen, or drag the control to the screen to position it more precisely.

3. Change the control's **Source** property to **3DObject1.Source**. (*3DObject1* is the name of the **3D object** control we added earlier.) This expression directs the **View in MR** control to overlay the 3D model on the device camera feed.



4. Save and publish the app and run it on your mobile device.

5. Select **View in MR** to view the 3D object in mixed reality. Select the camera icon to take a photo of the MR view.

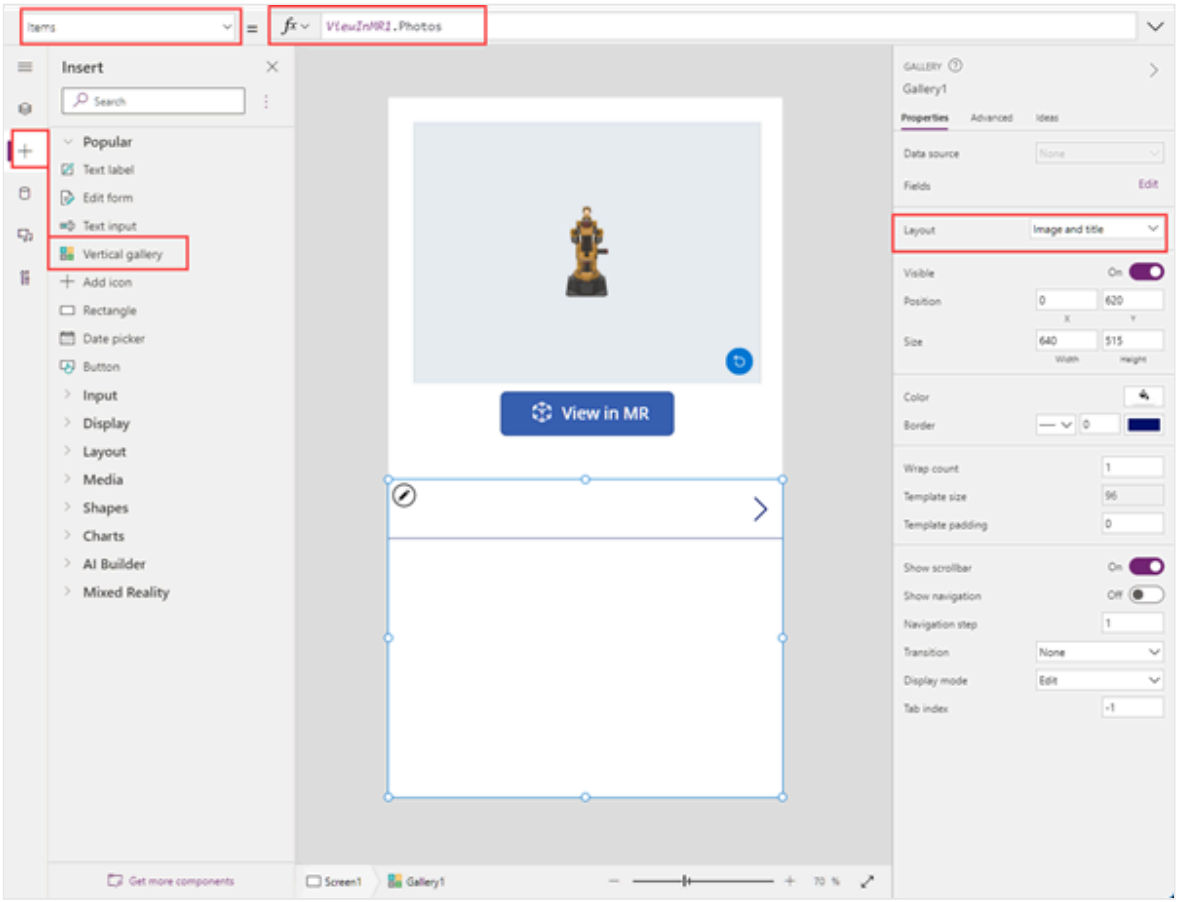
Insert a gallery control to view photos taken in the app

Next, we'll add a gallery so users can view the photos they've taken.

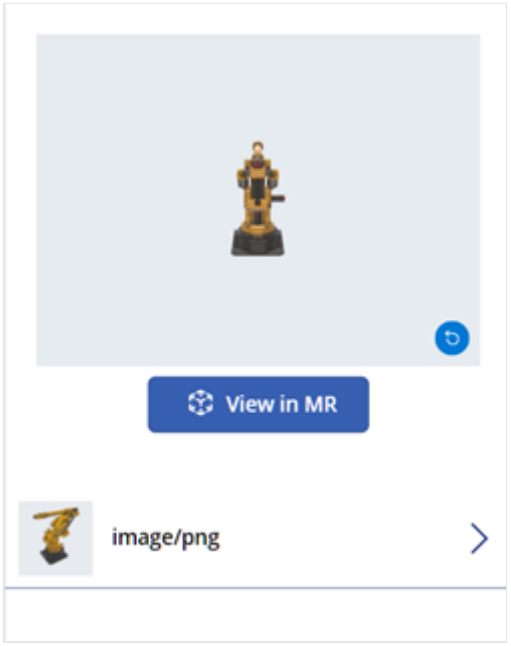
1. Edit your app again. Open the **Insert** tab and place a **Vertical gallery** control on the screen.

2. Change the control's **Items** property to **ViewInMR1.Photos**. (*ViewInMR1* is the name of the **View in MR** control we added earlier.)

3. Optionally, change the gallery's **Layout** property to **Image and title**.



4. Preview the app and select **View in MR** to generate a sample photo. The gallery populates with a sample picture.



> **Note**
>
> If users exit the MR view to see the gallery, then enter the MR view again and take more photos, the new photos will replace the ones they took earlier.

Add a larger overlay to the thumbnail images in the gallery

To make the photos in the gallery easier to see, you can add a full-size overlay that appears when the user selects a thumbnail image.

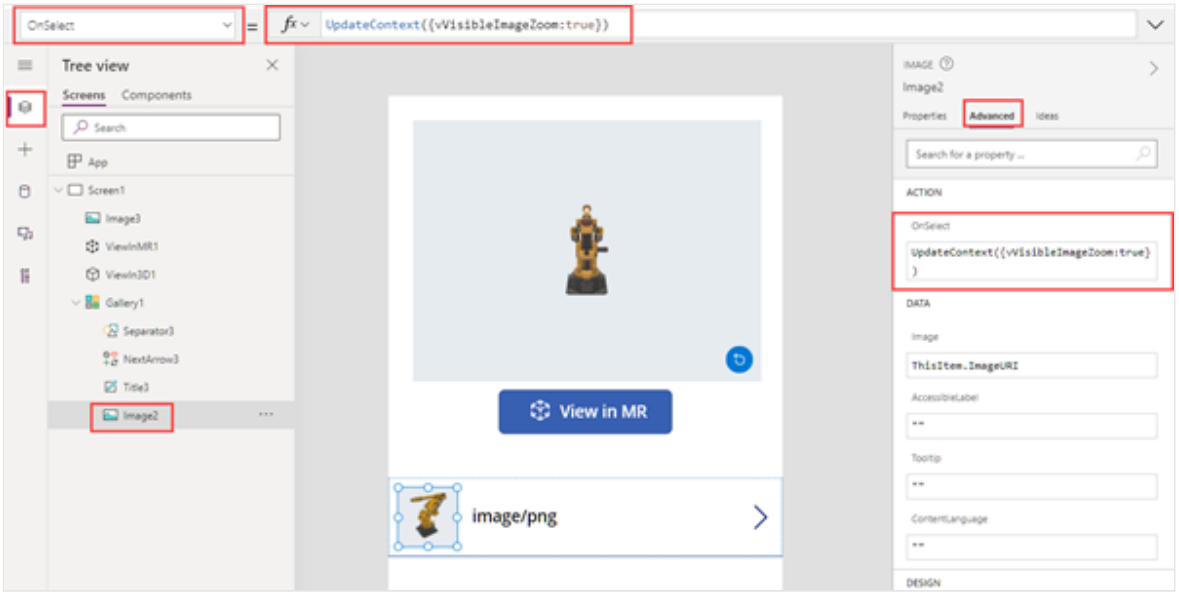
1. Edit your app again. Open the **Insert** tab and expand **Media**.

2. Select **Image** to place an image control on the screen. Move and size it according to how you want the larger photo to appear when a thumbnail image is selected.



3. Change the image control's properties as follows: | Property | Value | | - | - | | **OnSelect** | **UpdateContext({vVisibleImageZoom:false})** | | **Image** | **Gallery1.Selected.Image2** (assuming the gallery control is *Gallery1* and the first thumbnail image is *Image2*) | **Visible** | **vVisibleImageZoom**

4. Select the first thumbnail image in the gallery control. Change its **OnSelect** property to **UpdateContext({vVisibleImageZoom:true})**.



5. Save and publish the app and run it on your mobile device.

6. Select **View in MR**, and then select the camera icon to take a photo. Select the back arrow at the top of the screen to exit the MR view.

7. Select the thumbnail in the gallery to show a larger version of the photo. Select the image to hide it.

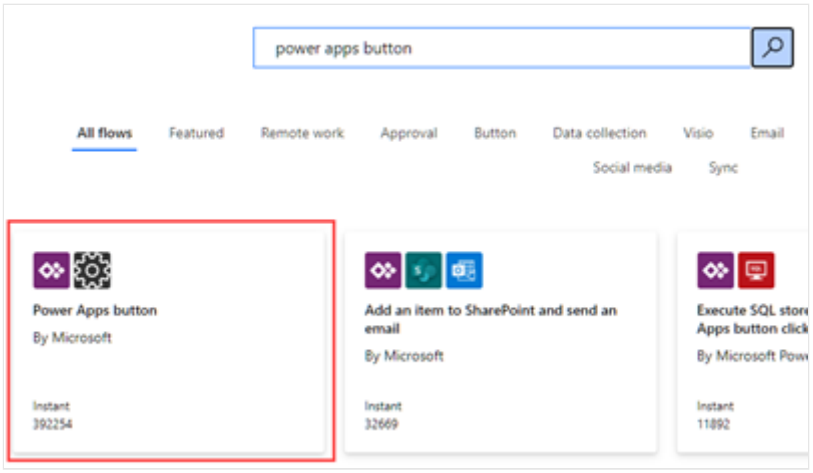
Upload photos to OneDrive with a Power Automate flow

Last, we'll create a workflow using the Power Automate pane. The workflow uploads photos from the app to a folder named **MRPhotos** on OneDrive.

Create a flow in Power Automate

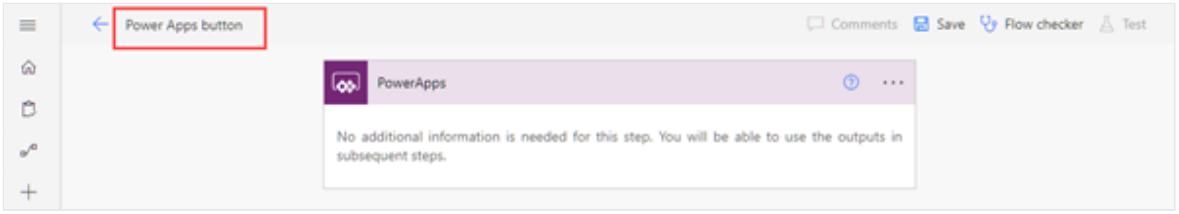
1. Edit your app. On the app authoring menu, select **Power Automate** > **Create new flow**.

2. Search for and select the Power Apps button template.

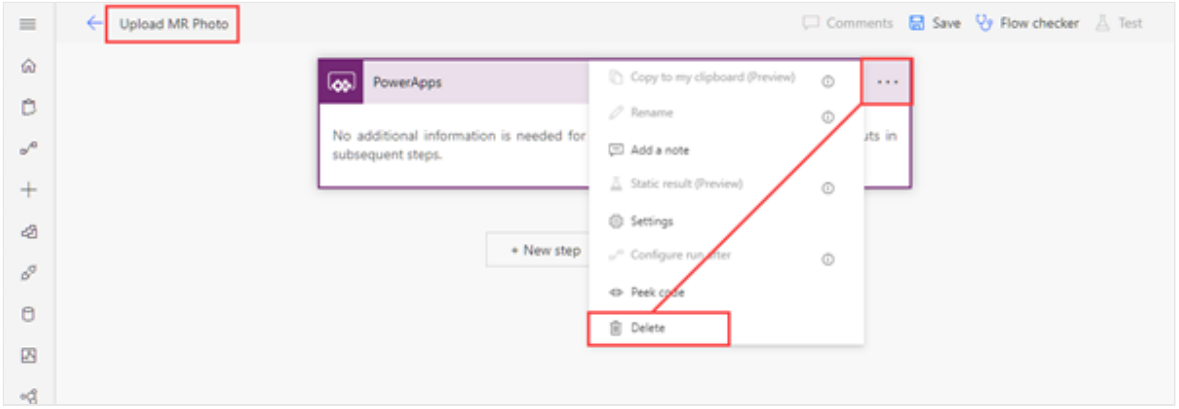


3. In the **Create your flow** window, select **Edit in advanced mode**.

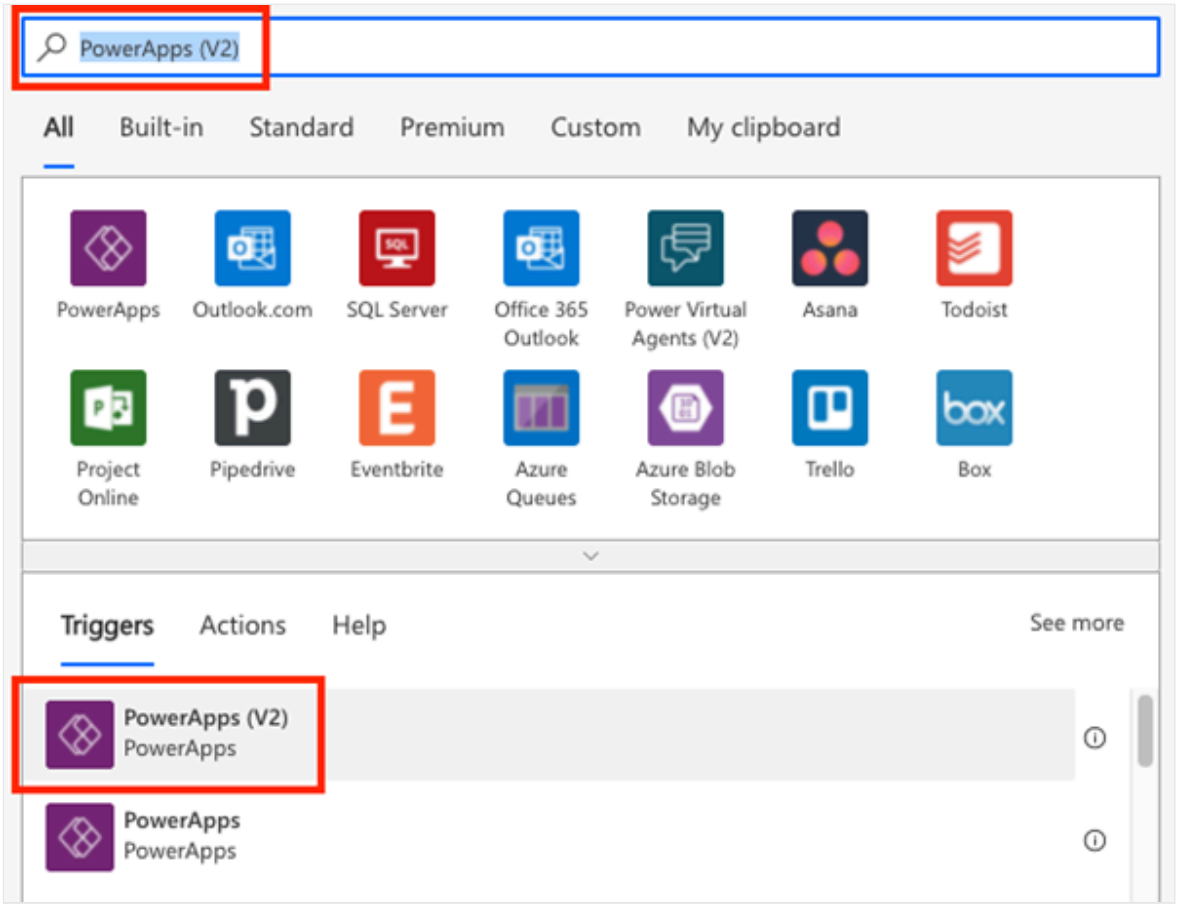
4. Select **Power Apps button** at the top of the window and enter a new name for your flow. In this example, we'll name the flow *Upload MR Photo*.



5. Select Power Apps button at the top of the window and enter a new name for your flow. In this example, we'll name the flow Upload MR Photo.

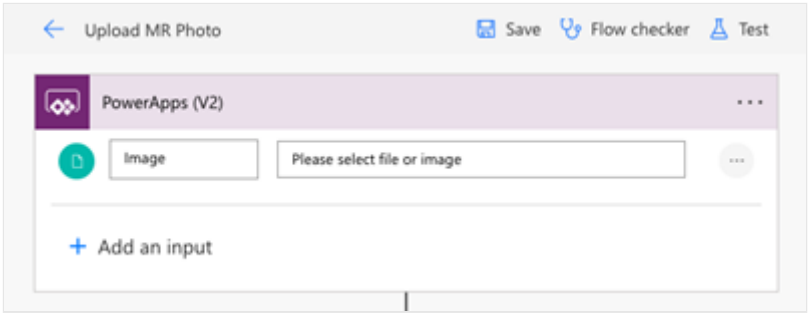


6. Search for **PowerApps (V2)** and select the PowerApps (V2) trigger.

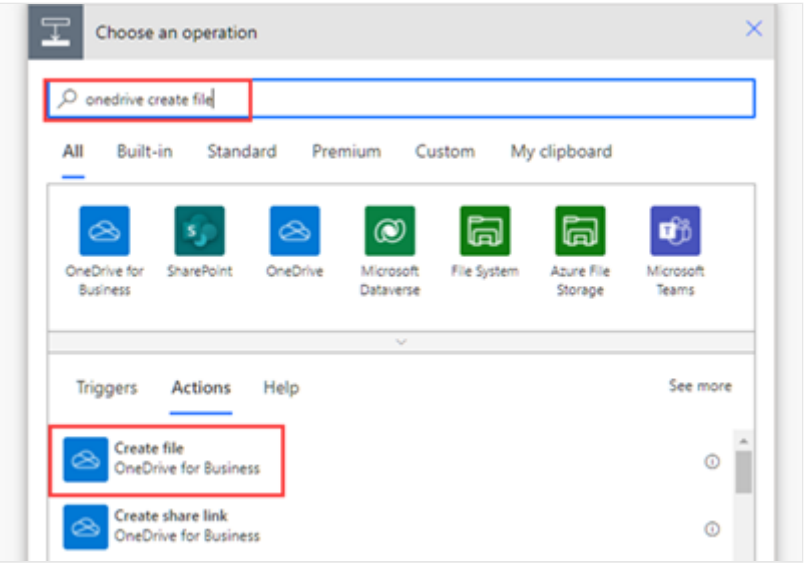


7. Select **Add an input**, and then select **File**.

8. Change the label *File Content* to **Image**.



9. Select **New step**. Search for **OneDrive Create file** and select the **Create file** action.



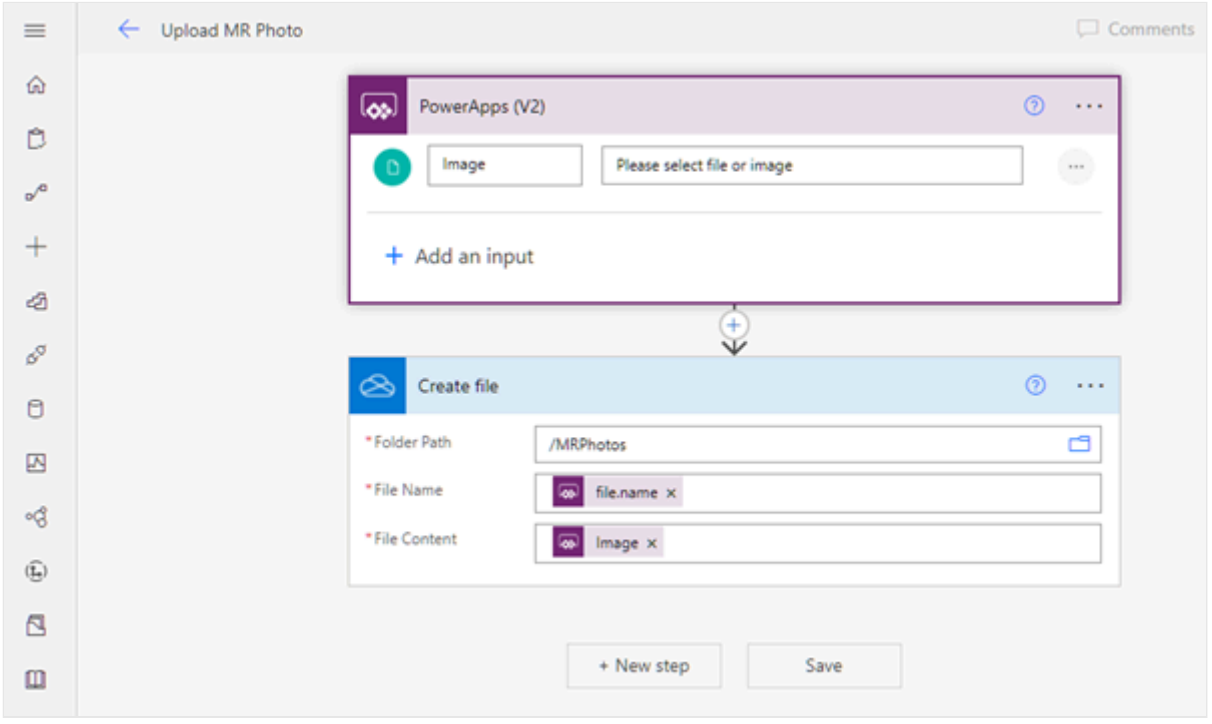
10. In **Folder Path**, select the folder icon and navigate to the **MRPhotos** folder you created earlier.

11. In **File Name**, enter **@{triggerBody()?['file']?['name']}** (Your text changes to "file.name.")

12. In **File Content**, enter **@{triggerBody()['file']['contentBytes']}** (Your text changes to "Image.")

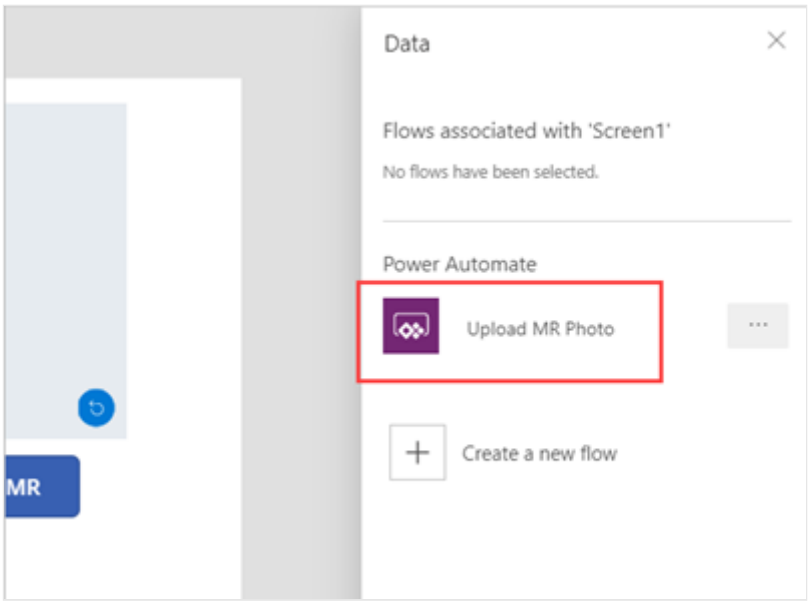
13. Save your flow.

The complete flow should look like this:



Connect the workflow to a button in your app

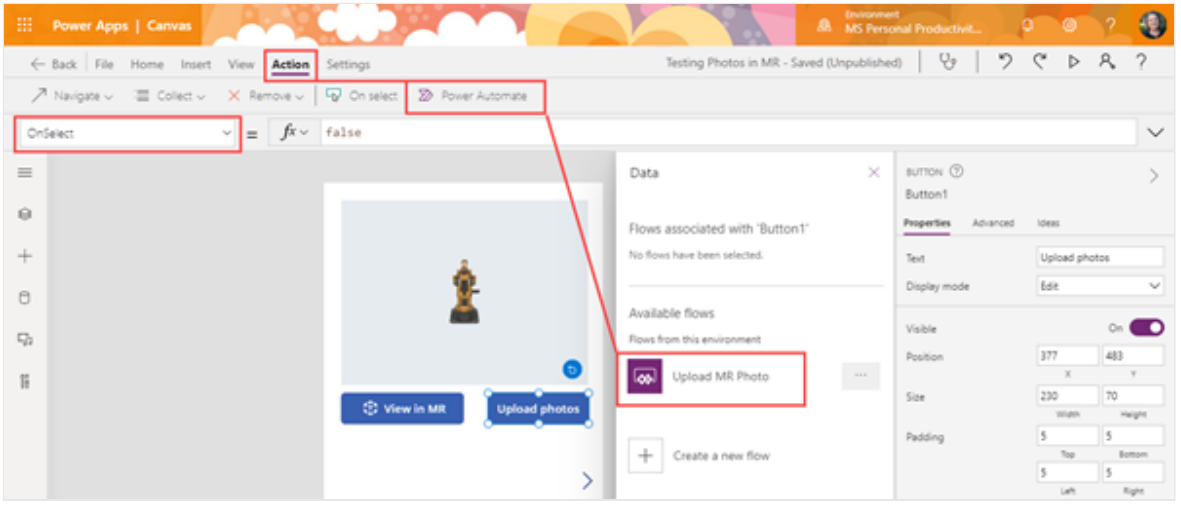
1. Return to your app in Power Apps Studio. Your flow is now listed under **Available flows**.



2. Open the **Insert** tab and select **Button**. Place the button control on the screen and resize it as needed.

3. Change the button control's **Text** property to **Upload photos**.

4. In the formula bar at the top of the Power Apps window, select the **OnSelect** property. Select **Action** > **Power Automate** > **Upload MR Photo**.

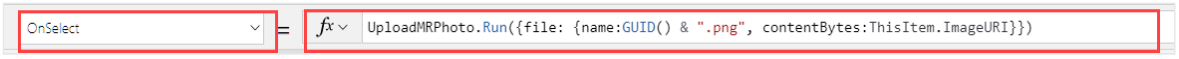


The button control's **OnSelect** property changes to **UploadMRPhoto.Run(**.

5. To upload the last photo taken, paste the following code after the opening parenthesis: **{file:{name:GUID() & ".png", contentBytes:Last(ViewInMR1.Photos).ImageURI}})**



If you placed the button control inside the gallery, paste the following code instead: **{file: {name:GUID() & ".png", contentBytes:ThisItem.ImageURI}})**



To make the button upload all the photos taken, delete **UploadMRPhoto.Run(** and paste the following code: **ForAll(ViewInMR1.Photos, UploadMRPhoto.Run({file: {name:GUID() & ".png", contentBytes:ImageURI}}))**



6. Preview the app and select **View in MR**, and then select **Upload photos**. Check the **MRPhotos** folder on OneDrive and confirm that the sample photo has been uploaded.

Add offline capability to your app

You can use your app even when you have limited or no network connectivity using the **SaveData** and **LoadData** functions.

Upload photos captured in mixed-reality to Dataverse

You can add photos to Dataverse tables through an Image data type column. Image columns in Dataverse have two required fields - Full and Value - which can be set to the ImageURI output of the MR controls.

For example, if you wanted to upload the first photo captured by the Markup in MR control to a Dataverse column called Image:

Power Fx

```
    Image: {Full: First(MarkupInMR.Photos).ImageURI, Value:
First(MarkupInMR.Photos).ImageURI}
```

See also

- 3D object control
- Measuring Camera control
- View shape in MR control

Feedback

Was this page helpful? Yes No

Provide product feedback

Load 3D models from CGTrader into mixed reality controls (preview)

Article • 07/15/2022

[This article is pre-release documentation and is subject to change.]

> **Important**
>
> This is a preview feature.
>
> Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Want to build apps with mixed-reality 3D content without having to create your own 3D models? Use the CGTrader connector to import 3D content from CGTrader into your canvas apps. CGTrader offers a library of thousands of free and premium 3D models. Browse and load models inside your app, and then use them with the 3D object and View in MR controls.

The connector provides a no-code replacement for working with the CGTrader API. Using *actions*, apps send requests to the CGTrader API and fetch model data. For more information, see the Connectors overview.

Prerequisites

Request an API key (OAuth client ID and client secret) at https://www.cgtrader.com/developers. Be sure to store your API key in a secure location.

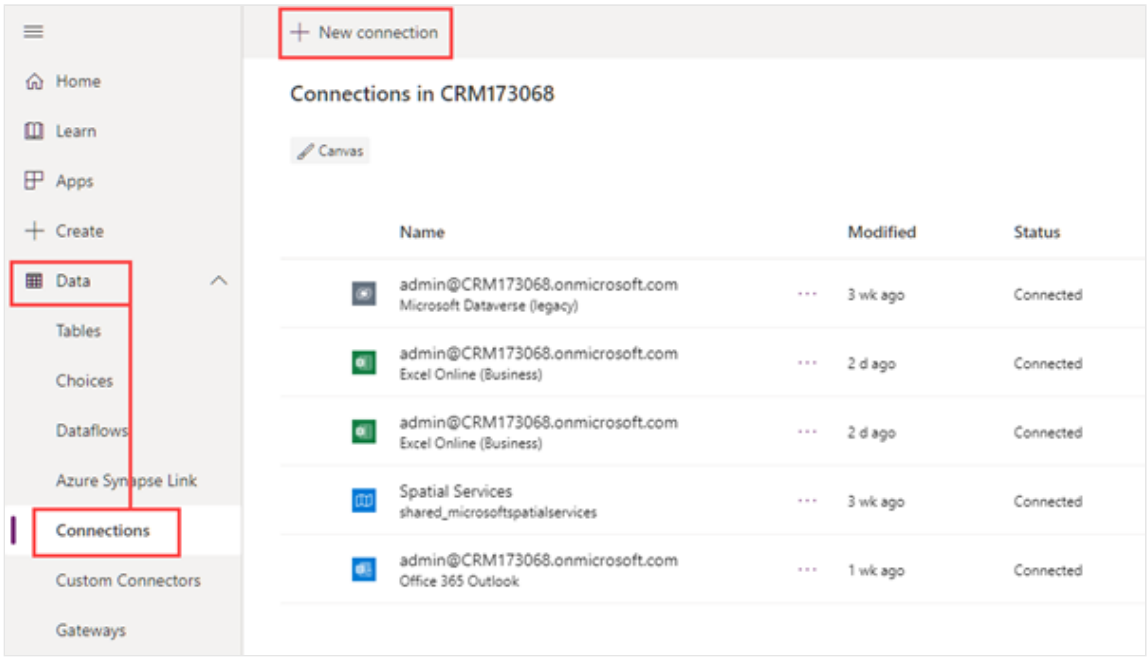
Create a CGTrader connection

You'll need to create a CGTrader connection in Power Apps after you receive your API key. In this example, we'll create a connection on the Power Apps home page. You can also create a connection when you're editing an app in Power Apps Studio.

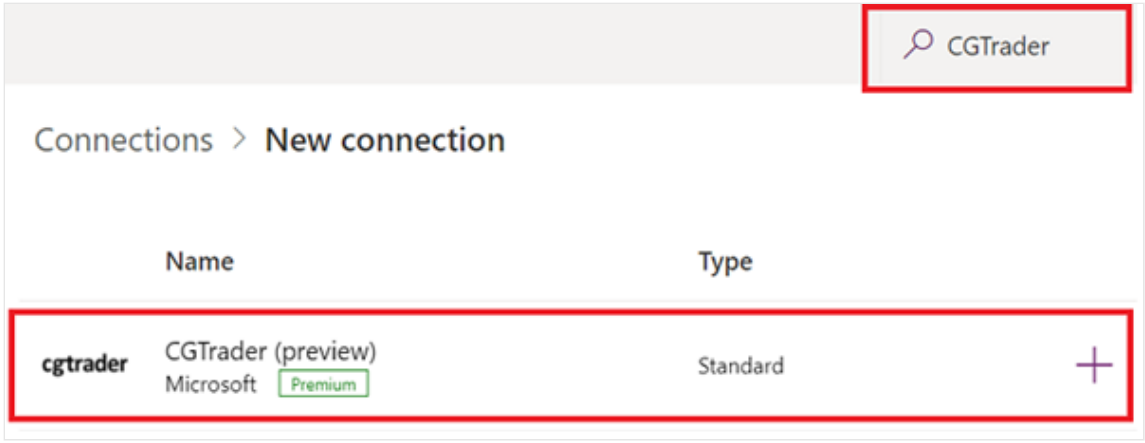
> **Tip**
>
> A connection is an instance of a connector. You can create multiple connections with different configurations.

1. Sign in to Power Apps.

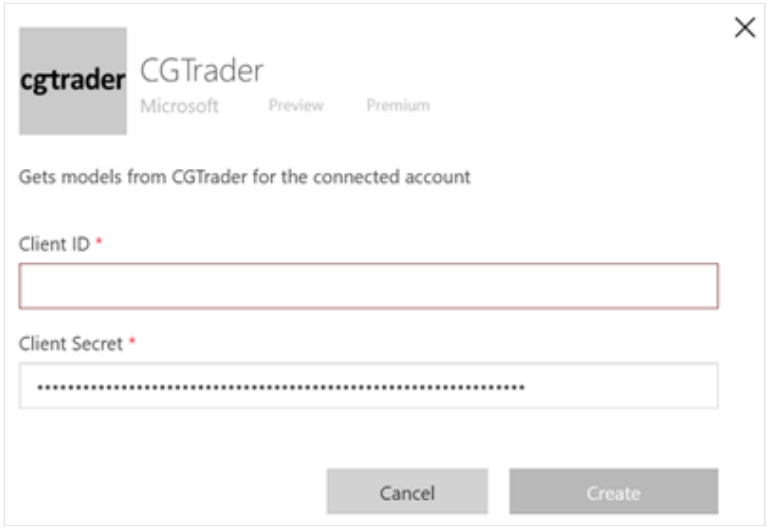
2. In the navigation bar, select **Data** > **Connections** > **New connection**.



3. Search for and select **CGTrader (preview)**.



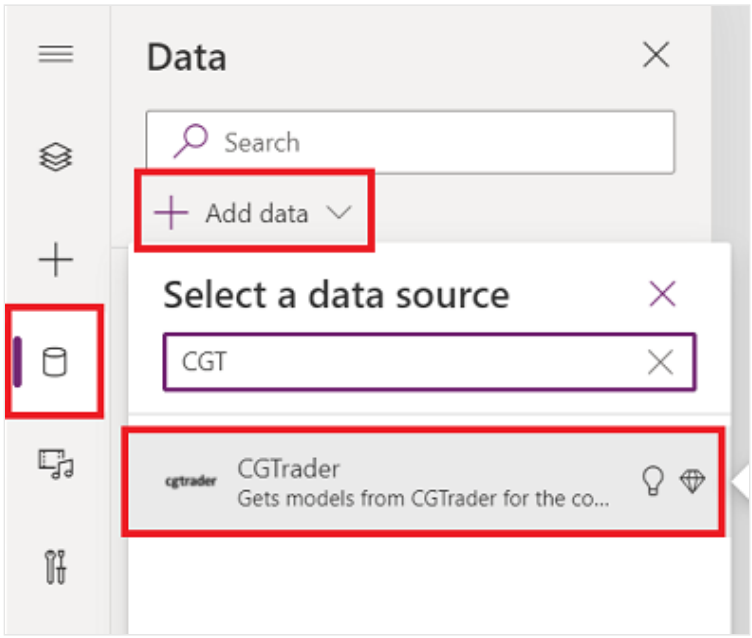
4. Enter the **Client ID** and **Client Secret** you received from CGTrader.



A CGTrader connection is now listed on the **Connections** page.

Add a CGTrader connection to your canvas app

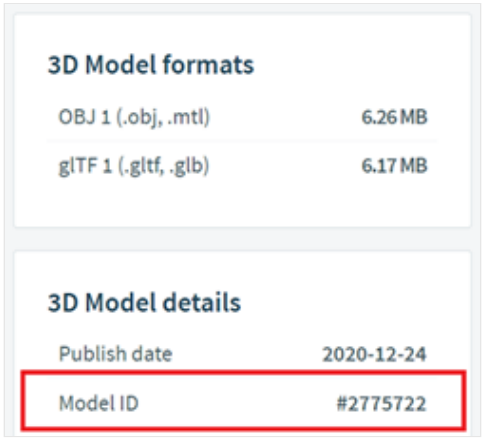
1. Edit or create a canvas app.
2. In the navigation bar, select **Data** > **Add data**. Search for and select **CGTrader**.



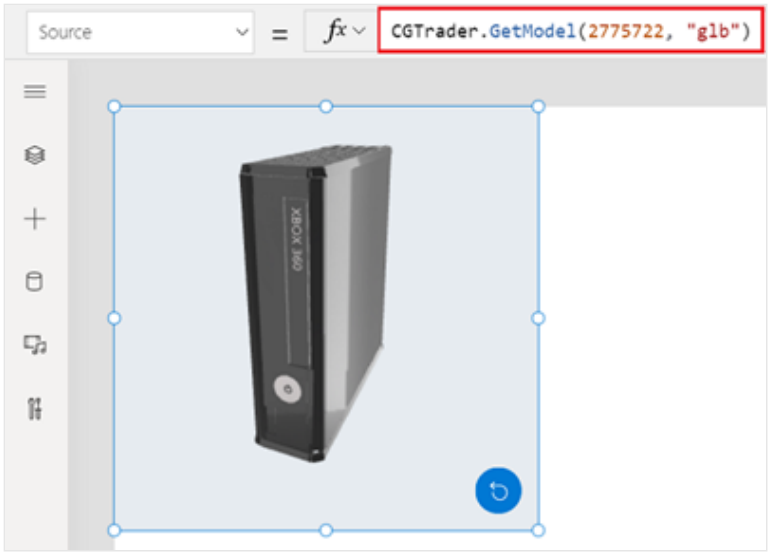
3. Select the connection you created earlier.

Load CGTrader models directly into 3D object or View in MR controls

1. Find and select a model on the CGTrader website.
2. Note the supported file types and Model ID.



3. In Power Apps Studio, add a **3D object** or **View in MR** control to the app screen.
4. Set the control's **Source** property to **CGTrader.GetModel(*model_id*, "*file_type*")**, where *model_id* is the ID of the model you selected, and *file_type* is one of the supported file types.



Connector actions

The following table lists the most common CGTrader connector actions and examples. For a list of all connector actions, their parameters, and return types, see the CGTrader connector reference.

Expand table

Action	Description	Example
GetModel	Downloads a 3D object with the given Model ID and file type.	CGTrader.GetModel(*model_id*, "*file_type*")
GetModelInfo	Gets information about a 3D object with the given Model ID.	CGTrader.GetModelInfo(*model_id*)
SearchModels	Searches for models from CGTrader based on the given filters. Consider binding this action to	CGTrader.SearchModels({keywords:"*keywords*",extensions:"*file_types*"}).Models

Action	Description	Example
	the items in a gallery.	
GetCategories	Gets available category names and Model IDs.	**CGTrader.GetCategories()**

See also

- View 3D content in canvas apps
- View 3D content or images in the real world

Use the Markup in MR control in Power Apps (preview)

Article • 07/15/2022

[This article is pre-release documentation and is subject to change.]

> **Important**
>
> This is a preview feature.
>
> Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Add the **Markup in MR** control in your app to allow users to add markups to visually specify an area or a particular asset in their environment. Users can instantly place mixed reality markups such as arrows, add drawings, and then take photos of the marked-up object.

The **Markup in MR** control creates a **Markup in MR** button in your mixed reality app. When a user selects the **Markup in MR** button, it shows a live camera feed of the device. A user can identify points of interest to begin inking or adding arrows directly on the object.

Ink and arrows will scale appropriately depending on the distance from the objects. When you exit the **Markup in MR** screen screenshots are captured in the control's **Photos** output property.

> **Tip**
>
> The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

> **Tip**
>
> The MR controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the **Babylon sandbox** should work in Power Apps through this shared MR platform. If your content works in Babylon but not in Power

Apps, ask a question in the **Power Apps Community Forum**. (Tag it with "mixed reality.")

Use Markup in MR

When a user selects the **Markup in MR** button, they will be prompted to move their phone side-to-side for better object or area detection. Once an object or area is detected the user can start using the markup toolbar.

Add an arrow

To position an arrow:

- Make sure the arrow is selected in the markup toolbar.
- Move your device around to position the dot on the object or area you want. to highlight
- Select to place the ghost arrow on your device screen.
- To rotate the arrow, select and drag the arrow in a circular motion (360 degrees).

Inking

To determine where you start drawing:

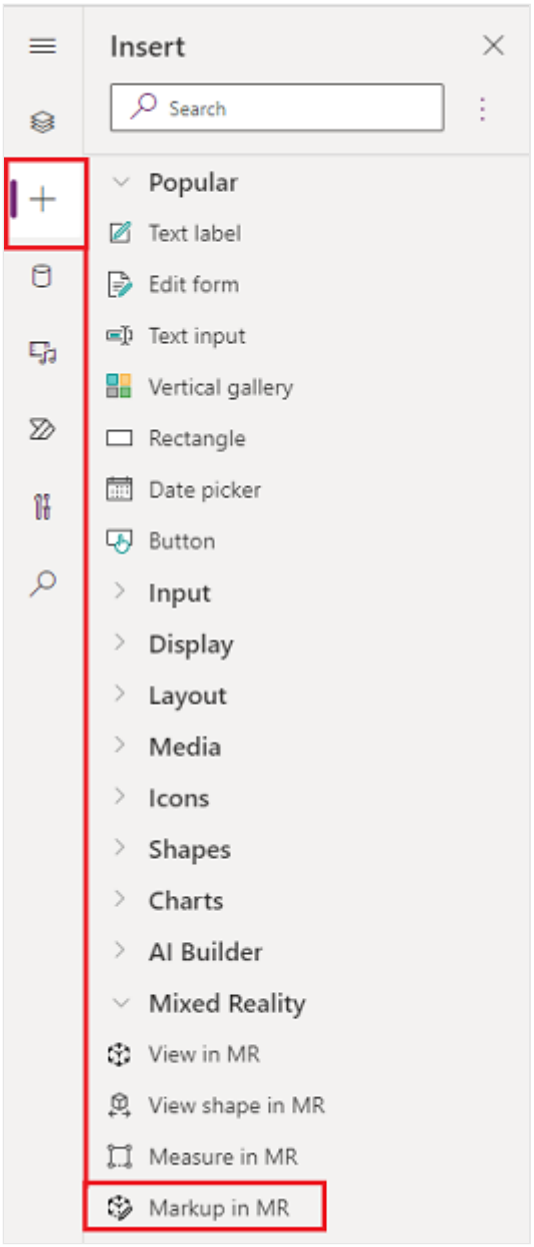
- Make sure the ink pen is selected in the markup toolbar.
- Select and drag on the screen to start drawing.

Add the Markup in MR control

Insert the control into your app as you normally would for any other button control.

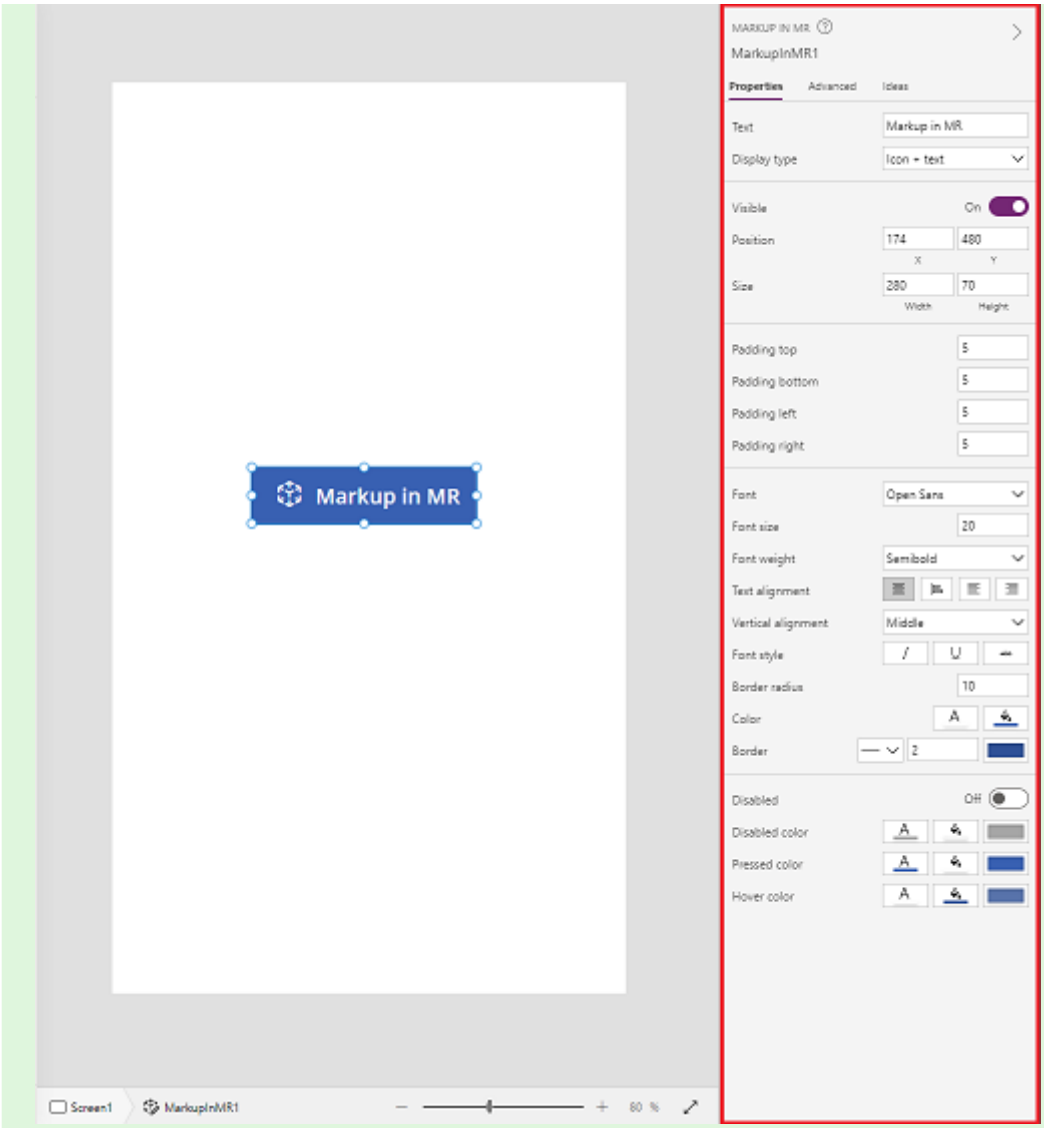
With an app open for editing in Power Apps Studio:

1. Open the **Insert** tab.
2. Expand **Mixed reality**.
3. Select the control **Markup in MR** to place it in the center of the app screen or drag and drop it to position it anywhere on the screen.



You can modify the control with several properties.

> **Tip**
>
> Some properties are only available under **More options** in the **Advanced** tab on the **Markup in MR** pane.



Property	Description	Type	Location
Photos	The photos captured during the mixed reality session. Note: You can upload the mixed-reality photos to OneDrive and show them in a gallery.	Not Applicable	Not Applicable (Output property only)
OnMixedRealitySelect	Behavior that is triggered when exiting the MR experience with new results.	Defined Action	Advanced
OnChange	Behavior that is triggered when any property on the button is changed.	Defined Action	Advanced

Additional properties

Property	Description
BorderColor	The color of a control's border.

Property	Description
BorderStyle	Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.
BorderThickness	The thickness of a control's border.
Color	The color of text in a control.
DisplayMode	Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).
DisabledBorderColor	The color of a control's border if the control's DisplayMode property is set to **Disabled**.
DisabledColor	The color of text in a control if its DisplayMode property is set to **Disabled**.
DisabledFill	The background color of a control if its DisplayMode property is set to **Disabled**.
FillColor	The background color of a control.
Font	The name of the family of fonts in which text appears.
FontStyle	The style of the text in the control: **None**, **Strikethrough**, **Underline**, or **Italic**.
FontSize	The font size of the text that appears on a control.
FontWeight	The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**
Height	The distance between a control's top and bottom edges.
HoverBorderColor	The color of a control's border when the user keeps the mouse pointer on that control.
HoverColor	The color of the text in a control when the user keeps the mouse pointer on it.
HoverFill	The background color of a control when the user keeps the mouse pointer on it.
PaddingBottom	The distance between text in a control and the bottom edge of that control.
PaddingLeft	The distance between text in a control and the left edge of that control.
PaddingRight	The distance between text in a control and the right edge of that control.
PaddingTop	The distance between text in a control and the top edge of that control.
PressedBorderColor	The color of a control's border when the user taps or clicks that control.

Property	Description
PressedColor	The color of text in a control when the user taps or clicks that control.
PressedFill	The background color of a control when the user taps or clicks that control.
TabIndex	Keyboard navigation order.
TextAlignment	The alignment of the text: **Center**, **Left**, **Right**, or **Justify**
Tooltip	Explanatory text that appears when the user hovers over a control.
VerticalAlign	The location of text on a control in relation to the vertical center of that control: **Middle**, **Top**, or **Bottom**
Visible	Whether a control appears or is hidden.
Width	The distance between a control's left and right edges.
X	The distance between the left edge of a control and the left edge of its parent container (or the screen if there's no parent container).
Y	The distance between the top edge of a control and the top edge of the parent container (or the screen if there's no parent container).

Output properties

Property	Description	Type
Photos	Collects the photos captured during the mixed reality session. You can upload the mixed-reality photos to OneDrive and show them in a gallery.	Not applicable

Other Mixed Reality controls

- View 3D content with the View in 3D control.
- View images and 3D content in the real world with the View in mixed reality control.
- Create and view predefined 3D shapes with the View shape in mixed reality control.
- Measure distance, area, and volume with the Measure in mixed reality control.

Use marker detection with mixed-reality controls (preview)

Article • 07/15/2022

[This article is pre-release documentation and is subject to change.]

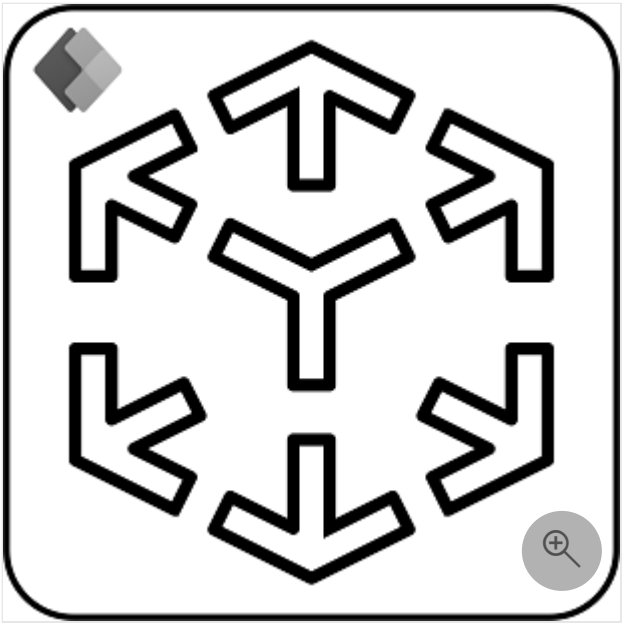
> **Important**
>
> This is a preview feature.
>
> Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Use physical markers to improve the precision and persistence of media that's placed using the View in MR and View shape in MR controls. This topic covers how to enable marker detection and how to add markers to your space.

Prerequisites

In addition to the standard device compatibility requirements outlined in add mixed reality controls to your canvas app, marker detection requires physical markers to be present in your space.

To create a marker, print out the image below or display the image on a monitor or tablet. For best results, the marker should be printed in portrait orientation on standard A4 or 8.5" x 11" letter paper. Ensure that the marker is fully visible and not cut off on either the top or bottom. The marker should be approximately 20 x 20 cm for optimal tracking speed and accuracy.

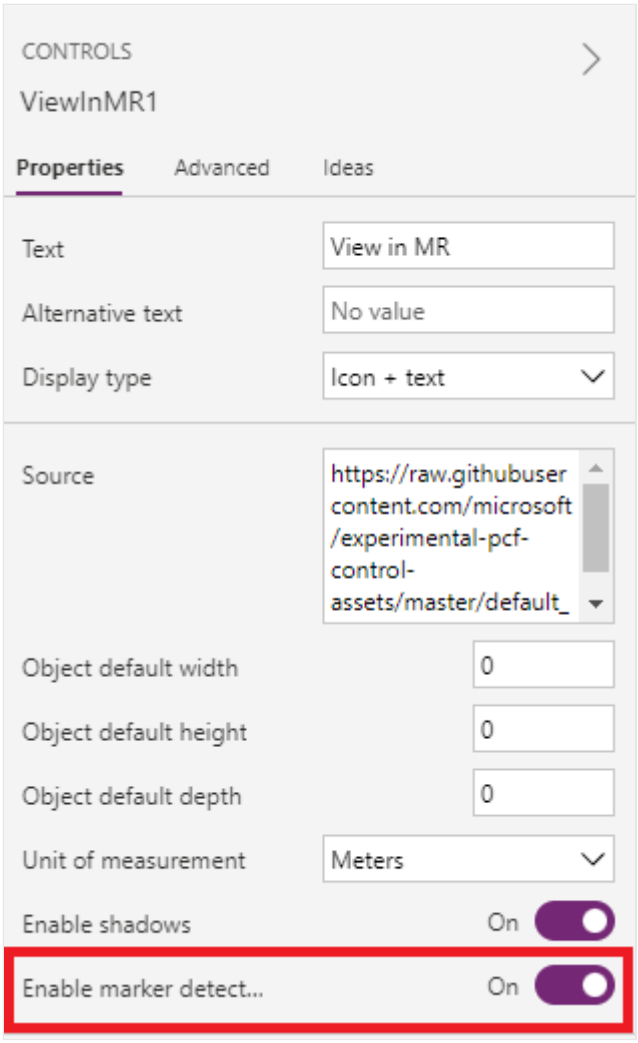


Enable marker detection

Enable marker detection for the **View in MR** and **View shape in MR** controls.

With an app open for editing in Power Apps Studio:

1. Open the **Insert** tab.
2. Expand **Mixed reality**.
3. Select the **View shape in MR** or **Markup in MR** control.
4. On the **Properties** tab, set the **Enable marker detection** property to **On** in the properties panel or the formula bar.

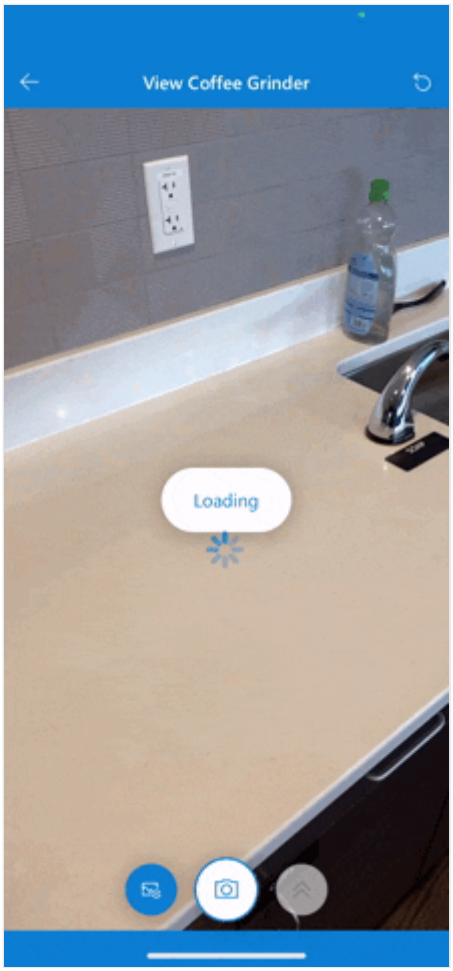


Use marker detection on device

When you enter the mixed-reality experience, you'll be prompted to point your device at a physical marker in your space. To quickly acquire tracking, make sure the marker is fully in frame and about 2-5 feet from the device.



Once the marker is detected, the loaded media will be automatically aligned and placed directly on top of the marker. You can manipulate the media's position or rotate it by using the on-screen gestures or by physically moving the marker.



If no marker is present or detection is taking too long, you can turn off the marker detection by selecting the highlighted marker detection button.



See also

- View in MR control
- View shape in MR control

Add geospatial controls to your canvas apps

Article • 12/16/2022

Add interactive maps with dynamic routing and address lookup to your canvas apps using geospatial controls.

Use the following prebuilt controls for geospatial applications:

- Interactive map
- Address input

The map control runs with limited features available by default. You can enable additional map control features and support for the address input control by turning on full support in the Power Platform admin center. Refer to the table at the bottom of the page for more details on data usage in limited and full support.

Prerequisites for full support

Before you can use geospatial features in your apps, your Power Platform administrator has a couple of tasks to complete:

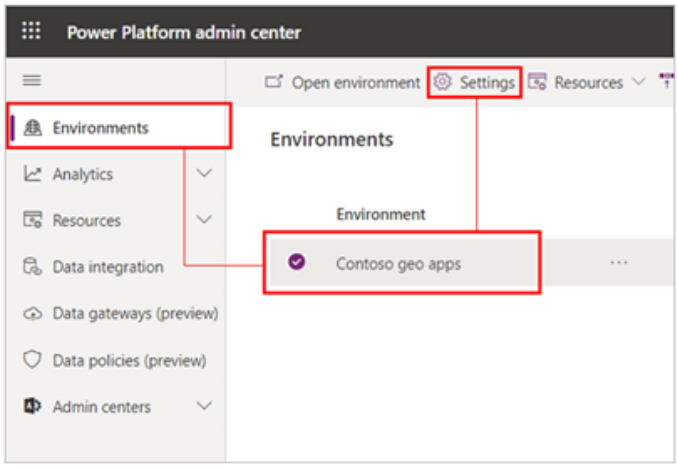
1. Enable geospatial features in the Power Platform admin center for the environment you intend to use. The geospatial features require their own terms of service, which your admin must review and agree to.
2. Make sure the geospatial controls aren't blocked by the environment's data loss prevention policies.

> ⓘ **Important**
>
> Geospatial controls are not available in **Power Apps US Government**.

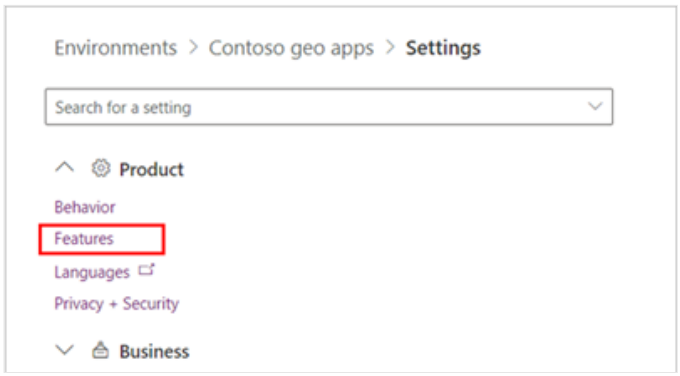
Enable geospatial features for the environment

If you don't have administrative access to the Power Platform admin center, ask your admin to enable geospatial features for you. You can similarly follow these steps to disable geospatial features.

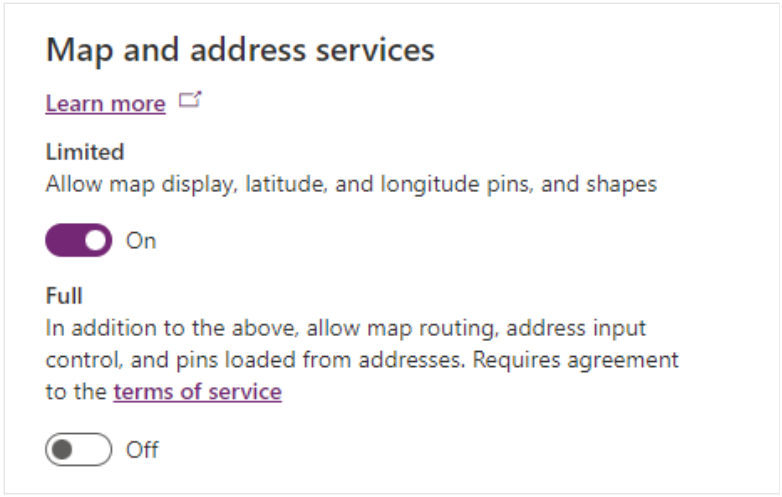
1. Open the Power Platform admin center.
2. On the **Environments** tab, select the environment you want to use for your apps, and then select **Settings**.



3. Expand **Product**, and then select **Features**.



4. Under **Map and address services**, turn on the **Full** toggle. You can use the toggles in this section to similarly turn on and off limited support as well.



5. Read the terms of service. Select **I agree to the terms of service**, and then select **Enable**.

> **Important**
>
> The terms of service must be agreed to before you can use all geospatial features in your apps.
>
> Mapping capabilities are provided by a third party. Microsoft shares address and location queries, but not customer or user names, with the provider. Requests are sent securely over HTTPS or aren't exposed to the public Internet.

6. Select **Save**.

Review the environment's data loss prevention policies

Geospatial controls use the Spatial Services connector to fetch map tiles and to look up and geocode addresses. Your Power Platform admin must make sure data loss prevention policies that apply to that connector don't conflict with the policies that affect the controls. Learn more about data loss prevention policies for Power Platform.

Privacy and security considerations

- Power Apps doesn't link search queries to any user or tenant when shared with TomTom, and the shared search queries can't be used to identify individuals or tenants.
- Azure Maps doesn't store the request information sent by you. For more information about Azure Maps compliance, see Azure global compliance
- Requests sent between TomTom and Azure Maps are not exposed over the public Internet.
- Requests sent between apps you create with the geospatial controls and Azure Maps are sent over HTTPS.
- The following table describes the user data that Power Apps sends to Azure Maps, Bing Maps, and TomTom on full support:

Expand table

Control	Feature	Data	Purpose	Sent to Azure Maps	Sent to Bing Maps	Sent to TomTom	User identifiers or tracking data sent	Enabled in Full Support	Enabled in Limited Support
Map	Show map tiles	Coordinates in and around the map view	To show the map tiles in the map view.	Yes	No	Yes	No	Yes	Yes
Map	Show pins and shapes from coordinates	Pin and shape coordinates	Feature does not require sending coordinate data.	No	No	No	No	Yes	Yes
Map	Show pins and route waypoints	Pins and route waypoint addresses	To translate addresses to latitude/longitude coordinates, and show them on the map.	No	Yes	Yes	No	Yes	No
Map	Show routes	Route waypoint coordinates	To calculate routes between waypoints.	Yes	No	Yes	No	Yes	No
Address Input	Address Search	Address search query string	To show address search results.	No	Yes	Yes	No	Yes	No

Control	Feature	Data	Purpose	Sent to Azure Maps	Sent to Bing Maps	Sent to TomTom	User identifiers or tracking data sent	Enabled in Full Support	Enabled in Limited Support
Address Input	Address Search	Current device location	To bias address search results around the device location.	No	Yes	Yes	No	Yes	No

Next steps

Add the controls to your apps:

- Visualize and interpret location data with the **Interactive map** control.
- See dynamic address suggestions as you type with the **Address input** control.

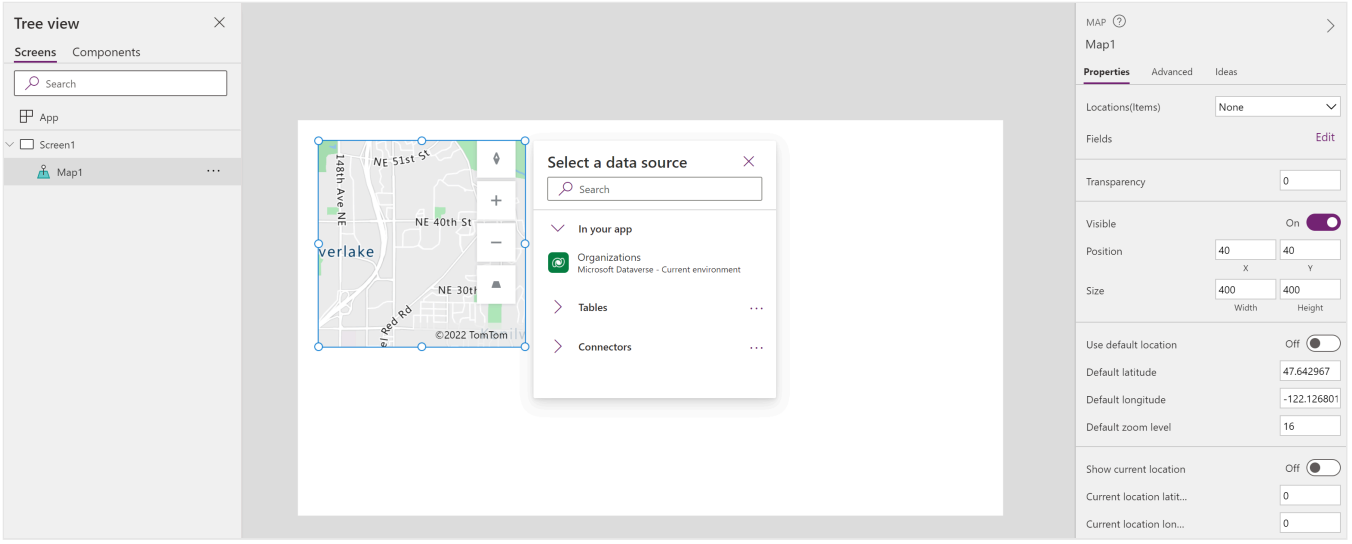
See also

Create an app that uses the address input and map controls

Interactive map control

Article • 03/15/2023

Easily add an interactive map to your canvas apps. Plot markers from a data source containing addresses or latitude and longitude pairs. As you zoom out, the markers will optionally cluster to condense groups of data. On mobile devices and web experiences, a map can show the user's current position and calculate a route to the user's destination. Maps can be switched between road and satellite views.



Interactive map features

- Use a data source to insert pins
- Display information about map pins
- Show routes between waypoints
- Draw and insert shapes

Prerequisites

You'll need to satisfy a few prerequisites before you can use maps in your canvas apps. Refer to the privacy and security table for more details on how different map features use data.

Add a map to an app screen

With your app open for editing in Power Apps Studio :

1. Select the **Insert** tab and expand **Media**.
2. Select **Map** to place a map in the app screen, or drag the control to the screen to position it more precisely.

Adding pins, routes, and shapes

Pins, routes, and shapes are data set properties that need to both identify a data source, which is a table from a collection or connector, and the relevant columns. The data source is set in the Items property (Items for pins, RouteWaypoints_Items for routes, Shape_Items for shapes) and the relevant columns are set in the related properties (e.g. ItemLatitudes, ItemLongitudes, etc for pins). The Properties section contains additional information about these data sets and their related properties.

For example, if you had a table collection named **Locations** with **Name**, **Longitude**, and **Latitude** columns:

Expand table

Name	Longitude	Latitude
Fourth Coffee (sample)	-98.29277	26.2774
Litware, Inc. (sample)	-96.85572	32.55253
Adventure Works (sample)	-96.99952	32.72058

To display these as labeled pins on the map:

1. Set the **Items** property to Locations
2. Set the **ItemLabels** property to "Name"
3. Set the **ItemLongitude** property to "Longitude"
4. Set the **ItemLatitude** property to "Latitude"

> **Important**
>
> The relevant column names need to be in quotation marks in the related properties. The data source should not have quotation marks.

You can see how to do display pins from an Excel sheet or build an app that uses the address input control to populate pins on the map control as additional examples.

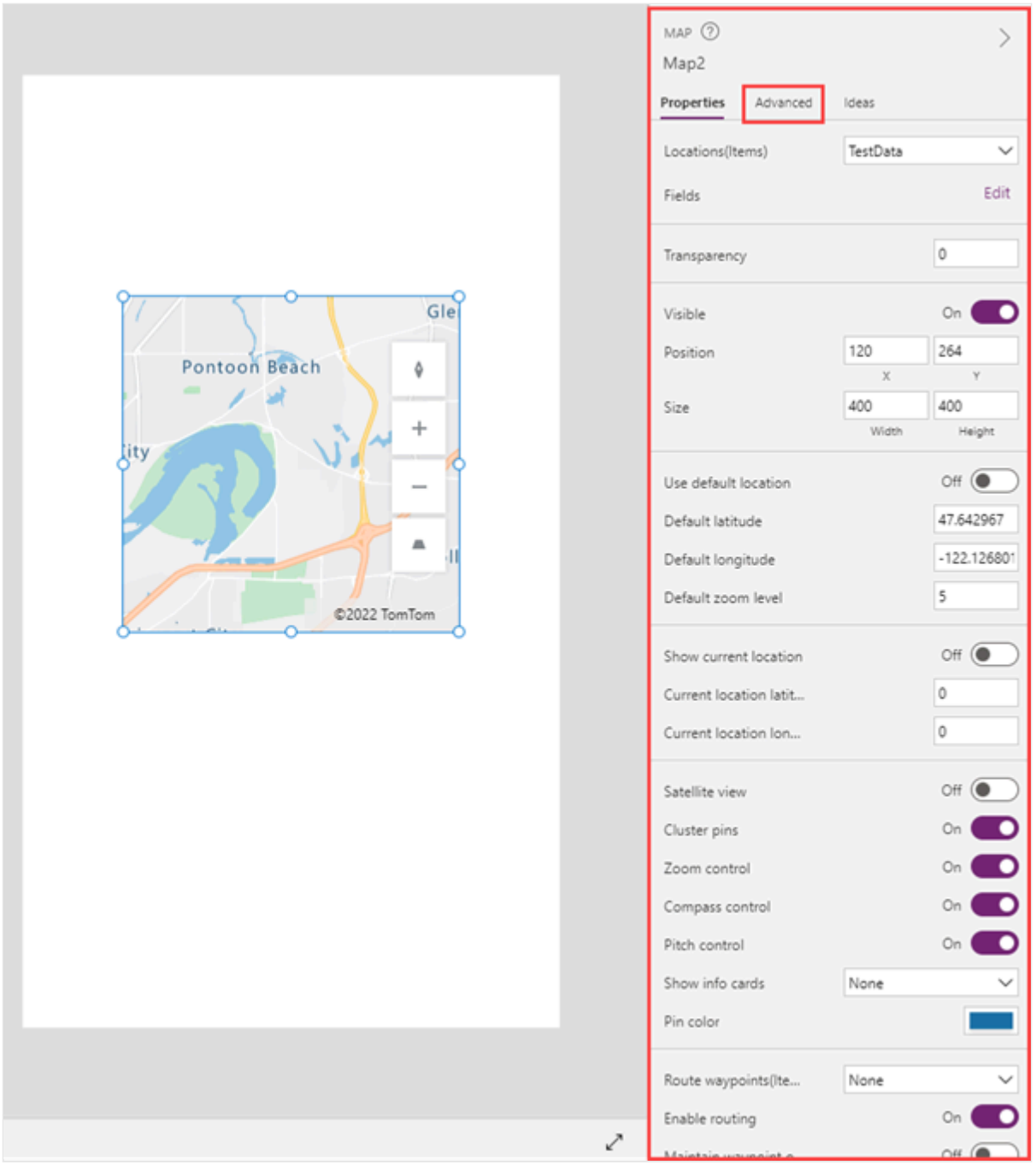
> **Note**
>
> - Each map control can display up to 5000 pins from latitude or longitude and 50 pins from addresses. The pin limit is lower for addresses as the map needs to geocode

these addresses into latitude or longitude to display them. We recommend not using addresses where possible. You can save geocoded addresses back to your data source.

- When both latitude or longitude and an address is given for a single pin, the map control will prioritize using the latitude or longitude to avoid geocoding the address.
- The maximum number of shapes that can be drawn in a map control is limited to 500.

Properties

Change a map's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



The map control has six different types of properties:

1. Styling properties
2. Behavioral properties
3. Pin properties
4. Route properties
5. Shape properties
6. Output properties

Styling properties

Expand table

Property	Description	Type	Tab
Satellite view	Displays the map in satellite view. Leave this property off to display the map in road view.	Boolean	Properties; Advanced: **SatelliteView**
Map style	Sets the map style. Options: Road, Night, Road shaded relief, Satellite, Satellite road labels, High contrast light, High contrast dark, Grayscale light, Grayscale dark.	Enum	Properties
Transparency	Determines the map's transparency, from 0 (opaque) to 100 (transparent).	Integer	Properties; Advanced: **Transparency**
Visible	Shows or hides the map.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the map at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the map using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Border radius	Determines the corner radius of the map border.	Floating point number	Properties; Advanced: **BorderRadius**
Border	Determines the style, width, and color of the map border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderColor**
DisplayMode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced

Behavioral properties

Expand table

Property	Description	Type	Tab
Show info cards	Shows information about a mapped location when the user selects it (*On click*) or points to it (*On hover*). If *None*, no information is shown.	Enum	Properties; Advanced: **InfoCards**
Use default location	Initializes the map at a default location.	Boolean	Properties; Advanced: **DefaultLocation**

Property	Description	Type	Tab
Default latitude	Sets the latitude coordinate the map shows if **Use default location** is enabled.	Floating point number	Properties; Advanced: **DefaultLatitude**
Default longitude	Sets the longitude coordinate the map shows if **Use default location** is enabled.	Floating point number	Properties; Advanced: **DefaultLongitude**
Default zoom level	Sets the zoom level if **Use default location** is enabled, from 0 to 22.	Integer	Properties; Advanced: **DefaultZoomLevel**
Show current location	Displays the user's current location.	Boolean	Properties; Advanced: **CurrentLocation**
Current location latitude	Sets the latitude coordinate of the current location pin that the map shows if **Show current location** is enabled. To place the pin at the user's current location, set this property to Location.Latitude.	Floating point number	Properties; Advanced: **CurrentLocationLatitude**
Current location longitude	Sets the longitude coordinate of the current location pin that the map shows if **Show current location** is enabled. To place the pin at the user's current location, set this property to Location.Longitude.	Floating point number	Properties; Advanced: **CurrentLocationLongitude**
Zoom control	Shows the zoom control.	Boolean	Properties; Advanced: **Zoom**
Compass control	Shows the compass control.	Boolean	Properties; Advanced: **Compass**
Pitch control	Shows the pitch (tilt) control.	Boolean	Properties; Advanced: **Pitch**
TabIndex	Specifies the order the map is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **Tab index**
Tooltip	Determines the text to display when the user hovers over a visual.	String	Advanced
ContentLanguage	Determines the display language of the map, if it's different from the language used in the app.	String	Advanced
OnLoad	Contains code that runs when the map is loaded.	Event	Advanced

Property	Description	Type	Tab
OnMapClick	Contains code that runs when the user selects the map. The latitude and longitude of the clicked point is in the **ClickedLocation** output property.	Event	Advanced
OnChange	Contains code that runs when any aspect of the map is changed.	Event	Advanced
OnSelect	Contains code that runs when the user selects something on the map.	Event	Advanced

Pin properties

Expand table

Property	Description	Type	Tab
Locations(Items)	Identifies a data source (**Items**) in the form of a table from which to get locations to show on the map. The table lists sets of longitudes and latitudes, or physical addresses, to display as pins. Using latitude or longitude is recommended as addresses need to be geocoded and so have a more restrictive pin limit. The table can be a collection or from a data source like Excel Online. Each row must have an entry for label, longitude, and latitude, or a physical address, and optionally the pin color and icon.	Not applicable	Properties; Advanced: **Items**
ItemsLabels	Identifies the column in **Items** that contains the labels for the pins.	ColumnName	Advanced
ItemsLatitudes	Identifies the column in **Items** that contains the latitude position of the pins.	ColumnName	Advanced
ItemsLongitudes	Identifies the column in **Items** that contains the longitude position of the pins.	ColumnName	Advanced
ItemsAddresses	Identifies the column in **Items** that contains addresses that represent the location of the pins. There is a limit to how many pins can be displayed from addresses. We recommend geocoding your addresses to latitude, longitude pairs and using those to display pins when possible.	ColumnName	Advanced
ItemsColors	Identifies the column in **Items** that contains the colors of the pins.	ColumnName	Advanced

Property	Description	Type	Tab
ItemsIcons	Identifies the column in **Items** that contains the icons of the pins.	ColumnName	Advanced
Cluster pins	Groups nearby map pins instead of displaying them individually.	Boolean	Properties; Advanced: **Clustering**
Pin color	Determines the default color of pins shown on the map. This color is overridden by the **ItemsColors** property if set	Color picker	Properties; Advanced: **PinColor**
OnItemsChange	Contains code that runs when pins on the map are changed.	Event	Advanced

Route properties

Expand table

Property	Description	Type	Tab
Enable routing	Determines whether the user can request directions to a specified location.	Boolean	Properties; Advanced: **UseRouting**
Route waypoints(Items)	Shows route waypoints, as provided in a data source (**RouteWaypoints_Items**) in the form of a table. The table can be a collection or from a data source like Excel Online. If *None*, no waypoints are shown.	Enum	Properties; Advanced: **RouteWaypoints_Items**
RouteWaypointsLabels	Identifies the column in **RouteWaypoints_Items** that contains the labels for the waypoints.	ColumnName	Advanced
RouteWaypointsLatitudes	Identifies the column in **RouteWaypoints_Items** that contains the latitude position of the waypoints.	ColumnName	Advanced
RouteWaypointsLongitudes	Identifies the column in **RouteWaypoints_Items** that contains the longitude position of the waypoints.	ColumnName	Advanced

Property	Description	Type	Tab
RouteWaypointsAddresses	Identifies the column in **RouteWaypoints_Items** that contains addresses that represent the location of waypoints.	ColumnName	Advanced
Maintain waypoint order	Determines whether a calculated route maintains waypoints in the order provided.	Boolean	Properties; Advanced: **RouteMaintainOrder**
Optimize route	Determines whether a calculated route is optimized for distance, time, or isn't optimized.	Enum	Properties; Advanced: **RouteOptimization**
Route travel mode	Determines whether a route is calculated for a car or a truck, which may require avoiding bridges with certain height or weight restrictions.	Enum	Properties; Advanced: **RouteTravelMode**
OnRouteDirectionChange	Contains code that runs when the app detects the user has changed direction while on a computed route.	Event	Advanced

Shape properties

Expand table

Property	Description	Type	Tab
Show shapes	Shows the shapes in **Shapes_Items**.	Boolean	Properties; Advanced: **ShowShapes**
Shapes_Items	Identifies a data source (**Shapes_Items**) in the form of a table from which to get shapes to show on the map. The table can be a collection or from a data source like Excel Online. Each row must have an entry for the shape (GeoJSON object) and (optionally) a label and color.	TableName	Advanced

Property	Description	Type	Tab
ShapeGeoJSONObjects	Identifies the column in **Shapes_Items** with strings that represent the GeoJSON objects of the shapes, in shape collection or single shape GeoJSON format.	ColumnName	Advanced
ShapeLabels	Identifies the column in **Shapes_Items** that contains the labels for the shapes.	ColumnName	Advanced
ShapeColors	Identifies the column in **Shapes_Items** that contains the colors of the shapes.	ColumnName	Advanced
Show shape labels	Shows the shape labels, if provided.	Boolean	Properties; Advanced: **ShowShapeLabels**
Enable shape drawing	Shows drawing tools on the map.	Boolean	Properties; Advanced: **ShapeDrawing**
Enable shape deleting and label editing	Determines whether the user can delete shapes and edit their labels.	Boolean	Properties; Advanced: **ShapeEditingDeleting**
OnShapeCreated	Contains code that runs when the user adds a shape to the map.	Event	Advanced
OnShapeSelected	Contains code that runs when the user selects a shape on the map.	Event	Advanced
OnShapeEdited	Contains code that runs when the user modifies a shape on the map.	Event	Advanced
OnShapeDeleted	Contains code that runs when the user deletes a shape on the map.	Event	Advanced

Output properties

Other properties become available when a user interacts with a map. You can use these *output properties* in other controls or to customize the app experience.

Expand table

Property	Description	Type
CenterLocation	Captures the map's center point.	Not Applicable

Property	Description	Type
ClickedLocation	Captures the last location the user selected, either *.Latitude* or *.Longitude*.	Record
Selected	Captures the selected pin.	Record
SelectedItems	Captures the selected pin or pins in the selected cluster.	Table
GeocodedItems	Captures the geocoded locations of the pins.	Table
RouteWaypoints_Selected	Corresponds to the record of the selected shape in **RouteWaypoints_Items**.	Record
RouteWaypoints_SelectedItems	Corresponds to the records of the selected overlapping shapes in **RouteWaypoints_Items**.	Table
Shapes_Selected	Corresponds to the record of the selected shape in **Shapes_Items**.	Record
Shapes_SelectedItems	Corresponds to the records of the selected overlapping shapes in **Shapes_Items**.	Table
SelectedShape	Captures the *.Perimeter* and *.Area* of the selected shape.	Record
DeletedShape	Captures the *.Perimeter* and *.Area* of the last deleted shape.	Record
GeoJSON	Captures the list of shapes in Feature Collection GeoJSON format.	String

Other geospatial controls

To see dynamic address suggestions as you type, use the **Address input** control.

Known limitations

- Requires internet connection

See also

Create an app with address input and map controls

Use a data source to insert pins

Article • 03/15/2023

You can load a table that contains existing data into the map control. The control will then plot each row in your table as a map pin.

Prerequisites

1. Create a Canvas app and make sure it meets the Geospatial prerequisites.
2. In your app, insert a map.

Add pin data from an Excel workbook

Your data source needs to contain a named table with the following columns that should then be mapped to the associated property in the control's **Advanced** pane.

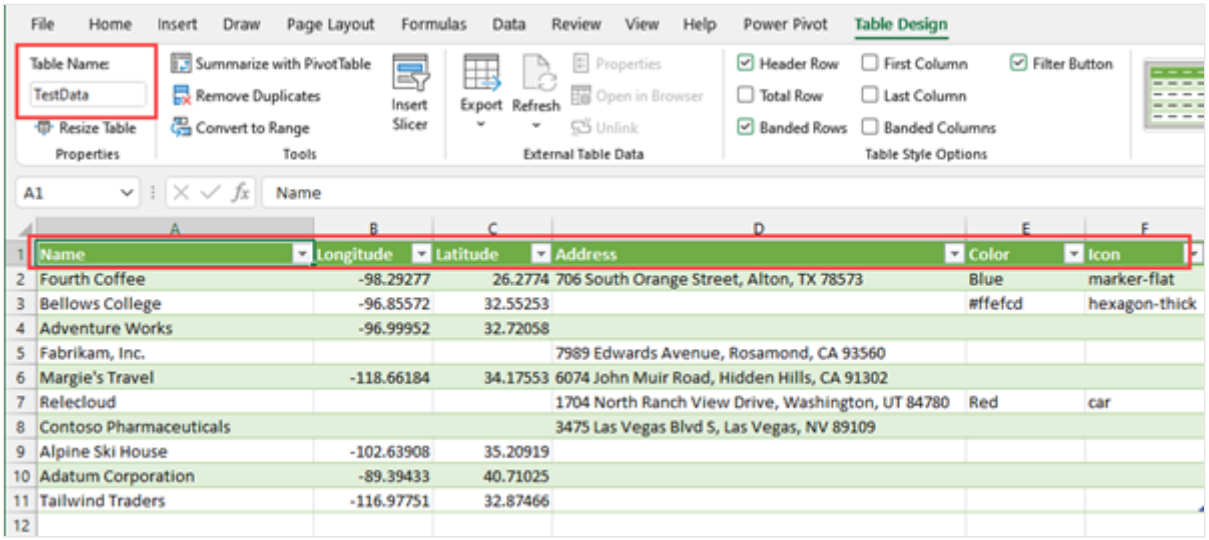
Expand table

Column description	Maps to property	Required
Label for the pin	ItemsLabels	Required
Longitude of the pin	ItemsLongitudes	Required
Latitude of the pin	ItemsLatitudes	Required
Color of the pin	ItemsColors	Optional
Icon for the pin	ItemsIcons	Optional

The color field accepts any CSS string, as defined in Color enumeration and ColorFade, ColorValue, and RGBA functions in Power Apps.

You can use the icons described in the List of image templates topic as your icon.

The following Excel table shows the required columns:



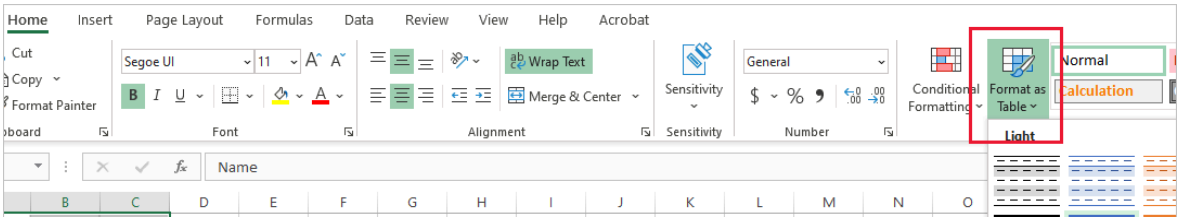
You can copy the following sample data to test this functionality:

Expand table

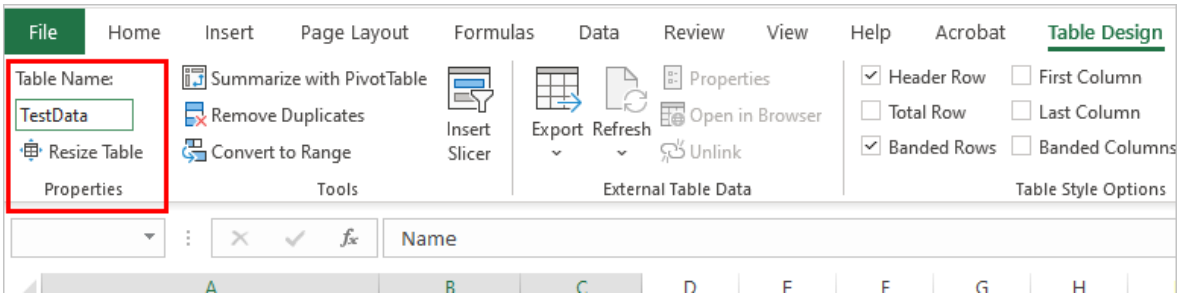
Name	Longitude	Latitude	Color	Icon
Fourth Coffee (sample)	-98.29277	26.2774	Blue	marker-flat
Litware, Inc. (sample)	-96.85572	32.55253	#ffefcd	hexagon-thick
Adventure Works (sample)	-96.99952	32.72058		car
Fabrikam, Inc. (sample)	-118.30746	34.86543		
Blue Yonder Airlines (sample)	-118.66184	34.17553		
City Power & Light (sample)	-113.46184	37.15363		
Contoso Pharmaceuticals (sample)	-80.26711	40.19918		
Alpine Ski House (sample)	-102.63908	35.20919		
A Datum Corporation (sample)	-89.39433	40.71025		
Coho Winery (sample)	-116.97751	32.87466		

1. Copy and paste the table into a new data source. In this example, we are using an Excel workbook.

2. Select one of the cells, and then on the Home tab in the ribbon, select **Format as Table** and choose any style, and then **OK**.



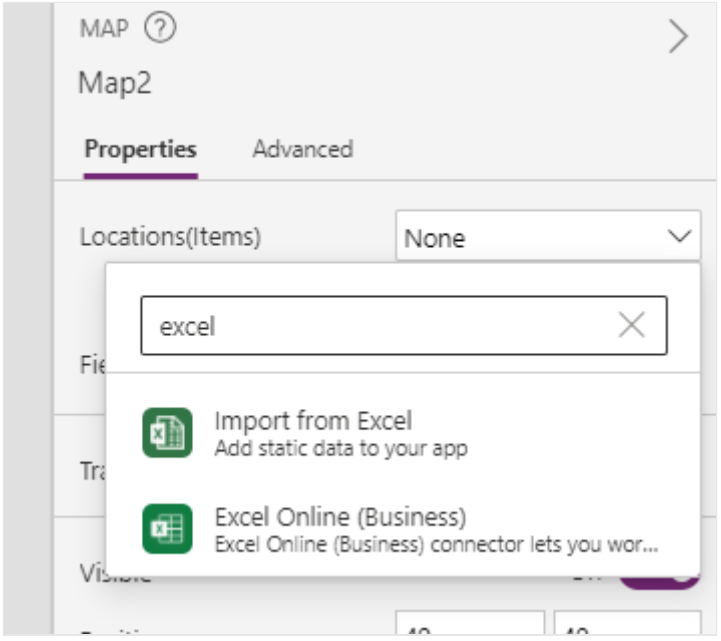
3. Select the table, and then go to the **Table Design** tab on the ribbon. Enter a name for the table under **Table Name:**, for example *TestData*.



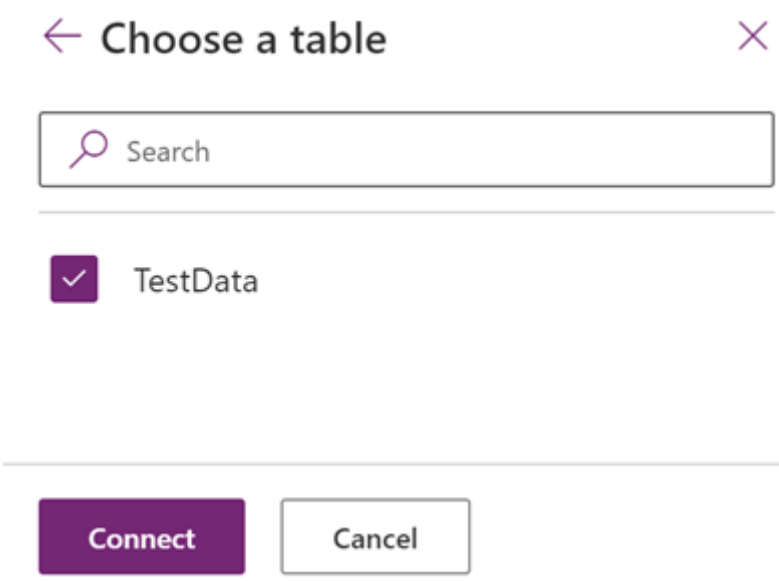
4. Save the workbook.

5. Open or create a new app in Power Apps, and insert the map control.

6. On the **Properties** pane, select the **Locations(Items)** field and then search for *excel* and select **Import from Excel**.



7. Locate the Excel workbook and then select **Open**. Select the table that contains the information, **TestData**, and then **Connect**.

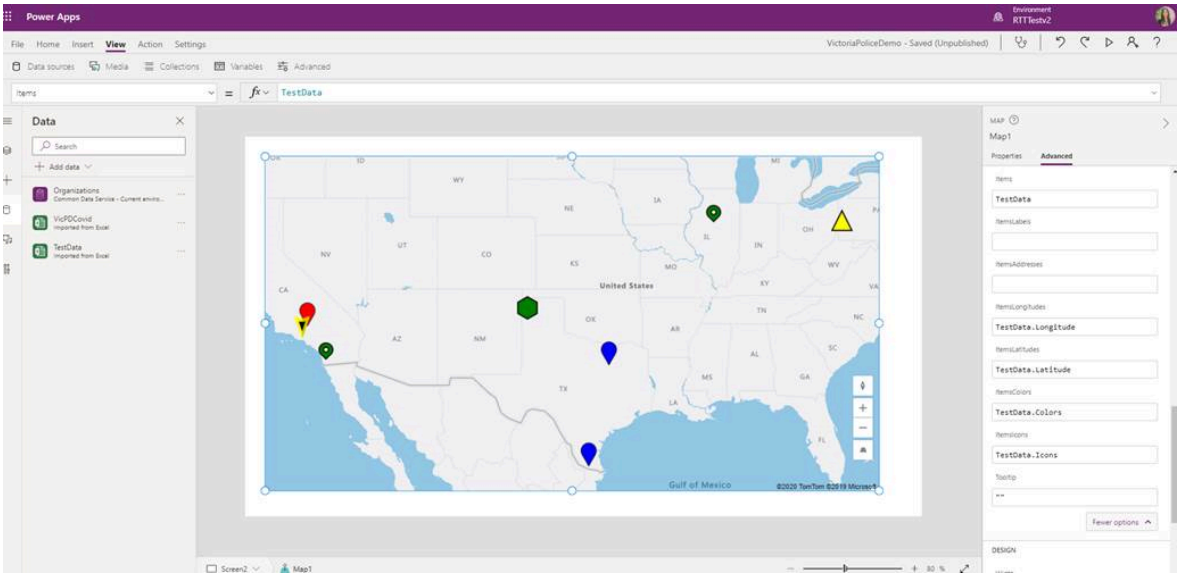


8. On the **Properties** pane, go to the **Advanced** tab, and select **More options**.

9. Set the following properties:

   - **ItemsLabels** as "*Name*"
   - **ItemsLatitudes** as "*Latitude*"
   - **ItemsLongitudes** as "*Longitude*"
   - (Optional) **ItemsColors** as "*Colors*"
   - (Optional) **ItemsIcons** as "*Icons*"

10. The map control will now show each row in the table as a pin, labeled with its *Name* as defined in the Excel table, and using the provided icons and colors. If an icon or color isn't provided, then the control will use the default icon and color.



> ⓘ **Note**

Each map control can display up to 5000 pins from latitude or longitude and 50 pins from addresses. The pin limit is lower for addresses as the map needs to geocode these addresses into latitude or longitude to display them. We recommend not using addresses where possible. You can save geocoded addresses back to your data source.

When both latitude or longitude and an address is given for a single pin, the map control will prioritize using the latitude or longitude to avoid geocoding the address.

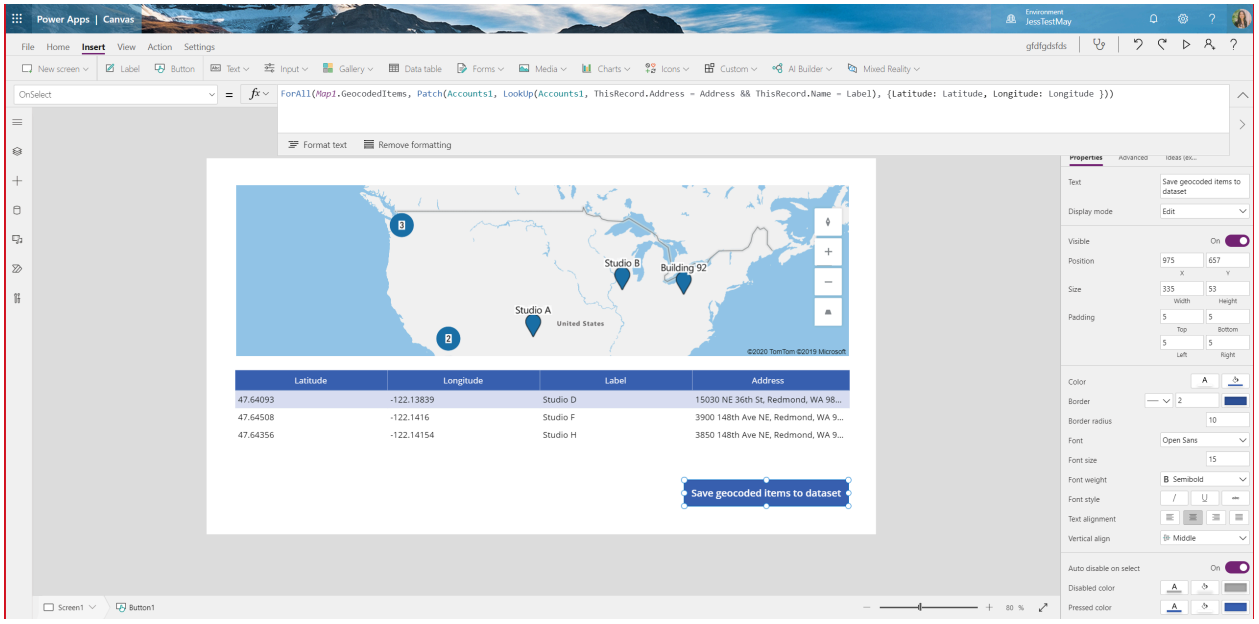
Save geocoded addresses from map control to data source

To avoid geocoding the same set of addresses every time the map control is launched, you can save the geocoded addresses back to the original data source, such that when the map relaunches, it will use the latitude and longitude instead of geocoding the addresses again.

To do so, add a button to save the geocoded address back to the original data source with the following formula:

Power Fx

```
 ForAll(Map1.GeocodedItems, Patch(Accounts1, LookUp(Accounts1,
ThisRecord.Address = Address && ThisRecord.Name = Label), {Latitude:
Latitude, Longitude: Longitude }))
```



Other interactive map features

- Add info cards to pins
- Show routes between waypoints
- Draw and insert shapes onto maps

Other geospatial controls

To see dynamic address suggestions as you type, use the **Address input** control.

Feedback

Was this page helpful? Yes No

Provide product feedback

Add informational cards to pins on a map

Article • 12/16/2022

Easily add cards to maps in your canvas apps to show details about pinned locations. When a user selects a pin, a card pops up with information from the data source you bound to the map control. Any column in the data source (for example, an Excel table) can be shown as a field on the card.

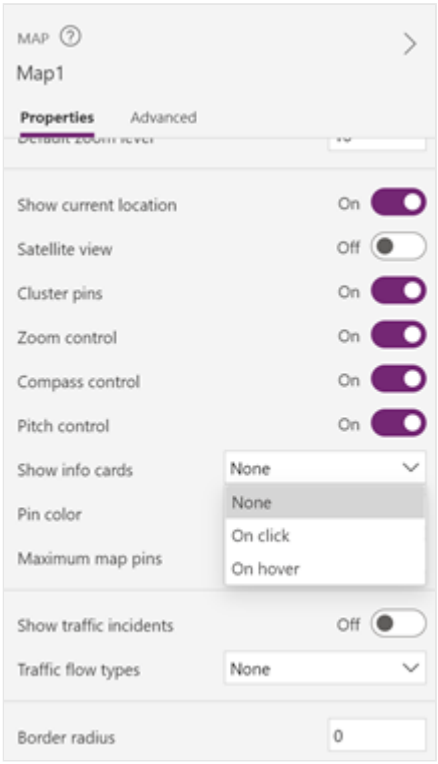
Prerequisites

1. Create a canvas app and make sure it meets the geospatial prerequisites.
2. Insert a map.

Add info cards to pins

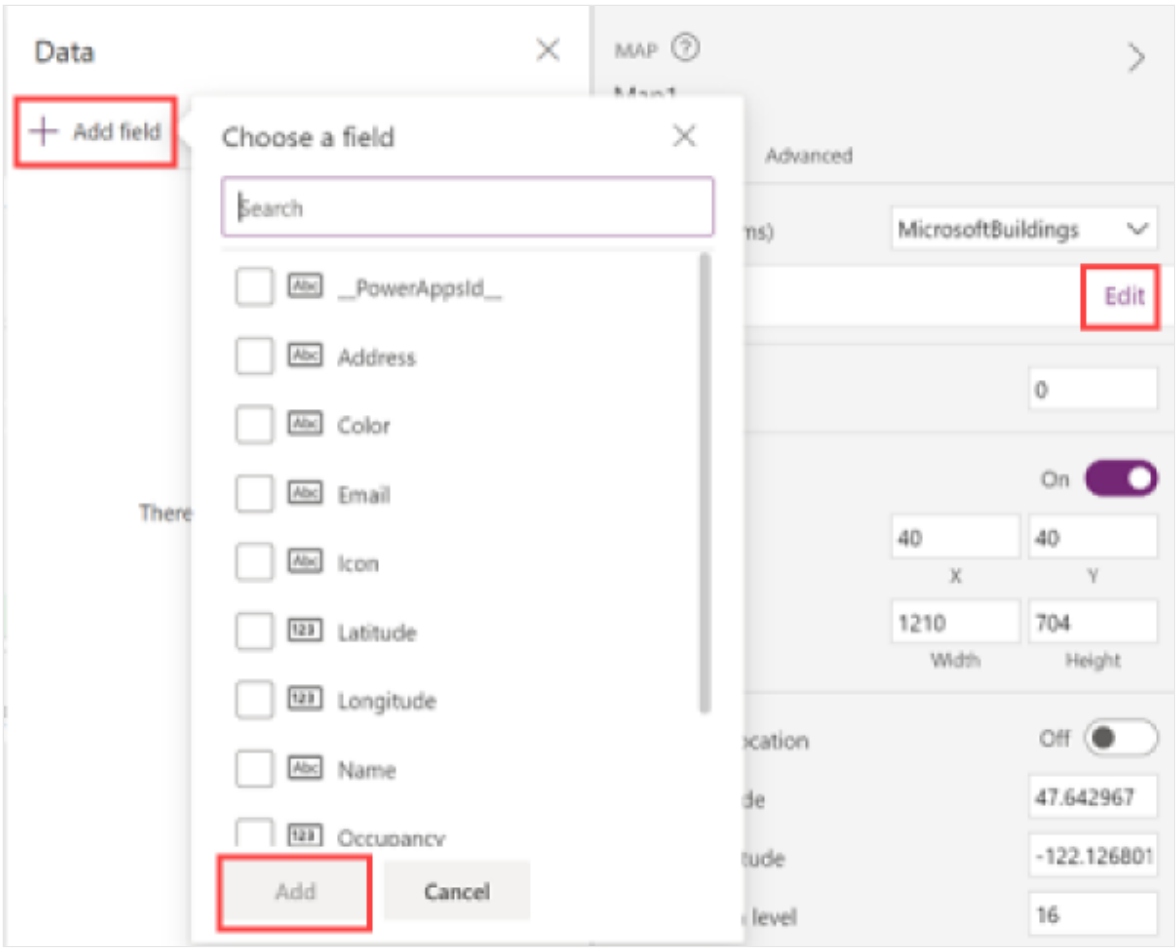
1. Bind the map to a data source.

2. In the map control's **Properties** pane, select **Show info cards**. Select **On click** to show the info card when the user selects a pin, or **On hover** if it should appear when the user hovers over the pin.



3. In the **Properties** pane, next to **Fields**, select **Edit**.

4. Select **Add field**, and then select the fields from the data source that you want to show in the card. Select **Add**.



When the user selects a pin, an info card shows the fields you selected.



Other interactive map features

- Use data from Excel to insert pins
- Show routes between waypoints
- Draw and insert shapes onto maps

Other geospatial controls

To see dynamic address suggestions as you type, use the **Address input** control.

Draw and insert shapes on a map (preview)

Article • 12/16/2022

[This topic is pre-release documentation and is subject to change.]

Add a drawing panel to your maps to let users draw and label shapes on them. You can also insert and label predefined shapes to highlight specific regions or areas of interest.



Important

- This is a preview feature.
- Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

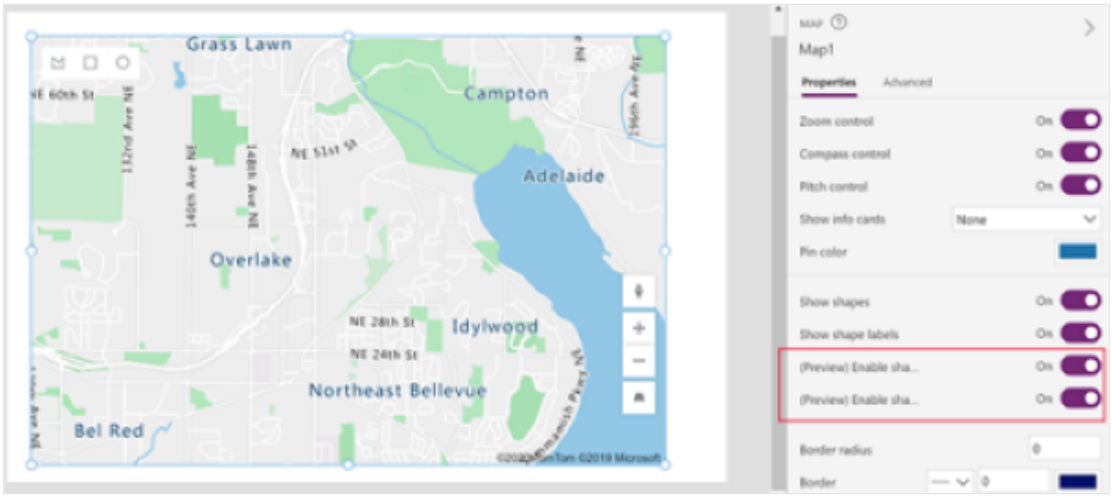
Prerequisites

1. Create a canvas app and make sure it meets the geospatial prerequisites.
2. Insert a map.

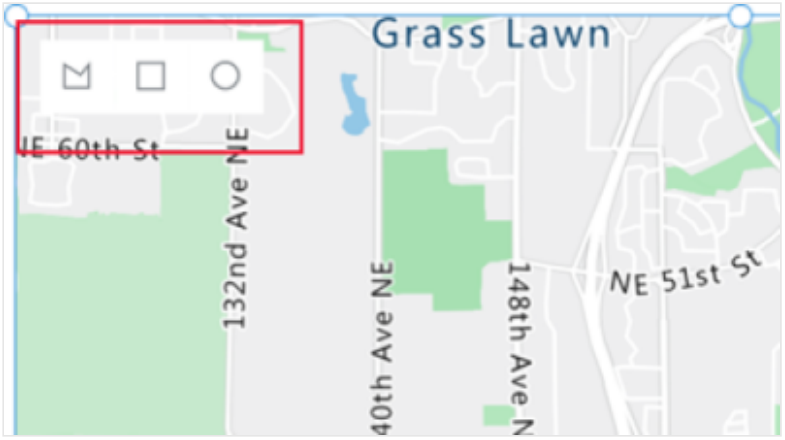
Draw and label shapes on an interactive map

1. In the map control's **Properties** pane, turn on the following properties:

    - **(Preview) Enable Shape Drawing**
    - **(Preview) Enable Shape Deleting and Label Editing**



A drawing panel appears in the upper-left corner of the map control.



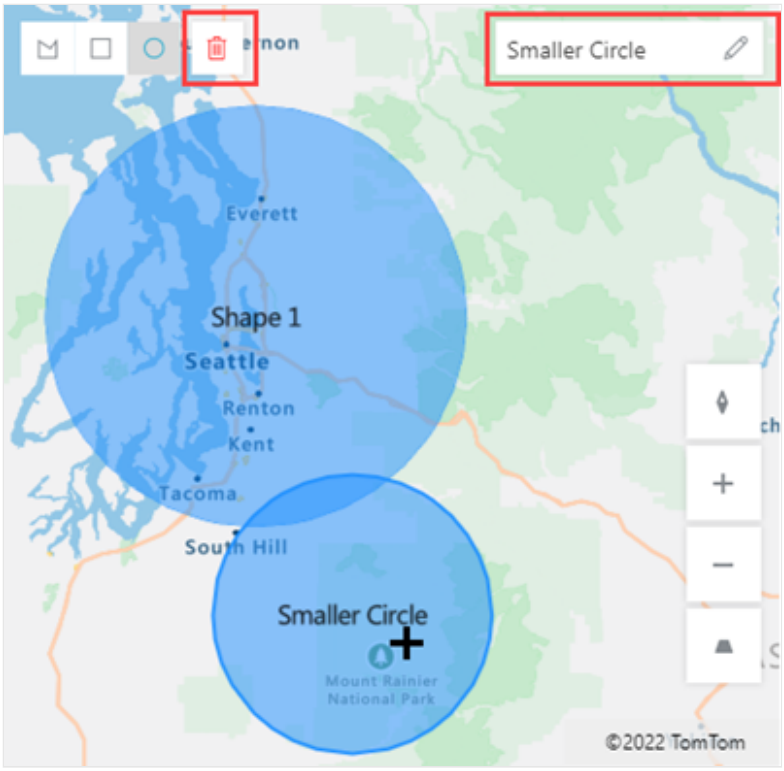
The panel features three tools:

- A polygon drawing tool for drawing freeform shapes
- A square drawing tool for drawing squares and rectangles
- A circle drawing tool for drawing circles and ovals

2. Select a tool and draw a shape on the map. To finish and close a freeform shape, double-click the final point.

Drawn shapes are automatically labeled. To change a label, select the shape, and then enter a new name in the box in the upper-right corner of the map.

To delete a shape, select it, and then select the trashcan.

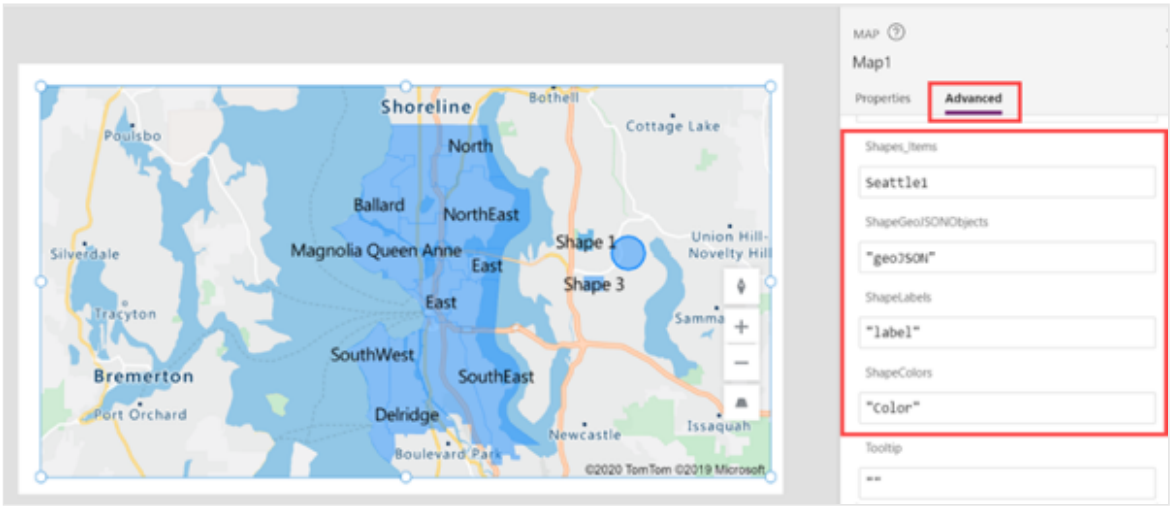


Import GeoJSON shapes into a map

If you already have shapes defined in GeoJSON format, you can import them into your map. For example, you might have a shape that defines a specific area that you want to highlight.

The GeoJSON string for each shape in your data source needs to be in the correct format, in a single cell.

To import shapes, first connect your data source to the map control. Then, assign the control's properties to the appropriate columns in the data source, as in the following example:

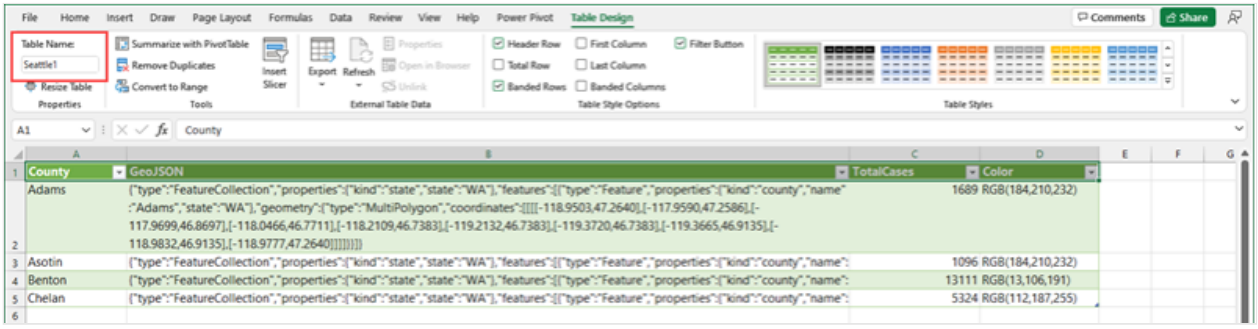


Your data source must be a named table with at least three columns. Each column corresponds to an advanced property of the map control.

Column description	Corresponds to	Required
The shape's GeoJSON coordinates in the correct format	ShapeGeoJSONObjects	Required
Shape label	ShapeLabels	Optional
Shape color	ShapeColors	Optional

> **Tip**
>
> Specify the color using a **name, CSS color definition, or RGBA value**. If the color isn't provided, then the shape will use the app's defaults.

The data source must be named. If you're using an Excel workbook as the data source, use the name of the table that contains the data. In this example, we'll import shapes from an Excel table named *Seattle1*.



1. Copy the following sample data into a blank workbook, format it as a table, and name the table *Seattle1*. Then, connect the workbook to your map as a data source. For detailed instructions using a similar example, see Use a data source to place pins on a map.

County	GeoJSON	TotalCases	Color
Adams	{"type":"FeatureCollection","properties": {"kind":"state","state":"WA"},"features": [{"type":"Feature","properties": {"kind":"county","name":"Adams","state":"WA"},"geometry": {"type":"MultiPolygon","coordinates":[[[[-118.9503,47.2640], [-117.9590,47.2586],[-117.9699,46.8697], [-118.0466,46.7711],[-118.2109,46.7383], [-119.2132,46.7383],[-119.3720,46.7383], [-119.3665,46.9135],[-118.9832,46.9135], [-118.9777,47.2640]]]]}]}	1689	RGB(184,210,232)

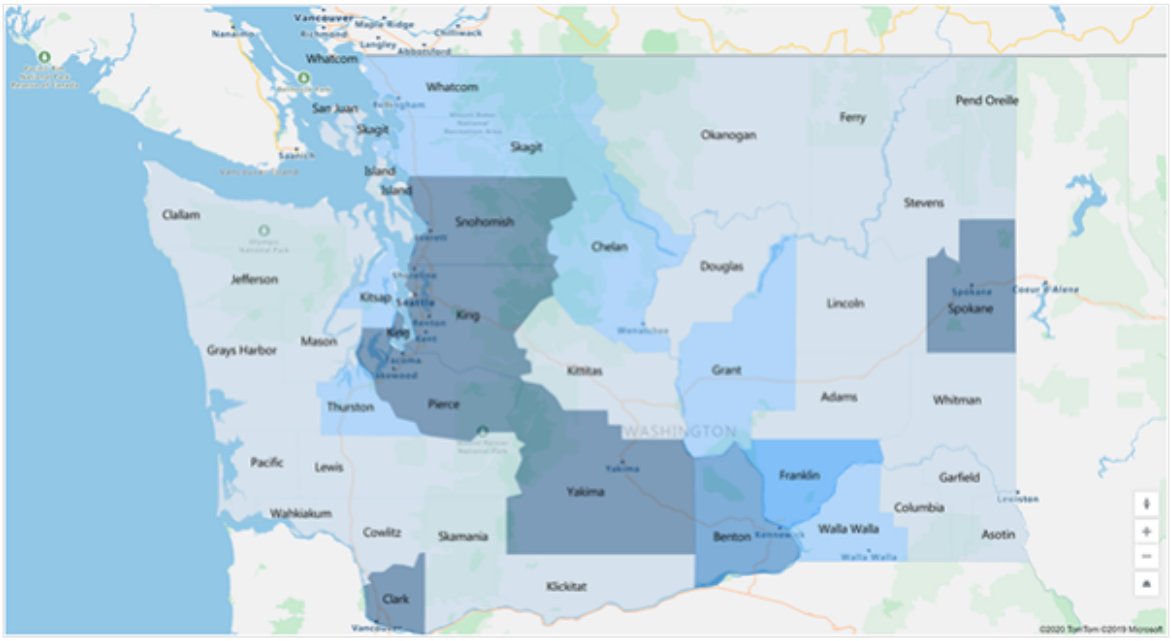
County	GeoJSON	TotalCases	Color
Asotin	{"type":"FeatureCollection","properties": {"kind":"state","state":"WA"},"features": [{"type":"Feature","properties": {"kind":"county","name":"Asotin","state":"WA"},"geometry": {"type":"MultiPolygon","coordinates":[[[[-117.0388,46.4261], [-117.0607,46.3549],[-116.9841,46.2946], [-116.9676,46.2015],[-116.9238,46.1687], [-116.9841,46.0920],[-116.9183,45.9934], [-117.4825,45.9989],[-117.4825,46.1194], [-117.4222,46.1194],[-117.4222,46.3823], [-117.2305,46.4096],[-117.2305,46.4644], [-117.1977,46.4206]]]]}}]}	1096	RGB(184,210,232)
Benton	{"type":"FeatureCollection","properties": {"kind":"state","state":"WA"},"features": [{"type":"Feature","properties": {"kind":"county","name":"Benton","state":"WA"},"geometry": {"type":"MultiPolygon","coordinates":[[[[-119.8759,46.6287], [-119.6240,46.6452],[-119.5144,46.7273], [-119.4542,46.6780],[-119.2680,46.5192], [-119.2680,46.2727],[-119.0434,46.1906], [-118.9448,46.0756],[-118.9393,46.0263], [-118.9886,45.9989],[-119.1256,45.9332], [-119.4323,45.9167],[-119.5692,45.9277], [-119.6678,45.8565],[-119.8704,45.8346], [-119.8649,46.0427],[-119.8759,46.6287]]]]}}]}	13111	RGB(13,106,191)

County	GeoJSON	TotalCases	Color
Chelan	{"type":"FeatureCollection","properties": {"kind":"state","state":"WA"},"features": [{"type":"Feature","properties": {"kind":"county","name":"Chelan","state":"WA"},"geometry": {"type":"MultiPolygon","coordinates":[[[[-120.7029,48.5292], [-120.6536,48.5347],[-120.6262,48.4964], [-120.6646,48.4471],[-120.6481,48.3978], [-120.5605,48.3704],[-120.5879,48.3211], [-120.5112,48.3101],[-120.3524,48.2170], [-120.3633,48.1568],[-120.3250,48.1294], [-120.1443,48.0637],[-120.1443,48.0363], [-120.0895,48.0199],[-120.0512,47.9596], [-119.8704,47.9596],[-119.9964,47.7789], [-120.2100,47.7515],[-120.1990,47.6803], [-120.2374,47.5872],[-120.3031,47.5215], [-120.3195,47.4557],[-120.2921,47.4010], [-120.0895,47.3407],[-120.0950,47.2640], [-120.3907,47.2586],[-120.5276,47.3352], [-120.5605,47.3079],[-120.8015,47.4229], [-120.9165,47.4284],[-121.1137,47.5981], [-121.1301,47.6748],[-121.0644,47.7132], [-121.1192,47.7789],[-121.0699,47.8282], [-121.1520,47.8446],[-121.1739,47.8884], [-121.1630,47.9541],[-121.1192,47.9980], [-121.1520,48.0418],[-121.0151,48.0746], [-120.9439,48.1130],[-120.9548,48.1513], [-120.9056,48.1623],[-121.0041,48.2937], [-121.0699,48.3156],[-121.0425,48.3485], [-121.0644,48.3923],[-121.0370,48.4306], [-121.0480,48.4854],[-120.8563,48.5511], [-120.7851,48.5073]]]]}}]}	5324	RGB(112,187,255)

2. Bind the following advanced properties of the map control to the appropriate columns in the data source:

Property	Column
ShapeGeoJSONObjects	GeoJSON
ShapeLabels	County
ShapeColors	Color

Your map should look something like this:

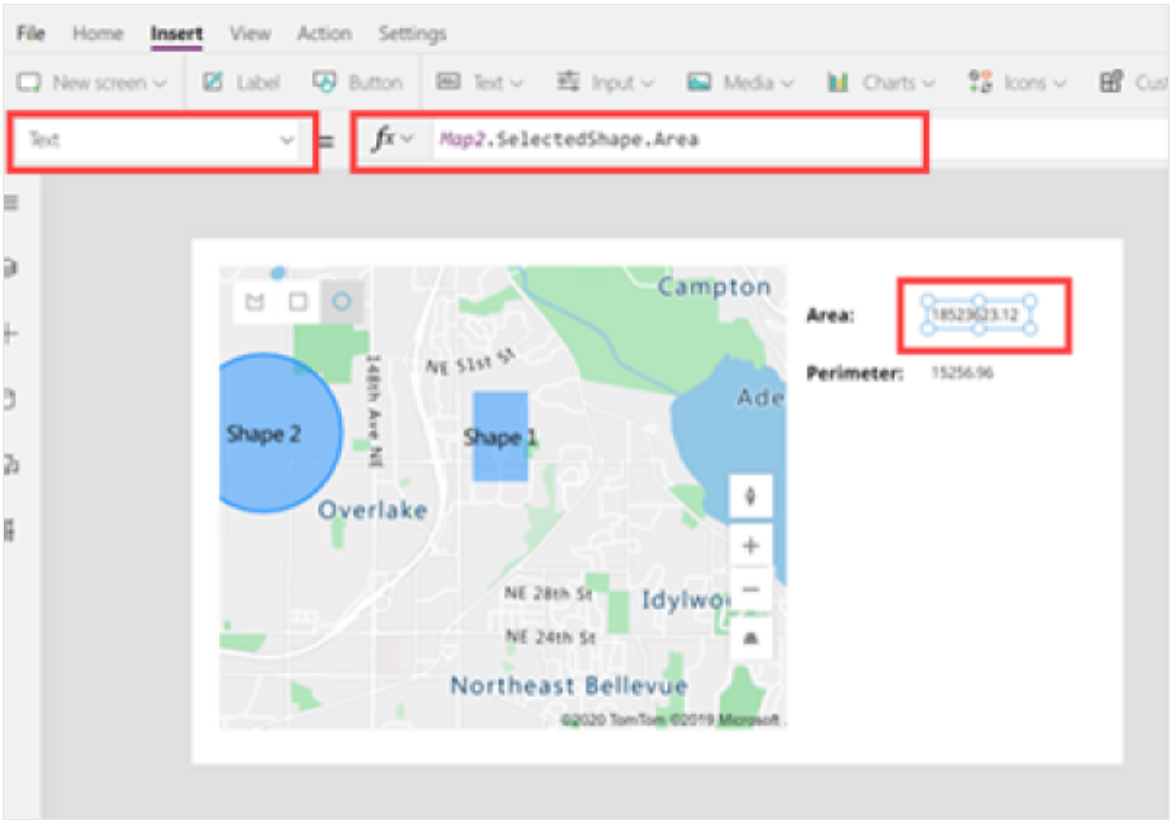


Display the perimeter and area of drawn or inserted shapes (preview)

Your apps can show the perimeter (in feet) and area (in square feet) of a selected shape on a map.

1. In your app, open the **Insert** tab and expand **Display**.
2. Select **Text label** to insert a label on the app screen. Move and resize it as you like.
3. Change the **Text** property of the label control to *[Map].SelectedShape.Area* or *[Map].SelectedShape.Perimeter* to display the selected shape's area or perimeter. (Hint: Enter the given value in the formula bar or on the **Properties** tab, whichever you prefer.)

In the following example, we used four text labels to display both the area and perimeter of the selected shape:



Other interactive map features

- Use data from Excel to insert pins
- Add info cards to pins
- Show routes between waypoints

Other geospatial controls

To see dynamic address suggestions as you type, use the **Address input** control.

Calculate routes between waypoint pins on a map (preview)

Article • 07/15/2022

[This topic is pre-release documentation and is subject to change.]

The map control in a canvas app can add waypoint pins using data imported from a table in an Excel workbook. The control can calculate routes between waypoints, reorder waypoints, and recalculate a route to improve travel time or distance.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

You'll need a data source that contains a named table with the following columns. Each column corresponds to an advanced property of the map control. Each row is pinned as a waypoint on the map.

Column name	Corresponds to	Required
Name (or Label)	RouteWaypointsLabels	Optional
Longitude	RouteWaypointsLongitudes	Optional (required if Address isn't given)
Latitude	RouteWaypointsLatitudes	Optional (required if Address isn't given)
Address	RouteWaypointsAddresses	Optional (required if Longitude and Latitude aren't given)

> **Note**
>
> All properties are technically optional. However, at least one of an address or a latitude/longitude pair must be provided for the waypoint location to be plotted.

Import waypoint pins from an Excel table

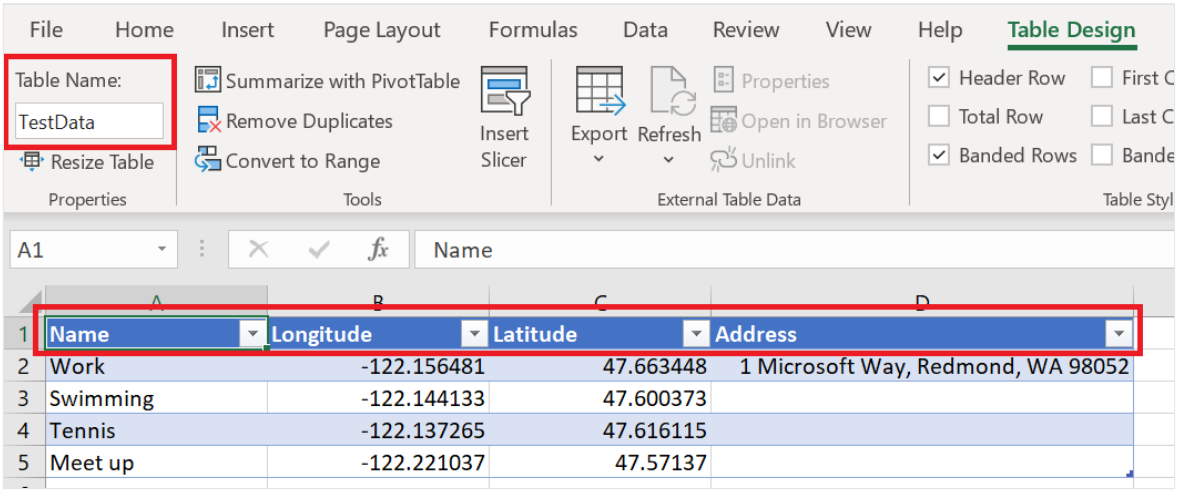
In this example, we'll import waypoint data from an Excel table named *TestData*.

Create a data source

1. Create a table in Excel with the following data. Name the table **TestData**.

Name	Longitude	Latitude	Address
Work	-122.156481	47.663448	1 Microsoft Way, Redmond, WA 98052
Meet up	-122.221037	47.57137	
Swimming	-122.144133	47.600373	
Tennis	-122.137265	47.616115	

   Your table should look something like this:



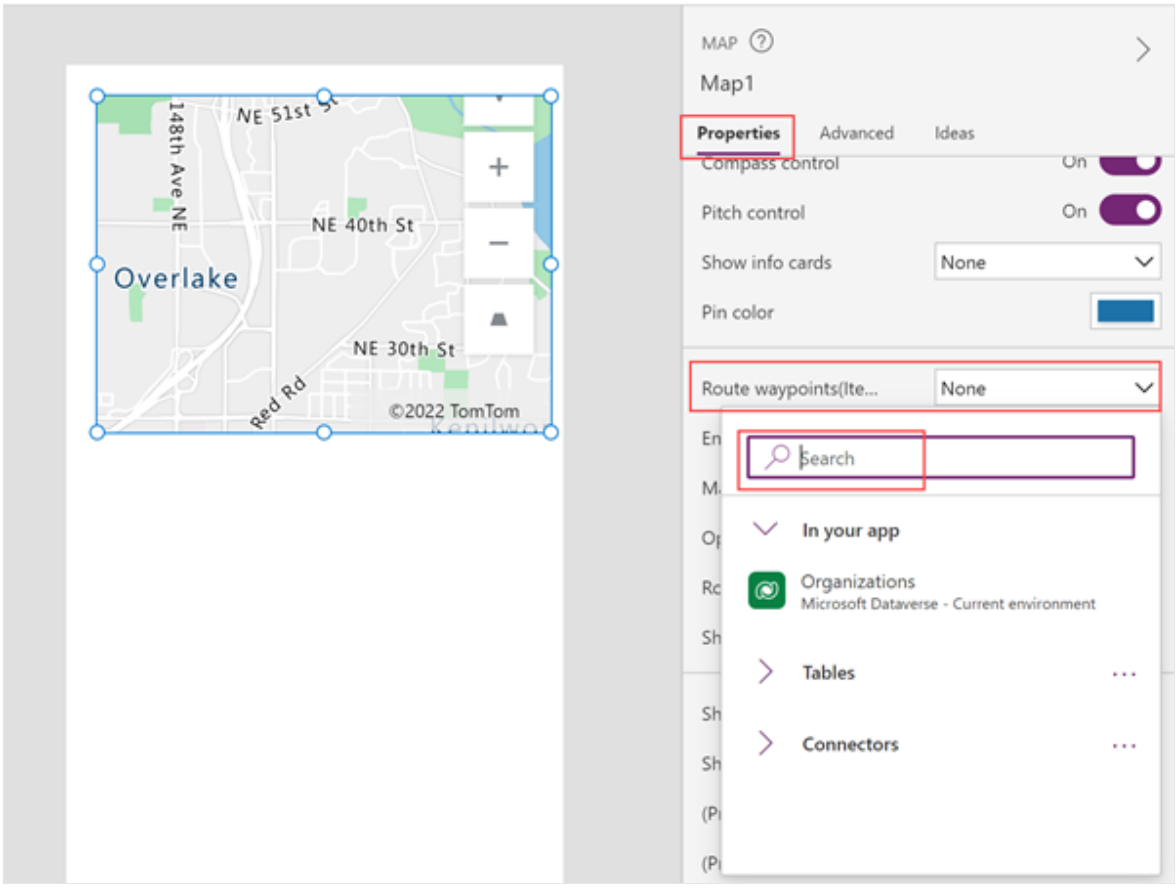
2. Save the workbook to your OneDrive for Business and close the file.

Bind the data source to a map control

1. Create a canvas app. Make sure it meets the geospatial prerequisites.

2. Insert a map control.

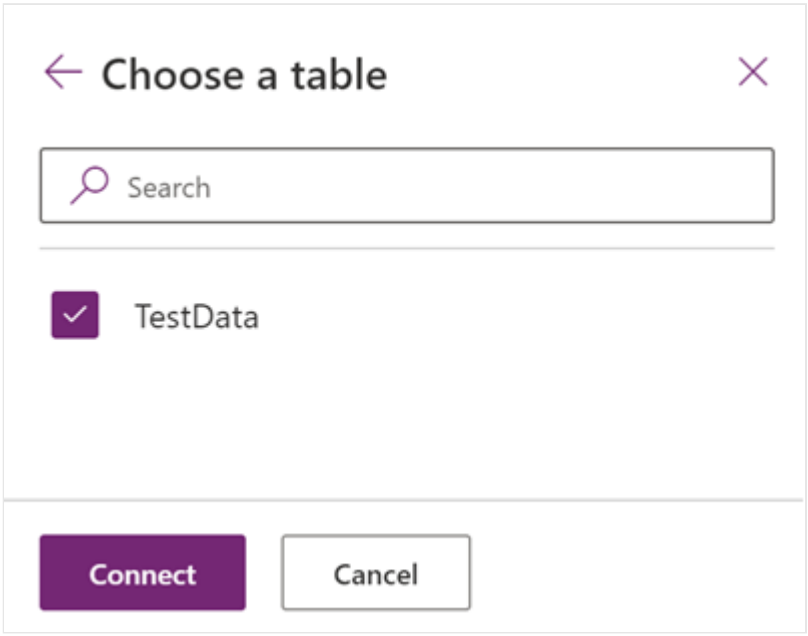
3. On the control's **Properties** tab, select the **Route waypoints(Items)** box and type *excel*.



4. Select **Import from Excel**.

5. Navigate to your OneDrive for Business and select the Excel workbook you saved earlier.

6. Select the table **TestData**, and then select **Connect**.

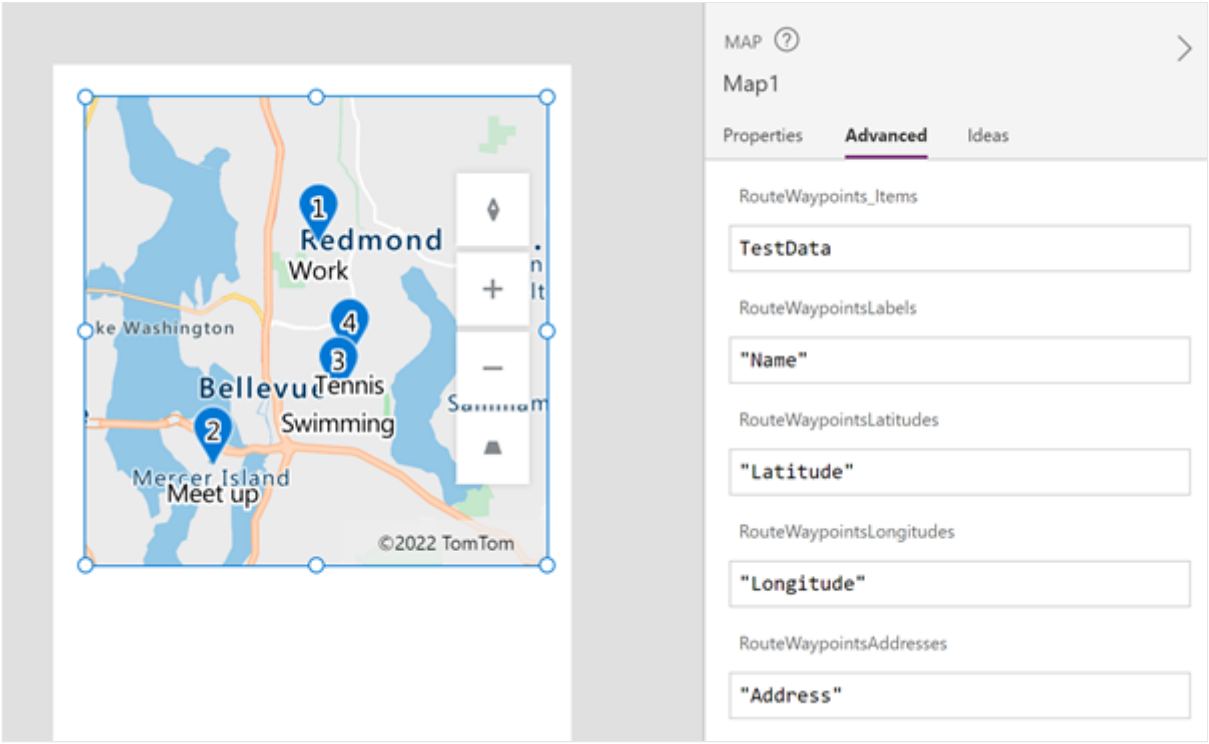


7. On the **Advanced** tab, find **RouteWaypointsLabels**, **RouteWaypointsLatitudes**, **RouteWaypointsLongitudes**, and **RouteWaypointsAddresses** and enter the name of the corresponding column in the table. (In this example, enter *Name* in

RouteWaypointsLabels, *Latitude* in **RouteWaypointsLatitudes**, and so on.) Enclose the column name in quotation marks.

> **Note**
>
> Address is interchangeable with Latitude and Longitude. If Latitude and Longitude are provided, then Address isn't used. If Address is provided, then Latitude and Longitude aren't needed. There is a limit to the number of waypoints that can be displayed when using address, so use Latitude and Longitude when possible.

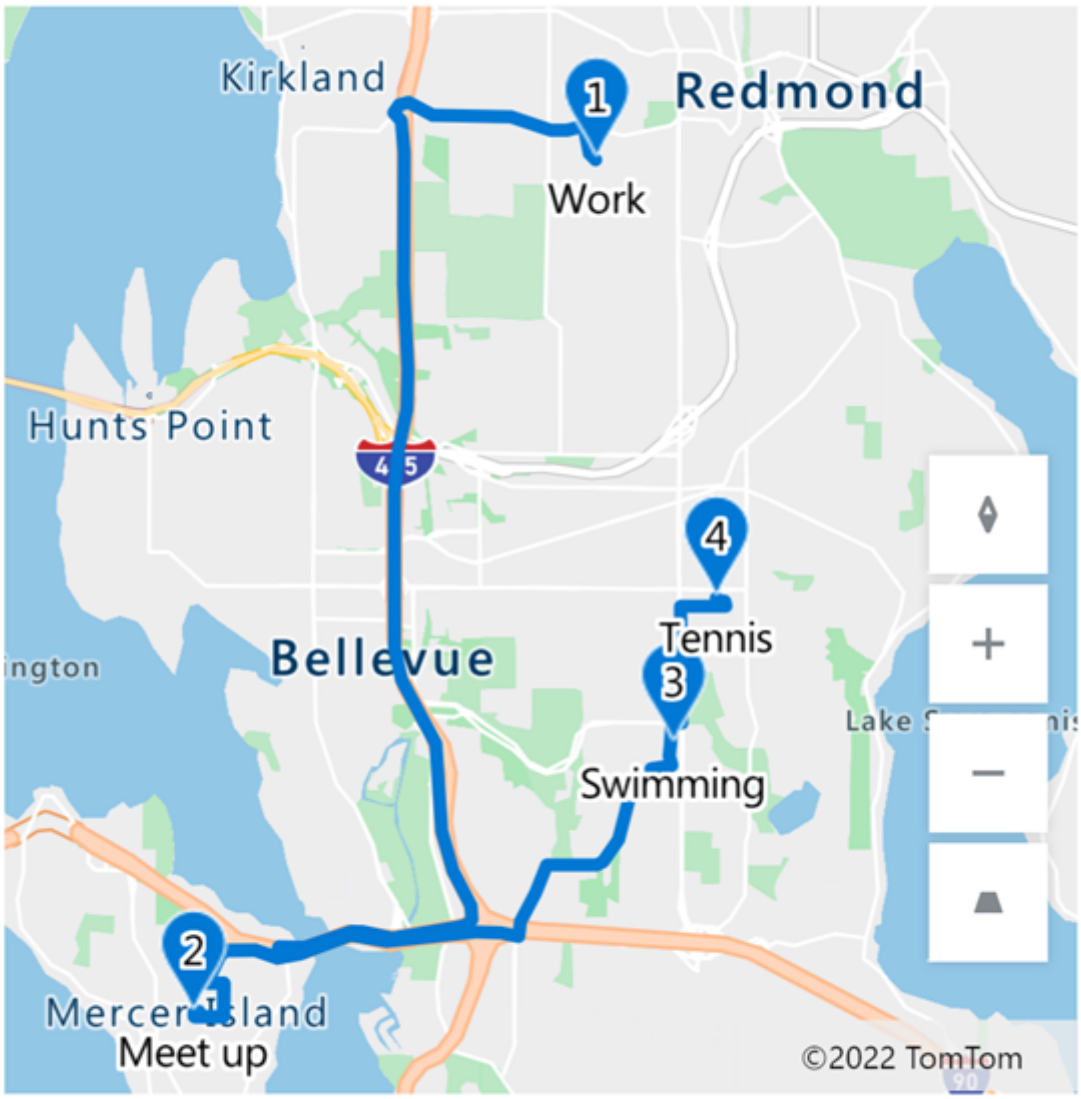
Pins appear on the map at the locations described by the coordinates or addresses in the table. If the table included labels, the pins are labeled. The pins are numbered in the order the waypoint locations appear in the table.



Calculate routes between the waypoints

With the map control selected, open the **Properties** tab and turn on **Enable routing**.

The control calculates routes between the pinned waypoints.



> **Note**
>
> By default, the map control reorders the middle waypoints to decrease travel time or travel distance. The first and last waypoints are considered the origin and destination and can't be reordered. To keep the waypoints in the order given in the data source, turn on the control's **Maintain waypoint order** property.

Properties

Change how a route is calculated using properties.

Property	Description	Type	Tab

Property	Description	Type	Tab
Enable routing	Calculates routes between waypoints.	Boolean	Properties; Advanced: **UseRouting**
Maintain waypoint order	Determines whether a calculated route maintains waypoints in the order provided in the data source.	Boolean	Properties; Advanced: **RouteMaintainOrder**
Optimize route	Determines whether a calculated route is optimized for distance, time, or isn't optimized.	Dropdown list	Properties; Advanced: **RouteOptimization**
Route travel mode	Determines whether a route is calculated for a car or a truck, which may require avoiding bridges with certain height or weight restrictions.	Dropdown list	Properties; Advanced: **RouteTravelMode**
Show route pins	Determines whether pins are shown over the route waypoints.	Boolean	Properties; Advanced: **ShowRoutePins**

Output Properties

Some properties become available only when a calculated route changes. These *output properties* are placed in the **RouteDirection** object. The **OnRouteDirectionChange** event is a recommended way to use the output in other controls or to customize the app experience.

Property	Description	Type	Tab
OnRouteDirectionChange	Contains code that runs when the route is changed.	Event	Advanced

Property	Description	Type	Tab
RouteDirection	Describes the routing outputs: • LengthInMeters: The length in meters of the entire route • TravelTimeInSeconds: The expected travel time in seconds of the entire route • RouteGeoJSON: A string that describes the route in GeoJSON format • RouteLegs: A table that describes each leg of the route: ◦ Index: A number that represents the leg's order in the route ◦ LengthInMeters: The length of the leg in meters ◦ TravelTimeInSeconds: The expected travel time of the leg in seconds ◦ StartLabel: The label of the leg's starting point ◦ StartLatitude: The latitude of the leg's starting point ◦ StartLongitude: The longitude of the leg's starting point ◦ StartAddress: The address of the leg's starting point ◦ EndLabel: The label of the leg's ending point ◦ EndLatitude: The latitude of the leg's ending point ◦ EndLongitude: The longitude of the leg's ending point ◦ EndAddress: The address of the leg's ending point • OrderedWaypoints: A table that describes each waypoint after the waypoints are ordered: ◦ Index: A number that represents the waypoint's order in the route ◦ Label: The label of the waypoint ◦ Latitude: The latitude of the waypoint ◦ Longitude: The longitude of the waypoint ◦ Address: The address of the waypoint	Object	Not applicable (output property only)

See also

Interactive map control

Use the address input control to easily enter addresses

Article • 07/15/2022

Entering addresses can be frustrating and error-prone, particularly in mobile apps. Use the address input control to make address entry easier. The control uses fuzzy logic to suggest potential matches as you type. Select the one you want to quickly and easily enter an accurate address.

The control returns the address as structured data. Your app can extract information such as city, street, municipality, and even latitude and longitude coordinates. The data is in a format that's friendly to many locales and international address formats.

Prerequisites

Before you can use the control in your apps, you'll need to enable geospatial features for the environment. Make sure you also review the prerequisites for using geospatial controls. Refer to the privacy and security table for more details on the address input control's data usage.

Add an address input control to your app

With your app open for editing in Power Apps Studio :

1. Open the **Insert** tab and expand **Input**.
2. Select **Address input** to place an address input box in the app screen, or drag the control to the screen to position it more precisely. You must enter at least three characters including one number for the address input control.

Set a default search radius

By default, the control will search around the user's location. You can refine the default search area to help narrow the initial results.

1. On the address input control's **Properties** tab, turn on the **Search within radius** property.
2. Enter a longitude, latitude, and radius in meters.

The control will start searching at the given latitude and longitude, out to the distance specified.

Use the map control with the address input control

You can add a button to your app to save entered addresses as a data collection. Then you can retrieve the addresses and display them in the map control.

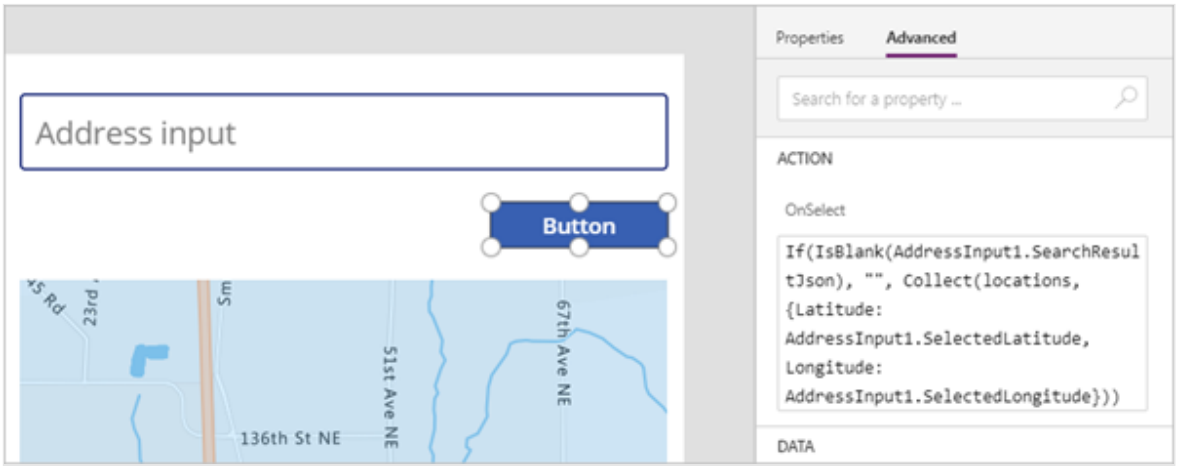
1. Add a map control and an address input control to your app.
2. Insert and place a **Button** control.
3. Change the **OnSelect** property of the button control as follows. (Hint: Copy the formula and paste it in the formula bar or on the **Advanced** properties tab, whichever you prefer.)
4. Enter a longitude, latitude, and radius (in meters).

   The control will start searching at the latitude and longitude, out to the distance specified in the radius field.

   Power Fx

   ```
   If(IsBlank(AddressInput1.SearchResultJson), "", Collect(locations,
   {Latitude: AddressInput1.SelectedLatitude, Longitude:
   AddressInput1.SelectedLongitude}))
   ```

   The formula saves the current latitude and longitude to a collection named *locations*, as long as the search results aren't blank.

5. Select the map control. Change its properties as follows:

Expand table

Property name	Value	Where
Items	"Locations"	**Properties** tab
ItemsLatitudes	"Latitude"	**Advanced** tab
ItemsLongitudes	"Longitude"	**Advanced** tab

When the user selects the button, the result from the address input control is added to the map as a new pin.

Properties

Change an address input control's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.

Expand table

Property	Description	Type	Tab
Default	Sets the initial value of the control.	String	Properties; Advanced: **Default**
Hint text	Sets the hint that appears in the control before the user enters text.	String	Properties; Advanced: **HintText**
Font	Sets the name of the family of fonts used for the control text.	Dropdown list	Properties; Advanced: **Font**
Font size	Sets the size of the control text.	Floating point number	Properties; Advanced: **FontSize**
Font weight	Sets the weight of the control text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	Dropdown list	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the control text, either *Center*, *Justify*, *Left*, or *Right*.	Dropdown list	Properties; Advanced: **TextAlignment**
Line height	Sets the vertical distance between lines of text in the control.	Floating point number	Properties; Advanced: **LineHeight**

Property	Description	Type	Tab
Display mode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Dropdown list	Properties; Advanced: DisplayMode
Font style	Sets the style of the control text, either *Italic*, *Underline*, *Strikethrough*, or none.	Dropdown list	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Search result limit	Sets the number of suggested addresses the control displays.	Integer	Properties; Advanced: **SearchResultLimit**
Search within radius	Determines whether the control should suggest addresses within the **Radius** of the **Latitude** and **Longitude**.	Boolean	Properties; Advanced: **SearchWithinRadius**
Latitude	Sets the latitude coordinate of the center point used for address suggestions. Requires **Search within radius** to be on.	Floating point number from -90 to 90	Properties; Advanced: **Latitude**
Longitude	Sets the longitude coordinate of the center point used for address suggestions. Requires **Search within radius** to be on.	Floating point number from -180 to 180	Properties; Advanced: **Longitude**
Radius	Sets the radius, in meters, around **Latitude** and **Longitude** to constrain address suggestions. Requires **Search within radius** to be on.	Floating point number	Properties; Advanced: **Radius**
Language	Sets the language that address suggestions are returned in.	String	Properties; Advanced: **Language**
Country set	Identifies a comma-separated list of countries/regions to constrain address suggestions to, in ISO 3166 alpha-2 format; for example, *US,CA,MX*.	String	Properties; Advanced: **CountrySet**
Visible	Shows or hides the control.	Boolean	Properties; Advanced: Visible
Padding top	Sets the distance between the control text and the top of the	Floating point number	Properties; Advanced: PaddingTop

Property	Description	Type	Tab
	control.		
Padding bottom	Sets the distance between the control text and the bottom of the control.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the control text and the left edge of the control.	Floating point number	Properties; Advanced: **PaddingLeft**
Padding right	Sets the distance between the control text and the right edge of the control.	Floating point number	Properties; Advanced: **PaddingRight**
Position	Places the upper-left corner of the control at the screen coordinates specified in *X* and *Y*.	Integer	Properties; Advanced: **X**, **Y**
Size	Determines the size of the control using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Border radius	Determines the corner radius of the control border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the control text and the control background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the control border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderColor**
Tab index	Specifies the order the control is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **TabIndex**
Tooltip	Determines the text to display when the user hovers over the control.	String	Properties; Advanced: **Tooltip**
Hover color	Sets the colors of the control text, the control background, and the control border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverFontColor**, **HoverFillColor**, **HoverBorderColor**

Property	Description	Type	Tab
Disabled color	Sets the colors of the control text, the control background, and the control border if **DisplayMode** is *Disabled*.	Not applicable	Properties; Advanced: **DisabledFontColor**, **DisabledFillColor**, **DisabledBorderColor**
Pressed color	Sets the colors of the control text, the control background, and the control border when the user selects the control.	Not applicable	Properties; Advanced: **PressedFontColor**, **PressedFillColor**, **PressedBorderColor**
ContentLanguage	Determines the display language of the control, if it's different from the language that's used in the app.	String	Advanced
OnAddressSelect	Contains code that runs when the user selects a suggested address.	Event	Advanced
OnChange	Contains code that runs when a control property is changed.	Event	Advanced

Output properties

Other properties become available when a user interacts with the address input control. You can use these *output properties* in other controls or to customize the app experience.

Expand table

Property	Description
UserInput	The text the user typed in the input box
SelectedLatitude	The latitude of the address that the user selected
SelectedLongitude	The longitude of the address that the user selected
SearchResultJson	The search result, based on **UserInput**, displayed as a string in JSON format
FreeformAddress	The address that the user selected from the list of suggested addresses
LocalName	The name of a geographic area or locality that groups a number of addressable objects for addressing purposes, without being an

Property	Description
	administrative unit
PostalCode	The postal code
ExtendedPostalCode	The extended postal code
CountryCode	The country code
Country	The country/region name
CountryCodeISO3	The country code in ISO alpha-3 format
CountrySubdivisionName	The country/region subdivision name
StreetName	The street name
StreetNumber	The street number
Municipality	The municipality
MunicipalitySubdivision	The municipality subdivision
CountryTertiarySubdivision	The country/region tertiary subdivision
CountrySecondarySubdivision	The country/region secondary subdivision
CountrySubdivision	The country/region subdivision

Other geospatial controls

To visualize and interpret location data, use the **interactive map** control.

Feedback

Was this page helpful? Yes No

Provide product feedback

Power Apps visual for Power BI

Article • 03/04/2025

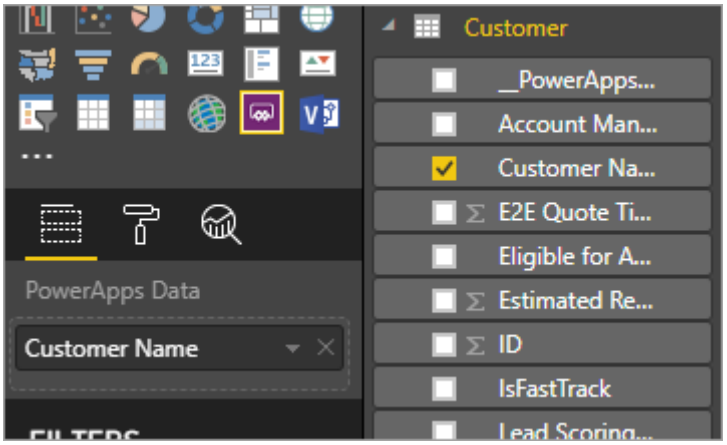
Power BI enables data insights and better decision-making, while Power Apps enables everyone to build and use apps that connect to business data. Using the Power Apps visual, you can pass context-aware data to a canvas app, which updates in real time as you make changes to your report. Now, your app users can derive business insights and take actions from right within their Power BI reports and dashboards.

Using the Power Apps visual

Let's look at the steps required to use the Power Apps visual in your Power BI report.

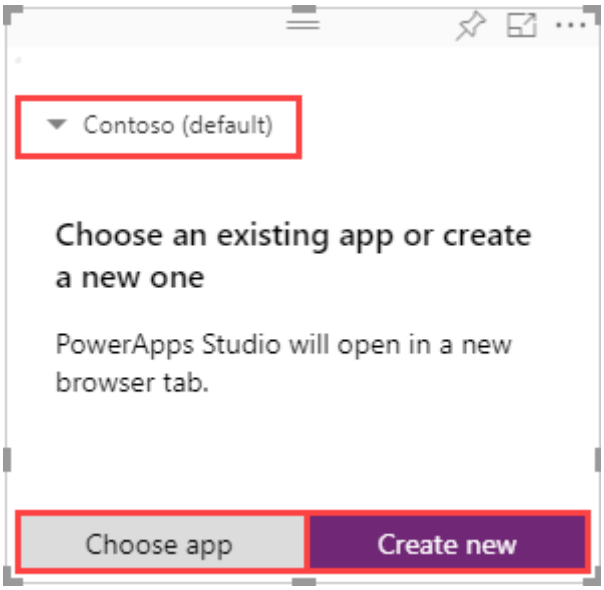
1. Power Apps visual is available by default in the Power BI service. If you're using Power BI Desktop and don't see it, you must upgrade to the latest version of Power BI Desktop.

2. Add the Power Apps visual to your report, and set the data fields associated with it.



   You can choose an existing app or create one, but the report must be published to the Power BI service and opened in Microsoft Edge or Google Chrome.

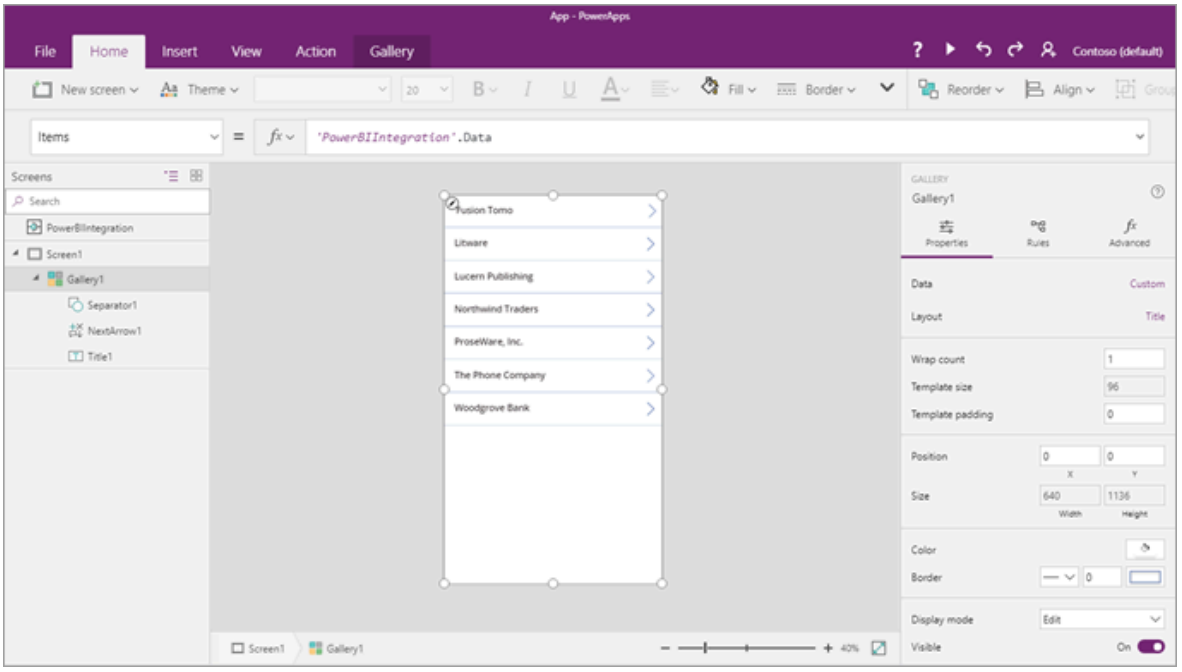
3. If you choose to create an app, you can choose in which environment to create it.



If you choose to use an existing app, the visual prompts you to open the app in Power Apps. The visual then sets up the required components in your app so that Power BI can send data to Power Apps.

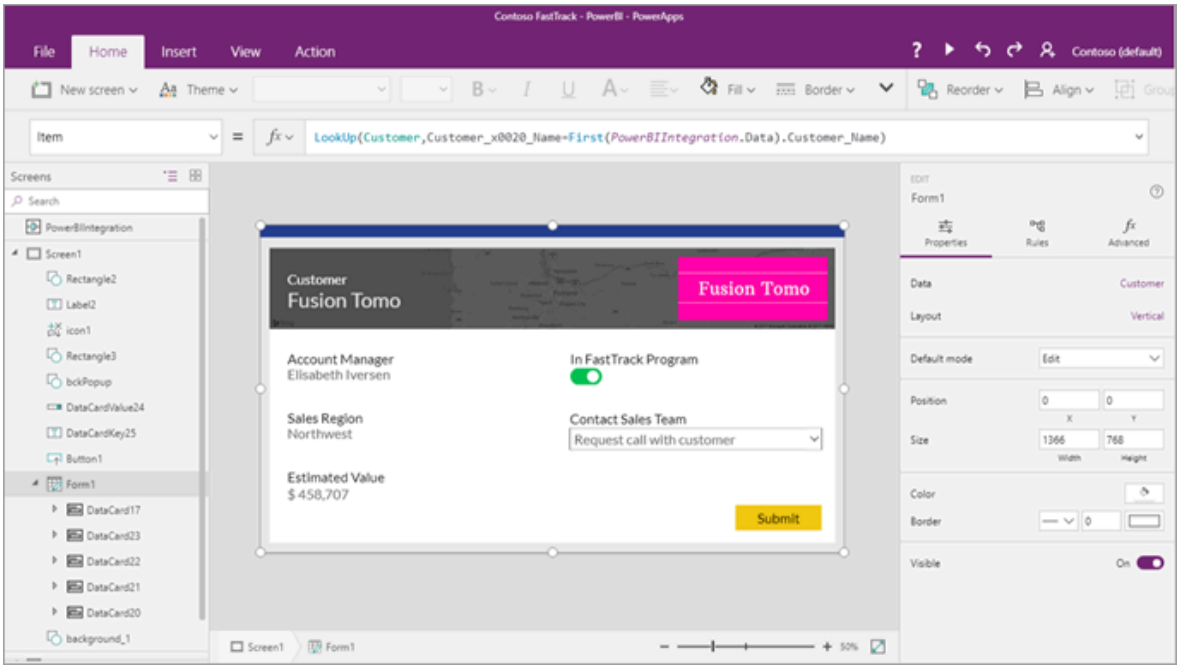
If you create a new app, Power Apps creates a simple app with the required components already set up.

> **Note**
>
> You must create a new app from Power Apps visual in Power BI report for the `PowerBIIntegration.Refresh()` function to be available in the app.



4. Now in Power Apps Studio, you can use the data fields you set in step 2. The `PowerBIIntegration` object acts like any other Power Apps read-only data source

or collection. You can use the object to populate any control, or join and filter with other data sources.



This formula joins Power BI data with the Customer data source:

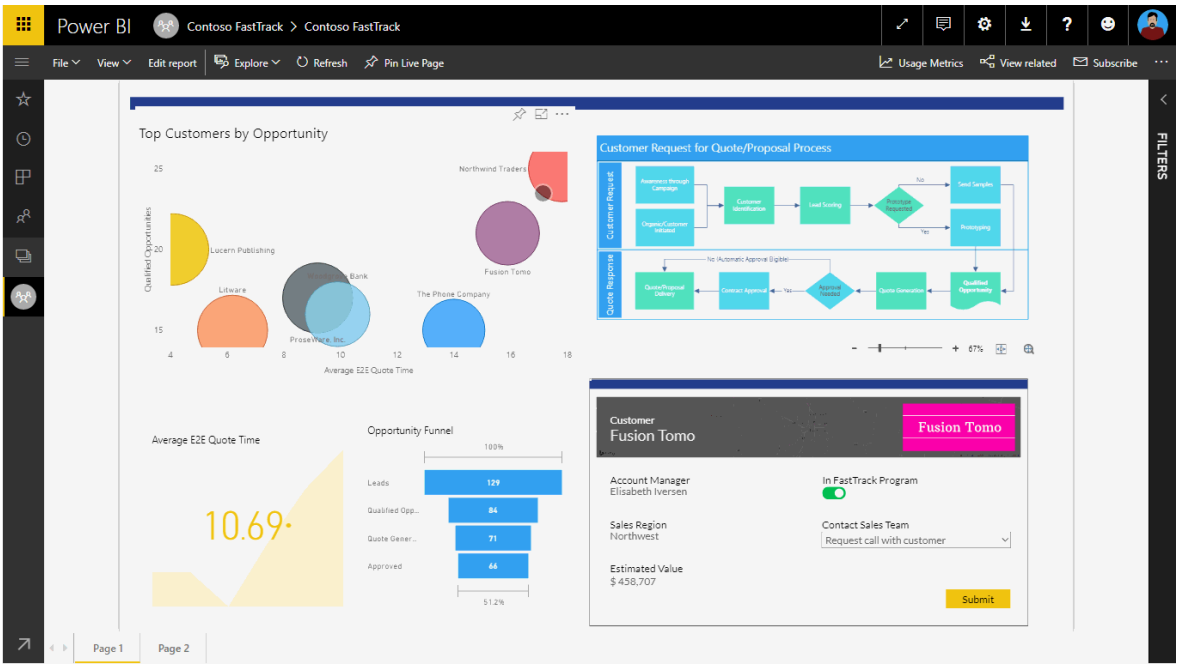
```
LookUp(Customer,Customer_x0020_Name=First(PowerBIIntegration.Data).Customer_Name)
```

The Power BI report and the instance of Power Apps Studio that was launched share a live data connection. While they're both open, you can filter or change the data in your report to see the updated data reflect immediately in your app in Power Apps Studio.

5. After you have completed building or making changes to your app, save and publish the app in Power Apps to see your app in the Power BI report.

6. Once you're satisfied with your changes, make sure to share the app with users of your report and then save your report.

7. You've created a report in which your users can take actions as they gain insights from your data.



If you need to make changes to an app, open the report in edit mode, select **More options** (. . .) on the Power Apps visual and select **Edit**.



Limitations of the Power Apps visual

The following limitations apply to the Power Apps visual:

- Only **https** is supported for the Launch function in the Power Apps Visual.
- Power Apps visual is only supported for Embed for your organization. Embed for your customers is not supported.
- The maximum number of records that can be passed from Power BI to Power Apps visual using "PowerBIIntegration" object is limited to 1000.
- Power Apps visual is only supported for guest users when the following conditions are met:
  - The canvas app URI (weblink) includes the tenantId of the app in the query string.
  - The Power BI portal authenticates the user, instead of allowing anonymous access.
  - The canvas app is shared with the guest user attempting to see the contents of the app.
- Power Apps visual doesn't support multi-level embedding of Power Apps in Sovereign clouds. For example, embedding a Power BI report with Power Apps

visual to a SharePoint page, or further embedding that SharePoint page to a Teams channel isn't supported on sovereign clouds.

- If you change the data fields associated with the visual, you must edit the app from within the Power BI service by selecting the ellipsis (...) and then selecting **Edit**. Otherwise, the changes won't be propagated to Power Apps, and the app will behave in unexpected ways.
- The Power Apps visual can't trigger a refresh of Power BI reports and Power BI data sources from within Power BI Desktop. If you write back data from the app to the same data source as the report, your changes won't be reflected immediately in Power BI Desktop. Changes are reflected on the next scheduled refresh.
- The Power Apps visual can't filter the data or send any data back to the report.
- You'll need to share the app separately from your report. Learn about sharing apps in Power Apps.
- Power BI Report Server doesn't support the Power Apps visual.
- Following limitations apply when using the `PowerBIIntegration.Refresh()` function:
  - You must create a new app from Power Apps visual in Power BI report for this function to be available in the app.
  - You must use a source that supports DirectQuery and the data connection must be created using DirectQuery method.
- Power Apps in Power BI Desktop provides data to Power Apps Studio when creating apps but not while editing. Use Power BI Web to preview the data while editing apps.
- The Power BI mobile app doesn't support microphone control in Power Apps visuals.

> **Note**
>
> We recommend that you first publish your report to the Power BI service and then create or modify apps.

Browser support

The following table lists the browser supportability for view, create, and modify actions of the Power Apps visual. Supported browsers and actions are identified by a check mark (✓).

Expand table

Browser	View	Create	Modify
Microsoft Edge	✓	✓	✓
Google Chrome	✓	✓	✓
Safari *	✓		
Mozilla Firefox			
All other browsers			

* In Safari, you must enable cross site tracking (**Preferences** > **Privacy**, and clear **Prevent cross site tracking**) to view Power Apps visual.

Accessibility support

To navigate the Power Apps visual using the keyboard, follow these steps:

1. Focus selection on the Power BI Report for the Power Apps visual that you want.
2. Use the **Tab** key on the keyboard until the visual is highlighted.
3. Use the **Ctrl+Right** key on the keyboard to enter the visual.
4. Use the **Tab** key on the keyboard until the component of the visual you want is selected.

For more information, go to Power BI Accessibility Documentation

Next steps

- Go through a simple step-by-step tutorial.
- Check out our video .

Feedback

Was this page helpful? Yes No

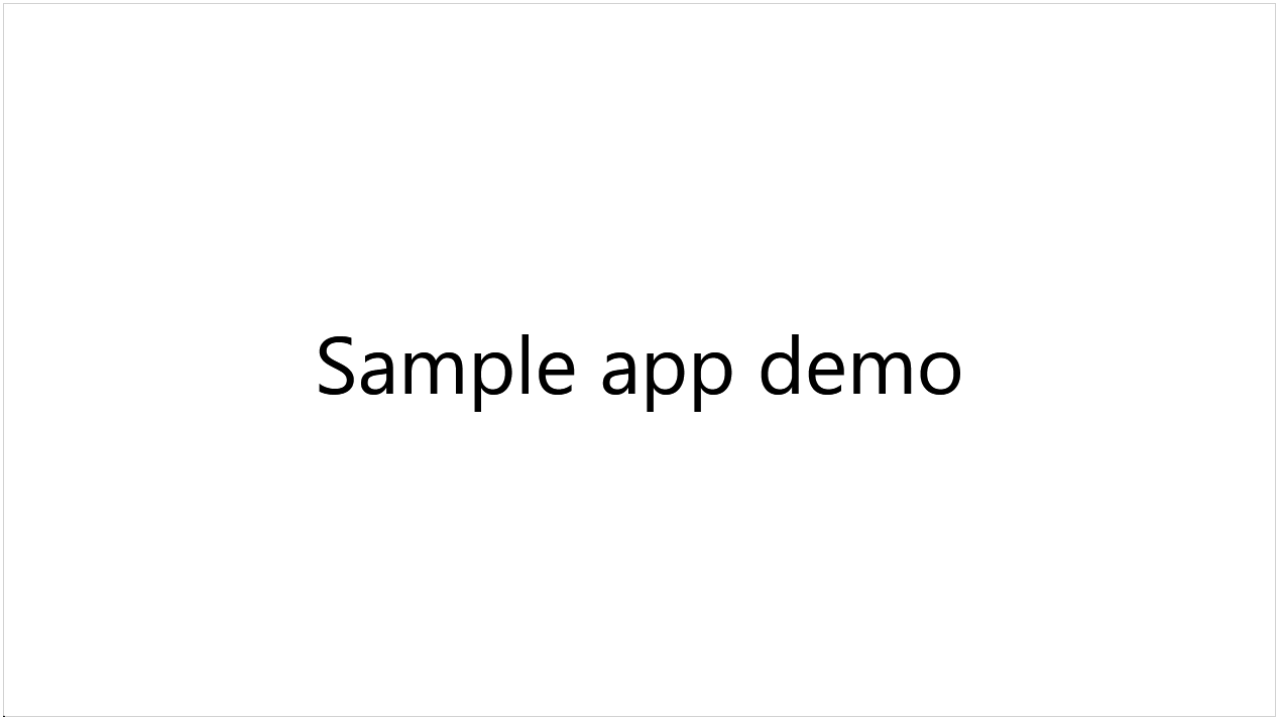
Provide product feedback

Create an app to edit tables in canvas apps

Article • 12/16/2022

Designing a productivity application to have related data and functions in one place enables you to achieve more without having to switch back and forth between the screens. Microsoft Excel is one such example that allows editing data real time in fast and efficient way.

Using Power Apps, you can apply the same concept by providing it as a front end to any data source. You're also able to customize it even more.



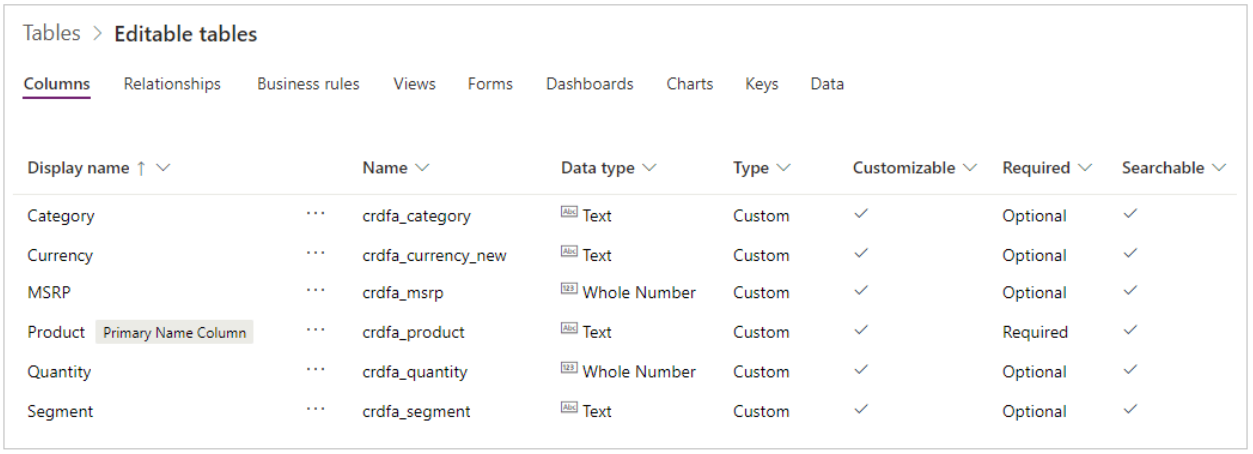
This tutorial uses the following components to make a sample app:

- A data source (Microsoft Dataverse, you can also use Excel instead)
- Form—For new items
- Gallery—To display existing items and
- Text input controls—To update existing items

Prerequisites

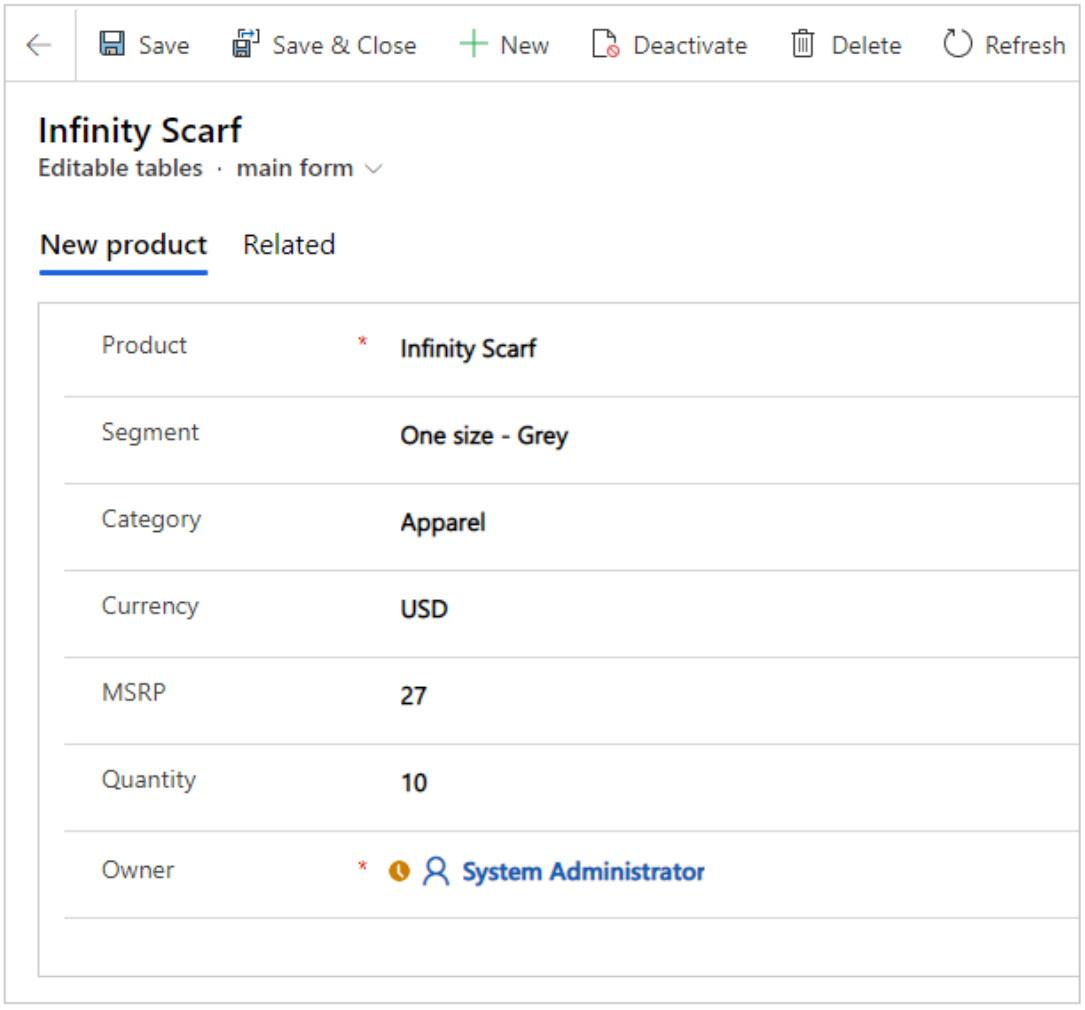
To follow this tutorial, you'll need access to a Power Platform environment, and the ability to create tables in Microsoft Dataverse.

The tutorial uses the following structure to create the sample app:



To learn about how to add columns, see Work with table columns.

A new main form has been created to add sample data:



To learn about how to create a main form with the required columns, see Create a form. Be sure to use the correct form order for adding records using the new form.

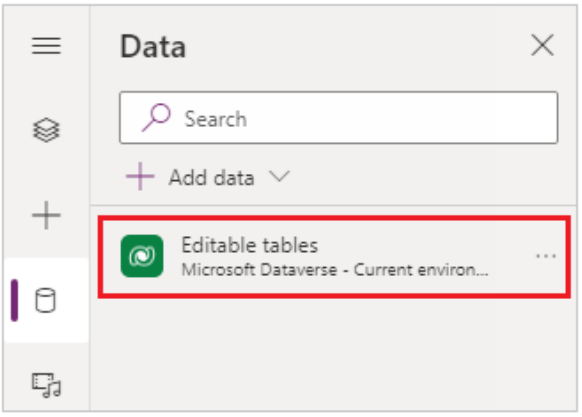
Step 1: Create blank app

Create a blank canvas app with the name "Catalog Management app" and **Tablet** layout.

Step 2: Add a data source

This section shows how to add a Dataverse table as the data source for the sample app. You can also use an Excel file from a SharePoint site, or OneDrive as the data source; or any other data source type of your choice.

1. From the left-pane, select **Data** > **Add data**.
2. Select **See all tables**.
3. Select **Editable tables**, or the table that you created earlier.

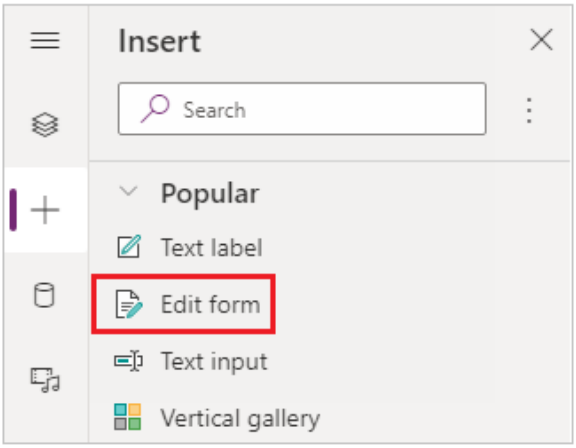


For more information about adding a connection to a canvas app, see Add data source.

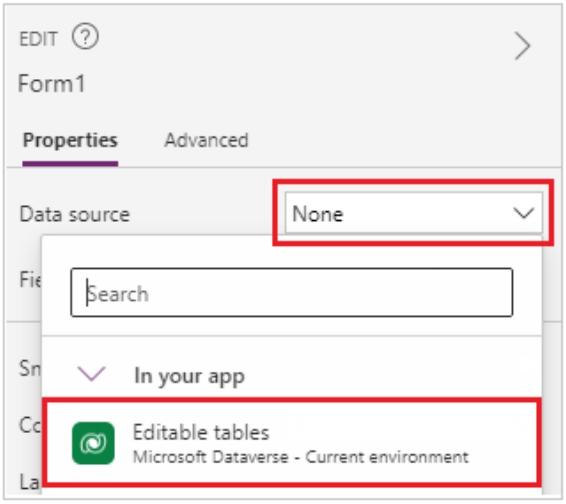
Step 3: Set up a form control

This step adds a form control to add new items.

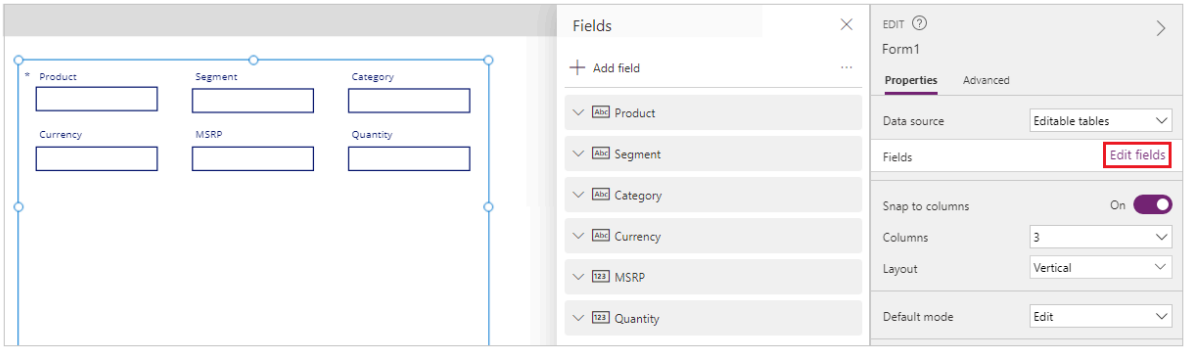
1. Select + (Insert) > **Edit form**.



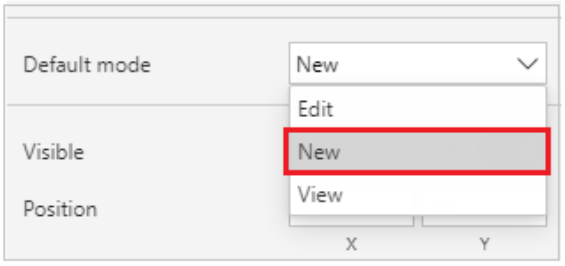
2. On the right-pane, choose the table as the data source for the edit form control.



3. Use the **Edit fields** properties option to select the columns to show on the edit form control. You can also change the column order as appropriate.



4. Choose the **Default mode** for the form to **New**.



5. Adjust the **Width**, **Height** properties for size of the data cards to fill the canvas as appropriate.

6. On the left-pane, select + (Insert) > **Button**.

7. Update the button text to **Add product**.

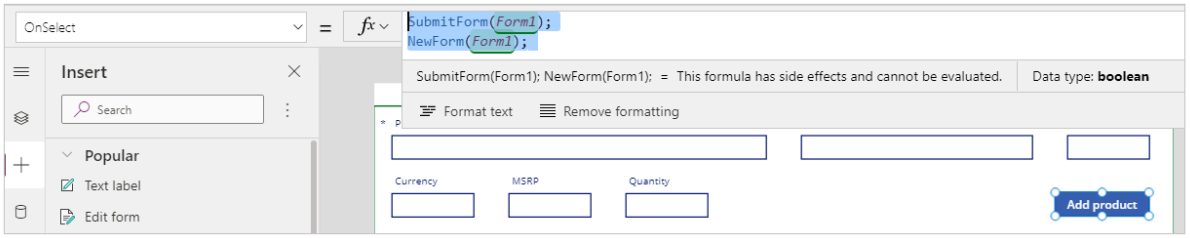
8. Select **OnSelect** property for the button control from the top-left side of the screen.

9. In the formula bar, enter the following formula.

Power Fx

```
SubmitForm(Form1);
NewForm(Form1);
```

- SubmitForm function submits the new product details to the Dataverse table.
- NewForm changes the mode of the form back to new form to add new products after submitting new product details.
- **Form1** in this formula is the name of the edit form control added earlier. Update the form name in this formula if your form name is different.



Step 4: Set up gallery as editable table

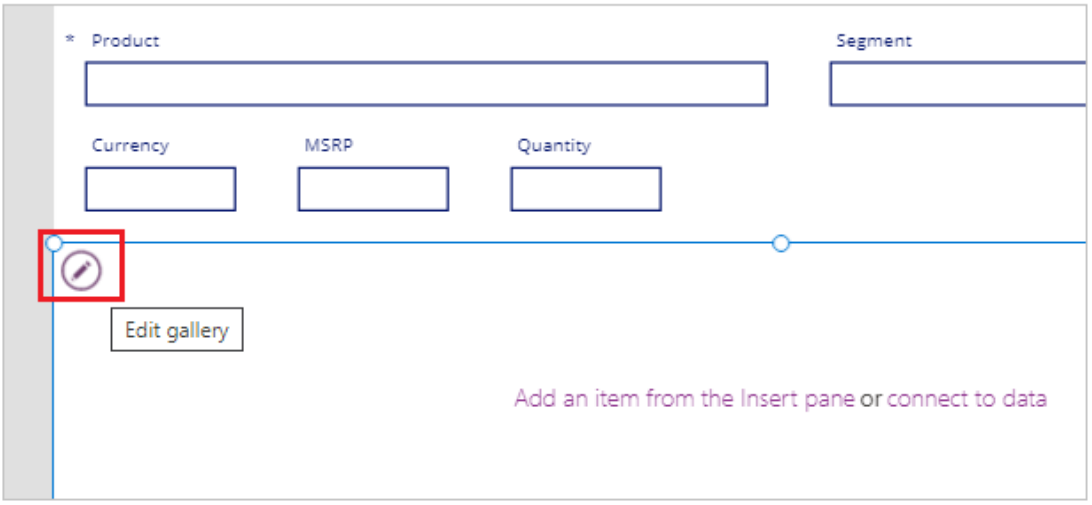
This step adds a blank vertical gallery to edit the items as an editable table.

1. On the left-pane, select + (Insert) > **Layout** > **Blank vertical gallery**.

2. Select the table from Dataverse as the data source.

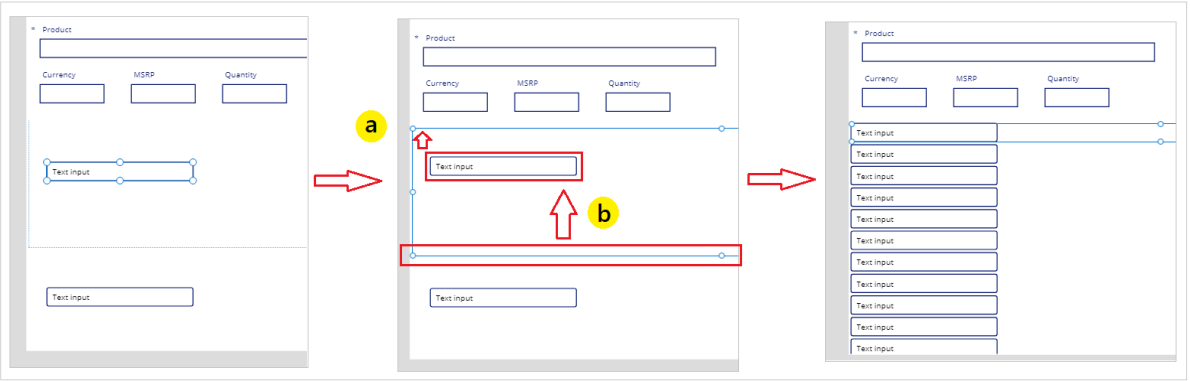
3. Resize the gallery to take over remaining half of the entire canvas since this gallery will contain all editable columns from the data source.

4. Select **Edit gallery**.



5. On the left-pane, select + (Insert) > **Input** > **Text input**.

6. Align gallery with the following actions:



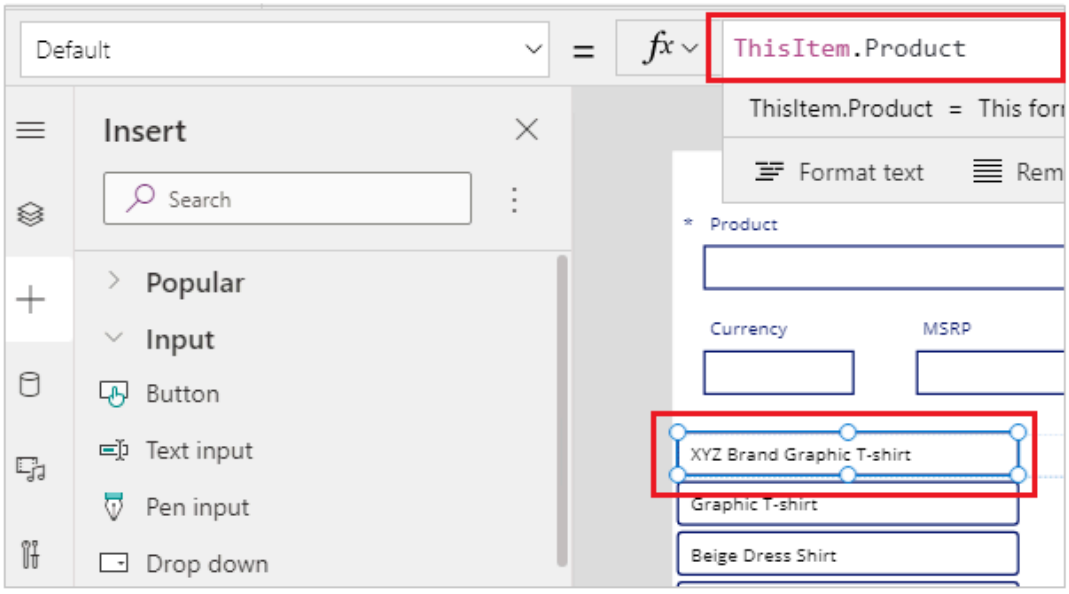
a. Move the top-most row for text input control to the top-left corner within the gallery. Moving this top-most row automatically moves remaining rows for same column.
b. Resize the first row height within the gallery to occupy the size of the first row you just moved to top-left.

At the end, you should see all rows stacked on left-side of the gallery.

7. Update the **Default** property formula for the first text input row within the gallery:

```
Power Fx

ThisItem.Product
```

- ThisItem in this formula relates to the item within the selected data source; which in this example is the Dataverse table.
- "Product" is the name of the column within the table.
- When this formula is updated for the first row in gallery, it automatically updates the entire column with product names. If you see an error, check if you've connected the gallery to the Dataverse table.



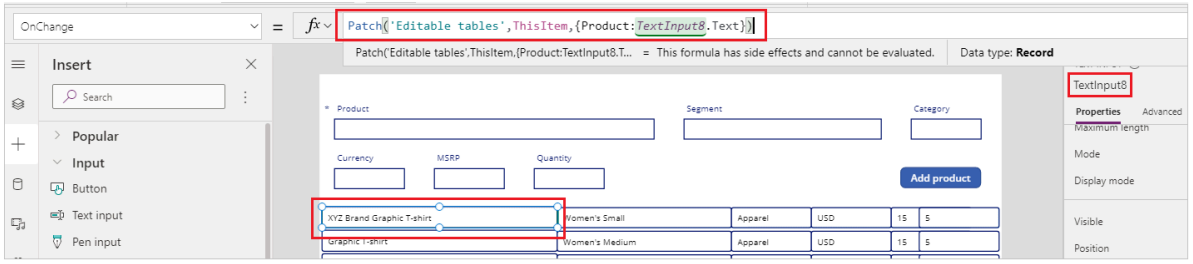
8. Repeat previous steps to add a text input control for each column that you want to show up inside the app as editable field. Ensure you choose the correct columns in the formula replacing "Product" with the appropriate column names.

> **Tip**
>
> To resize width, use **Width** property, or the drag using the buttons around the first row in each column.

9. Select the first row from the first column inside the gallery, and add the following formula to the **OnChange** property with changes to column and control names as appropriate:

```
Power Fx

Patch('Editable tables',ThisItem,{Product:TextInput8.Text})
```

   - Patch function in this formula updates the table selected as the data source with the values when changed.
   - "Editable tables" is the sample data source name.
   - ThisItem relates to the item within the selected data source for this column.
   - "Product" is the name of the column within the selected data source.
   - "TextInput8" is the name of the text input control added to the gallery attached to the "Product" column for the selected data source.
   - "Text" for "TextInput8" is the text entered in the selected cell within the editable table on the canvas.

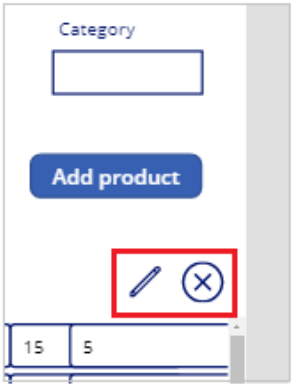


10. Repeat the previous step for each text input control added for rest of the columns from the connected data source. Ensure the formula is updated to use the correct column and control names.

Step 5: Set up edit and cancel options

This step adds the options to edit and cancel the edit progress. Gallery having text input controls editable by default may be subject to unintentional updates. Having an explicit edit option, and then to stop edit capability, ensures the edits are always expected.

1. Select **Insert** from the top menu > **Icons**, and add two icons - **Edit**, and **Cancel (badge)**.



2. Select **Tree view** from the left-pane, and then select **App**.

3. Select the app **OnStart** property, and add the following formula with changes to gallery name as appropriate:

```
Power Fx

Gallery2.DisplayMode = "galleryDisplayMode";
Set(galleryDisplayMode, DisplayMode.Disabled);
```

   - "Gallery2" in this formula is the name of the gallery.
   - **DisplayMode** is the DisplayMode property for the selected gallery.
   - "galleryDisplayMode" is the new variable that this formula creates to store the value of the gallery's display mode.
   - Set function sets the "galleryDisplayMode" variable defined in the previous line with the value of "Disabled" using "DisplayMode.Disabled". With disabled display mode, the gallery isn't editable by default when the app starts (app OnStart).

4. Select the blank vertical gallery, and then select the **DisplayMode** property.

5. Update the property value from `Edit` to `galleryDisplayMode`. This change ensures the gallery's mode is always defined using the value of the "galleryDisplayMode" variable created in the previous step.

6. Update the properties and formulas for the icons as below:

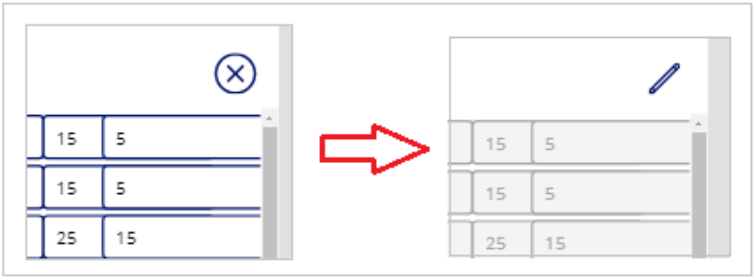
Expand table

Icon	Property	Formula
Edit	OnSelect	`Set(galleryDisplayMode, DisplayMode.Edit)`
Edit	Visible	`galleryDisplayMode = DisplayMode.Disabled`
Cancel	OnSelect	`Set(galleryDisplayMode, DisplayMode.Disabled)`
Cancel	Visible	`galleryDisplayMode = DisplayMode.Edit`

   - Set function sets the display mode of the gallery depending on which icon is selected.

- The variable "galleryDisplayMode" controls the visibility of the edit or cancel icons. When the gallery is editable, only cancel button appears. And only edit button appears when gallery is disabled.

7. Overlap edit and cancel icons, since we're managing the visibility for both icons depending on the state of the gallery.



> **Tip**
>
> You can use the **Preview** button from top-right side of the screen to preview the app, or use F5 key on the keyboard. Another quick alternative for testing is to press the **Alt** key on the keyboard, and select the button while keeping the key pressed. This method also simulates app's preview functionality for the specific selection while key remains pressed.

Step 6: Add search capability

As the data in the connected data source grows, it becomes difficult to find a specific row. This step adds search capability to the app to filter the gallery with searched keywords making it easier to find relevant data.

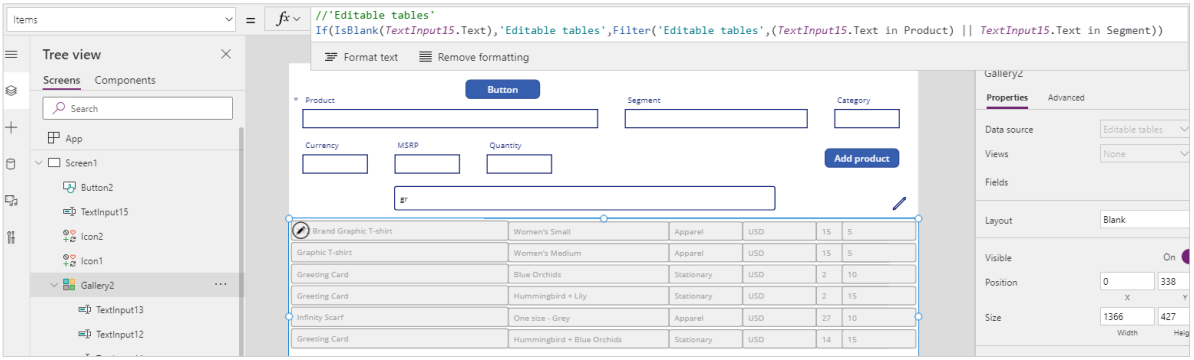
1. Add a text input control in the middle of the canvas, above the gallery.

2. Update the **Items** property for the gallery with the following formula instead of the table name, replacing table, and control names as appropriate.

```
Power Fx
If(IsBlank(TextInput15.Text),'Editable tables',Filter('Editable tables',
(TextInput15.Text in Product) || TextInput15.Text in Segment))
```

- If function checks for condition whether the search terms are specified, or blank. The gallery is populated depending on the listed items.
- IsBlank function checks the condition whether the search text is specified, or not.
- "Editable tables" is the name of the connected sample data source in this example. This value is set as the default if no search term is specified.
- Filter function filters the items in the gallery depending on the text criteria set within this function. This function is used to filter the gallery depending on the searched term.
- "TextInput15" is the name of the text input control being used as the search box to filter the gallery.

- "Product" and "Segment" are the names of the columns that are searched for using the text defined in the "TextInput15" control using the `.Text` syntax.
- "In" checks for the value of text input control to the specified column in the connected data source.
- "||" is the "OR" condition, signifying the search should be performed for Product, or Segment columns. Likewise, you can update formula to suite your specific business needs.



> **Tip**
>
> You can retain older functions, or add comments, using "//" inside the formula bar. This method can be more useful for more complex formulas, or to keep older functionality in case if you'd want to revert later.

Step 7: Miscellaneous changes such as branding, profile information, and reset

The sample app is complete with the expected functionality in this tutorial. However, extra considerations can be added—such as—the ability to reset the filter text input control, branding such as title for the gallery and app, and user profile details. You can also do more, start with Add, and configure controls in canvas apps.

The following table summarizes all the branding, profile, and reset information added to the sample app. Use the method described earlier in this article to work with different controls and properties. Ensure to replace the table, connector, and control names as appropriate.

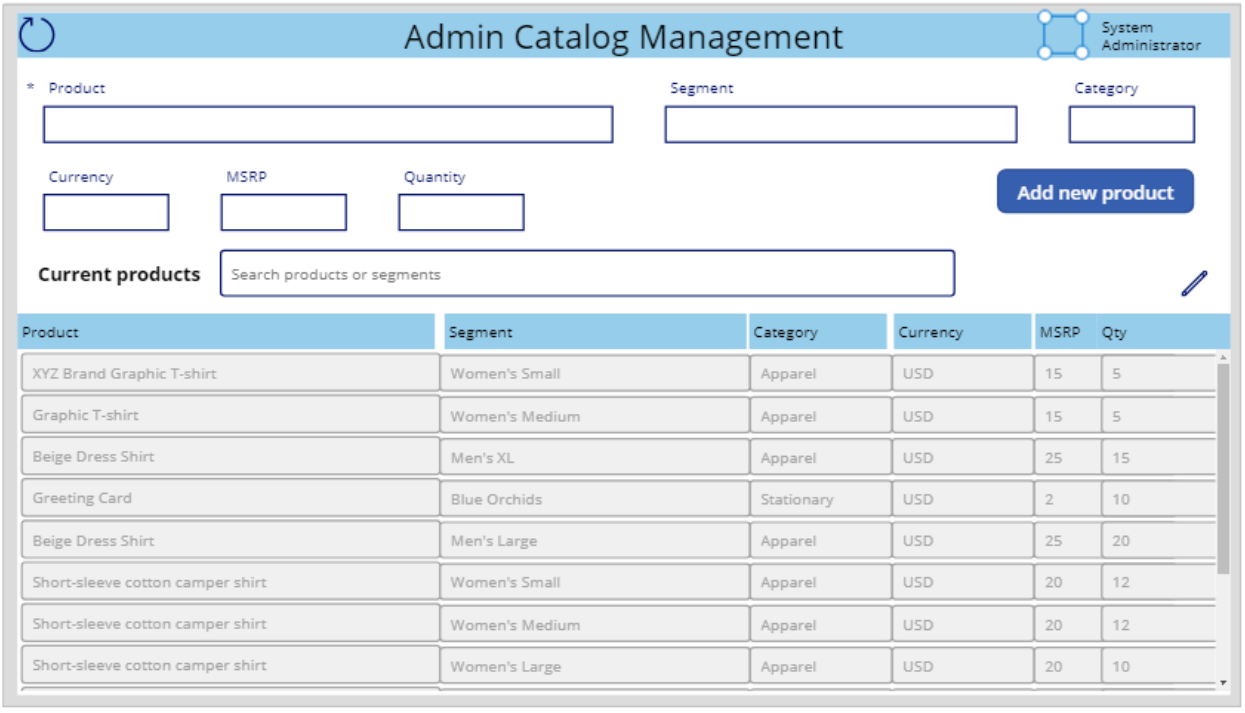
Expand table

Capability	Control	Properties	Additional information
App banner/label at the top of the screen	Text label	**Text** - "Admin Catalog Management" **Font size** - 28 **Fill color** - Blue **Text alignment** - Align center	Change all values as appropriate.
Add user display name to the	Text label	**Text** - `Office365Users.MyProfileV2().displayName`	Connect to Microsoft 365 first by

Capability	Control	Properties	Additional information
top-right side of the screen			adding a connection to Office 365 Users.
Add user profile photo on the left-side of the user display name on the top-right side of the screen	Image	**Image** - `Office365Users.UserPhotoV2(Office365Users.MyProfileV2().userPrincipalName)`	Connect to Microsoft 365 first by adding a connection to Office 365 Users.
Reset the search text box on the top-left corner of the screen	Icon of type "Reload"	**OnSelect** - `Reset(TextInput15)` where "TextInput15" is the text input control used as the search box.	When selected, it will reset the search text box resulting in the gallery to display all data by default.
Update text for **Add product** button	Add product button	**Text** - "Add new product"	Change as appropriate.
Add a label above the gallery, on the left-side of the search input box to signify current products	Text label	**Text** - "Current products" **Font size** - 16 **Font weight** - Bold	Change as appropriate.
Add hint text for search text input box	Text input	**Hint Text** - "Search products or segments"	Change as appropriate, and ensure the list of fields (such as Products, Segments) aligns with the column names in

Capability	Control	Properties	Additional information
			your data source.

For example, the completed screen looks like this with the controls and properties configured above:



Step 8: Save, publish and share

Now that you've completed the sample app, save, publish and share the app within your organization, or with guests.

See also

- Control reference
- Working with formulas

Feedback

Was this page helpful? Yes No

Provide product feedback

Show, sort, and filter data in a canvas app gallery

Article • 12/16/2022

Create a gallery to show images and text about several products, and sort and filter that information.

In Power Apps, you can use a gallery to show several related items, just as you see in a catalog. Galleries are great for showing information about products, such as names and prices. In this topic, we create a gallery and sort and filter the information using Excel-like functions. Also, when an item is selected, a border is placed around the item.

> **Note**
>
> This topic uses a tablet app. You can use a phone app but you will need to resize some of the controls.

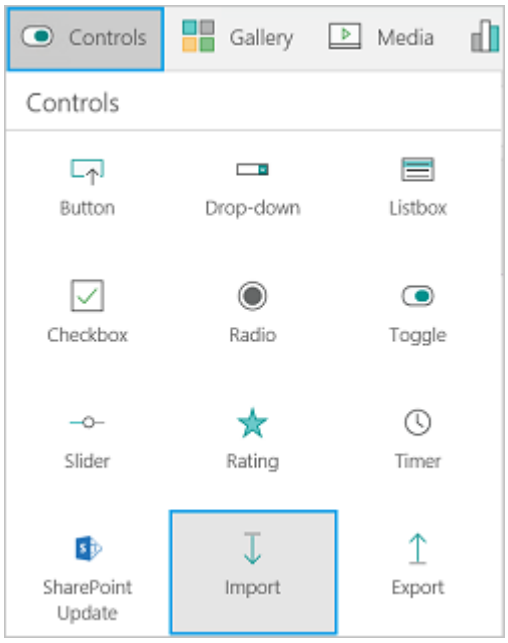
Prerequisites

- Sign up for Power Apps, and then sign in using the same credentials that you used to sign up.
- Create a tablet app from a template, from data, or from scratch.
- Learn how to configure a control.
- These steps use the CreateFirstApp as sample input data, which includes .jpg images. The zip file includes an XML file that can be converted to Excel. Otherwise, Power Apps automatically reads the files in the .zip files and imports it successfully. You can download and use this sample data, or import your own.

Show data in a gallery

1. Create a collection named **Inventory** using the sample data. Steps include:

    a. On the **Insert** tab, select **Controls**, and then select **Import**:

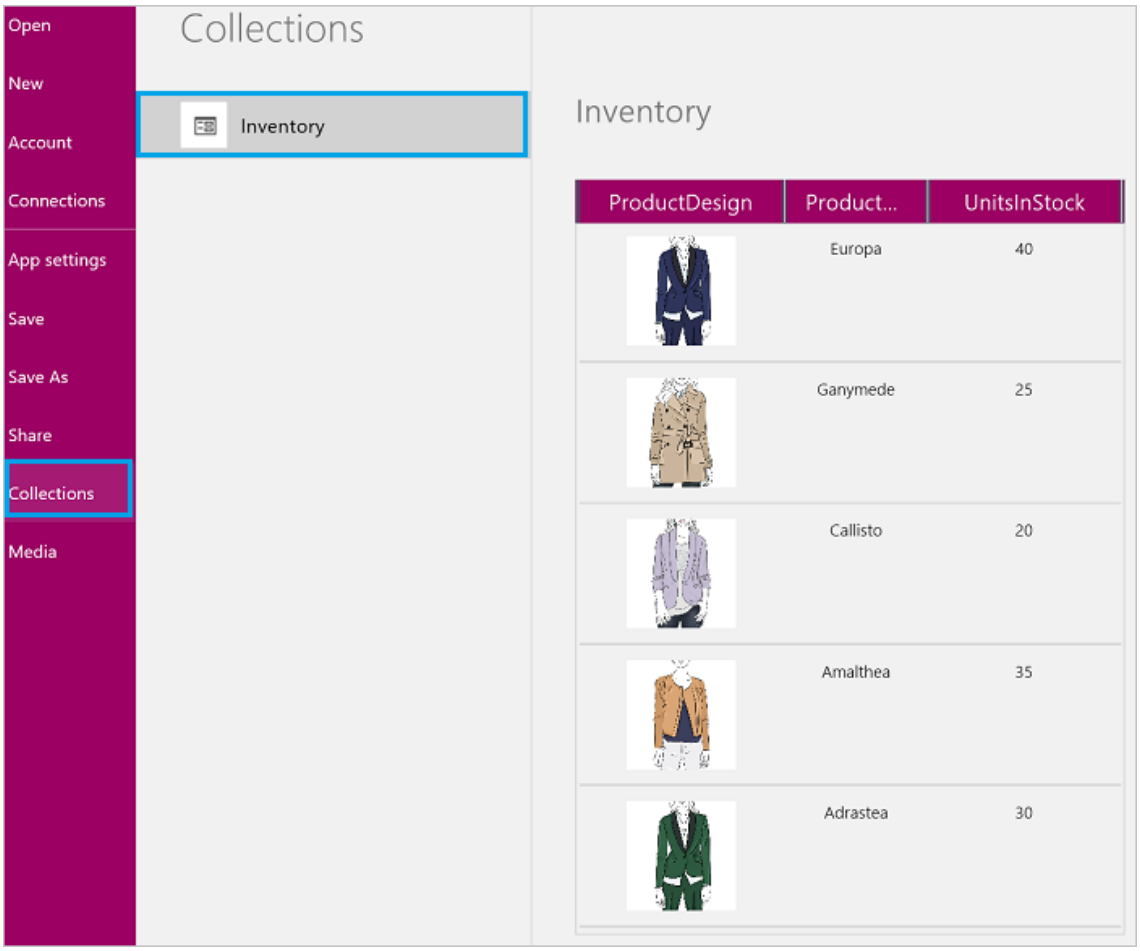


b. Set the OnSelect property of the import control to the following formula: **Collect(Inventory, Import1.Data)**



c. Select the **Import Data** button to open Windows Explorer. Select *CreateFirstApp.zip*, and select **Open**.

d. In the **File** menu, select **Collections**. The Inventory collection is listed with the data you imported:

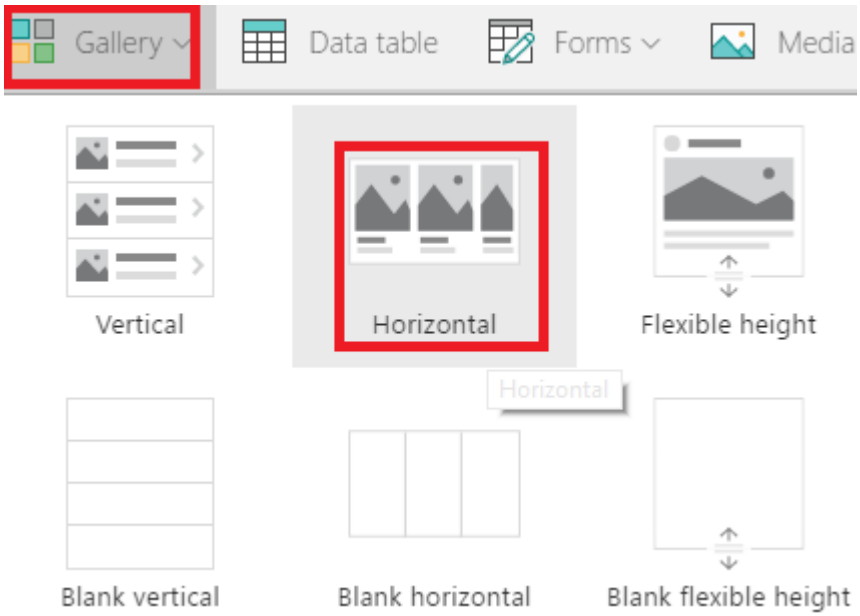


You've just created the Inventory collection, which contains information about five products, including a design image, the name of the product, and the number of units in stock.

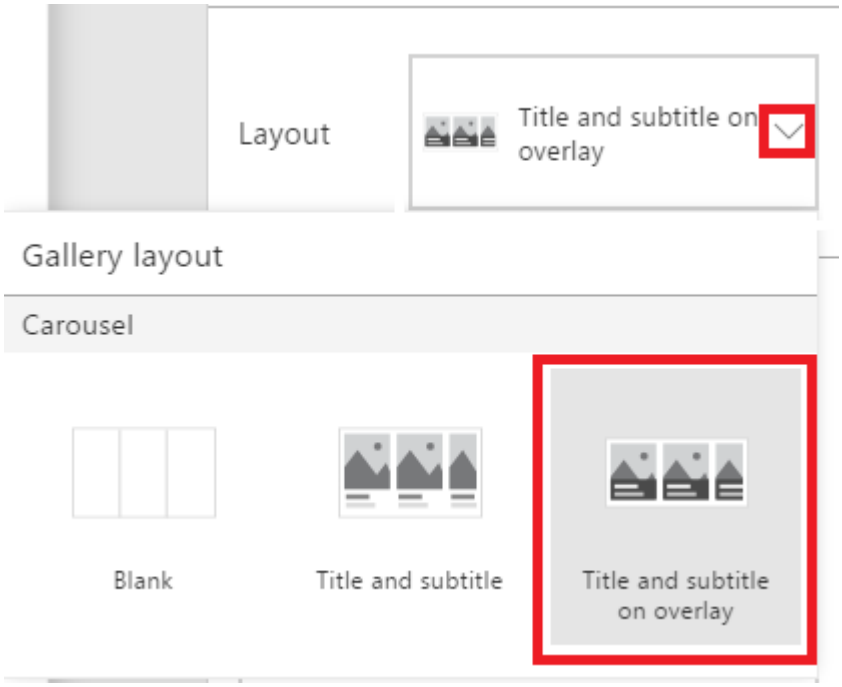
> ⓘ **Note**
>
> The import control is used to import Excel-like data and create the collection. The import control imports data when you are creating your app, and previewing your app. Currently, the import control does not import data when you publish your app.

2. Select the back arrow to return to the designer.

3. On the **Insert** tab, click or tap **Gallery**, and then click or tap the **Horizontal** gallery.



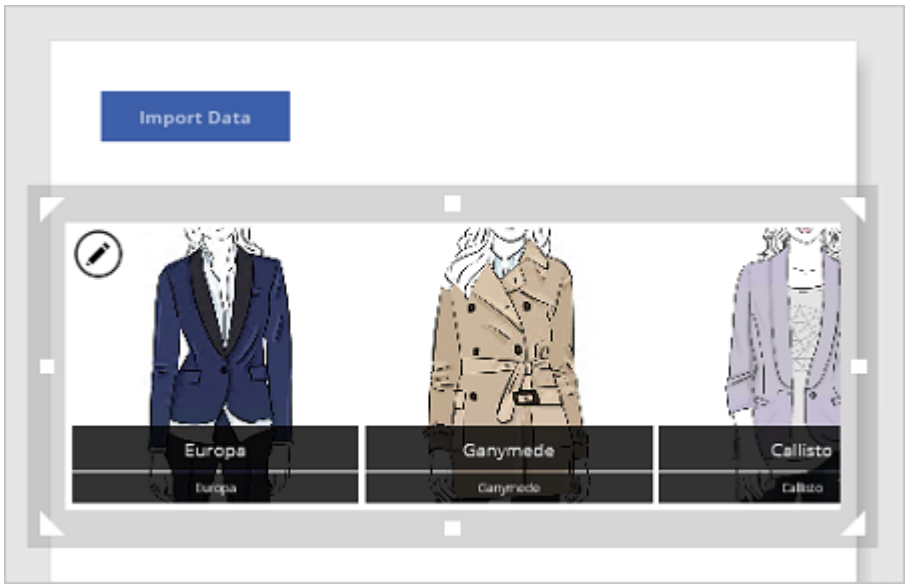
4. In the right-hand pane, click or tap the option in which the title and the subtitle overlay the graphic:



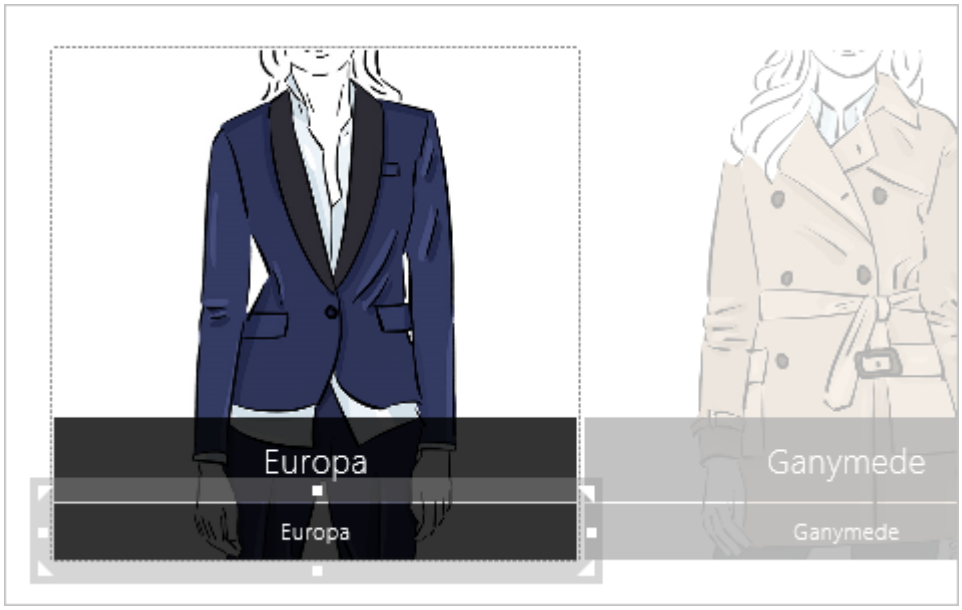
5. Set the Items property of the gallery to **Inventory**:



6. Rename the gallery to **ProductGallery**, and move the gallery so it doesn't block the other controls. Resize the gallery so it shows three products:



7. In the first item of the gallery, select the bottom label:



> **Note**
>
> When you change the first item in any gallery, you automatically change all other items in the gallery.

8. Set the **Text** property of the label to the following expression:
   ThisItem.UnitsInStock

   When you do this, the label shows the units in stock for each product:



> ⓘ **Note**
>
> By default, the Text property of the top label is set to `ThisItem.ProductName`. You can change it to any other item in your collection. For example, if your collection has *ProductDescription* or *Price* fields, you can set the label to `ThisItem.ProductDescription` or `ThisItem.Price`.

Using these steps, you imported data that includes .jpg images into a collection. You then added a gallery that displays the data and configured a label to show the units in stock for each product.

Highlight the gallery item you select

1. Select any item in the gallery *except* the first one. The edit icon displays (upper left corner). Select the edit icon:

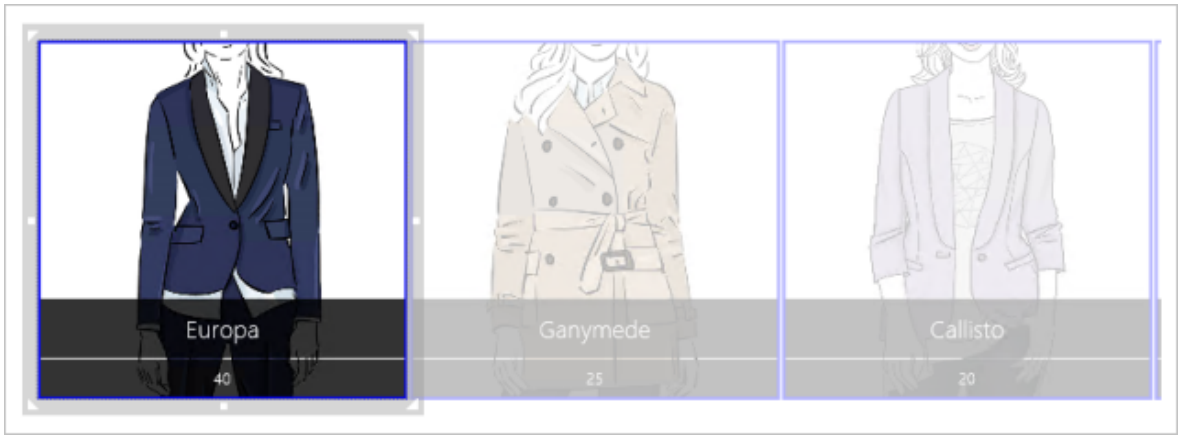
2. On the **Insert** tab, select **Shapes**, and then select the rectangle. A blue solid rectangle appears in each gallery item.

3. On the **Home** tab, select **Fill**, and then select **No Fill**.

4. Select **Border**, select **Border Style**, and then select the solid line.

5. Select **Border** again, and set the thickness to 3. Resize the rectangle so that it surrounds the gallery item. The items in your gallery now have a blue border and

should look similar to the following:



6. On the **Shape** tab, select **Visible**, and then enter the following formula in the Formula Bar:

   If(ThisItem.IsSelected, true)

   A blue rectangle surrounds the current selection in a gallery. Select a few gallery items to confirm that the rectangle appears around each item that you select.

   Remember, you can also open **Preview** to see and test what you're creating.

> **Tip**
>
> Select the rectangle, select **Reorder** on the **Home** tab, and then select **Send to Back**. Using this feature, you can select a gallery item without the border blocking anything.

Using these steps, you added a border around the current selection in the gallery.

Sort and filter items in the gallery

In these steps, we are going to sort the gallery items in ascending and descending order. Also, we'll add a slider control to 'filter' gallery items by the units in stock that you choose.

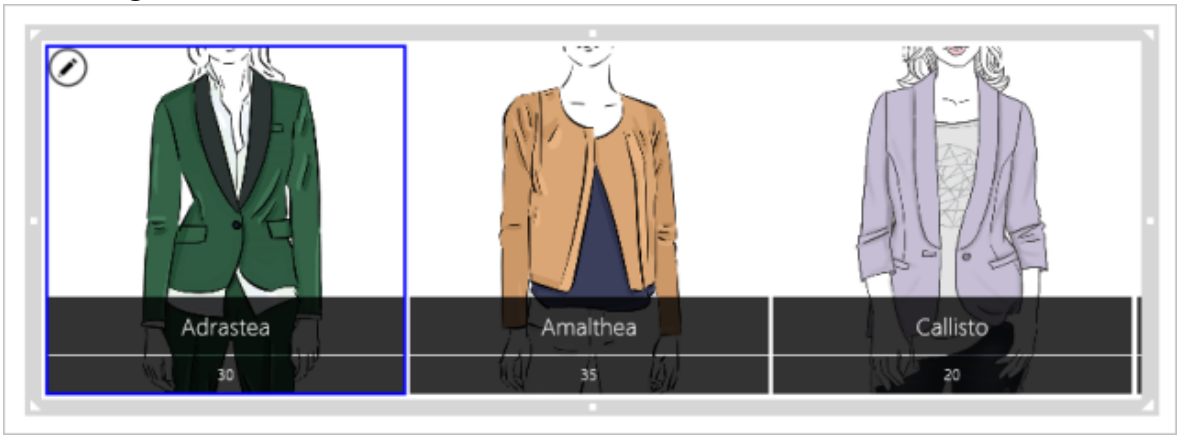
Sort in ascending or descending order

1. Select any item in the gallery *except* the first one.

2. The **Items** property is currently set to Inventory (the name of your collection). Change it to the following:

   Sort(Inventory, ProductName)

When you do this, the items in the gallery are sorted by the product name in ascending order:



Try descending order. Set the **Items** property of the gallery to the following formula:

Sort(Inventory, ProductName, Descending)

Add a slider control and filter items in the gallery

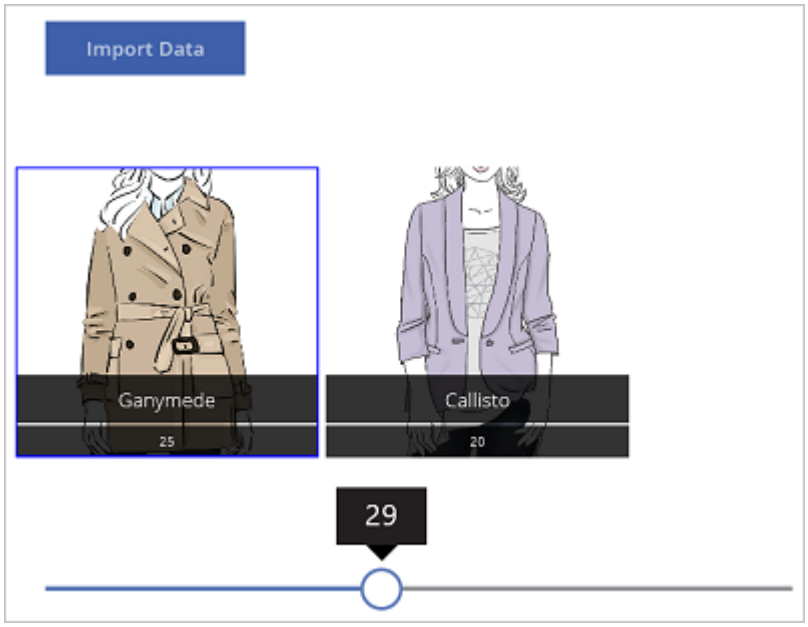
1. Add a Slider control (**Insert** tab > **Controls**), rename it to **StockFilter**, and move it under the gallery.

2. Configure the slider so that users can't set it to a value outside the range of units in stock:
   a. On the **Content** tab, select **Min**, and then enter the following expression:
   `Min(Inventory, UnitsInStock)`
   b. On the **Content** tab, select **Max**, and then enter the following expression:
   `Max(Inventory, UnitsInStock)`

3. Select any item in the gallery *except* the first one. Set the **Items** property of the gallery to the following expression:
   `Filter(Inventory, UnitsInStock<=StockFilter.Value)`

4. In **Preview**, adjust the slider to a value that's between the highest and the lowest quantity in the gallery. As you adjust the slider, the gallery shows only those

products that are less than the value you choose:



Now, let's add to our filter:

1. Go back to the designer.
2. On the **Insert** tab, select **Text**, select **Input Text**, and rename the new control to **NameFilter**. Move the text control below the slider.
3. Set the **Items** property of the gallery to the following expression:
   `Filter(Inventory, UnitsInStock<=StockFilter.Value && NameFilter.Text in ProductName)`
4. In **Preview**, set the slider to *30*, and type the letter *g* in the text-input control. The gallery shows the only product with less than 30 units in stock *and* has a name with the letter "g":



Tips and Tricks

- At anytime, you can select the preview button () to see what you created and test it.
- When designing your app, you can re-size the controls and move them around using click-and-drag.
- Press **ESC** or select the **X** to close the preview window.
- When using a gallery, select the first item in the gallery to change all items in the gallery. For example, select the first item to add a border to all items in the gallery.
- To update the properties of the gallery, select any item in the gallery *except* the first one. For example, select the second item to update the *Items*, *ShowScrollbar*, and other properties that apply to the gallery (not the items in the gallery).

What you learned

In this topic, you:

- Created a collection, imported data that includes .jpg images into that collection, and showed the data in a gallery.
- Under each image in the gallery, configured a label that lists the units in stock for that item.
- Added a border around the item that you select.
- Sorted the items by product name in ascending and descending order.
- Added a slider and an input text control to filter the products by units in stock and product name.

Use modern themes in canvas apps

Modern themes are preestablished style sets that transform the look of your canvas app in Power Apps. These themes, based on Microsoft's Fluent design language, modify various style aspects such as color, typography, borders, and shadows, ensuring a visually pleasing interface that streamlines app development.

Modern themes enable makers to effortlessly design apps with unified and consistent visual appeal, reducing customization time while maintaining professional aesthetics.

Prerequisites

Enable the modern themes feature in the settings of an app:

1. Open your app or create a new app.
2. From your app, open **Settings** from the app authoring menu or modern command bar.
3. Go to **Updates** > **New** > **Modern controls and themes**.
4. Set the **Modern controls and themes** toggle to **On**.
5. Close the **Settings** popup.

For more information, see Enable modern controls and themes for your app.

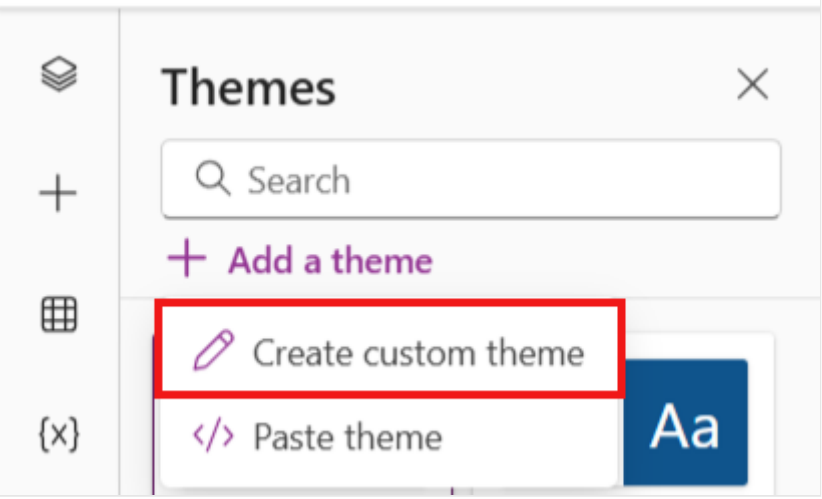
> **Note**
>
> When you enable modern controls and themes, you can't select classic themes from the command bar. To use *retired* classic themes, go to **Retired** > **Keep classic themes** in **Updates** and toggle it to **On**.

Create a theme

Use the available themes to style your app. Or, design your own theme for a more personalized visual aesthetic. You can also paste a theme from YAML format.

1. On the app authoring menu, select **Themes**.
2. On the **Themes** pane, select **Add a theme**.
3. Select **Create custom theme** to design a new theme, or select **Paste theme** to import a theme from YAML format. For more information about pasting themes, see Copy and paste themes.

   The **Create a theme** dialog appears.

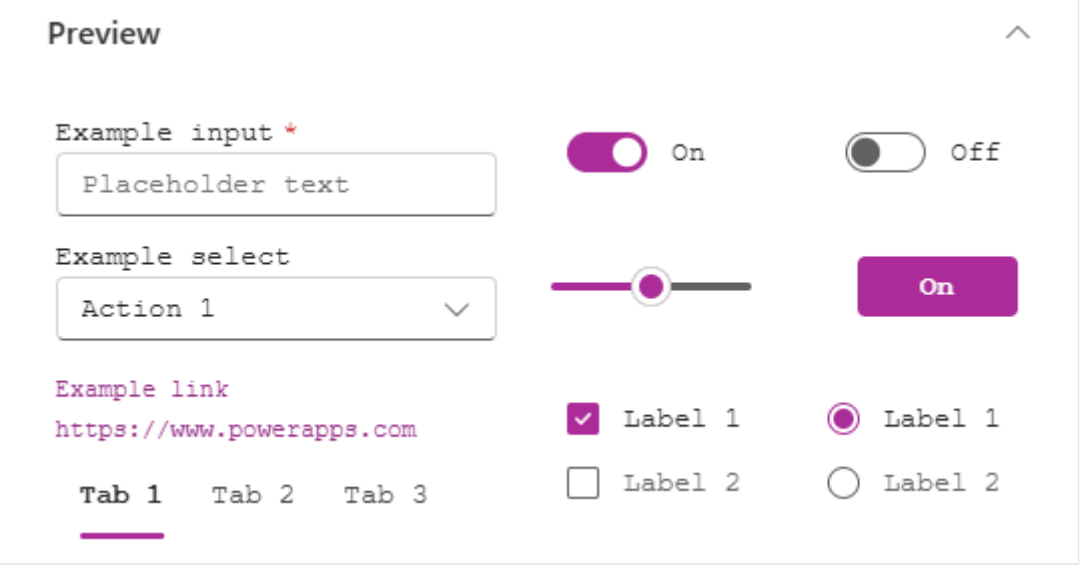


4. In the **Create a theme** dialog, choose or enter the following information:

Expand table

Feature	Action or description
Primary (seed) color	Choose a color from the color picker. You can, alternatively, enter the hexadecimal or RGB manually.
Lock primary color (preview)	The 16-slot theme palette is generated using the seed color. - Toggled off: The 16-slot palette is optimized for accessibility. Toggled off is the default option. - Toggled on: The seed color is placed in the primary (middle) slot of the palette. The remaining colors are generated by making the colors incrementally lighter on one side and darker on the other side. The generated palette might not meet contrast ratio accessibility requirements.
Theme name	Enter a unique theme name.
Font	Choose your default font used by the controls in the app.
Torsion	Impacts the tint, shade, or tone of the palette. Torsion isn't applicable if you choose **Lock primary color**.
Vibrancy	Impacts the muteness or brightness of the palette. Vibrancy isn't applicable if you choose **Lock primary color**.
Palette overrides	Override the color used for one or more slots in the palette by selecting a slot and choosing a new color. Select **Reset** to return to the generated value for that slot.

You can also look at a static preview of your new theme. Interact with the sample controls to see how your theme is applied to the rest state and various interaction states.



5. Select **Create**. Your new theme is created and applied to your app.

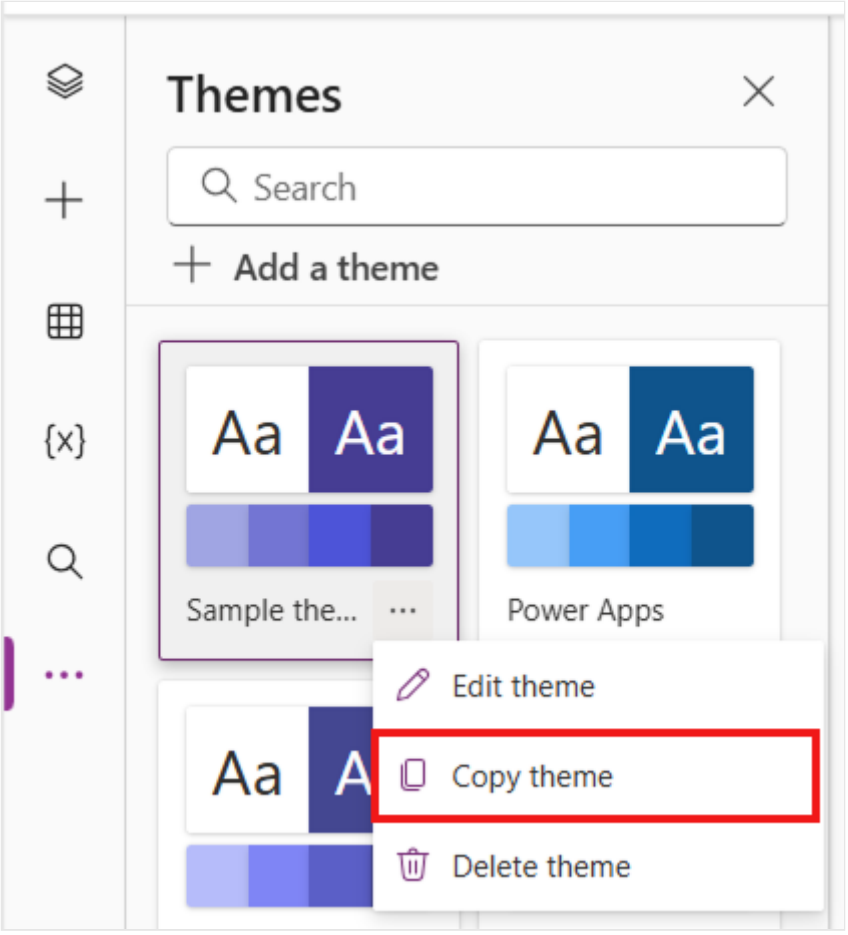
Copy and paste themes

You can copy and paste themes in YAML format to duplicate themes within your app or share them across different apps. This approach makes it easy to maintain consistent branding across multiple applications or back up your custom theme configurations.

Copy a theme

To copy a theme's YAML definition:

1. On the app authoring menu, select **Themes**.
2. On the **Themes** pane, locate the theme you want to copy.
3. Select the **More options** (...) button on the theme card, and then select **Copy theme**.

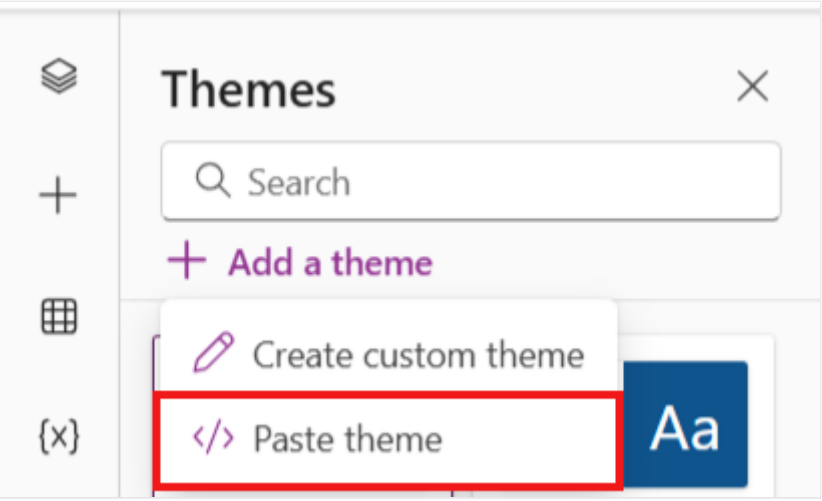


The theme definition is copied to your clipboard in YAML format.

Paste a theme

To paste a theme from YAML:

1. Copy a valid theme YAML definition to your clipboard.
2. On the app authoring menu, select **Themes**.
3. On the **Themes** pane, select **Add a theme**.
4. Select **Paste theme**.



The new theme is created and appears in your themes list with the name specified in the YAML. If a theme with the same name already exists, the pasted theme is automatically renamed to avoid conflicts.

YAML format

Theme definitions use the following YAML structure:

YAML

```yaml
Themes:
  Theme Name:
    Font: "'Segoe UI', 'Open Sans', sans-serif"
    BasePaletteColor: '#2e4bf0'
    HueTorsion: -27
    Vibrancy: -52
    ColorOverrides:
      Lighter30: '#bd3535'
      Darker10: '#1a2d8f'
```

The following table describes each property in the YAML format:

Expand table

Property	Description	Required
Theme Name	The display name for your theme. Must be unique within the app.	Yes
Font	CSS font family specification. Use standard web font names.	Yes
BasePaletteColor	The seed color for palette generation in hex format (for example, '#2e4bf0'). This color determines the base for all 16 color slots.	Yes
HueTorsion	Adjusts the hue rotation of generated colors. Range: -100 to 100. Negative values shift toward cooler tones, positive values toward warmer	Yes

Property	Description	Required
	tones.	
Vibrancy	Adjusts the saturation and brightness. Range: -100 to 100. Negative values create more muted colors, positive values create more vibrant colors.	Yes
ColorOverrides	Optional overrides for individual color slots (Lighter30, Lighter20, Lighter10, Base, Darker10, Darker20, Darker30, Darker40). Specify only the colors you want to customize.	No

Example: Share a corporate brand theme

This example shows a theme configured with corporate brand colors:

YAML

```yaml
Themes:
  Corporate Brand:
    Font: "'Segoe UI', 'Open Sans', sans-serif"
    BasePaletteColor: '#0078d4'
    HueTorsion: 0
    Vibrancy: 10
    ColorOverrides:
      Base: '#0078d4'
      Darker10: '#106ebe'
      Darker20: '#005a9e'
      Lighter10: '#2b88d8'
      Lighter20: '#c7e0f4'
```

To use this theme in another app:

1. Copy the YAML code.
2. Open the target app in Power Apps Studio.
3. In the **Themes** panel, select **Add a theme** > **Paste theme**.

The Corporate Brand theme is now available in your app.

Tips for using copy and paste with themes

- **Save themes externally**: Copy and save your custom themes to a text file or documentation system for backup and version control.
- **Share across teams**: Share theme YAML definitions with team members to ensure consistent branding across multiple apps.
- **Duplicate and modify**: To create theme variations, copy an existing theme, paste it, then modify the YAML properties rather than starting from scratch.

- **Font availability**: When pasting themes across apps or organizations, ensure the specified font families are available and properly licensed.
- **Minimal overrides**: For easier maintenance, only specify `ColorOverrides` for colors that must differ from the generated palette. Let the `BasePaletteColor`, `HueTorsion`, and `Vibrancy` properties handle the rest.
- **Test accessibility**: After pasting a theme, test it on both desktop and mobile views to ensure color contrast and readability meet accessibility standards (WCAG 2.1 AA minimum).

Apply modern theme

1. On the app authoring menu, select **Themes**.
2. On the **Themes** pane, select one of the default themes.
3. If you have classic controls in your app, you're prompted to apply the modern theme to these controls. You can always select **Undo** from the command bar if you don't like the theme.

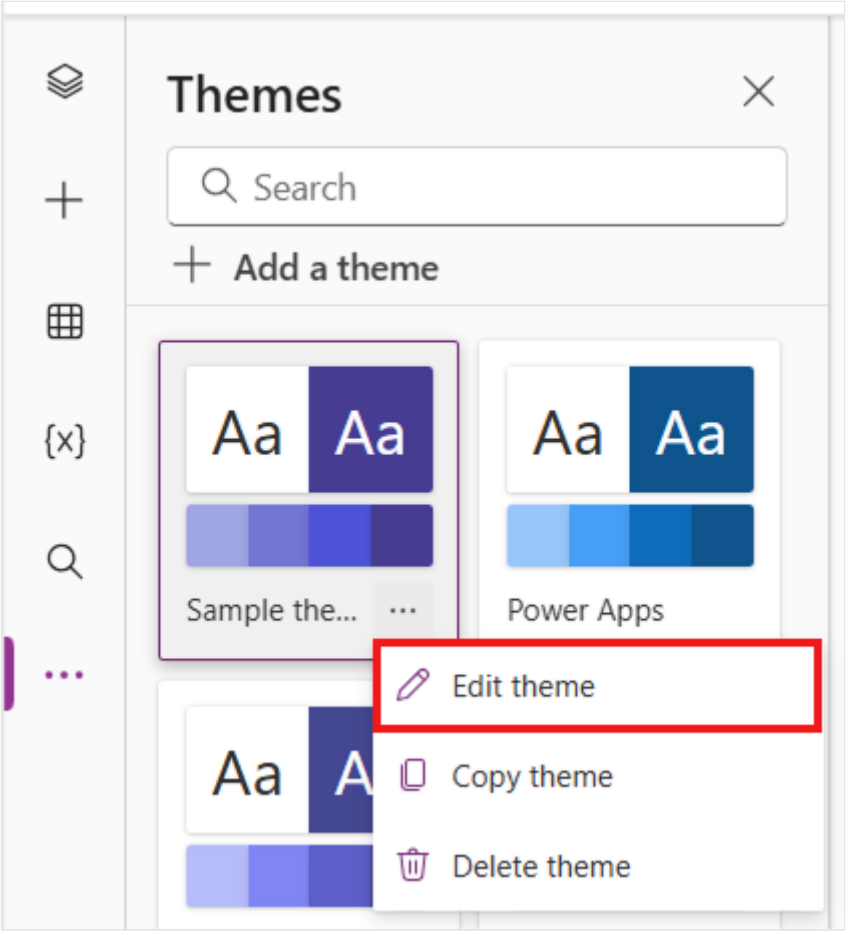
   > **Note**
   >
   > When you choose to apply modern themes to classic controls, Power Fx formulas are set on properties of the impacted controls that reference variables from the modern theme. Visually, classic controls might not align exactly with their modern control counterparts, as the classic controls aren't based on Fluent v9.

When you select a modern theme, the style of the theme automatically applies to the controls in your app. This action sets the **App.Theme** property.

Edit a theme

You can edit a theme and change one or more parameters of the custom theme.

1. On the app authoring menu, select **Themes**.
2. On the **Themes** pane, find the tile of the theme you want to edit. Select the overflow menu and then select **Edit**.



3. On the **Edit theme** pane, adjust the theme as needed.
4. Select **Save** or cancel out of the dialog.

> **Note**
>
> You can't edit default themes. Changes that you make through the visual theme editor update the underlying YAML properties. If you copy, modify, and re-paste a theme's YAML, those changes appear in the visual editor and anywhere the theme is applied.

Delete a theme

If you no longer need a custom theme in your app, you can delete it.

1. On the app authoring menu, select **Themes**.
2. On the **Themes** pane, find the theme that you want to delete, select the overflow menu, and then select **Delete**.
3. On the confirmation dialog box, select **Delete theme**.

If your app has any Power Fx formulas that reference this theme, you need to manually update the formula.

> **Note**
>
> You can't delete built-in themes.

> **Tip**
>
> Before deleting a custom theme, consider copying its YAML definition for backup. You can easily restore the theme later by pasting the YAML.

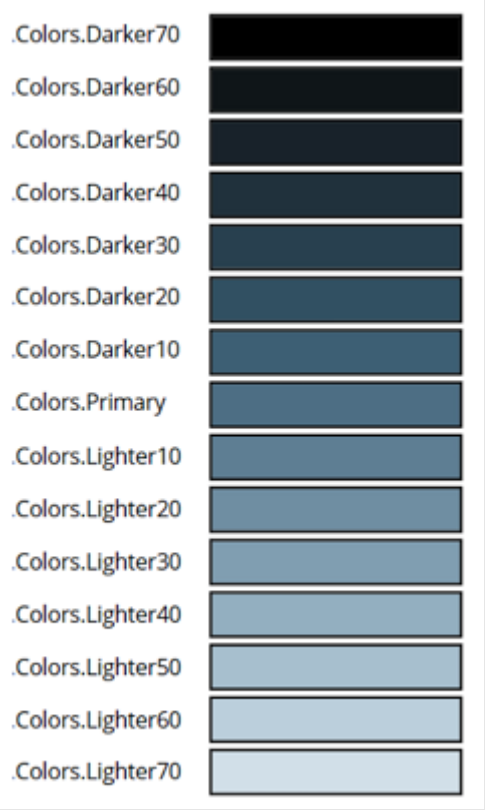
Use themes with Power Fx

Makers can use modern theme objects through Power Fx. Reference the currently active theme object by using **App.Theme**. Reference any theme loaded into the app by its instance name, such as **RedTheme**. To ensure that the color selections adapt to theme changes, reference the theme object by using **App.Theme**.

Each theme object includes the following information:

- **Name** of the theme
- **Colors**, a 16-color brand ramp collection that you can access individually by name; **Primary foreground color** is the default text color, such as the color of text on a background
- **Font** used in the theme

The following image shows the slot variables inside the **Colors** collection and, as an example, the corresponding colors for the **Steel** theme.



By using the theme brand ramp, you can manually style a classic control based on the current modern theme. For example, set `Button.Fill = App.Theme.Colors.Primary`.

> **Note**
>
> To provide feedback, see: Provide your feedback to Microsoft.

See also

Modern controls blog post | Using modern themes with Power Apps (video)

Last updated on 01/22/2026

Overview of modern controls and themes in canvas apps

Summarize this article for me

Canvas apps now support modern controls and theming based on the Microsoft Fluent 2 design system. Modern controls offer improved accessibility, performance, and usability compared to classic controls. The accompanying theming system lets you customize your app's appearance from a central location.

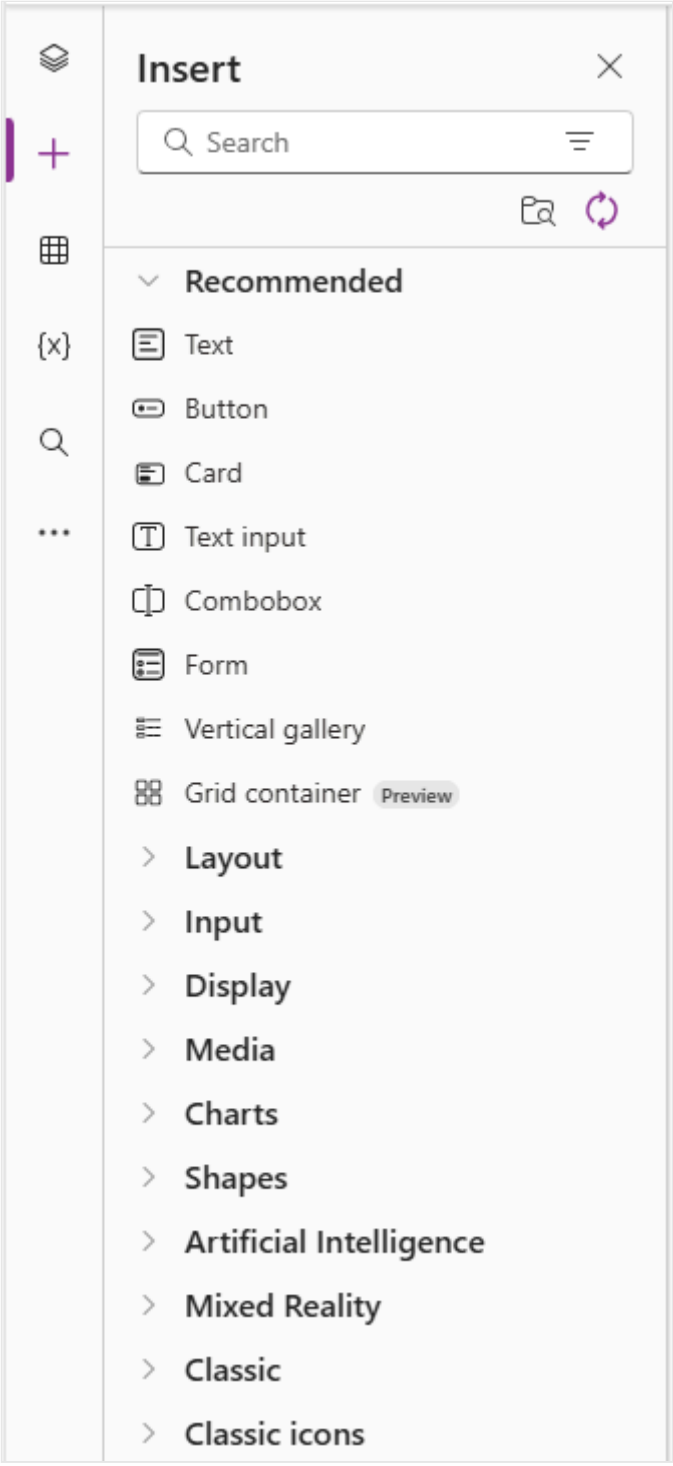
This article explains how to enable modern controls and themes in your canvas apps and provides information about ongoing development.

Enable modern controls and themes for your app

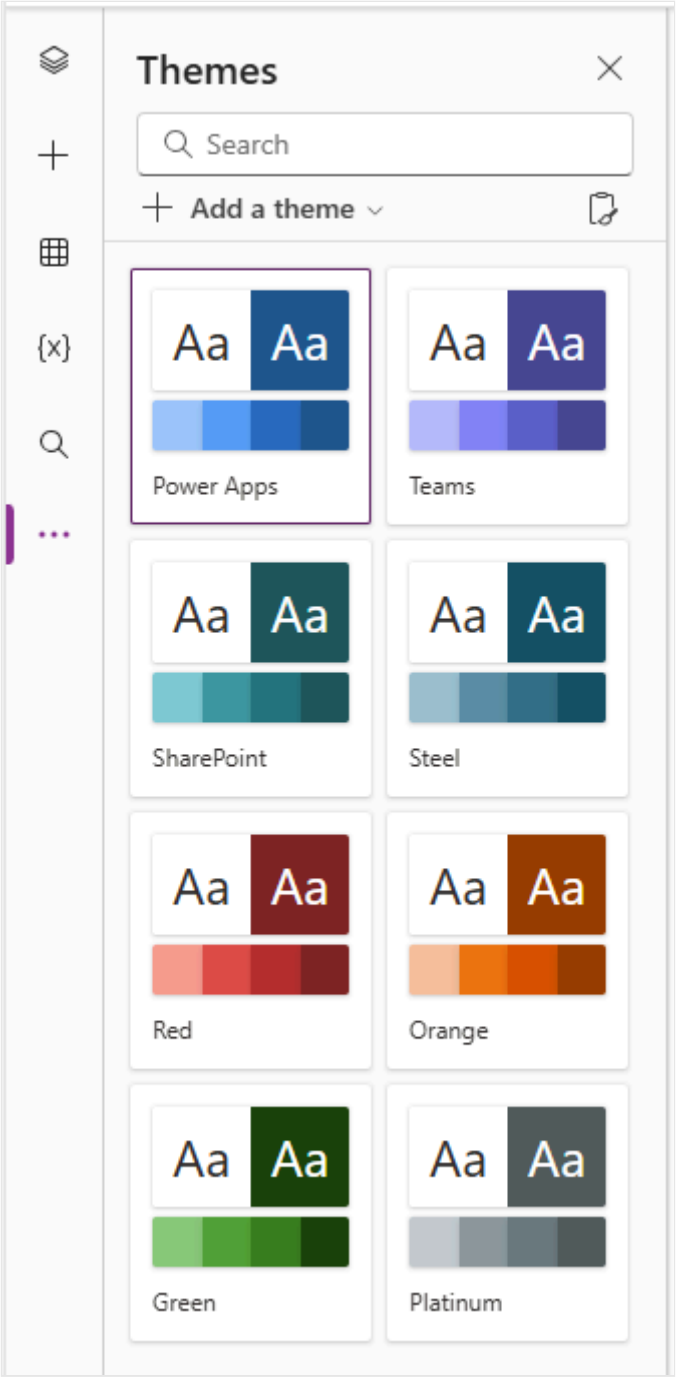
With your canvas app open for editing:

1. On the command bar, select **Settings** > **Updates**.
2. Select the **New** tab and then turn on **Modern controls and themes**.

- The app refreshes with the new app authoring menu. Modern controls are part of existing categories. Two additional categories exist for legacy controls - Classic and Classic icons.



- To see modern themes, on the app authoring menu, select **Themes**. For information, see Use modern themes in canvas apps.



What's next for these controls and themes?

The Power Apps team is consistently improving and adding new capabilities for modern controls and themes. They publish updates to the Power Apps blog every month on the progress made for modern controls.

Provide feedback to Microsoft

To provide feedback, offer suggestions, or report problems, go to the community forum.

You can also provide feedback for individual controls in the Power Apps Studio from the control properties by selecting the thumbs up or down and comments option. The same

feedback mechanism is available for modern themes in the **Themes** pane.

Last updated on 02/24/2026

Modern controls and properties in canvas apps

Modern controls in Power Apps provide an enhanced user experience with updated styling, improved accessibility, and better performance compared to classic controls. These controls follow Microsoft's Fluent Design System and offer a consistent, professional appearance across your canvas apps.

This reference guide covers the available modern controls and their properties. Each control serves a specific purpose - from basic interactions like buttons and text inputs to advanced features like Copilot integration and data visualization. Use these controls to build intuitive, accessible applications that align with modern design standards.

Before implementing modern controls in your app, review the overview of modern controls to understand key concepts and best practices.

Modern controls

Avatar – A control that shows a graphic representation of a user, team, or entity.

Badge – A badge is a visual decoration for UI elements.

Button – Interact with the app by clicking or tapping.

Card (preview) - Cards help makers create clean, responsive object summaries such as contact tiles, document previews, or product listings in either horizontal or vertical orientation.

Checkbox - Select or clear an option to specify **true** or **false**.

Combobox - A control that users can use to make selections from provided choices and supports search and multiple selections.

Copilot answer (preview) - A control that makers can use to add predefined questions that end users can use to get generated answers.

Data Grid (preview) - A control that displays records from a data source in a scrollable, sortable, and searchable grid.

Date picker - A control that the user can select to specify a date.

Dropdown (preview) – Select a value from the list of items.

Header – A control that creates a modern app header.

Info button – Use this control to provide additional information to users.

Link – Open hyperlinks in new tab.

Number input - A number input control the user can modify.

Progress bar – Displays the progress, can be configured as determinate showcasing exact progress or indeterminate for ongoing progress.

Radio group – An input control that shows multiple options, of which users can select only one at a time.

Spinner – Displays state in motion, for example, loading a page or table.

Slider - A control with which the user can specify a value by dragging a handle.

Stream (preview) - A control that lets you add Steam videos in your app.

Table (preview) – A control that shows a set of data in a tabular or list format.

Tabs or tab list – Select a tab to move screens or take action on app.

Text – Display text on the app, can also be used as label for fields.

Text input – A box in which the user can type text, numbers, and other data.

Toggle – A control that the user can turn on or off by moving the handle.

See also

Overview of modern controls in canvas apps

Last updated on 04/28/2026

Rename controls in canvas apps with Copilot (preview)

05/30/2025

[This article is prerelease documentation and is subject to change.]

Canvas apps often have many controls, so it's important to give them meaningful names for easier maintenance and collaboration. With the new **Proactive control rename** feature, Copilot suggests names for controls. Makers review the suggestions and apply changes to multiple controls at once. When a control's name is updated, the change appears across all its references.

Currently, the most frequently renamed and supported controls are:

- Labels
- Buttons

Additional support for more controls will be incorporated based on feedback from preview.

> **Important**
>
> - This is a preview feature.
> - This feature is available in English only.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so customers can get early access and provide feedback.

Prerequisites

Make sure you meet the prerequisites and region availability in Copilot in Power Apps overview (preview).

By default, the **Proactive control rename** setting is on for new apps.

To use this feature for an existing app, follow these steps:

1. Open your canvas app for editing in Power Apps Studio. On the command bar, select **Settings** > **Updates**.
2. On the **Preview** tab, find and turn on the **Proactive control rename** setting.

3. On the **New** tab, find and turn on the **New analysis engine** setting.

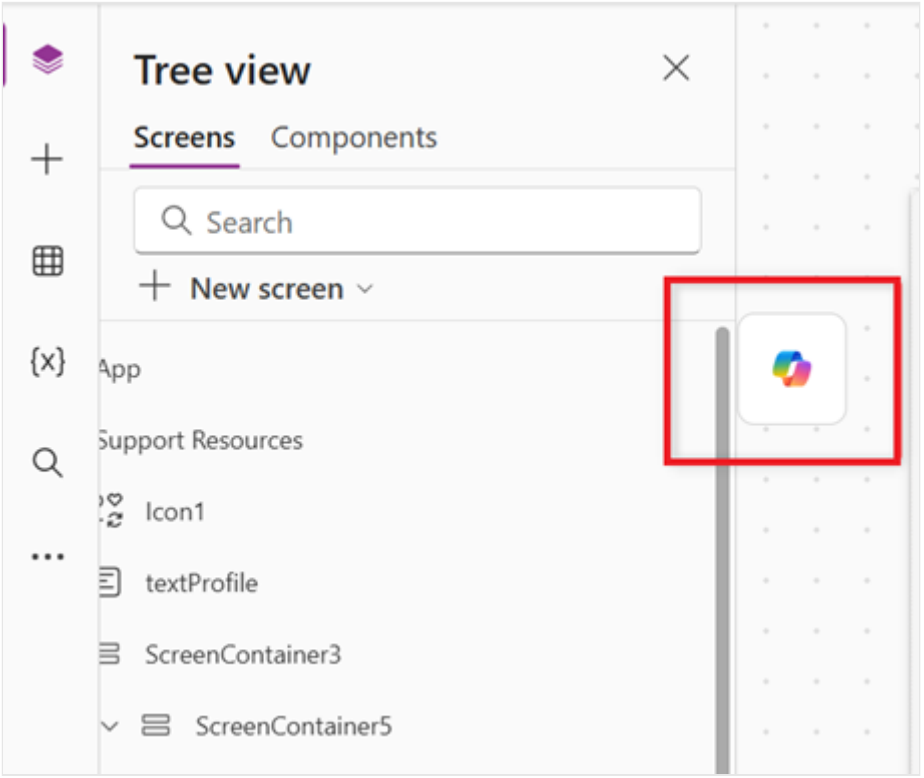
Rename controls

When you rename a control like the **Button** control, Copilot suggests names for other **Button** controls that have similar properties.

1. Open your canvas app for editing in Power Apps Studio.
2. In the **Tree view**, select a control, and then select **More options** (...).
3. Select **Rename**, type a new name for the control, and then press Enter.

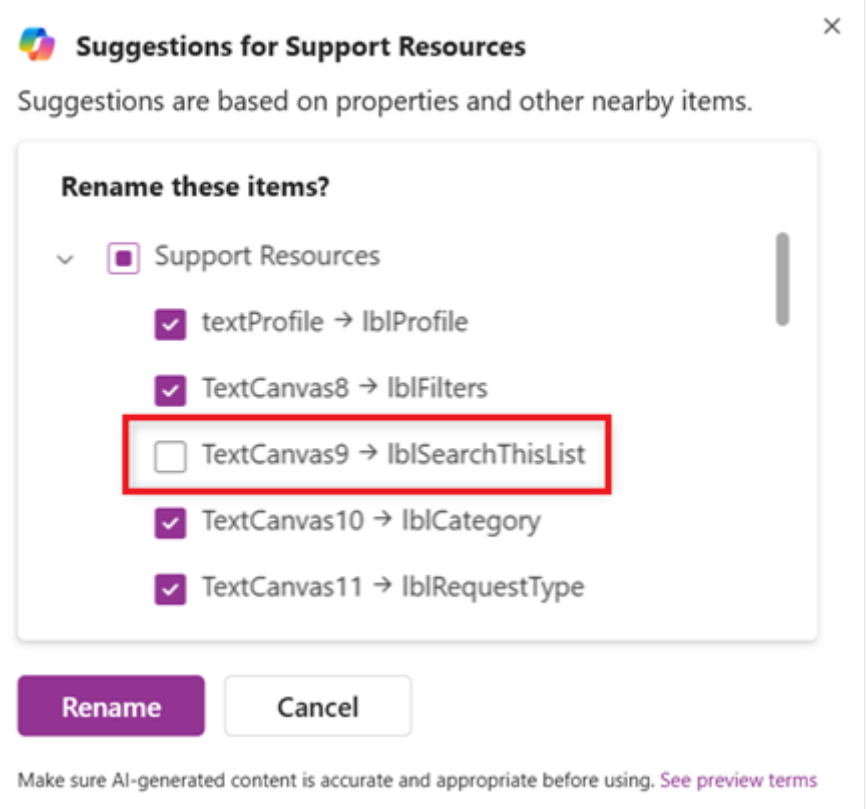
> **Note**
>
> When you manually rename a control, Copilot only appears if there are other controls on the same screen that can be renamed.

4. Copilot appears with suggestions to rename other controls of the same type.



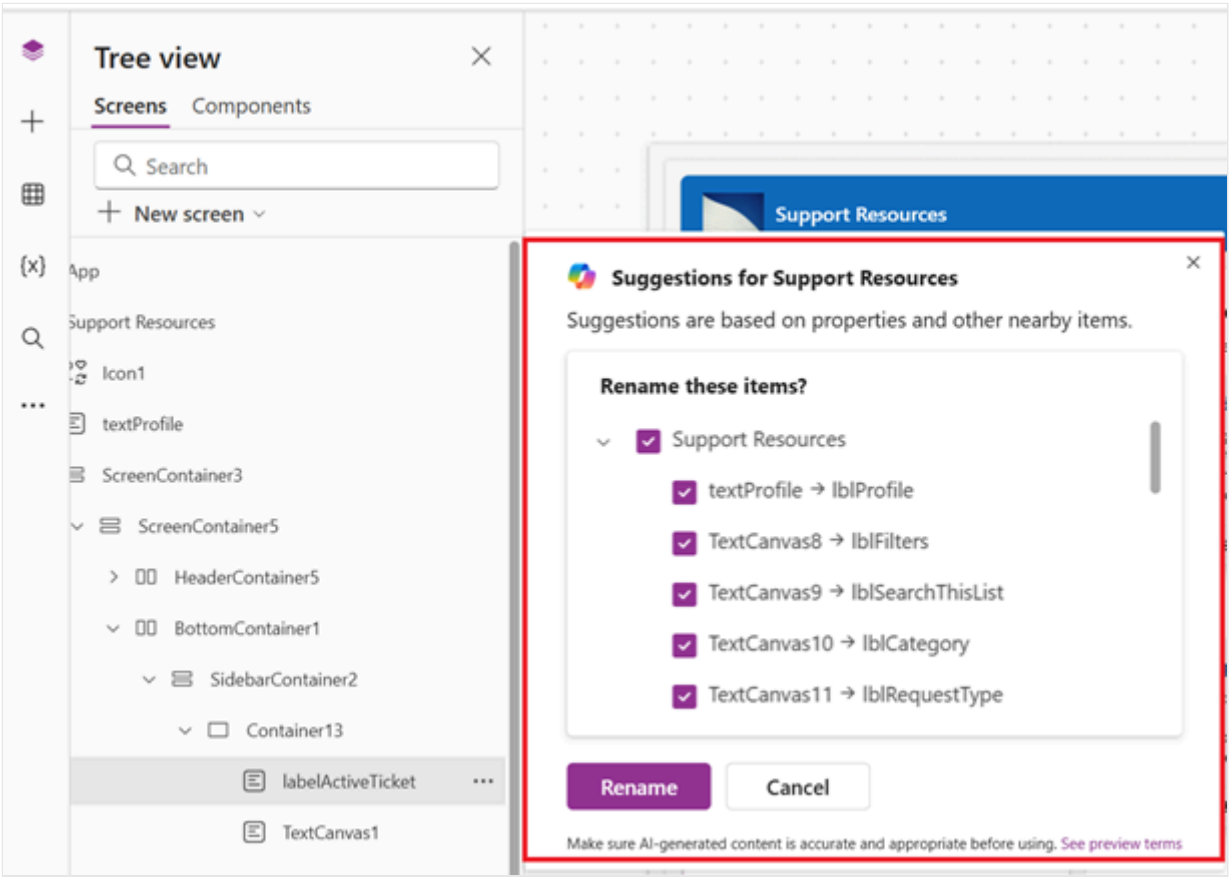
5. Select **View Suggestions**.

6. Review the suggested names, and clear any item that you don't want to rename.



7. When you're done, select **Rename** to apply the changes.

   If a formula references a control, the control name is automatically updated in the formula.



Best practices and recommendations

To get the best results from Copilot, use this feature after you set the **Text** property of a control. For example, Copilot doesn't suggest names for a button with the default Text property value **Button** or an invalid **Text** property value like **If()**.

If you still don't get the results you want, send us your feedback.

When you manually rename a control, use standard, relevant names and naming patterns so Copilot can reference them when suggesting renames. Avoid special characters and spaces. For example, renaming a **Button** to **collectResponse** gives better renaming suggestions for other buttons than using a name like **collect_1**.

Limitations

- Only the **Label** and **Button** controls are supported.
- When you rename a label manually, Copilot appears only if there are other labels under the same screen that can be renamed.
- Only controls with a valid **Text** property are considered for renaming by Copilot.
- The **Text** property can be a **Text** literal or a formula returning **Text** type.
- Only controls with nondefault **Text** property are considered for renaming. For example, the **Text** property value can't be the default value, such as **Button** or **Text**.

See also

FAQ for rename controls in canvas apps with Copilot

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Recent updates to modern controls in canvas apps

Starting February 2026, modern controls in canvas apps have updated versions with new property names, enum-based values, and behavior changes. This article explains what changed and how to update your existing apps.

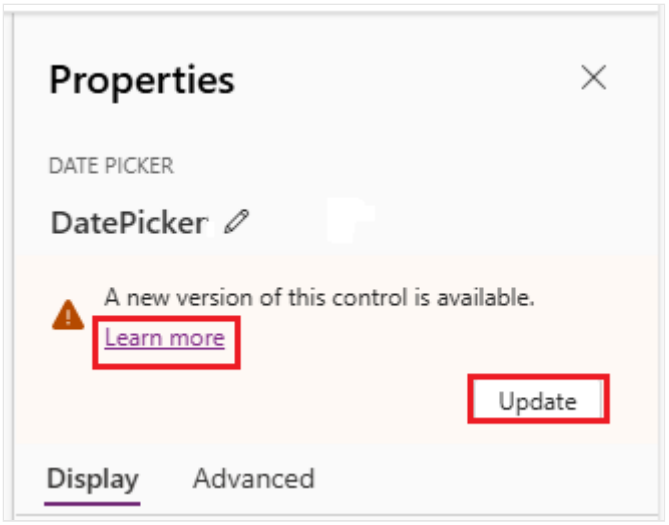
> **Important**
>
> These updates might require you to update formulas in your existing apps. Before you update any control, review the update guide for that control and fix any affected formulas.

Why update?

- **Better performance** — Events like OnChange fire less often, reducing unnecessary recalculations.
- **Better mobile experience** — Controls use touch-friendly sizes and font defaults on mobile layouts.
- **More styling options** — Separate corner radius properties and new border styles give you more design control.
- **IntelliSense and validation** — Enum-based values replace plain strings, so you get autocomplete and compile-time checks in your formulas.

Find and update controls in your app

When you open an app that uses an older version of a modern control, you see a notification on the control. The notification tells you that an update is available, and includes a **Learn more** link and an **Update** button.



Steps to update a control

1. Select **Learn more** to open the update guide for that control.
2. Review property renames, enum changes, and behavior differences listed in the guide.
3. Find formulas in your app that use renamed or removed properties, and update them.
4. Select **Update** to upgrade the control to the new version.
5. Test your app to make sure everything works as expected.

> **Tip**
>
> The **Update** button might not appear for all controls right away. Check back if you don't see it yet.

Controls with available updates

The following table lists each control that has an updated version, with a link to its specific update guide and a summary of what changed.

Expand table

Control	Update guide	Key changes
Text	Updates to Text control	Nine property renames, enum values, new OnSelect event, DisplayMode removed
Number Input	Updates to Number Input	OnChange on blur plus step clicks, TriggerOutput removed, new styling properties
Date Picker	Updates to Date Picker	View mode fixed, format honored, DateTimeZone respected, gallery crash fixed

Control	Update guide	Key changes
Text Input	Updates to Text Input	OnChange on focus out only, TriggerOutput default = Keypress, View mode truly read-only
Tab List	Updates to Tab List	New Appearance property, item order preserved, Items auto-refresh
Combo Box	Updates to Combo Box	SelectMultiple default = true, toggle to unselect, OnChange fires on every selection
Radio	Updates to Radio	Item order preserved, View mode read-only (not disabled), new styling properties
Link	Updates to Link	URL security validation (XSS protection), FontWeight fix, Wrap fix
Info Button	Updates to Info Button	Content flyout expansion fixed, click reliability improved, AcceptsFocus removed
Button	Updates to Button	Six property renames, enum values for Appearance/Layout/IconStyle, new Tooltip and border properties, AcceptsFocus removed
Dropdown	Updates to Dropdown control	Fluent-themed flyout on desktop, DefaultSelectedItems renamed to Default, FontSize renamed to Size, enum format changes
Icon	Updates to Icon control	OnSelect support, TabIndex removed, Style renamed to IconStyle, new border and padding properties
Slider	Updates to Slider	Value renamed to Default, Layout renamed to LayoutDirection, enum values for Size and LayoutDirection, new Tooltip property

Property changes across modern controls

This section covers changes that affect multiple controls. If you only need to update one control, use the update guide linked in the table above.

Property renames

Many properties were renamed for consistency across controls. If your formulas reference the old names, you need to update them to the new names:

Expand table

Old name (Previous)	New name (New)	Affected controls
`FontColor`	`Color`	Text, Link, Info Button, Radio, Text Input, Tab List, Number Input, Date Picker, Combo Box,

Old name (Previous)	New name (New)	Affected controls
		Button
`FontSize`	`Size`	Text, Link, Info Button, Radio, Text Input, Tab List, Number Input, Date Picker, Combo Box, Button, Dropdown
`FontItalic`	`Italic`	Text, Link, Info Button, Radio, Text Input, Tab List, Date Picker, Combo Box, Button
`FontStrikethrough`	`Strikethrough`	Text, Link, Info Button, Radio, Text Input, Date Picker, Combo Box, Button
`FontUnderline`	`Underline`	Text, Link, Info Button, Radio, Text Input, Date Picker, Combo Box, Button
`Weight`	`FontWeight`	Text only
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`	All controls with borders

New enum patterns

Properties that previously accepted plain text strings (like `"Bold"` or `"Start"`) now require typed enum values (like `FontWeight.Bold` or `Align.Left`). If your formulas hardcode the old string values, you need to update them.

Using enums gives you IntelliSense autocomplete, compile-time validation, and localization safety in your Power Fx formulas. Each control's update guide lists the specific enum changes for that control.

Align

Used by: Text, Text Input, Number Input, Link, Tab List, Date Picker, Combo Box

Expand table

Enum value	Description
`Align.Left`	Aligns text to the left edge
`Align.Center`	Centers text horizontally
`Align.Right`	Aligns text to the right edge

Enum value	Description
`Align.Justify`	Stretches text to fill the full width (Text, Link only)

The previous string values `"Start"` or `""` (empty string) map to `Align.Left`. The previous value `"End"` maps to `Align.Right`.

VerticalAlign

Used by: Text, Link

Expand table

Enum value	Description
`VerticalAlign.Top`	Aligns text to the top of the control
`VerticalAlign.Middle`	Centers text vertically (default)
`VerticalAlign.Bottom`	Aligns text to the bottom of the control

The previous Text control defaulted to `""` (empty/undefined). The new default is `VerticalAlign.Middle`.

FontWeight

Used by: Text, Text Input, Number Input, Link, Tab List, Radio, Combo Box, Date Picker, Info Button

Expand table

Enum value	Description
`FontWeight.Bold`	Bold weight
`FontWeight.Semibold`	Semibold weight
`FontWeight.Normal`	Normal/regular weight (default)
`FontWeight.Lighter`	Lighter weight

> **Important**
>
> **FontWeight.Medium removed**: The `Medium` value isn't available in the new controls. If your previous control used `FontWeight.Medium`, it automatically maps to `FontWeight.Normal`

during the update.

The previous Text control property was named `Weight` with string values (`"Regular"`, `"Bold"`, and so on). The new property is `FontWeight` with enum values. Both the property name and the value format changed.

ValidationState

Used by: Date Picker, Text Input, Number Input, Combo Box, Dropdown, Radio

Expand table

Enum value	Description
`ValidationState.None`	No validation styling (default)
`ValidationState.Error`	Shows error styling (red border)

> **Note**
>
> Only two values are available: `None` and `Error`. There's no `Valid` or `Success` state.

Appearance

Used by: Date Picker, Text Input, Number Input, Combo Box, Dropdown

Expand table

Enum value	Description
`Appearance.Filled`	Filled background style
`Appearance.FilledDarker`	Darker filled background (default for most controls)
`Appearance.Outline`	Outlined border with transparent background

BorderStyle

Used by: Text, Text Input, Number Input, Date Picker, Combo Box, Tab List, Link, Radio, Icon, Dropdown

Expand table

Enum value	Description
`BorderStyle.Solid`	Solid line border (default)
`BorderStyle.Dashed`	Dashed line border
`BorderStyle.Dotted`	Dotted line border
`BorderStyle.None`	No border

IconStyle

Used by: Icon, Button

Expand table

Enum value	Description
`IconStyle.Outline`	Renders the icon in outline (unfilled) style (default)
`IconStyle.Filled`	Renders the icon in filled style

Replace the previous string values `"Outline"` and `"Filled"` with `IconStyle.Outline` and `IconStyle.Filled`.

BorderRadius split into corner-specific properties

Controls that previously had a single **BorderRadius** property now have four separate properties for each corner:

```
// Previous version
Control.BorderRadius = 8

// New version
Control.RadiusTopLeft = 8
Control.RadiusTopRight = 8
Control.RadiusBottomLeft = 8
Control.RadiusBottomRight = 8
```

This change gives you more flexibility to create controls with different corner radii, such as rounded top corners only.

OnChange behavior improvements

Several controls changed *when* the **OnChange** event fires. This is one of the most impactful changes — if your app logic depends on OnChange, review this table:

Expand table

Control	Previous OnChange	New OnChange	Impact
Text Input	Every keystroke	Focus out (blur) only	**CRITICAL** - Major performance improvement, might require formula updates
Number Input	Every keystroke	Focus out + step button clicks	Major performance improvement
Combo Box	Delayed/Focus out	Every selection/deselection	More responsive interaction
Radio	Sometimes inconsistent	Every selection reliably	Improved reliability

> **Tip**
>
> If you need per-keystroke updates (for example, live search), reference the control's output property (such as `TextInput1.Text`) directly in your formulas instead of relying on OnChange.

TriggerOutput changes

The **TriggerOutput** property controls when a control pushes its value to dependent formulas. Here's what changed:

Expand table

Control	Change	Impact
Text Input	Default changed from **FocusOut** to **Keypress**	Text updates immediately on keystroke (Delayed in forms)
Number Input	**TriggerOutput** property removed entirely	No longer available
Combo Box	No longer fires OnChange	TriggerOutput and OnChange are now independent

Control	Change	Impact
Text Input	No longer fires OnChange	TriggerOutput and OnChange are now independent

Improved mobile defaults

When you add an updated control to a mobile layout canvas, it automatically uses touch-friendly defaults:

- Larger widths and heights for easier tapping
- Larger font sizes (typically 24 pt)
- Appropriate padding and spacing

You don't need to do anything extra — these defaults apply automatically for new controls on mobile layouts.

Updated command bar and right-click menus

Updated controls include refreshed command bar and right-click menus in the Studio authoring experience, giving you quick access to common style properties like font, color, fill, alignment, and borders.

See also

- Modern controls overview
- Individual control update guides (see table in the section above)

Last updated on 04/20/2026

Badge control in Power Apps

Article • 03/19/2024

A badge is a visual decoration for UI elements.

Description

Badge can be used to display content in a visually better way. The key properties for this control are **Content**, **Shape**, and **Appearance**.

General

Content – The content to be displayed inside Badge control

Visible - Whether a control appears or is hidden.

Size and position

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

Appearance – A Badge can be filled, outline, ghost, inverted.

Shape – The shape of the badge can be circular, square, or rounded.

ThemeColor – The style of the badge has different visual representations based on the selected scenarios: brand, danger, important, informative, severe, subtle, success, or warning.

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color. This property is on in effect if the **Color** property has a value of **Brand**.

Font - The name of the family of fonts in which text appears.

FontSize - The font size of the text that appears on a control. If the value is null or zero, then the font size is driven by selected Fluent theme.

FontColor - The color of text in a control.

FontWeight - The weight of the text in a control: **Bold**, **Lighter**, **Normal**, or **Semibold**.

FontItalic - Whether the text in a control is italic.

FontUnderline - Whether a line appears under the text that appears on a control.

FontStrikethrough - Whether a line appears through the text that appears on a control.

Additional properties

AccessibleLabel – Label for screen readers.

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

Feedback

Was this page helpful? Yes No

Provide product feedback

Button modern control in canvas apps

The **Button** modern control provides a clickable element that triggers actions in a canvas app, offering flexible styling and comprehensive interaction capabilities.

Description

The **Button** modern control provides a clickable element that triggers app logic when selected. Use it for form submissions, navigation, confirmations, and any interaction that requires user input. The control supports icon and text combinations, full font styling, and multiple appearance styles through Fluent theming. Key properties for this control are **Text**, **Appearance**, and **OnSelect**.

> **Note**
>
> This article describes the updated Button modern control. For information about what changed from the previous version, see Recent updates.

General

Text – The label displayed on the button. Supports any text or Power Fx formula that evaluates to a string. Use this property to set or change the button text.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the button based on app state.

Behavior

OnSelect – Use the **OnSelect** property to define what happens when the button is selected. It's how the app responds when the user selects the button. The control is accessible: **OnSelect** also triggers when the user presses Enter or Space while the control has keyboard focus.

> **Tip**
>
> Common OnSelect actions include:
>
> - `Navigate(Screen2)` - Navigate to another screen

- `SubmitForm(Form1)` - Submit a form
- `Set(varName, value)` - Set a variable
- `Notify("Message", NotificationType.Success)` - Show a notification
- `Patch(DataSource, Record, Updates)` - Update data

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). When **Disabled**, the button is visually dimmed and **OnSelect** doesn't fire.

Getting started

The **Button** modern control is one of the most commonly used controls in Power Apps. At its core, a button has two essential elements: the text that users see (the **Text** property) and the action that happens when selected (the **OnSelect** property). Once you understand these basics, you can create interactive apps that respond to user input.

Set button text

To change what text appears on your button, set the **Text** property:

- **In the Properties pane**: Enter text directly in the **Text** field
- **Using a formula**: `Text = "Click Me"` or `Text = "Save " & TextInput1.Text`

Make buttons interactive

Define what happens when users selects the button using the **OnSelect** property:

1. Select the button in your app.
2. In the Properties pane, find the **OnSelect** property.
3. Enter your Power Fx formula, such as `Navigate(NextScreen)`.

Configure button properties

Power Apps uses a streamlined approach for editing controls.

- Select the button on the canvas:
  - **Use the Properties pane** on the right to modify basic properties.
  - **Use the formula bar** at the top for advanced formulas.
  - **Switch to Advanced properties** for detailed customization.

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **96**.

Height – Distance between the control's top and bottom edges. Default is **32**.

Align – The horizontal alignment of the button content within the control. Accepts `Align` enum values: `Align.Left`, `Align.Center`, `Align.Right`, `Align.Justify`.

VerticalAlign – The vertical alignment of the button content. Accepts `VerticalAlign` enum values: `VerticalAlign.Top`, `VerticalAlign.Middle`, `VerticalAlign.Bottom`.

PaddingTop – Distance between the button content and the top edge of the control.

PaddingBottom – Distance between the button content and the bottom edge of the control.

PaddingLeft – Distance between the button content and the left edge of the control.

PaddingRight – Distance between the button content and the right edge of the control.

Style and theme

Appearance – The visual style of the button. Accepts `ButtonAppearance` enum values:

Expand table

Value	Description
`ButtonAppearance.Primary`	Filled button using the brand color. Default.
`ButtonAppearance.Secondary`	Subtle filled style, secondary emphasis.
`ButtonAppearance.Outline`	Outlined button with no background fill.
`ButtonAppearance.Subtle`	Minimal styling, no border or fill.
`ButtonAppearance.Transparent`	No visible styling; blends with the background.

BasePaletteColor – The base color used by the theme to generate the control's color palette. Change this property to apply a different theme color to the button.

Icon – The Fluent icon name to display alongside the button text (for example, `"Add"`, `"Save"`, `"Delete"`). Leave blank for a text-only button.

IconStyle – Whether the icon renders in outline or filled style. Accepts `IconStyle` enum values: `IconStyle.Outline` (default) or `IconStyle.Filled`.

IconRotation – Rotation in degrees applied to the icon. Default is **0**.

Layout – The position of the icon relative to the button text. Accepts `ButtonLayout` enum values:

Expand table

Value	Description
`ButtonLayout.IconBefore`	Icon appears to the left of the text. Default.
`ButtonLayout.IconAfter`	Icon appears to the right of the text.
`ButtonLayout.IconOnly`	Only the icon is shown; no text label.

Font – The font family used for the button label.

Size – The font size of the button label, in points.

Color – The color of the button label text.

FontWeight – The weight (thickness) of the button label. Accepts `FontWeight` enum values: `FontWeight.Bold`, `FontWeight.Semibold`, `FontWeight.Normal` (default), `FontWeight.Lighter`.

Italic – Whether the button label appears in italic style.

Underline – Whether a line appears under the button label text.

Strikethrough – Whether a line appears through the middle of the button label text.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Additional properties

AccessibleLabel – Label read by screen readers. Provide a meaningful description when the button label alone isn't sufficient (for example, when using an icon-only button).

Tooltip – Explanatory text that appears when the user hovers over the button.

ContentLanguage – The display language for the control content, if different from the app language.

Example

The following YAML example shows a submit button and a cancel button with OnSelect actions:

YAML

```yaml
- SubmitButton:
    Control: ModernButton@1.0.0
    Properties:
      Text: ="Submit"
      OnSelect: |
        SubmitForm(DataForm1);
        Navigate(SuccessScreen)
      Appearance: =ButtonAppearance.Primary
      Icon: ="Checkmark"
      IconStyle: =IconStyle.Outline
      Layout: =ButtonLayout.IconBefore
      AccessibleLabel: ="Submit the form"
      Tooltip: ="Submit your response"
      Width: =120
      Height: =36

- CancelButton:
    Control: ModernButton@1.0.0
    Properties:
      Text: ="Cancel"
      OnSelect: =Navigate(HomeScreen)
      Appearance: =ButtonAppearance.Outline
      Width: =120
      Height: =36
```

Common button patterns

Here are typical button implementations:

Navigation button:

```powerfx
OnSelect = Navigate(DetailsScreen, ScreenTransition.Fade)
```

Data submission:

```powerfx
OnSelect = SubmitForm(Form1); Notify("Data saved!", NotificationType.Success)
```

Variable update:

```powerfx
OnSelect = Set(IsVisible, !IsVisible)
```

Confirmation dialog:

```powerfx
OnSelect = If(
    Confirm("Are you sure you want to delete this item?"),
    Remove(Collection1, ThisItem)
)
```

Recent updates

The updated version of the **Button** modern control includes the following improvements and behavior changes.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontSize`	`Size`
`FontColor`	`Color`
`FontItalic`	`Italic`
`FontUnderline`	`Underline`
`FontStrikethrough`	`Strikethrough`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Removed properties

Expand table

Removed property	Notes
`AcceptsFocus`	Removed. The modern Button control always accepts keyboard focus. No replacement needed.

Bug fixes and improvements

- **Updated enums**: `Appearance`, `Layout`, and `IconStyle` now use typed Power Fx enums (`ButtonAppearance`, `ButtonLayout`, `IconStyle`) instead of string values, improving IntelliSense and reducing formula errors.
- **Tooltip support**: New `Tooltip` property shows explanatory text on hover.
- **Border improvements**: Added `BorderStyle` and `BorderThickness` for more precise border control. Four corner-specific properties replace `BorderRadius` for independent corner control.
- **Full font control**: Font properties are now consistent with other modern controls. Use `Font`, `Size`, `Color`, `Italic`, `Underline`, and `Strikethrough`.

See also

- Modern controls overview
- Size and location properties

Last updated on 04/20/2026

Card modern control

Cards help makers create clean, responsive object summaries such as contact tiles, document previews, or product listings in either horizontal or vertical orientation.

Description

Use a Card when you want to show a concise overview of an item with optional imagery and text.
The control adapts its layout to the sections a maker chooses to configure, enabling consistent patterns without requiring multiple independent controls.

General

Direction – Determines whether the card layout is vertical or horizontal

Title – Header title text displayed on the card

Subtitle – Supplemental text displayed under or beside the title in the header

Description – Description text displayed below the header of the card

Image – Main preview image for the card

ImageAltText – Alt text for the preview image for accessibility

HeaderImage – Small avatar or icon displayed in the header with the title and subtitle

HeaderImageAltText – Alt text for the header image

ImagePosition – Determines whether the preview image appears before or after the header section

Behavior

OnSelect – Action triggered when the user selects the card

DisplayMode – Controls whether the card is interactive, view-only, or disabled

Visible – Shows or hides the card

Style and theme

Radius – Controls corner roundness of the card container

DropShadow – Defines the shadow style applied to the card

Size and position

X, Y – Position of the control on the canvas

Width, Height – Dimensions of the control

Examples

The following examples show three Card configurations you can copy and paste directly into your app configuration.

Vertical card with description

YAML

```yaml
- VerticalCardwithDescription:
    Control: ModernCard@1.0.0
    Properties:
      Description: ="This is a sample description"
      Height: =240
      Width: =329
      X: =50
      Y: =45
```

Horizontal card with long description and no image

YAML

```yaml
- HorizontalCardwithLongDescriptionNoImage:
    Control: ModernCard@1.0.0
    Properties:
      Description: ="This is a long description text. It provides a rich, narrative
overview designed to help users understand the purpose, value, and key capabilities
of the experience. This section typically expands beyond a short summary by
offering contextual background, explaining how the feature solves real user
problems, and detailing what makes it uniquely impactful."
      Height: =240
      Image: =Blank()
      LayoutDirection: =LayoutDirection.Horizontal
      Width: =300
```

```
      X: =40
      Y: =40
```

Vertical card with header on top

YAML

```
- VerticalCardwithHeaderOnTop:
    Control: ModernCard@1.0.0
    Properties:
      HeaderImage: =Blank()
      Height: =240
      ImagePlacement: =ImagePlacement.AfterHeader
      Subtitle: ="This is subtitle which can take whole of single line."
      Title: ="Title for the card"
      Width: =350
      X: =40
      Y: =40
```

Last updated on 03/19/2026

Checkbox modern control in Power Apps

Article • 07/17/2024

A control that the user can select or clear to set its value to **true** or **false**.

Description

The user can specify a Boolean value by using this familiar control that is used in Graphical User Interface (GUI) for decades. The key properties for this control are **Checked**, **Label**, **OnCheck**, and **OnUncheck**.

General

Label – The Checkbox's label.

AccessibleLabel – Label for screen readers.

Visible - Whether a control appears or is hidden.

Behavior

Checked – The controlled value for the checkbox.

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

Size and position

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color.

Font - The name of the family of fonts in which text appears.

FontSize - The font size of the text that appears on a control. If the value is null or zero, then the font size is driven by selected Fluent theme.

FontColor - The color of text in a control.

FontWeight - The weight of the text in a control: Bold, Lighter, Normal, or Semibold.

FontItalic - Whether the text in a control is italic.

FontUnderline - Whether a line appears under the text that appears on a control.

FontStrikethrough - Whether a line appears through the text that appears on a control.

Additional properties

OnCheck - Actions to perform when the user checks the control.

OnSelect - Actions to perform when the user selects a control.

OnUncheck - Actions to perform when the user unchecks the control.

Feedback

Was this page helpful? Yes No

Provide product feedback

Combo box modern control in canvas apps

A searchable, multiselect dropdown control for selecting one or more items from a list.

Description

The **Combo box** modern control provides a flexible, searchable dropdown interface that enables users to select one or more items from a list of options. This control combines the functionality of a traditional dropdown with advanced search capabilities, making it ideal for scenarios where users need to choose from large datasets or multiple selections.

Key properties for this control are **Items**, **DefaultSelectedItems**, and **SelectMultiple**.

General

Items – The data source that contains the items shown in the combo box. Can be a collection, table, or Power Fx formula that returns a table.

ItemDisplayText – A formula that determines what text to display for each item in the dropdown list. Use `ThisItem` to reference the current item. Example: `ThisItem.DisplayName` or `ThisItem.Value1`.

DefaultSelectedItems – The initial items selected in the control before the user makes a change. Must be a table of records from the **Items** data source.

InputTextPlaceholder – Hint text that appears in the input field when no items are selected. Default is "**Find items**".

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

SelectMultiple – Whether users can select multiple items. When `true`, users can select multiple items and each selected item can be toggled on or off by clicking it again. When `false`, only one item can be selected at a time. Default is **true**.

IsSearchable – Whether the search box is enabled. When `true`, users can type to filter the list. When `false`, the control acts as a simple dropdown without search. Default is **true**.

OnChange – How the app responds when the user selects or deselects an item. This event fires immediately on every selection or deselection, making it ideal for real-time filtering or dependent dropdowns.

AllowMultipleSelection – (Display property) A toggle that controls whether users can select multiple items. Maps to the **SelectMultiple** property.

AllowSearching – (Display property) A toggle that controls whether search is enabled. Maps to the **IsSearchable** property.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **View** mode, the selection is shown but can't be changed.

DelayOutput – When set to `true`, delays updates to the **SearchText** output property until the user pauses typing. When `false`, **SearchText** updates immediately on each keystroke. This property does not affect when **OnChange** fires — **OnChange** fires immediately on every selection and deselection regardless of this setting. Default is **false**.

Data

SelectedItems – (Output) A table of records representing the items currently selected by the user. Use this property to reference the user's selections in formulas.

Selected – (Output) When **SelectMultiple** is `false`, returns the single selected item as a record. When **SelectMultiple** is `true`, returns the first selected item.

SearchText – (Output) The current text typed by the user in the search box. Use this to create custom filtering logic or to track search behavior.

> ⓘ **Note**
>
> Output properties such as **SelectedItems** in a Combo box control return a table of the currently selected records and can’t be set directly. Instead, other controls and formulas use this property to access the current selection. For example, reference `ComboBoxName.SelectedItems` in a label, gallery, or formula to work with the selected records.

MultiValueDelimiter – The separator character used when displaying multiple selected items in the input field. Default is **", "** (comma and space).

Validation

Required – Whether the control must have at least one item selected before the form can be submitted. When `true`, the control shows a validation error if empty.

ValidationState – The current validation state of the control. Accepts `ValidationState` enum values:

Expand table

Value	Description
`ValidationState.Error`	The control value has an error
`ValidationState.None`	No validation applied (default)

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **320**.

Height – Distance between the control's top and bottom edges. Default is **32**.

PaddingTop – Distance between the text and the top edge of the control.

PaddingBottom – Distance between the text and the bottom edge of the control.

PaddingLeft – Distance between the text and the left edge of the control.

PaddingRight – Distance between the text and the right edge of the control.

Style and theme

Appearance – The visual style of the control. Accepts `Appearance` enum values:

Expand table

Value	Description
`Appearance.Filled`	Filled background style
`Appearance.FilledDarker`	Filled with darker background
`Appearance.Outline`	Outline style with border

Default is `Appearance.FilledDarker`.

BasePaletteColor – The base color used by the theme to generate the control's color palette. Changes this property to apply a different theme color to the control.

Font – The name of the font family in which text appears.

Size – The font size of the text, in points. Default is **14**.

Color – The color of the text in the control.

FontWeight – The weight (thickness) of the text. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the text appears in italic style.

Underline – Whether a line appears under the text.

Strikethrough – Whether a line appears through the middle of the text.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Example

The following YAML example shows combo box controls with single and multiselect modes:

YAML

```
- DepartmentComboBox:
    Control: ModernCombobox@1.0.0
    Properties:
      Items: =["Engineering", "Sales", "Marketing", "Support"]
      ItemDisplayText: =ThisItem.Value
      InputTextPlaceholder: ="Select a department..."
      SelectMultiple: =false
      IsSearchable: =true
      Required: =true
      X: =40
      Y: =100
      Width: =320
      Height: =32

- TeamMemberComboBox:
    Control: ModernCombobox@1.0.0
    Properties:
      Items: =["Alice Smith", "Bob Johnson", "Carol Williams", "David Brown"]
      ItemDisplayText: =ThisItem.Value
      InputTextPlaceholder: ="Select team members..."
      SelectMultiple: =true
      IsSearchable: =true
      MultiValueDelimiter: ="; "
      X: =40
      Y: =150
      Width: =320
      Height: =32
```

Accessibility

AccessibleLabel – A label for screen readers to announce what the control is for. Describe the purpose of the combo box, not the current selection.

TabIndex – The tab order of the control relative to other controls. Controls with lower **TabIndex** values receive focus before controls with higher values.

Recent updates

The updated version of the **Combo box** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Enum format changes

The following properties now require typed enum syntax instead of plain string values:

Expand table

Property	Previous format	New format
`Appearance`	`"FilledDarker"`	`Appearance.FilledDarker`
`ValidationState`	`"None"`	`ValidationState.None`

Behavioral changes

- **SelectMultiple defaults to true**: The **SelectMultiple** property now defaults to `true`. If your app requires single selection, explicitly set `SelectMultiple = false`.
- **Toggle selection behavior**: Clicking a selected item now unselects it, making it easier to remove selections without clearing the entire list.
- **Immediate OnChange firing**: The **OnChange** event now fires immediately on every selection or deselection, enabling real-time filtering and dependent dropdowns. Note: **DelayOutput** controls when **SearchText** updates — it does not affect **OnChange** timing.
- **Search text clears on selection**: When an item is selected, the search text automatically clears.
- **Reset() includes SearchText**: The `Reset()` function now clears both the selection and the search text.

Removed and replaced properties

Expand table

Removed property	Replacement	Notes
`TriggerOutput`	`DelayOutput`	`TriggerOutput = "Keypress"` → `DelayOutput = false`. `TriggerOutput = "FocusOut"` → `DelayOutput = true`.
`Fields`	`ItemDisplayText`	Replace `Fields = ["FieldName"]` with `ItemDisplayText = ThisItem.FieldName`.

Other improvements

- **Improved data handling**: Better support for Dataverse many-to-one relationships and increased item limit handling (previously limited to about 800 items).
- **Mobile-optimized defaults**: When adding the control in a mobile layout, defaults are automatically adjusted (font size 24, width 560, height 64).

Limitations

- The control works best with lists that have up to several thousand items. For very large datasets (more than 10,000 items), consider using server-side filtering by using the **SearchText** output property.
- In the current version, chevron and checkbox icons don't scale with font size.

See also

- Modern controls overview
- Size and location properties

Last updated on 04/15/2026

Date picker modern control in canvas apps

Summarize this article for me

The **Date picker** modern control provides an intuitive calendar interface for date selection in canvas apps. This control combines visual calendar navigation with flexible formatting options to create a seamless date input experience for users.

Description

Use this control when you need users to select specific dates, such as appointment scheduling, event planning, or filtering data by date ranges. The control supports customizable date formats, timezone handling, and date range restrictions to meet diverse application requirements.

Key capabilities include:

- Visual calendar interface with month and year navigation
- Flexible date formatting using predefined formats or custom patterns
- Date range validation to restrict selectable dates
- Timezone support for local or UTC display
- Accessibility features for screen readers and keyboard navigation

The control outputs the selected date through the SelectedDate property, which you can use in formulas throughout your app.

Key properties for this control are **SelectedDate**, **Format**, and **DateTimeZone**.

General

Placeholder – Hint text that appears when no date is selected. Default is **"Select a date"**.

DefaultDate – The initial date selected in the control before the user makes a change. Use this property to set a default date when the picker first loads.

Format – The display format for the selected date. Accepts `DatePickerFormat` enum values or a custom format string:

Enum values:

Expand table

Value	Description	Example
`DatePickerFormat.LongAbbreviated`	Abbreviated month with full year (default)	Jan 15, 2024
`DatePickerFormat.Short`	Short numeric format	1/15/2024
`DatePickerFormat.Long`	Full month name with year	January 15, 2024

Custom format strings:

You can also use custom date format patterns as strings to create specific date representations:

Expand table

Format string	Example output
`"dd-mm-yyyy"`	15-01-2024
`"dd-mm"`	15-01
`"yyyy-mm-dd"`	2024-01-15
`"dd/mm/yyyy"`	15/01/2024
`"mm/dd/yyyy"`	01/15/2024

Use custom format strings when you need a specific date representation not available in the enum values. Example: `Format = "dd-mm-yyyy"`.

DateTimeZone – The timezone used for date display and storage. Accepts `DateTimeZone` enum values:

Expand table

Value	Description
`DateTimeZone.Local`	User's local timezone (default)
`DateTimeZone.UTC`	Coordinated Universal Time

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

OnChange – How the app responds when the user selects a different date. This event fires when the date selection changes.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **View** mode, the selected date is shown but the calendar can't be opened.

IsEditable – Whether the user can type a date directly into the input field or must use the calendar picker. When `false`, users must select the calendar icon to choose a date. Default is **false**.

Date range

StartDate – The earliest date that can be selected in the calendar. Dates before this date are disabled. Default is **Date(1900, 1, 1)**.

EndDate – The latest date that can be selected in the calendar. Dates after this date are disabled. Default is **Date(Year(Today())+100, 12, 31)**.

StartOfWeek – The day that appears as the first column in the calendar view. Accepts `StartOfWeek` enum values:

Expand table

Value	Description
`StartOfWeek.Sunday`	Week starts on Sunday (default)
`StartOfWeek.Monday`	Week starts on Monday
`StartOfWeek.Tuesday`	Week starts on Tuesday
`StartOfWeek.Wednesday`	Week starts on Wednesday
`StartOfWeek.Thursday`	Week starts on Thursday
`StartOfWeek.Friday`	Week starts on Friday
`StartOfWeek.Saturday`	Week starts on Saturday

Data

SelectedDate – (Output) The date currently selected by the user. Returns a date value that you can use in formulas. Returns `Blank()` if no date is selected.

Validation

ValidationState – The current validation state of the control. Accepts `ValidationState` enum values:

Expand table

Value	Description
`ValidationState.Error`	The control value has an error
`ValidationState.None`	No validation applied (default)

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **320**.

Height – Distance between the control's top and bottom edges. Default is **32**.

PaddingTop – Distance between the text and the top edge of the control.

PaddingBottom – Distance between the text and the bottom edge of the control.

PaddingLeft – Distance between the text and the left edge of the control.

PaddingRight – Distance between the text and the right edge of the control.

Style and theme

Appearance – The visual style of the control. Accepts `Appearance` enum values:

Expand table

Value	Description
`Appearance.Filled`	Filled background style
`Appearance.FilledDarker`	Filled with darker background (default)

Value	Description
`Appearance.Outline`	Outline style with border

BasePaletteColor – The base color used by the theme to generate the control's color palette. Change this property to apply a different theme color to the control.

Font – The name of the font family in which text appears.

Size – The font size of the text, in points. Default is **0** (uses theme default).

Color – The color of the text in the control.

FontWeight – The weight (thickness) of the text. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the text appears in italic style.

Underline – Whether a line appears under the text.

Strikethrough – Whether a line appears through the middle of the text.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Accessibility

AccessibleLabel – A label for screen readers to announce what the control is for. It should describe the purpose of the date picker, not the current selection.

ContentLanguage – The language used for formatting dates and calendar display. The app settings provide the value if you don't specify it.

Example

The following YAML example shows date picker controls for selecting dates:

YAML

```yaml
- StartDatePicker:
    Control: ModernDatePicker@1.0.0
    Properties:
      Placeholder: ="Select start date"
      DefaultDate: =Today()
      StartDate: =Today()
      EndDate: =Today() + 365
      StartOfWeek: =StartOfWeek.Monday
      X: =40
      Y: =100
      Width: =280
      Height: =32

- EndDatePicker:
    Control: ModernDatePicker@1.0.0
    Properties:
      Placeholder: ="Select end date"
      Format: =DatePickerFormat.Short
      DefaultDate: =Today() + 7
      StartOfWeek: =StartOfWeek.Monday
      X: =340
      Y: =100
      Width: =280
      Height: =32
```

Recent updates

The updated version of the **Date picker** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Removed properties

Expand table

Removed property	Notes
`Required`	Use `ValidationState` instead to control validation styling.

Bug fixes and improvements

- **DisplayMode.View properly enforced**: In **View** mode, the control is now properly read-only and the calendar can't be opened. Previously, the picker was still editable in View mode.
- **Format property fully honored**: All format values are now properly applied. Previously, certain format values were ignored until another change was made.
- **DateTimeZone respected**: The **DateTimeZone** property now correctly applies to date display and storage. Previously, this property value was sometimes ignored.
- **Blank value handling**: When the control is in **View** mode with a blank value, it no longer incorrectly shows a previous value.
- **Mobile calendar improvements**: Calendar flyout is now properly sized for mobile devices, with optimized defaults (width 560, height 64, font size 24).
- **Month/Year flyout positioning**: Fixed issue where the month/year selector went off-screen on smaller displays.

- **Gallery and data card stability**: Resolved crashes that occurred when adding the date picker to galleries or data cards.
- **Color palette inheritance**: The calendar flyout now properly respects the **BasePaletteColor** override.
- **Command bar integration**: The authoring command bar and right-click menu now show Font, Font Size, Font color, and Background color for quick access.
- **DefaultDate property**: New property allows setting an initial date when the picker first loads.

Limitations

- The calendar icon doesn't scale with font size in the current version.
- On mobile devices, the control shows the Fluent calendar view instead of the native mobile date picker.

See also

- Modern controls overview
- Size and location properties

Last updated on 02/24/2026

Dropdown modern control in canvas apps

Let users pick a single value from a collapsible list.

Description

The **Dropdown** modern control presents a list of choices in a compact, collapsible format. Users select one item and the list collapses back to a single line. Use the **Dropdown** control when you need users to choose one option from a list of more than seven items, for required fields with a fixed set of choices, or for forms where screen space is limited. For fewer options (two to seven), consider using a Radio group instead. If users need to select multiple items, use a Combo box.

Key properties for this control are **Items**, **Default**, **ValidationState**, and **OnChange**.

> **Note**
>
> This article describes the updated Dropdown modern control. For information about what changed from the previous version, see Recent updates.

On desktop, the updated control opens a **Fluent-themed flyout** that matches the app's visual theme. The previous version opened the browser's HTML `<select>` element, which couldn't be themed.

General

Items – The source of data that contains the items that appear in the control. Accepts a table, a collection, or an inline array (for example, `["Option A", "Option B", "Option C"]`).

ItemDisplayText – A Power Fx formula evaluated for each item in **Items** to determine the text shown in the list. Use `ThisItem` to reference the current item. Defaults to `ThisItem.Value` for string arrays and `ThisItem.Value1` for table data sources. Change this when your data source has a specific column to display (for example, `ThisItem.Name`).

Default – The initial selected value before the user makes a change. Set this to a value that exists in **Items** (for example, `="Option A"`). Leave blank for no default selection.

AccessibleLabel – Label read by screen readers. Describe the purpose of the dropdown (for example, `"Select a country"`).

Visible – Whether the control appears or is hidden.

Behavior

OnChange – Actions to perform when the user changes the selection. Use `Self.Selected.Value` inside the formula to read the newly selected item. For example, `Notify("You selected " & Self.Selected.Value)` displays a message each time the selection changes.

Required – Whether the user must make a selection. When **true**, an indicator appears beside the dropdown. Default is **false**.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

ValidationState – Applies validation styling to the control. Accepts `ValidationState` enum values:

Expand table

Value	Description
`ValidationState.None`	No validation styling. Default.
`ValidationState.Error`	Highlights the control border in red to indicate an error.

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **320**.

Height – Distance between the control's top and bottom edges. Default is **32**.

PaddingTop – Distance between the control content and the top edge of the control.

PaddingBottom – Distance between the control content and the bottom edge of the control.

PaddingLeft – Distance between the control content and the left edge of the control.

PaddingRight – Distance between the control content and the right edge of the control.

Style and theme

Appearance – The visual style of the dropdown field. Accepts `Appearance` enum values:

Expand table

Value	Description
`Appearance.FilledDarker`	Filled with a slightly darker background. Default.
`Appearance.Filled`	Filled with a standard background.
`Appearance.Outline`	Outlined border with a transparent background.

BasePaletteColor – The base color used by the theme to generate the control's color palette. Use this value to apply a different theme color to the control.

Font – The font family used for the dropdown text.

Size – The font size of the dropdown text, in points. Default is **14**.

Color – The color of the dropdown text.

FontWeight – The weight (thickness) of the dropdown text. Accepts `FontWeight` enum values: `FontWeight.Bold`, `FontWeight.Semibold`, `FontWeight.Normal` (default), `FontWeight.Lighter`.

Italic – Whether the dropdown text appears in italic style.

Underline – Whether a line appears under the dropdown text.

Strikethrough – Whether a line appears through the middle of the dropdown text.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Additional properties

ContentLanguage – The display language for the control content, if different from the app language.

Output properties

Selected – The currently selected item, returned as a record. For a string array, use `Dropdown1.Selected.Value` to read the selected text. For a table data source, use `Dropdown1.Selected.<ColumnName>`.

> ⓘ Note
>
> **Selected** is an output property. You can reference it in other controls and formulas, but you can't set it directly.

Examples

Basic dropdown with inline options

This example creates a required country dropdown with inline validation. When the user hasn't selected a value, the control shows an error state.

YAML

```
- CountryDropdown:
    Control: ModernDropdown@1.0.0
    Properties:
      AccessibleLabel: ="Select your country"
      Default: ={Value:"United States"}
      ItemDisplayText: =ThisItem.Value
      Items: =["Australia", "Canada", "United Kingdom", "United States"]
      Required: =true
      ValidationState: =If(IsBlank(CountryDropdown.Selected.Value),
ValidationState.Error, ValidationState.None)
      Width: =240
```

Dropdown bound to a data source

This example binds the dropdown to a table data source and displays the **Name** column. The **OnChange** action stores the selected record in a variable.

YAML

```
- DepartmentDropdown:
    Control: ModernDropdown@1.0.0
    Properties:
      AccessibleLabel: ="Select a department"
      Appearance: =Appearance.Outline
      ItemDisplayText: =ThisItem.Name
      Items: =Departments
      OnChange: =Set(varSelectedDept, Self.Selected)
      Width: =320
```

Recent updates

The updated version of the **Dropdown** modern control includes the following improvements and behavior changes.

Property renames

Expand table

Previous property	New property	Notes
`DefaultSelectedItems`	`Default`	Set the initial selection through `Default`. Change any formula referencing `Dropdown1.DefaultSelectedItems` to `Dropdown1.Default`.
`FontSize`	`Size`	Renamed to match other modern controls. Default changed from **0** (inherits from theme) to **14**.
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`	Split into four independent corner radius properties for greater styling flexibility.

Enum format changes

Expand table

Property	Previous value	New value
`Appearance`	`"FilledDarker"`	`Appearance.FilledDarker`
`Appearance`	`"Filled"`	`Appearance.Filled`
`Appearance`	`"Outline"`	`Appearance.Outline`
`ValidationState`	`"None"`	`ValidationState.None`
`ValidationState`	`"Error"`	`ValidationState.Error`

Bug fixes and improvements

- **Fluent-themed flyout**: On desktop, the dropdown list now opens as a Fluent-themed flyout that matches the app's visual design. The previous version used the browser's HTML `<select>` element, which couldn't be themed. On mobile, both versions use the device-native picker and it remains unchanged.
- **ItemDisplayText**: New property that controls which field of a data source record to show in the list. For string arrays, this property defaults automatically. For table data sources, set this property to the column you want to display (for example, `ThisItem.Name`).
- **Full font control**: New `Font`, `Color`, `FontWeight`, `Italic`, `Underline`, and `Strikethrough` properties give full text styling control, consistent with other modern controls.
- **Updated enums**: `Appearance` and `ValidationState` now use typed Power Fx enums, improving IntelliSense and reducing formula errors.

Accessibility

AccessibleLabel – Set this property to a meaningful description so screen readers can announce the purpose of the dropdown (for example, `"Select your country"`).

Required – When set to **true**, screen readers announce the field as required.

Keyboard users can press **Enter** or **Space** to open the dropdown, use **Arrow** keys to move between items, and press **Enter** to confirm a selection.

Limitations

- The Fluent-themed flyout is available on desktop only. On mobile, the control uses the device-native picker.
- Very large data sources may cause the dropdown list to load slowly. Filter or limit the data where possible.

See also

- Modern controls overview
- Size and location properties

Last updated on 04/20/2026

Header modern control in Power Apps

Article • 03/19/2024

A control that creates a modern app header.

Description

The **Header** control is a fundamental control that lets you design a user interface for a common app with ease. With minimal configuration, this control responds to themes that use modern theming and dynamic responsiveness, adjusting to the size of the parent container or app. The **Header** control supports a logo, a page title, and displays the user picture, while providing various unique style options. The key properties for this control are **Logo**, **OnSelectLogo**, and **Title**.

General

Title – The title label displayed in the header. Default value is the current screen name.

IsTitleVisible – Whether the title is displayed in the header.

Logo – The image displayed in the header.

IsLogoVisible – Whether the logo image is displayed.

IsProfilePictureVisible – Whether the user picture is displayed in the header. By default, a picture of the current user using the app is shown.

Visible - Whether a control appears or is hidden.

Size and position

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color. This property has no effect if the **ThemeStyle** property is set to **Neutral**.

Style – Header color variant, based on theme colors.

Font - The name of the family of fonts in which text appears.

TitleFontSize - The font size of the title that appears on a control. If the value is null or zero, then the font size is driven by selected Fluent theme.

FontColor - The color of text in a control.

Additional properties

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

LogoMaxHeight – The maximum height the logo can be placed within the control.

LogoToolTip - Tooltip for the logo.

OnSelectLogo - Actions to perform when the user selects the logo.

Unsupported capabilities

The following features aren't supported:

- Customization of colors
- App navigation

> ⓘ **Note**
>
> This list will change over time as we're working to support these capabilities.

The header doesn't automatically replicate across app screens, so it needs to be manually added to each screen.

Icon modern control in canvas apps

Displays a Fluent icon that can optionally respond to user interaction.

Description

The **Icon** modern control displays a Fluent icon from the Fluent icon library. You can use it in two ways:

- **Decorative** – Display a static icon as a visual indicator (for example, a status badge or label prefix). Leave the **OnSelect** property empty.
- **Interactive** – Respond to user input. Set the **OnSelect** property to a formula, and the control automatically renders with button semantics so it works with keyboard navigation and screen readers.

Common uses include toolbar actions, status indicators, inline navigation, and lightweight call-to-action elements.

The key properties for this control are **Icon**, **IconStyle**, and **OnSelect**.

> **Note**
>
> This article describes the updated Icon modern control. For information about what changed from the previous version, see Recent updates.

Add an icon to your app

1. Open your canvas app for editing.
2. On the left pane, select **Insert** (+).
3. Expand **Modern controls** and choose **Icon**.
4. In the property pane on the right, set the **Icon** property to a Fluent icon name such as `"Add"`, `"Delete"`, or `"Edit"`.
5. (Optional) Set **OnSelect** to a formula to make the icon interactive.

General

Icon – The name of the Fluent icon to display (for example, `"Add"`, `"Delete"`, `"Edit"`). Accepts any valid Fluent icon name as a string.

Visible – Whether the control appears or is hidden.

Behavior

OnSelect – Actions to perform when the user selects the icon. When this property contains an active formula, the icon renders with button semantics for keyboard and screen reader accessibility.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **32**.

Height – Distance between the control's top and bottom edges. Default is **32**.

Rotation – The angle in degrees to rotate the icon clockwise. Default is **0**.

PaddingTop – Distance between the icon and the top edge of the control.

PaddingBottom – Distance between the icon and the bottom edge of the control.

PaddingLeft – Distance between the icon and the left edge of the control.

PaddingRight – Distance between the icon and the right edge of the control.

Style and theme

IconStyle – Whether the icon renders in outline or filled style. Accepts `IconStyle` enum values:

Expand table

Value	Description
`IconStyle.Outline`	Renders the icon in outline (unfilled) style. Default.
`IconStyle.Filled`	Renders the icon in filled style.

IconColor – The color of the icon.

BasePaletteColor – The base color used by the theme to generate the control's color palette. Use this value to apply a different theme color to the icon.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Accessibility

AccessibleLabel – Label read by screen readers. When you set **OnSelect**, always provide a meaningful description of the action (for example, `"Delete item"`). Because an icon has no visible text, this label is the only way assistive technology can communicate the control's purpose.

> **Tip**
>
> For decorative icons that don't perform an action, you can leave **AccessibleLabel** empty so screen readers skip the control.

Additional properties

Tooltip – Explanatory text that appears when the user hovers over the control.

ContentLanguage – The display language for the control content, if different from the app language.

Examples

Interactive delete icon

This example shows a filled red delete icon that performs an action when selected. The `AccessibleLabel` and `Tooltip` properties ensure the control is usable with keyboard, screen readers, and mouse hover.

YAML

```yaml
- DeleteIcon:
    Control: ModernIcon@1.1.0
    Properties:
      AccessibleLabel: ="Delete this item"
      Icon: ="Delete"
      IconColor: =RGBA(176, 0, 32, 1)
      IconStyle: =IconStyle.Filled
      OnSelect: =Remove(MyDataSource, ThisItem)
      Tooltip: ="Delete this item"
```

Decorative status icon

This example shows a non-interactive icon used as a visual status indicator. Because `OnSelect` isn't set, the icon renders as a decorative element.

YAML

```yaml
- StatusIcon:
    Control: ModernIcon@1.1.0
    Properties:
      Icon: ="CheckmarkCircle"
      IconColor: =RGBA(16, 124, 16, 1)
      IconStyle: =IconStyle.Filled
      Width: =24
      Height: =24
```

Formula examples

Change the icon based on a condition:

Power Apps

```
If(ThisItem.Status = "Complete", "CheckmarkCircle", "Clock")
```

Toggle between outline and filled styles:

Power Apps

```
If(ThisItem.IsSelected, IconStyle.Filled, IconStyle.Outline)
```

Recent updates

The updated version of the **Icon** modern control includes the following improvements and behavior changes.

Property renames

Expand table

Previous property	New property	Notes
`Style`	`IconStyle`	Renamed for clarity; also changed from string to typed enum.

Removed properties

Expand table

Removed property	Notes
`TabIndex`	Removed. The control automatically handles focus management.

Bug fixes and improvements

- **OnSelect support**: The icon now triggers actions when selected. When you set `OnSelect` to an active formula, the control renders with button semantics, making it fully accessible via keyboard and screen readers.
- **AccessibleLabel**: New property for screen readers - especially important for interactive icons where the glyph alone doesn't convey meaning.
- **Updated enum**: `IconStyle` now uses a typed Power Fx enum (`IconStyle.Outline`, `IconStyle.Filled`) instead of a plain string, improving IntelliSense and reducing formula errors.

- **Full border and fill control**: New `Fill`, `BorderColor`, `BorderStyle`, `BorderThickness`, and four corner radius properties (`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`) provide complete styling options.
- **Padding support**: New `PaddingTop`, `PaddingBottom`, `PaddingLeft`, `PaddingRight` properties control spacing between the icon glyph and the control boundary.
- **BasePaletteColor**: Apply a custom theme color to the icon.

Best practices

- **Always set AccessibleLabel for interactive icons.** Icons don't have visible text, so the accessible label is essential for screen reader users.
- **Use Tooltip for discoverability.** A tooltip helps sighted users understand what an icon does when they hover over it.
- **Choose Filled vs. Outline intentionally.** Use `IconStyle.Filled` for active or selected states and `IconStyle.Outline` for default states to give users a visual cue.
- **Keep icon sizes consistent.** Use the same **Width** and **Height** values for icons that appear together (for example, in a toolbar) so the layout looks uniform.

See also

- Modern controls overview
- Size and location properties

Last updated on 04/20/2026

Info button modern control in canvas apps

Summarize this article for me

An interactive information icon that displays detailed content in a flyout when clicked or hovered.

Description

The **Info button** control provides contextual help and additional information in your canvas apps. When users select or hover over this interactive information icon, it displays detailed content in a flyout panel without disrupting their workflow.

Use this control to:

- Provide contextual help for form fields and complex features
- Display tooltips with additional guidance
- Offer explanatory content without cluttering your app interface
- Enhance user experience with on-demand information

The **Info button** appears as a standard information icon and supports rich text content, customizable styling, and accessibility features.This modern control integrates seamlessly with your app's design system through theme-based colors and flexible positioning options.

Key properties for this control are **Content**, **IconColor**, and **OnSelect**.

General

Content – The information text displayed in the flyout when the user clicks or hovers over the info button icon. Supports plain text and formatted content.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

OnSelect – How the app responds when the user clicks the info button. Use this property to trigger extra actions beyond showing the flyout, such as logging analytics or updating variables.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **View** mode, the icon is shown but the flyout can't be opened.

Size and position

X – Distance between the left edge of the control and the left edge of its parent container. If there's no parent container, the screen acts as the parent.

Y – Distance between the top edge of the control and the top edge of its parent container. If there's no parent container, the screen acts as the parent.

Width – Distance between the control's left and right edges. The default value is **32**.

Height – Distance between the control's top and bottom edges. The default value is **32**.

PaddingTop – Distance between the icon and the top edge of the control.

PaddingBottom – Distance between the icon and the bottom edge of the control.

PaddingLeft – Distance between the icon and the left edge of the control.

PaddingRight – Distance between the icon and the right edge of the control.

Style and theme

BasePaletteColor – The base color that the theme uses to generate the control's color palette. Change this property to apply a different theme color to the control.

Font – The name of the font family used for text in the flyout content.

Size – The font size of text in the flyout, in points. The default is **15**.

Color – The color of text in the flyout content.

FontWeight – The weight (thickness) of text in the flyout. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the text in the flyout appears in italic style.

Underline – Whether a line appears under the text in the flyout.

Strikethrough – Whether a line appears through the middle of the text in the flyout.

IconColor – The color of the information icon button itself. Use this value to match your app's theme or draw attention to important information.

Accessibility

AccessibleLabel – A label for screen readers to announce what the info button provides information about. It should describe the purpose or topic of the information, not just "info button".

ContentLanguage – The language used for the flyout content. If you don't specify this property, the app settings provide the language.

Example

The following YAML example shows contextual help buttons for form fields with conditional styling and tooltip content:

YAML

```yaml
- EmailInfoButton:
    Control: ModernInformationButton@1.0.0
    Properties:
      Content: ="We'll use this email to send you order confirmations and account
notifications. Your email is never shared with third parties."
      IconColor: =Color.Blue
      AccessibleLabel: ="Email address information"
      X: =200
      Y: =100
      Width: =28
      Height: =28

- PasswordInfoButton:
    Control: ModernInformationButton@1.0.0
    Properties:
      Content: '="Password requirements: At least 8 characters, one uppercase
letter, one number, and one special character (!@#$%)"'
      IconColor: =Color.Orange
      AccessibleLabel: ="Password requirements"
      X: =200
      Y: =150
      Width: =28
      Height: =28

- GeneralInfoButton:
```

```
    Control: ModernInformationButton@1.0.0
    Properties:
      Content: ="Click the info icon to view additional details and helpful tips."
      IconColor: =Color.Gray
      Size: =14
      X: =200
      Y: =200
      Width: =28
      Height: =28
```

Recent updates

The updated version of the **Info button** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`

Removed properties

Expand table

Removed property	Notes
`AcceptsFocus`	Removed. The control now automatically manages focus behavior for accessibility.

Bug fixes and improvements

- **Content expansion fixed**: The information flyout background now properly expands to fit all content. Previously, long text could be clipped or overflow the flyout.
- **Improved reliability**: Resolved intermittent crashes that occurred when you clicked the info button in certain scenarios.
- **Simplified properties**: Streamlined property organization with most styling properties now in the **Design** section for better discoverability.
- **Consistent styling**: Font and color properties now consistently apply to both the icon and flyout content.

Best practices

- **Keep content concise**: While the flyout can display lengthy text, aim for two to three short sentences for better user experience.
- **Use accessible labels**: Always provide descriptive **AccessibleLabel** text that explains what information the button provides.
- **Strategic placement**: Position info buttons next to form fields, labels, or unfamiliar terms where users might need clarification.
- **Consistent iconcolor**: Use consistent colors across your app to help users recognize info buttons quickly.
- **Mobile considerations**: Ensure info buttons are large enough to tap easily on mobile devices (minimum 32x32 pixels).

See also

- Modern controls overview
- Size and location properties

Last updated on 02/24/2026

Link modern control in canvas apps

Summarize this article for me

A clickable hyperlink control that navigates users to web URLs or triggers custom actions.

Description

The **Link** modern control creates clickable hyperlinks that can navigate to external websites or trigger custom actions in your canvas app. Use this control to provide users with quick access to related resources, documentation, or external services while maintaining a consistent visual design with your app's theme.

Key benefits:

- Navigate to external URLs with built-in security validation
- Customize appearance with multiple styling options
- Support for text wrapping and responsive design
- Full accessibility compliance with screen reader support
- Seamless integration with Power Apps theming system

Key properties for this control are **Text**, **Url**, and **Type**.

General

Text – The display text shown for the hyperlink. This is the text users see and click on. Can be different from the actual URL.

Url – The web address to go to when the link is clicked. Must be a valid URL that uses `http://`, `https://`, or other safe protocols. For security, the control blocks URLs that use `javascript:` or other unsafe schemes.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

Wrap – Whether the text wraps to the next line when it goes beyond the control width. When `true`, text wraps to multiple lines. When `false`, text is clipped or continues on a single line. Default is **true**.

DisplayMode – Whether the control accepts user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **Edit** mode, the link is clickable. In **View** mode, it displays as text without being clickable.

Size and position

Align – The horizontal alignment of text within the control. Accepts `Align` enum values:

Expand table

Value	Description
`Align.Left`	Aligns text to the left edge (default)
`Align.Center`	Centers text horizontally
`Align.Right`	Aligns text to the right edge
`Align.Justify`	Stretches text to fill the full width

VerticalAlign – The vertical alignment of text within the control. Accepts `VerticalAlign` enum values:

Expand table

Value	Description
`VerticalAlign.Top`	Aligns text to the top
`VerticalAlign.Middle`	Centers text vertically (default)
`VerticalAlign.Bottom`	Aligns text to the bottom

AutoHeight – Whether the control automatically increases its height when the text content exceeds the visible area.

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **150**.

Height – Distance between the control's top and bottom edges. Default is **32**.

PaddingTop – Distance between the text and the top edge of the control. Default is **5**.

PaddingBottom – Distance between the text and the bottom edge of the control. Default is **5**.

PaddingLeft – Distance between the text and the left edge of the control. Default is **5**.

PaddingRight – Distance between the text and the right edge of the control. Default is **5**.

Style and theme

Type – The visual style of the link. Accepts `LinkAppearance` enum values:

Expand table

Value	Description
`LinkAppearance.Default`	Standard link appearance with theme color (default)
`LinkAppearance.Subtle`	Subtle appearance with lighter styling

BasePaletteColor – The base color the theme uses to generate the control's color palette. Change this property to apply a different theme color to the control.

Font – The name of the font family for the text.

Size – The font size of the text, in points. Default is **15**.

Color – The color of the link text in the control.

FontWeight – The weight (thickness) of the text. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the text appears in italic style.

Underline – Whether a line appears under the text. Links typically have underlines by default.

Strikethrough – Whether a line appears through the middle of the text.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Accessibility

AccessibleLabel – A label for screen readers to announce what the link leads to. Use a description of the destination or purpose of the link, not just "link".

ContentLanguage – The language used for the link text. Inherits from app settings if you don't specify it.

Example

The following YAML example shows link controls with different styling options:

YAML

```
- DocumentationLink:
    Control: ModernLink@1.0.0
    Properties:
      Text: ="View Documentation"
      URL: ="https://docs.microsoft.com/powerapps"
      Type: =LinkAppearance.Default
      Size: =16
      FontWeight: =FontWeight.Semibold
      X: =40
      Y: =100
      Width: =200
      Height: =32

- SupportLink:
```

```
    Control: ModernLink@1.0.0
    Properties:
      Text: ="Contact Support"
      URL: ="https://support.microsoft.com"
      Type: =LinkAppearance.Subtle
      Size: =14
      X: =40
      Y: =150
      Width: =200
      Height: =32

- TermsLink:
    Control: ModernLink@1.0.0
    Properties:
      Text: ="Terms and Conditions"
      URL: ="https://company.com/terms"
      Type: =LinkAppearance.Subtle
      Size: =12
      Wrap: =false
      X: =40
      Y: =200
      Width: =150
      Height: =32
```

Recent updates

The updated version of the **Link** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Previous property	New property
`Appearance`	`Type` (uses `LinkAppearance` enum: `LinkAppearance.Default` or `LinkAppearance.Subtle`)

Removed properties

Expand table

Removed property	Notes
`AcceptsFocus`	Removed. The control now automatically manages focus behavior for accessibility.

Bug fixes and improvements

- **Wrap behavior fixed**: The **Wrap** property now correctly controls text wrapping. Previously, changing this property didn't update the text display.
- **Simplified property organization**: Properties are now logically organized into **Data** and **Design** sections for better discoverability.
- **Default padding**: All padding properties now default to **5** pixels for consistent spacing.

Best practices

- **Clear link text**: Use descriptive text that tells users where the link leads. Avoid generic text like "click here" or "link".
- **Accessible labels**: Always provide descriptive **AccessibleLabel** text for screen readers that explains the link's destination.
- **URL validation**: Only use standard `https://` or `http://` URLs. The control blocks unsafe protocols for security.
- **Visual differentiation**: Use the **Type**, **Color**, or **Underline** properties to make links visually distinct from regular text.
- **Test navigation**: Use Alt+Click in edit mode to test that links navigate to the correct destinations.
- **Mobile tap targets**: Ensure links are large enough to tap easily on mobile devices (minimum 32x32 pixels height).

See also

- Modern controls overview

- Size and location properties

Last updated on 02/24/2026

Number Input modern control in canvas apps

Summarize this article for me

A numeric input field with increment and decrement buttons, validation, and decimal precision control.

Description

The **Number Input** control is a modern control in Power Apps that lets users enter numeric values with built-in validation, increment/decrement buttons, and configurable precision settings. This control provides a user-friendly interface for numeric data entry while automatically enforcing data integrity through minimum and maximum value constraints.

Use this control when you need to collect numbers from users, such as quantities, prices, percentages, or measurements. The control includes step buttons for easy value adjustment, supports decimal precision control, and provides visual validation feedback to ensure data quality.

Key capabilities:

- Input validation with customizable min/max boundaries
- Increment and decrement buttons with configurable step values
- Decimal precision control from integers to three decimal places
- Built-in accessibility features and screen reader support
- Responsive design that adapts to different screen sizes

Key properties for this control are **Value**, **Min**, **Max**, **Step**, and **Precision**.

General

Default – The initial value displayed when the control first loads. Use this value to set a default numeric value before user interaction.

Value – The current numeric value in the control. This value is the output property you reference in formulas to get the user's entered number.

HintText – Placeholder text displayed in the input field when it's empty. Use this text to guide users on what value to enter.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

OnChange – How the app responds when the user changes the value. This event fires when the user moves focus away from the control (on blur) or clicks the increment or decrement step buttons, not on every keystroke.

Min – The minimum allowed value. Users can't enter or step below this number. The control shows a validation error if the user attempts to enter a value below the minimum.

Max – The maximum allowed value. Users can't enter or step above this number. The control shows a validation error if the user attempts to enter a value above the maximum.

Step – The increment or decrement amount when users click the step up or down buttons. Default is **1**.

Precision – The number of decimal places to display and allow. Accepts `DecimalPrecision` enum values:

Expand table

Value	Description
`DecimalPrecision.Auto`	Automatically determines decimal places based on input (default)
`DecimalPrecision.'0'`	No decimal places (integers only)
`DecimalPrecision.'1'`	One decimal place (e.g., 12.3)
`DecimalPrecision.'2'`	Two decimal places (e.g., 12.34)
`DecimalPrecision.'3'`	Three decimal places (e.g., 12.345)

ValidationState – The visual validation state of the control. Accepts `ValidationState` enum values:

Expand table

Value	Description
`ValidationState.None`	No validation styling (default)
`ValidationState.Error`	Shows error styling (red border)

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **View** mode, the value is shown but can't be edited.

Size and position

Align – The horizontal alignment of text within the control. Accepts `Align` enum values:

Expand table

Value	Description
`Align.Left`	Aligns text to the left edge (default)
`Align.Center`	Centers text horizontally
`Align.Right`	Aligns text to the right edge

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **320**.

Height – Distance between the control's top and bottom edges. Default is **32**.

PaddingTop – Distance between the text and the top edge of the control.

PaddingBottom – Distance between the text and the bottom edge of the control.

PaddingLeft – Distance between the text and the left edge of the control.

PaddingRight – Distance between the text and the right edge of the control.

Style and theme

Appearance – The visual style of the control. Accepts `Appearance` enum values:

Expand table

Value	Description
`Appearance.Filled`	Filled background style

Value	Description
`Appearance.FilledDarker`	Darker filled background (default)
`Appearance.Outline`	Outlined border with transparent background

BasePaletteColor – The base color that the theme uses to generate the control's color palette. Change this property to apply a different theme color to the control.

Font – The name of the font family for the text.

Size – The font size of the text, in points.

Color – The color of the text in the control.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Accessibility

AccessibleLabel – A label for screen readers to announce what the input field is for. Describe the type of numeric data expected, such as "Product quantity" or "Age in years".

ContentLanguage – The language used for the input text. If you don't specify this property, the app settings provide the value.

Example

The following YAML example shows a product quantity selector with inventory validation and a currency amount input with dynamic formatting:

YAML

```yaml
- QuantityInput:
    Control: ModernNumberInput@1.0.0
    Properties:
      Default: =1
      Min: =1
      Max: =100
      Step: =1
      Precision: =DecimalPrecision.'0'
      HintText: ="Enter quantity"
      AccessibleLabel: ="Product quantity"
      X: =40
      Y: =100
      Width: =200
      Height: =32

- PriceInput:
    Control: ModernNumberInput@1.0.0
    Properties:
      Default: =0
      Min: =0
      Max: =10000
      Step: =0.01
      Precision: =DecimalPrecision.'2'
      HintText: ="0.00"
      Align: =Align.Right
      Appearance: =Appearance.FilledDarker
      AccessibleLabel: ="Purchase amount in dollars"
      X: =40
      Y: =150
      Width: =200
      Height: =32

- DiscountInput:
    Control: ModernNumberInput@1.0.0
    Properties:
      Default: =0
      Min: =0
      Max: =100
      Step: =5
      Precision: =DecimalPrecision.'1'
      HintText: ="0.0"
      AccessibleLabel: ="Discount percentage"
      X: =40
      Y: =200
      Width: =200
      Height: =32
```

Recent updates

The updated version of the **Number input** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Removed properties

Expand table

Removed property	Notes
`TriggerOutput`	Removed entirely. **OnChange** now fires on focus out and step button clicks.

Bug fixes and improvements

- **OnChange optimization**: The **OnChange** event now fires on focus out (when the user leaves the field) and when clicking step buttons, not on every keystroke. This change improves performance and reduces unnecessary formula evaluations.
- **Default value**: The **Default** property allows you to set an initial value when the control loads, separate from the **Value** property.
- **HintText support**: New **HintText** property provides placeholder text to guide users.
- **Color property added**: The **Color** property for text color is now available. This property was missing in the previous version.

- **Fill property added**: The **Fill** property for background color is now available for better styling control.
- **Font property added**: The **Font** property allows you to change the font family.
- **Race condition fixed**: Resolved an issue where entering a new value and immediately submitting a form recorded the previous value instead of the new one.
- **Value overflow fixed**: Numbers no longer overflow or display incorrectly at high values.
- **Min/Max validation improved**: The control now properly validates that Min can't exceed Max.
- **Simplified properties**: Properties are now logically organized into **Data** and **Design** sections for better discoverability.

Best practices

- **Set reasonable boundaries**: Always define **Min** and **Max** values to prevent invalid data entry.
- **Match precision to data**: Use **Precision** to match your data requirements (integers for quantities, two decimals for currency, and so on).
- **Appropriate step values**: Set **Step** to meaningful increments (for example, 5 for ages, 0.01 for currency, 10 for large numbers).
- **Clear hints**: Use **HintText** to show the expected format or example values (for example, "0.00" for currency).
- **Validate input**: Use **ValidationState** to provide visual feedback when values are out of acceptable ranges.
- **Accessible labels**: Always provide descriptive **AccessibleLabel** text that explains what numeric value is expected.
- **OnChange efficiency**: Since **OnChange** fires on blur and step clicks (not every keystroke), it's safe to use for calculations or data updates without performance concerns.

Limitations

- The control doesn't directly support currency symbols. Use separate Text controls to display currency formatting.
- Step buttons might not scale proportionally with control size in all scenarios.

See also

- Modern controls overview
- Size and location properties

Last updated on 02/24/2026

Progress bar modern control in Power Apps

Article • 03/19/2024

Displays the progress, can be configured as determinate showcasing exact progress or indeterminate for ongoing progress.

Description

The progress bar can be configured to show various states of progress in the apps. This versatile control can be used to inform users on their progress as they work on the app or can be used to show loading scenarios. The key properties for this control are **Value**, **Max**, and **Indeterminate**.

General

Value – Number between 0 and 'max', which specifies how much of the task is completed. Only applicable for determinate state.

Max - The maximum value, which indicates the task is complete. The ProgressBar bar is full when value equals max. This is useful when you want to show capacity or how much of the total is uploaded or downloaded.

AccessibleLabel – Label for screen readers.

Visible - Whether a control appears or is hidden.

Behavior

Indeterminate – Set the bar to determinate or indeterminate. In the indeterminate state the value property and max property aren't honored.

Size and position

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

ProgressColor – For determinate state only, the color prop can be used to indicate a "brand" state (default), "error" state (red), "warning" state (orange), or "success" state (green). For indeterminate state, ProgressColor falls back to BasePaletteColor regardless of selected color.

Thickness – The thickness of the progress bar. Current options are only medium and large.

Shape – The shape of the bar and track. The shape property affects the corners of the bar.

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color. If the value is null or zero, then the color is driven by selected Fluent theme.

Additional properties

OnChange – Actions to perform when the value of control gets updated.

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

Radio modern control in canvas apps

Summarize this article for me

A group of mutually exclusive radio button options that users can use to select one item from a list.

Description

The **Radio** modern control provides a user-friendly way to present mutually exclusive options in your canvas apps. Users can select exactly one option from a group of choices, making it ideal for scenarios like selecting shipping methods, rating satisfaction, or choosing payment types.

This control automatically manages the selection state—when a user selects one option, any previously selected option is automatically deselected. The Radio control supports both vertical and horizontal layouts, customizable styling, and comprehensive accessibility features to ensure your apps work for all users.

Use the **Radio** control when you need users to make a single choice from 2-7 clearly defined options. For larger option sets, consider using a Combo Box control instead.

Key properties for this control are **Items**, **ItemDisplayText**, **Default**, and **Selected**.

General

Items – The data source that provides the list of options to display. Can be a table, collection, or list of values. Each item becomes a selectable radio button option.

ItemDisplayText – The formula that determines what text displays for each radio button option. Use `ThisItem` to reference the current item from **Items**. For example, `ThisItem.Title` or `ThisItem.OptionName`.

Default – The item that the control selects by default when it first loads. Must match an item from the **Items** source.

Selected – The currently selected item (output property). Use this in formulas to reference which option the user chooses.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

OnChange – How the app responds when the user selects a different radio button option. This event fires immediately every time the selection changes.

Layout – The arrangement of radio button options. Accepts `Layout` enum values:

Expand table

Value	Description
`Layout.Vertical`	Stacks options vertically (default)
`Layout.Horizontal`	Arranges options horizontally in a row

Required – Whether the user must select an option before submitting a form. When `true`, the control shows validation styling if no selection exists.

TriggerOutput – Forces the control to output its current selected value. Use this value when you need to manually trigger value updates.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **View** mode, the selected option appears with a read-only appearance (not grayed out like **Disabled**).

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges.

Height – Distance between the control's top and bottom edges. The control automatically sizes to fit all radio button options.

PaddingTop – Distance between the options and the top edge of the control.

PaddingBottom – Distance between the options and the bottom edge of the control.

PaddingLeft – Distance between the options and the left edge of the control.

PaddingRight – Distance between the options and the right edge of the control.

LineHeight – The spacing between radio button options, measured in pixels or points.

Style and theme

BasePaletteColor – The base color that the theme uses to generate the control's color palette. Change this property to apply a different theme color to the control.

Font – The name of the font family in which option text appears.

Size – The font size of the option text, in points.

Color – The color of the option text in the control.

FontWeight – The weight (thickness) of the option text. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the option text appears in italic style.

Underline – Whether a line appears under the option text.

Strikethrough – Whether a line appears through the middle of the option text.

Fill – The background fill color of the control.

HoverColor – The color of an option's text when the user hovers the mouse over it.

PressedColor – The color of an option's text when the user clicks it.

DisabledColor – The color of option text when the control is in **Disabled** mode.

RadioSize – The size of the radio button circles in pixels.

RadioBackgroundFill – The background fill color of the radio button circles.

RadioBorderColor – The border color of the radio button circles.

RadioSelectionFill – The fill color of the inner circle when a radio button is selected.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Accessibility

AccessibleLabel – A label for screen readers to announce what the radio group represents. It should describe the category or question being answered (for example, "Shipping method" or "Payment type").

ContentLanguage – The language used for the option text. Inherits from app settings if not specified.

Example

The following YAML example shows radio button controls with different option sets:

YAML

```yaml
- ShippingRadio:
    Control: ModernRadio@1.0.0
    Properties:
      Items: =["Standard (5-7 days)", "Express (2-3 days)", "Overnight"]
      Layout: =Layout.Vertical
      AccessibleLabel: ="Select shipping method"
      X: =40
      Y: =100
      Width: =300

- SatisfactionRadio:
    Control: ModernRadio@1.0.0
    Properties:
      Items: =["Very Satisfied", "Satisfied", "Neutral", "Unsatisfied", "Very
Unsatisfied"]
      Required: =true
      Layout: =Layout.Vertical
```

```
      AccessibleLabel: ="Service satisfaction rating"
      X: =40
      Y: =200
      Width: =250

- PaymentRadio:
    Control: ModernRadio@1.0.0
    Properties:
      Items: =["Credit Card", "Debit Card", "PayPal", "Bank Transfer"]
      Layout: =Layout.Vertical
      AccessibleLabel: ="Select payment method"
      X: =40
      Y: =350
      Width: =320
```

Recent updates

The updated version of the **Radio** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Bug fixes and improvements

- **Item order preserved**: The control now respects the order of items in the **Items** property and displays them in the same sequence. Previously, the control sometimes reordered items unexpectedly.

- **View mode appearance**: When you set **DisplayMode** to **View**, the control displays a proper read-only appearance instead of appearing disabled or grayed out.
- **OnChange reliability**: The **OnChange** event now fires immediately and reliably on every selection change.
- **Gallery selection fixed**: Radio buttons inside galleries now respond correctly to single clicks. Previously, a double-click was sometimes required.
- **Scrolling support**: Long lists of radio options now scroll properly instead of being cut off.
- **Theme color respected**: The **BasePaletteColor** property is now properly respected and no longer overridden by default colors.
- **Default value matching**: The **Default** property now correctly matches items from the **Items** data source without unexpected errors.
- **Hover and pressed states**: New **HoverColor** and **PressedColor** properties provide visual feedback for user interactions.
- **Radio styling control**: New properties (**RadioSize**, **RadioBackgroundFill**, **RadioBorderColor**, **RadioSelectionFill**) give you fine-grained control over radio button appearance.
- **Disabled state color**: New **DisabledColor** property controls text color when the control is disabled.

Best practices

- **Clear option text**: Use **ItemDisplayText** to provide clear, descriptive labels for each option. Avoid abbreviations or jargon.
- **Reasonable option count**: Radio buttons work best with 2-7 options. For more options, consider using a Combo Box control instead.
- **Default selection**: Always set a **Default** value to guide users toward the most common or recommended choice.
- **Accessible labels**: Provide descriptive **AccessibleLabel** text that explains what the user is choosing between (for example, "Select your preferred contact method").
- **Visual feedback**: Use **HoverColor** and **PressedColor** to provide clear visual feedback for user interactions.
- **Layout consideration**: Use **Layout.Vertical** for most cases. Only use **Layout.Horizontal** when you have short option labels and limited options (2-4).

Limitations

- The control height automatically adjusts to fit all options and you can't manually constrain it to a specific height with scrolling (items scroll if needed).
- Very small or very large width and height values might not be fully respected by the control.

See also

- Modern controls overview
- Size and location properties

Last updated on 02/24/2026

Slider modern control in canvas apps

Let users select a numeric value by dragging a handle along a track.

Description

The **Slider** modern control lets users set a value between a minimum and maximum by dragging a handle along a track. The control supports horizontal and vertical orientation.

Use this control when you want users to pick a value from a continuous range, such as:

- Volume or brightness controls
- Budget or price range selectors
- Quantity pickers
- Rating or satisfaction scales

Key properties for this control are **Default**, **Min**, **Max**, **Value** (output), and **OnChange**.

> ⓘ **Note**
>
> This article describes the updated Slider modern control. For information about what changed from the previous version, see Recent updates.

General

Default – The initial value of the slider before the user makes a change. Resets to this value when `Reset` is triggered. Default is **50**.

Min – The minimum value the user can set. Default is **0**.

Max – The maximum value the user can set. Default is **100**.

Visible – Whether the control appears or is hidden.

Behavior

OnChange – Actions to perform when the user changes the slider value. Use `Self.Value` to read the current value.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

LayoutDirection – The orientation of the slider track. Accepts `LayoutDirection` enum values:

Expand table

Value	Description
`LayoutDirection.Horizontal`	Handle moves left to right. Default.
`LayoutDirection.Vertical`	Handle moves bottom to top.

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **200**.

Height – Distance between the control's top and bottom edges. Default is **35**.

Style and theme

BasePaletteColor – The base color that the theme uses to generate the control's color palette. Change this value to apply a different theme color to the slider track and handle.

Size – The visual size of the slider handle and track. Accepts `SliderSize` enum values:

Expand table

Value	Description
`SliderSize.Small`	Smaller handle and track.
`SliderSize.Medium`	Standard size. Default.
`SliderSize.Large`	Larger handle and track.

Accessibility

AccessibleLabel – Label that screen readers read. Describe the purpose of the slider (for example, `"Volume"`). Always set this property so assistive technologies can identify the control.

ContentLanguage – The display language for the control content, if different from the app language.

Tooltip – Explanatory text that appears when the user hovers over the control. Helpful for sighted users who need additional context about the slider's purpose.

Output properties

Value – The current value of the slider, updated as the user drags the handle. Use this value in formulas to read the slider's current position (for example, `Slider1.Value`).

Examples

Budget selector with a label

This example shows a horizontal slider that lets users set a monthly budget. A label updates in real time as the user drags the handle.

YAML

```yaml
- BudgetSlider:
    Control: ModernSlider@1.0.0
    Properties:
      Default: =500
      Min: =0
      Max: =10000
      LayoutDirection: =LayoutDirection.Horizontal
      Size: =SliderSize.Medium
      AccessibleLabel: ="Monthly budget"
      Tooltip: ="Drag to set your monthly budget"
      Width: =300
      Height: =35

- BudgetLabel:
    Control: ModernText@1.0.0
    Properties:
      Size: =14
      Text: |-
        ="Budget: $" & BudgetSlider.Value
```

Vertical volume slider

This example uses a vertical slider for a volume control and writes the selected value to a variable.

YAML

```yaml
- VolumeSlider:
    Control: ModernSlider@1.0.0
    Properties:
      Default: =30
      Min: =0
      Max: =100
      LayoutDirection: =LayoutDirection.Vertical
      Size: =SliderSize.Small
      AccessibleLabel: ="Volume"
      Width: =35
      Height: =200
```

Recent updates

The updated version of the **Slider** modern control includes the following improvements and behavior changes.

Property renames

Expand table

Previous property	New property	Notes
`Value` (input)	`Default`	The input property that sets the initial value is now `Default`. `Value` is now a read-only output property.
`Layout`	`LayoutDirection`	Renamed for clarity and changed from string to typed enum.

Enum format changes

Expand table

Property	Previous value	New value
`LayoutDirection`	`"Horizontal"`	`LayoutDirection.Horizontal`
`LayoutDirection`	`"Vertical"`	`LayoutDirection.Vertical`
`Size`	`"Small"`	`SliderSize.Small`
`Size`	`"Medium"`	`SliderSize.Medium`

Property	Previous value	New value
`Size`	`"Large"`	`SliderSize.Large`

Bug fixes and improvements

- **Separate Default and Value**: `Default` sets the initial value; `Value` is now a dedicated read-only output property. This change matches the pattern used by Number Input and other modern controls, making formulas more predictable.
- **Tooltip support**: New `Tooltip` property shows explanatory text on hover.
- **Updated enums**: `LayoutDirection` and `Size` now use typed Power Fx enums, improving IntelliSense and reducing formula errors.

Best practices

- **Set meaningful Min and Max values.** A slider from 0 to 100 works well for percentages, but for dollar amounts or quantities, use a range that matches the expected input so each drag movement feels responsive.
- **Always provide an AccessibleLabel.** Screen readers rely on this property to describe the slider's purpose to users with visual impairments.
- **Pair the slider with a label** that displays the current value (for example, `"Budget: $" & Slider1.Value`). Users can see the exact number without guessing from the handle position.
- **Use vertical orientation sparingly.** Horizontal sliders are more familiar to most users. Reserve vertical sliders for cases where the metaphor matches, such as volume faders.

See also

- Modern controls overview
- Size and location properties

Last updated on 04/20/2026

Spinner modern control in Power Apps

Article • 03/19/2024

Displays state in motion such as loading a page or table.

Description

Spinner control can be used to display loading scenarios where an action is in progress. This powerful control provides multiple display options suited for many scenarios. The key properties for this control are **Label**, **Appearance**, and **SpinnerSize**.

General

Label – The label to the spinner. The label slot receives the styling related to the text associated with the Spinner.

AccessibleLabel – Label for screen readers.

Visible - Whether a control appears or is hidden.

Size and position

LabelPosition – Where the label is positioned relative to the spinner.

SpinnerSize – The size of the spinner.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

Appearance – The appearance of the Spinner. Primary or Inverted.

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color.

Font - The name of the family of fonts in which text appears.

FontSize - The font size of the text that appears on a control. If the value is null or zero, then the font size is driven by selected Fluent theme.

FontColor - The color of text in a control.

FontWeight - The weight of the text in a control: Bold, Lighter, Normal, or Semibold.

FontItalic - Whether the text in a control is italic.

FontUnderline - Whether a line appears under the text that appears on a control.

FontStrikethrough - Whether a line appears through the text that appears on a control.

Additional properties

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

OnChange - Actions to perform when the user changes the value of a control.

Feedback

Was this page helpful? Yes No

Provide product feedback

Tab List modern control in canvas apps

Summarize this article for me

A tabbed navigation control that users can use to switch between different content sections by clicking tab headers.

Description

The **Tab List** modern control enables you to create intuitive tabbed navigation interfaces in canvas apps. Users can switch between different content sections by selecting tab headers, making it ideal for organizing related information or features within a single screen.

This control serves as a navigation component that you combine with other controls to build comprehensive tabbed experiences. When users select a tab, you can use conditional visibility to show or hide content sections accordingly. The **Tab List** control offers flexible styling options and accessibility features to ensure your tabbed interfaces work effectively for all users.

Key capabilities:

- Display multiple clickable tab headers from a data source
- Support various visual styles including transparent, subtle, underline, and filled appearances
- Provide keyboard navigation and screen reader support
- Enable responsive alignment and sizing options
- Trigger events when users interact with tabs

Use the **Tab List** control when you need to organize content into logical sections that users can navigate independently, such as product details, dashboard views, or settings categories.

Key properties for this control are **Items**, **Default**, **Selected**, and **Appearance**.

General

Items – The data source that provides the list of tabs to display. Can be a table, collection, or list of values. Each item becomes a clickable tab header.

Default – The tab that is selected by default when the control first loads. Must match an item from the **Items** source.

Selected – The currently selected tab (output property). Use this in formulas to reference which tab the user chooses and to control visibility of content sections.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

OnChange – How the app responds when the user clicks a different tab. This event fires every time the selected tab changes. Use this value to update variables, load data, or track navigation.

OnSelect – How the app responds when the user clicks any tab (including the currently selected one). Use this value for additional interactions beyond navigation.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **View** mode, tabs are displayed but can't be clicked.

Size and position

Align – The horizontal alignment of tab text within each tab button. Accepts `Align` enum values:

Expand table

Value	Description
`Align.Left`	Aligns text to the left edge
`Align.Center`	Centers text horizontally (default)
`Align.Right`	Aligns text to the right edge

Alignment – The alignment and positioning of the entire tab list. Accepts `TabListAlignment` enum values:

Expand table

Value	Description
`TabListAlignment.Start`	Aligns tabs to the left or start edge
`TabListAlignment.Center`	Centers the tab list (default)
`TabListAlignment.End`	Aligns tabs to the right or end edge

X – Distance between the left edge of the control and the left edge of its parent container. If the control has no parent container, the screen acts as the parent container.

Y – Distance between the top edge of the control and the top edge of its parent container. If the control has no parent container, the screen acts as the parent container.

Width – Distance between the control's left and right edges. Default is **300**.

Height – Distance between the control's top and bottom edges. Default is **60**.

PaddingTop – Distance between the tab text and the top edge of each tab button.

PaddingBottom – Distance between the tab text and the bottom edge of each tab button.

PaddingLeft – Distance between the tab text and the left edge of each tab button.

PaddingRight – Distance between the tab text and the right edge of each tab button.

Style and theme

Appearance – The visual style of the tabs. Accepts `TabListAppearance` enum values:

Expand table

Value	Description
`TabListAppearance.Transparent`	Transparent background with text only
`TabListAppearance.Subtle`	Subtle background with minimal styling
`TabListAppearance.Underline`	Tabs with underline indicator for selection
`TabListAppearance.Filled`	Filled tab buttons with solid backgrounds

Font – The name of the font family for the tab text.

Size – The font size of the tab text, in points.

Color – The color of the tab text in the control.

FontWeight – The weight (thickness) of the tab text. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text

Value	Description
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the tab text appears in italic style.

Underline – Whether a line appears under the tab text.

Strikethrough – Whether a line appears through the middle of the tab text.

BorderColor – The color of the borders around tabs (when applicable based on Appearance).

BorderStyle – The style of the tab borders. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the tab borders in pixels.

RadiusTopLeft – The corner radius for the top-left corner of tab buttons.

RadiusTopRight – The corner radius for the top-right corner of tab buttons.

RadiusBottomLeft – The corner radius for the bottom-left corner of tab buttons.

RadiusBottomRight – The corner radius for the bottom-right corner of tab buttons.

Accessibility

AccessibleLabel – A label for screen readers to announce what the tab list represents. It should describe the category or sections being navigated (for example, "Product information sections" or "Report views").

ContentLanguage – The language used for the tab text. If you don't specify this property, the app settings provide the value.

Example

The following YAML example shows tab list controls with different appearance styles:

YAML

```yaml
- ProductTabs:
    Control: ModernTabList@1.0.0
```

```
    Properties:
      Items: =["Overview", "Specifications", "Reviews", "Support"]
      Default: ="Overview"
      Appearance: =TabListAppearance.Underline
      Alignment: =TabListAlignment.Center
      Size: =16
      FontWeight: =FontWeight.Semibold
      AccessibleLabel: ="Product information tabs"
      X: =40
      Y: =100
      Width: =600
      Height: =60

- DashboardTabs:
    Control: ModernTabList@1.0.0
    Properties:
      Items: =["Sales", "Inventory", "Reports"]
      Default: ="Sales"
      Appearance: =TabListAppearance.Filled
      Alignment: =TabListAlignment.Start
      Size: =15
      AccessibleLabel: ="Dashboard section tabs"
      X: =40
      Y: =180
      Width: =600
      Height: =60

- SettingsTabs:
    Control: ModernTabList@1.0.0
    Properties:
      Items: =["Account", "Privacy", "Notifications", "Billing"]
      Default: ="Account"
      Appearance: =TabListAppearance.Subtle
      Alignment: =TabListAlignment.Start
      Size: =14
      Align: =Align.Left
      AccessibleLabel: ="Settings categories"
      X: =40
      Y: =260
      Width: =250
      Height: =60
```

Recent updates

The updated version of the **Tab list** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Bug fixes and improvements

- **Appearance property**: The new **Appearance** property with four visual styles (**Transparent**, **Subtle**, **Underline**, **Filled**) gives you more design flexibility for tabbed interfaces.
- **Item order preserved**: Tabs now display in the exact order specified in the **Items** property. Previously, the control sometimes reordered tabs unexpectedly.
- **Items refresh fixed**: Updating the **Items** property now automatically refreshes the tab list. Previously, changes to **Items** didn't always trigger a visual update.
- **Default property**: The new **Default** property allows you to specify which tab is selected when the control loads.
- **Alignment control**: The new **Alignment** property controls the positioning of the entire tab list (start, center, end).
- **Improved tab sizing**: Tab buttons now size more consistently and responsively.

Best practices

- **Clear tab labels**: Use short, descriptive labels for tabs (one or two words). Avoid long phrases that make tabs too wide.
- **Reasonable tab count**: Use two to seven tabs for optimal usability. For more sections, consider alternative navigation patterns like a tree view or menu.
- **Default selection**: Always set a **Default** value to ensure a tab is selected when the control loads.
- **Conditional visibility**: Use the **Selected** output property to control visibility of content sections.

```
ContentSection.Visible = TabList1.Selected.Value = "Overview"
```

- **Accessible labels**: Provide descriptive **AccessibleLabel** text that explains what the tabs navigate between.
- **Consistent appearance**: Use the same **Appearance** style for all tab lists in your app for visual consistency.
- **OnChange for data loading**: Use **OnChange** to load data or update variables when tabs change, improving performance by loading only what's needed.

Limitations

- The Tab List control provides navigation only - you must manually control content visibility using the **Selected** property.
- Very small or very large width and height values might not be fully respected by the control.
- Tab buttons don't automatically scroll horizontally if there are too many tabs to fit the width.

See also

- Modern controls overview
- Size and location properties

Last updated on 02/24/2026

Text modern control in canvas apps

Summarize this article for me

Use this control to display text and labels on your app screen.

Description

The **Text** modern control displays static or dynamic text content in your canvas app. Use it for labels, headings, messages, and any text that doesn't require user input.

This versatile control offers rich formatting options, including font styling, colors, alignment, and automatic sizing. You can display static text strings or dynamic content using Power Fx formulas. The control also supports user interaction through the **OnSelect** event, making it useful for clickable labels and interactive text elements.

Key properties for this control are **Text**, **Size**, and **Color**.

General

Text – The text string to display on the canvas. Supports any text or Power Fx formula that evaluates to a string.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

OnSelect – How the app responds when the user selects the Text control. The control is accessible: **OnSelect** also triggers when the user presses Enter or Space while the control has keyboard focus.

Wrap – Whether the text wraps to the next line when it exceeds the control width. When set to `false`, text is clipped at the control boundary.

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges.

Height – Distance between the control's top and bottom edges.

AutoHeight – Whether the control automatically increases its height when the text content exceeds the visible area. When **Visible** is `false`, the **Height** output property doesn't update as text changes.

Align – The horizontal alignment of text within the control. Accepts `Align` enum values:

Expand table

Value	Description
`Align.Left`	Aligns text to the left edge
`Align.Center`	Centers text horizontally
`Align.Right`	Aligns text to the right edge
`Align.Justify`	Stretches text to fill the full width

VerticalAlign – The vertical alignment of text within the control. Accepts `VerticalAlign` enum values:

Expand table

Value	Description
`VerticalAlign.Top`	Aligns text to the top
`VerticalAlign.Middle`	Centers text vertically (default)
`VerticalAlign.Bottom`	Aligns text to the bottom

Style and theme

Font – The name of the font family for the text.

Size – The font size of the text, in points. The default is **15**.

Color – The color of the text in the control.

FontWeight – The weight (thickness) of the text. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the text appears in italic style.

Underline – Whether a line appears under the text.

Strikethrough – Whether a line appears through the middle of the text.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

PaddingTop – Distance between the text and the top edge of the control.

PaddingBottom – Distance between the text and the bottom edge of the control.

PaddingLeft – Distance between the text and the left edge of the control.

PaddingRight – Distance between the text and the right edge of the control.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Example

The following YAML example shows a clickable label with styling that you can use for headings or interactive text elements:

YAML

```yaml
- TitleLabel:
    Control: ModernText@1.0.0
    Properties:
      Text: ="Welcome to Power Apps"
      X: =40
      Y: =40
      Width: =300
      Height: =50
      Size: =24
      FontWeight: =FontWeight.Bold
      Color: =RGBA(0, 120, 212, 1)
      Align: =Align.Center
      VerticalAlign: =VerticalAlign.Middle

- StatusLabel:
    Control: ModernText@1.0.0
    Properties:
      Text: '="Status: Active"'
      X: =40
      Y: =120
      Width: =200
      Height: =40
      Size: =16
      FontWeight: =FontWeight.Semibold
      Color: =Color.Green
      Fill: =RGBA(0, 255, 0, 0.1)
      BorderColor: =Color.Green
      BorderThickness: =2
      RadiusTopLeft: =8
      RadiusTopRight: =8
      RadiusBottomLeft: =8
      RadiusBottomRight: =8
      OnSelect: =Set(varLabelClicked, true)
```

Recent updates

The updated version of the **Text** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`Weight`	`FontWeight`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Removed properties

Expand table

Removed property	Notes
`DisplayMode`	Removed from the property panel. The property still exists for backward compatibility but has no effect. The Text control is display-only; use the **OnSelect** event for interactive behavior.

Bug fixes and improvements

- **OnSelect event**: The control now supports an **OnSelect** event, making it possible to trigger actions when a user clicks or taps the text. This event is fully accessible and also responds to keyboard input (Enter or Space).
- **Vertical alignment default**: The default **VerticalAlign** is now `VerticalAlign.Middle` for better visual balance.
- **Auto height improvements**: Fixed scroll bar appearing in portrait mode and auto height not recalculating when the control is hidden.
- **Text scaling**: Fixed an issue where the text control didn't scale correctly within containers.
- **Updated command bar**: The authoring command bar and right-click menu now show Font, Font size, Alignment, Font color, and Background color for quick access.
- **Mobile-optimized defaults**: When adding the control in a mobile layout, the app automatically adjusts the defaults (font size 15, padding 5, width 150, height 32).

See also

- Modern controls overview
- Size and location properties

Last updated on 02/24/2026

Text Input modern control in canvas apps

Summarize this article for me

A text input field for capturing single-line or multi-line text, with support for password masking and search modes.

Description

Use the **Text Input** control to capture text input from users with flexible formatting options. The control supports single-line text, multi-line text areas, password fields with masking, and search input with appropriate styling. Key properties for this control are **Text**, **Placeholder**, **Type**, and **TriggerOutput**.

General

Default – The initial text displayed when the control first loads. Use this to set a default value before user interaction.

Text – The current text value in the control (output property). This is what you reference in formulas to get the user's entered text.

Placeholder – Hint text displayed in the input field when it's empty. Use this to guide users on what to enter. Default is **"Enter text"**.

Visible – Whether the control appears or is hidden. Use a Power Fx formula to show or hide the control based on app state.

Behavior

OnChange – How the app responds when the user changes the text. This event fires when the user moves focus away from the control (on blur), not on every keystroke. This improves performance and prevents issues with rapid formula execution.

OnSelect – How the app responds when the user clicks or taps inside the control.

Type – The input mode for the text field. Accepts `TextInputType` enum values:

Expand table

Value	Description
`TextInputType.SingleLine`	Standard single-line text input (default)
`TextInputType.Multiline`	Multi-line text area that allows line breaks
`TextInputType.Password`	Password field that masks characters as ***
`TextInputType.Search`	Search input with appropriate styling

TriggerOutput – Controls when the **Text** output property updates. Accepts `TriggerOutput` enum values:

Expand table

Value	Description
`TriggerOutput.Keypress`	Updates on every keystroke (default)
`TriggerOutput.FocusOut`	Updates only when focus leaves the control
`TriggerOutput.Delayed`	Updates after a short delay (used in forms)

> **Note**
>
> When added to a form, **TriggerOutput** automatically defaults to **Delayed** for better form performance.

MaxLength – The maximum number of characters allowed in the input field. Users cannot type beyond this limit.

Required – Whether the user must enter text before submitting a form. When `true`, the control shows validation styling if left empty.

ValidationState – The visual validation state of the control. Accepts `ValidationState` enum values:

Expand table

Value	Description
`ValidationState.None`	No validation styling (default)
`ValidationState.Error`	Shows error styling (red border)

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**). In **View** mode, the text is shown but cannot be edited (truly read-only).

Size and position

Align – The horizontal alignment of text within the control. Accepts `Align` enum values:

Expand table

Value	Description
`Align.Left`	Aligns text to the left edge (default)
`Align.Center`	Centers text horizontally
`Align.Right`	Aligns text to the right edge

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **320**.

Height – Distance between the control's top and bottom edges. Default is **32**.

PaddingTop – Distance between the text and the top edge of the control.

PaddingBottom – Distance between the text and the bottom edge of the control.

PaddingLeft – Distance between the text and the left edge of the control.

PaddingRight – Distance between the text and the right edge of the control.

Style and theme

Appearance – The visual style of the control. Accepts `Appearance` enum values:

Expand table

Value	Description
`Appearance.Filled`	Filled background style

Value	Description
`Appearance.FilledDarker`	Darker filled background (default)
`Appearance.Outline`	Outlined border with transparent background

BasePaletteColor – The base color used by the theme to generate the control's color palette. Changes this property to apply a different theme color to the control.

Font – The name of the font family in which text appears.

Size – The font size of the text, in points.

Color – The color of the text in the control.

FontWeight – The weight (thickness) of the text. Accepts `FontWeight` enum values:

Expand table

Value	Description
`FontWeight.Bold`	Bold text
`FontWeight.Semibold`	Semibold text
`FontWeight.Normal`	Normal weight (default)
`FontWeight.Lighter`	Lighter weight

Italic – Whether the text appears in italic style.

Underline – Whether a line appears under the text.

Strikethrough – Whether a line appears through the middle of the text.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Accessibility

AccessibleLabel – A label for screen readers to announce what the input field is for. Should describe the type of text expected (e.g., "Customer name" or "Feedback comments").

ContentLanguage – The language used for the input text. Inherits from app settings if not specified.

Example

The following YAML example shows text input controls with various input types:

YAML

```yaml
- NameInput:
    Control: ModernTextInput@1.0.0
    Properties:
      Placeholder: ="Enter your full name"
      Type: =TextInputType.SingleLine
      TriggerOutput: =TriggerOutput.Keypress
      Required: =true
      Appearance: =Appearance.FilledDarker
      AccessibleLabel: ="Full name, required field"
      X: =40
      Y: =100
      Width: =300
      Height: =32

- EmailInput:
    Control: ModernTextInput@1.0.0
    Properties:
      Placeholder: ="you@company.com"
      Type: =TextInputType.SingleLine
      TriggerOutput: =TriggerOutput.Keypress
      MaxLength: =100
      Required: =true
      AccessibleLabel: ="Email address, required field"
      X: =40
      Y: =150
      Width: =300
      Height: =32

- PasswordInput:
    Control: ModernTextInput@1.0.0
    Properties:
```

```
      Placeholder: ="At least 8 characters"
      Type: =TextInputType.Password
      TriggerOutput: =TriggerOutput.Keypress
      MaxLength: =50
      Required: =true
      AccessibleLabel: ="Password, must be at least 8 characters"
      X: =40
      Y: =200
      Width: =300
      Height: =32

- CommentsInput:
    Control: ModernTextInput@1.0.0
    Properties:
      Default: =""
      Placeholder: ="Tell us more about your inquiry..."
      Type: =TextInputType.Multiline
      TriggerOutput: =TriggerOutput.FocusOut
      MaxLength: =500
      Align: =Align.Left
      AccessibleLabel: ="Additional comments, optional, maximum 500 characters"
      X: =40
      Y: =250
      Width: =300
      Height: =120

- SearchInput:
    Control: ModernTextInput@1.0.0
    Properties:
      Placeholder: ="Search..."
      Type: =TextInputType.Search
      TriggerOutput: =TriggerOutput.Delayed
      Appearance: =Appearance.Outline
      AccessibleLabel: ="Search by name or description"
      X: =40
      Y: =390
      Width: =300
      Height: =32
```

Recent updates

The updated version of the **Text input** modern control includes improvements and behavior change.

Property renames

The following properties are renamed. Update any formulas in your app that reference the old property names.

Expand table

Previous property	New property
`FontColor`	`Color`
`FontSize`	`Size`
`FontItalic`	`Italic`
`FontStrikethrough`	`Strikethrough`
`FontUnderline`	`Underline`
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`

Bug fixes and improvements

- **OnChange optimization**: The **OnChange** event now fires only when the user leaves the field (on blur), not on every keystroke. This improves performance and prevents issues with rapid formula execution.
- **TriggerOutput flexibility**: The **TriggerOutput** property now defaults to **Keypress** for immediate updates, with **Delayed** used automatically in forms for better performance.
- **Default property**: New **Default** property allows you to set an initial value when the control loads.
- **Password preview**: Password fields now show masked characters (***) even in preview mode, making it easier to design secure forms.
- **DisplayMode.View fixed**: When set to **View** mode, the control is now truly read-only. Previously, users could still edit the text.
- **Null handling improved**: Blank or null values now properly return empty strings instead of null, preventing formula errors.
- **Multiline overflow fixed**: Multi-line text areas no longer have overflow issues when users press Enter to create new lines.
- **OnChange tabbing fixed**: **OnChange** no longer fires when users tab through fields without making changes.
- **Border rendering fixed**: Borders now render correctly even when control height is less than 32 pixels.
- **Color palette override**: The **BasePaletteColor** property in the property panel now properly overrides the global theme.

Best practices

- **Clear placeholders**: Use **Placeholder** to show example text or formatting hints (e.g., "MM/DD/YYYY" for dates).
- **Appropriate type**: Use **Type** appropriately - **Password** for sensitive data, **Multiline** for long text, **Search** for search boxes.
- **MaxLength validation**: Set **MaxLength** to prevent users from entering too much text, especially for database fields with character limits.
- **TriggerOutput consideration**: Use **Keypress** for immediate feedback (search, filters), **FocusOut** for validation, **Delayed** for expensive operations.
- **Required fields**: Mark required fields clearly with **Required** and provide visual feedback with **ValidationState**.
- **Accessible labels**: Always provide descriptive **AccessibleLabel** text that explains what text to enter.
- **OnChange efficiency**: Since **OnChange** fires on blur (not every keystroke), it's safe to use for data updates and validations without performance concerns.
- **Character counters**: For fields with **MaxLength**, consider showing a character count to help users stay within limits.

See also

- Modern controls overview
- Size and location properties

Last updated on 02/24/2026

Toggle modern control in Power Apps

Article • 07/17/2024

A control that the user can turn on or off by moving the handle.

Description

A toggle is a user interface element that has been created for modern graphical user interfaces (GUIs), but it functions in the same manner as a checkbo. The key properties for this control are **Checked**, **Label**, **OnCheck**, **OnSelect**, and **OnUncheck**.

General

Label – The text shown next to the toggle.

AccessibleLabel – Label for screen readers.

Visible - Whether a control appears or is hidden.

Behavior

Checked - The initial value of a control before it's changed by the user.

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

Size and position

LabelPostion – The text shown next to the toggle can be placed before, after, or above the toggle.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color.

Font - The name of the family of fonts in which text appears.

FontSize - The font size of the text that appears on a control. If the value is null or zero, then the font size is driven by selected Fluent theme.

FontColor - The color of text in a control.

FontWeight - The weight of the text in a control: **Bold**, **Lighter**, **Normal**, or **Semibold**.

FontItalic - Whether the text in a control is italic.

FontUnderline - Whether a line appears under the text that appears on a control.

FontStrikethrough - Whether a line appears through the text that appears on a control.

Additional properties

OnCheck – Actions to perform when the value of the toggle changes to **true**.

OnSelect – Actions to perform when the user selects a control.

OnUncheck – Actions to perform when the value of the toggle changes to **false**.

Feedback

Was this page helpful? Yes No

Provide product feedback

Avatar modern control in Power Apps

10/08/2025

A control that shows a graphic representation of a user, team, or entity.

Description

Use the avatar control to visually represent a user, team, or entity. This control supports both image and initials formats and allows a small badge to be included to help convey status. The key properties for this control are **Name** and **Image**.

General

Name - The name of the person or entity. Name is used to determine the initials displayed when there's no image and is also used for accessibility.

Image – The visual representation for the person or entity.

Badge – A small visual decoration added to convey status of the person or entity.

Visible - Whether a control appears or is hidden.

Size and position

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

Shape - Whether the avatar appears with a circular or square shape.

Appearance – An avatar can have portions of itself styled for greater emphasis or to be subtle.

The following options are available:

- **Brand**: Uses the modern theme to style the avatar. The Color palette property can be used to override this color scheme.
- **Neutral**: Uses a grayscale for the initials background to provide a subtle appearance.
- **Colorful**: Users a color from a predefined set of colors, based on a hash of the provided Name.

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color. If the value is null or zero, then the color is driven by the selected Fluent theme.

Font - The name of the family of fonts in which text appears.

FontSize - The font size of the text that appears on a control. If the value is null or zero, then the font size is driven by selected Fluent theme.

FontColor - The color of text in a control.

FontWeight - The weight of the text in a control: **Bold**, **Lighter**, **Normal**, or **Semibold**.

FontItalic - Whether the text in a control is italic.

FontUnderline - Whether a line appears under the text that appears on a control.

FontStrikethrough - Whether a line appears through the text that appears on a control.

Additional properties

Out of office - Adjusts the Badge icon to show its corresponding "out of office" variant.

Copilot answer modern control in Power Apps (preview)

Summarize this article for me

[This article is pre-release document and is subject to change.]

A control that makers can use to add predefined questions that end users can use to get generated answers.

> **Important**
>
> Starting February 2, 2026, adding a Copilot answer control to new canvas apps will be discontinued. Existing apps with this control will remain functional for a limited time but will eventually no longer be supported. Microsoft 365 Copilot Chat in canvas apps is the new the recommended solution for natural language interactions in canvas apps. To ensure a smooth transition and prevent any disruption, we recommend that you migrate to Microsoft 365 Copilot chat as soon as it becomes available in your environment.

> **Note**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - App Copilot for canvas apps is currently rolling out and might not yet be available in your region.

Description

Makers can use the Copilot answer control to set predefined questions that end users can use to get generative answers with a single click. The control is optimized for mobile users and is powered by Microsoft Copilot.

Key properties

Data source (Items) – The Copilot answer control only supports Dataverse as the data source. The answer to the question comes from the Dataverse table that you select.

Views – A user can select any of the views that are available in the data source.

Fields - Default list of fields available for the table. Users can select **Edit** to see the list of fields. They can also add or modify the fields based on their requirements.

Title – A label that is displayed at the top of the Copilot answer control.

Question for Copilot – This is the question a maker asks the Copilot answer control. The question is used as the prompt to generate a response that is displayed by the answer control that's based on the selected **Data source** property.

Additional properties

Show answer – Should the Copilot answer control answer a question when the user clicks on the 'Arrow' icon (After sending) or as soon as the control loads when the question is defined in the properties window (Immediate).

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

Position - Whether a control appears or is hidden.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Size – The size of the control on the canvas

Last updated on 02/24/2026

Data Grid modern control in canvas apps (preview)

[This article is pre-release document and is subject to change.]

Display records from a data source in a scrollable, sortable, and searchable grid.

Description

The **Data Grid** modern control displays records in a column-and-row layout built on Fluent UI. It supports optional search filtering, sortable columns, and selectable rows. Columns are configured as sub-controls and support multiple types including text, number, phone, email, URL, and button. Use it when you need a high-performance, data-dense view of tabular data with built-in interaction. Key properties for this control are **Items**, **Searchable**, **Sortable**, and **SelectMultiple**.

> ⓘ **Note**
>
> The **Data Grid** control is a new control separate from the existing **Table** control. It is not a replacement for the **Table** control.

Limitations

- **Attachment column not supported**: Attachment-type columns from Dataverse are not currently supported and do not render in the grid.
- **Row virtualization not yet available**: Virtual scrolling for large datasets is not available in this release. It is planned and will be available in the coming weeks.

General

Items – The data source for the grid. Accepts a Dataverse table, collection, or inline table expression.

Visible – Whether the control appears or is hidden.

Behavior

Searchable – Whether a search bar appears above the grid. When **true**, users can type to filter visible rows; the current search string is exposed via the **SearchText** output property. Default is **false**.

Sortable – Whether users can sort by a column by selecting its header. Default is **false**.

SelectMultiple – Whether users can select more than one row at a time. Default is **false**.

ShowHeaders – Whether column header labels appear at the top of the grid. Default is **true**.

ShowSelector – Whether a checkbox appears at the start of each row for row selection. Default is **false**.

Required – Whether the user must select at least one row.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **792**.

Height – Distance between the control's top and bottom edges. Default is **475**.

Style and theme

BasePaletteColor – The base color used by the theme to generate the control's color palette.

Font – The font family used for the grid text.

Size – The font size of the grid text, in points. Default is **14**.

Color – The color of the grid text.

Output properties

Selected – The most recently selected row, returned as a record.

SelectedItems – All currently selected rows, returned as a table. Use this when **SelectMultiple** is **true**.

SearchText – The current value the user has typed in the search bar. Only available when **Searchable** is **true**.

Configuring columns

The Data Grid uses **Data Grid Column** sub-controls to define how each column appears and what data it shows. Columns are added when you connect a data source and configure fields in the authoring panel. Column properties are locked by default — select a column and choose **Unlock** to customize it.

Column properties

Header Text – The label shown in the column header.

Visible – Whether the column appears in the grid.

Width – The width of the column in pixels. Default is **150**.

Text – (Text, Phone, Email, URL, Button columns) A Power Fx formula evaluated per row that returns the value to display. Evaluated in the context of `ThisItem` (for example, `ThisItem.'Full Name'`).

Image – (Image column) A Power Fx formula evaluated per row that returns an image value or URL.

Icon – The Fluent icon shown in the column cell.

AccessibleLabel – (Image column) The accessible label for the image, read by screen readers.

Column types

Expand table

Column type	Description
Text column	Displays a text value.
Number column	Displays a numeric value.
Phone column	Displays a phone number.

Column type	Description
Email column	Displays an email address as a link.
URL column	Displays a URL as a hyperlink.
Button column	Displays a button in each row cell.

Example

The following YAML example shows a searchable grid with two text columns:

YAML

```
- DataGrid2:
    Control: ModernDataGrid@1.1.0
    Properties:
      Items: |-
        =Table({Name: "Alice Contoso", Department: "Sales"}, {Name: "Bob Fabrikam",
Department: "Engineering"}, {Name: "Carol Northwind", Department: "Marketing"})
      Searchable: =true
      X: =40
      Y: =40
    Children:
      - Department_Column1:
          Control: ModernDataGridColumn@1.1.0
          Variant: Textual
          IsLocked: true
          Properties:
            FieldDisplayName: ="Department"
            Text: =ThisItem.Department
      - Name_Column1:
          Control: ModernDataGridColumn@1.1.0
          Variant: Textual
          IsLocked: true
          Properties:
            FieldDisplayName: ="Name"
            Text: =ThisItem.Name
```

See also

- Modern controls overview
- Table modern control
- Size and location properties

Last updated on 04/28/2026

Dropdown modern control in canvas apps

Let users pick a single value from a collapsible list.

Description

The **Dropdown** modern control presents a list of choices in a compact, collapsible format. Users select one item and the list collapses back to a single line. Use the **Dropdown** control when you need users to choose one option from a list of more than seven items, for required fields with a fixed set of choices, or for forms where screen space is limited. For fewer options (two to seven), consider using a Radio group instead. If users need to select multiple items, use a Combo box.

Key properties for this control are **Items**, **Default**, **ValidationState**, and **OnChange**.

> **Note**
>
> This article describes the updated Dropdown modern control. For information about what changed from the previous version, see Recent updates.

On desktop, the updated control opens a **Fluent-themed flyout** that matches the app's visual theme. The previous version opened the browser's HTML `<select>` element, which couldn't be themed.

General

Items – The source of data that contains the items that appear in the control. Accepts a table, a collection, or an inline array (for example, `["Option A", "Option B", "Option C"]`).

ItemDisplayText – A Power Fx formula evaluated for each item in **Items** to determine the text shown in the list. Use `ThisItem` to reference the current item. Defaults to `ThisItem.Value` for string arrays and `ThisItem.Value1` for table data sources. Change this when your data source has a specific column to display (for example, `ThisItem.Name`).

Default – The initial selected value before the user makes a change. Set this to a value that exists in **Items** (for example, `="Option A"`). Leave blank for no default selection.

AccessibleLabel – Label read by screen readers. Describe the purpose of the dropdown (for example, `"Select a country"`).

Visible – Whether the control appears or is hidden.

Behavior

OnChange – Actions to perform when the user changes the selection. Use `Self.Selected.Value` inside the formula to read the newly selected item. For example, `Notify("You selected " & Self.Selected.Value)` displays a message each time the selection changes.

Required – Whether the user must make a selection. When **true**, an indicator appears beside the dropdown. Default is **false**.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

ValidationState – Applies validation styling to the control. Accepts `ValidationState` enum values:

Expand table

Value	Description
`ValidationState.None`	No validation styling. Default.
`ValidationState.Error`	Highlights the control border in red to indicate an error.

Size and position

X – Distance between the left edge of the control and the left edge of its parent container (screen if no parent container).

Y – Distance between the top edge of the control and the top edge of its parent container (screen if no parent container).

Width – Distance between the control's left and right edges. Default is **320**.

Height – Distance between the control's top and bottom edges. Default is **32**.

PaddingTop – Distance between the control content and the top edge of the control.

PaddingBottom – Distance between the control content and the bottom edge of the control.

PaddingLeft – Distance between the control content and the left edge of the control.

PaddingRight – Distance between the control content and the right edge of the control.

Style and theme

Appearance – The visual style of the dropdown field. Accepts `Appearance` enum values:

Expand table

Value	Description
`Appearance.FilledDarker`	Filled with a slightly darker background. Default.
`Appearance.Filled`	Filled with a standard background.
`Appearance.Outline`	Outlined border with a transparent background.

BasePaletteColor – The base color used by the theme to generate the control's color palette. Use this value to apply a different theme color to the control.

Font – The font family used for the dropdown text.

Size – The font size of the dropdown text, in points. Default is **14**.

Color – The color of the dropdown text.

FontWeight – The weight (thickness) of the dropdown text. Accepts `FontWeight` enum values: `FontWeight.Bold`, `FontWeight.Semibold`, `FontWeight.Normal` (default), `FontWeight.Lighter`.

Italic – Whether the dropdown text appears in italic style.

Underline – Whether a line appears under the dropdown text.

Strikethrough – Whether a line appears through the middle of the dropdown text.

Fill – The background fill color of the control.

BorderColor – The color of the control's border.

BorderStyle – The style of the control's border. Accepts `BorderStyle` enum values: `BorderStyle.Solid`, `BorderStyle.Dashed`, `BorderStyle.Dotted`, or `BorderStyle.None`.

BorderThickness – The thickness of the control's border in pixels.

RadiusTopLeft – The corner radius for the top-left corner of the control.

RadiusTopRight – The corner radius for the top-right corner of the control.

RadiusBottomLeft – The corner radius for the bottom-left corner of the control.

RadiusBottomRight – The corner radius for the bottom-right corner of the control.

Additional properties

ContentLanguage – The display language for the control content, if different from the app language.

Output properties

Selected – The currently selected item, returned as a record. For a string array, use `Dropdown1.Selected.Value` to read the selected text. For a table data source, use `Dropdown1.Selected.<ColumnName>`.

> ⓘ Note
>
> **Selected** is an output property. You can reference it in other controls and formulas, but you can't set it directly.

Examples

Basic dropdown with inline options

This example creates a required country dropdown with inline validation. When the user hasn't selected a value, the control shows an error state.

YAML

```
- CountryDropdown:
    Control: ModernDropdown@1.0.0
    Properties:
      AccessibleLabel: ="Select your country"
      Default: ={Value:"United States"}
      ItemDisplayText: =ThisItem.Value
      Items: =["Australia", "Canada", "United Kingdom", "United States"]
      Required: =true
      ValidationState: =If(IsBlank(CountryDropdown.Selected.Value),
ValidationState.Error, ValidationState.None)
      Width: =240
```

Dropdown bound to a data source

This example binds the dropdown to a table data source and displays the **Name** column. The **OnChange** action stores the selected record in a variable.

YAML

```
- DepartmentDropdown:
    Control: ModernDropdown@1.0.0
    Properties:
      AccessibleLabel: ="Select a department"
      Appearance: =Appearance.Outline
      ItemDisplayText: =ThisItem.Name
      Items: =Departments
      OnChange: =Set(varSelectedDept, Self.Selected)
      Width: =320
```

Recent updates

The updated version of the **Dropdown** modern control includes the following improvements and behavior changes.

Property renames

Expand table

Previous property	New property	Notes
`DefaultSelectedItems`	`Default`	Set the initial selection through `Default`. Change any formula referencing `Dropdown1.DefaultSelectedItems` to `Dropdown1.Default`.
`FontSize`	`Size`	Renamed to match other modern controls. Default changed from **0** (inherits from theme) to **14**.
`BorderRadius`	`RadiusTopLeft`, `RadiusTopRight`, `RadiusBottomLeft`, `RadiusBottomRight`	Split into four independent corner radius properties for greater styling flexibility.

Enum format changes

Expand table

Property	Previous value	New value
`Appearance`	`"FilledDarker"`	`Appearance.FilledDarker`
`Appearance`	`"Filled"`	`Appearance.Filled`
`Appearance`	`"Outline"`	`Appearance.Outline`
`ValidationState`	`"None"`	`ValidationState.None`
`ValidationState`	`"Error"`	`ValidationState.Error`

Bug fixes and improvements

- **Fluent-themed flyout**: On desktop, the dropdown list now opens as a Fluent-themed flyout that matches the app's visual design. The previous version used the browser's HTML `<select>` element, which couldn't be themed. On mobile, both versions use the device-native picker and it remains unchanged.
- **ItemDisplayText**: New property that controls which field of a data source record to show in the list. For string arrays, this property defaults automatically. For table data sources, set this property to the column you want to display (for example, `ThisItem.Name`).
- **Full font control**: New `Font`, `Color`, `FontWeight`, `Italic`, `Underline`, and `Strikethrough` properties give full text styling control, consistent with other modern controls.
- **Updated enums**: `Appearance` and `ValidationState` now use typed Power Fx enums, improving IntelliSense and reducing formula errors.

Accessibility

AccessibleLabel – Set this property to a meaningful description so screen readers can announce the purpose of the dropdown (for example, `"Select your country"`).

Required – When set to **true**, screen readers announce the field as required.

Keyboard users can press **Enter** or **Space** to open the dropdown, use **Arrow** keys to move between items, and press **Enter** to confirm a selection.

Limitations

- The Fluent-themed flyout is available on desktop only. On mobile, the control uses the device-native picker.
- Very large data sources may cause the dropdown list to load slowly. Filter or limit the data where possible.

See also

- Modern controls overview
- Size and location properties

Last updated on 04/20/2026

Stream control (based on SharePoint) in Power Apps (preview)

Article • 05/21/2025

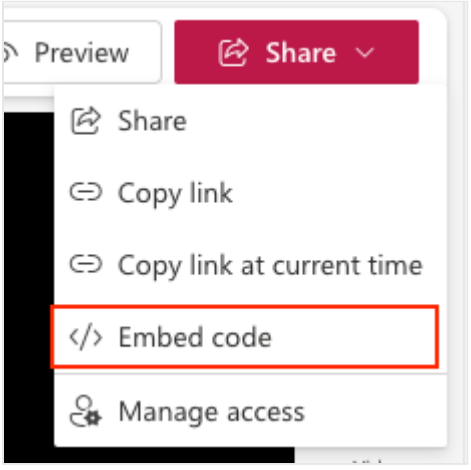
[This article is pre-release document and is subject to change.]

The Stream control in Power Apps lets you embed Microsoft Stream (on SharePoint) videos directly in your canvas app. Adding a Stream video in your app makes a smooth integration of videos hosted on Microsoft Stream in your canvas app. It also improves the user experience.

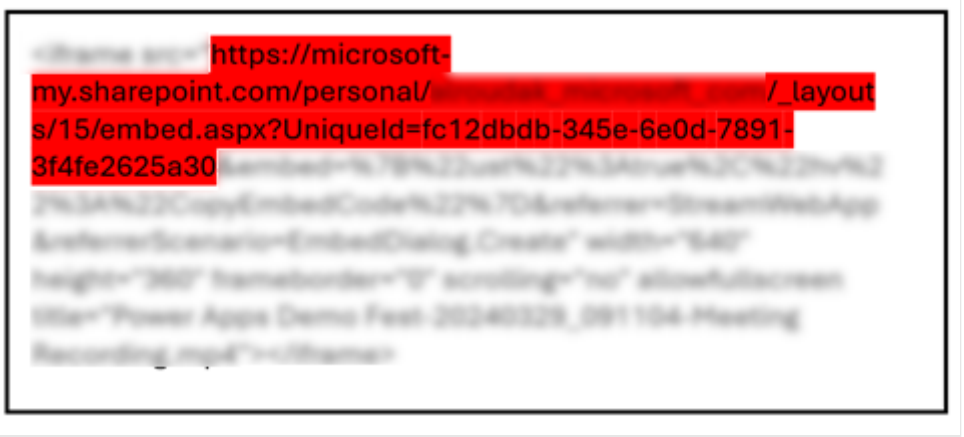
Prerequisites

To add a Stream video in your canvas app, follow these steps to get the embed URL of the video:

1. Open your Stream (on SharePoint) video.
2. Select **Share** > **Embed code**.



3. Copy the URL starting from **https://** to the **UniqueId=** including the Unique ID numbers.



> **Important**

The Microsoft Stream control in Power Apps only supports URLs that follow the *.sharepoint.com pattern. If your SharePoint domain doesn't follow this pattern, the Stream control will show an invalid URL error.

Add Stream control

1. Open your app canvas app for editing.
2. On the command bar, select **Insert** > **Media** > **Stream (preview)**.
3. In the **STREAM** pane, enter the **Stream URL** of the video. This is the embed URL from the Prerequisites section of the article.
4. Customize the size of the embedded video player by setting the width and height properties of the control.

Properties

Stream URL (Required) - The URL of the Stream video that you want to embed. This is the embed URL of the video. The URL should only have Unique ID. If URL detects other parameters, the control detects as invalid URL.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Mobile support for stream control

Stream control is now also supported on mobile. To comply with Apple's privacy policies, iOS users must follow these steps to enable cross-site tracking on Power Apps:

1. On iOS, select **Settings** > go to **Power Apps**.
2. Set the toggle for **Allow Cross-Website Tracking** to **On**.

Limitations

1. Control properties such as auto start, auto play, and start from are coming soon.

2. Don't add more than four videos on a single canvas app screen because it can cause performance issues.
3. The last played state isn't preserved for a video that's in a gallery.

Table modern control in Power Apps (preview)

Article • 03/19/2024

[This article is pre-release document and is subject to change.]

A control that shows a set of data in a tabular or list format.

Description

The **Table** control is a modern responsive control that shows a set of data in a tabular or list format. The tabular format includes column headings for each displayed field and a footer that provides details about the dataset such as the record count. The list format presents the data in a single column and has three fields for each row. As an app maker, you have full control over what fields are added to the control including the order. Although all added fields are visible in the tabular format, only the initial three fields appear in the narrower list format.

Similar to the **Gallery** and **Data** table controls, the **Table** maintains a **Selected** property that points to the selected row that can be used to link the **Table** control to other controls. The key properties for this control are **Items**, **ReflorBehavior**, **Selected**, and **OnSelect**.

Supported capabilities

- Data in a **Table** control is read-only.
- The tabular format offers a modern data browsing experience with infinite scroll, eliminating the need for paging buttons.
- The **Table** control supports linking to Microsoft Dataverse tables and collections.
- When you connect a **Table** control to a compatible connector such as Microsoft Dataverse, a predefined set of fields are displayed by default. You can also display or hide other fields based on your needs.
- The **Table** control can automatically switch between tabular and list formats. You can manually manage this behavior using the **Reflow** property.
- When you run your app, you can adjust column width in the **Table** control. However, your changes aren't saveontrol while you run the app. Your changes aren't saved.
- Change the color palette for the **Table** via the Base palette color property.

Unsupported capabilities

Currently, the listed capabilities aren't supported. However, since the **Table** control is still evolving, check back for any updates.

- Linking the Table to other data connectors.
- Customize the style of the **Table** other than color palette or modify individual columns.

General

Items - The source of data for the items that appear in the table.

NoDataText - The message shown to a user when there are no records to show in the **Table** control.

Visible - Whether a control appears or is hidden.

Behavior

ReflowBehavior - The default responsive behavior of the table. **Reflow** means the table displays in tabular form when the control is 480 or more pixels wide and transitions to a list format in narrower conditions. **GridOnly** (or **ListOnly**) means the table always displays in a tabular (or list) format regardless of the control width, and you can use Power Fx formulas to create your own breakpoints for defining the responsive behavior of the table.

EnableSorting – Allows users to sort the data in the table by using dropdowns from the column headers for supported data types.

EnableRangeSelection – Whether users can select a portion of the table to copy its values. Default is **Yes**.

Size and position

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Width - The distance between a control's left and right edges.

Height - The distance between a control's top and bottom edges.

Style and theme

HeaderFont - The font of the column headings.

HeaderFontSize - The font size of the column headings.

HeaderFontColor - The text color for the column headings.

HeaderFontWeight - The font weight of the column headings.

BasePaletteColor - The color palette applied to a control. This impacts all surfaces of the control that render a theme color.

Font - The name of the family of fonts in which text appears.

FontSize - The font size of the text that appears on a control. If the value is null or zero, then the font size is driven by selected Fluent theme.

FontColor - The color of text in a control.

FontWeight - The weight of the text in a control: Bold, Lighter, Normal, or Semibold.

Additional properties

Selected – The selected row in the **Table** control.

DisplayMode - Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

OnSelect – Defines the action triggered when a single record is selected.

ShowAvatar - Whether the user's avatar appears or is hidden in the list mode.

ShowColumnHeaders – Whether the column headers appear or are hidden.

ShowFooter - Whether the footer appears or is hidden.

Size – The size of the control on the canvas.

AccessibleLabel – Label for screen readers.

Controls and properties in canvas apps

Article • 03/13/2023

Configure the appearance and behavior of a control by setting one of its properties. Each type of control has a different set of properties. Some properties, such as **Height** and **Width**, are common to almost every type of control, but other properties, such as **CheckboxSize**, are specific to one type of control.

Controls

Add picture – Load images from the local device, for upload to a data source.

Address input - See dynamic address suggestions as you type.

Attachments – Download and upload files from the local device to a data source.

Audio – Play an audio clip or the audio portion of a video clip.

Barcode reader – Scans barcodes, QR codes, and data-matrix codes on an Android, iOS, or Windows device.

Barcode scanner (retired) – Scans barcodes, QR codes, and data-matrix codes on an Android or iOS device. Replaced by the barcode reader.

Button – Interact with the app by clicking or tapping.

Camera – Take and save photos in the app or to a data source.

Card – Display and edit an individual field of a record in a **Edit form** or **Display form** control.

Check box – Select or clear an option to specify **true** or **false**.

Column chart – Show values as vertical bars relative to two axes.

Column - Provides the display experience for a single field in a **Data table** control.

Combo box - Allows users to make selections from provided choices. Supports search and multi-select.

Container (experimental) - Create nested hierarchy for accessibility and responsiveness.

Data table (preview) - Show data in a tabular format.

Date picker – Specify a date by clicking or tapping.

Display form – Display records in a data source using a form.

Drop down – Show the first item in a list until a chevron is selected.

Edit form – Edit and create records in a data source using a form.

Display and Edit form - Experimental feature: Add dynamic forms in which users can view, navigate, and edit relational data from the Microsoft Dataverse.

Export – Export data for use elsewhere in Power Apps.

Gallery – Show a list of records that can contain multiple types of data.

HTML text – Convert HTML tags automatically.

Horizontal container – Determines the position of the child components so that you never have to set X, Y for a component inside the container.

Icon – Add graphic appeal and visual interest.

Image – Show an image from, for example, a local file or a data source.

Import – Import data from elsewhere in Power Apps.

Line chart – Show values as data points relative to two axes.

List box – Select one or more items in a list.

Map - View the physical position of objects from a data source, or by inputting new physical locations.

Measuring Camera - Measure distance, area, and volume in mixed reality.

Microphone – Record and save sounds in the app or to a data source.

PDF viewer (experimental) – Show the content of a PDF file on the Internet.

Pen input – Draw an image or text, and save it in the app or to a data source.

Pie chart – Show how values relate to each other.

Power BI tile – Display a Power BI tile inside an app.

Radio – Show options that are mutually exclusive.

Rating – Indicate a value between 1 and a specified number.

Rich text editor – Allows rich text formatting by app users.

Screen – Show and update data about a specific task.

Shape – Display arrows and geometric shapes, such as rectangles.

Slider – Specify a value by dragging a handle.

Stream Video – Play videos and browse through channels from the Microsoft Stream service.

Label – Shows data such as text, numbers, dates, or currency.

Text input – Type text, numbers, and other data.

Timer – Configure your app to respond after a certain amount of time passes.

Toggle – Drag a handle to specify **true** or **false**.

Video – Play a video clip from a local file, a data source, or YouTube.

Vertical container – Determines the position of the child components so that you never have to set X, Y for a component inside the container.

3D object - Rotate and zoom into the model with simple gestures in 3D.

View in MR - See how a particular item might fit within a specified space in mixed reality.

View shape in MR - See if a simple cube might fit within a specified space in mixed reality.

Web barcode scanner (experimental) – The legacy barcode scanner, which is obsolete but might be useful for scanning codes in a web browser.

Common properties by category

Color and border – Configure the color and border of a control that can change as a user interacts with it.

Core – Configure whether the user can see and interact with a control.

Image – Configure what image is shown and how it fills the control.

Size and location – Configure how big a control (or an element of a control) is and where it is in relation to the screen it's on.

Text – Configure how text appears in controls, such as font characteristics, alignment, line height.

All properties

A

AccesibleLabel – Reference information about properties related to accessibility in Power Apps.

ActualZoom – The actual zoom of the control, which may differ from the zoom requested with the **Zoom** property. Applies to the PDF viewer control.

Align – The location of text in relation to the horizontal center of its control. Applies to many controls.

AllItems – All items in a gallery, including additional control values that are a part of the gallery's template. Applies to the Gallery control.

AutoDisableOnSelect – Automatically disables the control while the OnSelect behavior is executing. Applies to Button and Image controls.

AutoHeight – Whether a label automatically increases its height if its Text property contains more characters than the control can show. Applies to the Label control.

AutoPause – Whether an audio or video clip automatically pauses if the user navigates to a different screen. Applies to Audio, Timer, and Video controls.

AutoStart – Whether an audio or video control automatically starts to play a clip when the user navigates to the screen that contains that control. Applies to Audio, Timer, and Video controls.

B

BackgroundImage – The name of an image file that appears in the background of a screen. Applies to the Screen control.

BorderColor – The color of a control's border. Applies to many controls.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**. Applies to many controls.

BorderThickness – The thickness of a control's border. Applies to many controls.

Brightness – How much light the user is likely to perceive in an image. Applies to the Camera control.

C

CalculateOriginalDimensions – Enables the **OriginalHeight** and **OriginalWidth** properties. Applies to the **Image** control.

Camera – On a device that has more than one camera, the numeric ID of the camera that the app uses. Applies to the **Camera** control.

CheckboxBackgroundFill – The background color of the box that surrounds the checkmark in a checkbox control. Applies to the **Check box** control.

CheckboxBorderColor – The color of the border that surrounds the checkmark in a checkbox control. Applies to the **Check box** control.

CheckboxSize – The width and height of the box that surrounds the checkmark in a checkbox control. Applies to the **Check box** control.

CheckmarkFill – The color of the checkmark in a checkbox control. Applies to the **Check box** control.

ChevronBackground – The color behind the down arrow in a dropdown list. Applies to the **Drop down** control.

ChevronFill – The color of the down arrow in a dropdown list. Applies to the **Drop down** control.

Clear – Whether a text-input control shows an "X" that the user can tap or click to clear the contents of that control. Applies to the **Text input** control.

Color – The color of text in a control. Applies to many controls.

ContentLanguage – The language of the control's content, if different from the control's container.

Contrast – How easily the user can distinguish between similar colors in an image. Applies to the **Camera** control.

CurrentFindText – The current search term that is in use. Applies to the **PDF viewer** control.

CurrentPage – The number of the page in a PDF file that is actually being shown. Applies to the **PDF viewer** control.

D

Data – The name of a collection that you want to export to a local file. Applies to the Export control.

DataField – The name of the field within a record that this card displays and edits. Applies to the Card control.

DataSource – The data source that contains the record that the user will show, edit, or create. Applies to Display form and Edit form controls.

Default – The initial value of a control before it is changed by the user. Applies to many controls.

DefaultDate – The initial value of a date control before it is changed by the user. Applies to the Date Picker control.

DefaultMode – The initial mode of a form control, either **Edit**, **New**, or **View**. Applies to the Edit form control.

DelayOutput – When set to true, user input is registered after half a second delay. Useful for delaying expensive operations until user completes inputting text (i.e. for filtering when input is used in other formulas). Applies to the Text input control.

Direction – Whether the first item in a gallery in landscape orientation appears near the left or right edge. Applies to the Gallery control.

Disabled – Whether the user can interact with the control. Applies to many controls.

DisabledBorderColor – The color of a control's border if the control's Disabled property is set to **true**. Applies to many controls.

DisabledColor – The color of text in a control if its Disabled property is set to **true**. Applies to many controls.

DisabledFill – The background color of a control if its Disabled property is set to **true**. Applies to many controls.

DisplayName – The user friendly name for a field in a data source. Applies to the Card control.

DisplayMode – Values can be Edit, View, or Disabled. Configures whether the control allows user input (Edit), only displays data (View) or is disabled (Disabled).

Document – The URL, enclosed in double-quotation marks, of a PDF file. Applies to the PDF viewer control.

Drop shadow - Adds shadow effects around the control. Applies to the Horizontal container and Vertical container control.

- Duration – How long a timer runs. Applies to the Timer control.

E

EnableSpellCheck – Whether a Text input control should use the browser spell check function. Power Apps for Windows doesn't support this property.

EndYear – The latest year to which the user can set value of a date-picker control. Applies to the Date Picker control.

Error – The meaning of this property is dependent on the control:

- Add picture control - If there is a problem uploading an image, this property will contain an appropriate error string.
- Card control – The user friendly error message to display for this field when validation fails.
- Edit form control – A user friendly error message to display for this form when the SubmitForm function fails.

ErrorKind – If an error occurs when **SubmitForm** runs, the kind of error that occurred. Applies to Display form and Edit form controls.

Explode – The distance between wedges in a pie chart. Applies to the Pie chart control.

F

Fill – The background color of a control. Applies to many controls.

FindNext – Finds the next instance of **FindText** in the document. Applies to the PDF viewer control.

FindPrevious – Finds the previous instance of **FindText** in the document. Applies to the PDF viewer control.

FindText – The search term to look for in the document. Applies to the PDF viewer control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears. Applies to many controls.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**. Applies to many controls.

G

GridStyle – Whether a column or line chart shows its x-axis, its y-axis, both, or neither. Applies to Column chart and Line chart controls.

H

HandleActiveFill – The color of the handle for a slider as the user changes its value. Applies to the Slider control.

HandleFill – The color of the handle (the element that changes position) in a toggle or slider control. Applies to the Slider control.

HandleHoverFill – The color of the handle in a slider when the user keeps the mouse pointer on it. Applies to the Slider control.

Height – The distance between a control's top and bottom edges. Applies to many controls.

HintText – Light-grey text that appears in a text-input control if it's blank. Applies to the Text input control.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control. Applies to many controls.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it. Applies to many controls.

HoverFill – The background color of a control when the user keeps the mouse pointer on it. Applies to many controls.

HTMLText – Text that appears in an HTML text control and that may contain HTML tags. Applies to the HTML text control.

I

Image – The name of the image that appears in an image, audio, or microphone control. Applies to Audio, Image, Microphone, and Video controls.

ImagePosition – The position (**Fill**, **Fit**, **Stretch**, **Tile**, or **Center**) of an image in a screen or a control if it isn't the same size as the image. Applies to many controls.

Input – Input. Applies to the Pen input control.

Italic – Whether the text in a control is italic. Applies to many controls.

Item – The record in the **DataSource** that the user will show or edit. Applies to Display form and Edit form controls.

ItemBorderColor – The color of the border around each wedge in a pie chart. Applies to the Pie chart control.

ItemBorderThickness – The thickness of the border around each wedge in a pie chart. Applies to the Pie chart control.

ItemColorSet – The color of each data point in a chart. Applies to Column chart, Line chart, and Pie chart controls.

ItemPaddingLeft – The distance between text in a listbox and its left edge. Applies to the List Box control.

Items – The source of data that appears in a control such as a gallery, a list, or a chart. Applies to many controls.

ItemsGap – The distance between columns in a column chart. Applies to the Column chart control.

L

LabelPosition – The location of labels in a pie chart relative to its wedges. Applies to the Pie chart control.

LastSubmit – The last successfully submitted record, including any server generated fields. Applies to the Edit form control.

Layout – Whether the user scrolls through a gallery or adjusts a slider top to bottom (**Vertical**) or left to right (**Horizontal**). Applies to Gallery and Slider controls.

LineHeight – The distance between, for example, lines of text or items in a list. Applies to List Box, Radio, Label, and Text input controls.

Loop – Whether an audio or video clip automatically starts over as soon as it finishes playing. Applies to Audio and Video controls.

M

Markers – Whether a column or line chart shows the value of each data point. Applies to Column chart and Line chart controls.

MarkerSuffix – Text that appears after each value in a column chart for which the Markers property is set to **true**. Applies to the Column chart control.

Max – The maximum value to which the user can set a slider or a rating. Applies to Rating and Slider controls.

MaxLength – The number of characters that the user can type into a text-input control. Applies to the Text input control.

Media – An identifier for the clip that an audio or video control plays. Applies to Add picture, Audio, and Video controls.

Mic – On a device that has more than one microphone, the numeric ID of the microphone that the app uses. Applies to the Microphone control.

Min – The minimum value to which the user can set a slider. Applies to the Slider control.

MinimumBarWidth – The narrowest possible width of columns in a column chart. Applies to the Column chart control.

Mode – The meaning of this property is dependent on the control:

- Edit form control – The control is in **Edit** or **New** mode.
- Pen input control – The control is in **Draw**, **Erase**, or **Select** mode.
- Text input control – The control is in **SingleLine**, **MultiLine**, or **Password** mode.

N

NavigationStep – How far a gallery scrolls if its ShowNavigation property is set to **true** and the user selects a navigation arrow at either end of that gallery. Applies to the Gallery control.

NumberOfSeries – How many columns of data are reflected in a column or line chart. Applies to Column chart and Line chart controls.

O

OnChange – The behavior of an app when the user changes the value of a control (for example, by adjusting a slider). Applies to many controls.

OnCheck – The behavior of an app when the value of a checkbox or a toggle changes to **true**. Applies to Check box and Toggle controls.

OnEnd – The behavior of an app when an audio or video clip finishes playing. Applies to Audio and Video controls.

OnFailure – The behavior of an app when a data operation has been unsuccessful. Applies to the Edit form control.

OnHidden – The behavior of an app when the user navigates away from a screen. Applies to the Screen control.

OnPause – The behavior of an app when the user pauses the clip that an audio or video control is playing. Applies to Audio and Video controls.

OnReset – The behavior of an app when an Edit form control is reset. Applies to the Edit form control.

OnSelect – The behavior of an app when the user taps or clicks a control. Applies to many controls.

OnStart – The behavior of an app when the user opens it or starts to record with a microphone control. Applies to Audio, Microphone, Screen, and Video controls.

OnStateChange – The behavior of an app when the state of the control changes. Applies to the PDF viewer control.

OnStop – The behavior of an app when the user stops recording with a microphone control. Applies to the Microphone control.

OnStream – The behavior of an app when the Stream property is updated. Applies to the Camera control.

OnSuccess – The behavior of an app when a data operation has been successful. Applies to the Edit form control.

OnTimerEnd – The behavior of an app when a timer finishes running. Applies to the Timer control.

OnTimerStart – The behavior of an app when a timer starts to run. Applies to the Timer control.

OnUncheck – The behavior of an app when the value of a checkbox or a toggle changes to **false**. Applies to **Check box** and **Toggle** controls.

OnVisible – The behavior of an app the user navigates to a screen. Applies to the **Screen** control.

OriginalHeight – Original height of an image, enabled with the **CalculateOriginalDimensions** property. Applies to the **Image** control.

OriginalWidth – Original width of an image, enabled with the **CalculateOriginalDimensions** property. Applies to the **Image** control.

Overflow – Whether a scrollbar appears in a label if its **Wrap** property is set to **true** and the value of the control's **Text** property contains more characters than the control can show at one time. Applies to the **Label** control.

P

Padding – The distance between the text on an import or export button and the edges of that button. Applies to **Add picture**, **Export**, and **Import** controls.

PaddingBottom – The distance between text in a control and the bottom edge of that control. Applies to many controls.

PaddingLeft – The distance between text in a control and the left edge of that control. Applies to many controls.

PaddingRight – The distance between text in a control and the right edge of that control. Applies to many controls.

PaddingTop – The distance between text in a control and the top edge of that control. Applies to many controls.

Page – The number of the page that you want to show. Applies to the **PDF viewer** control.

PageCount – The number of pages in a document. Applies to the **PDF viewer** control.

Paused – *True* if a media playback control is currently paused, *false* otherwise. Applies to **Audio** and **Video** controls.

Photo – The image captured when the user takes a picture. Applies to the **Camera** control.

Pressed – *True* while a control is being pressed, *false* otherwise. Applies to the **Button** control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control. Applies to many controls.

PressedColor – The color of text in a control when the user taps or clicks that control. Applies to many controls.

PressedFill – The background color of a control when the user taps or clicks that control. Applies to many controls.

R

RadioBackgroundFill – The background color of the circles in a radio-button control. Applies to the **Radio** control.

RadioBorderColor – The color of the circle for each option in a radio-button control. Applies to the **Radio** control.

RadioSelectionFill – The color that appears inside the circle of the selected option in a radio-button control. Applies to the **Radio** control.

RadioSize – The diameter of the circles in a radio-button control. Applies to the **Radio** control.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded. Applies to many controls.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded. Applies to many controls.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded. Applies to many controls.

RadiusTopRight – The degree to which the top-right corner of a control is rounded. Applies to many controls.

RailFill – The background color of the rectangle in a toggle control when its value is **false** or the color of the line to the right of the handle in a slider control. Applies to **Slider** and **Toggle** controls.

RailHoverFill – When you hover on a toggle control or a slider, the background color of the rectangle in a toggle control when its value is **false** or the color of the line to the right of the handle in a slider control. Applies to **Slider** and **Toggle** controls.

RatingFill – The color of the stars in a rating control. Applies to the Rating control.

ReadOnly – Whether a user can change the value of a slider or rating control. Applies to Rating and Slider controls.

Repeat – Whether a timer automatically restarts when it finishes running. Applies to the Timer control.

Required – Whether a card, editing the field of a data source, must contain a value. Applies to the Card control.

Reset – Whether a control reverts to its default value. Applies to many controls. Also see the Reset function.

S

Selected – The selected item. Applies to Drop down and Gallery controls.

SelectedDate – The date currently selected in a date control. Applies to the Date Picker control.

SelectionColor – The text color of a selected item or items in a list or the color of the selection tool in a pen control. Applies to Drop down, List Box, and Pen input controls.

SelectionFill – The background color of a selected item or items in a list or a selected area of a pen control. Applies to Drop down and List Box controls.

SelectionThickness – The thickness of the selection tool for a pen-input control. Applies to the Pen input control.

SelectMultiple – Whether a user can select more than one item in a listbox. Applies to the List Box control.

SeriesAxisMax – The maximum value of the y-axis for a column or line chart. Applies to the Column chart control.

SeriesAxisMin – A number that determines the minimum value of the y-axis for a column chart. Applies to the Column chart control.

ShowControls – Whether an audio or video player shows, for example, a play button and a volume slider, and a pen control shows, for example, icons for drawing, erasing, and clearing. Applies to Audio, PDF viewer, Pen input, and Video controls.

ShowLabels – Whether a pie chart shows the value that's associated with each of its wedges. Applies to the Pie chart control.

ShowNavigation – Whether an arrow appears at each end of a gallery so that a user can scroll through the items in the gallery by clicking or tapping an arrow. Applies to the Gallery control.

ShowScrollbar – Whether a scrollbar appears when the user hovers over a gallery. Applies to the Gallery control.

ShowValue – Whether a slider's or rating's value appears as the user changes that value or hovers over the control. Applies to Rating and Slider controls.

Size – The font size of the text that appears on a control. Applies to many controls.

Snap – Whether, when a user scrolls through a gallery, it automatically snaps so that the next item appears in full. Applies to the Gallery control.

Start – Whether an audio or video clip plays. Applies to Audio, Timer, and Video controls.

StartTime – The time after the start of an audio or video clip when the clip starts to play. Applies to Audio and Video controls.

StartYear – The earliest year to which the user can set the value of a date-picker control. Applies to the Date Picker control.

Stream – Automatically updated image based on the StreamRate property. Applies to the Camera control.

StreamRate – How often to update the image on the Stream property, in milliseconds. This value can range from 100 (1/10th of a second) to 3,600,000 (1 hour). Applies to the Camera control.

Strikethrough – Whether a line appears through the text that appears on a control. Applies to many controls.

T

TemplateFill – The background color of a gallery. Applies to the Gallery control.

TemplatePadding – The distance between items in a gallery. Applies to the Gallery control.

TemplateSize – The height of the template for a gallery in vertical/portrait orientation or the width of the template for a gallery in horizontal/landscape orientation. Applies to the Gallery control.

Text – Text that appears on a control or that the user types into a control. Applies to many controls.

Time – A media control's current position. Applies to **Audio** and **Video** controls.

Tooltip – Explanatory text that appears when the user hovers over a control. Applies to many controls.

Transition – The visual effect (**Pop**, **Push**, or **None**) when the user hovers over an item in a gallery. Applies to the **Gallery** control.

Transparency – The degree to which controls behind an image remain visible. Applies to the **Image** control. Decimal values range from 0 to 1. Where 0 is opaque, 0.5 is semi-transparent and 1 is transparent.

U

Underline – Whether a line appears under the text that appears on a control. Applies to many controls.

Unsaved – True if the **Edit form** control contains user changes that have not been saved. Applies to the **Edit form** control.

Update – The value to write back to the data source for a field. Applies to the **Card** control.

Updates – The values to write back to the data source for a record loaded in a form control. Applies to the **Edit form** control.

V

Valid – Whether a **Card** or **Edit form** control contains valid entries, ready to be submitted to the data source. Applies to **Card** and **Edit form** controls.

Value – The value of an input control. Applies to **Check box**, **Radio**, **Slider**, and **Toggle** controls.

ValueFill – The background color of the rectangle in a toggle control when its value is **true** or the color of the line to the left of the handle in a slider control. Applies to **Slider** and **Toggle** controls.

ValueHoverFill – When you keep the mouse pointer on a toggle control or a slider, the background color of the rectangle in a toggle control when its value is **true** or the color

of the line to the left of the handle in a slider control. Applies to **Slider** and **Toggle** controls.

VerticalAlign – The location of text on a control in relation to the vertical center of that control. Applies to many controls.

Visible – Whether a control appears or is hidden. Applies to many controls.

W

Width – The distance between a control's left and right edges. Applies to many controls.

WidthFit – Whether a control automatically grows horizontally to fill any empty space of a container control such as an **Edit form** control. If multiple cards have this property set to **true**, the space is divided between them. For more information, see Understand data form layout.

Wrap – Whether text that's too long to fit in a label wraps to the next line. Applies to the **Label** control.

WrapCount – How many records appear in each item of a gallery. Applies to the **Gallery** control.

X

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container). Applies to many controls. For a **Card** control in a container that has multiple columns, this property determines the column in which the card appears.

XLabelAngle – The angle of the labels below the x-axis of a column or line chart. Applies to **Column chart** and **Line chart** controls.

Y

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container). Applies to many controls. For a **Card** control in a container that has multiple rows, this property determines the row in which the card appears.

YAxisMax – The maximum value of the y-axis for a line chart. Applies to the **Line chart** control.

YAxisMin – The minimum value of the y-axis for a line chart. Applies to the **Line chart** control.

YLabelAngle – The angle of the labels next to the y-axis of a line or column chart. Applies to **Column chart** and **Line chart** controls.

Z

Zoom – The percentage by which an image from a camera is magnified or the view of a file in a PDF viewer. Applies to **Camera** and **PDF viewer** controls.

See also

Limitations of controls in Power Apps

Limitations of controls in canvas apps

Article • 03/13/2023

In this article, learn about the general limitations of the controls used in canvas apps. In addition, limitations of controls in Teams lists the control limitations applicable to apps played inside Microsoft Teams. For more information about environments, go to Types of environments.

- Attachments control - see Attachments control limitations.
- Audio, and video controls - see Audio and video alternatives.
- Barcode control, Camera control
  - On iOS, the camera control is supported in the Power Apps for mobile app. It's not supported within the browser or Teams Mobile.
  - On Android, the camera control isn't supported in Teams Mobile.
  - When using desktop browsers, the barcode reader isn't supported. Use the Power Apps for mobile app.
- Container control - see Container control limitations.
- Export and import controls - see Export and import control limitations.
- Gallery control - see Gallery control limitations.
- Image control - only supports external media URLs using HTTPS.
- Power BI tile control - see Power BI tile control limitations for passing a parameter.
- PDF viewer control (experimental) - see PDF viewer control (experimental) limitations.
- Stream Video control - see Stream Video control limitations.

Limitations of controls in Teams

This section lists limitations of canvas apps controls for apps within Microsoft Teams.

General limitations

Apps made with Power Apps in a Dataverse for Teams environment can't be opened outside of Teams. To use controls like *Camera*, upgrade your Power Apps license so that the app can be opened outside of Teams—for example—in the Power Apps mobile app.

Limitations of specific controls

- Barcode control - the barcode reader control isn't supported in Teams Mobile.
- Camera control - the Camera control isn't supported in Teams Mobile. If you're creating a Power Apps app for use inside Teams, use the Add picture control instead.
- Microphone control - the following conditions apply when using the Microphone control in apps created using Power Apps on Teams Mobile:
  - The audio format for microphone recordings in Teams will always be *AAC* with a file extension of .MP4.
  - Teams has its own recording experience. The microphone control inside apps created using Power Apps will be disabled during the recording period.
  - Microphone recordings are limited to a maximum duration of 10 minutes.

Unsupported controls

The following controls aren't supported by apps embedded in Teams:

- Address Input
- Camera (in Teams Mobile)
- Map
- Mixed reality (View in MR, View shape in MR, Measuring Camera)
- Power BI tile control
- Web barcode scanner control

Next steps

Controls and properties reference

Add picture control in canvas apps

Article • 03/25/2024

Takes a photo or loads images from the local device.

Description

With this control users can take photos or upload image files from their device and update the data source with this content. On a mobile device the user is presented with the device's choice dialog to choose between taking a photo or selecting one already available.

This control is a grouped control containing two controls: an **Image** and an **Add picture button**. The **Image** control shows the uploaded image or a placeholder if no image has been uploaded. The **Add picture button** prompts for an image to be uploaded.

See the Image control reference for **Image** properties.

Limitations

The picture control has these limitations:

- When using the **Add Picture** control on iOS, images imported by using the camera or gallery don't contain meta-data like orientation, location information, or camera settings.

Add picture button properties

AccessibleLabel – Label for screen readers. Should describe the purpose of adding a picture.

Align – The location of text in relation to the horizontal center of its control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

ChangePictureText – Text that appears on the button when an image has been uploaded.

Color – The color of text in a control.

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Error - If there is a problem uploading an image, this property will contain an appropriate error string.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

Media – An identifier for the clip that an audio or video control plays.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnSelect – Actions to perform when the user taps or clicks a control.

Padding – The distance between the text on an import or export button and the edges of that button.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

Reset – Whether a control reverts to its default value.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Text – Text that appears on the button when an image has not been uploaded.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

UseMobileCamera – Whether to use a mobile camera directly, when available. For Android devices, this property requires the experimental setting **Improved Media Capture** turned on.

VerticalAlign – The location of text on a control in relation to the vertical center of that control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Patch(*DataSource*, *BaseRecord*, *ChangeRecord*)

Examples

Add images to an Image gallery control

1. Add an **Add picture** control, and then triple-click it.

   Don't know how to add, name, and configure a control?

2. In the **Open** dialog box, click or tap an image file, and then click or tap **Open**.

3. Add a **Button** control, move it under the **Add picture** control, and set the **OnSelect** property for the **Button** control to this formula:
   Collect(MyPix, AddMediaButton1.Media)

   Want more information about the **Collect** function or other functions?

4. Add a **Vertical gallery** control, and set its **Items** property to **MyPix**.

5. Select the **Image** control in the gallery and set it's **Image** property to **ThisItem.Value**.

6. Press F5, and then click or tap the **Button** control.

   The image from the **Add picture** control appears in the **Vertical Gallery** control. If your image doesn't have the same aspect ratio as the **Image** control in the **Vertical gallery** control, set the **ImagePosition** property of the **Image** control to **Fit**.

7. Click or tap the **Add picture** control, click or tap another image file, click or tap **Open**, and then click or tap the **Button** control that you added.

   The second image appears in the **Image gallery** control.

8. (optional) Repeat the previous step one or more times, and then return to the default workspace by pressing Esc.

Use the **SaveData** function to save the images locally or the **Patch** function to update a data source.

Accessibility guidelines

The same guidelines for **Button** and **Image** apply. In addition, consider the following:

Color contrast

- **Add picture button** must have adequate contrast between its text and background. Since the uploaded image may have varying colors, use an opaque Fill on the **Add picture button** to ensure consistent contrast.

Screen reader support

- **Add picture button** must have **Text** and **ChangePictureText** that prompts the user to add or change a picture.

Keyboard support

- **Add picture button** must have TabIndex of zero or greater so that keyboard users can navigate to it.
- **Add picture button** must have clearly visible focus indicators. Use FocusedBorderColor and FocusedBorderThickness to achieve this.

Use the address input control to easily enter addresses

Article • 07/15/2022

Entering addresses can be frustrating and error-prone, particularly in mobile apps. Use the address input control to make address entry easier. The control uses fuzzy logic to suggest potential matches as you type. Select the one you want to quickly and easily enter an accurate address.

The control returns the address as structured data. Your app can extract information such as city, street, municipality, and even latitude and longitude coordinates. The data is in a format that's friendly to many locales and international address formats.

Prerequisites

Before you can use the control in your apps, you'll need to enable geospatial features for the environment. Make sure you also review the prerequisites for using geospatial controls. Refer to the privacy and security table for more details on the address input control's data usage.

Add an address input control to your app

With your app open for editing in Power Apps Studio :

1. Open the **Insert** tab and expand **Input**.
2. Select **Address input** to place an address input box in the app screen, or drag the control to the screen to position it more precisely. You must enter at least three characters including one number for the address input control.

Set a default search radius

By default, the control will search around the user's location. You can refine the default search area to help narrow the initial results.

1. On the address input control's **Properties** tab, turn on the **Search within radius** property.
2. Enter a longitude, latitude, and radius in meters.

The control will start searching at the given latitude and longitude, out to the distance specified.

Use the map control with the address input control

You can add a button to your app to save entered addresses as a data collection. Then you can retrieve the addresses and display them in the map control.

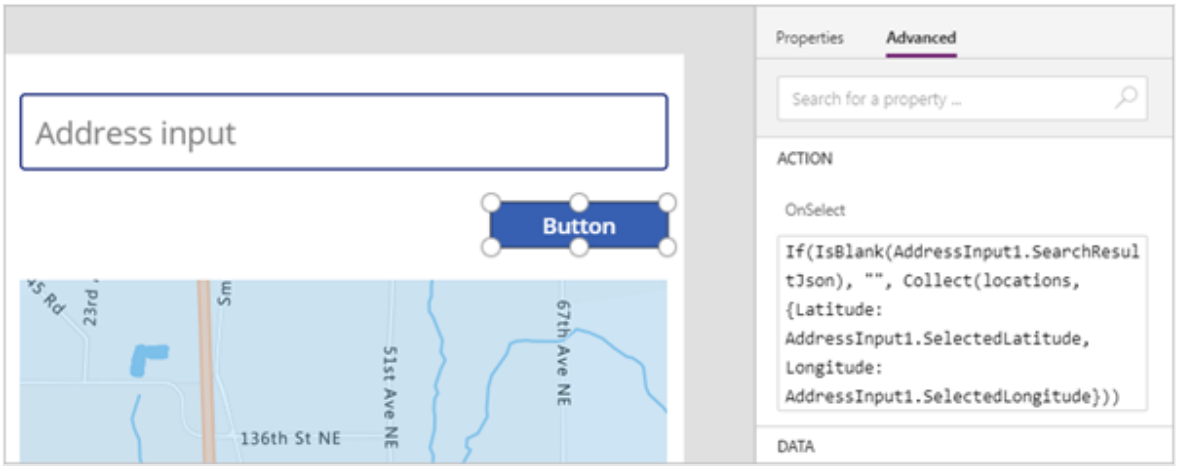
1. Add a map control and an address input control to your app.
2. Insert and place a **Button** control.
3. Change the **OnSelect** property of the button control as follows. (Hint: Copy the formula and paste it in the formula bar or on the **Advanced** properties tab, whichever you prefer.)
4. Enter a longitude, latitude, and radius (in meters).

   The control will start searching at the latitude and longitude, out to the distance specified in the radius field.

   Power Fx

   ```
   If(IsBlank(AddressInput1.SearchResultJson), "", Collect(locations,
   {Latitude: AddressInput1.SelectedLatitude, Longitude:
   AddressInput1.SelectedLongitude}))
   ```

   The formula saves the current latitude and longitude to a collection named *locations*, as long as the search results aren't blank.

5. Select the map control. Change its properties as follows:

Expand table

Property name	Value	Where
Items	"Locations"	**Properties** tab
ItemsLatitudes	"Latitude"	**Advanced** tab
ItemsLongitudes	"Longitude"	**Advanced** tab

When the user selects the button, the result from the address input control is added to the map as a new pin.

Properties

Change an address input control's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.

Expand table

Property	Description	Type	Tab
Default	Sets the initial value of the control.	String	Properties; Advanced: **Default**
Hint text	Sets the hint that appears in the control before the user enters text.	String	Properties; Advanced: **HintText**
Font	Sets the name of the family of fonts used for the control text.	Dropdown list	Properties; Advanced: **Font**
Font size	Sets the size of the control text.	Floating point number	Properties; Advanced: **FontSize**
Font weight	Sets the weight of the control text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	Dropdown list	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the control text, either *Center*, *Justify*, *Left*, or *Right*.	Dropdown list	Properties; Advanced: **TextAlignment**
Line height	Sets the vertical distance between lines of text in the control.	Floating point number	Properties; Advanced: **LineHeight**

Property	Description	Type	Tab
Display mode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Dropdown list	Properties; Advanced: DisplayMode
Font style	Sets the style of the control text, either *Italic*, *Underline*, *Strikethrough*, or none.	Dropdown list	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Search result limit	Sets the number of suggested addresses the control displays.	Integer	Properties; Advanced: **SearchResultLimit**
Search within radius	Determines whether the control should suggest addresses within the **Radius** of the **Latitude** and **Longitude**.	Boolean	Properties; Advanced: **SearchWithinRadius**
Latitude	Sets the latitude coordinate of the center point used for address suggestions. Requires **Search within radius** to be on.	Floating point number from -90 to 90	Properties; Advanced: **Latitude**
Longitude	Sets the longitude coordinate of the center point used for address suggestions. Requires **Search within radius** to be on.	Floating point number from -180 to 180	Properties; Advanced: **Longitude**
Radius	Sets the radius, in meters, around **Latitude** and **Longitude** to constrain address suggestions. Requires **Search within radius** to be on.	Floating point number	Properties; Advanced: **Radius**
Language	Sets the language that address suggestions are returned in.	String	Properties; Advanced: **Language**
Country set	Identifies a comma-separated list of countries/regions to constrain address suggestions to, in ISO 3166 alpha-2 format; for example, *US,CA,MX*.	String	Properties; Advanced: **CountrySet**
Visible	Shows or hides the control.	Boolean	Properties; Advanced: Visible
Padding top	Sets the distance between the control text and the top of the	Floating point number	Properties; Advanced: PaddingTop

Property	Description	Type	Tab
	control.		
Padding bottom	Sets the distance between the control text and the bottom of the control.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the control text and the left edge of the control.	Floating point number	Properties; Advanced: **PaddingLeft**
Padding right	Sets the distance between the control text and the right edge of the control.	Floating point number	Properties; Advanced: **PaddingRight**
Position	Places the upper-left corner of the control at the screen coordinates specified in *X* and *Y*.	Integer	Properties; Advanced: **X**, **Y**
Size	Determines the size of the control using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Border radius	Determines the corner radius of the control border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the control text and the control background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the control border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderColor**
Tab index	Specifies the order the control is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **TabIndex**
Tooltip	Determines the text to display when the user hovers over the control.	String	Properties; Advanced: **Tooltip**
Hover color	Sets the colors of the control text, the control background, and the control border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverFontColor**, **HoverFillColor**, **HoverBorderColor**

Property	Description	Type	Tab
Disabled color	Sets the colors of the control text, the control background, and the control border if **DisplayMode** is *Disabled*.	Not applicable	Properties; Advanced: **DisabledFontColor**, **DisabledFillColor**, **DisabledBorderColor**
Pressed color	Sets the colors of the control text, the control background, and the control border when the user selects the control.	Not applicable	Properties; Advanced: **PressedFontColor**, **PressedFillColor**, **PressedBorderColor**
ContentLanguage	Determines the display language of the control, if it's different from the language that's used in the app.	String	Advanced
OnAddressSelect	Contains code that runs when the user selects a suggested address.	Event	Advanced
OnChange	Contains code that runs when a control property is changed.	Event	Advanced

Output properties

Other properties become available when a user interacts with the address input control. You can use these *output properties* in other controls or to customize the app experience.

Expand table

Property	Description
UserInput	The text the user typed in the input box
SelectedLatitude	The latitude of the address that the user selected
SelectedLongitude	The longitude of the address that the user selected
SearchResultJson	The search result, based on **UserInput**, displayed as a string in JSON format
FreeformAddress	The address that the user selected from the list of suggested addresses
LocalName	The name of a geographic area or locality that groups a number of addressable objects for addressing purposes, without being an

Property	Description
	administrative unit
PostalCode	The postal code
ExtendedPostalCode	The extended postal code
CountryCode	The country code
Country	The country/region name
CountryCodeISO3	The country code in ISO alpha-3 format
CountrySubdivisionName	The country/region subdivision name
StreetName	The street name
StreetNumber	The street number
Municipality	The municipality
MunicipalitySubdivision	The municipality subdivision
CountryTertiarySubdivision	The country/region tertiary subdivision
CountrySecondarySubdivision	The country/region secondary subdivision
CountrySubdivision	The country/region subdivision

Other geospatial controls

To visualize and interpret location data, use the **interactive map** control.

Feedback

Was this page helpful? Yes No

Provide product feedback

Attachments control in Power Apps

Summarize this article for me

A control that users can use to download files to their device, and upload and delete files from a list created by using Microsoft Lists or a Microsoft Dataverse table.

Description

By using an **Attachments** control, you can open, add, and delete files from a list or a Dataverse table.

Limitations

The attachment control has these limitations:

1. The attachment control only supports lists and Dataverse tables as data sources. It doesn't support expressions that transform these data sources into tables, such as with the **ForAll** function.

2. The upload and delete functionality works only inside a form. The attachment control appears disabled when in **Edit** mode and not inside a form. To save file additions and deletions, the app user must save the form. Because of this limitation, the attachment control isn't available from the **Insert** tab but appears in the form when the attachment form field is enabled in a SharePoint or Dataverse form.

3. The attachments control on a web browser lets you select multiple files and use drag and drop functionality. However, when you use the attachments control on Power Apps Mobile, you can only add files one at a time.

4. You might not be able to attach files stored in OneDrive or any other cloud storage. Try downloading these files to the device before attaching them by using the device's file picker.

5. The Power Apps mobile app doesn't support capturing videos directly from the camera or selecting videos from the photo library when using the **Attachments** control. To add a video, users must use the **Browse** option.

    - **On Android**: The **Browse** option displays all file types, including videos, so you can select and attach your video files.
    - **On iOS**: The **Browse** option only shows files stored in the **Files** app. If your video isn't already saved there, you need to move it to the **Files** app before you attach it.

If a video doesn't appear as an option to attach, first save the video to your device's **Files** app and then try again.

6. Uploading and viewing attachments isn't supported when running canvas apps in Microsoft Teams on Android or iOS mobile devices.

Key properties

Items – The **Attachment** column of the list or Dataverse table. Collections and Tables aren't supported.

MaxAttachments – The maximum number of files the control accepts.

MaxAttachmentSize – The maximum allowed file size in MB for each new attachment. 1 MB here is 1,000,000 bytes (10^6 B) or 1,000 KB.

OnAddFile – Actions to perform when the user adds a new file attachment.

OnRemoveFile – Actions to perform when the user deletes an existing attachment.

OnUndoRemoveFile – Actions to perform when the user restores a deleted attachment.

Additional properties

AccessibleLabel – Label for screen readers. Should describe the purpose of the attachments.

AddAttachmentText – The label text for the link used to add a new attachment.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of a control's text.

DisabledBorderColor – The color of a control's border if the control's **DisplayMode** property is set to **Disabled**.

DisabledColor – The color of text in a control if its **DisplayMode** property is set to **Disabled**.

DisabledFill – The background color of a control if its **DisplayMode** property is set to **Disabled**.

DisplayMode – Whether the control allows adding and deleting files (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

[DropTargetBackgroundColor] – The color of the control's drop target background.

[DropTargetBorderColor] – The color of the control's drop target border.

[DropTargetBorderStyle] – Whether the control's drop target border is **Solid**, **Dashed**, **Dotted**, or **None**.

[DropTargetBorderThickness] – The thickness of the control's drop target border.

[DropTargetTextColor] – The color of the control's drop target text.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

MaxAttachmentsText – The text that replaces the "Attach file" link when the control contains the maximum number of files allowed.

NoAttachmentsText – Informational text shown to the user when there are no files attached.

Padding – The distance between the text on an import or export button and the edges of that button.

PressedBorderColor – The color of a control's border when the user selects that control.

PressedColor – The color of text in a control when the user selects that control.

PressedFill – The background color of a control when the user selects that control.

Reset – Reverts all changes to the attachments control and returns to the previously saved state.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

Visible – Whether a control is visible or hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (or screen, if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (or screen, if no parent container).

Example

1. Add a form to your app, and set a list as its data source.
2. Select the **Display Form** control in the tree view on the left-hand side. You can also use **Edit Form** instead.
3. Select **Data Source** in the Properties tab in the options panel on the right and then select the list you connected to.
4. Select **Edit fields** in the *Fields* section and select **Add field**.
5. Select the **Attachments** field and select **Add**.

   The Attachments field associated with the list appears in the form.

Learn how to add and configure a control

Accessibility guidelines

Color contrast

Ensure there's adequate color contrast between:

- **ItemColor** and **ItemFill**

- **ItemHoverColor** and **ItemHoverFill**
- **ItemPressedColor** and **ItemPressedFill**
- **AddedItemColor** and **AddedItemFill**
- **RemovedItemColor** and **RemovedItemFill**
- **ItemErrorColor** and **ItemErrorFill**
- **AddAttachmentColor** and **Fill**
- **MaxAttachmentsColor** and **Fill**
- **NoAttachmentsColor** and **Fill**

This requirement is in addition to the standard color contrast requirements.

Screen reader support

Include the following properties:

- **AccessibleLabel**
- **AddAttachmentsText**
- **MaxAttachmentsText**
- **NoAttachmentsText**

Keyboard support

- Use **TabIndex** with a value of zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this clarity.

Last updated on 03/11/2026

Audio and Video controls in Power Apps (retired)

Article • 04/03/2024

A control that plays an audio file, a video file, or a video on YouTube.

Description

An **Audio** control plays a sound clip from a file, a recording from a Microphone control, or the audio track from a video file.

A **Video** control plays a video clip from a file or from YouTube or any external source. Closed captions can optionally be shown when specified.

> **Important**
>
> Videos on Azure Media services are no longer supported. For more information, see Retirement of the service.

Limitations

The audio and video control has these limitations:

- Power Apps Mobile runs on various types of devices, and some of them have limitations that are specific to that platform:
  - You can play videos in multiple **Video** controls at the same time on all platforms except iPhone devices.
  - You can record audio with multiple **Microphone** controls at the same time on all platforms except the web player.

Key properties

Loop – Whether an audio or video clip automatically starts over as soon as it finishes playing.

Media – An identifier for the clip that an audio or video control plays.

> **Note**

- Use HTTPS for all external videos to ensure compatibility with modern browsers.
- External videos must be accessible anonymously (without any authentication).

ShowControls – Whether an audio or video player shows, for example, a play button and a volume slider, and a pen control shows, for example, icons for drawing, erasing, and clearing.

Additional properties

AccessibleLabel – Label for screen readers. Should be the title of the video or audio clip.

AutoPause – Whether an audio or video clip automatically pauses if the user navigates to a different screen.

AutoStart – Whether an audio or video control automatically starts to play a clip when the user navigates to the screen that contains that control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

ClosedCaptionsUrl – Video control only. URL of closed captions file in WebVTT format. Both video and captions URLs must be HTTPS. Server hosting both video and captions file needs to be CORS enabled.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

Image – The name of the image that appears in an image, audio, or microphone control.

ImagePosition – The position (**Fill**, **Fit**, **Stretch**, **Tile**, or **Center**) of an image in a screen or a control if it isn't the same size as the image.

OnEnd – Actions to perform when an audio or video clip finishes playing.

OnPause – Actions to perform when the user pauses the clip that an audio or video control is playing.

OnStart – Actions to perform when the user starts to record with a microphone control.

Paused – *True* if a media playback control is currently paused, *false* otherwise.

Reset – Whether a control reverts to its default value.

Start – Whether an audio or video clip plays.

StartTime – The time after the start of an audio or video clip when the clip starts to play.

Time – A media control's current position.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

First(*TableName*)

Examples

Play an audio or video file

1. On the **File** menu, click or tap **Media**, click or tap **Videos** or **Audio**, and then click or tap **Browse**.

2. Browse to the file you want to use, click or tap it, and then click or tap **Open**.

3. Press Esc to return to the default workspace, add an **Audio** or **Video** control, and set its **Media** property to the file that you added.

   Don't know how to add and configure a control?

4. Press F5, and then play the clip by clicking or tapping the play button of the control that you added.

   > **Tip**
   >
   > The play button of the **Video** control appears when you hover over the control.

5. Press Esc to return to the default workspace.

Play a YouTube video

1. Add a **Video** control, and set its **Media** property to the URL of the YouTube video, enclosed in double quotation marks.
2. Press F5, and then play the clip by clicking or tapping the play button of the **Video** control.
3. Press Esc to return to the default workspace.

Accessibility guidelines

Audio and video alternatives

- **ShowControls** must be true so that users can listen or watch multimedia at their own pace. This also allows users to toggle closed captions and full-screen mode on video players.
- Closed captions must be provided for videos.
  - For YouTube videos, use authoring tools provided by YouTube to add captions.
  - For other videos, create captions in WebVTT format, upload them, and set **ClosedCaptionsUrl** to the url location. There are several limitations. Server(s) hosting video and captions needs to be CORS-enabled and serve them using HTTPS protocol.
- Consider providing an audio or video transcript using one of these methods:

  1. Put the text in a **Label** and position it adjacent to the multimedia player. Optionally, create a **Button** to toggle the display of the text.

2. Put the text in a different screen. Create a **Button** that navigates to the screen and position the button adjacent to the multimedia player.
3. If the description is short, it can be put it in the **AccessibleLabel**.

Color contrast

There must be adequate color contrast between:

- **FocusedBorderColor** and the outside color
- **Image** and the multimedia player controls (if applicable)
- **Fill** and the multimedia player controls (if the fill is visible)

Provide closed captions and/or transcript if the video content has color contrast issues.

Screen reader support

- **AccessibleLabel** must be present.

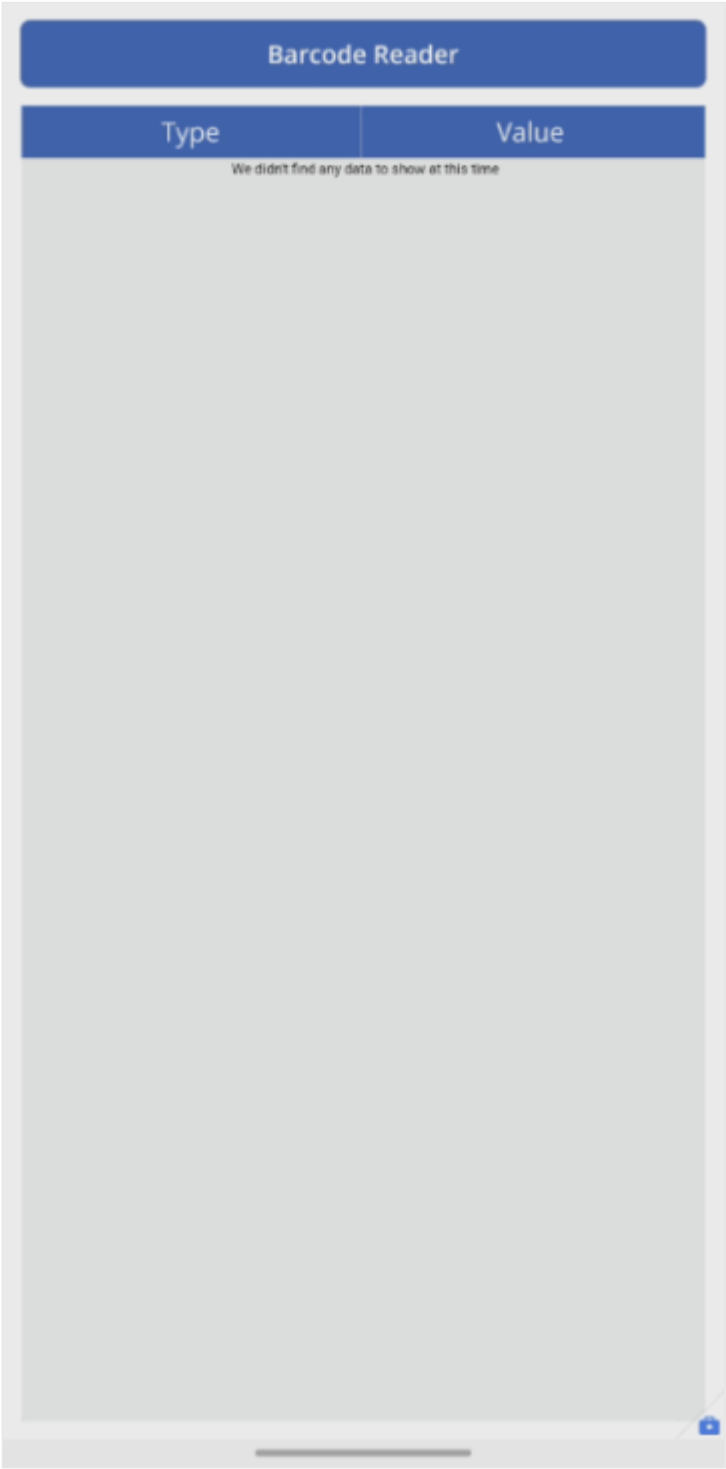
Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.
- **AutoStart** should be false because it can be difficult for keyboard users to stop playback quickly.

Barcode reader control in Power Apps

Article • 04/25/2023

Scans barcodes, QR codes, and data-matrix codes on Android, iOS, and Windows devices.



Description

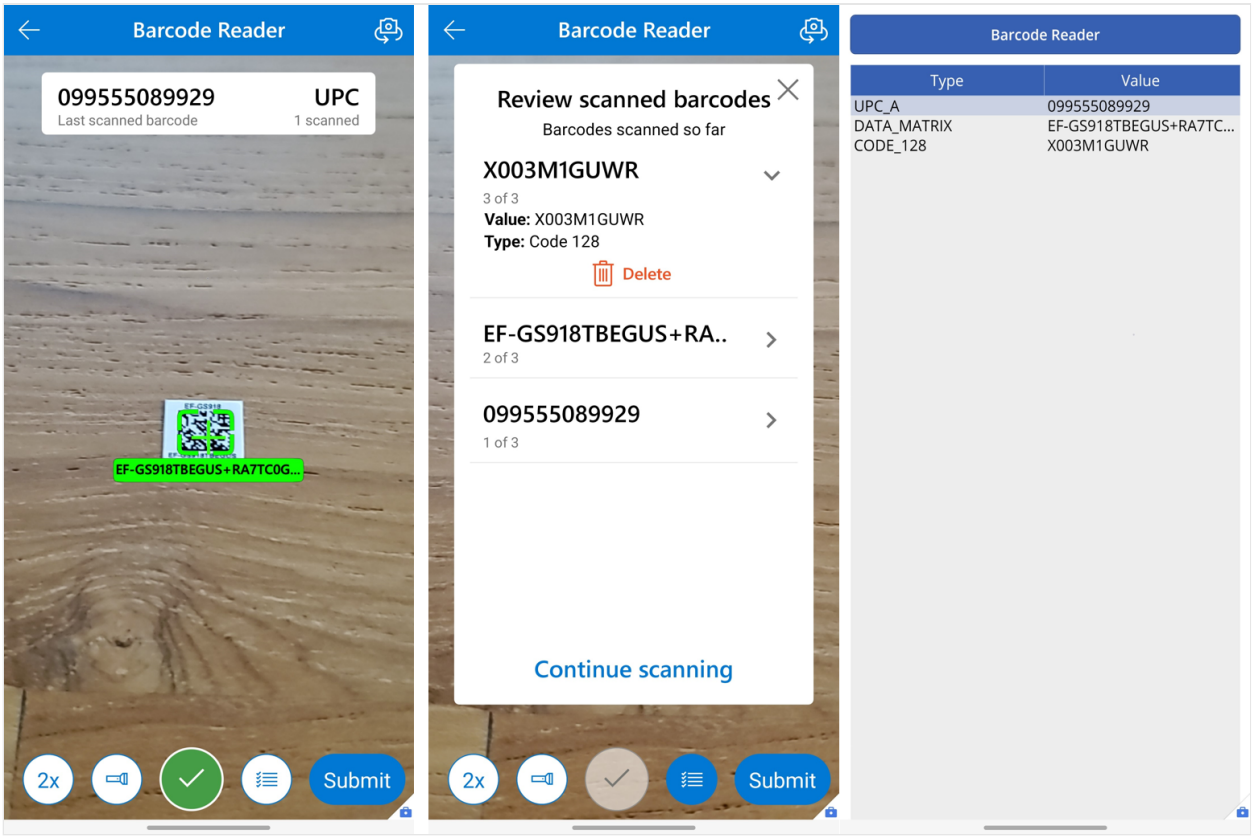
The control opens a native scanner on Android, iOS, and Windows devices. On iOS and Android, the scanner supports the following scanning modes:

- **Automatically scan** – A barcode is scanned as soon as it's detected.
- **Select to scan** – The user can determine which of the detected barcodes to scan. The control doesn't support scanning in a web browser.
- **Scan multiple** – The user can scan multiple barcodes during the barcode session.

On Windows devices, the barcode reader will always **automatically scan**.

The barcode reader control supports different zoom levels and turning on/off the device's flashlight from within the scanning experience.

> **Note**
>
> The barcode scanner control is supported on Android, iOS, and Windows devices. All other platforms will show a warning stating that some features of the app won't work.



Key properties

Barcodes – Output property that contains a table of the barcodes scanned with two columns: **Value**, and **Type**. **Value** is the text value of the code that was scanned, while **Type** is the type of the code that was scanned.

OnScan – Actions to perform when a barcode is successfully scanned.

OnCancel – Actions to perform when a barcode scan is canceled by the user.

OnChange - Actions to perform when a property on the barcode reader control is changed, including output properties.

BarcodeType - The barcode type to scan. You can target multiple barcode types by concatenating them. Ex. `'Microsoft.BarcodeReader.BarcodeType'.Code128 & 'Microsoft.BarcodeReader.BarcodeType'.Code39`. **Default: Auto**

Scanning mode - Whether to `Automatically scan` the first barcode detected in view, to allow the user to `Select to scan` which of the barcodes in view to scan, or to `Scan multiple` barcodes in a single session. On Windows, the barcode reader will always `Automatically scan`.

Scanning quality - The image quality the barcode reader scans at. A higher quality is good for small barcodes, but could lead to slow performance. On Windows, the barcode reader will always scan at the device's camera quality. **Default: Automatic**.

PreferFrontCamera - If enabled the barcode reader will default to using the front facing camera instead of the rear facing camera.

BeepOnScan - If enabled, the barcode reader will play a beep noise when a barcode is scanned.

VibrateOnScan - If enabled, the barcode reader will vibrate when a barcode is scanned.

Additional properties

Text - Text that appears on the button that activates the scanner.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Height – The height of the button that activates the scanner.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The width of the button that activates the scanner.

X – The distance between the left edge of a control and the left edge of its parent container (screen, if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen, if no parent container).

Accessibility guidelines

The same guidelines for the **Button** control apply to the **Barcode reader** control because it's a button that launches the scan.

Visual alternatives

- The barcode scanner is a button that doesn't display the scan result. Consider showing the scan result with a **Label** control. Set the label's **Text** property to `First(BarcodeReader.Barcodes).Value` where `BarcodeReader` is the name of the barcode reader control. Set the label's **Live** property to **Polite** so that screen-reader users are notified of changes. This change makes the scanned value accessible to everyone, irrespective of visual ability.
- Users who have visual and motor disabilities might prefer not to point the camera at a barcode. Consider adding another form of input, such as a **Text input** control, for users to enter barcodes.

Barcode Availability by Device

Barcode Type	Supported on iOS and Android	Supported on Windows	Notes
QR_CODE	✓	✓	
DATA_MATRIX	✓	✕	
AZTEC	✓	✕	
CODABAR	✓	✓	
CODE_128	✓	✓	
CODE_39	✓	✓	
CODE_93	✓	✕	

Barcode Type	Supported on iOS and Android	Supported on Windows	Notes
EAN	✓	✓	Supports EAN_8 and EAN_13
Interleaved 2 of 5 ITF	✓	✕	
PDF_417	✓	✕	
RSS14 Databar 14	✓	✕	Android and iOS support Stacked and Omnidirectional
RSS_EXPANDED Databar Expanded	✓	✕	Android and iOS support Stacked and Omnidirectional
UPC	✓	✓	Supports UPC_A and UPC_E
Micro QR Code	✓	✕	
GS1-DWCode	✕	✕	
MSI	✕	✕	

Select() barcode reader control

The barcode reader control doesn't support Select(). As a workaround, set the barcode reader fill color to transparent and put it in front of the control you wanted to select the barcode reader with. This will result in the same experience for the app user where selecting the control launches the barcode reader experience as they will be selecting an invisible barcode reader button instead of the actual control.

See also

Limitations of controls in Power Apps

Barcode scanner control (retired) in Power Apps

Article • 03/13/2023

The legacy control for scanning barcodes, QR codes, and data-matrix codes on an Android, iOS, and Windows device.

> **Important**
>
> The barcode scanner control is retired as it is replaced by the barcode reader control. While the legacy barcode scanner's existing functionality will continue to be supported in future versions of Power Apps, the barcode reader includes the latest improvements and should be used where possible.

Prerequisites

The barcode scanner control is retired and needs to be enabled to show up in the insert pane.

With your app open for editing in Power Apps Studio:

1. Open **Settings**
2. Select **Upcoming features** in the left-hand menu and select the **Retired** tab
3. Turn on the **Legacy barcode scanner** setting

Description

The control opens a native scanner on an Android, iOS, and Windows device. The scanner automatically detects a barcode, a QR code, or a data-matrix code when in view. The control doesn't support scanning in a web browser.

> **Note**
>
> The barcode scanner control is supported on an Android, iOS, and Windows device. All other platforms will show a warning that some features of the app won't work.

Key properties

Value – Output property that contains the text value of the code that was scanned most recently.

Type – Output property that contains the type of the code that was scanned most recently.

OnScan – Actions to perform when a barcode is successfully scanned.

OnCancel – Actions to perform when a barcode scan is canceled by the user.

BarcodeType - The barcode type to scan. You can target multiple barcode types by concatenating them. Ex. BarcodeType.Code128 & BarcodeType.Code39 **Default: Auto**

PreferFrontCamera - Whether the front-facing camera, when available, is used for scanning.

FlashlightEnabled - Whether the flashlight is enabled automatically when the scanner is opened.

Additional properties

Text - Text that appears on the button that activates the scanner.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Height – The height of the button that activates the scanner.

Tooltip – Explanatory text that appears when the user hovers over a control.

Type - The type of code that was detected in the scan that succeeded most recently.

Visible – Whether a control appears or is hidden.

Width – The width of the button that activates the scanner.

X – The distance between the left edge of a control and the left edge of its parent container (screen, if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen, if no parent container).

Accessibility guidelines

The same guidelines for the **Button** control apply to the **Barcode scanner** control because it's a button that launches the scan.

Visual alternatives

- The barcode scanner is a button that doesn't display the scan result. Consider showing the scan result with a **Label** control. Set the label's **Text** property to the barcode scanner's **Value** property. Set the label's **Live** property to **Polite** so that screen-reader users are notified of changes. This change makes the scanned value accessible to everyone, irrespective of visual ability.
- Users who have visual and motor disabilities might prefer not to point the camera at a barcode. Consider adding another form of input, such as a **Text input** control, for users to enter barcodes.

Barcode Availability by Device

Expand table

Barcode Type	Android	iOS	Windows
QR_CODE	✓	✓	✓
DATA_MATRIX	✓	✓	✕
UPC_A	✓	✓	✓
UPC_E	✓	✓	✓
EAN_8	✓	✓	✓
EAN_13	✓	✓	✓
CODE_39	✓	✓	✓
CODE_93	✓	✓	✕
CODE_128	✓	✓	✓
CODABAR	✓	✕	✓
Databar Omnidirectional	✕	✕	✓
Databar Stacked	✕	✕	✓

Barcode Type	Android	iOS	Windows
GS1-DWCode	✗	✗	✓
ITF	✓	✓	✗
Interleaved 2 of 5	✗	✗	✓
RSS14	✓	✗	✗
PDF_417	✓	✓	✗
RSS_EXPANDED	✓	✗	✗
MSI	✗	✗	✗
AZTEC	✓	✓	✗

> **Note**
>
> PDF_417 and AZTEC aren't supported in Auto mode.

See also

Limitations of controls in Power Apps

Button control in Power Apps

Article • 05/23/2025

A control that the user can click or tap to interact with the app.

Description

Configure the OnSelect property of a **Button** control to run one or more formulas when the user clicks or taps the control.

Key properties

OnSelect – Actions to perform when the user taps or clicks a control.

Text – Text that appears on a control or that the user types into a control.

Additional properties

Align – The location of text in relation to the horizontal center of its control.

AutoDisableOnSelect – Automatically disables the control while the **OnSelect** behavior is running.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

ContentLanguage - The language of the control's content, if different from the control's container.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Fill – The background color of a control.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

Pressed – *True* while a control is being pressed, *false* otherwise.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

VerticalAlign – The location of text on a control in relation to the vertical center of that control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Navigate(*ScreenName*, *ScreenTransitionValue*)

Examples

Add a basic formula to a button

1. Add a **Text input** control, and name it **Source**.

   Don't know how to add, name, and configure a control?

2. Add a **Button** control, set its **Text** property to "Add", and set its **OnSelect** property to this formula:
   UpdateContext({Total:Total + Value(Source.Text)})

   Want more information about the **UpdateContext** function or other functions?

3. Add a **Label** control, set its **Text** property in the formula bar to **Value(Total)**, and then press **F5**.

4. Clear the default text from **Source**, type a number in it, and then click or tap **Add**.

   The **Label** control shows the number that you typed.

5. Clear the number from **Source**, type another number in it, and then click or tap **Add**.

   The **Label** control shows the sum of the two numbers that you typed.

6. (optional) Repeat the previous step one or more times.
7. To return to the default workspace, press Esc (or click or tap the close icon in the upper-right corner).

Configure a button with multiple formulas

Add a formula that clears the **Text input** control between entries.

1. Set the HintText property of **Source** to "Enter a number".
2. Set the OnSelect property of **Add** to this formula:

   UpdateContext({Total:Total + Value(Source.Text)});
   UpdateContext({ClearInput: ""})

   > **Note**
   >
   > Separate multiple formulas with a semi-colon "**;**".

3. Set the Default property of **Source** to **ClearInput**.
4. Press **F5**, and then test the app by adding several numbers together.

Add another button to reset the total

Add a second button to clear the total between calculations.

1. Add another **Button** control, set its Text property to "Clear", and set its OnSelect property to this formula:

   UpdateContext({Total:0})

2. Press **F5**, add several numbers together, and then click or tap **Clear** to reset the total.

Change a button's appearance

Change a button's shape

By default, Power Apps creates a rectangular **Button** control with rounded corners. You can make basic modifications to the shape of a **Button** control by setting its Height, Width, and Radius properties.

> ⓘ Note
>
> Icons and Shapes provide a wide variety of designs and can perform some of the same basic functions that **Button** controls do. However, Icons and Shapes don't have a Text property.

1. Add a **Button** control, and set its Height and Width properties to **300** to make a large square button.
2. Modify the RadiusTopLeft, RadiusTopRight, RadiusBottomLeft, and RadiusBottomRight properties to adjust the amount of curvature on each corner. Here are some examples of different shapes, each one starting from a 300 x 300 square button:
   - Set all four Radius values to **150** to create a circle.
   - Set the values for RadiusTopLeft and RadiusBottomRight to **300** to create a leaf-shaped **Button**.
   - Set the values for RadiusTopLeft and RadiusTopRight to **300**, and the values for RadiusBottomLeft and RadiusBottomRight to **100** to create a tab-shaped button.

Change a button's color when you hover over it

By default, the fill color of a **Button** control will dim by 20% when you hover over it with a mouse. You can adjust this behavior by changing the HoverFill property, which uses the ColorFade function. If you set the ColorFade formula to a positive percentage, the color becomes lighter when you hover over the button, while a negative percentage makes the color darker.

- Change the ColorFade percentage in the HoverFill property of one of the buttons that you created, and observe the effects.

You can also specify the color of a **Button** control by setting its HoverFill property to a formula that contains the ColorValue function instead of the ColorFade function, as in **ColorValue("Red")**.

> ⓘ Note
>
> The color value can be any CSS color definition, either a name or a hex value.

- Replace the ColorFade function with a ColorValue function in one of the buttons that you created, and observe the effects.

Accessibility guidelines

Color contrast

- Standard color contrast requirements apply.

Screen reader support

- **Text** must be present.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.

Camera control in Power Apps

Article • 12/16/2022

A control that enables users to take pictures using the camera on a device.

Description

Use the **Camera** control to capture pictures with a device's camera. The device must have a camera and the user must authorize the app to use the camera.

Select the camera control to capture a picture from the camera.

The most recently captured picture is available through the **Photo** property. With this property, the images can be:

- **Viewed with the Image control.** Use the Image control to view the captured image. For more information, see the examples.
- **Temporarily put in a variable or a collection.** Use the Set or Collect functions to store images in a variable or a collection. Use caution when using multiple images in a collection at the same time consuming device's limited memory. Use the SaveData and LoadData functions to move images to the local storage on the device and for offline scenarios.
- **Stored in a database.** Use the Patch function to store images in a database.
- **Transmitted as a base64 encoded text string.** Use the JSON function to base64 encode images.

Use the **Stream**, **StreamRate**, and **OnStream** properties to automatically capture images on a timer, for example snapping a picture every minute to create a time-lapse sequence.

Captured media is referenced by a text string URI. For more information, read the data type documentation.

> **Note**
>
> - The camera control is only supported on Microsoft Edge, Chrome, Firefox, and Opera browsers; and Android and iOS devices. All other browsers and platforms will show a warning that some features of the app won't work.
> - Images generated by the camera control have a maximum resolution of 640 x 480 px. If you need full resolution images, use the Add picture control

instead.

- Depending on the make and model of your mobile device, the device camera might take a few seconds to initialize when using the camera control.

Limitations

The camera control has these limitations:

1. When you use the camera control, the image doesn't contain meta-data information. This is due to a limitation of how we take images with the camera. To mitigate this issue, use the Add picture control
2. If your mobile device is low on memory, the camera is temporarily disabled to avoid crashing the device.
3. Power Apps for Windows might crash if you open an app that uses a camera control. To avoid this problem, use the web player on the Windows platform. Also, multiple cameras aren't supported.

Key properties

AvailableDevices – Table of the available cameras on the device.

Table contains two columns:

- *Id* number to be used with the **Camera** property
- *Name* provided by the device to identify the camera. Some platforms may include *Front* or *Back* to help locate the camera.

Note: Not all devices in the table may be usable in your app. Some may be specialized drivers or applications intended for specific purposes.

Camera – The numeric ID of the camera to use. Helpful on devices with more than one camera.

OnStream – Actions to perform when the **Stream** property is updated.

Photo – The image captured when the user takes a picture.

Stream – Automatically updated image based on the **StreamRate** property.

StreamRate – How often to update the image on the **Stream** property, in milliseconds. This value can range from 100 (1/10th of a second) to 3,600,000 (1 hour).

Additional properties

AccessibleLabel – Label for screen readers. Should describe the purpose of taking a picture.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Brightness – How much light the user is likely to perceive in an image.

Contrast – How easily the user can distinguish between similar colors in an image.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

OnSelect – Actions to perform when the user taps or clicks a control.

TabIndex – Keyboard navigation order compared to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container or screen.

Y – The distance between the top edge of a control and the top edge of the parent container or screen.

Examples

For these examples, you'll need a device with a camera. To test your app, use a web cam accessible from your browser. Or by saving your app and loading it into an iOS or Android device with a camera.

Simple display of a captured picture

1. Add a **Camera** control.
2. Authorize the app to use device's camera if prompted.
3. Add an **Image** control.
4. Set the **Image** property of the **Image** control to the following formula:

   ```
   Power Fx
   Camera1.Photo
   ```

   > **Note**
   >
   > Replace camera control name *Camera1* as appropriate.

5. Press F5 to preview your app.
6. Take a picture by selecting or tapping the camera control. You should see the result in your image control.

Add pictures to an Image gallery control

1. Add a **Camera** control, name it **MyCamera**, and set its OnSelect property to this formula:

   ```
   Power Fx
   Collect( MyPix, MyCamera.Photo )
   ```

   For more information:

   - How to add, name, and configure a control?
   - Read more about Collect function or other functions.

2. Press F5, and then take a picture by selecting or tapping **MyCamera**.
3. Add a Vertical gallery control. And then resize its Image control, its template, and the **Image gallery** control itself to fit in the screen.
4. Set the Items property of the **Image gallery** control to this formula:

```
Power Fx

MyPix
```

5. Set the Image property of the **Image** control in the gallery to this formula:

   ```
   Power Fx

   ThisItem.Url
   ```

   The picture that you took appears in the **Image gallery** control.

6. Take as many pictures as you want, and then return to the default workspace by pressing Esc.

7. (optional) Set the **OnSelect** property of the **Image** control in the **Image gallery** control to the formula:

   ```
   Power Fx

   Remove( MyPix, ThisItem )
   ```

8. Press F5, and then select a picture to remove it.

Use the SaveData function to save the pictures locally or the Patch function to update a data source.

Change the active camera from a drop down

1. Add a **Camera** control.

2. Authorize the app to use device's camera if prompted.

3. Add a Drop down control.

4. Set the **Items** property of the dropdown to:

   ```
   Power Fx

   Camera1.AvailableDevices
   ```

   > ⓘ **Note**

Replace camera control name *Camera1* as appropriate.

5. Set **Camera** property of the camera to:

```
Power Fx

Dropdown1.Selected.Id
```

> **Note**
>
> Replace dropdown control name *Dropdown1* as appropriate.

6. Press F5, and then select an item from the dropdown to change the camera.

Accessibility guidelines

The camera control shows camera feed and also functions as a button that takes a picture. So, there are similar accessibility considerations as with buttons.

Video alternatives

Consider adding an alternative form of input for users with visual disabilities. For example, Add picture to allow users to upload an image from their device.

Color contrast

There must be adequate color contrast between FocusedBorderColor and the outside color.

Screen reader support

AccessibleLabel must be present.

Keyboard support

- TabIndex must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use FocusedBorderColor and FocusedBorderThickness to update the visibility of focus indicators.

See also

Limitations of controls in Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Card control in Power Apps

Article • 12/16/2022

Provides the display and editing experience for a single field of a **Display form** or **Edit form** control.

Description

Display form and **Edit form** controls act as containers for displaying and viewing entire records. Each container can hold a set of **Card** controls that display individual fields or provide a way to update those fields. Each card has a **DataField** property that specifies which field of the record it works on.

Predefined cards are defined for different data types and user experiences. For example, there may be a card to edit a number field with a **Text input** control, which is great for use with the keyboard. Another card might support editing a number by using a **Slider** control instead. With the form control selected, you can, in the right-hand pane, easily select a card based on a field.

Cards themselves contain controls. The controls of a card make up the experience for displaying and editing a single field. For example, a number card may consist of a **Label** control to provide the display name of the field and a **Text input** control to provide an editor for the value of the field. The card may also have a **Label** control that shows any validation errors that occur and a **Label** control for the common asterisk to indicate that a field is required.

You can customize the controls of a predefined card by resizing it, moving it, hiding it, adding controls to it, and making other changes. You can also start with an entirely blank card, a "custom card", to which you add controls from scratch.

Predefined cards are *locked* by default. In a locked card, you can modify only certain properties of the card or the controls within the card, and you can't delete a locked card. You can show the card lock and unlock it on the **View** tab of the **Advanced** view. If a property is locked and can't be modified, it appears with a lock icon next to its name. Unlocking a card is an advanced activity and should be done with care, because automatic formula generation will no longer occur for the card, and you can't relock a card.

Within the form's container, the **ThisItem** record is available and contains all the fields of the record. For example, the card's **Default** property is often set to **ThisItem**.*FieldName*.

You can use the **Parent** reference to configure a control to reference the properties of a card. For example, a control should use **Parent.Default** to read the initial state of the field from the data source. By using **Parent** instead of directly accessing the information that you want, the card is better encapsulated, and you can change it to a different field without breaking internal formulas.

See Understand data cards for examples of how to customize, unlock, and create cards.

Key properties

DataField – The name of the field within a record that this card displays and edits.

- Specify the name as a single static string that's enclosed in double quotation marks (for example, **"Name"**), not a formula.
- Unbind a card by setting its **DataField** property *blank*. The **Valid** and **Update** properties are ignored for unbound cards.

Default – The initial value of a control before it is changed by the user.

- For each control in a card, set this property to **Parent.Default** to refer to the default value of the field according to the data source. For example, set a slider's Default property to **Parent.Default** to ensure that the user starts with a generic value for that slider.

DisplayMode – Values can be **Edit, View,** or **Disabled**. Configures whether the control inside the card allows user input (**Edit**), only displays data (**View**) or is disabled (**Disabled**).

- Allows a single card to be used in both edit and view forms, by configuring this property, which is tied to the Form's behavior by default.
- In **View** mode, child controls such as Text input, Drop down, Date Picker will only display the text value and will not render any interactive elements or decorations.

DisplayName – The user friendly name for a field in a data source.

- The DataSourceInfo function provides this metadata from the data source.
- Controls within the card should use **Parent.DisplayName** to refer to the name of the field.

Error – The user friendly error message to display for this field when validation fails.

- This property is set when **SubmitForm** is called.
- The message describes validation problems based on the data source's metadata and checking the card's **Required** property.

Required – Whether a card, editing the field of a data source, must contain a value.

- The **DataSourceInfo** function provides the required metadata from the data source.
- Controls within the card should use **Parent.Required** to determine whether that card's field is required.

Update – The value to write back to the data source for a field.

- Use this property's formula to pull the values from the edit controls of the card in order to write back to the data source. For example, set a card's **Update** property to **Slider.Value** to update the data source with a value from the slider in that card.

Width – The distance between a control's left and right edges.

WidthFit – Whether a control automatically grows horizontally to fill any empty space in a container control such as an **Edit form** control. If multiple cards have this property set to **true**, the space is divided between them. For more information, see Understand data form layout.

Additional properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

Valid – Whether a **Card** or **Edit form** control contains valid entries, ready to be submitted to the data source.

Visible – Whether a control appears or is hidden.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container). For a **Card** control in a container that has multiple columns, this property determines the column in which the card appears.

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container). For a **Card** control in a container that has multiple rows, this property determines the row in which the card appears.

Examples

See Understand data cards and Understand data form layout for examples.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **Fill** and any child controls. For example, if a card contains a **Label** and the label has transparent fill, then the card's **Fill** effectively becomes the background color for the label. Thus, there should be adequate contrast between the card's **Fill** and the label's **Color**.

Screen reader support

- **DisplayName** must be present.

Check box control in Power Apps

Article • 12/16/2022

A control that the user can select or clear to set its value to **true** or **false**.

Description

The user can specify a Boolean value by using this familiar control, which has been used in GUIs for decades.

Key properties

Default – The initial value of a control before it is changed by the user.

Text – Text that appears on a control or that the user types into a control.

Value – The value of an input control.

Additional properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

CheckboxBackgroundFill – The background color of the box that surrounds the checkmark in a checkbox control.

CheckboxBorderColor – The color of the border that surrounds the checkmark in a checkbox control.

CheckboxSize – The width and height of the box that surrounds the checkmark in a checkbox control.

CheckmarkFill – The color of the checkmark in a checkbox control.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

OnCheck – Actions to perform when the value of a checkbox or a toggle changes to **true**.

OnSelect – Actions to perform when the user taps or clicks a control.

OnUncheck – Actions to perform when the value of a checkbox or a toggle changes to **false**.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

Reset – Whether a control reverts to its default value.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

VerticalAlign – The location of text on a control in relation to the vertical center of that control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

If(*Condition*, *Result*)

Example

1. Add a **Check box** control, name it **chkReserve**, and set its **Text** property to show **Reserve now**.

Don't know how to add, name, and configure a control?

2. Add a **Date picker** control, and set its **Visible** property to this formula:
   If(chkReserve.Value = true, true)

   Want more information about the **If** function or other functions?

3. Press F5, click or tap **chkReserve** to set its **Value** property to **true**, and then click or tap **chkReserve** again to set its **Value** property to **false**.

   The **Date Picker** control appears when the **Value** property of **chkReserve** is **true** but not when it's **false**.

4. To return to the default workspace, press Esc.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **CheckmarkFill** and **CheckboxBackgroundFill**
- **CheckboxBackgroundFill** and **Fill**
- **CheckboxBackgroundFill** and **PressedFill**
- **CheckboxBackgroundFill** and **HoverFill**

This is in addition to the standard color contrast requirements.

Screen reader support

- **Text** must be present.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.

Column chart and Line chart controls in Power Apps

Article • 12/16/2022

Controls that show data as graphs with x- and y-axes.

Description

Column chart and **Line chart** are grouped controls. Each group contains three controls: a Label for the title, the chart graphic, and a **Legend**.

Chart key properties

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

NumberOfSeries – How many columns of data are reflected in a column or line chart.

Additional chart properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Font – The name of the family of fonts in which text appears.

GridStyle – Whether a column or line chart shows its x-axis, its y-axis, both, or neither.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

ItemColorSet – The color of each data point in a chart.

ItemsGap – The distance between columns in a column chart.

- The **ItemsGap** property is available for the **Column chart** control but not the **Line chart** control.

Markers – Whether a column or line chart shows the value of each data point.

MarkerSuffix – Text that appears after each value in a column chart for which the **Markers** property is set to **true**.

- The **MarkerSuffix** property is available for the **Column chart** control but not the **Line chart** control.

MinimumBarWidth – The narrowest possible width of columns in a column chart.

- The **MinimumBarWidth** property is available for the **Column chart** control but not the **Line chart** control.

OnSelect – Actions to perform when the user taps or clicks a control.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

SeriesAxisMax – The maximum value of the y-axis for a column or line chart.

- The **SeriesAxisMax** property is available for the **Column chart** control but not the **Line chart** control.

SeriesAxisMin – A number that determines the minimum value of the y-axis for a column chart.

- The **SeriesAxisMin** property is available for the **Column chart** control but not the **Line chart** control.

Size – The font size of the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

XLabelAngle – The angle of the labels below the x-axis of a column or line chart.

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

YAxisMax – The maximum value of the y-axis for a line chart.

- The **YAxisMax** property is available for the **Line chart** control but not the **Column chart** control.

YAxisMin – The minimum value of the y-axis for a line chart.

- The **YAxisMin** property is available for the **Line chart** control but not the **Column chart** control.

YLabelAngle – The angle of the labels next to the y-axis of a line or column chart.

Related functions

Max(*DataSource, ColumnName*)

Example

1. Add a **Button** control, and set its **OnSelect** property to this formula:
   Collect(Revenue, {Year:"2013", Europa:24000, Ganymede:22300, Callisto:21200}, {Year:"2014", Europa:26500, Ganymede:25700, Callisto:24700},{Year:"2014", Europa:27900, Ganymede:28300, Callisto:25600})

   Don't know how to add and configure a control?

   Want more information about the **Collect** function or other functions?

2. Press F5, click or tap the **Button** control, and then press Esc to return to the default workspace.

3. Add a **Column chart** control or a **Line chart** control, set its **Items** property to **Revenue**, and set its **NumberOfSeries** property to **3**.

   The control shows revenue data for each product over three years.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- each item in **ItemColorSet**
- every item in **ItemColorSet** and the background color
- **Color** and the background color

Screen reader support

- There must be a **Label** immediately before the chart graphic to serve as the title.
- Consider adding a summary of the chart graphic. For example, "The line chart shows a steady increase in sales between March and August this year."

  > ⓘ **Note**
  >
  > Chart graphics and **Legend** are hidden from screen reader users. As an alternative, a tabular form of the data is presented to them. They can also cycle through buttons that select data in the chart.

Low vision support

- There must be a **Legend** if more than one series is shown.
- Consider setting **GridStyle** to GridStyle.All, which shows both axes. This helps all users accurately determine the scale of the data.
- For **Column chart**, consider setting **Markers** to **true**. This helps low-vision users determine the value of a column.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.

Note

When keyboard users navigate to the chart, they can cycle through buttons that select data in the chart.

Column control in Power Apps

Article • 12/16/2022

Provides the display experience for a single field in a Data table control.

Description

The Data table control shows a dataset in a tabular format, and each column in that tabular format is represented by a **Column** control. The **Column** control provides properties that the app maker can use to customize the appearance and behavior of the column.

Capabilities

Now available

- Change the width of a **Column** control.
- Change the text for a **Column** control.
- Navigate by clicking or tapping the value in a **Column** control.

Not yet available

- Customize the styling of a **Column** control.

Known issues

- The **Visible** property doesn't work yet.

Properties

- **DisplayName** – The text that appears in the header for the column.

  > ⓘ Note
  >
  > This property will soon be renamed **HeaderText**.

- **IsHyperlink** – A value that indicates whether the data in the column should be underlined to indicate that it's a hyperlink.

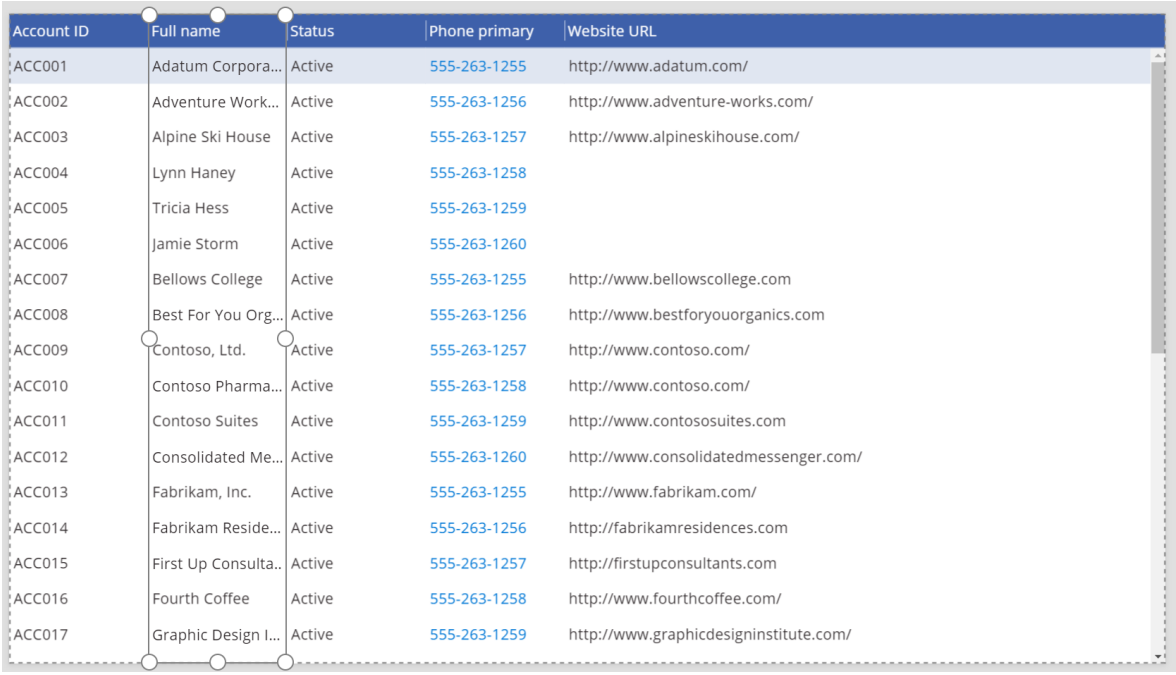
- **Width** – The distance between the **Column** control's left and right edges.

Examples

Resize a column

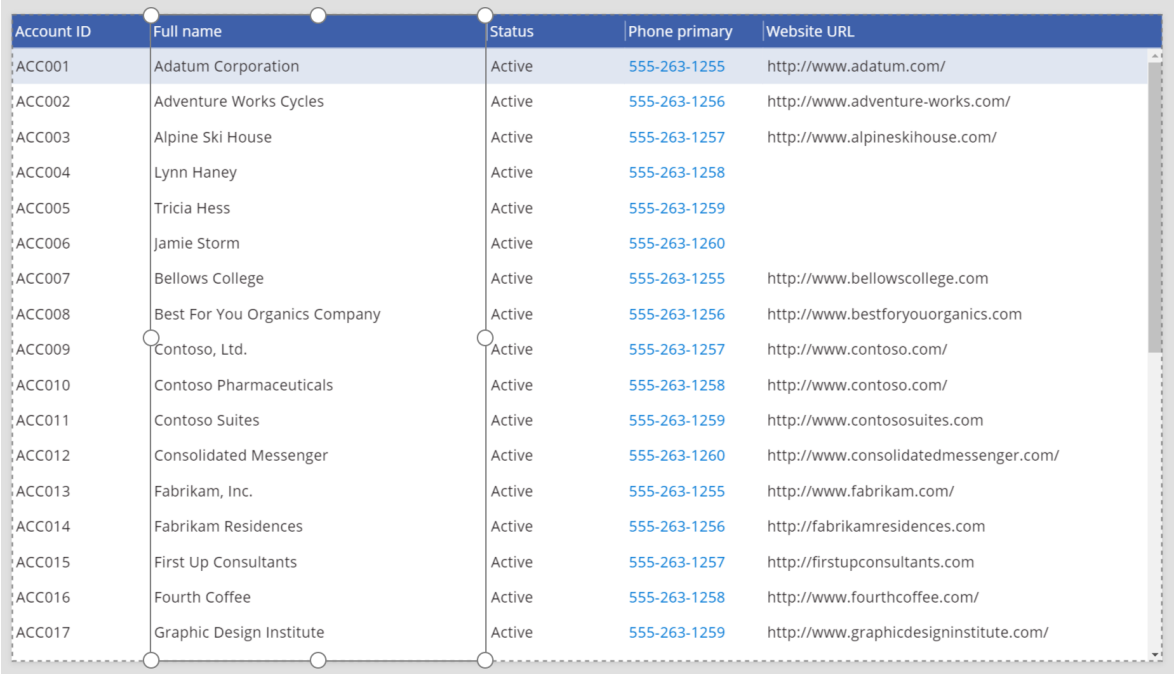
1. Create a blank tablet app.
2. On the **Insert** tab, click or tap **Data table**, and then resize the **Data table** control so that it covers the whole screen.
3. In the right pane, click or tap the down arrow to the right of **No data source selected**, and then click or tap **Add a data source**.
4. In the list of connections, click or tap the connection for your Microsoft Dataverse database.
5. In the list of tables, click or tap **Account**, and then click or tap **Connect**.

   The **Data table** control is initialized and shows a set of default fields.
6. Click or tap the **Full name** column.



Account ID	Full name	Status	Phone primary	Website URL
ACC001	Adatum Corpora...	Active	555-263-1255	http://www.adatum.com/
ACC002	Adventure Work...	Active	555-263-1256	http://www.adventure-works.com/
ACC003	Alpine Ski House	Active	555-263-1257	http://www.alpineskihouse.com/
ACC004	Lynn Haney	Active	555-263-1258	
ACC005	Tricia Hess	Active	555-263-1259	
ACC006	Jamie Storm	Active	555-263-1260	
ACC007	Bellows College	Active	555-263-1255	http://www.bellowscollege.com
ACC008	Best For You Org...	Active	555-263-1256	http://www.bestforyouorganics.com
ACC009	Contoso, Ltd.	Active	555-263-1257	http://www.contoso.com/
ACC010	Contoso Pharma...	Active	555-263-1258	http://www.contoso.com/
ACC011	Contoso Suites	Active	555-263-1259	http://www.contososuites.com
ACC012	Consolidated Me...	Active	555-263-1260	http://www.consolidatedmessenger.com/
ACC013	Fabrikam, Inc.	Active	555-263-1255	http://www.fabrikam.com/
ACC014	Fabrikam Reside...	Active	555-263-1256	http://fabrikamresidences.com
ACC015	First Up Consulta..	Active	555-263-1257	http://firstupconsultants.com
ACC016	Fourth Coffee	Active	555-263-1258	http://www.fourthcoffee.com/
ACC017	Graphic Design I...	Active	555-263-1259	http://www.graphicdesigninstitute.com/

7. Drag the adorner on the right side to resize the field.



Account ID	Full name	Status	Phone primary	Website URL
ACC001	Adatum Corporation	Active	555-263-1255	http://www.adatum.com/
ACC002	Adventure Works Cycles	Active	555-263-1256	http://www.adventure-works.com/
ACC003	Alpine Ski House	Active	555-263-1257	http://www.alpineskihouse.com/
ACC004	Lynn Haney	Active	555-263-1258	
ACC005	Tricia Hess	Active	555-263-1259	
ACC006	Jamie Storm	Active	555-263-1260	
ACC007	Bellows College	Active	555-263-1255	http://www.bellowscollege.com
ACC008	Best For You Organics Company	Active	555-263-1256	http://www.bestforyouorganics.com
ACC009	Contoso, Ltd.	Active	555-263-1257	http://www.contoso.com/
ACC010	Contoso Pharmaceuticals	Active	555-263-1258	http://www.contoso.com/
ACC011	Contoso Suites	Active	555-263-1259	http://www.contososuites.com
ACC012	Consolidated Messenger	Active	555-263-1260	http://www.consolidatedmessenger.com/
ACC013	Fabrikam, Inc.	Active	555-263-1255	http://www.fabrikam.com/
ACC014	Fabrikam Residences	Active	555-263-1256	http://fabrikamresidences.com
ACC015	First Up Consultants	Active	555-263-1257	http://firstupconsultants.com
ACC016	Fourth Coffee	Active	555-263-1258	http://www.fourthcoffee.com/
ACC017	Graphic Design Institute	Active	555-263-1259	http://www.graphicdesigninstitute.com/

Accessibility guidelines

Screen reader support

- **DisplayName** must be present.

Combo box control in Power Apps

Article • 05/23/2025

A control that allows users to make selections from provided choices. Supports search and multiple selections.

Description

A **Combo box** control allows you to search for items you will select. The search is performed server-side on the SearchField property so performance is not affected by large data sources.

Single or multi-select mode is configured via the SelectMultiple property.

When searching for items to select, for each item you can choose to show a single data value, two values, or a picture and two values (Person) by modifying the Layout settings in the Data pane.

When you view the control on small screens, the items list flyout becomes a full screen control for better usability.

> ⓘ **Note**
>
> If you want to search for items with *numbers*, convert numbers to text with Text() function. For example, *Text(12345)*.

Limitations

The combo box control has these limitations:

- When you use a Combo box control inside a gallery, its selections aren't maintained when the user scrolls the gallery. This isn't an issue if you use a Combo box control inside a gallery that doesn't scroll. A workaround isn't currently available.

People picker

To use **Combo box** as a people picker, choose the **Person** template from the Layout settings in the Data pane and configure the related data properties to be shown for the person below.

Key properties

Items – The source of data from which selections can be made.

DefaultSelectedItems – The initial selected item(s) before the user interacts with the control.

> ⓘ Note
>
> **Default** property is deprecated, use **DefaultSelectedItems** instead.

SelectedItems – List of selected items resulting from user interaction.

Selected – The last selected item resulting from user interaction.

SelectMultiple – Whether the user can select a single item or multiple items.

IsSearchable – Whether the user can search for items before selecting.

> ⓘ Note
>
> **IsSearchable** can be enabled only on a data source that contains at least one text field. The **Items** expression must be delegable for queries to be delegated with searching. That is, **If** statements within the **Items** expression are not supported.

SearchFields - The data fields of the data source searched when user is entering text.

> ⓘ Note
>
> To search on multiple fields, set **SearchFields** property in this format: ["MyFirstColumn", "MySecondColumn"]. Only text fields are supported.

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayFields – List of fields shown for each item returned by the search. Easiest to configure via the Data pane in the Properties option tab.

> ⓘ Note
>
> To update multiple display fields in given template, set **DisplayFields** property in this format: `["MyFirstColumn", "MySecondColumn"]`.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

InputTextPlaceholder – Instructional text shown to end users when no items are selected.

OnChange – Actions to perform when the user changes a selection.

OnNavigate – Actions to perform when the user selects an item.

OnSelect – Actions to perform when the user taps or selects a control.

TabIndex – Keyboard navigation order in relation to other controls.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

> ⓘ Note
>
> Flyout border properties are controlled by **Border** properties.

Examples

Basic Combo box

The steps in this example apply to any data source that provides tables, but to follow these steps exactly, use an environment with Microsoft Dataverse database having sample data.

1. Select **Insert** > **Input** > **Combo box**, and name it "Combobox1".
2. On the **Properties** pane on the right-side of the screen, open the **Select a data source** list (next to **Items**), and then add or select a data source such as the Accounts table.
3. On the same pane, select **Edit** (next to **Fields**).
4. In the **Data** pane, open the **Primary text** list, and then select the **Primary Name** column that will show in the **Combo box** control.
5. While holding down the **Alt** key, select the down arrow to open the **Combo box** control.

   The control shows the data from the Primary Name that you specified in the data source that you specified.

(Optional) Show the first record by default

1. Set the **DefaultSelectedItems** property to this expression, replacing *DataSource* with the name of your data source:

```
Power Fx

First(DataSource)
```

(Optional) Display selected Account Name value in a label

1. Select **Insert** > **Text label**, and then select **Label**.
2. Set the **Text** property to this expression, replacing *Text* with the following formula:

```
Power Fx

If(CountRows(ComboBox1.SelectedItems)>0,
Concat(ComboBox1.SelectedItems,'Account Name',", "), "NO SELECTED ITEM")
```

> **Note**
>
> The If statement will check to see how may selected items exist and display them in a comma delimited label or a "NO SELECTED ITEM" message when empty.

Simulate simple drop-down behavior

By setting **IsSearchable** to false and **SelectMultiple** to false, you can achieve the same functionality of a drop-down.

1. Select **Insert** > **Input**, and then select **Combo box**.
2. On the **Properties** pane on the right-side of the screen, open the **Select a data source** list (next to **Items**), and then add or select a data source.
3. On the same pane, select **Edit** (next to **Fields**).
4. In the **Data** pane, open the **Primary text** list, and then select the column that you want to show in the **Combo box** control.
5. Set the **IsSearchable** property to false.
6. Set the **SelectMultiple** property to false.
7. While holding down the **Alt** key, select the down arrow to open the **Combo box** control.

   The control shows the data from the column that you specified in the data source that you specified.

   > ⓘ Note
   >
   > "Find items" text has disappeared, and that you'll only be able to select a single item.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **ChevronFill** and **ChevronBackground**
- **ChevronHoverFill** and **ChevronHoverBackground**
- **SelectionColor** and **SelectionFill**
- **SelectionFill** and **Fill**
- **SelectionTagColor** and **SelectionTagFill**

This is in addition to the standard color contrast requirements.

Screen reader support

- AccessibleLabel must be present.

> **Note**
>
> On touch screens, screen reader users can navigate the contents of the combo box sequentially. The combo box acts as a button that shows or hides its contents when selected.

Keyboard support

- TabIndex must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use FocusedBorderColor and FocusedBorderThickness to achieve this.

> **Note**
>
> The tab key navigates to or away from the combo box. Arrow keys navigate the contents of the combo box. The escape key closes the drop-down when opened.

Container control in Power Apps

Article • 12/16/2022

Groups logically-related controls to create hierarchies.

Description

The container can hold a set of controls and has its own properties.

You can start with inserting a blank container. And then, customize it by adding controls to it, resizing it, moving it, hiding it, and making other changes. You can also start with a number of controls, select them and add using tree view.

Limitations

- Containers don't work within forms.
- The following controls are not supported within a container:
  - Data table
  - PDF viewer
  - Web barcode scanner

Properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

Width – The distance between a control's left and right edges.

Visible – Whether a control appears or is hidden.

X – The distance between the left edge of a control and the left edge of its parent container (screen, if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen, if no parent container).

Frequently asked questions

What is the difference between a container and a group?

When editing an app, you can select controls and group them using Ctrl + G or the context menu. You can modify common properties of controls within the group. Groups are an aid for app making. They don't have properties of their own and don't affect the layout of the app.

In contrast, containers are actual controls with their own properties like **Width** and **BorderColor**. Containers affect app layout and help screen reader users understand the structure of the app.

While you can add any controls in a group, you should only add logically-related controls in a container. For example, controls in a tile should be placed in containers.

Data table control in Power Apps

Article • 04/03/2024

Shows a set of data in a tabular format.

Description

The **Data table** control shows a dataset in a format that includes column headers for each field that the control shows. As an app maker, you have full control over which fields appear and in what order. Like the **Gallery** control, the **Data table** control maintains a **Selected** property that points to the selected row. Therefore, you can link the **Data table** control to other controls.

Limitations

The data table control has these limitations:

- If you copy and paste a **Data Table** control for which the **Items** property is set to a formula that contains a **Filter** function, the formula for the **Items** property on the new **Data Table** control ends up with field names that contain a **_1** suffix. This makes the field names invalid and results in no data showing up in the data table. To work around this issue, before you copy the control, confirm that the **Filter** function doesn't reference any field in the data source that has the same name as a column in the **Data Table** control. If it does, rename the column in the **Data Table** control. Alternatively, remove the **_1** suffix from the invalid field names so that they match the names in the table.

Capabilities

Power Apps introduced the **Data table** control on May 5, 2017. This section provides information about capabilities that are supported and capabilities that aren't supported.

Now available

- Data in a **Data table** control is read-only.
- A single row is always selected in a **Data table** control.
- Link a **Data table** control to a connected or local data source.
- Adjust column widths in a **Data table** control while you run the app, though your changes aren't saved.

- A set of default fields appear in a **Data table** control when you link it to a connector that has implemented this capability, such as the Microsoft Dataverse. You can then show or hide these fields and others as necessary.
- Customize column width and heading text.
- Show hyperlinks in a **Data table** control.
- Copy and paste a **Data table** control.

Not yet available

- Customize the styling of individual columns.
- Add a **Data table** control in a form control.
- Change the height of all rows.
- Show images in a **Data table** control.
- Show fields from related tables.
- Use built-in functionality to filter and sort data by column heading.
- Add a **Data table** control in a **Gallery** control.
- Edit data in the **Data table** control.
- Select multiple rows.

Known issues

- No data appears if you use the **FirstN** function in the **Items** property.

Key properties

- Items – The source of data that appears in the **Data table** control.
- **Selected** – The selected row in the **Data table** control.

Other properties

- **AutoWidth** - The property of a column inside a data table to define if the column width should be automatic, or manual.
- BorderColor – The color of the **Data table** control's border.
- BorderStyle – The style of the **Data table** control's border. The options are **Solid**, **Dashed**, **Dotted**, and **None**.
- BorderThickness – The thickness of the **Data table** control's border.
- Color – The default text color for all data rows.
- **FieldDisplayName** - The property of a column inside a data table that defines display name for the selected column.
- Fill – The default background color for all data rows.

- Font - The default font for all data rows.
- FontWeight – The default font weight for all data rows.
- **HeadingColor** – The text color for the column headings.
- **HeadingFill** – The background color of the column headings.
- **HeadingFont** – The font of the column headings.
- **HeadingFontWeight** – The font weight of the column headings.
- **HeadingSize** – The font size of the column headings.
- Height – The distance between the **Data table** control's top and bottom edges.
- HoverColor – The text color for the row that the mouse pointer is pointing at.
- HoverFill – The background color of the row that the mouse pointer is pointing at.
- **IsHyperlink** - The property of a column inside a data table that defines whether the selected column should be in hyperlink format or not.
- **NoDataText** – The message that the user receives when there are no records to show in the **Data table** control.
- **Order** - The property of a column inside a data table that determines order of the column respective to other columns. The order starts from left to right with first column order value of 1.
- **SelectedColor** – The color of the text in the selected row.
- **SelectedFill** – The background color of the selected row.
- Size – The default font size for all data rows.
- **Text** - The property of a column inside a data table that defines text value of the data to be displayed inside the selected column.
- Visible – A value that determines whether the **Data table** control appears or is hidden.
- Width – The distance between the **Data table** control's left and right edges.
- X – The distance between the left edge of the **Data table** control and the left edge of its parent container (or the left edge of the screen if there is no parent container).
- Y – The distance between the top edge of the **Data table** control and the top edge of its parent container (or the top edge of the screen if there is no parent container).

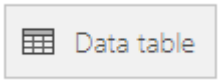
Related functions

- Filter(DataSource, Formula)(*DataSource*, *Formula*)
- Search(DataSource, SearchString, Column)(*DataSource*, *SearchString*, *Column*)

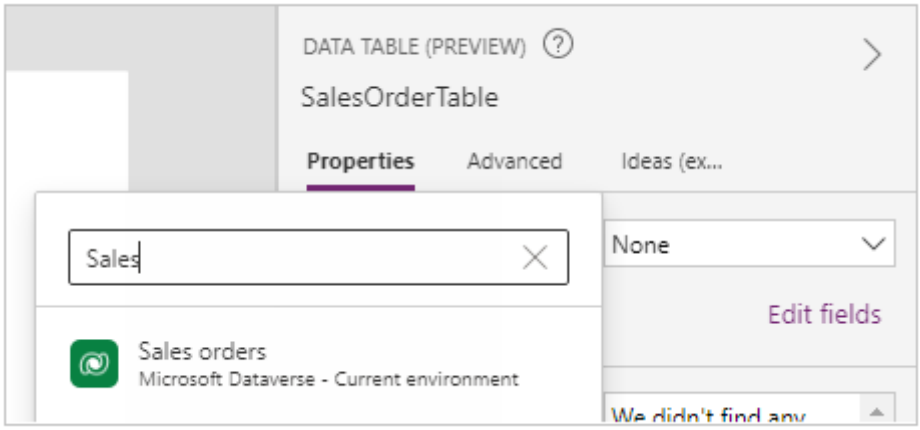
Examples

Basic usage

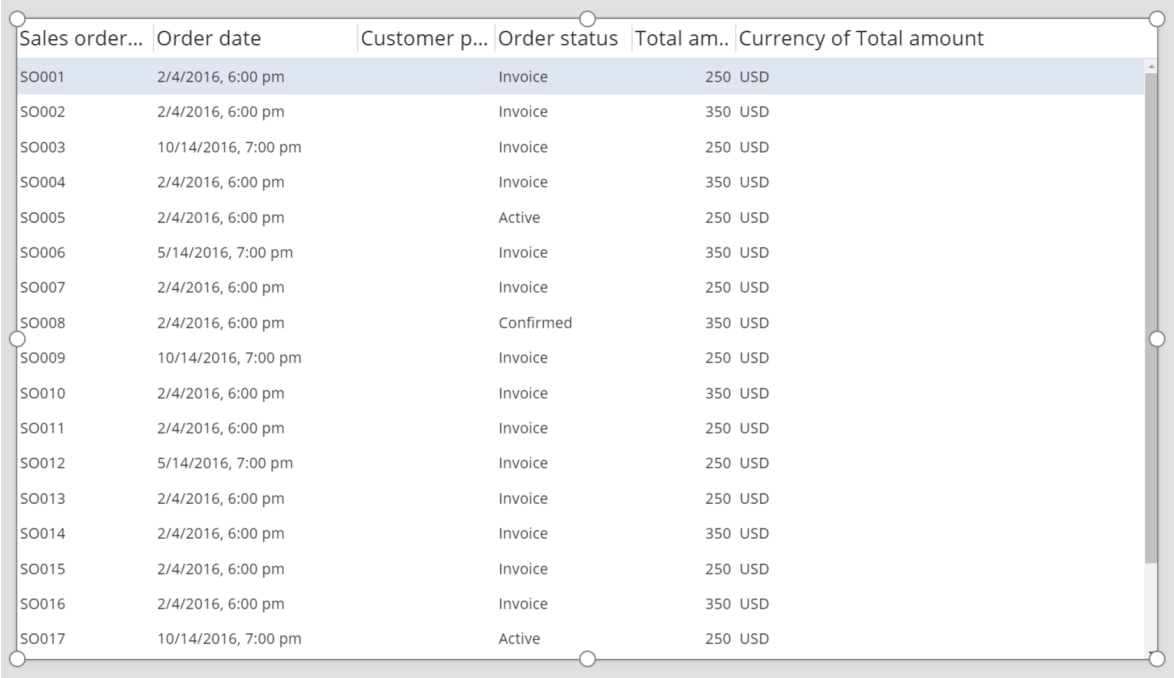
1. Create a blank tablet app.
2. On the **Insert** tab, click or tap **Data table**.

   A **Data table** control is added to the screen.
3. Rename the **Data table** control **SalesOrderTable**, and resize it so that it covers the whole screen.
4. In the right pane, select **Data sources**, and select **Sales orders**. If the table isn't listed, search for the table, and connect to Microsoft Dataverse.

   The **Data table** control is now attached to the **Sales order** data source. Several initial fields appear in the **Data table** control, because we're using a connector that supports that capability.

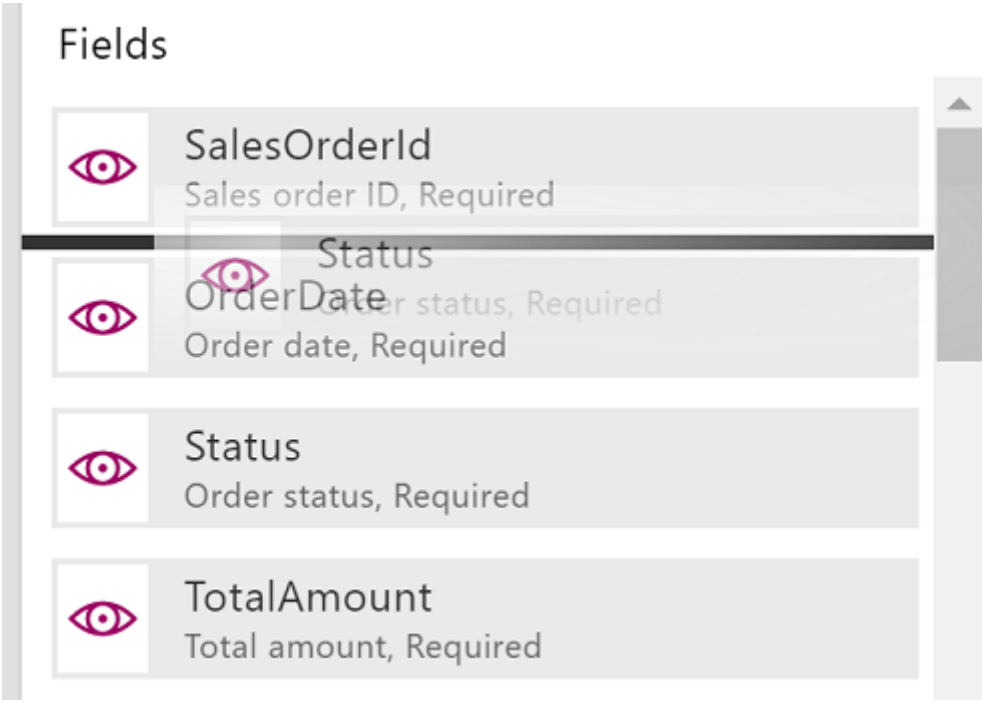


Sales order...	Order date	Customer p...	Order status	Total am..	Currency of Total amount
SO001	2/4/2016, 6:00 pm		Invoice	250	USD
SO002	2/4/2016, 6:00 pm		Invoice	350	USD
SO003	10/14/2016, 7:00 pm		Invoice	250	USD
SO004	2/4/2016, 6:00 pm		Invoice	350	USD
SO005	2/4/2016, 6:00 pm		Active	250	USD
SO006	5/14/2016, 7:00 pm		Invoice	350	USD
SO007	2/4/2016, 6:00 pm		Invoice	250	USD
SO008	2/4/2016, 6:00 pm		Confirmed	350	USD
SO009	10/14/2016, 7:00 pm		Invoice	250	USD
SO010	2/4/2016, 6:00 pm		Invoice	350	USD
SO011	2/4/2016, 6:00 pm		Invoice	250	USD
SO012	5/14/2016, 7:00 pm		Invoice	250	USD
SO013	2/4/2016, 6:00 pm		Invoice	250	USD
SO014	2/4/2016, 6:00 pm		Invoice	350	USD
SO015	2/4/2016, 6:00 pm		Invoice	250	USD
SO016	2/4/2016, 6:00 pm		Invoice	350	USD
SO017	10/14/2016, 7:00 pm		Active	250	USD

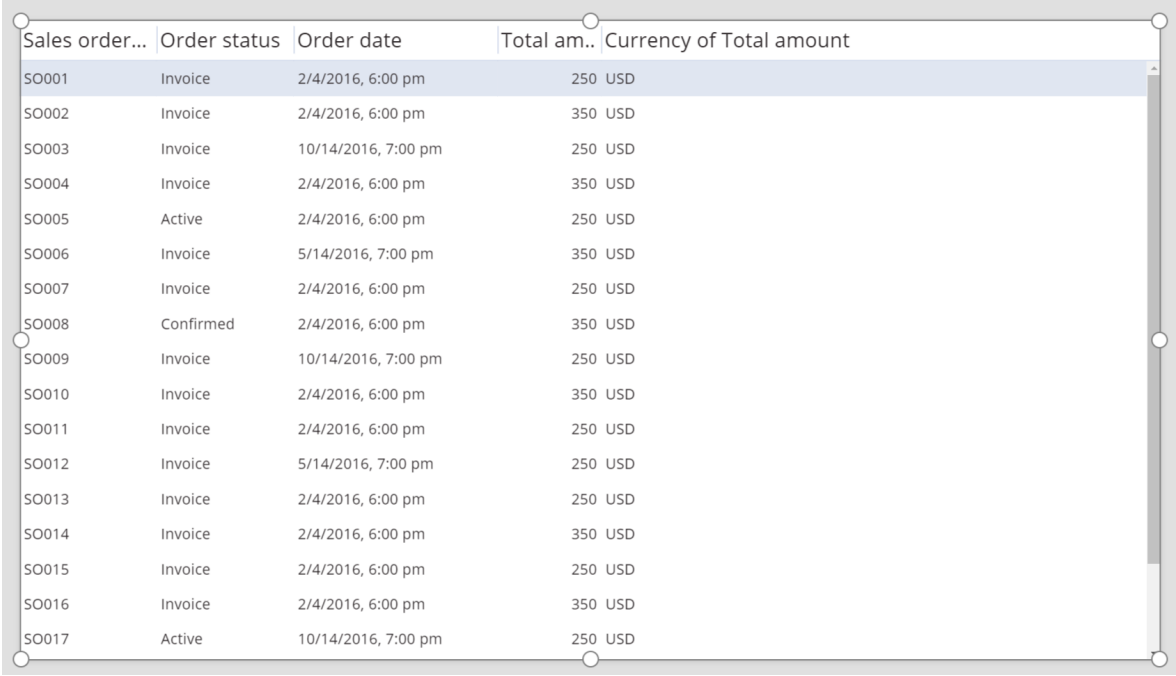
5. In the right pane, select one or more check boxes to show or hide individual fields.

   For example, select the check box next to **CustomerPurchaseOrderReference** to hide this field.

6. In the right pane, reorder the fields by dragging them up or down.



   The **SalesOrderTable** control shows the fields in the order that you specified.



Sales order...	Order status	Order date	Total am..	Currency of Total amount
SO001	Invoice	2/4/2016, 6:00 pm	250	USD
SO002	Invoice	2/4/2016, 6:00 pm	350	USD
SO003	Invoice	10/14/2016, 7:00 pm	250	USD
SO004	Invoice	2/4/2016, 6:00 pm	350	USD
SO005	Active	2/4/2016, 6:00 pm	250	USD
SO006	Invoice	5/14/2016, 7:00 pm	350	USD
SO007	Invoice	2/4/2016, 6:00 pm	250	USD
SO008	Confirmed	2/4/2016, 6:00 pm	350	USD
SO009	Invoice	10/14/2016, 7:00 pm	250	USD
SO010	Invoice	2/4/2016, 6:00 pm	350	USD
SO011	Invoice	2/4/2016, 6:00 pm	250	USD
SO012	Invoice	5/14/2016, 7:00 pm	250	USD
SO013	Invoice	2/4/2016, 6:00 pm	250	USD
SO014	Invoice	2/4/2016, 6:00 pm	350	USD
SO015	Invoice	2/4/2016, 6:00 pm	250	USD
SO016	Invoice	2/4/2016, 6:00 pm	350	USD
SO017	Active	10/14/2016, 7:00 pm	250	USD

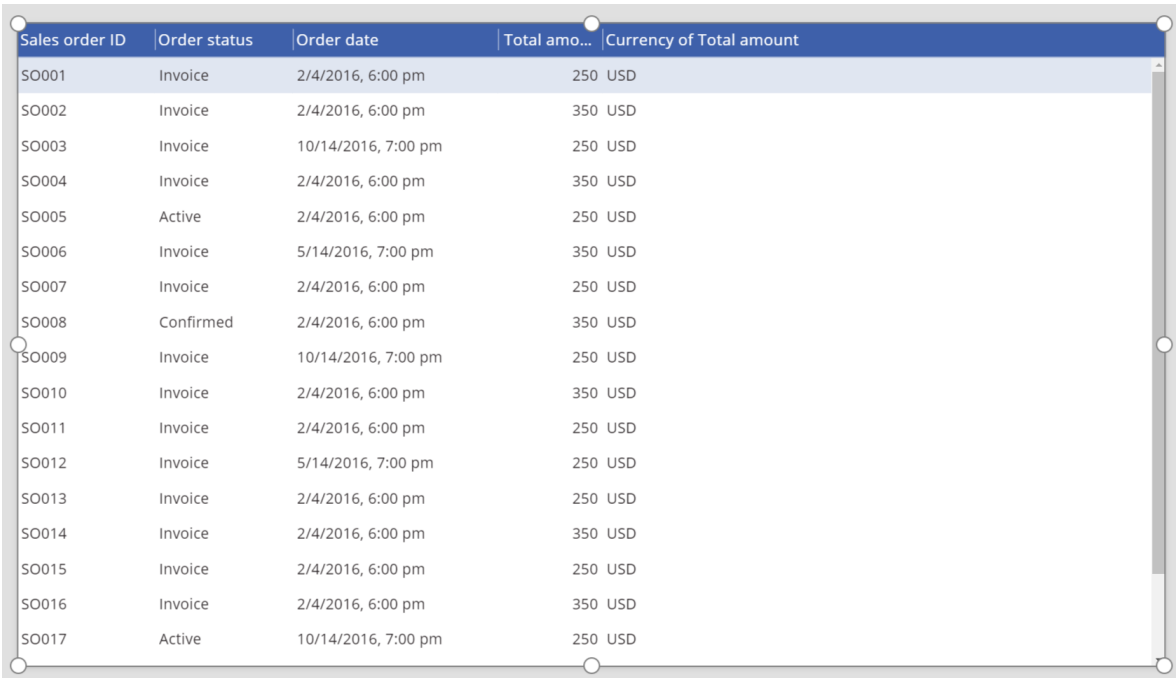
Restyle the header for the Data table control

1. While the **Data table** control is selected, in the right pane, click or tap the **Advanced** tab.

2. Click or tap the field for the **HeadingFill** property, and then change the value to **RGBA(62,96,170,1)**.

3. Click or tap the field for the **HeadingColor** property, and then change the value to **White**.

4. Click or tap the field for the **HeadingSize** property, and then change the value to **14**.

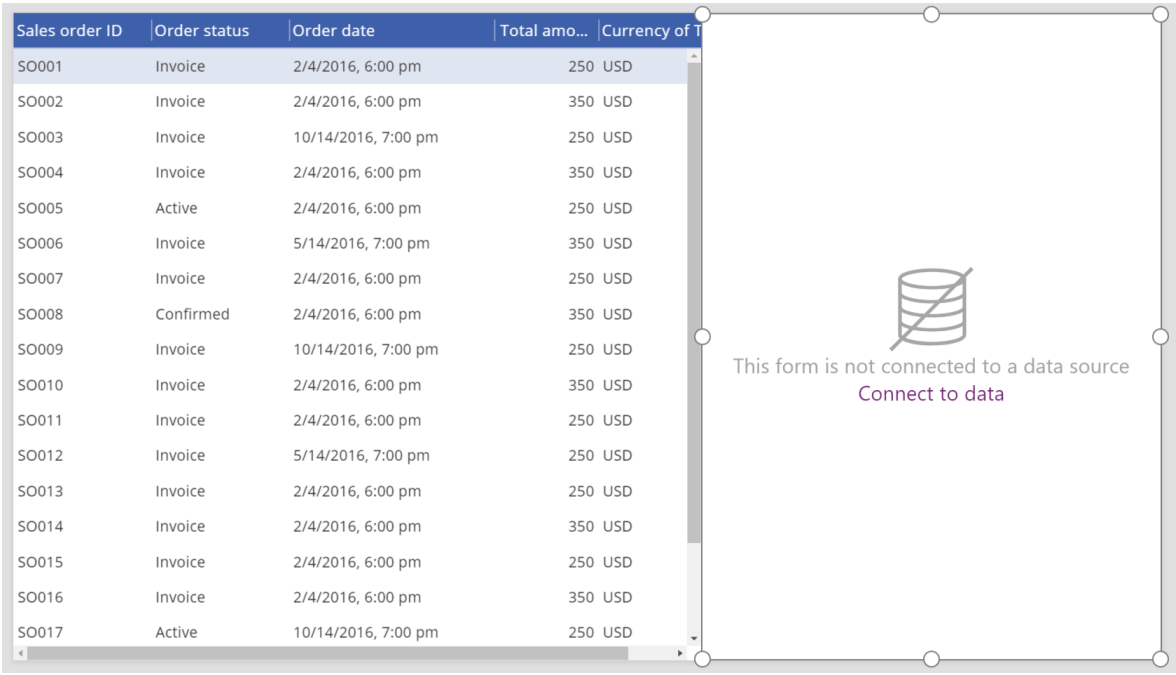


Sales order ID	Order status	Order date	Total amo...	Currency of Total amount
SO001	Invoice	2/4/2016, 6:00 pm	250	USD
SO002	Invoice	2/4/2016, 6:00 pm	350	USD
SO003	Invoice	10/14/2016, 7:00 pm	250	USD
SO004	Invoice	2/4/2016, 6:00 pm	350	USD
SO005	Active	2/4/2016, 6:00 pm	250	USD
SO006	Invoice	5/14/2016, 7:00 pm	350	USD
SO007	Invoice	2/4/2016, 6:00 pm	250	USD
SO008	Confirmed	2/4/2016, 6:00 pm	350	USD
SO009	Invoice	10/14/2016, 7:00 pm	250	USD
SO010	Invoice	2/4/2016, 6:00 pm	350	USD
SO011	Invoice	2/4/2016, 6:00 pm	250	USD
SO012	Invoice	5/14/2016, 7:00 pm	250	USD
SO013	Invoice	2/4/2016, 6:00 pm	250	USD
SO014	Invoice	2/4/2016, 6:00 pm	350	USD
SO015	Invoice	2/4/2016, 6:00 pm	250	USD
SO016	Invoice	2/4/2016, 6:00 pm	350	USD
SO017	Active	10/14/2016, 7:00 pm	250	USD

Connect a Data table control to another control

1. Add an **Edit form** control to the screen.

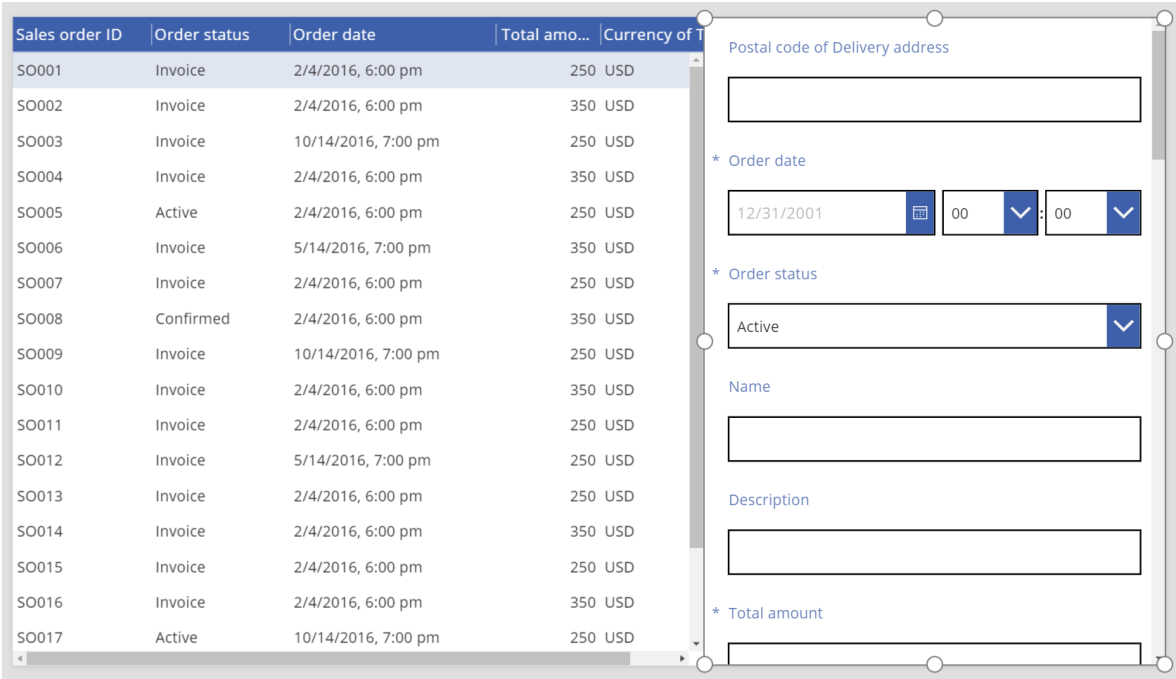
2. Resize the **Data table** and **Edit form** controls so that the **Data table** control appears in the left part of the screen and the **Edit form** control appears in the right part of the screen.



3. While **Form1** is selected, in the right pane, change the number of columns to **1**.

4. Connect **Form1** to the **Sales order** data source.

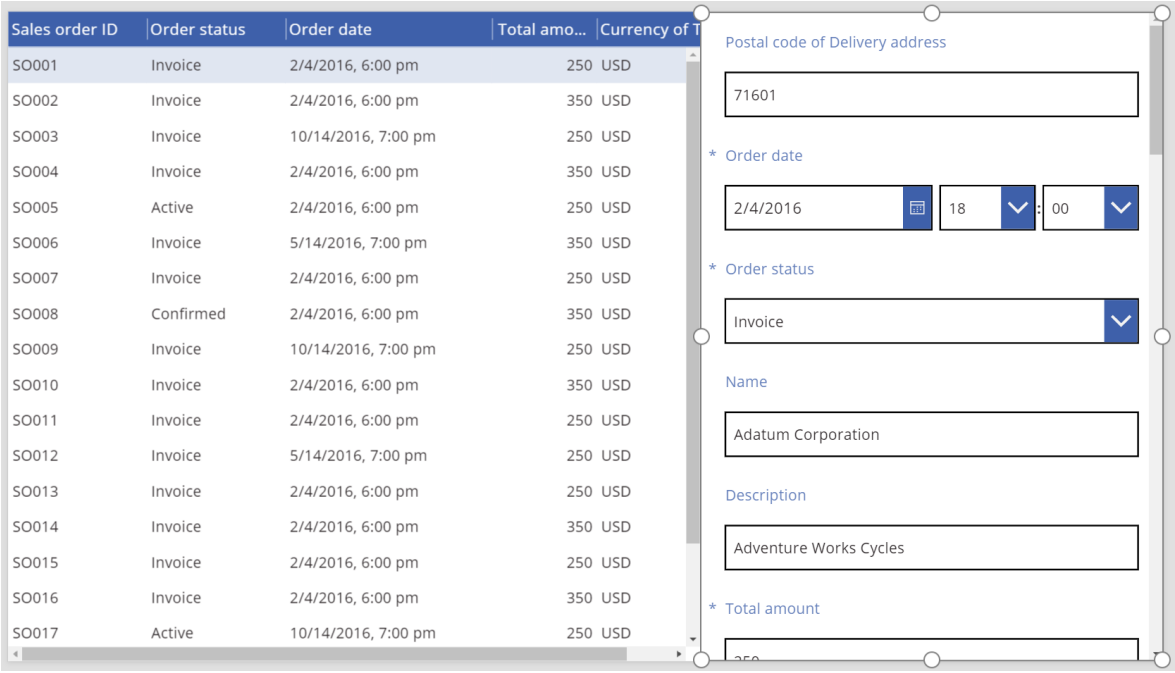
   Several initial fields appear in **Form1**.



5. In the right pane, click or tap the **Advanced** tab.

6. Set the **Item** property for **Form1** to **SalesOrderTable.Selected**.

   Form1 shows information from the row that's selected in the **Data table** control.



Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- Color and Fill
- **HeadingColor** and **HeadingFill**
- **SelectedColor** and **SelectedFill**
- HoverColor and HoverFill

This is in addition to the standard color contrast requirements.

Screen reader support

- **NoDataText** must be present.

Date Picker control in Power Apps

Article • 12/16/2022

A control that the user can select to specify a date.

Description

If you add a **Date Picker** control instead of a **Text input** control, you help ensure that the user specifies a date in the correct format.

Key properties

DefaultDate – The initial value of a date control unless the user changes it.

SelectedDate – The date currently selected in a date control. This date is represented in local time.

Format – The text format in which the control shows the date and the user specifies the date. You can set this property to **ShortDate** (default) or **LongDate** to format dates based on the **Language** property of this control. You can also set this property to an expression, such as **yyyy/mm/dd** if you want the same format regardless of language. For example:

- The control shows **12/31/2017** if the user clicks or taps the last day of 2017, the **Format** property is set to **ShortDate**, and the **Language** property is set to **en-us**.
- The control shows **dimanche 31 decembre 2017** if the user clicks or taps the last day of 2017, the **Format** property is set to **LongDate**, and the **Language** property is set to **fr-fr**.

Language – Determines the language that's used to format dates, including names of months. If this property isn't specified, the user's device setting determines the language. Supported values include "EN-us" and "FR".

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DateTimeZone – Whether to display the date in **UTC** or the user's **Local** time.

DisplayMode – Whether the control allows user input (Edit), only displays data (View), or is disabled (Disabled).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

EndYear – The latest year to which the user can set value of a date-picker control.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

IconFill – The foreground color of a the date picker icon.

IconBackground – The background color of a the date picker icon.

InputTextPlaceholder – Instructional text that appears if no dates are entered.

IsEditable – Whether the Date Picker text can be edited. If false, the date can only be changed by using the calendar.

Italic – Whether the text in a control is italic.

OnSelect – Actions to perform when the user taps or clicks a control.

OnChange – Actions to perform when the user changes the value of a control.

Difference between **OnChange** and **OnSelect**: OnSelect and OnChange trigger on the same user action if the user's *click* causes the change. In this case, OnSelect triggers **before** OnChange.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

Reset - Whether the Date Picker control should be reset to the **DefaultDate** value.

Size – The font size of the text that appears on a control.

StartOfWeek – The day of the week to display in the first day column of the date-picker control.

StartYear – The earliest year to which the user can set the value of a date-picker control.

TabIndex – Keyboard navigation order in relation to other controls.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Year(*DateTimeValue*)

Examples

Basic date picker

1. Add a **Date Picker** control, and name it "Deadline".

Don't know how to add, name, and configure a control?

2. Add a **Label** control, and set its **Text** property to the following formula:

```
Power Apps
DateDiff(Today(), Deadline.SelectedDate) & " days to go!"
```

Want more information about the **DateDiff** function or other functions?

3. Press **F5**, choose a date in **Deadline**, and then select **OK**.

The **Label** control shows the number of days between today and the date that you chose.

4. To return to the default workspace, press Esc.

Reset date picker to default date

1. Add a **Date Picker** control, and name it "DateTimeReset".

Don't know how to add, name, and configure a control?

2. Add a **Button** control, and set its **OnChange** property to the following formula:

```
Power Apps
Reset(DateTimeReset)
```

3. Press **F5**, choose a new date in **DateTimeReset**, and then select **OK**.

4. Press the button. The date will reset back to the **DefaultDate** (Today).

5. To return to the default workspace, press Esc.

Accessibility guidelines

Color contrast

- Standard color contrast requirements apply.

Screen reader support

- AccessibleLabel must be present.
- If **IsEditable** is true and **Format** is not what users expect, add a Label to explain accepted date formats.

Keyboard support

- TabIndex must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use FocusedBorderColor and FocusedBorderThickness to achieve this.

> **Tip**
>
> When the calendar is open, press **Page up** and **Page down** to navigate between months and **Shift+Page up** and **Shift+Page down** to navigate between years.

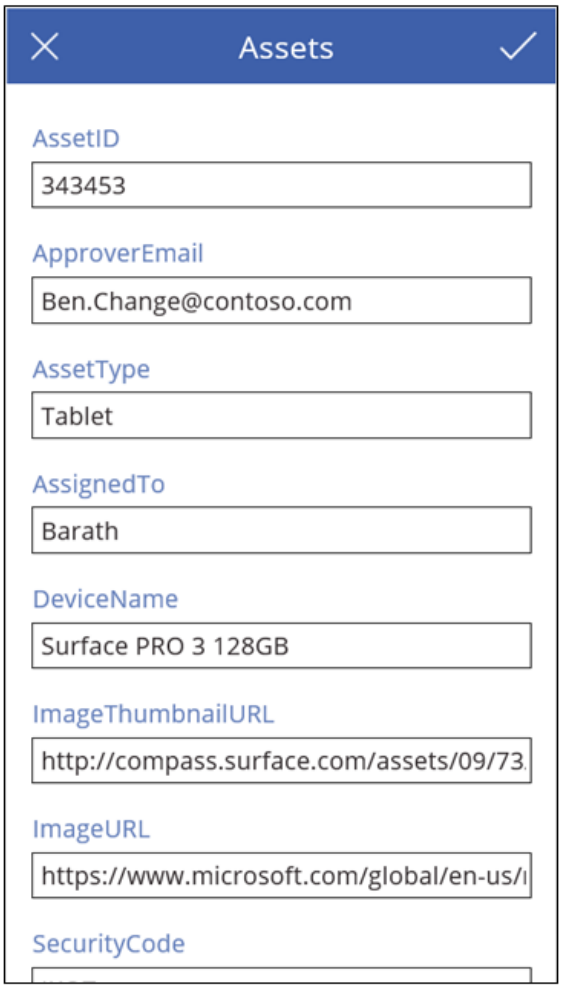
Edit form and Display form controls in Power Apps

Article • 03/26/2024

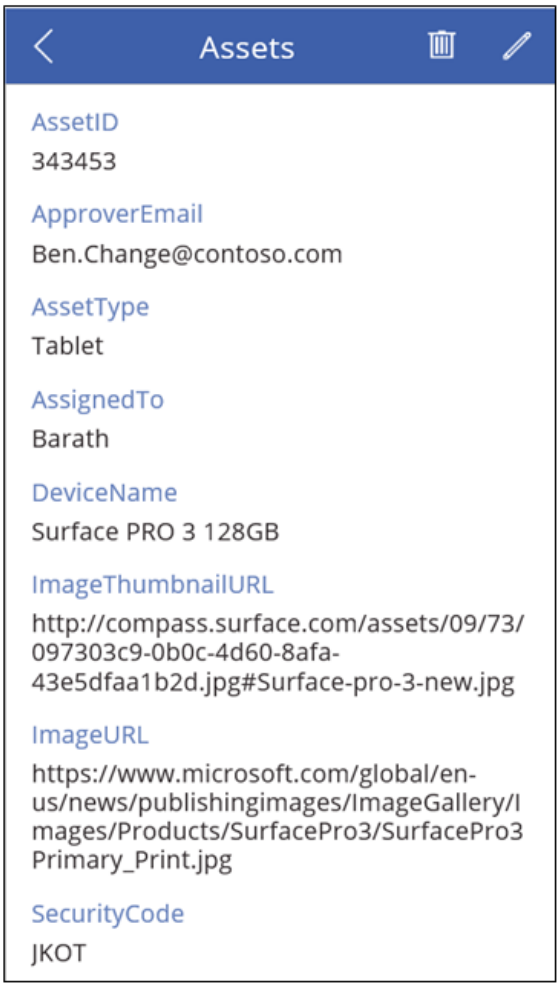
Display, edit, and create a record in a data source.

Description

If you add a **Display form** control, the user can display all fields of a record or only the fields that you specify. If you add an **Edit form** control, the user can edit those fields, create a record, and save those changes to a data source.



Edit form control
Read/Write



Display form control
Read only

If you add a **Gallery** control, you can configure it to show a table in a data source and then configure a form to show whichever record the user selects in the gallery. You can also add one or more **Button** controls that the user can select to save edits, cancel edits, and create a record. By using controls together, you can create a complete solution.

Limitations

The form control has these limitations:

- You can't copy and paste forms into data cards. This is to prevent certain combinations of controls from being created that risk the stability and performance of the app.

Record selection

For either type of form, you set its **DataSource** property to a table of records, and you set the form's **Item** property to show a specific record in that table. For example, you can set the **Item** property of a form to the **SelectedItem** property of a **Gallery** control. When the user selects a record in the gallery, the same record appears in the form, except that the form can show more fields. If the user returns to the gallery and selects a different record, the **SelectedItem** property of the gallery changes. This change updates the **Item** property of the form, which then shows the newly selected record.

You can also set a form's **Item** property by using a **Drop down** control, as Show, edit, or add a record describes, or a function such as **Lookup** or **First**. For example, you can set the **Item** property to either of these formulas to show the Fabrikam entry in the **Accounts** table in Microsoft Dataverse:

```
First(Accounts)
```

```
Lookup(Accounts, "Fabrikam" in name)
```

Each form control contains one or more **Card** controls. By setting the **DataField** property of a card, you specify which field that card shows and other details.

Create a record

When an **Edit form** control is in **Edit** mode, the user can update the record that's specified in the form's **Item** property. If inspected, the **Mode** property returns **Edit**.

When an **Edit form** control is in **New** mode, however, the **Item** property is ignored. The form doesn't show an existing record; instead, the values in each field match the default values of the data source with which you configured the form. The **NewForm** function causes a form to switch to this mode.

For example, you can set the **Text** property of a button to show **New** and its **OnSelect** property to a formula that includes the **NewForm** function. If the user selects that

button, the form switches to **New** mode so that the user can create a record starting with known values.

A form switches back to **Edit** mode if either the ResetForm function runs or the SubmitForm function runs successfully.

- You can set the Text property of a button to show **Cancel** and its OnSelect property to a formula that includes the ResetForm function. If the user selects that button, any changes in progress are discarded, and the values in the form, once again, match the default values of the data source.
- You can set the Text property of a button to show **Save changes** and its OnSelect property to a formula that includes the SubmitForm function. If the user selects that button and the data source is updated, the values in the form are reset to the default values of the data source.

Save changes

If you create a **Save changes** button as the previous section describes, the user can create or update a record and then select that button to save those changes to the data source. You could, instead, configure an Image control or some other control to perform the same task, as long as you configure that control with the SubmitForm function. In any case, the **Error**, **ErrorKind**, **OnSuccess**, and **OnFailure** properties provide feedback on the outcome.

When the SubmitForm function runs, it first validates the data that user wants to submit. If a required field doesn't contain a value or another value doesn't conform to some other constraint, the **ErrorKind** properties are set, and the **OnFailure** formula runs. You can configure the **Save changes** button or other control so that the user can select it only if the data is valid (that is, if the **Valid** property of the form is **true**). Note that the user must not only correct the problem but also select the **Save changes** button again (or discard the changes by selecting a **Cancel** button, as described earlier) to reset the **Error** and **ErrorKind** properties.

If the data passes validation, SubmitForm sends it to the data source, which can take some time depending on network latency.

- If the submission succeeds, the **Error** property is cleared, the **ErrorKind** property is set to **ErrorKind.None**, and the **OnSuccess** formula runs. If the user created a record (that is, if the form was previously in **New** mode), the form is switched to **Edit** mode so that the user can edit the newly created record or a different one.
- If the submission fails, the **Error** property contains a user-friendly error message from the data source, explaining the problem. The **ErrorKind** property is set

appropriately, depending on the issue, and the **OnFailure** formula runs.

Some data sources can detect when two people try to update the same record at the same time In this case, **ErrorKind** is set to **ErrorKind.Conflict**, and the remedy is to refresh the data source with the other user's changes and reapply the change made by this user.

> **Tip**
>
> If you offer a **Cancel** button on your form so that the user can abandon changes in progress, add the **ResetForm** function to the button's **OnSelect** property even that property also contains a **Navigate** function to change screens. Otherwise, the form will retain the user's changes.

Layout

By default, cards are placed in a single column for phone apps and three columns for tablet apps. You can specify how many columns a form has and whether cards should snap to them as you configure the form. These settings aren't exposed as properties because they're used only to set the **X**, **Y**, and **Width** properties of the cards.

For more information, see Understand data form layout.

Key properties

DataSource – The data source that contains the record that the user will show, edit, or create.

- If you don't set this property, the user can't show, edit, or create a record, and no additional metadata or validation is provided.

DefaultMode - The initial mode of the form control. See the description of **Mode** below for the acceptable values and their meanings.

DisplayMode - The mode to use for data cards and controls within the form control.

Derived from the **Mode** property based and cannot be set independently:

Expand table

Mode	DisplayMode	Description
FormMode.Edit	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept changes to a record.
FormMode.New	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept a new record.
FormMode.View	**DisplayMode.View**	Data cards and controls are not editable and optimized for viewing.

Error – A user friendly error message to display for this form when the SubmitForm function fails.

- This property applies only to the **Edit form** control.
- This property changes only when the SubmitForm, **EditForm**, or ResetForm function runs.
- If no error occurs, this property is *blank*, and **ErrorKind** is set to **ErrorKind.None**.
- When possible, the error message returned will be in the user's language. Some error messages come from the data source directly and may not be in the user's language.

ErrorKind – If an error occurs when **SubmitForm** runs, the kind of error that occurred.

- Applies only to an **Edit form** control.
- This property has the same enumeration as the Errors function. An **Edit form** control can return these values:

Expand table

ErrorKind	Description
ErrorKind.Conflict	Another user changed the same record, resulting in a change conflict. Execute the Refresh function to reload the record, and try the change again.
ErrorKind.None	The error is of an unknown kind.
ErrorKind.Sync	The data source reported an error. Check the **Error** property for more information.
ErrorKind.Validation	A general validation issue was detected.

Item – The record in the **DataSource** that the user will show or edit.

LastSubmit – The last successfully submitted record, including any server generated fields.

- This property applies only to the **Edit form** control.
- If the data source automatically generates or calculates any fields, such as an **ID** field with a unique number, the **LastSubmit** property will have this new value after **SubmitForm** successfully runs.
- The value of this property is available in the **OnSuccess** formula.

Mode – The control is in **Edit** or **New** mode.

Expand table

Mode	Description
FormMode.Edit	The user can edit a record by using the form. The values in the form's cards are pre-populated with the existing record, for the user to change. If the SubmitForm function runs successfully, an existing record is modified.
FormMode.New	The user can create a record by using the form. The values in the form's controls are pre-populated with the defaults for a record of the data source. If the SubmitForm function runs successfully, a record is created.
FormMode.View	The user can view a record by using the form. The values in the form's controls are pre-populated with the defaults for a record of the data source.

The form switches from **New** mode to **Edit** mode when any of these changes occurs:

- The form is successfully submitted, and a record is created. If the gallery is set to automatically move selection to this new record, the form will be in **Edit** mode for the created record so that the user can make additional changes.
- The EditForm function runs.
- The ResetForm function runs. For example, the user might select a **Cancel** button that's been configured with this function.

OnFailure – Actions to perform when a data operation has been unsuccessful.

- This property applies only to the **Edit form** control.

OnReset – Actions to perform when an **Edit form** control is reset.

- This property applies only to the **Edit form** control.

OnSuccess – Actions to perform when a data operation has been successful.

- This property applies only to the **Edit form** control.

Unsaved – True if the **Edit form** control contains user changes that have not been saved.

- This property applies only to the **Edit form** control.

- Use this property to warn the user before they lose any unsaved changes. To prevent the user from selecting a different record in a Gallery control before saving changes to the current record, set the gallery's Disabled property to **Form.Unsaved** and, likewise, disable refresh operations.

Updates – The values to write back to the data source for a record loaded in a form control.

- This property applies only to the **Edit form** control.
- Use this property to extract the field values from the cards within the control. You can then use these values to manually update the data source with a Patch function call or another method exposed by a connection. You do not need to use this property if you are using the SubmitForm function.
- This property returns a record of values. For example, if the form control contains card controls for **Name** and **Quantity** fields, and the values of the Update properties for those cards return "Widget" and 10 respectively, then the **Updates** property for the form control would return **{ Name: "Widget", Quantity: 10 }**.

Valid – Whether a Card or **Edit form** control contains valid entries, ready to be submitted to the data source.

- This property applies only to the **Edit form** control.
- A **Form** control's **Valid** property aggregates the **Valid** properties of all the Card controls in the form. A form's **Valid** property is **true** only if the data in all cards in that form is valid; otherwise, the form's **Valid** property is **false**.
- To enable a button to save changes only when the data in a form is valid but hasn't yet been submitted, set the button's **DisplayMode** property to this formula:

  SubmitButton.DisplayMode = If(IsBlank(Form.Error) || Form.Valid, DisplayMode.Edit, DisplayMode.Disabled)

Additional properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

More information

- For a comprehensive overview of how forms work, see Understand data forms.
- See the Power Fx formula reference for EditForm.

Accessibility guidelines

Screen reader support

- Consider adding a heading to the form using a **Label**.

Drop down control in Power Apps

Article • 03/13/2025

A list that shows only the first item unless the user opens it.

Description

A **Drop down** control conserves screen real estate, especially when the list contains many choices. The control takes up only one line unless the user selects the chevron to reveal more choices. The control shows a maximum of 500 items.

Key properties

Default – The initial value of a control before the user specifies a different value.

Items – The source of data that contains the items that appear in the control. If the source has multiple columns, set the control's **Value** property to the column of data that you want to show.

Value – The column of data that you want to show in the control (for example, if a data source has multiple columns).

Selected – The data record that represents the selected item.

AllowEmptySelection – Whether the control can have no selected items. When false, there will always be a selected item, which will initially be the **Default** or the first item.

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

ChevronBackground – The color behind the down arrow in a dropdown list.

ChevronFill – The color of the down arrow in a dropdown list.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnSelect – Actions to perform when the user selects a control.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user selects that control.

PressedColor – The color of text in a control when the user selects that control.

PressedFill – The background color of a control when the user selects that control.

Reset – Whether a control reverts to its default value.

SelectedText (Deprecated) – A string value that represents the selected item. Use **Dropdown1.SelectedText.Value** instead, replacing "Dropdown1" with drop-down control name in the app.

SelectionColor – The text color of a selected item or items in a list or the color of the selection tool in a pen control.

SelectionFill – The background color of a selected item or items in a list or a selected area of a pen control.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

> ⓘ **Note**
>
> Flyout border properties are controlled by **FocusedBorder** properties.

Examples

Simple list

1. Add a **Drop down** control and set its **Items** property to this expression:

   ```
   ["Seattle", "Tokyo", "London", "Johannesburg", "Rio de Janeiro"]
   ```

   Don't know how to add, name, and configure a control?

2. Show the items in the list by selecting the control's down arrow while pressing the **Alt** key.

List from a data source

The principles in this procedure apply to any data source that provides tables but, to follow these steps exactly, you must open an environment for which a Microsoft Dataverse database has been created and sample data added.

1. Open a blank app and add the **Accounts** table from Dataverse.

2. Add a **Drop down** control and set its **Items** property to this formula:

   ```
   Distinct(Accounts, 'Address 1: City')
   ```

   This formula shows all the cities in the **Accounts** table. If more than one record has the same city, the **Distinct** function hides the duplication in your drop-down control.

3. (Optional) Rename your **Drop down** control to **Cities**, add a vertical **Gallery** control, and set the gallery's **Items** property to this formula:

   ```
   Filter(Accounts, address1_city == Cities.Selected.Result)
   ```

   This **Filter** function shows only those records in the **Accounts** table for which the city matches the selected value in the **Cities** control.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **ChevronFill** and **ChevronBackground**
- **ChevronHoverFill** and **ChevronHoverBackground**
- **SelectionColor** and **SelectionFill**
- **SelectionFill** and **Fill**

This is in addition to the standard color contrast requirements.

Screen reader support

- **AccessibleLabel** must be present.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.

Feedback

Was this page helpful? Yes No

Provide product feedback

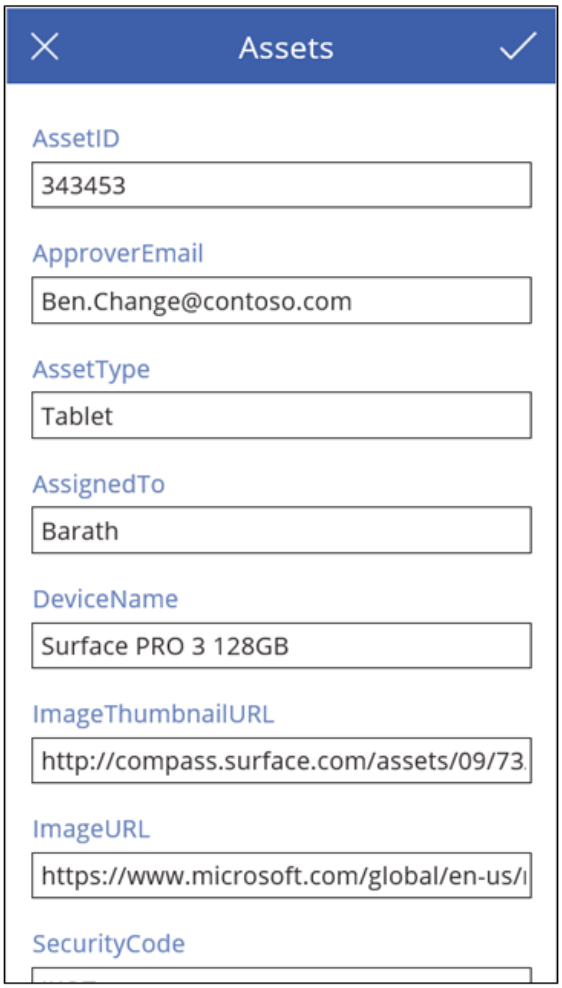
Edit form and Display form controls in Power Apps

Article • 03/26/2024

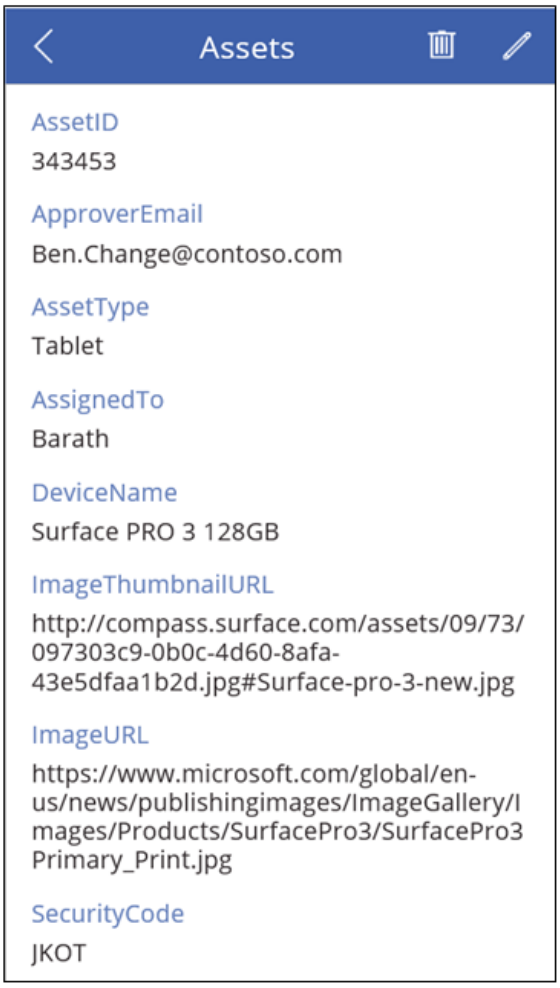
Display, edit, and create a record in a data source.

Description

If you add a **Display form** control, the user can display all fields of a record or only the fields that you specify. If you add an **Edit form** control, the user can edit those fields, create a record, and save those changes to a data source.



Edit form control
Read/Write



Display form control
Read only

If you add a **Gallery** control, you can configure it to show a table in a data source and then configure a form to show whichever record the user selects in the gallery. You can also add one or more **Button** controls that the user can select to save edits, cancel edits, and create a record. By using controls together, you can create a complete solution.

Limitations

The form control has these limitations:

- You can't copy and paste forms into data cards. This is to prevent certain combinations of controls from being created that risk the stability and performance of the app.

Record selection

For either type of form, you set its **DataSource** property to a table of records, and you set the form's **Item** property to show a specific record in that table. For example, you can set the **Item** property of a form to the **SelectedItem** property of a **Gallery** control. When the user selects a record in the gallery, the same record appears in the form, except that the form can show more fields. If the user returns to the gallery and selects a different record, the **SelectedItem** property of the gallery changes. This change updates the **Item** property of the form, which then shows the newly selected record.

You can also set a form's **Item** property by using a **Drop down** control, as Show, edit, or add a record describes, or a function such as **Lookup** or **First**. For example, you can set the **Item** property to either of these formulas to show the Fabrikam entry in the **Accounts** table in Microsoft Dataverse:

```
First(Accounts)
```

```
Lookup(Accounts, "Fabrikam" in name)
```

Each form control contains one or more **Card** controls. By setting the **DataField** property of a card, you specify which field that card shows and other details.

Create a record

When an **Edit form** control is in **Edit** mode, the user can update the record that's specified in the form's **Item** property. If inspected, the **Mode** property returns **Edit**.

When an **Edit form** control is in **New** mode, however, the **Item** property is ignored. The form doesn't show an existing record; instead, the values in each field match the default values of the data source with which you configured the form. The **NewForm** function causes a form to switch to this mode.

For example, you can set the **Text** property of a button to show **New** and its **OnSelect** property to a formula that includes the **NewForm** function. If the user selects that

button, the form switches to **New** mode so that the user can create a record starting with known values.

A form switches back to **Edit** mode if either the ResetForm function runs or the SubmitForm function runs successfully.

- You can set the Text property of a button to show **Cancel** and its OnSelect property to a formula that includes the ResetForm function. If the user selects that button, any changes in progress are discarded, and the values in the form, once again, match the default values of the data source.
- You can set the Text property of a button to show **Save changes** and its OnSelect property to a formula that includes the SubmitForm function. If the user selects that button and the data source is updated, the values in the form are reset to the default values of the data source.

Save changes

If you create a **Save changes** button as the previous section describes, the user can create or update a record and then select that button to save those changes to the data source. You could, instead, configure an Image control or some other control to perform the same task, as long as you configure that control with the SubmitForm function. In any case, the **Error**, **ErrorKind**, **OnSuccess**, and **OnFailure** properties provide feedback on the outcome.

When the SubmitForm function runs, it first validates the data that user wants to submit. If a required field doesn't contain a value or another value doesn't conform to some other constraint, the **ErrorKind** properties are set, and the **OnFailure** formula runs. You can configure the **Save changes** button or other control so that the user can select it only if the data is valid (that is, if the **Valid** property of the form is **true**). Note that the user must not only correct the problem but also select the **Save changes** button again (or discard the changes by selecting a **Cancel** button, as described earlier) to reset the **Error** and **ErrorKind** properties.

If the data passes validation, SubmitForm sends it to the data source, which can take some time depending on network latency.

- If the submission succeeds, the **Error** property is cleared, the **ErrorKind** property is set to **ErrorKind.None**, and the **OnSuccess** formula runs. If the user created a record (that is, if the form was previously in **New** mode), the form is switched to **Edit** mode so that the user can edit the newly created record or a different one.
- If the submission fails, the **Error** property contains a user-friendly error message from the data source, explaining the problem. The **ErrorKind** property is set

appropriately, depending on the issue, and the **OnFailure** formula runs.

Some data sources can detect when two people try to update the same record at the same time In this case, **ErrorKind** is set to **ErrorKind.Conflict**, and the remedy is to refresh the data source with the other user's changes and reapply the change made by this user.

> **Tip**
>
> If you offer a **Cancel** button on your form so that the user can abandon changes in progress, add the **ResetForm** function to the button's **OnSelect** property even that property also contains a **Navigate** function to change screens. Otherwise, the form will retain the user's changes.

Layout

By default, cards are placed in a single column for phone apps and three columns for tablet apps. You can specify how many columns a form has and whether cards should snap to them as you configure the form. These settings aren't exposed as properties because they're used only to set the **X**, **Y**, and **Width** properties of the cards.

For more information, see Understand data form layout.

Key properties

DataSource – The data source that contains the record that the user will show, edit, or create.

- If you don't set this property, the user can't show, edit, or create a record, and no additional metadata or validation is provided.

DefaultMode - The initial mode of the form control. See the description of **Mode** below for the acceptable values and their meanings.

DisplayMode - The mode to use for data cards and controls within the form control.

Derived from the **Mode** property based and cannot be set independently:

Expand table

Mode	DisplayMode	Description
FormMode.Edit	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept changes to a record.
FormMode.New	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept a new record.
FormMode.View	**DisplayMode.View**	Data cards and controls are not editable and optimized for viewing.

Error – A user friendly error message to display for this form when the SubmitForm function fails.

- This property applies only to the **Edit form** control.
- This property changes only when the SubmitForm, **EditForm**, or ResetForm function runs.
- If no error occurs, this property is *blank*, and **ErrorKind** is set to **ErrorKind.None**.
- When possible, the error message returned will be in the user's language. Some error messages come from the data source directly and may not be in the user's language.

ErrorKind – If an error occurs when **SubmitForm** runs, the kind of error that occurred.

- Applies only to an **Edit form** control.
- This property has the same enumeration as the Errors function. An **Edit form** control can return these values:

Expand table

ErrorKind	Description
ErrorKind.Conflict	Another user changed the same record, resulting in a change conflict. Execute the Refresh function to reload the record, and try the change again.
ErrorKind.None	The error is of an unknown kind.
ErrorKind.Sync	The data source reported an error. Check the **Error** property for more information.
ErrorKind.Validation	A general validation issue was detected.

Item – The record in the **DataSource** that the user will show or edit.

LastSubmit – The last successfully submitted record, including any server generated fields.

- This property applies only to the **Edit form** control.
- If the data source automatically generates or calculates any fields, such as an **ID** field with a unique number, the **LastSubmit** property will have this new value after **SubmitForm** successfully runs.
- The value of this property is available in the **OnSuccess** formula.

Mode – The control is in **Edit** or **New** mode.

Expand table

Mode	Description
FormMode.Edit	The user can edit a record by using the form. The values in the form's cards are pre-populated with the existing record, for the user to change. If the SubmitForm function runs successfully, an existing record is modified.
FormMode.New	The user can create a record by using the form. The values in the form's controls are pre-populated with the defaults for a record of the data source. If the SubmitForm function runs successfully, a record is created.
FormMode.View	The user can view a record by using the form. The values in the form's controls are pre-populated with the defaults for a record of the data source.

The form switches from **New** mode to **Edit** mode when any of these changes occurs:

- The form is successfully submitted, and a record is created. If the gallery is set to automatically move selection to this new record, the form will be in **Edit** mode for the created record so that the user can make additional changes.
- The EditForm function runs.
- The ResetForm function runs. For example, the user might select a **Cancel** button that's been configured with this function.

OnFailure – Actions to perform when a data operation has been unsuccessful.

- This property applies only to the **Edit form** control.

OnReset – Actions to perform when an **Edit form** control is reset.

- This property applies only to the **Edit form** control.

OnSuccess – Actions to perform when a data operation has been successful.

- This property applies only to the **Edit form** control.

Unsaved – True if the **Edit form** control contains user changes that have not been saved.

- This property applies only to the **Edit form** control.

- Use this property to warn the user before they lose any unsaved changes. To prevent the user from selecting a different record in a Gallery control before saving changes to the current record, set the gallery's Disabled property to **Form.Unsaved** and, likewise, disable refresh operations.

Updates – The values to write back to the data source for a record loaded in a form control.

- This property applies only to the **Edit form** control.
- Use this property to extract the field values from the cards within the control. You can then use these values to manually update the data source with a Patch function call or another method exposed by a connection. You do not need to use this property if you are using the SubmitForm function.
- This property returns a record of values. For example, if the form control contains card controls for **Name** and **Quantity** fields, and the values of the Update properties for those cards return "Widget" and 10 respectively, then the **Updates** property for the form control would return **{ Name: "Widget", Quantity: 10 }**.

Valid – Whether a Card or **Edit form** control contains valid entries, ready to be submitted to the data source.

- This property applies only to the **Edit form** control.
- A **Form** control's **Valid** property aggregates the **Valid** properties of all the Card controls in the form. A form's **Valid** property is **true** only if the data in all cards in that form is valid; otherwise, the form's **Valid** property is **false**.
- To enable a button to save changes only when the data in a form is valid but hasn't yet been submitted, set the button's **DisplayMode** property to this formula:

  SubmitButton.DisplayMode = If(IsBlank(Form.Error) || Form.Valid, DisplayMode.Edit, DisplayMode.Disabled)

Additional properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

More information

- For a comprehensive overview of how forms work, see Understand data forms.
- See the Power Fx formula reference for EditForm.

Accessibility guidelines

Screen reader support

- Consider adding a heading to the form using a **Label**.

Export and Import controls in Power Apps

Article • 12/16/2022

Controls for exporting data to a local file and then importing that data into another app in Power Apps.

Description

If you want to create more than one app that uses the same data but not share that data outside those apps, you can export it and import it by using an **Export** control and an **Import** control. When you export data, you create a compressed file that you can copy to another machine, but you can't read it in any program other than Power Apps.

Warning

Enabling this functionality in your app may expose it to security vulnerabilities and data leakage. It is recommended to advise users to import only recognized and trusted files and only export data that is not confidential or sensitive.

Limitations

The export functionality isn't supported in web browsers.

Key properties

Data – The name of a collection that you want to export to a local file.

- The **Data** property is available for an **Export** control but not an **Import** control.

OnSelect – Actions to perform when the user taps or clicks a control.

Additional properties

Align – The location of text in relation to the horizontal center of its control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

Padding – The distance between the text on an import or export button and the edges of that button.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Text – Text that appears on a control or that the user types into a control.

Underline – Whether a line appears under the text that appears on a control.

VerticalAlign – The location of text on a control in relation to the vertical center of that control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Example

1. Add a **Button** control, and set its **OnSelect** property to this formula:

   ```
   ClearCollect(Products, {Name:"Europa", Price:"10.99"},
   {Name:"Ganymede", Price:"12.49"}, {Name:"Callisto", Price:"11.79"})
   ```

   For more details, read adding, naming, and configuring a control, **ClearCollect** and other functions.
2. Press F5 and select **Button** control, and then press Esc.
3. Add an **Export** control, and set its **Data** property to **Products**.

4. Press F5 and select the **Export** control to download the file *Data.zip*.
5. Select **Save**, then press Esc to return to the default workspace.
6. In a new or existing app, add an **Import** control, name it **MyData**, and set its OnSelect property to this formula:
   Collect(ImportedProducts, MyData.Data)
7. Press F5 and select **MyData**, then select the file that you exported, and then select **Open**.
8. Press Esc and select **Collections** on the **File** menu, and confirm that the current app has the data that you exported.

Accessibility guidelines

The same guidelines for Button apply because **Export** and **Import** are just specialized buttons.

Gallery control in Power Apps

09/15/2025

A control that contains other controls and shows a set of data.

Description

A **Gallery** control can show multiple records from a data source, and each record can contain multiple types of data. For example, use a **Gallery** control to show multiple contacts with each item showing contact information that includes a name, an address, and a phone number for each contact.

Each data field appears in a separate control within the **Gallery** control. And you can configure those controls in its template. The template appears as the first item inside the gallery:

- On the left edge of a **Gallery** control in horizontal/landscape orientation.
- And at the top of a **Gallery** control in vertical/portrait orientation.

Any changes that you make in the template are reflected throughout the **Gallery** control.

Predefined templates for showing images and text in a gallery are available, and a gallery for variable-height items.

Limitations

The gallery control has these limitations:

1. If a user scrolls the **Flexible height** gallery control before all items are loaded, the item that's currently in view may be pushed down and out of view when the data loading is finished. To avoid this issue, use a standard **Gallery** control instead of the **Flexible height** variant.
2. The following controls are not supported within a gallery:

    - Display form
    - Edit form
    - PDF viewer
    - Power BI tile
    - Rich text editor
    - Scrollable screen (Fluid grid)
    - Web barcode scanner

3. You cannot add a gallery inside another gallery if the latter is already inside another gallery. The maximum level of nesting is 2.
4. The minimum **TemplateSize** is one. This can cause controls in the **Gallery** to drift from their expected **X** or **Y** values, when you don't account for the template size.
5. If you generate an app automatically from data, save the app, and then reopen it, the browse gallery might not immediately show any data. To resolve this issue, type at least one character in the search box, and then delete the text that you typed. The gallery will then show the data as expected.
6. If a gallery includes a ComboBox, DatePicker, Slider, or Toggle control with an `OnChange` rule that patches the same data source or collection as the gallery, it may lead to unexpected or reduced performance. This is because an `OnChange` rule that patches the data source of the gallery can create a never-ending cycle of patching and reloading. To avoid this issue, it is advisable not to use `OnChange` rules when these controls are placed within a gallery.

Key properties

Default – The item or record from the data source to be selected in the gallery when the app starts up.

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

Selected – The selected item.

Additional properties

AccessibleLabel – Label of the gallery (not the items it contains) for screen readers. Should describe what the list of items are.

AllItems – Items that are loaded in the gallery. This may be less than the actual number of **Items** of the data source. More items may be loaded when the gallery is scrolled. **AllItems** also contains controls that are used in each gallery item.

AllItemsCount - Count of items that are loaded in the gallery. This may be less than the actual number of **Items** of the data source. More items may be loaded when the gallery is scrolled.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DelayItemLoading - Delay loading of items (rows) until after the screen first loads.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

ItemAccessibleLabel – Label of each gallery item for screen readers. Should describe what each item is.

LoadingSpinner (**None**, **Controls** or **Data**) - When None, spinner will not be shown. When Controls | Data, spinner will be shown when a render pass occurs that results in visible empty rows.

LoadingSpinnerColor - The fill color of the loading spinner. Default is set to BorderColor.

NavigationStep – How far a gallery scrolls if its **ShowNavigation** property is set to **true** and the user selects a navigation arrow at either end of that gallery.

Selectable – Whether gallery items can be selected. When set to **true**, screen readers identify the gallery as a selectable list. And you select an item by selecting it. When set to **false**, screen readers identify the gallery as a regular list, and selecting an item doesn't select it.

ShowNavigation – Whether an arrow appears at each end of a gallery so that a user can scroll through the items in the gallery by selecting an arrow.

ShowScrollbar – Whether a scrollbar appears when the user hovers over a gallery.

TemplateFill – The background color of a gallery.

TemplatePadding – The distance between items in a gallery.

TemplateSize – The height of the template for a gallery in vertical/portrait orientation. Or the width of the template for a gallery in horizontal/landscape orientation. The minimum size is one.

Transition – The visual effect (**Pop**, **Push**, or **None**) when the user hovers over an item in a gallery.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

WrapCount – Number of items shown per row or column based on horizontal or vertical layout.

X – The distance between the left edge of a control and the left edge of its parent container or screen.

Y – The distance between the top edge of a control and the top edge of the parent container or screen.

Related functions

Filter(*DataSource*, *Formula*)

Reset(*Control*) - Resets your gallery back to its initial state. Initial state includes scrolling to the first item and selecting the first item or default if present.

> ⓘ **Note**
>
> **Reset** control does not recursively reset all the children of the gallery.

Examples

Show and filter data

- Show text
- Show images
- Filter data by selecting a list option
- Filter data by adjusting a slider

Get data from the user

- Get text
- Get images
- Get photos
- Get sounds
- Get drawings

Accessibility guidelines

Color contrast

If clicking anywhere in a gallery item is meant to select it, there must be adequate color contrast between:

- BorderColor and the color outside the gallery (if there is a border).
- Fill and the color outside the gallery (if there's no border).

Screen reader support

- AccessibleLabel must be present.

  > ⓘ Note
  >
  > Screen readers will announce when items in the gallery change. The **AccessibleLabel** is also mentioned. This gives context to the announcement and is even more important when there are multiple galleries on the same screen.

- When a gallery item contains multiple controls, use **ItemAccessibleLabel** to show the contents of gallery items.
- Set the value of **Selectable** to **true** if you want users to select a gallery item. Otherwise, set that value to **false**.
- When a gallery item contains multiple controls, use **ItemAccessibleLabel** to provide a summary of the gallery item's contents.
- **Selectable** should be set appropriately, depending on whether users are meant to select a gallery item.

Keyboard support

- Consider setting **ShowScrollbar** to **true**. On most touch screen devices, the scrollbar won't show until scrolling begins.
- If clicking anywhere in a gallery item is meant to select it, there must also be way for keyboard users to select the gallery item. For example, adding a Button that has its **OnSelect** property set to **Select(Parent)**.

  > ⓘ Note

Controls outside the gallery are not considered in the keyboard navigation order within the gallery. **TabIndex** controls inside a gallery are scoped. See **accessibility properties** to learn more.

See also

Use DelayItemLoading and Loading spinner to improve performance in Gallery

Grid container control in Power Apps (preview)

[This article is a pre-release document and is subject to change.]

The **Grid container** control gives you a way to **layout child components in a grid pattern**. You get precise control over where each component goes in terms of rows and columns.

This container works like CSS-style grids. You define the **rows and columns** and set the exact position for each child component using its properties.

> **Important**
>
> - This feature is in preview.
> - Preview features aren't meant for production use and might have restricted functionality These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is in the process of rolling out and might not be available in your region yet.

Description

The **Grid container** control offers a flexible layout system for canvas apps. It enables you to:

- Place child components in a **grid pattern**
- Specify the **row and column start and end** for each child
- Make responsive adjustments when the container or screen size changes
- Control alignment, spacing, and overflow within the grid

By using this approach, you don't need to manually position each component. Your apps become **more maintainable and responsive**.

Display properties

- **Gap** – The spacing in pixels between child components inside the grid.
- **Columns** – The number of columns in the grid layout.
- **Rows** – The number of rows in the grid layout.

Size and position

- **X** – The horizontal distance from the left edge of the parent container (or screen if no parent).
- **Y** – The vertical distance from the top edge of the parent container (or screen if no parent).
- **Width** – The distance between a control's left and right edges.
- **Height** – The distance between a control's top and bottom edges.
- **Padding** - The spacing between the container edges and its child components.

Color and border

- **Color** – Foreground color of the container, used for borders and text if applicable.
- **Border**
  - **Style** – Type of border: Solid, Dashed, Dotted, or None.
  - **Thickness** – Width of the border in pixels.
  - **Color** – Color of the border.
- **Border radius** – The degree to which the corners of the container are rounded. You can apply this setting to all corners uniformly or split it into individual corners in advanced settings.
- **Drop shadow** – Shadow effect applied to the container. Options: None, Light, Medium, or Heavy.

Visibility

- **Visible** – Whether the grid container is displayed. Toggle **On** or **Off**.

Grid layout properties (child-specific)

- **Column Start / Column End** – Defines the starting and ending columns for a child component.
- **Row Start / Row End** – Defines the starting and ending rows for a child component.

Example

1. Create a blank canvas app from scratch with a **Responsive** layout.
2. Create a new screen.
3. From the **Insert** pane under **Layout**, select **Grid container**.

4. Set container properties to occupy the full screen:

    - X = 0
    - Y = 0
    - Width = Parent.Width
    - Height = Parent.Height

5. Add several child components, such as buttons, text inputs, icons, and media. For each child, set grid placement properties:

    - Column Start / End
    - Row Start / End
    - Align in Cell

6. Press **F5** to preview. Resize the screen and observe how child components **remain in their grid positions** and adjust responsively.

Last updated on 12/18/2025

Horizontal container control in Power Apps

Article • 03/23/2023

Horizontal container control determines the position of the child components so that you never have to set X, Y for a component inside the container.

Description

Horizontal container control distributes the available space to its child components based on the settings, as well as determines alignment of the child components.

Properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color - The color of the container control.

Drop shadow - Adds shadow effects around the control.

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

Direction - Defines in what direction the container layouts its child components. **Horizontal** and **Vertical**.

Justify (vertical) - Defines how child elements are aligned with the primary axis. **Start**, **End**, **Center**, **Space Between**.

Align (Horizontal) - Defines how child components are positioned in the container, in the off axis (opposite from `LayoutDirection`). **Start**, **Center**, **End**, **Stretch**.

Gap - Defines the space between containers child components in pixels.

Horizontal Overflow - Defines whether the container shows scrollbars or remove content when it is too large to fit. **Scroll** and **Hide**.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

Vertical Overflow - Defines whether the container shows scrollbars or remove content when it is too large to fit. **Scroll** and **Hide**.

Wrap - Defines whether the content wraps to a new row or column when it cannot fit.

Align in container - Defines how the individual component is aligned to the parent. The default value, **Set by container**, inherits the value from the parent's `LayoutAlignItems` property, while other properties can be used to customize the alignment for the individual child component. **Set by container**, **Start**, **End**, **Center**, and **Stretch**.

Fill portions - Defines how the individual component grows when there is more screen real-estate assigned to its parent. The number represents the portion of the extra space given to the component out of all the available extra space claimed by children of its parent. For example, if child A has `Fill portions` set to 1 and child B has `Fill portions` set to 2, child A gets 1/3 of the extra space available while child B gets 2/3 of the extra available space.

Minimum width - Represent the minimum size of the component in the direction of the `Fill portions` (that is, the parent's `Direction`).

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

Visible – Whether a control appears or is hidden.

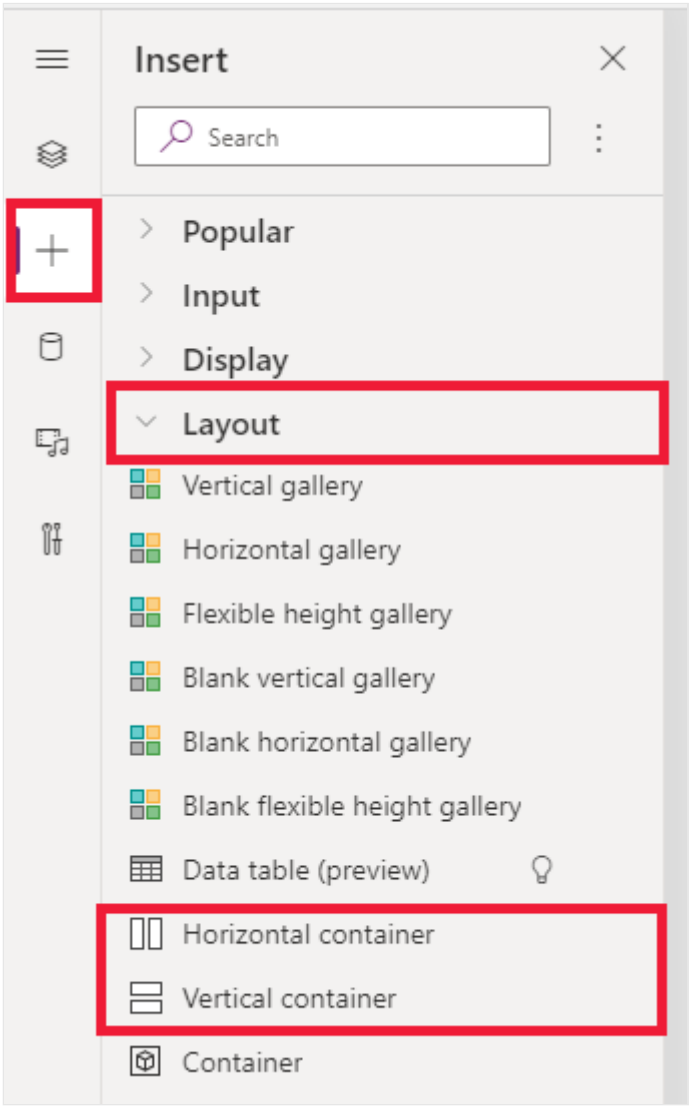
Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Example

1. Create a blank canvas app with **Tablet** layout.
2. Select **File** > **Settings** > **Screen size + orientation** and disable **Scale to fit**, **Lock aspect ratio**, and **Lock orientation** and select **Apply**.
3. Now from the **Insert** panes in the left sidebar, under **Layout** tab, select **Horizontal container**.



4. Set the following properties to occupy the full available space of the screen:
   a. X = 0
   b. Y= 0
   c. Width = Parent.Width
   d. Height = Parent. Height
5. Add few buttons, text inputs, media, icons, and select **F5** to see how the app adjusts to the screen changes.

HTML text control in Power Apps

Article • 05/22/2025

A box that shows text and converts HTML tags to formatting.

Description

An **HTML text** control not only shows plain text and numbers but also converts HTML tags, such as non-breaking spaces.

> **Note**
>
> HTML text control assumes the HtmlText is relatively positioned. If you need to use an absolute position for your HTML text, wrap the text around a relatively positioned div. For example, `"<div style='position:relative'>" & varPageContent & "</div>"`

> **Note**
>
> For some HTML elements default browser styling might be deleted. For instance, for HTML list (`<ul>`, `<ol>`) you will need to write your own inline styles to get the default styling back. For example,

HTML

```
<ul style='display: block;
           list-style-type: disc;
           margin-block-start: 1em;
           margin-block-end: 1em;
           margin-inline-start: 0px;
           margin-inline-end: 0px;
           padding-inline-start: 40px;'>
  ...
</ul>
```

Key properties

Color – The color of text in a control.

Font – The name of the family of fonts in which text appears.

HtmlText – Text that appears in an HTML text control and that may contain HTML tags.

Additional properties

AutoHeight – Set to true to allow the control to auto-grow its height to show all text. Set to false to truncate the text to the height assigned.

> ⓘ Note
>
> Enabling the **AutoHeight** property will grow the **Height** property of the control to a maximum value of 7680.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

OnSelect – Actions to do when the user selects a control.

> ⓘ Note
>
> **OnSelect** is ignored for hyperlinks within the content referenced inside **HtmlText** property.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

Size – The font size of the text that appears on a control.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Find(*FindString*, *WithinString*)

Example

1. Add a **Label** control, name it **Source**, and set its **Text** property to this string:

   "<p>We've done an unusually "deep" globalization and localization. <p>"

   Don't know how to add, name, and configure a control?

2. Add an **HTML text** control, and set its **HtmlText** property to this value:
   Source.Text

   The **HTML text** control shows the same text as the **Label** control but converts the tags to the appropriate characters.

Accessibility guidelines

ARIA mapping for elements inside the **HTML text** control are not defined automatically by Power Apps.

Color contrast

There must be adequate color contrast between:

- **Color** and **Fill**

- Text with custom colors and its background

Keyboard support

- The control can't act as a button. It doesn't have **TabIndex** and keyboard users won't be able to focus on it.
- The control can contain interactive parts in **HtmlText** like `<a>` elements, but the app setting **Simplified tab indexes" must be enabled. Otherwise, the tab navigation order will be wrong.

Shape and Icon controls in Power Apps

Article • 12/16/2022

Graphics for which you can configure appearance and behavior properties.

Description

These controls include arrows, geometric shapes, action icons, and symbols for which you can configure properties such as fill, size, and location. You can also configure their OnSelect property so that the app responds if the user selects the control.

Key properties (icons and shapes)

Fill – The background color of a control.

OnSelect – Actions to perform when the user selects a control.

Key properties (icons only)

Icon - The type of icon to display (for example, **ArrowDown** or **ShoppingCart**).

Rotation - The number of degrees to rotate the icon.

Color - The color of the icon by name or RGBA values.

Additional properties

AccessibleLabel – Label for screen readers.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

PressedBorderColor – The color of a control's border when the user selects that control.

PressedFill – The background color of a control when the user selects that control.

TabIndex – Keyboard-navigation order in relation to other controls.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Navigate(*ScreenName*, *ScreenTransition*)

Example

1. Name the default **Screen** control **Target**, add a **Label** control, and set its **Text** property to show **Target**.

   Don't know how to add and configure a control?

2. Add a **Screen** control, and name it **Source**.

3. In **Source**, add a **Shape** control, and set its **OnSelect** property to this formula:

```
Navigate(Target, ScreenTransition.Fade)
```

1. Press F5, and then select the **Shape** control.

   The **Target** screen appears.

2. (optional) Press Esc to return to the default workspace, add a **Shape** control to **Target**, and set the **OnSelect** property of the **Shape** control to this formula:

```
Navigate(Source, ScreenTransition.Fade)
```

Accessibility guidelines

Color contrast

The following applies only to graphics that are used as buttons or are otherwise not just for decoration.

For icons:

- **Color** and **Fill**
- Other standard color contrast requirements apply (if used as a button)

For shapes with borders:

- **BorderColor** and the color outside the control
- **FocusedBorderColor** and the color outside the control (if used as a button)

For shapes without borders:

- **Fill** and the color outside the control
- **PressedFill** and the color outside the control (if used as a button)
- **HoverFill** and the color outside the control (if used as a button)

Screen-reader support

- **AccessibleLabel** must be set for important graphics.
- If the graphic is for decoration or provides redundant information, leave **AccessibleLabel** empty or set it to the empty string "" . Screen readers will ignore these graphics.

For example, you might place a **Lock** icon next to a **Label** that says **This form cannot be modified**. You don't need an **AccessibleLabel** for the icon because the **Label** already explains its meaning.

> **Important**
>
> When **TabIndex** is zero or greater, the icon or shape becomes a button. Its appearance doesn't change, but screen readers will treat it as a button. They will not ignore the control, even if **AccessibleLabel** is empty. When **TabIndex** is less than zero, screen readers treat the icon or shape as an image.

Keyboard support

- **TabIndex** must be zero or greater if the graphic is used as a button. Keyboard users can then navigate to it.
- Focus indicators must be clearly visible if the graphic is used as a button. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this result.

Image control in Power Apps

Article • 01/19/2023

A control that shows an image from, for example, a local file or a data source.

Description

If you add one or more **Image** controls to your app, you can show individual images that aren't part of a data set, or you can incorporate images from records in data sources.

Key properties

Image – The name or the URL of the image that appears in an image, audio, or microphone control.

> **Note**
>
> - Use HTTPS for all external images to ensure compatibility with modern browsers.
> - External images must be accessible anonymously (without any authentication).

Additional properties

AccessibleLabel – Label for screen readers.

ApplyEXIFOrientation – Whether to automatically apply the orientation specified in the EXIF data embedded with the image.

AutoDisableOnSelect – Automatically disables the control while the OnSelect behavior is executing.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

CalculateOriginalDimensions – Enables the **OriginalHeight** and **OriginalWidth** properties.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FlipHorizontal – Whether to flip the image horizontally before displaying it.

FlipVertical – Whether to flip the image vertically before displaying it.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

ImagePosition – The position (**Fill**, **Fit**, **Stretch**, **Tile**, or **Center**) of an image in a screen or a control if it isn't the same size as the image.

ImageRotation – How to rotate the image before displaying it. Values can be none, clockwise (CW) 90 degrees, counter-clockwise (CCW) 90 degrees and clockwise 180 degrees.

OnSelect – Actions to perform when the user taps or clicks a control.

OriginalHeight – Original height of an image, enabled with the **CalculateOriginalDimensions** property.

OriginalWidth – Original width of an image, enabled with the **CalculateOriginalDimensions** property.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Transparency – The degree to which controls behind an image remain visible. Decimal values range from 0 to 1. Where 0 is opaque, 0.5 is semi-transparent and 1 is transparent.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Remove(*DataSource*, ThisItem)

Examples

Show an image from a local file

1. On the **File** menu, click or tap **Media**, and then click or tap **Browse**.
2. Click or tap the image file that you want to add, click or tap **Open**, and then press Esc to return to the default workspace.
3. Add an **Image** control, and set its **Image** property to the name of the file that you added.

   Don't know how to add and configure a control?

   The **Image** control shows the image that you specified.

Show a set of images from a data source

1. Download this Excel file, and save it on your local device.
2. In Power Apps Studio, create or open an app, and then click or tap **Add data source** in the right-hand pane.

   If **Add data source** doesn't appear in the right-hand pane, click or tap a screen in the left navigation bar.
3. Click or tap **Add static data to your app**, click or tap the Excel file that you downloaded, and then click or tap **Open**.
4. Select the **Flooring Estimates** check box, and then click or tap **Connect**.
5. Add a **Gallery** control with images, and set its **Items** property to **FlooringEstimates**.

   Don't know how to add and configure a control?

   The **Gallery** control shows images of carpet, hardwood, and tile products based on links in the Excel file that you downloaded.

Accessibility guidelines

Color contrast

- Standard color contrast requirements apply, if the graphic is used as a button.
- Consider checking for contrast issues within the image, if it is not purely decorative.

Screen reader support

- AccessibleLabel must be set for important graphics.
- If the graphic is for decoration or provides redundant information, leave AccessibleLabel empty or set it to the empty string "" . Screen readers will ignore these graphics.

For example, you might place an **Image** of a padlock next to a Label that says **This form cannot be modified**. You don't need an AccessibleLabel for the image because the **Label** already explains its meaning.

> ⓘ **Important**
>
> When TabIndex is zero or greater, the **Image** becomes a button. Its appearance doesn't change, but screen readers will treat it as a button. They will not ignore the control, even if **AccessibleLabel** is empty.

Keyboard support

- TabIndex must be zero or greater, if the graphic is used as a button. Keyboard users can then navigate to it.
- Focus indicators must be clearly visible, if the graphic is used as a button. Use FocusedBorderColor and FocusedBorderThickness to achieve this result.

Export and Import controls in Power Apps

Article • 12/16/2022

Controls for exporting data to a local file and then importing that data into another app in Power Apps.

Description

If you want to create more than one app that uses the same data but not share that data outside those apps, you can export it and import it by using an **Export** control and an **Import** control. When you export data, you create a compressed file that you can copy to another machine, but you can't read it in any program other than Power Apps.

Warning

Enabling this functionality in your app may expose it to security vulnerabilities and data leakage. It is recommended to advise users to import only recognized and trusted files and only export data that is not confidential or sensitive.

Limitations

The export functionality isn't supported in web browsers.

Key properties

Data – The name of a collection that you want to export to a local file.

- The **Data** property is available for an **Export** control but not an **Import** control.

OnSelect – Actions to perform when the user taps or clicks a control.

Additional properties

Align – The location of text in relation to the horizontal center of its control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

Padding – The distance between the text on an import or export button and the edges of that button.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Text – Text that appears on a control or that the user types into a control.

Underline – Whether a line appears under the text that appears on a control.

VerticalAlign – The location of text on a control in relation to the vertical center of that control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Example

1. Add a **Button** control, and set its **OnSelect** property to this formula:

   ```
   ClearCollect(Products, {Name:"Europa", Price:"10.99"},
   {Name:"Ganymede", Price:"12.49"}, {Name:"Callisto", Price:"11.79"})
   ```

   For more details, read adding, naming, and configuring a control, **ClearCollect** and other functions.
2. Press F5 and select **Button** control, and then press Esc.
3. Add an **Export** control, and set its **Data** property to **Products**.

4. Press F5 and select the **Export** control to download the file *Data.zip*.
5. Select **Save**, then press Esc to return to the default workspace.
6. In a new or existing app, add an **Import** control, name it **MyData**, and set its OnSelect property to this formula:
   Collect(ImportedProducts, MyData.Data)
7. Press F5 and select **MyData**, then select the file that you exported, and then select **Open**.
8. Press Esc and select **Collections** on the **File** menu, and confirm that the current app has the data that you exported.

Accessibility guidelines

The same guidelines for Button apply because **Export** and **Import** are just specialized buttons.

Column chart and Line chart controls in Power Apps

Article • 12/16/2022

Controls that show data as graphs with x- and y-axes.

Description

Column chart and **Line chart** are grouped controls. Each group contains three controls: a Label for the title, the chart graphic, and a **Legend**.

Chart key properties

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

NumberOfSeries – How many columns of data are reflected in a column or line chart.

Additional chart properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Font – The name of the family of fonts in which text appears.

GridStyle – Whether a column or line chart shows its x-axis, its y-axis, both, or neither.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

ItemColorSet – The color of each data point in a chart.

ItemsGap – The distance between columns in a column chart.

- The **ItemsGap** property is available for the **Column chart** control but not the **Line chart** control.

Markers – Whether a column or line chart shows the value of each data point.

MarkerSuffix – Text that appears after each value in a column chart for which the **Markers** property is set to **true**.

- The **MarkerSuffix** property is available for the **Column chart** control but not the **Line chart** control.

MinimumBarWidth – The narrowest possible width of columns in a column chart.

- The **MinimumBarWidth** property is available for the **Column chart** control but not the **Line chart** control.

OnSelect – Actions to perform when the user taps or clicks a control.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

SeriesAxisMax – The maximum value of the y-axis for a column or line chart.

- The **SeriesAxisMax** property is available for the **Column chart** control but not the **Line chart** control.

SeriesAxisMin – A number that determines the minimum value of the y-axis for a column chart.

- The **SeriesAxisMin** property is available for the **Column chart** control but not the **Line chart** control.

Size – The font size of the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

XLabelAngle – The angle of the labels below the x-axis of a column or line chart.

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

YAxisMax – The maximum value of the y-axis for a line chart.

- The **YAxisMax** property is available for the **Line chart** control but not the **Column chart** control.

YAxisMin – The minimum value of the y-axis for a line chart.

- The **YAxisMin** property is available for the **Line chart** control but not the **Column chart** control.

YLabelAngle – The angle of the labels next to the y-axis of a line or column chart.

Related functions

Max(*DataSource, ColumnName*)

Example

1. Add a **Button** control, and set its **OnSelect** property to this formula:
   Collect(Revenue, {Year:"2013", Europa:24000, Ganymede:22300, Callisto:21200}, {Year:"2014", Europa:26500, Ganymede:25700, Callisto:24700},{Year:"2014", Europa:27900, Ganymede:28300, Callisto:25600})

   Don't know how to add and configure a control?

   Want more information about the **Collect** function or other functions?

2. Press F5, click or tap the **Button** control, and then press Esc to return to the default workspace.

3. Add a **Column chart** control or a **Line chart** control, set its **Items** property to **Revenue**, and set its **NumberOfSeries** property to **3**.

   The control shows revenue data for each product over three years.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- each item in **ItemColorSet**
- every item in **ItemColorSet** and the background color
- **Color** and the background color

Screen reader support

- There must be a **Label** immediately before the chart graphic to serve as the title.
- Consider adding a summary of the chart graphic. For example, "The line chart shows a steady increase in sales between March and August this year."

  > ⓘ **Note**
  >
  > Chart graphics and **Legend** are hidden from screen reader users. As an alternative, a tabular form of the data is presented to them. They can also cycle through buttons that select data in the chart.

Low vision support

- There must be a **Legend** if more than one series is shown.
- Consider setting **GridStyle** to GridStyle.All, which shows both axes. This helps all users accurately determine the scale of the data.
- For **Column chart**, consider setting **Markers** to **true**. This helps low-vision users determine the value of a column.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.

Note

When keyboard users navigate to the chart, they can cycle through buttons that select data in the chart.

List Box control in Power Apps

Article • 01/20/2023

A list in which the user can select one or multiple items.

Description

A **List Box** control always shows all available choices (unlike a Drop down control) and in which the user can choose more than one item at a time (unlike a Radio control).

Key properties

Default – The initial value of a control before it is changed by the user.

> **Note**
>
> You can only have one default selected item. If you need multiple selected items please use the Combo Box control.

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

When you add a gallery, a list, or a chart, the property list shows **Items** by default so that you can easily specify the data that the new control should show. For example, you might set the **Items** property of a gallery to the **Account** table in Salesforce, a table named **Inventory** that you created in Excel and uploaded to the cloud, or a list named **ConferenceSpeakers**.

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

ItemPaddingLeft – The distance between text in a listbox and its left edge.

LineHeight – The distance between, for example, lines of text or items in a list.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnSelect – Actions to perform when the user taps or clicks a control.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

Reset – Whether a control reverts to its default value.

Selected – The data record that represents the selected item.

> ⓘ **Note**
>
> You can only have one default selected item. If you need multiple selected items please use the Combo Box control.

SelectedItems - **Read-only**. Represents a datatable of selected items for a multiselect listbox.

SelectedItemsText - **Read-only**. Represents a datatable of selected items text for a multiselect listbox.

SelectedText (Deprecated) – A string value that represents the selected item.

SelectionColor – The text color of a selected item or items in a list or the color of the selection tool in a pen control.

SelectionFill – The background color of a selected item or items in a list or a selected area of a pen control.

SelectMultiple – Whether a user can select more than one item in a listbox.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Distinct(*DataSource, ColumnName*)

Example

1. Add a **List box** control, name it **CategoryList**, and set its **Items** property to this formula:
   ["Carpet","Hardwood","Tile"]

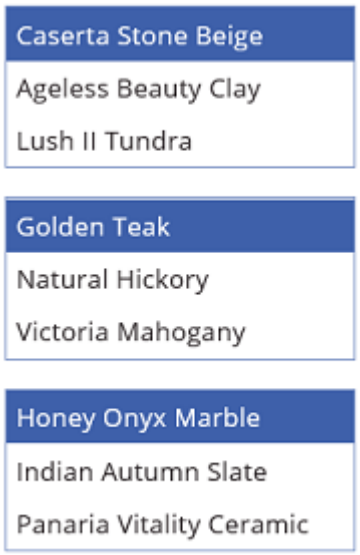
   Don't know how to add, name, and configure a control?

2. Add three **Drop down** controls, move them under **CategoryList**, and name them **CarpetList**, **HardwoodList**, and **TileList**.

3. Set the **Items** property of each **Drop down** control to one of these values:

   - CarpetList: **["Caserta Stone Beige","Ageless Beauty Clay", "Lush II Tundra"]**
   - HardwoodList: **["Golden Teak","Natural Hickory", "Victoria Mahogany"]**
   - TileList: **["Honey Onyx Marble","Indian Autumn Slate", "Panaria Vitality Ceramic"]**



4. Set the **Visible** property of each **Drop down** control to one of these values:

   - CarpetList: **If("Carpet" in CategoryList.SelectedItems.Value, true)**
   - HardwoodList: **If("Hardwood" in CategoryList.SelectedItems.Value, true)**
   - TileList: **If("Tile" in CategoryList.SelectedItems.Value, true)**

     Want more information about the **If** function or other functions?

5. Press F5, and then choose one or more items in **CategoryList**.

   The appropriate **Drop down** control or controls appear based on your choice or choices.



6. (optional) Press Esc to return to the default workspace.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **SelectionColor** and **SelectionFill**
- **SelectionFill** and **Fill**
- **HoverFill** and **Fill**
- **PressedFill** and **Fill**

This is in addition to the standard color contrast requirements.

Screen reader support

- **AccessibleLabel** must be present.

Keyboard support

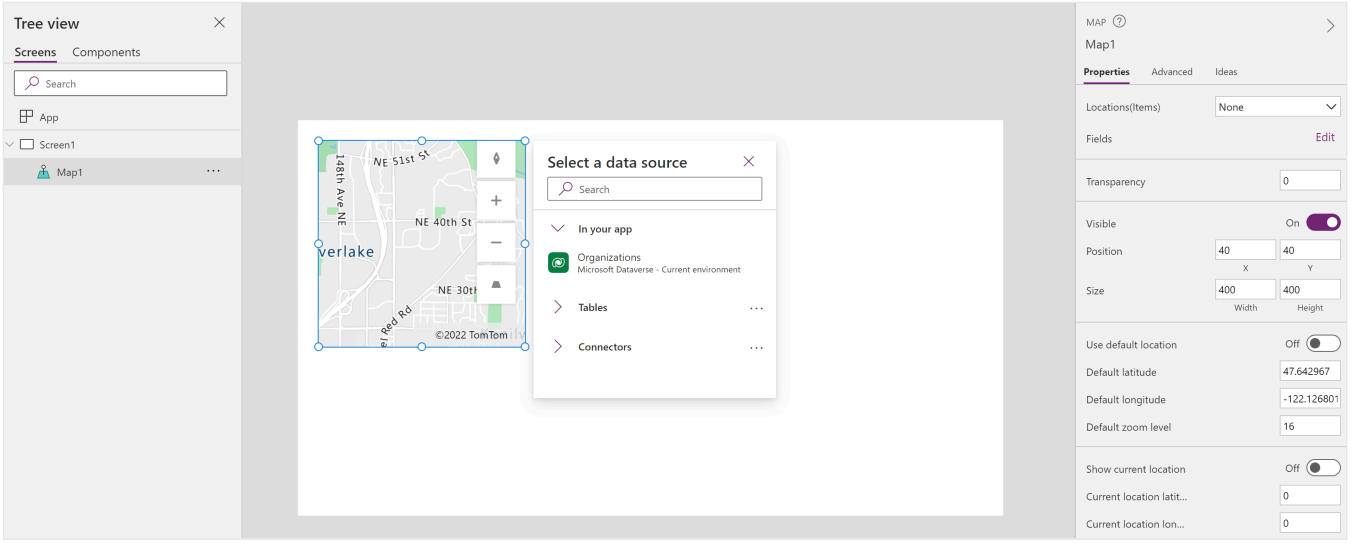
- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.

> ⓘ **Note**
>
> The tab key navigates to or away from the **List box**. Arrow keys navigate the contents of the **List box**.

Interactive map control

Article • 03/15/2023

Easily add an interactive map to your canvas apps. Plot markers from a data source containing addresses or latitude and longitude pairs. As you zoom out, the markers will optionally cluster to condense groups of data. On mobile devices and web experiences, a map can show the user's current position and calculate a route to the user's destination. Maps can be switched between road and satellite views.



Interactive map features

- Use a data source to insert pins
- Display information about map pins
- Show routes between waypoints
- Draw and insert shapes

Prerequisites

You'll need to satisfy a few prerequisites before you can use maps in your canvas apps. Refer to the privacy and security table for more details on how different map features use data.

Add a map to an app screen

With your app open for editing in Power Apps Studio :

1. Select the **Insert** tab and expand **Media**.
2. Select **Map** to place a map in the app screen, or drag the control to the screen to position it more precisely.

Adding pins, routes, and shapes

Pins, routes, and shapes are data set properties that need to both identify a data source, which is a table from a collection or connector, and the relevant columns. The data source is set in the Items property (Items for pins, RouteWaypoints_Items for routes, Shape_Items for shapes) and the relevant columns are set in the related properties (e.g. ItemLatitudes, ItemLongitudes, etc for pins). The Properties section contains additional information about these data sets and their related properties.

For example, if you had a table collection named **Locations** with **Name**, **Longitude**, and **Latitude** columns:

Expand table

Name	Longitude	Latitude
Fourth Coffee (sample)	-98.29277	26.2774
Litware, Inc. (sample)	-96.85572	32.55253
Adventure Works (sample)	-96.99952	32.72058

To display these as labeled pins on the map:

1. Set the **Items** property to Locations
2. Set the **ItemLabels** property to "Name"
3. Set the **ItemLongitude** property to "Longitude"
4. Set the **ItemLatitude** property to "Latitude"

> **Important**
>
> The relevant column names need to be in quotation marks in the related properties. The data source should not have quotation marks.

You can see how to do display pins from an Excel sheet or build an app that uses the address input control to populate pins on the map control as additional examples.

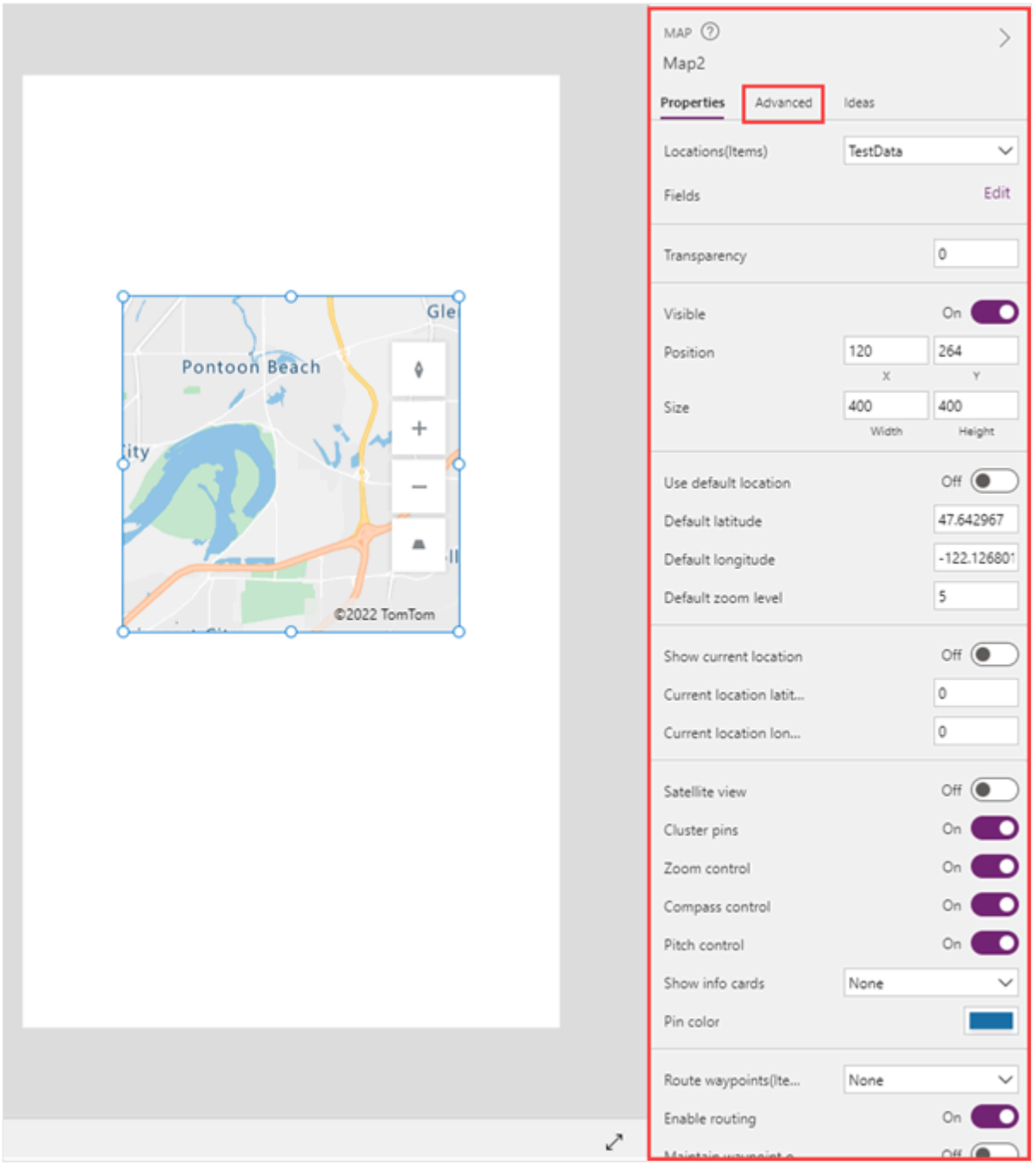
> **Note**
>
> - Each map control can display up to 5000 pins from latitude or longitude and 50 pins from addresses. The pin limit is lower for addresses as the map needs to geocode

these addresses into latitude or longitude to display them. We recommend not using addresses where possible. You can save geocoded addresses back to your data source.

- When both latitude or longitude and an address is given for a single pin, the map control will prioritize using the latitude or longitude to avoid geocoding the address.
- The maximum number of shapes that can be drawn in a map control is limited to 500.

Properties

Change a map's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



The map control has six different types of properties:

1. Styling properties
2. Behavioral properties
3. Pin properties
4. Route properties
5. Shape properties
6. Output properties

Styling properties

Expand table

Property	Description	Type	Tab
Satellite view	Displays the map in satellite view. Leave this property off to display the map in road view.	Boolean	Properties; Advanced: **SatelliteView**
Map style	Sets the map style. Options: Road, Night, Road shaded relief, Satellite, Satellite road labels, High contrast light, High contrast dark, Grayscale light, Grayscale dark.	Enum	Properties
Transparency	Determines the map's transparency, from 0 (opaque) to 100 (transparent).	Integer	Properties; Advanced: **Transparency**
Visible	Shows or hides the map.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the map at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the map using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Border radius	Determines the corner radius of the map border.	Floating point number	Properties; Advanced: **BorderRadius**
Border	Determines the style, width, and color of the map border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderColor**
DisplayMode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced

Behavioral properties

Expand table

Property	Description	Type	Tab
Show info cards	Shows information about a mapped location when the user selects it (*On click*) or points to it (*On hover*). If *None*, no information is shown.	Enum	Properties; Advanced: **InfoCards**
Use default location	Initializes the map at a default location.	Boolean	Properties; Advanced: **DefaultLocation**

Property	Description	Type	Tab
Default latitude	Sets the latitude coordinate the map shows if **Use default location** is enabled.	Floating point number	Properties; Advanced: **DefaultLatitude**
Default longitude	Sets the longitude coordinate the map shows if **Use default location** is enabled.	Floating point number	Properties; Advanced: **DefaultLongitude**
Default zoom level	Sets the zoom level if **Use default location** is enabled, from 0 to 22.	Integer	Properties; Advanced: **DefaultZoomLevel**
Show current location	Displays the user's current location.	Boolean	Properties; Advanced: **CurrentLocation**
Current location latitude	Sets the latitude coordinate of the current location pin that the map shows if **Show current location** is enabled. To place the pin at the user's current location, set this property to Location.Latitude.	Floating point number	Properties; Advanced: **CurrentLocationLatitude**
Current location longitude	Sets the longitude coordinate of the current location pin that the map shows if **Show current location** is enabled. To place the pin at the user's current location, set this property to Location.Longitude.	Floating point number	Properties; Advanced: **CurrentLocationLongitude**
Zoom control	Shows the zoom control.	Boolean	Properties; Advanced: **Zoom**
Compass control	Shows the compass control.	Boolean	Properties; Advanced: **Compass**
Pitch control	Shows the pitch (tilt) control.	Boolean	Properties; Advanced: **Pitch**
TabIndex	Specifies the order the map is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **Tab index**
Tooltip	Determines the text to display when the user hovers over a visual.	String	Advanced
ContentLanguage	Determines the display language of the map, if it's different from the language used in the app.	String	Advanced
OnLoad	Contains code that runs when the map is loaded.	Event	Advanced

Property	Description	Type	Tab
OnMapClick	Contains code that runs when the user selects the map. The latitude and longitude of the clicked point is in the **ClickedLocation** output property.	Event	Advanced
OnChange	Contains code that runs when any aspect of the map is changed.	Event	Advanced
OnSelect	Contains code that runs when the user selects something on the map.	Event	Advanced

Pin properties

Expand table

Property	Description	Type	Tab
Locations(Items)	Identifies a data source (**Items**) in the form of a table from which to get locations to show on the map. The table lists sets of longitudes and latitudes, or physical addresses, to display as pins. Using latitude or longitude is recommended as addresses need to be geocoded and so have a more restrictive pin limit. The table can be a collection or from a data source like Excel Online. Each row must have an entry for label, longitude, and latitude, or a physical address, and optionally the pin color and icon.	Not applicable	Properties; Advanced: **Items**
ItemsLabels	Identifies the column in **Items** that contains the labels for the pins.	ColumnName	Advanced
ItemsLatitudes	Identifies the column in **Items** that contains the latitude position of the pins.	ColumnName	Advanced
ItemsLongitudes	Identifies the column in **Items** that contains the longitude position of the pins.	ColumnName	Advanced
ItemsAddresses	Identifies the column in **Items** that contains addresses that represent the location of the pins. There is a limit to how many pins can be displayed from addresses. We recommend geocoding your addresses to latitude, longitude pairs and using those to display pins when possible.	ColumnName	Advanced
ItemsColors	Identifies the column in **Items** that contains the colors of the pins.	ColumnName	Advanced

Property	Description	Type	Tab
ItemsIcons	Identifies the column in **Items** that contains the icons of the pins.	ColumnName	Advanced
Cluster pins	Groups nearby map pins instead of displaying them individually.	Boolean	Properties; Advanced: **Clustering**
Pin color	Determines the default color of pins shown on the map. This color is overridden by the **ItemsColors** property if set	Color picker	Properties; Advanced: **PinColor**
OnItemsChange	Contains code that runs when pins on the map are changed.	Event	Advanced

Route properties

Expand table

Property	Description	Type	Tab
Enable routing	Determines whether the user can request directions to a specified location.	Boolean	Properties; Advanced: **UseRouting**
Route waypoints(Items)	Shows route waypoints, as provided in a data source (**RouteWaypoints_Items**) in the form of a table. The table can be a collection or from a data source like Excel Online. If *None*, no waypoints are shown.	Enum	Properties; Advanced: **RouteWaypoints_Items**
RouteWaypointsLabels	Identifies the column in **RouteWaypoints_Items** that contains the labels for the waypoints.	ColumnName	Advanced
RouteWaypointsLatitudes	Identifies the column in **RouteWaypoints_Items** that contains the latitude position of the waypoints.	ColumnName	Advanced
RouteWaypointsLongitudes	Identifies the column in **RouteWaypoints_Items** that contains the longitude position of the waypoints.	ColumnName	Advanced

Property	Description	Type	Tab
RouteWaypointsAddresses	Identifies the column in **RouteWaypoints_Items** that contains addresses that represent the location of waypoints.	ColumnName	Advanced
Maintain waypoint order	Determines whether a calculated route maintains waypoints in the order provided.	Boolean	Properties; Advanced: **RouteMaintainOrder**
Optimize route	Determines whether a calculated route is optimized for distance, time, or isn't optimized.	Enum	Properties; Advanced: **RouteOptimization**
Route travel mode	Determines whether a route is calculated for a car or a truck, which may require avoiding bridges with certain height or weight restrictions.	Enum	Properties; Advanced: **RouteTravelMode**
OnRouteDirectionChange	Contains code that runs when the app detects the user has changed direction while on a computed route.	Event	Advanced

Shape properties

Expand table

Property	Description	Type	Tab
Show shapes	Shows the shapes in **Shapes_Items**.	Boolean	Properties; Advanced: **ShowShapes**
Shapes_Items	Identifies a data source (**Shapes_Items**) in the form of a table from which to get shapes to show on the map. The table can be a collection or from a data source like Excel Online. Each row must have an entry for the shape (GeoJSON object) and (optionally) a label and color.	TableName	Advanced

Property	Description	Type	Tab
ShapeGeoJSONObjects	Identifies the column in **Shapes_Items** with strings that represent the GeoJSON objects of the shapes, in shape collection or single shape GeoJSON format.	ColumnName	Advanced
ShapeLabels	Identifies the column in **Shapes_Items** that contains the labels for the shapes.	ColumnName	Advanced
ShapeColors	Identifies the column in **Shapes_Items** that contains the colors of the shapes.	ColumnName	Advanced
Show shape labels	Shows the shape labels, if provided.	Boolean	Properties; Advanced: **ShowShapeLabels**
Enable shape drawing	Shows drawing tools on the map.	Boolean	Properties; Advanced: **ShapeDrawing**
Enable shape deleting and label editing	Determines whether the user can delete shapes and edit their labels.	Boolean	Properties; Advanced: **ShapeEditingDeleting**
OnShapeCreated	Contains code that runs when the user adds a shape to the map.	Event	Advanced
OnShapeSelected	Contains code that runs when the user selects a shape on the map.	Event	Advanced
OnShapeEdited	Contains code that runs when the user modifies a shape on the map.	Event	Advanced
OnShapeDeleted	Contains code that runs when the user deletes a shape on the map.	Event	Advanced

Output properties

Other properties become available when a user interacts with a map. You can use these *output properties* in other controls or to customize the app experience.

Expand table

Property	Description	Type
CenterLocation	Captures the map's center point.	Not Applicable

Property	Description	Type
ClickedLocation	Captures the last location the user selected, either *.Latitude* or *.Longitude*.	Record
Selected	Captures the selected pin.	Record
SelectedItems	Captures the selected pin or pins in the selected cluster.	Table
GeocodedItems	Captures the geocoded locations of the pins.	Table
RouteWaypoints_Selected	Corresponds to the record of the selected shape in **RouteWaypoints_Items**.	Record
RouteWaypoints_SelectedItems	Corresponds to the records of the selected overlapping shapes in **RouteWaypoints_Items**.	Table
Shapes_Selected	Corresponds to the record of the selected shape in **Shapes_Items**.	Record
Shapes_SelectedItems	Corresponds to the records of the selected overlapping shapes in **Shapes_Items**.	Table
SelectedShape	Captures the *.Perimeter* and *.Area* of the selected shape.	Record
DeletedShape	Captures the *.Perimeter* and *.Area* of the last deleted shape.	Record
GeoJSON	Captures the list of shapes in Feature Collection GeoJSON format.	String

Other geospatial controls

To see dynamic address suggestions as you type, use the **Address input** control.

Known limitations

- Requires internet connection

See also

Create an app with address input and map controls

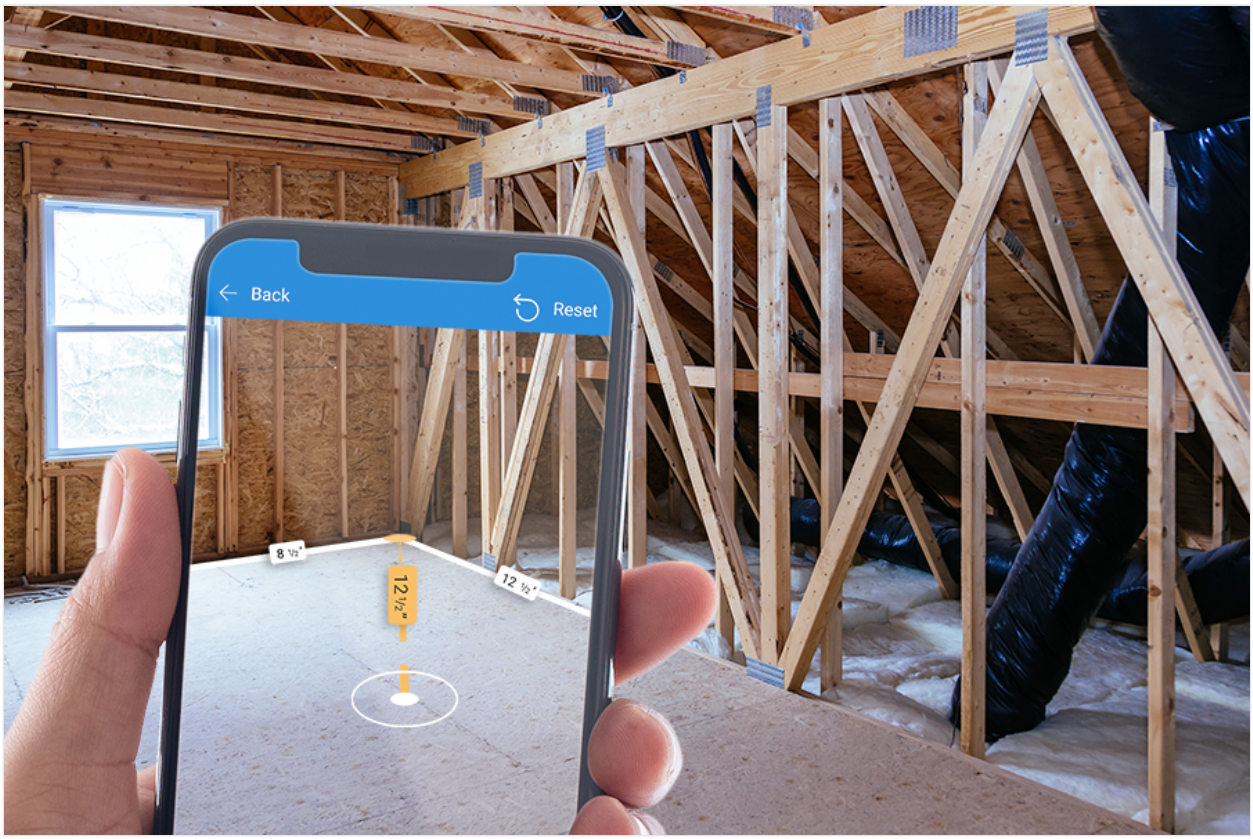
Take measurements in mixed reality

Article • 05/09/2022

Use the **Measuring camera** control in a canvas app to measure distance, area, and volume in the real world. You can also create two- and three-dimensional freeform shapes and see how they fit in a real-world space.

The **Measuring camera** control adds a button to your app. When a user selects the button, the app shows a live feed from the device camera. On-screen instructions direct the user to establish tracking by slowly panning the device from right to left across the surface to be measured. The user identifies a starting point and adds segments as needed until the measurement is complete. Dimensions of the measured space are overlaid on the camera feed.

The user can submit the measurements for capture in the control's **Measurements** output properties. The user can also take screenshots while a space is measured. The screenshots are captured in the control's **Photos** output property.



Tip

The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

Add a Measuring camera button to an app screen

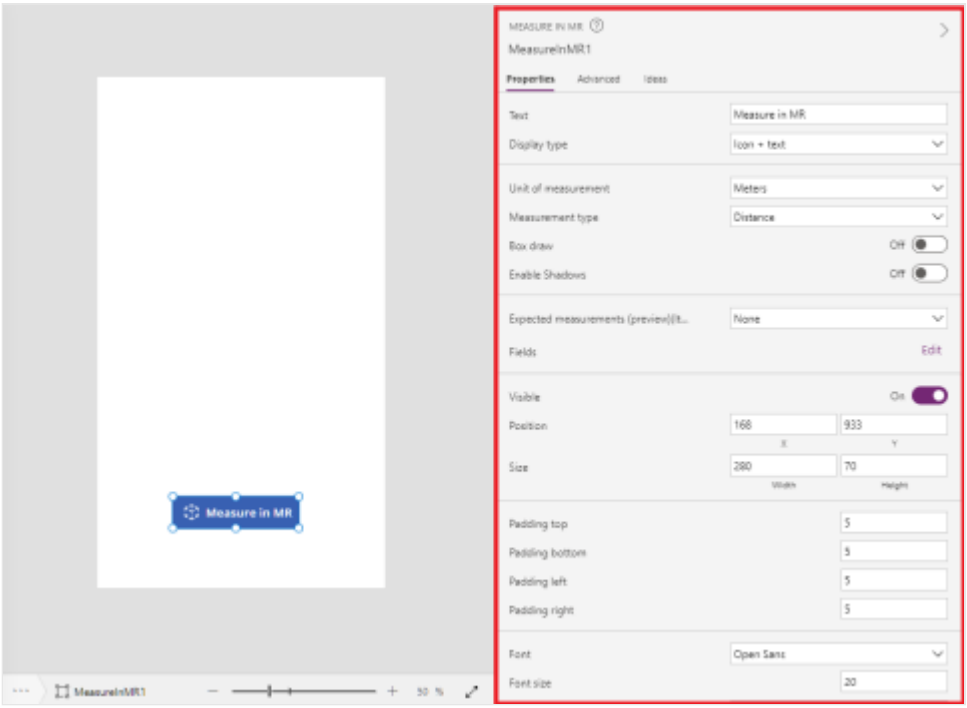
With your app open for editing in Power Apps Studio :

1. Open the **Insert** tab and expand **Media**.
2. Select **Measuring camera** to place the control in the app screen, or drag the control to the screen to position it more precisely.

The control is a button that's labeled with an icon of a cube and the text *Measure*. Change the label and hide the icon if you like in the **Text** and **Display type** properties.

Properties

Change the **Measuring camera** button's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Property	Description	Type	Location
Text	Sets the button label text.	String	Properties; Advanced: **Text**
Display type	Determines whether the button label shows an icon of a cube, text, or both.	Dropdown list	Properties; Advanced: **DisplayType**
Unit of measurement	Sets the unit used for the dimensions.	Dropdown list	Properties; Advanced: **Units**

Property	Description	Type	Location
Measurement type	Determines the kind of measurement the user can make, either *Distance*, *Area*, or *Volume* (area plus height or depth).	Dropdown list	Properties; Advanced: **MeasurementType**
Box draw	Locks captured measurements to rectangular shapes.	Boolean	Properties; Advanced: **BoxDraw**
Enable Shadows	Determines whether shadows are used to enhance the 3D effect when a shape is drawn.	Boolean	Properties; Advanced: **EnableShadows**
Expected measurements (Items)	Identifies a data source (**Items**) in the form of a table in an Excel workbook from which to get the measurements the user should capture.	Not applicable	Properties; Advanced: **Items**
Visible	Shows or hides the button.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the button at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the button using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Padding top	Sets the distance between the button label text and the top of the button.	Floating point number	Properties; Advanced: **PaddingTop**
Padding bottom	Sets the distance between the button label text and the bottom of the button.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the button label text and the left edge of the button.	Floating point number	Properties; Advanced: **PaddingLeft**

Property	Description	Type	Location
Padding right	Sets the distance between the button label text and the right edge of the button.	Floating point number	Properties; Advanced: **PaddingRight**
Font	Sets the name of the family of fonts used for the button label text.	Dropdown list	Properties; Advanced: **Font**
Font size	Sets the size of the button label text.	Floating point number	Properties; Advanced: **FontSize**
Font weight	Sets the weight of the button label text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	Dropdown list	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the label text in the button, either *Center*, *Justify*, *Left*, or *Right*.	Not applicable	Properties; Advanced: **TextAlignment**
Vertical alignment	Sets the vertical alignment of the label text in the button, either *Bottom*, *Middle*, or *Top*.	Dropdown list	Properties; Advanced: **VerticalAlign**
Font style	Sets the style of the button label text, either *Italic*, *Underline*, *Strikethrough*, or none.	Not applicable	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Border radius	Determines the corner radius of the button border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the button label text and the button background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the button border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderFillColor**
Disabled	Turns off the button but leaves it visible.	Boolean	Properties; Advanced: **Disabled**
Disabled color	Sets the colors of the button label text, the button background, and the button border if **DisplayMode** is **Disabled**.	Not applicable	Properties; Advanced: **DisabledContentColor**, **DisabledFillColor**, **DisabledBorderColor**

Property	Description	Type	Location
Pressed color	Sets the colors of the button label text, the button background, and the button border when the user selects the button.	Not applicable	Properties; Advanced: **PressedContentColor**, **PressedFillColor**, **PressedBorderColor**
Hover color	Sets the colors of the button label text, the button background, and the button border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverContentColor**, **HoverFillColor**, **HoverBorderColor**
OnMixedRealitySelect	Contains code that runs when the user exits the measurement screen with new results.	Event	Advanced
OnChange	Contains code that runs when a button property is changed.	Event	Advanced
OnSelect	Contains code that runs when the user selects the button.	Event	Advanced
ItemsLabels	Identifies the column in **Items** that contains the labels for the measurements users should capture.	ColumnName	Advanced
ItemsMeasurementType	(Optional) Identifies the column in **Items** that contains a string (*Distance*, *Area*, *Volume*, or *Freeform*) that specifies the type of measurement users should capture.	ColumnName	Advanced; see **MeasurementType**
ItemsBoxDraw	(Optional) Identifies the column in **Items** that contains a string (*True* or *False*) that locks measurements to rectangular shapes.	ColumnName	Advanced; see **BoxDraw**
Tooltip	Determines the text to display when the user hovers over a visual.	String	Advanced

Property	Description	Type	Location
ContentLanguage	Determines the display language of the control, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced
TabIndex	Specifies the order the control is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **Tab index**

Output properties

Your app can make use of more properties when a user interacts with the **View in MR** control. These are known as output properties. You can use these properties in other controls or to customize the app experience.

Property	Description
Photos	Captures, in a collection, information about photos the user takes while a space is measured. Use the collection to upload mixed-reality photos to OneDrive and show them in a gallery.
Measurements	Captures, in a table, the measured distances, volumes, and areas.
Segments	Captures, in a table, information about segments in a measured object.

The Measurements table has the following columns:

- **Label:** A string that identifies the measurement
- **Id:** A number that uniquely identifies the measurement
- **Units:** A string that identifies the unit of the measurement
- **Height:** A number that represents the height of the measured volume (0 if a 2D area was measured)
- **Length:** A number that represents the total length of the measured perimeter or path
- **BoundingWidth:** A number that represents the minimum width that bounds the shape

- **BoundingDepth:** A number that represents the minimum depth that bounds the shape
- **Area:** A number that represents the estimated area of the measured shape in units squared
- **Volume:** A number that represents the estimated volume of the measured shape in units cubed

The Segments table has the following columns:

- **Length:** A number that represents the total distance of a segment in the specified units
- **DirectionX:** A number that represents the X direction of the segment in world (3D) space
- **DirectionY:** A number that represents the Y direction of the segment in world(3D) space (typically 0)
- **DirectionZ:** A number that represents the Z direction of the segment in world (3D) space

Learn more about how to use output properties of the **Measuring camera** control in your apps: Validate measurements in mixed reality using a spatial test filter

Other mixed reality controls

- View 3D content with the **3D object** control.
- View images and 3D content in the real world with the **View in mixed reality** control.
- Create and view predefined 3D shapes with the **View shape in mixed reality** control
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

Create an app using the Measuring Camera control

Microphone control in Power Apps

Article • 03/25/2024

A control that enables app users to record sounds from their device.

Description

Use the **Microphone** control to capture audio with a device's microphone. The device must have a microphone and the user must authorize the app to use the microphone.

The most recently recorded audio clip is available through the **Audio** property. With this property, recorded audio can be:

- **Played back with the Audio control.** Use the Audio control to listen to the recording. For more information, see the examples.
- **Temporarily put in a variable or a collection.** Use the Set or Collect functions to store audio clips in a variable or a collection. Use caution with multiple audio clips in a collection at the same time with device's limited memory. Use the SaveData and LoadData functions to move audio clips to the local storage on the device and for offline scenarios.
- **Stored in a database.** Use the Patch function to store audio clips in a database.
- **Transmitted as a base64 encoded text string.** Use the JSON function to base64 encode audio clips.

Format of the recorded audio:

- *3gp* format for *Android*.
- *AAC* format for *iOS*.
- *OGG* format for *web browsers*.

Captured media is referenced by a text string URI. For more information, read the data type documentation.

> **Note**
>
> The microphone control is supported only on Microsoft Edge based on Chromium, Chrome, and Firefox browsers; and Android and iOS devices. All other browsers and platforms will show a warning that some features of the app won't work.

Limitations

The microphone control has these limitations:

- Power Apps Mobile runs on various types of devices, and some of them have limitations that are specific to that platform:
  - You can play videos in multiple **Video** controls at the same time on all platforms except iPhone devices.
  - You can record audio with multiple **Microphone** controls at the same time on all platforms except the web player.

Key properties

Audio – The audio clip captured when the user records with the device's microphone.

Mic – Numeric ID of the microphone on a device that has more than one microphone.

OnStop – Actions to perform when the user stops recording with a microphone control.

Additional properties

AccessibleLabel – Label for screen readers. Should describe the purpose of the microphone.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Image – The name of the image that appears in an image, audio, or microphone control.

ImagePosition – The position (**Fill**, **Fit**, **Stretch**, **Tile**, or **Center**) of an image in a screen or a control if it isn't the same size as the image.

OnSelect – Actions to perform when the user selects a control.

OnStart – Actions to perform when the user starts to record with a microphone control.

PressedBorderColor – The color of a control's border when the user selects that control.

PressedColor – The color of text in a control when the user selects that control.

PressedFill – The background color of a control when the user selects that control.

Reset – Whether a control reverts to its default value.

TabIndex – Keyboard navigation order compared to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container or screen.

Y – The distance between the top edge of a control and the top edge of the parent container or screen.

Examples

Simple direct playback

In this example, we'll directly connect a **Microphone** control with an **Audio** control for immediate playback:

1. Add a **Microphone** control to your app.
2. Authorize the app to use device's microphone if prompted.
3. Add an **Audio** control to your app.
4. Set the **Audio** control's **Media** property to the formula:

   ```
   Power Fx
   Microphone1.Audio
   ```

   > **Note**
   >
   > Replace microphone control name *Microphone1* as appropriate.

5. Preview the app.
6. Select the **Microphone** control to begin recording.
7. Speak to record audio.
8. Select the **Microphone** control again to end the recording.
9. Select the **Audio** control to hear the recording.

Add sounds to a Gallery control

In this example, we'll create a gallery of audio clips stored in a collection that can be individually selected for playback:

1. Add a **Microphone** control.
2. Set its **OnStop** property to this formula using the Collect function:

   ```
   Power Fx
   Collect( MySounds, MyMic.Audio )
   ```

3. Add a **Gallery** control, move it below **MyMic**.
4. Set the Items property for the gallery to this formula:

```
Power Fx

MySounds
```

5. In the template for the **Custom gallery** control, add an Audio control.
6. Set the audio control's **Media** property to this formula:

```
Power Fx

ThisItem.Url
```

7. Press F5 to preview the app.
8. Select **MyMic** to start recording, and then select it again to stop recording.
9. In the **Gallery** control, select the play button in the **Audio** control to play back your recording.
10. Add as many recordings as you want, and then return to the default workspace by pressing the Esc key.
11. (optional) In the template for the **Gallery** control, add a Button control.
12. Set its OnSelect property to the formula:

```
Power Fx

Remove( MySounds, ThisItem )
```

13. Press F5, and then remove a recording by selecting the corresponding **Button** control.

Use the SaveData function to save the recordings locally or the Patch function to update a data source.

Accessibility guidelines

The same guidelines for Button apply because **Microphone** is just a specialized button. Also, consider:

Audio alternatives

Consider adding an alternative form of input for users with speech disabilities or without a microphone. For example, Text input to allow users to enter text.

Color contrast

- Read the standard color contrast requirements.
- Ensure adequate color contrast between Image and the button text and icon (if applicable).

Screen reader support

- AccessibleLabel must be present.

See also

Limitations of controls in Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

PDF viewer control (experimental) in Power Apps

Article • 01/20/2024

An experimental control that shows the content of a PDF file.

Description

Show text, graphics, and other content in a PDF file by adding this type of control and setting its **Document** property to the URL, enclosed in double quotation marks, of the file that you want to show.

Limitations

1. The PDF viewer control only supports PDF files that conform to the PDF32000 specification.
2. The security architecture of Power Apps requires the PDF Viewer to support only HTTPS links, not HTTP.
3. The **Document** property must link directly to the PDF file. Server redirects or HTML views of the document aren't supported.
4. The file must be accessible anonymously without any authentication.
5. You won't be able to view a PDF document in your app if the document resides on a server that has restrictive cross-origin resource sharing (CORS) settings. To resolve this issue, the server that hosts PDF documents must permit cross-origin requests from powerapps.com.
6. Embedded objects like scripts and buttons won't run.

App users can work around these limitations by opening PDF documents in an external browser, as prompted if the control can't open a document. This option is also available in the control menu for all external documents.

Key properties

Document – The URL, enclosed in double-quotation marks, of a PDF file.

Additional properties

ActualZoom – The actual zoom of the control, which may differ from the zoom requested with the **Zoom** property.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

CurrentFindText – The current search term that is in use.

CurrentPage – The number of the page in a PDF file that is actually being shown.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FindNext – Finds the next instance of **FindText** in the document.

FindPrevious – Finds the previous instance of **FindText** in the document.

FindText – The search term to look for in the document.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

OnSelect – Actions to perform when the user taps or clicks a control.

OnStateChange – Actions to perform when the state of the control changes.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

Page – The number of the page that you want to show.

PageCount – The number of pages in a document.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

ShowControls – Whether an audio or video player shows, for example, a play button and a volume slider, and a pen control shows, for example, icons for drawing, erasing, and clearing.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Zoom – The percentage by which an image from a camera is magnified or the view of a file in a PDF viewer.

Keyboard shortcuts

- **J** - Page down
- **K** - Page up
- **N** - Page down
- **R** - Rotate all pages 90° clockwise

Example

Add a **PDF viewer** control, and set its **Document** property to the URL, enclosed in double quotation marks, of a PDF file as in this example:

"https://blog.mozilla.org/security/files/2015/05/HTTPS-FAQ.pdf"

The control shows the PDF file.

Don't know how to add and configure a control?

Accessibility guidelines

Not all accessibility features of PDF documents are supported because the **PDF viewer** is still in the experimental stage. Therefore, **ShowControls** should be set to **true** to allow users to open the document in an external application.

Learn how to create accessible PDF documents with the WCAG 2.0 and PDF/UA standards.

Screen reader support

- Consider adding a heading using a **Label**, if the PDF document does not have a title. The heading can be positioned immediately before the **PDF viewer**.

Pen input control in Power Apps

Article • 04/03/2024

A control in which the user can draw, erase, and highlight areas of an image.

Description

The user can use this control like a whiteboard, drawing diagrams and writing words that can be converted to typed text.

Limitations

The pen input control has these limitations:

- The pen control only has partial support for drawing using a mouse or touch input in the Windows app. Strokes might be intermittent. For smooth drawing, use a pen or run the app in a browser.

Key properties

Image – Output property that represents the image drawn by the end user.

Color – The color of input strokes.

Mode – The control is in **Draw** or **Erase** mode. Select mode has been deprecated.

Additional properties

AccessibleLabel – Label for screen readers. Can be used to describe the purpose of the control as well as alternative methods of input.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

Input – **Deprecated.** Whether the input supports mouse, pen, or touch inputs. Default value (7) supports all three.

OnSelect – Actions to perform when the user taps or clicks a control.

SelectionColor – The text color of a selected item or items in a list or the color of the selection tool in a pen control.

SelectionThickness – The thickness of the selection tool for a pen-input control.

ShowControls – Whether an audio or video player shows, for example, a play button and a volume slider, and a pen control shows, for example, icons for drawing, erasing, and clearing.

Size – The font size of the text that appears on a control.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Collect(*CollectionName*, *DatatoCollect*)

Example

Create a set of images

1. Add a **Pen input** control, name it **MyDoodles**, and set its **ShowControls** property to **true**.

   Don't know how to add, name, and configure a control?

2. Add a **Button** control, move it below **MyDoodles**, and set the **Text** property of the **Button** control to show **Add**.

3. Set the **OnSelect** property of the **Button** control to this formula:
   Collect(Doodles, {Sketch:MyDoodles.Image})

4. Add an **Image gallery** control, move it below the **Button** control, and shrink the width of the **Image gallery** control until it shows three items.

5. Set the **Items** property of the **Image gallery** control to **Doodles**, and then press F5.

6. Draw an image in **MyDoodles**, and then click or tap the **Button** control.

   The image that you drew appears in the **Image gallery** control.

7. (optional) In the **Pen input** control, click or tap the icon to clear the image that you drew, draw another image, and then click or tap the **Button** control.

8. In the **Image gallery** control, set the **OnSelect** property of the **Image** control to this formula:
   Remove(Doodles, ThisItem)

9. Remove a drawing by clicking or tapping it in the **Image gallery** control.

Use the **SaveData** function to save your drawings locally or the **Patch** function to save them to a data source.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **BorderColor** and the color outside the control (if there is a border)
- **Fill** and the color outside the control (if there is no border)

Screen reader support

- **AccessibleLabel** should be present.

> **Important**

Pen input is not accessible to screen reader users. Always provide an alternative form of input. For example, if a sketch is required, consider adding an Add picture control for users to upload an image. Both methods can be offered and the user can choose the one they are more comfortable with.

Keyboard support

Important

Pen input is not accessible to keyboard users. Always provide an alternative form of input. For example, if a signature is required, consider adding a Text input for users to enter their name. Both methods can be offered and the user can choose the one they are more comfortable with.

Pie chart control in Power Apps

Article • 12/16/2022

A control that shows relative values in comparison to each other.

Description

Add a **Pie chart** control if you want to show relative data from a table that contains labels in the leftmost column and values in the second column from the left.

This control is a grouped control containing three controls: a Label for the title, the chart graphic, and a **Legend**.

Chart key properties

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

ShowLabels – Whether a pie chart shows the value that's associated with each of its wedges.

Additional chart properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

Explode – The distance between wedges in a pie chart.

Font – The name of the family of fonts in which text appears.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

ItemBorderColor – The color of the border around each wedge in a pie chart.

ItemBorderThickness – The thickness of the border around each wedge in a pie chart.

ItemColorSet – The color of each data point in a chart.

LabelPosition – The location of labels in a pie chart relative to its wedges.

OnSelect – Actions to perform when the user taps or clicks a control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

Size – The font size of the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Max(*DataSource, ColumnName*)

Example

1. Add a **Button** control, and set its **OnSelect** property to this formula:
   Collect(Revenue2015, {Product:"Europa", Revenue:27000}, {Product:"Ganymede", Revenue:26300}, {Product:"Callisto", Revenue:29200})

   Don't know how to add and configure a control?

   Want more information about the **Collect** function or other functions?

2. Press F5, click or tap the **Button** control, and then press Esc to return to the default workspace.

3. Add a **Pie chart** control, and set its **Items** property to **Revenue2015**.

   The **Pie chart** control shows revenue data for each product in relation to the other products.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- each item in **ItemColorSet**
- every item in **ItemColorSet** and the background color
- **Color** and the background color

Screen reader support

- There must be a **Label** immediately before the chart graphic to serve as the title.

  > **Note**
  >
  > Chart graphics and **Legend** are hidden from screen reader users. As an alternative, a tabular form of the data is presented to them. They can also cycle through buttons that select data in the chart.

Low vision support

- There must be a **Legend**.
- Consider setting **ShowLabels** to **true**. This helps low-vision users quickly determine what each pie slice represents.
- Consider setting **LabelPosition** to **LabelPosition.Outside**. This increases legibility of labels because of a more consistent color contrast.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.

Note

When keyboard users navigate to the chart, they can cycle through buttons that select data in the chart.

Power BI tile control in Power Apps

A control that shows a Power BI tile inside an app.

Don't have Power BI? Sign up.

Description

Take advantage of your existing data analysis and reporting by displaying your **Power BI tiles** inside your apps. Specify the tile that you want to show by setting its **Workspace**, **Dashboard**, and **Tile** properties in the **Data** tab of the options panel.

> **Note**
>
> - Power BI tile control is not supported on some GCC High and DoD environments.
> - Power BI tile control isn't available in China.
> - For licensing requirements when embedding Power BI tile, see Licensing.

Sharing and security

When you share an app that contains Power BI content, you must share not only the app itself but also the dashboard where the tile comes from. Otherwise, the Power BI content won't appear even for users who open the app. Apps that contain Power BI content respect the permissions for that content.

Performance

It's not recommended to have more than three Power BI tiles loaded at the same time within an app. You can control tile loading and unloading by setting the **LoadPowerBIContent** property.

Embedding Options

Embedding is different between versions of the Power BI API. Because of the new Power BI API authentication scheme, your tile may not be accessible on mobile or within other embedded scenarios (Teams or SharePoint).

You can control the use of API version using the **AllowNewAPI** property. For more information, see Key properties.

Expand table

AllowNewAPI property value	Behavior
True	You can embed a dashboard, report, or tile by taking the **Embed URL** from Power BI and making it the **TileUrl** value.
False	You can embed a dashboard tile either by **Embed URL** and making it the **TileUrl** value, or using the graphical interface provided.

> **Note**
>
> Embedding a Power BI URL (for example, in an IFrame) is not supported on Power Apps mobile since authenticating the Power BI URL request is not handled by the mobile player.

Filtering

Filtering differs between the versions of the Power BI API. See the appropriate sections below depending on how you configure the control.

When using the new API to call Power BI service

When **AllowNewAPI** property is set to "True", you're using new API to call Power BI service. For more information, see Filter a report using query string parameters in the URL.

When using the original API to call Power BI service

When **AllowNewAPI** property is set to "False", you're using the original API to call Power BI service. In this case, by passing a single parameter from the app, you can filter the results that appear in a Power BI tile. However, only string values and the equals operator are supported, and the filter might not work if the table name or the column name contains spaces.

To pass a single filter value, modify the value of the **TileURL** property, which follows this syntax:

```
"https://app.powerbi.com/embed?dashboardId=<DashboardID>&tileId=<TileID>&config=
<SomeHash>"
```

To that value, append this syntax:

```
&$filter=<TableName>/<ColumnName> eq '<Value>'
```

For example, using a value from a list box:

```
"&$filter=Store/Territory eq '" & ListBox1.Selected.Abbr & "'"
```

The parameter will filter a value in the dataset of the report where the tile originates. However, filtering feature has the following limitations:

- Only one filter can be applied.
- Only the `eq` operator is supported.
- Field type must be string.
- Filtering is only available on pinned visualization tiles. It's not supported for pinned reports.
- R and Python script visuals cannot be filtered.

You can use computed fields in the Power BI report to convert other value types to string or combines multiple fields into one.

Key properties

AllowNewAPI - Whether to use the new API when calling the Power BI service. Setting the value to **True** will allow the use of the new Power BI API (which isn't supported in mobile and some embedded scenarios, but allows some more advanced filtering). **False** will use the original API. Default value is **false**.

Dashboard – The Power BI dashboard where the tile comes from.

LoadPowerBIContent – When set to **True**, the Power BI content is loaded and shown. When set to **False**, the Power BI content is unloaded, which releases memory and optimizes performance.

PowerBIInteractions - When set to **True**, the Power BI content can be interacted with, but Power Apps **OnSelect** event won't trigger. When set to **False**, the tile can't be interacted with, but Power Apps **OnSelect** event will trigger when the tile is selected.

Tile – The name of the Power BI tile that you want to display.

Workspace – The Power BI workspace where the tile comes from.

Additional properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Height – The distance between a control's top and bottom edges.

OnSelect – Actions to perform when the user selects a control. By default, the Power BI report that's associated with the tile opens.

TileUrl – The URL by which the tile is requested from the Power BI service. To add query string filtering to your URL, see the filtering section above.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen—if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen—if no parent container).

Example

1. On the **Insert** tab, open the **Charts** menu, and then add a **Power BI tile** control.

   Don't know how to add and configure a control?

2. On the **Data** tab of the options panel, select **My Workspace** for the **Workspace** setting.

3. Select a dashboard in the list of dashboards, and then select a tile in the list of tiles.

   The control renders the Power BI tile.

Accessibility guidelines

The **Power BI tile** is simply a container for Power BI content. Learn how to create accessible content with these Power BI accessibility tips.

If the Power BI content doesn't have a title, consider adding a heading using a **Label** control to support screen readers. You can position the label immediately before the Power BI tile.

Last updated on 12/19/2025

Radio control in Power Apps

Article • 12/16/2022

An input control that shows multiple options, of which users can select only one at a time.

Description

A **Radio** control, a standard HTML input control, is best used with only a few, mutually exclusive options.

The control can have a horizontal or vertical layout.

Key properties

Default – The value of a control before the user changes it.

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

Layout – Whether the options are laid out vertically or horizontally.

Value – The value of an input control.

Selected – The data record that represents the selected item.

All properties

Align – The location of text in relation to the horizontal center of its control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

LineHeight – The distance between, for example, lines of text or items in a list.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnSelect – Actions to perform when the user taps or clicks a control.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

RadioBackgroundFill – The background color of the circles in a radio-button control.

RadioBorderColor – The color of the circle for each option in a radio-button control.

RadioSelectionFill – The color that appears inside the circle of the selected option in a radio-button control.

RadioSize – The diameter of the circles in a radio-button control.

Reset – Whether a control reverts to its default value.

SelectedText (Deprecated) – A string value that represents the selected item.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard-navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Distinct(*DataSource, ColumnName*)

Example

1. Add a **Radio** control, name it **Pricing**, and set its **Items** property to this formula:

   ["Standard", "Premium"]

   Don't know how to add, name, and configure a control?

2. Add a **Label** control, move it below the **Radio** control, and set the **Text** property of the **Label** control to this formula:

   If("Premium" in Pricing.Selected.Value, "$200 per day", "$150 per day")

   Want more information about the **If** function or other functions?

3. While holding down the Alt key, select either option in the **Radio** control.

   The **Label** control shows the appropriate text for your choice.

4. (optional) While holding down the Alt key, select the other option to confirm that the appropriate text appears.

Accessibility guidelines

Color contrast

In addition to the standard color contrast requirements, ensure adequate color contrast between:

- **RadioSelectionFill** and **RadioBackgroundFill**
- **RadioBackgroundFill** and **Fill**

Screen-reader support

- Ensure that every option has a **Value**.
- Consider adding a **Label** immediately before the **Radio** control to serve as the heading.

Keyboard support

- Set the **TabIndex** property to zero or greater so that keyboard users can navigate to it.
- Set the **FocusedBorderColor** and **FocusedBorderThickness** properties so that focus indicators are clearly visible.

Rating control in Power Apps

Article • 12/16/2022

A control with which users can indicate a value between 1 and a maximum number that you specify.

Description

In this control, the user can indicate, for example, how much they liked something by selecting a certain number of stars.

Key properties

Default – The initial value of a control before it is changed by the user.

Max – The maximum value to which the user can set a slider or a rating.

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnSelect – Actions to perform when the user taps or clicks a control.

RatingFill – The color of the stars in a rating control.

ReadOnly – Whether a user can change the value of a slider or rating control.

Reset – Whether a control reverts to its default value.

ShowValue – Whether a slider's or rating's value appears as the user changes that value or hovers over the control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Average(*Value1*, *Value2*, ...)

Example

1. Add a **Rating** control, and name it **Quantitative**.

   Don't know how to add, name, and configure a control?

2. Add a **Text input** control, name it **Qualitative**, and move it below the **Rating** control.

3. Set the **Default** property of the **Text input** control to "", and set its **HintText** to this formula:
   If(Quantitative.Value > 3, "What did you especially like?", "How might we do better?")

   Want more information about the **If** function or other functions?

4. Press F5, and then click or tap either four or five stars in the **Rating** control.

The hint text in the **Text input** control changes to reflect the high rating.

5. Click or tap fewer than four stars in **Quantitative**.

   The hint text in the **Text input** control changes to reflect the low rating.

6. To return to the default workspace, press Esc.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **RatingFill** and **Fill**

This is in addition to the standard color contrast requirements.

Screen reader support

- **AccessibleLabel** must be present.

  > ⓘ **Note**
  >
  > Screen readers treat the **Rating** control as radio buttons.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.
- Consider using a different control if there are too many stars. It can be tedious to navigate with a keyboard and difficult to select accurately with a touch screen.

  > ⓘ **Note**
  >
  > The same keyboard interactions for radio buttons can be used on **Rating**.

Rich text editor control in Power Apps

Summarize this article for me

Let users format text in a WYSIWYG editing area. The output format is HTML.

Description

The **Rich text editor** control lets the app user format text in a WYSIWYG editing area.

The input and output format is HTML, but the control isn't an HTML editor. The editor removes script, style, object, and unsupported HTML elements and attributes.

Supported features include:

- Bold, italic, and underline
- Text and highlight color
- Text size
- Numbered and bulleted lists
- Hyperlinks
- Clear formatting

To use the control in a form, select the **Edit multi-line text** card, then customize it by inserting the RTE control.

Key properties

Default – Input property for the initial text value shown in the editor

HtmlText – Output property for the resulting rich text in HTML format

Additional properties

AccessibleLabel – Label for screen readers. Should describe the purpose of the attachments.

DisplayMode – Whether the control lets you add and delete files (**Edit**), only shows data (**View**), or is disabled (**Disabled**).

EnableSpellCheck – Whether the browser spell checker is enabled. This feature only checks spelling in the browser's default language. Power Apps for Windows doesn't support this property.

Height – The distance from the control's top edge to its bottom edge.

TabIndex – Keyboard navigation order in relation to other controls.

Visible – Whether a control is visible or hidden.

Width – The distance from the control's left edge to its right edge.

X – The distance from the control's left edge to the left edge of its parent container, or the screen if there's no parent container.

Y – The distance from the control's top edge to the top edge of its parent container, or the screen if there's no parent container.

Accessibility guidelines

Screen reader support

- Make sure AccessibleLabel is present.

Keyboard support

- Set TabIndex to zero or greater so keyboard users can navigate to it.

> **Tip**
>
> Select `Alt` + `0` while the editor is focused to learn about other keyboard shortcuts.

> **Note**
>
> When the toolbar is focused, select `Tab` or `Shift` + `Tab` to move between toolbar groups. You can't cycle back from the last group to the first group or vice versa.

Limitations

Appearance isn't guaranteed when working with other products

When you use rich text across different products, it might not look exactly the same.

- You can paste rich text from web pages, Microsoft Word, and other apps. However, the appearance might differ depending on the capabilities of the device, browser, and external source.
- Similarly, if you create rich text outside of canvas apps, it might look slightly different because built-in styles of the other app aren't present.
- If you embed the canvas app in another app, the host app might override the styles of the rich text. For example, when you use a canvas app as a custom page in a model-driven app, the host app removes list styles. Bulleted lists don't show bullet points in a custom page.

Sometimes, you might want rich text to display differently based on where it's shown. For example, the primary font might be different in another product, or the text color might change when the user enables dark mode. For a consistent display, compose and show rich text in canvas apps only.

Pasted images may not appear consistently

A pasted image might appear in a browser but not in a mobile app. Or it might appear intermittently or not at all. These are signs that the pasted image isn't supported because of:

- Cross-Origin Resource Sharing (CORS). The image host blocks the image from loading on Power Apps.
- Authentication. The image isn't publicly accessible and can only be accessed after you sign in to the image host.
- Image format support. Common image formats like jpg and png are supported, but less common types might not be supported by the browser or device.

Browsers or devices can represent images in rich text differently. Some copy the image as raw data, while others might simply reference the image's URL. Image URLs might not be accessible for the reasons above.

> **Tip**
>
> Use a screen clipping or screenshot tool to copy and paste images for the best results.

When editing an app, Alt key doesn't allow interaction with the control

In Power Apps Studio, pressing Alt key for quick interaction doesn't work. You can only interact with the rich text editor when previewing the app.

Hyperlinks don’t open in view mode

In view mode, the rich text editor can’t open hyperlinks while the app is running. This is a known limitation of the current editor.

Last updated on 02/10/2026

Screen control in Power Apps

Article • 11/14/2024

A UI element that contains one or more other controls in an app.

Description

Most apps have multiple **Screen** controls that contain **Label** controls, **Button** controls, and other controls that show data and support navigation. For information about how to add a screen, reorder screens, and configure navigation, review Add a screen.

Key properties

BackgroundImage – The name of an image file that appears in the background of a screen.

Fill – The background color of a control.

Additional properties

Height - The height of the screen. If the app is responsive (**Scale to fit** is **Off**) and the device on which the app is running is shorter than this property, the screen can scroll vertically.

ImagePosition – The position (**Fill**, **Fit**, **Stretch**, **Tile**, or **Center**) of an image in a screen or a control if it isn't the same size as the image.

LoadingSpinner (**None**, **Controls** or **Data**) - When None, spinner will not be shown. When Controls | Data, will show spinner until all child controls at the screen level are visible. **Note. Nested controls are not considered.**

LoadingSpinnerColor - The fill color of the loading spinner.

Name - The name of the screen.

OnHidden – The behavior of an app when the user navigates away from a screen. If the **Disable non-blocking App.OnStart** rule is enabled in app settings, this property may need to wait until **App.OnStart** is finished executing. By default, this setting is turned on.

OnVisible – The behavior of an app when the user navigates to a screen. Use this property to set up variables and preload data used by the screen. Use the **App.OnStart**

property for set up once when the app is started. If the **non-blocking App.OnStart** is enabled, the **Screen.OnVisible** event can run parallel with it. As a result, it's recommended to refrain from relying on variables or collections that are initialized by **App.OnStart**, as they might not be fully initialized.

Orientation - The orientation of the screen. If its **Width** is greater than its **Height**, the orientation will be **Layout.Horizontal**; otherwise, it will be **Layout.Vertical**.

Size - A positive integer that classifies the size of the screen. The classification is determined by comparing the screen's **Width** property to the values in the App.SizeBreakpoints property. The **ScreenSize** type consists of four values (**Small**, **Medium**, **Large**, and **ExtraLarge**) that correspond to the integers 1 through 4.

Width - The width of the screen. If the app is responsive (**Scale to fit** is **Off**) and the device on which the app is running is narrower than this property, the screen can scroll horizontally.

Related functions

Distinct(*DataSource, ColumnName*)

Example

1. Add a Radio control, name it **ScreenFills**, and set its Items property to this value:

   ```
   ["Red", "Green"]
   ```

   Don't know how to add, name, and configure a control?

2. Name the default **Screen** control **Source**, add another **Screen** control, and name it **Target**.

3. In **Source**, add a Shape control (such as an arrow), and set its OnSelect property to this formula:

   ```
   Navigate(Target, ScreenTransition.Fade)
   ```

   Want more information about the Navigate function or other functions?

4. In **Target**, add a Shape control (such as an arrow), and set its OnSelect property to this formula:

   ```
   Navigate(Source, ScreenTransition.Fade)
   ```

5. Set the Fill property of **Target** to this formula:

   ```
   If("Red" in ScreenFills.Selected.Value, RGBA(255, 0, 0, 1), RGBA(54, 176, 75, 1))
   ```

6. Select the **Source** screen and then, while holding down the Alt key, select either option in the Radio control, and then select the Shape control.

   Target appears in the color that you selected.

7. In **Target**, select the Shape control to return to **Source**.

8. (optional) Select the other option in the Radio control, and then select the Shape control to confirm that **Target** appears in the other color.

9. (optional) Reorder the screens by hovering over **Target** in the left navigation bar, selecting the ellipsis that appears, and then selecting **Move up**.

   Target appears first when the user opens the app.

Accessibility guidelines

Color contrast

When the **Screen** is the effective background for text, there must be adequate color contrast between:

- Fill and text
- BackgroundImage and text (if applicable)

For example, if a **Screen** contains a Label and the label has transparent fill, then the screen's Fill effectively becomes the background color for the label.

In addition to text, consider checking color contrast with essential graphical objects like the star images in a Rating control.

Screen reader support

- There must be a meaningful name for each **Screen**. The screen name can be viewed and edited in the same way as other controls: in the tree view of the controls pane or in the header of the properties pane.

  > ⓘ Note

When a new **Screen** is loaded, screen readers will announce its name.

Feedback

Was this page helpful? Yes No

Provide product feedback

Shape and Icon controls in Power Apps

Article • 12/16/2022

Graphics for which you can configure appearance and behavior properties.

Description

These controls include arrows, geometric shapes, action icons, and symbols for which you can configure properties such as fill, size, and location. You can also configure their OnSelect property so that the app responds if the user selects the control.

Key properties (icons and shapes)

Fill – The background color of a control.

OnSelect – Actions to perform when the user selects a control.

Key properties (icons only)

Icon - The type of icon to display (for example, **ArrowDown** or **ShoppingCart**).

Rotation - The number of degrees to rotate the icon.

Color - The color of the icon by name or RGBA values.

Additional properties

AccessibleLabel – Label for screen readers.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

PressedBorderColor – The color of a control's border when the user selects that control.

PressedFill – The background color of a control when the user selects that control.

TabIndex – Keyboard-navigation order in relation to other controls.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Navigate(*ScreenName*, *ScreenTransition*)

Example

1. Name the default **Screen** control **Target**, add a **Label** control, and set its **Text** property to show **Target**.

   Don't know how to add and configure a control?

2. Add a **Screen** control, and name it **Source**.

3. In **Source**, add a **Shape** control, and set its **OnSelect** property to this formula:

```
Navigate(Target, ScreenTransition.Fade)
```

1. Press F5, and then select the **Shape** control.

   The **Target** screen appears.

2. (optional) Press Esc to return to the default workspace, add a **Shape** control to **Target**, and set the **OnSelect** property of the **Shape** control to this formula:

```
Navigate(Source, ScreenTransition.Fade)
```

Accessibility guidelines

Color contrast

The following applies only to graphics that are used as buttons or are otherwise not just for decoration.

For icons:

- **Color** and **Fill**
- Other standard color contrast requirements apply (if used as a button)

For shapes with borders:

- **BorderColor** and the color outside the control
- **FocusedBorderColor** and the color outside the control (if used as a button)

For shapes without borders:

- **Fill** and the color outside the control
- **PressedFill** and the color outside the control (if used as a button)
- **HoverFill** and the color outside the control (if used as a button)

Screen-reader support

- **AccessibleLabel** must be set for important graphics.
- If the graphic is for decoration or provides redundant information, leave **AccessibleLabel** empty or set it to the empty string "" . Screen readers will ignore these graphics.

For example, you might place a **Lock** icon next to a **Label** that says **This form cannot be modified**. You don't need an **AccessibleLabel** for the icon because the **Label** already explains its meaning.

> **Important**
>
> When **TabIndex** is zero or greater, the icon or shape becomes a button. Its appearance doesn't change, but screen readers will treat it as a button. They will not ignore the control, even if **AccessibleLabel** is empty. When **TabIndex** is less than zero, screen readers treat the icon or shape as an image.

Keyboard support

- **TabIndex** must be zero or greater if the graphic is used as a button. Keyboard users can then navigate to it.
- Focus indicators must be clearly visible if the graphic is used as a button. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this result.

Slider control in Power Apps

Article • 12/16/2022

A control with which the user can specify a value by dragging a handle.

Description

The user can indicate a value, between a minimum and a maximum value that you specify, by dragging the handle of a slider left-right or up-down, depending on the direction that you choose.

Key properties

Default – The initial value of a control before it is changed by the user.

Max – The maximum value to which the user can set a slider or a rating.

Min – The minimum value to which the user can set a slider.

Value – The value of an input control.

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

HandleActiveFill – The color of the handle for a slider as the user changes its value.

HandleFill – The color of the handle (the element that changes position) in a toggle or slider control.

HandleHoverFill – The color of the handle in a slider when the user keeps the mouse pointer on it.

HandleSize – The diameter of the handle.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

Layout – Whether the user scrolls through a gallery or adjusts a slider top to bottom (**Vertical**) or left to right (**Horizontal**).

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnSelect – Actions to perform when the user taps or clicks a control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

RailFill – The background color of the rectangle in a toggle control when its value is **false** or the color of the line to the right of the handle in a slider control.

RailHoverFill – When you hover on a toggle control or a slider, the background color of the rectangle in a toggle control when its value is **false** or the color of the line to the right of the handle in a slider control.

ReadOnly – Whether a user can change the value of a slider or rating control.

Reset – Whether a control reverts to its default value.

ShowValue – Whether a slider's or rating's value appears as the user changes that value or hovers over the control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

ValueFill – The background color of the rectangle in a toggle control when its value is **true** or the color of the line to the left of the handle in a slider control.

ValueHoverFill – When you keep the mouse pointer on a toggle control or a slider, the background color of the rectangle in a toggle control when its value is **true** or the color

of the line to the left of the handle in a slider control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Sum(*Value1*, *Value2*)

Example

1. Add a button, and set its **OnSelect** property to this formula:
   ClearCollect(CityPopulations, {City:"London", Country:"United Kingdom", Population:8615000}, {City:"Berlin", Country:"Germany", Population:3562000}, {City:"Madrid", Country:"Spain", Population:3165000}, {City:"Rome", Country:"Italy", Population:2874000}, {City:"Paris", Country:"France", Population:2273000}, {City:"Hamburg", Country:"Germany", Population:1760000}, {City:"Barcelona", Country:"Spain", Population:1602000}, {City:"Munich", Country:"Germany", Population:1494000}, {City:"Milan", Country:"Italy", Population:1344000})

   Don't know how to add, name, and configure a control?

   Want more information about the **ClearCollect** function or other functions?

2. Press F5, select the button, and then press Esc.

3. Add a slider, move it below the button, and name the slider **MinPopulation**.

4. Set the slider's **Max** property to **5000000** and its **Min** property to **1000000**.

5. Add a text gallery in vertical/portrait orientation, move it below the slider, and set the gallery's **Items** property to this formula:
   Filter(CityPopulations, Population > MinPopulation)

6. In the first item of the gallery, set the **Text** property of the top label to **ThisItem.City**, and set the **Text** property of the bottom label to this formula:

Text(ThisItem.Population, "##,###")

7. Press F5, and then adjust **MinPopulation** to show only those cities that have a population that's greater than the value that you specify.

8. To return to the default workspace, press Esc.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **ValueFill** and **RailFill**
- **ValueHoverFill** and **RailHoverFill**
- FocusedBorderColor and color outside the control
- **ValueFill** and background color
- **RailFill** and background color
- **ValueHoverFill** and background color
- **RailHoverFill** and background color

Screen reader support

- AccessibleLabel must be present.

Keyboard support

- TabIndex must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use FocusedBorderColor and FocusedBorderThickness to achieve this.
- Slider value must be shown when interacting with the keyboard. This can be achieved by any one of these methods:
  - Set **ShowValue** to **true**.
  - Add a Label adjacent to the slider. Set the label's Text to the slider's Value.

Label control in Power Apps

Article • 12/16/2022

A box that shows data such as text, numbers, dates, or currency.

Description

A label shows data that you specify as a literal string of text, which appears exactly the way you type it, or as a formula that evaluates to a string of text. Labels often appear outside of any other control (such as a banner that identifies a screen), as a label that identifies another control (such as a rating or audio control), or in a gallery to show a specific type of information about an item.

Key properties

AutoHeight – Set to true to allow the label to auto-grow its height to show all text. Set to false to truncate the text to the height assigned.

Color – The color of text in a control.

Font – The name of the family of fonts in which text appears.

Text – Text that appears on a control or that the user types into a control.

DelayOutput – Set to true to delay action during text input.

Additional properties

Align – The location of text in relation to the horizontal center of its control.

AutoHeight – Whether a label automatically increases its **Height** property if its **Text** property contains more characters than the control can show at one time.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

LineHeight – The distance between, for example, lines of text or items in a list.

Live – How a screen reader announces changes to the value of the label's **Text** property.

- When set to **Off**, the screen reader doesn't announce changes.
- When set to **Polite**, the screen reader finishes speaking before announcing any changes that occurred while the screen reader was speaking.
- When set to **Assertive**, the screen reader interrupts itself to announce any changes that occurred while the screen reader was speaking.

OnSelect – Actions to perform when the user taps or clicks a control.

Overflow – Whether a scrollbar appears in a label if its **Wrap** property is set to **true** and the value of the control's Text property contains more characters than the control can show at one time.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

Role - The semantic role of the label text, such as Heading 1. Doesn't change the style of the label but makes the output semantically correct for interpretation by screen readers.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

VerticalAlign – The location of text on a control in relation to the vertical center of that control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

Wrap – Whether the text should ever wrap to multiple lines.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Text(*Number*, "*FormatCodes*")

Examples

Show a literal string

- Add a label, and set its **Text** property to **"Hello, world"** (including the double quotation marks).

  Don't know how to add and configure a control?

Show the result of a formula

- Add a label, and set its **Text** property to a formula such as this one:
  Today()

  > ⓘ **Note**
  >
  > When you specify a formula, you don't use quotation marks unless an argument of the formula is a literal string. In that case, enclose the argument, not the formula, in double quotation marks.

  Want more information about the **Today** function or other functions?

Show data in a gallery

In this procedure, you'll create a collection, called **CityPopulations**, that contains data about the population of various cities in Europe. Next, you'll show that data in a gallery that contains three labels, and you'll specify the type of data that each label will show.

1. Add a button, and set its **OnSelect** property to this formula:
   ClearCollect(CityPopulations, {City:"London", Country:"United Kingdom", Population:8615000}, {City:"Berlin", Country:"Germany", Population:3562000}, {City:"Madrid", Country:"Spain", Population:3165000}, {City:"Rome", Country:"Italy", Population:2874000}, {City:"Paris", Country:"France", Population:2273000}, {City:"Hamburg", Country:"Germany", Population:1760000}, {City:"Barcelona", Country:"Spain", Population:1602000}, {City:"Munich", Country:"Germany", Population:1494000}, {City:"Milan", Country:"Italy", Population:1344000})

2. Press F5, select the button, and then press Esc.
3. Add a text gallery, and set its **Items** property to **CityPopulations**.

   When the gallery is selected, the right pane shows options for that gallery.
4. In the **Gallery1** pane, set the top list to **Population**, set the middle list to **City**, and set the bottom list to **Country**.

Accessibility guidelines

Despite its name, a **Label** control does not have to be used as a label for another control. It can be used to display any piece of text.

A **Label** can be used as a button or link by specifying **OnSelect** behavior. When used this way, there are similar accessibility considerations as with buttons.

Color contrast

There must be adequate color contrast between:

- **Color** and **Fill**
- Other standard color contrast requirements apply (if used as a button or link)

Screen-reader support

- **Text** must be present.
- **Live** should be set to **Polite** or **Assertive** if a screen reader should announce changes to the value of the **Text** property.

  > ⓘ **Note**
  >
  > Screen readers will treat **Label**s as buttons when **TabIndex** is zero or greater.

Low vision support

- **Label** should look like a link, if it is used as a link.
  - Set **Underline** to **true**
  - **HoverColor** should be different from **Color**

Keyboard support

- **TabIndex** must be zero or greater, if the text is used as a button or link. This allows keyboard users to navigate to it.
- Focus indicators must be clearly visible, if the text is used as a button or link. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.

Text input control in Power Apps

Article • 12/16/2022

A box in which the user can type text, numbers, and other data.

Description

The user can specify data by typing into a text-input control. Depending on how you configure the app, that data might be added to a data source, used to calculate a temporary value, or incorporated in some other way.

Key properties

Default – The initial value of a control before it is changed by the user.

Text – Text that appears on a control or that the user types into a control.

Additional properties

AccessibleLabel – Label for screen readers.

Align – The location of text in relation to the horizontal center of its control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Clear – Whether a text-input control shows an "X" that the user can tap or click to clear the contents of that control. Only works in **SingleLine** mode.

Color – The color of text in a control.

DelayOutput – When set to true, user input is registered after half a second delay. Useful for delaying expensive operations until user completes inputting text (i.e. for filtering when input is used in other formulas).

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

EnableSpellCheck – Whether a text-input control should use the browser spell check function. Power Apps for Windows doesn't support this property.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Format – Whether the user input is restricted to numbers only or can be any text. When set to **Number**, only digits and certain numeric symbols can be entered. Supported symbols and formats depend on the browser language and cannot be set on individual controls.

Height – The distance between a control's top and bottom edges.

HintText – Light-grey text that appears in an input-text control if it's empty.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

LineHeight – The distance between, for example, lines of text or items in a list.

MaxLength – The number of characters that the user can type into a text-input control.

Mode – The control is in **SingleLine**, **MultiLine**, or **Password** mode.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnSelect – Actions to perform when the user taps or clicks a control.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

Reset – Whether a control reverts to its default value.

Size – The font size of the text that appears on a control.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

VirtualKeyboardMode – Type of virtual keyboard, text or numeric, that appears on an app user's touch screen. The **Format** property determines the default value. Device support varies. Devices that are running iOS must have at least version 12.2. The

recommended version of Android is 9.0, and capabilities of numeric keyboards vary for Android devices. Windows 10 doesn't support this property.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

DateTimeValue(*String*)

Examples

Collect data

1. Add two text-input controls, and name them **inputFirst** and **inputLast**.

   Don't know how to add, name, and configure a control?

2. Add a button, set its **Text** property to **Add**, and set its **OnSelect** property to this formula:
   Collect(Names, {FirstName:inputFirst.Text, LastName:inputLast.Text})

   Want more information about the **Collect** function or other functions?

3. Add a text gallery in portrait/vertical orientation, set its **Items** property to **Names**, and set the **Text** property of **Subtitle1** to **ThisItem.FirstName**.

4. (optional) In the template gallery, delete the bottom label, named **Body1**, and set the **TemplateSize** property of the gallery to **80**.

5. Press F5, type a string of text into **inputFirst** and **inputLast**, and then click or tap the **Add** button.

6. (optional) Add more names to the collection, and then press Esc to return to the default workspace.

Prompt for a password

1. Add a text-input control, name it **inputPassword**, and set its **Mode** property to **Password**.

2. Add a label, and set its **Text** property to this formula:
   If(inputPassword.Text = "P@ssw0rd", "Access granted", "Access denied")

   Want more information about the **If** function or other functions?

3. Press F5, and then type **P@ssw0rd** in **inputPassword**.

   When you finish typing the password, the label stops showing **Access denied** and starts to show **Access granted**.

4. To return to the default workspace, press Esc.

5. (optional) Add a control such as an arrow, configure it to navigate to another screen, and show it only after the user types the password.

6. (optional) Add a button, configure its **Text** property to show **Sign in**, add a timer, and disable the input-text control for a certain amount of time if the user types the wrong password and then clicks or taps the **Sign in** button.

Accessibility guidelines

Color contrast

- Standard color contrast requirements apply.

Screen reader support

- **AccessibleLabel** must be present.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.

Timer control in Power Apps

Article • 12/16/2022

A control that can determine how your app responds after a certain amount of time passes.

Description

Timers can, for example, determine how long a control appears or change other properties of a control after a certain amount of time has passed.

> ⓘ **Note**
>
> In Power Apps Studio, timers run only in Preview mode.

Key properties

Duration – How long a timer runs in milliseconds. The maximum is 24 hours expressed in milliseconds. Default is 60 seconds.

OnTimerEnd – Actions to perform when a timer finishes running.

Repeat – Whether a timer automatically restarts when it finishes running.

Additional properties

Align – The location of text in relation to the horizontal center of its control.

AutoPause – Whether the timer control automatically pauses if the user navigates to a different screen.

AutoStart – Whether the timer control automatically starts to play when the user navigates to the screen that contains that control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color – The color of text in a control.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Font – The name of the family of fonts in which text appears.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

Italic – Whether the text in a control is italic.

OnSelect – Actions to perform when the user taps or clicks a control.

OnTimerStart – Actions to perform when a timer starts to run.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

PressedColor – The color of text in a control when the user taps or clicks that control.

PressedFill – The background color of a control when the user taps or clicks that control.

Reset – Whether a control reverts to its default value.

Size – The font size of the text that appears on a control.

Start – Whether the timer starts.

Strikethrough – Whether a line appears through the text that appears on a control.

TabIndex – Keyboard navigation order in relation to other controls.

Text – Text that appears on a control or that the user types into a control.

Tooltip – Explanatory text that appears when the user hovers over a control.

Underline – Whether a line appears under the text that appears on a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

Refresh(*DataSource*)

Examples

Show a countdown

1. Add a timer, and name it **Countdown**.

   Don't know how to add, name, and configure a control?

2. Set the timer's **Duration** property to **10000** and its **Repeat** and **Autostart** properties to **true**.

3. (optional) Make the timer easier to read by setting its **Height** property to **160**, its **Width** property to **600**, and its **Size** property to **60**.

4. Add a label, and set its **Text** property to this formula:
   "Number of seconds remaining: " & RoundUp(10-Countdown.Value/1000, 0)

Want more information about the **RoundUp** function or other functions?

The label shows how many seconds remain before the timer restarts.

Animate a control

1. Add a timer, and name it **FadeIn**.

   Don't know how to add, name, and configure a control?

2. Set the timer's **Duration** property to **5000**, its **Repeat** property to **true**, and its **Text** property to **Toggle animation**.

3. (optional) Make the timer easier to read by setting its **Height** property to **160**, its **Width** property to **600**, and its **Size** property to **60**.

4. Add a label, set its **Text** property to show **Welcome!** and set its **Color** property to this formula:
   ColorFade(Color.BlueViolet, FadeIn.Value/5000)

   Want more information about the **ColorFade** function or other functions?

5. Select the timer button to start or stop the animation. The text in the label fades to white, returns to full intensity, and repeats the process.

Accessibility guidelines

The same guidelines for the **Button** control apply to the **Timer** control if users can interact with it.

Background timers

Background timers run automatically and are hidden. Use them in a supporting role where the elapsed time is of little interest to the user. For example, you can refresh data every minute or show a notification message only for a certain amount of time.

Background timers should have their **Visible** property set to false so that they are hidden from all users.

Timing considerations

If a **Timer** runs automatically, consider whether users have enough time to read and use content. Keyboard and screen-reader users may need more time to react to a timed

event.

Any of these strategies is sufficient:

- Allow users to cancel the timed event.
- Allow users to adjust the time limit before it begins.
- Warn 20 seconds before the time limit expires and provide an easy way to extend the limit.

Some scenarios are exempt from these requirements. Learn more in the WCAG 2.0 guideline for time limits.

Screen reader support

- If a timer triggers changes on the current screen, use a live region to tell screen-reader users what changed.

  > **Note**
  >
  > If the timer is visible and running, screen readers will announce the elapsed time every five seconds.

- Don't use the **Text** property of a control for time-sensitive and important information. Screen readers won't announce changes to **Text**.

- For interactive timers:
  - **Text** must be present.
  - Consider adding a **Label** control to show the elapsed time. Use the timer's **Text** property to instruct the user to start or stop the timer.

Toggle control in Power Apps

Article • 12/16/2022

A control that the user can turn on or off by moving its handle.

Description

A toggle is designed for recent GUIs but behaves the same way as a check box.

Key properties

Default – The initial value of a control before it is changed by the user.

Value – The value of an input control.

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

FalseFill – The toggle fill color when the toggle is off.

FalseHoverFill – The toggle hover fill color when toggle is off.

FalseText – The text shown when the toggle is off.

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

HandleFill – The fill color of the toggle handle.

Height – The distance between a control's top and bottom edges.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

OnCheck – Actions to perform when the value of a checkbox or a toggle changes to **true**.

OnSelect – Actions to perform when the user taps or clicks a control.

OnUncheck – Actions to perform when the value of a checkbox or a toggle changes to **false**.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

RailFill – The background color of the rectangle in a toggle control when its value is **false** or the color of the line to the right of the handle in a slider control.

RailHoverFill – When you hover on a toggle control or a slider, the background color of the rectangle in a toggle control when its value is **false** or the color of the line to the right of the handle in a slider control.

Reset – Whether a control reverts to its default value.

ShowLabel – Whether a text label is shown beside the toggle control.

TabIndex – Keyboard navigation order in relation to other controls.

TextPosition – Whether the label is to the left or the right of the toggle control.

Tooltip – Explanatory text that appears when the user hovers over a control.

TrueFill – Toggle fill color when the toggle is on.

TrueHoverFill – Toggle hover fill color when the toggle is on.

TrueText – Text shown when the toggle is on.

ValueFill – The background color of the rectangle in a toggle control when its value is **true** or the color of the line to the left of the handle in a slider control.

ValueHoverFill – When you keep the mouse pointer on a toggle control or a slider, the background color of the rectangle in a toggle control when its value is **true** or the color of the line to the left of the handle in a slider control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

If(*Condition*, *Result*)

Example

1. Add a toggle, and name it **MemberDiscount**.

   Don't know how to add, name, and configure a control?

2. Add a label, and set its **Text** property to this formula:
   If(MemberDiscount.Value = true, "Price: $75", "Price: $100")

   Want more information about the **If** function or other functions?

3. Press F5, and change the value of **MemberDiscount**.

   The label shows a different price, depending on whether **MemberDiscount** is on or off.

4. To return to the default workspace, press Esc.

Accessibility guidelines

Color contrast

There must be adequate color contrast between:

- **HandleFill** and **FalseFill**

- **HandleFill** and **FalseHoverFill**
- **HandleFill** and **TrueFill**
- **HandleFill** and **TrueHoverFill**
- **FalseFill** and color outside the control
- **FalseHoverFill** and color outside the control
- **TrueFill** and color outside the control
- **TrueHoverFill** and color outside the control

This is in addition to the standard color contrast requirements.

Screen reader support

- **AccessibleLabel** must be present.
- **FalseText** must be present.
- **TrueText** must be present.

Low vision support

- Consider setting **ShowLabel** to **true** so that users can quickly determine the toggle value.

Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.

Vertical container control in Power Apps

Article • 03/23/2023

Vertical container control determines the position of the child components so that you never have to set X, Y for a component inside the container.

Description

Vertical container control distributes the available space to its child components based on the settings, as well as determines alignment of the child components.

Properties

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

Color - The color of the container control.

Drop shadow - Adds shadow effects around the control.

Fill – The background color of a control.

Height – The distance between a control's top and bottom edges.

Direction - Defines in what direction the container layouts its child components. **Horizontal** and **Vertical**.

Justify (vertical) - Defines how child elements are aligned with the primary axis. **Start**, **End**, **Center**, **Space Between**.

Align (horizontal) - Defines how child components are positioned in the container, in the off axis (opposite from `LayoutDirection`). **Start**, **Center**, **End**, **Stretch**.

Gap - Defines the space between containers child components in pixels.

Horizontal Overflow - Defines whether the container shows scrollbars or remove content when it is too large to fit. **Scroll** and **Hide**.

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

Vertical Overflow - Defines whether the container shows scrollbars or remove content when it is too large to fit. **Scroll** and **Hide**.

Wrap - Defines whether the content wraps to a new row or column when it cannot fit.

Align in container - Defines how the individual component is aligned to the parent. The default value, **Set by container**, inherits the value from the parent's `LayoutAlignItems` property, while other properties can be used to customize the alignment for the individual child component. **Set by container**, **Start**, **End**, **Center**, and **Stretch**.

Fill portions - Defines how the individual component grows when there is more screen real-estate assigned to its parent. The number represents the portion of the extra space given to the component out of all the available extra space claimed by children of its parent. For example, if child A has `Fill portions` set to 1 and child B has `Fill portions` set to 2, child A gets 1/3 of the extra space available while child B gets 2/3 of the extra available space.

Minimum width - Represent the minimum size of the component in the direction of the `Fill portions` (that is, the parent's `Direction`).

PaddingBottom – The distance between text in a control and the bottom edge of that control.

PaddingLeft – The distance between text in a control and the left edge of that control.

PaddingRight – The distance between text in a control and the right edge of that control.

PaddingTop – The distance between text in a control and the top edge of that control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

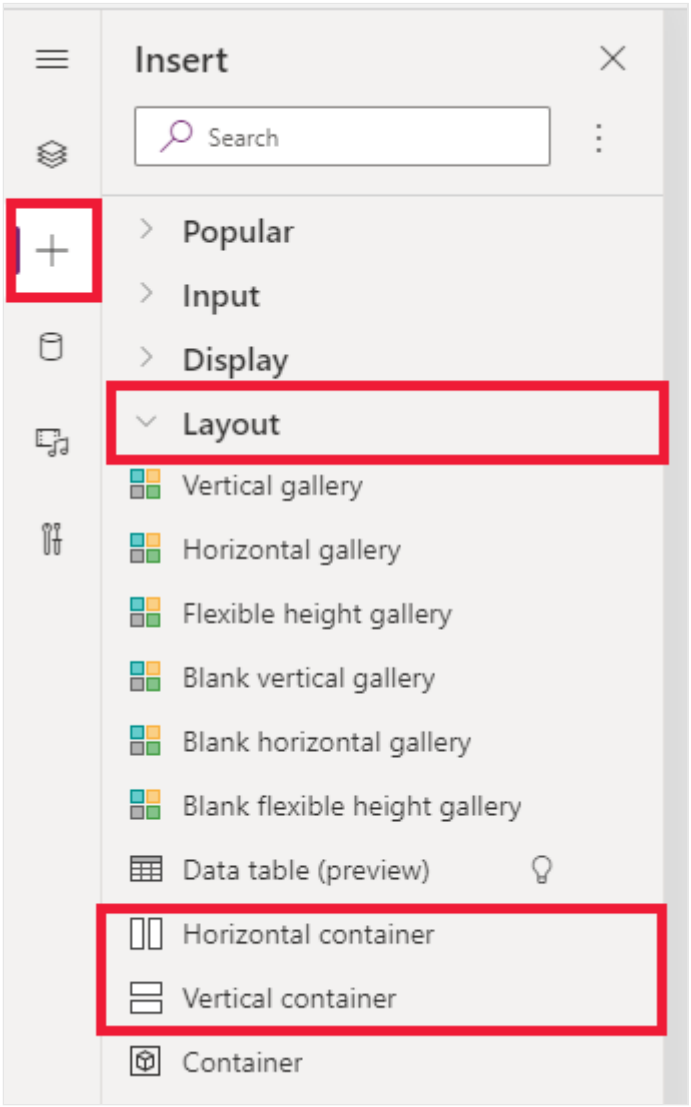
Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Example

1. Create a blank canvas app with **Tablet** layout.

2. Select **Settings** > **Display** and disable **Scale to fit**, **Lock aspect ratio**, and **Lock orientation** and select **Apply**.

3. Now from the **Insert** panes in the left sidebar, under **Layout** tab, select **Vertical container**.



4. Set the following properties to occupy the full available space of the screen:
   a. X = 0
   b. Y= 0
   c. Width = Parent.Width
   d. Height = Parent. Height

5. Add few buttons, text inputs, media, icons, and select **F5** to see how the app adjusts to the screen changes.

Audio and Video controls in Power Apps (retired)

Article • 04/03/2024

A control that plays an audio file, a video file, or a video on YouTube.

Description

An **Audio** control plays a sound clip from a file, a recording from a Microphone control, or the audio track from a video file.

A **Video** control plays a video clip from a file or from YouTube or any external source. Closed captions can optionally be shown when specified.

> **Important**
>
> Videos on Azure Media services are no longer supported. For more information, see Retirement of the service.

Limitations

The audio and video control has these limitations:

- Power Apps Mobile runs on various types of devices, and some of them have limitations that are specific to that platform:
  - You can play videos in multiple **Video** controls at the same time on all platforms except iPhone devices.
  - You can record audio with multiple **Microphone** controls at the same time on all platforms except the web player.

Key properties

Loop – Whether an audio or video clip automatically starts over as soon as it finishes playing.

Media – An identifier for the clip that an audio or video control plays.

> **Note**

- Use HTTPS for all external videos to ensure compatibility with modern browsers.
- External videos must be accessible anonymously (without any authentication).

ShowControls – Whether an audio or video player shows, for example, a play button and a volume slider, and a pen control shows, for example, icons for drawing, erasing, and clearing.

Additional properties

AccessibleLabel – Label for screen readers. Should be the title of the video or audio clip.

AutoPause – Whether an audio or video clip automatically pauses if the user navigates to a different screen.

AutoStart – Whether an audio or video control automatically starts to play a clip when the user navigates to the screen that contains that control.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

ClosedCaptionsUrl – Video control only. URL of closed captions file in WebVTT format. Both video and captions URLs must be HTTPS. Server hosting both video and captions file needs to be CORS enabled.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Fill – The background color of a control.

FocusedBorderColor – The color of a control's border when the control is focused.

FocusedBorderThickness – The thickness of a control's border when the control is focused.

Height – The distance between a control's top and bottom edges.

Image – The name of the image that appears in an image, audio, or microphone control.

ImagePosition – The position (**Fill**, **Fit**, **Stretch**, **Tile**, or **Center**) of an image in a screen or a control if it isn't the same size as the image.

OnEnd – Actions to perform when an audio or video clip finishes playing.

OnPause – Actions to perform when the user pauses the clip that an audio or video control is playing.

OnStart – Actions to perform when the user starts to record with a microphone control.

Paused – *True* if a media playback control is currently paused, *false* otherwise.

Reset – Whether a control reverts to its default value.

Start – Whether an audio or video clip plays.

StartTime – The time after the start of an audio or video clip when the clip starts to play.

Time – A media control's current position.

TabIndex – Keyboard navigation order in relation to other controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container).

Related functions

First(*TableName*)

Examples

Play an audio or video file

1. On the **File** menu, click or tap **Media**, click or tap **Videos** or **Audio**, and then click or tap **Browse**.

2. Browse to the file you want to use, click or tap it, and then click or tap **Open**.

3. Press Esc to return to the default workspace, add an **Audio** or **Video** control, and set its **Media** property to the file that you added.

   Don't know how to add and configure a control?

4. Press F5, and then play the clip by clicking or tapping the play button of the control that you added.

   > **Tip**
   >
   > The play button of the **Video** control appears when you hover over the control.

5. Press Esc to return to the default workspace.

Play a YouTube video

1. Add a **Video** control, and set its **Media** property to the URL of the YouTube video, enclosed in double quotation marks.
2. Press F5, and then play the clip by clicking or tapping the play button of the **Video** control.
3. Press Esc to return to the default workspace.

Accessibility guidelines

Audio and video alternatives

- **ShowControls** must be true so that users can listen or watch multimedia at their own pace. This also allows users to toggle closed captions and full-screen mode on video players.
- Closed captions must be provided for videos.
  - For YouTube videos, use authoring tools provided by YouTube to add captions.
  - For other videos, create captions in WebVTT format, upload them, and set **ClosedCaptionsUrl** to the url location. There are several limitations. Server(s) hosting video and captions needs to be CORS-enabled and serve them using HTTPS protocol.
- Consider providing an audio or video transcript using one of these methods:

  1. Put the text in a **Label** and position it adjacent to the multimedia player. Optionally, create a **Button** to toggle the display of the text.

2. Put the text in a different screen. Create a **Button** that navigates to the screen and position the button adjacent to the multimedia player.
3. If the description is short, it can be put it in the **AccessibleLabel**.

Color contrast

There must be adequate color contrast between:

- **FocusedBorderColor** and the outside color
- **Image** and the multimedia player controls (if applicable)
- **Fill** and the multimedia player controls (if the fill is visible)

Provide closed captions and/or transcript if the video content has color contrast issues.

Screen reader support

- **AccessibleLabel** must be present.

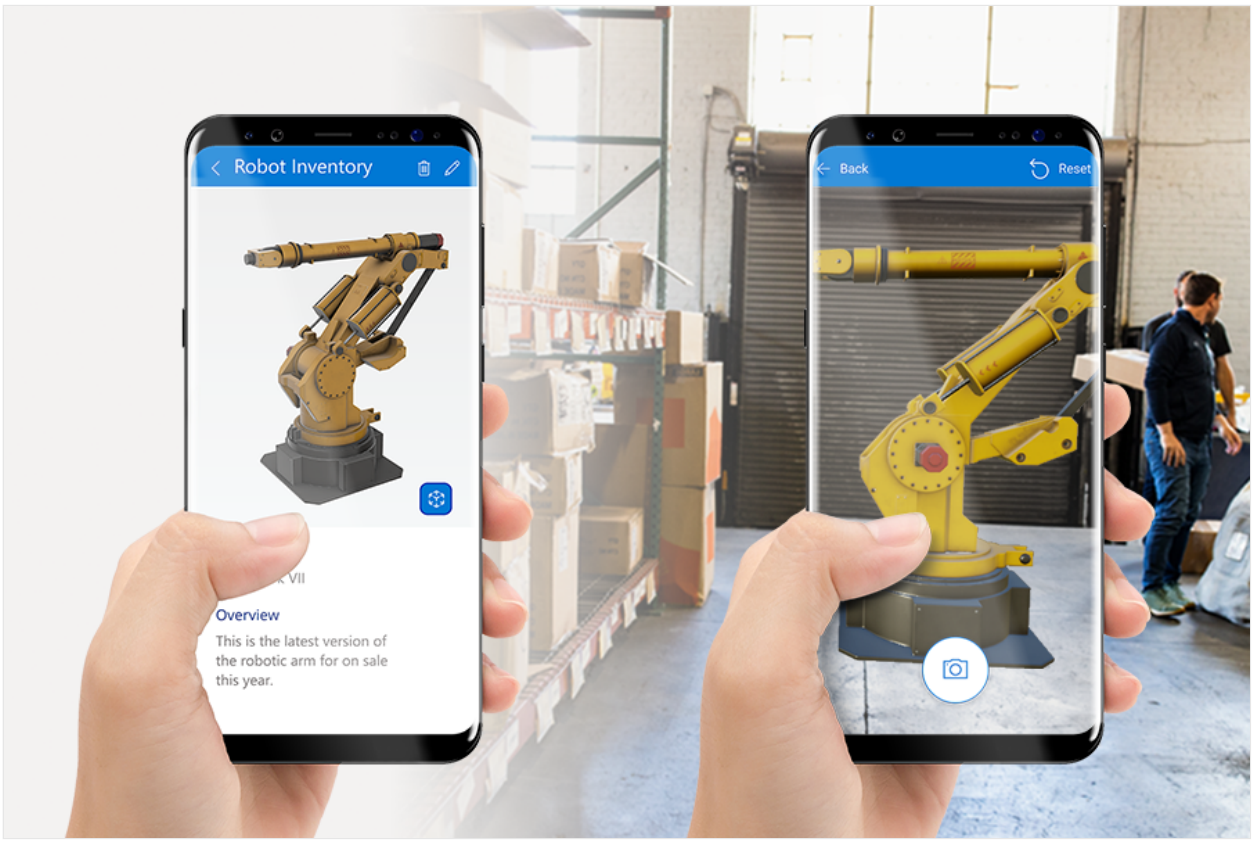
Keyboard support

- **TabIndex** must be zero or greater so that keyboard users can navigate to it.
- Focus indicators must be clearly visible. Use **FocusedBorderColor** and **FocusedBorderThickness** to achieve this.
- **AutoStart** should be false because it can be difficult for keyboard users to stop playback quickly.

View 3D content in Power Apps

Article • 05/09/2022

Easily add interactive 3D content to your canvas apps. You can select from a gallery of 3D content then rotate and zoom into the model with simple gestures.



Important

Your 3D content must be in the .glb, .obj, or .stl file formats. You can **convert your existing 3D models into the .glb file format** from a variety of 3D formats.

Tip

The mixed reality (MR) controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the **Babylon sandbox** should work in Power Apps through this shared MR platform. If your content works in Babylon but not in Power Apps, ask a question in the **Power Apps Community Forum**. (Tag it with "mixed reality.")

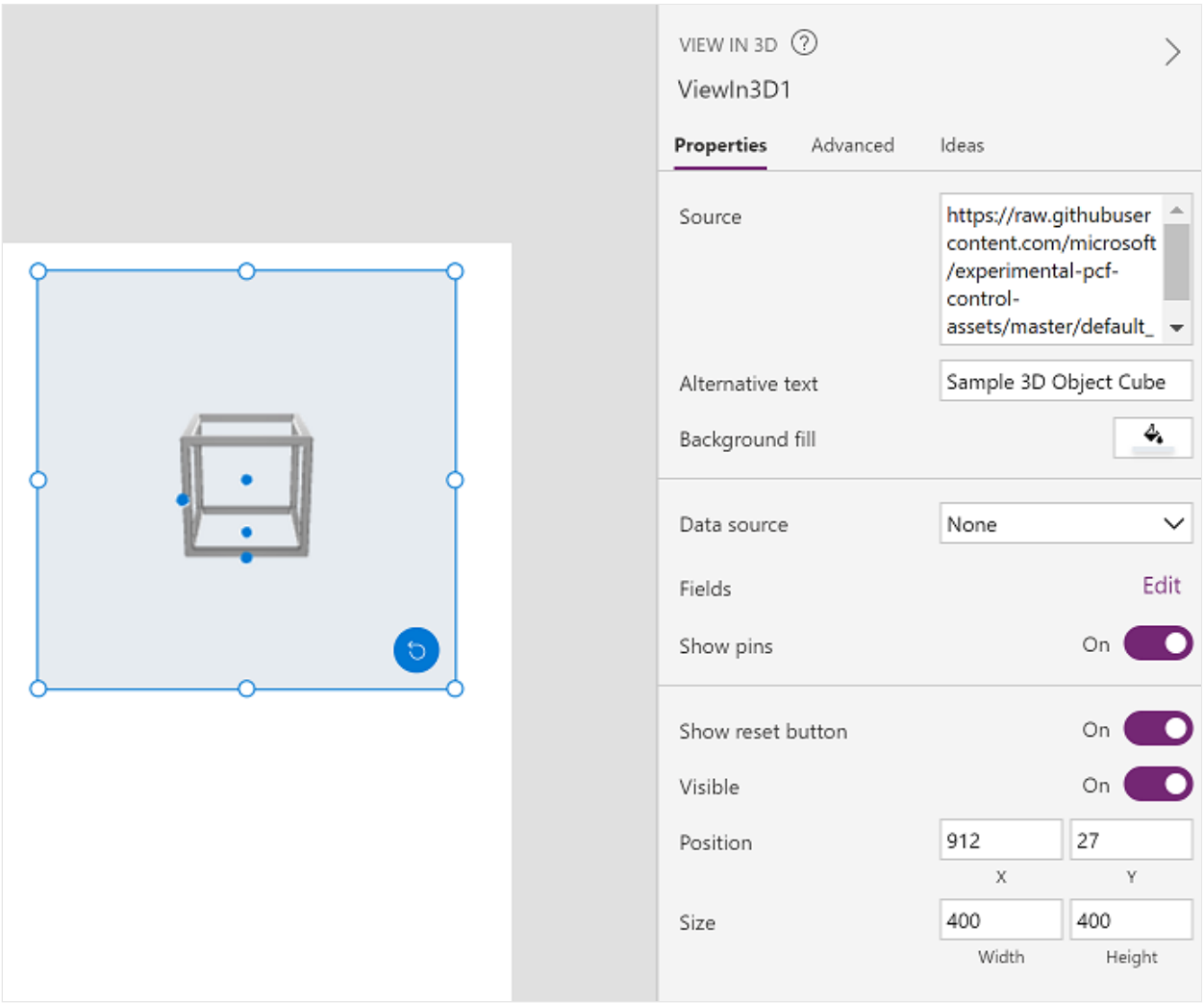
Add 3D object control to an app screen

With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Media**.
2. Select **3D object** to place the control in the app screen, or drag the control to the screen to position it more precisely.

Key properties

Change the 3D content control's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Property	Description	Type	Location
Source	Identifies the object file to display. The **3D object** control supports loading models from various sources. See Loading external 3D models for details.	Not applicable	Properties; Advanced: **Source**
Alternative text	Specifies the text to be displayed if the model can't load or if the user hovers over the model.	String	Properties; Advanced: **AltText**

Property	Description	Type	Location
Background fill	Sets the control's background color.	Color picker	Properties; Advanced: **BackgroundFill** (accepts RGBA or HTML hexadecimal color codes)
Pins(Items)	Shows pins on the model at specific coordinates, provided in a data source (**Items**). If *None*, no pins are shown.	Data table	Properties; Advanced: **Items**
Show pins	Shows the pins described in **Items**.	Boolean	Properties; Advanced: **ShowPins**
MaxPins	Specifies the maximum number of pins that can be shown on the model.	Integer	Advanced
PinsX	Specifies the location of pins on the X axis in 3D coordinate space.	Floating point number	Advanced
PinsY	Specifies the location of pins on the Y axis in 3D coordinate space.	Floating point number	Advanced
PinsZ	Specifies the location of pins on the Z axis in 3D coordinate space.	Floating point number	Advanced
Show reset button	Shows or hides a button that resets the model to its initial state.	Boolean	Properties; Advanced; **ShowReset**
OnModelLoad	Contains behavior formula that runs when a model is loaded.	Event	Advanced
OnChange	Contains behavior formula that runs when any property of the control is changed.	Event	Advanced
OnSelect	Contains behavior formula that runs when the user selects a pin or the control.	Event	Advanced

Additional properties

Property	Description	Type	Location

Property	Description	Type	Location
ContentLanguage	Determines the display language of the control, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the control allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Dropdown list	Advanced
Position	Places the upper-left corner of the control at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the control using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
TabIndex	Specifies the order the control is selected if the user navigates the app using the Tab key.	Integer	Advanced
Tooltip	Determines the text to display when the user hovers over a pin.	String	Advanced
Visible	Shows or hides the control.	Boolean	Properties; Advanced: **Visible**

Performance considerations

We recommend that you use one 3D control on a screen for the best user experience. Multiple instances of the **3D object** control on one screen will try to load their 3D models at the same time, which can severely degrade the performance of your app.

Other mixed reality controls

- View 3D content and images in the real world with the **View in mixed reality** control.
- Measure distance, area, and volume with the **Measure in mixed reality** control.
- Create and view predefined 3D shapes with the **View shape in mixed reality** control.
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

Create an app with 3D and mixed reality controls

View 3D content or images in mixed reality

Article • 07/15/2022

Use the **View in MR** control to place a 3D model of an object or an image in a real-world space. For instance, preview a painting on a wall before you hang it, or see whether a 3D virtual bookcase will fit where you intend it to go.

What if you don't have a 3D model of the object? Load an image of the object and set its dimensions to the object's size. The control applies the image as the face of a virtual cube of the desired size. Place this stand-in for the 3D model in your space.

The **View in MR** control adds a button to your app. When a user selects the button, the app overlays a selected 3D model (in .glb, .stl, or .obj file formats) or image (in .jpg or .png file formats) on the device camera feed.



You can also take photos using the **View in MR** control and upload them to OneDrive.

> **Important**
>
> Your 3D content must be in the .glb, .stl, or .obj file formats. You can convert your existing 3D models into the .glb file format from a variety of 3D formats. 3D content compressed with Draco compression Draco is not supported.

> **Tip**
>
> - Make sure your 3D models are optimized for use with Power Apps to minimize load times.
> - The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

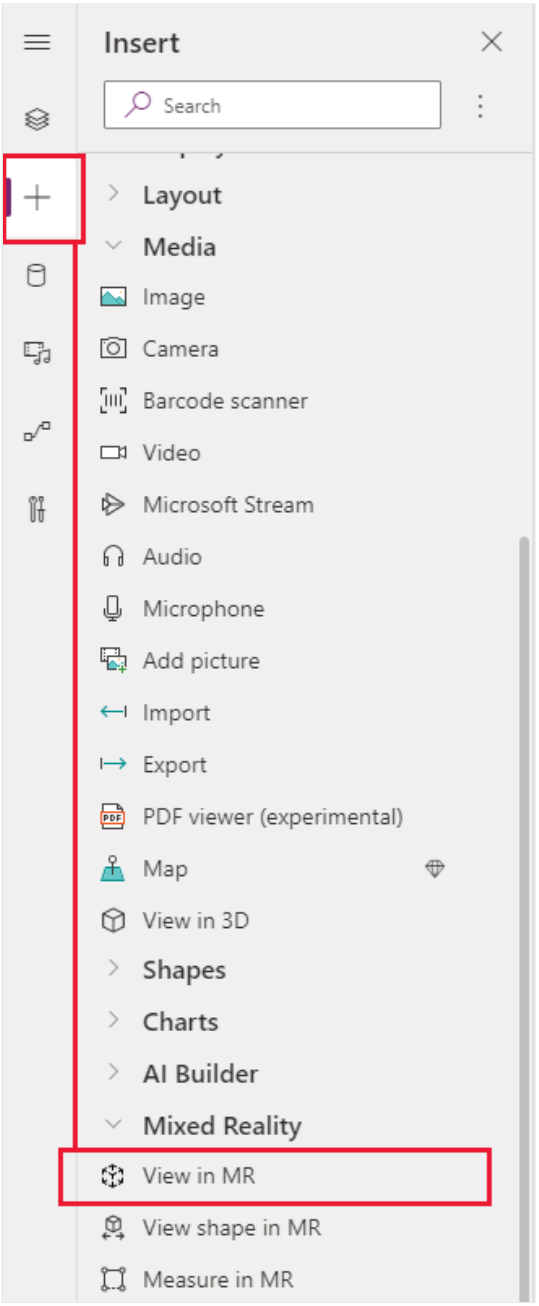
- The MR controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the Babylon sandbox should work in Power Apps through this shared MR platform. If your content works in Babylon but not in Power Apps, ask a question in the Power Apps Community Forum. (Tag it with "mixed reality.")

Add a View in MR button to an app screen

With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Mixed Reality**.
2. Select **View in MR** to place the control in the app screen, or drag the control to the screen to position it more precisely.

The control is a button that's labeled with an icon of a cube and the text *View in MR*. Change the label and hide the icon if you like in the **Text** and **Display type** properties.



View different models in a single session

The View in MR control displays one media file by default. To view different media files selected through a media gallery during the MR experience, use the control's *Items* advanced property. This property creates a table that lists media files the user can choose from in the mixed reality experience.

When you use the *Items* property, you need to set a value for the following additional advanced properties to map the columns of your table:

Expand table

Property	Description
ItemsSource (required)	Specifies which column in *Items* identifies the 3D model or image to show in the mixed reality view. See Connect 3D models to Power Apps to learn more about bringing models into Power Apps.
ItemsLabel	Specifies which column in *Items* contains the label to show in the media gallery. The *Alternative text* property is used for items that don't have a label specified.
ItemsThumbnail	Specifies which column in *Items* identifies the thumbnail to show in the media gallery. A default thumbnail is provided for items that don't have one specified.
ItemsObjectWidth, ItemsObjectHeight, ItemsObjectDepth	Specifies an explicit size for each media item in the table. *Object width*, *Object height*, and *Object depth* are used for items that don't have a specific size.

To test it, let's create a media gallery that shows three different plants in a room setting. Our *Items* table looks like this:

Expand table

label	source	thumbnail
Birds of Paradise	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/bird_of_paradise_blue.glb	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/bird_of_paradise_blue.png
Fiddle Leaf Fig	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/flf_orange.glb	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/flf_orange.png
Small Palm Tree	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/palm_green.glb	https://raw.githubusercontent.com/microsoft/experimental-pcf-control-assets/master/palm_green.png



With your app open for editing in Power Apps Studio:

1. Open the **Insert** tab and expand **Mixed Reality**.
2. Select **View in MR** to place the control in the app screen, or drag the control to the screen to position it more precisely.
3. On the **Advanced** properties tab, select the *Items* property and type or paste the following code:

```
Power Fx

Table(
{
label: "Birds of Paradise",
source: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/bird_of_paradise_blue.glb",
thumbnail: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/bird_of_paradise_blue.png"
},
{
label: "Fiddle Leaf Fig",
source: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
```

```
assets/master/flf_orange.glb",
thumbnail: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/flf_orange.png"
},
{
label: "Small Palm Tree",
source: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/palm_green.glb",
thumbnail: "https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/palm_green.png"
}
)
```

4. Set the *ItemsLabel* property to `"label"`.

5. Set the *ItemsSource* property to `"source"`.

6. Set the *ItemsThumbnail* property to `"thumbnail"`.



> **Note**
>
> The *Items* property is set to your data source which can be a collection, from a connector, or a hardcoded table like in this example. The *ItemsSource*, *ItemsLabel*, and *ItemsThumbnail* properties are set to the column names of that data source within quotation marks.

7. Save and publish the app, then open it on your mobile device to try it out.

How objects are scaled if you resize them

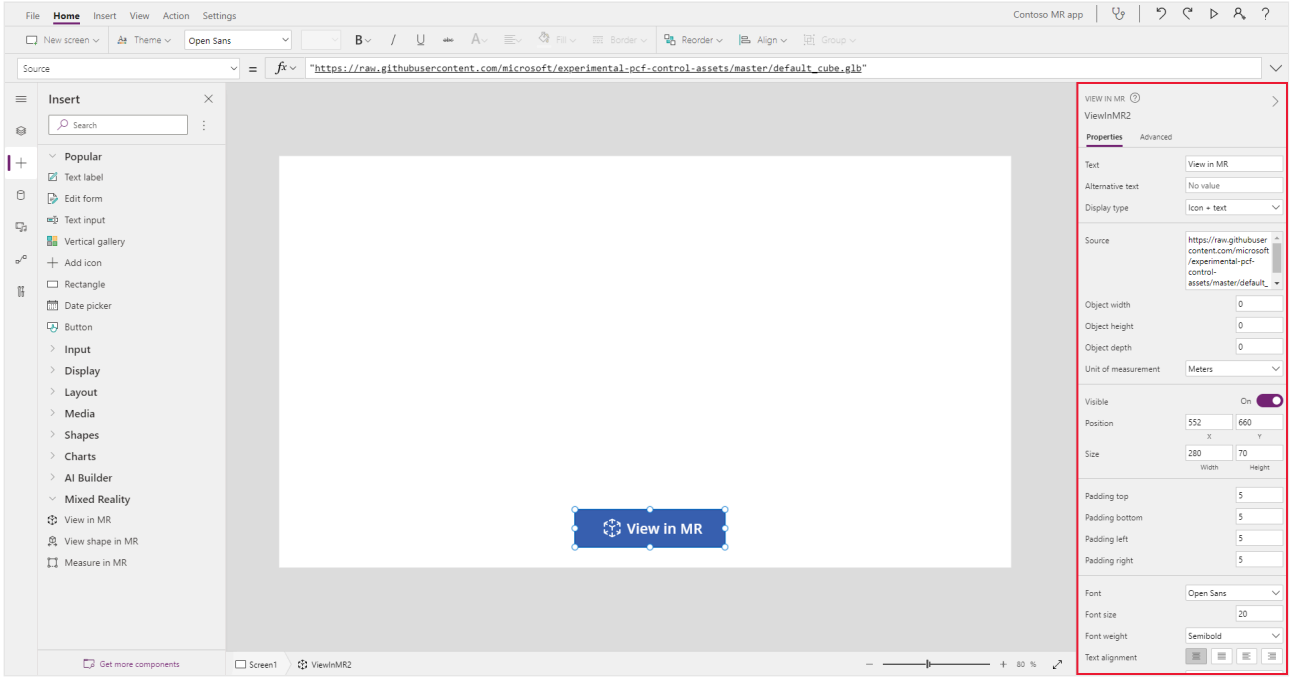
If you don't specify dimensions when you place a model, the **View in MR** control follows the dimensions given in the object file. You can resize the model by entering a value other than 0 in one or more of the control's width, height, and depth properties (**Object width**, **Object height**, and **Object depth**). How the model is scaled depends on how many dimensions you change.

- If all dimensions are *0* (the default), the model's size is left as it's set in the object file.
- If one dimension is set, the model is scaled equally based on the changed dimension, similar to setting "keep aspect ratio" when you resize an image.
- If two dimensions are set, the model scales to the two given dimensions and an average of the first two. For example, say you have a model that is 12 m high, 6 m wide, and 3 m deep. You set the **View in MR** control's properties as follows: **Object height**: **24**, **Object width**: **9**, **Object depth**: **0** (unchanged). The height increases by a factor of 2, and the width increases by a factor of 1.5. The increase in height and width is averaged to find the factor by which the depth should be increased: 2 + 1.5 = 3.5 / 2 = 1.75. The final dimensions of the model are **Object height** 24, **Object width** 9, and **Object depth** 5.25 (3 x 1.75).
- If all three dimensions are set, the model's size is changed to what you specified.

> **Note**
>
> If the measurements don't adhere to the original aspect ratio, the model could be squashed or distorted when it's viewed in MR.

Properties

Change the **View in MR** button's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Expand table

Property	Description	Type	Location
Text	Sets the button label text.	String	Properties; Advanced: **Text**

Property	Description	Type	Location
Alternative text	Specifies the text to be displayed if the model can't load or if the user hovers over the model.	String	Properties; Advanced: **AltText**
Display type	Determines whether the button label shows an icon of a cube, text, or both.	Drop-down selection	Properties; Advanced: **DisplayType**
Source	Identifies the data source (.glb, .stl, or .obj file) to display.	Not applicable	Properties; Advanced: **Source**
Sources List (Items)	Data source (table) that lists multiple media files to view in a single mixed reality session.	Table	Properties; Advanced: **Items**
Object width	Sets the width of the model.	Integer	Properties; Advanced: **ObjectWidth**
Object height	Sets the height of the model.	Integer	Properties; Advanced: **ObjectHeight**
Object depth	Sets the depth of the model.	Integer	Properties; Advanced: **ObjectDepth**
Unit of measurement	Sets the measurement unit used for the model's width, height, and depth.	List	Properties; Advanced: **Units**
Enable shadows	Determines whether shadows are used to enhance the 3D effect when the model is shown.	Boolean	Properties; Advanced: **EnableShadows**
Enable marker detection	Determines whether marker detection will be used to determine model position and rotation. See Use marker detection with mixed reality controls	Boolean	Properties; Advanced: **EnableMarkerDetection**
Visible	Shows or hides the button.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the button at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the button using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Padding top	Sets the distance between the button label text and the top of the button.	Floating point number	Properties; Advanced: **PaddingTop**
Padding bottom	Sets the distance between the button label text and the bottom of the button.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the button label text and the left edge of the button.	Floating point number	Properties; Advanced: **PaddingLeft**
Padding right	Sets the distance between the button label text and the right edge of the button.	Floating point number	Properties; Advanced: **PaddingRight**
Font	Sets the name of the family of fonts used for the button label text.	List	Properties; Advanced: **Font**
Font size	Sets the size of the button label text.	Floating point	Properties; Advanced: **FontSize**

Property	Description	Type	Location
		number	
Font weight	Sets the weight of the button label text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	List	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the label text in the button, either *Center*, *Justify*, *Left*, or *Right*.	Not applicable	Properties; Advanced: **TextAlignment**
Vertical alignment	Sets the vertical alignment of the label text in the button, either *Bottom*, *Middle*, or *Top*.	List	Properties; Advanced: **VerticalAlign**
Font style	Sets the style of the button label text, either *Italic*, *Underline*, *Strikethrough*, or none.	Not applicable	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Border radius	Determines the corner radius of the button border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the button label text and the button background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the button border.	Not applicable	Properties; Advanced: **BorderStyle**, **BorderThickness**, **BorderFillColor**
Disabled	Turns off the button but leaves it visible.	Boolean	Properties; Advanced: **Disabled**
Disabled color	Sets the colors of the button label text, the button background, and the button border if DisplayMode is **Disabled**.	Not applicable	Properties; Advanced: **DisabledContentColor**, **DisabledFillColor**, **DisabledBorderColor**
Pressed color	Sets the colors of the button label text, the button background, and the button border when the user selects the button.	Not applicable	Properties; Advanced: **PressedContentColor**, **PressedFillColor**, **PressedBorderColor**
Hover color	Sets the colors of the button label text, the button background, and the button border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverContentColor**, **HoverFillColor**, **HoverBorderColor**
OnMixedRealitySelect	Behavior that is triggered when user selects the button to launch into the mixed reality experience.	Defined action	Advanced
OnChange	Behavior that is triggered when any property on the button is changed.	Defined action	Advanced
Tooltip	Determines the text to display when the user hovers over the button.	String	Advanced
ContentLanguage	Determines the display language of the button label, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the button allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced
TabIndex	Specifies the order the button is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **TabIndex**

Output properties

Your app can make use of more properties when a user interacts with the **View in MR** control. These are known as output properties. You can use output properties in other controls or to customize the app experience.

Expand table

Property	Description	Type
Photos	Collects the photos captured during the mixed reality session. You can upload the mixed-reality photos to OneDrive and show them in a gallery.	Not applicable

Other mixed reality controls

- View 3D content with the **3D object** control.
- Measure distance, area, and volume with the **Measure in mixed reality** control.
- Create and view predefined 3D shapes with the **View shape in mixed reality** control
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

- Create an app with 3D and mixed reality controls
- Use marker detection with mixed reality controls

Feedback

Was this page helpful? Yes No

Provide product feedback

View predefined 3D shapes in mixed reality

Article • 06/17/2022

Use the **View shape in MR** control in a canvas app to see whether a digital cube of a given size fits in a real-world space. (To compare a 3D model of an object instead, use the View in MR control.)

The **View shape in MR** control adds a button to your app. When a user selects the button, the app overlays a cube on the device camera feed. The cube's dimensions are set as a property on the control when you author the app in Microsoft Power Apps Studio. Users may not change the size of the cube while they use the app.



Tip

The mixed-reality (MR) controls work best in well-lit environments with flat-textured surfaces. Tracking is better on LIDAR-enabled devices.

Tip

The MR controls in Power Apps use Babylon and Babylon React Native. Mixed reality content that works in the **Babylon sandbox** should work in Power Apps

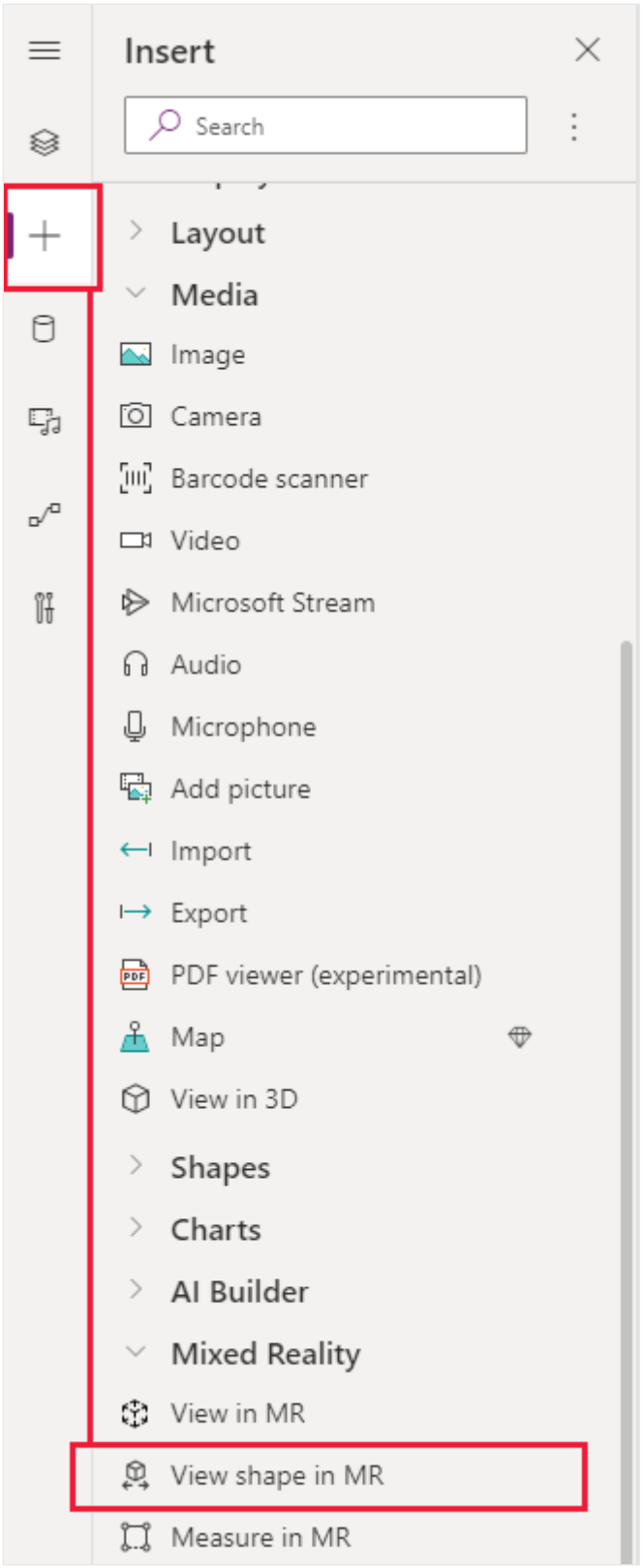
through this shared MR platform. If your content works in Babylon but not in Power Apps, ask a question in the **Power Apps Community Forum**. (Tag it with "mixed reality.")

Add a View shape in MR button to an app screen

With your app open for editing in Power Apps Studio:

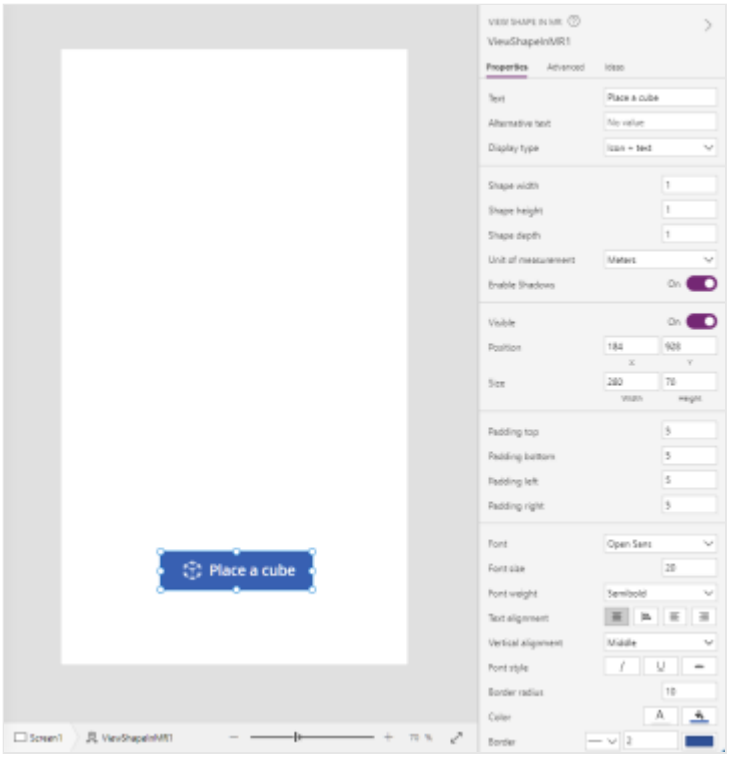
1. Open the **Insert** tab and expand **Mixed Reality**.
2. Select **View shape in MR** to place the control in the app screen, or drag the control to the screen to position it more precisely.

The control is a button that's labeled with an icon of a cube and the text *Place a cube*. Change the label and hide the icon if you like in the **Text** and **Display type** properties.



Properties

Change the **View shape in MR** button's behavior and appearance using properties. Some properties are only available on the **Advanced** tab.



Property	Description	Type	Location
Text	Sets the button label text.	String	Properties; Advanced: **Text**
Alternative text	Specifies the text to be displayed if the model can't load or if the user hovers over the model.	String	Properties; Advanced: **AltText**
Display type	Determines whether the button label shows an icon of a cube, text, or both.	Drop-down selection	Properties; Advanced: **DisplayType**
Shape width	Sets the width of the cube.	Integer	Properties; Advanced: **ShapeWidth**
Shape height	Sets the height of the cube.	Integer	Properties; Advanced: **ShapeHeight**
Shape depth	Sets the depth of the cube.	Integer	Properties; Advanced: **ShapeDepth**
Unit of measurement	Sets the measurement unit used for the cube's width, height, and depth.	Drop-down selection	Properties; Advanced: **Units**
Enable Shadows	Determines whether shadows are used to enhance the 3D effect when the cube is shown.	Boolean	Properties; Advanced: **EnableShadows**

Property	Description	Type	Location
Enable Marker Detection	Determines whether marker detection will be used to to determine model position and rotation. See Use marker detection with mixed reality controls	Boolean	Properties; Advanced: **EnableMarkerDetection**
Visible	Shows or hides the button.	Boolean	Properties; Advanced: **Visible**
Position	Places the upper-left corner of the button at the screen coordinates specified in *x* and *y*.	Floating point number	Properties; Advanced: **X**, **Y**
Size	Determines the size of the button using the pixel values provided in *Width* and *Height*.	Integer	Properties; Advanced: **Width**, **Height**
Padding top	Sets the distance between the button label text and the top of the button.	Floating point number	Properties; Advanced: **PaddingTop**
Padding bottom	Sets the distance between the button label text and the bottom of the button.	Floating point number	Properties; Advanced: **PaddingBottom**
Padding left	Sets the distance between the button label text and the left edge of the button.	Floating point number	Properties; Advanced: **PaddingLeft**
Padding right	Sets the distance between the button label text and the right edge of the button.	Floating point number	Properties; Advanced: **PaddingRight**
Font	Sets the name of the family of fonts used for the button label text.	Dropdown list	Properties; Advanced: **Font**
Font size	Sets the size of the button label text.	Floating point number	Properties; Advanced: **FontSize**
Font weight	Sets the weight of the button label text, either *Bold*, *Lighter*, *Normal*, or *Semibold*.	Dropdown list	Properties; Advanced: **FontWeight**
Text alignment	Sets the horizontal alignment of the label text in the button, either *Center*, *Justify*, *Left*, or *Right*.	Not applicable	Properties; Advanced: **TextAlignment**

Property	Description	Type	Location
Vertical alignment	Sets the vertical alignment of the label text in the button, either *Bottom*, *Middle*, or *Top*.	Dropdown list	Properties; Advanced: VerticalAlign
Font style	Sets the style of the button label text, either *Italic*, *Underline*, *Strikethrough*, or none.	Not applicable	Properties; Advanced: **Italic**, **Underline**, **Strikethrough**
Border radius	Determines the corner radius of the button border.	Floating point number	Properties; Advanced: **BorderRadius**
Color	Sets the colors of the button label text and the button background.	Not applicable	Properties; Advanced: **FillColor**, **TextColor**
Border	Determines the style, width, and color of the button border.	Not applicable	Properties; Advanced: BorderStyle, BorderThickness, BorderFillColor
Disabled	Turns off the button but leaves it visible.	Boolean	Properties; Advanced: **Disabled**
Disabled color	Sets the colors of the button label text, the button background, and the button border if DisplayMode is **Disabled**.	Not applicable	Properties; Advanced: **DisabledContentColor**, **DisabledFillColor**, **DisabledBorderColor**
Pressed color	Sets the colors of the button label text, the button background, and the button border when the user selects the button.	Not applicable	Properties; Advanced: **PressedContentColor**, **PressedFillColor**, **PressedBorderColor**
Hover color	Sets the colors of the button label text, the button background, and the button border when the user hovers the mouse pointer over it.	Not applicable	Properties; Advanced: **HoverContentColor**, **HoverFillColor**, **HoverBorderColor**
OnMixedRealitySelect	Behavior that is triggered when user selects the button to launch into the mixed reality experience.	Defined action	Advanced

Property	Description	Type	Location
OnChange	Behavior that is triggered when any property on the button is changed. This is a common property across multiple controls.	Defined action	Advanced
Tooltip	Determines the text to display when the user hovers over the button.	String	Advanced
ContentLanguage	Determines the display language of the button label, if it's different from the language used in the app.	String	Advanced
DisplayMode	Determines whether the button allows user input (*Edit*), only displays data (*View*), or is disabled (*Disabled*).	Enum	Advanced
TabIndex	Specifies the order the button is selected if the user navigates the app using the Tab key.	Integer	Properties; Advanced: **TabIndex**

Output properties

Your app can make use of more properties when a user interacts with the **View shape in MR** control. These are known as output properties. You can use these properties in other controls or to customize the app experience.

Property	Description	Type
Photos	Collects the photos captured during the mixed reality session. You can upload the mixed-reality photos to OneDrive and show them in a gallery.	Not applicable

Other mixed reality controls

- View 3D content with the **3D object** control.
- View images and 3D content in the real world with the **View in mixed reality** control.
- Measure distance, area, and volume with the **Measure in mixed reality** control.
- Paint 3D lines or draw 3D arrows to specify an area or asset in your environment with the **Markup in MR** control.

See also

- Create an app to view a shape in mixed reality
- Use marker detection with mixed reality controls

Web barcode scanner control (experimental) in Power Apps

Article • 03/13/2023

The legacy barcode scanning control, which is obsolete but might be useful for scanning codes in a web browser.

Description

The control shows the camera feed in the app so that users can scan barcodes on all devices. The control is obsolete because of poor performance, and the mobile Barcode reader control replaces this control.

> ⓘ Note
>
> The web barcode scanner control is only supported on Microsoft Edge, Chrome, Firefox, and Opera browsers. All other browsers will show a warning that some features of the app won't work.

Key properties

BarcodeType - The barcode type to scan. Supported types: Codabar, Code39, Code128, EAN, I2of5, UPC. **Default: UPC**

Camera – On a device that has more than one camera, the numeric ID of the camera that the app uses for scanning barcodes.

Additional properties

AccessibleLabel – Label for screen readers.

BorderColor – The color of a control's border.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

BorderThickness – The thickness of a control's border.

DisplayMode – Whether the control allows user input (**Edit**), only displays data (**View**), or is disabled (**Disabled**).

Height – The distance between a control's top and bottom edges.

ShowLiveBarcodeDetection – Whether visual cues are shown to indicate the status of barcode detection. Yellow rectangles represent areas that are being examined. A green line across a rectangle indicates successful barcode identification.

Stream – Automatically updated image based on the **StreamRate** property.

StreamRate – How often to update the image on the **Stream** property, in milliseconds. This value can range from 100 (1/10th of a second) to 3,600,000 (1 hour).

Text – Value of barcode that was last identified by the scanner.

Tooltip – Explanatory text that appears when the user hovers over a control.

Visible – Whether a control appears or is hidden.

Width – The distance between a control's left and right edges.

X – The distance between the left edge of a control and the left edge of its parent container (screen, if no parent container).

Y – The distance between the top edge of a control and the top edge of the parent container (screen, if no parent container).

Related functions

Patch(*DataSource*, *BaseRecord*, *ChangeRecord*)

Example

Add barcode scanner control

1. Add a **barcode scanner** control, name it "Mybarcode scanner".

   Don't know how to add, name, and configure a control?

2. Add a **Label** control, and set its output to the barcode scanner's **Text** property.

3. Scan a barcode of the type set under **BarcodeType** property.

   The label displays the scanned barcode.

Accessibility guidelines

Video alternatives

- Consider adding a **Label** with its **Text** set to the barcode scanner's **Text**. Since the barcode scanner doesn't display the identified barcode value, doing the above makes the scanner accessible to everyone, not just those with visual disabilities.

Screen reader support

- **AccessibleLabel** must be present.

  > **Note**
  >
  > Screen readers will announce when a new barcode has been found. The value won't be announced. As long as the barcode is in view, screen readers will remind the user every five seconds that the same barcode is still being identified.

Accessibility properties for Power Apps

Article • 07/13/2024

Configuration of properties that aid alternative ways of interacting with controls suitable for users with disabilities.

Properties

AccessibleLabel

Label for screen readers.

An empty value for Image, Icon, and Shape controls will hide the controls from screen reader users.

Live

How screen readers should announce changes to content. Available only in the **Label** control.

- When set to **Off**, the screen reader doesn't announce changes.
- When set to **Polite**, the screen reader finishes speaking before announcing any changes that occurred while the screen reader was speaking.
- When set to **Assertive**, the screen reader interrupts itself to announce any changes that occurred while the screen reader was speaking.

Learn how to announce dynamic changes with live regions.

Role

Intended purpose of a control. Available only in the **Label** control.

This lets screen reader users know whether a **Label** is a heading and allows them to navigate quickly to different parts of the app. There should be exactly one **Heading1** in each screen that serves as the main heading. Use **Heading2** for subheadings. **Heading3** and **Heading4** can be used for finer hierarchies of headings.

Use **Default** for normal text.

AcceptsFocus and TabIndex

Determines if the control participates in keyboard navigation.

You can use **AcceptsFocus** to configure keyboard navigation for modern controls. Classic controls use **TabIndex**.

Expand table

AcceptsFocus	TabIndex	Behavior	Default for
true	0 or greater	Control participates in keyboard navigation, unless it's hidden or disabled.	**Button**, **Text input**, **Combo box**, and other typically interactive controls.
false	−1 or less than 0	Control does not participate in keyboard navigation.	**Label**, **Image**, **Icon**, and other typically non-interactive controls.

Any keyboard navigation sequence can be achieved with just these properties, along with the use of the **Container** control. For **TabIndex**, we recommend using either 0 or -1 for simplicity.

Controls that have a **Visible** property value of *false* or a **DisplayMode** property value of **Disabled** are not included in keyboard navigation.

> **Important**
>
> **TabIndex** only affects keyboard navigation. A logical control order is still necessary for screen reader users to understand app structure. Some screen reader users don't even use keyboards.

See also

- Create accessible apps
- Accessible app structure
- Accessible colors in Power Apps
- Show or hide content from assistive technologies for canvas apps
- Announce dynamic changes with live regions for canvas apps
- Use the Accessibility checker
- Accessibility limitations in canvas apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Color and border properties in Power Apps

Article • 12/16/2022

Overview

Configure the style of a control based on how the user interacts with it.

You can specify colors in many ways:

- Color enumeration: Specify color names from cascading style sheets, as in these examples:
  - **Color.Red**
  - **Color.Indigo**
- ColorValue function: Specify text strings such as color names from cascading style sheets and hex-code notation (#), as in these examples:
  - **ColorValue("AliceBlue")**
  - **ColorValue("#ff00ff")**
- ColorFade function: Specify how faded a color is, from fully black (-100%) to fully white (100%), as in this example:
  - **ColorFade(Color.Red, 50%)**
- RGBA function: Specify the red, green, and blue components of a color from 0 to 255, and specify an alpha channel from 0% (fully transparent) to 100% (fully opaque), as in this example:
  - **RGBA(255, 0, 255, 25%)**

Color properties can also reference other color properties. For example, **Label.PressedColor** may be set to the formula **Label1.Color**, automatically cascading a change from one property to another.

Normal

These properties are in effect normally, when the user is not interacting with the control.

BorderColor – The color of a control's border.

- Applies to Add picture, Audio, Button, Camera, Card, Check box, Column chart, Date Picker, Display form, Drop down, Edit form, Export, Gallery, HTML text,

Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pen input, Pie chart, Radio, Rating, Slider, Text input, Timer, Toggle, and Video controls.

BorderStyle – Whether a control's border is **Solid**, **Dashed**, **Dotted**, or **None**.

- Applies to Add picture, Audio, Button, Camera, Card, Check box, Column chart, Date Picker, Display form, Drop down, Edit form, Export, Gallery, HTML text, Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pen input, Pie chart, Radio, Rating, Slider, Text input, Timer, Toggle, and Video controls.

BorderThickness – The thickness of a control's border.

- Applies to Add picture, Audio, Button, Camera, Card, Check box, Column chart, Date Picker, Display form, Drop down, Edit form, Export, Gallery, HTML text, Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pen input, Pie chart, Radio, Rating, Slider, Text input, Timer, Toggle, and Video controls.

Color – The color of text in a control.

- Applies to Add picture, Button, Check box, Column chart, Date Picker, Drop down, Export, HTML text, Import, Label, Line chart, List Box, Microphone, Pen input, Pie chart, Radio, Text input, and Timer controls.

Drop Shadow - Adds shadow effects around the control.

- Applies to Horizontal container and Vertical container controls.

Fill – The background color of a control.

- Applies to Add picture, Audio, Button, Card, Check box, Date Picker, Display form, Drop down, Edit form, Export, Gallery, HTML text, Icon, Image, Import, Label, List Box, Microphone, PDF viewer, Pen input, Radio, Rating, Screen, Shape, Text input, Timer, Toggle, and Video controls.

Focused

These properties are in effect when the control is focused.

FocusedBorderColor – The color of a control's border when it has focus.

FocusedBorderThickness – The thickness of a control's border when it has focus.

- These properties apply to Add picture, Attachments, Audio, Button, Camera, Check box, Combo box, Date Picker, Drop down, Export, Gallery, Icon, Image,

Import, Label, List Box, Microphone, Radio, Rating, Shape, Slider, Text input, Timer, Toggle, and Video controls.

Disabled

These properties are in effect when the control is disabled. A control can be disabled if the DisplayMode property is set to **Disabled**.

DisabledBorderColor – The color of a control's border if the control's DisplayMode property is set to **Disabled**.

- Applies to Add picture, Button, Check box, Column chart, Date Picker, Drop down, Export, HTML text, Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pie chart, Radio, Slider, Text input, Timer, and Toggle controls.

DisabledColor – The color of text in a control if its DisplayMode property is set to **Disabled**.

- Applies to Add picture, Button, Check box, Date Picker, Drop down, Export, Import, Label, List Box, Microphone, Radio, Text input, and Timer controls.

DisabledFill – The background color of a control if its DisplayMode property is set to **Disabled**.

- Applies to Add picture, Button, Check box, Date Picker, Drop down, Export, HTML text, Image, Import, Label, List Box, Microphone, Radio, Text input, and Timer controls.

Hover

These properties are in effect when the user hovers over the control with a mouse.

HoverBorderColor – The color of a control's border when the user keeps the mouse pointer on that control.

- Applies to Add picture, Button, Check box, Column chart, Drop down, Export, HTML text, Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pie chart, Slider, Text input, Timer, and Toggle controls.

HoverColor – The color of the text in a control when the user keeps the mouse pointer on it.

- Applies to Add picture, Button, Check box, Drop down, Export, Import, Label, List Box, Microphone, Radio, Text input, and Timer controls.

HoverFill – The background color of a control when the user keeps the mouse pointer on it.

- Applies to **Add picture**, **Button**, **Check box**, **Drop down**, **Export**, **Icon**, **Image**, **Import**, **Label**, **List Box**, **Microphone**, **Radio**, **Shape**, **Text input**, and **Timer** controls.

Pressed

These properties are in effect when a button or image control is pressed.

PressedBorderColor – The color of a control's border when the user taps or clicks that control.

- Applies to **Add picture**, **Button**, **Check box**, **Column chart**, **Drop down**, **Export**, **Icon**, **Image**, **Import**, **Label**, **Line chart**, **List Box**, **Microphone**, **PDF viewer**, **Pie chart**, **Shape**, **Slider**, **Text input**, **Timer**, and **Toggle** controls.

PressedColor – The color of text in a control when the user taps or clicks that control.

- Applies to **Add picture**, **Button**, **Check box**, **Drop down**, **Export**, **Import**, **Label**, **List Box**, **Microphone**, **Radio**, **Text input**, and **Timer** controls.

PressedFill – The background color of a control when the user taps or clicks that control.

- Applies to **Add picture**, **Button**, **Check box**, **Drop down**, **Export**, **Image**, **Import**, **Label**, **List Box**, **Microphone**, **Radio**, **Text input**, and **Timer** controls.

Selection

These properties are in effect when the user selects an item in a control.

SelectionColor – The text color of a selected item or items in a list or the color of the selection tool in a pen control.

- Applies to **Drop down**, **List Box**, and **Pen input** controls.

SelectionFill – The background color of a selected item or items in a list or a selected area of a pen control.

- Applies to **Drop down** and **List Box** controls.

Order of color

A control can be in multiple states. For example: focused and hovered. Only one color is used, in this order:

1. Disabled
2. Pressed
3. Hover
4. Focus
5. Normal

Core properties in Power Apps

07/25/2025

Configure whether the user can see and interact with a control.

Properties

Default – The initial value of a control before it is changed by the user.

- Applies to Card, Check box, Drop down, Gallery, List Box, Radio, Rating, Rich Text Editor, Slider, Text input, and Toggle controls.

DelayOutput – Set to true to delay action during text input.

- Applies to Text input, Card

DisplayMode – Values can be **Edit, View,** or **Disabled**. Configures whether the control allows user input (**Edit**), only displays data (**View**) or is disabled (**Disabled**). In **View** mode, input controls such as Text input, Drop down, Date Picker will only display the text value and will not render any interactive elements or decorations. This makes them suitable to be displayed in Forms or as readable output.

- Applies to Add picture, Audio, Button, Camera, Check box, Column chart, Combo box, Date Picker, Drop down, Export, Gallery, HTML text, Icon, Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pen input, Pie chart, Radio, Rating, Rich text editor, Shape, Slider, Text input, Timer, Toggle, and Video controls.

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

- Applies to Column chart, Combo box, Drop down, Gallery, Line chart, List Box, Pie chart, and Radio controls.

OnChange – Actions to perform when the user changes the value of a control (for example, by adjusting a slider).

- Applies to Add picture, Combo box, Drop down, List Box, Radio, Rating, Rich text editor, Slider, Text input, and Toggle controls.

OnSelect – Actions to perform when the user taps or clicks a control.

- Applies to Add picture, Button, Camera, Check box, Column chart, Combo box, Date Picker, Drop down, Export, HTML text, Icon, Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pen input, Pie chart, Radio, Rating, Shape, Slider, Text input, Timer, and Toggle controls.

Reset – Whether a control reverts to its default value. Also see the Reset function.

- Applies to Audio, Check box, Combo box, Drop down, List Box, Microphone, Radio, Rating, Slider, Text input, Timer, Toggle, and Video controls.

Text – Text that appears on a control or that the user types into a control.

- Applies to Add picture, Button, Check box, Export, Import, Label, Text input, and Timer controls.

Tooltip – Explanatory text that appears when the user hovers over a control.

- Applies to Audio, Button, Camera, Check box, Combo box, Drop down, HTML text, Image, Label, List Box, Microphone, PDF viewer, Pen input, Radio, Rating, Slider, Text input, Timer, Toggle, and Video controls.

Value – The value of an input control.

- Applies to Check box, Radio, Slider, and Toggle controls.

Visible – Whether a control appears or is hidden.

- Applies to Add picture, Audio, Button, Camera, Card, Check box, Column chart, Combo box, Date Picker, Display form, Drop down, Edit form, Export, Gallery, HTML text, Icon, Image, Import, Label, Line chart, List Box, Microphone, PDF viewer, Pen input, Pie chart, Radio, Rating, Rich text editor, Shape, Slider, Text input, Timer, Toggle, and Video controls.

Image properties in Power Apps

Article • 12/16/2022

Configure graphical elements in your app, including images, photos, and elements of a pen control.

BackgroundImage – The name of an image file that appears in the background of a screen.

- Applies to the **Screen** control.

Image – The name of the image that appears in an image, audio, or microphone control.

- Applies to **Audio**, **Image**, **Microphone**, and **Video** controls.

ImagePosition – The position (**Fill**, **Fit**, **Stretch**, **Tile**, or **Center**) of an image in a screen or a control if it isn't the same size as the image.

- Fill - Fills the entire specified size with the image. Scales the image proportionally. Crops the image as needed.
- Fit - Fits the entire image into the specified size. Scales the image proportionally. Doesn't crop the image.
- Stretch - Fills the entire specified size with the image. Stretches the image disproportionately as needed. Doesn't crop the image.
- Tile - Repeats the image across the control into the specified size. Doesn't scale the image. Crops the image as needed.
- Center - Places the image in the center of the control. Doesn't scale the image. Crops the image as needed.

- Applies to **Audio**, **Image**, **Microphone**, **Screen**, and **Video** controls.

Size and location properties in Power Apps

Article • 05/25/2023

Overview

Configure how big a control (or an element of a control) is and where it is in relation to the screen it's on.

Position

X – The distance between the left edge of a control and the left edge of its parent container (screen if no parent container). For a **Card** control in a container that has multiple columns, this property determines the column in which the card appears.

- Applies to **Add picture**, **Audio**, **Button**, **Camera**, **Card**, **Check box**, **Column chart**, **Date Picker**, **Display form**, **Drop down**, **Edit form**, **Export**, **Gallery**, **HTML text**, **Icon**, **Image**, **Import**, **Label**, **Line chart**, **List Box**, **Microphone**, **PDF viewer**, **Pen input**, **Pie chart**, **Radio**, **Rating**, **Shape**, **Slider**, **Text input**, **Timer**, **Toggle**, and **Video** controls.

Y – The distance between the top edge of a control and the top edge of the parent container (screen if no parent container). For a **Card** control in a container that has multiple rows, this property determines the row in which the card appears.

- Applies to **Add picture**, **Audio**, **Button**, **Camera**, **Card**, **Check box**, **Column chart**, **Date Picker**, **Display form**, **Drop down**, **Edit form**, **Export**, **Gallery**, **HTML text**, **Icon**, **Image**, **Import**, **Label**, **Line chart**, **List Box**, **Microphone**, **PDF viewer**, **Pen input**, **Pie chart**, **Radio**, **Rating**, **Shape**, **Slider**, **Text input**, **Timer**, **Toggle**, and **Video** controls.

Size

Height – The distance between a control's top and bottom edges.

- Applies to **Add picture**, **Audio**, **Button**, **Camera**, **Card**, **Check box**, **Column chart**, **Date Picker**, **Display form**, **Drop down**, **Edit form**, **Export**, **Gallery**, **HTML text**, **Icon**, **Image**, **Import**, **Label**, **Line chart**, **List Box**, **Microphone**, **PDF viewer**, **Pen**

input, Pie chart, Radio, Rating, Shape, Slider, Text input, Timer, Toggle, and Video controls.

AutoHeight – Whether a control automatically changes its height to fit the size of its contents, given a fixed width.

Not supported in Horizontal containers because the control's width is not fixed. In Vertical containers, the **Fill portions** and **Minimum width** properties take precedence over **Autoheight**.

- Applies to Label and HTML text

Width – The distance between a control's left and right edges.

- Applies to Add picture, Audio, Button, Camera, Card, Check box, Column chart, Date Picker, Display form, Drop down, Edit form, Export, Gallery, HTML text, Icon, Image, Label, Import, Line chart, List Box, Microphone, PDF viewer, Pen input, Pie chart, Radio, Rating, Shape, Slider, Text input, Timer, Toggle, and Video controls.

WidthFit – Whether a control automatically grows horizontally to fill any empty space of a container control such as an Edit form control. If multiple cards have this property set to **true**, the space is divided between them. For more information, see Understand data form layout.

- Applies to Card

Padding

Padding – The distance between the text on an import or export button and the edges of that button.

- Applies to Add picture, Export, and Import controls.

PaddingBottom – The distance between text in a control and the bottom edge of that control.

- Applies to Button, Check box, Column chart, Date Picker, Drop down, HTML text, Image, Label, Line chart, List Box, PDF viewer, Radio, and Text input controls.

PaddingLeft – The distance between text in a control and the left edge of that control.

- Applies to Button, Check box, Column chart, Date Picker, Drop down, HTML text, Image, Label, Line chart, List Box, PDF viewer, Radio, and Text input controls.

PaddingRight – The distance between text in a control and the right edge of that control.

- Applies to **Button**, **Check box**, **Column chart**, **Date Picker**, **Drop down**, **HTML text**, **Image**, **Label**, **Line chart**, **List Box**, **PDF viewer**, **Radio**, and **Text input** controls.

PaddingTop – The distance between text in a control and the top edge of that control.

- Applies to **Button**, **Check box**, **Column chart**, **Date Picker**, **Drop down**, **HTML text**, **Image**, **Label**, **Line chart**, **List Box**, **PDF viewer**, **Radio**, and **Text input** controls.

Radius

RadiusBottomLeft – The degree to which the bottom-left corner of a control is rounded.

- Applies to **Button**, **Export**, **Image**, **Import**, and **Text input** controls.

RadiusBottomRight – The degree to which the bottom-right corner of a control is rounded.

- Applies to **Button**, **Export**, **Image**, **Import**, and **Text input** controls.

RadiusTopLeft – The degree to which the top-left corner of a control is rounded.

- Applies to **Button**, **Export**, **Image**, **Import**, and **Text input** controls.

RadiusTopRight – The degree to which the top-right corner of a control is rounded.

- Applies to **Button**, **Export**, **Image**, **Import**, and **Text input** controls.

Text properties in Power Apps

Article • 12/16/2022

Configure the text that appears on a control, as a hint when the user types data, and specify other text-related characteristics.

Text appearance

Font – The name of the family of fonts in which text appears.

- Applies to **Add picture**, **Button**, **Check box**, **Column chart**, **Date Picker**, **Drop down**, **Export**, **HTML text**, **Import**, **Label**, **Line chart**, **List Box**, **Pie chart**, **Radio**, **Text input**, and **Timer** controls.

FontWeight – The weight of the text in a control: **Bold**, **Semibold**, **Normal**, or **Lighter**.

- Applies to **Add picture**, **Button**, **Check box**, **Date Picker**, **Drop down**, **Export**, **Import**, **Label**, **List Box**, **Radio**, **Text input**, and **Timer** controls.

Italic – Whether the text in a control is italic.

- Applies to **Add picture**, **Button**, **Check box**, **Date Picker**, **Drop down**, **Export**, **Import**, **Label**, **List Box**, **Radio**, **Text input**, and **Timer** controls.

Size – The font size of the text that appears on a control.

- Applies to **Add picture**, **Button**, **Check box**, **Column chart**, **Date Picker**, **Drop down**, **Export**, **HTML text**, **Import**, **Label**, **Line chart**, **List Box**, **Pen input**, **Pie chart**, **Radio**, **Text input**, and **Timer** controls.

Strikethrough – Whether a line appears through the text that appears on a control.

- Applies to **Add picture**, **Button**, **Check box**, **Drop down**, **Export**, **Import**, **Label**, **List Box**, **Radio**, **Text input**, and **Timer** controls.

Underline – Whether a line appears under the text that appears on a control.

- Applies to **Add picture**, **Button**, **Check box**, **Drop down**, **Export**, **Import**, **Label**, **List Box**, **Radio**, **Text input**, and **Timer** controls.

Text placement

Align – The location of text in relation to the horizontal center of its control.

- Applies to **Add picture**, **Button**, **Export**, **Import**, **Label**, **Radio**, **Text input**, and **Timer** controls.

LineHeight – The distance between, for example, lines of text or items in a list.

- Applies to **List Box**, **Label**, **Radio**, and **Text input** controls.

VerticalAlign – The location of text on a control in relation to the vertical center of that control.

- Applies to **Add picture**, **Button**, **Check box**, **Export**, **Import**, and **Label** controls.

Building responsive canvas apps

Article • 12/16/2022

Responsiveness refers to the ability of an app to automatically align to different screen sizes and form factors to use the available screen space sensibly, providing great UI and UX in every device, form factor, and screen size.

Responsiveness allows different elements of the app to specify how they:

- Stretch or resize with the screen size changes.
- Maintain or change positions with the screen size changes.

Why should you build responsive apps

End users can access your app from different devices such as phones, tablets, laptops, and desktops with large monitors, different screen sizes, and with varied number of pixels.

To ensure great user experience and usability of the app on each form factor and device, designing the app with responsive design principles is necessary. Even if the app is intended to be used only on the web browser or on mobile phones, user device screen sizes can be different, so designing the app with responsive principles is a good idea.

Designing the app with responsiveness principles

Before you start designing the UI for your app, you need to consider the following aspects:

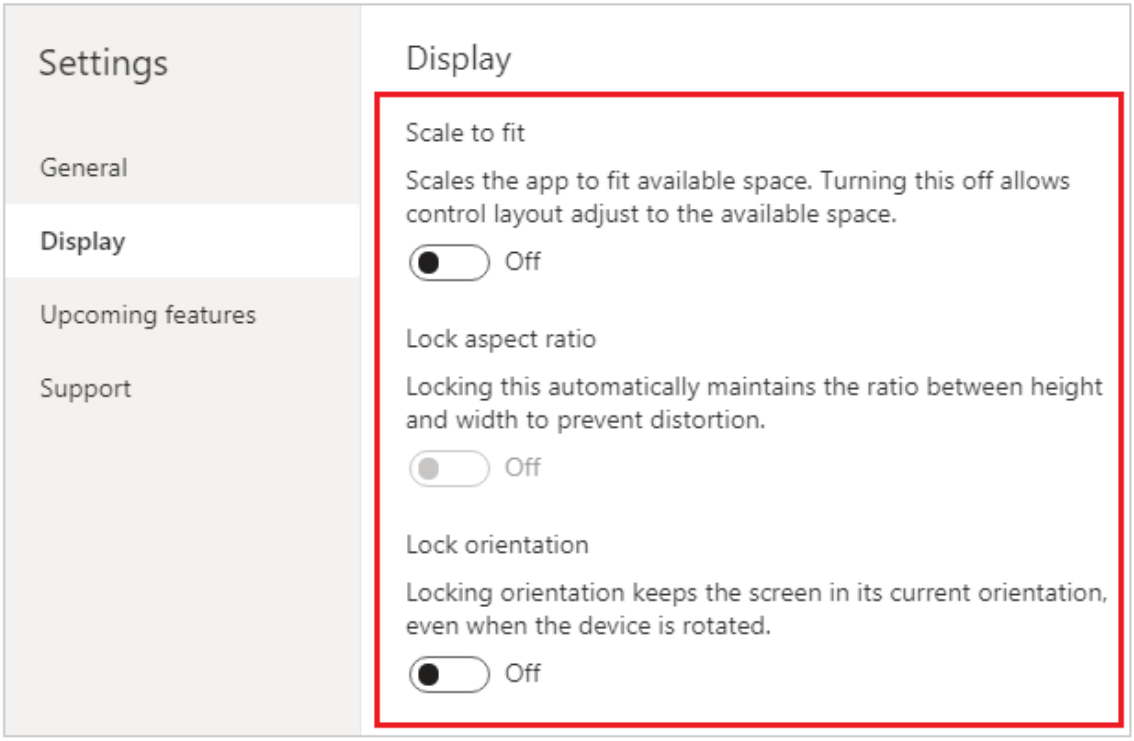
- What form factors or devices do you want to support.
- How the app should look on each form factor?
- What elements of the app need to stretch or resize?
- Are the elements hidden on some form factors?
- Does the app behave different in some form factors?

Once all these requirements are gathered, you should start thinking about how these different UI layouts can be created in a single application with the responsive tools

available in Power Apps.

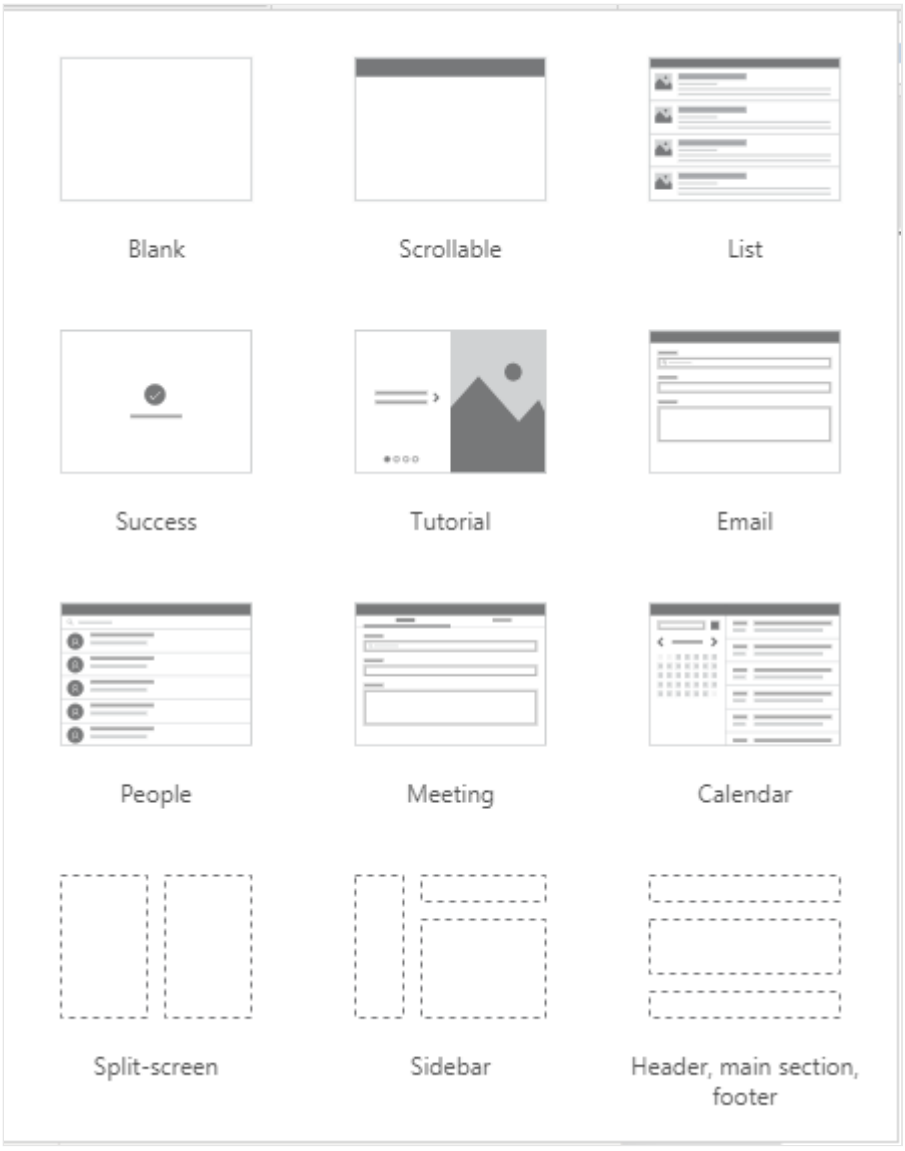
Before you start using the responsive layouts, you need to do the following:

1. Go to Power Apps.
2. Open the app where you want to use the responsive layout.
3. Go to **Settings** > **Display** to disable **Scale to fit**, **Lock aspect ratio**, and **Lock orientation** and select **Apply**.



Responsive layouts

The following responsive layouts can be created by adding a new screen and selecting the appropriate option from the **Layout** tab:



> ⓘ **Note**
>
> The new responsive layouts are available to the apps formats but the new screen templates are available only for **Tablet** format.

Split-screen

The Split-screen layout has two sections, each occupying 50% width of the screen on desktops. On mobile devices, the sections are placed one below the other, each occupying up the entire screen width.

Sidebar

The Sidebar layout has a fixed width sidebar on the left. The main body consists of a fixed height header, and the main section takes up the rest of the screen width. By

default, the template has the same behavior on mobile, however some customizations are recommended based on the UI pattern desired for the mobile experience.

Working with containers

Containers are building blocks of all the responsive design. A container can be an auto-layout container in vertical or horizontal direction, or a fixed-layout container, which in the future will support constraints.

Below are some tips for building your app's UI with containers:

- Always create UI elements that form a UI table inside a container.
- Allows the container to have its own responsive properties and settings to specify how it is positioned or resized on different screen sizes.
- Allows you to change how its child components are laid out with respect to responsiveness.

Choose one of two layout modes for a container: Manual layout or auto-layout (Horizontal or Vertical)

Auto-layout containers

Two controls, Horizontal container and Vertical container can be used to automatically lay out the child components. These containers determine the position of the child components so that you never have to set X, Y for a component inside the container. Also, it can distribute the available space to its child components based on the settings, as well as determines both the vertical and horizontal alignment of the child components.

When to use auto-layout containers

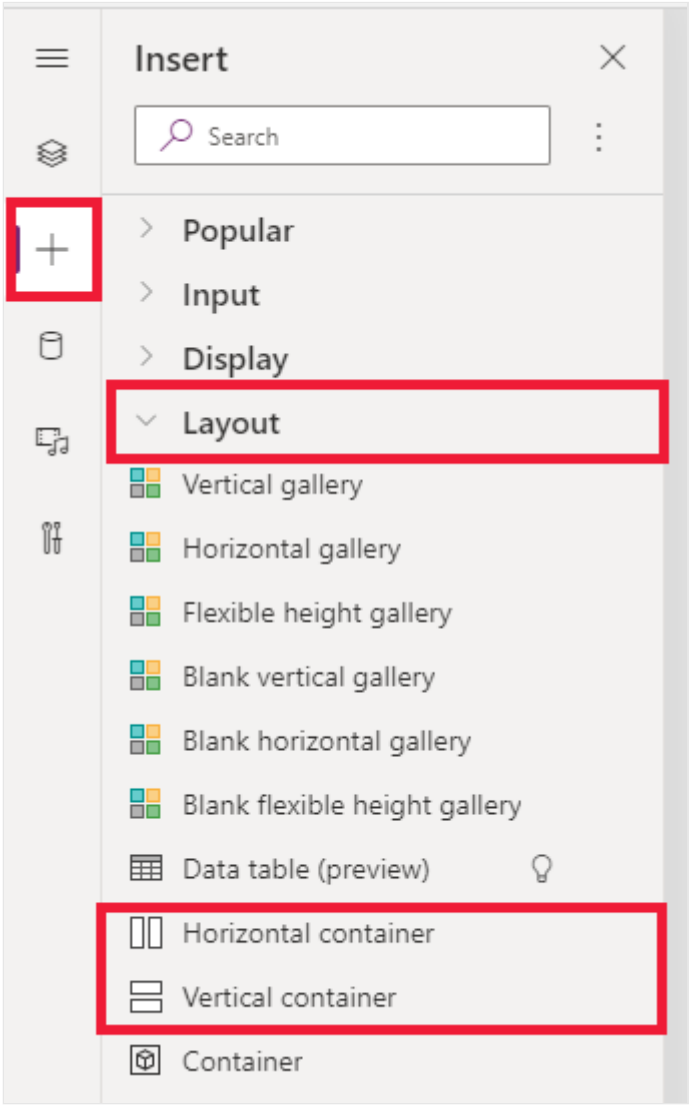
You can use auto-layout containers in the following scenarios:

- UI needs to be responsive to screen size or form factor changes.
- There is more than one child component that needs to resize or move around based on the screen size or form factor changes.
- When you need to stack items vertically or horizontally (regardless of their size).
- When you need to space items evenly on the screen.

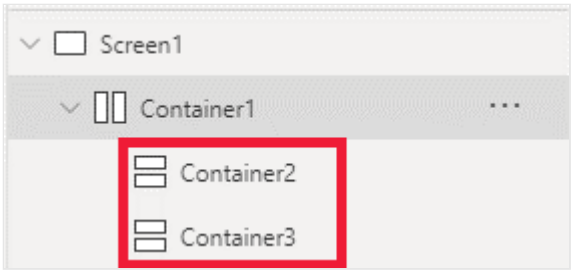
Example for auto-layout container

To build a responsive screen:

1. Create a blank canvas app with **Tablet** layout.
2. Select **Settings** > **Display** and disable **Scale to fit**, **Lock aspect ratio**, and **Lock orientation** and select **Apply**.
3. Now from the **Insert** panes in the left sidebar, under **Layout** tab, select **Horizontal container**.



4. The top-most container needs to be sized to take the entire space with these properties. It will be the same size as the screen as it resizes.
   a. X = 0
   b. Y= 0
   c. Width = Parent.Width
   d. Height = Parent. Height
5. Now, from the **Insert** pane add two **Vertical containers** into the Horizontal container.



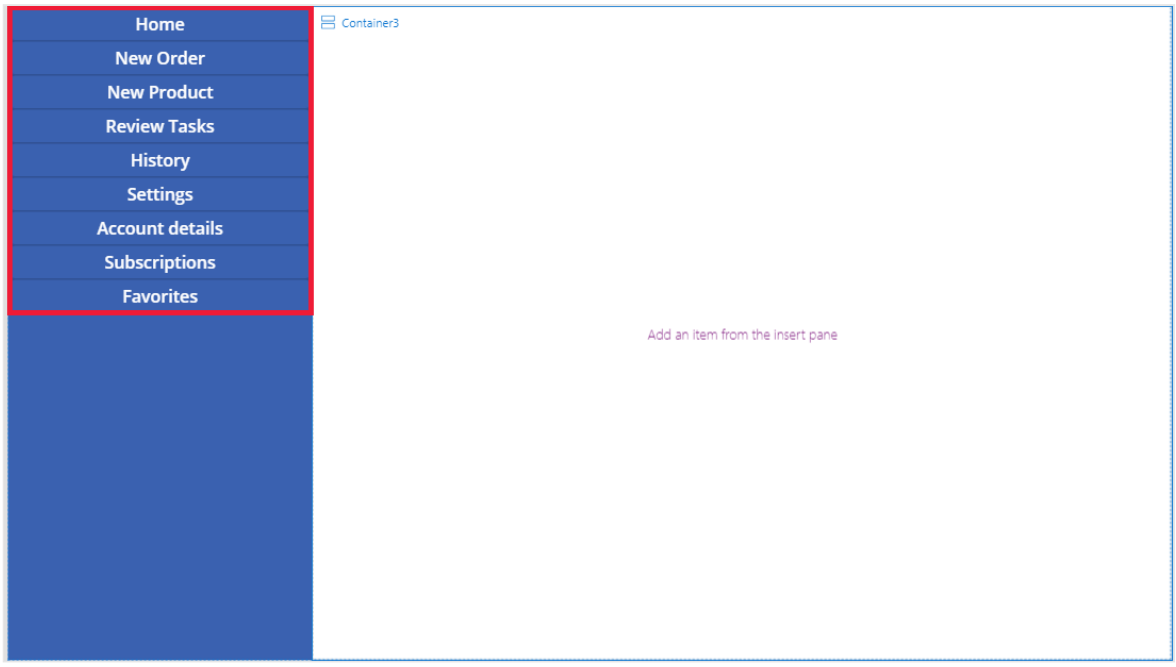
6. To make the containers, fill the full vertical space, Select **Container1** and set the `Align (vertical)` property to `Stretch`.



7. To divide the screen between the content, use the `Fill portions` property on the two child containers. The left container will take up 1/4 of the screen space.
   a. Select **Container2**. You'll see that the `Flexible width` property is turned on. Set the `Fill portions` to 1.
   b. Select **Container3**. You'll see that the `Flexible width` property is turned on. Set the `Fill portions` to 3.

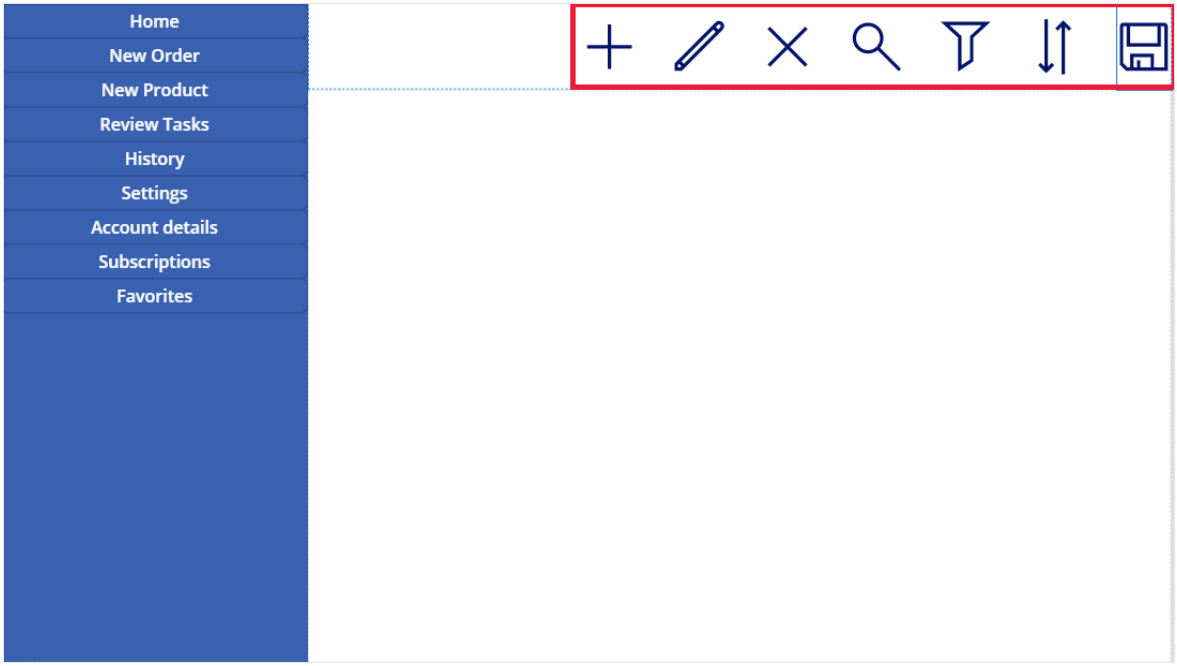
8. Select **Container2**. Set the `Fill = RGBA(56, 96, 178, 1)` property in the formula bar. Set `Align (horizontal)` to `Stretch`.

9. Add few buttons to create a menu. Rename the buttons as needed.



10. Select **Container3** and add a Horizontal container and turn off the `Flexible height` property. Set `Height` to 100. Set `Align (vertical)` to `Stretch`.

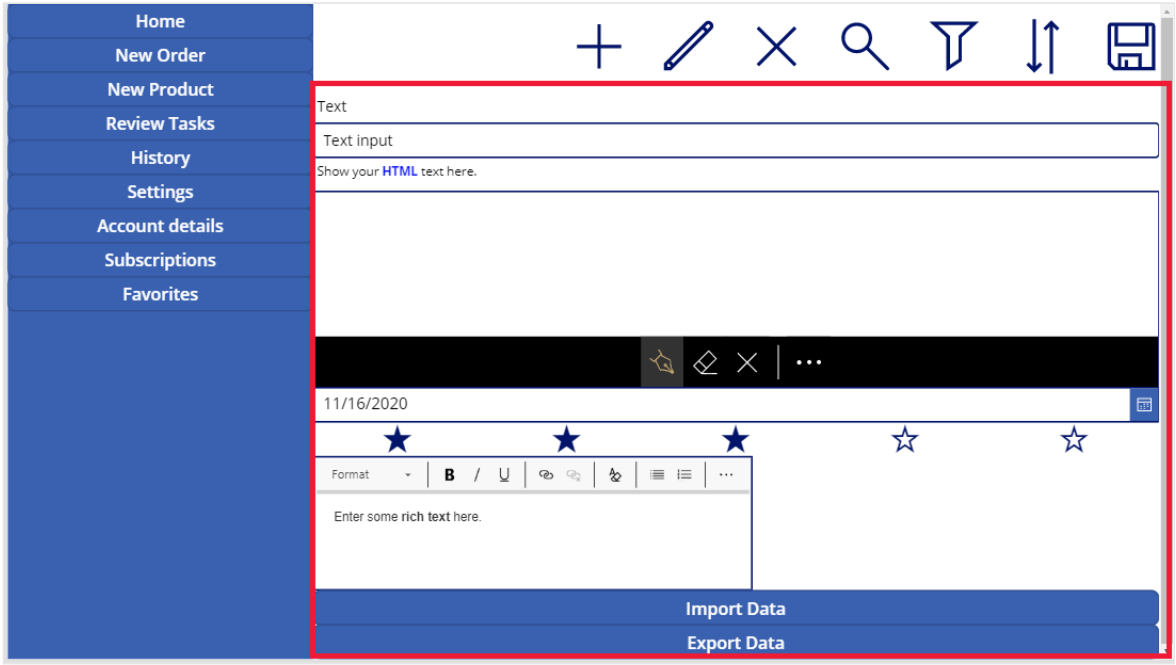
11. Add some **Icons** into **Container4**. Change the icons to be different with the icon property.



12. Select **Container4**. Set the `Justify (horizontal)` property to `End`. Set the `Align (vertical)` to `Center`. Set `Gap` to 40 to create space between the icons.

13. Select **Container3**. Set `Align (horizontal)` to `Stretch`. Set `Vertical Overflow` to `Scroll`.

14. Add different **Label**, **Inputs**, **Media** until they extend beyond the container. Change the color of the rectangles to be different. You'll see a scroll bar to access the unseen content.



15. Select Preview or F5. Change the size of the browser window to see how your app adjusts to different screen sizes.

Known issues

- Certain combinations of layout container properties are incompatible or can produce undesirable outputs, for example:
  - If the container's `Wrap` property is enabled, the `Align` property setting is ignored on child controls.
  - If the container's `Wrap` property is disabled and the container's primary-axis overflow is set to **Scroll** (Horizontal Overflow for Horizontal containers or Vertical Overflow for Vertical containers), it is recommended to set the `Justify` property to either **Start** or **Space Between**.
  - The **Center** or **End** options can cause child controls inaccessible when the container is too small to display all the controls, even though the `Overflow` property is set to **Scroll**.

- You can't resize or reposition the controls on the canvas app because the drag & drop controls are disabled in the layout containers. Instead, use the layout container properties to achieve the desired size and positioning. Control ordering can be changed via the **Tree view**, or by using the arrow keys as shortcuts.

- The **Data table**, **Charts**, and **Add Picture** controls are currently not supported in the layout containers.

- Some of the layout containers properties are hidden for child controls. The hidden properties can still be accessible via the formula bar or from the advanced panel. However, these properties will be ignored even if they are set in these places.

- When controls are moved into a layout container (for example, when copy or pasting the controls), they are inserted into the container by their order in the **Tree view**.

See also

Create responsive layouts

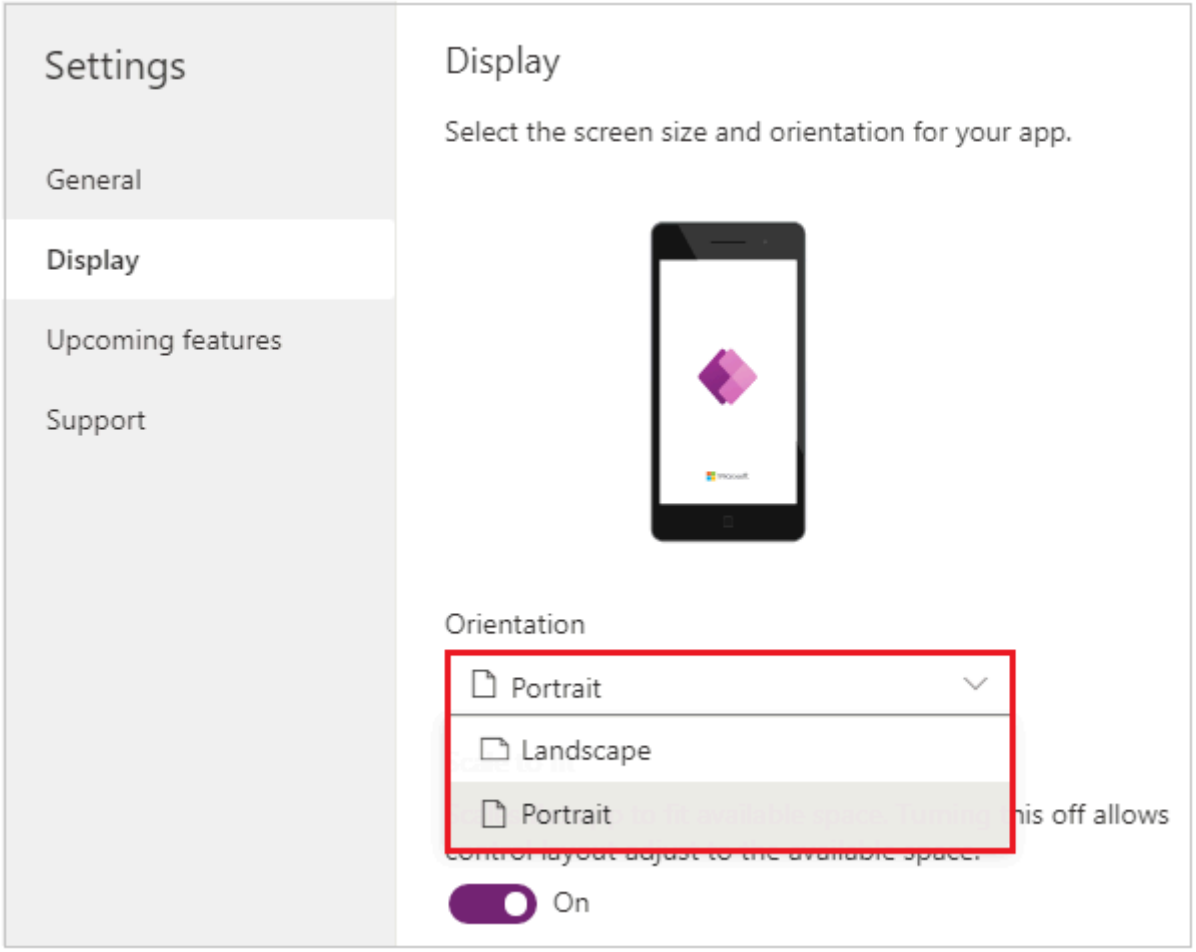
Change screen size and orientation of canvas apps

Article • 05/21/2025

Customize a canvas app by changing its screen size and orientation.

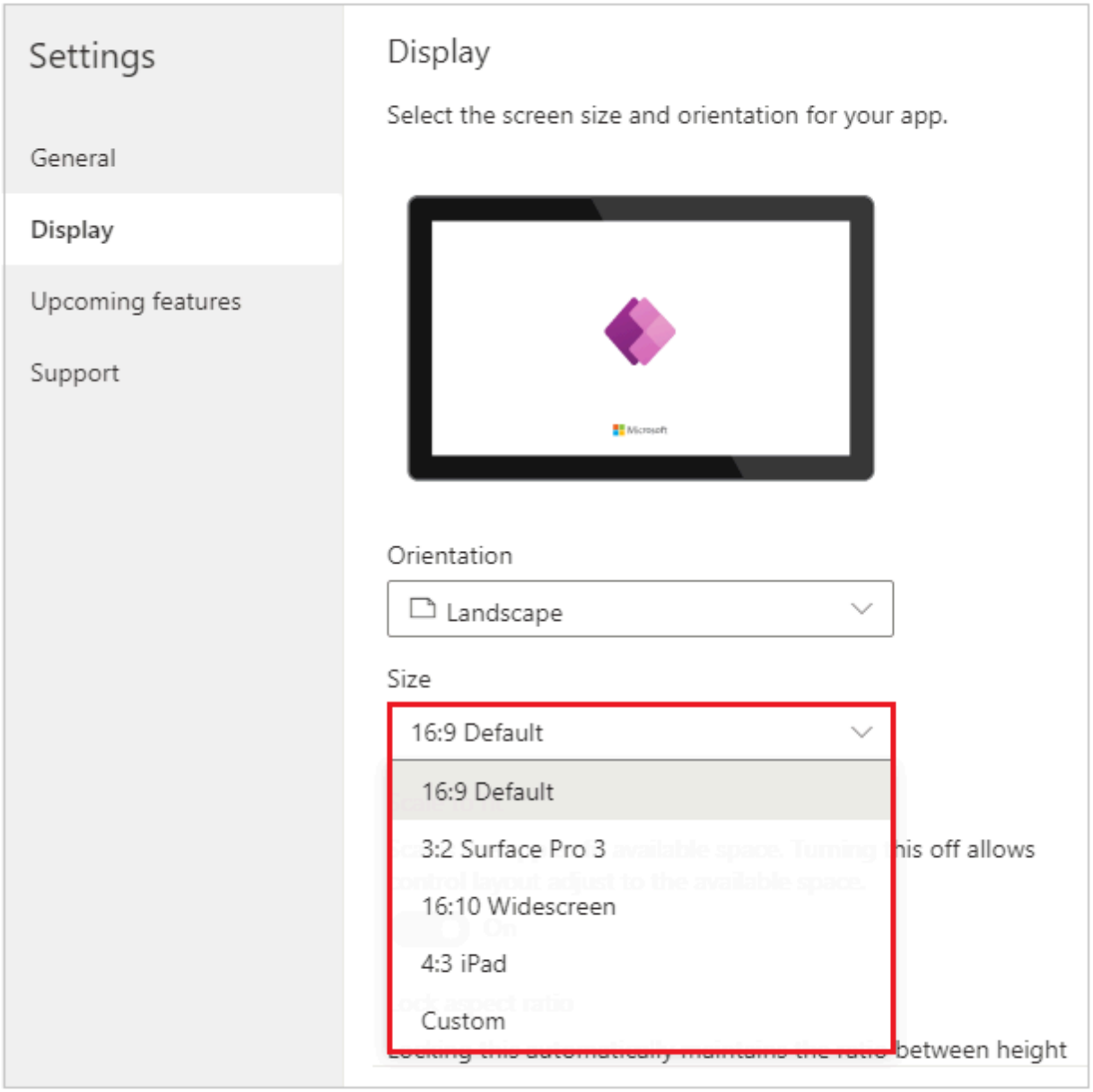
Change screen size and orientation

1. Sign in to Power Apps.
2. Open the app to edit.
3. Select **Settings**.
4. Select **Display**.
5. Under **Orientation** list, select **Portrait** or **Landscape**.



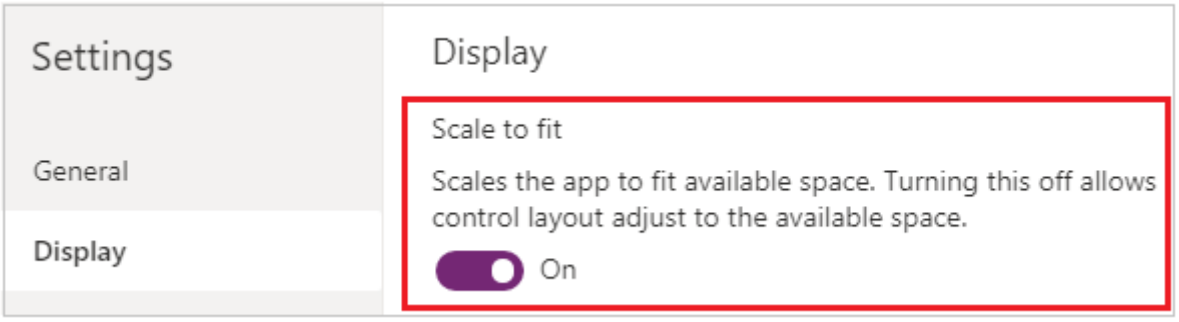
6. (Tablet apps only) Under **Aspect ratio**, perform either of these steps:
   - Select the ratio that matches the target device for this app.

- Select **Custom** to set your own size, and then specify a width between 50 - 3840 and a height between 50 - 2160.



> ⓘ **Note**
>
> **Size** is only available for apps with **Tablet** layout.

7. Under **Scale to fit**, specify either **On** or **Off**.

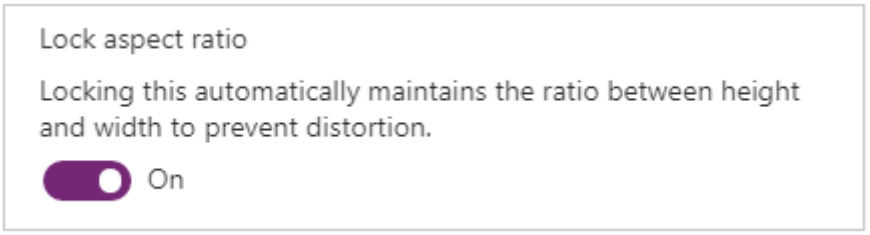


This setting is on by default so that app screens resize to fit the available space on the device. When this setting is on, the app's **Width** property matches its **DesignWidth**, and the app's **Height** matches its **DesignHeight**.

If you turn this setting off, the app adjusts to the aspect ratio of the device on which it's running and takes up all of the available space. The app doesn't scale and, as a result, screens can show more information.

When this setting is turned off, **Lock aspect ratio** is automatically turned off and disabled. In addition, the **Width** property of all screens is set to `Max(App.Width, App.DesignWidth)`, and their **Height** property is set to `Max(App.Height, App.DesignHeight)` so that they track the dimensions of the window in which the app is running. With this change, you can create apps that respond to different devices and window dimensions. More information: Create responsive layout

8. Under **Lock aspect ratio**, specify either **On** or **Off**.



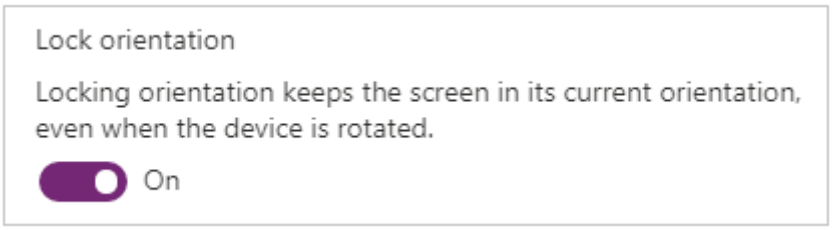
If this setting is on, the app retains the screen orientation and aspect ratio that you specified in steps 2 and 3, no matter the device. For example, a phone app that's running in a web browser retains the ratio for a phone, showing a dark bar on each side instead of filling the window.

If this setting is off, the app adjusts to the aspect ratio of the device on which it's running. We don't recommend this because the app can be distorted to the point of being unusable, depending on the screen size. For example, controls may overlap or text might be clipped.

> **Note**

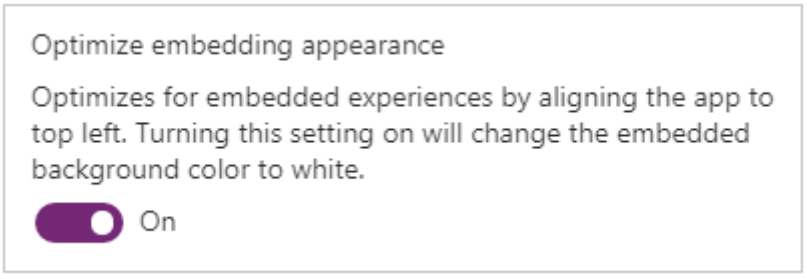
This setting can't be turned off for certain controls such as Rich text editor and Fluent UI controls. They won't distort according to screen size. Create a responsive layout to specify how these controls should adapt to different screen sizes.

9. Under **Lock orientation**, specify either **On** or **Off**.



If you lock the app's orientation, the app retains the orientation that you specify. If the app is running on a device for which the screen is in a different orientation, the app displays incorrectly and may show unwanted results. If you unlock the app's orientation, it adjusts to the screen orientation of the device on which it's running.

You can also modify the app's orientation by enabling **Optimize embedding appearance** in **Settings** > **Display**. This feature top-left aligns the app when it's embedded and changes the background color of the hosting canvas to white.



10. Close **Settings** dialog.

11. Save and publish your app.

Enumerated app behavior for scale to fit and lock aspect ratio settings

Expand table

Scenario	Scale to fit	Lock aspect ratio	App layout behavior on 'screen resize'	Notes to include in docs
1	Enabled	Enabled	The screen width and height are set by the maker. The screen scales to the	

Scenario	Scale to fit	Lock aspect ratio	App layout behavior on 'screen resize'	Notes to include in docs
			window size available.	
2	Disabled	Enabled	Not applicable. When scale to fit is disabled, lock aspect ratio is also disabled.	
3	Enabled	Disabled	In Power Apps Studio, the screen scales to the window size available. In the end-user experience, Power Apps scales to the smallest edge (width or height), and then fills the UI for the larger edge.	For apps built for mobile, we recommend makers use **lock orientation** with this setting.
4	Disabled	Disabled	App experiences can be responsive. Makers can manipulate control locations using Power Fx to set X and Y values as well as width and height.	For more information, see Responsive layout documentation and consider using auto-layout containers.

Next steps

- Create responsive layouts in canvas apps.
- Check common issues and resolutions if you're running into any problems.

Related information

Change screen size and orientation of canvas apps (video)

Create responsive layouts in canvas apps

Summarize this article for me

Before you building a canvas app in Power Apps, specify whether to tailor the app for a phone or a tablet. This choice determines the size and shape of the canvas on which you build your app.

After you make that choice, you can make a few more choices if you select **Settings** > **Display**. You can choose portrait or landscape orientation and screen size (tablet only). You can also lock or unlock the aspect ratio and support device rotation (or not).

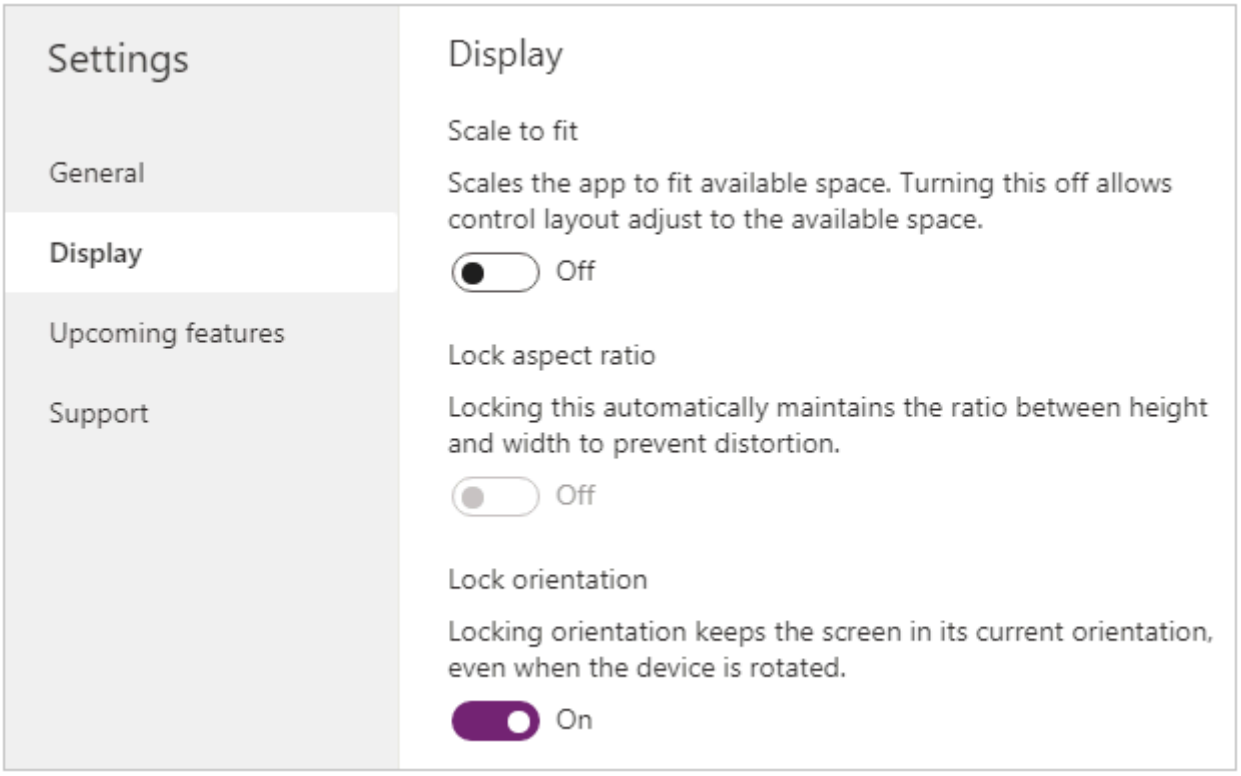
Those choices underlie every other choice you make as you design screen layouts. If your app runs on a device of a different size or on the web, your entire layout scales to fit the screen where the app is running. If an app designed for a phone runs in a large browser window, for example, the app scales to compensate and looks oversized for its space. The app can't take advantage of the additional pixels by showing more controls or more content.

If you create a responsive layout, controls can respond to different devices or window sizes, making various experiences feel more natural. To achieve a responsive layout, you adjust some settings and write expressions throughout your app.

Disable Scale to fit

You can configure each screen control so that its layout adapts to the actual space in which the app is running.

You activate responsiveness by turning off the app's **Scale to fit** setting, which is on by default. When you turn this setting off, you also turn off **Lock aspect ratio** because you're no longer designing for a specific screen shape. (You can still specify whether your app supports device rotation.)



To make your app responsive, you must take additional steps, but this change is the first step toward making responsiveness possible.

Understand app and device screen dimensions

To make your app's layouts respond to changes in the device screen, or web browser, dimensions, you'll write formulas that use the **Width** and **Height** properties of the screen. To show these properties, open an app in Power Apps Studio, and then select a screen control. The default formulas for these properties appear on the **Advanced** tab of the right-hand pane.

Width = `Max(App.Width, App.MinScreenWidth)`

Height = `Max(App.Height, App.MinScreenHeight)`

These formulas refer to the **Width**, **Height**, **MinScreenWidth**, and **MinScreenHeight** properties of the app. The app's **Width** and **Height** properties correspond to the dimensions of the device or browser window in which your app is running. If the user resizes the browser window (or rotates the device if you've turned off **Lock orientation**), the values of these properties change dynamically. The formulas in the screen control's **Width** and **Height** properties are reevaluated when these values change.

The values in the **App.Width** and **App.Height** properties come from the dimensions that you specify in the **Display** pane of **Settings**. For example, if you select the landscape "16:9 Default" size, **App.Width** is 1366, and **App.Height** is 768.

As they're used in the formulas for the screen control's **Width** and **Height** properties, **MinScreenWidth** and **MinScreenHeight** are the minimum dimensions for which you'll design the app. If the actual area available to your app is even smaller than these minimum dimensions, the formulas for the screen control's **Width** and **Height** properties ensure that their values won't become any smaller than minimums. In that case, the user must scroll to view all of the layout's content. The **MinScreenWidth** and **MinScreenHeight** properties can be set in the **App Object** > **Properties** > **Advanced**.

After you establish your app's **MinScreenWidth** and **MinScreenHeight**, you won't (in most cases) need to change default formulas for each screen control's **Width** and **Height** properties. Later, this topic discusses cases in which you might want to customize these formulas.

Use formulas for dynamic layout

To create a responsive design, you locate and size each control by using formulas instead of absolute (constant) coordinate values. These formulas express each control's position and size in terms of the overall screen size or relative to other controls in the current screen.

> **Important**
>
> After you write formulas for the **X**, **Y**, **Width** and **Height** properties of a control, your formulas will be overwritten with constant values if you subsequently drag the control in the canvas editor. When you start to use formulas to achieve dynamic layout, you should avoid dragging controls.

In the simplest case, one control fills an entire screen. To create this effect, set the control's properties to these values:

Expand table

Property	Value
X	`0`
Y	`0`
Width	`Parent.Width`
Height	`Parent.Height`

These formulas use the **Parent** operator. For any control placed directly on a screen, **Parent** refers to the screen. With these property values, the control appears in the upper-left corner of the screen (0, 0) and has the same **Width** and **Height** as the screen.

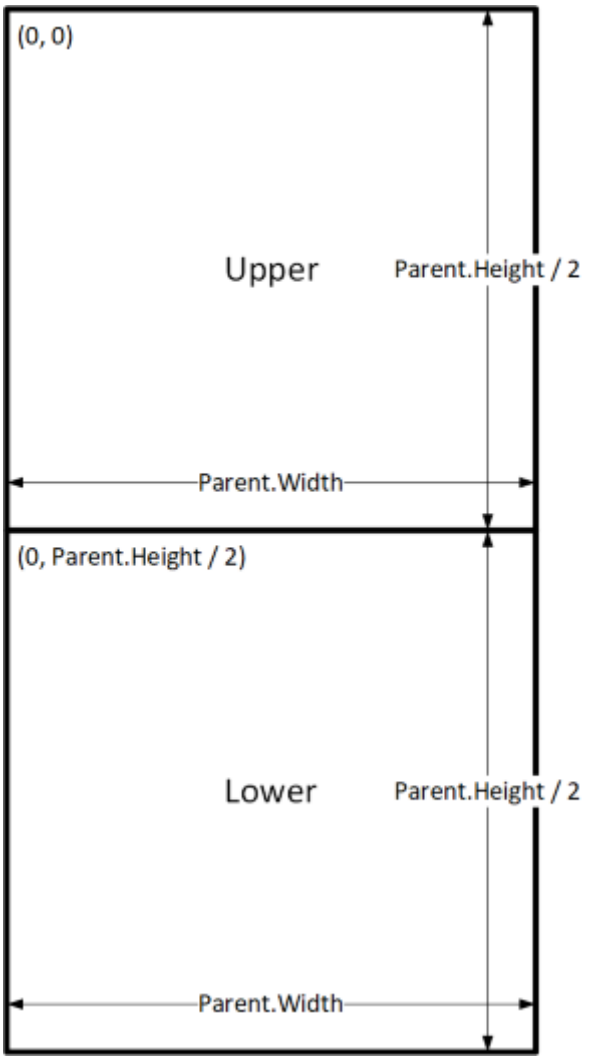
Later in this topic, you'll apply these principles (and the **Parent** operator) to position controls inside other containers, such as galleries, group controls, and components.

As an alternative, the control can fill only the top half of the screen. To create this effect, set the **Height** property to **Parent.Height** / 2, and leave the other formulas unchanged.

If you want a second control to fill the bottom half of the same screen, you can take at least two other approaches to constructing its formulas. For simplicity, you might take this approach:

Expand table

Control	Property	Formula
Upper	X	`0`
Upper	Y	`0`
Upper	Width	`Parent.Width`
Upper	Height	`Parent.Height / 2`
Lower	X	`0`
Lower	Y	`Parent.Height / 2`
Lower	Width	`Parent.Width`
Lower	Height	`Parent.Height / 2`



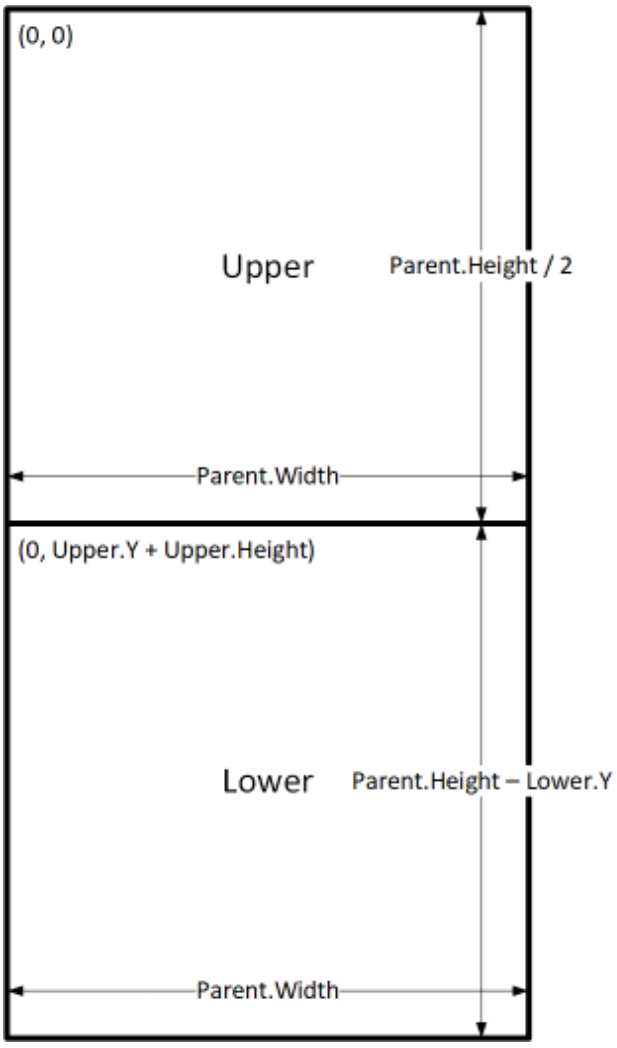
This configuration would achieve the effect that you want, but you'd need to edit each formula if you changed your mind about the relative sizes of the controls. For example, you might decide that the top control should occupy only the top one-third of the screen, with the bottom control filling the lower two-thirds.

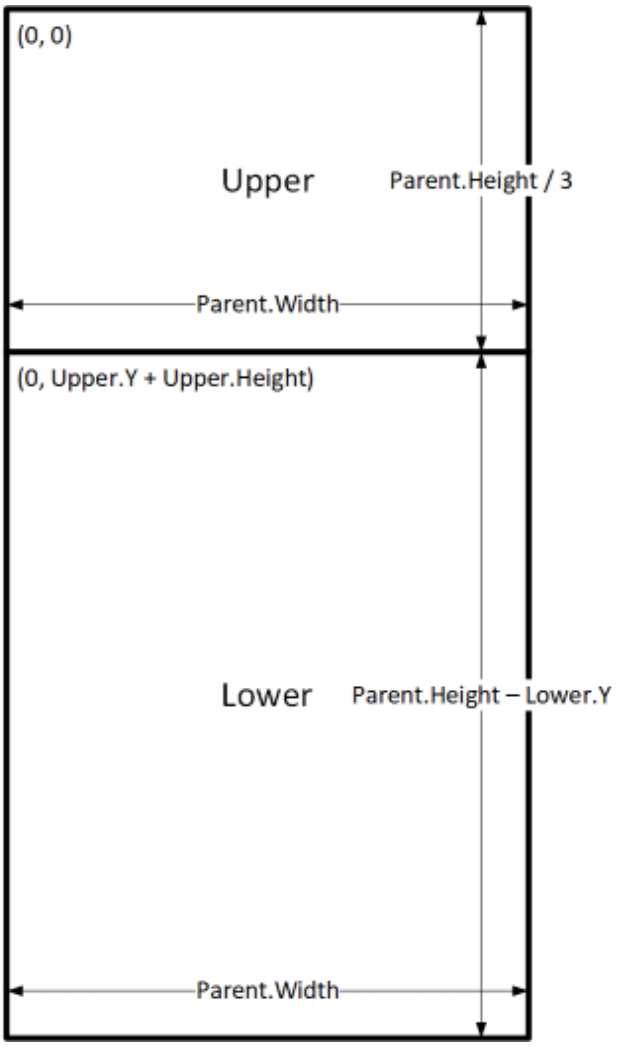
To create that effect, you'd need to update the **Height** property of the **Upper** control and the **Y** and **Height** properties of the **Lower** control. Instead, consider writing the formulas for the **Lower** control in terms of the **Upper** control (and itself), as in this example:

Expand table

Control	Property	Formula
Upper	X	`0`
Upper	Y	`0`
Upper	Width	`Parent.Width`
Upper	Height	`Parent.Height / 3`
Lower	X	`0`

Control	Property	Formula
Lower	Y	`Upper.Y + Upper.Height`
Lower	Width	`Parent.Width`
Lower	Height	`Parent.Height - Lower.Y`

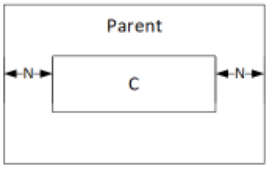
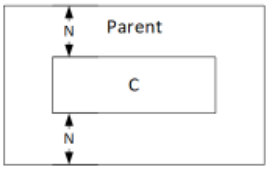
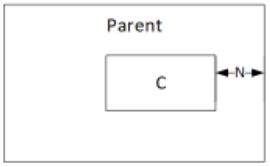
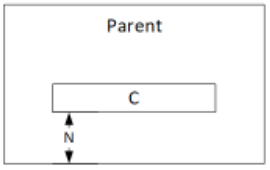
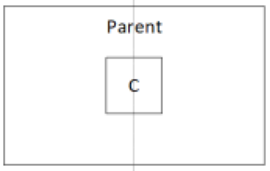
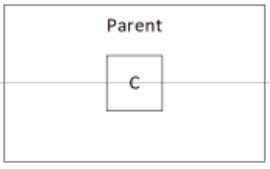




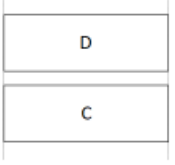
With these formulas in place, you need only change the **Height** property of the **Upper** control to express a different fraction of the height of the screen. The **Lower** control automatically moves and resizes to account for the change.

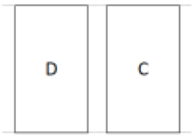
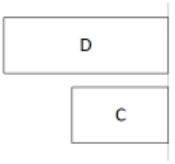
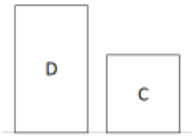
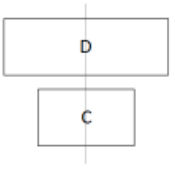
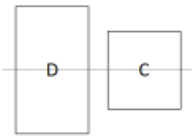
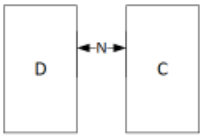
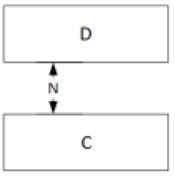
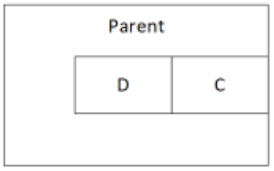
You can use these formula patterns for expressing common layout relationships between a control, named **C**, and its parent or a sibling control, named **D**.

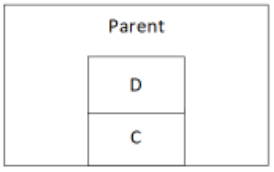
Expand table

Relationship between C and its parent	Property	Formula	Illustration
C fills width of parent, with a margin of *N*	**X**	`N`	
	Width	`Parent.Width - (N * 2)`	
C fills height of parent, with a margin of *N*	**Y**	`N`	
	Height	`Parent.Height - (N * 2)`	
C aligned with right edge of parent, with margin of *N*	**X**	`Parent.Width - (C.Width + N)`	
C aligned with bottom edge of parent, with margin of *N*	**Y**	`Parent.Height - (C.Height + N)`	
C centered horizontally on parent	**X**	`(Parent.Width - C.Width) / 2`	
C centered vertically on parent	**Y**	`(Parent.Height - C.Height) / 2`	

Expand table

Relationship between C and D	Property	Formula	Illustration
C horizontally aligned with D and the same width as D	**X**	`D.X`	

Relationship between C and D	Property	Formula	Illustration
	Width	`D.Width`	
C vertically aligned with **D** and same height as **D**	**Y**	`D.Y`	
	Height	`D.Height`	
Right edge of **C** aligned with right edge of **D**	**X**	`D.X + D.Width - C.Width`	
Bottom edge of **C** aligned with bottom edge of **D**	**Y**	`D.Y + D.Height - C.Height`	
C centered horizontally relative to **D**	**X**	`D.X + (D.Width - C.Width) / 2`	
C centered vertically relative to **D**	**Y**	`D.Y + (D.Height - C.Height) /2`	
C positioned to the right of **D** with a gap of N	**X**	`D.X + D.Width + N`	
C positioned below **D** with a gap of *N*	**Y**	`D.Y + D.Height + N`	
C fills space between **D** and right edge of parent	**X**	`D.X + D.Width`	

Relationship between C and D	Property	Formula	Illustration
	Width	`Parent.Width - C.X`	
C fills space between **D** and bottom edge of parent	Y	`D.Y + D.Height`	
	Height	`Parent.Height - C.Y`	

Hierarchical layout

As you construct screens that contain more controls, it will become more convenient (or even necessary) to position controls relative to a parent control, rather than relative to the screen or a sibling control. By organizing your controls into a hierarchical structure, you can make your formulas easier to write and maintain.

Galleries

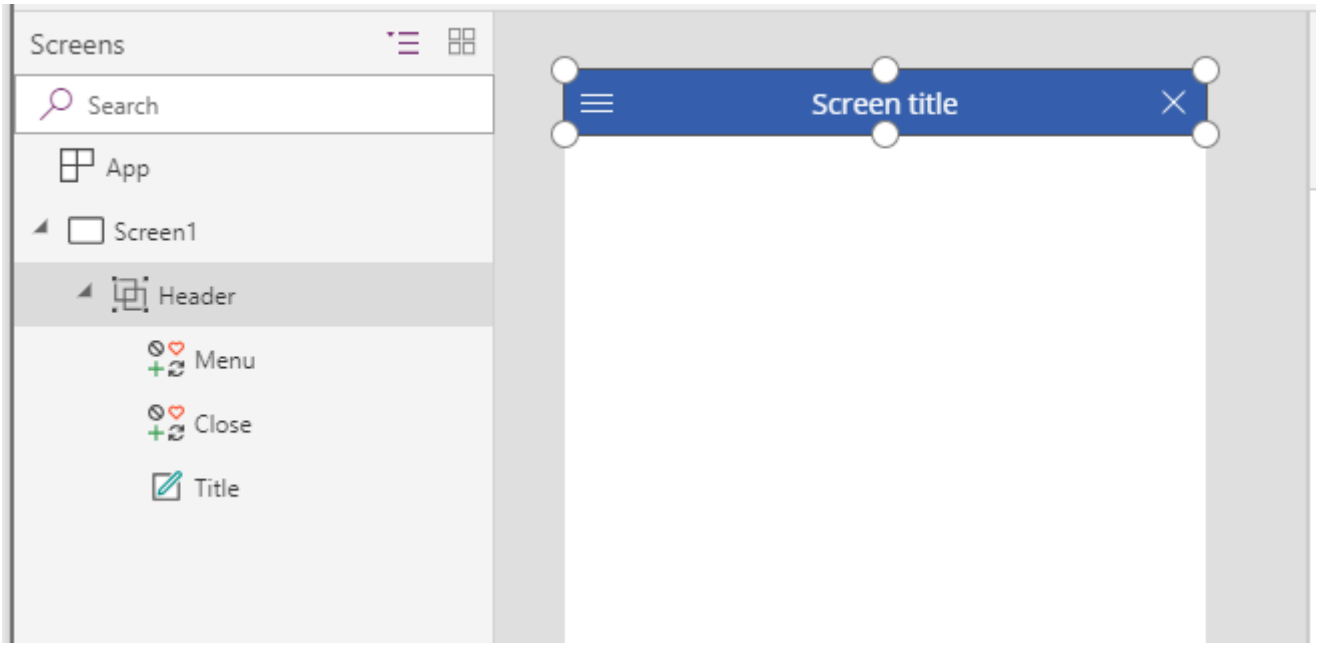
If you use a gallery in your app, you'll need to lay out controls within the gallery's template. You can position these controls by writing formulas that use the **Parent** operator, which will refer to the gallery template. In the formulas on controls within a gallery template, use the **Parent.TemplateHeight** and **Parent.TemplateWidth** properties; don't use **Parent.Width** and **Parent.Height**, which refer to the overall size of the gallery.



Container control

You can use the **Layout container** control, as a parent control.

Consider the example of a header at the top of a screen. It's common to have a header with a title and several icons with which your users can interact. You can construct such a header using the **Container** control, containing a **Label** control and two **Icon** controls:



Set the properties for these controls to these values:

Expand table

Property	Header	Menu	Close	Title
X	`0`	`0`	`Parent.Width - Close.Width`	`Menu.X + Menu.Width`
Y	`0`	`0`	`0`	`0`
Width	`Parent.Width`	`Parent.Height`	`Parent.Height`	`Close.X - Title.X`
Height	`64`	`Parent.Height`	`Parent.Height`	`Parent.Height`

For the **Header** control, `Parent` refers to the screen. For the others, `Parent` refers to the **Header** control.

Having written these formulas, you can adjust the size or position of the **Header** control by changing the formulas for its properties. The sizes and positions of the child controls will automatically adjust accordingly.

Auto-layout container controls

You can use an feature, the **Auto-layout** container controls to automatically lay out the child components. These containers determine the position of the child components so that you never have to set X, Y for a component inside the container. Also, it can distribute the available space to its child components based on the settings, as well as determines both the vertical and horizontal alignment of the child components. More information: Auto-layout container controls

Components

If you use another feature, named Components, you can construct building blocks and reuse them throughout your app. As with the **Container** control, the controls that you place within a component should base their position and size formulas on `Parent.Width` and `Parent.Height`, which refer to the size of the component. More information: Create a component.

Adapting layout for device size and orientation

So far, you've learned how to use formulas to change each control's size in response to the available space, while keeping controls aligned relative to each other. But you might want or need to make more substantial layout changes in response to different device sizes and orientations. When a device is rotated from portrait to landscape orientation, for example, you might want to switch from a vertical layout to a horizontal one. On a larger device, you can

present more content or rearrange it to provide a more appealing layout. On a smaller device, you might need to split up content across multiple screens.

Device orientation

The default formulas for a screen's **Width** and **Height** properties, as this topic described earlier, won't necessarily provide a good experience if a user rotates a device. For example, an app designed for a phone in portrait orientation has a **DesignWidth** of 640 and a **DesignHeight** of 1136. The same app on a phone in landscape orientation will have these property values:

- The screen's **Width** property is set to `Max(App.Width, App.DesignWidth)`. The app's **Width** (1136) is larger than its **DesignWidth** (640), so the formula evaluates to 1136.
- The screen's **Height** property is set to `Max(App.Height, App.DesignHeight)`. The app's **Height** (640) is smaller than its **DesignHeight** (1136), so the formula evaluates to 1136.

With a screen **Height** of 1136 and a device height (in this orientation) of 640, the user must scroll the screen vertically to show all of its content, which might not be the experience that you want.

To adapt the screen's **Width** and **Height** properties to the device orientation, you can use these formulas:

Width = `Max(App.Width, If(App.Width < App.Height, App.DesignWidth, App.DesignHeight))`

Height = `Max(App.Height, If(App.Width < App.Height, App.DesignHeight, App.DesignWidth))`

These formulas swap the app's **DesignWidth** and **DesignHeight** values, based on whether the device's width is less than its height (portrait orientation) or more than its height (landscape orientation).

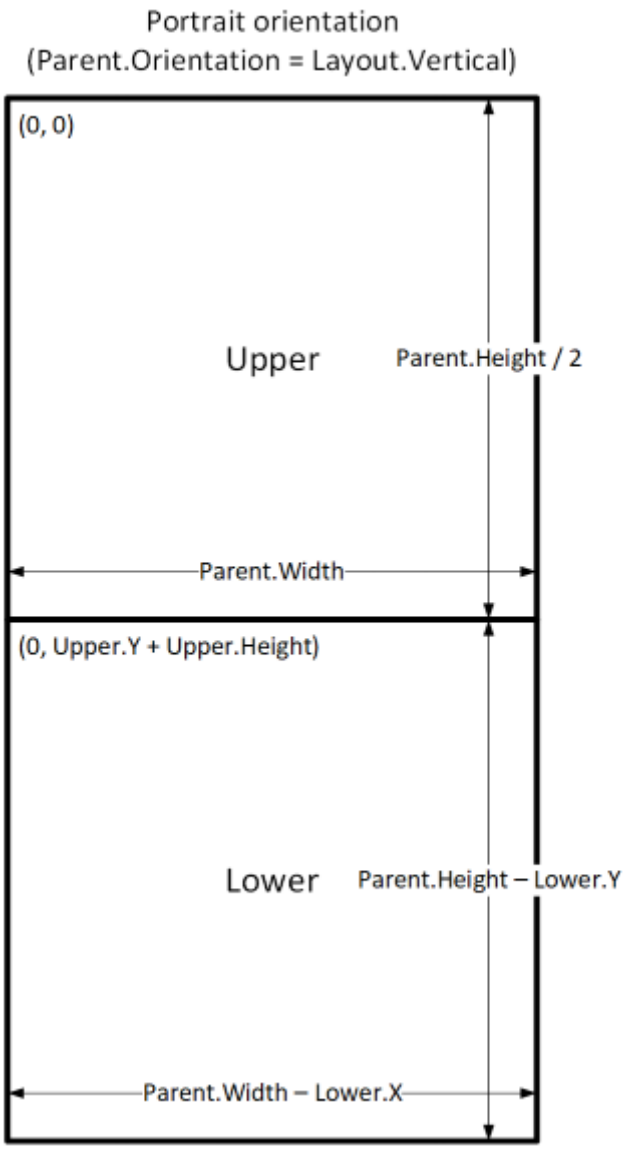
After you adjust the screen's **Width** and **Height** formulas, you might also want to rearrange controls within your screen to better use the available space. For example, if each of two controls occupies half of the screen, you might stack them vertically in portrait but arrange them side by side in landscape.

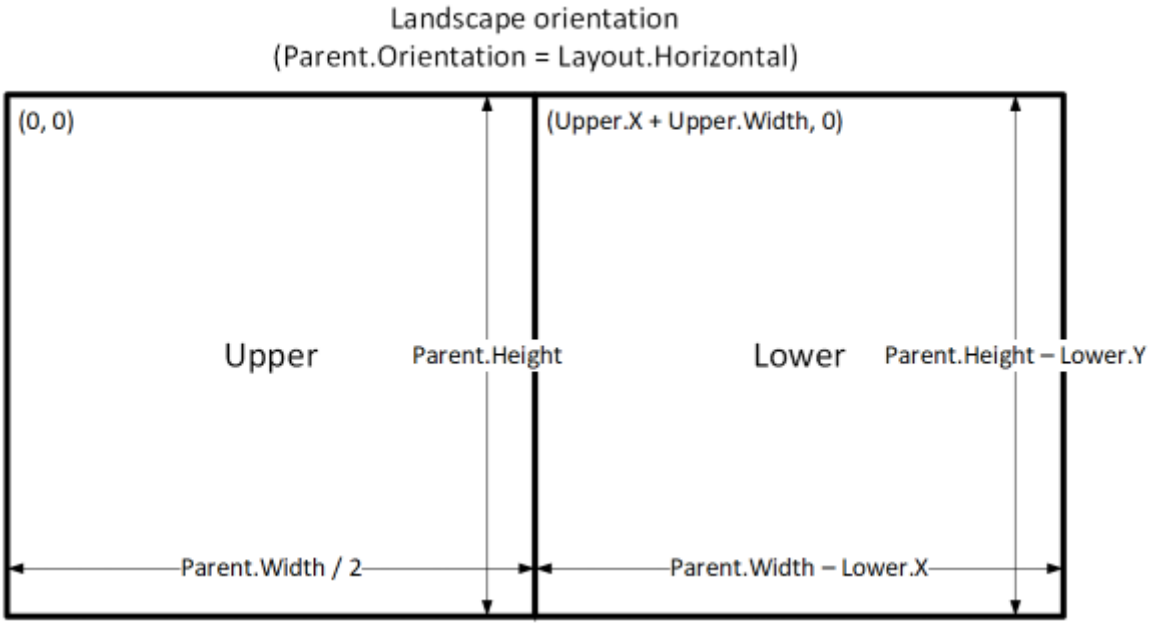
You can use the screen's **Orientation** property to determine whether the screen is oriented vertically or horizontally.

> ⓘ **Note**
>
> In landscape orientation, the **Upper** and **Lower** controls appear as left and right controls.

Expand table

Control	Property	Formula
Upper	X	`0`
Upper	Y	`0`
Upper	Width	`If(Parent.Orientation = Layout.Vertical, Parent.Width, Parent.Width / 2)`
Upper	Height	`If(Parent.Orientation = Layout.Vertical, Parent.Height / 2, Parent.Height)`
Lower	X	`If(Parent.Orientation = Layout.Vertical, 0, Upper.X + Upper.Width)`
Lower	Y	`If(Parent.Orientation = Layout.Vertical, Upper.Y + Upper.Height, 0)`
Lower	Width	`Parent.Width - Lower.X`
Lower	Height	`Parent.Height - Lower.Y`





Screen sizes and breakpoints

You can adjust your layout based on the size of the device. The screen's **Size** property classifies the current device size. The size is a positive integer; the ScreenSize type provides named constants to help with readability. This table lists the constants:

Expand table

Constant	Value	Typical device type (using default app settings)
ScreenSize.Small	1	Phone
ScreenSize.Medium	2	Tablet, held vertically
ScreenSize.Large	3	Tablet, held horizontally
ScreenSize.ExtraLarge	4	Desktop computer

Use these sizes to make decisions about your app's layout. For example, if you want a control to be hidden on a phone-sized device but visible otherwise, you could set the control's **Visible** property to this formula:

Power Fx

```
Parent.Size >= ScreenSize.Medium
```

This formula evaluates to **true** when the size is medium or larger and **false** otherwise.

If you want a control to occupy a different fraction of the screen width based on the screen size, set the control's **Width** property to this formula:

Power Fx

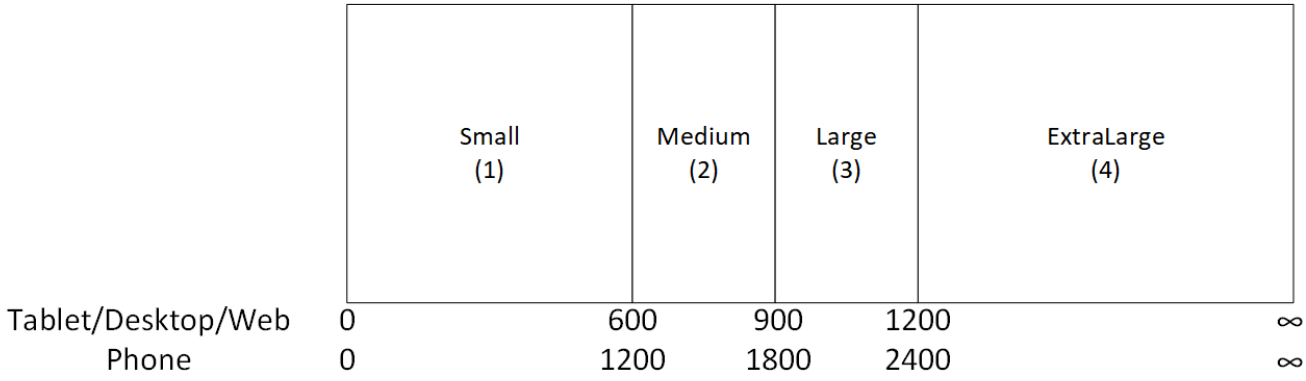
```
Parent.Width *
Switch(
    Parent.Size,
    ScreenSize.Small, 0.5,
    ScreenSize.Medium, 0.3,
    0.25
)
```

This formula sets the control's width to half of the screen width on a small screen, three-tenths of the screen width on a medium screen, and a quarter of the screen width on all other screens.

Custom breakpoints

The screen's **Size** property is calculated by comparing the screen's **Width** property to the values in the app's **SizeBreakpoints** property. This property is a single-column table of numbers that indicate the width breakpoints that separate the named screen sizes:

In an app created for tablet or web, the default value in the app's **SizeBreakpoints** property are **[600, 900, 1200]**. In an app created for phones, the value is **[1200, 1800, 2400]**. (The values for phone apps are doubled because such apps use coordinates that are effectively double the coordinates used in other apps.)



You can customize your app's breakpoints by changing the values in the app's **SizeBreakpoints** property. Select **App** in the tree view, select **SizeBreakpoints** in the property list, and then edit the values in the formula bar. You can create as many breakpoints as your app needs, but only sizes 1 through 4 correspond to named screen sizes. In formulas, you can refer to sizes beyond ExtraLarge by their numeric values (5, 6, and so forth).

You can also specify fewer breakpoints. For example, your app might need only three sizes (two breakpoints), so the possible screen sizes will be Small, Medium, and Large.

Known limitations

The authoring canvas doesn't respond to the sizing formulas created. To test responsive behavior, save and publish your app, and then open it on devices or in browser windows of various sizes and orientations.

If you write expressions or formulas in the **X**, **Y**, **Width**, and **Height** properties of a control, you'll overwrite those expressions or formulas if you later drag the control to a different location or resize the control by dragging its border.

Last updated on 02/28/2026

Get started with formulas in canvas apps

Article • 01/18/2025

Configure your canvas app with formulas that not only calculate values and perform other tasks (as they do in Excel) but also respond to user input (as an app requires).

- In Excel, you build formulas that, for example, populate cells and create tables and charts.
- In Power Apps, you build similar formulas as you configure controls instead of cells. In addition, you build formulas that apply specifically to apps instead of spreadsheets.

For example, you build a formula to determine how your app responds when users select a button, adjust a slider, or provide other input. These formulas might show a different screen, update a data source that's external to the app, or create a table that contains a subset of the data in an existing table.

You can use formulas for a wide variety of scenarios. For example, you can use your device's GPS, a map control, and a formula that uses **Location.Latitude** and **Location.Longitude** to display your current location. As you move, the map automatically tracks your location.

This article provides only an overview of working with formulas. Browse the formula reference for more details and the complete list of functions, operators, and other building blocks you can use.

Prerequisites

1. Sign up for Power Apps and sign in with your credentials.
2. Learn how to configure a control in Power Apps.

Use Power Fx formula bar

The Power Fx formula bar offers a more intuitive and efficient way to write formulas for your apps. Follow these steps to use the formula bar:

1. Open your app for editing in Power Apps Studio.
2. Open the formula bar by selecting it at the top of the screen.

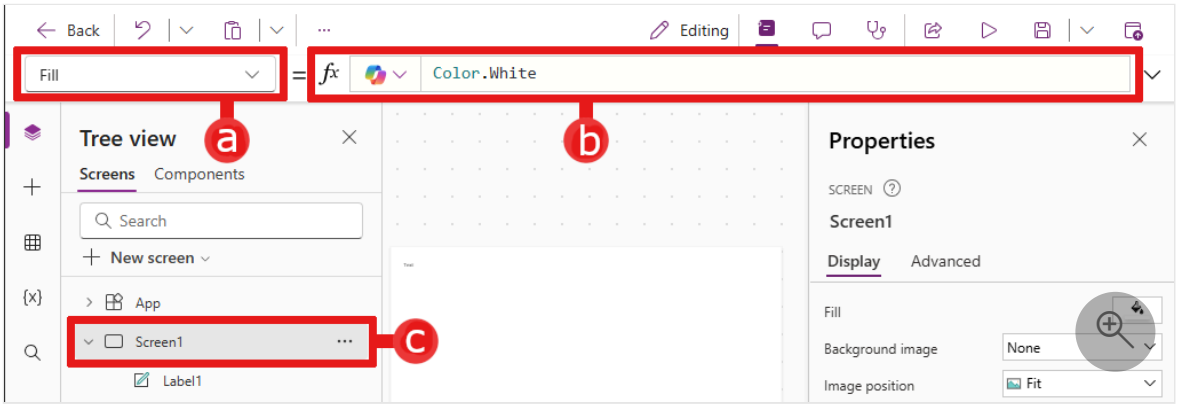
3. Start typing your formula in the bar. As you type, the formula bar provides suggestions for functions that match your input.
4. Continue typing your formula or select the suggestions until you're done.

Show a value

In Excel, you can enter a specific piece of data, such as the number **42** or the phrase **Hello World**, by typing it into a cell. That cell shows the data exactly as you type it. In Power Apps, you can similarly specify a piece of data that doesn't change by setting the **Text** property of a label to the exact sequence of characters that you want, surrounded by double quotation marks.

1. Create a blank canvas app.

   The formula bar sits at the top of the screen.

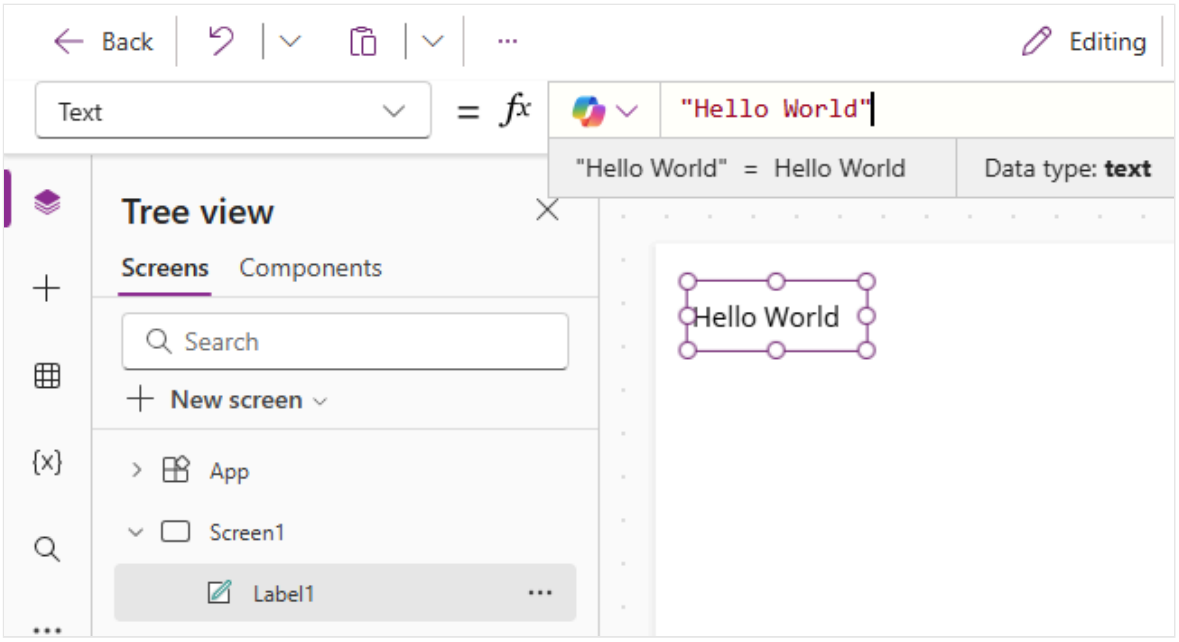


   a. *Property list*: Each control and screen has a set of properties. Use this list to select a specific property.
   b. *Formula*: The formula to be calculated for this property, made up of values, operators, and functions. As you type, Intellisense helps you with recommendations for formula, syntax, and errors.
   c. *Selected control*: In the formula bar, you can see and edit properties for the selected control or for the screen if no controls are selected.

2. Add a **Label** control to the screen.

   When you add a label, the property list automatically shows the **Text** property, which drives what the control shows. By default, the value of this property is **"Text"**.

3. Set the value of the **Text** property to **"Hello World"** by typing that string, surrounded by double quotes, into the formula bar:



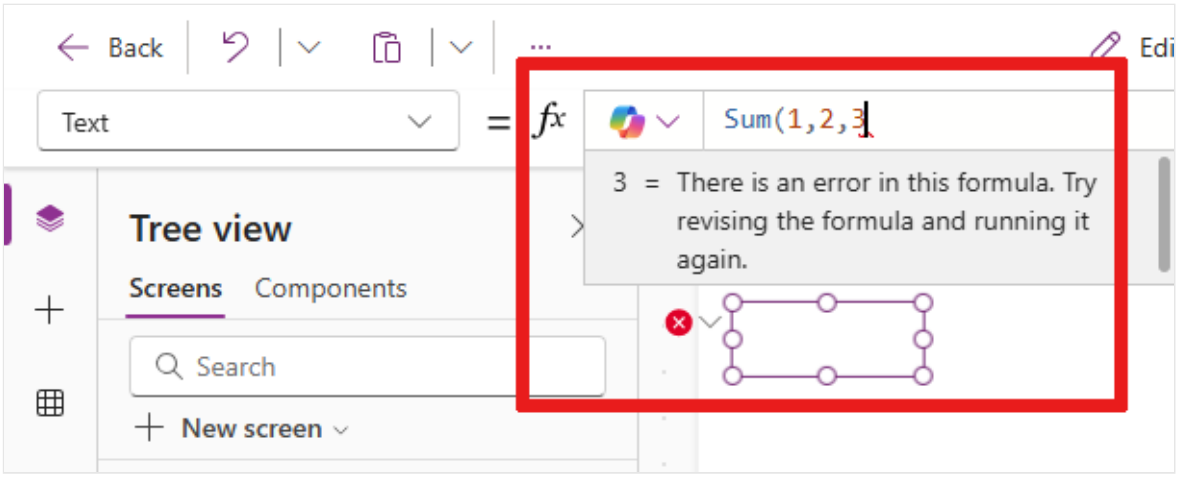
The label reflects this new value as you type it. The screen might show yellow exclamation-point icons while you type. These icons indicate errors, but go away when you finish entering a valid value. For example, a string without double quotation marks on both ends isn't valid.

In Excel, you can show a number, such as **42**, by typing it into a cell or by typing a formula that resolves to that number, such as **=SUM(30,12)**. In Power Apps, you can achieve the same effect by setting the **Text** property of a control, such as a label, to **42** or **Sum(30,12)**. The cell and the label show that number regardless of what else changes in the worksheet or the app.

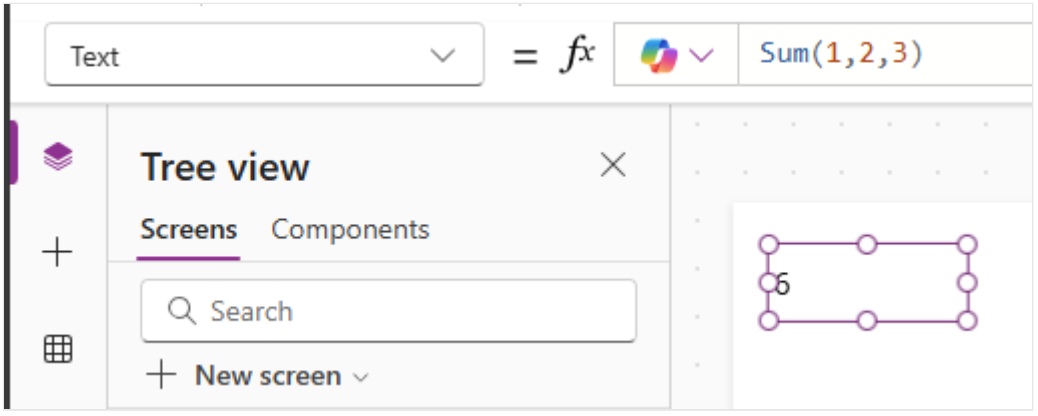
> **Note**
>
> In Power Apps, you don't precede a formula with an equals sign or a plus sign as you do in Excel. The formula bar treats anything you type there as a formula by default. You also don't surround a formula with double quotation marks ("), as you did earlier to specify a string of text.

4. In the Text property of the label, replace **"Hello World"** with **Sum(1,2,3)**.

   While you type, the formula bar helps you by showing the description and the expected arguments for this function. As with the final double quotation mark in **"Hello World"**, the screen shows a red cross, showing an error, until you type the final parenthesis of this formula.

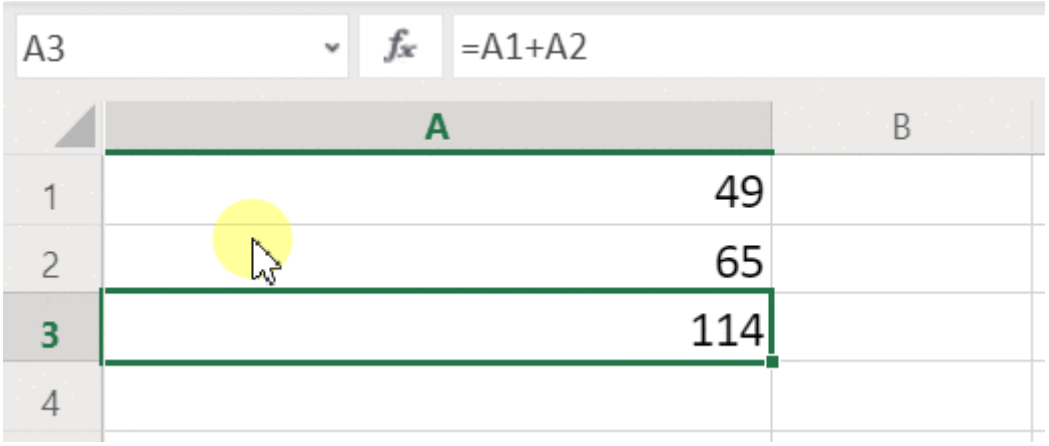


Completed formula with the final parenthesis added:



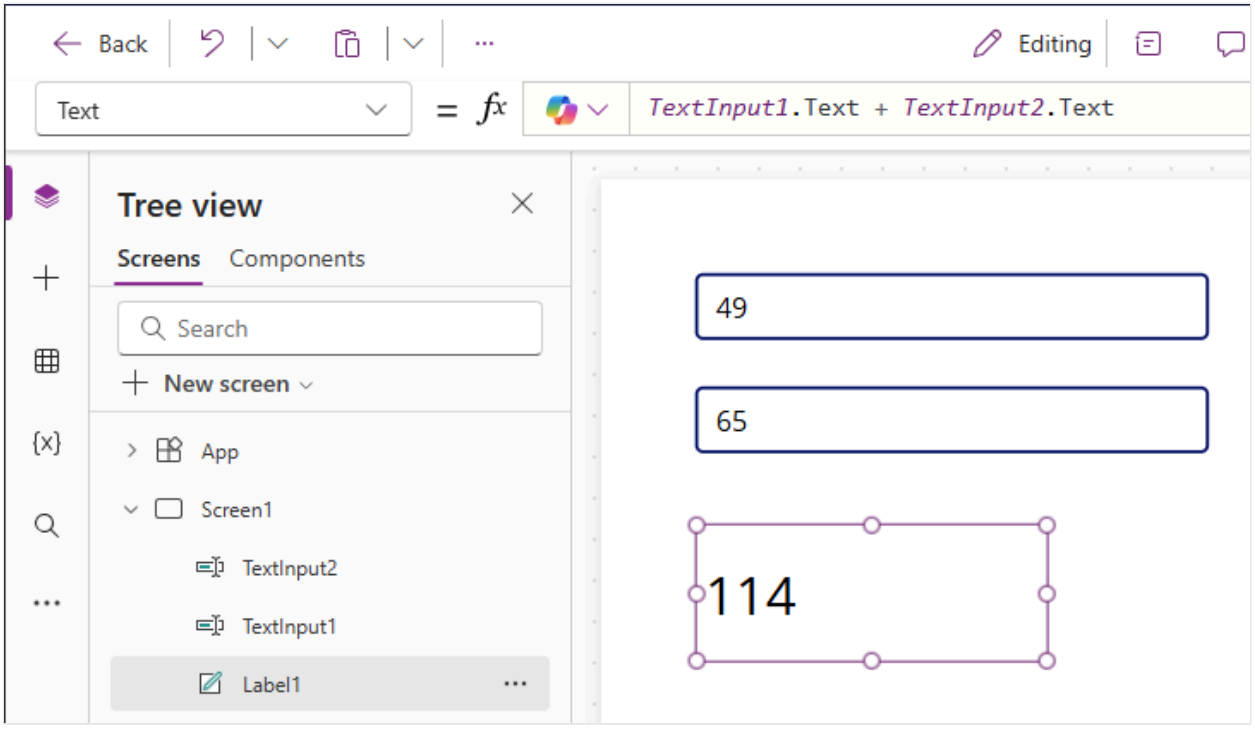
Change a value based on input

In Excel, you type **=A1+A2** into a cell to show the sum of whatever values cells **A1** and **A2** contain. If either or both of those values change, the cell that contains the formula automatically shows the updated result.



In Power Apps, you can achieve a similar result by adding controls to a screen and setting their properties. This example shows a label control named **Label1** and two Text input controls, named **TextInput1** and **TextInput2**. You can add a formula to the **Label1**

control so that when you input a number into **TextInput1** and **TextInput2**, they're added together and display in **Label1**.

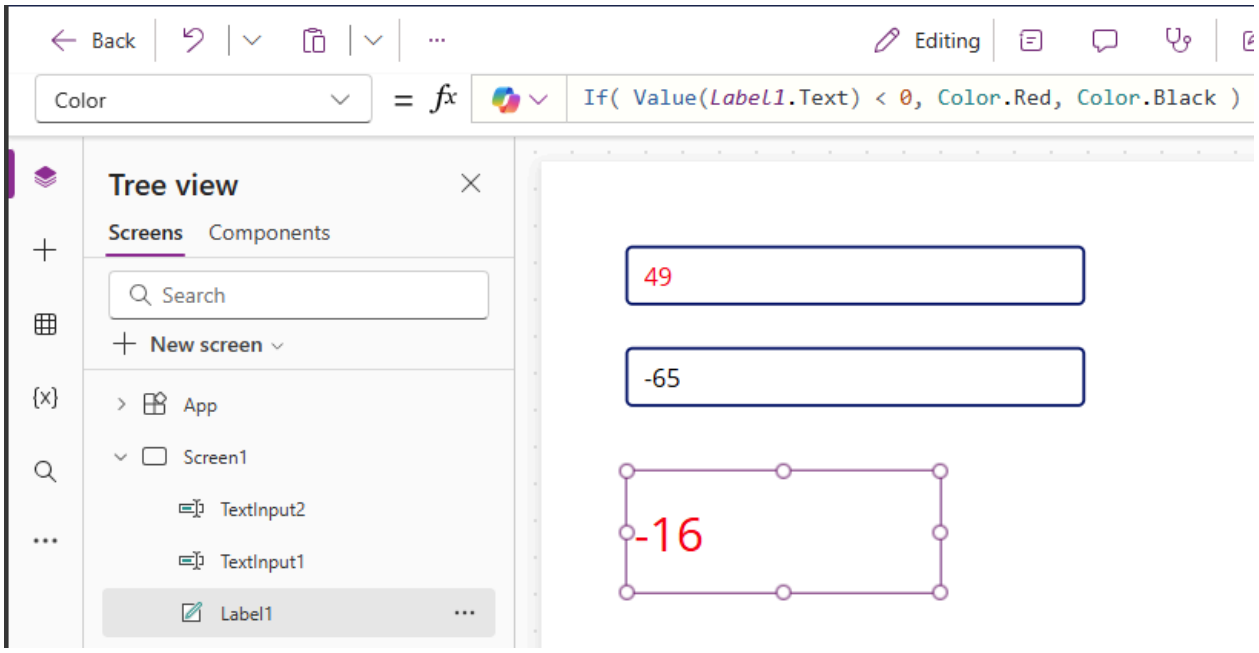


Regardless of what numbers you type in the text-input controls, the label always shows the sum of those numbers because its **Text** property is set to this formula: `TextInput1.Text + TextInput2.Text`.

In Excel, you can use conditional-formatting formulas to show, for example, negative values in red. In Power Apps, you can use formulas to determine not only the primary value of a control but also properties such as color.

In this example, a formula for the **Color** property of the label automatically shows negative values in red. The **If** function should look familiar from Excel:

```
If( Value(Label1.Text) < 0, Color.Red, Color.Black )
```

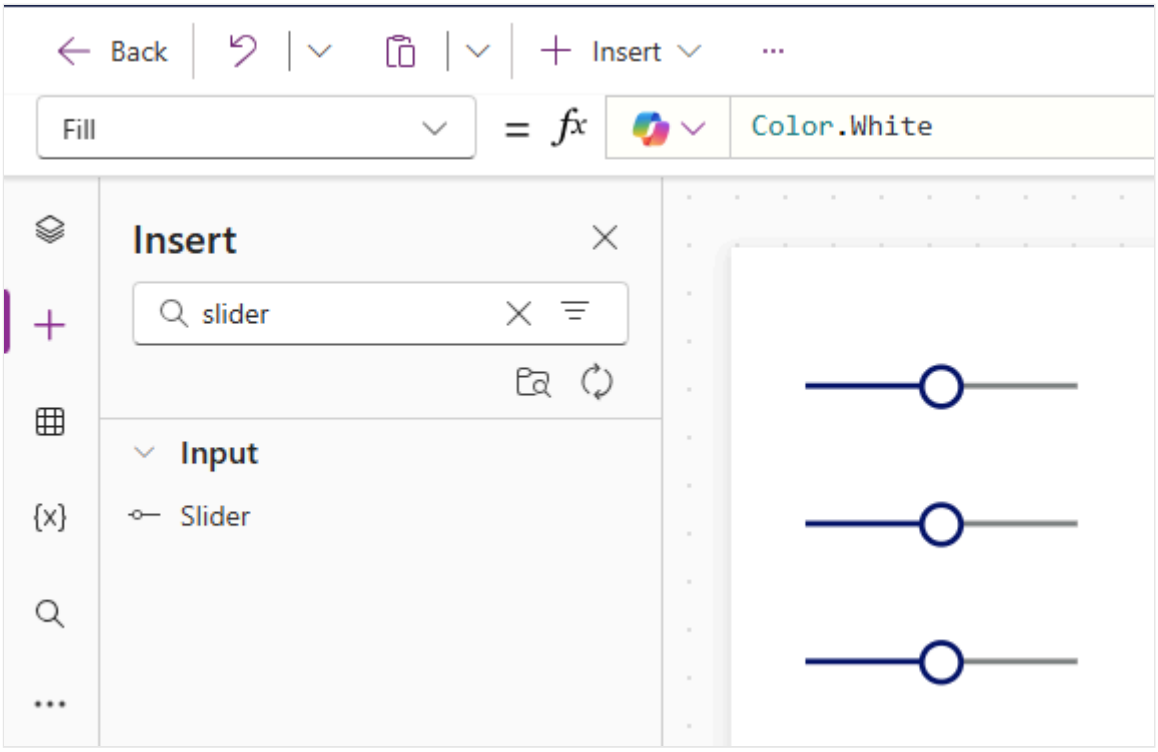


Change a color based on user input

You can configure your app with formulas so that users can change your app's appearance or behavior. For example, you can create a filter to show only data that contains a string of text that the user specifies. You can let users sort a set of data based on a certain column in the data set.

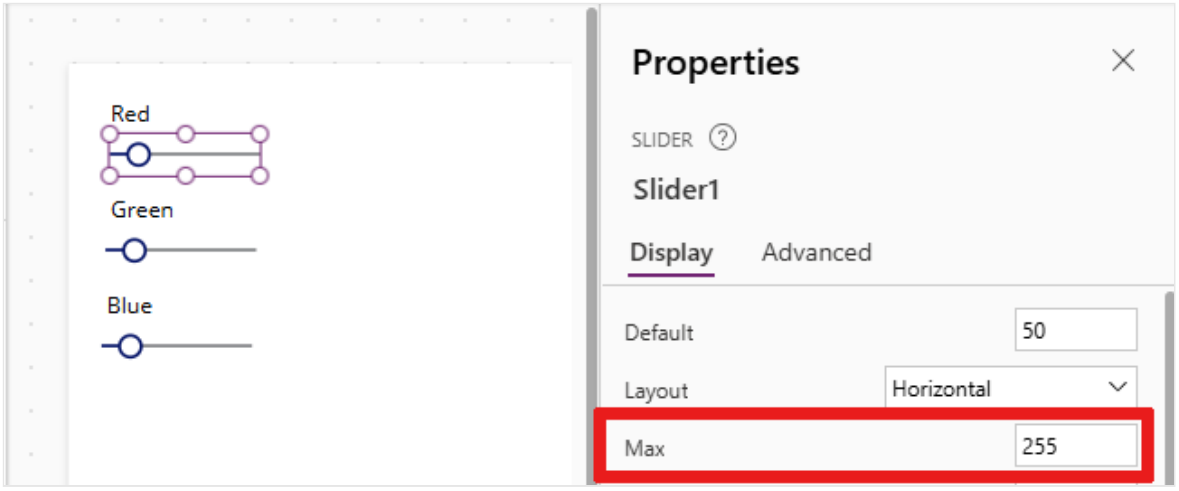
In this example, you can let users change the color of the screen by adjusting one or more sliders.

1. Remove the controls from the previous procedures, or create a blank app as you did previously, and add three slider controls to it. You can search for the slider control in the search box of the **Insert** page to find the control under **Input**. To add the sliders, drag and drop the control to the canvas.



2. Arrange the sliders so they don't overlap, add three labels, and configure the labels to show **Red**, **Green**, and **Blue** text.

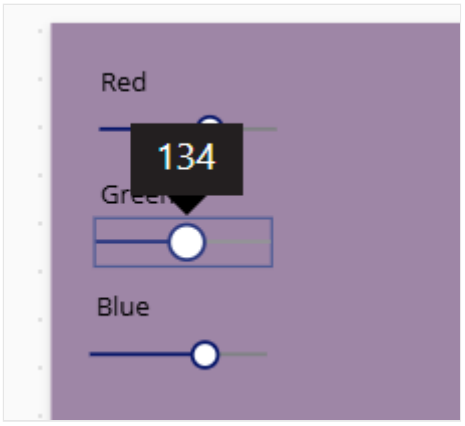
3. Set the **Max** property of each slider to 255 (default is 100), which is the maximum value of a color component for the RGBA function.



4. Select the screen to unselect any control, and then set the screen's **Fill** property to this formula: **RGBA(Slider1.Value, Slider2.Value, Slider3.Value, 1)**. Your screen turns dark gray, which reflects the sliders' current positions.

   In the formula, you can access control properties by using the property selector. For example, **Slider1.Value** refers to the slider's **Value** property, which represents where the user placed the slider between the **Min** and **Max** values.

5. **Preview the app** and adjust the sliders. You see how each control is color coded to change the background color of the screen.



As each slider changes, the formula that contains the **RGBA** function is recalculated, which immediately changes how the screen appears.

Manage app behavior

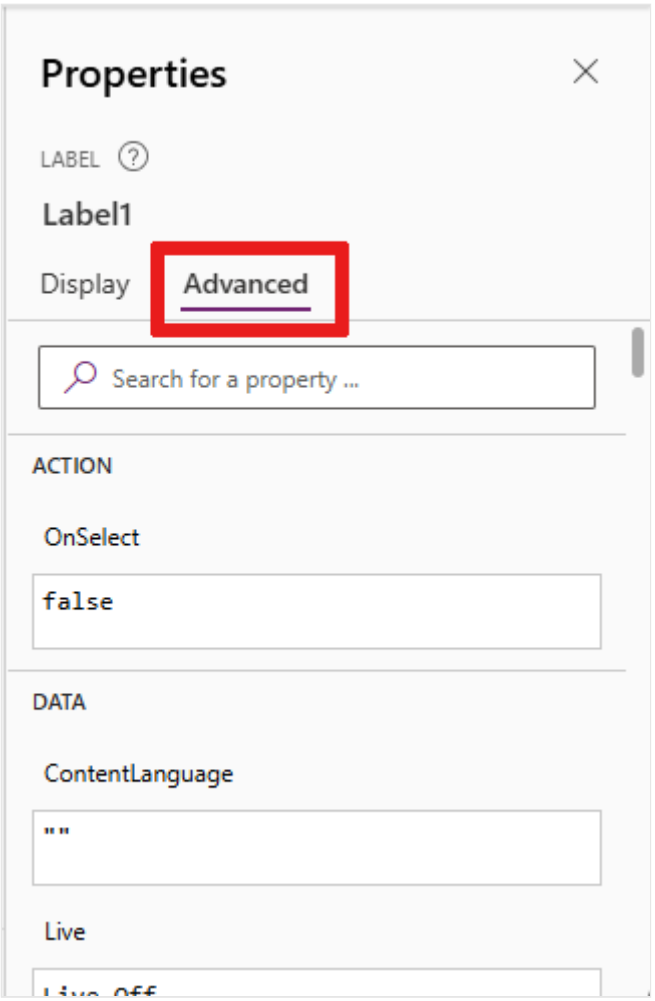
You can use formulas not only to perform calculations and change appearance but also to take action. For example, you can set the **OnSelect** property of a button to a formula that includes the **Navigate** function. When a user selects that button, the screen that you specify in the formula appears.

You can use some functions, such as **Navigate** and **Collect**, only in behavior formulas. The formula reference calls out if you can use a function only in this context.

You can take more than one action in a behavior formula if you separate functions with a semi-colon (;). For example, you might want to update a context variable, push data to a data source, and finally navigate to another screen.

View a list of properties by category

The properties list shows properties alphabetically, but you can also view all the properties of a control, organized by category, if you select a control, for example **Label1**, and then select the **Advanced** tab in the **Properties** pane, you see a long list of properties for that control.



You can edit formulas directly within this view. With the property search, you can quickly find a property of that control and change a control's behavior and appearance.

Formula syntax

As you type a formula in the formula bar, different syntax elements appear in different colors to improve readability and help you understand long formulas. Here's the color code list in Power Apps.

Function	Control	Global data source	Enumerated type
API reference	Keyword “ThisItem”	True, False	Number
String			

Related information

Use Find and Replace capability in the formula bar

Feedback

Was this page helpful? Yes No

Provide product feedback

Build an AI agent to automate your business process

08/14/2025

Agent builder in Power Apps lets organizations transition into the AI-first era by using the knowledge, logic, and actions of an app to create copilot agents. These agents handle daily tasks, streamline processes, enhance productivity, and improve overall business efficiency.

Makers can create agents that automate processes within their existing canvas apps. Agent builder uses the app's metadata and the desired agent goal to generate a comprehensive step-by-step process, extract knowledge, and identify triggers. This process, knowledge, and triggers are then combined with extracted skills from the app, resulting in a fully equipped copilot that offers detailed instructions, knowledge, triggers, and actions.

After creating the agent, makers can edit, test, and publish it in Microsoft Copilot Studio.

Prerequisites

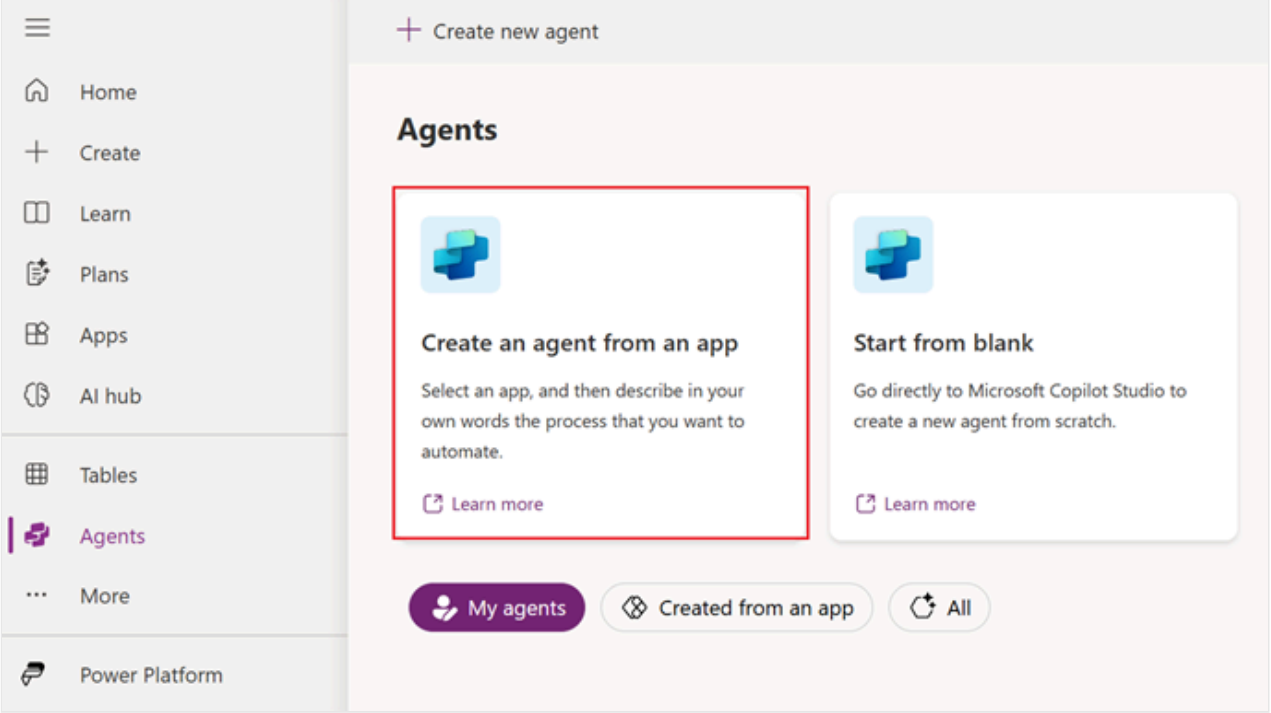
- Your tenant administrator must turn on the Publish Copilots with AI features setting in the Power Platform admin center.
- Include a Dataverse database in your environment. More information: Add a Microsoft Dataverse database.
- Ensure that block unmanaged customizations is disabled. Learn more in Block unmanaged customizations in Dataverse environments.
- Current environment has Power Virtual Agents solution version 2.0.0.649.1 or higher.

Create an agent

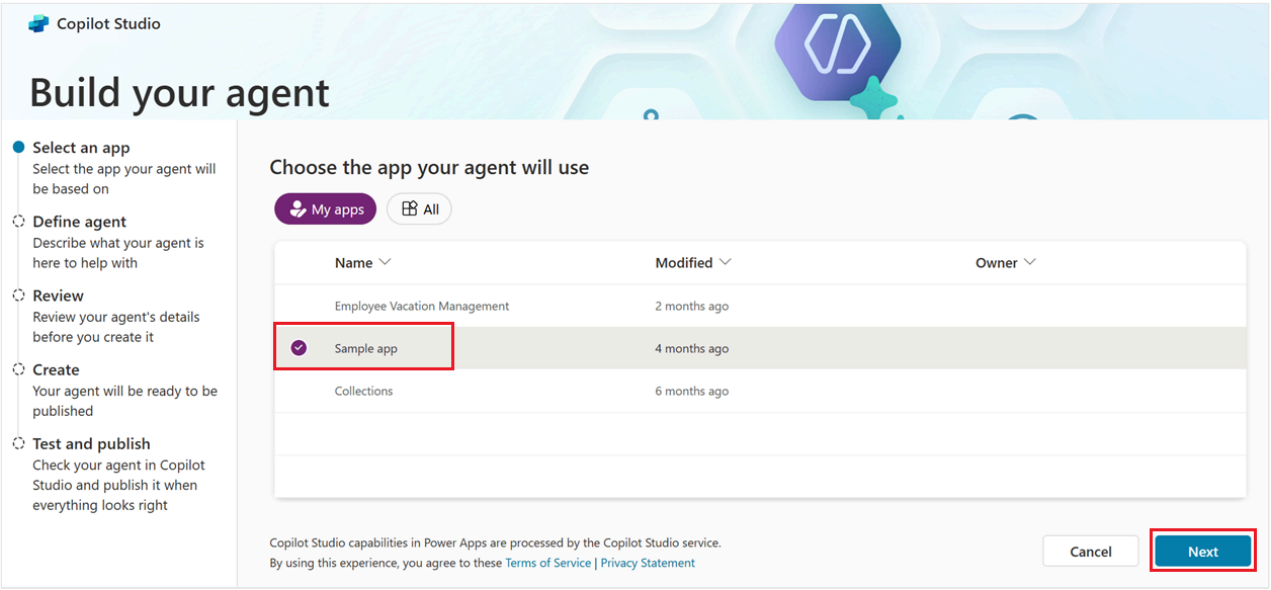
To streamline your manual process, generate an agent that replicates the steps you typically follow to complete tasks.

1. Sign in to Power Apps.
2. Select **Agents** in the left navigation pane. If you don't see **Agents**, select **More**, and then find and select **Agents**.

3. Select **Create an agent from an app**.

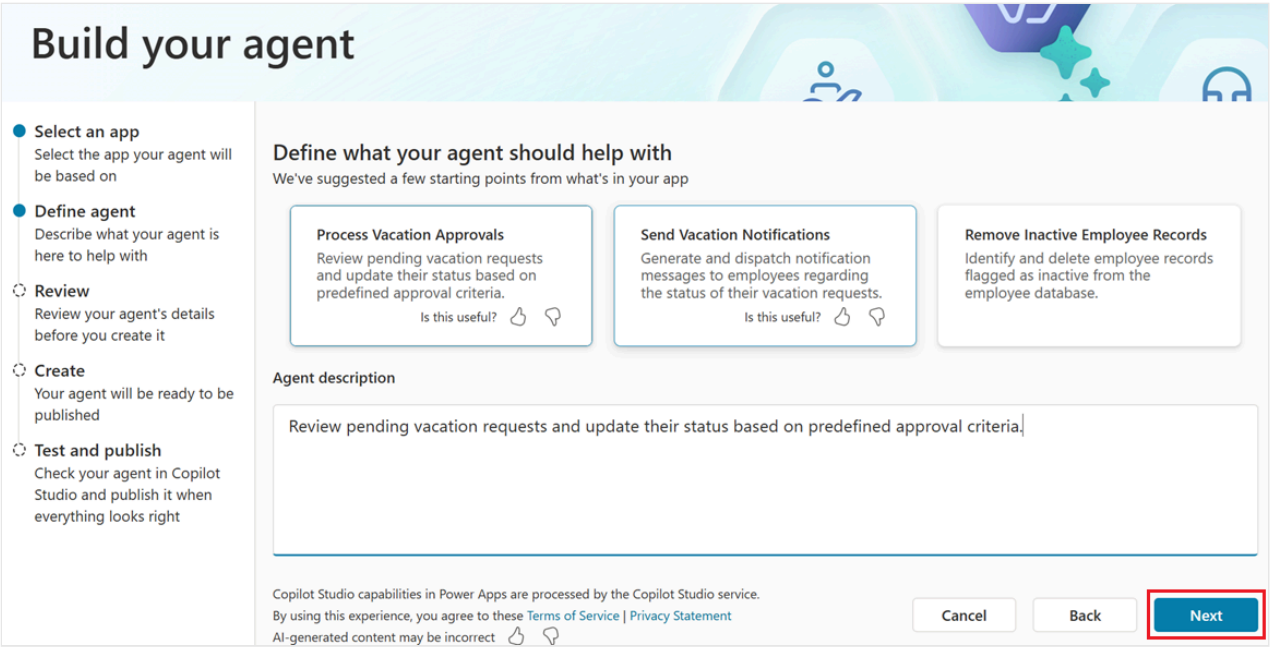


4. Select your app and then select **Next** on the command bar.



Alternatively, you can select **Apps** in the left navigation pane. Select your app, and then select **Create agent from app** on the command bar. You can also select **Commands** (⋮) for the app and then select **Create agent from app**.

5. Select a suggestion, or in the text box, describe the process you want to automate, and then select **Next**.



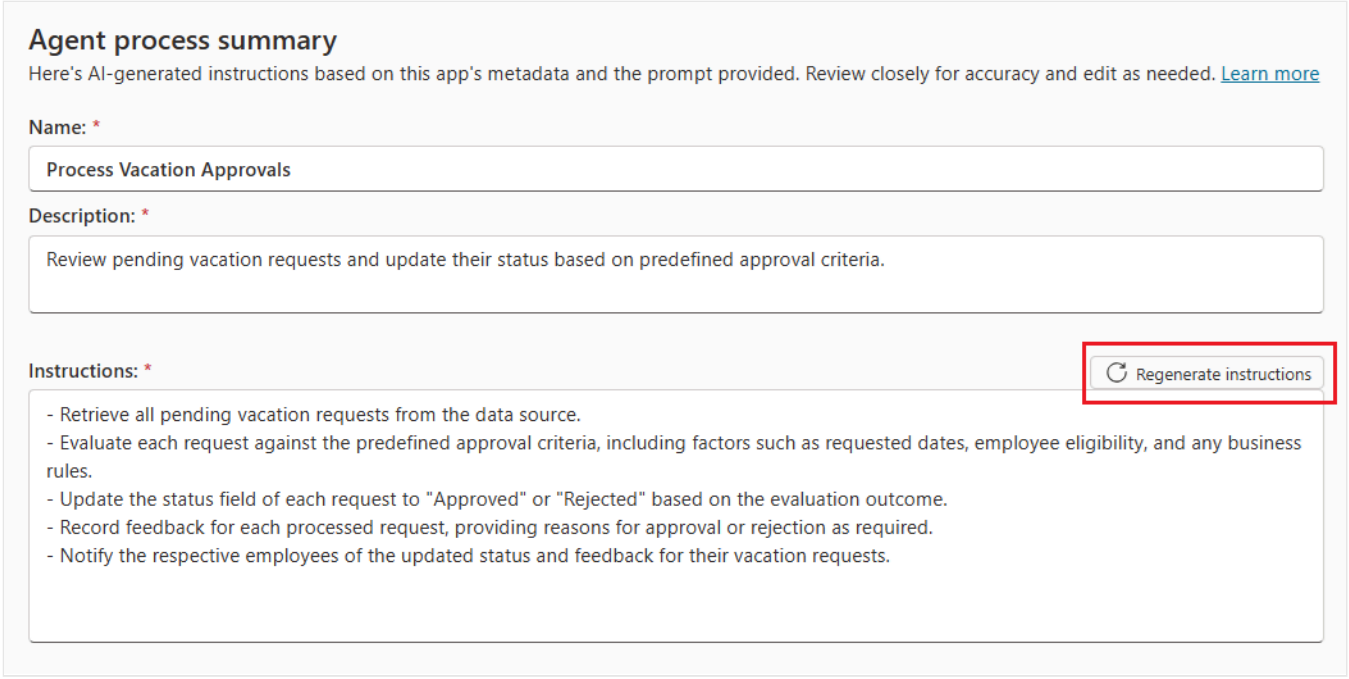
6. When you select a suggestion, the text box is automatically filled. You can edit it and add more details about what you want the agent to do. To improve the agent's accuracy, use simple, everyday language, and be specific, like:

    - Review pending vacation requests and update their status based on predefined approval criteria.
    - Generate and dispatch notification messages to employees regarding the status of their vacation requests.

7. Based on the suggestion or the information that you provide for the agent's goal and the app's metadata, agent builder generates step-by-step instructions to replace your manual process and extracts triggers and knowledge. Review the instructions for accuracy, and make any necessary edits. Then review the extracted knowledge and triggers for accuracy, and make any necessary edits. When you're done, select **Create**.

Regenerate instructions to improve the response

You can also revise the description to better represent the process you want to automate. When you're done, select the **Regenerate instructions** to receive updated instructions that match the new goal of the agent.



Best practices

When you edit the instructions, follow these suggestions:

- Use clear and specific wording in the instructions. For example, "Filter the claims data using the approval status" rather than "Filter the data."
- Verify the instructions to ensure they meet their goal.
- Keep each of the sentences for the instructions relatively simple. If there are too many details and conditions in one of the sentences, it might not work as intended.
- Ensure the logical flow of the instructions is easy to follow. Avoid adding new instructions at the end of the instruction set. Instead, move instructions in line to maintain a logical sequence.

  Here are example instructions:

  1. Read the data from the table.
  2. Process the data with pending status.
  3. If amount is less than $50 auto approve it and mark it as approved.
  4. Update the table with new status.
  5. And if the amount is more than $50 but less than $500 check to see if it's an approved item.
  6. Additionally if the amount is more than $500 update the status to manual review.

To improve the logical flow in this example, move steps 5 and 6 after step 3.

Limitations

- If a data source connection isn't found, you can't extract actions for the app. In such cases, the agent builder generates an agent with the actions that can be converted, or none if no actions can be converted.
- Deprecated Excel connectors aren't supported. However, the following Excel connectors are supported:
  - Excel Online (Business)
  - Excel Online
- Agent builder is available only in regions where Microsoft Copilot Studio generative agents are available.
- This feature is available only for canvas apps.

Provide feedback

Unless feedback is disabled by your admin, each step in the agent builder includes a **Like** (thumbs up) and **Dislike** button (thumbs down).

After reviewing the agent's responses, you can provide feedback in your own words. When you're done, select **Submit**.

Related information

FAQ for agent builder in canvas apps

Use Copilot to create and edit Power Fx formulas

Copilot is an AI feature in Power Apps Studio that helps you create and change Power Fx formulas quickly. Use Copilot in the formula bar to explain Power Fx formulas in everyday language or generate Power Fx formulas from natural language or code comments.

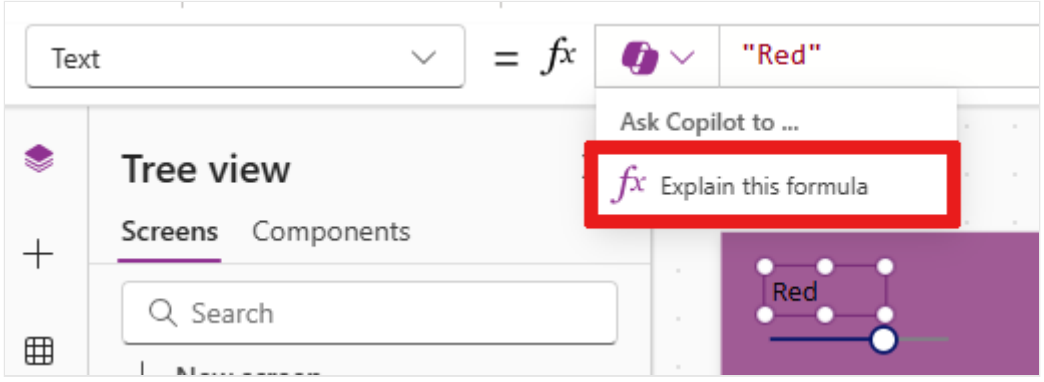
Prerequisites

An admin must enable Copilot for your environment and tenant. For more information, see Copilot in Power Apps overview.

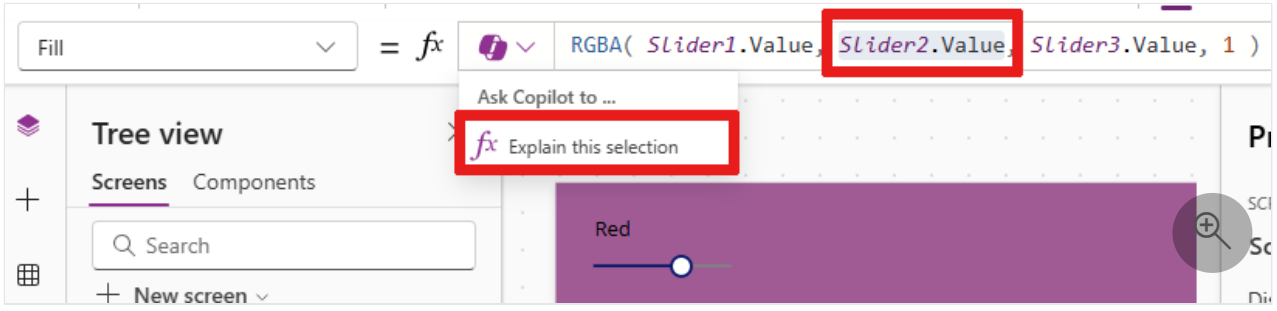
Explain a formula

Use Copilot in the formula bar to learn what a formula does in plain language.

1. Select a control and its property. For example, select a label control with its **Text** property.
2. On the formula bar, select the dropdown near the **fx** field label, and then select **Explain this formula**.

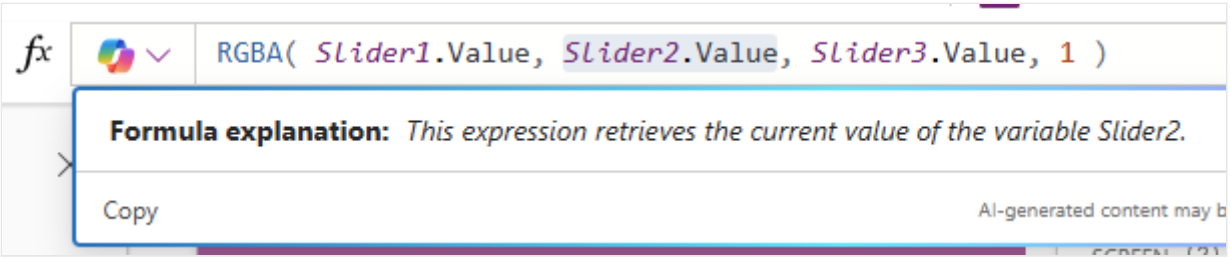


Alternatively, select part of a formula, select the dropdown near the **fx** field label, and then select **Explain this selection** to explain only that part.



Copilot gives you an explanation of the formula that you can copy and insert as a code comment, or share with other makers working on the same app. When Copilot explains a

selection, it can add context if that's important for the explanation.



Known limitations of explaining formulas

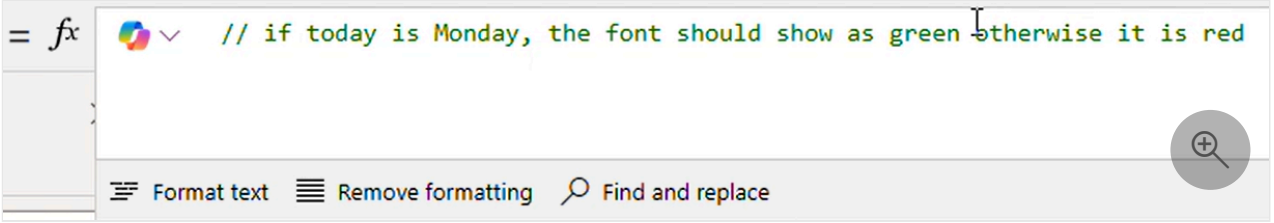
- The maximum length for a formula explanation is 5,000 characters. If the formula is longer, use a partial explanation to make sure it works.
- Copilot recognizes only default properties in Power Apps.
- Copilot doesn't recognize user-defined functions.

> **Note**
>
> AI-generated content can be inaccurate, so check it for accuracy.

Generate formulas from code comments

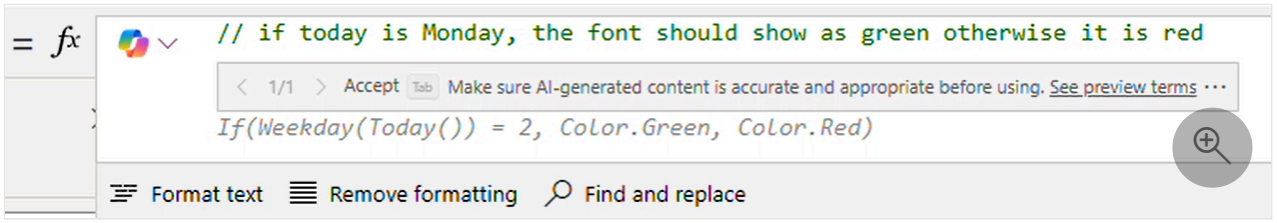
Copilot can generate Power Fx formulas from code comments in the formula bar. You can accept the suggestion or type the formula yourself. Formula code comments can also serve as documentation for your formulas, just like regular code comments.

1. With your canvas app open for editing, type a code comment using `//` or `/*` in the formula bar.



   Wait a few seconds for Copilot to generate a formula suggestion or press `Enter` to generate it immediately.

2. Press `Tab` on the keyboard to accept and use the suggested formula. You can also type through the suggestion.



The suggestion remains until you select elsewhere or type a character that doesn't match the suggestion.

You can keep the comments used for generating Power Fx formulas in the formula bar as documentation, similar to traditional code comments.

> **Note**
>
> - If Copilot doesn't provide a suggestion, then it couldn't generate a formula based on your comment. Try rephrasing the comment.
> - We recommend enclosing in quotes the elements that you want to include in your formula, such as `"Button1.text"`.

Known limitations of generating formulas from code comments

- Copilot works on a singular control and property, and it doesn't make changes to other controls or properties to achieve the desired outcome.
- Copilot doesn't take existing formula text into account.
- Copilot only recognizes default properties in Power Apps.
- Copilot doesn't recognize user-defined functions.
- Code comments only work with general Power Fx functions, and not Power Apps-specific functions such as `Navigate()`.
- The advanced panel doesn't trigger suggestions.
- Existing formulas for the property aren't included in the suggestion.

Create a formula (preview)

[This section is prerelease documentation and is subject to change.]

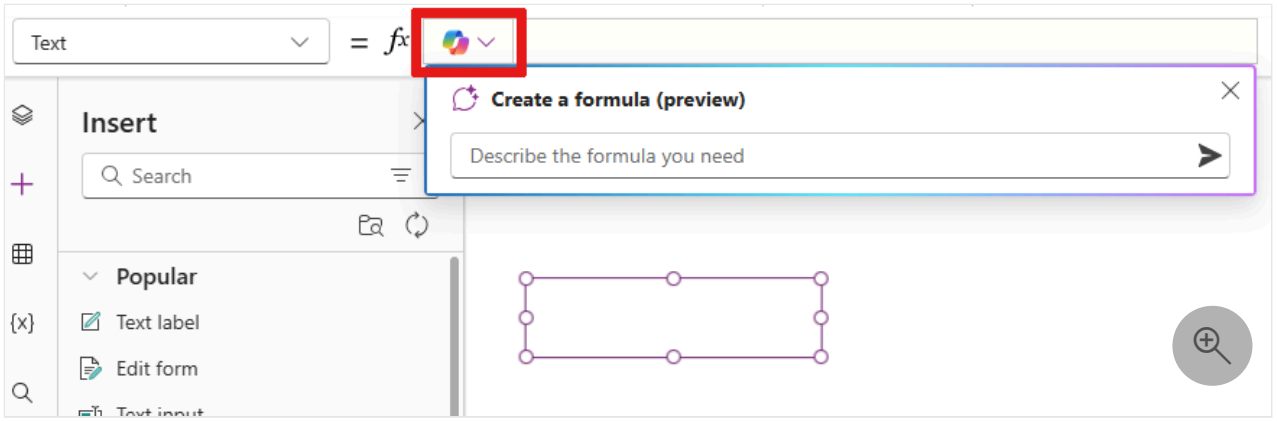
From the Copilot dropdown menu in the formula bar, you can now generate a formula by using natural language to make a discrete request.

> **Note**

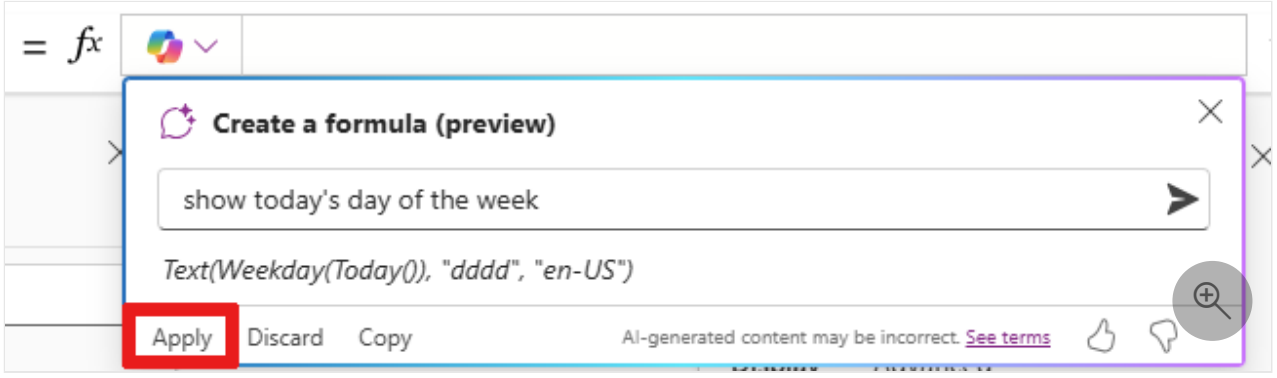
The **Copilot for formulas** feature is on by default. To turn it off, open your app, go to **Settings** > **Updates** > **Preview**, and find the **Copilot for formulas** toggle. If you turn off the setting, the **Create a formula (preview)** menu option disappears from the Copilot dropdown menu in the formula bar.

Use Copilot to create a formula from a discrete request

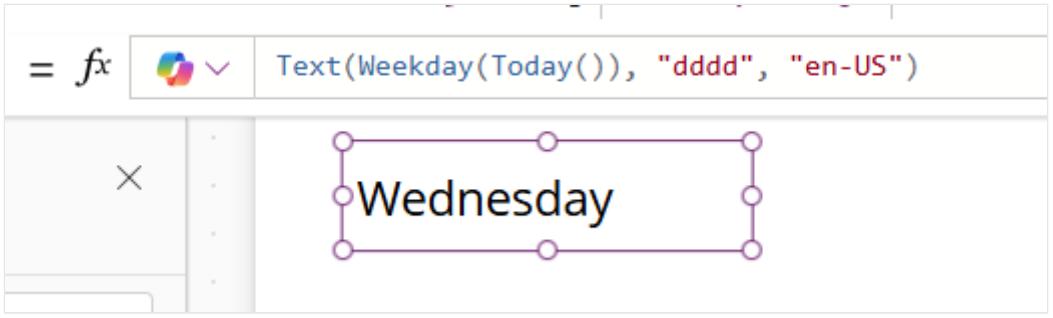
1. Select a control, like a **Text label**, and its corresponding property, such as **Text**.
2. On the formula bar, select the **Copilot functionality menu** > **Create a formula**.



3. In the Copilot text box, type your request, such as *show today's day of the week*.
4. Select the **Create** arrow to submit your request.
5. Copilot attempts to create a formula. Select **Apply** to insert the formula in the same position as the Copilot interface, or update and resubmit your request.



You see the result after you apply the formula.



> [!NOTE]
> The same limitations apply for creating formulas from a discrete request as generating formulas from code comments.

Responsible AI

Understand the choices you have when working with AI. Learn more in FAQ about using AI responsibly in Power Apps. Review the Power Fx Copilot FAQ for questions about this feature specifically.

Language support

The following table lists the languages available for the formula bar copilot features. The list is based on the settings in your Power Platform environment and browser.

Expand table

Name	Language Code	Functionality Supported
English	en-US	Explain a formula, Comment generated formulas, Create a formula
Chinese (simplified) - China	zh-Hans	Explain a formula, Comment generated formulas
Czech - Czech Republic	cs-CZ	Explain a formula, Comment generated formulas
Danish - Denmark	da-DK	Explain a formula, Comment generated formulas
Dutch - Netherlands	nl-NL	Explain a formula, Comment generated formulas
Finnish - Finland	fi-FI	Explain a formula, Comment generated formulas
French - France	fr-FR	Explain a formula, Comment generated formulas

Name	Language Code	Functionality Supported
German - Germany	de-DE	Explain a formula, Comment generated formulas
Greek - Greece	el-GR	Explain a formula, Comment generated formulas
Italian - Italy	it-IT	Explain a formula, Comment generated formulas
Japanese - Japan	ja-JP	Explain a formula, Comment generated formulas
Korean - Korea	ko-KR	Explain a formula, Comment generated formulas
Norwegian (Bokmål) - Norway	nb-NO	Explain a formula, Comment generated formulas
Polish - Poland	pl-PL	Explain a formula, Comment generated formulas
Portuguese - Brazil	pt-BR	Explain a formula, Comment generated formulas
Russian - Russia	ru-RU	Explain a formula, Comment generated formulas
Spanish (Traditional Sort) - Spain	es-ES	Explain a formula, Comment generated formulas
Swedish - Sweden	sv-SE	Explain a formula, Comment generated formulas
Thai - Thailand	th-TH	Explain a formula, Comment generated formulas
Turkish - Türkiye	tr-TR	Explain a formula, Comment generated formulas

Related information

Copilot comment generated formulas (video)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Last updated on 01/12/2026

Understand variables in canvas apps

08/27/2025

If you've used another programming tool, like Visual Basic or JavaScript, you might ask: **Where are the variables?** Power Apps works differently and needs a different approach. Instead of reaching for a variable when you build a canvas app, ask yourself: **What would I do in Excel?**

In other tools, you might explicitly perform a calculation and store the result in a variable. But Power Apps and Excel automatically recalculate formulas as the input data changes, so you usually don't need to create or update variables. By using this approach, you more easily create, understand, and maintain your app.

Sometimes, you need to use variables in Power Apps, which extends Excel's model by adding behavior formulas. These formulas run when, for example, a user selects a button. Within a behavior formula, it's often helpful to set a variable for use in other formulas.

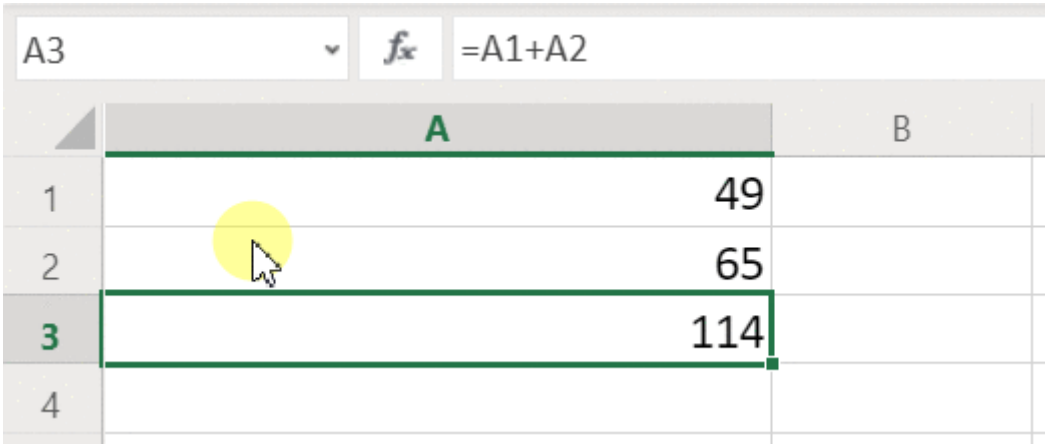
In general, avoid using variables. But sometimes only a variable lets you create the experience you want. Variables are implicitly created and typed when they appear in functions that set their values.

Translate Excel into Power Apps

Excel

Let's review how Excel works. A cell can have a value, like a number or a string, or a formula that's based on the values of other cells. When you enter a different value into a cell, Excel automatically recalculates any formulas that depend on the new value. You don't need to do any programming to enable this behavior.

In the following example, cell **A3** uses the formula **A1+A2**. If **A1** or **A2** changes, **A3** automatically recalculates to show the change. This behavior doesn't require any coding outside of the formula itself.

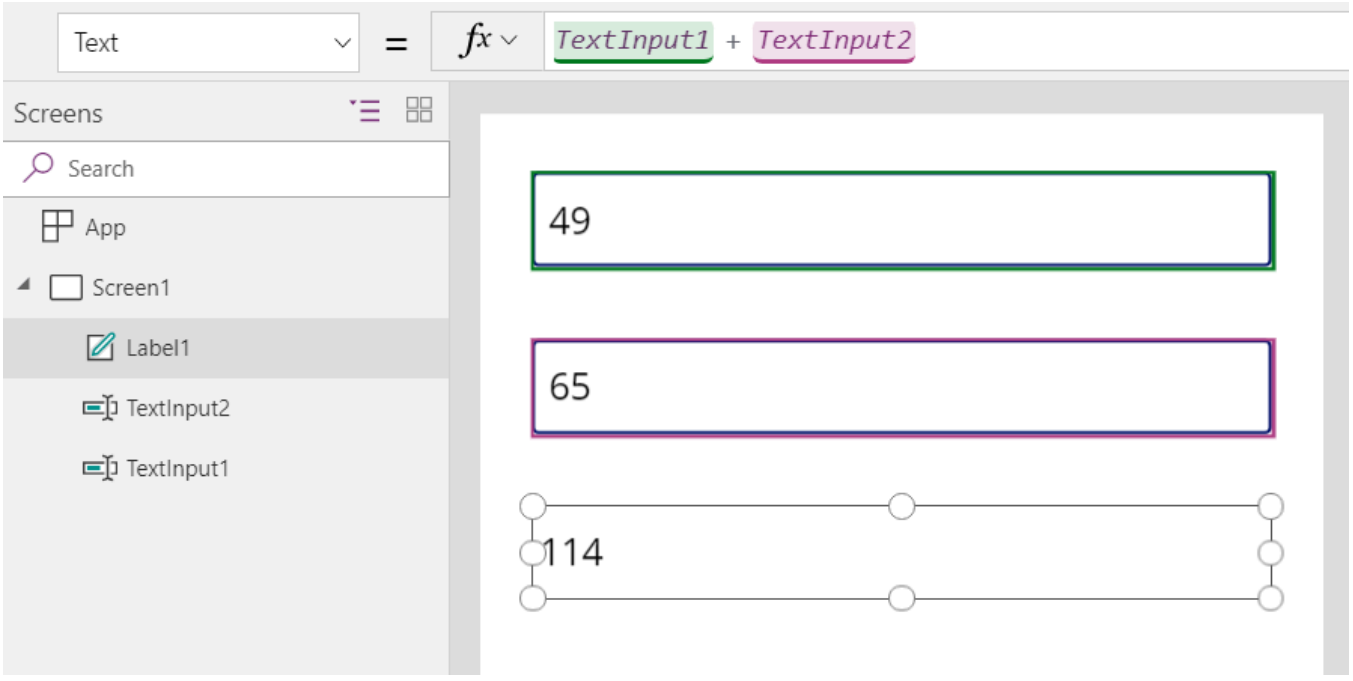


Excel doesn't have variables. The value of a cell with a formula changes based on its input, but there's no way to remember the result of a formula and store it in a cell or anywhere else. If you change a cell's value, the entire spreadsheet can change, and any previously calculated values are lost. You can copy and paste cells, but that's manual and isn't possible with formulas.

Power Apps

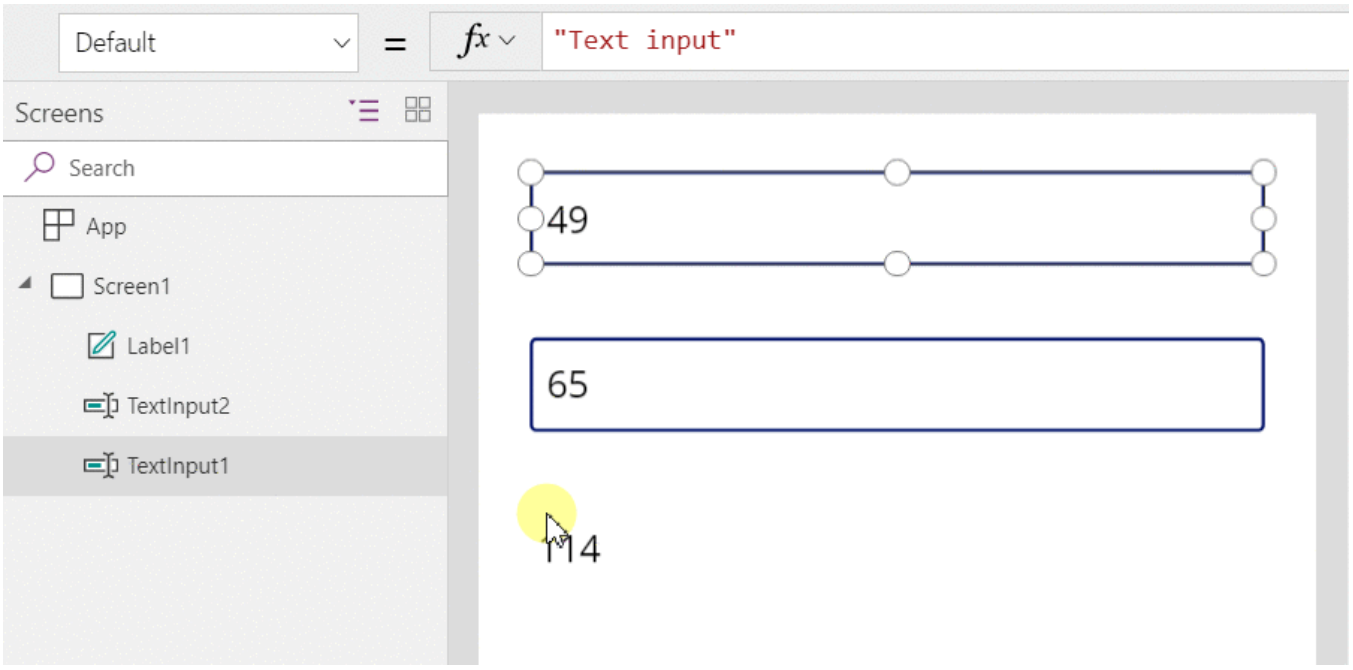
Apps you create in Power Apps work a lot like Excel. Instead of updating cells, you add controls anywhere on a screen and name them for use in formulas.

For example, you can copy the Excel behavior in an app by adding a **Label** control named **Label1**, and two **Text input** controls named **TextInput1** and **TextInput2**. If you set the **Text** property of **Label1** to **TextInput1** + **TextInput2**, it always shows the sum of the numbers in **TextInput1** and **TextInput2** automatically.



Notice that the **Label1** control is selected, showing its **Text** formula in the formula bar at the top of the screen. The formula **TextInput1** + **TextInput2** creates a dependency between these

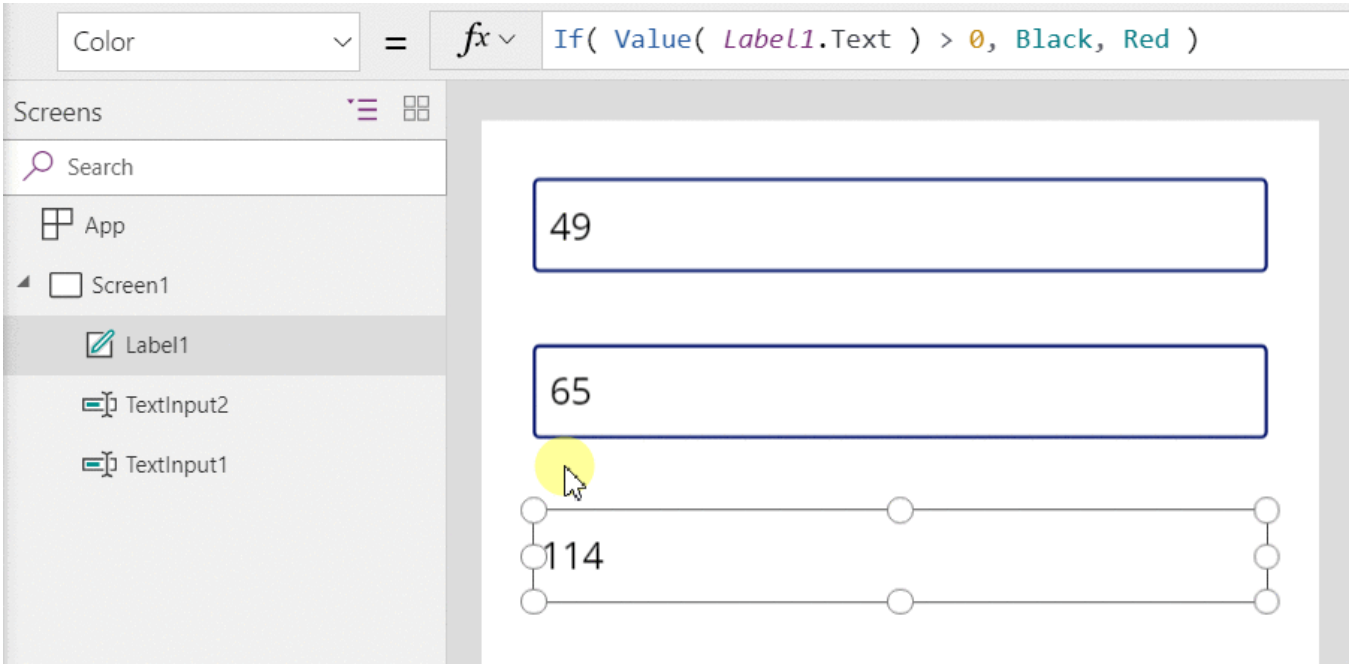
controls, just like dependencies between cells in an Excel workbook. Let's change the value of **TextInput1**:



The formula for **Label1** automatically recalculates and shows the new value.

In Power Apps, you use formulas to set not only the main value of a control but also properties like formatting. In the next example, a formula for the **Color** property of the label automatically shows negative values in red. The **If** function looks familiar from Excel:

```
If( Value(Label1.Text) < 0, Red, Black )
```



You use formulas for many scenarios:

- When your device's GPS is on, a map control shows your current location with a formula that uses **Location.Latitude** and **Location.Longitude**. As you move, the map automatically

tracks your location.
- Other users update data sources. For example, others on your team might update items in a list. When you refresh a data source, any dependent formulas automatically recalculate to show the updated data. For example, you might set a gallery's **Items** property to the formula **Filter(SharePointList)**, which automatically shows the newly filtered set of records.

Benefits

Using formulas to build apps has many advantages:

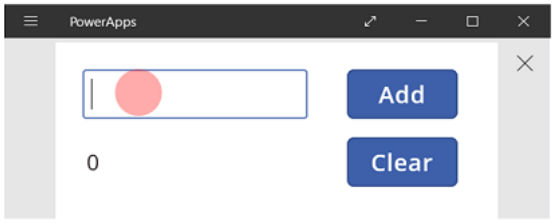
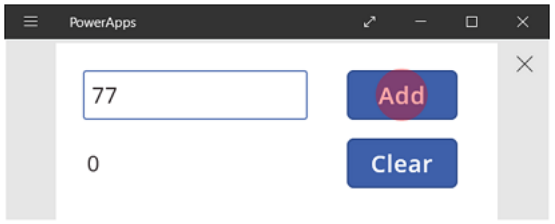
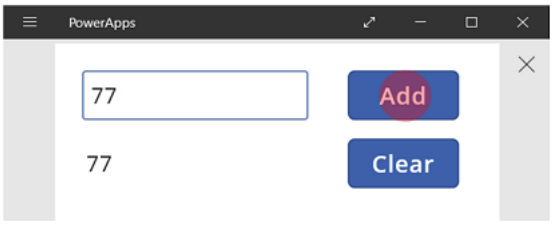
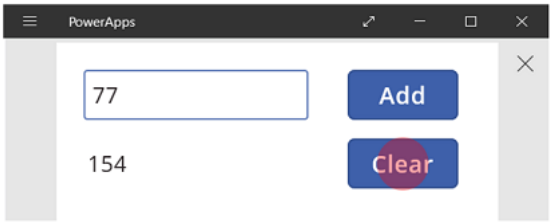
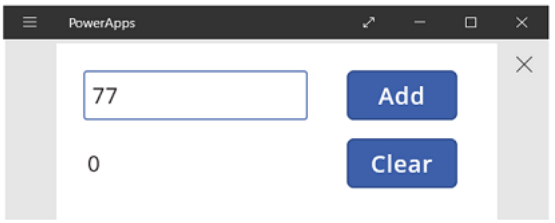
- If you know Excel, you know Power Apps. The model and formula language are the same.
- If you've used other programming tools, think about how much code you'd need to do these examples. In Visual Basic, you'd write an event handler for the change event on each text-input control. The code to do the calculation in each is redundant and can get out of sync, or you'd need to write a common subroutine. In Power Apps, you do all of that with a single, one-line formula.
- To see where **Label1**'s text comes from, you know exactly where to look: the formula in the **Text** property. There's no other way to change the text of this control. In a traditional programming tool, any event handler or subroutine can change the value of the label, from anywhere in the program. This makes it hard to track down when and where a variable changed.
- If you change a slider control and then change your mind, you can change the slider back to its original value. It's as if nothing ever changed: the app shows the same control values as before. There are no consequences for experimenting and asking "what if," just like in Excel.

In general, if you can do something with a formula, you're better off. Let the formula engine in Power Apps work for you.

Know when to use variables

Let's change our simple adder to act like an old-fashioned adding machine with a running total. When you select the **Add** button, you add a number to the running total. When you select the **Clear** button, you reset the running total to zero.

Expand table

Display	Description
	When the app starts, the running total is 0. The red dot represents the user's finger in the text-input box, where the user enters **77**.
	The user selects the **Add** button.
	77 is added to the running total. The user selects the **Add** button again.
	77 is again added to the running total, resulting in 154. The user selects the **Clear** button.
	The running total is reset to 0.

Our adding machine uses something that doesn't exist in Excel: a button. In this app, you can't use only formulas to calculate the running total because its value depends on a series of actions that the user takes. Instead, you record and update the running total manually. Most programming tools store this information in a *variable*.

Sometimes you need a variable for your app to behave the way you want, but this approach comes with caveats:

- You manually update the running total. Automatic recalculation doesn't do it for you.
- The running total isn't calculated based on the values of other controls. It depends on how many times the user selects the **Add** button and what value is in the text-input control each time. Did the user enter 77 and select **Add** twice, or did they enter 24 and 130 for each addition? You can't tell the difference after the total reaches 154.

- Changes to the total can come from different actions. In this example, both the **Add** and **Clear** buttons update the total. If the app doesn't behave the way you expect, which button causes the problem?

Use a global variable

To create an adding machine, you need a variable to hold the running total. The simplest variables to use in Power Apps are *global variables*.

How global variables work:

- Set the value of a global variable with the Set function. **Set(MyVar, 1)** sets the global variable **MyVar** to a value of **1**.
- Use the global variable by referencing the name used with the **Set** function. In this case, **MyVar** returns **1**.
- Global variables can hold any value, including strings, numbers, records, and tables.

Rebuild the adding machine by using a global variable:

1. Add a text-input control named **TextInput1**, and two buttons named **Button1** and **Button2**.

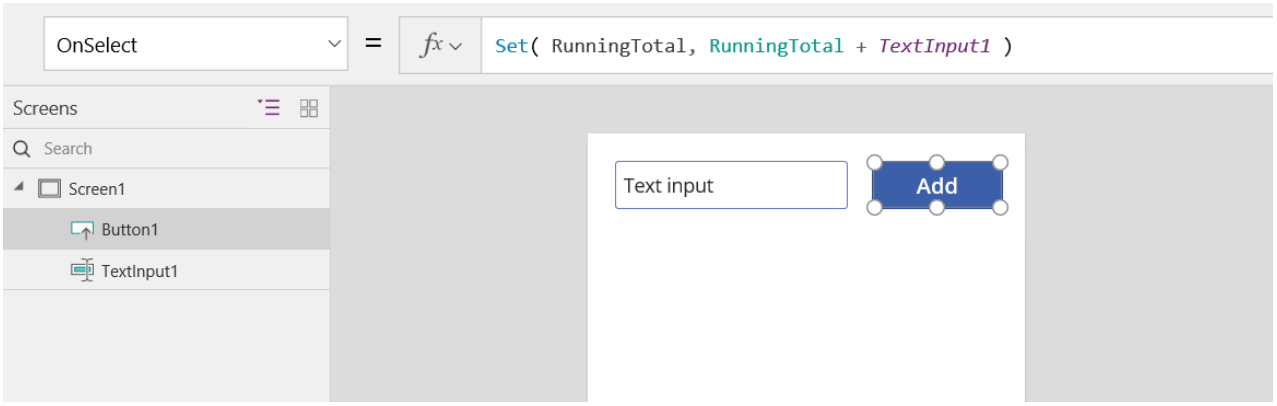
2. Set the Text property of **Button1** to **"Add"**, and set the **Text** property of **Button2** to **"Clear"**.

3. To update the running total when a user selects the **Add** button, set its OnSelect property to this formula:

   Set(RunningTotal, RunningTotal + TextInput1)

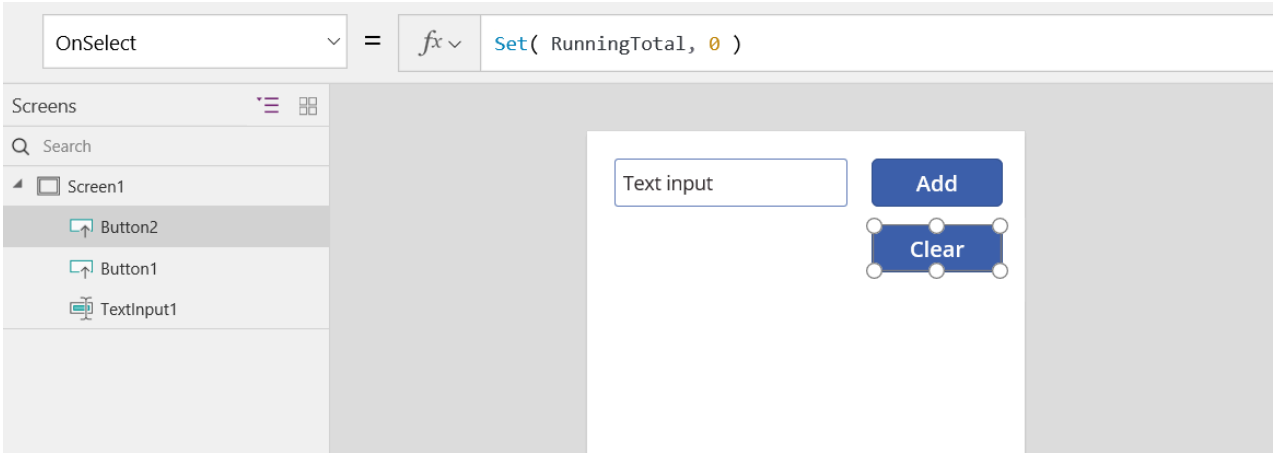
   This formula creates **RunningTotal** as a global variable that holds a number because of the + operator. Reference **RunningTotal** anywhere in the app. When the user opens the app, **RunningTotal** has an initial value of *blank*.

   The first time a user selects the **Add** button and Set runs, **RunningTotal** is set to the value **RunningTotal** + **TextInput1**.



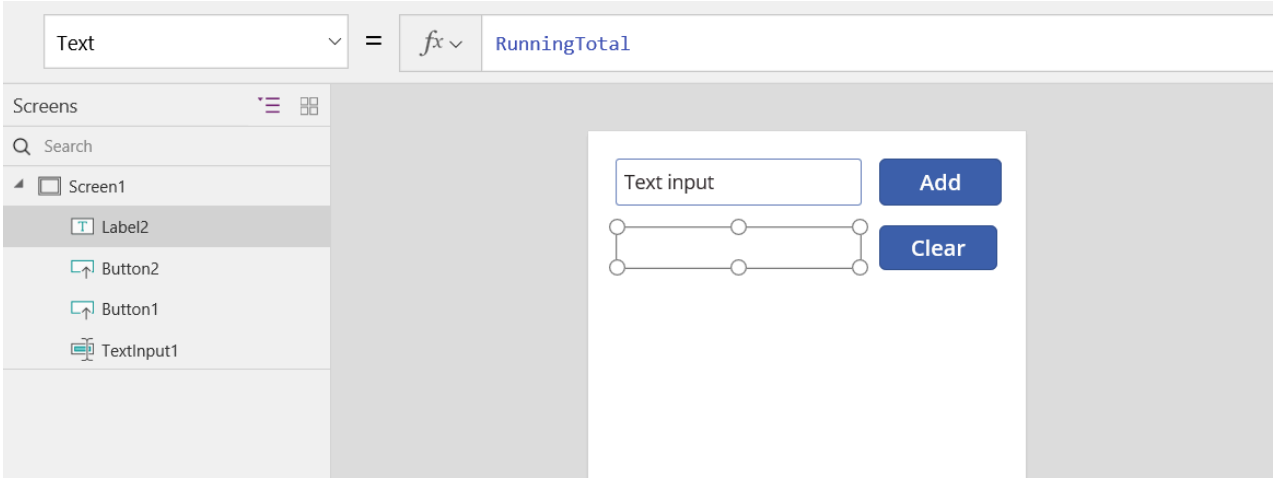
4. To set the running total to **0** when the user selects the **Clear** button, set its **OnSelect** property to this formula:

   Set(RunningTotal, 0)

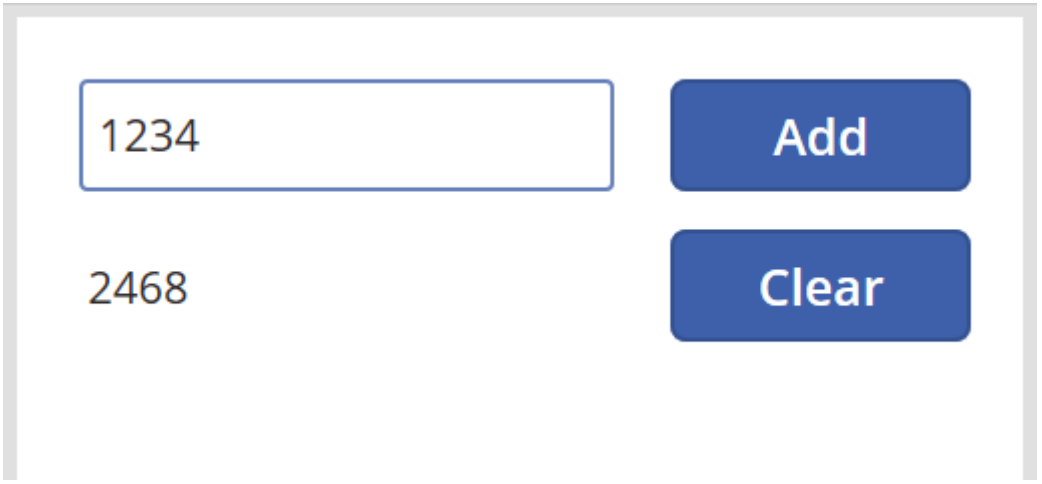


5. Add a **Label** control, and set its **Text** property to **RunningTotal**.

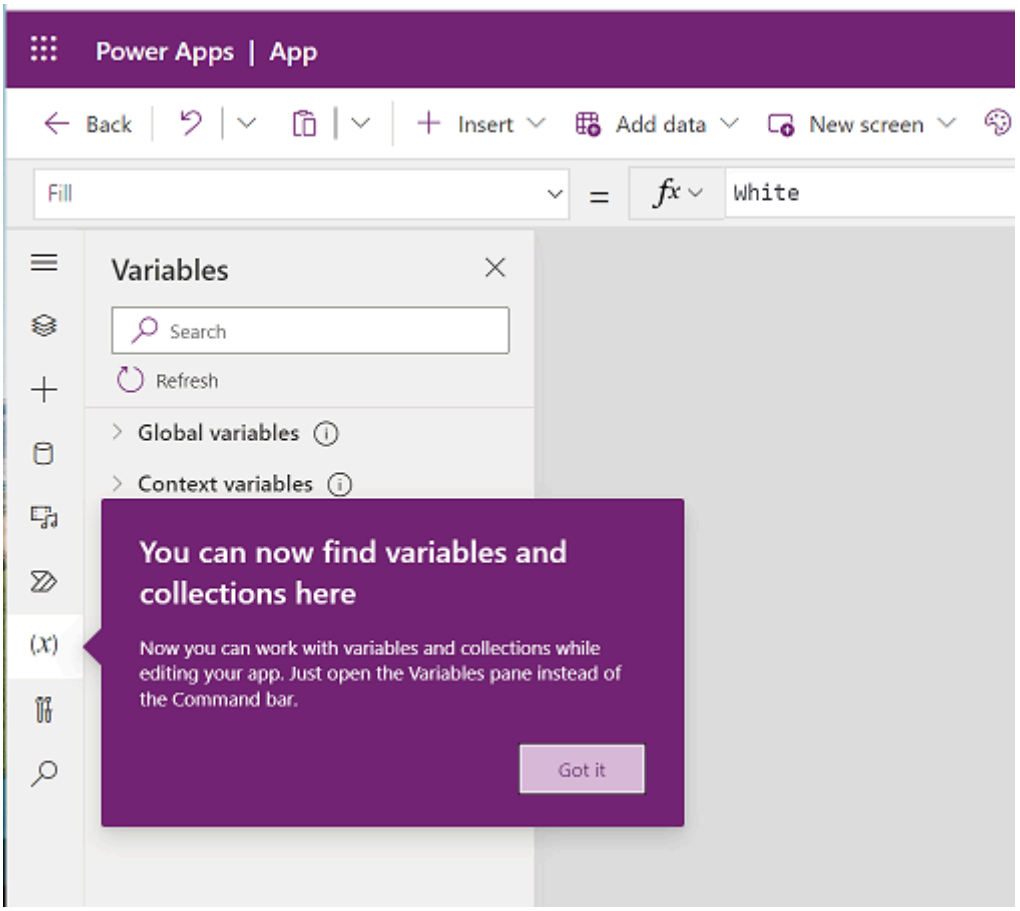
   This formula automatically recalculates and shows the user the value of **RunningTotal** as it changes based on the buttons the user selects.



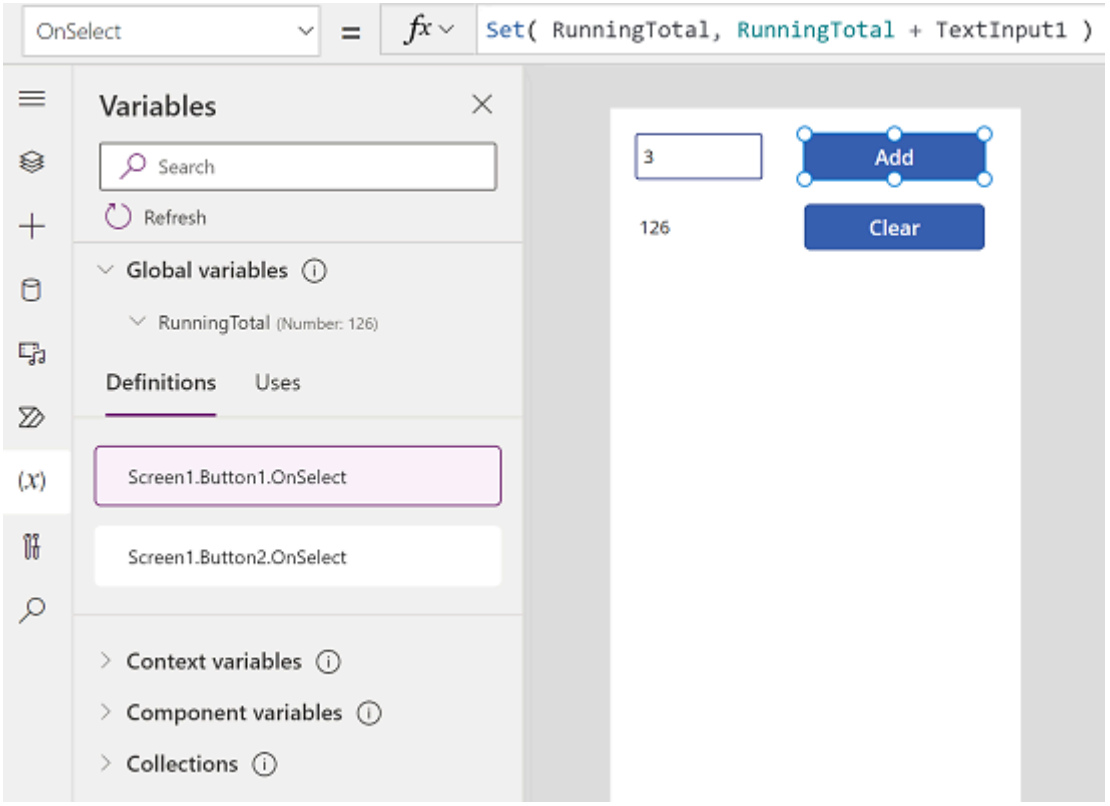
6. Preview the app. Enter a number in the text box and select the **Add** button a few times. When ready, return to the authoring experience using the `Esc` key.



7. To show the global variable's value, select **Variables** on the app authoring menu.



8. To show all the places where the variable is defined and used, select it under **Global variables**.



Types of variables

Power Apps has three types of variables:

Expand table

Variable type	Scope	Description	Functions that establish
Global variables	App	Use global variables to hold a number, text string, Boolean, record, table, or other value that you reference anywhere in the app.	Set
Context variables	Screen	Use context variables to pass values to a screen, like parameters in other languages. You reference them from only one screen.	UpdateContext Navigate
Collections	App	Use collections to hold a table that you reference anywhere in the app. You can change the contents of the table, and save it to the local device for later use.	Collect ClearCollect

Create and remove variables

Variables are created implicitly when they appear in a **Set**, **UpdateContext**, **Navigate**, **Collect**, or **ClearCollect** function. To declare a variable and its type, just include it in any of these functions anywhere in your app. These functions don't create variables; they only fill variables

with values. You never declare variables explicitly like you might in another programming tool, and all typing is implicit from usage.

For example, you might have a button control with an **OnSelect** formula equal to **Set(X, 1)**. This formula establishes **X** as a variable with a type of number. You use **X** in formulas as a number, and that variable has a value of *blank* after you open the app but before you select the button. When you select the button, you give **X** the value of **1**.

If you add another button and set its **OnSelect** property to **Set(X, "Hello")**, an error occurs because the type (text string) doesn't match the type in the previous **Set** (number). All implicit definitions of the variable must agree on type. Again, all this happens because you mention **X** in formulas, not because any of those formulas actually run.

Remove a variable by removing all the **Set**, **UpdateContext**, **Navigate**, **Collect**, or **ClearCollect** functions that implicitly establish the variable. Without these functions, the variable doesn't exist. Also remove any references to the variable because they cause an error.

Variable lifetime and initial value

Variables stay in memory while the app runs. When the app closes, the values in variables are lost.

Store the contents of a variable in a data source by using the **Patch** or **Collect** functions. Store values in collections on the local device by using the SaveData function.

When the user opens the app, all variables start with an initial value of *blank*.

Reading variables

Use the variable's name to read its value. For example, define a variable with this formula:

```
Set( Radius, 12 )
```

Then use **Radius** anywhere you use a number, and it's replaced with **12**:

```
Pi() * Power( Radius, 2 )
```

If you give a context variable the same name as a global variable or a collection, the context variable takes precedence. You can reference the global variable or collection by using the disambiguation operator **[@Radius]**.

Reserved variable names

Don't use any of the following variable names. These names are app properties, and using them can cause variable name collisions that result in unexpected app behavior.

- ActiveScreen
- DesignHeight
- DesignWidth
- Height
- MinScreenHeight
- MinScreenWidth
- SizeBreakpoints
- StudioVersion
- TestCaseId
- Testing
- TestSuiteId
- Theme
- Width

Use a context variable

Let's look at how you create an adding machine by using a context variable instead of a global variable.

How context variables work

- You implicitly establish and set context variables by using the **UpdateContext** or **Navigate** function. When the app starts, the initial value of each context variable is *blank*.
- You update context variables with records. In other programming tools, you usually use "=" for assignment, like "x = 1". For context variables, use **{ x: 1 }** instead. When you use a context variable, use its name directly without the record syntax.
- You can also set a context variable when you use the **Navigate** function to show a screen. If you think of a screen as a kind of procedure or subroutine, this approach is like parameter passing in other programming tools.
- Except for **Navigate**, context variables are limited to the context of a single screen, which is where they get their name. You can't use or set them outside this context.
- Context variables can hold any value, including strings, numbers, records, and tables.

Rebuild the adding machine by using a context variable:

1. Add a text-input control, named **TextInput1**, and two buttons, named **Button1** and **Button2**.

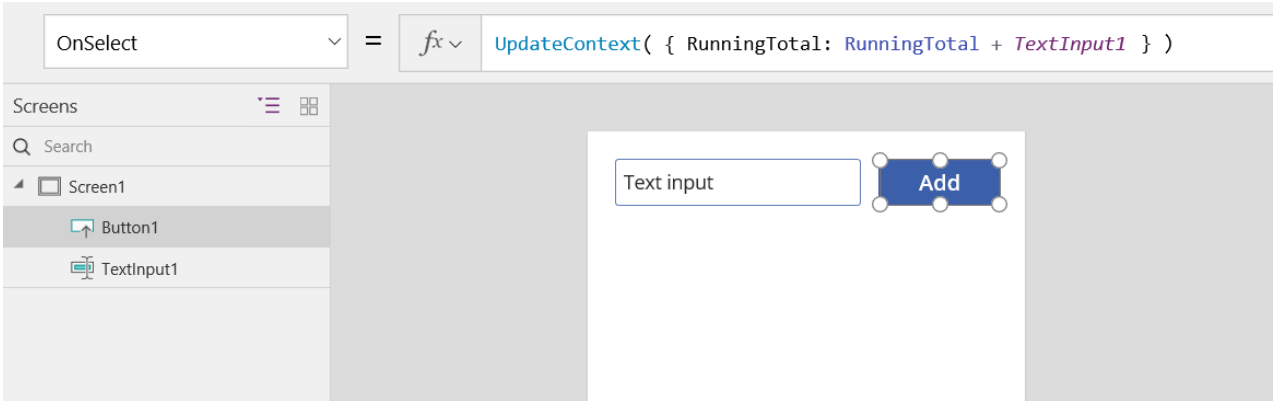
2. Set the Text property of **Button1** to "**Add**", and set the **Text** property of **Button2** to "**Clear**".

3. To update the running total whenever a user selects the **Add** button, set its OnSelect property to this formula:

   UpdateContext({ RunningTotal: RunningTotal + TextInput1 })

   This formula establishes **RunningTotal** as a context variable that holds a number because of the + operator. You reference **RunningTotal** anywhere in this screen. When the user opens the app, **RunningTotal** has an initial value of *blank*.

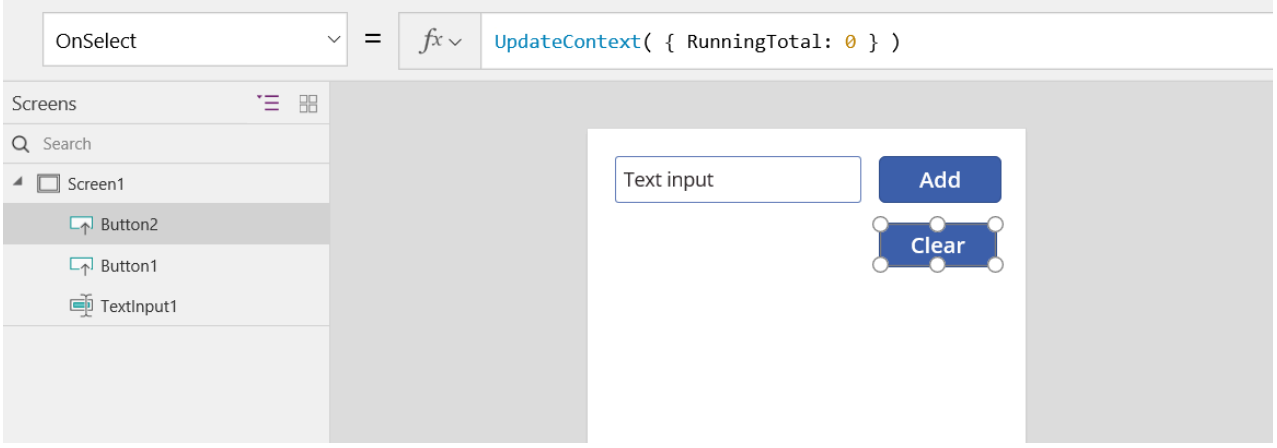
   The first time the user selects the **Add** button and UpdateContext runs, **RunningTotal** is set to the value **RunningTotal + TextInput1**.



4. To set the running total to **0** whenever the user selects the **Clear** button, set its OnSelect property to this formula:

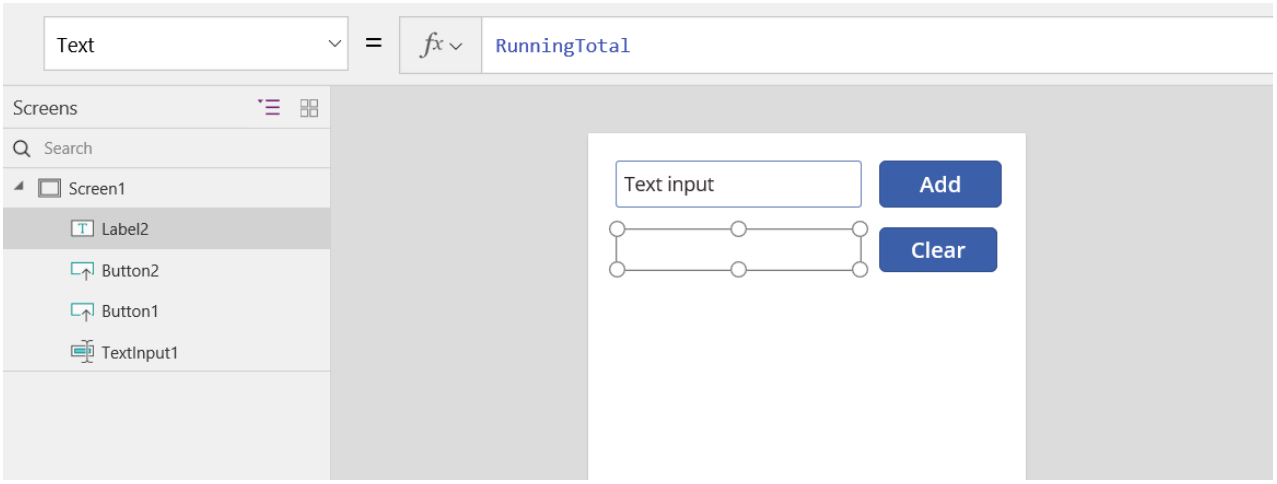
   UpdateContext({ RunningTotal: 0 })

   Again, use UpdateContext with the formula **UpdateContext({ RunningTotal: 0 })**.

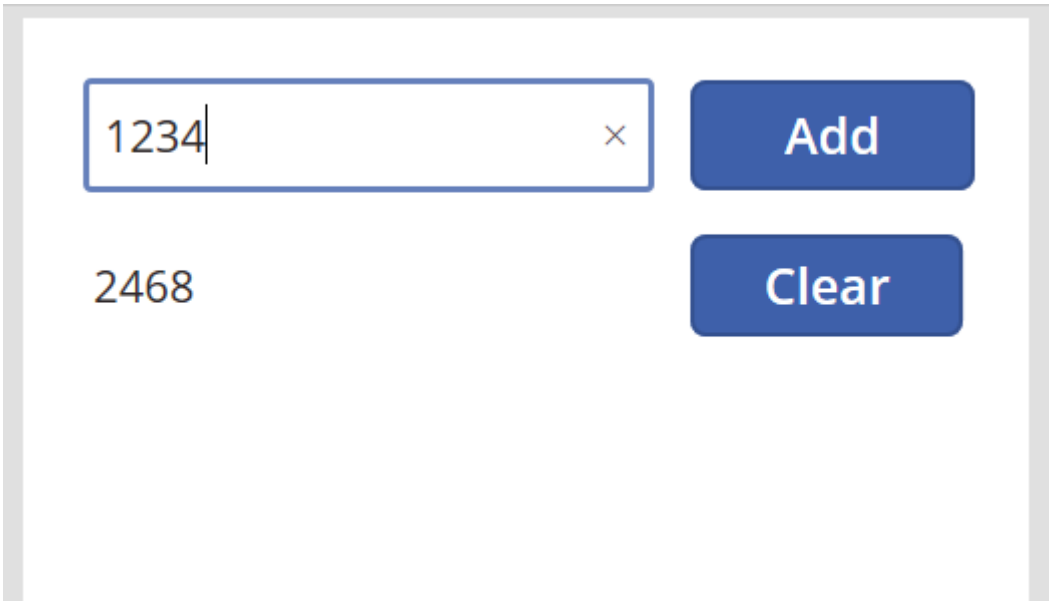


5. Add a Label control, and set its Text property to **RunningTotal**.

   This formula will automatically be recalculated and show the user the value of **RunningTotal** as it changes based on the buttons that the user selects.

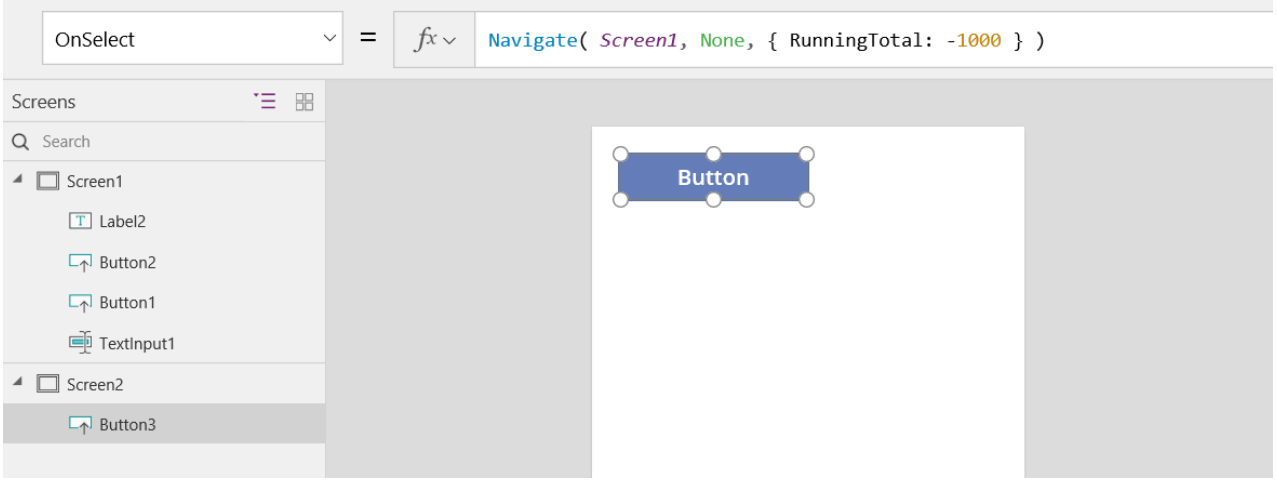


6. Preview the app. Enter a number in the text box and select the **Add** button a few times. When you're ready, return to the authoring experience by using the `Esc` key.

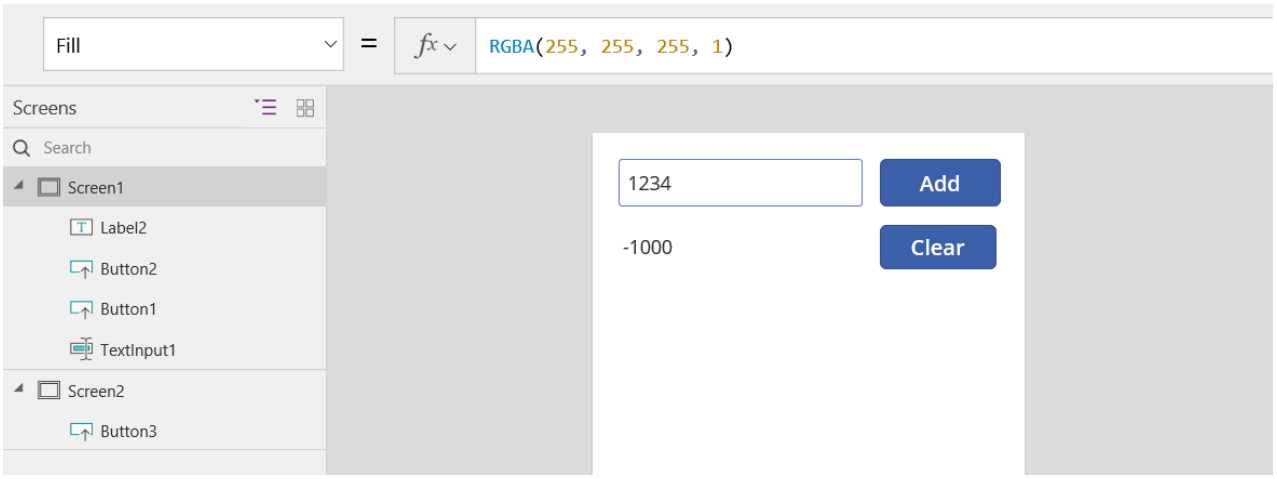


7. Set the value of a context variable while navigating to a screen to pass "context" or "parameters" from one screen to another. To demonstrate this technique, insert a screen, insert a button, and set its **OnSelect** property to this formula:

   Navigate(Screen1, None, { RunningTotal: -1000 })



Hold down the `Alt` key while you select this button to show **Screen1** and set the context variable **RunningTotal** to -1000.



8. To show the value of the context variable, do one of the following:

   - If you're using the preview version of Power Apps Studio, select **Variables** in the app authoring menu.
   - If you're using the classic version of Power Apps Studio, select the **File** menu, and then select **Variables** in the left pane.

9. To show where the context variable is defined and used, select it under **Context variables**.

Use a collection

Let's create an adding machine by using a collection. Because a collection holds a table that's easy to change, this adding machine keeps a "paper tape" of each value as you enter it.

How collections work

- Create and set collections by using the ClearCollect function. You can use the Collect function instead, but it requires another variable instead of replacing the old one.
- A collection is a type of data source and a table. To get a single value in a collection, use the First function, and get one field from the resulting record. If you use a single value with ClearCollect, this is the **Value** field, as in this example:
  First(*VariableName* **).Value**

Create an adding machine by using a collection:

1. Add a Text input control, named **TextInput1**, and two buttons, named **Button1** and **Button2**.

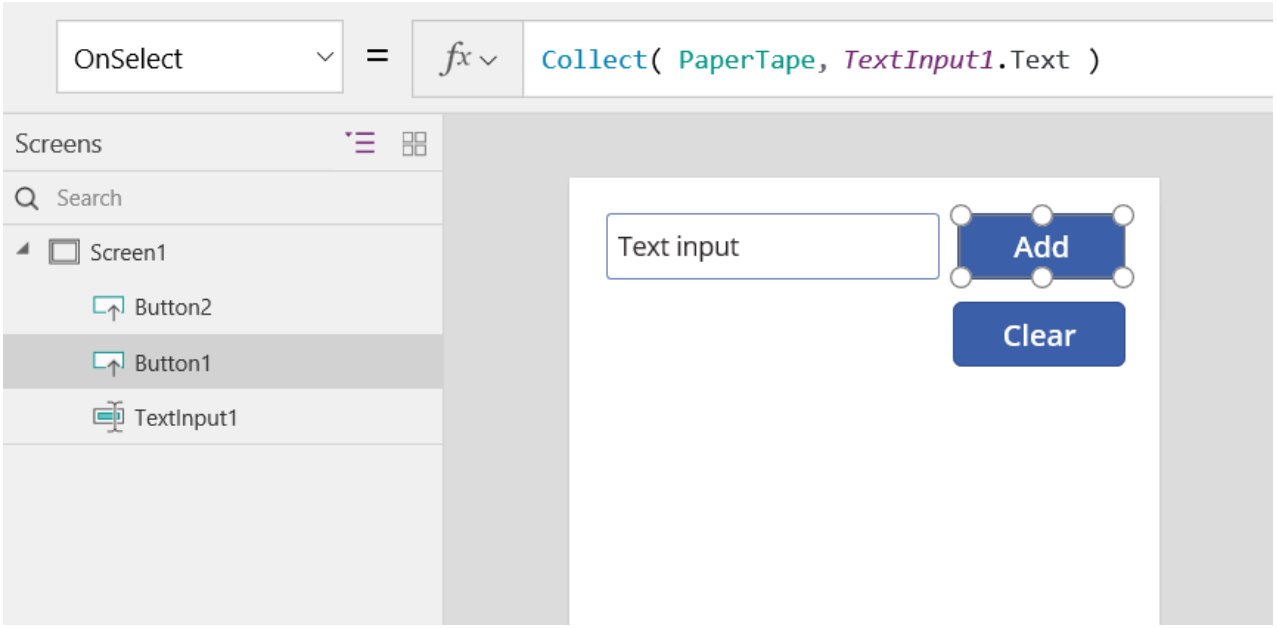
2. Set the Text property of **Button1** to **"Add"**, and set the **Text** property of **Button2** to **"Clear"**.

3. To update the running total whenever a user selects the **Add** button, set its **OnSelect** property to this formula:

   Collect(PaperTape, TextInput1.Text)

   The mere existence of this formula establishes **PaperTape** as a collection that holds a single-column table of text strings. You can reference **PaperTape** anywhere in this app. Whenever a user opens this app, **PaperTape** is an empty table.

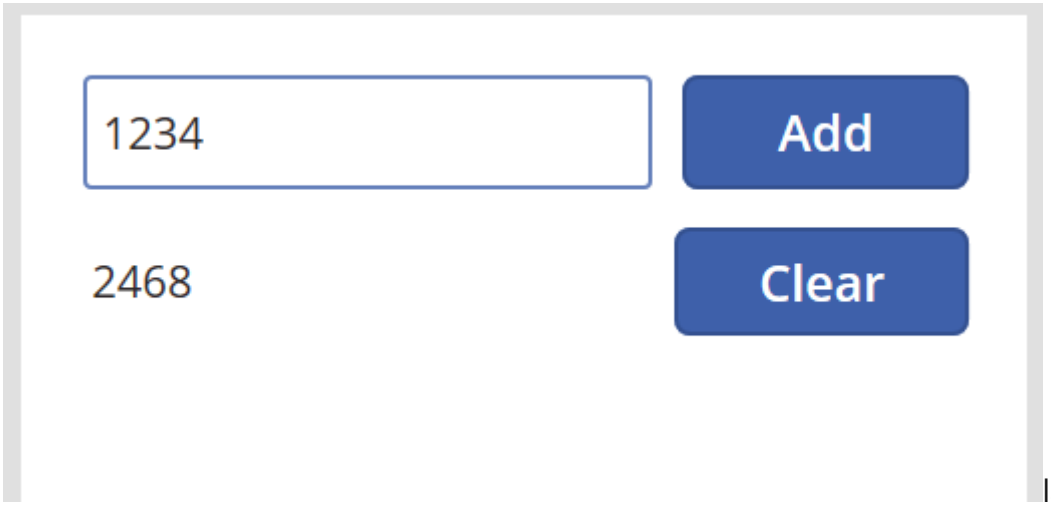
   When this formula runs, it adds the new value to the end of the collection. Because we're adding a single value, **Collect** automatically places it in a single-column table, and the column's name is **Value**, which you'll use later.



4. To clear the paper tape when the user selects the **Clear** button, set its **OnSelect** property to this formula:

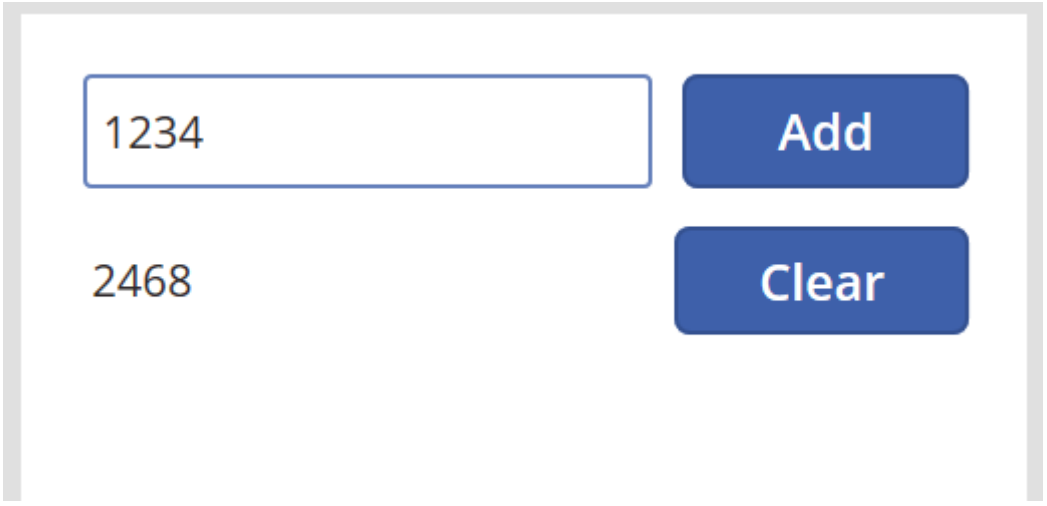
   Clear(PaperTape)

   ![OnSelect property of the Clear but1. To run the adding machine, select F5 to open Preview, enter numbers in the text input control, and select buttons.



label.]
(media/working-with-variables/papertape-3.png)

5. To run the adding machine, press F5 to open Preview, enter numbers in the text-input control, and select buttons.



6. To return to the default workspace, select `Esc`.

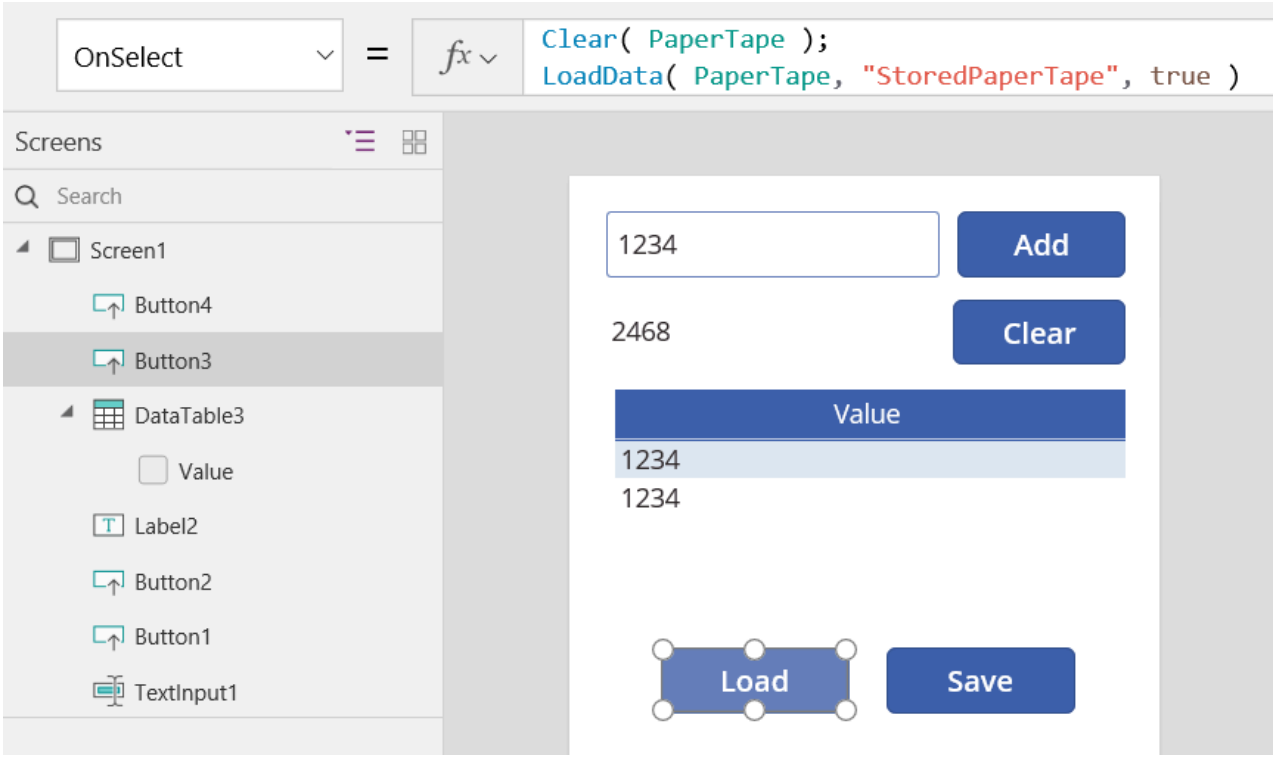
7. To display the paper tape, insert a **Data table** control, and set its **[Items] (controls/properties1. To see the values in your collection, do one of the following:

   - If you're using the preview version of Power Apps Studio, select **Variables** in the app authoring menu, and then select **Collections**.
   - If you're using the classic version of Power Apps Studio, select **Collections** on the **File** menu.. To see the values in your collection, do one of the following:
   - If you're using the preview version of Power Apps Studio, select **Variables** in the app authoring menu and then select **Collections**.
   - If you're using the classic versionof Power Apps Studio, select **Collections** on the **File** menu.

8. To store and get your collection, add two more button controls, and set their **Text** properties to **Load** and **Save**. Set the **OnSelect** property of the **Load** button to this formula:

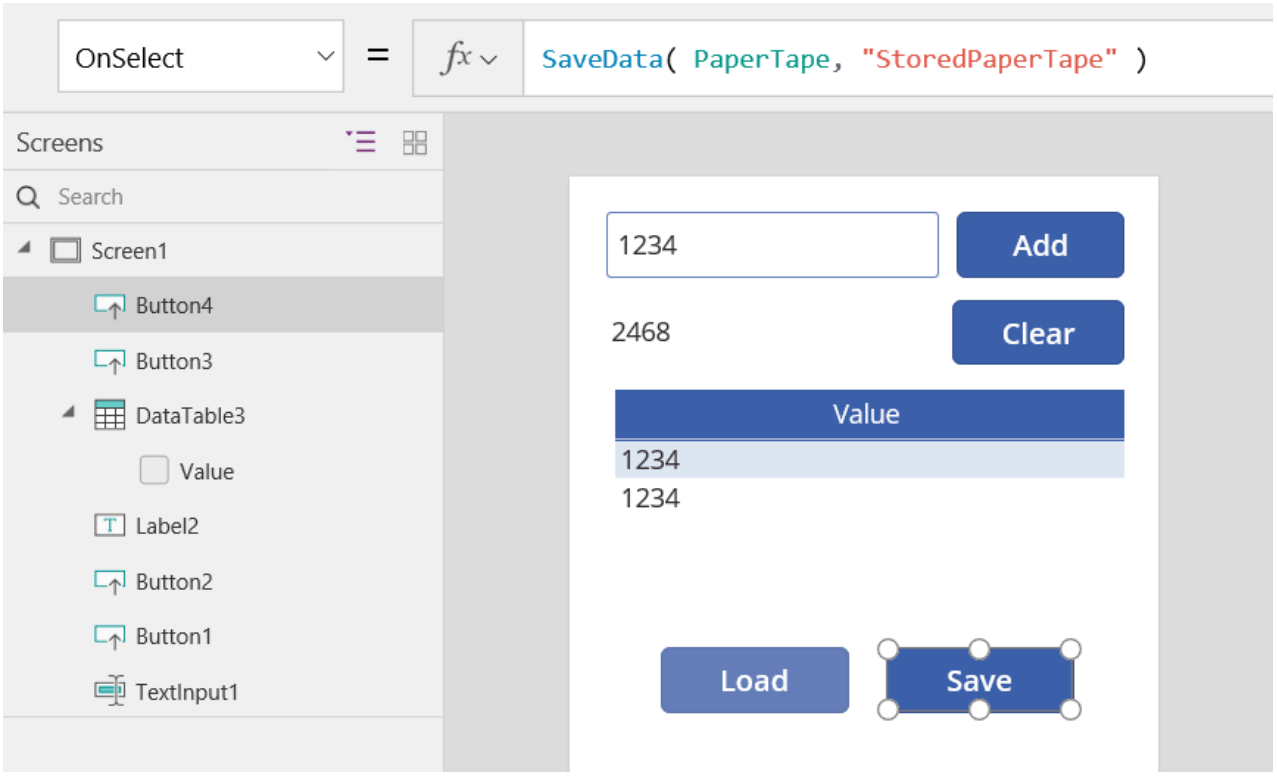
   Clear(PaperTape); LoadData(PaperTape, "StoredPaperTape", true)

   You need to clear the collection first because **LoadData** will append the stored values to the end of the collection.



9. Set the **OnSelect** property of the **Save** button to this formula:

   SaveData(PaperTape, "StoredPaperTape")



10. Preview again by selecting F5, enter numbers in the text input control, and select buttons. Select the **Save** button. Close and reload the app, and select the **Load** button to reload your collection.

> ⓘ **Note**
>
> **SaveData** and **LoadData** functions work in Power Apps Mobile but not Power Apps Studio or the web player for Power Apps.

Understand behavior formulas for canvas apps

Article • 12/16/2022

Most formulas calculate a value. Like an Excel spreadsheet, recalculation happens automatically as values change. For example, you might want to show the value in a **Label** control in red if the value is less than zero or in black otherwise. So you can set the **Color** property of that control to this formula:

Power Fx

```
If( Value(TextBox1.Text) >= 0, Color.Black, Color.Red )
```

In this context, what does it mean when the user selects a **Button** control? No value has changed, so there is nothing new to calculate. Excel has no equivalent to a **Button** control.

By selecting a **Button** control, the user initiates a sequence of actions, or behaviors, that will change the state of the app:

- Change the screen that's displayed: **Back** and **Navigate** functions.
- Control a signal: **Enable** and **Disable** functions.
- Refresh, update, or remove items in a data source: **Refresh**, **Update**, **UpdateIf**, **Patch**, **Remove**, **RemoveIf** functions.
- Update a context variable: **UpdateContext** function.
- Create, update, or remove items in a collection: **Collect**, **Clear**, **ClearCollect** functions.

Because these functions change the state of the app, they can't be automatically recalculated. You can use them in the formulas for the **OnSelect**, **OnVisible**, **OnHidden**, and other **On...** properties, which are called behavior formulas.

More than one action

Use semicolons to create a list of actions to perform. For example, you might want to update a context variable and then return to the previous screen:

Power Fx

```
UpdateContext( { x: 1 } ); Back()
```

Actions are performed in the order in which they appear in the formula. The next function won't start until the current function has completed. If an error occurs, subsequent functions might not start.

Feedback

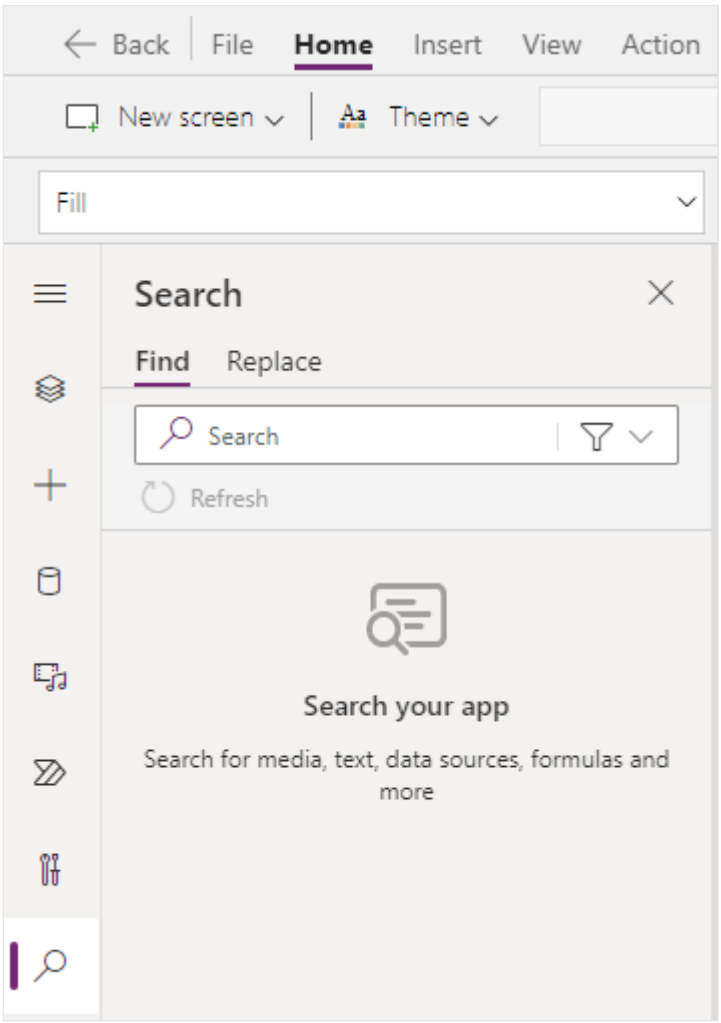
Was this page helpful? Yes No

Provide product feedback

Use the Search pane

Article • 04/24/2023

You can now use the Search pane to locate objects—like media files, variables, collections, and data sources—across your app. You can also use the Search pane to find repeated instances of strings in formulas. For example, using the Search pane, you can look for each instance of `HoverColor` to determine the setting for that property in other areas of your app. You can use the Search pane to learn more about what's being used by your app, where, and you can go right to those results to make any necessary updates. In addition, you can use the replace capability to update one or more instances of text, variables, collections, or formula references.



Prerequisites

- Sign up for Power Apps.
- Create an app from a template, create a new app, or open an existing app in Power Apps.
- Learn how to configure a control.

Open the Search pane

You can open the Search pane by selecting **Search** . You can also use shortcut keys by selecting **Ctrl**+**F** when your cursor is outside the formula bar, or selecting **Ctrl**+**Shift**+**F** when your cursor is inside the formula bar.

You can also open the Search pane directly to the Replace functionality by selecting **Ctrl**+**H** when your cursor is outside the formula bar, or selecting **Ctrl**+**Shift**+**H** when your cursor is inside the formula bar.

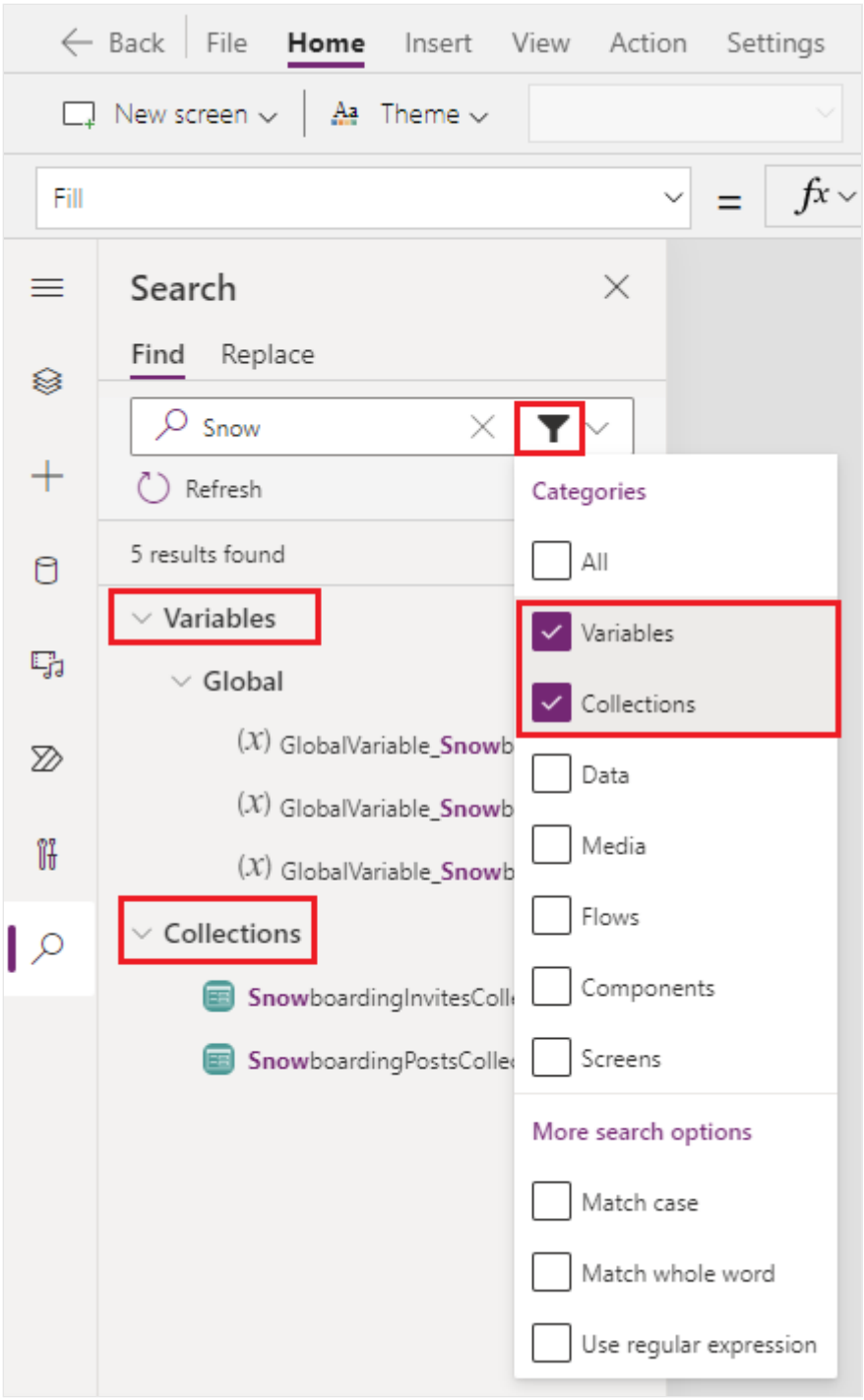
> **Note**
>
> Selecting **Ctrl**+**F** or **Ctrl**+**H** inside the formula bar will open the **Find and Replace** capability inside the formula bar.

Filter search results

You can filter search results by selecting one or more categories. These categories help you focus the search by narrowing down the areas within the app—such as screens, media, and collections.

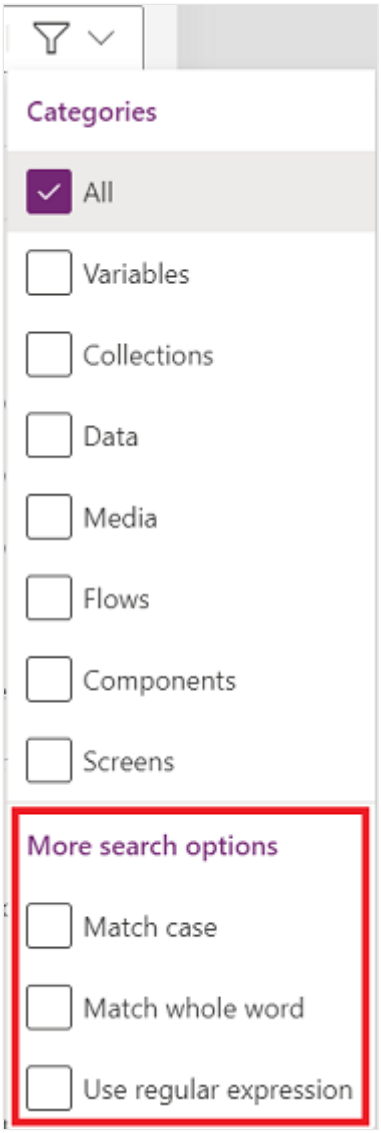
To select the categories you want to search on, use the filter option next to the search box. The default selection is **All**, which searches across all categories within the open app.

The following example shows search results filtered for the search term **Snow** with the selected categories of **Variables** and **Collections**.



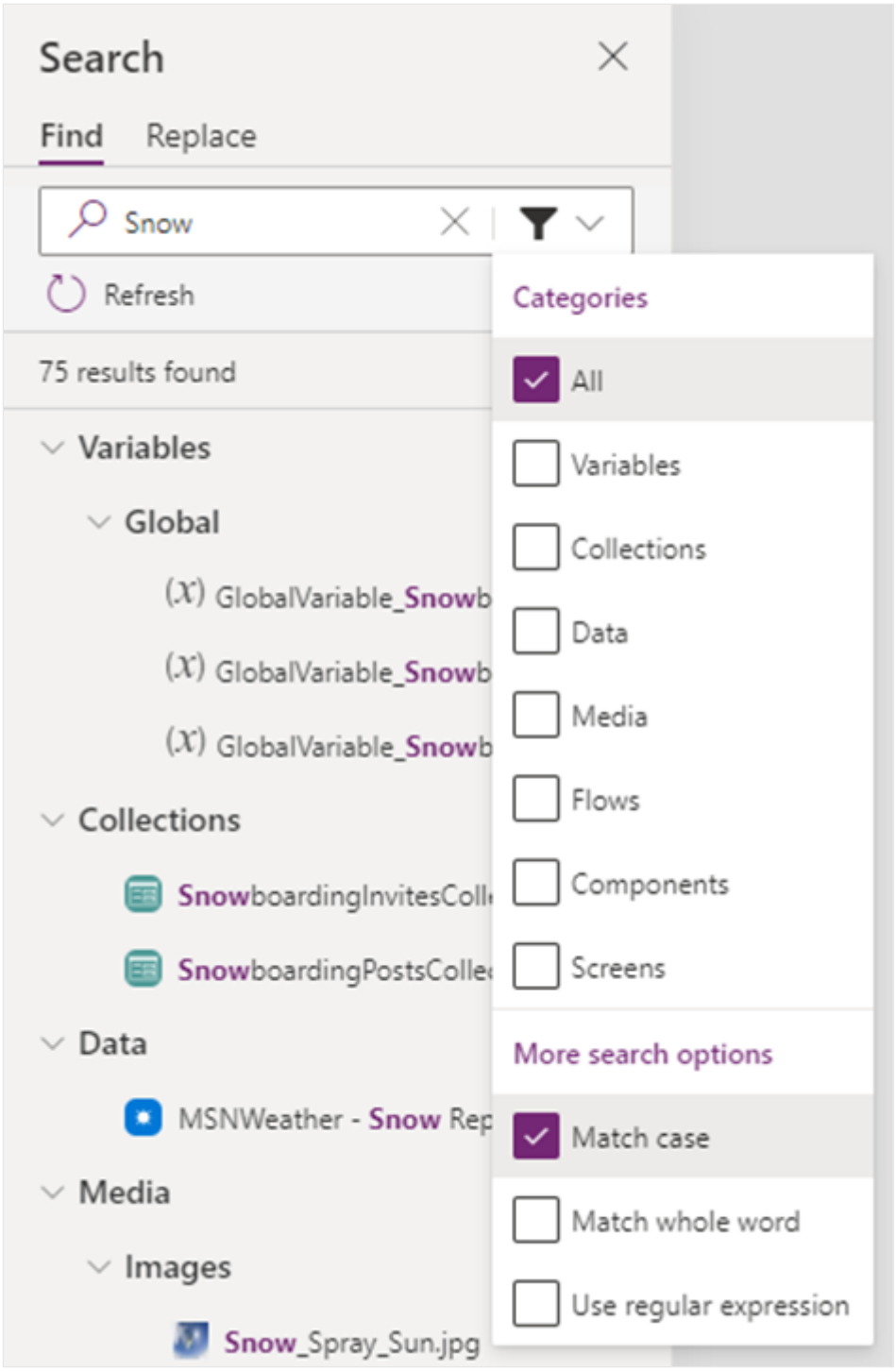
Refine your search

You can use more search options to further refine your search. These options are available inside the filter option named **More search options** (below the **Categories** section).



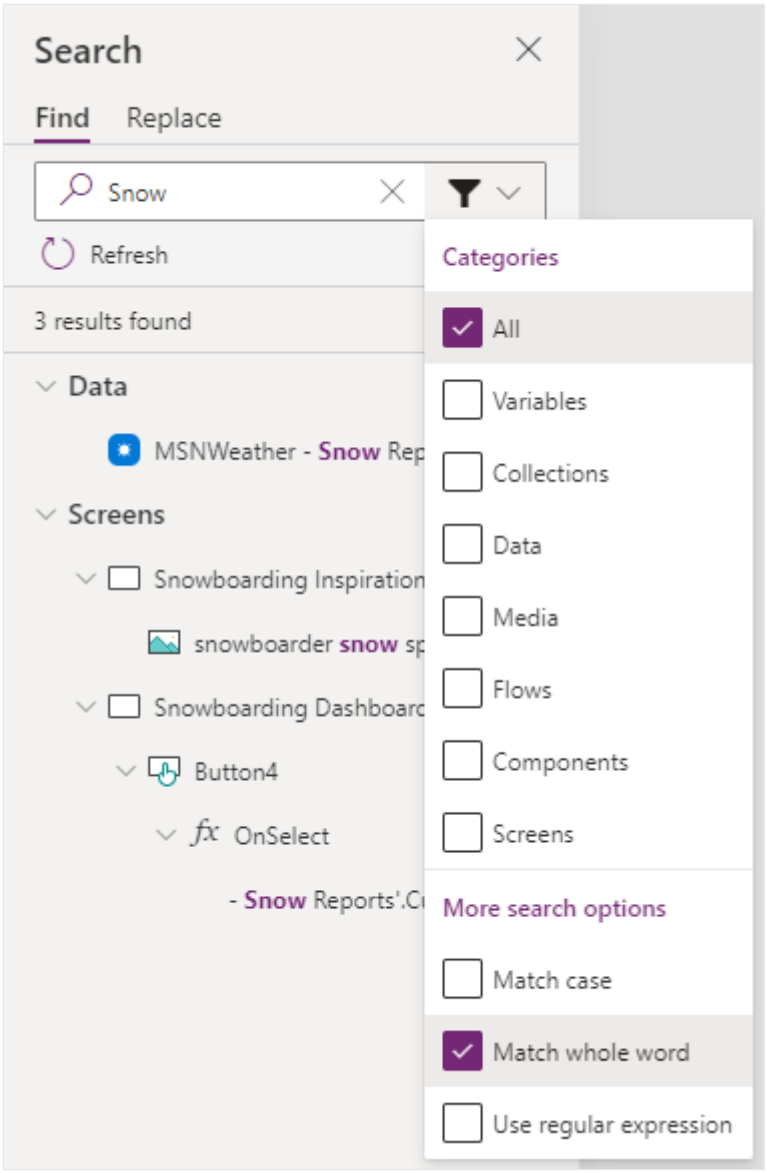
- **Match case** returns search results that only match the specified case.

  In the following example, instances of **Snow** (uppercase S) will appear as a match, but **snow** (lowercase S) won't.



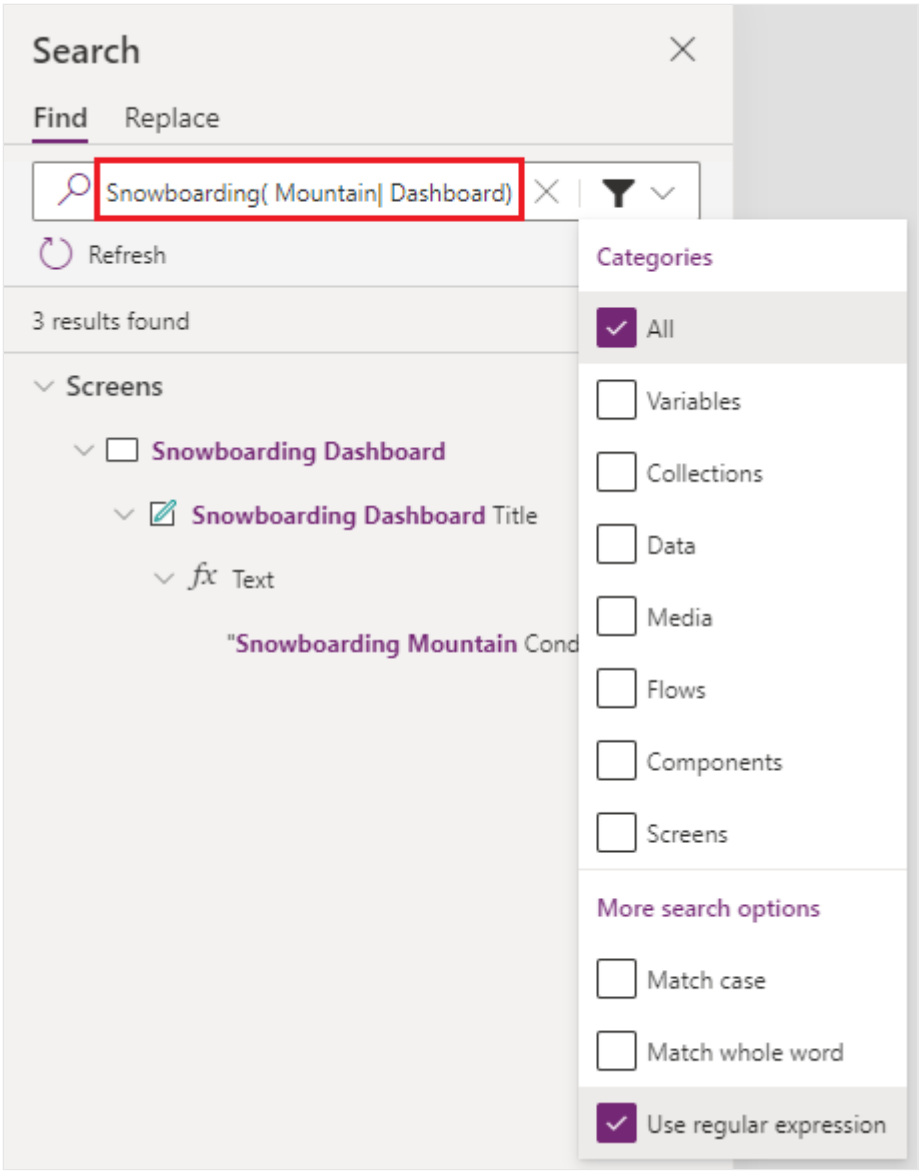
- **Match whole word** returns search results for exact matches of the entire sequence of characters.

  In the following example, **Snow** returns only a few results although **Snow** appears within names many times across all app objects.



- **Use regular expression (RegEx)** returns search results for matches conforming to the regular expression specified within the input area.

  In the following example, using the regular expression search capability with `Snowboarding( Mountain | Dashboard)` formula returns matches for **Snowboarding** when it appears together with either **Mountain** or **Dashboard**.



Work with search results

Selecting search results in different areas of the Search pane will behave differently depending on the context of the result. The search results can be classified under two broad categories—**Definitions** and **Instances**:

- **Definitions** describe the object being referred to in your app, usually in a formula.

  There are several category headers inside the Search pane that you can use to navigate through the results. Categories such as **Variables**, **Collections**, **Data**, **Media**, **Flows**, and **Components** are referred to as **definitions**.

  Selecting the results under such definition category headers takes you to the appropriate information screen such as for Variables and Collections, or to the appropriate pane where that object is available in your app (such as Data, Media, Flows, or Components).

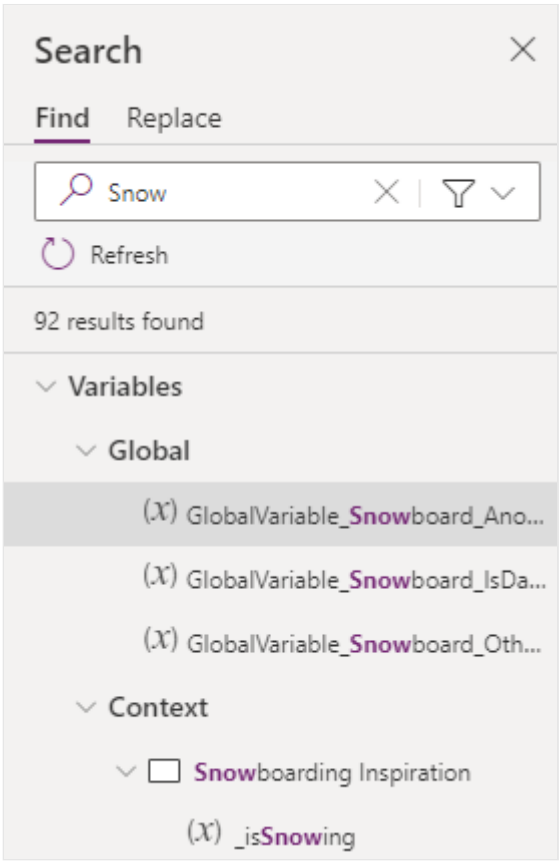
- **Instances** describe the search results that are tied to an individual app, screen, or control property in the formula bar. All these results are tied to the **Screens** category based on the structure found inside the **Tree View**. When you select a result under this **Screens** category, you'll be taken to that specific formula bar reference or the related control, as applicable.

Variables

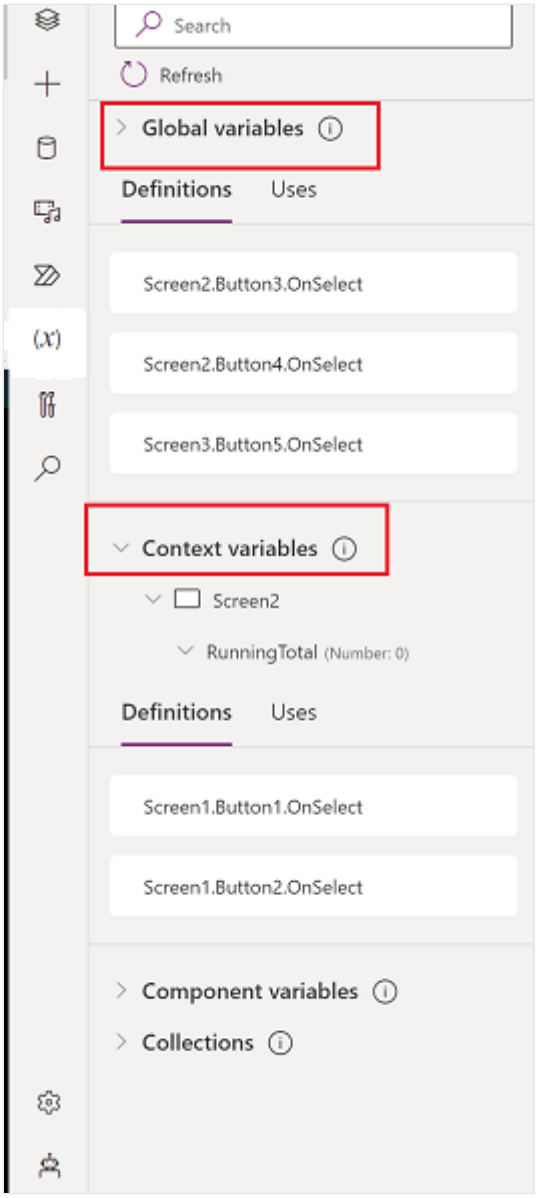
Selecting a global or a context variable under the **Variables** header in the search results will take you to the Variables information screen for the selection.

As shown below, you're taken to the definition of the global or context variable depending on your selection from the search results available.

Search result:

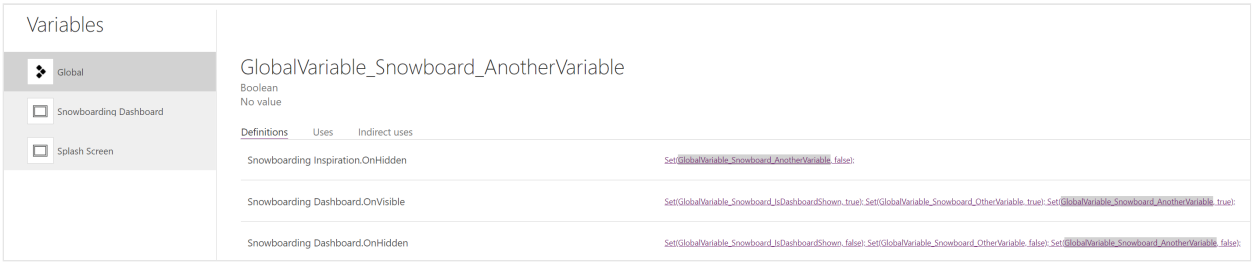


If you're using the preview version of Power Apps Studio, select the **Context variables** or **Global variables** to see more information about the variable.

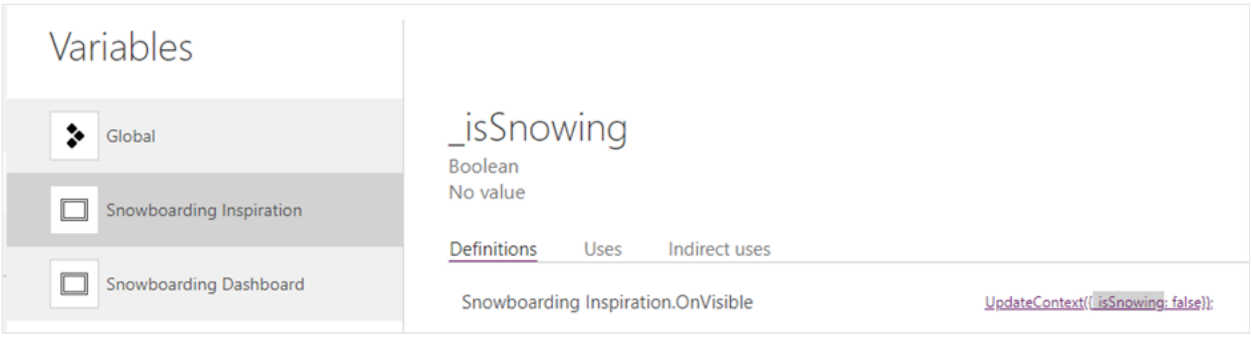


If you're using the classic version of Power Apps Studio, depending on the selected global or context variable, you're taken to the information screen for the selected type of variable.

Global variable selected:

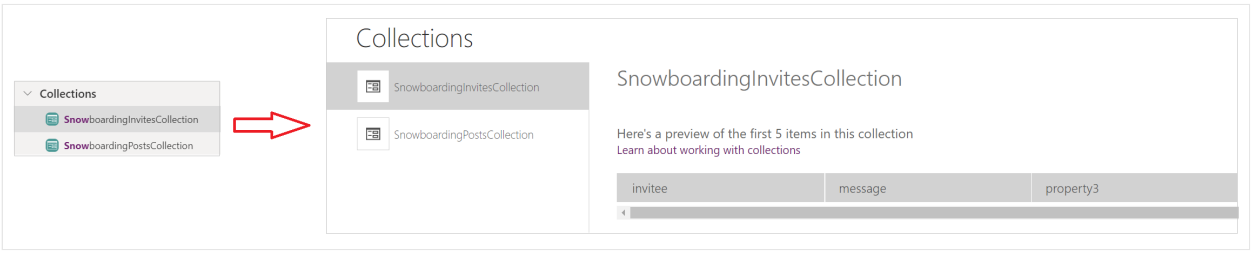


Context variable selected:



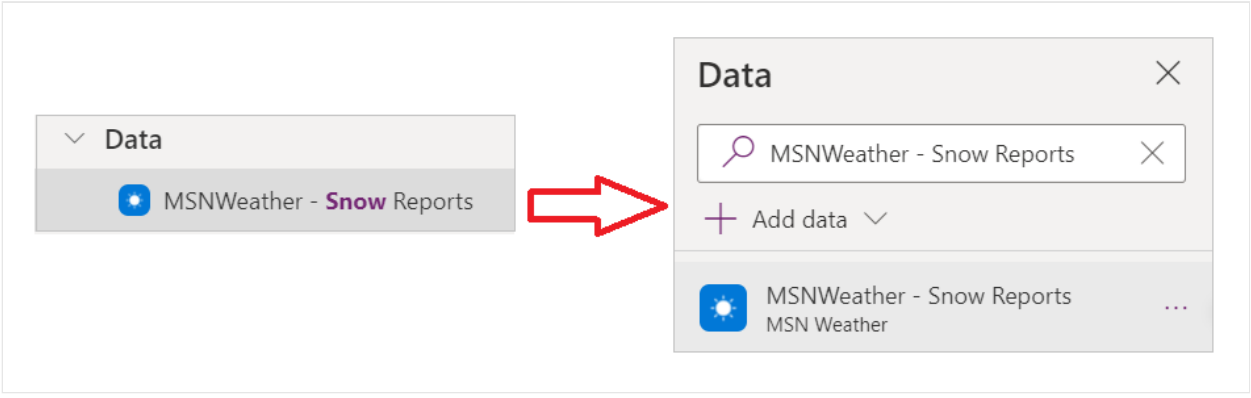
Collections

Selecting a collection under the **Collections** header will take you to the information screen for that collection.



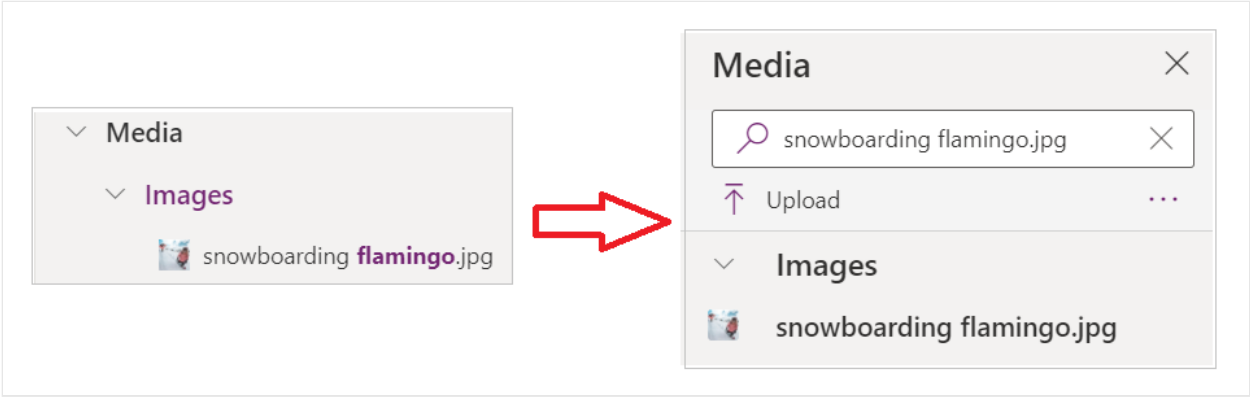
Data

Selecting a result under the **Data** header opens the **Data** pane, and prepopulates the search input for that pane by using the selected result text.



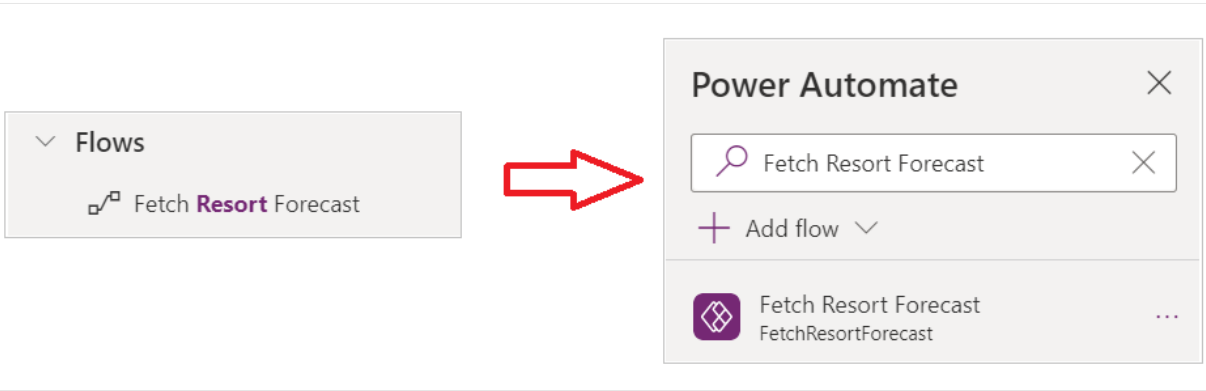
Media

Selecting a result under the **Media** header opens the **Media** pane, and prepopulates the search input for that pane with the selected result text.



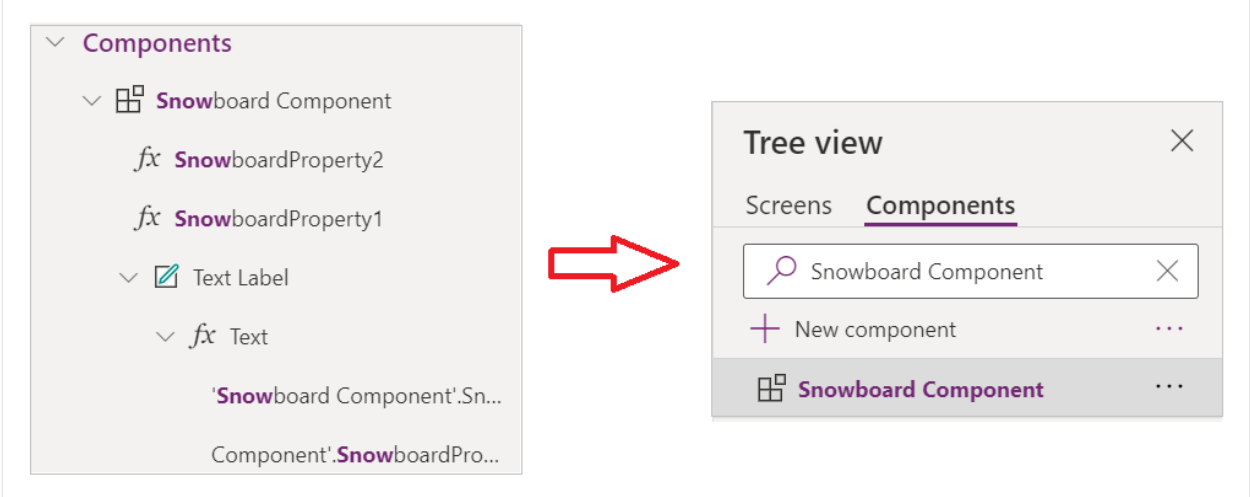
Flows

The **Flows** header only appears if you've enabled the Power Automate pane (preview). Selecting a result here opens the Power Automate pane and prepopulates the search input with the selected result text.



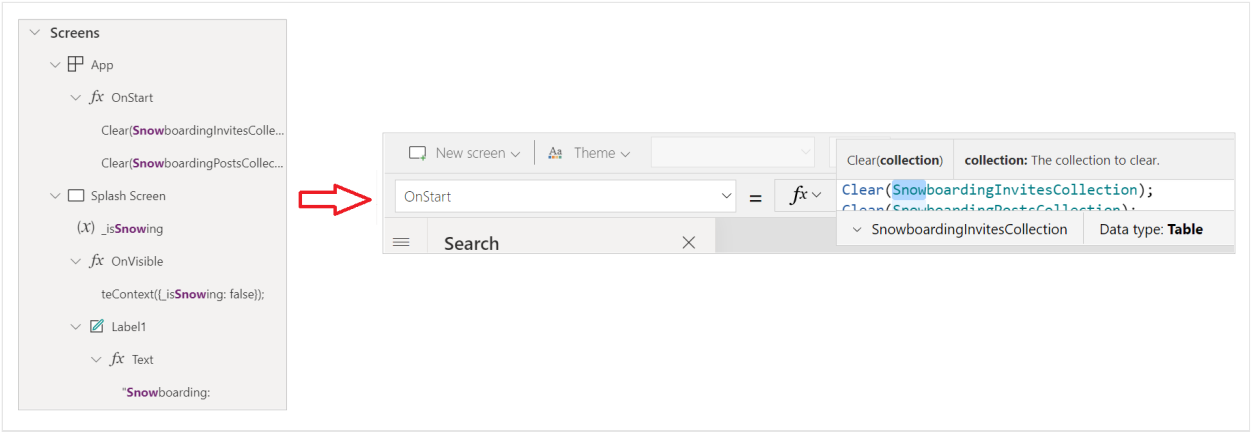
Components

Selecting a result under the **Components** header takes you to the components section in the **Tree view** pane, and prepopulates the search input for that pane with the selected result text.



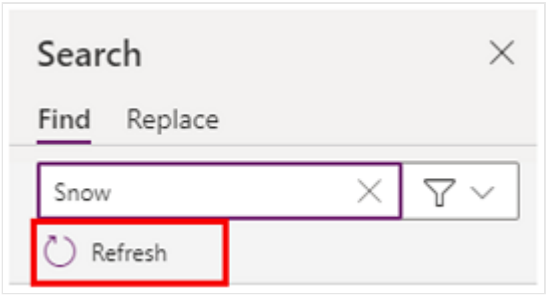
Screens

Selecting a result under the **Screen** header selects the appropriate control or screen on the canvas, and opens the property for the result in the formula bar. The selected result also gets highlighted in the formula bar.



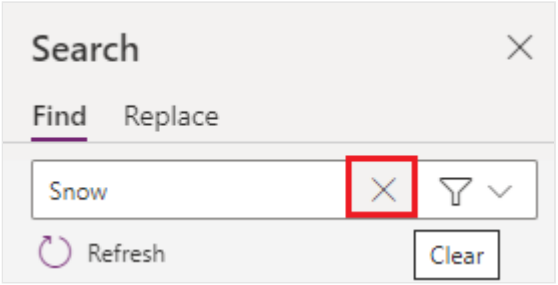
Refreshing the search results

If you make changes within the app, you'll need to refresh the results pane to see the change reflected in your search results.



Clear the search term

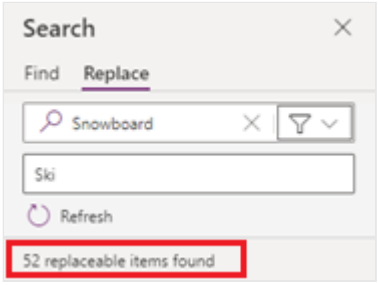
Search terms and results are kept until you no longer need them. To clear the search term and results, select the **X** to the right of the input area.



Replace

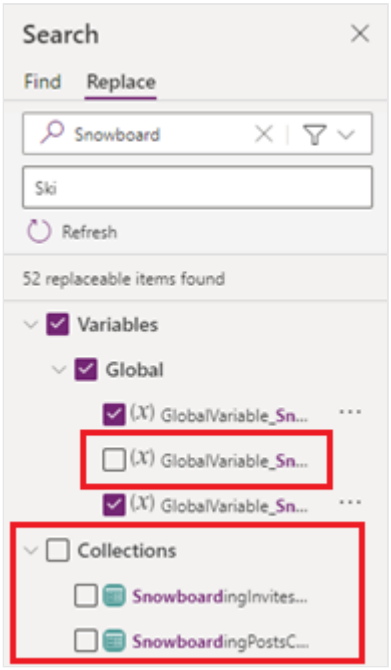
You can replace one or more instances of certain types of search results, including variable names, collection names, any formula text including strings, or other formula references.

After you perform a search, the replace pivot shows the list of search results that can be replaced.

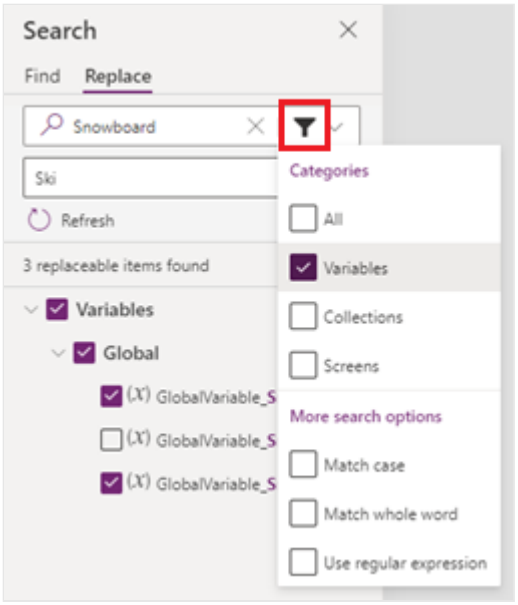


Replace results

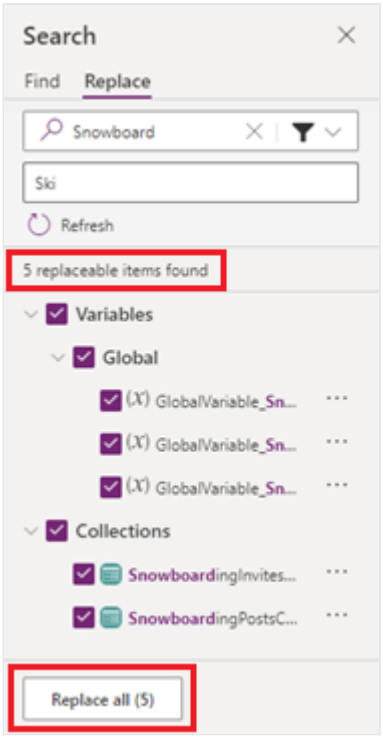
All results that can be replaced are selected by default. You can refine the list of results to replace by deselecting individual results or entire categories by selecting the checkbox next to the result or category heading.



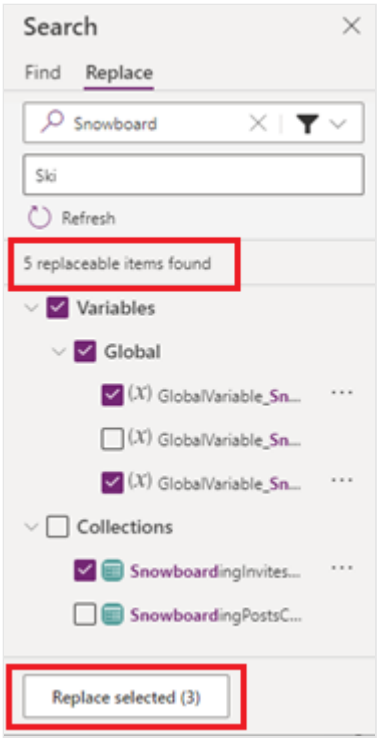
You can also filter the result list to only show selected categories of results.



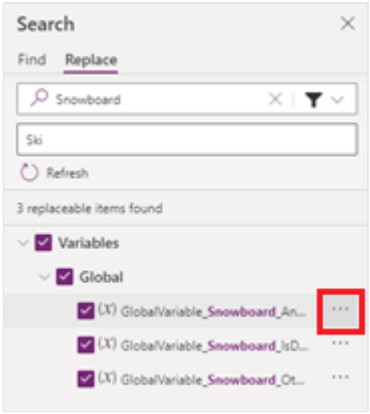
As you refine the list of results to replace, the replace button at the bottom of the panel keeps track of the number of replacements to be made. If all items are selected, the button reads **Replace all** with the total count of all replaceable items.



If you've selected results from the list, the button is updated to read **Replace selected** with the total count of selected items.

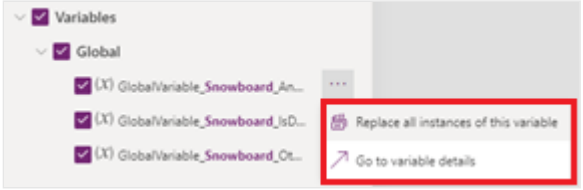


You can also interact with individual results in the results list directly to perform replacements by selecting the ellipsis to the right of each individual result in the result list.



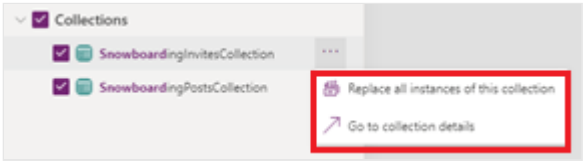
Replacing Variables results

Performing a replacement for results in the Variables category replaces the variable name at the definition level, meaning all instances of the matching variable name will be replaced across the entire app. You can also view the variable details on their information screen to review the usage before making the replacement.



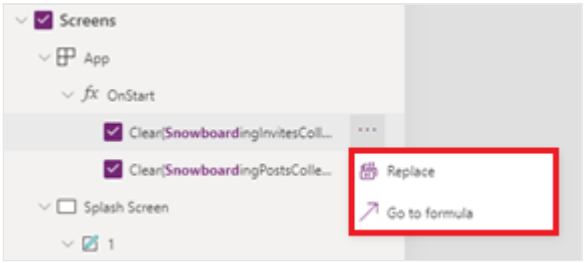
Replacing Collections results

Similar to variables, replacing a result in the Collections category replaces the collection name at the definition level, replacing all matching instances of the collection name across the app. You can view collection details on their information screen prior to making the replacement.



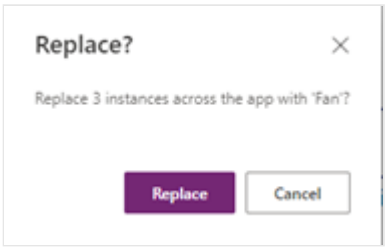
Replacing Screens results

The Screens category shows individual replaceable results following the structure in the tree view, so you can replace text in formulas, formula references, and control names. Replacements made in the Screens category only replace the single instance of the selected matching result, so you can pick and choose what you'd like to replace. You can also review the formula match before making the replacement.

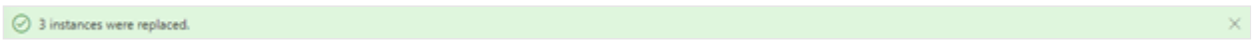


Completing the replacement and undoing replacements

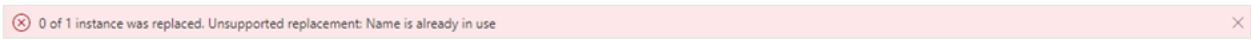
After you've refined your selection to the items you'd like to replace, selecting the button prompts you to confirm the operation.



Confirming the operation performs the replacement, and a notification indicates whether the replacement was successful.



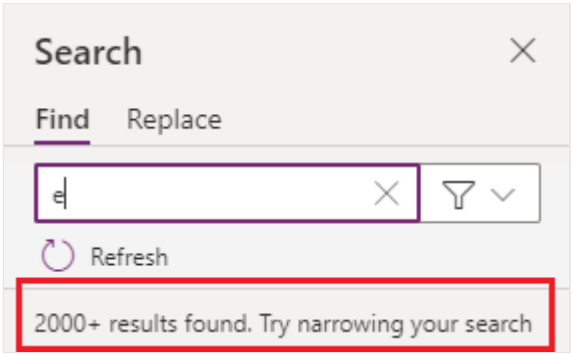
The replacement operation might fail if a control name is already in use or unsupported characters are included.



After the replacement is complete, you can undo the replacement by using the undo button or by selecting **Ctrl**+**Z**.

Limitations

The Search pane is limited to returning a maximum of 2,000 results. A notification appears if you exceed this limit.



See also

Use Find and Replace in the formula bar

Use Find and Replace in the formula bar

Article • 12/11/2023

In complex Power Apps, formulas can get lengthy. Locating and replacing portions of large formulas can become difficult. You can now use the **Find and Replace** capability for the selected control or property to easily find and replace the specified word or sequence of characters.

With **Find and Replace** capability, you can search for combinations of letters, numbers, words, and phrases within a formula in the formula bar. This capability also allows you to use text casing, whole words, and regular expressions to find and replace text inside formulas.

Prerequisites

- Sign up for Power Apps.
- Create an app or open an existing app in Power Apps.
- Learn how to configure a control.

Launch Find and Replace

You can launch the **Find and Replace** control in the formula bar using the **Find and replace** button, or shortcut keys. You can also pre-populate text in the search input.

To get started, create a new app, or edit an existing app in Power Apps Studio. Once open, use any of the following methods to launch the **Find and Replace** control.

Method 1: Use the Find and Replace button

1. Select the control or property that you want to edit.
2. Select the drop-down on the right-side of the formula bar to show the **Find and Replace** control.



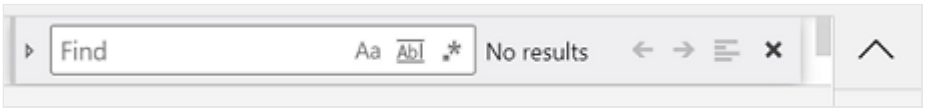
> **Tip**

You can also drag the formula bar down to expand the formula bar making it easier to read complex expressions.

3. Select **Find and replace** from the bottom of the formula bar.



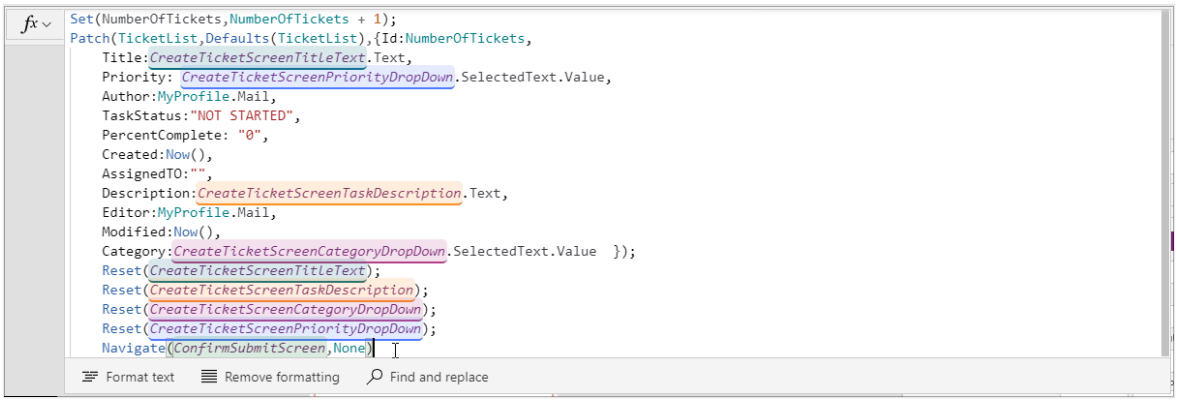
Find and Replace control appears on the upper-right hand side of the formula bar.



Method 2: Use shortcut keys

1. Select the control or property that you want to edit.

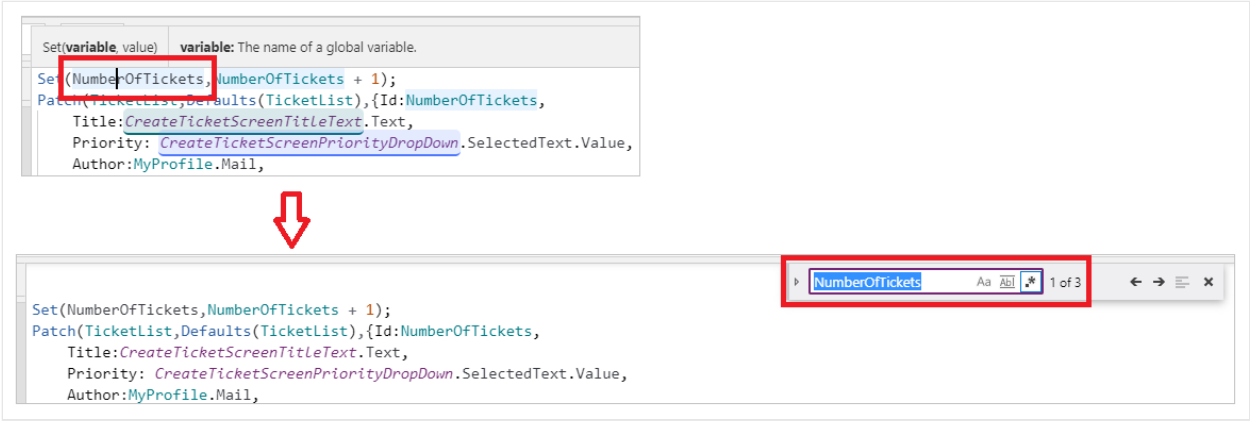
2. While your cursor is within the formula bar, press **Ctrl+F** to find, or **Ctrl+H** to find and replace the specified word or sequence of characters in the formula.



Method 3: Launch Find and Replace with text already selected

You can also launch the control pre-populated with text from your formula that you wish to search for.

To do this, select or keep the cursor inside the formula bar over the section that you want to find or replace. And then, use **Find and replace** button, or the shortcut keys as described earlier to launch **Find and Replace** control.



Working with Find and Replace

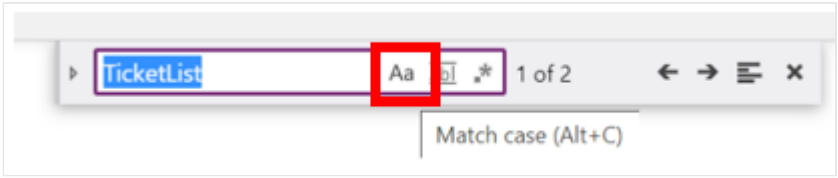
When working with **Find**, you have three options to search the formula. When working with **Replace**, you have two options to choose from. We'll take a look at the options for both **Find** and **Replace**.

Find

Add text or characters to search for into the provided input area. And then, use the icons on the right-hand side of the input area to help refine your search:

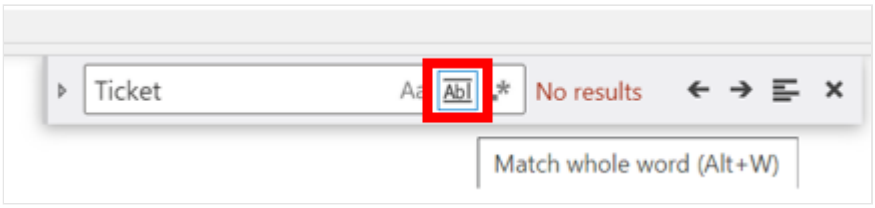
- **Match case** returns only matches with the specified case.

  In the example below, instances of `TicketList` will appear as a match, but `ticketlist` wouldn't.

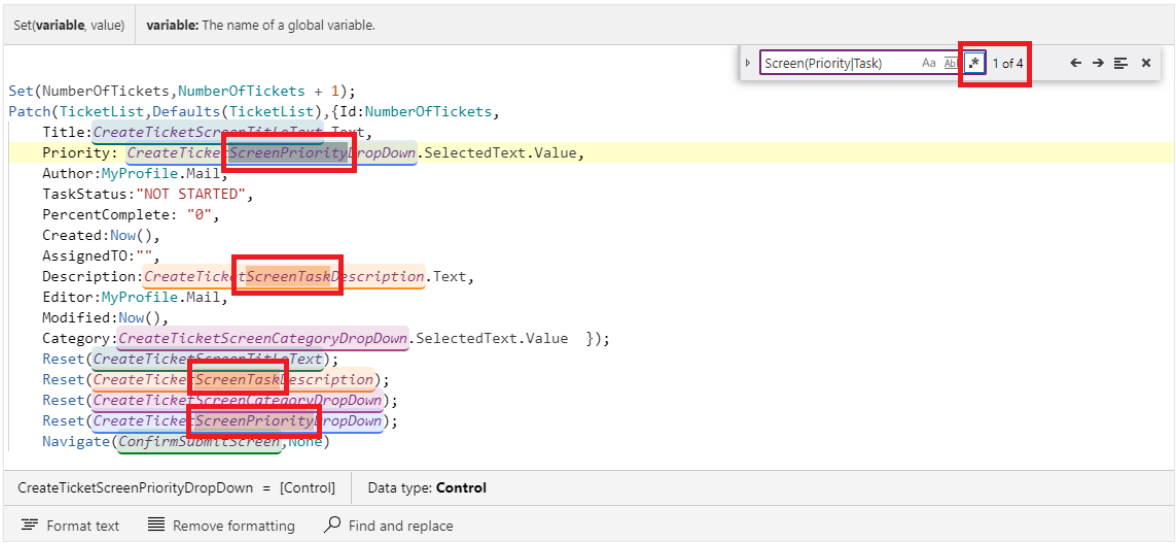
- **Match whole word** returns only exact matches of the entire sequence of characters.

  In the example below, instances of `Ticket` returns no matches although the word `Ticket` appears within the names several times in the formula.

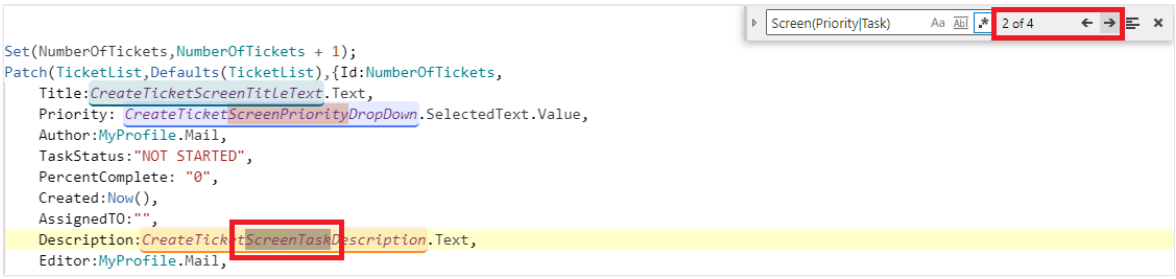
  

- **Use regular expression** (RegEx) returns only matches conforming to the regular expression specified within the input area. More information: Regular expressions

  In the example below, using the Regular Expression search capability with `Screen(Priority|Task)` returns matches for `Screen` when it appears together with either `Priority` or `Task` as shown below.



- **Previous match** or **Next match** arrows allow you to move forward and backward through any matches the search returns. As you move through the matches, the match position updates and the row containing the match is highlighted to let you know which match you're currently working with.



- **Find in selection** limits the search area within a formula to only the selected portion of a formula. To select part of the formula, select and hold at the beginning of the desired searching area and then drag your cursor to highlight the entire desired area. To select using the keyboard, move cursor focus to the beginning of the desired search area, then hold **Shift** key, and use the arrow keys to highlight the desired search area.

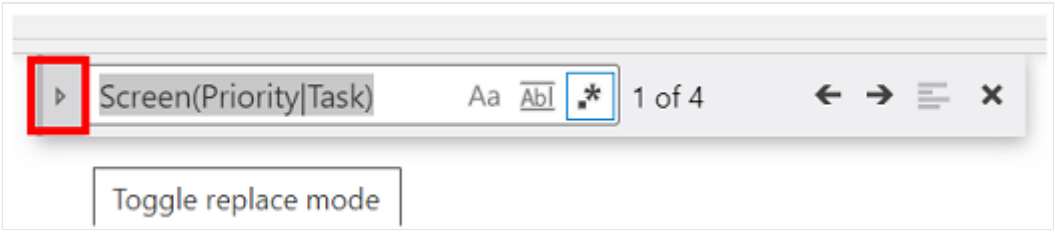
  In the example below, the search has been limited to the selected area, so the search now only returns two matches instead of the four matches returned previously.



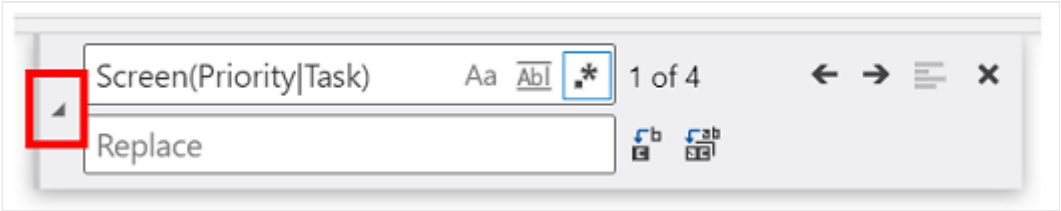
Replace

By default, the **Find and replace** control opens in collapsed format and only displays the Find capability. To expand the control and show the Replace capability, select the icon to the left of the search input box, or press **Ctrl+H** on your keyboard. To collapse again, select the rotated icon.

Collapsed replace mode:



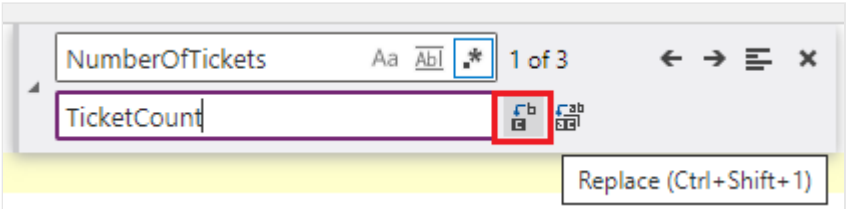
Expanded replace mode:



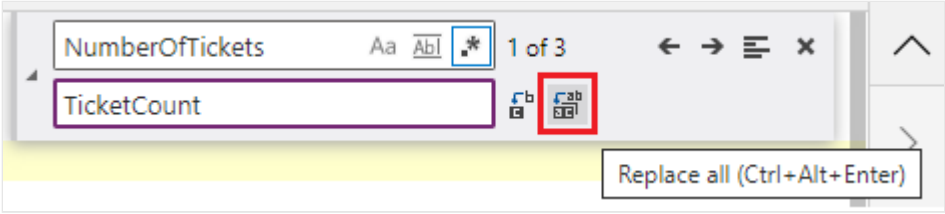
In the Replace input area, specify the word or sequence of characters you'd like to replace the search text.

Use the **Replace** or **Replace all** icons to update one or all matches returned with the specified word or characters.

Replace



Replace all



See also

- Use the Search pane (preview)
- Get started with formulas in canvas apps
- Add and configure controls

Show text, dates, and times in canvas apps

Article • 02/11/2025

In Power Apps, add dates and times to a canvas app, and format them to show the right level of detail or to reflect your locale. Calculate the amount of time between two dates, or calculate a date that's a certain amount of time before or after a date that you specify. Convert dates to or from separate values for days, months, and years, and convert times to or from separate values for hours, minutes, and seconds.

For example, add data from users about stock trades or client meetings, data from an external source, or data from another app created in Power Apps. If that data includes times down to the millisecond, round it to the nearest minute for simplicity. Calculate how many days remain before a major milestone. If you want to schedule client meetings every five days, calculate those dates automatically. If May 10, 1985, is stored in separate fields for the day, the month, and the year, consolidate them into a single value. Conversely, break each date into separate values if your app manages them separately.

Prerequisites

- Sign up for Power Apps, and then sign in by providing the same credentials that you used to sign up.
- Create an app or open an existing app in Power Apps.
- Learn how to configure a control in Power Apps.

Show text in a Label control

Show text in a **Label** control by setting the value of its **Text** property. Set this property by typing directly into the control or by typing an expression in the formula bar.

- If you type directly into the control, it shows exactly what you type.
- If you type an expression in the formula bar, the control shows the result of the expression.

Here are some examples.

1. Add a **Label** control named **ShowText**, and set its **Text** property to this formula:
   Now()

If your computer is set to the "en-us" locale, the current date and time appears in this format:
mm/dd/yyyy hh:mm AM/PM

If your computer is set to a locale such as "fr-fr", the current date and time appears in this format:
dd/mm/yyyy hh:mm AM/PM

2. Set the Text property of **ShowText** to this formula:
   DateDiff(Today(), DateValue("01/01/2025"))

   The control shows the number of days between today and January 1, 2020, by using these functions:

   - **DateDiff**, which calculates the number of days, quarters, or years between two dates.
   - **Today**, which calculates the current day as a value.
   - **DateValue**, which converts a literal string, as shown between double quotation marks, to a value on which calculations can be performed.

3. Add a Text input control named **BirthDate**, and move it under **ShowText**.

4. In **BirthDate**, type the month and the day of your birth (for example, **05/18**).

5. Set the Text property of **ShowText** to this formula:
   DateDiff(Today(), DateValue(BirthDate.Text))

   ShowText shows the number of days between today and whatever date you type into **BirthDate**. If your birthday has already occurred this year, **ShowText** displays a negative value.

Format dates and times by using DateTimeValue

Convert dates and times from strings of text to values, which you can format in a variety of ways and use in calculations. Specify the format by using built-in and custom options.

> ⓘ **Note**
>
> The DateTimeValue and DateValue functions can convert dates in any of these formats into values:
>
> - MM/DD/YYYY

- DD/MM/YYYY
- DD Mon YYYY
- Month DD, YYYY

1. Add a Text input control named **ArrivalDateTime**, and type a date and time in this format:
   5/10/85 6:15 AM

2. Add a Label control named **ShowDate**, and set its Text property to this formula:
   DateTimeValue(ArrivalDateTime.Text)

   ShowDate shows the same information that you typed, but it's been converted from text to a value and formatted differently. For example, the year appears as four digits instead of just two.

3. Change the Text property of **ShowDate** to this formula:
   DateTimeValue(ArrivalDateTime.Text, "fr")

   ShowDate shows the day before the month, as a French user would expect.

   > **Tip**
   >
   > To display a list of other locales in Intellisense, remove the closing quotation mark and **fr** from the formula, but leave the open quotation mark:

4. To use one of several built-in formats, change the Text property of **ShowDate** to this formula:
   Text(DateTimeValue(ArrivalDateTime.Text), DateTimeFormat.LongDateTime)

   ShowDate shows the day of the week, the date, and the time.

   > **Tip**
   >
   > The **DateTimeFormat** parameter supports several other built-in formats. To display that list, remove **LongDateTime** from the formula.

5. To use a custom format, change the Text property of **ShowDate** to this formula:
   Text(DateTimeValue(ArrivalDateTime.Text), "mm/dd/yyyy hh:mm:ss.fff AM/PM")

   ShowDate shows the date/time value in the format that you specified, including milliseconds.

> **Tip**
>
> To round the time to the nearest tenth or hundredth of a second, specify **hh:mm:ss.f** or **hh:mm:ss.ff** in the formula.

Format a date by using DateValue

1. Add a **Text input** control named **ArrivalDate**, and then type a date in it (for example, **5/10/85**).

2. Add a **Label** control named **FormatDate**, and set its **Text** property to this formula:
   DateValue(ArrivalDate.Text)

   FormatDate shows the date that you typed, except the year appears as four digits.

3. Set the **Text** property of **FormatDate** to this formula:
   DateValue(ArrivalDate.Text, "fr")

   FormatDate shows the day before the month, just as a French user would expect.

4. To use one of several built-in formats, set the **Text** property of **FormatDate** to this formula:
   Text(DateValue(ArrivalDate.Text), DateTimeFormat.LongDate)

   FormatDate shows the day of the week, the month, the day, and the year.

5. To use a custom format, set the **Text** property of **FormatDate** to this formula:
   Text(DateValue(ArrivalDate.Text), "yy/mm/dd")

   FormatDate shows the date in the format that you specified.

Format a time using DateTimeValue

1. Add a **Text input** control named **ArrivalTime**, and then type **6:15 AM** in it.

2. Add a **Label** control named **ShowTime**.

3. To use one of several built-in formats, set the **Text** property of **ShowTime** to this formula:
   Text(DateTimeValue(ArrivalTime.Text), DateTimeFormat.LongTime)

   ShowTime shows the time that you specified, including seconds.

4. To use a custom format, set the Text property of **ShowTime** to this formula:
   Text(DateTimeValue(ArrivalTime.Text), "hh:mm:ss.fff AM/PM")

   ShowTime shows the time that you specified, including seconds and milliseconds.

   > **Tip**
   >
   > To round the time to the nearest tenth or hundredth of a second, enter **hh:mm:ss.f** or **hh:mm:ss.ff** in the formula.

Show the time between dates

1. Add two Text input controls named **Start** and **End**.
2. Type **4/1/2015** in **Start**, and type **1/1/2016** in **End**.
3. Add a Label control named **DateDiff**, and set its Text property to this formula:
   DateDiff(DateValue(Start.Text), DateValue(End.Text))

   DateDiff shows **275**, which is the number of days between April 1, 2015, and January 1, 2016.
4. Set the Text property of **DateDiff** to this formula:
   DateDiff(DateValue(Start.Text), DateValue(End.Text), Months)

   DateDiff shows **9**, which is the number of months between April 1, 2015, and January 1, 2016. Replace **Months** with **Quarters** or **Years** to show the time in those units.

Identify a date before or after another date

1. Add a Text input control named **Start**, and type **5/10/1985** in it.
2. Add a Label control named **DateAdd**, and set its Text property to this formula:
   DateAdd(DateValue(Start.Text), 3)

   DateAdd shows **5/13/1985**, which is three days after the date in **Start**.
3. Set the Text property of **DateAdd** to this formula:
   DateAdd(DateValue(Start.Text), -3)

   DateAdd shows **5/7/1985**, which is three days before the date in **Start**.

4. Change the Text property of **DateAdd** to this formula:
   DateAdd(DateValue(Start.Text), 3, Months)

   The label shows **8/10/1985**, which is three months after the date in **Start**. Replace **Months** with **Quarters** or **Years** to identify a date that's the specified number of quarters or years before or after the date in **Start**.

Calculate dates based on years, months, and days

1. Add three Drop down controls named **Year**, **Month**, and **Day**.

2. Set the Items property of **Year** to this formula:
   Table({Year:"2014"}, {Year:"2015"}, {Year:"2016"})

3. Set the Items property of **Month** to this formula:
   Table({Month:"1"}, {Month:"2"}, {Month:"3"}, {Month:"4"}, {Month:"5"}, {Month:"6"}, {Month:"7"}, {Month:"8"}, {Month:"9"}, {Month:"10"}, {Month:"11"}, {Month:"12"})

4. Set the Items property of **Day** to this formula:
   Table({Day:"1"}, {Day:"2"}, {Day:"3"}, {Day:"4"}, {Day:"5"}, {Day:"6"}, {Day:"7"}, {Day:"8"}, {Day:"9"}, {Day:"10"}, {Day:"11"}, {Day:"12"}, {Day:"13"}, {Day:"14"}, {Day:"15"}, {Day:"16"}, {Day:"17"}, {Day:"18"}, {Day:"19"}, {Day:"20"}, {Day:"21"}, {Day:"22"}, {Day:"23"}, {Day:"24"}, {Day:"25"}, {Day:"26"}, {Day:"27"}, {Day:"28"}, {Day:"29"}, {Day:"30"}, {Day:"31"})

5. Add a Label control, and set its Text property to this formula:
   Text(Date(Value(Year.SelectedText.Value), Value(Month.SelectedText.Value), Value(Day.SelectedText.Value)), DateTimeFormat.LongDate)

   Wednesday, January 1, 2014 is listed by default. Select different values in the Drop down controls to change the date in the Label control.

You may need to convert data that you didn't expect. If you add Text input controls instead of Drop down controls, a user may enter an incorrect date, such as May 45. The Date function handles atypical data in the following ways:

- If a year value is between 0 and 1899 (inclusive), the function adds that value to 1900 to calculate the year.
- If a year value is between 1900 and 9999 (inclusive), the function uses that value as the year.

- If a year value is less than 0 or is 10000 or greater, the function returns an error value.
- If a month value is greater than 12, the function adds that number of months to the first month of the specified year.
- If a month value is less than 1, the function subtracts that many months, plus 1, from the first month of the specified year.
- If a day value is greater than the number of days in the specified month, the function adds that many days to the first day of the month and returns the corresponding date from a subsequent month.
- If a day value is less than 1, the function subtracts that many days, plus 1, from the first day of the specified month.

Calculate times based on hours, minutes, and seconds

1. Add two **Drop-down** lists named **Hour** and **Minute**.

2. Set the **Items** property of **Hour** to this formula:
   Table({Hour:"9"}, {Hour:"10"}, {Hour:"11"}, {Hour:"12"}, {Hour:"13"}, {Hour:"14"}, {Hour:"15"}, {Hour:"16"}, {Hour:"17"})

3. Set the **Items** property of **Minute** to this formula:
   Table({Minute:"0"}, {Minute:"15"}, {Minute:"30"}, {Minute:"45"})

4. Add a **Label** control, and set its **Text** property to this formula:

   Text(Time(Value(Hour.SelectedText.Value), Value(Minute.SelectedText.Value),0), DateTimeFormat.ShortTime)

5. Select **15** in **Hour** and **45** in **Minute**.

   The **Label** control shows **3:45 PM**.

   You can add entries to **Hour** and **Minute** so that users can select from a bigger range of hours and a more precise number of minutes. You can also add a third **Drop down** control so that users can specify seconds. If you add a third list, set the **Text** property of the **Label** control to the following expression:
   Text(Time(Value(Hour.Selected.Value), Value(Minute.Selected.Value), Value(Second.Selected.Value)), DateTimeFormat.LongTime)

See also

Date picker control examples

Feedback

Was this page helpful? Yes No

Provide product feedback

Create and update a collection in a canvas app

Use a collection to store data that users can manage in your app. A collection is a group of items that are similar, such as products in a product list. For more information about different types of variables, including collections, see Understand canvas-app variables.

Prerequisites

- Sign up for Power Apps, and then sign in by providing the same credentials that you used to sign up.
- Create an app or open an existing app in Power Apps.
- Learn how to configure a control in Power Apps.

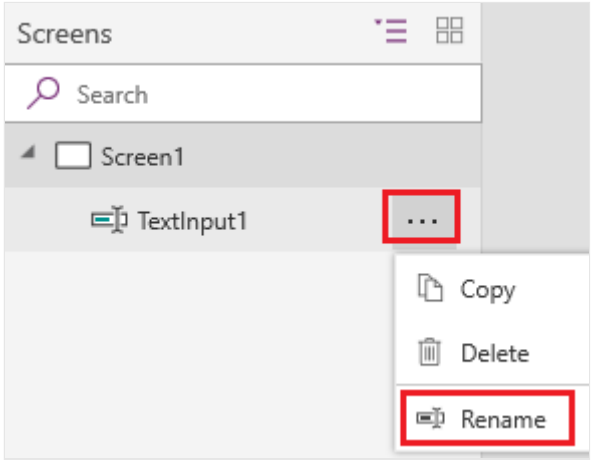
Watch this video to learn how to create and update a collection:

https://learn-video.azurefd.net/vod/player?id=b709db8a-7d9c-4869-b2f7-76027df3bee0&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fcreate-update-collection

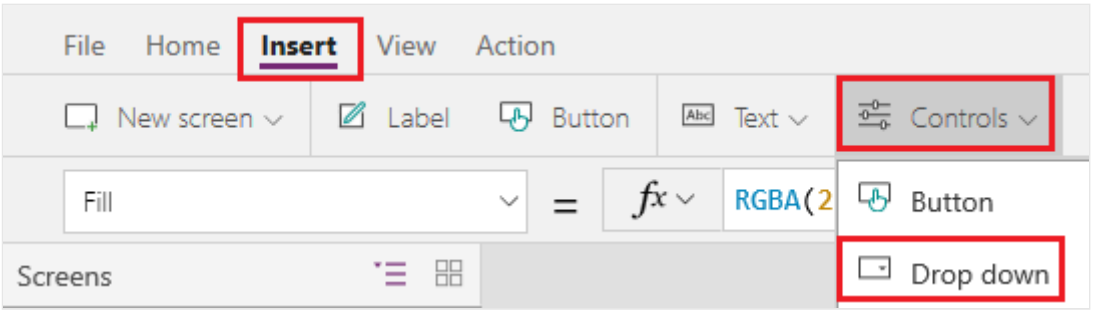
Create a multicolumn collection

1. In Power Apps Studio, add a **Text input** control.

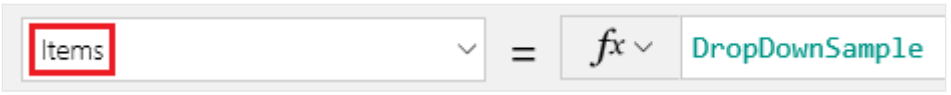
2. Rename the control by selecting its ellipsis in the left navigation pane, selecting **Rename**, and then typing **ProductName**.



3. Add a **Drop down** control.



4. Rename the **Drop down** control **Colors**, and make sure that the **Items** property is selected in the property list.



5. In the formula bar, replace **DropDownSample** with this expression:

   `["Red","Green","Blue"]`

6. Add a **Button** control, set its **Text** property to "**Add**", and set its **OnSelect** property to this formula:

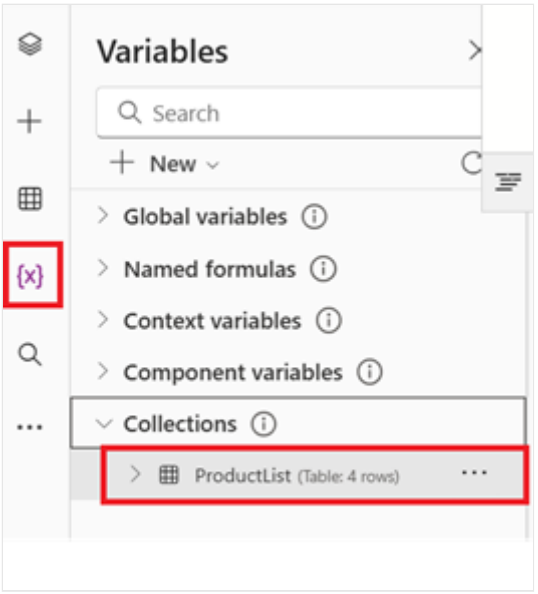
   Power Fx

   ```
   Collect(
       ProductList,
       {
           Product: "ProductName.Text",
           Color: "Colors.Selected.Value"
       }
   )
   ```

7. To preview the app, press `F5` or select the play button. Type some text into **ProductName**, select an option in **Colors**, and then select **Add**.

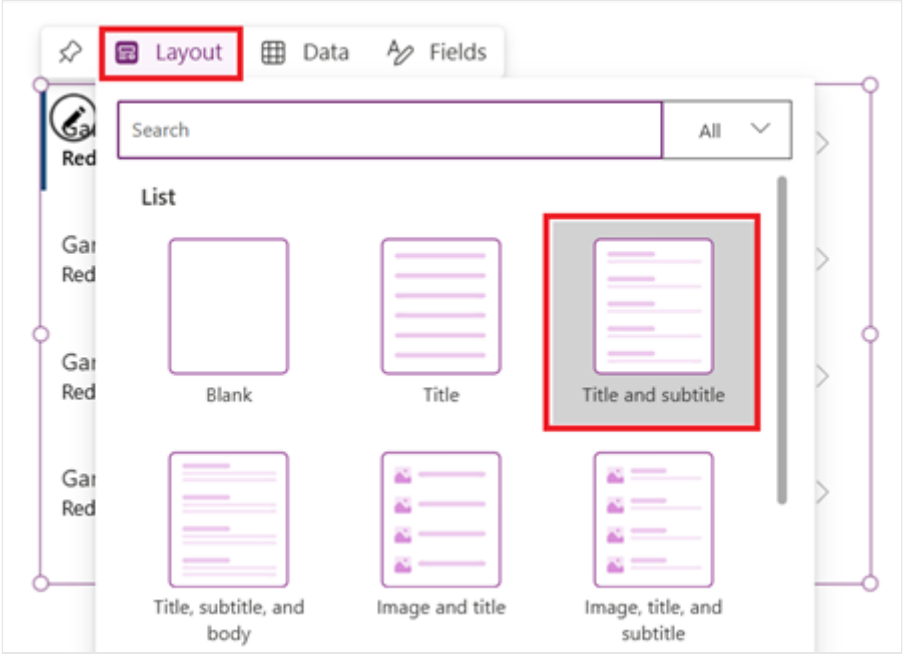
8. Repeat the previous step at least two more times, and then press `Esc`.

9. Select the **Variables** pane and then select **Collections** to see the collections that you created.



Show a collection

1. Select **Insert** > **Vertical gallery** control.
2. Set the gallery's **Items** property to **ProductList**.
3. Select the gallery in the tree view and then select **Fields**.
4. Set the subtitle field to **Color**, and set the title field to **Product**.
5. Select **Layout** > **Title and subtitle**.



Remove one or all items

1. Select the gallery and add a **Trash** icon to the gallery template.

2. Set the icon's **OnSelect** property to this formula:

   ```
   Remove(ProductList, ThisItem)
   ```

3. Outside the gallery, add a button, set its **Text** property to "**Clear**", and set its **OnSelect** property to this formula:

   ```
   Clear(ProductList)
   ```

4. While holding down the Alt key, select the **Trash** icon for an item to remove that item from the collection, or select the **Clear** button to remove all items from the collection.

Put a list into a collection

1. Create a connection to a list created using Microsoft Lists.

2. Add a button, and set its **OnSelect** property to this function, replacing *ListName* with the name of your list:

   ```
   Collect(MySPCollection, ListName)
   ```

   This function creates a collection named **MySPCollection** that contains the same data as your list.

3. Select the button while holding down the `Alt` key.

For information about how to show data in a form (with drop-down lists, date pickers, and people pickers), see Edit form and Display form controls.

Next steps

- Review the reference article for the **Collect** function.
- Learn how to shape data in a collection by using the AddColumns, DropColumns, RenameColumns, and ShowColumns functions.

Last updated on 01/13/2026

Show information about a user in a canvas app

Article • 12/16/2022

In Power Apps, show the full name, the email address, and the picture that's associated with the user who's signed in to a canvas app. You can use this information, for example, to automatically fill in a form.

For example, you can use this feature to:

- Create a sign-up "sheet" for users to attend training, volunteer for events, check in at a kiosk, and more.
- Display the full name on a Human Resources app.
- Automatically enter an email address when contacting your helpdesk.

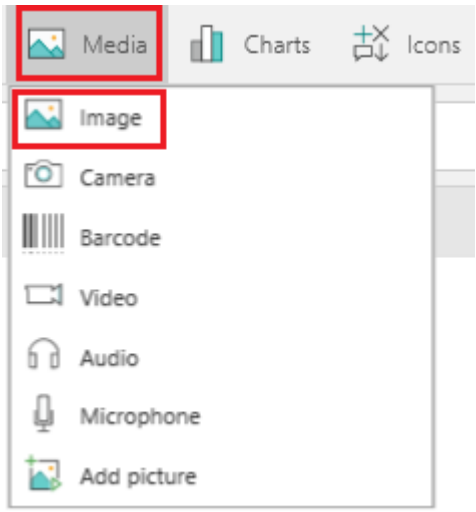
Basically, you can use this anywhere users would benefit from a form or labels that are populated automatically

Prerequisites

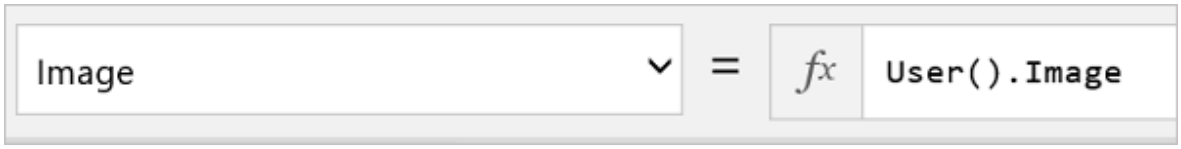
1. Create a blank canvas app.
2. Learn how to add and configure controls.

Show user details

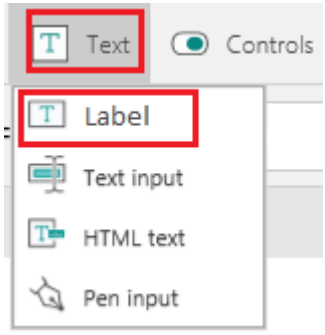
1. On the **Insert** tab, click or tap **Media**, and then click or tap **Image**.



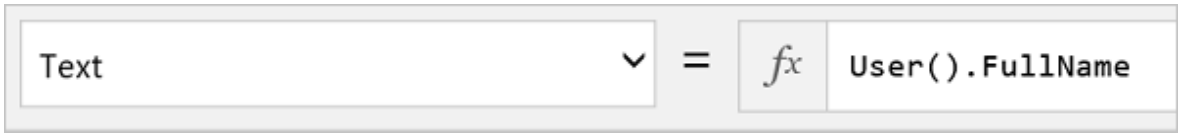
2. Set the Image property to this formula:
   User().Image



3. On the **Insert** tab, click or tap **Text**, and then click or tap **Label**:



4. Set the Text property to this formula:
   User().FullName



   When you do this, the label is automatically populated with your full name. Move the label so it's below the image control, similar to the following:



5. Add another label, and set its Text property to this formula:
   User().Email



   When you do this, the label is automatically populated with your email address. Move the label so it's below the first label, similar to the following:



Your Full Name

Your Email Address

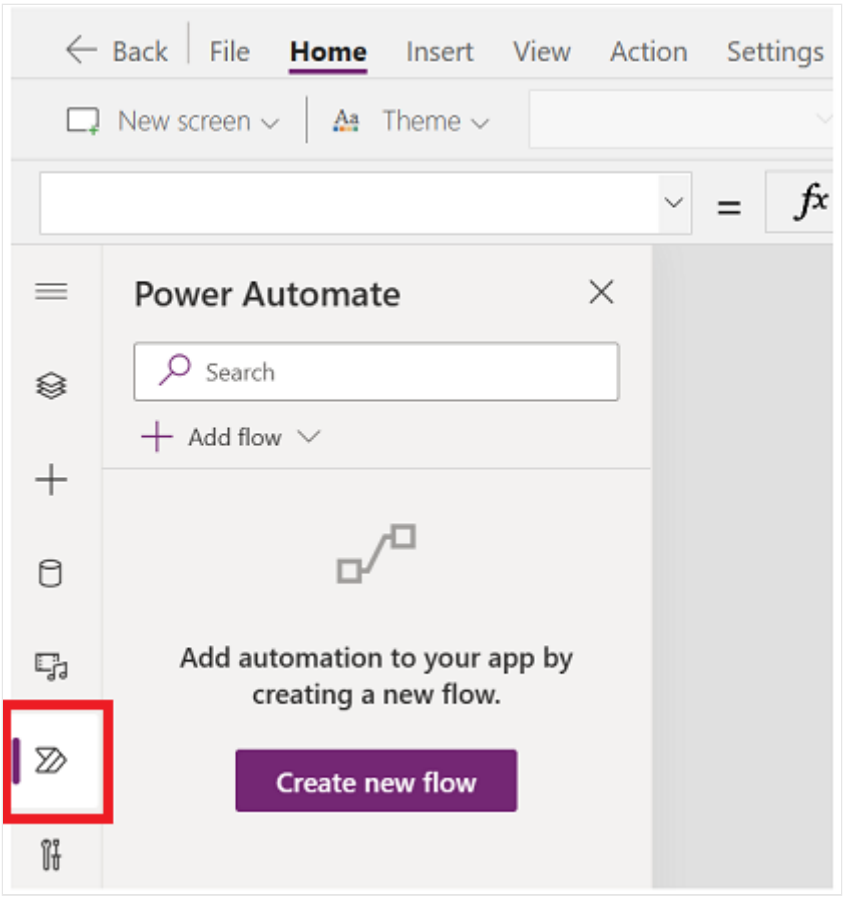
Use Power Automate pane

Create new flows or add and edit existing flows using the Power Automate pane within Power Apps Studio. The Power Automate pane is enabled by default.

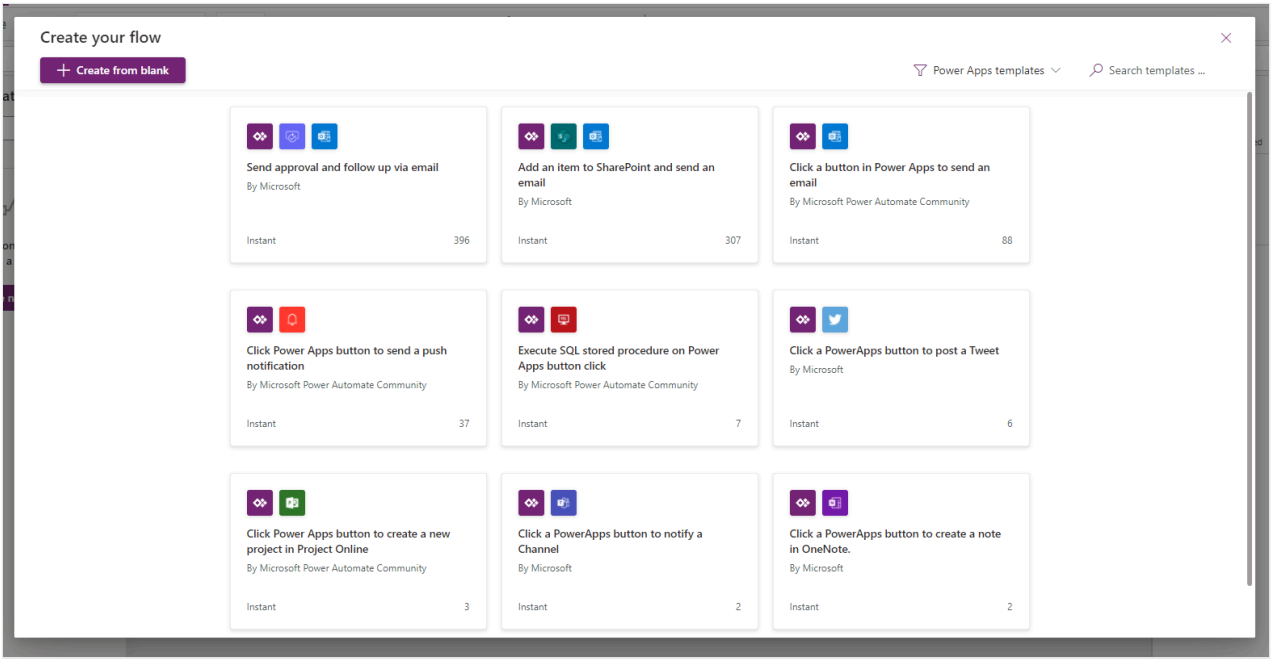
> **ⓘ Important**
>
> Arguments that pass from Power Apps to Power Automate are visible as network traffic and can be intercepted. Most traffic is likely harmless data that are irrelevant outside the application, for example arguments like "yes" or "Redmond".
>
> Evaluate the parameters passed to Power Automate and consider the consequences (if any) if they're changed by an exernal actor. To mitigate risks, validate the parameter values passed.
>
> For example, if you send sensitive data to a person in your organization through email, for example `someuser@contoso.com`, validate the address in Power Automate. You can check the incoming arguments to confirm the organization, `@contoso.com`, and only allow the flow to proceed if the right organization is present.

Create a new flow

1. Open a new or an existing app in Power Apps Studio.
2. On the app authoring menu, select **Power Automate**.



3. In the Power Automate pane, select **Create new flow**.



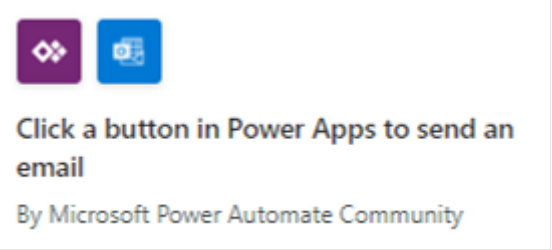
This action opens the **Create your flow** modal dialog within Power Apps Studio. You can choose to create the flow from the available templates, or create a flow from scratch.

Start with a template

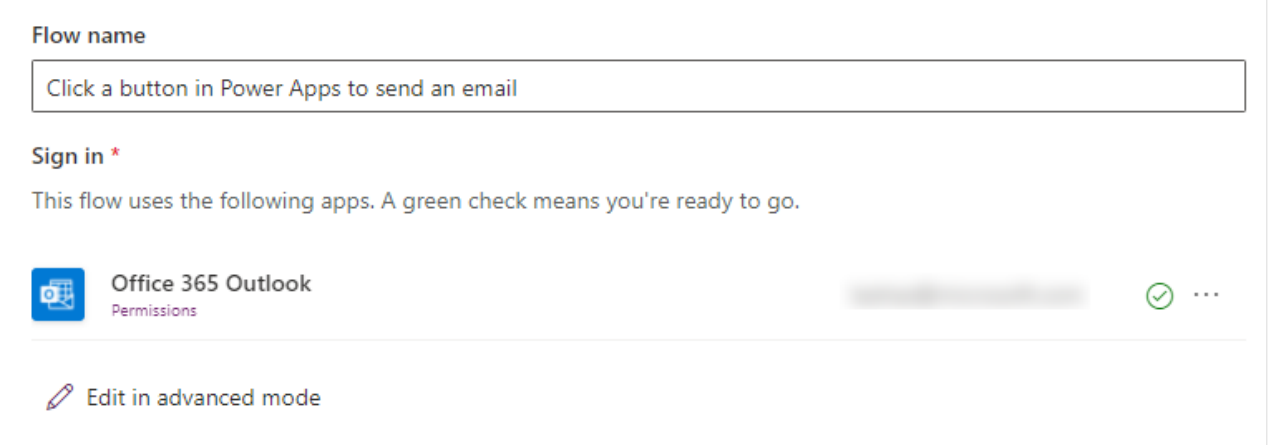
Power Automate provides several flow templates that you can choose from by default. This section explains how to create a flow using such available templates from within Power Apps

Studio.

1. From the list of available templates, select a template of your choice. For this example, we use **Click a button in Power Apps to send an email**.



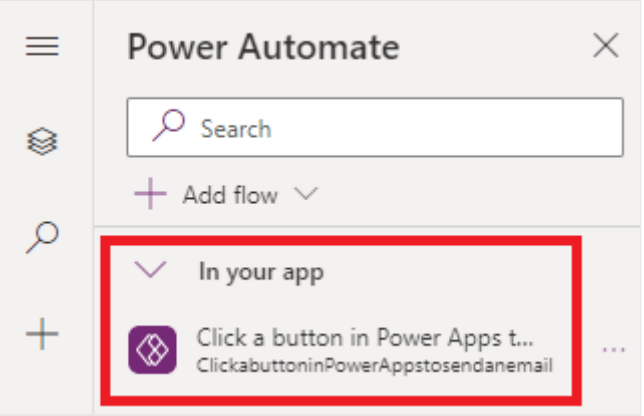
2. The next step shows the connections required to create a flow based on the selected template. More information: Connectors overview



3. (Optional) You can use **Edit in advanced mode** to update the connection, and customize the flow template. When done, select **Save** to save the flow. Skip the next step since your flow creation process is now complete.

4. Select **Next** > **Create** to create the flow.

   The flow now appears in the list of flows inside the Power Automate pane, under the **In your app** section.



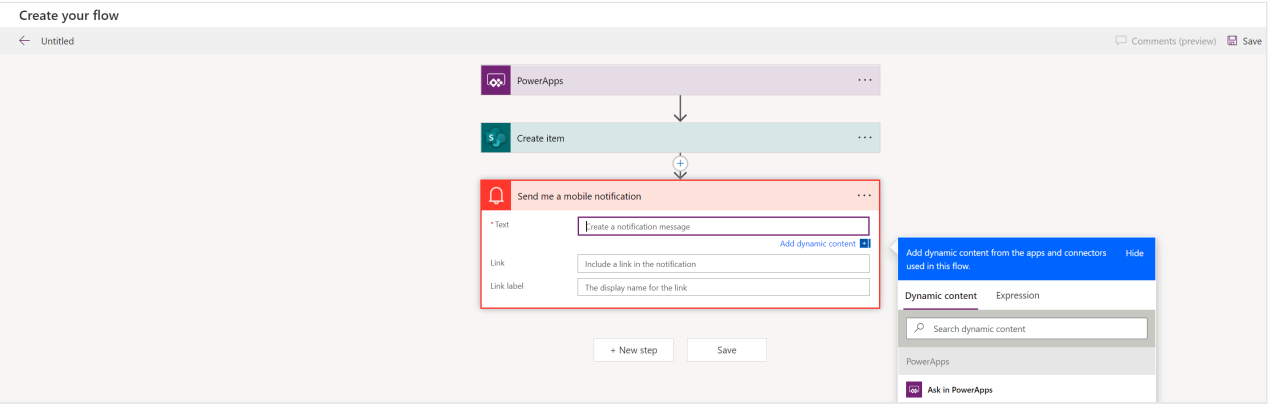
> **Tip**

For more information about creating cloud flows from templates, see Create a cloud flow from a template.

Create from blank

Instead of using the available templates, you can create a flow from scratch, and customize the steps as per your requirement.

1. Select **Create from blank**.
2. Add and customize the flow with steps and actions, as shown in the following example.

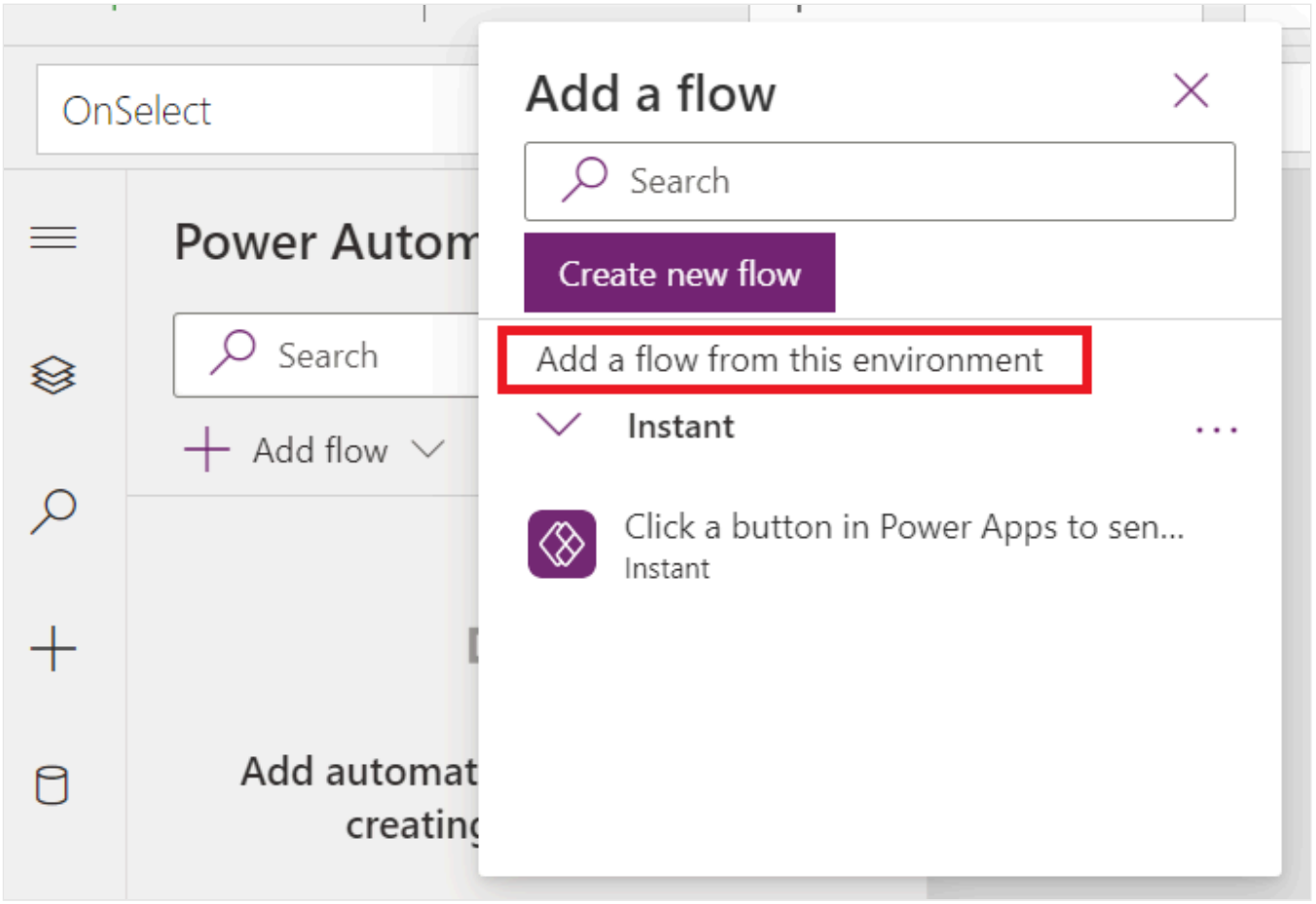


3. When done, select **Save** from upper right corner of the screen.

> **Tip**
>
> For more information about customizing and configuring cloud flows, see the **How to** articles starting from Add multiple actions and advanced options to a cloud flow.

Add an existing flow

You might have access to the existing flows in the environment where you're creating your app. To add and use these flows in your app, select **Add a flow**, and then choose the flow from the **Add a flow from this environment** section.



> **Tip**
>
> You can hover over the flow to see more details before adding to your app.

You must meet the following requirements to be able to add an existing flow to a canvas app:

- Have access to the flow
- Have access to see the flows that are part of the same solution
- Have a Power Apps trigger for your flows.

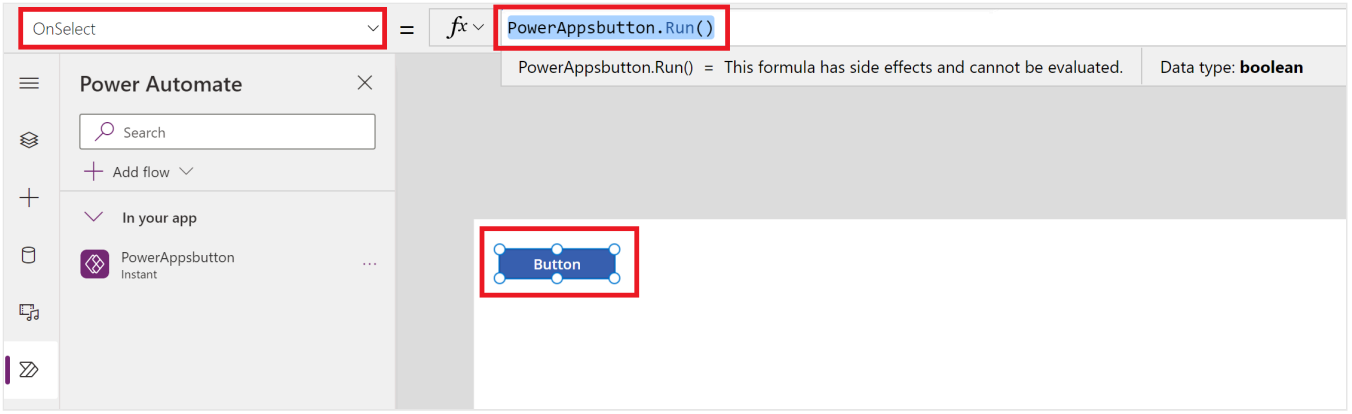
After being added, the flow will appear under the **In your app** section inside the Power Automate pane.

Reference a flow

To reference the added flow, update the formula bar for the control or component within Power Apps Studio with the flow details. For example, to reference a flow named "PowerAppsbutton" that triggers an email without accepting any input parameters, use the following formula on the button's **OnSelect** property:

```
Power Fx
```

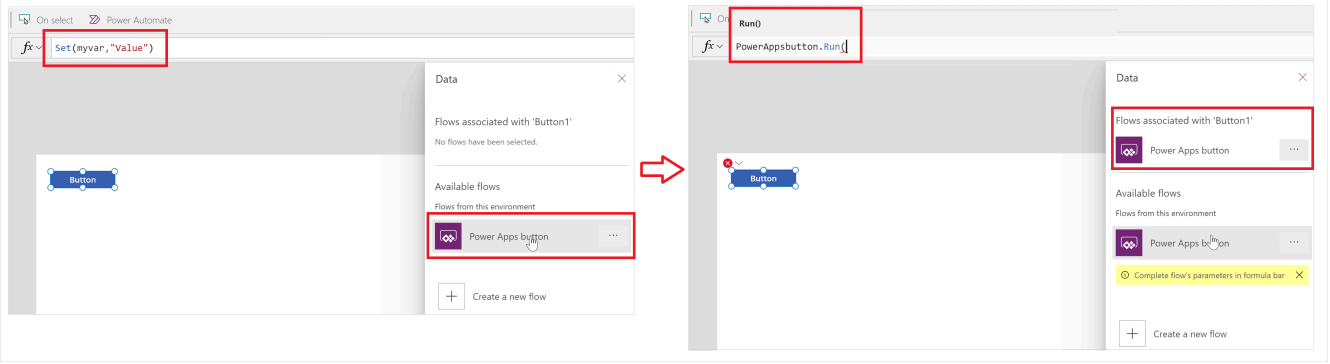
```
PowerAppsbutton.Run()
```



This behavior is different from how the reference to a flow works within Power Apps Studio if the Power Automate pane is disabled.

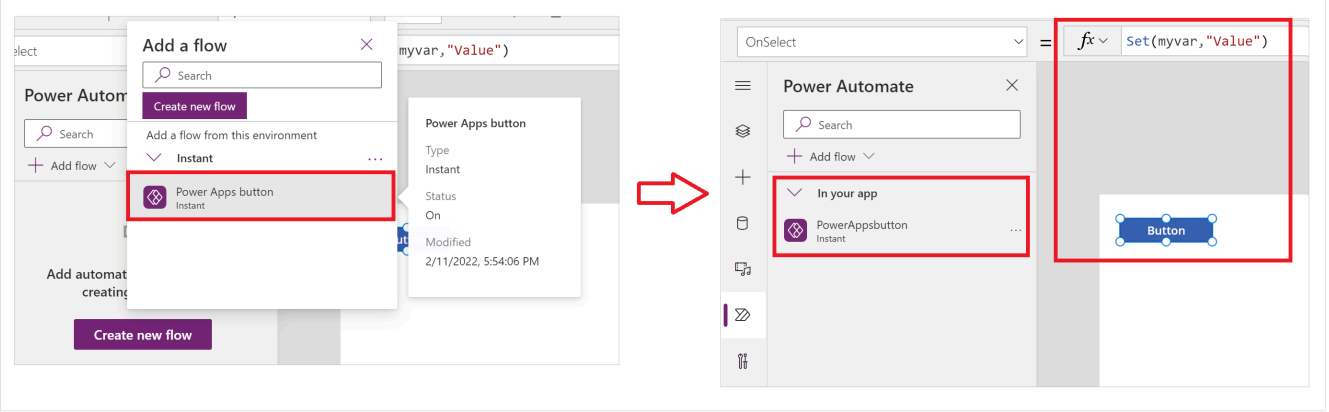
When you add a flow with the Power Automate pane disabled, you must choose a behavior property of the control or the component that you want to associate with the flow. Adding a flow, in this case, clears any formula already associated with the chosen property (when the flow reference is added). Any existing formula gets removed, and you must carefully make a copy of the existing formula before adding the flow in order to preserve it. You can then paste your copied formula back after the flow reference is added.

For example, on a button's OnSelect property, adding the flow removes your existing formula, and you have to carefully make a copy of the existing formula before adding the flow. The following example shows the process with the Power Automate pane disabled. In this case, the formula added for the button property is removed to reference the added flow.



Whereas with the Power Automate pane enabled, any existing formula is preserved automatically (isn't removed). You can add the flow reference in the formula for a behavior property as per your requirement.

The following example shows the process with the Power Automate pane enabled. When the flow is added, the existing formula for the button's OnSelect property is preserved, and the flow reference isn't added automatically.



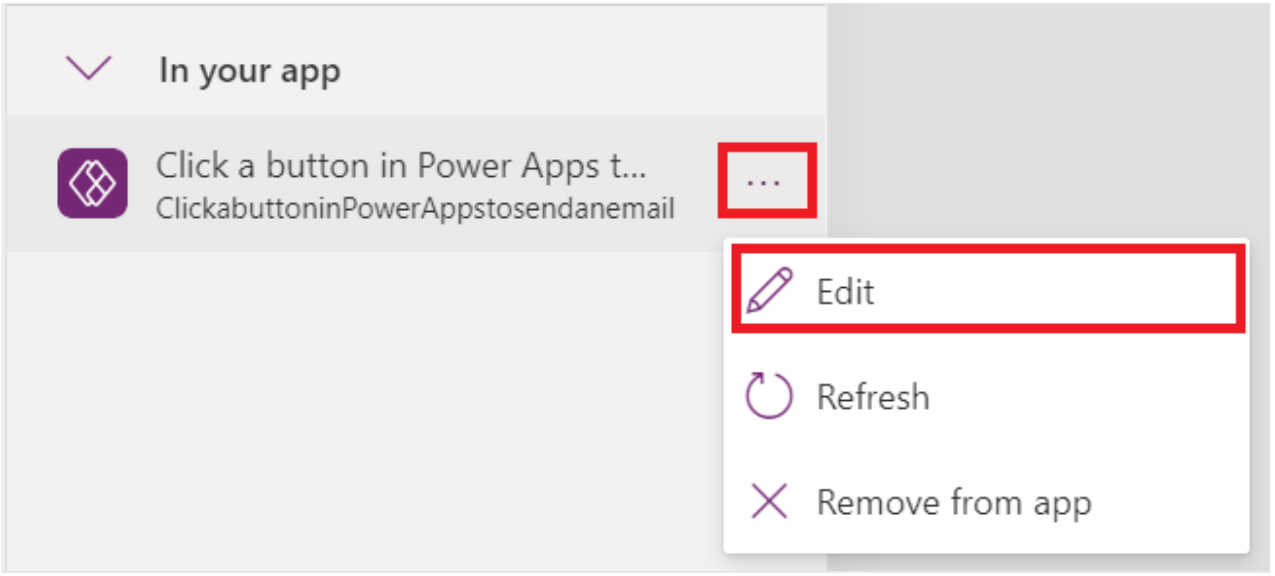
Now you can reference the flow through the regular process of updating the formula for the control or component—as explained earlier.

Edit an existing flow

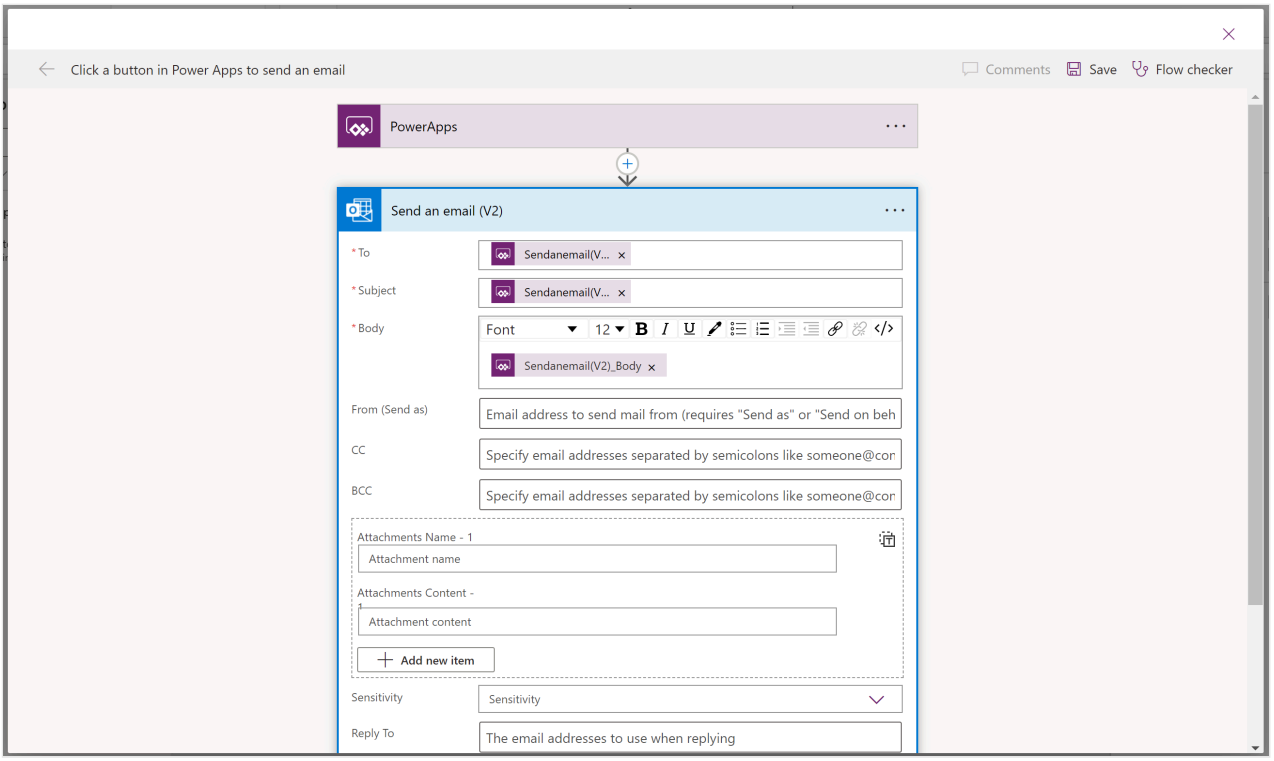
You can now edit flows added to your app without leaving Power Apps Studio.

1. Under the **In your app** section of the Power Automate pane, select **...** (ellipsis) to the right of a flow.

2. Select **Edit**.



The flow opens inside the editor.



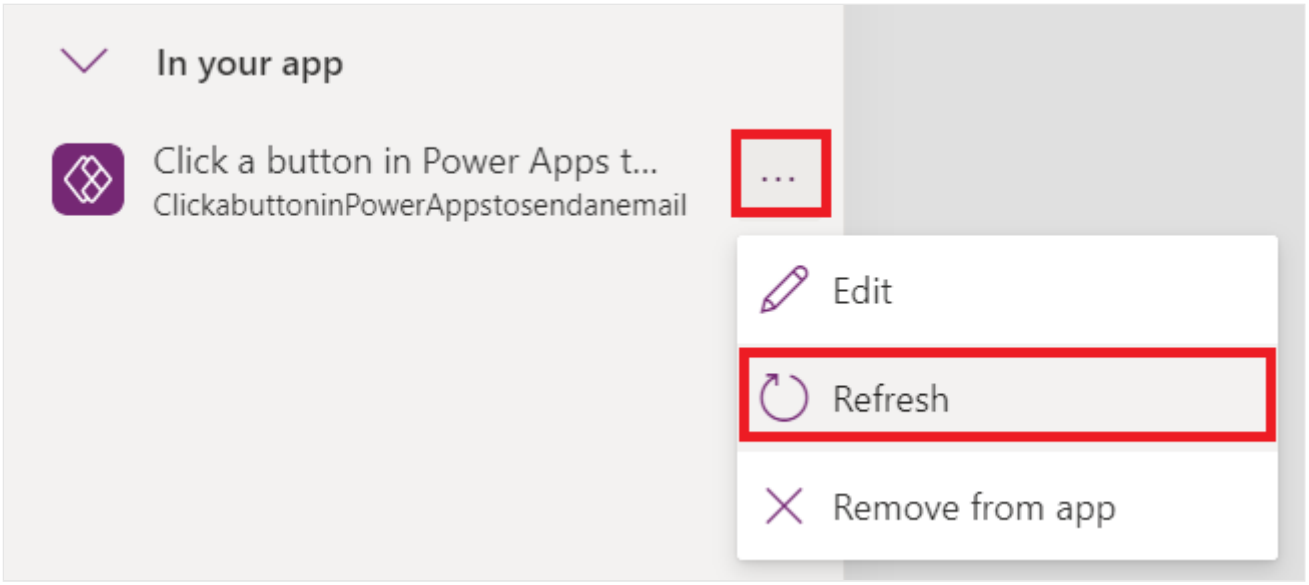
3. Make your changes, and then select **Save**.

> **Important**
>
> To avoid losing unsaved changes, ensure you select **Save** before closing the editor.

Refresh a flow

If any changes are made to the flow in Power Automate (instead of using Power Apps Studio), refresh your flow to get the latest changes.

To refresh the flow and pull the latest changes, select **...** (ellipsis) to the right of your flow, and then select **Refresh**.

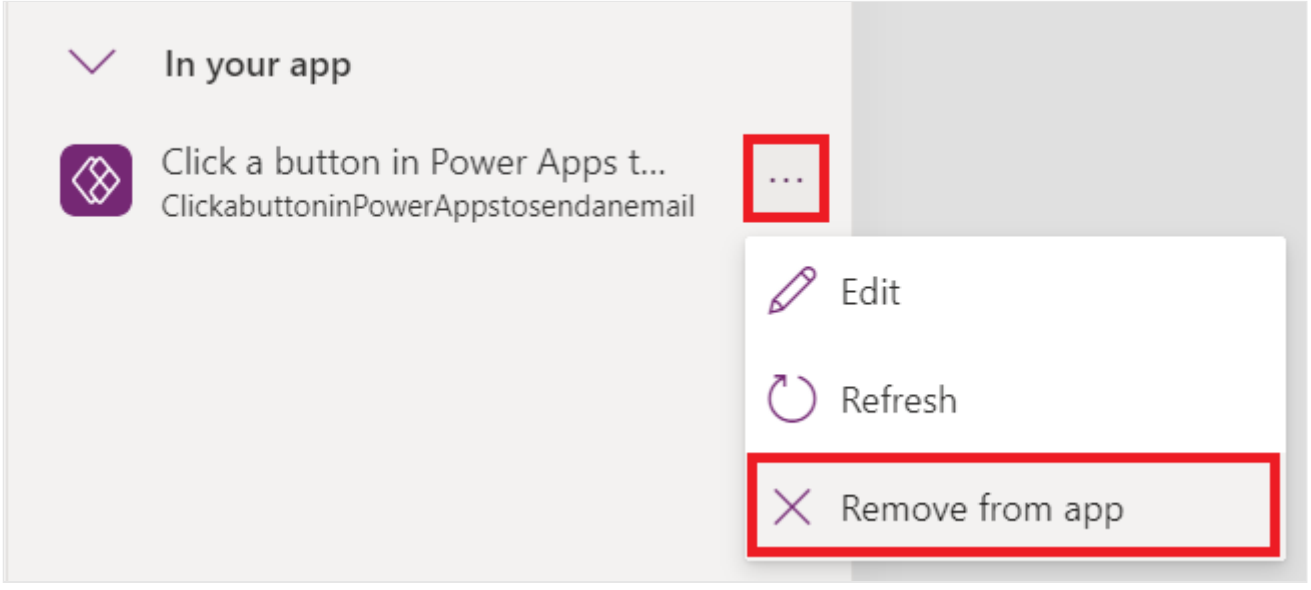


A loading spinner appears, and the flow is refreshed.

> **Important**
>
> After making changes to a flow outside of Power Apps Studio, you must perform this step; otherwise, the flow may fail to run—even in previously published apps. Skipping this step can also lead to performance issues in the Power Platform admin center, as each time the canvas app runs, it will try to validate the flow's connections using outdated information, which may not succeed.

Remove a flow

To remove a flow from your app, select **...** (ellipsis) to the right of your flow, and then select **Remove from app**.



This action only removes the flow from the app, while the flow remains intact in the environment. You can add the flow again to the same app, or use it in other apps.

Classic Power Automate experience

The Power Automate pane is now enabled by default. To use the classic Power Automate experience to create a flow, switch back to the old experience manually.

> **Important**
>
> It's recommended that you use the Power Automate pane to create a flow. The classic Power Automate experience should only be used for troubleshooting. The classic experience will be retired soon and won't be available.

1. Open a new or an existing app in Power Apps Studio.
2. Select **Settings** at the top.
3. Select **Upcoming features**.
4. Under the **Retired** tab, select **Enable Classic Power Automate pane** to set the toggle to **On**.

See also

- Add and configure controls

Last updated on 11/10/2025

Start a flow in a canvas app (retired)

Article • 12/18/2024

You can use Power Automate to create logic that performs one or more tasks when an event occurs in a canvas app. For example, you can configure a button that performs a single task or multiple tasks. When the button is pressed, an item is created in a list (Microsoft Lists), an email or meeting request is sent, and a file is added to the cloud. You can configure any control in the app to start the flow, which continues to run even if you close Power Apps.

> **Note**
>
> When a user runs a flow from within an app, that user must have permission to perform the tasks that are specified in the flow. Otherwise, the flow will fail.

Enable classic Power Automate experience

By default, the new Power Automate pane is enabled by default. To use the classic Power Automate experience to create a flow, switch back to the old experience manually.

> **Important**
>
> It's recommended that you use the **Power Automate pane** to create a flow. The classic Power Automate experience should only be used for troubleshooting. The classic experience will be retired soon and won't be available.

1. Open a new or an existing app in Power Apps Studio.
2. Select **Settings** at the top.
3. Select **Upcoming features**.
4. Under the **Retired** tab, select **Enable Classic Power Automate pane** to set the toggle to **On**.

Prerequisites

- Sign up for Power Apps.
- Learn how to configure a control.

- A SharePoint site with two lists.
  - List **RepairShop** contains Title, and ContactEmail columns.
  - List **Assets** contains Title, AssetType, and RepairShop columns. The AssetType column is a choice column with choices such as "Desktop" or "Laptop".
  - The RepairShop column in Assets list is a lookup column that gets values from the ContactEmail column in RepairShop list.
  - Create a few items in the **RepairShop** list for sample contacts.

Create a flow

In this section, you create a flow using Power Automate that creates an item in a list using the input value for the list column from an app created using Power Apps. You create the app that uses this flow in the next section.

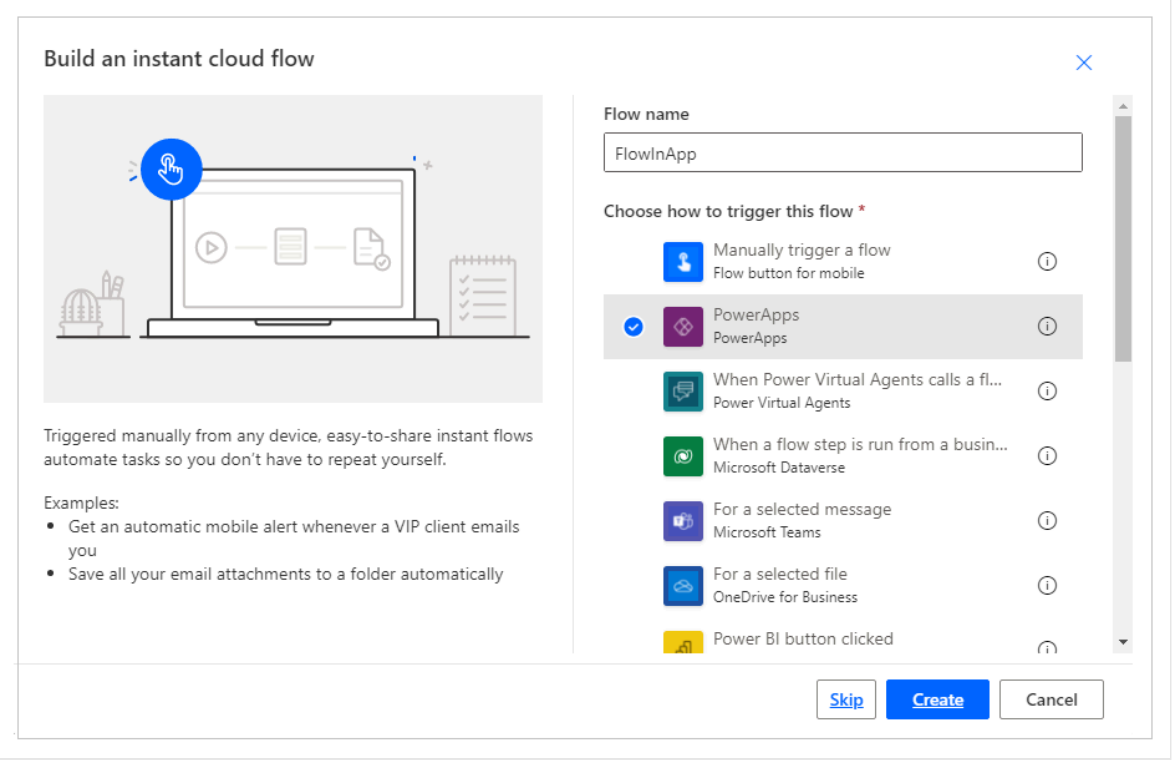
1. Sign in to Power Apps.

2. On the left-pane, select **Flows**.

3. Select **+ New flow** > **Instant cloud flow**.

4. Enter flow name, such as **FlowInApp**.

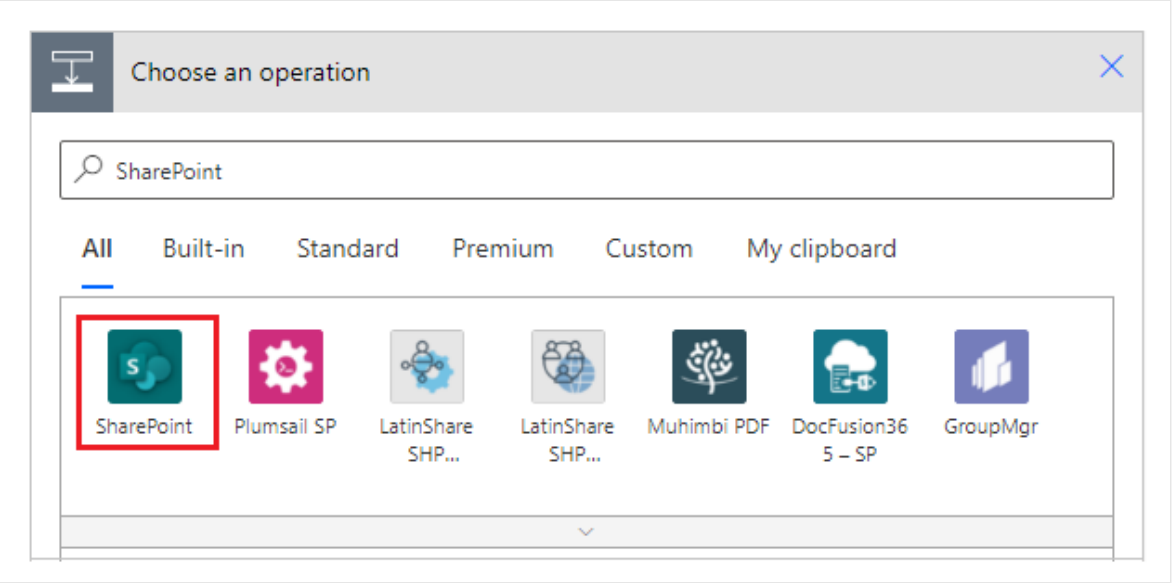
5. Under **Choose how to trigger this flow**, select **Power Apps**.



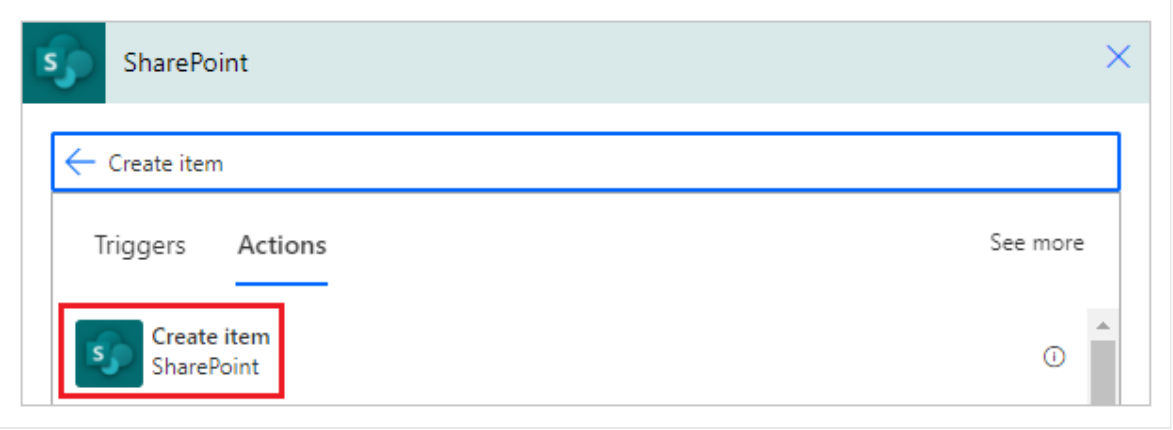
6. Select **Create**.

7. Select **+ New step**.

8. Under **Choose an operation**, search for or select **SharePoint** connector.

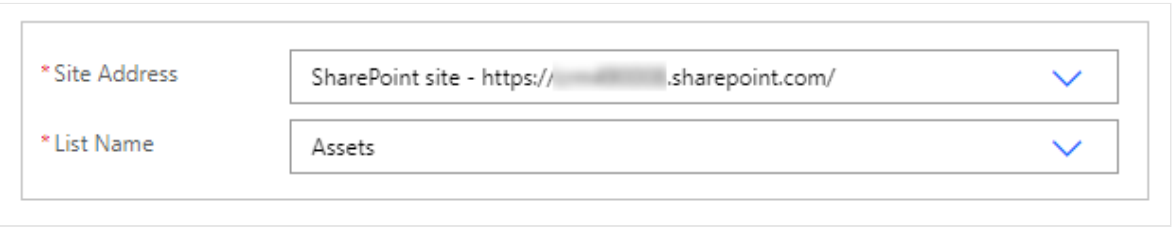


9. Search for, or select **Create item** action.

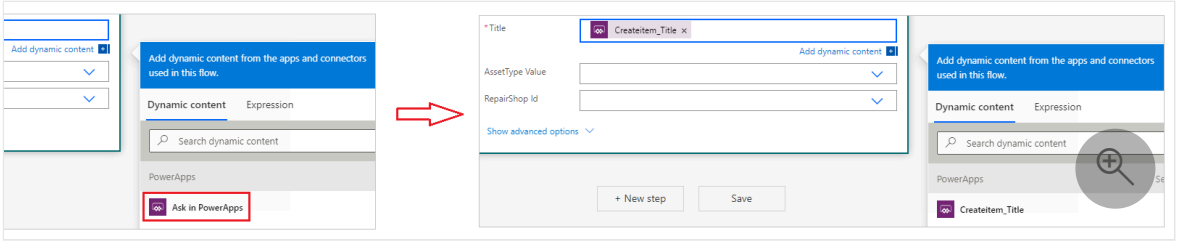


10. For **Site Address**, enter or choose your SharePoint site that has the lists **Assets** and **RepairShop**.

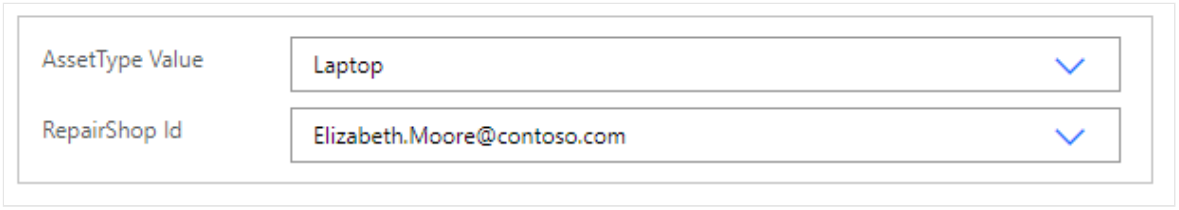
11. For **List Name**, select the **Assets** list.



12. For **Title**, choose **Ask in Power Apps** from the **Dynamic content** tab. The action automatically updates to **Createitem_Title**.



13. Choose values for **AssetType Value**, and **RepairShop Id** of your choice.



14. Select **Save**.

> **ⓘ Important**
>
> Arguments that pass from Power Apps to Power Automate are visible as network traffic and can be intercepted. Most traffic is likely harmless data that are irrelevant outside the application, for example arguments like "yes" or "Redmond".
>
> Evaluate the parameters passed to Power Automate and consider the consequences (if any) if they're changed by an exernal actor. To mitigate risks, validate the parameter values passed.
>
> For example, if you send sensitive data to a person in your organization through email, for example `someuser@contoso.com`, validate the address in Power Automate. You can check the incoming arguments to confirm the organization, `@contoso.com`, and only allow the flow to proceed if the right organization is present.

Add a flow to an app

In this section, you create an app using Power Apps that uses the flow created in the earlier section. The app uses text entered in a text box when the button is selected to trigger the flow to create an item in the selected list.

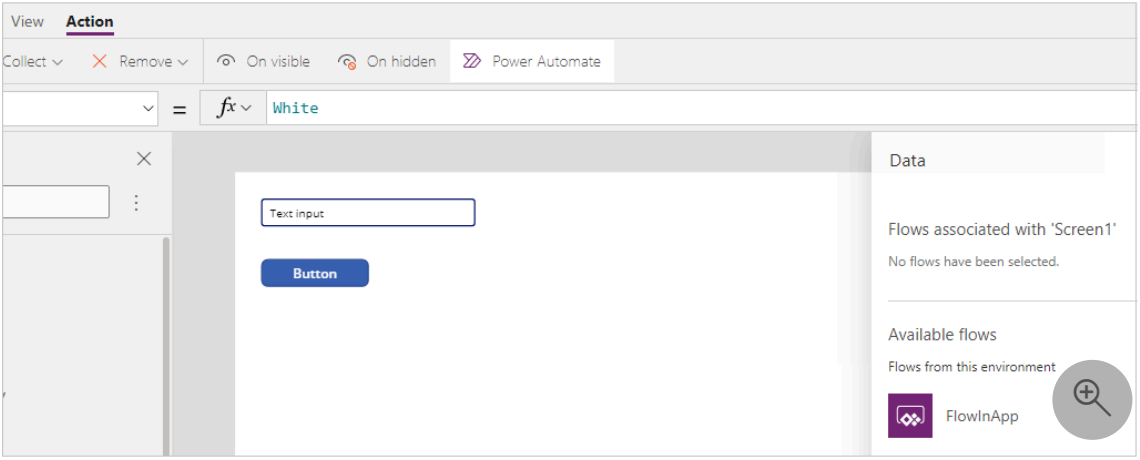
1. Create a blank canvas app with a name such as **AppWithFlow**.
2. Select + (Insert) on the left-pane.
3. Select **Text input** control.
4. Select **Button** control.

5. On the canvas, move the button control below the text input control.

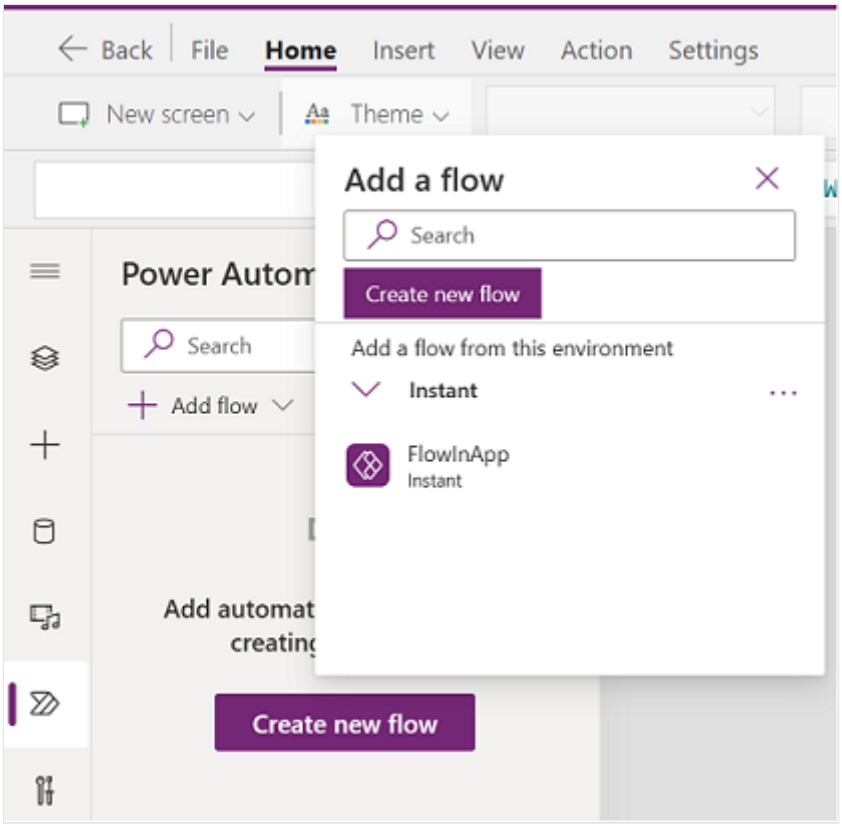


6. Select **Action** menu at the top, and then select **Power Automate**.

> **Note**
>
> Adding a flow to the selected control or component clears out any existing formula for the chosen property. For example, when you add a flow to a button's `OnSelect` property that has a complex formula, the flow addition clears out this formula. Ensure you make a copy of the formula before adding the flow. However, this behavior is different when adding flow with the Power Automate pane enabled. Learn more in Reference a flow.



Alternatively, if you enabled the **Enable Power Automate pane** preview feature, you can choose your flow from the left-pane inside Power Apps Studio.



More information: Use Power Automate pane (preview)

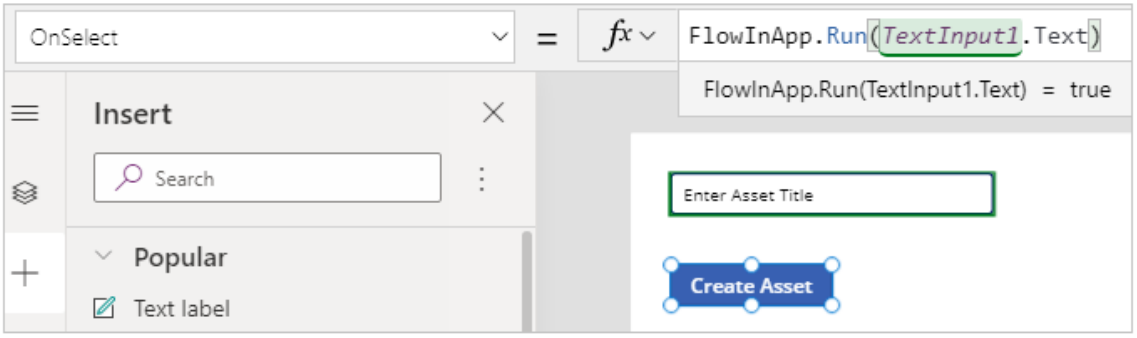
7. Select **FlowInApp**.
8. In the formula bar, remove the formula for the selected **OnVisible** property.
9. Select the text input control.
10. From the top-left side of the screen, select the property list drop-down, and then select the **Default** property.
11. In the formula bar, change the default property value from `"Text input"` to `"Enter Asset Title"`.
12. Select the button control.
13. From the top-left side of the screen, select the property list drop-down, and then select the **Text** property.
14. In the formula bar, change the text property value from `"Button"` to `"Create Asset"`.
15. From the top-left side property list, select the **OnSelect** property for the button.
16. Enter the following formula in the formula bar.

```
Power Fx
```

```
FlowInApp.Run(TextInput1.Text)
```

In this formula, **FlowInApp** is the name of the flow you added using Power Automate. The **.Run** specifies the flow to execute. The flow executes with **TextInput1** text input control added to this canvas, with the value entered in this text box (**.Text**).

When this button is selected, the app runs the flow with the value from the text input control, passing the text value to the flow to execute. And the flow creates the list item with this text input value along with rest of the selection inside the flow configuration.

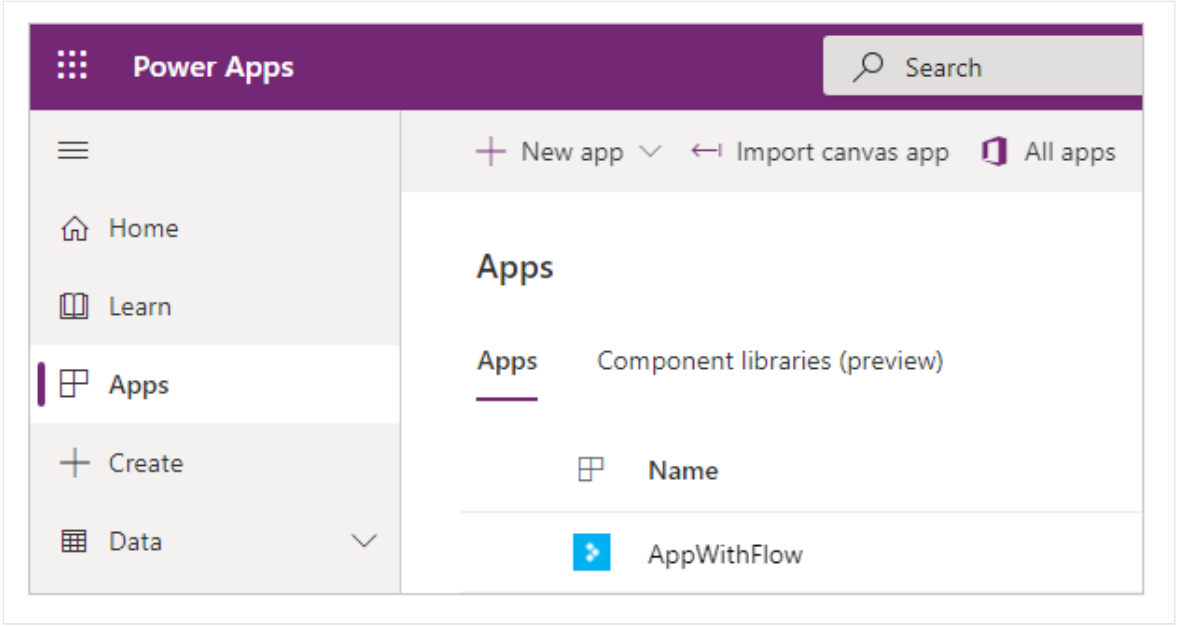


17. Save and publish the app.

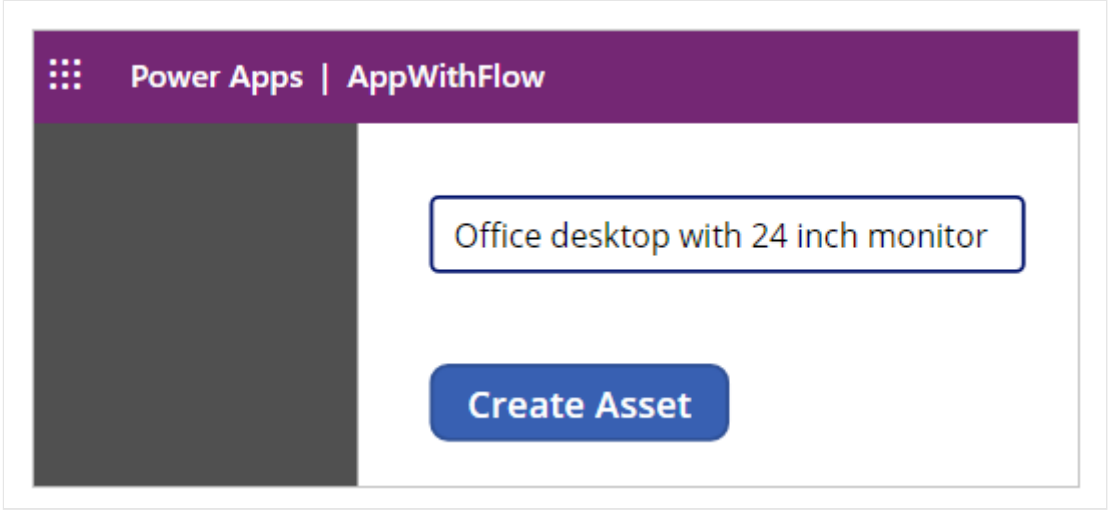
Test the flow

Now that you have both the flow and the app created, run the app and verify the creation of an item inside the selected list.

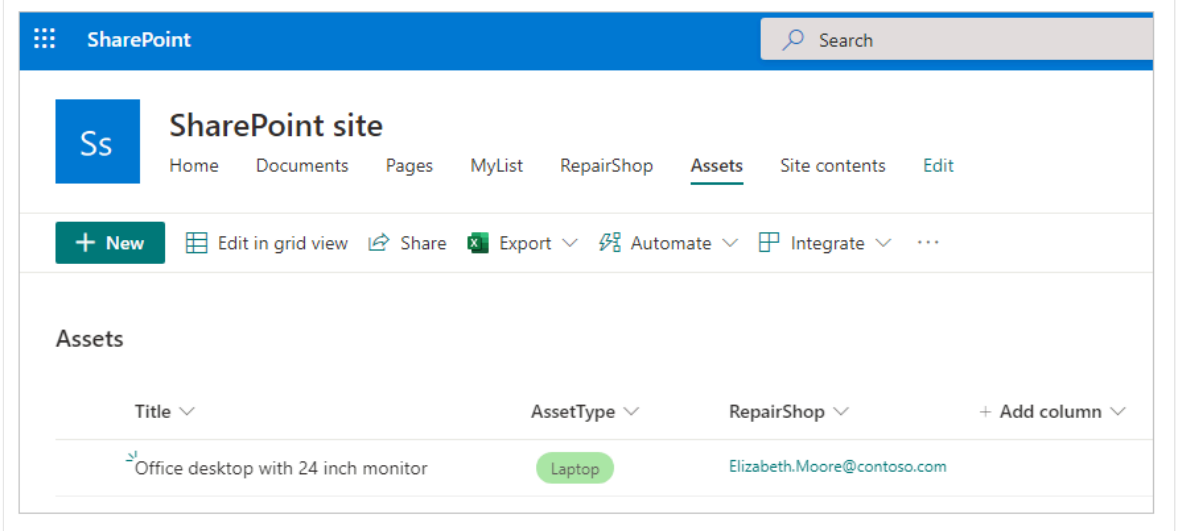
1. In Power Apps, select **Apps**, and then, select the **AppWithFlow** app.



2. Enter a value in the text input box, and then select **Create Asset**.



3. Verify that the item is created in your list.



Now that you created a sample app that runs a flow and adds an item inside a list, you can create more complex applications. Your apps can interact with Power Automate and

manipulate data inside various data sources.

Related information

- Use Power Automate pane (preview)
- Add and configure controls

Feedback

Was this page helpful? Yes No

Provide product feedback

Convert rules to expressions

Article • 03/16/2023

The capability to create rules in canvas apps for automated app modification based on specified criteria will be removed. This feature was retired in 2019 and will now be fully removed.

You'll have a few months to convert the rules into expressions using a converter. If there are any rules remaining in your canvas app that haven't been converted by the end of this period, the system will perform the conversion automatically.

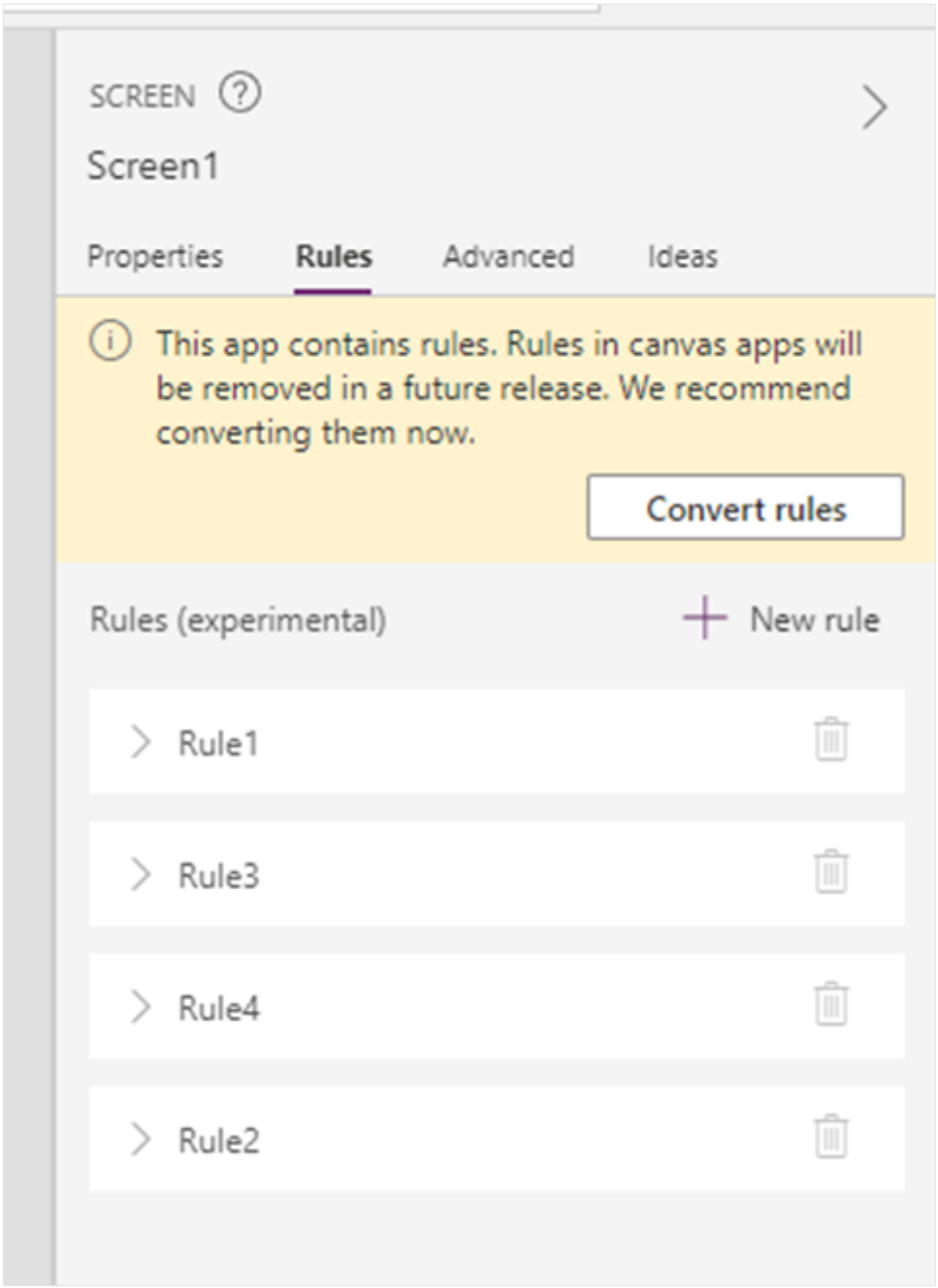
> **Important**
>
> Effective October 14, 2019, the rules feature in canvas apps is deprecated. More information: **Blog: Canvas Rules feature deprecation**.

Convert rules

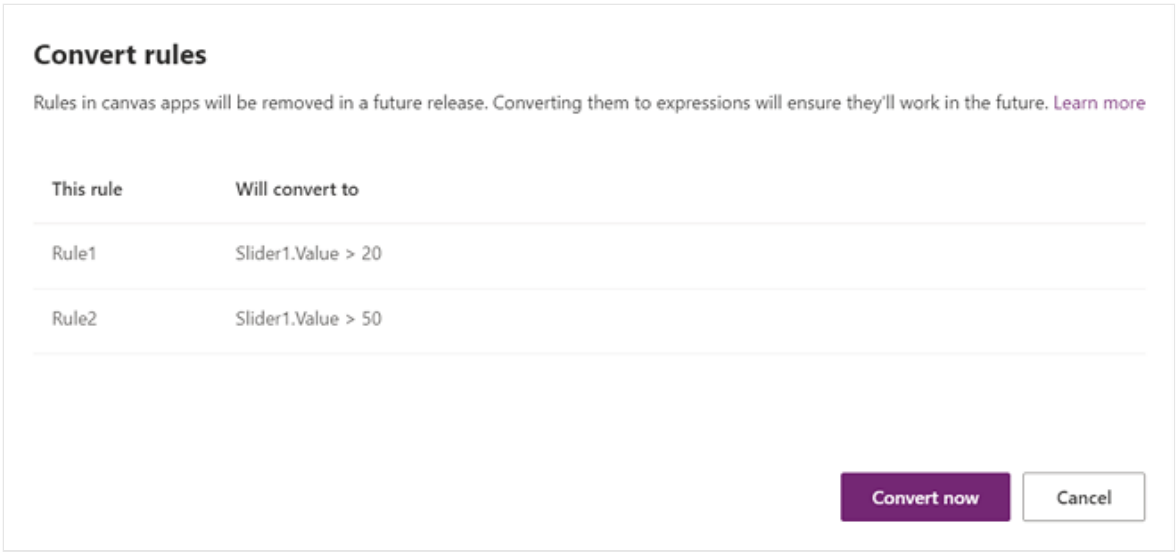
When you edit an app that has rules, you'll receive a prompt to convert them. The converter helps in migrating the rules within your app to a format that is compatible with future versions of Power Apps Studio. By utilizing the rules conditions in your app, the converter substitutes the references with corresponding inline expressions.

Follow these steps to convert rules in your app:

1. Open your app for editing and go to the Rules panel.
2. On the warning message, select **Convert rules**.



3. A dialog opens, which shows all the rules which will be converted. The rule name is shown on the left, and its associated conditional expression is shown on the right. On the **Convert rules** dialog box, select **Convert now**.



When the conversion is complete, a notification appears to alert you whether the conversion was successful. Formulas that previously referenced rule names have been updated to directly use their associated conditional expression instead.

Before the conversion, the Power Fx formula references rule names in the formula in place of their associated conditional expressions.

```
If(Rule2, DarkSeaGreen
, If(Rule1, DarkRed

, RGBA(56, 96, 178, 1)

)
)
```

After the conversion, rule name references have been removed and replaced. In this example, **Rule2** was converted to **Slider1.Value > 50**.

```
If(Slider1.Value > 50, DarkSeaGreen
, If(Slider1.Value > 20, DarkRed

, RGBA(56, 96, 178, 1)

)
)
```

Reverting the conversion

When the conversion is done, the rules are converted to the corresponding expression. During the conversion period, you can revert your changes to bring the rules back to your app for review and troubleshooting if necessary. While in the same session, you can use the Undo button or Ctrl-Z to revert the rule conversion. If you save the app, you can restore the previous version to revert the change. However, you will be prompted to convert the rules again the next time the app is opened for editing.

Known Issues

If your rules are in an error state or are empty, they will be replaced with the following formula: Boolean(Blank()). This formula preserves the behavior of rules in this state.

Create or update bulk records in Power Apps

Article • 12/16/2022

Updating or creating records in bulk allows you to act on many records at once. Here are some scenarios where you would want to bulk *update* records:

- You have many photos and want to upload them all at once.
- You want to submit a time sheet with a week's data at once.

You might also want to *create* new records in bulk. For example, when your app has to select many images, you might want to upload them all at once.

This article explains how to update or create bulk records depending on your scenario.

Formulas to bulk update records

The formulas in this section can be used to bulk update records in canvas apps.

- Patch() function—Use this function when the collection matches the data source.

  Power Fx

  ```
  Patch( DataSource, Collection )
  ```

- ForAll() function + nested Patch + disambiguation operator—Use this function when the data sources have different columns that you need to join.

  Power Fx

  ```
  ForAll( Collection,
      Patch( DataSource,
          LookUp( DataSource, Id = Collection[@Id] ),
          { Column: Value }
      )
  )
  ```

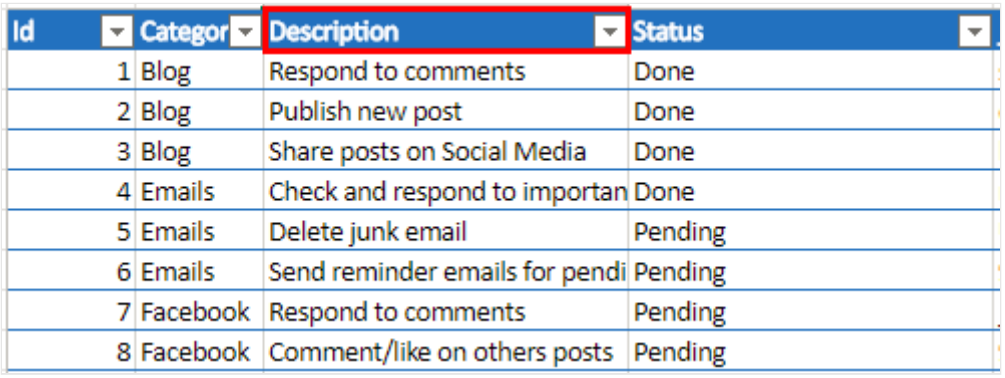
- AddColumns() function—This function can be used to provide a lookup reference in the Collection that contains the updates to the DataSource if it doesn't have fields that easily reference the table.

Example of a checklist

This example uses a checklist of tasks. When you're done with a few tasks, you can mark them as complete. You could extend this scenario to a product launch checklist, home inspection checklist, or other lists.

The following screenshot shows the checklist items in Microsoft Excel file. You can check off the tasks related to your blogging or posting on social media in this example. The table in Excel is called `ChecklistItems`.

The example uses collections to demo this behavior. The approach works for any tabular backend of your choice.

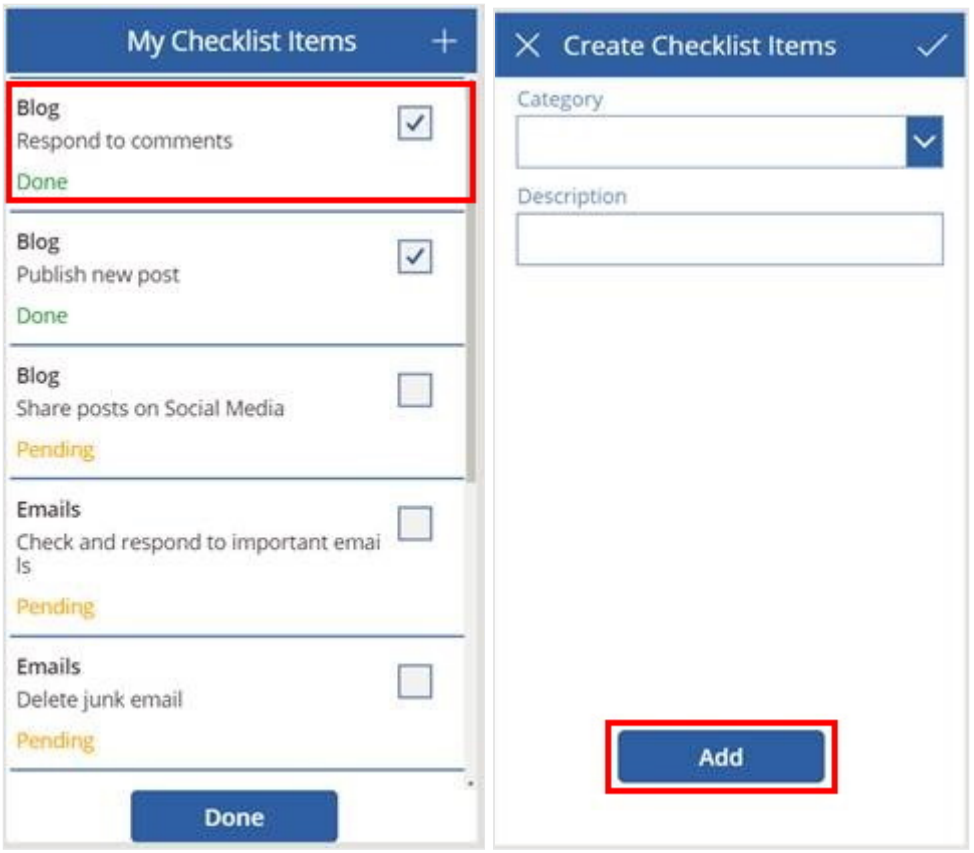


Id	Categor	Description	Status
1	Blog	Respond to comments	Done
2	Blog	Publish new post	Done
3	Blog	Share posts on Social Media	Done
4	Emails	Check and respond to importan	Done
5	Emails	Delete junk email	Pending
6	Emails	Send reminder emails for pendi	Pending
7	Facebook	Respond to comments	Pending
8	Facebook	Comment/like on others posts	Pending

The Excel data is imported as static data into the app. On the `App.OnStart` property, `Collect()` the Excel ChecklistItems data into the `ChecklistItemsSource` collection. The `ChecklistItemsSource` collection will be used as the source data throughout the app.

This example uses a two screen app:

- **Screen 1**—Used to review the checklist items and mark them as complete.
- **Screen 2**—Used to create a new checklist item. The new item will be added to `ChecklistItemsSource` collection.



Every time a checklist item is checked, it's added into a collection called `CheckedItems`. It uses the formula below on the `OnCheck` event property of the checkbox control. If the item is already checked and is part of the collection, it's removed. If not, the checked item is added.

You can toggle the status between **Done** and **Pending**, or you can use the `Oncheck` and `OnUncheck` events:

Power Fx

```
If( !IsBlank(
        LookUp( CheckedItems, Id = ThisItem.Id )
    ),
    Remove( CheckedItems, ThisItem ),
    Collect( CheckedItems, ThisItem )
)
```

Create multiple records at once

There are different ways of bulk updating records depending on the scenario. This section explains how to create multiple records at once in canvas apps.

When the user selects **Done** in the above example, you need to update `ChecklistItemsSource` with changes from the CheckedItems collection.

For columns with the same name

If your source and destination have the same column names, you can use a Patch statement. For example, `ChecklistItemsSource` and the `CheckedItems` collections have the same column names. You can use the formula below to update the source at once with all the changes.

Power Fx

```
Patch( ChecklistItemsSource, CheckedItems )
```

For columns with different names

If the columns in source and destination tables vary, use ForAll with Patch instead.

With `ForAll()`, you must loop through each record using a condition. The condition is a comparison between similar columns (for example, `Id` column) of the different tables. This comparison becomes complicated when the source table and the destination table have the same column names (for example, if `ProjectId` was a column found in both `Project` and `PurchaseOrder` tables).

Here are the alternatives:

Use with disambiguation operator

To update the `Status` of `CheckedItems` to "Done", when the source and destination table column names are the same, use this formula:

Power Fx

```
ForAll( CheckedItems,
    Patch( ChecklistItemsSource,
        LookUp( ChecklistItemsSource, Id = CheckedItems[@Id] ),
        { Status: "Done" }
    )
)
```

For each item in the `CheckedItems` collection, the Id (represented by the disambiguation operator `CheckedItems[@Id])` is compared against the Id column of the `ChecklistItemsSource` collection. Then, each matched record is updated with the status as "Done". The disambiguation operator is used when two columns belonging to

different tables have the same name. If you don't use this disambiguation operator, you'll see that only the first record gets updated.

Use an extra label within the gallery

If you don't want to use an extra collection to store the checked items, you can try the following steps:

1. Create an extra label within the gallery template.
   a. Bind it to the Id column.
   b. Rename the label to IdText.

2. Remove the code on the **OnCheck** of the checkbox control mentioned above.

3. Write the following formula on the OnSelect event of the Done Button:

```
Power Fx
```

```
ForAll(
    Filter( ChecklistGallery.AllItems,
      StatusCheckbox.Value = true
    ),
    Patch( ChecklistItemsSource,
      LookUp( ChecklistItemsSource, Id = IdText.Text ),
      { Status: "Done" }
    )
)
```

Here, you're directly applying the filter on the Gallery's items to find the checked items. For each record in the filtered items, a match is found on the **ChecklistItemsSource** table by comparing the Id with value stored in IdText label.

The status is updated to "Done".

The disambiguation operator can't be used on the Gallery's items. Instead, you can store a label within the gallery and reference it for comparison.

Use an alternative to create a new column with a different label

This method is an alternative to using the disambiguation operator or a label inside gallery.

While creating a local copy of your data source, you can use the `AddColumns()` formula to create a new column with a different label (`NewId`) for the Id column in your source

collection. When using ForAll with Patch, you compare the NewId column against the Id column in your source data.

Bulk create records

You may want to create new records in bulk if, for example, you want to upload many images all at once.

Here's an example using Example of a checklist.

On the Create Checklist Items screen, each time you select **Add**, the information is stored in the `NewChecklistItems` collection. When you select **Submit**, `ForAll()` with `Patch()` are used to update the source collection.

Power Fx

```
ForAll( NewChecklistItems,
    Patch( ChecklistItemsSource,
        Defaults( ChecklistItemsSource ),{
            Id: Id,
            Category: Category,
            Description: Description,
            Status:Status
        }
    )
)
```

For each item in the `NewChecklistItems`, a new record is created (indicated by `Defaults(ChecklistItemsSource)`) in the `ChecklistItemsSource` collection. The `Id` is set to the `Id` from the `NewChecklistItems` collection. Similarly, `Category`, `Description`, and `Status` values are set.

See also

- Formula reference for Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Power Apps Ideas

Article • 03/20/2025

Power Fx is the open-source programming language for low code. It enables millions of people with Excel-like skills to add advanced logic to their apps. However, sometimes it's not easy to write a formula, even for the most experienced Power Fx users, as it can sometimes take a lot of time searching for, learning about, and debugging complex formulas.

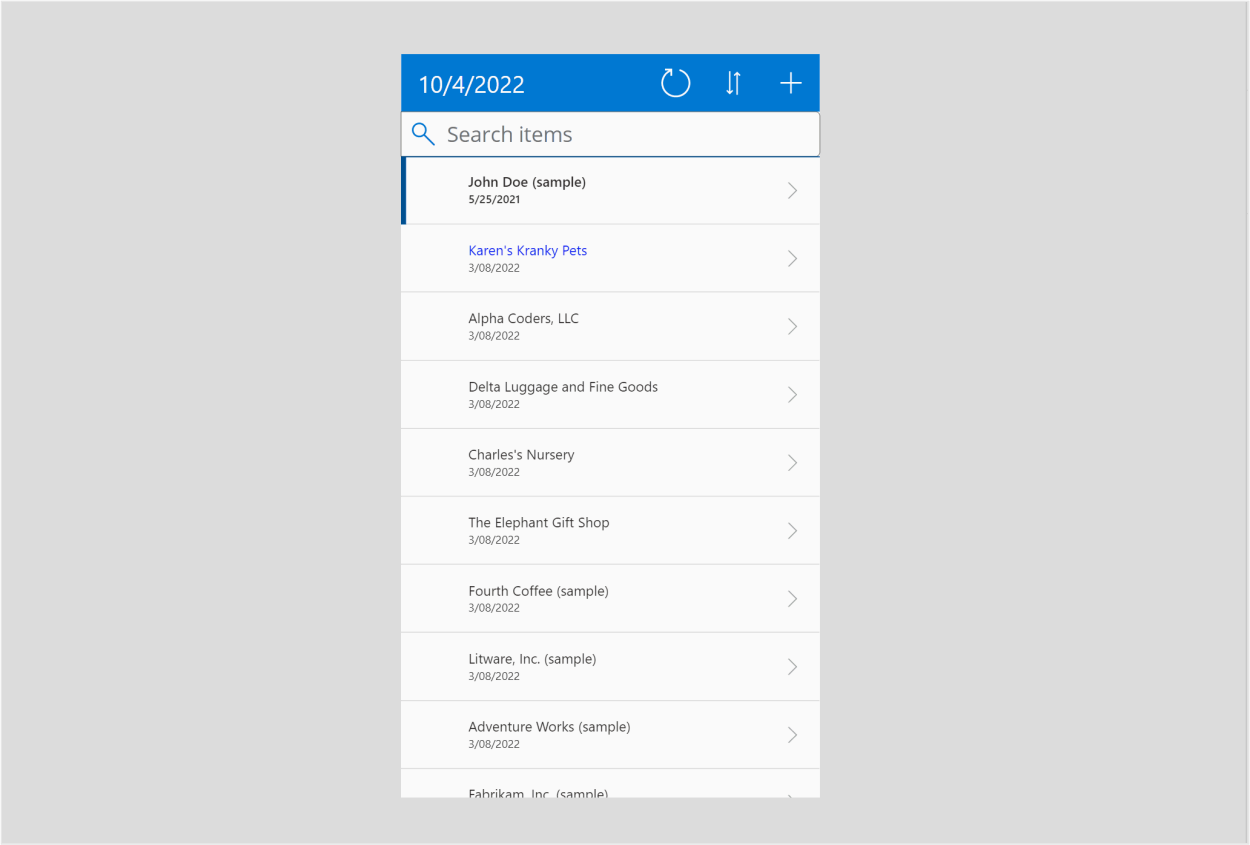
Power Apps Ideas is created to help everyone from the new makers to the seasoned IT pros to ease and speed up the formula authoring experience by using the power of AI models.

If you’re struggling with Power Fx and don’t want to spend time on it, try Power Apps Ideas.

Use Power Apps Ideas in your app

When you're building a canvas app, Power Apps Ideas will automatically suggest ideas for supported controls such as **Gallery**, **Data table**, **Text box**, **Drop down**, **Combo box**, and **Text label**.

When you select the name field in a gallery, you'll see an idea for a formula that you may want to use. However, if you select the plus icon, then you won’t see an idea because we currently don’t have ideas for icons.



There are two methods to benefit from Power Apps Ideas in your app.

Transform natural language to Power Fx formulas

You can enter your requirements in plain natural language (currently limited to English (en-us)), and Power Apps will transform your requirement to a Power Fx formula.

For details, go to Transform natural language to Power Fx formulas.

Dismiss an idea

If you don't want to see Ideas, select the close button (**X**) and then confirm that you don't want to see Ideas anymore.

If you want to see Ideas again, right-click on the control to see suggested Ideas.

Next steps

- Transform natural language to Power Fx formulas

See also

Formula reference

Feedback

Was this page helpful? Yes No

Provide product feedback

Transform natural language to Power Fx formulas

Article • 01/18/2023

Power Apps Ideas can generate ideas for formulas using the following controls on different properties:

- **Gallery**
- **Data table**
- **Text box**
- **Drop down**
- **Combo box**
- **Text label**

Power Apps Ideas also supports Microsoft Dataverse, Microsoft Lists, SharePoint library, and Excel as connectors. So, if you're using one of the supported controls and want to write a formula for Gallery or Dropdown items, or for Label's text content, color, or visibility, then check out Power Apps Ideas.

For example, instead of figuring out how to write the following formula ...

```
Power Fx
```

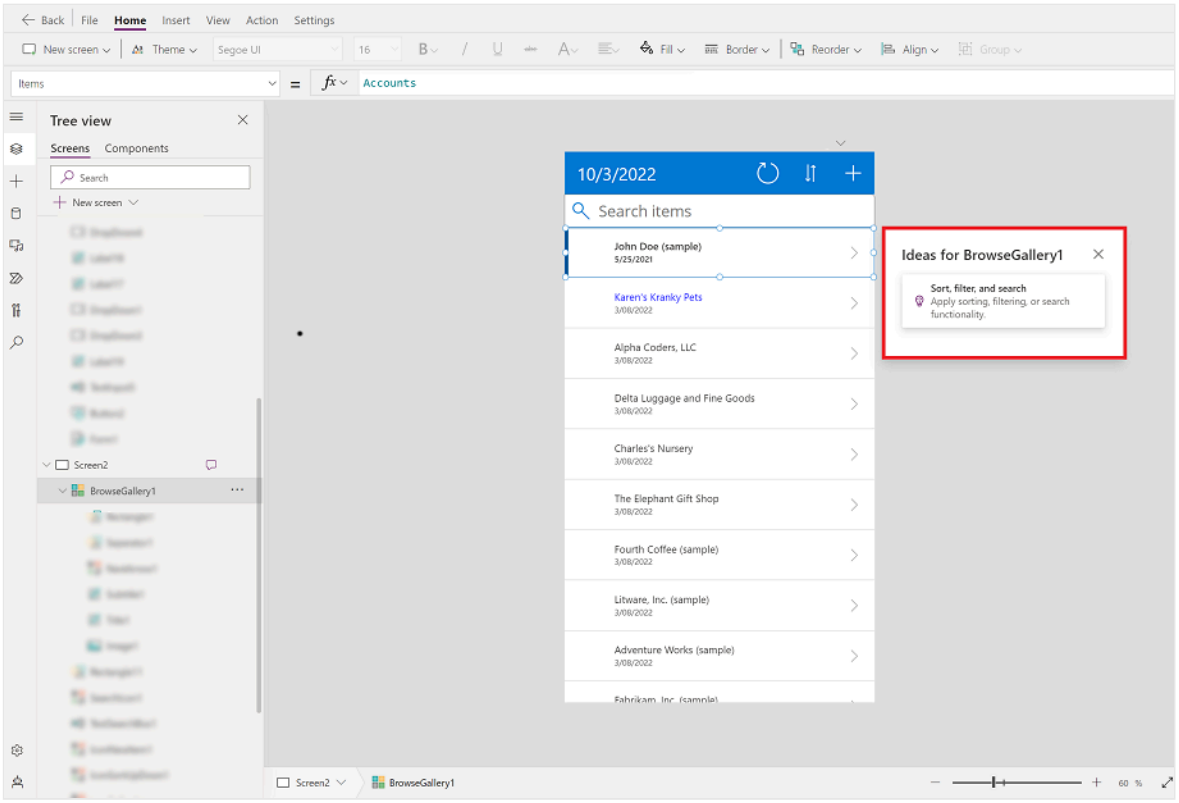
```
Filter('Table1', StartsWith('fullname', "Yvonne"))
```

... You can type `'name starts with Yvonne` instead.

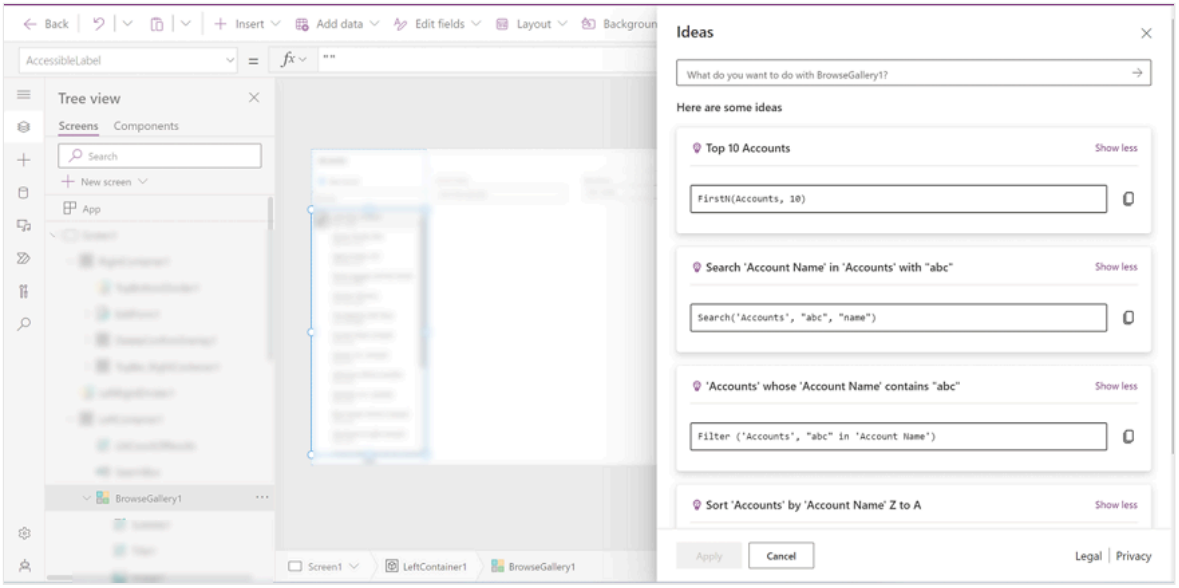
To begin, let's look at a simple app to see how you can use Power Apps Ideas to help write formulas.

You can follow the steps mentioned in Create a canvas app from Microsoft Dataverse to generate that app using the **Accounts** table from Dataverse.

1. Select **BrowseGallery1,** which is bound to the **Accounts** table.
2. Select **Sort, filter, and search** on the ideas box.



3. Select one of the default suggestions to check how it works or start entering your own query in natural language such as English (en-us).



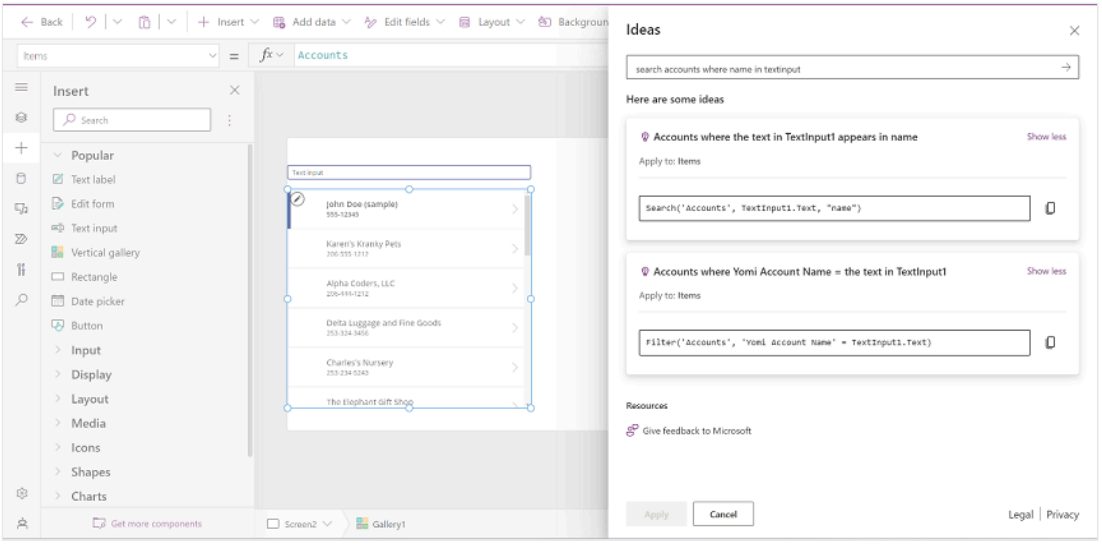
4. When typing you can use the default query, modify it, or describe it in your own way.

5. You'll see one or more formula suggestions produced. Let's use the example **search accounts where name in textinput**. You'll see two suggestions in the results. Each of those formula suggestions includes three parts:

    - **Accounts** where the text in **TextInput1** appears in **name**. This is trying to explain what the suggested formula is doing so that it's easier for users who

aren't familiar with Power Fx to understand the result. The names of the components used in the app are shown in bold.

- **Apply to: Items**: This suggests which property the formula will be applied to. As Power Apps controls have many properties, Ideas can identify which property a formula should be applied to.
- The formula suggestion in this scenario is `Search('Accounts', TextInput1.Text, "name")`.



6. Select your preferred suggestion and then select **Apply**. In this case, let's select the first suggestion since it's the best fit. The formula gets updated automatically and runs so you can check the result in your app. You can then decide whether to keep the selected formula or modify it.

More examples

Consider the following examples for inspiration while working with Power Apps Ideas.

Expand table

Scenario	Description	Ideas example
Sort your table	Sort by a single column	sort accounts by Account Name
		sort accounts by Account Name A-Z
		sort accounts by Account Name Z-A
	Sort accounts by Account	

Scenario	Description	Ideas example
	Name and Created On	
		sort accounts by Account Name A-Z and Created On latest on top
	Sort existing tables	sort Gallery3 by Account Name Z to A
Filter or search table by some conditions	A number condition	accounts with length of Account Name bigger than 10
	A date condition	accounts created a week ago, accounts modified last 3 days
	A text condition	accounts with name contains Karen
	A few conditions	accounts whose name contains Karen and created 7 days ago
	A choice value	inactive accounts
	Search table by user input	search accounts with name in textinput
	Find a single record	first record in accounts where name contains Karen
	Find records on the top or the bottom of the table	top 10 accounts
		last 10 accounts
Conditional formatting (apply to color and visible)	A number condition	accounts with length of Account Name bigger than 10
	A date condition	red if created 7 days ago
	A text condition	blue if name contains Karen
Text generation	Numbers	count of gallery sum of slider1 and slider2
	String	label4 and label5 Trim textinput1First 3 characters of TextInput1

Best practices

To get the best results, follow these tips when writing the plain natural language query in the Ideas pane.

- Leverage default queries to see what kind of scenarios are supported.
- Be precise in your query. Ideas can recognize controls such as tables and columns. However, if you don't get a suggestion, try using the full name of the assets you're referencing. For example, use **TextInput1** when you have multiple text input controls in the app. This way the system knows which control you're referring to.

Limitations

> **Note**
>
> We're improving the model so it can accomplish more complex tasks and support more functions, controls, and properties. If you have a wish list, submit your ideas through Power Apps Ideas - Power Platform Community.

Power Apps Ideas currently has the following limitations:

- Regions and language: Currently available in environments created inside the United States, with English (en-us) set as the browser's default language.
- Recognizes control value of **TextInput**, **Dropdown**, **DatePicker**, **Slider**, **Toggle**, **Checkbox**, and **Radio** as input.

Expand table

Controls	Properties
Gallery, **Data table**, **Drop down**, **Combo box**	Items
Text Box, **Label**	Text, Color, Visible

- Data sources: Currently Ideas supports Dataverse tables, Microsoft Lists, and Excel sheets.
- Functions: See Supported Power Fx functions. We'll continue to add support for more functions.
- Data types: Supports Text, Whole Number, Date and Time, Date Only, Decimal Number, Choice, Choices, Yes/No Lookup. Not supported: File and Image.

- This feature's model understands some commonly used expressions in natural language. For example, it can translate from `latest on top`, `big to small` to a descending order and `oldest on top`, `small to big` to ascending order. And it can understand most of the context so even if you didn't specify the table names, it will make the best prediction based on the current table that is bound to the control.

Supported Power Fx functions

Functions supported by the Power Apps Ideas feature:

Abs

Average

Concat

Countif

Countrows

Date

DateAdd

DateDiff

DateTimeValue

DateValue

Day

Distinct

EndsWith

Filter

FirstN

Hour

If

IsBlank

IsEmpty

LastN

Lower

Max

Min

Minute

Month

Not

Now

Power

Proper

Rand

Round

RoundDown

RoundUp

Search

Second

Split

Sort

SortByColumns

StartsWith

Sum

Time

TimeValue

Today

Trim

TrimEnds

Upper

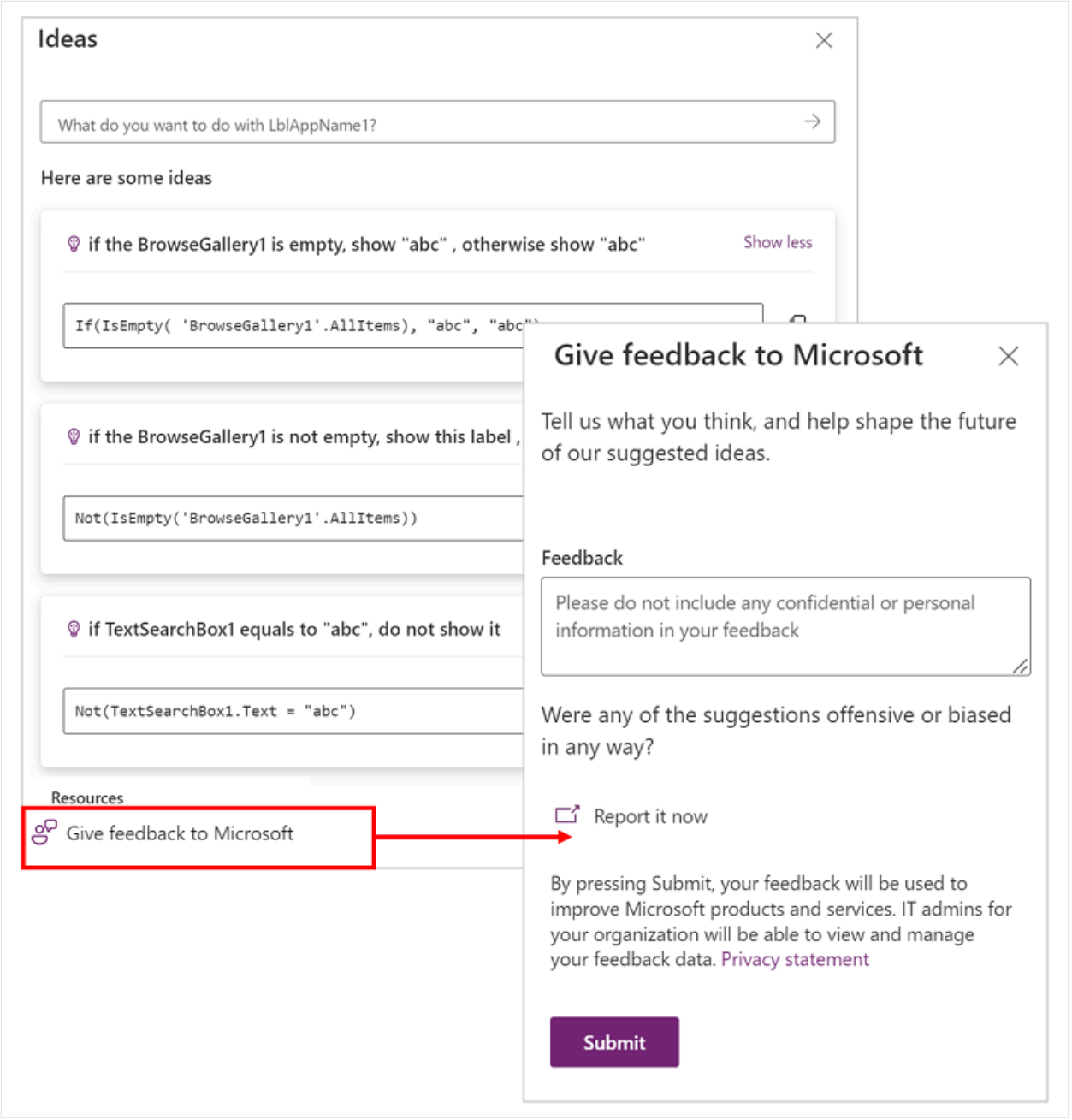
Weekday

Year

Give feedback to Microsoft

If you have feedback for this feature—for example, you find the formula suggestions inaccurate or you have suggestions or ideas to share with us—you can find the feedback card in the Ideas panel. By selecting Send feedback, you can enter your thoughts. (Don't include confidential or personal information in your feedback.)

Microsoft is committed to developing and deploying AI technologies in a responsible manner. If you find any inappropriate or absurd results generated by Power Apps Ideas, look in the same feedback card for a Report it now link below the feedback box that can help us keep our AI model behaving in a responsible manner. For a timely response, select **Threat type** as "URL" and **Incident type** as "Responsible AI" as shown in the following screenshot.



See also

- Formula reference
- Power Apps Ideas overview

Feedback

Was this page helpful? Yes No

Provide product feedback

Formula reference - canvas apps

This article explains all the functions you can use in canvas apps.

Formulas in canvas apps combine several elements. Here are the main types:

- **Functions** take parameters, perform an operation, and return a value. For example, **Sqrt(25)** returns **5**. Functions are similar to Microsoft Excel functions. Some functions can have side effects, like **SubmitForm**, which you use only in a behavior formula such as **Button.OnSelect**.
- **Signals** return information about the environment. For example, **Location** returns the device's current GPS coordinates. Signals don't take parameters or have side effects.
- **Enumerations** return a predefined constant value. For example, **Color** is an enumeration with predefined values like **Color.Red** and **Color.Blue**. Common enumerations are included here, and function-specific enumerations are described with the function.
- **Named operators**, like **ThisItem** and **Self**, let you access information from within a container.

Other elements include:

- Operators and identifiers
- Controls and their properties
- Data types

A

Abs – Returns the absolute value of a number.

Acceleration – Reads the acceleration sensor in your device.

Acos – Returns the arccosine of a number, in radians.

Acot – Returns the arccotangent of a number, in radians.

AddColumns – Returns a table with additional columns.

And – Boolean logic AND. Returns **true** if all arguments are **true**. You can also use the **&&** operator.

AIClassify – Classifies text into one or more categories.

AIExtract – Extracts specified entities like registration numbers, phone numbers, or people's names.

AIReply – Drafts a reply to the message that you provide.

AISentiment – Detects the sentiment of the text that you provide.

AISummarize – Summarizes the text that you provide.

AITranslate – Translates text from one language to another.

App – Provides information about the currently running app and lets you control the app's behavior.

Asin – Returns the arcsine of a number, in radians.

Assert – Evaluates to true or false in a test.

As – Names the current record in gallery, form, and record scope functions like **ForAll**, **With**, and **Sum**.

AsType – Treats a record reference as a specific table type.

Atan – Returns the arctangent of a number, in radians.

Atan2 – Returns the arctangent based on an (*x*,*y*) coordinate, in radians.

Average – Calculates the average of a table expression or a set of arguments.

B

Back – Shows the previous screen.

Blank – Returns a *blank* value you can use to insert a NULL value in a data source.

Boolean – Converts a text string, number, or dynamic value to a Boolean value.

C

Calendar – Retrieves information about the calendar for the current locale.

Char – Translates a character code into a string.

Choices – Returns a table of the possible values for a lookup column.

Clear – Deletes all data from a collection.

ClearCollect – Deletes all data from a collection and then adds a set of records.

ClearData – Clears a collection or all collections from an app host such as a local device.

Clock – Retrieves information about the clock for the current locale.

Coalesce – Replaces *blank* values while leaving non-*blank* values unchanged.

Collect – Creates a collection or adds data to a data source.

Color – Sets a property to a built-in color value.

ColorFade – Fades a color value.

ColorValue – Translates a CSS color name or a hex code to a color value.

Column – Retrieves column names and values from a Dynamic value.

ColumnNames – Retrieves column names and values from a Dynamic value.

Compass – Returns your compass heading.

Concat – Concatenates strings in a data source.

Concatenate – Concatenates strings.

Concurrent – Evaluates multiple formulas concurrently with one another.

Confirm - Display a confirmation dialog box.

Connection – Returns information about your network connection.

Copy – Copies text to the clipboard on the device where the app is running.

Count – Counts table records that contain numbers.

Cos – Returns the cosine of an angle specified in radians.

Cot – Returns the cotangent of an angle specified in radians.

CountA – Counts table records that aren't empty.

CountIf – Counts table records that satisfy a condition.

CountRows – Counts table records.

D

DataSourceInfo – Provides information about a data source.

Date – Returns a date/time value, based on **Year**, **Month**, and **Day** values.

DateAdd – Adds days, months, quarters, or years to a date/time value.

DateDiff – Subtracts two date values, and shows the result in days, months, quarters, or years.

DateTime – Returns a date/time value, based on both date and time components.

DateTimeValue – Converts a date and time string to a date/time value.

DateValue – Converts a date-only string to a date/time value.

Day – Retrieves the day portion of a date/time value.

Dec2Hex – Convert a number to a hexadecimal text string.

Defaults – Returns the default values for a data source.

Degrees – Converts radians to degrees.

Disable – Disables a signal, such as **Location** for reading the GPS.

Distinct – Summarizes records of a table, removing duplicates.

Download – Downloads a file from the web to the local device.

DropColumns – Returns a table with one or more columns removed.

E

EDate – Adds or subtracts months to a date, without changing the day of the month.

EditForm – Resets a form control for editing of an item.

Enable – Enables a signal, such as **Location** for reading the GPS.

EncodeHTML – Encodes characters that need to be escaped to be used in an HTML context.

EncodeUrl – Encodes special characters using URL encoding.

EndsWith – Checks whether a text string ends with another text string.

EOMonth – Adds or subtracts months to a date, returning the last day of that month.

Error – Create a custom error or pass through an error.

Errors – Provides error information for previous changes to a data source.

exactin – Checks if a text string is contained within another text string or table, case dependent. Also used to check if a record is in a table.

Exit – Exits the currently running app and optionally signs out the current user.

Exp – Returns *e* raised to a power.

F

Filter – Returns a filtered table based on one or more criteria.

Find – Checks whether one string appears within another and returns the location.

First – Returns the first record of a table.

FirstN – Returns the first set of records (N records) of a table.

Float – Converts a string to a Float value.

ForAll – Calculates values and performs actions for all records of a table.

G

GroupBy – Returns a table with records grouped together.

GUID – Converts a GUID string to a GUID value or creates a new GUID value.

H

HashTags – Extracts the hashtags (#strings) from a string.

Hex2Dec – Convert a hexadecimal text string to a number.

Host – Provides information about the current host running the app.

Hour – Returns the hour portion of a date/time value.

I

If – Returns one value if a condition is true and another value if not.

IfError – Detects errors and provides an alternative value or takes action.

in – Checks if a text string is contained within another text string or table, case independent. Also used to check if a record is in a table.

Index – Returns a record from a table based on ordered position.

Int – Rounds down to the nearest integer.

IsBlank – Checks for a blank value.

IsBlankOrError – Checks for a blank value or error.

IsEmpty – Checks for an empty table.

IsError – Checks for an error.

IsMatch – Checks a string against a pattern. Regular expressions can be used.

IsNumeric – Checks for a numeric value.

ISOWeekNum – Returns the ISO week number of a date/time value.

IsToday – Checks whether a date/time value is sometime today in the user's time zone.

IsType – Checks whether a record reference refers to a specific table type.

J

JSON – Generates a JSON text string for a table, a record, or a value.

L

Language – Returns the language tag of the current user.

Last – Returns the last record of a table.

LastN – Returns the last set of records (N records) of a table.

Launch – Launches a webpage or a canvas app.

Left – Returns the left-most portion of a string.

Len – Returns the length of a string.

Ln – Returns the natural log.

LoadData – Loads a collection from an app host such as a local device.

Location – Returns your location as a map coordinate by using the Global Positioning System (GPS) and other information.

Log – Returns the logarithm in any base of a number.

LookUp – Looks up a single record in a table based on one or more criteria.

Lower – Converts letters in a string of text to all lowercase.

M

Match – Extracts a substring based on a pattern. Regular expressions can be used.

MatchAll – Extracts multiple substrings based on a pattern. Regular expressions can be used.

Max – Maximum value of a table expression or a set of arguments.

Mid – Returns the middle portion of a string.

Min – Minimum value of a table expression or a set of arguments.

Minute – Retrieves the minute portion of a date/time value.

Mod – Returns the remainder after a dividend is divided by a divisor.

Month – Retrieves the month portion of a date/time value.

N

Navigate – Changes which screen is displayed.

NewForm – Resets a form control for creation of an item.

Not – Boolean logic NOT. Returns **true** if its argument is **false**, and returns **false** if its argument is **true**. You can also use the ! operator.

Notify – Displays a banner message to the user.

Now – Returns the current date/time value in the user's time zone.

O

Or – Boolean logic OR. Returns **true** if any of its arguments are **true**. You can also use the || operator.

P

Param – Access parameters passed to a canvas app when launched.

Parent – Provides access to a container control's properties.

ParseJSON – Converts JSON document represented as text to a Dynamic value value.

Patch – Modifies or creates a record in a data source, or merges records outside of a data source.

PDF – Export contents from the current screen to an object for use in multiple scenarios.

Pi – Returns the number π.

PlainText – Removes HTML and XML tags from a string.

Power – Returns a number raised to a power. You can also use the ^ operator.

Print - Opens the current screen in the default browser print dialog.

Proper – Converts the first letter of each word in a string to uppercase, and converts the rest to lowercase.

R

Radians – Converts degrees to radians.

Rand – Returns a pseudo-random number between 0 and 1.

RandBetween – Returns a pseudo-random number between two numbers.

ReadNFC – Reads a Near Field Communication (NFC) tag.

RecordInfo – Provides information about a record of a data source.

RecordOf - Extracts the type of a record from a table.

Refresh – Refreshes the records of a data source.

Relate – Relates records of two tables through a one-to-many or many-to-many relationship.

Remove – Removes one or more specific records from a data source.

RemoveIf – Removes records from a data source based on a condition.

RenameColumns – Renames columns of a table.

Replace – Replaces part of a string with another string, by starting position of the string.

RequestHide – Hides a SharePoint form.

Reset – Resets an input control to its default value, discarding any user changes.

ResetForm – Resets a form control for editing of an existing item.

Revert – Reloads and clears errors for the records of a data source.

RGBA – Returns a color value for a set of red, green, blue, and alpha components.

Right – Returns the right-most portion of a string.

Round – Rounds to the closest number.

RoundDown – Rounds down to the largest previous number.

RoundUp – Rounds up to the smallest next number.

S

SaveData – Saves a collection to an app host such as a local device.

Search – Finds records in a table that contain a string in one of their columns.

Second – Retrieves the second portion of a date/time value.

Select – Simulates a select action on a control, causing the **OnSelect** formula to be evaluated.

Self – Provides access to the properties of the current control.

Sequence – Generate a table of sequential numbers, useful when iterating with **ForAll**.

Set – Sets the value of a global variable.

SetFocus – Moves input focus to a specific control.

SetProperty – Simulates interactions with input controls.

ShowColumns – Returns a table with only selected columns.

ShowHostInfo – Displays information to the user in the app.

Shuffle – Randomly reorders the records of a table.

Sin – Returns the sine of an angle specified in radians.

Sort – Returns a sorted table based on a formula.

SortByColumns – Returns a sorted table based on one or more columns.

Split – Splits a text string into a table of substrings.

Sqrt – Returns the square root of a number.

StartsWith – Checks if a text string begins with another text string.

StdevP – Returns the standard deviation of its arguments.

Substitute – Replaces part of a string with another string, by matching strings.

SubmitForm – Saves the item in a form control to the data source.

Sum – Calculates the sum of a table expression or a set of arguments.

Switch – Matches with a set of values and then evaluates a corresponding formula.

T

Table – Creates a temporary table.

Tan – Returns the tangent of an angle specified in radians.

Text – Converts any value and formats a number or date/time value to a string of text.

ThisItem – Returns the record for the current item in a gallery or form control.

ThisRecord – Returns the record for the current item in a record scope function, such as **ForAll**, **With**, and **Sum**.

Time – Returns a date/time value, based on **Hour**, **Minute**, and **Second** values.

TimeValue – Converts a time-only string to a date/time value.

TimeZoneOffset – Returns the difference between UTC and the user's local time in minutes.

Today – Returns the current date-only value.

Trace – Provide additional information in your test results.

Trim – Removes extra spaces from the ends and interior of a string of text.

TrimEnds – Removes extra spaces from the ends of a string of text only.

Trunc – Truncates the number to only the integer portion by removing any decimal portion.

Type - Defines a user defined type.

U

Ungroup – Removes a grouping.

UniChar – Translates a Unicode code into a string.

Unrelate – Unrelates records of two tables from a one-to-many or many-to-many relationship.

Update – Replaces a record in a data source.

UpdateContext – Sets the value of one or more context variables of the current screen.

UpdateIf – Modifies a set of records in a data source based on a condition.

Upper – Converts letters in a string of text to all uppercase.

User – Returns information about the current user.

V

Validate – Checks whether the value of a single column or a complete record is valid for a data source.

Value – Converts a string to a number.

VarP – Returns the variance of its arguments.

ViewForm – Resets a form control for viewing of an existing item.

W

Weekday – Retrieves the weekday portion of a date/time value.

WeekNum – Returns the week number of a date/time value.

With – Calculates values and performs actions for a single record, including inline records of named values.

Y

Year – Retrieves the year portion of a date/time value.

Last updated on 06/19/2025

Abs, Exp, Ln, Power, Log, and Sqrt functions

Article • 04/20/2025

Expand table

Functions	Applies to
Abs **Exp** **Ln** **Power** **Sqrt**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Log	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Calculates absolute values, logarithms, square roots, and the results of raising *e* or any number to specified powers.

Description

The **Abs** function returns the non-negative value of its argument. If a number is negative, **Abs** returns the positive equivalent.

The **Exp** function returns *e* raised to the power of its argument. The transcendental number *e* begins 2.7182818...

The **Ln** function returns the natural logarithm (base *e*) of its argument.

The **Power** function returns a number raised to a power. It's equivalent to using the ^ operator.

The **Log** function returns the logarithm of its first argument in the base specified by its second argument (or 10 if not specified).

The **Sqrt** function returns the number that, when multiplied by itself, equals its argument.

If you pass a single number, the return value is a single result based on the function called. If you pass a single-column table that contains numbers, the return value is a single-column table of results in a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined valued, the result is *blank*. Which can happen with square roots and logarithms of negative numbers.

Syntax

Abs(*Number*)
Exp(*Number*)
Ln(*Number*)
Sqrt(*Number*)

- *Number* - Required. Number to operate on.

Power(*Base*, *Exponent*)

- *Base* - Required. Base number to raise.
- *Exponent* - Required. The exponent to which the base number is raised.

Log(*Number*, *Base*)

- *Number* - Required. Number to calculate the logarithm.
- *Base* - Optional. The base of the logarithm to calculate. By default, 10 (when not specified).

Abs(*SingleColumnTable*)
Exp(*SingleColumnTable*)
Ln(*SingleColumnTable*)
Sqrt(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Examples

Single number

Expand table

Formula	Description	Result
Abs(-55)	Returns the number without the negative sign.	55
Exp(2)	Returns *e* raised to the power of 2, or *e* * *e*.	7.389056...
Ln(100)	Returns the natural logarithm (base *e*) of the number 100.	4.605170...
Log(100)	Returns the logarithm in base 10 of the number 100.	2
Log(64, 2)	Returns the logarithm in base 2 of the number 64.	6

Formula	Description	Result
Power(5, 3)	Returns 5 raised to the power of 3, or 5 * 5 * 5.	125
Sqrt(9)	Returns the number that, when multiplied by itself, results in 9.	3

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains this data:

Expand table

Value
9
-4
2

Expand table

Formula	Description	Result
Abs(ValueTable)	Returns the absolute value of each number in the table.	A single-column table with a `Value` column containing the following values: 9, 4, 2
Exp(ValueTable)	Returns *e* raised to the power of each number in the table.	A single-column table with a `Value` column containing the following values: 8103.083927..., 0.018315..., 7.389056...
Ln(ValueTable)	Returns the natural logarithm of each number in the table.	A single-column table with a `Value` column containing the following values: 2.197224..., Blank(), 0.693147...
Sqrt(ValueTable)	Returns the square root of each number in the table	A single-column table with a `Value` column containing the following values: 3, Blank(), 1.414213...

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a **Label** control, and set its **Text** property to this formula:
   Sqrt(Value(Source.Text))

3. Type a number into **Source**, and confirm that the **Label** control shows the square root of the number that you typed.

Acceleration, App, Compass, Connection, and Location signals in Power Apps

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Returns information about the app's environment, such as where the user is located in the world and which screen is displayed.

Description and syntax

Signals are values that can change at any time, independent of how the user may be interacting with the app. Formulas that are based on signals automatically recalculate as these values change.

Signals typically return a record of information. You can use and store this information as a record, or you can extract individual properties by using the . operator.

> **Note**
>
> The **Acceleration** and **Compass** functions return accurate values in a native player such as on iOS or Android, but those functions return zero values as you create or modify an app in the browser.

Acceleration

The **Acceleration** signal returns the device's acceleration in three dimensions relative to the device's screen. Acceleration is measured in *g* units of 9.81 m/second^2 or 32.2 ft/second^2 (the acceleration that the Earth imparts to objects at its surface due to gravity).

Expand table

Property	Description
Acceleration.X	Right and left. Right is a positive number.
Acceleration.Y	Forward and back. Forward is a positive number.

Property	Description
Acceleration.Z	Up and down. Up is a positive number.

App

Among other properties, the **App** object includes a signal that indicates which screen is showing.

Expand table

Property	Description
App.ActiveScreen	Screen that's showing. Returns a screen object, which you can use to reference properties of the screen or compare to another screen to determine which screen is showing. You can use the **Back** or **Navigate** function to change the screen that's showing.

More information: **App** object documentation.

Compass

The **Compass** signal returns the compass heading of the top of the screen. The heading is based on magnetic north.

Expand table

Property	Description
Compass.Heading	Heading in degrees. Returns a number 0 to 360, and 0 is north.

Connection

The **Connection** signal returns the information about the network connection. When on a metered connection, you may want to limit how much data you send or receive over the network. Or, if your app is enabled for offline, you may want to display different information depending on the current sync state.

Expand table

Property	Description
Connection.Connected	Returns a Boolean **true** or **false** value that indicates whether the device is connected to a network.
Connection.Metered	Returns a Boolean **true** or **false** value that indicates whether the connection is metered.
Connection.Sync	Returns an enum named **ConnectionSync** which holds the current synchronization state for an app that is enabled for offline.

Below are the enums values available for Connection.Sync.

Expand table

Enum value	Description
Connected	Your device is connected to the network and your app is ready to work offline
ConnectedWithWarning	Your device is connected with a warning.
ConnectedPendingUpsync	Some data on your device must be synchronized with the service.
ConnectedError	There was an error on the last synchronization attempt.
ConnectedRefresh	Your app is currently synchronizing data with the service.
NotConnected	Your device isn't connected to the service.
NotConnectedWithWarning	The last synchronization warning.
NotConnectedPendingUpsync	There are updates on your device waiting to be synchronized.
NotConnectedSyncError	The last synchronization error.

Location

The **Location** signal returns the location of the device based on the Global Positioning System (GPS) and other device information, such as cell-tower communications and IP address.

When a user accesses the location information for the first time, the device may prompt that user to allow access to this information.

As the location changes, dependencies on the location will continuously recalculate, which will consume power from the device's battery. To conserve battery life, you can

use the **Enable** and **Disable** functions to turn location updates on and off. Location is automatically turned off if the displayed screen doesn't depend on location information.

Expand table

Property	Description
Location.Altitude	Returns a number that indicates the altitude, measured in meters, above sea level.
Location.Latitude	Returns a number, from –90 to 90, that indicates the latitude, as measured in degrees from the equator. A positive number indicates a location that's north of the equator.
Location.Longitude	Returns a number, from –180 to 180, that indicates the longitude, as measured in degrees from Greenwich, England. A positive number indicates a location that's east of Greenwich.

Examples

In a baseball field, a pitcher throws a phone from the pitcher's mound to a catcher at home plate. The phone is lying flat with respect to the ground, the top of the screen is pointed at the catcher, and the pitcher adds no spin. At this location, the phone has cellular network service that's metered but no WiFi. The **PlayBall** screen is displayed. The app is enabled for offline and is connected to the service.

Expand table

Formula	Description	Result
Location.Latitude	Returns the latitude of the current location. The field is located at map coordinates 47.591 N, 122.333 W.	47.591 The latitude will change continuously as the ball moves between the pitcher and the catcher.
Location.Longitude	Returns the longitude of the current location.	122.333 The longitude will change continuously as the ball moves between the pitcher and the catcher.
Location	Returns the latitude and longitude of the current location, as a record.	{ Latitude: 47.591, Longitude: 122.333 }

Formula	Description	Result
Compass.Heading	Returns the compass heading of the top of the screen. At this field, home plate is roughly southwest from the pitcher's mound.	230.25
Acceleration.X	Returns the acceleration of the device side to side. The pitcher is throwing the phone straight ahead with respect to the screen's top, so the device isn't accelerating side to side.	0
Acceleration.Y	Returns the acceleration of the device front to back. The pitcher initially gives the device a large acceleration when throwing the device, going from 0 to 90 miles per hour (132 feet per second) in half a second. After the device is in the air, ignoring air friction, the device doesn't accelerate further. The device decelerates when the catcher catches it, bringing it to a stop.	8.2, while the pitcher throws the device. 0, while the device is in the air. -8.2, as the catcher catches the device.
Acceleration.Z	Returns the acceleration of the device top to bottom. While in the air, the device experiences the effects of gravity.	0, before the pitcher throws the device. 1, while the device is in the air. 0, after the catcher catches the device.
Acceleration	Returns the acceleration as a record.	{ X: 0, Y: 264, Z: 0 } as the pitcher throws the device.
Connection.Connected	Returns a Boolean value that indicates whether the device is connected to a network	**true**
Connection.Metered	Returns a Boolean value that indicates whether the connection is metered	**true**
Connection.Sync	Returns an enum value that indicates current sync state.	**ConnectionSync.Connected**
App.ActiveScreen = PlayBall	Returns a Boolean value that indicates whether **PlayBall** is	**true**

Formula	Description	Result
	displayed.	
App.ActiveScreen.Fill	Returns the background color for the displayed screen.	**Color.Green**

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

AddColumns, DropColumns, RenameColumns, and ShowColumns functions

06/21/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Shapes a table or record by adding, dropping, renaming, and selecting its columns.

The ForAll function can also be used to shape a table, by returning a table of new records created from existing columns.

Overview

These functions shape a table or record by adjusting its columns:

- Reduce a table or record that contains multiple columns down to a single column for use with single-column functions, such as **Lower** or **Sqrt**.
- Add a calculated column to a table or record (for example, a **Total Price** column that shows the results of multiplying **Quantity** by **Unit Price**).
- Rename a column to something more meaningful, for display to users or for use in formulas.

A table is a value in Power Apps, just like a string or a number. You can specify a table as an argument in a formula, and functions can return a table as a result.

> **Note**
>
> The functions that this topic describes don't modify the original table. Instead, they take that table as an argument and return a new table with a transform applied. For more information, see working with tables.

You can't modify the columns of a data source by using these functions. You must modify the data at its source. You can add columns to a collection with the **Collect** function. For more information, see working with data sources.

Description

The **AddColumns** function adds a column to a table or record, and a formula defines the values in that column. Existing columns remain unmodified.

The formula is evaluated for the provided record, or for each record of the provided table.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

The **DropColumns** function excludes columns from a table or record. All other columns remain unmodified. **DropColumns** excludes columns, and **ShowColumns** includes columns.

Use the **RenameColumns** function to rename one or more columns of a table or record by providing at least one argument pair that specifies the name of a column that the table or record contains (the old name, which you want to replace) and the name of a column that the table or record doesn't contain (the new name, which you want to use). The old name must already exist in the table or record, and the new name must not exist. Each column name may appear only once in the argument list as either an old column name or a new column name. To rename a column to an existing column name, first drop the existing column with **DropColumns**, or rename the existing column out of the way by nesting one **RenameColumns** function within another.

The **ShowColumns** function includes columns of a table or record and drops all other columns. You can use **ShowColumns** to create a single-column table or record from a multi-column table or record. **ShowColumns** includes columns, and **DropColumns** excludes columns.

For all these functions, the result is a new table or record with the transform applied. The original table or record isn't modified. You can't modify an existing table or record with a formula. SharePoint, Microsoft Dataverse, SQL Server, and other data sources provide tools for modifying the columns of lists and tables, which are often referred to as the schema. The functions in this topic only transform an input table or record, without modifying the original, into an output table or record for further use.

The arguments to these functions support delegation. For example, a **Filter** function used as an argument to pull in related records searches through all listings, even if the '**[dbo].[AllListings]**' data source contains a million rows:

Power Fx

```
AddColumns( RealEstateAgents,
   Listings,
```

```
    Filter(  '[dbo].[AllListings]', ListingAgentName = AgentName )
)
```

However, the output of these functions is subject to the non-delegation record limit. In this example, only 500 records are returned even if the **RealEstateAgents** data source has 501 or more records.

If you use **AddColumns** in this manner, **Filter** must make separate calls to the data source for each of those first records in **RealEstateAgents**, which causes lots of network chatter. If **[dbo](.[AllListings]** is small enough and doesn't change often. You can call the **Collect** function in OnStart to cache the data source in your app when it starts. As an alternative, you could restructure your app so that you pull in the related records only when the user asks for them.

> **Note**
>
> In Power Apps prior to version 3.24042, column names were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

Syntax

AddColumns(*TableOrRecord*, *ColumnName1*, *Formula1* [, *ColumnName2*, *Formula2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to add.
- *Formula(s)* - Required. Formulas to evaluate for each record. The result is added as the value of the corresponding new column. You can reference other columns of the table or record in this formula.

DropColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to drop.

RenameColumns(*TableOrRecord*, *OldColumnName1*, *NewColumnName1* [, *OldColumnName2*, *NewColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.

- *OldColumnName(s)* - Required. Names of the columns to rename from the original table or record. This element appears first in the argument pair (or first in each argument pair if the formula includes more than one pair).
- *NewColumnName(s)* - Required. Replacement names. This element appears last in the argument pair (or last in each argument pair if the formula includes more than one pair).

ShowColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to include.

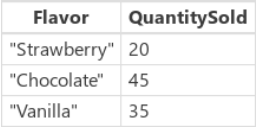
Examples

The examples in this section use the **IceCreamSales** data source, which contains the data in this table:

Flavor	UnitPrice	QuantitySold
"Strawberry"	1.99	20
"Chocolate"	2.99	45
"Vanilla"	1.50	35

None of these examples modify the **IceCreamSales** data source. Each function transforms the value of the data source as a table and returns that value as the result.

Expand table

Formula	Description	Result
AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold)	Adds a **Revenue** column to the result. For each record, **UnitPrice * QuantitySold** is evaluated, and the result is placed in the new column.	
DropColumns(IceCreamSales, UnitPrice)	Excludes the **UnitPrice** column from the result. Use this function to exclude columns, and use **ShowColumns** to include them.	
ShowColumns(IceCreamSales, Flavor)	Includes only the **Flavor** column in the result. Use this function include columns, and use **DropColumns** to exclude them.	
RenameColumns(IceCreamSales, UnitPrice, Price)	Renames the **UnitPrice** column in the result.	

Formula	Description	Result
RenameColumns(IceCreamSales, UnitPrice, Price, QuantitySold, Number)	Renames the **UnitPrice** and **QuantitySold** columns in the result.	<table><tr><th>Flavor</th><th>Price</th><th>Number</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>20</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>45</td></tr><tr><td>"Vanilla"</td><td>1.50</td><td>35</td></tr></table>
DropColumns(RenameColumns(AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold), UnitPrice, Price), Quantity)	Performs the following table transforms in order, starting from the inside of the formula: 1. Adds a **Revenue** column based on the per record calculation of **UnitPrice * Quantity**. 2. Renames **UnitPrice** to **Price**. 3. Excludes the **Quantity** column. Order is important, for example we can't calculate with **UnitPrice** after it's renamed.	<table><tr><th>Flavor</th><th>Price</th><th>Revenue</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>39.80</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>134.55</td></tr><tr><td>"Vanilla"</td><td>1.49</td><td>52.15</td></tr></table>

Step by step

Let's try some of the examples from earlier in this article.

1. Create a collection by adding a **Button** control and setting its **OnSelect** property to this formula:

   Power Fx

   ```
   ClearCollect( IceCreamSales,
      Table(
          { Flavor: "Strawberry", UnitPrice: 1.99, QuantitySold: 20 },
          { Flavor: "Chocolate", UnitPrice: 2.99, QuantitySold: 45 },
          { Flavor: "Vanilla", UnitPrice: 1.50, QuantitySold: 35 }
      )
   )
   ```

2. Run the formula by selecting the button while holding down the Alt key.

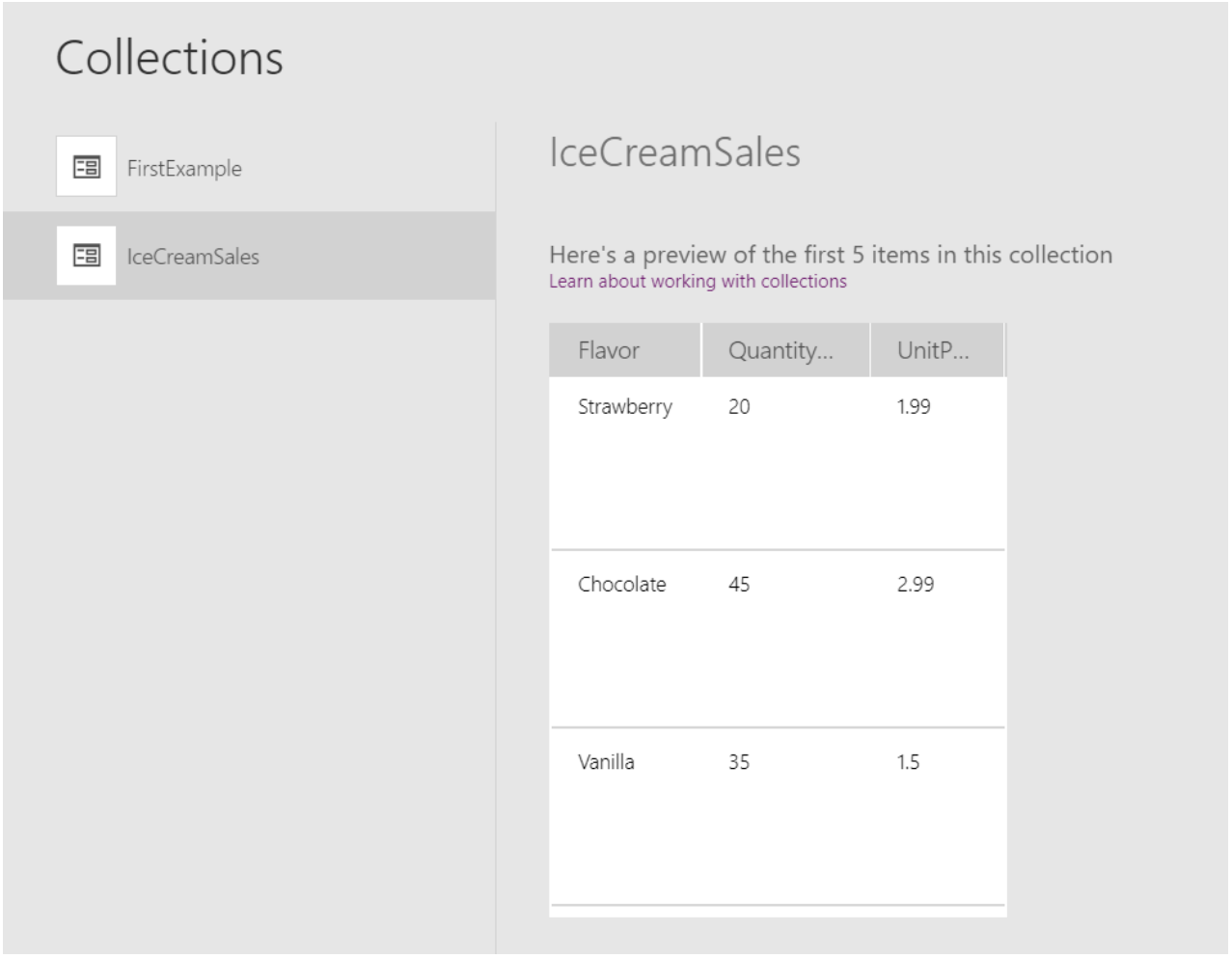
3. Add a second **Button** control, set its **OnSelect** property to this formula, and then run it:

   Power Fx

   ```
   ClearCollect( FirstExample,
      AddColumns( IceCreamSales, Revenue, UnitPrice * QuantitySold )
   )
   ```

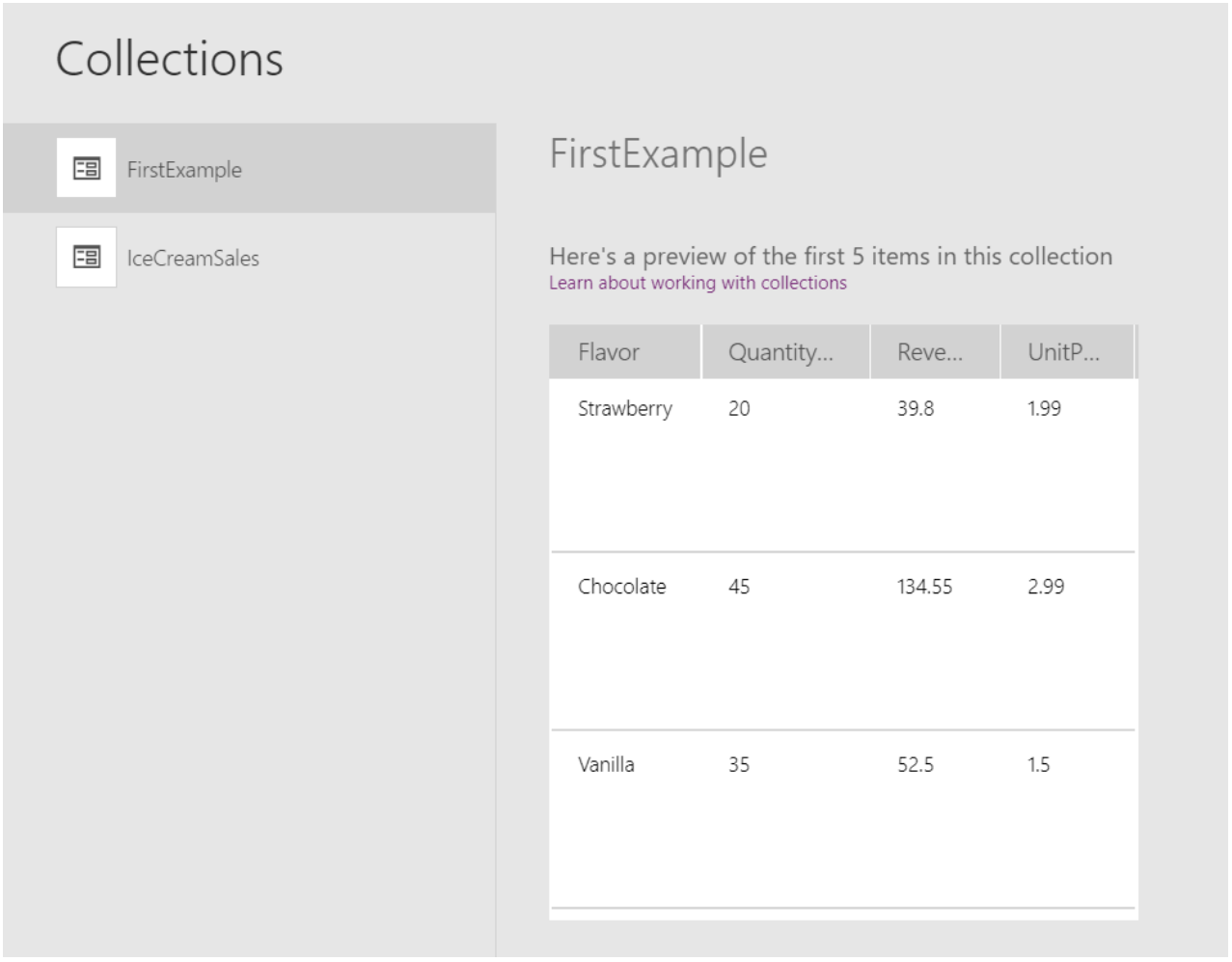
4. On the **File** menu, select **Collections**, and then select **IceCreamSales** to show that collection.

As this graphic shows, the second formula didn't modify this collection. The **AddColumns** function used **IceCreamSales** as a read-only argument; the function didn't modify the table to which that argument refers.



5. Select **FirstExample**.

   As this graphic shows, the second formula returned a new table with the added column. The **ClearCollect** function captured the new table in the **FirstExample** collection, adding something to the original table as it flowed through the function without modifying the source:



Map columns in a component

See Map columns.

And, Or, and Not functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Boolean logic functions, commonly used to manipulate the results of comparisons and tests.

Description

The **And** function returns **true** if all of its arguments are **true**.

The **Or** function returns **true** if any of its arguments are **true**.

The **Not** function returns **true** if its argument is **false**; it returns **false** if its argument is **true**.

These functions work the same way as they do in Excel. You can also use operators to perform these same operations, using either Visual Basic or JavaScript syntax:

Expand table

Function notation	Visual Basic operator notation	JavaScript operator notation
And(x, y)	x And y	x && y
Or(x, y)	x Or y	x || y
Not(x)	Not x	! x

These functions work with logical values. You can't pass them a number or a string directly; instead, you must make a comparison or a test. For example, this logical formula **x** > **1** evaluates to the Boolean value **true** if **x** is greater than **1**. If **x** is less than **1**, the formula evaluates to **false**.

Syntax

And(*LogicalFormula1*, *LogicalFormula2* [, *LogicalFormula3*, ...])
Or(*LogicalFormula1*, *LogicalFormula2* [, *LogicalFormula3*, ...])
Not(*LogicalFormula*)

- *LogicalFormula(s)* - Required. Logical formulas to evaluate and operate on.

Examples

The examples in this section use these global variables:

- **a** = *false*
- **b** = *true*
- **x** = 10
- **y** = 100
- **s** = "Hello World"

To create these global variables in an app, insert a **Button** control, and set its **OnSelect** property to this formula:

Power Apps

```
Set( a, false ); Set( b, true ); Set( x, 10 ); Set( y, 100 ); Set( s, "Hello World" )
```

Select the button (by clicking it while you hold down the Alt key), and then set the **Text** property of a **Label** control to a formula in the first column of the next table.

Expand table

Formula	Description	Result
And(a, b)	Tests the values of **a** and **b**. One of the arguments is *false*, so the function returns *false*.	*false*
a And b	Same as the previous example, using Visual Basic notation.	*false*
a && b	Same as the previous example, using JavaScript notation.	*false*
Or(a, b)	Tests the values of **a** and **b**. One of the arguments is *true*, so the function returns *true*.	*true*
a Or b	Same as the previous example, using Visual Basic notation.	*true*
a || b	Same as the previous example, using JavaScript notation.	*true*
Not(a)	Tests the value of **a**. The argument is *false*, so the function returns the opposite result.	*true*
Not a	Same as the previous example, using Visual Basic notation.	*true*
! a	Same as the previous example, using JavaScript notation.	*true*
Len(s) < 20 And Not IsBlank(s)	Tests whether the length of **s** is less than 20 and whether it isn't a **blank** value. The length is less than 20, and the value	*true*

Formula	Description	Result
	isn't blank. Therefore, the result is *true*.	
Or(Len(s) < 10, x < 100, y < 100)	Tests whether the length of **s** is less than 10, whether **x** is less than 100, and whether **y** is less than 100. The first and third arguments are false, but the second one is true. Therefore, the function returns *true*.	*true*
Not IsBlank(s)	Tests whether **s** is *blank*, which returns *false*. **Not** returns the opposite of this result, which is *true*.	*true*

Feedback

Was this page helpful? Yes No

Provide product feedback

App object in Power Apps

06/19/2025

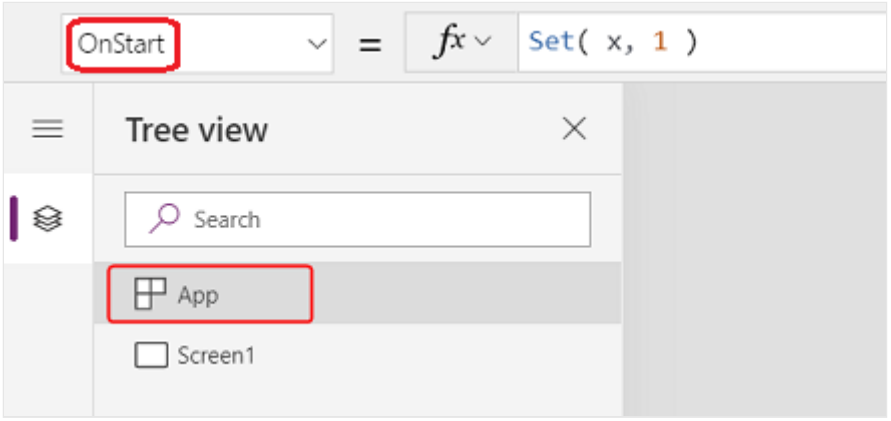
Applies to: Canvas apps

Get information about the currently running app and control the app's behavior.

Description

Like a control, the **App** object has properties that identify which screen is showing and prompt you to save changes so they aren't lost. Every app has an **App** object.

Write formulas for some properties of the **App** object. At the top of the **Tree view** pane, select the **App** object as you would any other control or screen. To view or edit one of the object's properties, select it in the drop-down list to the left of the formula bar.



ActiveScreen property

The **ActiveScreen** property identifies the screen that's currently displayed.

This property returns a screen object. Use it to reference properties of the current screen, like the name with the formula **App.ActiveScreen.Name**. You can also compare this property to another screen object, like with the comparison formula **App.ActiveScreen** = **Screen2** to check if **Screen2** is the current screen.

Use the Back or Navigate function to switch the screen that's displayed.

BackEnabled property

The **BackEnabled** property changes how the app responds to the device back gesture (swipe or use the hardware back button on Android devices, or swipe from the left on iOS devices) when

running in Power Apps mobile. When enabled, the device back gesture goes back to the screen that was most recently shown, which is similar to the **Back** formula. When disabled, the device back gesture takes the user to the app list.

ConfirmExit properties

Nobody wants to lose unsaved changes. Use the **ConfirmExit** and **ConfirmExitMessage** properties to warn the user before closing your app.

> ⓘ **Note**
>
> - **ConfirmExit** doesn't work in apps embedded in, for example, Power BI and SharePoint.
> - **ConfirmExit** isn't supported in custom pages.
> - Now, these properties can reference controls only on the first screen if the **Delayed load** preview feature is enabled (which it is by default for new apps). If you reference other screens, Power Apps Studio doesn't show an error, but the published app doesn't open in Power Apps Mobile or a browser. We're working to lift this limitation. In the meantime, turn off **Delayed load** in **Settings** > **Upcoming features** (under **Preview**).

ConfirmExit

ConfirmExit is a Boolean property that, when *true*, opens a confirmation dialog box before the app closes. By default, this property is *false*, and no dialog box appears.

When the user might have unsaved changes in the app, use this property to show a confirmation dialog box before exiting the app. Use a formula that checks variables and control properties (for example, the **Unsaved** property of the Edit form control).

The confirmation dialog box appears in any situation where data can be lost, such as:

- Run the Exit function.
- If the app runs in a browser:
  - Close the browser or the browser tab where the app runs.
  - Select the browser's back button.
  - Run the Launch function with a *LaunchTarget* of **Self**.
- If the app runs in Power Apps Mobile (iOS or Android):
  - Swipe to switch to a different app in Power Apps Mobile.
  - Select the back button on an Android device.

- Run the **Launch** function to launch another canvas app.

The exact look of the confirmation dialog box can vary across devices and versions of Power Apps.

The confirmation dialog box doesn't show in Power Apps Studio.

ConfirmExitMessage

By default, the confirmation dialog box shows a generic message, such as **"You may have unsaved changes."** in the user's language.

Use **ConfirmExitMessage** to provide a custom message in the confirmation dialog box. If this property is *blank*, the default value is used. Custom messages are truncated as needed to fit within the confirmation dialog box, so keep the message to a few lines at most.

In a browser, the confirmation dialog box can show a generic message from the browser.

> **Note**
>
> The App object has two more properties, `OnMessage` and `BackEnabled`, that are experimental. These properties are removed from the app object eventually. Don't use these properties in your production environment.

Example

1. Set the **App** object's **ConfirmExit** property to this expression:

   ```
   Power Fx

   AccountForm.Unsaved Or ContactForm.Unsaved
   ```

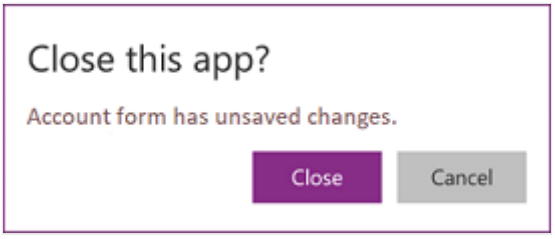
   The dialog box shows if the user changes data in either form and then tries to close the app without saving those changes.e user changes data in either form and then tries to close the app without saving those changes.

   > [!div clas1. Set the **App** object's **ConfirmExitMessage** property to this formula:

   ```
   Power Fx

   If( AccountsForm.Unsaved,
       "Accounts form has unsaved changes.",
   ```

```
    "Contacts form has unsaved changes."
)
```

The dialog box shows if the user changes data in the Account form and then tries to close the app without saving those changes.



Set up connection string for Application Insights

To export system-generated application logs to Application Insights, set up the **Connection string** for your canvas app.

1. Open your app for editing in Power Apps Studio.
2. Select the **App** object in the left navigation tree view.
3. Enter the **Connection string** in the properties pane.

If data isn't sent to Application Insights, contact your Power Platform admin and check if **App Insights** is disabled at the tenant level.

Formulas property

Use named formulas, in the **Formulas** property, to define a formula that can be reused throughout your app.

In Power Apps, formulas determine the value of control properties. For example, to set the background color consistently across an app, you might set the **Fill** property for each to a common formula:

Power Fx

```
Label1.Fill: ColorValue( Param( "BackgroundColor" ) )
Label2.Fill: ColorValue( Param( "BackgroundColor" ) )
Label3.Fill: ColorValue( Param( "BackgroundColor" ) )
```

With so many places where this formula may appear, it becomes tedious and error prone to update them all if a change is needed. Instead, you can create a global variable in **OnStart** to set the color once, and then reuse the value throughout the app:

Power Fx

```
App.OnStart: Set( BGColor, ColorValue( Param( "BackgroundColor" ) ) )
Label1.Fill: BGColor
Label2.Fill: BGColor
Label3.Fill: BGColor
```

While this method is better, it also depends on **OnStart** running before the value for **BGColor** is established. **BGColor** might also be manipulated in some corner of the app that the maker is unaware of, a change made by someone else, and that can be hard to track down.

Named formulas provide an alternative. Just as we commonly write *control-property* = *expression*, we can instead write *name* = *expression* and then reuse *name* throughout our app to replace *expression*. The definitions of these formulas are done in the **Formulas** property:

Power Fx

```
App.Formulas: BGColor = ColorValue( Param( "BackgroundColor" ) );
Label1.Fill: BGColor
Label2.Fill: BGColor
Label3.Fill: BGColor
```

The advantages of using named formulas include:

- **The formula's value is always available.** There's no timing dependency, no **OnStart** that must run first before the value is set, no time in which the formula's value is incorrect. Named formulas can refer to each other in any order, so long as they don't create a circular reference. They can be calculated in parallel.
- **The formula's value is always up to date.** The formula can perform a calculation that is dependent on control properties or database records, and as they change, the formula's value automatically updates. You don't need to manually update the value as you do with a variable. And formulas only recalculate when needed.
- **The formula's definition is immutable.** The definition in **Formulas** is the single source of truth and the value can't be changed somewhere else in the app. With variables, it's possible that some code unexpectedly changes a value, but this difficult to debug situation isn't possible with named formulas.
- **The formula's calculation can be deferred.** Because its value is immutable, it can always be calculated when needed, which means it need not be calculated until it's needed. Formula values that aren't used until **screen2** of an app is displayed need not be calculated until **screen2** is visible. Deferring this work can improve app load time. Named formulas are declarative and provide opportunities for the system to optimize how and when they're computed.

- **Named formulas is an Excel concept.** Power Fx uses Excel concepts where possible since so many people know Excel well. Named formulas are the equivalent of named cells and named formulas in Excel, managed with the Name Manager. They recalculate automatically like cells of a spreadsheet and control properties do.

Named formulas are defined, one after another in the **Formulas** property, each ending with a semi-colon. The type of the formula is inferred from the types of the elements within the formula and how they're used together. For example, these named formulas retrieve useful information about the current user from Dataverse:

Power Fx

```
UserEmail = User().Email;
UserInfo = LookUp( Users, 'Primary Email' = User().Email );
UserTitle = UserInfo.Title;
UserPhone = Switch( UserInfo.'Preferred Phone',
                    'Preferred Phone (Users)'.'Mobile Phone', UserInfo.'Mobile
Phone',
                    UserInfo.'Main Phone' );
```

If the formula for **UserTitle** needs to be updated, it can be done easily in this one location. If **UserPhone** isn't needed in the app, then these calls to the **Users** table in Dataverse aren't made. There's no penalty for including a formula definition that isn't used.

Some limitations of named formulas:

- They can't use behavior functions or otherwise cause side effects within the app.
- They can't create a circular reference. Having **a = b;** and **b = a;** in the same app isn't allowed.

User defined functions

> **Important**
>
> - User defined functions are a preview feature.
> - Preview features aren't meant for production use and may not be complete. These features are available before an official release so that you can get early access and provide feedback. More information: Understand experimental, preview, and retired features in canvas apps
> - The behavior that this article describes is available only when the **User-defined functions** preview feature in Settings > Upcoming features > Experimental is turned on (it's off by default).

- Your feedback is valuable to us. Let us know what you think in the Power Apps experimental features community forum.

Power Fx includes a long list of built-in functions, such as **If**, **Text**, and **Set**. User defined functions enable you to write your own functions that take parameters and return a value, just as the built-in functions do. You can think of user defined functions as an extension to named formulas that adds parameters and supports behavior formulas.

For example, you might define a named formula that returns fiction books from a library:

Power Apps

```
Library = [ { Title: "The Hobbit", Author: "J. R. R. Tolkien", Genre: "Fiction" },
            { Title: "Oxford English Dictionary", Author: "Oxford University",
Genre: "Reference" } ];

LibraryFiction = Filter( Library, Genre = "Fiction" );
```

Without parameters, we would need to define separate named formulas for each genre. But instead, let's parameterize our named formula:

Power Apps

```
LibraryType := Type( [ { Title: Text, Author: Text, Genre: Text } ] );

LibraryGenre( SelectedGenre: Text ): LibraryType = Filter( Library, Genre =
SelectedGenre );
```

Now we can call `LibraryGenre( "Fiction" )`, `LibraryGenre( "Reference" )`, or filter on other genres with a single user defined function.

The syntax is:

FunctionName([*ParameterName1*: *ParameterType1* [, *ParameterName2*: *ParameterType2* ...]]) : *ReturnType* = *Formula*;

- *FunctionName* – Required. The name of the user defined function.
- *ParameterName(s)* – Optional. The name of a function parameter.
- *ParameterType(s)* – Optional. The name of a type, either a built-in data type name, a data source name, or a type defined with the **Type** function.
- *ReturnType* – Required. The type of the return value from the function.
- *Formula* – Required. The formula that calculates the value of the function based on the parameters.

Each parameter and the output from the user define function must be typed. In this example, `SelectedGenre: Text` defines the first parameter to our function to be of type **Text** and `SelectedGenre` is the name of the parameter that is used in the body for the **Filter** operation. See **Data types** for the supported type names. The **Type** function is used to create an aggregate type for our library, so that we can return a table of books from our function.

We defined `LibraryType` as a plural table of records type. If we want to pass a single book to a function, we can extract the type of the record for this table with the **RecordOf** function:

Power Apps

```
BookType := Type( RecordOf( LibraryType ) );

IsGenre( Book: BookType, SelectedGenre: Text ): Boolean = (Book.Genre =
SelectedGenre);
```

Record matching for function parameters is tighter than it is in other parts of Power Fx. The fields of a record value must be a proper subset of the type definition and can't include additional fields. For example, `IsGenre( { Title: "My Book", Published: 2001 }, "Fiction" )` will result in an error.

Note, recursion isn't yet supported by user defined functions.

Behavior user defined functions

Named formulas and most user defined functions don't support behavior functions with side effects, such as **Set** or **Notify**. In general, it's best to avoid updating state if you can, instead relying on functional programming patterns and allowing Power Fx to recalculate formulas as needed automatically. But, there are cases where it's unavoidable. To include behavior logic in a user defined function, wrap the body in curly braces:

Power Apps

```
Spend( Amount: Number ) : Void = {
    If( Amount > Savings,
        Error( $"{Amount} is more than available savings" ),
        Set( Savings, Savings - Amount );
        Set( Spent, Spent + Amount)
    );
}
```

Now we can call `Spend( 12 )` to check if we have 12 in our Savings, and if so, to debit it by 12 and add 12 to the Spent variable. The return type of this function is **Void** as it doesn't return a value.

The syntax of a behavior user defined function is:

FunctionName([*ParameterName1*: *ParameterType1* [, *ParameterName2*: *ParameterType2* ...]]) : *ReturnType* = { *Formula1* [; *Formula2* ...] };

- *FunctionName* – Required. The name of the user defined function.
- *ParameterName(s)* – Optional. The name of a function parameter.
- *ParameterType(s)* – Optional. The name of a type, either a built-in data type name, a data source name, or a type defined with the **Type** function.
- *ReturnType* – Required. The type of the return value from the function. Use **Void** if the function doesn't return a value.
- *Formula(s)* – Required. The formula that calculates the value of the function based on the parameters.

As with all Power Fx formulas, execution doesn't end when an error is encountered. After the **Error** function has been called, the **If** function prevents the changes to **Savings** and **Spent** from happening. The **IfError** function can also be used to prevent further execution after an error. Even though it returns **Void**, the formula can still return an error if there's a problem.

User defined types

> **Important**
>
> - User defined types is an experimental feature.
> - Experimental features aren't meant for production use and may not be complete. These features are available before an official release so that you can get early access and provide feedback. More information: Understand experimental, preview, and retired features in canvas apps
> - The behavior that this article describes is available only when the **User-defined types** experimental feature in Settings > Upcoming features > Experimental is turned on (it's off by default).
> - Your feedback is valuable to us. Let us know what you think in the Power Apps experimental features community forum.

Named formulas can be used with the **Type** function to create user defined types. Use `:=` instead of `=` to define a user defined type, for example `Book := Type( { Title: Text, Author: Text } )`. See the **Type** function for more information and examples.

OnError property

Use **OnError** to take action when an error happens anywhere in the app. It provides a global opportunity to intercept an error banner before being displayed to the end user. It can also be used to log an error with the **Trace** function or write to a database or web service.

In Canvas apps, the result of every formula evaluation is checked for an error. If an error is encountered, **OnError** is evaluated with the same **FirstError** and **AllErrors** scope variables that would have been used if the entire formula had been wrapped in an **IfError** function.

If **OnError** is empty, a default error banner shows the **FirstError.Message** of the error. Defining an **OnError** formula overrides this behavior, so the maker can handle error reporting as needed. You can request the default behavior in **OnError** by rethrowing the error with the **Error** function. Use the rethrowing approach if you want to filter out or handle some errors differently, but let others pass through.

OnError can't replace an error in calculations the way that **IfError** can. If **OnError** is invoked, the error has already happened and it has already been processed through formula calculations like **IfError**; **OnError** controls error reporting only.

OnError formulas are evaluated concurrently and it's possible that their evaluation might overlap with the processing of other errors. For example, if you set a global variable at the top of an **OnError** and read it later on in the same formula, the value might have changed. Use the **With** function to create a named value that is local to the formula.

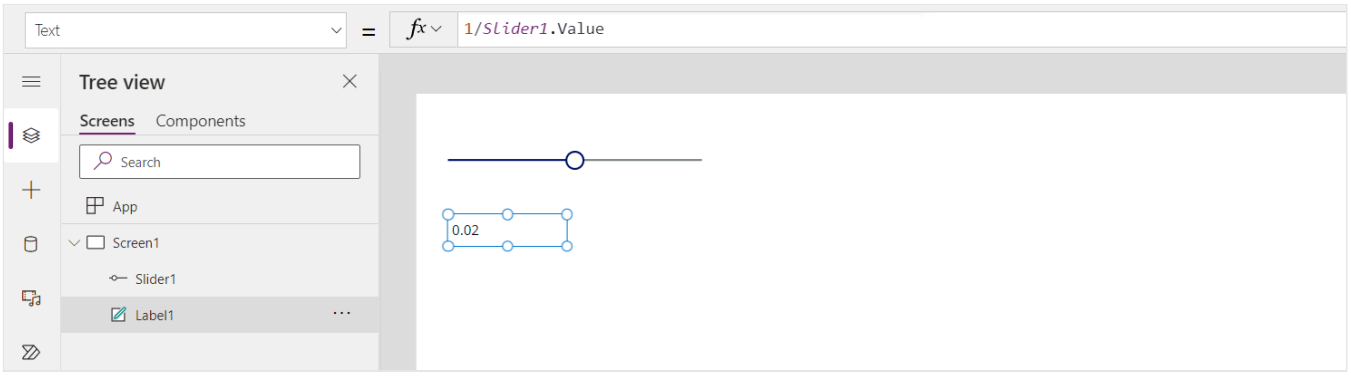
Although each error is processed individually by **OnError**, the default error banner might not appear for each error individually. To avoid having too many error banners displayed at the same time, the same error banner won't be displayed again if it has recently been shown.

Example

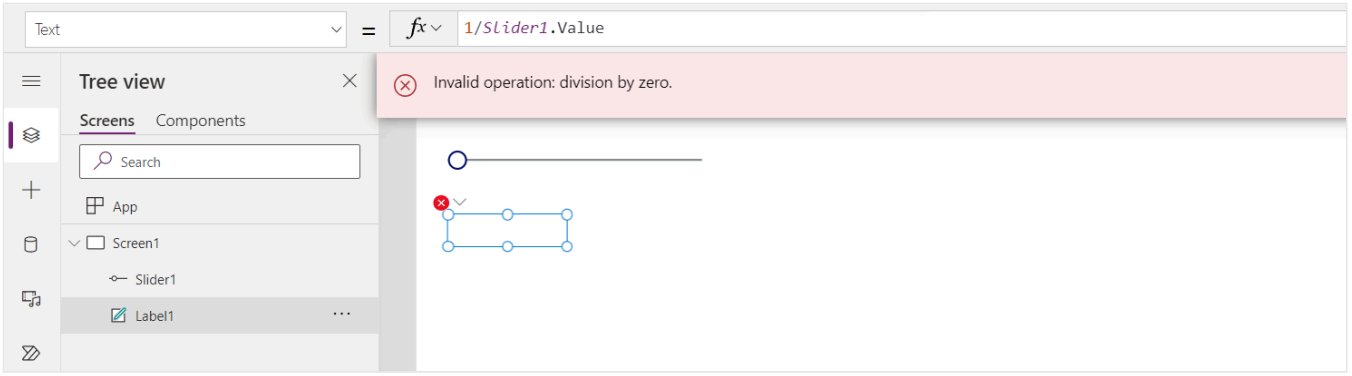
Consider a **Label** control and **Slider** control that are bound together through the formula:

Power Fx

```
Label1.Text = 1/Slider1.Value
```



The slider defaults to 50. If the slider is moved to 0, **Label1** will show no value, and an error banner is shown:



Let's look at what happened in detail:

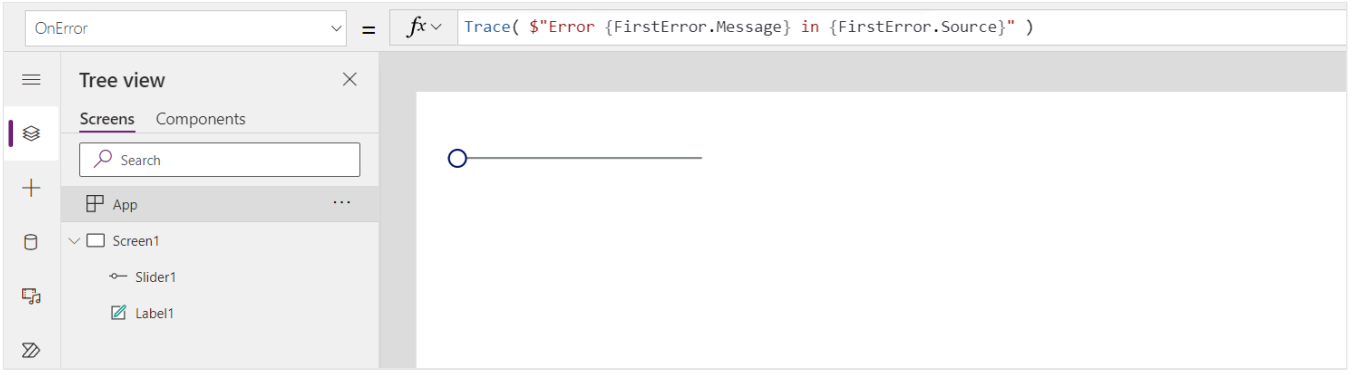
1. User moved the slide to the left and the **Slide1.Value** property changed to 0.
2. **Label1.Text** was automatically reevaluated. Division by zero occurred, generating an error.
3. There's no **IfError** in this formula. The division by zero error is returned by the formula evaluation.
4. **Label1.Text** can't show anything for this error, so it shows a *blank* state.
5. **OnError** is invoked. Since there's no handler, the standard error banner is displayed with error information.

If needed, you can also change the formula to `Label1.Text = IfError( 1/Slider1.Value, 0 )`. Using **IfError** means there's no error or error banner. You can't change the value of an error from **OnError** because the error already happened—**OnError** only controls how it's reported.

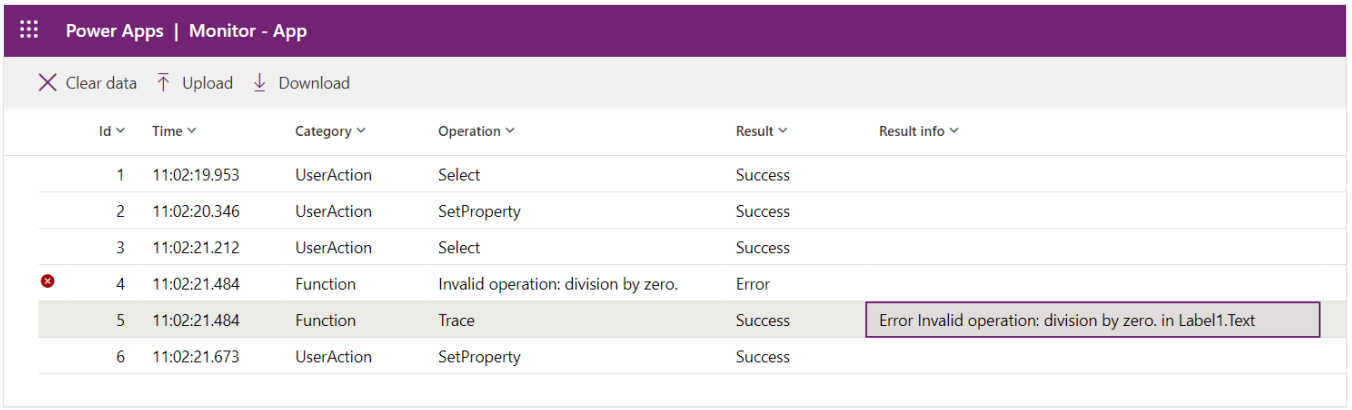
If you add an **OnError** handler, it doesn't affect steps before step 5, but it does change how the error is reported:

Power Fx

```
Trace( $"Error {FirstError.Message} in {FirstError.Source}" )
```



With this **OnError** handler, the app user doesn't see any error. But the error is added to the Monitor's trace, including the source of the error information from **FirstError**:



If you also want to show the default error banner along with the trace, rethrow the error with the **Error** function after the **Trace** call, as if the **Trace** wasn't there:

Power Fx

```
Trace( $"Error {FirstError.Message} in {FirstError.Source}" );
Error( FirstError )
```

OnStart property

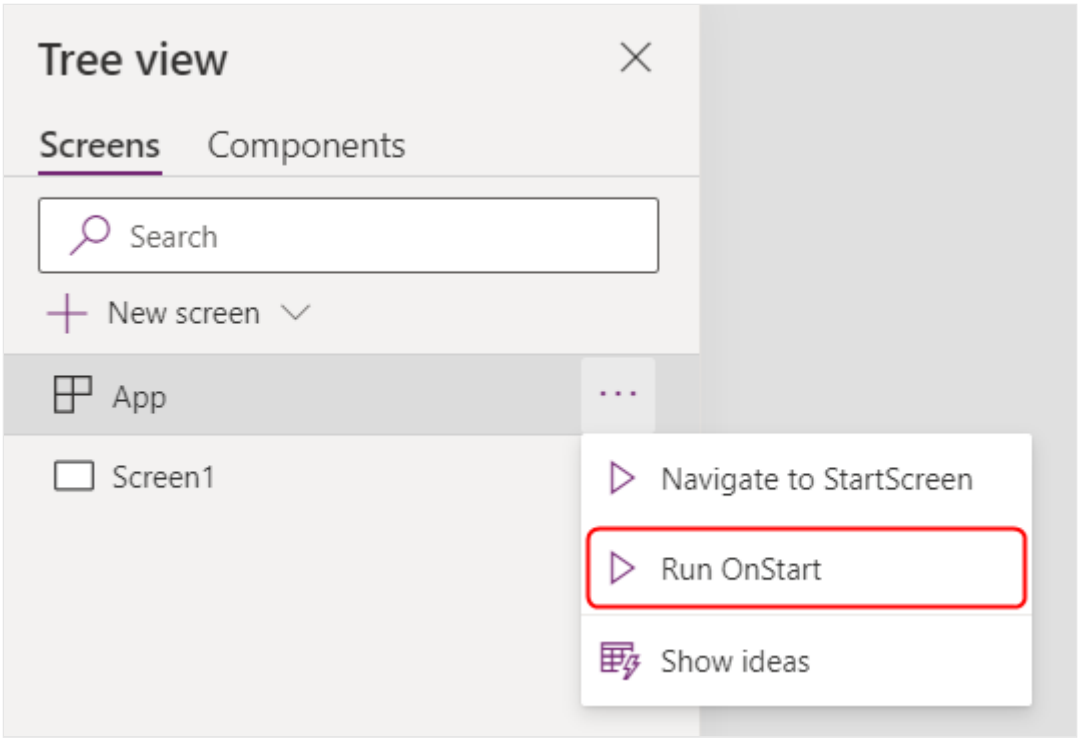
> **Note**
>
> Using the **OnStart** property can cause performance problems when loading an app. We're creating alternatives for the top two reasons to use this property—caching data and setting up global variables. There's already an alternative for defining the first screen to show with Navigate. Depending on your context, this property can be disabled by default. If you don't see it and need to use it, check the app's Advanced settings for a switch to enable it. You can also use the **OnVisible** property of a screen. By default, when the nonblocking **OnStart** rule is enabled, it lets the **OnStart** function run at the same time as other app rules. So, if variables referenced in other app rules are initialized within the **OnStart** function, they might not be fully initialized yet. Also, a screen can render and become interactive before either the **Screen.OnVisible** or **App.OnStart** functions finish running, especially if they take a long time to finish.

The **OnStart** property runs when the user starts the app. This property is often used to:

- Retrieve and cache data in collections by using the Collect function.
- Set up global variables by using the Set function.

This formula runs before the first screen appears. No screen is loaded, so you can't set context variables with the UpdateContext function. But you can pass context variables with the **Navigate** function.

After you change the **OnStart** property, test it by hovering over the **App** object in the **Tree view** pane, selecting the ellipsis (...), and then selecting **Run OnStart**. Unlike when the app loads for the first time, existing collections and variables are already set. To start with empty collections, use the ClearCollect function instead of the **Collect** function.



> **Note**
>
> - Using the Navigate function in the **OnStart** property is retired. Existing apps still work. For a limited time, you can enable it in the app settings (under **Retired**). But using **Navigate** this way can cause app load delays because it forces the system to finish running **OnStart** before showing the first screen. Use the **StartScreen** property instead to set the first screen displayed.
> - The retired switch is off for apps created before March 2021 where you added **Navigate** to **OnStart** between March 2021 and now. When you edit these apps in Power Apps Studio, you can see an error. Turn the retired switch on to clear this error.

StartScreen property

The **StartScreen** property sets which screen shows first. It's evaluated once when the app loads and returns the screen object to show. By default, this property is empty, and the first screen in the Studio Tree view shows first.

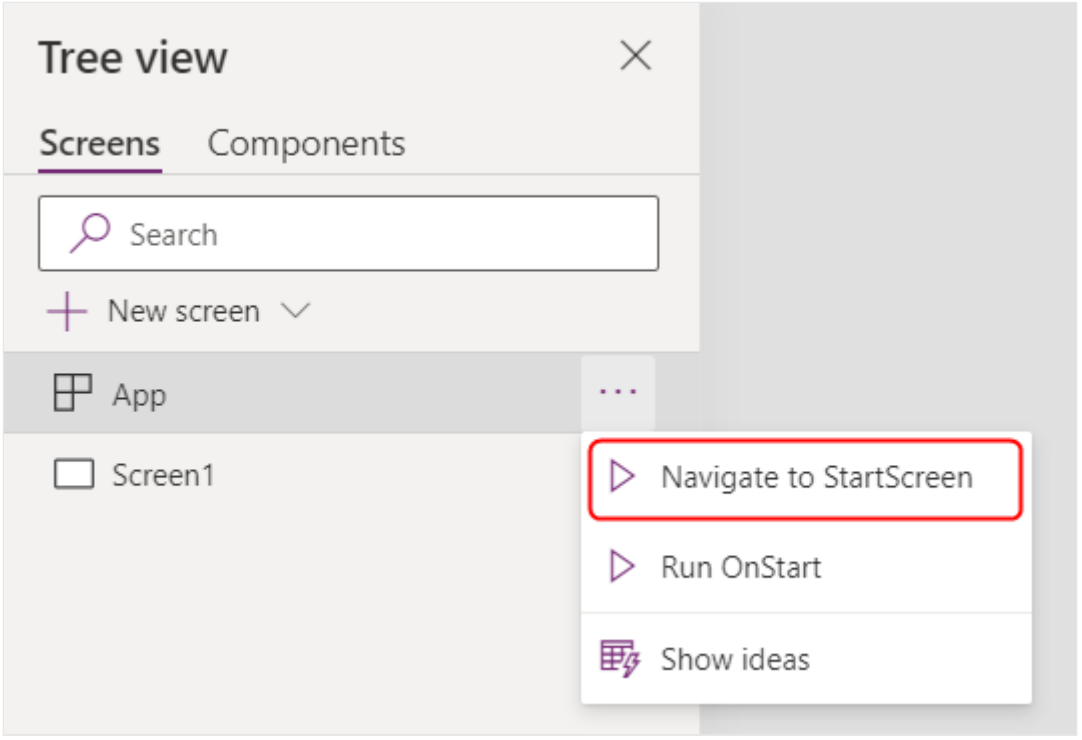
StartScreen is a data flow property that can't contain behavior functions. All data flow functions are available. Use these functions and signals to decide which screen to show first:

- Param function to read parameters used to start the app.
- User function to read information about the current user.
- LookUp, Filter, CountRows, Max, and other functions that read from a data source.
- API calls through a connector. Make sure the call returns quickly.
- Signals such as Connection, Compass, and **App**.

> **Note**
>
> Global variables and collections, including those created in **OnStart**, aren't available in **StartScreen**. Named formulas are available and are often a better alternative for formula reuse across the app.

If **StartScreen** returns an error, the first screen in the Studio Tree view shows as if **StartScreen** isn't set. Use the **IfError** function to catch any errors and redirect to an error screen.

After you change **StartScreen** in Studio, test it by hovering over the **App** object in the **Tree view** pane, selecting the ellipsis (...), and then selecting **Navigate to StartScreen**. The screen changes as if the app just loaded.



Examples

Power Fx

```
Screen9
```

`Screen9` shows first whenever the app starts.

Power Fx

```
If( Param( "admin-mode" ) = 1, HomeScreen, AdminScreen )
```

Checks if the Param "admin-mode" is set and uses it to decide if HomeScreen or AdminScreen shows first.

Power Fx

```
If( LookUp( Attendees, User = User().Email ).Staff, StaffPortal, HomeScreen )
```

Checks if an attendee is a staff member and directs them to the correct screen on startup.

Power Fx

```
IfError( If( CustomConnector.APICall() = "Forest",
             ForestScreen,
             OceanScreen
         ),
         ErrorScreen
)
```

Directs the app based on an API call to either `ForestScreen` or `OceanScreen`. If the API fails, the app uses `ErrorScreen` instead.

StudioVersion property

Use the **StudioVersion** property to show or log the version of Power Apps Studio used to publish an app. This property helps when you're debugging and checking that your app is republished with a recent version of Power Apps Studio.

StudioVersion returns text. The format of this text can change over time, so treat it as a whole and don't extract individual parts.

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

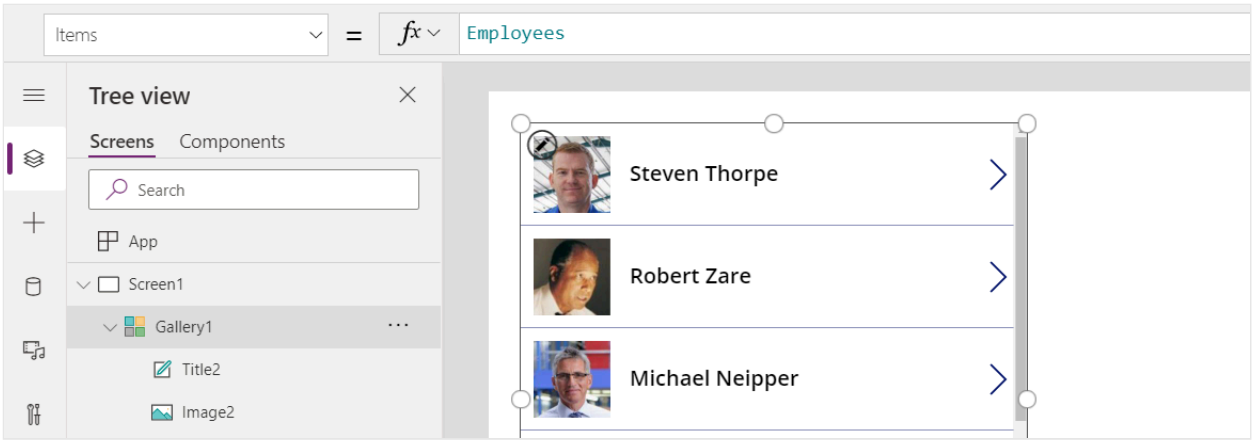
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

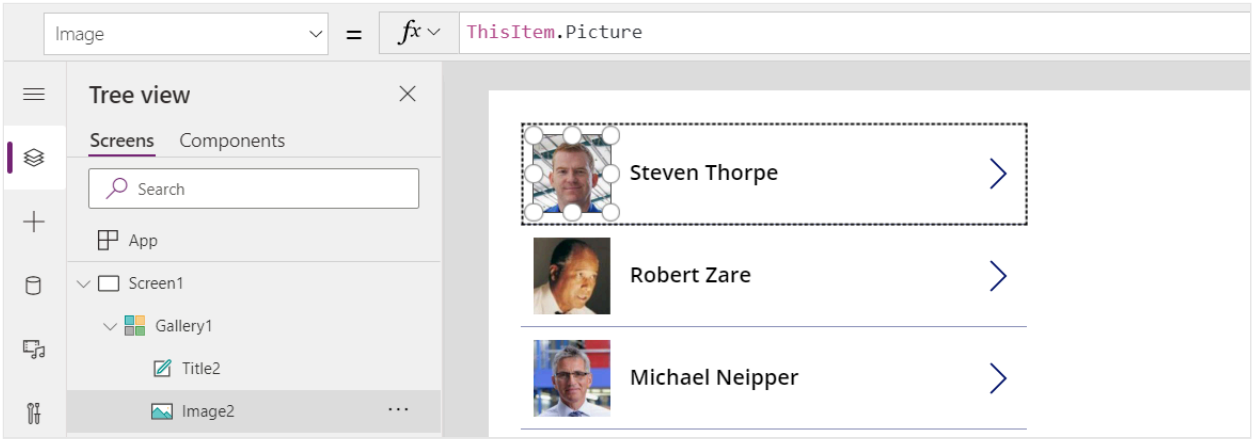
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

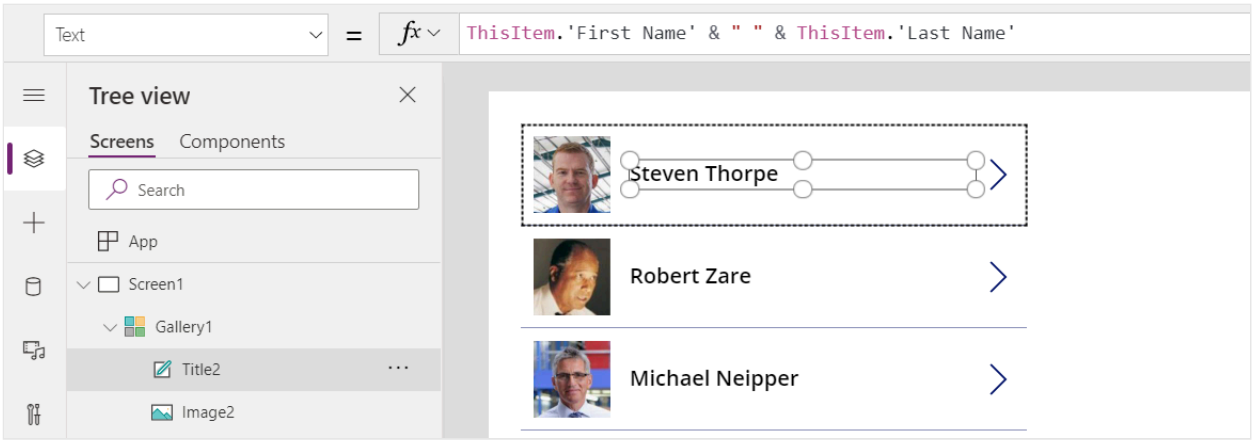
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

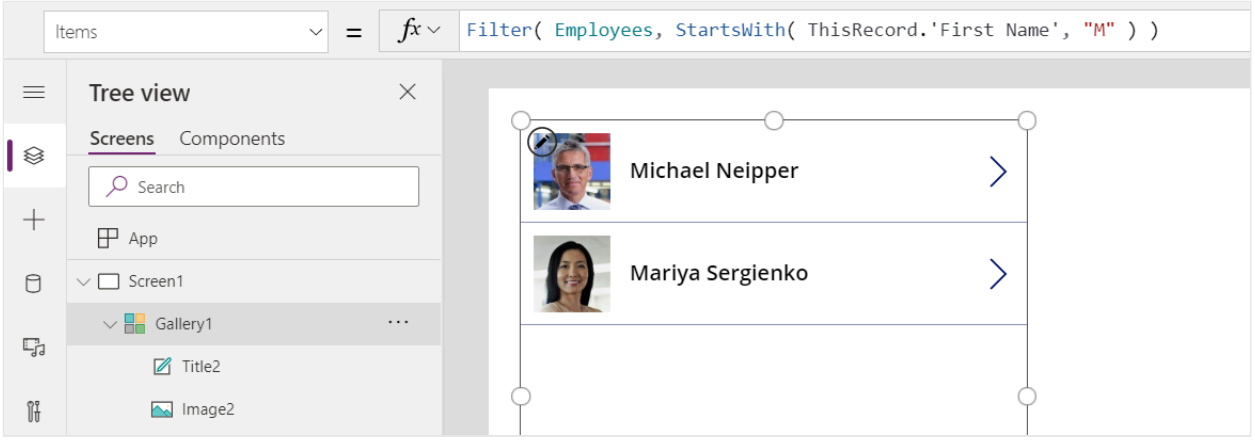


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

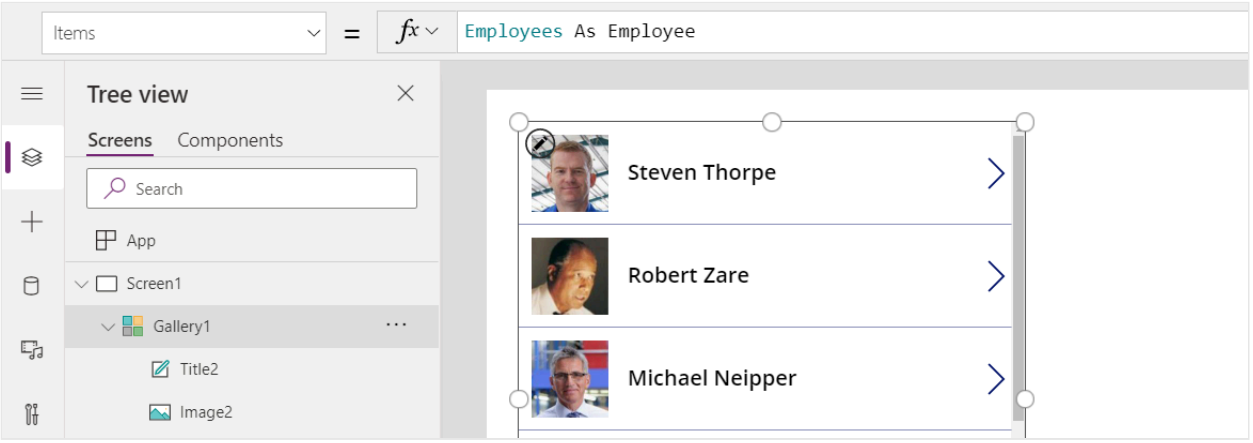
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

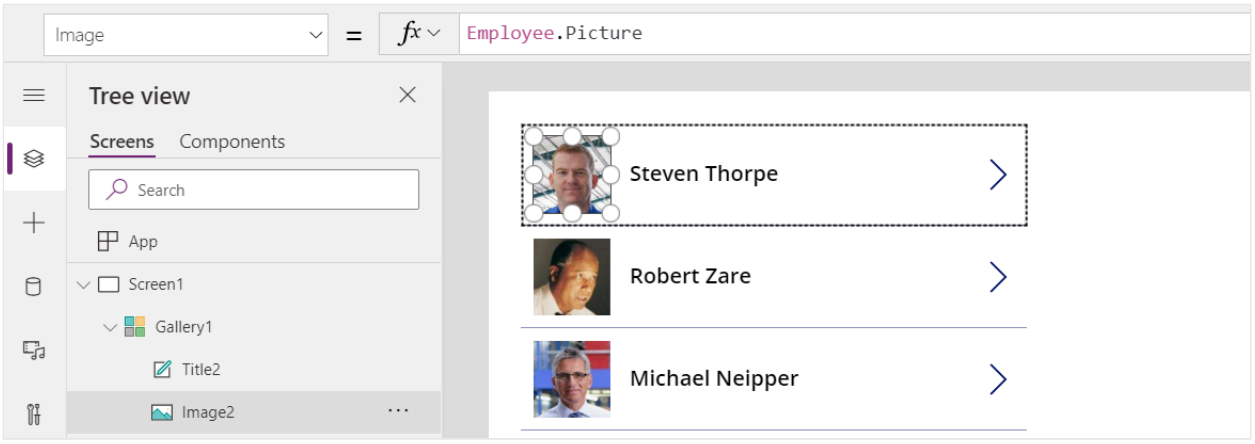
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

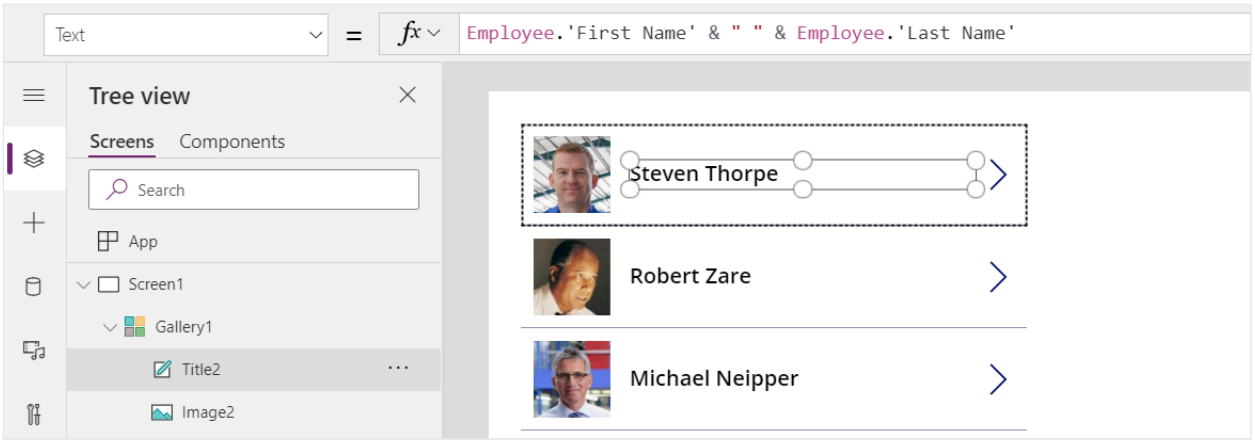
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

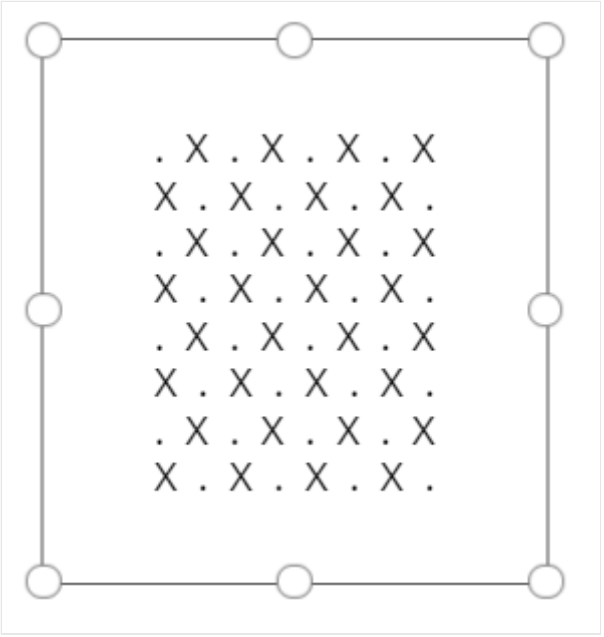
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



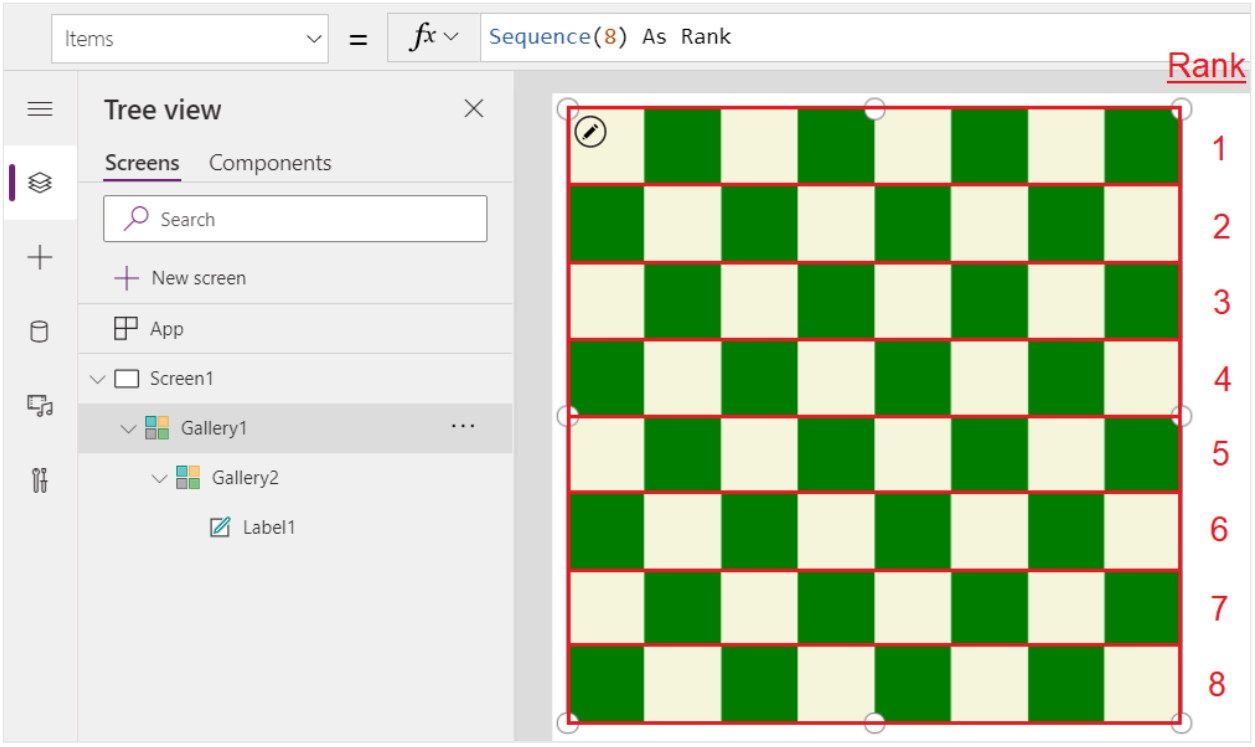
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

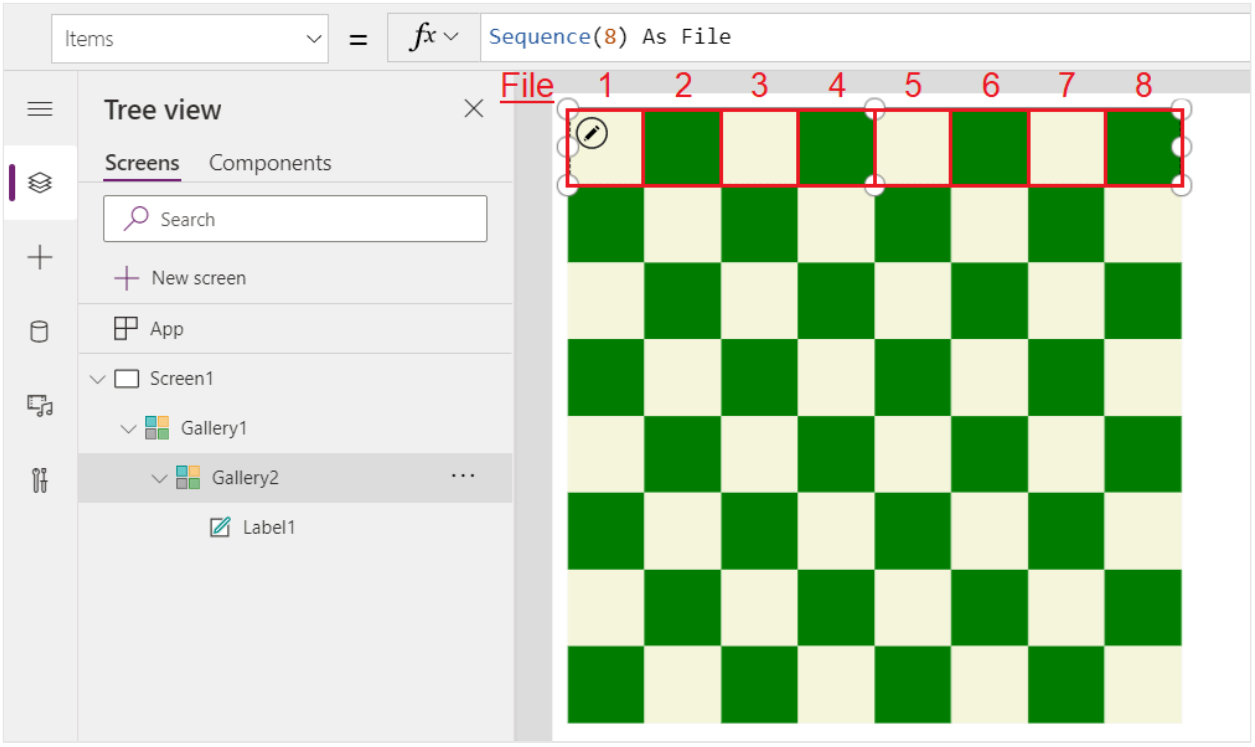
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

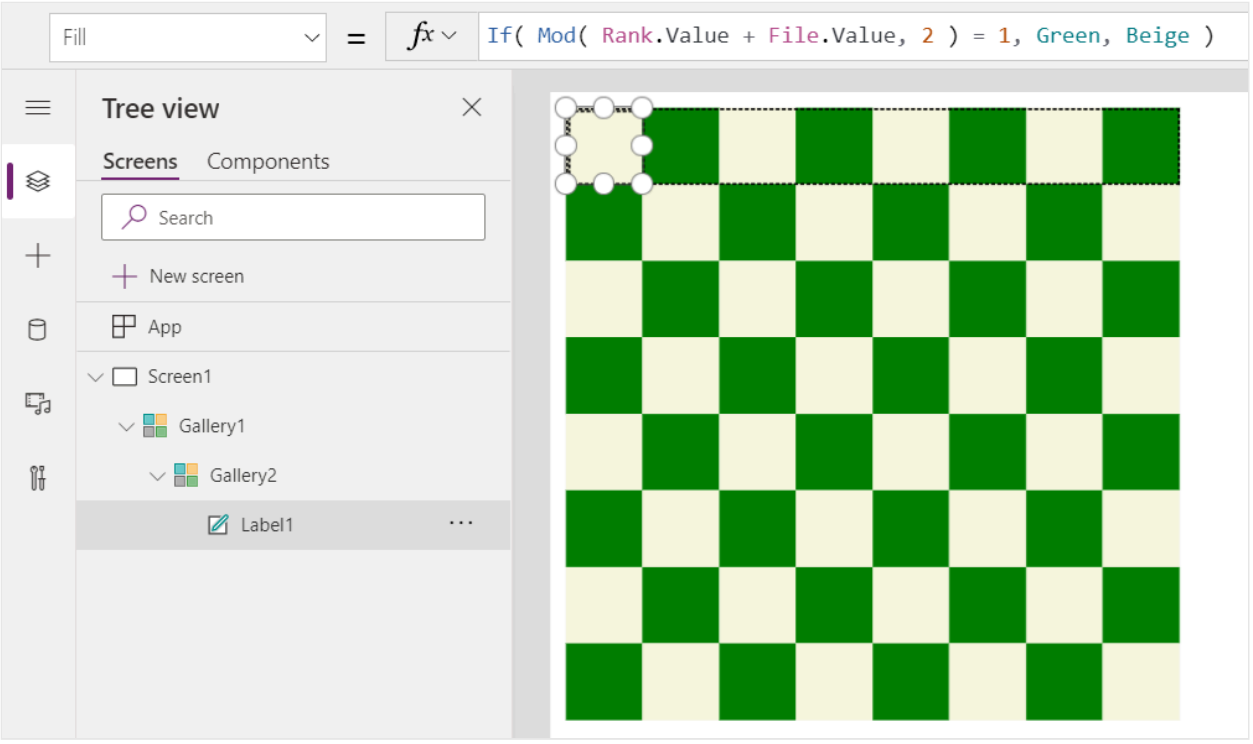
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

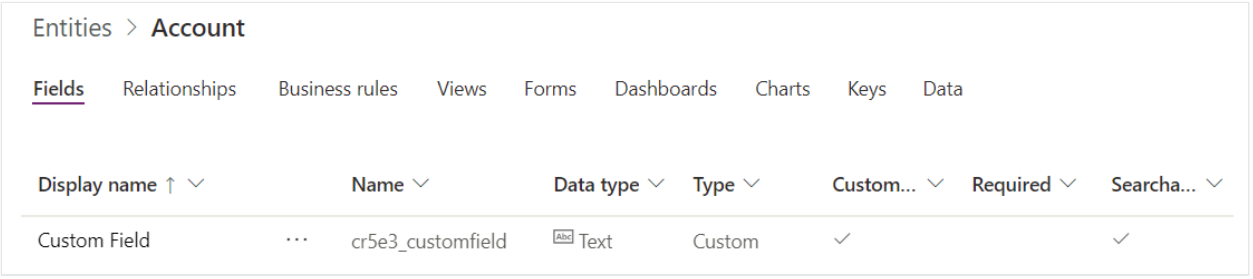
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

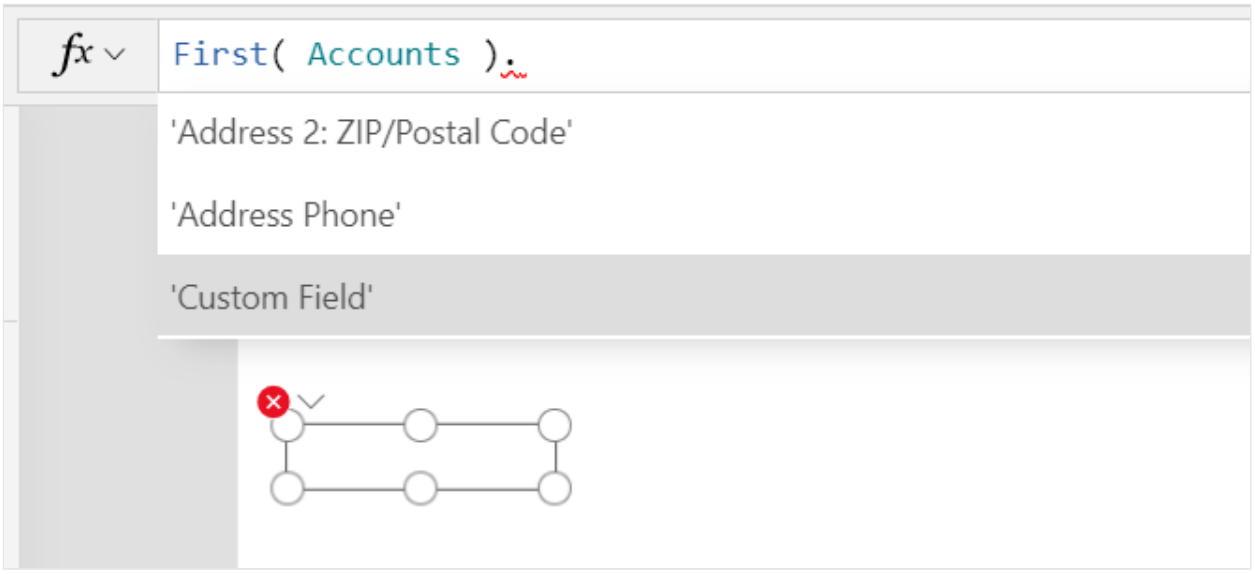
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

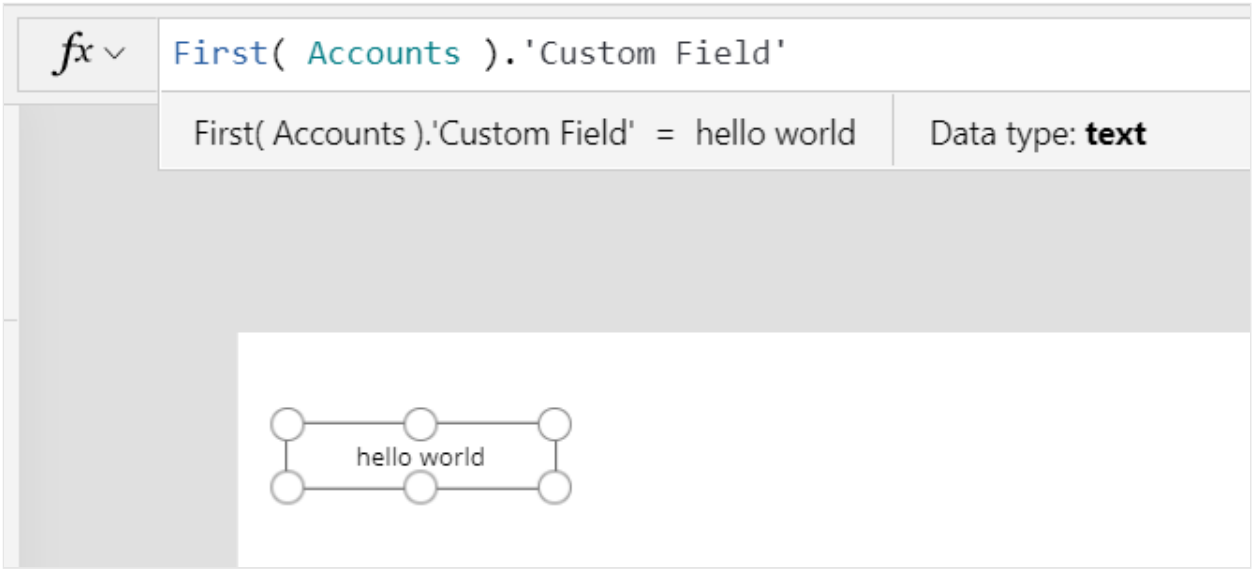
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



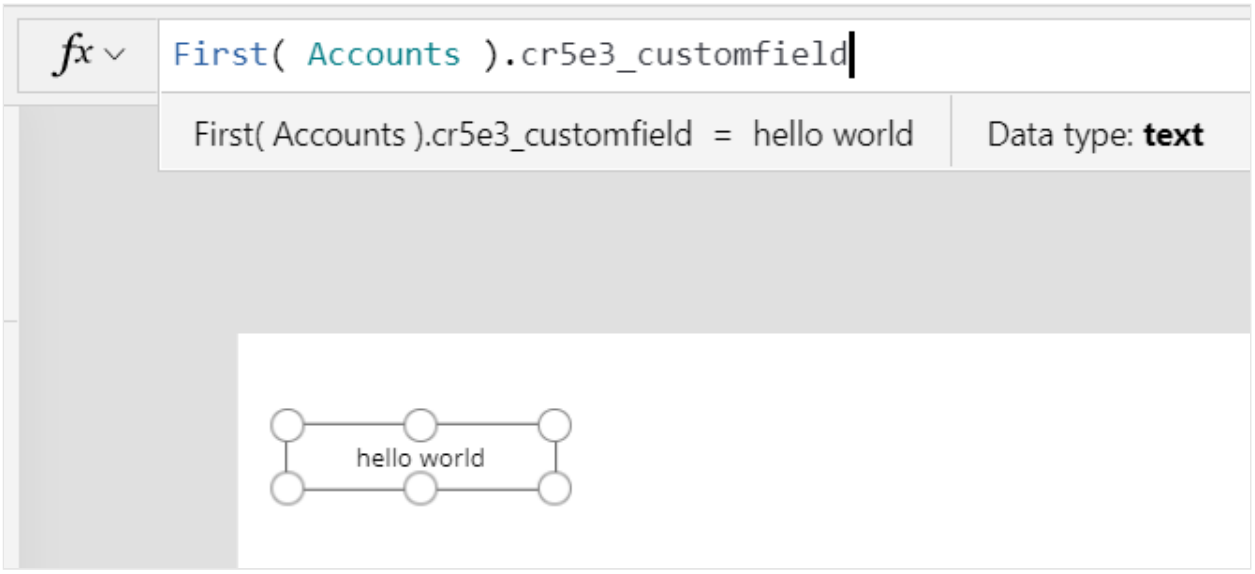
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

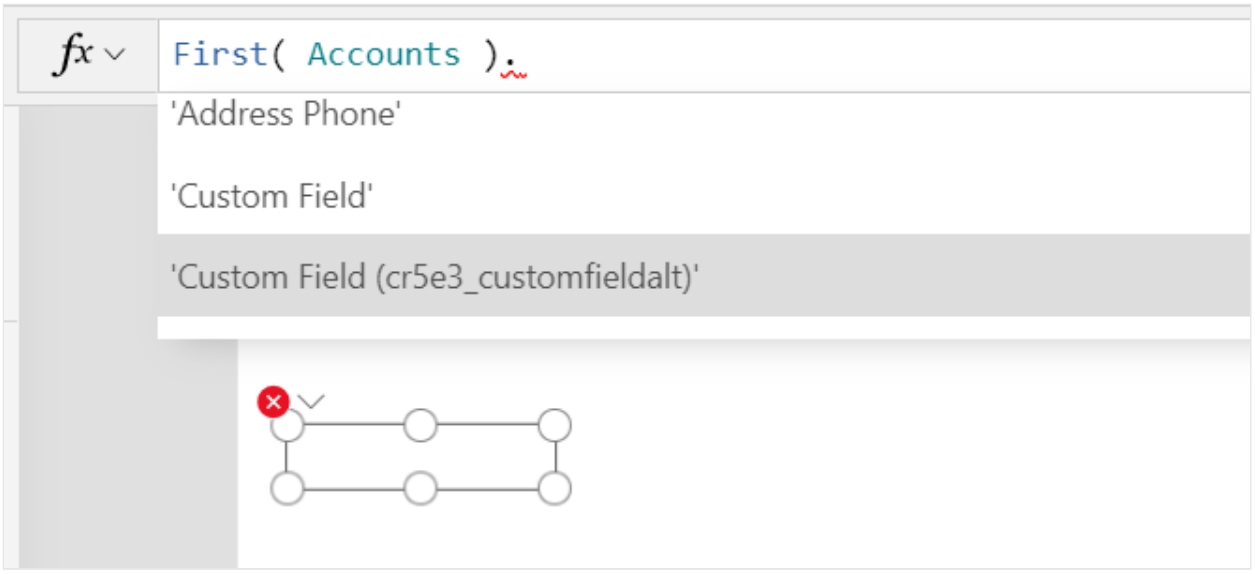


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Assert function

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

An assertion is a condition or an expression that evaluates to true or false in a test. If the expression returns false, the test case will fail. Assertions are used to validate the expected result of a test or test step, against the actual result and to fail the test if the condition is false. Assertions can be used to validate the state of controls in your app such as label values, list box selections and other control properties.

> ⓘ **Important**
>
> **Assert** function is only available for Test Studio in Power Apps.

Assertion messages, for both passed and failed assertions, are also contained in a Traces table in the TestCaseResult record.

Syntax

Assert(expression, message)

- *Expression* – Required. An expression that evaluates to true or false.
- *Message* – Not Required. A message that describes the assertion failure.

Examples

```
Assert(lblResult.Text = "Success", "lblResult value Expected : Success , Actual : " & lblResult.Text)
```

```
Assert(ListBox1.Selected.Value = "Success", "ListBox1 selection Expected : Success, Actual : " & ListBox1.Selected.Value)
```

```
Assert(kudosAfterTest = kudosBeforeTest + 1, "Kudos count. Expected : " & kudosBeforeTest + 1 & " Actual :" & kudosAfterTest)
```

See Also

Test Studio Overview
Working with Test Studio

Feedback

Was this page helpful? Yes No

Provide product feedback

AsType and IsType functions

06/19/2025

Expand table

Function	Applies to
AsType	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
IsType	✅ Canvas apps ✅ Copilot Studio ✅ Model-driven apps ✅ Dataverse functions ✅ Power Pages

Checks if a record reference is a specific table type (**IsType**) and treats the reference as that type (**AsType**).

> ⓘ **Note**
>
> PAC CLI pac power-fx commands don't support **IsType**.

Description

The **AsType** and **IsType** functions can be used to convert record references (for example polymorphic lookups in Dataverse) and dynamic values to typed values that can be used directly.

Record References

Read Understand record references and polymorphic lookups for a broader introduction and more details.

A lookup field usually refers to records in a particular table. Because the table type is well established, access the fields of the lookup using simple dot notation. For example, **First(Accounts).'Primary Contact'.'Full Name'** walks from the **Accounts** table to the **Primary Contact** record in the **Contacts** table and extracts the **Full Name** field.

Microsoft Dataverse also supports polymorphic lookup fields, which can refer to records from a set of tables, as in these examples.

Expand table

Lookup field	Can refer to
Owner	**Users** or **Teams**
Customer	**Accounts** or **Contacts**
Regarding	**Accounts**, **Contacts**, **Knowledge Articles**, etc.

In canvas-app formulas, use record references to work with polymorphic lookups. Because a record reference can refer to different tables, you can't know which fields are available at runtime when you write a formula. The *Record.Field* notation isn't available. These formulas need to adapt to the records that the app encounters when it runs.

The **IsType** function checks if a record reference refers to a specific table type. The function returns a Boolean TRUE or FALSE.

The **AsType** function treats a record reference as a specific table type, also called *casting*. You use the result as if it's a record of the table and use the *Record.Field* notation to access all the fields of that record. If the reference isn't of the specific type, an error occurs.

Use these functions together to first check the table type of a record and then treat it as a record of that type so the fields are available:

Power Fx

```
If( IsType( First( Accounts ).Owner, Users ),
    AsType( First( Accounts ).Owner, Users ).'Full Name',
    AsType( First( Accounts ).Owner, Teams ).'Team Name'
)
```

You need these functions only if you access the fields of a record reference. For example, you use record references in the **Filter** function without **IsType** or **AsType**:

Power Fx

```
Filter( Accounts, Owner = First( Users ) )
```

Similarly, you can use record references with the **Patch** function:

Power Fx

```
Patch( Accounts, First( Accounts ), { Owner: First( Teams ) } )
```

When you use these functions in a record context, such as within a **Gallery** or **Edit form** control, you might need the global disambiguation operator to reference the table type. For

example, this formula works for a gallery that displays a list of contacts where **Company Name** is a **Customer** lookup:

```
Power Fx
```

```
If( IsType( ThisItem.'Company Name', Accounts ),
    AsType( ThisItem.'Company Name', Accounts ).'Account Name',
    AsType( ThisItem.'Company Name', Contacts ).'Full Name'
)
```

For both functions, specify the type by using the name of the data source connected to the table. For the formula to work, add a data source to the app for any types you want to check or cast. For example, add the **Users** table as a data source if you want to use **IsType** and **AsType** with an **Owner** lookup and records from that table. Add only the data sources you use in your app; you don't need to add all the tables that a lookup could reference.

If the record reference is *blank*, **IsType** returns FALSE, and **AsType** returns *blank*. All fields of a *blank* record are also *blank*.

Dynamic values

> **Important**
>
> - Using **AsType** and **IsType** with **Dynamic** values is an experimental feature.
> - Experimental features aren't meant for production use and might not be complete. These features are available before an official release so that you can get early access and provide feedback. More information: Understand experimental, preview, and retired features in canvas apps
> - The behavior that this article describes is available only when the **User-defined types** experimental feature in Settings > Upcoming features > Experimental is turned on (it's off by default).
> - Your feedback is valuable to us. Let us know what you think in the Power Apps experimental features community forum.

A **Dynamic** value from a web API or the [**ParseJSON** function] needs to be converted to a specific typed value before you can use it in Power Fx. Here are some options:

1. Implicitly type the field at the point you use it. For example, an object converts to a number if you use it with the `+` operator, if it can be converted to a number. This option can cause unexpected conversions and can't convert records and tables as a whole.

2. Explicitly type each field individually with the **Decimal**, **Text**, **DateTime**, **GUID**, and other type constructor functions. This option is the most invasive to your formulas because you need to do each field separately.
3. Explicitly type JSON with the second argument to the **ParseJSON** function. This option is easy and avoids needing the **Dynamic** value.
4. Explicitly type a **Dynamic** value using the **AsType** function. You can also check the type before you try the conversion with the **IsType** function.

Syntax

AsType(*RecordReference*, *TableType*)

- *RecordReference* - Required. A record reference, often a lookup field that refers to a record in any of several tables.
- *TableType* - Required. The specific table to cast the record to.

AsType(*DynamicValue*, *TypeSpecification*)

- *DynamicValue* - Required. A dynamic value from the **ParseJSON** function or API call.
- *TypeSpecification* - Required. A type name or type specification you define with the **Type** function.

IsType(*RecordReference*, *TableType*)

- *RecordReference* - Required. A record reference, often a lookup field that can refer to a record in any of multiple tables.
- *TableType* - Required. The specific table to test for.

IsType(*DynamicValue*, *TypeSpecification*)

- *DynamicValue* - Required. A dynamic value from the **ParseJSON** function or an API call.
- *TypeSpecification* - Required. A type name or type specification defined with the **Type** function.

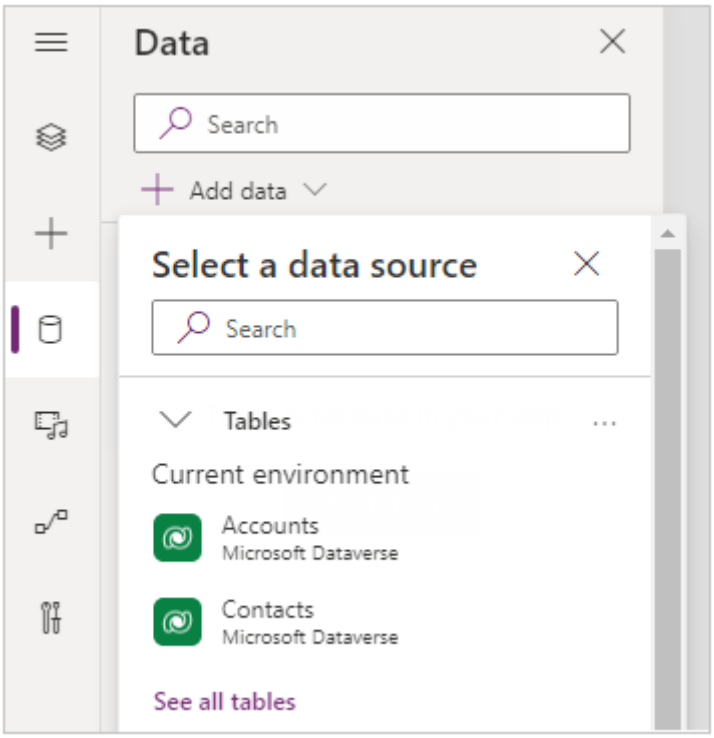
Examples

Record References

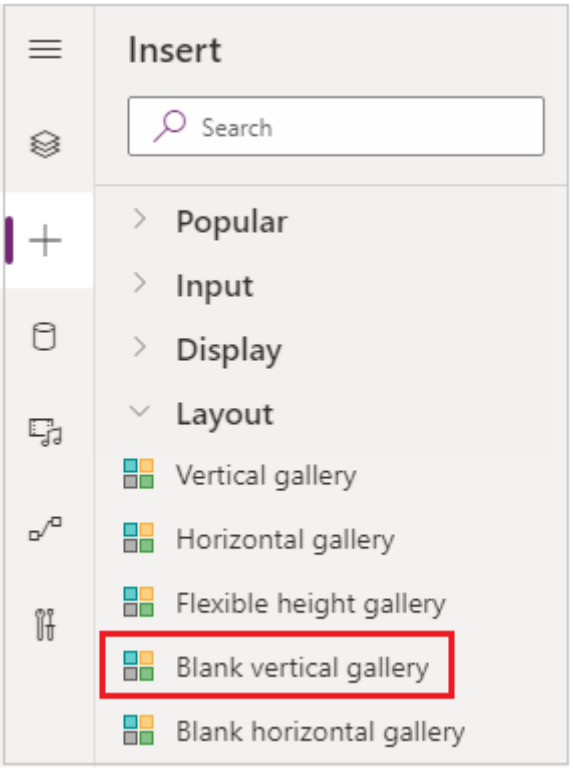
Understand record references and polymorphic lookups contains extensive examples.

1. Create a blank canvas app for tablets.

2. On the left pane, select **Data** > **Add data**, and then add **Accounts** and **Contacts** tables.

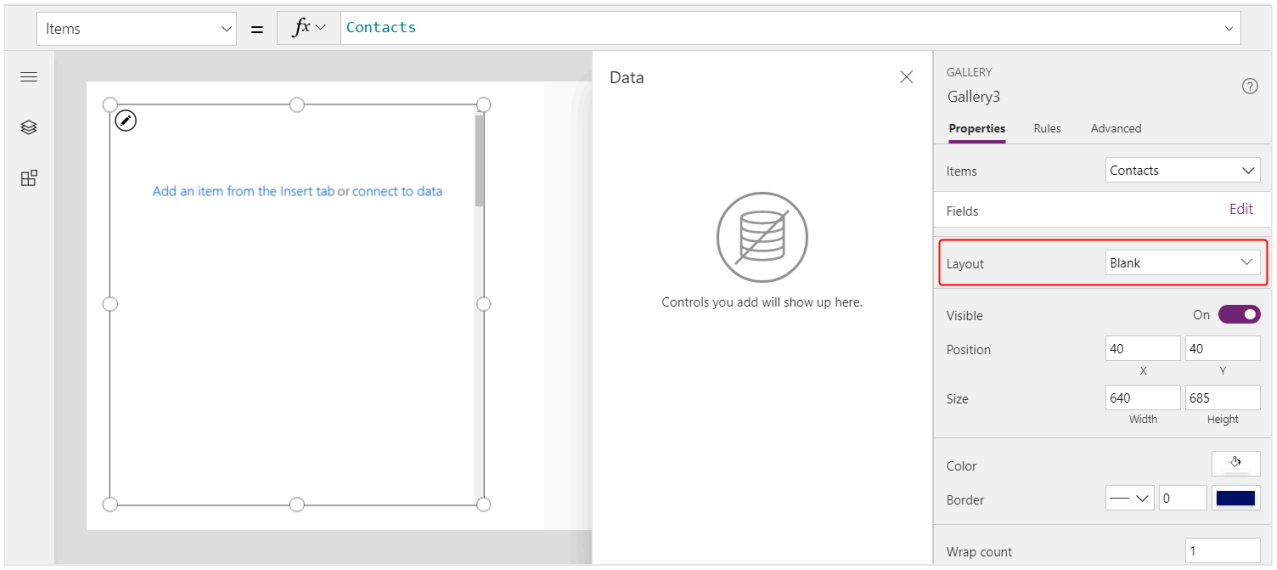


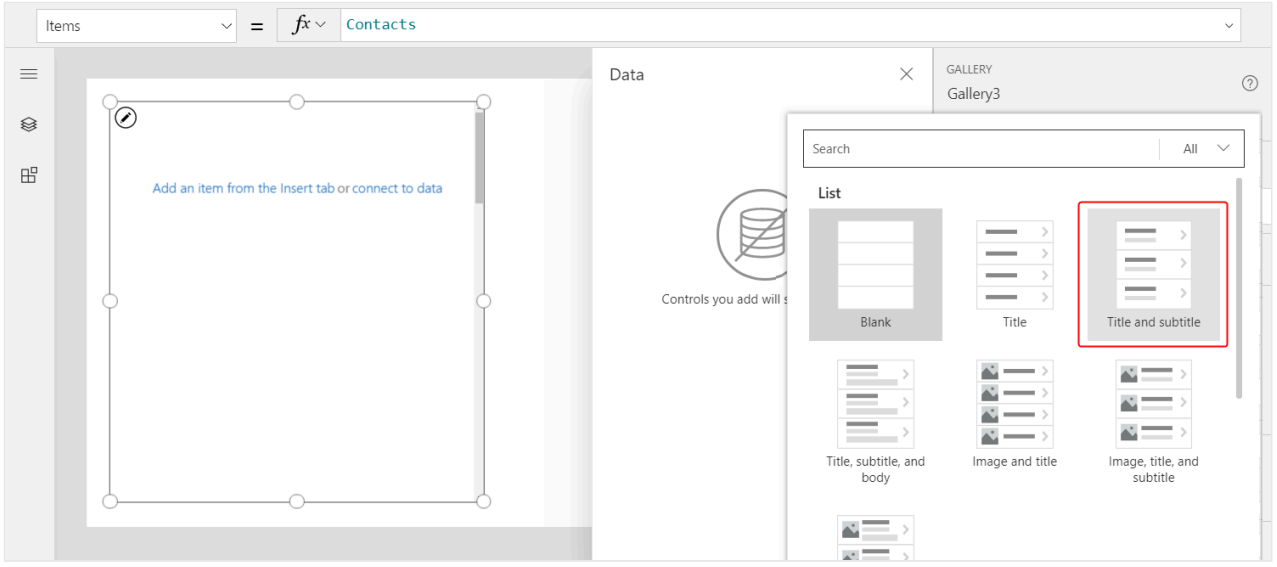
3. On the left pane, select + (Insert) > **Layout** > **Blank vertical gallery**.



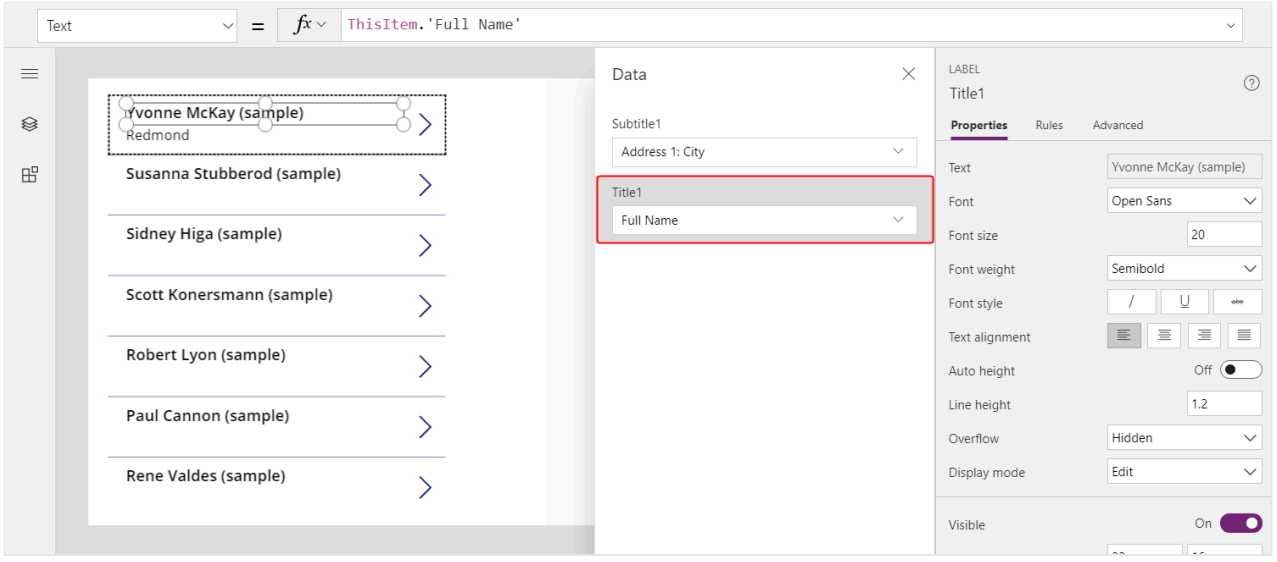
4. Select **Connect to data**, and then select **Contacts** as the data source.

5. Set the gallery layout to **Title and subtitle**.

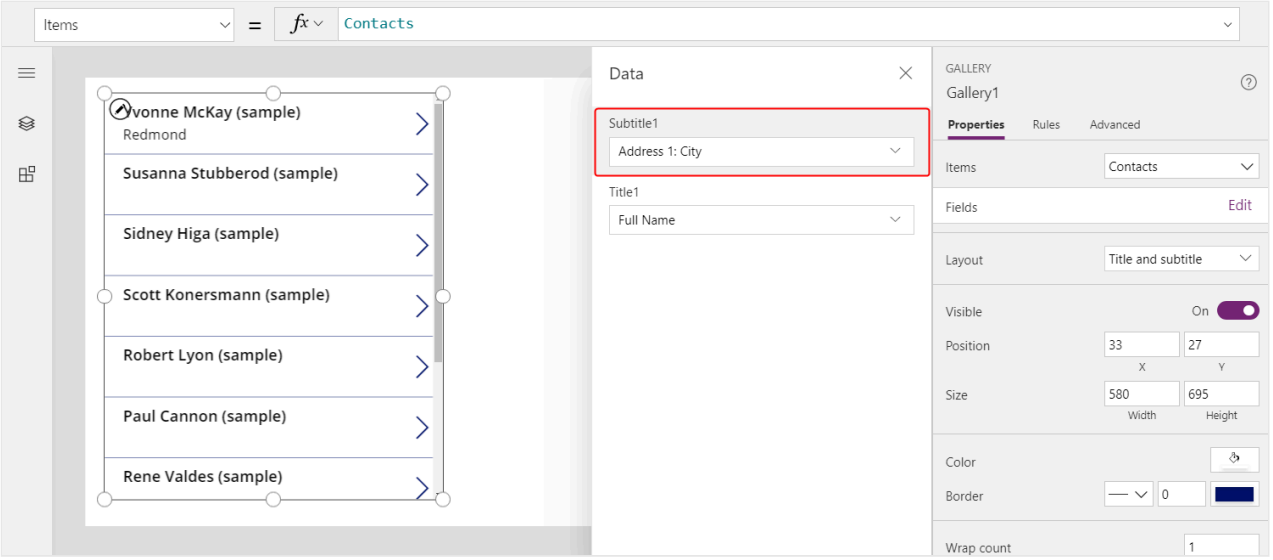




6. In the **Data** pane, open the **Title1** list, and then select **Full Name**.



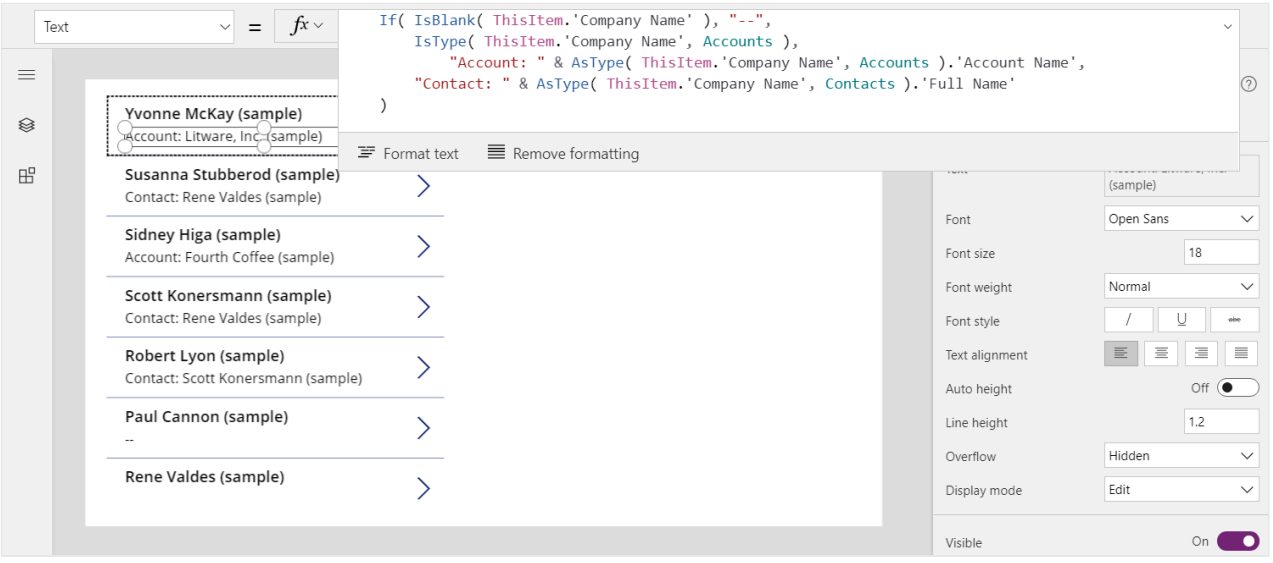
7. Select the **Subtitle1** label control.



8. Set the **Text** property of **Subtitle1** to this formula:

Power Fx

```
If( IsBlank( ThisItem.'Company Name' ), "--",
    IsType( ThisItem.'Company Name', Accounts ),
        "Account: " & AsType( ThisItem.'Company Name', Accounts ).'Account
Name',
    "Contact: " & AsType( ThisItem.'Company Name', Contacts ).'Full Name'
)
```



The subtitle in the gallery shows these values:

- "--" if the '**Company Name**' is *blank*.
- "Account: " and the **Account Name** field from the **Accounts** table if the **Company Name** field refers to an account.
- "Contact: " and the **Full Name** field from the **Contacts** table if the **Company Name** field refers to a contact.

Your results can differ because the sample data can be modified to show more types of results.

Dynamic values

The following examples show a simple JSON record interpreted in various ways by **ParseJSON**, **AsType**, and **IsType** in the Pac CLI Power Fx REPL.

In this first example, no type information is provided to **ParseJSON**, so it returns a dynamic value.

Power Apps

```
>> Set( rec, ParseJSON( "{""a"":1}" ) )
rec: <Dynamic: Use Value, Text, Boolean, or other functions to establish the type>
```

The field is implicitly converted to a number when used in a numerical context.

Power Apps

```
>> 1 + rec.a
2
```

As an alternative, this example explicitly converts the record to a typed Power Fx record with the second argument to **ParseJSON**.

Power Apps

```
>> ParseJSON( "{""a"":1}", Type( {a: Number} ) )
{a:1}
```

And another alternative, this example explicitly converts the record to a typed Power Fx record using **AsType**.

Power Apps

```
>> AsType( ParseJSON( "{""a"":1}" ), Type( {a: Number} ) )
{a:1}
```

Finally, if you aren't sure, this example tests the type before converting it with the **IsType** function.

Power Apps

```
>> IsType( ParseJSON( "{""a"":1}" ), Type( {a: Number} ) )
true
```

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Average, Max, Min, StdevP, Sum, and VarP functions

Article • 04/20/2025

Expand table

Functions	Applies to
Average **Max** **Min** **Sum**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
StdevP **VarP**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Aggregate functions that summarize a set of numbers.

Description

The **Average** function calculates the average, or arithmetic mean, of its arguments.

The **Max** function finds the maximum value.

The **Min** function finds the minimum value.

The **Sum** function calculates the sum of its arguments.

The **StdevP** function calculates the standard deviation of its arguments.

The **VarP** function calculates the variance of its arguments.

You can supply the values for these functions as:

- Separate arguments. For example, **Sum(1, 2, 3)** returns 6.
- A table and a formula to operate over that table. The aggregate will be calculated on the values of the formula for each record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

These functions operate on numeric values only. Other types of values, such as strings or records, are ignored. Use the **Value** function to convert a string into a number.

The **Average**, **Max**, **Min**, and **Sum** functions can be delegated when used with a data source that supports delegation for these functions. However, **StdevP** and **VarP** can't be delegated for any data sources. If delegation is not supported, only the first portion of the data will be retrieved and then the function applied locally. The result may not represent the complete story. A delegation warning will appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Average(*NumericalFormula1*, [*NumericalFormula2*, ...])
Max(*NumericalFormula1*, [*NumericalFormula2*, ...])
Min(*NumericalFormula1*, [*NumericalFormula2*, ...])
Sum(*NumericalFormula1*, [*NumericalFormula2*, ...])
StdevP(*NumericalFormula1*, [*NumericalFormula2*, ...])
VarP(*NumericalFormula1*, [*NumericalFormula2*, ...])

- *NumericalFormula(s)* - Required. Numeric values to operate on.

Average(*Table*, *NumericalFormula*)
Max(*Table*, *NumericalFormula*)
Min(*Table*, *NumericalFormula*)
Sum(*Table*, *NumericalFormula*)
StdevP(*Table*, *NumericalFormula*)
VarP(*Table*, *NumericalFormula*)

- *Table* - Required. Table to operate on.
- *NumericalFormula* - Required. Formula to evaluate for each record. The result of this formula is used for the aggregation. You can use columns of the table in the formula.

Examples

Step by step

Let's say that you had a data source named **Sales** that contained a **CostPerUnit** column and a **UnitsSold** column, and you set the **Text** property of a label to this function:
Sum(Sales, CostPerUnit * UnitsSold)

The label would show total sales by multiplying the values in those columns for each record and then adding the results from all records together:

Units sold	Cost per unit
30	$150
45	$125
20	$175

Total sales:

$13625

As a different example, let's say that you had sliders that were named **Slider1**, **Slider2**, and **Slider3** and a label with its **Text** property set to this formula:

Sum(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the sum of all values to which the sliders were set.

Average(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the average of all values to which the sliders were set.

Max(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the maximum of all values to which the sliders were set.

Min(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the minimum of all values to which the sliders were set.

StdevP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the standard deviation of all values to which the sliders were set.

VarP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the variance of all values to which the sliders were set.

Back and Navigate functions

Article • 03/22/2024

Changes which screen is displayed.

Overview

Most apps contain multiple screens. Use the **Back** and **Navigate** function to change which screen is displayed. For example, set the **OnSelect** property of a button to a formula that includes a **Navigate** function if you want to show a different screen when a user selects that button. In that formula, you can specify a visual transition, such as **Fade**, to control how one screen changes to another.

Back and **Navigate** change only which screen is displayed. Screens that aren't currently displayed continue to operate behind the scenes. You can build formulas that refer to properties of controls on other screens. For example, a user can change the value of a slider on one screen, navigate to a different screen that uses that value in a formula, and determine how it affects what happens in the new screen. The user can then navigate back to the original screen and confirm that the slider has kept its value.

Context variables are also preserved when a user navigates between screens. You can use **Navigate** to set one or more context variables for the screen that the formula will display, which is the only way to set a context variable from outside the screen. You can use this approach to pass parameters to a screen. If you've used another programming tool, this approach is similar to passing parameters to procedures.

Use the **App** object's **StartScreen** property to control the first screen to be displayed.

You can use either function only within a behavior formula.

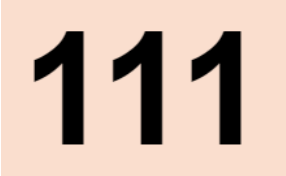
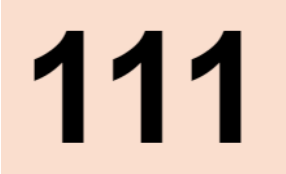
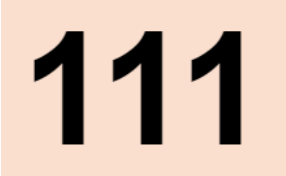
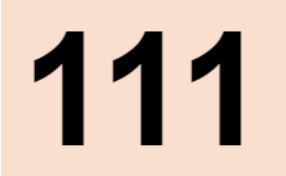
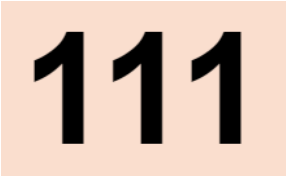
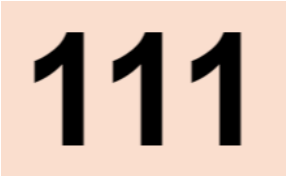
Navigate

Applies to: ✅ Canvas apps ✅ Model-driven apps

In the first argument, specify the name of the screen to display.

In the second argument, specify how the old screen changes to the new screen:

Expand table

Transition Argument	Description	Demonstration
ScreenTransition.Cover	The new screen slides into view, moving right to left, to cover the current screen.	
ScreenTransition.CoverRight	The new screen slides into view, moving left to right, to cover the current screen.	
ScreenTransition.Fade	The current screen fades away to reveal the new screen.	
ScreenTransition.None (Default)	The new screen quickly replaces the current screen.	
ScreenTransition.UnCover	The current screen slides out of view, moving right to left, to uncover the new screen.	
ScreenTransition.UnCoverRight	The current screen slides out of view, moving left to right, to uncover the new screen.	

You can use **Navigate** to create or update context variables of the new screen. As an optional third argument, pass a record that contains the context-variable name as a column name and the new value for the context variable. This record is the same as the record that you use with the **UpdateContext** function.

Set the **OnHidden** property of the old screen, the **OnVisible** property of the new screen, or both to make additional changes during the transition. The **App.ActiveScreen** property will be updated to reflect the change.

Navigate normally returns **true** but will return **false** if an error is encountered.

Context variables for navigation are explained in the article navigate between screens.

Back

Applies to: Canvas apps

The **Back** function returns to the screen that was most recently displayed.

For each **Navigate** call, the app tracks the screen that appeared and the transition. You can use successive **Back** calls to return all the way to the screen that appeared when the user started the app.

When the **Back** function runs, the inverse transition is used by default. For example, if a screen appeared through the **CoverRight** transition, **Back** uses **UnCover** (which is to the left) to return. **Fade** and **None** are their own inverses. Pass an optional argument to **Back** to force a specific transition.

Back normally returns **true** but returns **false** if the user hasn't navigated to another screen since starting the app.

Syntax

Back([*Transition*])

- *Transition* - Optional. The visual transition to use between the current screen and the previous screen. Refer to the list of valid values for this argument earlier in this article. By default, the transition through which a screen returns is the inverse of the transition through which it appeared.

Navigate(*Screen* [, *Transition* [, *UpdateContextRecord*]])

- *Screen* - Required. The screen to display. In place of *Screen*, you can also use a control that is on the screen you wish to navigate to.
- *Transition* - Optional. The visual transition to use between the current screen and the next screen. See the list of valid values for this argument earlier in this article. The default value is **None**.
- *UpdateContextRecord* - Optional. A record that contains the name of at least one column and a value for each column. This record updates the context variables of the new screen as if passed to the **UpdateContext** function.

Examples

Expand table

Formula	Description	Result
Navigate(Details)	Displays the **Details** screen with no transition or change in value for a context variable.	The **Details** screen appears quickly.
Navigate(Details, ScreenTransition.Fade)	Displays the **Details** screen with a **Fade** transition. No value of a context variable is changed.	The current screen fades away to show the **Details** screen.
Navigate(Details, ScreenTransition.Fade, { ID: 12 })	Displays the **Details** screen with a **Fade** transition, and updates the value of the **ID** context variable to **12**.	The current screen fades away to show the **Details** screen, and the context variable **ID** on that screen is set to **12**.
Navigate(Details, ScreenTransition.Fade, { ID: 12 , Shade: Color.Red })	Displays the **Details** screen with a **Fade** transition. Updates the value of the **ID** context variable to **12**, and updates the value of the **Shade** context variable to **Color.Red**.	The current screen fades away to show the **Details** screen. The context variable **ID** on the **Details** screen is set to **12**, and the context variable **Shade** is set to **Color.Red**. If you set the **Fill** property of a control on the **Details** screen to **Shade**, that control would display as red.
Back()	Displays the previous screen with the default return transition.	Displays the previous screen through the inverse transition of the transition through which the current screen appeared.
Back(ScreenTransition.Cover)	Displays the previous screen with the **Cover** transition.	Displays the previous screen through the **Cover** transition, regardless of the transition through which the current screen appeared.

Step-by-step

1. Create a blank app.
2. Add a second screen to it.

   The app contains two blank screens: **Screen1** and **Screen2**.
3. Set the **Fill** property of **Screen2** to the value `Gray`.

4. On **Screen2**, add a button, and set its **OnSelect** property to this formula:

```
Power Apps
Navigate( Screen1, ScreenTransition.Cover )
```

5. While holding down the **Alt** key, select the button.

   Screen1 appears with a white background through a transition that covers to the left.

6. On **Screen1**, add a button, and set its **OnSelect** property to this formula:

```
Power Apps
Back()
```

7. While holding down the **Alt** key, select the button.

   The second screen appears with a gray background through a transition that uncovers to the right (the inverse of **Cover**).

8. Select the button on each screen repeatedly to bounce back and forth.

See also

Using context variables

Blank, Coalesce, IsBlank, and IsEmpty functions

06/19/2025

Expand table

Functions	Applies to
Blank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
Coalesce **IsEmpty**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
IsBlank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests whether a value is blank or a table contains no records, and provides a way to create *blank* values.

Overview

Blank is a placeholder for "no value" or "unknown value." For example, a **Combo box** control's **Selected** property is *blank* if the user hasn't made a selection. Many data sources can store and return NULL values, which are represented in Power Apps as *blank*.

Any property or calculated value in Power Apps can be *blank*. For example, a Boolean value normally has one of two values: **true** or **false**. But in addition to these two, it can also be *blank* indicating that the state is not known. This is similar to Microsoft Excel, where a worksheet cell starts out as blank with no contents but can hold the values **TRUE** or **FALSE** (among others). At any time, the contents of the cell can again be cleared, returning it to a *blank* state.

Empty string refers to a string that contains no characters. The **Len** function returns zero for such a string and it can be written in a formulas as two double quotes with nothing in between `""`. Some controls and data sources use an empty string to indicate a "no value" condition. To simplify app creation, the **IsBlank** and **Coalesce** functions test for both *blank* values or empty strings.

In the context of the **IsEmpty** function, *empty* is specific to tables that contain no records. The table structure may be intact, complete with column names, but no data is in the table. A table

may start as empty, take on records and no longer be empty, and then have the records removed and again be empty.

Blank

The **Blank** function returns a *blank* value. Use this to store a NULL value in a data source that supports these values, effectively removing any value from the field.

IsBlank

The **IsBlank** function tests for a *blank* value or an empty string. The test includes empty strings to ease app creation since some data sources and controls use an empty string when there is no value present. To test specifically for a *blank* value use `if( Value = Blank(), ...` instead of **IsBlank**. The **IsBlank** function considers empty tables as not blank, and **IsEmpty** should be used to test a table.

When enabling error handling for existing apps, consider replacing **IsBlank** with IsBlankOrError to preserve existing app behavior. Prior to the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value for **IsBlank** is a boolean **true** or **false**.

Coalesce

The **Coalesce** function evaluates its arguments in order and returns the first value that isn't *blank* or an empty string. Use this function to replace a *blank* value or empty string with a different value but leave non-*blank* and non-empty string values unchanged. If all the arguments are *blank* or empty strings then the function returns *blank*, making **Coalesce** a good way to convert empty strings to *blank* values.

`Coalesce( value1, value2 )` is the more concise equivalent of `If( Not IsBlank( value1 ), value1, Not IsBlank( value2 ), value2 )` and doesn't require **value1** and **value2** to be evaluated twice. The **If** function returns *blank* if there is no "else" formula as is the case here.

All arguments to **Coalesce** must be of the same type; for example, you can't mix numbers with text strings. The return value from **Coalesce** is of this common type.

IsEmpty

The **IsEmpty** function tests whether a table contains any records. It's equivalent to using the CountRows function and checking for zero. You can check for data-source errors by combining **IsEmpty** with the Errors function.

The return value for **IsEmpty** is a Boolean **true** or **false**.

Syntax

Blank()

Coalesce(*Value1* [, *Value2*, ...])

- *Value(s)* – Required. Values to test. Each value is evaluated in order until a value that is not *blank* and not an empty string is found. Values after this point are not evaluated.

IsBlank(*Value*)

- *Value* – Required. Value to test for a *blank* value or empty string.

IsEmpty(*Table*)

- *Table* - Required. Table to test for records.

Examples

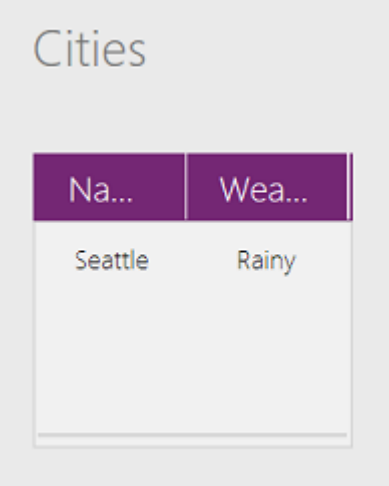
Blank

1. Create an app from scratch, and add a **Button** control.
2. Set the button's OnSelect property to this formula:

   ```
   Power Fx
   ClearCollect( Cities, { Name: "Seattle", Weather: "Rainy" } )
   ```

3. Preview your app, click or tap the button that you added, and then close Preview.
4. On the **File** menu, click or tap **Collections**.

   The **Cities** collection appears, showing one record with "Seattle" and "Rainy":



5. Click or tap the back arrow to return to the default workspace.

6. Add a **Label** control, and set its **Text** property to this formula:

```
Power Fx
IsBlank( First( Cities ).Weather )
```

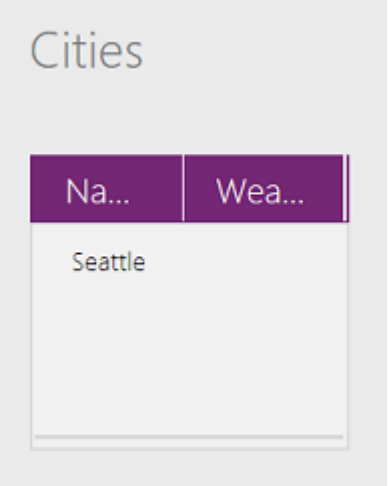
   The label shows **false** because the **Weather** field contains a value ("Rainy").

7. Add a second button, and set its **OnSelect** property to this formula:

```
Power Fx
Patch( Cities, First( Cities ), { Weather: Blank() } )
```

8. Preview your app, click or tap the button that you added, and then close Preview.

   The **Weather** field of the first record in **Cities** is replaced with a *blank*, removing the "Rainy" that was there previously.



   The label shows **true** because the **Weather** field no longer contains a value.

Coalesce

Expand table

Formula	Description	Result
Coalesce(Blank(), 1)	Tests the return value from the **Blank** function, which always returns a *blank* value. Because the first argument is *blank*, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**1**
Coalesce("", "2")	Tests the first argument which is an empty string. Because the first argument is an empty string, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**2**
Coalesce(Blank(), "", Blank(), "", "3", "4")	**Coalesce** starts at the beginning of the argument list and evaluates each argument in turn until a non-*blank* value and non-empty string is found. In this case, the first four arguments all return *blank* or an empty string, so evaluation continues to the fifth argument. The fifth argument is non-*blank* and non-empty string, so evaluation stops here. The value of the fifth argument is returned, and the sixth argument isn't evaluated.	**3**
Coalesce("")	Tests the first argument which is an empty string. Because the first argument is an empty string, and there are no more arguments, the function returns *blank*.	*blank*

IsBlank

1. Create an app from scratch, add a text-input control, and name it **FirstName**.
2. Add a label, and set its **Text** property to this formula:

   ```
   Power Fx
   ```

   ```
   If( IsBlank( FirstName.Text ), "First Name is a required field." )
   ```

   By default, the **Text** property of a text-input control is set to **"Text input"**. Because the property contains a value, it isn't blank, and the label doesn't display any message.
3. Remove all the characters from the text-input control, including any spaces.

   Because the **Text** property no longer contains any characters, it's an empty string, and **IsBlank(FirstName.Text)** will be **true**. The required field message is displayed.

For information about how to perform validation by using other tools, see the **Validate** function and working with data sources.

Other examples:

Expand table

Formula	Description	Result
IsBlank(Blank())	Tests the return value from the **Blank** function, which always returns a *blank* value.	**true**
IsBlank("")	A string that contains no characters.	**true**
IsBlank("Hello")	A string that contains one or more characters.	**false**
IsBlank(*AnyCollection*)	Because the collection exists, it isn't blank, even if it doesn't contain any records. To check for an empty collection, use **IsEmpty** instead.	**false**
IsBlank(Mid("Hello", 17, 2))	The starting character for **Mid** is beyond the end of the string. The result is an empty string.	**true**
IsBlank(If(false, false))	An **If** function with no *ElseResult*. Because the condition is always **false**, this **If** always returns *blank*.	**true**

IsEmpty

1. Create an app from scratch, and add a **Button** control.
2. Set the button's **OnSelect** property to this formula:

   Collect(IceCream, { Flavor: "Strawberry", Quantity: 300 }, { Flavor: "Chocolate", Quantity: 100 })
3. Preview your app, click or tap the button that you added, and then close Preview.

   A collection named **IceCream** is created and contains this data:

Flavor	Quantity
"Strawberry"	300
"Chocolate"	100

   This collection has two records and isn't empty. **IsEmpty(IceCream)** returns **false**, and **CountRows(IceCream)** returns **2**.
4. Add a second button, and set its **OnSelect** property to this formula:

   Clear(IceCream)
5. Preview your app, click or tap the second button, and then close Preview.

The collection is now empty:

Flavor	Quantity

The Clear function removes all the records from a collection, resulting in an empty collection. **IsEmpty(IceCream)** returns **true**, and **CountRows(IceCream)** returns **0**.

You can also use **IsEmpty** to test whether a calculated table is empty, as these examples show:

Expand table

Formula	Description	Result
IsEmpty([1, 2, 3])	The single-column table contains three records and, therefore, isn't empty.	**false**
IsEmpty([ ])	The single-column table contains no records and is empty.	**true**
IsEmpty(Filter([1, 2, 3], Value > 5))	The single-column table contains no values that are greater than 5. The result from the filter doesn't contain any records and is empty.	**true**

Calendar and Clock functions

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Retrieves calendar and clock information about the current locale.

Description

The **Calendar** and **Clock** functions are a set of functions that retrieve information about the current locale.

You can use these functions to display dates and times in the language of the current user. The single-column tables returned by **Calendar** and **Clock** functions can be used directly with the **Items** property of Dropdown and Listbox controls.

Expand table

Function	Description
Calendar.MonthsLong()	Single-column table containing the full names of each month, starting with "January".
Calendar.MonthsShort()	Single-column table containing the abbreviated names of each month, starting with "Jan" for January.
Calendar.WeekdaysLong()	Single-column table containing the full names of each weekday, starting with "Sunday".
Calendar.WeekdaysShort()	Single-column table containing the full names of each weekday, starting with "Sun" for Sunday.
Clock.AmPm()	Single-column table containing the long uppercase "AM" and "PM" designations. If the language uses a 24-hour clock, the table will be empty.
Clock.AmPmShort()	Single-column table containing the short uppercase "A" and "P" designations. If the language uses a 24-hour clock, the table will be empty.
Clock.IsClock24()	Boolean indicating if a 24-hour clock is used in this locale.

Use the **Text** function to format date and time values using this same information. The **Language** function returns the current language and region code.

Syntax

Calendar.MonthsLong()

Calendar.MonthsShort()

Calendar.WeekdaysLong()

Calendar.WeekdaysShort()

Clock.AmPm()

Clock.AmPmShort()

Clock.IsClock24()

Examples

1. Insert a Dropdown control.
2. Set the formula for the **Items** property to:
    - **Calendar.MonthsLong()**
3. Users of your app can now select a month in their own language. **MonthsLong** can be replaced with any of the single-column tables that are returned by **Calendar** to create weekday and time selectors.

In the United States, with **Language** returning "en-US", the following is returned by the **Calendar** functions:

Expand table

Formula	Description	Result
Calendar.MonthsLong()	The return value contains the full name of each month, starting with "January".	["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]
Calendar.MonthsShort()	The return value contains the abbreviated name of each month, starting with "January".	["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
Calendar.WeekdaysLong()	The return value contains the full name of each day, starting	["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",

Formula	Description	Result
	with "Sunday".	"Friday", "Saturday"]
Calendar.WeekdaysShort()	The return value contains the abbreviated name of each day, starting with "Sunday".	["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
Clock.AmPm()	This language uses a 12-hour clock. The return value contains the uppercase versions of the full AM and PM designations.	["AM", "PM"]
Clock.AmPmShort()	This language uses a 12-hour clock. The return value contains the uppercase versions of the short AM and PM designations.	["A", "P"]
Clock.IsClock24()	This language uses a 12-hour clock.	**false**

Feedback

Was this page helpful? Yes No

Provide product feedback

Char and UniChar functions

Article • 04/20/2025

Expand table

Function	Applies to
Char	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
UniChar	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Translates a character code into a string.

Description

The **Char** function translates a number into a string with the corresponding ASCII character.

The **UniChar** function translates a number into a string with the corresponding Unicode character.

If you pass a single number, the return value is the translated string version of that number. If you pass a single-column table that contains numbers, the return value is a single-column table of strings in a **Value** column. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

Char(*CharacterCode*)

- *CharacterCode* - Required. ASCII character code to translate.

Char(*CharacterCodeTable*)

- *CharacterCodeTable* - Required. Table of ASCII character codes to translate.

UniChar(*UnicodeCode*)

- *UnicodeCode* - Required. Unicode character code to translate.

UniChar(*UnicodeCodeTable*)

- *UnicodeCodeTable* - Required. Table of Unicode character codes to translate.

Examples

Single number

Expand table

Formula	Description	Result
Char(65)	Returns the character that corresponds to ASCII code 65.	"A"
Char(105)	Returns the character that corresponds to ASCII code 105.	"i"
Char(35)	Returns the character that corresponds to ASCII code 35.	"#"
UniChar(35)	Returns the character that corresponds to Unicode code 35.	"#"
UniChar(233)	Returns the character that corresponds to Unicode code 233.	"á"
UniChar(9829)	Returns the character that corresponds to Unicode code 9829.	"♥"

Single-column table

The example in this section converts numbers from a single-column table.

Expand table

Formula	Result
`Char( [ 65, 105 ] )`	A single-column table with a `Value` column containing the following values: "A", "i"
`Char( [ 35, 52 ] )`	A single-column table with a `Value` column containing the following values: "#", "4"
`UniChar( [ 71, 97, 114, 231, 111, 110 ] )`	A single-column table with a `Value` column containing the following values: "G", "a", "r", "ç", "o", "n"

Display a character map

1. On an empty screen in a tablet app, add a Gallery control with a **Blank Horizontal** layout, and then set these properties:

   - **Items**: `Sequence( 8, 0, 16 ) As HighNibble`
   - **Width**: `Parent.Width`
   - **Height**: `Parent.Height`

- **TemplateSize**: `Parent.Width / 8`
- **TemplatePadding**: 0
- **X**: 0
- **Y**: 0

2. Inside that gallery, add a **Gallery** control with a **Blank Vertical** layout, and then set these properties:

   - **Items**: `Sequence( 16, HighNibble.Value ) As FullCode`
   - **Width**: `Parent.Width / 8`
   - **Height**: `Parent.Height`
   - **TemplateSize**: `Parent.Height / 16`
   - **TemplatePadding**: 0
   - **X**: 0
   - **Y**: 0

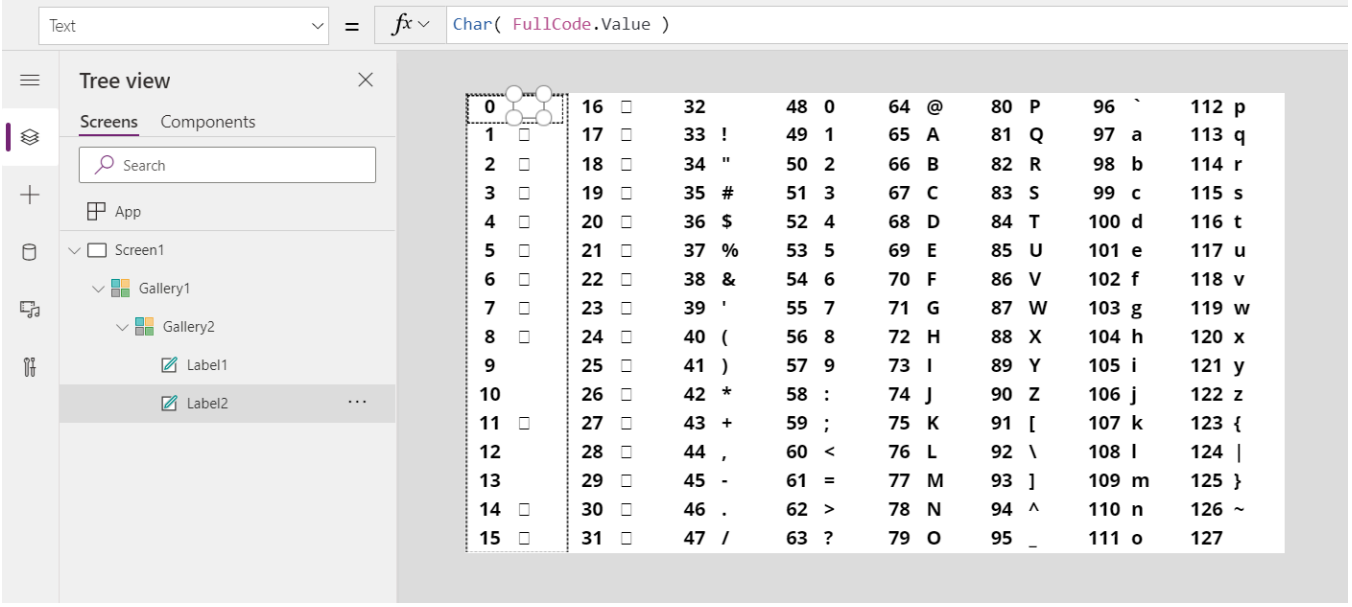
3. Inside the second (vertical) gallery, add a **Label** control, and set these properties:

   - **Text**: `FullCode.Value`
   - **Width**: `Parent.Width / 2`
   - **X**: 0
   - **Y**: 0
   - **Align**: `Center`
   - **FontWeight**: `Bold`
   - **Size**: 24

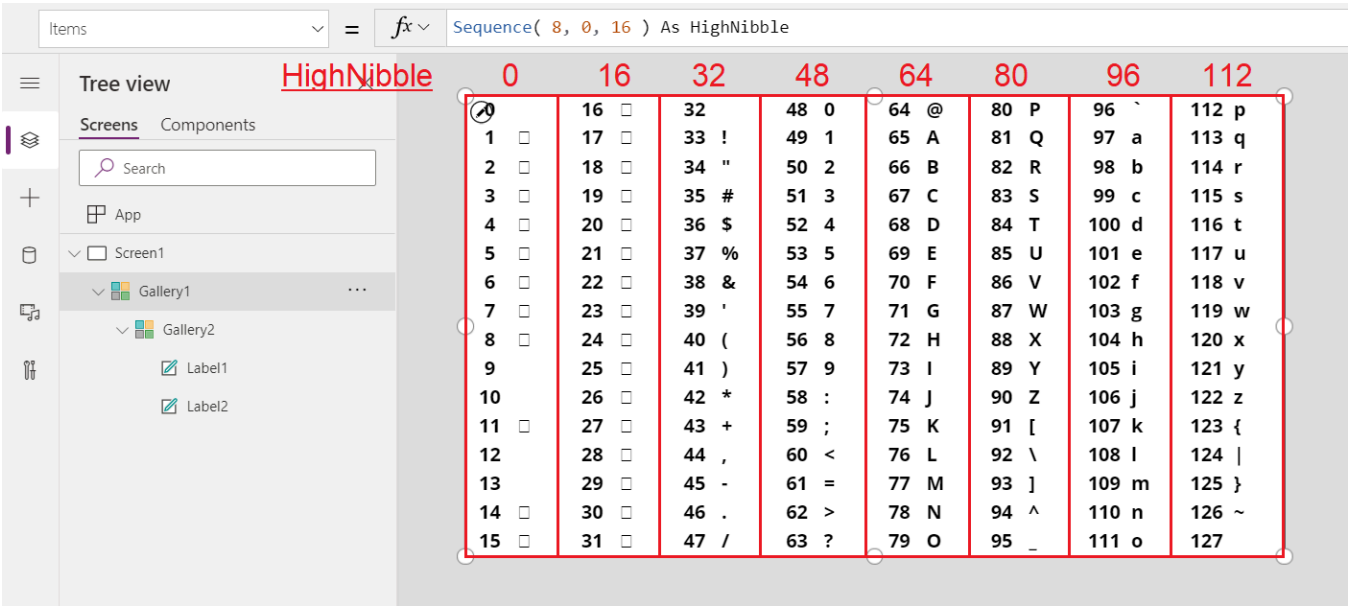
4. Inside the second (vertical) gallery, add another **Label** control, and set these properties:

   - **Text**: `Char( FullCode.Value )`
   - **Width**: `Parent.Width / 2`
   - **X**: `Parent.Width / 2`
   - **Y**: 0
   - **FontWeight**: `Bold`
   - **Size**: 24

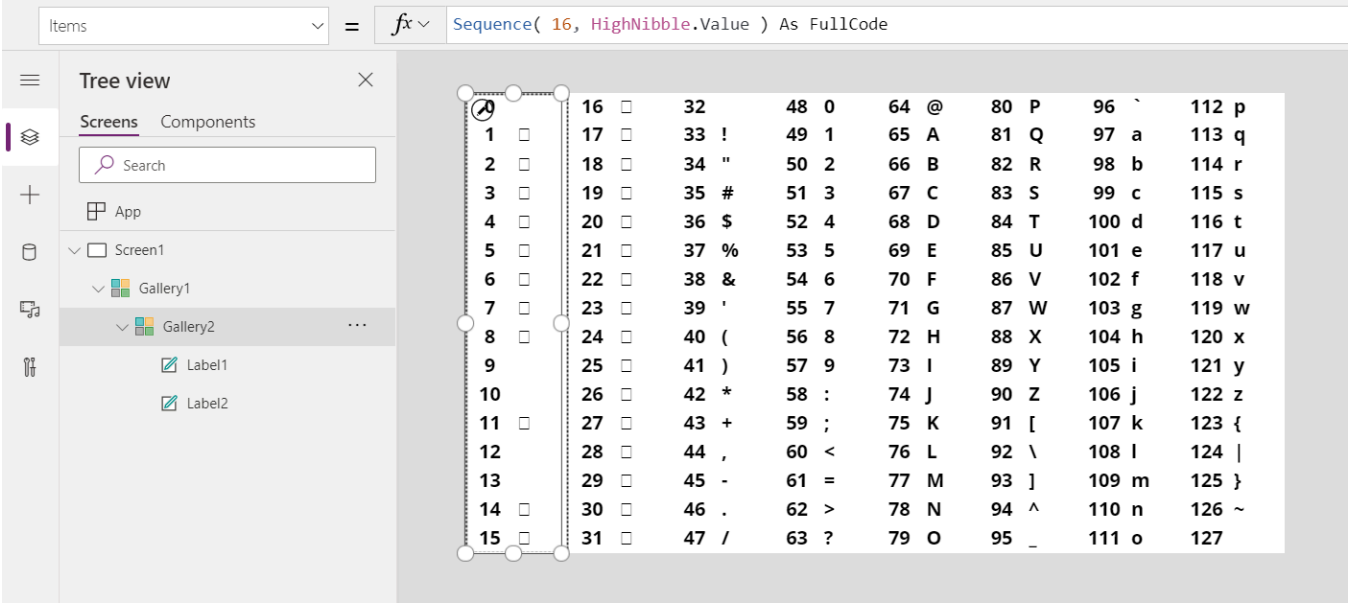
You've created a chart of the first 128 ASCII characters. Characters that appear as a small square can't be printed.



If you want to see how **FullCode.Value** gets its values. Let's begin with the outer horizontal gallery. Its **Items** property uses the **Sequence** function to create 8 columns, starting with 0 with increments of 16:

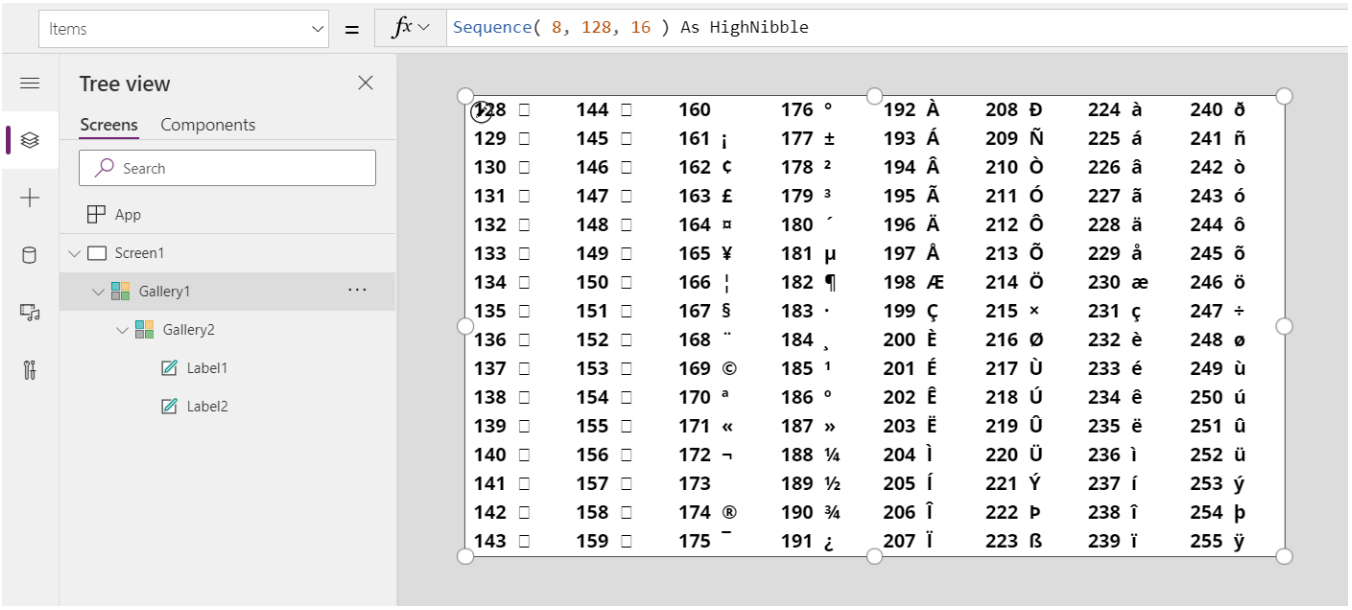


Nested within this gallery is another vertical gallery. Its **Items** property fills in the gap left by the increment of 16 from the outer gallery:

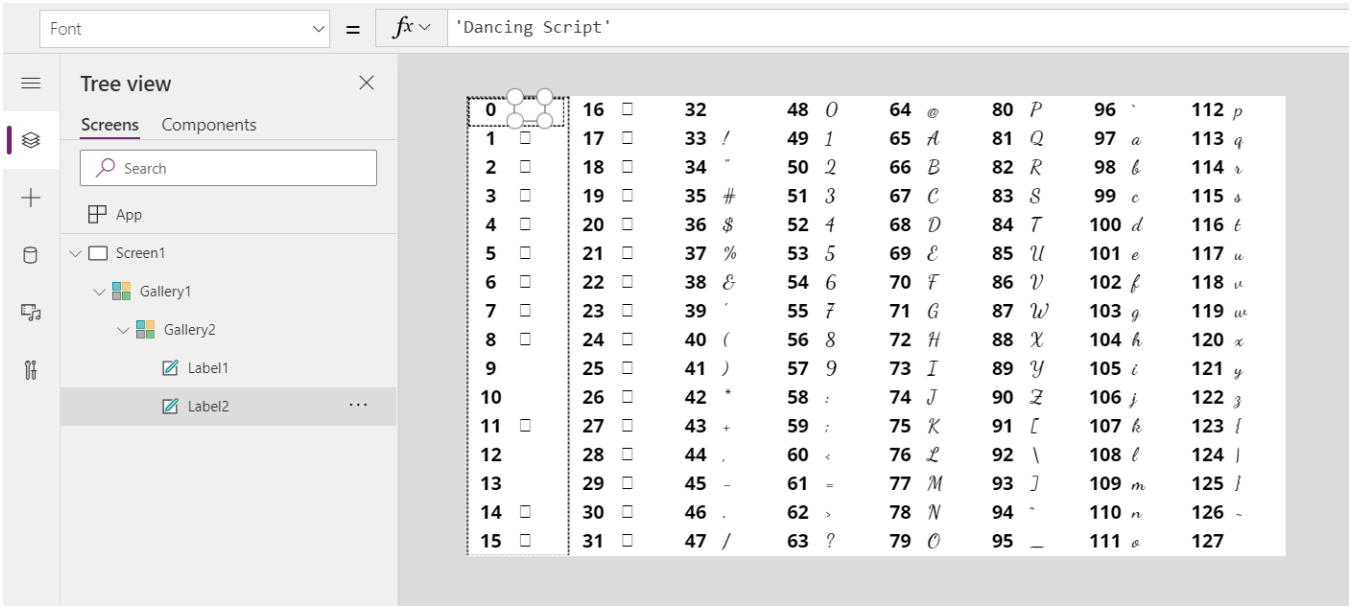


To show the extended ASCII characters, it is a simple matter of changing the starting point for the chart, set in the Sequence function for the outer gallery:

```
Sequence( 8, 128, 16 ) As HighNibble
```



Finally, to show the characters in a different font, set the **Font** property of the second label to a value such as '**Dancing Script**'.



0		16	□	32		48	0	64	@	80	P	96	`	112	p
1	□	17	□	33	!	49	1	65	A	81	Q	97	a	113	q
2	□	18	□	34	"	50	2	66	B	82	R	98	b	114	r
3	□	19	□	35	#	51	3	67	C	83	S	99	c	115	s
4	□	20	□	36	$	52	4	68	D	84	T	100	d	116	t
5	□	21	□	37	%	53	5	69	E	85	U	101	e	117	u
6	□	22	□	38	&	54	6	70	F	86	V	102	f	118	v
7	□	23	□	39	'	55	7	71	G	87	W	103	g	119	w
8	□	24	□	40	(	56	8	72	H	88	X	104	h	120	x
9		25	□	41	)	57	9	73	I	89	Y	105	i	121	y
10		26	□	42	*	58	:	74	J	90	Z	106	j	122	z
11	□	27	□	43	+	59	;	75	K	91	[	107	k	123	{
12		28	□	44	,	60	<	76	L	92	\	108	l	124	|
13		29	□	45	-	61	=	77	M	93	]	109	m	125	}
14	□	30	□	46	.	62	>	78	N	94	^	110	n	126	~
15	□	31	□	47	/	63	?	79	O	95	_	111	o	127	

Choices function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Returns a table of the possible values for a lookup column.

Description

The **Choices** function returns a table of the possible values for a lookup column.

Use the **Choices** function to provide a list of choices for your user to select from. This function is commonly used with the **Combo box** control in edit forms.

For a lookup, the table that **Choices** returns matches the foreign table that's associated with the lookup. By using **Choices**, you eliminate the need to add the foreign table as an additional data source. **Choices** returns all columns of the foreign table.

Because **Choices** returns a table, you can use **Filter**, **Sort**, **AddColumns**, and all the other table-manipulation functions to filter, sort, and shape the table.

At this time, you can't delegate **Choices**. If this limitation poses a problem in your app, add the foreign table as a data source, and use it directly.

Choices doesn't require column names to be strings and enclosed in double quotes, unlike the **ShowColumns**, **Search**, and other table functions. Provide the formula as if you were referencing the column directly.

Column references must be direct to the data source. For example, if the data source is **Accounts** and the lookup is **SLA**, the column reference would be **Accounts.SLA**. The reference can't pass through a function, a variable, or a control. Furthering this example, if **Accounts** is fed to a **Gallery** control, use the formula **Gallery.Selected.SLA** to reference the SLA for the selected account. However, this reference has passed through a control, so it can't be passed to the **Columns** function - you must still use **Accounts.SLA**.

At this time, you can use lookup columns only with SharePoint and Microsoft Dataverse.

Syntax

Choices(*column-reference* [, *text-filter*])

- *column-reference* - Required. A lookup column of a data source. Don't enclose the column name in double quotes. The reference must be directly to the column of the data source and not pass through a function or a control.
- *text-filter* - Optional. Filters the list of choices by only returning choices that start with the text specified in *text-filter*. If an empty string is specified, all choices will be returned.

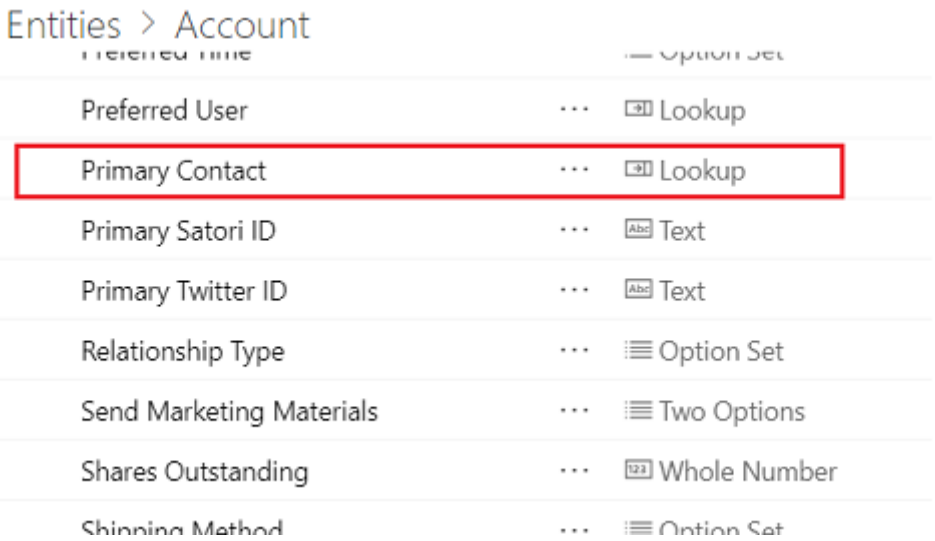
Examples

Choices for a lookup

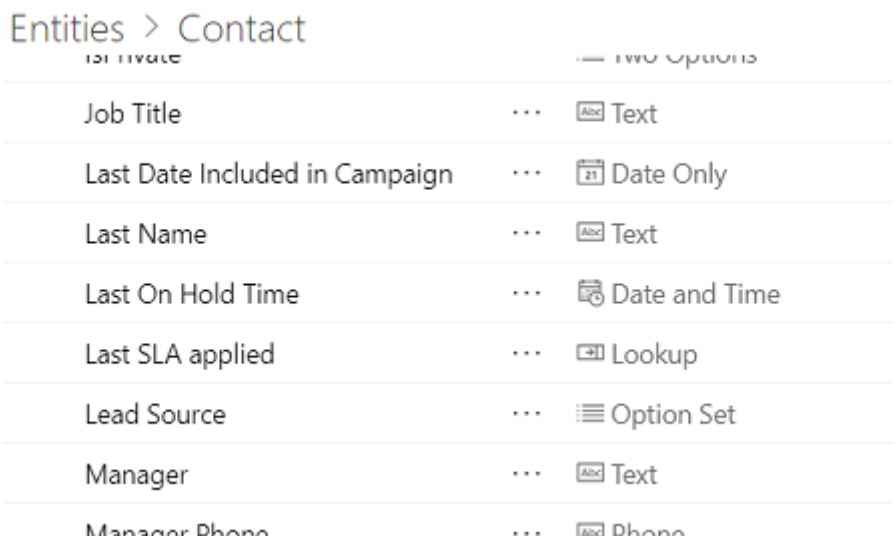
1. Create a database in Dataverse, and select the **Include sample apps and data** box.

   Many tables, such as **Accounts**, are created.

   Note: Table names are singular on make.powerapps.com and plural in Power Apps Studio.

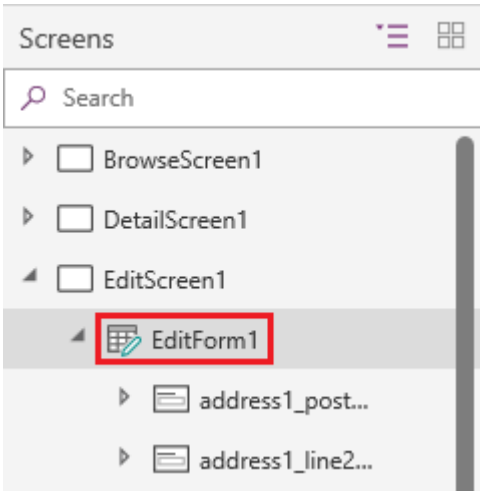
   The **Accounts** table has a **Primary Contact** column, which is a lookup to the **Contacts** table.



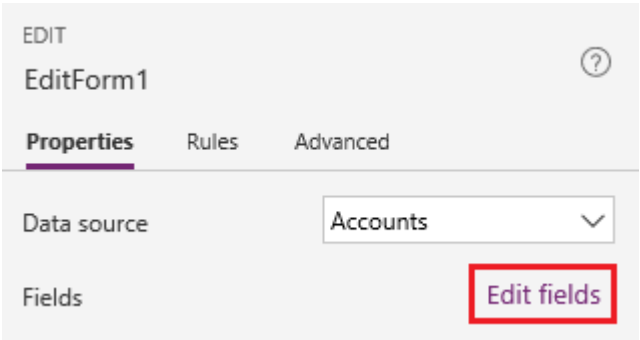
For each account, a contact is designated as the primary contact, or the primary contact is *blank*.

2. Generate an app from the **Accounts** table.

3. In the list of screens and controls near the left edge, scroll down until **EditScreen1** appears, and then select **EditForm1** just under it.

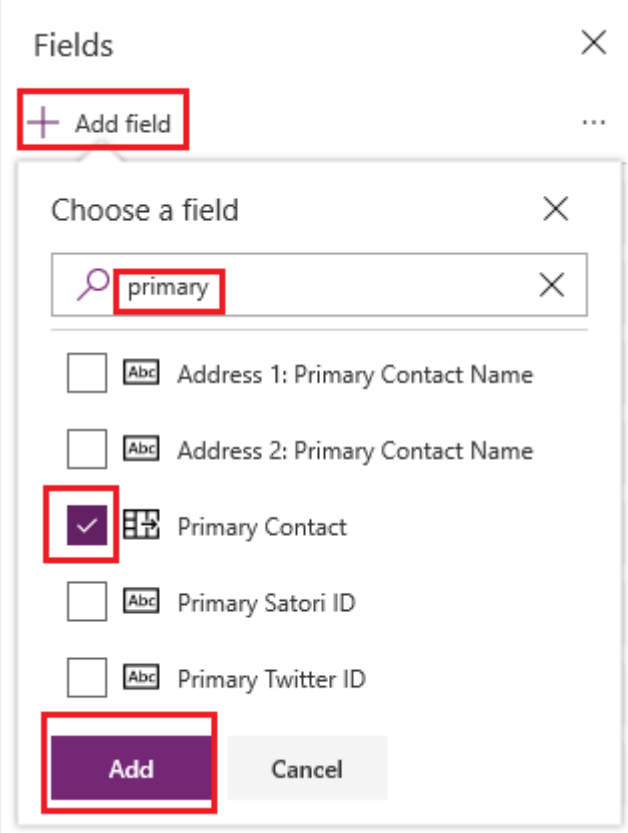


4. On the **Properties** tab of the right pane, select **Edit fields**.



5. In the **Fields** pane, select **Add field**.

6. Search for the **Primary Contact** field, select its check box, and then select **Add**.



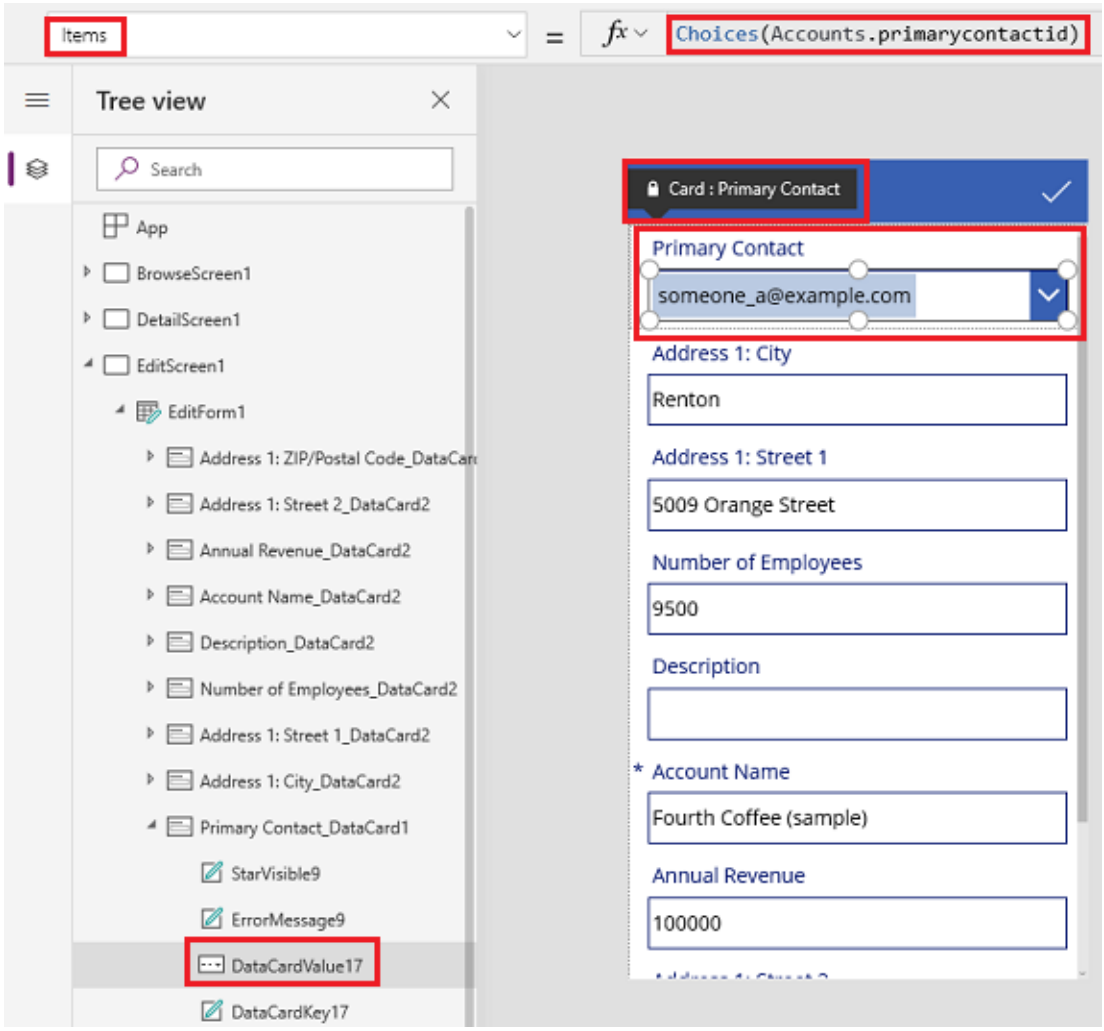
The **Primary Contact** field appears at the bottom of the form. If the field shows an error, select **Data sources** on the **View** tab, select the ellipsis (...) for the **Accounts** data source, and then select **Refresh**.

7. (optional) Drag the **Primary Contact** field from the bottom to the top of the list of fields.

8. In the card for **Primary Contact**, select the **Combo box** control.

   The **Items** property of that control is set to a formula that identifies the column by either its display name, as in the first example, or its logical name, as in the second example:

   - **Choices(Accounts.'Primary Contact')**
   - **Choices(Accounts.primarycontactid)**



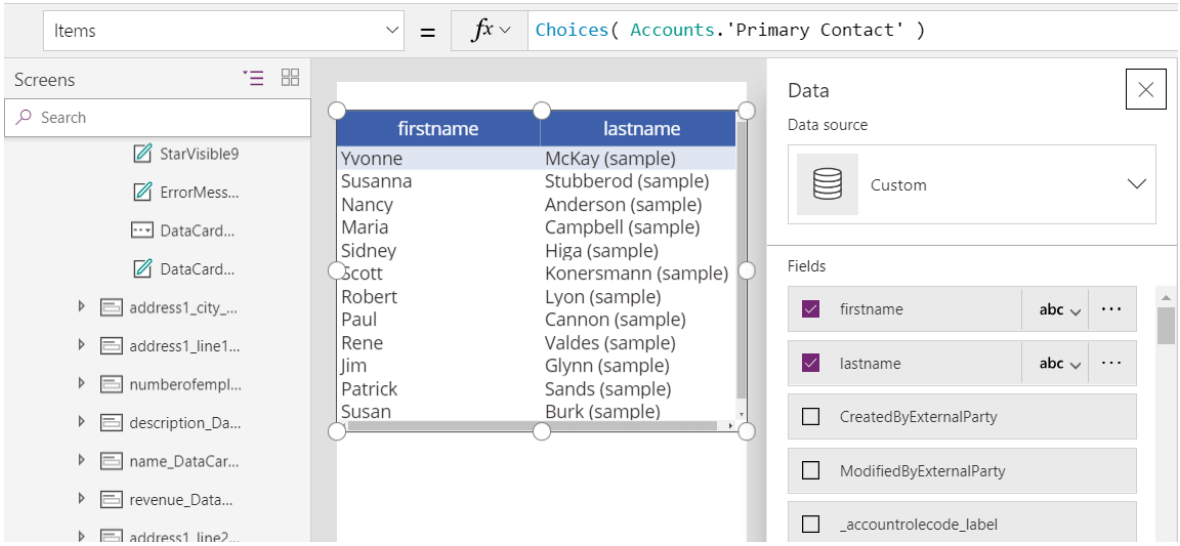
9. For illustration purposes, we can view the complete table returned by the **Choices** function in a **Data table** control. On the **Home** tab, select **New screen**, and then select **Blank**.

10. On the **Insert** tab, select **Data table**.

11. Set the **Items** property of the **Data table** control to this formula:

    Choices(Accounts.'Primary Contact')

12. In the middle of the **Data table** control, select the link that starts **Choose the fields...**, and then select the check boxes for the field or fields that you want to show (for example, **firstname** and **lastname**).


Items
=
Choices(Accounts.'Primary Contact')
Screens
Search
StarVisible9
ErrorMess...
DataCard...
DataCard...
address1_city_...
address1_line1...
numberofempl...
description_Da...
name_DataCar...
revenue_Data...
address1_line2...
firstname
lastname
Yvonne
McKay (sample)
Susanna
Stubberod (sample)
Nancy
Anderson (sample)
Maria
Campbell (sample)
Sidney
Higa (sample)
Scott
Konersmann (sample)
Robert
Lyon (sample)
Paul
Cannon (sample)
Rene
Valdes (sample)
Jim
Glynn (sample)
Patrick
Sands (sample)
Susan
Burk (sample)
Data
Data source
Custom
Fields
firstname
abc
lastname
abc
CreatedByExternalParty
ModifiedByExternalParty
_accountrolecode_label

Collect, Clear, and ClearCollect functions

Article • 04/11/2024

Creates and clears collections and adds records to any data source.

Collect

Applies to: ✅ Canvas apps ✅ Cards ✅ Dataverse low-code plug-ins ✅ Power Platform CLI ✅ Desktop flows

The **Collect** function adds records to a data source. The items to be added can be:

- A single value: The value is placed in the **Value** field of a new record. All other properties are left blank.
- A record: Each named property is placed in the corresponding property of a new record. All other properties are left blank.
- A table: Each record of the table is added as a separate record of the data source as described above. The table isn't added as a nested table to a record. To do this, wrap the table in a record first.

When used with a collection, additional columns will be created as needed. The columns for other data sources are fixed by the data source and new columns can't be added.

If the data source doesn't already exist, a collection is created.

Collections are sometimes used to hold global variables or make a temporary copy of a data source. Canvas apps are based on formulas that automatically recalculate as the user interacts with an app. Collections don't enjoy this benefit and their use can make your app harder to create and understand. Before using a collection in this manner, review working with variables.

You can also use the **Patch** function to create records in a data source.

Collect returns the modified data source as a table. **Collect** can only be used in a behavior formula.

Clear

Applies to: ✅ Canvas apps

The **Clear** function deletes all the records of a collection. The columns of the collection will remain.

Note that **Clear** only operates on collections and not other data sources. You can use **RemoveIf(*DataSource*, true)** for this purpose. Use caution as this will remove all records from the data source's storage and can affect other users.

You can use the **Remove** function to selectively remove records.

Clear has no return value. It can only be used in a behavior formula.

ClearCollect

Applies to: Canvas apps Model-driven apps

The **ClearCollect** function deletes all the records from a collection. And then adds a different set of records to the same collection. With a single function, **ClearCollect** offers the combination of **Clear** and then **Collect**.

ClearCollect returns the modified collection as a table. **ClearCollect** can only be used in a behavior formula.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Collect(*DataSource*, *Item*, ...)

- *DataSource* – Required. The data source that you want to add data to. If it doesn't already exist, a new collection is created.
- *Item(s)* - Required. One or more records or tables to add to the data source.

Clear(*Collection*)

- *Collection* – Required. The collection that you want to clear.

ClearCollect(*Collection*, *Item*, ...)

- *Collection* – Required. The collection that you want to clear and then add data to.
- *Item(s)* - Required. One or more records or tables to add to the data source.

Examples

Clearing and adding records to a data source

In these examples, you'll erase and add to a collection that's named **IceCream**. The data source begins with these contents:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200

Expand table

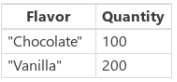
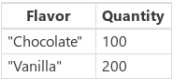
Formula	Description	Result
ClearCollect(IceCream, { Flavor: "Strawberry", Quantity: 300 })	Clears all data from the **IceCream** collection and then adds a record that includes a quantity of strawberry ice cream.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Strawberry"</td><td>300</td></tr></table> The **IceCream** collection has also been modified.
Collect(IceCream, { Flavor: "Pistachio", Quantity: 40 }, { Flavor: "Orange", Quantity: 200 })	Adds two records to the **IceCream** collection that includes a quantity of pistachio and orange ice cream.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Strawberry"</td><td>300</td></tr><tr><td>"Pistachio"</td><td>40</td></tr><tr><td>"Orange"</td><td>200</td></tr></table> The **IceCream** collection has also been modified.
Clear(IceCream)	Removes all records from the **IceCream** collection.	<table><tr><th>Flavor</th><th>Quantity</th></tr></table> The **IceCream** collection has also been modified.

For step-by-step examples of how to create a collection, see Create and update a collection.

Records and tables

These examples examine how record and table arguments to **Collect** and **ClearCollect** are handled.

Expand table

Formula	Description	Result
ClearCollect(IceCream, { Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 })	Clear all data and then adds two records to the **IceCream** collection that includes a quantity of chocolate and vanilla ice cream. The records to be added are provided as individual arguments to the function.	 The **IceCream** collection has also been modified.
ClearCollect(IceCream, Table({ Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 }))	Same as the previous example except that the records are combined in a table and passed in through a single argument. The contents of the table are extracted record by record before being added to the **IceCream** collection.	 The **IceCream** collection has also been modified.
ClearCollect(IceCream, { MyFavorites: Table({ Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 }) })	Same as the previous example except that the table is wrapped in a record. The records of the table aren't extracted and instead the entire table is added as a cell of the record.	 The **IceCream** collection has also been modified.

Collect, Clear, and ClearCollect functions

Article • 04/11/2024

Creates and clears collections and adds records to any data source.

Collect

Applies to: ✅ Canvas apps ✅ Cards ✅ Dataverse low-code plug-ins ✅ Power Platform CLI ✅ Desktop flows

The **Collect** function adds records to a data source. The items to be added can be:

- A single value: The value is placed in the **Value** field of a new record. All other properties are left blank.
- A record: Each named property is placed in the corresponding property of a new record. All other properties are left blank.
- A table: Each record of the table is added as a separate record of the data source as described above. The table isn't added as a nested table to a record. To do this, wrap the table in a record first.

When used with a collection, additional columns will be created as needed. The columns for other data sources are fixed by the data source and new columns can't be added.

If the data source doesn't already exist, a collection is created.

Collections are sometimes used to hold global variables or make a temporary copy of a data source. Canvas apps are based on formulas that automatically recalculate as the user interacts with an app. Collections don't enjoy this benefit and their use can make your app harder to create and understand. Before using a collection in this manner, review working with variables.

You can also use the **Patch** function to create records in a data source.

Collect returns the modified data source as a table. **Collect** can only be used in a behavior formula.

Clear

Applies to: ✅ Canvas apps

The **Clear** function deletes all the records of a collection. The columns of the collection will remain.

Note that **Clear** only operates on collections and not other data sources. You can use **RemoveIf(*DataSource*, true)** for this purpose. Use caution as this will remove all records from the data source's storage and can affect other users.

You can use the **Remove** function to selectively remove records.

Clear has no return value. It can only be used in a behavior formula.

ClearCollect

Applies to: Canvas apps Model-driven apps

The **ClearCollect** function deletes all the records from a collection. And then adds a different set of records to the same collection. With a single function, **ClearCollect** offers the combination of **Clear** and then **Collect**.

ClearCollect returns the modified collection as a table. **ClearCollect** can only be used in a behavior formula.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Collect(*DataSource*, *Item*, ...)

- *DataSource* – Required. The data source that you want to add data to. If it doesn't already exist, a new collection is created.
- *Item(s)* - Required. One or more records or tables to add to the data source.

Clear(*Collection*)

- *Collection* – Required. The collection that you want to clear.

ClearCollect(*Collection*, *Item*, ...)

- *Collection* – Required. The collection that you want to clear and then add data to.
- *Item(s)* - Required. One or more records or tables to add to the data source.

Examples

Clearing and adding records to a data source

In these examples, you'll erase and add to a collection that's named **IceCream**. The data source begins with these contents:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200

Expand table

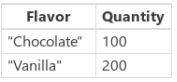
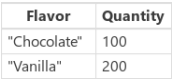
Formula	Description	Result
ClearCollect(IceCream, { Flavor: "Strawberry", Quantity: 300 })	Clears all data from the **IceCream** collection and then adds a record that includes a quantity of strawberry ice cream.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Strawberry"</td><td>300</td></tr></table> The **IceCream** collection has also been modified.
Collect(IceCream, { Flavor: "Pistachio", Quantity: 40 }, { Flavor: "Orange", Quantity: 200 })	Adds two records to the **IceCream** collection that includes a quantity of pistachio and orange ice cream.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Strawberry"</td><td>300</td></tr><tr><td>"Pistachio"</td><td>40</td></tr><tr><td>"Orange"</td><td>200</td></tr></table> The **IceCream** collection has also been modified.
Clear(IceCream)	Removes all records from the **IceCream** collection.	<table><tr><th>Flavor</th><th>Quantity</th></tr></table> The **IceCream** collection has also been modified.

For step-by-step examples of how to create a collection, see Create and update a collection.

Records and tables

These examples examine how record and table arguments to **Collect** and **ClearCollect** are handled.

Expand table

Formula	Description	Result
ClearCollect(IceCream, { Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 })	Clear all data and then adds two records to the **IceCream** collection that includes a quantity of chocolate and vanilla ice cream. The records to be added are provided as individual arguments to the function.	 The **IceCream** collection has also been modified.
ClearCollect(IceCream, Table({ Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 }))	Same as the previous example except that the records are combined in a table and passed in through a single argument. The contents of the table are extracted record by record before being added to the **IceCream** collection.	 The **IceCream** collection has also been modified.
ClearCollect(IceCream, { MyFavorites: Table({ Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 }) })	Same as the previous example except that the table is wrapped in a record. The records of the table aren't extracted and instead the entire table is added as a cell of the record.	 The **IceCream** collection has also been modified.

Calendar and Clock functions

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Retrieves calendar and clock information about the current locale.

Description

The **Calendar** and **Clock** functions are a set of functions that retrieve information about the current locale.

You can use these functions to display dates and times in the language of the current user. The single-column tables returned by **Calendar** and **Clock** functions can be used directly with the **Items** property of Dropdown and Listbox controls.

Expand table

Function	Description
Calendar.MonthsLong()	Single-column table containing the full names of each month, starting with "January".
Calendar.MonthsShort()	Single-column table containing the abbreviated names of each month, starting with "Jan" for January.
Calendar.WeekdaysLong()	Single-column table containing the full names of each weekday, starting with "Sunday".
Calendar.WeekdaysShort()	Single-column table containing the full names of each weekday, starting with "Sun" for Sunday.
Clock.AmPm()	Single-column table containing the long uppercase "AM" and "PM" designations. If the language uses a 24-hour clock, the table will be empty.
Clock.AmPmShort()	Single-column table containing the short uppercase "A" and "P" designations. If the language uses a 24-hour clock, the table will be empty.
Clock.IsClock24()	Boolean indicating if a 24-hour clock is used in this locale.

Use the **Text** function to format date and time values using this same information. The **Language** function returns the current language and region code.

Syntax

Calendar.MonthsLong()

Calendar.MonthsShort()

Calendar.WeekdaysLong()

Calendar.WeekdaysShort()

Clock.AmPm()

Clock.AmPmShort()

Clock.IsClock24()

Examples

1. Insert a Dropdown control.
2. Set the formula for the **Items** property to:
    - **Calendar.MonthsLong()**
3. Users of your app can now select a month in their own language. **MonthsLong** can be replaced with any of the single-column tables that are returned by **Calendar** to create weekday and time selectors.

In the United States, with **Language** returning "en-US", the following is returned by the **Calendar** functions:

Expand table

Formula	Description	Result
Calendar.MonthsLong()	The return value contains the full name of each month, starting with "January".	["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]
Calendar.MonthsShort()	The return value contains the abbreviated name of each month, starting with "January".	["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
Calendar.WeekdaysLong()	The return value contains the full name of each day, starting	["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",

Formula	Description	Result
	with "Sunday".	"Friday", "Saturday"]
Calendar.WeekdaysShort()	The return value contains the abbreviated name of each day, starting with "Sunday".	["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"]
Clock.AmPm()	This language uses a 12-hour clock. The return value contains the uppercase versions of the full AM and PM designations.	["AM", "PM"]
Clock.AmPmShort()	This language uses a 12-hour clock. The return value contains the uppercase versions of the short AM and PM designations.	["A", "P"]
Clock.IsClock24()	This language uses a 12-hour clock.	**false**

Feedback

Was this page helpful? Yes No

Provide product feedback

Blank, Coalesce, IsBlank, and IsEmpty functions

06/19/2025

Expand table

Functions	Applies to
Blank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
Coalesce **IsEmpty**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
IsBlank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests whether a value is blank or a table contains no records, and provides a way to create *blank* values.

Overview

Blank is a placeholder for "no value" or "unknown value." For example, a **Combo box** control's **Selected** property is *blank* if the user hasn't made a selection. Many data sources can store and return NULL values, which are represented in Power Apps as *blank*.

Any property or calculated value in Power Apps can be *blank*. For example, a Boolean value normally has one of two values: **true** or **false**. But in addition to these two, it can also be *blank* indicating that the state is not known. This is similar to Microsoft Excel, where a worksheet cell starts out as blank with no contents but can hold the values **TRUE** or **FALSE** (among others). At any time, the contents of the cell can again be cleared, returning it to a *blank* state.

Empty string refers to a string that contains no characters. The **Len** function returns zero for such a string and it can be written in a formulas as two double quotes with nothing in between `""`. Some controls and data sources use an empty string to indicate a "no value" condition. To simplify app creation, the **IsBlank** and **Coalesce** functions test for both *blank* values or empty strings.

In the context of the **IsEmpty** function, *empty* is specific to tables that contain no records. The table structure may be intact, complete with column names, but no data is in the table. A table

may start as empty, take on records and no longer be empty, and then have the records removed and again be empty.

Blank

The **Blank** function returns a *blank* value. Use this to store a NULL value in a data source that supports these values, effectively removing any value from the field.

IsBlank

The **IsBlank** function tests for a *blank* value or an empty string. The test includes empty strings to ease app creation since some data sources and controls use an empty string when there is no value present. To test specifically for a *blank* value use `if( Value = Blank(), ...` instead of **IsBlank**. The **IsBlank** function considers empty tables as not blank, and **IsEmpty** should be used to test a table.

When enabling error handling for existing apps, consider replacing **IsBlank** with IsBlankOrError to preserve existing app behavior. Prior to the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value for **IsBlank** is a boolean **true** or **false**.

Coalesce

The **Coalesce** function evaluates its arguments in order and returns the first value that isn't *blank* or an empty string. Use this function to replace a *blank* value or empty string with a different value but leave non-*blank* and non-empty string values unchanged. If all the arguments are *blank* or empty strings then the function returns *blank*, making **Coalesce** a good way to convert empty strings to *blank* values.

`Coalesce( value1, value2 )` is the more concise equivalent of `If( Not IsBlank( value1 ), value1, Not IsBlank( value2 ), value2 )` and doesn't require **value1** and **value2** to be evaluated twice. The **If** function returns *blank* if there is no "else" formula as is the case here.

All arguments to **Coalesce** must be of the same type; for example, you can't mix numbers with text strings. The return value from **Coalesce** is of this common type.

IsEmpty

The **IsEmpty** function tests whether a table contains any records. It's equivalent to using the CountRows function and checking for zero. You can check for data-source errors by combining **IsEmpty** with the Errors function.

The return value for **IsEmpty** is a Boolean **true** or **false**.

Syntax

Blank()

Coalesce(*Value1* [, *Value2*, ...])

- *Value(s)* – Required. Values to test. Each value is evaluated in order until a value that is not *blank* and not an empty string is found. Values after this point are not evaluated.

IsBlank(*Value*)

- *Value* – Required. Value to test for a *blank* value or empty string.

IsEmpty(*Table*)

- *Table* - Required. Table to test for records.

Examples

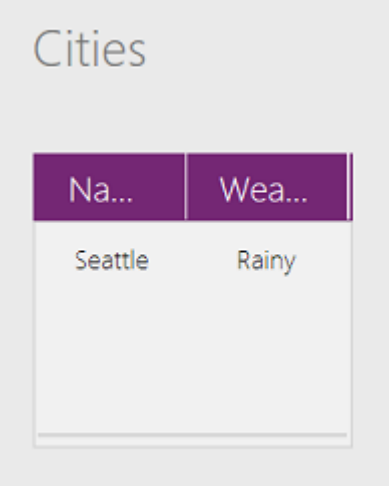
Blank

1. Create an app from scratch, and add a **Button** control.
2. Set the button's OnSelect property to this formula:

   ```
   Power Fx
   ClearCollect( Cities, { Name: "Seattle", Weather: "Rainy" } )
   ```

3. Preview your app, click or tap the button that you added, and then close Preview.
4. On the **File** menu, click or tap **Collections**.

   The **Cities** collection appears, showing one record with "Seattle" and "Rainy":



5. Click or tap the back arrow to return to the default workspace.

6. Add a **Label** control, and set its **Text** property to this formula:

```
Power Fx
IsBlank( First( Cities ).Weather )
```

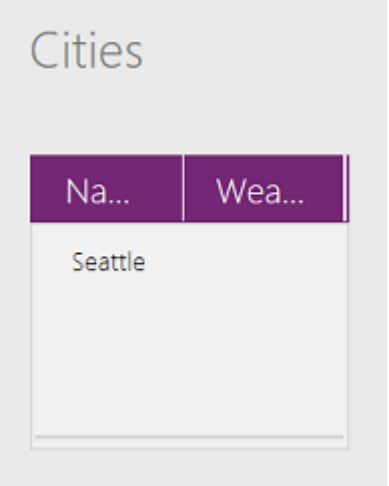
   The label shows **false** because the **Weather** field contains a value ("Rainy").

7. Add a second button, and set its **OnSelect** property to this formula:

```
Power Fx
Patch( Cities, First( Cities ), { Weather: Blank() } )
```

8. Preview your app, click or tap the button that you added, and then close Preview.

   The **Weather** field of the first record in **Cities** is replaced with a *blank*, removing the "Rainy" that was there previously.



   The label shows **true** because the **Weather** field no longer contains a value.

Coalesce

Expand table

Formula	Description	Result
Coalesce(Blank(), 1)	Tests the return value from the **Blank** function, which always returns a *blank* value. Because the first argument is *blank*, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**1**
Coalesce("", "2")	Tests the first argument which is an empty string. Because the first argument is an empty string, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**2**
Coalesce(Blank(), "", Blank(), "", "3", "4")	**Coalesce** starts at the beginning of the argument list and evaluates each argument in turn until a non-*blank* value and non-empty string is found. In this case, the first four arguments all return *blank* or an empty string, so evaluation continues to the fifth argument. The fifth argument is non-*blank* and non-empty string, so evaluation stops here. The value of the fifth argument is returned, and the sixth argument isn't evaluated.	**3**
Coalesce("")	Tests the first argument which is an empty string. Because the first argument is an empty string, and there are no more arguments, the function returns *blank*.	*blank*

IsBlank

1. Create an app from scratch, add a text-input control, and name it **FirstName**.
2. Add a label, and set its **Text** property to this formula:

   ```
   Power Fx
   ```

   ```
   If( IsBlank( FirstName.Text ), "First Name is a required field." )
   ```

   By default, the **Text** property of a text-input control is set to **"Text input"**. Because the property contains a value, it isn't blank, and the label doesn't display any message.
3. Remove all the characters from the text-input control, including any spaces.

   Because the **Text** property no longer contains any characters, it's an empty string, and **IsBlank(FirstName.Text)** will be **true**. The required field message is displayed.

For information about how to perform validation by using other tools, see the **Validate** function and working with data sources.

Other examples:

Expand table

Formula	Description	Result
IsBlank(Blank())	Tests the return value from the **Blank** function, which always returns a *blank* value.	**true**
IsBlank("")	A string that contains no characters.	**true**
IsBlank("Hello")	A string that contains one or more characters.	**false**
IsBlank(*AnyCollection*)	Because the collection exists, it isn't blank, even if it doesn't contain any records. To check for an empty collection, use **IsEmpty** instead.	**false**
IsBlank(Mid("Hello", 17, 2))	The starting character for **Mid** is beyond the end of the string. The result is an empty string.	**true**
IsBlank(If(false, false))	An **If** function with no *ElseResult*. Because the condition is always **false**, this **If** always returns *blank*.	**true**

IsEmpty

1. Create an app from scratch, and add a **Button** control.
2. Set the button's **OnSelect** property to this formula:

   Collect(IceCream, { Flavor: "Strawberry", Quantity: 300 }, { Flavor: "Chocolate", Quantity: 100 })
3. Preview your app, click or tap the button that you added, and then close Preview.

   A collection named **IceCream** is created and contains this data:

Flavor	Quantity
"Strawberry"	300
"Chocolate"	100

   This collection has two records and isn't empty. **IsEmpty(IceCream)** returns **false**, and **CountRows(IceCream)** returns **2**.
4. Add a second button, and set its **OnSelect** property to this formula:

   Clear(IceCream)
5. Preview your app, click or tap the second button, and then close Preview.

The collection is now empty:

Flavor	Quantity

The Clear function removes all the records from a collection, resulting in an empty collection. **IsEmpty(IceCream)** returns **true**, and **CountRows(IceCream)** returns **0**.

You can also use **IsEmpty** to test whether a calculated table is empty, as these examples show:

Expand table

Formula	Description	Result
IsEmpty([1, 2, 3])	The single-column table contains three records and, therefore, isn't empty.	**false**
IsEmpty([ ])	The single-column table contains no records and is empty.	**true**
IsEmpty(Filter([1, 2, 3], Value > 5))	The single-column table contains no values that are greater than 5. The result from the filter doesn't contain any records and is empty.	**true**

Collect, Clear, and ClearCollect functions

Article • 04/11/2024

Creates and clears collections and adds records to any data source.

Collect

Applies to: ✅ Canvas apps ✅ Cards ✅ Dataverse low-code plug-ins ✅ Power Platform CLI ✅ Desktop flows

The **Collect** function adds records to a data source. The items to be added can be:

- A single value: The value is placed in the **Value** field of a new record. All other properties are left blank.
- A record: Each named property is placed in the corresponding property of a new record. All other properties are left blank.
- A table: Each record of the table is added as a separate record of the data source as described above. The table isn't added as a nested table to a record. To do this, wrap the table in a record first.

When used with a collection, additional columns will be created as needed. The columns for other data sources are fixed by the data source and new columns can't be added.

If the data source doesn't already exist, a collection is created.

Collections are sometimes used to hold global variables or make a temporary copy of a data source. Canvas apps are based on formulas that automatically recalculate as the user interacts with an app. Collections don't enjoy this benefit and their use can make your app harder to create and understand. Before using a collection in this manner, review working with variables.

You can also use the **Patch** function to create records in a data source.

Collect returns the modified data source as a table. **Collect** can only be used in a behavior formula.

Clear

Applies to: ✅ Canvas apps

The **Clear** function deletes all the records of a collection. The columns of the collection will remain.

Note that **Clear** only operates on collections and not other data sources. You can use **RemoveIf(*DataSource*, true)** for this purpose. Use caution as this will remove all records from the data source's storage and can affect other users.

You can use the **Remove** function to selectively remove records.

Clear has no return value. It can only be used in a behavior formula.

ClearCollect

Applies to: Canvas apps Model-driven apps

The **ClearCollect** function deletes all the records from a collection. And then adds a different set of records to the same collection. With a single function, **ClearCollect** offers the combination of **Clear** and then **Collect**.

ClearCollect returns the modified collection as a table. **ClearCollect** can only be used in a behavior formula.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Collect(*DataSource*, *Item*, ...)

- *DataSource* – Required. The data source that you want to add data to. If it doesn't already exist, a new collection is created.
- *Item(s)* - Required. One or more records or tables to add to the data source.

Clear(*Collection*)

- *Collection* – Required. The collection that you want to clear.

ClearCollect(*Collection*, *Item*, ...)

- *Collection* – Required. The collection that you want to clear and then add data to.
- *Item(s)* - Required. One or more records or tables to add to the data source.

Examples

Clearing and adding records to a data source

In these examples, you'll erase and add to a collection that's named **IceCream**. The data source begins with these contents:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200

Expand table

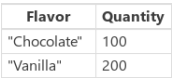
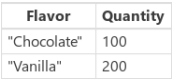
Formula	Description	Result
ClearCollect(IceCream, { Flavor: "Strawberry", Quantity: 300 })	Clears all data from the **IceCream** collection and then adds a record that includes a quantity of strawberry ice cream.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Strawberry"</td><td>300</td></tr></table> The **IceCream** collection has also been modified.
Collect(IceCream, { Flavor: "Pistachio", Quantity: 40 }, { Flavor: "Orange", Quantity: 200 })	Adds two records to the **IceCream** collection that includes a quantity of pistachio and orange ice cream.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Strawberry"</td><td>300</td></tr><tr><td>"Pistachio"</td><td>40</td></tr><tr><td>"Orange"</td><td>200</td></tr></table> The **IceCream** collection has also been modified.
Clear(IceCream)	Removes all records from the **IceCream** collection.	<table><tr><th>Flavor</th><th>Quantity</th></tr></table> The **IceCream** collection has also been modified.

For step-by-step examples of how to create a collection, see Create and update a collection.

Records and tables

These examples examine how record and table arguments to **Collect** and **ClearCollect** are handled.

Expand table

Formula	Description	Result
ClearCollect(IceCream, { Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 })	Clear all data and then adds two records to the **IceCream** collection that includes a quantity of chocolate and vanilla ice cream. The records to be added are provided as individual arguments to the function.	 The **IceCream** collection has also been modified.
ClearCollect(IceCream, Table({ Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 }))	Same as the previous example except that the records are combined in a table and passed in through a single argument. The contents of the table are extracted record by record before being added to the **IceCream** collection.	 The **IceCream** collection has also been modified.
ClearCollect(IceCream, { MyFavorites: Table({ Flavor: "Chocolate", Quantity: 100 }, { Flavor: "Vanilla", Quantity: 200 }) })	Same as the previous example except that the table is wrapped in a record. The records of the table aren't extracted and instead the entire table is added as a cell of the record.	 The **IceCream** collection has also been modified.

Color enumeration and ColorFade, ColorValue, and RGBA functions

06/19/2025

Expand table

Functions	Applies to
Color	✅ Model-driven apps
ColorFade **ColorValue**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
RGBA	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Use built-in color values, define custom colors, and use the alpha channel to control transparency.

> ⓘ Note
>
> PAC CLI pac power-fx commands don't support the **Color** enumeration.

Description

Use the **Color** enumeration to access colors defined by HTML's Cascading Style Sheets (CSS). For example, **Color.Red** returns pure red. A list of these colors appears at the end of this article.

The **ColorValue** function returns a color based on a CSS color string. The string can be in any of these forms:

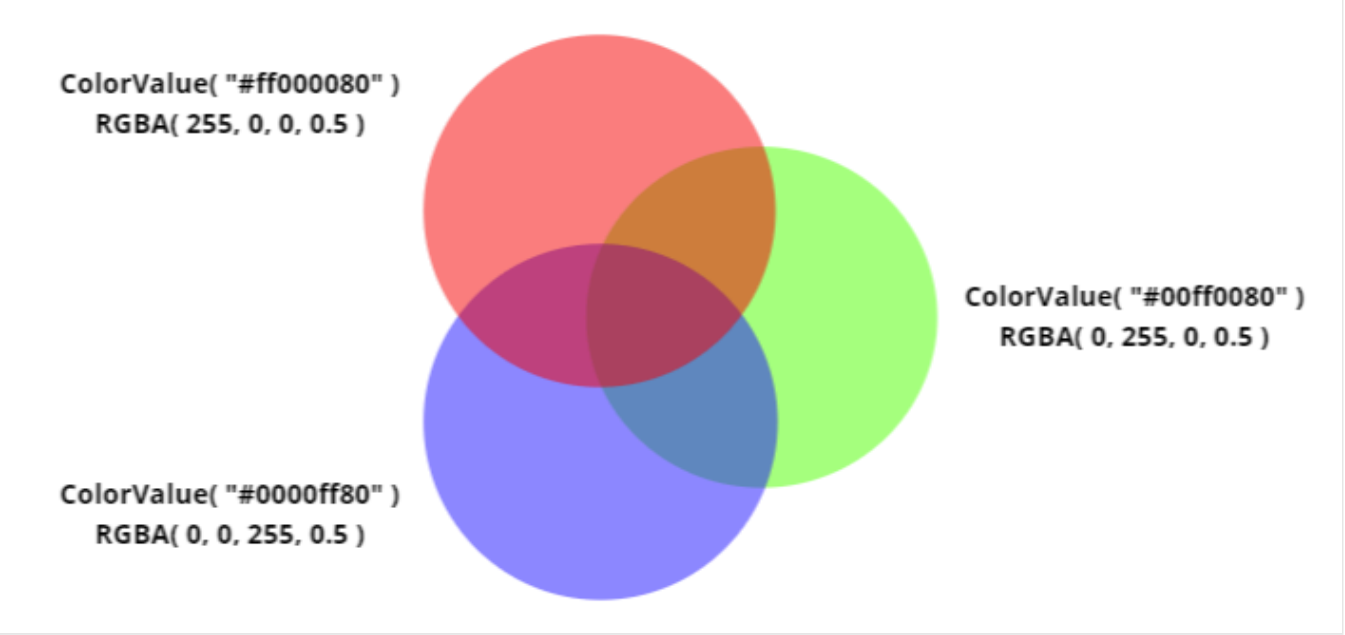
- **CSS color name:** **"RoxyBrown"** and **"OliveDrab"** are examples. These names don't include spaces. The list of supported colors appears later in this article.
- **6-digit hex value:** For example, **"#ffd700"** is the same as **"Gold"**. The string uses the format "*#rrggbb*", where *rr* is the red portion in two hexadecimal digits, *gg* is the green, and *bb* is the blue.
- **8-digit hex value:** For example, **"#ff7f5080"** is the same as **"Coral"** with a 50% alpha channel. The string uses the format "*#rrggbbaa*", where *rr*, *gg*, and *bb* are identical to the 6-digit form. The alpha channel is represented by *aa*: **00** means fully transparent, and **ff** means fully opaque.

The **RGBA** function returns a color based on red, green, and blue components. The function also includes an alpha channel for mixing colors of controls layered in front of each other. The alpha channel ranges from 0% or 0% (fully transparent and invisible) to 1% or 100% (fully opaque and completely blocks any layers behind a control).

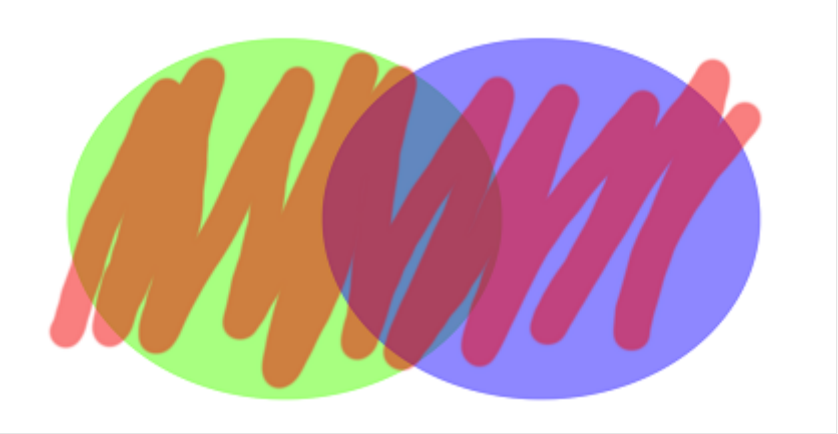
The **ColorFade** function returns a brighter or darker version of a color. The amount of fade ranges from -1 (fully darkens a color to black), to 0 (doesn't affect the color), to 1 (fully brightens a color to white).

Alpha channel

In a canvas app, you layer controls in front of one another and set the transparency of a control to any controls behind it. As a result, colors blend through the layers. For example, this diagram shows how the three primary colors mix with an alpha setting of 50 percent:



You can also blend images in file formats that support alpha channels. For example, you can't blend .jpeg files, but you can blend .png files. The next graphic shows the same red, green, and blue colors from the previous example, but the red color appears as a squiggle (instead of a circle) in a .png file with a 50 percent alpha channel:



If you specify a **Color** enumeration value or build a **ColorValue** formula with a color name or a 6-digit hexadecimal value, the alpha setting is 100 percent, which is fully opaque.

Syntax

Color.*ColorName*

- *ColorName* - Required. A Cascading Style Sheet (CSS) color name. The list of possible enumeration values is at the end of this article.

ColorValue(*CSSColor*)

- *CSSColor* - Required. A Cascading Style Sheet (CSS) color definition. S name, OliveDralike a hex value, like **#6b8e23** olikeffd420**. Hex values can use either #_rrggbb__usgormat. format

ColorValue(*Dynamic*)

- *Dynamic* - Required. A **Dynamic** value with a string that represents a Cascading Style Sheet (CSS) color definition.

RGBA(*Red*, *Green*, *Blue*, *Alpha*)

- *Red*, *Green*, *Blue* - Required. Color component values that range from 0 (no saturation) to 255 (full saturation).
- *Alpha* - Required. Alpha component that ranges from 0 (fully transparent) to 1 (fully opaque). You can also use a percentage, 0% to 100%.

ColorFade(*Color*, *FadeAmount*)

- *Color* - Required. A color value like **Color.Red** or the output from **ColorValue** or **RGBA**.
- *FadeAmount* - Required. A number between -1 and 1. -1 fully darkens a color to black, 0 doesn't affect the color, and 1 fully brightens a color to white. You can also use a percentage, from -100% to 100%.

Built-in colors

Expand table

Color enumeration	ColorValue	RGBA	Color Swatch
Color.AliceBlue	ColorValue("#f0f8ff") ColorValue("aliceblue")	RGBA(240, 248, 255, 1)	
Color.AntiqueWhite	ColorValue("#faebd7") ColorValue("AntiqueWhite")	RGBA(250, 235, 215, 1)	
Color.Aqua	ColorValue("#00ffff") ColorValue("AQUA")	RGBA(0, 255, 255, 1)	
Color.Aquamarine	ColorValue("#7fffd4") ColorValue("Aquamarine")	RGBA(127, 255, 212, 1)	
Color.Azure	ColorValue("#f0ffff") ColorValue("azure")	RGBA(240, 255, 255, 1)	
Color.Beige	ColorValue("#f5f5dc") ColorValue("Beige")	RGBA(245, 245, 220, 1)	
Color.Bisque	ColorValue("#ffe4c4") ColorValue("BISQUE")	RGBA(255, 228, 196, 1)	
Color.Black	ColorValue("#000000") ColorValue("Black")	RGBA(0, 0, 0, 1)	
Color.BlanchedAlmond	ColorValue("#ffebcd") ColorValue("blanchedalmond")	RGBA(255, 235, 205, 1)	
Color.Blue	ColorValue("#0000ff") ColorValue("Blue")	RGBA(0, 0, 255, 1)	
Color.BlueViolet	ColorValue("#8a2be2") ColorValue("BLUEVIOLET")	RGBA(138, 43, 226, 1)	
Color.Brown	ColorValue("#a52a2a") ColorValue("Brown")	RGBA(165, 42, 42, 1)	
Color.Burlywood	ColorValue("#deb887") ColorValue("burlywood")	RGBA(222, 184, 135, 1)	
Color.CadetBlue	ColorValue("#5f9ea0") ColorValue("CadetBlue")	RGBA(95, 158, 160, 1)	
Color.Chartreuse	ColorValue("#7fff00") ColorValue("CHARTREUSE")	RGBA(127, 255, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Chocolate	ColorValue("#d2691e") ColorValue("Chocolate")	RGBA(210, 105, 30, 1)	
Color.Coral	ColorValue("#ff7f50") ColorValue("coral")	RGBA(255, 127, 80, 1)	
Color.CornflowerBlue	ColorValue("#6495ed") ColorValue("CornflowerBlue")	RGBA(100, 149, 237, 1)	
Color.Cornsilk	ColorValue("#fff8dc") ColorValue("CORNSILK")	RGBA(255, 248, 220, 1)	
Color.Crimson	ColorValue("#dc143c") ColorValue("Crimson")	RGBA(220, 20, 60, 1)	
Color.Cyan	ColorValue("#00ffff") ColorValue("cyan")	RGBA(0, 255, 255, 1)	
Color.DarkBlue	ColorValue("#00008b") ColorValue("DarkBlue")	RGBA(0, 0, 139, 1)	
Color.DarkCyan	ColorValue("#008b8b") ColorValue("DARKCYAN")	RGBA(0, 139, 139, 1)	
Color.DarkGoldenRod	ColorValue("#b8860b") ColorValue("DarkGoldenRod")	RGBA(184, 134, 11, 1)	
Color.DarkGray	ColorValue("#a9a9a9") ColorValue("darkgray")	RGBA(169, 169, 169, 1)	
Color.DarkGreen	ColorValue("#006400") ColorValue("DarkGreen")	RGBA(0, 100, 0, 1)	
Color.DarkGrey	ColorValue("#a9a9a9") ColorValue("DARKGREY")	RGBA(169, 169, 169, 1)	
Color.DarkKhaki	ColorValue("#bdb76b") ColorValue("DarkKhaki")	RGBA(189, 183, 107, 1)	
Color.DarkMagenta	ColorValue("#8b008b") ColorValue("darkmagenta")	RGBA(139, 0, 139, 1)	
Color.DarkOliveGreen	ColorValue("#556b2f") ColorValue("DarkOliveGreen")	RGBA(85, 107, 47, 1)	
Color.DarkOrange	ColorValue("#ff8c00") ColorValue(	RGBA(255, 140, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"DARKORANGE")		
Color.DarkOrchid	ColorValue("#9932cc") ColorValue("DarkOrchid")	RGBA(153, 50, 204, 1)	
Color.DarkRed	ColorValue("#8b0000") ColorValue("darkred")	RGBA(139, 0, 0, 1)	
Color.DarkSalmon	ColorValue("#e9967a") ColorValue("DarkSalmon")	RGBA(233, 150, 122, 1)	
Color.DarkSeaGreen	ColorValue("#8fbc8f") ColorValue("DARKSEAGREEN")	RGBA(143, 188, 143, 1)	
Color.DarkSlateBlue	ColorValue("#483d8b") ColorValue("DarkSlateBlue")	RGBA(72, 61, 139, 1)	
Color.DarkSlateGray	ColorValue("#2f4f4f") ColorValue("darkslategray")	RGBA(47, 79, 79, 1)	
Color.DarkSlateGrey	ColorValue("#2f4f4f") ColorValue("DarkSlateGrey")	RGBA(47, 79, 79, 1)	
Color.DarkTurquoise	ColorValue("#00ced1") ColorValue("DARKTURQUOISE")	RGBA(0, 206, 209, 1)	
Color.DarkViolet	ColorValue("#9400d3") ColorValue("DarkViolet")	RGBA(148, 0, 211, 1)	
Color.DeepPink	ColorValue("#ff1493") ColorValue("deeppink")	RGBA(255, 20, 147, 1)	
Color.DeepSkyBlue	ColorValue("#00bfff") ColorValue("DeepSkyBlue")	RGBA(0, 191, 255, 1)	
Color.DimGray	ColorValue("#696969") ColorValue("DIMGRAY")	RGBA(105, 105, 105, 1)	
Color.DimGrey	ColorValue("#696969") ColorValue("DimGrey")	RGBA(105, 105, 105, 1)	
Color.DodgerBlue	ColorValue("#1e90ff") ColorValue("dodgerblue")	RGBA(30, 144, 255, 1)	
Color.FireBrick	ColorValue("#b22222") ColorValue("FireBrick")	RGBA(178, 34, 34, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.FloralWhite	ColorValue("#fffaf0") ColorValue("FLORALWHITE")	RGBA(255, 250, 240, 1)	
Color.ForestGreen	ColorValue("#228b22") ColorValue("ForestGreen")	RGBA(34, 139, 34, 1)	
Color.Fuchsia	ColorValue("#ff00ff") ColorValue("fuchsia")	RGBA(255, 0, 255, 1)	
Color.Gainsboro	ColorValue("#dcdcdc") ColorValue("Gainsboro")	RGBA(220, 220, 220, 1)	
Color.GhostWhite	ColorValue("#f8f8ff") ColorValue("GHOSTWHITE")	RGBA(248, 248, 255, 1)	
Color.Gold	ColorValue("#ffd700") ColorValue("Gold")	RGBA(255, 215, 0, 1)	
Color.GoldenRod	ColorValue("#daa520") ColorValue("goldenrod")	RGBA(218, 165, 32, 1)	
Color.Gray	ColorValue("#808080") ColorValue("Gray")	RGBA(128, 128, 128, 1)	
Color.Green	ColorValue("#008000") ColorValue("GREEN")	RGBA(0, 128, 0, 1)	
Color.GreenYellow	ColorValue("#adff2f") ColorValue("GreenYellow")	RGBA(173, 255, 47, 1)	
Color.Grey	ColorValue("#808080") ColorValue("grey")	RGBA(128, 128, 128, 1)	
Color.Honeydew	ColorValue("#f0fff0") ColorValue("Honeydew")	RGBA(240, 255, 240, 1)	
Color.HotPink	ColorValue("#ff69b4") ColorValue("HOTPINK")	RGBA(255, 105, 180, 1)	
Color.IndianRed	ColorValue("#cd5c5c") ColorValue("IndianRed")	RGBA(205, 92, 92, 1)	
Color.Indigo	ColorValue("#4b0082") ColorValue("indigo")	RGBA(75, 0, 130, 1)	
Color.Ivory	ColorValue("#fffff0") ColorValue("Ivory")	RGBA(255, 255, 240, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Khaki	ColorValue("#f0e68c") ColorValue("KHAKI")	RGBA(240, 230, 140, 1)	
Color.Lavender	ColorValue("#e6e6fa") ColorValue("Lavender")	RGBA(230, 230, 250, 1)	
Color.LavenderBlush	ColorValue("#fff0f5") ColorValue("lavenderblush")	RGBA(255, 240, 245, 1)	
Color.LawnGreen	ColorValue("#7cfc00") ColorValue("LawnGreen")	RGBA(124, 252, 0, 1)	
Color.LemonChiffon	ColorValue("#fffacd") ColorValue("LEMONCHIFFON")	RGBA(255, 250, 205, 1)	
Color.LightBlue	ColorValue("#add8e6") ColorValue("LightBlue")	RGBA(173, 216, 230, 1)	
Color.LightCoral	ColorValue("#f08080") ColorValue("lightcoral")	RGBA(240, 128, 128, 1)	
Color.LightCyan	ColorValue("#e0ffff") ColorValue("LightCyan")	RGBA(224, 255, 255, 1)	
Color.LightGoldenRodYellow	ColorValue("#fafad2") ColorValue("lightgoldenrodyellow")	RGBA(250, 250, 210, 1)	
Color.LightGray	ColorValue("#d3d3d3") ColorValue("LightGray")	RGBA(211, 211, 211, 1)	
Color.LightGreen	ColorValue("#90ee90") ColorValue("lightgreen")	RGBA(144, 238, 144, 1)	
Color.LightGrey	ColorValue("#d3d3d3") ColorValue("LightGrey")	RGBA(211, 211, 211, 1)	
Color.LightPink	ColorValue("#ffb6c1") ColorValue("LIGHTPINK")	RGBA(255, 182, 193, 1)	
Color.LightSalmon	ColorValue("#ffa07a") ColorValue("LightSalmon")	RGBA(255, 160, 122, 1)	
Color.LightSeaGreen	ColorValue("#20b2aa") ColorValue("lightseagreen")	RGBA(32, 178, 170, 1)	
Color.LightSkyBlue	ColorValue("#87cefa") ColorValue("LightSkyBlue")	RGBA(135, 206, 250, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.LightSlateGray	ColorValue("#778899") ColorValue("LIGHTSLATEGRAY")	RGBA(119, 136, 153, 1)	
Color.LightSlateGrey	ColorValue("#778899") ColorValue("LightSlateGrey")	RGBA(119, 136, 153, 1)	
Color.LightSteelBlue	ColorValue("#b0c4de") ColorValue("lightsteelblue")	RGBA(176, 196, 222, 1)	
Color.LightYellow	ColorValue("#ffffe0") ColorValue("LightYellow")	RGBA(255, 255, 224, 1)	
Color.Lime	ColorValue("#00ff00") ColorValue("LIME")	RGBA(0, 255, 0, 1)	
Color.LimeGreen	ColorValue("#32cd32") ColorValue("LimeGreen")	RGBA(50, 205, 50, 1)	
Color.Linen	ColorValue("#faf0e6") ColorValue("linen")	RGBA(250, 240, 230, 1)	
Color.Magenta	ColorValue("#ff00ff") ColorValue("Magenta")	RGBA(255, 0, 255, 1)	
Color.Maroon	ColorValue("#800000") ColorValue("MAROON")	RGBA(128, 0, 0, 1)	
Color.MediumAquamarine	ColorValue("#66cdaa") ColorValue("MediumAquamarine")	RGBA(102, 205, 170, 1)	
Color.MediumBlue	ColorValue("#0000cd") ColorValue("mediumblue")	RGBA(0, 0, 205, 1)	
Color.MediumOrchid	ColorValue("#ba55d3") ColorValue("MediumOrchid")	RGBA(186, 85, 211, 1)	
Color.MediumPurple	ColorValue("#9370db") ColorValue("MEDIUMPURPLE")	RGBA(147, 112, 219, 1)	
Color.MediumSeaGreen	ColorValue("#3cb371") ColorValue("MediumSeaGreen")	RGBA(60, 179, 113, 1)	
Color.MediumSlateBlue	ColorValue("#7b68ee") ColorValue(	RGBA(123, 104, 238, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"mediumslateblue")		
Color.MediumSpringGreen	ColorValue("#00fa9a") ColorValue("MediumSpringGreen")	RGBA(0, 250, 154, 1)	
Color.MediumTurquoise	ColorValue("#48d1cc") ColorValue("MEDIUMTURQUOISE")	RGBA(72, 209, 204, 1)	
Color.MediumVioletRed	ColorValue("#c71585") ColorValue("MediumVioletRed")	RGBA(199, 21, 133, 1)	
Color.MidnightBlue	ColorValue("#191970") ColorValue("midnightblue")	RGBA(25, 25, 112, 1)	
Color.MintCream	ColorValue("#f5fffa") ColorValue("MintCream")	RGBA(245, 255, 250, 1)	
Color.MistyRose	ColorValue("#ffe4e1") ColorValue("MISTYROSE")	RGBA(255, 228, 225, 1)	
Color.Moccasin	ColorValue("#ffe4b5") ColorValue("Moccasin")	RGBA(255, 228, 181, 1)	
Color.NavajoWhite	ColorValue("#ffdead") ColorValue("navajowhite")	RGBA(255, 222, 173, 1)	
Color.Navy	ColorValue("#000080") ColorValue("Navy")	RGBA(0, 0, 128, 1)	
Color.OldLace	ColorValue("#fdf5e6") ColorValue("OLDLACE")	RGBA(253, 245, 230, 1)	
Color.Olive	ColorValue("#808000") ColorValue("Olive")	RGBA(128, 128, 0, 1)	
Color.OliveDrab	ColorValue("#6b8e23") ColorValue("olivedrab")	RGBA(107, 142, 35, 1)	
Color.Orange	ColorValue("#ffa500") ColorValue("Orange")	RGBA(255, 165, 0, 1)	
Color.OrangeRed	ColorValue("#ff4500") ColorValue("ORANGERED")	RGBA(255, 69, 0, 1)	
Color.Orchid	ColorValue("#da70d6") ColorValue("Orchid")	RGBA(218, 112, 214, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.PaleGoldenRod	ColorValue("#eee8aa") ColorValue("palegoldenrod")	RGBA(238, 232, 170, 1)	
Color.PaleGreen	ColorValue("#98fb98") ColorValue("PaleGreen")	RGBA(152, 251, 152, 1)	
Color.PaleTurquoise	ColorValue("#afeeee") ColorValue("PALETURQUOISE")	RGBA(175, 238, 238, 1)	
Color.PaleVioletRed	ColorValue("#db7093") ColorValue("PaleVioletRed")	RGBA(219, 112, 147, 1)	
Color.PapayaWhip	ColorValue("#ffefd5") ColorValue("papayawhip")	RGBA(255, 239, 213, 1)	
Color.PeachPuff	ColorValue("#ffdab9") ColorValue("PeachPuff")	RGBA(255, 218, 185, 1)	
Color.Peru	ColorValue("#cd853f") ColorValue("PERU")	RGBA(205, 133, 63, 1)	
Color.Pink	ColorValue("#ffc0cb") ColorValue("Pink")	RGBA(255, 192, 203, 1)	
Color.Plum	ColorValue("#dda0dd") ColorValue("plum")	RGBA(221, 160, 221, 1)	
Color.PowderBlue	ColorValue("#b0e0e6") ColorValue("PowderBlue")	RGBA(176, 224, 230, 1)	
Color.Purple	ColorValue("#800080") ColorValue("PURPLE")	RGBA(128, 0, 128, 1)	
Color.Red	ColorValue("#ff0000") ColorValue("Red")	RGBA(255, 0, 0, 1)	
Color.RosyBrown	ColorValue("#bc8f8f") ColorValue("rosybrown")	RGBA(188, 143, 143, 1)	
Color.RoyalBlue	ColorValue("#4169e1") ColorValue("RoyalBlue")	RGBA(65, 105, 225, 1)	
Color.SaddleBrown	ColorValue("#8b4513") ColorValue("SADDLEBROWN")	RGBA(139, 69, 19, 1)	
Color.Salmon	ColorValue("#fa8072") ColorValue("Salmon")	RGBA(250, 128, 114, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.SandyBrown	ColorValue("#f4a460") ColorValue("sandybrown")	RGBA(244, 164, 96, 1)	
Color.SeaGreen	ColorValue("#2e8b57") ColorValue("SeaGreen")	RGBA(46, 139, 87, 1)	
Color.SeaShell	ColorValue("#fff5ee") ColorValue("SEASHELL")	RGBA(255, 245, 238, 1)	
Color.Sienna	ColorValue("#a0522d") ColorValue("Sienna")	RGBA(160, 82, 45, 1)	
Color.Silver	ColorValue("#c0c0c0") ColorValue("silver")	RGBA(192, 192, 192, 1)	
Color.SkyBlue	ColorValue("#87ceeb") ColorValue("SkyBlue")	RGBA(135, 206, 235, 1)	
Color.SlateBlue	ColorValue("#6a5acd") ColorValue("SLATEBLUE")	RGBA(106, 90, 205, 1)	
Color.SlateGray	ColorValue("#708090") ColorValue("SlateGray")	RGBA(112, 128, 144, 1)	
Color.SlateGrey	ColorValue("#708090") ColorValue("slategrey")	RGBA(112, 128, 144, 1)	
Color.Snow	ColorValue("#fffafa") ColorValue("Snow")	RGBA(255, 250, 250, 1)	
Color.SpringGreen	ColorValue("#00ff7f") ColorValue("SPRINGGREEN")	RGBA(0, 255, 127, 1)	
Color.SteelBlue	ColorValue("#4682b4") ColorValue("SteelBlue")	RGBA(70, 130, 180, 1)	
Color.Tan	ColorValue("#d2b48c") ColorValue("tan")	RGBA(210, 180, 140, 1)	
Color.Teal	ColorValue("#008080") ColorValue("Teal")	RGBA(0, 128, 128, 1)	
Color.Thistle	ColorValue("#d8bfd8") ColorValue("THISTLE")	RGBA(216, 191, 216, 1)	
Color.Tomato	ColorValue("#ff6347") ColorValue("Tomato")	RGBA(255, 99, 71, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Transparent	ColorValue("#00000000") ColorValue("Transparent")	RGBA(0, 0, 0, 0)	
Color.Turquoise	ColorValue("#40e0d0") ColorValue("turquoise")	RGBA(64, 224, 208, 1)	
Color.Violet	ColorValue("#ee82ee") ColorValue("Violet")	RGBA(238, 130, 238, 1)	
Color.Wheat	ColorValue("#f5deb3") ColorValue("WHEAT")	RGBA(245, 222, 179, 1)	
Color.White	ColorValue("#ffffff") ColorValue("White")	RGBA(255, 255, 255, 1)	
Color.WhiteSmoke	ColorValue("#f5f5f5") ColorValue("whitesmoke")	RGBA(245, 245, 245, 1)	
Color.Yellow	ColorValue("#ffff00") ColorValue("Yellow")	RGBA(255, 255, 0, 1)	
Color.YellowGreen	ColorValue("#9acd32") ColorValue("YELLOWGREEN")	RGBA(154, 205, 50, 1)	

Color enumeration and ColorFade, ColorValue, and RGBA functions

06/19/2025

Expand table

Functions	Applies to
Color	✅ Model-driven apps
ColorFade **ColorValue**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
RGBA	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Use built-in color values, define custom colors, and use the alpha channel to control transparency.

> ⓘ Note
>
> PAC CLI pac power-fx commands don't support the **Color** enumeration.

Description

Use the **Color** enumeration to access colors defined by HTML's Cascading Style Sheets (CSS). For example, **Color.Red** returns pure red. A list of these colors appears at the end of this article.

The **ColorValue** function returns a color based on a CSS color string. The string can be in any of these forms:

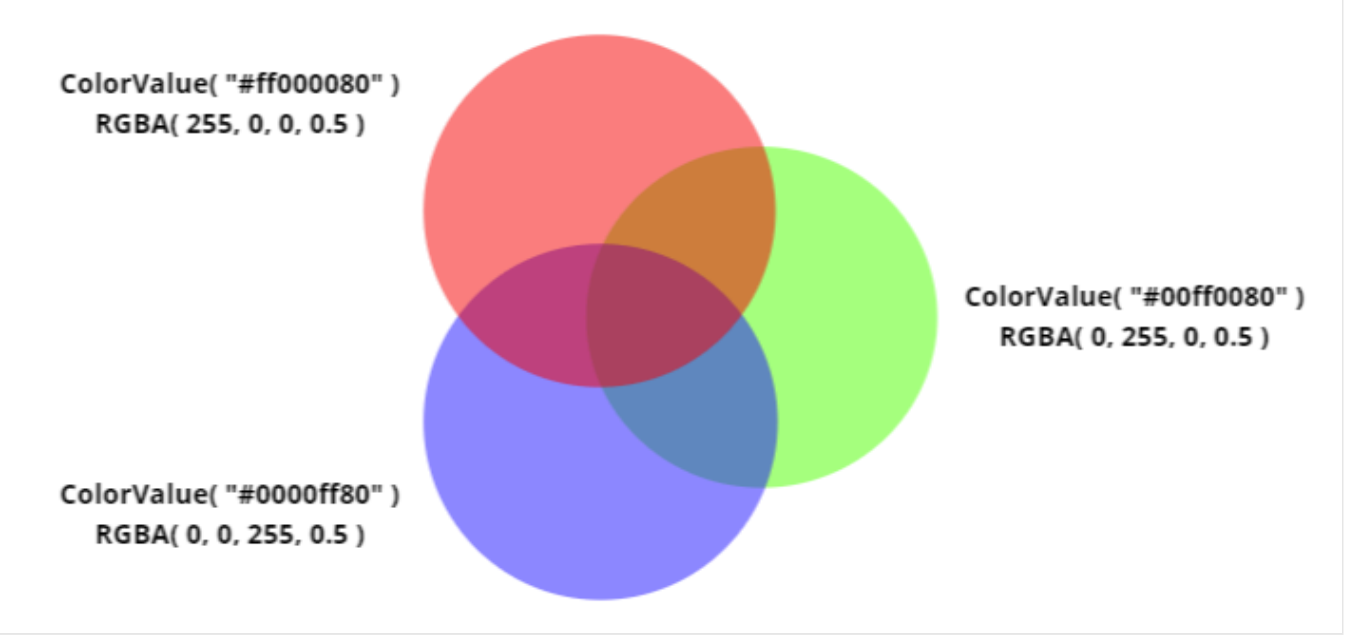
- **CSS color name:** **"RoxyBrown"** and **"OliveDrab"** are examples. These names don't include spaces. The list of supported colors appears later in this article.
- **6-digit hex value:** For example, **"#ffd700"** is the same as **"Gold"**. The string uses the format "*#rrggbb*", where *rr* is the red portion in two hexadecimal digits, *gg* is the green, and *bb* is the blue.
- **8-digit hex value:** For example, **"#ff7f5080"** is the same as **"Coral"** with a 50% alpha channel. The string uses the format "*#rrggbbaa*", where *rr*, *gg*, and *bb* are identical to the 6-digit form. The alpha channel is represented by *aa*: **00** means fully transparent, and **ff** means fully opaque.

The **RGBA** function returns a color based on red, green, and blue components. The function also includes an alpha channel for mixing colors of controls layered in front of each other. The alpha channel ranges from 0% or 0% (fully transparent and invisible) to 1% or 100% (fully opaque and completely blocks any layers behind a control).

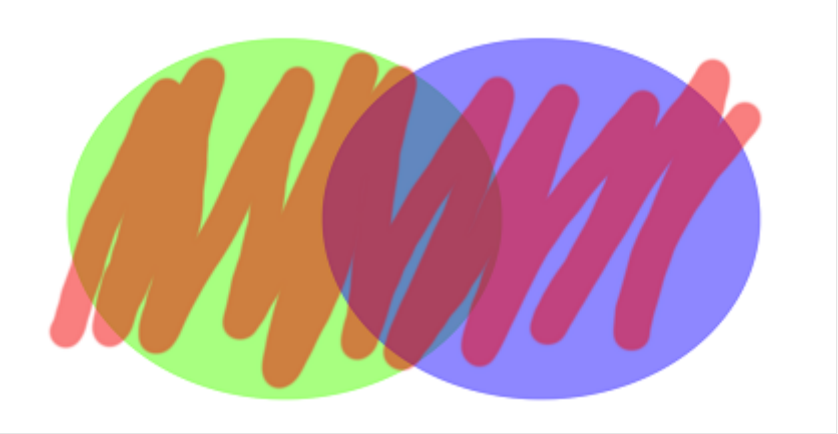
The **ColorFade** function returns a brighter or darker version of a color. The amount of fade ranges from -1 (fully darkens a color to black), to 0 (doesn't affect the color), to 1 (fully brightens a color to white).

Alpha channel

In a canvas app, you layer controls in front of one another and set the transparency of a control to any controls behind it. As a result, colors blend through the layers. For example, this diagram shows how the three primary colors mix with an alpha setting of 50 percent:



You can also blend images in file formats that support alpha channels. For example, you can't blend .jpeg files, but you can blend .png files. The next graphic shows the same red, green, and blue colors from the previous example, but the red color appears as a squiggle (instead of a circle) in a .png file with a 50 percent alpha channel:



If you specify a **Color** enumeration value or build a **ColorValue** formula with a color name or a 6-digit hexadecimal value, the alpha setting is 100 percent, which is fully opaque.

Syntax

Color.*ColorName*

- *ColorName* - Required. A Cascading Style Sheet (CSS) color name. The list of possible enumeration values is at the end of this article.

ColorValue(*CSSColor*)

- *CSSColor* - Required. A Cascading Style Sheet (CSS) color definition. S name, OliveDralike a hex value, like **#6b8e23** olikeffd420**. Hex values can use either #_rrggbb__usgormat. format

ColorValue(*Dynamic*)

- *Dynamic* - Required. A **Dynamic** value with a string that represents a Cascading Style Sheet (CSS) color definition.

RGBA(*Red*, *Green*, *Blue*, *Alpha*)

- *Red*, *Green*, *Blue* - Required. Color component values that range from 0 (no saturation) to 255 (full saturation).
- *Alpha* - Required. Alpha component that ranges from 0 (fully transparent) to 1 (fully opaque). You can also use a percentage, 0% to 100%.

ColorFade(*Color*, *FadeAmount*)

- *Color* - Required. A color value like **Color.Red** or the output from **ColorValue** or **RGBA**.
- *FadeAmount* - Required. A number between -1 and 1. -1 fully darkens a color to black, 0 doesn't affect the color, and 1 fully brightens a color to white. You can also use a percentage, from -100% to 100%.

Built-in colors

Expand table

Color enumeration	ColorValue	RGBA	Color Swatch
Color.AliceBlue	ColorValue("#f0f8ff") ColorValue("aliceblue")	RGBA(240, 248, 255, 1)	
Color.AntiqueWhite	ColorValue("#faebd7") ColorValue("AntiqueWhite")	RGBA(250, 235, 215, 1)	
Color.Aqua	ColorValue("#00ffff") ColorValue("AQUA")	RGBA(0, 255, 255, 1)	
Color.Aquamarine	ColorValue("#7fffd4") ColorValue("Aquamarine")	RGBA(127, 255, 212, 1)	
Color.Azure	ColorValue("#f0ffff") ColorValue("azure")	RGBA(240, 255, 255, 1)	
Color.Beige	ColorValue("#f5f5dc") ColorValue("Beige")	RGBA(245, 245, 220, 1)	
Color.Bisque	ColorValue("#ffe4c4") ColorValue("BISQUE")	RGBA(255, 228, 196, 1)	
Color.Black	ColorValue("#000000") ColorValue("Black")	RGBA(0, 0, 0, 1)	
Color.BlanchedAlmond	ColorValue("#ffebcd") ColorValue("blanchedalmond")	RGBA(255, 235, 205, 1)	
Color.Blue	ColorValue("#0000ff") ColorValue("Blue")	RGBA(0, 0, 255, 1)	
Color.BlueViolet	ColorValue("#8a2be2") ColorValue("BLUEVIOLET")	RGBA(138, 43, 226, 1)	
Color.Brown	ColorValue("#a52a2a") ColorValue("Brown")	RGBA(165, 42, 42, 1)	
Color.Burlywood	ColorValue("#deb887") ColorValue("burlywood")	RGBA(222, 184, 135, 1)	
Color.CadetBlue	ColorValue("#5f9ea0") ColorValue("CadetBlue")	RGBA(95, 158, 160, 1)	
Color.Chartreuse	ColorValue("#7fff00") ColorValue("CHARTREUSE")	RGBA(127, 255, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Chocolate	ColorValue("#d2691e") ColorValue("Chocolate")	RGBA(210, 105, 30, 1)	
Color.Coral	ColorValue("#ff7f50") ColorValue("coral")	RGBA(255, 127, 80, 1)	
Color.CornflowerBlue	ColorValue("#6495ed") ColorValue("CornflowerBlue")	RGBA(100, 149, 237, 1)	
Color.Cornsilk	ColorValue("#fff8dc") ColorValue("CORNSILK")	RGBA(255, 248, 220, 1)	
Color.Crimson	ColorValue("#dc143c") ColorValue("Crimson")	RGBA(220, 20, 60, 1)	
Color.Cyan	ColorValue("#00ffff") ColorValue("cyan")	RGBA(0, 255, 255, 1)	
Color.DarkBlue	ColorValue("#00008b") ColorValue("DarkBlue")	RGBA(0, 0, 139, 1)	
Color.DarkCyan	ColorValue("#008b8b") ColorValue("DARKCYAN")	RGBA(0, 139, 139, 1)	
Color.DarkGoldenRod	ColorValue("#b8860b") ColorValue("DarkGoldenRod")	RGBA(184, 134, 11, 1)	
Color.DarkGray	ColorValue("#a9a9a9") ColorValue("darkgray")	RGBA(169, 169, 169, 1)	
Color.DarkGreen	ColorValue("#006400") ColorValue("DarkGreen")	RGBA(0, 100, 0, 1)	
Color.DarkGrey	ColorValue("#a9a9a9") ColorValue("DARKGREY")	RGBA(169, 169, 169, 1)	
Color.DarkKhaki	ColorValue("#bdb76b") ColorValue("DarkKhaki")	RGBA(189, 183, 107, 1)	
Color.DarkMagenta	ColorValue("#8b008b") ColorValue("darkmagenta")	RGBA(139, 0, 139, 1)	
Color.DarkOliveGreen	ColorValue("#556b2f") ColorValue("DarkOliveGreen")	RGBA(85, 107, 47, 1)	
Color.DarkOrange	ColorValue("#ff8c00") ColorValue(	RGBA(255, 140, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"DARKORANGE")		
Color.DarkOrchid	ColorValue("#9932cc") ColorValue("DarkOrchid")	RGBA(153, 50, 204, 1)	
Color.DarkRed	ColorValue("#8b0000") ColorValue("darkred")	RGBA(139, 0, 0, 1)	
Color.DarkSalmon	ColorValue("#e9967a") ColorValue("DarkSalmon")	RGBA(233, 150, 122, 1)	
Color.DarkSeaGreen	ColorValue("#8fbc8f") ColorValue("DARKSEAGREEN")	RGBA(143, 188, 143, 1)	
Color.DarkSlateBlue	ColorValue("#483d8b") ColorValue("DarkSlateBlue")	RGBA(72, 61, 139, 1)	
Color.DarkSlateGray	ColorValue("#2f4f4f") ColorValue("darkslategray")	RGBA(47, 79, 79, 1)	
Color.DarkSlateGrey	ColorValue("#2f4f4f") ColorValue("DarkSlateGrey")	RGBA(47, 79, 79, 1)	
Color.DarkTurquoise	ColorValue("#00ced1") ColorValue("DARKTURQUOISE")	RGBA(0, 206, 209, 1)	
Color.DarkViolet	ColorValue("#9400d3") ColorValue("DarkViolet")	RGBA(148, 0, 211, 1)	
Color.DeepPink	ColorValue("#ff1493") ColorValue("deeppink")	RGBA(255, 20, 147, 1)	
Color.DeepSkyBlue	ColorValue("#00bfff") ColorValue("DeepSkyBlue")	RGBA(0, 191, 255, 1)	
Color.DimGray	ColorValue("#696969") ColorValue("DIMGRAY")	RGBA(105, 105, 105, 1)	
Color.DimGrey	ColorValue("#696969") ColorValue("DimGrey")	RGBA(105, 105, 105, 1)	
Color.DodgerBlue	ColorValue("#1e90ff") ColorValue("dodgerblue")	RGBA(30, 144, 255, 1)	
Color.FireBrick	ColorValue("#b22222") ColorValue("FireBrick")	RGBA(178, 34, 34, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.FloralWhite	ColorValue("#fffaf0") ColorValue("FLORALWHITE")	RGBA(255, 250, 240, 1)	
Color.ForestGreen	ColorValue("#228b22") ColorValue("ForestGreen")	RGBA(34, 139, 34, 1)	
Color.Fuchsia	ColorValue("#ff00ff") ColorValue("fuchsia")	RGBA(255, 0, 255, 1)	
Color.Gainsboro	ColorValue("#dcdcdc") ColorValue("Gainsboro")	RGBA(220, 220, 220, 1)	
Color.GhostWhite	ColorValue("#f8f8ff") ColorValue("GHOSTWHITE")	RGBA(248, 248, 255, 1)	
Color.Gold	ColorValue("#ffd700") ColorValue("Gold")	RGBA(255, 215, 0, 1)	
Color.GoldenRod	ColorValue("#daa520") ColorValue("goldenrod")	RGBA(218, 165, 32, 1)	
Color.Gray	ColorValue("#808080") ColorValue("Gray")	RGBA(128, 128, 128, 1)	
Color.Green	ColorValue("#008000") ColorValue("GREEN")	RGBA(0, 128, 0, 1)	
Color.GreenYellow	ColorValue("#adff2f") ColorValue("GreenYellow")	RGBA(173, 255, 47, 1)	
Color.Grey	ColorValue("#808080") ColorValue("grey")	RGBA(128, 128, 128, 1)	
Color.Honeydew	ColorValue("#f0fff0") ColorValue("Honeydew")	RGBA(240, 255, 240, 1)	
Color.HotPink	ColorValue("#ff69b4") ColorValue("HOTPINK")	RGBA(255, 105, 180, 1)	
Color.IndianRed	ColorValue("#cd5c5c") ColorValue("IndianRed")	RGBA(205, 92, 92, 1)	
Color.Indigo	ColorValue("#4b0082") ColorValue("indigo")	RGBA(75, 0, 130, 1)	
Color.Ivory	ColorValue("#fffff0") ColorValue("Ivory")	RGBA(255, 255, 240, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Khaki	ColorValue("#f0e68c") ColorValue("KHAKI")	RGBA(240, 230, 140, 1)	
Color.Lavender	ColorValue("#e6e6fa") ColorValue("Lavender")	RGBA(230, 230, 250, 1)	
Color.LavenderBlush	ColorValue("#fff0f5") ColorValue("lavenderblush")	RGBA(255, 240, 245, 1)	
Color.LawnGreen	ColorValue("#7cfc00") ColorValue("LawnGreen")	RGBA(124, 252, 0, 1)	
Color.LemonChiffon	ColorValue("#fffacd") ColorValue("LEMONCHIFFON")	RGBA(255, 250, 205, 1)	
Color.LightBlue	ColorValue("#add8e6") ColorValue("LightBlue")	RGBA(173, 216, 230, 1)	
Color.LightCoral	ColorValue("#f08080") ColorValue("lightcoral")	RGBA(240, 128, 128, 1)	
Color.LightCyan	ColorValue("#e0ffff") ColorValue("LightCyan")	RGBA(224, 255, 255, 1)	
Color.LightGoldenRodYellow	ColorValue("#fafad2") ColorValue("lightgoldenrodyellow")	RGBA(250, 250, 210, 1)	
Color.LightGray	ColorValue("#d3d3d3") ColorValue("LightGray")	RGBA(211, 211, 211, 1)	
Color.LightGreen	ColorValue("#90ee90") ColorValue("lightgreen")	RGBA(144, 238, 144, 1)	
Color.LightGrey	ColorValue("#d3d3d3") ColorValue("LightGrey")	RGBA(211, 211, 211, 1)	
Color.LightPink	ColorValue("#ffb6c1") ColorValue("LIGHTPINK")	RGBA(255, 182, 193, 1)	
Color.LightSalmon	ColorValue("#ffa07a") ColorValue("LightSalmon")	RGBA(255, 160, 122, 1)	
Color.LightSeaGreen	ColorValue("#20b2aa") ColorValue("lightseagreen")	RGBA(32, 178, 170, 1)	
Color.LightSkyBlue	ColorValue("#87cefa") ColorValue("LightSkyBlue")	RGBA(135, 206, 250, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.LightSlateGray	ColorValue("#778899") ColorValue("LIGHTSLATEGRAY")	RGBA(119, 136, 153, 1)	
Color.LightSlateGrey	ColorValue("#778899") ColorValue("LightSlateGrey")	RGBA(119, 136, 153, 1)	
Color.LightSteelBlue	ColorValue("#b0c4de") ColorValue("lightsteelblue")	RGBA(176, 196, 222, 1)	
Color.LightYellow	ColorValue("#ffffe0") ColorValue("LightYellow")	RGBA(255, 255, 224, 1)	
Color.Lime	ColorValue("#00ff00") ColorValue("LIME")	RGBA(0, 255, 0, 1)	
Color.LimeGreen	ColorValue("#32cd32") ColorValue("LimeGreen")	RGBA(50, 205, 50, 1)	
Color.Linen	ColorValue("#faf0e6") ColorValue("linen")	RGBA(250, 240, 230, 1)	
Color.Magenta	ColorValue("#ff00ff") ColorValue("Magenta")	RGBA(255, 0, 255, 1)	
Color.Maroon	ColorValue("#800000") ColorValue("MAROON")	RGBA(128, 0, 0, 1)	
Color.MediumAquamarine	ColorValue("#66cdaa") ColorValue("MediumAquamarine")	RGBA(102, 205, 170, 1)	
Color.MediumBlue	ColorValue("#0000cd") ColorValue("mediumblue")	RGBA(0, 0, 205, 1)	
Color.MediumOrchid	ColorValue("#ba55d3") ColorValue("MediumOrchid")	RGBA(186, 85, 211, 1)	
Color.MediumPurple	ColorValue("#9370db") ColorValue("MEDIUMPURPLE")	RGBA(147, 112, 219, 1)	
Color.MediumSeaGreen	ColorValue("#3cb371") ColorValue("MediumSeaGreen")	RGBA(60, 179, 113, 1)	
Color.MediumSlateBlue	ColorValue("#7b68ee") ColorValue(	RGBA(123, 104, 238, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"mediumslateblue")		
Color.MediumSpringGreen	ColorValue("#00fa9a") ColorValue("MediumSpringGreen")	RGBA(0, 250, 154, 1)	
Color.MediumTurquoise	ColorValue("#48d1cc") ColorValue("MEDIUMTURQUOISE")	RGBA(72, 209, 204, 1)	
Color.MediumVioletRed	ColorValue("#c71585") ColorValue("MediumVioletRed")	RGBA(199, 21, 133, 1)	
Color.MidnightBlue	ColorValue("#191970") ColorValue("midnightblue")	RGBA(25, 25, 112, 1)	
Color.MintCream	ColorValue("#f5fffa") ColorValue("MintCream")	RGBA(245, 255, 250, 1)	
Color.MistyRose	ColorValue("#ffe4e1") ColorValue("MISTYROSE")	RGBA(255, 228, 225, 1)	
Color.Moccasin	ColorValue("#ffe4b5") ColorValue("Moccasin")	RGBA(255, 228, 181, 1)	
Color.NavajoWhite	ColorValue("#ffdead") ColorValue("navajowhite")	RGBA(255, 222, 173, 1)	
Color.Navy	ColorValue("#000080") ColorValue("Navy")	RGBA(0, 0, 128, 1)	
Color.OldLace	ColorValue("#fdf5e6") ColorValue("OLDLACE")	RGBA(253, 245, 230, 1)	
Color.Olive	ColorValue("#808000") ColorValue("Olive")	RGBA(128, 128, 0, 1)	
Color.OliveDrab	ColorValue("#6b8e23") ColorValue("olivedrab")	RGBA(107, 142, 35, 1)	
Color.Orange	ColorValue("#ffa500") ColorValue("Orange")	RGBA(255, 165, 0, 1)	
Color.OrangeRed	ColorValue("#ff4500") ColorValue("ORANGERED")	RGBA(255, 69, 0, 1)	
Color.Orchid	ColorValue("#da70d6") ColorValue("Orchid")	RGBA(218, 112, 214, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.PaleGoldenRod	ColorValue("#eee8aa") ColorValue("palegoldenrod")	RGBA(238, 232, 170, 1)	
Color.PaleGreen	ColorValue("#98fb98") ColorValue("PaleGreen")	RGBA(152, 251, 152, 1)	
Color.PaleTurquoise	ColorValue("#afeeee") ColorValue("PALETURQUOISE")	RGBA(175, 238, 238, 1)	
Color.PaleVioletRed	ColorValue("#db7093") ColorValue("PaleVioletRed")	RGBA(219, 112, 147, 1)	
Color.PapayaWhip	ColorValue("#ffefd5") ColorValue("papayawhip")	RGBA(255, 239, 213, 1)	
Color.PeachPuff	ColorValue("#ffdab9") ColorValue("PeachPuff")	RGBA(255, 218, 185, 1)	
Color.Peru	ColorValue("#cd853f") ColorValue("PERU")	RGBA(205, 133, 63, 1)	
Color.Pink	ColorValue("#ffc0cb") ColorValue("Pink")	RGBA(255, 192, 203, 1)	
Color.Plum	ColorValue("#dda0dd") ColorValue("plum")	RGBA(221, 160, 221, 1)	
Color.PowderBlue	ColorValue("#b0e0e6") ColorValue("PowderBlue")	RGBA(176, 224, 230, 1)	
Color.Purple	ColorValue("#800080") ColorValue("PURPLE")	RGBA(128, 0, 128, 1)	
Color.Red	ColorValue("#ff0000") ColorValue("Red")	RGBA(255, 0, 0, 1)	
Color.RosyBrown	ColorValue("#bc8f8f") ColorValue("rosybrown")	RGBA(188, 143, 143, 1)	
Color.RoyalBlue	ColorValue("#4169e1") ColorValue("RoyalBlue")	RGBA(65, 105, 225, 1)	
Color.SaddleBrown	ColorValue("#8b4513") ColorValue("SADDLEBROWN")	RGBA(139, 69, 19, 1)	
Color.Salmon	ColorValue("#fa8072") ColorValue("Salmon")	RGBA(250, 128, 114, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.SandyBrown	ColorValue("#f4a460") ColorValue("sandybrown")	RGBA(244, 164, 96, 1)	
Color.SeaGreen	ColorValue("#2e8b57") ColorValue("SeaGreen")	RGBA(46, 139, 87, 1)	
Color.SeaShell	ColorValue("#fff5ee") ColorValue("SEASHELL")	RGBA(255, 245, 238, 1)	
Color.Sienna	ColorValue("#a0522d") ColorValue("Sienna")	RGBA(160, 82, 45, 1)	
Color.Silver	ColorValue("#c0c0c0") ColorValue("silver")	RGBA(192, 192, 192, 1)	
Color.SkyBlue	ColorValue("#87ceeb") ColorValue("SkyBlue")	RGBA(135, 206, 235, 1)	
Color.SlateBlue	ColorValue("#6a5acd") ColorValue("SLATEBLUE")	RGBA(106, 90, 205, 1)	
Color.SlateGray	ColorValue("#708090") ColorValue("SlateGray")	RGBA(112, 128, 144, 1)	
Color.SlateGrey	ColorValue("#708090") ColorValue("slategrey")	RGBA(112, 128, 144, 1)	
Color.Snow	ColorValue("#fffafa") ColorValue("Snow")	RGBA(255, 250, 250, 1)	
Color.SpringGreen	ColorValue("#00ff7f") ColorValue("SPRINGGREEN")	RGBA(0, 255, 127, 1)	
Color.SteelBlue	ColorValue("#4682b4") ColorValue("SteelBlue")	RGBA(70, 130, 180, 1)	
Color.Tan	ColorValue("#d2b48c") ColorValue("tan")	RGBA(210, 180, 140, 1)	
Color.Teal	ColorValue("#008080") ColorValue("Teal")	RGBA(0, 128, 128, 1)	
Color.Thistle	ColorValue("#d8bfd8") ColorValue("THISTLE")	RGBA(216, 191, 216, 1)	
Color.Tomato	ColorValue("#ff6347") ColorValue("Tomato")	RGBA(255, 99, 71, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Transparent	ColorValue("#00000000") ColorValue("Transparent")	RGBA(0, 0, 0, 0)	
Color.Turquoise	ColorValue("#40e0d0") ColorValue("turquoise")	RGBA(64, 224, 208, 1)	
Color.Violet	ColorValue("#ee82ee") ColorValue("Violet")	RGBA(238, 130, 238, 1)	
Color.Wheat	ColorValue("#f5deb3") ColorValue("WHEAT")	RGBA(245, 222, 179, 1)	
Color.White	ColorValue("#ffffff") ColorValue("White")	RGBA(255, 255, 255, 1)	
Color.WhiteSmoke	ColorValue("#f5f5f5") ColorValue("whitesmoke")	RGBA(245, 245, 245, 1)	
Color.Yellow	ColorValue("#ffff00") ColorValue("Yellow")	RGBA(255, 255, 0, 1)	
Color.YellowGreen	ColorValue("#9acd32") ColorValue("YELLOWGREEN")	RGBA(154, 205, 50, 1)	

Color enumeration and ColorFade, ColorValue, and RGBA functions

06/19/2025

Expand table

Functions	Applies to
Color	✅ Model-driven apps
ColorFade **ColorValue**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
RGBA	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Use built-in color values, define custom colors, and use the alpha channel to control transparency.

> ⓘ Note
>
> PAC CLI pac power-fx commands don't support the **Color** enumeration.

Description

Use the **Color** enumeration to access colors defined by HTML's Cascading Style Sheets (CSS). For example, **Color.Red** returns pure red. A list of these colors appears at the end of this article.

The **ColorValue** function returns a color based on a CSS color string. The string can be in any of these forms:

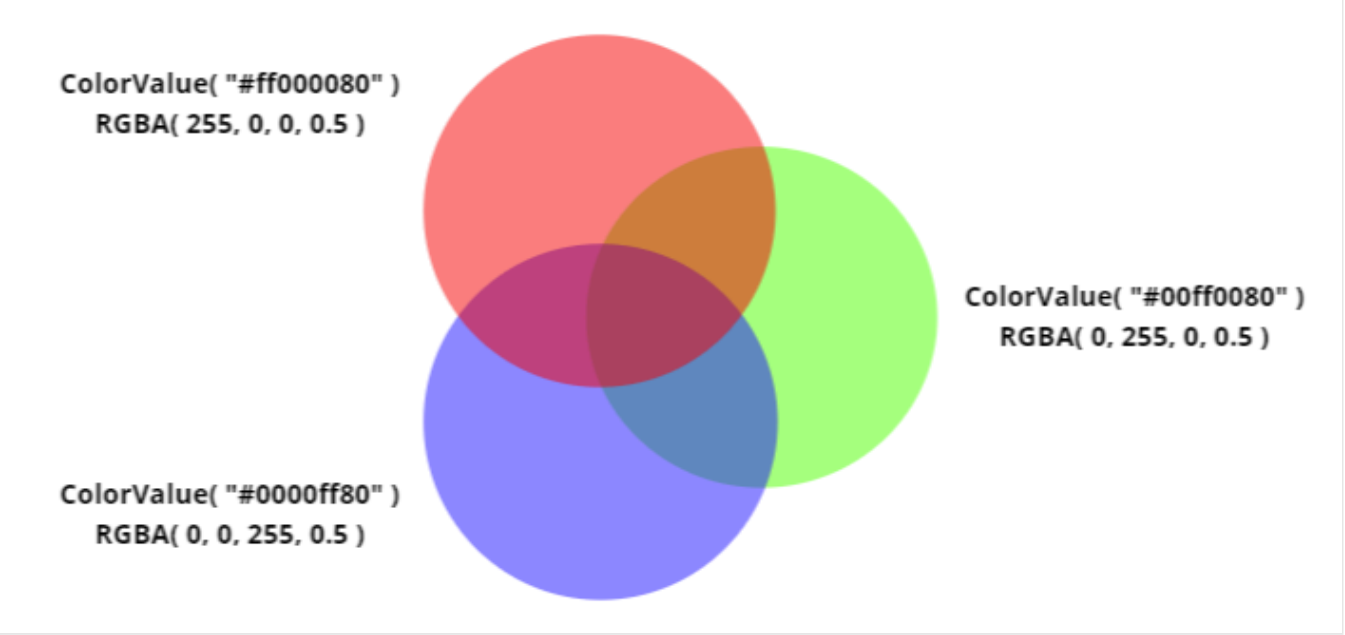
- **CSS color name:** **"RoxyBrown"** and **"OliveDrab"** are examples. These names don't include spaces. The list of supported colors appears later in this article.
- **6-digit hex value:** For example, **"#ffd700"** is the same as **"Gold"**. The string uses the format "*#rrggbb*", where *rr* is the red portion in two hexadecimal digits, *gg* is the green, and *bb* is the blue.
- **8-digit hex value:** For example, **"#ff7f5080"** is the same as **"Coral"** with a 50% alpha channel. The string uses the format "*#rrggbbaa*", where *rr*, *gg*, and *bb* are identical to the 6-digit form. The alpha channel is represented by *aa*: **00** means fully transparent, and **ff** means fully opaque.

The **RGBA** function returns a color based on red, green, and blue components. The function also includes an alpha channel for mixing colors of controls layered in front of each other. The alpha channel ranges from 0% or 0% (fully transparent and invisible) to 1% or 100% (fully opaque and completely blocks any layers behind a control).

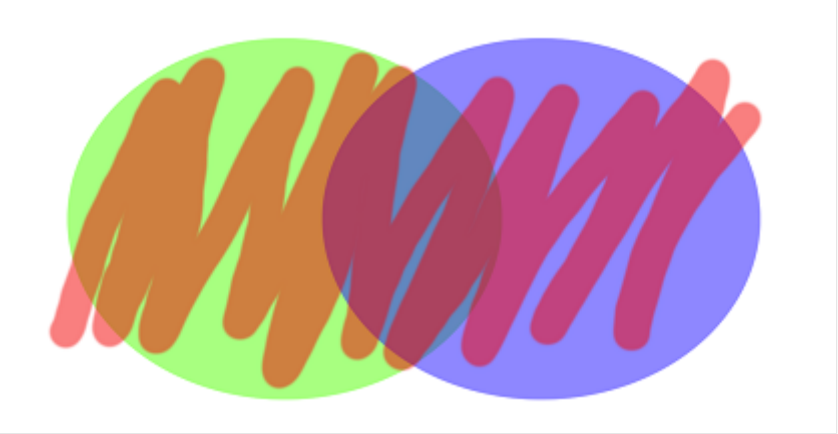
The **ColorFade** function returns a brighter or darker version of a color. The amount of fade ranges from -1 (fully darkens a color to black), to 0 (doesn't affect the color), to 1 (fully brightens a color to white).

Alpha channel

In a canvas app, you layer controls in front of one another and set the transparency of a control to any controls behind it. As a result, colors blend through the layers. For example, this diagram shows how the three primary colors mix with an alpha setting of 50 percent:



You can also blend images in file formats that support alpha channels. For example, you can't blend .jpeg files, but you can blend .png files. The next graphic shows the same red, green, and blue colors from the previous example, but the red color appears as a squiggle (instead of a circle) in a .png file with a 50 percent alpha channel:



If you specify a **Color** enumeration value or build a **ColorValue** formula with a color name or a 6-digit hexadecimal value, the alpha setting is 100 percent, which is fully opaque.

Syntax

Color.*ColorName*

- *ColorName* - Required. A Cascading Style Sheet (CSS) color name. The list of possible enumeration values is at the end of this article.

ColorValue(*CSSColor*)

- *CSSColor* - Required. A Cascading Style Sheet (CSS) color definition. S name, OliveDralike a hex value, like **#6b8e23** olikeffd420**. Hex values can use either #_rrggbb__usgormat. format

ColorValue(*Dynamic*)

- *Dynamic* - Required. A **Dynamic** value with a string that represents a Cascading Style Sheet (CSS) color definition.

RGBA(*Red*, *Green*, *Blue*, *Alpha*)

- *Red*, *Green*, *Blue* - Required. Color component values that range from 0 (no saturation) to 255 (full saturation).
- *Alpha* - Required. Alpha component that ranges from 0 (fully transparent) to 1 (fully opaque). You can also use a percentage, 0% to 100%.

ColorFade(*Color*, *FadeAmount*)

- *Color* - Required. A color value like **Color.Red** or the output from **ColorValue** or **RGBA**.
- *FadeAmount* - Required. A number between -1 and 1. -1 fully darkens a color to black, 0 doesn't affect the color, and 1 fully brightens a color to white. You can also use a percentage, from -100% to 100%.

Built-in colors

Expand table

Color enumeration	ColorValue	RGBA	Color Swatch
Color.AliceBlue	ColorValue("#f0f8ff") ColorValue("aliceblue")	RGBA(240, 248, 255, 1)	
Color.AntiqueWhite	ColorValue("#faebd7") ColorValue("AntiqueWhite")	RGBA(250, 235, 215, 1)	
Color.Aqua	ColorValue("#00ffff") ColorValue("AQUA")	RGBA(0, 255, 255, 1)	
Color.Aquamarine	ColorValue("#7fffd4") ColorValue("Aquamarine")	RGBA(127, 255, 212, 1)	
Color.Azure	ColorValue("#f0ffff") ColorValue("azure")	RGBA(240, 255, 255, 1)	
Color.Beige	ColorValue("#f5f5dc") ColorValue("Beige")	RGBA(245, 245, 220, 1)	
Color.Bisque	ColorValue("#ffe4c4") ColorValue("BISQUE")	RGBA(255, 228, 196, 1)	
Color.Black	ColorValue("#000000") ColorValue("Black")	RGBA(0, 0, 0, 1)	
Color.BlanchedAlmond	ColorValue("#ffebcd") ColorValue("blanchedalmond")	RGBA(255, 235, 205, 1)	
Color.Blue	ColorValue("#0000ff") ColorValue("Blue")	RGBA(0, 0, 255, 1)	
Color.BlueViolet	ColorValue("#8a2be2") ColorValue("BLUEVIOLET")	RGBA(138, 43, 226, 1)	
Color.Brown	ColorValue("#a52a2a") ColorValue("Brown")	RGBA(165, 42, 42, 1)	
Color.Burlywood	ColorValue("#deb887") ColorValue("burlywood")	RGBA(222, 184, 135, 1)	
Color.CadetBlue	ColorValue("#5f9ea0") ColorValue("CadetBlue")	RGBA(95, 158, 160, 1)	
Color.Chartreuse	ColorValue("#7fff00") ColorValue("CHARTREUSE")	RGBA(127, 255, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Chocolate	ColorValue("#d2691e") ColorValue("Chocolate")	RGBA(210, 105, 30, 1)	
Color.Coral	ColorValue("#ff7f50") ColorValue("coral")	RGBA(255, 127, 80, 1)	
Color.CornflowerBlue	ColorValue("#6495ed") ColorValue("CornflowerBlue")	RGBA(100, 149, 237, 1)	
Color.Cornsilk	ColorValue("#fff8dc") ColorValue("CORNSILK")	RGBA(255, 248, 220, 1)	
Color.Crimson	ColorValue("#dc143c") ColorValue("Crimson")	RGBA(220, 20, 60, 1)	
Color.Cyan	ColorValue("#00ffff") ColorValue("cyan")	RGBA(0, 255, 255, 1)	
Color.DarkBlue	ColorValue("#00008b") ColorValue("DarkBlue")	RGBA(0, 0, 139, 1)	
Color.DarkCyan	ColorValue("#008b8b") ColorValue("DARKCYAN")	RGBA(0, 139, 139, 1)	
Color.DarkGoldenRod	ColorValue("#b8860b") ColorValue("DarkGoldenRod")	RGBA(184, 134, 11, 1)	
Color.DarkGray	ColorValue("#a9a9a9") ColorValue("darkgray")	RGBA(169, 169, 169, 1)	
Color.DarkGreen	ColorValue("#006400") ColorValue("DarkGreen")	RGBA(0, 100, 0, 1)	
Color.DarkGrey	ColorValue("#a9a9a9") ColorValue("DARKGREY")	RGBA(169, 169, 169, 1)	
Color.DarkKhaki	ColorValue("#bdb76b") ColorValue("DarkKhaki")	RGBA(189, 183, 107, 1)	
Color.DarkMagenta	ColorValue("#8b008b") ColorValue("darkmagenta")	RGBA(139, 0, 139, 1)	
Color.DarkOliveGreen	ColorValue("#556b2f") ColorValue("DarkOliveGreen")	RGBA(85, 107, 47, 1)	
Color.DarkOrange	ColorValue("#ff8c00") ColorValue(	RGBA(255, 140, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"DARKORANGE")		
Color.DarkOrchid	ColorValue("#9932cc") ColorValue("DarkOrchid")	RGBA(153, 50, 204, 1)	
Color.DarkRed	ColorValue("#8b0000") ColorValue("darkred")	RGBA(139, 0, 0, 1)	
Color.DarkSalmon	ColorValue("#e9967a") ColorValue("DarkSalmon")	RGBA(233, 150, 122, 1)	
Color.DarkSeaGreen	ColorValue("#8fbc8f") ColorValue("DARKSEAGREEN")	RGBA(143, 188, 143, 1)	
Color.DarkSlateBlue	ColorValue("#483d8b") ColorValue("DarkSlateBlue")	RGBA(72, 61, 139, 1)	
Color.DarkSlateGray	ColorValue("#2f4f4f") ColorValue("darkslategray")	RGBA(47, 79, 79, 1)	
Color.DarkSlateGrey	ColorValue("#2f4f4f") ColorValue("DarkSlateGrey")	RGBA(47, 79, 79, 1)	
Color.DarkTurquoise	ColorValue("#00ced1") ColorValue("DARKTURQUOISE")	RGBA(0, 206, 209, 1)	
Color.DarkViolet	ColorValue("#9400d3") ColorValue("DarkViolet")	RGBA(148, 0, 211, 1)	
Color.DeepPink	ColorValue("#ff1493") ColorValue("deeppink")	RGBA(255, 20, 147, 1)	
Color.DeepSkyBlue	ColorValue("#00bfff") ColorValue("DeepSkyBlue")	RGBA(0, 191, 255, 1)	
Color.DimGray	ColorValue("#696969") ColorValue("DIMGRAY")	RGBA(105, 105, 105, 1)	
Color.DimGrey	ColorValue("#696969") ColorValue("DimGrey")	RGBA(105, 105, 105, 1)	
Color.DodgerBlue	ColorValue("#1e90ff") ColorValue("dodgerblue")	RGBA(30, 144, 255, 1)	
Color.FireBrick	ColorValue("#b22222") ColorValue("FireBrick")	RGBA(178, 34, 34, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.FloralWhite	ColorValue("#fffaf0") ColorValue("FLORALWHITE")	RGBA(255, 250, 240, 1)	
Color.ForestGreen	ColorValue("#228b22") ColorValue("ForestGreen")	RGBA(34, 139, 34, 1)	
Color.Fuchsia	ColorValue("#ff00ff") ColorValue("fuchsia")	RGBA(255, 0, 255, 1)	
Color.Gainsboro	ColorValue("#dcdcdc") ColorValue("Gainsboro")	RGBA(220, 220, 220, 1)	
Color.GhostWhite	ColorValue("#f8f8ff") ColorValue("GHOSTWHITE")	RGBA(248, 248, 255, 1)	
Color.Gold	ColorValue("#ffd700") ColorValue("Gold")	RGBA(255, 215, 0, 1)	
Color.GoldenRod	ColorValue("#daa520") ColorValue("goldenrod")	RGBA(218, 165, 32, 1)	
Color.Gray	ColorValue("#808080") ColorValue("Gray")	RGBA(128, 128, 128, 1)	
Color.Green	ColorValue("#008000") ColorValue("GREEN")	RGBA(0, 128, 0, 1)	
Color.GreenYellow	ColorValue("#adff2f") ColorValue("GreenYellow")	RGBA(173, 255, 47, 1)	
Color.Grey	ColorValue("#808080") ColorValue("grey")	RGBA(128, 128, 128, 1)	
Color.Honeydew	ColorValue("#f0fff0") ColorValue("Honeydew")	RGBA(240, 255, 240, 1)	
Color.HotPink	ColorValue("#ff69b4") ColorValue("HOTPINK")	RGBA(255, 105, 180, 1)	
Color.IndianRed	ColorValue("#cd5c5c") ColorValue("IndianRed")	RGBA(205, 92, 92, 1)	
Color.Indigo	ColorValue("#4b0082") ColorValue("indigo")	RGBA(75, 0, 130, 1)	
Color.Ivory	ColorValue("#fffff0") ColorValue("Ivory")	RGBA(255, 255, 240, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Khaki	ColorValue("#f0e68c") ColorValue("KHAKI")	RGBA(240, 230, 140, 1)	
Color.Lavender	ColorValue("#e6e6fa") ColorValue("Lavender")	RGBA(230, 230, 250, 1)	
Color.LavenderBlush	ColorValue("#fff0f5") ColorValue("lavenderblush")	RGBA(255, 240, 245, 1)	
Color.LawnGreen	ColorValue("#7cfc00") ColorValue("LawnGreen")	RGBA(124, 252, 0, 1)	
Color.LemonChiffon	ColorValue("#fffacd") ColorValue("LEMONCHIFFON")	RGBA(255, 250, 205, 1)	
Color.LightBlue	ColorValue("#add8e6") ColorValue("LightBlue")	RGBA(173, 216, 230, 1)	
Color.LightCoral	ColorValue("#f08080") ColorValue("lightcoral")	RGBA(240, 128, 128, 1)	
Color.LightCyan	ColorValue("#e0ffff") ColorValue("LightCyan")	RGBA(224, 255, 255, 1)	
Color.LightGoldenRodYellow	ColorValue("#fafad2") ColorValue("lightgoldenrodyellow")	RGBA(250, 250, 210, 1)	
Color.LightGray	ColorValue("#d3d3d3") ColorValue("LightGray")	RGBA(211, 211, 211, 1)	
Color.LightGreen	ColorValue("#90ee90") ColorValue("lightgreen")	RGBA(144, 238, 144, 1)	
Color.LightGrey	ColorValue("#d3d3d3") ColorValue("LightGrey")	RGBA(211, 211, 211, 1)	
Color.LightPink	ColorValue("#ffb6c1") ColorValue("LIGHTPINK")	RGBA(255, 182, 193, 1)	
Color.LightSalmon	ColorValue("#ffa07a") ColorValue("LightSalmon")	RGBA(255, 160, 122, 1)	
Color.LightSeaGreen	ColorValue("#20b2aa") ColorValue("lightseagreen")	RGBA(32, 178, 170, 1)	
Color.LightSkyBlue	ColorValue("#87cefa") ColorValue("LightSkyBlue")	RGBA(135, 206, 250, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.LightSlateGray	ColorValue("#778899") ColorValue("LIGHTSLATEGRAY")	RGBA(119, 136, 153, 1)	
Color.LightSlateGrey	ColorValue("#778899") ColorValue("LightSlateGrey")	RGBA(119, 136, 153, 1)	
Color.LightSteelBlue	ColorValue("#b0c4de") ColorValue("lightsteelblue")	RGBA(176, 196, 222, 1)	
Color.LightYellow	ColorValue("#ffffe0") ColorValue("LightYellow")	RGBA(255, 255, 224, 1)	
Color.Lime	ColorValue("#00ff00") ColorValue("LIME")	RGBA(0, 255, 0, 1)	
Color.LimeGreen	ColorValue("#32cd32") ColorValue("LimeGreen")	RGBA(50, 205, 50, 1)	
Color.Linen	ColorValue("#faf0e6") ColorValue("linen")	RGBA(250, 240, 230, 1)	
Color.Magenta	ColorValue("#ff00ff") ColorValue("Magenta")	RGBA(255, 0, 255, 1)	
Color.Maroon	ColorValue("#800000") ColorValue("MAROON")	RGBA(128, 0, 0, 1)	
Color.MediumAquamarine	ColorValue("#66cdaa") ColorValue("MediumAquamarine")	RGBA(102, 205, 170, 1)	
Color.MediumBlue	ColorValue("#0000cd") ColorValue("mediumblue")	RGBA(0, 0, 205, 1)	
Color.MediumOrchid	ColorValue("#ba55d3") ColorValue("MediumOrchid")	RGBA(186, 85, 211, 1)	
Color.MediumPurple	ColorValue("#9370db") ColorValue("MEDIUMPURPLE")	RGBA(147, 112, 219, 1)	
Color.MediumSeaGreen	ColorValue("#3cb371") ColorValue("MediumSeaGreen")	RGBA(60, 179, 113, 1)	
Color.MediumSlateBlue	ColorValue("#7b68ee") ColorValue(	RGBA(123, 104, 238, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"mediumslateblue")		
Color.MediumSpringGreen	ColorValue("#00fa9a") ColorValue("MediumSpringGreen")	RGBA(0, 250, 154, 1)	
Color.MediumTurquoise	ColorValue("#48d1cc") ColorValue("MEDIUMTURQUOISE")	RGBA(72, 209, 204, 1)	
Color.MediumVioletRed	ColorValue("#c71585") ColorValue("MediumVioletRed")	RGBA(199, 21, 133, 1)	
Color.MidnightBlue	ColorValue("#191970") ColorValue("midnightblue")	RGBA(25, 25, 112, 1)	
Color.MintCream	ColorValue("#f5fffa") ColorValue("MintCream")	RGBA(245, 255, 250, 1)	
Color.MistyRose	ColorValue("#ffe4e1") ColorValue("MISTYROSE")	RGBA(255, 228, 225, 1)	
Color.Moccasin	ColorValue("#ffe4b5") ColorValue("Moccasin")	RGBA(255, 228, 181, 1)	
Color.NavajoWhite	ColorValue("#ffdead") ColorValue("navajowhite")	RGBA(255, 222, 173, 1)	
Color.Navy	ColorValue("#000080") ColorValue("Navy")	RGBA(0, 0, 128, 1)	
Color.OldLace	ColorValue("#fdf5e6") ColorValue("OLDLACE")	RGBA(253, 245, 230, 1)	
Color.Olive	ColorValue("#808000") ColorValue("Olive")	RGBA(128, 128, 0, 1)	
Color.OliveDrab	ColorValue("#6b8e23") ColorValue("olivedrab")	RGBA(107, 142, 35, 1)	
Color.Orange	ColorValue("#ffa500") ColorValue("Orange")	RGBA(255, 165, 0, 1)	
Color.OrangeRed	ColorValue("#ff4500") ColorValue("ORANGERED")	RGBA(255, 69, 0, 1)	
Color.Orchid	ColorValue("#da70d6") ColorValue("Orchid")	RGBA(218, 112, 214, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.PaleGoldenRod	ColorValue("#eee8aa") ColorValue("palegoldenrod")	RGBA(238, 232, 170, 1)	
Color.PaleGreen	ColorValue("#98fb98") ColorValue("PaleGreen")	RGBA(152, 251, 152, 1)	
Color.PaleTurquoise	ColorValue("#afeeee") ColorValue("PALETURQUOISE")	RGBA(175, 238, 238, 1)	
Color.PaleVioletRed	ColorValue("#db7093") ColorValue("PaleVioletRed")	RGBA(219, 112, 147, 1)	
Color.PapayaWhip	ColorValue("#ffefd5") ColorValue("papayawhip")	RGBA(255, 239, 213, 1)	
Color.PeachPuff	ColorValue("#ffdab9") ColorValue("PeachPuff")	RGBA(255, 218, 185, 1)	
Color.Peru	ColorValue("#cd853f") ColorValue("PERU")	RGBA(205, 133, 63, 1)	
Color.Pink	ColorValue("#ffc0cb") ColorValue("Pink")	RGBA(255, 192, 203, 1)	
Color.Plum	ColorValue("#dda0dd") ColorValue("plum")	RGBA(221, 160, 221, 1)	
Color.PowderBlue	ColorValue("#b0e0e6") ColorValue("PowderBlue")	RGBA(176, 224, 230, 1)	
Color.Purple	ColorValue("#800080") ColorValue("PURPLE")	RGBA(128, 0, 128, 1)	
Color.Red	ColorValue("#ff0000") ColorValue("Red")	RGBA(255, 0, 0, 1)	
Color.RosyBrown	ColorValue("#bc8f8f") ColorValue("rosybrown")	RGBA(188, 143, 143, 1)	
Color.RoyalBlue	ColorValue("#4169e1") ColorValue("RoyalBlue")	RGBA(65, 105, 225, 1)	
Color.SaddleBrown	ColorValue("#8b4513") ColorValue("SADDLEBROWN")	RGBA(139, 69, 19, 1)	
Color.Salmon	ColorValue("#fa8072") ColorValue("Salmon")	RGBA(250, 128, 114, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.SandyBrown	ColorValue("#f4a460") ColorValue("sandybrown")	RGBA(244, 164, 96, 1)	
Color.SeaGreen	ColorValue("#2e8b57") ColorValue("SeaGreen")	RGBA(46, 139, 87, 1)	
Color.SeaShell	ColorValue("#fff5ee") ColorValue("SEASHELL")	RGBA(255, 245, 238, 1)	
Color.Sienna	ColorValue("#a0522d") ColorValue("Sienna")	RGBA(160, 82, 45, 1)	
Color.Silver	ColorValue("#c0c0c0") ColorValue("silver")	RGBA(192, 192, 192, 1)	
Color.SkyBlue	ColorValue("#87ceeb") ColorValue("SkyBlue")	RGBA(135, 206, 235, 1)	
Color.SlateBlue	ColorValue("#6a5acd") ColorValue("SLATEBLUE")	RGBA(106, 90, 205, 1)	
Color.SlateGray	ColorValue("#708090") ColorValue("SlateGray")	RGBA(112, 128, 144, 1)	
Color.SlateGrey	ColorValue("#708090") ColorValue("slategrey")	RGBA(112, 128, 144, 1)	
Color.Snow	ColorValue("#fffafa") ColorValue("Snow")	RGBA(255, 250, 250, 1)	
Color.SpringGreen	ColorValue("#00ff7f") ColorValue("SPRINGGREEN")	RGBA(0, 255, 127, 1)	
Color.SteelBlue	ColorValue("#4682b4") ColorValue("SteelBlue")	RGBA(70, 130, 180, 1)	
Color.Tan	ColorValue("#d2b48c") ColorValue("tan")	RGBA(210, 180, 140, 1)	
Color.Teal	ColorValue("#008080") ColorValue("Teal")	RGBA(0, 128, 128, 1)	
Color.Thistle	ColorValue("#d8bfd8") ColorValue("THISTLE")	RGBA(216, 191, 216, 1)	
Color.Tomato	ColorValue("#ff6347") ColorValue("Tomato")	RGBA(255, 99, 71, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Transparent	ColorValue("#00000000") ColorValue("Transparent")	RGBA(0, 0, 0, 0)	
Color.Turquoise	ColorValue("#40e0d0") ColorValue("turquoise")	RGBA(64, 224, 208, 1)	
Color.Violet	ColorValue("#ee82ee") ColorValue("Violet")	RGBA(238, 130, 238, 1)	
Color.Wheat	ColorValue("#f5deb3") ColorValue("WHEAT")	RGBA(245, 222, 179, 1)	
Color.White	ColorValue("#ffffff") ColorValue("White")	RGBA(255, 255, 255, 1)	
Color.WhiteSmoke	ColorValue("#f5f5f5") ColorValue("whitesmoke")	RGBA(245, 245, 245, 1)	
Color.Yellow	ColorValue("#ffff00") ColorValue("Yellow")	RGBA(255, 255, 0, 1)	
Color.YellowGreen	ColorValue("#9acd32") ColorValue("YELLOWGREEN")	RGBA(154, 205, 50, 1)	

Acceleration, App, Compass, Connection, and Location signals in Power Apps

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Returns information about the app's environment, such as where the user is located in the world and which screen is displayed.

Description and syntax

Signals are values that can change at any time, independent of how the user may be interacting with the app. Formulas that are based on signals automatically recalculate as these values change.

Signals typically return a record of information. You can use and store this information as a record, or you can extract individual properties by using the . operator.

> **Note**
>
> The **Acceleration** and **Compass** functions return accurate values in a native player such as on iOS or Android, but those functions return zero values as you create or modify an app in the browser.

Acceleration

The **Acceleration** signal returns the device's acceleration in three dimensions relative to the device's screen. Acceleration is measured in *g* units of 9.81 m/second^2 or 32.2 ft/second^2 (the acceleration that the Earth imparts to objects at its surface due to gravity).

Expand table

Property	Description
Acceleration.X	Right and left. Right is a positive number.
Acceleration.Y	Forward and back. Forward is a positive number.

Property	Description
Acceleration.Z	Up and down. Up is a positive number.

App

Among other properties, the **App** object includes a signal that indicates which screen is showing.

Expand table

Property	Description
App.ActiveScreen	Screen that's showing. Returns a screen object, which you can use to reference properties of the screen or compare to another screen to determine which screen is showing. You can use the **Back** or **Navigate** function to change the screen that's showing.

More information: **App** object documentation.

Compass

The **Compass** signal returns the compass heading of the top of the screen. The heading is based on magnetic north.

Expand table

Property	Description
Compass.Heading	Heading in degrees. Returns a number 0 to 360, and 0 is north.

Connection

The **Connection** signal returns the information about the network connection. When on a metered connection, you may want to limit how much data you send or receive over the network. Or, if your app is enabled for offline, you may want to display different information depending on the current sync state.

Expand table

Property	Description
Connection.Connected	Returns a Boolean **true** or **false** value that indicates whether the device is connected to a network.
Connection.Metered	Returns a Boolean **true** or **false** value that indicates whether the connection is metered.
Connection.Sync	Returns an enum named **ConnectionSync** which holds the current synchronization state for an app that is enabled for offline.

Below are the enums values available for Connection.Sync.

Expand table

Enum value	Description
Connected	Your device is connected to the network and your app is ready to work offline
ConnectedWithWarning	Your device is connected with a warning.
ConnectedPendingUpsync	Some data on your device must be synchronized with the service.
ConnectedError	There was an error on the last synchronization attempt.
ConnectedRefresh	Your app is currently synchronizing data with the service.
NotConnected	Your device isn't connected to the service.
NotConnectedWithWarning	The last synchronization warning.
NotConnectedPendingUpsync	There are updates on your device waiting to be synchronized.
NotConnectedSyncError	The last synchronization error.

Location

The **Location** signal returns the location of the device based on the Global Positioning System (GPS) and other device information, such as cell-tower communications and IP address.

When a user accesses the location information for the first time, the device may prompt that user to allow access to this information.

As the location changes, dependencies on the location will continuously recalculate, which will consume power from the device's battery. To conserve battery life, you can

use the **Enable** and **Disable** functions to turn location updates on and off. Location is automatically turned off if the displayed screen doesn't depend on location information.

Expand table

Property	Description
Location.Altitude	Returns a number that indicates the altitude, measured in meters, above sea level.
Location.Latitude	Returns a number, from –90 to 90, that indicates the latitude, as measured in degrees from the equator. A positive number indicates a location that's north of the equator.
Location.Longitude	Returns a number, from –180 to 180, that indicates the longitude, as measured in degrees from Greenwich, England. A positive number indicates a location that's east of Greenwich.

Examples

In a baseball field, a pitcher throws a phone from the pitcher's mound to a catcher at home plate. The phone is lying flat with respect to the ground, the top of the screen is pointed at the catcher, and the pitcher adds no spin. At this location, the phone has cellular network service that's metered but no WiFi. The **PlayBall** screen is displayed. The app is enabled for offline and is connected to the service.

Expand table

Formula	Description	Result
Location.Latitude	Returns the latitude of the current location. The field is located at map coordinates 47.591 N, 122.333 W.	47.591 The latitude will change continuously as the ball moves between the pitcher and the catcher.
Location.Longitude	Returns the longitude of the current location.	122.333 The longitude will change continuously as the ball moves between the pitcher and the catcher.
Location	Returns the latitude and longitude of the current location, as a record.	{ Latitude: 47.591, Longitude: 122.333 }

Formula	Description	Result
Compass.Heading	Returns the compass heading of the top of the screen. At this field, home plate is roughly southwest from the pitcher's mound.	230.25
Acceleration.X	Returns the acceleration of the device side to side. The pitcher is throwing the phone straight ahead with respect to the screen's top, so the device isn't accelerating side to side.	0
Acceleration.Y	Returns the acceleration of the device front to back. The pitcher initially gives the device a large acceleration when throwing the device, going from 0 to 90 miles per hour (132 feet per second) in half a second. After the device is in the air, ignoring air friction, the device doesn't accelerate further. The device decelerates when the catcher catches it, bringing it to a stop.	8.2, while the pitcher throws the device. 0, while the device is in the air. -8.2, as the catcher catches the device.
Acceleration.Z	Returns the acceleration of the device top to bottom. While in the air, the device experiences the effects of gravity.	0, before the pitcher throws the device. 1, while the device is in the air. 0, after the catcher catches the device.
Acceleration	Returns the acceleration as a record.	{ X: 0, Y: 264, Z: 0 } as the pitcher throws the device.
Connection.Connected	Returns a Boolean value that indicates whether the device is connected to a network	**true**
Connection.Metered	Returns a Boolean value that indicates whether the connection is metered	**true**
Connection.Sync	Returns an enum value that indicates current sync state.	**ConnectionSync.Connected**
App.ActiveScreen = PlayBall	Returns a Boolean value that indicates whether **PlayBall** is	**true**

Formula	Description	Result
	displayed.	
App.ActiveScreen.Fill	Returns the background color for the displayed screen.	**Color.Green**

Concat and Concatenate functions

06/20/2025

Expand table

Function	Applies to
Concat	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Concatenate	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Concatenates individual strings of text and strings in tables.

Description

The **Concatenate** function concatenates a mix of individual strings and a single-column table of strings. When you use this function with individual strings, it's equivalent to using the **&** operator.

The **Concat** function concatenates the result of a formula applied across all the records of a table, resulting in a single string. Use this function to summarize the strings of a table, just as the **Sum** function does for numbers.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

Use the **Split** or **MatchAll** function to split a string into a table of substrings.

Syntax

Concat(*Table*, *Formula*, *separator*)

- *Table* - Required. Table to operate on.
- *Formula* - Required. Formula to apply across the records of the table.
- *Separator* - Optional. A text value to be inserted between concatenated rows of the table.

Concatenate(*String1* [, *String2*, ...])

- *String(s)* - Required. Mix of individual strings or a single-column table of strings.

Examples

The examples in this section use these global variables:

- **FirstName** = "Jane"
- **LastName** = "Doe"
- **Products** =

Name	Type
"Violin"	"String"
"Cello"	"String"
"Trumpet"	"Wind"

To create these global variables in an app, insert a **Button** control, and set its **OnSelect** property to this formula:

Power Fx

```
Set( FirstName, "Jane" ); Set( LastName, "Doe" );
Set( Products,
    Table(
        { Name: "Violin", Type: "String" },
        { Name: "Cello", Type: "String" },
        { Name: "Trumpet", Type: "Wind" }
    )
)
```

Select the button (by clicking it while you hold down the Alt key).

Concatenate function and the & operator

For these examples, set the **Text** property of a **Label** control to a formula from the first column of the next table.

Expand table

Formula	Description	Result
Concatenate(LastName, ", ", FirstName)	Concatenates the value in **LastName**, the string ", " (a comma followed by a space), and the value in **FirstName**.	"Doe, Jane"
LastName & ", " & FirstName	Same as the previous example except using the **&** operator instead of the function.	"Doe, Jane"

Formula	Description	Result
Concatenate(FirstName, " ", LastName)	Concatenates the value in **FirstName**, the string " " (a single space), and the value in **LastName**.	"Jane Doe"
FirstName & " " & LastName	Same as the previous example, using the **&** operator instead of the function.	"Jane Doe"

Concatenate with a single-column table

For this example, add a blank, vertical Gallery control, set its **Items** property to the formula in the next table, and then add a label in the gallery template.

Expand table

Formula	Description	Result
Concatenate("Name: ", Products.Name, ", Type: ", Products.Type)	For each record in the **Products** table, concatenates the string **"Name: "**, the name of the product, the string **", Type: "** and the type of the product.	A single-column table with a `Value` column containing the following values: "Name: Violin, Type: String", "Name: "Cello, Type: String", "Name: Trumpet, Type: Wind"

Concat function

For these examples, set the **Text** property of a label to a formula from the first column of the next table.

Expand table

Formula	Description	Result
Concat(Products, Name, ", ")	Evaluates the expression **Name** for each record of **Products** and concatenates the results together into a single text string separated by **", "**.	"Violin, Cello, Trumpet"
Concat(Products, "'" & Name & "'", ", ")	Evaluates the expression **"'" & Name & "'"** for each record of **Products** and concatenates the results together into a single text string separated by **", "**.	"'Violin', 'Cello', 'Trumpet'"

Formula	Description	Result
Concat(**Filter(Products, Type = "String"),** **Name, ", ")**	Evaluates the formula **Name** for each record of **Products** that satisfies the filter **Type** = "**String**", and concatenates the results into a single text string separated by **", "**.	"Violin, Cello"

Split and MatchAll

If you used **Concat** with a separator, you can reverse the operation by using the **Split** function.

For these examples, add a blank, vertical gallery, set its **Items** property to a formula in the next table, and then add a label in the gallery template.

Expand table

Formula	Description	Result
Split(**Concat(Products, Name, ", "), ", "** **)**	Splits the text string with the separator **", "**.	A single-column table with a `Value` column containing the following values: "Violin", "Cello", "Trumpet"

Concat and Concatenate functions

06/20/2025

Expand table

Function	Applies to
Concat	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Concatenate	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Concatenates individual strings of text and strings in tables.

Description

The **Concatenate** function concatenates a mix of individual strings and a single-column table of strings. When you use this function with individual strings, it's equivalent to using the **&** operator.

The **Concat** function concatenates the result of a formula applied across all the records of a table, resulting in a single string. Use this function to summarize the strings of a table, just as the **Sum** function does for numbers.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

Use the **Split** or **MatchAll** function to split a string into a table of substrings.

Syntax

Concat(*Table*, *Formula*, *separator*)

- *Table* - Required. Table to operate on.
- *Formula* - Required. Formula to apply across the records of the table.
- *Separator* - Optional. A text value to be inserted between concatenated rows of the table.

Concatenate(*String1* [, *String2*, ...])

- *String(s)* - Required. Mix of individual strings or a single-column table of strings.

Examples

The examples in this section use these global variables:

- **FirstName** = "Jane"
- **LastName** = "Doe"
- **Products** =

Name	Type
"Violin"	"String"
"Cello"	"String"
"Trumpet"	"Wind"

To create these global variables in an app, insert a **Button** control, and set its **OnSelect** property to this formula:

Power Fx

```
Set( FirstName, "Jane" ); Set( LastName, "Doe" );
Set( Products,
    Table(
        { Name: "Violin", Type: "String" },
        { Name: "Cello", Type: "String" },
        { Name: "Trumpet", Type: "Wind" }
    )
)
```

Select the button (by clicking it while you hold down the Alt key).

Concatenate function and the & operator

For these examples, set the **Text** property of a **Label** control to a formula from the first column of the next table.

Expand table

Formula	Description	Result
Concatenate(LastName, ", ", FirstName)	Concatenates the value in **LastName**, the string ", " (a comma followed by a space), and the value in **FirstName**.	"Doe, Jane"
LastName & ", " & FirstName	Same as the previous example except using the **&** operator instead of the function.	"Doe, Jane"

Formula	Description	Result
Concatenate(FirstName, " ", LastName)	Concatenates the value in **FirstName**, the string " " (a single space), and the value in **LastName**.	"Jane Doe"
FirstName & " " & LastName	Same as the previous example, using the **&** operator instead of the function.	"Jane Doe"

Concatenate with a single-column table

For this example, add a blank, vertical Gallery control, set its **Items** property to the formula in the next table, and then add a label in the gallery template.

Expand table

Formula	Description	Result
Concatenate("Name: ", Products.Name, ", Type: ", Products.Type)	For each record in the **Products** table, concatenates the string **"Name: "**, the name of the product, the string **", Type: "** and the type of the product.	A single-column table with a `Value` column containing the following values: "Name: Violin, Type: String", "Name: "Cello, Type: String", "Name: Trumpet, Type: Wind"

Concat function

For these examples, set the **Text** property of a label to a formula from the first column of the next table.

Expand table

Formula	Description	Result
Concat(Products, Name, ", ")	Evaluates the expression **Name** for each record of **Products** and concatenates the results together into a single text string separated by **", "**.	"Violin, Cello, Trumpet"
Concat(Products, "'" & Name & "'", ", ")	Evaluates the expression **"'" & Name & "'"** for each record of **Products** and concatenates the results together into a single text string separated by **", "**.	"'Violin', 'Cello', 'Trumpet'"

Formula	Description	Result
Concat(**Filter(Products, Type = "String"),** **Name, ", ")**	Evaluates the formula **Name** for each record of **Products** that satisfies the filter **Type** = "**String**", and concatenates the results into a single text string separated by **", "**.	"Violin, Cello"

Split and MatchAll

If you used **Concat** with a separator, you can reverse the operation by using the **Split** function.

For these examples, add a blank, vertical gallery, set its **Items** property to a formula in the next table, and then add a label in the gallery template.

Expand table

Formula	Description	Result
Split(**Concat(Products, Name, ", "), ", "** **)**	Splits the text string with the separator **", "**.	A single-column table with a `Value` column containing the following values: "Violin", "Cello", "Trumpet"

Concurrent function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Evaluates multiple formulas concurrently with one another.

Description

The **Concurrent** function allows multiple formulas specified within the same property to be evaluated at the same time if they have connector or Dataverse calls. Normally, multiple formulas are evaluated by chaining them together with the ; (semi-colon) operator, which evaluates each formula sequentially. With the Concurrent function, the app will evaluate all formulas within a property concurrently even after using the ; operator. This concurrency will help users wait less for the same result.

In the OnStart property of your app, use **Concurrent** to improve performance when the app loads data. When data calls don't start until the previous calls finish, the app must wait for the sum of all request times. If data calls start at the same time, the app needs to wait only for the longest request time. Web browsers often improve performance by performing network calls concurrently.

You can't predict the order in which formulas within the **Concurrent** function start and end evaluation. Formulas within the **Concurrent** function shouldn't contain dependencies on other formulas within the same **Concurrent** function, and Power Apps shows an error if you try. From within, you can safely take dependencies on formulas outside the **Concurrent** function because they will complete before the **Concurrent** function starts. Formulas after the **Concurrent** function can safely take dependencies on formulas within: they'll all complete before the **Concurrent** function finishes and moves on to the next formula in a chain (if you use the **;** operator). Watch out for subtle order dependencies if you're calling functions or service methods that have side effects.

You can chain formulas together with the **;** operator within an argument to **Concurrent**. For example, **Concurrent(Set(a, 1); Set(b, a+1), Set(x, 2); Set(y, x+2))** evaluates **Set(a, 1); Set(b, a+1)** concurrently with **Set(x, 2); Set(y, x+2)**. In this case, the dependencies within the formulas are fine: **a** will be set before **b**, and **x** will be set before **y**.

Depending on the device or browser in which the app is running, only a handful of formulas might actually be evaluated concurrently. **Concurrent** uses the available capabilities and won't finish until all formulas have been evaluated.

If you enable **Formula-level error management** (in advanced settings), the first error encountered in argument order is returned from **Concurrent**; otherwise, *blank* is returned. If all formulas are successful, *true* is returned. If one formula fails, the rest of that formula stops, but other formulas continue evaluating.

You can use **Concurrent** only in behavior formulas.

Syntax

Concurrent(*Formula1*, *Formula2* [, ...])

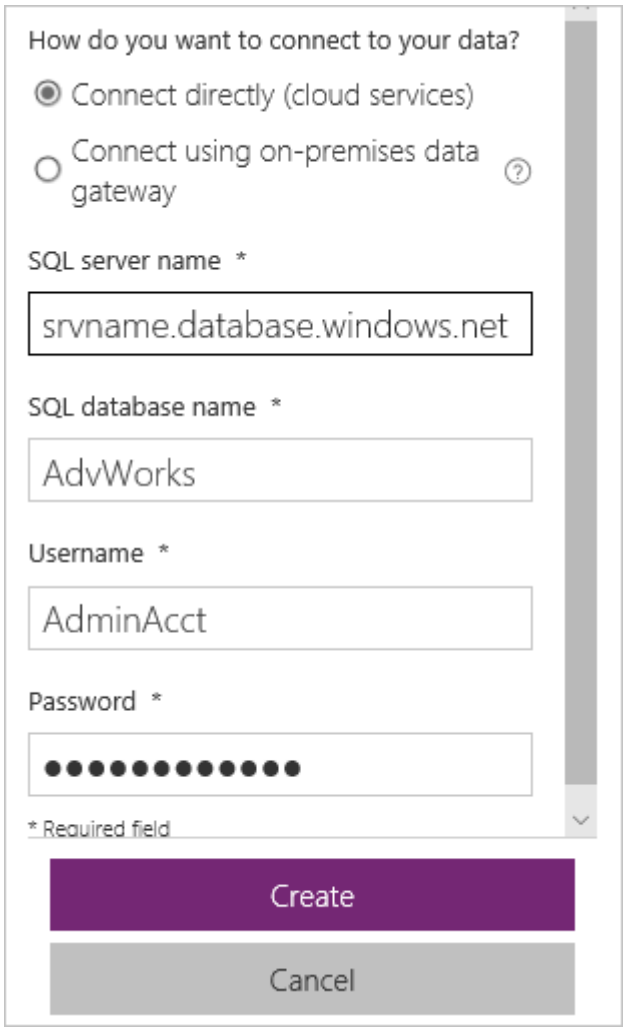
- *Formula(s)* – Required. Formulas to evaluate concurrently. You must supply at least two formulas.

Examples

Loading data faster

1. Create an app, and add four data sources from Microsoft Dataverse, SQL Server, or SharePoint.

   This example uses four tables from the sample Adventure Works database on SQL Azure. After you create the database, connect to it from Power Apps using the fully qualified server name (for example, srvname.database.windows.net):



2. Add a **Button** control, and set its **OnSelect** property to this formula:

Power Apps

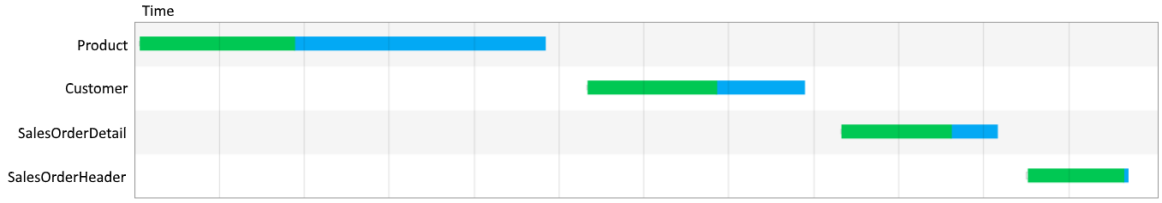
```
ClearCollect( Product, '[SalesLT].[Product]' );
ClearCollect( Customer, '[SalesLT].[Customer]' );
ClearCollect( SalesOrderDetail, '[SalesLT].[SalesOrderDetail]' );
ClearCollect( SalesOrderHeader, '[SalesLT].[SalesOrderHeader]' )
```

3. In Microsoft Edge or Google Chrome, turn on developer tools to monitor network traffic while your app is running.

4. (optional) Turn on network throttling to exaggerate the effects of this comparison.

5. While holding down the Alt key, select the button, and then watch the network traffic.

   The tools show four requests performed in series, similar to this example. Actual times have been removed as they will vary wildly. The graph shows that each call starts after the last has finished:



6. Save, close, and reopen the app.

   Power Apps caches data, so selecting the button again won't necessarily cause four new requests. Each time you want to test performance, close and reopen your app. If you turned network throttling on, you may want to turn it off until you're ready for another test.

7. Add a second **Button** control, and set its **OnSelect** property to this formula:

   Power Apps

   ```
   Concurrent(
       ClearCollect( Product, '[SalesLT].[Product]' ),
       ClearCollect( Customer, '[SalesLT].[Customer]' ),
       ClearCollect( SalesOrderDetail, '[SalesLT].[SalesOrderDetail]' ),
       ClearCollect( SalesOrderHeader, '[SalesLT].[SalesOrderHeader]' )
   )
   ```

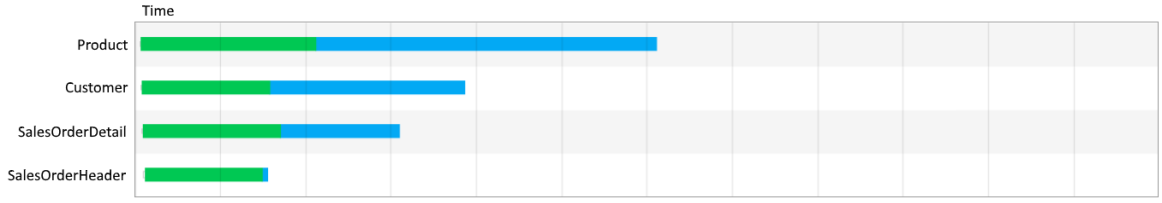
   Note that you added the same **ClearCollect** calls to the first button, but they're wrapped in a **Concurrent** function and separated by commas this time.

8. Clear the network monitor in the browser.

9. If you were using network throttling before, turn it on again.

10. While holding down the Alt key, select the second button, and then watch the network traffic.

    The tools show four requests performed concurrently, similar to this example. Again, actual times have been removed as they will vary wildly. The graph shows that all the calls start at about the same time and do not wait for the previous one to finish:



    These graphs are based on the same scale. By using **Concurrent**, you halved the total amount of time these operations took to finish.

11. Save, close, and reopen the app.

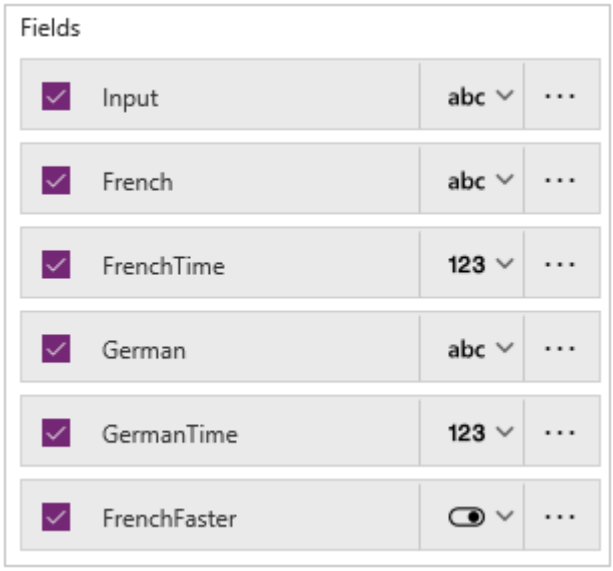
Race condition

1. Add a connection to the Microsoft Translator service to your app.
2. Add a **Text input** control, and rename it **TextInput1** if it has a different name.
3. Add a **Button** control, and set its **OnSelect** property to this formula:

```
Power Apps
```

```
Set( StartTime, Value( Now() ) );
Concurrent(
    Set( FRTrans, MicrosoftTranslator.Translate( TextInput1.Text, "fr"
) );
        Set( FRTransTime, Value( Now() ) ),
    Set( DETrans, MicrosoftTranslator.Translate( TextInput1.Text, "de"
) );
        Set( DETransTime, Value( Now() ) )
);
Collect( Results,
    {
        Input: TextInput1.Text,
        French: FRTrans, FrenchTime: FRTransTime - StartTime,
        German: DETrans, GermanTime: DETransTime - StartTime,
        FrenchFaster: FRTransTime < DETransTime
    }
)
```

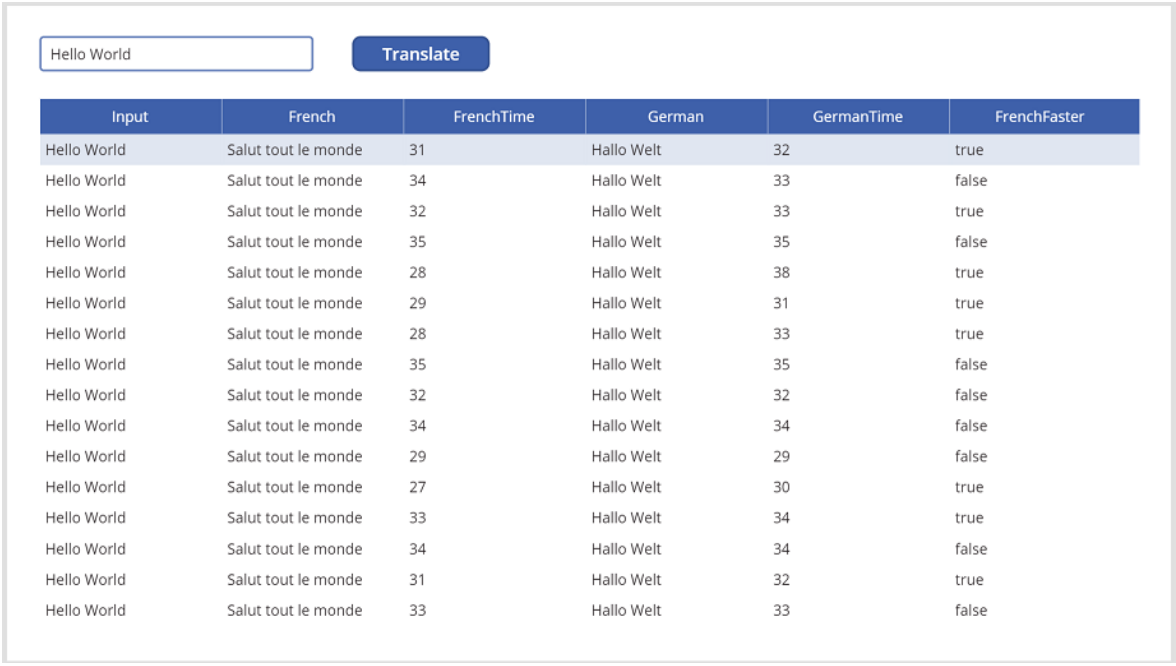
4. Add a **Data table** control, and set its **Items** property to **Results**.
5. On the **Properties** tab of the right pane, select **Edit fields** to open the **Fields** pane.
6. In the list of fields, select the check box for each field to show them all in the data table.
7. (optional) Drag the **Input** field to the top of the list, and drag the **FrenchFaster** field to the bottom of the list.



8. In the **Text input** control, type or paste a phrase to translate.

9. While holding down the Alt key, select the button multiple times to fill the table.

   The times are shown in milliseconds.



Input	French	FrenchTime	German	GermanTime	FrenchFaster
Hello World	Salut tout le monde	31	Hallo Welt	32	true
Hello World	Salut tout le monde	34	Hallo Welt	33	false
Hello World	Salut tout le monde	32	Hallo Welt	33	true
Hello World	Salut tout le monde	35	Hallo Welt	35	false
Hello World	Salut tout le monde	28	Hallo Welt	38	true
Hello World	Salut tout le monde	29	Hallo Welt	31	true
Hello World	Salut tout le monde	28	Hallo Welt	33	true
Hello World	Salut tout le monde	35	Hallo Welt	35	false
Hello World	Salut tout le monde	32	Hallo Welt	32	false
Hello World	Salut tout le monde	34	Hallo Welt	34	false
Hello World	Salut tout le monde	29	Hallo Welt	29	false
Hello World	Salut tout le monde	27	Hallo Welt	30	true
Hello World	Salut tout le monde	33	Hallo Welt	34	true
Hello World	Salut tout le monde	34	Hallo Welt	34	false
Hello World	Salut tout le monde	31	Hallo Welt	32	true
Hello World	Salut tout le monde	33	Hallo Welt	33	false

In some cases, the French translation is faster than the German translation, and vice versa. Both start at the same time, but one returns before the other for a variety of reasons, including network latency and server-side processing.

A race condition would occur if the app depended on one translation ending first. Fortunately, Power Apps flags most timing dependencies that it can detect.

Acceleration, App, Compass, Connection, and Location signals in Power Apps

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Returns information about the app's environment, such as where the user is located in the world and which screen is displayed.

Description and syntax

Signals are values that can change at any time, independent of how the user may be interacting with the app. Formulas that are based on signals automatically recalculate as these values change.

Signals typically return a record of information. You can use and store this information as a record, or you can extract individual properties by using the . operator.

> **Note**
>
> The **Acceleration** and **Compass** functions return accurate values in a native player such as on iOS or Android, but those functions return zero values as you create or modify an app in the browser.

Acceleration

The **Acceleration** signal returns the device's acceleration in three dimensions relative to the device's screen. Acceleration is measured in *g* units of 9.81 m/second^2 or 32.2 ft/second^2 (the acceleration that the Earth imparts to objects at its surface due to gravity).

Expand table

Property	Description
Acceleration.X	Right and left. Right is a positive number.
Acceleration.Y	Forward and back. Forward is a positive number.

Property	Description
Acceleration.Z	Up and down. Up is a positive number.

App

Among other properties, the **App** object includes a signal that indicates which screen is showing.

Expand table

Property	Description
App.ActiveScreen	Screen that's showing. Returns a screen object, which you can use to reference properties of the screen or compare to another screen to determine which screen is showing. You can use the **Back** or **Navigate** function to change the screen that's showing.

More information: **App** object documentation.

Compass

The **Compass** signal returns the compass heading of the top of the screen. The heading is based on magnetic north.

Expand table

Property	Description
Compass.Heading	Heading in degrees. Returns a number 0 to 360, and 0 is north.

Connection

The **Connection** signal returns the information about the network connection. When on a metered connection, you may want to limit how much data you send or receive over the network. Or, if your app is enabled for offline, you may want to display different information depending on the current sync state.

Expand table

Property	Description
Connection.Connected	Returns a Boolean **true** or **false** value that indicates whether the device is connected to a network.
Connection.Metered	Returns a Boolean **true** or **false** value that indicates whether the connection is metered.
Connection.Sync	Returns an enum named **ConnectionSync** which holds the current synchronization state for an app that is enabled for offline.

Below are the enums values available for Connection.Sync.

Expand table

Enum value	Description
Connected	Your device is connected to the network and your app is ready to work offline
ConnectedWithWarning	Your device is connected with a warning.
ConnectedPendingUpsync	Some data on your device must be synchronized with the service.
ConnectedError	There was an error on the last synchronization attempt.
ConnectedRefresh	Your app is currently synchronizing data with the service.
NotConnected	Your device isn't connected to the service.
NotConnectedWithWarning	The last synchronization warning.
NotConnectedPendingUpsync	There are updates on your device waiting to be synchronized.
NotConnectedSyncError	The last synchronization error.

Location

The **Location** signal returns the location of the device based on the Global Positioning System (GPS) and other device information, such as cell-tower communications and IP address.

When a user accesses the location information for the first time, the device may prompt that user to allow access to this information.

As the location changes, dependencies on the location will continuously recalculate, which will consume power from the device's battery. To conserve battery life, you can

use the **Enable** and **Disable** functions to turn location updates on and off. Location is automatically turned off if the displayed screen doesn't depend on location information.

Expand table

Property	Description
Location.Altitude	Returns a number that indicates the altitude, measured in meters, above sea level.
Location.Latitude	Returns a number, from –90 to 90, that indicates the latitude, as measured in degrees from the equator. A positive number indicates a location that's north of the equator.
Location.Longitude	Returns a number, from –180 to 180, that indicates the longitude, as measured in degrees from Greenwich, England. A positive number indicates a location that's east of Greenwich.

Examples

In a baseball field, a pitcher throws a phone from the pitcher's mound to a catcher at home plate. The phone is lying flat with respect to the ground, the top of the screen is pointed at the catcher, and the pitcher adds no spin. At this location, the phone has cellular network service that's metered but no WiFi. The **PlayBall** screen is displayed. The app is enabled for offline and is connected to the service.

Expand table

Formula	Description	Result
Location.Latitude	Returns the latitude of the current location. The field is located at map coordinates 47.591 N, 122.333 W.	47.591 The latitude will change continuously as the ball moves between the pitcher and the catcher.
Location.Longitude	Returns the longitude of the current location.	122.333 The longitude will change continuously as the ball moves between the pitcher and the catcher.
Location	Returns the latitude and longitude of the current location, as a record.	{ Latitude: 47.591, Longitude: 122.333 }

Formula	Description	Result
Compass.Heading	Returns the compass heading of the top of the screen. At this field, home plate is roughly southwest from the pitcher's mound.	230.25
Acceleration.X	Returns the acceleration of the device side to side. The pitcher is throwing the phone straight ahead with respect to the screen's top, so the device isn't accelerating side to side.	0
Acceleration.Y	Returns the acceleration of the device front to back. The pitcher initially gives the device a large acceleration when throwing the device, going from 0 to 90 miles per hour (132 feet per second) in half a second. After the device is in the air, ignoring air friction, the device doesn't accelerate further. The device decelerates when the catcher catches it, bringing it to a stop.	8.2, while the pitcher throws the device. 0, while the device is in the air. -8.2, as the catcher catches the device.
Acceleration.Z	Returns the acceleration of the device top to bottom. While in the air, the device experiences the effects of gravity.	0, before the pitcher throws the device. 1, while the device is in the air. 0, after the catcher catches the device.
Acceleration	Returns the acceleration as a record.	{ X: 0, Y: 264, Z: 0 } as the pitcher throws the device.
Connection.Connected	Returns a Boolean value that indicates whether the device is connected to a network	**true**
Connection.Metered	Returns a Boolean value that indicates whether the connection is metered	**true**
Connection.Sync	Returns an enum value that indicates current sync state.	**ConnectionSync.Connected**
App.ActiveScreen = PlayBall	Returns a Boolean value that indicates whether **PlayBall** is	**true**

Formula	Description	Result
	displayed.	
App.ActiveScreen.Fill	Returns the background color for the displayed screen.	**Color.Green**

Count, CountA, CountIf, and CountRows functions

Article • 04/20/2025

Applies to: Canvas apps Cards Copilot Studio Desktop flows Model-driven apps Power Platform CLI Dataverse functions Power Pages

Counts all records in a table, or counts all records that satisfy a condition.

Description

The **Count** function counts the number of records that contain a number in a single-column table.

The **CountA** function counts the number of records that aren't *blank* in a single-column table. This function includes empty text ("") in the count.

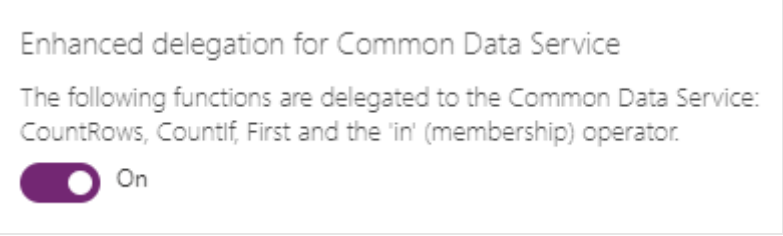
The **CountIf** function counts the number of records in a table that are **true** for a logical formula. The formula can reference columns of the table.

The **CountRows** function counts the number of records in a table.

Each of these functions returns a number.

You need to enable the **Enhanced delegation for Microsoft Dataverse** option in the advance settings to make **CountIf** and **CountRows** functions delegation to work. To enable the option:

1. Open the app where you want to use the functions.
2. Select **Settings** > **Upcoming features** > **Preview**.
3. Turn on the **Enhanced delegation for Microsoft Dataverse** option.



> **Important**
>
> - There is a 50K delegation limit if you use `CountRows` and `CountIf` functions with filters. There is no hard limit on the `CountRows` function when extracted directly from

the data source because of the cached count that Dataverse keeps.

- If the `CountRows(<Data Source>)` function is used without filtering, the count might not be 100% accurate, because the cached count updates periodically. If you need precise count and expect the result to be under the aggregate limit, you can bypass the cached count via `CountIf(<Data Source>, True)`.

Syntax

Count(*SingleColumnTable*)
CountA(*SingleColumnTable*)

- *SingleColumnTable* - Required. Column of records to count.

CountIf(*Table*, *LogicalFormula*)

- *Table* - Required. Table of records to count.
- *LogicalFormula* - Required. Formula to evaluate for each record of the table. Records that return **true** for this formula are counted. The formula can reference columns of the table.

CountRows(*Table*)

- *Table* - Required. Table of records to count.

Example

1. Import or create a collection named **Inventory**, as the first subprocedure in Show images and text in a gallery describes.

2. Add a label, and set its **Text** property to this formula:

   CountIf(Inventory, UnitsInStock < 30)

   The label shows **2** because two products (Ganymede and Callisto) have fewer than 30 units in stock.

3. Add another label, and set its **Text** property to this formula:

   CountA(Inventory.UnitsInStock)

   The label shows **5**, the number of non-empty cells in the **UnitsInStock** column.

4. Add another label, and set its **Text** property to this formula:

CountRows(Inventory)

The label shows **5** because the collection contains five rows.

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Count, CountA, CountIf, and CountRows functions

Article • 04/20/2025

Applies to: Canvas apps Cards Copilot Studio Desktop flows Model-driven apps Power Platform CLI Dataverse functions Power Pages

Counts all records in a table, or counts all records that satisfy a condition.

Description

The **Count** function counts the number of records that contain a number in a single-column table.

The **CountA** function counts the number of records that aren't *blank* in a single-column table. This function includes empty text ("") in the count.

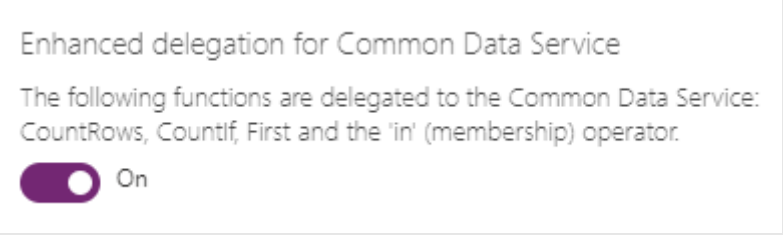
The **CountIf** function counts the number of records in a table that are **true** for a logical formula. The formula can reference columns of the table.

The **CountRows** function counts the number of records in a table.

Each of these functions returns a number.

You need to enable the **Enhanced delegation for Microsoft Dataverse** option in the advance settings to make **CountIf** and **CountRows** functions delegation to work. To enable the option:

1. Open the app where you want to use the functions.
2. Select **Settings** > **Upcoming features** > **Preview**.
3. Turn on the **Enhanced delegation for Microsoft Dataverse** option.



> **Important**
>
> - There is a 50K delegation limit if you use `CountRows` and `CountIf` functions with filters. There is no hard limit on the `CountRows` function when extracted directly from

the data source because of the cached count that Dataverse keeps.

- If the `CountRows(<Data Source>)` function is used without filtering, the count might not be 100% accurate, because the cached count updates periodically. If you need precise count and expect the result to be under the aggregate limit, you can bypass the cached count via `CountIf(<Data Source>, True)`.

Syntax

Count(*SingleColumnTable*)
CountA(*SingleColumnTable*)

- *SingleColumnTable* - Required. Column of records to count.

CountIf(*Table*, *LogicalFormula*)

- *Table* - Required. Table of records to count.
- *LogicalFormula* - Required. Formula to evaluate for each record of the table. Records that return **true** for this formula are counted. The formula can reference columns of the table.

CountRows(*Table*)

- *Table* - Required. Table of records to count.

Example

1. Import or create a collection named **Inventory**, as the first subprocedure in Show images and text in a gallery describes.

2. Add a label, and set its **Text** property to this formula:

   CountIf(Inventory, UnitsInStock < 30)

   The label shows **2** because two products (Ganymede and Callisto) have fewer than 30 units in stock.

3. Add another label, and set its **Text** property to this formula:

   CountA(Inventory.UnitsInStock)

   The label shows **5**, the number of non-empty cells in the **UnitsInStock** column.

4. Add another label, and set its **Text** property to this formula:

CountRows(Inventory)

The label shows **5** because the collection contains five rows.

Count, CountA, CountIf, and CountRows functions

Article • 04/20/2025

Applies to: Canvas apps Cards Copilot Studio Desktop flows Model-driven apps Power Platform CLI Dataverse functions Power Pages

Counts all records in a table, or counts all records that satisfy a condition.

Description

The **Count** function counts the number of records that contain a number in a single-column table.

The **CountA** function counts the number of records that aren't *blank* in a single-column table. This function includes empty text ("") in the count.

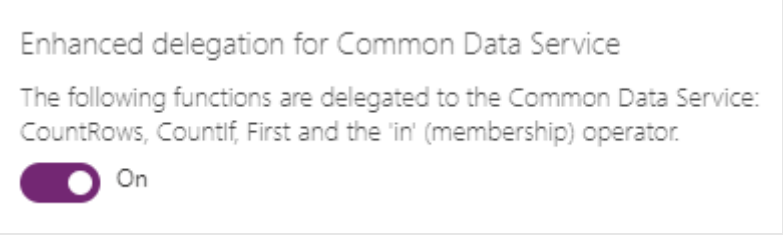
The **CountIf** function counts the number of records in a table that are **true** for a logical formula. The formula can reference columns of the table.

The **CountRows** function counts the number of records in a table.

Each of these functions returns a number.

You need to enable the **Enhanced delegation for Microsoft Dataverse** option in the advance settings to make **CountIf** and **CountRows** functions delegation to work. To enable the option:

1. Open the app where you want to use the functions.
2. Select **Settings** > **Upcoming features** > **Preview**.
3. Turn on the **Enhanced delegation for Microsoft Dataverse** option.



> **Important**
>
> - There is a 50K delegation limit if you use `CountRows` and `CountIf` functions with filters. There is no hard limit on the `CountRows` function when extracted directly from

the data source because of the cached count that Dataverse keeps.

- If the `CountRows(<Data Source>)` function is used without filtering, the count might not be 100% accurate, because the cached count updates periodically. If you need precise count and expect the result to be under the aggregate limit, you can bypass the cached count via `CountIf(<Data Source>, True)`.

Syntax

Count(*SingleColumnTable*)
CountA(*SingleColumnTable*)

- *SingleColumnTable* - Required. Column of records to count.

CountIf(*Table*, *LogicalFormula*)

- *Table* - Required. Table of records to count.
- *LogicalFormula* - Required. Formula to evaluate for each record of the table. Records that return **true** for this formula are counted. The formula can reference columns of the table.

CountRows(*Table*)

- *Table* - Required. Table of records to count.

Example

1. Import or create a collection named **Inventory**, as the first subprocedure in Show images and text in a gallery describes.

2. Add a label, and set its **Text** property to this formula:

   CountIf(Inventory, UnitsInStock < 30)

   The label shows **2** because two products (Ganymede and Callisto) have fewer than 30 units in stock.

3. Add another label, and set its **Text** property to this formula:

   CountA(Inventory.UnitsInStock)

   The label shows **5**, the number of non-empty cells in the **UnitsInStock** column.

4. Add another label, and set its **Text** property to this formula:

CountRows(Inventory)

The label shows **5** because the collection contains five rows.

Count, CountA, CountIf, and CountRows functions

Article • 04/20/2025

Applies to: Canvas apps Cards Copilot Studio Desktop flows Model-driven apps Power Platform CLI Dataverse functions Power Pages

Counts all records in a table, or counts all records that satisfy a condition.

Description

The **Count** function counts the number of records that contain a number in a single-column table.

The **CountA** function counts the number of records that aren't *blank* in a single-column table. This function includes empty text ("") in the count.

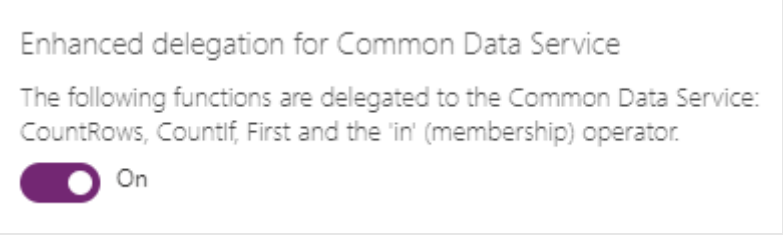
The **CountIf** function counts the number of records in a table that are **true** for a logical formula. The formula can reference columns of the table.

The **CountRows** function counts the number of records in a table.

Each of these functions returns a number.

You need to enable the **Enhanced delegation for Microsoft Dataverse** option in the advance settings to make **CountIf** and **CountRows** functions delegation to work. To enable the option:

1. Open the app where you want to use the functions.
2. Select **Settings** > **Upcoming features** > **Preview**.
3. Turn on the **Enhanced delegation for Microsoft Dataverse** option.



> **Important**
>
> - There is a 50K delegation limit if you use `CountRows` and `CountIf` functions with filters. There is no hard limit on the `CountRows` function when extracted directly from

the data source because of the cached count that Dataverse keeps.

- If the `CountRows(<Data Source>)` function is used without filtering, the count might not be 100% accurate, because the cached count updates periodically. If you need precise count and expect the result to be under the aggregate limit, you can bypass the cached count via `CountIf(<Data Source>, True)`.

Syntax

Count(*SingleColumnTable*)
CountA(*SingleColumnTable*)

- *SingleColumnTable* - Required. Column of records to count.

CountIf(*Table*, *LogicalFormula*)

- *Table* - Required. Table of records to count.
- *LogicalFormula* - Required. Formula to evaluate for each record of the table. Records that return **true** for this formula are counted. The formula can reference columns of the table.

CountRows(*Table*)

- *Table* - Required. Table of records to count.

Example

1. Import or create a collection named **Inventory**, as the first subprocedure in Show images and text in a gallery describes.

2. Add a label, and set its **Text** property to this formula:

   CountIf(Inventory, UnitsInStock < 30)

   The label shows **2** because two products (Ganymede and Callisto) have fewer than 30 units in stock.

3. Add another label, and set its **Text** property to this formula:

   CountA(Inventory.UnitsInStock)

   The label shows **5**, the number of non-empty cells in the **UnitsInStock** column.

4. Add another label, and set its **Text** property to this formula:

CountRows(Inventory)

The label shows **5** because the collection contains five rows.

Data types

06/19/2025

Information flows through Power Fx in small, discrete values, like the cells of a spreadsheet. For example, data in a **Birthday** field and an **Anniversary** field both flow through as a **Date** value that includes the year, month, and day. Power Fx formats these values, constrains input to what's appropriate for each, and shares the values with a database. Birthdays and anniversaries might differ to people, but the system handles them in the same way. In this case, **Date** is an example of a data type.

This article explains each data type in Power Fx, how they work, and how they map to external data sources.

Expand table

Data type	Description	Examples
Boolean	A *true* or *false* value. You can use it directly in **If**, **Filter**, and other functions without a comparison.	*true*
Choice	A choice from a set of options, backed by a number. This data type combines a localizable text label with a numeric value. The label appears in the app, and the numeric value is stored and used for comparisons. The **Type** function supports this data type if you use an instance of a **Choice** field by name.	**ThisItem.OrderStatus**
Color	A color specification that includes an alpha channel.	**Color.Red** **ColorValue("#102030")** **RGBA(255, 128, 0, 0.5)**
Currency	A currency value stored in a floating-point number. Currency values are the same as number values with currency-formatting options. The **Type** function doesn't support the **Currency** data type.	**123** **4.56**
Date	A date without a time, in the app user's time zone.	**Date(2019, 5, 16)**
DateTime	A date with a time, in the app user's time zone.	**DateTimeValue("May 16, 2019 1:23:09 PM")**

Data type	Description	Examples
Decimal	A number with high precision, base 10 operations, and a limited range.	**123** **Decimal("1.2345")**
Dynamic (formerly **UntypedObject**)	The value's type is dynamic and can vary at runtime. A Dynamic value could be any existing type, and can be converted into compatible types using functions such as Boolean(), Value(), Table(), etc. For more information, see Dynamic values and Working with JSON.	**ParseJSON("{ ""Field"" : 1234 }").Field**
Float	A number with standard precision, base 2 operations, and a wide range.	**123** **8.903e121** **1.234e-200**
GUID	A globally unique identifier.	**GUID()** **GUID("123e4567-e89b-12d3-a456-426655440000")**
Hyperlink	A text string that contains a hyperlink.	**"https://powerapps.microsoft.com"**
Image	A universal resource identifier (URI) text string to an image in .jpeg, .png, .svg, .gif, or another common web-image format. The **Type** function doesn't support the **Image** data type.	**MyImage** added as an app resource **"https://northwindtraders.com/logo.jpg"** **"appres://blobmanager/7b12ffa2..."**
Media	A URI text string to a video or audio recording. The **Type** function doesn't support the **Media** data type.	**MyVideo** added as an app resource **"https://northwindtraders.com/intro.mp4"** **"appres://blobmanager/3ba411c..."**
Number	An alias for **Decimal** (most Power Fx hosts) or **Float** (Canvas apps). If you can use either variety of number for a situation, use **Number** for maximum compatibility.	**123** **0.0123** **1e4**
Record	A record of data values. This compound data type contains instances of other data types listed in this article. For more information, see Working with tables. The **Type** function supports this data type if you use an instance of a **Record**.	**{ Company: "Northwind Traders",** **Staff: 35,** **NonProfit: false }**
Record reference	A reference to a record in a table. These references are often used with polymorphic lookups. For more	**First(Accounts).Owner**

Data type	Description	Examples
	information, see Working with references. The **Type** function doesn't support this data type.	
Table	A table of records. All records must have the same field names and data types, and omitted fields are treated as *blank*. This compound data type contains instances of other data types listed in this article. For more information, see Working with tables. The **Type** function supports this data type if you use an instance of a **Table**.	**Table({ FirstName: "Sidney", LastName: "Higa" }, { FirstName: "Nancy", LastName: "Anderson" })**
Text	A Unicode text string.	**"Hello, World"**
Time	A time without a date, in the app user's time zone.	**Time(11, 23, 45)**
Void	Used only by behavior user defined functions, it indicates that a function doesn't have a return type. The **Type** function doesn't support this data type. Even though a function doesn't have a return type or value, it can always return an error.	**Hi(): Void = { Notify("Hello!") }**
Yes/No	A choice from a set of two options, backed by a boolean value. This data type combines a localizable text label with a boolean value. The label appears in the app, and the boolean value is stored and used for comparisons. The **Type** function supports this data type if you use an instance of a **Yes/No** field by name.	**ThisItem.Taxable**

Many of these data types are similar and have the same underlying representation, like a **Hyperlink** field being treated as **Text**. The extra data types give better default experiences in forms and other controls.

Blank

All data types can have a value of *blank* (no value). The term "null" is often used in databases for this concept.

Use the **Blank** function with the **Set** or **Patch** function to set a variable or field to *blank*. For example, **Set(x, Blank())** removes the value in the global variable **x**.

Test for a *blank* value with the **IsBlank** function. Replace possible *blank* values with non-*blank* values by using the **Coalesce** function.

Because all data types support *blank*, the **Boolean** and **Two option** data types have three possible values.

Text, Hyperlink, Image, and Media

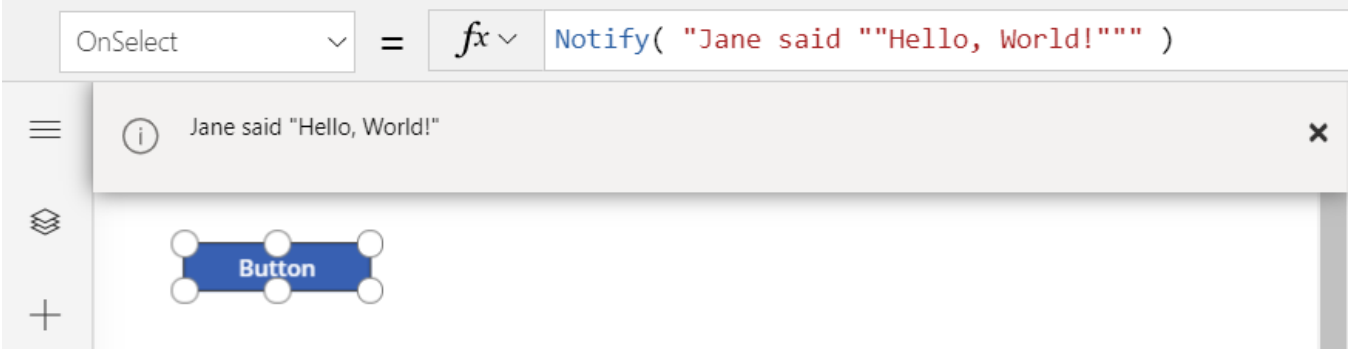
All four of these data types are based on a Unicode text string.

Embedded text

Embedded text strings in a formula are enclosed in double quotation marks. Use two double quotes together to represent a single double quote in the text string. For example, using the following formula in the **OnSelect** property of a **Button** control:

```
Power Fx

Notify( "Jane said ""Hello, World!""" )
```

Results in a banner when the button is pressed, where the first and last double quotes are omitted (as they delimit the text string) and the repeated double quotes around **Hello, World!** are replaced with a single double quote:



Single quotation marks are used for identifier names that contain special characters and have no special significance within a text string.

String interpolation

Use string interpolation to embed formulas within a text string. This approach is often easier to work with and visualize the output than using the **Concatenate** function or **&** operator.

Prefix the text string with a dollar sign **$** and enclose the formula to be embedded with curly braces **{ }**. To include a curly brace in the text string, use repeated curly braces: **{{** or **}}**. String interpolation can be used anywhere a standard text string can be used.

For example, consider this formula with global variables **Apples** set to 3 and **Bananas** set to 4:

```
Power Fx
```

```
$"We have {Apples} apples, {Bananas} bananas, yielding {Apples+Bananas} fruit
total."
```

This formula returns the text string **We have 3 apples, 4 bananas, yielding 7 fruit total.** The variables **Apples** and **Bananas** are inserted in the text replacing the curly braces, along with the result of the mathematical formula **Apples+Bananas**. Spaces and other characters around the curly braces are preserved as they are.

Embedded formulas can include any functions or operators. All that is required is that the result of the formula can be coerced to a text string. For example, this formula inserts **NickName** if it's supplied, or the **FirstName** if not, in a greeting:

```
Power Fx
```

```
$"Welcome {Coalesce( NickName, FirstName )}, it's great to meet you!"
```

If **NickName** is set to "Joe", then this formula produces the text string **Welcome Joe, it's great to meet you!**. But if **NickName** is *blank* and **FirstName** is "Joseph", then this formula produces **Dear Joseph, great to meet you!** instead.

String interpolation can include standard text strings in the embedded formula. For example, if neither **NickName** nor **FirstName** were supplied, we could still provide **"Friend"** as a substitute:

```
Power Fx
```

```
$"Welcome {Coalesce( NickName, FirstName, "Friend" )}!"
```

String interpolations can even be nested. Consider this example where **First**, **Middle**, and **Last** names are combined into a greeting. Even if one or two of these values are *blank*, the correct number of spaces are maintained between the name parts. If none of the parts are provided, the inner string interpolation will collapse to an empty string, and be replaced by the **Coalesce** function by "Friend".

```
Power Fx
```

```
$"Welcome {Coalesce( Trim( $"{First} {Middle} {Last}"}), "Friend" )}!"
```

Expand table

First	Middle	Last	Result
John	Qunicy	Doe	`Welcome John Quincy Doe!`
John	*blank*	Doe	`Welcome John Doe!`
blank	*blank*	Doe	`Welcome Doe!`
blank	*blank*	*blank*	`Welcome Friend!`

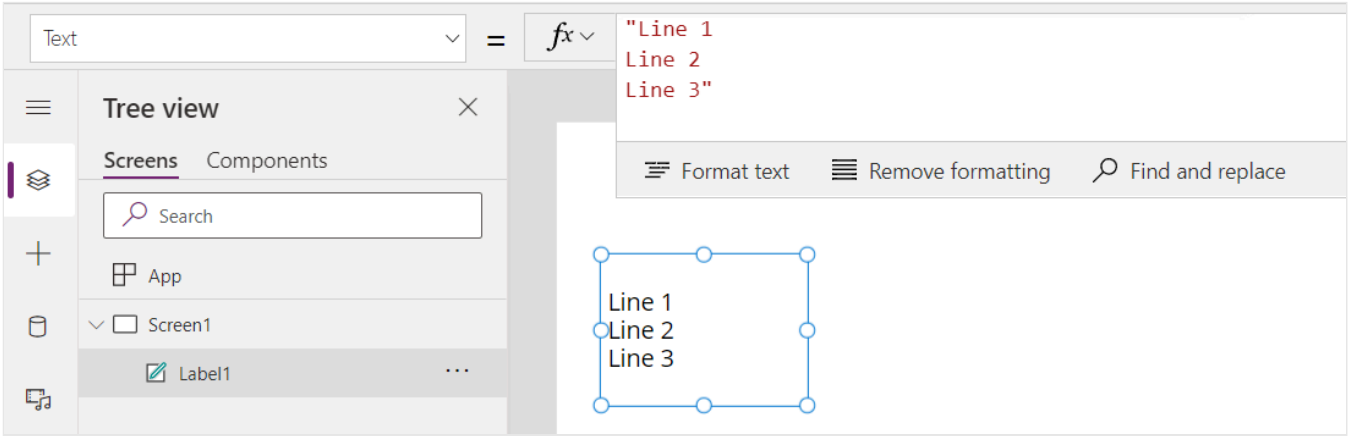
Newlines

Embedded text strings can contain newlines. For example, consider setting the **Text** property of a **Label** control to the following:

Power Fx

```
"Line 1
Line 2
Line 3"
```

This formula results in three lines shown in the label control:



Newlines are also supported with string interpolation:

Power Fx

```
$"Line {1}
Line {1+1}
Line {1+1+1}"
```

Which results in the same output:

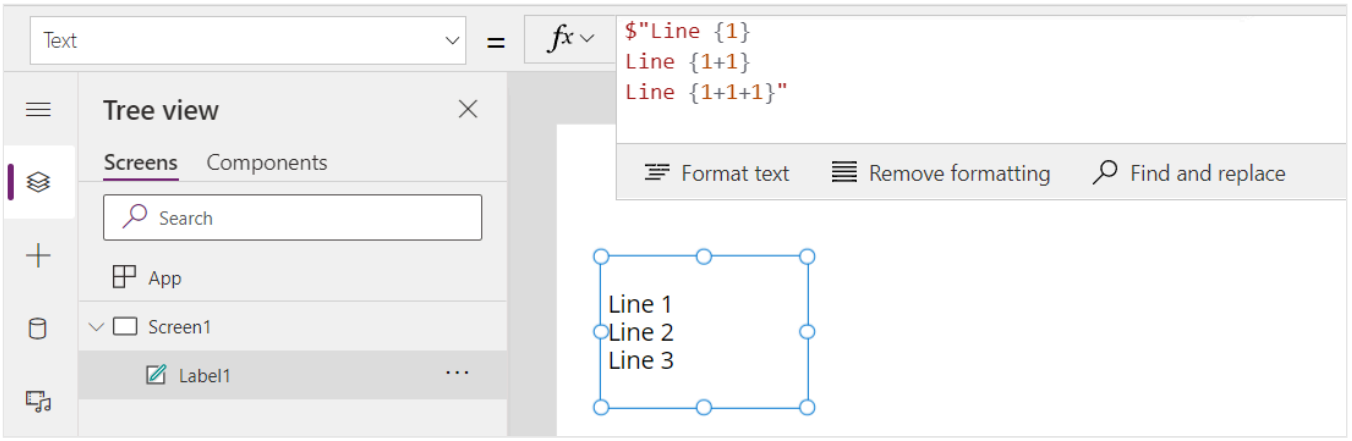
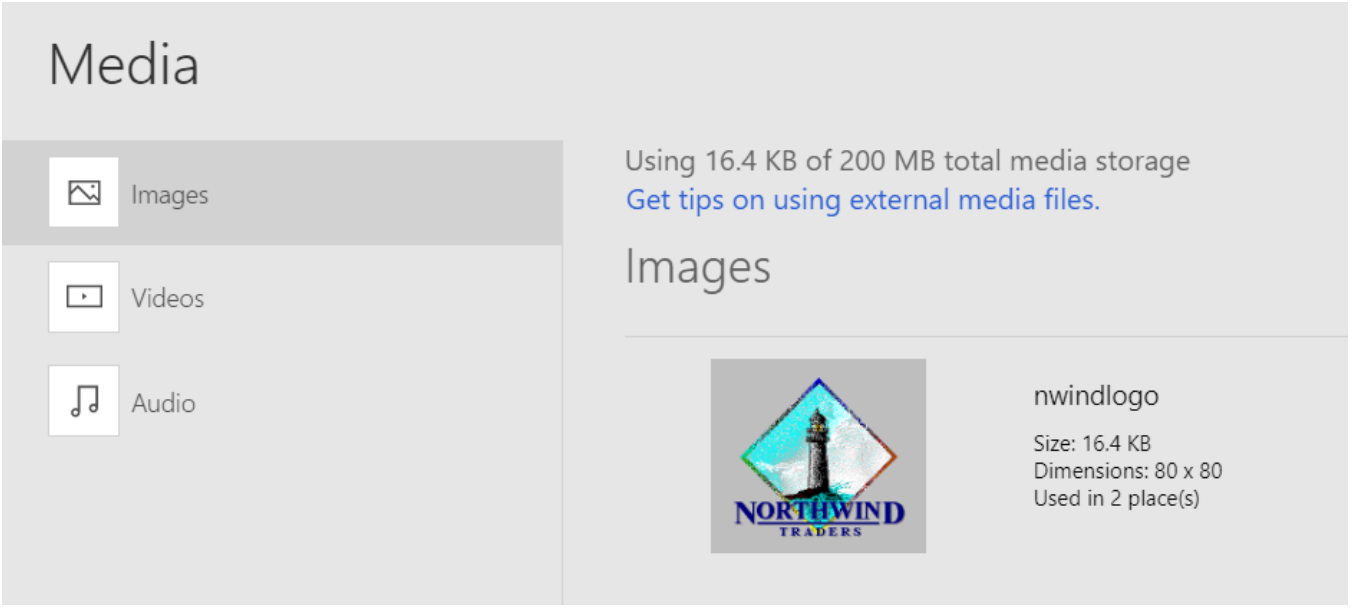


Image and Media resources

Through the **File** menu, you can add image, video, and audio files as app resources. The name of the imported file becomes the resource name in the app. In this graphic, the Northwind Traders logo, which is named **nwindlogo**, has been added to the app:



To use this resource in an app, specify it in the **Image** property of an **Image** control:



URIs for images and other media

You can dig a little deeper into that last example by setting the **Text** property of a **Label** control to **nwindlogo**. The label shows a text string:



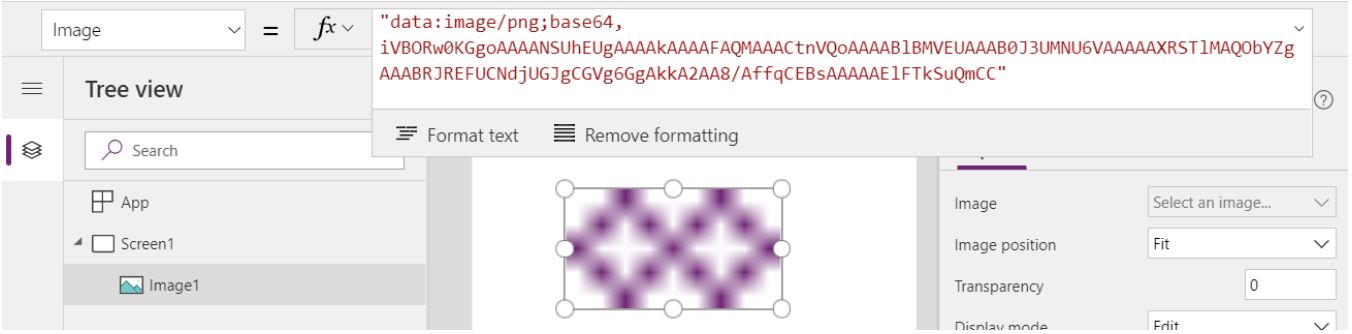
Canvas apps reference each image or other media file, whether it's in the cloud or added as an app resource, by a URI text string.

For example, the **Image** property of an image control accepts not only app resources but also links to images on the web, such as "https://northwindtraders.com/logo.jpg". The property also accepts inline images that use the data URI scheme, as in this example:

Power Fx

```
""
```

That URI displays a scaled-up version of two purple diamonds:



You can show the most recent image captured in a **Camera** control if you set the **Image** property of an image control to the **Photo** property of the camera control. The app holds the image in memory, and the **Photo** property of the camera control returns a URI reference to the image. For example, you might take a picture, and the camera's **Photo** property could return **"appres://blobmanager/7b12ffa2ea4547e5b3812cb1c7b0a2a0/1"**.

You use a URI to reference an image or another media file stored in a database. That way, the app doesn't retrieve the actual data until it's needed. For example, an attachment in a Microsoft Dataverse table might return **"appres://datasources/Contacts/table/..."** As in the camera example, you can display this image by setting the **Image** property of an image control to this reference, which retrieves the binary data.

When you save a media data type, such as an image, to a database, the app sends the actual image or media data, not the URI reference.

Size limits

As text strings and URIs, these data types have no preset limit on their length.

The binary data that these data types reference also has no preset limit on size. For example, an image captured through the camera control that's referenced as **"appres://..."** can be as large and high resolution as the device's camera can muster. The resolution, frame rate, and other attributes of media files aren't limited by the data type, but specific controls for playing and capturing media may have their own limitations.

However, all data sizes are subject to the amount of available memory in the app. Browsers running on a desktop computer typically support more than 100 megabytes of data. However, the amount of available memory on a device such as a phone might be far lower, typically in the range 30-70 megabytes. To determine whether your app runs within these limits, test common scenarios on all devices on which it should run.

As a best practice, hold data in memory only as long as necessary. Upload images to a database as soon as you can; download images only when the app's user requests them.

Numbers

> **Note**
>
> Power Apps only supports **Float** today and it's the type of all numbers. **Decimal** support will be added soon.

Power Fx supports two kinds of numbers: **Decimal** and **Float** (with synonyms **Number** and **Currency**).

Decimal is best for most business calculations. It can accurately represent numbers in base 10 meaning that `0.1` can be exactly represented and will avoid rounding errors during calculations. It has a large enough range for any business need, up to 10^{28} with up to 28 digits

of precision. **Decimal** is the default numeric data type for most Power Fx hosts, used if one simply writes `2*2`.

Float is best for scientific calculations. It can represent numbers in a larger range, up to 10^{308}. Precision is limited to 15 decimal places and math is based on base 2 so it can't represent some common decimal values precisely. **Float** also has higher performance and is favored if that is a factor and precision isn't critical.

Decimal numbers

The **Decimal** data type most often uses the .NET decimal data type. Some hosts, such as Dataverse formula columns that are run in SQL Server, use the SQL Server decimal data type.

Decimal does math the way you learned in school, using base 10 digits, important to avoid rounding errors from very small differences that can accumulate when using base 2 math (as used by **Float**).

The range is from positive 79,228,162,514,264,337,593,543,950,335 to negative 79,228,162,514,264,337,593,543,950,335. The decimal separator can be placed anywhere within these numbers, providing up to 28 digits of precision, and still be precisely represented. For example, 79,228,162,514,264.337593543950335 can be exactly represented, as can 7.9228162514264337593543950335.

Floating point numbers

The **Float** data type, also known as **Number** or **Currency**, uses the IEEE 754 double-precision floating-point standard. This standard provides a very large range of numbers in which to work, from -1.79769×10^{308} to 1.79769×10^{308}. The smallest value that can be represented is 5×10^{-324}.

Float can exactly represent whole numbers (or integers) between –9,007,199,254,740,991 ($-(2^{53} - 1)$) and 9,007,199,254,740,991 ($2^{53} - 1$), inclusive. This range is larger than the 32-bit (or 4-byte) integer data types that databases commonly use. However, canvas apps can't represent 64-bit (or 8-byte) integer data types. You might want to store the number in a text field or use a calculated column to make a copy of the number in a text field, so that it maps into a **Text** data type in the canvas app. In this manner, you can hold, display, and enter these values, and comparing them to determine whether they're equal; however, you can't perform numerical calculations on them in this form.

Floating-point arithmetic is approximate, so it can sometimes give unexpected results with many documented examples. You might expect the formula **55 / 100 * 100** to return exactly 55 and **(55 / 100 * 100) - 55** to return exactly zero. However, the latter formula returns $7.1054 \times 10^{-}$

^{15}, which is very small but not zero. That tiny difference doesn't normally cause a problem, and the app rounds it away when showing the result. However, small differences can compound in subsequent calculations and appear to give the wrong answer.

Database systems often store currencies and perform calculations by using decimal math, which offers a smaller range but greater control over the precision. By default, canvas apps map currencies in and out of floating-point values; therefore, the result might differ from calculations that are done in a native decimal data type. Depending on the precision needs of your app, you might want to work with these values as **Text**, as described for large integers earlier.

Defaults and conversions

> ⓘ **Note**
>
> Power Apps only supports **Float** today and it's the type of all numbers. **Decimal** support will be added soon.

Most Power Fx hosts use **Decimal** by default. Having this default means:

- Literal numbers in formulas. The number `1.234` is interpreted as a **Decimal** value. For example, the formula `1.234 * 2` interprets the `1.234` and `2` as **Decimal** and return a **Decimal** result.
- Value function. `Value( "1.234" )` returns a **Decimal** value. For example, in the formula `Value( "1.234" ) * 2`, the **Value** function interprets the contents of the text string `"1.234"` as a **Decimal**.

To work with **Float** values, the **Float** function is used. Extending our example, `Float( 1.234 )` converts the **Decimal** `1.234` to **Float**. **Float** can also be used as a replacement for **Value** to convert a string containing a floating point number such as `Float( "1.234" )` to a **Float** value, which is required if the number can't be represented as a **Decimal**.

In summary:

Expand table

Usage	Decimal	Float
Literal numbers in formulas	`1.234`	`Float( 1.234 )` `Float( "1.234" )`

Usage	Decimal	Float
Conversion from text string	`Value( "1.234" )` `Decimal( "1.234" )`	`Float( "1.234" )`
Conversion between numeric types	`Decimal( float )`	`Float( decimal )`
Conversion to text string	`Text( decimal )`	`Text( float )`

Mixing numeric types

Float and **Decimal** values can be freely mixed. When mixed, **Decimal** values are converted to **Float** values due to the larger range. As this conversion can result in a loss of precision, it's important to not mix the two needlessly. Since **Decimal** is the default literal data type and most numeric functions preserve the type, it's relatively easy to avoid moving to **Float** without desiring it.

For example, consider the following calculation using `pac power-fx repl` after installing the Power Platform CLI. Since both numbers are **Decimal**, the calculation is done in **Decimal**, and the result retains full precision:

Power Fx

```
>> 1.0000000000000000000000000001 * 2
2.0000000000000000000000000002
```

If instead, the second operand was changed to **Float** then the entire calculation would be done in **Float**, and the tiny fractional part would be lost:

Power Fx

```
>> 1.0000000000000000000000000001 * Float(2)
2
```

Date, Time, and DateTime

Time zones

Date/time values fall in these categories:

- **User local**: These values are stored in UTC (Coordinated Universal Time), but the app user's time zone affects how the app shows these values and how the app user specifies

them. As an example, the same moment appears differently to a user in Canada than it does to a user in Japan.
- **Time zone independent**: The app shows these values the same way and the app user specifies them the same way, regardless of time zone. The same moment appears the same way to a user in Canada as it does to a user in Japan. App authors who don't expect their apps to run in different time zones use these values because they're simpler overall.

This table shows some examples:

Expand table

Date/time type	Value stored in the database	Value displayed and entered 7 hours west of UTC	Value displayed and entered 4 hours east of UTC
User local	Sunday, May 19, 2019 4:00 AM	Saturday, May 18, 2019 9:00 PM	Sunday, May 19, 2019 8:00 AM
Time zone independent	Sunday, May 19, 2019 4:00 AM	Sunday, May 19, 2019 4:00 AM	Sunday, May 19, 2019 4:00 AM

For **User local** date/times, canvas apps use the time zone of the browser or device, but model-driven apps use the user's setting in Dataverse. These settings typically match, but results differ if these settings differ.

Use the DateAdd and TimeZoneInformation functions to convert local time to UTC and back again. See the examples at the end of the documentation for these functions.

Numeric equivalents

Canvas apps hold and calculate all date/time values, whether **User local** or **Time zone independent** in UTC. The app translates the values based on the app user's time zone when showing them and when the app user specifies them.

When a canvas app reads a **Time zone independent** value from a data source or writes such a value to a data source, the app automatically adjusts the value to compensate for the time zone of the app's user. The app then treats the value as a UTC value, consistent with all other date/time values in the app. Because of this compensation, the original **Time zone independent** value appears when the app adjusts the UTC value for the app user's time zone.

You can observe this behavior more closely by using the Value function to access the underlying numerical value for a date/time value. This function returns the date/time value as the number of milliseconds since January 1, 1970 00:00:00.000 UTC.

Because every date/time value is held in UTC, the formula **Value(Date(1970, 1, 1))** won't return zero in most parts of the world because the **Date** function returns a date in UTC. For example, the formula would return 28,800,000 in a time zone that's offset from UTC by eight hours. That number reflects the number of milliseconds in eight hours.

Returning to our example:

Expand table

Date/time type	Value stored in the database	Value displayed and entered 7 hours west of UTC	Value function returns
User local	Sunday, May 19, 2019 4:00 AM	Saturday, May 18, 2019 9:00 PM	1,558,238,400,000 (Sunday, May 19, 2019 4:00 AM UTC)
Time zone independent	Sunday, May 19, 2019 4:00 AM	Sunday, May 19, 2019 4:00 AM	1,558,263,600,000 (Sunday, May 19, 2019 11:00 AM UTC)

Converting Unix times

Unix times reflect the number of seconds since January 1, 1970 00:00:00 UTC. Because canvas apps use milliseconds instead of seconds, you can convert between the two by multiplying or dividing by 1,000.

For example, Unix time shows September 9, 2001, at 01:46:40 UTC as 1,000,000,000. To show that date/time value in a canvas app, multiply that number by 1,000 to convert it to milliseconds, and then use it in a Text function. The formula **Text(1000000000 * 1000, DateTimeFormat.UTC)** returns the string **2001-09-09T01:46:40.000Z**.

However, that function returns **Saturday, September 8, 2001 18:46:40** if you use the **DateTimeFormat.LongDateTime24** format in a time zone that's -7 hours offset from UTC (7 hours west of UTC). This result shows the **DateTime** value correctly based on the local time zone.

To convert to a Unix time, divide the result from **Value** by 1,000:
RoundDown(Value(UnixTime) / 1000, 0)

If you need the Unix time in a **Date** value for further calculations or display within Power Apps, use this formula:
DateAdd(Date(1970,1,1), UnixTime, Seconds)

SQL Server

SQL Server has **Datetime**, **Datetime2**, and other date/time data types that don't include a time-zone offset and don't indicate which time zone they're in. Canvas apps assume these values are stored in UTC and treat them as **User local**. If the values are meant to be time-zone independent, correct for the UTC translations by using the **TimeZoneOffset** function.

Canvas apps use the included time-zone information in **Datetimeoffset** fields when converting a value to the app's internal UTC representation. The apps always use UTC as the time zone (zero time zone offset) when they write data.

Canvas apps read and write values of the **Time** data type in SQL Server as text strings in the ISO 8601 duration format. For example, you must parse this string format and use the **Time** function to convert the text string **"PT2H1M39S"** to a **Time** value:

Power Fx

```
With(
    Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?
<seconds>\d+)S)?" ),
    Time( Value( hours ), Value( minutes ), Value( seconds ) )
)
// Result: 2:01 AM (as shown in a label control, use the Text function to see the
seconds)
```

Mixing date and time information

Date, **Time**, and **DateTime** have different names, but they all hold the same information about dates and times.

A **Date** value can include time information with it, which is usually midnight. A **Time** value can carry date information, which is usually January 1, 1970. Dataverse also stores time information with a **Date Only** field but shows only the date information by default. Similarly, canvas apps sometimes distinguish between these data types to determine default formats and controls.

Adding and subtracting date and time values directly isn't recommended because time-zone and other conversions could cause confusing results. Either use the **Value** function to convert date/time values to milliseconds first and take into account the app user's time zone, or use the **DateAdd** and **DateDiff** functions to add or subtract from one of these values.

Choices and Yes/No

Choices and two-option data types give two or more choices for an app user to select. For example, an **Order Status** choice might offer the choices **New**, **Shipped**, **Invoiced**, and **Closed**.

The two-option data type gives only two choices.

Both of these data types show their labels as text. For example, a label control shows one of the order-status options if the control's **Text** property is set to a formula that references that choice. Option labels can be localized for app users in different locations.

When an app user selects an option and saves the change, the app sends the data to the database, which stores it in a way that's independent of language. An option in a choice is sent and stored as a number, and an option in a two-option data type is sent and stored as a boolean value.

The labels are for display only. You can't compare the labels directly because they're specific to a language. Instead, each choice has an enumeration that works with the underlying number or boolean value. For example, you can't use this formula:

```
If( ThisItem.OrderStatus = "Active", ...
```

But you can use this formula:

```
If( ThisItem.OrderStatus = OrderStatus.Active, ...
```

For global choices (which tables share), the name of the option-set enumeration matches the name of the global choice. For local choices (which are scoped to a table), the name can include the table name. This behavior avoids conflicts if multiple tables have choices with the same name. For example, the **Accounts** table might have an **OrderStatus** choice, and its name might be **OrderStatus (Accounts)**. That name has spaces and parentheses, so you need to surround it with single quotation marks if you reference it in a formula.

Two-option values also behave as boolean values. For example, a two-option value named **TaxStatus** might have the labels **Taxable** and **Non-Taxable**, which correspond to *true* and *false*. To show this, use this formula:

```
If( ThisItem.Taxable = TaxStatus.Taxable, ...
```

You can also use this formula:

```
If( ThisItem.Taxable, ...
```

DataSourceInfo function

Article • 04/11/2024

Applies to: Canvas apps Model-driven apps

Provides information about a data source.

Overview

Data sources can provide a wealth of information to optimize the user experience.

You can use column-level information to validate user input and provide immediate feedback to the user before using the **Patch** function. The **Validate** function uses this same information.

You can use information at the data-source level, for example, to disable or hide **Edit** and **New** buttons for users who don't have permissions to edit and create records.

Data sources vary in how much information they provide, including not providing any at all. Collections provide no information. If a piece of information isn't provided, a default is used, or *blank* is returned.

> **Note**
>
> Currently, the DataSourceInfo function is not supported in Microsoft Lists.

Description

Column information

You can use **DataSourceInfo** to obtain information about a particular column of a data source:

Expand table

Information Argument	Result Type	Description
DataSourceInfo.DisplayName	String	Display name for the column. If no display name is defined, returns the column name.

Information Argument	Result Type	Description
DataSourceInfo.MaxLength	Number	Maximum number of characters that the column can hold. Applies only to columns that contain strings. If a maximum isn't set, returns *blank*.
DataSourceInfo.MaxValue	Number	Maximum numeric value that a column can hold. Applies only to columns that contain numbers. If a maximum isn't set, returns *blank*.
DataSourceInfo.MinValue	Number	Minimum numeric value that a column can hold. Applies only to columns that contain numbers. If a minimum isn't set, returns *blank*.
DataSourceInfo.Required	Boolean	Is a value required for this column? If not set by the data source, returns **false**.

The third argument is the name of a column as a string. For example, column **Phone** in collection **People** would be passed as **"Phone"** including the double quotes.

Data-source information

You can also use **DataSourceInfo** to obtain information about a data source as a whole:

Expand table

Information Argument	Result Type	Description
DataSourceInfo.AllowedValues	Boolean	What types of permissions can users be granted for this data source? If not set by the data source, returns *blank*.
DataSourceInfo.CreatePermission	Boolean	Does the current user have permission to create records in this data source? If not set by the data source, returns **true**.
DataSourceInfo.DeletePermission	Boolean	Does the current user have permission to delete records in this data source? If not set by the data source, returns **true**.
DataSourceInfo.EditPermission	Boolean	Does the current user have permission to edit records in this data source? If not set by the data source, returns **true**.
DataSourceInfo.ReadPermission	Boolean	Does the current user have permission to read records in this data source? If not set by the data

Information Argument	Result Type	Description
		source, returns **true**.

> ⓘ Note
>
> **DataSourceInfo** returns *true* if it cannot determine whether the current user has the requested permission. Permissions will be checked again by the server when the actual operation is carried out and an error is displayed if it was not allowed. At this time, permissions checking with **DataSourceInfo** is only possible when using Microsoft Dataverse.

Syntax

DataSourceInfo(*DataSource*, *Information* [, *ColumnName*])

- *DataSource* – Required. The data source to use.
- *Information* – Required. The type of information that you want to retrieve.
- *ColumnName* – Optional. The column name for which to retrieve column-level information. For information at the data-source level, the *ColumnName* argument can't be used.

> ⓘ Note
>
> In Power Apps prior to version 3.24042, column names were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. Afer this version, all apps were automatically updated to the new syntax described in this article.

Examples

The examples in this section use this data source, named **IceCream**:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200

The data source has also provided this information:

- The display name for **Quantity** is "Quantity on Hand".
- The maximum length of **Flavor** is 30 characters.
- The **Flavor** column must contain a value. The **Quantity** column isn't required.
- The minimum **Quantity** is 0.
- The maximum **Quantity** is 100.
- The current user can read and edit the records of the **IceCream** data source but can't create or delete records.

Expand table

Formula	Description	Result
DataSourceInfo(IceCream, DataSourceInfo.DisplayName, Quantity)	Returns the display name for the **Quantity** column of the **IceCream** data source.	"Quantity on Hand"
DataSourceInfo(IceCream, DataSourceInfo.MaxLength, Flavor)	Returns the maximum length of the string for the **Flavor** column of the **IceCream** data source.	30
DataSourceInfo(IceCream, DataSourceInfo.Required, Flavor)	Is the **Flavor** column of the **IceCream** data source required?	**true**
DataSourceInfo(IceCream, DataSourceInfo.Required, Quantity)	Is the **Quantity** column of the **IceCream** data source required?	**false**
DataSourceInfo(IceCream, DataSourceInfo.MaxValue, Quantity)	Returns the maximum numeric value for the **Quantity** column of the **IceCream** data source.	100
DataSourceInfo(IceCream, DataSourceInfo.MinValue, Quantity)	Returns the minimum numeric value for the **Quantity** column of the **IceCream** data source.	0
DataSourceInfo(IceCream, DataSourceInfo.ReadPermission)	Can the current user read records in the **IceCream** data source?	**true**
DataSourceInfo(IceCream, DataSourceInfo.EditPermission)	Can the current user edit records in the **IceCream** data source?	**true**

Formula	Description	Result
DataSourceInfo(IceCream, DataSourceInfo.CreatePermission)	Can the current user create records in the **IceCream** data source?	**false**
DataSourceInfo(IceCream, DataSourceInfo.DeletePermission)	Can the current user delete records in the **IceCream** data source?	**false**

Date, DateTime, and Time functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Converts date and time components to a date/time value.

Description

The **Date** function converts individual Year, Month, and Day values to a Date/Time value. The time portion is midnight.

- If Year is between 0 and 1899 (inclusive), the function adds that value to 1900 to calculate the year. **70** becomes **1970**.
- If Month is less than 1 or more than 12, the result subtracts or adds that many months from the beginning of the specified year.
- If Day is greater than the number of days in the specified month, the function adds that many days to the first day of the month and returns the corresponding date from a subsequent month. If Day is less than 1, the function subtracts that many days, plus 1, from the first day of the specified month.

The **Time** function converts individual Hour, Minute, Second, and optionally Millisecond values to a Date/Time value. The result has no date associated with it.

The **DateTime** function combines the **Date** and **Time** functions into a single function, taking both date and time arguments and returning a Date/Time value that has both date and time components.

See the **DateValue**, **TimeValue**, and **DateTimeValue** functions for information about how to convert a string to a value.

Also see working with dates and times for more information.

Syntax

Date(*Year*, *Month*, *Day*)

- *Year* - Required. Numbers greater than 1899 are interpreted as absolute (1980 is interpreted as 1980); numbers that range from 0 to 1899 are interpreted as relative to 1900. (For example, 80 is interpreted as 1980.)

- *Month* - Required. A number that ranges from 1 to 12.
- *Day* - Required. A number that ranges from 1 to 31.

Time(*Hour*, *Minute*, *Second* [, *Milliseconds*])

- *Hour* - Required. Number of hours, usually in the range 0 (12:00 AM) to 23 (11:00 PM).
- *Minute* - Required. Number of minutes, usually in the range 0 to 59.
- *Second* - Required. Number of seconds, usually in the range 0 to 59.
- *Milliseconds* - Optional. Number of milliseconds, usually in the range 0 to 999.

DateTime(*Year*, *Month*, *Day*, *Hour*, *Minute*, *Second* [, *Milliseconds*])

- *Year* - Required. Numbers greater than 1899 are interpreted as absolute (1980 is interpreted as 1980); numbers that range from 0 to 1899 are interpreted as relative to 1900. (For example, 80 is interpreted as 1980.)
- *Month* - Required. A number that ranges from 1 to 12.
- *Day* - Required. A number that ranges from 1 to 31.
- *Hour* - Required. Number of hours, usually in the range 0 (12:00 AM) to 23 (11:00 PM)
- *Minute* - Required. Number of minutes, usually in the range 0 to 59.
- *Second* - Required. Number of seconds, usually in the range 0 to 59.
- *Milliseconds* - Optional. Number of milliseconds, usually in the range 0 to 999.

Examples

Date

If a user typed

- **1979** in a text-input control named **HireYear**
- **3** in a text-input control named **HireMonth**
- **17** in a text-input control named **HireDay**

this formula would return `3/17/1979`:

Power Apps

```
Date( Value(HireYear.Text), Value(HireMonth.Text), Value(HireDay.Text) )
```

Time

If a user typed

- **14** in a text-input control named **BirthHour**
- **50** in a text-input control named **BirthMinute**
- **24** in a text-input control named **BirthSecond**

this formula would return `02:50:24 P`:

Power Apps

```
Text(
    Time(Value(BirthHour.Text), Value(BirthMinute.Text),
Value(BirthSecond.Text)),
    "hh:mm:ss A/P"
)
```

DateTime

If a user typed

- **2023** in a text-input control named **EclipseYear**
- **10** in a text-input control named **EclipseMonth**
- **28** in a text-input control named **EclipseDate**
- **13** in a text-input control named **EclipseHour**
- **14** in a text-input control named **EclipseMinute**
- **5** in a text-input control named **EclipseSecond**
- **231** in a text-input control named **EclipseMillisecond**

this formula would return `10/28/23 01:14:05.231 PM`:

Power Apps

```
Text(
    DateTime(Value(EclipseYear.Text), Value(EclipseMonth.Text),
Value(EclipseDate.Text),
        Value(EclipseHour.Text), Value(EclipseMinute.Text),
        Value(EclipseSecond.Text), Value(EclipseSecond.Millisecond)
    ),
    "yy/mm/dd hh:mm:ss.000 AM/PM"
)
```

Feedback

Was this page helpful? Yes No

Provide product feedback

DateAdd, DateDiff, and TimeZoneOffset functions

Article • 04/20/2025

Expand table

Functions	Applies to
DateAdd **DateDiff**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
TimeZoneOffset	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Expand table

Function	Applies to
DateAdd	
DateDiff	
TimeZoneOffset	

Adds to or finds the difference in date/time values and converts between local time and UTC.

Description

The **DateAdd** function adds a number of units to a date/time value. The result is a new date/time value. You can also subtract a number of units from a date/time value by specifying a negative value.

The **DateDiff** function returns the difference between two date/time values. The result is a whole number of units.

For both functions, units can be **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. By default, both functions use **TimeUnit.Days** as units.

The **TimeZoneOffset** function returns the number of minutes between the user's local time and UTC (Coordinated Universal Time).

You can use **DateAdd** with the **TimeZoneOffset** to convert between the user's local time and UTC (Coordinated Universal Time). Adding **TimeZoneOffset** will convert a local time to UTC, and subtracting it (adding the negative) will convert from UTC to local time.

Also see Date, Time, and DateTime data types and working with dates and times for more information.

Syntax

DateAdd(*DateTime*, *Addition* [, *Units*])

- *DateTime* - Required. Date/time value to operate on.
- *Addition* - Required. Number, in *Units*, to add to the *DateTime*.
- *Units* - Optional. The type of *Units* to add: **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. If not specified, **TimeUnit.Days** are used.

DateDiff(*StartDateTime*, *EndDateTime* [, *Units*])

- *StartDateTime* - Required. Starting date/time value.
- *EndDateTime* - Required. Ending date/time value.
- *Units* - Optional. The type of *Units* to subtract: **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. If not specified, **TimeUnit.Days** are used.

TimeZoneOffset([*DateTime*])

- *DateTime* - Optional. Date/time value for which to return the offset. By default, the current date/time is used.

Examples

In all of these examples, assume that the current date and time is **July 15, 2013, 1:02 PM**.

Simple DateAdd

Expand table

Formula	Description	Result
Text(DateAdd(Now(), 3), "dd-mm-yyyy hh:mm")	Adds three days (default units) to the current date and time.	"18-07-2013 13:02"

Formula	Description	Result
Text(DateAdd(Now(), 4, TimeUnit.Hours), "dd-mm-yyyy hh:mm")	Add four hours to the current date and time.	"15-07-2013 17:02"
Text(DateAdd(Today(), 1, TimeUnit.Months), "dd-mm-yyyy hh:mm")	Adds one month to the current date, without time as **Today** doesn't return a time component.	"15-08-2013 00:00"
Text(DateAdd(Now(), -30, TimeUnit.Minutes), "dd-mm-yyyy hh:mm")	Subtracts 30 minutes from the current date and time.	"15-07-2013 12:32"

Simple DateDiff

Expand table

Formula	Description	Result
DateDiff(Now(), DateValue("1/1/2014"))	Returns the difference between the two units in the default units of **TimeUnit.Days**	170
DateDiff(Now(), DateValue("1/1/2014"), TimeUnit.Months)	Returns the difference between the two values in **TimeUnit.Months**	6
DateDiff(Now(), Today(), TimeUnit.Minutes)	Returns the difference between the current date/time and the current date only (no time) in minutes. Since the **Now** is later than **Today** the result will be negative.	-782

Difference of dates with fractional results

The function DateDiff only returns a whole number of the units being subtracted, and the precision is given in the unit specified. To calculate the difference with a higher precision, use a smaller unit, and convert the result appropriately, like in the examples below.

Expand table

Formula	Description	Result
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Hours)	The minutes/seconds are ignored, the difference is based on the time up to the hour.	1
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Minutes	The minutes are used in the difference, and the result is divided by 60 to have the difference in	0.5

Formula	Description	Result
)/60	hours.	
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Seconds)/3600	The minutes and seconds are used in the difference; the result is divided by 3600 to have the difference in hours.	0.51

Converting to UTC

To convert to UTC (Coordinated Universal Time), add the **TimeZoneOffset** for the given time.

For example, imagine the current date and time is **July 15, 2013, 1:02 PM** in Pacific Daylight Time (PDT, UTC-7). To determine the current time in UTC, use:

- **DateAdd(Now(), TimeZoneOffset(), TimeUnit.Minutes)**

TimeZoneOffset defaults to the current time, so you don't need to pass it an argument.

To see the result, use the **Text** function with the format *dd-mm-yyyy hh:mm*, which will return **15-07-2013 20:02**.

Converting from UTC

To convert from UTC, subtract the **TimeZoneOffset** (by adding the negative) for the given time.

For example, imagine the UTC date and time **July 15, 2013, 8:02 PM** is stored in a variable named **StartTime**. To adjust the time for the user's time zone, use:

- **DateAdd(StartTime, -TimeZoneOffset(StartTime), TimeUnit.Minutes)**

Note the negative sign before **TimeZoneOffset** to subtract the offset rather than add it.

To see the result, use the **Text** function with the format *dd-mm-yyyy hh:mm*, which will result in **15-07-2013 13:02** if you're in Pacific Daylight Time.

DateAdd, DateDiff, and TimeZoneOffset functions

Article • 04/20/2025

Expand table

Functions	Applies to
DateAdd **DateDiff**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
TimeZoneOffset	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Expand table

Function	Applies to
DateAdd	
DateDiff	
TimeZoneOffset	

Adds to or finds the difference in date/time values and converts between local time and UTC.

Description

The **DateAdd** function adds a number of units to a date/time value. The result is a new date/time value. You can also subtract a number of units from a date/time value by specifying a negative value.

The **DateDiff** function returns the difference between two date/time values. The result is a whole number of units.

For both functions, units can be **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. By default, both functions use **TimeUnit.Days** as units.

The **TimeZoneOffset** function returns the number of minutes between the user's local time and UTC (Coordinated Universal Time).

You can use **DateAdd** with the **TimeZoneOffset** to convert between the user's local time and UTC (Coordinated Universal Time). Adding **TimeZoneOffset** will convert a local time to UTC, and subtracting it (adding the negative) will convert from UTC to local time.

Also see Date, Time, and DateTime data types and working with dates and times for more information.

Syntax

DateAdd(*DateTime*, *Addition* [, *Units*])

- *DateTime* - Required. Date/time value to operate on.
- *Addition* - Required. Number, in *Units*, to add to the *DateTime*.
- *Units* - Optional. The type of *Units* to add: **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. If not specified, **TimeUnit.Days** are used.

DateDiff(*StartDateTime*, *EndDateTime* [, *Units*])

- *StartDateTime* - Required. Starting date/time value.
- *EndDateTime* - Required. Ending date/time value.
- *Units* - Optional. The type of *Units* to subtract: **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. If not specified, **TimeUnit.Days** are used.

TimeZoneOffset([*DateTime*])

- *DateTime* - Optional. Date/time value for which to return the offset. By default, the current date/time is used.

Examples

In all of these examples, assume that the current date and time is **July 15, 2013, 1:02 PM**.

Simple DateAdd

Expand table

Formula	Description	Result
Text(DateAdd(Now(), 3), "dd-mm-yyyy hh:mm")	Adds three days (default units) to the current date and time.	"18-07-2013 13:02"

Formula	Description	Result
Text(DateAdd(Now(), 4, TimeUnit.Hours), "dd-mm-yyyy hh:mm")	Add four hours to the current date and time.	"15-07-2013 17:02"
Text(DateAdd(Today(), 1, TimeUnit.Months), "dd-mm-yyyy hh:mm")	Adds one month to the current date, without time as **Today** doesn't return a time component.	"15-08-2013 00:00"
Text(DateAdd(Now(), -30, TimeUnit.Minutes), "dd-mm-yyyy hh:mm")	Subtracts 30 minutes from the current date and time.	"15-07-2013 12:32"

Simple DateDiff

Expand table

Formula	Description	Result
DateDiff(Now(), DateValue("1/1/2014"))	Returns the difference between the two units in the default units of **TimeUnit.Days**	170
DateDiff(Now(), DateValue("1/1/2014"), TimeUnit.Months)	Returns the difference between the two values in **TimeUnit.Months**	6
DateDiff(Now(), Today(), TimeUnit.Minutes)	Returns the difference between the current date/time and the current date only (no time) in minutes. Since the **Now** is later than **Today** the result will be negative.	-782

Difference of dates with fractional results

The function DateDiff only returns a whole number of the units being subtracted, and the precision is given in the unit specified. To calculate the difference with a higher precision, use a smaller unit, and convert the result appropriately, like in the examples below.

Expand table

Formula	Description	Result
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Hours)	The minutes/seconds are ignored, the difference is based on the time up to the hour.	1
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Minutes	The minutes are used in the difference, and the result is divided by 60 to have the difference in	0.5

Formula	Description	Result
)/60	hours.	
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Seconds)/3600	The minutes and seconds are used in the difference; the result is divided by 3600 to have the difference in hours.	0.51

Converting to UTC

To convert to UTC (Coordinated Universal Time), add the **TimeZoneOffset** for the given time.

For example, imagine the current date and time is **July 15, 2013, 1:02 PM** in Pacific Daylight Time (PDT, UTC-7). To determine the current time in UTC, use:

- **DateAdd(Now(), TimeZoneOffset(), TimeUnit.Minutes)**

TimeZoneOffset defaults to the current time, so you don't need to pass it an argument.

To see the result, use the **Text** function with the format *dd-mm-yyyy hh:mm*, which will return **15-07-2013 20:02**.

Converting from UTC

To convert from UTC, subtract the **TimeZoneOffset** (by adding the negative) for the given time.

For example, imagine the UTC date and time **July 15, 2013, 8:02 PM** is stored in a variable named **StartTime**. To adjust the time for the user's time zone, use:

- **DateAdd(StartTime, -TimeZoneOffset(StartTime), TimeUnit.Minutes)**

Note the negative sign before **TimeZoneOffset** to subtract the offset rather than add it.

To see the result, use the **Text** function with the format *dd-mm-yyyy hh:mm*, which will result in **15-07-2013 13:02** if you're in Pacific Daylight Time.

DateValue, TimeValue, and DateTimeValue functions

06/19/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts date, time, or both in a *string* to a *date/time* value.

Description

- **DateValue** function converts a *date string* (like "10/01/2014") to a *date/time* value.
- **TimeValue** function converts a *time string* (like "12:15 PM") to a *date/time* value.
- **DateTimeValue** function converts a *date and time string* (like "January 10, 2013 12:13 AM") to a *date/time* value.

DateValue function ignores any time information in the date string, and **TimeValue** function ignores any date information in the time string.

> ⓘ **Note**
>
> The DateValue, TimeValue, and DateTimeValue functions by default use the language from the current user's settings. You can override it to ensure that strings are interpreted properly. For example, "10/1/1920" is interpreted as *October 1st* in "*en*" and as *January 10th* in "*fr*".

Dates must use one of these formats:

- MM/DD/YYYY or MM-DD-YYYY
- DD/MM/YYYY or DD-MM-YYYY
- YYYY/MM/DD or YYYY-MM-DD
- MM/DD/YY or MM-DD-YY
- DD/MM/YY or DD-MM-YY
- DD Mon YYYY
- Month DD, YYYY

To convert from numeric date, month and year components, read Date.
To convert from numeric hour, minute and second components, read Time.

For more information, see:

- Working with date and time
- Date/time and data types

Syntax

DateValue(*String* [, *Language*])
DateTimeValue(*String* [, *Language*])
TimeValue(*String* [, *Language*])

- *String* - Required. A text string that contains a date, time, or combination date and time value.
- *Language* - Optional. A language string, such as would be returned by the first two characters from the Language function. If not provided, the language of the current user's settings is used.

DateValue(*Dynamic*)
DateTimeValue(*Dynamic*)
TimeValue(*Dynamic*)

- *Dynamic* - Required. **Dynamic** value that represents a date or time. Acceptable values are dependent on the untyped provider. For **JSON**, the dynamic value is expected to be a JSON string that contains a date and time in ISO 8601 format. Dates or times in other formats will result in an error. Consider converting such values to **Text** first, then to a date or time. Keep in mind that time zones and locale-related formats are important considerations when communicating with external systems.

Examples

DateValue

If you type **10/11/2014** into a text-input control named **Startdate**, and then set the Text property of a label to these formulas:

- Convert a date from a string in the user's locale and show the result as a long date.

  Power Fx

  ```
  Text( DateValue( Startdate.Text ), DateTimeFormat.LongDate )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014**.

> **Note**
>
> You can use several options with the **DateTimeFormat** enum. To display a list of options, type the parameter followed by a dot or period (.) in the formula bar or check Text function reference.

- Convert date from a string in the French locale and show the result as a long date. In this example, the months and day of the month are interpreted differently from English.

  ```
  Power Fx
  Text( DateValue( Startdate.Text, "fr" ), DateTimeFormat.LongDate )
  ```

  Device set to **en** locale shows the label as **Monday, November 10, 2014**.

If you enter **October 20, 2014** instead:

- Convert a date from a string in the user's locale and calculate the difference between two days, in days

  ```
  Power Fx
  DateDiff( DateValue( Startdate.Text ), Today() )
  ```

  Device set to **en** locale shows the label as **9**, indicating the number of days between October 11 and October 20. The DateDiff function can also show the difference in months, quarters, or years.

DateTimeValue

If you typed **10/11/2014 1:50:24.765 PM** into a text-input control named **Start**, and then set the Text property of a label to the following formula:

- Convert both a date and time string in the current locale.

  ```
  Power Fx
  Text( DateTimeValue( Start.Text ), DateTimeFormat.LongDateTime )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014 1:50:24 PM**.

> **Note**

You can use several options with the **DateTimeFormat** enum. To display a list of options, type the parameter followed by a dot or period (.) in the formula bar or check Text function reference.

- Convert both a date and time string in the French locale. Month and day of the month are interpreted differently.

  Power Fx

  ```
  Text( DateTimeValue( Start.Text, "fr"), DateTimeFormat.LongDateTime )
  ```

  Device set to **en** locale shows the label as **Monday, November 10, 2014 1:50:24 PM**.

- Convert both a date and time string in the user's locale, and display the result with a fractional second.

  Power Fx

  ```
  Text( DateTimeValue( Start.Text ), "dddd, mmmm dd, yyyy hh:mm:ss.fff AM/PM" )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014 01:50:24.765 PM**.

  As an alternative, you can specify **hh:mm:ss.f** or **hh:mm:ss.ff** to round the time to the nearest 10th or 100th of a second.

TimeValue

Name a text-input control **FinishedAt**, and set the Text property of a label to this formula:

Power Fx

```
If( TimeValue( FinishedAt.Text ) < TimeValue( "5:00:00.000 PM" ),
    "You made it!",
    "Too late!"
)
```

- If you type **4:59:59.999 PM** in the **FinishedAt** control, the label shows "*You made it!*"
- If you type **5:00:00.000 PM** in the **FinishedAt** control, the label shows "*Too late!*"

DateValue, TimeValue, and DateTimeValue functions

06/19/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts date, time, or both in a *string* to a *date/time* value.

Description

- **DateValue** function converts a *date string* (like "10/01/2014") to a *date/time* value.
- **TimeValue** function converts a *time string* (like "12:15 PM") to a *date/time* value.
- **DateTimeValue** function converts a *date and time string* (like "January 10, 2013 12:13 AM") to a *date/time* value.

DateValue function ignores any time information in the date string, and **TimeValue** function ignores any date information in the time string.

> ⓘ **Note**
>
> The DateValue, TimeValue, and DateTimeValue functions by default use the language from the current user's settings. You can override it to ensure that strings are interpreted properly. For example, "10/1/1920" is interpreted as *October 1st* in "*en*" and as *January 10th* in "*fr*".

Dates must use one of these formats:

- MM/DD/YYYY or MM-DD-YYYY
- DD/MM/YYYY or DD-MM-YYYY
- YYYY/MM/DD or YYYY-MM-DD
- MM/DD/YY or MM-DD-YY
- DD/MM/YY or DD-MM-YY
- DD Mon YYYY
- Month DD, YYYY

To convert from numeric date, month and year components, read Date.
To convert from numeric hour, minute and second components, read Time.

For more information, see:

- Working with date and time
- Date/time and data types

Syntax

DateValue(*String* [, *Language*])
DateTimeValue(*String* [, *Language*])
TimeValue(*String* [, *Language*])

- *String* - Required. A text string that contains a date, time, or combination date and time value.
- *Language* - Optional. A language string, such as would be returned by the first two characters from the Language function. If not provided, the language of the current user's settings is used.

DateValue(*Dynamic*)
DateTimeValue(*Dynamic*)
TimeValue(*Dynamic*)

- *Dynamic* - Required. **Dynamic** value that represents a date or time. Acceptable values are dependent on the untyped provider. For **JSON**, the dynamic value is expected to be a JSON string that contains a date and time in ISO 8601 format. Dates or times in other formats will result in an error. Consider converting such values to **Text** first, then to a date or time. Keep in mind that time zones and locale-related formats are important considerations when communicating with external systems.

Examples

DateValue

If you type **10/11/2014** into a text-input control named **Startdate**, and then set the Text property of a label to these formulas:

- Convert a date from a string in the user's locale and show the result as a long date.

  Power Fx

  ```
  Text( DateValue( Startdate.Text ), DateTimeFormat.LongDate )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014**.

> **Note**
>
> You can use several options with the **DateTimeFormat** enum. To display a list of options, type the parameter followed by a dot or period (.) in the formula bar or check Text function reference.

- Convert date from a string in the French locale and show the result as a long date. In this example, the months and day of the month are interpreted differently from English.

  ```
  Power Fx
  Text( DateValue( Startdate.Text, "fr" ), DateTimeFormat.LongDate )
  ```

  Device set to **en** locale shows the label as **Monday, November 10, 2014**.

If you enter **October 20, 2014** instead:

- Convert a date from a string in the user's locale and calculate the difference between two days, in days

  ```
  Power Fx
  DateDiff( DateValue( Startdate.Text ), Today() )
  ```

  Device set to **en** locale shows the label as **9**, indicating the number of days between October 11 and October 20. The DateDiff function can also show the difference in months, quarters, or years.

DateTimeValue

If you typed **10/11/2014 1:50:24.765 PM** into a text-input control named **Start**, and then set the Text property of a label to the following formula:

- Convert both a date and time string in the current locale.

  ```
  Power Fx
  Text( DateTimeValue( Start.Text ), DateTimeFormat.LongDateTime )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014 1:50:24 PM**.

> **Note**

You can use several options with the **DateTimeFormat** enum. To display a list of options, type the parameter followed by a dot or period (.) in the formula bar or check Text function reference.

- Convert both a date and time string in the French locale. Month and day of the month are interpreted differently.

  Power Fx

  ```
  Text( DateTimeValue( Start.Text, "fr"), DateTimeFormat.LongDateTime )
  ```

  Device set to **en** locale shows the label as **Monday, November 10, 2014 1:50:24 PM**.

- Convert both a date and time string in the user's locale, and display the result with a fractional second.

  Power Fx

  ```
  Text( DateTimeValue( Start.Text ), "dddd, mmmm dd, yyyy hh:mm:ss.fff AM/PM" )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014 01:50:24.765 PM**.

  As an alternative, you can specify **hh:mm:ss.f** or **hh:mm:ss.ff** to round the time to the nearest 10th or 100th of a second.

TimeValue

Name a text-input control **FinishedAt**, and set the Text property of a label to this formula:

Power Fx

```
If( TimeValue( FinishedAt.Text ) < TimeValue( "5:00:00.000 PM" ),
    "You made it!",
    "Too late!"
)
```

- If you type **4:59:59.999 PM** in the **FinishedAt** control, the label shows "*You made it!*"
- If you type **5:00:00.000 PM** in the **FinishedAt** control, the label shows "*Too late!*"

Day, Month, Year, Hour, Minute, Second, and Weekday functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns individual components of a Date/Time value.

Description

The **Day** function returns the day component of a Date/Time value, ranging from 1 to 31.

The **Month** function returns the month component of a Date/Time value, ranging from 1 to 12.

The **Year** function returns the year component of a Date/Time value, starting with 1900.

The **Hour** function returns the hour component of a Date/Time value, ranging from 0 (12:00 AM) to 23 (11:00 PM).

The **Minute** function returns the minute component of a Date/Time value, ranging from 0 to 59.

The **Second** function returns the second component of a Date/Time value, ranging from 0 to 59.

The **Weekday** function returns the weekday of a Date/Time value. By default, the result ranges from 1 (Sunday) to 7 (Saturday). You can specify a different range with a Microsoft Excel Weekday function code or a StartOfWeek enumeration value:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	StartOfWeek.Sunday	Numbers 1 (Sunday) through 7 (Saturday). Default.
2, 11	StartOfWeek.Monday	Numbers 1 (Monday) through 7 (Sunday).
3	StartOfWeek.MondayZero	Numbers 0 (Monday) through 6 (Sunday).
12	StartOfWeek.Tuesday	Numbers 1 (Tuesday) through 7 (Monday).

Excel code	StartOfWeek enumeration	Description
13	**StartOfWeek.Wednesday**	Numbers 1 (Wednesday) through 7 (Tuesday).
14	**StartOfWeek.Thursday**	Numbers 1 (Thursday) through 7 (Wednesday).
15	**StartOfWeek.Friday**	Numbers 1 (Friday) through 7 (Thursday).
16	**StartOfWeek.Saturday**	Numbers 1 (Saturday) through 7 (Friday).

All functions return a number.

See working with dates and times for more information.

Syntax

Day(*DateTime*)
Month(*DateTime*)
Year(*DateTime*)
Hour(*DateTime*)
Minute(*DateTime*)
Second(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on.

Weekday(*DateTime* [, *WeekdayFirst*])

- *DateTime* - Required. Date/Time value to operate on.
- *WeekdayFirst* - Optional. Excel code specifying which day starts the week. If not supplied, 1 (Sunday first) is used.

Examples

For the following example, the current time is **3:59:37 PM** on **Thursday, April 9, 2015**.

Expand table

Formula	Description	Result
Year(Now())	Returns the year component of the current time and date.	2015
Month(Now())	Returns the month component of the current time and date.	4

Formula	Description	Result
Day(Now())	Returns the day component of the current time and date.	9
Hour(Now())	Returns the hour component of the current time and date.	15
Minute(Now())	Returns the minute component of the current time and date.	59
Second(Now())	Returns the second component of the current time and date.	37
Weekday(Now())	Returns the weekday component of the current time and date, using the default start of the week as Sunday.	5
Weekday(Now(), 14)	Returns the weekday component of the current time and date, using an Excel code to specify the start of the week as Thursday.	1
Weekday(Now(), StartOfWeek.Wednesday)	Returns the weekday component of the current time and date, using a **StartOfWeek** enumeration to specify the start of the week as Wednesday.	2

Feedback

Was this page helpful? Yes No

Provide product feedback

Defaults function

10/10/2025

Applies to: ✅ Canvas apps ✅ Model-driven apps

Returns the default values for a data source.

Description

Use the **Defaults** function to prepopulate a data entry form, making it easier to fill.

This function returns a record that contains the default values for the data source. If a column within the data source doesn't have a default value, that property isn't present.

Data sources vary in how much default information they provide, including not providing any at all. When you work with a collection or another data source that doesn't support default values, the **Defaults** function returns an empty record.

You can combine the **Defaults** function with the **Patch** function to create a record.

Syntax

Defaults(*DataSource*)

- *DataSource* – Required. The data source for which you want default values.

Examples

Expand table

Formula	Description	Result
Defaults(Scores)	Returns the default values for the **Scores** data source.	**{ Score: 0 }**

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Enable and Disable functions

10/10/2025

Applies to: Canvas apps

Turns a signal on or off.

Overview

Some signals can change often, requiring the app to recalculate as they do. Rapid changes over a long period of time can drain a device's battery. You can use these functions to manually turn a signal on or off.

When a signal isn't in use, it's automatically turned off.

Description

The **Enable** and **Disable** functions turn a signal on and off, respectively.

These functions currently only work for the **Location** signal.

These functions have no return value. You can use them only in behavior formulas.

Syntax

Enable(*Signal*)
Disable(*Signal*)

- *Signal* - Required. The signal to turn on or off.

Distinct function

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Summarizes records of a table, removing duplicates.

Description

The **Distinct** function evaluates a formula across each record of a table and returns a one-column table of the results with duplicate values removed. The name of the column is **Value**.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

When used with a data source, this function can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Distinct(*Table*, *Formula*)

- *Table* - Required. Table to evaluate across.
- *Formula* - Required. Formula to evaluate for each record.

Example

1. Insert a **Button** control, and set its **OnSelect** property to this formula.

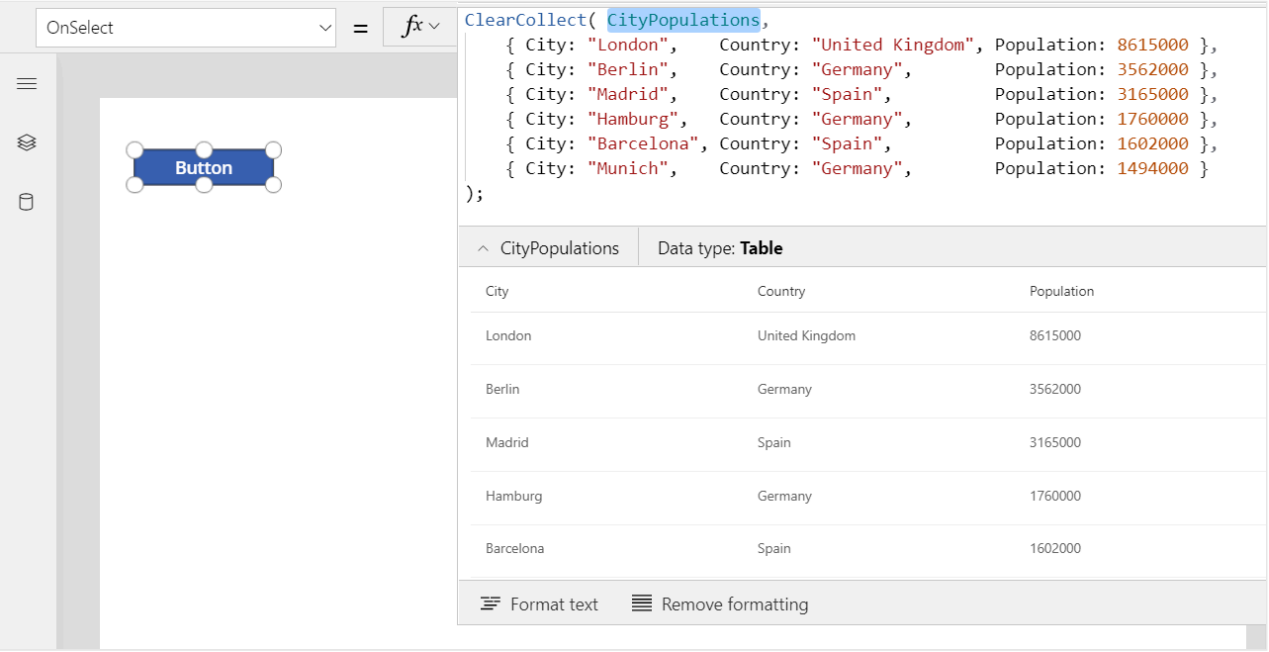
```
Power Fx
```

```
ClearCollect( CityPopulations,
    { City: "London",    Country: "United Kingdom", Population: 8615000 },
    { City: "Berlin",    Country: "Germany",        Population: 3562000 },
    { City: "Madrid",    Country: "Spain",          Population: 3165000 },
    { City: "Hamburg",   Country: "Germany",        Population: 1760000 },
```

```
    { City: "Barcelona", Country: "Spain",          Population: 1602000 },
    { City: "Munich",    Country: "Germany",        Population: 1494000 }
);
```

2. Select the button while holding down the Alt key.

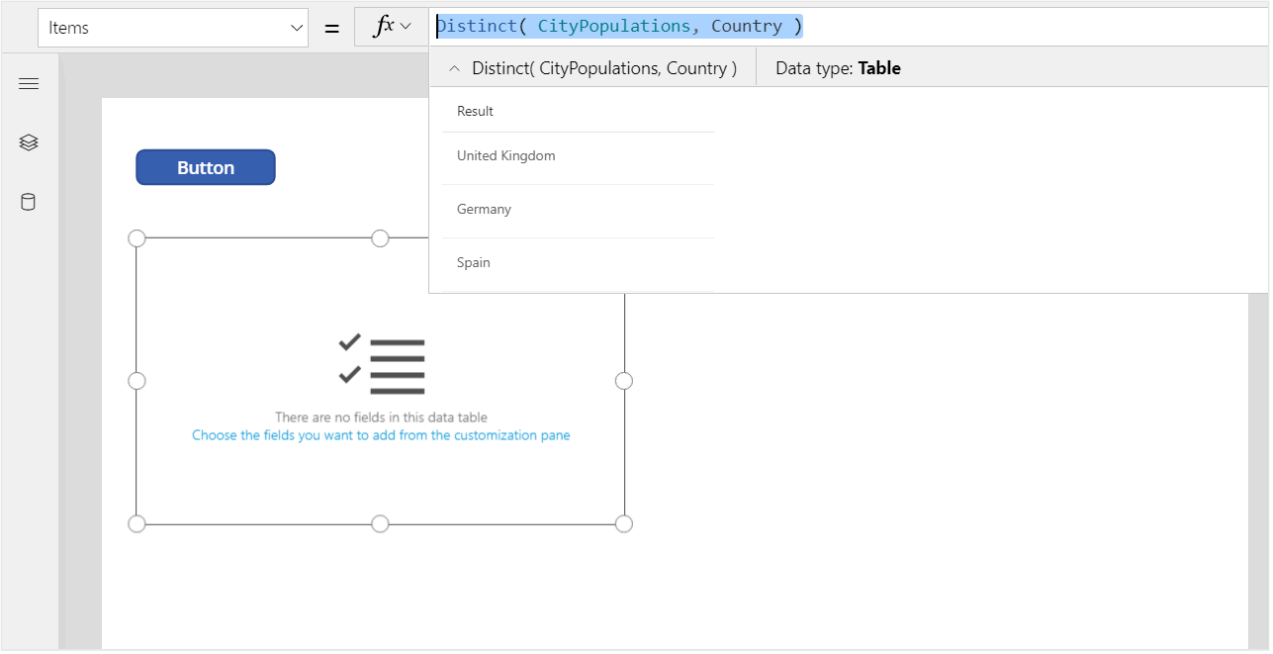
   The formula is evaluated and the **CityPopulations** collection is created which you can show by selecting **CityPopulations** in the formula bar:



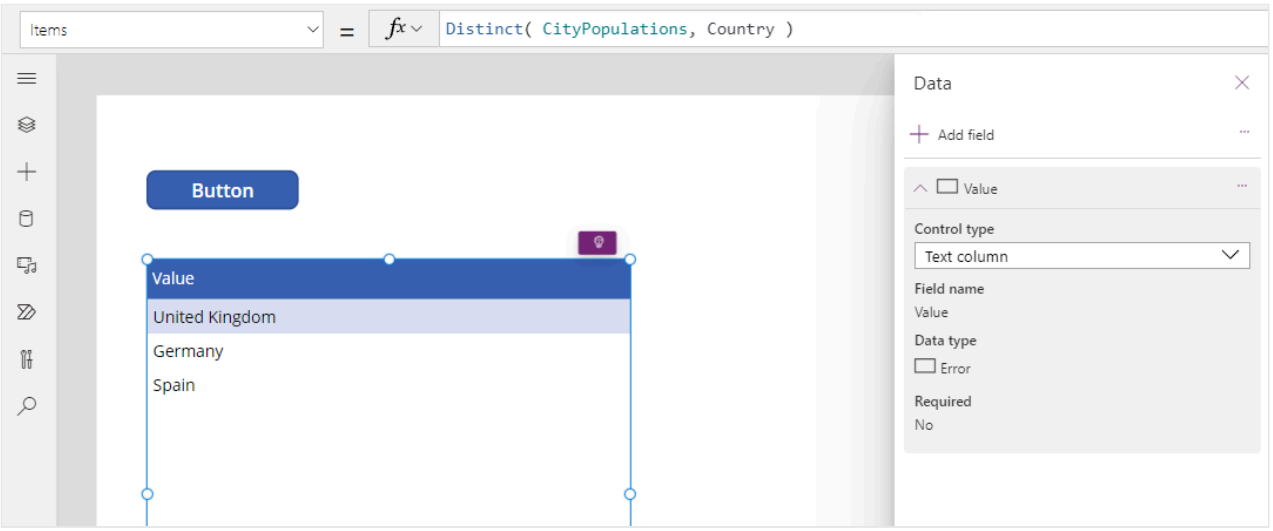
3. Insert a Data table control, and set its **Items** property to this formula:

```
Power Fx

Distinct( CityPopulations, Country )
```

   You can view the result of this formula in the formula bar by selecting the entire formula:



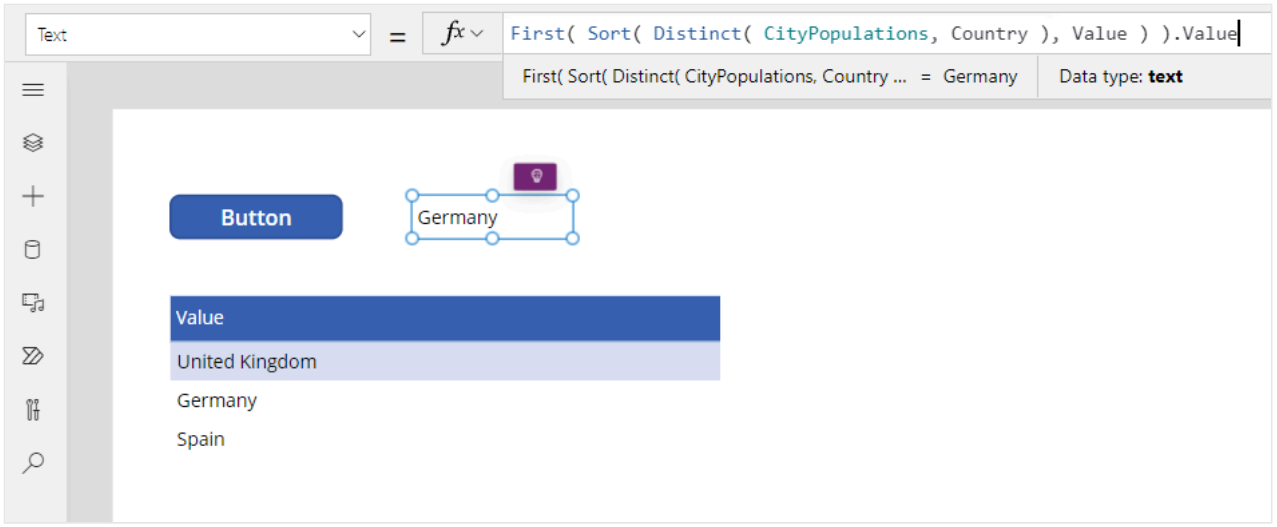
4. Use the **Edit fields** link in the data table's properties pane to add the **Value** column:



5. Insert a Label control, and set its **Text** property to the formula:

```
Power Fx

First( Sort( Distinct( CityPopulations, Country ), Value ) ).Value
```

This formula sorts the results from **Distinct** with the Sort function, takes the first record from the resulting table with the First function, and extracts the **Result** field to obtain just the country/region name.


Text
=
First(Sort(Distinct(CityPopulations, Country), Value)).Value
First(Sort(Distinct(CityPopulations, Country ... = Germany
Data type: text
Button
Germany
Value
United Kingdom
Germany
Spain

Download function

Applies to: ✅ Canvas apps ✅ Model-driven apps

Downloads a file from the web to the local device.

Description

The **Download** function downloads a file from the web to the local device.

In native players (Windows, Android, and iOS), the user is prompted for a location to save the file.

When used on the web, **Download** is dependent on the browser's settings and pop-up blocker configuration to determine what happens with the file. For images, videos, PDFs, and other file types that the browser natively supports, a new browser tab is opened to display the file. Many browsers support saving the contents to the local file system.

Only on Windows, **Download** returns the location where the file was stored locally as a text string.

Download can only be used in behavior formulas.

Syntax

Download(*Address*)

- *Address* – Required. The URL address of a web resource to download.

> ⓘ **Note**
>
> Power Apps cannot authenticate download requests to the address provided in the Download() function.
>
> For example, when using this function to download a file stored on a SharePoint site that requires authentication, the request might work when using a web browser since the browser session might authenticate against the SharePoint site using cached credentials. However, in Power Apps mobile app, the request won't work since authenticating the download request is not handled by the mobile device.

Examples

Simple Download

The following formula downloads the user's guide for the Surface Book, a PDF file:

Power Fx

```
Download( "https://go.microsoft.com/fwlink/?linkid=827480" )
```

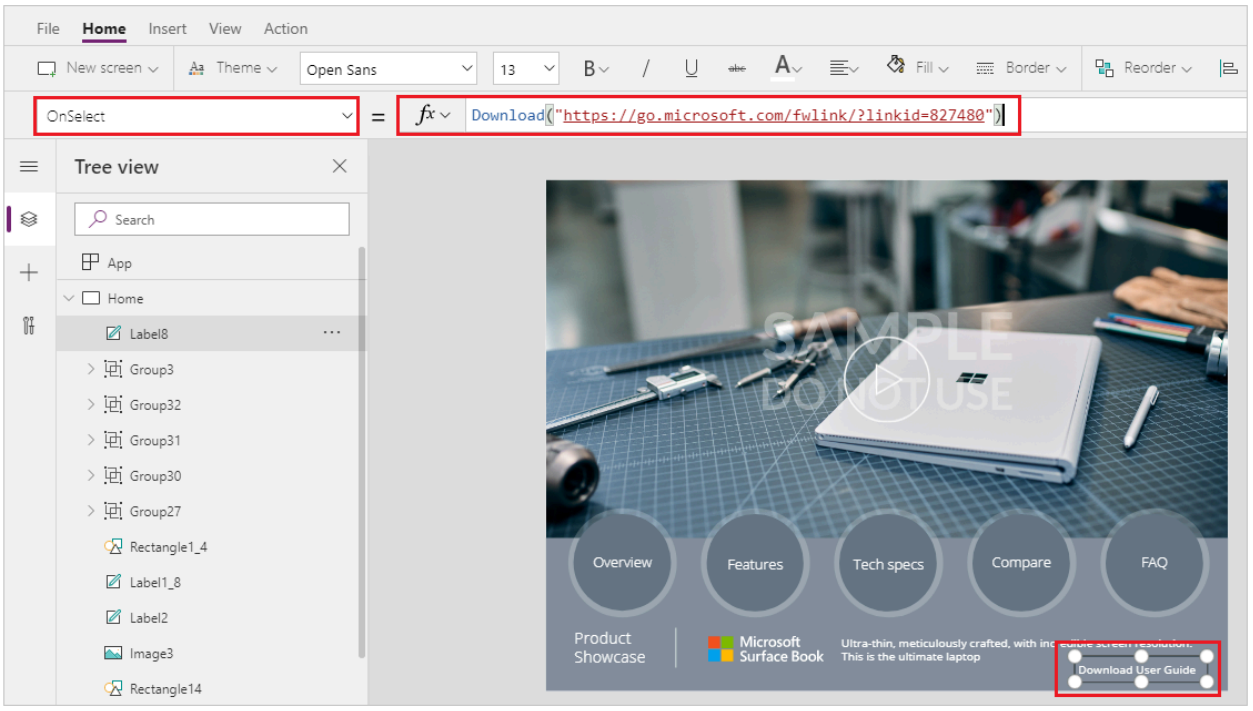
When run in a mobile device, the user is prompted for a location to save the file.

When run in most web browsers, a new tab is opened to display the PDF file as most browsers natively support this file type.

Step by Step

The **Product Showcase** tablet layout template was used for the following example. To create an app with this template, follow the steps from create an app article and select the **Product Showcase** template. You can also use your own app.

1. Go to Power Apps.
2. Select **Apps** from left navigation pane.
3. Select your app and then select **Edit**.
4. Select **Insert** from the menu and then select **Label**.
5. Move the label to the bottom right of the screen.
6. From the properties pane on the right-side, select **Color** as *white* and set **Border thickness** at *1*.
7. Select the **Text** property from right-side and enter text as *Download User Guide*.
8. From property list on top left, select **OnSelect**.
9. Enter formula as `Download("https://go.microsoft.com/fwlink/?linkid=827480")`. You can also use any other URL of your choice.



10. Save and publish the app.
11. Play the app.
12. Select the **Download User Guide** button to download the guide.

> **Note**
>
> - Your browser settings determine whether to download the file or open the file directly in a new tab. For more details, go to Download function description.
> - If the Download formula is used to retrieve a file and the file isn't downloaded successfully, it's recommended to check whether the download is blocked by a browser pop-up blocker or if the browser is waiting for the user's response to a dialog box asking for permission to download the file.

See also

Canvas app formula reference

Last updated on 11/19/2025

AddColumns, DropColumns, RenameColumns, and ShowColumns functions

06/21/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Shapes a table or record by adding, dropping, renaming, and selecting its columns.

The ForAll function can also be used to shape a table, by returning a table of new records created from existing columns.

Overview

These functions shape a table or record by adjusting its columns:

- Reduce a table or record that contains multiple columns down to a single column for use with single-column functions, such as **Lower** or **Sqrt**.
- Add a calculated column to a table or record (for example, a **Total Price** column that shows the results of multiplying **Quantity** by **Unit Price**).
- Rename a column to something more meaningful, for display to users or for use in formulas.

A table is a value in Power Apps, just like a string or a number. You can specify a table as an argument in a formula, and functions can return a table as a result.

> **Note**
>
> The functions that this topic describes don't modify the original table. Instead, they take that table as an argument and return a new table with a transform applied. For more information, see working with tables.

You can't modify the columns of a data source by using these functions. You must modify the data at its source. You can add columns to a collection with the **Collect** function. For more information, see working with data sources.

Description

The **AddColumns** function adds a column to a table or record, and a formula defines the values in that column. Existing columns remain unmodified.

The formula is evaluated for the provided record, or for each record of the provided table.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

The **DropColumns** function excludes columns from a table or record. All other columns remain unmodified. **DropColumns** excludes columns, and **ShowColumns** includes columns.

Use the **RenameColumns** function to rename one or more columns of a table or record by providing at least one argument pair that specifies the name of a column that the table or record contains (the old name, which you want to replace) and the name of a column that the table or record doesn't contain (the new name, which you want to use). The old name must already exist in the table or record, and the new name must not exist. Each column name may appear only once in the argument list as either an old column name or a new column name. To rename a column to an existing column name, first drop the existing column with **DropColumns**, or rename the existing column out of the way by nesting one **RenameColumns** function within another.

The **ShowColumns** function includes columns of a table or record and drops all other columns. You can use **ShowColumns** to create a single-column table or record from a multi-column table or record. **ShowColumns** includes columns, and **DropColumns** excludes columns.

For all these functions, the result is a new table or record with the transform applied. The original table or record isn't modified. You can't modify an existing table or record with a formula. SharePoint, Microsoft Dataverse, SQL Server, and other data sources provide tools for modifying the columns of lists and tables, which are often referred to as the schema. The functions in this topic only transform an input table or record, without modifying the original, into an output table or record for further use.

The arguments to these functions support delegation. For example, a **Filter** function used as an argument to pull in related records searches through all listings, even if the '**[dbo].[AllListings]**' data source contains a million rows:

Power Fx

```
AddColumns( RealEstateAgents,
   Listings,
```

```
    Filter(  '[dbo].[AllListings]', ListingAgentName = AgentName )
)
```

However, the output of these functions is subject to the non-delegation record limit. In this example, only 500 records are returned even if the **RealEstateAgents** data source has 501 or more records.

If you use **AddColumns** in this manner, **Filter** must make separate calls to the data source for each of those first records in **RealEstateAgents**, which causes lots of network chatter. If **[dbo](.[AllListings]** is small enough and doesn't change often. You can call the **Collect** function in OnStart to cache the data source in your app when it starts. As an alternative, you could restructure your app so that you pull in the related records only when the user asks for them.

> **Note**
>
> In Power Apps prior to version 3.24042, column names were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

Syntax

AddColumns(*TableOrRecord*, *ColumnName1*, *Formula1* [, *ColumnName2*, *Formula2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to add.
- *Formula(s)* - Required. Formulas to evaluate for each record. The result is added as the value of the corresponding new column. You can reference other columns of the table or record in this formula.

DropColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to drop.

RenameColumns(*TableOrRecord*, *OldColumnName1*, *NewColumnName1* [, *OldColumnName2*, *NewColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.

- *OldColumnName(s)* - Required. Names of the columns to rename from the original table or record. This element appears first in the argument pair (or first in each argument pair if the formula includes more than one pair).
- *NewColumnName(s)* - Required. Replacement names. This element appears last in the argument pair (or last in each argument pair if the formula includes more than one pair).

ShowColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to include.

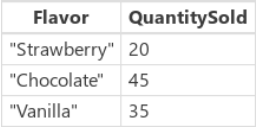
Examples

The examples in this section use the **IceCreamSales** data source, which contains the data in this table:

Flavor	UnitPrice	QuantitySold
"Strawberry"	1.99	20
"Chocolate"	2.99	45
"Vanilla"	1.50	35

None of these examples modify the **IceCreamSales** data source. Each function transforms the value of the data source as a table and returns that value as the result.

Expand table

Formula	Description	Result
AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold)	Adds a **Revenue** column to the result. For each record, **UnitPrice * QuantitySold** is evaluated, and the result is placed in the new column.	
DropColumns(IceCreamSales, UnitPrice)	Excludes the **UnitPrice** column from the result. Use this function to exclude columns, and use **ShowColumns** to include them.	
ShowColumns(IceCreamSales, Flavor)	Includes only the **Flavor** column in the result. Use this function include columns, and use **DropColumns** to exclude them.	
RenameColumns(IceCreamSales, UnitPrice, Price)	Renames the **UnitPrice** column in the result.	

Formula	Description	Result
RenameColumns(IceCreamSales, UnitPrice, Price, QuantitySold, Number)	Renames the **UnitPrice** and **QuantitySold** columns in the result.	<table><tr><th>Flavor</th><th>Price</th><th>Number</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>20</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>45</td></tr><tr><td>"Vanilla"</td><td>1.50</td><td>35</td></tr></table>
DropColumns(RenameColumns(AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold), UnitPrice, Price), Quantity)	Performs the following table transforms in order, starting from the inside of the formula: 1. Adds a **Revenue** column based on the per record calculation of **UnitPrice * Quantity**. 2. Renames **UnitPrice** to **Price**. 3. Excludes the **Quantity** column. Order is important, for example we can't calculate with **UnitPrice** after it's renamed.	<table><tr><th>Flavor</th><th>Price</th><th>Revenue</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>39.80</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>134.55</td></tr><tr><td>"Vanilla"</td><td>1.49</td><td>52.15</td></tr></table>

Step by step

Let's try some of the examples from earlier in this article.

1. Create a collection by adding a **Button** control and setting its **OnSelect** property to this formula:

   Power Fx

   ```
   ClearCollect( IceCreamSales,
      Table(
          { Flavor: "Strawberry", UnitPrice: 1.99, QuantitySold: 20 },
          { Flavor: "Chocolate", UnitPrice: 2.99, QuantitySold: 45 },
          { Flavor: "Vanilla", UnitPrice: 1.50, QuantitySold: 35 }
      )
   )
   ```

2. Run the formula by selecting the button while holding down the Alt key.

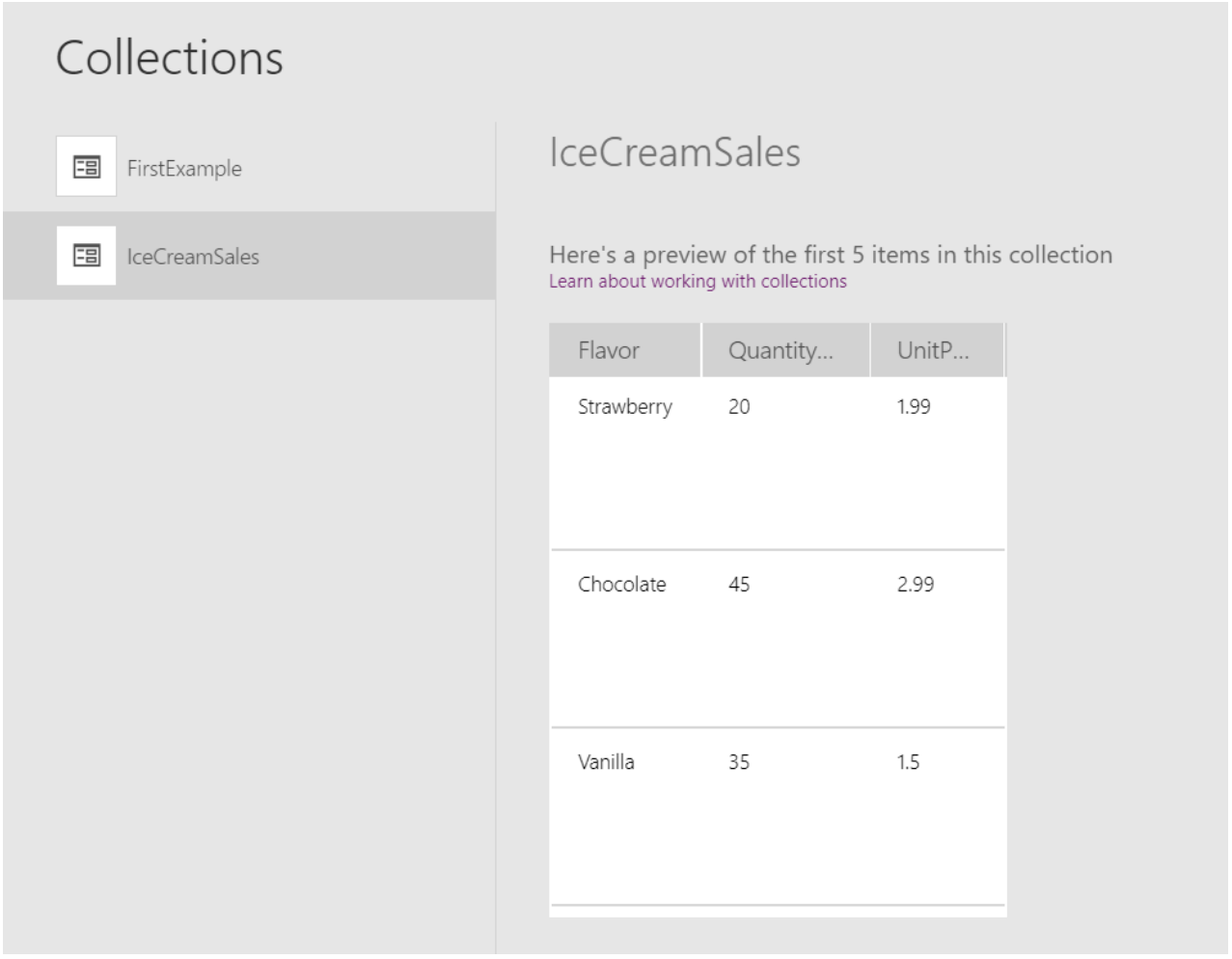
3. Add a second **Button** control, set its **OnSelect** property to this formula, and then run it:

   Power Fx

   ```
   ClearCollect( FirstExample,
      AddColumns( IceCreamSales, Revenue, UnitPrice * QuantitySold )
   )
   ```

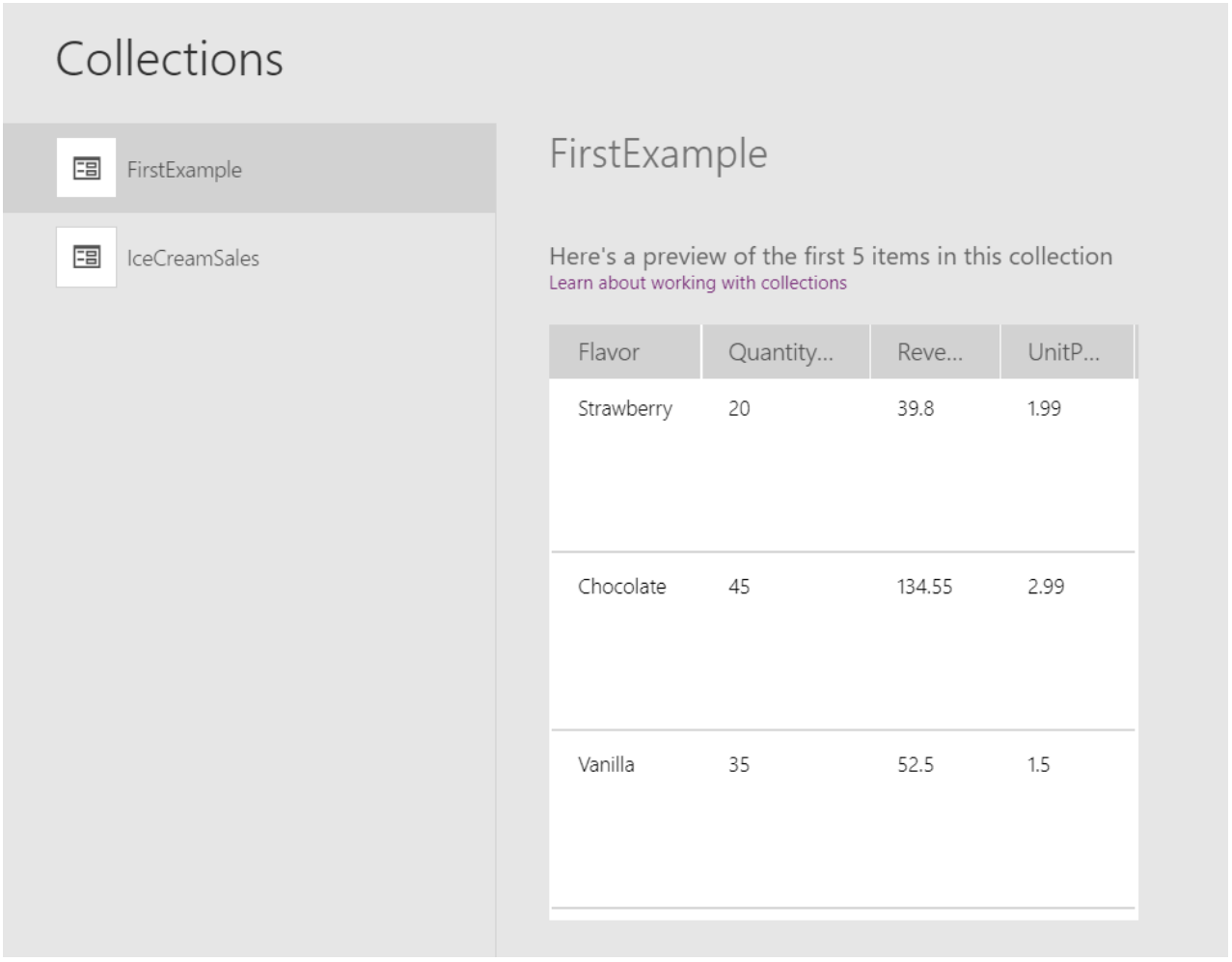
4. On the **File** menu, select **Collections**, and then select **IceCreamSales** to show that collection.

As this graphic shows, the second formula didn't modify this collection. The **AddColumns** function used **IceCreamSales** as a read-only argument; the function didn't modify the table to which that argument refers.



5. Select **FirstExample**.

   As this graphic shows, the second formula returned a new table with the added column. The **ClearCollect** function captured the new table in the **FirstExample** collection, adding something to the original table as it flowed through the function without modifying the source:



Map columns in a component

See Map columns.

EditForm, NewForm, SubmitForm, ResetForm, and ViewForm functions

Article • 03/22/2024

Applies to: Canvas apps

View, edit, or create an item, save the contents, and reset the controls in an **Edit form** control.

Overview

These functions change the state of the **Edit form** control. The form control can be in one of these modes:

Expand table

Mode	Description
FormMode.Edit	The form is populated with an existing record and the user can modify the values of the fields. Once complete, the user can save the changes to the record.
FormMode.New	The form is populated with default values and the user can modify the values of the fields. Once complete, the user can add the record to the data source.
FormMode.View	The form is populated with an existing record but the user cannot modify the values of the fields.

Description

These functions are often invoked from the **OnSelect** formula of a **Button** or **Image** control so that the user can save edits, abandon edits, or create a record. You can use controls and these functions together to create a complete solution.

These functions return no values.

You can use these functions only in behavior formulas.

SubmitForm

Use the **SubmitForm** function in the **OnSelect** property of a Button control to save any changes in a Form control to the data source.

Before submitting any changes, this function checks for validation issues with any field that's marked as required or that has one or more constraints on its value. This behavior matches that of the **Validate** function.

SubmitForm also checks the **Valid** property of the Form, which is an aggregation of all the **Valid** properties of the **Card** controls that the Form control contains. If a problem occurs, the data isn't submitted, and the **Error** and **ErrorKind** properties of the Form control are set accordingly.

If validation passes, **SubmitForm** submits the change to the data source.

- If successful, the Form's **OnSuccess** behavior runs, and the **Error** and **ErrorKind** properties are cleared. If the form was in **FormMode.New** mode, it is returned to **FormMode.Edit** mode.
- If unsuccessful, the Form's **OnFailure** behavior runs, and the **Error** and **ErrorKind** properties are set accordingly. The mode of the form is unchanged.

EditForm

The **EditForm** function changes the Form control's mode to **FormMode.Edit**. In this mode, the contents of the Form control's **Item** property are used to populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is changed, not created. **FormMode.Edit** is the default for the Form control.

> ⓘ **Note**
>
> When the form is in edit mode and Item is null, data card properties are not evaluated and will return default values.

NewForm

The **NewForm** function changes the Form control's mode to **FormMode.New**. In this mode, the contents of the Form control's **Item** property are ignored, and the default values of the Form's **DataSource** property populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is created, not changed.

ResetForm

The **ResetForm** function resets the contents of a form to their initial values, before the user made any changes. If the form is in **FormMode.New** mode, the form is reset to **FormMode.Edit** mode. The **OnReset** behavior of the form control also runs. You can also reset individual controls with the **Reset** function but only from within the form.

ViewForm

The **ViewForm** function changes the Form control's mode to **FormMode.View**. In this mode, the contents of the Form control's **Item** property are used to populate the form. The **SubmitForm** and **ResetForm** functions have no effect when in this mode.

DisplayMode Property

The current mode can be read through the **Mode** property. The mode also determines the value of the **DisplayMode** property, which can be used by data cards and controls within the form control. Often, the data card's **DisplayMode** property will be set to **Parent.DisplayMode** (referencing the form) as will the control's **DisplayMode** property (referencing the data card):

Expand table

Mode	DisplayMode	Description
FormMode.Edit	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept changes to a record.
FormMode.New	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept a new record.
FormMode.View	**DisplayMode.View**	Data cards and controls are not editable and optimized for viewing.

Syntax

SubmitForm(*FormName*)

- *FormName* - Required. Form control to submit to the data source.

EditForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.Edit** mode.

NewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.New** mode.

ResetForm(*FormName*)

- *FormName* - Required. Form control to reset to initial values. Also switches the form from **FormMode.New** mode to **FormMode.Edit** mode.

ViewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.View** mode.

Examples

See Understand data forms for complete examples.

1. Add a Button control, set its **Text** property to show **Save**, and set its **OnSelect** property to this formula:

   SubmitForm(EditForm)

2. Set the **OnFailure** property of a Form control to blank and its **OnSuccess** property to this formula:

   Back()

3. Name a **Label** control **ErrorText**, and set its **Text** property to this formula:

   EditForm.Error

   When the user selects the **Save** button, any changes in the Form control are submitted to the underlying data source.

   - If the submission succeeds, any changes are saved or, if the Form control is in **New** mode, a record is created. **ErrorText** is *blank* and the previous screen reappears.
   - If the submission fails, **ErrorText** shows a user-friendly error message, and the current screen remains visible so that the user can correct the problem and try again.

4. Add a Button control, set its **Text** property to show **Cancel**, and set its **OnSelect** property to this formula:

   ResetForm(EditForm); Back()

   When the user selects the **Cancel** button, the values in the Form control are reset to what they were before the user started to edit it, the previous screen reappears,

and the Form control is returned to **Edit** mode if it was in **New** mode.

5. Add a Button control, set its Text property to show **New**, and set its OnSelect property to this formula:

   NewForm(EditForm); Navigate(EditScreen, None)

   When the user selects the **New** button, the Form control switches to **New** mode, the default values for the Form control's data source populate that control, and the screen that contains the Form control appears. When the **SubmitForm** function runs, a record is created instead of updated.

Enable and Disable functions

10/10/2025

Applies to: Canvas apps

Turns a signal on or off.

Overview

Some signals can change often, requiring the app to recalculate as they do. Rapid changes over a long period of time can drain a device's battery. You can use these functions to manually turn a signal on or off.

When a signal isn't in use, it's automatically turned off.

Description

The **Enable** and **Disable** functions turn a signal on and off, respectively.

These functions currently only work for the **Location** signal.

These functions have no return value. You can use them only in behavior formulas.

Syntax

Enable(*Signal*)
Disable(*Signal*)

- *Signal* - Required. The signal to turn on or off.

EncodeHTML, EncodeUrl, and PlainText functions

Article • 04/20/2025

Expand table

Functions	Applies to
EncodeHTML	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
EncodeUrl **PlainText**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Encodes and decodes strings.

Description

The **EncodeUrl** function encodes a URL string, replacing certain non-alphanumeric characters with % and a hexadecimal number.

The **EncodeHTML** function encodes a string, replacing certain characters that need to be escaped to appear properly in an HTML context. For example, the characters <, >, and & are replaced by <, >, and &, respectively.

The **PlainText** function removes HTML and XML tags, converting certain tags such as these to an appropriate symbol:

- ** **
- **"**

The return value from these functions is the encoded or decoded string. This function doesn't remove all HTML and XML tags.

Syntax

EncodeUrl(*String*)

- *String* - Required. URL to be encoded.

EncodeHTML(*String*)

- *String* - Required. HTML to be encoded.

PlainText(*String*)

- *String* - Required. String from which HTML and XML tags are stripped.

Examples

If you show an RSS feed in a text gallery and then set the **Text** property of a label in that gallery to **ThisItem.description**, the label might show raw HTML or XML code as in this example:

HTML

```
<p>
  We have done an unusually &quot;deep&quot; globalization and
  localization.
</p>
```

If you set the **Text** property of the label to **PlainText(ThisItem.description)**, the text appears as in this example:

```
    We have done an unusually "deep" globalization and localization.
```

With the formula **EncodeHTML(ThisItem.description)**, the HTML tags are encoded so that they show up as text when interpreted as HTML, which can be useful to see the HTML structure:

```
&lt;p&gt;
  We have done an unusually&amp;nbsp;&amp;quot;deep&amp;quot; globalization and
  localization.
&lt;/p&gt;
```

If instead of a label you have an HTML text control in the gallery, and you set the **HtmlText** property of that control to **ThisItem.description** you also see the decoded text, because the HTML text control is interpreting the tags as HTML.

EndsWith and StartsWith functions

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests whether a text string begins or ends another text string.

Description

The **EndsWith** function tests whether one text string ends with another.

The **StartsWith** function tests whether one text string begins with another.

For both functions, the tests are case insensitive if the back-end datasource supports it by default. For example, collections, Dataverse, SharePoint, and SQL Server support case insensitivity by default. Oracle does not. The return value of both is a Boolean **true** or **false**.

Use **EndsWith** and **StartsWith** with the **Filter** function to search the data within your app. You can also use the **in** operator or the **Search** function to look anywhere within text strings, not just at the beginning or end. Your choice of functions will depend on the needs of your app and which function can be delegated for your particular data source. If one of these functions can't be delegated, a delegation warning will appear at authoring time to warn you of this limitation.

Syntax

EndsWith(*Text*, *EndText*)

- *Text* – Required. The text to test.
- *EndText* – Required. The text to search for at the end of *Text*. If *EndText* is an empty string, **EndsWith** returns *true*.

StartsWith(*Text*, *StartText*)

- *Text* – Required. The text to test.
- *StartText* – Required. The text to search for at the beginning of *Text*. If *StartText* is an empty string, **StartsWith** returns *true*.

Examples

Expand table

Formula	Description	Result
EndsWith("Hello World", "world")	Tests whether **"Hello World"** ends with **"world"**. The test is case insensitive.	**true**
EndsWith("Good bye", "good")	Tests whether **"Good bye"** ends with **"good"**. The *EndText* argument (**"good"**) appears in the text but not at the end.	**false**
EndsWith("Always say hello", "hello")	Tests whether **"Always say hello"** ends with **"hello"**.	**true**
EndsWith("Bye bye", "")	Tests whether **"Bye bye"** ends with an empty text string (**Len** returns 0). Easing its use in **Filter** expressions, **EndsWith** is defined to return **true** in this case.	**true**

Expand table

Formula	Description	Result
StartsWith("Hello World", "hello")	Tests whether **"Hello World"** begins with **"hello"**. The test is case insensitive.	**true**
StartsWith("Good bye", "hello")	Tests whether **"Good bye"** begins with **"hello"**.	**false**
StartsWith("Always say hello", "hello")	Tests whether **"Always say hello"** begins with **"hello"**. Although **"hello"** appears in the text, it doesn't appear at the beginning.	**false**
StartsWith("Bye bye", "")	Tests whether **"Bye bye"** starts with an empty text string (**Len** returns 0). Easing its use in **Filter** expressions, **StartsWith** is defined to return **true** in this case.	**true**

Search user experience

In many apps, you can type one or more characters into a search box to filter a list of records in a large data set. As you type, the list shows only those records that match the search criteria.

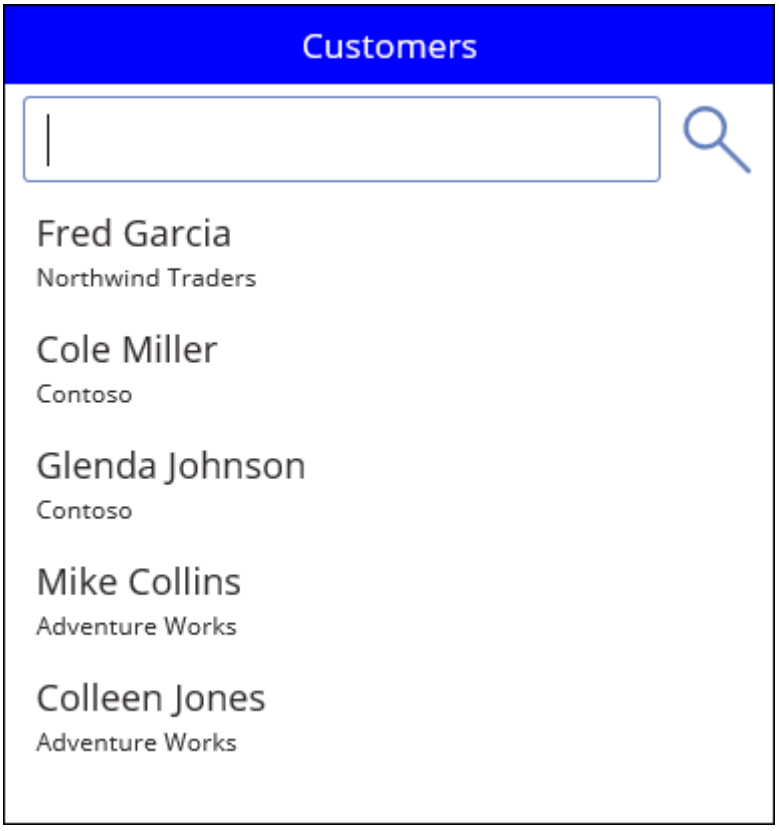
The examples in the rest of this topic show the results of searching a **Customers** list that contains this data:

Name	Company
"Fred Garcia"	"Northwind Traders"
"Cole Miller"	"Contoso"
"Glenda Johnson"	"Contoso"
"Mike Collins"	"Adventure Works"
"Colleen Jones"	"Adventure Works"

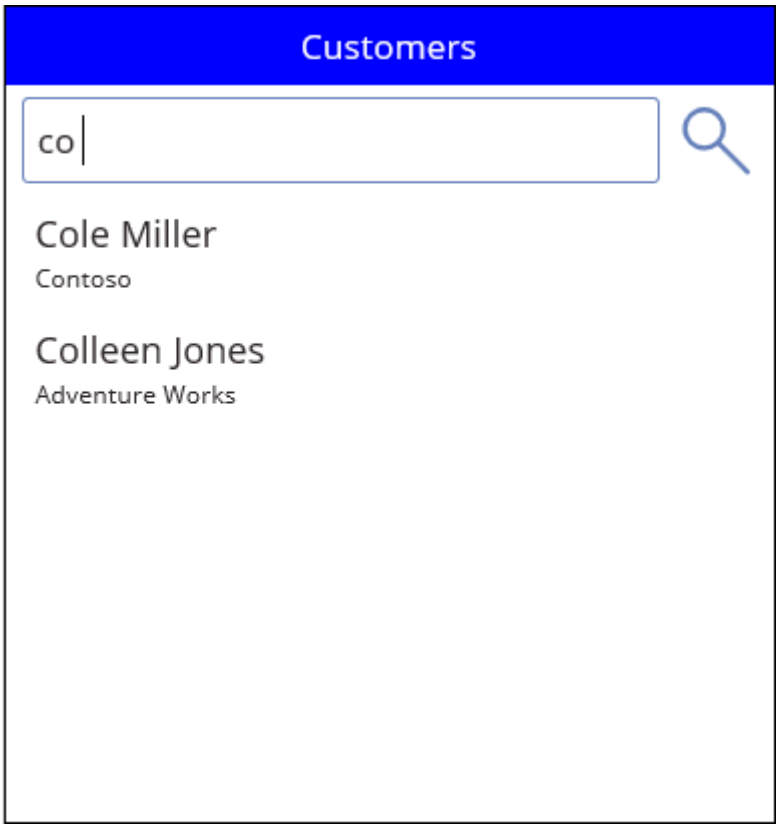
To create this data source as a collection, create a **Button** control and set its **OnSelect** property to this formula:

ClearCollect(Customers, Table({ Name: "Fred Garcia", Company: "Northwind Traders" }, { Name: "Cole Miller", Company: "Contoso" }, { Name: "Glenda Johnson", Company: "Contoso" }, { Name: "Mike Collins", Company: "Adventure Works" }, { Name: "Colleen Jones", Company: "Adventure Works" }))

As in this example, you can show a list of records in a **Gallery control** at the bottom of a screen. Near the top of the screen, you can add a **Text input** control, named **SearchInput**, so that users can specify which records interest them.



As the user types characters in **SearchInput**, the results in the gallery are automatically filtered. In this case, the gallery is configured to show records for which the name of the customer (not the name of the company) starts with the sequence of characters in **SearchInput**.If the user types **co** in the search box, the gallery shows these results:



To filter based on the **Name** column, set the **Items** property of the gallery control to one of these formulas:

Expand table

Formula	Description	Result
Filter(Customers, StartsWith(Name, SearchInput.Text))	Filters the **Customers** data source for records in which the search string appears at the start of the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones** and **Cole Miller**. The gallery doesn't show **Mike Collins** because the **Name** column for that record doesn't start with the search string.	
Filter(Customers, SearchInput.Text in Name)	Filters the **Customers** data source for records in which the search string appears anywhere in the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones**, **Cole Miller,** and **Mike Collins** because the search string appears somewhere in the **Name** column of all of those records.	
Search(Customers, SearchInput.Text, "Name")	Similar to using the **in** operator, the **Search** function searches for a match anywhere within the **Name** column of each record. Note that you must enclose the column name in double quotation marks.	

You can expand your search to include the **Company** column as well as the **Name** column:

Expand table

Formula	Description	Result
Filter(Customers, StartsWith(Name, SearchInput.Text) || StartsWith(Company, SearchInput.Text))	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column starts with the search string (for example, **co**). The || operator is *true* if either **StartsWith** function is *true*.	
Filter(Customers, SearchInput.Text in Name || SearchInput.Text in Company)	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it.	
Search(Customers, SearchInput.Text, "Name", "Company")	Similar to using the **in** operator, the **Search** function searches the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it. The **Search** function is easier to read and write than **Filter** if you want to specify multiple columns and multiple **in** operators. Note that you must enclose the names of the columns in double quotation marks.	

Error, IfError, IsError, IsBlankOrError functions

Article • 10/16/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Detects errors and provides an alternative value or takes action. Create a custom error or pass through an error.

> **Note**
>
> If an app has disabled the *Formula-level error management* feature under **Settings** > **Updates** > **Retired**, those functions will not work correctly.

IfError

The **IfError** function tests values until it finds an error. If the function discovers an error, the function evaluates and returns a corresponding replacement value and stops further evaluation. A default value can also be supplied for when no errors are found. The structure of **IfError** resembles that of the **If** function: **IfError** tests for errors, while **If** tests for **true**.

Use **IfError** to replace an error with a valid value so that downstream calculations can continue. For example, use this function if user input might result in a division by zero:

Power Apps

```
IfError( 1/x, 0 )
```

This formula returns `0` if the value of `x` is zero, as `1/x` produces an error. If `x` isn't zero, then `1/x` is returned.

Stopping further processing

When chaining formulas together in behavior formulas, such as:

Power Apps

```
Patch( DS1, ... );
Patch( DS2, ... )
```

The second Patch function to `DS2` will be attempted even if the **Patch** to `DS1` fails. The scope of an error is limited to each formula that is chained.

Use **IfError** to do an action and only continue processing if the action was successful. Applying **IfError** to this example:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" ),
    Patch( DS2, ... ), Notify( "problem in the second action" )
)
```

If the **Patch** of `DS1` has a problem, the first **Notify** is executed. No further processing occurs including the second **Patch** of `DS2`. If the first **Patch** succeeds, the second **Patch** will execute.

If supplied, the optional *DefaultResult* argument is returned if no errors are discovered. Without this argument, the last *Value* argument is returned.

Building on the last example, the return value from **IfError** can be checked to determine if there were any problems:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" );  false,
    Patch( DS2, ... ), Notify( "problem in the second action" ); false,
    true
)
```

Type compatibility

IfError returns the value of one of its arguments. The types of all values that might be returned by **IfError** must be compatible.

In the last example, **Patch** returns a record that isn't compatible with the Booleans used for the *Replacement* formulas or the *DefaultResult*. Which is fine, since there's no situation in which the return value from these **Patch** calls would be returned by **IfError**.

> ⓘ **Note**
>
> While the behavior in process for a change, the types of all arguments to **IfError** must be compatible currently.

In the simple example described earlier:

```
Power Apps
IfError( 1/x, 0 )
```

The types of `1/x` and `0` were compatible as both were numbers. If they're not, the second argument is coerced to match the type of the first argument.

Excel displays **#DIV/0!** when a division by zero occurs.

Consider **IfError** with the following instead:

```
Power Apps
IfError( 1/x, "#DIV/0!" )
```

The above formula won't work. The text string `"#DIV/0!"` is coerced to the type of the first argument to **IfError**, which is a number. The result of **IfError** is yet another error since the text string can't be coerced. As a fix, convert the first argument to a text string so that **IfError** always returns a text string:

```
Power Apps
IfError( Text( 1/x ), "#DIV/0!" )
```

As seen above, **IfError** can return an error if the *Replacement* or *DefaultResult* is an error.

FirstError / AllErrors

Within in the replacement formulas, information about the errors found is available through the **FirstError** record and **AllErrors** table. **AllErrors** is a table of error information records with **FirstError** being a shortcut to the first record of this table. **FirstError** always returns the same value as **First(AllErrors)**.

Error records include:

Expand table

Field	Type	Description
Kind	ErrorKind enum (number)	Category of the error.
Message	Text string	Message about the error, suitable to be displayed to the end user.
Source	Text string	Location in where the error originated, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Observed	Text string	Location in where the error is surfaced to the user, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Details	Record	Details about the error. Currently, details are provided only for network errors. This record includes **HttpStatusCode** which contains the HTTP status code and **HttpResponse** which contains the body of the response from the connector or service.

For example, consider the following formula as a **Button** control's **OnSelect** property:

Power Apps

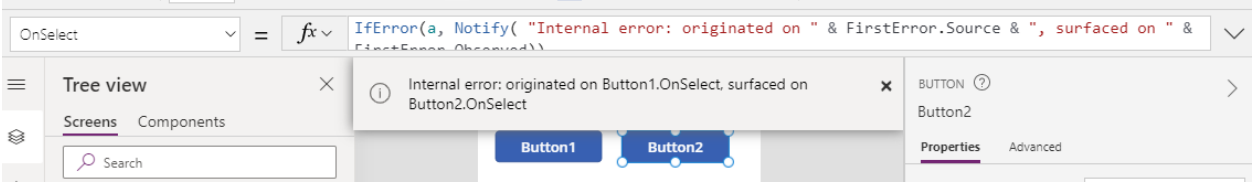
```
Set( a, 1/0 )
```

And this formula on the **OnSelect** property of a second **Button** control:

Power Apps

```
IfError( a, Notify( "Internal error: originated on " & FirstError.Source &
", surfaced on " & FirstError.Observed ) )
```

The example formula above would display the following banner when the two buttons are activated in sequence:



Typically, there will be only one error that **FirstError** can sufficiently work with. However, there are scenarios where multiple errors may be returned. For example, when using a formula chaining operator or the **Concurrent** function. Even in these situations,

reporting **FirstError** might be enough to reveal a problem instead overloading a user with multiple errors. If you still have a requirement to work with each error individually, you can use the **AllErrors** table.

IsError

The **IsError** function tests for an error value.

The return value is a Boolean *true* or *false*.

Using **IsError** prevents any further processing of the error.

IsBlankOrError

The **IsBlankOrError** function tests for either a blank value or an error value and is the equivalent of `Or( IsBlank( X ), IsError( X ) )`.

When enabling error handling for existing apps, consider replacing **IsBlank** with **IsBlankOrError** to preserve existing app behavior. Before the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value is a boolean *true* or *false*.

Using **IsBlankOrError** prevents any further processing of the error.

Use the **Error** function to create and report a custom error. For example, you might have logic to determine whether any given value is valid for your context or not—something not checked for a problem automatically. You can create and return your own error, complete with **Kind** and **Message**, using the same record described above for the **IfError** function.

In the context of **IfError**, use the **Error** function to rethrow or pass through an error. For example, your logic in **IfError** may decide that in some cases an error can be safely ignored, but in other cases the error is important to send through. Within **IfError** or **App.OnError**, use **Error(FirstError)** to pass through an error.

The **Error** function can also be passed a table of errors, as would be found in the **AllErrors** table. Use **Error(AllErrors)** to rethrow all the errors and not just the first.

A *blank* record or empty table passed to **Error** results in no error.

Syntax

Error(**ErrorRecord**)
Error(**ErrorTable**)

- *ErrorRecord* – Required. Error information record, including **Kind**, **Message**, and other fields. **Kind** is required. **FirstError** can be passed directly.
- *ErrorTable* – Required. Table of error information records. **AllErrors** can be passed directly.

IfError(*Value1*, *Replacement1* [, *Value2*, *Replacement2*, ... [, *DefaultResult*]])

- *Value(s)* – Required. Formula(s) to test for an error value.
- *Replacement(s)* – Required. The formulas to evaluate and values to return if matching *Value* arguments returned an error.
- *DefaultResult* – Optional. The formulas to evaluate if the formula doesn't find any errors.

IsError(*Value*)
IsBlankOrError(*Value*)

- *Value* – Required. Formula to test.

Examples

Simple IfError

Expand table

Formula	Description	Result
IfError(1, 2)	The first argument isn't an error. The function has no other errors to check and no default return value. The function returns the last *value* argument evaluated.	1
IfError(1/0, 2)	The first argument returns an error value (because of division by zero). The function evaluates the second argument and returns it as the result.	2
IfError(10, 20, 30)	The first argument isn't an error. The function has no other errors to check but does have a default return value. The function returns the *DefaultResult* argument.	30
IfError(10, 11, 20, 21, 300)	The first argument **10** isn't an error, so the function doesn't evaluate that argument's corresponding replacement **11**. The third argument	300

Formula	Description	Result
	20 isn't an error either, so the function doesn't evaluate that argument's corresponding replacement **21**. The fifth argument **300** has no corresponding replacement and is the default result. The function returns that result because the formula contains no errors.	
IfError(1/0, Notify("There was an internal problem"))	The first argument returns an error value (due to division by zero). The function evaluates the second argument and displays a message to the user. The return value of **IfError** is the return value of **Notify**, coerced to the same type as the first argument to **IfError** (a number).	1

Simple IsError

Expand table

Formula	Description	Result
IsError(1)	The argument isn't an error.	*false*
IsError(Blank())	The argument is a blank, but not an error.	*false*
IsError(1/0)	The argument is an error.	*true*
If(IsError(1/0), Notify("There was an internal problem"))	The argument to **IsError** returns an error value (because of division by zero). This function returns *true*, which causes the **If** to display a message to the user with the **Notify** function. The return value of **If** is the return value of **Notify**, coerced to the same type as the first argument to **If** (a boolean).	*true*

Simple IsBlankOrError

Expand table

Formula	Description	Result
IsBlankOrError(1)	The argument isn't an error or a blank.	*false*
IsBlankOrError(Blank())	The argument is a blank.	*true*
IsBlankOrError(1/0)	The argument is an error.	*true*

Simple Error

In this example, dates are validated relative to one another, resulting in an error if there is a problem.

Power Apps

```
If( StartDate > EndDate,
    Error( { Kind: ErrorKind.Validation, Message: "Start Date must be before
End Date" } ) )
```

In this example, some errors are allowed to pass through while others are suppressed and replaced with a value. In the first case, **b** is in an error state because the **Value** function has an invalid argument. Because this is unexpected by the formula writer, it is passed through so the user sees it. In the second case, with the same formula, **b** has the value 0, resulting in a division by zero. In this case, the formula writer may know that this is acceptable for this logic, suppress the error (no banner is shown), and return -1 instead.

Power Apps

```
With( {a: 1, b: Value("a")},
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns an error with Kind = ErrorKind.InvalidArgument

With( {a: 1, b: 0} )
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns -1
```

The **AllErrors** table can be filtered like any other table. Used with the **Error** function, expected errors can be removed and the remaining errors retained and reported. For example, if we knew that division by zero wasn't going to be a problem in a particular context, those errors could be filtered out, leaving all other errors intact with the following formula:

Power Apps

```
Error( Filter( AllErrors, Kind <> ErrorKind.Div0 ) )
```

Step by step

1. Add a Text input control, and name it **TextInput1** if it doesn't have that name by default.

2. Add a Label control, and name it **Label1** if it doesn't have that name by default.

3. Set the formula for **Label1**'s **Text** property to:

Power Apps

```
IfError( Value( TextInput1.Text ), -1 )
```

4. In **TextInput1**, enter **1234**.

   Label1 shows the value **1234** as this is a valid input to the Value function.

5. In **TextInput1**, enter **ToInfinity**.

   Label1 shows the value **-1** as this isn't a valid input to the Value function. Without wrapping the Value function with IfError, the label would show no value as the error value is treated as a *blank*.

See also

Formula reference for Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Errors function

10/10/2025

Applies to: ✅ Canvas apps ✅ Model-driven apps

Provides error information for previous changes to a data source.

Overview

Errors can happen when a record of a data source is changed. Many causes are possible, including network outages, inadequate permissions, and edit conflicts.

The functions that modify data in data sources, such as **Patch**, **Collect**, **Remove**, **RemoveIf**, **Update**, **UpdateIf**, and **SubmitForm** report errors in two ways:

- Each of these functions returns an error value as the result of the operation. Errors can be detected with **IsError** and replaced or suppressed with **IfError** and **App.OnError** as usual. See Error Handling for more information.
- After the operation, the **Errors** function will also return the errors for previous operations. It can be useful for displaying the error message on a form screen without needing to capture the error in a state variable.

You can avoid some errors before they happen by using the **Validate** and **DataSourceInfo** functions. See working with data sources for more suggestions on how to work with and avoid errors.

Description

The **Errors** function returns a table of errors that contains the following columns:

- **Record**. The record in the data source that had the error. If the error occurred during the creation of a record, this column is *blank*.
- **Column**. The column that caused the error, if the error can be attributed to a single column. If not, it's *blank*.
- **Message**. A description of the error. This error string can be displayed for the end user. This message may be generated by the data source and could be long and contain raw column names that may not have any meaning to the user.
- **Error**. An error code that can be used in formulas to help resolve the error:

Expand table

ErrorKind	Description
ErrorKind.Conflict	Another change was made to the same record, resulting in a change conflict. Use the **Refresh** function to reload the record and try the change again.
ErrorKind.ConstraintViolation	One or more constraints have been violated.
ErrorKind.CreatePermission	An attempt was made to create a record, and the current user doesn't have permission to create records.
ErrorKind.DeletePermission	An attempt was made to delete a record, and the current user doesn't have permission to delete records.
ErrorKind.EditPermission	An attempt was made to edit a record, and the current user doesn't have permission to edit records.
ErrorKind.GeneratedValue	An attempt was made to change a column that the data source generates automatically.
ErrorKind.MissingRequired	The value for a required column is missing from the record.
ErrorKind.None	There's no error.
ErrorKind.NotFound	An attempt was made to edit or delete a record, but the record couldn't be found. Another user may have changed the record.
ErrorKind.ReadOnlyValue	An attempt was made to change a column that's read only.
ErrorKind.Sync	An error was reported by the data source. Check the Message column for more information.
ErrorKind.Unknown	There was an error, but of an unknown kind.
ErrorKind.Validation	There was a general validation issue detected, that didn't fit one of the other kinds.

Errors can be returned for the entire data source, or for only a selected row by providing the *Record* argument to the function.

Patch or another data function may return a *blank* value if, for example, a record couldn't be created. You can pass *blank* to **Errors**, and it returns appropriate error information in these cases. Subsequent use of data functions on the same data source clears this error information.

If there are no errors, the table that **Errors** returns will be empty and can be tested with the **IsEmpty** function.

Syntax

Errors(*DataSource* [, *Record*])

- *DataSource* – Required. The data source for which you want to return errors.
- *Record* – Optional. A specific record for which you want to return errors. If you don't specify this argument, the function returns errors for the entire data source.

Examples

Step by Step

For this example, we'll be working with the **IceCream** data source, a data source for which Power Apps will attempt to detect conflicts:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200

> ⓘ **Note**
>
> Power Apps does not attempt to detect conflicts for changes to Dataverse tables.

Through the app, a user loads the Chocolate record into a data-entry form and then changes the value of **Quantity** to 90. The record to be worked with is placed in the context variable **EditRecord**:

- **UpdateContext({ EditRecord: LookUp(IceCream, Flavor = "Chocolate") })**

To make this change in the data source, the **Patch** function is used:

- **Patch(IceCream, EditRecord, Gallery.Updates)**

Where **Gallery.Updates** evaluates to **{ Quantity: 90 }**, since only the **Quantity** property has been modified.

Unfortunately, just before the **Patch** function was invoked, somebody else modifies the **Quantity** for Chocolate to 80. Because this is a supported data source for conflict detection, Power Apps will detect this and not allow the conflicting change to occur. You can check for this situation with the formula:

- **IsEmpty(Errors(IceCream, EditRecord))**

Which returns **false**, because the **Errors** function returned the following table:

Expand table

Record	Column	Message	Error
{ Flavor: "Chocolate", Quantity: 100 }	*blank*	"Another user has modified the record that you're trying to modify. Reload the record and try again."	ErrorKind.Conflict

You can place a label on the form to show this error to the user.

- To show the error, set the label's **Text** property to this formula:
  Label.Text = First(Errors(IceCream, EditRecord)).Message

You can also add a **Reload** button on the form, so that the user can efficiently resolve the conflict.

- To show the button only when a conflict has occurred, set the button's **Visible** property to this formula:
  !IsEmpty(Lookup(Errors(IceCream, EditRecord), Error = ErrorKind.Conflict))
- To revert the change, which the user selects the button, set its **OnSelect** property to this formula:
  ReloadButton.OnSelect = Revert(IceCream, EditRecord)

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

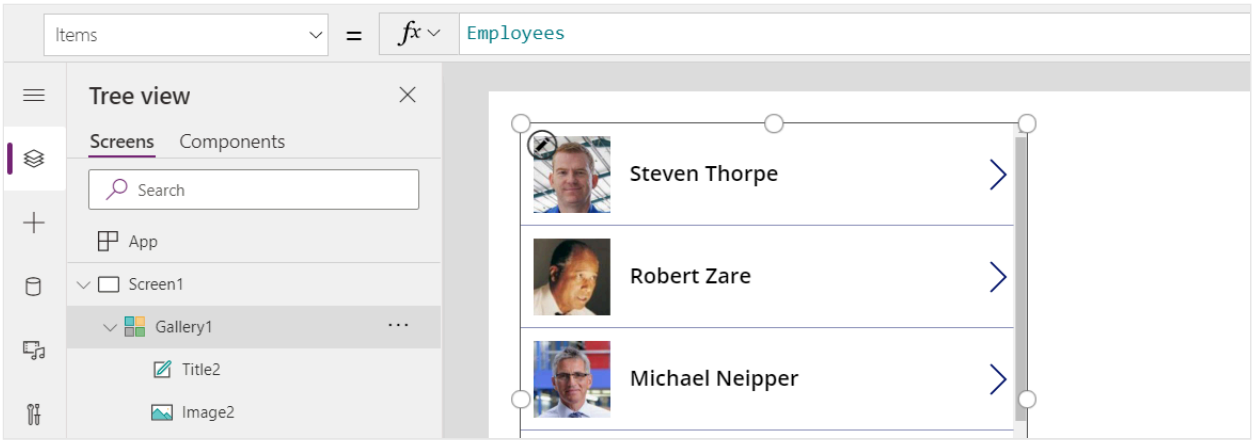
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

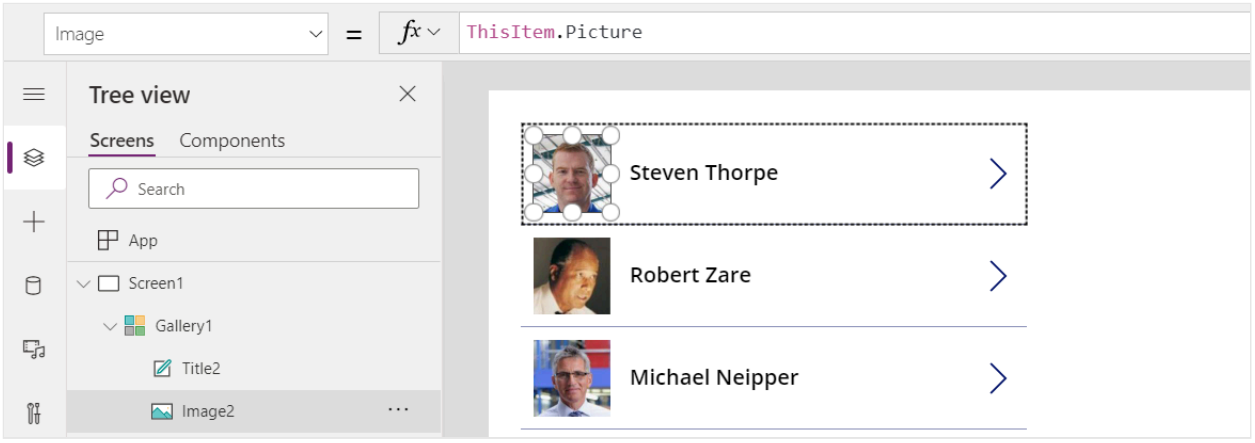
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

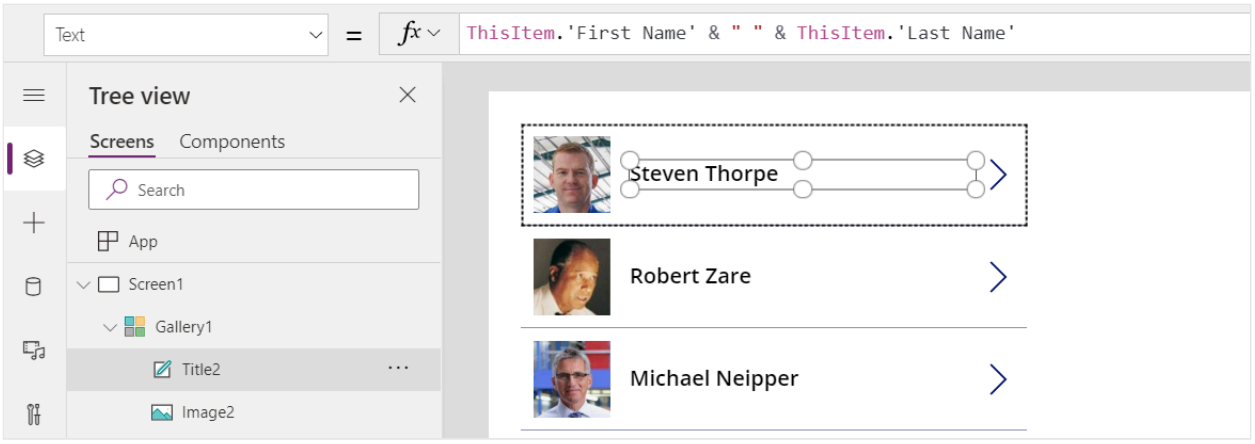
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

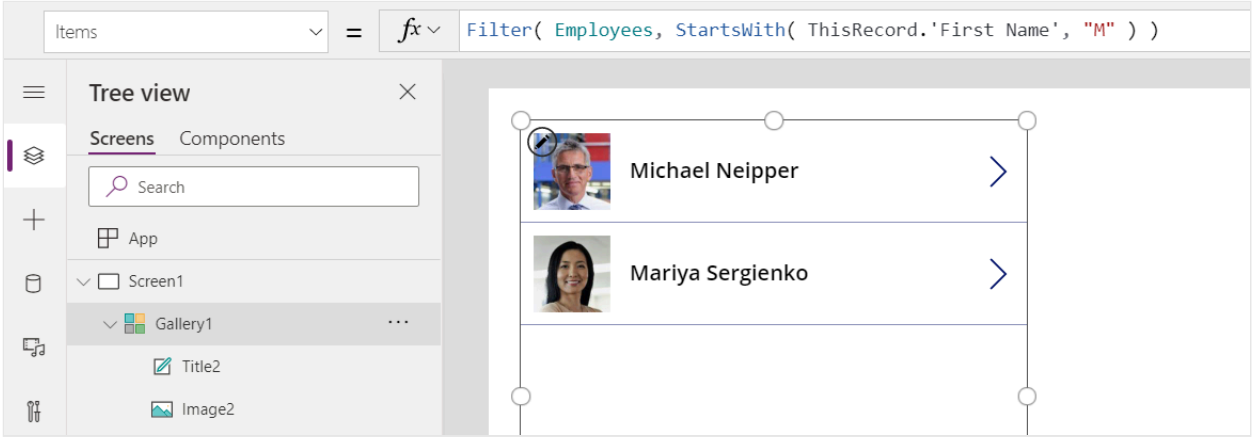


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

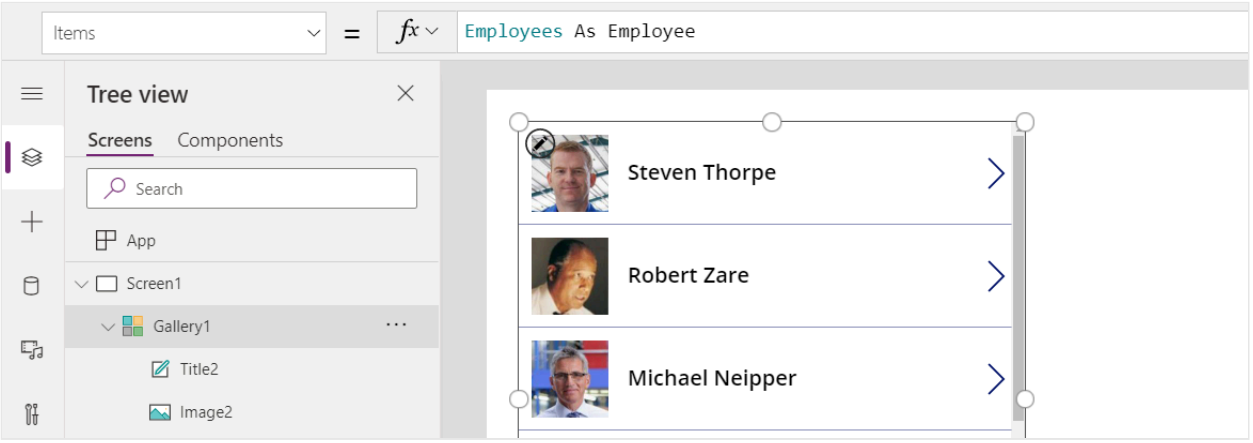
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

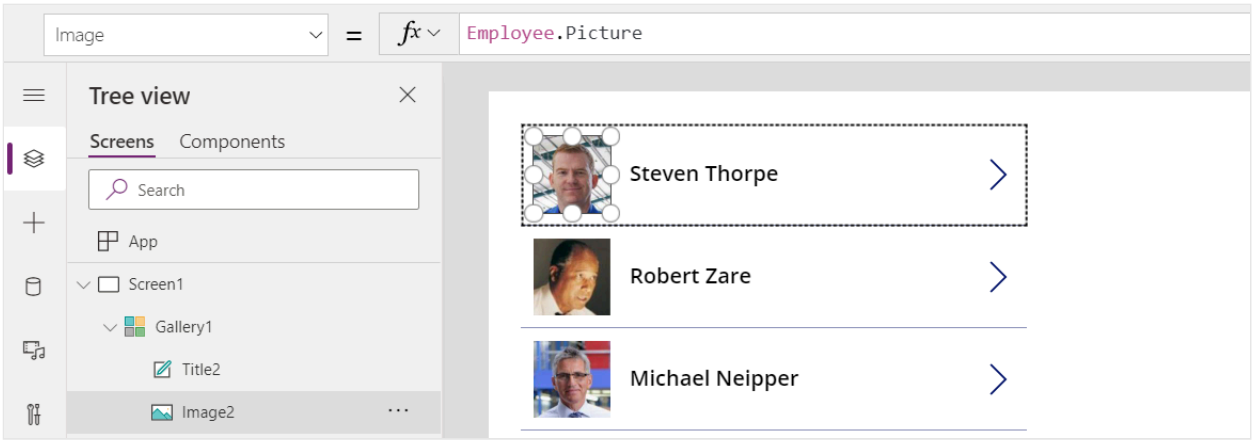
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

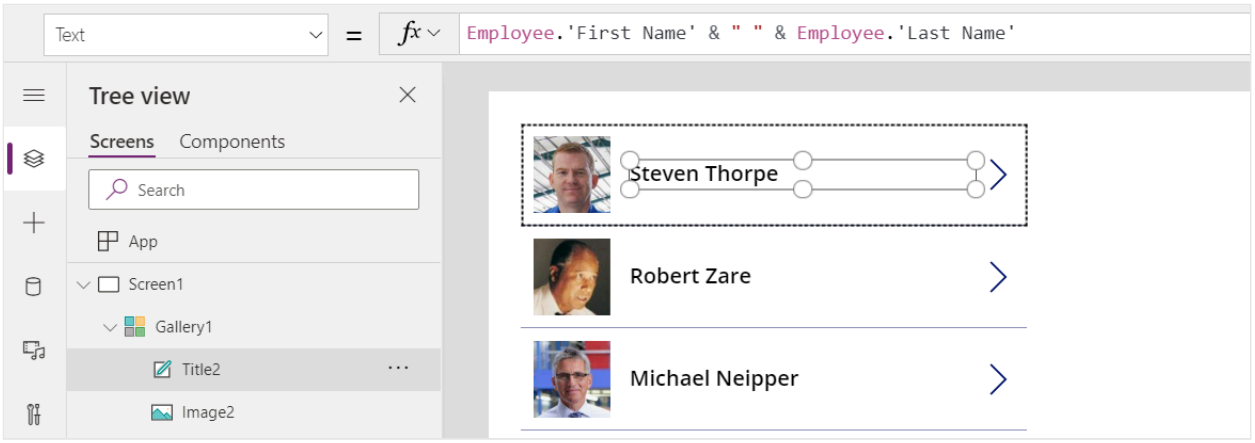
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

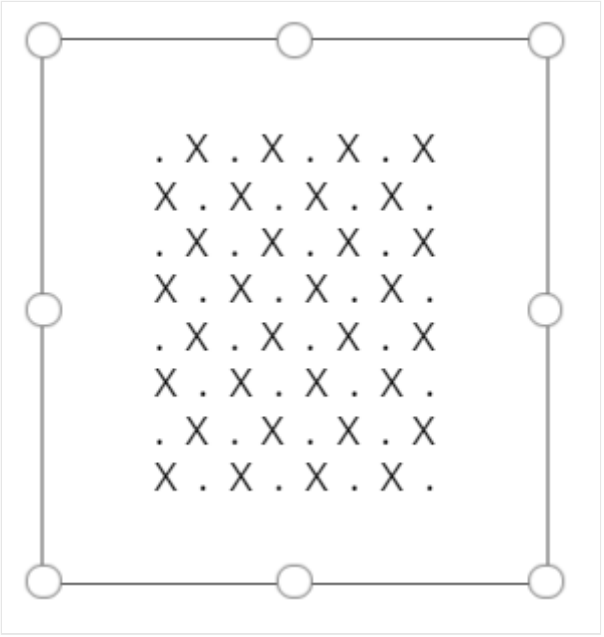
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



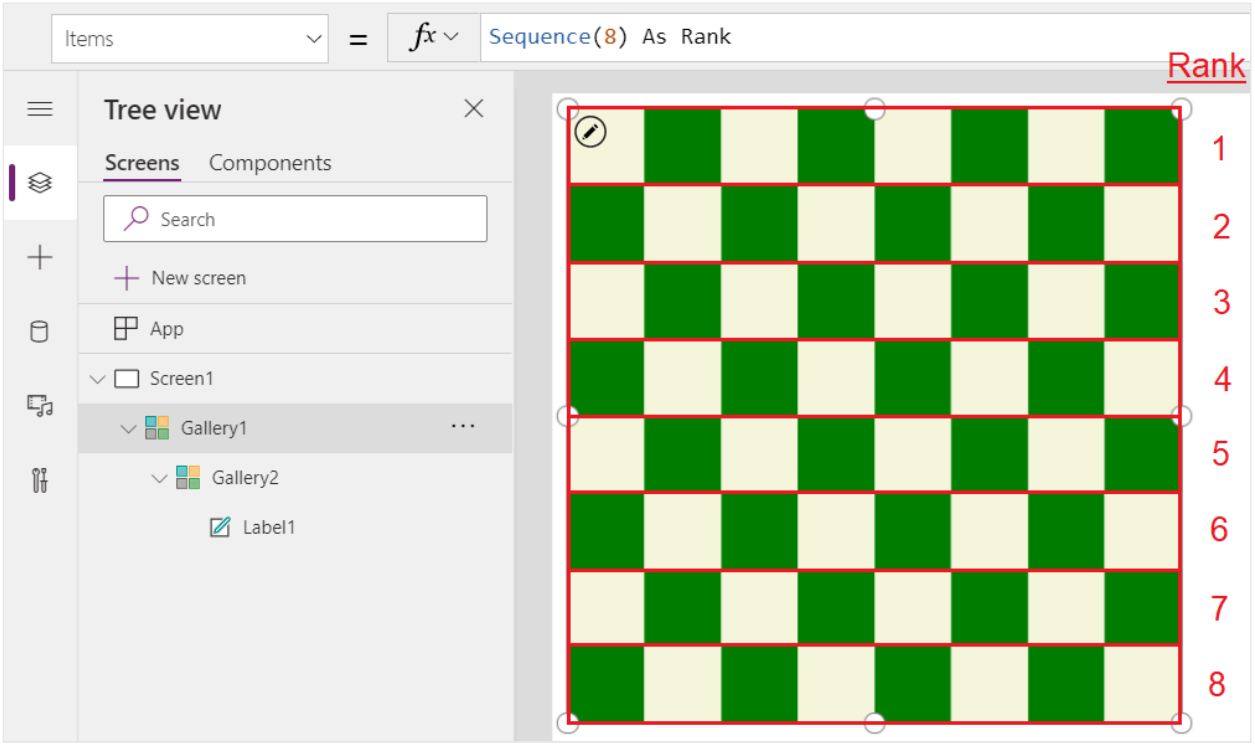
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

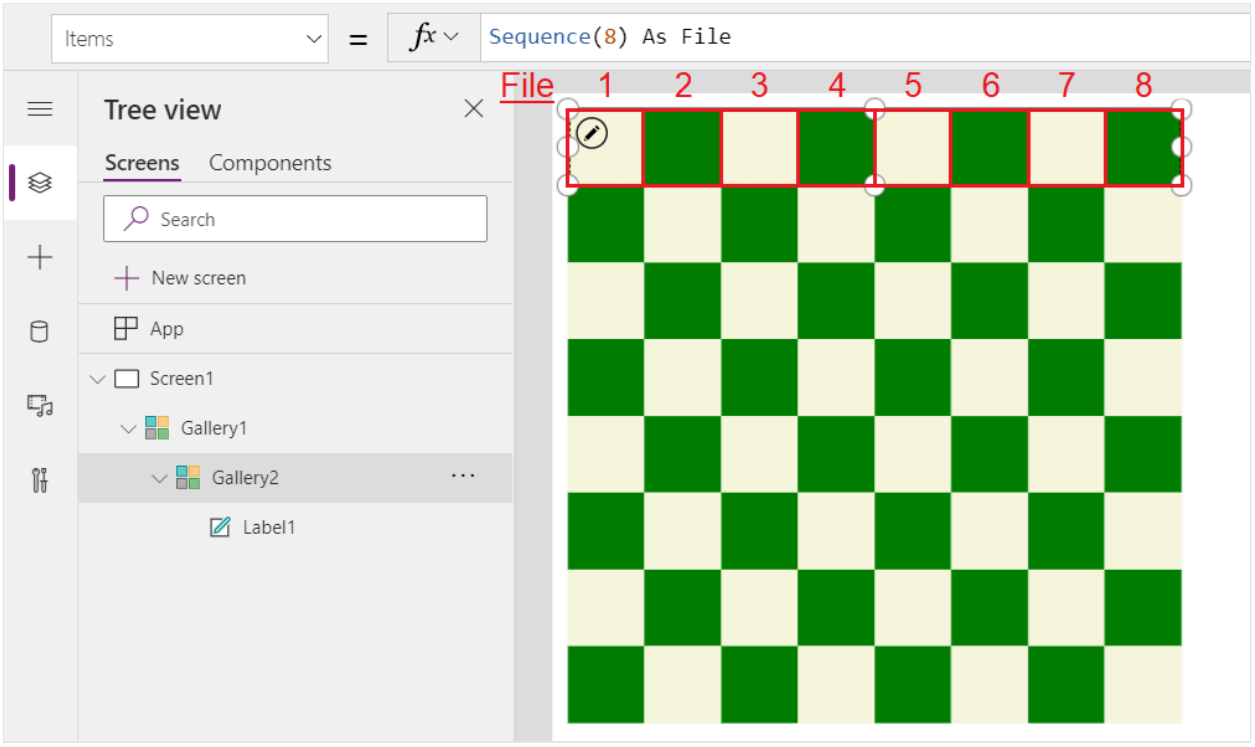
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

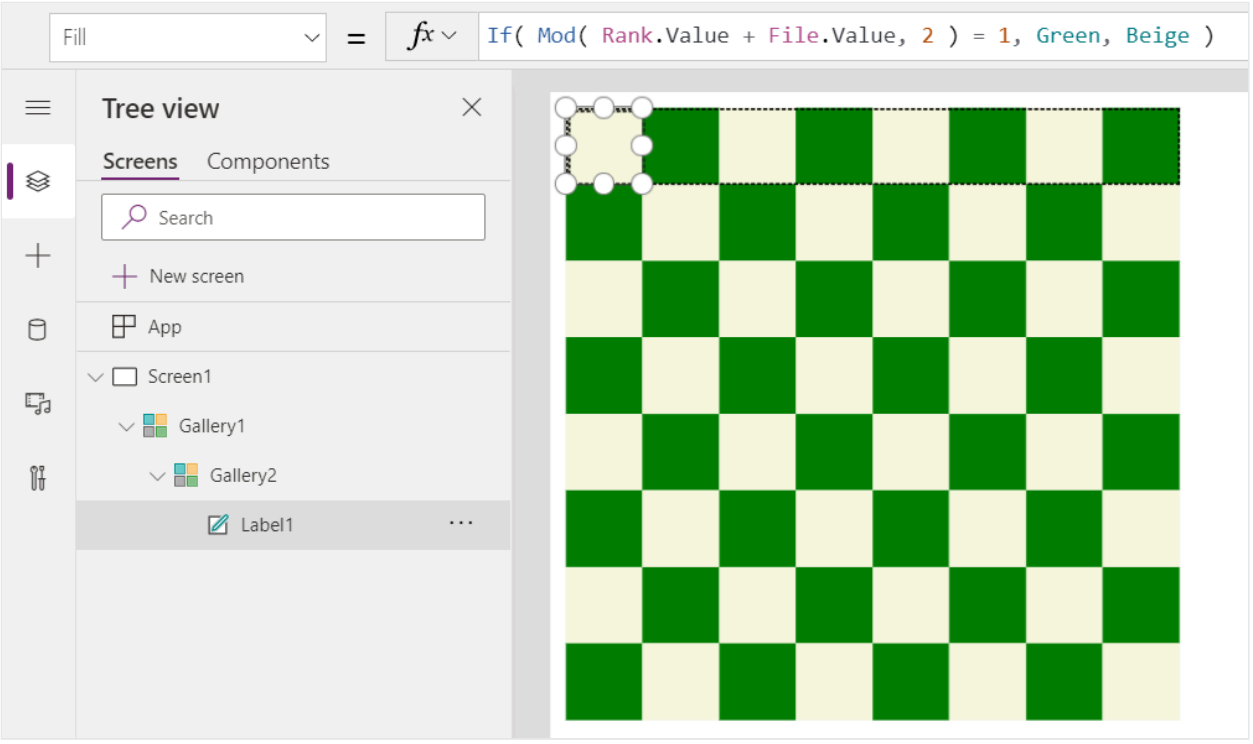
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

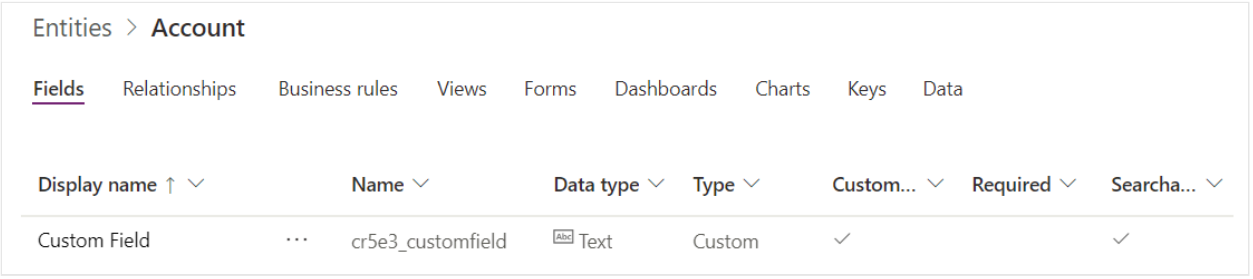
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

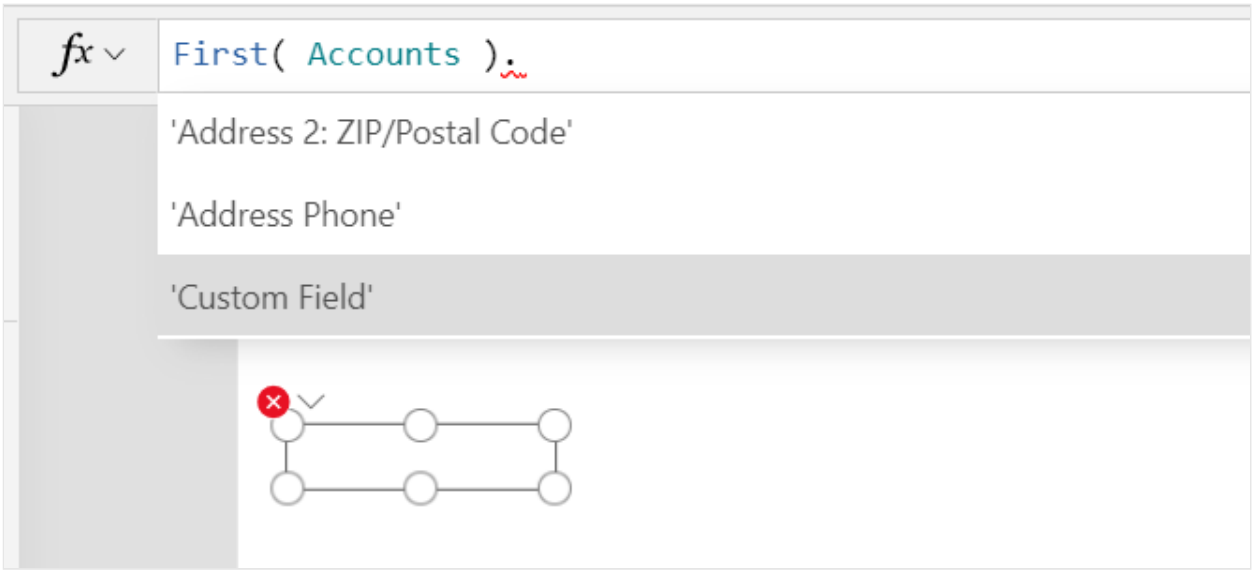
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

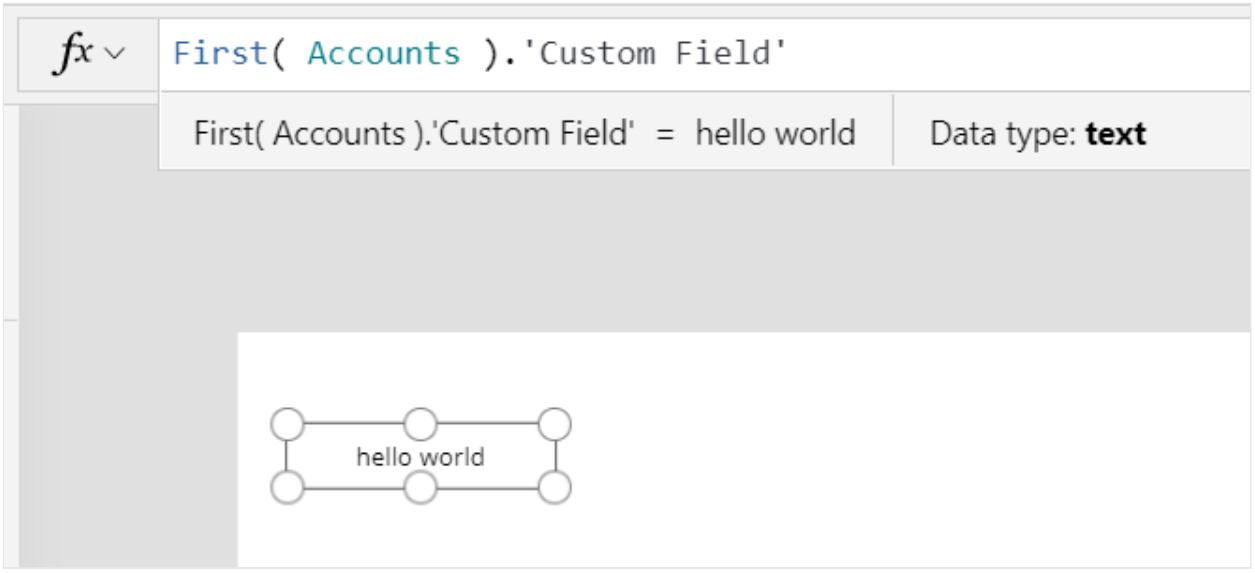
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



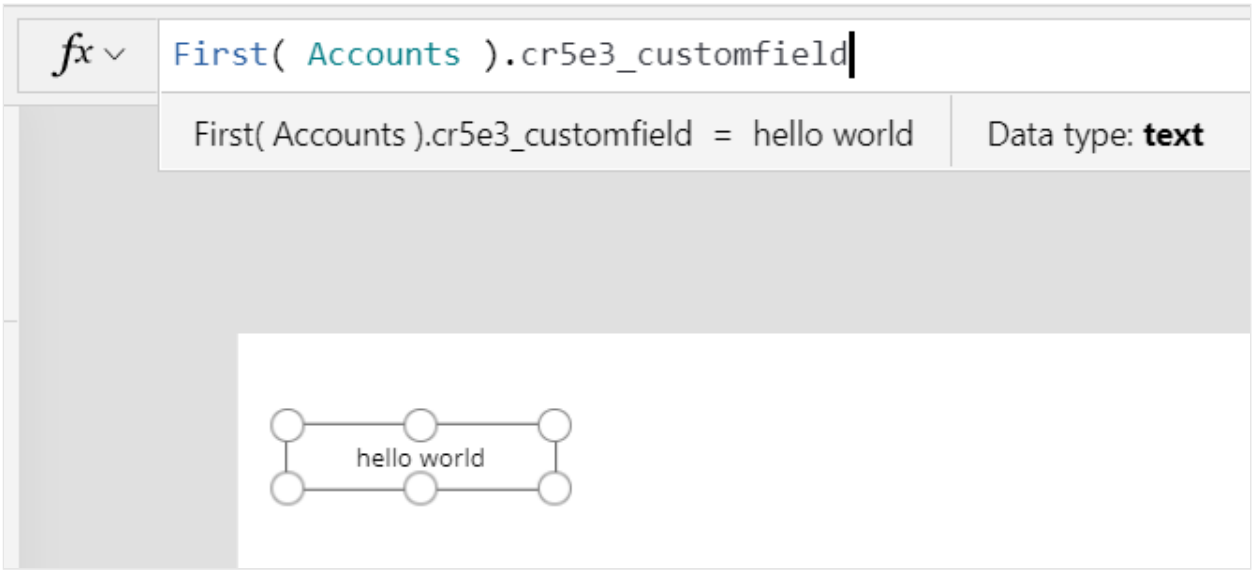
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

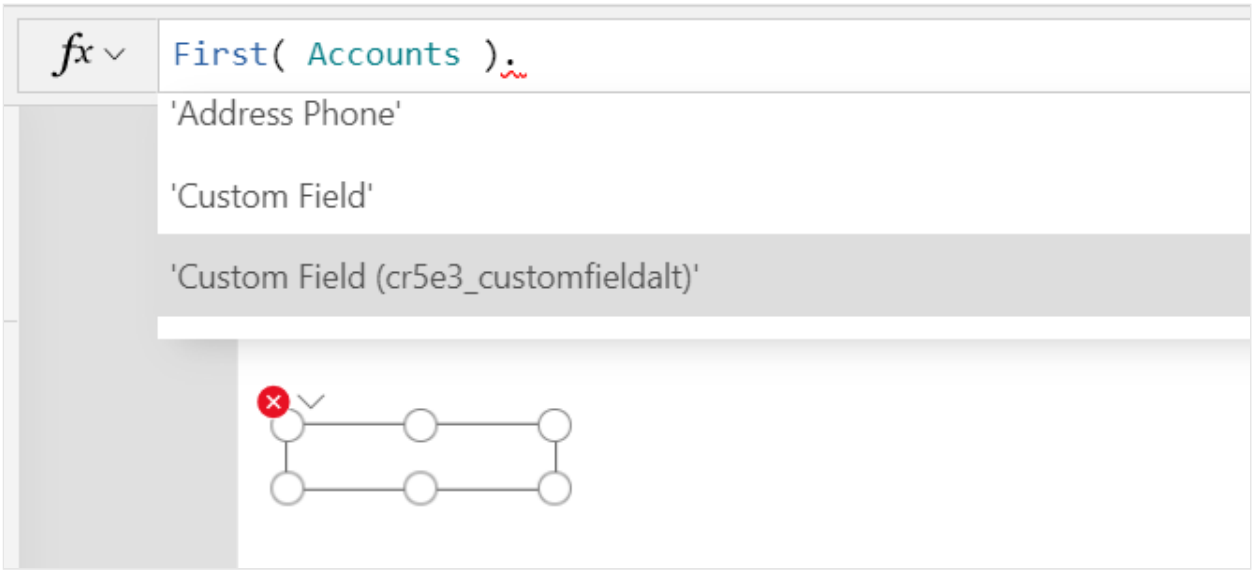


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

Exit function

10/10/2025

Applies to: ✅ Canvas apps ✅ Power Platform CLI

Exits the currently running app and optionally signs out the current user.

Description

The **Exit** function exits the currently running app. The user is returned to the list of apps. The user can then select another app to open.

Exit stops any further formula evaluation. Any function calls chained with a semicolon operator after the **Exit** aren't carried out.

Use the optional *Signout* argument to sign the current user out of Power Apps. *Signout* is useful when devices are shared to ensure user security.

While authoring the app, calling **Exit** doesn't exit or sign out the user. However, it stops the evaluation of the rest of the formula.

Exit can only be used in behavior formulas.

Syntax

Exit([*Signout*])

- *Signout* – Optional. A Boolean value that if *true* signs the current user out of Power Apps. The default is *false* and the user remains signed in.

Examples

Expand table

Formula	Description
Exit()	Exits the current app and leaves the user signed in. The user is returned to the list of apps.
Exit(true)	Exits the current app and the user is signed out. The user needs to sign back in with their credentials before running an app.

Abs, Exp, Ln, Power, Log, and Sqrt functions

Article • 04/20/2025

Expand table

Functions	Applies to
Abs **Exp** **Ln** **Power** **Sqrt**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Log	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Calculates absolute values, logarithms, square roots, and the results of raising *e* or any number to specified powers.

Description

The **Abs** function returns the non-negative value of its argument. If a number is negative, **Abs** returns the positive equivalent.

The **Exp** function returns *e* raised to the power of its argument. The transcendental number *e* begins 2.7182818...

The **Ln** function returns the natural logarithm (base *e*) of its argument.

The **Power** function returns a number raised to a power. It's equivalent to using the ^ operator.

The **Log** function returns the logarithm of its first argument in the base specified by its second argument (or 10 if not specified).

The **Sqrt** function returns the number that, when multiplied by itself, equals its argument.

If you pass a single number, the return value is a single result based on the function called. If you pass a single-column table that contains numbers, the return value is a single-column table of results in a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined valued, the result is *blank*. Which can happen with square roots and logarithms of negative numbers.

Syntax

Abs(*Number*)
Exp(*Number*)
Ln(*Number*)
Sqrt(*Number*)

- *Number* - Required. Number to operate on.

Power(*Base*, *Exponent*)

- *Base* - Required. Base number to raise.
- *Exponent* - Required. The exponent to which the base number is raised.

Log(*Number*, *Base*)

- *Number* - Required. Number to calculate the logarithm.
- *Base* - Optional. The base of the logarithm to calculate. By default, 10 (when not specified).

Abs(*SingleColumnTable*)
Exp(*SingleColumnTable*)
Ln(*SingleColumnTable*)
Sqrt(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Examples

Single number

Expand table

Formula	Description	Result
Abs(-55)	Returns the number without the negative sign.	55
Exp(2)	Returns *e* raised to the power of 2, or *e* * *e*.	7.389056...
Ln(100)	Returns the natural logarithm (base *e*) of the number 100.	4.605170...
Log(100)	Returns the logarithm in base 10 of the number 100.	2
Log(64, 2)	Returns the logarithm in base 2 of the number 64.	6

Formula	Description	Result
Power(5, 3)	Returns 5 raised to the power of 3, or 5 * 5 * 5.	125
Sqrt(9)	Returns the number that, when multiplied by itself, results in 9.	3

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains this data:

Expand table

Value
9
-4
2

Expand table

Formula	Description	Result
Abs(ValueTable)	Returns the absolute value of each number in the table.	A single-column table with a `Value` column containing the following values: 9, 4, 2
Exp(ValueTable)	Returns *e* raised to the power of each number in the table.	A single-column table with a `Value` column containing the following values: 8103.083927..., 0.018315..., 7.389056...
Ln(ValueTable)	Returns the natural logarithm of each number in the table.	A single-column table with a `Value` column containing the following values: 2.197224..., Blank(), 0.693147...
Sqrt(ValueTable)	Returns the square root of each number in the table	A single-column table with a `Value` column containing the following values: 3, Blank(), 1.414213...

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a **Label** control, and set its **Text** property to this formula:
   Sqrt(Value(Source.Text))

3. Type a number into **Source**, and confirm that the **Label** control shows the square root of the number that you typed.

Filter, Search, and LookUp functions

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Finds one or more records in a table.

Watch this video to learn how to use **Filter**, **Search**, and **LookUp** functions:

https://learn-video.azurefd.net/vod/player?id=55b2c144-b5ac-4d66-b398-fea03eb21f17&locale=en-us&embedUrl=%2Fpower-platform%2Fpower-fx%2Freference%2Ffunction-filter-lookup

> **Note**
>
> PAC CLI pac power-fx commands do not support the **Search** function.

Description

The **Filter** function finds records in a table that satisfy a formula. Use **Filter** to find a set of records that match one or more criteria and discard those records that don't.

The **LookUp** function finds the first record in a table that satisfies a formula. Use **LookUp** to find a single record that matches one or more criteria.

For both, the formula is evaluated for each record of the table. Records that result in *true* are included in the result. Besides the normal formula operators, you can use the **in** and **exactin** operators for substring matches.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

The **Search** function finds records in a table that contain a string in one of their columns. The string might occur anywhere within the column; for example, searching for "rob" or "bert" would find a match in a column that contains "Robert". Searching is case-insensitive. Unlike **Filter** and **LookUp**, the **Search** function uses a single string to match instead of a formula.

Filter and **Search** return a table that contains the same columns as the original table and the records that match the criteria. **LookUp** returns only the first record found, after applying a

formula to reduce the record to a single value. If no records are found, **Filter** and **Search** return an empty table, and **LookUp** returns *blank*.

Tables are a value in Power Apps, just like a string or number. They can be passed to and returned from functions. **Filter**, **Search**, and **LookUp** don't modify a table. Instead, they take a table as an argument and return a table, a record, or a single value from it. See working with tables for more details.

Delegation

When possible, Power Apps will delegate filter and sort operations to the data source and page through the results on demand. For example, when you start an app that shows a **Gallery** control filled with data, only the first set of records will be initially brought to the device. As the user scrolls, additional data is brought down from the data source. The result is a faster start time for the app and access to very large data sets.

However, delegation may not always be possible. Data sources vary on what functions and operators they support with delegation. If complete delegation of a formula isn't possible, the authoring environment will flag the portion that can't be delegated with a warning. When possible, consider changing the formula to avoid functions and operators that can't be delegated. The delegation list details which data sources and operations can be delegated.

If delegation is not possible, Power Apps will pull down only a small set of records to work on locally. Filter and sort functions will operate on a reduced set of records. What is available in the **Gallery** may not be the complete story, which could be confusing to users.

See the delegation overview for more information.

Syntax

Filter(Table*, *Formula1* [, *Formula2*, ...])

- *Table* - Required. Table to search.
- *Formula(s)* - Required. The formula by which each record of the table is evaluated. The function returns all records that result in **true**. You can reference columns within the table. If you supply more than one formula, the results of all formulas are combined with the **And** function.

Search(Table*, *SearchString*, *Column1* [, *Column2*, ...])

- *Table* - Required. Table to search.

- *SearchString* - Required. The string to search for. If *blank* or an empty string, all records are returned.
- *Column(s)* - Required. The names of columns within *Table* to search. If *SearchString* is found within the data of any of these columns as a partial match, the full record will be returned.

> **Note**
>
> In Power Apps prior to version 3.24042, column names for the **Search** function were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

LookUp(Table*, *Formula* [, *ReductionFormula*])

- *Table* - Required. Table to search. In the UI, the syntax is shown as *source* above the function box.
- *Formula* - Required. The formula by which each record of the table is evaluated. The function returns the first record that results in **true**. You can reference columns within the table. In the UI, the syntax is shown as *condition* above the function box.
- *ReductionFormula* - Optional. This formula is evaluated over the record that was found, and then reduces the record to a single value. You can reference columns within the table. If you don't use this parameter, the function returns the full record from the table. In the UI, the syntax is shown as *result* above the function box.

Examples

The following examples use the **IceCream** data source:

Flavor	Quantity	OnOrder
"Chocolate"	100	75
"Vanilla"	200	50
"Strawberry"	300	0
"Mint Chocolate"	60	100

Expand table

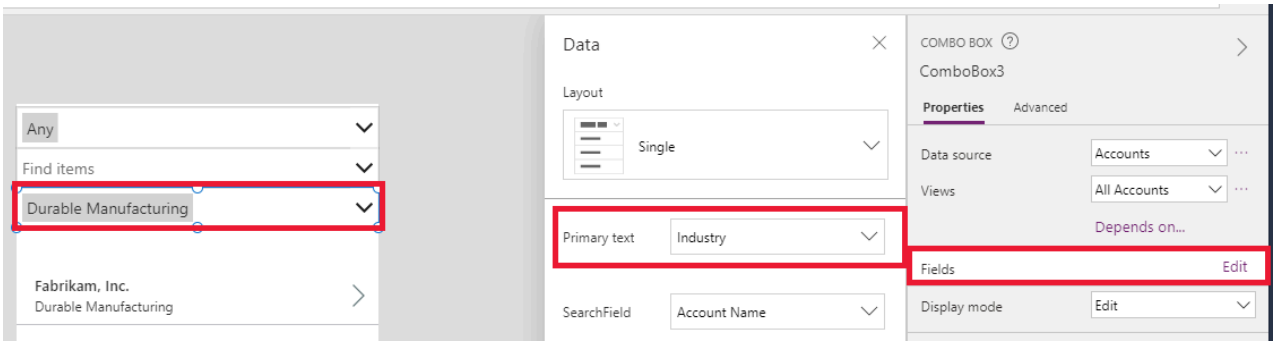
Formula	Description	Result
Filter(IceCream, OnOrder > 0)	Returns records where **OnOrder** is greater than zero.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Filter(IceCream, Quantity + OnOrder > 225)	Returns records where the sum of **Quantity** and **OnOrder** columns is greater than 225.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Strawberry"</td><td>300</td><td>0</td></tr></table>
Filter(IceCream, "chocolate" in Lower(Flavor))	Returns records where the word "chocolate" appears in the **Flavor** name, independent of uppercase or lowercase letters.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Filter(IceCream, Quantity < 10 && OnOrder < 20)	Returns records where the **Quantity** is less than 10 and **OnOrder** is less than 20. No records match these criteria, so an empty table is returned.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr></table>
Search(IceCream, "choc", Flavor)	Returns records where the string "choc" appears in the **Flavor** name, independent of uppercase or lowercase letters.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Search(IceCream, "", Flavor)	Because the search term is empty, all records are returned.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Strawberry"</td><td>300</td><td>0</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
LookUp(IceCream, Flavor = "Chocolate", Quantity)	Searches for a record with **Flavor** equal to "Chocolate", of which there's one. For the first record that's found, returns the **Quantity** of that record.	100
LookUp(IceCream, Quantity > 150, Quantity + OnOrder)	Searches for a record with **Quantity** greater than 150, of which there are multiple. For the first record that's found, which is "Vanilla" **Flavor**, returns the sum of **Quantity** and **OnOrder** columns.	250
LookUp(IceCream, Flavor = "Pistachio", OnOrder)	Searches for a record with **Flavor** equal to "Pistachio", of which there are none. Because none is found, **Lookup** returns *blank*.	*blank*
LookUp(IceCream, Flavor = "Vanilla")	Searches for a record with **Flavor** equal to "Vanilla", of which there's one. Since no reduction formula was supplied, the entire record is returned.	{ Flavor: "Vanilla", Quantity: 200, OnOrder: 75 }

Filtering with choice columns

The following example uses the **Account** table in Microsoft Dataverse as data source. This example shows how to **Filter** list of accounts based on selected Combo box control values:

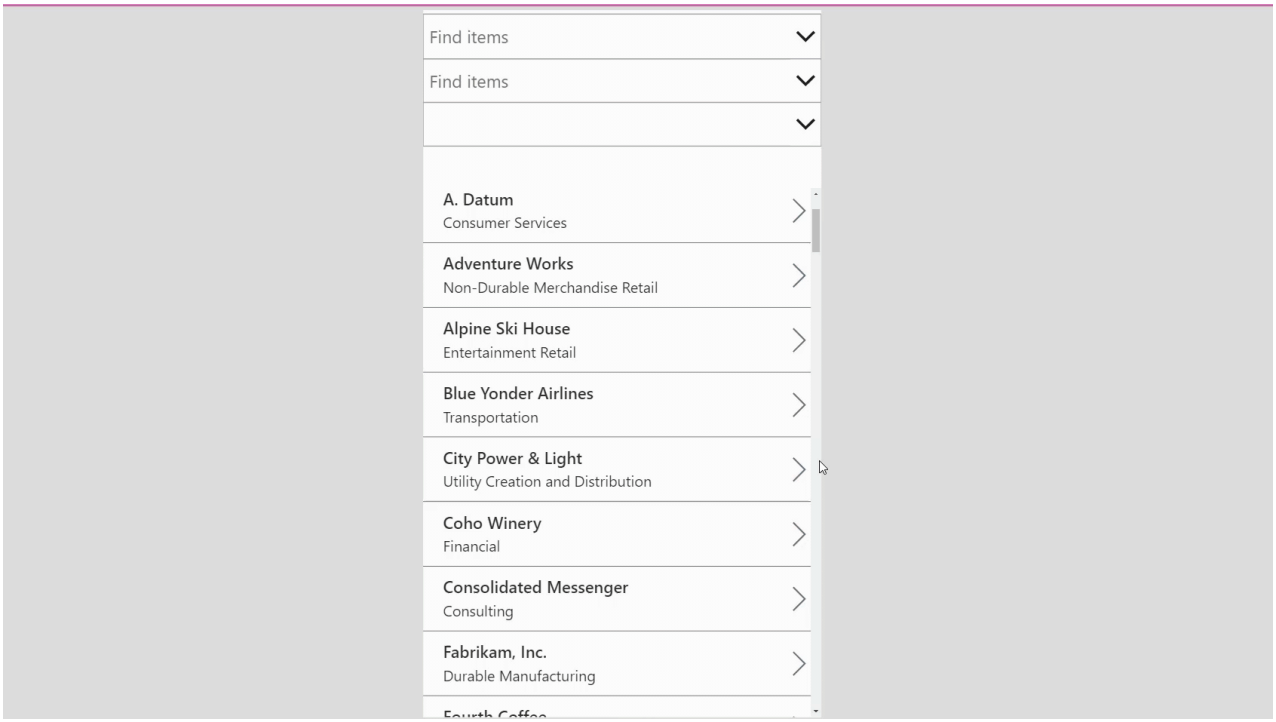
Step by step

1. Open a blank app.
2. Add a new screen by selecting the **New Screen** option.
3. On the **Insert** tab, select **Gallery** and then select **Vertical**.
4. On the **Properties** tab of the right-hand pane, open **Data Source** and then select **Accounts**.
5. (Optional) In the **Layout** list, select different options.
6. On the **Insert** tab, select **Input** and then select **Combo box**. Repeat the step to add two more combo box controls.
7. For each combo box control, on the **Properties** tab of the right-hand pane, open **Data Source** and then select **Accounts**. Select **Edit** next to **Fields** option and then select the **Primary text** and **SearchField** values. The **Primary text** should be the choices column you want to add to the combo box. Repeat the step for other two combo box controls.



8. Now select **Gallery** control and set the **Items** property to the following formula:

```
Filter(Accounts,
 'Industry' = ComboBox3.Selected.Industry Or
IsBlank(ComboBox3.Selected.Industry),
 'Relationship Type' = ComboBox2.Selected.'Relationship Type' Or
   IsBlank(ComboBox2.Selected.'Relationship Type'),
 'Preferred Method of Contact' = ComboBox1.Selected.'Preferred Method of
Contact' Or
   IsBlank(ComboBox1.Selected.'Preferred Method of Contact'))
```



Search user experience

The following examples use the **IceCream** data source:

In many apps, you can type one or more characters into a search box to filter a list of records in a large data set. As you type, the list shows only those records that match the search criteria.

The examples in the rest of this article show the results of searching a list, named **Customers**, that contain this data:

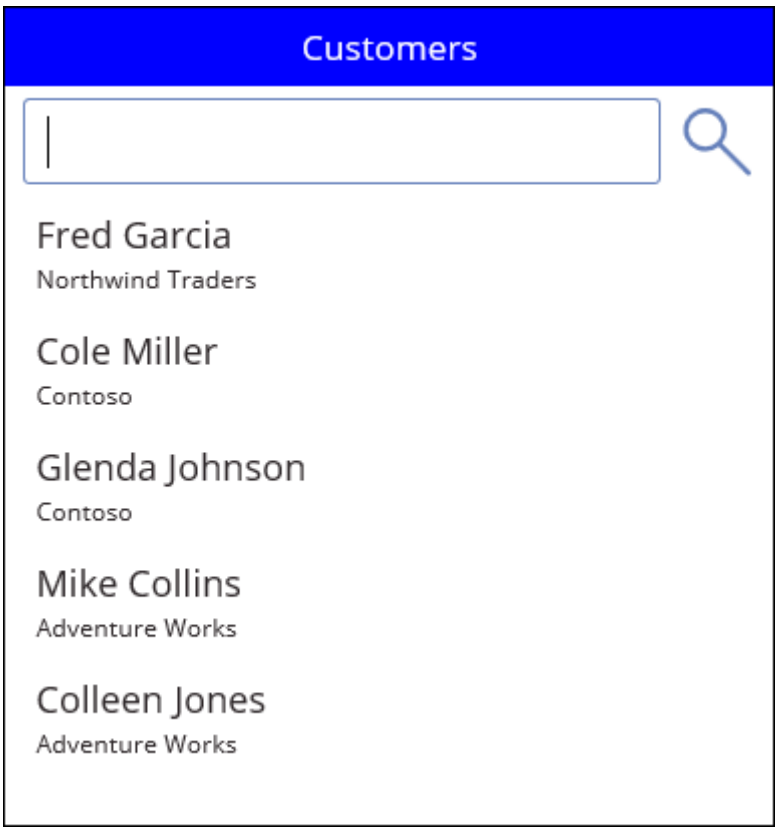
Name	Company
"Fred Garcia"	"Northwind Traders"
"Cole Miller"	"Contoso"
"Glenda Johnson"	"Contoso"
"Mike Collins"	"Adventure Works"
"Colleen Jones"	"Adventure Works"

To create this data source as a collection, create a **Button** control and set its **OnSelect** property to this formula:

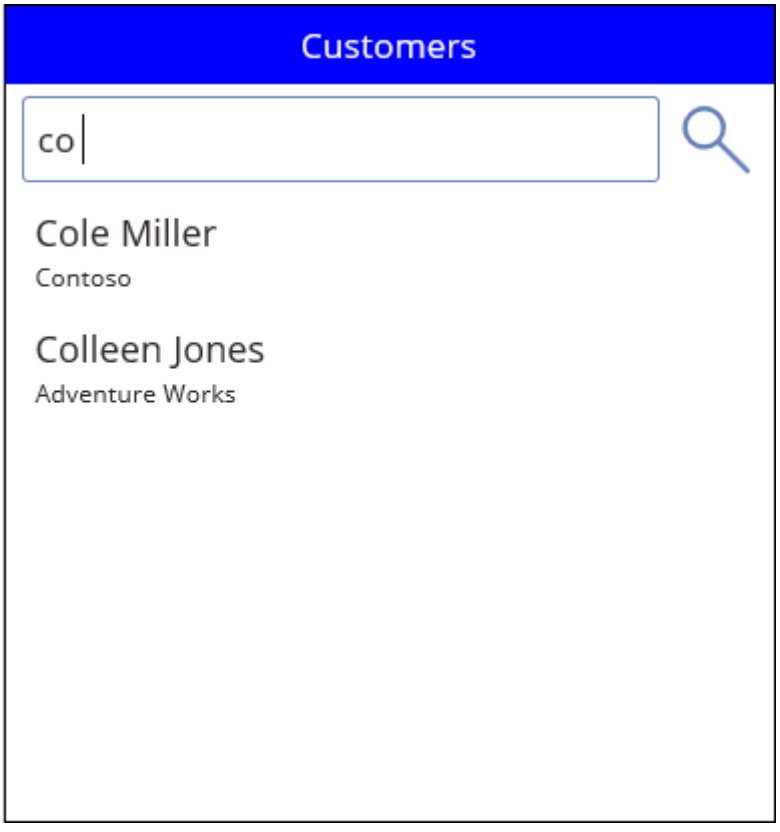
ClearCollect(Customers, Table({ Name: "Fred Garcia", Company: "Northwind Traders" }, { Name: "Cole Miller", Company: "Contoso" }, { Name: "Glenda Johnson", Company: "Contoso" }, { Name: "Mike Collins", Company: "Adventure Works" }, { Name: "Colleen Jones", Company: "Adventure Works" }))

As in this example, you can show a list of records in a **Gallery control** at the bottom of a screen. Near the top of the screen, you can add a **Text input** control, named **SearchInput**, so

that users can specify which records interest them.

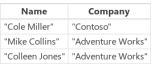


As the user types characters in **SearchInput**, the results in the gallery are automatically filtered. In this case, the gallery is configured to show records for which the name of the customer (not the name of the company) starts with the sequence of characters in **SearchInput**. If the user types **co** in the search box, the gallery shows these results:



To filter based on the **Name** column, set the **Items** property of the gallery control to one of these formulas:

Expand table

Formula	Description	Result
Filter(Customers, StartsWith(Name, SearchInput.Text))	Filters the **Customers** data source for records in which the search string appears at the start of the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones** and **Cole Miller**. The gallery doesn't show **Mike Collins** because the **Name** column for that record doesn't start with the search string.	
Filter(Customers, SearchInput.Text in Name)	Filters the **Customers** data source for records in which the search string appears anywhere in the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones, Cole Miller,** and **Mike Collins** because the search string appears somewhere in the **Name** column of all of those records.	
Search(Customers, SearchInput.Text, Name)	Similar to using the **in** operator, the **Search** function searches for a match anywhere within the **Name** column of each record. You must enclose the column name in double quotation marks.	

You can expand your search to include the **Company** column and the **Name** column:

Expand table

Formula	Description	Result
`Filter(Customers, StartsWith(Name, SearchInput.Text) || StartsWith(Company, SearchInput.Text) )`	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column starts with the search string (for example, **co**). The || operator is *true* if either **StartsWith** function is *true*.	
`Filter(Customers, SearchInput.Text in Name || SearchInput. Text in Company)`	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it.	
`Search(Customers, SearchInput.Text, Name, Company)`	Similar to using the **in** operator, the **Search** function searches the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it. The **Search** function is easier to read and write	

Formula	Description	Result
	than **Filter** if you want to specify multiple columns and multiple **in** operators.	

Last updated on 02/06/2025

Find function

Article • 04/20/2025

Applies to: Canvas apps Cards Copilot Studio Desktop flows Model-driven apps Power Platform CLI Dataverse functions Power Pages

Finds a string of text, if it exists, within another string.

Description

The **Find** function looks for a string within another string and is case sensitive. To ignore case, first use the Lower function on the arguments.

Find returns the starting position of the string that was found. Position 1 is the first character of the string. **Find** returns *blank* if the string in which you're searching doesn't contain the string for which you're searching.

Syntax

Find(*FindString*, *WithinString* [, *StartingPosition*])

- *FindString* - Required. The string to find.
- *WithinString* - Required. The string to search within.
- *StartingPosition* - Optional. The starting position to start searching. Position 1 is the first character.

Examples

Expand table

Formula	Description	Result
Find("World", "Hello World")	Returns the starting position of **"World"**.	7
Find("World", "Hello World, Hello World", 10)	Returns the starting position of the first occurrence of **"World"** after the 10th character.	20
Find("Mars", "Hello World")	Returns blank since the FindString is not in the WithinString	Blank()

First, FirstN, Index, Last, and LastN functions

04/20/2025

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the first, last, or a specific record, or a set of first or last records, from a table.

Description

The **First** function returns the first record of a table.

The **FirstN** function returns the first set of records of a table; the second argument specifies the number of records to return.

The **Last** function returns the last record of a table.

The **LastN** function returns the last set of records of a table; the second argument specifies the number of records to return.

The **Index** function returns a record of a table based on its ordered position in the table. Record numbering begins with 1 so `First( table )` returning the same record as `Index( table, 1 )`. **Index** returns an error if the requested record index is less than 1, greater than the number of records in the table, or the table is empty.

First, **Index**, and **Last** return a single record. **FirstN** and **LastN** return a table, even if you specify only a single record.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

For example, when used with a data source containing a large table with 1 million records, **Last** will be subject to the non-delegation limit and will not return the last record of the entire data source. Likewise, using **Index** to request a record in the middle of 1 million records will result in an error because the index is out of range based on the non-delegation limit.

Syntax

First(*Table*)
Last(*Table*)

- *Table* - Required. Table to operate on.

FirstN(*Table* [, *NumberOfRecords*])
LastN(*Table* [, *NumberOfRecords*])

- *Table* - Required. Table to operate on.
- *NumberOfRecords* - Optional. Number of records to return. If you don't specify this argument, the function returns one record.

Index(*Table*, *RecordIndex*)

- *Table* - Required. Table to operate on.
- *RecordIndex* - Required. The index of the record to return. Record numbering begins with 1.

Examples

For the following examples, we'll use the **IceCream** data source, which contains the data in this table:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200
"Strawberry"	300
"Mint Chocolate"	60
"Pistachio"	200

This table can be placed in a collection with this formula (put in the OnStart formula for a Button control and press the button):

Power Fx

```
Collect( IceCream, Table( { Flavor: "Chocolate", Quantity: 100 },
                          { Flavor: "Vanilla", Quantity: 200 },
                          { Flavor: "Strawberry", Quantity: 300 },
                          { Flavor: "Mint Chocolate", Quantity: 60 },
                          { Flavor: "Pistachio", Quantity: 200 } ) )
```

Expand table

Formula	Description	Result
First(IceCream)	Returns the first record of **IceCream**.	{ Flavor: "Chocolate", Quantity: 100 }
Last(IceCream)	Returns the last record of **IceCream**.	{ Flavor: "Pistachio", Quantity: 200 }
Index(IceCream, 3)	Returns the third record of **IceCream**.	{ Flavor: "Strawberry", Quantity: 300 }
FirstN(IceCream, 2)	Returns a table containing the first two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Chocolate"</td><td>100</td></tr><tr><td>"Vanilla"</td><td>200</td></tr></table>
LastN(IceCream, 2)	Returns a table containing the last two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Mint Chocolate"</td><td>60</td></tr><tr><td>"Pistachio"</td><td>200</td></tr></table>
Index(IceCream, 4).Quantity	Returns the fourth record of the table, and extracts the Quantity column.	60
Index(IceCream, 10)	Returns an error since the record requested is beyond the bounds of the table.	*Error*

First, FirstN, Index, Last, and LastN functions

04/20/2025

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the first, last, or a specific record, or a set of first or last records, from a table.

Description

The **First** function returns the first record of a table.

The **FirstN** function returns the first set of records of a table; the second argument specifies the number of records to return.

The **Last** function returns the last record of a table.

The **LastN** function returns the last set of records of a table; the second argument specifies the number of records to return.

The **Index** function returns a record of a table based on its ordered position in the table. Record numbering begins with 1 so `First( table )` returning the same record as `Index( table, 1 )`. **Index** returns an error if the requested record index is less than 1, greater than the number of records in the table, or the table is empty.

First, **Index**, and **Last** return a single record. **FirstN** and **LastN** return a table, even if you specify only a single record.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

For example, when used with a data source containing a large table with 1 million records, **Last** will be subject to the non-delegation limit and will not return the last record of the entire data source. Likewise, using **Index** to request a record in the middle of 1 million records will result in an error because the index is out of range based on the non-delegation limit.

Syntax

First(*Table*)
Last(*Table*)

- *Table* - Required. Table to operate on.

FirstN(*Table* [, *NumberOfRecords*])
LastN(*Table* [, *NumberOfRecords*])

- *Table* - Required. Table to operate on.
- *NumberOfRecords* - Optional. Number of records to return. If you don't specify this argument, the function returns one record.

Index(*Table*, *RecordIndex*)

- *Table* - Required. Table to operate on.
- *RecordIndex* - Required. The index of the record to return. Record numbering begins with 1.

Examples

For the following examples, we'll use the **IceCream** data source, which contains the data in this table:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200
"Strawberry"	300
"Mint Chocolate"	60
"Pistachio"	200

This table can be placed in a collection with this formula (put in the OnStart formula for a Button control and press the button):

Power Fx

```
Collect( IceCream, Table( { Flavor: "Chocolate", Quantity: 100 },
                          { Flavor: "Vanilla", Quantity: 200 },
                          { Flavor: "Strawberry", Quantity: 300 },
                          { Flavor: "Mint Chocolate", Quantity: 60 },
                          { Flavor: "Pistachio", Quantity: 200 } ) )
```

Expand table

Formula	Description	Result
First(IceCream)	Returns the first record of **IceCream**.	{ Flavor: "Chocolate", Quantity: 100 }
Last(IceCream)	Returns the last record of **IceCream**.	{ Flavor: "Pistachio", Quantity: 200 }
Index(IceCream, 3)	Returns the third record of **IceCream**.	{ Flavor: "Strawberry", Quantity: 300 }
FirstN(IceCream, 2)	Returns a table containing the first two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Chocolate"</td><td>100</td></tr><tr><td>"Vanilla"</td><td>200</td></tr></table>
LastN(IceCream, 2)	Returns a table containing the last two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Mint Chocolate"</td><td>60</td></tr><tr><td>"Pistachio"</td><td>200</td></tr></table>
Index(IceCream, 4).Quantity	Returns the fourth record of the table, and extracts the Quantity column.	60
Index(IceCream, 10)	Returns an error since the record requested is beyond the bounds of the table.	*Error*

ForAll function

10/10/2025

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Calculates values and performs actions for all the records in a table.

Description

The **ForAll** function evaluates a formula for all the records in a table. The formula can calculate a value and/or perform actions, such as modifying data or working with a connection. Use the **With** function to evaluate the formula for a single record.

Use the **Sequence** function with the **ForAll** function to iterate based on a count.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

Return value

The result of each formula evaluation is returned in a table, in the same order as the input table.

If the result of the formula is a single value, the resulting table is a single-column table. If the result of the formula is a record, the resulting table contains records with the same columns as the result record.

If the result of the formula is a *blank* value, then there's no record in the result table for that input record. In this case, there are fewer records in the result table than the source table.

Taking action

The formula can include functions that take action, such as modifying the records of a data source with the **Patch** and **Collect** functions. The formula can also call methods on connections. Multiple actions can be performed per record by using the **;** operator. You can't modify the table that is the subject of the **ForAll** function.

When writing your formula, keep in mind that records can be processed in any order and, when possible, in parallel. The first record of the table may be processed after the last record.

Take care to avoid ordering dependencies. For this reason, you can't use the **UpdateContext**, **Clear**, and **ClearCollect** functions within a **ForAll** function because they could easily be used to hold variables that would be susceptible to this effect. You can use **Collect**, but the order in which records are added is undefined.

Several functions that modify data sources, including **Collect**, **Remove**, and **Update**, return the changed data source as their return value. These return values can be large and consume significant resources if returned for every record of the **ForAll** table. You may also find that these return values aren't what you expect because **ForAll** can operate in parallel and may separate the side effects of these functions from obtaining their result. If the return value from **ForAll** isn't used, which is often the case with data modification functions, then the return value won't be created and there are no resource or order concerns. But if you're using the result of a **ForAll** and one of the functions that returns a data source, think carefully about how you structure the result and try it out first on small data sets.

Alternatives

Many functions in Power Apps can process more than one value at a time by using a single-column table. For example, the **Len** function can process a table of text values, returning a table of lengths, in the same manner, that **ForAll** could. This can eliminate the need to use **ForAll** in many cases, can be more efficient, and is easier to read.

Another consideration is that **ForAll** isn't delegable while other functions may be, such as **Filter**.

Delegation

When used with a data source, this function can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

ForAll(*Table*, *Formula*)

- *Table* - Required. Table to be acted upon.
- *Formula* - Required. The formula to evaluate for all records of the *Table*.

Examples

Calculations

The following examples use the **Squares** data source:

Value
1
4
9

To create this data source as a collection, set the **OnSelect** property of a **Button** control to this formula, open Preview mode, and then select the button:

```
ClearCollect( Squares, [ "1", "4", "9" ] )
```

Expand table

Formula	Description	Result
ForAll(Squares, Sqrt(Value)) **Sqrt(Squares)**	For all the records of the input table, calculates the square root of the **Value** column. The **Sqrt** function can also be used with a single-column table, making it possible to perform this example without using **ForAll**.	**Value** 1 2 3
ForAll(Squares, Power(Value, 3))	For all the records of the input table, raises the **Value** column to the third power. The **Power** function doesn't support single-column tables. Therefore, **ForAll** must be used in this case.	**Value** 1 64 729

Using a connection

The following examples use the **Expressions** data source:

Value
"Hello"
"Good morning"
"Thank you"
"Goodbye"

To create this data source as a collection, set the **OnSelect** property of a **Button** control to this formula, open Preview mode, and then select the button:

```
ClearCollect( Expressions, [ "Hello", "Good morning", "Thank you", "Goodbye" ] )
```

This example also uses a Microsoft Translator connection. To add this connection to your app, see the article about how to manage connections.

Expand table

Formula	Description	Result
ForAll(Expressions, MicrosoftTranslator.Translate(Value, "es"))	For all the records in the Expressions table, translate the contents of the **Value** column into Spanish (abbreviated "es").	Value "Hola" "Buenos días" "Gracias" "Adiós"
ForAll(Expressions, MicrosoftTranslator.Translate(Value, "fr"))	For all the records in the Expressions table, translate the contents of the **Value** column into French (abbreviated "fr").	Value "Salut" "Bonjour" "Merci" "Au revoir"

Copying a table

Sometimes you need to filter, shape, sort, and manipulate data. Power Apps provides many functions for doing this, such as **Filter**, **AddColumns**, and **Sort**. Power Apps treats each table as a value, allowing it to flow through formulas and be consumed easily.

And sometimes you want to make a copy of this result for later use, or you want to move information from one data source to another. Power Apps provides the **Collect** function to copy data.

But before you make that copy, think carefully if it's needed. Many situations can be addressed by filtering and shaping the underlying data source on-demand with a formula. Some of the downsides to making a copy include:

- Two copies of the same information mean that one of them can fall out of sync.
- Making a copy can consume much of the computer memory, network bandwidth, and/or time.
- For most data sources, copying can't be delegated, limiting how much data can be moved.

The following examples use the **Products** data source:

Product	Quantity Requested	Quantity Available
"Widget"	6	3
"Gadget"	10	20
"Gizmo"	4	11
"Apparatus"	7	6

To create this data source as a collection, set the **OnSelect** property of a **Button** control to this formula, open Preview mode, and then select the button:

Power Fx

```
ClearCollect( Products,
    Table(
        { Product: "Widget",    'Quantity Requested': 6,  'Quantity Available': 3
},
        { Product: "Gadget",    'Quantity Requested': 10, 'Quantity Available': 20
},
        { Product: "Gizmo",     'Quantity Requested': 4,  'Quantity Available': 11
},
        { Product: "Apparatus", 'Quantity Requested': 7,  'Quantity Available': 6
}
    )
)
```

Our goal is to work with a derivative table that includes only the items where more has been requested than is available, and for which we need to place an order:

Product	Quantity To Order
"Widget"	3
"Apparatus"	1

We can perform this task in a couple of different ways, all of which produce the same result, with various pros and cons.

Table shaping on demand

Don't make that copy! We can use the following formula anywhere we need:

Power Fx

```
// Table shaping on demand, no need for a copy of the result
ShowColumns(
    AddColumns(
        Filter( Products, 'Quantity Requested' > 'Quantity Available' ),
        "Quantity To Order", 'Quantity Requested' - 'Quantity Available'
    ),
```

```
    "Product",
    "Quantity To Order"
)
```

A record scope is created by the **Filter** and **AddColumns** functions to perform the comparison and subtraction operations, respectively, with the '**Quantity Requested**' and '**Quantity Available**' fields of each record.

In this example, the **Filter** function can be delegated. This is important, as it can find all the products that meet the criteria, even if that is only a few records out of a table of millions. At this time, **ShowColumns** and **AddColumns** cannot be delegated, so the actual number of products that need to be ordered is limited. If you know the size of this result is always be relatively small, this approach is fine.

And because we didn't make a copy, there's no extra copy of the information to manage or fall out of date.

ForAll on demand

Another approach is to use the **ForAll** function to replace the table-shaping functions:

Power Fx

```
ForAll( Products,
    If( 'Quantity Requested' > 'Quantity Available',
        {
            Product: Product,
            'Quantity To Order': 'Quantity Requested' - 'Quantity Available'
        }
    )
)
```

This formula may be simpler for some people to read and write.

No part of the **ForAll** is delegable. Only the first portion of the **Products** table is evaluated, which could be a problem if this table is large. Because **Filter** could be delegated in the previous example, it could work better with large data sets.

Collect the result

In some situations, a copy of data may be required. You may need to move information from one data source to another. In this example, orders are placed through a **NewOrder** table on a vendor's system. For high-speed user interactions, you may want to cache a local copy of a table so that there's no server latency.

We use the same table shaping as the previous two examples, but we capture the result into a collection:

Power Fx

```
ClearCollect( NewOrder,
    ShowColumns(
        AddColumns(
            Filter( Products, 'Quantity Requested' > 'Quantity Available' ),
            "Quantity To Order", 'Quantity Requested' - 'Quantity Available'
        ),
        "Product",
        "Quantity To Order"
    )
)
```

Power Fx

```
ClearCollect( NewOrder,
    ForAll( Products,
        If( 'Quantity Requested' > 'Quantity Available',
            {
                Product: Product,
                'Quantity To Order': 'Quantity Requested' - 'Quantity Available'
            }
        )
    )
)
```

ClearCollect and **Collect** can't be delegated. As a result, the amount of data that can be moved in this manner is limited.

Collect within ForAll

Finally, we can perform the **Collect** directly within the **ForAll**:

Power Fx

```
Clear( NewOrder );
ForAll( Products,
    If( 'Quantity Requested' > 'Quantity Available',
        Collect( NewOrder,
            {
                Product: Product,
                'Quantity To Order': 'Quantity Requested' - 'Quantity Available'
            }
        )
```

```
    )
)
```

Again, the **ForAll** function can't be delegated at this time. If our **Products** table is large, **ForAll** looks at the first set of records only and we may miss some products that need to be ordered. But for tables that we know are remain small, this approach is fine.

Note that we aren't capturing the result of the **ForAll**. The **Collect** function calls made from within it returns the **NewOrder** data source for all the records, which could add up to numerous data if we were capturing it.

Map table in a component

See Map tables.

GroupBy and Ungroup functions

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Groups and ungroups records of a table.

Description

The **GroupBy** function returns a table with records grouped together based on the values in one or more columns. Records in the same group are placed into a single record, with a column added that holds a nested table of the remaining columns.

The **Ungroup** function reverses the **GroupBy** process. This function returns a table, breaking into separate records any records that were grouped together.

You can group records by using **GroupBy**, modify the table that it returns, and then ungroup records in the modified table by using **Ungroup**. For example, you can remove a group of records by following this approach:

- Use the **GroupBy** function.
- Use the **Filter** function to remove the entire group of records.
- Use the **Ungroup** function.

You can also aggregate results based on a grouping:

- Use the **GroupBy** function.
- Use the **AddColumns** function with **Sum**, **Average**, and other aggregate functions to add a new column, which is an aggregate of the group tables.
- Use the **DropColumns** function to drop the group table.

Ungroup tries to preserve the original order of the records that were fed to **GroupBy**. This isn't always possible (for example, if the original table contains *blank* records).

A table is a value in Power Apps, just like a string or a number. You can specify a table as an argument for a function, and a function can return a table. **GroupBy** and **Ungroup** don't modify a table; instead they take a table as an argument and return a different table. See working with tables for more details.

> **Note**

In Power Apps prior to version 3.24042, column names were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

Syntax

GroupBy(*Table*, *ColumnName1* [, *ColumnName2*, ...], *GroupColumnName*)

- *Table* - Required. Table to be grouped.
- *ColumnName(s)* - Required. The column names in *Table* by which to group records. These columns become columns in the resulting table.
- *GroupColumnName* - Required. The column name for the storage of record data not in the *ColumnName(s)*.

Ungroup(*Table*, *GroupColumnName*)

- *Table* - Required. Table to be ungrouped.
- *GroupColumnName* - Required. The column that contains the record data setup with the **GroupBy** function.

Examples

Create a collection

1. Add a button, and set its **Text** property so that the button shows **Original**.
2. Set the **OnSelect** property of the **Original** button to this formula:

Power Apps

```
ClearCollect( CityPopulations,
    { City: "London",    Country: "United Kingdom", Population: 8615000},
    { City: "Berlin",    Country: "Germany",        Population: 3562000},
    { City: "Madrid",    Country: "Spain",          Population: 3165000},
    { City: "Rome",      Country: "Italy",          Population: 2874000},
    { City: "Paris",     Country: "France",         Population: 2273000},
    { City: "Hamburg",   Country: "Germany",        Population: 1760000},
    { City: "Barcelona", Country: "Spain",          Population: 1602000},
    { City: "Munich",    Country: "Germany",        Population: 1494000},
```

```
    { City: "Milan",     Country: "Italy",          Population: 1344000}
)
```

3. While holding down the Alt key, select the **Original** button.

   You just created a collection, named **CityPopulations**, that contains this data:

City	Country	Population
"London"	"United Kingdom"	8,615,000
"Berlin"	"Germany"	3,562,000
"Madrid"	"Spain"	3,165,000
"Rome"	"Italy"	2,874,000
"Paris"	"France"	2,273,000
"Hamburg"	"Germany"	1,760,000
"Barcelona"	"Spain"	1,602,000
"Munich"	"Germany"	1,494,000
"Milan"	"Italy"	1,344,000

4. To display this collection, select **Collections** on the **File** menu and then select the **CityPopulations** collection. The first five records in the collection appear:



Group records

1. Add another button, and set its **Text** property to "**Group**".

2. Set the **OnSelect** property of this button to this formula:

ClearCollect(CitiesByCountry, GroupBy(CityPopulations, Country, Cities))

3. While holding down the Alt key, select the **Group** button.

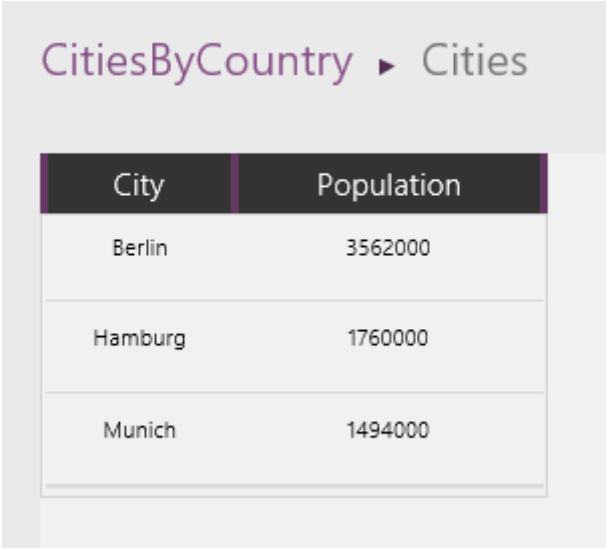
   You just created a collection, named **CitiesByCountry**, in which the records of the previous collection are grouped by the **Country** column.

Cities	Country
City / **Population** "London" / 8,615,000	"United Kingdom"
City / **Population** "Berlin" / 3,562,000 "Hamburg" / 1,760,000 "Munich" / 1,494,000	"Germany"
City / **Population** "Madrid" / 3,165,000 "Barcelona" / 1,602,000	"Spain"
City / **Population** "Rome" / 2,874,000 "Milan" / 1,344,000	"Italy"
City / **Population** "Paris" / 2,273,000	"France"

4. To display the first five records in this collection, select **Collections** on the **File** menu.



5. To display the populations of cities in a country/region, select the table icon in the **Cities** column for that country/region (for example, Germany):



Filter and ungroup records

1. Add another button, and set its **Text** property so that the button shows **"Filter"**.

2. Set the **OnSelect** property of this button to this formula:

   ClearCollect(CitiesByCountryFiltered, Filter(CitiesByCountry, "e" in Country))

3. While holding down the Alt key, select the button that you added.

   You just created a third collection, named **CitiesByCountryFiltered**, that includes only those countries that have an "e" in their names (that is, not Spain or Italy).

Cities	Country
City / Population "London" / 8,615,000	"United Kingdom"
City / Population "Berlin" / 3,562,000 "Hamburg" / 1,760,000 "Munich" / 1,494,000	"Germany"
City / Population "Paris" / 2,273,000	"France"

4. Add one more button, and set its **Text** property so that the button shows **"Ungroup"**.

5. Set the **OnSelect** property of this button to this formula:

   ClearCollect(CityPopulationsUngrouped, Ungroup(CitiesByCountryFiltered, Cities))

   Which results in:

City	Country	Population
"London"	"United Kingdom"	8,615,000
"Berlin"	"Germany"	3,562,000
"Paris"	"France"	2,273,000
"Hamburg"	"Germany"	1,760,000
"Munich"	"Germany"	1,494,000

Aggregate results

Another operation we can perform on a grouped table is to compile the results. In this example, we'll sum the population of the major cities in each country/region.

1. Add another button, and set its **Text** property so that the button shows **"Sum"**.

2. Set the **OnSelect** property of the **"Sum"** button to this formula:

   ClearCollect(CityPopulationsSum, AddColumns(CitiesByCountry, 'Sum of City Populations', Sum(Cities, Population)))

   Which results in:

Cities	Country	Sum of City Populations
City: "London", Population: 8,615,000	"United Kingdom"	8,615,000
City: "Berlin", Population: 3,562,000 City: "Hamburg", Population: 1,760,000 City: "Munich", Population: 1,494,000	"Germany"	6,816,000
City: "Madrid", Population: 3,165,000 City: "Barcelona", Population: 1,602,000	"Spain"	4,767,000
City: "Rome", Population: 2,874,000 City: "Milan", Population: 1,344,000	"Italy"	4,218,000
City: "Paris", Population: 2,273,000	"France"	2,273,000

AddColumns starts with the base **CitiesByCountry** collection and adds a new column **Sum of City Populations**. This column's values are calculated row-by-row, based on the formula **Sum(Cities, Population)**. **AddColumns** provides the value of the **Cities** column (a table) for each row, and Sum adds up the **Population** for each row of this sub table.

Now that we have the sum that we want, we can use DropColumns to remove the sub tables.

3. Add another button, and set its Text property so that the button shows **"SumOnly"**.

4. Set the OnSelect property of the **"SumOnly"** button to this formula:

   ClearCollect(CityPopulationsSumOnly, DropColumns(CityPopulationsSum, Cities))

   Which results in:

Country	Sum of City Populations
"United Kingdom"	8,615,000
"Germany"	6,816,000
"Spain"	4,767,000
"Italy"	4,218,000
"France"	2,273,000

Note that we didn't need to ungroup this table.

Feedback

Was this page helpful? Yes No

Provide product feedback

GUID function

06/19/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Convert a GUID (globally unique identifier) string to a GUID value, or create a new GUID value.

Description

Use the **GUID** function to convert a string with the hexadecimal representation of a GUID into a GUID value you can pass to a database. Database systems like Microsoft Dataverse and SQL Server use GUID values as keys.

The string can have uppercase or lowercase letters, but it must be 32 hexadecimal digits in one of these formats:

- **"123e4567-e89b-12d3-a456-426655440000"** (hyphens in standard locations)
- **"123e4567e89b12d3a456426655440000"** (no hyphens)

If you don't specify an argument, the function creates a new GUID.

To convert a GUID value to a string, use it in a string context. The GUID value converts to a hexadecimal string with hyphens and lowercase letters.

When generating a new GUID, the function uses pseudo-random numbers to create a version 4 IETF RFC 4122 GUID. When converting a string to a GUID, the function supports any GUID version by accepting any string of 32 hexadecimal digits.

Volatile functions

GUID is a volatile function when used without an argument. Each time the function is evaluated, it returns a different value.

When you use a volatile function in a data-flow formula, it returns a different value only if the formula is reevaluated. If nothing else changes in the formula, it keeps the same value while your app is open.

For example, if you set the **Text** property of a label control to **GUID()**, it doesn't change while your app is active. You get a different value only when you close and reopen the app.

The function is reevaluated if it's part of a formula where something else changes. For example, if you set the **Text** property of a **Label** control to this formula, a GUID is generated each time the user changes the value of the **Text input** control:

TextInput1.Text & " " & GUID()

When you use **GUID** in a behavior formula, it's evaluated each time the formula is evaluated. For more information, see the examples later in this topic.

Syntax

GUID([*GUIDString*])

- *GUIDString* – Optional. A text string that has the hexadecimal representation of a GUID. If you don't supply a string, the function creates a new GUID.

GUID(*Dynamic*)

- *Dynamic* – Required. **Dynamic** value that represents a GUID. Acceptable values depend on the untyped provider. For **JSON**, the dynamic value is expected to be a GUID represented as a JSON string.

Examples

Basic usage

To return a GUID value based on the hexadecimal string representation:

```
Power Fx
GUID( "0f8fad5b-d9cb-469f-a165-70867728950e" )
```

You can also provide the GUID string without hyphens. This formula returns the same GUID value:

```
Power Fx
GUID( "0f8fad5bd9cb469fa16570867728950e" )
```

Used in context, to set the **Status** field of a new database record to a well-established value:

```
Power Fx
```

```
Patch( Products, Default( Products ), { Status: GUID( "F9168C5E-CEB2-4faa-B6BF-329BF39FA1E4" ) } )
```

You probably don't want to show GUIDs to users, but GUIDs can help you debug your app. To show the value of the **Status** field in the record you created in the previous example, set the **Text** property of a **Label** control to this formula:

Power Fx

```
First( Products ).Status
```

The **Label** control shows **f9168c5e-ceb2-4faa-b6bf-329bf39fa1e4**.

Create a table of GUIDs

1. Set the **OnSelect** property of a **Button** control to this formula:

   Power Fx

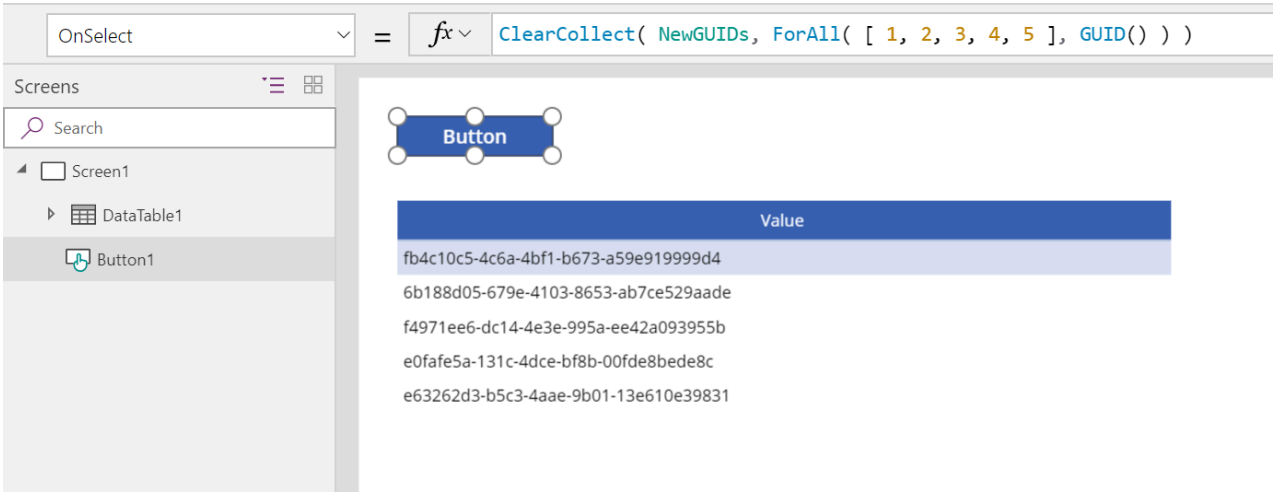
   ```
   ClearCollect( NewGUIDs, ForAll( Sequence(5), GUID() ) )
   ```

   This formula creates a single-column table that's used to iterate five times, resulting in five GUIDs.

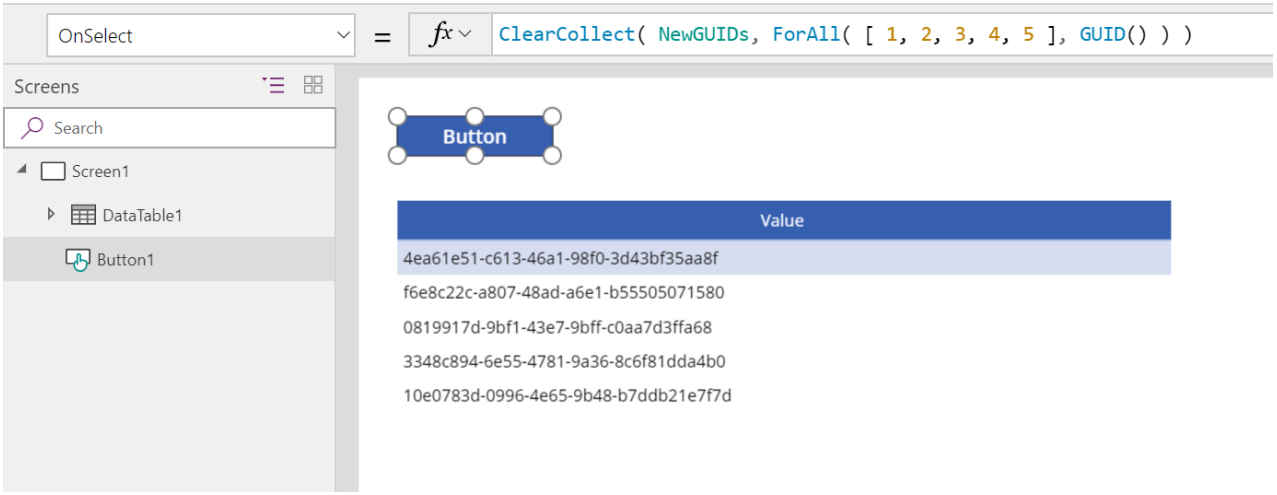
2. Add a **Data table** control, set its **Items** property to **NewGUIDs**, and show the **Value** field.

3. Hold down the Alt key and select the button.

   The data table shows a list of GUIDs:



4. Select the button again to show a new list of GUIDs:



To generate a single GUID instead of a table, use this formula:

Power Fx

```
Set( NewGUID, GUID() )
```

HashTags function

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Model-driven apps ✅ Power Pages

Extracts the hashtags (#strings) from a string of text.

Description

The **HashTags** function scans a string for hashtags. Hashtags start with a pound character (#), which is followed by any combination of:

- uppercase and lowercase letters
- numerals
- underscores
- currency symbols (such as $)

HashTags returns a one-column table that contains the hashtags in the string. If the string contains no hashtags, the function returns a one-column table that's empty.

Syntax

HashTags(*String*)

- *String* - Required. String to scan for hashtags.

Examples

Step by step

1. Add a **Text input** control, name it **Tweet**, and type this sentence into it:

   This #app is #AMAZING and can #coUnt123 or #123abc but not #1-23 or #$* (#@")

2. Add a vertical custom gallery, and set its **Items** property to this function:

   HashTags(Tweet.Text)

3. Add a **Label** control to the gallery template.

The gallery shows these hashtags:

- #app
- #AMAZING
- #coUnt123
- #123abc
- #1

Feedback

Was this page helpful? Yes No

Provide product feedback

Day, Month, Year, Hour, Minute, Second, and Weekday functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns individual components of a Date/Time value.

Description

The **Day** function returns the day component of a Date/Time value, ranging from 1 to 31.

The **Month** function returns the month component of a Date/Time value, ranging from 1 to 12.

The **Year** function returns the year component of a Date/Time value, starting with 1900.

The **Hour** function returns the hour component of a Date/Time value, ranging from 0 (12:00 AM) to 23 (11:00 PM).

The **Minute** function returns the minute component of a Date/Time value, ranging from 0 to 59.

The **Second** function returns the second component of a Date/Time value, ranging from 0 to 59.

The **Weekday** function returns the weekday of a Date/Time value. By default, the result ranges from 1 (Sunday) to 7 (Saturday). You can specify a different range with a Microsoft Excel Weekday function code or a StartOfWeek enumeration value:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	StartOfWeek.Sunday	Numbers 1 (Sunday) through 7 (Saturday). Default.
2, 11	StartOfWeek.Monday	Numbers 1 (Monday) through 7 (Sunday).
3	StartOfWeek.MondayZero	Numbers 0 (Monday) through 6 (Sunday).
12	StartOfWeek.Tuesday	Numbers 1 (Tuesday) through 7 (Monday).

Excel code	StartOfWeek enumeration	Description
13	**StartOfWeek.Wednesday**	Numbers 1 (Wednesday) through 7 (Tuesday).
14	**StartOfWeek.Thursday**	Numbers 1 (Thursday) through 7 (Wednesday).
15	**StartOfWeek.Friday**	Numbers 1 (Friday) through 7 (Thursday).
16	**StartOfWeek.Saturday**	Numbers 1 (Saturday) through 7 (Friday).

All functions return a number.

See working with dates and times for more information.

Syntax

Day(*DateTime*)
Month(*DateTime*)
Year(*DateTime*)
Hour(*DateTime*)
Minute(*DateTime*)
Second(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on.

Weekday(*DateTime* [, *WeekdayFirst*])

- *DateTime* - Required. Date/Time value to operate on.
- *WeekdayFirst* - Optional. Excel code specifying which day starts the week. If not supplied, 1 (Sunday first) is used.

Examples

For the following example, the current time is **3:59:37 PM** on **Thursday, April 9, 2015**.

Expand table

Formula	Description	Result
Year(Now())	Returns the year component of the current time and date.	2015
Month(Now())	Returns the month component of the current time and date.	4

Formula	Description	Result
Day(Now())	Returns the day component of the current time and date.	9
Hour(Now())	Returns the hour component of the current time and date.	15
Minute(Now())	Returns the minute component of the current time and date.	59
Second(Now())	Returns the second component of the current time and date.	37
Weekday(Now())	Returns the weekday component of the current time and date, using the default start of the week as Sunday.	5
Weekday(Now(), 14)	Returns the weekday component of the current time and date, using an Excel code to specify the start of the week as Thursday.	1
Weekday(Now(), StartOfWeek.Wednesday)	Returns the weekday component of the current time and date, using a **StartOfWeek** enumeration to specify the start of the week as Wednesday.	2

Feedback

Was this page helpful? Yes No

Provide product feedback

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

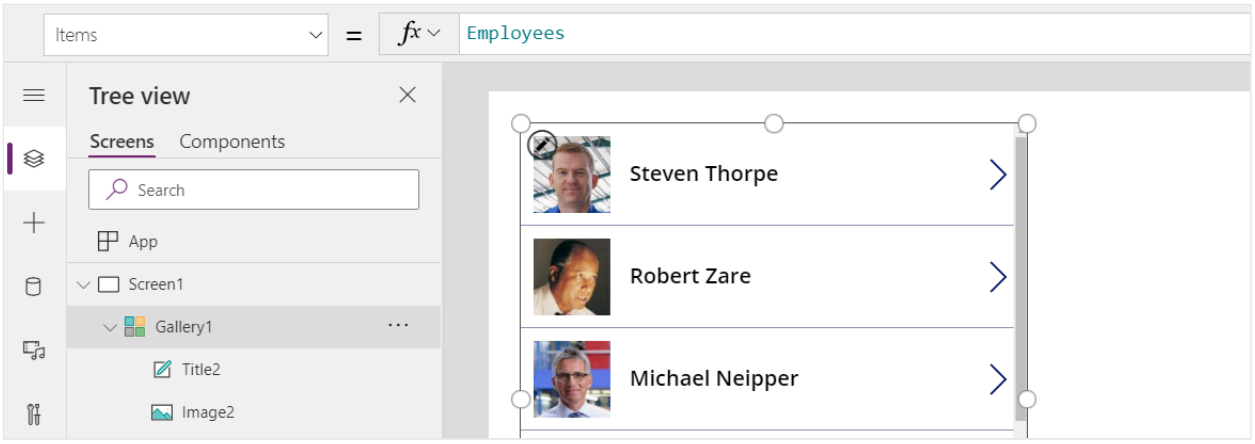
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

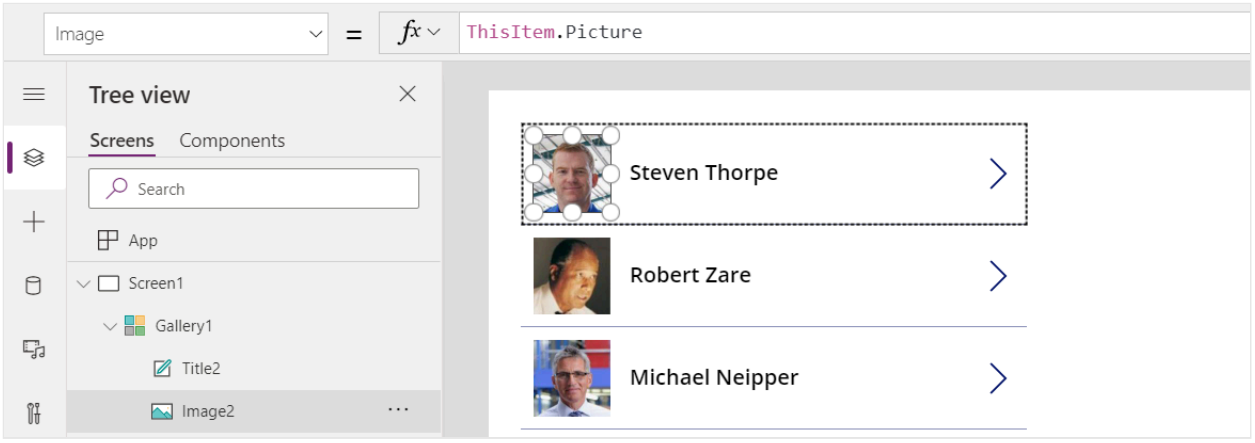
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

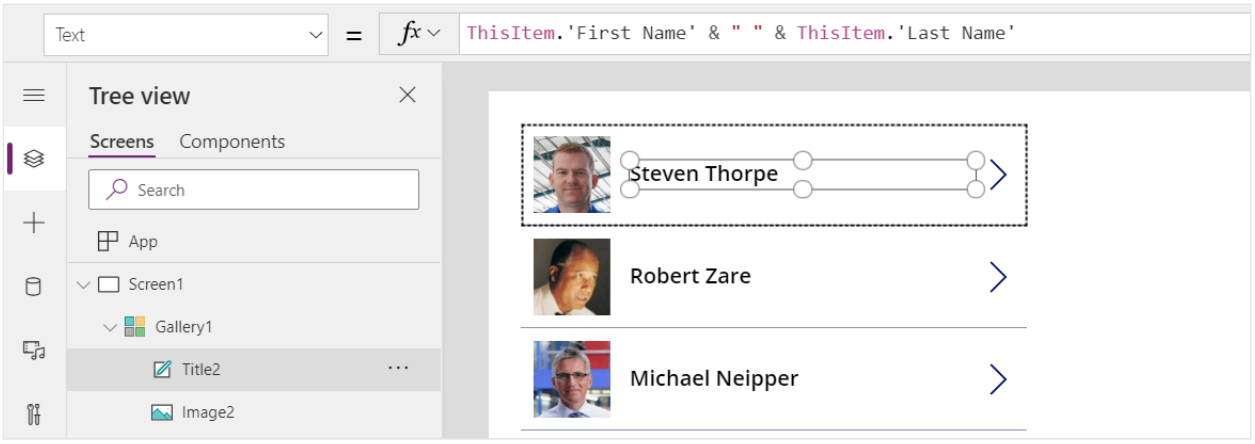
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

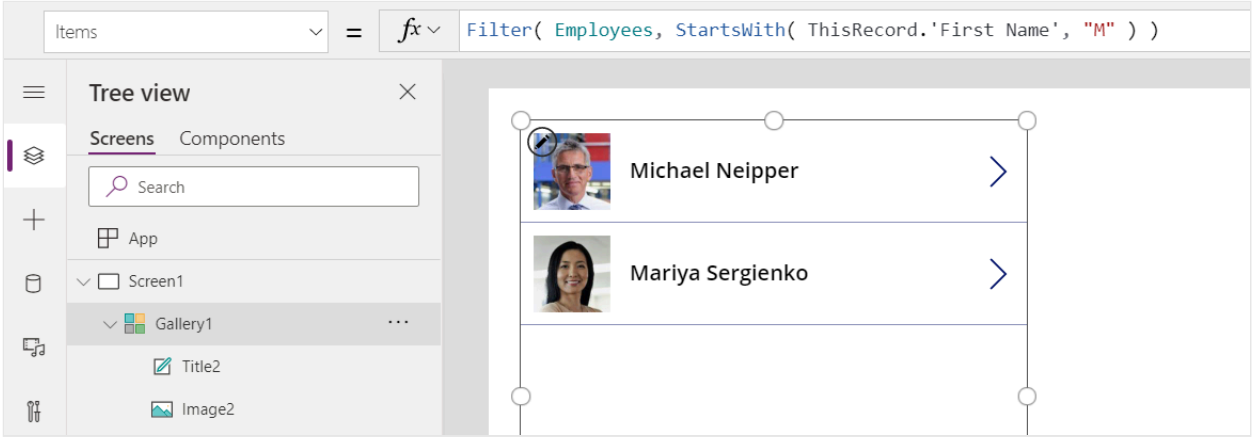


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

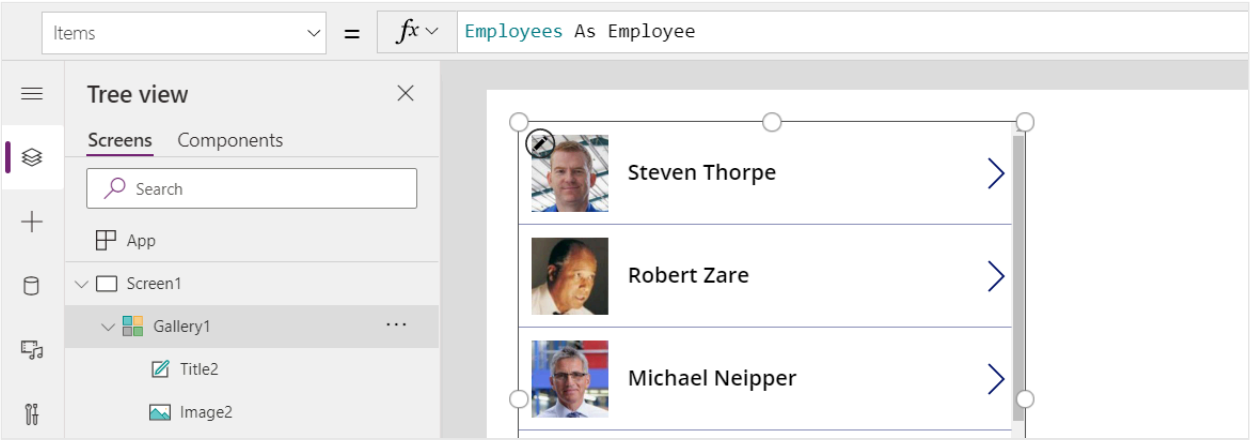
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

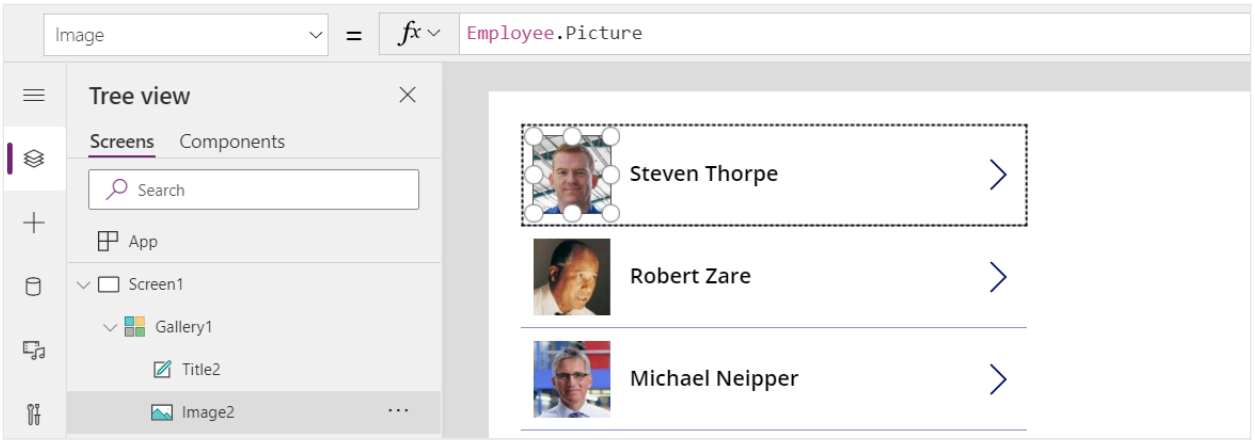
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

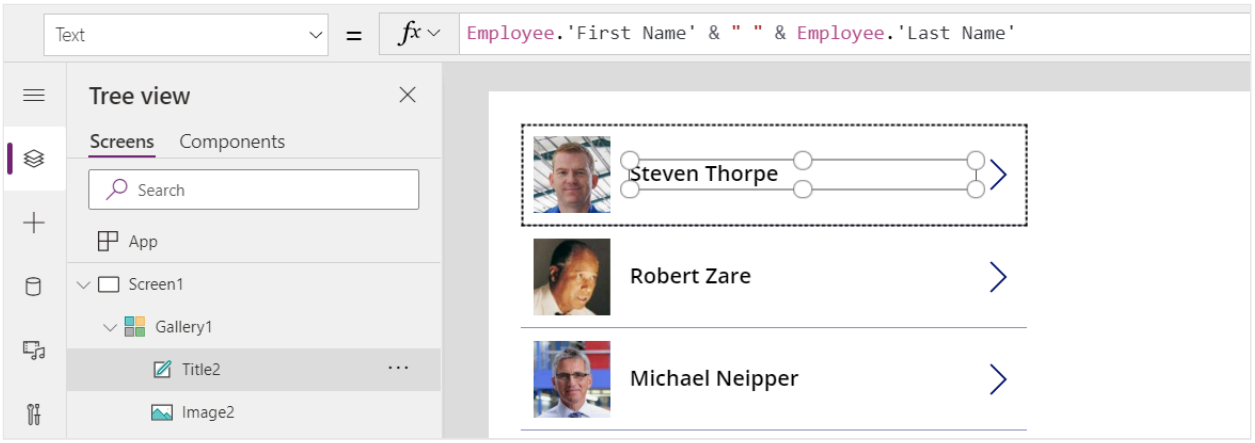
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

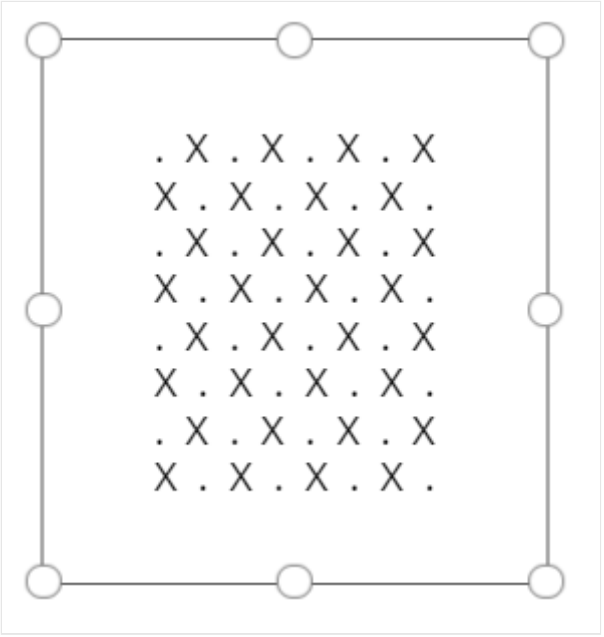
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



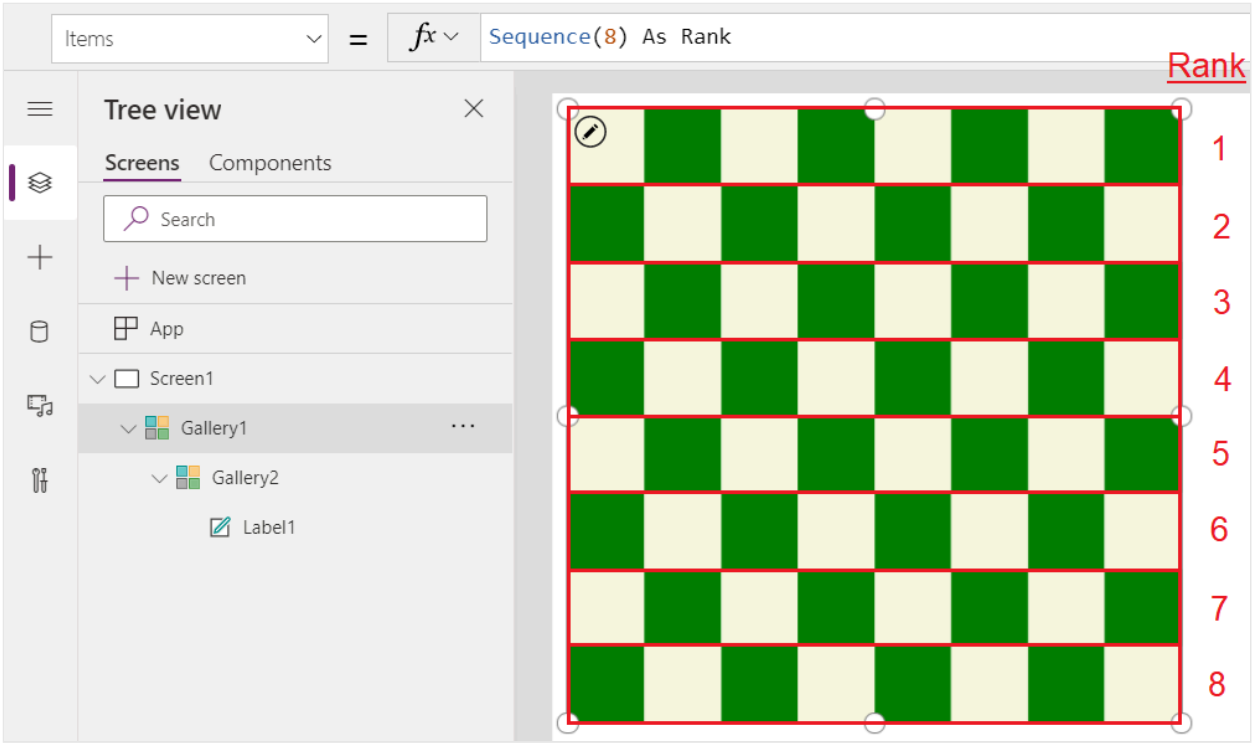
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

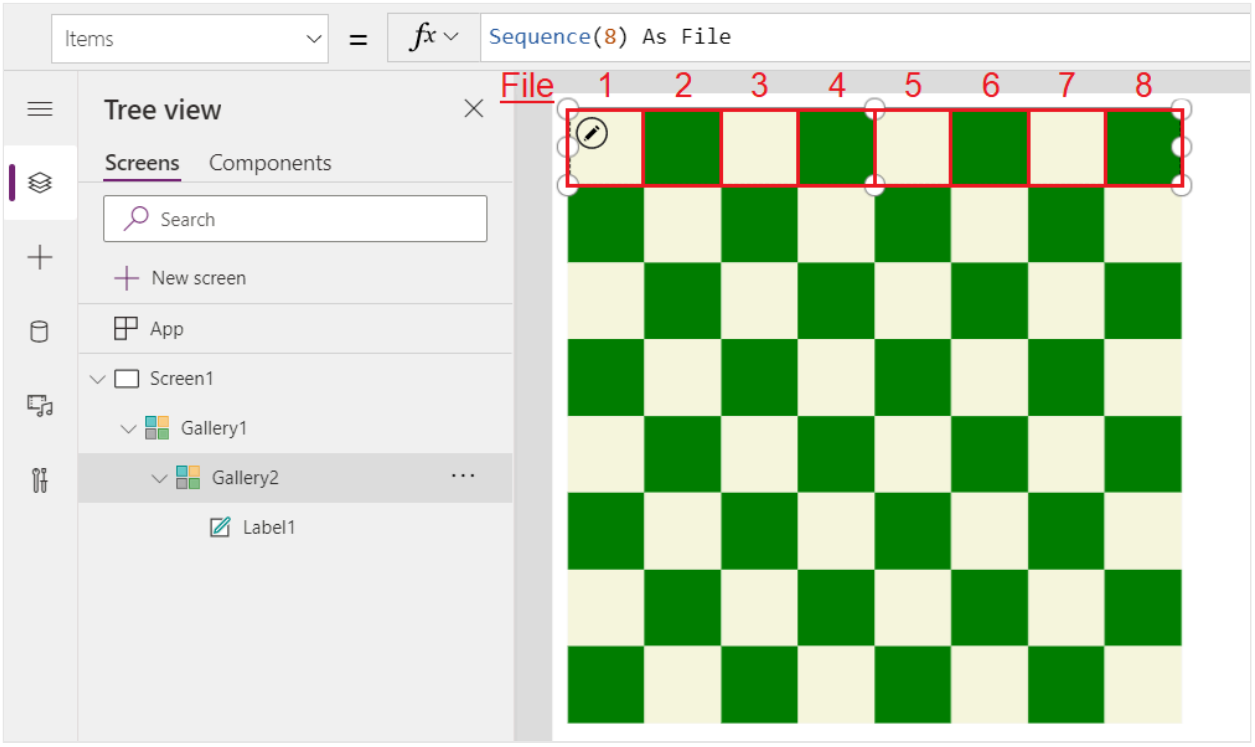
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

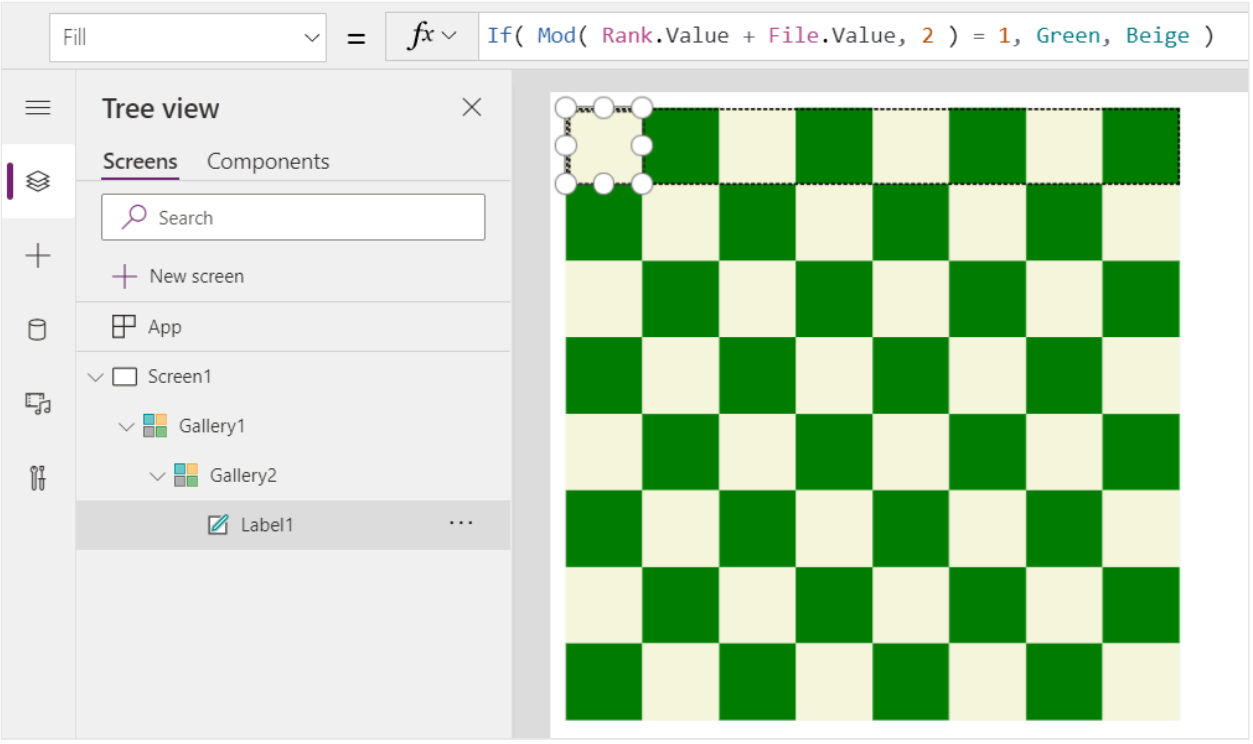
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

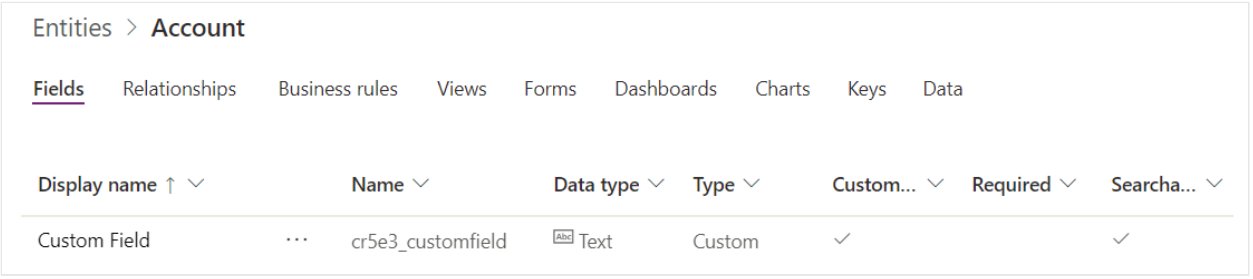
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

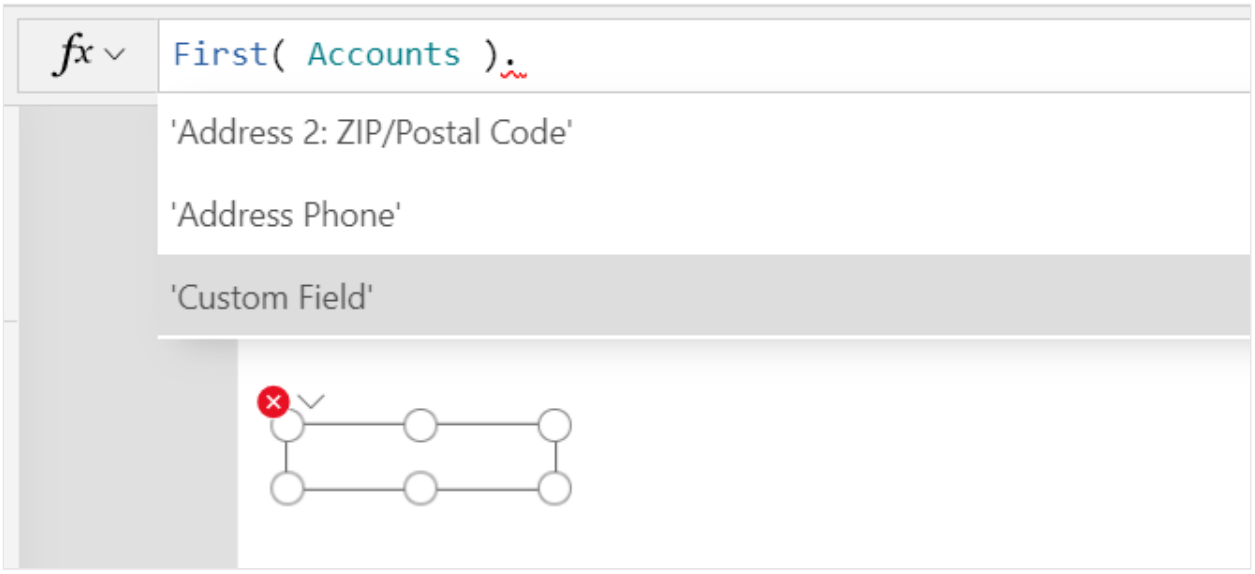
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

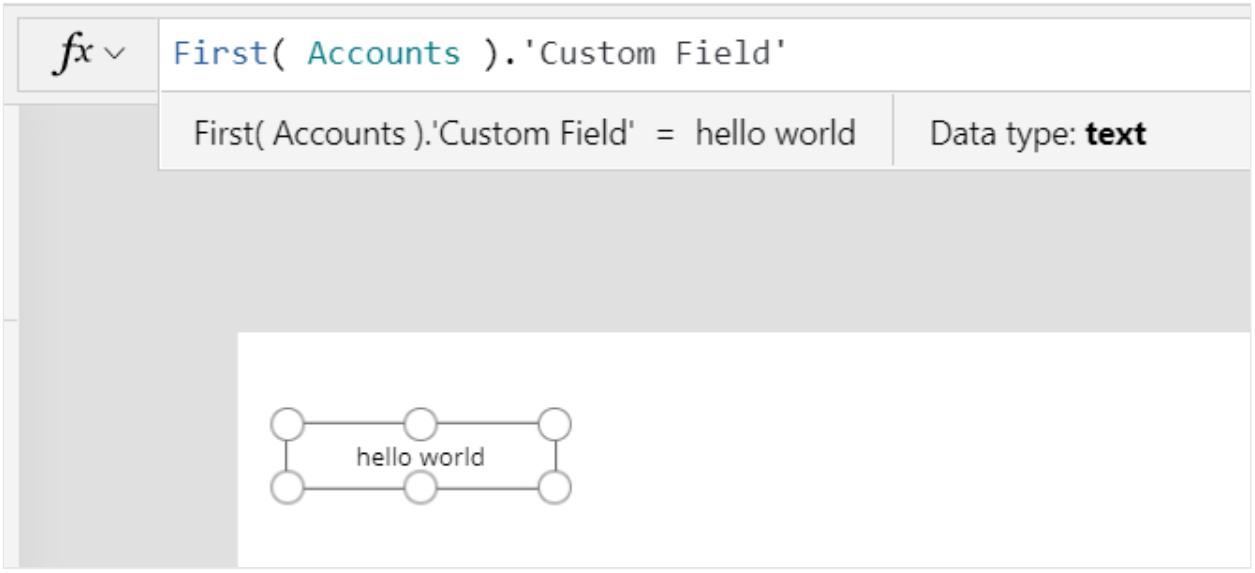
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



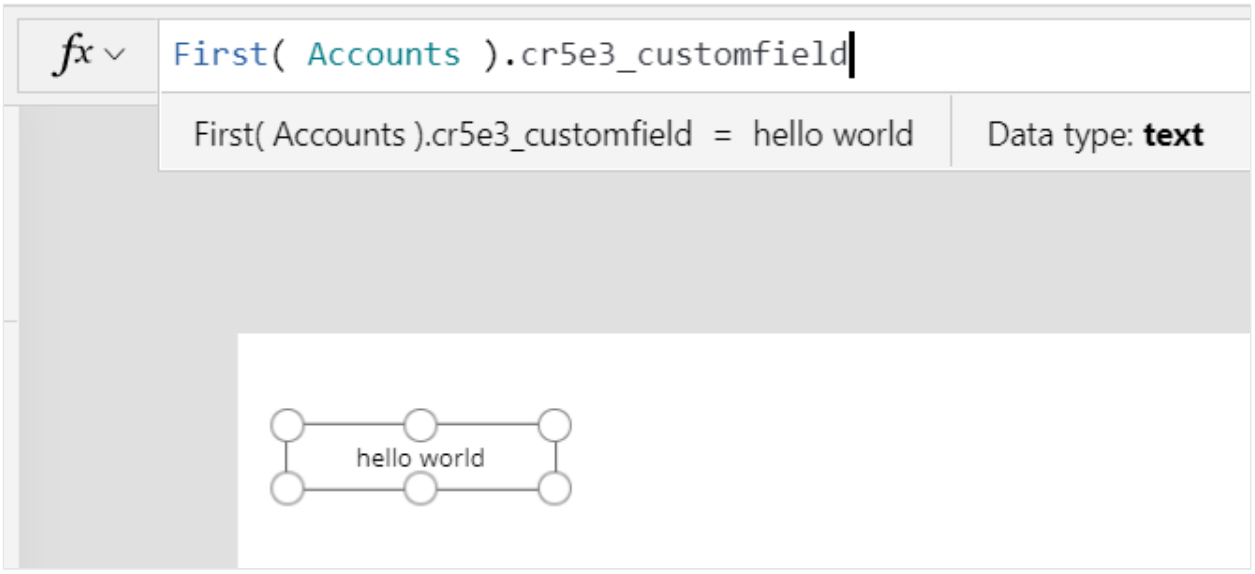
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

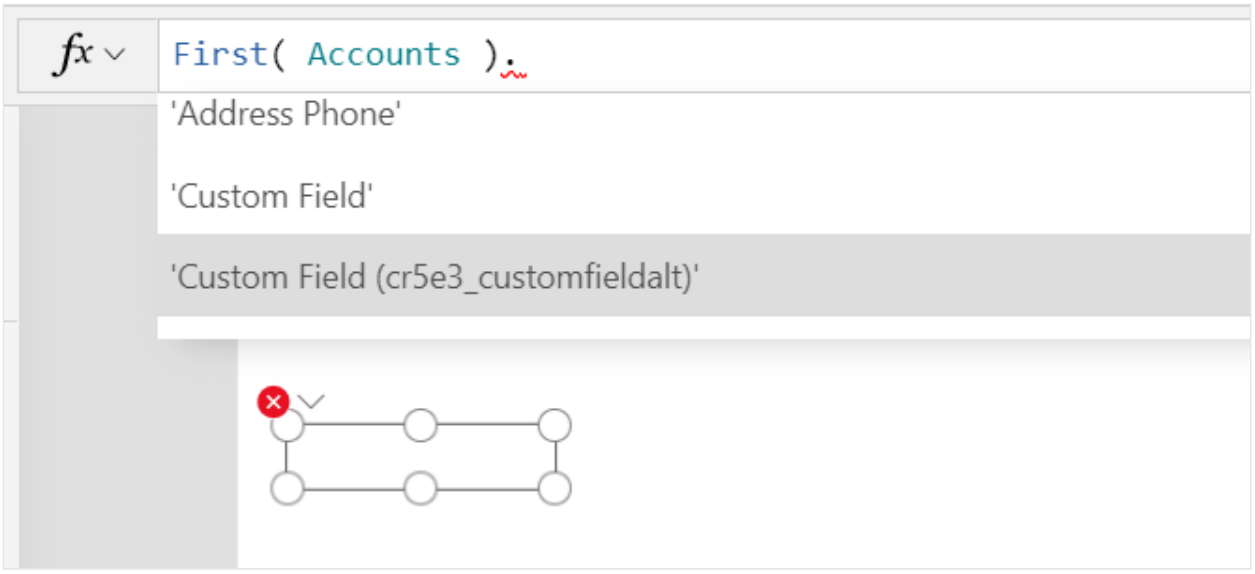


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

If and Switch functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Determines whether any condition in a set is true (**If**) or the result of a formula matches any value in a set (**Switch**) and then returns a result or executes an action.

Description

The **If** function tests one or more conditions until a **true** result is found. If such a result is found, a corresponding value is returned. If no such result is found, a default value is returned. In either case, the returned value might be a string to show, a formula to evaluate, or another form of result.

The **Switch** function evaluates a formula and determines whether the result matches any value in a sequence that you specify. If a match is found, a corresponding value is returned. If no match is found, a default value is returned. In either case, the returned value might be a string to show, a formula to evaluate, or another form of result.

If and **Switch** are very similar, but you should use the best function for your situation:

- Use **If** to evaluate a single condition. The most common syntax for this function is **If**(*Condition*, *ThenResult*, *DefaultResult*), which provides the common "if ... then ... else ..." pattern seen in other programming tools.
- Use **If** to evaluate multiple unrelated conditions. In Power Apps (unlike Microsoft Excel), you can specify multiple conditions without having to nest **If** formulas.
- Use **Switch** to evaluate a single condition against multiple possible matches. You can also use **If** in this case, but you'd need to repeat the formula for each possible match.

You can use both of these functions in behavior formulas to branch between two or more actions. Only one branch will trigger an action. Conditions and matches are evaluated in order, and they stop if a condition is **true** or a match is found.

Blank is returned if no conditions are **true**, no matches are found, and you don't specify a default result.

Syntax

If(*Condition*, *ThenResult* [, *DefaultResult*])
If(*Condition1*, *ThenResult1* [, *Condition2*, *ThenResult2*, ... [, *DefaultResult*]])

- *Condition(s)* - Required. Formula(s) to test for **true**. Such formulas commonly contain comparison operators (such as <, >, and =) and test functions such as **IsBlank** and **IsEmpty**.
- *ThenResult(s)* - Required. The corresponding value to return for a condition that evaluates to **true**.
- *DefaultResult* - Optional. The value to return if no condition evaluates to **true**. If you don't specify this argument, *blank* is returned.

Switch(*Formula*, *Match1*, *Result1* [, *Match2*, *Result2*, ... [, *DefaultResult*]])

- *Formula* - Required. Formula to evaluate for matches. This formula is evaluated only once.
- *Match(s)* - Required. Values to compare with the result from *Formula*. If an exact match is found, the corresponding *Result* is returned.
- *Result(s)* - Required. The corresponding value to return when an exact match is found.
- *DefaultResult* - Optional. If an exact match isn't found, this value is returned. If you don't specify this argument, *blank* is returned.

Examples

Values in formulas

In the following examples, a **Slider** control (named **Slider1**) has a value of **25**.

Expand table

Formula	Description	Result
If(Slider1.Value = 25, "Result1")	The condition is **true**, and the corresponding result is returned.	"Result1"
If(Slider1.Value = 25, "Result1", "Result2")	The condition is **true**, and the corresponding result is returned.	"Result1"
If(Slider1.Value > 1000, "Result1")	The condition is **false**, and no *DefaultResult* was provided.	*blank*
If(Slider1.Value > 1000, "Result1", "Result2")	The condition is **false**, a *DefaultResult* was provided, and it's returned.	"Result2"

Formula	Description	Result
If(Slider1.Value = 25, "Result1", Slider1.Value > 0, "Result2")	The first condition is **true**, and the corresponding result is returned. The second condition is also **true**, but it isn't evaluated because it appears later in the argument list than a condition that evaluates to **true**.	"Result1"
If(IsBlank(Slider1.Value), "Result1", IsNumeric(Slider1.Value), "Result2")	The first condition is **false** because the slider isn't *blank*. The second condition is **true** because the slider's value is a number, and the corresponding result is returned.	"Result2"
If(Slider1.Value > 1000, "Result1", Slider1.Value > 50, "Result2", "Result3")	Both the first and second conditions are **false**, a *DefaultResult* was provided, and it's returned.	"Result3"
Switch(Slider1.Value, 25, "Result1")	The slider's value matches the first value to be checked, and the corresponding result is returned.	"Result1"
Switch(Slider1.Value, 20, "Result1", 25, "Result2", 30, "Result3")	The slider's value matches the second value to be checked, and the corresponding result is returned.	"Result2"
Switch(Slider1.Value, 20, "Result1", 10, "Result2", 0, "Result3", "DefaultResult")	The slider's value doesn't match any value to be checked. A *DefaultResult* was provided, so it's returned.	"DefaultResult"

Branching in behavior formulas

In these examples, a Text input control named **FirstName** has the value "John" typed into it.

Expand table

Formula	Description	Result
If(! IsBlank(FirstName.Text), Navigate(Screen1, ScreenTransition.None))	The condition is **true**, so the Navigate function runs. You can use the IsBlank function to test whether a required form field has been filled in. If **FirstName** were blank, this formula would have no effect.	**true** The display is changed to **Screen1**.
If(IsBlank(FirstName.Text), Navigate(Screen1,	Without the **!** operator, the condition is **false**, so the Navigate	**true**

Formula	Description	Result
ScreenTransition.None), Back())	function doesn't run. The **Back** function was provided as a *DefaultResult*, so it runs.	The display goes back to the screen that was previously shown.
Switch(FirstName.Text, "Carlos", Navigate(Screen1, ScreenTransition.None), "Kirstin", Navigate(Screen2, ScreenTransition.None), "John", Navigate(Screen3, ScreenTransition.None))	The value of **FirstName.Text** is compared against "Carlos", "Kirstin", and "John" in that order. A match is found with "John", so the app navigates to **Screen3**.	**true** The display is changed to **Screen3**.

Step by step

1. Add a **Text input** control, and name it **Text1** if it doesn't have that name by default.
2. In **Text1**, type **30**.
3. Add a **Label** control, and set its **Text** property to this formula:
   If(Value(Text1.Text) < 20, "Order MANY more!", Value(Text1.Text) < 40, "Order more!", Text1.Text)

   The **Label** control shows **Order more!** because the value of **Text1** is more than 20 but less than 40.
4. In **Text1**, type **15**.

   The **Label** control shows **Order MANY more!** because the value of **Text1** is less than 20.
5. In **Text1**, type **50**.

   The **Label** control shows the value that you typed because it's more than 40.

Feedback

Was this page helpful? Yes No

Provide product feedback

Error, IfError, IsError, IsBlankOrError functions

Article • 10/16/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Detects errors and provides an alternative value or takes action. Create a custom error or pass through an error.

> **Note**
>
> If an app has disabled the *Formula-level error management* feature under **Settings** > **Updates** > **Retired**, those functions will not work correctly.

IfError

The **IfError** function tests values until it finds an error. If the function discovers an error, the function evaluates and returns a corresponding replacement value and stops further evaluation. A default value can also be supplied for when no errors are found. The structure of **IfError** resembles that of the **If** function: **IfError** tests for errors, while **If** tests for **true**.

Use **IfError** to replace an error with a valid value so that downstream calculations can continue. For example, use this function if user input might result in a division by zero:

Power Apps

```
IfError( 1/x, 0 )
```

This formula returns `0` if the value of `x` is zero, as `1/x` produces an error. If `x` isn't zero, then `1/x` is returned.

Stopping further processing

When chaining formulas together in behavior formulas, such as:

Power Apps

```
Patch( DS1, ... );
Patch( DS2, ... )
```

The second Patch function to `DS2` will be attempted even if the **Patch** to `DS1` fails. The scope of an error is limited to each formula that is chained.

Use **IfError** to do an action and only continue processing if the action was successful. Applying **IfError** to this example:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" ),
    Patch( DS2, ... ), Notify( "problem in the second action" )
)
```

If the **Patch** of `DS1` has a problem, the first **Notify** is executed. No further processing occurs including the second **Patch** of `DS2`. If the first **Patch** succeeds, the second **Patch** will execute.

If supplied, the optional *DefaultResult* argument is returned if no errors are discovered. Without this argument, the last *Value* argument is returned.

Building on the last example, the return value from **IfError** can be checked to determine if there were any problems:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" );  false,
    Patch( DS2, ... ), Notify( "problem in the second action" ); false,
    true
)
```

Type compatibility

IfError returns the value of one of its arguments. The types of all values that might be returned by **IfError** must be compatible.

In the last example, **Patch** returns a record that isn't compatible with the Booleans used for the *Replacement* formulas or the *DefaultResult*. Which is fine, since there's no situation in which the return value from these **Patch** calls would be returned by **IfError**.

> ⓘ **Note**
>
> While the behavior in process for a change, the types of all arguments to **IfError** must be compatible currently.

In the simple example described earlier:

```
Power Apps
IfError( 1/x, 0 )
```

The types of `1/x` and `0` were compatible as both were numbers. If they're not, the second argument is coerced to match the type of the first argument.

Excel displays **#DIV/0!** when a division by zero occurs.

Consider **IfError** with the following instead:

```
Power Apps
IfError( 1/x, "#DIV/0!" )
```

The above formula won't work. The text string `"#DIV/0!"` is coerced to the type of the first argument to **IfError**, which is a number. The result of **IfError** is yet another error since the text string can't be coerced. As a fix, convert the first argument to a text string so that **IfError** always returns a text string:

```
Power Apps
IfError( Text( 1/x ), "#DIV/0!" )
```

As seen above, **IfError** can return an error if the *Replacement* or *DefaultResult* is an error.

FirstError / AllErrors

Within in the replacement formulas, information about the errors found is available through the **FirstError** record and **AllErrors** table. **AllErrors** is a table of error information records with **FirstError** being a shortcut to the first record of this table. **FirstError** always returns the same value as **First(AllErrors)**.

Error records include:

Expand table

Field	Type	Description
Kind	ErrorKind enum (number)	Category of the error.
Message	Text string	Message about the error, suitable to be displayed to the end user.
Source	Text string	Location in where the error originated, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Observed	Text string	Location in where the error is surfaced to the user, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Details	Record	Details about the error. Currently, details are provided only for network errors. This record includes **HttpStatusCode** which contains the HTTP status code and **HttpResponse** which contains the body of the response from the connector or service.

For example, consider the following formula as a **Button** control's **OnSelect** property:

Power Apps

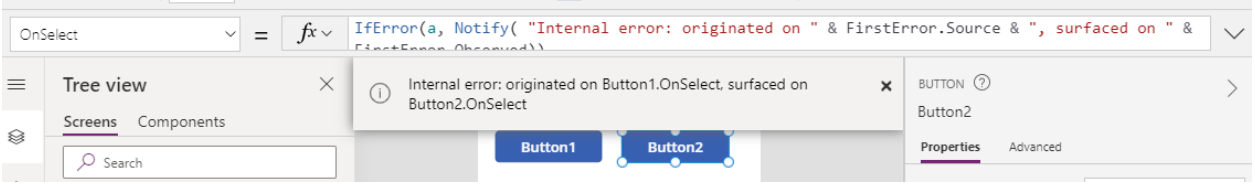
```
Set( a, 1/0 )
```

And this formula on the **OnSelect** property of a second **Button** control:

Power Apps

```
IfError( a, Notify( "Internal error: originated on " & FirstError.Source &
", surfaced on " & FirstError.Observed ) )
```

The example formula above would display the following banner when the two buttons are activated in sequence:



Typically, there will be only one error that **FirstError** can sufficiently work with. However, there are scenarios where multiple errors may be returned. For example, when using a formula chaining operator or the **Concurrent** function. Even in these situations,

reporting **FirstError** might be enough to reveal a problem instead overloading a user with multiple errors. If you still have a requirement to work with each error individually, you can use the **AllErrors** table.

IsError

The **IsError** function tests for an error value.

The return value is a Boolean *true* or *false*.

Using **IsError** prevents any further processing of the error.

IsBlankOrError

The **IsBlankOrError** function tests for either a blank value or an error value and is the equivalent of `Or( IsBlank( X ), IsError( X ) )`.

When enabling error handling for existing apps, consider replacing **IsBlank** with **IsBlankOrError** to preserve existing app behavior. Before the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value is a boolean *true* or *false*.

Using **IsBlankOrError** prevents any further processing of the error.

Use the **Error** function to create and report a custom error. For example, you might have logic to determine whether any given value is valid for your context or not—something not checked for a problem automatically. You can create and return your own error, complete with **Kind** and **Message**, using the same record described above for the **IfError** function.

In the context of **IfError**, use the **Error** function to rethrow or pass through an error. For example, your logic in **IfError** may decide that in some cases an error can be safely ignored, but in other cases the error is important to send through. Within **IfError** or **App.OnError**, use **Error(FirstError)** to pass through an error.

The **Error** function can also be passed a table of errors, as would be found in the **AllErrors** table. Use **Error(AllErrors)** to rethrow all the errors and not just the first.

A *blank* record or empty table passed to **Error** results in no error.

Syntax

Error(**ErrorRecord**)
Error(**ErrorTable**)

- *ErrorRecord* – Required. Error information record, including **Kind**, **Message**, and other fields. **Kind** is required. **FirstError** can be passed directly.
- *ErrorTable* – Required. Table of error information records. **AllErrors** can be passed directly.

IfError(*Value1*, *Replacement1* [, *Value2*, *Replacement2*, ... [, *DefaultResult*]])

- *Value(s)* – Required. Formula(s) to test for an error value.
- *Replacement(s)* – Required. The formulas to evaluate and values to return if matching *Value* arguments returned an error.
- *DefaultResult* – Optional. The formulas to evaluate if the formula doesn't find any errors.

IsError(*Value*)
IsBlankOrError(*Value*)

- *Value* – Required. Formula to test.

Examples

Simple IfError

Expand table

Formula	Description	Result
IfError(1, 2)	The first argument isn't an error. The function has no other errors to check and no default return value. The function returns the last *value* argument evaluated.	1
IfError(1/0, 2)	The first argument returns an error value (because of division by zero). The function evaluates the second argument and returns it as the result.	2
IfError(10, 20, 30)	The first argument isn't an error. The function has no other errors to check but does have a default return value. The function returns the *DefaultResult* argument.	30
IfError(10, 11, 20, 21, 300)	The first argument **10** isn't an error, so the function doesn't evaluate that argument's corresponding replacement **11**. The third argument	300

Formula	Description	Result
	20 isn't an error either, so the function doesn't evaluate that argument's corresponding replacement **21**. The fifth argument **300** has no corresponding replacement and is the default result. The function returns that result because the formula contains no errors.	
IfError(1/0, Notify("There was an internal problem"))	The first argument returns an error value (due to division by zero). The function evaluates the second argument and displays a message to the user. The return value of **IfError** is the return value of **Notify**, coerced to the same type as the first argument to **IfError** (a number).	1

Simple IsError

Expand table

Formula	Description	Result
IsError(1)	The argument isn't an error.	*false*
IsError(Blank())	The argument is a blank, but not an error.	*false*
IsError(1/0)	The argument is an error.	*true*
If(IsError(1/0), Notify("There was an internal problem"))	The argument to **IsError** returns an error value (because of division by zero). This function returns *true*, which causes the **If** to display a message to the user with the **Notify** function. The return value of **If** is the return value of **Notify**, coerced to the same type as the first argument to **If** (a boolean).	*true*

Simple IsBlankOrError

Expand table

Formula	Description	Result
IsBlankOrError(1)	The argument isn't an error or a blank.	*false*
IsBlankOrError(Blank())	The argument is a blank.	*true*
IsBlankOrError(1/0)	The argument is an error.	*true*

Simple Error

In this example, dates are validated relative to one another, resulting in an error if there is a problem.

Power Apps

```
If( StartDate > EndDate,
    Error( { Kind: ErrorKind.Validation, Message: "Start Date must be before
End Date" } ) )
```

In this example, some errors are allowed to pass through while others are suppressed and replaced with a value. In the first case, **b** is in an error state because the **Value** function has an invalid argument. Because this is unexpected by the formula writer, it is passed through so the user sees it. In the second case, with the same formula, **b** has the value 0, resulting in a division by zero. In this case, the formula writer may know that this is acceptable for this logic, suppress the error (no banner is shown), and return -1 instead.

Power Apps

```
With( {a: 1, b: Value("a")},
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns an error with Kind = ErrorKind.InvalidArgument

With( {a: 1, b: 0} )
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns -1
```

The **AllErrors** table can be filtered like any other table. Used with the **Error** function, expected errors can be removed and the remaining errors retained and reported. For example, if we knew that division by zero wasn't going to be a problem in a particular context, those errors could be filtered out, leaving all other errors intact with the following formula:

Power Apps

```
Error( Filter( AllErrors, Kind <> ErrorKind.Div0 ) )
```

Step by step

1. Add a Text input control, and name it **TextInput1** if it doesn't have that name by default.

2. Add a Label control, and name it **Label1** if it doesn't have that name by default.

3. Set the formula for **Label1**'s **Text** property to:

Power Apps

```
IfError( Value( TextInput1.Text ), -1 )
```

4. In **TextInput1**, enter **1234**.

   Label1 shows the value **1234** as this is a valid input to the Value function.

5. In **TextInput1**, enter **ToInfinity**.

   Label1 shows the value **-1** as this isn't a valid input to the Value function. Without wrapping the Value function with IfError, the label would show no value as the error value is treated as a *blank*.

See also

Formula reference for Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

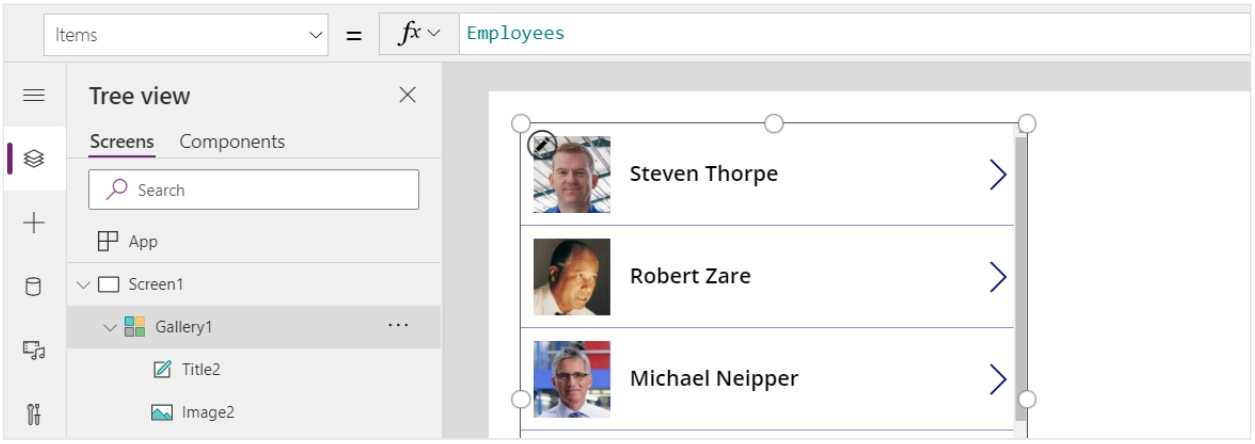
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

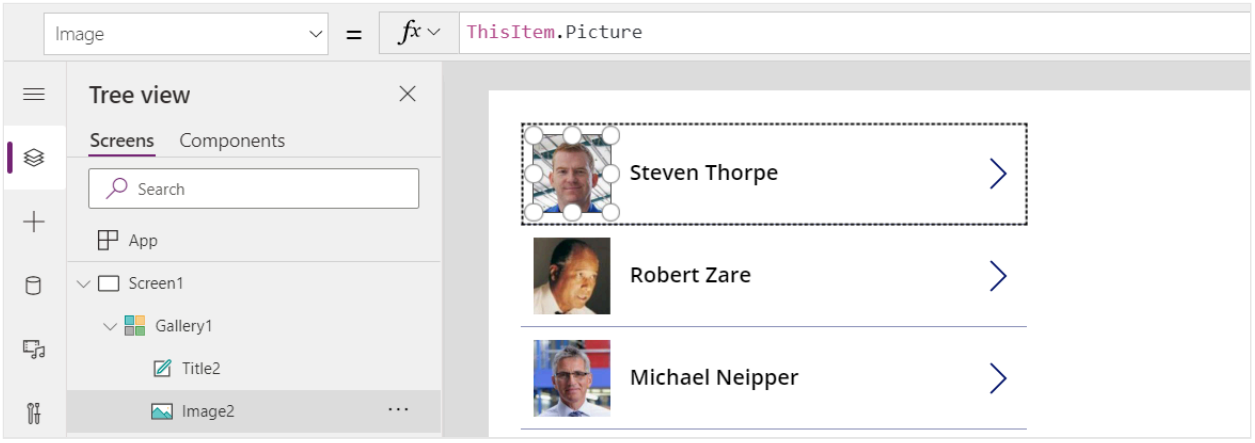
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

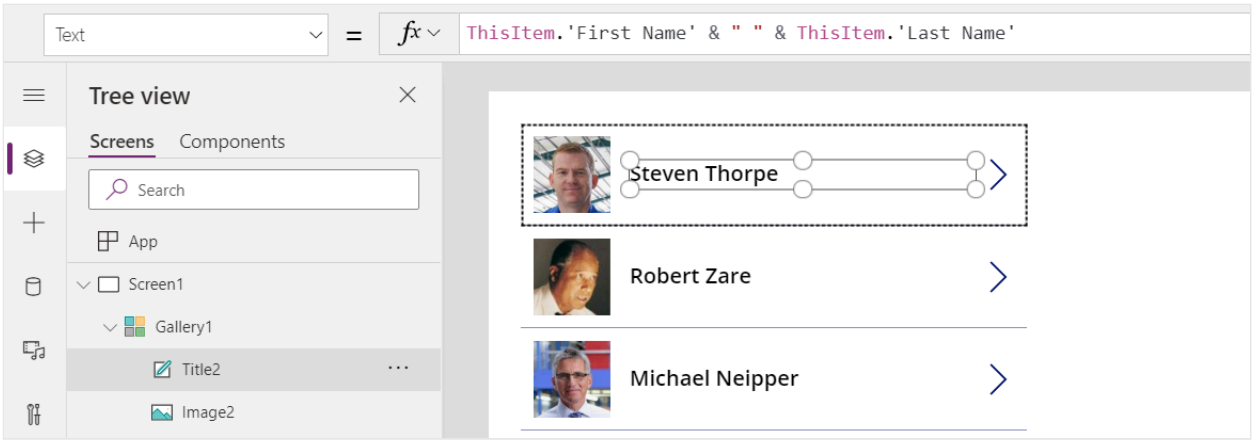
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

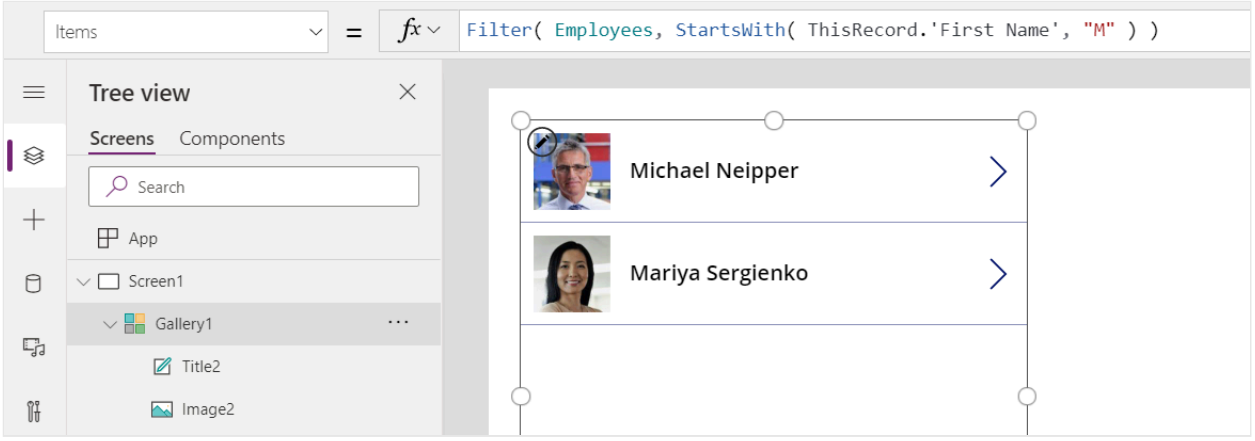


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

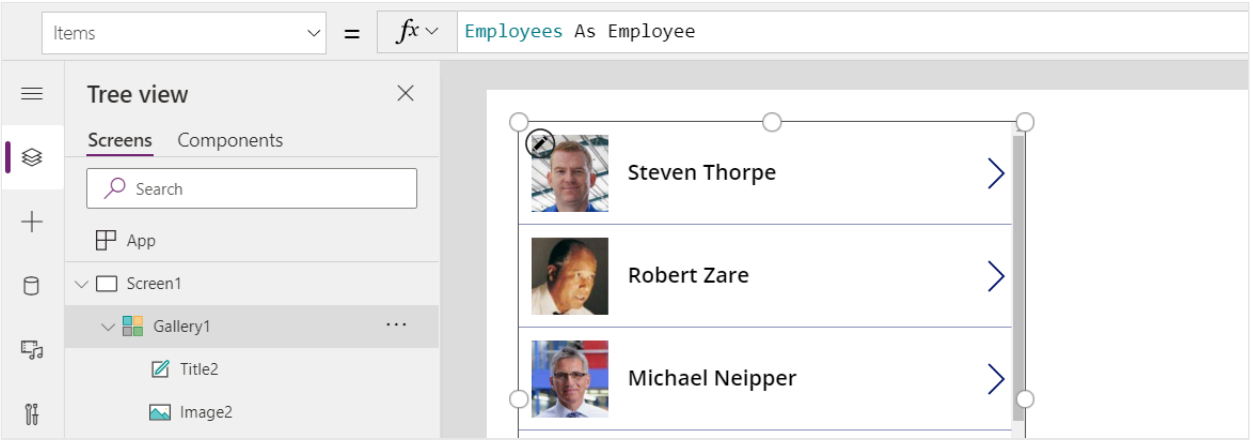
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

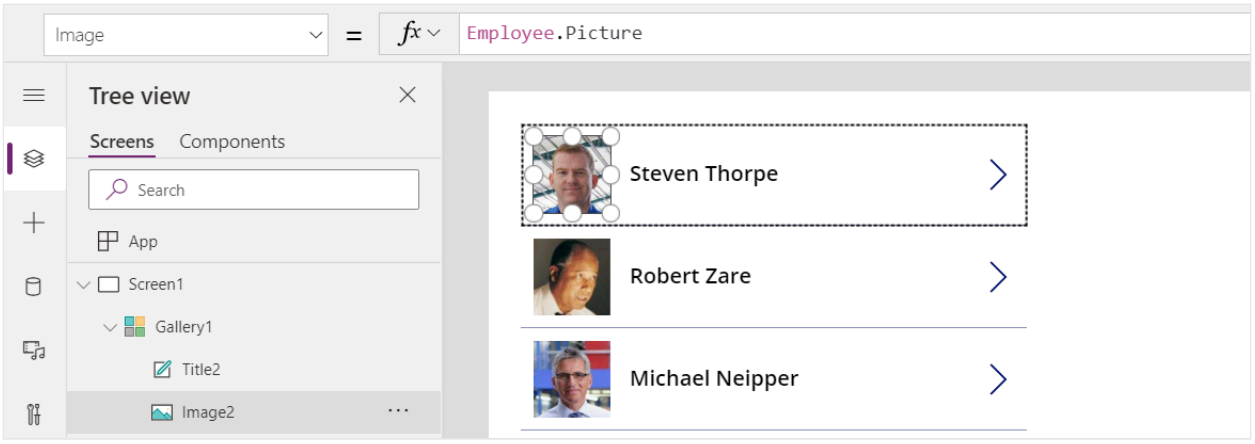
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

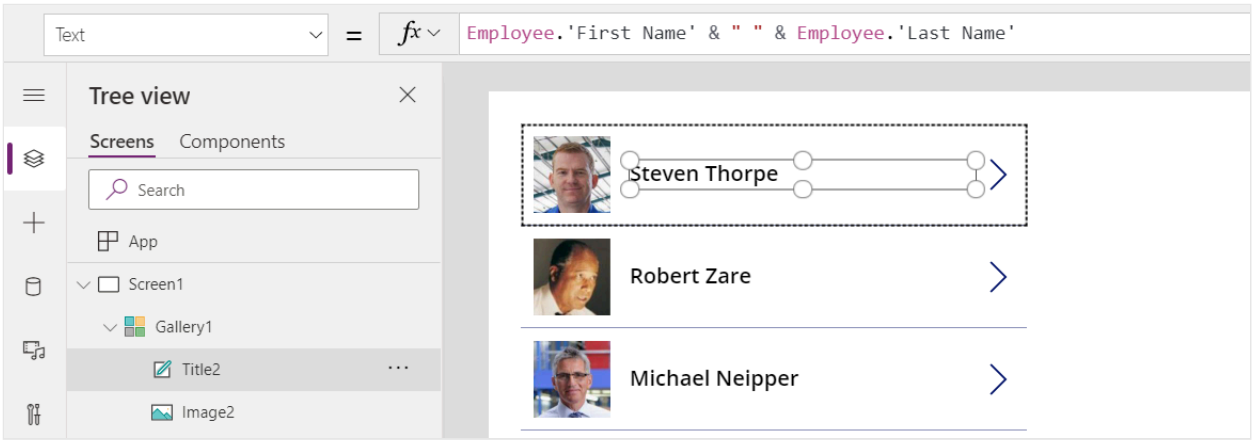
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

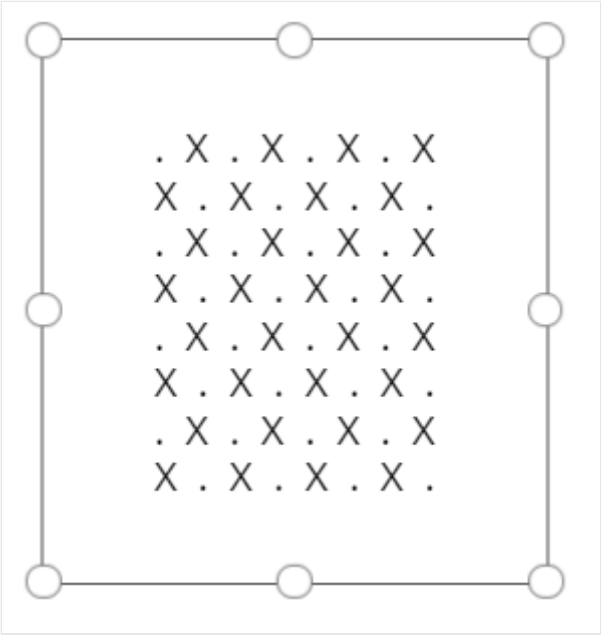
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



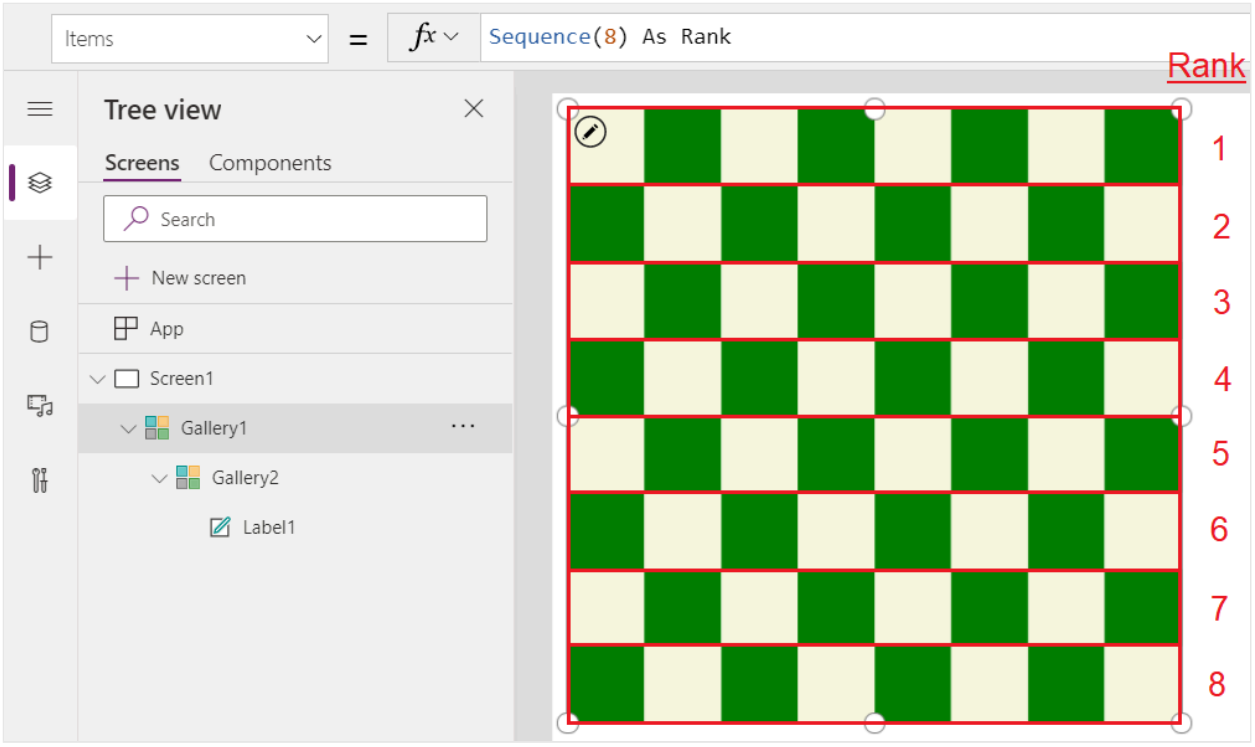
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

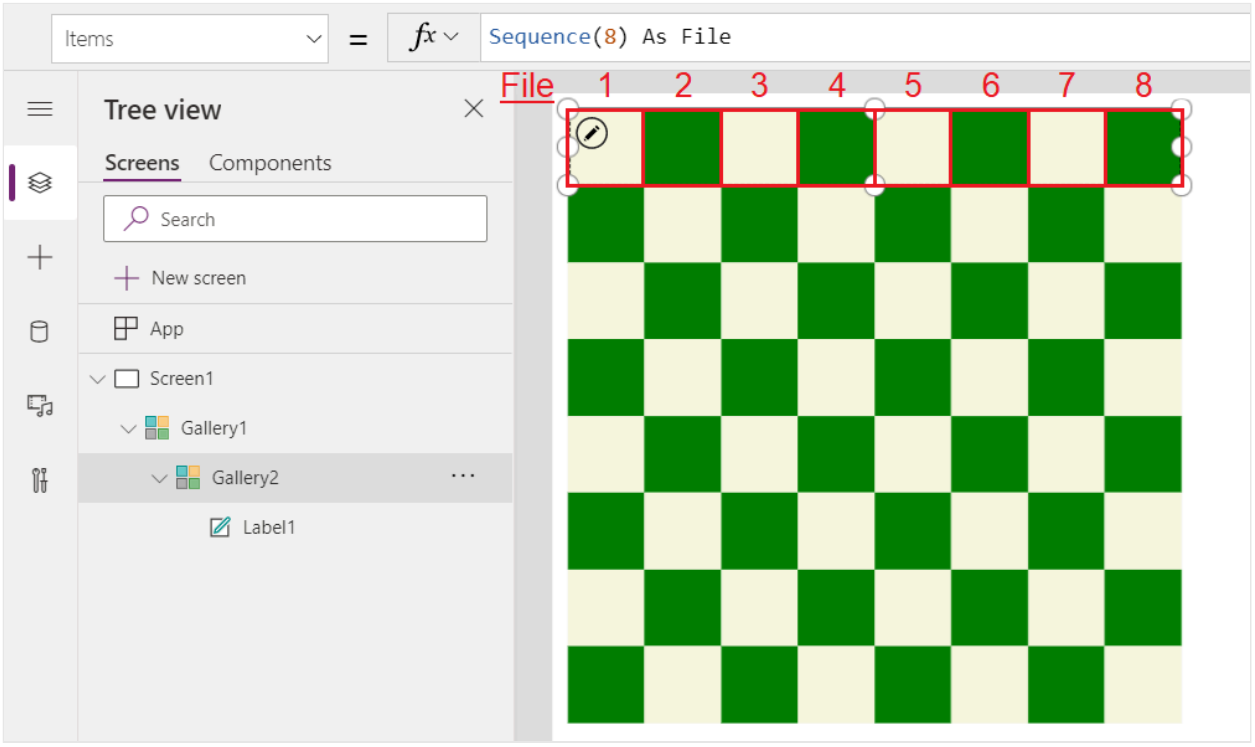
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

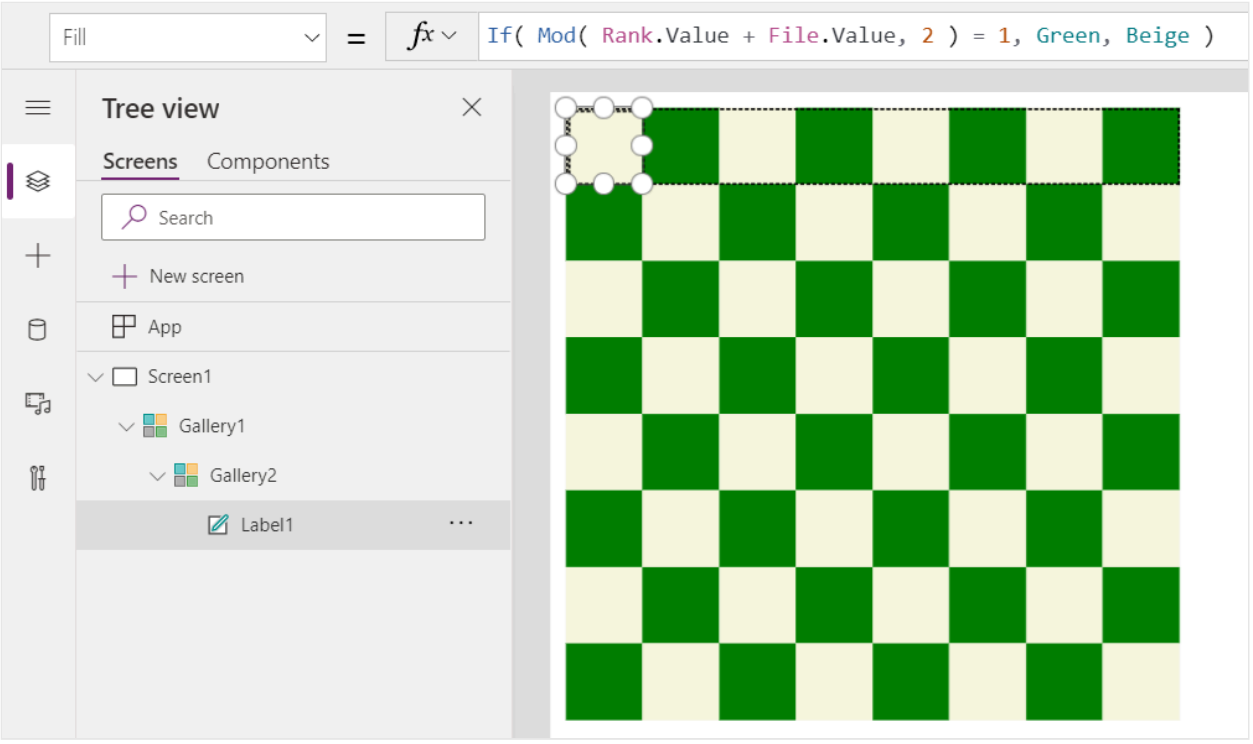
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

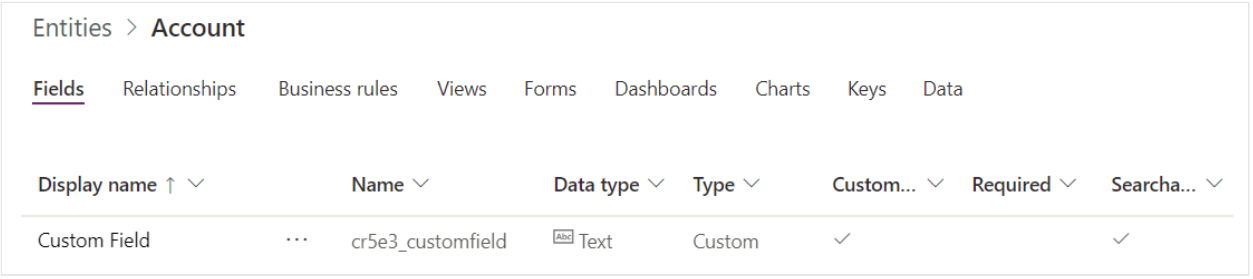
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

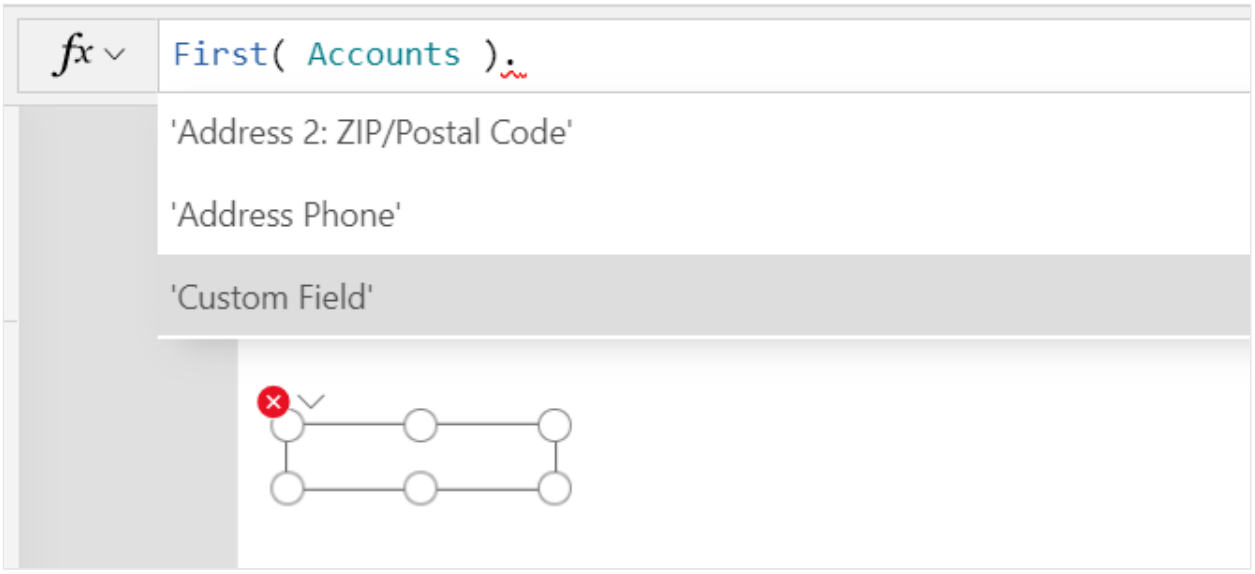
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

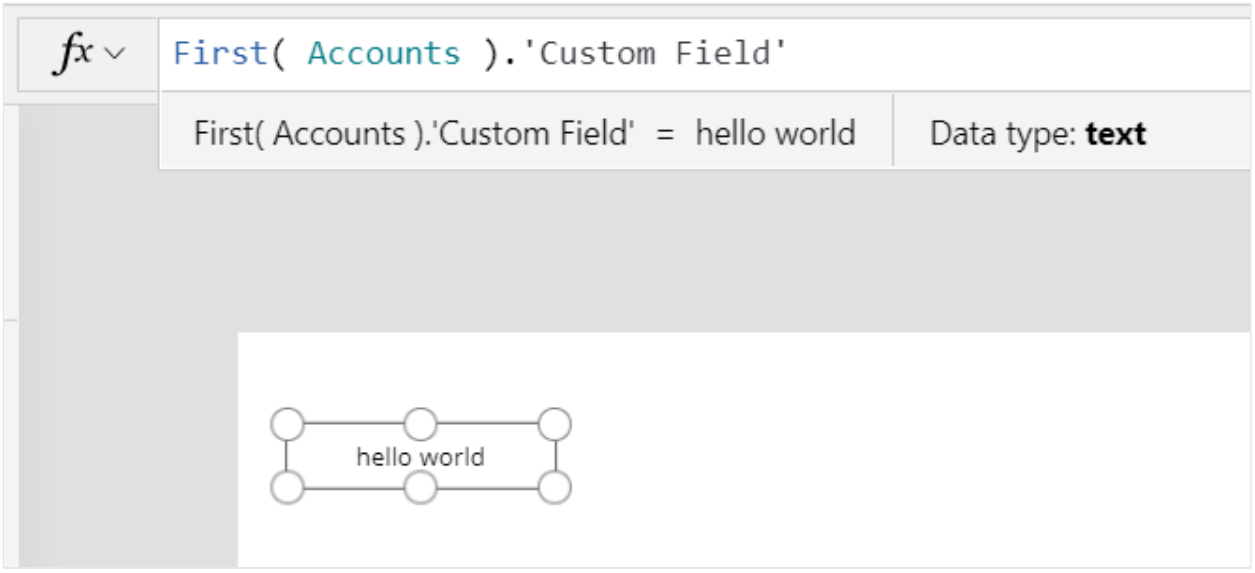
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



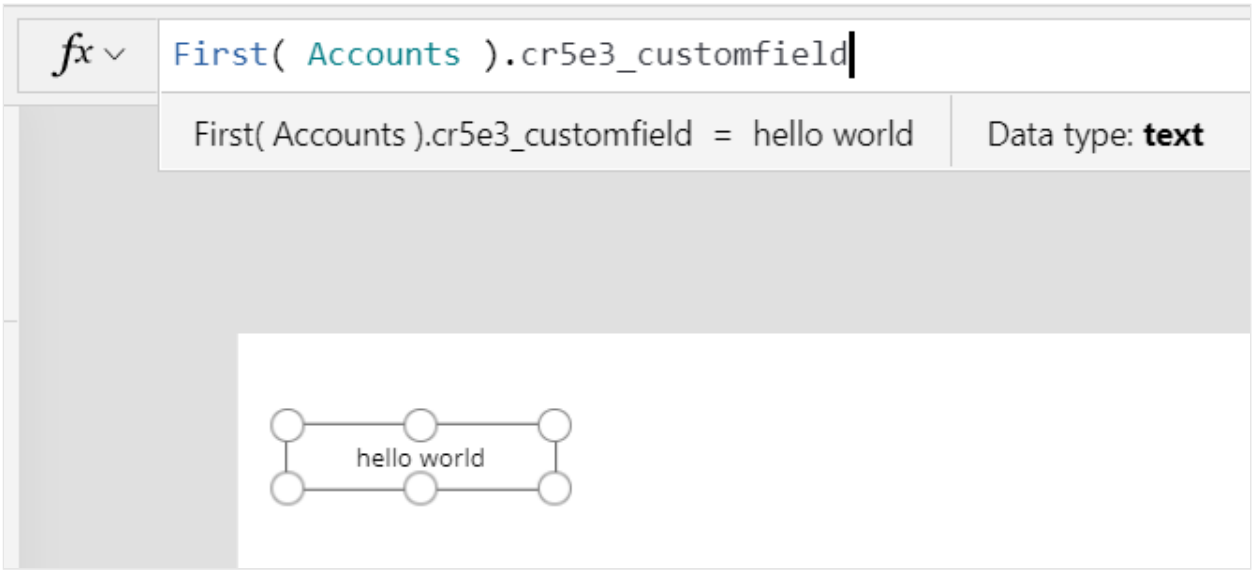
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

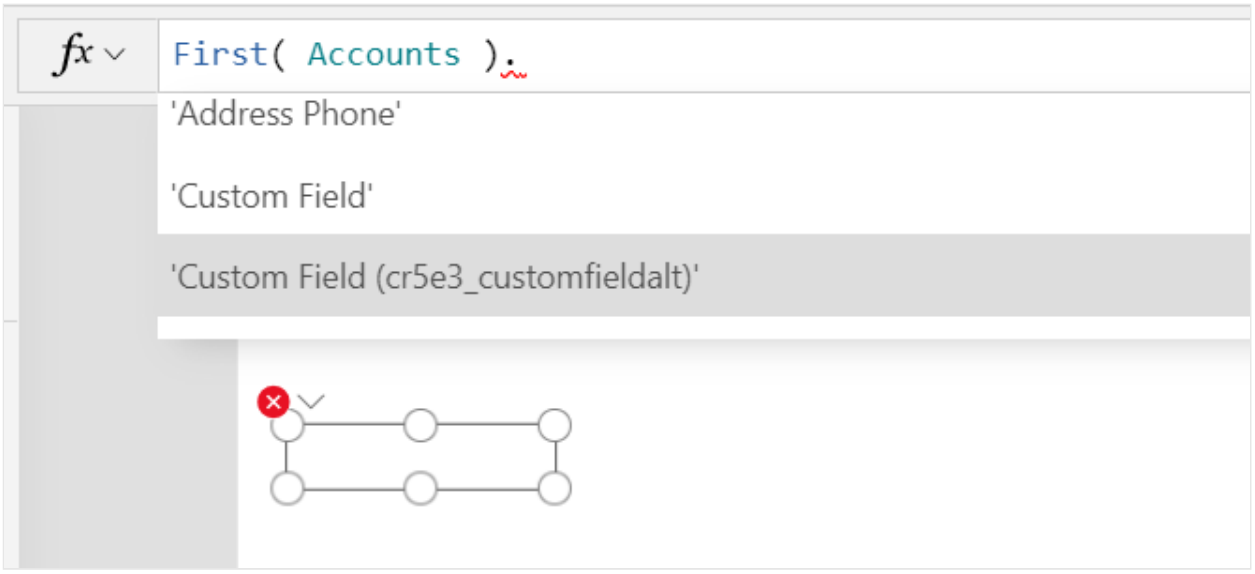


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

First, FirstN, Index, Last, and LastN functions

04/20/2025

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the first, last, or a specific record, or a set of first or last records, from a table.

Description

The **First** function returns the first record of a table.

The **FirstN** function returns the first set of records of a table; the second argument specifies the number of records to return.

The **Last** function returns the last record of a table.

The **LastN** function returns the last set of records of a table; the second argument specifies the number of records to return.

The **Index** function returns a record of a table based on its ordered position in the table. Record numbering begins with 1 so `First( table )` returning the same record as `Index( table, 1 )`. **Index** returns an error if the requested record index is less than 1, greater than the number of records in the table, or the table is empty.

First, **Index**, and **Last** return a single record. **FirstN** and **LastN** return a table, even if you specify only a single record.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

For example, when used with a data source containing a large table with 1 million records, **Last** will be subject to the non-delegation limit and will not return the last record of the entire data source. Likewise, using **Index** to request a record in the middle of 1 million records will result in an error because the index is out of range based on the non-delegation limit.

Syntax

First(*Table*)
Last(*Table*)

- *Table* - Required. Table to operate on.

FirstN(*Table* [, *NumberOfRecords*])
LastN(*Table* [, *NumberOfRecords*])

- *Table* - Required. Table to operate on.
- *NumberOfRecords* - Optional. Number of records to return. If you don't specify this argument, the function returns one record.

Index(*Table*, *RecordIndex*)

- *Table* - Required. Table to operate on.
- *RecordIndex* - Required. The index of the record to return. Record numbering begins with 1.

Examples

For the following examples, we'll use the **IceCream** data source, which contains the data in this table:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200
"Strawberry"	300
"Mint Chocolate"	60
"Pistachio"	200

This table can be placed in a collection with this formula (put in the OnStart formula for a Button control and press the button):

Power Fx

```
Collect( IceCream, Table( { Flavor: "Chocolate", Quantity: 100 },
                          { Flavor: "Vanilla", Quantity: 200 },
                          { Flavor: "Strawberry", Quantity: 300 },
                          { Flavor: "Mint Chocolate", Quantity: 60 },
                          { Flavor: "Pistachio", Quantity: 200 } ) )
```

Expand table

Formula	Description	Result
First(IceCream)	Returns the first record of **IceCream**.	{ Flavor: "Chocolate", Quantity: 100 }
Last(IceCream)	Returns the last record of **IceCream**.	{ Flavor: "Pistachio", Quantity: 200 }
Index(IceCream, 3)	Returns the third record of **IceCream**.	{ Flavor: "Strawberry", Quantity: 300 }
FirstN(IceCream, 2)	Returns a table containing the first two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Chocolate"</td><td>100</td></tr><tr><td>"Vanilla"</td><td>200</td></tr></table>
LastN(IceCream, 2)	Returns a table containing the last two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Mint Chocolate"</td><td>60</td></tr><tr><td>"Pistachio"</td><td>200</td></tr></table>
Index(IceCream, 4).Quantity	Returns the fourth record of the table, and extracts the Quantity column.	60
Index(IceCream, 10)	Returns an error since the record requested is beyond the bounds of the table.	*Error*

Int, Round, RoundDown, RoundUp, and Trunc functions

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Rounds a number.

Round, RoundDown, and RoundUp

The **Round**, **RoundDown**, and **RoundUp** functions round a number to the specified number of decimal places:

- **Round** rounds up if the next digit is 5 or higher. Otherwise, this function rounds down.
- **RoundDown** always rounds down to the previous lower number, towards zero.
- **RoundUp** always rounds up to the next higher number, away from zero.

The number of decimal places can be specified for these functions:

Expand table

Decimal places	Description	Example
Greater than 0	The number is rounded to the right of the decimal separator.	`Round( 12.37, 1 )` returns 12.4.
0	The number is rounded to the nearest integer.	`Round( 12.37, 0 )` returns 12.
Less than 0	The number is rounded to the left of the decimal separator.	`Round( 12.37, -1 )` returns 10.

Int and Trunc

The **Int** and **Trunc** functions round a number to an integer (whole number without a decimal):

- **Int** rounds down to the nearest integer.
- **Trunc** truncates the number to just the integer portion by removing any decimal portion.

The difference between **Int** and **Trunc** is in the handling of negative numbers. For example, for an argument of `-4.3`, **Int** will return the integer further away from zero, `-5`, while **Trunc** will return the integer closer to zero, `-4`. **Int** returns values that are unique amongst the five rounding functions, while **Trunc** returns the same values as **RoundDown**.

Use **Trunc** to extract the decimal portion of a number by subtracting it from the original, for example `X - Trunc(X)`.

Decimal places cannot be specified with **Trunc** as it can with Microsoft Excel. Use **RoundDown** instead when this is needed.

Single-column tables

These functions support single-column tables. If you pass a single number, the return value is the rounded version of that number. If you pass a single-column table that contains numbers, the return value is a single-column table of rounded numbers. The *DecimalPlaces* parameter can be a single value or a single-column table. If the single-column table has less values that the *Number*, zero is used for the remaining values. Use **ShowColumns** and other table shaping functions to extract a single-column table from a larger table.

Syntax

Round(*Number*, *DecimalPlaces*)
RoundDown(*Number*, *DecimalPlaces*)
RoundUp(*Number*, *DecimalPlaces*)

- *Number* - Required. The number to round.
- *DecimalPlaces* - Required. Number of decimal places to round to. Use a positive value to indicate decimal places right of the decimal separator, a negative value to the left, and zero for a whole number.

Int(*Number*)
Trunc(*Number*)

- *Number* - Required. The number to be rounded to an integer.

Examples

Rounding to a whole number.

Expand table

X	Round(X, 0)	RoundUp(X, 0)	RoundDown(X, 0)	Int(X)	Trunc(X)
7.9	8	8	7	7	7
-7.9	-8	-8	-7	-8	-7
7.5	8	8	7	7	7
-7.5	-8	-8	-7	-8	-7
7.1	7	8	7	7	7
-7.1	-7	-8	-7	-8	-7

Rounding to two decimal places to the right of the decimal separator (0.01).

Expand table

X	Round(X, 2)	RoundUp(X, 2)	RoundDown(X, 2)
430.123	430.12	430.13	430.12
430.125	430.13	430.13	430.12
430.128	430.13	430.13	430.12

Rounding to two decimal places to the left of the decimal separator (100).

Expand table

X	Round(X, -2)	RoundUp(X, -2)	RoundDown(X, -2)
430.123	400	500	400
449.942	400	500	400
450.000	500	500	400
450.124	500	500	400
479.128	500	500	400

Rounding a single-column table of values.

Expand table

`X`	`Int( X )`	`Round( X, 2 )`	`RoundDown( X, [ 0, 1, 2 ] )`	`RoundUp( X, [ 2 ] )`
[123.456, 987.593, 542.639]	[123, 987, 542]	[123.46, 987.59, 542.64]	[123, 987.5, 542.63]	[123.46, 988, 543]

Blank, Coalesce, IsBlank, and IsEmpty functions

06/19/2025

Expand table

Functions	Applies to
Blank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
Coalesce **IsEmpty**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
IsBlank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests whether a value is blank or a table contains no records, and provides a way to create *blank* values.

Overview

Blank is a placeholder for "no value" or "unknown value." For example, a **Combo box** control's **Selected** property is *blank* if the user hasn't made a selection. Many data sources can store and return NULL values, which are represented in Power Apps as *blank*.

Any property or calculated value in Power Apps can be *blank*. For example, a Boolean value normally has one of two values: **true** or **false**. But in addition to these two, it can also be *blank* indicating that the state is not known. This is similar to Microsoft Excel, where a worksheet cell starts out as blank with no contents but can hold the values **TRUE** or **FALSE** (among others). At any time, the contents of the cell can again be cleared, returning it to a *blank* state.

Empty string refers to a string that contains no characters. The **Len** function returns zero for such a string and it can be written in a formulas as two double quotes with nothing in between `""`. Some controls and data sources use an empty string to indicate a "no value" condition. To simplify app creation, the **IsBlank** and **Coalesce** functions test for both *blank* values or empty strings.

In the context of the **IsEmpty** function, *empty* is specific to tables that contain no records. The table structure may be intact, complete with column names, but no data is in the table. A table

may start as empty, take on records and no longer be empty, and then have the records removed and again be empty.

Blank

The **Blank** function returns a *blank* value. Use this to store a NULL value in a data source that supports these values, effectively removing any value from the field.

IsBlank

The **IsBlank** function tests for a *blank* value or an empty string. The test includes empty strings to ease app creation since some data sources and controls use an empty string when there is no value present. To test specifically for a *blank* value use `if( Value = Blank(), ...` instead of **IsBlank**. The **IsBlank** function considers empty tables as not blank, and **IsEmpty** should be used to test a table.

When enabling error handling for existing apps, consider replacing **IsBlank** with IsBlankOrError to preserve existing app behavior. Prior to the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value for **IsBlank** is a boolean **true** or **false**.

Coalesce

The **Coalesce** function evaluates its arguments in order and returns the first value that isn't *blank* or an empty string. Use this function to replace a *blank* value or empty string with a different value but leave non-*blank* and non-empty string values unchanged. If all the arguments are *blank* or empty strings then the function returns *blank*, making **Coalesce** a good way to convert empty strings to *blank* values.

`Coalesce( value1, value2 )` is the more concise equivalent of `If( Not IsBlank( value1 ), value1, Not IsBlank( value2 ), value2 )` and doesn't require **value1** and **value2** to be evaluated twice. The **If** function returns *blank* if there is no "else" formula as is the case here.

All arguments to **Coalesce** must be of the same type; for example, you can't mix numbers with text strings. The return value from **Coalesce** is of this common type.

IsEmpty

The **IsEmpty** function tests whether a table contains any records. It's equivalent to using the CountRows function and checking for zero. You can check for data-source errors by combining **IsEmpty** with the Errors function.

The return value for **IsEmpty** is a Boolean **true** or **false**.

Syntax

Blank()

Coalesce(*Value1* [, *Value2*, ...])

- *Value(s)* – Required. Values to test. Each value is evaluated in order until a value that is not *blank* and not an empty string is found. Values after this point are not evaluated.

IsBlank(*Value*)

- *Value* – Required. Value to test for a *blank* value or empty string.

IsEmpty(*Table*)

- *Table* - Required. Table to test for records.

Examples

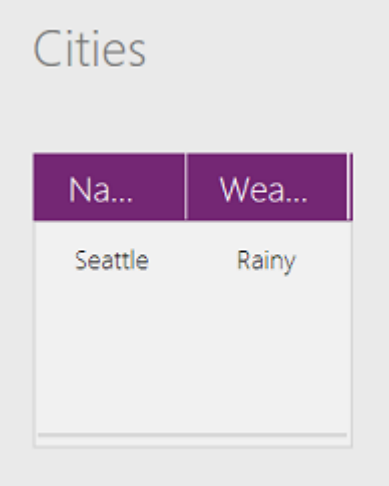
Blank

1. Create an app from scratch, and add a **Button** control.
2. Set the button's OnSelect property to this formula:

   ```
   Power Fx
   ClearCollect( Cities, { Name: "Seattle", Weather: "Rainy" } )
   ```

3. Preview your app, click or tap the button that you added, and then close Preview.
4. On the **File** menu, click or tap **Collections**.

   The **Cities** collection appears, showing one record with "Seattle" and "Rainy":



5. Click or tap the back arrow to return to the default workspace.

6. Add a **Label** control, and set its **Text** property to this formula:

```
Power Fx
IsBlank( First( Cities ).Weather )
```

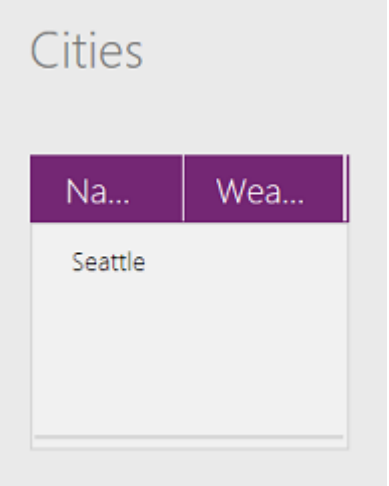
   The label shows **false** because the **Weather** field contains a value ("Rainy").

7. Add a second button, and set its **OnSelect** property to this formula:

```
Power Fx
Patch( Cities, First( Cities ), { Weather: Blank() } )
```

8. Preview your app, click or tap the button that you added, and then close Preview.

   The **Weather** field of the first record in **Cities** is replaced with a *blank*, removing the "Rainy" that was there previously.



   The label shows **true** because the **Weather** field no longer contains a value.

Coalesce

Expand table

Formula	Description	Result
Coalesce(Blank(), 1)	Tests the return value from the **Blank** function, which always returns a *blank* value. Because the first argument is *blank*, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**1**
Coalesce("", "2")	Tests the first argument which is an empty string. Because the first argument is an empty string, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**2**
Coalesce(Blank(), "", Blank(), "", "3", "4")	**Coalesce** starts at the beginning of the argument list and evaluates each argument in turn until a non-*blank* value and non-empty string is found. In this case, the first four arguments all return *blank* or an empty string, so evaluation continues to the fifth argument. The fifth argument is non-*blank* and non-empty string, so evaluation stops here. The value of the fifth argument is returned, and the sixth argument isn't evaluated.	**3**
Coalesce("")	Tests the first argument which is an empty string. Because the first argument is an empty string, and there are no more arguments, the function returns *blank*.	*blank*

IsBlank

1. Create an app from scratch, add a text-input control, and name it **FirstName**.
2. Add a label, and set its **Text** property to this formula:

   ```
   Power Fx
   ```

   ```
   If( IsBlank( FirstName.Text ), "First Name is a required field." )
   ```

   By default, the **Text** property of a text-input control is set to **"Text input"**. Because the property contains a value, it isn't blank, and the label doesn't display any message.
3. Remove all the characters from the text-input control, including any spaces.

   Because the **Text** property no longer contains any characters, it's an empty string, and **IsBlank(FirstName.Text)** will be **true**. The required field message is displayed.

For information about how to perform validation by using other tools, see the **Validate** function and working with data sources.

Other examples:

Expand table

Formula	Description	Result
IsBlank(Blank())	Tests the return value from the **Blank** function, which always returns a *blank* value.	**true**
IsBlank("")	A string that contains no characters.	**true**
IsBlank("Hello")	A string that contains one or more characters.	**false**
IsBlank(*AnyCollection*)	Because the collection exists, it isn't blank, even if it doesn't contain any records. To check for an empty collection, use **IsEmpty** instead.	**false**
IsBlank(Mid("Hello", 17, 2))	The starting character for **Mid** is beyond the end of the string. The result is an empty string.	**true**
IsBlank(If(false, false))	An **If** function with no *ElseResult*. Because the condition is always **false**, this **If** always returns *blank*.	**true**

IsEmpty

1. Create an app from scratch, and add a **Button** control.
2. Set the button's **OnSelect** property to this formula:

   Collect(IceCream, { Flavor: "Strawberry", Quantity: 300 }, { Flavor: "Chocolate", Quantity: 100 })
3. Preview your app, click or tap the button that you added, and then close Preview.

   A collection named **IceCream** is created and contains this data:

Flavor	Quantity
"Strawberry"	300
"Chocolate"	100

   This collection has two records and isn't empty. **IsEmpty(IceCream)** returns **false**, and **CountRows(IceCream)** returns **2**.
4. Add a second button, and set its **OnSelect** property to this formula:

   Clear(IceCream)
5. Preview your app, click or tap the second button, and then close Preview.

The collection is now empty:

Flavor	Quantity

The Clear function removes all the records from a collection, resulting in an empty collection. **IsEmpty(IceCream)** returns **true**, and **CountRows(IceCream)** returns **0**.

You can also use **IsEmpty** to test whether a calculated table is empty, as these examples show:

Expand table

Formula	Description	Result
IsEmpty([1, 2, 3])	The single-column table contains three records and, therefore, isn't empty.	**false**
IsEmpty([ ])	The single-column table contains no records and is empty.	**true**
IsEmpty(Filter([1, 2, 3], Value > 5))	The single-column table contains no values that are greater than 5. The result from the filter doesn't contain any records and is empty.	**true**

Error, IfError, IsError, IsBlankOrError functions

Article • 10/16/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Detects errors and provides an alternative value or takes action. Create a custom error or pass through an error.

> **Note**
>
> If an app has disabled the *Formula-level error management* feature under **Settings** > **Updates** > **Retired**, those functions will not work correctly.

IfError

The **IfError** function tests values until it finds an error. If the function discovers an error, the function evaluates and returns a corresponding replacement value and stops further evaluation. A default value can also be supplied for when no errors are found. The structure of **IfError** resembles that of the **If** function: **IfError** tests for errors, while **If** tests for **true**.

Use **IfError** to replace an error with a valid value so that downstream calculations can continue. For example, use this function if user input might result in a division by zero:

Power Apps

```
IfError( 1/x, 0 )
```

This formula returns `0` if the value of `x` is zero, as `1/x` produces an error. If `x` isn't zero, then `1/x` is returned.

Stopping further processing

When chaining formulas together in behavior formulas, such as:

Power Apps

```
Patch( DS1, ... );
Patch( DS2, ... )
```

The second Patch function to `DS2` will be attempted even if the **Patch** to `DS1` fails. The scope of an error is limited to each formula that is chained.

Use **IfError** to do an action and only continue processing if the action was successful. Applying **IfError** to this example:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" ),
    Patch( DS2, ... ), Notify( "problem in the second action" )
)
```

If the **Patch** of `DS1` has a problem, the first **Notify** is executed. No further processing occurs including the second **Patch** of `DS2`. If the first **Patch** succeeds, the second **Patch** will execute.

If supplied, the optional *DefaultResult* argument is returned if no errors are discovered. Without this argument, the last *Value* argument is returned.

Building on the last example, the return value from **IfError** can be checked to determine if there were any problems:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" );  false,
    Patch( DS2, ... ), Notify( "problem in the second action" ); false,
    true
)
```

Type compatibility

IfError returns the value of one of its arguments. The types of all values that might be returned by **IfError** must be compatible.

In the last example, **Patch** returns a record that isn't compatible with the Booleans used for the *Replacement* formulas or the *DefaultResult*. Which is fine, since there's no situation in which the return value from these **Patch** calls would be returned by **IfError**.

> ⓘ **Note**
>
> While the behavior in process for a change, the types of all arguments to **IfError** must be compatible currently.

In the simple example described earlier:

```
Power Apps
IfError( 1/x, 0 )
```

The types of `1/x` and `0` were compatible as both were numbers. If they're not, the second argument is coerced to match the type of the first argument.

Excel displays **#DIV/0!** when a division by zero occurs.

Consider **IfError** with the following instead:

```
Power Apps
IfError( 1/x, "#DIV/0!" )
```

The above formula won't work. The text string `"#DIV/0!"` is coerced to the type of the first argument to **IfError**, which is a number. The result of **IfError** is yet another error since the text string can't be coerced. As a fix, convert the first argument to a text string so that **IfError** always returns a text string:

```
Power Apps
IfError( Text( 1/x ), "#DIV/0!" )
```

As seen above, **IfError** can return an error if the *Replacement* or *DefaultResult* is an error.

FirstError / AllErrors

Within in the replacement formulas, information about the errors found is available through the **FirstError** record and **AllErrors** table. **AllErrors** is a table of error information records with **FirstError** being a shortcut to the first record of this table. **FirstError** always returns the same value as **First(AllErrors)**.

Error records include:

Expand table

Field	Type	Description
Kind	ErrorKind enum (number)	Category of the error.
Message	Text string	Message about the error, suitable to be displayed to the end user.
Source	Text string	Location in where the error originated, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Observed	Text string	Location in where the error is surfaced to the user, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Details	Record	Details about the error. Currently, details are provided only for network errors. This record includes **HttpStatusCode** which contains the HTTP status code and **HttpResponse** which contains the body of the response from the connector or service.

For example, consider the following formula as a **Button** control's **OnSelect** property:

Power Apps

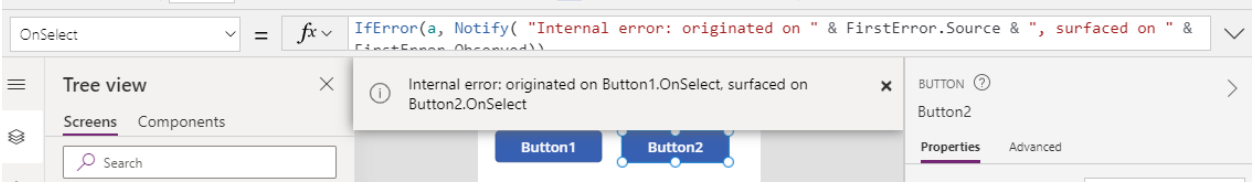
```
Set( a, 1/0 )
```

And this formula on the **OnSelect** property of a second **Button** control:

Power Apps

```
IfError( a, Notify( "Internal error: originated on " & FirstError.Source &
", surfaced on " & FirstError.Observed ) )
```

The example formula above would display the following banner when the two buttons are activated in sequence:



Typically, there will be only one error that **FirstError** can sufficiently work with. However, there are scenarios where multiple errors may be returned. For example, when using a formula chaining operator or the **Concurrent** function. Even in these situations,

reporting **FirstError** might be enough to reveal a problem instead overloading a user with multiple errors. If you still have a requirement to work with each error individually, you can use the **AllErrors** table.

IsError

The **IsError** function tests for an error value.

The return value is a Boolean *true* or *false*.

Using **IsError** prevents any further processing of the error.

IsBlankOrError

The **IsBlankOrError** function tests for either a blank value or an error value and is the equivalent of `Or( IsBlank( X ), IsError( X ) )`.

When enabling error handling for existing apps, consider replacing **IsBlank** with **IsBlankOrError** to preserve existing app behavior. Before the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value is a boolean *true* or *false*.

Using **IsBlankOrError** prevents any further processing of the error.

Use the **Error** function to create and report a custom error. For example, you might have logic to determine whether any given value is valid for your context or not—something not checked for a problem automatically. You can create and return your own error, complete with **Kind** and **Message**, using the same record described above for the **IfError** function.

In the context of **IfError**, use the **Error** function to rethrow or pass through an error. For example, your logic in **IfError** may decide that in some cases an error can be safely ignored, but in other cases the error is important to send through. Within **IfError** or **App.OnError**, use **Error(FirstError)** to pass through an error.

The **Error** function can also be passed a table of errors, as would be found in the **AllErrors** table. Use **Error(AllErrors)** to rethrow all the errors and not just the first.

A *blank* record or empty table passed to **Error** results in no error.

Syntax

Error(**ErrorRecord**)
Error(**ErrorTable**)

- *ErrorRecord* – Required. Error information record, including **Kind**, **Message**, and other fields. **Kind** is required. **FirstError** can be passed directly.
- *ErrorTable* – Required. Table of error information records. **AllErrors** can be passed directly.

IfError(*Value1*, *Replacement1* [, *Value2*, *Replacement2*, ... [, *DefaultResult*]])

- *Value(s)* – Required. Formula(s) to test for an error value.
- *Replacement(s)* – Required. The formulas to evaluate and values to return if matching *Value* arguments returned an error.
- *DefaultResult* – Optional. The formulas to evaluate if the formula doesn't find any errors.

IsError(*Value*)
IsBlankOrError(*Value*)

- *Value* – Required. Formula to test.

Examples

Simple IfError

Expand table

Formula	Description	Result
IfError(1, 2)	The first argument isn't an error. The function has no other errors to check and no default return value. The function returns the last *value* argument evaluated.	1
IfError(1/0, 2)	The first argument returns an error value (because of division by zero). The function evaluates the second argument and returns it as the result.	2
IfError(10, 20, 30)	The first argument isn't an error. The function has no other errors to check but does have a default return value. The function returns the *DefaultResult* argument.	30
IfError(10, 11, 20, 21, 300)	The first argument **10** isn't an error, so the function doesn't evaluate that argument's corresponding replacement **11**. The third argument	300

Formula	Description	Result
	20 isn't an error either, so the function doesn't evaluate that argument's corresponding replacement **21**. The fifth argument **300** has no corresponding replacement and is the default result. The function returns that result because the formula contains no errors.	
IfError(1/0, Notify("There was an internal problem"))	The first argument returns an error value (due to division by zero). The function evaluates the second argument and displays a message to the user. The return value of **IfError** is the return value of **Notify**, coerced to the same type as the first argument to **IfError** (a number).	1

Simple IsError

Expand table

Formula	Description	Result
IsError(1)	The argument isn't an error.	*false*
IsError(Blank())	The argument is a blank, but not an error.	*false*
IsError(1/0)	The argument is an error.	*true*
If(IsError(1/0), Notify("There was an internal problem"))	The argument to **IsError** returns an error value (because of division by zero). This function returns *true*, which causes the **If** to display a message to the user with the **Notify** function. The return value of **If** is the return value of **Notify**, coerced to the same type as the first argument to **If** (a boolean).	*true*

Simple IsBlankOrError

Expand table

Formula	Description	Result
IsBlankOrError(1)	The argument isn't an error or a blank.	*false*
IsBlankOrError(Blank())	The argument is a blank.	*true*
IsBlankOrError(1/0)	The argument is an error.	*true*

Simple Error

In this example, dates are validated relative to one another, resulting in an error if there is a problem.

Power Apps

```
If( StartDate > EndDate,
    Error( { Kind: ErrorKind.Validation, Message: "Start Date must be before
End Date" } ) )
```

In this example, some errors are allowed to pass through while others are suppressed and replaced with a value. In the first case, **b** is in an error state because the **Value** function has an invalid argument. Because this is unexpected by the formula writer, it is passed through so the user sees it. In the second case, with the same formula, **b** has the value 0, resulting in a division by zero. In this case, the formula writer may know that this is acceptable for this logic, suppress the error (no banner is shown), and return -1 instead.

Power Apps

```
With( {a: 1, b: Value("a")},
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns an error with Kind = ErrorKind.InvalidArgument

With( {a: 1, b: 0} )
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns -1
```

The **AllErrors** table can be filtered like any other table. Used with the **Error** function, expected errors can be removed and the remaining errors retained and reported. For example, if we knew that division by zero wasn't going to be a problem in a particular context, those errors could be filtered out, leaving all other errors intact with the following formula:

Power Apps

```
Error( Filter( AllErrors, Kind <> ErrorKind.Div0 ) )
```

Step by step

1. Add a Text input control, and name it **TextInput1** if it doesn't have that name by default.

2. Add a Label control, and name it **Label1** if it doesn't have that name by default.

3. Set the formula for **Label1**'s **Text** property to:

Power Apps

```
IfError( Value( TextInput1.Text ), -1 )
```

4. In **TextInput1**, enter **1234**.

   Label1 shows the value **1234** as this is a valid input to the Value function.

5. In **TextInput1**, enter **ToInfinity**.

   Label1 shows the value **-1** as this isn't a valid input to the Value function. Without wrapping the Value function with IfError, the label would show no value as the error value is treated as a *blank*.

See also

Formula reference for Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Blank, Coalesce, IsBlank, and IsEmpty functions

06/19/2025

Expand table

Functions	Applies to
Blank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
Coalesce **IsEmpty**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
IsBlank	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests whether a value is blank or a table contains no records, and provides a way to create *blank* values.

Overview

Blank is a placeholder for "no value" or "unknown value." For example, a **Combo box** control's **Selected** property is *blank* if the user hasn't made a selection. Many data sources can store and return NULL values, which are represented in Power Apps as *blank*.

Any property or calculated value in Power Apps can be *blank*. For example, a Boolean value normally has one of two values: **true** or **false**. But in addition to these two, it can also be *blank* indicating that the state is not known. This is similar to Microsoft Excel, where a worksheet cell starts out as blank with no contents but can hold the values **TRUE** or **FALSE** (among others). At any time, the contents of the cell can again be cleared, returning it to a *blank* state.

Empty string refers to a string that contains no characters. The **Len** function returns zero for such a string and it can be written in a formulas as two double quotes with nothing in between `""`. Some controls and data sources use an empty string to indicate a "no value" condition. To simplify app creation, the **IsBlank** and **Coalesce** functions test for both *blank* values or empty strings.

In the context of the **IsEmpty** function, *empty* is specific to tables that contain no records. The table structure may be intact, complete with column names, but no data is in the table. A table

may start as empty, take on records and no longer be empty, and then have the records removed and again be empty.

Blank

The **Blank** function returns a *blank* value. Use this to store a NULL value in a data source that supports these values, effectively removing any value from the field.

IsBlank

The **IsBlank** function tests for a *blank* value or an empty string. The test includes empty strings to ease app creation since some data sources and controls use an empty string when there is no value present. To test specifically for a *blank* value use `if( Value = Blank(), ...` instead of **IsBlank**. The **IsBlank** function considers empty tables as not blank, and **IsEmpty** should be used to test a table.

When enabling error handling for existing apps, consider replacing **IsBlank** with IsBlankOrError to preserve existing app behavior. Prior to the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value for **IsBlank** is a boolean **true** or **false**.

Coalesce

The **Coalesce** function evaluates its arguments in order and returns the first value that isn't *blank* or an empty string. Use this function to replace a *blank* value or empty string with a different value but leave non-*blank* and non-empty string values unchanged. If all the arguments are *blank* or empty strings then the function returns *blank*, making **Coalesce** a good way to convert empty strings to *blank* values.

`Coalesce( value1, value2 )` is the more concise equivalent of `If( Not IsBlank( value1 ), value1, Not IsBlank( value2 ), value2 )` and doesn't require **value1** and **value2** to be evaluated twice. The **If** function returns *blank* if there is no "else" formula as is the case here.

All arguments to **Coalesce** must be of the same type; for example, you can't mix numbers with text strings. The return value from **Coalesce** is of this common type.

IsEmpty

The **IsEmpty** function tests whether a table contains any records. It's equivalent to using the CountRows function and checking for zero. You can check for data-source errors by combining **IsEmpty** with the Errors function.

The return value for **IsEmpty** is a Boolean **true** or **false**.

Syntax

Blank()

Coalesce(*Value1* [, *Value2*, ...])

- *Value(s)* – Required. Values to test. Each value is evaluated in order until a value that is not *blank* and not an empty string is found. Values after this point are not evaluated.

IsBlank(*Value*)

- *Value* – Required. Value to test for a *blank* value or empty string.

IsEmpty(*Table*)

- *Table* - Required. Table to test for records.

Examples

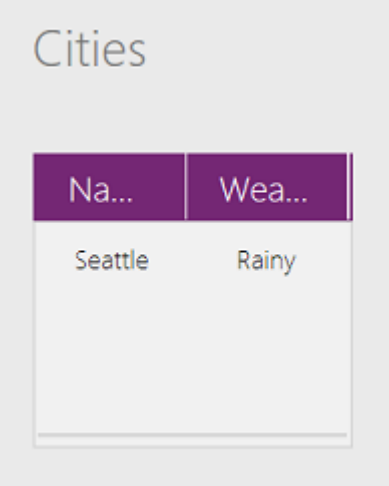
Blank

1. Create an app from scratch, and add a **Button** control.
2. Set the button's OnSelect property to this formula:

   ```
   Power Fx
   ClearCollect( Cities, { Name: "Seattle", Weather: "Rainy" } )
   ```

3. Preview your app, click or tap the button that you added, and then close Preview.
4. On the **File** menu, click or tap **Collections**.

   The **Cities** collection appears, showing one record with "Seattle" and "Rainy":



5. Click or tap the back arrow to return to the default workspace.

6. Add a **Label** control, and set its **Text** property to this formula:

```
Power Fx
IsBlank( First( Cities ).Weather )
```

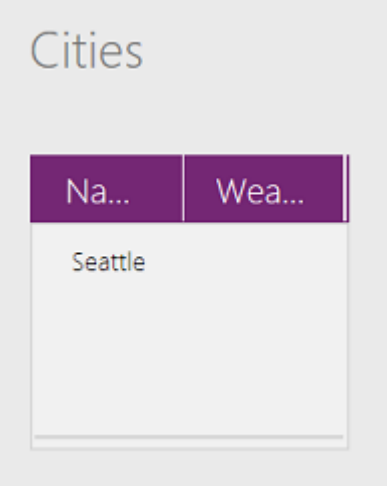
   The label shows **false** because the **Weather** field contains a value ("Rainy").

7. Add a second button, and set its **OnSelect** property to this formula:

```
Power Fx
Patch( Cities, First( Cities ), { Weather: Blank() } )
```

8. Preview your app, click or tap the button that you added, and then close Preview.

   The **Weather** field of the first record in **Cities** is replaced with a *blank*, removing the "Rainy" that was there previously.



   The label shows **true** because the **Weather** field no longer contains a value.

Coalesce

Expand table

Formula	Description	Result
Coalesce(Blank(), 1)	Tests the return value from the **Blank** function, which always returns a *blank* value. Because the first argument is *blank*, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**1**
Coalesce("", "2")	Tests the first argument which is an empty string. Because the first argument is an empty string, evaluation continues with the next argument until a non-*blank* value and non-empty string is found.	**2**
Coalesce(Blank(), "", Blank(), "", "3", "4")	**Coalesce** starts at the beginning of the argument list and evaluates each argument in turn until a non-*blank* value and non-empty string is found. In this case, the first four arguments all return *blank* or an empty string, so evaluation continues to the fifth argument. The fifth argument is non-*blank* and non-empty string, so evaluation stops here. The value of the fifth argument is returned, and the sixth argument isn't evaluated.	**3**
Coalesce("")	Tests the first argument which is an empty string. Because the first argument is an empty string, and there are no more arguments, the function returns *blank*.	*blank*

IsBlank

1. Create an app from scratch, add a text-input control, and name it **FirstName**.
2. Add a label, and set its **Text** property to this formula:

   ```
   Power Fx
   ```

   ```
   If( IsBlank( FirstName.Text ), "First Name is a required field." )
   ```

   By default, the **Text** property of a text-input control is set to **"Text input"**. Because the property contains a value, it isn't blank, and the label doesn't display any message.
3. Remove all the characters from the text-input control, including any spaces.

   Because the **Text** property no longer contains any characters, it's an empty string, and **IsBlank(FirstName.Text)** will be **true**. The required field message is displayed.

For information about how to perform validation by using other tools, see the **Validate** function and working with data sources.

Other examples:

Expand table

Formula	Description	Result
IsBlank(Blank())	Tests the return value from the **Blank** function, which always returns a *blank* value.	**true**
IsBlank("")	A string that contains no characters.	**true**
IsBlank("Hello")	A string that contains one or more characters.	**false**
IsBlank(*AnyCollection*)	Because the collection exists, it isn't blank, even if it doesn't contain any records. To check for an empty collection, use **IsEmpty** instead.	**false**
IsBlank(Mid("Hello", 17, 2))	The starting character for **Mid** is beyond the end of the string. The result is an empty string.	**true**
IsBlank(If(false, false))	An **If** function with no *ElseResult*. Because the condition is always **false**, this **If** always returns *blank*.	**true**

IsEmpty

1. Create an app from scratch, and add a **Button** control.
2. Set the button's **OnSelect** property to this formula:

   Collect(IceCream, { Flavor: "Strawberry", Quantity: 300 }, { Flavor: "Chocolate", Quantity: 100 })
3. Preview your app, click or tap the button that you added, and then close Preview.

   A collection named **IceCream** is created and contains this data:

Flavor	Quantity
"Strawberry"	300
"Chocolate"	100

   This collection has two records and isn't empty. **IsEmpty(IceCream)** returns **false**, and **CountRows(IceCream)** returns **2**.
4. Add a second button, and set its **OnSelect** property to this formula:

   Clear(IceCream)
5. Preview your app, click or tap the second button, and then close Preview.

The collection is now empty:

Flavor	Quantity

The Clear function removes all the records from a collection, resulting in an empty collection. **IsEmpty(IceCream)** returns **true**, and **CountRows(IceCream)** returns **0**.

You can also use **IsEmpty** to test whether a calculated table is empty, as these examples show:

Expand table

Formula	Description	Result
IsEmpty([1, 2, 3])	The single-column table contains three records and, therefore, isn't empty.	**false**
IsEmpty([ ])	The single-column table contains no records and is empty.	**true**
IsEmpty(Filter([1, 2, 3], Value > 5))	The single-column table contains no values that are greater than 5. The result from the filter doesn't contain any records and is empty.	**true**

Error, IfError, IsError, IsBlankOrError functions

Article • 10/16/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Detects errors and provides an alternative value or takes action. Create a custom error or pass through an error.

> **Note**
>
> If an app has disabled the *Formula-level error management* feature under **Settings** > **Updates** > **Retired**, those functions will not work correctly.

IfError

The **IfError** function tests values until it finds an error. If the function discovers an error, the function evaluates and returns a corresponding replacement value and stops further evaluation. A default value can also be supplied for when no errors are found. The structure of **IfError** resembles that of the **If** function: **IfError** tests for errors, while **If** tests for **true**.

Use **IfError** to replace an error with a valid value so that downstream calculations can continue. For example, use this function if user input might result in a division by zero:

Power Apps

```
IfError( 1/x, 0 )
```

This formula returns `0` if the value of `x` is zero, as `1/x` produces an error. If `x` isn't zero, then `1/x` is returned.

Stopping further processing

When chaining formulas together in behavior formulas, such as:

Power Apps

```
Patch( DS1, ... );
Patch( DS2, ... )
```

The second Patch function to `DS2` will be attempted even if the **Patch** to `DS1` fails. The scope of an error is limited to each formula that is chained.

Use **IfError** to do an action and only continue processing if the action was successful. Applying **IfError** to this example:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" ),
    Patch( DS2, ... ), Notify( "problem in the second action" )
)
```

If the **Patch** of `DS1` has a problem, the first **Notify** is executed. No further processing occurs including the second **Patch** of `DS2`. If the first **Patch** succeeds, the second **Patch** will execute.

If supplied, the optional *DefaultResult* argument is returned if no errors are discovered. Without this argument, the last *Value* argument is returned.

Building on the last example, the return value from **IfError** can be checked to determine if there were any problems:

Power Apps

```
IfError(
    Patch( DS1, ... ), Notify( "problem in the first action" );  false,
    Patch( DS2, ... ), Notify( "problem in the second action" ); false,
    true
)
```

Type compatibility

IfError returns the value of one of its arguments. The types of all values that might be returned by **IfError** must be compatible.

In the last example, **Patch** returns a record that isn't compatible with the Booleans used for the *Replacement* formulas or the *DefaultResult*. Which is fine, since there's no situation in which the return value from these **Patch** calls would be returned by **IfError**.

> ⓘ **Note**
>
> While the behavior in process for a change, the types of all arguments to **IfError** must be compatible currently.

In the simple example described earlier:

```
Power Apps
IfError( 1/x, 0 )
```

The types of `1/x` and `0` were compatible as both were numbers. If they're not, the second argument is coerced to match the type of the first argument.

Excel displays **#DIV/0!** when a division by zero occurs.

Consider **IfError** with the following instead:

```
Power Apps
IfError( 1/x, "#DIV/0!" )
```

The above formula won't work. The text string `"#DIV/0!"` is coerced to the type of the first argument to **IfError**, which is a number. The result of **IfError** is yet another error since the text string can't be coerced. As a fix, convert the first argument to a text string so that **IfError** always returns a text string:

```
Power Apps
IfError( Text( 1/x ), "#DIV/0!" )
```

As seen above, **IfError** can return an error if the *Replacement* or *DefaultResult* is an error.

FirstError / AllErrors

Within in the replacement formulas, information about the errors found is available through the **FirstError** record and **AllErrors** table. **AllErrors** is a table of error information records with **FirstError** being a shortcut to the first record of this table. **FirstError** always returns the same value as **First(AllErrors)**.

Error records include:

Expand table

Field	Type	Description
Kind	ErrorKind enum (number)	Category of the error.
Message	Text string	Message about the error, suitable to be displayed to the end user.
Source	Text string	Location in where the error originated, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Observed	Text string	Location in where the error is surfaced to the user, used for reporting. For example, for a formula bound to a control property, this value is in the form *ControlName.PropertyName*.
Details	Record	Details about the error. Currently, details are provided only for network errors. This record includes **HttpStatusCode** which contains the HTTP status code and **HttpResponse** which contains the body of the response from the connector or service.

For example, consider the following formula as a **Button** control's **OnSelect** property:

Power Apps

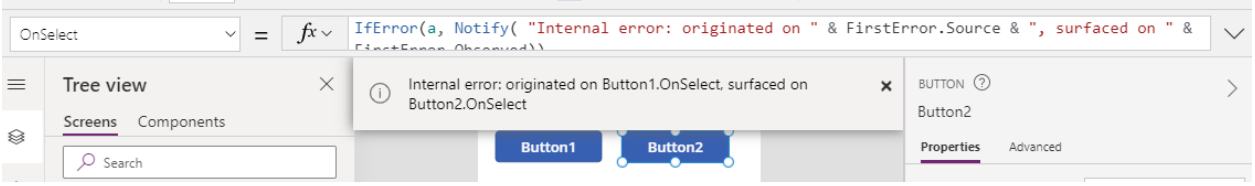
```
Set( a, 1/0 )
```

And this formula on the **OnSelect** property of a second **Button** control:

Power Apps

```
IfError( a, Notify( "Internal error: originated on " & FirstError.Source &
", surfaced on " & FirstError.Observed ) )
```

The example formula above would display the following banner when the two buttons are activated in sequence:



Typically, there will be only one error that **FirstError** can sufficiently work with. However, there are scenarios where multiple errors may be returned. For example, when using a formula chaining operator or the **Concurrent** function. Even in these situations,

reporting **FirstError** might be enough to reveal a problem instead overloading a user with multiple errors. If you still have a requirement to work with each error individually, you can use the **AllErrors** table.

IsError

The **IsError** function tests for an error value.

The return value is a Boolean *true* or *false*.

Using **IsError** prevents any further processing of the error.

IsBlankOrError

The **IsBlankOrError** function tests for either a blank value or an error value and is the equivalent of `Or( IsBlank( X ), IsError( X ) )`.

When enabling error handling for existing apps, consider replacing **IsBlank** with **IsBlankOrError** to preserve existing app behavior. Before the addition of error handling, a *blank* value was used to represent both null values from databases and error values. Error handling separates these two interpretations of *blank* which could change the behavior of existing apps that continue to use **IsBlank**.

The return value is a boolean *true* or *false*.

Using **IsBlankOrError** prevents any further processing of the error.

Use the **Error** function to create and report a custom error. For example, you might have logic to determine whether any given value is valid for your context or not—something not checked for a problem automatically. You can create and return your own error, complete with **Kind** and **Message**, using the same record described above for the **IfError** function.

In the context of **IfError**, use the **Error** function to rethrow or pass through an error. For example, your logic in **IfError** may decide that in some cases an error can be safely ignored, but in other cases the error is important to send through. Within **IfError** or **App.OnError**, use **Error(FirstError)** to pass through an error.

The **Error** function can also be passed a table of errors, as would be found in the **AllErrors** table. Use **Error(AllErrors)** to rethrow all the errors and not just the first.

A *blank* record or empty table passed to **Error** results in no error.

Syntax

Error(**ErrorRecord**)
Error(**ErrorTable**)

- *ErrorRecord* – Required. Error information record, including **Kind**, **Message**, and other fields. **Kind** is required. **FirstError** can be passed directly.
- *ErrorTable* – Required. Table of error information records. **AllErrors** can be passed directly.

IfError(*Value1*, *Replacement1* [, *Value2*, *Replacement2*, ... [, *DefaultResult*]])

- *Value(s)* – Required. Formula(s) to test for an error value.
- *Replacement(s)* – Required. The formulas to evaluate and values to return if matching *Value* arguments returned an error.
- *DefaultResult* – Optional. The formulas to evaluate if the formula doesn't find any errors.

IsError(*Value*)
IsBlankOrError(*Value*)

- *Value* – Required. Formula to test.

Examples

Simple IfError

Expand table

Formula	Description	Result
IfError(1, 2)	The first argument isn't an error. The function has no other errors to check and no default return value. The function returns the last *value* argument evaluated.	1
IfError(1/0, 2)	The first argument returns an error value (because of division by zero). The function evaluates the second argument and returns it as the result.	2
IfError(10, 20, 30)	The first argument isn't an error. The function has no other errors to check but does have a default return value. The function returns the *DefaultResult* argument.	30
IfError(10, 11, 20, 21, 300)	The first argument **10** isn't an error, so the function doesn't evaluate that argument's corresponding replacement **11**. The third argument	300

Formula	Description	Result
	20 isn't an error either, so the function doesn't evaluate that argument's corresponding replacement **21**. The fifth argument **300** has no corresponding replacement and is the default result. The function returns that result because the formula contains no errors.	
IfError(1/0, Notify("There was an internal problem"))	The first argument returns an error value (due to division by zero). The function evaluates the second argument and displays a message to the user. The return value of **IfError** is the return value of **Notify**, coerced to the same type as the first argument to **IfError** (a number).	1

Simple IsError

Expand table

Formula	Description	Result
IsError(1)	The argument isn't an error.	*false*
IsError(Blank())	The argument is a blank, but not an error.	*false*
IsError(1/0)	The argument is an error.	*true*
If(IsError(1/0), Notify("There was an internal problem"))	The argument to **IsError** returns an error value (because of division by zero). This function returns *true*, which causes the **If** to display a message to the user with the **Notify** function. The return value of **If** is the return value of **Notify**, coerced to the same type as the first argument to **If** (a boolean).	*true*

Simple IsBlankOrError

Expand table

Formula	Description	Result
IsBlankOrError(1)	The argument isn't an error or a blank.	*false*
IsBlankOrError(Blank())	The argument is a blank.	*true*
IsBlankOrError(1/0)	The argument is an error.	*true*

Simple Error

In this example, dates are validated relative to one another, resulting in an error if there is a problem.

Power Apps

```
If( StartDate > EndDate,
    Error( { Kind: ErrorKind.Validation, Message: "Start Date must be before
End Date" } ) )
```

In this example, some errors are allowed to pass through while others are suppressed and replaced with a value. In the first case, **b** is in an error state because the **Value** function has an invalid argument. Because this is unexpected by the formula writer, it is passed through so the user sees it. In the second case, with the same formula, **b** has the value 0, resulting in a division by zero. In this case, the formula writer may know that this is acceptable for this logic, suppress the error (no banner is shown), and return -1 instead.

Power Apps

```
With( {a: 1, b: Value("a")},
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns an error with Kind = ErrorKind.InvalidArgument

With( {a: 1, b: 0} )
      IfError( a/b, If( FirstError.Kind <> ErrorKind.Div0, Error( FirstError
), -1 ) ) )
// returns -1
```

The **AllErrors** table can be filtered like any other table. Used with the **Error** function, expected errors can be removed and the remaining errors retained and reported. For example, if we knew that division by zero wasn't going to be a problem in a particular context, those errors could be filtered out, leaving all other errors intact with the following formula:

Power Apps

```
Error( Filter( AllErrors, Kind <> ErrorKind.Div0 ) )
```

Step by step

1. Add a Text input control, and name it **TextInput1** if it doesn't have that name by default.

2. Add a Label control, and name it **Label1** if it doesn't have that name by default.

3. Set the formula for **Label1**'s **Text** property to:

Power Apps

```
IfError( Value( TextInput1.Text ), -1 )
```

4. In **TextInput1**, enter **1234**.

   Label1 shows the value **1234** as this is a valid input to the Value function.

5. In **TextInput1**, enter **ToInfinity**.

   Label1 shows the value **-1** as this isn't a valid input to the Value function. Without wrapping the Value function with IfError, the label would show no value as the error value is treated as a *blank*.

See also

Formula reference for Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

IsMatch, Match, and MatchAll functions

06/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests for a match or extracts parts of a text string based on a pattern.

Description

The **IsMatch** function tests whether a text string matches a pattern that can comprise ordinary characters, predefined patterns, or a regular expression. The **Match** and **MatchAll** functions return what was matched, including submatches.

Use **IsMatch** to validate what a user typed in a **Text input** control. For example, confirm whether the user entered a valid email address before the result is saved to your data source. If the entry doesn't match your criteria, add other controls that prompt the user to fix the entry.

Use **Match** to extract the first text string that matches a pattern and **MatchAll** to extract all text strings that match. Extract submatches to parse complex strings.

Match returns a record of information for the first match found, and **MatchAll** returns a table of records for every match found. The record or records contain:

Expand table

Column	Type	Description
named sub-match or sub-matches	Text	Each named submatch has its own column. Create a named submatch by using **(?<*name*>...)** in the regular expression. If a named submatch has the same name as one of the predefined columns, the submatch takes precedence, and a warning is generated. Rename the submatch to avoid this warning.
FullMatch	Text	All of the text string that was matched.
StartMatch	Number	The starting position of the match within the input text string. The first character of the string returns 1.
SubMatches, only if **MatchOptions.NumberedSubMatches** is used.	Single-column table of	The table of numbered submatches in the order in which they appear in the regular expression. Generally, named submatches are

Column	Type	Description
	Text (column **Value**)	easier to work with and are encouraged. Use the ForAll function or Index function to work with an individual sub-match. If no submatches are defined in the regular expression, this table will be present but empty.

These functions support MatchOptions. By default:

- These functions perform a case-sensitive match. Use **MatchOptions.IgnoreCase** to perform case-insensitive matches.
- **IsMatch** matches the entire text string (**Complete** MatchOption), while **Match** and **MatchAll** search for a match anywhere in the text string (**Contains** MatchOption). Use **Complete**, **Contains**, **BeginsWith**, or **EndsWith** as appropriate for your scenario.

IsMatch returns *true* if the text string matches the pattern or *false* if it doesn't. **Match** returns *blank* if no match is found that can be tested with the IsBlank function. **MatchAll** returns an empty table if no match is found that can be tested with the IsEmpty function.

If you use **MatchAll** to split a text string, consider using the Split function, which is simpler and faster.

Patterns

The key to using these functions is in describing the pattern to match. You describe the pattern in a text string as a combination of:

- Ordinary characters, such as "**abc**" or "**123**".
- Predefined patterns, such as **Letter**, **MultipleDigits**, or **Email**. (The **Match** enum defines these patterns.)
- Regular-expression codes, such as "**\d+\s+\d+**" or "**[a-z]+**".

Combine these elements using the string-concatenation operator &. For example, "**abc**" & **Digit** & "**\s+**" is a valid pattern that matches the characters "a", "b", and "c", followed by a digit from 0 to 9, followed by at least one whitespace character.

Ordinary characters

The simplest pattern is a sequence of ordinary characters that match exactly.

For example, when used with the **IsMatch** function, the string "Hello" matches the pattern "**Hello**" exactly. No more and no less. The string `"hello!"` doesn't match the pattern because

of the exclamation point on the end and because the case is wrong for the letter "h". (See Match options for ways to modify this behavior.)

In the pattern language, the characters `. ? * + ( ) [ ] ^ $ | \` are reserved for special purposes. To use these characters, either prefix the character with a **** (backslash) to indicate that the character should be taken literally, or use one of the predefined patterns. For example, you can match the string `"Hello?"` by using the pattern `"Hello\\?"` with a backslash before the question mark.

Predefined patterns

Predefined patterns provide a simple way to match either one of a set of characters or a sequence of multiple characters. Use the string-concatenation operator **&** to combine your own text strings with members of the **Match** enum:

Expand table

Match enum	Description	Regular expression
Any	Matches any character.	`.`
Comma	Matches a comma `,`.	`,`
Digit	Matches a single digit ("0" through "9").	`\d`
Email	Matches an email address that contains an "at" symbol ("@") and a domain name that contains a dot (".")	*See note*
Hyphen	Matches a hyphen.	`-` *See note*
LeftParen	Matches a left parenthesis `(`.	`\(`
Letter	Matches a letter.	`\p{L}`
MultipleDigits	Matches one or more digits.	`\d+`
MultipleLetters	Matches one or more letters.	`\p{L}+`
MultipleNonSpaces	Matches one or more characters that don't add whitespace (not space, tab, or newline).	`\S+`
MultipleSpaces	Matches one or more characters that add whitespace (space, tab, or newline).	`\s+`
NonSpace	Matches a single character that doesn't add whitespace.	`\S`
OptionalDigits	Matches zero, one, or more digits.	`\d*`

Match enum	Description	Regular expression
OptionalLetters	Matches zero, one, or more letters.	`\p{L}*`
OptionalNonSpaces	Matches zero, one, or more characters that don't add whitespace.	`\S*`
OptionalSpaces	Matches zero, one, or more characters that add whitespace.	`\s*`
Period	Matches a period or dot `.`.	`\.`
RightParen	Matches a right parenthesis `)`.	`\)`
Space	Matches a character that adds whitespace.	`\s`
Tab	Matches a tab character.	`\t`

For example, the pattern **"A" & MultipleDigits** matches the letter "A" followed by one or more digits.

Power Apps uses a different definition for **Match.EMail** and **Match.Hyphen**. Evaluate `Text( Match.Email )` to see the regular expression used by your host.

Regular expressions

The pattern they these functions use is called a regular expression. Power Fx's specific dialect of regular expressions is detailed in Regular expressions in Power Fx.

Regular expressions are powerful and serve a wide variety of purposes. They can also look like a random sequence of punctuation marks. This article doesn't describe all aspects of regular expressions, but a wealth of information, tutorials, and tools are available online.

Regular expressions have a long history and are available in many programming languages. Every programming language has its own dialect of regular expressions, and there are few standards. We strive to ensure the same regular expression gives the same result across all Power Fx implementations. Compatibility isn't easy to accomplish as Power Fx runs on top of JavaScript and .NET which have significant differences. To accommodate running on different platforms, Power Fx regular expressions are limited to a subset of features that are widely supported across the industry.

As a result, some regular expressions that may work in other environments may be blocked or require a tweak in Power Fx. Authoring time errors are reported as unsupported features are encountered. This is one of the reasons that the regular expression and options must be an authoring time constant and not dynamic (for example, provided in a variable).

> **Note**
>
> Power Apps uses an earlier version of Power Fx regular expressions that has fewer limitations but also fewer features. **MatchOptions.DotAll** and **MatchOptions.FreeSpacing** aren't available and the definitions of **Match.Email** and **Match.Hyphen** are different. Unicode surrogate pairs aren't treated as a single character. **MatchOptions.NumberedSubMatches** is the default. The version of regular expressions described here will be available in Power Apps soon, under a "Power Fx V1.0 compatibility" switch.

Here are some basic elements of regular expressions that build up complexity in parsing a number.

Expand table

Feature	Example	Description
Predfined character class	`\d`	This regular expression matches a single number, such as `1`. A character class matches a set of characters and `\d` matches the standard digits `0` to `9` and also digits defined in the Unicode character category "Nd". There are character classes for letters and numbers with `\w` and spaces including newlines with `\s`. There are also inverse character classes that are capitalized: `\D` matches everything that `\d` does not.
One or More	`\d+`	This regular expression matches one or more numbers, such as `123`. A `+` after an element says "one or more" of the last element.
Zero or One	`\+?\d`	This regular expression matches an optional `+` sign followed by one ore more numbers, such as `+123` as well as just `123`. A `?` after an element says "this is optional, it can occur zero or one time". The `+` has a backslash before it to distinguish it as a literal character rather than the "one or more" usage.
Grouping and alternation	`(-|\+)?\d+`	This regular expression matches either an `+` sign or a `-`, optionally, and then a series of numbers, such as `-123`, `+123`, and `123`. We are introducing two concepts here that are often used together. First, we have the parenthesis that group as set of elements together, for the `?` to act upon. Second, we have `|` which says "either this or that".
Custom character class	`(-|\+)?\d+[eE][\-\+]?\d+`	This regular expression adds an exponent to the mix with two character classes, matching `+123e-12`. A character class is like `|` alternation providing a "match one of these things" in a more

Feature	Example	Description
		compact form. As with the `+`, `-` has special meaning in regular expression character classes so we need to escape it.
Zero or More	`(-|\+)?\d+\.?\d*[eE][\-\+]?\d+`	This regular expression adds the decimal after the integer portion of the number, such as `-123.456e-89` or `-123.E+32`. The `\d` after the decimal point has a `*` quantifier that says "zero or more times" for decimal digits after the `.`
Capture groups	`(?<number>(-|\+)?\d+\.?\d*)[eE](?<exponent>[\-\+]?\d+)`	Finally, we add capture groups for `number` and `exponent`. Not only can a regular expression match the entire string, it can also extract portions for use in your formulas, in this case the part before the `e` (or `E`) and the part after.

These examples only give a small taste of what regular expressions can do. They are commonly used to validate ID numbers, email addresses, phone numbers, dates and times, and to extract information from all kinds of text files. Continue your journey by reading Regular expressions in Power Fx, experiment, and use the web to learn more.

Match options

Modify the behavior of these functions by specifying one or more options, which you combine using the string- concatenation operator (**&**).

Expand table

MatchOptions enum	Description	Impact on a regular expression
MatchOptions.BeginsWith	The pattern must match from the beginning of the text.	Adds a ^ to the start of the regular expression.
MatchOptions.Complete	Default for **IsMatch** on Power Apps. The pattern matches the entire string of text, from beginning to end.	Adds a ^ to the start and a $ to the end of the regular expression.
MatchOptions.Contains	Default for **Match** and **MatchAll**, and **IsMatch** outside of Power Apps. The pattern must appear somewhere in the text but doesn't need to begin or end it.	Doesn't modify the regular expression.
MatchOptions.DotAll	Changes the behavior of the `.` (dot) operator to match all characters,	Doesn't modify the regular expression.

MatchOptions enum	Description	Impact on a regular expression
	including newline characters. Not available in Power Apps.	This option is the equivalent of the standard "s" modifier for regular expressions.
MatchOptions.EndsWith	The pattern must match the end of the string of text.	Adds a **$** to the end of the regular expression.
MatchOptions.FreeSpacing	Whitespace characters, including newlines, are ignored in the regular expression. End-of-line comments beginning with a `#` are ignored. Not available in Power Apps.	Only changes how the regular expression syntax. This option is the equivalent of the standard "x" modifier for regular expressions.
MatchOptions.IgnoreCase	Treats uppercase and lowercase letters as identical. By default, matching is case sensitive.	Doesn't modify the regular expression. This option is the equivalent of the standard "i" modifier for regular expressions.
MatchOptions.Multiline	Changes the behavior of `^` and `$` to match at the end of aline.	Doesn't modify the regular expression. This option is the equivalent of the standard "m" modifier for regular expressions.
MatchOptions.NumberedSubMatches	Named captures are preferred because they are easier to understand and maintain. Performance also improves as unneeded captures aren't retained. But for older regular expressions, treats each set of parenthesis as a numbered capture that is included with the **SubMatches** table in the result. Default in Power Apps.	Doesn't modify the regular expression. Named captures are disabled and `\1` style back references are enabled.

Using **MatchAll** is the same as using the standard "g" modifier for regular expressions.

Syntax

IsMatch(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to test.
- *Pattern* – Required. The pattern to test as a text string. Concatenate predefined patterns that the **Match** enum defines or provide a regular expression. *Pattern* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs. Note, that the formula must be expressed as "Match.PredefinedPattern" e.g. Match.Email
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Complete** is used. *Options* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.

Match(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to match.
- *Pattern* – Required. The pattern to match as a text string. Concatenate predefined patterns that the **Match** enum defines, or provide a regular expression. *Pattern* must be a constant formula without any variables, data sources, or other dynamic references that change as the app runs.
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Contains** is used. *Options* must be a constant formula without any variables, data sources, or other dynamic references that change as the app runs.

MatchAll(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to match.
- *Pattern* – Required. The pattern to match as a text string. Concatenate predefined patterns that the **Match** enum defines or provide a regular expression. *Pattern* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Contains** is used. *Options* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.

IsMatch examples

Ordinary characters

Imagine your app has a **Text input** control named **TextInput1**. Users enter values into this control to store them in a database.

Users type **Hello world** into **TextInput1**.

Expand table

Formula	Description	Result
`IsMatch( TextInput1.Text, "Hello world" )`	Tests whether the user's input matches, exactly, the string "Hello world".	**true**
`IsMatch( TextInput1.Text, "Good bye" )`	Tests whether the user's input matches, exactly, the string "Good bye".	**false**
`IsMatch( TextInput1.Text, "hello", Contains )`	Tests whether the user's input contains the word "hello" (case sensitive).	**false**
`IsMatch( TextInput1.Text, "hello", Contains & IgnoreCase )`	Tests whether the user's input contains the word "hello" (case insensitive).	**true**

Predefined patterns

Expand table

Formula	Description	Result
`IsMatch( "123-45-7890", Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit )`	Matches a United States Social Security Number	**true**
`IsMatch( "joan@contoso.com", Match.Email )`	Matches an email address	**true**
`IsMatch( "123.456", Match.MultipleDigits & Match.Period & Match.OptionalDigits )`	Matches a sequence of digits, a period, and then zero or more digits.	**true**
`IsMatch( "123", Match.MultipleDigits & Match.Period & Match.OptionalDigits )`	Matches a sequence of digits, a period, and then zero or more digits. A period doesn't appear in the text to match, so this pattern isn't matched.	**false**

Regular expressions

Expand table

Formula	Description	Result
`IsMatch( "986", "\d+" )`	Matches an integer greater than zero.	**true**

Formula	Description	Result
`IsMatch( "1.02", "\d+(\.\d\d)?" )`	Matches a positive currency amount. If the input contains a decimal point, the input must also contain two numeric characters after the decimal point. For example, 3.00 is valid, but 3.1 isn't.	**true**
`IsMatch( "-4.95", "(-)?\d+(\.\d\d)?" )`	Matches a positive or negative currency amount. If the input contains a decimal point, the input must also contain two numeric characters after the decimal point.	**true**
`IsMatch( "111-11-1111", "\d{3}-\d{2}-\d{4}" )`	Matches a United States Social Security number. Validates the format, type, and length of the supplied input field. The string to match must consist of three numeric characters followed by a dash, then two numeric characters followed by a dash, and then four numeric characters.	**true**
`IsMatch( "111-111-111", "\d{3}-\d{2}-\d{4}" )`	Same as the previous example, but one of the hyphens is out of place in the input.	**false**
`IsMatch( "AStrongPasswordNot", "(?!^[0-9]\*$)(?!^[a-zA-Z]\*$)([a-zA-Z0-9]{8,10})" )`	Validates a strong password that must contain eight, nine, or 10 characters, at least one digit, and at least one alphabetic character. The string can't contain special characters.	**false**

Match and MatchAll examples

Expand table

Formula	Description	Result

`Match( "Bob Jones <bob.jones@contoso.com>", "<(?<email>" & Match.Email & ")>")`	Extracts only the email portion of the contact information.	{
		email: "bob.jones@contoso.com",
		FullMatch: "<bob.jones@contoso.com>",
		StartMatch: 11
		}

`Match( "Bob Jones <InvalidEmailAddress>", "<(?<email>" & Match.Email & ")>"`	Extracts only the email portion of the contact information. No legal address is found (there is no @ sign), so the function returns *blank*.	*blank*
`Match( Language(), "(<language>\w{2})(?:-(?<script>\w{4}))?(?:-(?<region>\w{2}))?" )`	Extracts the language, script, and region portions of the language tag that the Language function returns. These results reflect the	{ language: "en", script: *blank*, region: "US",

Formula	Description	Result
	United States; see the Language function documentation for more examples. The **(?:** operator groups characters without creating another sub-match.	FullMatch: "en-US", StartMatch: 1 }

`Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?<seconds>\d+)S)?" )`	Extracts the hours, minutes, and seconds from an ISO 8601 duration value. The extracted numbers are still in a text string; use the Value function to convert it to a number before mathematical operations are performed on it.	{
		hours: "2",
		minutes: "1",
		seconds: "39",
		FullMatch: "PT2H1M39S",
		StartMatch: 1
		}

Let's drill into that last example. If you wanted to convert this string to a date/time value using the Time function, you must pass in the named submatches individually. To do this, use the With function operating on the record that **Match** returns:

Power Fx

```
With(
    Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?<seconds>\d+)S)?" ),
   Time( Value( hours ), Value( minutes ), Value( seconds ) )
)
```

For these examples, add a Button control, set its **OnSelect** property to this formula, and then select the button:

Power Fx

```
Set( pangram, "The quick brown fox jumps over the lazy dog." )
```

Expand table

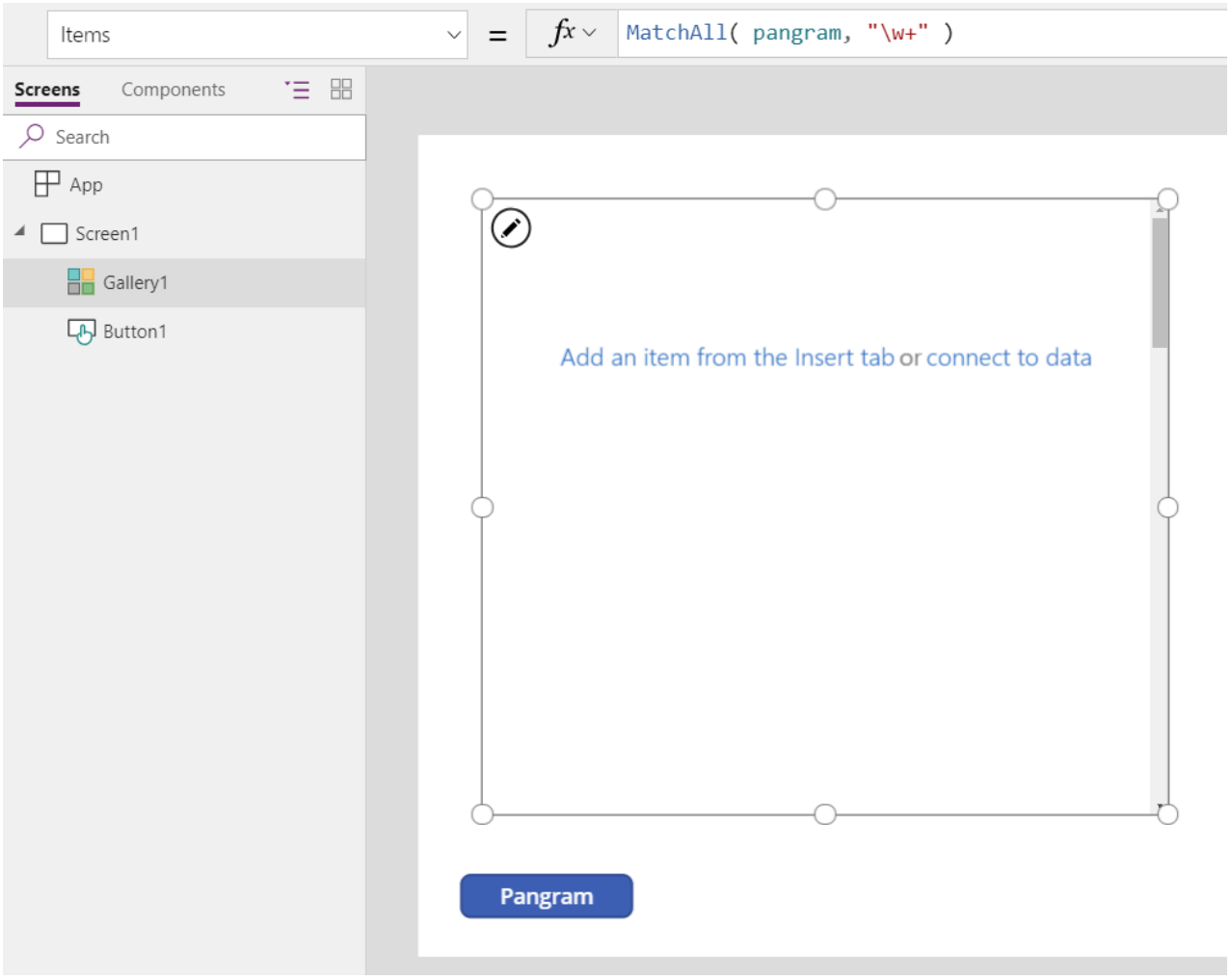
Formula	Description	Result

`Match( pangram, "THE", IgnoreCase )`	Find all matches of "THE" in the text string that the **pangram** variable contains. The string contains two matches, but only the first is returned because	{
		FullMatch: "The",
		StartMatch: 32
		}

<table>
<tr><th>Formula</th><th>Description</th><th>Result</th></tr>
<tr><td></td><td>you're using Match and not MatchAll.</td><td></td></tr>
<tr><td><code>MatchAll(pangram, "the")</code></td><td>Find all matches of "the" in the text string that the pangram variable contains. The test is case sensitive, so only the second instance of "the" is found.</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"the"</td><td>[ ]</td><td>32</td></tr></table></td></tr>
<tr><td><code>MatchAll(pangram, "the", IgnoreCase)</code></td><td>Find all matches of "the" in the text string that the pangram variable contains. In this case, the test is case insensitive, so both instances of the word are found.</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"The"</td><td>[ ]</td><td>1</td></tr><tr><td>"the"</td><td>[ ]</td><td>32</td></tr></table></td></tr>
<tr><td><code>MatchAll(pangram, "\b\wo\w\b")</code></td><td>Finds all three-letter words with an "o" in the middle. "brown" is excluded because it's not a three-letter word and, therefore, fails to match "\b" (word boundary).</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"fox"</td><td>[ ]</td><td>17</td></tr><tr><td>"dog"</td><td>[ ]</td><td>41</td></tr></table></td></tr>
<tr><td><code>Match(pangram, "\b\wo\w\b\s*(?<between>\w.+\w)\s*\b\wo\w\b")</code></td><td>Matches all the characters between "fox" and "dog".</td><td>{
between: "jumps over the lazy",
FullMatch: "fox jumps over the lazy dog",
StartMatch: 17
}</td></tr>
</table>

To see the results of **MatchAll** in a gallery:

1. In an empty screen, insert a blank vertical gallery control.
2. Set the gallery's **Items** property to **MatchAll(pangram, "\w+")** or **MatchAll(pangram, MultipleLetters)**.

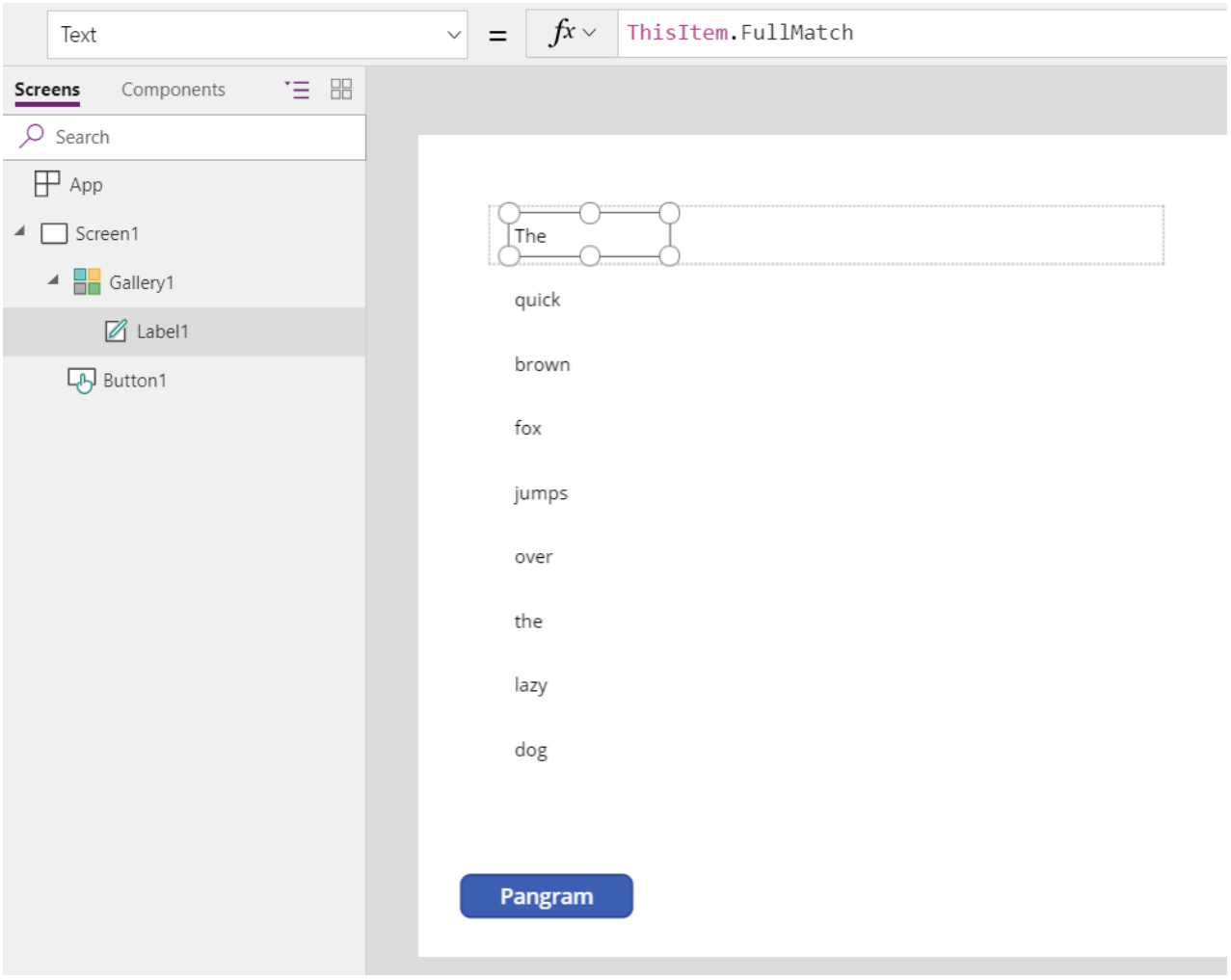


3. Select "Add an item from the Insert tab" in the middle of the gallery control to select the template of the gallery.

4. Add a Label control to the gallery's template.

5. Set the label's **Text** property to **ThisItem.FullMatch**.

   The gallery is filled with each word in our example text. To see all the words on one screen, resize the gallery's template and the label control.


Text
=
ThisItem.FullMatch
Screens
Components
Search
App
Screen1
Gallery1
Label1
Button1
The
quick
brown
fox
jumps
over
the
lazy
dog
Pangram

IsNumeric function

06/21/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests whether a value is numeric.

Description

The **IsNumeric** function tests whether a value is numeric. Other kinds of values include Boolean, string, table, and record.

The return value is a Boolean **true** or **false**.

Syntax

IsNumeric(*Value*)

Examples

Expand table

Formula	Description	Result
IsNumeric(1)	Evaluates if the number 1 is a number	*true*
IsNumeric("1")	Evaluates if the text "1" is a number	*true*
IsNumeric("A")	Evaluates if the text A is a number	*false*
IsNumeric(1+1)	Evaluates if the outcome of 1+1 is a number	*true*
IsNumeric("1+1")	Evaluates if the text 1+1 is a number	*false*

WeekNum and ISOWeekNum functions

Article • 04/20/2025

Expand table

Function	Applies to
ISOWeekNum	✅ Canvas apps ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Pages
WeekNum	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the week number of a specific date.

Description

Use the **WeekNum** and **ISOWeekNum** functions to determine the week number of a date.

These functions differ in how they determine the first week of the year (week 1):

- **WeekNum** uses the week containing January 1 as the first week of the year. The result from this function can range from 1 to 54.
- **ISOWeekNum** uses the week containing the first Thursday of the year as the first week of the year. This follows the ISO 8601 date and time standard definition for week numbering. The result from this function can range from 1 to 53. It is possible that 52 or 53 may be returned for the first days of January since the dates could belong to the last week of the previous year.

Use the second parameter to **WeekNum** to specify which day begins a week. You can provide either an Excel code number or use the StartOfWeek enumeration:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	**StartOfWeek.Sunday**	Week begins on Sunday. Default.
2, 11	**StartOfWeek.Monday**	Week begins on Monday.
12	**StartOfWeek.Tuesday**	Week begins on Tuesday.
13	**StartOfWeek.Wednesday**	Week begins on Wednesday.

Excel code	StartOfWeek enumeration	Description
14	**StartOfWeek.Thursday**	Week begins on Thursday.
15	**StartOfWeek.Friday**	Week begins on Friday.
16	**StartOfWeek.Saturday**	Week begins on Saturday.

ISOWeekNum always uses Monday as the start of the week. In Excel, the **WeekNum** function supports an addition code **21** that is not supported here; use **ISOWeekNum** instead.

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

WeekNum(*DateTime* [, *StartOfWeek*])

- *DateTime* - Required. Date/Time value to operate on.
- *StartOfWeek* - Optional. Excel code or StartOfWeek enumeration that determines which day the week begins.

ISOWeekNum(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on. The week always begins on Monday.

Examples

First and last calendar weeks of 2021

Expand table

Date	WeekNum(Date)	ISOWeekNum(Date)	WeekNum(Date, StartOfWeek.Wednesday)
Friday, January 1, 2021	1	53	1
Saturday, January 2, 2021	1	53	1
Sunday, January 3, 2021	2	53	1
Monday, January 4, 2021	2	1	1
Tuesday, January 5, 2021	2	1	1

Date	WeekNum(Date)	ISOWeekNum(Date)	WeekNum(Date, StartOfWeek.Wednesday)
Wednesday, January 6, 2021	2	1	2
Thursday, January 7, 2021	2	1	2
Saturday, December 25, 2021	52	51	52
Sunday, December 26, 2021	53	51	52
Monday, December 27, 2021	53	52	52
Tuesday, December 28, 2021	53	52	52
Wednesday, December 29, 2021	53	52	53
Thursday, December 30, 2021	53	52	53
Friday, December 31, 2021	53	52	53

Now, Today, IsToday, UTCNow, UTCToday, IsUTCToday functions

Article • 04/20/2025

Expand table

Functions	Applies to
Now	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Today **IsToday**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
UTCNow **UTCToday**	✅ Dataverse formula columns ✅ Power Pages

Returns the current date and time, and tests whether a date/time value is today.

> ⓘ Note
>
> PAC CLI pac power-fx commands do not support the **UTCNow**, **UTCToday**, and **IsUTCToday** functions.

Description

The **Now** function returns the current date and time as a date/time value.

The **Today** function returns the current date as a date/time value. The time portion is midnight. **Today** has the same value throughout a day, from midnight today to midnight tomorrow.

The **IsToday** function tests whether a date/time value is between midnight today and midnight tomorrow. This function returns a Boolean (**true** or **false**) value.

Now, **Today**, and **IsToday** functions work with the local time of the current user.

UTCNow, **UTCToday**, and **IsUTCToday** functions are the same as their non-UTC counterparts but work with time zone independent values and use Coordinated Universal Time (UTC).

> ⓘ Note

- **UTCNow**, **UTCToday**, and **IsUTCToday** are only available in Microsoft Dataverse for Teams formula columns, and only for time-independent fields and values.
- **Now**, **Today**, and **IsToday** are not available in Dataverse for Teams formula columns as evaluations are done without the knowledge of the current user's local time zone. More information: Work with formula table columns in Dataverse for Teams

See Date, Time, and DateTime in the data types documentation and working with dates and times for more information.

Volatile Functions

Now, **Today**, **UTCNow**, and **UTCToday** are volatile functions. These functions return a different value for each evaluation.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Now()** will not change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it is part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = DateAdd(Now(), Slider1.Value, Minutes)** then the current time is retrieved each time the Slider control's value changes and the label's text property is reevaluated.

When used in a behavior formula, volatile functions will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Using the user's local time

Now()

Today()

IsToday(*DateTime*)

- *DateTime* - Required. The date/time value to test.

Using Coordinated Universal Time (UTC)

UTCNow()

UTCToday()

IsUTCToday(*TimeZoneIndependentTime*)

- *TimeZoneIndependentDateTime* - Required. The time zone independent date/time value to test.

Examples

For the examples in this section, the current time is **8:58 PM** on **July 11, 2021** in the Pacific Time Zone (UTC-8) and the language is **en-us**.

Expand table

Formula	Description	Result
Text(Now(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time in the user's time zone, and displays it as a string.	"07/11/2021 20:58:00"
Text(Today(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date only, leaving the time portion as midnight, and displays it as a string.	"07/12/2021 00:00:00"
IsToday(Now())	Tests whether the current date and time is between midnight today and midnight tomorrow.	**true**
IsToday(Today())	Tests whether the current date is between midnight today and midnight tomorrow.	**true**
Text(DateAdd(Now(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time, adds 12 days to the result, and displays it as a string.	"07/23/2021 20:58:00"
Text(DateAdd(Today(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date, adds 12 days to the result, and displays it as a string.	"07/23/2021 00:00:00"
IsToday(DateAdd(Now(), 12))	Tests whether the current date and time, plus 12 days, is between midnight today and midnight tomorrow.	**false**
IsToday(DateAdd(Today(), 12))	Tests whether the current date, plus 12 days, is between midnight today and midnight tomorrow.	**false**
Hour(UTCNow())	Retrieves the current date and time in UTC and extracts the hour only, which is 8 hours ahead of local time.	4

Formula	Description	Result
Day(UTCToday())	Retrieves the current date only in UTC and extracts the day, which is 1 day ahead of local time.	12
IsUTCToday(UTCNow())	Tests whether the current date and time is between midnight today and midnight tomorrow, all in UTC time.	**true**
IsUTCToday(UTCToday())	Tests whether the current date is between midnight today and midnight tomorrow, all in UTC time.	**true**

Display a clock that updates in real time

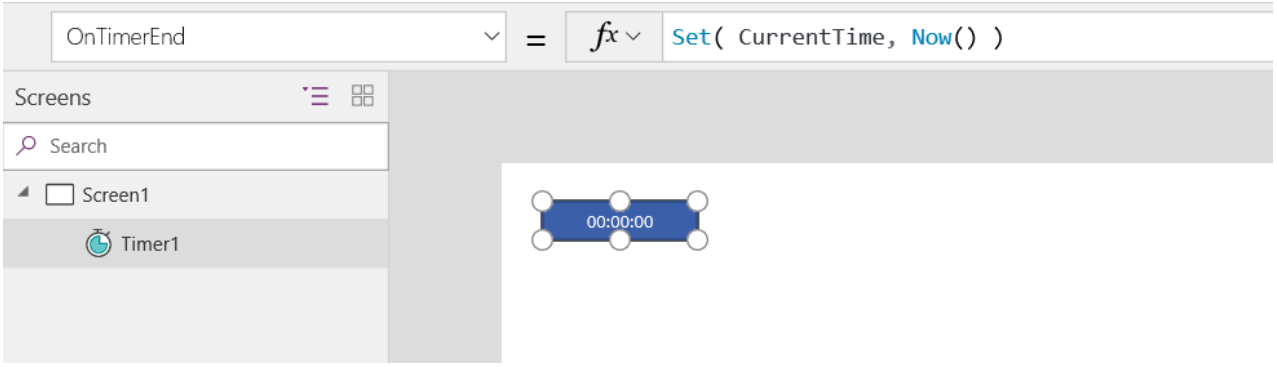
1. Add a Timer control, set its **Duration** property to **1000**, and set its **Repeat** property to **true**.

   The timer will run for one second, automatically start over, and continue that pattern.

2. Set the control's **OnTimerEnd** property to this formula:

   Set(CurrentTime, Now())

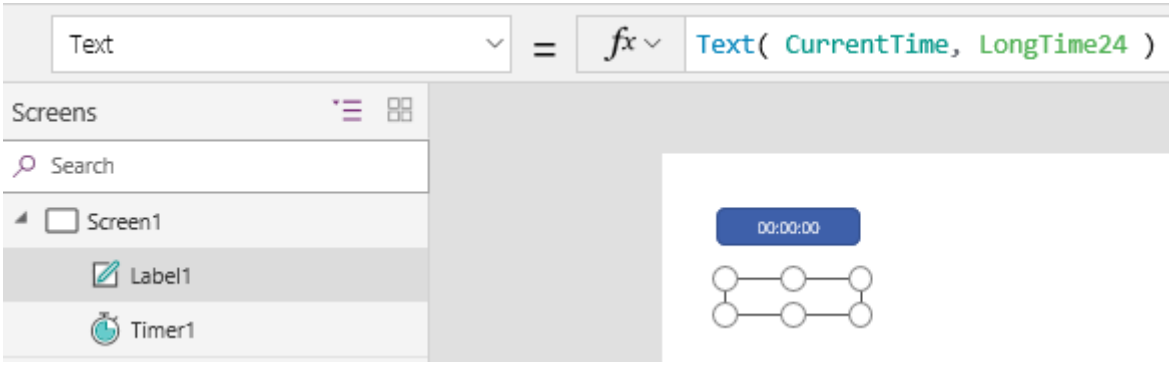
   Whenever the timer starts over (after each second), this formula sets the **CurrentTime** global variable to the current value of the **Now** function.

3. Add a Label control, and set its **Text** property to this formula:

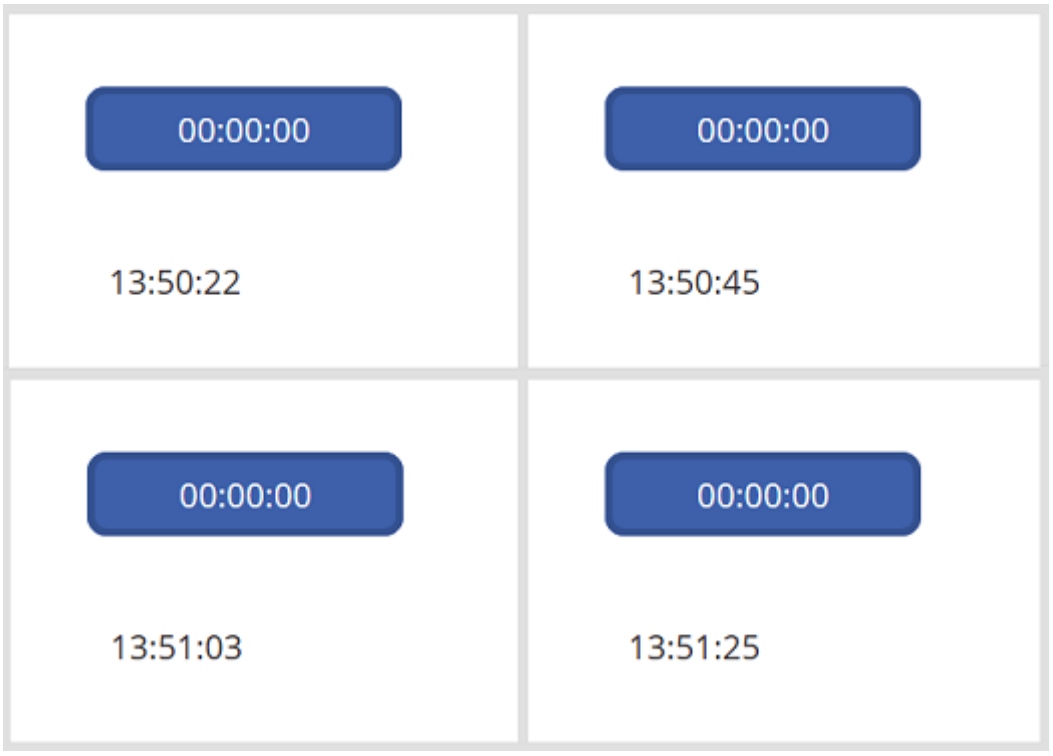
   Text(CurrentTime, LongTime24)

   Use the Text function to format the date and time however you want, or set this property to just **CurrentTime** to show hours and minutes but not seconds.



4. Preview the app by pressing F5, and then start the timer by clicking or tapping it.

   The label continually shows the current time, down to the second.



5. Set the timer's **AutoStart** property to **true** and its **Visible** property to **false**.

   The timer is invisible and starts automatically.

6. Set the screen's OnStart property so that the **CurrentTime** variable has a valid value, as in this example:

   Set(CurrentTime, Now())

   The label appears as soon as the app starts (before the timer runs for a full second).

AsType and IsType functions

06/19/2025

Expand table

Function	Applies to
AsType	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
IsType	✅ Canvas apps ✅ Copilot Studio ✅ Model-driven apps ✅ Dataverse functions ✅ Power Pages

Checks if a record reference is a specific table type (**IsType**) and treats the reference as that type (**AsType**).

> ⓘ **Note**
>
> PAC CLI pac power-fx commands don't support **IsType**.

Description

The **AsType** and **IsType** functions can be used to convert record references (for example polymorphic lookups in Dataverse) and dynamic values to typed values that can be used directly.

Record References

Read Understand record references and polymorphic lookups for a broader introduction and more details.

A lookup field usually refers to records in a particular table. Because the table type is well established, access the fields of the lookup using simple dot notation. For example, **First(Accounts).'Primary Contact'.'Full Name'** walks from the **Accounts** table to the **Primary Contact** record in the **Contacts** table and extracts the **Full Name** field.

Microsoft Dataverse also supports polymorphic lookup fields, which can refer to records from a set of tables, as in these examples.

Expand table

Lookup field	Can refer to
Owner	**Users** or **Teams**
Customer	**Accounts** or **Contacts**
Regarding	**Accounts**, **Contacts**, **Knowledge Articles**, etc.

In canvas-app formulas, use record references to work with polymorphic lookups. Because a record reference can refer to different tables, you can't know which fields are available at runtime when you write a formula. The *Record.Field* notation isn't available. These formulas need to adapt to the records that the app encounters when it runs.

The **IsType** function checks if a record reference refers to a specific table type. The function returns a Boolean TRUE or FALSE.

The **AsType** function treats a record reference as a specific table type, also called *casting*. You use the result as if it's a record of the table and use the *Record.Field* notation to access all the fields of that record. If the reference isn't of the specific type, an error occurs.

Use these functions together to first check the table type of a record and then treat it as a record of that type so the fields are available:

Power Fx

```
If( IsType( First( Accounts ).Owner, Users ),
    AsType( First( Accounts ).Owner, Users ).'Full Name',
    AsType( First( Accounts ).Owner, Teams ).'Team Name'
)
```

You need these functions only if you access the fields of a record reference. For example, you use record references in the **Filter** function without **IsType** or **AsType**:

Power Fx

```
Filter( Accounts, Owner = First( Users ) )
```

Similarly, you can use record references with the **Patch** function:

Power Fx

```
Patch( Accounts, First( Accounts ), { Owner: First( Teams ) } )
```

When you use these functions in a record context, such as within a **Gallery** or **Edit form** control, you might need the global disambiguation operator to reference the table type. For

example, this formula works for a gallery that displays a list of contacts where **Company Name** is a **Customer** lookup:

```
Power Fx
```

```
If( IsType( ThisItem.'Company Name', Accounts ),
    AsType( ThisItem.'Company Name', Accounts ).'Account Name',
    AsType( ThisItem.'Company Name', Contacts ).'Full Name'
)
```

For both functions, specify the type by using the name of the data source connected to the table. For the formula to work, add a data source to the app for any types you want to check or cast. For example, add the **Users** table as a data source if you want to use **IsType** and **AsType** with an **Owner** lookup and records from that table. Add only the data sources you use in your app; you don't need to add all the tables that a lookup could reference.

If the record reference is *blank*, **IsType** returns FALSE, and **AsType** returns *blank*. All fields of a *blank* record are also *blank*.

Dynamic values

> **Important**
>
> - Using **AsType** and **IsType** with **Dynamic** values is an experimental feature.
> - Experimental features aren't meant for production use and might not be complete. These features are available before an official release so that you can get early access and provide feedback. More information: Understand experimental, preview, and retired features in canvas apps
> - The behavior that this article describes is available only when the **User-defined types** experimental feature in Settings > Upcoming features > Experimental is turned on (it's off by default).
> - Your feedback is valuable to us. Let us know what you think in the Power Apps experimental features community forum.

A **Dynamic** value from a web API or the [**ParseJSON** function] needs to be converted to a specific typed value before you can use it in Power Fx. Here are some options:

1. Implicitly type the field at the point you use it. For example, an object converts to a number if you use it with the `+` operator, if it can be converted to a number. This option can cause unexpected conversions and can't convert records and tables as a whole.

2. Explicitly type each field individually with the **Decimal**, **Text**, **DateTime**, **GUID**, and other type constructor functions. This option is the most invasive to your formulas because you need to do each field separately.
3. Explicitly type JSON with the second argument to the **ParseJSON** function. This option is easy and avoids needing the **Dynamic** value.
4. Explicitly type a **Dynamic** value using the **AsType** function. You can also check the type before you try the conversion with the **IsType** function.

Syntax

AsType(*RecordReference*, *TableType*)

- *RecordReference* - Required. A record reference, often a lookup field that refers to a record in any of several tables.
- *TableType* - Required. The specific table to cast the record to.

AsType(*DynamicValue*, *TypeSpecification*)

- *DynamicValue* - Required. A dynamic value from the **ParseJSON** function or API call.
- *TypeSpecification* - Required. A type name or type specification you define with the **Type** function.

IsType(*RecordReference*, *TableType*)

- *RecordReference* - Required. A record reference, often a lookup field that can refer to a record in any of multiple tables.
- *TableType* - Required. The specific table to test for.

IsType(*DynamicValue*, *TypeSpecification*)

- *DynamicValue* - Required. A dynamic value from the **ParseJSON** function or an API call.
- *TypeSpecification* - Required. A type name or type specification defined with the **Type** function.

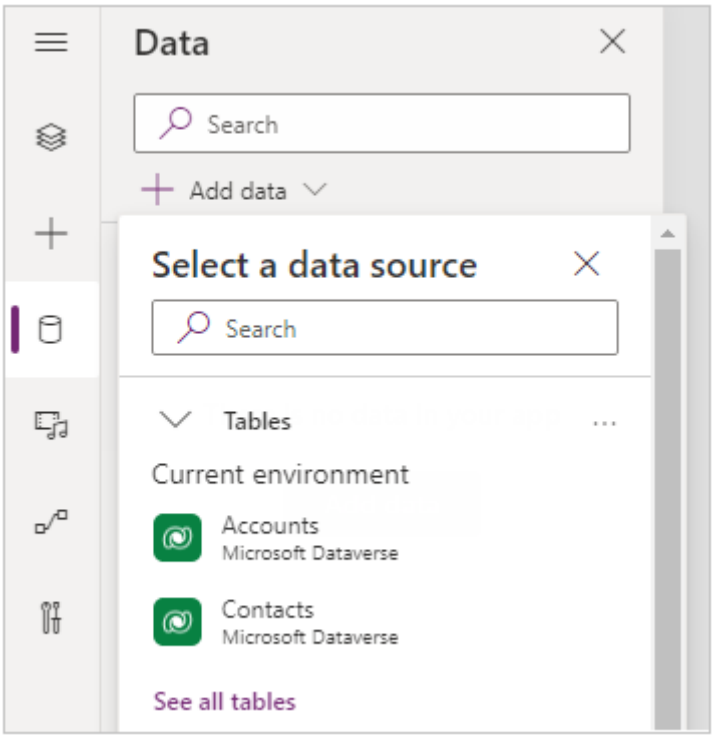
Examples

Record References

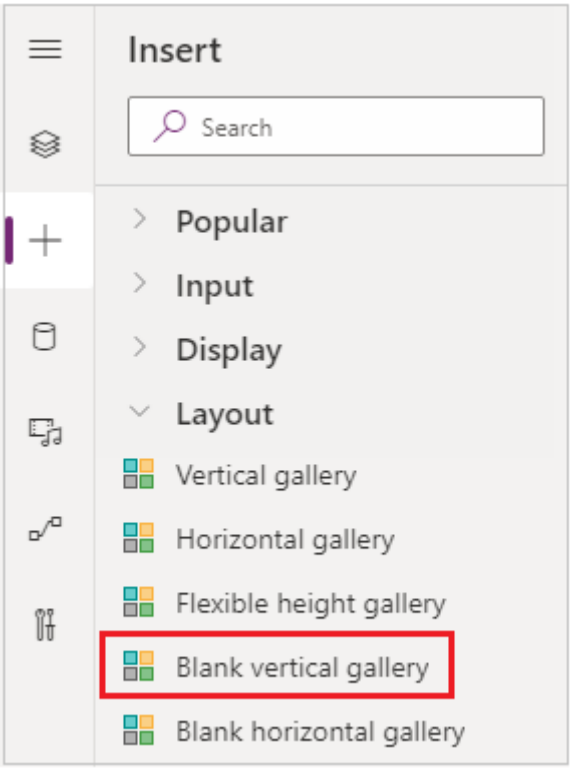
Understand record references and polymorphic lookups contains extensive examples.

1. Create a blank canvas app for tablets.

2. On the left pane, select **Data** > **Add data**, and then add **Accounts** and **Contacts** tables.

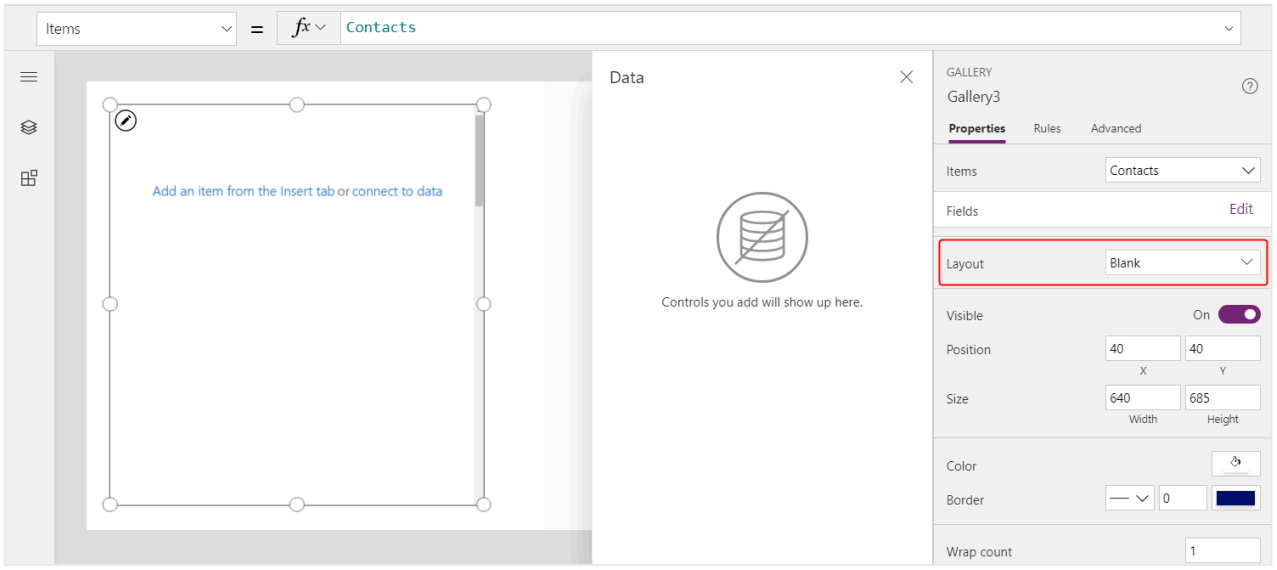


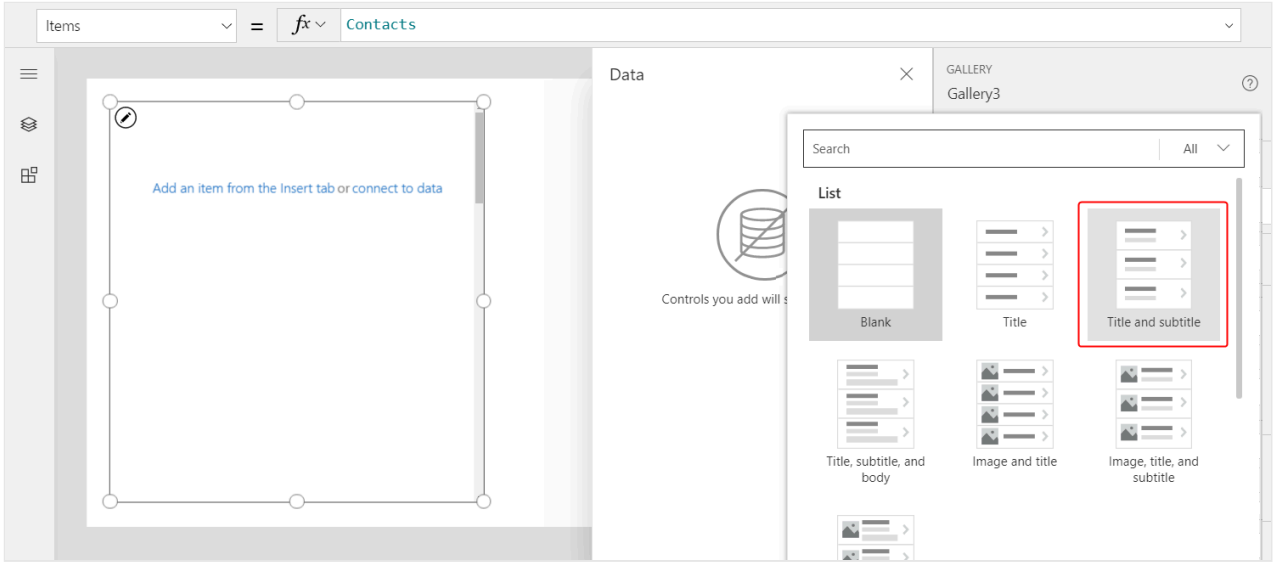
3. On the left pane, select + (Insert) > **Layout** > **Blank vertical gallery**.



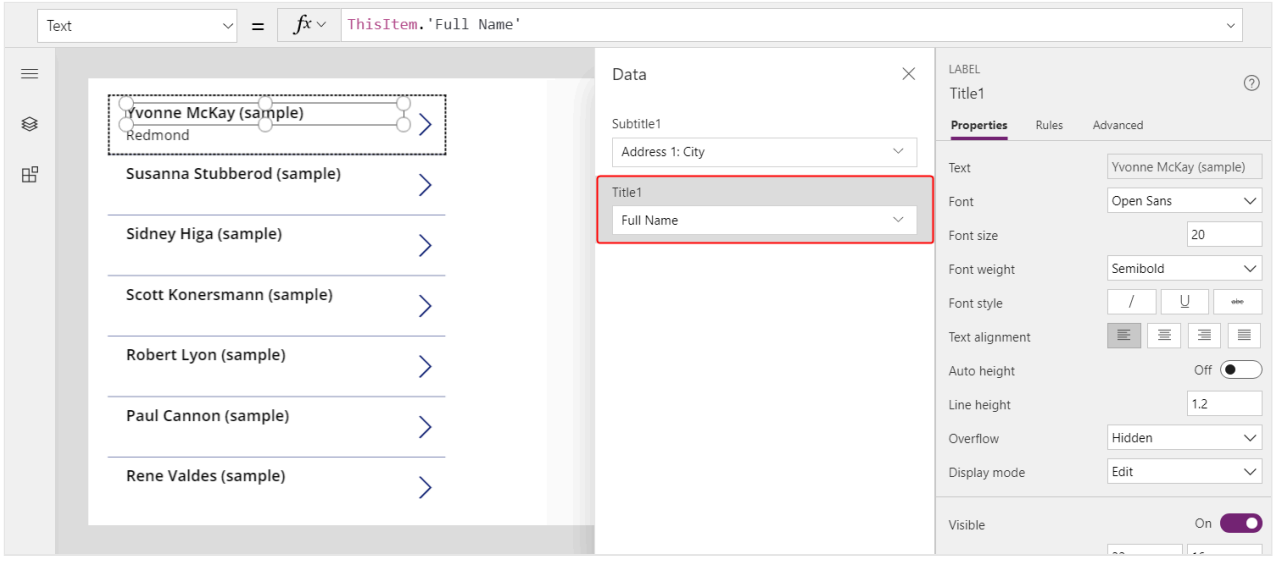
4. Select **Connect to data**, and then select **Contacts** as the data source.

5. Set the gallery layout to **Title and subtitle**.

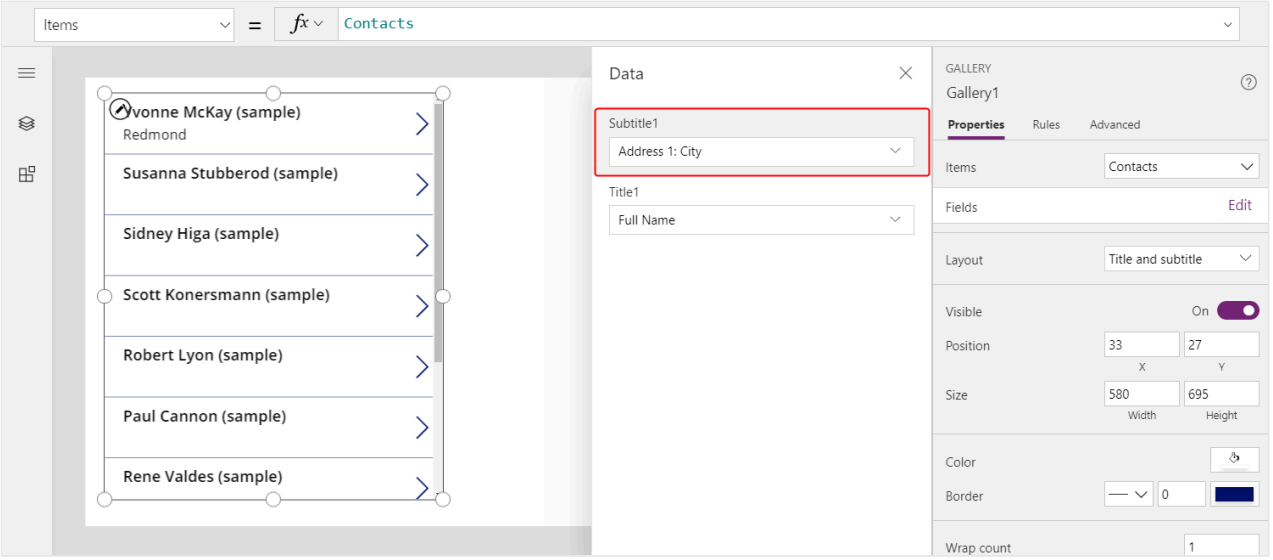




6. In the **Data** pane, open the **Title1** list, and then select **Full Name**.



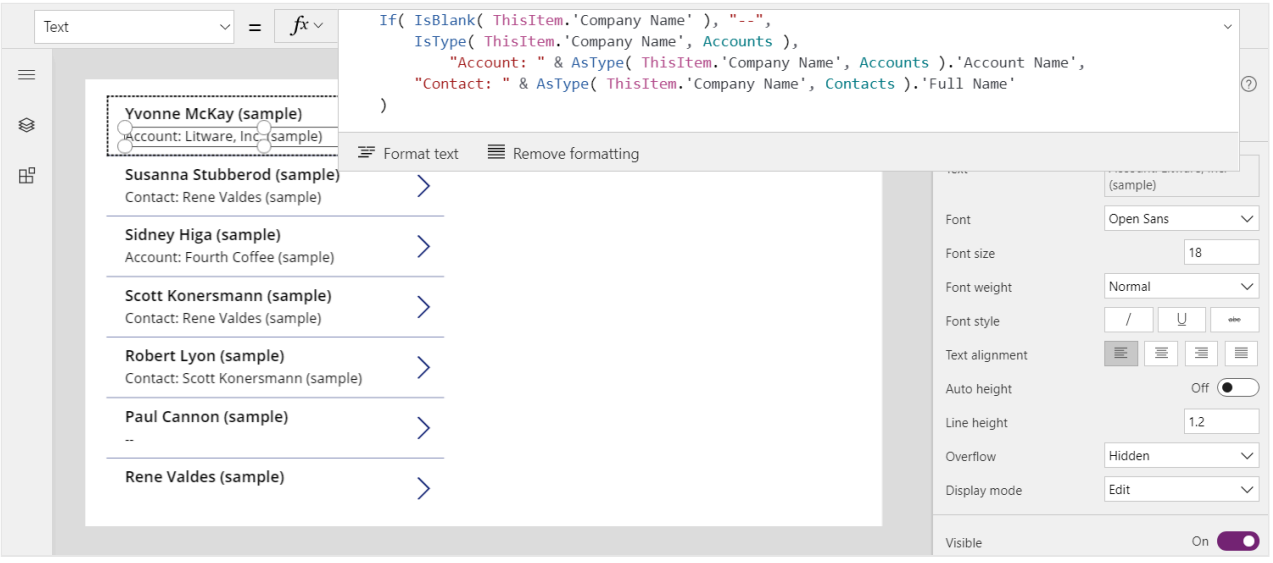
7. Select the **Subtitle1** label control.



8. Set the **Text** property of **Subtitle1** to this formula:

Power Fx

```
If( IsBlank( ThisItem.'Company Name' ), "--",
    IsType( ThisItem.'Company Name', Accounts ),
        "Account: " & AsType( ThisItem.'Company Name', Accounts ).'Account
Name',
    "Contact: " & AsType( ThisItem.'Company Name', Contacts ).'Full Name'
)
```



The subtitle in the gallery shows these values:

- "--" if the '**Company Name**' is *blank*.
- "Account: " and the **Account Name** field from the **Accounts** table if the **Company Name** field refers to an account.
- "Contact: " and the **Full Name** field from the **Contacts** table if the **Company Name** field refers to a contact.

Your results can differ because the sample data can be modified to show more types of results.

Dynamic values

The following examples show a simple JSON record interpreted in various ways by **ParseJSON**, **AsType**, and **IsType** in the Pac CLI Power Fx REPL.

In this first example, no type information is provided to **ParseJSON**, so it returns a dynamic value.

Power Apps

```
>> Set( rec, ParseJSON( "{""a"":1}" ) )
rec: <Dynamic: Use Value, Text, Boolean, or other functions to establish the type>
```

The field is implicitly converted to a number when used in a numerical context.

Power Apps

```
>> 1 + rec.a
2
```

As an alternative, this example explicitly converts the record to a typed Power Fx record with the second argument to **ParseJSON**.

Power Apps

```
>> ParseJSON( "{""a"":1}", Type( {a: Number} ) )
{a:1}
```

And another alternative, this example explicitly converts the record to a typed Power Fx record using **AsType**.

Power Apps

```
>> AsType( ParseJSON( "{""a"":1}" ), Type( {a: Number} ) )
{a:1}
```

Finally, if you aren't sure, this example tests the type before converting it with the **IsType** function.

Power Apps

```
>> IsType( ParseJSON( "{""a"":1}" ), Type( {a: Number} ) )
true
```

Now, Today, IsToday, UTCNow, UTCToday, IsUTCToday functions

Article • 04/20/2025

Expand table

Functions	Applies to
Now	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Today **IsToday**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
UTCNow **UTCToday**	✅ Dataverse formula columns ✅ Power Pages

Returns the current date and time, and tests whether a date/time value is today.

> ⓘ Note
>
> PAC CLI pac power-fx commands do not support the **UTCNow**, **UTCToday**, and **IsUTCToday** functions.

Description

The **Now** function returns the current date and time as a date/time value.

The **Today** function returns the current date as a date/time value. The time portion is midnight. **Today** has the same value throughout a day, from midnight today to midnight tomorrow.

The **IsToday** function tests whether a date/time value is between midnight today and midnight tomorrow. This function returns a Boolean (**true** or **false**) value.

Now, **Today**, and **IsToday** functions work with the local time of the current user.

UTCNow, **UTCToday**, and **IsUTCToday** functions are the same as their non-UTC counterparts but work with time zone independent values and use Coordinated Universal Time (UTC).

> ⓘ Note

- **UTCNow**, **UTCToday**, and **IsUTCToday** are only available in Microsoft Dataverse for Teams formula columns, and only for time-independent fields and values.
- **Now**, **Today**, and **IsToday** are not available in Dataverse for Teams formula columns as evaluations are done without the knowledge of the current user's local time zone. More information: Work with formula table columns in Dataverse for Teams

See Date, Time, and DateTime in the data types documentation and working with dates and times for more information.

Volatile Functions

Now, **Today**, **UTCNow**, and **UTCToday** are volatile functions. These functions return a different value for each evaluation.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Now()** will not change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it is part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = DateAdd(Now(), Slider1.Value, Minutes)** then the current time is retrieved each time the Slider control's value changes and the label's text property is reevaluated.

When used in a behavior formula, volatile functions will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Using the user's local time

Now()

Today()

IsToday(*DateTime*)

- *DateTime* - Required. The date/time value to test.

Using Coordinated Universal Time (UTC)

UTCNow()

UTCToday()

IsUTCToday(*TimeZoneIndependentTime*)

- *TimeZoneIndependentDateTime* - Required. The time zone independent date/time value to test.

Examples

For the examples in this section, the current time is **8:58 PM** on **July 11, 2021** in the Pacific Time Zone (UTC-8) and the language is **en-us**.

Expand table

Formula	Description	Result
Text(Now(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time in the user's time zone, and displays it as a string.	"07/11/2021 20:58:00"
Text(Today(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date only, leaving the time portion as midnight, and displays it as a string.	"07/12/2021 00:00:00"
IsToday(Now())	Tests whether the current date and time is between midnight today and midnight tomorrow.	**true**
IsToday(Today())	Tests whether the current date is between midnight today and midnight tomorrow.	**true**
Text(DateAdd(Now(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time, adds 12 days to the result, and displays it as a string.	"07/23/2021 20:58:00"
Text(DateAdd(Today(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date, adds 12 days to the result, and displays it as a string.	"07/23/2021 00:00:00"
IsToday(DateAdd(Now(), 12))	Tests whether the current date and time, plus 12 days, is between midnight today and midnight tomorrow.	**false**
IsToday(DateAdd(Today(), 12))	Tests whether the current date, plus 12 days, is between midnight today and midnight tomorrow.	**false**
Hour(UTCNow())	Retrieves the current date and time in UTC and extracts the hour only, which is 8 hours ahead of local time.	4

Formula	Description	Result
Day(UTCToday())	Retrieves the current date only in UTC and extracts the day, which is 1 day ahead of local time.	12
IsUTCToday(UTCNow())	Tests whether the current date and time is between midnight today and midnight tomorrow, all in UTC time.	**true**
IsUTCToday(UTCToday())	Tests whether the current date is between midnight today and midnight tomorrow, all in UTC time.	**true**

Display a clock that updates in real time

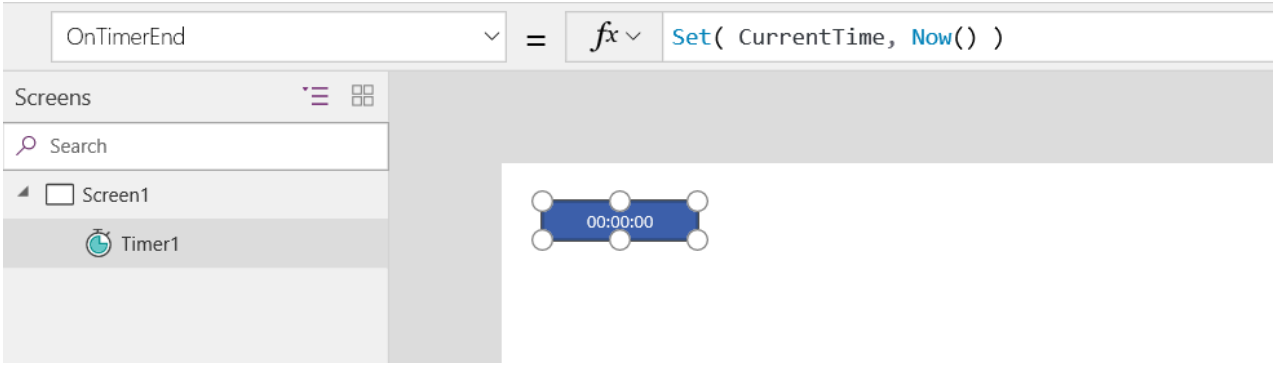
1. Add a Timer control, set its **Duration** property to **1000**, and set its **Repeat** property to **true**.

   The timer will run for one second, automatically start over, and continue that pattern.

2. Set the control's **OnTimerEnd** property to this formula:

   Set(CurrentTime, Now())

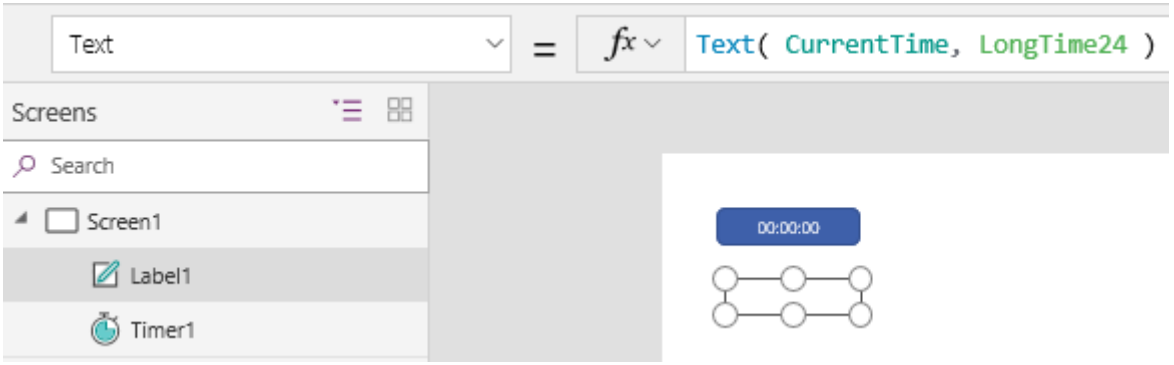
   Whenever the timer starts over (after each second), this formula sets the **CurrentTime** global variable to the current value of the **Now** function.

3. Add a Label control, and set its **Text** property to this formula:

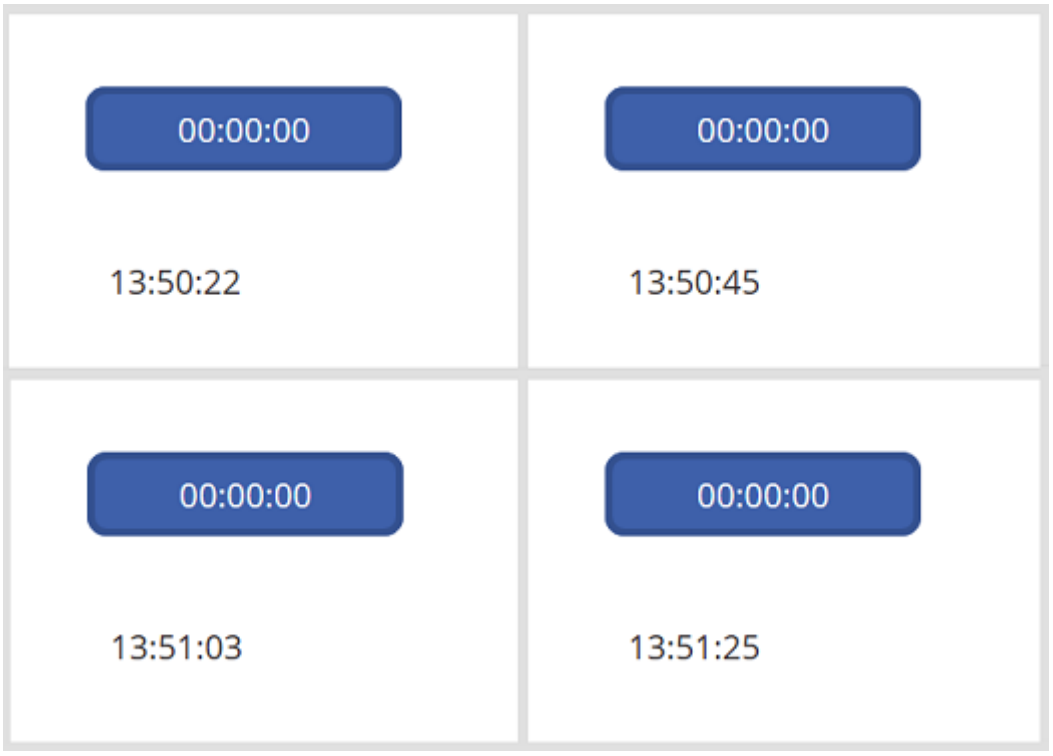
   Text(CurrentTime, LongTime24)

   Use the Text function to format the date and time however you want, or set this property to just **CurrentTime** to show hours and minutes but not seconds.



4. Preview the app by pressing F5, and then start the timer by clicking or tapping it.

   The label continually shows the current time, down to the second.



5. Set the timer's **AutoStart** property to **true** and its **Visible** property to **false**.

   The timer is invisible and starts automatically.

6. Set the screen's OnStart property so that the **CurrentTime** variable has a valid value, as in this example:

   Set(CurrentTime, Now())

   The label appears as soon as the app starts (before the timer runs for a full second).

JSON function

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Generates a JSON text string for a table, a record, or a value.

Description

The **JSON** function returns the JavaScript Object Notation (JSON) representation of a data structure as text so that it's suitable for storing or transmitting across a network. [ECMA-404](https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf) and IETF RFC 8259 describe the format, which is widely used by JavaScript and other programming languages.

Canvas apps support the data types that this table lists with details about their text representation:

Expand table

Data type	Description	Result example
Boolean	*true* or *false*.	`true`
Color	String that contains the 8-digit hexadecimal representation for the color. This representation takes the format *#rrggbbaa*, where *rr* is the red component, *gg* is green, *bb* is blue, and *aa* is the alpha channel. For the alpha channel, **00** is fully transparent, and **ff** is fully opaque. You can pass the string to the **ColorValue** function.	`"#102030ff"`
Currency	Number that uses the appropriate decimal separator for the user's language. Scientific notation is used if needed.	`1.345`
Date	String that contains the date in ISO 8601 **yyyy-mm-dd** format.	`"2019-03-31"`
DateTime	String that contains an ISO 8601 date/time. Date/time values are in UTC, as the ending "Z" indicates.	`"2019-03-31T22:32:06.822Z"`

Data type	Description	Result example
GUID	String that contains the GUID value. Letters are lowercase.	`"751b58ac-380e-4a04-a925-9f375995cc40"`
Image, Media	If **JSONFormat.IncludeBinaryData** is specified, media files are encoded in a string. Web references that use the http: or https: URL scheme aren't modified. References to in-memory binary data are encoded with the "data:*mimetype*; base64,..." format. In-memory data includes images that users capture by using the **Camera** control and any other references with the appres: and blob: URL schemes.	`"..."`
Number	Number that uses the appropriate decimal separator for the user's language. Scientific notation is used if needed.	`1.345`
Option set	Numeric value of the choice, not the label that's used for display. The numeric value is used because it's language independent.	`1001`
Time	String that contains an ISO 8601 *hh:mm:ss.fff* format.	`"23:12:49.000"`
Record	Comma-delimited list, between **{** and **}**, of fields and their values. This notation resembles that for records in canvas apps, but the name is always between double quotation marks. This format doesn't support records that are based on many-to-one relationships.	`{ "First Name": "Fred", "Age": 21 }`
Table	Comma-delimited list, between **[** and **]**, of records. This format doesn't support tables that are based on one-to-many relationships. Use the **JSONFormat.FlattenValueTables** option to remove the record for single column tables with the column named **Value**.	`[ { "First Name": "Fred", "Age": 21 }, { "First Name": "Jean", "Age": 20 } ]`
Two option	Boolean value of the two option, *true* or *false*, not the label that's used for display. The Boolean value is used because it's language independent.	`false`

Data type	Description	Result example
Hyperlink, Text	String between double quotation marks. The function escapes embedded double-quotation marks with a backslash, replaces newlines with "\n", and makes other standard JavaScript replacements.	`"This is a string."`

Specify the optional *Format* argument to control how readable the result is and how unsupported and binary data types are handled. By default, the output is as compact as possible with no unnecessary spaces or newlines, and unsupported data types and binary data aren't allowed. You can combine multiple formats if you specify the **&** operator.

Expand table

JSONFormat enum	Description
JSONFormat.Compact	Default. The output is as compact as possible with no added spaces or newlines.
JSONFormat.FlattenValueTables	As a Value table, `[1,2,3]` notation results in a table containing records where each record has a single **Value** column. In JSON, that same notation represents an array of three numbers. To make interoperability between the two easier, this option flattens a Power Fx Value table to a JSON friendly array instead of an array of records.
JSONFormat.IndentFour	To improve readability, the output contains a newline for each column and nesting level and uses four spaces for each indentation level.
JSONFormat.IncludeBinaryData	The result includes image, video, and audio-clip columns. This format can dramatically increase the result's size and degrade your app's performance.
JSONFormat.IgnoreBinaryData	The result doesn't include image, video, or audio-clip columns. If you specify neither **JSONFormat.IncludeBinaryData** nor **JSONFormat.IgnoreBinaryData**, the function produces an error if it encounters binary data.
JSONFormat.IgnoreUnsupportedTypes	Unsupported data types are allowed, but the result won't include them. By default, unsupported data types produce an error.

Use the **ShowColumns** and **DropColumns** functions to control which data the result includes and to remove unsupported data types.

Because **JSON** can be both memory and compute intensive, you can use this function only in behavior functions. You can capture the result from **JSON** into a variable, which you can then use in data flow.

If a column has both a display name and a logical name, the result contains the logical name. Display names reflect the language of the app user and are, therefore, inappropriate for data transfer to a common service.

Syntax

JSON(*DataStructure* [, *Format*])

- *DataStructure* – Required. The data structure to convert to JSON. Tables, records, and primitive values are supported, arbitrarily nested.
- *Format* - Optional. **JSONFormat** enum value. The default value is **JSONFormat.Compact**, which doesn't add newlines or spaces and blocks binary data and unsupported columns.

Examples

Hierarchical data

1. Insert a **Button** control, and set its **OnSelect** property to this formula.

```
Power Apps
```

```
ClearCollect( CityPopulations,
    { City: "London",    Country: "United Kingdom", Population: 8615000
},
    { City: "Berlin",    Country: "Germany",        Population: 3562000
},
    { City: "Madrid",    Country: "Spain",          Population: 3165000
},
    { City: "Hamburg",   Country: "Germany",        Population: 1760000
},
    { City: "Barcelona", Country: "Spain",          Population: 1602000
},
    { City: "Munich",    Country: "Germany",        Population: 1494000
}
);
```

```
ClearCollect( CitiesByCountry, GroupBy( CityPopulations, "Country",
"Cities" ) )
```

2. Select the button while holding down the Alt key.

   The **CitiesByCountry** collection is created with this data structure, which you can show by selecting **Collections** on the **File** menu and then selecting the name of the collection.

Cities	Country
City: "London", Population: 8,615,000	"United Kingdom"
City: "Berlin", Population: 3,562,000 City: "Hamburg", Population: 1,760,000 City: "Munich", Population: 1,494,000	"Germany"
City: "Madrid", Population: 3,165,000 City: "Barcelona", Population: 1,602,000	"Spain"

   You can also show this collection by selecting **Settings** > **Upcoming features** > **Enable formula bar result view**, selecting the name of the collection in the formula bar, and then selecting the down arrow next to the name of the collection under the formula bar.

```
ClearCollect( CityPopulations,
    { City: "London",    Country: "United Kingdom", Population: 8615000 },
    { City: "Berlin",    Country: "Germany",        Population: 3562000 },
    { City: "Madrid",    Country: "Spain",          Population: 3165000 },
    { City: "Hamburg",   Country: "Germany",        Population: 1760000 },
    { City: "Barcelona", Country: "Spain",          Population: 1602000 },
    { City: "Munich",    Country: "Germany",        Population: 1494000 }
);
ClearCollect( CitiesByCountry, GroupBy( CityPopulations, "Country", "Cities" ) )
```

CitiesByCountry | Data type: **Table**

Cities	Country
[Table]	United Kingdom
[Table]	Germany
[Table]	Spain

3. Insert another button, and set its **OnSelect** property to this formula:

```
Power Apps

Set( CitiesByCountryJSON, JSON( CitiesByCountry ) )
```

   This formula sets the global variable **CitiesByCountryJSON** to the JSON representation for **CitiesByCountry**.

4. Select the button while holding down the Alt key.

5. Insert a **Label** control, and set its **Text** property to this variable.

```
Power Apps

CitiesByCountryJSON
```

   The label shows this result, all on a single line with no spaces, suitable for transmission across a network:

```
JSON

[
  {
    "Cities": [{ "City": "London", "Population": 8615000 }],
    "Country": "United Kingdom"
  },
  {
    "Cities": [
      { "City": "Berlin", "Population": 3562000 },
      { "City": "Hamburg", "Population": 1760000 },
      { "City": "Munich", "Population": 1494000 }
    ],
    "Country": "Germany"
  },
  {
    "Cities": [
      { "City": "Madrid", "Population": 3165000 },
      { "City": "Barcelona", "Population": 1602000 }
    ],
    "Country": "Spain"
  }
]
```

6. Change the second button's formula to make the output more readable.

```
Power Apps
```

```
Set( CitiesByCountryJSON, JSON(CitiesByCountry, JSONFormat.IndentFour
))
```

7. Select the second button while holding down the Alt key.

   The label shows the more readable result.

JSON


```
[
  {
    "Cities": [
      {
        "City": "London",
        "Population": 8615000
      }
    ],
    "Country": "United Kingdom"
  },
  {
    "Cities": [
      {
        "City": "Berlin",
        "Population": 3562000
      },
      {
        "City": "Hamburg",
        "Population": 1760000
      },
      {
        "City": "Munich",
        "Population": 1494000
      }
    ],
    "Country": "Germany"
  },
  {
    "Cities": [
      {
        "City": "Madrid",
        "Population": 3165000
      },
      {
        "City": "Barcelona",
        "Population": 1602000
      }
    ],
    "Country": "Spain"
  }
]
```

Images and media in base64

1. Add an **Image** control.

   This control brings **SampleImage** with it.

2. Add a **Button** control, and set its **OnSelect** property to this formula.

   Power Apps

   ```
   Set( ImageJSON, JSON( SampleImage, JSONFormat.IncludeBinaryData ) )
   ```

3. Select the button while holding down the Alt key.

4. Add a label, and set its **Text** property to this variable.

   Power Apps

   ```
   ImageJSON
   ```

5. Resize the control and reduce the font size as needed to show most of the result.

   The label shows the text string that the **JSON** function captured.

   JSON

   ```
   "
   LTgiPz4NCjxzdmcgdmVyc2lvbj0iMS4xIg0KCSB4bWxucz0iaHR0cDovL3d3dy53My5vcmc
   vMjAwMC9zdmciIHhtbG5zOnhsaW5rPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5L3hsaW5rIi
   B4bWxuczphPSJodHRwOi8vbnMuYWRvYmUuY29tL0Fkb2JlU1ZHVmlld2VyRXh0ZW5zaW9uc
   y8zLjAvIg0KCSB4PSIwcHgiIHk9IjBweCIgd2lkdGg9IjI3MHB4IiBoZWlnaHQ9IjI3MHB4
   IiBlbmFibGUtYmFja2dyb3VuZD0ibmV3IDAgMCAyNzAgMjcwIiB4bWw6c3BhY2U9InByZXN
   lcnZlIj4NCgk8ZyBjbGFzcz0ic3QwIj4NCgkJPHJlY3QgeT0iMC43IiBmaWxsPSIjRTlFOU
   U5IiB3aWR0aD0iMjY5IiBoZWlnaHQ9IjI2OS4zIi8+DQoJCTxwb2x5Z29uIGZpbGw9IiNDQ
   kNCQ0EiIHBvaW50cz0iMjc3LjksMTg3LjEgMjQ1LDE0My40IDE4OC42LDIwMi44IDc1LDgw
   LjUgLTQuMSwxNjUuMyAtNC4xLDI3MiAyNzcuOSwyNzIiLz4NCgkJPGVsbGlwc2UgaWQ0
   iI0NCQ0JDQSIgY3g9IjIwMi40IiBjeT0iODQuMSIgcng9IjI0LjQiIHJ5PSIyNC4zIi8+DQ
   oJPC9nPg0KPC9zdmc+"
   ```

Value tables

This formula:

Power Apps

```
JSON( [1,2,3] )
```

produces the text string **[{"Value":1},{"Value":2},{"Value":3}]**.

The same formula with the JSONFormat.FlattenValueTables option:

```
Power Apps
JSON( [1,2,3], JSONFormat.FlattenValueTables )
```

produces the text string **[1,2,3]**.

Note that the FlattenValueTables option has no impact when using JSON with the **CityPopulations** or **CitiesByCountry** collections as these tables are not Value tables. A Value table has a single column and it must be named "Value".

Feedback

Was this page helpful? Yes No

Provide product feedback

Language function

10/10/2025

Applies to: Canvas apps Copilot Studio Desktop flows Power Platform CLI Dataverse functions Power Pages

Returns the language tag of the current user.

Description

The **Language** function returns the language, script, and region of the current user as a language tag.

Use the language information to tailor your app across locales. For example, if you're creating an app that is used in Italy and France, you can use **Language** to automatically display Italian and French strings to your users in those different locations.

Language tags

A *language tag* can be in one of three formats:

Expand table

Return value	Description
"***lg-RE***"	*lg* is the two character abbreviation for the language and *RE* is the two character abbreviation for the region. It's the most common return type. For example, "en-GB" is returned for United Kingdom.
"***lg***"	*lg* is the two character abbreviation for the language. It's the format used when Power Apps has information about the language, but doesn't have information for the specific region.
"***lg-scrp-RE***"	*lg* is the two character abbreviation for the language, *scrp* is the four character abbreviation for the script, and *RE* is the two character abbreviation for the region.

Power Apps uses the IETF BCP-47 language tag format.

To see the list of supported language tags, type **Value("1",)** in the formula bar or advanced view and scroll through the list of locales suggested for the second argument.

The **Text** and **Value** functions also use language tags. Use these functions for translating to and from text strings in a globally aware manner. When passing a language tag to these functions

and the region wouldn't make a difference, you can use just the language portion of the tag.

Syntax

Language()

Examples

User's locale

It's assumed that the host operating system and/or browser are using the default locale for the location.

Expand table

Formula	Location	Return Value
Language()	Lisbon, Portugal	"pt-PT"
Language()	Rio de Janeiro, Brazil	"pt-BR"
Language()	Atlanta, USA	"en-US"
Language()	Manchester, United Kingdom	"en-GB"
Language()	Paris, France	"fr-FR"
Language()	Roseau, Dominica	"en"
Language()	Belgrade, Serbia	"sr-cyrl-RS" or "sr-latn-RS," depending on the user's system settings

Localization table

A simple approach to localization is to create an Excel spreadsheet mapping an author defined **TextID** to a translated text for the user's language. Although you could use a collection or any other data source for this table, we chose Excel because it's easy to edit and manage outside of the app by translators.

1. Create the following table in Excel:

TextID	LanguageTag	LocalizedText
"Hello"	"it"	"Ciao"
"Hello"	"fr"	"Salut"
"Hello"	*blank*	"Hello"
"Goodbye"	"it"	"Arrivederci"
"Goodbye"	"fr"	"Au revoir"
"Goodbye"	*blank*	"Goodbye"

The entry with *blank* for the **Language** column is used as the default if there's no specific text string found for a given language. This entry must appear after all other entries for a given **TextID**.

For our purposes, we only need to look at the language of the locale and not the region. If regional considerations were important, we could included the full language tag value in the table above.

2. Use the **Insert** ribbon, **Table** command, to make this into a proper Excel table. By default, it's named **Table1** but you can name it whatever you like with the **Table Tools/Design** ribbon and the **Table Name:** text box on the far left hand side.

3. Save the Excel file to your local file system.

4. In Power Apps, in the right-hand pane, click, or tap the **Data Sources** tab, and then click or tap **Add data source**.

5. Click or tap **Add static data to your app**, click, or tap the Excel file that you saved, and then click or tap **Open**.

6. Select the table that you created, and then click, or tap **Connect**.

In your app, wherever you used the text **"Hello"** before, use this formula instead:

- **LookUp(Table1, TextID = "Hello" && (LanguageTag = Left(Language(), 2) || IsBlank(LanguageTag))).LocalizedText**

This formula will lookup the appropriate **LocalizedText** value for the language of the user, and if that isn't found, will fall back on the default *blank* version.

Translated strings in other languages could be longer than they're in your language. In many cases, the labels and other elements that display the strings in your user interface are needed to be wider to accommodate.

Translation service

You can translate text on demand using a translation service, such as the Microsoft Translator service:

1. In Power Apps, in the right-hand pane, click or tap the **Data Sources** tab, and then click or tap **Add data source**.
2. Click or tap **Microsoft Translator**.

In your app, wherever you would have used the text **"Hello"** before, use this formula instead:

- **MicrosoftTranslator.Translate("Hello", Language())**

The Microsoft Translator service uses the same language tags that the **Language** function returns.

This approach comes with some drawbacks when compared to the previous example which utilized a pretranslated table of text strings:

- The translation takes time to complete, requiring a call to a service across the network. It results in a lag to see the translated text in your app.
- The translation is mechanical and may not be what you anticipate or be the best choice for the situation within your app.

First, FirstN, Index, Last, and LastN functions

04/20/2025

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the first, last, or a specific record, or a set of first or last records, from a table.

Description

The **First** function returns the first record of a table.

The **FirstN** function returns the first set of records of a table; the second argument specifies the number of records to return.

The **Last** function returns the last record of a table.

The **LastN** function returns the last set of records of a table; the second argument specifies the number of records to return.

The **Index** function returns a record of a table based on its ordered position in the table. Record numbering begins with 1 so `First( table )` returning the same record as `Index( table, 1 )`. **Index** returns an error if the requested record index is less than 1, greater than the number of records in the table, or the table is empty.

First, **Index**, and **Last** return a single record. **FirstN** and **LastN** return a table, even if you specify only a single record.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

For example, when used with a data source containing a large table with 1 million records, **Last** will be subject to the non-delegation limit and will not return the last record of the entire data source. Likewise, using **Index** to request a record in the middle of 1 million records will result in an error because the index is out of range based on the non-delegation limit.

Syntax

First(*Table*)
Last(*Table*)

- *Table* - Required. Table to operate on.

FirstN(*Table* [, *NumberOfRecords*])
LastN(*Table* [, *NumberOfRecords*])

- *Table* - Required. Table to operate on.
- *NumberOfRecords* - Optional. Number of records to return. If you don't specify this argument, the function returns one record.

Index(*Table*, *RecordIndex*)

- *Table* - Required. Table to operate on.
- *RecordIndex* - Required. The index of the record to return. Record numbering begins with 1.

Examples

For the following examples, we'll use the **IceCream** data source, which contains the data in this table:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200
"Strawberry"	300
"Mint Chocolate"	60
"Pistachio"	200

This table can be placed in a collection with this formula (put in the OnStart formula for a Button control and press the button):

Power Fx

```
Collect( IceCream, Table( { Flavor: "Chocolate", Quantity: 100 },
                          { Flavor: "Vanilla", Quantity: 200 },
                          { Flavor: "Strawberry", Quantity: 300 },
                          { Flavor: "Mint Chocolate", Quantity: 60 },
                          { Flavor: "Pistachio", Quantity: 200 } ) )
```

Expand table

Formula	Description	Result
First(IceCream)	Returns the first record of **IceCream**.	{ Flavor: "Chocolate", Quantity: 100 }
Last(IceCream)	Returns the last record of **IceCream**.	{ Flavor: "Pistachio", Quantity: 200 }
Index(IceCream, 3)	Returns the third record of **IceCream**.	{ Flavor: "Strawberry", Quantity: 300 }
FirstN(IceCream, 2)	Returns a table containing the first two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Chocolate"</td><td>100</td></tr><tr><td>"Vanilla"</td><td>200</td></tr></table>
LastN(IceCream, 2)	Returns a table containing the last two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Mint Chocolate"</td><td>60</td></tr><tr><td>"Pistachio"</td><td>200</td></tr></table>
Index(IceCream, 4).Quantity	Returns the fourth record of the table, and extracts the Quantity column.	60
Index(IceCream, 10)	Returns an error since the record requested is beyond the bounds of the table.	*Error*

First, FirstN, Index, Last, and LastN functions

04/20/2025

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the first, last, or a specific record, or a set of first or last records, from a table.

Description

The **First** function returns the first record of a table.

The **FirstN** function returns the first set of records of a table; the second argument specifies the number of records to return.

The **Last** function returns the last record of a table.

The **LastN** function returns the last set of records of a table; the second argument specifies the number of records to return.

The **Index** function returns a record of a table based on its ordered position in the table. Record numbering begins with 1 so `First( table )` returning the same record as `Index( table, 1 )`. **Index** returns an error if the requested record index is less than 1, greater than the number of records in the table, or the table is empty.

First, **Index**, and **Last** return a single record. **FirstN** and **LastN** return a table, even if you specify only a single record.

Delegation

When used with a data source, these functions can't be delegated. Only the first portion of the data source will be retrieved and then the function applied. The result may not represent the complete story. A warning may appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

For example, when used with a data source containing a large table with 1 million records, **Last** will be subject to the non-delegation limit and will not return the last record of the entire data source. Likewise, using **Index** to request a record in the middle of 1 million records will result in an error because the index is out of range based on the non-delegation limit.

Syntax

First(*Table*)
Last(*Table*)

- *Table* - Required. Table to operate on.

FirstN(*Table* [, *NumberOfRecords*])
LastN(*Table* [, *NumberOfRecords*])

- *Table* - Required. Table to operate on.
- *NumberOfRecords* - Optional. Number of records to return. If you don't specify this argument, the function returns one record.

Index(*Table*, *RecordIndex*)

- *Table* - Required. Table to operate on.
- *RecordIndex* - Required. The index of the record to return. Record numbering begins with 1.

Examples

For the following examples, we'll use the **IceCream** data source, which contains the data in this table:

Flavor	Quantity
"Chocolate"	100
"Vanilla"	200
"Strawberry"	300
"Mint Chocolate"	60
"Pistachio"	200

This table can be placed in a collection with this formula (put in the OnStart formula for a Button control and press the button):

Power Fx

```
Collect( IceCream, Table( { Flavor: "Chocolate", Quantity: 100 },
                          { Flavor: "Vanilla", Quantity: 200 },
                          { Flavor: "Strawberry", Quantity: 300 },
                          { Flavor: "Mint Chocolate", Quantity: 60 },
                          { Flavor: "Pistachio", Quantity: 200 } ) )
```

Expand table

Formula	Description	Result
First(IceCream)	Returns the first record of **IceCream**.	{ Flavor: "Chocolate", Quantity: 100 }
Last(IceCream)	Returns the last record of **IceCream**.	{ Flavor: "Pistachio", Quantity: 200 }
Index(IceCream, 3)	Returns the third record of **IceCream**.	{ Flavor: "Strawberry", Quantity: 300 }
FirstN(IceCream, 2)	Returns a table containing the first two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Chocolate"</td><td>100</td></tr><tr><td>"Vanilla"</td><td>200</td></tr></table>
LastN(IceCream, 2)	Returns a table containing the last two records of **IceCream**.	<table><tr><th>Flavor</th><th>Quantity</th></tr><tr><td>"Mint Chocolate"</td><td>60</td></tr><tr><td>"Pistachio"</td><td>200</td></tr></table>
Index(IceCream, 4).Quantity	Returns the fourth record of the table, and extracts the Quantity column.	60
Index(IceCream, 10)	Returns an error since the record requested is beyond the bounds of the table.	*Error*

Launch and Param functions

Article • 03/04/2025

Applies to: Canvas apps

Launches a webpage or a canvas app and provides access to launch parameters.

Launch

Launches a webpage or a canvas app. The function supports:

- **Address** (required), the URL of the webpage of the canvas app.
- **Parameters** (optional), named values to pass to the webpage or canvas app. In a canvas app, parameters can be read with the **Param** function.
- **Target** (optional), the browser tab in which to launch the webpage or canvas app.

Launch can only be used in behavior formulas.

> **Note**
>
> - If the Launch formula is used to open Power Apps and the app isn't visible to the end user, it's recommended to check if the app is blocked by a browser pop-up blocker or if the browser is wait for the user's response to a dialog box that asks whether they want to navigate to a new screen.
> - Only **https**: URIs are supported when using Launch in the Power Apps Visual in PowerBI.

Address

Webpages are launched via a URL address. For example:

Power Fx

```
Launch( "https://bing.com" )
```

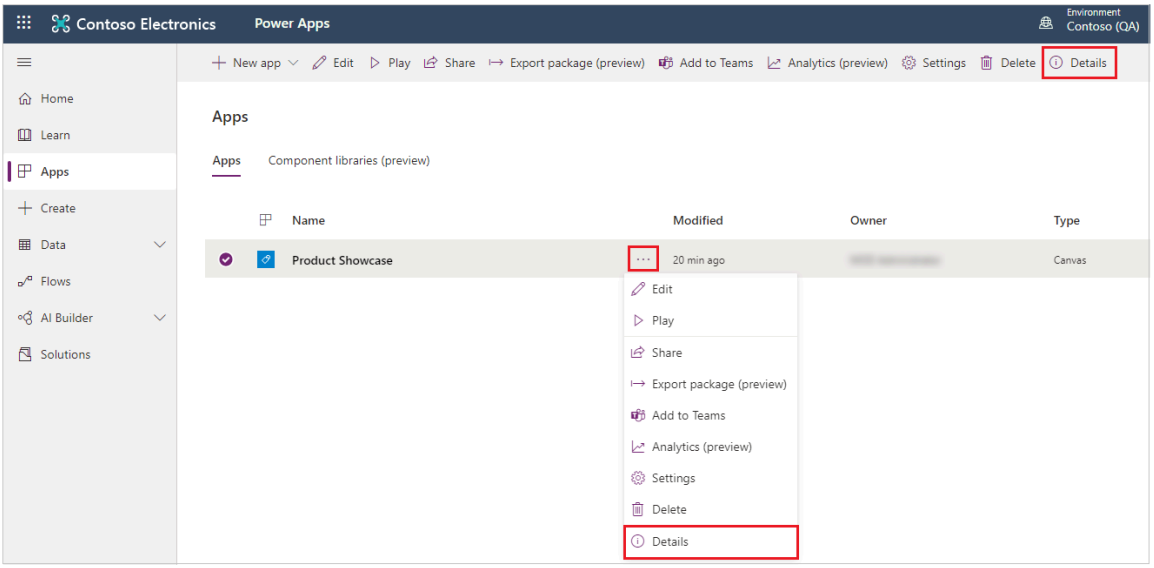
You can launch canvas apps with **Web link** or **App URI** (app id prefixed with `/providers/Microsoft.PowerApps/apps/`). To find these values for an app:

1. Go to Power Apps.

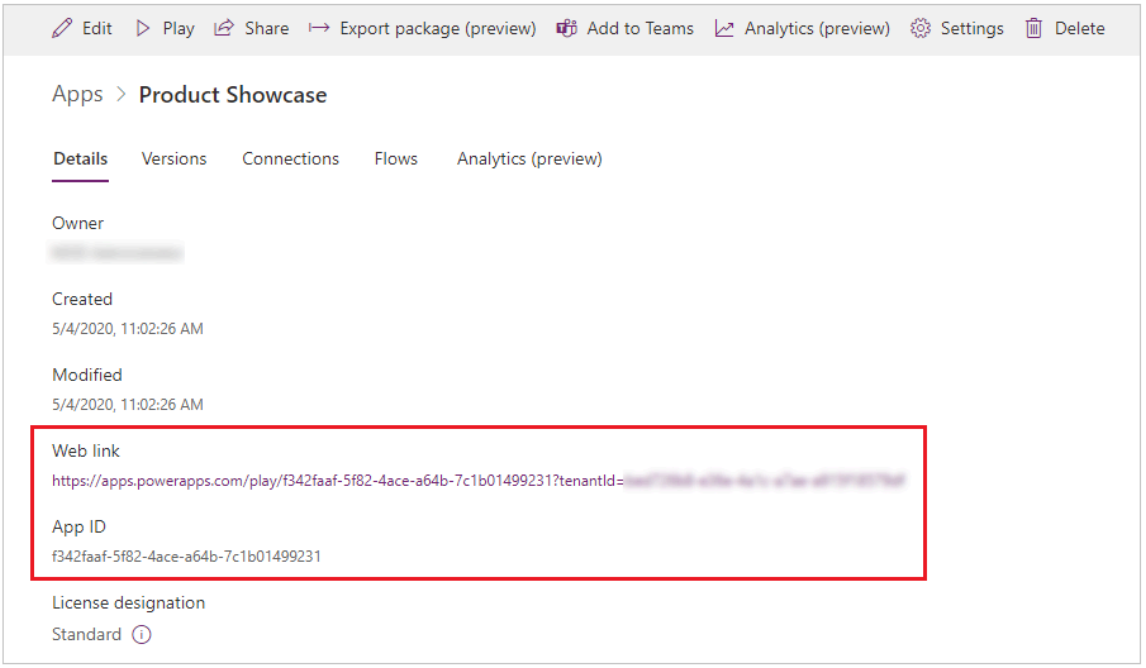
2. Select **Apps** from left navigation pane.

3. Select your app.

4. Select **Details** from top menu.
   You can also select **...** (**More Commands**) and then select **Details** from the drop-down menu.



5. Copy **Web link**.



The **Web link** can be used in any web page and will launch the canvas app. It can also be used with the **Launch** function. For example:

```
Power Fx
```

```
Launch( "https://apps.powerapps.com/e/{environment id}/a/{app id}?tenantId=
{tenantId}" )
```

Native apps on a device can't be launched directly. There may be indirect options available on some platforms, such as a native app installing a custom URL scheme or registering with the web browser to offer an option for specific web sites.

Parameters

Launch can pass parameters to the webpage or canvas app. Parameters can be provided in two ways:

- An argument list of name value pairs. For example:

  Power Fx

  ```
  Launch( "https://bing.com/search", "q", "Power Apps", "count", 1 )
  ```

- A record of field values. For example:

  Power Fx

  ```
  Launch( "https://bing.com/search", { q: "Power Apps", count: 1 } )
  ```

  This form can be easier to work with as it makes the association between name and value clearer. It's the only form that supports the optional *LaunchTarget* argument.

The address and parameters are URL encoded before being passed to replace certain non-alphanumeric characters with `%` and a hexadecimal number as if the **EncodeUrl** function has been used on each.

When launching a webpage, a query string of parameters can be included at the end of the URL address. Any additional parameters provided to **Launch** will be added to the end of the query string. Query strings don't work when launching a canvas app.

When launching an app on a mobile device where that app is already running, parameters will not be refreshed in the running app. An app reload is required for parameters to be refreshed.

Target

Use the *LaunchTarget* argument to specify the target browser window in which to open the webpage or app. Use one of the following **LaunchTarget** enum values or provide a custom window *name*.

Expand table

LaunchTarget enum	Description
New	The webpage or app is opened in a new window or tab.
Replace	The webpage or app replaces the current window or tab.
name	Instead of an enum value, use your own text string to *name* the window or tab. *Self* is an internal only name that is only used by the **Launch** function. It has no impact on nor will it match the title of the window that your users see. If a window or tab with the given *name* already exists, its contents will be replaced. Otherwise, a new window or tab will be created. *name* can't begin with the underscore character "_".

New is the default enum when running in a web browser with **Replace** and *name* as available options. In a mobile player, **New** is the default for webpages with *name* as an available option; while the current canvas app will always be replaced by another canvas app.

> **Note**
>
> - Using a *LaunchTarget* with any value other than **New** in embedded scenarios (for example, Power BI or SharePoint) is not supported and may result in unexpected behavior. In the future, this behavior may change, or may cause an error.

Param

The **Param** function retrieves a parameter passed to the app when it was launched. If the named parameter wasn't passed, **Param** returns *blank*.

- When launching a canvas app from another canvas app, use the *Parameter* arguments to the **Launch** function. Parameter names and values will be automatically URL encoded.
- When launching a canvas app from a web page, add parameters to the query string of the canvas app web link. This involves adding `&parametername=parametervalue` assuming the query string has already been

started for the `tenantId`. For example, adding `&First%20Name=Vicki&category=3` would pass two parameters: `First Name` with a value of `"Vicki"` and `category` with a value of `"3"` (value type is *text*). The parameter name and value must be URL encoded if they contain spaces or special characters, similar to using the **EncodeURL** function.

- Param names are case-sensitive.
- Param names and values will be automatically URL decoded for use in your app.
- Parameter values do not change unless the app is reloaded. Using **Launch** on a mobile device where the app is already running does not refresh the parameters.
- Even if the parameter contains a number, the type returned by **Param** will always be a text string. Conversion to other types will automatically occur or use explicit conversions such as the **Value** function to convert explicitly to a number.

> ⓘ **Note**
>
> For custom pages, the only parameters accepted by the page are: recordId and entityName.

Syntax

Launch(*Address* [, *ParameterName1*, *ParameterValue1*, ...])

- *Address* – Required. The address of a webpage or the ID of an app to launch.
- *ParameterName(s)* – Optional. Parameter name.
- *ParameterValue(s)* – Optional. Corresponding parameter values to pass to the app or the webpage.

Launch(*Address*, { [*ParameterName1*: *ParameterValue1*, ...] } [, *LaunchTarget*])

- *Address* – Required. The address of a webpage or the ID of an app to launch.
- *ParameterName(s)* – Optional. Parameter name.
- *ParameterValue(s)* – Optional. Corresponding parameter values to pass to the app or the webpage.
- *LaunchTarget* – Optional. A **LaunchTarget** enum value or a custom *name*.

Param(*ParameterName*)

- *ParameterName* - Required. The name of the parameter passed to the app.

Reserved parameters

The following keywords are reserved (regardless of case) for internal use, and shouldn't be used as a custom parameter in the **Param()** function:

- amp%3Bauthmode
- amp%3Benableonbehalfof
- amp%3Bhidenavbar
- amp%3Blocale
- appmetadataversion
- authmode
- channeltype
- cordovapath
- correlationid
- debug
- delegatelaunchurl
- delegatelaunchurl
- disablepreviewredirect
- embedderorigin
- enableonbehalfof
- groupid
- hideappsplash
- hidenavbar
- hint
- hostclienttype
- hostmode
- iframecontainerid
- isfullscreen
- ispreviewmode
- loader
- loaderType
- locale
- location
- packagekind
- packageproperties
- playerresourcespath
- playersessionid
- powerappslanguage
- screencolor
- sdkversion
- site
- skipappmetadata
- skipiframecreation

- skiplaunchappcache
- source
- sourcetime
- standaloneconsent
- teamid
- teamtype
- tenantId
- theme
- uselocalpackagehostresources
- userteamrole

Examples

Simple Launch

From a canvas app to a web page:

Expand table

Formula	Description
Launch("http://bing.com/search", "q", "Power Apps", "count", 1)	Opens the webpage **https://bing.com/search?q=Power%20Apps&count=1**. A new window or tab is opened.
Launch("http://bing.com/search", { q: "Power Apps", count: 1 })	The same as the previous examples using the equivalent record notation. A new window or tab is opened.
Launch("http://bing.com/search", { q: "Power Apps", count: 1 }, LaunchTarget.Replace)	The same as the previous examples, replacing the current window or tab with the result if running in a web browser.
Launch("http://bing.com/search", { q: "Power Apps", count: 1 }, "Search Results")	The same as the previous example, creating or replacing the contents of the window or tab named **Search Results**.

From a canvas app to a canvas app

Update the app ID, screen name, and record number as appropriate.

```
Power Fx
```

```
Launch( "/providers/Microsoft.PowerApps/apps/YOUR-APP-ID",
        { Navigate: "Second Screen", Record: 34 }
)
```

From a web page to a canvas app

Update the app ID, tenant ID, screen name, and record number as appropriate.

HTML

```
<html>
  <body>
    <a
      href="https://apps.powerapps.com/play/e/YOUR-APP-ENVIRONMENT-
ID/a/YOUR-APP-ID?tenantId=YOUR-TENANT-ID&Navigate=Second%20Screen&Record=34"
    >
      Launch canvas app
    </a>
  </body>
</html>
```

Simple Param

Simple launch examples above to launch canvas app from web page or from another canvas app show simple examples for Param function:

Expand table

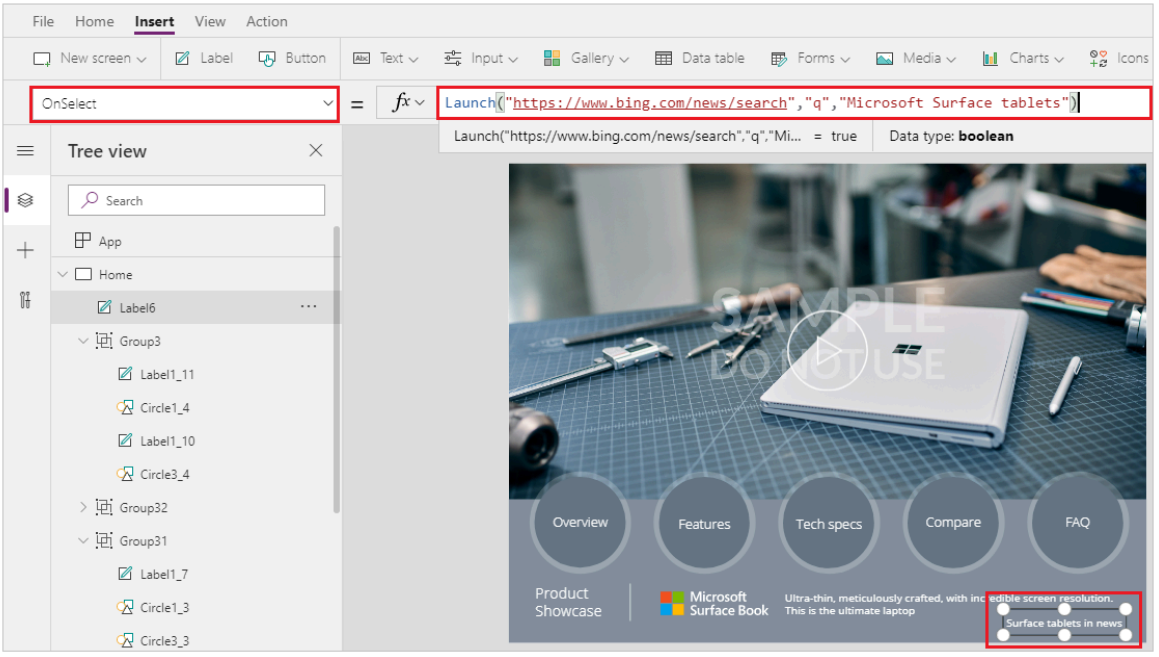
Formula	Description	Result
Param("Navigate")	The **Navigate** parameter was provided when the app was launched and is returned.	"Second Screen"
Param("Record")	The **Record** parameter was provided when the app was launched. Even though it was passed in as a number to the **Launch** function, the result from **Param** will be a text string that can be implicitly or explicitly converted to other types.	"34"
Param("User")	The **User** parameter wasn't provided. A *blank* value is returned that can be tested with the **IsBlank** function.	*blank*

Step by Step examples for Launch and Param

The **Product Showcase** tablet layout template was used for the following examples. To create an app with this template, follow the steps from create an app article and select the **Product Showcase** template. You can also use your own app.

Example - Launch

1. Go to Power Apps.
2. Select **Apps** from left navigation pane.
3. Select your app and then select **Edit**.
4. Select **Insert** from the menu and then select **Label**.
5. Move the label to the bottom right of the screen.
6. From the properties pane on the right-side, select **Color** as *white* and set **Border thickness** at *1*.
7. Select the **Text** property from right-side and enter text as *Surface tablets in news*.
8. From property list on top left, select **OnSelect**.
9. Enter formula as `Launch("https://www.bing.com/news/search","q","Microsoft Surface tablets")`. You can also use any other URL, parameter, and keywords of your choice.



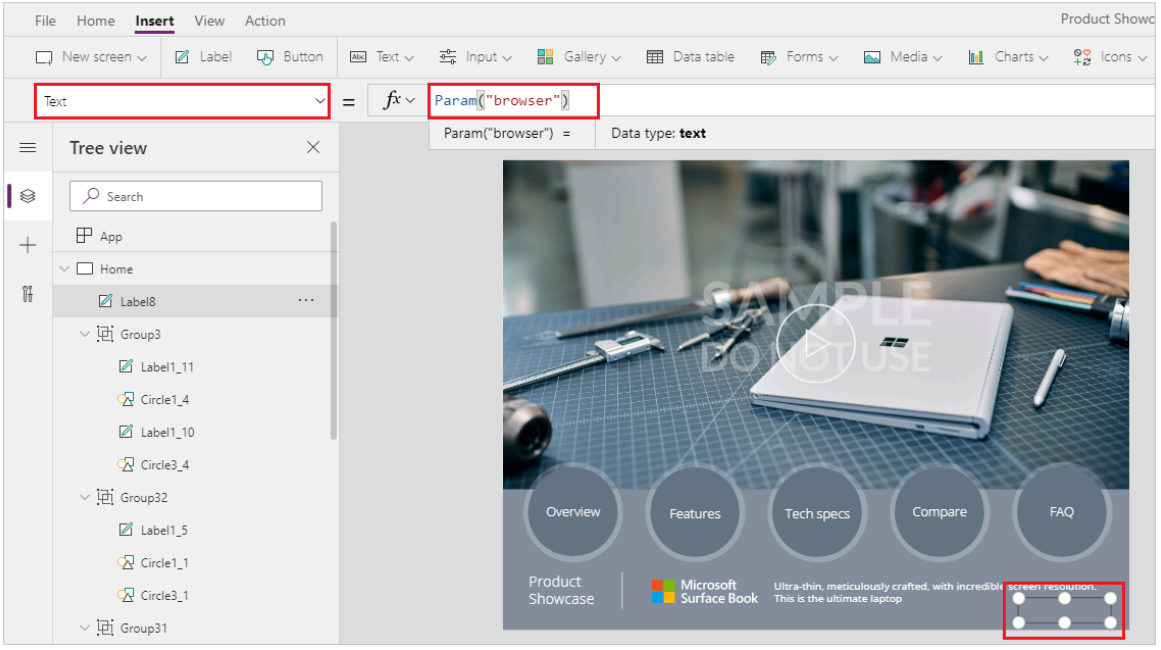
10. Save and publish the app.
11. Play the app.

12. Select label **Surface tablets in news** to launch news search with keywords *Microsoft Surface tablets*.

> **Tip**
>
> For scalability, you can replace the manually entered keywords in Launch function with variables.

Example - Param

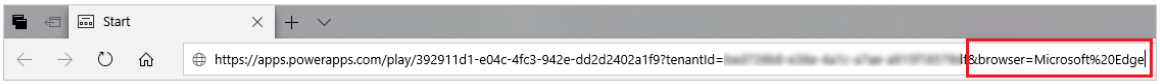
1. Go to Power Apps.
2. Select **Apps** from left navigation pane.
3. Select your app and then select **Edit**.
4. Select **Insert** from the menu and then select **Label**.
5. Move the label to the bottom right of the screen.
6. Select **Text** property for the label from top left.
7. Enter formula as `Param("browser")`. You can also use a different parameter of your choice.



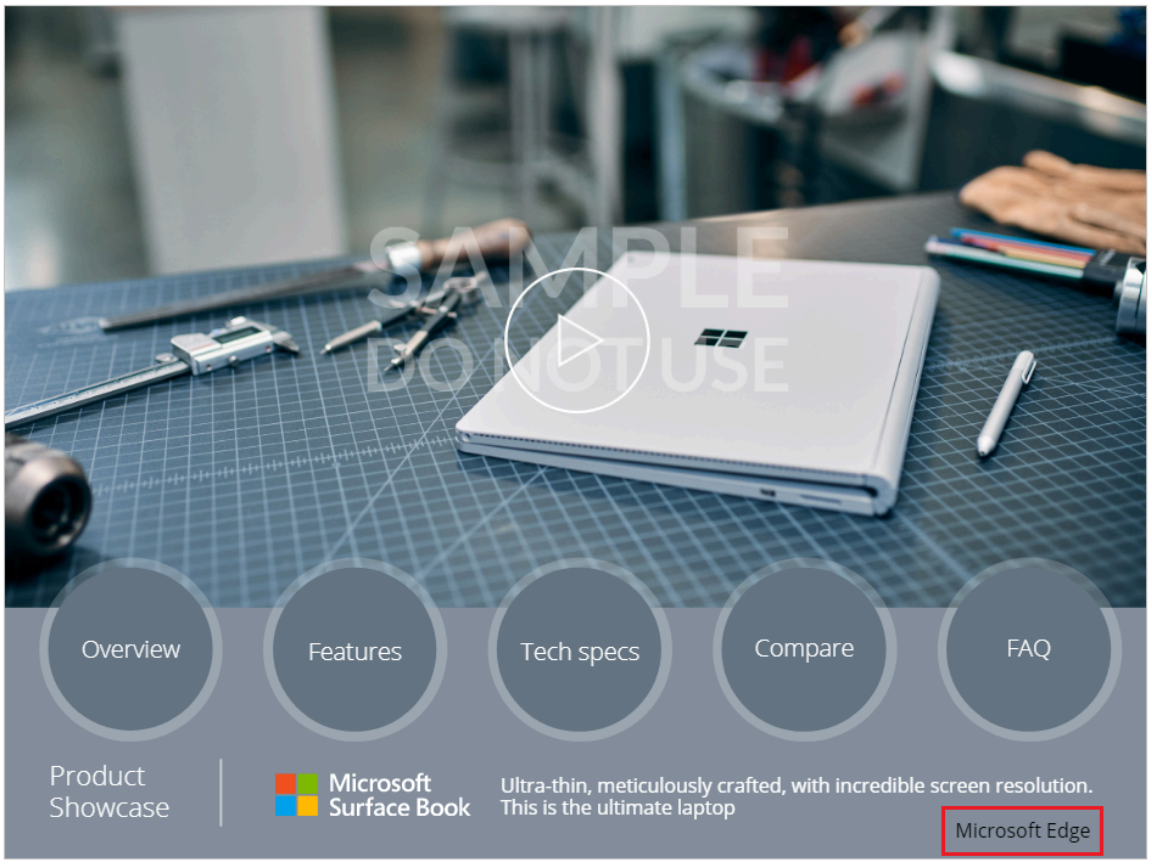
8. Save and publish the app.
9. Copy web link for your app from Power Apps.

10. Open a new browser.

11. Paste the app web link in the browser and append `&browser=Microsoft%20Edge` at the end.



12. When your app launches, the label shows the parameter value passed.

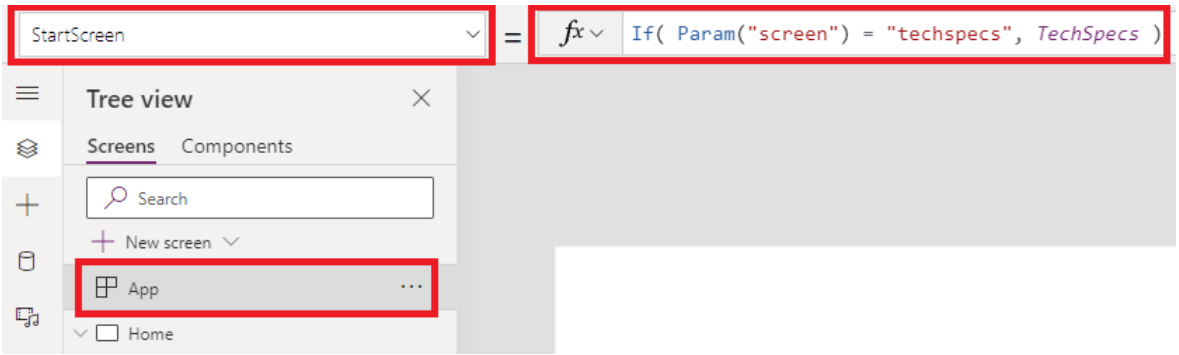


13. Close the app player and edit the app.

14. Select **App** from the Tree view on left navigation.

15. Select **StartScreen** property on top left.

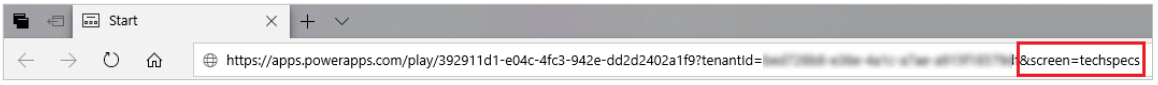
16. Enter the formula as `If( Param("screen") = "techspecs", TechSpecs )`.



If function in StartScreen property checks if parameter equals a certain value, in this case the value *techspecs*. And if it matches, returns the *TechSpecs* screen control to the **StartScreen** property.

> **Note**
>
> Replace the **TechSpecs** control name in the **If** function with the name of a screen control in your own app if you're not using the **Product Showcase** app template.

17. Save and publish the app.
18. Open a new browser.
19. Paste the app web link in the browser and append `&screen=techspecs` at the end.



20. The app directly launches with **TechSpecs** as the startscreen.

See also

Canvas app formula reference

Feedback

Was this page helpful? Yes No

Provide product feedback

Left, Mid, and Right functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Extracts the left, middle, or right portion of a string of text.

Description

The **Left**, **Mid**, and **Right** functions return a portion of a string.

- **Left** returns the beginning characters of a string.
- **Mid** returns the middle characters of a string.
- **Right** returns the ending characters of a string.

If you specify a single string as an argument, the function returns the portion that you requested of the string. If you specify a single-column table that contains strings, the function returns a single-column table with a **Value** column containing the portions that you requested of those strings. If you specify a multi-column table, you can shape it into a single-column table, as working with tables describes.

If the starting position is negative or beyond the end of the string, **Mid** returns *blank*. You can check the length of a string by using the **Len** function. If you request more characters than the string contains, the function returns as many characters as possible.

Syntax

Left(*String*, *NumberOfCharacters*)
Mid(*String*, *StartingPosition* [, *NumberOfCharacters*])
Right(*String*, *NumberOfCharacters*)

- *String* - Required. The string to from which to extract the result.
- *StartingPosition* - Required (**Mid** only). The starting position. The first character of the string is position 1.
- *NumberOfCharacters* - Required (**Left** and **Right** only). The number of characters to return. If omitted for the **Mid** function, the function returns the portion from the starting position until the end of the string.

Left(*SingleColumnTable*, *NumberOfCharacters*)
Mid(*SingleColumnTable*, *StartingPosition* [, *NumberOfCharacters*])

Right(*SingleColumnTable*, *NumberOfCharacters*)

- *SingleColumnTable* - Required. A single-column table of strings from which to extract the results.
- *StartingPosition* - Required (**Mid** only). The starting position. The first character of the string is position 1.
- *NumberOfCharacters* - Required (**Left** and **Right** only). The number of characters to return. If omitted for the **Mid** function, the function returns the portion from the starting position until the end of the string.

Examples

Single string

The examples in this section use a text-input control as their data source. The control is named **Author** and contains the string "E. E. Cummings".

Expand table

Formula	Description	Result
Left(Author.Text, 5)	Extracts up to five characters from the start of the string.	"E. E."
Mid(Author.Text, 7, 4)	Extracts up to four characters, starting with the seventh character, from the string.	"Cumm"
Mid(Author.Text, 7)	Extracts all characters, starting with the seventh character, from the string.	"Cummings"
Right(Author.Text, 5)	Extracts up to five characters from the end of the string.	"mings"

Single-column table

Each example in this section extracts strings from the **Address** column of this data source, named **People**, and returns a single-column table that contains the results:

Expand table

Name	Address
"Jean"	"123 Main St NE"

Name	Address
"Fred"	"789 SW 39th #3B"

Expand table

Formula	Description	Result
Left(ShowColumns(People, "Address"), 8)	Extracts the first eight characters of each string.	A single-column table with a `Value` column containing the following values: "123 Main", "789 SW 3"
Mid(ShowColumns(People, "Address"), 5, 7)	Extracts the middle seven characters of each string, starting with the fifth character.	A single-column table with a `Value` column containing the following values: "Main St", "SW 39th"
Right(ShowColumns(People, "Address"), 7)	Extracts the last seven characters of each string.	A single-column table with a `Value` column containing the following values: "n St NE", "9th #3B"

Step-by-step example

1. Import or create a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.
2. Set the **Text** property of the lower label in the gallery to this function:

   Right(ThisItem.ProductName, 3)

   The label shows the last three characters of each product name.

Feedback

Was this page helpful? Yes No

Provide product feedback

Len function

Article • 04/20/2025

Applies to: Canvas apps Cards Copilot Studio Desktop flows Dataverse formula columns Model-driven apps Power Platform CLI Dataverse functions Power Pages

Returns the length of a string of text.

Description

If you specify a single string as the argument, the return value is the length as a number. If you specify a single-column table that contains strings, the return value is a single-column table with a **Value** column that contains the length of each string. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If you specify a blank string, **Len** returns 0.

Syntax

Len(*String*)

- *String* - Required. The string to measure.

Len(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of strings to measure.

Examples

Single string

For the examples in this section, the data source is a text-input control that's named **Author** and that contains the string "E. E. Cummings".

Expand table

Formula	Description	Result
Len(Author.Text)	Measures the length of the string in the **Author** control.	14
Len("")	Measures the length of an empty string.	0

Single-column table

For the first example in this section, the data source is named **People** and contains this data:

Expand table

Name	Address
"Jean"	"123 Main St NE"
"Fred"	"789 SW 39th #3B"

Expand table

Formula	Description	Result
Len(ShowColumns(People, "Address"))	In the **Address** column of the **People** table: • Measures the length of each string. • Returns a single-column table that contains the length of each string.	A single-column table with a `Value` column containing the following values: 14, 15
Len(["Hello", "to the", "World", ""])	In the **Value** column of the inline table: • Measures the length of each string. • Returns a single-column table that contains the length of each string.	A single-column table with a `Value` column containing the following values: 5, 6, 5, 0

Abs, Exp, Ln, Power, Log, and Sqrt functions

Article • 04/20/2025

Expand table

Functions	Applies to
Abs **Exp** **Ln** **Power** **Sqrt**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Log	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Calculates absolute values, logarithms, square roots, and the results of raising *e* or any number to specified powers.

Description

The **Abs** function returns the non-negative value of its argument. If a number is negative, **Abs** returns the positive equivalent.

The **Exp** function returns *e* raised to the power of its argument. The transcendental number *e* begins 2.7182818...

The **Ln** function returns the natural logarithm (base *e*) of its argument.

The **Power** function returns a number raised to a power. It's equivalent to using the ^ operator.

The **Log** function returns the logarithm of its first argument in the base specified by its second argument (or 10 if not specified).

The **Sqrt** function returns the number that, when multiplied by itself, equals its argument.

If you pass a single number, the return value is a single result based on the function called. If you pass a single-column table that contains numbers, the return value is a single-column table of results in a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined valued, the result is *blank*. Which can happen with square roots and logarithms of negative numbers.

Syntax

Abs(*Number*)
Exp(*Number*)
Ln(*Number*)
Sqrt(*Number*)

- *Number* - Required. Number to operate on.

Power(*Base*, *Exponent*)

- *Base* - Required. Base number to raise.
- *Exponent* - Required. The exponent to which the base number is raised.

Log(*Number*, *Base*)

- *Number* - Required. Number to calculate the logarithm.
- *Base* - Optional. The base of the logarithm to calculate. By default, 10 (when not specified).

Abs(*SingleColumnTable*)
Exp(*SingleColumnTable*)
Ln(*SingleColumnTable*)
Sqrt(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Examples

Single number

Expand table

Formula	Description	Result
Abs(-55)	Returns the number without the negative sign.	55
Exp(2)	Returns *e* raised to the power of 2, or *e* * *e*.	7.389056...
Ln(100)	Returns the natural logarithm (base *e*) of the number 100.	4.605170...
Log(100)	Returns the logarithm in base 10 of the number 100.	2
Log(64, 2)	Returns the logarithm in base 2 of the number 64.	6

Formula	Description	Result
Power(5, 3)	Returns 5 raised to the power of 3, or 5 * 5 * 5.	125
Sqrt(9)	Returns the number that, when multiplied by itself, results in 9.	3

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains this data:

Expand table

Value
9
-4
2

Expand table

Formula	Description	Result
Abs(ValueTable)	Returns the absolute value of each number in the table.	A single-column table with a `Value` column containing the following values: 9, 4, 2
Exp(ValueTable)	Returns *e* raised to the power of each number in the table.	A single-column table with a `Value` column containing the following values: 8103.083927..., 0.018315..., 7.389056...
Ln(ValueTable)	Returns the natural logarithm of each number in the table.	A single-column table with a `Value` column containing the following values: 2.197224..., Blank(), 0.693147...
Sqrt(ValueTable)	Returns the square root of each number in the table	A single-column table with a `Value` column containing the following values: 3, Blank(), 1.414213...

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a **Label** control, and set its **Text** property to this formula:
   Sqrt(Value(Source.Text))

3. Type a number into **Source**, and confirm that the **Label** control shows the square root of the number that you typed.

SaveData, LoadData, and ClearData functions

06/18/2025

Applies to: Canvas apps

Saves and reloads a collection from the app host's storage.

Description

The **SaveData** function stores a collection for later use under a name.

The **LoadData** function reloads a collection by name that was previously saved with **SaveData**. You can't use this function to load a collection from another source.

The **ClearData** function clears the storage under a specific name or clears all storage associated with the app if no name is provided.

> **Note**
>
> - The name shared between **SaveData**, **LoadData**, and **ClearData** is a key, not a file name. It need not be complex as names are unique to each app and there is no danger of name conflict. The name must not contain any of these characters: `*".?:\<>|/`.
> - SaveData is limited to 1 MB of data for Power Apps running in Teams and in a web browser. There is no fixed limit for Power Apps running in a mobile player but there are practical limits discussed below.
> - Don't use **SaveData** to store sensitive data in the web since it'll be stored in plain text.

Use these functions to improve app-startup performance by:

- Caching data in the **App.OnStart** formula on a first run.
- Reloading the local cache on next runs.

You can also use these functions to add simple offline capabilities to your app.

You can't use these functions inside a browser when:

- Authoring the app in Power Apps Studio.

To test your app, run it in Power Apps Mobile on an iPhone or Android device.

These functions are limited by the amount of available app memory as they operate on an in-memory collection. Available memory can vary depending on factors such as:

- The device and operating system.
- The memory that the Power Apps player uses.
- Complexity of the app with screens and controls.

Test your app with expected scenarios on the type of devices you expect the app to run when storing large data. Expect to have between 30 MB and 70 MB of available memory generally.

These functions depend on the collection being implicitly defined with **Collect** or **ClearCollect**. You don't need to call **Collect** or **ClearCollect** to load data into the collection for defining it. It's a common case when using **LoadData** after a previous **SaveData**. All that is needed is the presence of these functions in a formula to implicitly define the structure of the collection. For more information, see creating and removing variables.

The loaded data will be appended to the collection. Use the **Clear** function before calling **LoadData** if you want to start with an empty collection.

Data security

Consider carefully the isolation and encryption of data stored with **SaveData** and decide if it's appropriate for your needs, especially if devices are shared by multiple users.

Data stored with **SaveData** is isolated from other Power Apps by the Power Apps players. Data is stored based on the app's App ID, automatically isolating the **SaveData** name space between Power Apps.

The operating system and browser is responsible for isolating data between Power Apps and other apps on a device and with websites. For example, the operating system is responsible for isolating data stored in Microsoft Outlook from data stored in Power Apps, and also isolating that data from websites such as Bing.com or PowerApps.com. The operating system's built in app sandbox facilities are used for **SaveData** storage which is usually not accessible to or hidden from the user.

When using the same app, the operating system and browser is also responsible for isolating the data between different operating system level users. For example, if two different users share a computer and use two different Windows login credentials, the operating system is responsible for isolating data between the two Windows users.

Data may or may not be isolated between different Power Apps users if the operating system user is the same. Not every Power Apps player treats this the same way. For example, while

logged in as the same Windows user, in the Power Apps player, the user signs out of Power Apps and signs in as a different Power Apps user. Data stored in an app before the change of Power Apps user, may be accessible to the second Power Apps user within the same app. The data may also be removed and the first Power Apps user may no longer be able to access it. The behavior varies between Power Apps players.

The operating system may also encrypt the data or you can use a mobile device management tool such as Microsoft Intune. Data stored when playing an app in a web browser is not encrypted.

Syntax

SaveData(*Collection*, *Name*)
LoadData(*Collection*, *Name* [, *IgnoreNonexistentFile*])

- *Collection* - Required. Collection to be stored or loaded.
- *Name* - Required. Name of the storage. The name must be same to save and load same set of data. The name space isn't shared with other apps. Names must not contain any of these characters: `*".?:\<>|/`.
- *IgnoreNonexistentFile* - Optional. A Boolean value indicating what to do if the file doesn't already exist. Use *false* (default) to return an error and *true* to suppress the error.

ClearData([*Name*])

- *Name* - Optional. Name of the storage previously saved with **SaveData**. If *Name* is not provided, all storage associated with the app is cleared.

Examples

Expand table

Formula	Description	Result
SaveData(LocalCache, "MyCache")	Save the **LocalCache** collection to the user's device under the name "MyCache", suitable for **LoadData** to retrieve later.	Data is saved to the app host under the name "MyCache".
LoadData(LocalCache, "MyCache")	Loads the **LocalCache** collection from the user's device under the name "MyCache", previously stored with a call to **SaveData**.	Data is loaded from the app host under the name "MyCache".
ClearData("MyCache")	Clears the storage under the name "MyCache". Any data stored under this name will no longer be available through **LoadData**.	Data is removed from the app host under the name "MyCache".

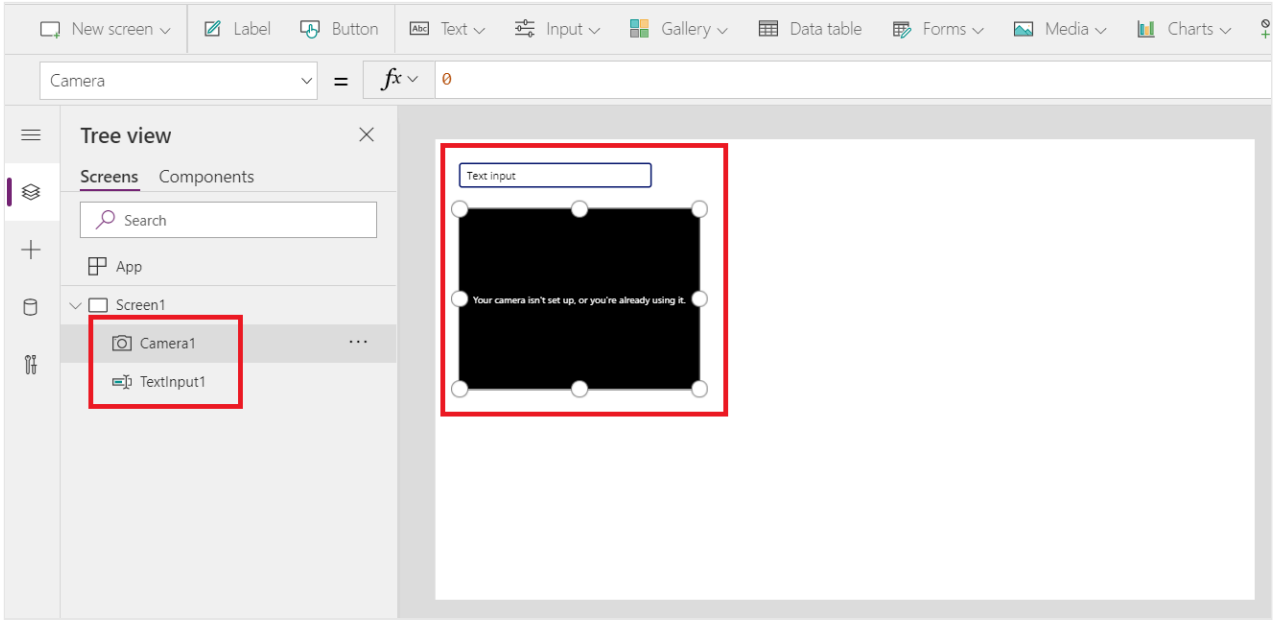
Formula	Description	Result
ClearData()	Clear all storage associated with this app. Data stored by other apps is not affected.	All data is removed from the app host.

Simple offline example

Following simple example captures and stores the names and pictures of everyday items while offline. It stores the information in the device's local storage for later use. This allows the app to be closed or the device to restart without losing data.

> **Note**
>
> This example uses a camera control to capture images. Since **SaveData** is limited to 1 MB of data when running in Teams or a web browser, this example will not work with more than a few images. Also, depending on the camera, it may not work with even one image. Use a device to work through this full example, or remove the camera control and picture part of this example to run in Teams or in a web browser.

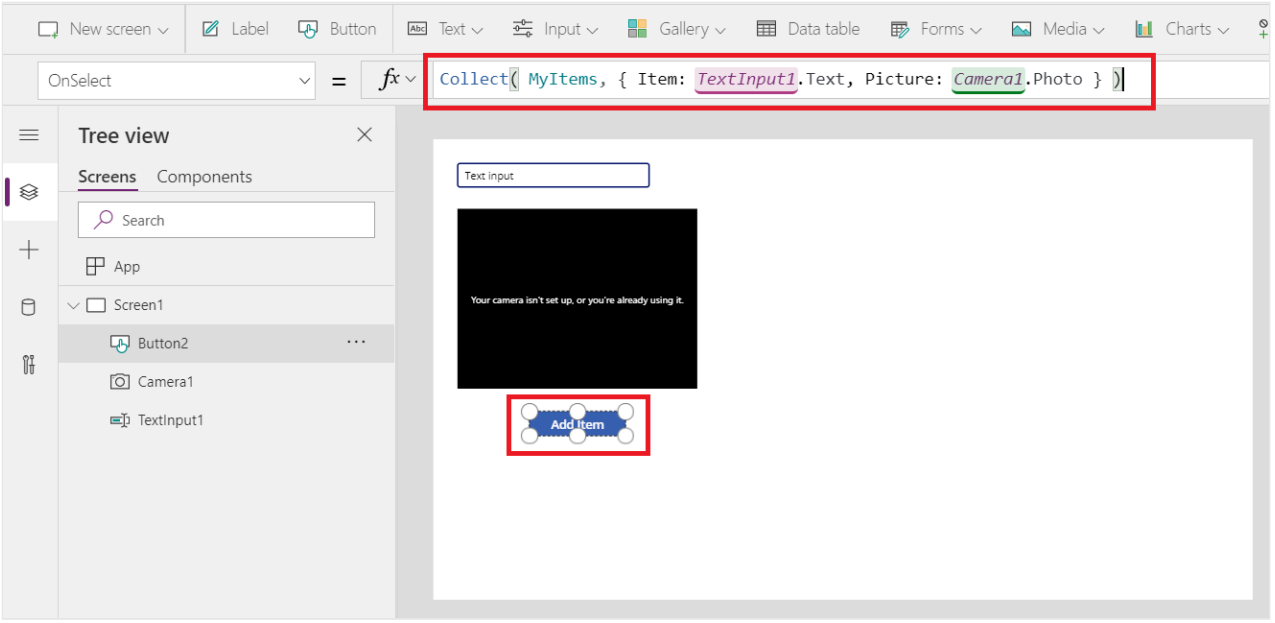
1. Create a blank canvas app with a tablet layout. For more details, read creating an app from a template and select **Tablet layout** under **Blank app**.
2. Add a **Text input** control and a **Camera** control and arrange them roughly as shown:



3. Add a **Button** control.
4. Double-click the button control to change the button text to **Add Item** (or modify the **Text** property).

5. Set the **OnSelect** property of the button control to this formula that will add an item to our collection:

```
Power Fx

Collect( MyItems, { Item: TextInput1.Text, Picture: Camera1.Photo } )
```

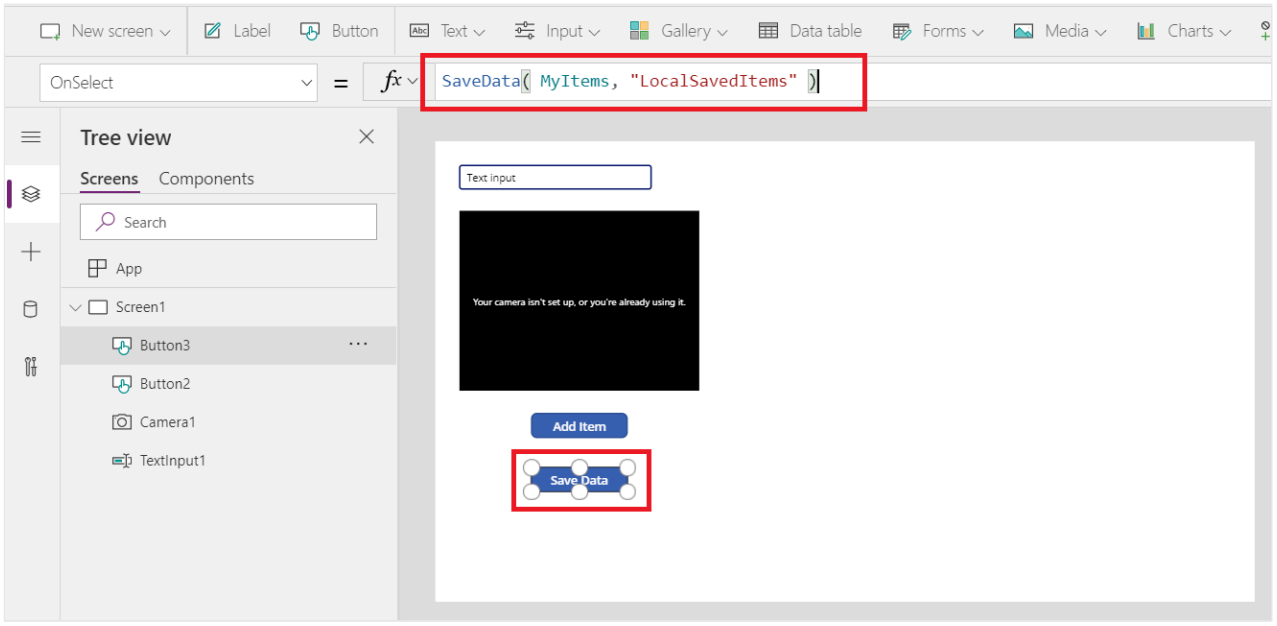


6. Add another **Button** control.

7. Double-click the button control to change the button text to **Save Data** (or modify the **Text** property).

8. Set the **OnSelect** property of the button control to this formula in order to save our collection to the local device:

```
Power Fx

SaveData( MyItems, "LocalSavedItems" )
```



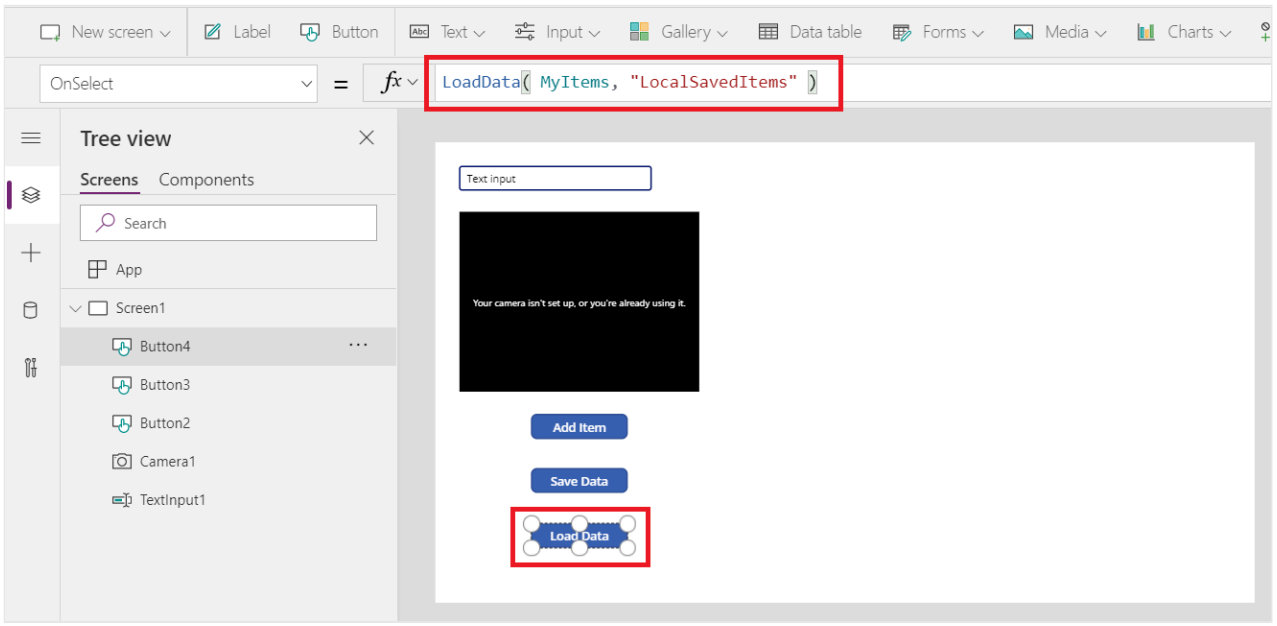
It's tempting to test the button as it doesn't affect anything. But you'll only see an error as you're authoring in a web browser. Save the app first and open on a device before you follow the next steps to test this formula:

9. Add a third **Button** control.

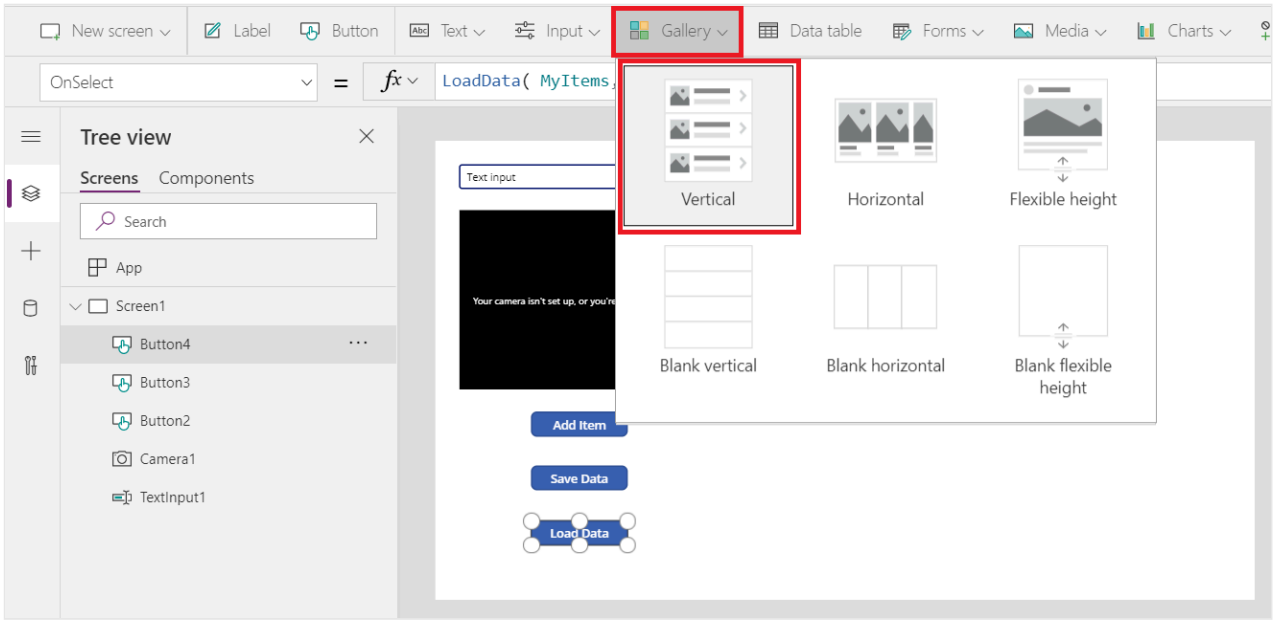
10. Double-click the button control to change the button text to **Load Data** (or modify the **Text** property).

11. Set the **OnSelect** property of the button control to this formula in order to load our collection from the local device:

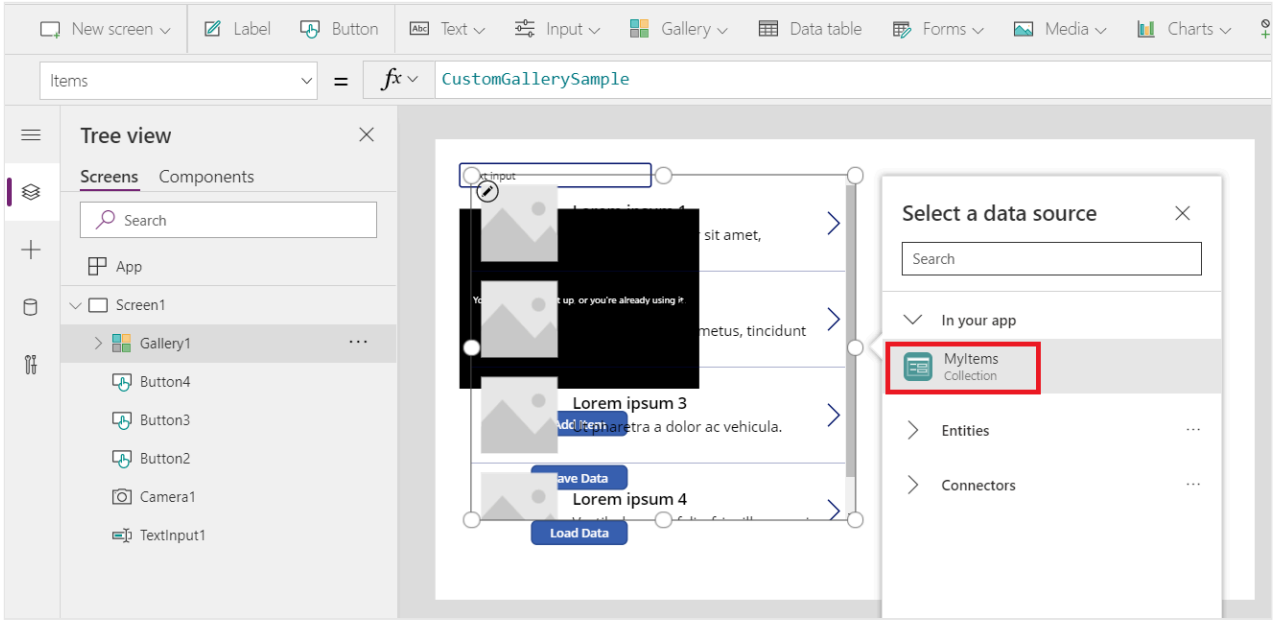
```
Power Fx

LoadData( MyItems, "LocalSavedItems" )
```



12. Add a Gallery control with a Vertical layout that includes a picture and text areas:

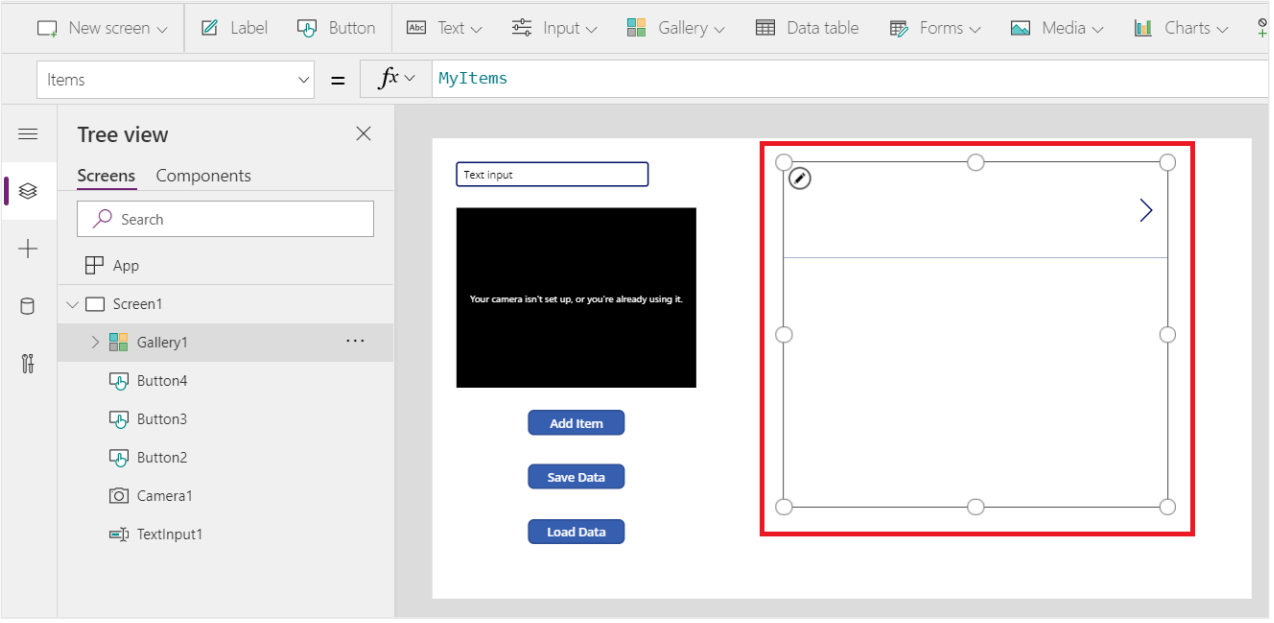


13. When prompted, select the **MyItems** collection as the data source for this gallery. This will set the **Items** property of the **Gallery** control:



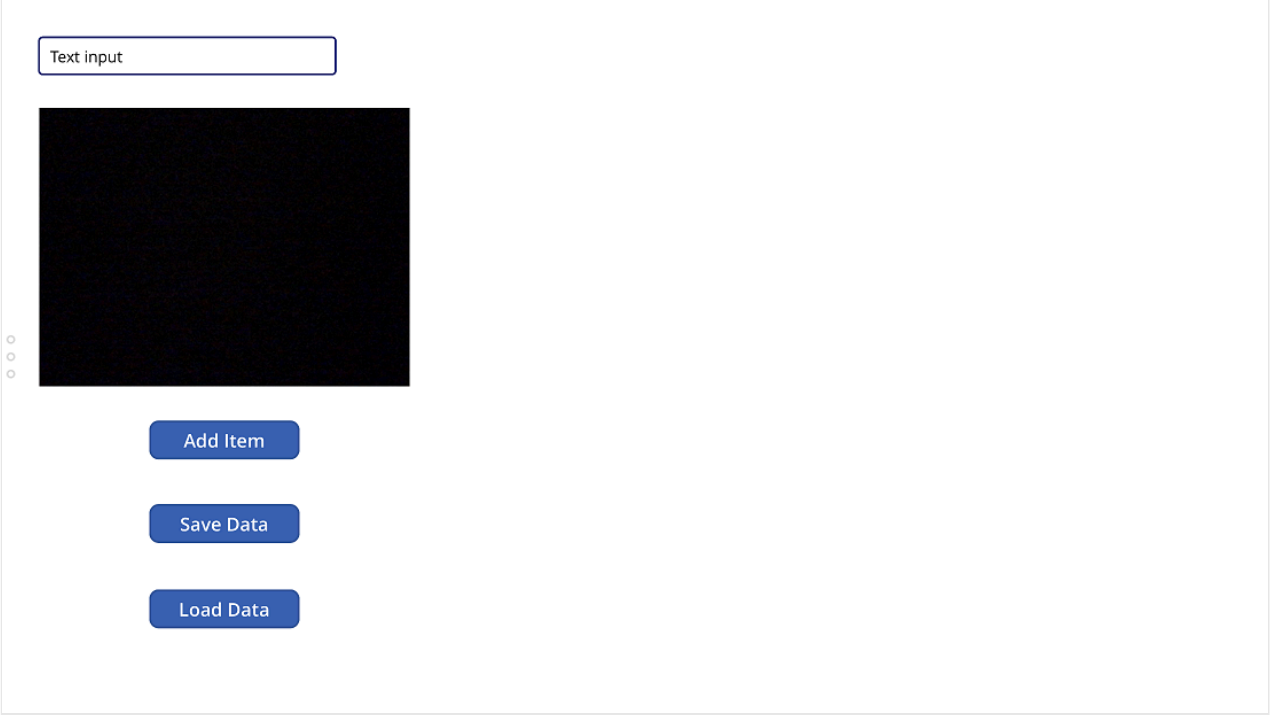
The image control in the gallery template should default its **Image** property to **ThisItem.Picture** and the label controls should both default their **Text** properties to **ThisItem.Item**. Check these formulas if after adding items in the following steps you don't see anything in the gallery.

14. Position the control to the right of the other controls:



15. Save your app. If it's the first time it has been saved, there's no need to publish it. If it's not the first time, publish the app after you save.

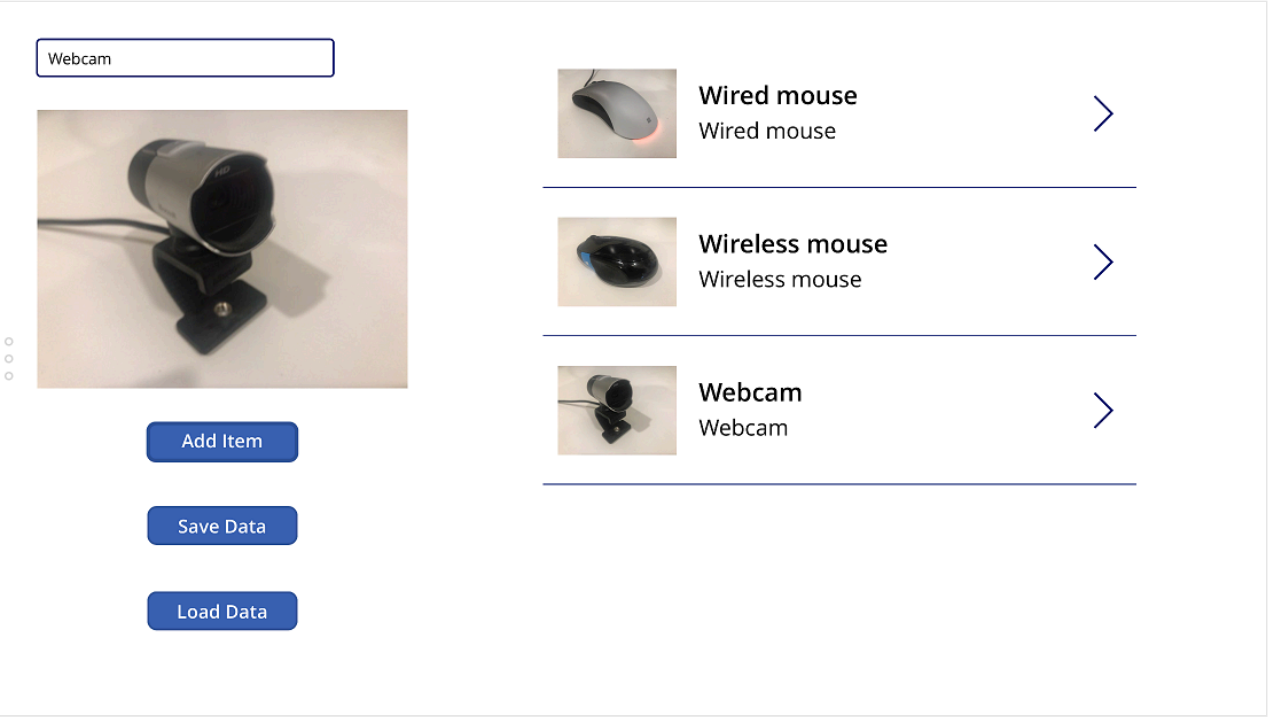
16. Open your app on a device such as a phone or tablet. **SaveData** and **LoadData** can't be used in Studio or in a web browser. Refresh your app list if you don't see your app immediately, it can take a few seconds for the app to appear on your device. Signing out and back in to your account can help too.



Once your app has been downloaded, you can disconnect from the network and run the app offline.

17. Enter the name and take a picture of an item.

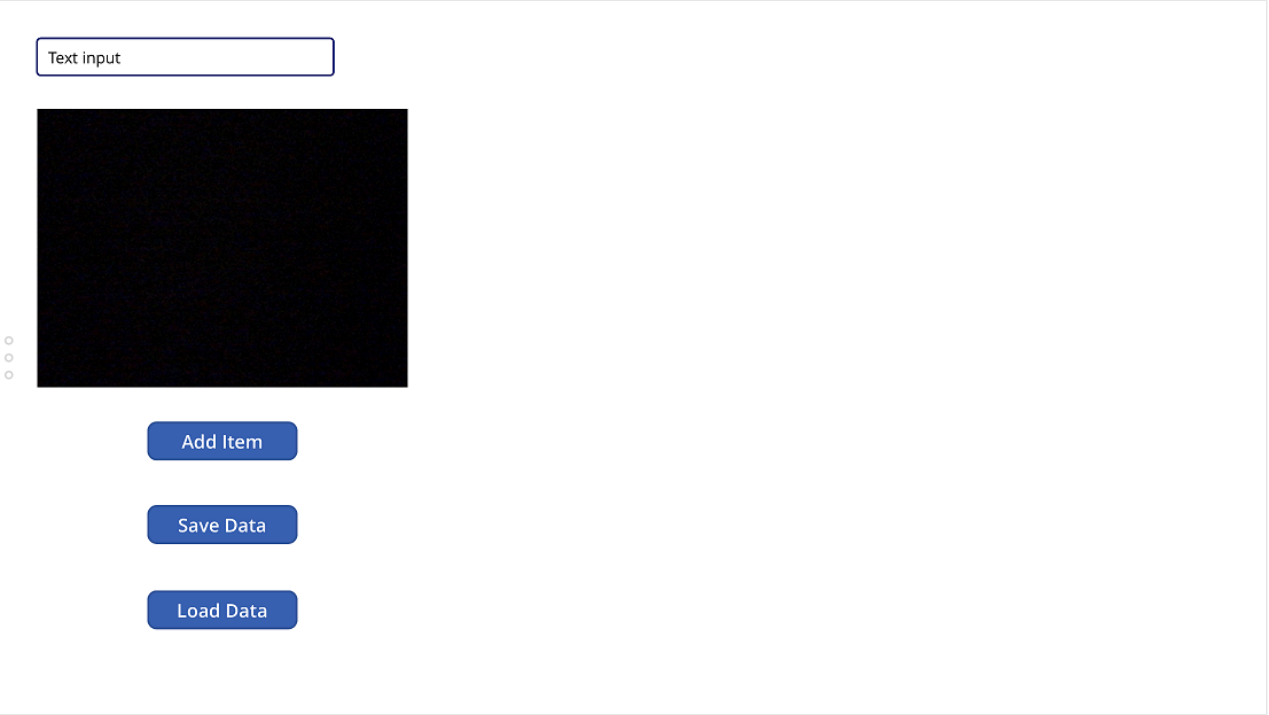
18. Select the **Add Item** button. Repeat adding items a couple of times to load up your collection.



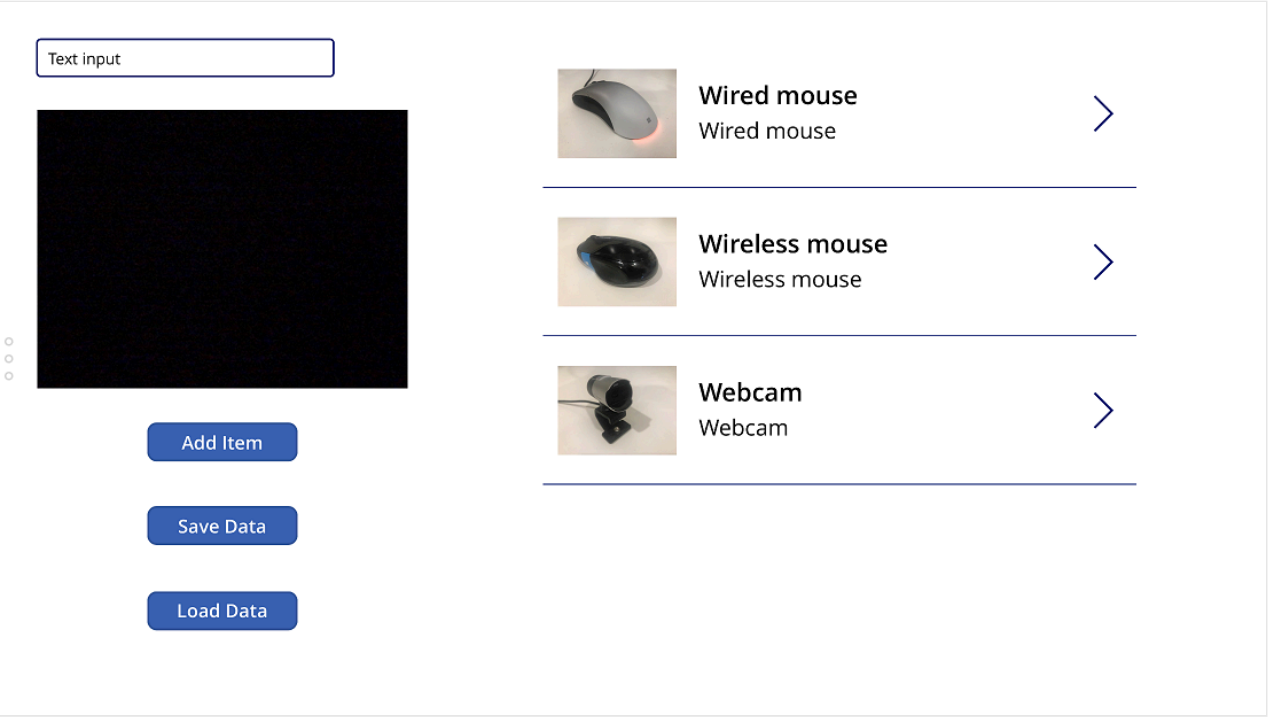
19. Select the **Save Data** button. This will save the data in your collection to your local device.

20. Close the app. Your collection in memory will be lost including all item names and pictures, but they'll still be there in the device's storage.

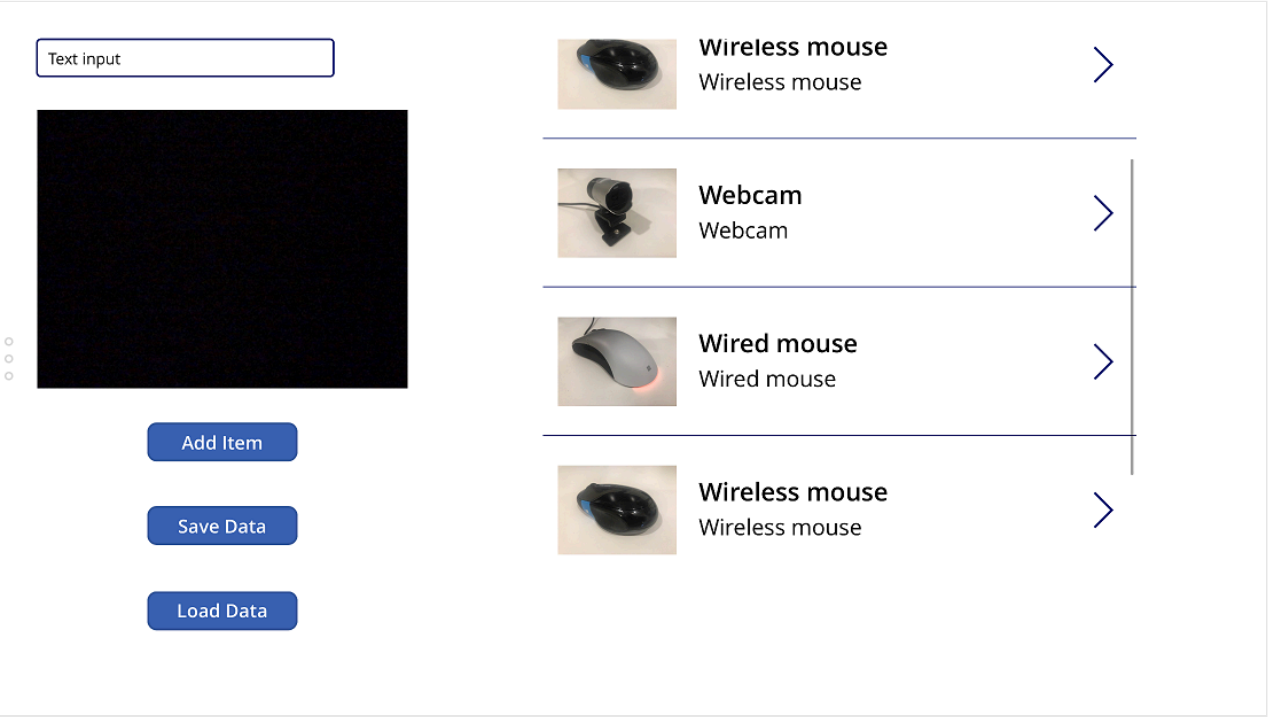
21. Launch the app again. The collection in memory will again show as empty in the gallery.



22. Select the **Load Data** button. The collection will be repopulated from the stored data on your device and your items will be back in the gallery. The collection was empty before this button calls the **LoadData** function; there was no need to call **Collect** or **ClearCollect** before loading the data from storage.



23. Select the **Load Data** button again. The stored data will be appended to the end of the collection and a scroll bar will appear on the gallery. If you would like to replace rather than append, use the **Clear** function first to clear out the collection before calling the **LoadData** function.



More advanced offline example

For a detailed example, see the article on simple offline capabilities.

Acceleration, App, Compass, Connection, and Location signals in Power Apps

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Returns information about the app's environment, such as where the user is located in the world and which screen is displayed.

Description and syntax

Signals are values that can change at any time, independent of how the user may be interacting with the app. Formulas that are based on signals automatically recalculate as these values change.

Signals typically return a record of information. You can use and store this information as a record, or you can extract individual properties by using the . operator.

> **Note**
>
> The **Acceleration** and **Compass** functions return accurate values in a native player such as on iOS or Android, but those functions return zero values as you create or modify an app in the browser.

Acceleration

The **Acceleration** signal returns the device's acceleration in three dimensions relative to the device's screen. Acceleration is measured in *g* units of 9.81 m/second^2 or 32.2 ft/second^2 (the acceleration that the Earth imparts to objects at its surface due to gravity).

Expand table

Property	Description
Acceleration.X	Right and left. Right is a positive number.
Acceleration.Y	Forward and back. Forward is a positive number.

Property	Description
Acceleration.Z	Up and down. Up is a positive number.

App

Among other properties, the **App** object includes a signal that indicates which screen is showing.

Expand table

Property	Description
App.ActiveScreen	Screen that's showing. Returns a screen object, which you can use to reference properties of the screen or compare to another screen to determine which screen is showing. You can use the **Back** or **Navigate** function to change the screen that's showing.

More information: **App** object documentation.

Compass

The **Compass** signal returns the compass heading of the top of the screen. The heading is based on magnetic north.

Expand table

Property	Description
Compass.Heading	Heading in degrees. Returns a number 0 to 360, and 0 is north.

Connection

The **Connection** signal returns the information about the network connection. When on a metered connection, you may want to limit how much data you send or receive over the network. Or, if your app is enabled for offline, you may want to display different information depending on the current sync state.

Expand table

Property	Description
Connection.Connected	Returns a Boolean **true** or **false** value that indicates whether the device is connected to a network.
Connection.Metered	Returns a Boolean **true** or **false** value that indicates whether the connection is metered.
Connection.Sync	Returns an enum named **ConnectionSync** which holds the current synchronization state for an app that is enabled for offline.

Below are the enums values available for Connection.Sync.

Expand table

Enum value	Description
Connected	Your device is connected to the network and your app is ready to work offline
ConnectedWithWarning	Your device is connected with a warning.
ConnectedPendingUpsync	Some data on your device must be synchronized with the service.
ConnectedError	There was an error on the last synchronization attempt.
ConnectedRefresh	Your app is currently synchronizing data with the service.
NotConnected	Your device isn't connected to the service.
NotConnectedWithWarning	The last synchronization warning.
NotConnectedPendingUpsync	There are updates on your device waiting to be synchronized.
NotConnectedSyncError	The last synchronization error.

Location

The **Location** signal returns the location of the device based on the Global Positioning System (GPS) and other device information, such as cell-tower communications and IP address.

When a user accesses the location information for the first time, the device may prompt that user to allow access to this information.

As the location changes, dependencies on the location will continuously recalculate, which will consume power from the device's battery. To conserve battery life, you can

use the **Enable** and **Disable** functions to turn location updates on and off. Location is automatically turned off if the displayed screen doesn't depend on location information.

Expand table

Property	Description
Location.Altitude	Returns a number that indicates the altitude, measured in meters, above sea level.
Location.Latitude	Returns a number, from –90 to 90, that indicates the latitude, as measured in degrees from the equator. A positive number indicates a location that's north of the equator.
Location.Longitude	Returns a number, from –180 to 180, that indicates the longitude, as measured in degrees from Greenwich, England. A positive number indicates a location that's east of Greenwich.

Examples

In a baseball field, a pitcher throws a phone from the pitcher's mound to a catcher at home plate. The phone is lying flat with respect to the ground, the top of the screen is pointed at the catcher, and the pitcher adds no spin. At this location, the phone has cellular network service that's metered but no WiFi. The **PlayBall** screen is displayed. The app is enabled for offline and is connected to the service.

Expand table

Formula	Description	Result
Location.Latitude	Returns the latitude of the current location. The field is located at map coordinates 47.591 N, 122.333 W.	47.591 The latitude will change continuously as the ball moves between the pitcher and the catcher.
Location.Longitude	Returns the longitude of the current location.	122.333 The longitude will change continuously as the ball moves between the pitcher and the catcher.
Location	Returns the latitude and longitude of the current location, as a record.	{ Latitude: 47.591, Longitude: 122.333 }

Formula	Description	Result
Compass.Heading	Returns the compass heading of the top of the screen. At this field, home plate is roughly southwest from the pitcher's mound.	230.25
Acceleration.X	Returns the acceleration of the device side to side. The pitcher is throwing the phone straight ahead with respect to the screen's top, so the device isn't accelerating side to side.	0
Acceleration.Y	Returns the acceleration of the device front to back. The pitcher initially gives the device a large acceleration when throwing the device, going from 0 to 90 miles per hour (132 feet per second) in half a second. After the device is in the air, ignoring air friction, the device doesn't accelerate further. The device decelerates when the catcher catches it, bringing it to a stop.	8.2, while the pitcher throws the device. 0, while the device is in the air. -8.2, as the catcher catches the device.
Acceleration.Z	Returns the acceleration of the device top to bottom. While in the air, the device experiences the effects of gravity.	0, before the pitcher throws the device. 1, while the device is in the air. 0, after the catcher catches the device.
Acceleration	Returns the acceleration as a record.	{ X: 0, Y: 264, Z: 0 } as the pitcher throws the device.
Connection.Connected	Returns a Boolean value that indicates whether the device is connected to a network	**true**
Connection.Metered	Returns a Boolean value that indicates whether the connection is metered	**true**
Connection.Sync	Returns an enum value that indicates current sync state.	**ConnectionSync.Connected**
App.ActiveScreen = PlayBall	Returns a Boolean value that indicates whether **PlayBall** is	**true**

Formula	Description	Result
	displayed.	
App.ActiveScreen.Fill	Returns the background color for the displayed screen.	**Color.Green**

Filter, Search, and LookUp functions

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Finds one or more records in a table.

Watch this video to learn how to use **Filter**, **Search**, and **LookUp** functions:

https://learn-video.azurefd.net/vod/player?id=55b2c144-b5ac-4d66-b398-fea03eb21f17&locale=en-us&embedUrl=%2Fpower-platform%2Fpower-fx%2Freference%2Ffunction-filter-lookup

> **Note**
>
> PAC CLI pac power-fx commands do not support the **Search** function.

Description

The **Filter** function finds records in a table that satisfy a formula. Use **Filter** to find a set of records that match one or more criteria and discard those records that don't.

The **LookUp** function finds the first record in a table that satisfies a formula. Use **LookUp** to find a single record that matches one or more criteria.

For both, the formula is evaluated for each record of the table. Records that result in *true* are included in the result. Besides the normal formula operators, you can use the **in** and **exactin** operators for substring matches.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

The **Search** function finds records in a table that contain a string in one of their columns. The string might occur anywhere within the column; for example, searching for "rob" or "bert" would find a match in a column that contains "Robert". Searching is case-insensitive. Unlike **Filter** and **LookUp**, the **Search** function uses a single string to match instead of a formula.

Filter and **Search** return a table that contains the same columns as the original table and the records that match the criteria. **LookUp** returns only the first record found, after applying a

formula to reduce the record to a single value. If no records are found, **Filter** and **Search** return an empty table, and **LookUp** returns *blank*.

Tables are a value in Power Apps, just like a string or number. They can be passed to and returned from functions. **Filter**, **Search**, and **LookUp** don't modify a table. Instead, they take a table as an argument and return a table, a record, or a single value from it. See working with tables for more details.

Delegation

When possible, Power Apps will delegate filter and sort operations to the data source and page through the results on demand. For example, when you start an app that shows a **Gallery** control filled with data, only the first set of records will be initially brought to the device. As the user scrolls, additional data is brought down from the data source. The result is a faster start time for the app and access to very large data sets.

However, delegation may not always be possible. Data sources vary on what functions and operators they support with delegation. If complete delegation of a formula isn't possible, the authoring environment will flag the portion that can't be delegated with a warning. When possible, consider changing the formula to avoid functions and operators that can't be delegated. The delegation list details which data sources and operations can be delegated.

If delegation is not possible, Power Apps will pull down only a small set of records to work on locally. Filter and sort functions will operate on a reduced set of records. What is available in the **Gallery** may not be the complete story, which could be confusing to users.

See the delegation overview for more information.

Syntax

Filter(Table*, *Formula1* [, *Formula2*, ...])

- *Table* - Required. Table to search.
- *Formula(s)* - Required. The formula by which each record of the table is evaluated. The function returns all records that result in **true**. You can reference columns within the table. If you supply more than one formula, the results of all formulas are combined with the **And** function.

Search(Table*, *SearchString*, *Column1* [, *Column2*, ...])

- *Table* - Required. Table to search.

- *SearchString* - Required. The string to search for. If *blank* or an empty string, all records are returned.
- *Column(s)* - Required. The names of columns within *Table* to search. If *SearchString* is found within the data of any of these columns as a partial match, the full record will be returned.

> **Note**
>
> In Power Apps prior to version 3.24042, column names for the **Search** function were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

LookUp(Table*, *Formula* [, *ReductionFormula*])

- *Table* - Required. Table to search. In the UI, the syntax is shown as *source* above the function box.
- *Formula* - Required. The formula by which each record of the table is evaluated. The function returns the first record that results in **true**. You can reference columns within the table. In the UI, the syntax is shown as *condition* above the function box.
- *ReductionFormula* - Optional. This formula is evaluated over the record that was found, and then reduces the record to a single value. You can reference columns within the table. If you don't use this parameter, the function returns the full record from the table. In the UI, the syntax is shown as *result* above the function box.

Examples

The following examples use the **IceCream** data source:

Flavor	Quantity	OnOrder
"Chocolate"	100	75
"Vanilla"	200	50
"Strawberry"	300	0
"Mint Chocolate"	60	100

Expand table

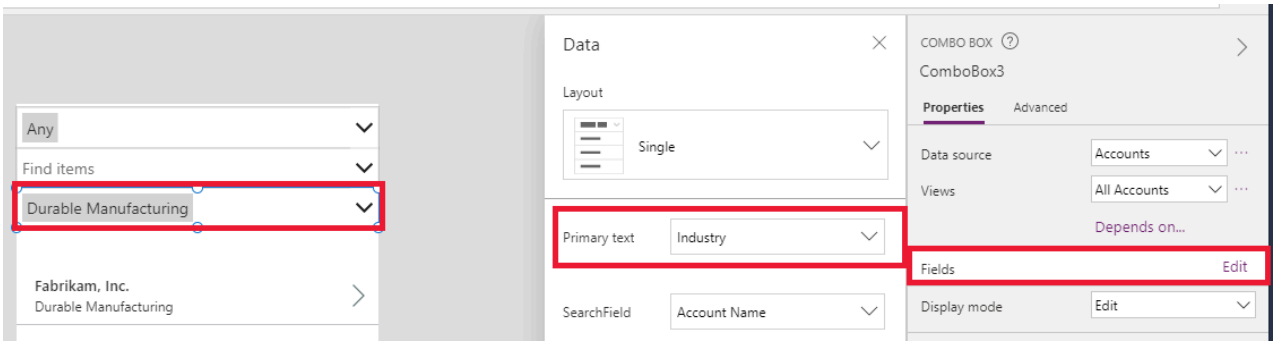
Formula	Description	Result
Filter(IceCream, OnOrder > 0)	Returns records where **OnOrder** is greater than zero.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Filter(IceCream, Quantity + OnOrder > 225)	Returns records where the sum of **Quantity** and **OnOrder** columns is greater than 225.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Strawberry"</td><td>300</td><td>0</td></tr></table>
Filter(IceCream, "chocolate" in Lower(Flavor))	Returns records where the word "chocolate" appears in the **Flavor** name, independent of uppercase or lowercase letters.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Filter(IceCream, Quantity < 10 && OnOrder < 20)	Returns records where the **Quantity** is less than 10 and **OnOrder** is less than 20. No records match these criteria, so an empty table is returned.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr></table>
Search(IceCream, "choc", Flavor)	Returns records where the string "choc" appears in the **Flavor** name, independent of uppercase or lowercase letters.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Search(IceCream, "", Flavor)	Because the search term is empty, all records are returned.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Strawberry"</td><td>300</td><td>0</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
LookUp(IceCream, Flavor = "Chocolate", Quantity)	Searches for a record with **Flavor** equal to "Chocolate", of which there's one. For the first record that's found, returns the **Quantity** of that record.	100
LookUp(IceCream, Quantity > 150, Quantity + OnOrder)	Searches for a record with **Quantity** greater than 150, of which there are multiple. For the first record that's found, which is "Vanilla" **Flavor**, returns the sum of **Quantity** and **OnOrder** columns.	250
LookUp(IceCream, Flavor = "Pistachio", OnOrder)	Searches for a record with **Flavor** equal to "Pistachio", of which there are none. Because none is found, **Lookup** returns *blank*.	*blank*
LookUp(IceCream, Flavor = "Vanilla")	Searches for a record with **Flavor** equal to "Vanilla", of which there's one. Since no reduction formula was supplied, the entire record is returned.	{ Flavor: "Vanilla", Quantity: 200, OnOrder: 75 }

Filtering with choice columns

The following example uses the **Account** table in Microsoft Dataverse as data source. This example shows how to **Filter** list of accounts based on selected Combo box control values:

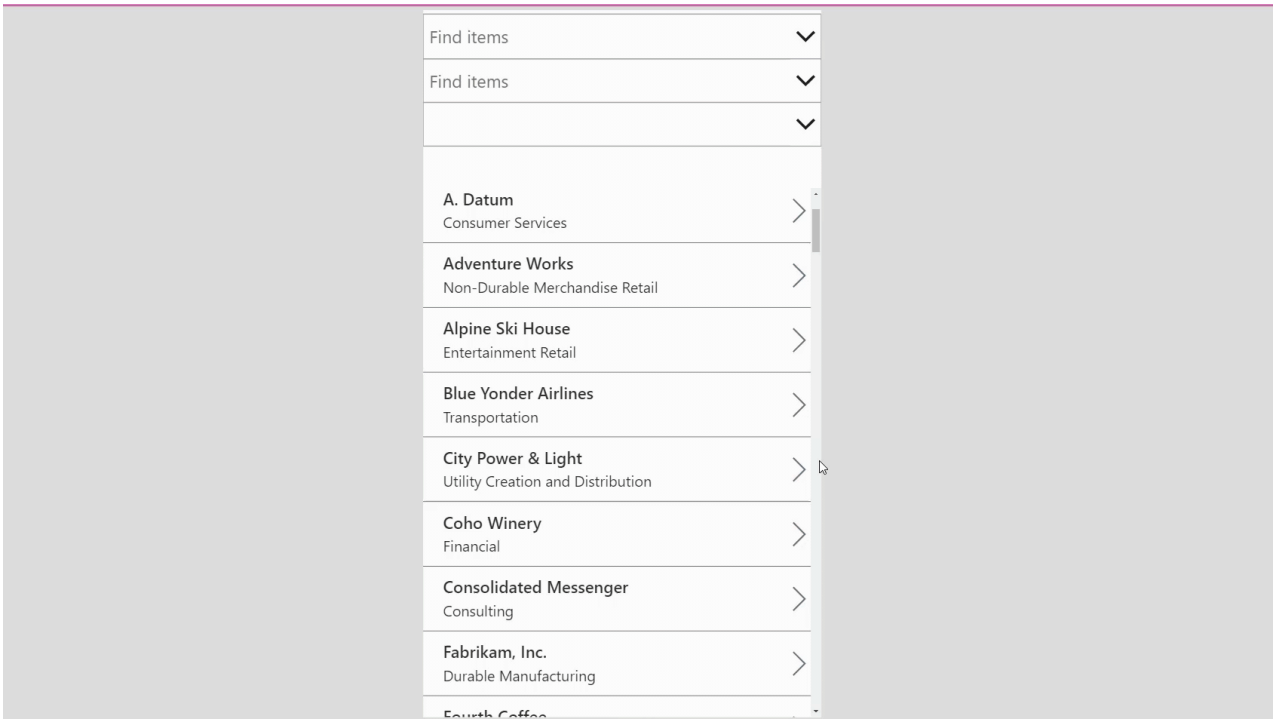
Step by step

1. Open a blank app.
2. Add a new screen by selecting the **New Screen** option.
3. On the **Insert** tab, select **Gallery** and then select **Vertical**.
4. On the **Properties** tab of the right-hand pane, open **Data Source** and then select **Accounts**.
5. (Optional) In the **Layout** list, select different options.
6. On the **Insert** tab, select **Input** and then select **Combo box**. Repeat the step to add two more combo box controls.
7. For each combo box control, on the **Properties** tab of the right-hand pane, open **Data Source** and then select **Accounts**. Select **Edit** next to **Fields** option and then select the **Primary text** and **SearchField** values. The **Primary text** should be the choices column you want to add to the combo box. Repeat the step for other two combo box controls.



8. Now select **Gallery** control and set the **Items** property to the following formula:

```
Filter(Accounts,
 'Industry' = ComboBox3.Selected.Industry Or
IsBlank(ComboBox3.Selected.Industry),
 'Relationship Type' = ComboBox2.Selected.'Relationship Type' Or
   IsBlank(ComboBox2.Selected.'Relationship Type'),
 'Preferred Method of Contact' = ComboBox1.Selected.'Preferred Method of
Contact' Or
   IsBlank(ComboBox1.Selected.'Preferred Method of Contact'))
```



Search user experience

The following examples use the **IceCream** data source:

In many apps, you can type one or more characters into a search box to filter a list of records in a large data set. As you type, the list shows only those records that match the search criteria.

The examples in the rest of this article show the results of searching a list, named **Customers**, that contain this data:

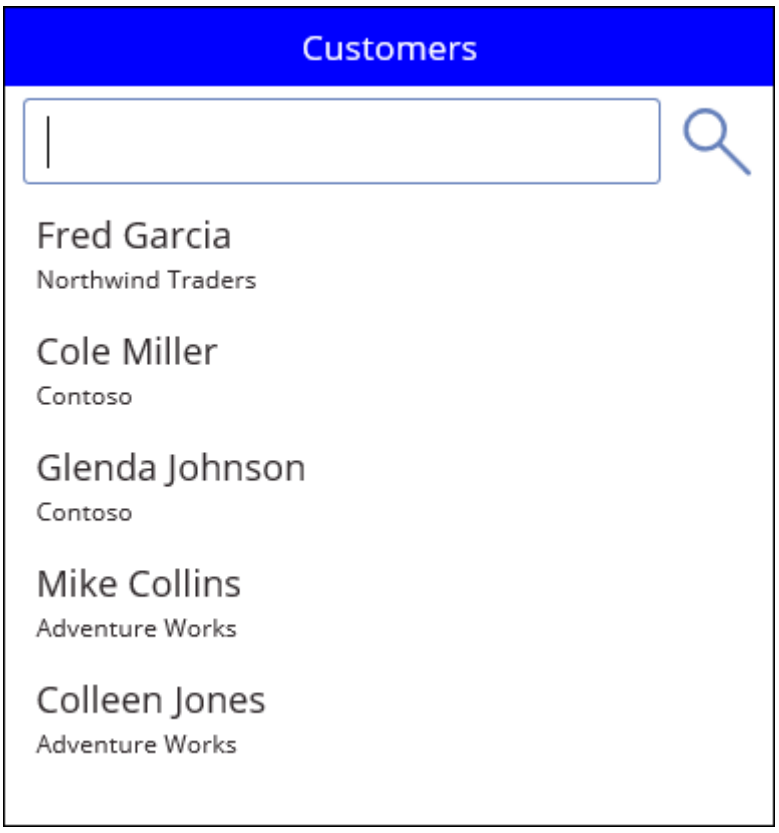
Name	Company
"Fred Garcia"	"Northwind Traders"
"Cole Miller"	"Contoso"
"Glenda Johnson"	"Contoso"
"Mike Collins"	"Adventure Works"
"Colleen Jones"	"Adventure Works"

To create this data source as a collection, create a **Button** control and set its **OnSelect** property to this formula:

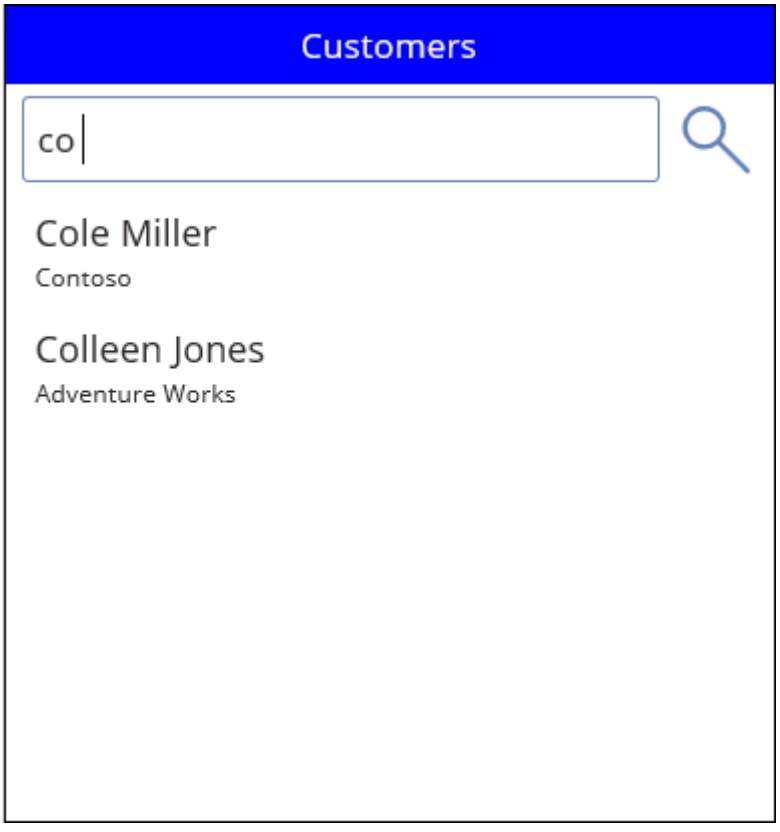
ClearCollect(Customers, Table({ Name: "Fred Garcia", Company: "Northwind Traders" }, { Name: "Cole Miller", Company: "Contoso" }, { Name: "Glenda Johnson", Company: "Contoso" }, { Name: "Mike Collins", Company: "Adventure Works" }, { Name: "Colleen Jones", Company: "Adventure Works" }))

As in this example, you can show a list of records in a **Gallery control** at the bottom of a screen. Near the top of the screen, you can add a **Text input** control, named **SearchInput**, so

that users can specify which records interest them.

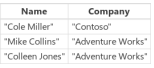


As the user types characters in **SearchInput**, the results in the gallery are automatically filtered. In this case, the gallery is configured to show records for which the name of the customer (not the name of the company) starts with the sequence of characters in **SearchInput**. If the user types **co** in the search box, the gallery shows these results:



To filter based on the **Name** column, set the **Items** property of the gallery control to one of these formulas:

Expand table

Formula	Description	Result
Filter(Customers, StartsWith(Name, SearchInput.Text))	Filters the **Customers** data source for records in which the search string appears at the start of the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones** and **Cole Miller**. The gallery doesn't show **Mike Collins** because the **Name** column for that record doesn't start with the search string.	
Filter(Customers, SearchInput.Text in Name)	Filters the **Customers** data source for records in which the search string appears anywhere in the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones, Cole Miller,** and **Mike Collins** because the search string appears somewhere in the **Name** column of all of those records.	
Search(Customers, SearchInput.Text, Name)	Similar to using the **in** operator, the **Search** function searches for a match anywhere within the **Name** column of each record. You must enclose the column name in double quotation marks.	

You can expand your search to include the **Company** column and the **Name** column:

Expand table

Formula	Description	Result
`Filter(Customers, StartsWith(Name, SearchInput.Text) || StartsWith(Company, SearchInput.Text) )`	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column starts with the search string (for example, **co**). The || operator is *true* if either **StartsWith** function is *true*.	
`Filter(Customers, SearchInput.Text in Name || SearchInput. Text in Company)`	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it.	
`Search(Customers, SearchInput.Text, Name, Company)`	Similar to using the **in** operator, the **Search** function searches the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it. The **Search** function is easier to read and write	

Formula	Description	Result
	than **Filter** if you want to specify multiple columns and multiple **in** operators.	

Last updated on 02/06/2025

Lower, Upper, and Proper functions

Article • 04/20/2025

Expand table

Functions	Applies to
Lower **Upper**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Proper	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts letters in a string of text to all lowercase, all uppercase, or proper case.

Description

The **Lower**, **Upper**, and **Proper** functions convert the case of letters in strings.

- **Lower** converts any uppercase letters to lowercase.
- **Upper** converts any lowercase letters to uppercase.
- **Proper** converts the first letter in each word to uppercase if it's lowercase and converts any other uppercase letters to lowercase.

All three functions ignore characters that aren't letters.

If you pass a single string, the return value is the converted version of that string. If you pass a single-column table that contains strings, the return value is a single-column table of converted strings. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

Lower(*String*)
Upper(*String*)
Proper(*String*)

- *String* - Required. The string to convert.

Lower(*SingleColumnTable*)
Upper(*SingleColumnTable*)
Proper(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of strings to convert.

Examples

Single string

The examples in this section use a text-input control, named **Author**, as their data source. The control contains the string "E. E. CummINGS".

Expand table

Formula	Description	Result
Lower(Author.Text)	Converts any uppercase letters in the string to lowercase.	"e. e. cummings"
Upper(Author.Text)	Converts any lowercase letters in the string to uppercase.	"E. E. CUMMINGS"
Proper(Author.Text)	Converts the first letter of each word to uppercase if it's lowercase, and converts any other uppercase letters to lowercase.	"E. E. Cummings"

Single-column table

The examples in this section convert strings from the **Address** column of the **People** data source, which contains this data:

Expand table

Name	Address
"Jean"	"123 Main St NE"
"Fred"	"789 SW 39th #3B"

Each formula returns a single-column table that contains the converted strings.

Expand table

Formula	Description	Result
Lower(ShowColumns(People, "Address"))	Converts any letter that's lowercase to uppercase.	A single-column table with a `Value` column containing the

Formula	Description	Result
		following values: "123 main st ne", "789 sw 39th #3b"
Upper(ShowColumns(People, "Address"))	Converts any letter that's lowercase to uppercase.	A single-column table with a `Value` column containing the following values: "123 MAIN ST NE", "789 SW 39TH #3B"
Proper(ShowColumns(People, "Address"))	Converts any first letter of a word that's lowercase to uppercase, and converts any other letter that's uppercase to lowercase.	A single-column table with a `Value` column containing the following values: "123 Main St Ne", "789 Sw 39th #3b"

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a label, and set its **Text** property to this function:
   Proper(Source.Text)
3. Press F5, and then type **WE ARE THE BEST!** into the **Source** box.
   The label shows **We Are The Best!**

IsMatch, Match, and MatchAll functions

06/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests for a match or extracts parts of a text string based on a pattern.

Description

The **IsMatch** function tests whether a text string matches a pattern that can comprise ordinary characters, predefined patterns, or a regular expression. The **Match** and **MatchAll** functions return what was matched, including submatches.

Use **IsMatch** to validate what a user typed in a **Text input** control. For example, confirm whether the user entered a valid email address before the result is saved to your data source. If the entry doesn't match your criteria, add other controls that prompt the user to fix the entry.

Use **Match** to extract the first text string that matches a pattern and **MatchAll** to extract all text strings that match. Extract submatches to parse complex strings.

Match returns a record of information for the first match found, and **MatchAll** returns a table of records for every match found. The record or records contain:

Expand table

Column	Type	Description
named sub-match or sub-matches	Text	Each named submatch has its own column. Create a named submatch by using **(?<*name*>...)** in the regular expression. If a named submatch has the same name as one of the predefined columns, the submatch takes precedence, and a warning is generated. Rename the submatch to avoid this warning.
FullMatch	Text	All of the text string that was matched.
StartMatch	Number	The starting position of the match within the input text string. The first character of the string returns 1.
SubMatches, only if **MatchOptions.NumberedSubMatches** is used.	Single-column table of	The table of numbered submatches in the order in which they appear in the regular expression. Generally, named submatches are

Column	Type	Description
	Text (column **Value**)	easier to work with and are encouraged. Use the ForAll function or Index function to work with an individual sub-match. If no submatches are defined in the regular expression, this table will be present but empty.

These functions support MatchOptions. By default:

- These functions perform a case-sensitive match. Use **MatchOptions.IgnoreCase** to perform case-insensitive matches.
- **IsMatch** matches the entire text string (**Complete** MatchOption), while **Match** and **MatchAll** search for a match anywhere in the text string (**Contains** MatchOption). Use **Complete**, **Contains**, **BeginsWith**, or **EndsWith** as appropriate for your scenario.

IsMatch returns *true* if the text string matches the pattern or *false* if it doesn't. **Match** returns *blank* if no match is found that can be tested with the IsBlank function. **MatchAll** returns an empty table if no match is found that can be tested with the IsEmpty function.

If you use **MatchAll** to split a text string, consider using the Split function, which is simpler and faster.

Patterns

The key to using these functions is in describing the pattern to match. You describe the pattern in a text string as a combination of:

- Ordinary characters, such as "**abc**" or "**123**".
- Predefined patterns, such as **Letter**, **MultipleDigits**, or **Email**. (The **Match** enum defines these patterns.)
- Regular-expression codes, such as "**\d+\s+\d+**" or "**[a-z]+**".

Combine these elements using the string-concatenation operator &. For example, "**abc**" & **Digit** & "**\s+**" is a valid pattern that matches the characters "a", "b", and "c", followed by a digit from 0 to 9, followed by at least one whitespace character.

Ordinary characters

The simplest pattern is a sequence of ordinary characters that match exactly.

For example, when used with the **IsMatch** function, the string "Hello" matches the pattern "**Hello**" exactly. No more and no less. The string `"hello!"` doesn't match the pattern because

of the exclamation point on the end and because the case is wrong for the letter "h". (See Match options for ways to modify this behavior.)

In the pattern language, the characters `. ? * + ( ) [ ] ^ $ | \` are reserved for special purposes. To use these characters, either prefix the character with a **** (backslash) to indicate that the character should be taken literally, or use one of the predefined patterns. For example, you can match the string `"Hello?"` by using the pattern `"Hello\\?"` with a backslash before the question mark.

Predefined patterns

Predefined patterns provide a simple way to match either one of a set of characters or a sequence of multiple characters. Use the string-concatenation operator **&** to combine your own text strings with members of the **Match** enum:

Expand table

Match enum	Description	Regular expression
Any	Matches any character.	`.`
Comma	Matches a comma `,`.	`,`
Digit	Matches a single digit ("0" through "9").	`\d`
Email	Matches an email address that contains an "at" symbol ("@") and a domain name that contains a dot (".")	*See note*
Hyphen	Matches a hyphen.	`-` *See note*
LeftParen	Matches a left parenthesis `(`.	`\(`
Letter	Matches a letter.	`\p{L}`
MultipleDigits	Matches one or more digits.	`\d+`
MultipleLetters	Matches one or more letters.	`\p{L}+`
MultipleNonSpaces	Matches one or more characters that don't add whitespace (not space, tab, or newline).	`\S+`
MultipleSpaces	Matches one or more characters that add whitespace (space, tab, or newline).	`\s+`
NonSpace	Matches a single character that doesn't add whitespace.	`\S`
OptionalDigits	Matches zero, one, or more digits.	`\d*`

Match enum	Description	Regular expression
OptionalLetters	Matches zero, one, or more letters.	`\p{L}*`
OptionalNonSpaces	Matches zero, one, or more characters that don't add whitespace.	`\S*`
OptionalSpaces	Matches zero, one, or more characters that add whitespace.	`\s*`
Period	Matches a period or dot `.`.	`\.`
RightParen	Matches a right parenthesis `)`.	`\)`
Space	Matches a character that adds whitespace.	`\s`
Tab	Matches a tab character.	`\t`

For example, the pattern **"A" & MultipleDigits** matches the letter "A" followed by one or more digits.

Power Apps uses a different definition for **Match.EMail** and **Match.Hyphen**. Evaluate `Text( Match.Email )` to see the regular expression used by your host.

Regular expressions

The pattern they these functions use is called a regular expression. Power Fx's specific dialect of regular expressions is detailed in Regular expressions in Power Fx.

Regular expressions are powerful and serve a wide variety of purposes. They can also look like a random sequence of punctuation marks. This article doesn't describe all aspects of regular expressions, but a wealth of information, tutorials, and tools are available online.

Regular expressions have a long history and are available in many programming languages. Every programming language has its own dialect of regular expressions, and there are few standards. We strive to ensure the same regular expression gives the same result across all Power Fx implementations. Compatibility isn't easy to accomplish as Power Fx runs on top of JavaScript and .NET which have significant differences. To accommodate running on different platforms, Power Fx regular expressions are limited to a subset of features that are widely supported across the industry.

As a result, some regular expressions that may work in other environments may be blocked or require a tweak in Power Fx. Authoring time errors are reported as unsupported features are encountered. This is one of the reasons that the regular expression and options must be an authoring time constant and not dynamic (for example, provided in a variable).

> **Note**
>
> Power Apps uses an earlier version of Power Fx regular expressions that has fewer limitations but also fewer features. **MatchOptions.DotAll** and **MatchOptions.FreeSpacing** aren't available and the definitions of **Match.Email** and **Match.Hyphen** are different. Unicode surrogate pairs aren't treated as a single character. **MatchOptions.NumberedSubMatches** is the default. The version of regular expressions described here will be available in Power Apps soon, under a "Power Fx V1.0 compatibility" switch.

Here are some basic elements of regular expressions that build up complexity in parsing a number.

Expand table

Feature	Example	Description
Predfined character class	`\d`	This regular expression matches a single number, such as `1`. A character class matches a set of characters and `\d` matches the standard digits `0` to `9` and also digits defined in the Unicode character category "Nd". There are character classes for letters and numbers with `\w` and spaces including newlines with `\s`. There are also inverse character classes that are capitalized: `\D` matches everything that `\d` does not.
One or More	`\d+`	This regular expression matches one or more numbers, such as `123`. A `+` after an element says "one or more" of the last element.
Zero or One	`\+?\d`	This regular expression matches an optional `+` sign followed by one ore more numbers, such as `+123` as well as just `123`. A `?` after an element says "this is optional, it can occur zero or one time". The `+` has a backslash before it to distinguish it as a literal character rather than the "one or more" usage.
Grouping and alternation	`(-|\+)?\d+`	This regular expression matches either an `+` sign or a `-`, optionally, and then a series of numbers, such as `-123`, `+123`, and `123`. We are introducing two concepts here that are often used together. First, we have the parenthesis that group as set of elements together, for the `?` to act upon. Second, we have `|` which says "either this or that".
Custom character class	`(-|\+)?\d+[eE][\-\+]?\d+`	This regular expression adds an exponent to the mix with two character classes, matching `+123e-12`. A character class is like `|` alternation providing a "match one of these things" in a more

Feature	Example	Description
		compact form. As with the `+`, `-` has special meaning in regular expression character classes so we need to escape it.
Zero or More	`(-|\+)?\d+\.?\d*[eE][\-\+]?\d+`	This regular expression adds the decimal after the integer portion of the number, such as `-123.456e-89` or `-123.E+32`. The `\d` after the decimal point has a `*` quantifier that says "zero or more times" for decimal digits after the `.`
Capture groups	`(?<number>(-|\+)?\d+\.?\d*)[eE](?<exponent>[\-\+]?\d+)`	Finally, we add capture groups for `number` and `exponent`. Not only can a regular expression match the entire string, it can also extract portions for use in your formulas, in this case the part before the `e` (or `E`) and the part after.

These examples only give a small taste of what regular expressions can do. They are commonly used to validate ID numbers, email addresses, phone numbers, dates and times, and to extract information from all kinds of text files. Continue your journey by reading Regular expressions in Power Fx, experiment, and use the web to learn more.

Match options

Modify the behavior of these functions by specifying one or more options, which you combine using the string- concatenation operator (**&**).

Expand table

MatchOptions enum	Description	Impact on a regular expression
MatchOptions.BeginsWith	The pattern must match from the beginning of the text.	Adds a ^ to the start of the regular expression.
MatchOptions.Complete	Default for **IsMatch** on Power Apps. The pattern matches the entire string of text, from beginning to end.	Adds a ^ to the start and a $ to the end of the regular expression.
MatchOptions.Contains	Default for **Match** and **MatchAll**, and **IsMatch** outside of Power Apps. The pattern must appear somewhere in the text but doesn't need to begin or end it.	Doesn't modify the regular expression.
MatchOptions.DotAll	Changes the behavior of the `.` (dot) operator to match all characters,	Doesn't modify the regular expression.

MatchOptions enum	Description	Impact on a regular expression
	including newline characters. Not available in Power Apps.	This option is the equivalent of the standard "s" modifier for regular expressions.
MatchOptions.EndsWith	The pattern must match the end of the string of text.	Adds a **$** to the end of the regular expression.
MatchOptions.FreeSpacing	Whitespace characters, including newlines, are ignored in the regular expression. End-of-line comments beginning with a `#` are ignored. Not available in Power Apps.	Only changes how the regular expression syntax. This option is the equivalent of the standard "x" modifier for regular expressions.
MatchOptions.IgnoreCase	Treats uppercase and lowercase letters as identical. By default, matching is case sensitive.	Doesn't modify the regular expression. This option is the equivalent of the standard "i" modifier for regular expressions.
MatchOptions.Multiline	Changes the behavior of `^` and `$` to match at the end of aline.	Doesn't modify the regular expression. This option is the equivalent of the standard "m" modifier for regular expressions.
MatchOptions.NumberedSubMatches	Named captures are preferred because they are easier to understand and maintain. Performance also improves as unneeded captures aren't retained. But for older regular expressions, treats each set of parenthesis as a numbered capture that is included with the **SubMatches** table in the result. Default in Power Apps.	Doesn't modify the regular expression. Named captures are disabled and `\1` style back references are enabled.

Using **MatchAll** is the same as using the standard "g" modifier for regular expressions.

Syntax

IsMatch(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to test.
- *Pattern* – Required. The pattern to test as a text string. Concatenate predefined patterns that the **Match** enum defines or provide a regular expression. *Pattern* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs. Note, that the formula must be expressed as "Match.PredefinedPattern" e.g. Match.Email
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Complete** is used. *Options* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.

Match(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to match.
- *Pattern* – Required. The pattern to match as a text string. Concatenate predefined patterns that the **Match** enum defines, or provide a regular expression. *Pattern* must be a constant formula without any variables, data sources, or other dynamic references that change as the app runs.
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Contains** is used. *Options* must be a constant formula without any variables, data sources, or other dynamic references that change as the app runs.

MatchAll(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to match.
- *Pattern* – Required. The pattern to match as a text string. Concatenate predefined patterns that the **Match** enum defines or provide a regular expression. *Pattern* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Contains** is used. *Options* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.

IsMatch examples

Ordinary characters

Imagine your app has a **Text input** control named **TextInput1**. Users enter values into this control to store them in a database.

Users type **Hello world** into **TextInput1**.

Expand table

Formula	Description	Result
`IsMatch( TextInput1.Text, "Hello world" )`	Tests whether the user's input matches, exactly, the string "Hello world".	**true**
`IsMatch( TextInput1.Text, "Good bye" )`	Tests whether the user's input matches, exactly, the string "Good bye".	**false**
`IsMatch( TextInput1.Text, "hello", Contains )`	Tests whether the user's input contains the word "hello" (case sensitive).	**false**
`IsMatch( TextInput1.Text, "hello", Contains & IgnoreCase )`	Tests whether the user's input contains the word "hello" (case insensitive).	**true**

Predefined patterns

Expand table

Formula	Description	Result
`IsMatch( "123-45-7890", Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit )`	Matches a United States Social Security Number	**true**
`IsMatch( "joan@contoso.com", Match.Email )`	Matches an email address	**true**
`IsMatch( "123.456", Match.MultipleDigits & Match.Period & Match.OptionalDigits )`	Matches a sequence of digits, a period, and then zero or more digits.	**true**
`IsMatch( "123", Match.MultipleDigits & Match.Period & Match.OptionalDigits )`	Matches a sequence of digits, a period, and then zero or more digits. A period doesn't appear in the text to match, so this pattern isn't matched.	**false**

Regular expressions

Expand table

Formula	Description	Result
`IsMatch( "986", "\d+" )`	Matches an integer greater than zero.	**true**

Formula	Description	Result
`IsMatch( "1.02", "\d+(\.\d\d)?" )`	Matches a positive currency amount. If the input contains a decimal point, the input must also contain two numeric characters after the decimal point. For example, 3.00 is valid, but 3.1 isn't.	**true**
`IsMatch( "-4.95", "(-)?\d+(\.\d\d)?" )`	Matches a positive or negative currency amount. If the input contains a decimal point, the input must also contain two numeric characters after the decimal point.	**true**
`IsMatch( "111-11-1111", "\d{3}-\d{2}-\d{4}" )`	Matches a United States Social Security number. Validates the format, type, and length of the supplied input field. The string to match must consist of three numeric characters followed by a dash, then two numeric characters followed by a dash, and then four numeric characters.	**true**
`IsMatch( "111-111-111", "\d{3}-\d{2}-\d{4}" )`	Same as the previous example, but one of the hyphens is out of place in the input.	**false**
`IsMatch( "AStrongPasswordNot", "(?!^[0-9]\*$)(?!^[a-zA-Z]\*$)([a-zA-Z0-9]{8,10})" )`	Validates a strong password that must contain eight, nine, or 10 characters, at least one digit, and at least one alphabetic character. The string can't contain special characters.	**false**

Match and MatchAll examples

Expand table

Formula	Description	Result

`Match( "Bob Jones <bob.jones@contoso.com>", "<(?<email>" & Match.Email & ")>")`	Extracts only the email portion of the contact information.	{
		email: "bob.jones@contoso.com",
		FullMatch: "<bob.jones@contoso.com>",
		StartMatch: 11
		}

`Match( "Bob Jones <InvalidEmailAddress>", "<(?<email>" & Match.Email & ")>"`	Extracts only the email portion of the contact information. No legal address is found (there is no @ sign), so the function returns *blank*.	*blank*
`Match( Language(), "(<language>\w{2})(?:-(?<script>\w{4}))?(?:-(?<region>\w{2}))?" )`	Extracts the language, script, and region portions of the language tag that the Language function returns. These results reflect the	{ language: "en", script: *blank*, region: "US",

Formula	Description	Result
	United States; see the Language function documentation for more examples. The **(?:** operator groups characters without creating another sub-match.	FullMatch: "en-US", StartMatch: 1 }

`Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?<seconds>\d+)S)?" )`	Extracts the hours, minutes, and seconds from an ISO 8601 duration value. The extracted numbers are still in a text string; use the Value function to convert it to a number before mathematical operations are performed on it.	{
		hours: "2",
		minutes: "1",
		seconds: "39",
		FullMatch: "PT2H1M39S",
		StartMatch: 1
		}

Let's drill into that last example. If you wanted to convert this string to a date/time value using the Time function, you must pass in the named submatches individually. To do this, use the With function operating on the record that **Match** returns:

Power Fx

```
With(
    Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?<seconds>\d+)S)?" ),
   Time( Value( hours ), Value( minutes ), Value( seconds ) )
)
```

For these examples, add a Button control, set its **OnSelect** property to this formula, and then select the button:

Power Fx

```
Set( pangram, "The quick brown fox jumps over the lazy dog." )
```

Expand table

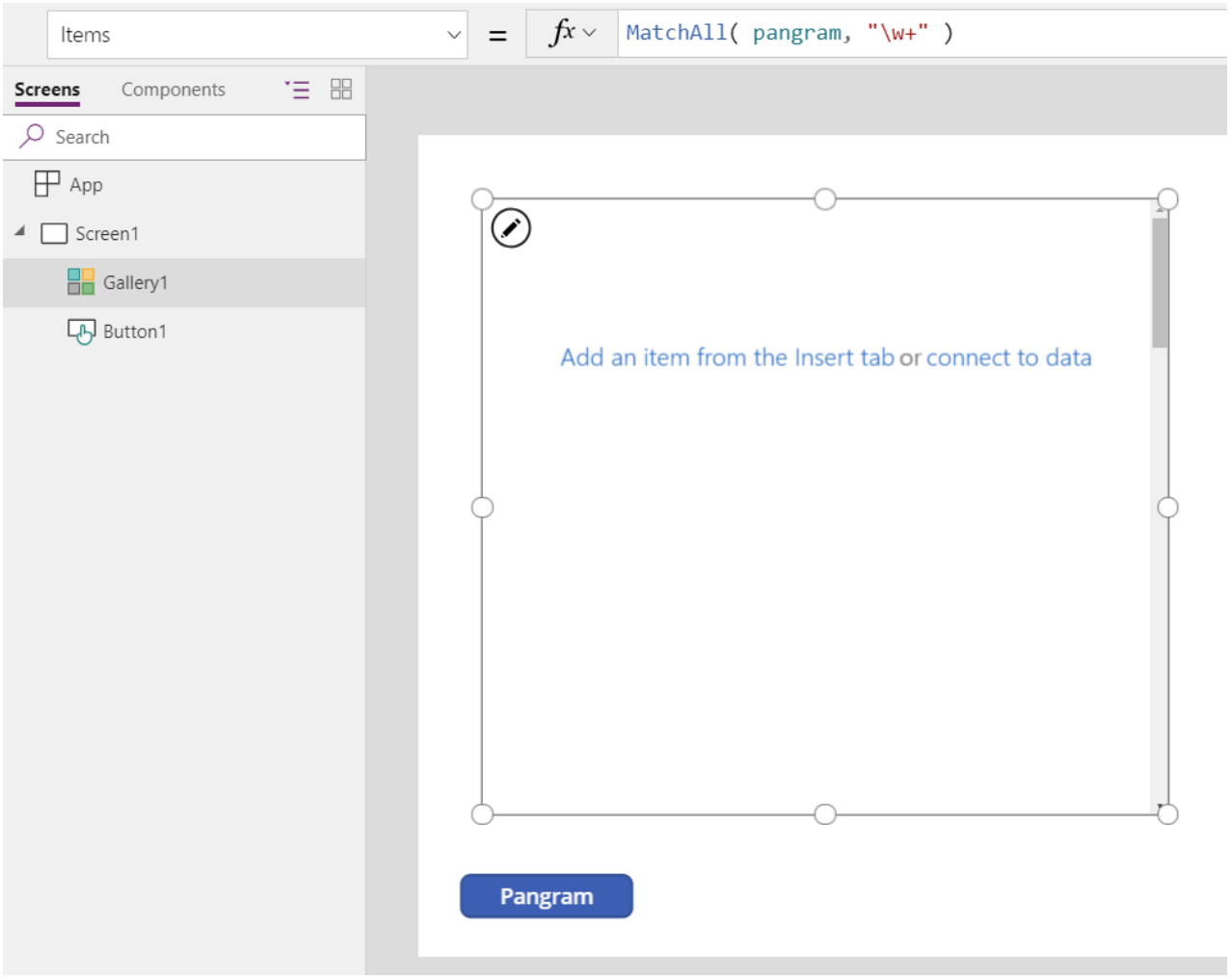
Formula	Description	Result

`Match( pangram, "THE", IgnoreCase )`	Find all matches of "THE" in the text string that the **pangram** variable contains. The string contains two matches, but only the first is returned because	{
		FullMatch: "The",
		StartMatch: 32
		}

<table>
<tr><th>Formula</th><th>Description</th><th>Result</th></tr>
<tr><td></td><td>you're using Match and not MatchAll.</td><td></td></tr>
<tr><td><code>MatchAll(pangram, "the")</code></td><td>Find all matches of "the" in the text string that the pangram variable contains. The test is case sensitive, so only the second instance of "the" is found.</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"the"</td><td>[ ]</td><td>32</td></tr></table></td></tr>
<tr><td><code>MatchAll(pangram, "the", IgnoreCase)</code></td><td>Find all matches of "the" in the text string that the pangram variable contains. In this case, the test is case insensitive, so both instances of the word are found.</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"The"</td><td>[ ]</td><td>1</td></tr><tr><td>"the"</td><td>[ ]</td><td>32</td></tr></table></td></tr>
<tr><td><code>MatchAll(pangram, "\b\wo\w\b")</code></td><td>Finds all three-letter words with an "o" in the middle. "brown" is excluded because it's not a three-letter word and, therefore, fails to match "\b" (word boundary).</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"fox"</td><td>[ ]</td><td>17</td></tr><tr><td>"dog"</td><td>[ ]</td><td>41</td></tr></table></td></tr>
<tr><td><code>Match(pangram, "\b\wo\w\b\s*(?<between>\w.+\w)\s*\b\wo\w\b")</code></td><td>Matches all the characters between "fox" and "dog".</td><td>{
between: "jumps over the lazy",
FullMatch: "fox jumps over the lazy dog",
StartMatch: 17
}</td></tr>
</table>

To see the results of **MatchAll** in a gallery:

1. In an empty screen, insert a blank vertical gallery control.
2. Set the gallery's **Items** property to **MatchAll(pangram, "\w+")** or **MatchAll(pangram, MultipleLetters)**.

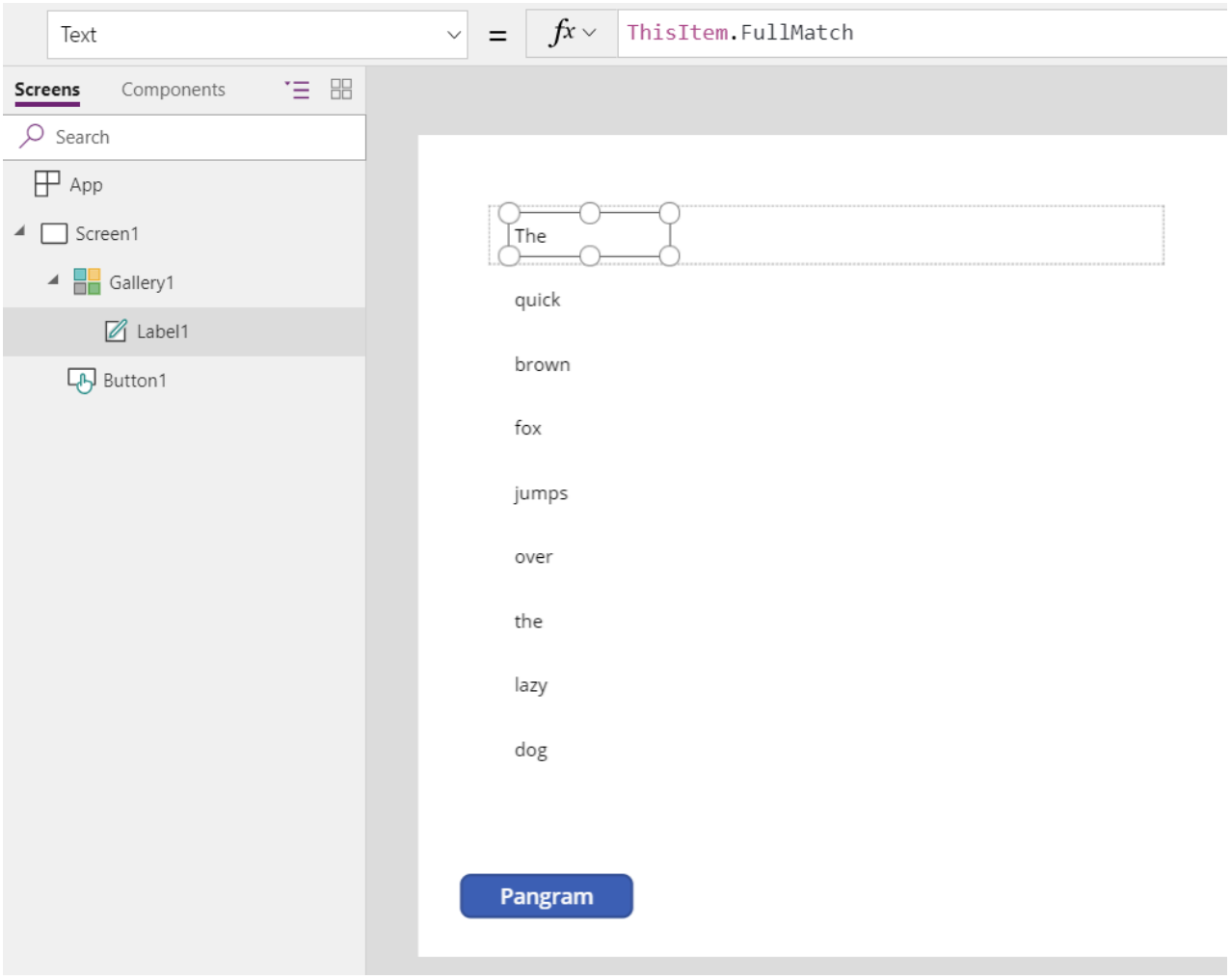


3. Select "Add an item from the Insert tab" in the middle of the gallery control to select the template of the gallery.

4. Add a Label control to the gallery's template.

5. Set the label's **Text** property to **ThisItem.FullMatch**.

   The gallery is filled with each word in our example text. To see all the words on one screen, resize the gallery's template and the label control.


Text
=
ThisItem.FullMatch
Screens
Components
Search
App
Screen1
Gallery1
Label1
Button1
The
quick
brown
fox
jumps
over
the
lazy
dog
Pangram

IsMatch, Match, and MatchAll functions

06/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests for a match or extracts parts of a text string based on a pattern.

Description

The **IsMatch** function tests whether a text string matches a pattern that can comprise ordinary characters, predefined patterns, or a regular expression. The **Match** and **MatchAll** functions return what was matched, including submatches.

Use **IsMatch** to validate what a user typed in a **Text input** control. For example, confirm whether the user entered a valid email address before the result is saved to your data source. If the entry doesn't match your criteria, add other controls that prompt the user to fix the entry.

Use **Match** to extract the first text string that matches a pattern and **MatchAll** to extract all text strings that match. Extract submatches to parse complex strings.

Match returns a record of information for the first match found, and **MatchAll** returns a table of records for every match found. The record or records contain:

Expand table

Column	Type	Description
named sub-match or sub-matches	Text	Each named submatch has its own column. Create a named submatch by using **(?<*name*>...)** in the regular expression. If a named submatch has the same name as one of the predefined columns, the submatch takes precedence, and a warning is generated. Rename the submatch to avoid this warning.
FullMatch	Text	All of the text string that was matched.
StartMatch	Number	The starting position of the match within the input text string. The first character of the string returns 1.
SubMatches, only if **MatchOptions.NumberedSubMatches** is used.	Single-column table of	The table of numbered submatches in the order in which they appear in the regular expression. Generally, named submatches are

Column	Type	Description
	Text (column **Value**)	easier to work with and are encouraged. Use the ForAll function or Index function to work with an individual sub-match. If no submatches are defined in the regular expression, this table will be present but empty.

These functions support MatchOptions. By default:

- These functions perform a case-sensitive match. Use **MatchOptions.IgnoreCase** to perform case-insensitive matches.
- **IsMatch** matches the entire text string (**Complete** MatchOption), while **Match** and **MatchAll** search for a match anywhere in the text string (**Contains** MatchOption). Use **Complete**, **Contains**, **BeginsWith**, or **EndsWith** as appropriate for your scenario.

IsMatch returns *true* if the text string matches the pattern or *false* if it doesn't. **Match** returns *blank* if no match is found that can be tested with the IsBlank function. **MatchAll** returns an empty table if no match is found that can be tested with the IsEmpty function.

If you use **MatchAll** to split a text string, consider using the Split function, which is simpler and faster.

Patterns

The key to using these functions is in describing the pattern to match. You describe the pattern in a text string as a combination of:

- Ordinary characters, such as "**abc**" or "**123**".
- Predefined patterns, such as **Letter**, **MultipleDigits**, or **Email**. (The **Match** enum defines these patterns.)
- Regular-expression codes, such as "**\d+\s+\d+**" or "**[a-z]+**".

Combine these elements using the string-concatenation operator &. For example, "**abc**" & **Digit** & "**\s+**" is a valid pattern that matches the characters "a", "b", and "c", followed by a digit from 0 to 9, followed by at least one whitespace character.

Ordinary characters

The simplest pattern is a sequence of ordinary characters that match exactly.

For example, when used with the **IsMatch** function, the string "Hello" matches the pattern "**Hello**" exactly. No more and no less. The string `"hello!"` doesn't match the pattern because

of the exclamation point on the end and because the case is wrong for the letter "h". (See Match options for ways to modify this behavior.)

In the pattern language, the characters `. ? * + ( ) [ ] ^ $ | \` are reserved for special purposes. To use these characters, either prefix the character with a **** (backslash) to indicate that the character should be taken literally, or use one of the predefined patterns. For example, you can match the string `"Hello?"` by using the pattern `"Hello\\?"` with a backslash before the question mark.

Predefined patterns

Predefined patterns provide a simple way to match either one of a set of characters or a sequence of multiple characters. Use the string-concatenation operator **&** to combine your own text strings with members of the **Match** enum:

Expand table

Match enum	Description	Regular expression
Any	Matches any character.	`.`
Comma	Matches a comma `,`.	`,`
Digit	Matches a single digit ("0" through "9").	`\d`
Email	Matches an email address that contains an "at" symbol ("@") and a domain name that contains a dot (".")	*See note*
Hyphen	Matches a hyphen.	`-` *See note*
LeftParen	Matches a left parenthesis `(`.	`\(`
Letter	Matches a letter.	`\p{L}`
MultipleDigits	Matches one or more digits.	`\d+`
MultipleLetters	Matches one or more letters.	`\p{L}+`
MultipleNonSpaces	Matches one or more characters that don't add whitespace (not space, tab, or newline).	`\S+`
MultipleSpaces	Matches one or more characters that add whitespace (space, tab, or newline).	`\s+`
NonSpace	Matches a single character that doesn't add whitespace.	`\S`
OptionalDigits	Matches zero, one, or more digits.	`\d*`

Match enum	Description	Regular expression
OptionalLetters	Matches zero, one, or more letters.	`\p{L}*`
OptionalNonSpaces	Matches zero, one, or more characters that don't add whitespace.	`\S*`
OptionalSpaces	Matches zero, one, or more characters that add whitespace.	`\s*`
Period	Matches a period or dot `.`.	`\.`
RightParen	Matches a right parenthesis `)`.	`\)`
Space	Matches a character that adds whitespace.	`\s`
Tab	Matches a tab character.	`\t`

For example, the pattern **"A" & MultipleDigits** matches the letter "A" followed by one or more digits.

Power Apps uses a different definition for **Match.EMail** and **Match.Hyphen**. Evaluate `Text( Match.Email )` to see the regular expression used by your host.

Regular expressions

The pattern they these functions use is called a regular expression. Power Fx's specific dialect of regular expressions is detailed in Regular expressions in Power Fx.

Regular expressions are powerful and serve a wide variety of purposes. They can also look like a random sequence of punctuation marks. This article doesn't describe all aspects of regular expressions, but a wealth of information, tutorials, and tools are available online.

Regular expressions have a long history and are available in many programming languages. Every programming language has its own dialect of regular expressions, and there are few standards. We strive to ensure the same regular expression gives the same result across all Power Fx implementations. Compatibility isn't easy to accomplish as Power Fx runs on top of JavaScript and .NET which have significant differences. To accommodate running on different platforms, Power Fx regular expressions are limited to a subset of features that are widely supported across the industry.

As a result, some regular expressions that may work in other environments may be blocked or require a tweak in Power Fx. Authoring time errors are reported as unsupported features are encountered. This is one of the reasons that the regular expression and options must be an authoring time constant and not dynamic (for example, provided in a variable).

> **Note**
>
> Power Apps uses an earlier version of Power Fx regular expressions that has fewer limitations but also fewer features. **MatchOptions.DotAll** and **MatchOptions.FreeSpacing** aren't available and the definitions of **Match.Email** and **Match.Hyphen** are different. Unicode surrogate pairs aren't treated as a single character. **MatchOptions.NumberedSubMatches** is the default. The version of regular expressions described here will be available in Power Apps soon, under a "Power Fx V1.0 compatibility" switch.

Here are some basic elements of regular expressions that build up complexity in parsing a number.

Expand table

Feature	Example	Description
Predfined character class	`\d`	This regular expression matches a single number, such as `1`. A character class matches a set of characters and `\d` matches the standard digits `0` to `9` and also digits defined in the Unicode character category "Nd". There are character classes for letters and numbers with `\w` and spaces including newlines with `\s`. There are also inverse character classes that are capitalized: `\D` matches everything that `\d` does not.
One or More	`\d+`	This regular expression matches one or more numbers, such as `123`. A `+` after an element says "one or more" of the last element.
Zero or One	`\+?\d`	This regular expression matches an optional `+` sign followed by one ore more numbers, such as `+123` as well as just `123`. A `?` after an element says "this is optional, it can occur zero or one time". The `+` has a backslash before it to distinguish it as a literal character rather than the "one or more" usage.
Grouping and alternation	`(-|\+)?\d+`	This regular expression matches either an `+` sign or a `-`, optionally, and then a series of numbers, such as `-123`, `+123`, and `123`. We are introducing two concepts here that are often used together. First, we have the parenthesis that group as set of elements together, for the `?` to act upon. Second, we have `|` which says "either this or that".
Custom character class	`(-|\+)?\d+[eE][\-\+]?\d+`	This regular expression adds an exponent to the mix with two character classes, matching `+123e-12`. A character class is like `|` alternation providing a "match one of these things" in a more

Feature	Example	Description
		compact form. As with the `+`, `-` has special meaning in regular expression character classes so we need to escape it.
Zero or More	`(-|\+)?\d+\.?\d*[eE][\-\+]?\d+`	This regular expression adds the decimal after the integer portion of the number, such as `-123.456e-89` or `-123.E+32`. The `\d` after the decimal point has a `*` quantifier that says "zero or more times" for decimal digits after the `.`
Capture groups	`(?<number>(-|\+)?\d+\.?\d*)[eE](?<exponent>[\-\+]?\d+)`	Finally, we add capture groups for `number` and `exponent`. Not only can a regular expression match the entire string, it can also extract portions for use in your formulas, in this case the part before the `e` (or `E`) and the part after.

These examples only give a small taste of what regular expressions can do. They are commonly used to validate ID numbers, email addresses, phone numbers, dates and times, and to extract information from all kinds of text files. Continue your journey by reading Regular expressions in Power Fx, experiment, and use the web to learn more.

Match options

Modify the behavior of these functions by specifying one or more options, which you combine using the string- concatenation operator (**&**).

Expand table

MatchOptions enum	Description	Impact on a regular expression
MatchOptions.BeginsWith	The pattern must match from the beginning of the text.	Adds a ^ to the start of the regular expression.
MatchOptions.Complete	Default for **IsMatch** on Power Apps. The pattern matches the entire string of text, from beginning to end.	Adds a ^ to the start and a $ to the end of the regular expression.
MatchOptions.Contains	Default for **Match** and **MatchAll**, and **IsMatch** outside of Power Apps. The pattern must appear somewhere in the text but doesn't need to begin or end it.	Doesn't modify the regular expression.
MatchOptions.DotAll	Changes the behavior of the `.` (dot) operator to match all characters,	Doesn't modify the regular expression.

MatchOptions enum	Description	Impact on a regular expression
	including newline characters. Not available in Power Apps.	This option is the equivalent of the standard "s" modifier for regular expressions.
MatchOptions.EndsWith	The pattern must match the end of the string of text.	Adds a **$** to the end of the regular expression.
MatchOptions.FreeSpacing	Whitespace characters, including newlines, are ignored in the regular expression. End-of-line comments beginning with a `#` are ignored. Not available in Power Apps.	Only changes how the regular expression syntax. This option is the equivalent of the standard "x" modifier for regular expressions.
MatchOptions.IgnoreCase	Treats uppercase and lowercase letters as identical. By default, matching is case sensitive.	Doesn't modify the regular expression. This option is the equivalent of the standard "i" modifier for regular expressions.
MatchOptions.Multiline	Changes the behavior of `^` and `$` to match at the end of aline.	Doesn't modify the regular expression. This option is the equivalent of the standard "m" modifier for regular expressions.
MatchOptions.NumberedSubMatches	Named captures are preferred because they are easier to understand and maintain. Performance also improves as unneeded captures aren't retained. But for older regular expressions, treats each set of parenthesis as a numbered capture that is included with the **SubMatches** table in the result. Default in Power Apps.	Doesn't modify the regular expression. Named captures are disabled and `\1` style back references are enabled.

Using **MatchAll** is the same as using the standard "g" modifier for regular expressions.

Syntax

IsMatch(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to test.
- *Pattern* – Required. The pattern to test as a text string. Concatenate predefined patterns that the **Match** enum defines or provide a regular expression. *Pattern* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs. Note, that the formula must be expressed as "Match.PredefinedPattern" e.g. Match.Email
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Complete** is used. *Options* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.

Match(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to match.
- *Pattern* – Required. The pattern to match as a text string. Concatenate predefined patterns that the **Match** enum defines, or provide a regular expression. *Pattern* must be a constant formula without any variables, data sources, or other dynamic references that change as the app runs.
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Contains** is used. *Options* must be a constant formula without any variables, data sources, or other dynamic references that change as the app runs.

MatchAll(*Text*, *Pattern* [, *Options*])

- *Text* – Required. The text string to match.
- *Pattern* – Required. The pattern to match as a text string. Concatenate predefined patterns that the **Match** enum defines or provide a regular expression. *Pattern* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.
- *Options* – Optional. A text-string combination of **MatchOptions** enum values. By default, **MatchOptions.Contains** is used. *Options* must be a constant formula without variables, data sources, or other dynamic references that change as the app runs.

IsMatch examples

Ordinary characters

Imagine your app has a **Text input** control named **TextInput1**. Users enter values into this control to store them in a database.

Users type **Hello world** into **TextInput1**.

Expand table

Formula	Description	Result
`IsMatch( TextInput1.Text, "Hello world" )`	Tests whether the user's input matches, exactly, the string "Hello world".	**true**
`IsMatch( TextInput1.Text, "Good bye" )`	Tests whether the user's input matches, exactly, the string "Good bye".	**false**
`IsMatch( TextInput1.Text, "hello", Contains )`	Tests whether the user's input contains the word "hello" (case sensitive).	**false**
`IsMatch( TextInput1.Text, "hello", Contains & IgnoreCase )`	Tests whether the user's input contains the word "hello" (case insensitive).	**true**

Predefined patterns

Expand table

Formula	Description	Result
`IsMatch( "123-45-7890", Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit & Match.Digit )`	Matches a United States Social Security Number	**true**
`IsMatch( "joan@contoso.com", Match.Email )`	Matches an email address	**true**
`IsMatch( "123.456", Match.MultipleDigits & Match.Period & Match.OptionalDigits )`	Matches a sequence of digits, a period, and then zero or more digits.	**true**
`IsMatch( "123", Match.MultipleDigits & Match.Period & Match.OptionalDigits )`	Matches a sequence of digits, a period, and then zero or more digits. A period doesn't appear in the text to match, so this pattern isn't matched.	**false**

Regular expressions

Expand table

Formula	Description	Result
`IsMatch( "986", "\d+" )`	Matches an integer greater than zero.	**true**

Formula	Description	Result
`IsMatch( "1.02", "\d+(\.\d\d)?" )`	Matches a positive currency amount. If the input contains a decimal point, the input must also contain two numeric characters after the decimal point. For example, 3.00 is valid, but 3.1 isn't.	**true**
`IsMatch( "-4.95", "(-)?\d+(\.\d\d)?" )`	Matches a positive or negative currency amount. If the input contains a decimal point, the input must also contain two numeric characters after the decimal point.	**true**
`IsMatch( "111-11-1111", "\d{3}-\d{2}-\d{4}" )`	Matches a United States Social Security number. Validates the format, type, and length of the supplied input field. The string to match must consist of three numeric characters followed by a dash, then two numeric characters followed by a dash, and then four numeric characters.	**true**
`IsMatch( "111-111-111", "\d{3}-\d{2}-\d{4}" )`	Same as the previous example, but one of the hyphens is out of place in the input.	**false**
`IsMatch( "AStrongPasswordNot", "(?!^[0-9]\*$)(?!^[a-zA-Z]\*$)([a-zA-Z0-9]{8,10})" )`	Validates a strong password that must contain eight, nine, or 10 characters, at least one digit, and at least one alphabetic character. The string can't contain special characters.	**false**

Match and MatchAll examples

Expand table

Formula	Description	Result

`Match( "Bob Jones <bob.jones@contoso.com>", "<(?<email>" & Match.Email & ")>")`	Extracts only the email portion of the contact information.	{
		email: "bob.jones@contoso.com",
		FullMatch: "<bob.jones@contoso.com>",
		StartMatch: 11
		}

`Match( "Bob Jones <InvalidEmailAddress>", "<(?<email>" & Match.Email & ")>"`	Extracts only the email portion of the contact information. No legal address is found (there is no @ sign), so the function returns *blank*.	*blank*
`Match( Language(), "(<language>\w{2})(?:-(?<script>\w{4}))?(?:-(?<region>\w{2}))?" )`	Extracts the language, script, and region portions of the language tag that the Language function returns. These results reflect the	{ language: "en", script: *blank*, region: "US",

Formula	Description	Result
	United States; see the Language function documentation for more examples. The **(?:** operator groups characters without creating another sub-match.	FullMatch: "en-US", StartMatch: 1 }

`Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?<seconds>\d+)S)?" )`	Extracts the hours, minutes, and seconds from an ISO 8601 duration value. The extracted numbers are still in a text string; use the Value function to convert it to a number before mathematical operations are performed on it.	{
		hours: "2",
		minutes: "1",
		seconds: "39",
		FullMatch: "PT2H1M39S",
		StartMatch: 1
		}

Let's drill into that last example. If you wanted to convert this string to a date/time value using the Time function, you must pass in the named submatches individually. To do this, use the With function operating on the record that **Match** returns:

Power Fx

```
With(
    Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?<seconds>\d+)S)?" ),
   Time( Value( hours ), Value( minutes ), Value( seconds ) )
)
```

For these examples, add a Button control, set its **OnSelect** property to this formula, and then select the button:

Power Fx

```
Set( pangram, "The quick brown fox jumps over the lazy dog." )
```

Expand table

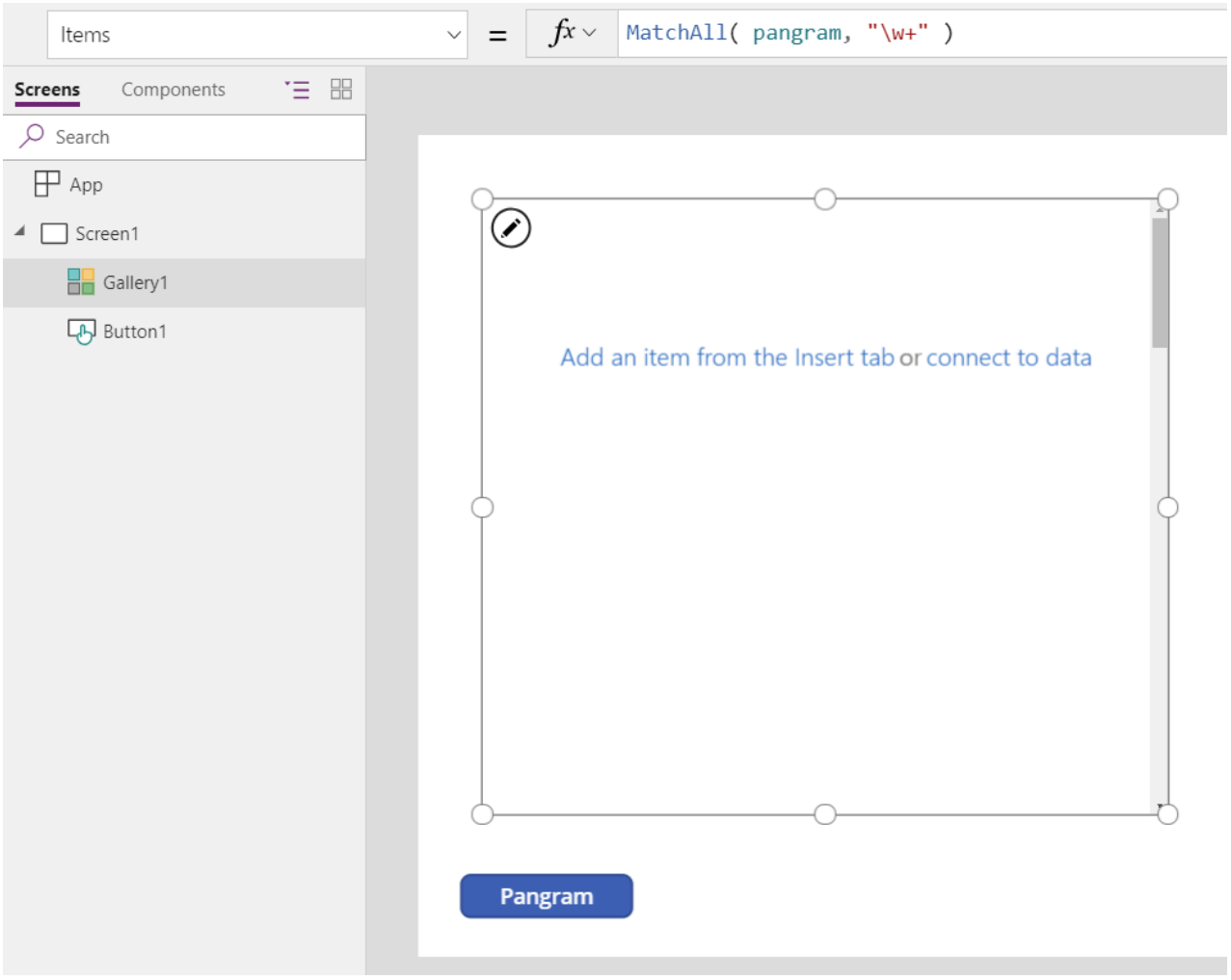
Formula	Description	Result

`Match( pangram, "THE", IgnoreCase )`	Find all matches of "THE" in the text string that the **pangram** variable contains. The string contains two matches, but only the first is returned because	{
		FullMatch: "The",
		StartMatch: 32
		}

<table>
<tr><th>Formula</th><th>Description</th><th>Result</th></tr>
<tr><td></td><td>you're using Match and not MatchAll.</td><td></td></tr>
<tr><td><code>MatchAll(pangram, "the")</code></td><td>Find all matches of "the" in the text string that the pangram variable contains. The test is case sensitive, so only the second instance of "the" is found.</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"the"</td><td>[ ]</td><td>32</td></tr></table></td></tr>
<tr><td><code>MatchAll(pangram, "the", IgnoreCase)</code></td><td>Find all matches of "the" in the text string that the pangram variable contains. In this case, the test is case insensitive, so both instances of the word are found.</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"The"</td><td>[ ]</td><td>1</td></tr><tr><td>"the"</td><td>[ ]</td><td>32</td></tr></table></td></tr>
<tr><td><code>MatchAll(pangram, "\b\wo\w\b")</code></td><td>Finds all three-letter words with an "o" in the middle. "brown" is excluded because it's not a three-letter word and, therefore, fails to match "\b" (word boundary).</td><td><table><tr><th>FullMatch</th><th>SubMatches</th><th>StartMatch</th></tr><tr><td>"fox"</td><td>[ ]</td><td>17</td></tr><tr><td>"dog"</td><td>[ ]</td><td>41</td></tr></table></td></tr>
<tr><td><code>Match(pangram, "\b\wo\w\b\s*(?<between>\w.+\w)\s*\b\wo\w\b")</code></td><td>Matches all the characters between "fox" and "dog".</td><td>{
between: "jumps over the lazy",
FullMatch: "fox jumps over the lazy dog",
StartMatch: 17
}</td></tr>
</table>

To see the results of **MatchAll** in a gallery:

1. In an empty screen, insert a blank vertical gallery control.
2. Set the gallery's **Items** property to **MatchAll(pangram, "\w+")** or **MatchAll(pangram, MultipleLetters)**.

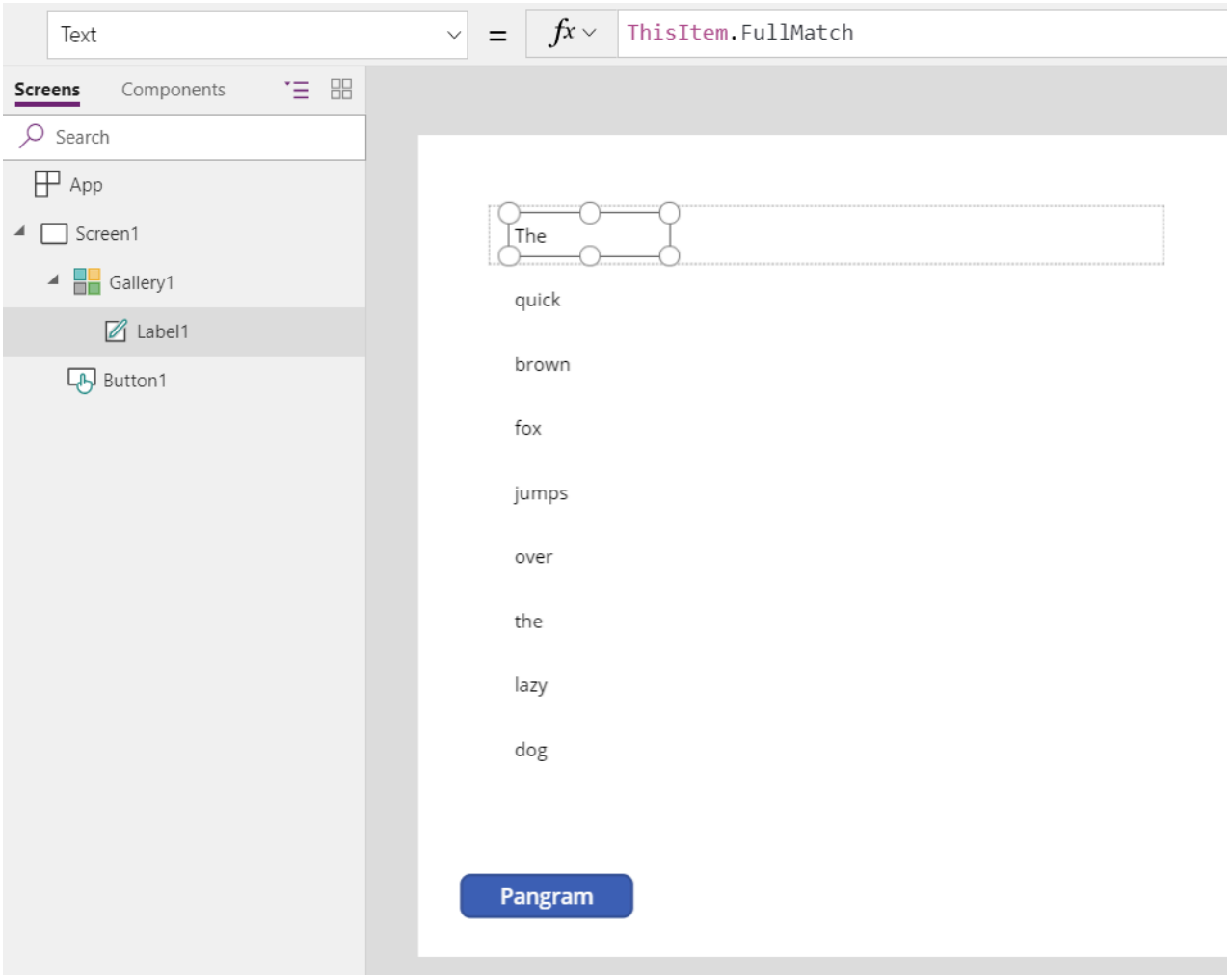


3. Select "Add an item from the Insert tab" in the middle of the gallery control to select the template of the gallery.

4. Add a Label control to the gallery's template.

5. Set the label's **Text** property to **ThisItem.FullMatch**.

   The gallery is filled with each word in our example text. To see all the words on one screen, resize the gallery's template and the label control.


Text
=
ThisItem.FullMatch
Screens
Components
Search
App
Screen1
Gallery1
Label1
Button1
The
quick
brown
fox
jumps
over
the
lazy
dog
Pangram

Average, Max, Min, StdevP, Sum, and VarP functions

Article • 04/20/2025

Expand table

Functions	Applies to
Average **Max** **Min** **Sum**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
StdevP **VarP**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Aggregate functions that summarize a set of numbers.

Description

The **Average** function calculates the average, or arithmetic mean, of its arguments.

The **Max** function finds the maximum value.

The **Min** function finds the minimum value.

The **Sum** function calculates the sum of its arguments.

The **StdevP** function calculates the standard deviation of its arguments.

The **VarP** function calculates the variance of its arguments.

You can supply the values for these functions as:

- Separate arguments. For example, **Sum(1, 2, 3)** returns 6.
- A table and a formula to operate over that table. The aggregate will be calculated on the values of the formula for each record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

These functions operate on numeric values only. Other types of values, such as strings or records, are ignored. Use the **Value** function to convert a string into a number.

The **Average**, **Max**, **Min**, and **Sum** functions can be delegated when used with a data source that supports delegation for these functions. However, **StdevP** and **VarP** can't be delegated for any data sources. If delegation is not supported, only the first portion of the data will be retrieved and then the function applied locally. The result may not represent the complete story. A delegation warning will appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Average(*NumericalFormula1*, [*NumericalFormula2*, ...])
Max(*NumericalFormula1*, [*NumericalFormula2*, ...])
Min(*NumericalFormula1*, [*NumericalFormula2*, ...])
Sum(*NumericalFormula1*, [*NumericalFormula2*, ...])
StdevP(*NumericalFormula1*, [*NumericalFormula2*, ...])
VarP(*NumericalFormula1*, [*NumericalFormula2*, ...])

- *NumericalFormula(s)* - Required. Numeric values to operate on.

Average(*Table*, *NumericalFormula*)
Max(*Table*, *NumericalFormula*)
Min(*Table*, *NumericalFormula*)
Sum(*Table*, *NumericalFormula*)
StdevP(*Table*, *NumericalFormula*)
VarP(*Table*, *NumericalFormula*)

- *Table* - Required. Table to operate on.
- *NumericalFormula* - Required. Formula to evaluate for each record. The result of this formula is used for the aggregation. You can use columns of the table in the formula.

Examples

Step by step

Let's say that you had a data source named **Sales** that contained a **CostPerUnit** column and a **UnitsSold** column, and you set the **Text** property of a label to this function:
Sum(Sales, CostPerUnit * UnitsSold)

The label would show total sales by multiplying the values in those columns for each record and then adding the results from all records together:

Units sold	Cost per unit
30	$150
45	$125
20	$175

Total sales:

$13625

As a different example, let's say that you had sliders that were named **Slider1**, **Slider2**, and **Slider3** and a label with its **Text** property set to this formula:

Sum(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the sum of all values to which the sliders were set.

Average(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the average of all values to which the sliders were set.

Max(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the maximum of all values to which the sliders were set.

Min(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the minimum of all values to which the sliders were set.

StdevP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the standard deviation of all values to which the sliders were set.

VarP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the variance of all values to which the sliders were set.

Left, Mid, and Right functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Extracts the left, middle, or right portion of a string of text.

Description

The **Left**, **Mid**, and **Right** functions return a portion of a string.

- **Left** returns the beginning characters of a string.
- **Mid** returns the middle characters of a string.
- **Right** returns the ending characters of a string.

If you specify a single string as an argument, the function returns the portion that you requested of the string. If you specify a single-column table that contains strings, the function returns a single-column table with a **Value** column containing the portions that you requested of those strings. If you specify a multi-column table, you can shape it into a single-column table, as working with tables describes.

If the starting position is negative or beyond the end of the string, **Mid** returns *blank*. You can check the length of a string by using the **Len** function. If you request more characters than the string contains, the function returns as many characters as possible.

Syntax

Left(*String*, *NumberOfCharacters*)
Mid(*String*, *StartingPosition* [, *NumberOfCharacters*])
Right(*String*, *NumberOfCharacters*)

- *String* - Required. The string to from which to extract the result.
- *StartingPosition* - Required (**Mid** only). The starting position. The first character of the string is position 1.
- *NumberOfCharacters* - Required (**Left** and **Right** only). The number of characters to return. If omitted for the **Mid** function, the function returns the portion from the starting position until the end of the string.

Left(*SingleColumnTable*, *NumberOfCharacters*)
Mid(*SingleColumnTable*, *StartingPosition* [, *NumberOfCharacters*])

Right(*SingleColumnTable*, *NumberOfCharacters*)

- *SingleColumnTable* - Required. A single-column table of strings from which to extract the results.
- *StartingPosition* - Required (**Mid** only). The starting position. The first character of the string is position 1.
- *NumberOfCharacters* - Required (**Left** and **Right** only). The number of characters to return. If omitted for the **Mid** function, the function returns the portion from the starting position until the end of the string.

Examples

Single string

The examples in this section use a text-input control as their data source. The control is named **Author** and contains the string "E. E. Cummings".

Expand table

Formula	Description	Result
Left(Author.Text, 5)	Extracts up to five characters from the start of the string.	"E. E."
Mid(Author.Text, 7, 4)	Extracts up to four characters, starting with the seventh character, from the string.	"Cumm"
Mid(Author.Text, 7)	Extracts all characters, starting with the seventh character, from the string.	"Cummings"
Right(Author.Text, 5)	Extracts up to five characters from the end of the string.	"mings"

Single-column table

Each example in this section extracts strings from the **Address** column of this data source, named **People**, and returns a single-column table that contains the results:

Expand table

Name	Address
"Jean"	"123 Main St NE"

Name	Address
"Fred"	"789 SW 39th #3B"

Expand table

Formula	Description	Result
Left(ShowColumns(People, "Address"), 8)	Extracts the first eight characters of each string.	A single-column table with a `Value` column containing the following values: "123 Main", "789 SW 3"
Mid(ShowColumns(People, "Address"), 5, 7)	Extracts the middle seven characters of each string, starting with the fifth character.	A single-column table with a `Value` column containing the following values: "Main St", "SW 39th"
Right(ShowColumns(People, "Address"), 7)	Extracts the last seven characters of each string.	A single-column table with a `Value` column containing the following values: "n St NE", "9th #3B"

Step-by-step example

1. Import or create a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.
2. Set the **Text** property of the lower label in the gallery to this function:

   Right(ThisItem.ProductName, 3)

   The label shows the last three characters of each product name.

Feedback

Was this page helpful? Yes No

Provide product feedback

Average, Max, Min, StdevP, Sum, and VarP functions

Article • 04/20/2025

Expand table

Functions	Applies to
Average **Max** **Min** **Sum**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
StdevP **VarP**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Aggregate functions that summarize a set of numbers.

Description

The **Average** function calculates the average, or arithmetic mean, of its arguments.

The **Max** function finds the maximum value.

The **Min** function finds the minimum value.

The **Sum** function calculates the sum of its arguments.

The **StdevP** function calculates the standard deviation of its arguments.

The **VarP** function calculates the variance of its arguments.

You can supply the values for these functions as:

- Separate arguments. For example, **Sum(1, 2, 3)** returns 6.
- A table and a formula to operate over that table. The aggregate will be calculated on the values of the formula for each record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

These functions operate on numeric values only. Other types of values, such as strings or records, are ignored. Use the **Value** function to convert a string into a number.

The **Average**, **Max**, **Min**, and **Sum** functions can be delegated when used with a data source that supports delegation for these functions. However, **StdevP** and **VarP** can't be delegated for any data sources. If delegation is not supported, only the first portion of the data will be retrieved and then the function applied locally. The result may not represent the complete story. A delegation warning will appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Average(*NumericalFormula1*, [*NumericalFormula2*, ...])
Max(*NumericalFormula1*, [*NumericalFormula2*, ...])
Min(*NumericalFormula1*, [*NumericalFormula2*, ...])
Sum(*NumericalFormula1*, [*NumericalFormula2*, ...])
StdevP(*NumericalFormula1*, [*NumericalFormula2*, ...])
VarP(*NumericalFormula1*, [*NumericalFormula2*, ...])

- *NumericalFormula(s)* - Required. Numeric values to operate on.

Average(*Table*, *NumericalFormula*)
Max(*Table*, *NumericalFormula*)
Min(*Table*, *NumericalFormula*)
Sum(*Table*, *NumericalFormula*)
StdevP(*Table*, *NumericalFormula*)
VarP(*Table*, *NumericalFormula*)

- *Table* - Required. Table to operate on.
- *NumericalFormula* - Required. Formula to evaluate for each record. The result of this formula is used for the aggregation. You can use columns of the table in the formula.

Examples

Step by step

Let's say that you had a data source named **Sales** that contained a **CostPerUnit** column and a **UnitsSold** column, and you set the **Text** property of a label to this function:
Sum(Sales, CostPerUnit * UnitsSold)

The label would show total sales by multiplying the values in those columns for each record and then adding the results from all records together:

Units sold	Cost per unit
30	$150
45	$125
20	$175

Total sales:

$13625

As a different example, let's say that you had sliders that were named **Slider1**, **Slider2**, and **Slider3** and a label with its **Text** property set to this formula:

Sum(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the sum of all values to which the sliders were set.

Average(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the average of all values to which the sliders were set.

Max(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the maximum of all values to which the sliders were set.

Min(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the minimum of all values to which the sliders were set.

StdevP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the standard deviation of all values to which the sliders were set.

VarP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the variance of all values to which the sliders were set.

Day, Month, Year, Hour, Minute, Second, and Weekday functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns individual components of a Date/Time value.

Description

The **Day** function returns the day component of a Date/Time value, ranging from 1 to 31.

The **Month** function returns the month component of a Date/Time value, ranging from 1 to 12.

The **Year** function returns the year component of a Date/Time value, starting with 1900.

The **Hour** function returns the hour component of a Date/Time value, ranging from 0 (12:00 AM) to 23 (11:00 PM).

The **Minute** function returns the minute component of a Date/Time value, ranging from 0 to 59.

The **Second** function returns the second component of a Date/Time value, ranging from 0 to 59.

The **Weekday** function returns the weekday of a Date/Time value. By default, the result ranges from 1 (Sunday) to 7 (Saturday). You can specify a different range with a Microsoft Excel Weekday function code or a StartOfWeek enumeration value:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	StartOfWeek.Sunday	Numbers 1 (Sunday) through 7 (Saturday). Default.
2, 11	StartOfWeek.Monday	Numbers 1 (Monday) through 7 (Sunday).
3	StartOfWeek.MondayZero	Numbers 0 (Monday) through 6 (Sunday).
12	StartOfWeek.Tuesday	Numbers 1 (Tuesday) through 7 (Monday).

Excel code	StartOfWeek enumeration	Description
13	**StartOfWeek.Wednesday**	Numbers 1 (Wednesday) through 7 (Tuesday).
14	**StartOfWeek.Thursday**	Numbers 1 (Thursday) through 7 (Wednesday).
15	**StartOfWeek.Friday**	Numbers 1 (Friday) through 7 (Thursday).
16	**StartOfWeek.Saturday**	Numbers 1 (Saturday) through 7 (Friday).

All functions return a number.

See working with dates and times for more information.

Syntax

Day(*DateTime*)
Month(*DateTime*)
Year(*DateTime*)
Hour(*DateTime*)
Minute(*DateTime*)
Second(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on.

Weekday(*DateTime* [, *WeekdayFirst*])

- *DateTime* - Required. Date/Time value to operate on.
- *WeekdayFirst* - Optional. Excel code specifying which day starts the week. If not supplied, 1 (Sunday first) is used.

Examples

For the following example, the current time is **3:59:37 PM** on **Thursday, April 9, 2015**.

Expand table

Formula	Description	Result
Year(Now())	Returns the year component of the current time and date.	2015
Month(Now())	Returns the month component of the current time and date.	4

Formula	Description	Result
Day(Now())	Returns the day component of the current time and date.	9
Hour(Now())	Returns the hour component of the current time and date.	15
Minute(Now())	Returns the minute component of the current time and date.	59
Second(Now())	Returns the second component of the current time and date.	37
Weekday(Now())	Returns the weekday component of the current time and date, using the default start of the week as Sunday.	5
Weekday(Now(), 14)	Returns the weekday component of the current time and date, using an Excel code to specify the start of the week as Thursday.	1
Weekday(Now(), StartOfWeek.Wednesday)	Returns the weekday component of the current time and date, using a **StartOfWeek** enumeration to specify the start of the week as Wednesday.	2

Feedback

Was this page helpful? Yes No

Provide product feedback

Mod function

06/21/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the remainder of a division.

Description

The **Mod** function returns the remainder after a number is divided by a divisor.

The result has the same sign as the divisor.

Syntax

Mod(*Number*, *Divisor*)

- *Number* - Required. Number to divide.
- *Divisor* - Required. Number to divide by.

Examples

Expand table

Formula	Description	Result
Mod(25, 4)	The number that remains after dividing 25 by 4	1
Mod(25, -4)	The number that remains after dividing 25 by -4	-3
Mod(-25, -4)	The number that remains after dividing -25 by -4	-1
Mod(-25, 4)	The number that remains after dividing -25 by 4	3

Day, Month, Year, Hour, Minute, Second, and Weekday functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns individual components of a Date/Time value.

Description

The **Day** function returns the day component of a Date/Time value, ranging from 1 to 31.

The **Month** function returns the month component of a Date/Time value, ranging from 1 to 12.

The **Year** function returns the year component of a Date/Time value, starting with 1900.

The **Hour** function returns the hour component of a Date/Time value, ranging from 0 (12:00 AM) to 23 (11:00 PM).

The **Minute** function returns the minute component of a Date/Time value, ranging from 0 to 59.

The **Second** function returns the second component of a Date/Time value, ranging from 0 to 59.

The **Weekday** function returns the weekday of a Date/Time value. By default, the result ranges from 1 (Sunday) to 7 (Saturday). You can specify a different range with a Microsoft Excel Weekday function code or a StartOfWeek enumeration value:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	StartOfWeek.Sunday	Numbers 1 (Sunday) through 7 (Saturday). Default.
2, 11	StartOfWeek.Monday	Numbers 1 (Monday) through 7 (Sunday).
3	StartOfWeek.MondayZero	Numbers 0 (Monday) through 6 (Sunday).
12	StartOfWeek.Tuesday	Numbers 1 (Tuesday) through 7 (Monday).

Excel code	StartOfWeek enumeration	Description
13	**StartOfWeek.Wednesday**	Numbers 1 (Wednesday) through 7 (Tuesday).
14	**StartOfWeek.Thursday**	Numbers 1 (Thursday) through 7 (Wednesday).
15	**StartOfWeek.Friday**	Numbers 1 (Friday) through 7 (Thursday).
16	**StartOfWeek.Saturday**	Numbers 1 (Saturday) through 7 (Friday).

All functions return a number.

See working with dates and times for more information.

Syntax

Day(*DateTime*)
Month(*DateTime*)
Year(*DateTime*)
Hour(*DateTime*)
Minute(*DateTime*)
Second(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on.

Weekday(*DateTime* [, *WeekdayFirst*])

- *DateTime* - Required. Date/Time value to operate on.
- *WeekdayFirst* - Optional. Excel code specifying which day starts the week. If not supplied, 1 (Sunday first) is used.

Examples

For the following example, the current time is **3:59:37 PM** on **Thursday, April 9, 2015**.

Expand table

Formula	Description	Result
Year(Now())	Returns the year component of the current time and date.	2015
Month(Now())	Returns the month component of the current time and date.	4

Formula	Description	Result
Day(Now())	Returns the day component of the current time and date.	9
Hour(Now())	Returns the hour component of the current time and date.	15
Minute(Now())	Returns the minute component of the current time and date.	59
Second(Now())	Returns the second component of the current time and date.	37
Weekday(Now())	Returns the weekday component of the current time and date, using the default start of the week as Sunday.	5
Weekday(Now(), 14)	Returns the weekday component of the current time and date, using an Excel code to specify the start of the week as Thursday.	1
Weekday(Now(), StartOfWeek.Wednesday)	Returns the weekday component of the current time and date, using a **StartOfWeek** enumeration to specify the start of the week as Wednesday.	2

Feedback

Was this page helpful? Yes No

Provide product feedback

Back and Navigate functions

Article • 03/22/2024

Changes which screen is displayed.

Overview

Most apps contain multiple screens. Use the **Back** and **Navigate** function to change which screen is displayed. For example, set the **OnSelect** property of a button to a formula that includes a **Navigate** function if you want to show a different screen when a user selects that button. In that formula, you can specify a visual transition, such as **Fade**, to control how one screen changes to another.

Back and **Navigate** change only which screen is displayed. Screens that aren't currently displayed continue to operate behind the scenes. You can build formulas that refer to properties of controls on other screens. For example, a user can change the value of a slider on one screen, navigate to a different screen that uses that value in a formula, and determine how it affects what happens in the new screen. The user can then navigate back to the original screen and confirm that the slider has kept its value.

Context variables are also preserved when a user navigates between screens. You can use **Navigate** to set one or more context variables for the screen that the formula will display, which is the only way to set a context variable from outside the screen. You can use this approach to pass parameters to a screen. If you've used another programming tool, this approach is similar to passing parameters to procedures.

Use the **App** object's **StartScreen** property to control the first screen to be displayed.

You can use either function only within a behavior formula.

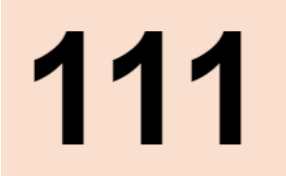
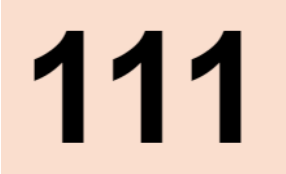
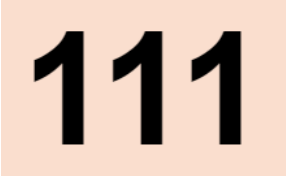
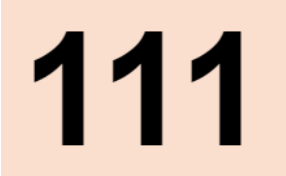
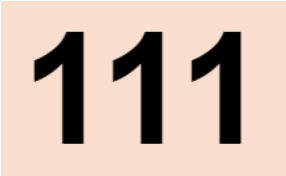
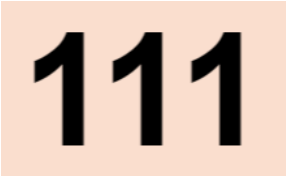
Navigate

Applies to: ✅ Canvas apps ✅ Model-driven apps

In the first argument, specify the name of the screen to display.

In the second argument, specify how the old screen changes to the new screen:

Expand table

Transition Argument	Description	Demonstration
ScreenTransition.Cover	The new screen slides into view, moving right to left, to cover the current screen.	
ScreenTransition.CoverRight	The new screen slides into view, moving left to right, to cover the current screen.	
ScreenTransition.Fade	The current screen fades away to reveal the new screen.	
ScreenTransition.None (Default)	The new screen quickly replaces the current screen.	
ScreenTransition.UnCover	The current screen slides out of view, moving right to left, to uncover the new screen.	
ScreenTransition.UnCoverRight	The current screen slides out of view, moving left to right, to uncover the new screen.	

You can use **Navigate** to create or update context variables of the new screen. As an optional third argument, pass a record that contains the context-variable name as a column name and the new value for the context variable. This record is the same as the record that you use with the **UpdateContext** function.

Set the **OnHidden** property of the old screen, the **OnVisible** property of the new screen, or both to make additional changes during the transition. The **App.ActiveScreen** property will be updated to reflect the change.

Navigate normally returns **true** but will return **false** if an error is encountered.

Context variables for navigation are explained in the article navigate between screens.

Back

Applies to: Canvas apps

The **Back** function returns to the screen that was most recently displayed.

For each **Navigate** call, the app tracks the screen that appeared and the transition. You can use successive **Back** calls to return all the way to the screen that appeared when the user started the app.

When the **Back** function runs, the inverse transition is used by default. For example, if a screen appeared through the **CoverRight** transition, **Back** uses **UnCover** (which is to the left) to return. **Fade** and **None** are their own inverses. Pass an optional argument to **Back** to force a specific transition.

Back normally returns **true** but returns **false** if the user hasn't navigated to another screen since starting the app.

Syntax

Back([*Transition*])

- *Transition* - Optional. The visual transition to use between the current screen and the previous screen. Refer to the list of valid values for this argument earlier in this article. By default, the transition through which a screen returns is the inverse of the transition through which it appeared.

Navigate(*Screen* [, *Transition* [, *UpdateContextRecord*]])

- *Screen* - Required. The screen to display. In place of *Screen*, you can also use a control that is on the screen you wish to navigate to.
- *Transition* - Optional. The visual transition to use between the current screen and the next screen. See the list of valid values for this argument earlier in this article. The default value is **None**.
- *UpdateContextRecord* - Optional. A record that contains the name of at least one column and a value for each column. This record updates the context variables of the new screen as if passed to the **UpdateContext** function.

Examples

Expand table

Formula	Description	Result
Navigate(Details)	Displays the **Details** screen with no transition or change in value for a context variable.	The **Details** screen appears quickly.
Navigate(Details, ScreenTransition.Fade)	Displays the **Details** screen with a **Fade** transition. No value of a context variable is changed.	The current screen fades away to show the **Details** screen.
Navigate(Details, ScreenTransition.Fade, { ID: 12 })	Displays the **Details** screen with a **Fade** transition, and updates the value of the **ID** context variable to **12**.	The current screen fades away to show the **Details** screen, and the context variable **ID** on that screen is set to **12**.
Navigate(Details, ScreenTransition.Fade, { ID: 12 , Shade: Color.Red })	Displays the **Details** screen with a **Fade** transition. Updates the value of the **ID** context variable to **12**, and updates the value of the **Shade** context variable to **Color.Red**.	The current screen fades away to show the **Details** screen. The context variable **ID** on the **Details** screen is set to **12**, and the context variable **Shade** is set to **Color.Red**. If you set the **Fill** property of a control on the **Details** screen to **Shade**, that control would display as red.
Back()	Displays the previous screen with the default return transition.	Displays the previous screen through the inverse transition of the transition through which the current screen appeared.
Back(ScreenTransition.Cover)	Displays the previous screen with the **Cover** transition.	Displays the previous screen through the **Cover** transition, regardless of the transition through which the current screen appeared.

Step-by-step

1. Create a blank app.
2. Add a second screen to it.

   The app contains two blank screens: **Screen1** and **Screen2**.
3. Set the **Fill** property of **Screen2** to the value `Gray`.

4. On **Screen2**, add a button, and set its **OnSelect** property to this formula:

```
Power Apps
Navigate( Screen1, ScreenTransition.Cover )
```

5. While holding down the **Alt** key, select the button.

   Screen1 appears with a white background through a transition that covers to the left.

6. On **Screen1**, add a button, and set its **OnSelect** property to this formula:

```
Power Apps
Back()
```

7. While holding down the **Alt** key, select the button.

   The second screen appears with a gray background through a transition that uncovers to the right (the inverse of **Cover**).

8. Select the button on each screen repeatedly to bounce back and forth.

See also

Using context variables

EditForm, NewForm, SubmitForm, ResetForm, and ViewForm functions

Article • 03/22/2024

Applies to: Canvas apps

View, edit, or create an item, save the contents, and reset the controls in an **Edit form** control.

Overview

These functions change the state of the **Edit form** control. The form control can be in one of these modes:

Expand table

Mode	Description
FormMode.Edit	The form is populated with an existing record and the user can modify the values of the fields. Once complete, the user can save the changes to the record.
FormMode.New	The form is populated with default values and the user can modify the values of the fields. Once complete, the user can add the record to the data source.
FormMode.View	The form is populated with an existing record but the user cannot modify the values of the fields.

Description

These functions are often invoked from the **OnSelect** formula of a **Button** or **Image** control so that the user can save edits, abandon edits, or create a record. You can use controls and these functions together to create a complete solution.

These functions return no values.

You can use these functions only in behavior formulas.

SubmitForm

Use the **SubmitForm** function in the **OnSelect** property of a Button control to save any changes in a Form control to the data source.

Before submitting any changes, this function checks for validation issues with any field that's marked as required or that has one or more constraints on its value. This behavior matches that of the **Validate** function.

SubmitForm also checks the **Valid** property of the Form, which is an aggregation of all the **Valid** properties of the **Card** controls that the Form control contains. If a problem occurs, the data isn't submitted, and the **Error** and **ErrorKind** properties of the Form control are set accordingly.

If validation passes, **SubmitForm** submits the change to the data source.

- If successful, the Form's **OnSuccess** behavior runs, and the **Error** and **ErrorKind** properties are cleared. If the form was in **FormMode.New** mode, it is returned to **FormMode.Edit** mode.
- If unsuccessful, the Form's **OnFailure** behavior runs, and the **Error** and **ErrorKind** properties are set accordingly. The mode of the form is unchanged.

EditForm

The **EditForm** function changes the Form control's mode to **FormMode.Edit**. In this mode, the contents of the Form control's **Item** property are used to populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is changed, not created. **FormMode.Edit** is the default for the Form control.

> ⓘ **Note**
>
> When the form is in edit mode and Item is null, data card properties are not evaluated and will return default values.

NewForm

The **NewForm** function changes the Form control's mode to **FormMode.New**. In this mode, the contents of the Form control's **Item** property are ignored, and the default values of the Form's **DataSource** property populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is created, not changed.

ResetForm

The **ResetForm** function resets the contents of a form to their initial values, before the user made any changes. If the form is in **FormMode.New** mode, the form is reset to **FormMode.Edit** mode. The **OnReset** behavior of the form control also runs. You can also reset individual controls with the **Reset** function but only from within the form.

ViewForm

The **ViewForm** function changes the Form control's mode to **FormMode.View**. In this mode, the contents of the Form control's **Item** property are used to populate the form. The **SubmitForm** and **ResetForm** functions have no effect when in this mode.

DisplayMode Property

The current mode can be read through the **Mode** property. The mode also determines the value of the **DisplayMode** property, which can be used by data cards and controls within the form control. Often, the data card's **DisplayMode** property will be set to **Parent.DisplayMode** (referencing the form) as will the control's **DisplayMode** property (referencing the data card):

Expand table

Mode	DisplayMode	Description
FormMode.Edit	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept changes to a record.
FormMode.New	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept a new record.
FormMode.View	**DisplayMode.View**	Data cards and controls are not editable and optimized for viewing.

Syntax

SubmitForm(*FormName*)

- *FormName* - Required. Form control to submit to the data source.

EditForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.Edit** mode.

NewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.New** mode.

ResetForm(*FormName*)

- *FormName* - Required. Form control to reset to initial values. Also switches the form from **FormMode.New** mode to **FormMode.Edit** mode.

ViewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.View** mode.

Examples

See Understand data forms for complete examples.

1. Add a Button control, set its **Text** property to show **Save**, and set its **OnSelect** property to this formula:

   SubmitForm(EditForm)

2. Set the **OnFailure** property of a Form control to blank and its **OnSuccess** property to this formula:

   Back()

3. Name a **Label** control **ErrorText**, and set its **Text** property to this formula:

   EditForm.Error

   When the user selects the **Save** button, any changes in the Form control are submitted to the underlying data source.

   - If the submission succeeds, any changes are saved or, if the Form control is in **New** mode, a record is created. **ErrorText** is *blank* and the previous screen reappears.
   - If the submission fails, **ErrorText** shows a user-friendly error message, and the current screen remains visible so that the user can correct the problem and try again.

4. Add a Button control, set its **Text** property to show **Cancel**, and set its **OnSelect** property to this formula:

   ResetForm(EditForm); Back()

   When the user selects the **Cancel** button, the values in the Form control are reset to what they were before the user started to edit it, the previous screen reappears,

and the Form control is returned to **Edit** mode if it was in **New** mode.

5. Add a Button control, set its Text property to show **New**, and set its OnSelect property to this formula:

   NewForm(EditForm); Navigate(EditScreen, None)

   When the user selects the **New** button, the Form control switches to **New** mode, the default values for the Form control's data source populate that control, and the screen that contains the Form control appears. When the **SubmitForm** function runs, a record is created instead of updated.

And, Or, and Not functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Boolean logic functions, commonly used to manipulate the results of comparisons and tests.

Description

The **And** function returns **true** if all of its arguments are **true**.

The **Or** function returns **true** if any of its arguments are **true**.

The **Not** function returns **true** if its argument is **false**; it returns **false** if its argument is **true**.

These functions work the same way as they do in Excel. You can also use operators to perform these same operations, using either Visual Basic or JavaScript syntax:

Expand table

Function notation	Visual Basic operator notation	JavaScript operator notation
And(x, y)	x And y	x && y
Or(x, y)	x Or y	x || y
Not(x)	Not x	! x

These functions work with logical values. You can't pass them a number or a string directly; instead, you must make a comparison or a test. For example, this logical formula **x** > **1** evaluates to the Boolean value **true** if **x** is greater than **1**. If **x** is less than **1**, the formula evaluates to **false**.

Syntax

And(*LogicalFormula1*, *LogicalFormula2* [, *LogicalFormula3*, ...])
Or(*LogicalFormula1*, *LogicalFormula2* [, *LogicalFormula3*, ...])
Not(*LogicalFormula*)

- *LogicalFormula(s)* - Required. Logical formulas to evaluate and operate on.

Examples

The examples in this section use these global variables:

- **a** = *false*
- **b** = *true*
- **x** = 10
- **y** = 100
- **s** = "Hello World"

To create these global variables in an app, insert a **Button** control, and set its **OnSelect** property to this formula:

Power Apps

```
Set( a, false ); Set( b, true ); Set( x, 10 ); Set( y, 100 ); Set( s, "Hello World" )
```

Select the button (by clicking it while you hold down the Alt key), and then set the **Text** property of a **Label** control to a formula in the first column of the next table.

Expand table

Formula	Description	Result
And(a, b)	Tests the values of **a** and **b**. One of the arguments is *false*, so the function returns *false*.	*false*
a And b	Same as the previous example, using Visual Basic notation.	*false*
a && b	Same as the previous example, using JavaScript notation.	*false*
Or(a, b)	Tests the values of **a** and **b**. One of the arguments is *true*, so the function returns *true*.	*true*
a Or b	Same as the previous example, using Visual Basic notation.	*true*
a || b	Same as the previous example, using JavaScript notation.	*true*
Not(a)	Tests the value of **a**. The argument is *false*, so the function returns the opposite result.	*true*
Not a	Same as the previous example, using Visual Basic notation.	*true*
! a	Same as the previous example, using JavaScript notation.	*true*
Len(s) < 20 And Not IsBlank(s)	Tests whether the length of **s** is less than 20 and whether it isn't a **blank** value. The length is less than 20, and the value	*true*

Formula	Description	Result
	isn't blank. Therefore, the result is *true*.	
Or(Len(s) < 10, x < 100, y < 100)	Tests whether the length of **s** is less than 10, whether **x** is less than 100, and whether **y** is less than 100. The first and third arguments are false, but the second one is true. Therefore, the function returns *true*.	*true*
Not IsBlank(s)	Tests whether **s** is *blank*, which returns *false*. **Not** returns the opposite of this result, which is *true*.	*true*

Feedback

Was this page helpful? Yes No

Provide product feedback

Notify function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps Power Platform CLI

Displays a banner message to the user.

Description

The **Notify** function displays a banner message to the user at the top of the screen. The notification remains until the user dismisses it or the timeout expires which defaults to 10 seconds.

An appropriate color and icon are used depending on the type of the message. The type is specified by the second argument to the function:

> **Note**
>
> The character limit for **Notify** function is 500 characters.

Expand table

NotificationType Argument	Description
NotificationType.Error	Displays the message as an error.
NotificationType.Information (Default)	Displays the message as informational.
NotificationType.Success	Displays the message as success.
NotificationType.Warning	Displays the message as a warning.

Messages are shown both when authoring your app and when end users are using your app.

Notify can only be used in behavior formulas.

Notify can be paired with the **IfError** function to detect and report errors with a custom error message.

Power Apps can also send push notifications using an entirely different mechanism from **Notify**. For more information, see Send a notification in Power Apps.

Notify always returns *true*.

Note: This function was previously named **ShowError** when it could only display error messages.

> ⓘ **Note**
>
> Notify can be used in standalone canvas app, custom page and Power Fx commanding experiences. The default values for NotificationType and Timeout vary by the experience its used in. You can specify the values you want set for NotificationType and Timeout by explicitly setting them in your Power Fx expression and these values will be respected regardless of the experience.

Syntax

Notify(*Message* [, *NotificationType* [, *Timeout*]])

- *Message* – Required. Message to display to the user.
- *NotificationType* – Optional. Type of the message to display from the table above. The default is **NotificationType.Information**.
- *Timeout* – Optional. Number of milliseconds to wait before automatically dismissing the notification. The default is 10 seconds (or 10,000 milliseconds). The notification is displayed indefinitely with a *Timeout* of 0.

Examples

Step by step

1. Add a **Button** control to your screen.

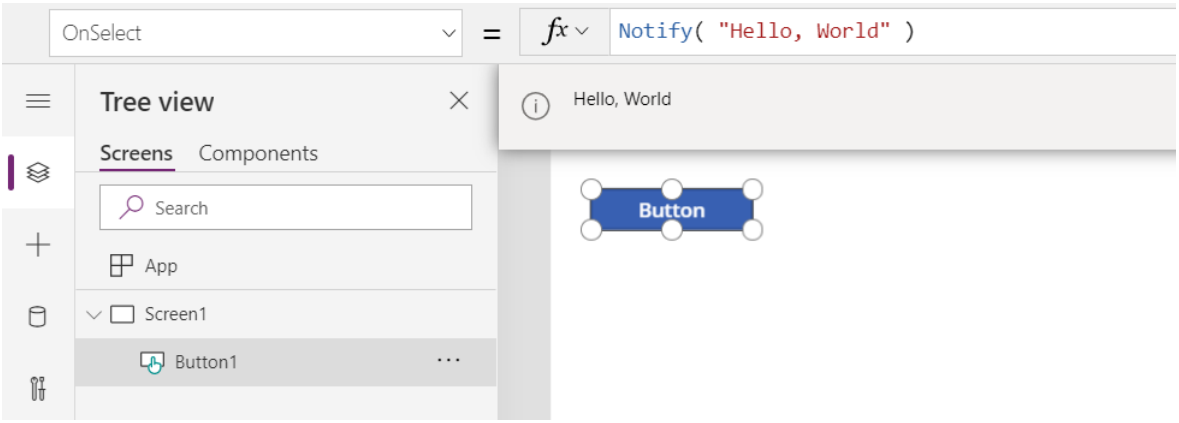
2. Set the **OnSelect** property of the **Button** to the formula:

   Power Apps

   ```
   Notify( "Hello, World" )
   ```

3. Click or press the button.

   Each time the button is clicked, the message **Hello, World** is displayed to the user as informational. It dismisses automatically in 10 seconds (default timeout) if the user doesn't dismiss it or press the button again.

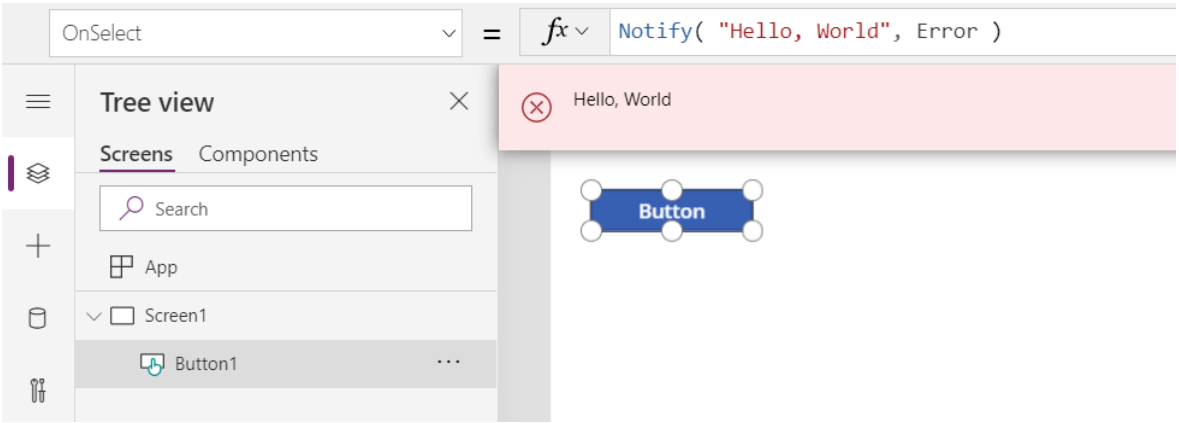


4. Change the type of message to indicate an error. Add a second argument to our formula:

```
Power Apps

Notify( "Hello, World", NotificationType.Error )
```

5. Click or press the button.

   Now each time the button is clicked, the message **Hello, World** is displayed to the user as an error. It dismisses automatically in 10 seconds (default timeout) if the user doesn't dismiss it or press the button again.



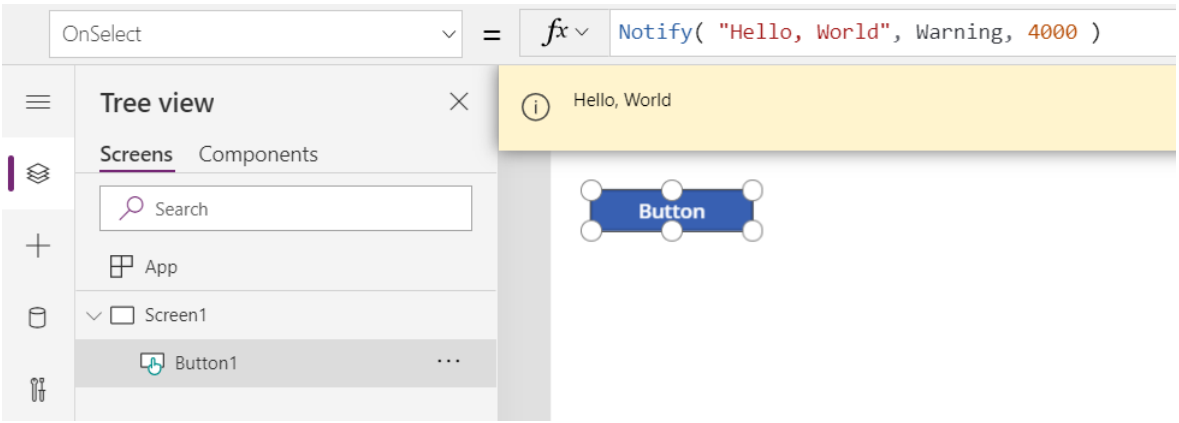
6. Change the type of message to indicate a warning. Change the second argument in our formula:

```
Power Apps

Notify( "Hello, World", NotificationType.Warning, 4000 )
```

7. Click or press the button.

   Now each time the button is clicked, the message **Hello, World** is displayed to the user as a warning. It dismisses automatically in 4 seconds (4,000 milliseconds) if the

user doesn't dismiss it or press the button again.

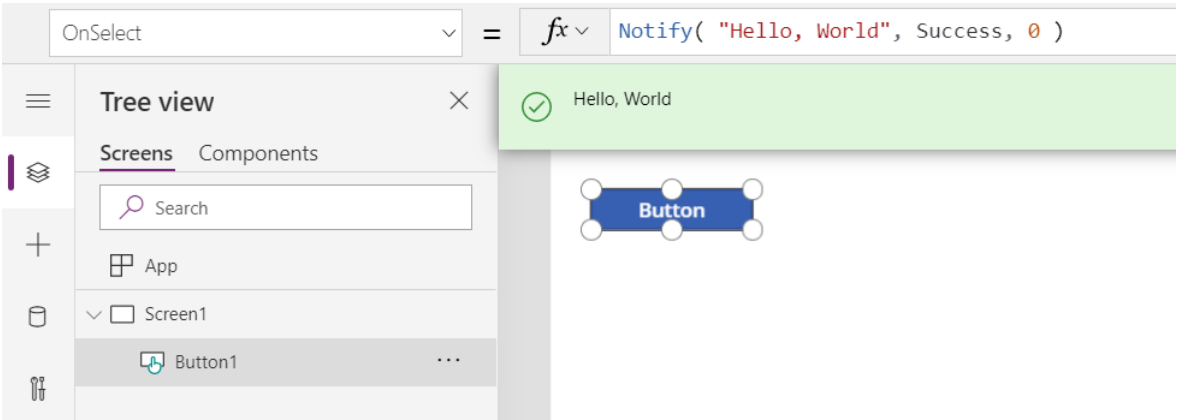


8. Change the type of message to indicate success. Change the second argument in our formula:

```
Power Apps

Notify( "Hello, World", NotificationType.Success, 0 )
```

9. Click or press the button.

   Now each time the button is clicked, the message **Hello, World** is displayed to the user as success. With a **0** timeout, the notification will only be dismissed by the user or by pressing the button again.



Now, Today, IsToday, UTCNow, UTCToday, IsUTCToday functions

Article • 04/20/2025

Expand table

Functions	Applies to
Now	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Today **IsToday**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
UTCNow **UTCToday**	✅ Dataverse formula columns ✅ Power Pages

Returns the current date and time, and tests whether a date/time value is today.

> ⓘ Note
>
> PAC CLI pac power-fx commands do not support the **UTCNow**, **UTCToday**, and **IsUTCToday** functions.

Description

The **Now** function returns the current date and time as a date/time value.

The **Today** function returns the current date as a date/time value. The time portion is midnight. **Today** has the same value throughout a day, from midnight today to midnight tomorrow.

The **IsToday** function tests whether a date/time value is between midnight today and midnight tomorrow. This function returns a Boolean (**true** or **false**) value.

Now, **Today**, and **IsToday** functions work with the local time of the current user.

UTCNow, **UTCToday**, and **IsUTCToday** functions are the same as their non-UTC counterparts but work with time zone independent values and use Coordinated Universal Time (UTC).

> ⓘ Note

- **UTCNow**, **UTCToday**, and **IsUTCToday** are only available in Microsoft Dataverse for Teams formula columns, and only for time-independent fields and values.
- **Now**, **Today**, and **IsToday** are not available in Dataverse for Teams formula columns as evaluations are done without the knowledge of the current user's local time zone. More information: Work with formula table columns in Dataverse for Teams

See Date, Time, and DateTime in the data types documentation and working with dates and times for more information.

Volatile Functions

Now, **Today**, **UTCNow**, and **UTCToday** are volatile functions. These functions return a different value for each evaluation.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Now()** will not change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it is part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = DateAdd(Now(), Slider1.Value, Minutes)** then the current time is retrieved each time the Slider control's value changes and the label's text property is reevaluated.

When used in a behavior formula, volatile functions will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Using the user's local time

Now()

Today()

IsToday(*DateTime*)

- *DateTime* - Required. The date/time value to test.

Using Coordinated Universal Time (UTC)

UTCNow()

UTCToday()

IsUTCToday(*TimeZoneIndependentTime*)

- *TimeZoneIndependentDateTime* - Required. The time zone independent date/time value to test.

Examples

For the examples in this section, the current time is **8:58 PM** on **July 11, 2021** in the Pacific Time Zone (UTC-8) and the language is **en-us**.

Expand table

Formula	Description	Result
Text(Now(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time in the user's time zone, and displays it as a string.	"07/11/2021 20:58:00"
Text(Today(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date only, leaving the time portion as midnight, and displays it as a string.	"07/12/2021 00:00:00"
IsToday(Now())	Tests whether the current date and time is between midnight today and midnight tomorrow.	**true**
IsToday(Today())	Tests whether the current date is between midnight today and midnight tomorrow.	**true**
Text(DateAdd(Now(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time, adds 12 days to the result, and displays it as a string.	"07/23/2021 20:58:00"
Text(DateAdd(Today(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date, adds 12 days to the result, and displays it as a string.	"07/23/2021 00:00:00"
IsToday(DateAdd(Now(), 12))	Tests whether the current date and time, plus 12 days, is between midnight today and midnight tomorrow.	**false**
IsToday(DateAdd(Today(), 12))	Tests whether the current date, plus 12 days, is between midnight today and midnight tomorrow.	**false**
Hour(UTCNow())	Retrieves the current date and time in UTC and extracts the hour only, which is 8 hours ahead of local time.	4

Formula	Description	Result
Day(UTCToday())	Retrieves the current date only in UTC and extracts the day, which is 1 day ahead of local time.	12
IsUTCToday(UTCNow())	Tests whether the current date and time is between midnight today and midnight tomorrow, all in UTC time.	**true**
IsUTCToday(UTCToday())	Tests whether the current date is between midnight today and midnight tomorrow, all in UTC time.	**true**

Display a clock that updates in real time

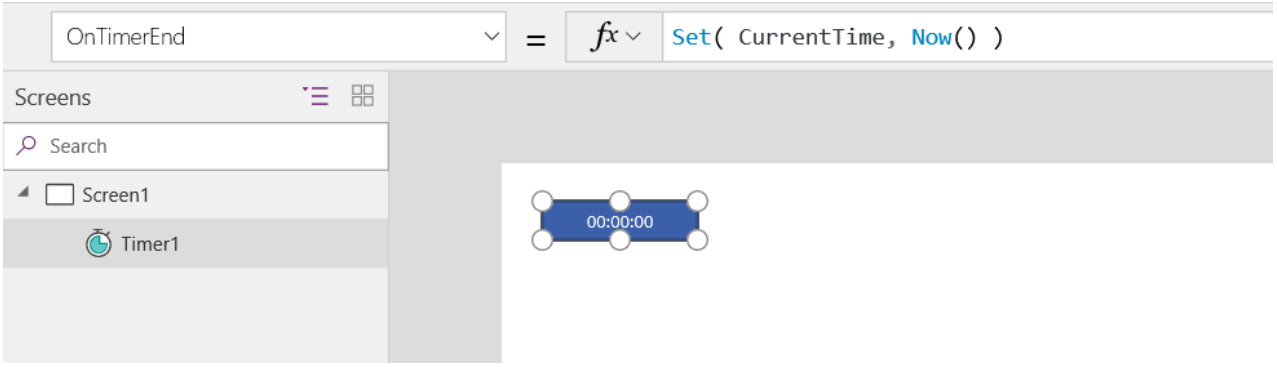
1. Add a Timer control, set its **Duration** property to **1000**, and set its **Repeat** property to **true**.

   The timer will run for one second, automatically start over, and continue that pattern.

2. Set the control's **OnTimerEnd** property to this formula:

   Set(CurrentTime, Now())

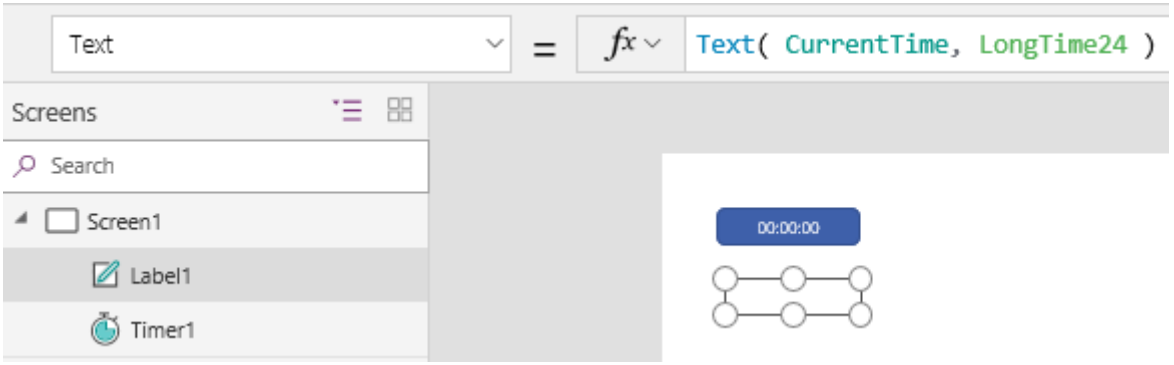
   Whenever the timer starts over (after each second), this formula sets the **CurrentTime** global variable to the current value of the **Now** function.

3. Add a Label control, and set its **Text** property to this formula:

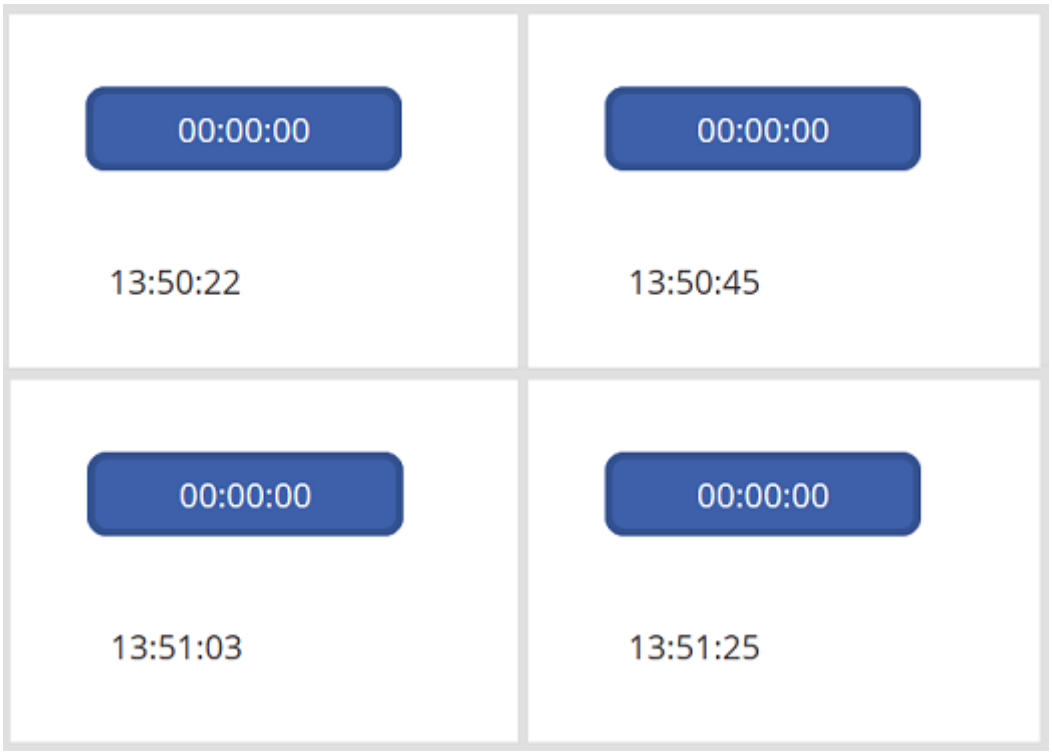
   Text(CurrentTime, LongTime24)

   Use the Text function to format the date and time however you want, or set this property to just **CurrentTime** to show hours and minutes but not seconds.



4. Preview the app by pressing F5, and then start the timer by clicking or tapping it.

   The label continually shows the current time, down to the second.



5. Set the timer's **AutoStart** property to **true** and its **Visible** property to **false**.

   The timer is invisible and starts automatically.

6. Set the screen's OnStart property so that the **CurrentTime** variable has a valid value, as in this example:

   Set(CurrentTime, Now())

   The label appears as soon as the app starts (before the timer runs for a full second).

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

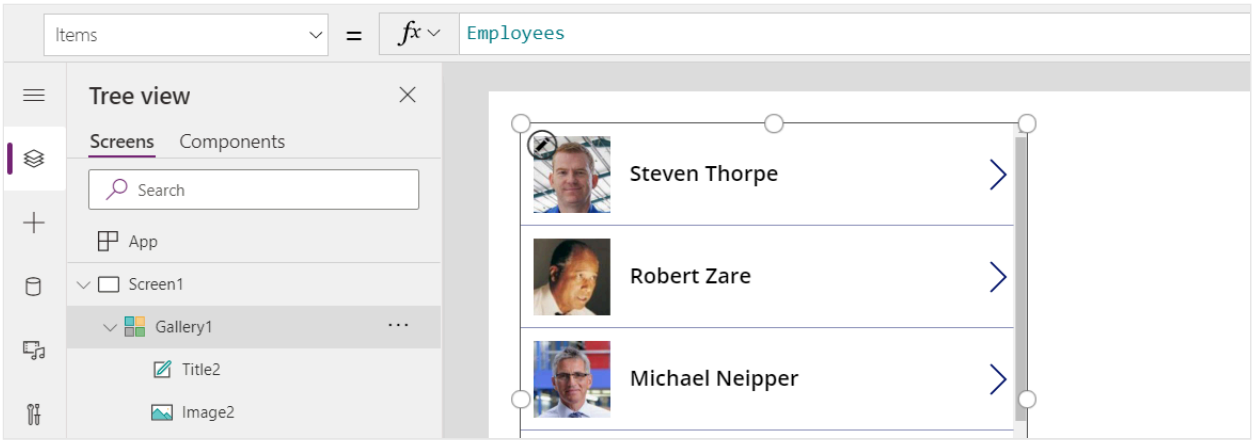
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

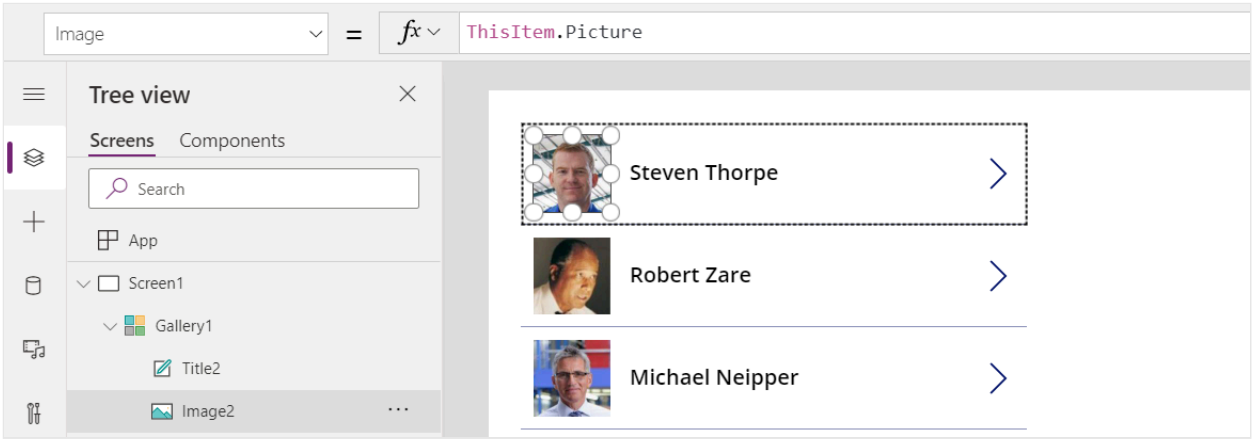
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

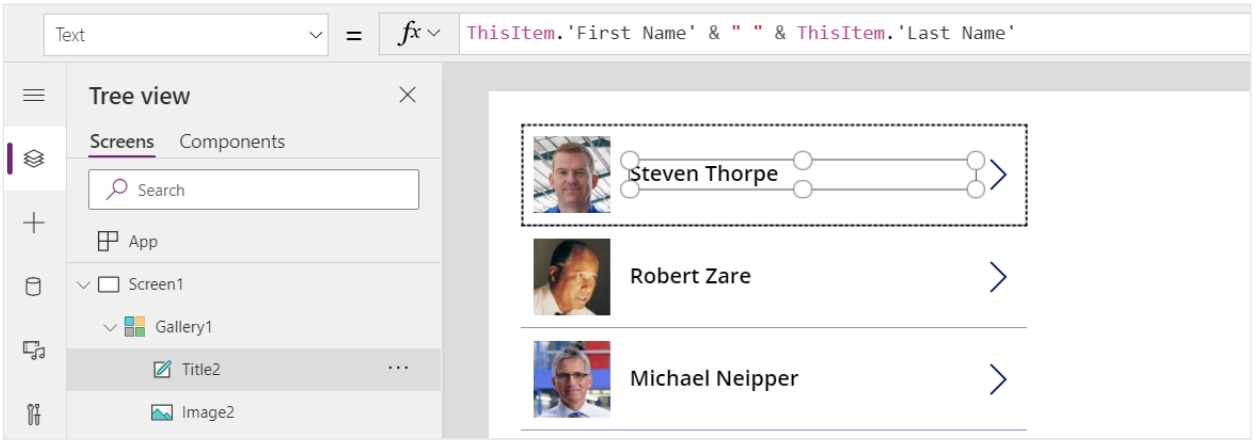
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

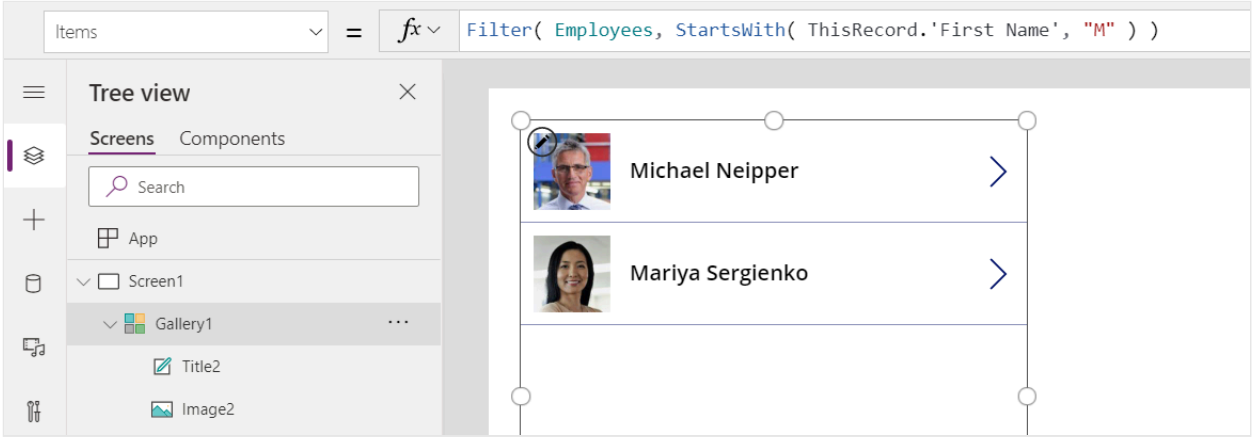


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

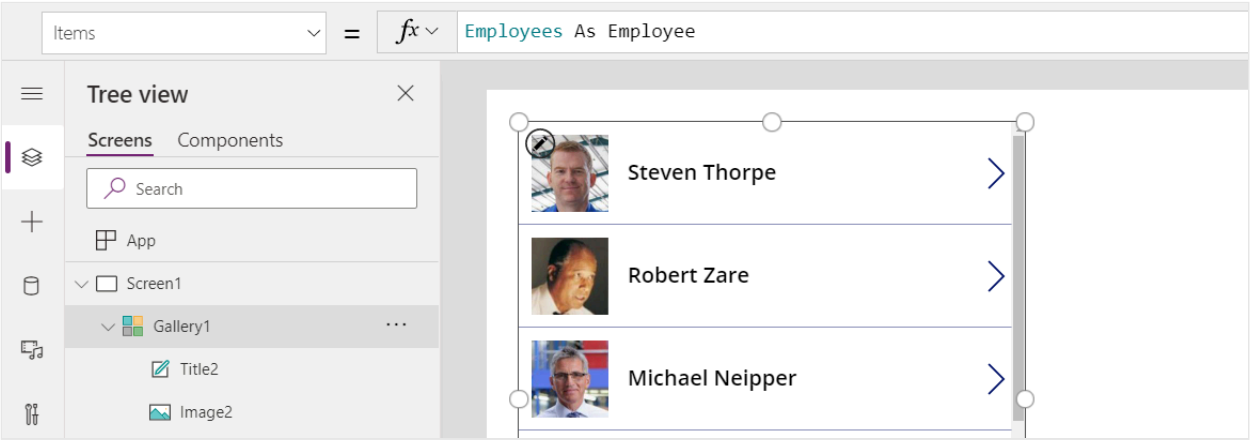
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

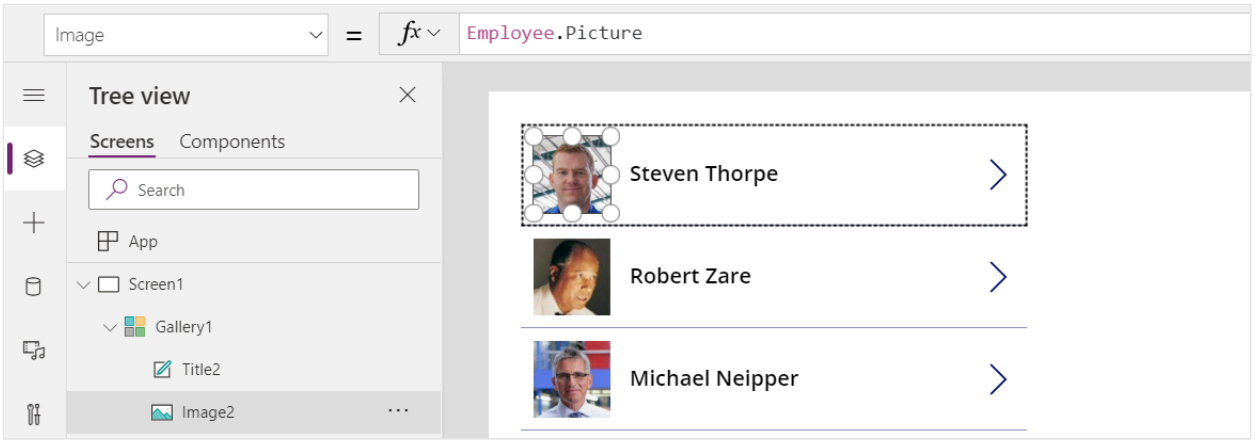
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

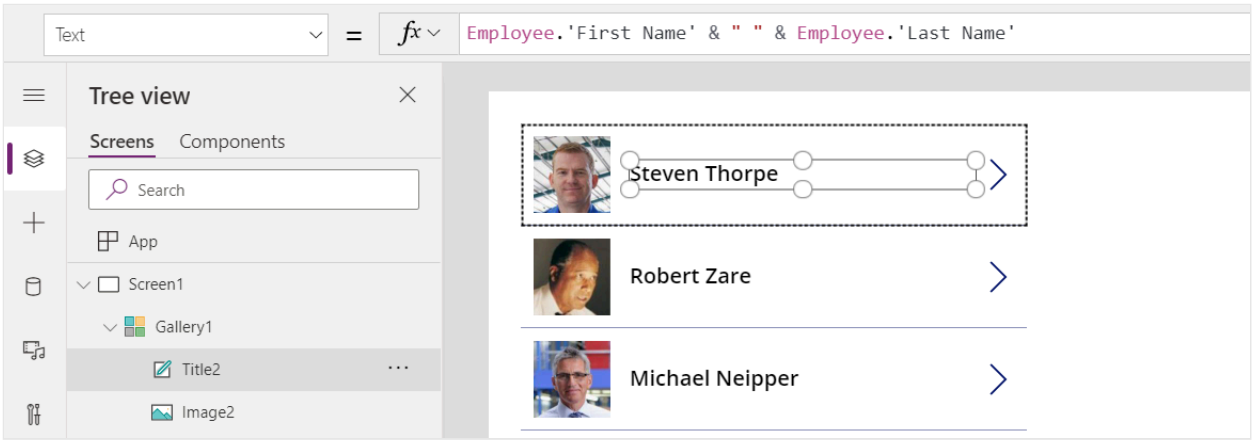
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

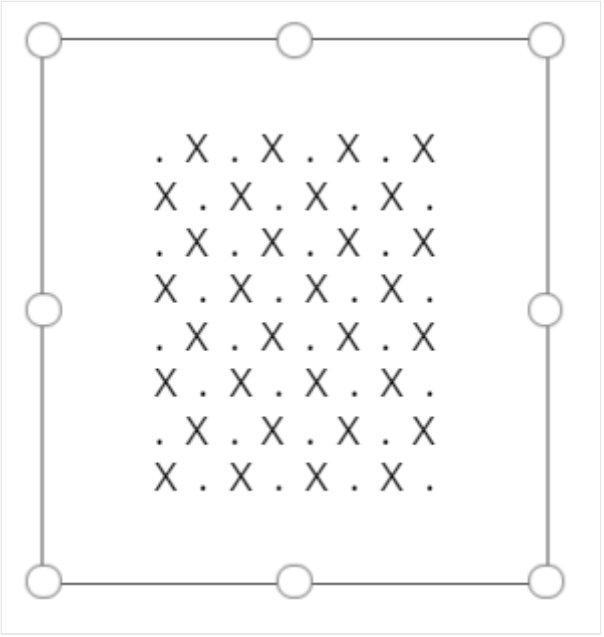
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



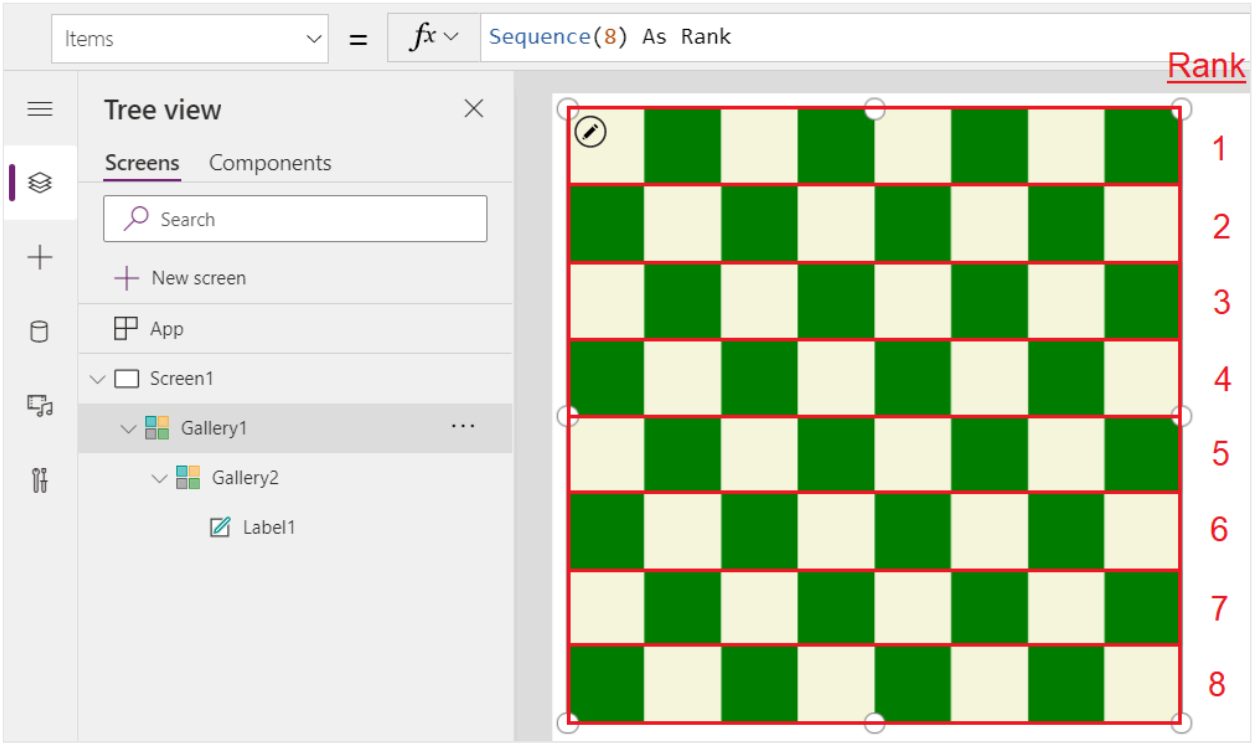
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

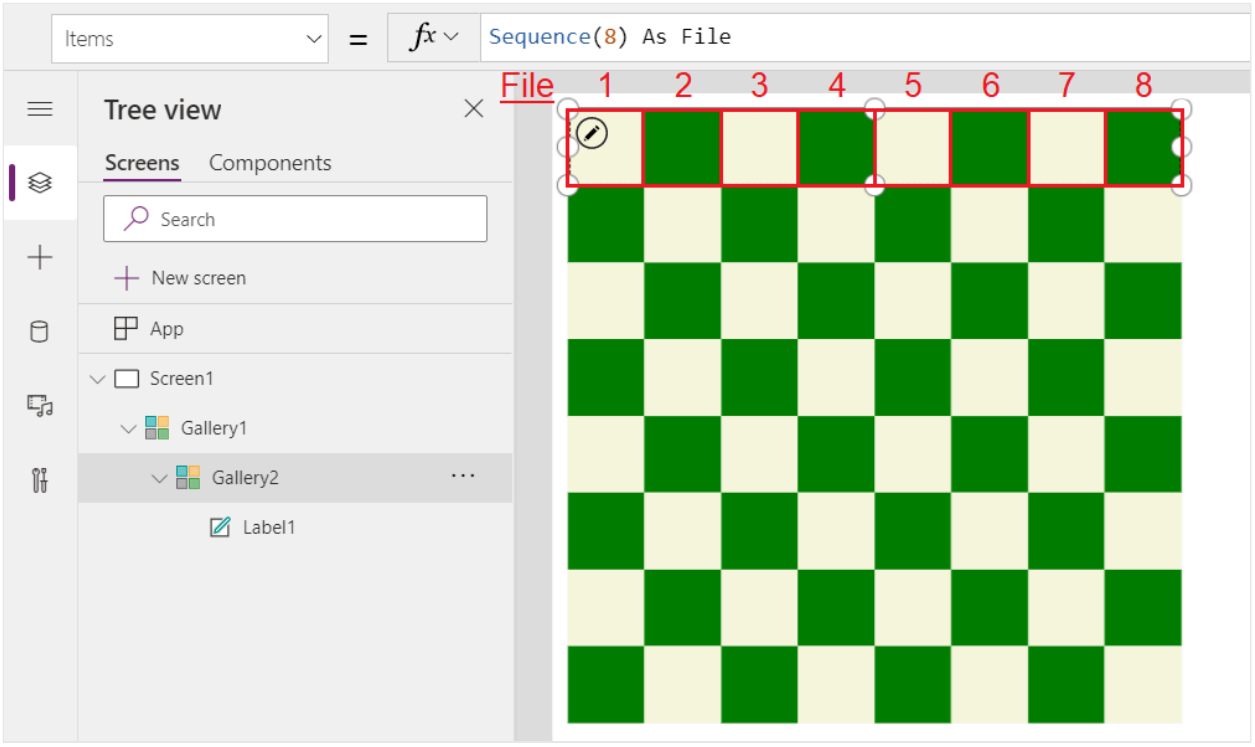
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

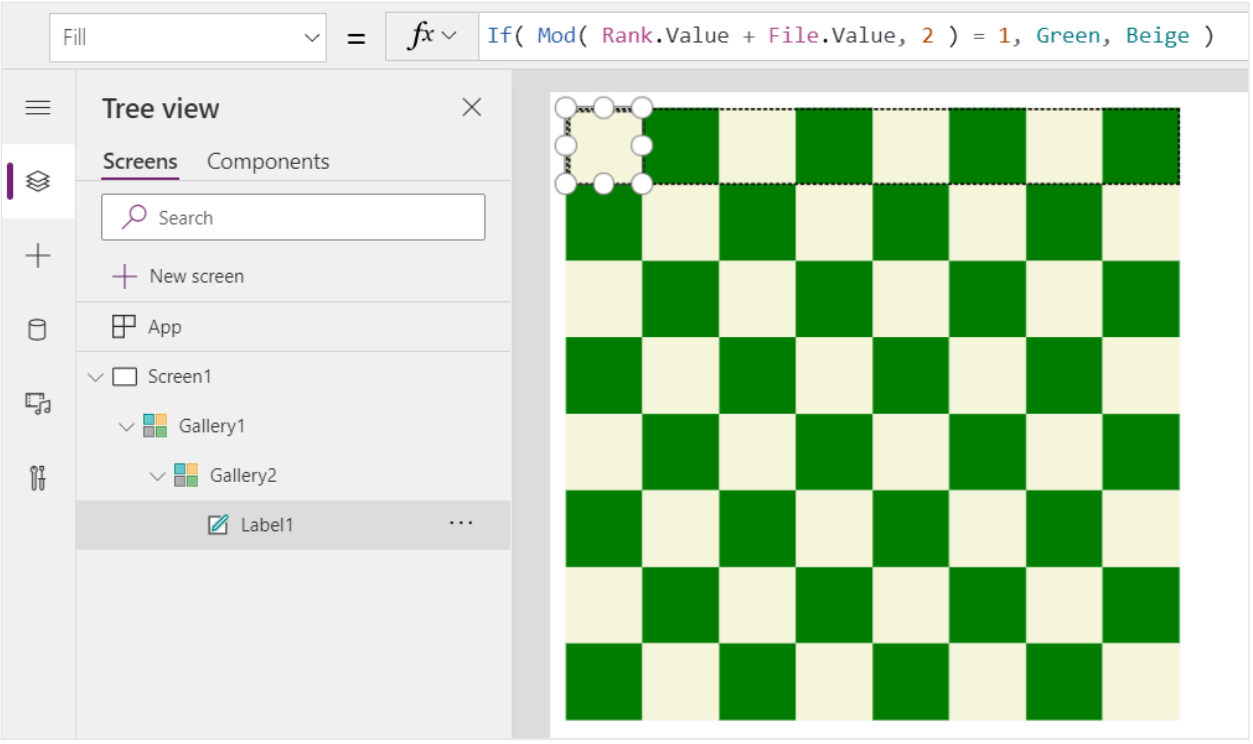
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

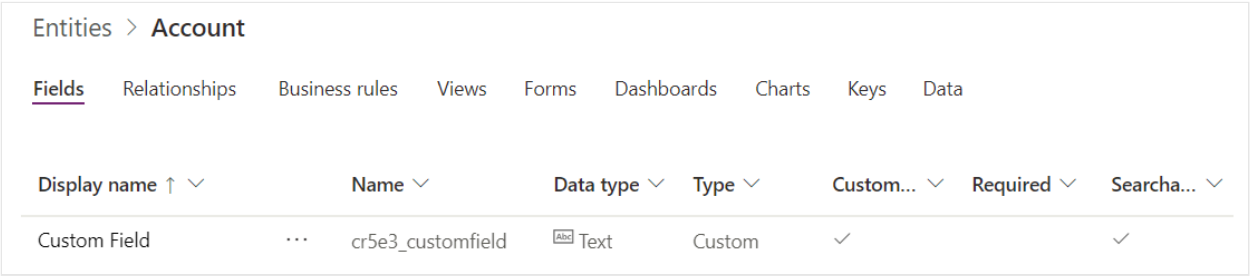
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

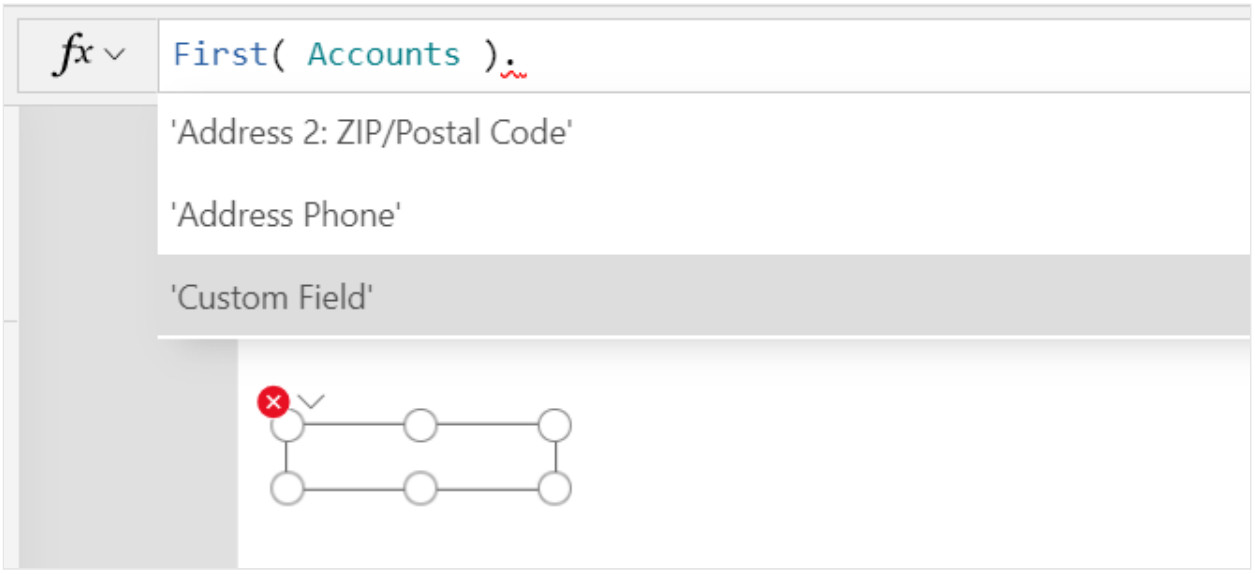
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

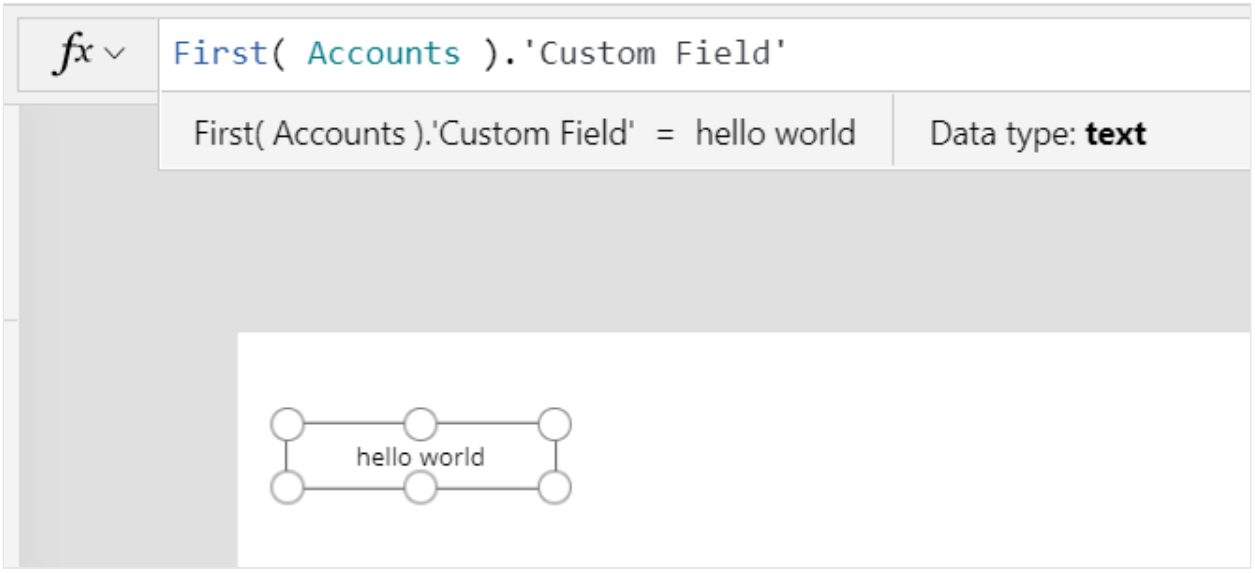
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



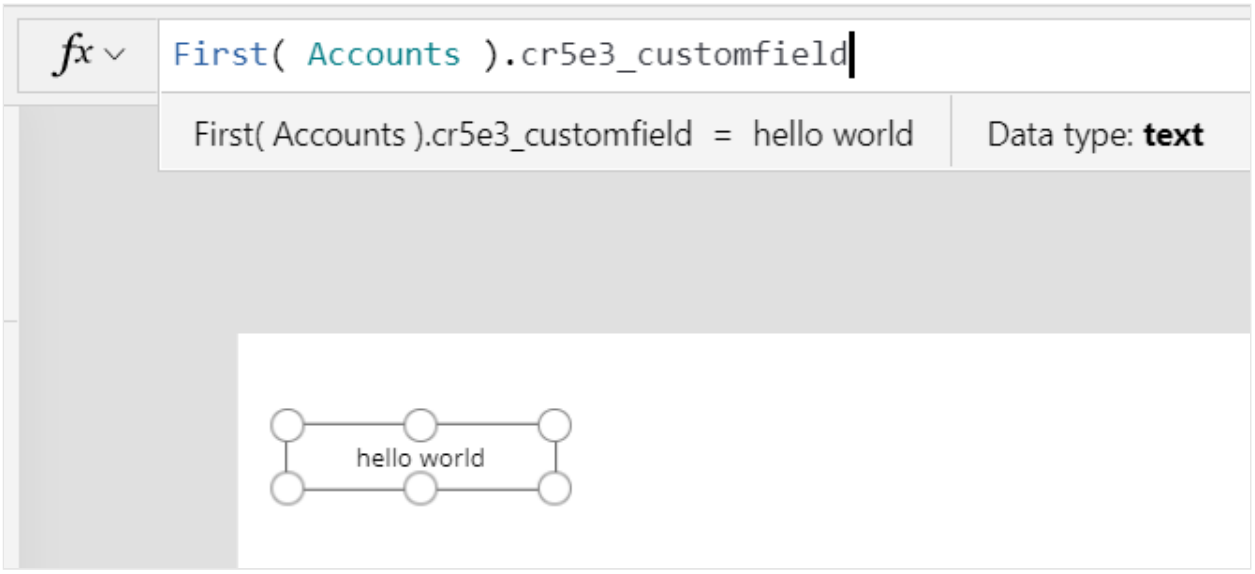
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

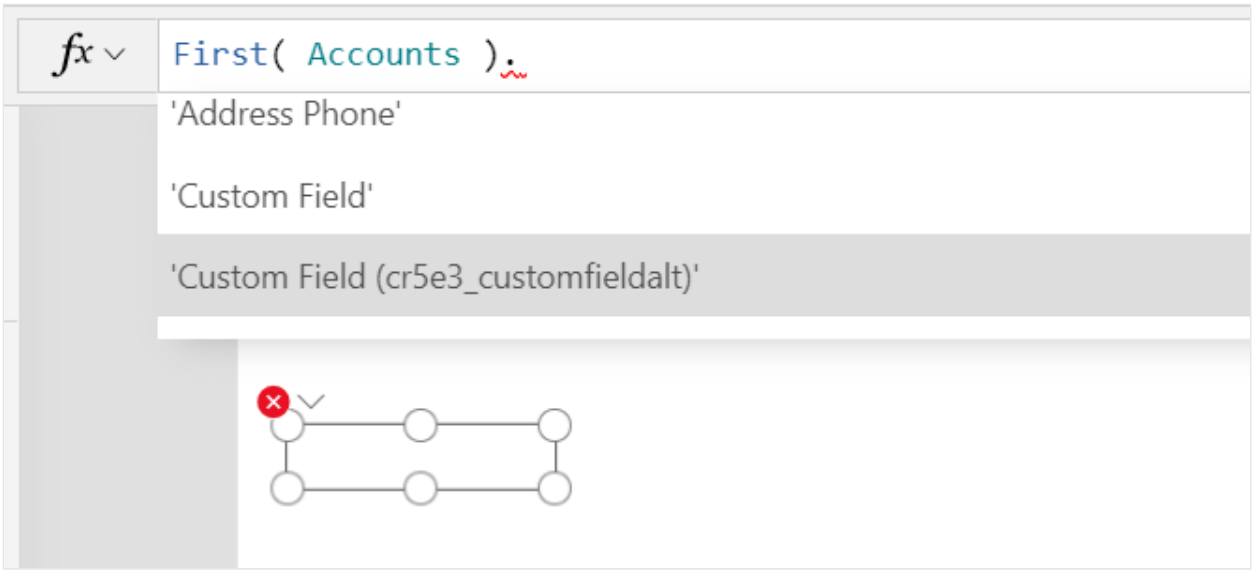


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

And, Or, and Not functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Boolean logic functions, commonly used to manipulate the results of comparisons and tests.

Description

The **And** function returns **true** if all of its arguments are **true**.

The **Or** function returns **true** if any of its arguments are **true**.

The **Not** function returns **true** if its argument is **false**; it returns **false** if its argument is **true**.

These functions work the same way as they do in Excel. You can also use operators to perform these same operations, using either Visual Basic or JavaScript syntax:

Expand table

Function notation	Visual Basic operator notation	JavaScript operator notation
And(x, y)	x And y	x && y
Or(x, y)	x Or y	x || y
Not(x)	Not x	! x

These functions work with logical values. You can't pass them a number or a string directly; instead, you must make a comparison or a test. For example, this logical formula **x** > **1** evaluates to the Boolean value **true** if **x** is greater than **1**. If **x** is less than **1**, the formula evaluates to **false**.

Syntax

And(*LogicalFormula1*, *LogicalFormula2* [, *LogicalFormula3*, ...])
Or(*LogicalFormula1*, *LogicalFormula2* [, *LogicalFormula3*, ...])
Not(*LogicalFormula*)

- *LogicalFormula(s)* - Required. Logical formulas to evaluate and operate on.

Examples

The examples in this section use these global variables:

- **a** = *false*
- **b** = *true*
- **x** = 10
- **y** = 100
- **s** = "Hello World"

To create these global variables in an app, insert a **Button** control, and set its **OnSelect** property to this formula:

Power Apps

```
Set( a, false ); Set( b, true ); Set( x, 10 ); Set( y, 100 ); Set( s, "Hello World" )
```

Select the button (by clicking it while you hold down the Alt key), and then set the **Text** property of a **Label** control to a formula in the first column of the next table.

Expand table

Formula	Description	Result
And(a, b)	Tests the values of **a** and **b**. One of the arguments is *false*, so the function returns *false*.	*false*
a And b	Same as the previous example, using Visual Basic notation.	*false*
a && b	Same as the previous example, using JavaScript notation.	*false*
Or(a, b)	Tests the values of **a** and **b**. One of the arguments is *true*, so the function returns *true*.	*true*
a Or b	Same as the previous example, using Visual Basic notation.	*true*
a || b	Same as the previous example, using JavaScript notation.	*true*
Not(a)	Tests the value of **a**. The argument is *false*, so the function returns the opposite result.	*true*
Not a	Same as the previous example, using Visual Basic notation.	*true*
! a	Same as the previous example, using JavaScript notation.	*true*
Len(s) < 20 And Not IsBlank(s)	Tests whether the length of **s** is less than 20 and whether it isn't a **blank** value. The length is less than 20, and the value	*true*

Formula	Description	Result
	isn't blank. Therefore, the result is *true*.	
Or(Len(s) < 10, x < 100, y < 100)	Tests whether the length of **s** is less than 10, whether **x** is less than 100, and whether **y** is less than 100. The first and third arguments are false, but the second one is true. Therefore, the function returns *true*.	*true*
Not IsBlank(s)	Tests whether **s** is *blank*, which returns *false*. **Not** returns the opposite of this result, which is *true*.	*true*

Feedback

Was this page helpful? Yes No

Provide product feedback

Launch and Param functions

Article • 03/04/2025

Applies to: Canvas apps

Launches a webpage or a canvas app and provides access to launch parameters.

Launch

Launches a webpage or a canvas app. The function supports:

- **Address** (required), the URL of the webpage of the canvas app.
- **Parameters** (optional), named values to pass to the webpage or canvas app. In a canvas app, parameters can be read with the **Param** function.
- **Target** (optional), the browser tab in which to launch the webpage or canvas app.

Launch can only be used in behavior formulas.

> **Note**
>
> - If the Launch formula is used to open Power Apps and the app isn't visible to the end user, it's recommended to check if the app is blocked by a browser pop-up blocker or if the browser is wait for the user's response to a dialog box that asks whether they want to navigate to a new screen.
> - Only **https**: URIs are supported when using Launch in the Power Apps Visual in PowerBI.

Address

Webpages are launched via a URL address. For example:

Power Fx

```
Launch( "https://bing.com" )
```

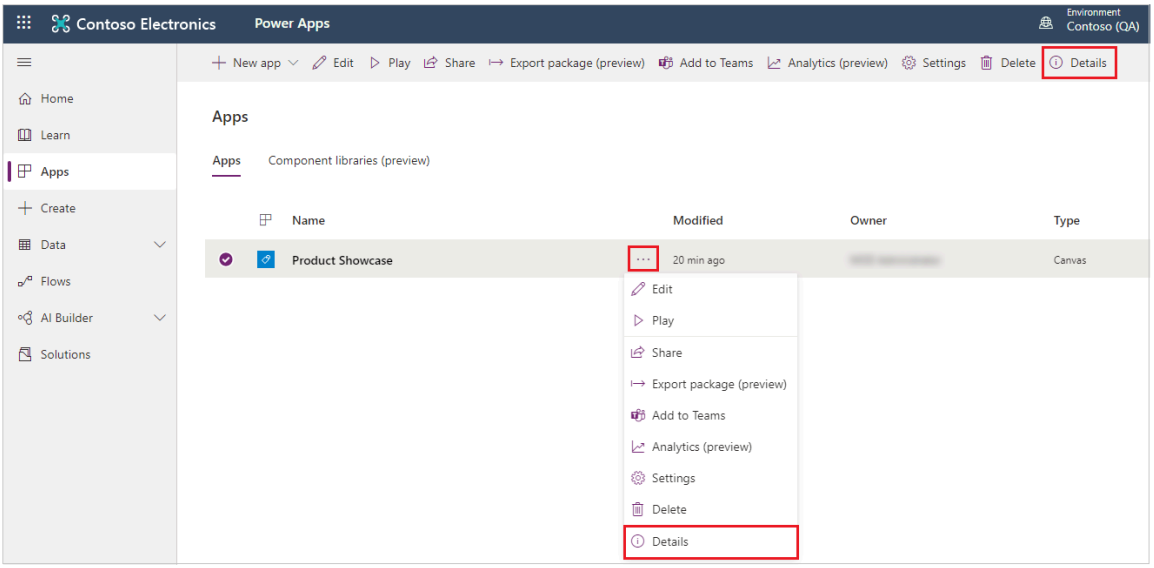
You can launch canvas apps with **Web link** or **App URI** (app id prefixed with `/providers/Microsoft.PowerApps/apps/`). To find these values for an app:

1. Go to Power Apps.

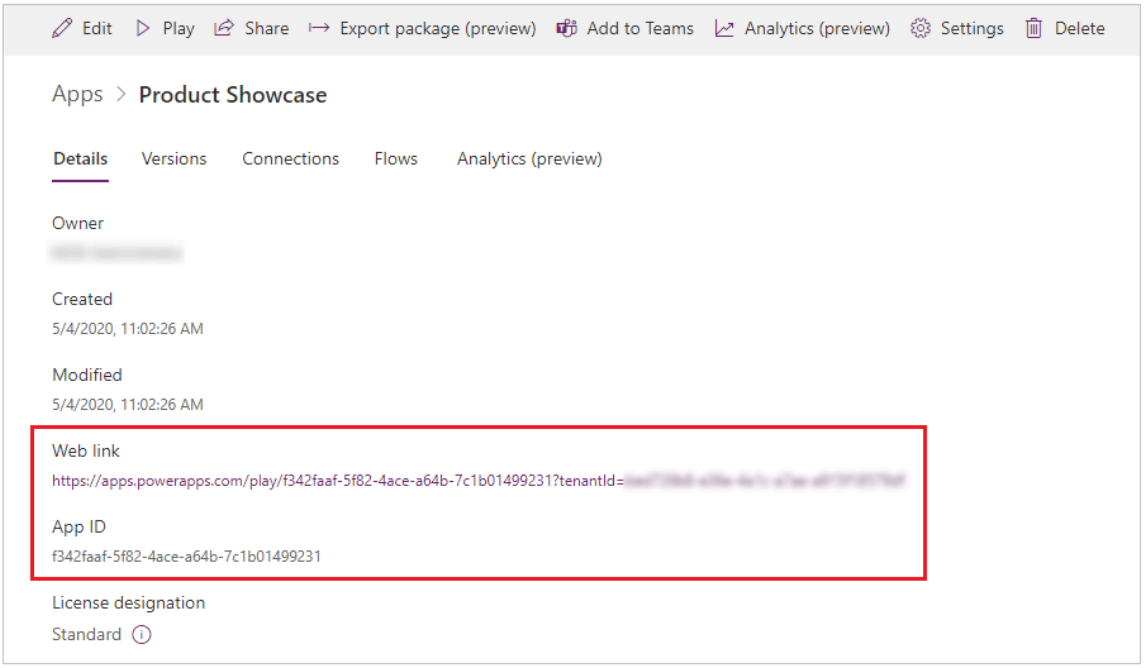
2. Select **Apps** from left navigation pane.

3. Select your app.

4. Select **Details** from top menu.
   You can also select **...** (**More Commands**) and then select **Details** from the drop-down menu.



5. Copy **Web link**.



The **Web link** can be used in any web page and will launch the canvas app. It can also be used with the **Launch** function. For example:

```
Power Fx
```

```
Launch( "https://apps.powerapps.com/e/{environment id}/a/{app id}?tenantId=
{tenantId}" )
```

Native apps on a device can't be launched directly. There may be indirect options available on some platforms, such as a native app installing a custom URL scheme or registering with the web browser to offer an option for specific web sites.

Parameters

Launch can pass parameters to the webpage or canvas app. Parameters can be provided in two ways:

- An argument list of name value pairs. For example:

  Power Fx

  ```
  Launch( "https://bing.com/search", "q", "Power Apps", "count", 1 )
  ```

- A record of field values. For example:

  Power Fx

  ```
  Launch( "https://bing.com/search", { q: "Power Apps", count: 1 } )
  ```

  This form can be easier to work with as it makes the association between name and value clearer. It's the only form that supports the optional *LaunchTarget* argument.

The address and parameters are URL encoded before being passed to replace certain non-alphanumeric characters with `%` and a hexadecimal number as if the **EncodeUrl** function has been used on each.

When launching a webpage, a query string of parameters can be included at the end of the URL address. Any additional parameters provided to **Launch** will be added to the end of the query string. Query strings don't work when launching a canvas app.

When launching an app on a mobile device where that app is already running, parameters will not be refreshed in the running app. An app reload is required for parameters to be refreshed.

Target

Use the *LaunchTarget* argument to specify the target browser window in which to open the webpage or app. Use one of the following **LaunchTarget** enum values or provide a custom window *name*.

Expand table

LaunchTarget enum	Description
New	The webpage or app is opened in a new window or tab.
Replace	The webpage or app replaces the current window or tab.
name	Instead of an enum value, use your own text string to *name* the window or tab. *Self* is an internal only name that is only used by the **Launch** function. It has no impact on nor will it match the title of the window that your users see. If a window or tab with the given *name* already exists, its contents will be replaced. Otherwise, a new window or tab will be created. *name* can't begin with the underscore character "_".

New is the default enum when running in a web browser with **Replace** and *name* as available options. In a mobile player, **New** is the default for webpages with *name* as an available option; while the current canvas app will always be replaced by another canvas app.

> **Note**
>
> - Using a *LaunchTarget* with any value other than **New** in embedded scenarios (for example, Power BI or SharePoint) is not supported and may result in unexpected behavior. In the future, this behavior may change, or may cause an error.

Param

The **Param** function retrieves a parameter passed to the app when it was launched. If the named parameter wasn't passed, **Param** returns *blank*.

- When launching a canvas app from another canvas app, use the *Parameter* arguments to the **Launch** function. Parameter names and values will be automatically URL encoded.
- When launching a canvas app from a web page, add parameters to the query string of the canvas app web link. This involves adding `&parametername=parametervalue` assuming the query string has already been

started for the `tenantId`. For example, adding `&First%20Name=Vicki&category=3` would pass two parameters: `First Name` with a value of `"Vicki"` and `category` with a value of `"3"` (value type is *text*). The parameter name and value must be URL encoded if they contain spaces or special characters, similar to using the **EncodeURL** function.

- Param names are case-sensitive.
- Param names and values will be automatically URL decoded for use in your app.
- Parameter values do not change unless the app is reloaded. Using **Launch** on a mobile device where the app is already running does not refresh the parameters.
- Even if the parameter contains a number, the type returned by **Param** will always be a text string. Conversion to other types will automatically occur or use explicit conversions such as the **Value** function to convert explicitly to a number.

> ⓘ **Note**
>
> For custom pages, the only parameters accepted by the page are: recordId and entityName.

Syntax

Launch(*Address* [, *ParameterName1*, *ParameterValue1*, ...])

- *Address* – Required. The address of a webpage or the ID of an app to launch.
- *ParameterName(s)* – Optional. Parameter name.
- *ParameterValue(s)* – Optional. Corresponding parameter values to pass to the app or the webpage.

Launch(*Address*, { [*ParameterName1*: *ParameterValue1*, ...] } [, *LaunchTarget*])

- *Address* – Required. The address of a webpage or the ID of an app to launch.
- *ParameterName(s)* – Optional. Parameter name.
- *ParameterValue(s)* – Optional. Corresponding parameter values to pass to the app or the webpage.
- *LaunchTarget* – Optional. A **LaunchTarget** enum value or a custom *name*.

Param(*ParameterName*)

- *ParameterName* - Required. The name of the parameter passed to the app.

Reserved parameters

The following keywords are reserved (regardless of case) for internal use, and shouldn't be used as a custom parameter in the **Param()** function:

- amp%3Bauthmode
- amp%3Benableonbehalfof
- amp%3Bhidenavbar
- amp%3Blocale
- appmetadataversion
- authmode
- channeltype
- cordovapath
- correlationid
- debug
- delegatelaunchurl
- delegatelaunchurl
- disablepreviewredirect
- embedderorigin
- enableonbehalfof
- groupid
- hideappsplash
- hidenavbar
- hint
- hostclienttype
- hostmode
- iframecontainerid
- isfullscreen
- ispreviewmode
- loader
- loaderType
- locale
- location
- packagekind
- packageproperties
- playerresourcespath
- playersessionid
- powerappslanguage
- screencolor
- sdkversion
- site
- skipappmetadata
- skipiframecreation

- skiplaunchappcache
- source
- sourcetime
- standaloneconsent
- teamid
- teamtype
- tenantId
- theme
- uselocalpackagehostresources
- userteamrole

Examples

Simple Launch

From a canvas app to a web page:

Expand table

Formula	Description
Launch("http://bing.com/search", "q", "Power Apps", "count", 1)	Opens the webpage **https://bing.com/search?q=Power%20Apps&count=1**. A new window or tab is opened.
Launch("http://bing.com/search", { q: "Power Apps", count: 1 })	The same as the previous examples using the equivalent record notation. A new window or tab is opened.
Launch("http://bing.com/search", { q: "Power Apps", count: 1 }, LaunchTarget.Replace)	The same as the previous examples, replacing the current window or tab with the result if running in a web browser.
Launch("http://bing.com/search", { q: "Power Apps", count: 1 }, "Search Results")	The same as the previous example, creating or replacing the contents of the window or tab named **Search Results**.

From a canvas app to a canvas app

Update the app ID, screen name, and record number as appropriate.

```
Power Fx
```

```
Launch( "/providers/Microsoft.PowerApps/apps/YOUR-APP-ID",
        { Navigate: "Second Screen", Record: 34 }
)
```

From a web page to a canvas app

Update the app ID, tenant ID, screen name, and record number as appropriate.

HTML

```
<html>
  <body>
    <a
      href="https://apps.powerapps.com/play/e/YOUR-APP-ENVIRONMENT-
ID/a/YOUR-APP-ID?tenantId=YOUR-TENANT-ID&Navigate=Second%20Screen&Record=34"
    >
      Launch canvas app
    </a>
  </body>
</html>
```

Simple Param

Simple launch examples above to launch canvas app from web page or from another canvas app show simple examples for Param function:

Expand table

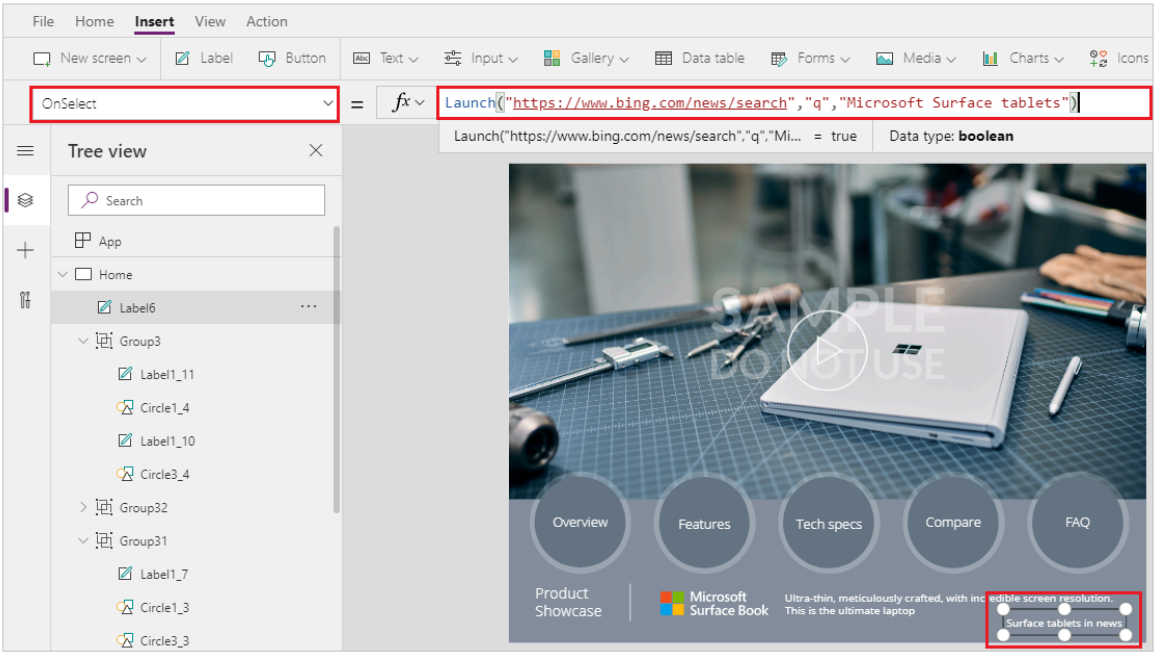
Formula	Description	Result
Param("Navigate")	The **Navigate** parameter was provided when the app was launched and is returned.	"Second Screen"
Param("Record")	The **Record** parameter was provided when the app was launched. Even though it was passed in as a number to the **Launch** function, the result from **Param** will be a text string that can be implicitly or explicitly converted to other types.	"34"
Param("User")	The **User** parameter wasn't provided. A *blank* value is returned that can be tested with the **IsBlank** function.	*blank*

Step by Step examples for Launch and Param

The **Product Showcase** tablet layout template was used for the following examples. To create an app with this template, follow the steps from create an app article and select the **Product Showcase** template. You can also use your own app.

Example - Launch

1. Go to Power Apps.
2. Select **Apps** from left navigation pane.
3. Select your app and then select **Edit**.
4. Select **Insert** from the menu and then select **Label**.
5. Move the label to the bottom right of the screen.
6. From the properties pane on the right-side, select **Color** as *white* and set **Border thickness** at *1*.
7. Select the **Text** property from right-side and enter text as *Surface tablets in news*.
8. From property list on top left, select **OnSelect**.
9. Enter formula as `Launch("https://www.bing.com/news/search","q","Microsoft Surface tablets")`. You can also use any other URL, parameter, and keywords of your choice.



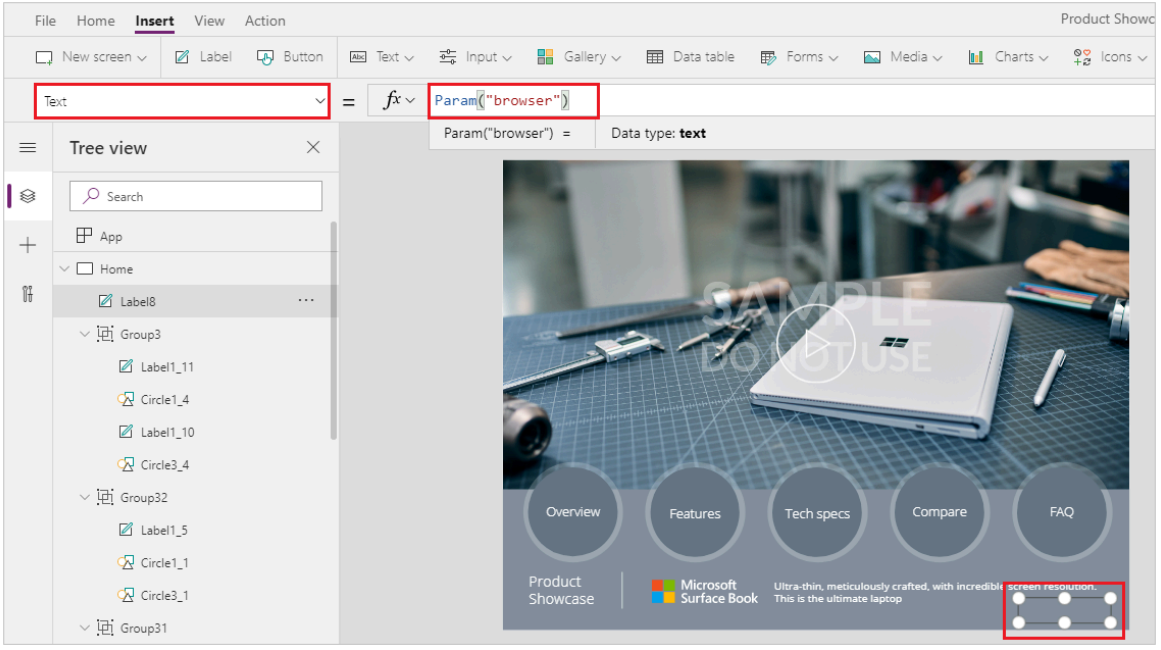
10. Save and publish the app.
11. Play the app.

12. Select label **Surface tablets in news** to launch news search with keywords *Microsoft Surface tablets*.

> **Tip**
>
> For scalability, you can replace the manually entered keywords in Launch function with variables.

Example - Param

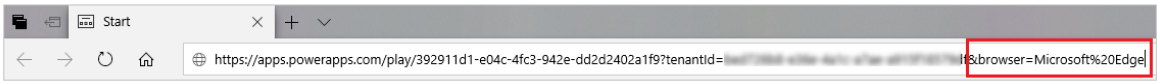
1. Go to Power Apps.
2. Select **Apps** from left navigation pane.
3. Select your app and then select **Edit**.
4. Select **Insert** from the menu and then select **Label**.
5. Move the label to the bottom right of the screen.
6. Select **Text** property for the label from top left.
7. Enter formula as `Param("browser")`. You can also use a different parameter of your choice.



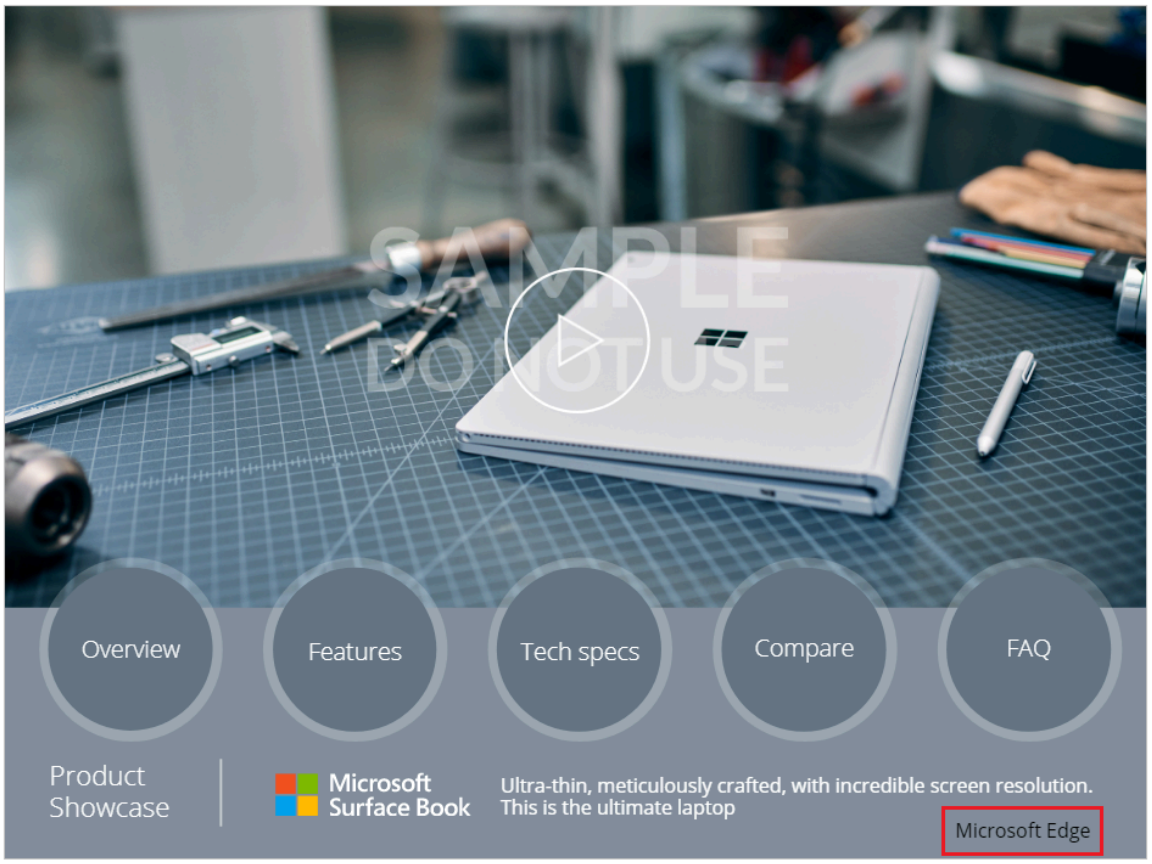
8. Save and publish the app.
9. Copy web link for your app from Power Apps.

10. Open a new browser.

11. Paste the app web link in the browser and append `&browser=Microsoft%20Edge` at the end.



12. When your app launches, the label shows the parameter value passed.

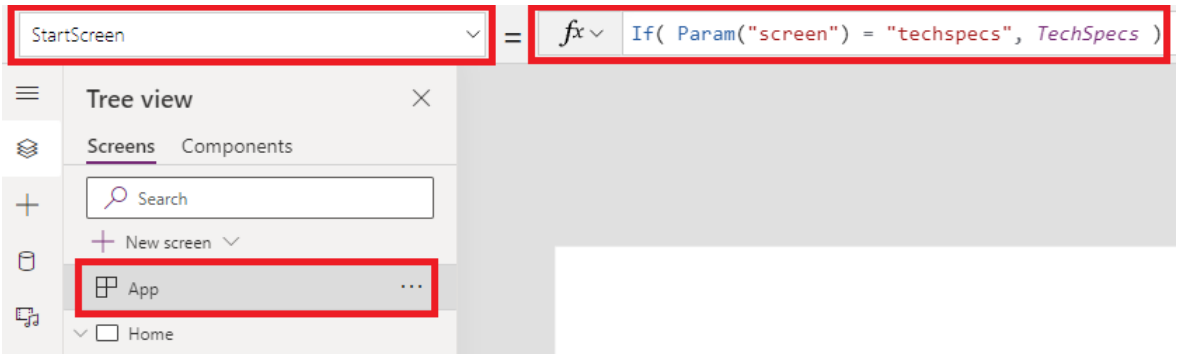


13. Close the app player and edit the app.

14. Select **App** from the Tree view on left navigation.

15. Select **StartScreen** property on top left.

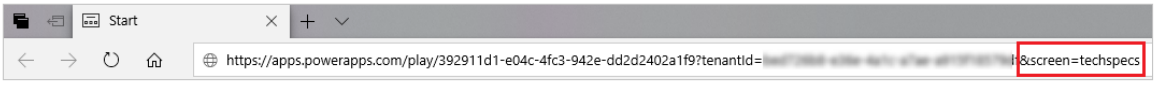
16. Enter the formula as `If( Param("screen") = "techspecs", TechSpecs )`.



If function in StartScreen property checks if parameter equals a certain value, in this case the value *techspecs*. And if it matches, returns the *TechSpecs* screen control to the **StartScreen** property.

> **Note**
>
> Replace the **TechSpecs** control name in the **If** function with the name of a screen control in your own app if you're not using the **Product Showcase** app template.

17. Save and publish the app.
18. Open a new browser.
19. Paste the app web link in the browser and append `&screen=techspecs` at the end.



20. The app directly launches with **TechSpecs** as the startscreen.

See also

Canvas app formula reference

Feedback

Was this page helpful? Yes No

Provide product feedback

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

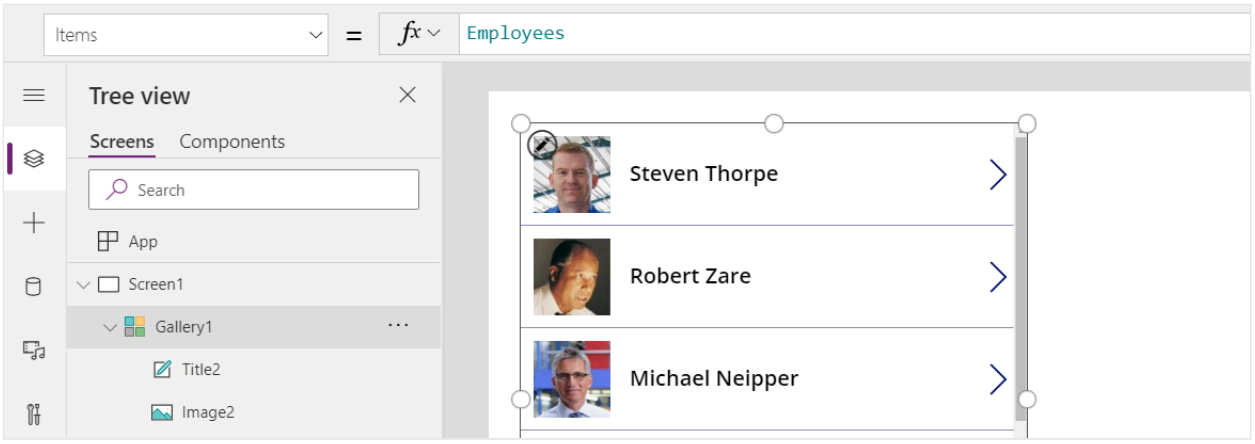
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

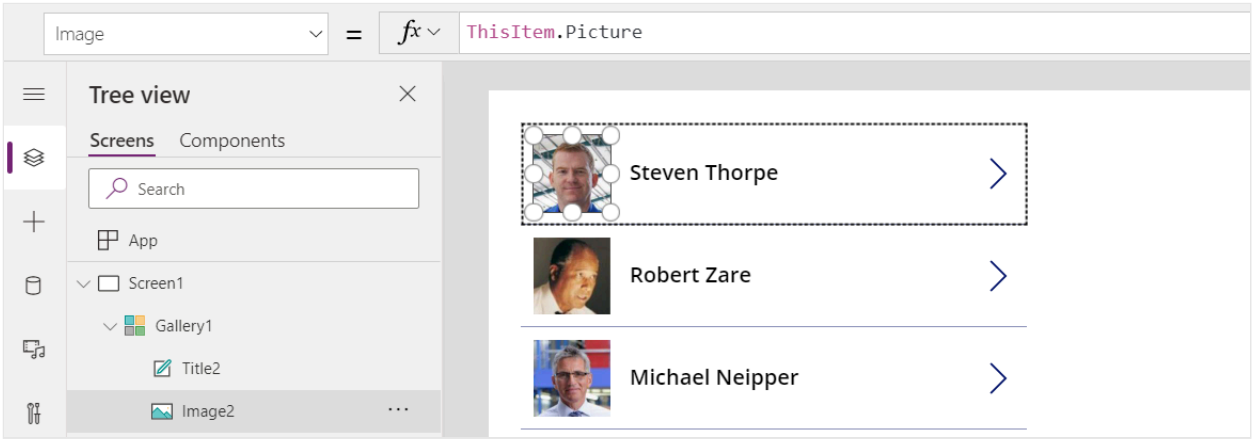
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

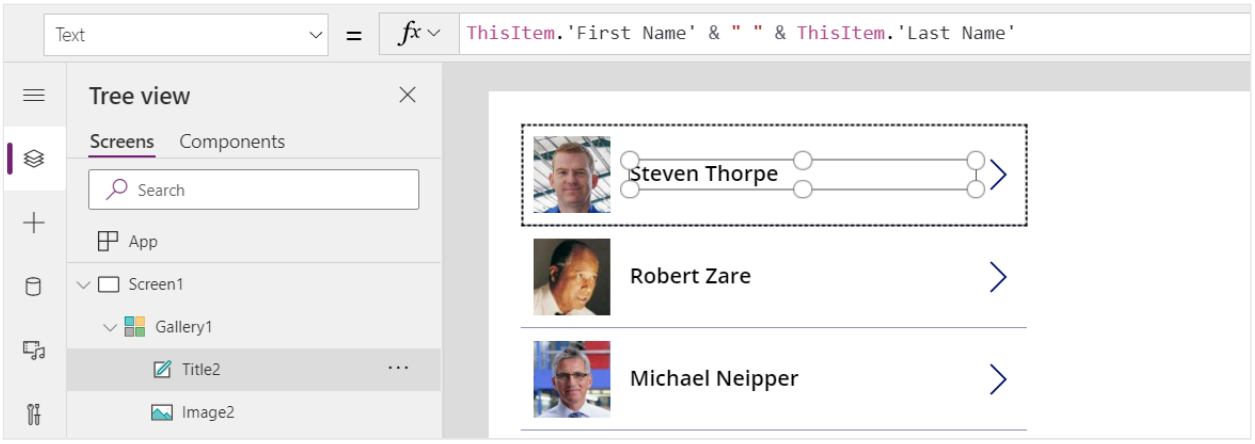
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

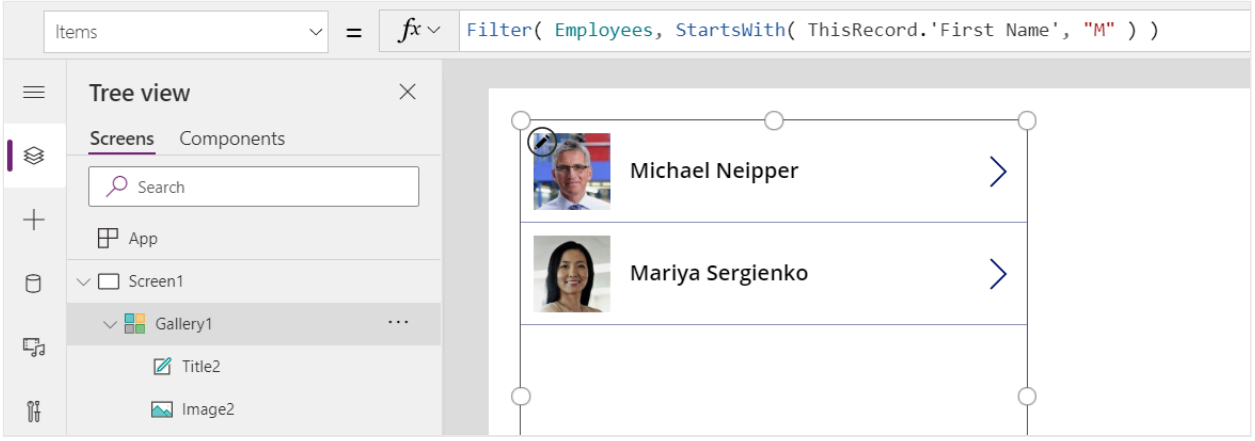


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

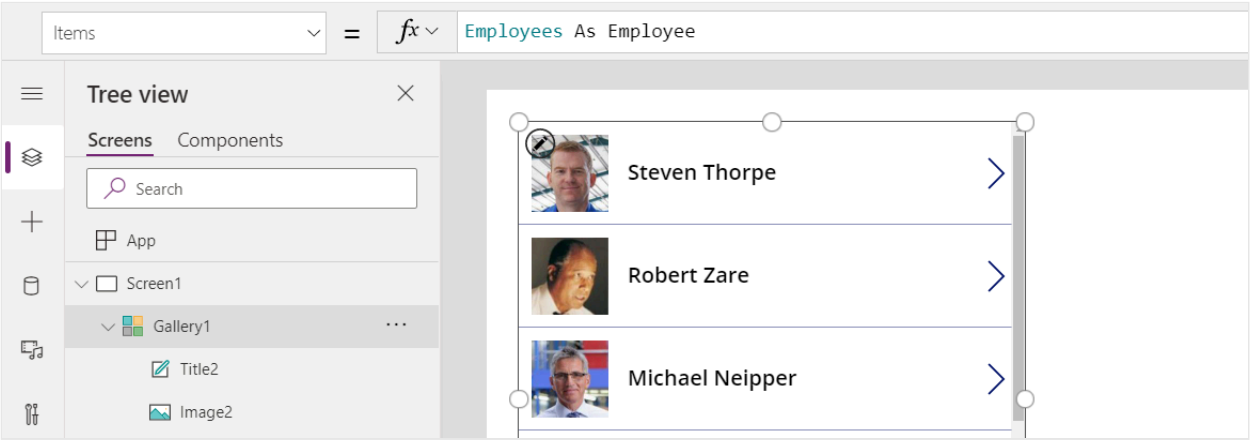
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

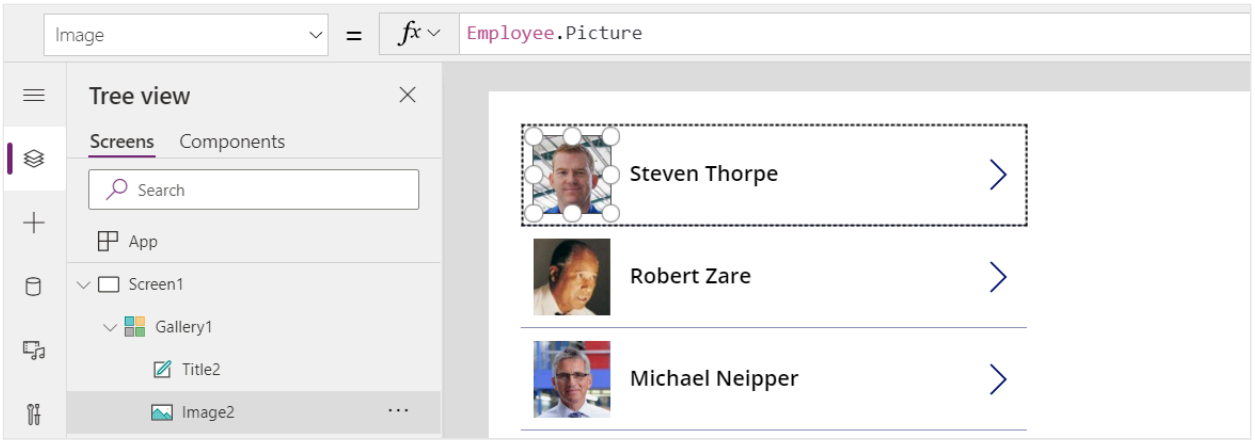
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

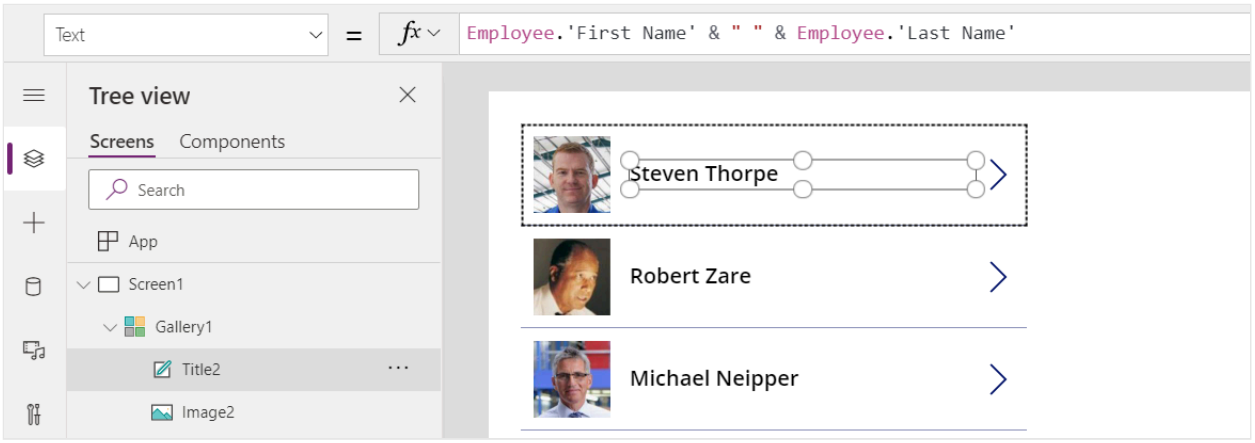
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

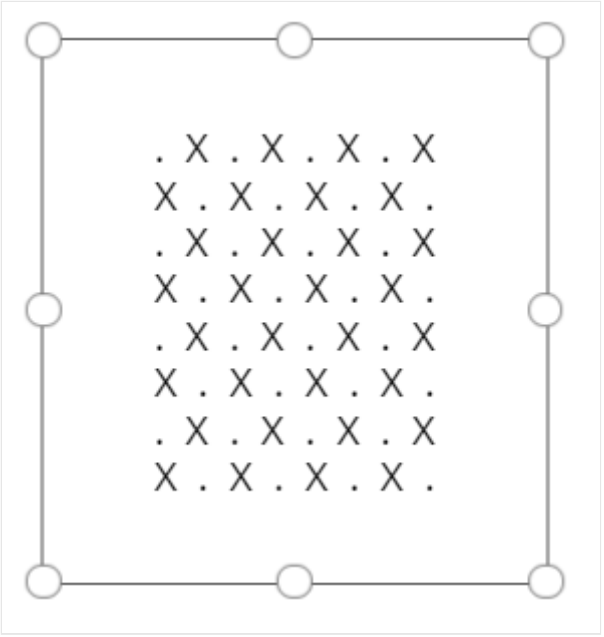
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



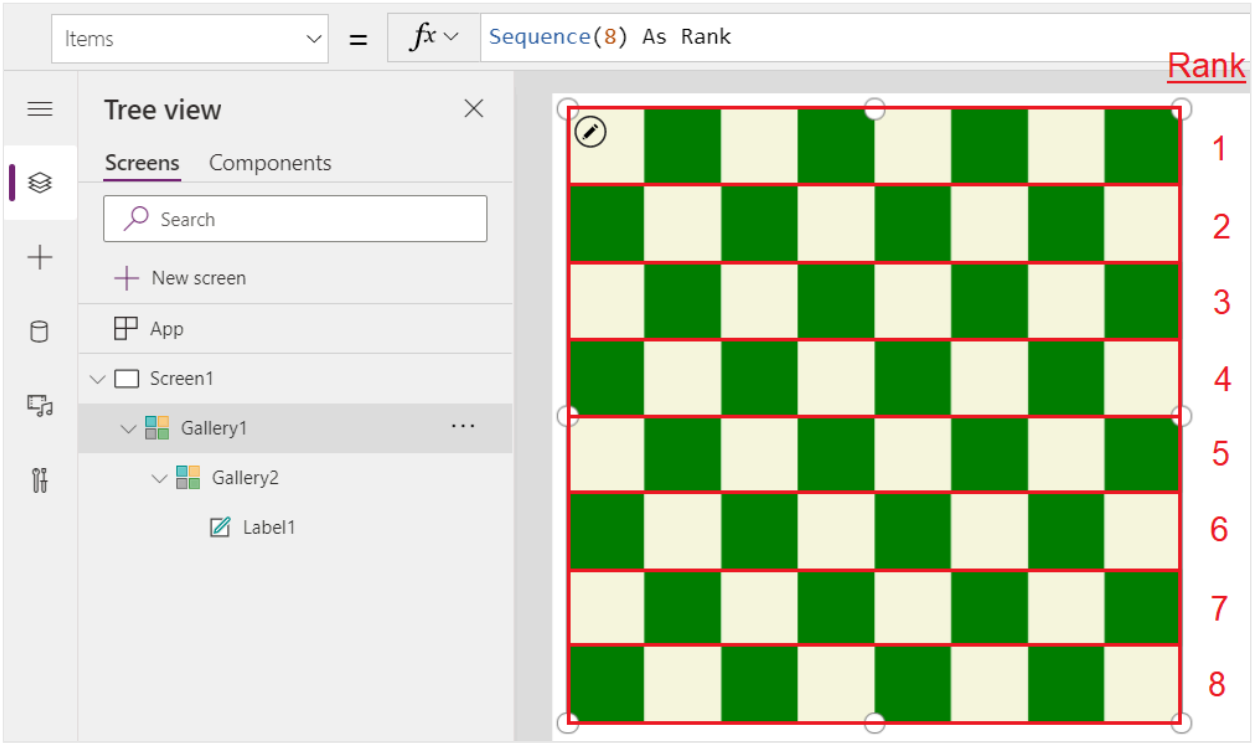
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

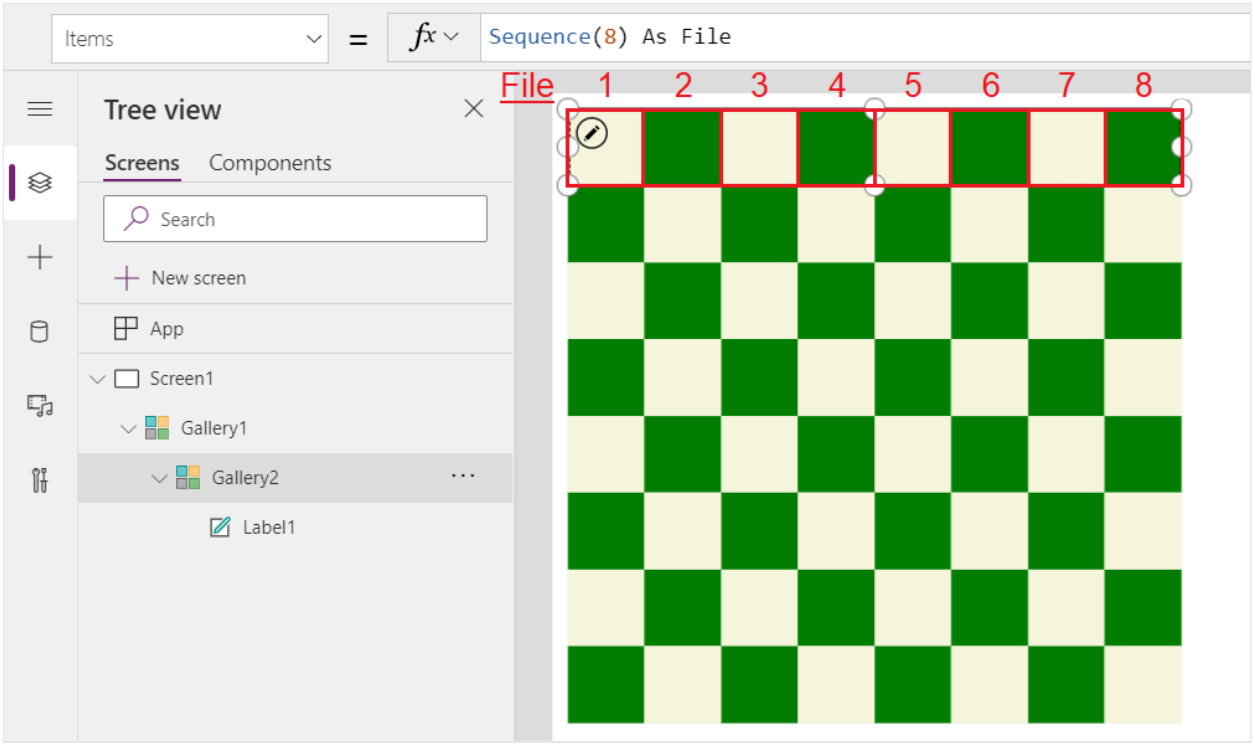
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

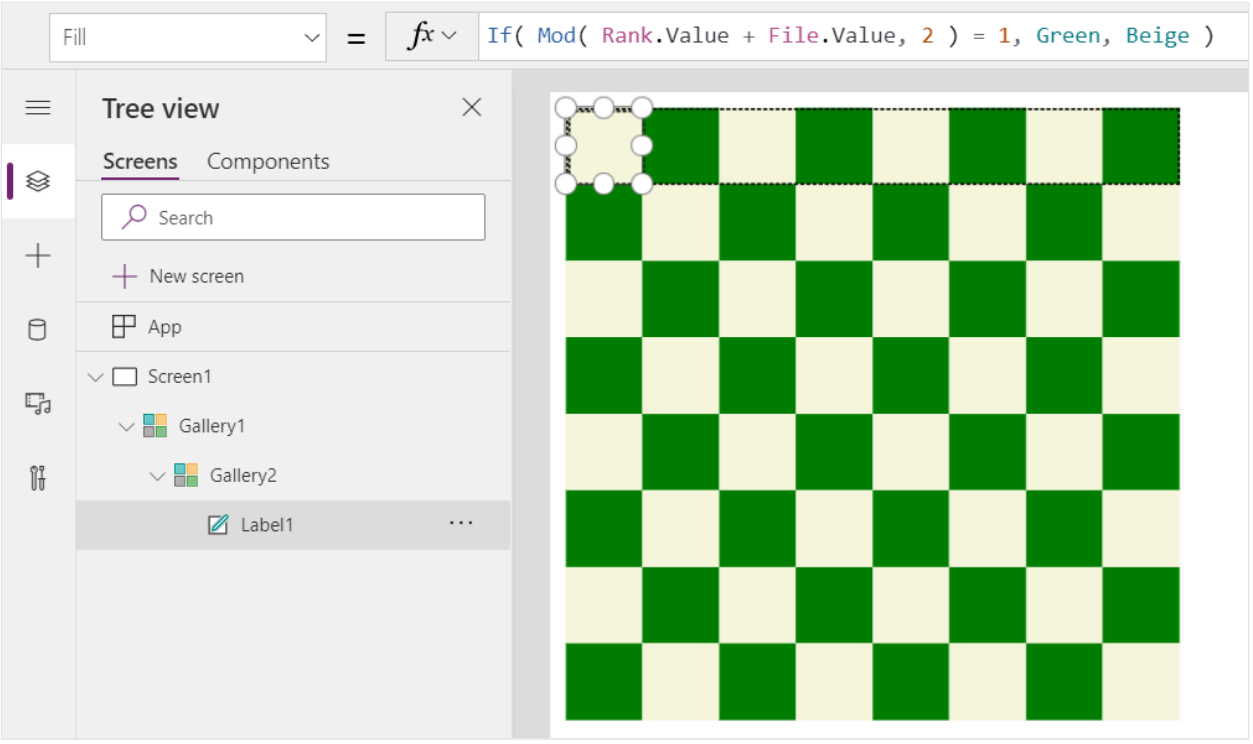
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

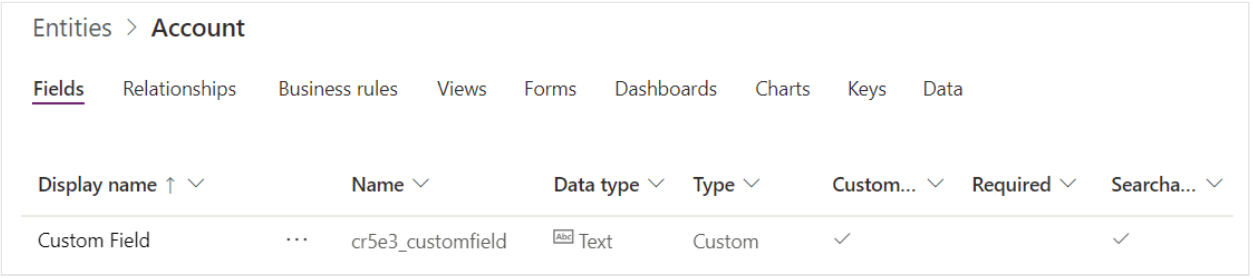
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

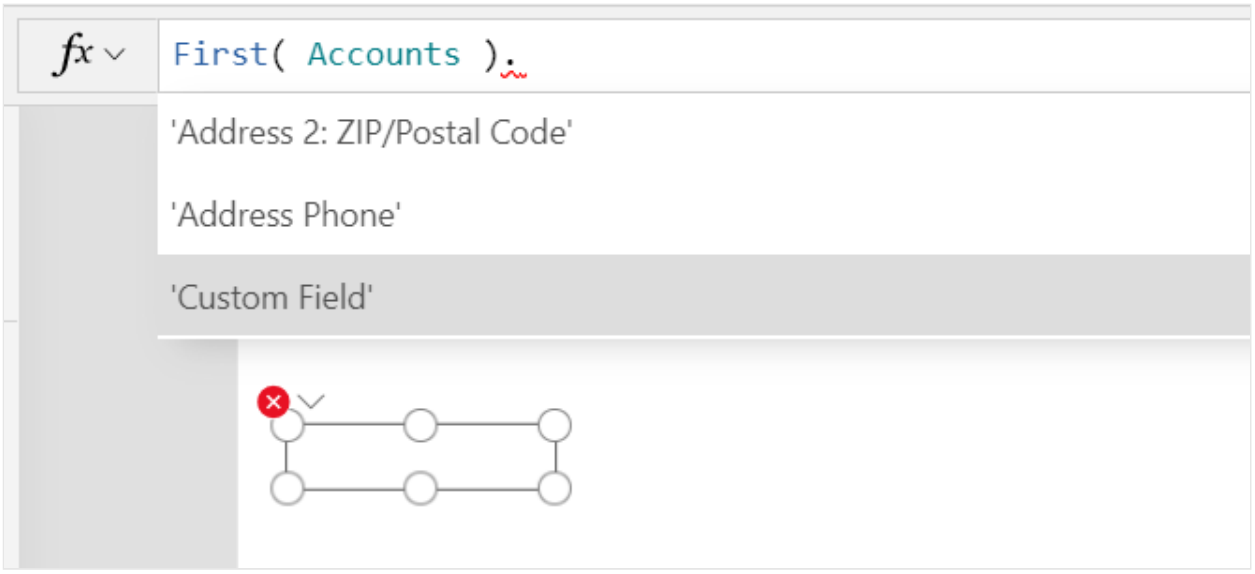
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

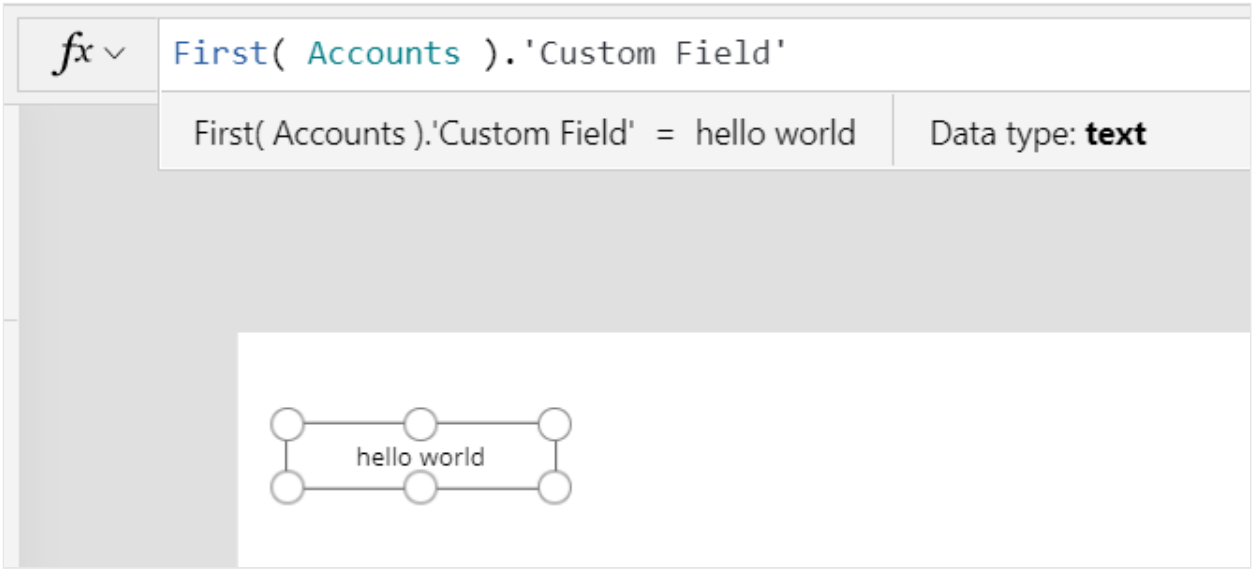
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



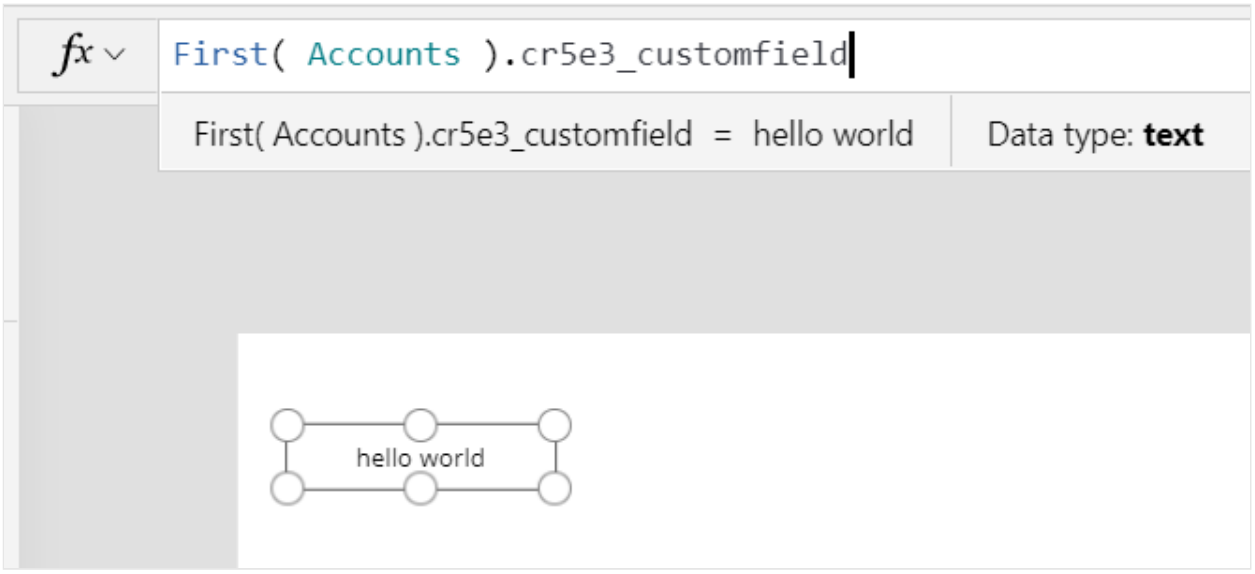
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

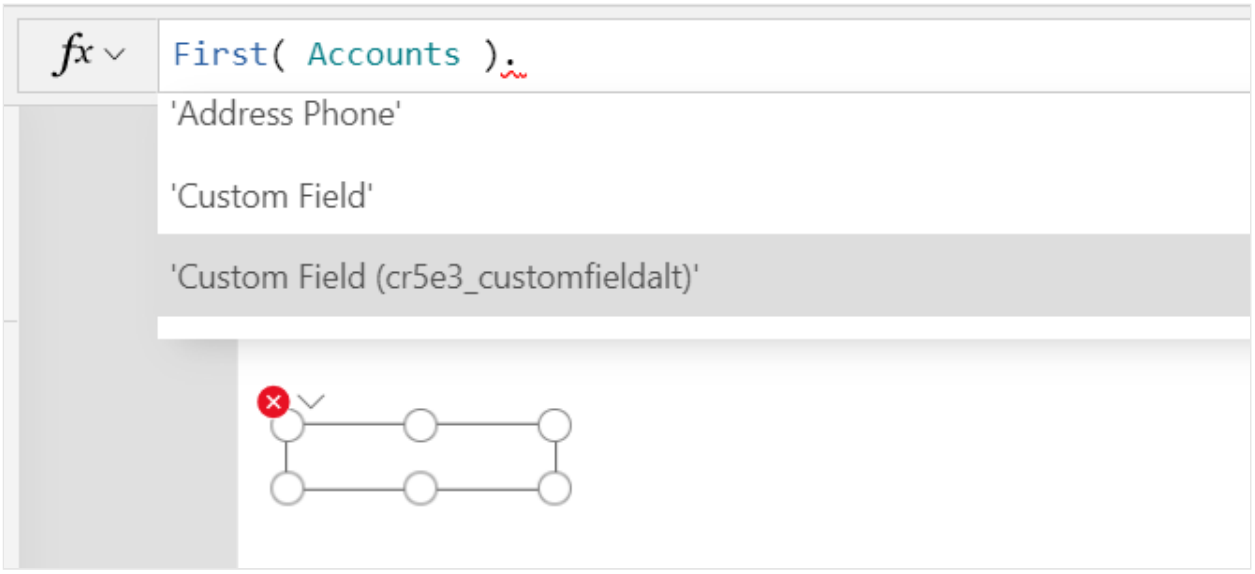


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

Patch function

06/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Modifies or creates one or more records in a data source, or merges records outside of a data source.

Use the **Patch** function to modify records in complex situations, such as when you do updates that require no user interaction or use forms that span multiple screens.

To update records in a data source more easily for simple changes, use the **Edit form** control instead. When you add an **Edit form** control, you provide users with a form to fill in and then save the changes to a data source. For more information, see Understand data forms.

Watch this video to learn how to use the Patch function:
https://learn-video.azurefd.net/vod/player?id=4f09e796-64f3-4895-89d9-70c8462dac37&locale=en-us&embedUrl=%2Fpower-platform%2Fpower-fx%2Freference%2Ffunction-patch

Overview

Use the **Patch** function to modify one or more records of a data source. The values of specific fields are modified without affecting other properties. For example, this formula changes the phone number for a customer named Contoso:

```
Patch( Customers, LookUp( Customers, Name = "Contoso" ), { Phone: "1-212-555-1234" } )
```

Use **Patch** with the **Defaults** function to create records. Use this behavior to build a single screen for both creating and editing records. For example, this formula creates a record for a customer named Contoso:

```
Patch( Customers, Defaults( Customers ), { Name: "Contoso" } )
```

> **Note**
>
> When you patch a collection using a record from a data source with default values, the patch operation updates the collection with both the specified patch values and the default values from the data source. The DataSource of the patch statement and the DataSource of the Defaults function must match in order to create a new record.

Even if you're not working with a data source, you can use **Patch** to merge two or more records. For example, this formula merges two records into one that identifies both the phone number and the location for Contoso:

```
Patch( { Name: "Contoso", Phone: "1-212-555-1234" }, { Name: "Contoso", Location: "Midtown" } )
```

Description

Modify or create a record in a data source

To use this function with a data source, specify the data source, and then specify a base record:

- To modify a record, the base record needs to have come from a data source. The base record might have come through a gallery's **Items** property, been placed in a context variable, or come through some other path. But, you can trace the base record back to the data source. This is important as the record includes additional information to help find the record again for modification.
- To create a record, use the **Defaults** function to create a base record with default values.

Then specify one or more change records, each of which contains new property values that override property values in the base record. Change records are processed in order from the beginning of the argument list to the end, with later property values overriding earlier ones.

The return value of **Patch** is the record that you modified or created. If you created a record, the return value might include properties that the data source generated automatically. However, the return value doesn't provide a value for fields of a related table.

For example, you use `Set(MyAccount, Patch(Accounts, First(Account), 'Account Name': "Example name"));` and then `MyAccount.'Primary Contact'.'Full Name'`. You can't yield a full name in this case. Instead, to access the fields of a related table, use a separate lookup such as:

Power Fx

```
LookUp(Accounts, Account = MyAccount.Account).'Primary Contact'.'Full Name'
```

When you update a data source, one or more issues might arise. Use **IfError** and **IsError** with the return value from **Patch** to detect and respond to errors, as Error Handling describes. You can also use the **Errors** function to identify and examine issues, as Working with Data Sources describes.

Related functions include the **Update** function to replace an entire record, and the **Collect** function to create a record. Use the **UpdateIf** function to modify specific properties of multiple records based on a condition.

Modify or create a set of records in a data source

Patch can also be used to create or modify multiple records with a single call.

Instead of passing a single base record, a table of base records can be provided in the second argument. Change records are provided in a table as well, corresponding one-for-one with the base records. The number of records in each change table must be the same as the number of records in the base table.

When using **Patch** in this manner, the return value is also a table with each record corresponding one-for-one with the base and change records.

Merge records outside of a data source

Specify two or more records that you want to merge. Records are processed in the order from the beginning of the argument list to the end, with later property values overriding earlier ones.

Patch returns the merged record and doesn't modify its arguments or records in any data sources.

Syntax

Modify or create a record in a data source

Patch(*DataSource*, *BaseRecord*, *ChangeRecord1* [, *ChangeRecord2*, ...])

- *DataSource* – Required. The data source that contains the record that you want to modify or will contain the record that you want to create.
- *BaseRecord* – Required. The record to modify or create. If the record came from a data source, the record is found and modified. If the result of **Defaults** is used, a record is created. The DataSource of the patch statement and the DataSource of the Defaults function must match in order to create a new record.
- *ChangeRecord(s)* – Required. One or more records that contain properties to modify in the *BaseRecord*. Change records are processed in order from the beginning of the argument list to the end, with later property values overriding earlier ones.

Modify or create a set of records in a data source

Patch(*DataSource*, *BaseRecordsTable*, *ChangeRecordTable1* [, *ChangeRecordTable2*, ...])

- *DataSource* – Required. The data source that contains the records that you want to modify or will contain the records that you want to create.
- *BaseRecordTable* – Required. A table of records to modify or create. If the record came from a data source, the record is found and modified. If the result of **Defaults** is used, a record is created. The DataSource of the patch statement and the DataSource of the Defaults function must match in order to create a new record.
- *ChangeRecordTable(s)* – Required. One or more tables of records that contain properties to modify for each record of the *BaseRecordTable*. Change records are processed in order from the beginning of the argument list to the end, with later property values overriding earlier ones.

Merge records

Patch(*Record1*, *Record2* [, ...])

- *Record(s)* - Required. At least two records that you want to merge. Records are processed in order from the beginning of the argument list to the end, with later property values overriding earlier ones.

Examples

Modify or create a record (in a data source)

In these examples, you'll modify or create a record in a data source, named **IceCream**, that contains the data in this table and automatically generates the values in the **ID** column:

ID (auto generated)	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200

Expand table

Formula	Description	Result
Patch(IceCream, LookUp(IceCream, Flavor = "Chocolate"), { Quantity: 400 })	Modifies a record in the **IceCream** data source:	{ ID: 1, Flavor: "Chocolate", Quantity: 400 } The **Chocolate** entry in the

Formula	Description	Result
	• The **ID** column of the record to modify contains the value of **1**. (The **Chocolate** record has that ID.) • The value in the **Quantity** column changes to **400**.	**IceCream** data source has been modified.
Patch(IceCream, Defaults(IceCream), { Flavor: "Strawberry" })	Creates a record in the **IceCream** data source: • The **ID** column contains the value **3**, which the data source generates automatically. • The **Quantity** column contains **0**, which is the default value for that column in the **IceCream** data source, as the Defaults function specifies. • The **Flavor** column contains the value of **Strawberry**.	{ ID: 3, Flavor: "Strawberry", Quantity: 0 } The **Strawberry** entry in the **IceCream** data source has been created.

After the previous formulas have been evaluated, the data source ends with these values:

ID (auto generated)	Flavor	Quantity
1	"Chocolate"	400
2	"Vanilla"	200
3	"Strawberry"	0

Merge records (outside of a data source)

Expand table

Formula	Description	Result
Patch({ Name: "James", Score: 90 }, { Name: "Jim", Passed: true })	Merges two records outside of a data source: • The values in the **Name** column of each record don't match. The result contains the value (**Jim**) in the record that's closer to the end of the argument list instead of the value (**James**) in the record that's closer to the start.	{ Name: "Jim", Score: 90, Passed: true }

Formula	Description	Result
	• The first record contains a column (**Score**) that doesn't exist in the second record. The result contains that column with its value (**90**). • The second record contains a column (**Passed**) that doesn't exist in the first record. The result contains that column with its value (**true**).	

Use of As or ThisRecord

Using the **As** or **ThisRecord** keyword in the formula avoids ambiguous evaluation context.

In the example below, consider the first lookup in the `If` statement. `(OrderID = A[@OrderID])` is expected to compare the `OrderId` in the lookup scope with the `OrderId` of collection `A` in the `ForAll` scope. In this case, you likely want `A[@OrderId]` to be resolved as a local parameter. But it is ambiguous.

Power Apps currently interprets both the left-hand side `OrderId` and right-hand side `A[@OrderId]` as a field in the lookup scope. Therefore, lookup will always find the first row in `[dbo].[Orders1]` because the condition is always true (that is, any row's `OrderId` is equal to itself.)

Power Fx

```
ClearCollect(
    A,
    Filter(
        '[dbo].[Orders1]',
        OrderId = 8888888
    )
);
ForAll(
    A,
    If(
        LookUp(
            '[dbo].[Orders1]',
            OrderId = A[@OrderId],
            "OK"
        ) = "OK",
        Patch(
            '[dbo].[Orders1]',
            LookUp(
                '[dbo].[Orders1]',
                OrderId = A[@OrderId]
            ),
```

```
            {
      OrderName: "val1"
       }
   ),
   Patch(
            '[dbo].[Orders1]',
            Defaults('[dbo].[Orders1]'),
            {
      OrderName: "val2"
       }
   )
    )
)
```

Using As or ThisRecord

Whenever possible use the **As** operator or the **ThisRecord** to disambiguate the left-hand side. **As** is recommended for the above scenario.

When your formula uses multiple scopes with `ForAll`, `Filter`, and `Lookup` on the same data source or table, it is possible that the scope parameters may collide with a same field elsewhere. Therefore, it is recommended to use the **As** operator or **ThisRecord** to resolve the field name and avoid ambiguity.

For example, you can use the **As** operator to disambiguate in the example below.

Power Fx

```
ClearCollect(
    A,
    Filter(
        '[dbo].[Orders1]',
        OrderId = 8888888
    )
);
ForAll(
    A,
    If(
        LookUp(
            '[dbo].[Orders1]' As B,
            B.OrderId = A[@OrderId],
            "OK"
        ) = "OK",
        Patch(
            '[dbo].[Orders1]',
            LookUp(
                '[dbo].[Orders1]' As C,
                C.OrderId = A[@OrderId]
            ),
            {
```

```
      OrderName: "val1"
       }
   ),
   Patch(
            '[dbo].[Orders1]',
            Defaults('[dbo].[Orders1]'),
            {
      OrderName: "val2"
       }
   )
    )
)
```

Alternatively, you can use **ThisRecord** for the same purpose.

Power Fx

```
ClearCollect(
    A,
    Filter(
        '[dbo].[Orders1]',
        OrderId = 8888888
    )
);
ForAll(
    A,
    If(
        LookUp(
            '[dbo].[Orders1]',
            ThisRecord.OrderId = A[@OrderId],
            "OK"
        ) = "OK",
        Patch(
            '[dbo].[Orders1]',
            LookUp(
                '[dbo].[Orders1]',
                ThisRecord.OrderId = A[@OrderId]
            ),
            {
      OrderName: "val1"
       }
   ),
   Patch(
            '[dbo].[Orders1]',
            Defaults('[dbo].[Orders1]'),
            {
      OrderName: "val2"
       }
   )
    )
)
```

To learn more about the usage of **As** operator and **ThisRecord** see Operators article.

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

EncodeHTML, EncodeUrl, and PlainText functions

Article • 04/20/2025

Expand table

Functions	Applies to
EncodeHTML	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
EncodeUrl **PlainText**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Encodes and decodes strings.

Description

The **EncodeUrl** function encodes a URL string, replacing certain non-alphanumeric characters with % and a hexadecimal number.

The **EncodeHTML** function encodes a string, replacing certain characters that need to be escaped to appear properly in an HTML context. For example, the characters <, >, and & are replaced by <, >, and &, respectively.

The **PlainText** function removes HTML and XML tags, converting certain tags such as these to an appropriate symbol:

- ** **
- **"**

The return value from these functions is the encoded or decoded string. This function doesn't remove all HTML and XML tags.

Syntax

EncodeUrl(*String*)

- *String* - Required. URL to be encoded.

EncodeHTML(*String*)

- *String* - Required. HTML to be encoded.

PlainText(*String*)

- *String* - Required. String from which HTML and XML tags are stripped.

Examples

If you show an RSS feed in a text gallery and then set the **Text** property of a label in that gallery to **ThisItem.description**, the label might show raw HTML or XML code as in this example:

HTML

```
<p>
  We have done an unusually &quot;deep&quot; globalization and
  localization.
</p>
```

If you set the **Text** property of the label to **PlainText(ThisItem.description)**, the text appears as in this example:

```
    We have done an unusually "deep" globalization and localization.
```

With the formula **EncodeHTML(ThisItem.description)**, the HTML tags are encoded so that they show up as text when interpreted as HTML, which can be useful to see the HTML structure:

```
&lt;p&gt;
  We have done an unusually&amp;nbsp;&amp;quot;deep&amp;quot; globalization and
  localization.
&lt;/p&gt;
```

If instead of a label you have an HTML text control in the gallery, and you set the **HtmlText** property of that control to **ThisItem.description** you also see the decoded text, because the HTML text control is interpreting the tags as HTML.

Abs, Exp, Ln, Power, Log, and Sqrt functions

Article • 04/20/2025

Expand table

Functions	Applies to
Abs **Exp** **Ln** **Power** **Sqrt**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Log	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Calculates absolute values, logarithms, square roots, and the results of raising *e* or any number to specified powers.

Description

The **Abs** function returns the non-negative value of its argument. If a number is negative, **Abs** returns the positive equivalent.

The **Exp** function returns *e* raised to the power of its argument. The transcendental number *e* begins 2.7182818...

The **Ln** function returns the natural logarithm (base *e*) of its argument.

The **Power** function returns a number raised to a power. It's equivalent to using the ^ operator.

The **Log** function returns the logarithm of its first argument in the base specified by its second argument (or 10 if not specified).

The **Sqrt** function returns the number that, when multiplied by itself, equals its argument.

If you pass a single number, the return value is a single result based on the function called. If you pass a single-column table that contains numbers, the return value is a single-column table of results in a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined valued, the result is *blank*. Which can happen with square roots and logarithms of negative numbers.

Syntax

Abs(*Number*)
Exp(*Number*)
Ln(*Number*)
Sqrt(*Number*)

- *Number* - Required. Number to operate on.

Power(*Base*, *Exponent*)

- *Base* - Required. Base number to raise.
- *Exponent* - Required. The exponent to which the base number is raised.

Log(*Number*, *Base*)

- *Number* - Required. Number to calculate the logarithm.
- *Base* - Optional. The base of the logarithm to calculate. By default, 10 (when not specified).

Abs(*SingleColumnTable*)
Exp(*SingleColumnTable*)
Ln(*SingleColumnTable*)
Sqrt(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Examples

Single number

Expand table

Formula	Description	Result
Abs(-55)	Returns the number without the negative sign.	55
Exp(2)	Returns *e* raised to the power of 2, or *e* * *e*.	7.389056...
Ln(100)	Returns the natural logarithm (base *e*) of the number 100.	4.605170...
Log(100)	Returns the logarithm in base 10 of the number 100.	2
Log(64, 2)	Returns the logarithm in base 2 of the number 64.	6

Formula	Description	Result
Power(5, 3)	Returns 5 raised to the power of 3, or 5 * 5 * 5.	125
Sqrt(9)	Returns the number that, when multiplied by itself, results in 9.	3

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains this data:

Expand table

Value
9
-4
2

Expand table

Formula	Description	Result
Abs(ValueTable)	Returns the absolute value of each number in the table.	A single-column table with a `Value` column containing the following values: 9, 4, 2
Exp(ValueTable)	Returns *e* raised to the power of each number in the table.	A single-column table with a `Value` column containing the following values: 8103.083927..., 0.018315..., 7.389056...
Ln(ValueTable)	Returns the natural logarithm of each number in the table.	A single-column table with a `Value` column containing the following values: 2.197224..., Blank(), 0.693147...
Sqrt(ValueTable)	Returns the square root of each number in the table	A single-column table with a `Value` column containing the following values: 3, Blank(), 1.414213...

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a **Label** control, and set its **Text** property to this formula:
   Sqrt(Value(Source.Text))

3. Type a number into **Source**, and confirm that the **Label** control shows the square root of the number that you typed.

Print function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Opens the current screen in the default browser print dialog.

Description

The print function allows you to select any screen and fit it to a page in order to send it to a printer for printing or allows you to save it as a PDF file.

The different configurations of the screen enable different printing outcomes. For fixed screens, they fit the size of the page, for use of the screen templates/special sized screens, we will fit the content to the size of the print.

> **Note**
>
> You can only print the screen you have added a button and defined **Print** function on `OnSelect` property of the button. For example, if you have added a button on `screen 2` and when you select the button, only the content in the `screen 2` gets printed.

Syntax

Print()

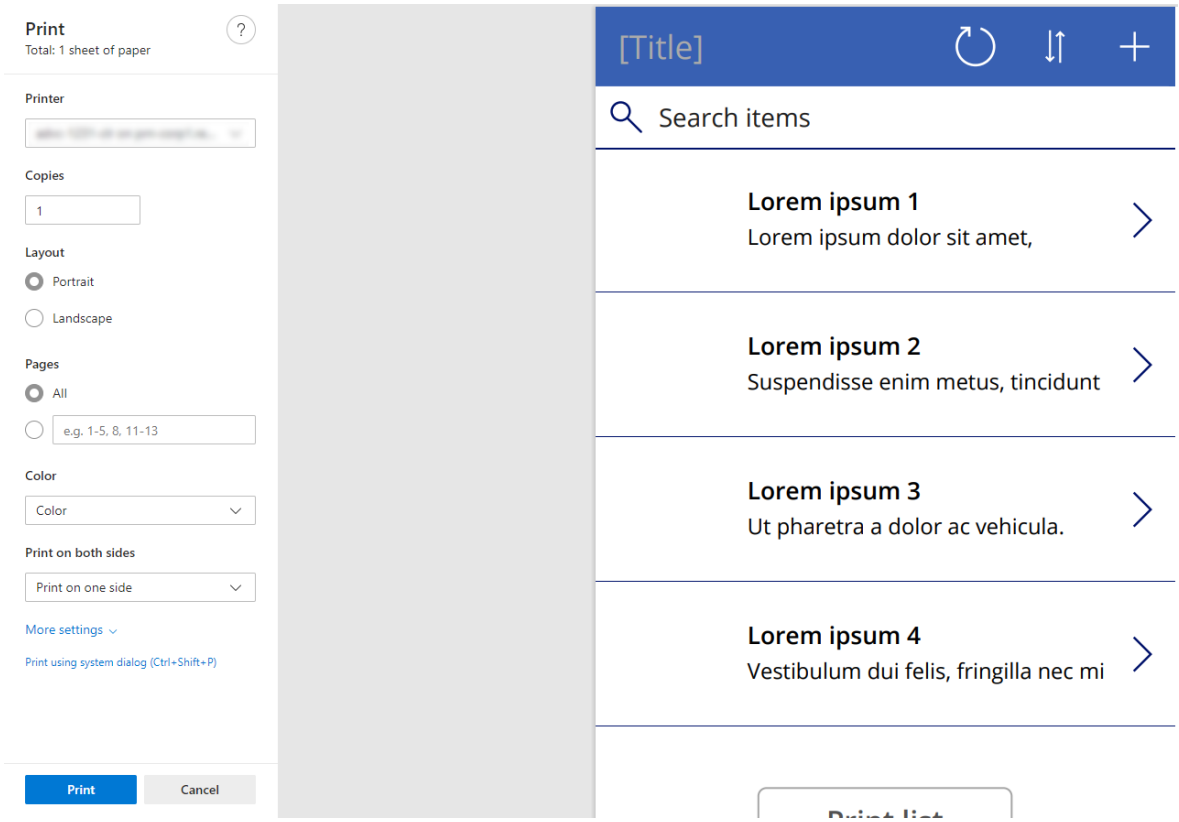
Examples

1. Go to Power Apps.
2. Select **Apps** from the left navigation pane.
3. Select your app or create an app from scratch.
4. Select **Insert** from the menu and then select **Button**.
5. From the property list on the top left, select **OnSelect**.
6. Enter the formula `Print()`.

7. Save and publish the app.

8. Play the app.

9. Select the button that you added. When you select the button, a default print browser pops up and allows you to choose from the available options to print or save it as a PDF file.



Hide controls while printing

To enable more customizable content, you can access the screen when the screen is printing to the change properties of the canvas app. For example, hiding buttons, or changing a form to view mode.

From the above example, where you insert a button on the screen and when you print the screen, you notice that the button is also gets printed. To remove the button from the screen while printing, on the button's **Visible** property add the formula `Not Screen1.Printing`. Preview the app and select the button. Observe that the button does not show in the content in the browser preview print dialog.

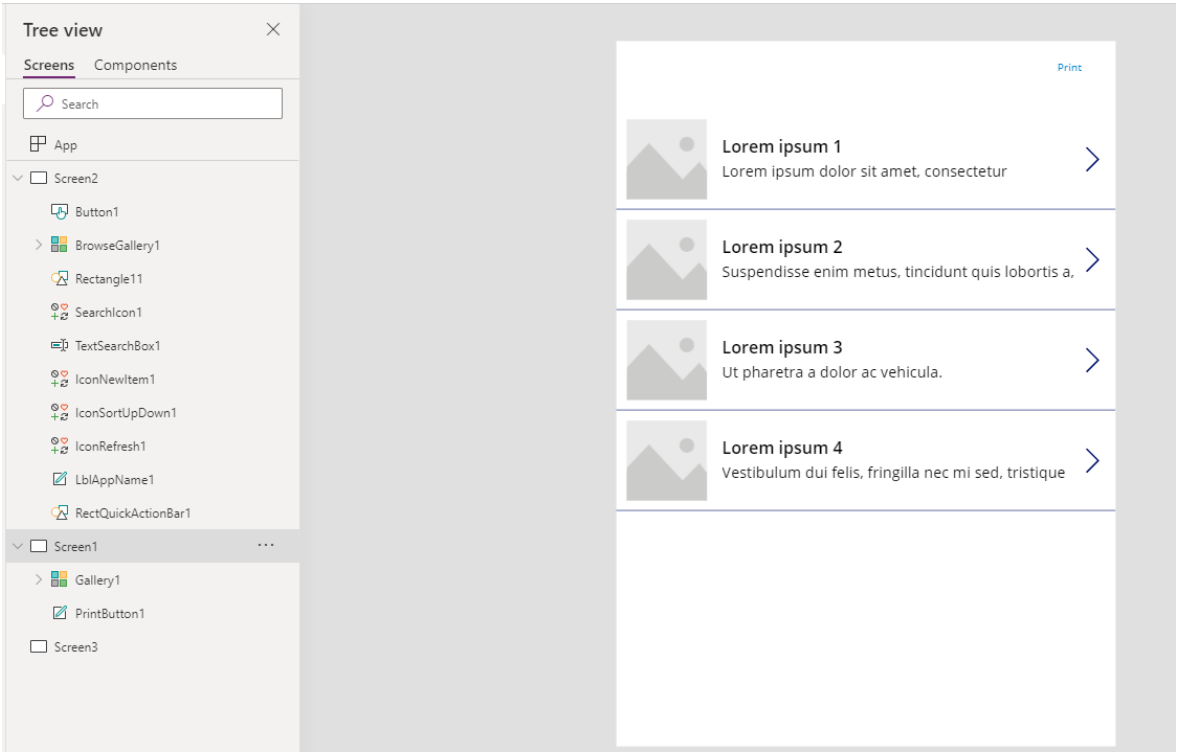
Use a screen template

An easy way to get started is to use a screen template to size your print to the letter size.

- Open or create a new canvas app.
- Select **New Screen** and select **Portrait print** option.



- Observe that the screen is sized differently from other screens with the print button on the top-right corner.



- Add content to this screen.
- Observe that the content is sized to the print. This enables more custom control over the experience.

Screen sizes for common prints

To build out a print for a specific size, you can build a responsive app, or create a special screen to manage your print.

Expand table

Print size	Screen height	Screen width
A4 portrait	1123	794
A4 landscape	794	1123
Letter portrait	1056	816
Letter landscape	816	1056

Known limitations

- The **Print** function currently doesn't work on mobile devices and on SharePoint forms.

- The **Print** function currently doesn't work on custom pages or embedded scenarios.
- The default browser printers are the ones that will be available to print to.
- In some browser print dialogs, a setting called **Background graphics** should be enabled to see all the images and colors in the print.
- The **Print** function can't print multiple pages.

See also

Canvas app formula reference

Lower, Upper, and Proper functions

Article • 04/20/2025

Expand table

Functions	Applies to
Lower **Upper**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Proper	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts letters in a string of text to all lowercase, all uppercase, or proper case.

Description

The **Lower**, **Upper**, and **Proper** functions convert the case of letters in strings.

- **Lower** converts any uppercase letters to lowercase.
- **Upper** converts any lowercase letters to uppercase.
- **Proper** converts the first letter in each word to uppercase if it's lowercase and converts any other uppercase letters to lowercase.

All three functions ignore characters that aren't letters.

If you pass a single string, the return value is the converted version of that string. If you pass a single-column table that contains strings, the return value is a single-column table of converted strings. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

Lower(*String*)
Upper(*String*)
Proper(*String*)

- *String* - Required. The string to convert.

Lower(*SingleColumnTable*)
Upper(*SingleColumnTable*)
Proper(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of strings to convert.

Examples

Single string

The examples in this section use a text-input control, named **Author**, as their data source. The control contains the string "E. E. CummINGS".

Expand table

Formula	Description	Result
Lower(Author.Text)	Converts any uppercase letters in the string to lowercase.	"e. e. cummings"
Upper(Author.Text)	Converts any lowercase letters in the string to uppercase.	"E. E. CUMMINGS"
Proper(Author.Text)	Converts the first letter of each word to uppercase if it's lowercase, and converts any other uppercase letters to lowercase.	"E. E. Cummings"

Single-column table

The examples in this section convert strings from the **Address** column of the **People** data source, which contains this data:

Expand table

Name	Address
"Jean"	"123 Main St NE"
"Fred"	"789 SW 39th #3B"

Each formula returns a single-column table that contains the converted strings.

Expand table

Formula	Description	Result
Lower(ShowColumns(People, "Address"))	Converts any letter that's lowercase to uppercase.	A single-column table with a `Value` column containing the

Formula	Description	Result
		following values: "123 main st ne", "789 sw 39th #3b"
Upper(ShowColumns(People, "Address"))	Converts any letter that's lowercase to uppercase.	A single-column table with a `Value` column containing the following values: "123 MAIN ST NE", "789 SW 39TH #3B"
Proper(ShowColumns(People, "Address"))	Converts any first letter of a word that's lowercase to uppercase, and converts any other letter that's uppercase to lowercase.	A single-column table with a `Value` column containing the following values: "123 Main St Ne", "789 Sw 39th #3b"

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a label, and set its **Text** property to this function:
   Proper(Source.Text)
3. Press F5, and then type **WE ARE THE BEST!** into the **Source** box.
   The label shows **We Are The Best!**

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Rand and RandBetween functions

Article • 06/10/2024

Applies to: Canvas apps Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns a pseudo-random number.

Description

The **Rand** function returns a pseudo-random number that's greater than or equal to 0 and less than 1. For example, **Rand()** might return **0.43147** and could return **0** but not **1**.

The **RandBetween** function returns a pseudo-random integer (whole number with no decimal portion) that is between two numbers, inclusive. For example, **RandBetween(1, 3)** may return **1**, **2**, or **3**.

Volatile Functions

Rand and **RandBetween** are volatile function. Each time the function is evaluated it returns a different value.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Rand()** won't change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it's part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = Slider1.Value + Rand()** then a new random number is generated each time the Slider control's value changes and the label's text property is reevaluated. See below for this example.

When used in a behavior formula, **Rand** and **RandBetween** will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Rand()

RandBetween(*Bottom*, *Top*)

- *Bottom* - Required. The smallest integer that the function can return.
- *Top* - Required. The largest integer that the function can return. Must be equal to or greater than *Bottom*.

Examples

Basic usage

Expand table

Formula	Description	Result
Rand()	Returns a pseudo-random number that's greater than or equal to 0 and less than 1.	Varies with each evaluation, for example **0.874252**.
Rand() * 100	Building on the previous example, uses multiplication to extend the range to greater than or equal to 0 and less than 100.	Varies with each evaluation, for example **78.42521**.
Int(Rand() * 100)	Building on the previous example, uses the **Int** function to remove the decimal portion, resulting in an integer greater than or equal to 0 and less than 100	Varies with each evaluation, for example **84**.
RandBetween(0, 99)	Building on the previous example, performs the same operation using the **RandBetween** function	Varies with each evaluation, for example **21**.
RandBetween(-1, 1)	Returns a pseudo-random number that is between -1 and 1 inclusive: **-1**, **0**, or **1**.	Varies with each evaluation, for example **-1**.

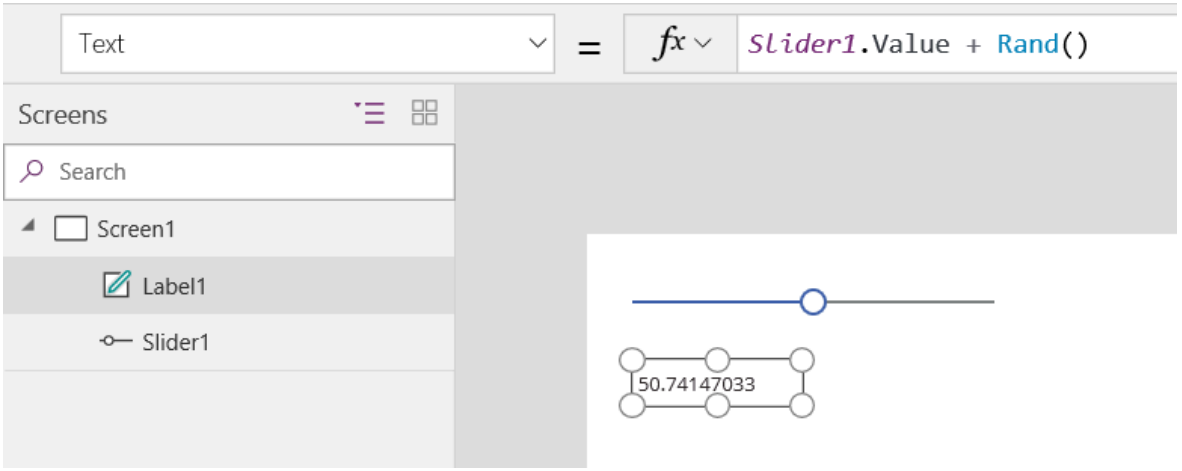
Display a different random number as user input changes with Rand

1. Add a Slider control, and rename it **Slider1** if it has a different name.

2. Add a Label control, and set its **Text** property to this formula:

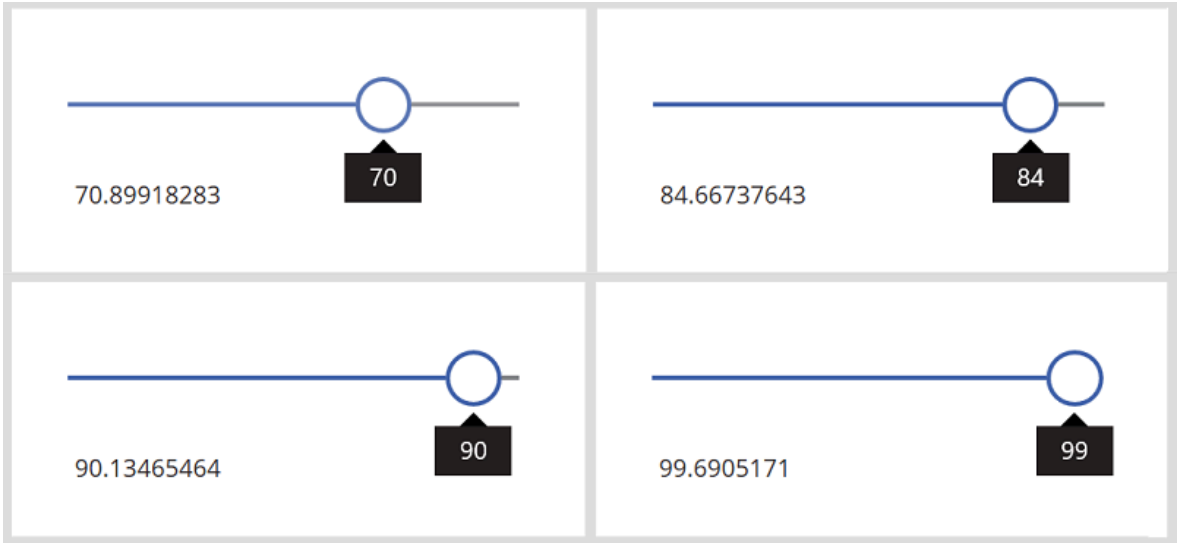
   Slider1.Value + Rand()

   The label shows **50** (the default value for the slider) plus a random decimal:



3. While holding down the Alt key, change the value of the slider.

   Every time you change the value of the slider, the decimal portion of the label shows a different random number:



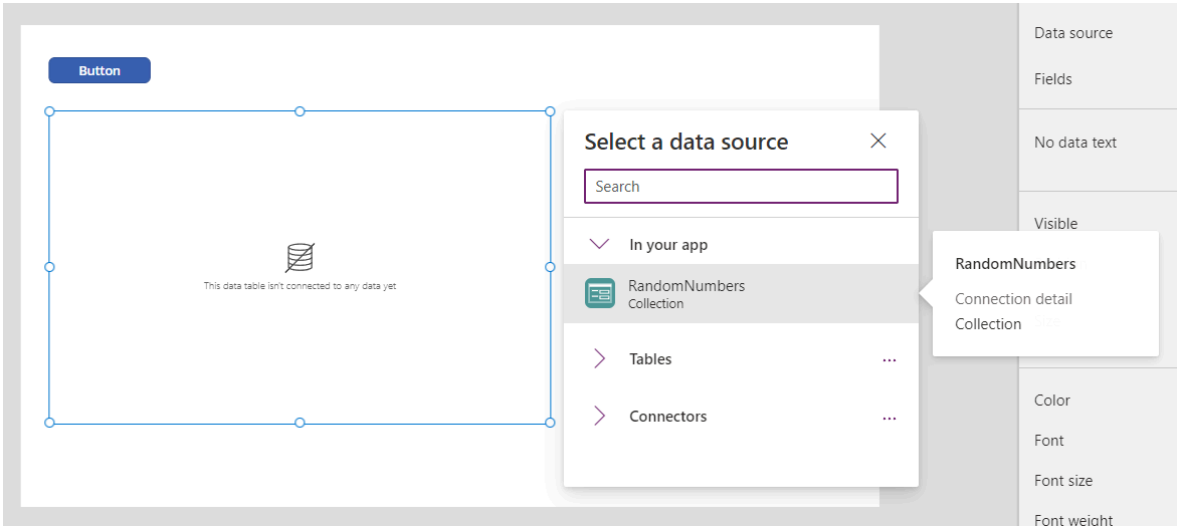
Create a table of random numbers with RandBetween

1. Add a **Button** control, and set its **OnSelect** property to this formula:

   ClearCollect(RandomNumbers, ForAll(Sequence(100), RandBetween(1, 20)))

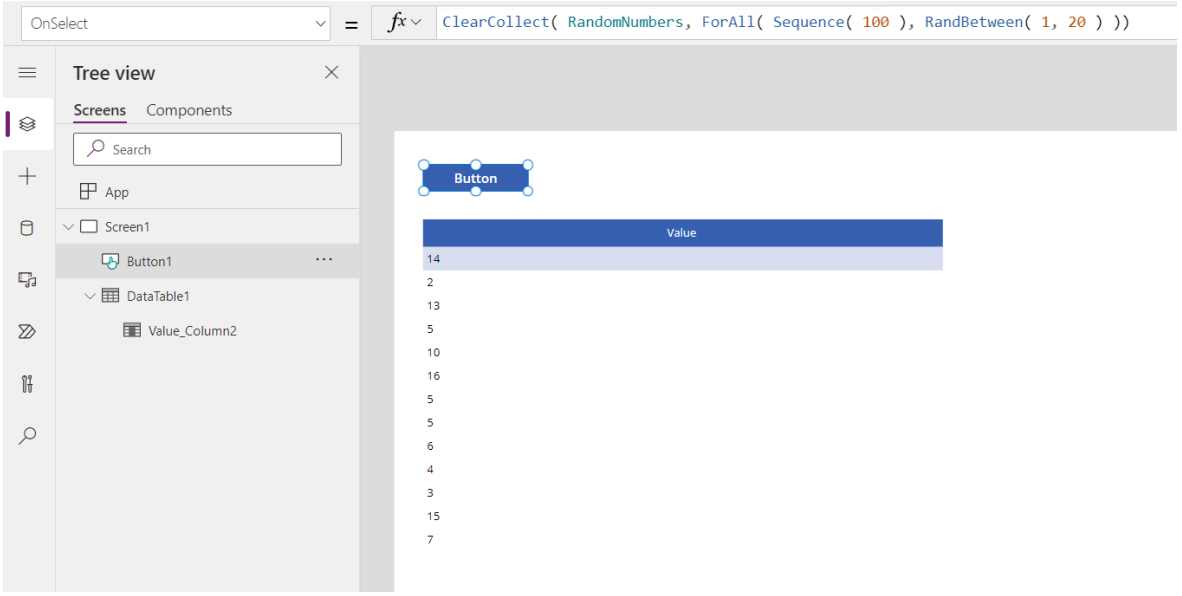
   This formula creates a single-column table that's used to iterate 100 times, resulting in 100 random numbers.

2. Add a **Data table**, set its **Items** property to **RandomNumbers**, and show the **Value** field.

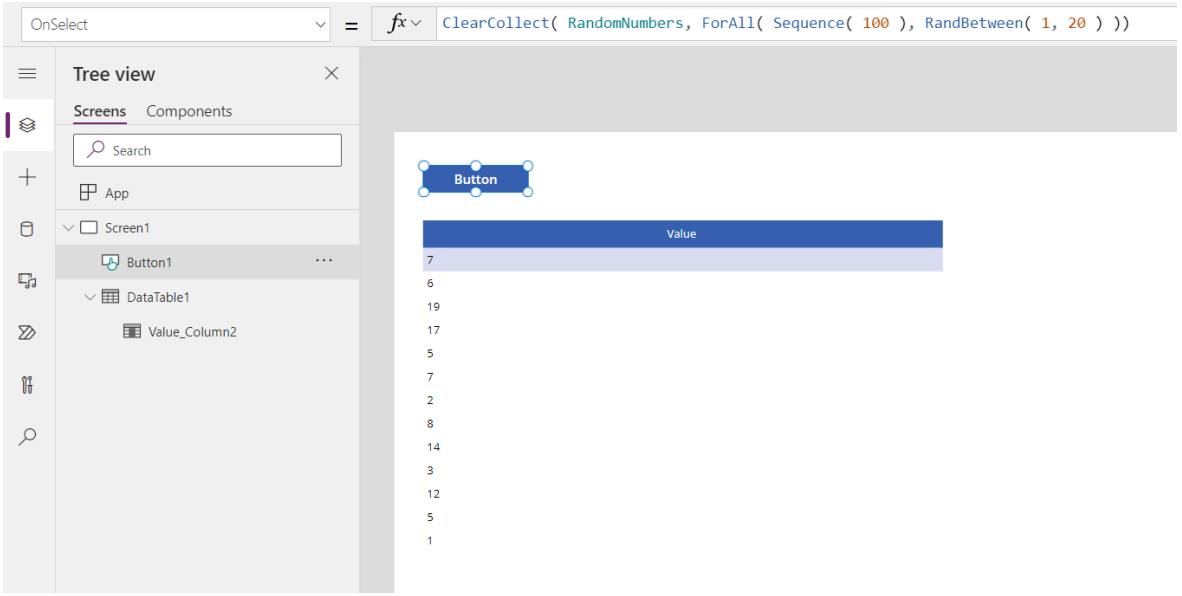


3. While holding down the **Alt** key, select the button by clicking or tapping it.

   The data table shows 100 hundred random numbers between 1 and 20:



4. Select the button again to show a different list of random numbers:



To generate a single random number instead of a table, use **Set(RandomNumber, Rand())** or **Set(RandNumber, RandBetween(1, 20))**.

Feedback

Was this page helpful? Yes No

Provide product feedback

Rand and RandBetween functions

Article • 06/10/2024

Applies to: Canvas apps Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns a pseudo-random number.

Description

The **Rand** function returns a pseudo-random number that's greater than or equal to 0 and less than 1. For example, **Rand()** might return **0.43147** and could return **0** but not **1**.

The **RandBetween** function returns a pseudo-random integer (whole number with no decimal portion) that is between two numbers, inclusive. For example, **RandBetween(1, 3)** may return **1**, **2**, or **3**.

Volatile Functions

Rand and **RandBetween** are volatile function. Each time the function is evaluated it returns a different value.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Rand()** won't change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it's part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = Slider1.Value + Rand()** then a new random number is generated each time the Slider control's value changes and the label's text property is reevaluated. See below for this example.

When used in a behavior formula, **Rand** and **RandBetween** will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Rand()

RandBetween(*Bottom*, *Top*)

- *Bottom* - Required. The smallest integer that the function can return.
- *Top* - Required. The largest integer that the function can return. Must be equal to or greater than *Bottom*.

Examples

Basic usage

Expand table

Formula	Description	Result
Rand()	Returns a pseudo-random number that's greater than or equal to 0 and less than 1.	Varies with each evaluation, for example **0.874252**.
Rand() * 100	Building on the previous example, uses multiplication to extend the range to greater than or equal to 0 and less than 100.	Varies with each evaluation, for example **78.42521**.
Int(Rand() * 100)	Building on the previous example, uses the **Int** function to remove the decimal portion, resulting in an integer greater than or equal to 0 and less than 100	Varies with each evaluation, for example **84**.
RandBetween(0, 99)	Building on the previous example, performs the same operation using the **RandBetween** function	Varies with each evaluation, for example **21**.
RandBetween(-1, 1)	Returns a pseudo-random number that is between -1 and 1 inclusive: **-1**, **0**, or **1**.	Varies with each evaluation, for example **-1**.

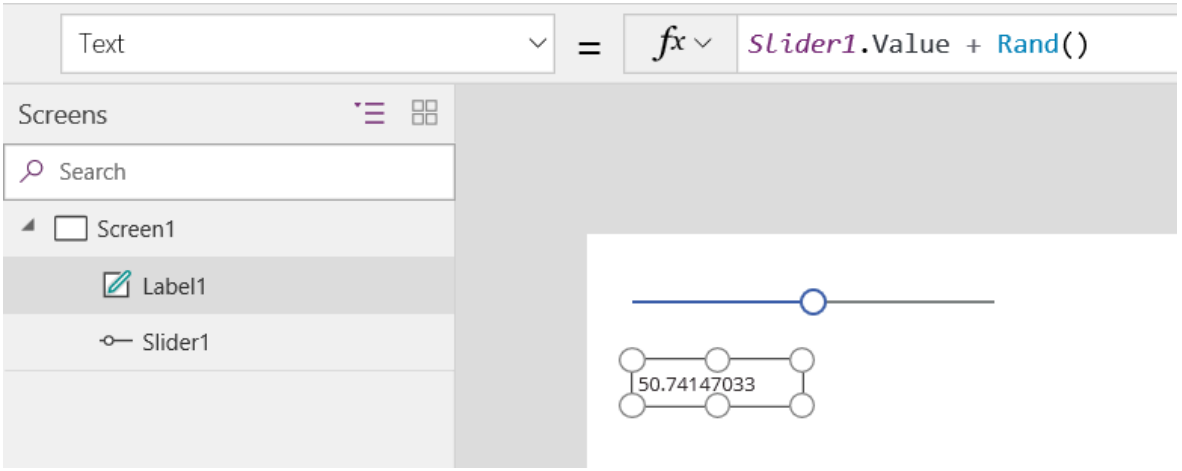
Display a different random number as user input changes with Rand

1. Add a Slider control, and rename it **Slider1** if it has a different name.

2. Add a Label control, and set its **Text** property to this formula:

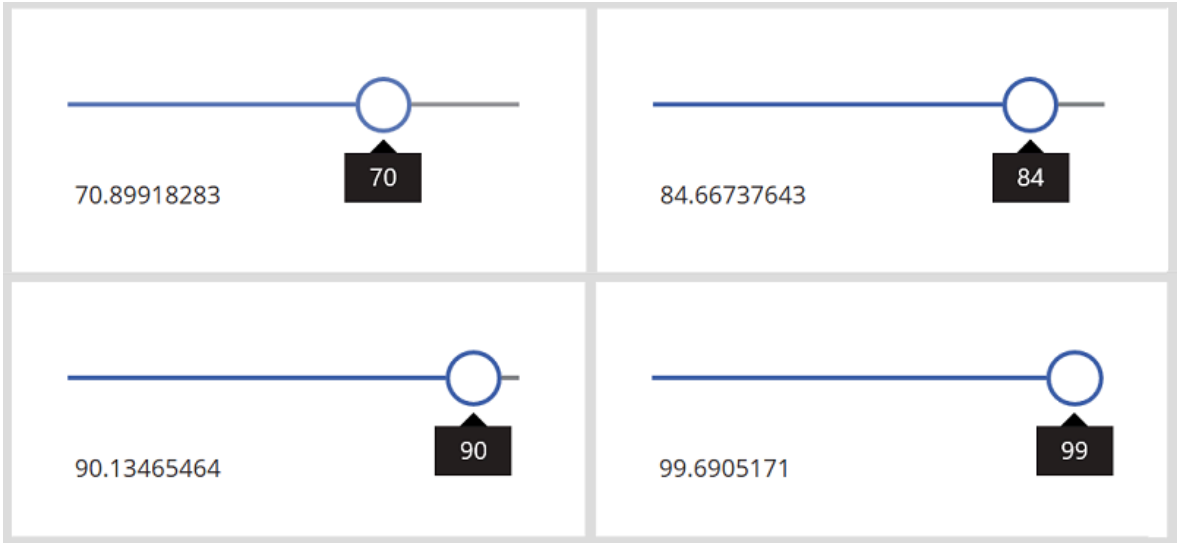
   Slider1.Value + Rand()

   The label shows **50** (the default value for the slider) plus a random decimal:



3. While holding down the Alt key, change the value of the slider.

   Every time you change the value of the slider, the decimal portion of the label shows a different random number:



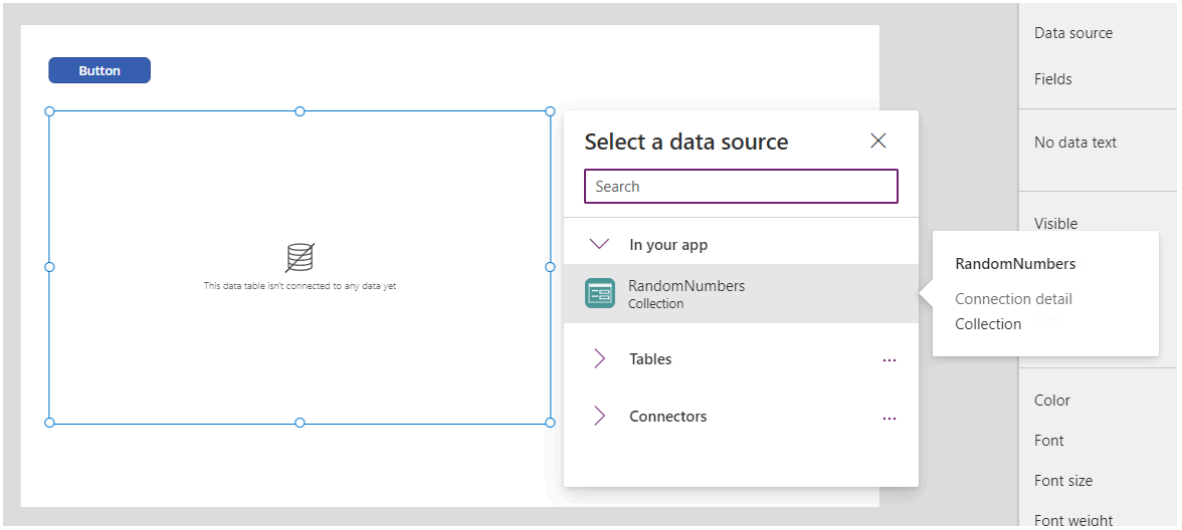
Create a table of random numbers with RandBetween

1. Add a **Button** control, and set its **OnSelect** property to this formula:

   ClearCollect(RandomNumbers, ForAll(Sequence(100), RandBetween(1, 20)))

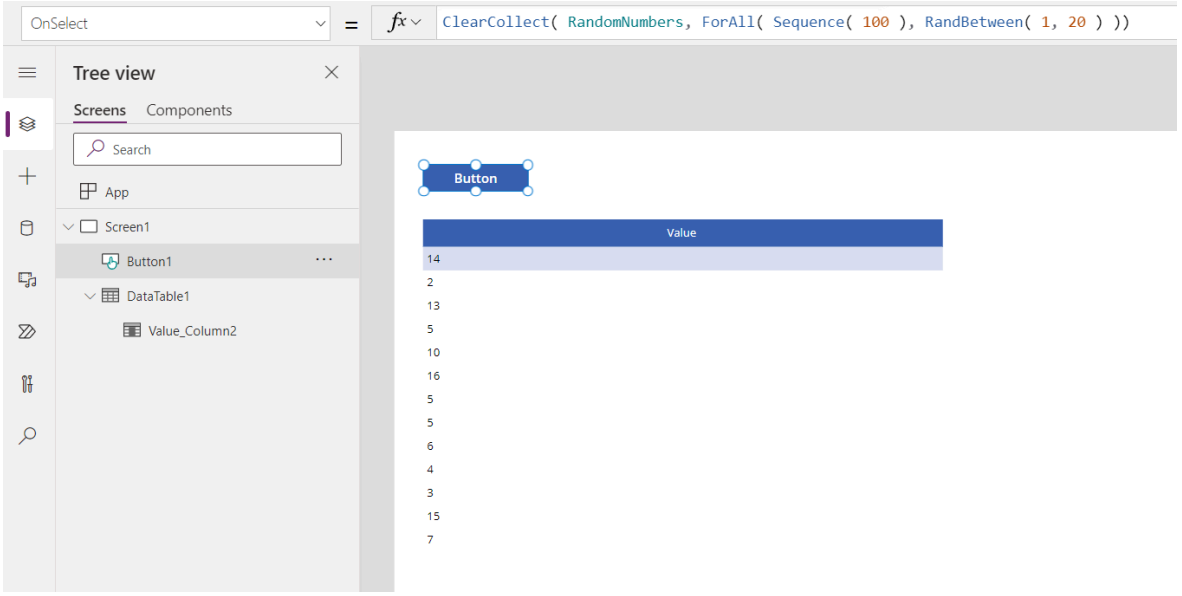
   This formula creates a single-column table that's used to iterate 100 times, resulting in 100 random numbers.

2. Add a **Data table**, set its **Items** property to **RandomNumbers**, and show the **Value** field.

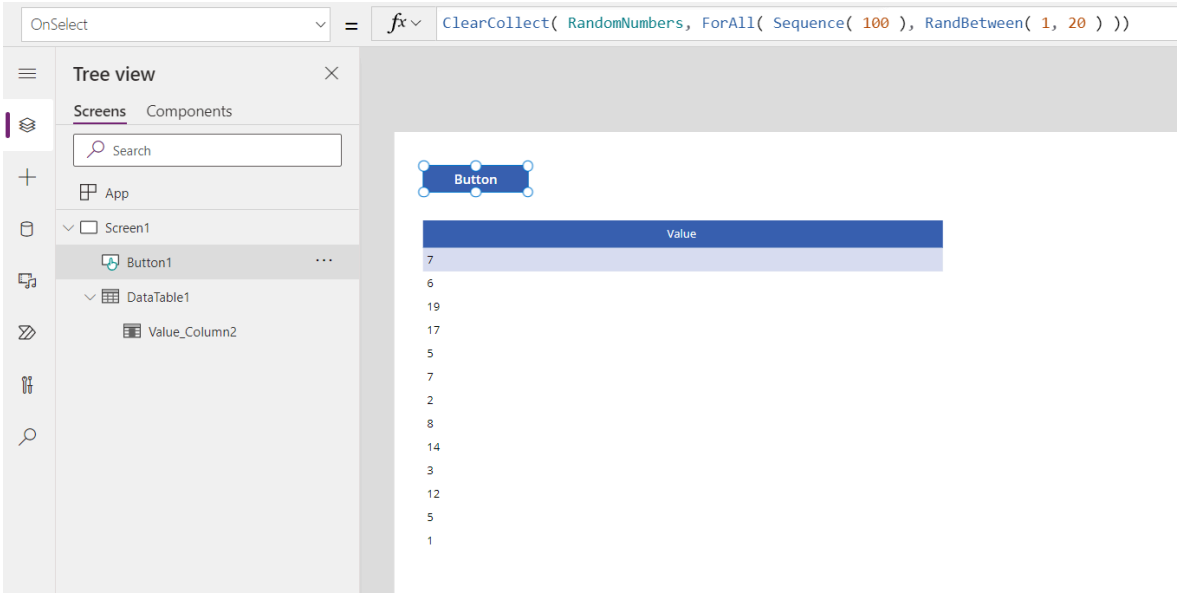


3. While holding down the **Alt** key, select the button by clicking or tapping it.

   The data table shows 100 hundred random numbers between 1 and 20:



4. Select the button again to show a different list of random numbers:



To generate a single random number instead of a table, use **Set(RandomNumber, Rand())** or **Set(RandNumber, RandBetween(1, 20))**.

Feedback

Was this page helpful? Yes No

Provide product feedback

ReadNFC function

Article • 03/22/2024

Applies to: Canvas apps

Reads a Near Field Communication (NFC) tag.

Description

Use the **ReadNFC** function to read an NFC tag that is close to your device. When invoked, the screen displays instructions for scanning an NFC tag, and only returns after the tag has been scanned or it times out.

Expand table

Column	Type	Description
Identifier	Text	The NFC tags identifier if available.
NDEFRecords	Table	The supported NDEF records found on the tag.

A single **NDEFRecord** contains the following columns:

Expand table

Column	Type	Description
RTD	Text	The tag's Record Type Definition (RTD). Only *Text* and *URI* are supported at this time.
TNF	Number	The tag's Type Name Format (TNF). Only TNFs of *Well Known(1)* are supported at this time.
Text	Text	The text payload of the NFC tag if RTD is *TEXT*, *blank* otherwise.
URI	Hyperlink	The URI payload of the NFC tag if RTD is *URI*, *blank* otherwise.

If the NDEF record isn't supported (for example, the TNF isn't of type *Well Known*), then it won't be returned as part of the **NDEFRecords** table.

Always check the payload values for *blank* using the **IsBlank** function before using it. You don't need to check the **RTD** and **TNF** values yourself as they must be the correct values for **Text** and **URI** to have a non-*blank* value.

Additional **RTD** and **TNF** values may be supported in the future. If more values are supported, additional payload columns will also be added. The raw **RTD** and **TNF** values are provided for informational purposes and don't need to be consulted if the payload column is checked for *blank*. More information about these values and their use is available through the NFC Forum.

ReadNFC doesn't require a tag containing NDEF records to be used, but you may still get the tag identifier if one is available.

ReadNFC can only be used in behavior formulas.

> **Note**
>
> - **ReadNFC** is only supported when running the app on a native mobile app, such as the iOS and Android apps. Even with a supported player, a device may not support NFC. If your application has **Formula-level error management** turned on, the function will return an error. Otherwise, an error message will be shown to the user and the function will return a *blank* record.
> - Ensure your device has the NFC setting enabled to use this capability.

Syntax

ReadNFC()

Examples

Expand table

Formula	Description	Result
`ReadNFC().Identifier`	Returns the identifier of the NFC tag if one exists.	`04A1C301314003`
`ReadNFC().NDEFRecords`	Returns a table of NDEF records found on the NFC tag if they're a supported type and payload.	<table><tr><th>RTD</th><th>TNF</th><th>Text</th><th>URI</th></tr><tr><td>T</td><td>1</td><td>Blue</td><td></td></tr><tr><td>U</td><td>1</td><td></td><td>microsoft.com</td></tr></table>

Step-by-step example

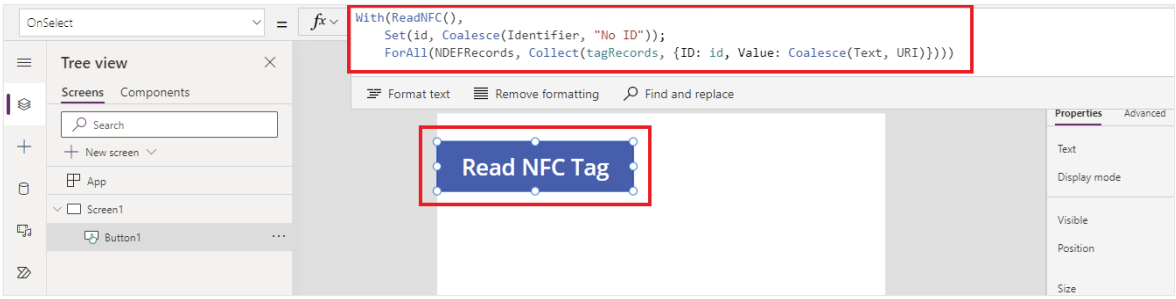
1. Create a blank canvas app with **Phone** format.

2. Add a **Button** control.

3. Double-click the button control to change the button text to **Read NFC Tag** (or modify the **Text** property).

4. Set the **OnSelect** property of the button control to this formula that will add an item to our collection:

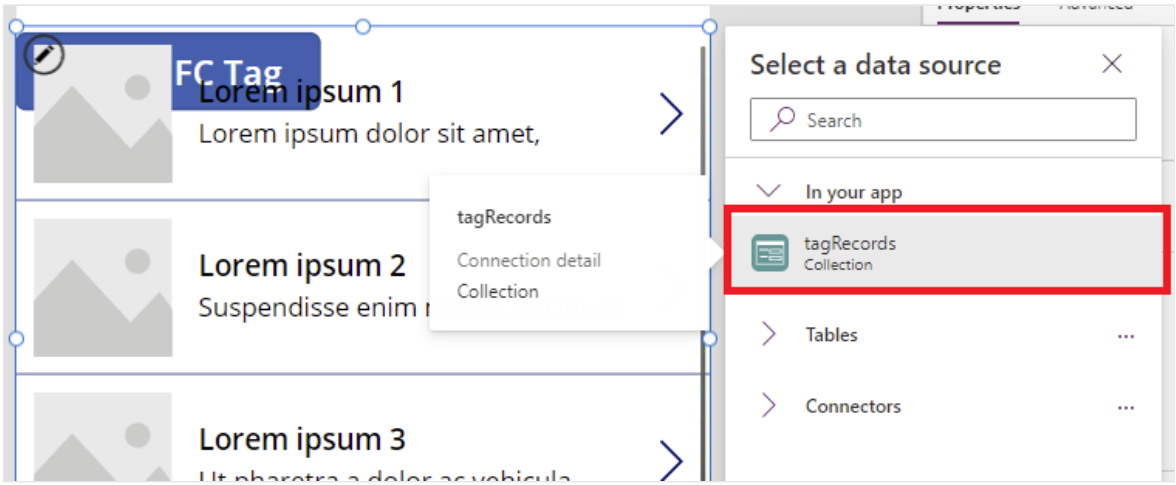
```
Power Apps

With(ReadNFC(),
    Set(id, Coalesce(Identifier, "No ID"));
    ForAll(NDEFRecords, Collect(tagRecords, {ID: id, Value:
Coalesce(Text, URI)})))
```



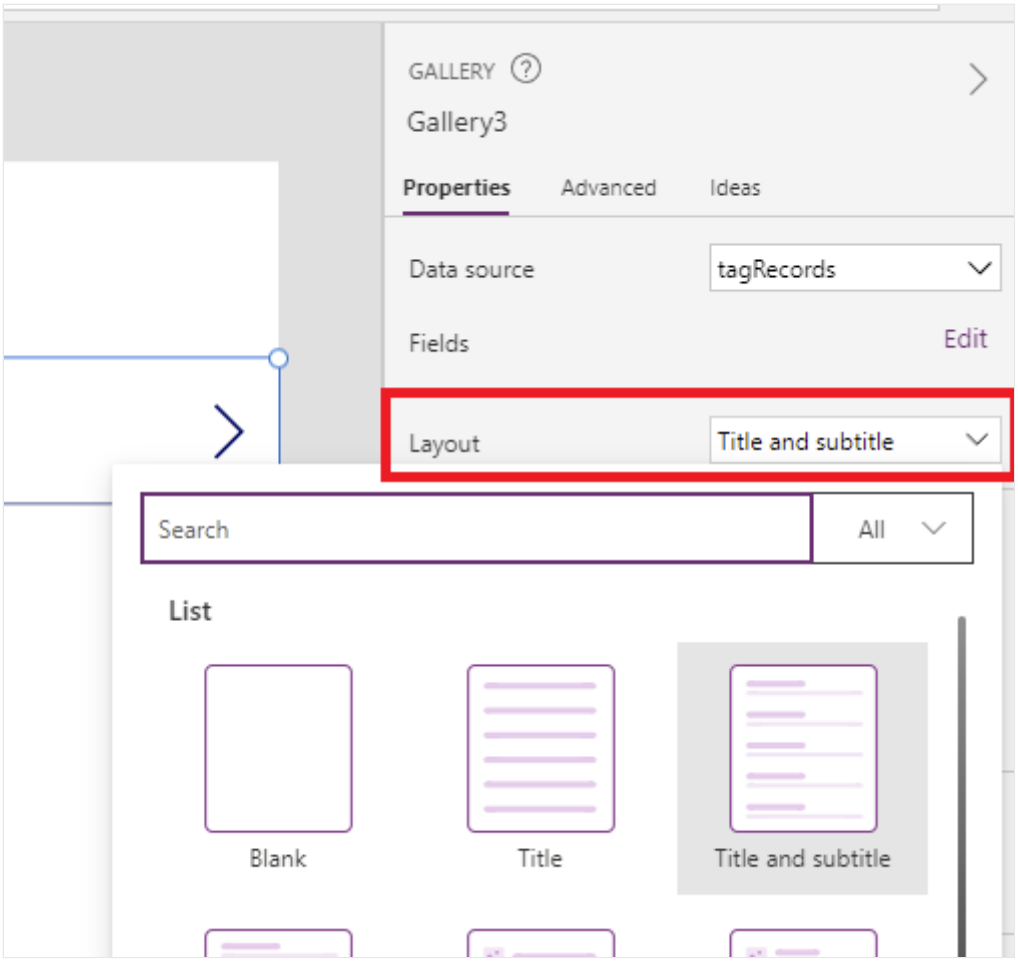
This formula reads an NFC tag using the `ReadNFC()` function, and displays type information about the result. Then it collects the read NFC tags to populate the `tagRecords` collection to be used for the gallery in the next steps.

5. Add a **Gallery** control with a vertical layout.

6. When prompted, select the **tagRecords** collection as the data source for this gallery. This action will set the **Items** property of the **Gallery** control.



7. Reposition the gallery control so that it doesn't overlap the button, and change the layout type to **Title and subtitle**.



8. Change the **Title** formula in the gallery control to **ThisItem.Value** and change the **Subtitle** formula in the gallery control to **ThisItem.ID**.

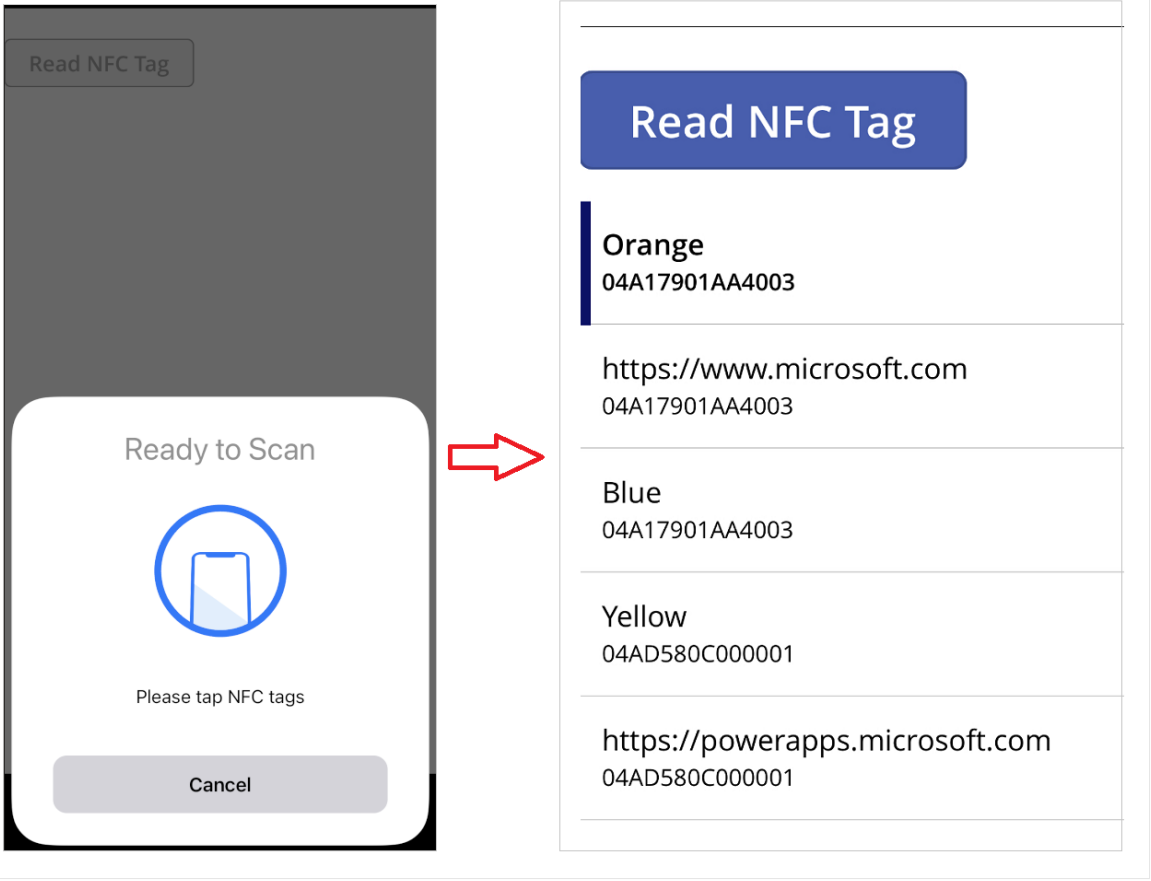
   You can also delete the **NextArrow** from the gallery control since it's not used.

9. Save and publish the app.

10. Since the function `ReadNFC()` can't be used in Power Apps Studio or in a web browser, open your app on a mobile device.

    > **Tip**
    >
    > The app might take a few moments to appear on your mobile device. If you don't see the app listed, try refreshing your app list.

11. Select **Read NFC Tag** and scan a tag. Repeat the process to add multiple tags to your collection.


Read NFC Tag
Ready to Scan
Please tap NFC tags
Cancel
Read NFC Tag
Orange
04A17901AA4003
https://www.microsoft.com
04A17901AA4003
Blue
04A17901AA4003
Yellow
04AD580C000001
https://powerapps.microsoft.com
04AD580C000001

RecordInfo function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Provides information about a record of a data source.

Use **RecordInfo** to obtain information about a particular record of a data source. The data source must be tabular and compatible with the **Remove** and **Patch** functions.

At this time, only Microsoft Dataverse is supported. Records from all other data sources will result in a formula error.

The information available:

Expand table

Information argument	Description
RecordInfo.DeletePermission	Does the current user have permission to remove this record from the data source?
RecordInfo.EditPermission	Does the current user have permission to modify this record in the data source?
RecordInfo.ReadPermission	Does the current user have permission to read this record from the data source?

RecordInfo returns a Boolean value:

Expand table

Return value	Description
true	The user has the permission.
false	The user does not have permission. If the record is *blank* then **RecordInfo** will also return *false*.

RecordInfo takes into account permissions at the data source level too. For example, if the user has permission at the record level to modify a record, but the user does not have permissions at the table level, then it will return *false* for **ModifyPermission**. Use the **DataSourceInfo** function to obtain information about the data source as a whole.

Syntax

RecordInfo(*Record*, *Information*)

- *Record* – Required. The record to test.
- *Information* – Required. The desired information for the record.

Examples

Power Apps

```
RecordInfo( First(Accounts), RecordInfo.EditPermission )
```

Checks the edit permission for the first record in the `Accounts` data source, which could be in Dataverse, SharePoint, SQL Server, or another tabular data source. If the user has permission to edit this record and modify the `Accounts` data source in general, then **RecordInfo** will return *true*.

Power Apps

```
With( { MyRecord: First( Accounts ) },
      RecordInfo( MyRecord, RecordInfo.EditPermission ) )
```

Captures a record using the **With** function and then passes this value to the `RecordInfo` function. The result will be the same as the last example.

Power Apps

```
Collect( MyAccounts, FirstN( Accounts, 10 ) );
RecordInfo( First( MyAccounts ), RecordInfo.EditPermission ) )
```

Captures the first 10 records from the `Accounts` data source into the `MyAccounts` collection. Since the records originated from a data source, they can be used with the **RecordInfo** function. The result will be the same as the last example.

Power Apps

```
Collect( MyCollection, [ 1, 2, 3 ] );
RecordInfo( First(MyCollection), RecordInfo.DeletePermission )
```

Creates the `MyCollection` collection and tests the first record to determine if it can be removed. Since the record's origin is a collection and not a data source, **RecordInfo** will

return an error.

Refresh function

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Refreshes the records of a data source.

Description

The **Refresh** function retrieves a fresh copy of a data source. You'll see changes that other users made.

Refresh has no return value, and you can use it only in behavior formulas.

Syntax

Refresh(*DataSource*)

- *DataSource* – Required. The data source that you want to refresh.

Example

In this example, you'll refresh the data source named **IceCream**, which starts with this data:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	300

A user on another device changes the **Quantity** in the **Strawberry** record to **400**. You won't see this change until this formula executes:

Refresh(IceCream)

After that formula executes, galleries that are bound to the **IceCream** data source will show the updated value for **Strawberry**:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	400

Relate and Unrelate functions

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Relate and unrelate records of two tables through a one-to-many or many-to-many relationship.

Description

The **Relate** function links two records through a one-to-many or many-to-many relationship in Microsoft Dataverse. The **Unrelate** function reverses the process and removes the link.

For one-to-many relationships, the Many table has a foreign-key field that points to a record of the One table. **Relate** sets this field to point to a specific record of the One table, while **Unrelate** sets this field to *blank*. If the field is already set when **Relate** is called, the existing link is lost in favor of the new link. You can also set this field by using the **Patch** function or an **Edit form** control; you need not use the **Relate** function.

For many-to-many relationships, the system that links the records maintains a hidden join table. You can't access this join table directly; it can be read only through a one-to-many projection and set through the **Relate** and **Unrelate** functions. Neither related table has a foreign key.

The data for the table that you specify in the first argument will be refreshed to reflect the change, but the data for the table that you specify in the second argument won't. That data must be manually refreshed with the **Refresh** function to show the result of the operation.

These functions never create or delete a record. They only relate or unrelate two records that already exist.

You can use these functions only in behavior formulas.

> **Note**
>
> These functions are part of a preview feature, and their behavior is available only when the **Relational data, option sets, and other new features for CDS** feature is enabled. This is an app-level setting that's enabled by default for new apps. To find this feature switch, select **Settings**, and then select **Upcoming features**. Your

feedback is very valuable to us - please let us know what you think in the Power Apps community forums.

Syntax

Relate(*Table1RelatedTable*, *Table2Record*)

- *Table1RelatedTable* - Required. For a record of *Table1*, the table of *Table2* records related through a one-to-many or many-to-many relationship.
- *Table2Record* - Required. The *Table2* record to add to the relationship.

Unrelate(*Table1RelatedTable*, *Table2Record*)

- *Table1RelatedTable* - Required. For a record of *Table1*, the table of *Table2* records related through a one-to-many or many-to-many relationship.
- *Table2Record* - Required. The *Table2* record to remove from the relationship.

Examples

Consider a **Products** table with the following relationships as seen in the Power Apps portal's table viewer:

Expand table

Relationship display name	Related table	Relationship type
Product Reservation	Reservation	One-to-many
Product ↔ Contact	Contact	Many-to-many

Products and **Reservations** are related through a One-to-Many relationship. To relate the first record of the **Reservations** table with the first record of the **Products** table:

```
Relate( First( Products ).Reservations, First( Reservations ) )
```

To remove the relationship between these records:

```
Unrelate( First( Products ).Reservations, First( Reservations ) )
```

At no time did we create or remove or a record, only the relationship between records was modified.

Products and **Contacts** are related through a Many-to-Many relationship. To relate the first record of the **Contacts** table with the first record of the **Products** table:

```
Relate( First( Products ).Contacts, First( Contacts ) )
```

As Many-to-Many relationships are symmetric, we could also have done this in the opposite direction:

```
Relate( First( Contacts ).Products, First( Products ) )
```

To remove the relationship between these records:

```
Unrelate( First( Products ).Contacts, First( Contacts ) )
```

or:

```
Unrelate( First( Contacts ).Products, First( Products ) )
```

The walk through that follows does exactly these operations on these tables using an app with **Gallery** and **Combo box** controls for selecting the records involved.

These examples depend on the sample data being installed in your environment. Either create a trial environment including sample data or add sample data to an existing environment.

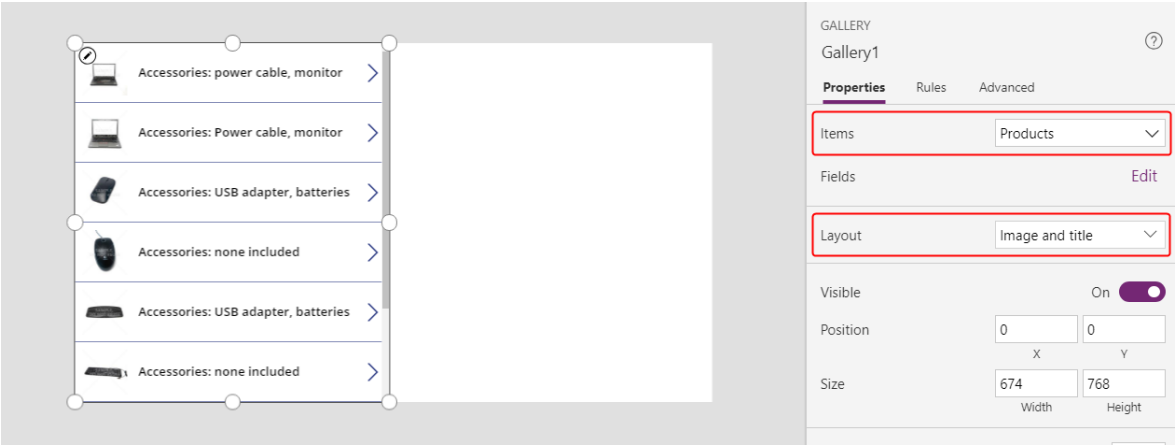
One-to-many

Relate function

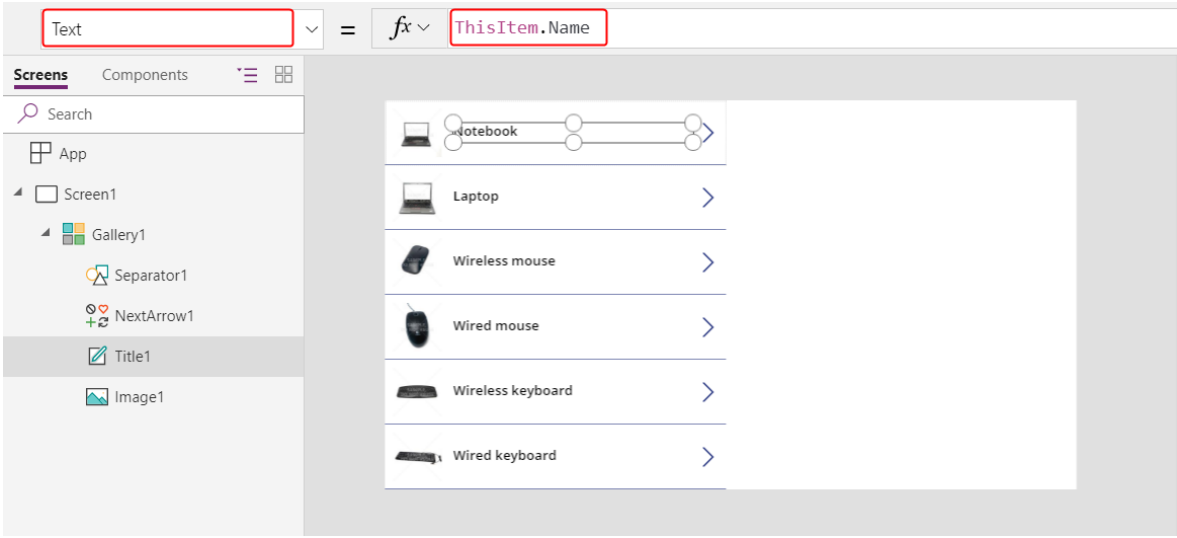
You'll first create a simple app to view and reassign the reservations that are associated with a product.

1. Create a tablet app from blank.
2. On the **View** tab, select **Data sources**.
3. In the **Data** pane, select **Add data** > select **Products**.
   The Products table is part of the sample data loaded above.
4. On the **Insert** tab, add a blank vertical **Gallery** control.
5. Ensure that the control that you just added is named **Gallery1**, and then move and resize it to fill the left-hand side of the screen.

6. On the **Properties** tab, set **Gallery1**'s **Items** property to **Products** and its **Layout** to **Image and title**.



7. In **Gallery1**, ensure that the **Label** control is named **Title1**, and then set its **Text** property to **ThisItem.Name**.



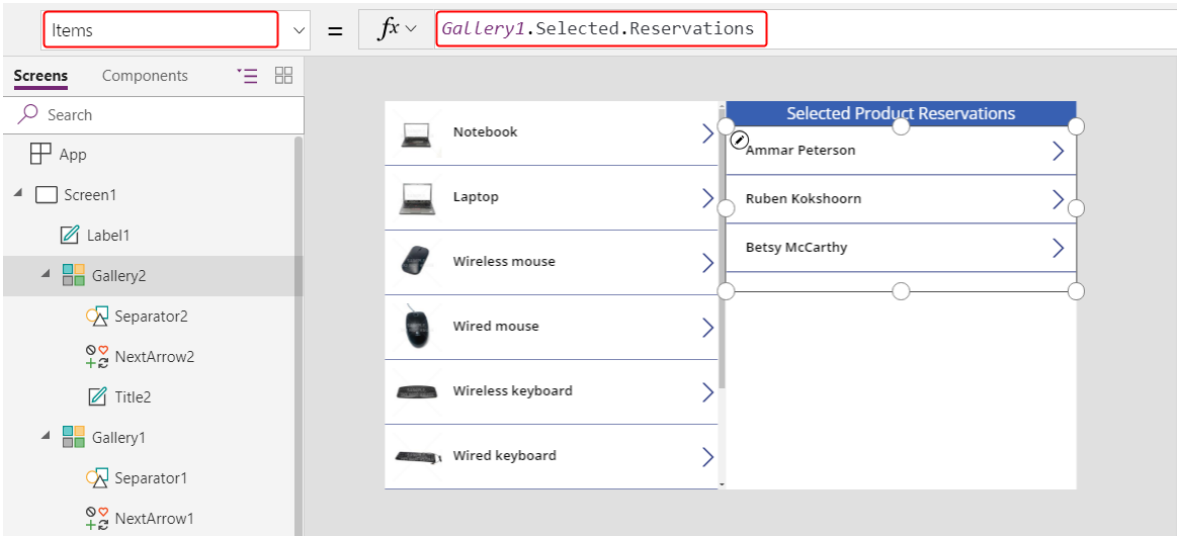
8. Select the screen to avoid inserting the next item into **Gallery1**. Add a second blank vertical **Gallery** control, and ensure that it's named **Gallery2**.

   Gallery2 will show the reservations for whatever product the user selects in **Gallery1**.

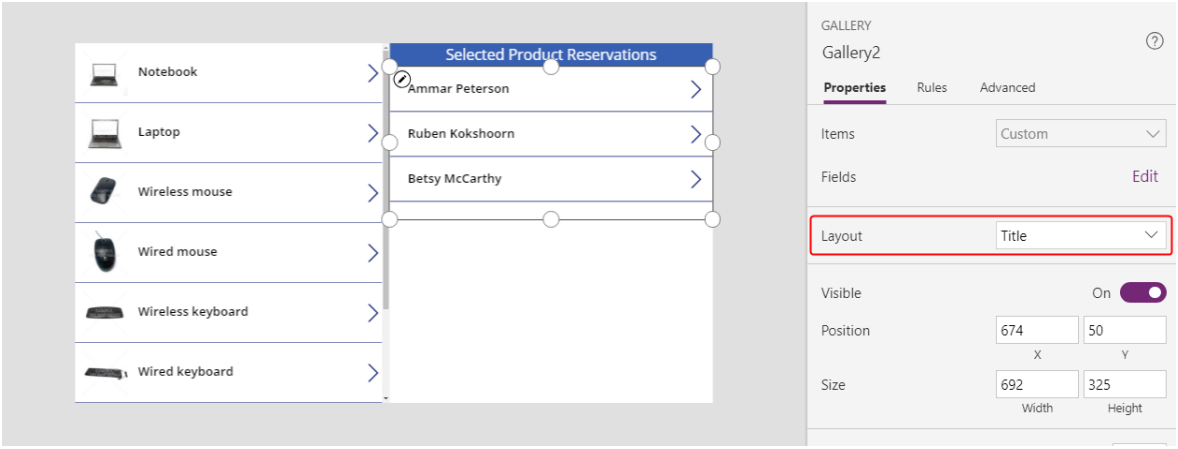
9. Move and resize **Gallery2** to fill the upper-right quadrant of the screen.

10. (optional) Add the blue **Label** control above **Gallery2**, as the next graphic shows.

11. In the formula bar, set the **Items** property of **Gallery2** to **Gallery1.Selected.Reservations**.



12. In the properties pane, set **Gallery2**'s **Layout** to **Title**.

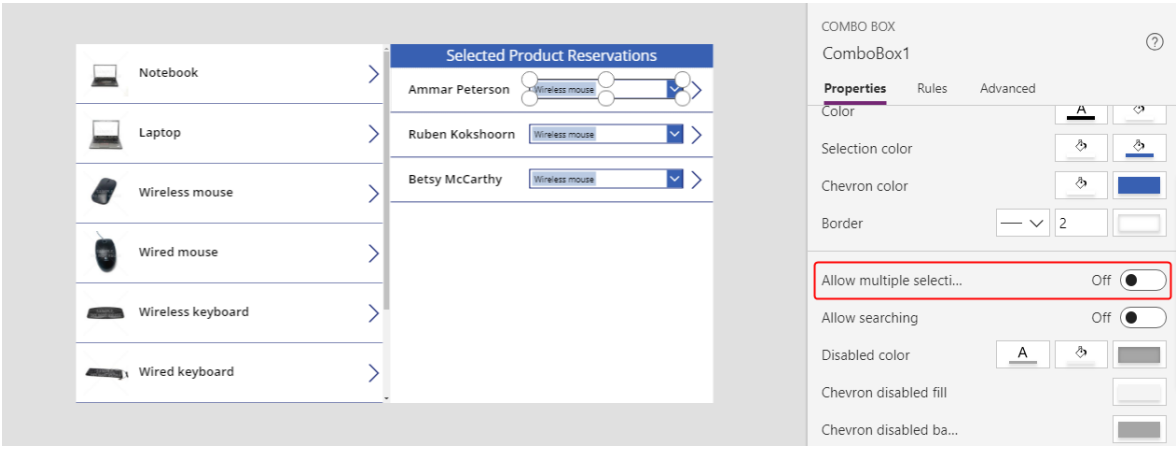


13. In **Gallery2**, add a Combo box control, ensure that it's named **ComboBox1**, and then move and resize it to avoid blocking the other controls in **Gallery2**.

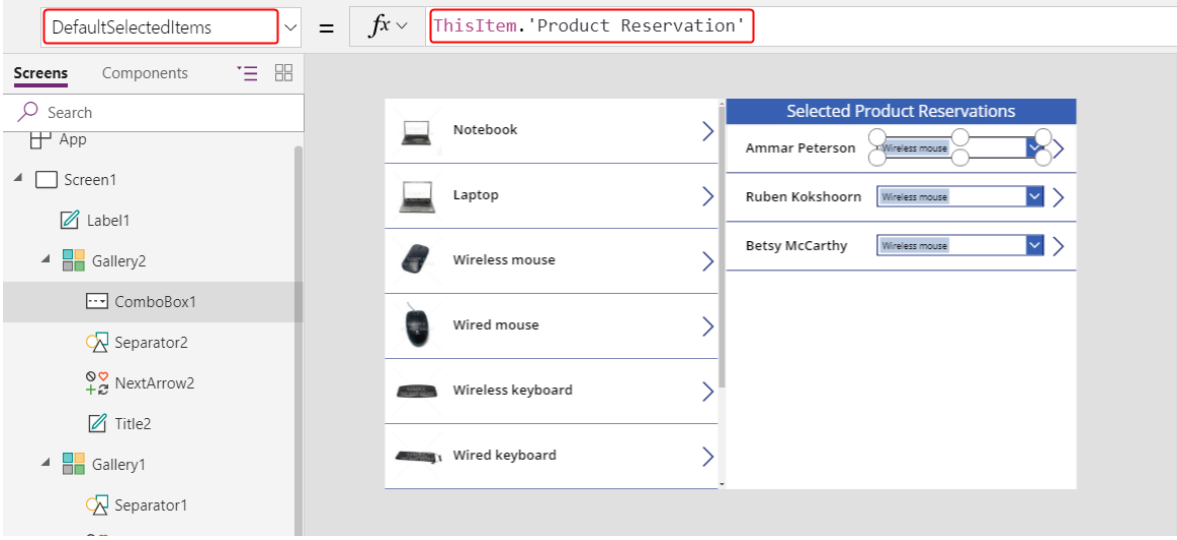
14. On the **Properties** tab, set **ComboBox1**'s **Items** property to **Products**.



15. Scroll down in the **Properties** tab and set **ComboBox1**'s **Allow multiple selection** property to **Off**.



16. In the formula bar, set **ComboBox1**'s **DefaultSelectedItems** property to **ThisItem.'Product Reservation'**.

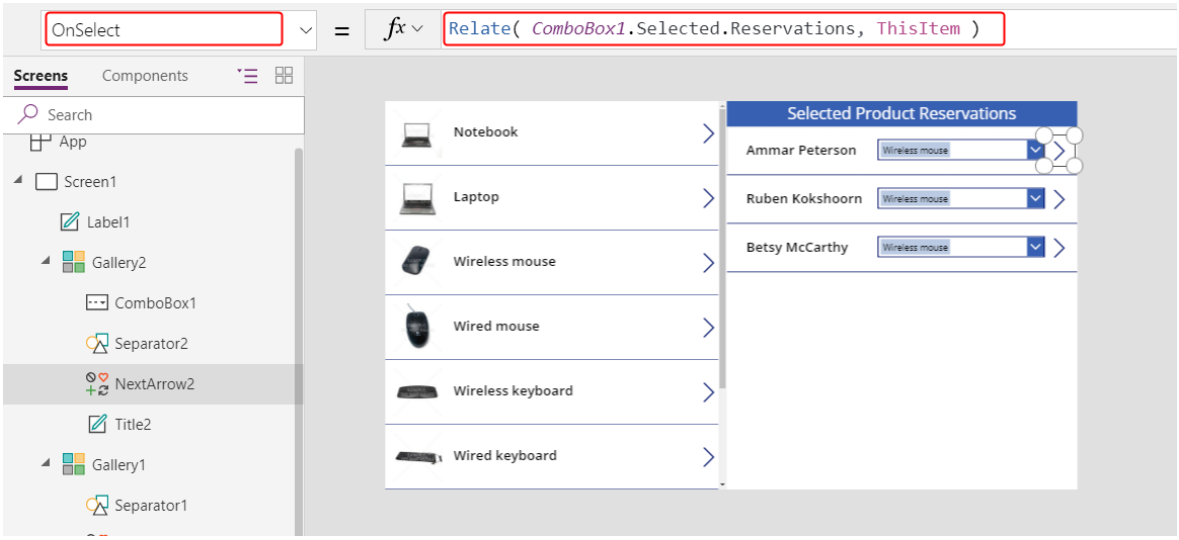


17. In **Gallery2**, set **NextArrow2**'s **OnSelect** property to this formula:

Power Apps

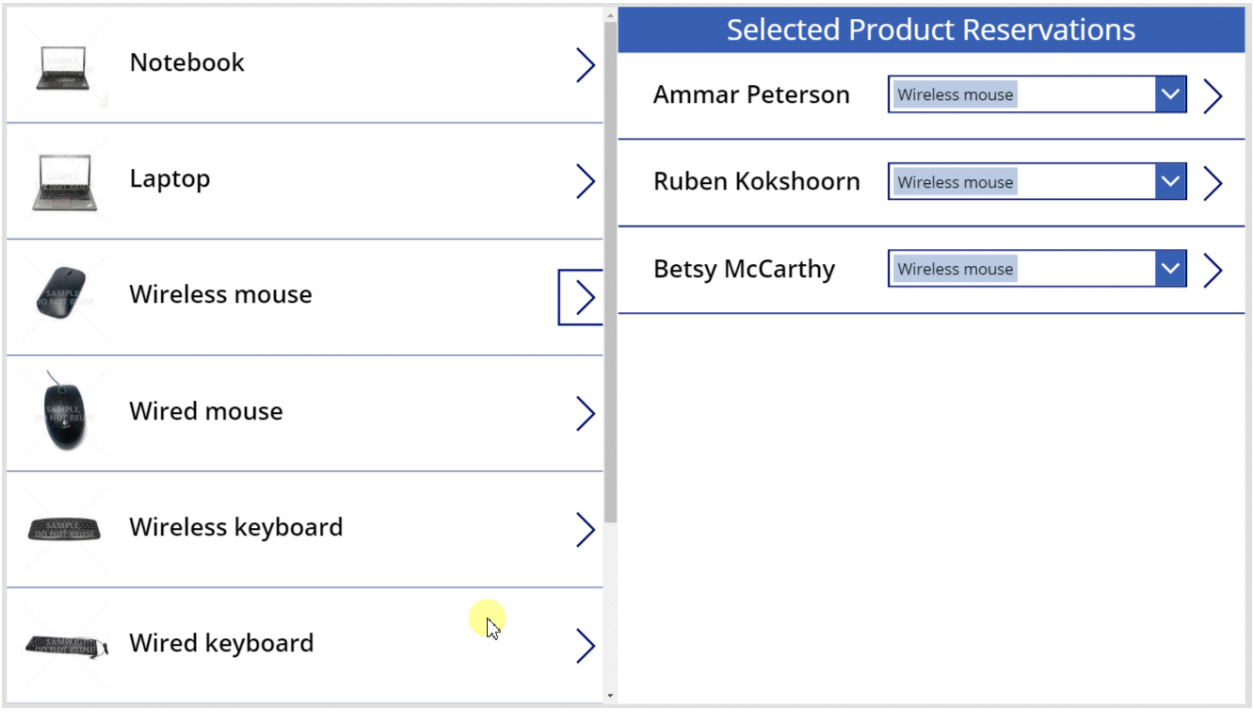
```
Relate( ComboBox1.Selected.Reservations, ThisItem )
```

When the user selects this icon, the current reservation changes to the product that the user selected in **ComboBox1**.



18. Press F5 to test the app in Preview mode.

With this app, the user can move a reservation from one product to another. For a reservation on one product, the user can select a different product in **ComboBox1** and then select **NextArrow2** to change that reservation.



Unrelate function

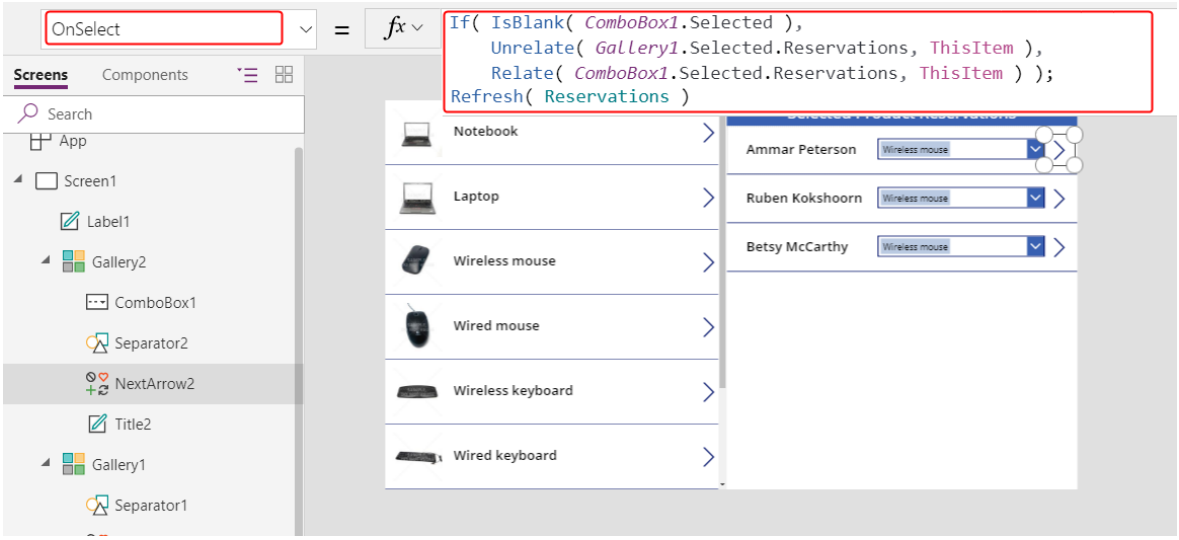
At this point, you can move the relationship from one record to another, but you can't remove the relationship altogether. You can use the **Unrelate** function to disconnect a reservation record from any product.

1. On the **View** tab, select **Data sources**.

2. In the **Data** pane, select **Add data source** > **Microsoft Dataverse** > **Reservations** > **Connect**.

3. In **Gallery2**, set the **OnSelect** formula for **NextArrow2** to this formula:

```
Power Apps

If( IsBlank( ComboBox1.Selected ),
    Unrelate( Gallery1.Selected.Reservations, ThisItem ),
    Relate( ComboBox1.Selected.Reservations, ThisItem )
);
Refresh( Reservations )
```



4. Copy **Gallery2** to the Clipboard by selecting it and then pressing Ctrl-C.

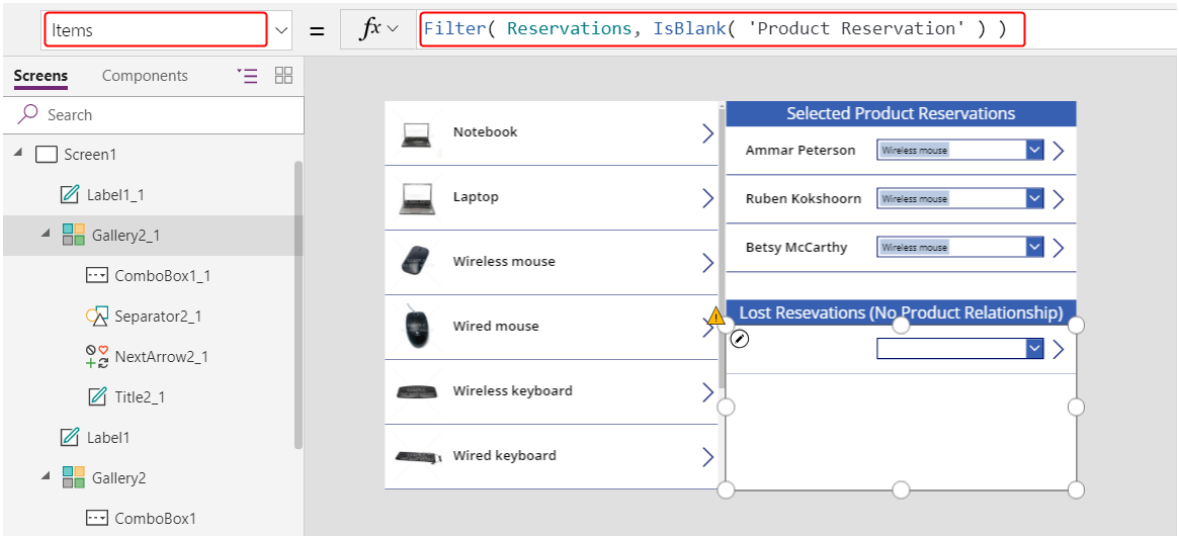
5. Paste a duplicate of **Gallery2** to the same screen by pressing Ctrl-V, and then move it to the lower-right quadrant of the screen.

6. (optional) If you added a label above **Gallery2**, repeat the previous two steps for that label.

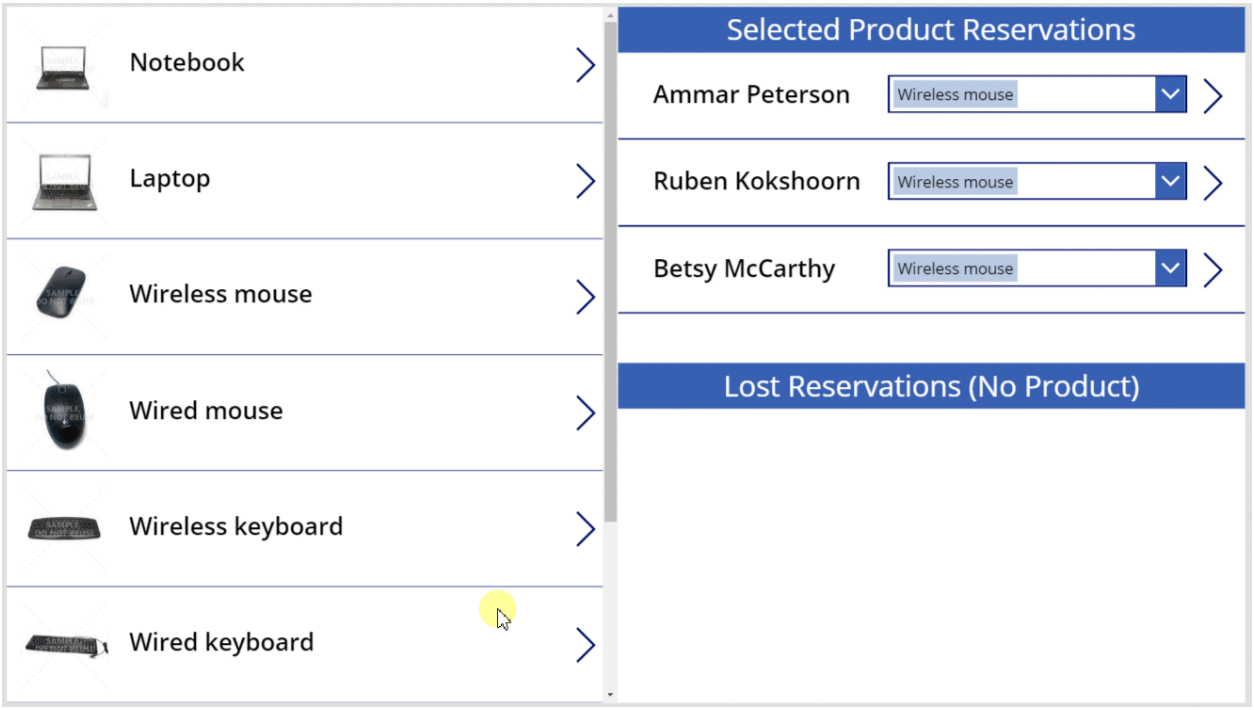
7. Ensure that the duplicate of **Gallery2** is named **Gallery2_1**, and then set its **Items** property to this formula:

```
Power Apps

Filter( Reservations, IsBlank( 'Product Reservation' ) )
```

   A delegation warning appears, but it won't matter with the small amount of data in this example.



With these changes, users can clear the selection in **ComboBox1** for a contact if that person hasn't reserved a product. Contacts who haven't reserved a product appear in **Gallery2_1** where users can assign each contact to a product.

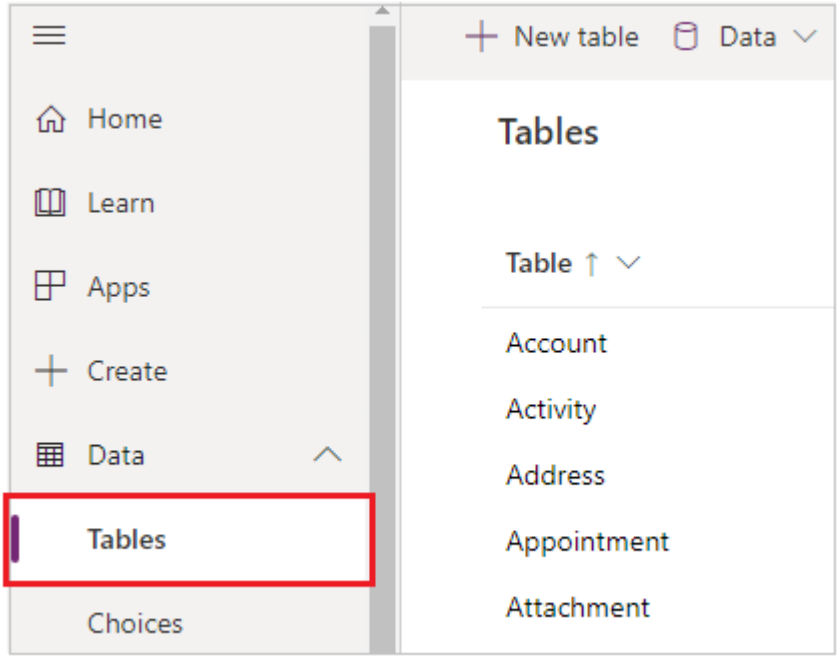


Many-to-many

Create a many-to-many relationship

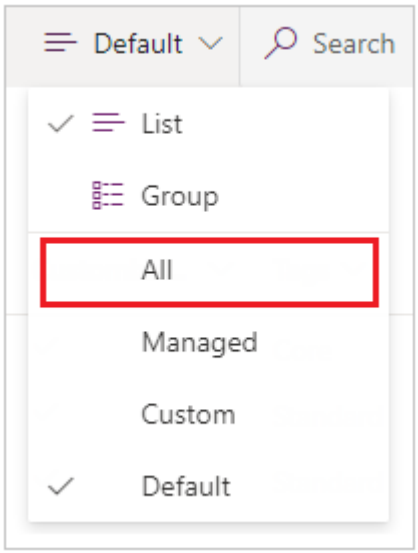
The sample data doesn't include a many-to-many relationship, but you'll create one between the Products table and the Contacts table. Users can relate each product to more than one contact and each contact to more than one product.

1. From this page, select **Data** in the left navigation bar, and then select **Tables**.



2. Change the table filter to include all tables.

   By default, sample tables don't appear.



3. Scroll down, open the **Product** table, and select **Relationships**.

4. Select **Add relationship** > **Many-to-many**.

5. Select the **Contact** table for the relationship.

6. Select **Done** > **Save table**.

Relate and unrelate contacts with one or more products

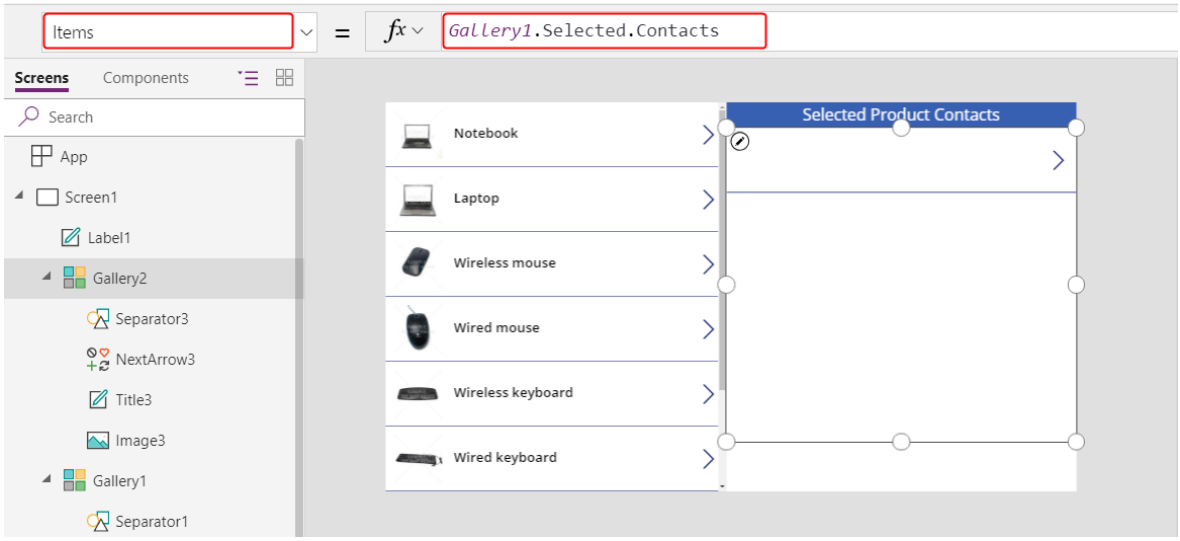
You'll create another app that resembles the one you created earlier in this topic, but the new app will offer a many-to-many relationship. Each contact will be able to reserve multiple products instead of only one.

1. In a blank app for tablets, create **Gallery1** as the first procedure in this topic describes.

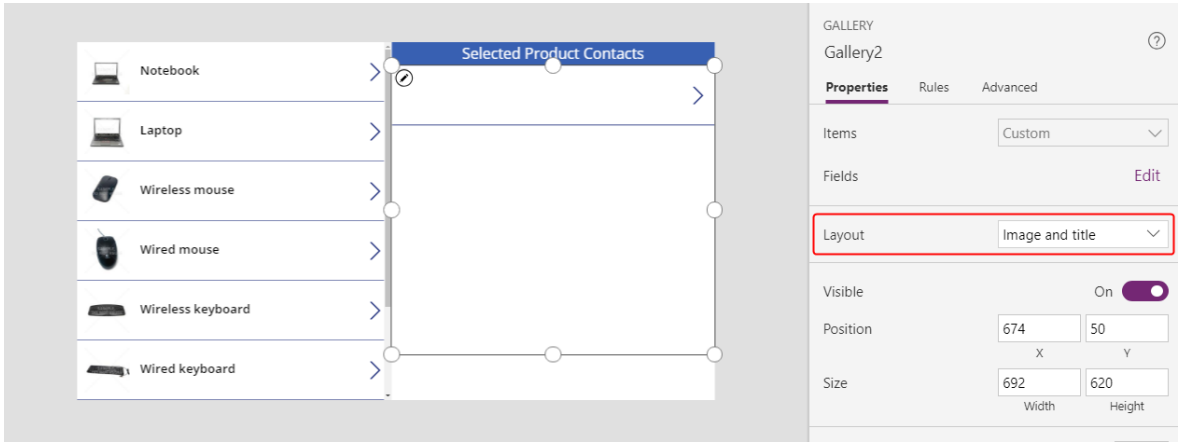
2. Add another blank vertical **Gallery** control, ensure that it's named **Gallery2**, and then move it into the upper-right corner of the screen.

   Later in this topic, you'll add a **Combo box** control under **Gallery2**.

3. In the formula bar, set **Gallery2**'s **Items** property to **Gallery1.Selected.Contacts**.

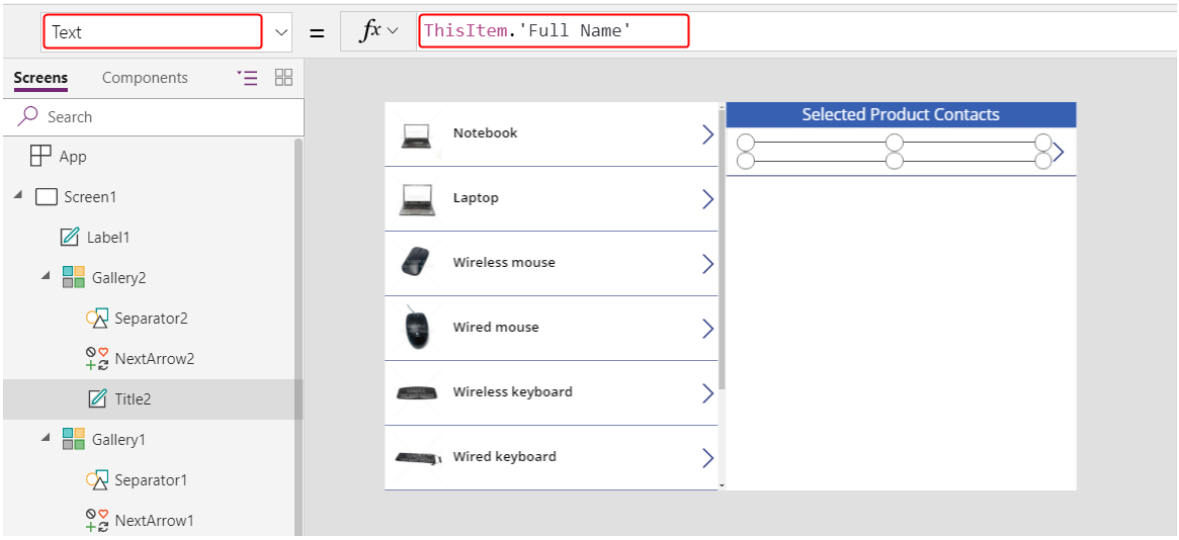


4. On the **Properties** tab, set **Layout** to **Image and title**.



5. In **Gallery2**, ensure that the **Label** control is named **Title2**, and then set its **Text** property to **ThisItem.'Full Name'**.

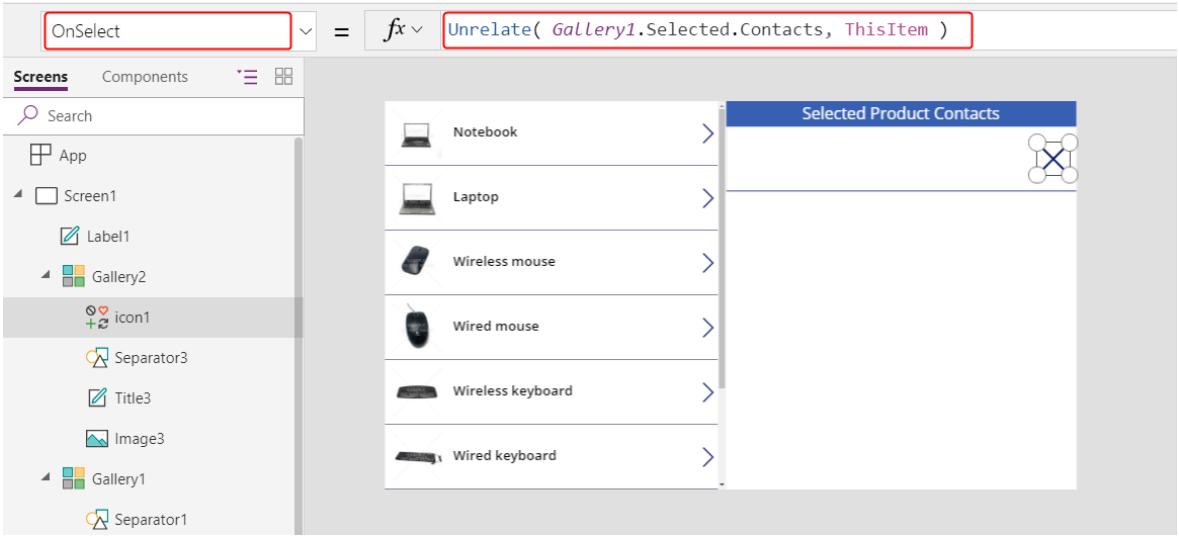
   No text will appear in that control until you finish this procedure and assign a contact to a product.



6. Delete **NextArrow2**, insert a **Cancel** icon, and ensure that it's named **icon1**.

7. Set the **Cancel** icon's **OnSelect** property to this formula:

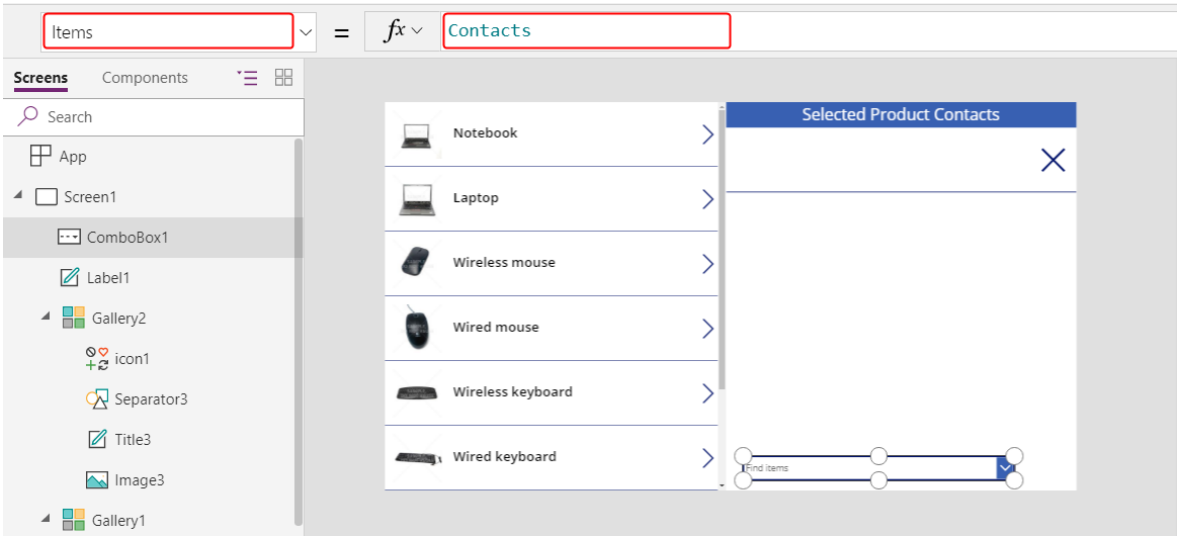
```
Power Apps

Unrelate( Gallery1.Selected.Contacts, ThisItem )
```



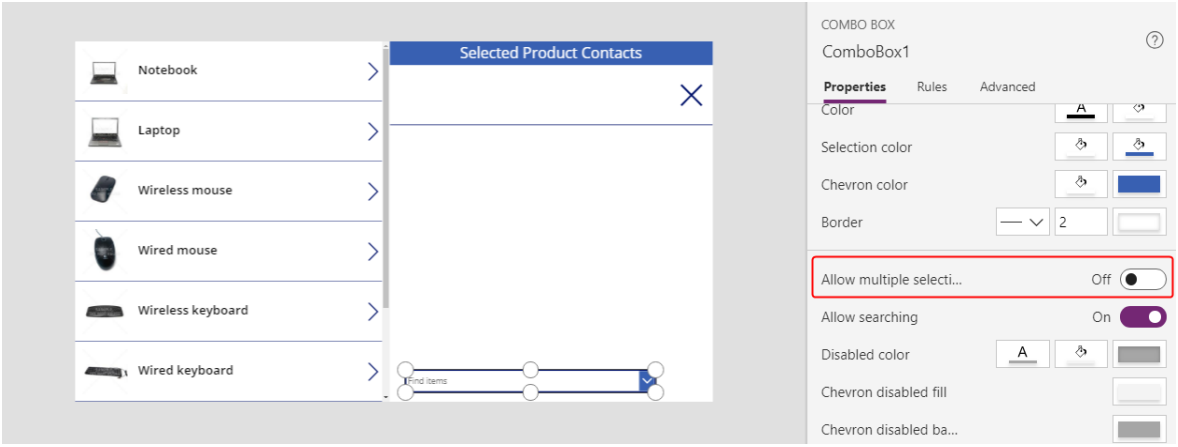
8. On the **View** tab, select **Data sources**.

9. In the **Data** pane, select **Add data source** > **Microsoft Dataverse** > **Contacts** > **Connect**.

10. Under **Gallery2**, add a **Combo box** control, ensure that it's named **ComboBox1**, and then set its **Items** property to **Contacts**.



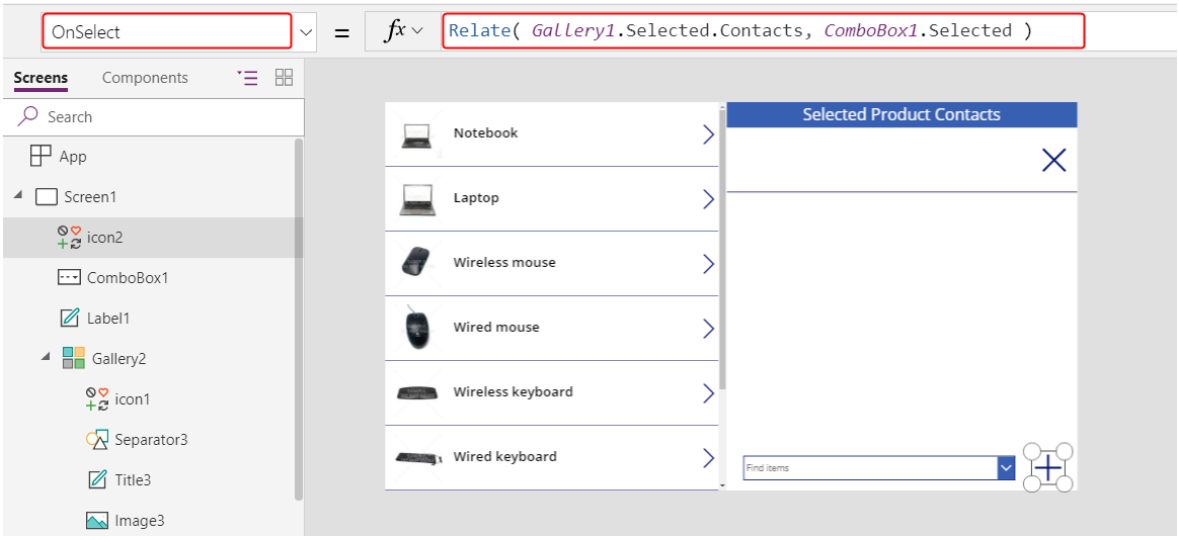
11. On the **Properties** tab, set **Allow multiple selection** to **Off**.



12. Insert an **Add** icon, and set its **OnSelect** property to this formula:

Power Apps

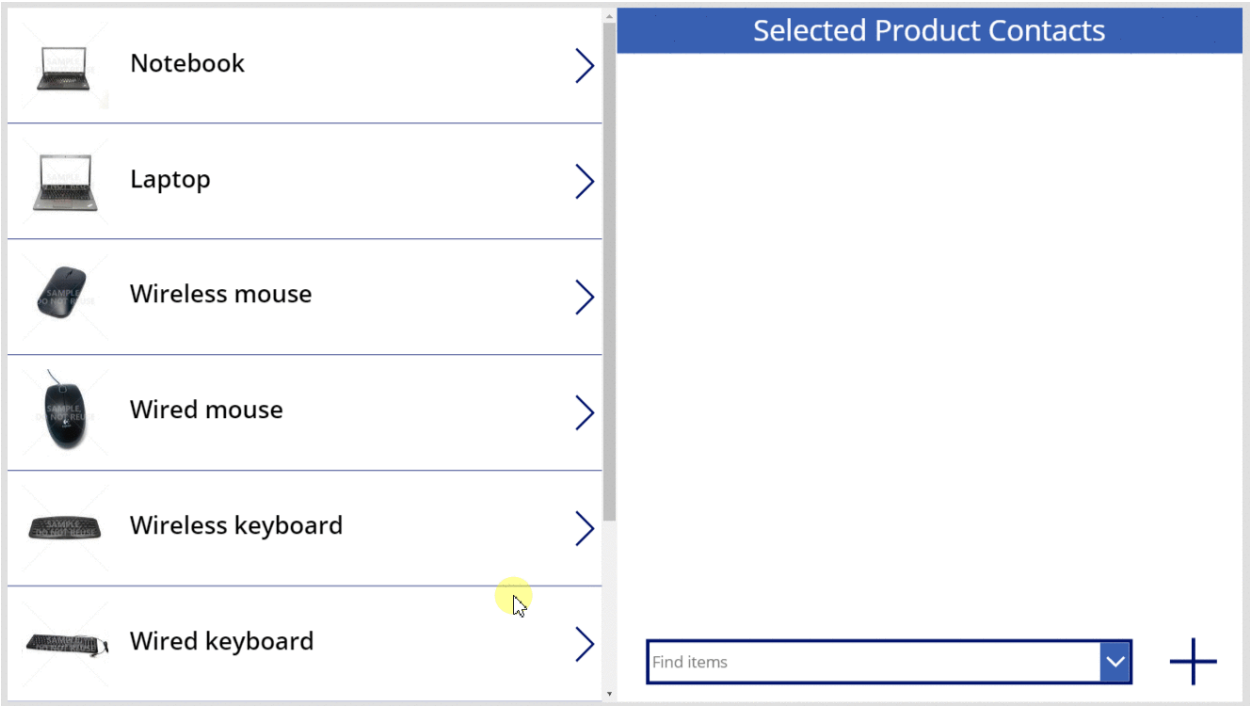
```
Relate( Gallery1.Selected.Contacts, ComboBox1.Selected )
```



With this app, users can now freely relate and unrelate a set of contacts to each product.

- To add a contact to a product, select the contact in the combo box at the bottom of the screen, and then select the **Add** icon.
- To remove a contact from a product, select the **Cancel** icon for that contact.

  Unlike one-to-many, a many-to-many relationship allows users to associate the same contact with multiple products.

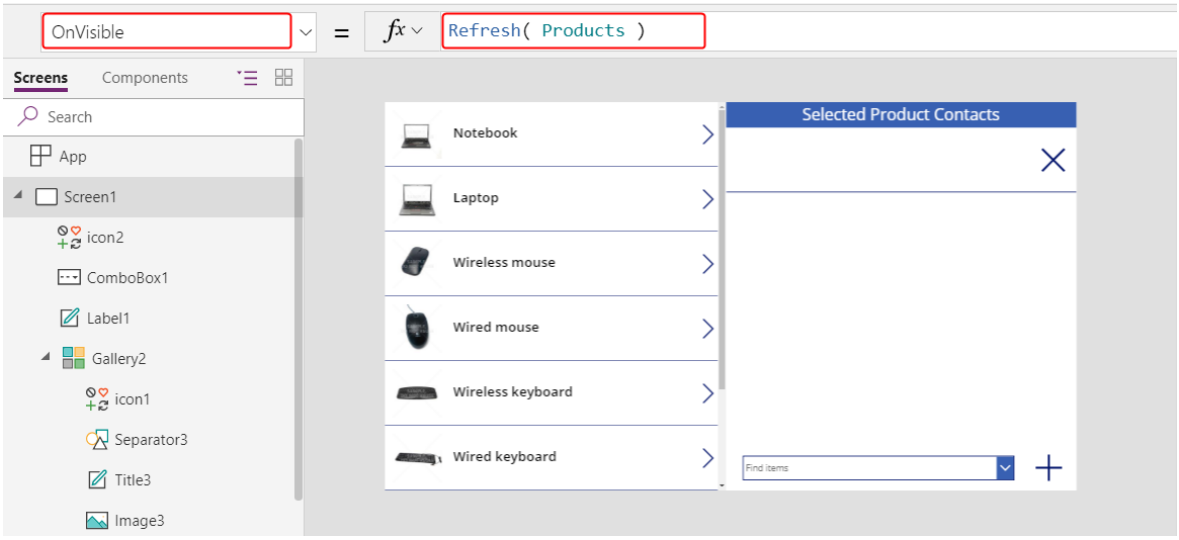


In reverse: relate and unrelate products with multiple contacts

Many-to-many relationships are symmetric. You can extend the example to add products to a contact and then flip between the two screens to show how the relationship appears from either direction.

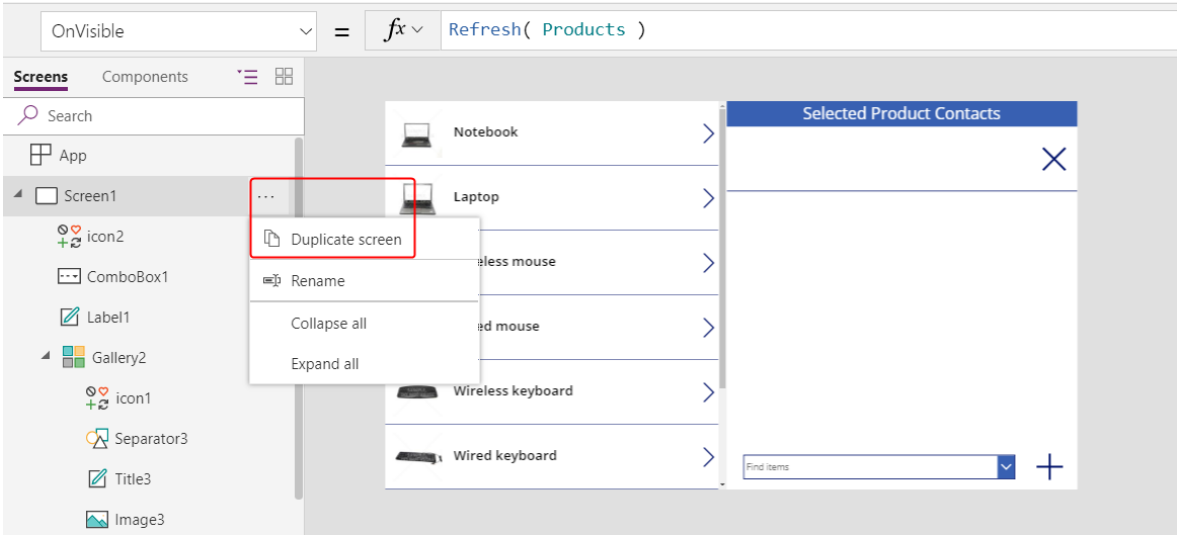
1. Set the **OnVisible** property of **Screen1** to **Refresh(Products)**.

   When you update a one-to-many or many-to-many relationship, only the data of the first argument table of the **Relate** or **Unrelate** call is refreshed. The second must be refreshed manually if you want to flip between the screens of this app.



2. Duplicate **Screen1**.

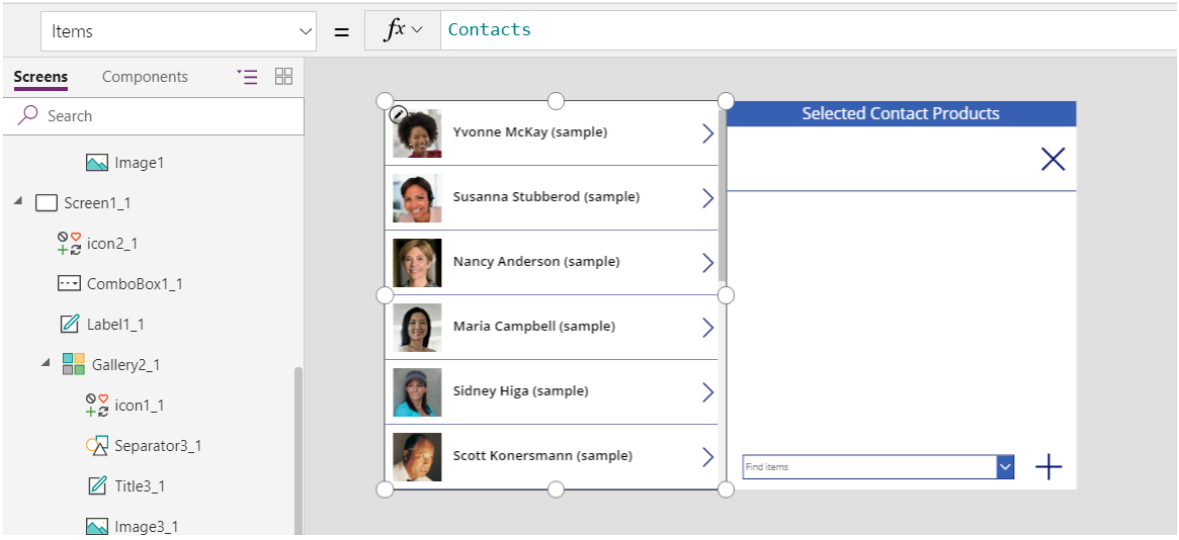
   The duplicate will be named **Screen1_1** and form the basis for looking at the relationships from the contacts side.



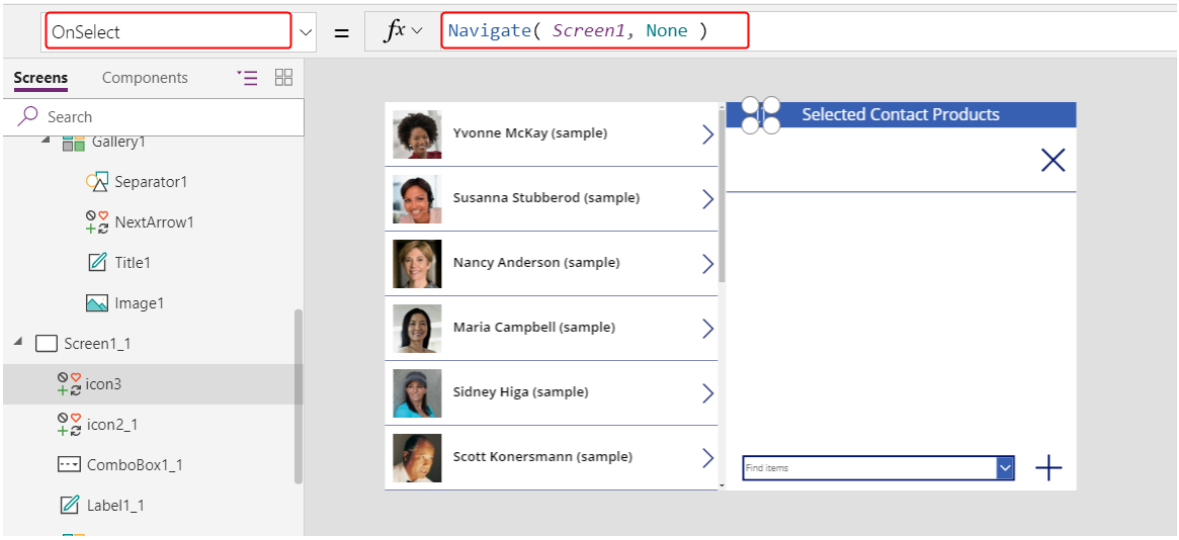
3. To create the reverse view, change these formulas on the controls of **Screen1_1**:

   - Screen1_1.OnVisible = `Refresh( Contacts )`
   - Gallery1_1.Items = `Contacts`
   - Title1_1.Text = `ThisItem.'Full Name'`
   - Label1_1.Text = `"Selected Contact Products"`
   - Gallery2_1.Items = `Gallery1_1.Selected.Products`
   - Title2_1.Text = `ThisItem.Name`
   - Icon1_1.OnSelect = `Unrelate( Gallery1_1.Selected.Products, ThisItem )`
   - ComboBox1_1.Items = `Products`
   - Icon2_1.OnSelect = `Relate( Gallery1_1.Selected.Products, ComboBox1_1.Selected )`

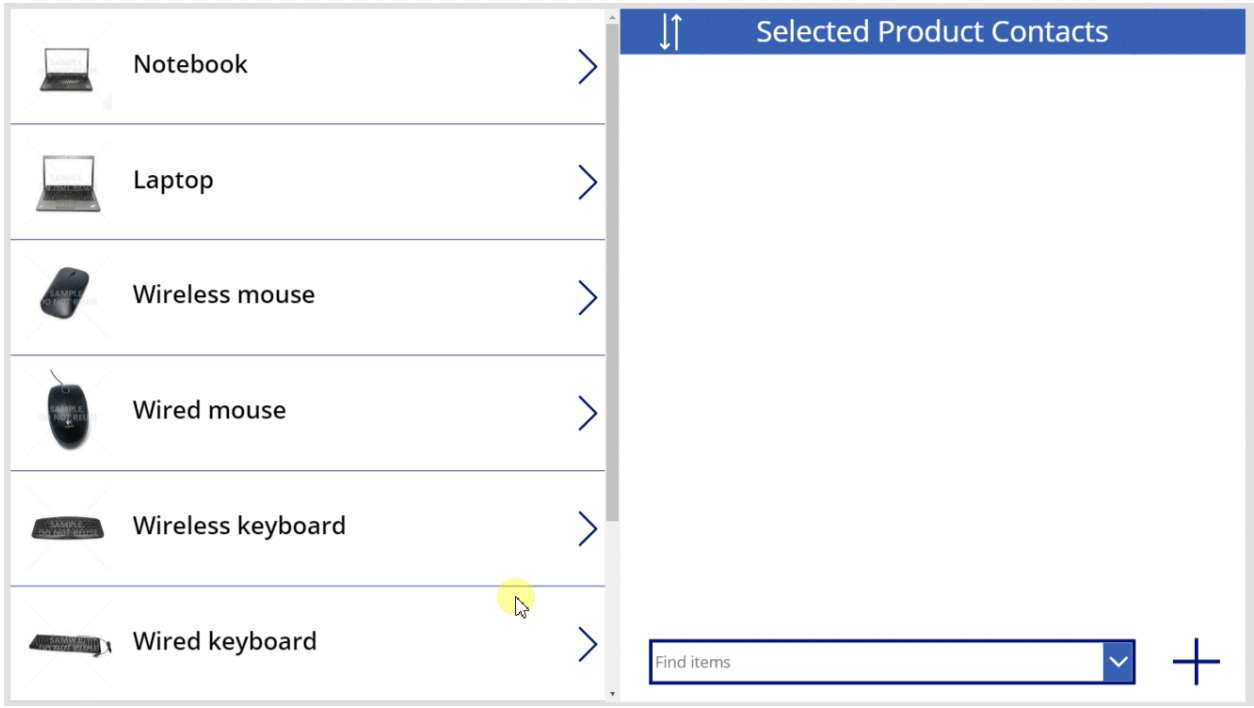
The result will look very similar to the previous screen but comes at the relationship from the **Contacts** side.



4. Insert an **Arrows up down** icon and set its **OnSelect** property to **Navigate(Screen1, None)**. Do the same thing on **Screen1** with the formula **Navigate(Screen1_1, None)**.



With this new screen, users can add a contact to a product and then flip to a view of contacts and see the associated product. The relationships are symmetric and shared between the two screens.


Notebook
Laptop
Wireless mouse
Wired mouse
Wireless keyboard
Wired keyboard
Selected Product Contacts
Find items

Remove and RemoveIf functions

Article • 10/31/2024

Applies to: Canvas apps Model-driven apps Power Platform CLI

Removes records from a data source.

> **Note**
>
> PAC CLI pac power-fx commands do not support the **RemoveIf** function.

Description

Remove function

Use the **Remove** function to remove a specific record or records from a data source.

For collections, the entire record must match. You can use the **RemoveFlags.All** argument to remove all copies of a record; otherwise, only one copy of the record is removed.

RemoveIf function

Use the **RemoveIf** function to remove a record or records based on a condition or a set of conditions. Each condition can be any formula that results in a **true** or **false** and can reference columns of the data source by name. Each condition is evaluated individually for each record, and the record is removed if all conditions evaluate to **true**.

Remove and **RemoveIf** return the modified data source as a table. You can use both functions only in behavior formulas.

You can also use the **Clear** function to remove all of the records in a collection.

Delegation

RemoveIf is supported by only a few data sources. For data sources that don't upport this feature, Power Apps will query the server and retrieve all data matching the filter expression, up to a maximum of either 500 or 2000 records or the data page size. Subsequently, it will delete each of those records individually by making separate calls to the server.

Syntax

Remove(*DataSource*, *Record1* [, *Record2*, ...] [, **RemoveFlags.All**])

- *DataSource* – Required. The data source that contains the record or records that you want to remove.
- *Record(s)* – Required. The record or records to remove.
- **RemoveFlags.All** – Optional. In a collection, the same record may appear more than once. You can add the **RemoveFlags.All** argument to remove all copies of the record.

Remove(*DataSource*, *Table* [, **RemoveFlags.All**])

- *DataSource* – Required. The data source that contains the records that you want to remove.
- *Table* – Required. A table of records to remove.
- **RemoveFlags.All** – Optional. In a collection, the same record may appear more than once. You can add the **RemoveFlags.All** argument to remove all copies of the record.

RemoveIf(*DataSource*, *Condition* [, ...])

- *DataSource* – Required. The data source that contains the record or records that you want to remove.
- *Condition(s)* – Required. A formula that evaluates to **true** for the record or records to remove. You can use column names from the *DataSource* in the formula. If you specify multiple *Conditions*, all must evaluate to **true** for the record or records to be removed.

Examples - single formulas

In these examples, you'll remove a record or records in a data source that's named **IceCream** and that starts with the data in this table:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	300

Create a collection with sample records

To create a collection with this data:

1. Insert a **Button** control.

2. Set button control's **OnSelect** property to the below formula:

```
Power Fx
```

```
ClearCollect( IceCream,
              { ID: 1, Flavor: "Chocolate",  Quantity: 100 },
              { ID: 2, Flavor: "Vanilla",    Quantity: 200 },
              { ID: 3, Flavor: "Strawberry", Quantity: 300 }
)
```

3. Select the button while holding down the Alt key:

Remove sample records from collection using a formula

Expand table

Formula	Description	Result
Remove(IceCream, LookUp(IceCream, Flavor="Chocolate"))	Removes the **Chocolate** record from the data source.	 The **IceCream** data source has been modified.
Remove(IceCream, LookUp(IceCream, Flavor="Chocolate"), LookUp(IceCream, Flavor="Strawberry"))	Removes two records from the data source.	 The **IceCream** data source has been modified.
RemoveIf(IceCream, Quantity > 150)	Removes records that have a **Quantity** that's greater than **150**.	 The **IceCream** data source has been modified.
RemoveIf(IceCream, Quantity > 150, Left(Flavor, 1) = "S")	Removes records that have a **Quantity** that's greater than 150 and **Flavor** starts with an **S**.	 The **IceCream** data source has been modified.

Formula	Description	Result
RemoveIf(IceCream, true)	Removes all records from the data source.	ID Flavor Quantity The **IceCream** data source has been modified.

Examples - remove button outside a gallery

In this example, you'll use a **Gallery** control to list the records in a table. And then use the **Remove** function to selectively remove an item.

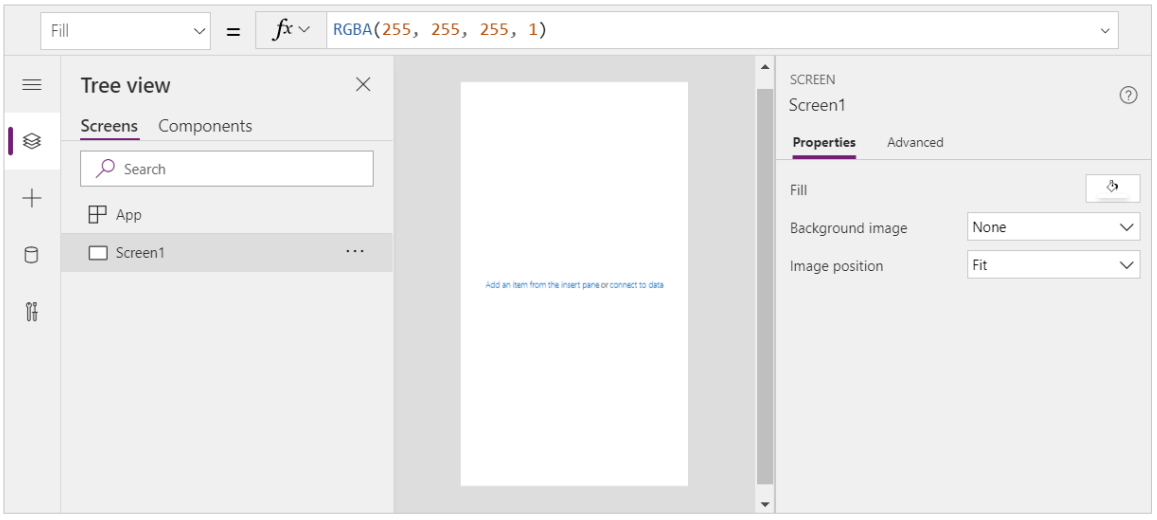
Prepare for sample data

This example uses the **Contacts** table in Microsoft Dataverse available with the *sample apps and data*. You can deploy *sample apps and data* when you create an environment. You can also use any other data source instead.

Remove button outside a gallery

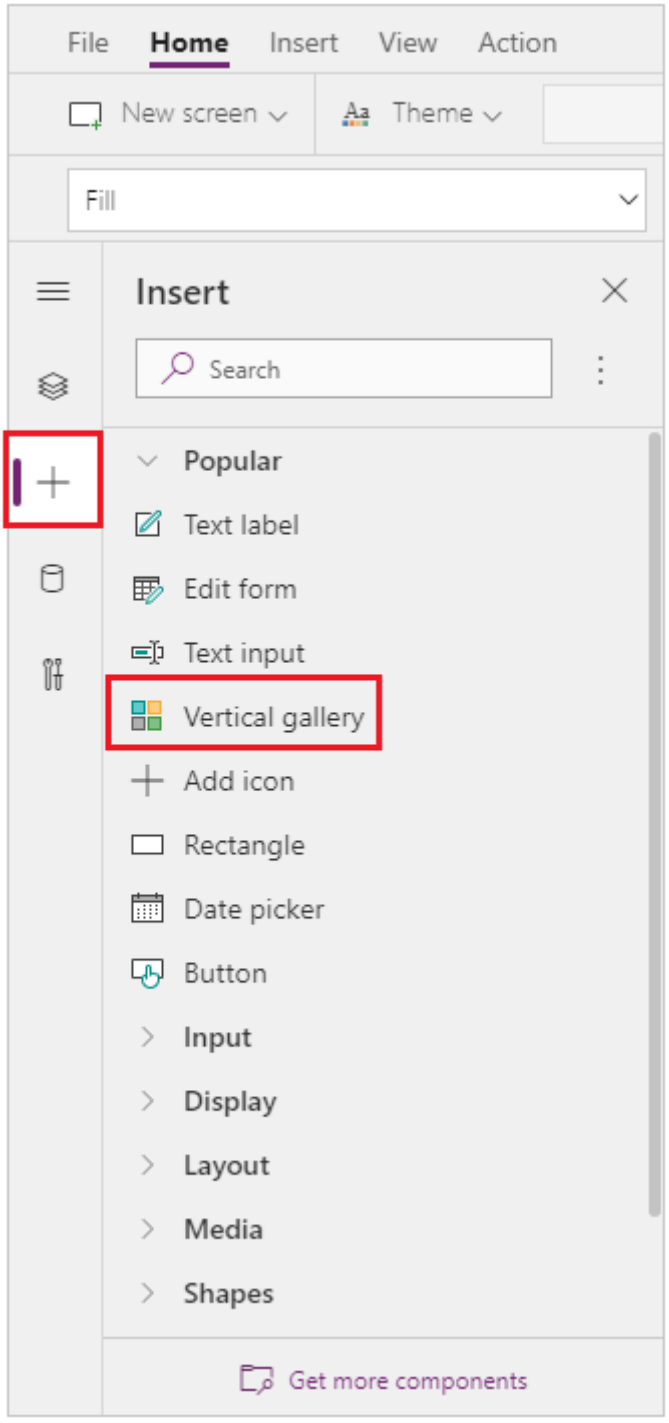
In this example, you'll remove an item by using a *button* that is outside the gallery.

1. Create a new blank canvas app using a Phone layout.

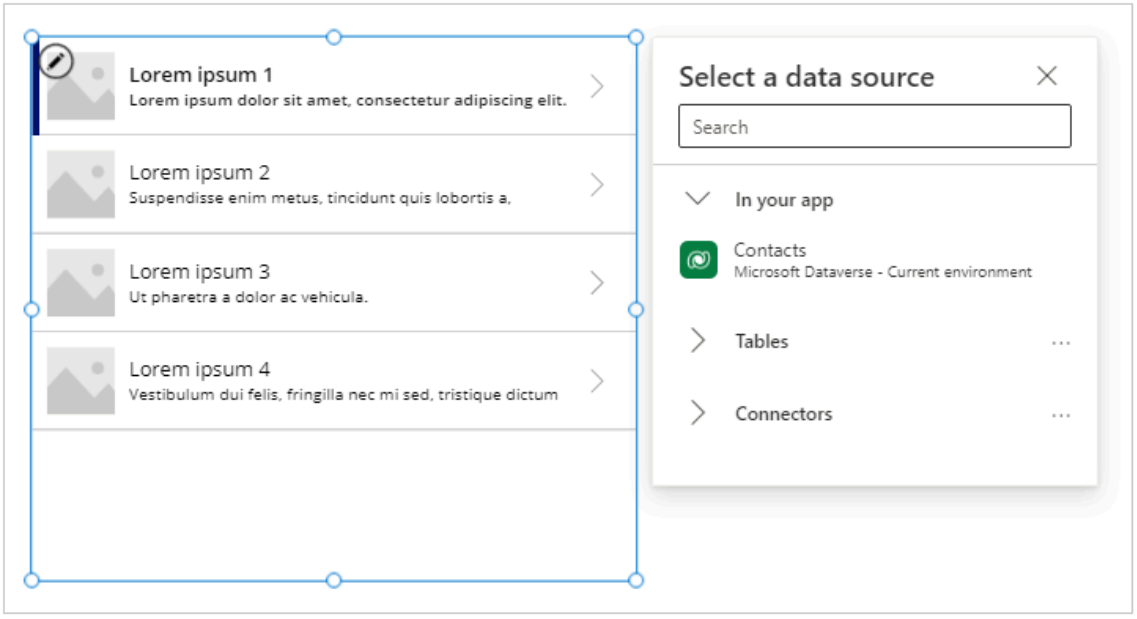


2. Select the **Insert** from the left pane.

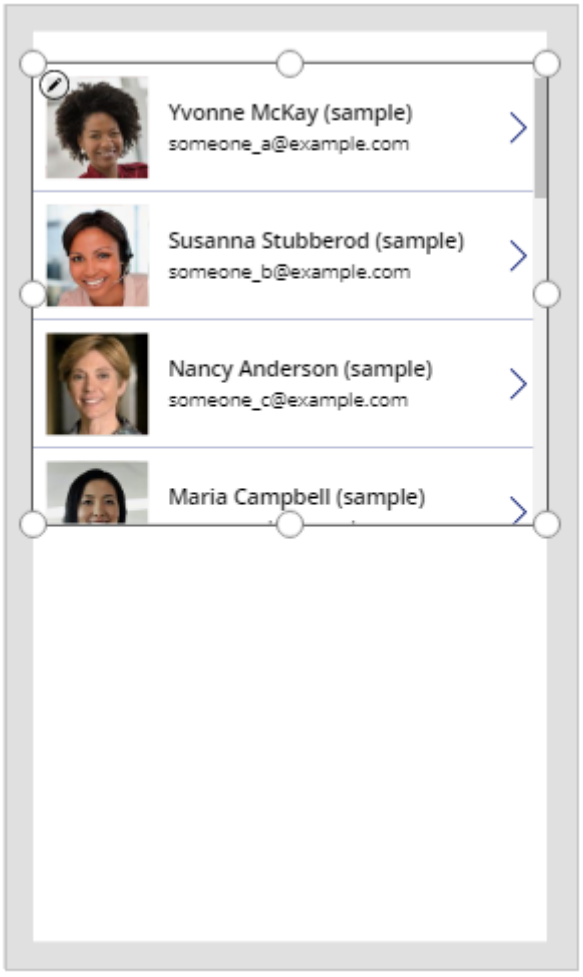
3. Select **Vertical gallery**.
   A **Gallery** control is be added to your screen.



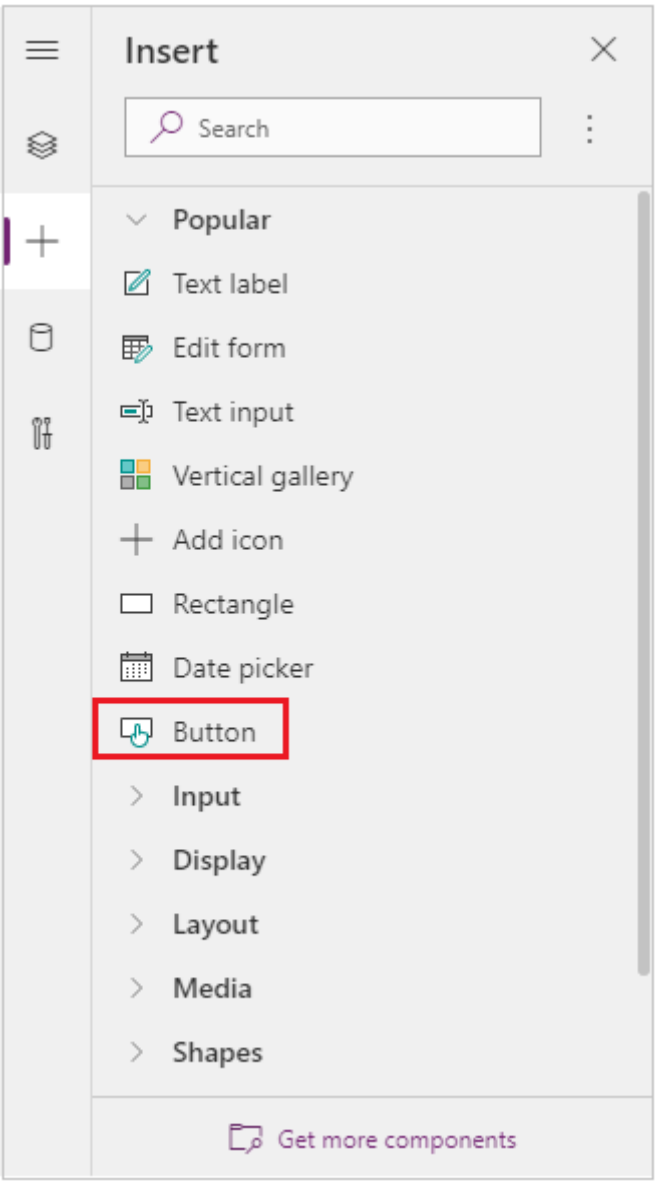
4. You're prompted to select a data source where you can select a data source from the available data sources.
   For example, select the **Contacts** table to use *sample data*:



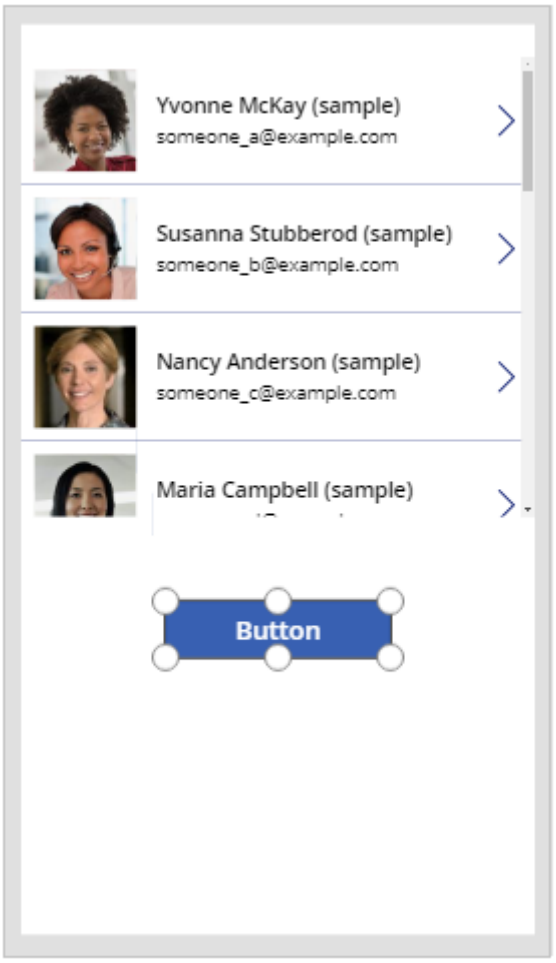
The gallery shows items from this table:



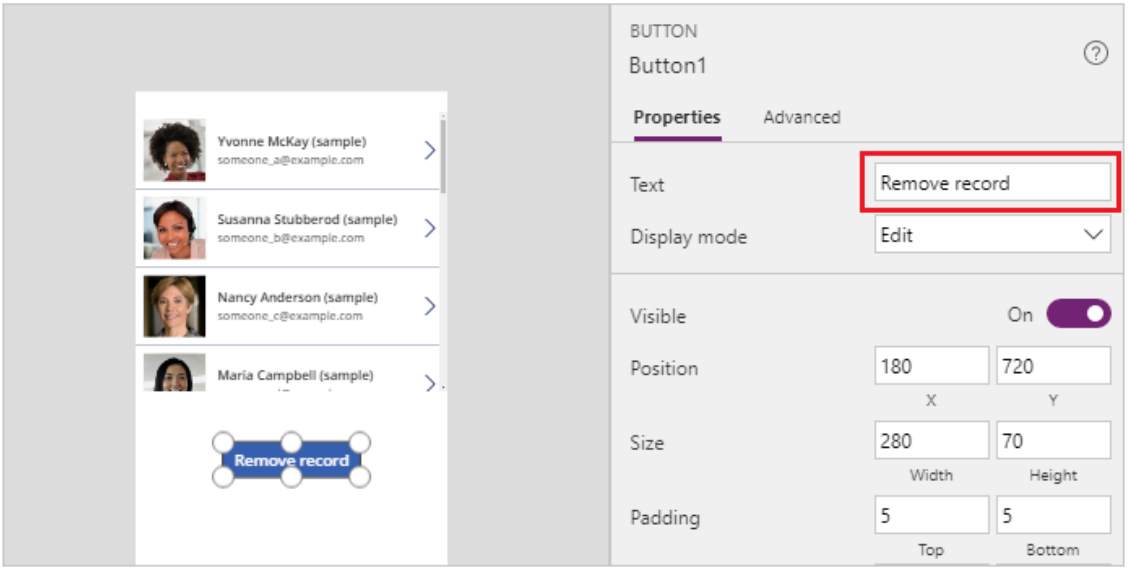
5. Insert a **Button** control from left pane:



6. Move the added button below the gallery items:

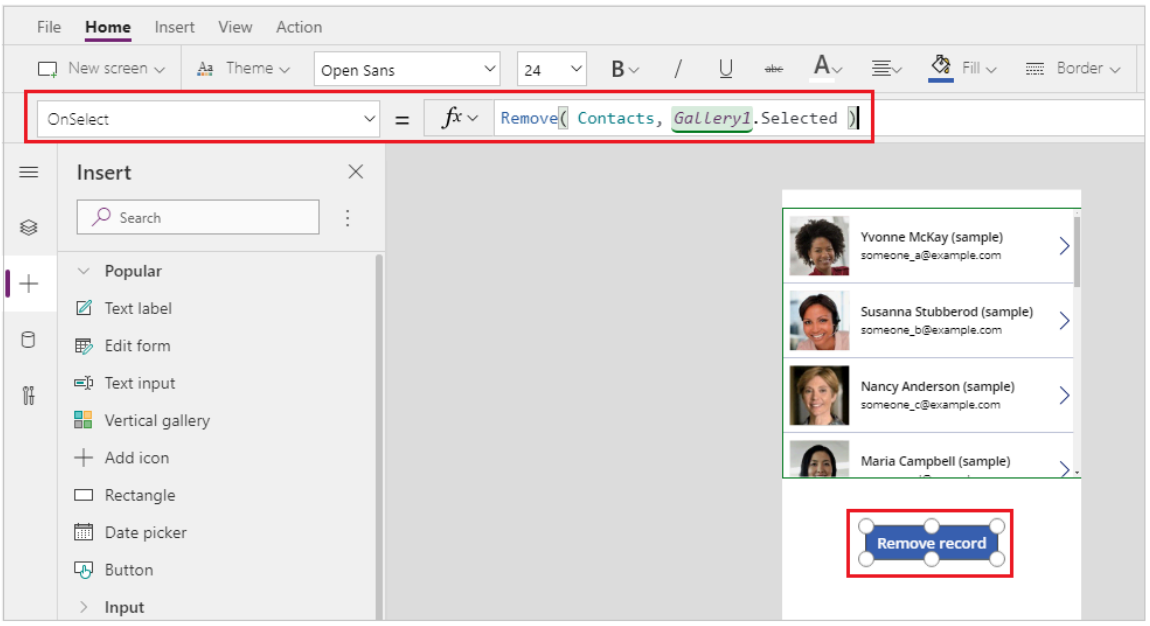


7. Update button text property to *Remove record*. You can also use text of your choice:



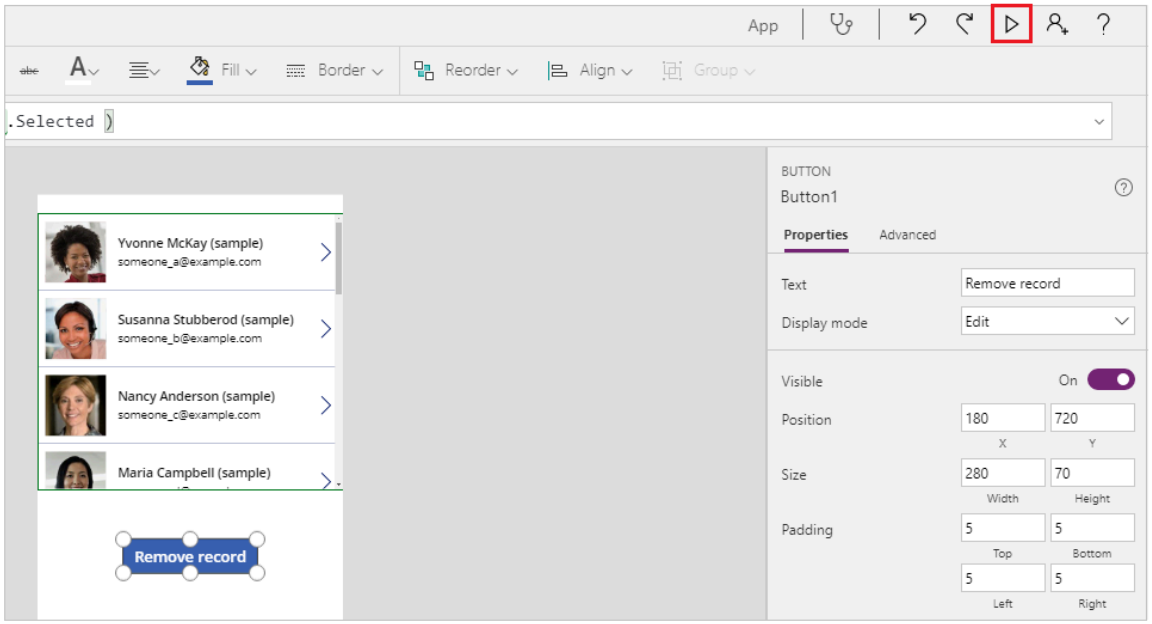
8. Set the **OnSelect** property for this button control to the following formula:

```
Power Fx

Remove( Contacts, Gallery1.Selected )
```

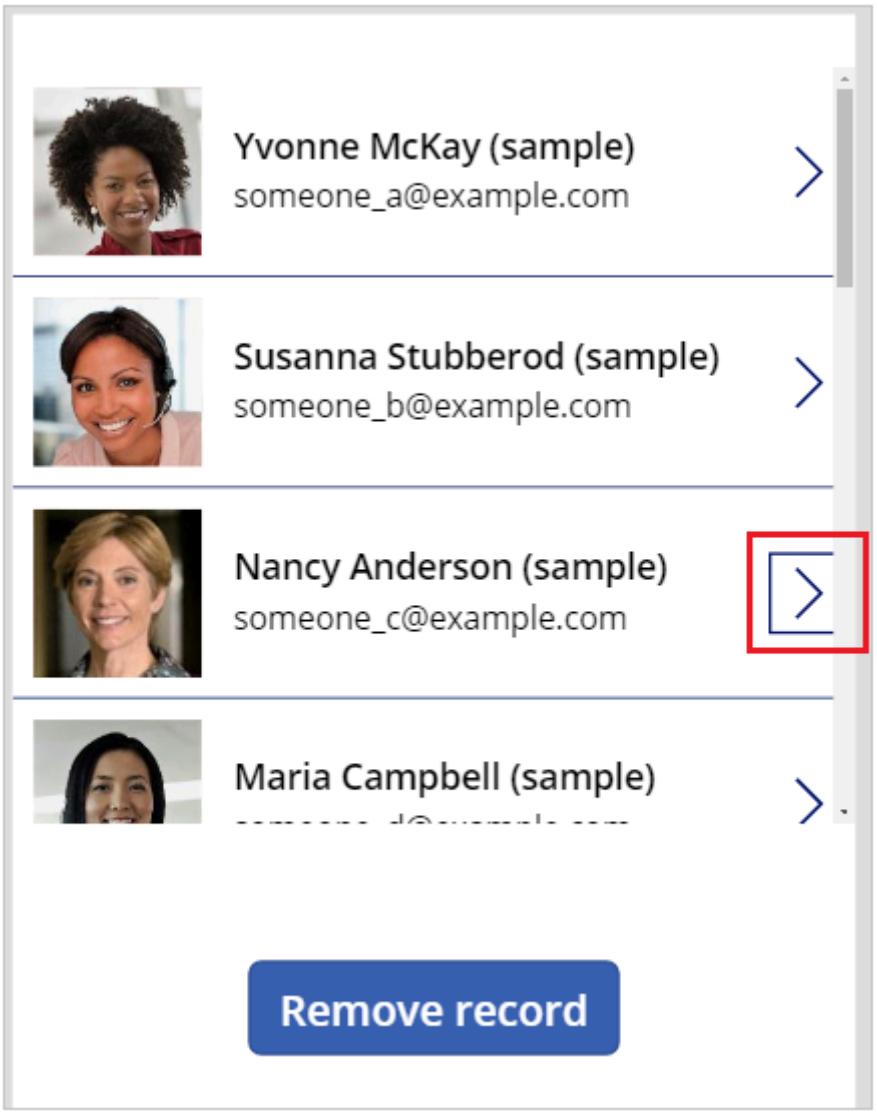


The gallery control makes the currently selected record available using **Selected** property. **Remove** function refers to this selected record to remove it.

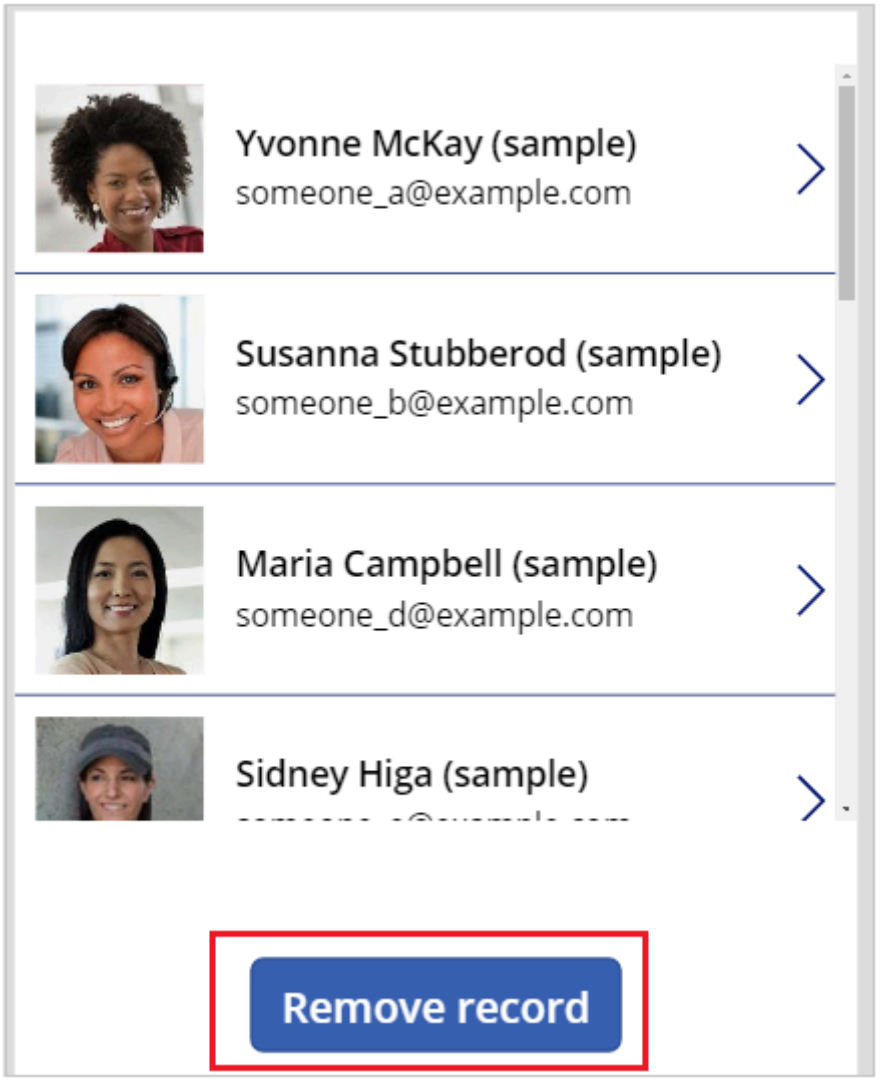
9. Preview the app using the *Play* button on the top right, or press *F5* on keyboard:



10. Select a record to remove, such as *Nancy*'s record in this example:



11. Select **Remove record**:



Selecting the button removes the selected record (in this example, Nancy's record).

12. Close the app preview.

> **Tip**
>
> You can also use alternate behavior with ***Alt key*** instead of using the app preview with *Play* button or *F5*.

Examples - trash can icon inside a gallery

In this example, you'll remove an item by using an *icon* placed inside the gallery.

Create a collection with sample data

If you already have prepared sample data, skip this step and move to Trash can icon inside a gallery.

1. Add a **Button** control to your screen.

2. Set the **OnSelect** property to the following formula:

```
Power Fx
```

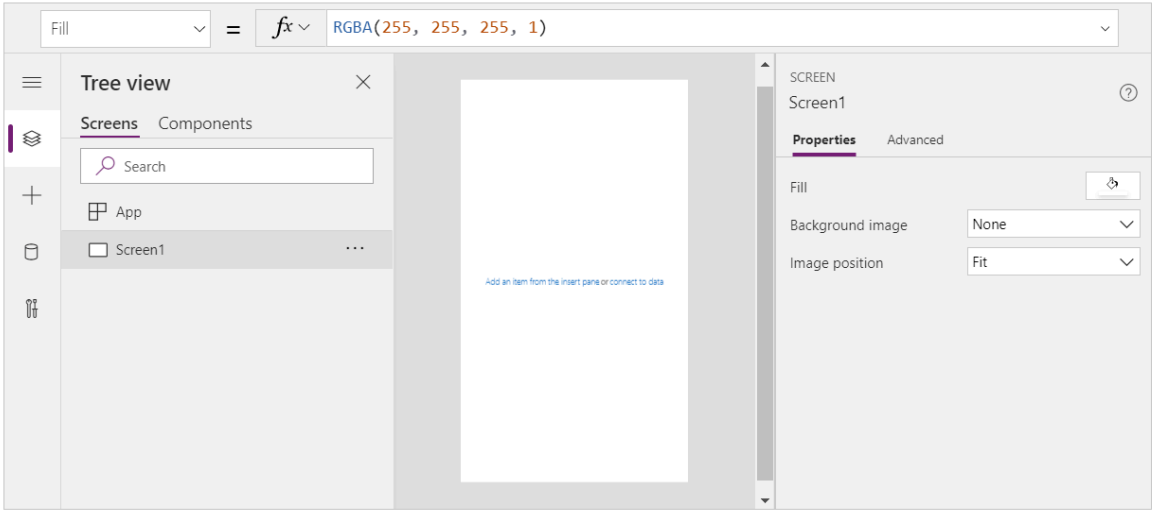
```
ClearCollect( SampleContacts,
      { 'Full Name': "Yvonne McKay (sample)",      'Primary Email':
"someone_a@example.com" },
      { 'Full Name': "Susanna Stubberod (sample)", 'Primary Email':
"someone_b@example.com" },
      { 'Full Name': "Nancy Anderson (sample)",    'Primary Email':
"someone_c@example.com" },
      { 'Full Name': "Maria Campbell (sample)",    'Primary Email':
"someone_d@example.com" },
      { 'Full Name': "Robert Lyon (sample)",       'Primary Email':
"someone_e@example.com" },
      { 'Full Name': "Paul Cannon (sample)",       'Primary Email':
"someone_f@example.com" },
      { 'Full Name': "Rene Valdes (sample)",       'Primary Email':
"someone_g@example.com" }
)
```

3. Select the button while holding down the Alt key.

Sample collection is created that you can use in the following example.

Trash can icon inside a gallery

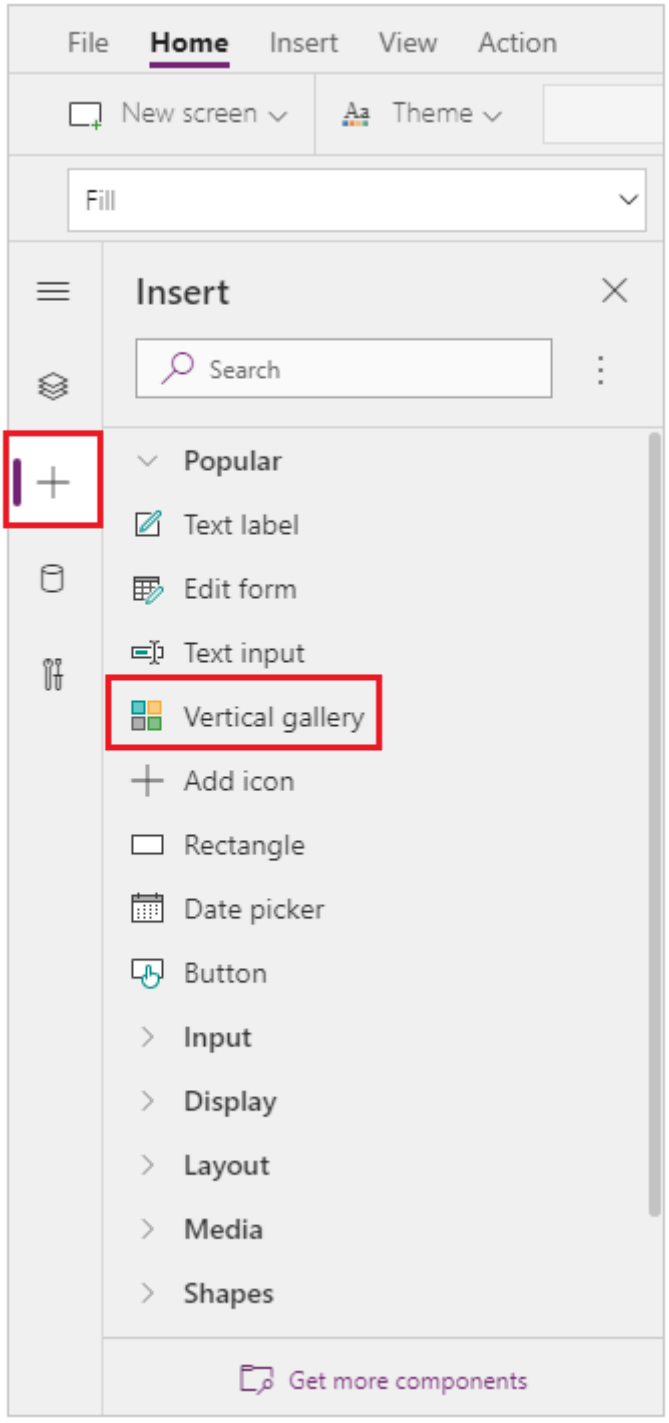
1. Create a new blank canvas app using a Phone layout.



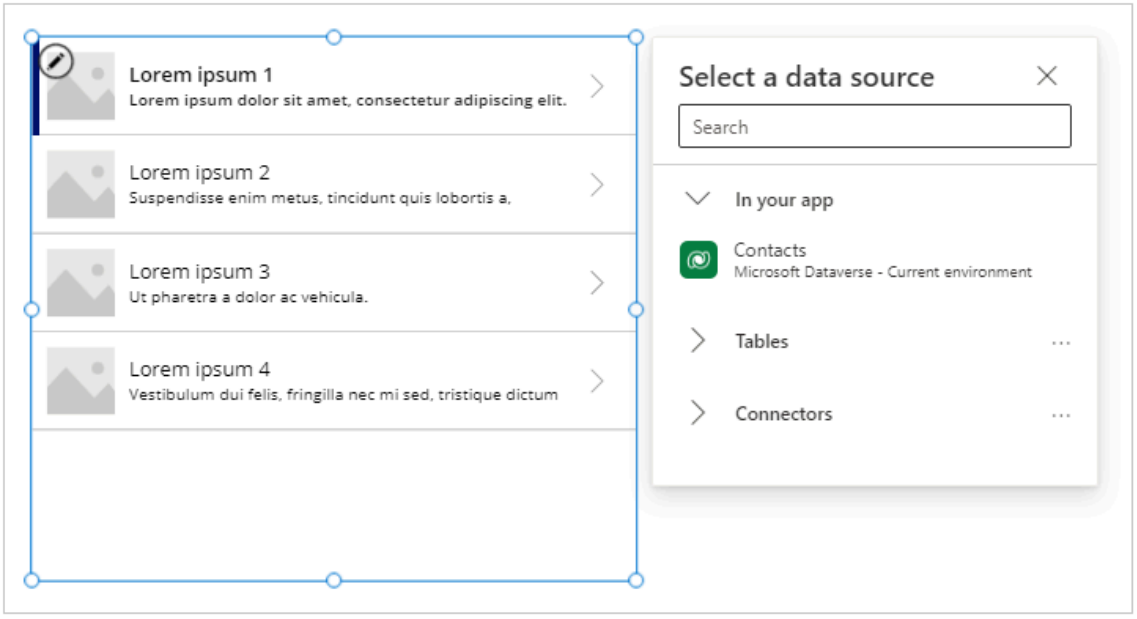
2. Select the **Insert** from the left pane.

3. Select **Vertical gallery**.

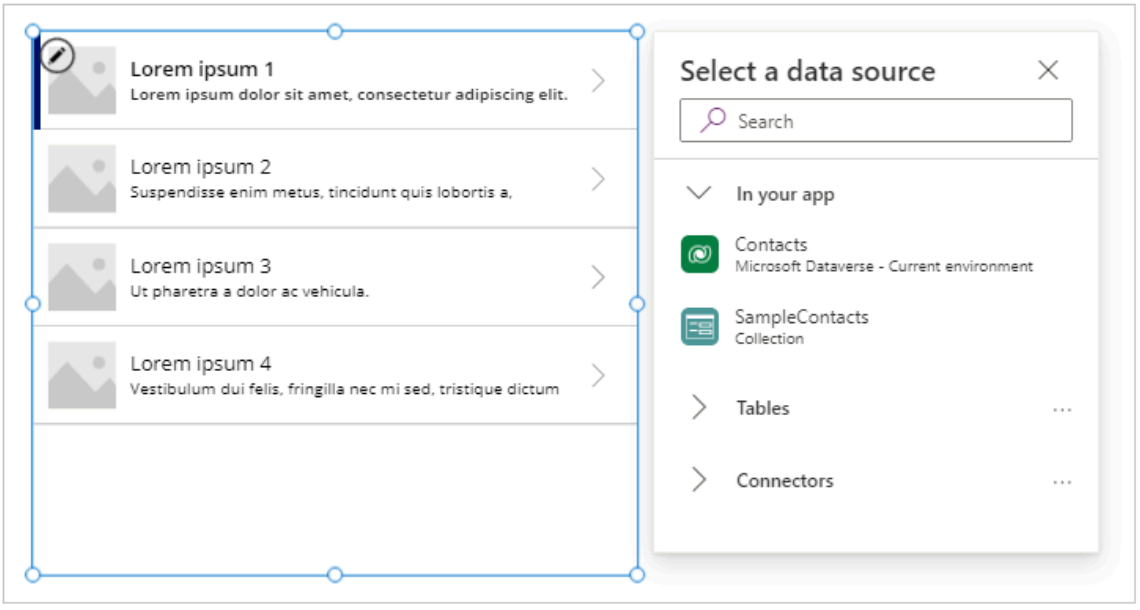
   A **Gallery** control is be added to your screen.



4. You're prompted to select a data source where you can select a data source from the available data sources.
   For example, select the **Contacts** table to use *sample data*:

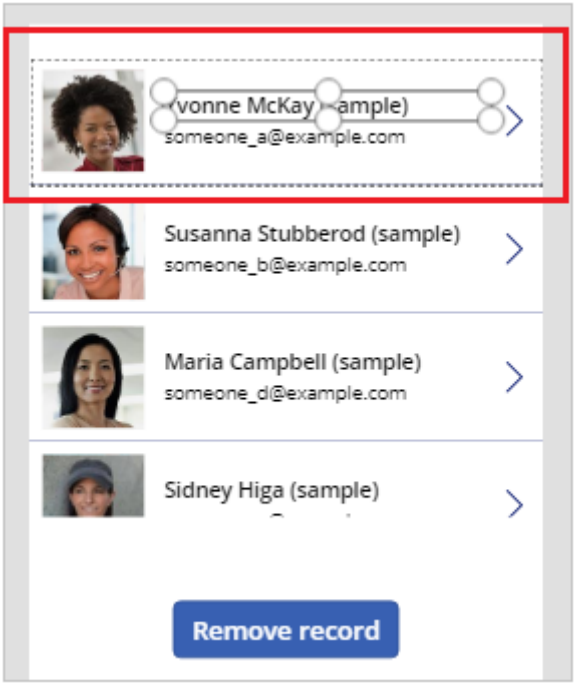


If you created a collection, select your collection instead:

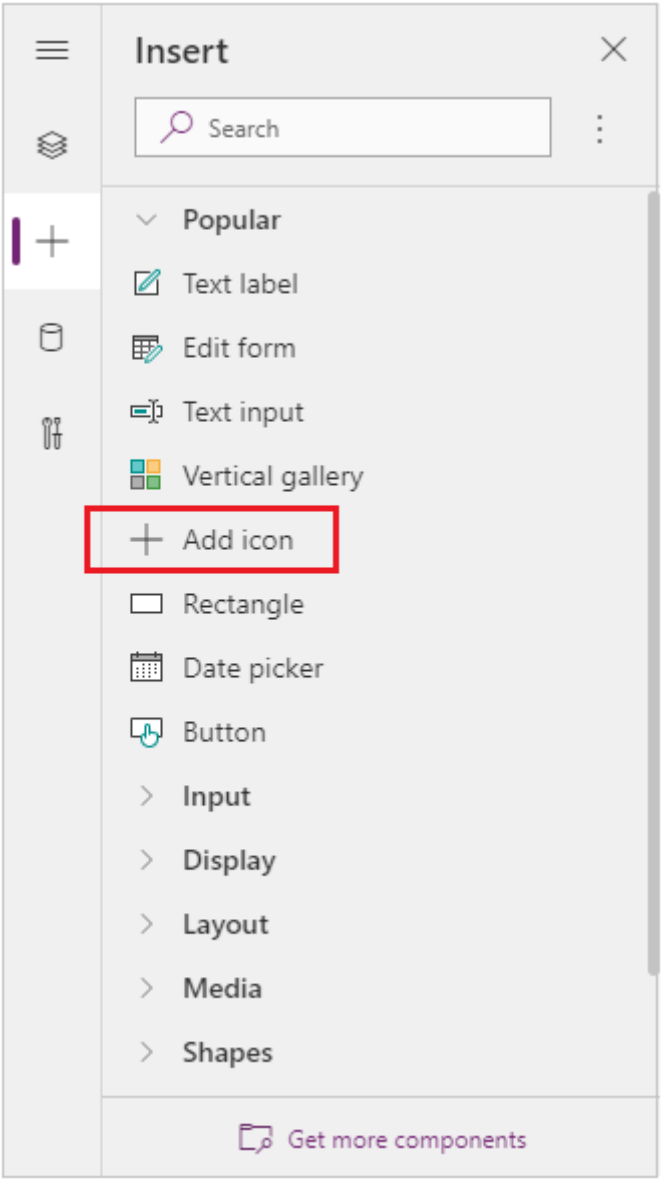


5. Select a control within the top item in the gallery.

   To ensure next step inserts item into gallery's template and not outside the gallery, ensure you follow this step before moving to the next step.

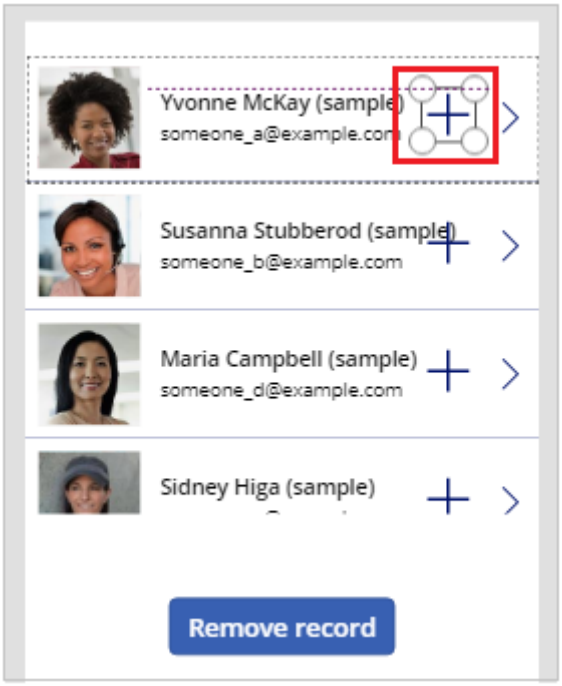


6. Select **Add icon** from left pane.



> ⓘ Note
>
> **Add icon** inserts a + icon on the left side of the gallery, replicated for each item in the gallery.

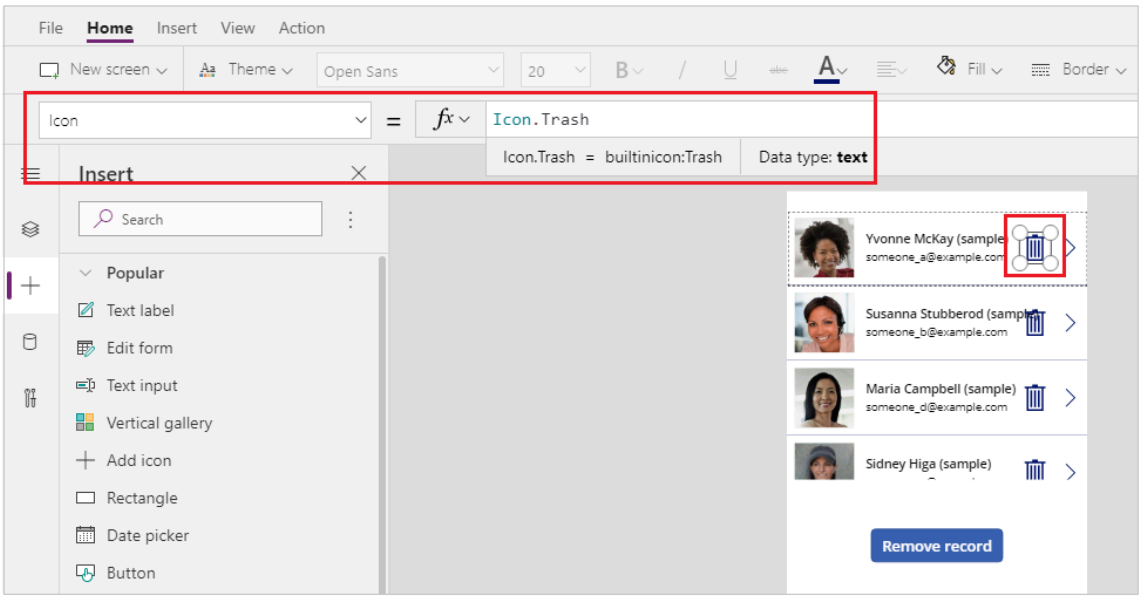
7. In the top item, move the icon to the right side of the screen.



8. Select the **Icon** property for icon and set it to the following formula to update the icon image as trash icon:

```
Power Fx

Icon.Trash
```

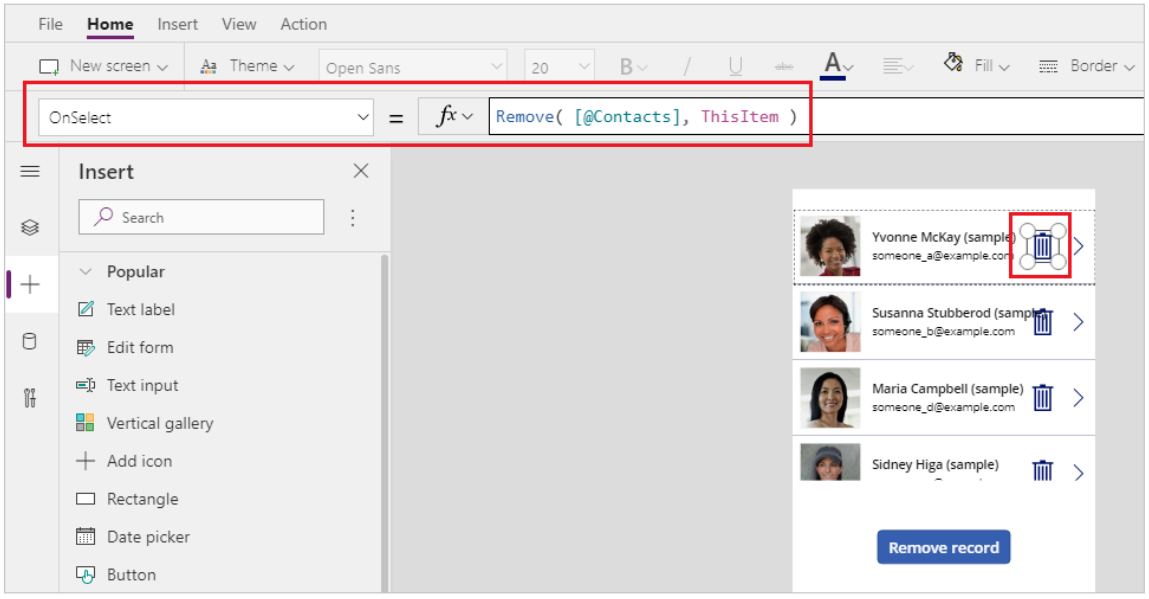
> ⓘ Note
>
> The **Icon.** prefix is only shown when you're actively editing the formula.



9. Set the **OnSelect** property to the following formula:

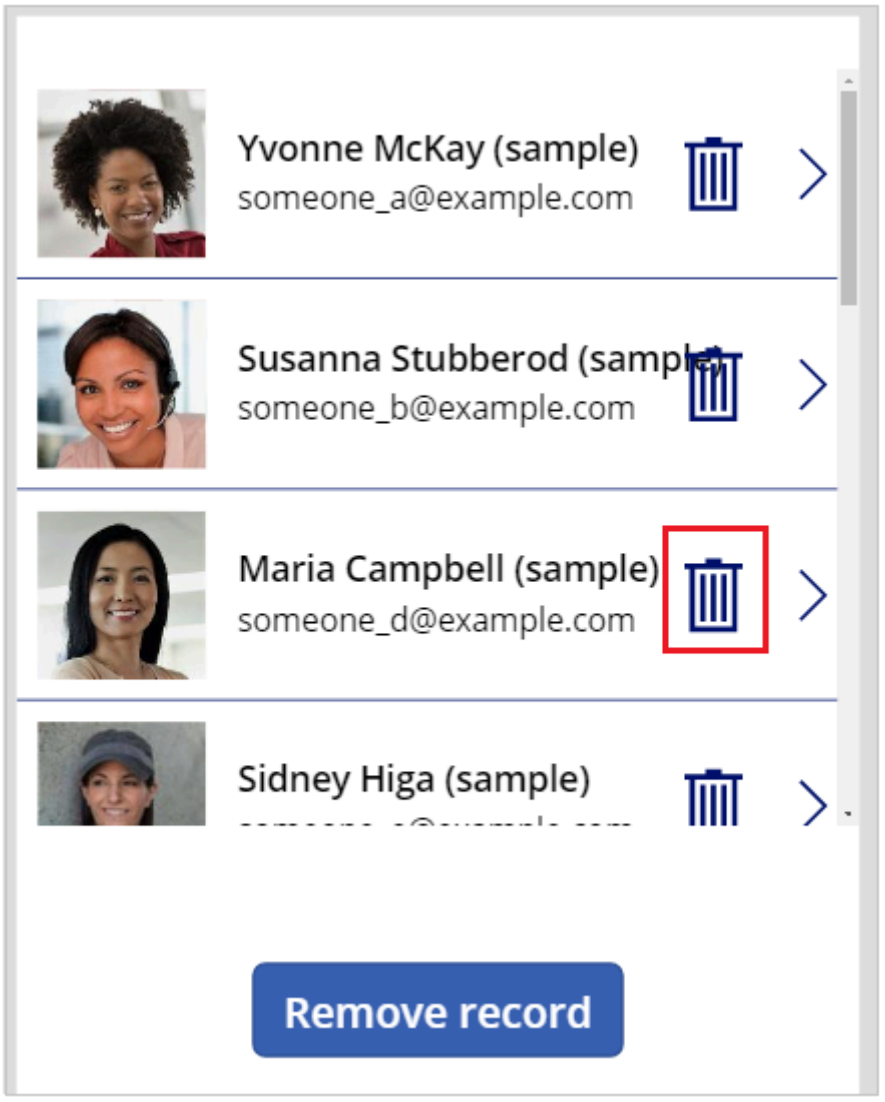
```
Power Fx

Remove( [@Contacts], ThisItem )
```

> **Note**
>
> You must use global disambiguation operator [@...] in this example with sample data that uses the *Contacts* table to avoid conflict with a *One-to-Many* relationship. If you use data sources such as a list or a SQL Server table, using *global disambiguation operator* is not required.

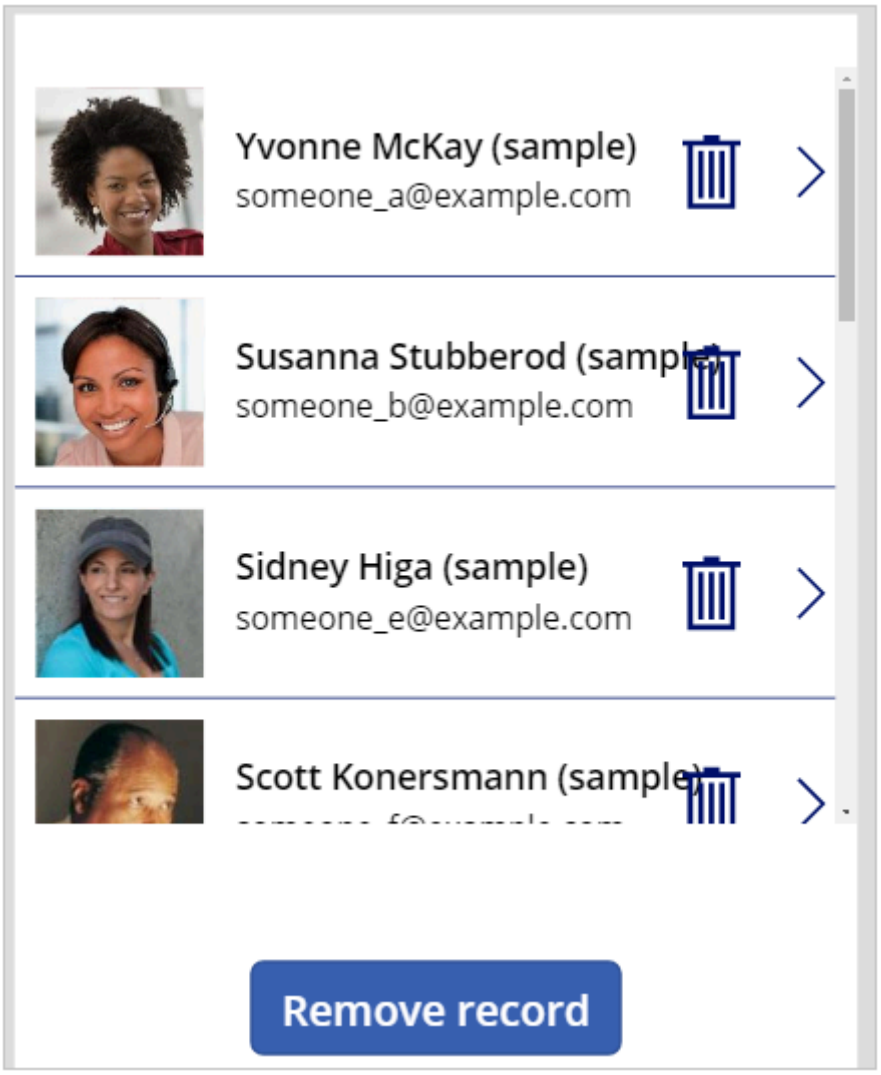


10. Preview the app using the *Play* button on the top right, or press *F5* on keyboard.

11. Select the trash icon next to a record, for example *Maria*'s:



The record is deleted:



12. Close the app preview.

Feedback

Was this page helpful? Yes No

Provide product feedback

Remove and RemoveIf functions

Article • 10/31/2024

Applies to: Canvas apps Model-driven apps Power Platform CLI

Removes records from a data source.

> **Note**
>
> PAC CLI pac power-fx commands do not support the **RemoveIf** function.

Description

Remove function

Use the **Remove** function to remove a specific record or records from a data source.

For collections, the entire record must match. You can use the **RemoveFlags.All** argument to remove all copies of a record; otherwise, only one copy of the record is removed.

RemoveIf function

Use the **RemoveIf** function to remove a record or records based on a condition or a set of conditions. Each condition can be any formula that results in a **true** or **false** and can reference columns of the data source by name. Each condition is evaluated individually for each record, and the record is removed if all conditions evaluate to **true**.

Remove and **RemoveIf** return the modified data source as a table. You can use both functions only in behavior formulas.

You can also use the **Clear** function to remove all of the records in a collection.

Delegation

RemoveIf is supported by only a few data sources. For data sources that don't upport this feature, Power Apps will query the server and retrieve all data matching the filter expression, up to a maximum of either 500 or 2000 records or the data page size. Subsequently, it will delete each of those records individually by making separate calls to the server.

Syntax

Remove(*DataSource*, *Record1* [, *Record2*, ...] [, **RemoveFlags.All**])

- *DataSource* – Required. The data source that contains the record or records that you want to remove.
- *Record(s)* – Required. The record or records to remove.
- **RemoveFlags.All** – Optional. In a collection, the same record may appear more than once. You can add the **RemoveFlags.All** argument to remove all copies of the record.

Remove(*DataSource*, *Table* [, **RemoveFlags.All**])

- *DataSource* – Required. The data source that contains the records that you want to remove.
- *Table* – Required. A table of records to remove.
- **RemoveFlags.All** – Optional. In a collection, the same record may appear more than once. You can add the **RemoveFlags.All** argument to remove all copies of the record.

RemoveIf(*DataSource*, *Condition* [, ...])

- *DataSource* – Required. The data source that contains the record or records that you want to remove.
- *Condition(s)* – Required. A formula that evaluates to **true** for the record or records to remove. You can use column names from the *DataSource* in the formula. If you specify multiple *Conditions*, all must evaluate to **true** for the record or records to be removed.

Examples - single formulas

In these examples, you'll remove a record or records in a data source that's named **IceCream** and that starts with the data in this table:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	300

Create a collection with sample records

To create a collection with this data:

1. Insert a **Button** control.

2. Set button control's **OnSelect** property to the below formula:

```
Power Fx
```

```
ClearCollect( IceCream,
              { ID: 1, Flavor: "Chocolate",  Quantity: 100 },
              { ID: 2, Flavor: "Vanilla",    Quantity: 200 },
              { ID: 3, Flavor: "Strawberry", Quantity: 300 }
)
```

3. Select the button while holding down the Alt key:

Remove sample records from collection using a formula

Expand table

Formula	Description	Result
Remove(IceCream, LookUp(IceCream, Flavor="Chocolate"))	Removes the **Chocolate** record from the data source.	 The **IceCream** data source has been modified.
Remove(IceCream, LookUp(IceCream, Flavor="Chocolate"), LookUp(IceCream, Flavor="Strawberry"))	Removes two records from the data source.	 The **IceCream** data source has been modified.
RemoveIf(IceCream, Quantity > 150)	Removes records that have a **Quantity** that's greater than **150**.	 The **IceCream** data source has been modified.
RemoveIf(IceCream, Quantity > 150, Left(Flavor, 1) = "S")	Removes records that have a **Quantity** that's greater than 150 and **Flavor** starts with an **S**.	 The **IceCream** data source has been modified.

Formula	Description	Result
RemoveIf(IceCream, true)	Removes all records from the data source.	ID Flavor Quantity The **IceCream** data source has been modified.

Examples - remove button outside a gallery

In this example, you'll use a **Gallery** control to list the records in a table. And then use the **Remove** function to selectively remove an item.

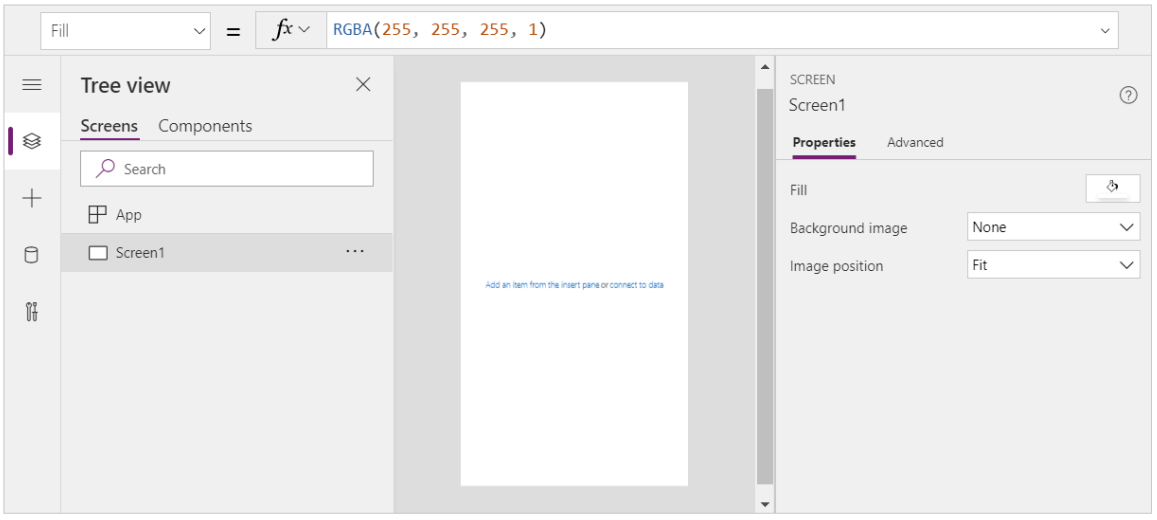
Prepare for sample data

This example uses the **Contacts** table in Microsoft Dataverse available with the *sample apps and data*. You can deploy *sample apps and data* when you create an environment. You can also use any other data source instead.

Remove button outside a gallery

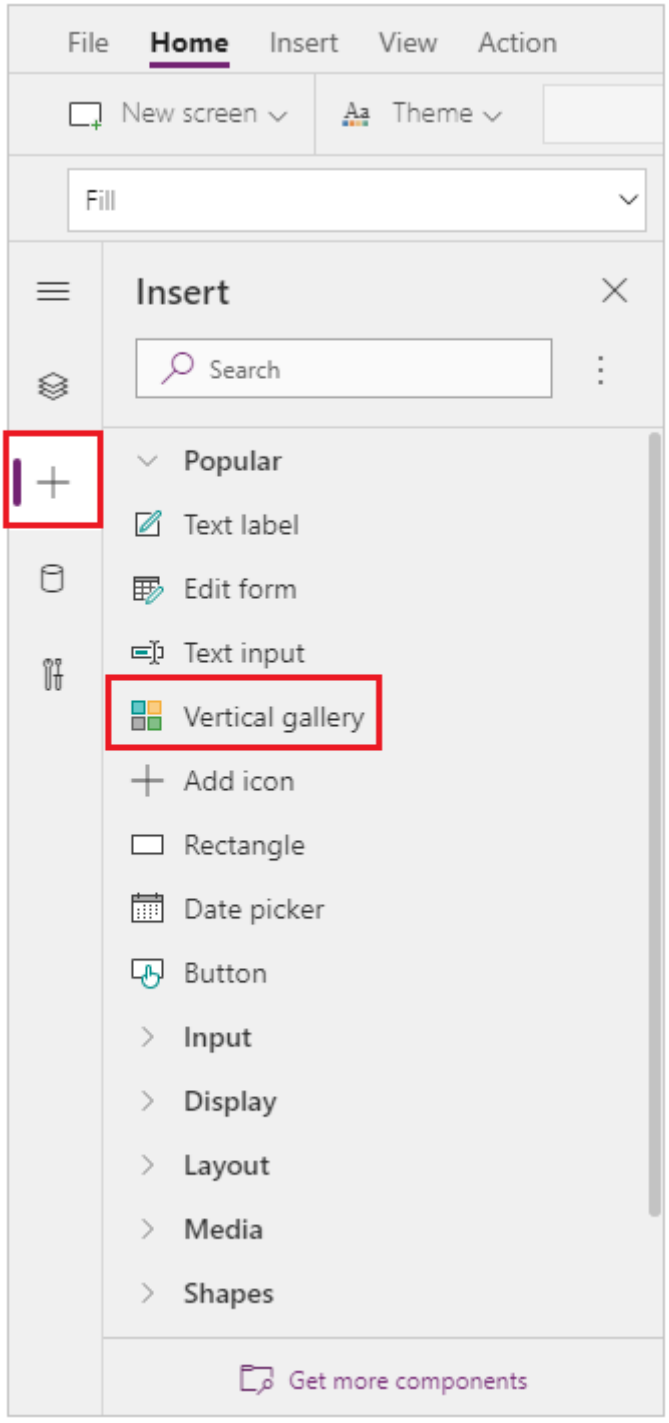
In this example, you'll remove an item by using a *button* that is outside the gallery.

1. Create a new blank canvas app using a Phone layout.

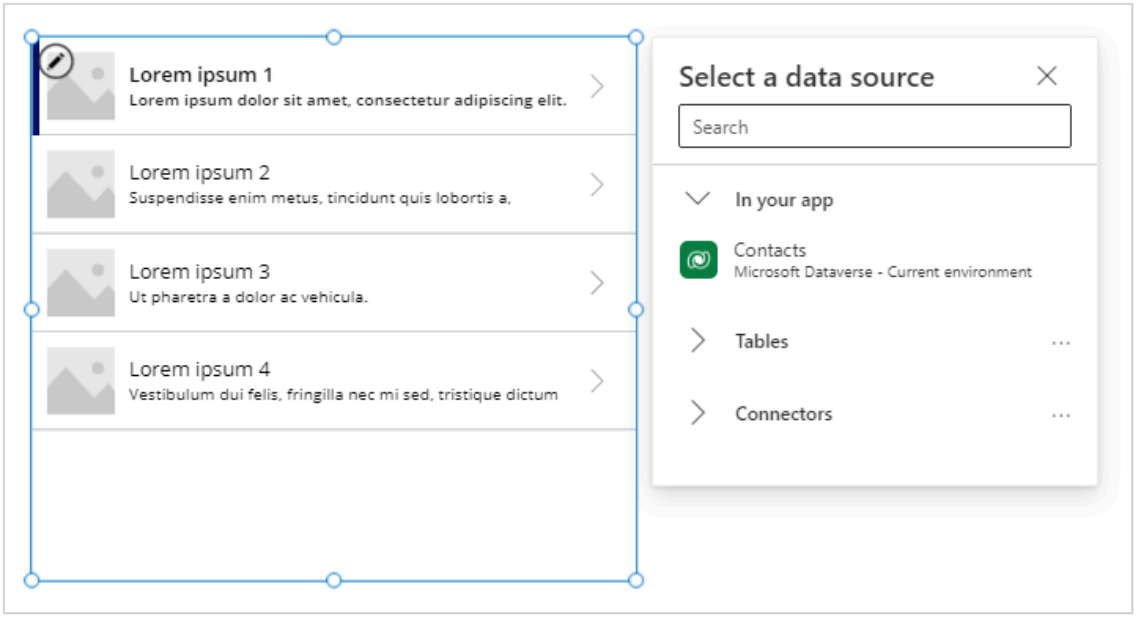


2. Select the **Insert** from the left pane.

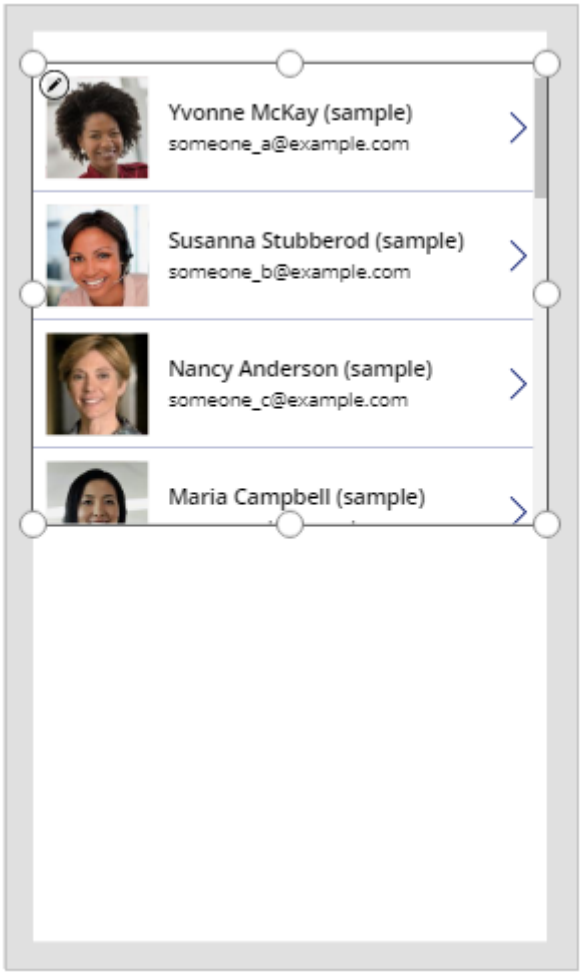
3. Select **Vertical gallery**.
   A **Gallery** control is be added to your screen.



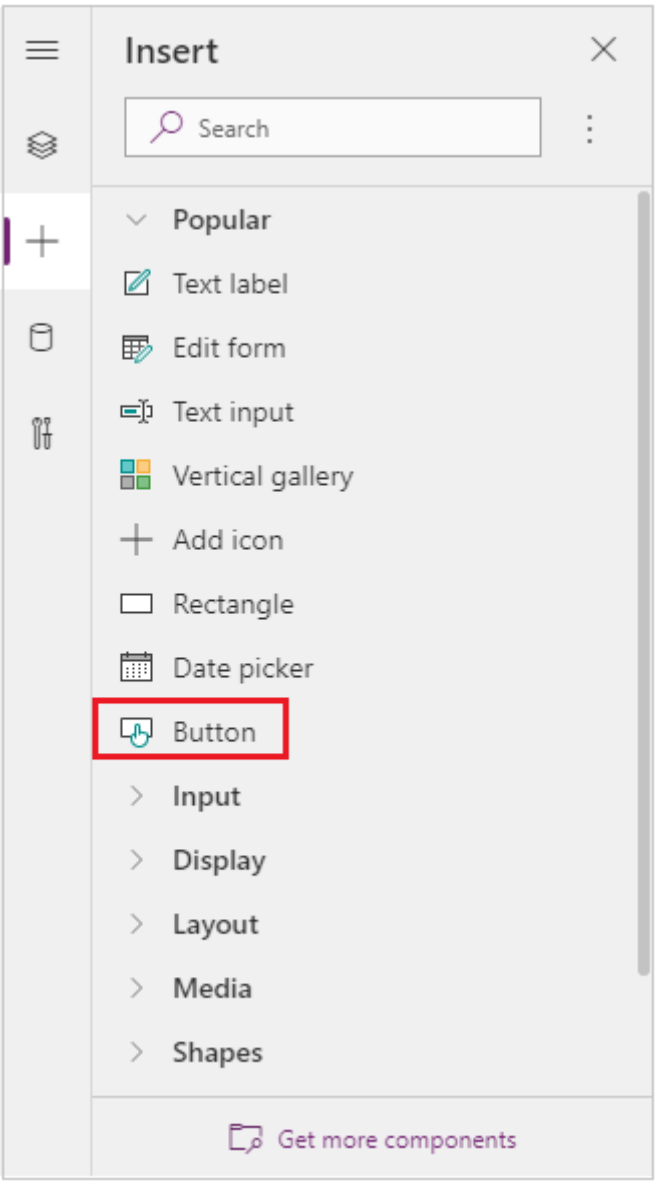
4. You're prompted to select a data source where you can select a data source from the available data sources.
   For example, select the **Contacts** table to use *sample data*:



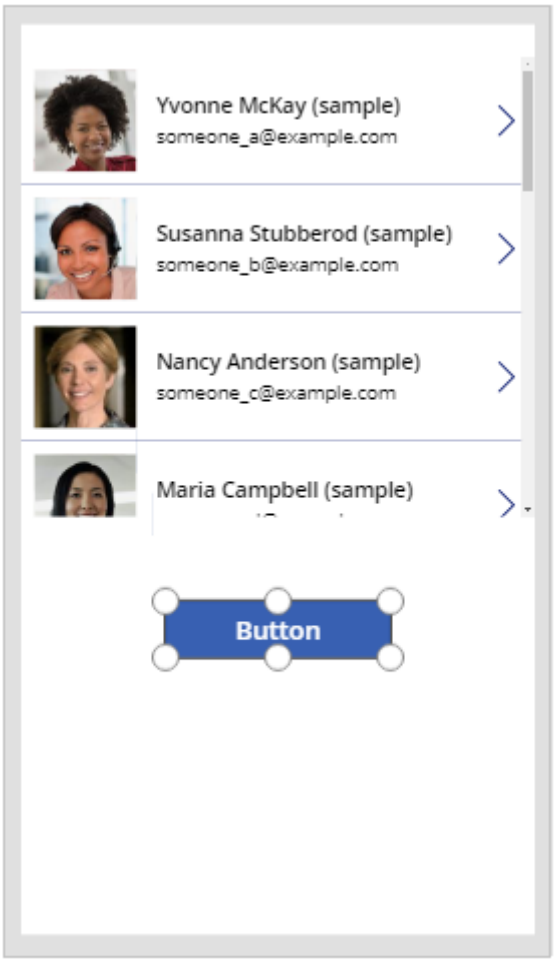
The gallery shows items from this table:



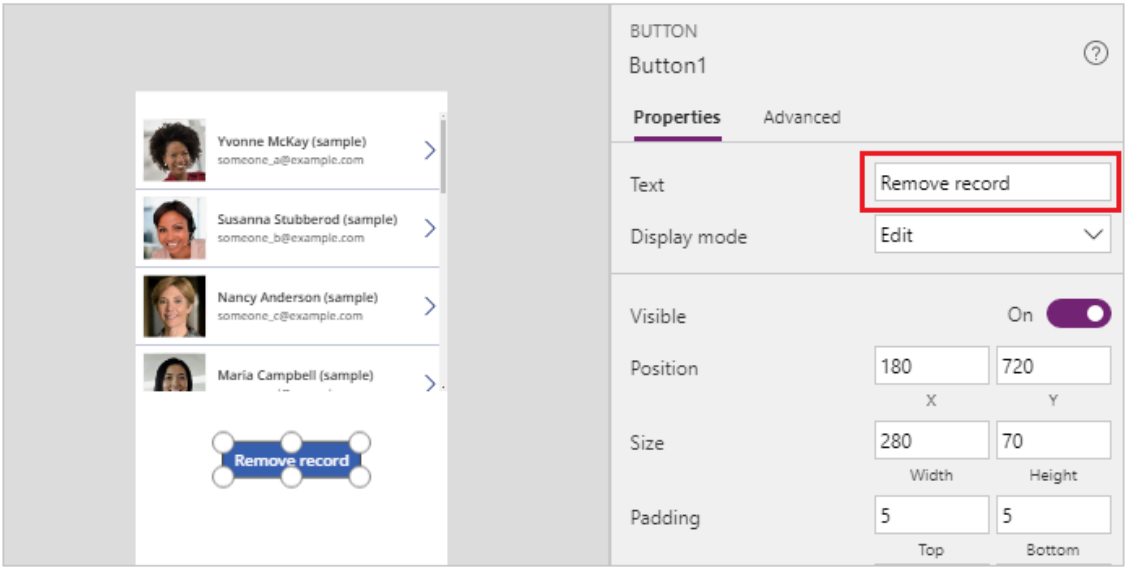
5. Insert a **Button** control from left pane:



6. Move the added button below the gallery items:

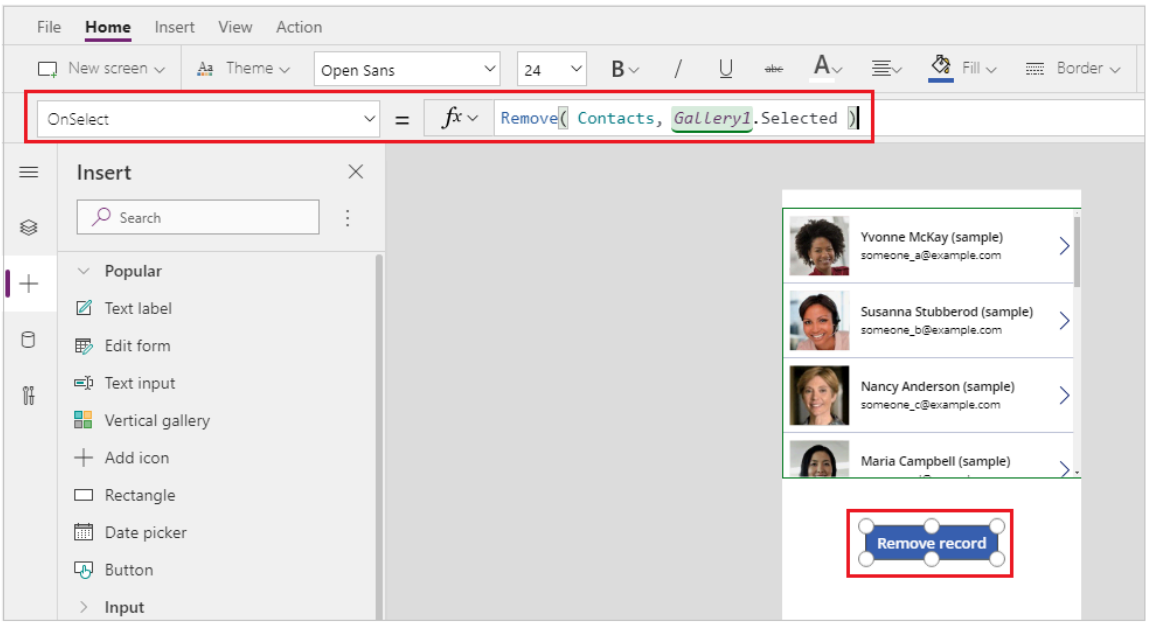


7. Update button text property to *Remove record*. You can also use text of your choice:



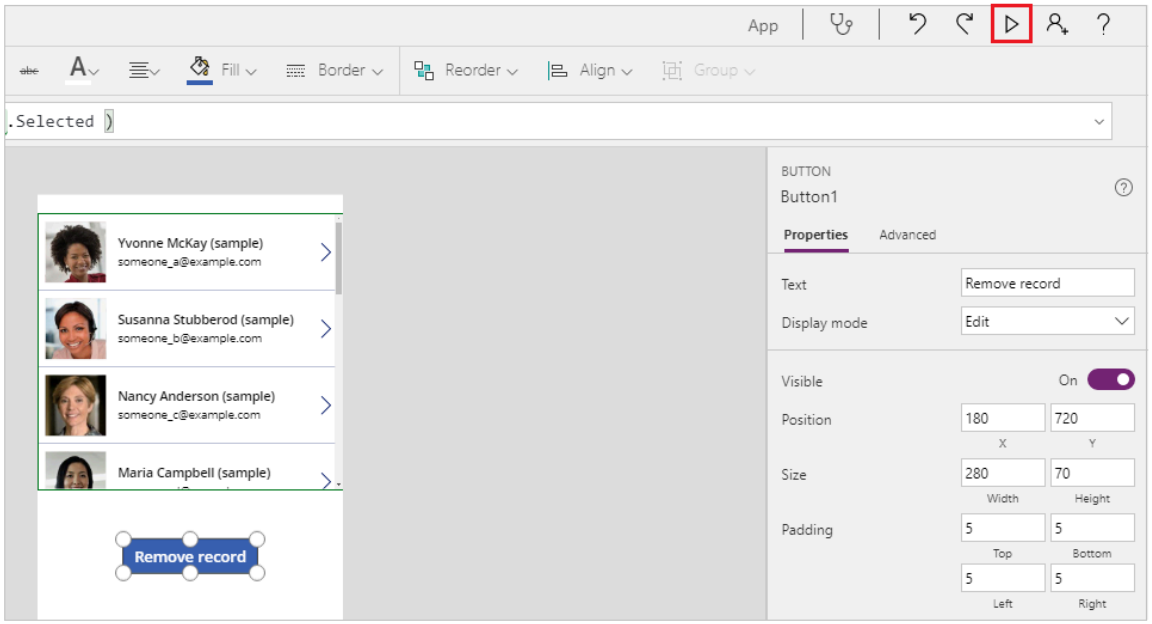
8. Set the **OnSelect** property for this button control to the following formula:

```
Power Fx

Remove( Contacts, Gallery1.Selected )
```

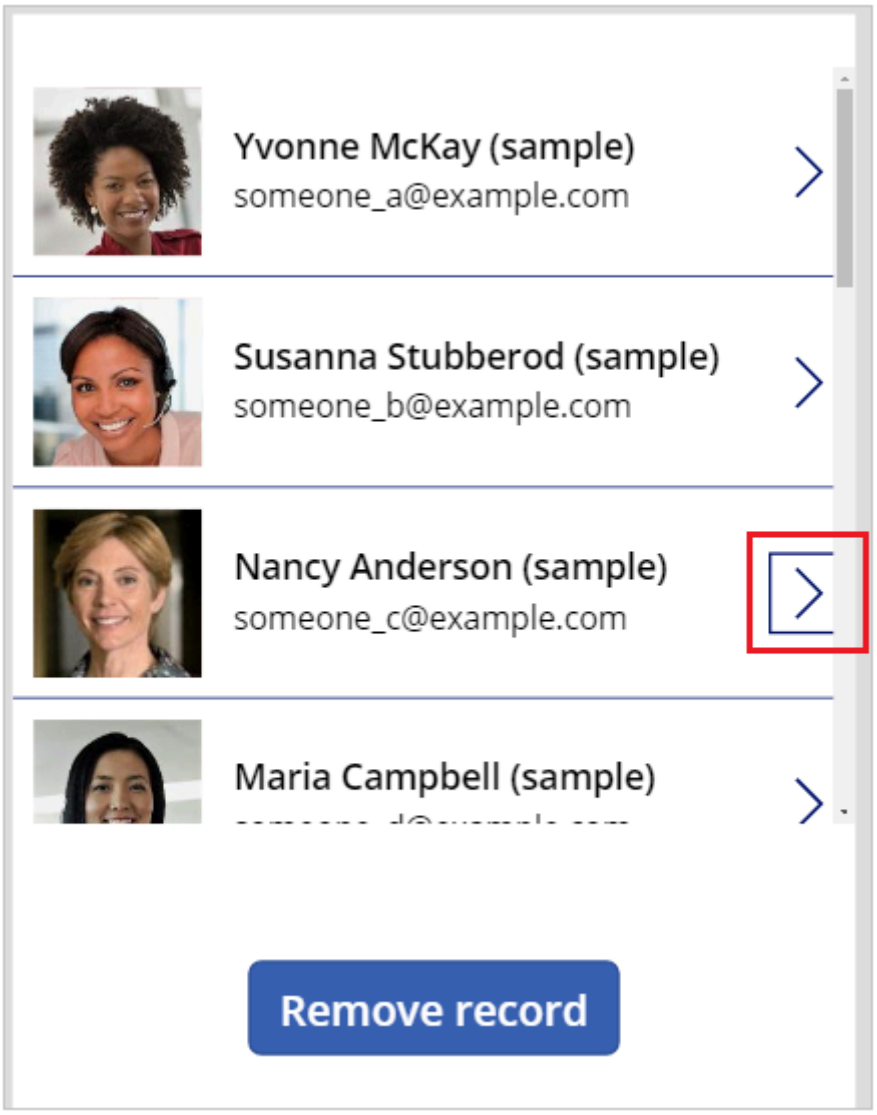


The gallery control makes the currently selected record available using **Selected** property. **Remove** function refers to this selected record to remove it.

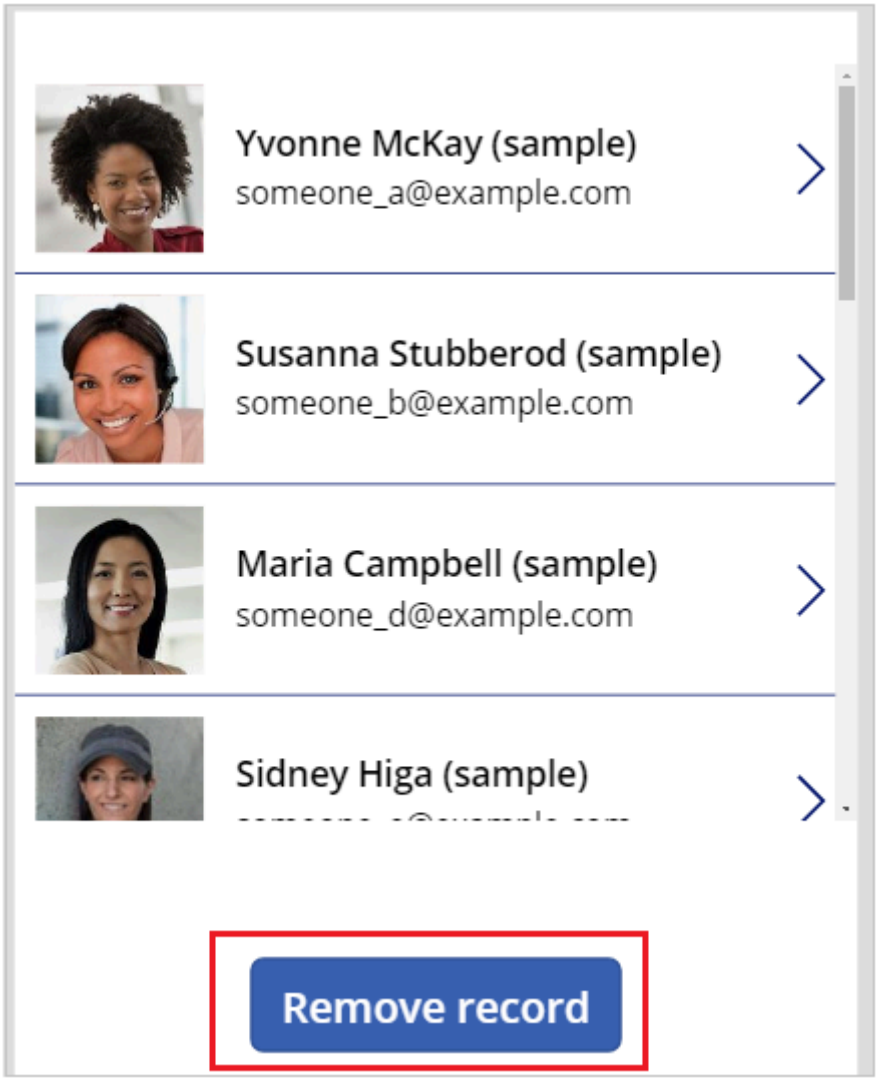
9. Preview the app using the *Play* button on the top right, or press *F5* on keyboard:



10. Select a record to remove, such as *Nancy*'s record in this example:



11. Select **Remove record**:



Selecting the button removes the selected record (in this example, Nancy's record).

12. Close the app preview.

> **Tip**
>
> You can also use alternate behavior with ***Alt key*** instead of using the app preview with *Play* button or *F5*.

Examples - trash can icon inside a gallery

In this example, you'll remove an item by using an *icon* placed inside the gallery.

Create a collection with sample data

If you already have prepared sample data, skip this step and move to Trash can icon inside a gallery.

1. Add a **Button** control to your screen.

2. Set the **OnSelect** property to the following formula:

```
Power Fx
```

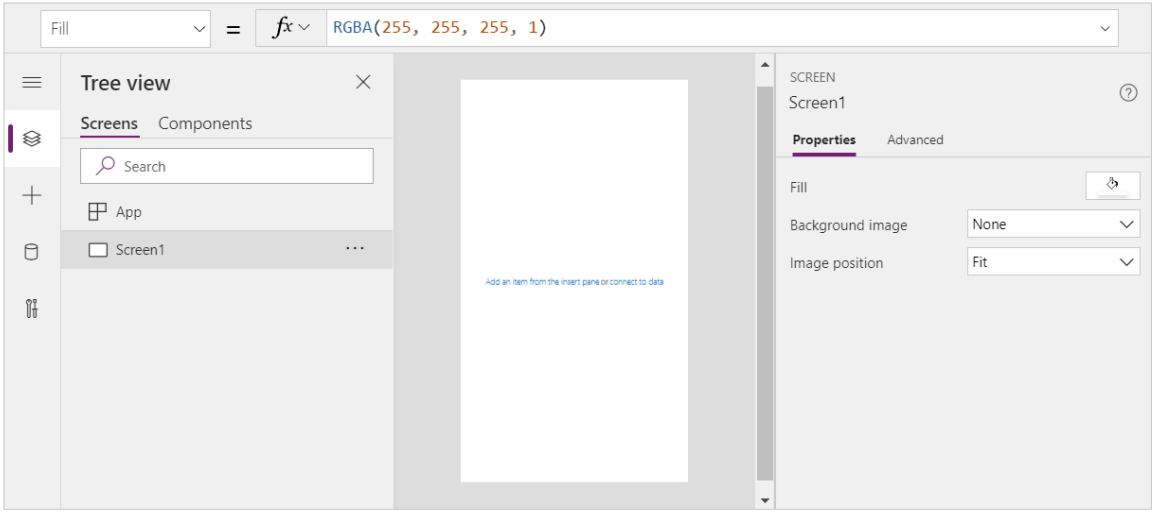
```
ClearCollect( SampleContacts,
      { 'Full Name': "Yvonne McKay (sample)",      'Primary Email':
"someone_a@example.com" },
      { 'Full Name': "Susanna Stubberod (sample)", 'Primary Email':
"someone_b@example.com" },
      { 'Full Name': "Nancy Anderson (sample)",    'Primary Email':
"someone_c@example.com" },
      { 'Full Name': "Maria Campbell (sample)",    'Primary Email':
"someone_d@example.com" },
      { 'Full Name': "Robert Lyon (sample)",       'Primary Email':
"someone_e@example.com" },
      { 'Full Name': "Paul Cannon (sample)",       'Primary Email':
"someone_f@example.com" },
      { 'Full Name': "Rene Valdes (sample)",       'Primary Email':
"someone_g@example.com" }
)
```

3. Select the button while holding down the Alt key.

Sample collection is created that you can use in the following example.

Trash can icon inside a gallery

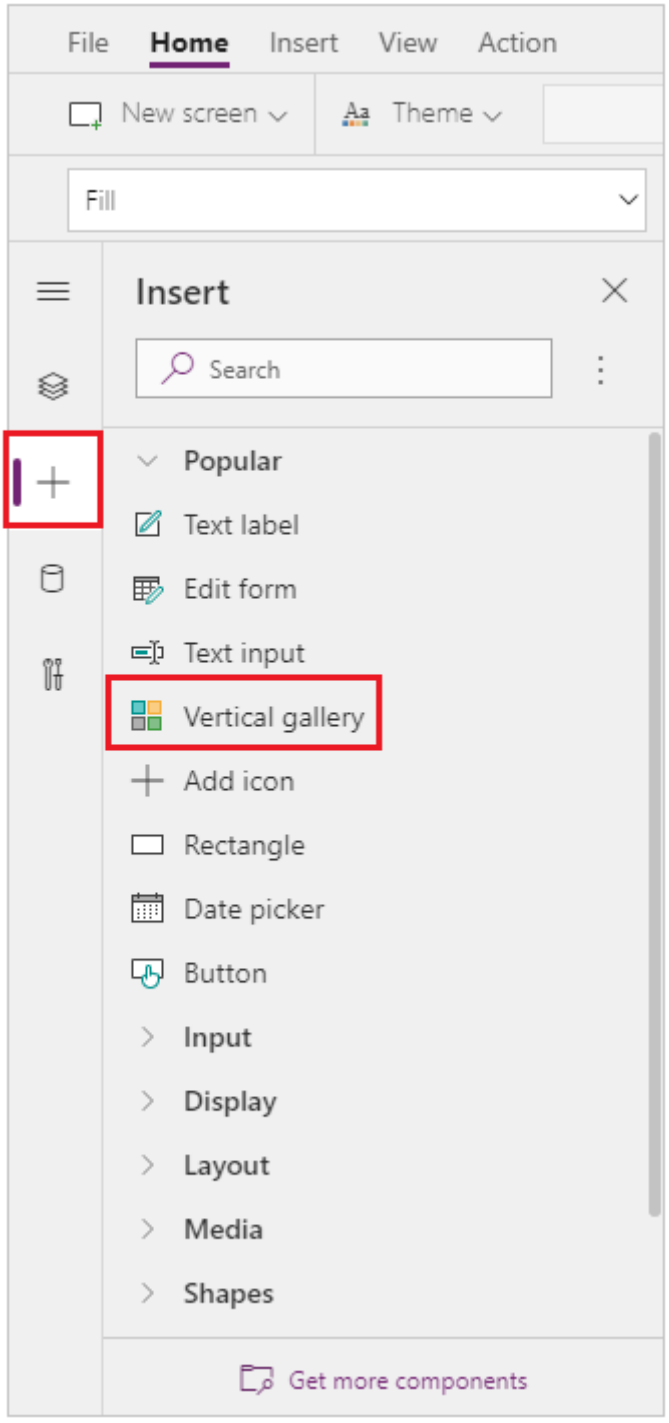
1. Create a new blank canvas app using a Phone layout.



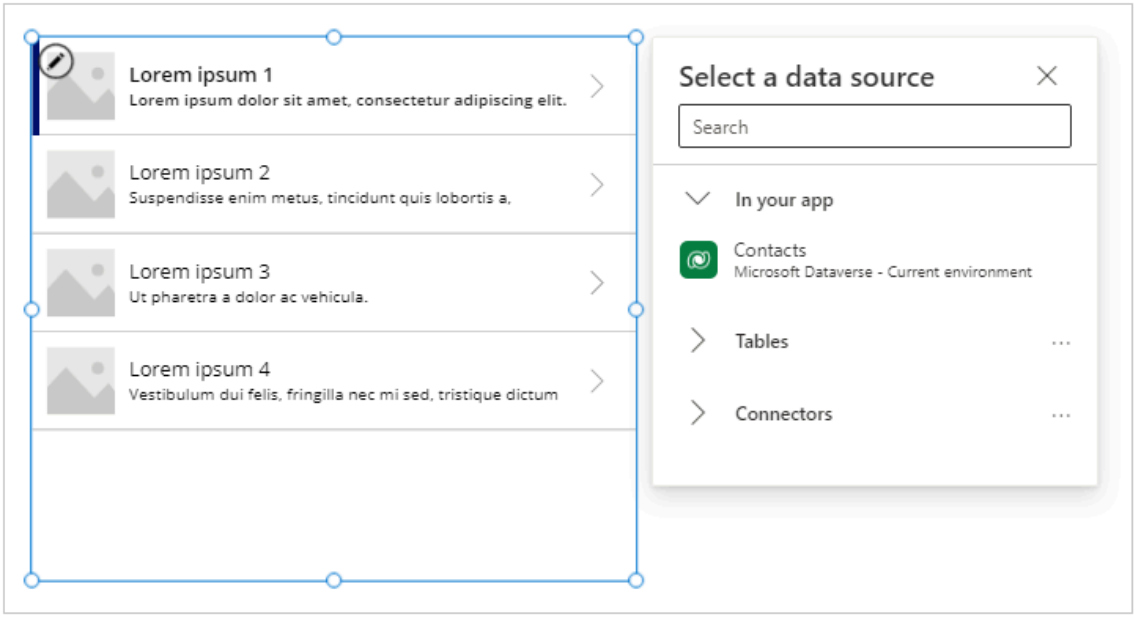
2. Select the **Insert** from the left pane.

3. Select **Vertical gallery**.

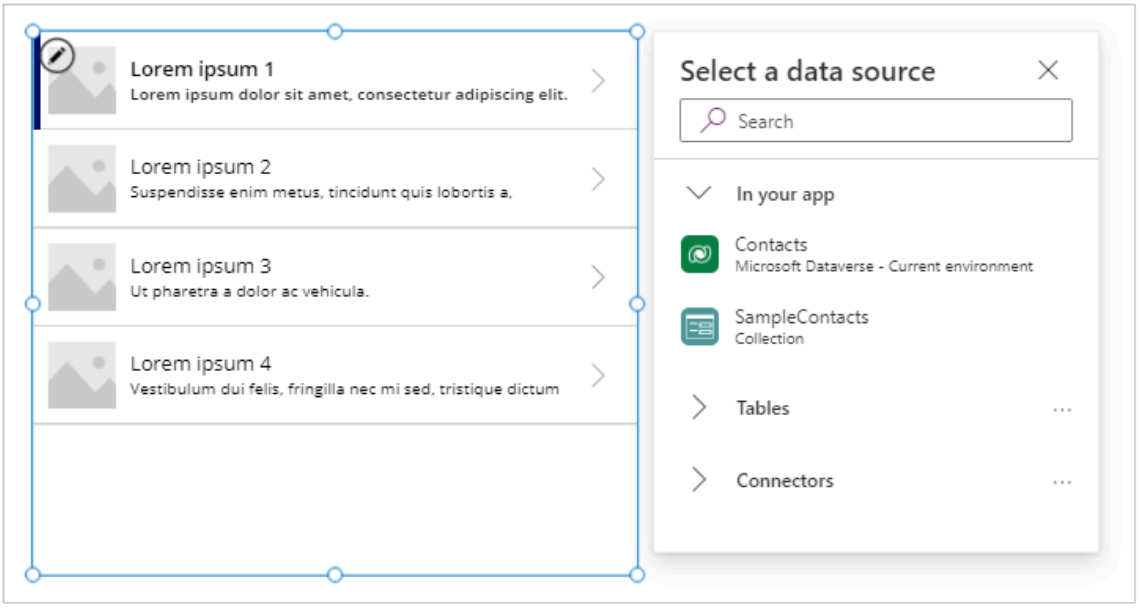
   A **Gallery** control is be added to your screen.



4. You're prompted to select a data source where you can select a data source from the available data sources.
   For example, select the **Contacts** table to use *sample data*:

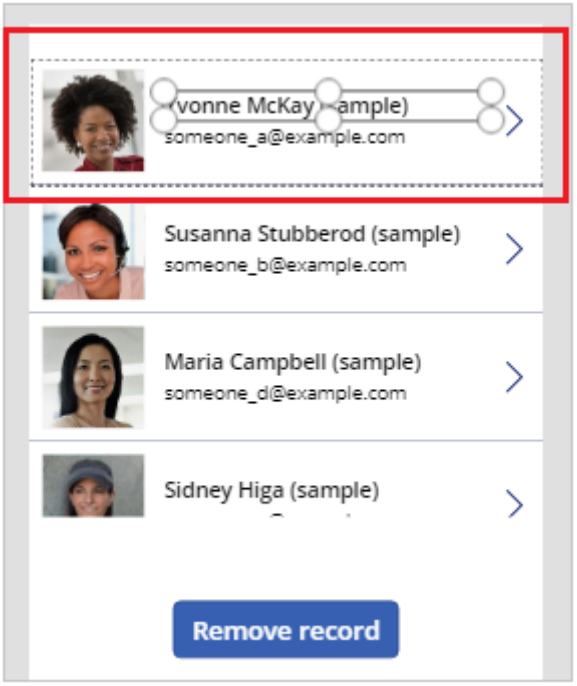


If you created a collection, select your collection instead:

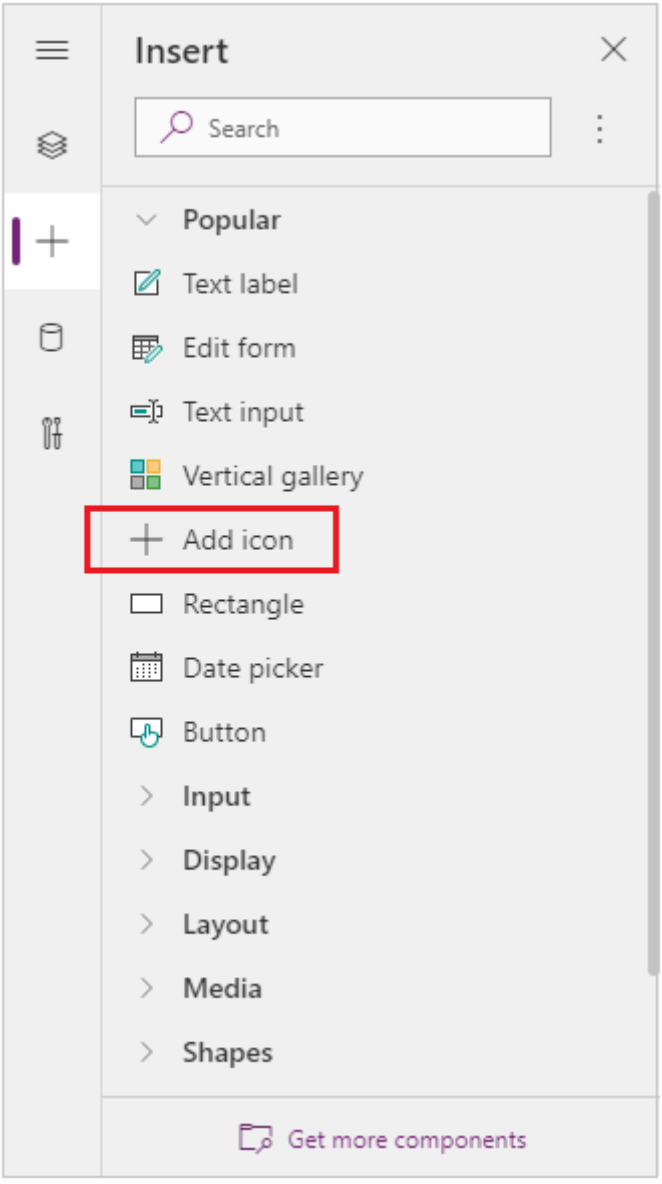


5. Select a control within the top item in the gallery.

   To ensure next step inserts item into gallery's template and not outside the gallery, ensure you follow this step before moving to the next step.

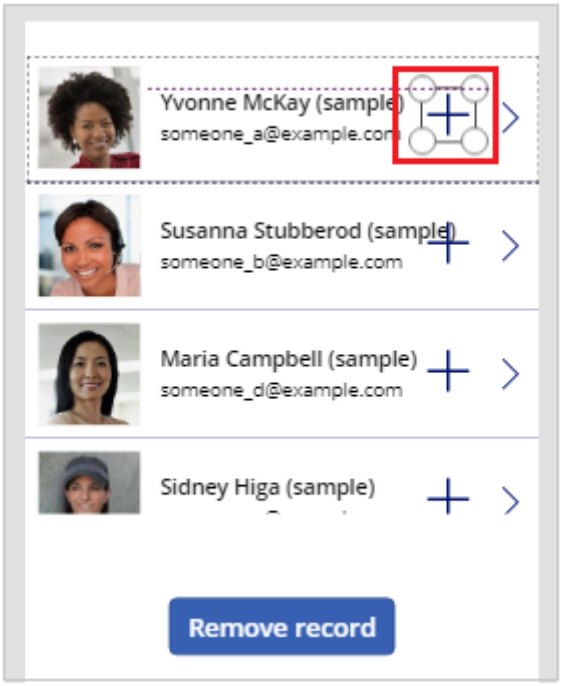


6. Select **Add icon** from left pane.



> ⓘ Note
>
> **Add icon** inserts a + icon on the left side of the gallery, replicated for each item in the gallery.

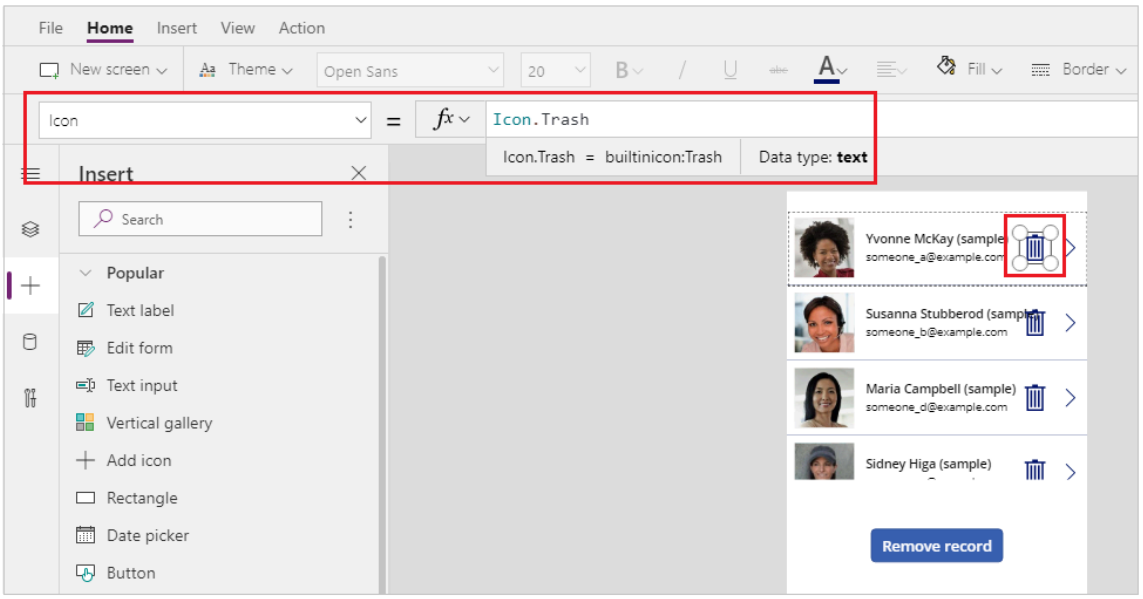
7. In the top item, move the icon to the right side of the screen.



8. Select the **Icon** property for icon and set it to the following formula to update the icon image as trash icon:

```
Power Fx

Icon.Trash
```

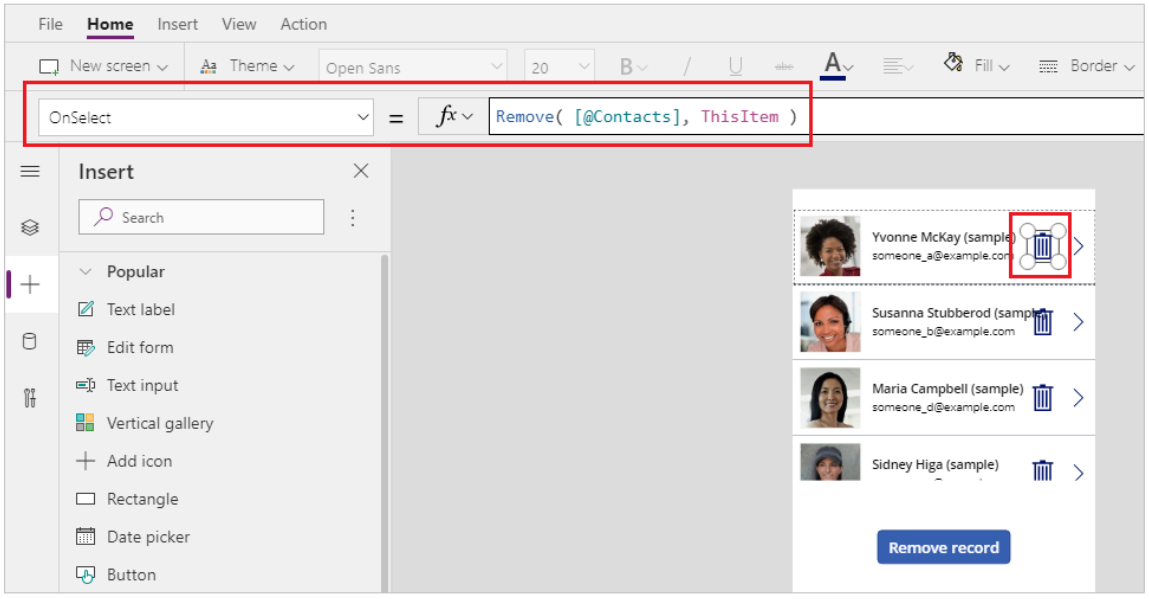
> ⓘ Note
>
> The **Icon.** prefix is only shown when you're actively editing the formula.



9. Set the **OnSelect** property to the following formula:

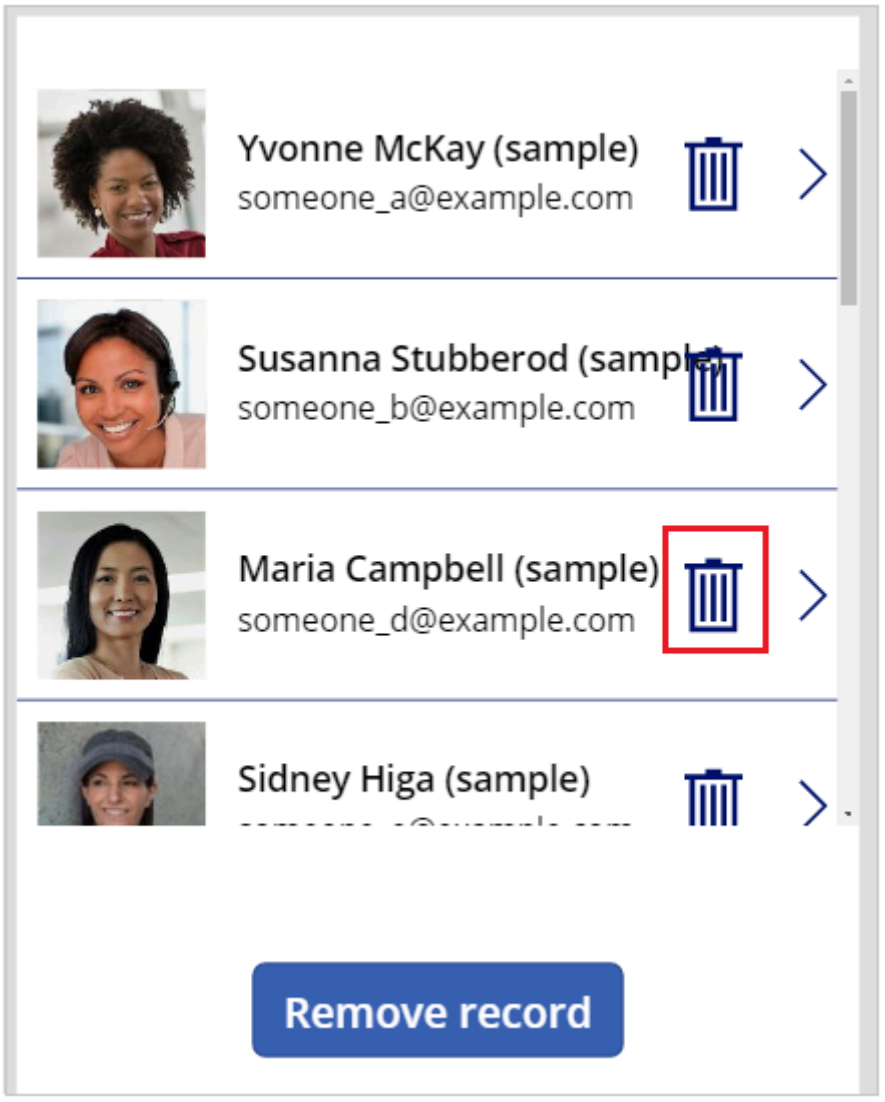
```
Power Fx

Remove( [@Contacts], ThisItem )
```

> **Note**
>
> You must use global disambiguation operator [@...] in this example with sample data that uses the *Contacts* table to avoid conflict with a *One-to-Many* relationship. If you use data sources such as a list or a SQL Server table, using *global disambiguation operator* is not required.

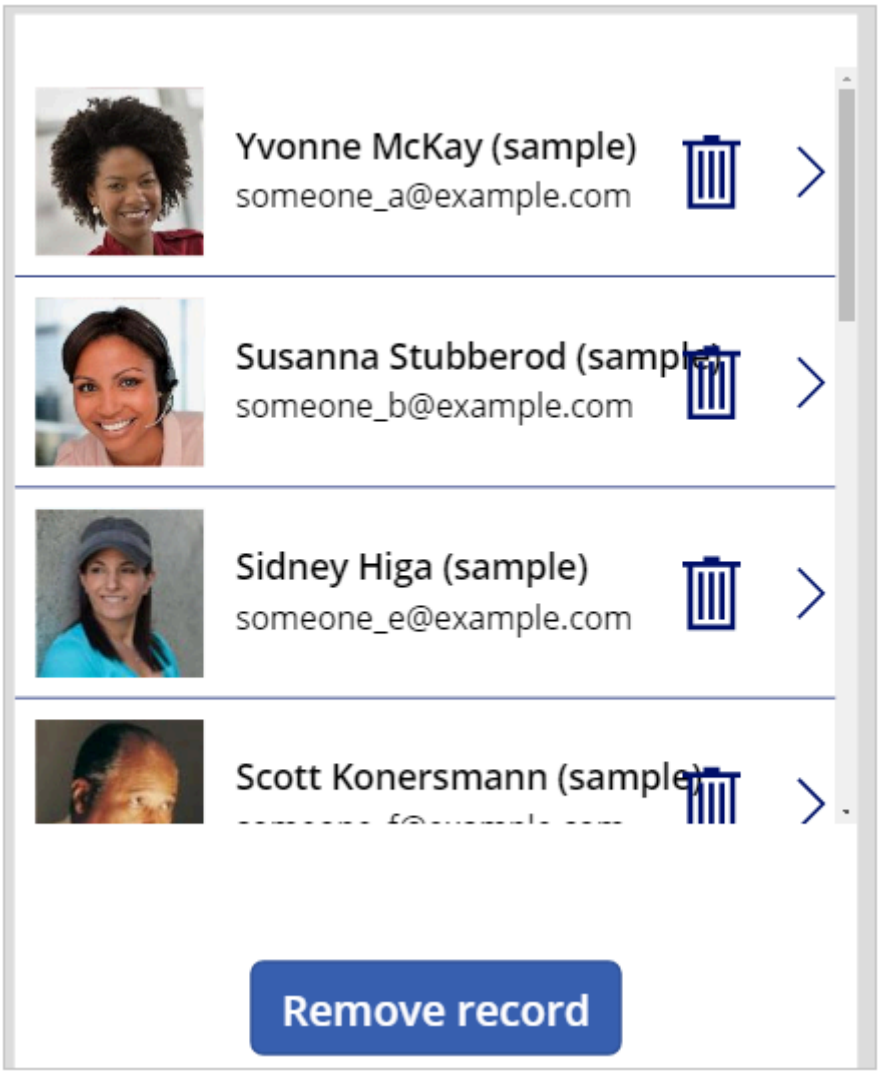


10. Preview the app using the *Play* button on the top right, or press *F5* on keyboard.

11. Select the trash icon next to a record, for example *Maria*'s:



The record is deleted:



12. Close the app preview.

Feedback

Was this page helpful? Yes No

Provide product feedback

AddColumns, DropColumns, RenameColumns, and ShowColumns functions

06/21/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Shapes a table or record by adding, dropping, renaming, and selecting its columns.

The ForAll function can also be used to shape a table, by returning a table of new records created from existing columns.

Overview

These functions shape a table or record by adjusting its columns:

- Reduce a table or record that contains multiple columns down to a single column for use with single-column functions, such as **Lower** or **Sqrt**.
- Add a calculated column to a table or record (for example, a **Total Price** column that shows the results of multiplying **Quantity** by **Unit Price**).
- Rename a column to something more meaningful, for display to users or for use in formulas.

A table is a value in Power Apps, just like a string or a number. You can specify a table as an argument in a formula, and functions can return a table as a result.

> **Note**
>
> The functions that this topic describes don't modify the original table. Instead, they take that table as an argument and return a new table with a transform applied. For more information, see working with tables.

You can't modify the columns of a data source by using these functions. You must modify the data at its source. You can add columns to a collection with the **Collect** function. For more information, see working with data sources.

Description

The **AddColumns** function adds a column to a table or record, and a formula defines the values in that column. Existing columns remain unmodified.

The formula is evaluated for the provided record, or for each record of the provided table.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

The **DropColumns** function excludes columns from a table or record. All other columns remain unmodified. **DropColumns** excludes columns, and **ShowColumns** includes columns.

Use the **RenameColumns** function to rename one or more columns of a table or record by providing at least one argument pair that specifies the name of a column that the table or record contains (the old name, which you want to replace) and the name of a column that the table or record doesn't contain (the new name, which you want to use). The old name must already exist in the table or record, and the new name must not exist. Each column name may appear only once in the argument list as either an old column name or a new column name. To rename a column to an existing column name, first drop the existing column with **DropColumns**, or rename the existing column out of the way by nesting one **RenameColumns** function within another.

The **ShowColumns** function includes columns of a table or record and drops all other columns. You can use **ShowColumns** to create a single-column table or record from a multi-column table or record. **ShowColumns** includes columns, and **DropColumns** excludes columns.

For all these functions, the result is a new table or record with the transform applied. The original table or record isn't modified. You can't modify an existing table or record with a formula. SharePoint, Microsoft Dataverse, SQL Server, and other data sources provide tools for modifying the columns of lists and tables, which are often referred to as the schema. The functions in this topic only transform an input table or record, without modifying the original, into an output table or record for further use.

The arguments to these functions support delegation. For example, a **Filter** function used as an argument to pull in related records searches through all listings, even if the '**[dbo].[AllListings]**' data source contains a million rows:

Power Fx

```
AddColumns( RealEstateAgents,
   Listings,
```

```
    Filter(  '[dbo].[AllListings]', ListingAgentName = AgentName )
)
```

However, the output of these functions is subject to the non-delegation record limit. In this example, only 500 records are returned even if the **RealEstateAgents** data source has 501 or more records.

If you use **AddColumns** in this manner, **Filter** must make separate calls to the data source for each of those first records in **RealEstateAgents**, which causes lots of network chatter. If **[dbo](.[AllListings]** is small enough and doesn't change often. You can call the **Collect** function in OnStart to cache the data source in your app when it starts. As an alternative, you could restructure your app so that you pull in the related records only when the user asks for them.

> **Note**
>
> In Power Apps prior to version 3.24042, column names were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

Syntax

AddColumns(*TableOrRecord*, *ColumnName1*, *Formula1* [, *ColumnName2*, *Formula2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to add.
- *Formula(s)* - Required. Formulas to evaluate for each record. The result is added as the value of the corresponding new column. You can reference other columns of the table or record in this formula.

DropColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to drop.

RenameColumns(*TableOrRecord*, *OldColumnName1*, *NewColumnName1* [, *OldColumnName2*, *NewColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.

- *OldColumnName(s)* - Required. Names of the columns to rename from the original table or record. This element appears first in the argument pair (or first in each argument pair if the formula includes more than one pair).
- *NewColumnName(s)* - Required. Replacement names. This element appears last in the argument pair (or last in each argument pair if the formula includes more than one pair).

ShowColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to include.

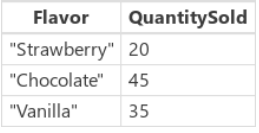
Examples

The examples in this section use the **IceCreamSales** data source, which contains the data in this table:

Flavor	UnitPrice	QuantitySold
"Strawberry"	1.99	20
"Chocolate"	2.99	45
"Vanilla"	1.50	35

None of these examples modify the **IceCreamSales** data source. Each function transforms the value of the data source as a table and returns that value as the result.

Expand table

Formula	Description	Result
AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold)	Adds a **Revenue** column to the result. For each record, **UnitPrice * QuantitySold** is evaluated, and the result is placed in the new column.	
DropColumns(IceCreamSales, UnitPrice)	Excludes the **UnitPrice** column from the result. Use this function to exclude columns, and use **ShowColumns** to include them.	
ShowColumns(IceCreamSales, Flavor)	Includes only the **Flavor** column in the result. Use this function include columns, and use **DropColumns** to exclude them.	
RenameColumns(IceCreamSales, UnitPrice, Price)	Renames the **UnitPrice** column in the result.	

Formula	Description	Result
RenameColumns(IceCreamSales, UnitPrice, Price, QuantitySold, Number)	Renames the **UnitPrice** and **QuantitySold** columns in the result.	<table><tr><th>Flavor</th><th>Price</th><th>Number</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>20</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>45</td></tr><tr><td>"Vanilla"</td><td>1.50</td><td>35</td></tr></table>
DropColumns(RenameColumns(AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold), UnitPrice, Price), Quantity)	Performs the following table transforms in order, starting from the inside of the formula: 1. Adds a **Revenue** column based on the per record calculation of **UnitPrice * Quantity**. 2. Renames **UnitPrice** to **Price**. 3. Excludes the **Quantity** column. Order is important, for example we can't calculate with **UnitPrice** after it's renamed.	<table><tr><th>Flavor</th><th>Price</th><th>Revenue</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>39.80</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>134.55</td></tr><tr><td>"Vanilla"</td><td>1.49</td><td>52.15</td></tr></table>

Step by step

Let's try some of the examples from earlier in this article.

1. Create a collection by adding a **Button** control and setting its **OnSelect** property to this formula:

   Power Fx

   ```
   ClearCollect( IceCreamSales,
      Table(
          { Flavor: "Strawberry", UnitPrice: 1.99, QuantitySold: 20 },
          { Flavor: "Chocolate", UnitPrice: 2.99, QuantitySold: 45 },
          { Flavor: "Vanilla", UnitPrice: 1.50, QuantitySold: 35 }
      )
   )
   ```

2. Run the formula by selecting the button while holding down the Alt key.

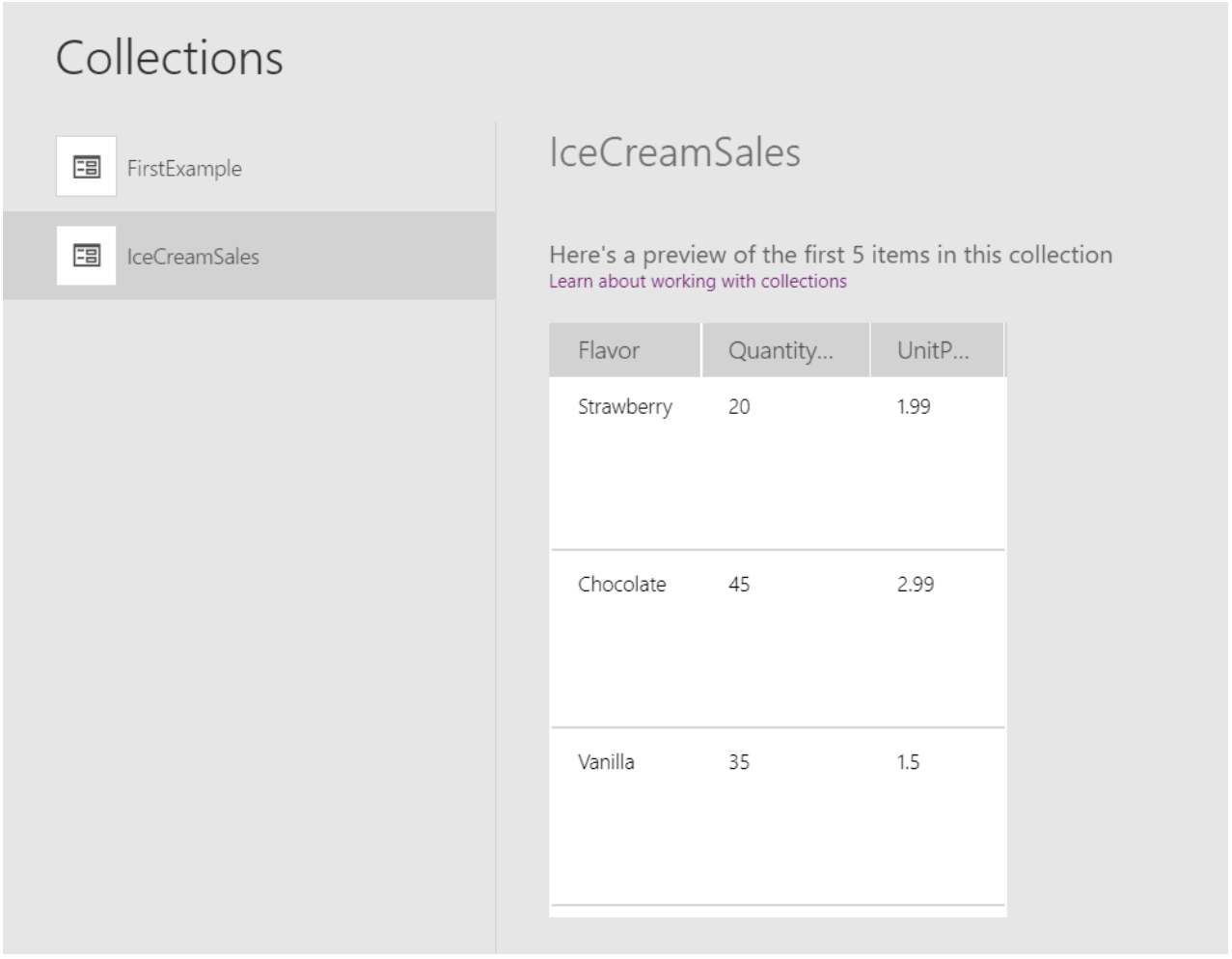
3. Add a second **Button** control, set its **OnSelect** property to this formula, and then run it:

   Power Fx

   ```
   ClearCollect( FirstExample,
      AddColumns( IceCreamSales, Revenue, UnitPrice * QuantitySold )
   )
   ```

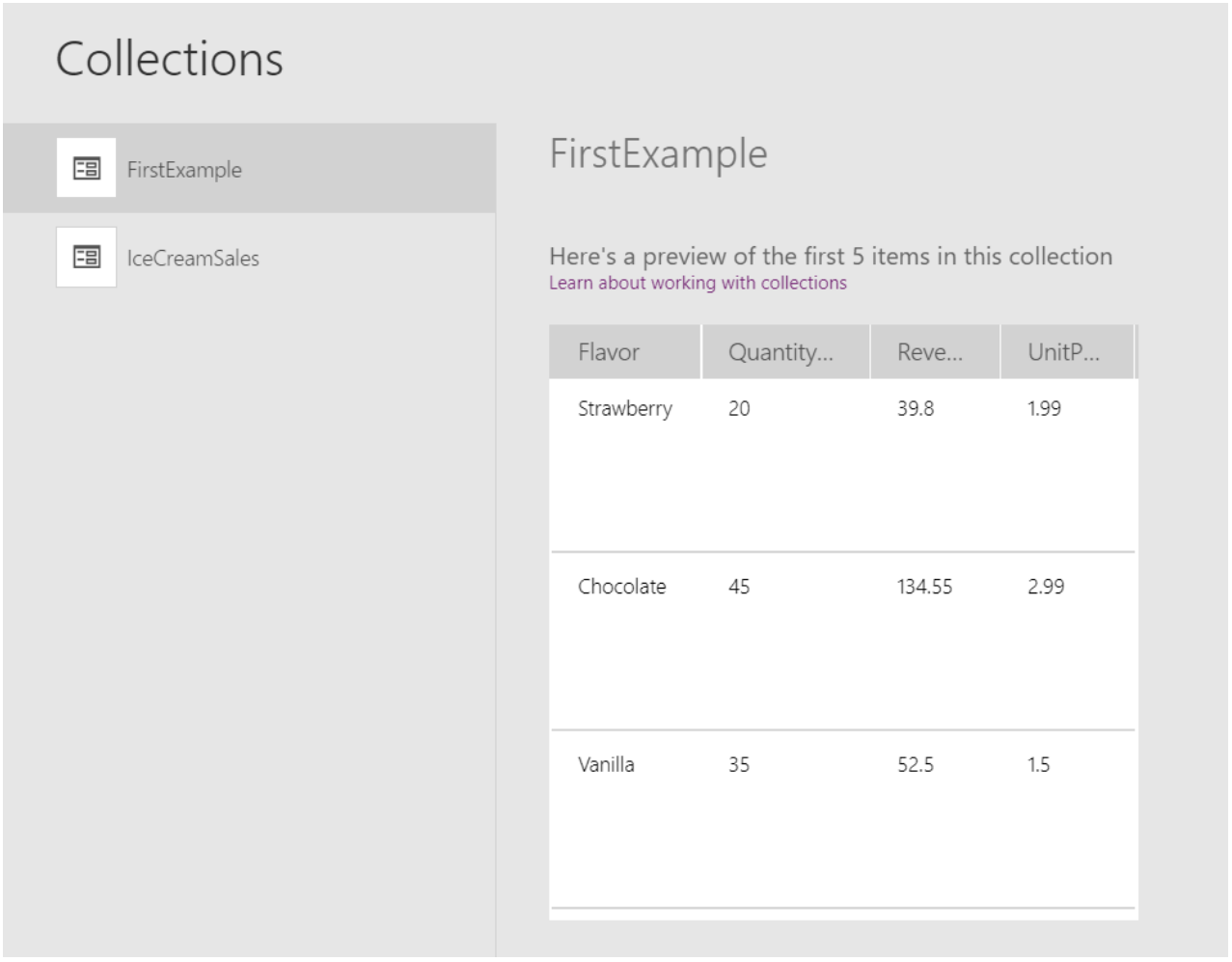
4. On the **File** menu, select **Collections**, and then select **IceCreamSales** to show that collection.

As this graphic shows, the second formula didn't modify this collection. The **AddColumns** function used **IceCreamSales** as a read-only argument; the function didn't modify the table to which that argument refers.



5. Select **FirstExample**.

   As this graphic shows, the second formula returned a new table with the added column. The **ClearCollect** function captured the new table in the **FirstExample** collection, adding something to the original table as it flowed through the function without modifying the source:



Map columns in a component

See Map columns.

Replace and Substitute functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Replace a portion of a string of text with another string.

Description

The **Replace** function identifies the text to replace by starting position and length.

The **Substitute** function identifies the text to replace by matching a string. If more than one match is found, you can replace all of them or specify one to replace.

If you pass a single string, the return value is the modified string. If you pass a single-column table that contains strings, the return value is a single-column table with a **Value** column of modified strings. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

Replace(*String*, *StartingPosition*, *NumberOfCharacters*, *NewString*)

- *String* - Required. The string to operate on.
- *StartingPosition* - Required. Character position to start the replacement. The first character of *String* is at position 1.
- *NumberOfCharacters* - Required. The number of characters to replace in *String*.
- *NewString* - Required. The replacement string. The number of characters in this argument can differ from the *NumberOfCharacters* argument.

Substitute(*String*, *OldString*, *NewString* [, *InstanceNumber*])

- *String* - Required. The string to operate on.
- *OldString* - Required. The string to replace.
- *NewString* - Required. The replacement string. *OldString* and *NewString* can have different lengths.
- *InstanceNumber* - Optional. Use this argument to specify which instance of *OldString* to replace if *String* contains more than one instance. If you don't specify this argument, all instances will be replaced.

Replace(*SingleColumnTable*, *StartingPosition*, *NumberOfCharacters*, *NewString*)

- *SingleColumnTable* - Required. A single-column table of strings to operate on.
- *StartingPosition* - Required. Character position to start the replacement. The first character of each string in the table is at position 1.
- *NumberOfCharacters* - Required. The number of characters to replace in each string.
- *NewString* - Required. The replacement string. The number of characters in this argument can differ from the *NumberOfCharacters* argument.

Substitute(*SingleColumnTable*, *OldString*, *NewString* [, *InstanceNumber*])

- *SingleColumnTable* - Required. A single-column table of strings to operate on.
- *OldString* - Required. The string to replace.
- *NewString* - Required. The replacement string. *OldString* and *NewString* can have different lengths.
- *InstanceNumber* - Optional. Use this argument to specify which instance of *OldString* to replace if *String* contains more than one instance. If you don't specify this argument, all instances will be replaced.

Examples

Expand table

Formula	Description	Result
Replace("abcdefghijk", 6, 5, "*")	Replaces five characters in "abcdefghijk" with a single "*" character, starting with the sixth character ("f").	"abcde*k"
Replace("2019", 3, 2, "20")	Replaces the last two characters of "2019" with "20".	"2020"
Replace("123456", 1, 3, "_")	Replaces the first three characters of "123456" with a single "_" character.	"_456"
Substitute("Sales Data", "Sales", "Cost")	Substitutes the string "Cost" for "Sales".	"Cost Data"
Substitute("Quarter 1, 2018", "1", "2", 1)	Substitutes only the first instance of "1" with "2" because the fourth argument	"Quarter 2, 2018"

Formula	Description	Result
	(*InstanceNumber*) is provided with a 1.	
Substitute("Quarter 1, 2011", "1", "2", 3)	Substitutes only the third instance of "1" with "2" because the fourth argument (*InstanceNumber*) is provided with a 3.	"Quarter 1, 2012"
Substitute("Quarter 1, 2011", "1", "2")	Substitutes all instances of "1" with "2" because the fourth argument (*InstanceNumber*) isn't provided.	"Quarter 2, 2022"
Replace(["Quarter 1, 2018", "Quarter 2, 2011", "Quarter 4, 2019"], 9, 1, "3")	Replaces the ninth character in each record of the single-column table with "3".	A single-column table with a `Value` column containing the following values: ["Quarter 3, 2018", "Quarter 3, 2011", "Quarter 3, 2019"]
Substitute(["Qtr 1, 2018", "Quarter 1, 2011", "Q1, 2019"], "1", "3", 1)	Because the fourth argument (*InstanceNumber*) is provided with a value of 1, substitutes only the first instance of "1" in each record of the single-column table with "3".	A single-column table with a `Value` column containing the following values: ["Qtr 3, 2018", "Quarter 3, 2011", "Q3, 2019"]
Substitute(["Qtr 1, 2018", "Quarter 1, 2011", "Q1, 2019"], "1", "3")	Because the fourth argument (*InstanceNumber*) isn't provided, substitutes all instances of "1" in each record of the single-column table with "3".	A single-column table with a `Value` column containing the following values: ["Qtr 3, 2038", "Quarter 3, 2033", "Q3, 2039"]

Feedback

Was this page helpful? Yes No

Provide product feedback

Reset function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Resets a control to its default value, discarding any user changes.

Description

The **Reset** function resets a control to its **Default** property value. Any user changes are discarded.

You cannot reset controls that are within a **Gallery** or **Edit form** control from outside those controls. You can reset controls from formulas on controls within the same gallery or form. You can also reset all the controls within a form with the **ResetForm** function.

Using the **Reset** function is an alternative to toggling the **Reset** property of input controls and is generally preferred. The **Reset** property may be a better choice if many controls need to be reset together from multiple formulas. Toggling the **Reset** property can be done from a **Button** control with the formula **Reset** = **Button.Pressed** or from a variable with **Reset** = **MyVar** and toggling **MyVar** with the formula **Button.OnSelect** = **Set(MyVar, true); Set(MyVar, false)**.

Input controls are also reset when their **Default** property changes.

Reset has no return value, and you can use it only in behavior formulas.

Syntax

Reset(*Control*)

- *Control* – Required. The control to reset.

Example

1. Insert a **Text input** control on a screen. By default, its name will be **TextInput1** and its **Default** property will be set to **"Text input"**.
2. Type a new value in the text box.
3. Insert a **Button** control on the screen.
4. Set the button's **OnSelect** property to **Reset(TextInput1)**.

5. Select the button. This can be done even when authoring by selecting toward the ends of the control.
6. The contents of the text box will return to the value of the **Default** property.

EditForm, NewForm, SubmitForm, ResetForm, and ViewForm functions

Article • 03/22/2024

Applies to: Canvas apps

View, edit, or create an item, save the contents, and reset the controls in an **Edit form** control.

Overview

These functions change the state of the **Edit form** control. The form control can be in one of these modes:

Expand table

Mode	Description
FormMode.Edit	The form is populated with an existing record and the user can modify the values of the fields. Once complete, the user can save the changes to the record.
FormMode.New	The form is populated with default values and the user can modify the values of the fields. Once complete, the user can add the record to the data source.
FormMode.View	The form is populated with an existing record but the user cannot modify the values of the fields.

Description

These functions are often invoked from the **OnSelect** formula of a **Button** or **Image** control so that the user can save edits, abandon edits, or create a record. You can use controls and these functions together to create a complete solution.

These functions return no values.

You can use these functions only in behavior formulas.

SubmitForm

Use the **SubmitForm** function in the **OnSelect** property of a Button control to save any changes in a Form control to the data source.

Before submitting any changes, this function checks for validation issues with any field that's marked as required or that has one or more constraints on its value. This behavior matches that of the **Validate** function.

SubmitForm also checks the **Valid** property of the Form, which is an aggregation of all the **Valid** properties of the **Card** controls that the Form control contains. If a problem occurs, the data isn't submitted, and the **Error** and **ErrorKind** properties of the Form control are set accordingly.

If validation passes, **SubmitForm** submits the change to the data source.

- If successful, the Form's **OnSuccess** behavior runs, and the **Error** and **ErrorKind** properties are cleared. If the form was in **FormMode.New** mode, it is returned to **FormMode.Edit** mode.
- If unsuccessful, the Form's **OnFailure** behavior runs, and the **Error** and **ErrorKind** properties are set accordingly. The mode of the form is unchanged.

EditForm

The **EditForm** function changes the Form control's mode to **FormMode.Edit**. In this mode, the contents of the Form control's **Item** property are used to populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is changed, not created. **FormMode.Edit** is the default for the Form control.

> ⓘ **Note**
>
> When the form is in edit mode and Item is null, data card properties are not evaluated and will return default values.

NewForm

The **NewForm** function changes the Form control's mode to **FormMode.New**. In this mode, the contents of the Form control's **Item** property are ignored, and the default values of the Form's **DataSource** property populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is created, not changed.

ResetForm

The **ResetForm** function resets the contents of a form to their initial values, before the user made any changes. If the form is in **FormMode.New** mode, the form is reset to **FormMode.Edit** mode. The **OnReset** behavior of the form control also runs. You can also reset individual controls with the **Reset** function but only from within the form.

ViewForm

The **ViewForm** function changes the Form control's mode to **FormMode.View**. In this mode, the contents of the Form control's **Item** property are used to populate the form. The **SubmitForm** and **ResetForm** functions have no effect when in this mode.

DisplayMode Property

The current mode can be read through the **Mode** property. The mode also determines the value of the **DisplayMode** property, which can be used by data cards and controls within the form control. Often, the data card's **DisplayMode** property will be set to **Parent.DisplayMode** (referencing the form) as will the control's **DisplayMode** property (referencing the data card):

Expand table

Mode	DisplayMode	Description
FormMode.Edit	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept changes to a record.
FormMode.New	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept a new record.
FormMode.View	**DisplayMode.View**	Data cards and controls are not editable and optimized for viewing.

Syntax

SubmitForm(*FormName*)

- *FormName* - Required. Form control to submit to the data source.

EditForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.Edit** mode.

NewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.New** mode.

ResetForm(*FormName*)

- *FormName* - Required. Form control to reset to initial values. Also switches the form from **FormMode.New** mode to **FormMode.Edit** mode.

ViewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.View** mode.

Examples

See Understand data forms for complete examples.

1. Add a Button control, set its **Text** property to show **Save**, and set its **OnSelect** property to this formula:

   SubmitForm(EditForm)

2. Set the **OnFailure** property of a Form control to blank and its **OnSuccess** property to this formula:

   Back()

3. Name a **Label** control **ErrorText**, and set its **Text** property to this formula:

   EditForm.Error

   When the user selects the **Save** button, any changes in the Form control are submitted to the underlying data source.

   - If the submission succeeds, any changes are saved or, if the Form control is in **New** mode, a record is created. **ErrorText** is *blank* and the previous screen reappears.
   - If the submission fails, **ErrorText** shows a user-friendly error message, and the current screen remains visible so that the user can correct the problem and try again.

4. Add a Button control, set its **Text** property to show **Cancel**, and set its **OnSelect** property to this formula:

   ResetForm(EditForm); Back()

   When the user selects the **Cancel** button, the values in the Form control are reset to what they were before the user started to edit it, the previous screen reappears,

and the Form control is returned to **Edit** mode if it was in **New** mode.

5. Add a Button control, set its Text property to show **New**, and set its OnSelect property to this formula:

   NewForm(EditForm); Navigate(EditScreen, None)

   When the user selects the **New** button, the Form control switches to **New** mode, the default values for the Form control's data source populate that control, and the screen that contains the Form control appears. When the **SubmitForm** function runs, a record is created instead of updated.

Revert function

Article • 03/22/2024

Applies to: Canvas apps

Refreshes and clears errors for the records of a data source.

Description

The **Revert** function refreshes an entire data source or a single record in that data source. You'll see changes that other users made.

For the records reverted, **Revert** also clears any errors from the table that the **Errors** function returned.

If the **Errors** function reports a conflict after a **Patch** or other data operation, **Revert** the record to start with the conflicting version and reapply the change.

Revert has no return value. You can use it only in a behavior formula.

Syntax

Revert(*DataSource* [, *Record*])

- *DataSource* – Required. The data source that you want to revert.
- *Record* - Optional. The record that you want to revert. If you don't specify a record, the entire data source is reverted.

Example

In this example, you'll revert the data source named **IceCream**, which starts with the data in this table:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	300

A user on another device changes the **Quantity** property of the **Strawberry** record to **400**. At about the same time, you change the same property of the same record to **500**, not knowing about the other change.

You use the Patch function to update the record:
Patch(IceCream, LookUp(IceCream, Flavor = "Strawberry"), { Quantity: 500 })

You check the Errors table and find an error:

Expand table

Record	Column	Message	Error
{ ID: 1, Flavor: "Strawberry", Quantity: 300 }	*blank*	"The record you are trying to modify has been modified by another user. Please revert the record and try again."	ErrorKind.Conflict

Based on the **Error** column, you have a **Reload** button for which the OnSelect property to set to this formula:
Revert(IceCream, LookUp(IceCream, Flavor = "Strawberry"))

After you select the **Reload** button, the Errors table is empty, and the new value for **Strawberry** has been loaded:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	400

You reapply your change on top of the previous change, and your change succeed because the conflict has been resolved.

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	500

RequestHide function

Article • 03/22/2024

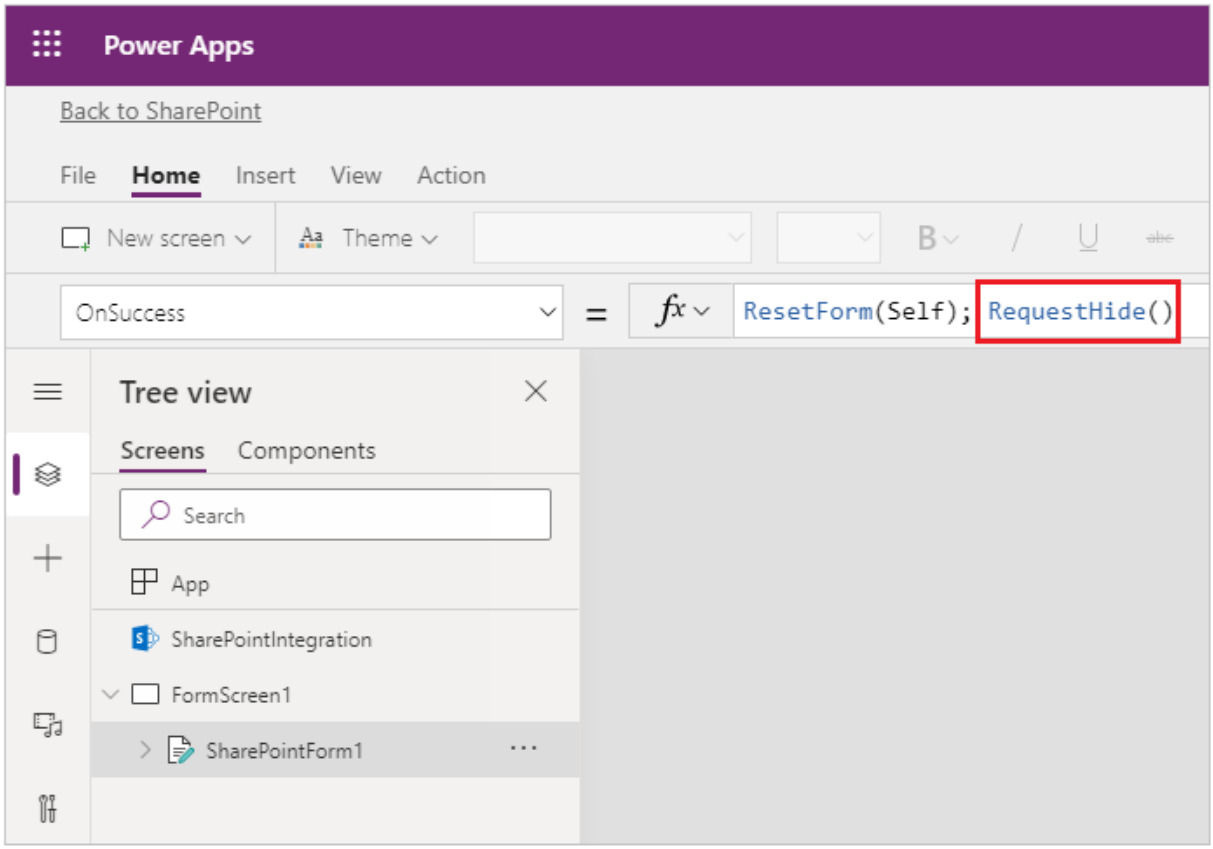
Applies to: Canvas apps

Hides the SharePoint form.

> **Note**
>
> Only works with SharePoint forms.

Description

Use the **RequestHide** function to hide the SharePoint form. By default, RequestHide() is used for the *OnSuccess* property of a SharePoint form being customized.



This function is not required for the **SharePointIntegration** control's **OnCancel** event as SharePoint by default hides the form when a user selects **Cancel**, and the function only reacts to a SharePoint form.

Syntax

RequestHide ( )

- No parameters.

Examples

Expand table

Formula	Description
RequestHide()	Hides the form.

Color enumeration and ColorFade, ColorValue, and RGBA functions

06/19/2025

Expand table

Functions	Applies to
Color	✅ Model-driven apps
ColorFade **ColorValue**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions
RGBA	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Use built-in color values, define custom colors, and use the alpha channel to control transparency.

> ⓘ Note
>
> PAC CLI pac power-fx commands don't support the **Color** enumeration.

Description

Use the **Color** enumeration to access colors defined by HTML's Cascading Style Sheets (CSS). For example, **Color.Red** returns pure red. A list of these colors appears at the end of this article.

The **ColorValue** function returns a color based on a CSS color string. The string can be in any of these forms:

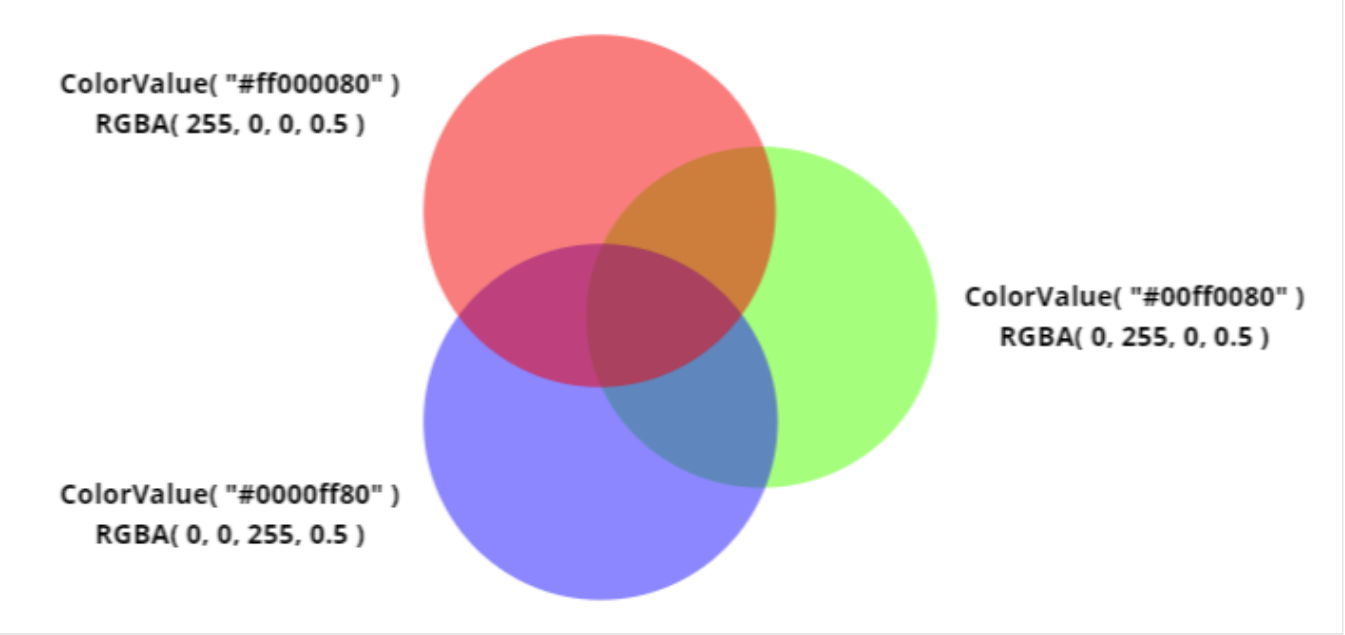
- **CSS color name:** **"RoxyBrown"** and **"OliveDrab"** are examples. These names don't include spaces. The list of supported colors appears later in this article.
- **6-digit hex value:** For example, **"#ffd700"** is the same as **"Gold"**. The string uses the format "*#rrggbb*", where *rr* is the red portion in two hexadecimal digits, *gg* is the green, and *bb* is the blue.
- **8-digit hex value:** For example, **"#ff7f5080"** is the same as **"Coral"** with a 50% alpha channel. The string uses the format "*#rrggbbaa*", where *rr*, *gg*, and *bb* are identical to the 6-digit form. The alpha channel is represented by *aa*: **00** means fully transparent, and **ff** means fully opaque.

The **RGBA** function returns a color based on red, green, and blue components. The function also includes an alpha channel for mixing colors of controls layered in front of each other. The alpha channel ranges from 0% or 0% (fully transparent and invisible) to 1% or 100% (fully opaque and completely blocks any layers behind a control).

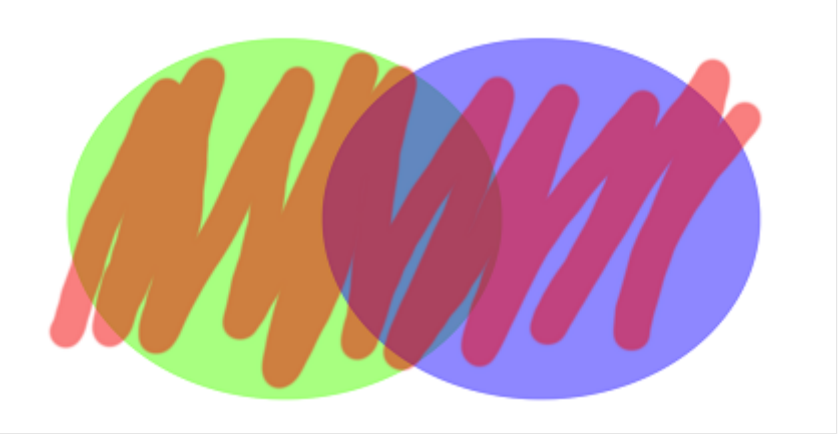
The **ColorFade** function returns a brighter or darker version of a color. The amount of fade ranges from -1 (fully darkens a color to black), to 0 (doesn't affect the color), to 1 (fully brightens a color to white).

Alpha channel

In a canvas app, you layer controls in front of one another and set the transparency of a control to any controls behind it. As a result, colors blend through the layers. For example, this diagram shows how the three primary colors mix with an alpha setting of 50 percent:



You can also blend images in file formats that support alpha channels. For example, you can't blend .jpeg files, but you can blend .png files. The next graphic shows the same red, green, and blue colors from the previous example, but the red color appears as a squiggle (instead of a circle) in a .png file with a 50 percent alpha channel:



If you specify a **Color** enumeration value or build a **ColorValue** formula with a color name or a 6-digit hexadecimal value, the alpha setting is 100 percent, which is fully opaque.

Syntax

Color.*ColorName*

- *ColorName* - Required. A Cascading Style Sheet (CSS) color name. The list of possible enumeration values is at the end of this article.

ColorValue(*CSSColor*)

- *CSSColor* - Required. A Cascading Style Sheet (CSS) color definition. S name, OliveDralike a hex value, like **#6b8e23** olikeffd420**. Hex values can use either #_rrggbb__usgormat. format

ColorValue(*Dynamic*)

- *Dynamic* - Required. A **Dynamic** value with a string that represents a Cascading Style Sheet (CSS) color definition.

RGBA(*Red*, *Green*, *Blue*, *Alpha*)

- *Red*, *Green*, *Blue* - Required. Color component values that range from 0 (no saturation) to 255 (full saturation).
- *Alpha* - Required. Alpha component that ranges from 0 (fully transparent) to 1 (fully opaque). You can also use a percentage, 0% to 100%.

ColorFade(*Color*, *FadeAmount*)

- *Color* - Required. A color value like **Color.Red** or the output from **ColorValue** or **RGBA**.
- *FadeAmount* - Required. A number between -1 and 1. -1 fully darkens a color to black, 0 doesn't affect the color, and 1 fully brightens a color to white. You can also use a percentage, from -100% to 100%.

Built-in colors

Expand table

Color enumeration	ColorValue	RGBA	Color Swatch
Color.AliceBlue	ColorValue("#f0f8ff") ColorValue("aliceblue")	RGBA(240, 248, 255, 1)	
Color.AntiqueWhite	ColorValue("#faebd7") ColorValue("AntiqueWhite")	RGBA(250, 235, 215, 1)	
Color.Aqua	ColorValue("#00ffff") ColorValue("AQUA")	RGBA(0, 255, 255, 1)	
Color.Aquamarine	ColorValue("#7fffd4") ColorValue("Aquamarine")	RGBA(127, 255, 212, 1)	
Color.Azure	ColorValue("#f0ffff") ColorValue("azure")	RGBA(240, 255, 255, 1)	
Color.Beige	ColorValue("#f5f5dc") ColorValue("Beige")	RGBA(245, 245, 220, 1)	
Color.Bisque	ColorValue("#ffe4c4") ColorValue("BISQUE")	RGBA(255, 228, 196, 1)	
Color.Black	ColorValue("#000000") ColorValue("Black")	RGBA(0, 0, 0, 1)	
Color.BlanchedAlmond	ColorValue("#ffebcd") ColorValue("blanchedalmond")	RGBA(255, 235, 205, 1)	
Color.Blue	ColorValue("#0000ff") ColorValue("Blue")	RGBA(0, 0, 255, 1)	
Color.BlueViolet	ColorValue("#8a2be2") ColorValue("BLUEVIOLET")	RGBA(138, 43, 226, 1)	
Color.Brown	ColorValue("#a52a2a") ColorValue("Brown")	RGBA(165, 42, 42, 1)	
Color.Burlywood	ColorValue("#deb887") ColorValue("burlywood")	RGBA(222, 184, 135, 1)	
Color.CadetBlue	ColorValue("#5f9ea0") ColorValue("CadetBlue")	RGBA(95, 158, 160, 1)	
Color.Chartreuse	ColorValue("#7fff00") ColorValue("CHARTREUSE")	RGBA(127, 255, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Chocolate	ColorValue("#d2691e") ColorValue("Chocolate")	RGBA(210, 105, 30, 1)	
Color.Coral	ColorValue("#ff7f50") ColorValue("coral")	RGBA(255, 127, 80, 1)	
Color.CornflowerBlue	ColorValue("#6495ed") ColorValue("CornflowerBlue")	RGBA(100, 149, 237, 1)	
Color.Cornsilk	ColorValue("#fff8dc") ColorValue("CORNSILK")	RGBA(255, 248, 220, 1)	
Color.Crimson	ColorValue("#dc143c") ColorValue("Crimson")	RGBA(220, 20, 60, 1)	
Color.Cyan	ColorValue("#00ffff") ColorValue("cyan")	RGBA(0, 255, 255, 1)	
Color.DarkBlue	ColorValue("#00008b") ColorValue("DarkBlue")	RGBA(0, 0, 139, 1)	
Color.DarkCyan	ColorValue("#008b8b") ColorValue("DARKCYAN")	RGBA(0, 139, 139, 1)	
Color.DarkGoldenRod	ColorValue("#b8860b") ColorValue("DarkGoldenRod")	RGBA(184, 134, 11, 1)	
Color.DarkGray	ColorValue("#a9a9a9") ColorValue("darkgray")	RGBA(169, 169, 169, 1)	
Color.DarkGreen	ColorValue("#006400") ColorValue("DarkGreen")	RGBA(0, 100, 0, 1)	
Color.DarkGrey	ColorValue("#a9a9a9") ColorValue("DARKGREY")	RGBA(169, 169, 169, 1)	
Color.DarkKhaki	ColorValue("#bdb76b") ColorValue("DarkKhaki")	RGBA(189, 183, 107, 1)	
Color.DarkMagenta	ColorValue("#8b008b") ColorValue("darkmagenta")	RGBA(139, 0, 139, 1)	
Color.DarkOliveGreen	ColorValue("#556b2f") ColorValue("DarkOliveGreen")	RGBA(85, 107, 47, 1)	
Color.DarkOrange	ColorValue("#ff8c00") ColorValue(	RGBA(255, 140, 0, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"DARKORANGE")		
Color.DarkOrchid	ColorValue("#9932cc") ColorValue("DarkOrchid")	RGBA(153, 50, 204, 1)	
Color.DarkRed	ColorValue("#8b0000") ColorValue("darkred")	RGBA(139, 0, 0, 1)	
Color.DarkSalmon	ColorValue("#e9967a") ColorValue("DarkSalmon")	RGBA(233, 150, 122, 1)	
Color.DarkSeaGreen	ColorValue("#8fbc8f") ColorValue("DARKSEAGREEN")	RGBA(143, 188, 143, 1)	
Color.DarkSlateBlue	ColorValue("#483d8b") ColorValue("DarkSlateBlue")	RGBA(72, 61, 139, 1)	
Color.DarkSlateGray	ColorValue("#2f4f4f") ColorValue("darkslategray")	RGBA(47, 79, 79, 1)	
Color.DarkSlateGrey	ColorValue("#2f4f4f") ColorValue("DarkSlateGrey")	RGBA(47, 79, 79, 1)	
Color.DarkTurquoise	ColorValue("#00ced1") ColorValue("DARKTURQUOISE")	RGBA(0, 206, 209, 1)	
Color.DarkViolet	ColorValue("#9400d3") ColorValue("DarkViolet")	RGBA(148, 0, 211, 1)	
Color.DeepPink	ColorValue("#ff1493") ColorValue("deeppink")	RGBA(255, 20, 147, 1)	
Color.DeepSkyBlue	ColorValue("#00bfff") ColorValue("DeepSkyBlue")	RGBA(0, 191, 255, 1)	
Color.DimGray	ColorValue("#696969") ColorValue("DIMGRAY")	RGBA(105, 105, 105, 1)	
Color.DimGrey	ColorValue("#696969") ColorValue("DimGrey")	RGBA(105, 105, 105, 1)	
Color.DodgerBlue	ColorValue("#1e90ff") ColorValue("dodgerblue")	RGBA(30, 144, 255, 1)	
Color.FireBrick	ColorValue("#b22222") ColorValue("FireBrick")	RGBA(178, 34, 34, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.FloralWhite	ColorValue("#fffaf0") ColorValue("FLORALWHITE")	RGBA(255, 250, 240, 1)	
Color.ForestGreen	ColorValue("#228b22") ColorValue("ForestGreen")	RGBA(34, 139, 34, 1)	
Color.Fuchsia	ColorValue("#ff00ff") ColorValue("fuchsia")	RGBA(255, 0, 255, 1)	
Color.Gainsboro	ColorValue("#dcdcdc") ColorValue("Gainsboro")	RGBA(220, 220, 220, 1)	
Color.GhostWhite	ColorValue("#f8f8ff") ColorValue("GHOSTWHITE")	RGBA(248, 248, 255, 1)	
Color.Gold	ColorValue("#ffd700") ColorValue("Gold")	RGBA(255, 215, 0, 1)	
Color.GoldenRod	ColorValue("#daa520") ColorValue("goldenrod")	RGBA(218, 165, 32, 1)	
Color.Gray	ColorValue("#808080") ColorValue("Gray")	RGBA(128, 128, 128, 1)	
Color.Green	ColorValue("#008000") ColorValue("GREEN")	RGBA(0, 128, 0, 1)	
Color.GreenYellow	ColorValue("#adff2f") ColorValue("GreenYellow")	RGBA(173, 255, 47, 1)	
Color.Grey	ColorValue("#808080") ColorValue("grey")	RGBA(128, 128, 128, 1)	
Color.Honeydew	ColorValue("#f0fff0") ColorValue("Honeydew")	RGBA(240, 255, 240, 1)	
Color.HotPink	ColorValue("#ff69b4") ColorValue("HOTPINK")	RGBA(255, 105, 180, 1)	
Color.IndianRed	ColorValue("#cd5c5c") ColorValue("IndianRed")	RGBA(205, 92, 92, 1)	
Color.Indigo	ColorValue("#4b0082") ColorValue("indigo")	RGBA(75, 0, 130, 1)	
Color.Ivory	ColorValue("#fffff0") ColorValue("Ivory")	RGBA(255, 255, 240, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Khaki	ColorValue("#f0e68c") ColorValue("KHAKI")	RGBA(240, 230, 140, 1)	
Color.Lavender	ColorValue("#e6e6fa") ColorValue("Lavender")	RGBA(230, 230, 250, 1)	
Color.LavenderBlush	ColorValue("#fff0f5") ColorValue("lavenderblush")	RGBA(255, 240, 245, 1)	
Color.LawnGreen	ColorValue("#7cfc00") ColorValue("LawnGreen")	RGBA(124, 252, 0, 1)	
Color.LemonChiffon	ColorValue("#fffacd") ColorValue("LEMONCHIFFON")	RGBA(255, 250, 205, 1)	
Color.LightBlue	ColorValue("#add8e6") ColorValue("LightBlue")	RGBA(173, 216, 230, 1)	
Color.LightCoral	ColorValue("#f08080") ColorValue("lightcoral")	RGBA(240, 128, 128, 1)	
Color.LightCyan	ColorValue("#e0ffff") ColorValue("LightCyan")	RGBA(224, 255, 255, 1)	
Color.LightGoldenRodYellow	ColorValue("#fafad2") ColorValue("lightgoldenrodyellow")	RGBA(250, 250, 210, 1)	
Color.LightGray	ColorValue("#d3d3d3") ColorValue("LightGray")	RGBA(211, 211, 211, 1)	
Color.LightGreen	ColorValue("#90ee90") ColorValue("lightgreen")	RGBA(144, 238, 144, 1)	
Color.LightGrey	ColorValue("#d3d3d3") ColorValue("LightGrey")	RGBA(211, 211, 211, 1)	
Color.LightPink	ColorValue("#ffb6c1") ColorValue("LIGHTPINK")	RGBA(255, 182, 193, 1)	
Color.LightSalmon	ColorValue("#ffa07a") ColorValue("LightSalmon")	RGBA(255, 160, 122, 1)	
Color.LightSeaGreen	ColorValue("#20b2aa") ColorValue("lightseagreen")	RGBA(32, 178, 170, 1)	
Color.LightSkyBlue	ColorValue("#87cefa") ColorValue("LightSkyBlue")	RGBA(135, 206, 250, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.LightSlateGray	ColorValue("#778899") ColorValue("LIGHTSLATEGRAY")	RGBA(119, 136, 153, 1)	
Color.LightSlateGrey	ColorValue("#778899") ColorValue("LightSlateGrey")	RGBA(119, 136, 153, 1)	
Color.LightSteelBlue	ColorValue("#b0c4de") ColorValue("lightsteelblue")	RGBA(176, 196, 222, 1)	
Color.LightYellow	ColorValue("#ffffe0") ColorValue("LightYellow")	RGBA(255, 255, 224, 1)	
Color.Lime	ColorValue("#00ff00") ColorValue("LIME")	RGBA(0, 255, 0, 1)	
Color.LimeGreen	ColorValue("#32cd32") ColorValue("LimeGreen")	RGBA(50, 205, 50, 1)	
Color.Linen	ColorValue("#faf0e6") ColorValue("linen")	RGBA(250, 240, 230, 1)	
Color.Magenta	ColorValue("#ff00ff") ColorValue("Magenta")	RGBA(255, 0, 255, 1)	
Color.Maroon	ColorValue("#800000") ColorValue("MAROON")	RGBA(128, 0, 0, 1)	
Color.MediumAquamarine	ColorValue("#66cdaa") ColorValue("MediumAquamarine")	RGBA(102, 205, 170, 1)	
Color.MediumBlue	ColorValue("#0000cd") ColorValue("mediumblue")	RGBA(0, 0, 205, 1)	
Color.MediumOrchid	ColorValue("#ba55d3") ColorValue("MediumOrchid")	RGBA(186, 85, 211, 1)	
Color.MediumPurple	ColorValue("#9370db") ColorValue("MEDIUMPURPLE")	RGBA(147, 112, 219, 1)	
Color.MediumSeaGreen	ColorValue("#3cb371") ColorValue("MediumSeaGreen")	RGBA(60, 179, 113, 1)	
Color.MediumSlateBlue	ColorValue("#7b68ee") ColorValue(	RGBA(123, 104, 238, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
	"mediumslateblue")		
Color.MediumSpringGreen	ColorValue("#00fa9a") ColorValue("MediumSpringGreen")	RGBA(0, 250, 154, 1)	
Color.MediumTurquoise	ColorValue("#48d1cc") ColorValue("MEDIUMTURQUOISE")	RGBA(72, 209, 204, 1)	
Color.MediumVioletRed	ColorValue("#c71585") ColorValue("MediumVioletRed")	RGBA(199, 21, 133, 1)	
Color.MidnightBlue	ColorValue("#191970") ColorValue("midnightblue")	RGBA(25, 25, 112, 1)	
Color.MintCream	ColorValue("#f5fffa") ColorValue("MintCream")	RGBA(245, 255, 250, 1)	
Color.MistyRose	ColorValue("#ffe4e1") ColorValue("MISTYROSE")	RGBA(255, 228, 225, 1)	
Color.Moccasin	ColorValue("#ffe4b5") ColorValue("Moccasin")	RGBA(255, 228, 181, 1)	
Color.NavajoWhite	ColorValue("#ffdead") ColorValue("navajowhite")	RGBA(255, 222, 173, 1)	
Color.Navy	ColorValue("#000080") ColorValue("Navy")	RGBA(0, 0, 128, 1)	
Color.OldLace	ColorValue("#fdf5e6") ColorValue("OLDLACE")	RGBA(253, 245, 230, 1)	
Color.Olive	ColorValue("#808000") ColorValue("Olive")	RGBA(128, 128, 0, 1)	
Color.OliveDrab	ColorValue("#6b8e23") ColorValue("olivedrab")	RGBA(107, 142, 35, 1)	
Color.Orange	ColorValue("#ffa500") ColorValue("Orange")	RGBA(255, 165, 0, 1)	
Color.OrangeRed	ColorValue("#ff4500") ColorValue("ORANGERED")	RGBA(255, 69, 0, 1)	
Color.Orchid	ColorValue("#da70d6") ColorValue("Orchid")	RGBA(218, 112, 214, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.PaleGoldenRod	ColorValue("#eee8aa") ColorValue("palegoldenrod")	RGBA(238, 232, 170, 1)	
Color.PaleGreen	ColorValue("#98fb98") ColorValue("PaleGreen")	RGBA(152, 251, 152, 1)	
Color.PaleTurquoise	ColorValue("#afeeee") ColorValue("PALETURQUOISE")	RGBA(175, 238, 238, 1)	
Color.PaleVioletRed	ColorValue("#db7093") ColorValue("PaleVioletRed")	RGBA(219, 112, 147, 1)	
Color.PapayaWhip	ColorValue("#ffefd5") ColorValue("papayawhip")	RGBA(255, 239, 213, 1)	
Color.PeachPuff	ColorValue("#ffdab9") ColorValue("PeachPuff")	RGBA(255, 218, 185, 1)	
Color.Peru	ColorValue("#cd853f") ColorValue("PERU")	RGBA(205, 133, 63, 1)	
Color.Pink	ColorValue("#ffc0cb") ColorValue("Pink")	RGBA(255, 192, 203, 1)	
Color.Plum	ColorValue("#dda0dd") ColorValue("plum")	RGBA(221, 160, 221, 1)	
Color.PowderBlue	ColorValue("#b0e0e6") ColorValue("PowderBlue")	RGBA(176, 224, 230, 1)	
Color.Purple	ColorValue("#800080") ColorValue("PURPLE")	RGBA(128, 0, 128, 1)	
Color.Red	ColorValue("#ff0000") ColorValue("Red")	RGBA(255, 0, 0, 1)	
Color.RosyBrown	ColorValue("#bc8f8f") ColorValue("rosybrown")	RGBA(188, 143, 143, 1)	
Color.RoyalBlue	ColorValue("#4169e1") ColorValue("RoyalBlue")	RGBA(65, 105, 225, 1)	
Color.SaddleBrown	ColorValue("#8b4513") ColorValue("SADDLEBROWN")	RGBA(139, 69, 19, 1)	
Color.Salmon	ColorValue("#fa8072") ColorValue("Salmon")	RGBA(250, 128, 114, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.SandyBrown	ColorValue("#f4a460") ColorValue("sandybrown")	RGBA(244, 164, 96, 1)	
Color.SeaGreen	ColorValue("#2e8b57") ColorValue("SeaGreen")	RGBA(46, 139, 87, 1)	
Color.SeaShell	ColorValue("#fff5ee") ColorValue("SEASHELL")	RGBA(255, 245, 238, 1)	
Color.Sienna	ColorValue("#a0522d") ColorValue("Sienna")	RGBA(160, 82, 45, 1)	
Color.Silver	ColorValue("#c0c0c0") ColorValue("silver")	RGBA(192, 192, 192, 1)	
Color.SkyBlue	ColorValue("#87ceeb") ColorValue("SkyBlue")	RGBA(135, 206, 235, 1)	
Color.SlateBlue	ColorValue("#6a5acd") ColorValue("SLATEBLUE")	RGBA(106, 90, 205, 1)	
Color.SlateGray	ColorValue("#708090") ColorValue("SlateGray")	RGBA(112, 128, 144, 1)	
Color.SlateGrey	ColorValue("#708090") ColorValue("slategrey")	RGBA(112, 128, 144, 1)	
Color.Snow	ColorValue("#fffafa") ColorValue("Snow")	RGBA(255, 250, 250, 1)	
Color.SpringGreen	ColorValue("#00ff7f") ColorValue("SPRINGGREEN")	RGBA(0, 255, 127, 1)	
Color.SteelBlue	ColorValue("#4682b4") ColorValue("SteelBlue")	RGBA(70, 130, 180, 1)	
Color.Tan	ColorValue("#d2b48c") ColorValue("tan")	RGBA(210, 180, 140, 1)	
Color.Teal	ColorValue("#008080") ColorValue("Teal")	RGBA(0, 128, 128, 1)	
Color.Thistle	ColorValue("#d8bfd8") ColorValue("THISTLE")	RGBA(216, 191, 216, 1)	
Color.Tomato	ColorValue("#ff6347") ColorValue("Tomato")	RGBA(255, 99, 71, 1)	

Color enumeration	ColorValue	RGBA	Color Swatch
Color.Transparent	ColorValue("#00000000") ColorValue("Transparent")	RGBA(0, 0, 0, 0)	
Color.Turquoise	ColorValue("#40e0d0") ColorValue("turquoise")	RGBA(64, 224, 208, 1)	
Color.Violet	ColorValue("#ee82ee") ColorValue("Violet")	RGBA(238, 130, 238, 1)	
Color.Wheat	ColorValue("#f5deb3") ColorValue("WHEAT")	RGBA(245, 222, 179, 1)	
Color.White	ColorValue("#ffffff") ColorValue("White")	RGBA(255, 255, 255, 1)	
Color.WhiteSmoke	ColorValue("#f5f5f5") ColorValue("whitesmoke")	RGBA(245, 245, 245, 1)	
Color.Yellow	ColorValue("#ffff00") ColorValue("Yellow")	RGBA(255, 255, 0, 1)	
Color.YellowGreen	ColorValue("#9acd32") ColorValue("YELLOWGREEN")	RGBA(154, 205, 50, 1)	

Left, Mid, and Right functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Extracts the left, middle, or right portion of a string of text.

Description

The **Left**, **Mid**, and **Right** functions return a portion of a string.

- **Left** returns the beginning characters of a string.
- **Mid** returns the middle characters of a string.
- **Right** returns the ending characters of a string.

If you specify a single string as an argument, the function returns the portion that you requested of the string. If you specify a single-column table that contains strings, the function returns a single-column table with a **Value** column containing the portions that you requested of those strings. If you specify a multi-column table, you can shape it into a single-column table, as working with tables describes.

If the starting position is negative or beyond the end of the string, **Mid** returns *blank*. You can check the length of a string by using the **Len** function. If you request more characters than the string contains, the function returns as many characters as possible.

Syntax

Left(*String*, *NumberOfCharacters*)
Mid(*String*, *StartingPosition* [, *NumberOfCharacters*])
Right(*String*, *NumberOfCharacters*)

- *String* - Required. The string to from which to extract the result.
- *StartingPosition* - Required (**Mid** only). The starting position. The first character of the string is position 1.
- *NumberOfCharacters* - Required (**Left** and **Right** only). The number of characters to return. If omitted for the **Mid** function, the function returns the portion from the starting position until the end of the string.

Left(*SingleColumnTable*, *NumberOfCharacters*)
Mid(*SingleColumnTable*, *StartingPosition* [, *NumberOfCharacters*])

Right(*SingleColumnTable*, *NumberOfCharacters*)

- *SingleColumnTable* - Required. A single-column table of strings from which to extract the results.
- *StartingPosition* - Required (**Mid** only). The starting position. The first character of the string is position 1.
- *NumberOfCharacters* - Required (**Left** and **Right** only). The number of characters to return. If omitted for the **Mid** function, the function returns the portion from the starting position until the end of the string.

Examples

Single string

The examples in this section use a text-input control as their data source. The control is named **Author** and contains the string "E. E. Cummings".

Expand table

Formula	Description	Result
Left(Author.Text, 5)	Extracts up to five characters from the start of the string.	"E. E."
Mid(Author.Text, 7, 4)	Extracts up to four characters, starting with the seventh character, from the string.	"Cumm"
Mid(Author.Text, 7)	Extracts all characters, starting with the seventh character, from the string.	"Cummings"
Right(Author.Text, 5)	Extracts up to five characters from the end of the string.	"mings"

Single-column table

Each example in this section extracts strings from the **Address** column of this data source, named **People**, and returns a single-column table that contains the results:

Expand table

Name	Address
"Jean"	"123 Main St NE"

Name	Address
"Fred"	"789 SW 39th #3B"

Expand table

Formula	Description	Result
Left(ShowColumns(People, "Address"), 8)	Extracts the first eight characters of each string.	A single-column table with a `Value` column containing the following values: "123 Main", "789 SW 3"
Mid(ShowColumns(People, "Address"), 5, 7)	Extracts the middle seven characters of each string, starting with the fifth character.	A single-column table with a `Value` column containing the following values: "Main St", "SW 39th"
Right(ShowColumns(People, "Address"), 7)	Extracts the last seven characters of each string.	A single-column table with a `Value` column containing the following values: "n St NE", "9th #3B"

Step-by-step example

1. Import or create a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.
2. Set the **Text** property of the lower label in the gallery to this function:

   Right(ThisItem.ProductName, 3)

   The label shows the last three characters of each product name.

Feedback

Was this page helpful? Yes No

Provide product feedback

Int, Round, RoundDown, RoundUp, and Trunc functions

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Rounds a number.

Round, RoundDown, and RoundUp

The **Round**, **RoundDown**, and **RoundUp** functions round a number to the specified number of decimal places:

- **Round** rounds up if the next digit is 5 or higher. Otherwise, this function rounds down.
- **RoundDown** always rounds down to the previous lower number, towards zero.
- **RoundUp** always rounds up to the next higher number, away from zero.

The number of decimal places can be specified for these functions:

Expand table

Decimal places	Description	Example
Greater than 0	The number is rounded to the right of the decimal separator.	`Round( 12.37, 1 )` returns 12.4.
0	The number is rounded to the nearest integer.	`Round( 12.37, 0 )` returns 12.
Less than 0	The number is rounded to the left of the decimal separator.	`Round( 12.37, -1 )` returns 10.

Int and Trunc

The **Int** and **Trunc** functions round a number to an integer (whole number without a decimal):

- **Int** rounds down to the nearest integer.
- **Trunc** truncates the number to just the integer portion by removing any decimal portion.

The difference between **Int** and **Trunc** is in the handling of negative numbers. For example, for an argument of `-4.3`, **Int** will return the integer further away from zero, `-5`, while **Trunc** will return the integer closer to zero, `-4`. **Int** returns values that are unique amongst the five rounding functions, while **Trunc** returns the same values as **RoundDown**.

Use **Trunc** to extract the decimal portion of a number by subtracting it from the original, for example `X - Trunc(X)`.

Decimal places cannot be specified with **Trunc** as it can with Microsoft Excel. Use **RoundDown** instead when this is needed.

Single-column tables

These functions support single-column tables. If you pass a single number, the return value is the rounded version of that number. If you pass a single-column table that contains numbers, the return value is a single-column table of rounded numbers. The *DecimalPlaces* parameter can be a single value or a single-column table. If the single-column table has less values that the *Number*, zero is used for the remaining values. Use **ShowColumns** and other table shaping functions to extract a single-column table from a larger table.

Syntax

Round(*Number*, *DecimalPlaces*)
RoundDown(*Number*, *DecimalPlaces*)
RoundUp(*Number*, *DecimalPlaces*)

- *Number* - Required. The number to round.
- *DecimalPlaces* - Required. Number of decimal places to round to. Use a positive value to indicate decimal places right of the decimal separator, a negative value to the left, and zero for a whole number.

Int(*Number*)
Trunc(*Number*)

- *Number* - Required. The number to be rounded to an integer.

Examples

Rounding to a whole number.

Expand table

X	Round(X, 0)	RoundUp(X, 0)	RoundDown(X, 0)	Int(X)	Trunc(X)
7.9	8	8	7	7	7
-7.9	-8	-8	-7	-8	-7
7.5	8	8	7	7	7
-7.5	-8	-8	-7	-8	-7
7.1	7	8	7	7	7
-7.1	-7	-8	-7	-8	-7

Rounding to two decimal places to the right of the decimal separator (0.01).

Expand table

X	Round(X, 2)	RoundUp(X, 2)	RoundDown(X, 2)
430.123	430.12	430.13	430.12
430.125	430.13	430.13	430.12
430.128	430.13	430.13	430.12

Rounding to two decimal places to the left of the decimal separator (100).

Expand table

X	Round(X, -2)	RoundUp(X, -2)	RoundDown(X, -2)
430.123	400	500	400
449.942	400	500	400
450.000	500	500	400
450.124	500	500	400
479.128	500	500	400

Rounding a single-column table of values.

Expand table

`X`	`Int( X )`	`Round( X, 2 )`	`RoundDown( X, [ 0, 1, 2 ] )`	`RoundUp( X, [ 2 ] )`
[123.456, 987.593, 542.639]	[123, 987, 542]	[123.46, 987.59, 542.64]	[123, 987.5, 542.63]	[123.46, 988, 543]

Int, Round, RoundDown, RoundUp, and Trunc functions

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Rounds a number.

Round, RoundDown, and RoundUp

The **Round**, **RoundDown**, and **RoundUp** functions round a number to the specified number of decimal places:

- **Round** rounds up if the next digit is 5 or higher. Otherwise, this function rounds down.
- **RoundDown** always rounds down to the previous lower number, towards zero.
- **RoundUp** always rounds up to the next higher number, away from zero.

The number of decimal places can be specified for these functions:

Expand table

Decimal places	Description	Example
Greater than 0	The number is rounded to the right of the decimal separator.	`Round( 12.37, 1 )` returns 12.4.
0	The number is rounded to the nearest integer.	`Round( 12.37, 0 )` returns 12.
Less than 0	The number is rounded to the left of the decimal separator.	`Round( 12.37, -1 )` returns 10.

Int and Trunc

The **Int** and **Trunc** functions round a number to an integer (whole number without a decimal):

- **Int** rounds down to the nearest integer.
- **Trunc** truncates the number to just the integer portion by removing any decimal portion.

The difference between **Int** and **Trunc** is in the handling of negative numbers. For example, for an argument of `-4.3`, **Int** will return the integer further away from zero, `-5`, while **Trunc** will return the integer closer to zero, `-4`. **Int** returns values that are unique amongst the five rounding functions, while **Trunc** returns the same values as **RoundDown**.

Use **Trunc** to extract the decimal portion of a number by subtracting it from the original, for example `X - Trunc(X)`.

Decimal places cannot be specified with **Trunc** as it can with Microsoft Excel. Use **RoundDown** instead when this is needed.

Single-column tables

These functions support single-column tables. If you pass a single number, the return value is the rounded version of that number. If you pass a single-column table that contains numbers, the return value is a single-column table of rounded numbers. The *DecimalPlaces* parameter can be a single value or a single-column table. If the single-column table has less values that the *Number*, zero is used for the remaining values. Use **ShowColumns** and other table shaping functions to extract a single-column table from a larger table.

Syntax

Round(*Number*, *DecimalPlaces*)
RoundDown(*Number*, *DecimalPlaces*)
RoundUp(*Number*, *DecimalPlaces*)

- *Number* - Required. The number to round.
- *DecimalPlaces* - Required. Number of decimal places to round to. Use a positive value to indicate decimal places right of the decimal separator, a negative value to the left, and zero for a whole number.

Int(*Number*)
Trunc(*Number*)

- *Number* - Required. The number to be rounded to an integer.

Examples

Rounding to a whole number.

Expand table

X	Round(X, 0)	RoundUp(X, 0)	RoundDown(X, 0)	Int(X)	Trunc(X)
7.9	8	8	7	7	7
-7.9	-8	-8	-7	-8	-7
7.5	8	8	7	7	7
-7.5	-8	-8	-7	-8	-7
7.1	7	8	7	7	7
-7.1	-7	-8	-7	-8	-7

Rounding to two decimal places to the right of the decimal separator (0.01).

Expand table

X	Round(X, 2)	RoundUp(X, 2)	RoundDown(X, 2)
430.123	430.12	430.13	430.12
430.125	430.13	430.13	430.12
430.128	430.13	430.13	430.12

Rounding to two decimal places to the left of the decimal separator (100).

Expand table

X	Round(X, -2)	RoundUp(X, -2)	RoundDown(X, -2)
430.123	400	500	400
449.942	400	500	400
450.000	500	500	400
450.124	500	500	400
479.128	500	500	400

Rounding a single-column table of values.

Expand table

`X`	`Int( X )`	`Round( X, 2 )`	`RoundDown( X, [ 0, 1, 2 ] )`	`RoundUp( X, [ 2 ] )`
[123.456, 987.593, 542.639]	[123, 987, 542]	[123.46, 987.59, 542.64]	[123, 987.5, 542.63]	[123.46, 988, 543]

Int, Round, RoundDown, RoundUp, and Trunc functions

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Rounds a number.

Round, RoundDown, and RoundUp

The **Round**, **RoundDown**, and **RoundUp** functions round a number to the specified number of decimal places:

- **Round** rounds up if the next digit is 5 or higher. Otherwise, this function rounds down.
- **RoundDown** always rounds down to the previous lower number, towards zero.
- **RoundUp** always rounds up to the next higher number, away from zero.

The number of decimal places can be specified for these functions:

Expand table

Decimal places	Description	Example
Greater than 0	The number is rounded to the right of the decimal separator.	`Round( 12.37, 1 )` returns 12.4.
0	The number is rounded to the nearest integer.	`Round( 12.37, 0 )` returns 12.
Less than 0	The number is rounded to the left of the decimal separator.	`Round( 12.37, -1 )` returns 10.

Int and Trunc

The **Int** and **Trunc** functions round a number to an integer (whole number without a decimal):

- **Int** rounds down to the nearest integer.
- **Trunc** truncates the number to just the integer portion by removing any decimal portion.

The difference between **Int** and **Trunc** is in the handling of negative numbers. For example, for an argument of `-4.3`, **Int** will return the integer further away from zero, `-5`, while **Trunc** will return the integer closer to zero, `-4`. **Int** returns values that are unique amongst the five rounding functions, while **Trunc** returns the same values as **RoundDown**.

Use **Trunc** to extract the decimal portion of a number by subtracting it from the original, for example `X - Trunc(X)`.

Decimal places cannot be specified with **Trunc** as it can with Microsoft Excel. Use **RoundDown** instead when this is needed.

Single-column tables

These functions support single-column tables. If you pass a single number, the return value is the rounded version of that number. If you pass a single-column table that contains numbers, the return value is a single-column table of rounded numbers. The *DecimalPlaces* parameter can be a single value or a single-column table. If the single-column table has less values that the *Number*, zero is used for the remaining values. Use **ShowColumns** and other table shaping functions to extract a single-column table from a larger table.

Syntax

Round(*Number*, *DecimalPlaces*)
RoundDown(*Number*, *DecimalPlaces*)
RoundUp(*Number*, *DecimalPlaces*)

- *Number* - Required. The number to round.
- *DecimalPlaces* - Required. Number of decimal places to round to. Use a positive value to indicate decimal places right of the decimal separator, a negative value to the left, and zero for a whole number.

Int(*Number*)
Trunc(*Number*)

- *Number* - Required. The number to be rounded to an integer.

Examples

Rounding to a whole number.

Expand table

X	Round(X, 0)	RoundUp(X, 0)	RoundDown(X, 0)	Int(X)	Trunc(X)
7.9	8	8	7	7	7
-7.9	-8	-8	-7	-8	-7
7.5	8	8	7	7	7
-7.5	-8	-8	-7	-8	-7
7.1	7	8	7	7	7
-7.1	-7	-8	-7	-8	-7

Rounding to two decimal places to the right of the decimal separator (0.01).

Expand table

X	Round(X, 2)	RoundUp(X, 2)	RoundDown(X, 2)
430.123	430.12	430.13	430.12
430.125	430.13	430.13	430.12
430.128	430.13	430.13	430.12

Rounding to two decimal places to the left of the decimal separator (100).

Expand table

X	Round(X, -2)	RoundUp(X, -2)	RoundDown(X, -2)
430.123	400	500	400
449.942	400	500	400
450.000	500	500	400
450.124	500	500	400
479.128	500	500	400

Rounding a single-column table of values.

Expand table

`X`	`Int( X )`	`Round( X, 2 )`	`RoundDown( X, [ 0, 1, 2 ] )`	`RoundUp( X, [ 2 ] )`
[123.456, 987.593, 542.639]	[123, 987, 542]	[123.46, 987.59, 542.64]	[123, 987.5, 542.63]	[123.46, 988, 543]

SaveData, LoadData, and ClearData functions

06/18/2025

Applies to: Canvas apps

Saves and reloads a collection from the app host's storage.

Description

The **SaveData** function stores a collection for later use under a name.

The **LoadData** function reloads a collection by name that was previously saved with **SaveData**. You can't use this function to load a collection from another source.

The **ClearData** function clears the storage under a specific name or clears all storage associated with the app if no name is provided.

> **Note**
>
> - The name shared between **SaveData**, **LoadData**, and **ClearData** is a key, not a file name. It need not be complex as names are unique to each app and there is no danger of name conflict. The name must not contain any of these characters: `*".?:\<>|/`.
> - SaveData is limited to 1 MB of data for Power Apps running in Teams and in a web browser. There is no fixed limit for Power Apps running in a mobile player but there are practical limits discussed below.
> - Don't use **SaveData** to store sensitive data in the web since it'll be stored in plain text.

Use these functions to improve app-startup performance by:

- Caching data in the **App.OnStart** formula on a first run.
- Reloading the local cache on next runs.

You can also use these functions to add simple offline capabilities to your app.

You can't use these functions inside a browser when:

- Authoring the app in Power Apps Studio.

To test your app, run it in Power Apps Mobile on an iPhone or Android device.

These functions are limited by the amount of available app memory as they operate on an in-memory collection. Available memory can vary depending on factors such as:

- The device and operating system.
- The memory that the Power Apps player uses.
- Complexity of the app with screens and controls.

Test your app with expected scenarios on the type of devices you expect the app to run when storing large data. Expect to have between 30 MB and 70 MB of available memory generally.

These functions depend on the collection being implicitly defined with **Collect** or **ClearCollect**. You don't need to call **Collect** or **ClearCollect** to load data into the collection for defining it. It's a common case when using **LoadData** after a previous **SaveData**. All that is needed is the presence of these functions in a formula to implicitly define the structure of the collection. For more information, see creating and removing variables.

The loaded data will be appended to the collection. Use the **Clear** function before calling **LoadData** if you want to start with an empty collection.

Data security

Consider carefully the isolation and encryption of data stored with **SaveData** and decide if it's appropriate for your needs, especially if devices are shared by multiple users.

Data stored with **SaveData** is isolated from other Power Apps by the Power Apps players. Data is stored based on the app's App ID, automatically isolating the **SaveData** name space between Power Apps.

The operating system and browser is responsible for isolating data between Power Apps and other apps on a device and with websites. For example, the operating system is responsible for isolating data stored in Microsoft Outlook from data stored in Power Apps, and also isolating that data from websites such as Bing.com or PowerApps.com. The operating system's built in app sandbox facilities are used for **SaveData** storage which is usually not accessible to or hidden from the user.

When using the same app, the operating system and browser is also responsible for isolating the data between different operating system level users. For example, if two different users share a computer and use two different Windows login credentials, the operating system is responsible for isolating data between the two Windows users.

Data may or may not be isolated between different Power Apps users if the operating system user is the same. Not every Power Apps player treats this the same way. For example, while

logged in as the same Windows user, in the Power Apps player, the user signs out of Power Apps and signs in as a different Power Apps user. Data stored in an app before the change of Power Apps user, may be accessible to the second Power Apps user within the same app. The data may also be removed and the first Power Apps user may no longer be able to access it. The behavior varies between Power Apps players.

The operating system may also encrypt the data or you can use a mobile device management tool such as Microsoft Intune. Data stored when playing an app in a web browser is not encrypted.

Syntax

SaveData(*Collection*, *Name*)
LoadData(*Collection*, *Name* [, *IgnoreNonexistentFile*])

- *Collection* - Required. Collection to be stored or loaded.
- *Name* - Required. Name of the storage. The name must be same to save and load same set of data. The name space isn't shared with other apps. Names must not contain any of these characters: `*".?:\<>|/`.
- *IgnoreNonexistentFile* - Optional. A Boolean value indicating what to do if the file doesn't already exist. Use *false* (default) to return an error and *true* to suppress the error.

ClearData([*Name*])

- *Name* - Optional. Name of the storage previously saved with **SaveData**. If *Name* is not provided, all storage associated with the app is cleared.

Examples

Expand table

Formula	Description	Result
SaveData(LocalCache, "MyCache")	Save the **LocalCache** collection to the user's device under the name "MyCache", suitable for **LoadData** to retrieve later.	Data is saved to the app host under the name "MyCache".
LoadData(LocalCache, "MyCache")	Loads the **LocalCache** collection from the user's device under the name "MyCache", previously stored with a call to **SaveData**.	Data is loaded from the app host under the name "MyCache".
ClearData("MyCache")	Clears the storage under the name "MyCache". Any data stored under this name will no longer be available through **LoadData**.	Data is removed from the app host under the name "MyCache".

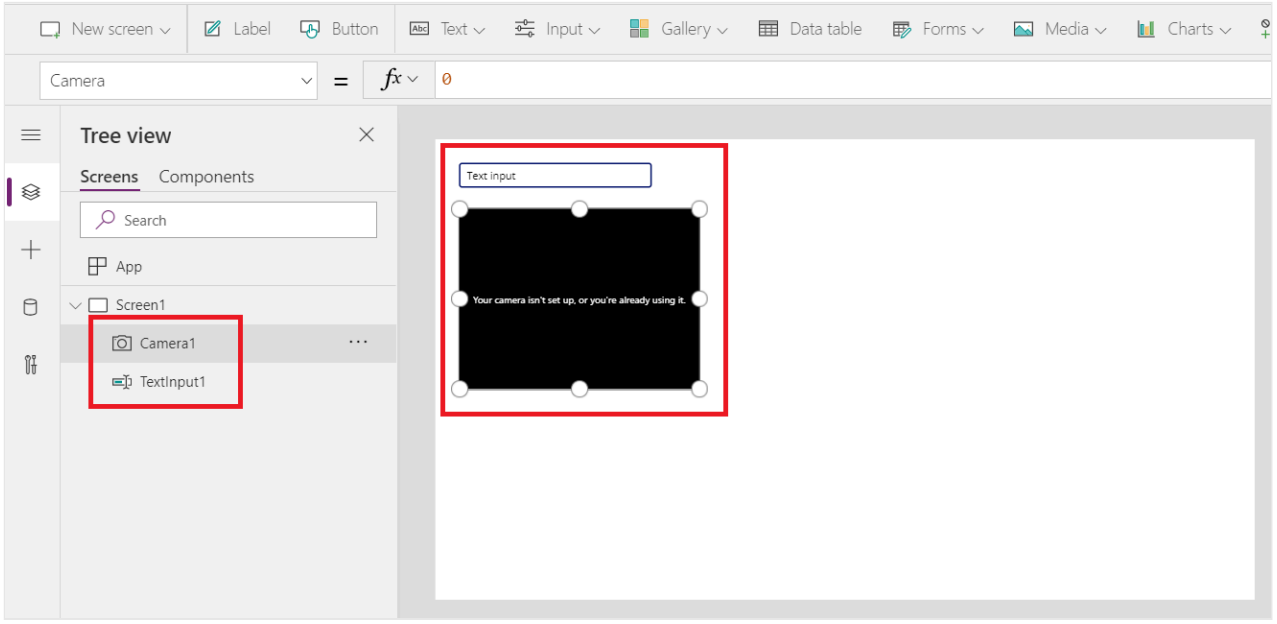
Formula	Description	Result
ClearData()	Clear all storage associated with this app. Data stored by other apps is not affected.	All data is removed from the app host.

Simple offline example

Following simple example captures and stores the names and pictures of everyday items while offline. It stores the information in the device's local storage for later use. This allows the app to be closed or the device to restart without losing data.

> **Note**
>
> This example uses a camera control to capture images. Since **SaveData** is limited to 1 MB of data when running in Teams or a web browser, this example will not work with more than a few images. Also, depending on the camera, it may not work with even one image. Use a device to work through this full example, or remove the camera control and picture part of this example to run in Teams or in a web browser.

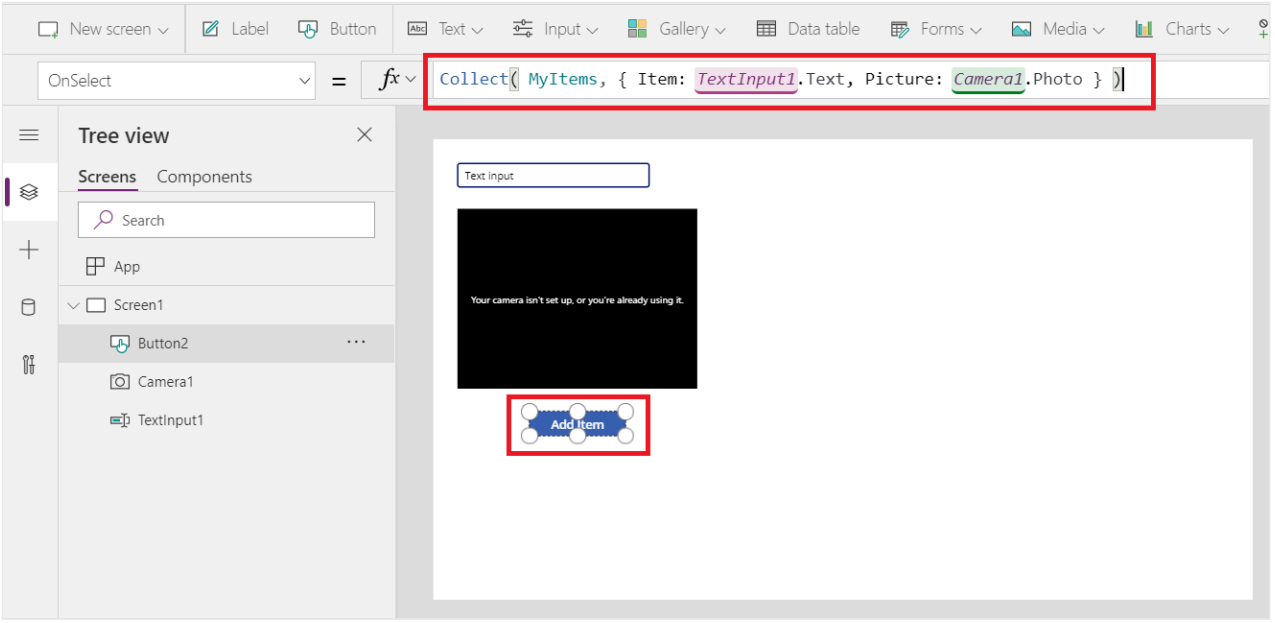
1. Create a blank canvas app with a tablet layout. For more details, read creating an app from a template and select **Tablet layout** under **Blank app**.
2. Add a **Text input** control and a **Camera** control and arrange them roughly as shown:



3. Add a **Button** control.
4. Double-click the button control to change the button text to **Add Item** (or modify the **Text** property).

5. Set the **OnSelect** property of the button control to this formula that will add an item to our collection:

```
Power Fx

Collect( MyItems, { Item: TextInput1.Text, Picture: Camera1.Photo } )
```

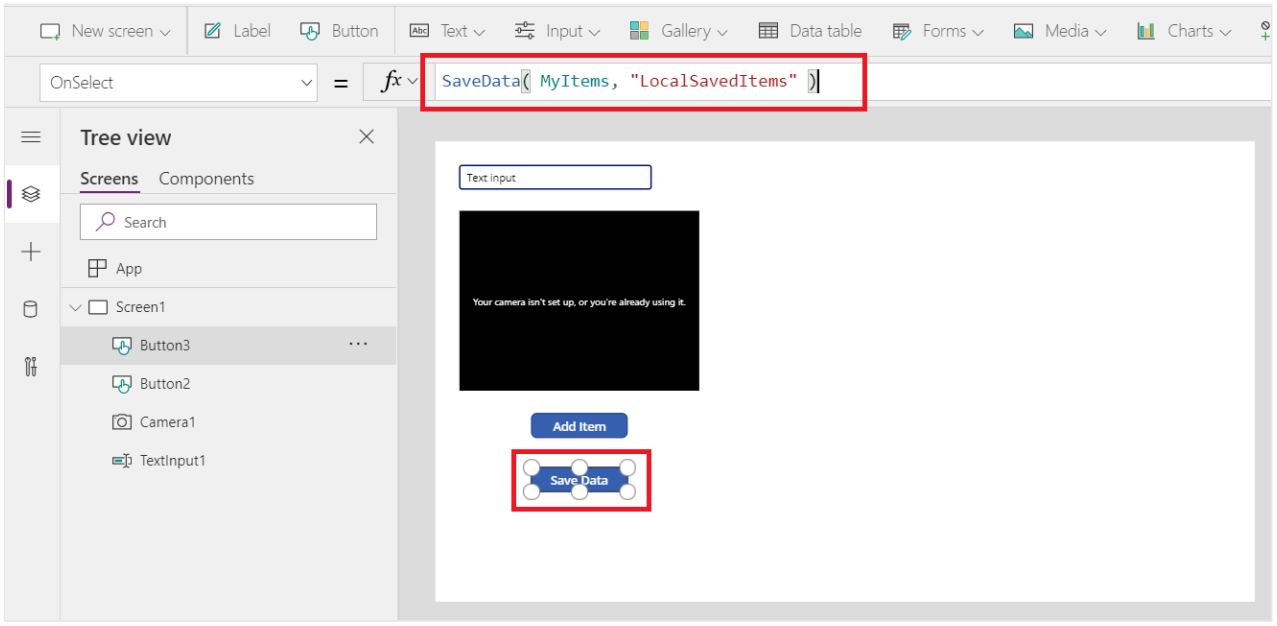


6. Add another **Button** control.

7. Double-click the button control to change the button text to **Save Data** (or modify the **Text** property).

8. Set the **OnSelect** property of the button control to this formula in order to save our collection to the local device:

```
Power Fx

SaveData( MyItems, "LocalSavedItems" )
```



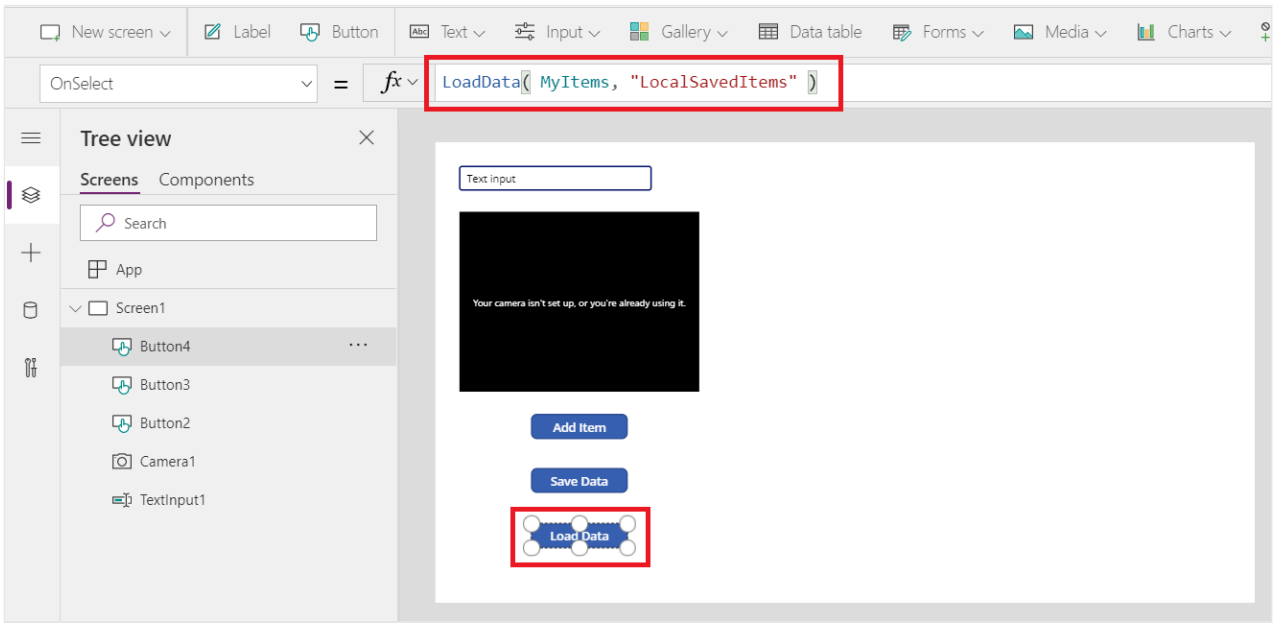
It's tempting to test the button as it doesn't affect anything. But you'll only see an error as you're authoring in a web browser. Save the app first and open on a device before you follow the next steps to test this formula:

9. Add a third **Button** control.

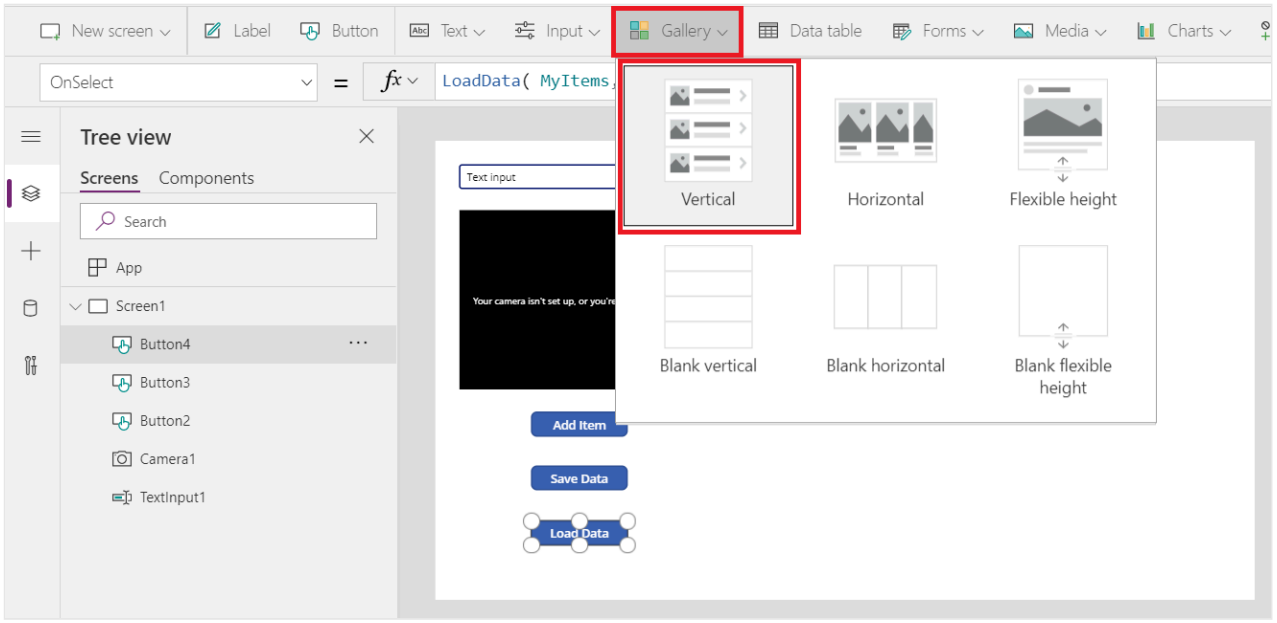
10. Double-click the button control to change the button text to **Load Data** (or modify the **Text** property).

11. Set the **OnSelect** property of the button control to this formula in order to load our collection from the local device:

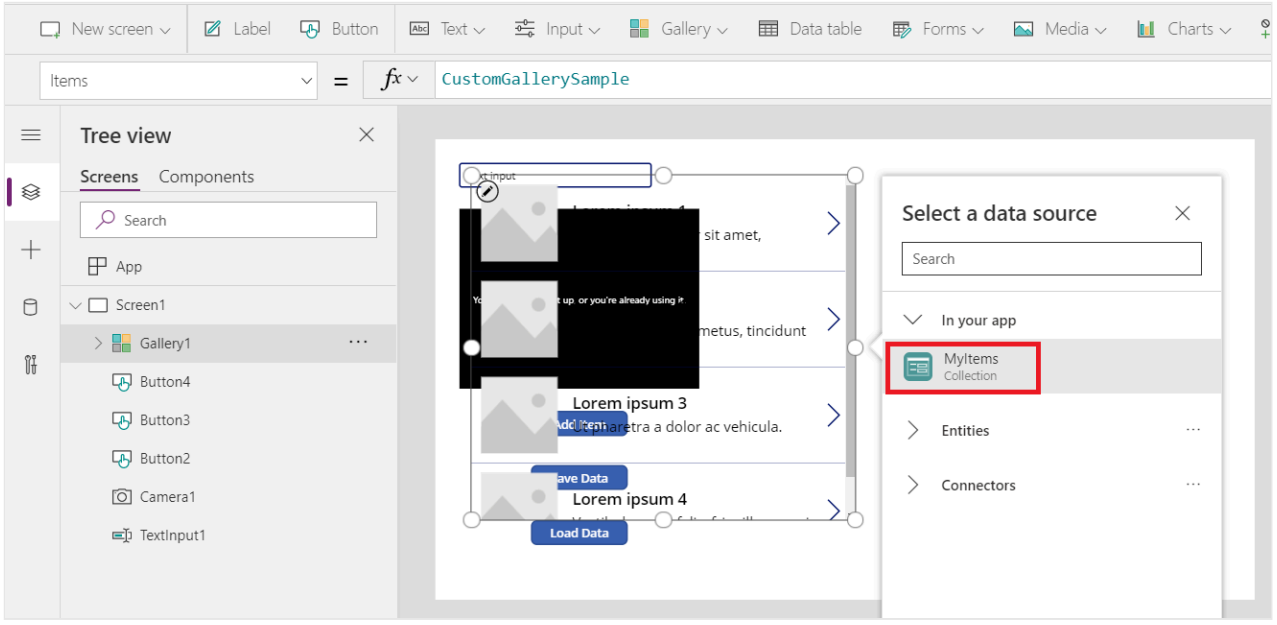
```
Power Fx

LoadData( MyItems, "LocalSavedItems" )
```



12. Add a Gallery control with a Vertical layout that includes a picture and text areas:

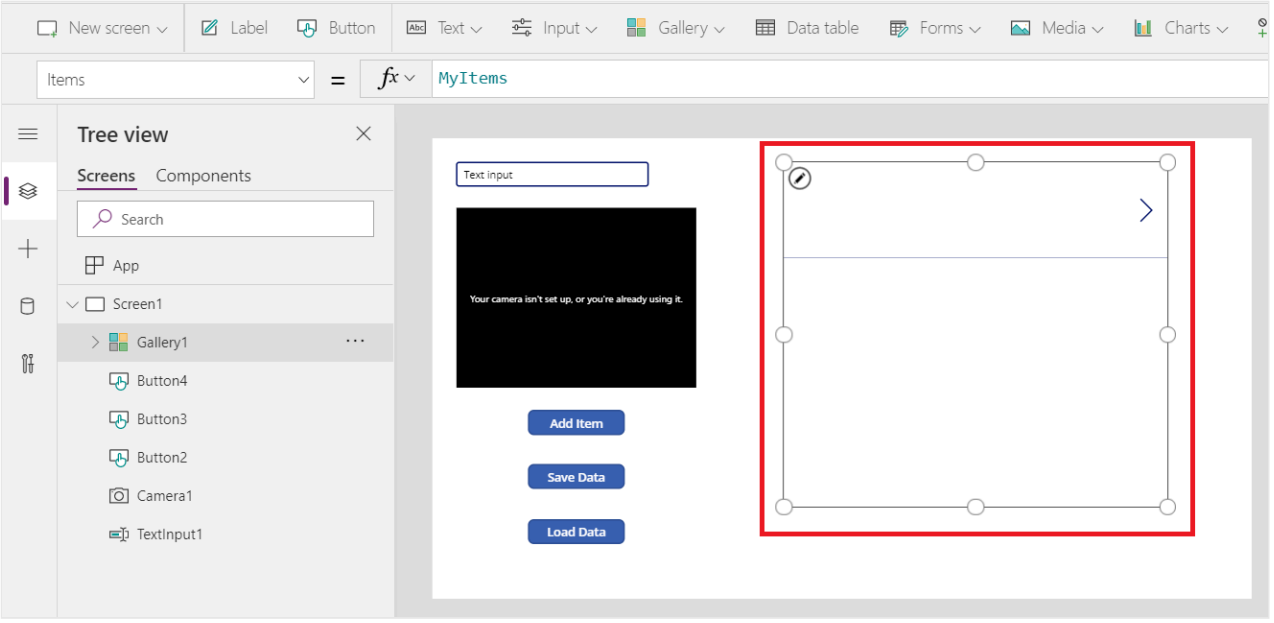


13. When prompted, select the **MyItems** collection as the data source for this gallery. This will set the **Items** property of the **Gallery** control:



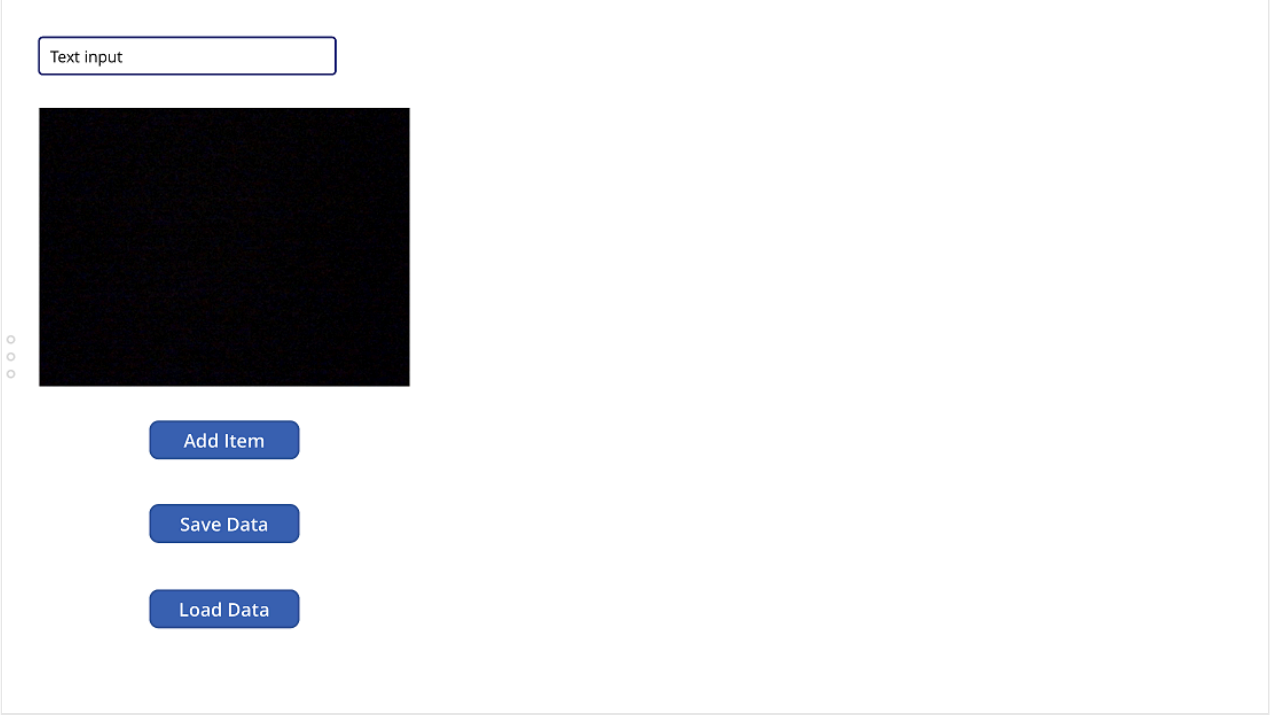
The image control in the gallery template should default its **Image** property to **ThisItem.Picture** and the label controls should both default their **Text** properties to **ThisItem.Item**. Check these formulas if after adding items in the following steps you don't see anything in the gallery.

14. Position the control to the right of the other controls:



15. Save your app. If it's the first time it has been saved, there's no need to publish it. If it's not the first time, publish the app after you save.

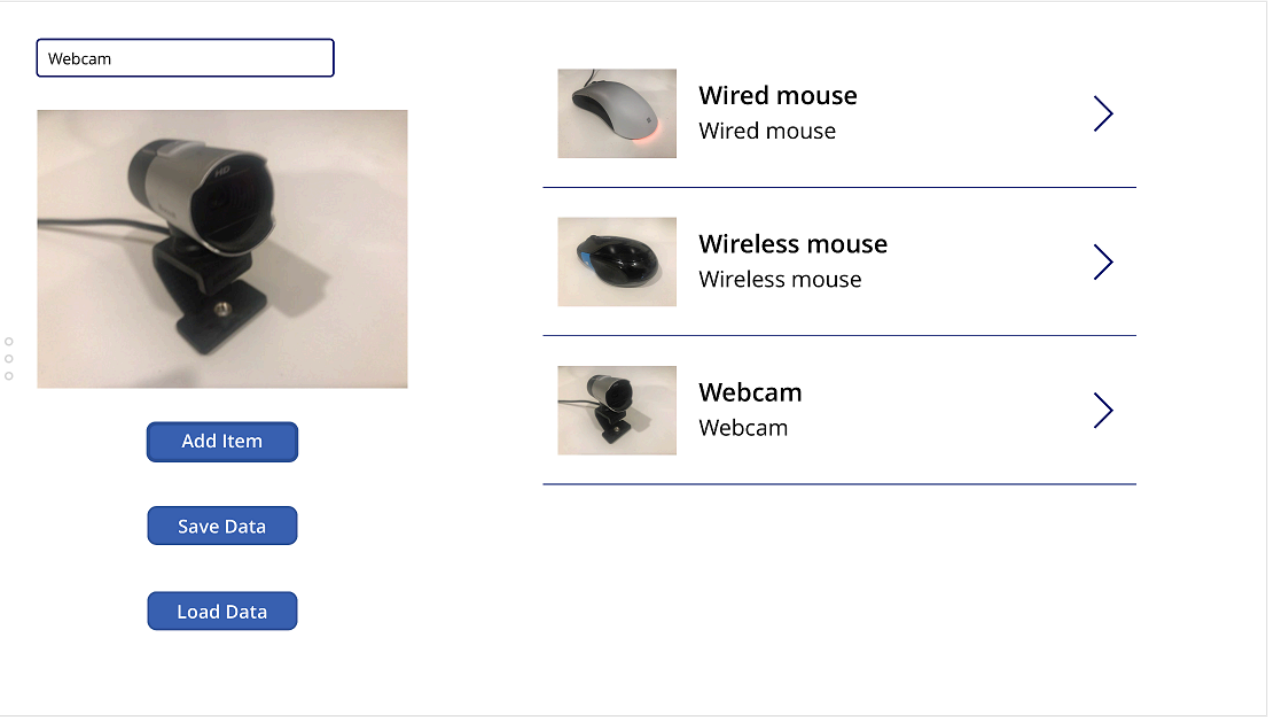
16. Open your app on a device such as a phone or tablet. **SaveData** and **LoadData** can't be used in Studio or in a web browser. Refresh your app list if you don't see your app immediately, it can take a few seconds for the app to appear on your device. Signing out and back in to your account can help too.



Once your app has been downloaded, you can disconnect from the network and run the app offline.

17. Enter the name and take a picture of an item.

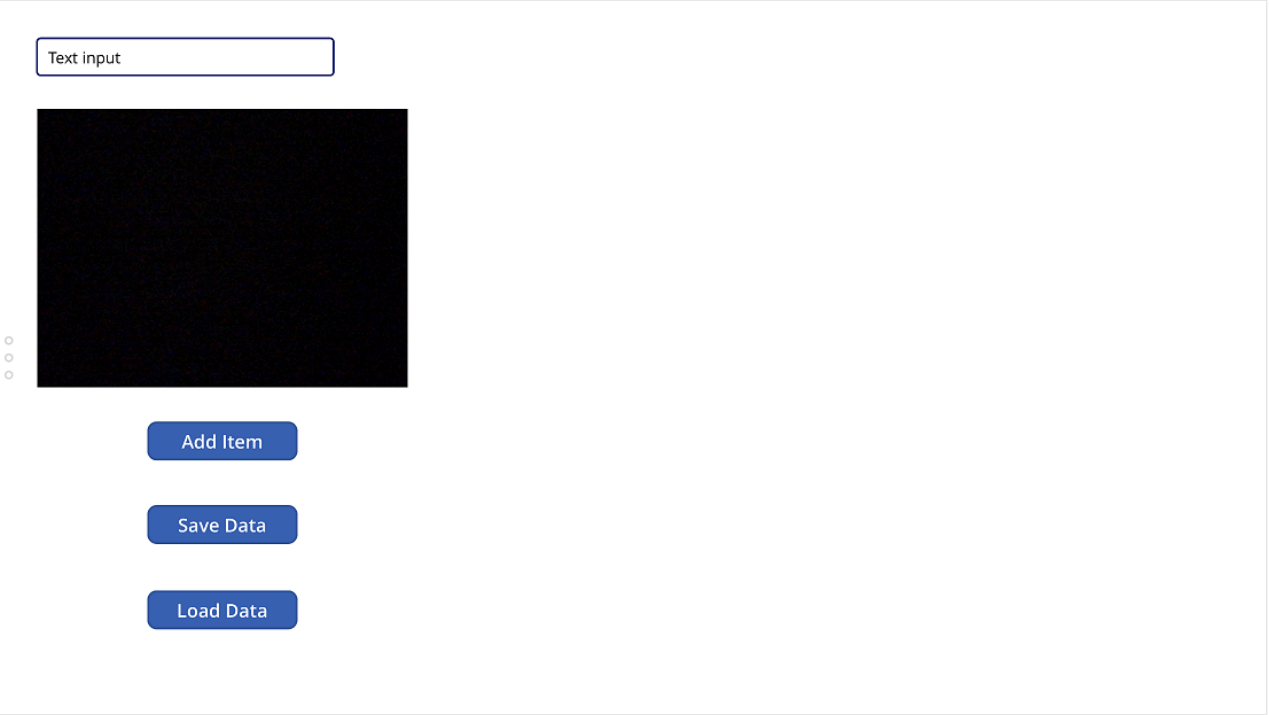
18. Select the **Add Item** button. Repeat adding items a couple of times to load up your collection.



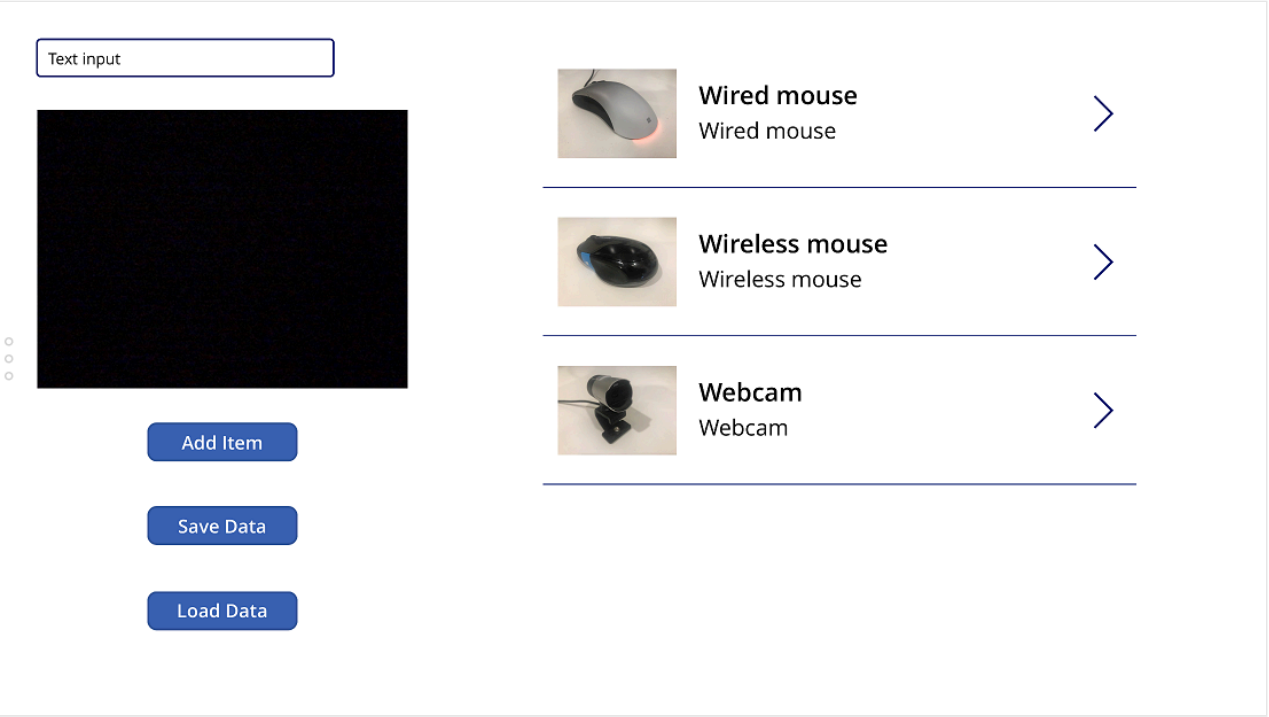
19. Select the **Save Data** button. This will save the data in your collection to your local device.

20. Close the app. Your collection in memory will be lost including all item names and pictures, but they'll still be there in the device's storage.

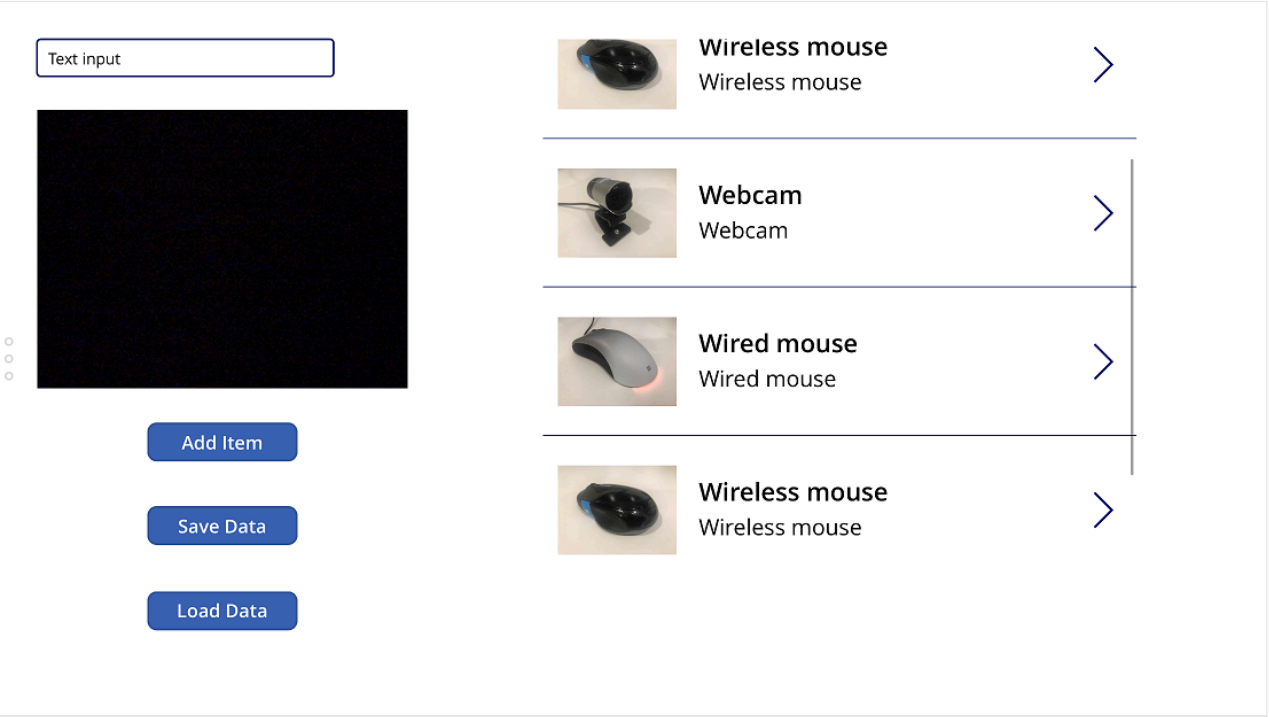
21. Launch the app again. The collection in memory will again show as empty in the gallery.



22. Select the **Load Data** button. The collection will be repopulated from the stored data on your device and your items will be back in the gallery. The collection was empty before this button calls the **LoadData** function; there was no need to call **Collect** or **ClearCollect** before loading the data from storage.



23. Select the **Load Data** button again. The stored data will be appended to the end of the collection and a scroll bar will appear on the gallery. If you would like to replace rather than append, use the **Clear** function first to clear out the collection before calling the **LoadData** function.



More advanced offline example

For a detailed example, see the article on simple offline capabilities.

Filter, Search, and LookUp functions

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Finds one or more records in a table.

Watch this video to learn how to use **Filter**, **Search**, and **LookUp** functions:

https://learn-video.azurefd.net/vod/player?id=55b2c144-b5ac-4d66-b398-fea03eb21f17&locale=en-us&embedUrl=%2Fpower-platform%2Fpower-fx%2Freference%2Ffunction-filter-lookup

> **Note**
>
> PAC CLI pac power-fx commands do not support the **Search** function.

Description

The **Filter** function finds records in a table that satisfy a formula. Use **Filter** to find a set of records that match one or more criteria and discard those records that don't.

The **LookUp** function finds the first record in a table that satisfies a formula. Use **LookUp** to find a single record that matches one or more criteria.

For both, the formula is evaluated for each record of the table. Records that result in *true* are included in the result. Besides the normal formula operators, you can use the **in** and **exactin** operators for substring matches.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

The **Search** function finds records in a table that contain a string in one of their columns. The string might occur anywhere within the column; for example, searching for "rob" or "bert" would find a match in a column that contains "Robert". Searching is case-insensitive. Unlike **Filter** and **LookUp**, the **Search** function uses a single string to match instead of a formula.

Filter and **Search** return a table that contains the same columns as the original table and the records that match the criteria. **LookUp** returns only the first record found, after applying a

formula to reduce the record to a single value. If no records are found, **Filter** and **Search** return an empty table, and **LookUp** returns *blank*.

Tables are a value in Power Apps, just like a string or number. They can be passed to and returned from functions. **Filter**, **Search**, and **LookUp** don't modify a table. Instead, they take a table as an argument and return a table, a record, or a single value from it. See working with tables for more details.

Delegation

When possible, Power Apps will delegate filter and sort operations to the data source and page through the results on demand. For example, when you start an app that shows a **Gallery** control filled with data, only the first set of records will be initially brought to the device. As the user scrolls, additional data is brought down from the data source. The result is a faster start time for the app and access to very large data sets.

However, delegation may not always be possible. Data sources vary on what functions and operators they support with delegation. If complete delegation of a formula isn't possible, the authoring environment will flag the portion that can't be delegated with a warning. When possible, consider changing the formula to avoid functions and operators that can't be delegated. The delegation list details which data sources and operations can be delegated.

If delegation is not possible, Power Apps will pull down only a small set of records to work on locally. Filter and sort functions will operate on a reduced set of records. What is available in the **Gallery** may not be the complete story, which could be confusing to users.

See the delegation overview for more information.

Syntax

Filter(Table*, *Formula1* [, *Formula2*, ...])

- *Table* - Required. Table to search.
- *Formula(s)* - Required. The formula by which each record of the table is evaluated. The function returns all records that result in **true**. You can reference columns within the table. If you supply more than one formula, the results of all formulas are combined with the **And** function.

Search(Table*, *SearchString*, *Column1* [, *Column2*, ...])

- *Table* - Required. Table to search.

- *SearchString* - Required. The string to search for. If *blank* or an empty string, all records are returned.
- *Column(s)* - Required. The names of columns within *Table* to search. If *SearchString* is found within the data of any of these columns as a partial match, the full record will be returned.

> **Note**
>
> In Power Apps prior to version 3.24042, column names for the **Search** function were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

LookUp(Table*, *Formula* [, *ReductionFormula*])

- *Table* - Required. Table to search. In the UI, the syntax is shown as *source* above the function box.
- *Formula* - Required. The formula by which each record of the table is evaluated. The function returns the first record that results in **true**. You can reference columns within the table. In the UI, the syntax is shown as *condition* above the function box.
- *ReductionFormula* - Optional. This formula is evaluated over the record that was found, and then reduces the record to a single value. You can reference columns within the table. If you don't use this parameter, the function returns the full record from the table. In the UI, the syntax is shown as *result* above the function box.

Examples

The following examples use the **IceCream** data source:

Flavor	Quantity	OnOrder
"Chocolate"	100	75
"Vanilla"	200	50
"Strawberry"	300	0
"Mint Chocolate"	60	100

Expand table

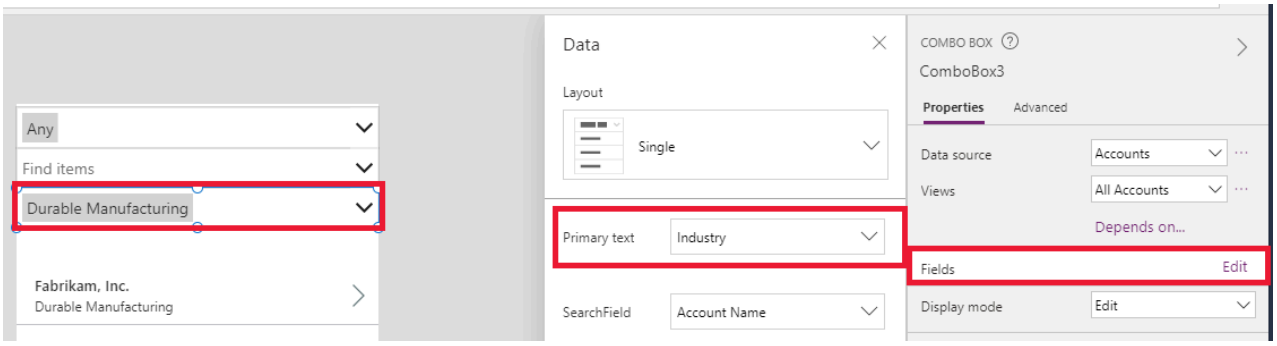
Formula	Description	Result
Filter(IceCream, OnOrder > 0)	Returns records where **OnOrder** is greater than zero.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Filter(IceCream, Quantity + OnOrder > 225)	Returns records where the sum of **Quantity** and **OnOrder** columns is greater than 225.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Strawberry"</td><td>300</td><td>0</td></tr></table>
Filter(IceCream, "chocolate" in Lower(Flavor))	Returns records where the word "chocolate" appears in the **Flavor** name, independent of uppercase or lowercase letters.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Filter(IceCream, Quantity < 10 && OnOrder < 20)	Returns records where the **Quantity** is less than 10 and **OnOrder** is less than 20. No records match these criteria, so an empty table is returned.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr></table>
Search(IceCream, "choc", Flavor)	Returns records where the string "choc" appears in the **Flavor** name, independent of uppercase or lowercase letters.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
Search(IceCream, "", Flavor)	Because the search term is empty, all records are returned.	<table><tr><th>Flavor</th><th>Quantity</th><th>OnOrder</th></tr><tr><td>"Chocolate"</td><td>100</td><td>75</td></tr><tr><td>"Vanilla"</td><td>200</td><td>50</td></tr><tr><td>"Strawberry"</td><td>300</td><td>0</td></tr><tr><td>"Mint Chocolate"</td><td>60</td><td>100</td></tr></table>
LookUp(IceCream, Flavor = "Chocolate", Quantity)	Searches for a record with **Flavor** equal to "Chocolate", of which there's one. For the first record that's found, returns the **Quantity** of that record.	100
LookUp(IceCream, Quantity > 150, Quantity + OnOrder)	Searches for a record with **Quantity** greater than 150, of which there are multiple. For the first record that's found, which is "Vanilla" **Flavor**, returns the sum of **Quantity** and **OnOrder** columns.	250
LookUp(IceCream, Flavor = "Pistachio", OnOrder)	Searches for a record with **Flavor** equal to "Pistachio", of which there are none. Because none is found, **Lookup** returns *blank*.	*blank*
LookUp(IceCream, Flavor = "Vanilla")	Searches for a record with **Flavor** equal to "Vanilla", of which there's one. Since no reduction formula was supplied, the entire record is returned.	{ Flavor: "Vanilla", Quantity: 200, OnOrder: 75 }

Filtering with choice columns

The following example uses the **Account** table in Microsoft Dataverse as data source. This example shows how to **Filter** list of accounts based on selected Combo box control values:

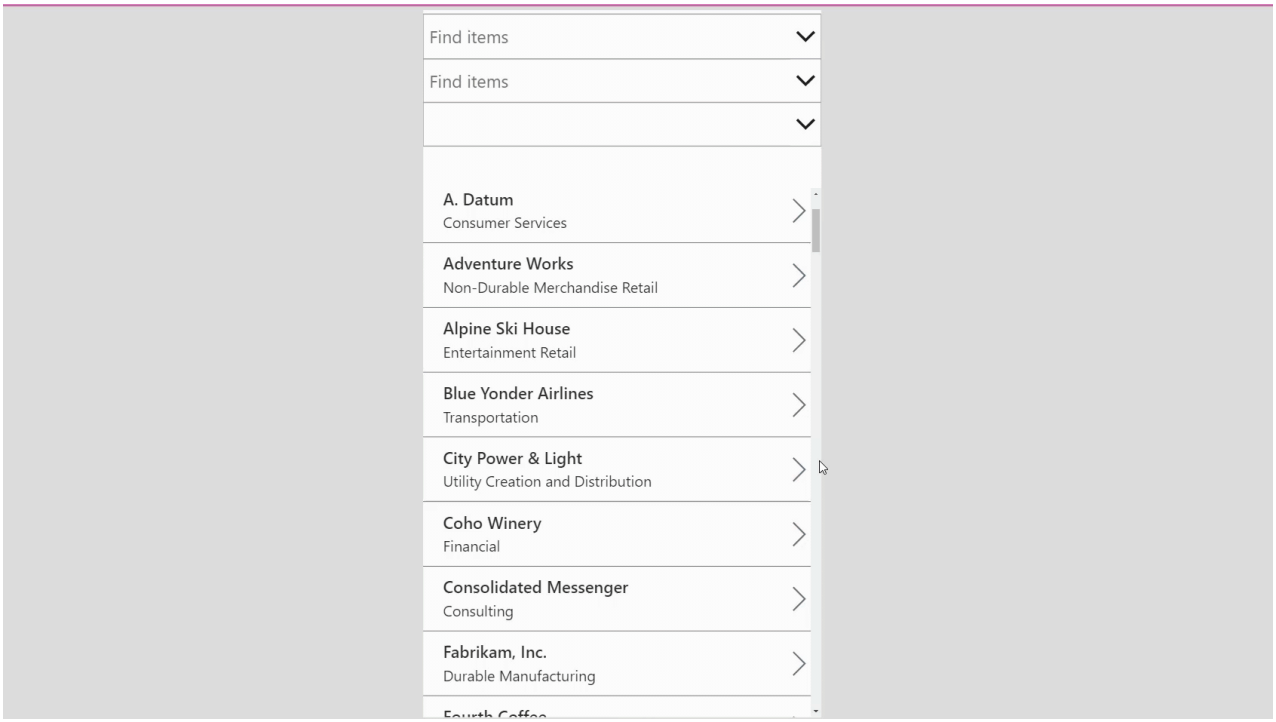
Step by step

1. Open a blank app.
2. Add a new screen by selecting the **New Screen** option.
3. On the **Insert** tab, select **Gallery** and then select **Vertical**.
4. On the **Properties** tab of the right-hand pane, open **Data Source** and then select **Accounts**.
5. (Optional) In the **Layout** list, select different options.
6. On the **Insert** tab, select **Input** and then select **Combo box**. Repeat the step to add two more combo box controls.
7. For each combo box control, on the **Properties** tab of the right-hand pane, open **Data Source** and then select **Accounts**. Select **Edit** next to **Fields** option and then select the **Primary text** and **SearchField** values. The **Primary text** should be the choices column you want to add to the combo box. Repeat the step for other two combo box controls.



8. Now select **Gallery** control and set the **Items** property to the following formula:

```
Filter(Accounts,
 'Industry' = ComboBox3.Selected.Industry Or
IsBlank(ComboBox3.Selected.Industry),
 'Relationship Type' = ComboBox2.Selected.'Relationship Type' Or
   IsBlank(ComboBox2.Selected.'Relationship Type'),
 'Preferred Method of Contact' = ComboBox1.Selected.'Preferred Method of
Contact' Or
   IsBlank(ComboBox1.Selected.'Preferred Method of Contact'))
```



Search user experience

The following examples use the **IceCream** data source:

In many apps, you can type one or more characters into a search box to filter a list of records in a large data set. As you type, the list shows only those records that match the search criteria.

The examples in the rest of this article show the results of searching a list, named **Customers**, that contain this data:

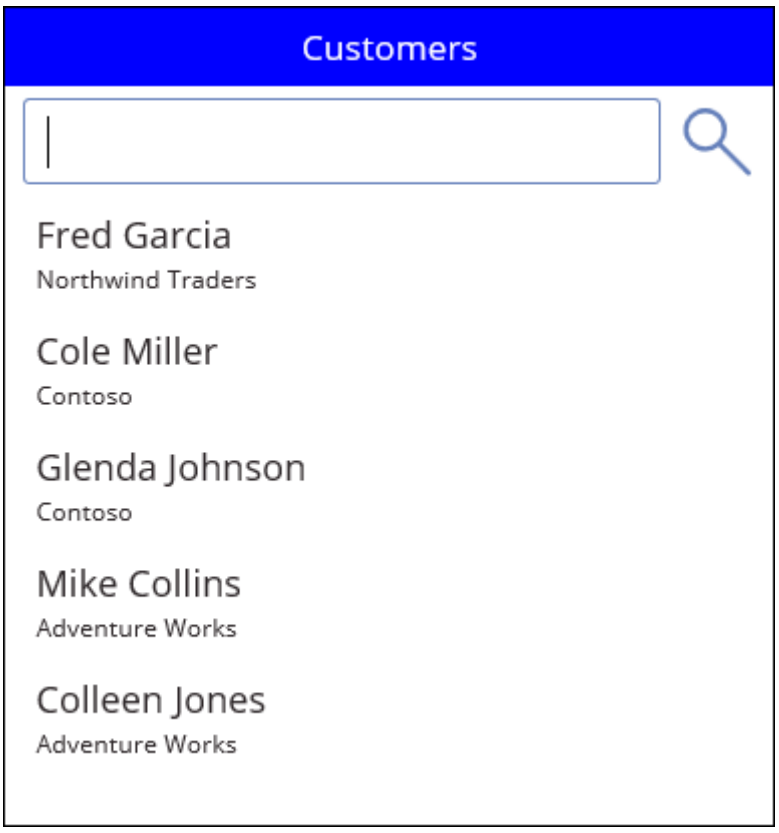
Name	Company
"Fred Garcia"	"Northwind Traders"
"Cole Miller"	"Contoso"
"Glenda Johnson"	"Contoso"
"Mike Collins"	"Adventure Works"
"Colleen Jones"	"Adventure Works"

To create this data source as a collection, create a **Button** control and set its **OnSelect** property to this formula:

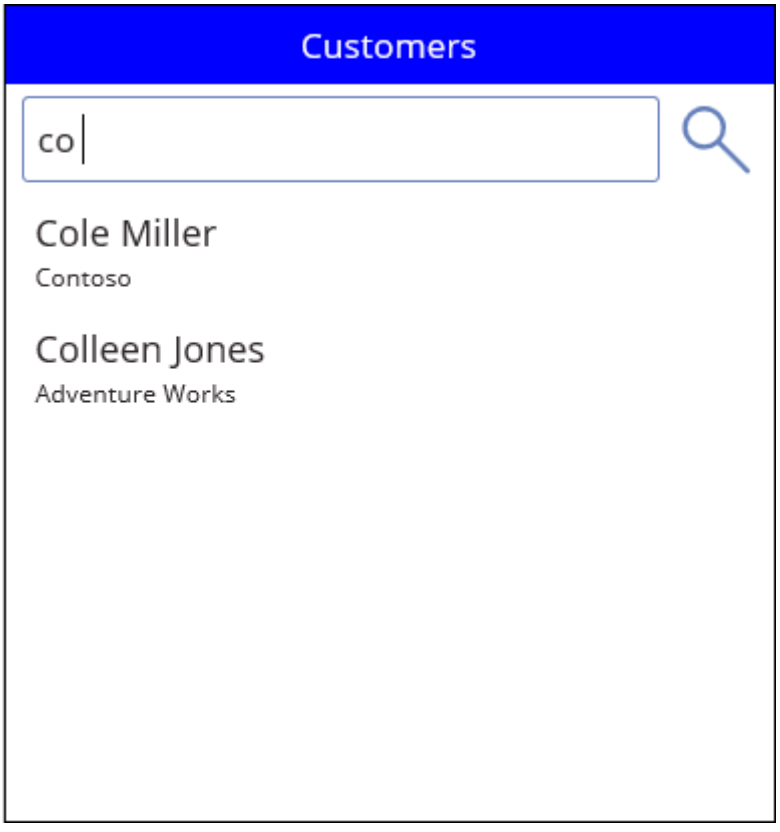
ClearCollect(Customers, Table({ Name: "Fred Garcia", Company: "Northwind Traders" }, { Name: "Cole Miller", Company: "Contoso" }, { Name: "Glenda Johnson", Company: "Contoso" }, { Name: "Mike Collins", Company: "Adventure Works" }, { Name: "Colleen Jones", Company: "Adventure Works" }))

As in this example, you can show a list of records in a **Gallery control** at the bottom of a screen. Near the top of the screen, you can add a **Text input** control, named **SearchInput**, so

that users can specify which records interest them.

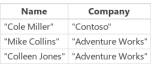


As the user types characters in **SearchInput**, the results in the gallery are automatically filtered. In this case, the gallery is configured to show records for which the name of the customer (not the name of the company) starts with the sequence of characters in **SearchInput**. If the user types **co** in the search box, the gallery shows these results:



To filter based on the **Name** column, set the **Items** property of the gallery control to one of these formulas:

Expand table

Formula	Description	Result
Filter(Customers, StartsWith(Name, SearchInput.Text))	Filters the **Customers** data source for records in which the search string appears at the start of the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones** and **Cole Miller**. The gallery doesn't show **Mike Collins** because the **Name** column for that record doesn't start with the search string.	
Filter(Customers, SearchInput.Text in Name)	Filters the **Customers** data source for records in which the search string appears anywhere in the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones, Cole Miller,** and **Mike Collins** because the search string appears somewhere in the **Name** column of all of those records.	
Search(Customers, SearchInput.Text, Name)	Similar to using the **in** operator, the **Search** function searches for a match anywhere within the **Name** column of each record. You must enclose the column name in double quotation marks.	

You can expand your search to include the **Company** column and the **Name** column:

Expand table

Formula	Description	Result
`Filter(Customers, StartsWith(Name, SearchInput.Text) || StartsWith(Company, SearchInput.Text) )`	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column starts with the search string (for example, **co**). The || operator is *true* if either **StartsWith** function is *true*.	
`Filter(Customers, SearchInput.Text in Name || SearchInput. Text in Company)`	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it.	
`Search(Customers, SearchInput.Text, Name, Company)`	Similar to using the **in** operator, the **Search** function searches the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it. The **Search** function is easier to read and write	

Formula	Description	Result
	than **Filter** if you want to specify multiple columns and multiple **in** operators.	

Last updated on 02/06/2025

Day, Month, Year, Hour, Minute, Second, and Weekday functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns individual components of a Date/Time value.

Description

The **Day** function returns the day component of a Date/Time value, ranging from 1 to 31.

The **Month** function returns the month component of a Date/Time value, ranging from 1 to 12.

The **Year** function returns the year component of a Date/Time value, starting with 1900.

The **Hour** function returns the hour component of a Date/Time value, ranging from 0 (12:00 AM) to 23 (11:00 PM).

The **Minute** function returns the minute component of a Date/Time value, ranging from 0 to 59.

The **Second** function returns the second component of a Date/Time value, ranging from 0 to 59.

The **Weekday** function returns the weekday of a Date/Time value. By default, the result ranges from 1 (Sunday) to 7 (Saturday). You can specify a different range with a Microsoft Excel Weekday function code or a StartOfWeek enumeration value:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	StartOfWeek.Sunday	Numbers 1 (Sunday) through 7 (Saturday). Default.
2, 11	StartOfWeek.Monday	Numbers 1 (Monday) through 7 (Sunday).
3	StartOfWeek.MondayZero	Numbers 0 (Monday) through 6 (Sunday).
12	StartOfWeek.Tuesday	Numbers 1 (Tuesday) through 7 (Monday).

Excel code	StartOfWeek enumeration	Description
13	**StartOfWeek.Wednesday**	Numbers 1 (Wednesday) through 7 (Tuesday).
14	**StartOfWeek.Thursday**	Numbers 1 (Thursday) through 7 (Wednesday).
15	**StartOfWeek.Friday**	Numbers 1 (Friday) through 7 (Thursday).
16	**StartOfWeek.Saturday**	Numbers 1 (Saturday) through 7 (Friday).

All functions return a number.

See working with dates and times for more information.

Syntax

Day(*DateTime*)
Month(*DateTime*)
Year(*DateTime*)
Hour(*DateTime*)
Minute(*DateTime*)
Second(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on.

Weekday(*DateTime* [, *WeekdayFirst*])

- *DateTime* - Required. Date/Time value to operate on.
- *WeekdayFirst* - Optional. Excel code specifying which day starts the week. If not supplied, 1 (Sunday first) is used.

Examples

For the following example, the current time is **3:59:37 PM** on **Thursday, April 9, 2015**.

Expand table

Formula	Description	Result
Year(Now())	Returns the year component of the current time and date.	2015
Month(Now())	Returns the month component of the current time and date.	4

Formula	Description	Result
Day(Now())	Returns the day component of the current time and date.	9
Hour(Now())	Returns the hour component of the current time and date.	15
Minute(Now())	Returns the minute component of the current time and date.	59
Second(Now())	Returns the second component of the current time and date.	37
Weekday(Now())	Returns the weekday component of the current time and date, using the default start of the week as Sunday.	5
Weekday(Now(), 14)	Returns the weekday component of the current time and date, using an Excel code to specify the start of the week as Thursday.	1
Weekday(Now(), StartOfWeek.Wednesday)	Returns the weekday component of the current time and date, using a **StartOfWeek** enumeration to specify the start of the week as Wednesday.	2

Feedback

Was this page helpful? Yes No

Provide product feedback

Select function

06/20/2025

Applies to: Canvas apps

Simulates a select action on a control, causing the **OnSelect** formula to be evaluated.

Description

The **Select** function simulates a select action on a control as if the user had clicked or tapped the control. As a result, the **OnSelect** formula on the target control is evaluated.

Use **Select** to propagate a select action to a parent control. This type of propagation is the default behavior in, for example, galleries. By default, the **OnSelect** property of any control in a Gallery control is set to **Select(Parent)**. That way, you can set the value of the **OnSelect** property of the gallery control itself, and that formula will be evaluated regardless of where in the gallery a user might click or tap.

If you want one or more controls in the gallery to perform different actions from the gallery itself, set the **OnSelect** property for those controls to something other than the default value. You can leave the default values for the **OnSelect** properties of most controls in the gallery if you want them to perform the same action as the gallery itself.

Select queues the target **OnSelect** for later processing, which may happen after the current formula has finished being evaluated. **Select** doesn't cause the target **OnSelect** to evaluate immediately, nor does **Select** wait for **OnSelect** to finish being evaluated.

You can't use **Select** across screens.

You can use **Select** only with controls that have an **OnSelect** property.

You can use **Select** only in behavior formulas.

A control can't **Select** itself directly or indirectly through other controls.

The select function can also be used with a gallery. For example, it can be used to specify the row or column to select in a gallery and the control to select within that row or column of the gallery. When you select a row or column, the gallery selection changes and the **OnSelect** formula on the gallery control is evaluated. If a control within the row or column is provided, the **OnSelect** formula for the child control will be evaluated.

Syntax

Select(*Control*)

- *Control* – Required. The control to select on behalf of the user.

Select(*Control, Row or column, Child Control*)

- *Control* – Required. The control to select on behalf of the user.
- *Row or column* – Not required. The number of row or column (starting with 1) in a gallery control to select on behalf of the user.
- *Child Control* - Not required. The child control of the control identified in the 'control' parameter to select.

Examples

- *Button*

  `Select(button1)`

- *Gallery*

  `Select(Gallery1, 1)`

  Simulates a user selecting row 1 or column 1 in Gallery1.

- *Gallery*

  `Select(Gallery1, 1, ChildControl1)`

  Simulates a user selecting ChildConttrol1 in row 1 or column 1 of Gallery1.

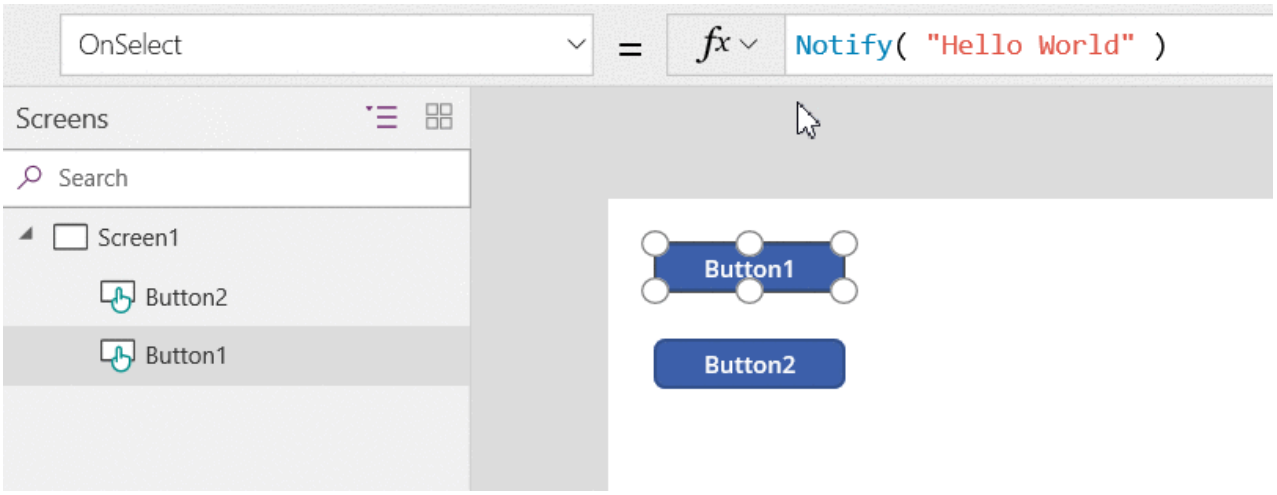
Basic usage

1. Add a Button control, and rename it **Button1** if it has a different name.
2. Set the **OnSelect** property of **Button1** to this formula:

   Notify("Hello World")
3. On the same screen, add a second **Button** control, and set its **OnSelect** property to this formula:

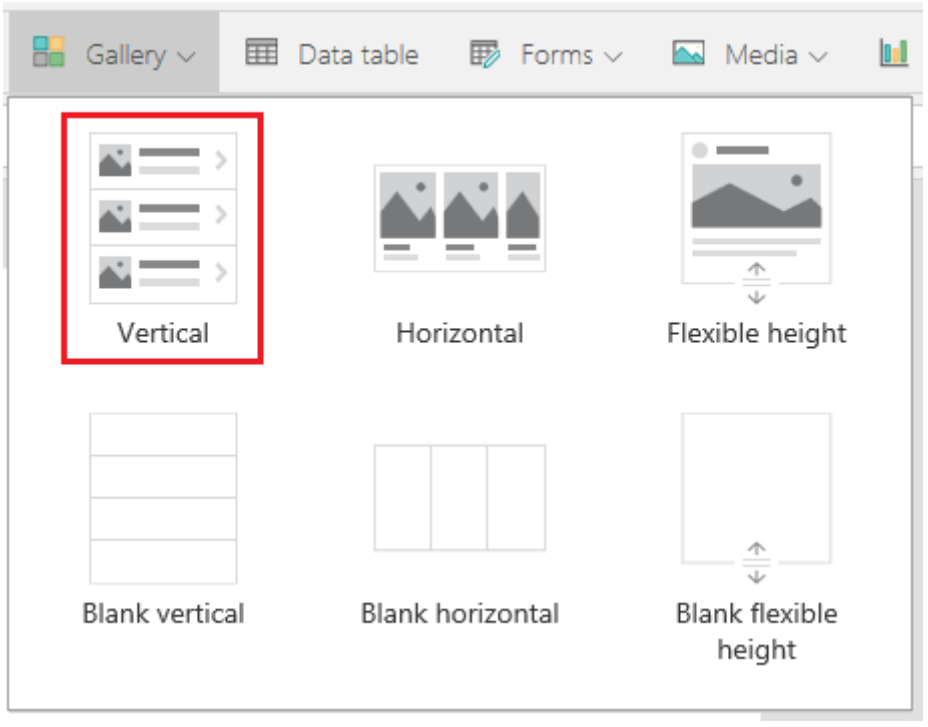
   Select(Button1)
4. While holding down the Alt key, select the second button.

A notification appears across the top of your app. The **OnSelect** property of **Button1** generated this notification.



Gallery control

1. Add a vertical Gallery control that contains other controls.



2. Set the **OnSelect** property of the gallery to this formula:

   Notify("Gallery Selected")

3. While holding down the Alt key, click or tap the background of the gallery or any control in the gallery.

   All actions will show the **Gallery Selected** notification at the top of the app.

Use the gallery's **OnSelect** property to specify the default action to take when the user clicks or taps an item in the gallery.

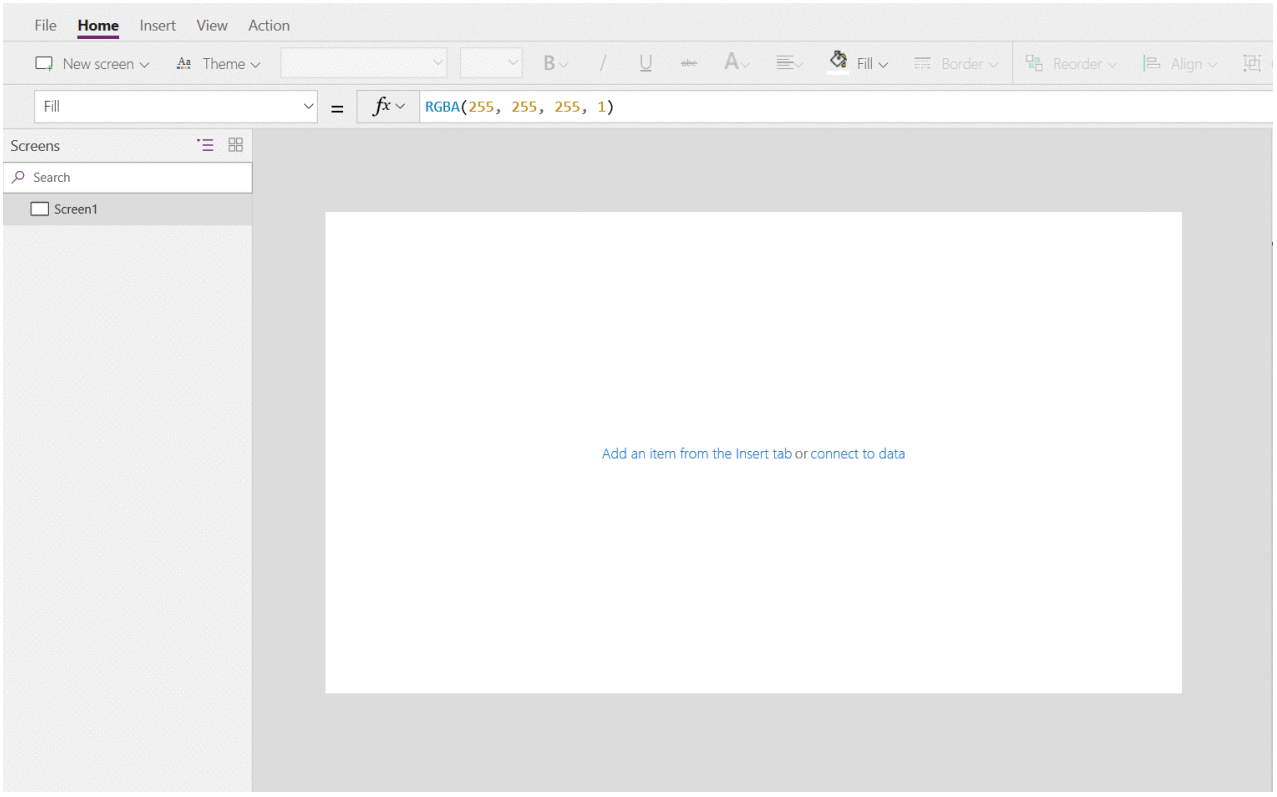
4. Set the **OnSelect** property of the image control to this formula:

   Notify("Image Selected", NotificationType.Success)

5. While holding down the Alt key, click or tap the various elements of the gallery.

   When you click or tap any control in the gallery except the image, **Gallery Selected** appears as before. When you click or tap the image, **Image Selected** appears.

   Use individual controls in the gallery to take actions that differ from the gallery's default action.



6. On the same screen, add a **Button** control, and set its **OnSelect** property to this formula:

   Select(Gallery1,2,Image1)

7. While holding down the Alt key, select the button.

   A **Image Selected** notification appears across the top of your app. The button click simulated selecting the image in row 2 of the gallery.

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

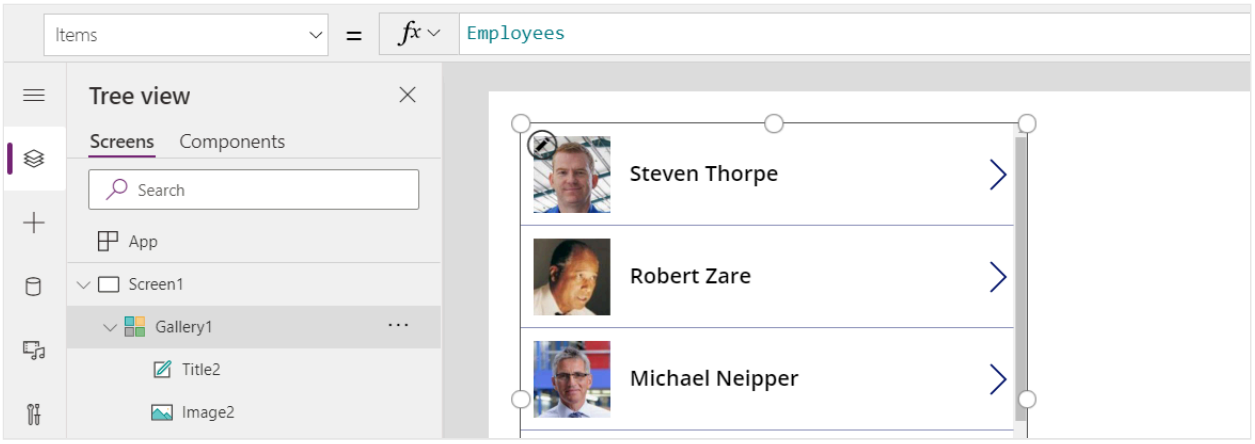
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

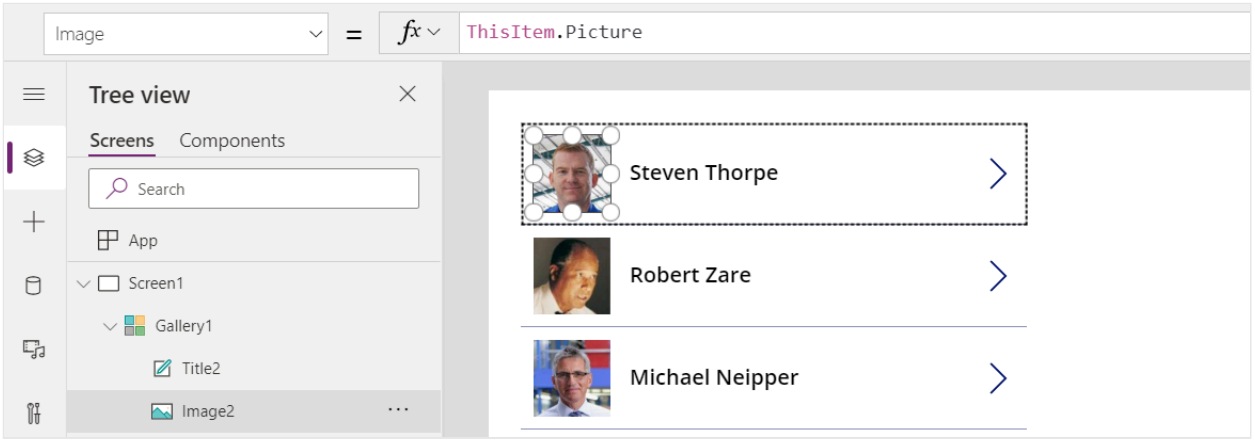
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

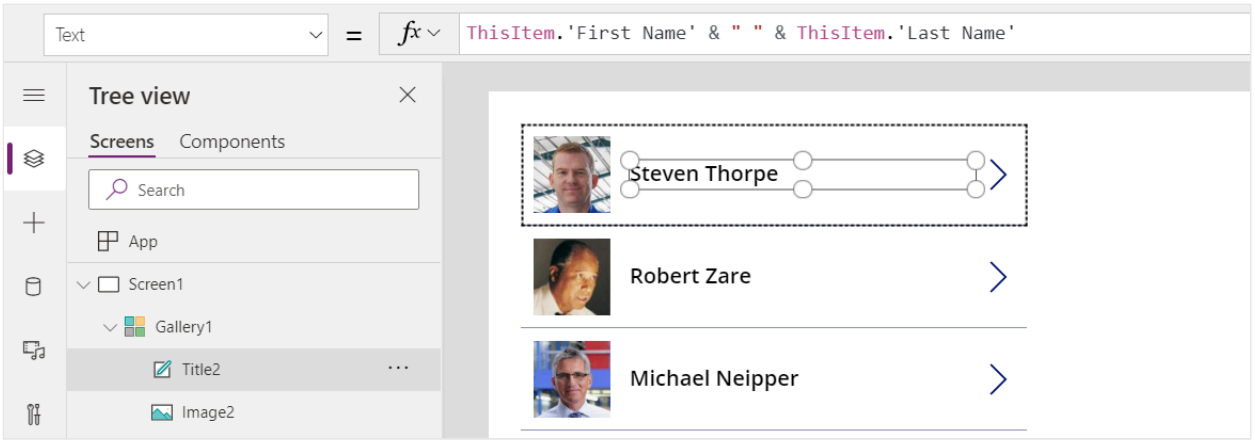
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

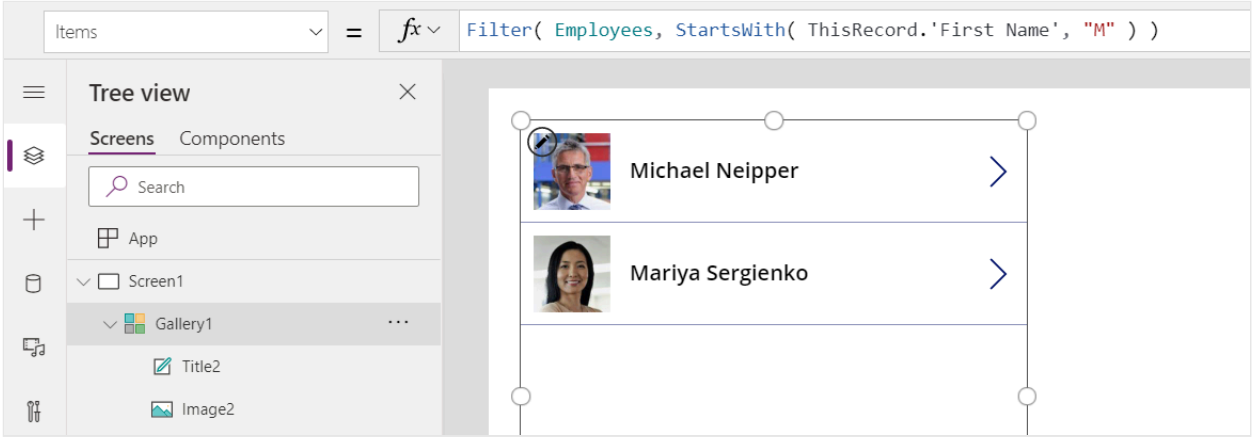


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

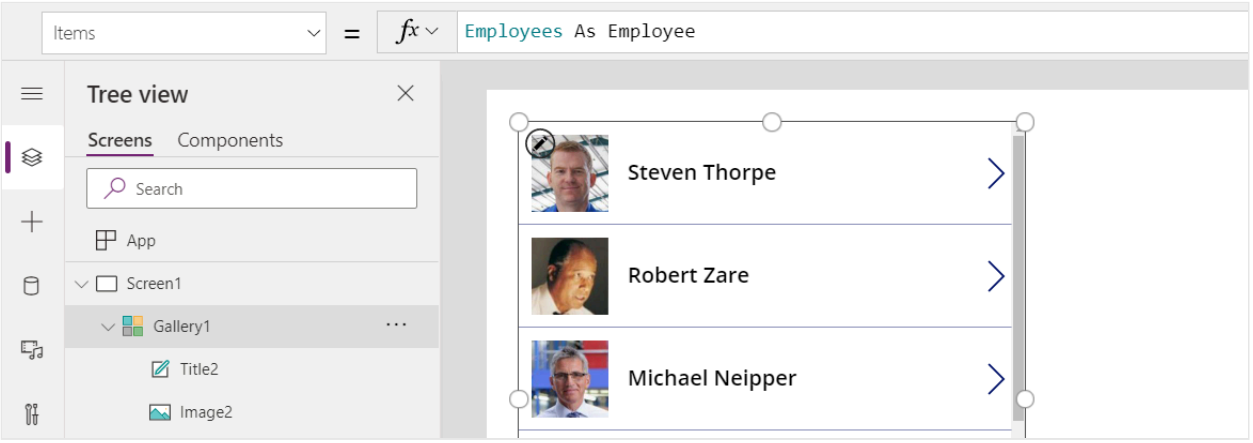
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

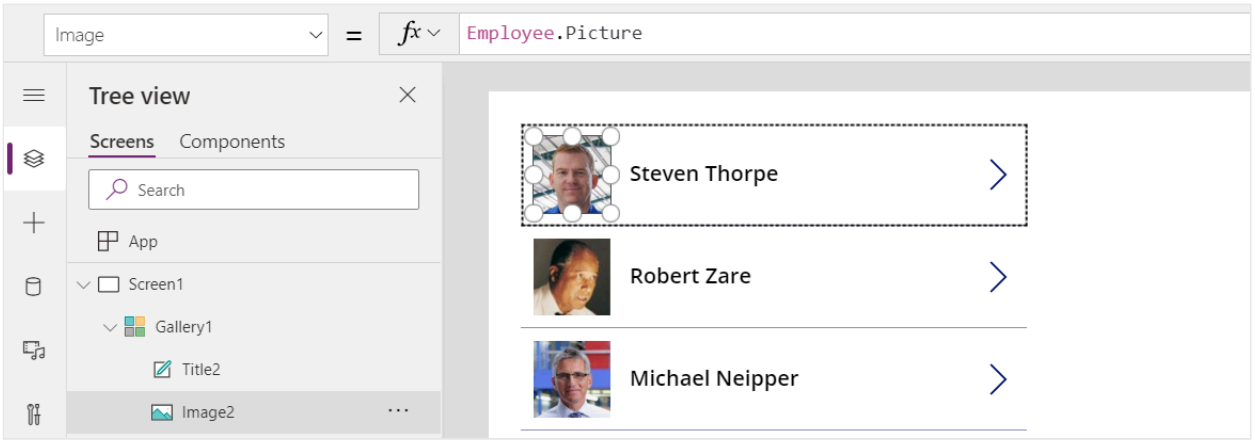
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

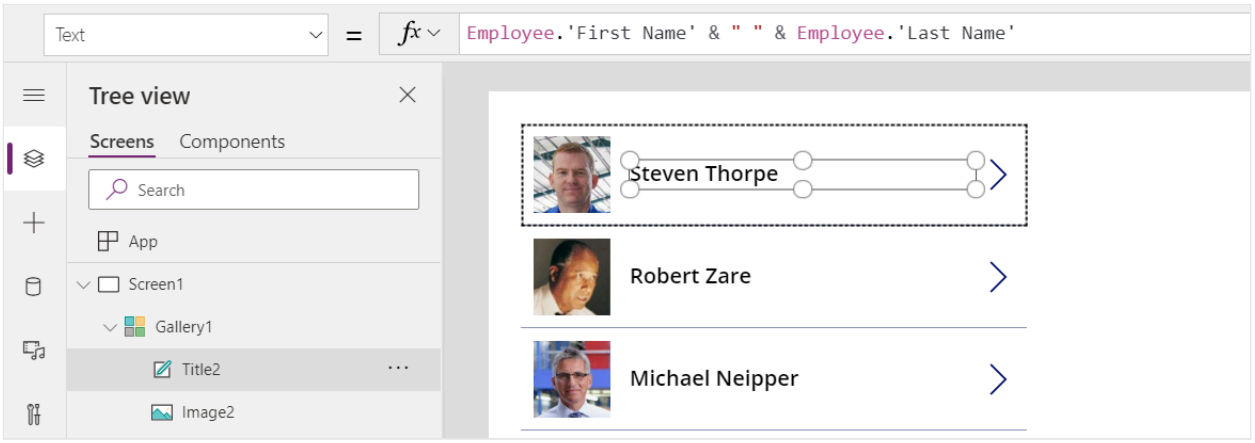
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

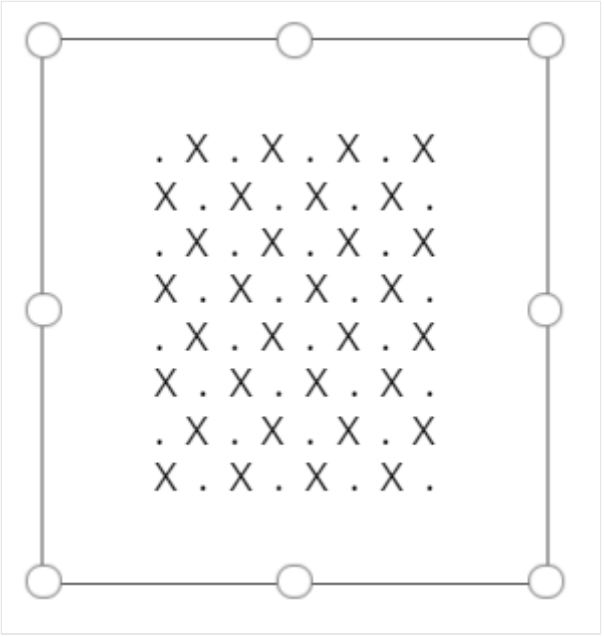
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



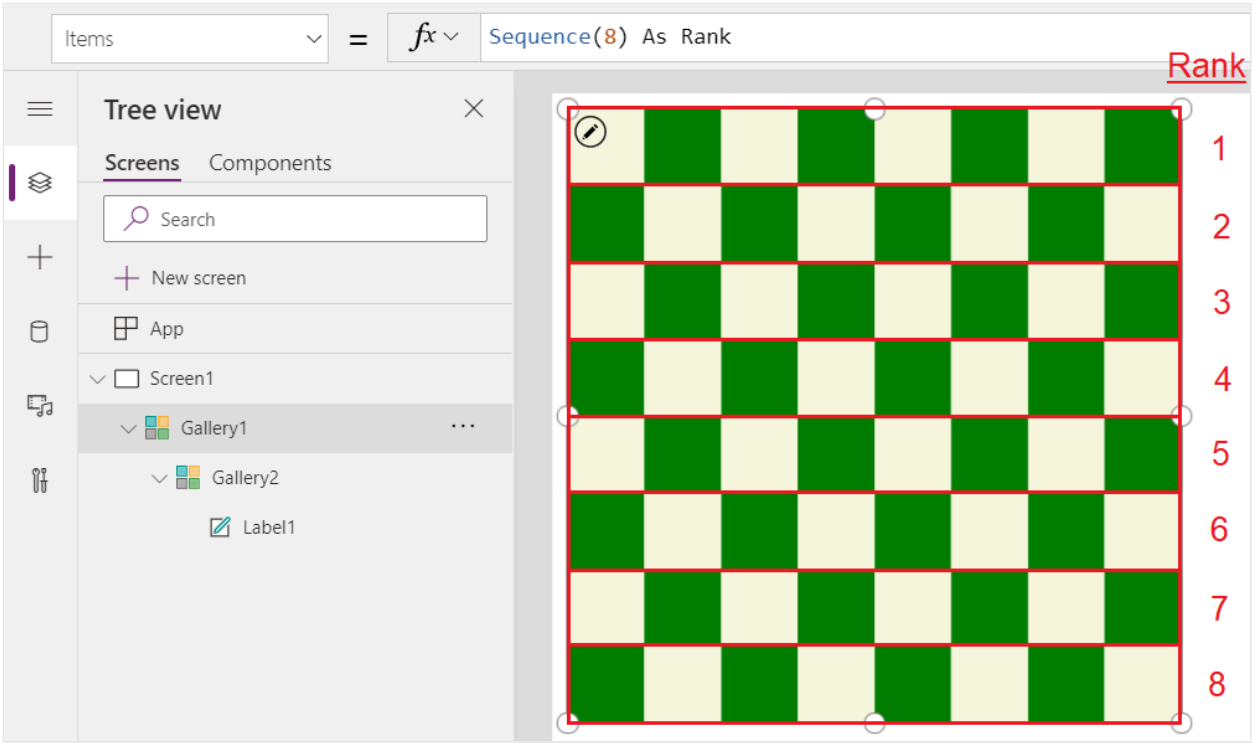
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

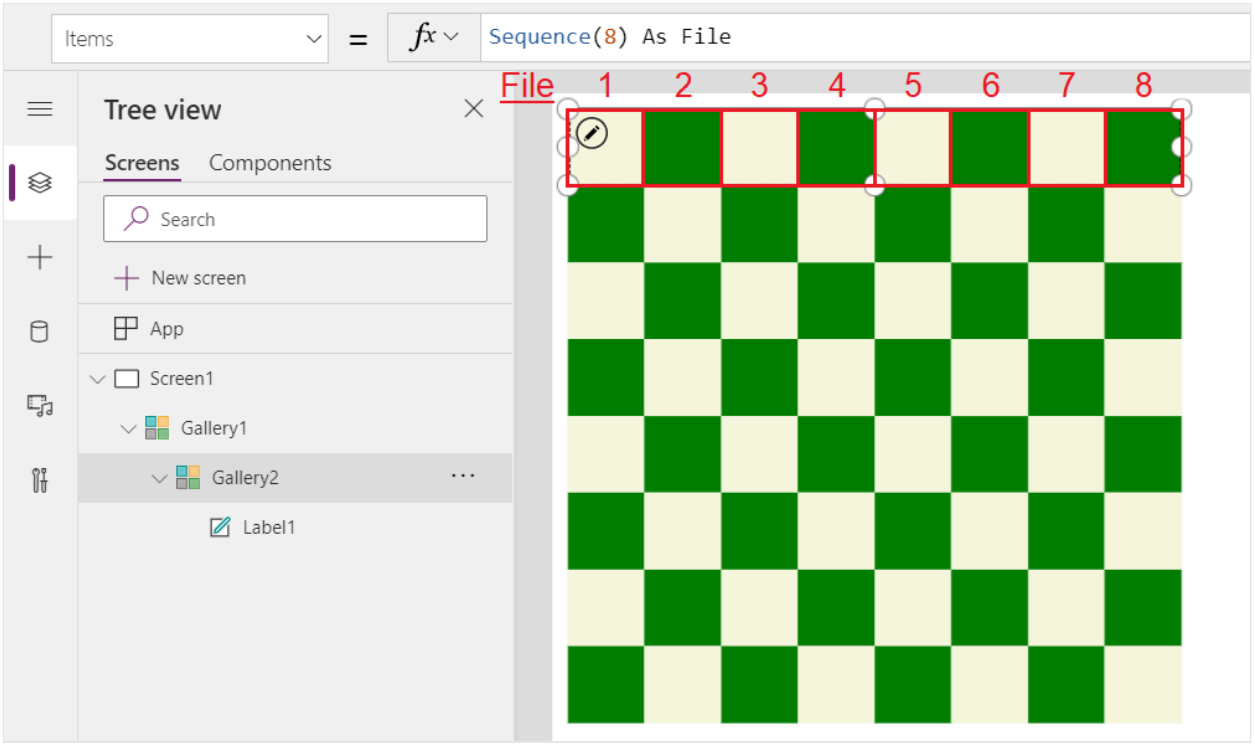
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

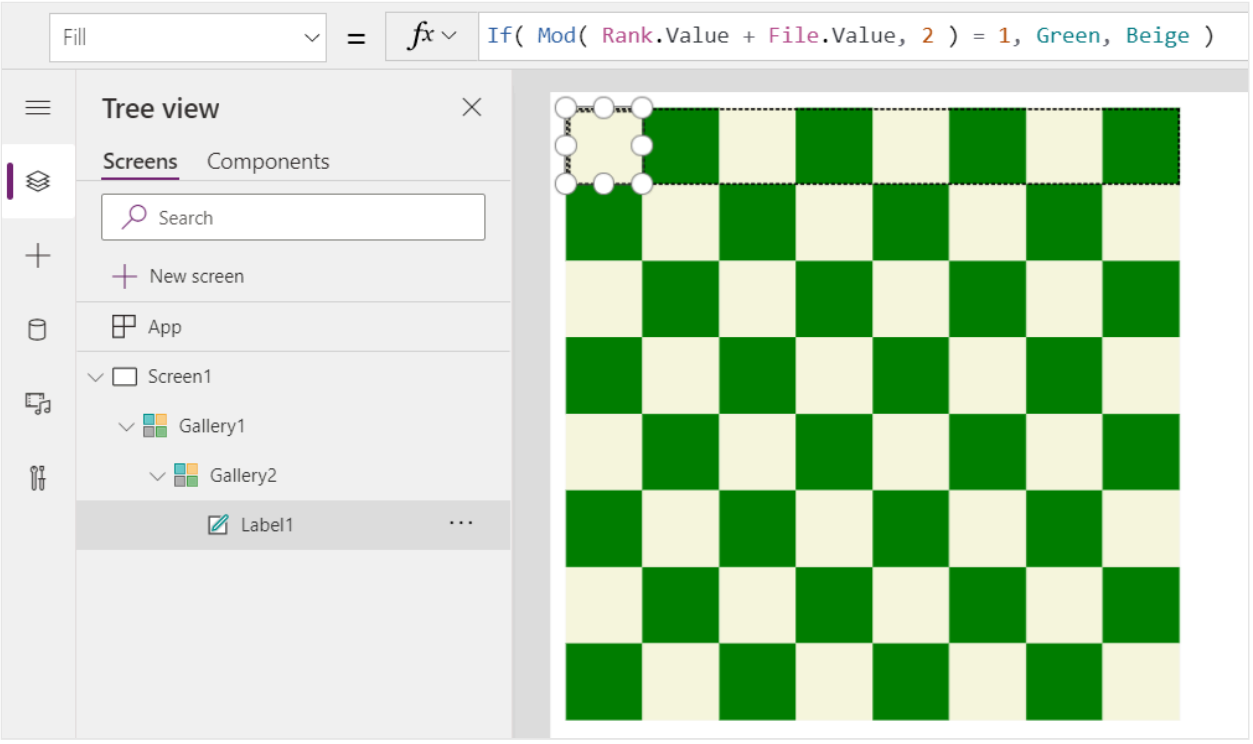
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

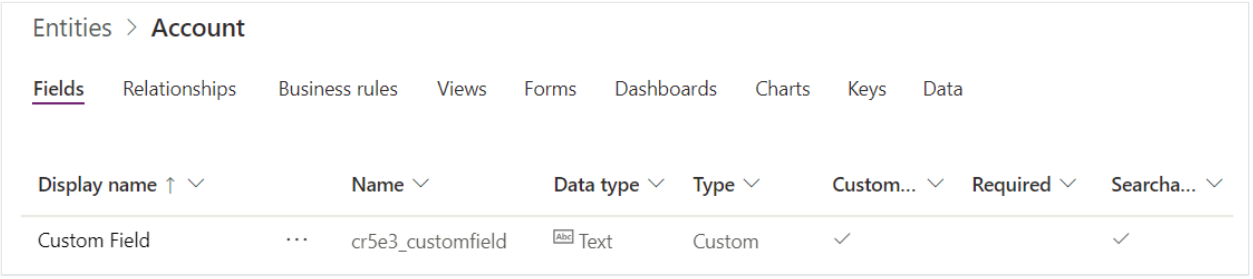
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

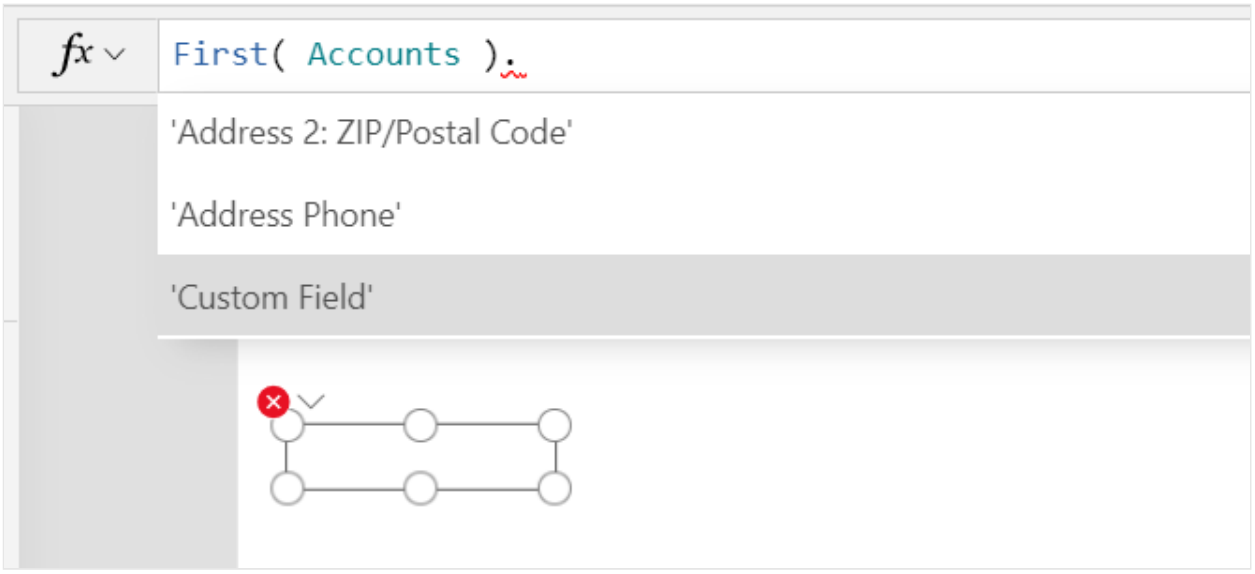
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

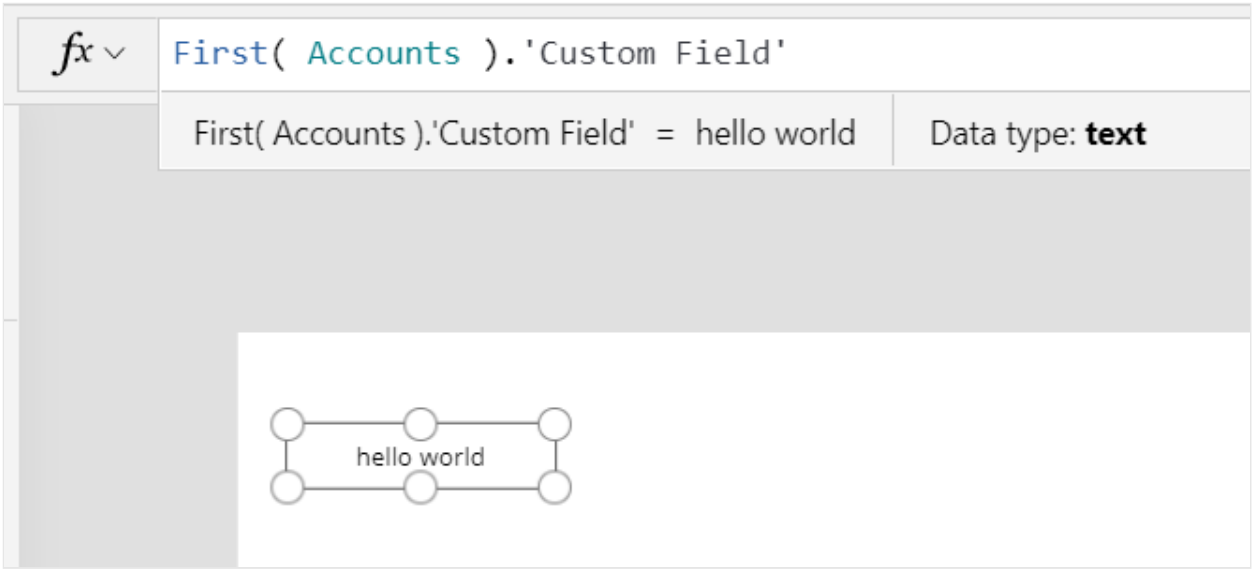
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



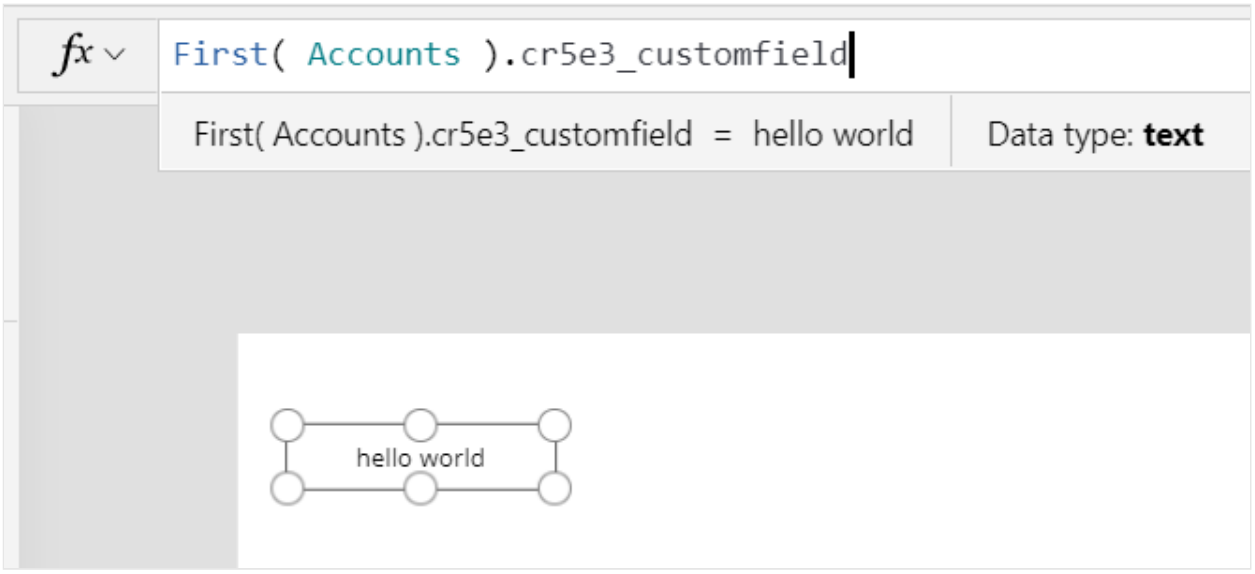
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

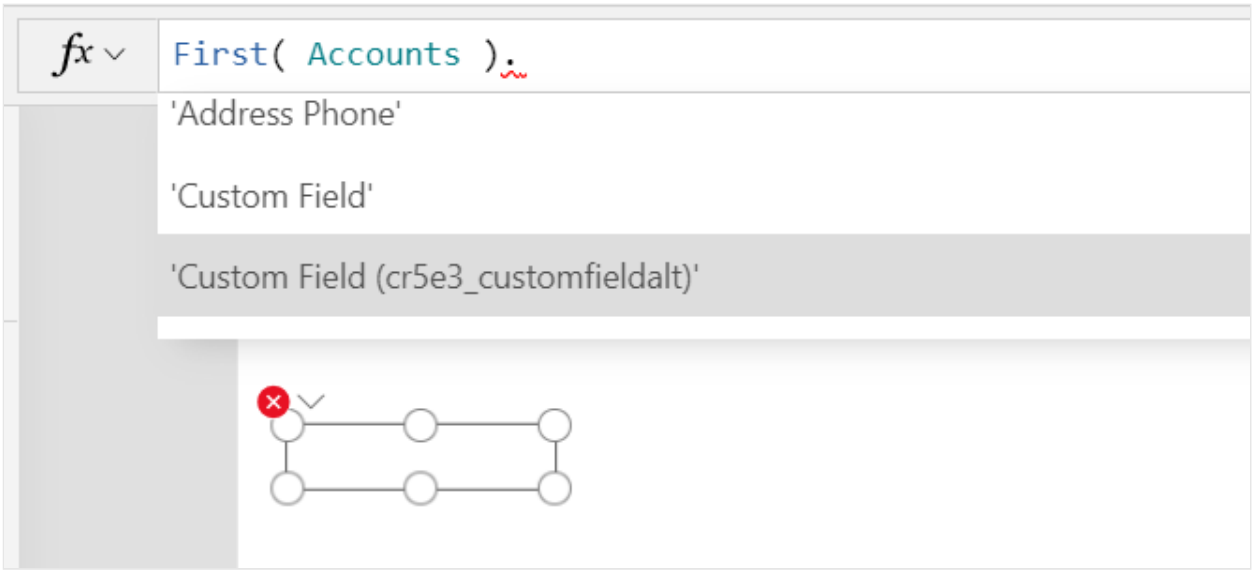


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

Set function

Article • 03/22/2024

Applies to: ✅ Canvas apps ✅ Cards ✅ Dataverse low-code plug-ins ✅ Power Platform CLI

Sets the value of a global variable.

Overview

Use the **Set** function to set the value of a global variable, which temporarily holds a piece of information, such as the number of times the user has selected a button or the result of a data operation.

Global variables are available throughout your app on all screens. These are the simplest kind of variables and fill the needs of most situations. There are also context variables which are scoped to a single screen and collections that allow row level modifications to tables. For more information about these other options, review Understand variables.

Power Apps are based on formulas that automatically recalculate as the user interacts with an app. Any formulas that depend on a variable will automatically update when it changes. However, the variable won't be automatically updated if the value of the formula used in the **Set** function changes. This requires the app maker to manually update the variable, which can be error prone and harder for others to understand. Before you use a variable, review Understand variables.

Description

Global variables are implicitly created by using the **Set** function. No explicit declaration is required. If you remove all the **Set** functions for a global variable, that global variable will cease to exist. To clear a variable, set its value to the result of the **Blank** function.

You can see your variables' values, definitions, and uses with the Variables view under the **File** menu in Power Apps Studio.

As the examples later in this topic show, global variables can hold several kinds of information, including these:

- a single value
- a record
- a table

- an object reference
- any result from a formula

A global variable holds its value until the app is closed. Once closed, the global variable's value will be lost and must be recreated when the app is loaded again.

Global variables cannot use the same name as an existing collection or control. It can use the same name as a context variable. To disambiguate between the two, use the disambiguation operator.

Set has no return value, and you can use it only within a behavior formula.

Syntax

Set(*VariableName*, *Value*)

- *VariableName* - Required. The name of a global variable to create or update.
- *Value* - Required. The value to assign to the context variable.

Examples

Expand table

Formula	Description	Result
Set(Counter, 1)	Creates or modifies the global variable **Counter**, setting its value to **1**.	**Counter** has the value **1**. You can reference that variable by using the name **Counter** in a formula on any screen.
Set(Counter, 2)	Sets the value of the **Counter** global variable from the previous example to **2**.	**Counter** has the value **2**.
Set(Counter, Counter + 1)	Increments the value of the **Counter** global variable from the previous example to **3**.	**Counter** has the value **3**.
Set(Name, "Lily")	Creates or modifies the global variable **Name** setting its value to **Lily**.	**Name** has the value **Lily**.

Formula	Description	Result
Set(Person, { Name: "Milton", Address: "1 Main St" })	Creates or modifies the global variable **Person**, setting its value to a record. The record contains two columns, named **Name** and **Address**. The value of the **Name** column is **Milton**, and the value of the **Address** column is **1 Main St**.	**Person** has the value of record **{ Name: "Milton", Address: "1 Main St" }**. Reference this record as a whole with the name **Person**, or reference an individual column of this record with **Person.Name** or **Person.Address**.
Set(Person, Patch(Person, {Address: "2 Main St" }))	Works with the **Patch** function to update the **Person** global variable by setting the value of the **Address** column to **2 Main St**.	**Person** now has the value of record **{ Name: "Milton", Address: "2 Main St" }**.

SetFocus function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Moves input focus to a specific control.

Description

The **SetFocus** function gives a control the input focus. The user's keystrokes are then received by that control, allowing them to type into a text input control or use the *Enter* key to select a button. The user can also use the *Tab* key, touch, mouse, or other gesture to move the input focus themselves. *Tab* key behavior is governed by the **TabIndex** property.

Use the **SetFocus** function to set the focus when (each with an example below):

- a newly exposed or enabled input control, to guide the user in what comes next and for faster data entry.
- a form is validated, to focus and display the offending input control for quick resolution.
- a screen is displayed, to focus the first input control with the **OnVisible** property of the **Screen**.

The control with focus may be visually different based on the **FocusedBorderColor** and **FocusedBorderThickness** properties.

Limitations

SetFocus can only be used with:

- **Button** control
- **Icon** control
- **Image** control
- **Label** control
- **TextInput** control

You cannot set the focus to controls that are within a **Gallery** control, **Edit form** control, or Component. **SetFocus** can be used with a control in a scrollbale screen.

You cannot set the focus to controls that are within a **Container** control.

You can only set the focus to controls on the same screen as the formula containing the **SetFocus** call.

Attempting to set the focus to a control that has its **DisplayMode** property set to **Disabled** has no effect. Focus will remain where it was previously.

On Apple iOS, the soft keyboard will only be displayed automatically if **SetFocus** was initiated by a direct user action. For example, invoking from a button's **OnSelect** property will display the soft keyboard while invoking from a screen's **OnVisible** will not.

You can use **SetFocus** only in behavior formulas.

Syntax

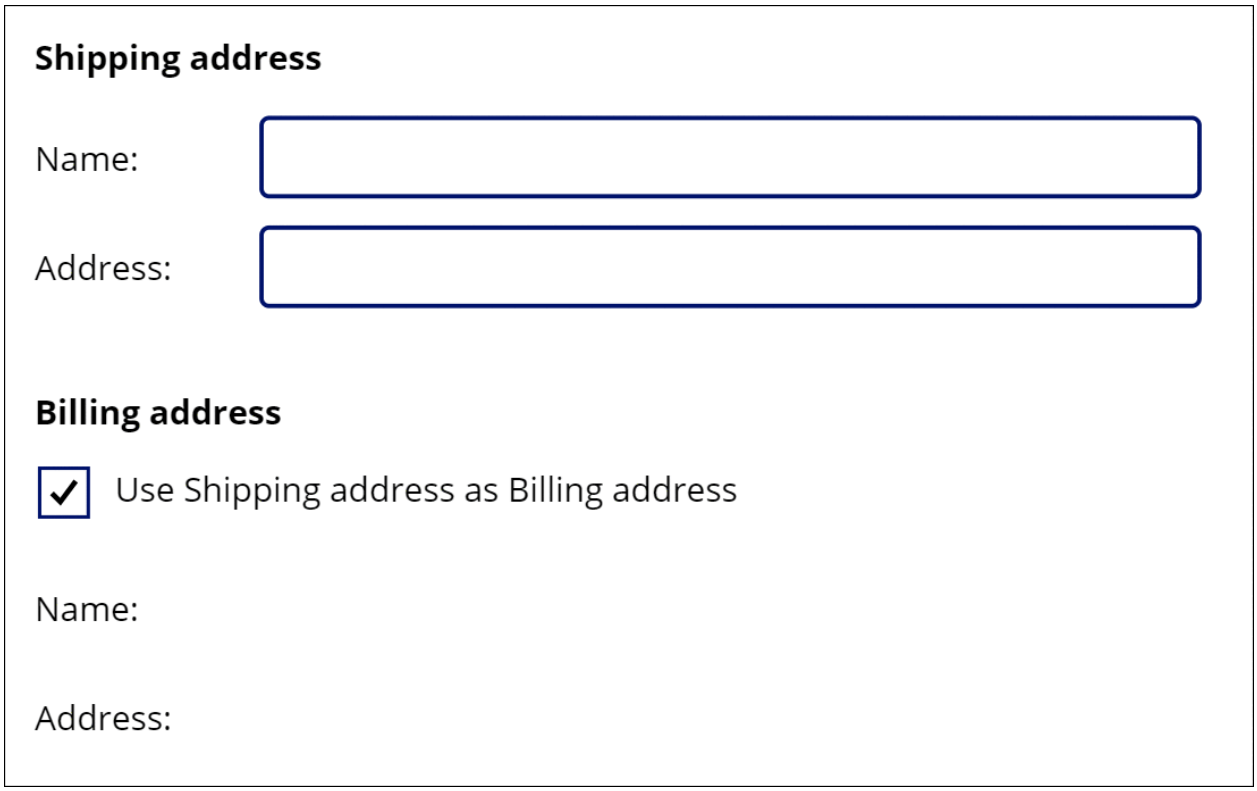
SetFocus(*Control*)

- *Control* – Required. The control to give the input focus.

Examples

Focus on a newly exposed or enabled input control

Many shopping carts allow the customer to use the shipping address as the billing address, alleviating the need to enter the same information twice. If a different billing address is desired, the billing address text input boxes are enabled, and it is helpful to guide the customer to the these newly enabled controls for faster data entry.



There are many formulas in play here, but the one that moves the focus is on the **OnUncheck** property of the **Check box** control:

```
powerappa-dot
SetFocus( BillingName )
```

The *Tab* key can also be used to move focus quickly from one field to another. To better illustrate, the *Tab* key was not used in the animation.

To create this example:

1. Create a new app.
2. Add **Label** controls with the text "Shipping address", "Name:", "Address:", "Billing Address", "Name:", and "Address:" and position them as shown in the animation.
3. Add a **Text Input** control and rename it **ShippingName**.
4. Add a **Text Input** control and rename it **ShippingAddress**.
5. Add a **Check box** control and rename it **SyncAddresses**.
6. Set the **Text** property of this control to the formula `"Use Shipping address as Billing address"`.
7. Add a **Text Input** control and rename it **BillingName**.
8. Set the **Default** property on this control to the formula `ShippingName`.
9. Set the **DisplayMode** property on this control to the formula `If( SyncAddresses.Value, DisplayMode.View, DisplayMode.Edit )`. This will

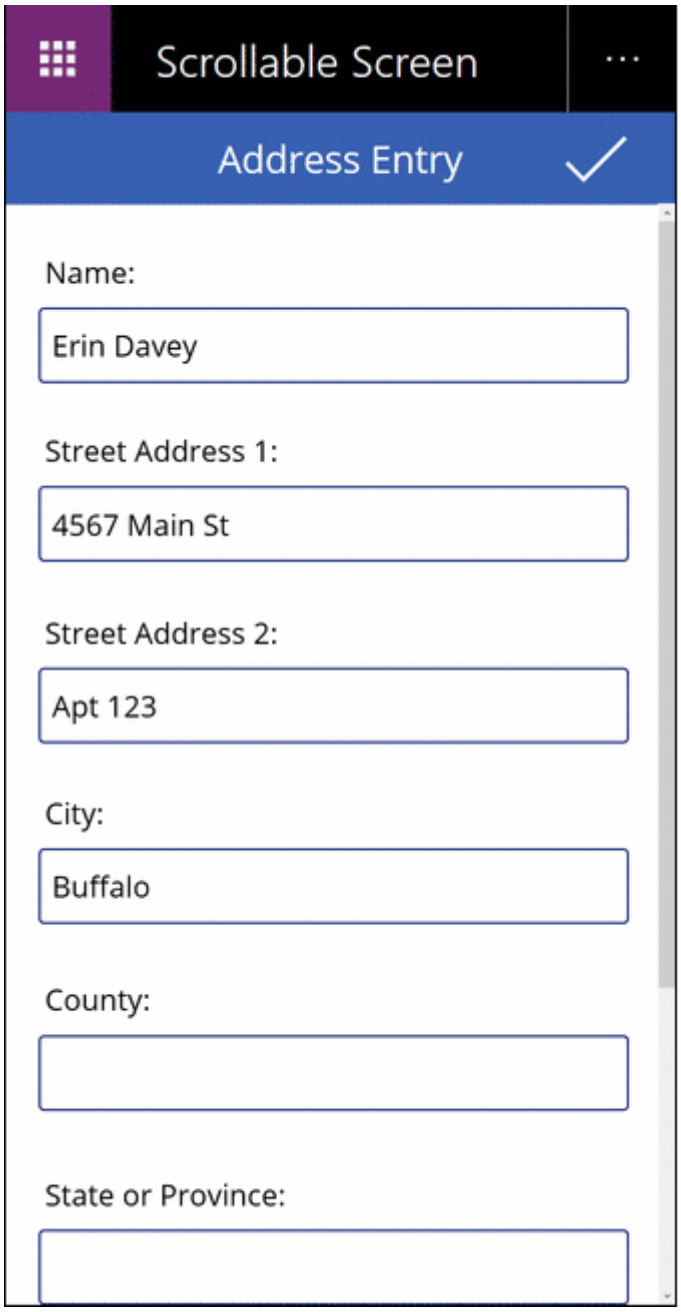
automatically enable or disable this control based on the state of the check box control.

10. Add a **Text Input** control and rename it **BillingAddress**.
11. Set the **Default** property on this control to the formula `ShippingAddress`.
12. Set the **DisplayMode** property on this control to the formula `If( SyncAddresses.Value, DisplayMode.View, DisplayMode.Edit )`. This will automatically enable or disable this control based on the state of the check box control.
13. Set the **Default** property of the check box to the formula `true`. This will default the Billing address to use the same value as the Shipping address.
14. Set the **OnCheck** property of the check box to the formula `Reset( BillingName ); Reset( BillingAddress )`. If the user chooses to sync Shipping and Billing addresses, this will clear any user input in the Billing address fields allowing the **Default** properties on each to pull the values from the corresponding Shipping address fields.
15. Set the **OnUncheck** property of the check box to the formula `SetFocus( BillingName )`. If the user chooses to have a different billing address, this will move the focus to the first control in the Billing address. The controls will have already been enabled due to their **DisplayMode** properties.

Focus on validation issues

> **Note**
>
> Although this example appears to be an **Edit form** control, unfortunately **SetFocus** is not yet supported by that control. Instead, this example uses a scrollable screen to host the input controls.

When validating a form, it can be helpful to not only display a message if there is a problem but to also take the user to the field that is offending. It can be particularly helpful if the field in question is scrolled off the screen and not visible.



In this animation, the validation button is repeatedly pressed until all the fields have been filled in properly. Note that the mouse pointer doesn't move down from the top of the screen. Instead the **SetFocus** function hsa moved the input focus to the control that requires attention with this formula:

Power Apps

```
If( IsBlank( Name ),
        Notify( "Name requires a value", Error ); SetFocus( Name ),
    IsBlank( Street1 ),
        Notify( "Street Address 1 requires a value", Error ); SetFocus(
Street1 ),
    IsBlank( Street2 ),
        Notify( "Street Address 2 requires a value", Error ); SetFocus(
Street2 ),
    IsBlank( City ),
        Notify( "City requires a value", Error ); SetFocus( City ),
    IsBlank( County ),
```

```
        Notify( "County requires a value", Error ); SetFocus( County ),
    IsBlank( StateProvince ),
        Notify( "State or Province requires a value", Error ); SetFocus(
StateProvince ),
    IsBlank( PostalCode ),
        Notify( "Postal Code requires a value", Error ); SetFocus(
PostalCode ),
    IsBlank( Phone ),
        Notify( "Contact Phone requires a value", Error ); SetFocus( Phone
),
    Notify( "Form is Complete", Success )
)
```

To create this example:

1. Create a new, blank phone app.
2. From the **Insert** menu, select **New screen**, and then select **Scrollable**.
3. In the center section of the screen, add **Text input** controls and name them **Name**, **Street1**, **Street2**, **City**, **County**, **StateProvince**, **PostalCode**, and **Phone**. Add **Label** controls above each one to identify the fields. You may need to resize the section if it is not long enough to fit all the controls.
4. Add a checkmark **Icon** control at the top of the screen, above the scrollable section.
5. Set the **OnSelect** property of the icon control to the formula `If( IsBlank( ...` given above.

Focus when displaying a screen

> ⓘ Note
>
> Although this example appears to be an **Edit form** control, unforutnatley **SetFocus** is not yet supported by that control. Instead, this example uses a scrollable screen to host the input controls.

Similar to exposing an input control, when displaying a data entry screen it is helpful to focus the first input control for faster data entry.



In this animation, the data entry screen on the left is not using **SetFocus**. Upon display no input control has focus, requiring the user to tab, touch, mouse, or use another means to focus the **Name** field before a value can be typed into it.

On the right we have exactly the same app with the **OnVisible** property of the data entry screen set to this formula:

Power Apps

```
SetFocus( Name )
```

This sets the focus to the **Name** field automatically. The user can begin typing and tabbing between fields immediately with no prior action required.

To create this example:

1. Create the "Focus on validation issues" app above.
2. On this screen, set the **OnVisible** property to the formula `SetFocus( Name )`.

3. Add a second screen.
4. Add a **Button** control.
5. Set the **OnSelect** property of this control to the formula `Navigate( Screen1 )`.
6. Preview the app from this screen. Press the button. The **OnVisible** formula will be evaluated and the **Name** field will automatically be in focus.

SetProperty function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

The SetProperty function simulates interactions with input controls as if the user had entered or set a value on the control. This function is only available if you are writing tests in the Power Apps Test Studio. The following properties can be set using the SetProperty function.

> **Important**
>
> **Assert** function is only available for Test Studio in Power Apps.

Syntax

SetProperty(Control Property, value)

- *Control Property* – Required. The control property to set on behalf of the user.
- *Value* – Required. The value of the property to set on behalf of the user.

Examples

Expand table

Control	Property	Example expression
TextInput	Text	`SetProperty(TextInput1.Text, "Sample text")`
RichTextEditor	HtmlText	`SetProperty(RichTextEditor1.HtmlText, "<p>Sample text</p>")`
Toggle	Value	`SetProperty(Toggle1.Value, false)`
Checkbox	Value	`SetProperty(Checkbox1.Value, false)`
Slider	Value	`SetProperty(Slider1.Value, 10)`
Rating	Value	`SetProperty(Rating1.Value, 5)`
DatePicker	SelectedDate	`SetProperty(DatePicker1.SelectedDate, Date(2020,3,10))`
Radio	Selected	`SetProperty(Radio1.Selected, "Yes")`

Control	Property	Example expression
Radio	SelectedText	`SetProperty(Radio1.SelectedText, "Yes")`
Dropdown	Selected	`SetProperty(Dropdown1.Selected, {Value:"Sample value"})`
Dropdown	SelectedText	`SetProperty(Dropdown1.SelectedText, {Value:"Sample value"})`
Combobox	Selected	`SetProperty(Dropdown1.Selected, {Value:"Sample value"})`
Combobox	SelectedItems	`SetProperty(ComboBox1.SelectedItems, Table({Value:"Sample value"},({Value:"Sample value"}))`
ListBox	Selected	`SetProperty(Listbox1.Selected, {'Value':"Sample value"})`
ListBox	SelectedItems	`SetProperty(Listbox1.SelectedItems, Table({Value:"Sample value"},({Value:"Sample value"}))`

See Also

Test Studio Overview
Working with Test Studio

Sequence function

04/20/2025

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Generate a table of sequential numbers.

Description

The **Sequence** function generates a single column table of sequential numbers, such as 1, 2, 3. The name of the column is **Value**. `Sequence( 4 )` is equivalent to `[1,2,3,4]`.

Use **Sequence** with the **ForAll** function to iterate a specific number of times. For example, the following formula adds 10 random numbers to the collection **MyRandomNumbers**:

```
Power Fx
ForAll( Sequence( 10 ), Collect( MyRandomNumbers, Rand() ) )
```

ForAll can also be used to transform the value into other data types and return a new table. For example, the following formula returns a table of the next 10 days:

```
Power Fx
ForAll( Sequence( 10 ), DateAdd( Today(), Value, Days ) )
```

The number of records to generate is rounded down to the nearest whole number and must be in the range 0 to 50,000. Generating a table with zero records results in an *empty* table.

> ⓘ **Note**
>
> **Sequence** is limited to 50,000 records.

Syntax

Sequence(*Records* [, *Start* [, *Step*]])

- *Records* – Required. The number of records to create. Must be in the range 0 to 50,000.
- *Start* – Optional. The starting number for the sequence. Default is 1.

- *Step* – Optional. The increment for each successive number in the sequence. *Step* can be negative to count down from the *Start*. Default is 1.

Examples

Basic usage

Expand table

Formula	Description	Result
Sequence(4)	Generates a 4 record table starting at the default 1 and incrementing by the default 1.	**Value** 1 2 3 4
Sequence(4, 24)	Generates a 4 record table starting at 24 and incrementing by the default 1.	**Value** 24 25 26 27
Sequence(4, 4, -1)	Generates a 4 record table starting at 4 and incrementing by -1, effectively counting backward.	**Value** 4 3 2 1
Sequence(4, -100, 0.5)	Generates a 4 record table starting at -100 and incrementing by 0.5.	**Value** -100 -99.5 -99 -98.5
Sequence(0.9)	Generates an *empty* table as the count rounds down to 0.	**Value**

Formula	Description	Result
ForAll(Sequence(4), Rand())	Generates a 4 record table of random numbers.	**Value** 0.51171399 0.21144117 0.95787123 0.22094083 *Actual numbers will vary.*
Concat(Sequence(5), Text(Value) & " ")	Generates a string of numbers from 1 to 5.	**"1 2 3 4 5 "**

Character map

See the Char function reference for two **Sequence** functions working together to display a character map in a two-dimensional layout.

Chessboard

See the As operator reference for two **Sequence** functions working together to create a chessboard in a text string and in two nested galleries.

AddColumns, DropColumns, RenameColumns, and ShowColumns functions

06/21/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Shapes a table or record by adding, dropping, renaming, and selecting its columns.

The ForAll function can also be used to shape a table, by returning a table of new records created from existing columns.

Overview

These functions shape a table or record by adjusting its columns:

- Reduce a table or record that contains multiple columns down to a single column for use with single-column functions, such as **Lower** or **Sqrt**.
- Add a calculated column to a table or record (for example, a **Total Price** column that shows the results of multiplying **Quantity** by **Unit Price**).
- Rename a column to something more meaningful, for display to users or for use in formulas.

A table is a value in Power Apps, just like a string or a number. You can specify a table as an argument in a formula, and functions can return a table as a result.

> **Note**
>
> The functions that this topic describes don't modify the original table. Instead, they take that table as an argument and return a new table with a transform applied. For more information, see working with tables.

You can't modify the columns of a data source by using these functions. You must modify the data at its source. You can add columns to a collection with the **Collect** function. For more information, see working with data sources.

Description

The **AddColumns** function adds a column to a table or record, and a formula defines the values in that column. Existing columns remain unmodified.

The formula is evaluated for the provided record, or for each record of the provided table.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

The **DropColumns** function excludes columns from a table or record. All other columns remain unmodified. **DropColumns** excludes columns, and **ShowColumns** includes columns.

Use the **RenameColumns** function to rename one or more columns of a table or record by providing at least one argument pair that specifies the name of a column that the table or record contains (the old name, which you want to replace) and the name of a column that the table or record doesn't contain (the new name, which you want to use). The old name must already exist in the table or record, and the new name must not exist. Each column name may appear only once in the argument list as either an old column name or a new column name. To rename a column to an existing column name, first drop the existing column with **DropColumns**, or rename the existing column out of the way by nesting one **RenameColumns** function within another.

The **ShowColumns** function includes columns of a table or record and drops all other columns. You can use **ShowColumns** to create a single-column table or record from a multi-column table or record. **ShowColumns** includes columns, and **DropColumns** excludes columns.

For all these functions, the result is a new table or record with the transform applied. The original table or record isn't modified. You can't modify an existing table or record with a formula. SharePoint, Microsoft Dataverse, SQL Server, and other data sources provide tools for modifying the columns of lists and tables, which are often referred to as the schema. The functions in this topic only transform an input table or record, without modifying the original, into an output table or record for further use.

The arguments to these functions support delegation. For example, a **Filter** function used as an argument to pull in related records searches through all listings, even if the '**[dbo].[AllListings]**' data source contains a million rows:

Power Fx

```
AddColumns( RealEstateAgents,
   Listings,
```

```
    Filter(  '[dbo].[AllListings]', ListingAgentName = AgentName )
)
```

However, the output of these functions is subject to the non-delegation record limit. In this example, only 500 records are returned even if the **RealEstateAgents** data source has 501 or more records.

If you use **AddColumns** in this manner, **Filter** must make separate calls to the data source for each of those first records in **RealEstateAgents**, which causes lots of network chatter. If **[dbo](.[AllListings]** is small enough and doesn't change often. You can call the **Collect** function in OnStart to cache the data source in your app when it starts. As an alternative, you could restructure your app so that you pull in the related records only when the user asks for them.

> **Note**
>
> In Power Apps prior to version 3.24042, column names were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

Syntax

AddColumns(*TableOrRecord*, *ColumnName1*, *Formula1* [, *ColumnName2*, *Formula2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to add.
- *Formula(s)* - Required. Formulas to evaluate for each record. The result is added as the value of the corresponding new column. You can reference other columns of the table or record in this formula.

DropColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to drop.

RenameColumns(*TableOrRecord*, *OldColumnName1*, *NewColumnName1* [, *OldColumnName2*, *NewColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.

- *OldColumnName(s)* - Required. Names of the columns to rename from the original table or record. This element appears first in the argument pair (or first in each argument pair if the formula includes more than one pair).
- *NewColumnName(s)* - Required. Replacement names. This element appears last in the argument pair (or last in each argument pair if the formula includes more than one pair).

ShowColumns(*TableOrRecord*, *ColumnName1* [, *ColumnName2*, ...])

- *TableOrRecord* - Required. Table or record to operate on.
- *ColumnName(s)* - Required. Names of the columns to include.

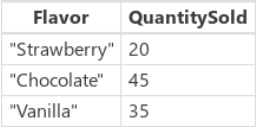
Examples

The examples in this section use the **IceCreamSales** data source, which contains the data in this table:

Flavor	UnitPrice	QuantitySold
"Strawberry"	1.99	20
"Chocolate"	2.99	45
"Vanilla"	1.50	35

None of these examples modify the **IceCreamSales** data source. Each function transforms the value of the data source as a table and returns that value as the result.

Expand table

Formula	Description	Result
AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold)	Adds a **Revenue** column to the result. For each record, **UnitPrice * QuantitySold** is evaluated, and the result is placed in the new column.	
DropColumns(IceCreamSales, UnitPrice)	Excludes the **UnitPrice** column from the result. Use this function to exclude columns, and use **ShowColumns** to include them.	
ShowColumns(IceCreamSales, Flavor)	Includes only the **Flavor** column in the result. Use this function include columns, and use **DropColumns** to exclude them.	
RenameColumns(IceCreamSales, UnitPrice, Price)	Renames the **UnitPrice** column in the result.	

Formula	Description	Result
RenameColumns(IceCreamSales, UnitPrice, Price, QuantitySold, Number)	Renames the **UnitPrice** and **QuantitySold** columns in the result.	<table><tr><th>Flavor</th><th>Price</th><th>Number</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>20</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>45</td></tr><tr><td>"Vanilla"</td><td>1.50</td><td>35</td></tr></table>
DropColumns(RenameColumns(AddColumns(IceCreamSales, Revenue, UnitPrice * QuantitySold), UnitPrice, Price), Quantity)	Performs the following table transforms in order, starting from the inside of the formula: 1. Adds a **Revenue** column based on the per record calculation of **UnitPrice * Quantity**. 2. Renames **UnitPrice** to **Price**. 3. Excludes the **Quantity** column. Order is important, for example we can't calculate with **UnitPrice** after it's renamed.	<table><tr><th>Flavor</th><th>Price</th><th>Revenue</th></tr><tr><td>"Strawberry"</td><td>1.99</td><td>39.80</td></tr><tr><td>"Chocolate"</td><td>2.99</td><td>134.55</td></tr><tr><td>"Vanilla"</td><td>1.49</td><td>52.15</td></tr></table>

Step by step

Let's try some of the examples from earlier in this article.

1. Create a collection by adding a **Button** control and setting its **OnSelect** property to this formula:

   Power Fx

   ```
   ClearCollect( IceCreamSales,
      Table(
          { Flavor: "Strawberry", UnitPrice: 1.99, QuantitySold: 20 },
          { Flavor: "Chocolate", UnitPrice: 2.99, QuantitySold: 45 },
          { Flavor: "Vanilla", UnitPrice: 1.50, QuantitySold: 35 }
      )
   )
   ```

2. Run the formula by selecting the button while holding down the Alt key.

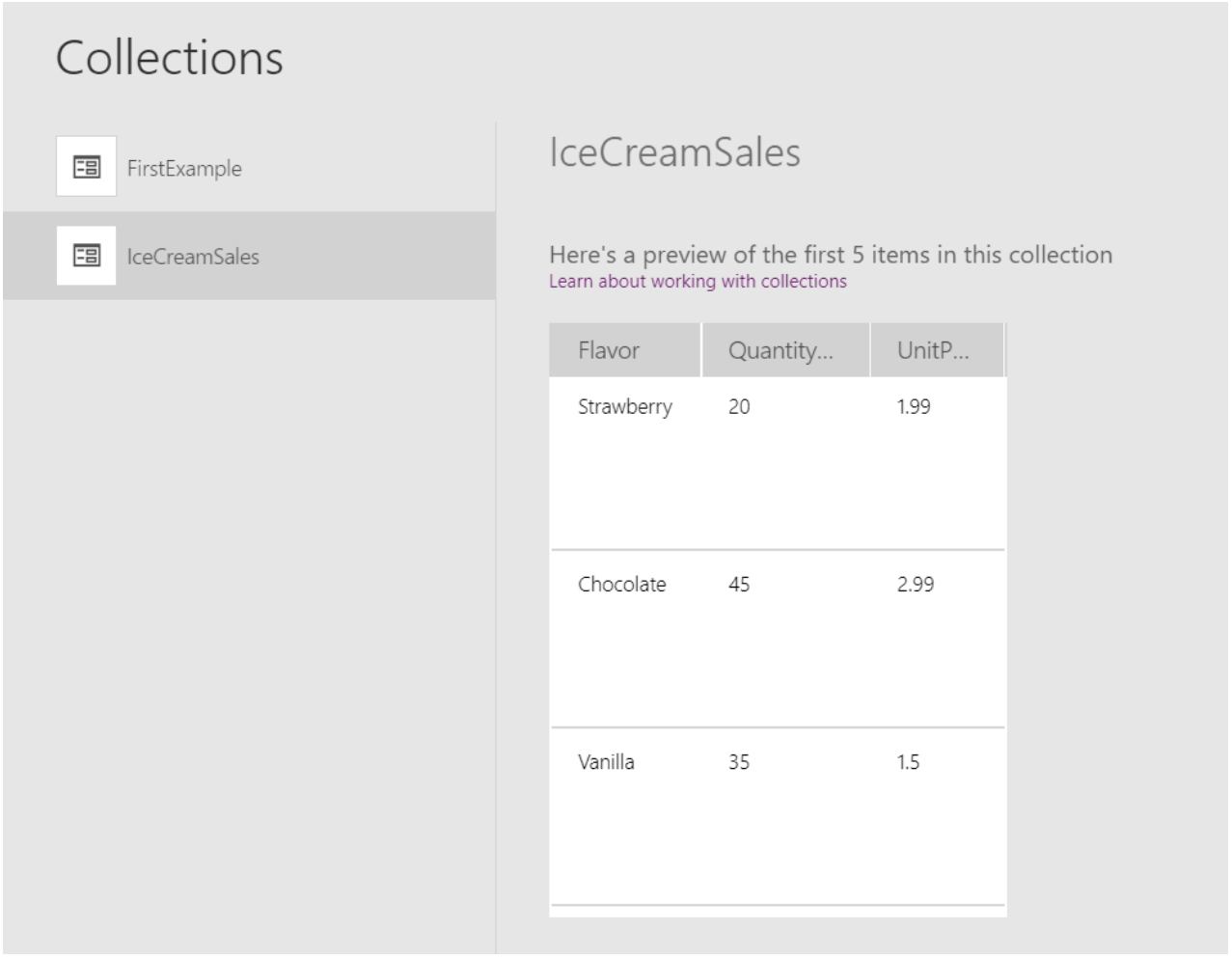
3. Add a second **Button** control, set its **OnSelect** property to this formula, and then run it:

   Power Fx

   ```
   ClearCollect( FirstExample,
      AddColumns( IceCreamSales, Revenue, UnitPrice * QuantitySold )
   )
   ```

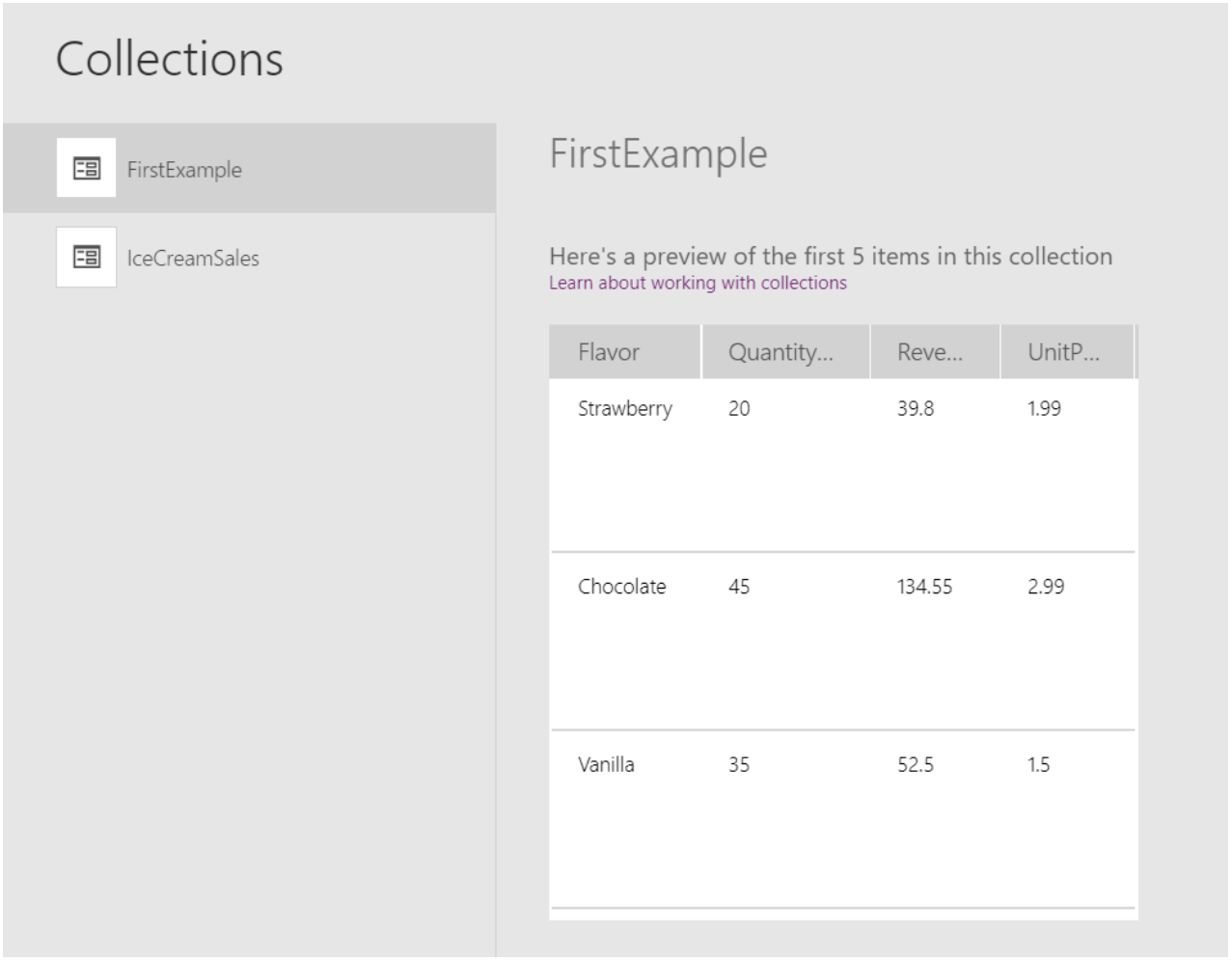
4. On the **File** menu, select **Collections**, and then select **IceCreamSales** to show that collection.

As this graphic shows, the second formula didn't modify this collection. The **AddColumns** function used **IceCreamSales** as a read-only argument; the function didn't modify the table to which that argument refers.



5. Select **FirstExample**.

   As this graphic shows, the second formula returned a new table with the added column. The **ClearCollect** function captured the new table in the **FirstExample** collection, adding something to the original table as it flowed through the function without modifying the source:



Map columns in a component

See Map columns.

Shuffle function

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Randomly reorders the records of a table.

Description

The **Shuffle** function reorders the records of a table.

Shuffle returns a table that has the same columns and number of rows as the argument.

Syntax

Shuffle(*Table*)

- *Table* - Required. Table to shuffle.

Example

If you stored details about playing cards in a collection named **Deck**, this formula would return a copy of that collection that has been randomly shuffled.

Shuffle(Deck)

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Sort and SortByColumns functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Sorts a table.

> ⓘ **Note**
>
> PAC CLI pac power-fx commands do not support the **SortByColumns** function.

Description

The **Sort** function sorts a table based on a formula.

The formula is evaluated for each record of the table, and the results are used to sort the table. The formula must result in a number, a string, or a Boolean value; it can't result in a table or a record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

To sort first by one column and then by another, you embed a **Sort** formula within another. For example, you can use this formula to sort a **Contacts** table first by a **LastName** column and then by a **FirstName** column: **Sort(Sort(Contacts, LastName), FirstName)**

The **SortByColumns** function can also be used to sort a table based on one or more columns.

The parameter list for **SortByColumns** provides the names of the columns to sort by and the sort direction per column. Sorting is performed in the order of the parameters (sorted first by the first column, then the second, and so on). Column names are specified as strings, requiring double quotes if directly included in the parameter list. For example, **SortByColumns(CustomerTable, "LastName")**.

You can combine **SortByColumns** with a **Drop down** or **List box** control to enable users to select which column to sort by.

In addition to sorting ascending or descending, **SortByColumns** can sort based on a single column table of values. For example, you can sort record based on the name of a day of the week by supplying **["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]** as the sort order. All records which have **Monday"** will come first, followed by **Tuesday**, and so on. Records found that do not appear in the sort table are put at the end of the list.

Tables are a value in Power Apps, just like a string or number. They can be passed to and returned from functions. **Sort** and **SortByColumn** don't modify a table; instead they take a table as an argument and return a new table that has been sorted. See working with tables for more details.

Delegation

When possible, Power Apps will delegate filter and sort operations to the data source and page through the results on demand. For example, when you start an app that shows a **Gallery** control filled with data, only the first set of records will be initially brought to the device. As the user scrolls, additional data is brought down from the data source. The result is a faster start time for the app and access to very large data sets.

However, delegation may not always be possible. Data sources vary on what functions and operators they support with delegation. If complete delegation of a formula isn't possible, the authoring environment will flag the portion that can't be delegated with a warning. When possible, consider changing the formula to avoid functions and operators that can't be delegated. The delegation list details which data sources and operations can be delegated.

If delegation is not possible, Power Apps will pull down only a small set of records to work on locally. Filter and sort functions will operate on a reduced set of records. What is available in the **Gallery** may not be the complete story, which could be confusing to users.

See the delegation overview for more information.

Syntax

Sort(*Table*, *Formula* [, *SortOrder*])

- *Table* - Required. Table to sort.

- *Formula* - Required. This formula is evaluated for each record of the table, and the results are used to sort the table. You can reference columns within the table.
- *SortOrder* - Optional. Specify **SortOrder.Descending** to sort the table in descending order. **SortOrder.Ascending** is the default value.

SortByColumns(*Table*, *ColumnName1* [, *SortOrder1*, *ColumnName2*, *SortOrder2*, ...])

- *Table* - Required. Table to sort.
- *ColumnName(s)* - Required. The column names to sort on, as strings.
- *SortOrder(s)* - Optional. **SortOrder.Ascending** or **SortOrder.Descending**. **SortOrder.Ascending** is the default. If multiple *ColumnNames* are supplied, all but the last column must include a *SortOrder*.

  > ⓘ **Note**
  >
  > For SharePoint and Excel data sources that contain column names with spaces, specify each space as "**_x0020_**". For example, specify "**Column Name**" as "**Column_x0020_Name**".

SortByColumns(*Table*, *ColumnName*, *SortOrderTable*)

- *Table* - Required. Table to sort.
- *ColumnName* - Required. The column name to sort on, as strings.
- *SortOrderTable* - Required. Single column table of values to sort by.

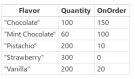
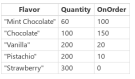
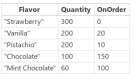
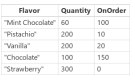
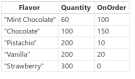
  > ⓘ **Note**
  >
  > For SharePoint and Excel data sources that contain column names with spaces, specify each space as "**_x0020_**". For example, specify "**Column Name**" as "**Column_x0020_Name**".

Examples

For the following examples, we'll use the **IceCream** data source, which contains the data in this table:

Flavor	Quantity	OnOrder
"Chocolate"	100	150
"Vanilla"	200	20
"Strawberry"	300	0
"Mint Chocolate"	60	100
"Pistachio"	200	10

Expand table

Formula	Description	Result
Sort(IceCream, Flavor) **SortByColumns(IceCream, "Flavor")**	Sorts **IceCream** by its **Flavor** column. The **Flavor** column contains strings, so the table is sorted alphabetically. By default, the sort order is ascending.	
Sort(IceCream, Quantity) **SortByColumns(IceCream, "Quantity")**	Sorts **IceCream** by its **Quantity** column. The **Quantity** column contains numbers, so the table is sorted numerically. By default, the sort order is ascending.	
Sort(IceCream, Quantity, SortOrder.Descending) **SortByColumns(IceCream, "Quantity", SortOrder.Descending)**	Sorts **IceCream** by its **Quantity** column. The **Quantity** column contains numbers, so the sort is done numerically. The sort order has been specified as descending.	
Sort(IceCream, Quantity + OnOrder)	Sorts **IceCream** by the sum of its **Quantity** and **OnOrder** columns for each record individually. The sum is a number, so the table is sorted numerically. By default, the sort order is ascending. Since we are sorting by a formula and not by raw column values, there is no equivalent using **SortByColumns**.	
Sort(Sort(IceCream, OnOrder), Quantity) **SortByColumns(IceCream, "OnOrder", SortOrder.Ascending, "Quantity", SortOrder.Ascending)**	Sorts **IceCream** first by its **OnOrder** column, and then by its **Quantity** column. Note that "Pistachio" rose above "Vanilla" in the first sort based on **OnOrder**, and then together they moved to their appropriate place based on **Quantity**.	
SortByColumns(IceCream, "Flavor", ["Pistachio", "Strawberry"])	Sorts **IceCream** by it's **Flavor** column based on the single column table containing "Pistachio" and "Strawberry". Records which have a **Flavor** of "Pistachio" will appear first in the result,	

Formula	Description	Result
	followed by records that contain "Strawberry". For values in the **Flavor** column that are not matched, such as "Vanilla", they will appear after the items that were matched.	

Step by step

To run these examples yourself, create the **IceCream** data source as a collection:

1. Add a button, and set its OnSelect property to this formula:
   ClearCollect(IceCream, { Flavor: "Chocolate", Quantity: 100, OnOrder: 150 }, { Flavor: "Vanilla", Quantity: 200, OnOrder: 20 }, { Flavor: "Strawberry", Quantity: 300, OnOrder: 0 }, { Flavor: "Mint Chocolate", Quantity: 60, OnOrder: 100 }, { Flavor: "Pistachio", Quantity: 200, OnOrder: 10 })
2. Preview the app, select the button, and then press Esc to return to the default workspace.
3. Select **Collections** on the **File** menu to display the collection that you just created, and then press Esc to return to the default workspace.

Sort

1. Add another button, and set its OnSelect property to this formula:
   ClearCollect(SortByFlavor, Sort(IceCream, Flavor))

   The previous formula creates a second collection, named **SortByFlavor**, that contains the same data as **Ice Cream**. However, the new collection contains the data sorted alphabetically by the **Flavor** column in ascending order.

2. Press F5, select the new button, and then press Esc.

3. Select **Collections** on the **File** menu to display both collections, and then press Esc to return to the default workspace.

4. Repeat the last three steps, but change the name of the collection that you want to create, and replace the **Sort** formula with a different formula from the table of examples earlier in this section that uses **Sort**.

SortByColumns

1. Add another button, and set its OnSelect property to this formula:
   ClearCollect(SortByQuantity, SortByColumns(IceCream, "Quantity",

SortOrder.Ascending, "Flavor", SortOrder.Descending))

The previous formula creates a third collection, named **SortByQuantity**, that contains the same data as **Ice Cream**. However, the new collection contains the data sorted numerically by the **Quantity** column in ascending order, and then by the **Flavor** column in descending order.

2. Press F5, select the new button, and then press Esc.

3. Select **Collections** on the **File** menu to display all three collections, and then press Esc to return to the default workspace.

4. Repeat the last three steps, but change the name of the collection that you want to create, and replace the **SortByColumns** formula with a different formula from the table of examples earlier in this section that uses **SortByColumns**.

Feedback

Was this page helpful? Yes No

Provide product feedback

Sort and SortByColumns functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Sorts a table.

> ⓘ **Note**
>
> PAC CLI pac power-fx commands do not support the **SortByColumns** function.

Description

The **Sort** function sorts a table based on a formula.

The formula is evaluated for each record of the table, and the results are used to sort the table. The formula must result in a number, a string, or a Boolean value; it can't result in a table or a record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

To sort first by one column and then by another, you embed a **Sort** formula within another. For example, you can use this formula to sort a **Contacts** table first by a **LastName** column and then by a **FirstName** column: **Sort(Sort(Contacts, LastName), FirstName)**

The **SortByColumns** function can also be used to sort a table based on one or more columns.

The parameter list for **SortByColumns** provides the names of the columns to sort by and the sort direction per column. Sorting is performed in the order of the parameters (sorted first by the first column, then the second, and so on). Column names are specified as strings, requiring double quotes if directly included in the parameter list. For example, **SortByColumns(CustomerTable, "LastName")**.

You can combine **SortByColumns** with a **Drop down** or **List box** control to enable users to select which column to sort by.

In addition to sorting ascending or descending, **SortByColumns** can sort based on a single column table of values. For example, you can sort record based on the name of a day of the week by supplying **["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]** as the sort order. All records which have **Monday"** will come first, followed by **Tuesday**, and so on. Records found that do not appear in the sort table are put at the end of the list.

Tables are a value in Power Apps, just like a string or number. They can be passed to and returned from functions. **Sort** and **SortByColumn** don't modify a table; instead they take a table as an argument and return a new table that has been sorted. See working with tables for more details.

Delegation

When possible, Power Apps will delegate filter and sort operations to the data source and page through the results on demand. For example, when you start an app that shows a **Gallery** control filled with data, only the first set of records will be initially brought to the device. As the user scrolls, additional data is brought down from the data source. The result is a faster start time for the app and access to very large data sets.

However, delegation may not always be possible. Data sources vary on what functions and operators they support with delegation. If complete delegation of a formula isn't possible, the authoring environment will flag the portion that can't be delegated with a warning. When possible, consider changing the formula to avoid functions and operators that can't be delegated. The delegation list details which data sources and operations can be delegated.

If delegation is not possible, Power Apps will pull down only a small set of records to work on locally. Filter and sort functions will operate on a reduced set of records. What is available in the **Gallery** may not be the complete story, which could be confusing to users.

See the delegation overview for more information.

Syntax

Sort(*Table*, *Formula* [, *SortOrder*])

- *Table* - Required. Table to sort.

- *Formula* - Required. This formula is evaluated for each record of the table, and the results are used to sort the table. You can reference columns within the table.
- *SortOrder* - Optional. Specify **SortOrder.Descending** to sort the table in descending order. **SortOrder.Ascending** is the default value.

SortByColumns(*Table*, *ColumnName1* [, *SortOrder1*, *ColumnName2*, *SortOrder2*, ...])

- *Table* - Required. Table to sort.
- *ColumnName(s)* - Required. The column names to sort on, as strings.
- *SortOrder(s)* - Optional. **SortOrder.Ascending** or **SortOrder.Descending**. **SortOrder.Ascending** is the default. If multiple *ColumnNames* are supplied, all but the last column must include a *SortOrder*.

  > ⓘ **Note**
  >
  > For SharePoint and Excel data sources that contain column names with spaces, specify each space as "**_x0020_**". For example, specify "**Column Name**" as "**Column_x0020_Name**".

SortByColumns(*Table*, *ColumnName*, *SortOrderTable*)

- *Table* - Required. Table to sort.
- *ColumnName* - Required. The column name to sort on, as strings.
- *SortOrderTable* - Required. Single column table of values to sort by.

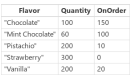
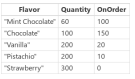
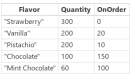
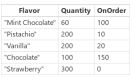
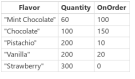
  > ⓘ **Note**
  >
  > For SharePoint and Excel data sources that contain column names with spaces, specify each space as "**_x0020_**". For example, specify "**Column Name**" as "**Column_x0020_Name**".

Examples

For the following examples, we'll use the **IceCream** data source, which contains the data in this table:

Flavor	Quantity	OnOrder
"Chocolate"	100	150
"Vanilla"	200	20
"Strawberry"	300	0
"Mint Chocolate"	60	100
"Pistachio"	200	10

Expand table

Formula	Description	Result
Sort(IceCream, Flavor) **SortByColumns(IceCream, "Flavor")**	Sorts **IceCream** by its **Flavor** column. The **Flavor** column contains strings, so the table is sorted alphabetically. By default, the sort order is ascending.	
Sort(IceCream, Quantity) **SortByColumns(IceCream, "Quantity")**	Sorts **IceCream** by its **Quantity** column. The **Quantity** column contains numbers, so the table is sorted numerically. By default, the sort order is ascending.	
Sort(IceCream, Quantity, SortOrder.Descending) **SortByColumns(IceCream, "Quantity", SortOrder.Descending)**	Sorts **IceCream** by its **Quantity** column. The **Quantity** column contains numbers, so the sort is done numerically. The sort order has been specified as descending.	
Sort(IceCream, Quantity + OnOrder)	Sorts **IceCream** by the sum of its **Quantity** and **OnOrder** columns for each record individually. The sum is a number, so the table is sorted numerically. By default, the sort order is ascending. Since we are sorting by a formula and not by raw column values, there is no equivalent using **SortByColumns**.	
Sort(Sort(IceCream, OnOrder), Quantity) **SortByColumns(IceCream, "OnOrder", SortOrder.Ascending, "Quantity", SortOrder.Ascending)**	Sorts **IceCream** first by its **OnOrder** column, and then by its **Quantity** column. Note that "Pistachio" rose above "Vanilla" in the first sort based on **OnOrder**, and then together they moved to their appropriate place based on **Quantity**.	
SortByColumns(IceCream, "Flavor", ["Pistachio", "Strawberry"])	Sorts **IceCream** by it's **Flavor** column based on the single column table containing "Pistachio" and "Strawberry". Records which have a **Flavor** of "Pistachio" will appear first in the result,	

Formula	Description	Result
	followed by records that contain "Strawberry". For values in the **Flavor** column that are not matched, such as "Vanilla", they will appear after the items that were matched.	

Step by step

To run these examples yourself, create the **IceCream** data source as a collection:

1. Add a button, and set its OnSelect property to this formula:
   ClearCollect(IceCream, { Flavor: "Chocolate", Quantity: 100, OnOrder: 150 }, { Flavor: "Vanilla", Quantity: 200, OnOrder: 20 }, { Flavor: "Strawberry", Quantity: 300, OnOrder: 0 }, { Flavor: "Mint Chocolate", Quantity: 60, OnOrder: 100 }, { Flavor: "Pistachio", Quantity: 200, OnOrder: 10 })
2. Preview the app, select the button, and then press Esc to return to the default workspace.
3. Select **Collections** on the **File** menu to display the collection that you just created, and then press Esc to return to the default workspace.

Sort

1. Add another button, and set its OnSelect property to this formula:
   ClearCollect(SortByFlavor, Sort(IceCream, Flavor))

   The previous formula creates a second collection, named **SortByFlavor**, that contains the same data as **Ice Cream**. However, the new collection contains the data sorted alphabetically by the **Flavor** column in ascending order.

2. Press F5, select the new button, and then press Esc.

3. Select **Collections** on the **File** menu to display both collections, and then press Esc to return to the default workspace.

4. Repeat the last three steps, but change the name of the collection that you want to create, and replace the **Sort** formula with a different formula from the table of examples earlier in this section that uses **Sort**.

SortByColumns

1. Add another button, and set its OnSelect property to this formula:
   ClearCollect(SortByQuantity, SortByColumns(IceCream, "Quantity",

SortOrder.Ascending, "Flavor", SortOrder.Descending))

The previous formula creates a third collection, named **SortByQuantity**, that contains the same data as **Ice Cream**. However, the new collection contains the data sorted numerically by the **Quantity** column in ascending order, and then by the **Flavor** column in descending order.

2. Press F5, select the new button, and then press Esc.

3. Select **Collections** on the **File** menu to display all three collections, and then press Esc to return to the default workspace.

4. Repeat the last three steps, but change the name of the collection that you want to create, and replace the **SortByColumns** formula with a different formula from the table of examples earlier in this section that uses **SortByColumns**.

Feedback

Was this page helpful? Yes No

Provide product feedback

Split function

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Splits a text string into a table of substrings.

Description

The **Split** function breaks a text string into a table of substrings. Use **Split** to break up comma delimited lists, dates that use a slash between date parts, and in other situations where a well defined delimiter is used.

A separator string is used to break the text string apart. The separator can be zero, one, or more characters that are matched as a whole in the text string. Using a zero length or *blank* string results in each character being broken out individually. The matched separator characters are not returned in the result. If no separator match is found, then the entire text string is returned as a single result.

Use the **Concat** function to recombine the string without the separators.

Use the **MatchAll** function to split a string using a regular expression.

The examples show how **Split** can be used with the **First** and **Last** functions to extract a single delimited substring. The **Match** function is often a more concise and powerful choice for regular expressions.

Syntax

Split(*Text*, *Separator*)

- *Text* - Required. Text to split.
- *Separator* - Required. Separator to use in splitting the string. Can be zero, one, or more characters.

Examples

Basic usage

Expand table

Formula	Description	Result
`Split( "Apples, Oranges, Bananas", "," )`	Splits the different fruits apart, based on the comma separator. The split is performed based on only the comma and not the space after it, resulting in a space at the front of " Oranges" and " Bananas".	A single-column table with a `Value` column containing the following values: "Apples", " Oranges", " Bananas"
`TrimEnds( Split( "Apples, Oranges, Bananas", "," ) )`	Same as the previous example, but in this case the space is removed by the **TrimEnds** function, operating on the single column table that is produced by **Split**. We could have also used the separator **", "** which includes the space after the comma, but that wouldn't have worked properly if there's no space or there are two spaces.	A single-column table with a `Value` column containing the following values: "Apples", "Oranges", "Bananas"
`Split( "08/28/17", "/" )`	Splits the date apart, using a forward slash as the separator.	A single-column table with a `Value` column containing the following values: "08", "28", "17"

Different delimiters

Expand table

Formula	Description	Result
`Split( "Hello, World", "," )`	Splits the words apart, using a comma as the separator. The second result starts with a space since it is the character immediately following the comma.	A single-column table with a `Value` column containing the following values: "Hello", " World"
`Split( "Hello, World", "o" )`	Splits the string apart, using the character "o" as the separator.	A single-column table with a `Value` column containing the following values: "Hell", ", W", "rld"
`Split( "Hello, World", "l" )`	Splits the string apart, using the single character "l" as the separator. Since there were no characters between both the l's in **Hello**, a *blank* value was returned.	A single-column table with a `Value` column containing the following values: "He", Blank(), "o, Wor", "d"
`Split( "Hello, World", "ll" )`	Splits the string apart, using the double character "ll" as the separator.	A single-column table with a `Value` column containing the following values: "He", "o, World"

Formula	Description	Result
`Split( "Hello, World", "%" )`	Splits the string apart, using the percent sign as the separator. Since this separator doesn't appear in the string, the entire string is returned as one result.	A single-column table with a `Value` column containing the following value: "Hello, World"
`Split( "Hello, World", "" )`	Splits the string apart, using an empty string as the separator (zero characters). This will break the string on each character.	A single-column table with a `Value` column containing the following values: "H", "e", "l", "l", "o", ",", " ", "W", "o", "r", "l", "d"

Substring extraction

Expand table

Formula	Description	Result
`First( Split( Last( Split( "Bob Jones <bob.jones@contoso.com>", " <" ) ).Result, ">" ) ).Result`	Splits the string apart based on an opening delimiter (<) and extracts the string to the right of the delimiter with **Last**. The formula then splits that result based on the closing delimiter (>) and extracts the string the left of the delimiter with **Right**.	"bob.jones@contoso.com"
`Match( "Bob Jones <bob.jones@contoso.com>", "<(?<email>.+)>" ).email`	Performs the same delimiter based extraction as the last example but uses the **Match** function and a regular expression instead.	"bob.jones@contoso.com"

Abs, Exp, Ln, Power, Log, and Sqrt functions

Article • 04/20/2025

Expand table

Functions	Applies to
Abs **Exp** **Ln** **Power** **Sqrt**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Log	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Calculates absolute values, logarithms, square roots, and the results of raising *e* or any number to specified powers.

Description

The **Abs** function returns the non-negative value of its argument. If a number is negative, **Abs** returns the positive equivalent.

The **Exp** function returns *e* raised to the power of its argument. The transcendental number *e* begins 2.7182818...

The **Ln** function returns the natural logarithm (base *e*) of its argument.

The **Power** function returns a number raised to a power. It's equivalent to using the ^ operator.

The **Log** function returns the logarithm of its first argument in the base specified by its second argument (or 10 if not specified).

The **Sqrt** function returns the number that, when multiplied by itself, equals its argument.

If you pass a single number, the return value is a single result based on the function called. If you pass a single-column table that contains numbers, the return value is a single-column table of results in a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined valued, the result is *blank*. Which can happen with square roots and logarithms of negative numbers.

Syntax

Abs(*Number*)
Exp(*Number*)
Ln(*Number*)
Sqrt(*Number*)

- *Number* - Required. Number to operate on.

Power(*Base*, *Exponent*)

- *Base* - Required. Base number to raise.
- *Exponent* - Required. The exponent to which the base number is raised.

Log(*Number*, *Base*)

- *Number* - Required. Number to calculate the logarithm.
- *Base* - Optional. The base of the logarithm to calculate. By default, 10 (when not specified).

Abs(*SingleColumnTable*)
Exp(*SingleColumnTable*)
Ln(*SingleColumnTable*)
Sqrt(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Examples

Single number

Expand table

Formula	Description	Result
Abs(-55)	Returns the number without the negative sign.	55
Exp(2)	Returns *e* raised to the power of 2, or *e* * *e*.	7.389056...
Ln(100)	Returns the natural logarithm (base *e*) of the number 100.	4.605170...
Log(100)	Returns the logarithm in base 10 of the number 100.	2
Log(64, 2)	Returns the logarithm in base 2 of the number 64.	6

Formula	Description	Result
Power(5, 3)	Returns 5 raised to the power of 3, or 5 * 5 * 5.	125
Sqrt(9)	Returns the number that, when multiplied by itself, results in 9.	3

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains this data:

Expand table

Value
9
-4
2

Expand table

Formula	Description	Result
Abs(ValueTable)	Returns the absolute value of each number in the table.	A single-column table with a `Value` column containing the following values: 9, 4, 2
Exp(ValueTable)	Returns *e* raised to the power of each number in the table.	A single-column table with a `Value` column containing the following values: 8103.083927..., 0.018315..., 7.389056...
Ln(ValueTable)	Returns the natural logarithm of each number in the table.	A single-column table with a `Value` column containing the following values: 2.197224..., Blank(), 0.693147...
Sqrt(ValueTable)	Returns the square root of each number in the table	A single-column table with a `Value` column containing the following values: 3, Blank(), 1.414213...

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a **Label** control, and set its **Text** property to this formula:
   Sqrt(Value(Source.Text))

3. Type a number into **Source**, and confirm that the **Label** control shows the square root of the number that you typed.

EndsWith and StartsWith functions

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Tests whether a text string begins or ends another text string.

Description

The **EndsWith** function tests whether one text string ends with another.

The **StartsWith** function tests whether one text string begins with another.

For both functions, the tests are case insensitive if the back-end datasource supports it by default. For example, collections, Dataverse, SharePoint, and SQL Server support case insensitivity by default. Oracle does not. The return value of both is a Boolean **true** or **false**.

Use **EndsWith** and **StartsWith** with the **Filter** function to search the data within your app. You can also use the **in** operator or the **Search** function to look anywhere within text strings, not just at the beginning or end. Your choice of functions will depend on the needs of your app and which function can be delegated for your particular data source. If one of these functions can't be delegated, a delegation warning will appear at authoring time to warn you of this limitation.

Syntax

EndsWith(*Text*, *EndText*)

- *Text* – Required. The text to test.
- *EndText* – Required. The text to search for at the end of *Text*. If *EndText* is an empty string, **EndsWith** returns *true*.

StartsWith(*Text*, *StartText*)

- *Text* – Required. The text to test.
- *StartText* – Required. The text to search for at the beginning of *Text*. If *StartText* is an empty string, **StartsWith** returns *true*.

Examples

Expand table

Formula	Description	Result
EndsWith("Hello World", "world")	Tests whether **"Hello World"** ends with **"world"**. The test is case insensitive.	**true**
EndsWith("Good bye", "good")	Tests whether **"Good bye"** ends with **"good"**. The *EndText* argument (**"good"**) appears in the text but not at the end.	**false**
EndsWith("Always say hello", "hello")	Tests whether **"Always say hello"** ends with **"hello"**.	**true**
EndsWith("Bye bye", "")	Tests whether **"Bye bye"** ends with an empty text string (**Len** returns 0). Easing its use in **Filter** expressions, **EndsWith** is defined to return **true** in this case.	**true**

Expand table

Formula	Description	Result
StartsWith("Hello World", "hello")	Tests whether **"Hello World"** begins with **"hello"**. The test is case insensitive.	**true**
StartsWith("Good bye", "hello")	Tests whether **"Good bye"** begins with **"hello"**.	**false**
StartsWith("Always say hello", "hello")	Tests whether **"Always say hello"** begins with **"hello"**. Although **"hello"** appears in the text, it doesn't appear at the beginning.	**false**
StartsWith("Bye bye", "")	Tests whether **"Bye bye"** starts with an empty text string (**Len** returns 0). Easing its use in **Filter** expressions, **StartsWith** is defined to return **true** in this case.	**true**

Search user experience

In many apps, you can type one or more characters into a search box to filter a list of records in a large data set. As you type, the list shows only those records that match the search criteria.

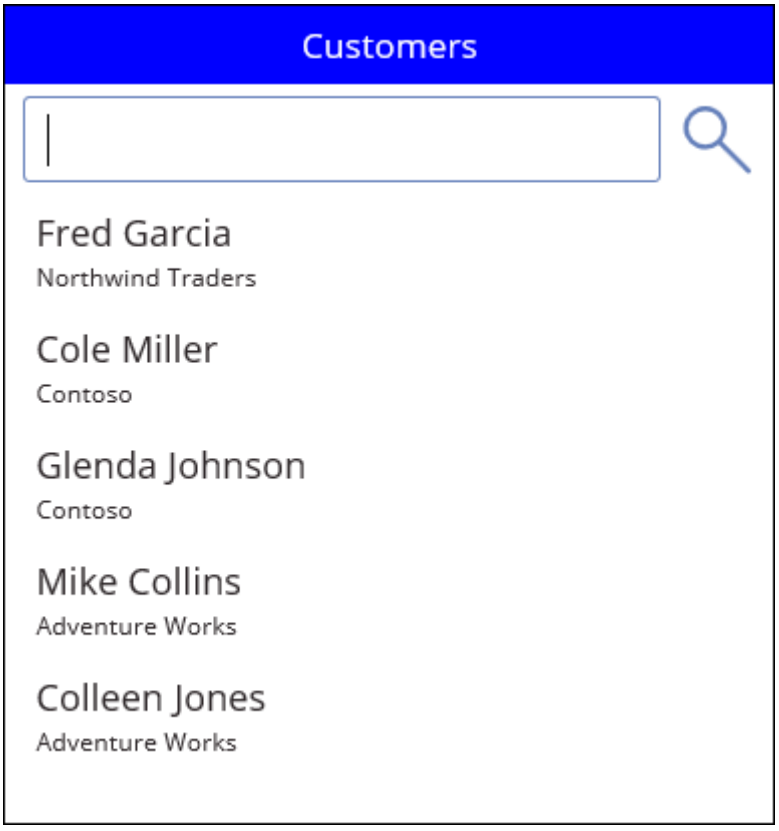
The examples in the rest of this topic show the results of searching a **Customers** list that contains this data:

Name	Company
"Fred Garcia"	"Northwind Traders"
"Cole Miller"	"Contoso"
"Glenda Johnson"	"Contoso"
"Mike Collins"	"Adventure Works"
"Colleen Jones"	"Adventure Works"

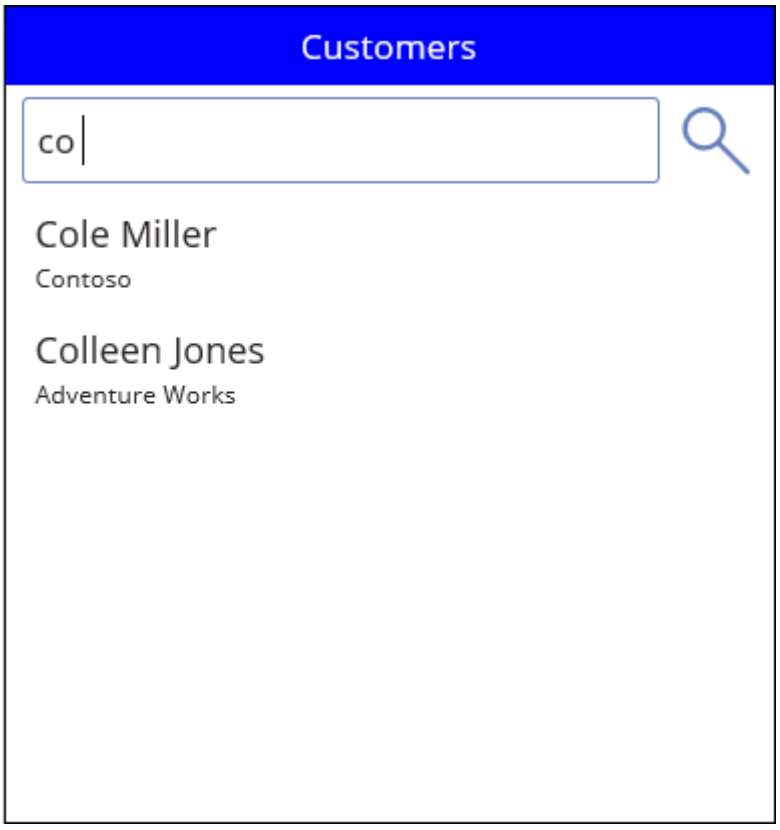
To create this data source as a collection, create a **Button** control and set its **OnSelect** property to this formula:

ClearCollect(Customers, Table({ Name: "Fred Garcia", Company: "Northwind Traders" }, { Name: "Cole Miller", Company: "Contoso" }, { Name: "Glenda Johnson", Company: "Contoso" }, { Name: "Mike Collins", Company: "Adventure Works" }, { Name: "Colleen Jones", Company: "Adventure Works" }))

As in this example, you can show a list of records in a **Gallery control** at the bottom of a screen. Near the top of the screen, you can add a **Text input** control, named **SearchInput**, so that users can specify which records interest them.



As the user types characters in **SearchInput**, the results in the gallery are automatically filtered. In this case, the gallery is configured to show records for which the name of the customer (not the name of the company) starts with the sequence of characters in **SearchInput**.If the user types **co** in the search box, the gallery shows these results:



To filter based on the **Name** column, set the **Items** property of the gallery control to one of these formulas:

Expand table

Formula	Description	Result
Filter(Customers, StartsWith(Name, SearchInput.Text))	Filters the **Customers** data source for records in which the search string appears at the start of the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones** and **Cole Miller**. The gallery doesn't show **Mike Collins** because the **Name** column for that record doesn't start with the search string.	
Filter(Customers, SearchInput.Text in Name)	Filters the **Customers** data source for records in which the search string appears anywhere in the **Name** column. The test is case insensitive. If the user types **co** in the search box, the gallery shows **Colleen Jones**, **Cole Miller,** and **Mike Collins** because the search string appears somewhere in the **Name** column of all of those records.	
Search(Customers, SearchInput.Text, "Name")	Similar to using the **in** operator, the **Search** function searches for a match anywhere within the **Name** column of each record. Note that you must enclose the column name in double quotation marks.	

You can expand your search to include the **Company** column as well as the **Name** column:

Expand table

Formula	Description	Result
Filter(Customers, StartsWith(Name, SearchInput.Text) || StartsWith(Company, SearchInput.Text))	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column starts with the search string (for example, **co**). The || operator is *true* if either **StartsWith** function is *true*.	
Filter(Customers, SearchInput.Text in Name || SearchInput.Text in Company)	Filters the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it.	
Search(Customers, SearchInput.Text, "Name", "Company")	Similar to using the **in** operator, the **Search** function searches the **Customers** data source for records in which either the **Name** column or the **Company** column contains the search string (for example, **co**) anywhere within it. The **Search** function is easier to read and write than **Filter** if you want to specify multiple columns and multiple **in** operators. Note that you must enclose the names of the columns in double quotation marks.	

Average, Max, Min, StdevP, Sum, and VarP functions

Article • 04/20/2025

Expand table

Functions	Applies to
Average **Max** **Min** **Sum**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
StdevP **VarP**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Aggregate functions that summarize a set of numbers.

Description

The **Average** function calculates the average, or arithmetic mean, of its arguments.

The **Max** function finds the maximum value.

The **Min** function finds the minimum value.

The **Sum** function calculates the sum of its arguments.

The **StdevP** function calculates the standard deviation of its arguments.

The **VarP** function calculates the variance of its arguments.

You can supply the values for these functions as:

- Separate arguments. For example, **Sum(1, 2, 3)** returns 6.
- A table and a formula to operate over that table. The aggregate will be calculated on the values of the formula for each record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

These functions operate on numeric values only. Other types of values, such as strings or records, are ignored. Use the **Value** function to convert a string into a number.

The **Average**, **Max**, **Min**, and **Sum** functions can be delegated when used with a data source that supports delegation for these functions. However, **StdevP** and **VarP** can't be delegated for any data sources. If delegation is not supported, only the first portion of the data will be retrieved and then the function applied locally. The result may not represent the complete story. A delegation warning will appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Average(*NumericalFormula1*, [*NumericalFormula2*, ...])
Max(*NumericalFormula1*, [*NumericalFormula2*, ...])
Min(*NumericalFormula1*, [*NumericalFormula2*, ...])
Sum(*NumericalFormula1*, [*NumericalFormula2*, ...])
StdevP(*NumericalFormula1*, [*NumericalFormula2*, ...])
VarP(*NumericalFormula1*, [*NumericalFormula2*, ...])

- *NumericalFormula(s)* - Required. Numeric values to operate on.

Average(*Table*, *NumericalFormula*)
Max(*Table*, *NumericalFormula*)
Min(*Table*, *NumericalFormula*)
Sum(*Table*, *NumericalFormula*)
StdevP(*Table*, *NumericalFormula*)
VarP(*Table*, *NumericalFormula*)

- *Table* - Required. Table to operate on.
- *NumericalFormula* - Required. Formula to evaluate for each record. The result of this formula is used for the aggregation. You can use columns of the table in the formula.

Examples

Step by step

Let's say that you had a data source named **Sales** that contained a **CostPerUnit** column and a **UnitsSold** column, and you set the **Text** property of a label to this function:
Sum(Sales, CostPerUnit * UnitsSold)

The label would show total sales by multiplying the values in those columns for each record and then adding the results from all records together:

Units sold	Cost per unit
30	$150
45	$125
20	$175

Total sales:

$13625

As a different example, let's say that you had sliders that were named **Slider1**, **Slider2**, and **Slider3** and a label with its **Text** property set to this formula:

Sum(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the sum of all values to which the sliders were set.

Average(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the average of all values to which the sliders were set.

Max(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the maximum of all values to which the sliders were set.

Min(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the minimum of all values to which the sliders were set.

StdevP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the standard deviation of all values to which the sliders were set.

VarP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the variance of all values to which the sliders were set.

Replace and Substitute functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Replace a portion of a string of text with another string.

Description

The **Replace** function identifies the text to replace by starting position and length.

The **Substitute** function identifies the text to replace by matching a string. If more than one match is found, you can replace all of them or specify one to replace.

If you pass a single string, the return value is the modified string. If you pass a single-column table that contains strings, the return value is a single-column table with a **Value** column of modified strings. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

Replace(*String*, *StartingPosition*, *NumberOfCharacters*, *NewString*)

- *String* - Required. The string to operate on.
- *StartingPosition* - Required. Character position to start the replacement. The first character of *String* is at position 1.
- *NumberOfCharacters* - Required. The number of characters to replace in *String*.
- *NewString* - Required. The replacement string. The number of characters in this argument can differ from the *NumberOfCharacters* argument.

Substitute(*String*, *OldString*, *NewString* [, *InstanceNumber*])

- *String* - Required. The string to operate on.
- *OldString* - Required. The string to replace.
- *NewString* - Required. The replacement string. *OldString* and *NewString* can have different lengths.
- *InstanceNumber* - Optional. Use this argument to specify which instance of *OldString* to replace if *String* contains more than one instance. If you don't specify this argument, all instances will be replaced.

Replace(*SingleColumnTable*, *StartingPosition*, *NumberOfCharacters*, *NewString*)

- *SingleColumnTable* - Required. A single-column table of strings to operate on.
- *StartingPosition* - Required. Character position to start the replacement. The first character of each string in the table is at position 1.
- *NumberOfCharacters* - Required. The number of characters to replace in each string.
- *NewString* - Required. The replacement string. The number of characters in this argument can differ from the *NumberOfCharacters* argument.

Substitute(*SingleColumnTable*, *OldString*, *NewString* [, *InstanceNumber*])

- *SingleColumnTable* - Required. A single-column table of strings to operate on.
- *OldString* - Required. The string to replace.
- *NewString* - Required. The replacement string. *OldString* and *NewString* can have different lengths.
- *InstanceNumber* - Optional. Use this argument to specify which instance of *OldString* to replace if *String* contains more than one instance. If you don't specify this argument, all instances will be replaced.

Examples

Expand table

Formula	Description	Result
Replace("abcdefghijk", 6, 5, "*")	Replaces five characters in "abcdefghijk" with a single "*" character, starting with the sixth character ("f").	"abcde*k"
Replace("2019", 3, 2, "20")	Replaces the last two characters of "2019" with "20".	"2020"
Replace("123456", 1, 3, "_")	Replaces the first three characters of "123456" with a single "_" character.	"_456"
Substitute("Sales Data", "Sales", "Cost")	Substitutes the string "Cost" for "Sales".	"Cost Data"
Substitute("Quarter 1, 2018", "1", "2", 1)	Substitutes only the first instance of "1" with "2" because the fourth argument	"Quarter 2, 2018"

Formula	Description	Result
	(*InstanceNumber*) is provided with a 1.	
Substitute("Quarter 1, 2011", "1", "2", 3)	Substitutes only the third instance of "1" with "2" because the fourth argument (*InstanceNumber*) is provided with a 3.	"Quarter 1, 2012"
Substitute("Quarter 1, 2011", "1", "2")	Substitutes all instances of "1" with "2" because the fourth argument (*InstanceNumber*) isn't provided.	"Quarter 2, 2022"
Replace(["Quarter 1, 2018", "Quarter 2, 2011", "Quarter 4, 2019"], 9, 1, "3")	Replaces the ninth character in each record of the single-column table with "3".	A single-column table with a `Value` column containing the following values: ["Quarter 3, 2018", "Quarter 3, 2011", "Quarter 3, 2019"]
Substitute(["Qtr 1, 2018", "Quarter 1, 2011", "Q1, 2019"], "1", "3", 1)	Because the fourth argument (*InstanceNumber*) is provided with a value of 1, substitutes only the first instance of "1" in each record of the single-column table with "3".	A single-column table with a `Value` column containing the following values: ["Qtr 3, 2018", "Quarter 3, 2011", "Q3, 2019"]
Substitute(["Qtr 1, 2018", "Quarter 1, 2011", "Q1, 2019"], "1", "3")	Because the fourth argument (*InstanceNumber*) isn't provided, substitutes all instances of "1" in each record of the single-column table with "3".	A single-column table with a `Value` column containing the following values: ["Qtr 3, 2038", "Quarter 3, 2033", "Q3, 2039"]

Feedback

Was this page helpful? Yes No

Provide product feedback

EditForm, NewForm, SubmitForm, ResetForm, and ViewForm functions

Article • 03/22/2024

Applies to: Canvas apps

View, edit, or create an item, save the contents, and reset the controls in an **Edit form** control.

Overview

These functions change the state of the **Edit form** control. The form control can be in one of these modes:

Expand table

Mode	Description
FormMode.Edit	The form is populated with an existing record and the user can modify the values of the fields. Once complete, the user can save the changes to the record.
FormMode.New	The form is populated with default values and the user can modify the values of the fields. Once complete, the user can add the record to the data source.
FormMode.View	The form is populated with an existing record but the user cannot modify the values of the fields.

Description

These functions are often invoked from the **OnSelect** formula of a **Button** or **Image** control so that the user can save edits, abandon edits, or create a record. You can use controls and these functions together to create a complete solution.

These functions return no values.

You can use these functions only in behavior formulas.

SubmitForm

Use the **SubmitForm** function in the **OnSelect** property of a Button control to save any changes in a Form control to the data source.

Before submitting any changes, this function checks for validation issues with any field that's marked as required or that has one or more constraints on its value. This behavior matches that of the **Validate** function.

SubmitForm also checks the **Valid** property of the Form, which is an aggregation of all the **Valid** properties of the **Card** controls that the Form control contains. If a problem occurs, the data isn't submitted, and the **Error** and **ErrorKind** properties of the Form control are set accordingly.

If validation passes, **SubmitForm** submits the change to the data source.

- If successful, the Form's **OnSuccess** behavior runs, and the **Error** and **ErrorKind** properties are cleared. If the form was in **FormMode.New** mode, it is returned to **FormMode.Edit** mode.
- If unsuccessful, the Form's **OnFailure** behavior runs, and the **Error** and **ErrorKind** properties are set accordingly. The mode of the form is unchanged.

EditForm

The **EditForm** function changes the Form control's mode to **FormMode.Edit**. In this mode, the contents of the Form control's **Item** property are used to populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is changed, not created. **FormMode.Edit** is the default for the Form control.

> ⓘ **Note**
>
> When the form is in edit mode and Item is null, data card properties are not evaluated and will return default values.

NewForm

The **NewForm** function changes the Form control's mode to **FormMode.New**. In this mode, the contents of the Form control's **Item** property are ignored, and the default values of the Form's **DataSource** property populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is created, not changed.

ResetForm

The **ResetForm** function resets the contents of a form to their initial values, before the user made any changes. If the form is in **FormMode.New** mode, the form is reset to **FormMode.Edit** mode. The **OnReset** behavior of the form control also runs. You can also reset individual controls with the **Reset** function but only from within the form.

ViewForm

The **ViewForm** function changes the Form control's mode to **FormMode.View**. In this mode, the contents of the Form control's **Item** property are used to populate the form. The **SubmitForm** and **ResetForm** functions have no effect when in this mode.

DisplayMode Property

The current mode can be read through the **Mode** property. The mode also determines the value of the **DisplayMode** property, which can be used by data cards and controls within the form control. Often, the data card's **DisplayMode** property will be set to **Parent.DisplayMode** (referencing the form) as will the control's **DisplayMode** property (referencing the data card):

Expand table

Mode	DisplayMode	Description
FormMode.Edit	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept changes to a record.
FormMode.New	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept a new record.
FormMode.View	**DisplayMode.View**	Data cards and controls are not editable and optimized for viewing.

Syntax

SubmitForm(*FormName*)

- *FormName* - Required. Form control to submit to the data source.

EditForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.Edit** mode.

NewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.New** mode.

ResetForm(*FormName*)

- *FormName* - Required. Form control to reset to initial values. Also switches the form from **FormMode.New** mode to **FormMode.Edit** mode.

ViewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.View** mode.

Examples

See Understand data forms for complete examples.

1. Add a Button control, set its **Text** property to show **Save**, and set its **OnSelect** property to this formula:

   SubmitForm(EditForm)

2. Set the **OnFailure** property of a Form control to blank and its **OnSuccess** property to this formula:

   Back()

3. Name a **Label** control **ErrorText**, and set its **Text** property to this formula:

   EditForm.Error

   When the user selects the **Save** button, any changes in the Form control are submitted to the underlying data source.

   - If the submission succeeds, any changes are saved or, if the Form control is in **New** mode, a record is created. **ErrorText** is *blank* and the previous screen reappears.
   - If the submission fails, **ErrorText** shows a user-friendly error message, and the current screen remains visible so that the user can correct the problem and try again.

4. Add a Button control, set its **Text** property to show **Cancel**, and set its **OnSelect** property to this formula:

   ResetForm(EditForm); Back()

   When the user selects the **Cancel** button, the values in the Form control are reset to what they were before the user started to edit it, the previous screen reappears,

and the Form control is returned to **Edit** mode if it was in **New** mode.

5. Add a Button control, set its Text property to show **New**, and set its OnSelect property to this formula:

   NewForm(EditForm); Navigate(EditScreen, None)

   When the user selects the **New** button, the Form control switches to **New** mode, the default values for the Form control's data source populate that control, and the screen that contains the Form control appears. When the **SubmitForm** function runs, a record is created instead of updated.

Average, Max, Min, StdevP, Sum, and VarP functions

Article • 04/20/2025

Expand table

Functions	Applies to
Average **Max** **Min** **Sum**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
StdevP **VarP**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Aggregate functions that summarize a set of numbers.

Description

The **Average** function calculates the average, or arithmetic mean, of its arguments.

The **Max** function finds the maximum value.

The **Min** function finds the minimum value.

The **Sum** function calculates the sum of its arguments.

The **StdevP** function calculates the standard deviation of its arguments.

The **VarP** function calculates the variance of its arguments.

You can supply the values for these functions as:

- Separate arguments. For example, **Sum(1, 2, 3)** returns 6.
- A table and a formula to operate over that table. The aggregate will be calculated on the values of the formula for each record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

These functions operate on numeric values only. Other types of values, such as strings or records, are ignored. Use the **Value** function to convert a string into a number.

The **Average**, **Max**, **Min**, and **Sum** functions can be delegated when used with a data source that supports delegation for these functions. However, **StdevP** and **VarP** can't be delegated for any data sources. If delegation is not supported, only the first portion of the data will be retrieved and then the function applied locally. The result may not represent the complete story. A delegation warning will appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Average(*NumericalFormula1*, [*NumericalFormula2*, ...])
Max(*NumericalFormula1*, [*NumericalFormula2*, ...])
Min(*NumericalFormula1*, [*NumericalFormula2*, ...])
Sum(*NumericalFormula1*, [*NumericalFormula2*, ...])
StdevP(*NumericalFormula1*, [*NumericalFormula2*, ...])
VarP(*NumericalFormula1*, [*NumericalFormula2*, ...])

- *NumericalFormula(s)* - Required. Numeric values to operate on.

Average(*Table*, *NumericalFormula*)
Max(*Table*, *NumericalFormula*)
Min(*Table*, *NumericalFormula*)
Sum(*Table*, *NumericalFormula*)
StdevP(*Table*, *NumericalFormula*)
VarP(*Table*, *NumericalFormula*)

- *Table* - Required. Table to operate on.
- *NumericalFormula* - Required. Formula to evaluate for each record. The result of this formula is used for the aggregation. You can use columns of the table in the formula.

Examples

Step by step

Let's say that you had a data source named **Sales** that contained a **CostPerUnit** column and a **UnitsSold** column, and you set the **Text** property of a label to this function:
Sum(Sales, CostPerUnit * UnitsSold)

The label would show total sales by multiplying the values in those columns for each record and then adding the results from all records together:

Units sold	Cost per unit
30	$150
45	$125
20	$175

Total sales:

$13625

As a different example, let's say that you had sliders that were named **Slider1**, **Slider2**, and **Slider3** and a label with its **Text** property set to this formula:

Sum(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the sum of all values to which the sliders were set.

Average(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the average of all values to which the sliders were set.

Max(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the maximum of all values to which the sliders were set.

Min(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the minimum of all values to which the sliders were set.

StdevP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the standard deviation of all values to which the sliders were set.

VarP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the variance of all values to which the sliders were set.

If and Switch functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Determines whether any condition in a set is true (**If**) or the result of a formula matches any value in a set (**Switch**) and then returns a result or executes an action.

Description

The **If** function tests one or more conditions until a **true** result is found. If such a result is found, a corresponding value is returned. If no such result is found, a default value is returned. In either case, the returned value might be a string to show, a formula to evaluate, or another form of result.

The **Switch** function evaluates a formula and determines whether the result matches any value in a sequence that you specify. If a match is found, a corresponding value is returned. If no match is found, a default value is returned. In either case, the returned value might be a string to show, a formula to evaluate, or another form of result.

If and **Switch** are very similar, but you should use the best function for your situation:

- Use **If** to evaluate a single condition. The most common syntax for this function is **If**(*Condition*, *ThenResult*, *DefaultResult*), which provides the common "if ... then ... else ..." pattern seen in other programming tools.
- Use **If** to evaluate multiple unrelated conditions. In Power Apps (unlike Microsoft Excel), you can specify multiple conditions without having to nest **If** formulas.
- Use **Switch** to evaluate a single condition against multiple possible matches. You can also use **If** in this case, but you'd need to repeat the formula for each possible match.

You can use both of these functions in behavior formulas to branch between two or more actions. Only one branch will trigger an action. Conditions and matches are evaluated in order, and they stop if a condition is **true** or a match is found.

Blank is returned if no conditions are **true**, no matches are found, and you don't specify a default result.

Syntax

If(*Condition*, *ThenResult* [, *DefaultResult*])
If(*Condition1*, *ThenResult1* [, *Condition2*, *ThenResult2*, ... [, *DefaultResult*]])

- *Condition(s)* - Required. Formula(s) to test for **true**. Such formulas commonly contain comparison operators (such as <, >, and =) and test functions such as **IsBlank** and **IsEmpty**.
- *ThenResult(s)* - Required. The corresponding value to return for a condition that evaluates to **true**.
- *DefaultResult* - Optional. The value to return if no condition evaluates to **true**. If you don't specify this argument, *blank* is returned.

Switch(*Formula*, *Match1*, *Result1* [, *Match2*, *Result2*, ... [, *DefaultResult*]])

- *Formula* - Required. Formula to evaluate for matches. This formula is evaluated only once.
- *Match(s)* - Required. Values to compare with the result from *Formula*. If an exact match is found, the corresponding *Result* is returned.
- *Result(s)* - Required. The corresponding value to return when an exact match is found.
- *DefaultResult* - Optional. If an exact match isn't found, this value is returned. If you don't specify this argument, *blank* is returned.

Examples

Values in formulas

In the following examples, a **Slider** control (named **Slider1**) has a value of **25**.

Expand table

Formula	Description	Result
If(Slider1.Value = 25, "Result1")	The condition is **true**, and the corresponding result is returned.	"Result1"
If(Slider1.Value = 25, "Result1", "Result2")	The condition is **true**, and the corresponding result is returned.	"Result1"
If(Slider1.Value > 1000, "Result1")	The condition is **false**, and no *DefaultResult* was provided.	*blank*
If(Slider1.Value > 1000, "Result1", "Result2")	The condition is **false**, a *DefaultResult* was provided, and it's returned.	"Result2"

Formula	Description	Result
If(Slider1.Value = 25, "Result1", Slider1.Value > 0, "Result2")	The first condition is **true**, and the corresponding result is returned. The second condition is also **true**, but it isn't evaluated because it appears later in the argument list than a condition that evaluates to **true**.	"Result1"
If(IsBlank(Slider1.Value), "Result1", IsNumeric(Slider1.Value), "Result2")	The first condition is **false** because the slider isn't *blank*. The second condition is **true** because the slider's value is a number, and the corresponding result is returned.	"Result2"
If(Slider1.Value > 1000, "Result1", Slider1.Value > 50, "Result2", "Result3")	Both the first and second conditions are **false**, a *DefaultResult* was provided, and it's returned.	"Result3"
Switch(Slider1.Value, 25, "Result1")	The slider's value matches the first value to be checked, and the corresponding result is returned.	"Result1"
Switch(Slider1.Value, 20, "Result1", 25, "Result2", 30, "Result3")	The slider's value matches the second value to be checked, and the corresponding result is returned.	"Result2"
Switch(Slider1.Value, 20, "Result1", 10, "Result2", 0, "Result3", "DefaultResult")	The slider's value doesn't match any value to be checked. A *DefaultResult* was provided, so it's returned.	"DefaultResult"

Branching in behavior formulas

In these examples, a Text input control named **FirstName** has the value "John" typed into it.

Expand table

Formula	Description	Result
If(! IsBlank(FirstName.Text), Navigate(Screen1, ScreenTransition.None))	The condition is **true**, so the Navigate function runs. You can use the IsBlank function to test whether a required form field has been filled in. If **FirstName** were blank, this formula would have no effect.	**true** The display is changed to **Screen1**.
If(IsBlank(FirstName.Text), Navigate(Screen1,	Without the **!** operator, the condition is **false**, so the Navigate	**true**

Formula	Description	Result
ScreenTransition.None), Back())	function doesn't run. The **Back** function was provided as a *DefaultResult*, so it runs.	The display goes back to the screen that was previously shown.
Switch(FirstName.Text, "Carlos", Navigate(Screen1, ScreenTransition.None), "Kirstin", Navigate(Screen2, ScreenTransition.None), "John", Navigate(Screen3, ScreenTransition.None))	The value of **FirstName.Text** is compared against "Carlos", "Kirstin", and "John" in that order. A match is found with "John", so the app navigates to **Screen3**.	**true** The display is changed to **Screen3**.

Step by step

1. Add a **Text input** control, and name it **Text1** if it doesn't have that name by default.
2. In **Text1**, type **30**.
3. Add a **Label** control, and set its **Text** property to this formula:
   If(Value(Text1.Text) < 20, "Order MANY more!", Value(Text1.Text) < 40, "Order more!", Text1.Text)

   The **Label** control shows **Order more!** because the value of **Text1** is more than 20 but less than 40.
4. In **Text1**, type **15**.

   The **Label** control shows **Order MANY more!** because the value of **Text1** is less than 20.
5. In **Text1**, type **50**.

   The **Label** control shows the value that you typed because it's more than 40.

Feedback

Was this page helpful? Yes No

Provide product feedback

Table function

06/19/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Creates a temporary table.

Description

The **Table** function creates a table from an argument list of records or tables.

The result table's columns are the union of all the columns from all the argument records and tables. A *blank* value is added to any column for which a record doesn't include a value.

A table is a value in Power Apps, just like a string or a number. You can specify a table as an argument for a function, and functions can return a table as a result. **Table** doesn't create a permanent table. Instead it returns a temporary table made of its arguments. You can specify this temporary table as an argument for another function, visualize it in a gallery, or embed it in another table. Fore more information, see working with tables.

You can also create a single-column table with the **[value1, value2, ...]** syntax.

Syntax

Table(*RecordOrTable1* [, *RecordOrTable2*, ...])

- *RecordOrTable(s)* - Required. The records or table to add to the result table. If a table is provided, the records of the table are added to the resulting table as if they had been individually added.

Table(*Dynamic*)

- *Dynamic* - Required. **Dynamic** value that represents a table or array. Acceptable values are dependent on the untyped provider. For **JSON**, the dynamic value is expected to be a JSON array. Regardless of the content type of the Untyped array, the resulting table is a single-column table of dynamic values.

Examples

- Set the **Items** property of a listbox to this formula:

Power Fx

```
Table( {Color: "red"}, {Color: "green"}, {Color: "blue" } )
```

The listbox shows each color as an option.

- Add a text gallery, and set its **Items** property to this function:

  Power Fx

  ```
  Table( {Item: "Violin123", Location:"France", Owner:"Fabrikam"},
  {Item:"Violin456", Location:"Chile"} )
  ```

  The gallery shows two records, both of which contain the name and location of an item. Only one record contains the name of the owner.

- This formula combines standard sizes with extended sizes into a single table

  Power Fx

  ```
  Table( { Value: "XS" }, [ "S", "M", "L" ], { Value: "XL" } )
  ```

Acos, Acot, Asin, Atan, Atan2, Cos, Cot, Degrees, Pi, Radians, Sin, and Tan functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates trigonometric values.

Description

Primary functions

The **Cos** function returns the cosine of its argument, an angle specified in radians.

The **Cot** function returns the cotangent of its argument, an angle specified in radians.

The **Sin** function returns the sine of its argument, an angle specified in radians.

The **Tan** function returns the tangent of its argument, an angle specified in radians.

Inverse functions

The **Acos** function returns the arccosine, or inverse cosine, of its argument. The arccosine is the angle whose cosine is the argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Acot** function returns the principal value of the arccotangent, or inverse cotangent, of its argument. The returned angle is given in radians in the range 0 (zero) to π.

The **Asin** function returns the arcsine, or inverse sine, of its argument. The arcsine is the angle whose sine is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan** function returns the arctangent, or inverse tangent, of its argument. The arctangent is the angle whose tangent is the argument. The returned angle is given in radians in the range -π/2 to π/2.

The **Atan2** function returns the arctangent, or inverse tangent, of the specified *x* and *y* coordinates as arguments. The arctangent is the angle from the *x*-axis to a line that contains the origin (0, 0) and a point with coordinates (*x*, *y*). The angle is given in radians between -π and π, excluding -π. A positive result represents a counterclockwise angle from the *x*-axis; a negative result represents a clockwise angle. **Atan2(*a*, *b*)** equals **Atan(*b*/*a*)**, except that ***a*** can equal 0 (zero) with the **Atan2** function.

Helper functions

The **Degrees** function converts radians to degrees. π radians equals 180 degrees.

The **Pi** function returns the transcendental number π, which begins 3.141592...

The **Radians** function converts degrees to radians.

Notes

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results with a **Value** column, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

If an argument would result in an undefined value, the result is *blank*. This can happen, for example, when using inverse functions with arguments that are out of range.

Syntax

Primary Functions

Cos(*Radians*)
Cot(*Radians*)
Sin(*Radians*)
Tan(*Radians*)

- *Radians* - Required. Angle to operate on.

Cos(*SingleColumnTable*)
Cot(*SingleColumnTable*)
Sin(*SingleColumnTable*)
Tan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of angles to operate on.

Inverse Functions

Acos(*Number*)
Acot(*Number*)
Asin(*Number*)
Atan(*Number*)

- *Number* - Required. Number to operate on.

Acos(*SingleColumnTable*)
Acot(*SingleColumnTable*)
Asin(*SingleColumnTable*)
Atan(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of numbers to operate on.

Atan2(*X*, *Y*)

- *X* - Required. *X*-axis coordinate.
- *Y* - Required. *Y*-axis coordinate.

Helper Functions

Degrees(*Radians*)

- *Radians* - Required. Angle in radians to convert to degrees.

Pi()

Radians(*Degrees*)

- *Degrees* - Required. Angle in degrees to convert to radians.

Examples

Single number

Expand table

Formula	Description	Result
Cos(1.047197)	Returns the cosine of 1.047197 radians or 60 degrees.	0.5
Cot(Pi()/4)	Returns the cotangent of 0.785398... radians or 45 degrees.	1
Sin(Pi()/2)	Returns the sine of 1.570796... radians or 90 degrees.	1
Tan(Radians(60))	Returns the tangent of 1.047197... radians or 60 degrees.	1.732050...
Acos(0.5)	Returns the arccosine of 0.5, in radians.	1.047197...
Acot(1)	Returns the arccotangent of 1, in radians.	0.785398...
Asin(1)	Returns the arcsine of 1, in radians.	1.570796...
Atan(1.732050)	Returns the arctangent of 1.732050, in radians.	1.047197...
Atan2(5, 3)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (5,3), which is approximately 31 degrees.	0.540419...
Atan2(4, 4)	Returns the arctangent of the angle from the *x*-axis of the line that contains the origin (0,0) and the coordinate (4,4), which is exactly π/4 radians or 45 degrees.	0.785398...
Degrees(1.047197)	Returns the equivalent number of degrees for 1.047197 radians.	60
Pi()	Returns the transcendental number π.	3.141592...
Radians(15)	Returns the equivalent number of radians for 15 degrees.	0.261799...

Single-column table

The examples in this section use a data source that's named **ValueTable** and that contains the following data. The last record in the table is π/2 radians or 90 degrees.

Expand table

Value
0.5
-2
1.570796...

Expand table

Formula	Description	Result
Cos(ValueTable)	Returns the cosine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.877582..., -0.416146..., 0
Cot(ValueTable)	Returns the cotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.830487..., 0.457657..., 0
Sin(ValueTable)	Returns the sine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.479425, -0.909297..., 1
Tan(ValueTable)	Returns the tangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.546302..., 2.185039..., 3060023.306952...
Acos(ValueTable)	Returns the arccosine of each number in the table.	A single-column table with a `Value` column containing the following values: 1.047197..., Blank(), Blank()
Acot(ValueTable)	Returns the arccotangent of each number in the table.	A single-column table with a `Value` column containing the following values: 1.107138..., 2.677945..., 0.566911...
Asin(ValueTable)	Returns the arcsine of each number in the table.	A single-column table with a `Value` column containing the following values: 0.523598..., Blank(), Blank()
Atan(ValueTable)	Returns the arctangent of each number in the table.	A single-column table with a `Value` column containing the following values: 0.463647..., -1.107148..., 1.00388...
Degrees(ValueTable)	Returns the equivalent number of degrees for each number in the table, assumed to be angles in radians.	A single-column table with a `Value` column containing the following values: 28.647889..., -114.591559..., 90
Radians(ValueTable)	Returns the equivalent number of radians for each number in the table, assumed to be angles in degrees.	A single-column table with a `Value` column containing the following values: 0.008726..., -0.034906..., 0.027415...

Feedback

Was this page helpful? Yes No

Provide product feedback

Text function

06/18/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts any value and formats a number or date/time value to a string of text.

Description

The **Text** function formats a number or a date/time value based on one of these types of arguments:

- A predefined date/time format, which you specify by using the **DateTimeFormat** enumeration. For dates and times, this approach is preferred as it automatically adjusts to each user's language and region.
- A custom format, which comprises a string of placeholders that define, for example, whether numbers show a decimal separator and dates show the full name of the month, the month as an abbreviation, or the month as a number. Power Apps supports a subset of the placeholders that Microsoft Excel does. In this string, the language placeholder specifies the language in which to interpret the other placeholders. If the custom format includes a period, for example, the language-format placeholder specifies whether the period is a decimal separator (ja-JP) or a thousands separator (es-ES).

See working with dates and times for more information.

The **Text** function can also convert any data type to a text representation using a default format. Use this to pass non-text values to text-based functions such as **Len**, **Right**, and **IsMatch**.

Predefined date/time formats

For these examples, date and time used is Tuesday, April 7, 2020 8:26:59.180 PM, in the time zone UTC-7 hours.

Expand table

DateTimeFormat enum	Description	Examples (using en-US)
LongDate	Four-digit year, month name, day of the month, and day of the week. The names of the month and day of the week aren't abbreviated.	"Tuesday, April 7, 2020"
LongDateTime	Four-digit year, month name, day of the month, and day of the week, plus hour (12-hour clock), minutes, seconds, and AM/PM designation. The names of the month and day of the week aren't abbreviated.	"Tuesday, April 7, 2020 8:26:59 PM"
LongDateTime24	Four-digit year, month, day of the month, and day of the week, plus hour (24-hour clock), minutes, and seconds. The names of the month and day of the week aren't abbreviated.	"Tuesday, April 7, 2020 20:26:59"
LongTime	Hour (12-hour clock), minutes, seconds, and AM/PM designation.	"8:26:59 PM"
LongTime24	Hour (24-hour clock), minutes, seconds.	"20:26:59"
ShortDate	Four-digit year with numerical month and day of the month.	"4/7/2020"
ShortDateTime	Four-digit year with numerical month and day of the month, plus hour (12-hour clock), minutes, and AM/PM designation.	"4/7/2020 8:26 PM"
ShortDateTime24	Four-digit year with numerical month and day of the month, plus hour (24-hour clock) and minutes.	"4/7/2020 20:26"
ShortTime	Hour (12-hour clock), minutes, and AM/PM designation.	"8:26 PM"
ShortTime24	Hour (24-hour clock) and minutes.	"20:26"
UTC	The date/time value is converted to UTC based on the current user's time zone and formatted according to the ISO 8601 standard.	"2020-04-08T03:26:59.180Z"

Number placeholders

Expand table

Placeholder	Description
0 (*zero*)	Displays insignificant zeros if a number has fewer digits than there are zeros in the format. For example, use the format **#.00** if you want to display **8.9** as **8.90**.

Placeholder	Description
#	Follows the same rules as the **0** (zero). However, **Text** doesn't return extra zeros when the number has fewer digits on either side of the decimal than there are # symbols in the format. For example, **8.9** is displayed if the custom format is **#.##** and the number to format is **8.9**.
. (*period*)	Displays the decimal point in a number. Depends on the language of the custom format; see global apps for more details.
, (*comma*)	Displays the grouping separator in a number, often used for thousands. **Text** separates groups by commas if the format contains a comma that's enclosed by number signs (#) or by zeros. Depends on the language of the custom format; see global apps for more details.

If a number has more digits to the right of the decimal point than there are placeholders in the format, the number rounds to as many decimal places as there are placeholders. If there are more digits to the left of the decimal point than there are placeholders, the extra digits are displayed. If the format contains only number signs (#) to the left of the decimal point, numbers less than 1 start with a decimal point (for example, **.47**).

Date and time placeholders

Expand table

Placeholder	Description
m	Displays the month as a number without a leading zero.
mm	Displays the month as a number with a leading zero when appropriate.
mmm	Displays the month as an abbreviation (**Jan** to **Dec**).
mmmm	Displays the month as a full name (**January** to **December**).
d	Displays the day as a number without a leading zero.
dd	Displays the day as a number with a leading zero when appropriate.
ddd	Displays the day as an abbreviation (**Sun** to **Sat**).
dddd	Displays the day as a full name (**Sunday** to **Saturday**).
yy	Displays the year as a two-digit number.
yyyy	Displays the year as a four-digit number.
h	Displays the hour as a number without a leading zero.

Placeholder	Description
hh	Displays the hour as a number with a leading zero when appropriate. If the format contains **AM** or **PM**, the hour is shown based on the 12-hour clock. Otherwise, the hour is shown based on the 24-hour clock.
m	Displays the minute as a number without a leading zero. This placeholder must appear immediately after the **h** or **hh** code or immediately before the **ss** code; otherwise, **Text** returns the month instead of minutes.
mm	Displays the minute as a number with a leading zero when appropriate. This placeholder must appear immediately after the **h** or **hh** placeholder or immediately before the **ss** placeholder. Otherwise, **Text** returns the month instead of minutes.
s	Displays the second as a number without a leading zero.
ss	Displays the second as a number with a leading zero when appropriate.
f	Displays the fractions of seconds.
AM/PM, **a/p**	Displays the hour based on a 12-hour clock. **Text** returns "AM" or "a" for times from midnight until noon and "PM" or "p" for times from noon until midnight

Literal placeholders

You can include any of these characters in your format string. They will appear in the result of **Text** as is. Additional characters are reserved for future placeholders, so you shouldn't use them.

Expand table

Character	Description
Any currency symbol	Dollar sign, cents sign, euro sign, etc.
+	Plus sign
(	Left parenthesis
:	Colon
^	Circumflex accent (caret)
'	Apostrophe
{	Left curly bracket

Character	Description
<	Less-than sign
=	Equal sign
-	Minus sign
/	Slash mark
)	Right parenthesis
&	Ampersand
~	Tilde
}	Right curly bracket
>	Greater-than sign
	Space character

Global apps

The **Text** function is globally aware. For a wide array of languages, it knows how to properly write out dates, times, currencies, and numbers. To do its job, it needs two pieces of information:

- **The language of the custom format:** For makers, how should a custom format be interpreted? The separator characters (. and ,) have different meanings in different languages. If you specify a custom format, you can include a language placeholder or take the default value, which reflects the language to which your device is set. Even easier, you can use one of the predefined date/time formats, which are language agnostic.
- **The language of the result:** For users, in what language should the function result appear? Names of months and weekdays must be in the appropriate language for the user of the app, which you can specify by adding a third, optional argument to the **Text** function.

For both, you specify the language by using a language tag. To see the list of supported languages, type **Text(1234, "",)** in the formula bar or the **Advanced** tab of the right-hand pane, and then scroll through the list of locales suggested for the third argument.

Language placeholder

To specify the language of the custom format, use:

Expand table

Placeholder	Description
[$-*LanguageTag*]	*LanguageTag* is a language tag as returned from the **Language** function. It can specify just the language (such as **[$-en]** for English), or it can also specify the region (such as **[$-en-GB]** to further specify Great Britain).

The language placeholder can appear anywhere in the custom format but only once.

If you specify a custom format without a language placeholder and the format is ambiguous from a global standpoint, the language tag for your current language is inserted automatically.

[$-en-US] is assumed if this placeholder isn't present when your app is run.

> **Note**
>
> In a future version, the syntax of this placeholder may change to avoid confusion with a similar, but different, placeholder that Excel supports.

Result language tag

The result of **Text** includes translated strings for months, weekdays, and AM/PM designations, as well as the appropriate group and decimal separators.

By default, **Text** uses the language of the user running the app. The **Language** function returns the language tag for the current user. You can override this default value by supplying a language tag for the third argument to **Text**.

Syntax

Text(*NumberOrDateTime*, *DateTimeFormatEnum* [, *ResultLanguageTag*])

- *NumberOrDateTime* - Required. The number or the date/time value to format.
- *DateTimeFormat* - Required. A member of the **DateTimeFormat** enumeration.
- *ResultLanguageTag* - Optional. The language tag to use for the result text. By default, the language of the current user is used.

Text(*NumberOrDateTime*, *CustomFormat* [, *ResultLanguageTag*])

- *Number* - Required. The number or the date/time value to format.
- *CustomFormat* - Required. One or more placeholders enclosed in double quotation marks.

- *ResultLanguageTag* - Optional. The language tag to use for the result text. By default, the language of the current user is used.

Text(*AnyValue*)

- *AnyValue* - Required. Value to convert to a text representation. A default format is used.

Text (*Dynamic*)

- *Dynamic* - Required. **Dynamic** value that represents a string. Acceptable values are dependent on the untyped provider. For **JSON**, if the dynamic value is a number or boolean, it will be converted to text.

Examples

Unless otherwise specified, the user running these formulas is located in the United States and has selected English as their language. The **Language** function is returning "en-US".

Number

Expand table

Formula	Description	Result
Text(1234.59, "####.#")	Formats the number with one decimal place.	"1234.6"
Text(8.9, "#.000")	Pads the decimal portion of the number with trailing zeros, if needed.	"8.900"
Text(0.631, "0.#")	Pads the whole portion of the number with leading zeros, if needed.	"0.6"
Text(12, "#.0#") **Text(1234.568, "#.0#")**	Pads the decimal portion of the number with zeros for one decimal place, and includes a second decimal place if supplied.	"12.0" "1234.57"
Text(12000, "$ #,###") **Text(1200000, "$ #,###")**	Places a thousands separator every three digits, and includes a currency symbol.	"$ 12,000" "$ 1,200,000"

Date/Time

- At **2:37:47 PM** on **Monday, November 23, 2015**
- United States Pacific Time Zone (UTC-8)

Expand table

Formula	Description	Result
Text(Now(), DateTimeFormat.LongDate)	Formats as a long date string, in the language and locale of the current user.	"Monday, November 23, 2015"
Text(Now(), DateTimeFormat.LongDateTime)	Formats as a long date and time string, in the language and locale of the current user, using a 12-hour clock.	"Monday, November 23, 2015 2:37:47 PM"
Text(Now(), DateTimeFormat.LongTime24)	Formats as a long time string, using a 24-hour clock.	"14:37:47"
Text(Now(), DateTimeFormat.ShortDate)	Formats as a short date string, in the language and locale of the current user.	"11/23/2015"
Text(Now(), "d-mmm-yy")	Formats using placeholder characters: • **d** for a single-digit or double-digit day of the month • **-** as a literal character copied to the result • **mmm** for a three-letter abbreviation of the month • **-** as another literal character copied to the result • **yy** for a two-digit abbreviation of the year	"23-Nov-15"
Text(1448318857*1000, "mmm. dd, yyyy (hh:mm:ss AM/PM)")	Shows a Unix date-time value in human-readable format if you multiply the source value by 1,000.	"Nov. 23, 2015 (02:47:37 PM)"

Global apps

Expand table

Formula	Description	Result
Text(1234567.89, "[$-fr-FR]# ###,## €", "fr-FR")	Shows a space as a grouping separator, the comma as a decimal separator, and **€** as the currency symbol.	"1 234 567,89 €"
Text(1234567,89; "[$-fr-FR]# ###,## €")	If the source data follows the French custom of using a comma as the decimal separator, you must change your locale to French and separate the arguments with a	"1 234 567,89 €"

Formula	Description	Result
	semi-colon instead of a comma to get the same result as above.	
Text(Date(2016,1,31), "dddd mmmm d")	Returns the weekday, month, and day of the month in the language of the current user. Because none of the placeholders are language dependent, there is no need for a format text language tag.	"Sunday January 31"
Text(Date(2016,1,31), "dddd mmmm d", "es-ES")	Returns the weekday, month, and day of the month in the "es-ES" language.	"domingo enero 31"

Converting values to text

Expand table

Formula	Description	Result
Text(1234567.89)	Converts a number to a string. There are no thousands separators or control over the number of digits before or after the decimal separator; for more control, supply number placeholders as the second argument.	"1234567.89"
Text(DateTimeValue("01/04/2003"))	Converts a date/time value to a string of text. To control the conversion, provide either a member of the DateTimeFormat enumeration or a custom-format string.	"1/4/2003 12:00 AM"
Text(true)	Converts a Boolean value to a string.	"true"
Text(GUID())	Converts a generated GUID value to a string.	"f8b10550-0f12-4f08-9aa3-bb10958bc3ff"
Left(Text(GUID()), 4)	Returns the first four characters of a generated GUID.	"2d9c"

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

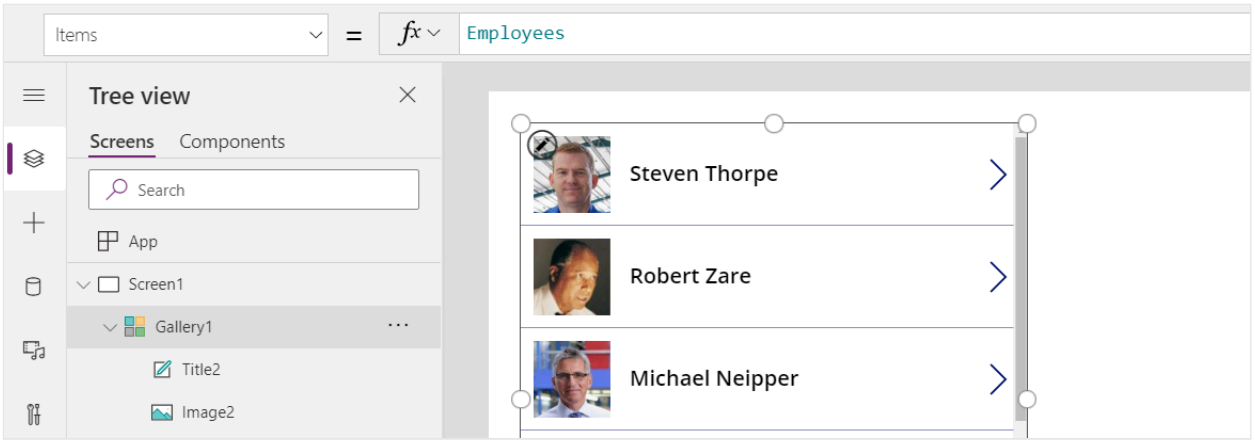
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

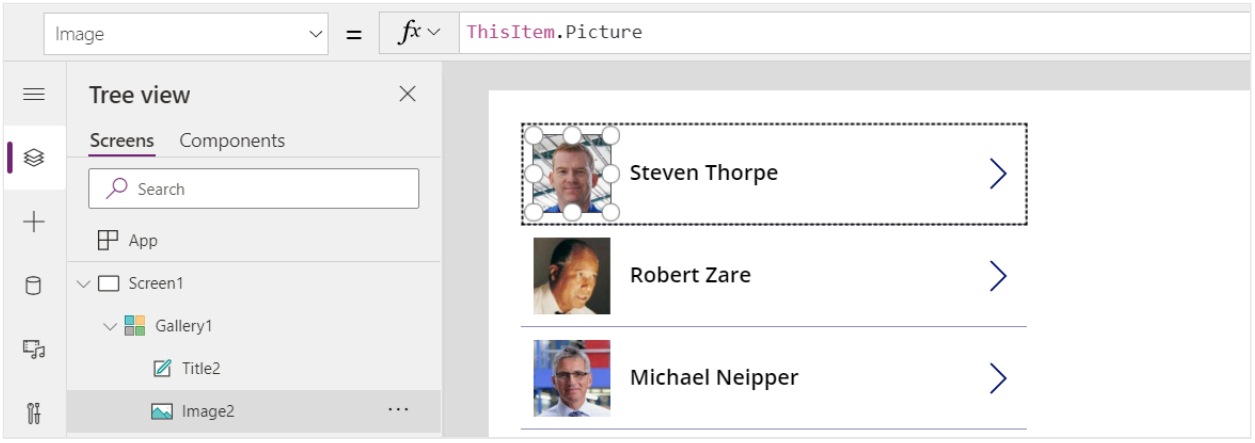
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

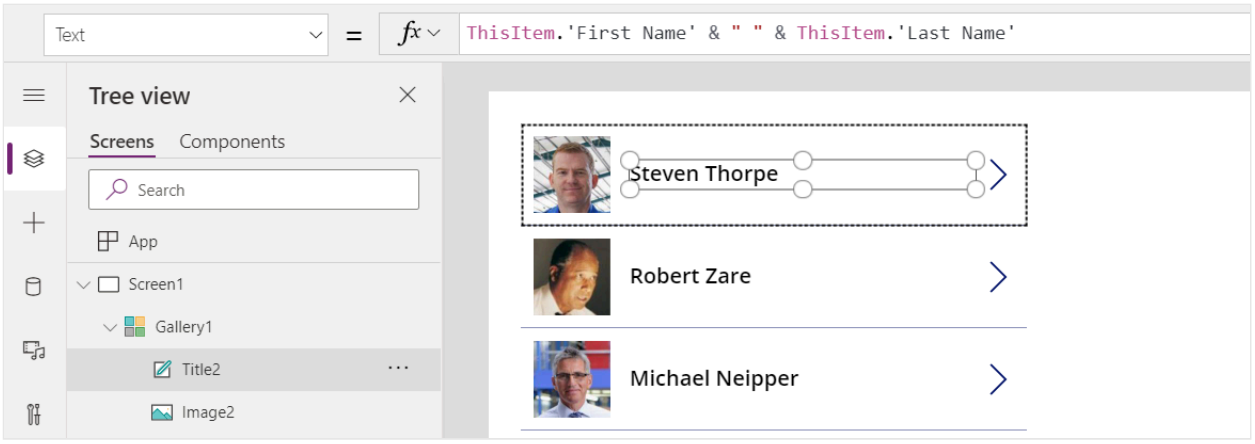
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

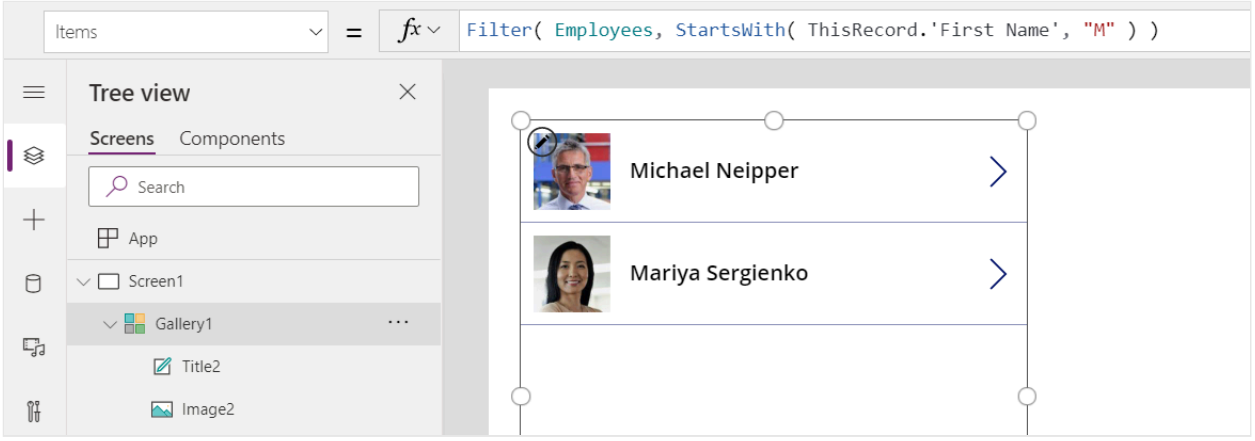


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

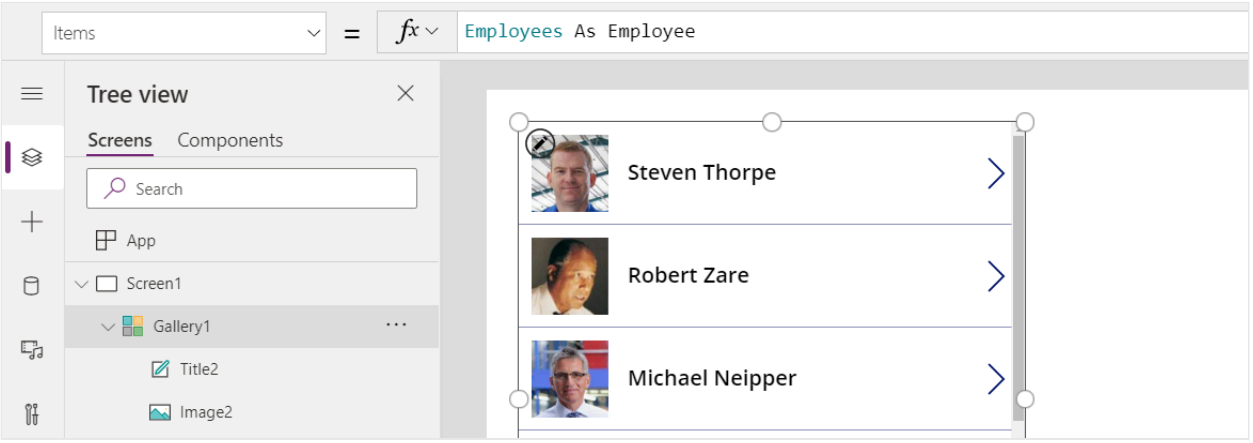
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

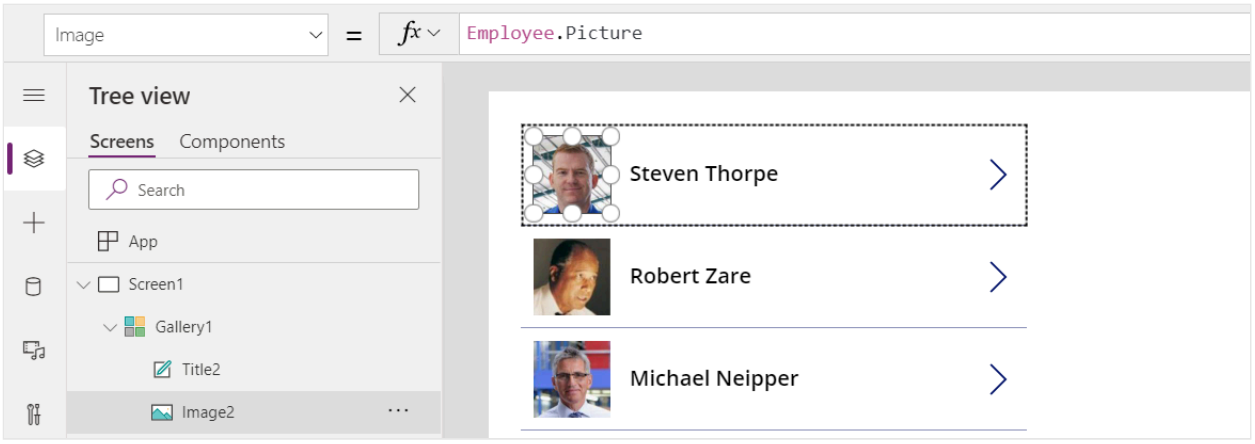
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

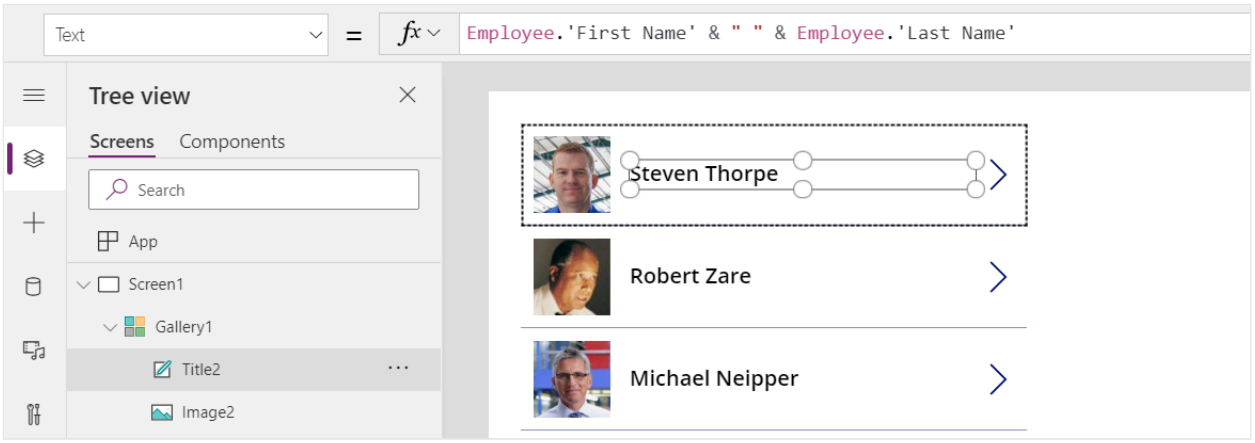
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

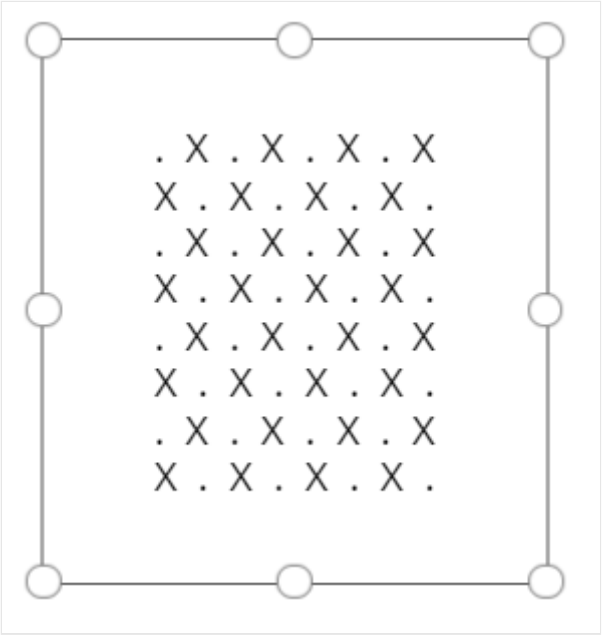
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



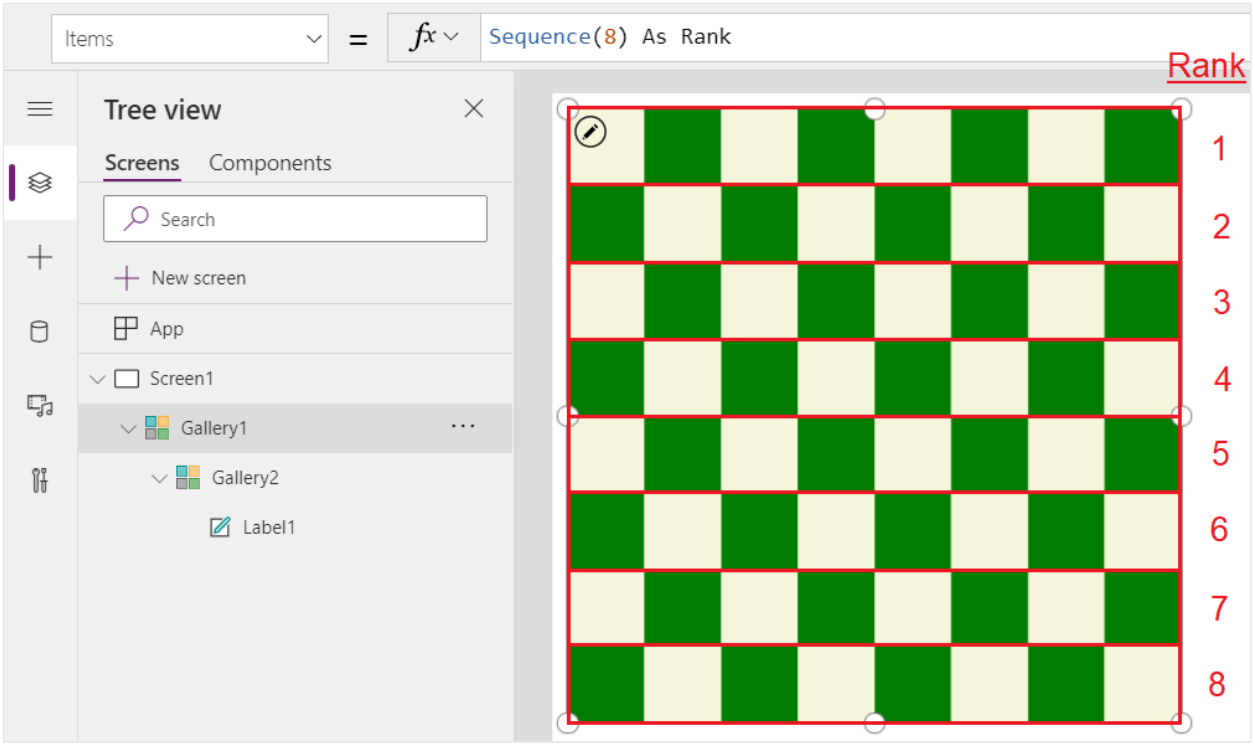
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

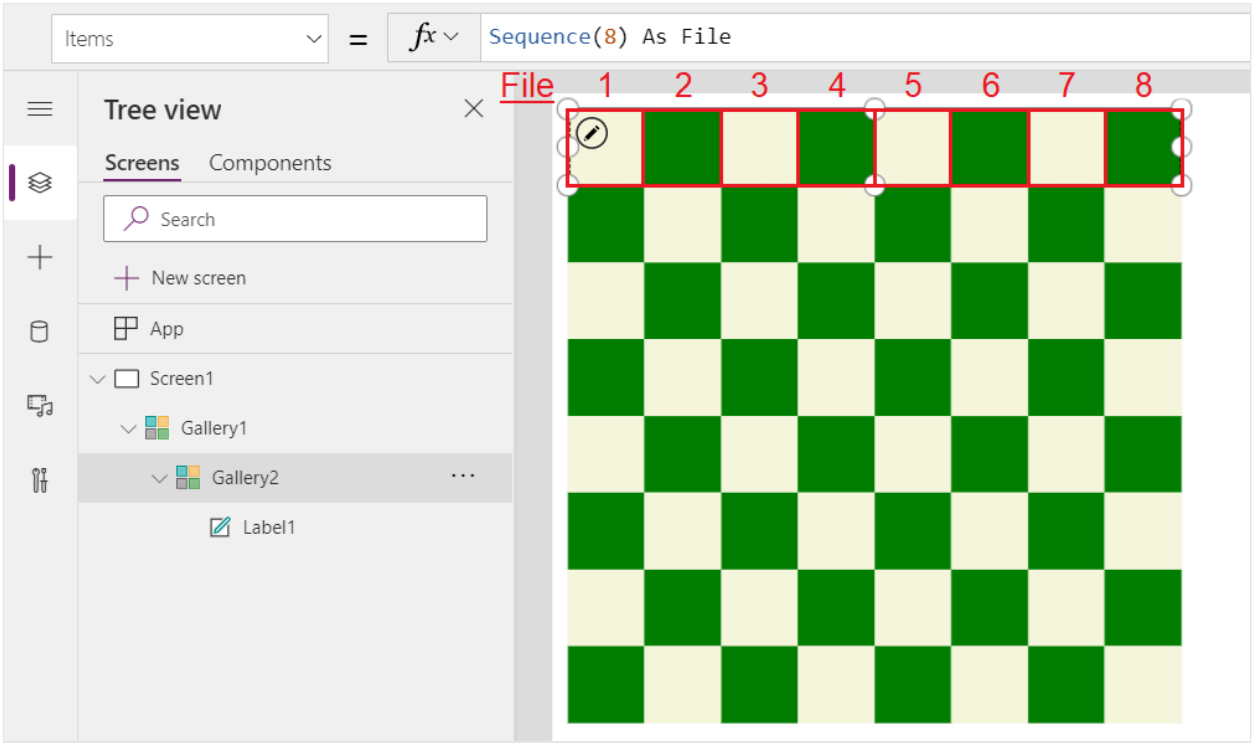
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

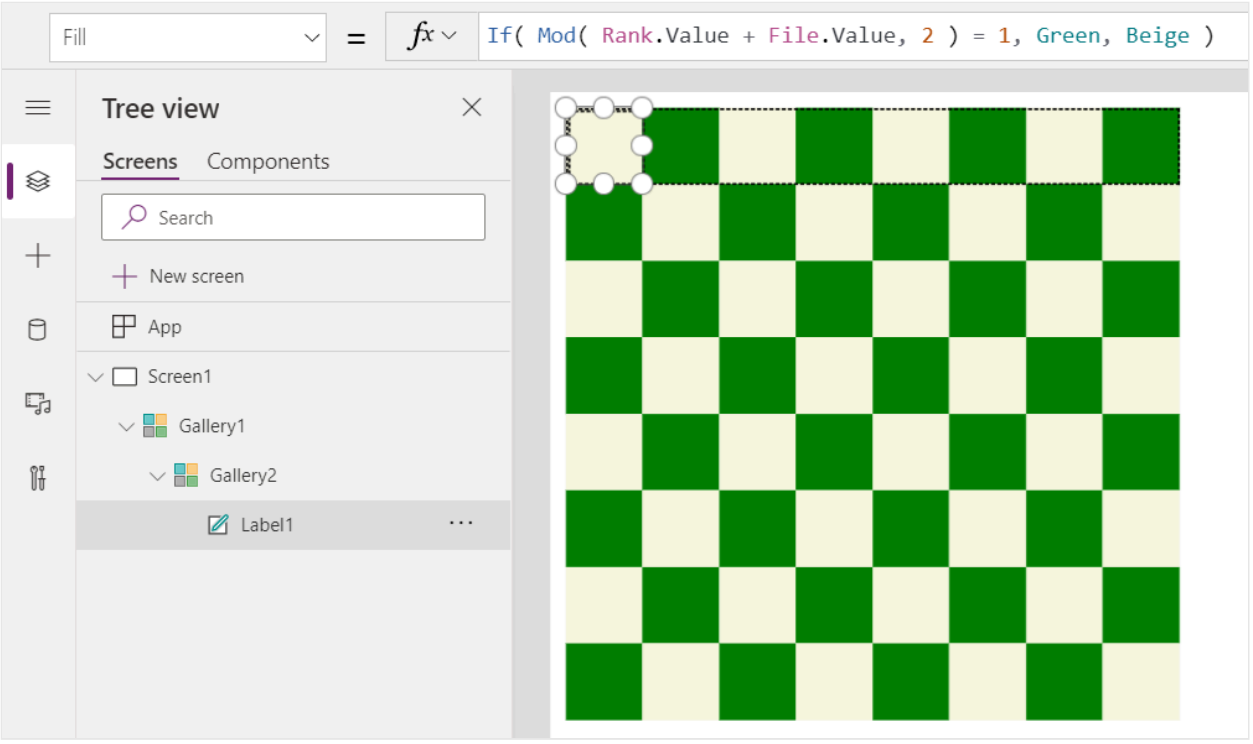
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

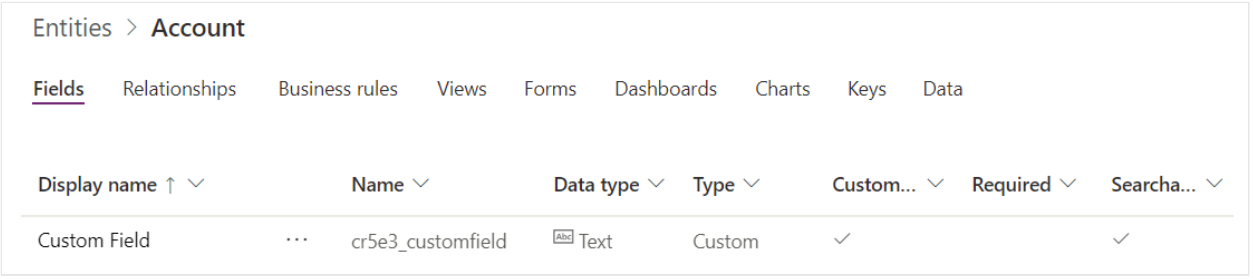
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

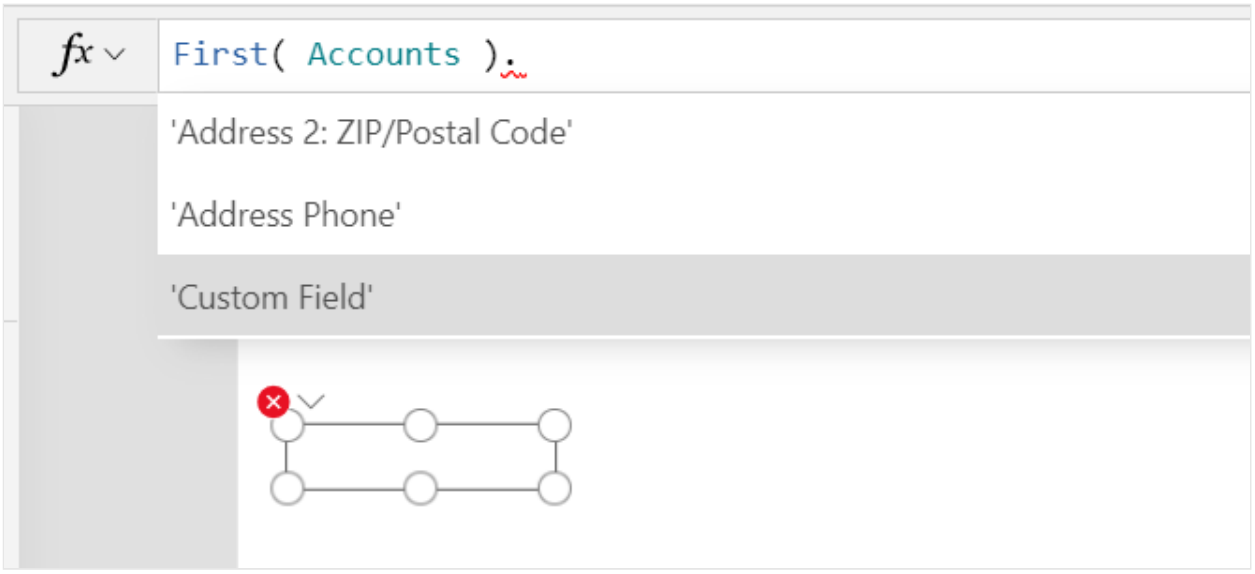
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

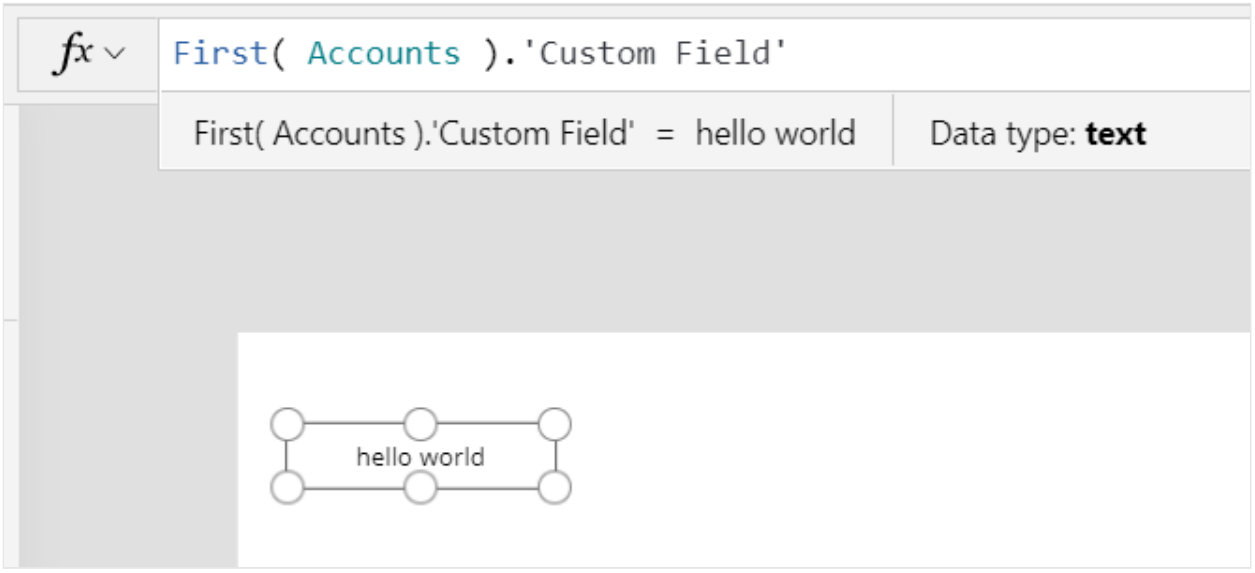
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



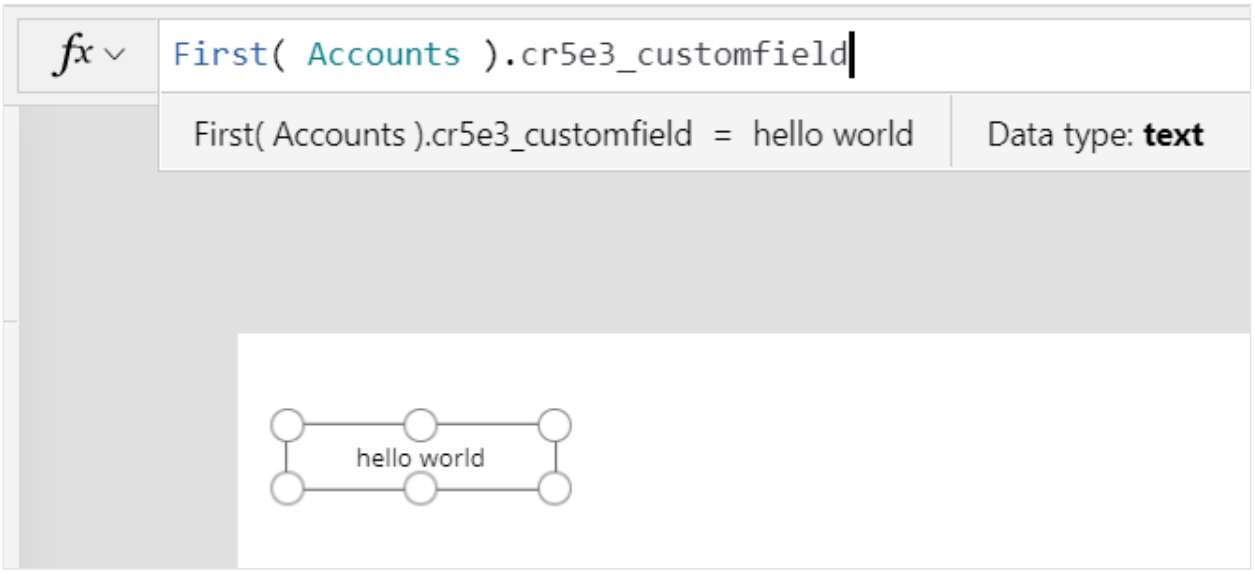
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

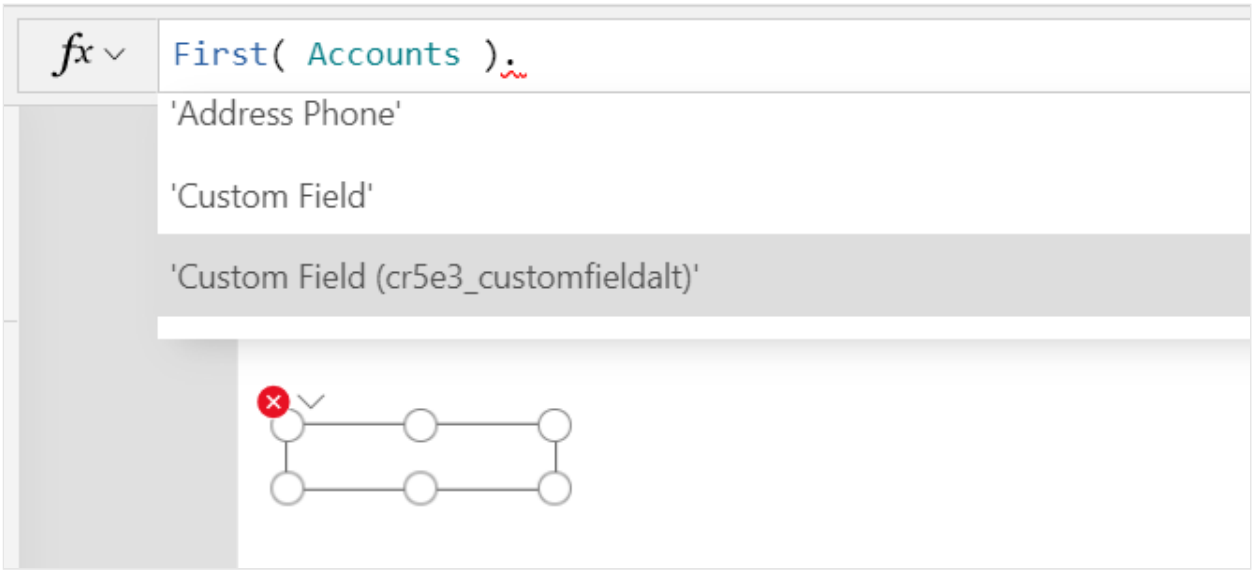


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

Operators and Identifiers in Power Apps

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Some of these operators are dependent on the language of the author. For more information about language support in canvas apps, see Global apps.

Expand table

Symbol	Type	Example	Description
'...'	Identifier	**'Account Name'**	Identifiers that contain special characters, including spaces, are enclosed in single quotes
"..."	Text string	**"Hello, World"**	Text strings are enclosed in double quotes
$"..."	String interpolation	**$"Dear {FirstName},"**	Formulas embedded within a text string
.	Property Selector	**Slider1.Value** **Color.Red** **Acceleration.X**	Extracts a property from a table, control, signal, or enumeration. For backward compatibility, **!** may also be used.
. [language dependent]	Decimal separator	**1.23**	Separator between whole and fractional parts of a number. The character depends on the language.
( )	Parentheses	**Filter(T, A < 10)** **(1 + 2) * 3**	Enforces precedence order, and groups subexpressions in a larger expression
+	Arithmetic operators	**1 + 2**	Addition
-		**2 - 1**	Subtraction and sign
*		**2 * 3**	Multiplication
/		**2 / 3**	Division (also see the **Mod** function)
^		**2 ^ 3**	Exponentiation, equivalent to the **Power** function

Symbol	Type	Example	Description
%		**20%**	Percentage (equivalent to "* 1/100")
=	Comparison operators	**Price = 100**	Equal to
>		**Price > 100**	Greater than
>=		**Price >= 100**	Greater than or equal to
<		**Price < 100**	Less than
<=		**Price <= 100**	Less than or equal to
<>		**Price <> 100**	Not equal to
&	String concatenation operator	**"hello" & " " & "world"**	Makes multiple strings appear continuous
&& or **And**	Logical operators	**Price < 100 && Slider1.Value = 20** or **Price < 100 And Slider1.Value = 20**	Logical conjunction, equivalent to the **And** function
|| or **Or**		**Price < 100 || Slider1.Value = 20** or **Price < 100 Or Slider1.Value = 20**	Logical disjunction, equivalent to the **Or** function
! or **Not**		**!(Price < 100)** or **Not (Price < 100)**	Logical negation, equivalent to the **Not** function
exactin	Membership operators	**Gallery1.Selected exactin SavedItems**	Belonging to a collection or a table
exactin		**"Windows" exactin "To display windows in the Windows operating system..."**	Substring test (case-sensitive)
in		**Gallery1.Selected in SavedItems**	Belonging to a collection or a table
in		**"The" in "The keyboard and the monitor..."**	Substring test (case-insensitive)
@	Disambiguation operator	**MyTable[@fieldname]**	Field disambiguation
@		**[@MyVariable]**	Global disambiguation

Symbol	Type	Example	Description
, [language dependent]	List separator	**If(X < 10, "Low", "Good")** **{ X: 12, Y: 32 }** **[1, 2, 3]**	Separates: • arguments in function calls • fields in a record • records in a table This character depends on the language.
; [language dependent]	Formula chaining	**Collect(T, A); Navigate(S1, "")**	Separate invocations of functions in behavior properties. The chaining operator depends on the language.
As	As operator	**AllCustomers As Customer**	Overrides **ThisItem** and **ThisRecord** in galleries and record scope functions. **As** is useful for providing a better, specific name and is especially important in nested scenarios.
Self	Self operator	**Self.Fill**	Access to properties of the current control
Parent	Parent operator	**Parent.Fill**	Access to properties of a control container
ThisItem	ThisItem operator	**ThisItem.FirstName**	Access to fields of a Gallery or form control
ThisRecord	ThisRecord operator	**ThisRecord.FirstName**	Access to the complete record and individual fields of the record within **ForAll**, **Sum**, **With**, and other record scope functions. Can be overridden with the **As** operator.

> ⓘ Note
>
> The @ operator can also be used to validate the type of the record object against a data source. For example, `Collect(coll,Account@{'Account Number: 1111')`

in and exactin operators

Use the **in** and **exactin** operators to find a string in a data source, such as a collection or an imported table. The **in** operator identifies matches regardless of case, and the **exactin** operator identifies matches only if they're capitalized the same way. Here's an example:

1. Create or import a collection named **Inventory**, and show it in a gallery, as the first procedure in Show images and text in a gallery describes.

2. Set the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" in ProductName)

   The gallery shows all products except Callisto because the name of that product is the only one that doesn't contain the letter you specified.

3. Change the **Items** property of the gallery to this formula:
   Filter(Inventory, "E" exactin ProductName)

   The gallery shows only Europa because only its name contains the letter that you specified in the case that you specified.

ThisItem, ThisRecord, and As operators

A few controls and functions apply formulas to individual records of a table. To refer to the individual record in a formula, use one of the following:

Expand table

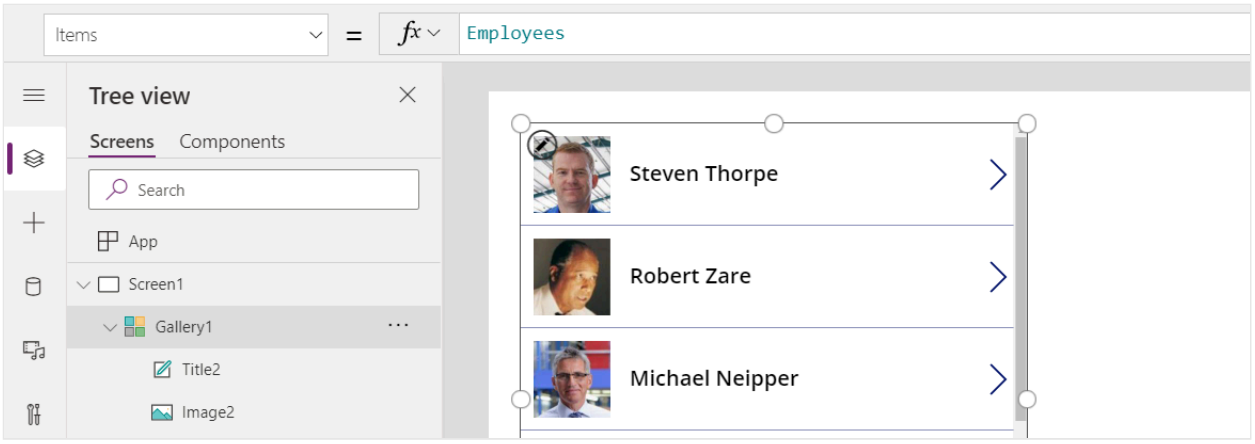
Operator	Applies to	Description
ThisItem	**Gallery** control **Edit form** control **Display form** control	The default name for the current record in a **Gallery** or form control.
ThisRecord	**ForAll**, **Filter**, **With**, **Sum** and other record scope functions	The default name for the current record in **ForAll** and other record scope functions.
As *name*	**Gallery** control **ForAll**, **Filter**, **With**, **Sum** and other record scope functions	Defines a *name* for the current record, replacing default **ThisItem** or **ThisRecord**. Use **As** to make formulas easier to understand and resolve ambiguity when nesting.

ThisItem operator

For example, in the following **Gallery** control, the **Items** property is set to the **Employees** data source (such as the **Employees** table included with the Northwind Traders sample):

Power Apps

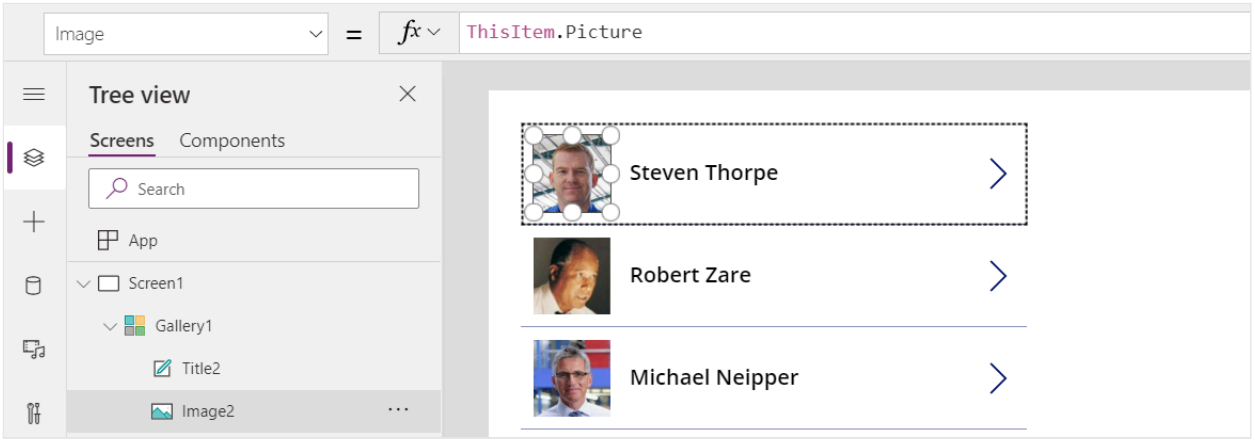
```
Employees
```



The first item in the gallery is a template that is replicated for each employee. In the template, the formula for the picture uses **ThisItem** to refer to the current item:

Power Apps

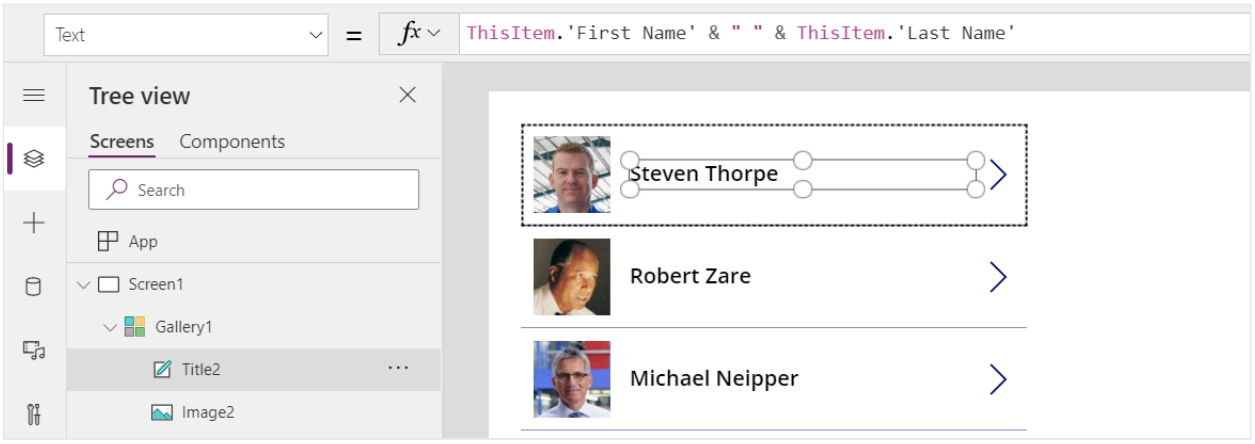
```
ThisItem.Picture
```



Likewise, the formula for the name also uses **ThisItem**:

Power Apps

```
ThisItem.'First Name' & " " & ThisItem.'Last Name'
```

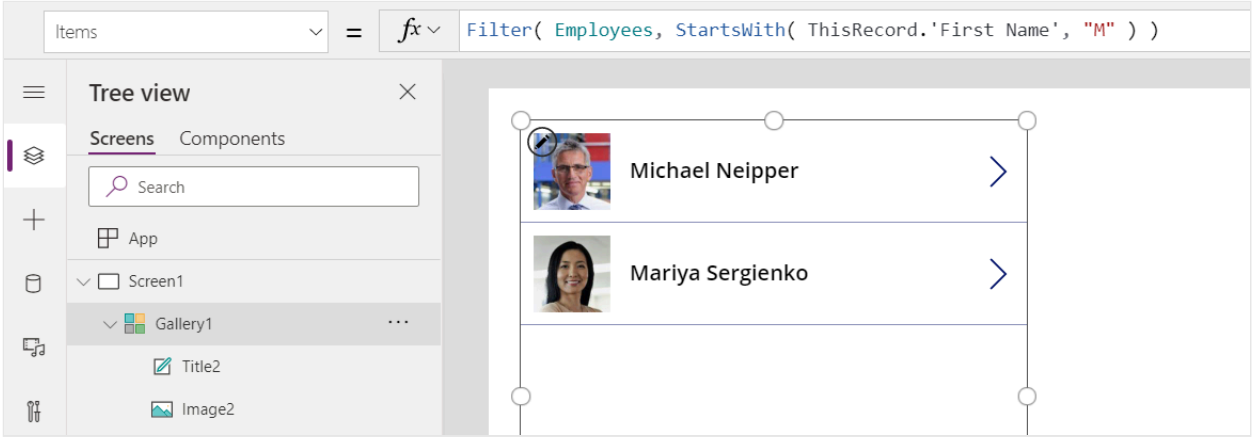


ThisRecord operator

ThisRecord is used in functions that have a record scope. For example, we can use the **Filter** function with our gallery's **Items** property to only show first names that being with *M*:

Power Apps

```
Filter( Employees, StartsWith( ThisRecord.Employee.'First Name', "M" ) )
```



ThisRecord is optional and implied by using the fields directly, for example, in this case, we could have written:

Power Apps

```
Filter( Employees, StartsWith( 'First Name', "M" ) )
```

Although optional, using **ThisRecord** can make formulas easier to understand and may be required in ambiguous situations where a field name may also be a relationship name. **ThisRecord** is optional while **ThisItem** is always required.

Use **ThisRecord** to reference the whole record with **Patch**, **Collect**, and other record scope functions. For example, the following formula sets the status for all inactive employees to active:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees,
              Patch( Employees, ThisRecord, { Status: 'Status
(Employees)'.Active } ) ) )
```

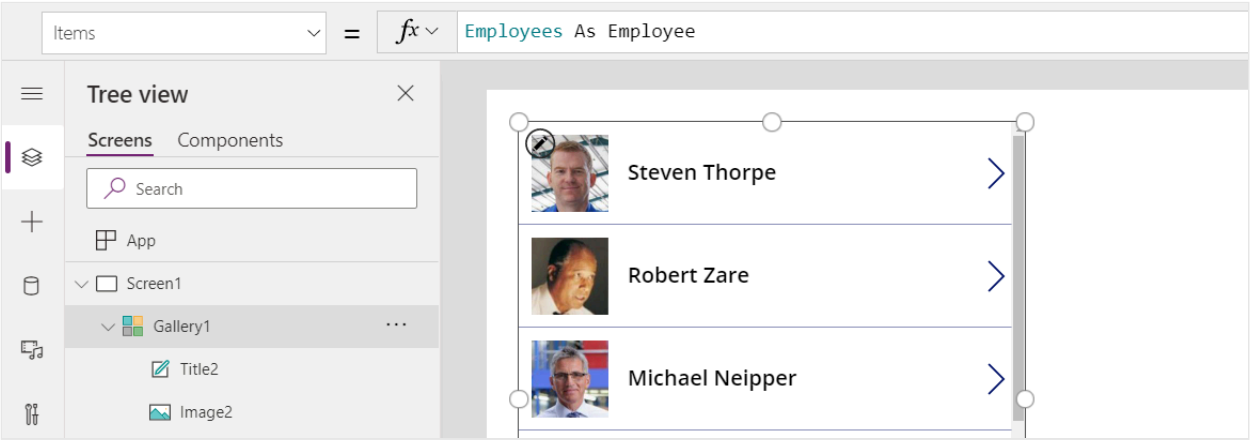
As operator

Use the **As** operator to name a record in a gallery or record scope function, overriding the default **ThisItem** or **ThisRecord**. Naming the record can make your formulas easier to understand and may be required in nested situations to access records in other scopes.

For example, you can modify the **Items** property of our gallery to use **As** to identify that we are working with an Employee:

Power Apps

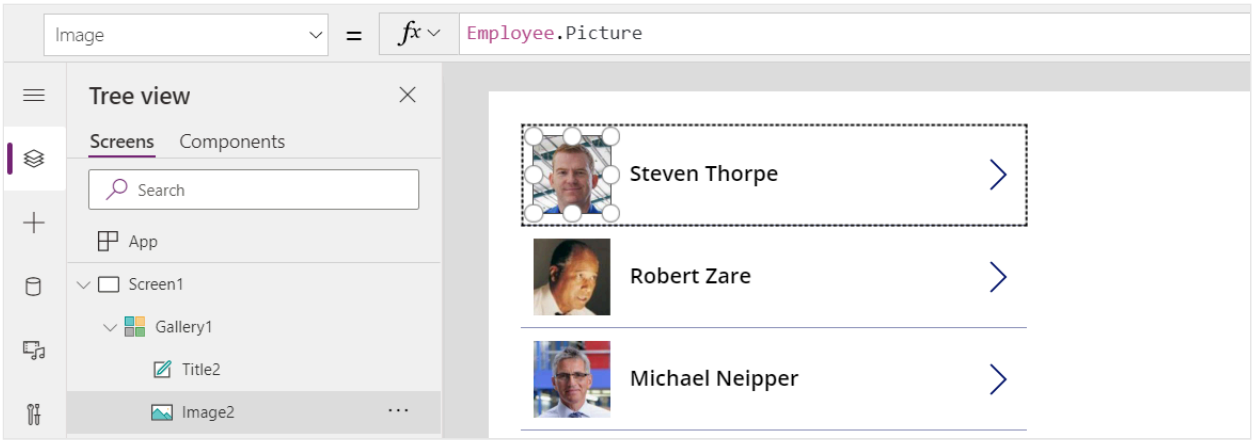
```
Employees As Employee
```



The formulas for the picture and name are adjusted to use this name for the current record:

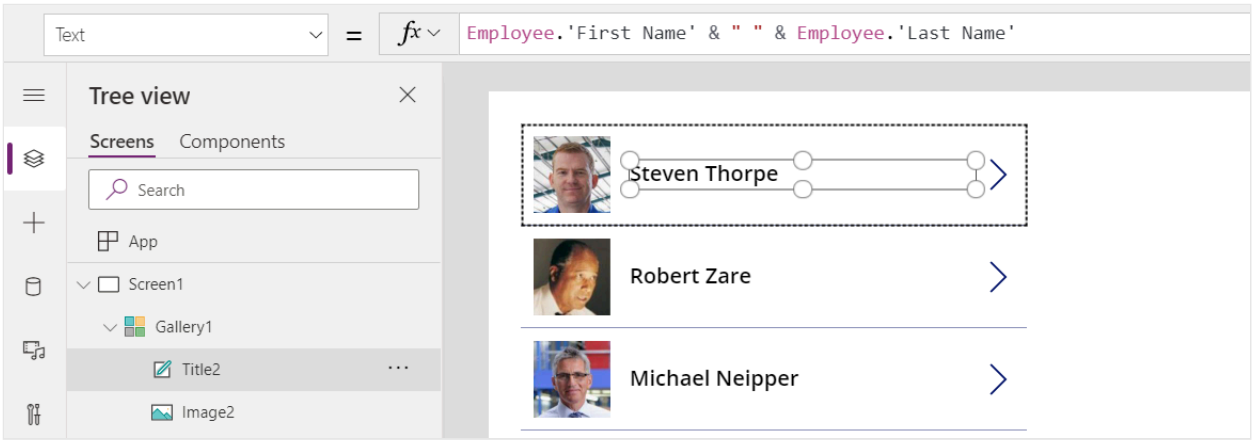
Power Apps

```
Employee.Picture
```



Power Apps

```
Employee.'First Name' & " " & Employee.'Last Name'
```



As can also be used with record scope functions to replace the default name **ThisRecord**. We can apply this to our previous example to clarify the record we're working with:

Power Apps

```
With( { InactiveEmployees: Filter( Employees, Status = 'Status
(Employees)'.Inactive ) },
      ForAll( InactiveEmployees As Employee,
              Patch( Employees, Employee, { Status: 'Status
(Employees)'.Active } ) ) )
```

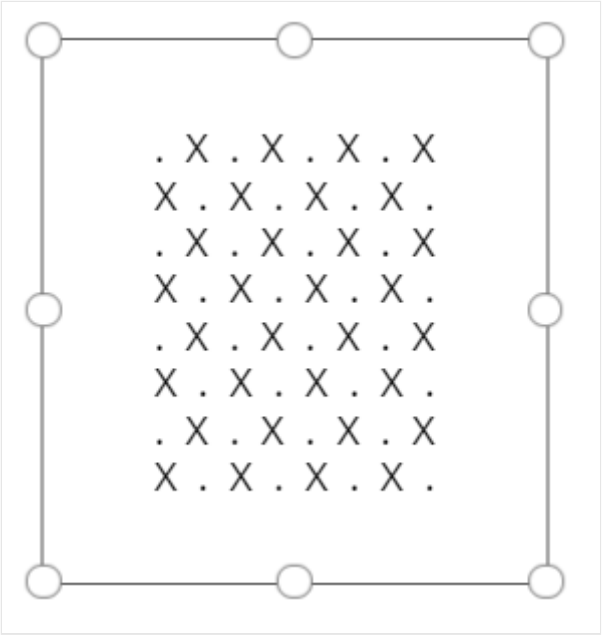
When nesting galleries and record scope functions, **ThisItem** and **ThisRecord** always refers to the inner most scope, leaving records in outer scopes unavailable. Use **As** to make all record scopes available by giving each a unique name.

For example, this formula produces a chessboard pattern as a text string by nesting two **ForAll** functions:

Power Apps

```
Concat(
    ForAll( Sequence(8) As Rank,
        Concat(
            ForAll( Sequence(8) As File,
                    If( Mod(Rank.Value + File.Value, 2) = 1, " X ", " . " )
            ),
            Value
        ) & Char(10)
    ),
    Value
)
```

Setting a **Label** control's **Text** property to this formula displays:



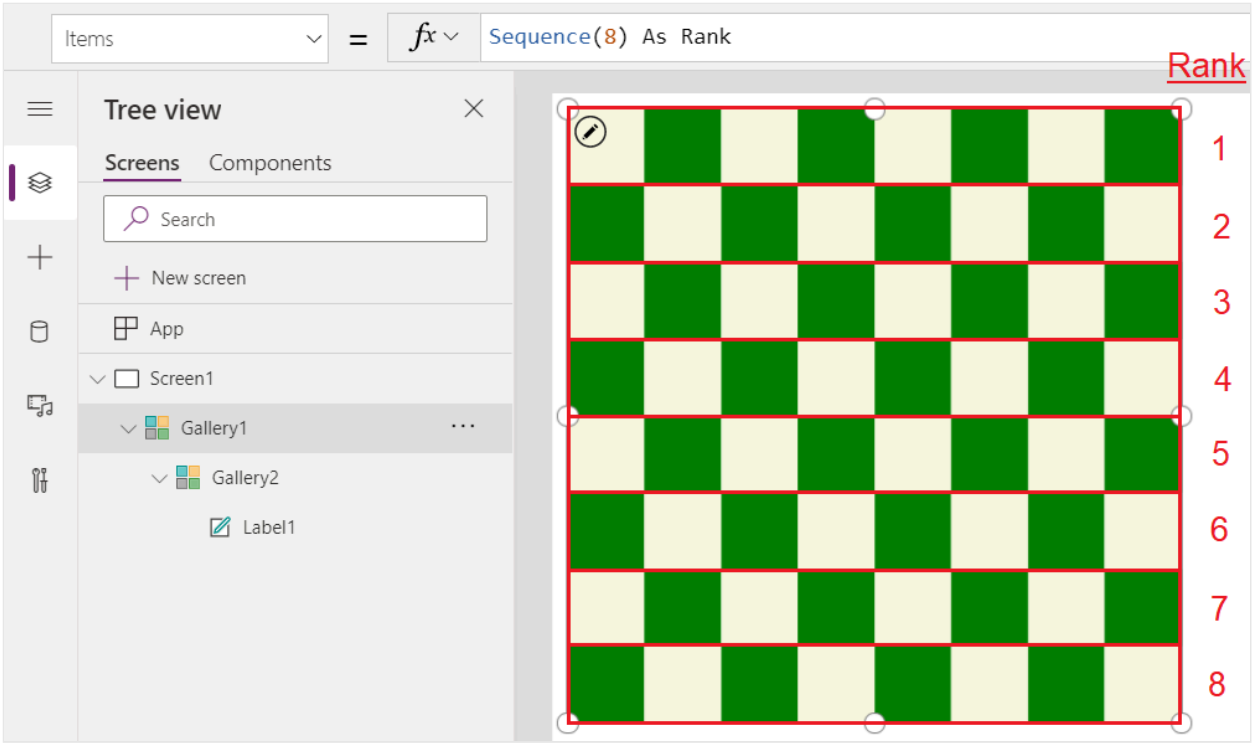
Let's unpack what is happening here:

- We start by iterating an unnamed table of 8 numbered records from the Sequence function. This loop is for each row of the board, which is commonly referred to as **Rank** and so we give it this name.
- For each row, we iterate another unnamed table of 8 columns, and we give the common name **File**.
- If **Rank.Value** + **File.Value** is an odd number, the square gets an **X**, otherwise a dot. This part of the formula is referencing both **ForAll** loops, made possible by using the **As** operator.
- Concat is used twice, first to assemble the columns and then the rows, with a Char(10) thrown in to create a new line.

A similar example is possible with nested **Gallery** controls instead of **ForAll** functions. Let's start with the vertical gallery for the **Rank**. This gallery control will have an **Items** formula of:

Power Apps

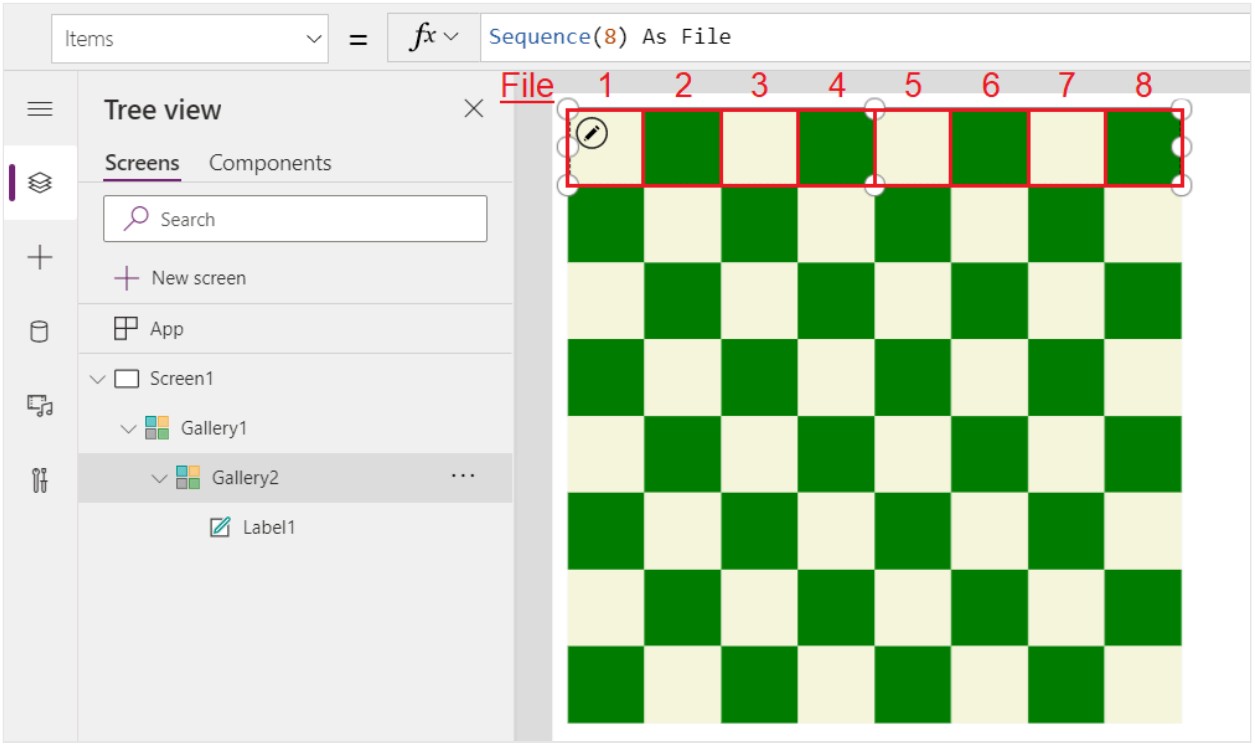
```
Sequence(8) as Rank
```



Within this gallery, we'll place a horizontal gallery for the **File**, that will be replicated for each **Rank**, with an **Items** property of:

Power Apps

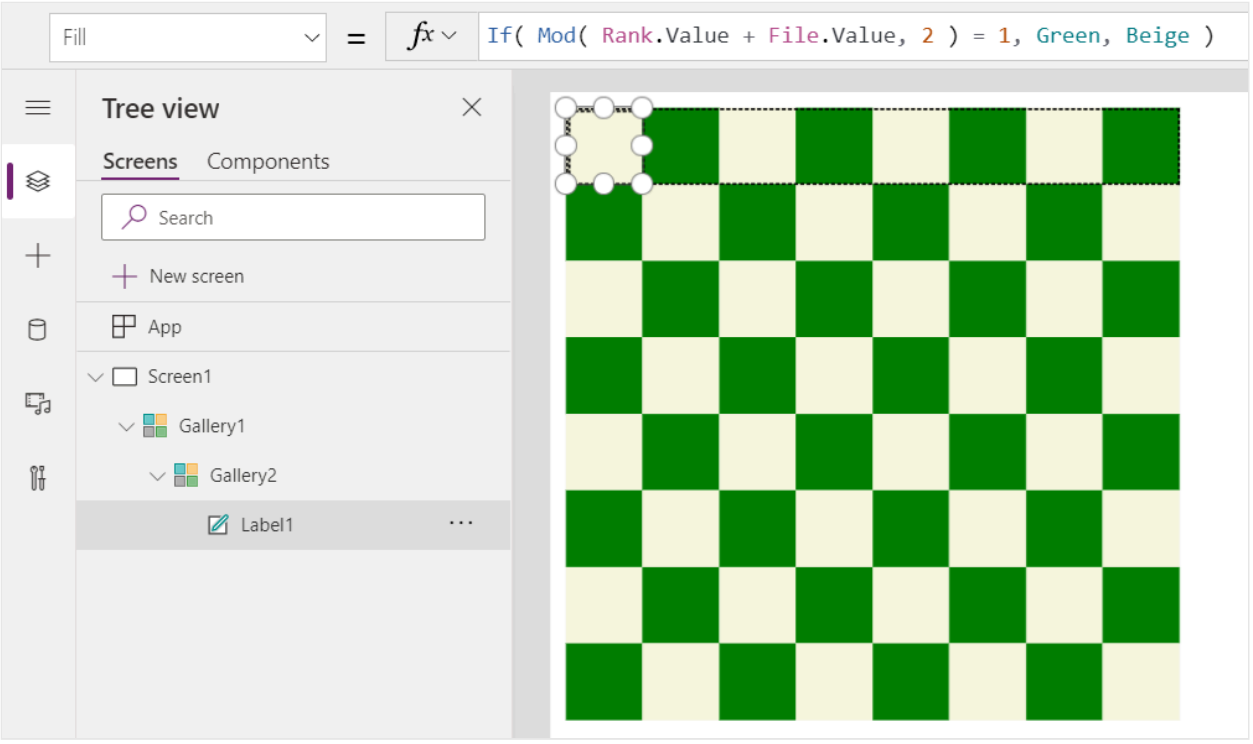
```
Sequence(8) as File
```



And finally, within this gallery, we'll add a **Label** control that will be replicated for each **File** and each **Rank**. We'll size it to fill the entire space and use the **Fill** property to provide the color with this formula:

Power Apps

```
If( Mod( Rank.Value + File.Value, 2 ) = 1, Green, Beige )
```



Self and Parent operators

There are three ways to refer to a control and its properties within a formula:

Expand table

Method	Description
By control name	Any control can be referenced by name from anywhere within the app. For example, **Label1.Fill** refers to the fill property of the control who's name is **Label1**.
Self operator	It's often convenient to reference another property of the same control when writing a formula. Instead of using an absolute reference by name, it's easier and more portable to use a relative reference to one*self*. The **Self** operator provides that easy access to the current control. For example, **Self.Fill** refers to the fill color of the current control.

Method	Description
Parent operator	Some controls host other controls, such as the **Screen** and **Gallery** controls. The hosting control of the controls within it's called the *parent*. Like the **Self** operator, the **Parent** operator provides an easy relative reference to the container control. For example, **Parent.Fill** refers to the fill property of the control that is the container for the current control.

Self and **Parent** are operators and not properties on the controls themselves. Referring to **Parent.Parent**, **Self.Parent** or **Parent.Self** is not supported.

Identifier names

The names of variables, data sources, columns, and other objects can contain any Unicode.

Use single quotes around a name that contains a space or other special character. Use two single quotes together to represent one single quote in the name. Names that don't contain special characters don't require single quotes.

Here are some example column names you might encounter in a table, and how they're represented in a formula:

Expand table

Column name in a database	Column reference in a formula
SimpleName	`SimpleName`
NameWith123Numbers	`NameWith123Numbers`
Name with spaces	`'Name with spaces'`
Name with "double" quotes	`'Name with "double" quotes'`
Name with 'single' quotes	`'Name with ''single'' quotes'`
Name with an @ at sign	`'Name with an @ at sign'`

Double quotes are used to designate text strings.

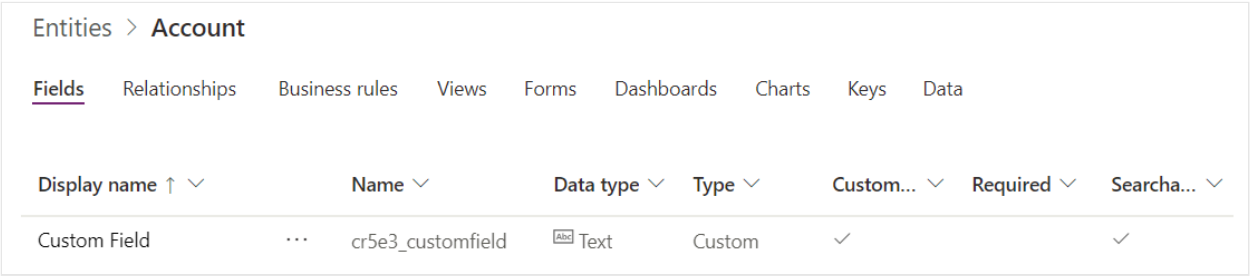
Display names and logical names

Some data sources such as SharePoint and Microsoft Dataverse have two different names to refer to the same table or column of data:

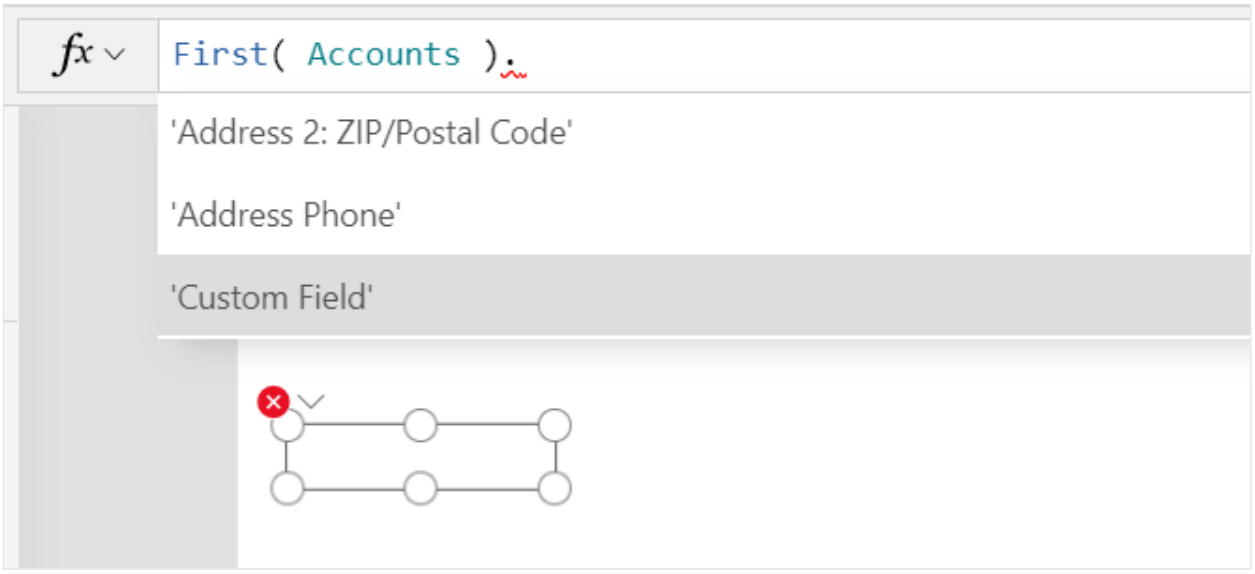
- **Logical name** - A name that is guaranteed to be unique, doesn't change after being created, usually doesn't allow spaces or other special characters, and isn't localized into different languages. As a result, the name can be cryptic. These names are used by professional developers. For example, **cra3a_customfield**. This name may also be referred to as **schema name** or just **name**.
- **Display name** - A name that is user-friendly and intended to be seen by end users. This name may not be unique, may change over time, may contain spaces and any Unicode character, and may be localized into different languages. Corresponding to the example above, the display name may be **Custom Field** with space in between the words.

Since display names are easier to understand, Canvas apps will suggest them as choices and not suggest logical names. Although logical names aren't suggested, they can still be used if typed indirectly.

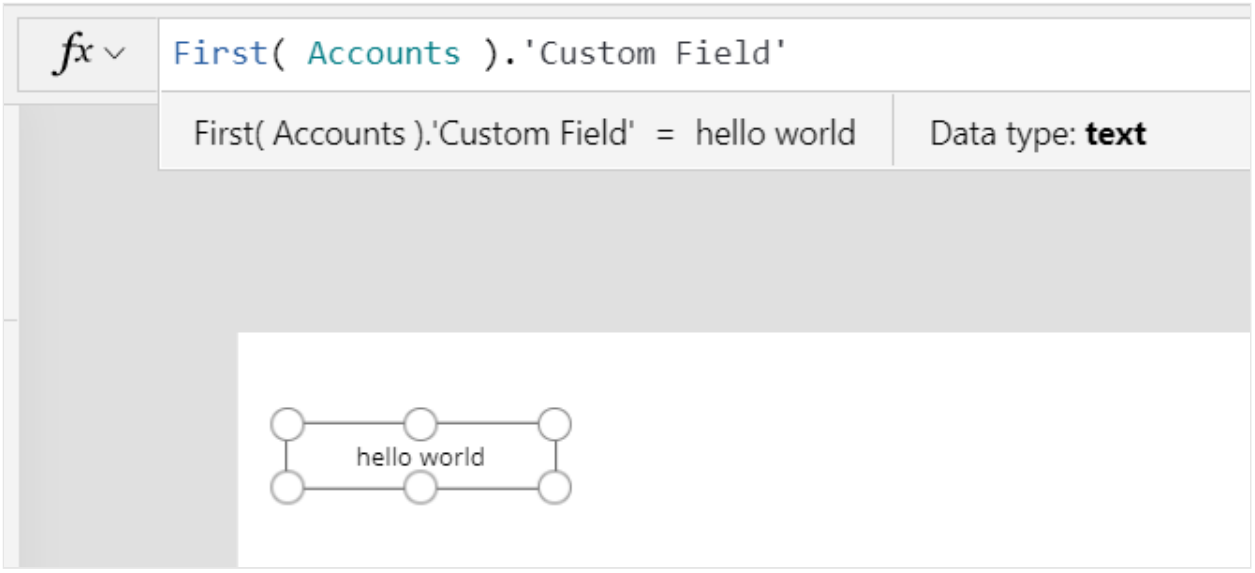
For example, imagine you've added a **Custom Field** to a table in Dataverse. A logical name will be assigned for you by the system, which you can modify only when creating the field. The result would look similar to:



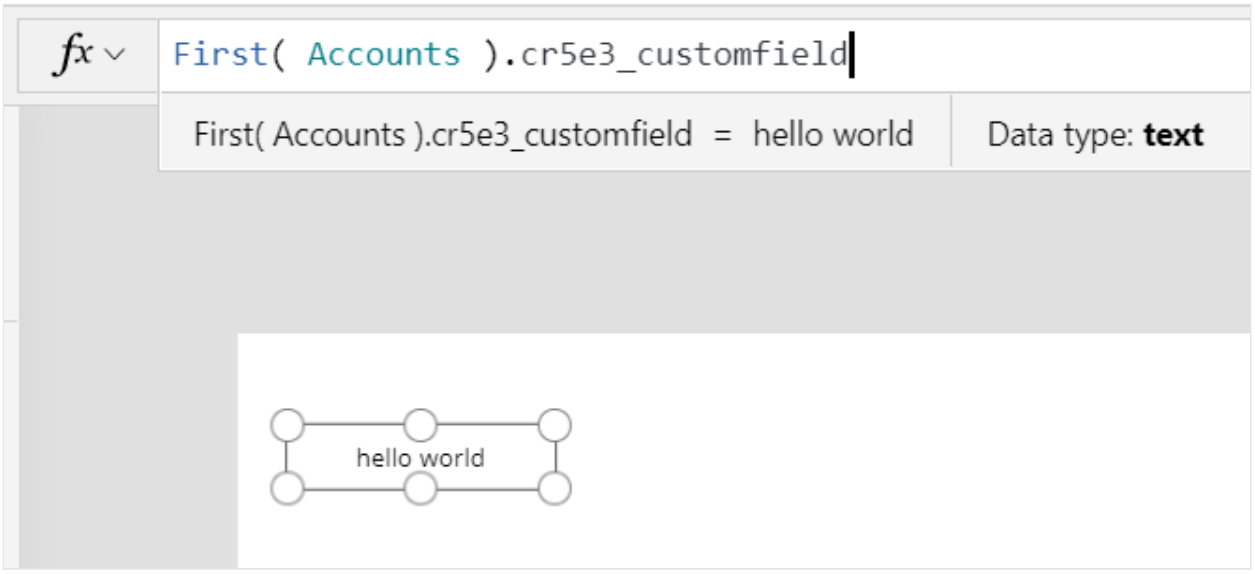
When authoring a reference to a field of Accounts, the suggestion will be made to use '**Custom Field**' since this is the display name. Single quotes must be used because this name has a space in it:



After selecting the suggestion, 'Custom Field' is shown in the formula bar and the data is retrieved:



Although it isn't suggested, we could also use the logical name for this field. This will result in the same data being retrieved. Single quotes are not required since this name doesn't contain spaces or special characters:

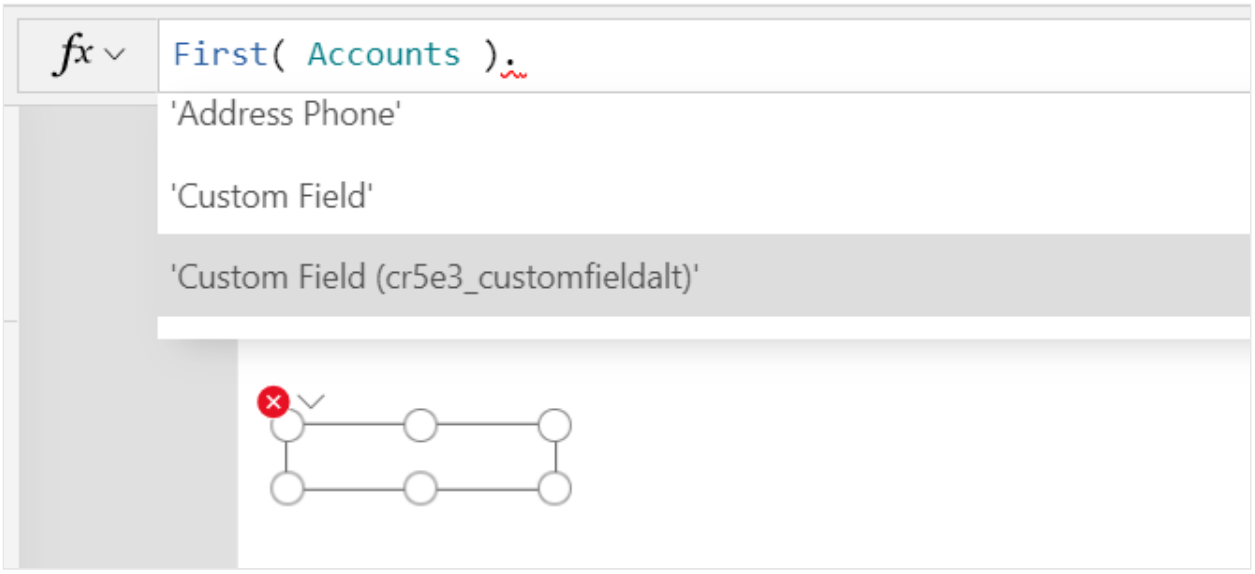


Behind the scenes, a mapping is maintained between the display names seen in formulas and the underlying logical names. Since logical names must be used to interact with the data source, this mapping is used to convert from the current display name to the logical name automatically and that is what is seen in the network traffic. This mapping is also used to convert back to logical names to switch into new display names, for example, if a display name changes or a maker in a different language edits the app.

> **Note**
>
> Logical names are not translated when moving an app between environments. For Dataverse system table and field names, this should not be a problem as logical names are consistent across environments. But any custom fields, such as **cra3a_customfield** in this example above, may have a different environment prefix (**cra3a** in this case). Display names are preferred as they can be matched against display names in the new environment.

Name disambiguation

Since display names aren't unique, the same display name may appear more than once in the same table. When this happens, the logical name will be added to the end of the display name in parentheses for one of more of the conflicting names. Building on the example above, if there was a second field with the same display name of **Custom Field** with a logical name of **cra3a_customfieldalt** then the suggestions would show:



Name disambiguation strings are added in other situations where name conflicts occur, such as the names of table, choices, and other Dataverse items.

Disambiguation operator

Some functions create record scopes for accessing the fields of table while processing each record, such as **Filter**, **AddColumns**, and **Sum**. Field names added with the record scope override the same names from elsewhere in the app. When this happens, you can still access values from outside the record scope with the @ disambiguation operator:

- To access values from nested record scopes, use the @ operator with the name of the table being operated upon using this pattern:
  Table[@*FieldName*]
- To access global values, such as data sources, collections, and context variables, use the pattern [@*ObjectName*] (without a table designation).

For more information and examples, see record scopes.

Feedback

Was this page helpful? Yes No

Provide product feedback

Date, DateTime, and Time functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Converts date and time components to a date/time value.

Description

The **Date** function converts individual Year, Month, and Day values to a Date/Time value. The time portion is midnight.

- If Year is between 0 and 1899 (inclusive), the function adds that value to 1900 to calculate the year. **70** becomes **1970**.
- If Month is less than 1 or more than 12, the result subtracts or adds that many months from the beginning of the specified year.
- If Day is greater than the number of days in the specified month, the function adds that many days to the first day of the month and returns the corresponding date from a subsequent month. If Day is less than 1, the function subtracts that many days, plus 1, from the first day of the specified month.

The **Time** function converts individual Hour, Minute, Second, and optionally Millisecond values to a Date/Time value. The result has no date associated with it.

The **DateTime** function combines the **Date** and **Time** functions into a single function, taking both date and time arguments and returning a Date/Time value that has both date and time components.

See the **DateValue**, **TimeValue**, and **DateTimeValue** functions for information about how to convert a string to a value.

Also see working with dates and times for more information.

Syntax

Date(*Year*, *Month*, *Day*)

- *Year* - Required. Numbers greater than 1899 are interpreted as absolute (1980 is interpreted as 1980); numbers that range from 0 to 1899 are interpreted as relative to 1900. (For example, 80 is interpreted as 1980.)

- *Month* - Required. A number that ranges from 1 to 12.
- *Day* - Required. A number that ranges from 1 to 31.

Time(*Hour*, *Minute*, *Second* [, *Milliseconds*])

- *Hour* - Required. Number of hours, usually in the range 0 (12:00 AM) to 23 (11:00 PM).
- *Minute* - Required. Number of minutes, usually in the range 0 to 59.
- *Second* - Required. Number of seconds, usually in the range 0 to 59.
- *Milliseconds* - Optional. Number of milliseconds, usually in the range 0 to 999.

DateTime(*Year*, *Month*, *Day*, *Hour*, *Minute*, *Second* [, *Milliseconds*])

- *Year* - Required. Numbers greater than 1899 are interpreted as absolute (1980 is interpreted as 1980); numbers that range from 0 to 1899 are interpreted as relative to 1900. (For example, 80 is interpreted as 1980.)
- *Month* - Required. A number that ranges from 1 to 12.
- *Day* - Required. A number that ranges from 1 to 31.
- *Hour* - Required. Number of hours, usually in the range 0 (12:00 AM) to 23 (11:00 PM)
- *Minute* - Required. Number of minutes, usually in the range 0 to 59.
- *Second* - Required. Number of seconds, usually in the range 0 to 59.
- *Milliseconds* - Optional. Number of milliseconds, usually in the range 0 to 999.

Examples

Date

If a user typed

- **1979** in a text-input control named **HireYear**
- **3** in a text-input control named **HireMonth**
- **17** in a text-input control named **HireDay**

this formula would return `3/17/1979`:

Power Apps

```
Date( Value(HireYear.Text), Value(HireMonth.Text), Value(HireDay.Text) )
```

Time

If a user typed

- **14** in a text-input control named **BirthHour**
- **50** in a text-input control named **BirthMinute**
- **24** in a text-input control named **BirthSecond**

this formula would return `02:50:24 P`:

Power Apps

```
Text(
    Time(Value(BirthHour.Text), Value(BirthMinute.Text),
Value(BirthSecond.Text)),
    "hh:mm:ss A/P"
)
```

DateTime

If a user typed

- **2023** in a text-input control named **EclipseYear**
- **10** in a text-input control named **EclipseMonth**
- **28** in a text-input control named **EclipseDate**
- **13** in a text-input control named **EclipseHour**
- **14** in a text-input control named **EclipseMinute**
- **5** in a text-input control named **EclipseSecond**
- **231** in a text-input control named **EclipseMillisecond**

this formula would return `10/28/23 01:14:05.231 PM`:

Power Apps

```
Text(
    DateTime(Value(EclipseYear.Text), Value(EclipseMonth.Text),
Value(EclipseDate.Text),
        Value(EclipseHour.Text), Value(EclipseMinute.Text),
        Value(EclipseSecond.Text), Value(EclipseSecond.Millisecond)
    ),
    "yy/mm/dd hh:mm:ss.000 AM/PM"
)
```

Feedback

Was this page helpful? Yes No

Provide product feedback

DateValue, TimeValue, and DateTimeValue functions

06/19/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts date, time, or both in a *string* to a *date/time* value.

Description

- **DateValue** function converts a *date string* (like "10/01/2014") to a *date/time* value.
- **TimeValue** function converts a *time string* (like "12:15 PM") to a *date/time* value.
- **DateTimeValue** function converts a *date and time string* (like "January 10, 2013 12:13 AM") to a *date/time* value.

DateValue function ignores any time information in the date string, and **TimeValue** function ignores any date information in the time string.

> ⓘ **Note**
>
> The DateValue, TimeValue, and DateTimeValue functions by default use the language from the current user's settings. You can override it to ensure that strings are interpreted properly. For example, "10/1/1920" is interpreted as *October 1st* in "*en*" and as *January 10th* in "*fr*".

Dates must use one of these formats:

- MM/DD/YYYY or MM-DD-YYYY
- DD/MM/YYYY or DD-MM-YYYY
- YYYY/MM/DD or YYYY-MM-DD
- MM/DD/YY or MM-DD-YY
- DD/MM/YY or DD-MM-YY
- DD Mon YYYY
- Month DD, YYYY

To convert from numeric date, month and year components, read Date.
To convert from numeric hour, minute and second components, read Time.

For more information, see:

- Working with date and time
- Date/time and data types

Syntax

DateValue(*String* [, *Language*])
DateTimeValue(*String* [, *Language*])
TimeValue(*String* [, *Language*])

- *String* - Required. A text string that contains a date, time, or combination date and time value.
- *Language* - Optional. A language string, such as would be returned by the first two characters from the Language function. If not provided, the language of the current user's settings is used.

DateValue(*Dynamic*)
DateTimeValue(*Dynamic*)
TimeValue(*Dynamic*)

- *Dynamic* - Required. **Dynamic** value that represents a date or time. Acceptable values are dependent on the untyped provider. For **JSON**, the dynamic value is expected to be a JSON string that contains a date and time in ISO 8601 format. Dates or times in other formats will result in an error. Consider converting such values to **Text** first, then to a date or time. Keep in mind that time zones and locale-related formats are important considerations when communicating with external systems.

Examples

DateValue

If you type **10/11/2014** into a text-input control named **Startdate**, and then set the Text property of a label to these formulas:

- Convert a date from a string in the user's locale and show the result as a long date.

  Power Fx

  ```
  Text( DateValue( Startdate.Text ), DateTimeFormat.LongDate )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014**.

> **Note**
>
> You can use several options with the **DateTimeFormat** enum. To display a list of options, type the parameter followed by a dot or period (.) in the formula bar or check Text function reference.

- Convert date from a string in the French locale and show the result as a long date. In this example, the months and day of the month are interpreted differently from English.

  ```
  Power Fx
  Text( DateValue( Startdate.Text, "fr" ), DateTimeFormat.LongDate )
  ```

  Device set to **en** locale shows the label as **Monday, November 10, 2014**.

If you enter **October 20, 2014** instead:

- Convert a date from a string in the user's locale and calculate the difference between two days, in days

  ```
  Power Fx
  DateDiff( DateValue( Startdate.Text ), Today() )
  ```

  Device set to **en** locale shows the label as **9**, indicating the number of days between October 11 and October 20. The DateDiff function can also show the difference in months, quarters, or years.

DateTimeValue

If you typed **10/11/2014 1:50:24.765 PM** into a text-input control named **Start**, and then set the Text property of a label to the following formula:

- Convert both a date and time string in the current locale.

  ```
  Power Fx
  Text( DateTimeValue( Start.Text ), DateTimeFormat.LongDateTime )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014 1:50:24 PM**.

> **Note**

You can use several options with the **DateTimeFormat** enum. To display a list of options, type the parameter followed by a dot or period (.) in the formula bar or check Text function reference.

- Convert both a date and time string in the French locale. Month and day of the month are interpreted differently.

  Power Fx

  ```
  Text( DateTimeValue( Start.Text, "fr"), DateTimeFormat.LongDateTime )
  ```

  Device set to **en** locale shows the label as **Monday, November 10, 2014 1:50:24 PM**.

- Convert both a date and time string in the user's locale, and display the result with a fractional second.

  Power Fx

  ```
  Text( DateTimeValue( Start.Text ), "dddd, mmmm dd, yyyy hh:mm:ss.fff AM/PM" )
  ```

  Device set to **en** locale shows the label as **Saturday, October 11, 2014 01:50:24.765 PM**.

  As an alternative, you can specify **hh:mm:ss.f** or **hh:mm:ss.ff** to round the time to the nearest 10th or 100th of a second.

TimeValue

Name a text-input control **FinishedAt**, and set the Text property of a label to this formula:

Power Fx

```
If( TimeValue( FinishedAt.Text ) < TimeValue( "5:00:00.000 PM" ),
    "You made it!",
    "Too late!"
)
```

- If you type **4:59:59.999 PM** in the **FinishedAt** control, the label shows "*You made it!*"
- If you type **5:00:00.000 PM** in the **FinishedAt** control, the label shows "*Too late!*"

DateAdd, DateDiff, and TimeZoneOffset functions

Article • 04/20/2025

Expand table

Functions	Applies to
DateAdd **DateDiff**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
TimeZoneOffset	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Expand table

Function	Applies to
DateAdd	
DateDiff	
TimeZoneOffset	

Adds to or finds the difference in date/time values and converts between local time and UTC.

Description

The **DateAdd** function adds a number of units to a date/time value. The result is a new date/time value. You can also subtract a number of units from a date/time value by specifying a negative value.

The **DateDiff** function returns the difference between two date/time values. The result is a whole number of units.

For both functions, units can be **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. By default, both functions use **TimeUnit.Days** as units.

The **TimeZoneOffset** function returns the number of minutes between the user's local time and UTC (Coordinated Universal Time).

You can use **DateAdd** with the **TimeZoneOffset** to convert between the user's local time and UTC (Coordinated Universal Time). Adding **TimeZoneOffset** will convert a local time to UTC, and subtracting it (adding the negative) will convert from UTC to local time.

Also see Date, Time, and DateTime data types and working with dates and times for more information.

Syntax

DateAdd(*DateTime*, *Addition* [, *Units*])

- *DateTime* - Required. Date/time value to operate on.
- *Addition* - Required. Number, in *Units*, to add to the *DateTime*.
- *Units* - Optional. The type of *Units* to add: **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. If not specified, **TimeUnit.Days** are used.

DateDiff(*StartDateTime*, *EndDateTime* [, *Units*])

- *StartDateTime* - Required. Starting date/time value.
- *EndDateTime* - Required. Ending date/time value.
- *Units* - Optional. The type of *Units* to subtract: **TimeUnit.Milliseconds**, **TimeUnit.Seconds**, **TimeUnit.Minutes**, **TimeUnit.Hours**, **TimeUnit.Days**, **TimeUnit.Months**, **TimeUnit.Quarters**, or **TimeUnit.Years**. If not specified, **TimeUnit.Days** are used.

TimeZoneOffset([*DateTime*])

- *DateTime* - Optional. Date/time value for which to return the offset. By default, the current date/time is used.

Examples

In all of these examples, assume that the current date and time is **July 15, 2013, 1:02 PM**.

Simple DateAdd

Expand table

Formula	Description	Result
Text(DateAdd(Now(), 3), "dd-mm-yyyy hh:mm")	Adds three days (default units) to the current date and time.	"18-07-2013 13:02"

Formula	Description	Result
Text(DateAdd(Now(), 4, TimeUnit.Hours), "dd-mm-yyyy hh:mm")	Add four hours to the current date and time.	"15-07-2013 17:02"
Text(DateAdd(Today(), 1, TimeUnit.Months), "dd-mm-yyyy hh:mm")	Adds one month to the current date, without time as **Today** doesn't return a time component.	"15-08-2013 00:00"
Text(DateAdd(Now(), -30, TimeUnit.Minutes), "dd-mm-yyyy hh:mm")	Subtracts 30 minutes from the current date and time.	"15-07-2013 12:32"

Simple DateDiff

Expand table

Formula	Description	Result
DateDiff(Now(), DateValue("1/1/2014"))	Returns the difference between the two units in the default units of **TimeUnit.Days**	170
DateDiff(Now(), DateValue("1/1/2014"), TimeUnit.Months)	Returns the difference between the two values in **TimeUnit.Months**	6
DateDiff(Now(), Today(), TimeUnit.Minutes)	Returns the difference between the current date/time and the current date only (no time) in minutes. Since the **Now** is later than **Today** the result will be negative.	-782

Difference of dates with fractional results

The function DateDiff only returns a whole number of the units being subtracted, and the precision is given in the unit specified. To calculate the difference with a higher precision, use a smaller unit, and convert the result appropriately, like in the examples below.

Expand table

Formula	Description	Result
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Hours)	The minutes/seconds are ignored, the difference is based on the time up to the hour.	1
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Minutes	The minutes are used in the difference, and the result is divided by 60 to have the difference in	0.5

Formula	Description	Result
)/60	hours.	
DateDiff(TimeValue("09:45:00"), TimeValue("10:15:36"), TimeUnit.Seconds)/3600	The minutes and seconds are used in the difference; the result is divided by 3600 to have the difference in hours.	0.51

Converting to UTC

To convert to UTC (Coordinated Universal Time), add the **TimeZoneOffset** for the given time.

For example, imagine the current date and time is **July 15, 2013, 1:02 PM** in Pacific Daylight Time (PDT, UTC-7). To determine the current time in UTC, use:

- **DateAdd(Now(), TimeZoneOffset(), TimeUnit.Minutes)**

TimeZoneOffset defaults to the current time, so you don't need to pass it an argument.

To see the result, use the **Text** function with the format *dd-mm-yyyy hh:mm*, which will return **15-07-2013 20:02**.

Converting from UTC

To convert from UTC, subtract the **TimeZoneOffset** (by adding the negative) for the given time.

For example, imagine the UTC date and time **July 15, 2013, 8:02 PM** is stored in a variable named **StartTime**. To adjust the time for the user's time zone, use:

- **DateAdd(StartTime, -TimeZoneOffset(StartTime), TimeUnit.Minutes)**

Note the negative sign before **TimeZoneOffset** to subtract the offset rather than add it.

To see the result, use the **Text** function with the format *dd-mm-yyyy hh:mm*, which will result in **15-07-2013 13:02** if you're in Pacific Daylight Time.

Now, Today, IsToday, UTCNow, UTCToday, IsUTCToday functions

Article • 04/20/2025

Expand table

Functions	Applies to
Now	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Today **IsToday**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
UTCNow **UTCToday**	✅ Dataverse formula columns ✅ Power Pages

Returns the current date and time, and tests whether a date/time value is today.

> ⓘ Note
>
> PAC CLI pac power-fx commands do not support the **UTCNow**, **UTCToday**, and **IsUTCToday** functions.

Description

The **Now** function returns the current date and time as a date/time value.

The **Today** function returns the current date as a date/time value. The time portion is midnight. **Today** has the same value throughout a day, from midnight today to midnight tomorrow.

The **IsToday** function tests whether a date/time value is between midnight today and midnight tomorrow. This function returns a Boolean (**true** or **false**) value.

Now, **Today**, and **IsToday** functions work with the local time of the current user.

UTCNow, **UTCToday**, and **IsUTCToday** functions are the same as their non-UTC counterparts but work with time zone independent values and use Coordinated Universal Time (UTC).

> ⓘ Note

- **UTCNow**, **UTCToday**, and **IsUTCToday** are only available in Microsoft Dataverse for Teams formula columns, and only for time-independent fields and values.
- **Now**, **Today**, and **IsToday** are not available in Dataverse for Teams formula columns as evaluations are done without the knowledge of the current user's local time zone. More information: Work with formula table columns in Dataverse for Teams

See Date, Time, and DateTime in the data types documentation and working with dates and times for more information.

Volatile Functions

Now, **Today**, **UTCNow**, and **UTCToday** are volatile functions. These functions return a different value for each evaluation.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Now()** will not change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it is part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = DateAdd(Now(), Slider1.Value, Minutes)** then the current time is retrieved each time the Slider control's value changes and the label's text property is reevaluated.

When used in a behavior formula, volatile functions will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Using the user's local time

Now()

Today()

IsToday(*DateTime*)

- *DateTime* - Required. The date/time value to test.

Using Coordinated Universal Time (UTC)

UTCNow()

UTCToday()

IsUTCToday(*TimeZoneIndependentTime*)

- *TimeZoneIndependentDateTime* - Required. The time zone independent date/time value to test.

Examples

For the examples in this section, the current time is **8:58 PM** on **July 11, 2021** in the Pacific Time Zone (UTC-8) and the language is **en-us**.

Expand table

Formula	Description	Result
Text(Now(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time in the user's time zone, and displays it as a string.	"07/11/2021 20:58:00"
Text(Today(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date only, leaving the time portion as midnight, and displays it as a string.	"07/12/2021 00:00:00"
IsToday(Now())	Tests whether the current date and time is between midnight today and midnight tomorrow.	**true**
IsToday(Today())	Tests whether the current date is between midnight today and midnight tomorrow.	**true**
Text(DateAdd(Now(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time, adds 12 days to the result, and displays it as a string.	"07/23/2021 20:58:00"
Text(DateAdd(Today(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date, adds 12 days to the result, and displays it as a string.	"07/23/2021 00:00:00"
IsToday(DateAdd(Now(), 12))	Tests whether the current date and time, plus 12 days, is between midnight today and midnight tomorrow.	**false**
IsToday(DateAdd(Today(), 12))	Tests whether the current date, plus 12 days, is between midnight today and midnight tomorrow.	**false**
Hour(UTCNow())	Retrieves the current date and time in UTC and extracts the hour only, which is 8 hours ahead of local time.	4

Formula	Description	Result
Day(UTCToday())	Retrieves the current date only in UTC and extracts the day, which is 1 day ahead of local time.	12
IsUTCToday(UTCNow())	Tests whether the current date and time is between midnight today and midnight tomorrow, all in UTC time.	**true**
IsUTCToday(UTCToday())	Tests whether the current date is between midnight today and midnight tomorrow, all in UTC time.	**true**

Display a clock that updates in real time

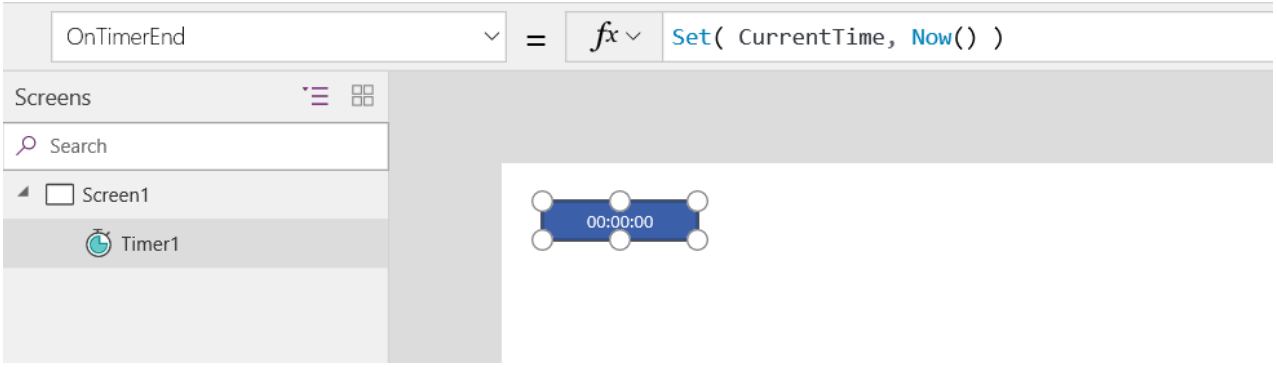
1. Add a Timer control, set its **Duration** property to **1000**, and set its **Repeat** property to **true**.

   The timer will run for one second, automatically start over, and continue that pattern.

2. Set the control's **OnTimerEnd** property to this formula:

   Set(CurrentTime, Now())

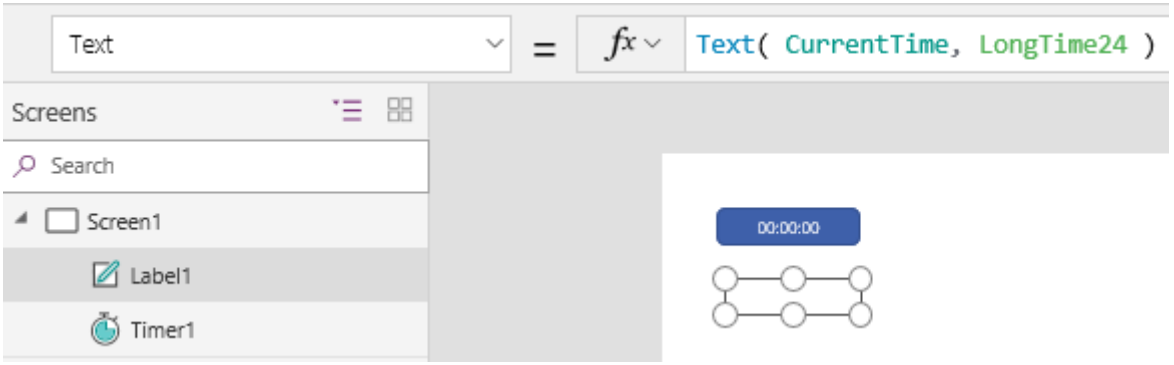
   Whenever the timer starts over (after each second), this formula sets the **CurrentTime** global variable to the current value of the **Now** function.

3. Add a Label control, and set its **Text** property to this formula:

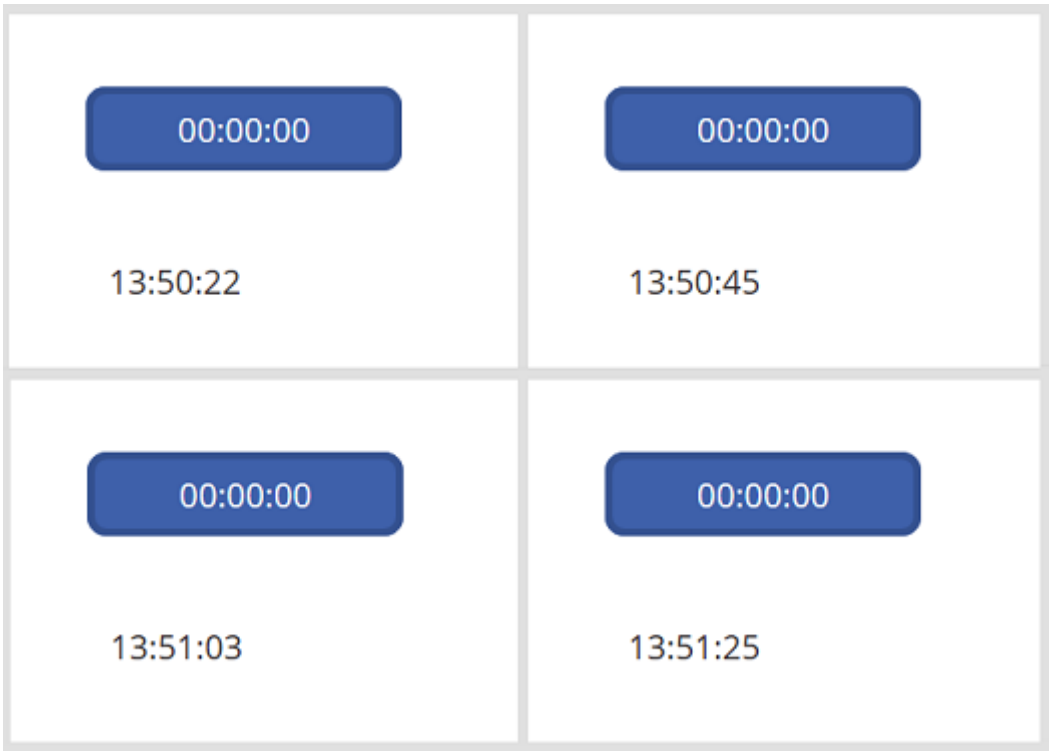
   Text(CurrentTime, LongTime24)

   Use the Text function to format the date and time however you want, or set this property to just **CurrentTime** to show hours and minutes but not seconds.



4. Preview the app by pressing F5, and then start the timer by clicking or tapping it.

   The label continually shows the current time, down to the second.



5. Set the timer's **AutoStart** property to **true** and its **Visible** property to **false**.

   The timer is invisible and starts automatically.

6. Set the screen's OnStart property so that the **CurrentTime** variable has a valid value, as in this example:

   Set(CurrentTime, Now())

   The label appears as soon as the app starts (before the timer runs for a full second).

Trace function

Applies to: ✅ Canvas apps ✅ Copilot Studio ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions

Often there is a lot of logic working behind the scenes in an app. Sometimes the impact is obvious, for example, a control's value or color changes, confirming the logic operated correctly. However, sometimes it isn't obvious and it can be difficult to visualize and understand what is happening inside the app. Use the **Trace** function to record diagnostic information from behind the scenes, creating a timeline of what actually happened, to better understand how your app is operating and to help debug issues.

When used in Power Apps, the output from **Trace** appears in the Power Apps Live Monitor tool along with other app activities. If you allowed your app to send telemetry data to Azure Application Insights, the **Trace** function can also be used to send information to your Application Insights resource. **Trace** can only be used in behavior formulas - to trace values from data properties, you need to use alternatives such as debug buttons.

When used with Test Studio, **Trace** is an optional expression that can be used to provide additional information in your test results from the **OnTestCaseComplete** event. **Trace** event messages are combined with passed and failed assertion messages in the **Traces** table of the **TestCaseResult** record. The **Traces** table has two properties, **Message** and **Timestamp**. Trace information used in tests will also be recorded in Application Insights. Test trace information isn't be available in the Live Monitor tool as the Live Monitor is connected to the app when played from the Power Apps Studio.

Syntax

Trace(*Message* [, *TraceSeverity* [, *CustomRecord* [, *TraceOptions*]]])

- *Message* – Required. The information to be traced. Numbers, Dates, Booleans, and any other data type that can be coerced to Text.
- *TraceSeverity* – Optional. The severity level of the Trace recorded in Live Monitor and Application Insights. Options are **TraceSeverity.Information** (default), **TraceSeverity.Warning**, **TraceSeverity.Error**, or **TraceSeverity.Critical**.
- *CustomRecord* – Optional. A record containing custom data that is recorded in Live Monitor or Application Insights.
- *TraceOptions* – Optional. Options are **TraceOptions.None** (default) and **TraceOptions.IgnoreUnsupportedTypes** which ignores data types in *CustomRecord* that can't be serialized.

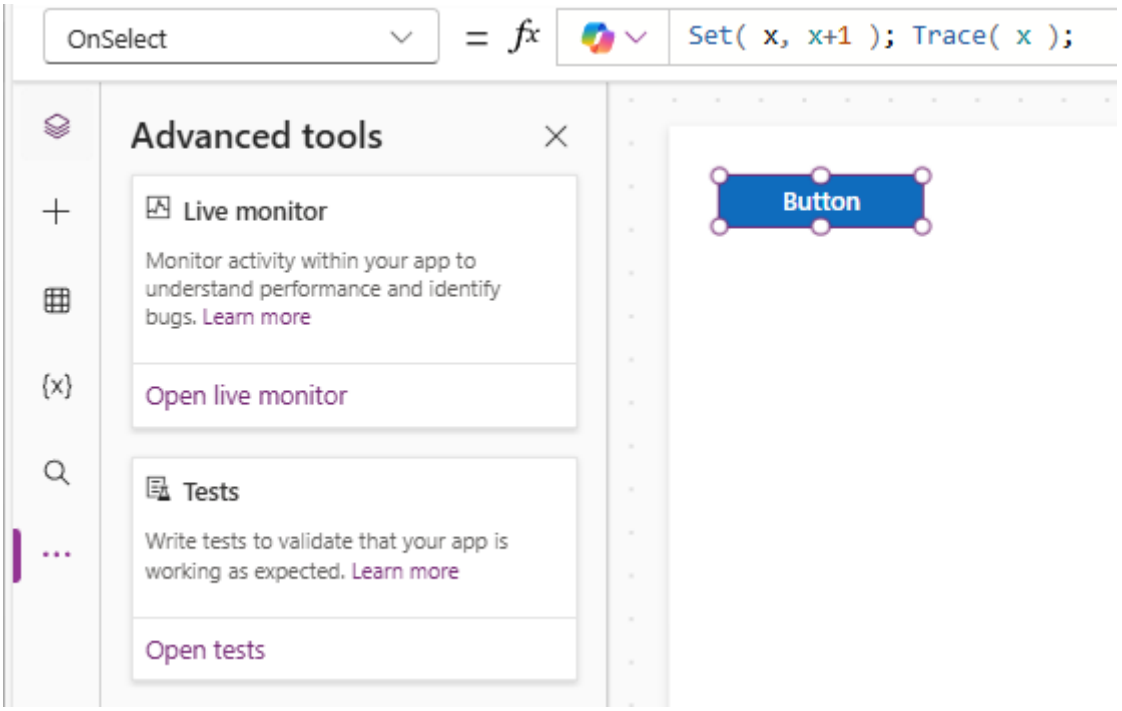
Example

1. Create a button control in Power Apps Studio.
2. Set the **OnSelect** formula to the formula:

Power Fx

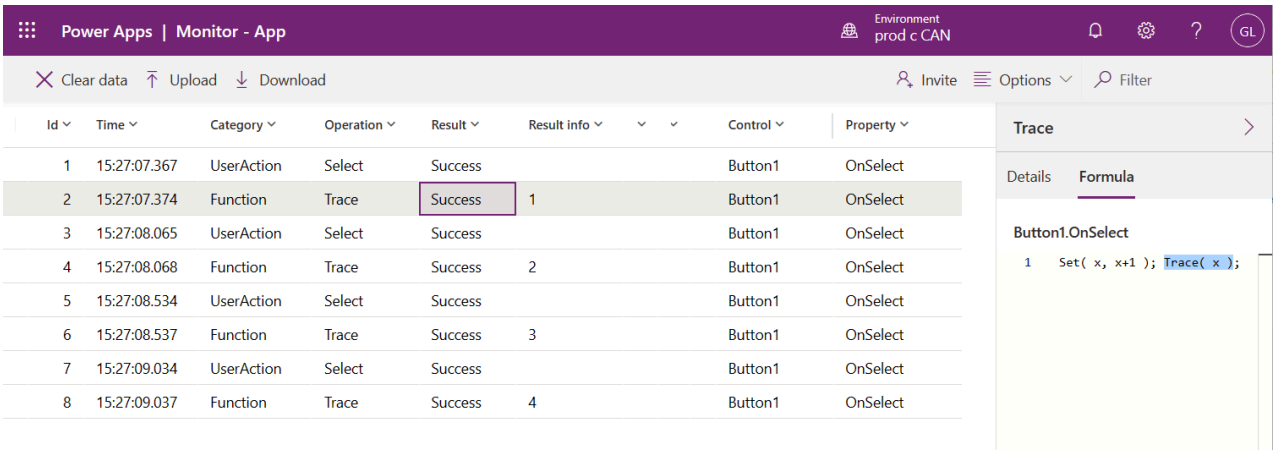
```
Set( x, x+1 );
Trace( x );
```

3. Open the Power Apps Live Monitor in another browser window by selecting the "Advanced tools" icon in the left hand pane and select "Open live monitor":



4. Return to the original Studio browser window and select your button four times. Use **Alt-click** on your mouse if in design mode.
5. View the Power Apps Live Monitor.
6. The Live Monitor's grid will contain an event for each button click and for each **Trace** call, which will show the value of the variable after each increment. Drill into a Trace event to see where the **Trace** was initiated, and the expression used for the message, in the right

hand panel:



See Also

Power Apps Live Monitor Overview
Test Studio Overview
Working with Test Studio

Last updated on 02/03/2026

Trim and TrimEnds functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Removes extra spaces from a string of text.

Description

The **Trim** function removes all spaces from a string of text except for single spaces between words.

The **TrimEnds** function removes all spaces from the start and end of a string of text but leaves spaces between words intact.

If you specify a single string of text, the return value for either function is the string with any extra spaces removed. If you specify a single-column table that contains strings, the return value is a single-column table of trimmed strings. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

By trimming spaces between words, **Trim** is consistent with the function of the same name in Microsoft Excel. However, **TrimEnds** is consistent with programming tools that trim spaces only from the start and end of each string.

Syntax

Trim(*String*)
TrimEnds(*String*)

- *String* - Required. The string of text to remove spaces from.

Trim(*SingleColumnTable*)
TrimEnds(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of strings to remove spaces from.

Example

Expand table

Formula	Description	Result
Trim(" Hello World ")	Removes all spaces from the start and end of a string and extra spaces from within the string.	"Hello World"
TrimEnds(" Hello World ")	Removes all spaces from the start and end of a string.	"Hello World"

The following examples use a single-column collection, named **Spaces**, that contains these strings:

Expand table

Value
" Jane Doe "
" Jack and Jill"
"Already trimmed"
" Venus, Earth, Mars "
"Oil and Water "

To create this collection, set the **OnSelect** property of a **Button** control to this formula, open Preview mode, and then click or tap the button:
ClearCollect(Spaces, [" Jane Doe ", " Jack and Jill", "Already trimmed", " Venus, Earth, Mars ", "Oil and Water "])

Expand table

Formula	Description	Result
Trim(Spaces)	Trims all spaces from the start and end of each string and extra spaces from within each string in the **Spaces** collection.	A single-column table with a `Value` column containing the following values: "Jane Doe", "Jack and Jill", "Already trimmed", "Venus, Earth, Mars", "Oil and Water"
TrimEnds(Spaces)	Trims all spaces from the start and end of each string in the **Spaces** collection.	A single-column table with a `Value` column containing the following values: "Jane Doe", "Jack and Jill", "Already trimmed", "Venus, Earth, Mars", "Oil and Water"

> **Note**
>
> Extra spaces don't appear if you display a collection by clicking or tapping **Collections** on the **File** menu. To verify string length, use the Len function.

Feedback

Was this page helpful? Yes No

Provide product feedback

Trim and TrimEnds functions

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Dataverse formula columns ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Removes extra spaces from a string of text.

Description

The **Trim** function removes all spaces from a string of text except for single spaces between words.

The **TrimEnds** function removes all spaces from the start and end of a string of text but leaves spaces between words intact.

If you specify a single string of text, the return value for either function is the string with any extra spaces removed. If you specify a single-column table that contains strings, the return value is a single-column table of trimmed strings. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

By trimming spaces between words, **Trim** is consistent with the function of the same name in Microsoft Excel. However, **TrimEnds** is consistent with programming tools that trim spaces only from the start and end of each string.

Syntax

Trim(*String*)
TrimEnds(*String*)

- *String* - Required. The string of text to remove spaces from.

Trim(*SingleColumnTable*)
TrimEnds(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of strings to remove spaces from.

Example

Expand table

Formula	Description	Result
Trim(" Hello World ")	Removes all spaces from the start and end of a string and extra spaces from within the string.	"Hello World"
TrimEnds(" Hello World ")	Removes all spaces from the start and end of a string.	"Hello World"

The following examples use a single-column collection, named **Spaces**, that contains these strings:

Expand table

Value
" Jane Doe "
" Jack and Jill"
"Already trimmed"
" Venus, Earth, Mars "
"Oil and Water "

To create this collection, set the **OnSelect** property of a **Button** control to this formula, open Preview mode, and then click or tap the button:
ClearCollect(Spaces, [" Jane Doe ", " Jack and Jill", "Already trimmed", " Venus, Earth, Mars ", "Oil and Water "])

Expand table

Formula	Description	Result
Trim(Spaces)	Trims all spaces from the start and end of each string and extra spaces from within each string in the **Spaces** collection.	A single-column table with a `Value` column containing the following values: "Jane Doe", "Jack and Jill", "Already trimmed", "Venus, Earth, Mars", "Oil and Water"
TrimEnds(Spaces)	Trims all spaces from the start and end of each string in the **Spaces** collection.	A single-column table with a `Value` column containing the following values: "Jane Doe", "Jack and Jill", "Already trimmed", "Venus, Earth, Mars", "Oil and Water"

> **Note**
>
> Extra spaces don't appear if you display a collection by clicking or tapping **Collections** on the **File** menu. To verify string length, use the Len function.

Feedback

Was this page helpful? Yes No

Provide product feedback

Int, Round, RoundDown, RoundUp, and Trunc functions

Article • 04/20/2025

Applies to: ✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Rounds a number.

Round, RoundDown, and RoundUp

The **Round**, **RoundDown**, and **RoundUp** functions round a number to the specified number of decimal places:

- **Round** rounds up if the next digit is 5 or higher. Otherwise, this function rounds down.
- **RoundDown** always rounds down to the previous lower number, towards zero.
- **RoundUp** always rounds up to the next higher number, away from zero.

The number of decimal places can be specified for these functions:

Expand table

Decimal places	Description	Example
Greater than 0	The number is rounded to the right of the decimal separator.	`Round( 12.37, 1 )` returns 12.4.
0	The number is rounded to the nearest integer.	`Round( 12.37, 0 )` returns 12.
Less than 0	The number is rounded to the left of the decimal separator.	`Round( 12.37, -1 )` returns 10.

Int and Trunc

The **Int** and **Trunc** functions round a number to an integer (whole number without a decimal):

- **Int** rounds down to the nearest integer.
- **Trunc** truncates the number to just the integer portion by removing any decimal portion.

The difference between **Int** and **Trunc** is in the handling of negative numbers. For example, for an argument of `-4.3`, **Int** will return the integer further away from zero, `-5`, while **Trunc** will return the integer closer to zero, `-4`. **Int** returns values that are unique amongst the five rounding functions, while **Trunc** returns the same values as **RoundDown**.

Use **Trunc** to extract the decimal portion of a number by subtracting it from the original, for example `X - Trunc(X)`.

Decimal places cannot be specified with **Trunc** as it can with Microsoft Excel. Use **RoundDown** instead when this is needed.

Single-column tables

These functions support single-column tables. If you pass a single number, the return value is the rounded version of that number. If you pass a single-column table that contains numbers, the return value is a single-column table of rounded numbers. The *DecimalPlaces* parameter can be a single value or a single-column table. If the single-column table has less values that the *Number*, zero is used for the remaining values. Use **ShowColumns** and other table shaping functions to extract a single-column table from a larger table.

Syntax

Round(*Number*, *DecimalPlaces*)
RoundDown(*Number*, *DecimalPlaces*)
RoundUp(*Number*, *DecimalPlaces*)

- *Number* - Required. The number to round.
- *DecimalPlaces* - Required. Number of decimal places to round to. Use a positive value to indicate decimal places right of the decimal separator, a negative value to the left, and zero for a whole number.

Int(*Number*)
Trunc(*Number*)

- *Number* - Required. The number to be rounded to an integer.

Examples

Rounding to a whole number.

Expand table

X	Round(X, 0)	RoundUp(X, 0)	RoundDown(X, 0)	Int(X)	Trunc(X)
7.9	8	8	7	7	7
-7.9	-8	-8	-7	-8	-7
7.5	8	8	7	7	7
-7.5	-8	-8	-7	-8	-7
7.1	7	8	7	7	7
-7.1	-7	-8	-7	-8	-7

Rounding to two decimal places to the right of the decimal separator (0.01).

Expand table

X	Round(X, 2)	RoundUp(X, 2)	RoundDown(X, 2)
430.123	430.12	430.13	430.12
430.125	430.13	430.13	430.12
430.128	430.13	430.13	430.12

Rounding to two decimal places to the left of the decimal separator (100).

Expand table

X	Round(X, -2)	RoundUp(X, -2)	RoundDown(X, -2)
430.123	400	500	400
449.942	400	500	400
450.000	500	500	400
450.124	500	500	400
479.128	500	500	400

Rounding a single-column table of values.

Expand table

`X`	`Int( X )`	`Round( X, 2 )`	`RoundDown( X, [ 0, 1, 2 ] )`	`RoundUp( X, [ 2 ] )`
[123.456, 987.593, 542.639]	[123, 987, 542]	[123.46, 987.59, 542.64]	[123, 987.5, 542.63]	[123.46, 988, 543]

GroupBy and Ungroup functions

Article • 06/10/2024

Applies to: Canvas apps Model-driven apps Power Pages

Groups and ungroups records of a table.

Description

The **GroupBy** function returns a table with records grouped together based on the values in one or more columns. Records in the same group are placed into a single record, with a column added that holds a nested table of the remaining columns.

The **Ungroup** function reverses the **GroupBy** process. This function returns a table, breaking into separate records any records that were grouped together.

You can group records by using **GroupBy**, modify the table that it returns, and then ungroup records in the modified table by using **Ungroup**. For example, you can remove a group of records by following this approach:

- Use the **GroupBy** function.
- Use the **Filter** function to remove the entire group of records.
- Use the **Ungroup** function.

You can also aggregate results based on a grouping:

- Use the **GroupBy** function.
- Use the **AddColumns** function with **Sum**, **Average**, and other aggregate functions to add a new column, which is an aggregate of the group tables.
- Use the **DropColumns** function to drop the group table.

Ungroup tries to preserve the original order of the records that were fed to **GroupBy**. This isn't always possible (for example, if the original table contains *blank* records).

A table is a value in Power Apps, just like a string or a number. You can specify a table as an argument for a function, and a function can return a table. **GroupBy** and **Ungroup** don't modify a table; instead they take a table as an argument and return a different table. See working with tables for more details.

> **Note**

In Power Apps prior to version 3.24042, column names were specified with a text string using double quotes, and if connected to a data source they also needed to be logical names. For example, the logical name **"cr43e_name"** with double quotes was used instead of the display name **Name** without quotes. For SharePoint and Excel data sources that contain column names with spaces, each space was specified with **"_x0020_"**, for example **"Column Name"** as **"Column_x0020_Name"**. After this version, all apps were automatically updated to the new syntax described in this article.

Syntax

GroupBy(*Table*, *ColumnName1* [, *ColumnName2*, ...], *GroupColumnName*)

- *Table* - Required. Table to be grouped.
- *ColumnName(s)* - Required. The column names in *Table* by which to group records. These columns become columns in the resulting table.
- *GroupColumnName* - Required. The column name for the storage of record data not in the *ColumnName(s)*.

Ungroup(*Table*, *GroupColumnName*)

- *Table* - Required. Table to be ungrouped.
- *GroupColumnName* - Required. The column that contains the record data setup with the **GroupBy** function.

Examples

Create a collection

1. Add a button, and set its **Text** property so that the button shows **Original**.
2. Set the **OnSelect** property of the **Original** button to this formula:

Power Apps

```
ClearCollect( CityPopulations,
    { City: "London",    Country: "United Kingdom", Population: 8615000},
    { City: "Berlin",    Country: "Germany",        Population: 3562000},
    { City: "Madrid",    Country: "Spain",          Population: 3165000},
    { City: "Rome",      Country: "Italy",          Population: 2874000},
    { City: "Paris",     Country: "France",         Population: 2273000},
    { City: "Hamburg",   Country: "Germany",        Population: 1760000},
    { City: "Barcelona", Country: "Spain",          Population: 1602000},
    { City: "Munich",    Country: "Germany",        Population: 1494000},
```

```
    { City: "Milan",     Country: "Italy",          Population: 1344000}
)
```

3. While holding down the Alt key, select the **Original** button.

   You just created a collection, named **CityPopulations**, that contains this data:

City	Country	Population
"London"	"United Kingdom"	8,615,000
"Berlin"	"Germany"	3,562,000
"Madrid"	"Spain"	3,165,000
"Rome"	"Italy"	2,874,000
"Paris"	"France"	2,273,000
"Hamburg"	"Germany"	1,760,000
"Barcelona"	"Spain"	1,602,000
"Munich"	"Germany"	1,494,000
"Milan"	"Italy"	1,344,000

4. To display this collection, select **Collections** on the **File** menu and then select the **CityPopulations** collection. The first five records in the collection appear:



Group records

1. Add another button, and set its **Text** property to "**Group**".

2. Set the **OnSelect** property of this button to this formula:

ClearCollect(CitiesByCountry, GroupBy(CityPopulations, Country, Cities))

3. While holding down the Alt key, select the **Group** button.

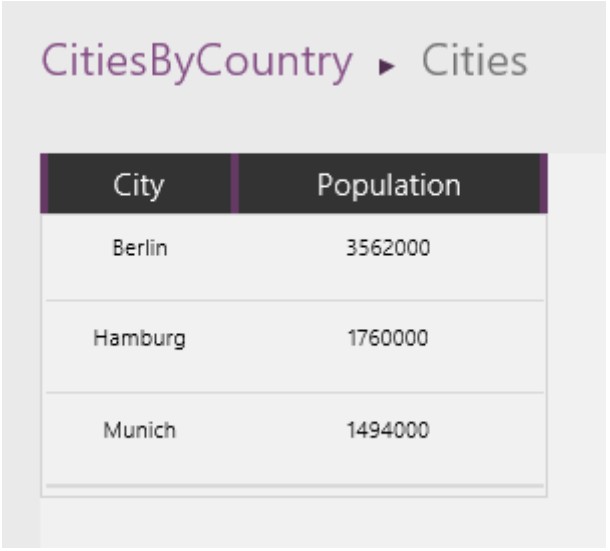
   You just created a collection, named **CitiesByCountry**, in which the records of the previous collection are grouped by the **Country** column.

Cities	Country
City / **Population** "London" / 8,615,000	"United Kingdom"
City / **Population** "Berlin" / 3,562,000 "Hamburg" / 1,760,000 "Munich" / 1,494,000	"Germany"
City / **Population** "Madrid" / 3,165,000 "Barcelona" / 1,602,000	"Spain"
City / **Population** "Rome" / 2,874,000 "Milan" / 1,344,000	"Italy"
City / **Population** "Paris" / 2,273,000	"France"

4. To display the first five records in this collection, select **Collections** on the **File** menu.



5. To display the populations of cities in a country/region, select the table icon in the **Cities** column for that country/region (for example, Germany):



Filter and ungroup records

1. Add another button, and set its **Text** property so that the button shows **"Filter"**.

2. Set the **OnSelect** property of this button to this formula:

   ClearCollect(CitiesByCountryFiltered, Filter(CitiesByCountry, "e" in Country))

3. While holding down the Alt key, select the button that you added.

   You just created a third collection, named **CitiesByCountryFiltered**, that includes only those countries that have an "e" in their names (that is, not Spain or Italy).

Cities	Country
City / Population "London" / 8,615,000	"United Kingdom"
City / Population "Berlin" / 3,562,000 "Hamburg" / 1,760,000 "Munich" / 1,494,000	"Germany"
City / Population "Paris" / 2,273,000	"France"

4. Add one more button, and set its **Text** property so that the button shows **"Ungroup"**.

5. Set the **OnSelect** property of this button to this formula:

   ClearCollect(CityPopulationsUngrouped, Ungroup(CitiesByCountryFiltered, Cities))

   Which results in:

City	Country	Population
"London"	"United Kingdom"	8,615,000
"Berlin"	"Germany"	3,562,000
"Paris"	"France"	2,273,000
"Hamburg"	"Germany"	1,760,000
"Munich"	"Germany"	1,494,000

Aggregate results

Another operation we can perform on a grouped table is to compile the results. In this example, we'll sum the population of the major cities in each country/region.

1. Add another button, and set its **Text** property so that the button shows **"Sum"**.

2. Set the **OnSelect** property of the **"Sum"** button to this formula:

   ClearCollect(CityPopulationsSum, AddColumns(CitiesByCountry, 'Sum of City Populations', Sum(Cities, Population)))

   Which results in:

Cities	Country	Sum of City Populations
City: "London", Population: 8,615,000	"United Kingdom"	8,615,000
City: "Berlin", Population: 3,562,000 City: "Hamburg", Population: 1,760,000 City: "Munich", Population: 1,494,000	"Germany"	6,816,000
City: "Madrid", Population: 3,165,000 City: "Barcelona", Population: 1,602,000	"Spain"	4,767,000
City: "Rome", Population: 2,874,000 City: "Milan", Population: 1,344,000	"Italy"	4,218,000
City: "Paris", Population: 2,273,000	"France"	2,273,000

AddColumns starts with the base **CitiesByCountry** collection and adds a new column **Sum of City Populations**. This column's values are calculated row-by-row, based on the formula **Sum(Cities, Population)**. **AddColumns** provides the value of the **Cities** column (a table) for each row, and Sum adds up the **Population** for each row of this sub table.

Now that we have the sum that we want, we can use DropColumns to remove the sub tables.

3. Add another button, and set its Text property so that the button shows **"SumOnly"**.

4. Set the OnSelect property of the **"SumOnly"** button to this formula:

   ClearCollect(CityPopulationsSumOnly, DropColumns(CityPopulationsSum, Cities))

   Which results in:

Country	Sum of City Populations
"United Kingdom"	8,615,000
"Germany"	6,816,000
"Spain"	4,767,000
"Italy"	4,218,000
"France"	2,273,000

Note that we didn't need to ungroup this table.

Feedback

Was this page helpful? Yes No

Provide product feedback

Relate and Unrelate functions

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

Relate and unrelate records of two tables through a one-to-many or many-to-many relationship.

Description

The **Relate** function links two records through a one-to-many or many-to-many relationship in Microsoft Dataverse. The **Unrelate** function reverses the process and removes the link.

For one-to-many relationships, the Many table has a foreign-key field that points to a record of the One table. **Relate** sets this field to point to a specific record of the One table, while **Unrelate** sets this field to *blank*. If the field is already set when **Relate** is called, the existing link is lost in favor of the new link. You can also set this field by using the **Patch** function or an **Edit form** control; you need not use the **Relate** function.

For many-to-many relationships, the system that links the records maintains a hidden join table. You can't access this join table directly; it can be read only through a one-to-many projection and set through the **Relate** and **Unrelate** functions. Neither related table has a foreign key.

The data for the table that you specify in the first argument will be refreshed to reflect the change, but the data for the table that you specify in the second argument won't. That data must be manually refreshed with the **Refresh** function to show the result of the operation.

These functions never create or delete a record. They only relate or unrelate two records that already exist.

You can use these functions only in behavior formulas.

> **Note**
>
> These functions are part of a preview feature, and their behavior is available only when the **Relational data, option sets, and other new features for CDS** feature is enabled. This is an app-level setting that's enabled by default for new apps. To find this feature switch, select **Settings**, and then select **Upcoming features**. Your

feedback is very valuable to us - please let us know what you think in the Power Apps community forums.

Syntax

Relate(*Table1RelatedTable*, *Table2Record*)

- *Table1RelatedTable* - Required. For a record of *Table1*, the table of *Table2* records related through a one-to-many or many-to-many relationship.
- *Table2Record* - Required. The *Table2* record to add to the relationship.

Unrelate(*Table1RelatedTable*, *Table2Record*)

- *Table1RelatedTable* - Required. For a record of *Table1*, the table of *Table2* records related through a one-to-many or many-to-many relationship.
- *Table2Record* - Required. The *Table2* record to remove from the relationship.

Examples

Consider a **Products** table with the following relationships as seen in the Power Apps portal's table viewer:

Expand table

Relationship display name	Related table	Relationship type
Product Reservation	Reservation	One-to-many
Product ↔ Contact	Contact	Many-to-many

Products and **Reservations** are related through a One-to-Many relationship. To relate the first record of the **Reservations** table with the first record of the **Products** table:

```
Relate( First( Products ).Reservations, First( Reservations ) )
```

To remove the relationship between these records:

```
Unrelate( First( Products ).Reservations, First( Reservations ) )
```

At no time did we create or remove or a record, only the relationship between records was modified.

Products and **Contacts** are related through a Many-to-Many relationship. To relate the first record of the **Contacts** table with the first record of the **Products** table:

```
Relate( First( Products ).Contacts, First( Contacts ) )
```

As Many-to-Many relationships are symmetric, we could also have done this in the opposite direction:

```
Relate( First( Contacts ).Products, First( Products ) )
```

To remove the relationship between these records:

```
Unrelate( First( Products ).Contacts, First( Contacts ) )
```

or:

```
Unrelate( First( Contacts ).Products, First( Products ) )
```

The walk through that follows does exactly these operations on these tables using an app with **Gallery** and **Combo box** controls for selecting the records involved.

These examples depend on the sample data being installed in your environment. Either create a trial environment including sample data or add sample data to an existing environment.

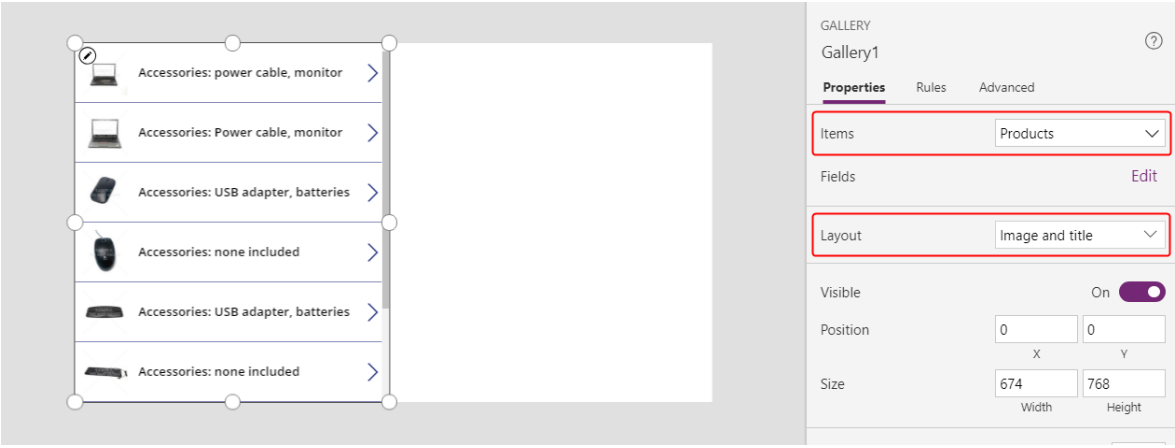
One-to-many

Relate function

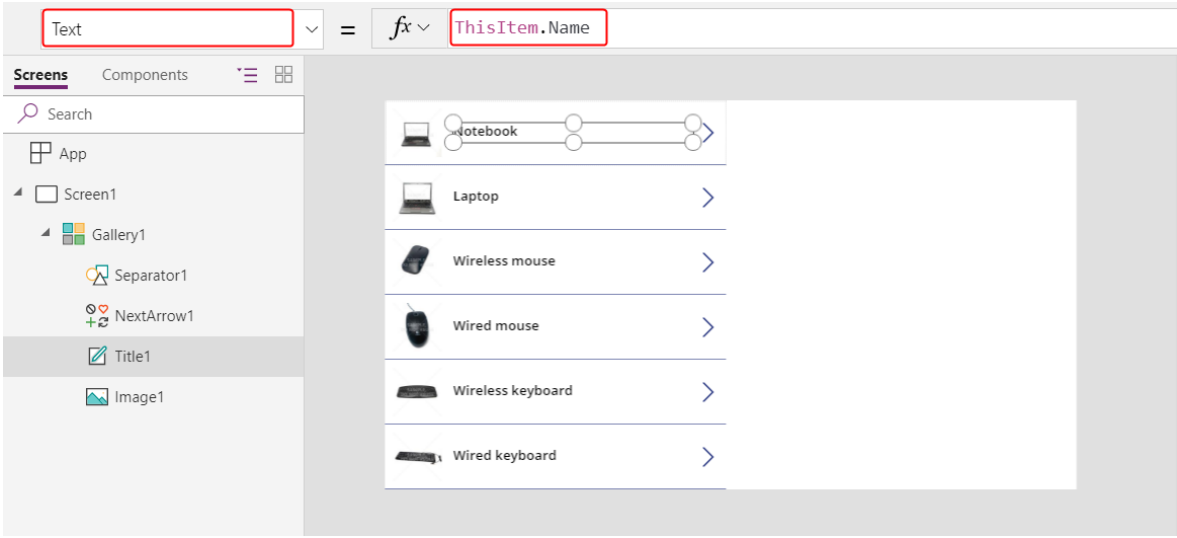
You'll first create a simple app to view and reassign the reservations that are associated with a product.

1. Create a tablet app from blank.
2. On the **View** tab, select **Data sources**.
3. In the **Data** pane, select **Add data** > select **Products**.
   The Products table is part of the sample data loaded above.
4. On the **Insert** tab, add a blank vertical **Gallery** control.
5. Ensure that the control that you just added is named **Gallery1**, and then move and resize it to fill the left-hand side of the screen.

6. On the **Properties** tab, set **Gallery1**'s **Items** property to **Products** and its **Layout** to **Image and title**.



7. In **Gallery1**, ensure that the **Label** control is named **Title1**, and then set its **Text** property to **ThisItem.Name**.



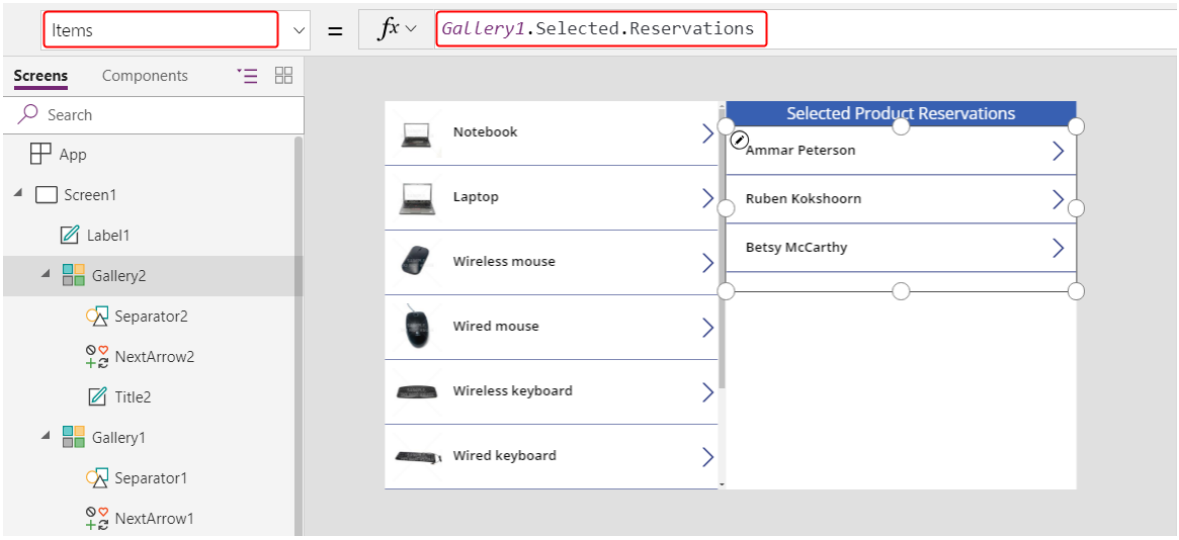
8. Select the screen to avoid inserting the next item into **Gallery1**. Add a second blank vertical **Gallery** control, and ensure that it's named **Gallery2**.

   Gallery2 will show the reservations for whatever product the user selects in **Gallery1**.

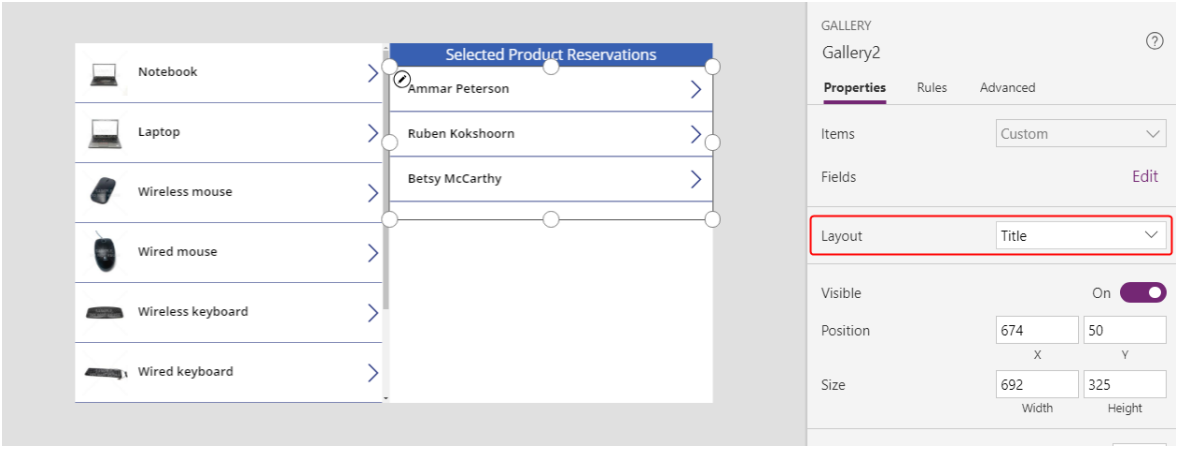
9. Move and resize **Gallery2** to fill the upper-right quadrant of the screen.

10. (optional) Add the blue **Label** control above **Gallery2**, as the next graphic shows.

11. In the formula bar, set the **Items** property of **Gallery2** to **Gallery1.Selected.Reservations**.



12. In the properties pane, set **Gallery2**'s **Layout** to **Title**.

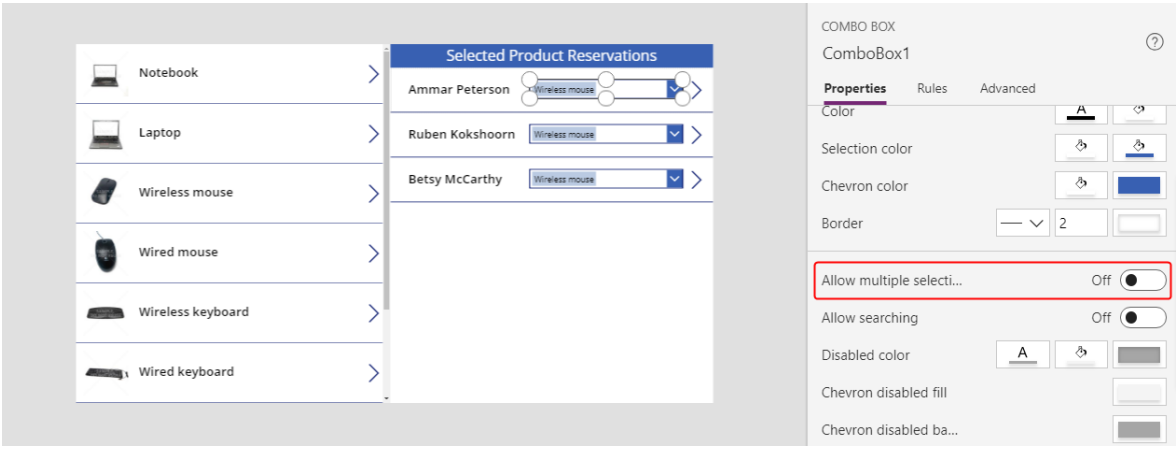


13. In **Gallery2**, add a Combo box control, ensure that it's named **ComboBox1**, and then move and resize it to avoid blocking the other controls in **Gallery2**.

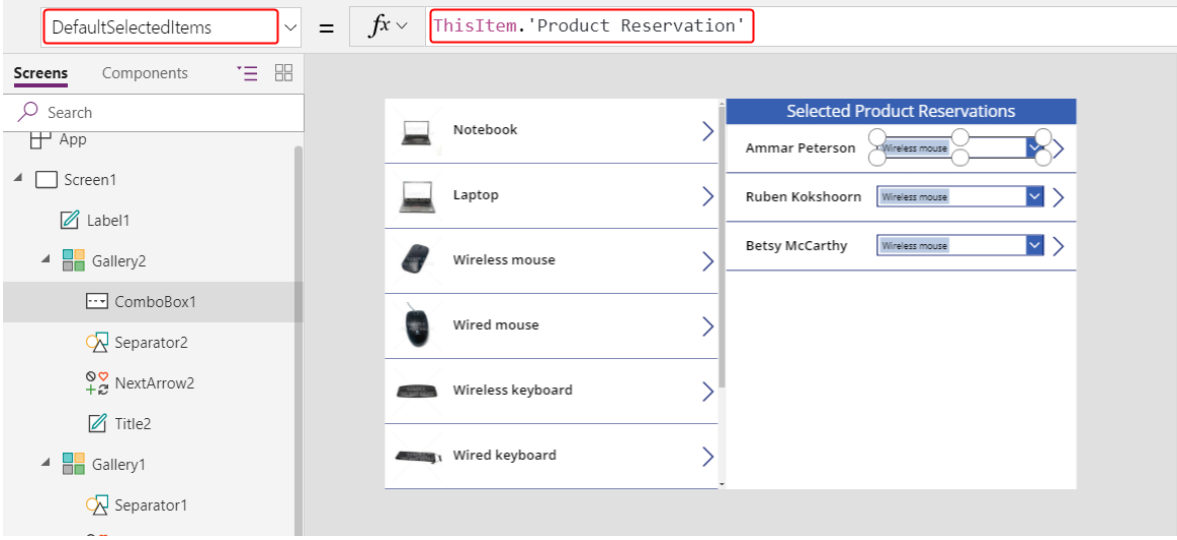
14. On the **Properties** tab, set **ComboBox1**'s **Items** property to **Products**.



15. Scroll down in the **Properties** tab and set **ComboBox1**'s **Allow multiple selection** property to **Off**.



16. In the formula bar, set **ComboBox1**'s **DefaultSelectedItems** property to **ThisItem.'Product Reservation'**.

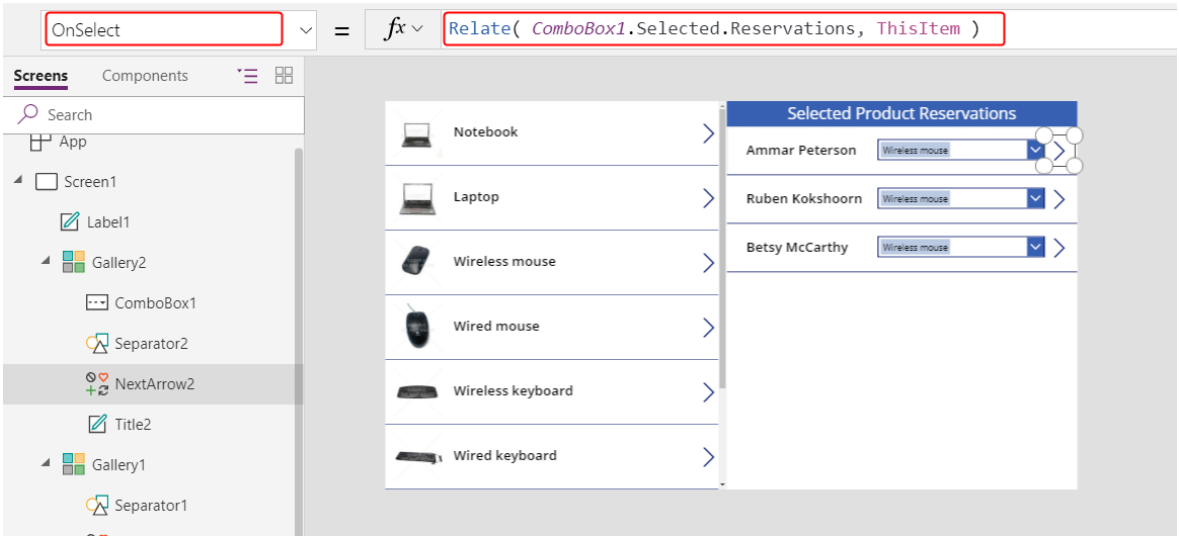


17. In **Gallery2**, set **NextArrow2**'s **OnSelect** property to this formula:

Power Apps

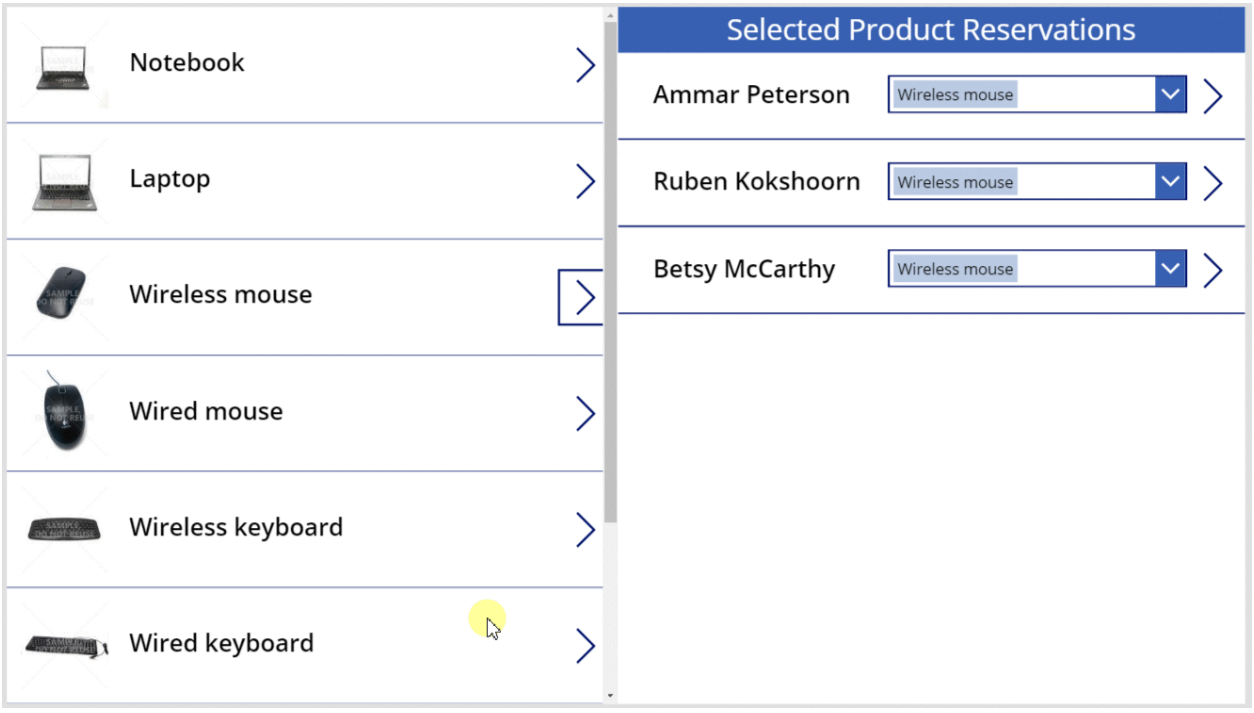
```
Relate( ComboBox1.Selected.Reservations, ThisItem )
```

When the user selects this icon, the current reservation changes to the product that the user selected in **ComboBox1**.



18. Press F5 to test the app in Preview mode.

With this app, the user can move a reservation from one product to another. For a reservation on one product, the user can select a different product in **ComboBox1** and then select **NextArrow2** to change that reservation.



Unrelate function

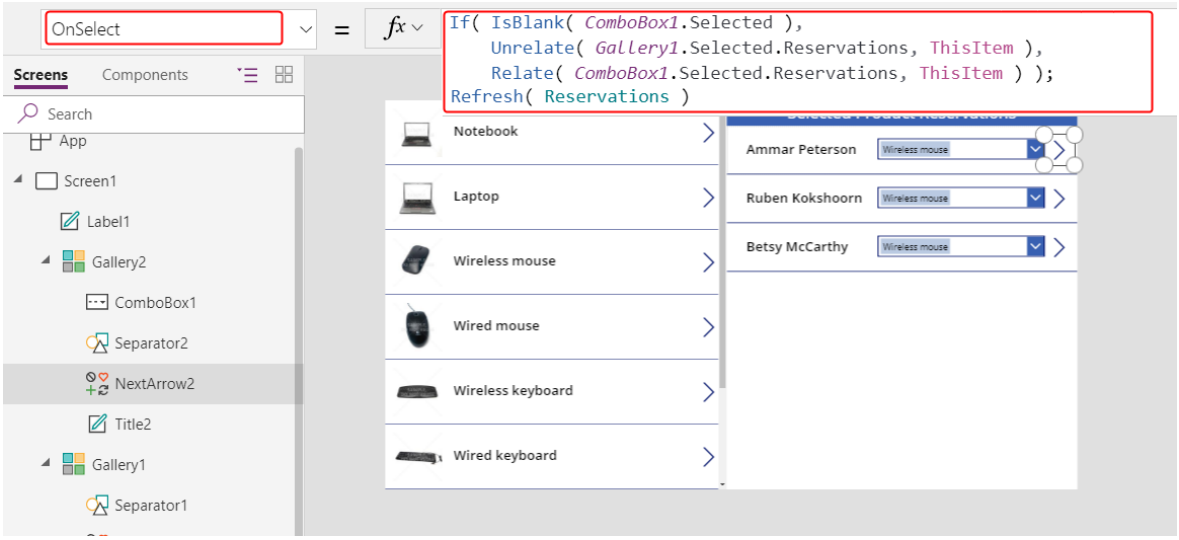
At this point, you can move the relationship from one record to another, but you can't remove the relationship altogether. You can use the **Unrelate** function to disconnect a reservation record from any product.

1. On the **View** tab, select **Data sources**.

2. In the **Data** pane, select **Add data source** > **Microsoft Dataverse** > **Reservations** > **Connect**.

3. In **Gallery2**, set the **OnSelect** formula for **NextArrow2** to this formula:

```
Power Apps

If( IsBlank( ComboBox1.Selected ),
    Unrelate( Gallery1.Selected.Reservations, ThisItem ),
    Relate( ComboBox1.Selected.Reservations, ThisItem )
);
Refresh( Reservations )
```



4. Copy **Gallery2** to the Clipboard by selecting it and then pressing Ctrl-C.

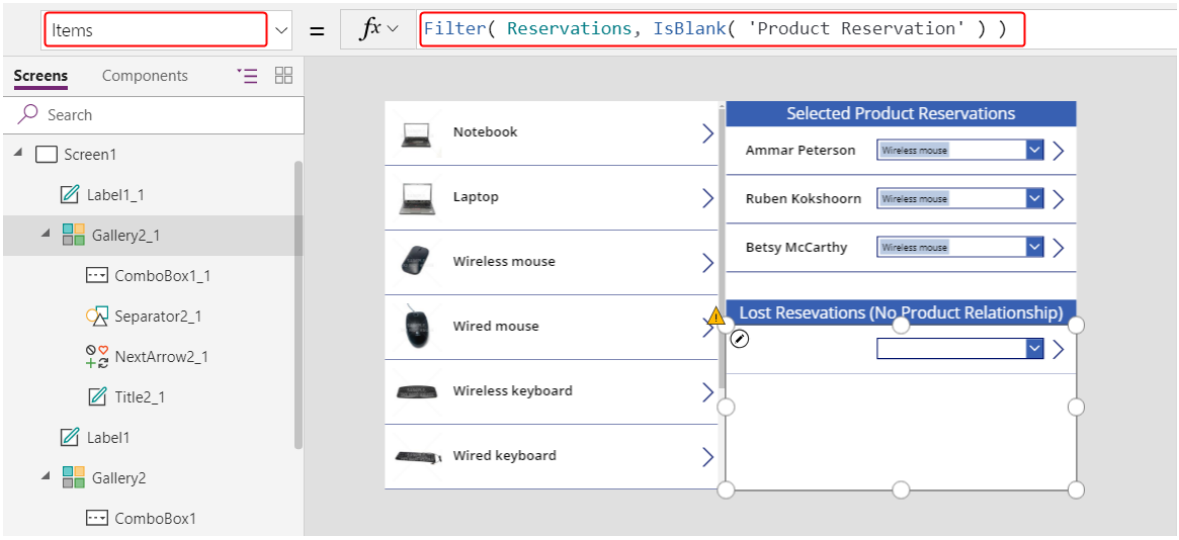
5. Paste a duplicate of **Gallery2** to the same screen by pressing Ctrl-V, and then move it to the lower-right quadrant of the screen.

6. (optional) If you added a label above **Gallery2**, repeat the previous two steps for that label.

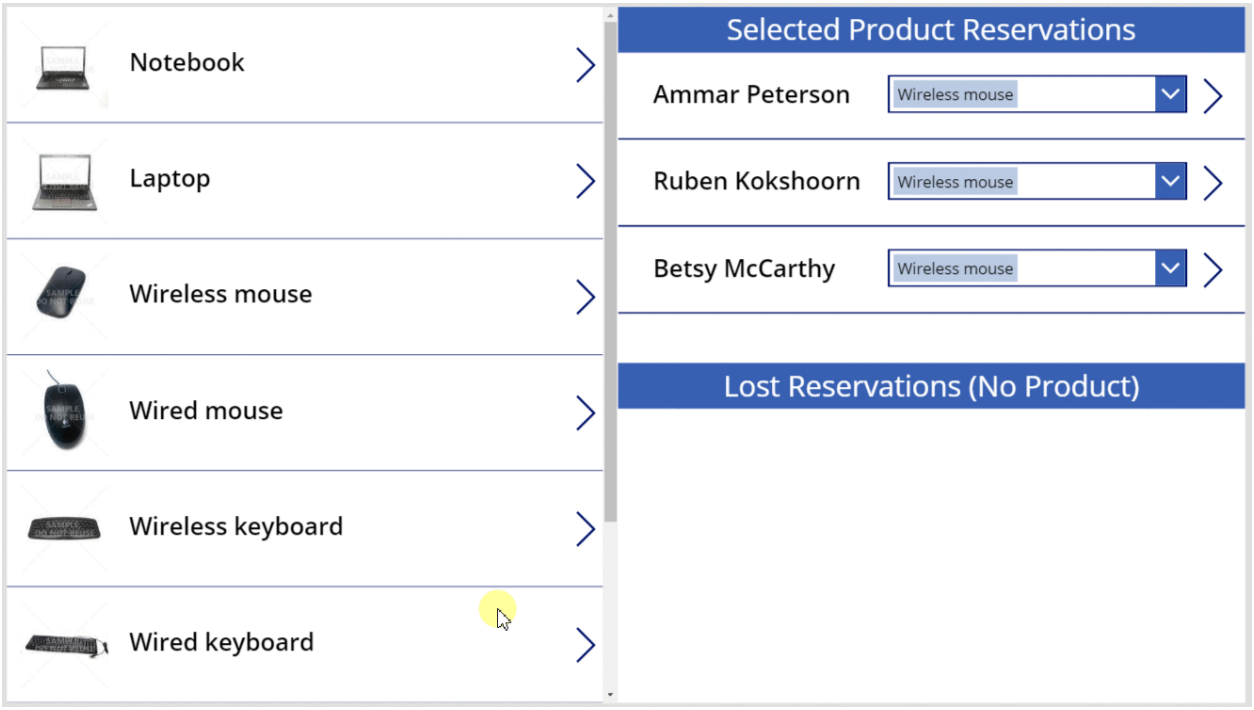
7. Ensure that the duplicate of **Gallery2** is named **Gallery2_1**, and then set its **Items** property to this formula:

```
Power Apps

Filter( Reservations, IsBlank( 'Product Reservation' ) )
```

   A delegation warning appears, but it won't matter with the small amount of data in this example.



With these changes, users can clear the selection in **ComboBox1** for a contact if that person hasn't reserved a product. Contacts who haven't reserved a product appear in **Gallery2_1** where users can assign each contact to a product.

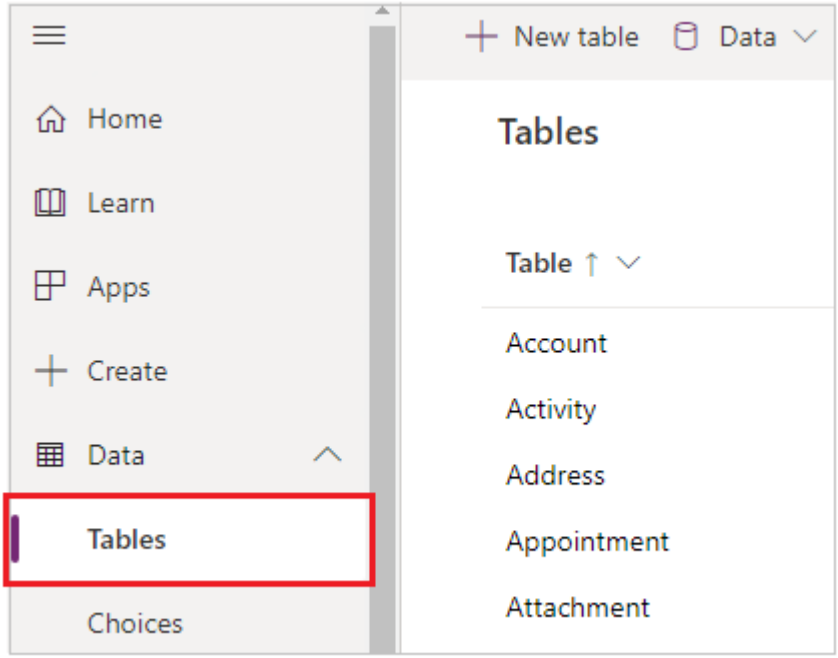


Many-to-many

Create a many-to-many relationship

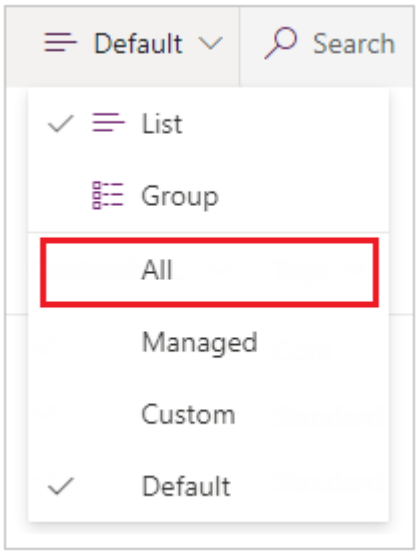
The sample data doesn't include a many-to-many relationship, but you'll create one between the Products table and the Contacts table. Users can relate each product to more than one contact and each contact to more than one product.

1. From this page, select **Data** in the left navigation bar, and then select **Tables**.



2. Change the table filter to include all tables.

   By default, sample tables don't appear.



3. Scroll down, open the **Product** table, and select **Relationships**.

4. Select **Add relationship** > **Many-to-many**.

5. Select the **Contact** table for the relationship.

6. Select **Done** > **Save table**.

Relate and unrelate contacts with one or more products

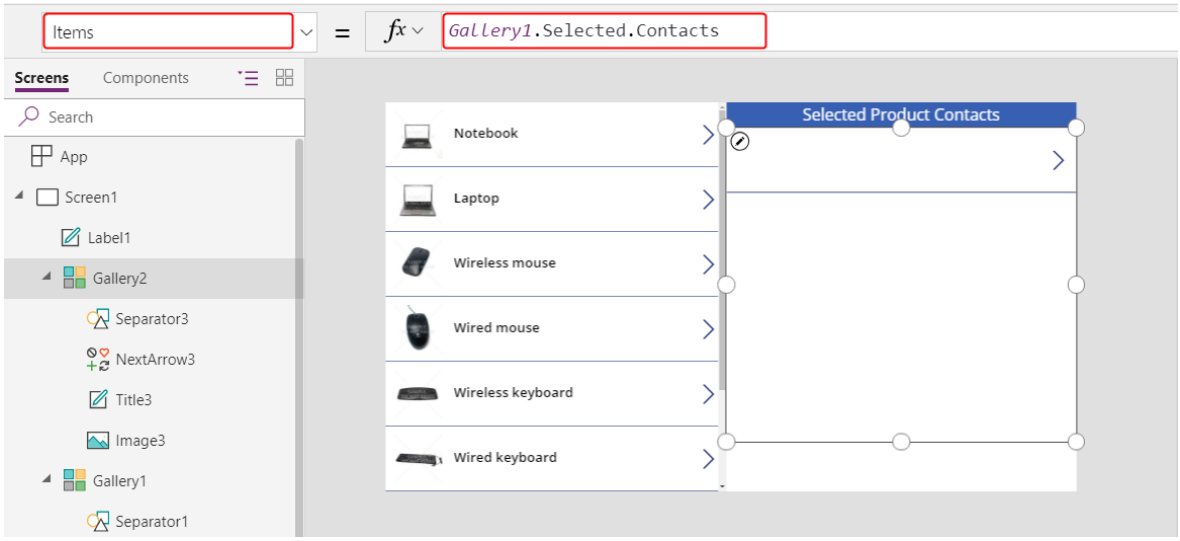
You'll create another app that resembles the one you created earlier in this topic, but the new app will offer a many-to-many relationship. Each contact will be able to reserve multiple products instead of only one.

1. In a blank app for tablets, create **Gallery1** as the first procedure in this topic describes.

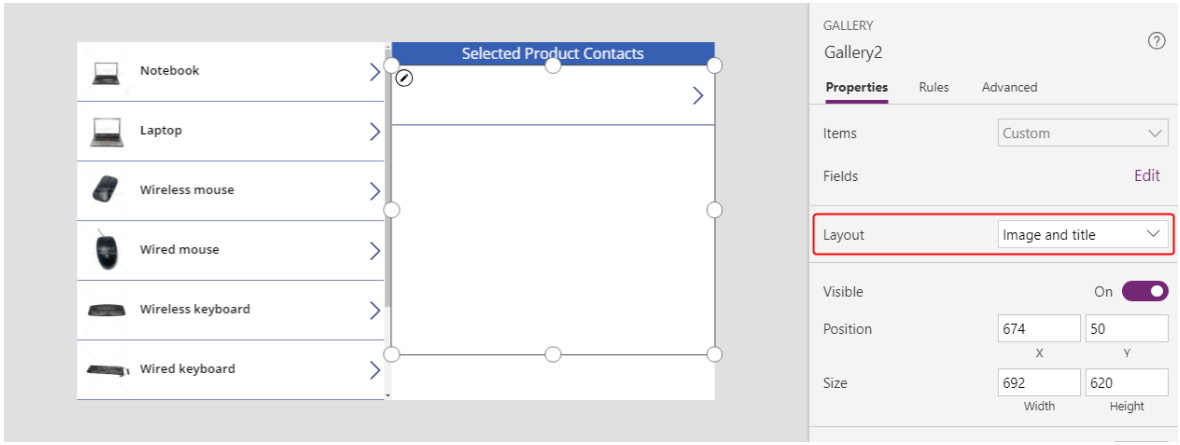
2. Add another blank vertical **Gallery** control, ensure that it's named **Gallery2**, and then move it into the upper-right corner of the screen.

   Later in this topic, you'll add a **Combo box** control under **Gallery2**.

3. In the formula bar, set **Gallery2**'s **Items** property to **Gallery1.Selected.Contacts**.

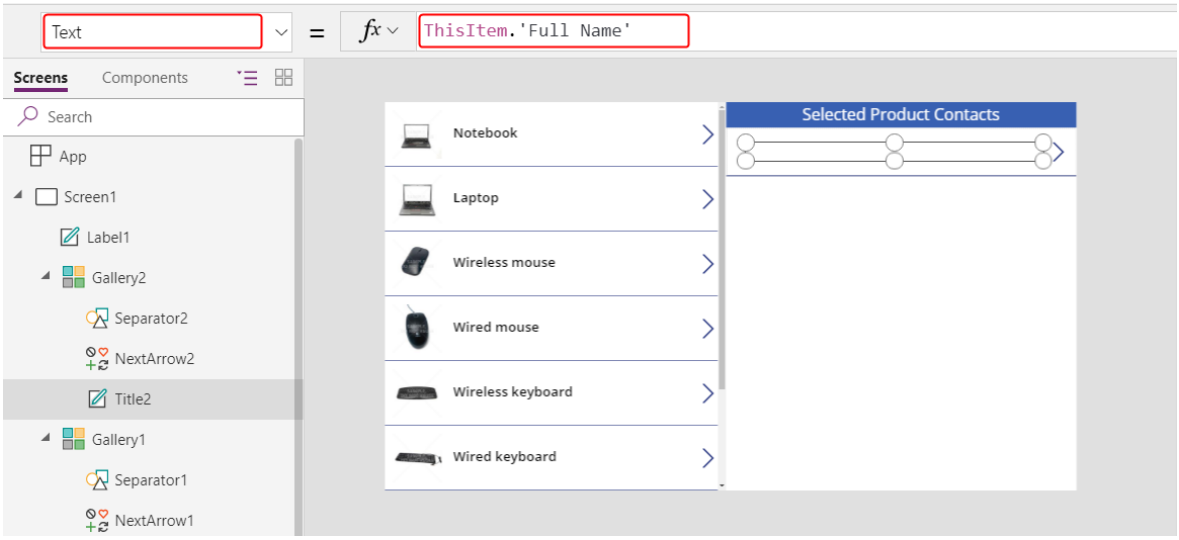


4. On the **Properties** tab, set **Layout** to **Image and title**.



5. In **Gallery2**, ensure that the **Label** control is named **Title2**, and then set its **Text** property to **ThisItem.'Full Name'**.

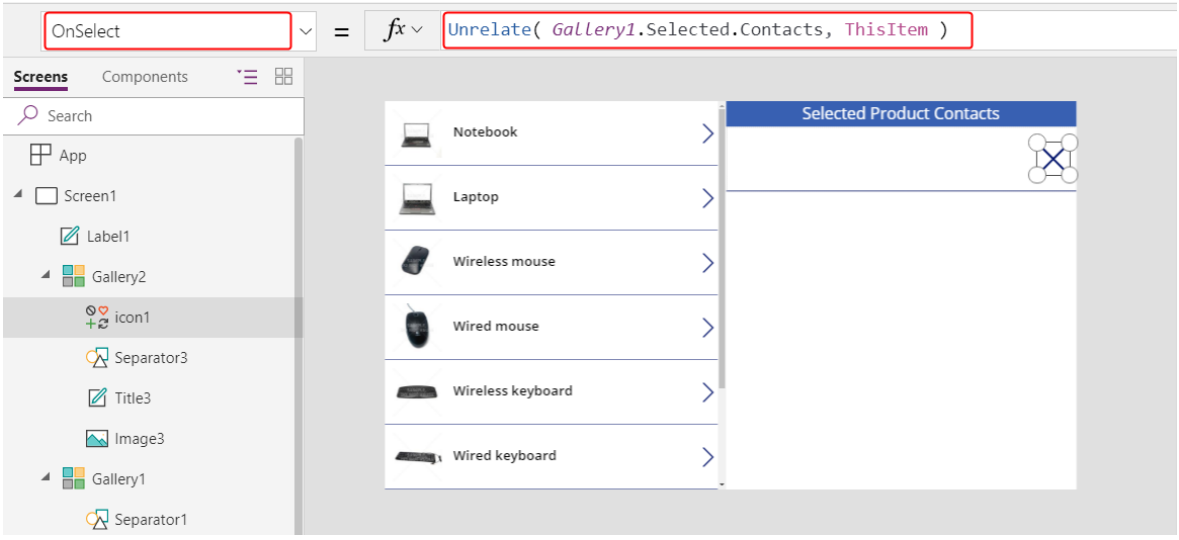
   No text will appear in that control until you finish this procedure and assign a contact to a product.



6. Delete **NextArrow2**, insert a **Cancel** icon, and ensure that it's named **icon1**.

7. Set the **Cancel** icon's **OnSelect** property to this formula:

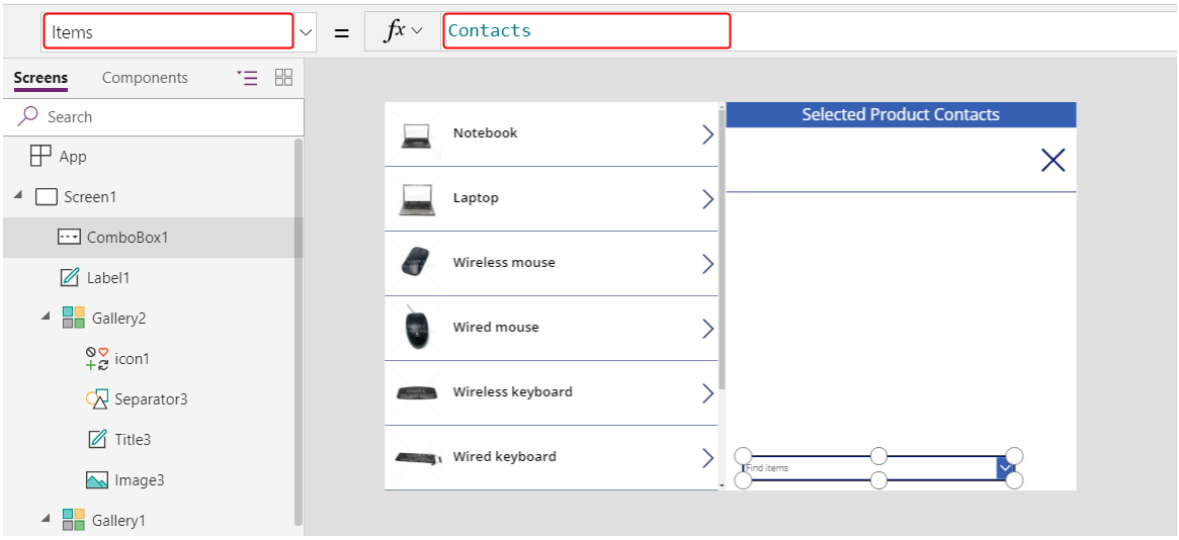
```
Power Apps

Unrelate( Gallery1.Selected.Contacts, ThisItem )
```



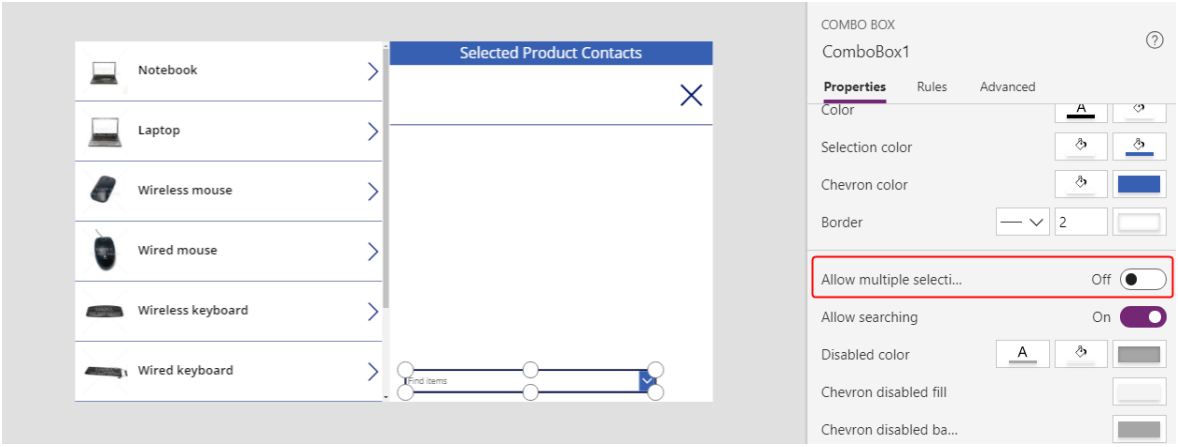
8. On the **View** tab, select **Data sources**.

9. In the **Data** pane, select **Add data source** > **Microsoft Dataverse** > **Contacts** > **Connect**.

10. Under **Gallery2**, add a **Combo box** control, ensure that it's named **ComboBox1**, and then set its **Items** property to **Contacts**.



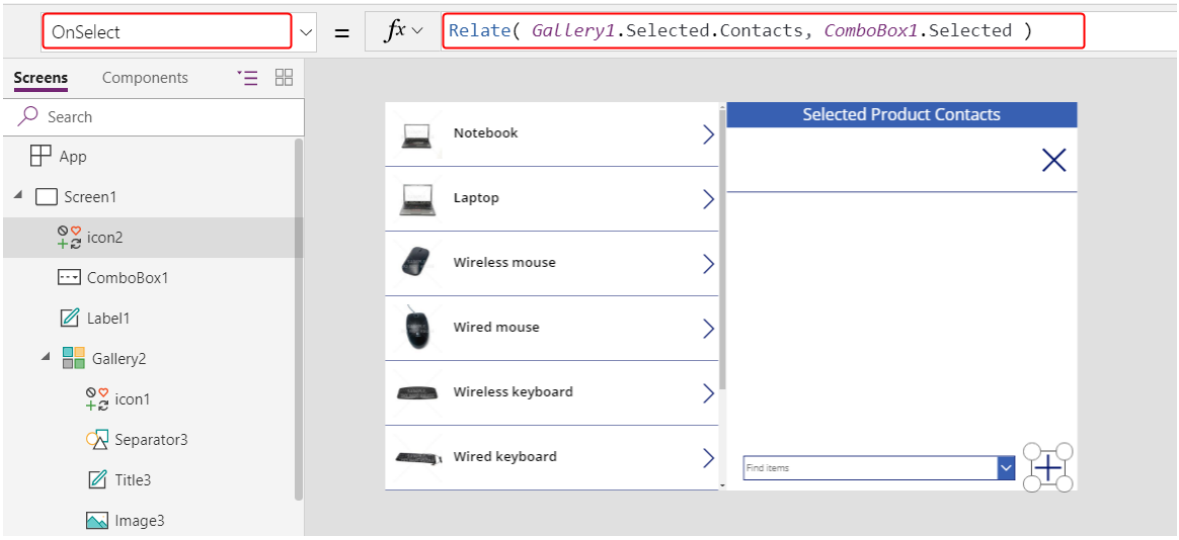
11. On the **Properties** tab, set **Allow multiple selection** to **Off**.



12. Insert an **Add** icon, and set its **OnSelect** property to this formula:

Power Apps

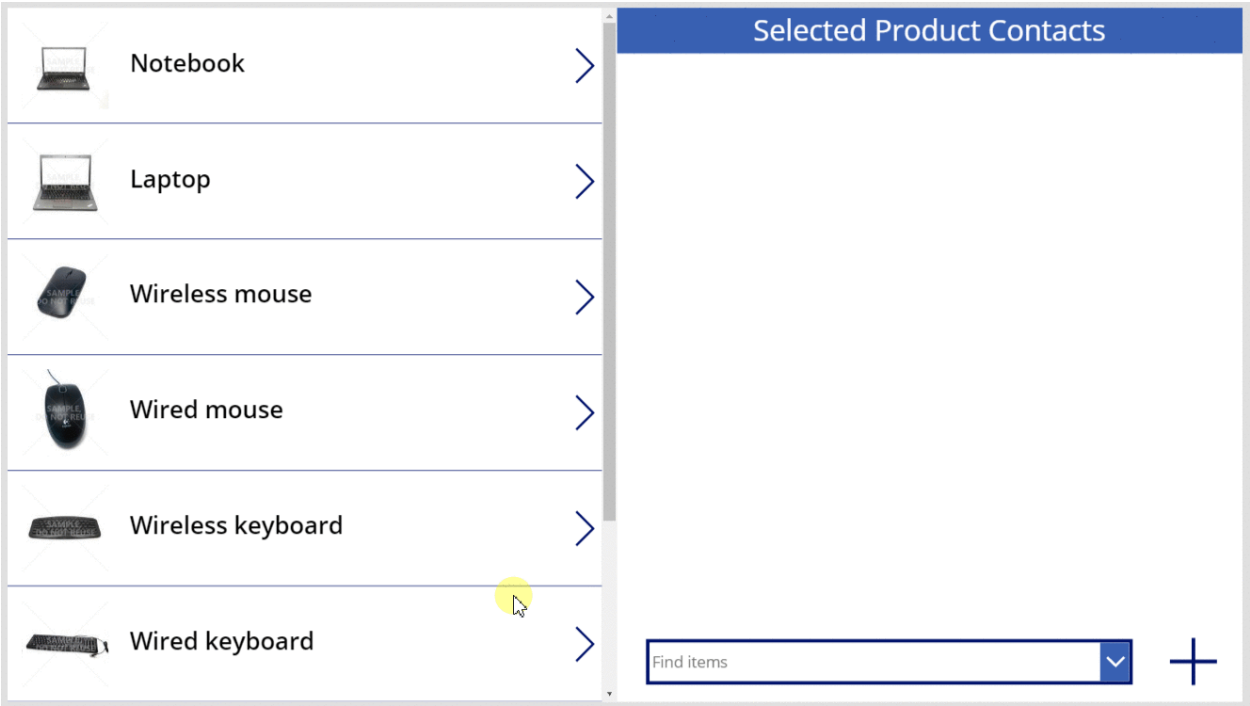
```
Relate( Gallery1.Selected.Contacts, ComboBox1.Selected )
```



With this app, users can now freely relate and unrelate a set of contacts to each product.

- To add a contact to a product, select the contact in the combo box at the bottom of the screen, and then select the **Add** icon.
- To remove a contact from a product, select the **Cancel** icon for that contact.

  Unlike one-to-many, a many-to-many relationship allows users to associate the same contact with multiple products.

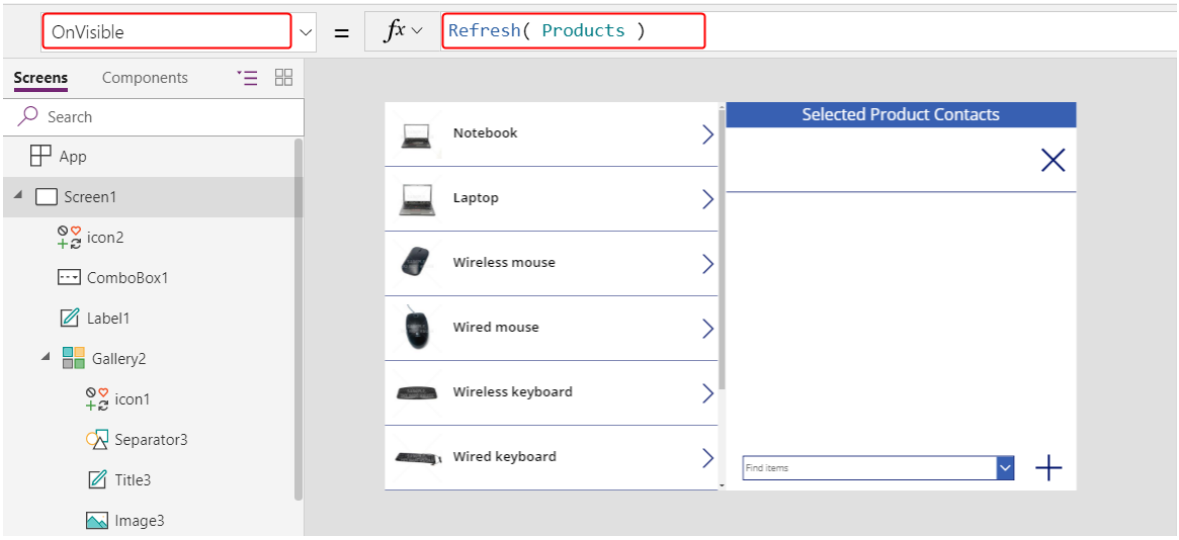


In reverse: relate and unrelate products with multiple contacts

Many-to-many relationships are symmetric. You can extend the example to add products to a contact and then flip between the two screens to show how the relationship appears from either direction.

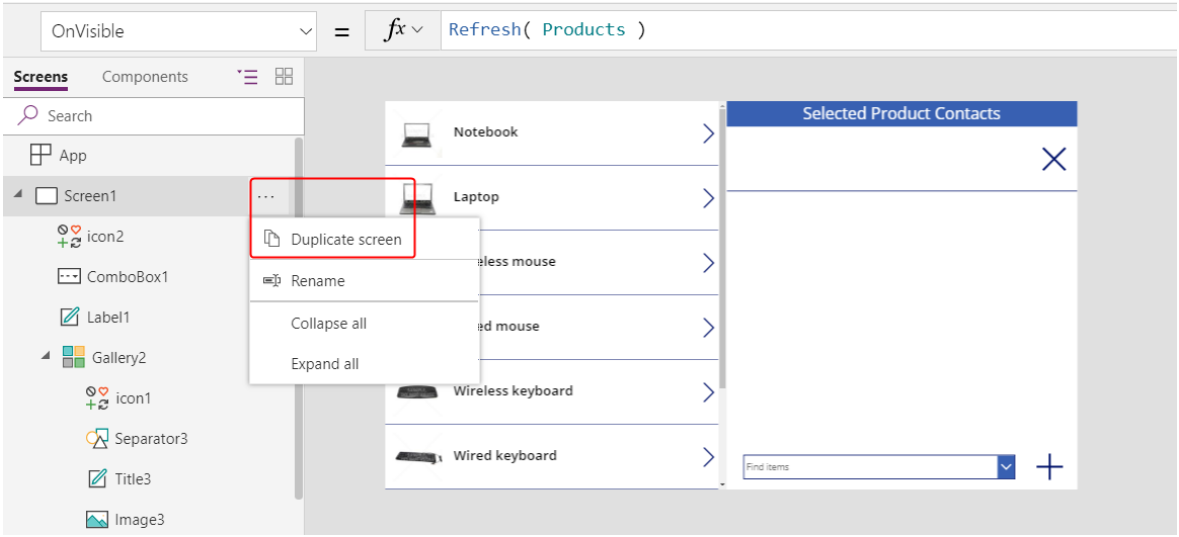
1. Set the **OnVisible** property of **Screen1** to **Refresh(Products)**.

   When you update a one-to-many or many-to-many relationship, only the data of the first argument table of the **Relate** or **Unrelate** call is refreshed. The second must be refreshed manually if you want to flip between the screens of this app.



2. Duplicate **Screen1**.

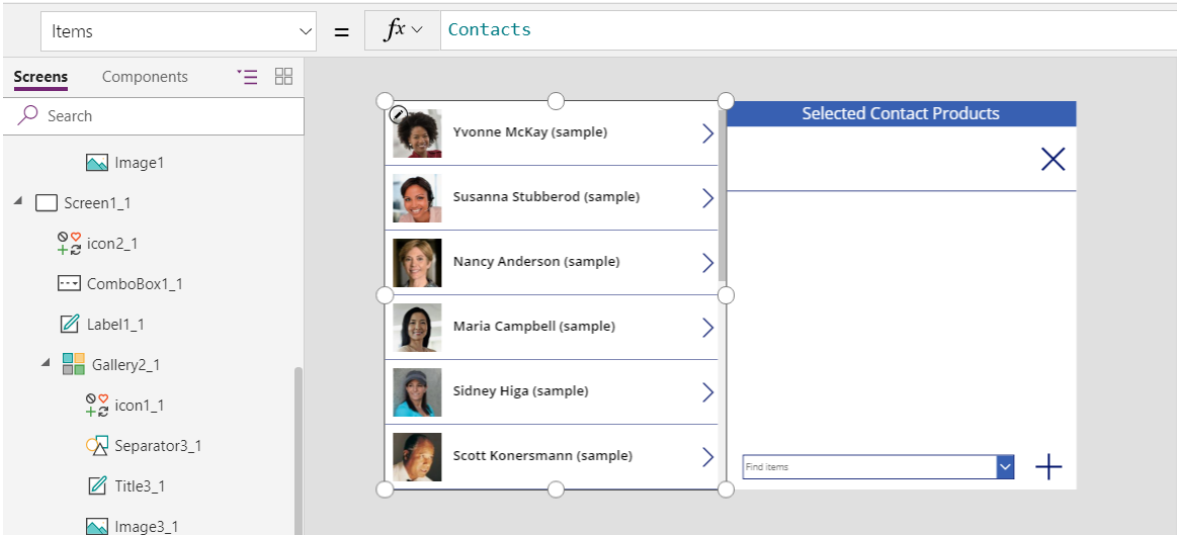
   The duplicate will be named **Screen1_1** and form the basis for looking at the relationships from the contacts side.



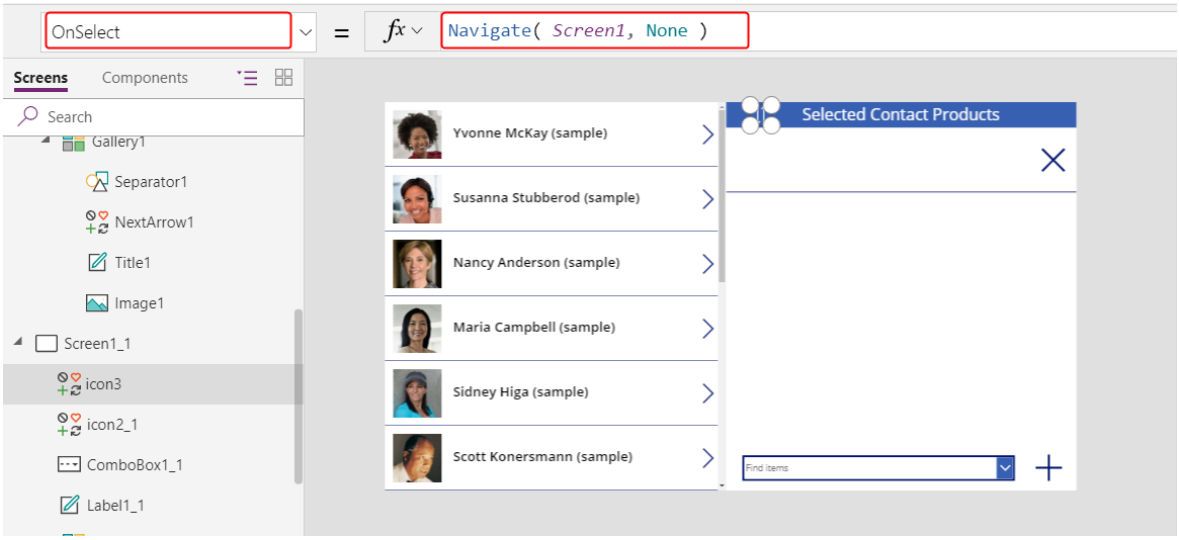
3. To create the reverse view, change these formulas on the controls of **Screen1_1**:

   - Screen1_1.OnVisible = `Refresh( Contacts )`
   - Gallery1_1.Items = `Contacts`
   - Title1_1.Text = `ThisItem.'Full Name'`
   - Label1_1.Text = `"Selected Contact Products"`
   - Gallery2_1.Items = `Gallery1_1.Selected.Products`
   - Title2_1.Text = `ThisItem.Name`
   - Icon1_1.OnSelect = `Unrelate( Gallery1_1.Selected.Products, ThisItem )`
   - ComboBox1_1.Items = `Products`
   - Icon2_1.OnSelect = `Relate( Gallery1_1.Selected.Products, ComboBox1_1.Selected )`

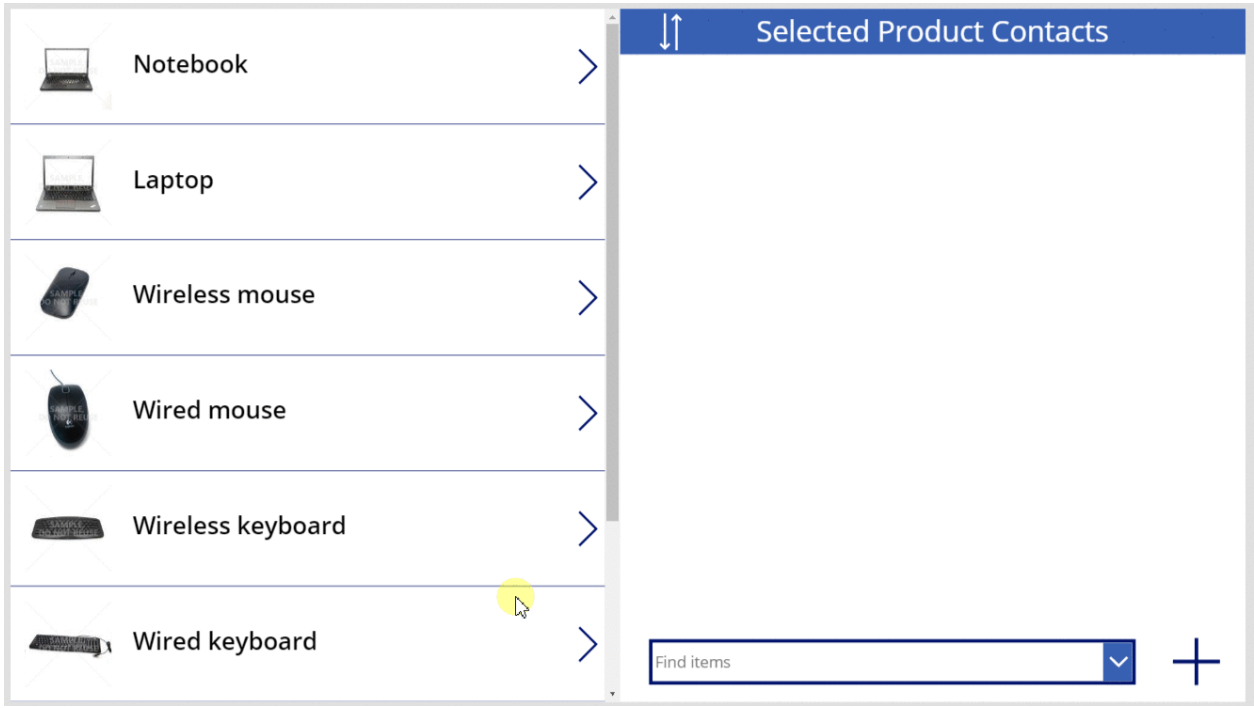
The result will look very similar to the previous screen but comes at the relationship from the **Contacts** side.



4. Insert an **Arrows up down** icon and set its **OnSelect** property to **Navigate(Screen1, None)**. Do the same thing on **Screen1** with the formula **Navigate(Screen1_1, None)**.



With this new screen, users can add a contact to a product and then flip to a view of contacts and see the associated product. The relationships are symmetric and shared between the two screens.


Notebook
Laptop
Wireless mouse
Wired mouse
Wireless keyboard
Wired keyboard
Selected Product Contacts
Find items

Update and UpdateIf functions

06/21/2025

Applies to: ✅ Canvas apps ✅ Model-driven apps

Updates records in a data source.

Description

Update function

Use the **Update** function to replace an entire record in a data source. In contrast, the **UpdateIf** and the **Patch** functions modify one or more values in a record, leaving the other values alone.

For a collection, the entire record must match. Collections allow duplicate records, so multiple records might match. You can use the **RemoveFlags.All** argument to update all copies of a record; otherwise, only one copy of the record is updated.

If the data source generates a column's value automatically, the value of that column must be reaffirmed.

UpdateIf function

Use the **UpdateIf** function to modify one or more values in one or more records that match one or more conditions. The condition can be any formula that results in a **true** or **false** and can reference columns of the data source by name. The function evaluates the condition for each record and modifies any record for which the result is **true**.

To specify a modification, use a change record that contains new property values. If you provide this change record inline with curly braces, property formulas can reference properties of the record that's being modified. You can use this behavior to modify records based on a formula.

Similar to **UpdateIf**, you can also use the **Patch** function to change specific columns of a record without affecting other columns.

Both **Update** and **UpdateIf** return the modified data source as a table. You must use either function in a behavior formula.

Delegation

These functions do not delegate to a data source. However, **UpdateIf** and **RemoveIf** work locally to simulate delegation up to a limit of 500/2000 records. They progressively bring down records beyond the non-delegation limit of 500/2000 records. Records that meet the **If** condition are collected. Generally, a maximum of 500/2000 records are collected separately and then modified per execution. However, more records may be updated if the existing local data cache is large, as the function may have access to more records for evaluation. Only the initial portion of the data source will be retrieved, and then the function will be applied. This may not represent the complete picture. A warning may appear during authoring to remind you of this limitation.

Syntax

Update(DataSource_, *OldRecord*, *NewRecord* [, **RemoveFlags.All**])

- *DataSource* – Required. The data source that contains the record that you want to replace.
- *OldRecord* – Required. The record to replace.
- *NewRecord* – Required. The replacement record. This isn't a change record. The entire record is replaced, and missing properties contain *blank*.
- *RemoveFlags.All* – Optional. In a collection, the same record may appear more than once. Specify the **RemoveFlags.All** argument to update all copies of the record.

UpdateIf(DataSource_, *Condition1*, *ChangeRecord1* [, *Condition2*, *ChangeRecord2*, ...])

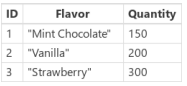
- *DataSource* – Required. The data source that contains the record or records that you want to modify.
- *Conditions* – Required. A formula that evaluates to **true** for the record or records that you want to modify. You can use column names of *DataSource* in the formula. In case multiple *Conditions* are passed, only the *ChangeRecord* related to the first *Condition* that evaluates to **true** is applied.
- *ChangeRecord(s)* - Required. For each corresponding condition, a change record of new property values to apply to records of *DataSource* that satisfy the condition. If you provide the record inline using curly braces, property values of the existing record can be used in the property formulas.

Examples

In these examples, you'll replace or modify records in a data source that's named **IceCream** and that starts with the data in this table:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	300

Expand table

Formula	Description	Result
Update(IceCream, LookUp(IceCream, Flavor="Chocolate"), { ID: 1, Flavor: "Mint Chocolate", Quantity:150 })	Replaces a record from the data source.	 The **IceCream** data source has been modified.
UpdateIf(IceCream, Quantity > 175, { Quantity: Quantity + 10 })	Modifies records that have a **Quantity** that is greater than **175**. The **Quantity** field is incremented by 10, and no other fields are modified.	 The **IceCream** data source has been modified.
Update(IceCream, LookUp(IceCream, Flavor="Strawberry"), { ID: 3, Flavor: "Strawberry Swirl"})	Replaces a record from the data source. The **Quantity** property hasn't been supplied in the replacement record, so that property will be *blank* in the result.	 The **IceCream** data source has been modified.
UpdateIf(IceCream, true, { Quantity: 0 })	Sets the value of the **Quantity** property for all records in the data source to 0.	 The **IceCream** data source has been modified.

Step by step

1. Import or create a collection named **Inventory**, and show it in a gallery as Show data in a gallery describes.

2. Name the gallery **ProductGallery**.

3. Add a slider named **UnitsSold**, and set its **Max** property to this expression:
   ProductGallery.Selected.UnitsInStock

4. Add a button, and set its **OnSelect** property to this formula:
   UpdateIf(Inventory, ProductName = ProductGallery.Selected.ProductName, {UnitsInStock:UnitsInStock-UnitsSold.Value})

5. Press F5, select a product in the gallery, specify a value with the slider, and then select the button.

   The number of units in stock for the product you specified decreases by the amount that you specified.

UpdateContext function

Article • 03/22/2024

Applies to: Canvas apps

Creates or updates context variables of the current screen.

Overview

Use the **UpdateContext** function to create a context variable, which temporarily holds a piece of information, such as the number of times the user has selected a button or the result of a data operation.

Context variables are scoped to a screen, which means that you can't build a formula that refers to a context variable on another screen. If you've used another programming tool, you can think of a context variable as similar to a local variable. Use the **Set** function to work with global variables that are available throughout your app.

Power Apps are based on formulas that automatically recalculate as the user interacts with an app. Context variables don't offer this benefit and can make your app harder to create and understand. Before you use a context variable, review working with variables.

Description

To create or update a context variable, pass a single record to the **UpdateContext** function. In each record, specify the name of a column, which defines or matches the name of the variable, and the value to which you want to set that variable.

- If you specify the name of a variable that you've previously defined, **UpdateContext** sets the value of the variable to the value that you specify.
- If you specify the name of a variable that doesn't yet exist, **UpdateContext** creates a variable with that name and sets the value of that variable to the value that you specify.
- If you've previously defined a variable but don't specify it in this particular **UpdateContext** formula, its value remains the same.

Context variables are implicitly created by using the **UpdateContext** or **Navigate** function. There is no explicit declaration required. If you remove all the **UpdateContext** and **Navigate** references to a context variable, then that context variable will cease to exist. To clear a variable set its value to the result of the **Blank** function.

You can see your variables' values, definitions, and uses with the Variables view under the File menu in the authoring environment.

You reference a context variable in a formula by using the variable's column name. For example, **UpdateContext({ ShowLogo: true })** creates a context variable named **ShowLogo** and sets its value to **true**. You can then use the value of this context variable by using the name **ShowLogo** in a formula. You can write **ShowLogo** as the formula for the **Visible** property of an image control and show or hide that control based on whether the value of the context variable is **true** or **false**.

As the examples later in this topic show, context variables can hold several kinds of information, including these:

- a single value
- a record
- a table
- an object reference
- any result from a formula

A context variable holds its value until the app is closed. If you define a context variable and set its value on a particular screen, that information remains intact even if the user switches to a different screen. Once the app is closed, the context variable's value will be lost and must be recreated when the app is loaded again.

Every context variable is scoped to a screen. If you want to define a context variable on one screen and modify that variable from another screen, you must build a formula that's based on the **Navigate** function. Or use a global variable.

UpdateContext has no return value, and you can use it only within a behavior formula.

Syntax

UpdateContext(*UpdateRecord*)

- *UpdateRecord* – Required. A record that contains the name of at least one column and a value for that column. A context variable is created or updated for each column and value that you specify.

UpdateContext({ *ContextVariable1*: *Value1* [, *ContextVariable2*: *Value2* [, ...]] })

- *ContextVariable1* - Required. The name of a context variable to create or update.
- *Value1* - Required. The value to assign to the context variable.
- *ContextVariable2*: *Value2*, ... - Optional. Additional context variables to create or update and their values.

Examples

Expand table

Formula	Description	Result
UpdateContext({ Counter: 1 })	Creates or modifies the context variable **Counter**, setting its value to **1**.	**Counter** has the value **1**. You can reference that variable by using the name **Counter** in a formula.
UpdateContext({ Counter: 2 })	Sets the value of the **Counter** context variable from the previous example to **2**.	**Counter** has the value **2**.
UpdateContext({ Name: "Lily", Score: 10 })	Creates or modifies the context variables **Name** and **Score**, setting their values to **Lily** and **10** respectively.	**Name** has the value **Lily**, and **Score** has the value **10**.
UpdateContext({ Person: { Name: "Milton", Address: "1 Main St" } })	Creates or modifies the context variable **Person**, setting its value to a record. The record contains two columns, named **Name** and **Address**. The value of the **Name** column is **Milton**, and the value of the **Address** column is **1 Main St**.	**Person** has the value of record **{ Name: "Milton", Address: "1 Main St" }**. Reference this record as a whole with the name **Person**, or reference an individual column of this record with **Person.Name** or **Person.Address**.
UpdateContext({ Person: Patch(Person, {Address: "2 Main St" }) })	Works with the Patch function to update the **Person** context variable by setting the value of the **Address** column to **2 Main St**.	**Person** now has the value of record **{ Name: "Milton", Address: "2 Main St" }**.

Step-by-step example 1

1. Name the default screen **Source**, add another screen, and name it **Target**.

2. On the **Source** screen, add two buttons, and set their **Text** properties so that one says **English** and the other says **Spanish**.

3. Set the **OnSelect** property of the **English** button to this expression:
   Navigate(Target, ScreenTransition.Fade, {Language:"English"})

4. Set the **OnSelect** property of the **Spanish** button to this expression:
   Navigate(Target, ScreenTransition.Fade, {Language:"Spanish"})

5. On the **Target** screen, add a label, and set its **Text** property to this expression:
   If(Language="English", "Hello!", "Hola!")

6. On the **Target** screen, select **Shapes** on the **Insert** tab, and then select the Back arrow.

7. Set the Back arrow's **OnSelect** property to this formula:
   Navigate(Source, ScreenTransition.Fade)

8. From the **Source** screen, press F5, and then select the button for either language.

   On the **Target** screen, the label appears in the language that corresponds to the button that you selected.

9. Select the Back arrow to return to the **Source** screen, and then select the button for the other language.

   On the **Target** screen, the label appears in the language that corresponds to the button that you selected.

10. Press Esc to return to the default workspace.

Step-by-step example 2

1. Open the canvas app where you want to use this formula.
2. Add a new blank screen by selecting **New screen** from the command bar.
3. Add a button, and set its **OnSelect** property to this formula:
   UpdateContext({ Name: "Lily", Score: 10 })

Update and UpdateIf functions

06/21/2025

Applies to: ✅ Canvas apps ✅ Model-driven apps

Updates records in a data source.

Description

Update function

Use the **Update** function to replace an entire record in a data source. In contrast, the **UpdateIf** and the **Patch** functions modify one or more values in a record, leaving the other values alone.

For a collection, the entire record must match. Collections allow duplicate records, so multiple records might match. You can use the **RemoveFlags.All** argument to update all copies of a record; otherwise, only one copy of the record is updated.

If the data source generates a column's value automatically, the value of that column must be reaffirmed.

UpdateIf function

Use the **UpdateIf** function to modify one or more values in one or more records that match one or more conditions. The condition can be any formula that results in a **true** or **false** and can reference columns of the data source by name. The function evaluates the condition for each record and modifies any record for which the result is **true**.

To specify a modification, use a change record that contains new property values. If you provide this change record inline with curly braces, property formulas can reference properties of the record that's being modified. You can use this behavior to modify records based on a formula.

Similar to **UpdateIf**, you can also use the **Patch** function to change specific columns of a record without affecting other columns.

Both **Update** and **UpdateIf** return the modified data source as a table. You must use either function in a behavior formula.

Delegation

These functions do not delegate to a data source. However, **UpdateIf** and **RemoveIf** work locally to simulate delegation up to a limit of 500/2000 records. They progressively bring down records beyond the non-delegation limit of 500/2000 records. Records that meet the **If** condition are collected. Generally, a maximum of 500/2000 records are collected separately and then modified per execution. However, more records may be updated if the existing local data cache is large, as the function may have access to more records for evaluation. Only the initial portion of the data source will be retrieved, and then the function will be applied. This may not represent the complete picture. A warning may appear during authoring to remind you of this limitation.

Syntax

Update(DataSource_, *OldRecord*, *NewRecord* [, **RemoveFlags.All**])

- *DataSource* – Required. The data source that contains the record that you want to replace.
- *OldRecord* – Required. The record to replace.
- *NewRecord* – Required. The replacement record. This isn't a change record. The entire record is replaced, and missing properties contain *blank*.
- *RemoveFlags.All* – Optional. In a collection, the same record may appear more than once. Specify the **RemoveFlags.All** argument to update all copies of the record.

UpdateIf(DataSource_, *Condition1*, *ChangeRecord1* [, *Condition2*, *ChangeRecord2*, ...])

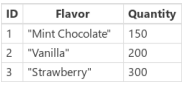
- *DataSource* – Required. The data source that contains the record or records that you want to modify.
- *Conditions* – Required. A formula that evaluates to **true** for the record or records that you want to modify. You can use column names of *DataSource* in the formula. In case multiple *Conditions* are passed, only the *ChangeRecord* related to the first *Condition* that evaluates to **true** is applied.
- *ChangeRecord(s)* - Required. For each corresponding condition, a change record of new property values to apply to records of *DataSource* that satisfy the condition. If you provide the record inline using curly braces, property values of the existing record can be used in the property formulas.

Examples

In these examples, you'll replace or modify records in a data source that's named **IceCream** and that starts with the data in this table:

ID	Flavor	Quantity
1	"Chocolate"	100
2	"Vanilla"	200
3	"Strawberry"	300

Expand table

Formula	Description	Result
Update(IceCream, LookUp(IceCream, Flavor="Chocolate"), { ID: 1, Flavor: "Mint Chocolate", Quantity:150 })	Replaces a record from the data source.	 The **IceCream** data source has been modified.
UpdateIf(IceCream, Quantity > 175, { Quantity: Quantity + 10 })	Modifies records that have a **Quantity** that is greater than **175**. The **Quantity** field is incremented by 10, and no other fields are modified.	 The **IceCream** data source has been modified.
Update(IceCream, LookUp(IceCream, Flavor="Strawberry"), { ID: 3, Flavor: "Strawberry Swirl"})	Replaces a record from the data source. The **Quantity** property hasn't been supplied in the replacement record, so that property will be *blank* in the result.	 The **IceCream** data source has been modified.
UpdateIf(IceCream, true, { Quantity: 0 })	Sets the value of the **Quantity** property for all records in the data source to 0.	 The **IceCream** data source has been modified.

Step by step

1. Import or create a collection named **Inventory**, and show it in a gallery as Show data in a gallery describes.

2. Name the gallery **ProductGallery**.

3. Add a slider named **UnitsSold**, and set its **Max** property to this expression:
   ProductGallery.Selected.UnitsInStock

4. Add a button, and set its **OnSelect** property to this formula:
   UpdateIf(Inventory, ProductName = ProductGallery.Selected.ProductName, {UnitsInStock:UnitsInStock-UnitsSold.Value})

5. Press F5, select a product in the gallery, specify a value with the slider, and then select the button.

   The number of units in stock for the product you specified decreases by the amount that you specified.

Lower, Upper, and Proper functions

Article • 04/20/2025

Expand table

Functions	Applies to
Lower **Upper**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Proper	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts letters in a string of text to all lowercase, all uppercase, or proper case.

Description

The **Lower**, **Upper**, and **Proper** functions convert the case of letters in strings.

- **Lower** converts any uppercase letters to lowercase.
- **Upper** converts any lowercase letters to uppercase.
- **Proper** converts the first letter in each word to uppercase if it's lowercase and converts any other uppercase letters to lowercase.

All three functions ignore characters that aren't letters.

If you pass a single string, the return value is the converted version of that string. If you pass a single-column table that contains strings, the return value is a single-column table of converted strings. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

Lower(*String*)
Upper(*String*)
Proper(*String*)

- *String* - Required. The string to convert.

Lower(*SingleColumnTable*)
Upper(*SingleColumnTable*)
Proper(*SingleColumnTable*)

- *SingleColumnTable* - Required. A single-column table of strings to convert.

Examples

Single string

The examples in this section use a text-input control, named **Author**, as their data source. The control contains the string "E. E. CummINGS".

Expand table

Formula	Description	Result
Lower(Author.Text)	Converts any uppercase letters in the string to lowercase.	"e. e. cummings"
Upper(Author.Text)	Converts any lowercase letters in the string to uppercase.	"E. E. CUMMINGS"
Proper(Author.Text)	Converts the first letter of each word to uppercase if it's lowercase, and converts any other uppercase letters to lowercase.	"E. E. Cummings"

Single-column table

The examples in this section convert strings from the **Address** column of the **People** data source, which contains this data:

Expand table

Name	Address
"Jean"	"123 Main St NE"
"Fred"	"789 SW 39th #3B"

Each formula returns a single-column table that contains the converted strings.

Expand table

Formula	Description	Result
Lower(ShowColumns(People, "Address"))	Converts any letter that's lowercase to uppercase.	A single-column table with a `Value` column containing the

Formula	Description	Result
		following values: "123 main st ne", "789 sw 39th #3b"
Upper(ShowColumns(People, "Address"))	Converts any letter that's lowercase to uppercase.	A single-column table with a `Value` column containing the following values: "123 MAIN ST NE", "789 SW 39TH #3B"
Proper(ShowColumns(People, "Address"))	Converts any first letter of a word that's lowercase to uppercase, and converts any other letter that's uppercase to lowercase.	A single-column table with a `Value` column containing the following values: "123 Main St Ne", "789 Sw 39th #3b"

Step-by-step example

1. Add a **Text input** control, and name it **Source**.
2. Add a label, and set its **Text** property to this function:
   Proper(Source.Text)
3. Press F5, and then type **WE ARE THE BEST!** into the **Source** box.
   The label shows **We Are The Best!**

User function

Article • 11/24/2024

Applies to: Canvas apps Power Pages

Returns information about the current user.

Description

The **User** function returns a record of information about the current user:

Expand table

Property	Description
User().Email	Email address of the current user. The `User().Email` function returns the user's UPN and not the SMTP email address.
User().EntraObjectId	Microsoft Entra Object ID of the current user, useful for calling APIs that use this value. This is a GUID value and unique for each user.
User().FullName	Full name of the current user, including first and last names.
User().Image	Image of the current user. This will be an image URL of the form "blob:*identifier*". Set the **Image** property of the **Image** control to this value to display the image in the app.

> **Note**
>
> The information returned is for the current Power Apps user. It will match the "Account" information that is displayed in the Power Apps players and studio, which can be found outside of any authored apps. This may not match the current user's information in Office 365 or other services.

> **Note**
>
> If you published your application with a User function prior to March 2020, you may find that it, intermittently, will not retrieve photos. The issues were fixed in the late March 2020 release. To take advantage of the updated implementation, simply re-open your application, save it, and republish it.

Syntax

User()

Examples

The current Power Apps user has the following information:

- Full Name: **"John Doe"**
- Email address: **"john.doe@contoso.com"**
- Entra Object Id: **aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb** *(GUID)*
- Image: 

Expand table

Formula	Description	Result
User()	Record of all information for the current Power Apps user.	{ FullName: "John Doe", Email: "john.doe@contoso.com", Image: "blob:1234...5678", EntraObjectId: a90c6800-e58c-4495-81f7-55819b56fe2a }
User().Email	The email address of the current Power Apps user.	"john.doe@contoso.com"
User().FullName	The full name of the current Power Apps user.	"John Doe"
User().EntraObjectId	Microsoft Entra Object ID of the current user.	aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb *(GUID)*
User().Image	The image URL for the current Power Apps user. Set the **Image**	"blob:1234...5678"

Formula	Description	Result
	property of the **Image** control to this value to display the image in the app.	With **ImageControl.Image**: 

Feedback

Was this page helpful? Yes No

Provide product feedback

Now, Today, IsToday, UTCNow, UTCToday, IsUTCToday functions

Article • 04/20/2025

Expand table

Functions	Applies to
Now	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Today **IsToday**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
UTCNow **UTCToday**	✅ Dataverse formula columns ✅ Power Pages

Returns the current date and time, and tests whether a date/time value is today.

> ⓘ Note
>
> PAC CLI pac power-fx commands do not support the **UTCNow**, **UTCToday**, and **IsUTCToday** functions.

Description

The **Now** function returns the current date and time as a date/time value.

The **Today** function returns the current date as a date/time value. The time portion is midnight. **Today** has the same value throughout a day, from midnight today to midnight tomorrow.

The **IsToday** function tests whether a date/time value is between midnight today and midnight tomorrow. This function returns a Boolean (**true** or **false**) value.

Now, **Today**, and **IsToday** functions work with the local time of the current user.

UTCNow, **UTCToday**, and **IsUTCToday** functions are the same as their non-UTC counterparts but work with time zone independent values and use Coordinated Universal Time (UTC).

> ⓘ Note

- **UTCNow**, **UTCToday**, and **IsUTCToday** are only available in Microsoft Dataverse for Teams formula columns, and only for time-independent fields and values.
- **Now**, **Today**, and **IsToday** are not available in Dataverse for Teams formula columns as evaluations are done without the knowledge of the current user's local time zone. More information: Work with formula table columns in Dataverse for Teams

See Date, Time, and DateTime in the data types documentation and working with dates and times for more information.

Volatile Functions

Now, **Today**, **UTCNow**, and **UTCToday** are volatile functions. These functions return a different value for each evaluation.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Now()** will not change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it is part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = DateAdd(Now(), Slider1.Value, Minutes)** then the current time is retrieved each time the Slider control's value changes and the label's text property is reevaluated.

When used in a behavior formula, volatile functions will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Using the user's local time

Now()

Today()

IsToday(*DateTime*)

- *DateTime* - Required. The date/time value to test.

Using Coordinated Universal Time (UTC)

UTCNow()

UTCToday()

IsUTCToday(*TimeZoneIndependentTime*)

- *TimeZoneIndependentDateTime* - Required. The time zone independent date/time value to test.

Examples

For the examples in this section, the current time is **8:58 PM** on **July 11, 2021** in the Pacific Time Zone (UTC-8) and the language is **en-us**.

Expand table

Formula	Description	Result
Text(Now(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time in the user's time zone, and displays it as a string.	"07/11/2021 20:58:00"
Text(Today(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date only, leaving the time portion as midnight, and displays it as a string.	"07/12/2021 00:00:00"
IsToday(Now())	Tests whether the current date and time is between midnight today and midnight tomorrow.	**true**
IsToday(Today())	Tests whether the current date is between midnight today and midnight tomorrow.	**true**
Text(DateAdd(Now(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time, adds 12 days to the result, and displays it as a string.	"07/23/2021 20:58:00"
Text(DateAdd(Today(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date, adds 12 days to the result, and displays it as a string.	"07/23/2021 00:00:00"
IsToday(DateAdd(Now(), 12))	Tests whether the current date and time, plus 12 days, is between midnight today and midnight tomorrow.	**false**
IsToday(DateAdd(Today(), 12))	Tests whether the current date, plus 12 days, is between midnight today and midnight tomorrow.	**false**
Hour(UTCNow())	Retrieves the current date and time in UTC and extracts the hour only, which is 8 hours ahead of local time.	4

Formula	Description	Result
Day(UTCToday())	Retrieves the current date only in UTC and extracts the day, which is 1 day ahead of local time.	12
IsUTCToday(UTCNow())	Tests whether the current date and time is between midnight today and midnight tomorrow, all in UTC time.	**true**
IsUTCToday(UTCToday())	Tests whether the current date is between midnight today and midnight tomorrow, all in UTC time.	**true**

Display a clock that updates in real time

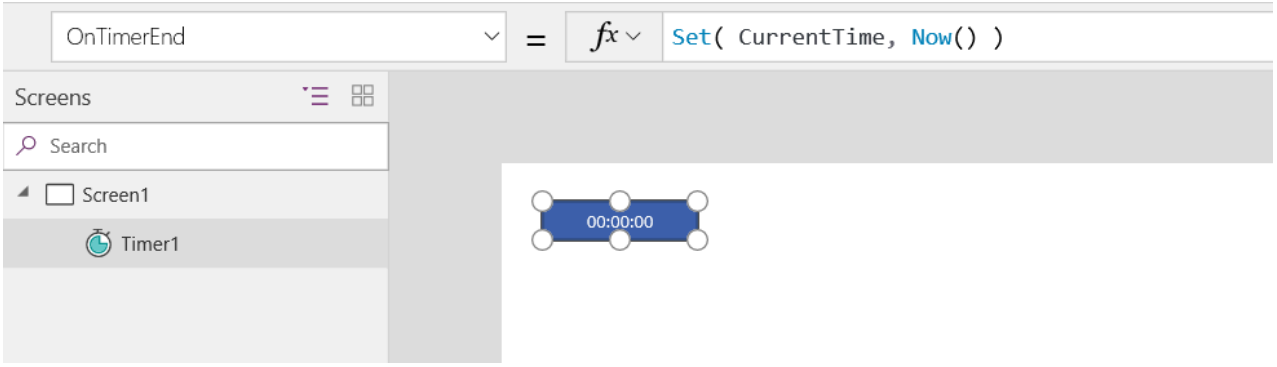
1. Add a Timer control, set its **Duration** property to **1000**, and set its **Repeat** property to **true**.

   The timer will run for one second, automatically start over, and continue that pattern.

2. Set the control's **OnTimerEnd** property to this formula:

   Set(CurrentTime, Now())

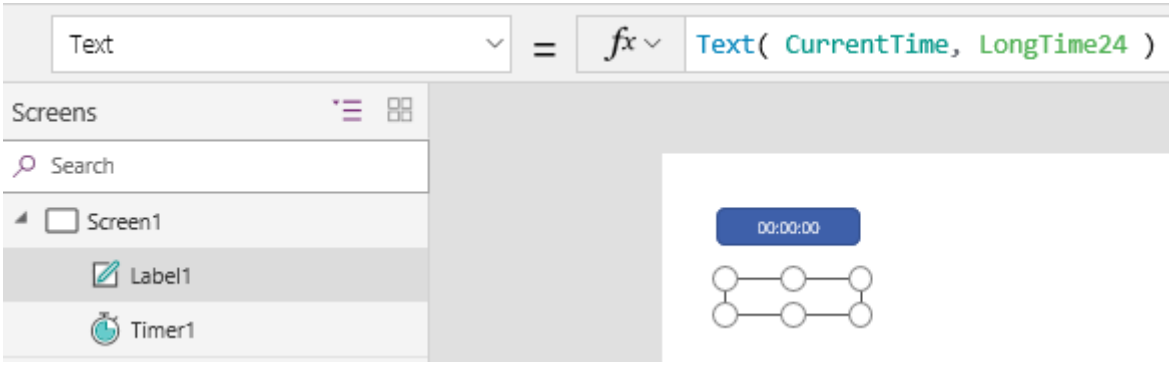
   Whenever the timer starts over (after each second), this formula sets the **CurrentTime** global variable to the current value of the **Now** function.

3. Add a Label control, and set its **Text** property to this formula:

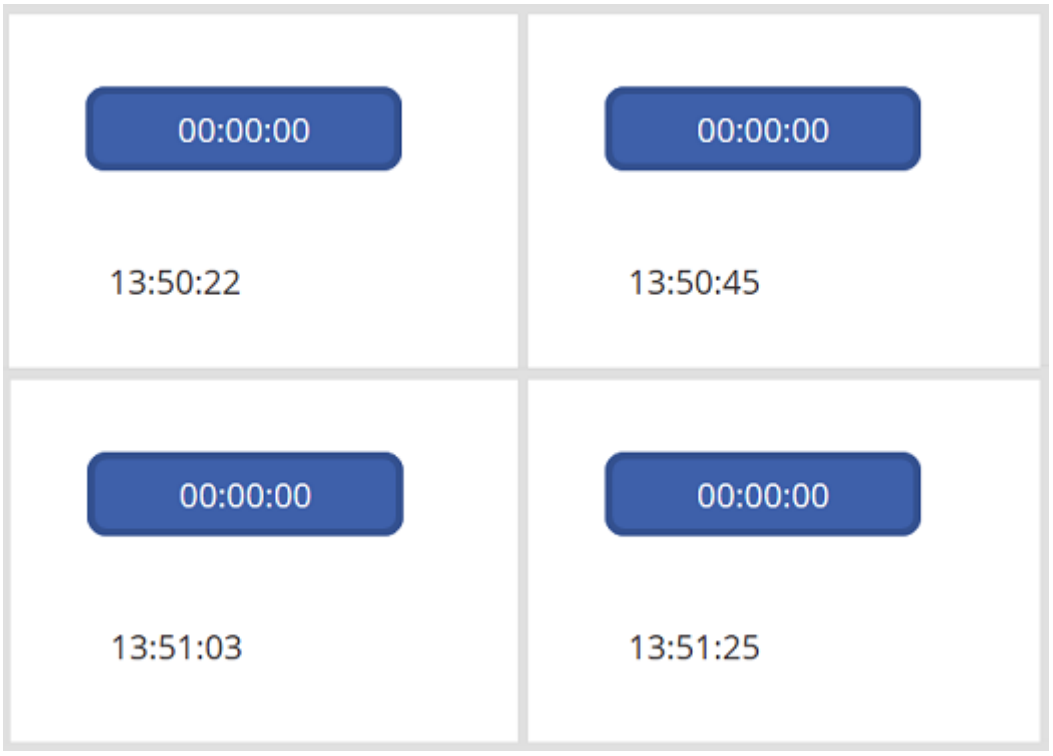
   Text(CurrentTime, LongTime24)

   Use the Text function to format the date and time however you want, or set this property to just **CurrentTime** to show hours and minutes but not seconds.



4. Preview the app by pressing F5, and then start the timer by clicking or tapping it.

   The label continually shows the current time, down to the second.



5. Set the timer's **AutoStart** property to **true** and its **Visible** property to **false**.

   The timer is invisible and starts automatically.

6. Set the screen's OnStart property so that the **CurrentTime** variable has a valid value, as in this example:

   Set(CurrentTime, Now())

   The label appears as soon as the app starts (before the timer runs for a full second).

Now, Today, IsToday, UTCNow, UTCToday, IsUTCToday functions

Article • 04/20/2025

Expand table

Functions	Applies to
Now	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
Today **IsToday**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
UTCNow **UTCToday**	✅ Dataverse formula columns ✅ Power Pages

Returns the current date and time, and tests whether a date/time value is today.

> ⓘ Note
>
> PAC CLI pac power-fx commands do not support the **UTCNow**, **UTCToday**, and **IsUTCToday** functions.

Description

The **Now** function returns the current date and time as a date/time value.

The **Today** function returns the current date as a date/time value. The time portion is midnight. **Today** has the same value throughout a day, from midnight today to midnight tomorrow.

The **IsToday** function tests whether a date/time value is between midnight today and midnight tomorrow. This function returns a Boolean (**true** or **false**) value.

Now, **Today**, and **IsToday** functions work with the local time of the current user.

UTCNow, **UTCToday**, and **IsUTCToday** functions are the same as their non-UTC counterparts but work with time zone independent values and use Coordinated Universal Time (UTC).

> ⓘ Note

- **UTCNow**, **UTCToday**, and **IsUTCToday** are only available in Microsoft Dataverse for Teams formula columns, and only for time-independent fields and values.
- **Now**, **Today**, and **IsToday** are not available in Dataverse for Teams formula columns as evaluations are done without the knowledge of the current user's local time zone. More information: Work with formula table columns in Dataverse for Teams

See Date, Time, and DateTime in the data types documentation and working with dates and times for more information.

Volatile Functions

Now, **Today**, **UTCNow**, and **UTCToday** are volatile functions. These functions return a different value for each evaluation.

When used in a data flow formula, a volatile function will only return a different value if the formula in which it appears is reevaluated. If nothing else changes in the formula then it will have the same value throughout the execution of your app.

For example, a label control with **Label1.Text = Now()** will not change while your app is active. Only closing and reopening the app will result in a new value.

The function will be reevaluated if it is part of a formula in which something else has changed. For example, if we change our example to involve a slider control with **Label1.Text = DateAdd(Now(), Slider1.Value, Minutes)** then the current time is retrieved each time the Slider control's value changes and the label's text property is reevaluated.

When used in a behavior formula, volatile functions will be evaluated each time the behavior formula is evaluated. See below for an example.

Syntax

Using the user's local time

Now()

Today()

IsToday(*DateTime*)

- *DateTime* - Required. The date/time value to test.

Using Coordinated Universal Time (UTC)

UTCNow()

UTCToday()

IsUTCToday(*TimeZoneIndependentTime*)

- *TimeZoneIndependentDateTime* - Required. The time zone independent date/time value to test.

Examples

For the examples in this section, the current time is **8:58 PM** on **July 11, 2021** in the Pacific Time Zone (UTC-8) and the language is **en-us**.

Expand table

Formula	Description	Result
Text(Now(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time in the user's time zone, and displays it as a string.	"07/11/2021 20:58:00"
Text(Today(), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date only, leaving the time portion as midnight, and displays it as a string.	"07/12/2021 00:00:00"
IsToday(Now())	Tests whether the current date and time is between midnight today and midnight tomorrow.	**true**
IsToday(Today())	Tests whether the current date is between midnight today and midnight tomorrow.	**true**
Text(DateAdd(Now(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date and time, adds 12 days to the result, and displays it as a string.	"07/23/2021 20:58:00"
Text(DateAdd(Today(), 12), "mm/dd/yyyy hh:mm:ss")	Retrieves the current date, adds 12 days to the result, and displays it as a string.	"07/23/2021 00:00:00"
IsToday(DateAdd(Now(), 12))	Tests whether the current date and time, plus 12 days, is between midnight today and midnight tomorrow.	**false**
IsToday(DateAdd(Today(), 12))	Tests whether the current date, plus 12 days, is between midnight today and midnight tomorrow.	**false**
Hour(UTCNow())	Retrieves the current date and time in UTC and extracts the hour only, which is 8 hours ahead of local time.	4

Formula	Description	Result
Day(UTCToday())	Retrieves the current date only in UTC and extracts the day, which is 1 day ahead of local time.	12
IsUTCToday(UTCNow())	Tests whether the current date and time is between midnight today and midnight tomorrow, all in UTC time.	**true**
IsUTCToday(UTCToday())	Tests whether the current date is between midnight today and midnight tomorrow, all in UTC time.	**true**

Display a clock that updates in real time

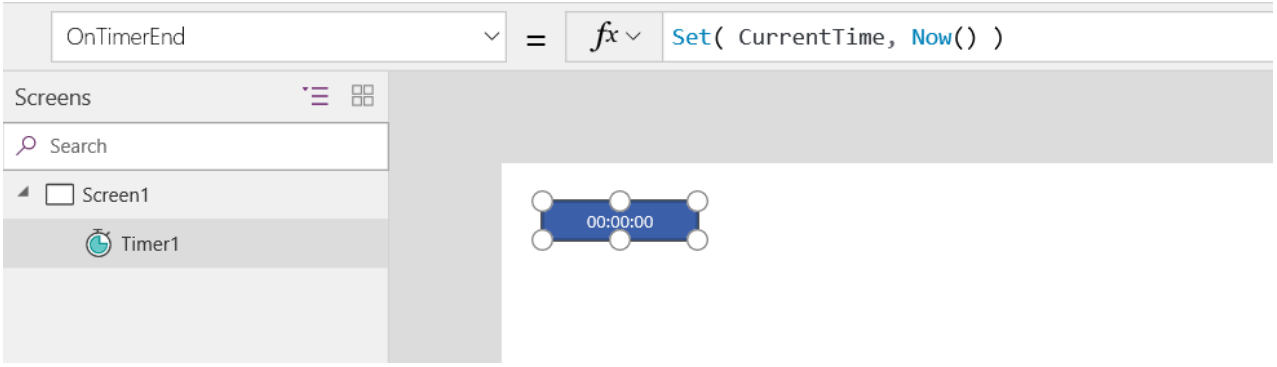
1. Add a Timer control, set its **Duration** property to **1000**, and set its **Repeat** property to **true**.

   The timer will run for one second, automatically start over, and continue that pattern.

2. Set the control's **OnTimerEnd** property to this formula:

   Set(CurrentTime, Now())

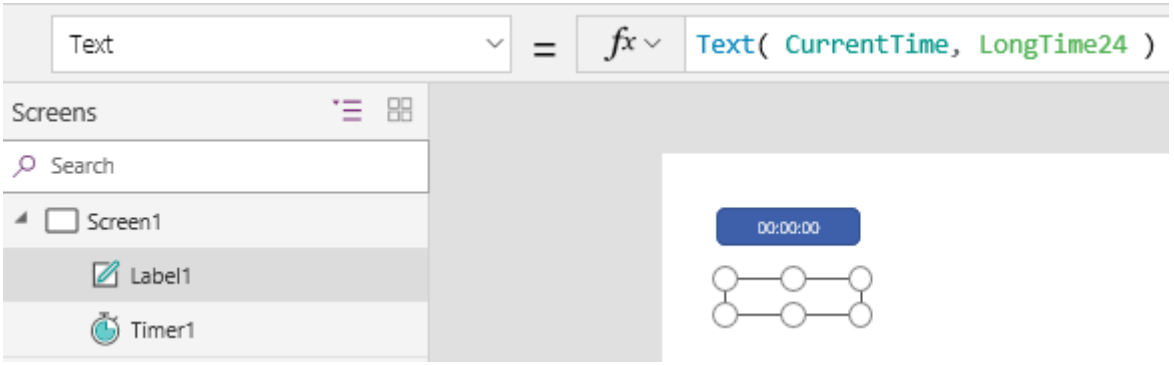
   Whenever the timer starts over (after each second), this formula sets the **CurrentTime** global variable to the current value of the **Now** function.

3. Add a Label control, and set its **Text** property to this formula:

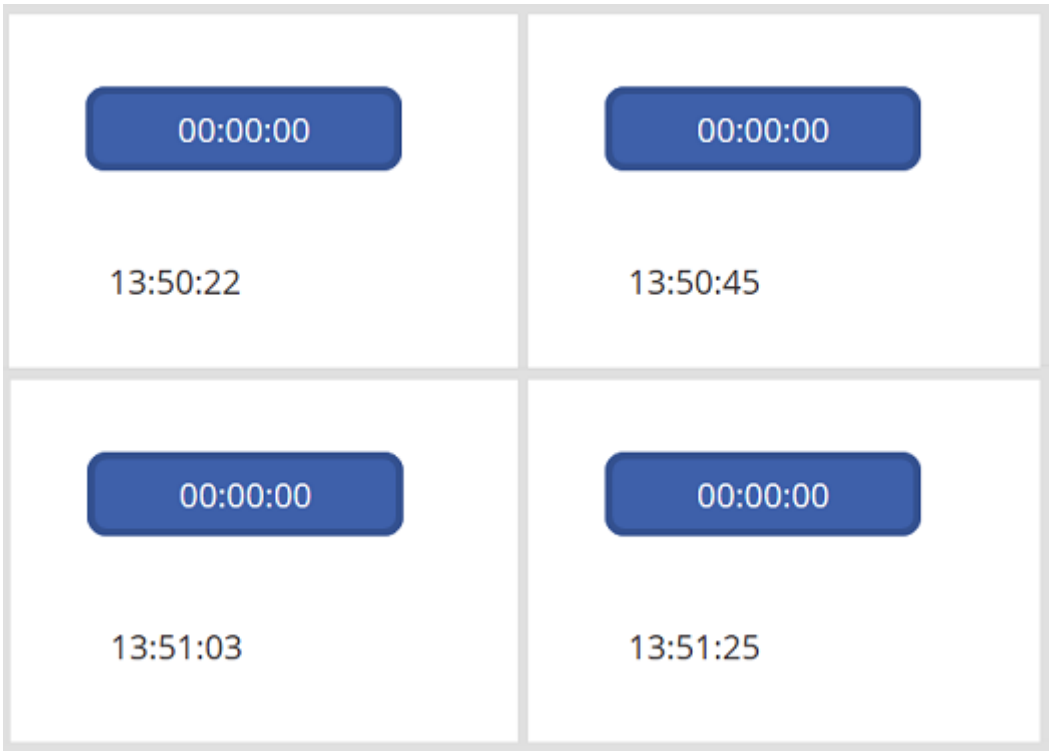
   Text(CurrentTime, LongTime24)

   Use the Text function to format the date and time however you want, or set this property to just **CurrentTime** to show hours and minutes but not seconds.



4. Preview the app by pressing F5, and then start the timer by clicking or tapping it.

   The label continually shows the current time, down to the second.



5. Set the timer's **AutoStart** property to **true** and its **Visible** property to **false**.

   The timer is invisible and starts automatically.

6. Set the screen's OnStart property so that the **CurrentTime** variable has a valid value, as in this example:

   Set(CurrentTime, Now())

   The label appears as soon as the app starts (before the timer runs for a full second).

Validate function

Article • 03/22/2024

Applies to: Canvas apps Model-driven apps

The **Validate** function checks whether the value of a single column or a complete record is valid for a data source.

Description

Before a user submits a data change, you can provide immediate feedback on the validity of that submission, resulting in a better user experience.

Data sources can provide information on what constitutes valid values within a record. This information can include many constraints, such as these examples:

- whether a column requires a value
- how long a string of text can be
- how high and low a number can be
- how early and late a date can be

The **Validate** function uses this information to determine whether a value is valid and to return an appropriate error message if not. You can use the **DataSourceInfo** function to view the same information that **Validate** uses.

Data sources vary in how much validation information they provide, including not providing any at all. **Validate** can only verify values based on this information. Even if **Validate** doesn't find a problem, applying the data change may still fail. You can use the **Errors** function to obtain information about the failure.

If **Validate** finds a problem, the function returns an error message that you can show to the user of the app. If all values are valid, **Validate** returns blank. When you work with a collection that has no validation information, values are always valid.

Syntax

Validate(*DataSource*, *Column*, *Value*)

- *DataSource* – Required. The data source to validate with.
- *Column* – Required. The column to validate.
- *Value* – Required. The value for the selected column to be validated.

Validate(*DataSource*, *OriginalRecord*, *Updates*)

- *DataSource* – Required. The data source to validate with.
- *OriginalRecord* - Required. The record to which updates are to be validated.
- *Updates* - Required. The changes to apply to the original record.

Examples

For these examples, values in the **Percentage** column of the **Scores** data source must be between 0 and 100, inclusive. If the data passes validation, the function returns *blank*. Otherwise, the function returns an error message.

Validate with a single column

Expand table

Formula	Description	Result
Validate(Scores, Percentage, 10)	Checks whether **10** is a valid value for the **Percentage** column in the **Scores** data source.	*blank*
Validate(Scores, Percentage, 120)	Checks whether **120** is a valid value for the **Percentage** column in the **Scores** data source.	"Values must be between 0 and 100."

Validate with a complete record

Expand table

Formula	Description	Result
Validate(Scores, EditRecord, Gallery.Updates)	Checks whether values in all columns are valid for the **Scores** data source. In this example, the value in the **Percentage** column is **10**.	*blank*
Validate(Scores, EditRecord, Gallery.Updates)	Checks whether values in all columns are valid for the **Scores** data source. In this example, the value in the **Percentage** column is **120**.	"Values must be between 0 and 100."

Decimal, Float, and Value functions

06/19/2025

Expand table

Functions	Applies to
Decimal **Float**	✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Power Platform CLI ✅ Dataverse functions
Value	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Converts a string of text or other types to a number.

Description

> **Note**
>
> Power Apps only supports the **Value** function and it returns a **Float** value. Support for the **Decimal** and **Float** functions will be added soon.

Use the **Decimal**, **Float**, and **Value** functions to convert a string of text that contains number characters to a numeric value. Use these function when you need to perform calculations on numbers that were entered as text by a user. These functions can also be used to convert other types to a number, such as date/time and Boolean.

The **Value** function will return the default numeric data type for the Power Fx host you are using, which is usually **Decimal** and the best choice for most situations. Use the **Decimal** and **Float** functions when you need a specific data type for a particular scenario, such as a scientific calculation with a very large number. For more details on working with these data types, see the Numbers section of Data types.

Different languages interpret **,** and **.** differently. By default, the text is interpreted in the language of the current user. You can specify the language to use with a language tag, using the same language tags that are returned by the **Language** function.

Notes on the format of the string:

- The string may be prefixed with the currency symbol for the current language. The currency symbol is ignored. Currency symbols for other languages aren't ignored.

- The string may be include a percent sign (%) at the end, indicating that it's a percentage. The number will be divided by 100 before being returned. Percentages and currency symbols can't be mixed.
- The string may be in scientific notation, with 12 x 10^3 expressed as "12e3".

If the number isn't in a proper format, these functions will return an error.

To convert date and time values, use the **DateValue**, **TimeValue**, or **DateTimeValue** functions.

Syntax

Decimal(*String* [, *LanguageTag*])
Float(*String* [, *LanguageTag*])
Value(*String* [, *LanguageTag*])

- *String* - Required. String to convert to a numeric value.
- *LanguageTag* - Optional. The language tag in which to parse the string. If not specified, the language of the current user is used.

Decimal(*Dynamic*)
Float(*Dynamic*)
Value(*Dynamic*)

- *Dynamic* - Required. **Dynamic** value that represents a number. Acceptable values are dependent on the untyped provider. For **JSON**, the dynamic value is expected to be a JSON number, boolean, or text that can be converted to a number. Keep in mind that locale-related formats are important considerations when communicating with external systems.

Examples

The user running these formulas is located in the United States and has selected English as their language. The **Language** function is returning "en-US". The Power Fx host uses **Decimal** by default.

Value and Decimal

Since we are using a host that has **Decimal** as the default, **Value** and **Decimal** will return the same results.

Expand table

Formula	Description	Result
Value("123.456") **Decimal("123.456")**	The default language of "en-US" will be used, which uses a period as the decimal separator.	123.456 (**Decimal**)
Value("123.456", "es-ES") **Decimal("123.456", "es-ES")**	"es-ES" is the language tag for Spanish in Spain. In Spain, a period is a thousands separator.	123456 (**Decimal**)
Value("123,456") **Decimal("123,456")**	The default language of "en-US" will be used, which uses a comma as a thousands separator.	123456 (**Decimal**)
Value("123,456", "es-ES") **Decimal("123,456", "es-ES")**	"es-ES" is the language tag for Spanish in Spain. In Spain, a comma is the decimal separator.	123.456 (**Decimal**)
Value("12.34%") **Decimal("12.34%")**	The percentage sign at the end of the string indicates that this is a percentage.	0.1234 (**Decimal**)
Value("$ 12.34") **Decimal("$ 12.34")**	The currency symbol for the current language is ignored.	12.34 (**Decimal**)
Value("24e3") **Decimal("24e3")**	Scientific notation for 24 x 10^3.	24000 (**Decimal**)
Value(true) **Decimal(true)**	Converts a Boolean to a number, 0 for *false* and 1 for *true*	1 **Decimal**

Float

The **Float** function will have very close to the same results as above. Since 123.456 cannot be precisely represented in **Float**, the result is an approximation that is very close (123.456000000000003069544618484E2) and compounding rounding errors calculations could result in an unexpected result. The resulting type will be **Float** instead.

Where things diverge is if larger or smaller numbers are used.

Expand table

Formula	Description	Result
Float(1e100)	Because the literal number `1e100` is beyond the range of a **Decimal**, this results in an error before ever calling the **Float** function.	*error (overflow)*
Decimal(1e100)	Same problem as with the **Float** function.	*error (overflow)*

Formula	Description	Result
Float("1e100")	The number in the text string is within the range of **Float** numbers.	1e100 **Float**
Decimal("1e100")	The number in the text string is beyond the range of **Decimal** numbers.	*error (overflow)*
Float("10000000000.0000000001")	The number in the text string is within the range of **Float** numbers. However, the number requires more precision than a **Float** can provide and will be truncated.	1 (**Float**)
Decimal("10000000000.0000000001")	The number in the text string is within both the range and precision of a **Decimal** numbers.	10000000000.0000000001 (**Decimal**)

Average, Max, Min, StdevP, Sum, and VarP functions

Article • 04/20/2025

Expand table

Functions	Applies to
Average **Max** **Min** **Sum**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages
StdevP **VarP**	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Aggregate functions that summarize a set of numbers.

Description

The **Average** function calculates the average, or arithmetic mean, of its arguments.

The **Max** function finds the maximum value.

The **Min** function finds the minimum value.

The **Sum** function calculates the sum of its arguments.

The **StdevP** function calculates the standard deviation of its arguments.

The **VarP** function calculates the variance of its arguments.

You can supply the values for these functions as:

- Separate arguments. For example, **Sum(1, 2, 3)** returns 6.
- A table and a formula to operate over that table. The aggregate will be calculated on the values of the formula for each record.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

These functions operate on numeric values only. Other types of values, such as strings or records, are ignored. Use the **Value** function to convert a string into a number.

The **Average**, **Max**, **Min**, and **Sum** functions can be delegated when used with a data source that supports delegation for these functions. However, **StdevP** and **VarP** can't be delegated for any data sources. If delegation is not supported, only the first portion of the data will be retrieved and then the function applied locally. The result may not represent the complete story. A delegation warning will appear at authoring time to remind you of this limitation and to suggest switching to delegable alternatives where possible. For more information, see the delegation overview.

Syntax

Average(*NumericalFormula1*, [*NumericalFormula2*, ...])
Max(*NumericalFormula1*, [*NumericalFormula2*, ...])
Min(*NumericalFormula1*, [*NumericalFormula2*, ...])
Sum(*NumericalFormula1*, [*NumericalFormula2*, ...])
StdevP(*NumericalFormula1*, [*NumericalFormula2*, ...])
VarP(*NumericalFormula1*, [*NumericalFormula2*, ...])

- *NumericalFormula(s)* - Required. Numeric values to operate on.

Average(*Table*, *NumericalFormula*)
Max(*Table*, *NumericalFormula*)
Min(*Table*, *NumericalFormula*)
Sum(*Table*, *NumericalFormula*)
StdevP(*Table*, *NumericalFormula*)
VarP(*Table*, *NumericalFormula*)

- *Table* - Required. Table to operate on.
- *NumericalFormula* - Required. Formula to evaluate for each record. The result of this formula is used for the aggregation. You can use columns of the table in the formula.

Examples

Step by step

Let's say that you had a data source named **Sales** that contained a **CostPerUnit** column and a **UnitsSold** column, and you set the **Text** property of a label to this function:
Sum(Sales, CostPerUnit * UnitsSold)

The label would show total sales by multiplying the values in those columns for each record and then adding the results from all records together:

Units sold	Cost per unit
30	$150
45	$125
20	$175

Total sales:

$13625

As a different example, let's say that you had sliders that were named **Slider1**, **Slider2**, and **Slider3** and a label with its **Text** property set to this formula:

Sum(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the sum of all values to which the sliders were set.

Average(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the average of all values to which the sliders were set.

Max(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the maximum of all values to which the sliders were set.

Min(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the minimum of all values to which the sliders were set.

StdevP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the standard deviation of all values to which the sliders were set.

VarP(Slider1.Value, Slider2.Value, Slider3.Value): The label would show the variance of all values to which the sliders were set.

EditForm, NewForm, SubmitForm, ResetForm, and ViewForm functions

Article • 03/22/2024

Applies to: Canvas apps

View, edit, or create an item, save the contents, and reset the controls in an **Edit form** control.

Overview

These functions change the state of the **Edit form** control. The form control can be in one of these modes:

Expand table

Mode	Description
FormMode.Edit	The form is populated with an existing record and the user can modify the values of the fields. Once complete, the user can save the changes to the record.
FormMode.New	The form is populated with default values and the user can modify the values of the fields. Once complete, the user can add the record to the data source.
FormMode.View	The form is populated with an existing record but the user cannot modify the values of the fields.

Description

These functions are often invoked from the **OnSelect** formula of a **Button** or **Image** control so that the user can save edits, abandon edits, or create a record. You can use controls and these functions together to create a complete solution.

These functions return no values.

You can use these functions only in behavior formulas.

SubmitForm

Use the **SubmitForm** function in the **OnSelect** property of a Button control to save any changes in a Form control to the data source.

Before submitting any changes, this function checks for validation issues with any field that's marked as required or that has one or more constraints on its value. This behavior matches that of the **Validate** function.

SubmitForm also checks the **Valid** property of the Form, which is an aggregation of all the **Valid** properties of the **Card** controls that the Form control contains. If a problem occurs, the data isn't submitted, and the **Error** and **ErrorKind** properties of the Form control are set accordingly.

If validation passes, **SubmitForm** submits the change to the data source.

- If successful, the Form's **OnSuccess** behavior runs, and the **Error** and **ErrorKind** properties are cleared. If the form was in **FormMode.New** mode, it is returned to **FormMode.Edit** mode.
- If unsuccessful, the Form's **OnFailure** behavior runs, and the **Error** and **ErrorKind** properties are set accordingly. The mode of the form is unchanged.

EditForm

The **EditForm** function changes the Form control's mode to **FormMode.Edit**. In this mode, the contents of the Form control's **Item** property are used to populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is changed, not created. **FormMode.Edit** is the default for the Form control.

> ⓘ **Note**
>
> When the form is in edit mode and Item is null, data card properties are not evaluated and will return default values.

NewForm

The **NewForm** function changes the Form control's mode to **FormMode.New**. In this mode, the contents of the Form control's **Item** property are ignored, and the default values of the Form's **DataSource** property populate the form. If the **SubmitForm** function runs when the form is in this mode, a record is created, not changed.

ResetForm

The **ResetForm** function resets the contents of a form to their initial values, before the user made any changes. If the form is in **FormMode.New** mode, the form is reset to **FormMode.Edit** mode. The **OnReset** behavior of the form control also runs. You can also reset individual controls with the **Reset** function but only from within the form.

ViewForm

The **ViewForm** function changes the Form control's mode to **FormMode.View**. In this mode, the contents of the Form control's **Item** property are used to populate the form. The **SubmitForm** and **ResetForm** functions have no effect when in this mode.

DisplayMode Property

The current mode can be read through the **Mode** property. The mode also determines the value of the **DisplayMode** property, which can be used by data cards and controls within the form control. Often, the data card's **DisplayMode** property will be set to **Parent.DisplayMode** (referencing the form) as will the control's **DisplayMode** property (referencing the data card):

Expand table

Mode	DisplayMode	Description
FormMode.Edit	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept changes to a record.
FormMode.New	**DisplayMode.Edit**	Data cards and controls are editable, ready to accept a new record.
FormMode.View	**DisplayMode.View**	Data cards and controls are not editable and optimized for viewing.

Syntax

SubmitForm(*FormName*)

- *FormName* - Required. Form control to submit to the data source.

EditForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.Edit** mode.

NewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.New** mode.

ResetForm(*FormName*)

- *FormName* - Required. Form control to reset to initial values. Also switches the form from **FormMode.New** mode to **FormMode.Edit** mode.

ViewForm(*FormName*)

- *FormName* - Required. Form control to switch to **FormMode.View** mode.

Examples

See Understand data forms for complete examples.

1. Add a Button control, set its **Text** property to show **Save**, and set its **OnSelect** property to this formula:

   SubmitForm(EditForm)

2. Set the **OnFailure** property of a Form control to blank and its **OnSuccess** property to this formula:

   Back()

3. Name a **Label** control **ErrorText**, and set its **Text** property to this formula:

   EditForm.Error

   When the user selects the **Save** button, any changes in the Form control are submitted to the underlying data source.

   - If the submission succeeds, any changes are saved or, if the Form control is in **New** mode, a record is created. **ErrorText** is *blank* and the previous screen reappears.
   - If the submission fails, **ErrorText** shows a user-friendly error message, and the current screen remains visible so that the user can correct the problem and try again.

4. Add a Button control, set its **Text** property to show **Cancel**, and set its **OnSelect** property to this formula:

   ResetForm(EditForm); Back()

   When the user selects the **Cancel** button, the values in the Form control are reset to what they were before the user started to edit it, the previous screen reappears,

and the Form control is returned to **Edit** mode if it was in **New** mode.

5. Add a Button control, set its Text property to show **New**, and set its OnSelect property to this formula:

   NewForm(EditForm); Navigate(EditScreen, None)

   When the user selects the **New** button, the Form control switches to **New** mode, the default values for the Form control's data source populate that control, and the screen that contains the Form control appears. When the **SubmitForm** function runs, a record is created instead of updated.

Day, Month, Year, Hour, Minute, Second, and Weekday functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns individual components of a Date/Time value.

Description

The **Day** function returns the day component of a Date/Time value, ranging from 1 to 31.

The **Month** function returns the month component of a Date/Time value, ranging from 1 to 12.

The **Year** function returns the year component of a Date/Time value, starting with 1900.

The **Hour** function returns the hour component of a Date/Time value, ranging from 0 (12:00 AM) to 23 (11:00 PM).

The **Minute** function returns the minute component of a Date/Time value, ranging from 0 to 59.

The **Second** function returns the second component of a Date/Time value, ranging from 0 to 59.

The **Weekday** function returns the weekday of a Date/Time value. By default, the result ranges from 1 (Sunday) to 7 (Saturday). You can specify a different range with a Microsoft Excel Weekday function code or a StartOfWeek enumeration value:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	StartOfWeek.Sunday	Numbers 1 (Sunday) through 7 (Saturday). Default.
2, 11	StartOfWeek.Monday	Numbers 1 (Monday) through 7 (Sunday).
3	StartOfWeek.MondayZero	Numbers 0 (Monday) through 6 (Sunday).
12	StartOfWeek.Tuesday	Numbers 1 (Tuesday) through 7 (Monday).

Excel code	StartOfWeek enumeration	Description
13	**StartOfWeek.Wednesday**	Numbers 1 (Wednesday) through 7 (Tuesday).
14	**StartOfWeek.Thursday**	Numbers 1 (Thursday) through 7 (Wednesday).
15	**StartOfWeek.Friday**	Numbers 1 (Friday) through 7 (Thursday).
16	**StartOfWeek.Saturday**	Numbers 1 (Saturday) through 7 (Friday).

All functions return a number.

See working with dates and times for more information.

Syntax

Day(*DateTime*)
Month(*DateTime*)
Year(*DateTime*)
Hour(*DateTime*)
Minute(*DateTime*)
Second(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on.

Weekday(*DateTime* [, *WeekdayFirst*])

- *DateTime* - Required. Date/Time value to operate on.
- *WeekdayFirst* - Optional. Excel code specifying which day starts the week. If not supplied, 1 (Sunday first) is used.

Examples

For the following example, the current time is **3:59:37 PM** on **Thursday, April 9, 2015**.

Expand table

Formula	Description	Result
Year(Now())	Returns the year component of the current time and date.	2015
Month(Now())	Returns the month component of the current time and date.	4

Formula	Description	Result
Day(Now())	Returns the day component of the current time and date.	9
Hour(Now())	Returns the hour component of the current time and date.	15
Minute(Now())	Returns the minute component of the current time and date.	59
Second(Now())	Returns the second component of the current time and date.	37
Weekday(Now())	Returns the weekday component of the current time and date, using the default start of the week as Sunday.	5
Weekday(Now(), 14)	Returns the weekday component of the current time and date, using an Excel code to specify the start of the week as Thursday.	1
Weekday(Now(), StartOfWeek.Wednesday)	Returns the weekday component of the current time and date, using a **StartOfWeek** enumeration to specify the start of the week as Wednesday.	2

Feedback

Was this page helpful? Yes No

Provide product feedback

WeekNum and ISOWeekNum functions

Article • 04/20/2025

Expand table

Function	Applies to
ISOWeekNum	✅ Canvas apps ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Pages
WeekNum	✅ Canvas apps ✅ Cards ✅ Copilot Studio ✅ Desktop flows ✅ Dataverse formula columns ✅ Model-driven apps ✅ Power Platform CLI ✅ Dataverse functions ✅ Power Pages

Returns the week number of a specific date.

Description

Use the **WeekNum** and **ISOWeekNum** functions to determine the week number of a date.

These functions differ in how they determine the first week of the year (week 1):

- **WeekNum** uses the week containing January 1 as the first week of the year. The result from this function can range from 1 to 54.
- **ISOWeekNum** uses the week containing the first Thursday of the year as the first week of the year. This follows the ISO 8601 date and time standard definition for week numbering. The result from this function can range from 1 to 53. It is possible that 52 or 53 may be returned for the first days of January since the dates could belong to the last week of the previous year.

Use the second parameter to **WeekNum** to specify which day begins a week. You can provide either an Excel code number or use the StartOfWeek enumeration:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	**StartOfWeek.Sunday**	Week begins on Sunday. Default.
2, 11	**StartOfWeek.Monday**	Week begins on Monday.
12	**StartOfWeek.Tuesday**	Week begins on Tuesday.
13	**StartOfWeek.Wednesday**	Week begins on Wednesday.

Excel code	StartOfWeek enumeration	Description
14	**StartOfWeek.Thursday**	Week begins on Thursday.
15	**StartOfWeek.Friday**	Week begins on Friday.
16	**StartOfWeek.Saturday**	Week begins on Saturday.

ISOWeekNum always uses Monday as the start of the week. In Excel, the **WeekNum** function supports an addition code **21** that is not supported here; use **ISOWeekNum** instead.

If you pass a single number to these functions, the return value is a single result. If you pass a single-column table that contains numbers, the return value is a single-column table of results, one result for each record in the argument's table. If you have a multi-column table, you can shape it into a single-column table, as working with tables describes.

Syntax

WeekNum(*DateTime* [, *StartOfWeek*])

- *DateTime* - Required. Date/Time value to operate on.
- *StartOfWeek* - Optional. Excel code or StartOfWeek enumeration that determines which day the week begins.

ISOWeekNum(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on. The week always begins on Monday.

Examples

First and last calendar weeks of 2021

Expand table

Date	WeekNum(Date)	ISOWeekNum(Date)	WeekNum(Date, StartOfWeek.Wednesday)
Friday, January 1, 2021	1	53	1
Saturday, January 2, 2021	1	53	1
Sunday, January 3, 2021	2	53	1
Monday, January 4, 2021	2	1	1
Tuesday, January 5, 2021	2	1	1

Date	WeekNum(Date)	ISOWeekNum(Date)	WeekNum(Date, StartOfWeek.Wednesday)
Wednesday, January 6, 2021	2	1	2
Thursday, January 7, 2021	2	1	2
Saturday, December 25, 2021	52	51	52
Sunday, December 26, 2021	53	51	52
Monday, December 27, 2021	53	52	52
Tuesday, December 28, 2021	53	52	52
Wednesday, December 29, 2021	53	52	53
Thursday, December 30, 2021	53	52	53
Friday, December 31, 2021	53	52	53

With function

Article • 06/10/2024

Applies to: ✅ Canvas apps ✅ Desktop flows ✅ Model-driven apps ✅ Power Pages ✅ Power Platform CLI

Calculates values and performs actions for a single record, including inline records of named values.

Description

The **With** function evaluates a formula for a single record. The formula can calculate a value and/or perform actions, such as modifying data or working with a connection. Use the **ForAll** function to evaluate a formula for all the records in a table of records.

Fields of the record currently being processed are available within the formula. Use the **ThisRecord** operator or simply reference fields by name as you would any other value. The **As** operator can also be used to name the record being processed which can help make your formula easier to understand and make nested records accessible. For more information, see the examples below and working with record scope.

Use **With** to improve the readability of complex formulas by dividing it into smaller named sub-formulas. These named values act like simple local variables confined to the scope of the **With**. The same inline record syntax that is used with the **UpdateContext** function can be used with **With**. Using **With** is preferred over context or global variables as it is self contained, easy to understand, and can be used in any declarative formula context.

Use **With** to access the fields of the record that are returned by functions such as **Patch** or **Match**. **With** holds the value from these functions long enough to be used in further calculations or actions.

If the *Record* argument to **With** is an error, that error will be returned by the function and the *Formula* will not be evaluated.

Syntax

With(*Record*, *Formula*)

- *Record* – Required. The record to be acted upon. For names values, use the inline syntax `{ name1: value1, name2: value2, ... }`

- *Formula* – Required. The formula to evaluate for *Record*. The formula can reference any of the fields of *Record* directly as a record scope.

Examples

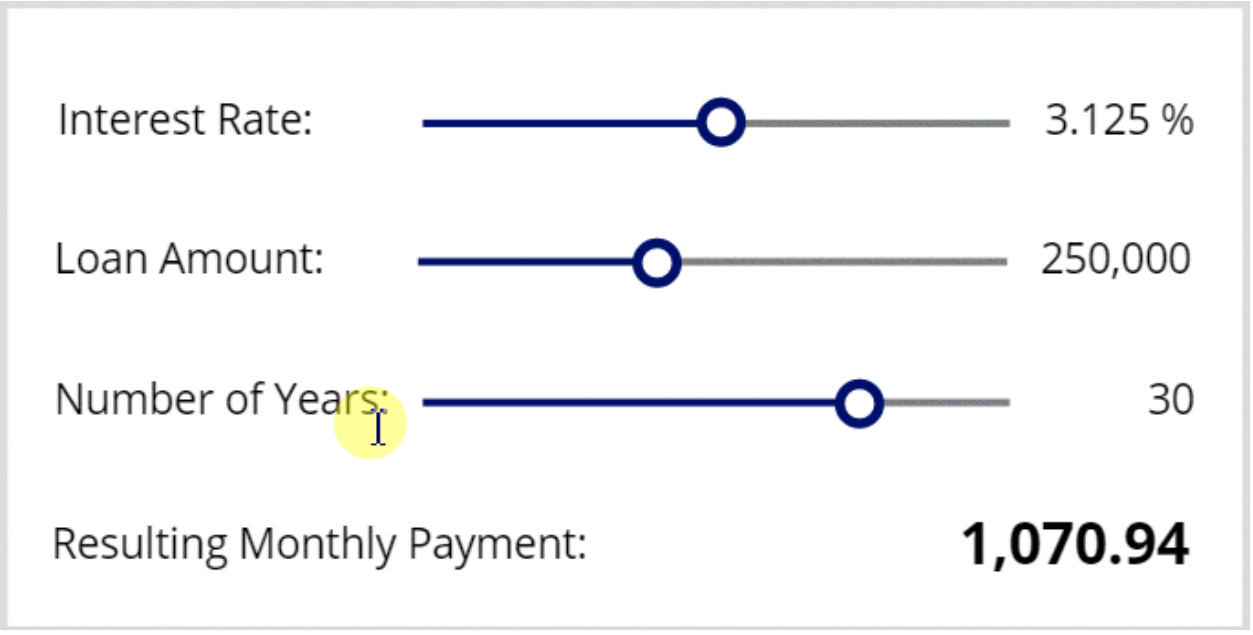
Simple named values

Power Apps

```
With( { radius: 10,
        height: 15 },
    Pi() * (radius*radius) * height
)
// Result: 4712.38898038 (as shown in a label control)
```

This example uses a record of named values to calculate the volume of a cylinder. **With** is being used to capture all the input values together, making it easy to separate them from the calculation itself.

Nested With



Power Apps

```
With( { AnnualRate: RateSlider/8/100,        // slider moves in 1/8th
increments and convert to decimal
        Amount: AmountSlider*10000,          // slider moves by 10,000
increment
        Years: YearsSlider,                  // slider moves in single year
increments, no adjustment required
```

```
        AnnualPayments: 12 },                // number of payments per year
      With( { r: AnnualRate/AnnualPayments,  // interest rate
              P: Amount,                     // loan amount
              n: Years*AnnualPayments },     // number of payments
            r*P / (1 - (1+r)^-n)             // standard interest
calculation
      )
)
```

This example nests **With** functions to create a two-tier calculation for monthly mortgage payments. As long as there is no conflict, all of the outer **With** named values are available within the inner **With**.

Since the slider controls can only move in increments of 1, the sliders are divided or multiplied to create effectively a custom increment. In the case of the interest rate, the **RateSlider** has its **Max** property set to **48**, divided by 8 for a 1/8 percentage point increment and divided by 100 to convert from a percentage to a decimal, covering the range 0.125% to 6%. In the case of of the loan amount, the **AmountSlider** has its **Max** property set to **60** and multiplied by 10,000, covering the range 10,000 to 600,000.

The **With** is automatically recalculated as the sliders move and the new loan payment displayed. No variables are used and there is no need to use the **OnChange** property of the slider controls.

Here are the detailed instructions for creating this app:

1. Create a new app.
2. Add a **Slider** control and name it **RateSlider**. Set its **Max** property to 48.
3. Add a **Label** control to the left of the slider control. Set its **Text** property to **"Interest Rate:"**.
4. Add a **Label** control to the right of the slider control. Set its **Text** property to the formula **RateSlider/8 & " %"**.
5. Add another **Slider** control and name it **AmountSlider**. Set its **Max** property to 60.
6. Add a **Label** control to the left of this slider control. Set its **Text** property to **"Loan Amount:"**.
7. Add a **Label** control to the right of this slider control. Set its **Text** property to the formula **AmountSlider/8 * 10000**.
8. Add another **Slider** control and name it **YearsSlider**. Set its **Max** property to 40.
9. Add a **Label** control to the left of this slider control. Set its **Text** property to **"Number of Years:"**.
10. Add a **Label** control to the right of this slider control. Set its **Text** property to the formula **YearsSlider**.
11. Add a **Label** control and set its **Text** property to the formula shown above.

12. Add a **Label** control to the left of the last label control. Set its **Text** property to "**Recurring Monthly Payment:**".

Primary key returned from Patch

Power Apps

```
With( Patch( Orders, Defaults( Orders ), { OrderStatus: "New" } ),
      ForAll( NewOrderDetails,
              Patch( OrderDetails, Defaults( OrderDetails ),
                     { Order: OrderID,          // from With's first
argument, primary key of Patch result
                       Quantity: Quantity,      // from ForAll's
NewOrderDetails table
                       ProductID: ProductID }   // from ForAll's
NewOrderDetails table
              )
      )
)
```

This example adds a record to the **Order** table in SQL Server. It then uses the returned primary key for the order, returned by the **Patch** function in the **OrderID** field, to create related records in the **OrderDetails** table.

Extracted values with a regular expression

Power Apps

```
With(
    Match( "PT2H1M39S", "PT(?:(?<hours>\d+)H)?(?:(?<minutes>\d+)M)?(?:(?
<seconds>\d+)S)?" ),
    Time( Value( hours ), Value( minutes ), Value( seconds ) )
)
// Result: 2:01 AM (as shown in a label control, use the Text function to
see the seconds)
```

This example extracts the hours, minutes, and seconds from an ISO 8601 duration value and then uses these sub-matches to create a Date/Time value.

Note that although the sub-matches contain numbers they are still in a text string. Use the **Value** function to convert to a number before performing mathematical operations.

Map a record in a component

See Map record.

Feedback

Was this page helpful? Yes No

Provide product feedback

Day, Month, Year, Hour, Minute, Second, and Weekday functions

Article • 06/10/2024

Applies to: Canvas apps Dataverse formula columns Desktop flows Model-driven apps Power Pages Power Platform CLI

Returns individual components of a Date/Time value.

Description

The **Day** function returns the day component of a Date/Time value, ranging from 1 to 31.

The **Month** function returns the month component of a Date/Time value, ranging from 1 to 12.

The **Year** function returns the year component of a Date/Time value, starting with 1900.

The **Hour** function returns the hour component of a Date/Time value, ranging from 0 (12:00 AM) to 23 (11:00 PM).

The **Minute** function returns the minute component of a Date/Time value, ranging from 0 to 59.

The **Second** function returns the second component of a Date/Time value, ranging from 0 to 59.

The **Weekday** function returns the weekday of a Date/Time value. By default, the result ranges from 1 (Sunday) to 7 (Saturday). You can specify a different range with a Microsoft Excel Weekday function code or a StartOfWeek enumeration value:

Expand table

Excel code	StartOfWeek enumeration	Description
1, 17	StartOfWeek.Sunday	Numbers 1 (Sunday) through 7 (Saturday). Default.
2, 11	StartOfWeek.Monday	Numbers 1 (Monday) through 7 (Sunday).
3	StartOfWeek.MondayZero	Numbers 0 (Monday) through 6 (Sunday).
12	StartOfWeek.Tuesday	Numbers 1 (Tuesday) through 7 (Monday).

Excel code	StartOfWeek enumeration	Description
13	**StartOfWeek.Wednesday**	Numbers 1 (Wednesday) through 7 (Tuesday).
14	**StartOfWeek.Thursday**	Numbers 1 (Thursday) through 7 (Wednesday).
15	**StartOfWeek.Friday**	Numbers 1 (Friday) through 7 (Thursday).
16	**StartOfWeek.Saturday**	Numbers 1 (Saturday) through 7 (Friday).

All functions return a number.

See working with dates and times for more information.

Syntax

Day(*DateTime*)
Month(*DateTime*)
Year(*DateTime*)
Hour(*DateTime*)
Minute(*DateTime*)
Second(*DateTime*)

- *DateTime* - Required. Date/Time value to operate on.

Weekday(*DateTime* [, *WeekdayFirst*])

- *DateTime* - Required. Date/Time value to operate on.
- *WeekdayFirst* - Optional. Excel code specifying which day starts the week. If not supplied, 1 (Sunday first) is used.

Examples

For the following example, the current time is **3:59:37 PM** on **Thursday, April 9, 2015**.

Expand table

Formula	Description	Result
Year(Now())	Returns the year component of the current time and date.	2015
Month(Now())	Returns the month component of the current time and date.	4

Formula	Description	Result
Day(Now())	Returns the day component of the current time and date.	9
Hour(Now())	Returns the hour component of the current time and date.	15
Minute(Now())	Returns the minute component of the current time and date.	59
Second(Now())	Returns the second component of the current time and date.	37
Weekday(Now())	Returns the weekday component of the current time and date, using the default start of the week as Sunday.	5
Weekday(Now(), 14)	Returns the weekday component of the current time and date, using an Excel code to specify the start of the week as Thursday.	1
Weekday(Now(), StartOfWeek.Wednesday)	Returns the weekday component of the current time and date, using a **StartOfWeek** enumeration to specify the start of the week as Wednesday.	2

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a canvas app from within a solution

Article • 12/16/2022

Create an app from within a solution if, for example, you want to deploy the app to a different environment. Solutions can contain not only apps but also customized tables, choices, and other components. You can quickly customize an environment in a variety of ways by creating apps and other components from within a solution, exporting the solution, and then importing it into another environment.

For more information about solutions, see Solutions overview.

Prerequisite

To follow the steps in this topic, you must switch to an environment that contains a Microsoft Dataverse database.

> **Note**
>
> To use or run a flow in a canvas app that is included in a **solution**, you must be assigned a Dataverse security role with the Read privilege on the **Process (Workflow)** table. For more information about security roles and privileges in Dataverse, see **Security roles and privileges**.

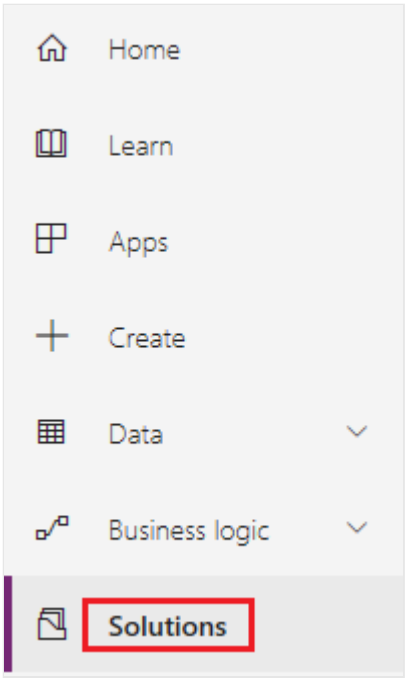
Create a solution

You can skip this procedure if you already have a solution in which you want to create an app or to which you want to link an app.

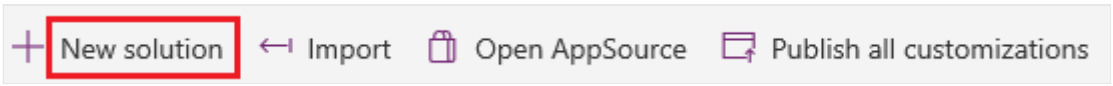
1. Sign in to Power Apps, and then (if necessary) switch to the appropriate environment:

   - If you want to create an app from within a solution, switch to any environment that contains a Dataverse database.
   - If you want to link an existing app to a solution, switch to the environment that contains that app.

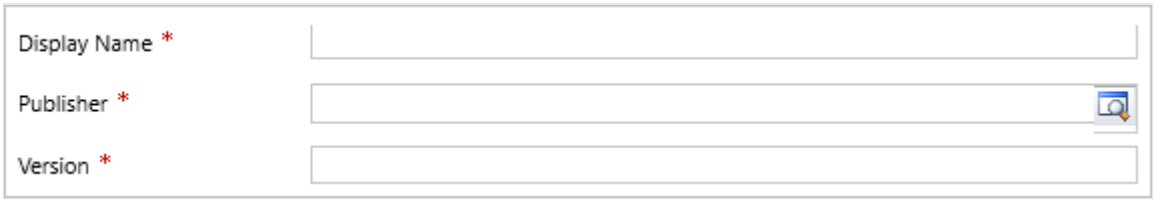
2. In the left navigation bar, select **Solutions**.



3. In the banner under the title bar, select **New solution**.

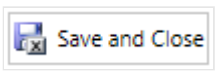


4. In the window that appears, specify a display name, a publisher, and a version for your solution.



A name (with no spaces) will be generated automatically based on the display name that you specify, but you can customize the generated name if you want. You can specify the default publisher for your environment and **1.0** for the version if you don't have specific needs in those areas.

5. Near the upper-left corner, select **Save and Close**.



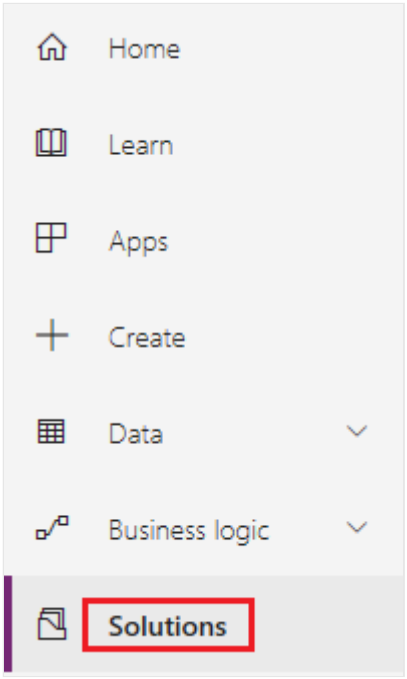
Create a canvas app in a solution

You can create a blank canvas app from within a solution. You can't automatically generate a three-screen app or customize a template or sample app from within a solution.

1. Sign in to Power Apps.

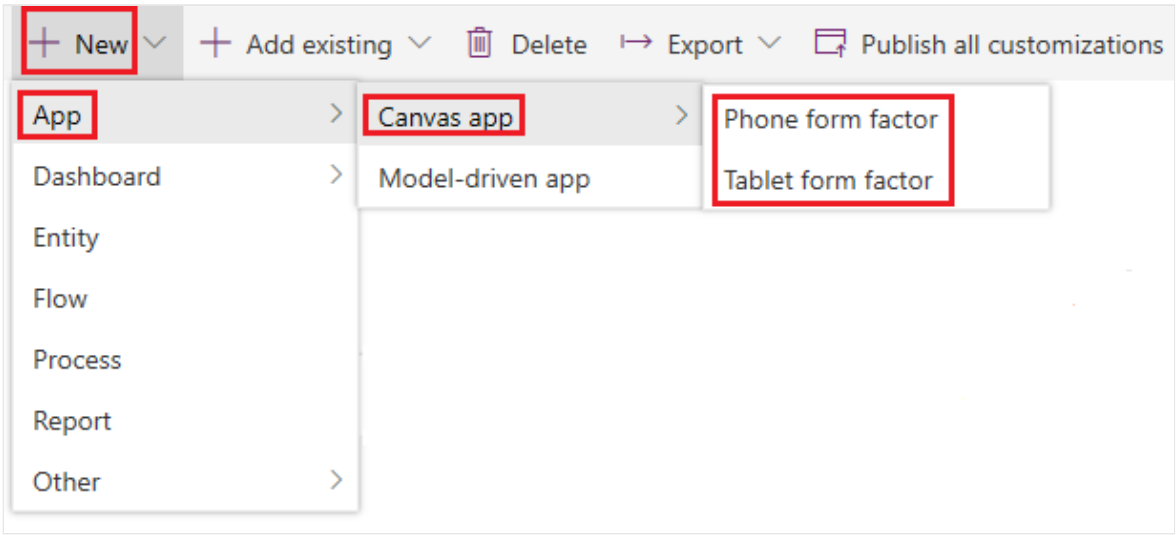
2. If necessary, switch to the environment that contains the solution in which you want to create a canvas app.

3. In the left navigation bar, select **Solutions**.



4. In the list of solutions, select the solution in which you want to create a canvas app.

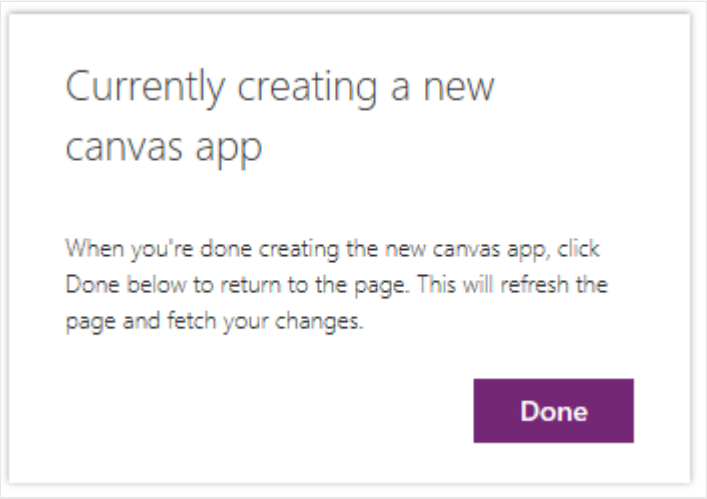
5. In the banner under the title bar, select **New** > **App** > **Canvas app**, and then select the form factor (phone or tablet) of the app that you want to create.



   Power Apps Studio opens with a blank canvas in another browser tab.

6. Create your app (or make at least one change), and then save your changes.

7. On the browser tab where you selected your solution, select **Done** to refresh the list of components in your solution.

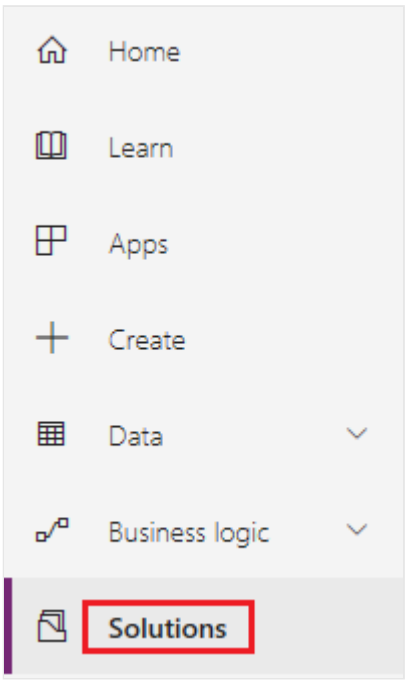


Your new app appears in the list of components for that solution. If you save any changes to your app, they will be reflected in the version that's in the solution.

Add an existing canvas app to a solution

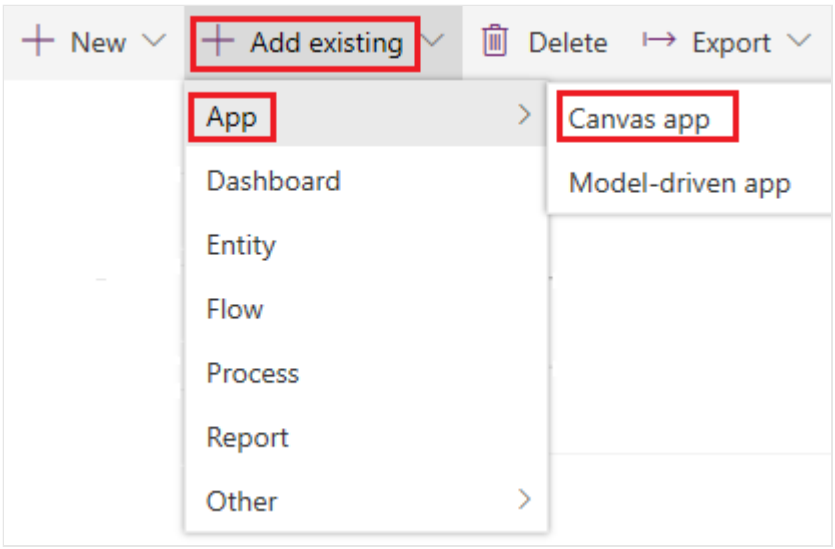
If you want to add an app to a solution, both must be in the same environment, and the app must have been created from within a solution.

1. Sign in to Power Apps.
2. If necessary, switch to the environment that contains the solution to which you want to link an app.
3. In the left navigation bar, select **Solutions**.



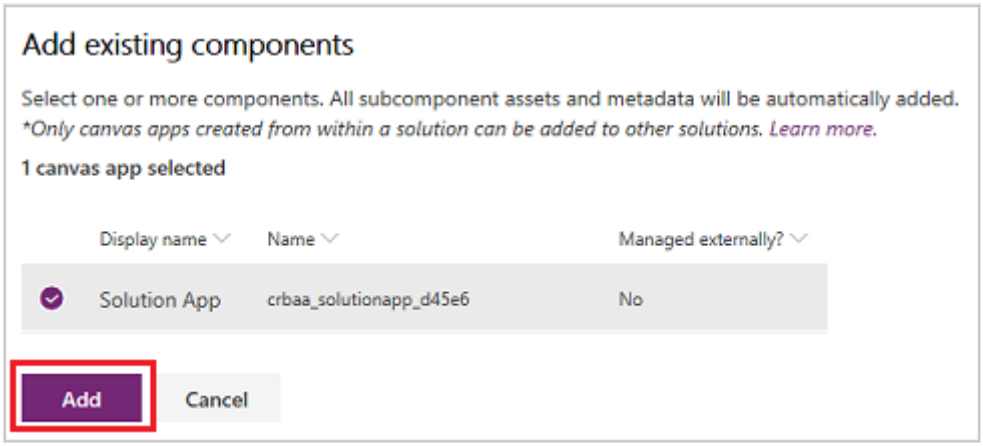
4. In the list of solutions, select the solution to which you want to link an app.

5. In the banner under the title bar, select **Add existing** > **App** > **Canvas app**.



   A list of canvas apps that were created within a solution in this environment appears.

6. In the list of apps, select the app that you want to link to the solution, and then select **Add**.



Known limitations

For information about known limitations, see Solutions overview.

Next steps

- Create or link more apps and other components, such as tables, flows, and dashboards, to your solution.
- Export your solution so that you can deploy it to another environment, on AppSource, and so forth.

See also

Add canvas apps to solution by default (preview)

Add canvas apps and cloud flows to a solution by default

You can create a canvas app in a solution in Power Apps or create a cloud flow in a solution in Power Automate. Creating a canvas app or a cloud flow in a solution previously required manual steps. Now you can create them in a solution by default.

Because canvas apps and cloud flows in a solution are defined in Microsoft Dataverse, you can use Dataverse capabilities to manage them, including connection references, environment variables, the Dataverse API, role-based security, and solution-based application lifecycle management (ALM).

> **Important**
>
> - The canvas apps setting is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Prerequisites

- Solutions are stored in Dataverse, so the environment must have a Dataverse database to use this feature. If yours doesn't, add a Dataverse database.
- To create canvas apps in solutions, you must have **write** privileges to the CanvasApp table. To create cloud flows in solutions, you should have the Environment Maker role. Learn more in Security roles and privileges.
- A canvas app or cloud flow in a solution must be shared with you before you can view or edit it.

> **Important**
>
> Before you enable the creation of canvas apps and cloud flows in solutions by default, review the Considerations section in this article.

Supported creation methods and entry points

Several methods and entry points are available for creating canvas apps and cloud flows.

Canvas apps creation methods and entry points

The following table summarizes the methods that allow you to create canvas apps in a solution by default.

Expand table

Method	Available for canvas apps in a solution by default?
Create a canvas app from Dataverse	Yes
Create a canvas app from other data sources, such as SharePoint, Excel, Azure Blob Storage, and others	Yes
Create a blank canvas app **NOTE**: Power Apps > **Apps** > **New app** > **Canvas** also creates a blank canvas app.	Yes
Create a canvas app from a template	No
Create a canvas app by customizing SharePoint forms	No

Cloud flows creation methods and entry points

The following table summarizes the methods that allow you to create cloud flows in a solution by default.

Expand table

Method	Available for cloud flows in a solution by default?
Create a cloud flow from blank in a portal	Yes
Create a cloud flow from a template in a portal	Yes
Create a cloud flow from Power Automate Management connector	Yes
Create a cloud flow from PowerShell	Yes
Create a cloud flow from Power Apps	Yes
Create a cloud flow from Teams	Yes
Create a cloud flow from OneDrive	Yes

Method	Available for cloud flows in a solution by default?
Create a cloud flow from Dynamics 365	Yes
Create a cloud flow from the Excel add-in	Yes
Create a cloud flow from the Power Automate mobile app	Yes
Create a cloud flow from SharePoint	Yes
Create a cloud flow from Project Roadmap	Yes

> **Important**
>
> Unsupported entry points will fail to create any cloud flow until that support is added to create a solution cloud flow with that entry point. You must first create a cloud flow in a solution using a supported method.

Manage feature settings for Cloud flows and Canvas apps in Dataverse solutions

You can control whether makers can create new canvas apps and cloud flows directly in Dataverse solutions.

Existing environments

1. Sign in to the Power Platform admin center and select an environment.
2. Go to **Settings** > **Product** > **Features**.
3. Under **Create new canvas apps and cloud flows in Dataverse solutions**, turn on **Canvas apps (Preview)** or **Cloud flows** as desired.

New environments

- All new environments provisioned with a Dataverse database have the **Cloud flows** setting enabled by default.
- The **Canvas apps (Preview)** setting is also enabled by default, though it remains in preview.
- If you prefer not to use these features, you must manually disable them in the environment settings.

Learn more about managing feature settings in Manage feature settings.

View canvas apps and cloud flows in solutions

To view canvas app and cloud flow objects in a solution, use the solution view. You must have access to the object added inside a solution to view it in a solution view.

Export and import using solutions

Once the feature is enabled, use solutions export and import instead of exporting and importing legacy packages of canvas apps and flows. Legacy packages can be imported in environments with the environment setting turned off.

Move flows into a solution with PowerShell

Use the Add-AdminFlowsToSolution cmdlet to move non-solution Power Automate flows into a solution for better lifecycle management and deployment. This command lets you migrate multiple flows by specifying environment and solution IDs, and targeting flows by name or ID. For more information, see Microsoft.PowerApps.Administration.PowerShell Module.

Considerations

Take the following considerations into account before you decide to create canvas apps and cloud flows in a solution by default.

- Your Dataverse environment capacity consumption and related cost might increase.
- Known solution limitations related to canvas apps still apply to canvas apps created in solutions by default.
- This feature automatically creates environment variables when you add data sources for your apps.
- By default, this feature saves all canvas apps to the default solution, **Common Data Services Default Solution**, published by **Microsoft Dataverse Default Publisher**. However, we recommend that you save your canvas apps to a different solution. Learn more about adding existing canvas apps to solutions.
- When you turn the feature on or off for canvas apps and cloud flows in the Power Platform admin center, the Dataverse properties `enablecanvasappsinsolutionsbydefault` and `enableFlowsInSolutionByDefault` in the organization table are updated. You can view the value of these properties using the following sample ODATA snippet: `[org URI]/api/data/v9.0/organizations`. Learn more in Organization table reference.
- If you're planning to change the prefix for the default publisher or solution for your environment, you can learn more in Solution publisher prefix, Create solution publisher

prefix and Change solution publisher prefix.
- Non-solution canvas apps and non-solution cloud flows can be added into a solution to add them into Dataverse, but there's no way to revert back.

Improvements

The following scenarios were improved to support cloud flows and canvas apps defined in Dataverse:

- The **Monitor** > **Cloud flow activity** page now supports solution cloud flows.
- The List My Flows API doesn't return any solution cloud flows.
- Audit log events for cloud flow permissions that provide visibility into sharing now include solution cloud flows.
- The List Flows as Admin API now returns solution cloud flows that weren't previously turned on (published). This API now returns all non-solution and solution cloud flows.
- Flows with delegated authentication to Roadmap can be added into a solution and migrated to Dataverse.
- Flows with delegated authentication to SharePoint can be added into a solution and migrated to Dataverse.

Is this feature generally available or preview?

As of October 29, 2024, the cloud flows setting is generally available. The canvas apps setting remains in preview.

The cloud flows setting continues to be off by default and optional.

Does putting flows into Dataverse use Dataverse capacity?

Moderately. Flow definitions use a small amount of Dataverse storage. A large flow definition could reach 10 KB. With this worst case scenario: 10,000 flows x 10 KB = 100 MB or 0.1 GB. The small increase in Dataverse storage used provides significant benefits, since solution cloud flows are needed to leverage capabilities including application lifecycle management (ALM), governance with Dataverse, automation center, drafts and versioning, and expanded security capabilities.

Related information

- Solutions overview

- Application lifecycle management (ALM) guide

Last updated on 11/07/2025

Install Northwind Traders database and apps

Article • 03/11/2023

By following the steps in this series of topics, you can discover concepts about relational data as implemented in a sample database in Microsoft Dataverse. You can also explore sample business apps, both canvas and model-driven, for managing that data and earn practical experience by creating such an app. This first topic explains how to install the Northwind Traders database in your own environment and gain access to the sample apps, which you can open for editing to reveal how they were built.

Northwind Traders is a fictitious organization that manages orders, products, customers, suppliers, and many other aspects of a small business. This sample appeared with the first versions of Microsoft Access and is still available as an Access template.

Prerequisites

- A Power Apps license that supports Dataverse. You can use a free trial license for 30 days.
- An environment with a Dataverse database. You can create such an environment if you have appropriate permissions.

Download the solution

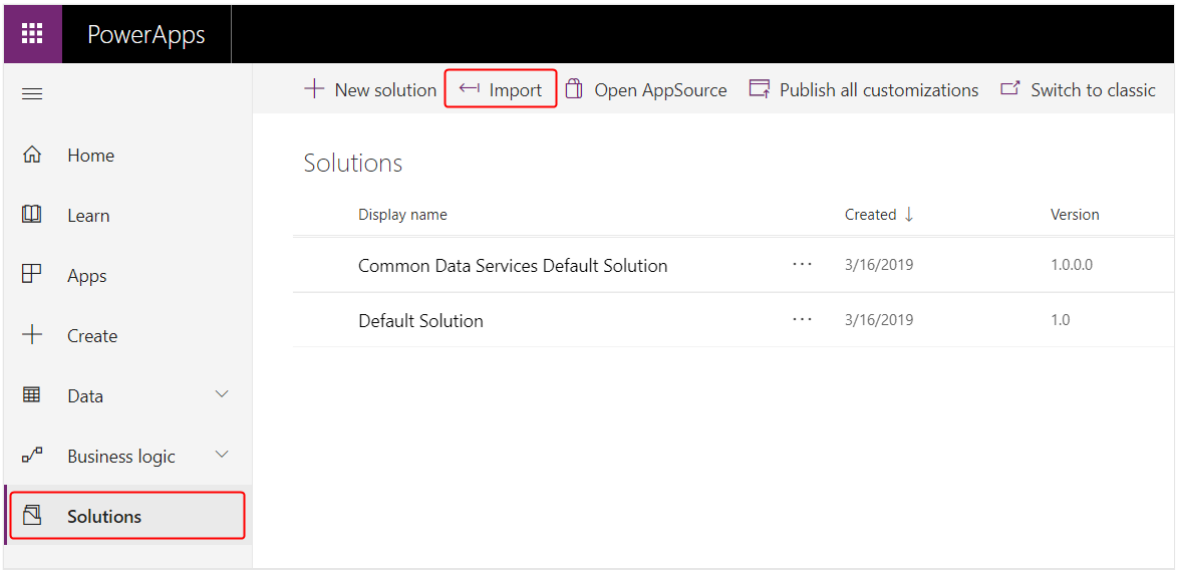
Download the Northwind Traders Solution File

This solution file (.zip) contains the definitions of tables, choices, and business processes; the canvas and model-driven apps; and any other pieces that are used together.

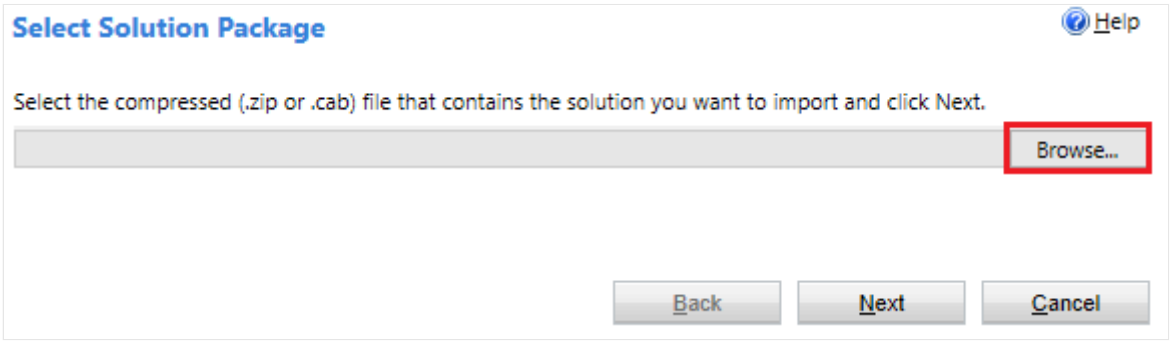
Install the solution

1. Sign in to Power Apps, and then ensure that you're working in an environment that contains a Dataverse database.

2. In the left navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want. Then select **Import** in the toolbar near the top of the screen:



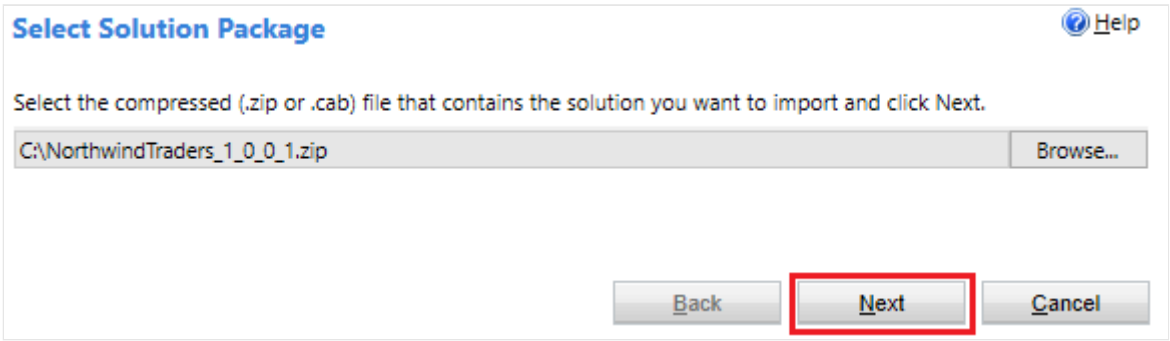
3. In the **Select Solution Package** page, select **Browse**.



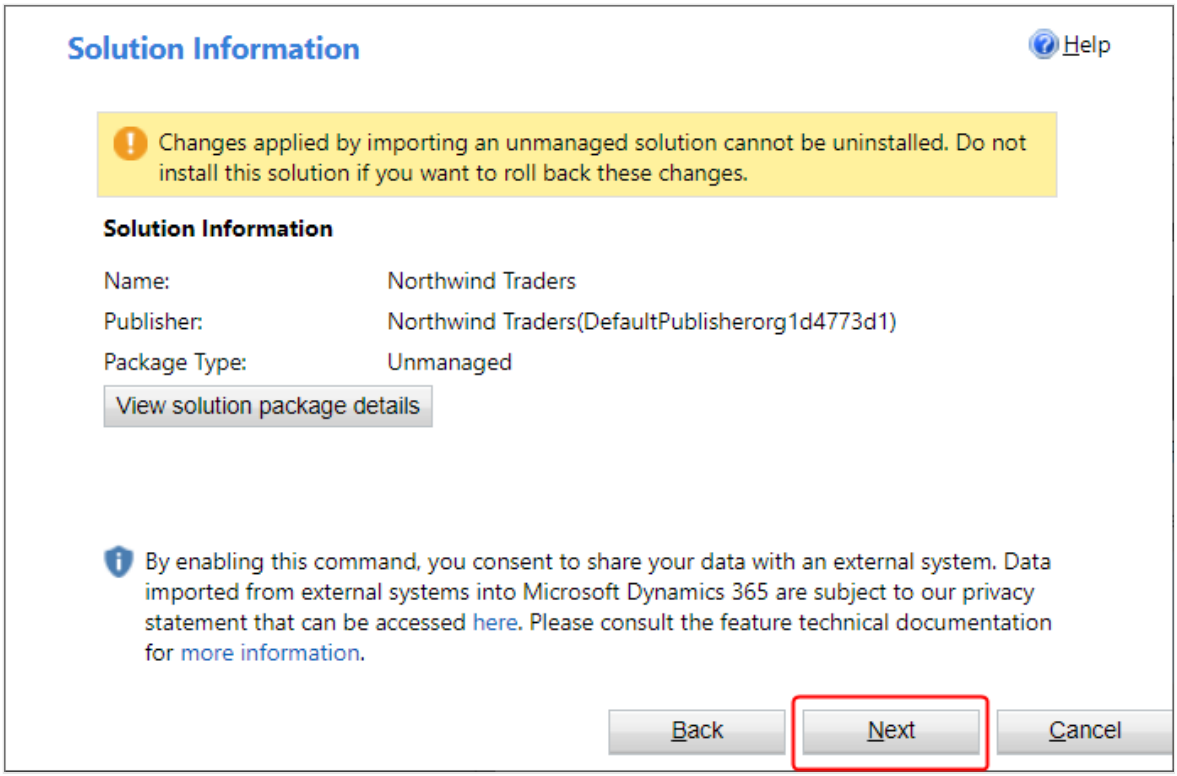
4. Find the file that you downloaded, and then select **Open**.

   Unless you selected a different location, the file will be in your Downloads folder.

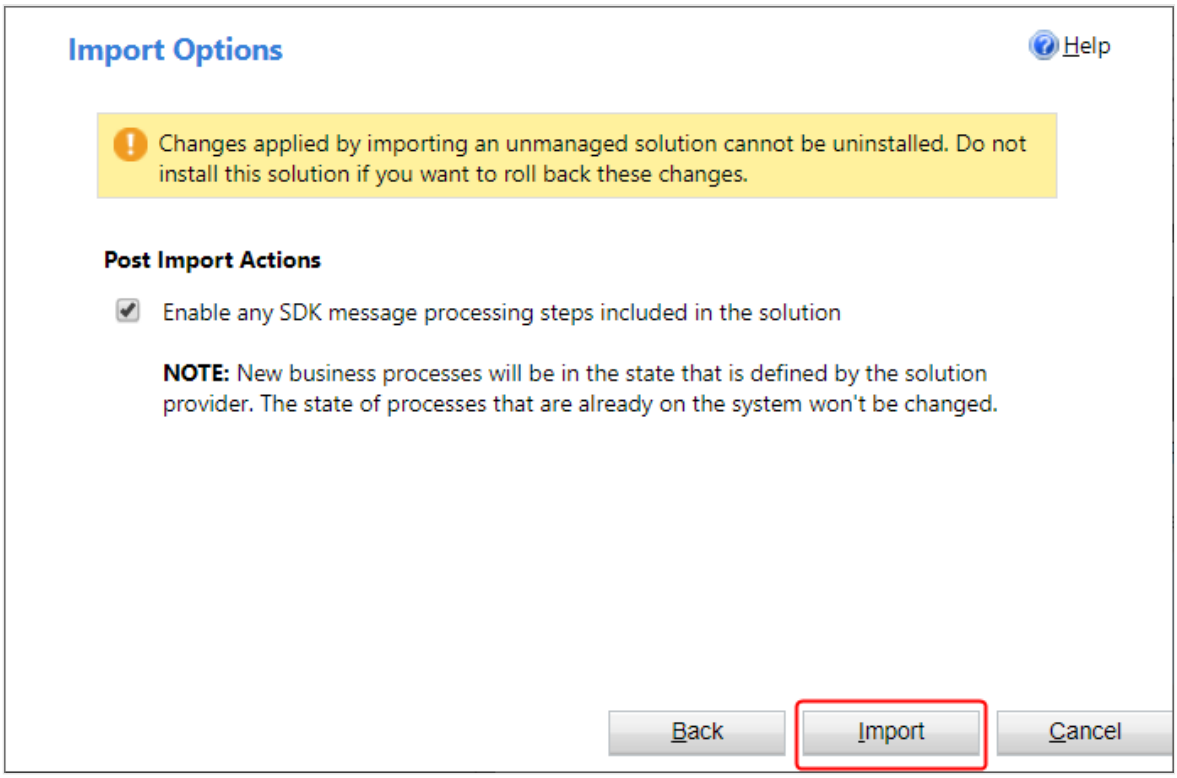
5. If you have the correct file (the version number might vary), select **Next**:



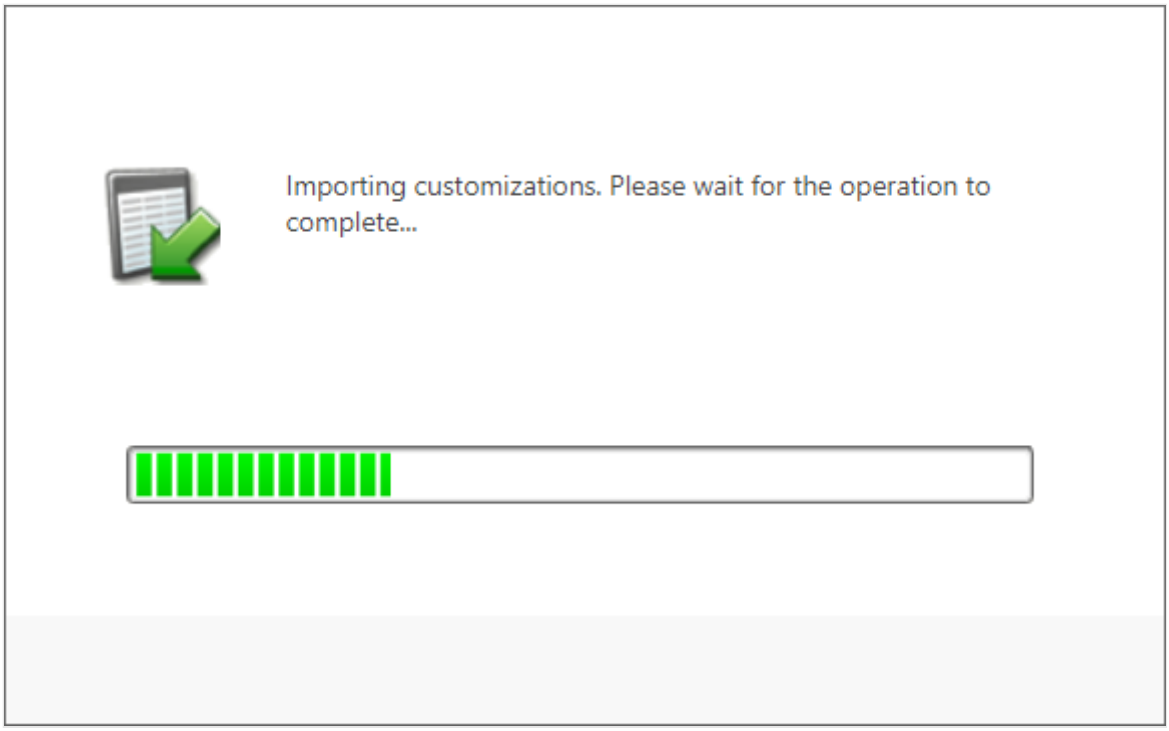
6. In the **Solution Information** page, select **Next** if the name of the solution and the publisher are correct.



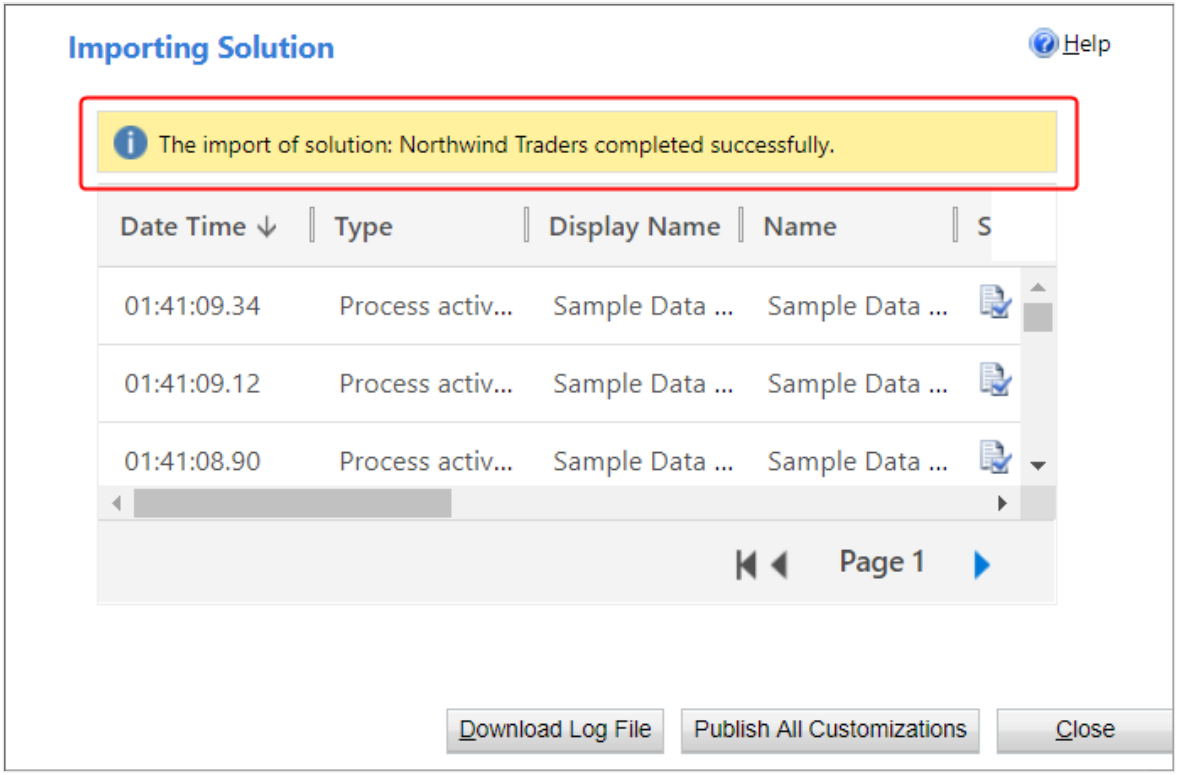
7. In the **Import Options** page, select **Import** to confirm SDK message handling, which the sample requires:



Another page appears and shows progress as the solution is installed over the next few minutes:



When the installation finishes, the original page shows the result:

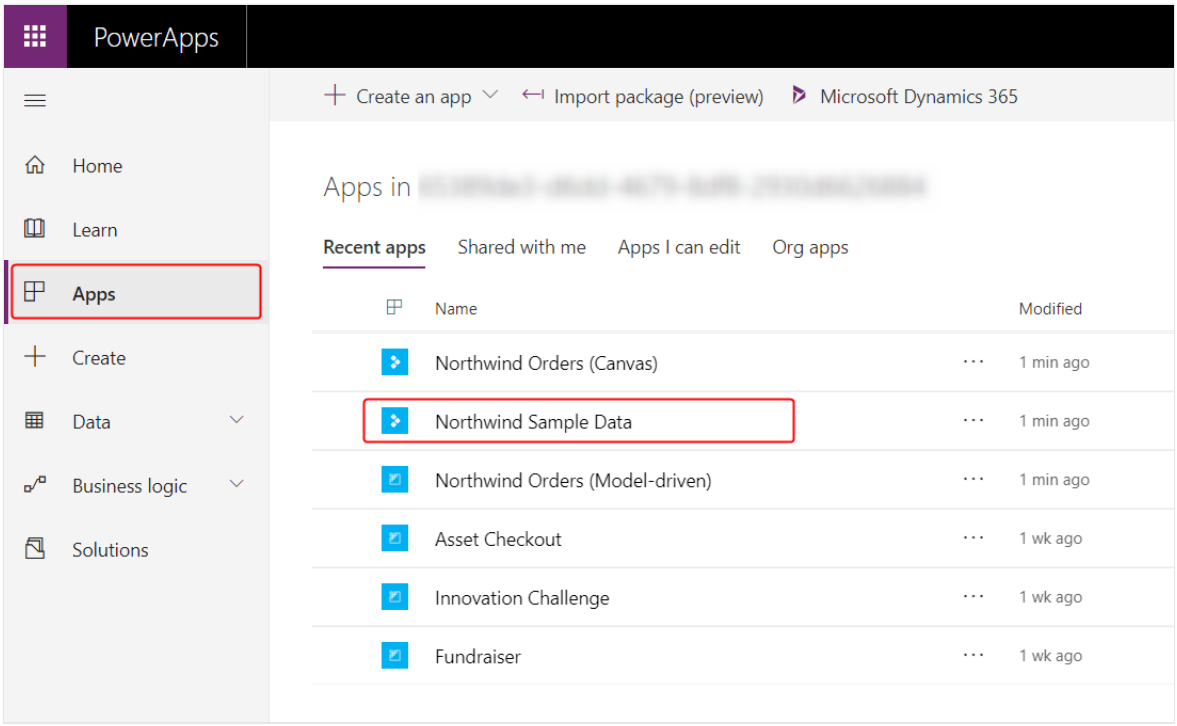


8. Select **Close**.

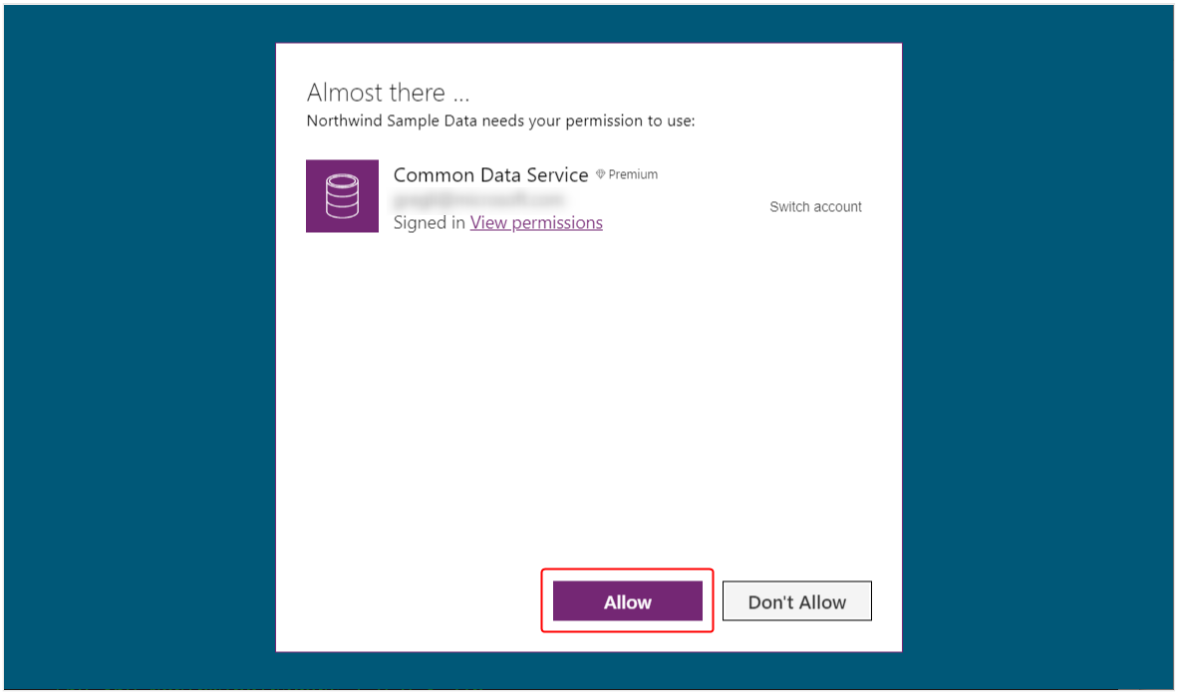
Load the sample data

1. Select **Apps**, and then select **Northwind Sample Data**.

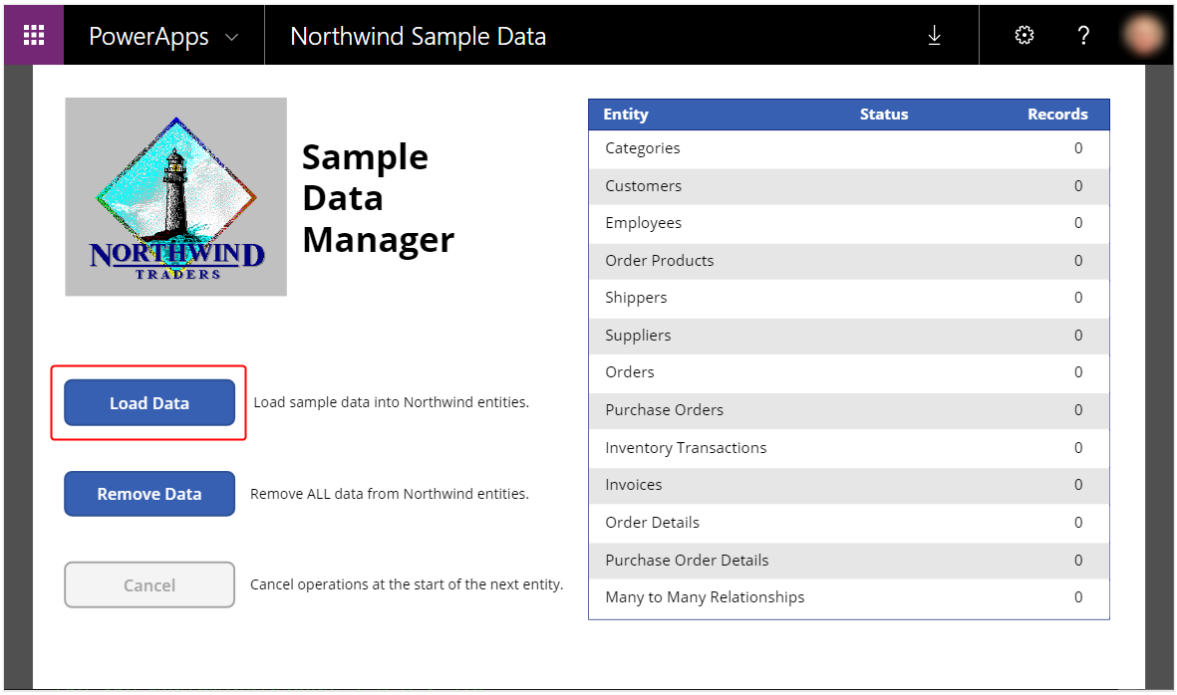
   Wait a few minutes if the Northwind apps don't appear immediately after you install the solution:



2. When the app asks for permission to interact with Dataverse, select **Allow**:



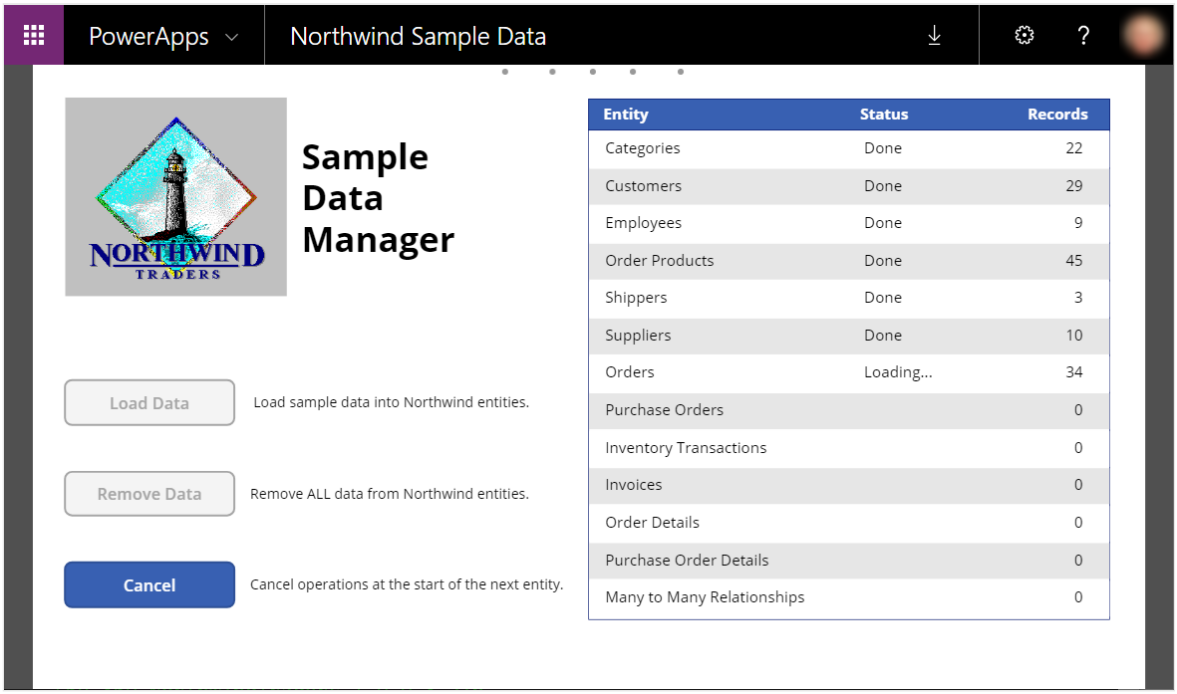
3. After the app loads and shows that the sample tables contain no records, select **Load Data** to populate the tables:



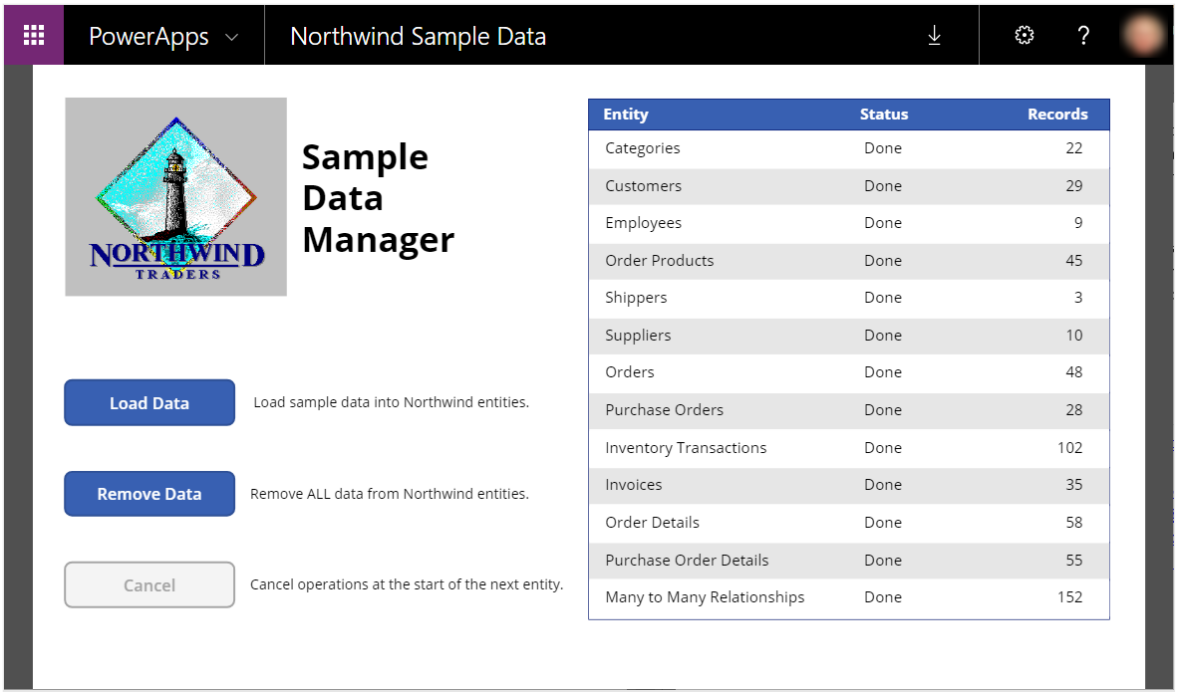
As the app loads the data, dots march across the top of the app, and the number of records increases.

Tables are loaded in a specific order so that relationships can be established between records. For example, the **Order Details** table has a many-to-one relationship with the **Orders** and **Order Products** tables, which are loaded first.

You can cancel the process at any time by selecting **Cancel**, and you can remove the data at any time by selecting **Remove Data**:



When the data finishes loading, the last row (**Many to Many Relationships**) shows **Done**, and the **Load Data** and **Remove Data** buttons are enabled again:



Sample apps

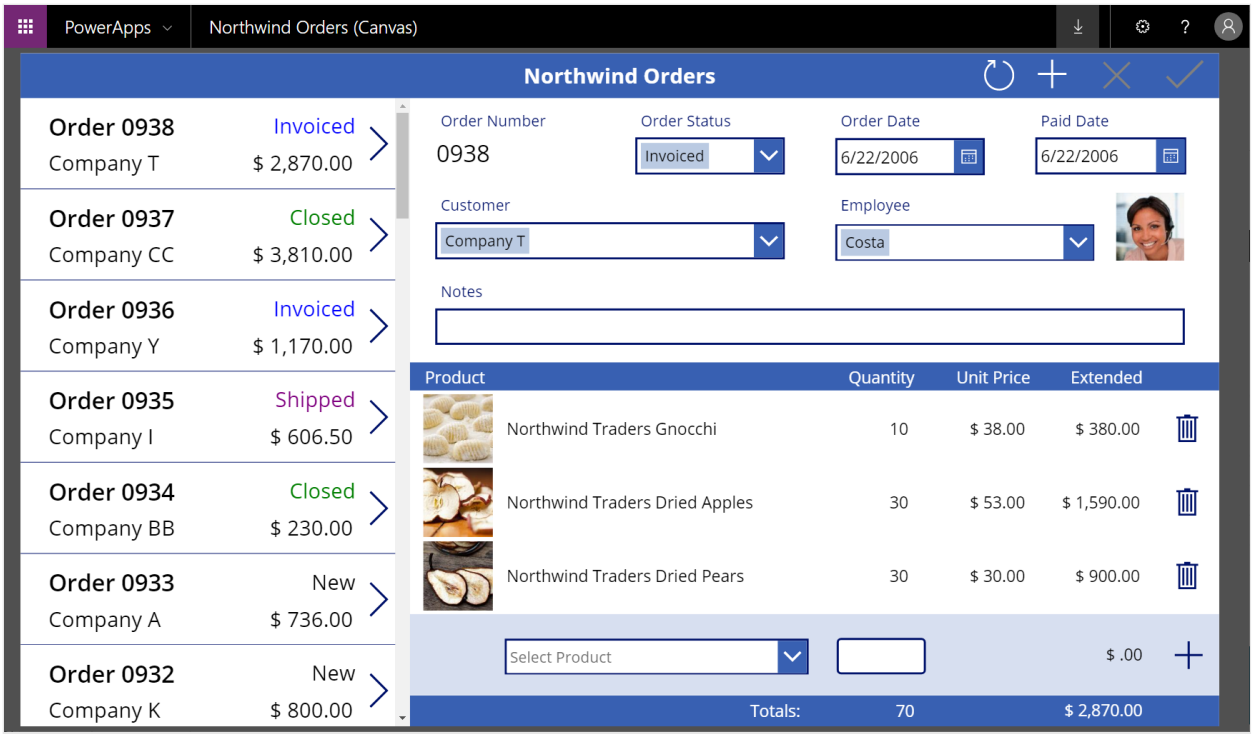
The Northwind solution includes these apps for interacting with this data:

- Northwind Orders (Canvas)
- Northwind Orders (Model-driven)

You open these apps the same way that you opened the app in the previous procedure.

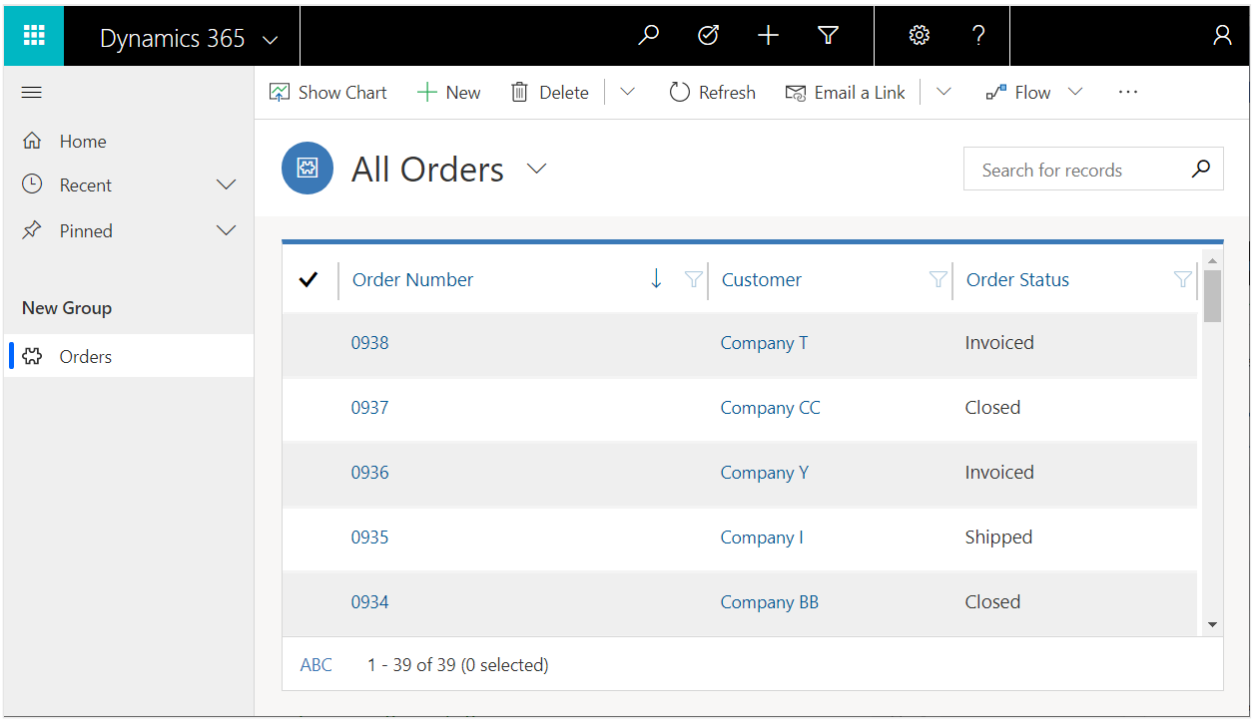
Canvas

This single-screen app offers a simple master-detail view of the **Orders** table, where you can view and edit a summary of the order and each line item for an order. A list of orders appears near the left edge, and you can select an arrow in that list to show a summary and the details of that order. More information: Overview of the canvas app for Northwind Traders.

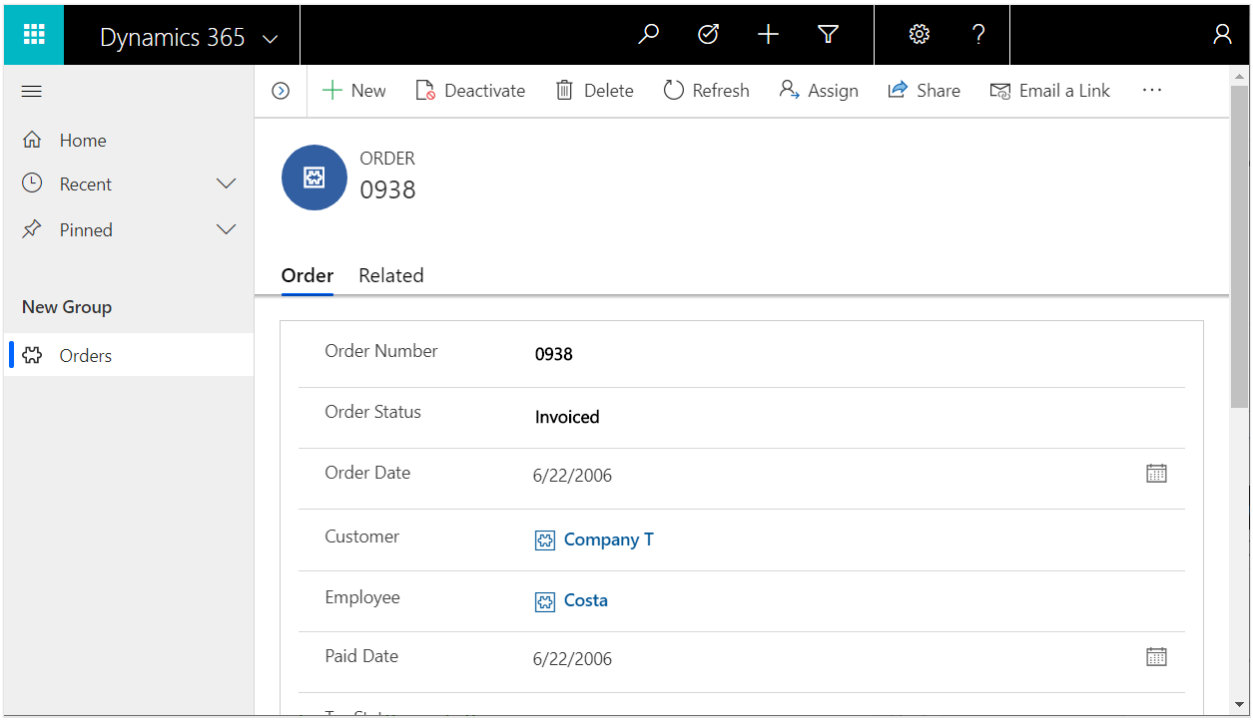


Model-driven

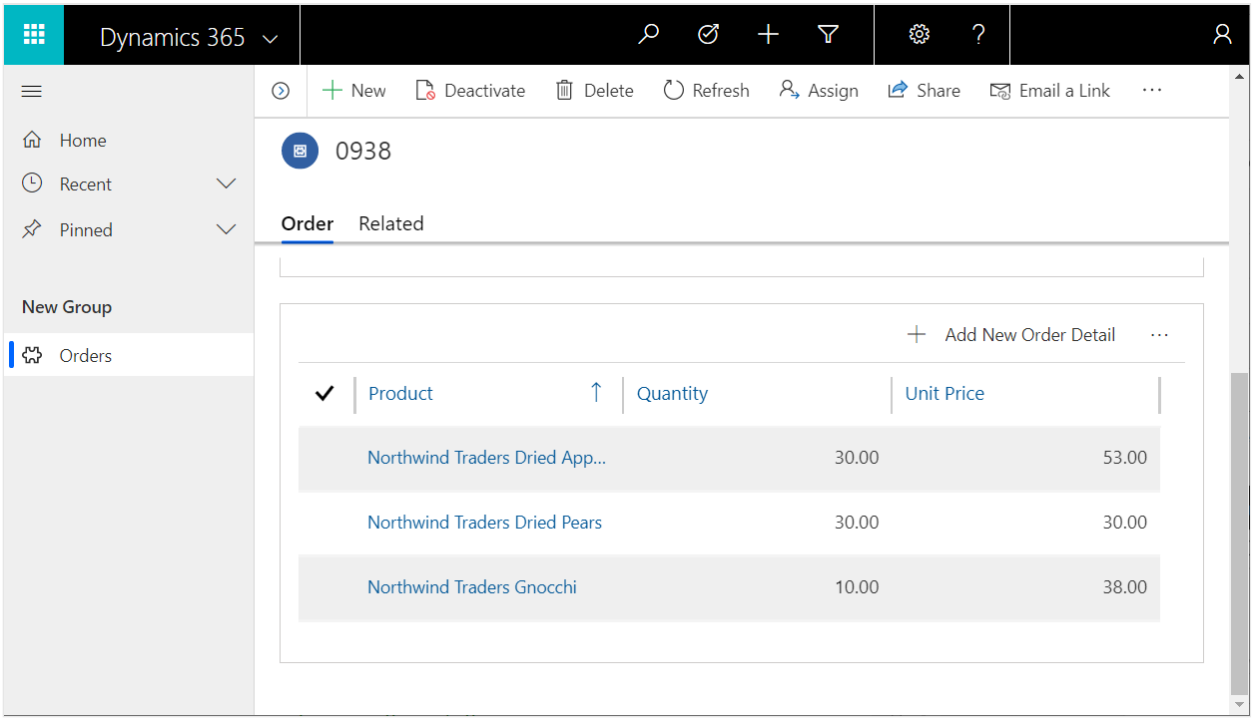
This app operates on the same data (in the **Orders** table) as the canvas app. In the list of orders, show more information about an order by selecting its number:



A summary of the order appears on a separate form:



If you scroll down the form, it shows the same line items as the canvas app does:



Do it yourself

You can follow step-by-step instructions to create the canvas app shown earlier in this topic. The instructions are divided into three parts:

1. Create an order gallery.
2. Create a summary form.
3. Create a detail gallery.

If you want to skip ahead, the solution contains a starting-point app for each part. In the list of apps, look for **Northwind Orders (Canvas) - Begin Part 1** and so on.

This overview of the app explains the user interface, data sources, and how relationships are used.

Start by reading the overview

Overview of the canvas app for Northwind Traders

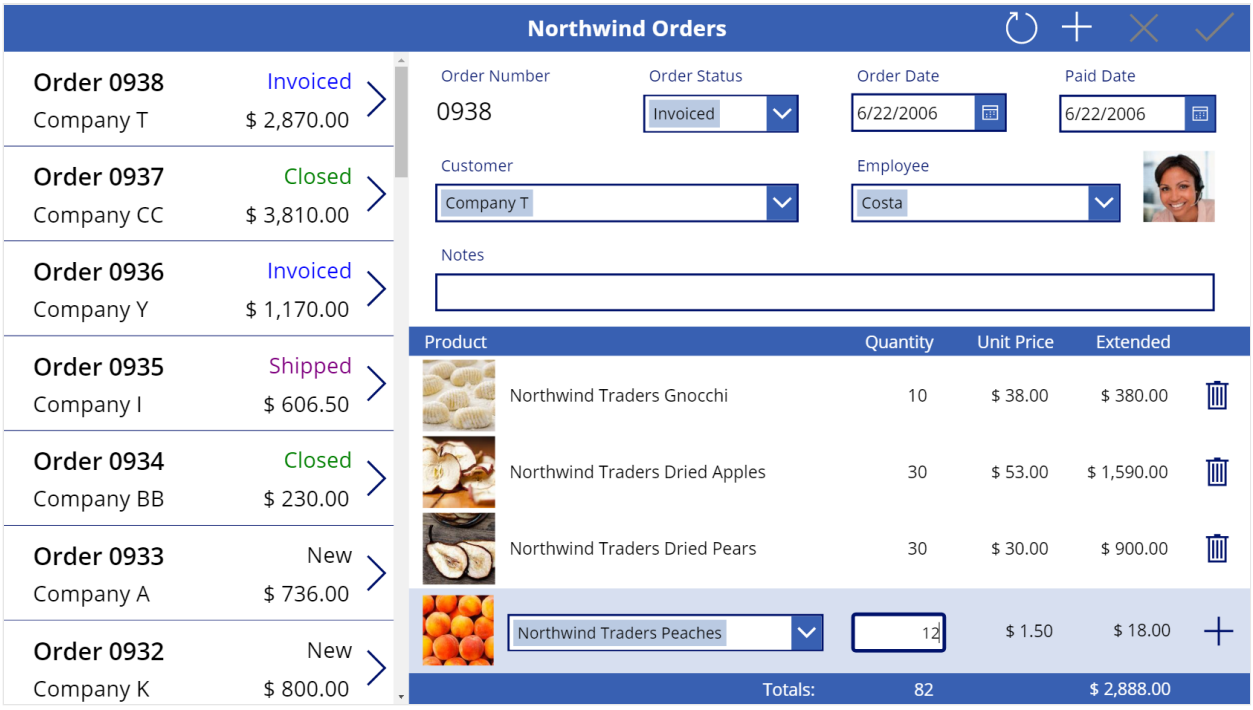
Article • 12/16/2022

Learn about the canvas app for managing relational data in the Northwind Traders database that you installed in your environment. Then follow step-by-step instructions in subsequent topics to build this app from scratch, thereby gaining hands-on experience working with relational data.

In this topic, discover:

- How an app user shows and manages relational data in the app.
- Which types of data drive the app.
- How relationships between those types of data were created.

In a single screen, the app user can show, update, create, and delete orders.



Explore the user interface

Order gallery

On the left edge of the app, a gallery shows a list of orders, including the order number, the status, the name of the customer, and the total cost of the order. The user can scroll through the list to find an order and then show more information about it by selecting the order's arrow. More information: Create the order gallery.

Summary form

In the upper-right corner, a form summarizes the order that the user selected in the order gallery. The summary includes much of the same information as that gallery does, but the summary also shows the dates when the order was created and paid, as well as the name and the picture of the employee who managed the order. The user can change the data in the form, save those changes, cancel them, or delete the order by selecting an icon near the right edge of the title bar. More information: Create the summary form.

Detail gallery

In the lower-right corner, another gallery shows information about which products the selected order contains and in what quantities. Each item in this gallery is known as an order detail. The app user can add and delete any item in that gallery by using controls in and under it. More information: Create the detail gallery.



Explore the data sources

To create this app, you'll show data from five tables and an choice. In fact, most areas of this app show data from multiple tables. For example, the order gallery contains this information:

- The order number is a field in the **Orders** table.
- The status is another field in the **Orders** table, an option from the **Orders Status** choice.

- The customer name is a field in the **Customers** table.
- The total cost is calculated based on data in the **Order Details** table.

The summary contains some of the same information as the list of orders, but it also contains the name and the picture of the employee who managed the order. That information is pulled from fields in the **Employees** table. The detail gallery shows records in the **Order Details** table, and each product in those details is a record in the **Order Products** table.

Explore the relationships

You can show data from different sources (for example, tables) in the same gallery or form because those tables have relationships that were created for you in the database.

Many-to-one relationships

For example, information about the customer and the employee for each order resides in the **Customers** and **Employees** tables. Therefore, the **Orders** table has many-to-one relationships with those tables because there are many orders, each of which can be placed by only one customer and managed by only one employee.

Each order also has one or more line items that represent the products that the order contains and their quantities. Each line item is a record in the **Order Details** table, which pulls information about each product from the **Order Products** table. Each detail identifies only one product, but each product can appear in multiple details. Therefore, the **Order Details** table has a many-to-one relationship with the **Order Products** table.

One-to-many relationships

Each order can contain multiple line items, but each line item relates to only one order. Therefore, the **Orders** table has a one-to-many relationship with the **Order Details** table.

Dot notation for relationships

To show data based on a relationship between tables, you can use the dot property selector to walk across a relationship from one table to another. For example, each record in the **Orders** table pulls information from the **Customers** table so that the order gallery can show the customer names. In that gallery, you configure this behavior by setting the **Text** property of a label to this expression:

`ThisItem.Customer.Company`

ThisItem specifies a record in the **Orders** table and pulls information from the **Customers** table about the customer who placed the order. In this case, the expression specifies that the customer's company name appears. However, the entire record for that customer is pulled, so you could just as easily show, for example, an email address for that customer instead.

As another example of walking from one table to another, you can specify that a gallery should show records in one table based on a record that the user selected in another gallery and that's in another table. To show the order details, you'll set the detail gallery's **Items** property to this expression:

```
Gallery1.Selected.'Order Details'
```

In this case, **Gallery1.Selected** specifies a record in the **Orders** table, just as **ThisItem** did in the previous example. However, this expression doesn't pull just one record as the previous expression did. Instead, it pulls an entire table of records to show the name and per-unit cost of each product (as reflected in the **Order Products** table) and the quantity (as reflected in the **Order Details** table).

Do it yourself

You can follow step-by-step instructions to create the Northwind Orders canvas app. The instructions are divided into three parts:

1. Create an order gallery.
2. Create a summary form.
3. Create a detail gallery.

If you want to skip ahead, the solution contains a starting-point app for each part. In the list of apps, look for **Northwind Orders (Canvas) - Begin Part 1** and so on.

Continue by creating the order gallery

Create an order gallery in a canvas app

Article • 12/16/2022

Follow step-by-step instructions to create an order gallery in a canvas app for managing fictitious data in the Northwind Traders database. This topic is part of a series that explains how to build a business app on relational data in Microsoft Dataverse. For best results, explore these topics in this sequence:

1. Create an order gallery (**this topic**).
2. Create a summary form.
3. Create a detail gallery.



Prerequisites

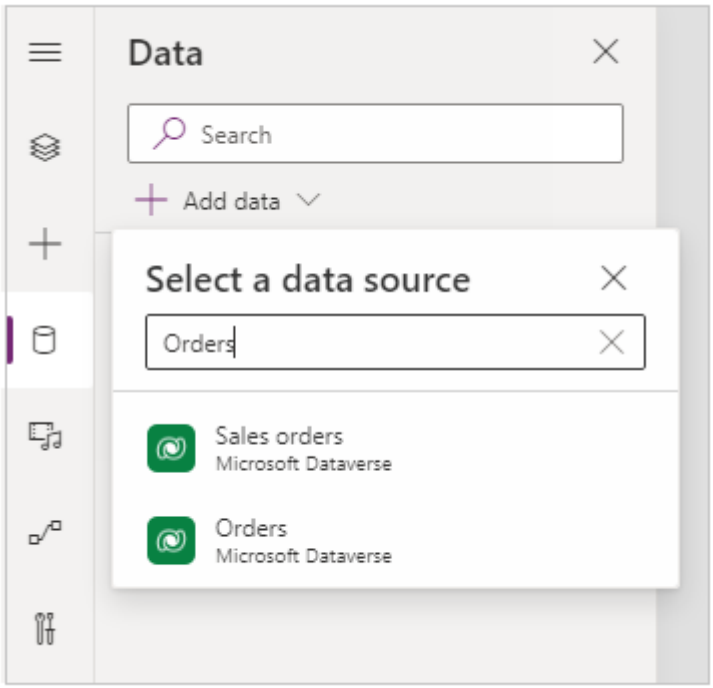
- Install the Northwind Traders database and apps.
- Read through the overview of the canvas app for Northwind Traders.

Create a blank app

Create a blank canvas app, and give it a name such as "My Northwind Orders (Canvas app)".

Add the data

1. On the left-pane, select **Data** > **Add data** > search for, and select **Orders** table.



The **Orders** table contains many fields of various types:

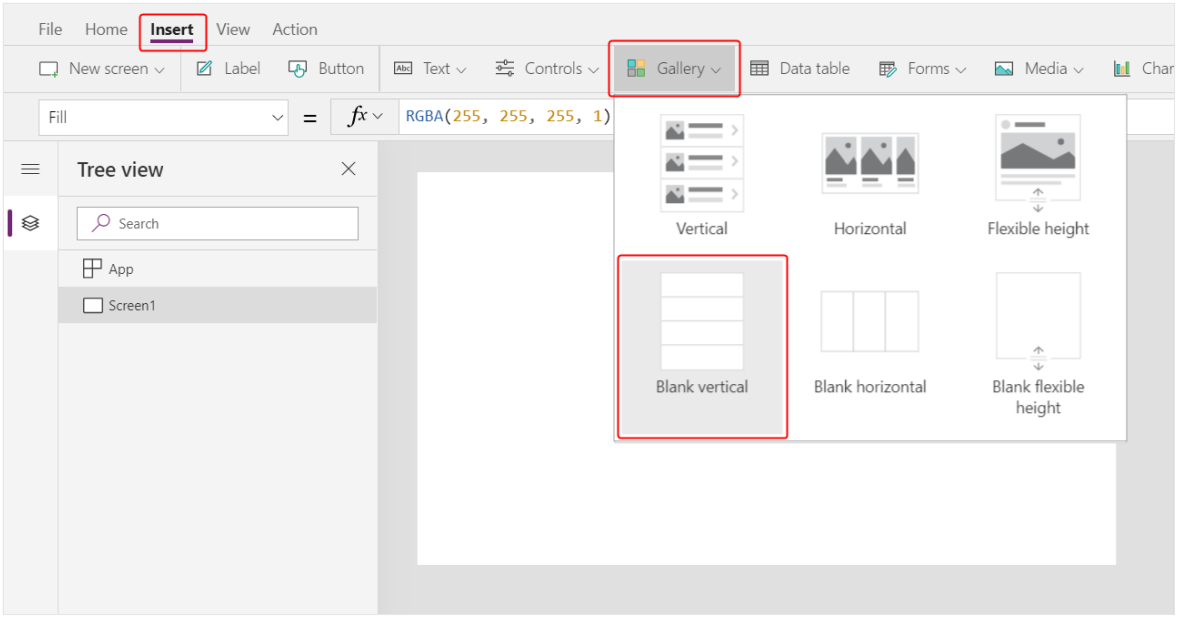
Display name	Name	Data type	Type	Required	Searcha...
Customer	nwind_customerid	Lookup	Custom		✓
Employee	nwind_employeeid	Lookup	Custom		✓
Notes	nwind_notes	Multilin...	Custom		✓
Order Date	nwind_orderdate	Date Only	Custom		✓
Order Number	nwind_ordernumber	Autonu...	Custom		✓
Order Status	nwind_orderstatusid	Option ...	Custom		✓
Paid Date	nwind_paiddate	Date Only	Custom		✓
Payment Type	nwind_paymenttype	Text	Custom		✓
Sample Data Order Status	nwind_sampledataorder...	Text	Custom		✓
Sample Data Original ID	nwind_sampledataorigi...	Text	Custom		✓
Ship Address	nwind_shipaddress	Multilin...	Custom		✓
Ship City	nwind_shipcity	Text	Custom		✓

Each field has a **Display name** and a **Name**, which is sometimes called the logical name. Both names refer to the same thing. In general, you'll use the display name when you build an app, but some cases require the more cryptic **Name**, as noted in a procedure.

2. As we will be working with screens and controls next, in Power Apps Studio switch back to the **Tree View** on the left hand side by pressing the three stacked squares icon. You can return to the **Data Sources** at any time by pressing the cylinder icon.

Create the order gallery

1. On the **Insert** tab, select **Gallery** > **Blank vertical** to add a Gallery control, which will show the orders.

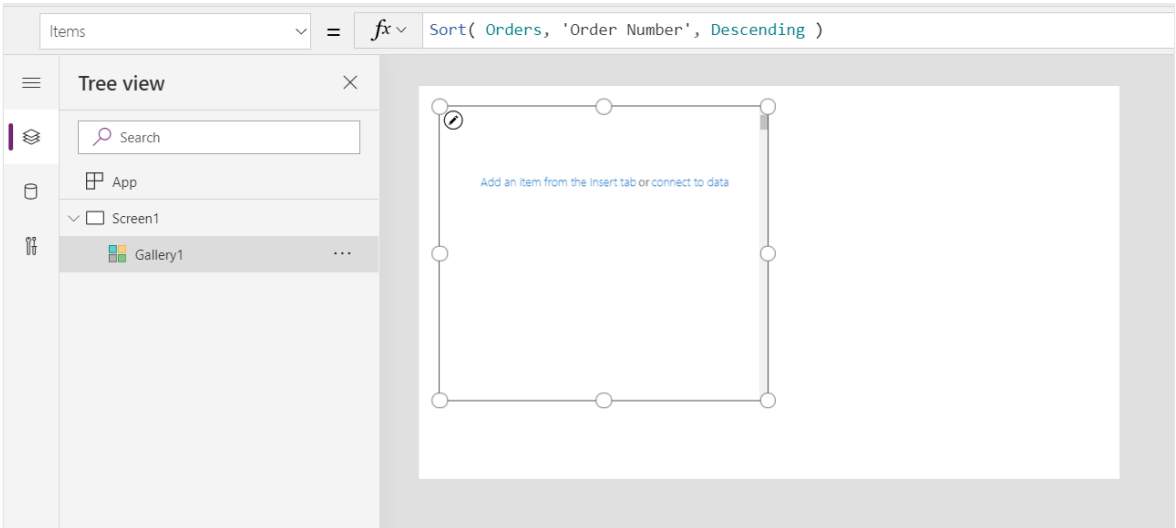


   The control will be placed on the canvas and a fly out dialog will appear asking which data source to connect to.

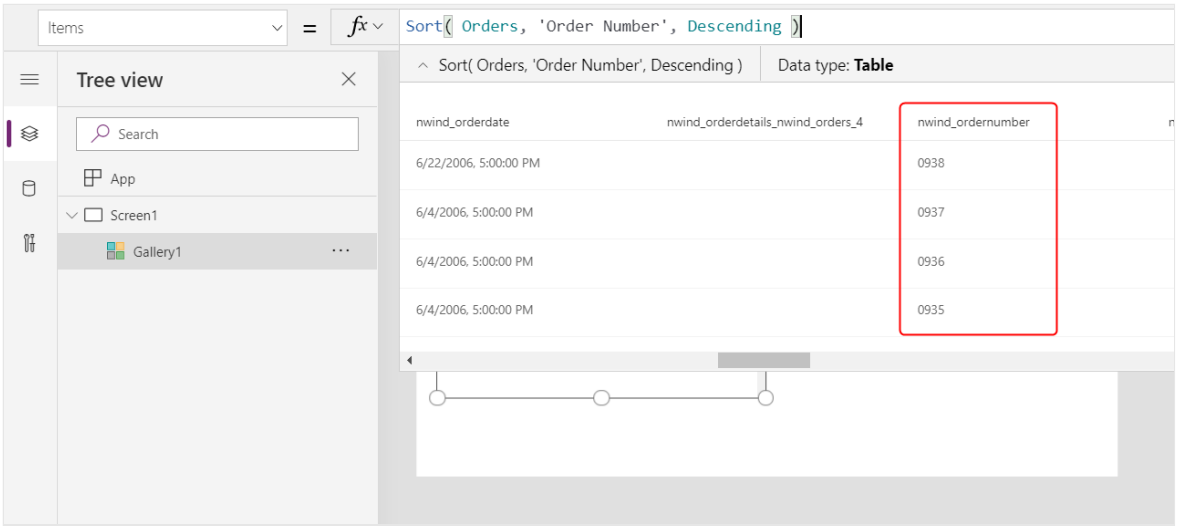
2. We could connect it directly to **Orders** here, but instead we'd like to control the sort order of the gallery. Ignore the fly out dialog and in the formula bar set the gallery's **Items** property to this formula:

```
Power Apps
Sort( Orders, 'Order Number', Descending )
```

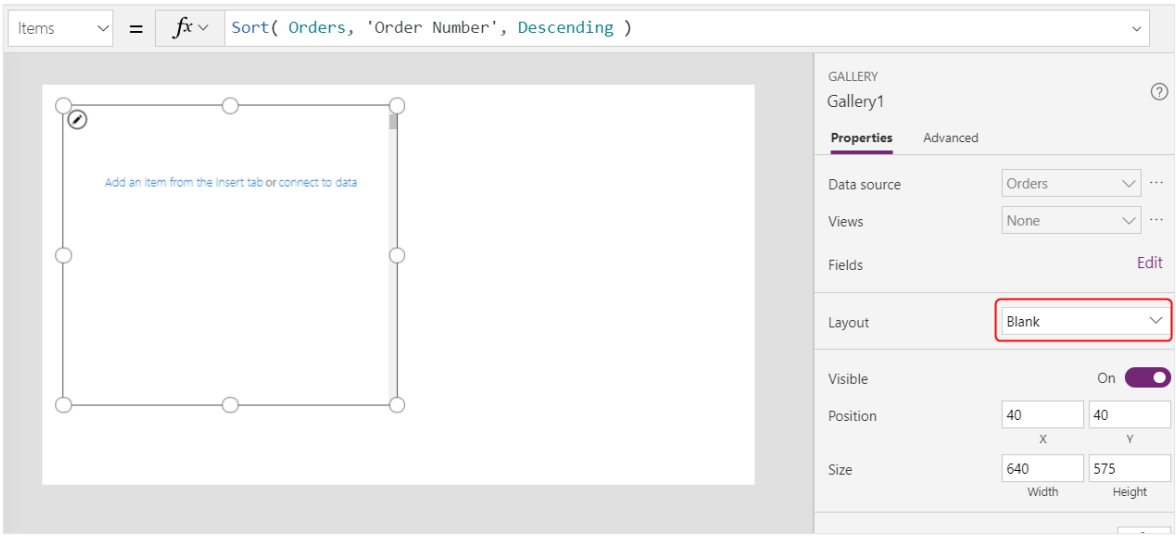
   The Sort function orders the list so that the newest order (which has the highest order number) appears first.



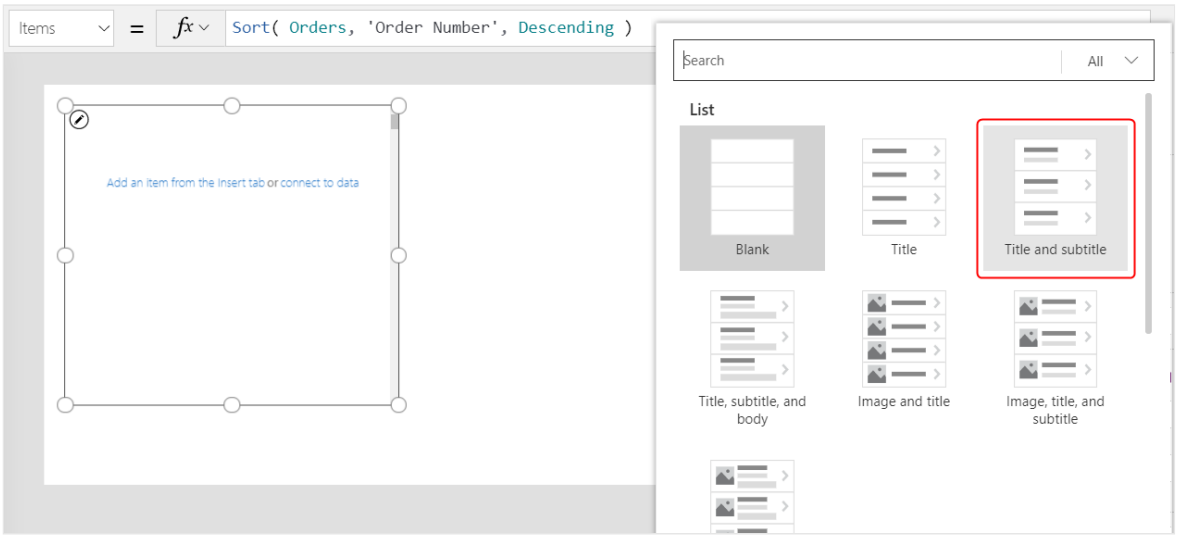
3. After a few moments the Result View will appear below the formula bar. Pull down on the arrow at the left to see the result of our formula. Scroll to the right to see the **Order Number** column and ensure it is sorted the way want (highest to lowest).



4. In the **Properties** tab near the right edge, open the **Layout** list:

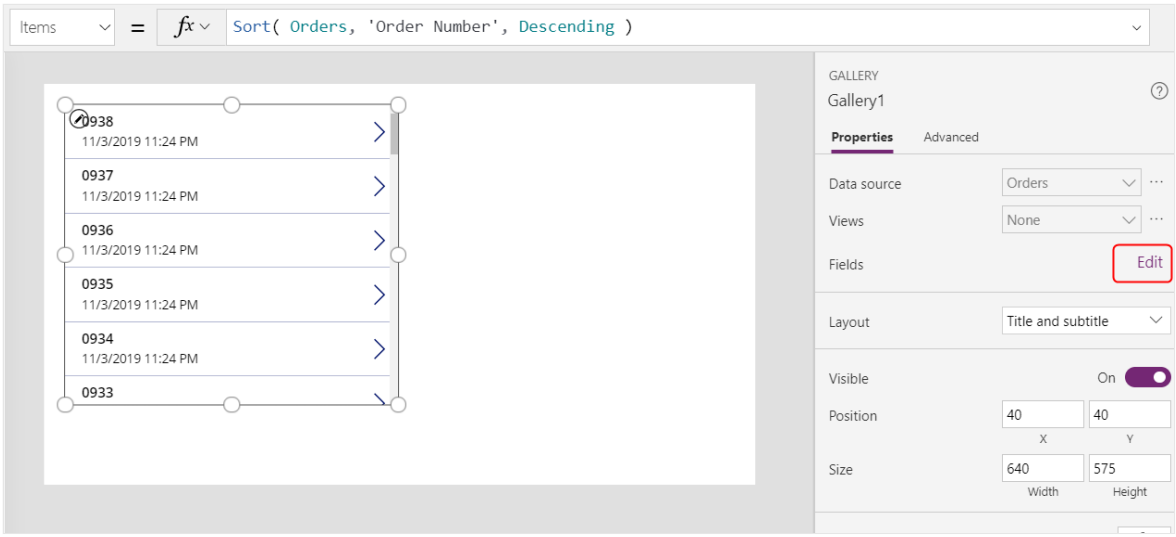


5. In the list of options, select **Title and subtitle**:



Two **Label** controls are added in the gallery's template. By default, these controls show two columns of the **Orders** table, which you'll change next. The gallery's template is replicated vertically for each record in the table.

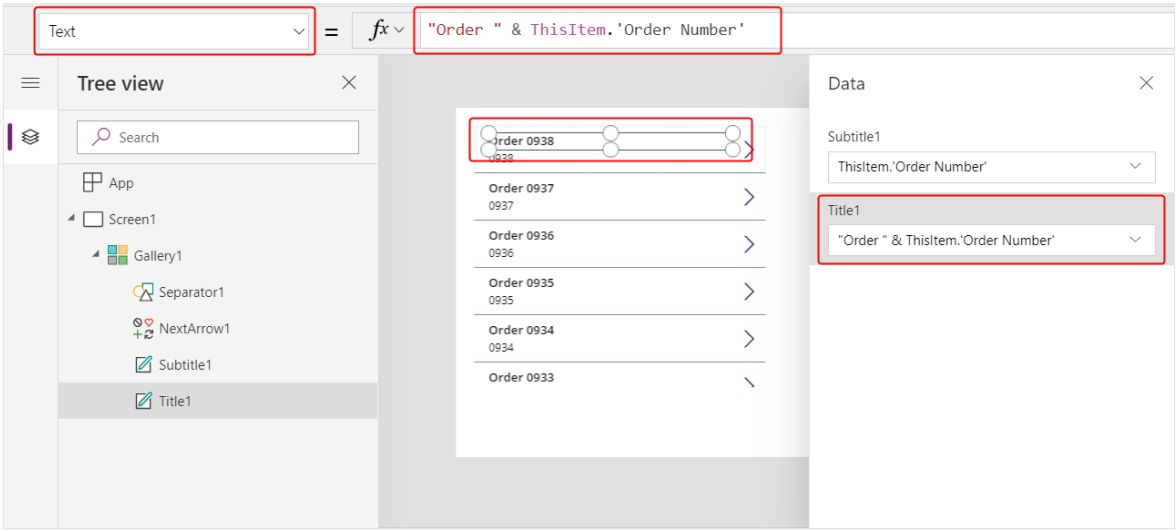
6. Select **Edit** (next to **Fields**) in the **Properties** tab near the right edge.



7. In the **Data** pane, select **Title1** (or select the upper label in the gallery's template).

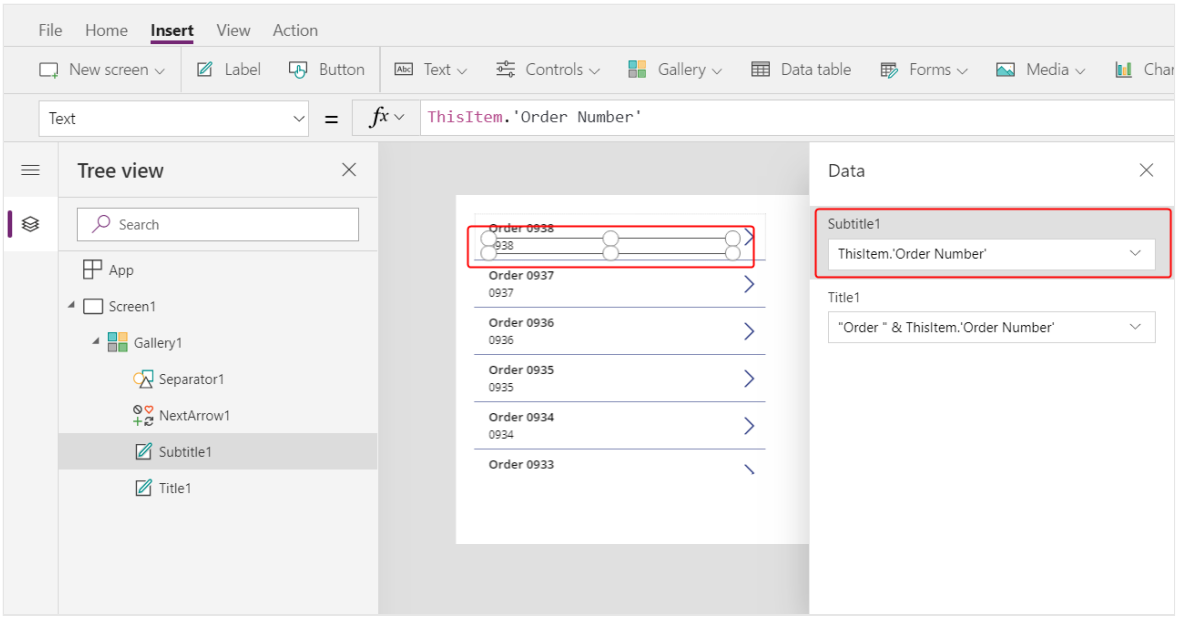
8. In the formula bar, set the label's **Text** property to this expression:

```
Power Apps

"Order " & ThisItem.'Order Number'
```



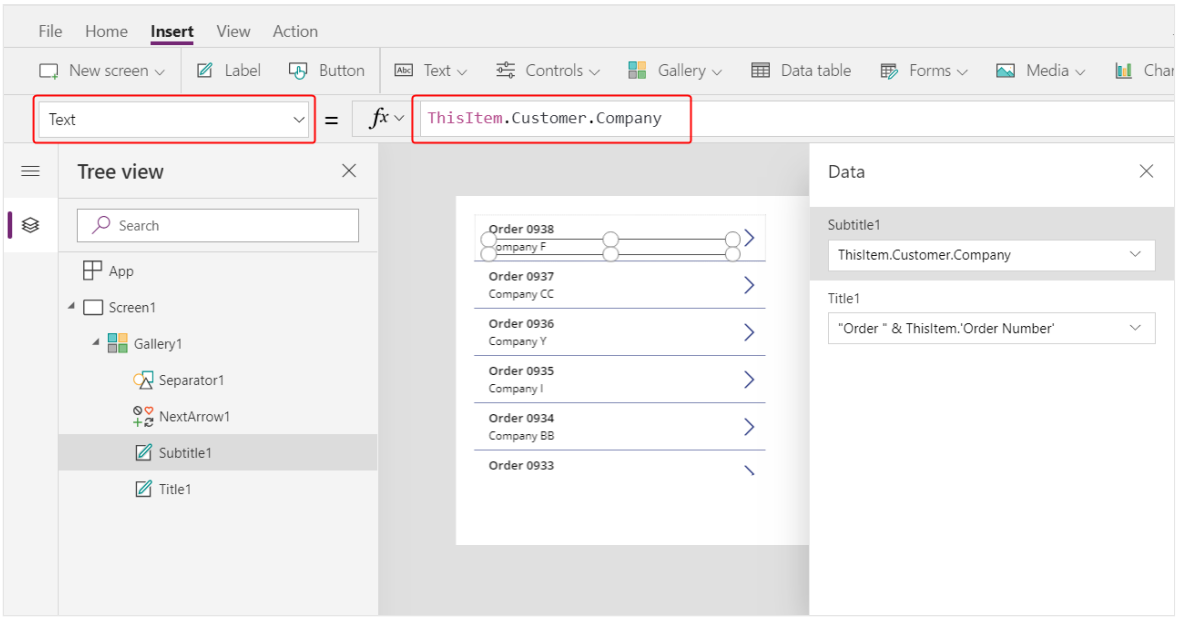
The order number appears at the top of each gallery item. In the gallery template, **ThisItem** grants access to all fields in the **Order** table.

9. In the **Data** pane, select **Subtitle1** (or select the lower label in the gallery's template):



10. In the formula bar, set the label's **Text** property to this expression:

```
Power Apps

ThisItem.Customer.Company
```



After you enter this formula, it may show a red squiggly error for a moment. The error should clear if you select anything outside the formula bar and then return the cursor to the formula bar. If the error persists or you don't see a value, select the **View** tab, select **Data sources**, and then refresh the **Orders** table by selecting the ellipsis (...) to the right of the data-source name.

When you specify **ThisItem.Customer**, you're leveraging a many-to-one relationship between the **Orders** and **Customers** tables and retrieving the customer record that's associated with each order. From the customer record, you're extracting data in the **Company** column for display.

You can show all the relationships from the **Orders** table to other tables, including the **Customer** table:



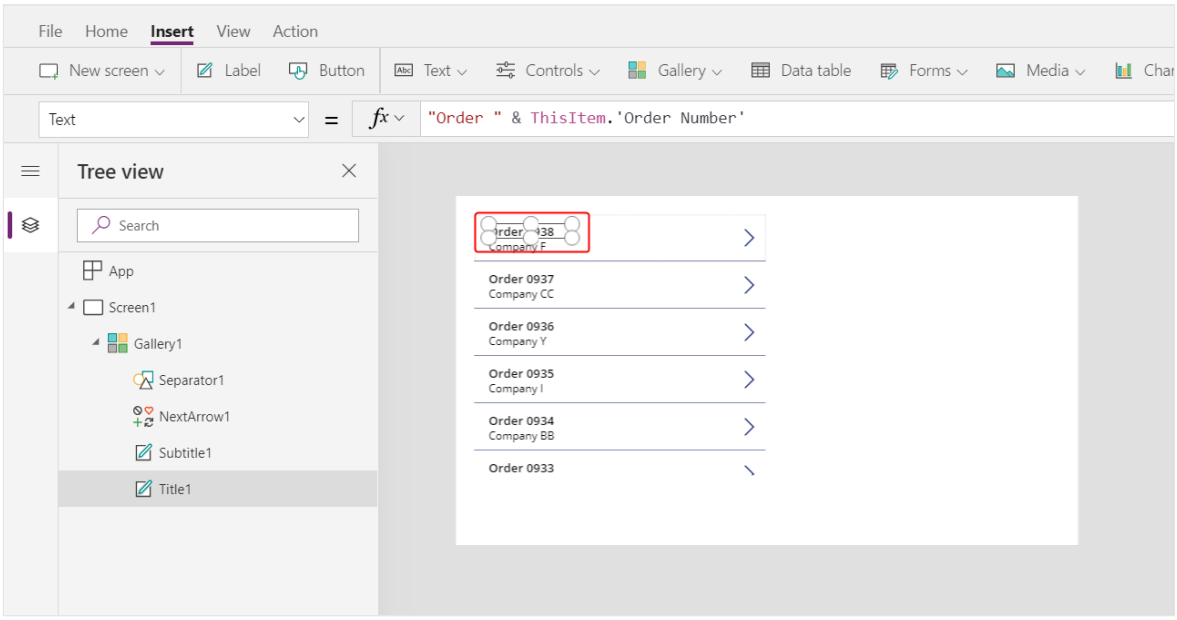
Display name	Relationship name	Related ...	Relationshi...	Type
Customer	nwind_orders_nwind_c...	Customer	Many-to-one	Custom
Customer Order	nwind_inventorytransa...	Inventory T...	One-to-many	Custom
Employee	nwind_orders_nwind_e...	Employee	Many-to-one	Custom
Order	nwind_orderdetails_nw...	Order Detail	One-to-many	Custom
Order	nwind_invoices_nwind_...	Invoice	One-to-many	Custom
Shipper	nwind_orders_nwind_s...	Shipper	Many-to-one	Custom

11. Close the **Data** pane by selecting the close icon (x) in its upper-right corner.

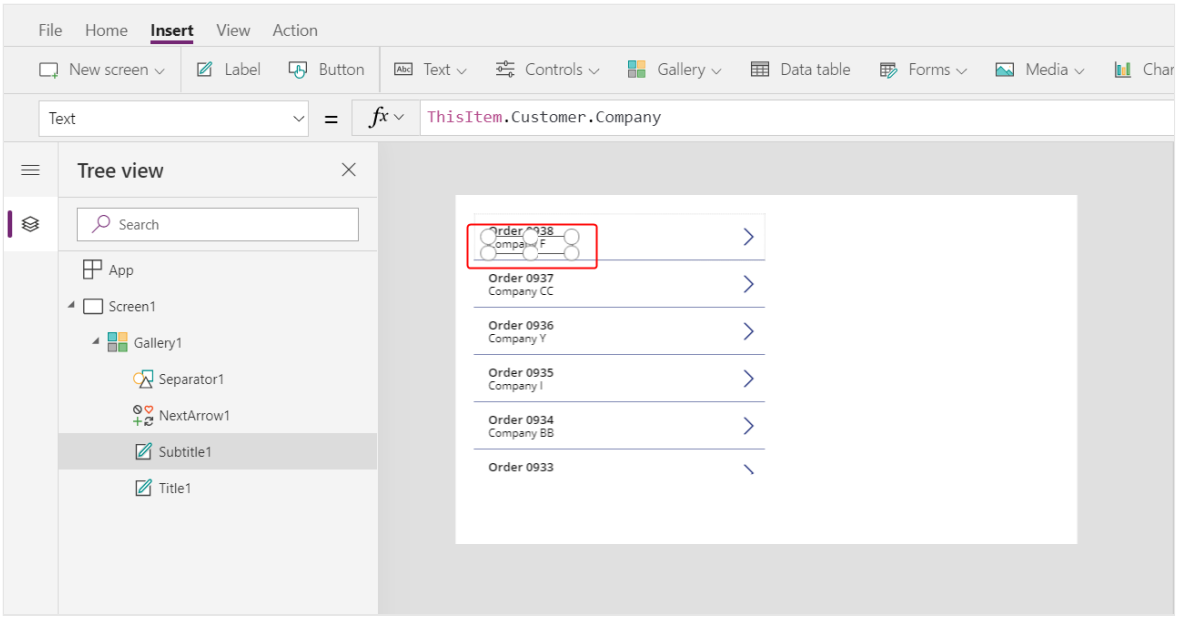
Show each order's status

In this procedure, you'll add space in the gallery for a label and configure it to show each order's status in a different color based on the data.

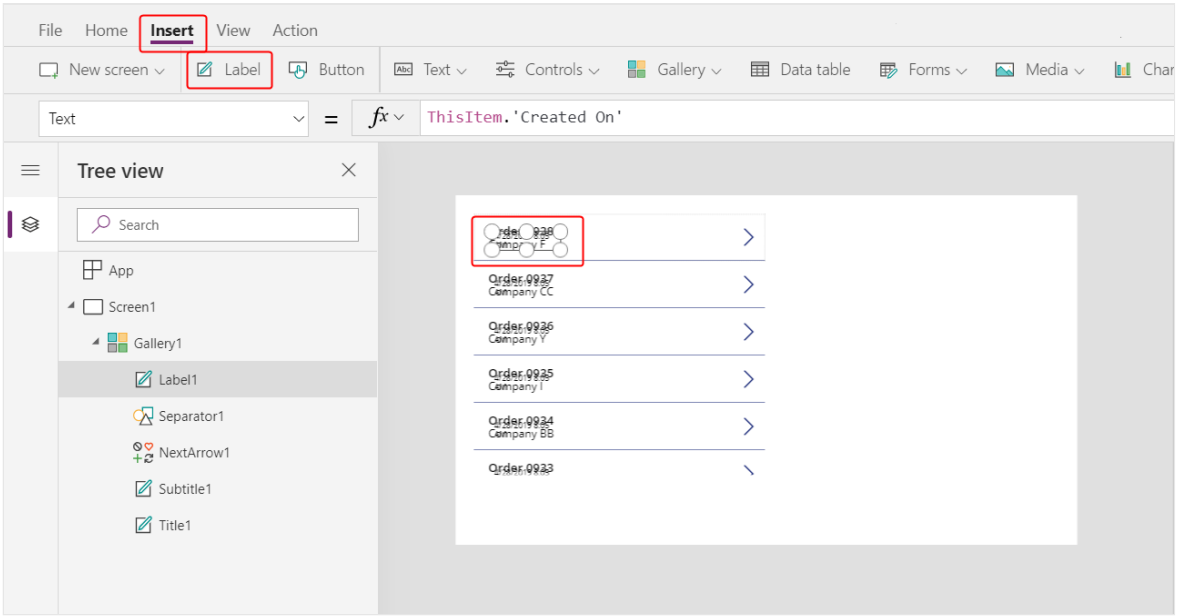
1. In the gallery's template, reduce the width of the first label, **Title1**:



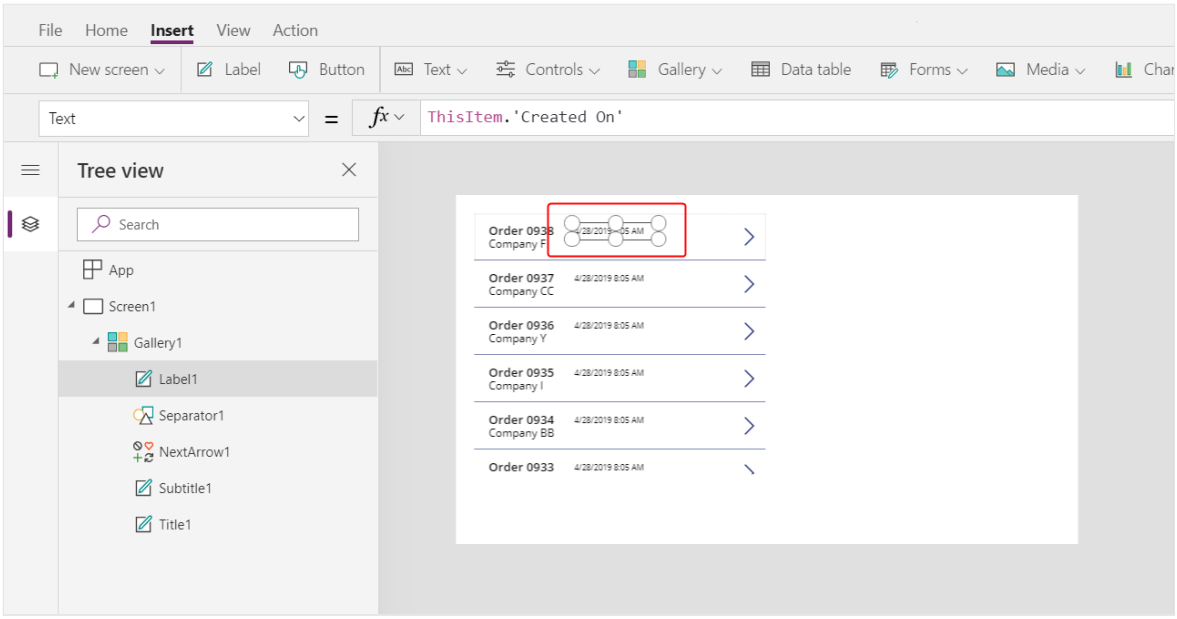
2. Repeat the previous step with the second label, **Subtitle1**:



3. With the gallery template (or a control in the template) selected, select **Label** on the **Insert** tab:

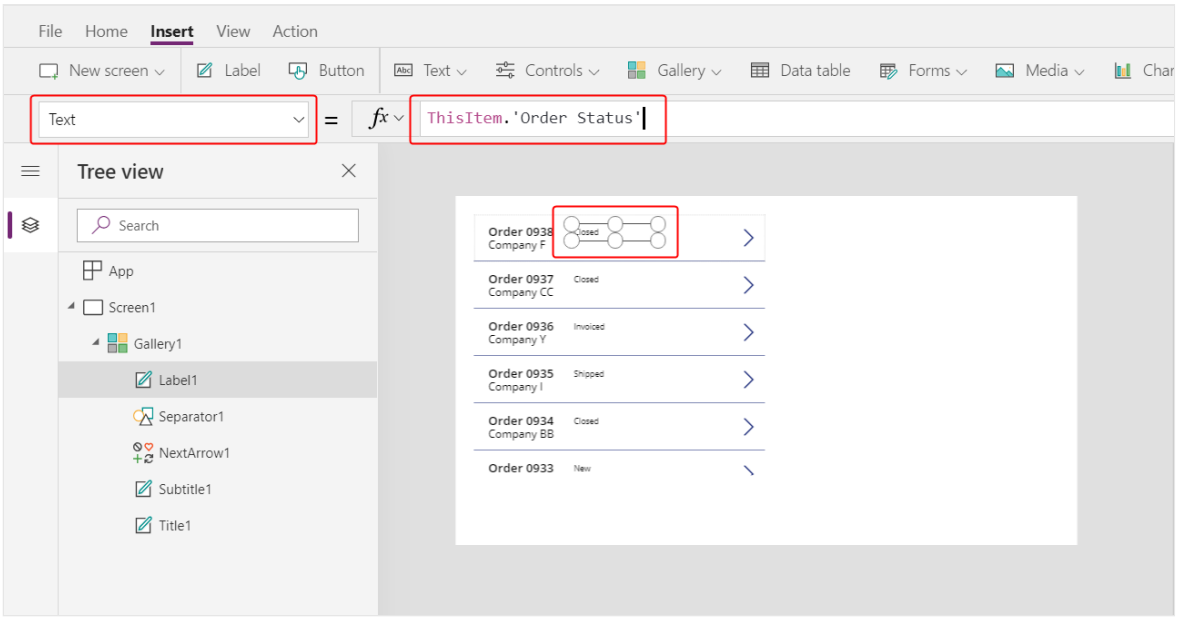


4. Move the new label to the right of the **Title1** label:

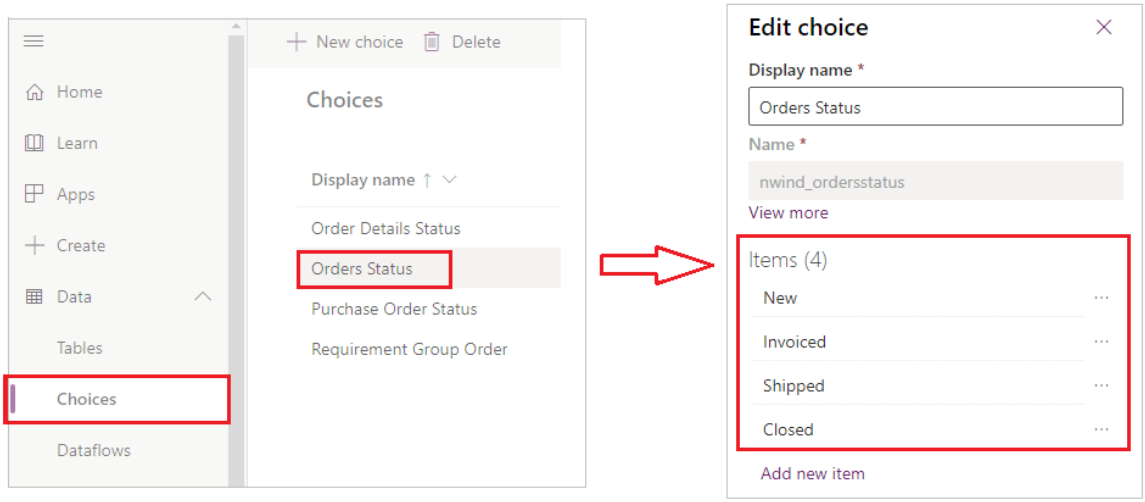


5. Set the **Text** property of the new label to this expression:

```
Power Apps

ThisItem.'Order Status'
```



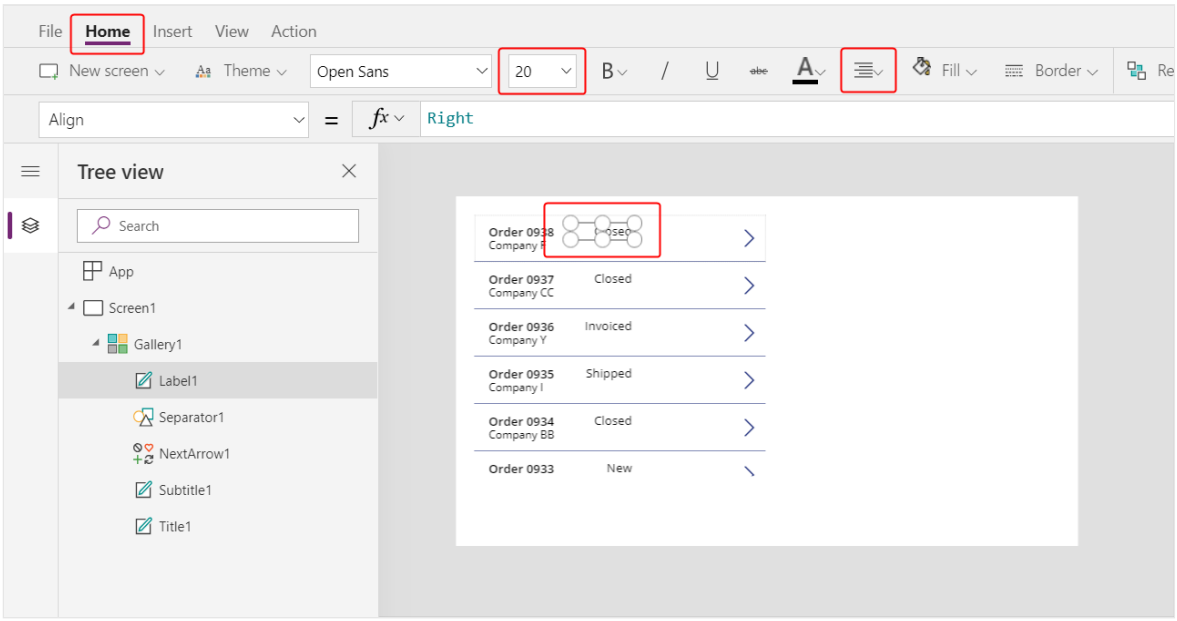
In the **Orders** table, the **Order Status** field holds a value from the **Orders Status** choice. A choice is similar to an enumeration in other programming tools. Each set of options is defined in the database, so users can specify only those options that are in the set. The **Orders Status** choice is also global, not local, so you can use it in other tables:



Each option in a set has a name that appears if you show it in a label. These names can be localized, and the app recognizes the same option whether an English user selects **Apple**, a French user selects **Pomme**, or a Spanish user selects **Manzana**. For this reason, you can't create a formula that relies on a hard-coded string for an option, as this topic demonstrates later.

In formulas, you must surround **Order Status** with single quotation marks because it contains a space. However, that name functions the same way as any other name in Power Apps, such as **Customer** or **Company**, does.

6. On the **Home** tab, increase the font size of the status label to 20 points, and right align the text:

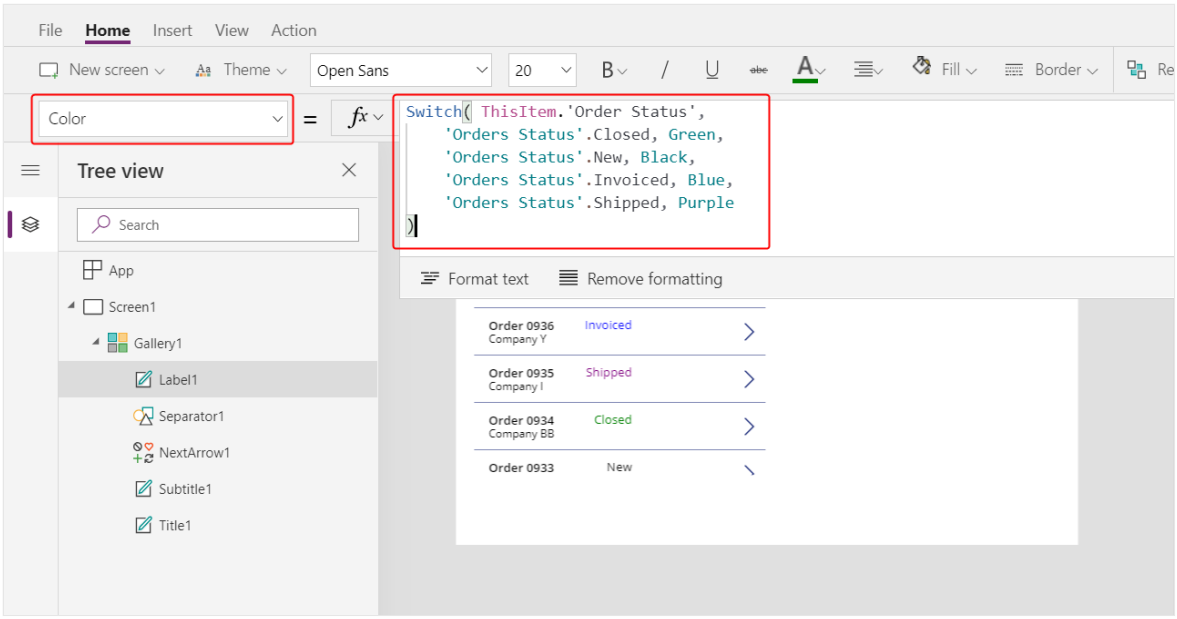


7. In the formula bar, set the **Color** property of the status label to this formula:

Power Apps

```
Switch( ThisItem.'Order Status',
    'Orders Status'.Closed, Green,
```

```
    'Orders Status'.New, Black,
    'Orders Status'.Invoiced, Blue,
    'Orders Status'.Shipped, Purple
)
```

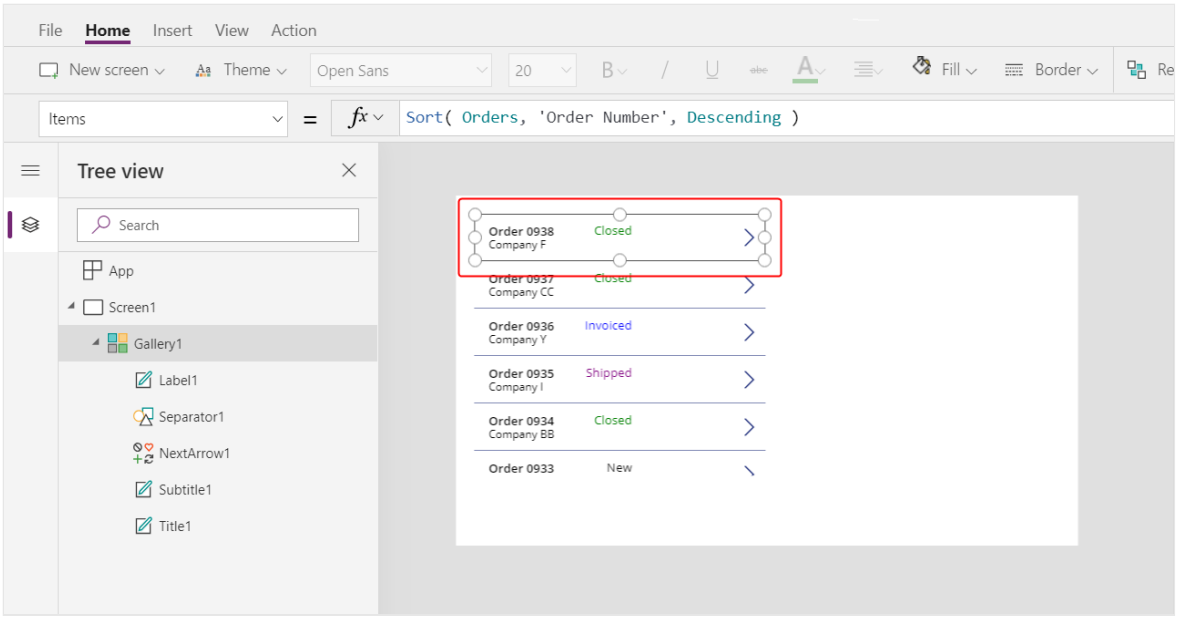


Power Apps prevents you from creating a formula that relies on a hard-coded string for each option in a set because such formulas could produce inappropriate results if the option names are localized. Instead, the **Switch** function determines the color based on whatever string appears in the label based on the user's settings.

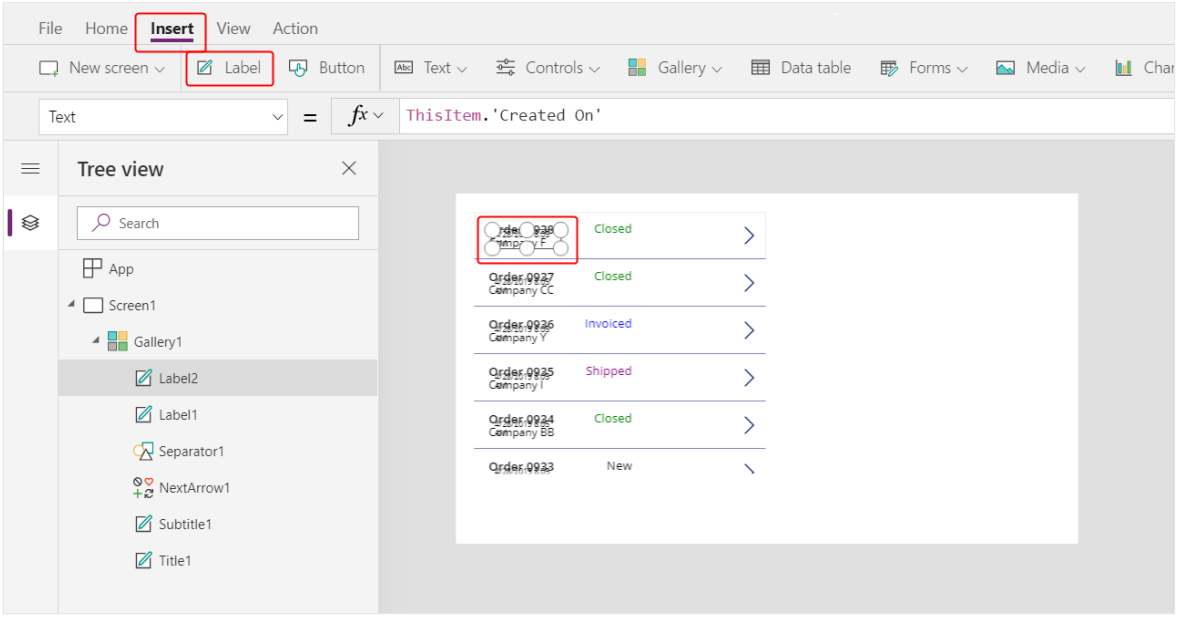
With this formula in place, different status values appear in different colors, as the previous graphic shows.

Display each order's total

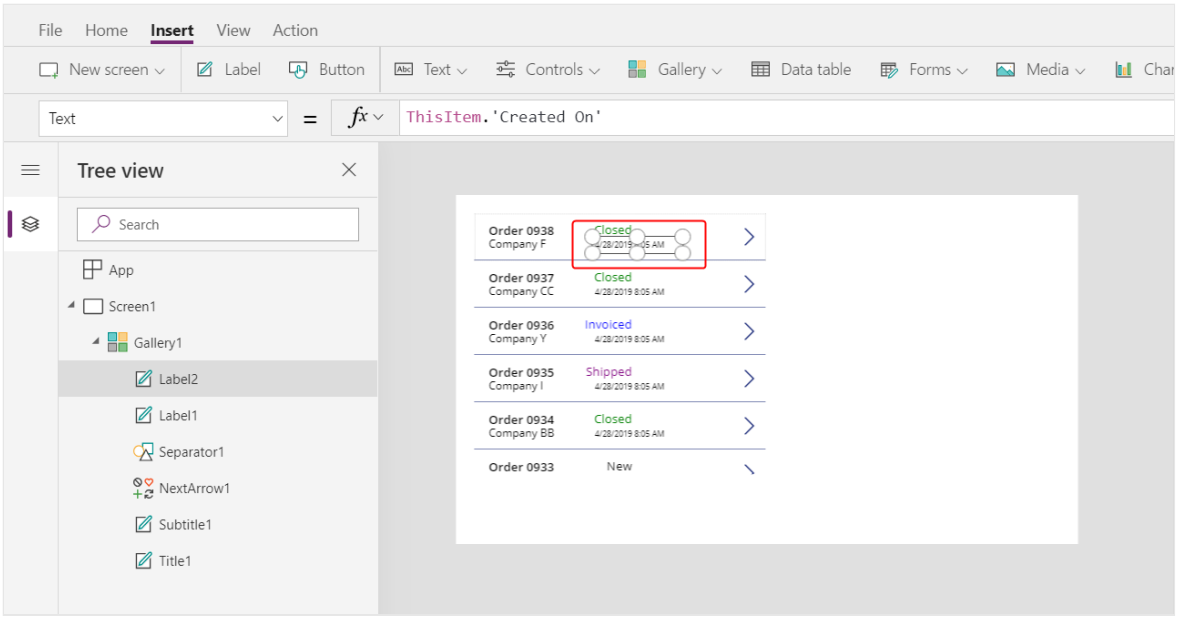
1. Select the first item in the gallery, which is the gallery's template:



2. On the **Insert** tab, select **Label** to add another label:

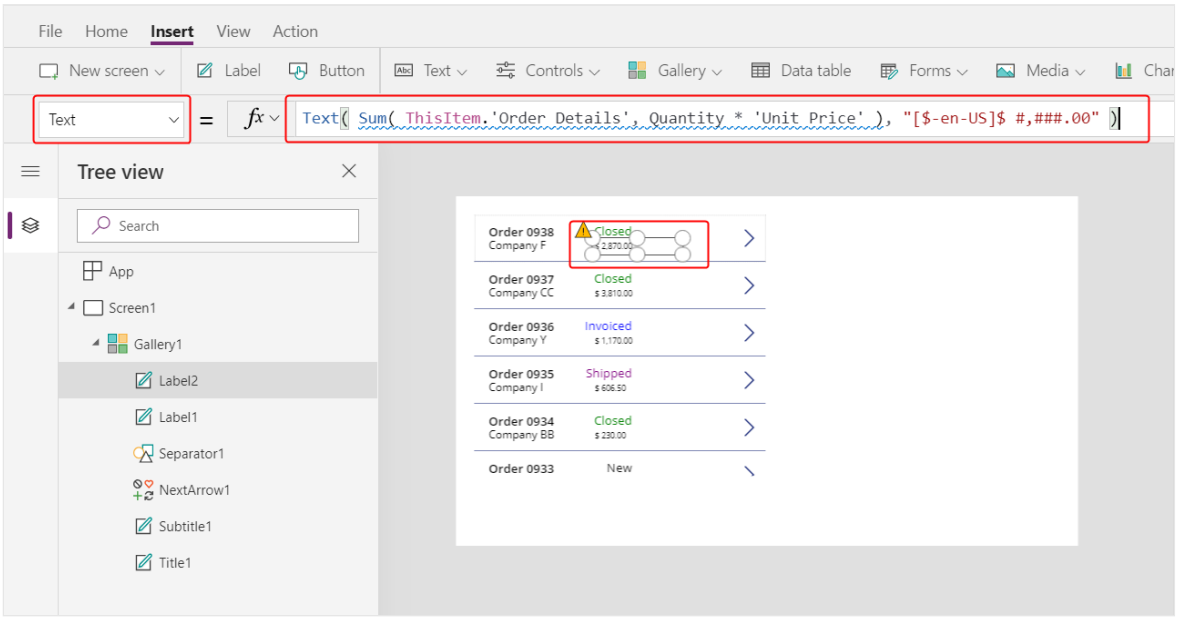


3. Move the new label so that it appears under the status label:



4. In the formula bar, set the new label's **Text** property to this formula:

```
Power Apps

Text( Sum( ThisItem.'Order Details', Quantity * 'Unit Price' ), "[$-en-US]$ #,###.00" )
```



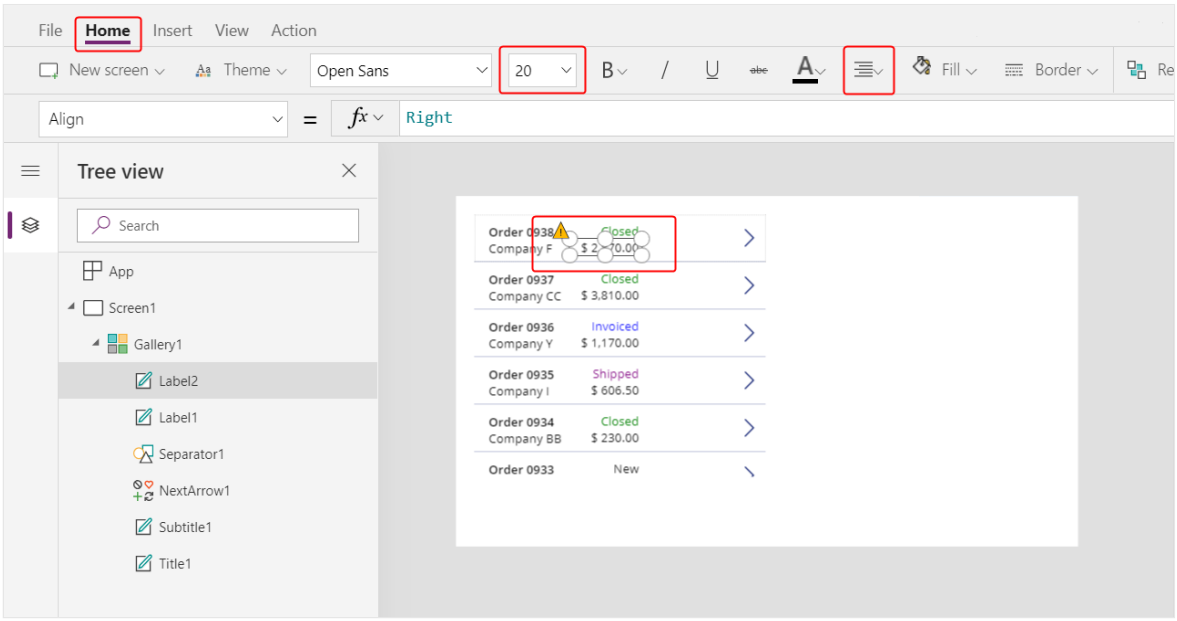
In this formula, the Sum function adds up the records in the **Order Details** table that are associated with each record in the **Order** table through a one-to-many relationship. These line items make up each order, and you'll use the same one-to-many relationship to show and edit the line items in the lower-right area of the screen.

This formula shows a blue underline and a delegation warning because Dataverse doesn't support delegation of complex aggregate functions (for example, the sum of a multiplication). You can ignore this information because no order in this

example will contain more than 500 line items. If necessary for a different app, you can increase that limit in **App settings**.

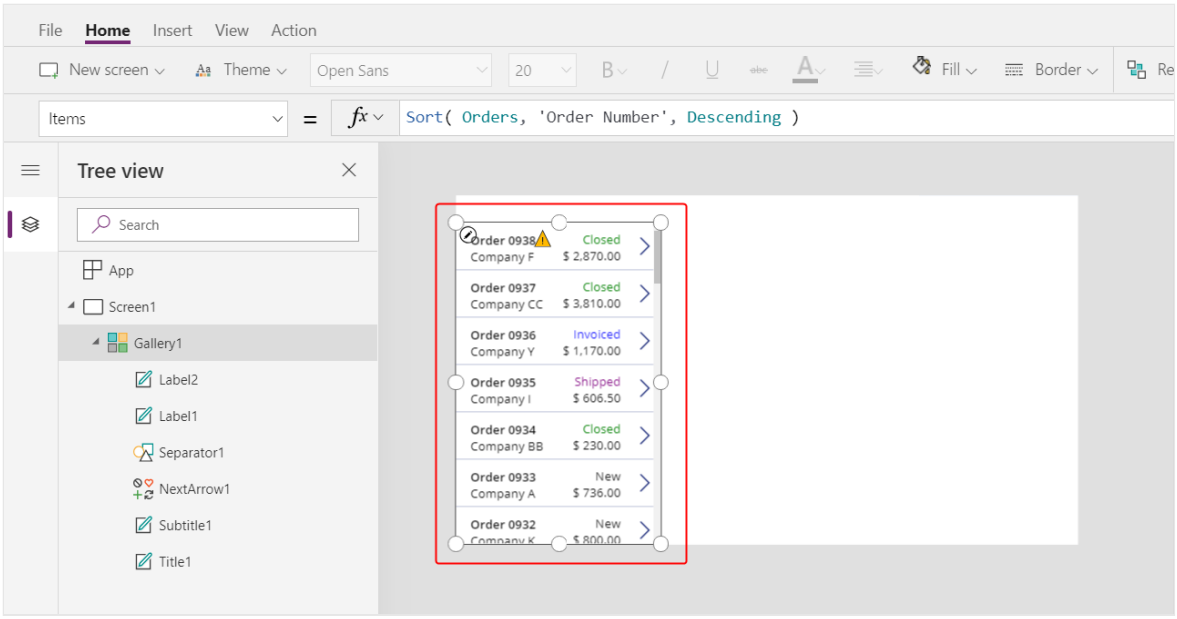
The **Text** function in this formula adds a currency symbol and formats the result with thousands and decimal separators. As written, the formula includes the language tag for U.S. English (**[$-en-US]**) and a dollar symbol (**$**). If you remove the language tag, it will be replaced with one based on your language settings, and the label will show the appropriate formats for that tag. If you leave the dollar symbol, the label will show the appropriate currency symbol based on the user's settings. However, you can force a different symbol to appear by replacing the dollar symbol with the one that you prefer.

5. On the **Home** tab, change the font size of the newest label to 20 points, and right align its text:



6. Move the gallery to the left edge of the screen, and decrease the gallery's width to close up some space.

7. Increase the gallery's height so that it's almost as tall as the screen, but leave a little room at the top for a title bar, which you'll add at the start of the next topic:



Summary

To recap, you started to build a single-screen canvas app by adding the order gallery, which includes these elements:

- An expression to show the order number: `"Orders " & ThisItem.OrderNumber`
- A field in a many-to-one relationship: `ThisItem.Customer.Company`
- A label that shows the name of an option in a set: `ThisItem.'Order Status'`
- A label that changes format based on which option in a set the label shows: `Switch( ThisItem.'Order Status', 'Orders Status'.Closed, Green, ...`
- A complex aggregate function over a one-to-many relationship: `Sum( ThisItem.'Order Details', Quantity * 'Unit Price' )`

Next topic

In the next topic, you'll add an **Edit form** control to display and edit an summary of whatever order the user selects in the gallery that you just created.

Create the summary form

Create a summary form in a canvas app

Article • 12/16/2022

Follow step-by-step instructions to create a summary form in a canvas app for managing fictitious data in the Northwind Traders database. This topic is part of a series that explains how to build a business app on relational data in Microsoft Dataverse. For best results, explore these topics in this sequence:

1. Create an order gallery.
2. Create a summary form (**this topic**).
3. Create a detail gallery.



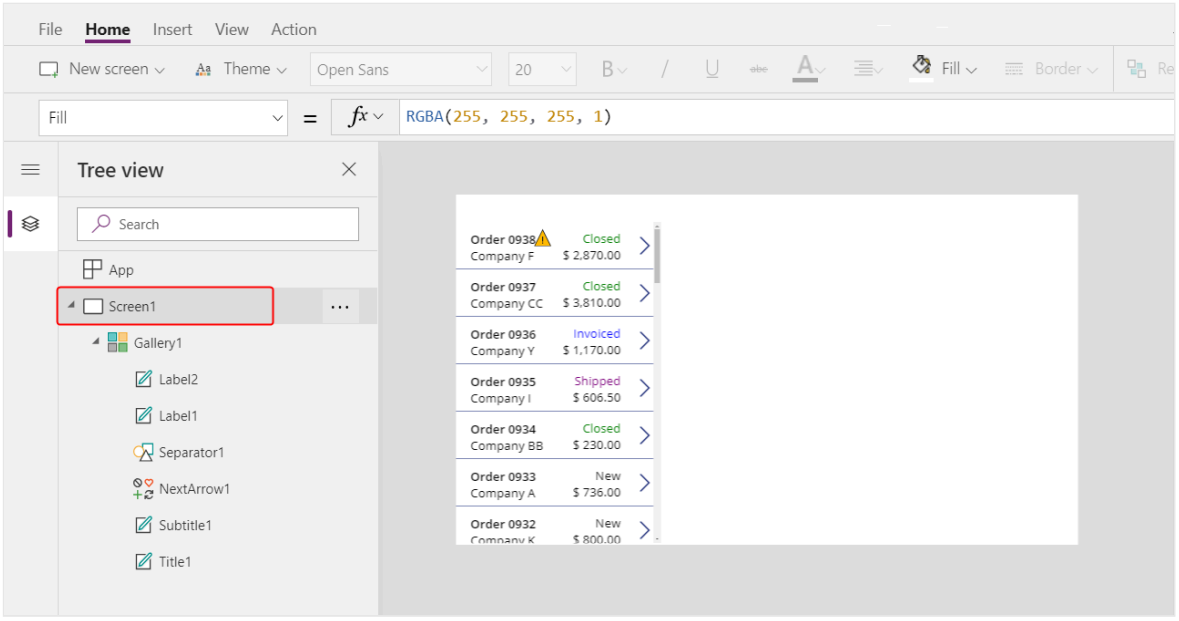
Prerequisites

1. Install the Northwind Traders database and apps.
2. Review the overview of the canvas app for Northwind Traders.
3. Create the order gallery yourself, or open the **Northwind Orders (Canvas) - Begin Part 2** app, which already contains that gallery.

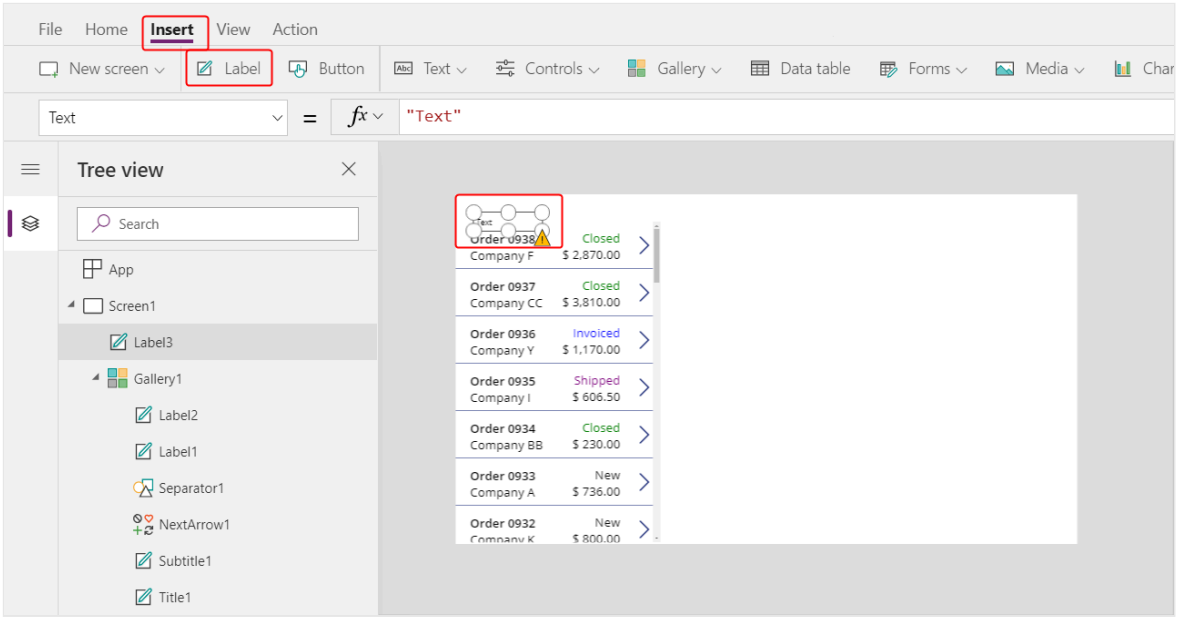
Add a title bar

Across the top of the app, create a title bar, which will hold action buttons by the end of this topic.

1. In the **Tree view** pane, select **Screen1** to ensure that you don't accidentally add a control to the order gallery:

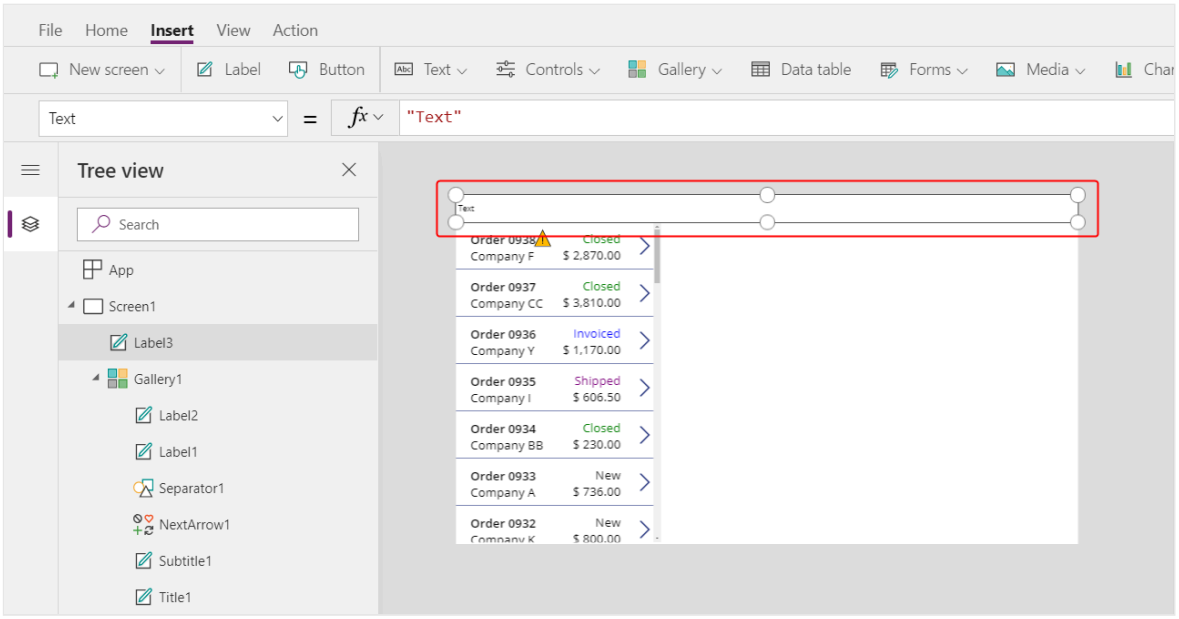


2. On the **Insert** tab, select **Label** to insert a Label control:



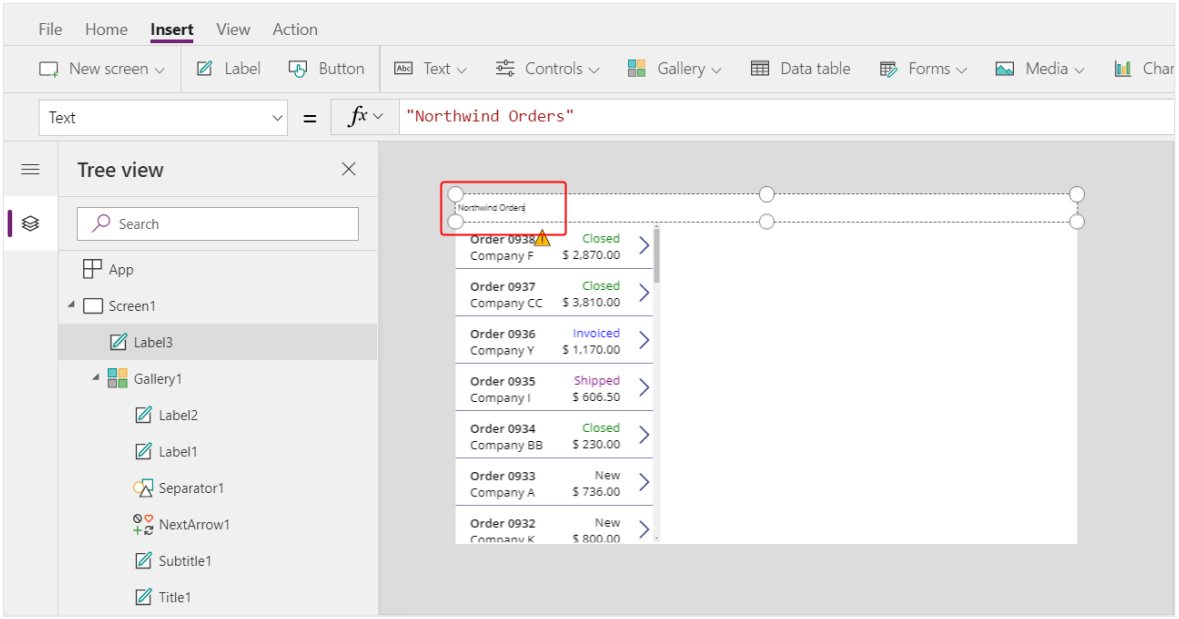
   The new label should appear only once, above the gallery. If it appears in each item of the gallery, delete the first instance of the label, ensure that the screen is selected (as the previous step describes), and then insert the label again.

3. Move and resize the new label to span the top of the screen:



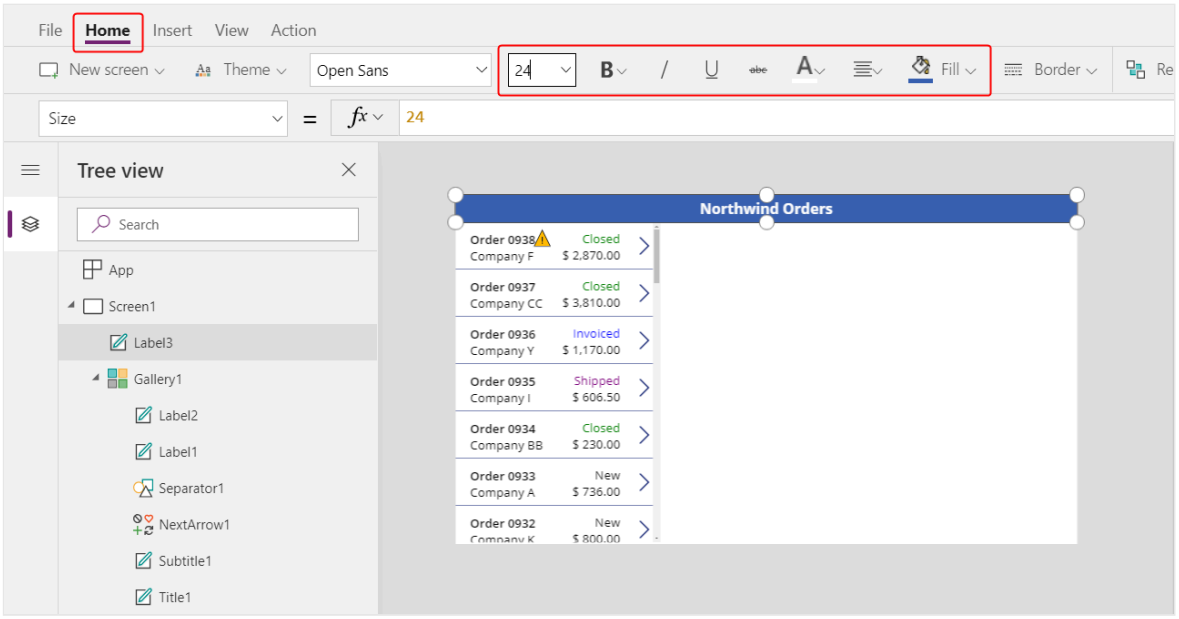
4. Double-click the text of the label, and then type **Northwind Orders**.

   As an alternative, modify the **Text** property in the formula bar to achieve the same result:



5. On the **Home** tab, format the label:

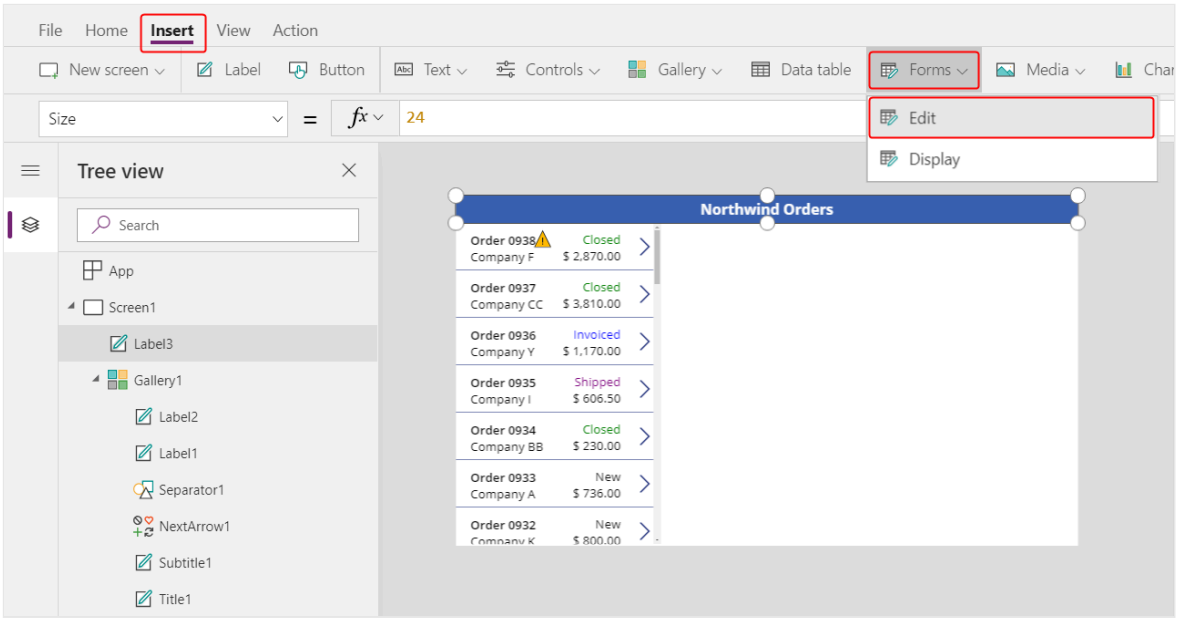
   - Increase the font size to 24 points.
   - Make the text bold.
   - Make the text white.
   - Center the text.
   - Add a dark-blue fill to the background.



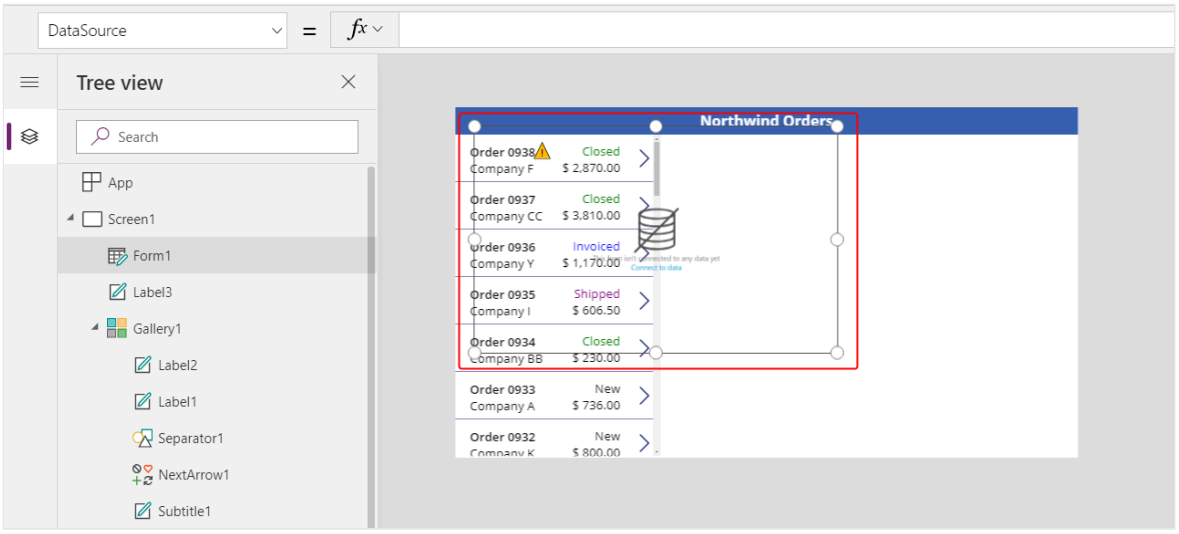
Add an Edit form control

In this section, you'll add controls to show a summary of any order that the user selects in the gallery.

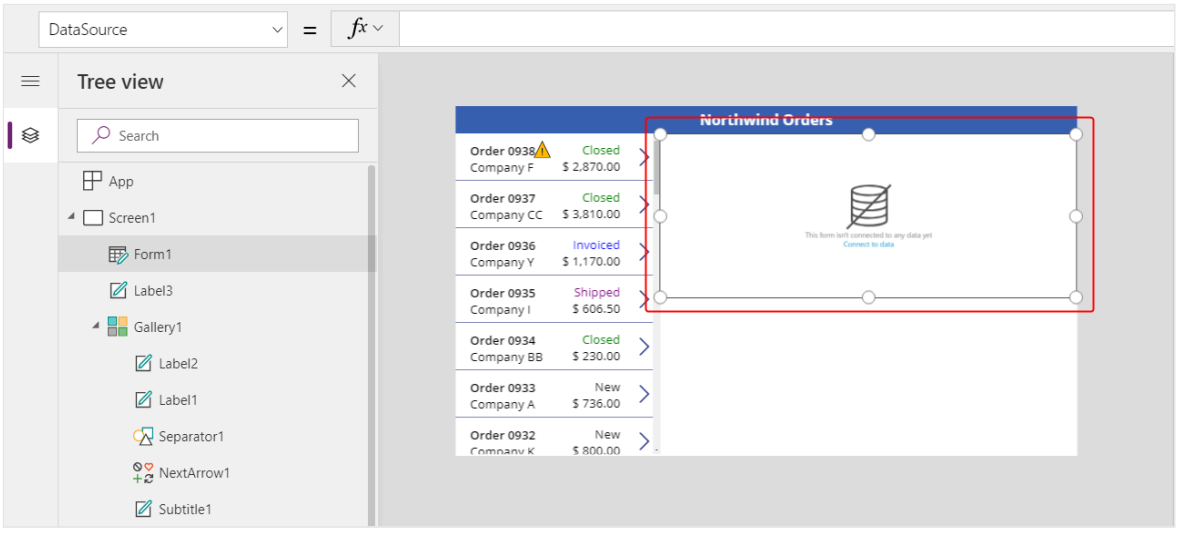
1. On the **Insert** tab, insert an **Edit form** control:



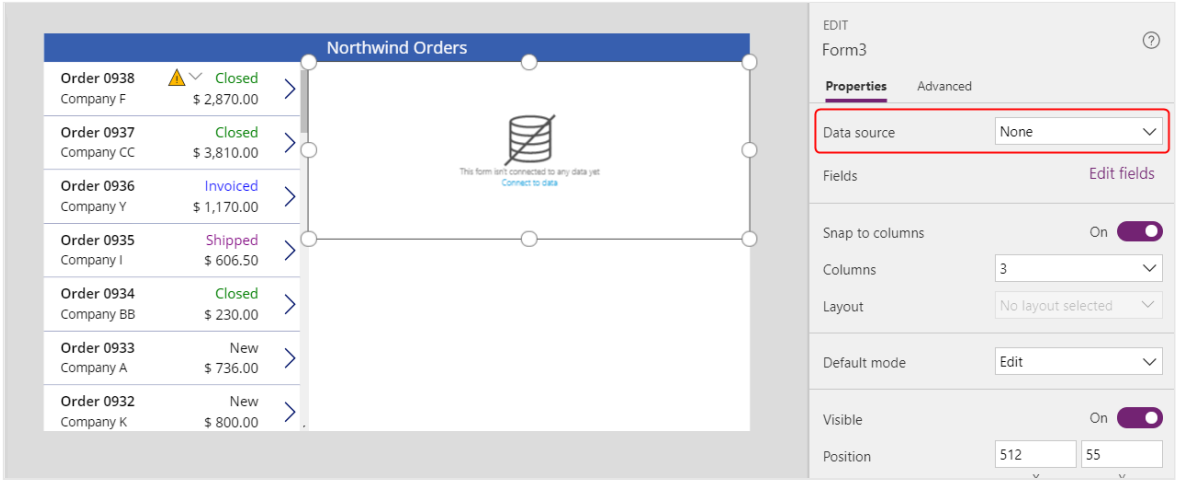
   By default, the form appears in the upper-left corner, where other controls might make it difficult to find:



2. Move and resize the form to cover the upper-right corner of the screen under the title bar:



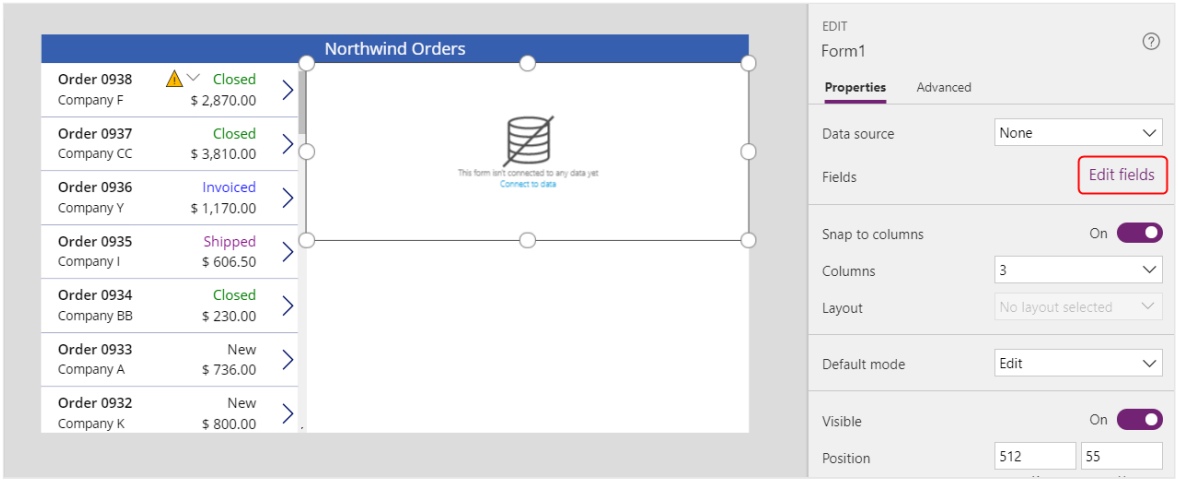
3. In the **Properties** pane, select the **Data source** drop down.



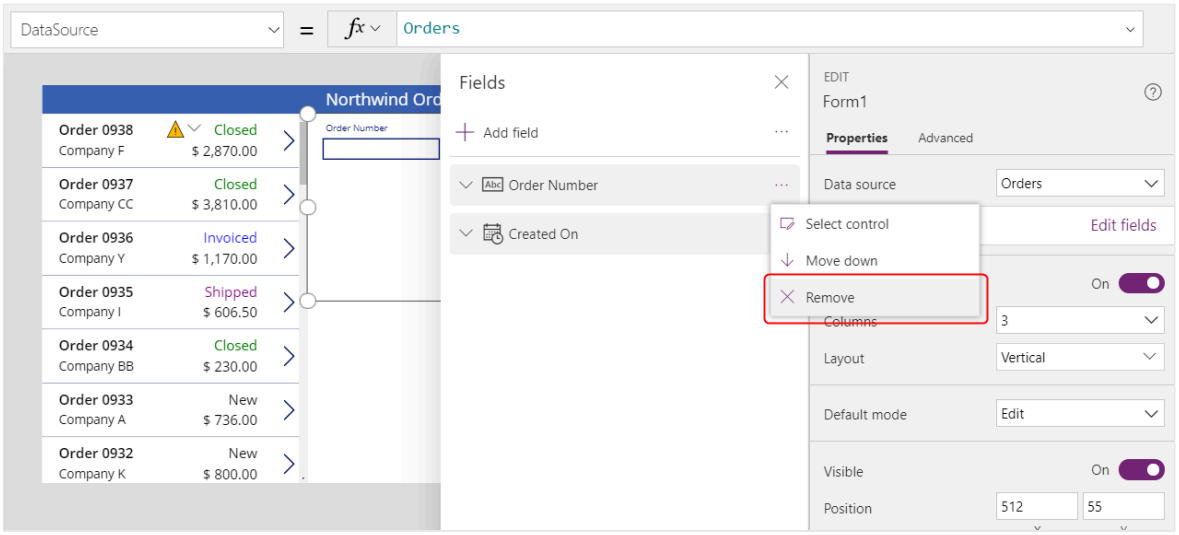
4. Select the **Orders** data source.

Add and arrange fields

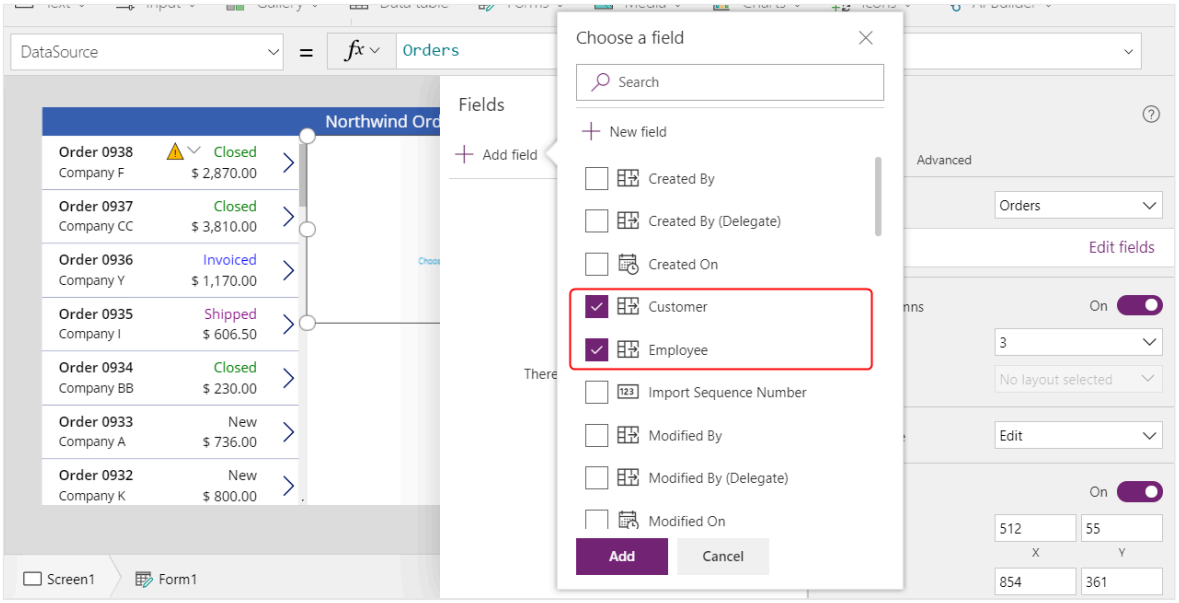
1. In the **Properties** tab near the right edge, select **Edit fields** to open the **Fields** pane.



2. If the **Fields** pane is not empty, remove the fields that have already been inserted.

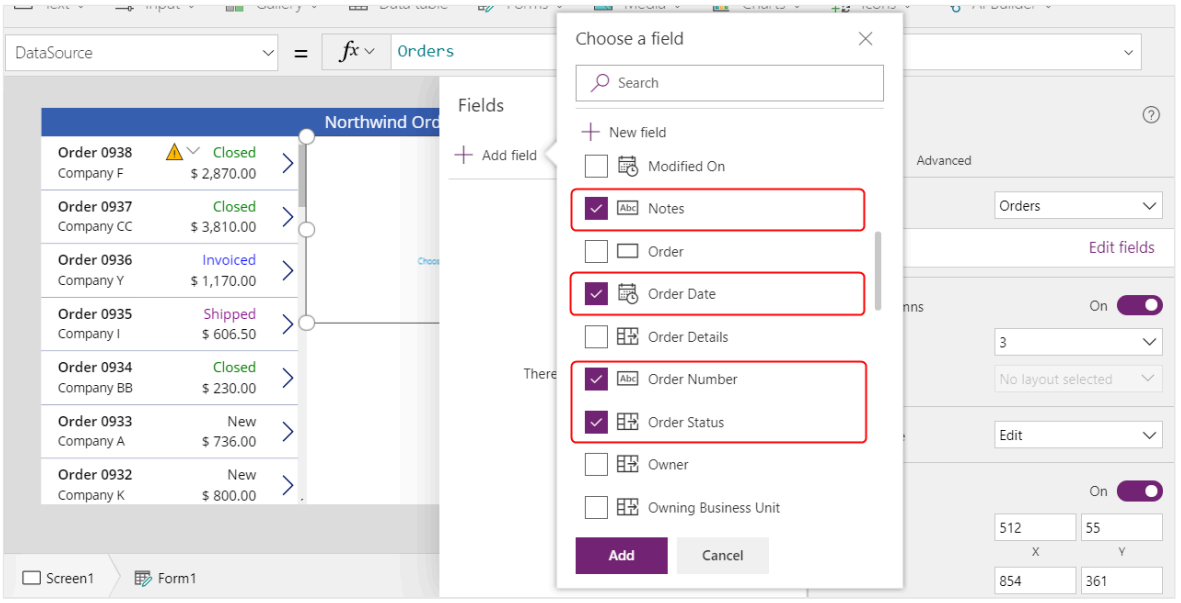


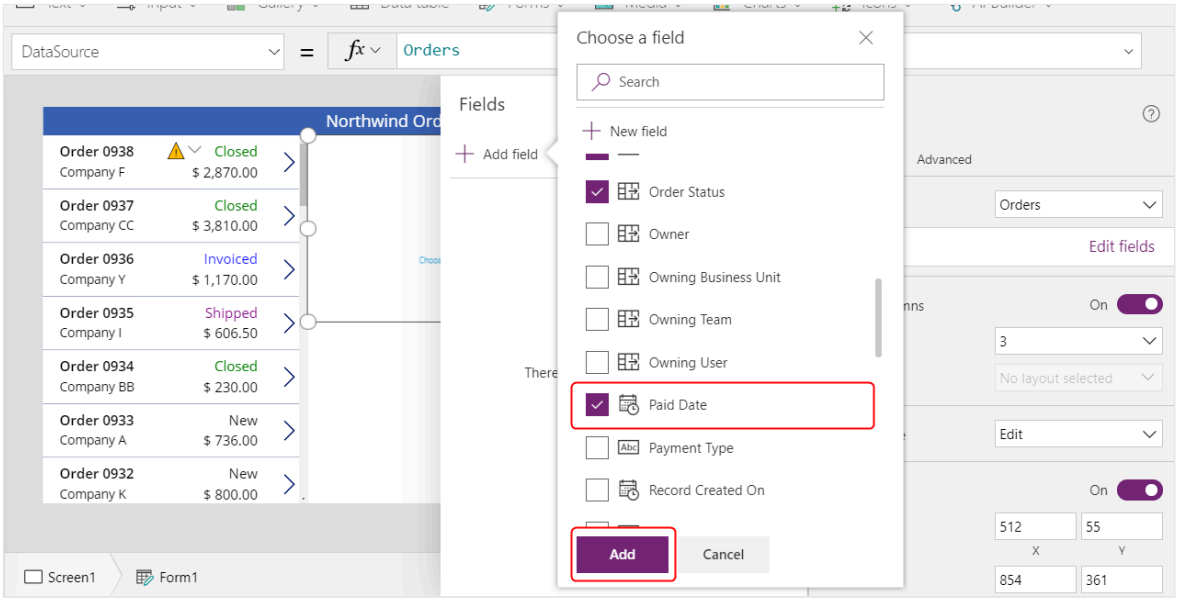
3. After the fields list is empty, select **Add field**, and then select the check boxes for the **Customer** and **Employee** fields.



4. Scroll down until these fields appear, and then select their check boxes:

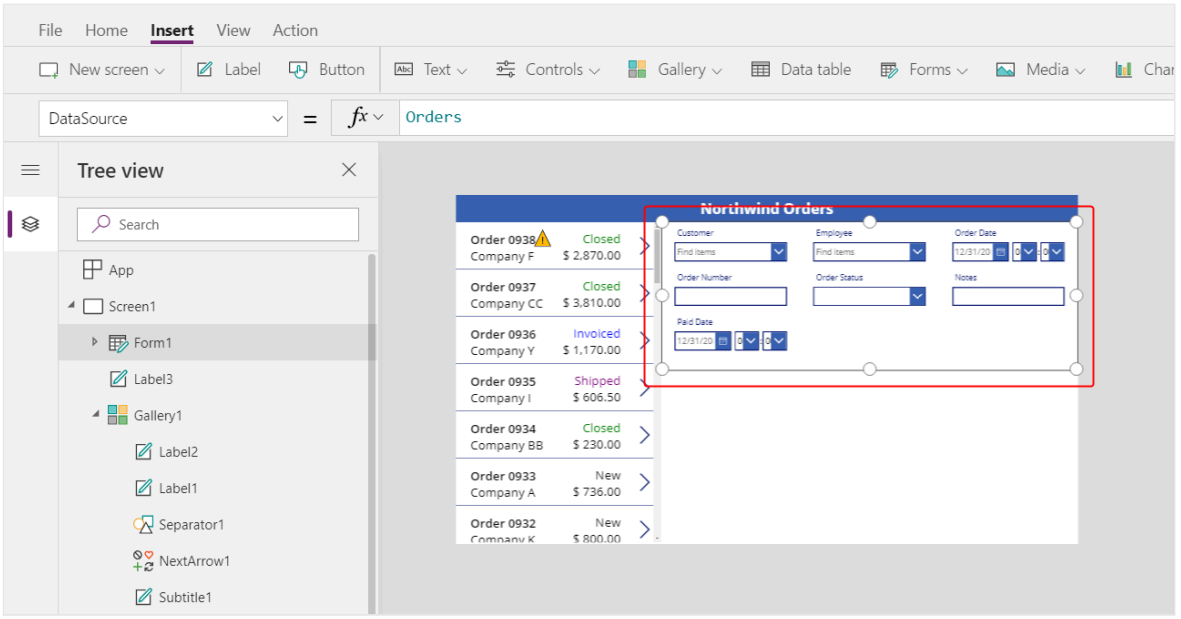
   - **Notes**
   - **Order Date**
   - **Order Number**
   - **Order Status**
   - **Paid Date**





5. At the bottom of the **Fields** pane, select **Add**, and then close the **Fields** pane.

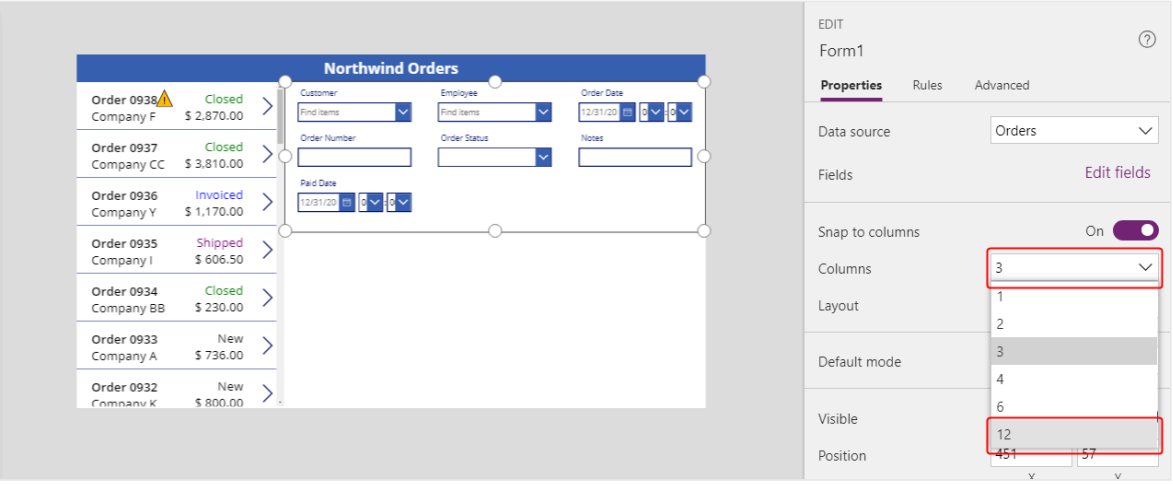
   The form shows seven fields, which may be in a different order:



> **Note**
>
> If any field shows a red error icon, a problem might have occurred when data was pulled from the source. To resolve the error, refresh the data:
>
> a. On the **View** tab, select **Data sources**.
>
> b. In the **Data** pane, select **Data sources**.
>
> c. Next to **Orders**, select the ellipsis (...), select **Refresh**, and then close the **Data** pane.
>
> If the combo box for the customer or employee name still shows an error, check the **Primary text** and **SearchField** of each box by selecting it and then opening the **Data** pane. For the customer box, both fields should be set to **nwind_company**. For the employee box, both fields should be set to **nwind_lastname**.

6. With the form selected, change the number of columns in the form from 3 to 12 in the **Properties** tab near the right edge.

   This step adds flexibility as you arrange the fields:

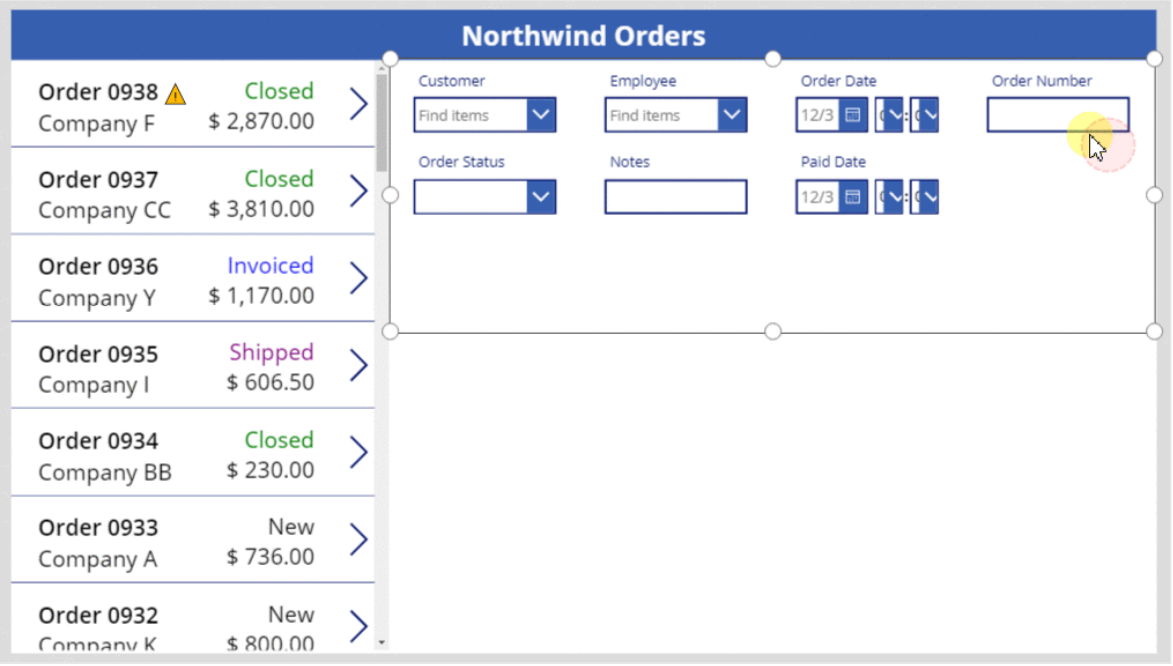


Many UI designs rely on 12-column layouts because they can evenly accommodate rows of 1, 2, 3, 4, 6, and 12 controls. In this topic, you'll create rows that contain 1, 2, or 4 controls.

7. Move and resize the fields by dragging their handles, just as you would any other control, so that each row contains these data cards in the specified order:

   - First row: **Order Number**, **Order Status**, **Order Date**, and **Paid Date**
   - Second row: **Customer** and **Employee**
   - Third row: **Notes**

> **Note**
>
> You might find it easier to widen the **Notes**, **Customer**, and **Employee** data cards before you arrange them.



More information about how to arrange fields in a form: Understand data-form layout for canvas apps.

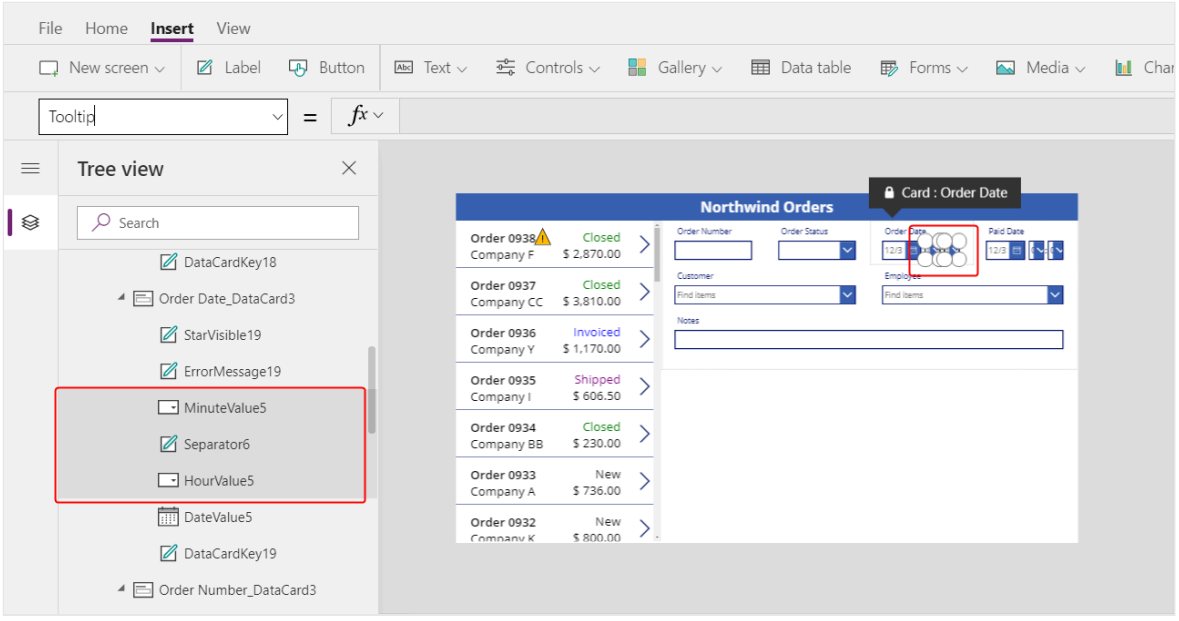
Hide time controls

In this example, you don't need the time portions of the date fields because that level of granularity can distract the user. If you delete them, you might cause problems in formulas that rely on those controls to update date values or determine the position of another control in the data card. Instead, you'll hide the time controls by setting their **Visible** property.

1. In the **Tree view** pane, select the **Order Date** data card.

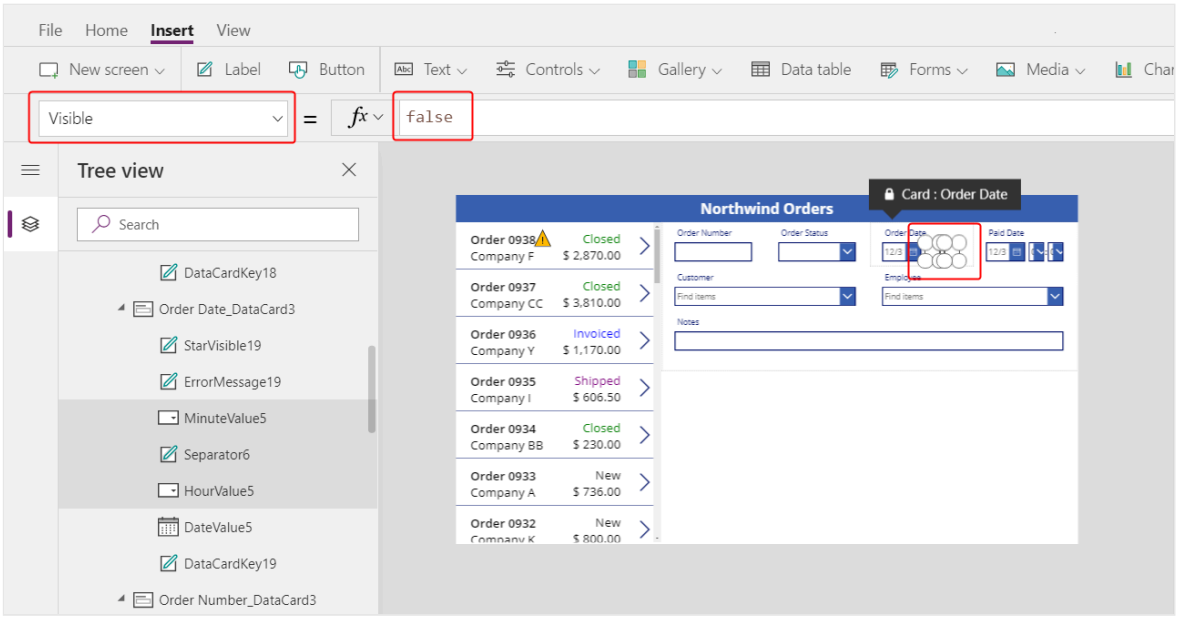
   The card might have a different name, but it contains **Order Date**.

2. While holding down the Shift key, select the hour, minute, and colon-separator controls in the **Order Date** data card.

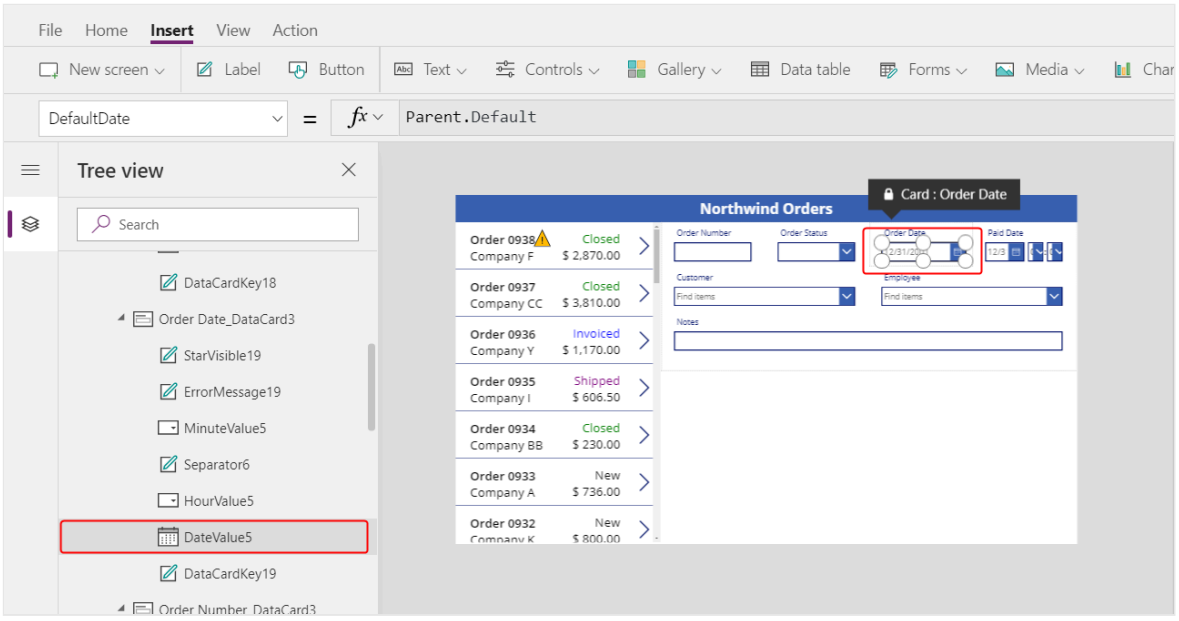


3. Set the controls' **Visible** property to **false**.

   All selected controls disappear from the form:

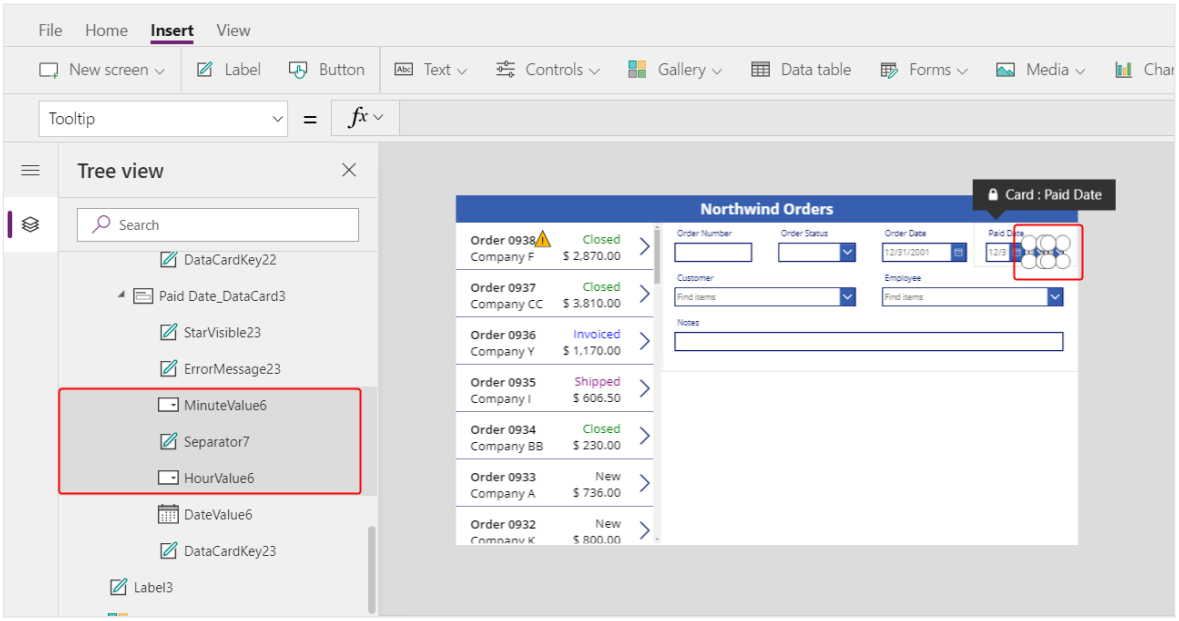


4. Resize the **Date picker** control to show the complete date:

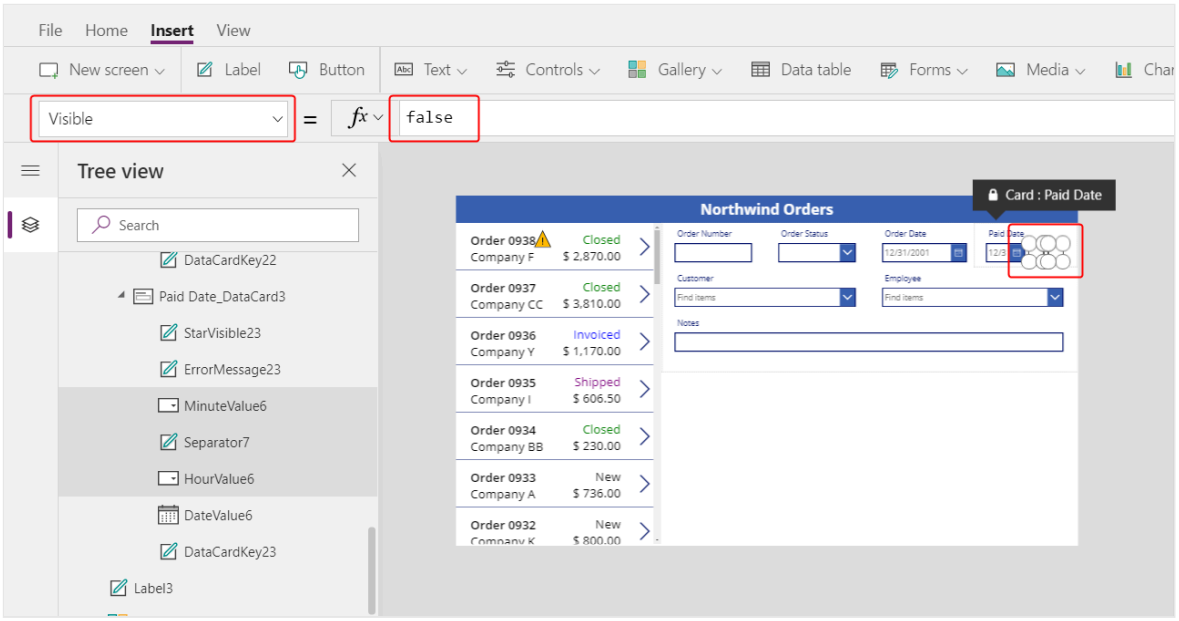


Next, you'll repeat the last few steps for the **Paid Date** field.

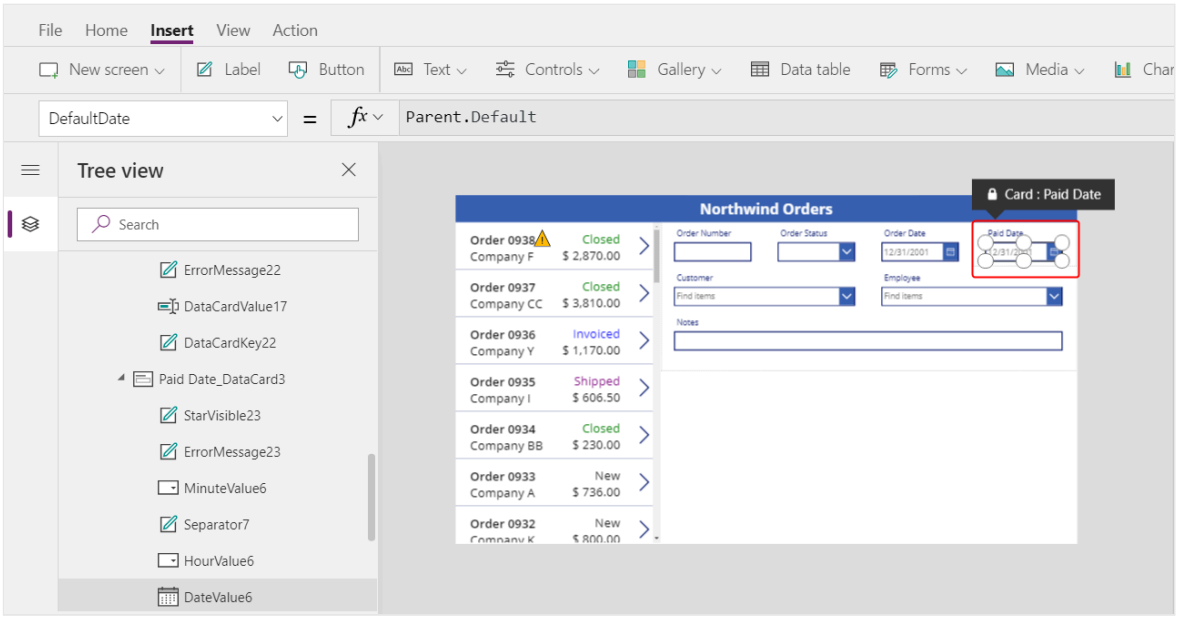
5. In the **Tree view** pane, select the time controls in the **Paid Date** data card:



6. Set the selected controls' **Visible** property to **false**:



7. Resize the date picker in the **Date Paid** card:

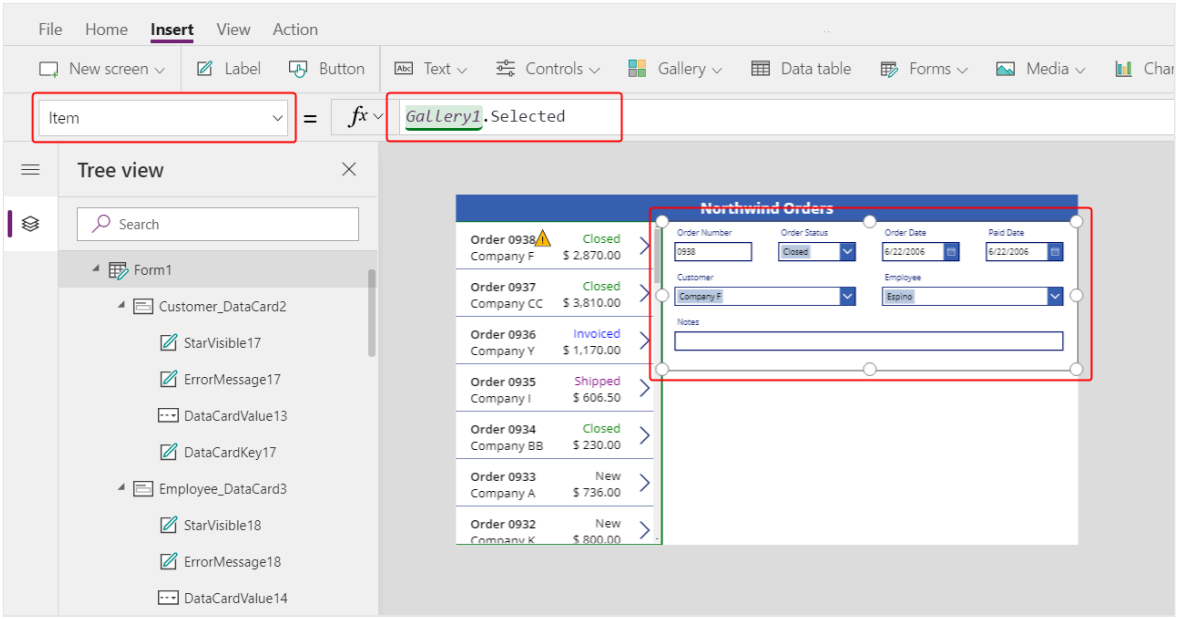


Connect the order gallery

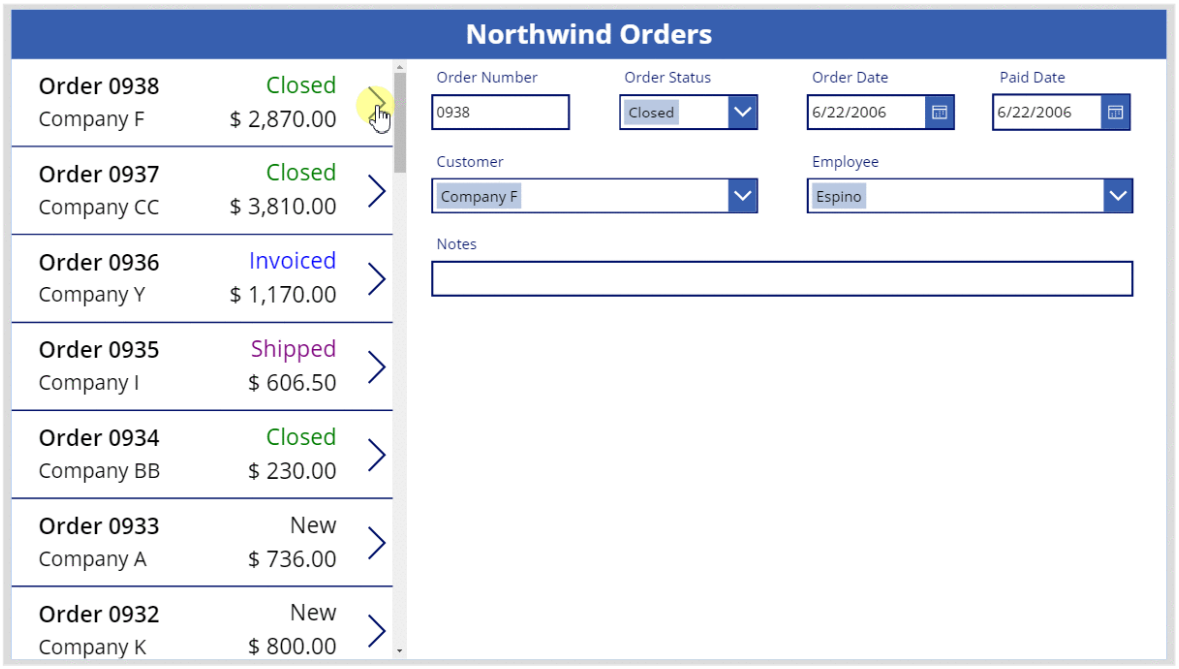
1. In the **Tree view** pane, collapse the form to more easily find the name of the order gallery, and then, if necessary, rename it to **Gallery1**.

2. Set the summary form's **Item** property to this expression:

```
Power Fx

Gallery1.Selected
```



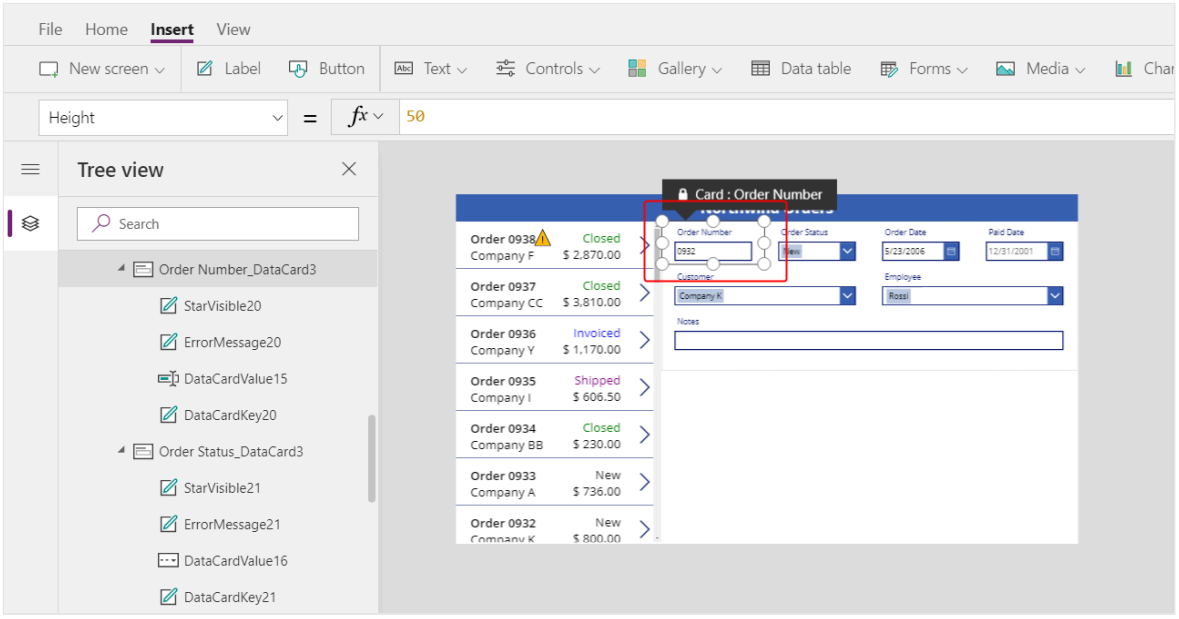
The form shows an summary of whatever order the app user selects in the list.



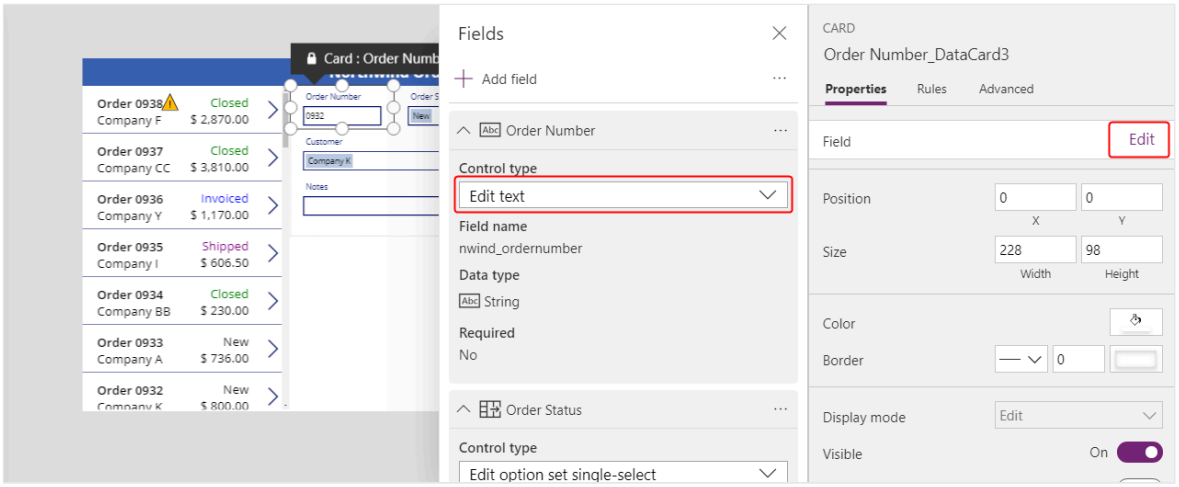
Replace a data card

Order number is an identifier that Dataverse assigns automatically when you create a record. This field has a **Text input** control by default, but you'll replace it with a label so that the user can't edit this field.

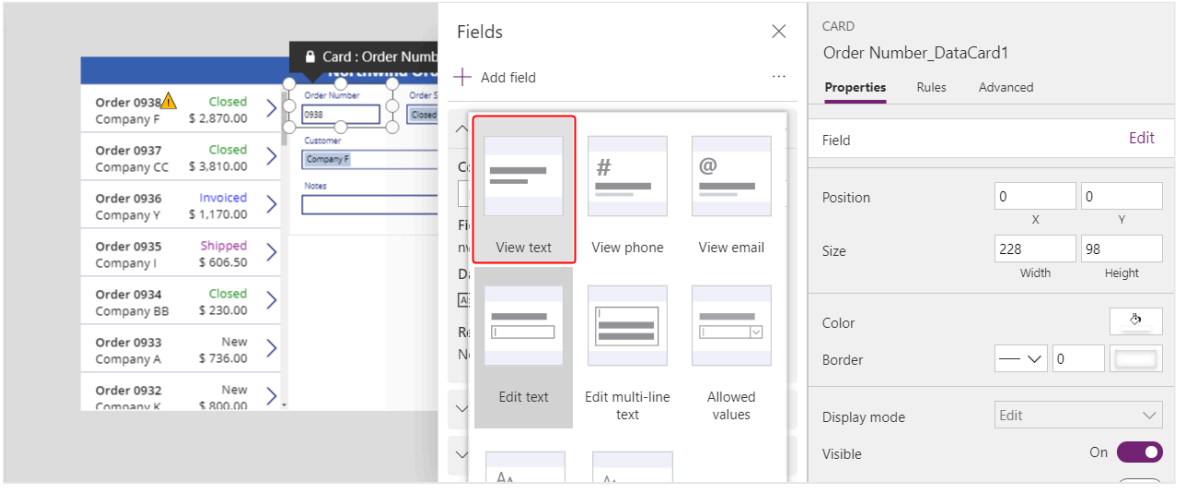
1. Select the form, select **Edit fields** in the **Properties** tab near the right edge, and then select the **Order number** field:



2. Open the **Control type** list:

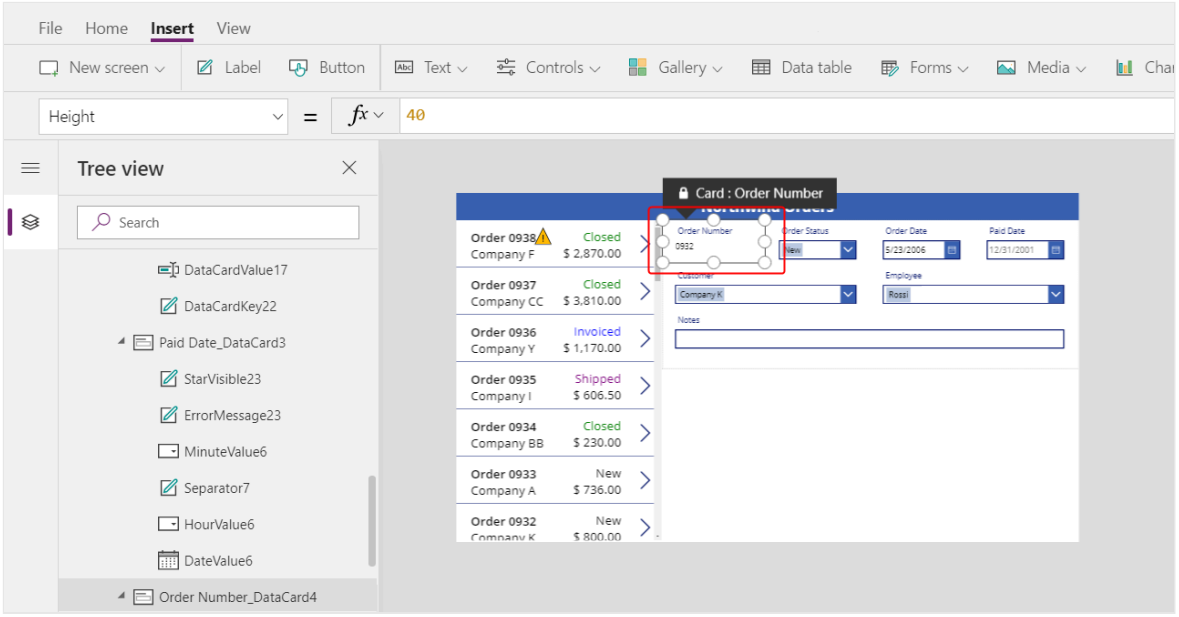


3. Select the **View text** data card:

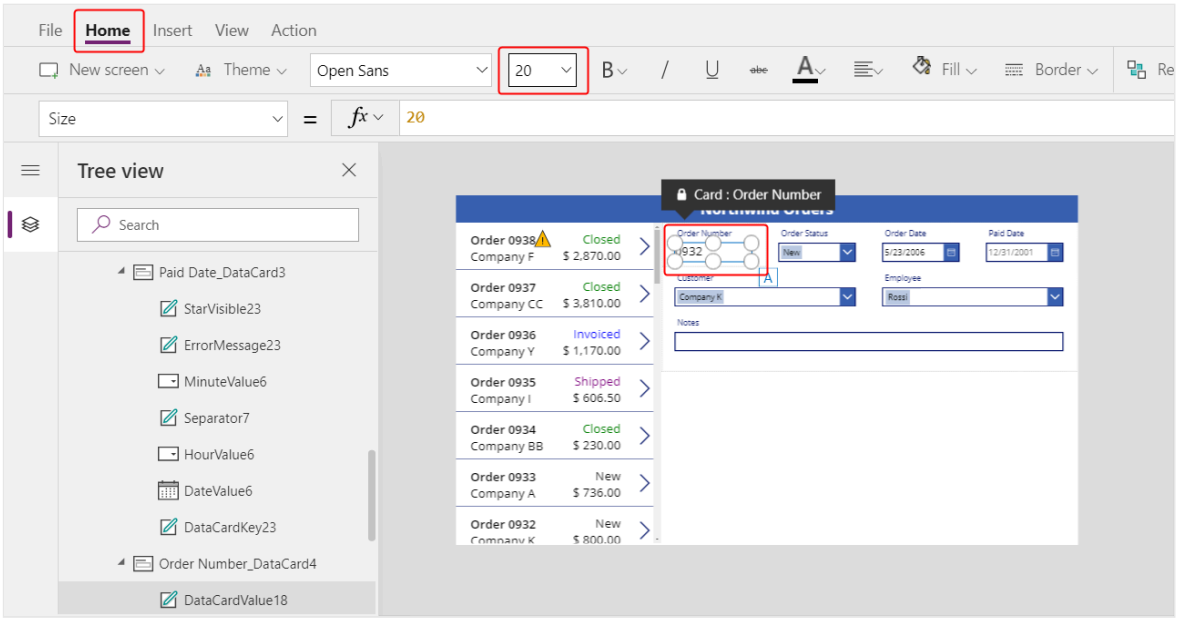


4. Close the **Fields** pane.

   The user can no longer change the order number:



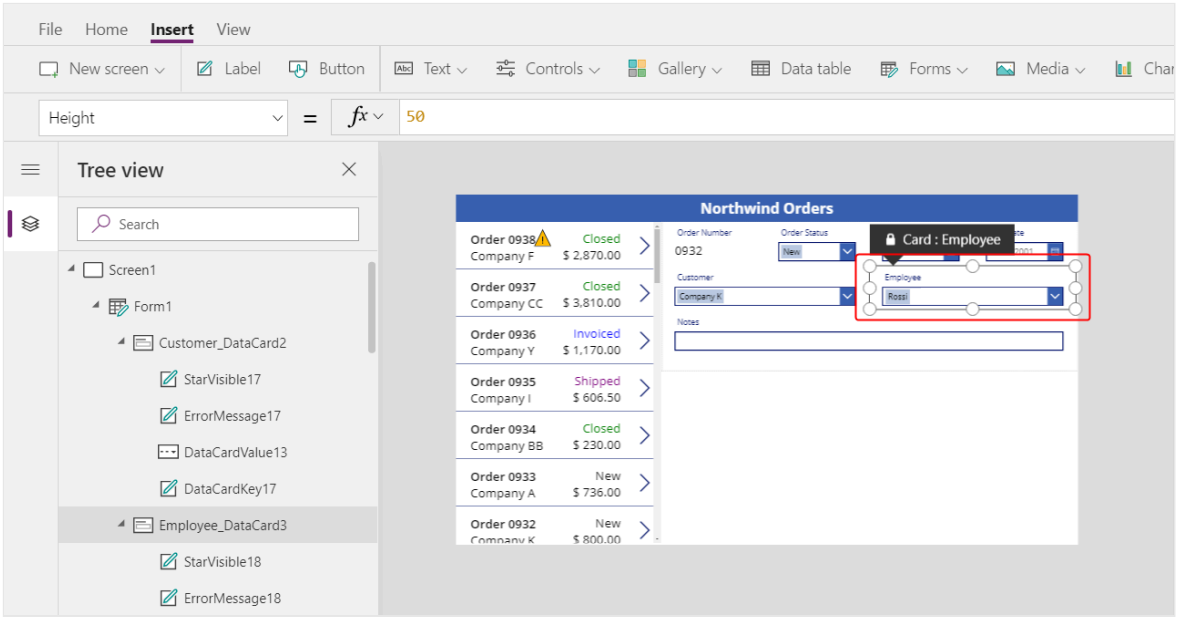
5. On the **Home** tab, change the order number's font size to 20 points so that the field is easier to find:



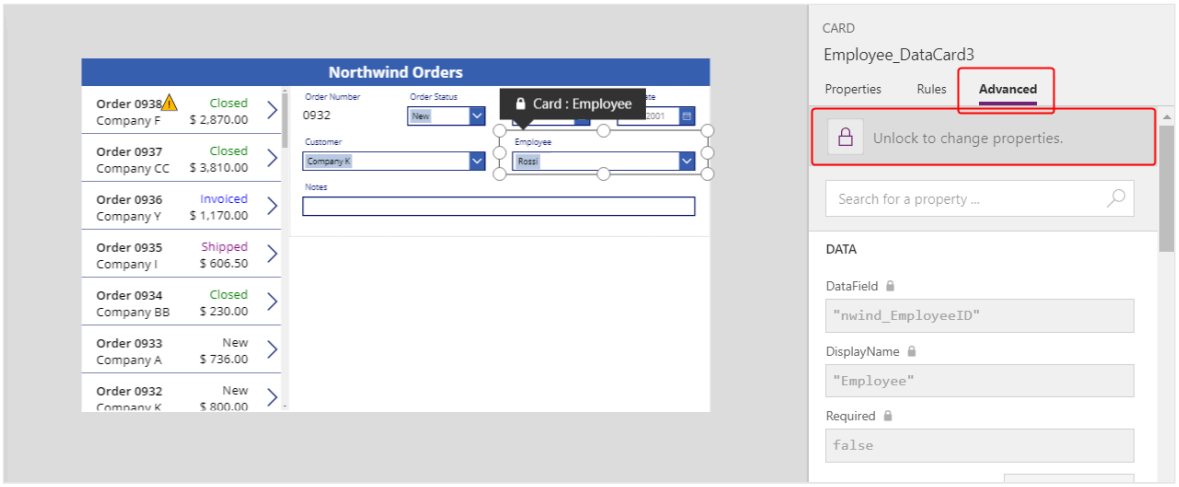
Use a many-to-one relationship

The **Orders** table has a many-to-one relationship with the **Employees** table: each employee can create many orders, but each order can be assigned to only one employee. When the user selects an employee in the Combo box control, its **Selected** property provides that employee's entire record from the **Employees** table. As a result, you can configure an Image control to show the picture of whatever employee the user selects in the combo box.

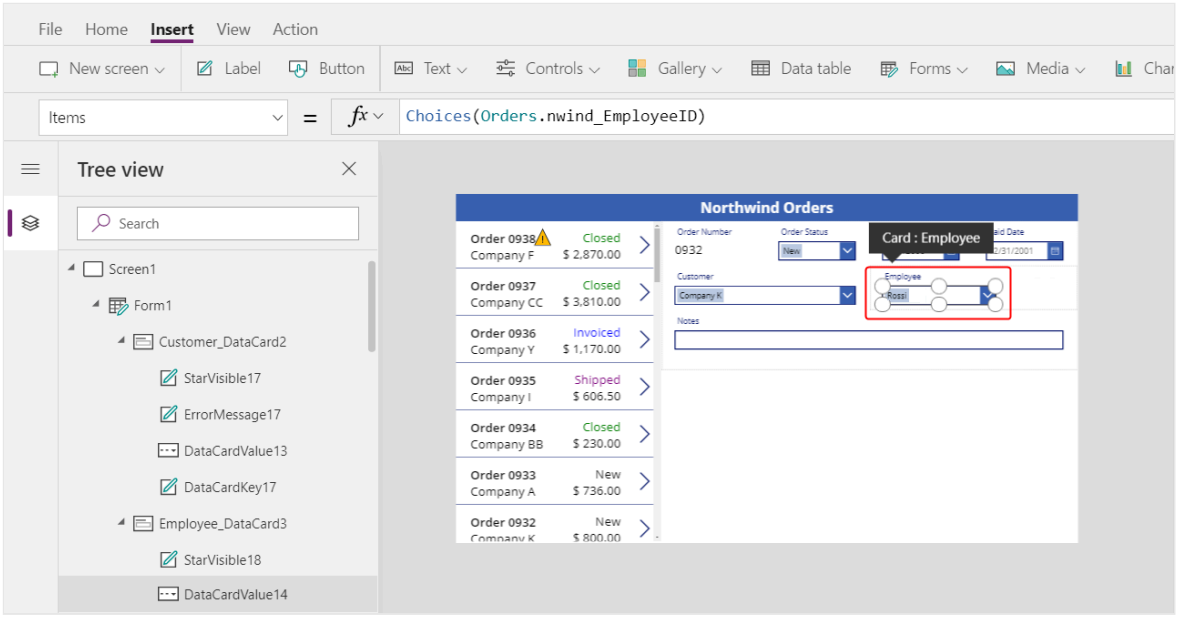
1. Select the **Employee** data card:



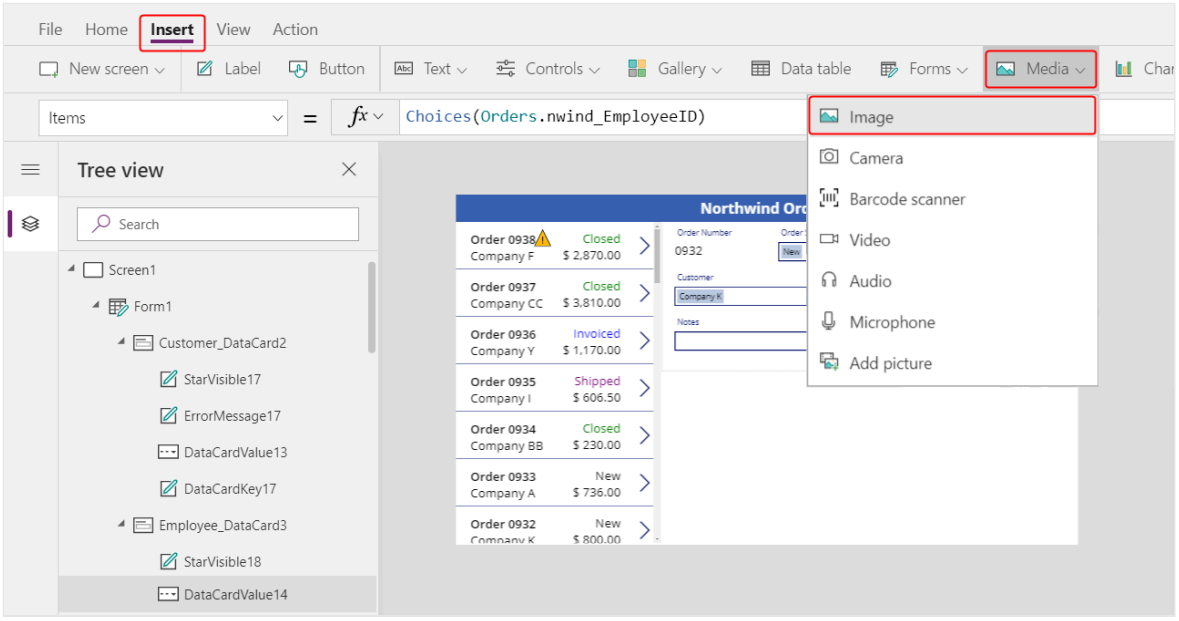
2. In the **Advanced** tab near the right edge, unlock the data card so that you can edit formulas that were previously read-only:



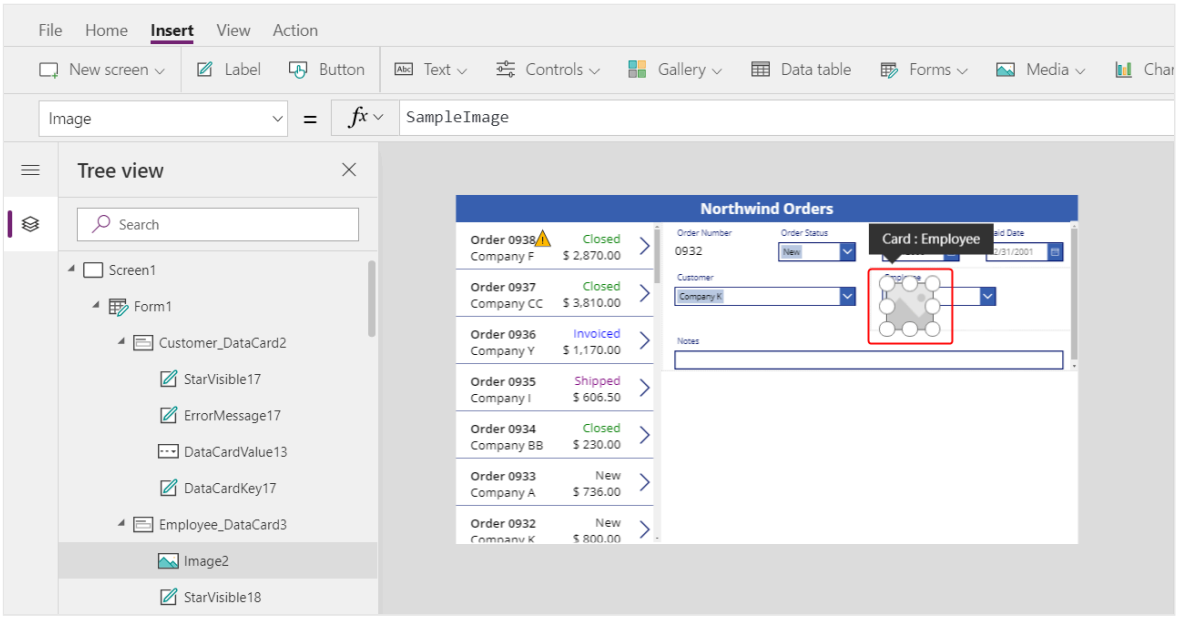
3. In the data card, reduce the width of the combo box to make room for the employee picture:



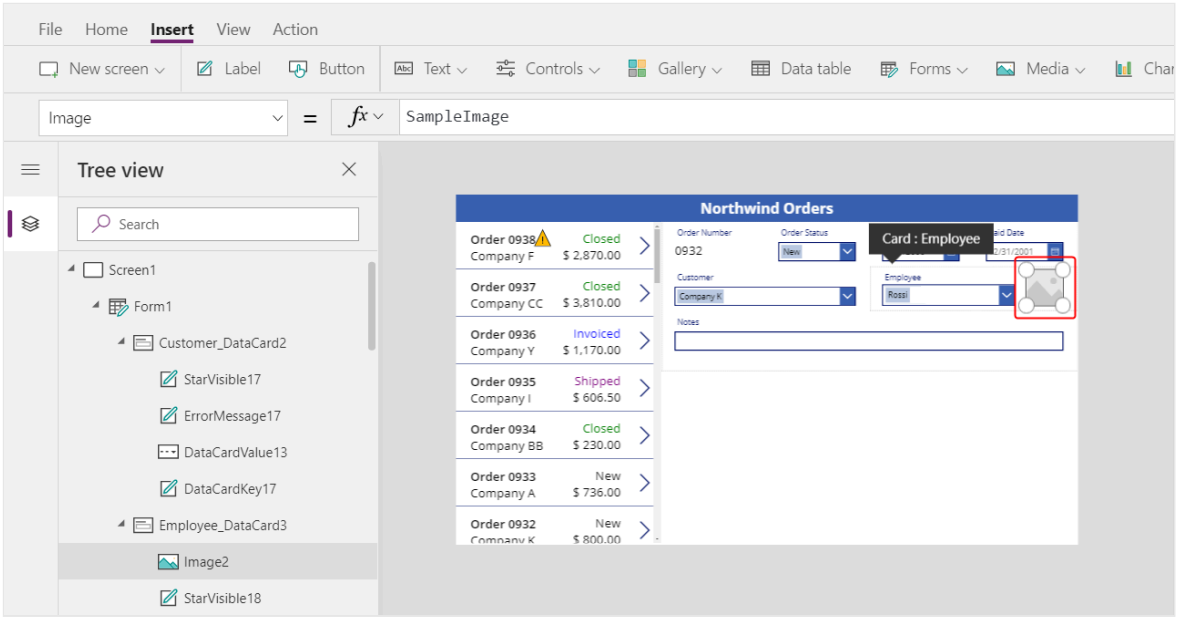
4. On the **Insert** tab, select **Media** > **Image**:



An image appears in the data card, which expands to accommodate it:



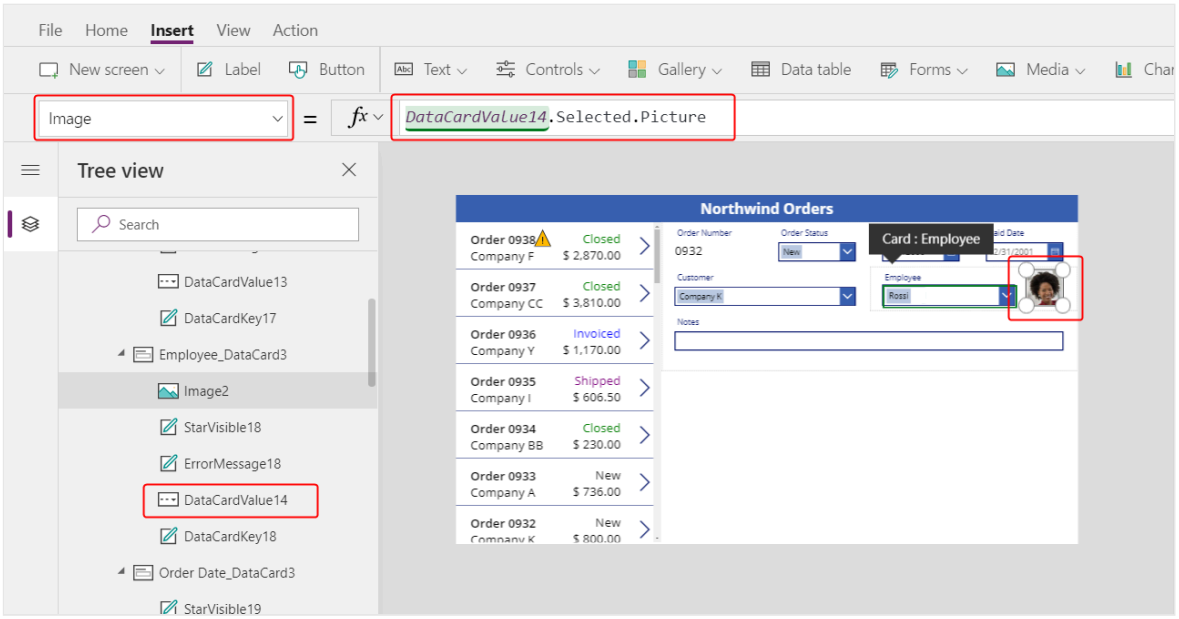
5. Resize the image, and move it to the right of the combo box:



6. Set the **Image** property of the image to this formula, replacing the number at the end of DataCardValue if necessary:

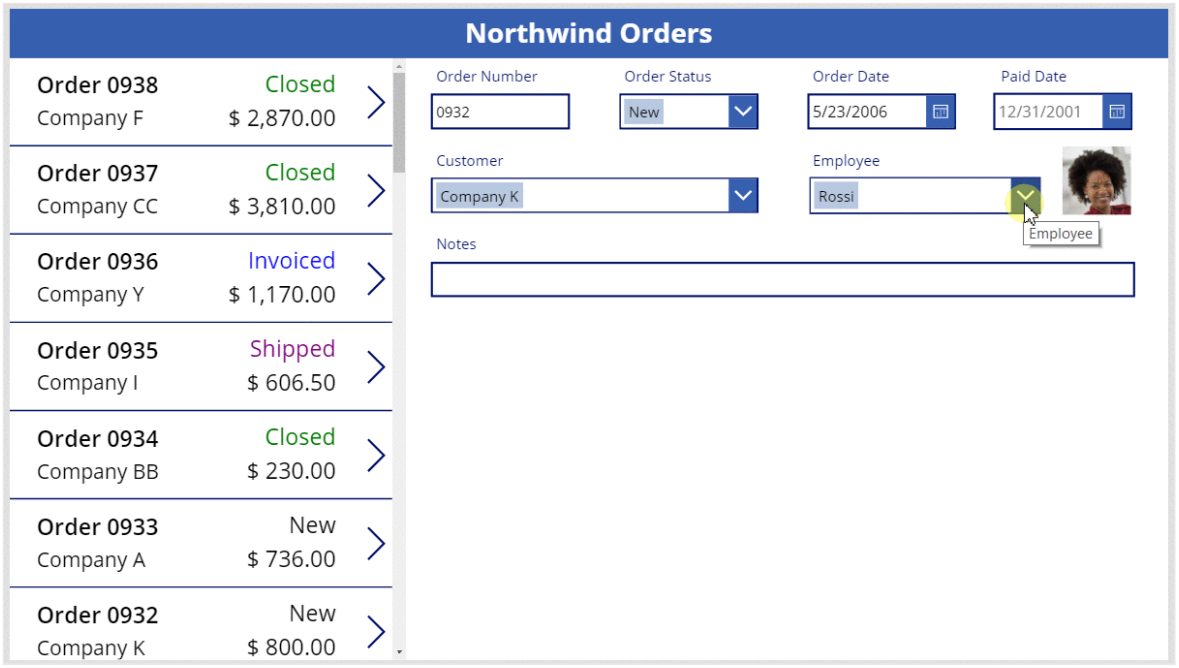
Power Fx

```
DataCardValue7.Selected.Picture
```



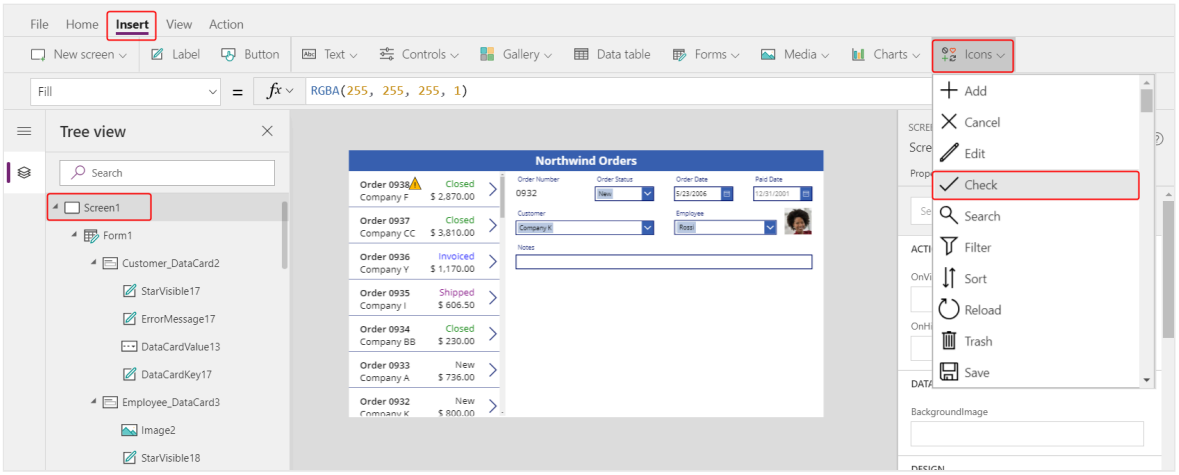
The picture of the selected employee appears.

7. While holding down the Alt key, select a different employee in the combo box to confirm that the picture also changes.

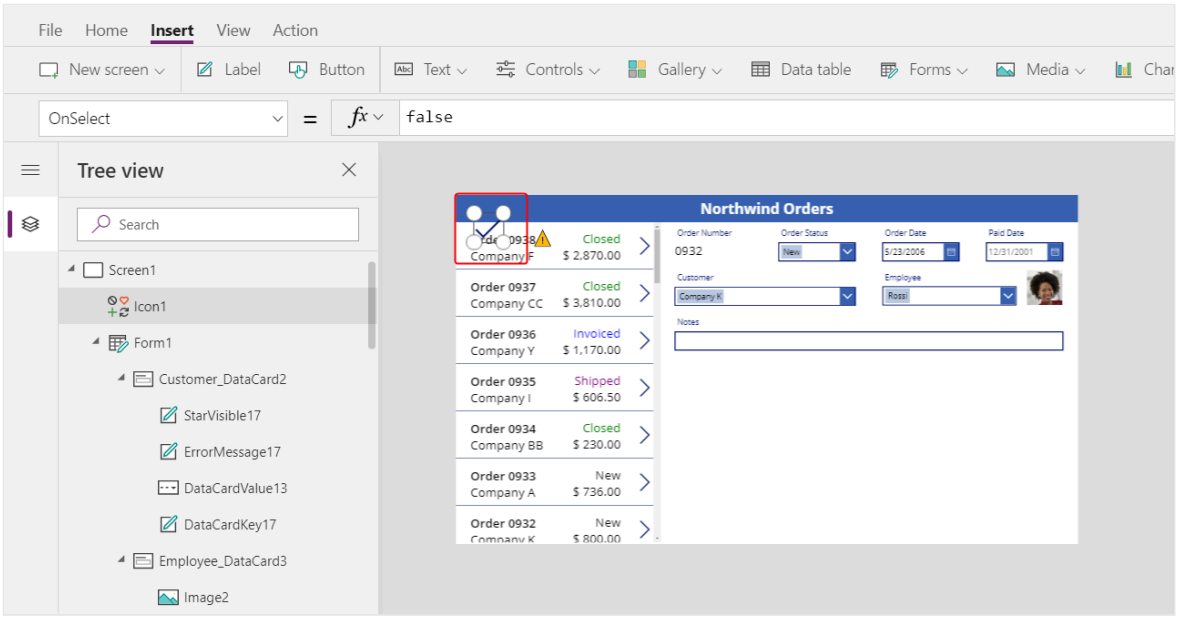


Add a Save icon

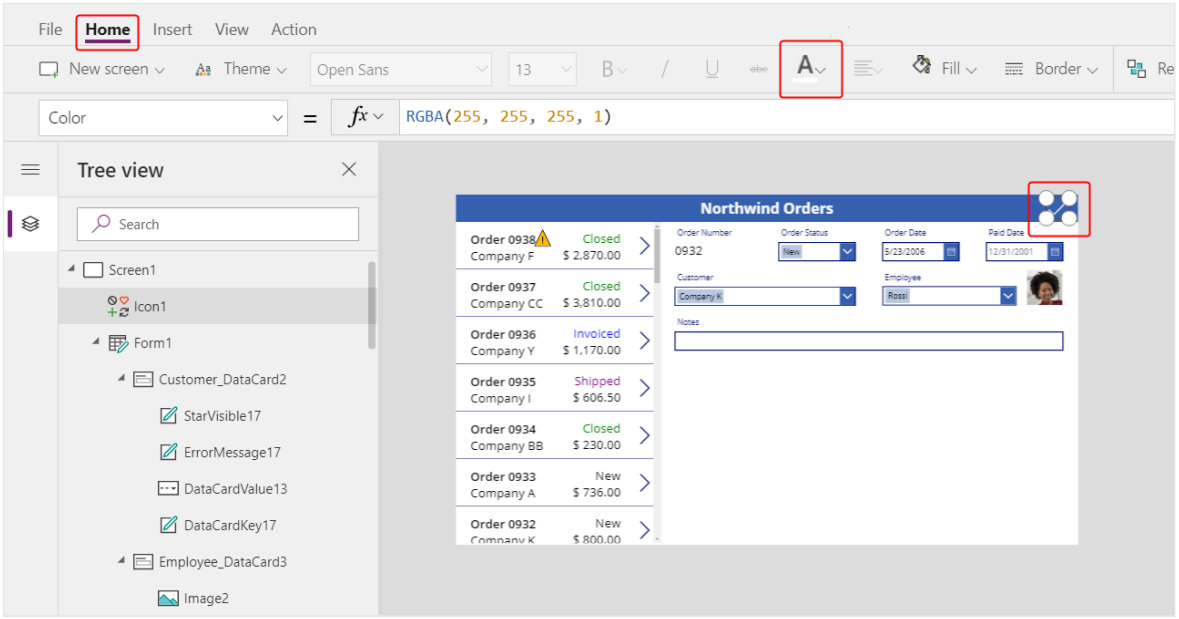
1. In the **Tree view** pane, select **Screen1**, and then select **Insert** > **Icons** > **Check**:



The **Check** icon appears in the upper-left corner by default, where other controls might make the icon difficult to find:

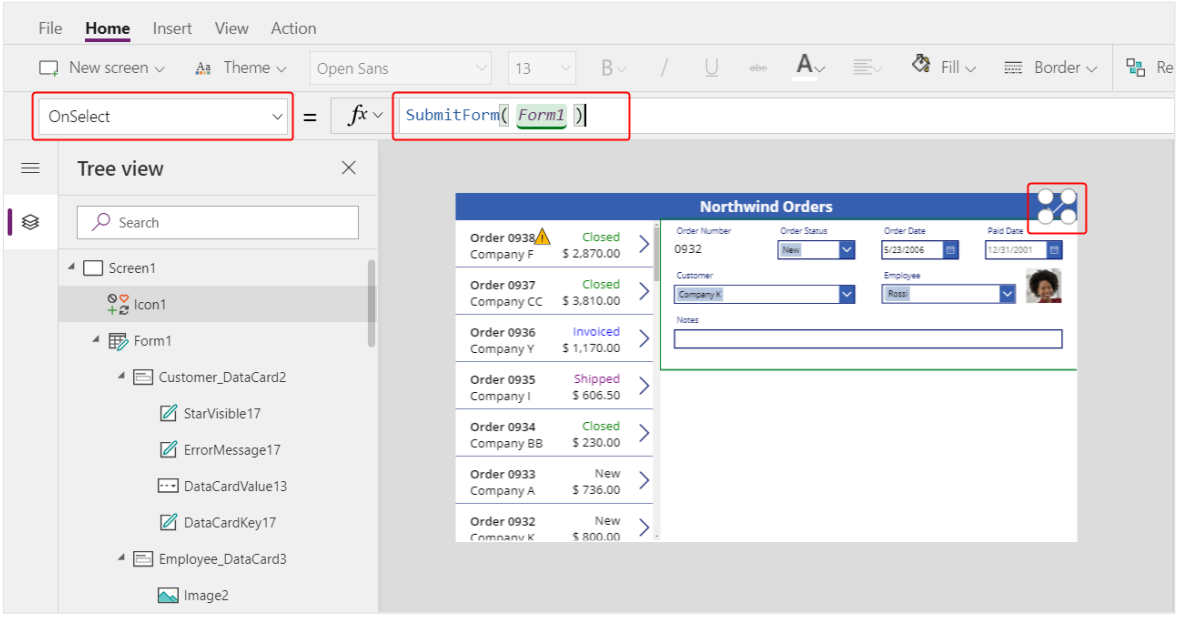


2. On the **Home** tab, change the **Color** property of the icon to white, resize the icon, and move it near the right edge of the title bar:



3. In the **Tree view** pane, confirm that the form's name is **Form1**, and then set the icon's **OnSelect** property to this formula:

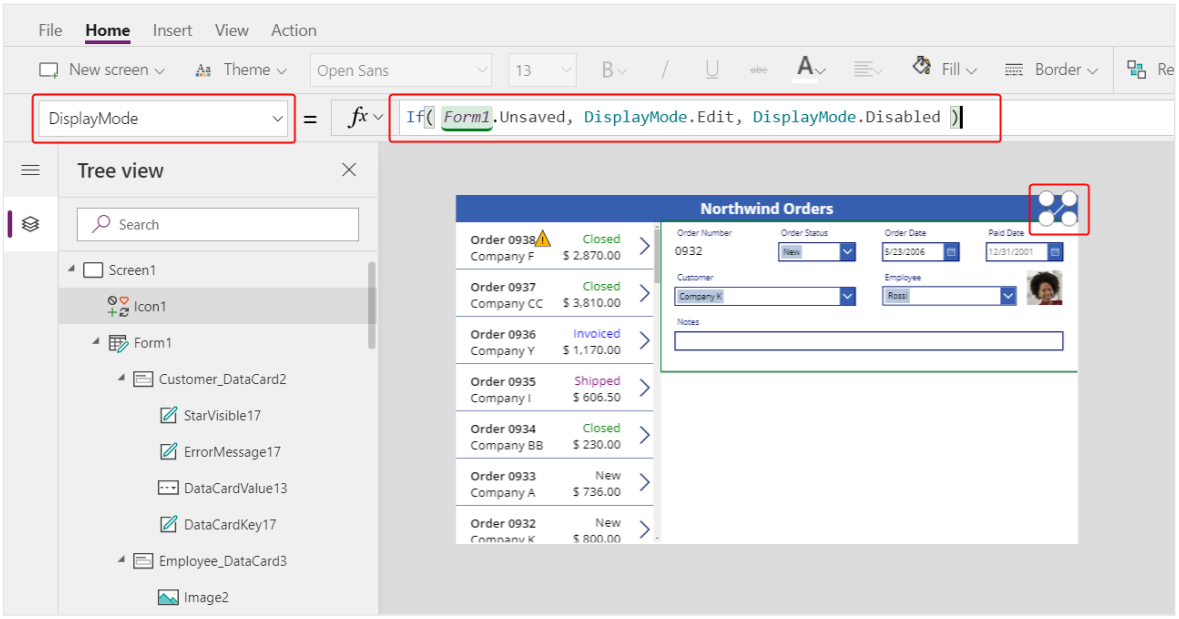
```
Power Fx

SubmitForm( Form1 )
```



When the user selects the icon, the **SubmitForm** function gathers any changed values in the form and submits them to the data source. Dots march across the top of the screen as the data is submitted, and the order gallery reflects the changes after the process finishes.

4. Set the icon's **DisplayMode** property to this formula:

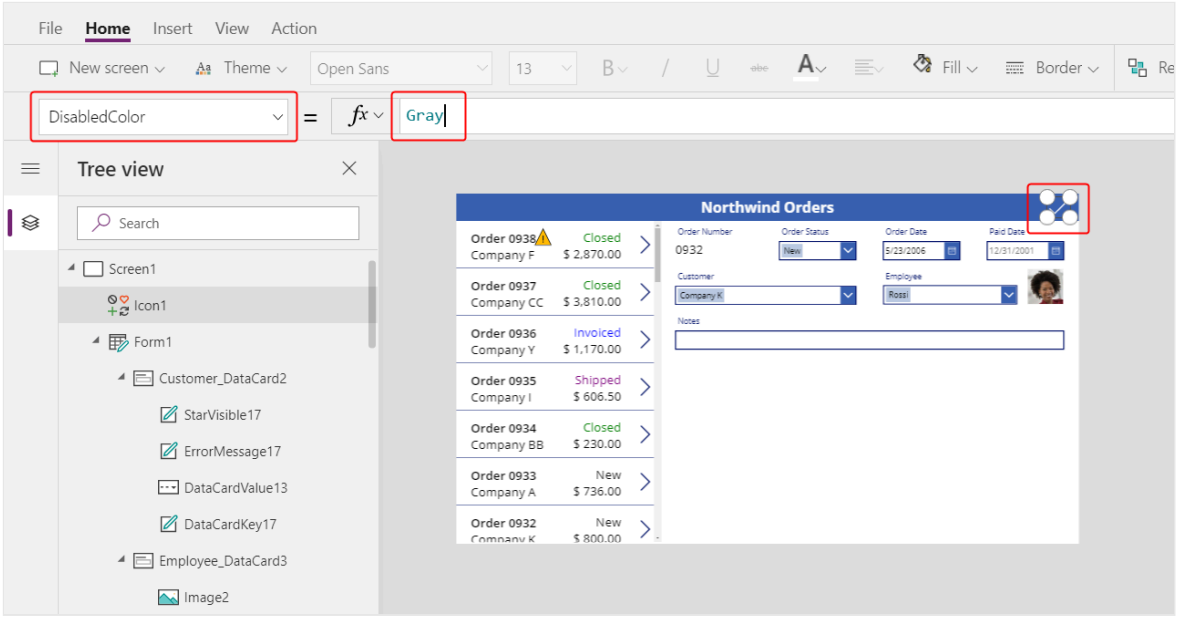
```
Power Fx

If( Form1.Unsaved, DisplayMode.Edit, DisplayMode.Disabled )
```



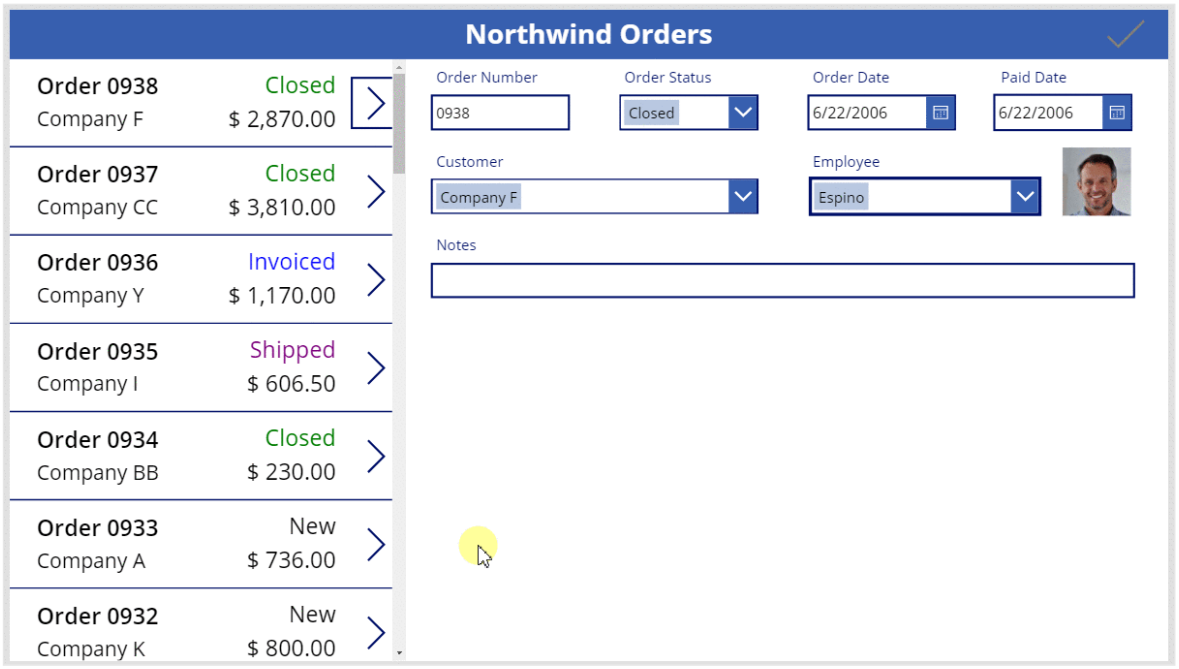
If all changes in the form have been saved, the icon is disabled and appears in the **DisabledColor**, which you'll set next.

5. Set the icon's **DisabledColor** property to this value:

```
Power Fx

Gray
```

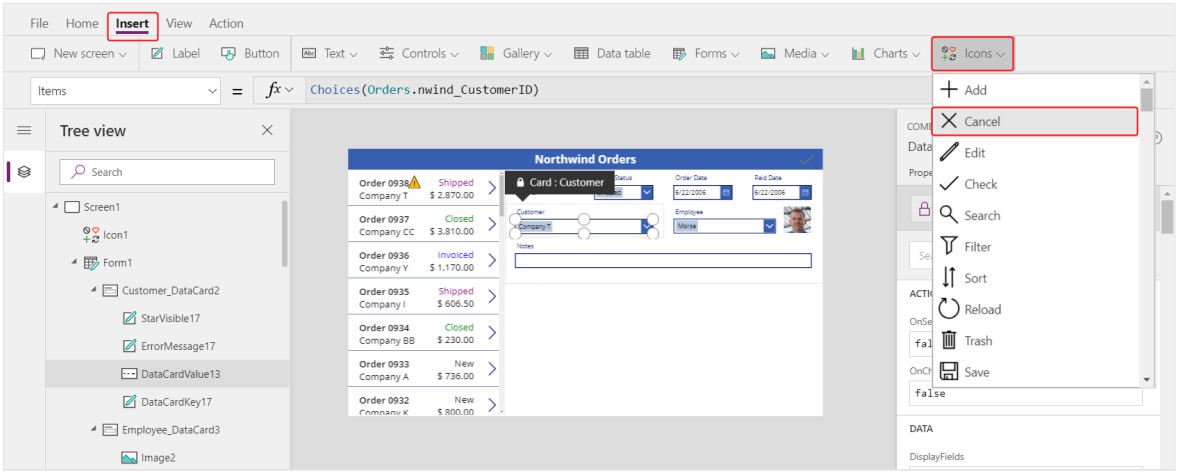


The user can save changes to an order by selecting the Check icon, which is then disabled and dimmed until the user makes another change:

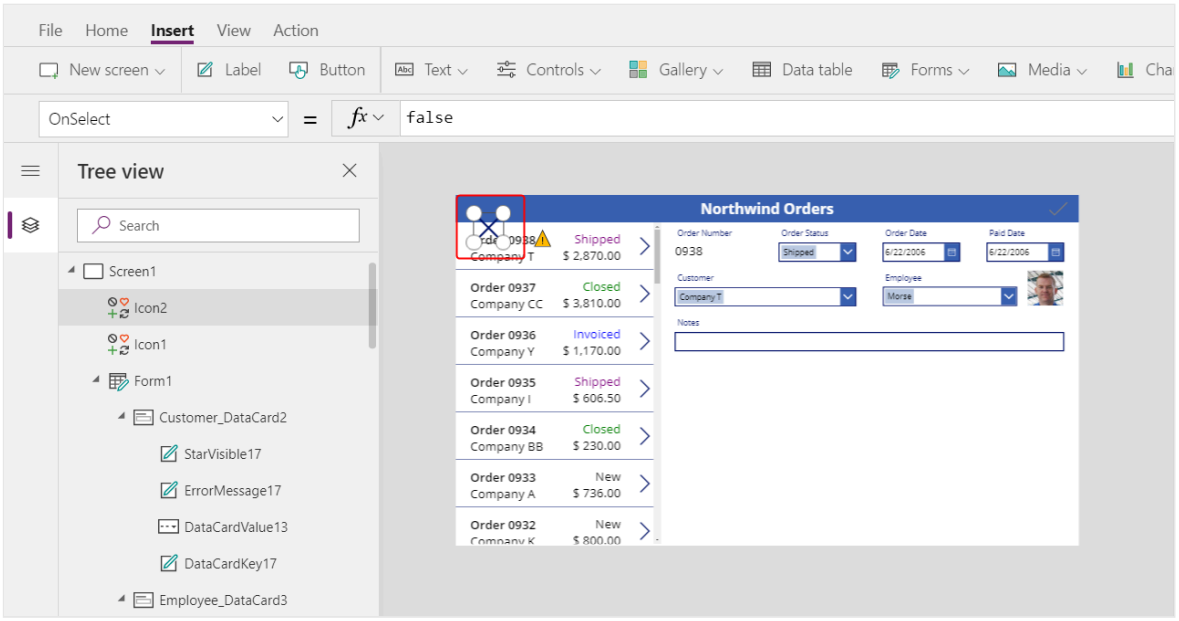


Add a Cancel icon

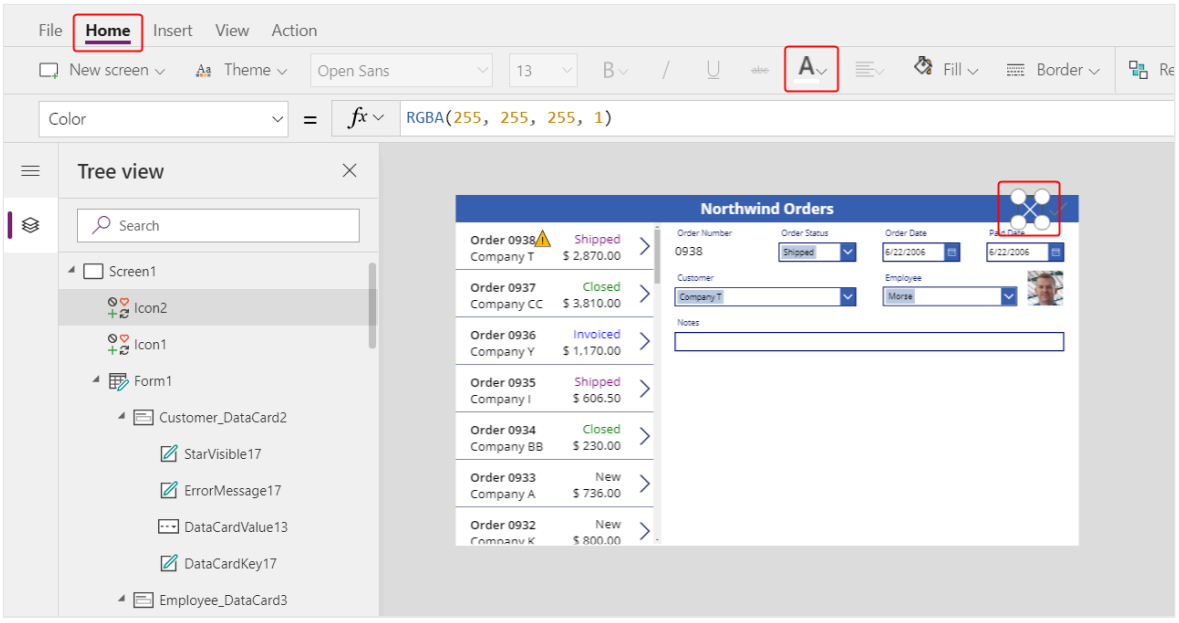
1. On the **Insert** tab, select **Icons** > **Cancel**:



The icon appears in the upper-left corner by default, where other controls might make the icon difficult to find:

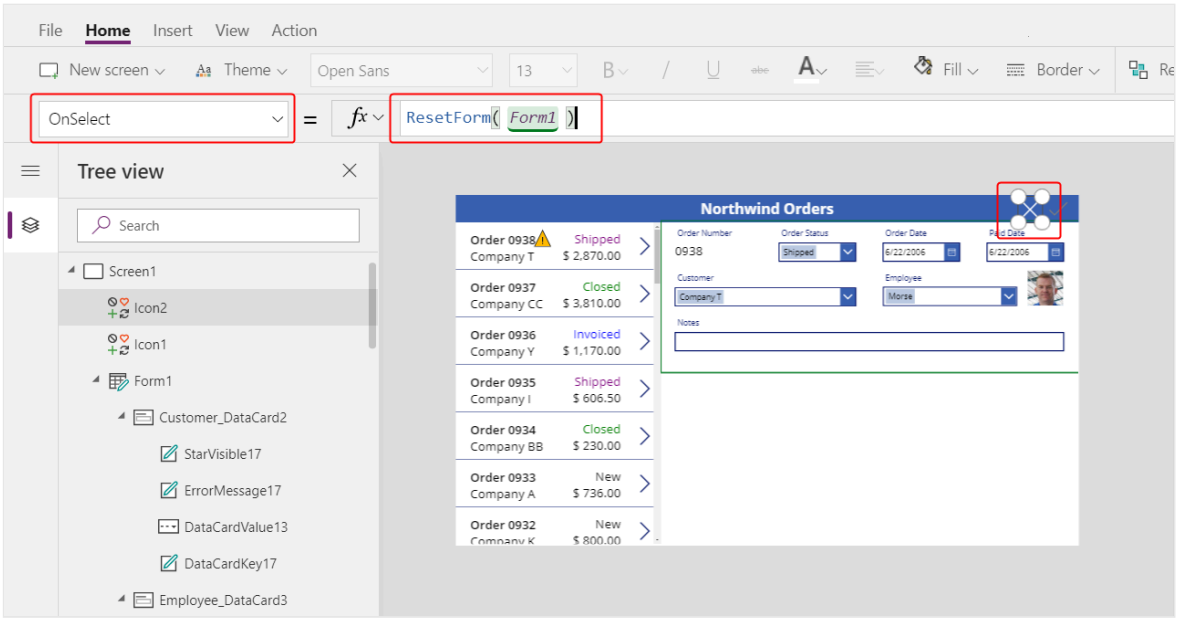


2. On the **Home** tab, change the icon's **Color** property to white, resize the icon, and move it to the left of the Check icon:



3. Set the Cancel icon's **OnSelect** property to this formula:

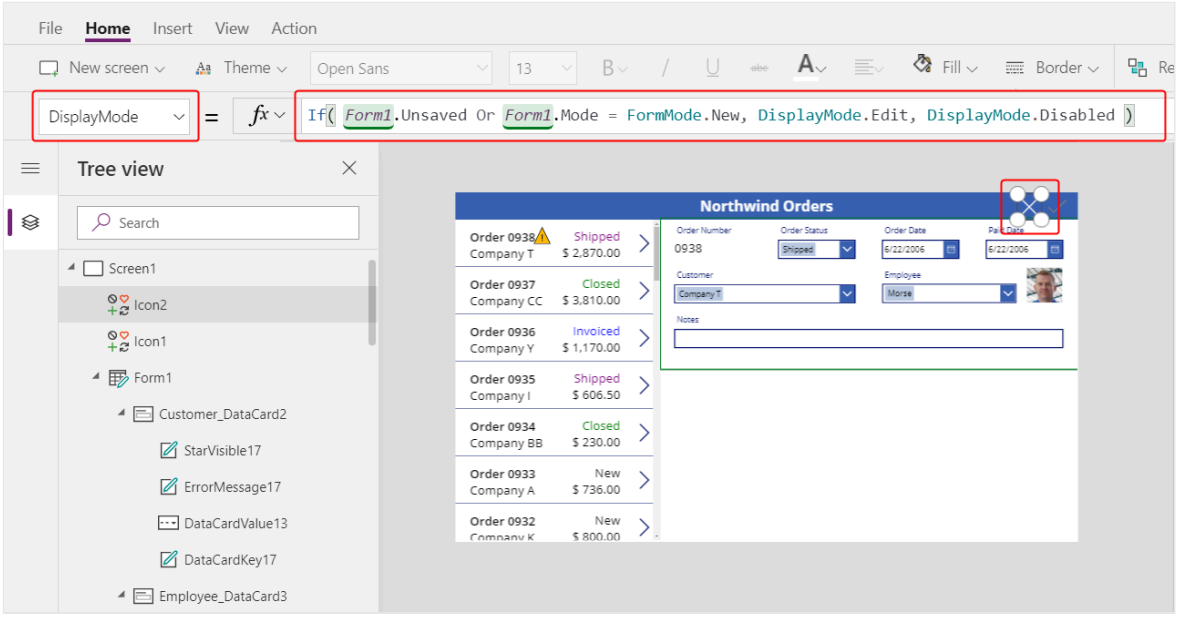
```
Power Fx

ResetForm( Form1 )
```



The ResetForm function discards all changes in the form, which returns it to its original state.

4. Set the Cancel icon's **DisplayMode** property to this formula:

```
Power Fx

If( Form1.Unsaved Or Form1.Mode = FormMode.New, DisplayMode.Edit,
DisplayMode.Disabled )
```

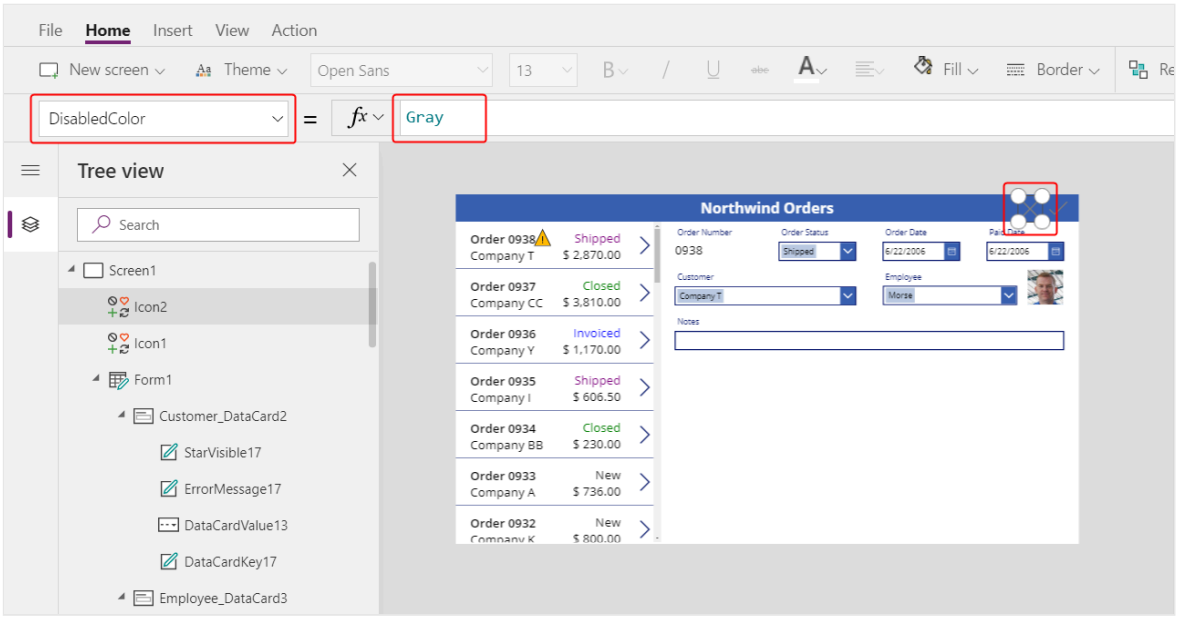


This formula differs slightly from the one for the Check icon. The Cancel icon is disabled if all changes have been saved or the form is in **New** mode, which you'll enable next. In that case, **ResetForm** discards the new record.

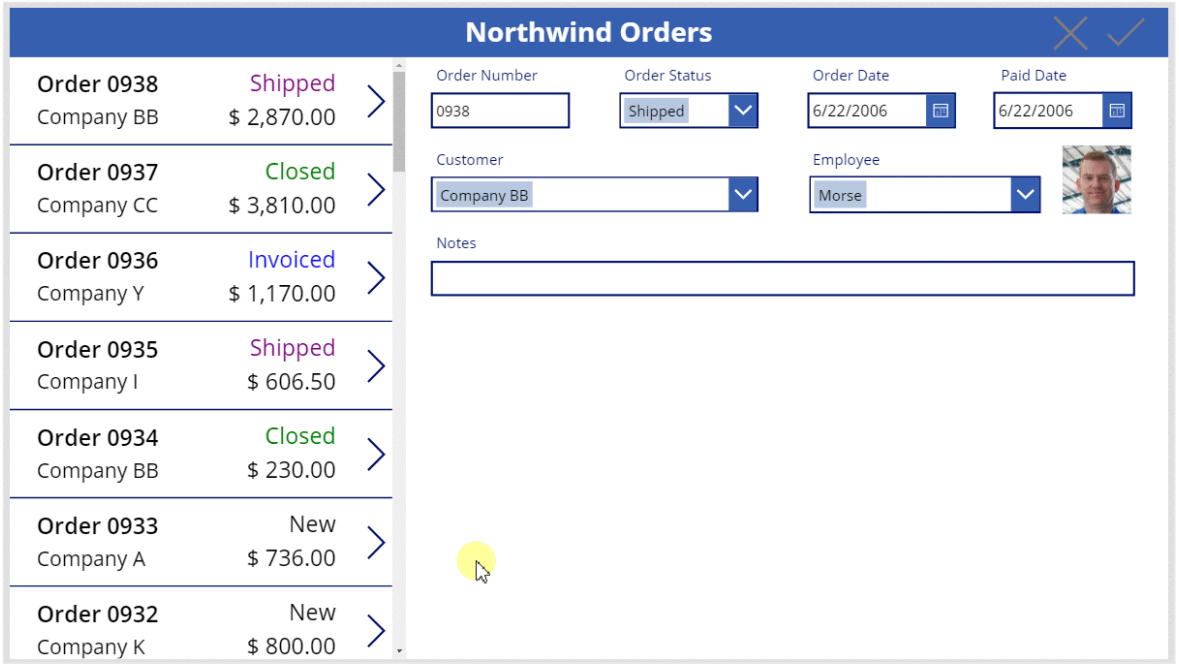
5. Set the Cancel icon's **DisabledColor** property to this value:

Power Fx

```
Gray
```

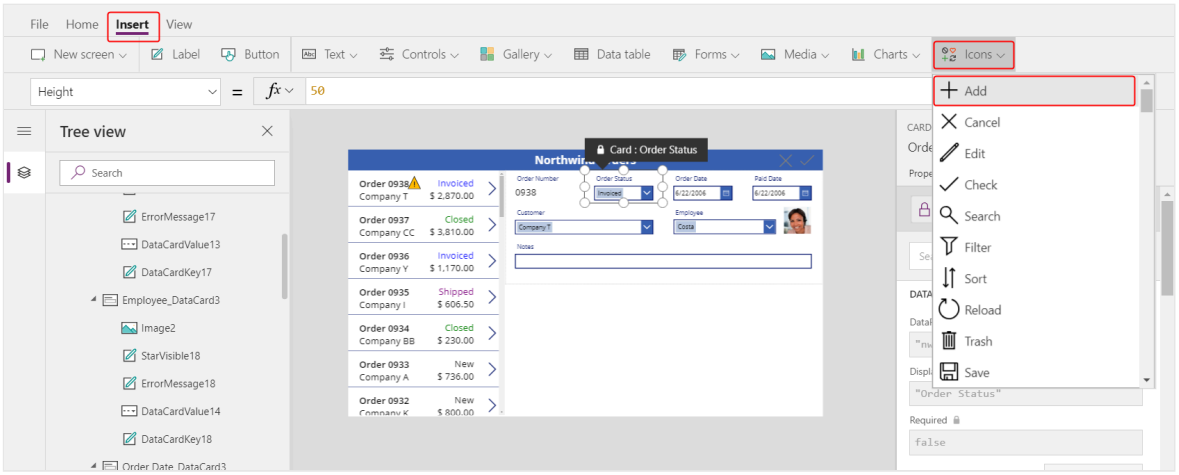


The user can cancel changes to an order, and the Check and Cancel icons are disabled and dimmed if all changes have been saved:

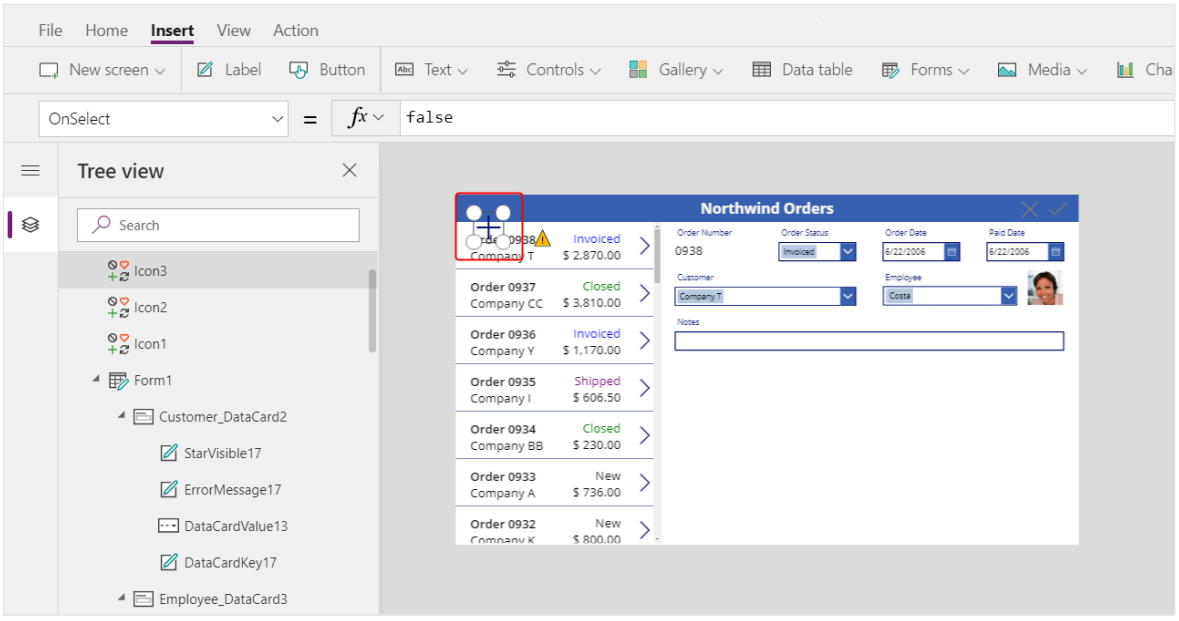


Add an Add icon

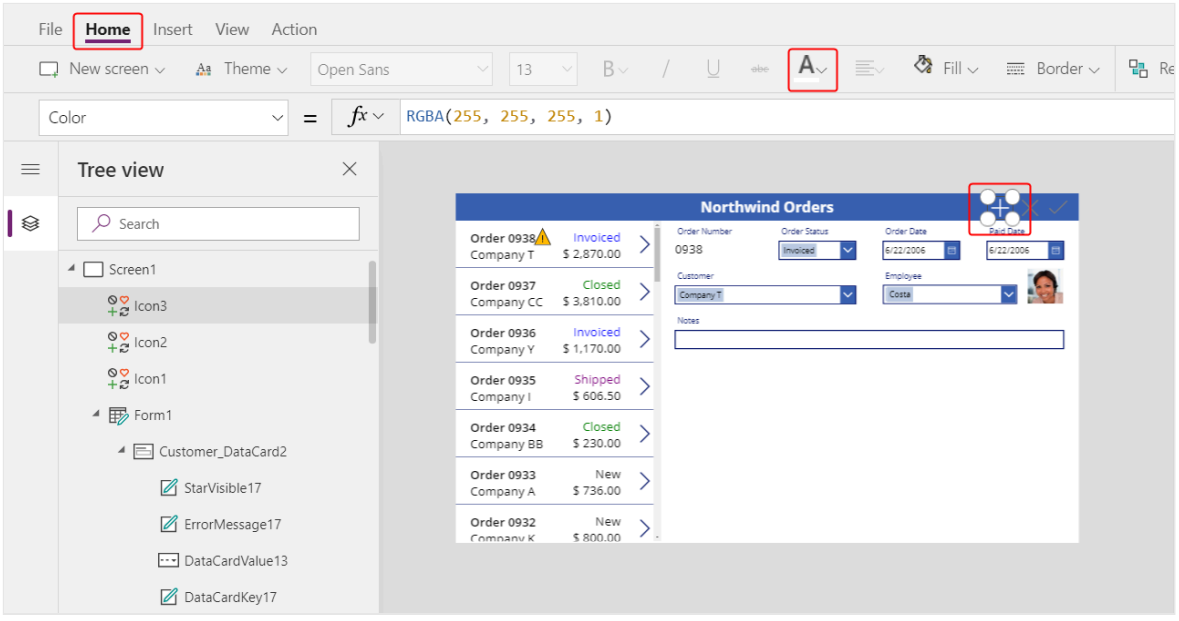
1. On the **Insert** tab, select **Icons** > **Add**.



The **Add** icon appears in the upper-left corner by default, where other controls might make it difficult to find:

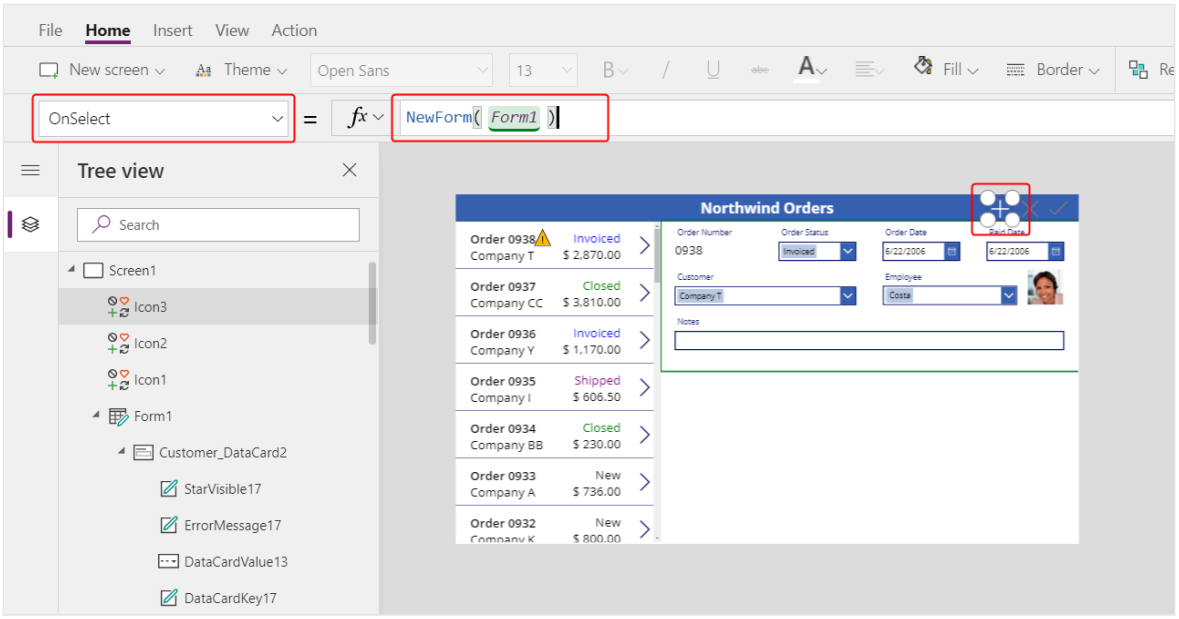


2. On the **Home** tab, set the **Color** property of the Add icon to white, resize the icon, and move it to the left of the Cancel icon:



3. Set the Add icon's **OnSelect** property to this formula:

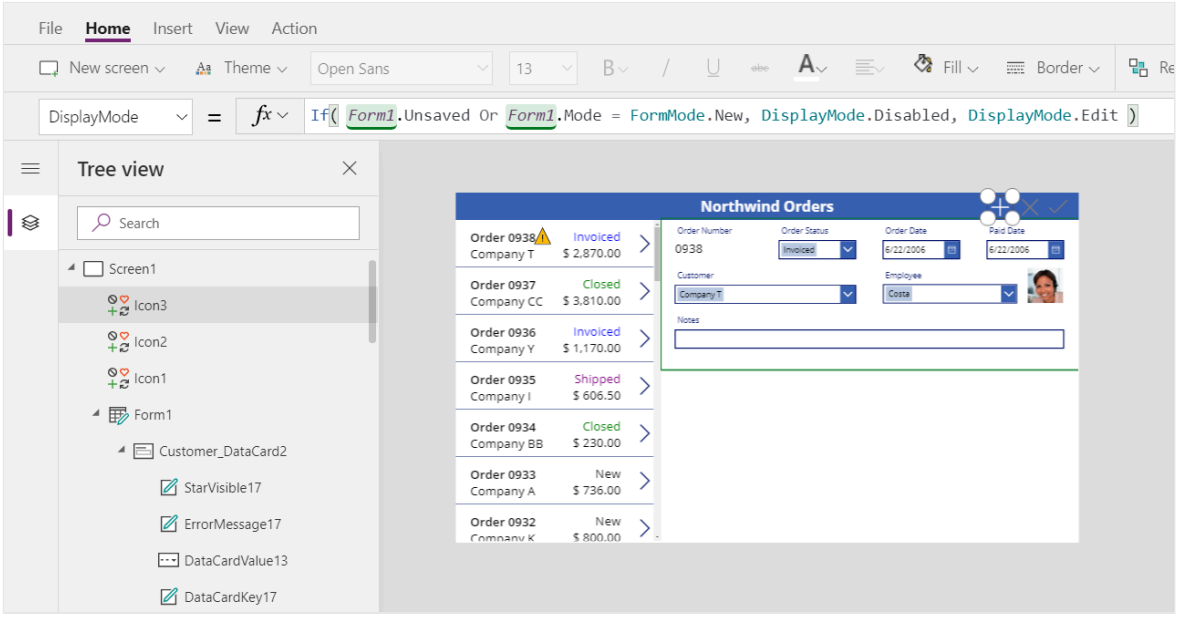
```
Power Fx

NewForm( Form1 )
```



The NewForm function shows a blank record in the form.

4. Set the Add icon's **DisplayMode** property to this formula:

```
Power Fx

If( Form1.Unsaved Or Form1.Mode = FormMode.New, DisplayMode.Disabled,
DisplayMode.Edit )
```

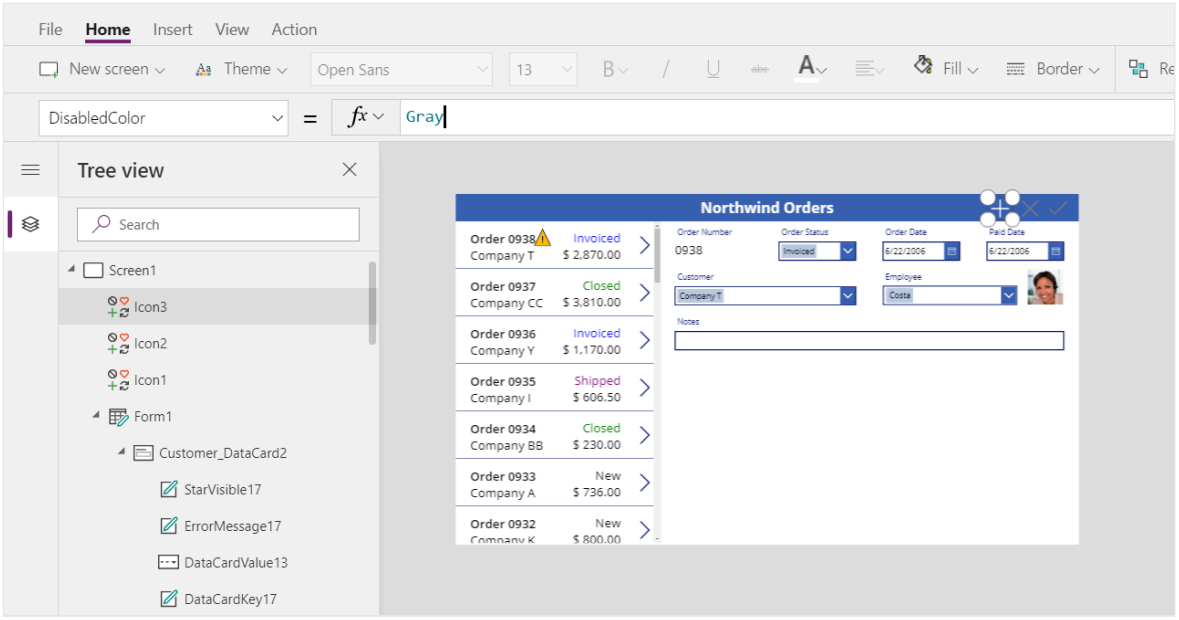


The formula disables the Add icon under these conditions:

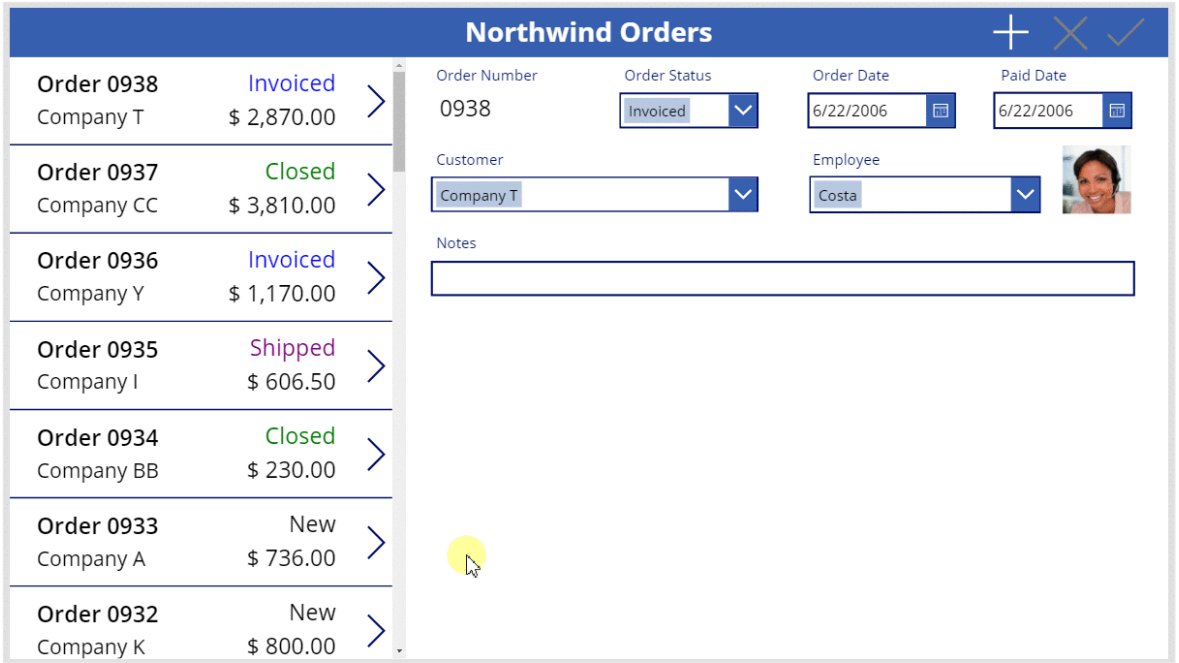
- The user makes changes but doesn't save or cancel them, which is the opposite behavior from the Check and Cancel icons.
- The user selects the Add icon but makes no changes.

5. Set the Add icon's **DisabledColor** property to this value:

```
Power Fx

Gray
```



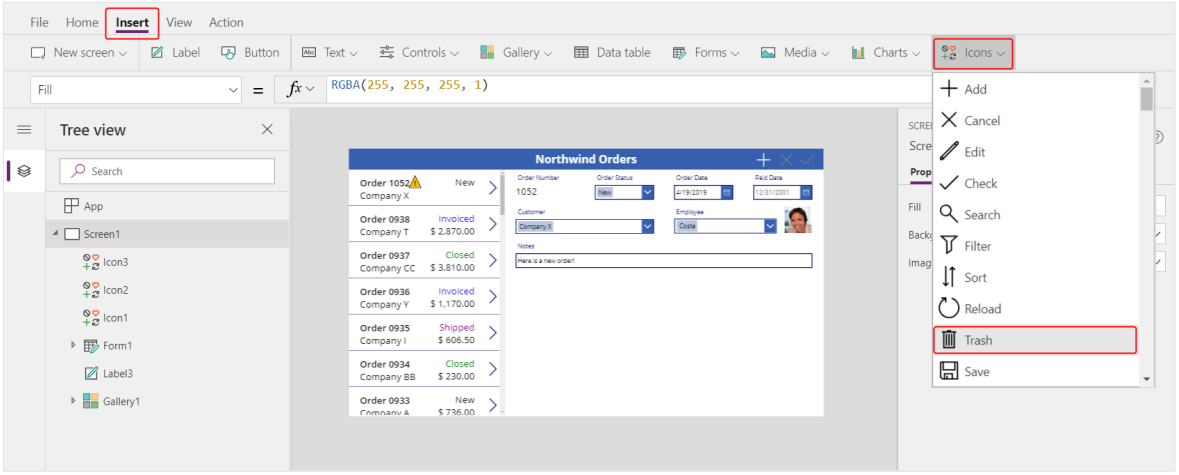
The user can create an order if they make no changes or they save or cancel any changes they've made. (If the user selects this icon, they can't select it again until they make one or more changes and then save or cancel those changes):



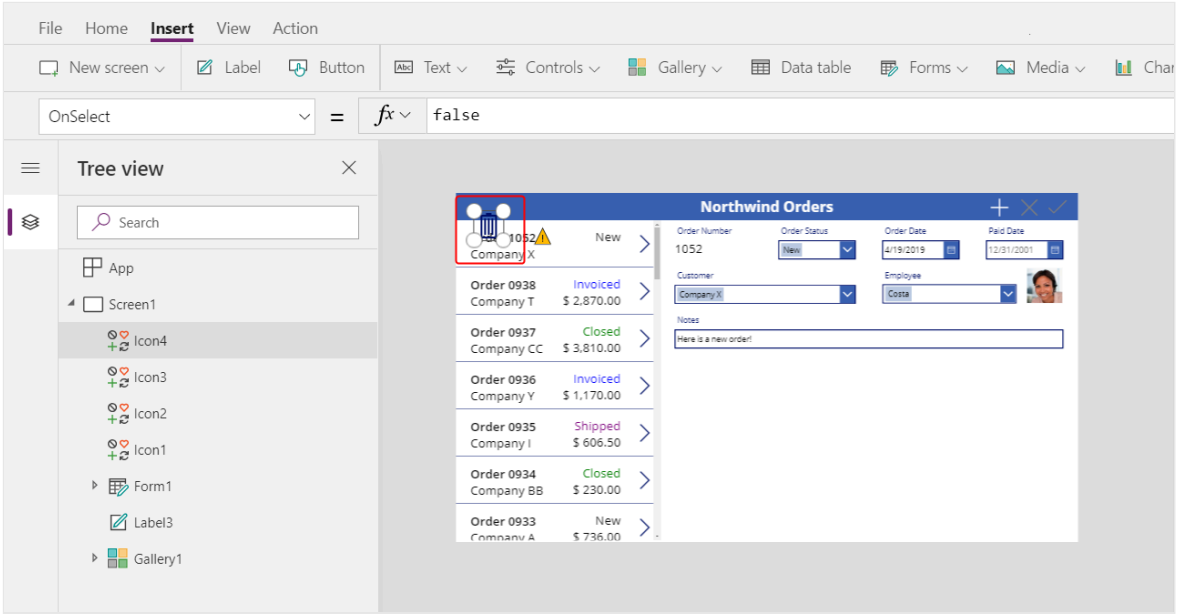
> **Note**
>
> If you create and save an order, you might need to scroll down in the order gallery to show your new order. It won't have a total price because you haven't added any order details yet.

Add a Trash icon

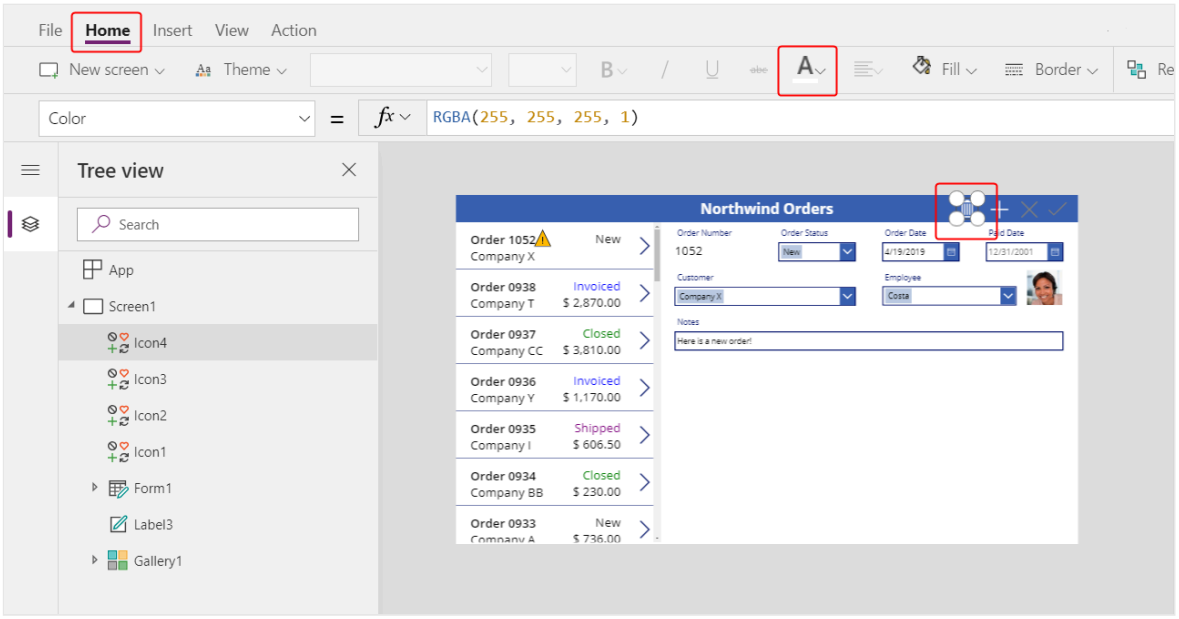
1. On the **Insert** tab, select **Icons** > **Trash**.



   The **Trash** icon appears in the upper-left corner by default, where other controls might make it difficult to find:

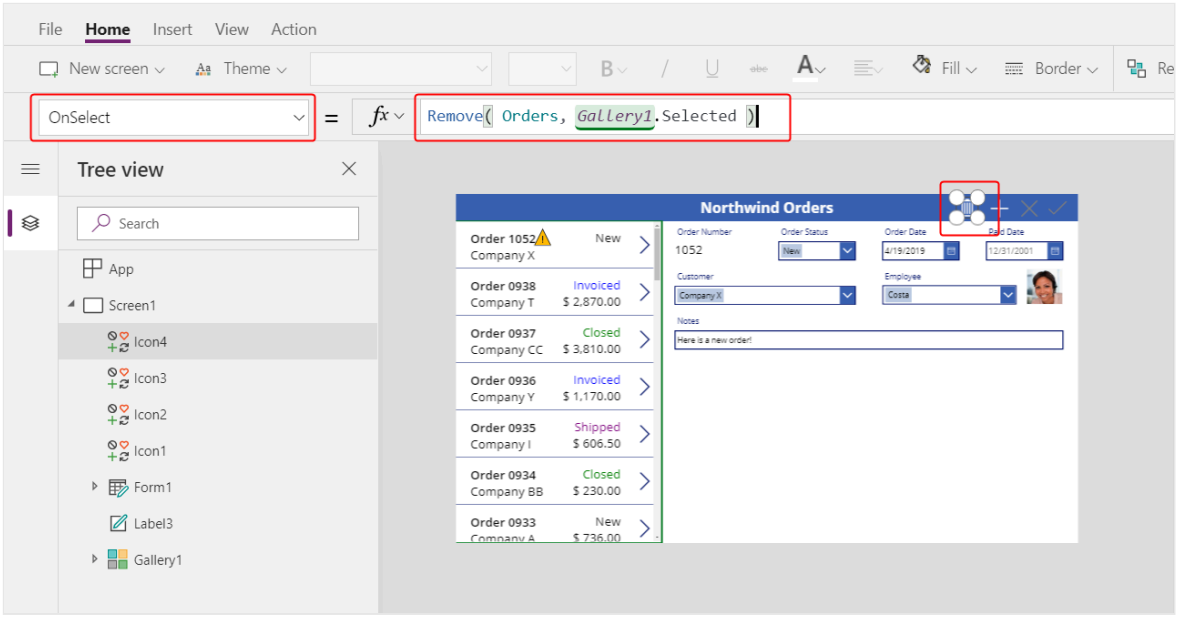


2. On the **Home** tab, change the Trash icon's **Color** property to white, resize the icon, and move it to the left of the Add icon:



3. Set the Trash icon's **OnSelect** property to this formula:

```
Power Fx

Remove( Orders, Gallery1.Selected )
```

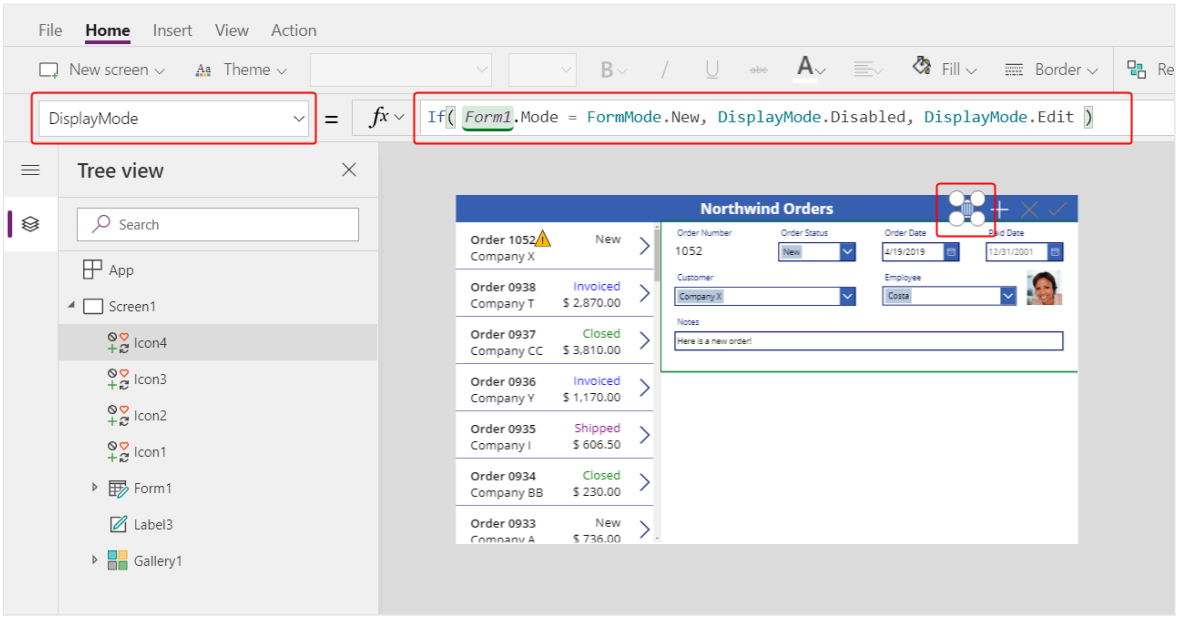


The **Remove** function removes a record from a data source. In this formula, the function removes the record that's selected in the order gallery. The Trash icon appears near the summary form (not the order gallery) because the form shows more details about the record, so the user can more easily identify the record that the formula will delete.

4. Set the Trash icon's **DisplayMode** property to this formula:

```
Power Fx
```

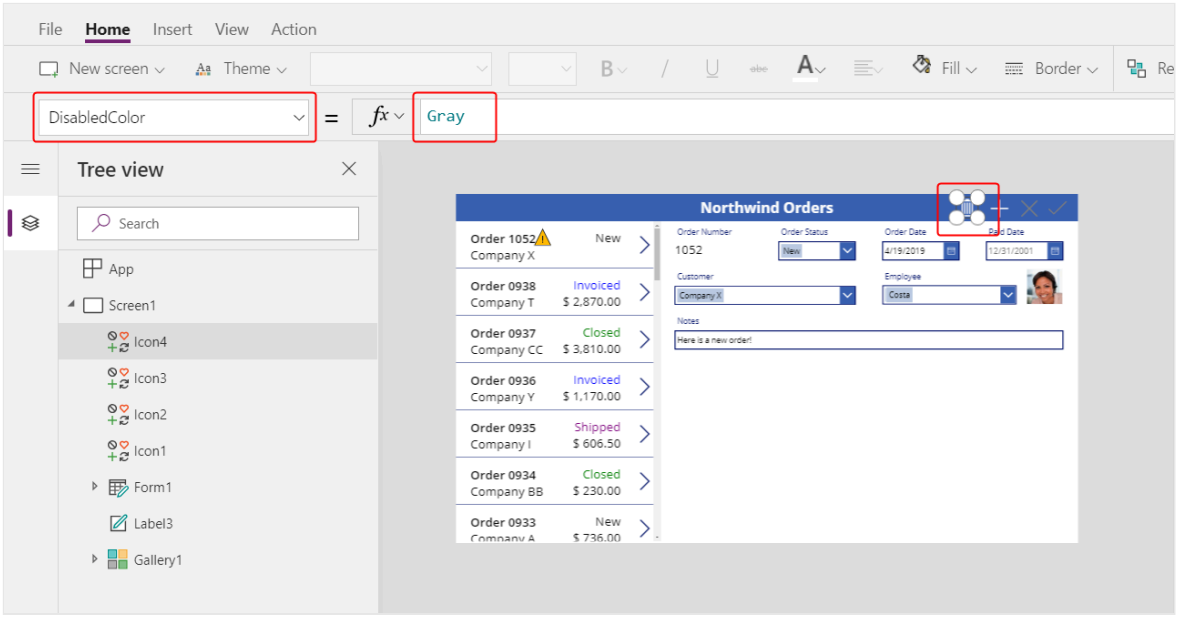
```
If( Form1.Mode = FormMode.New, DisplayMode.Disabled, DisplayMode.Edit )
```



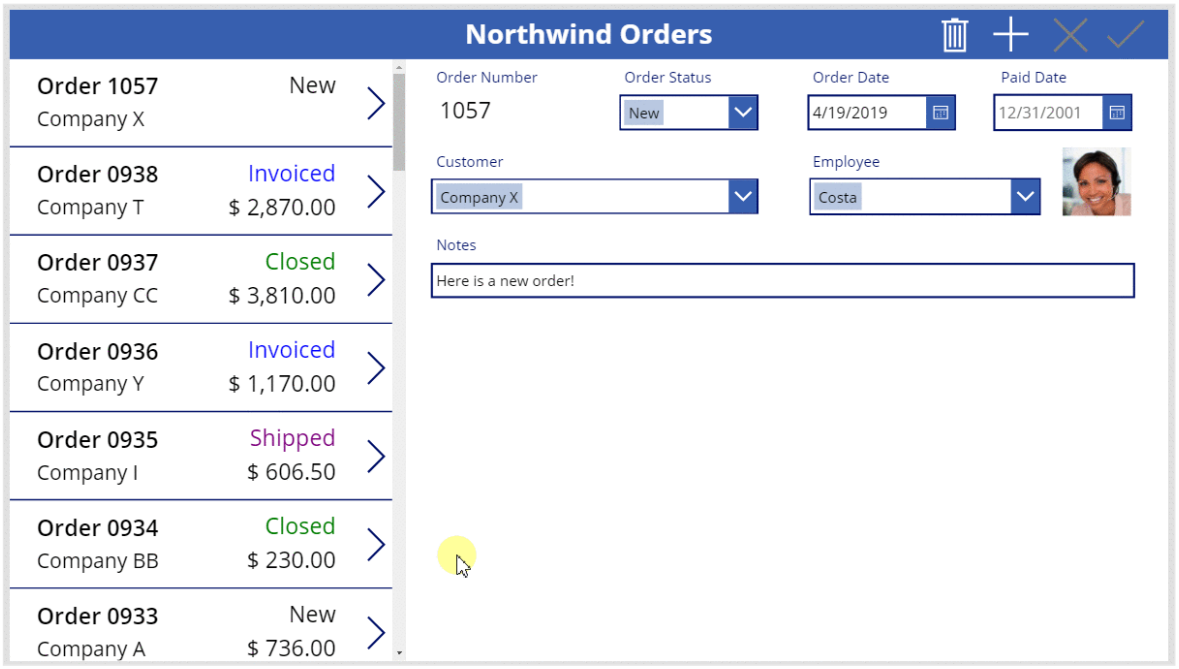
This formula disables the Trash icon if the user is creating a record. Until the user saves the record, the **Remove** function has no record to delete.

5. Set the Trash icon's **DisabledColor** property to this value:

```
Power Fx

Gray
```



The user can delete an order.



Summary

To recap, you added a form in which the user can show and edit a summary of each order, and you used these elements:

- A form that shows data from the **Orders** table: **Form1.DataSource** = `Orders`
- A connection between the form and the order gallery: **Form1.Item** = `Gallery1.Selected`
- An alternate control for the **Order number** field: **View text**
- A many-to-one relationship to show the employee's picture in the **Employee** data card: `DataCardValue1.Selected.Picture`
- An icon to save changes to an order: `SubmitForm( Form1 )`
- An icon to cancel changes to an order: `ResetForm( Form1 )`
- An icon to create an order: `NewForm( Form1 )`
- An icon to delete an order: `Remove( Orders, Gallery1.Selected )`

Next step

In the next topic, you'll add another gallery to show the products in each order, and you'll change those details by using the **Patch** function.

Create the detail gallery

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a detail gallery in a canvas app

Article • 12/16/2022

Follow step-by-step instructions to create a detail gallery in a canvas app for managing fictitious data in the Northwind Traders database. This topic is part of a series that explains how to build a business app on relational data in Microsoft Dataverse. For best results, explore these topics in this sequence:

1. Create an order gallery.
2. Create a summary form.
3. Create a detail gallery (**this topic**).

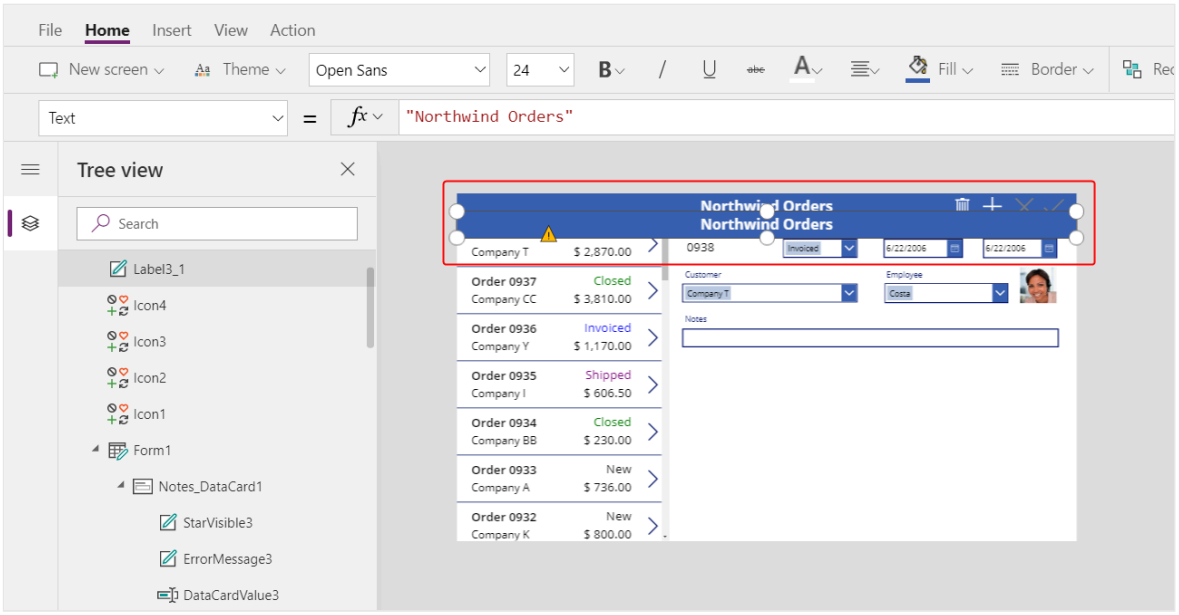


Prerequisites

Before you start this topic, you must install the database as described earlier in this topic. You must then either create the order gallery and the summary form or open the **Northwind Orders (Canvas) - Begin Part 3** app, which already contains that gallery and that form.

Create another title bar

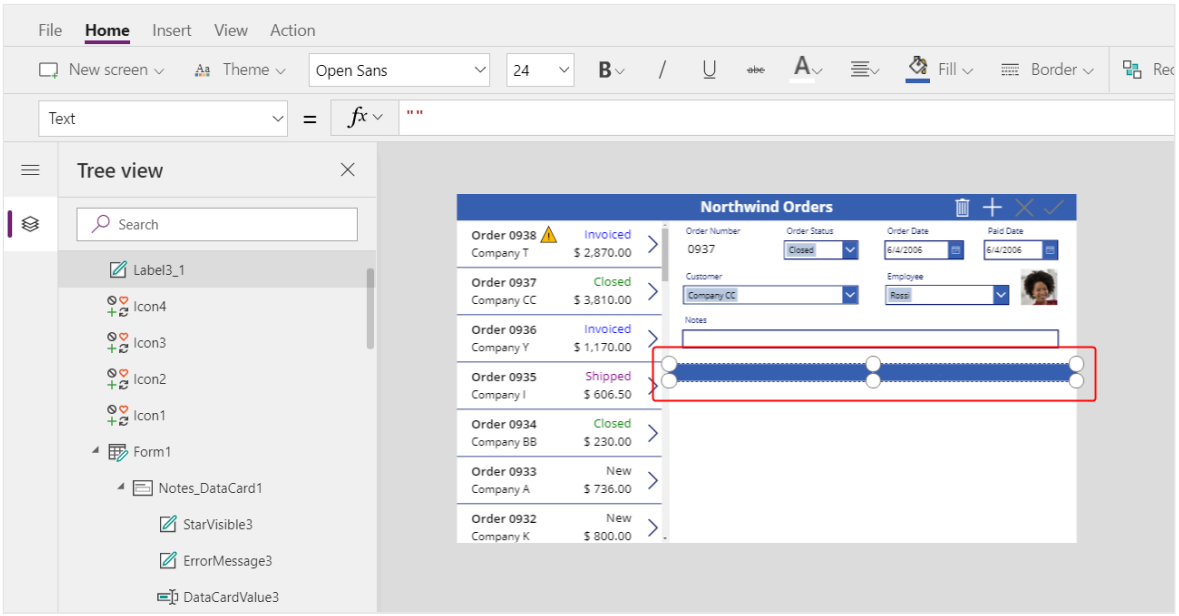
1. At the top of the screen, select the **Label** control that acts as a title bar, copy it by pressing Ctrl-C, and then paste it by pressing Ctrl-V:



2. Resize and move the copy so that it appears just under the summary form.

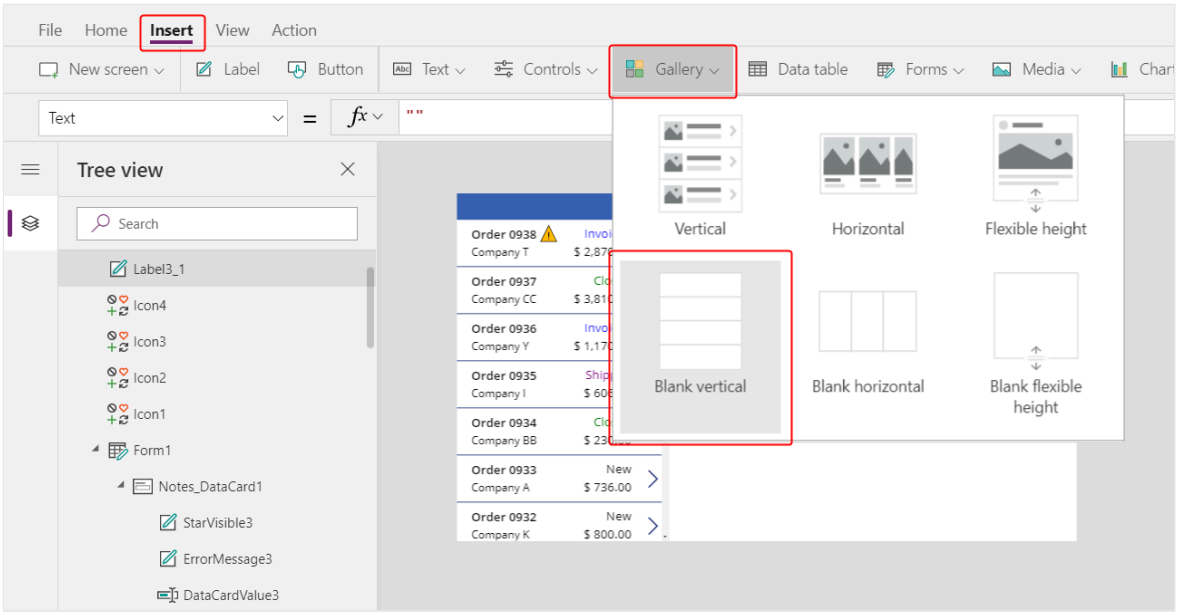
3. Remove the text from the copy in either of these ways:

   - Double-click the text to select it, and then press Delete.
   - Set the label's **Text** property to an empty string ("").



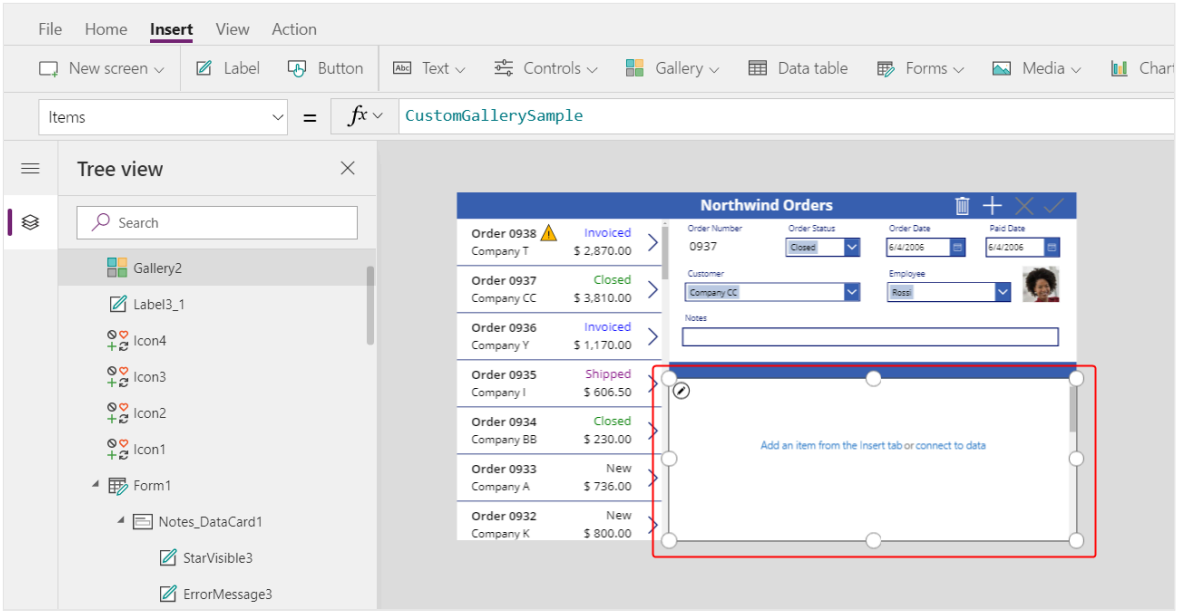
Add a gallery

1. Insert a Gallery control with a **Blank vertical** layout:



The new gallery, which will show order details, appears in the upper-left corner.

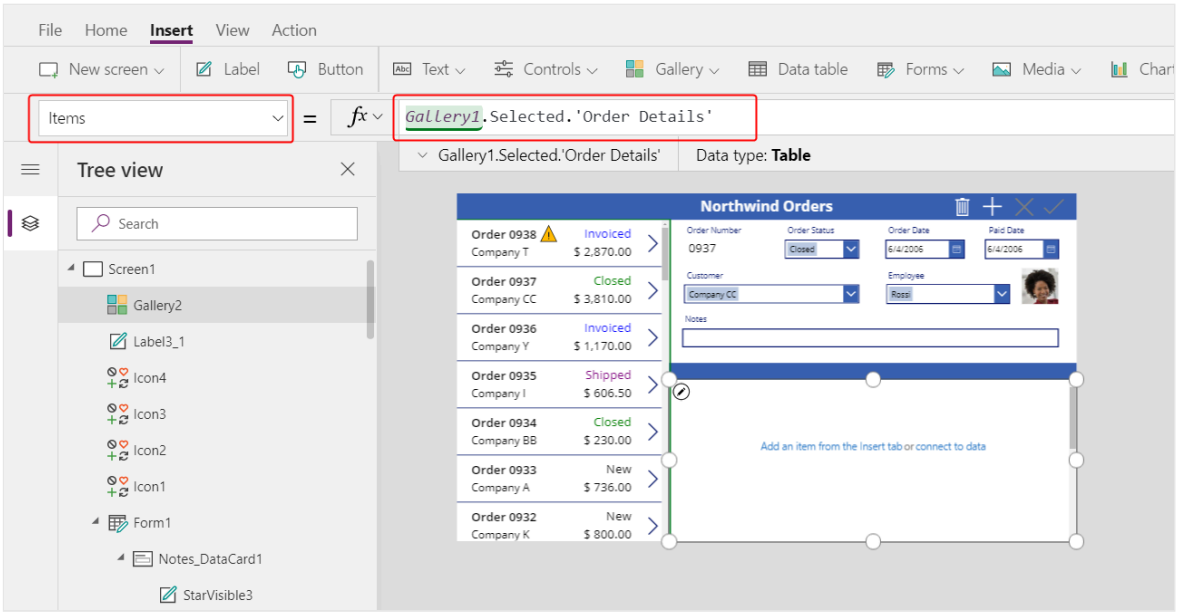
2. Close the fly out data source dialog, and then resize and move the detail gallery to the lower-right corner, below the new title bar:



3. Set the **Items** property of the detail gallery to this formula:

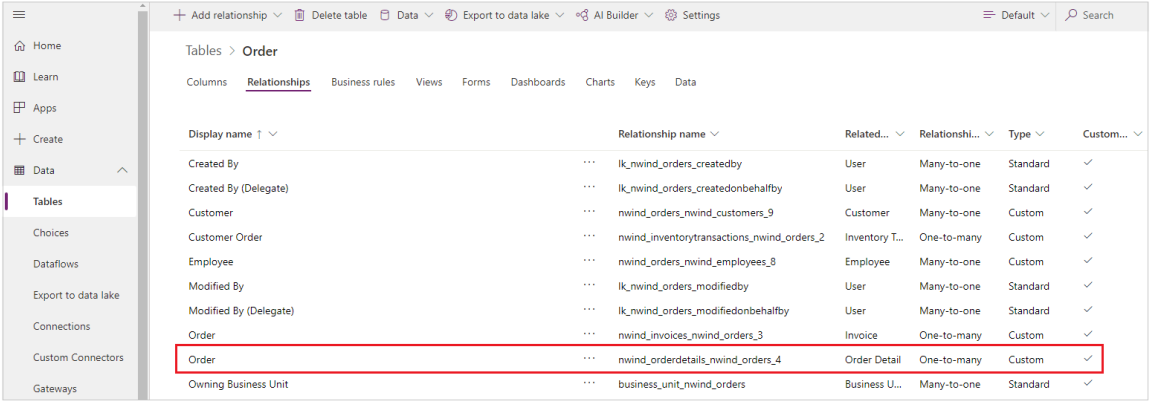
Power Fx

```
Gallery1.Selected.'Order Details'
```



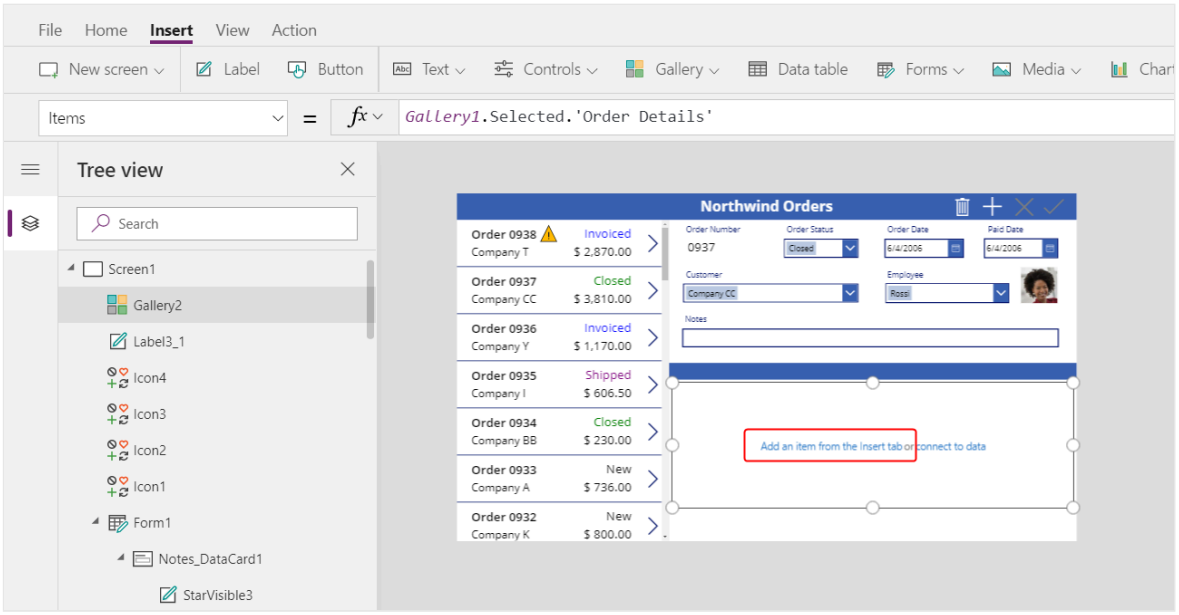
If an error appears, confirm that the order gallery is named **Gallery1** (in the **Tree view** pane near the left edge). If that gallery has a different name, rename it **Gallery1**.

You've just linked the two galleries. When the user selects an order in the order gallery, that selection identifies a record in the **Orders** table. If that order contains one or more line items, the record in the **Orders** table is linked to one or more records in the **Order details** table, and data from those records appears in the detail gallery. This behavior reflects the one-to-many relationship that was created for you between the **Orders** and **Order Details** tables. The formula that you specified "walks" that relationship by using dot notation:



Show product names

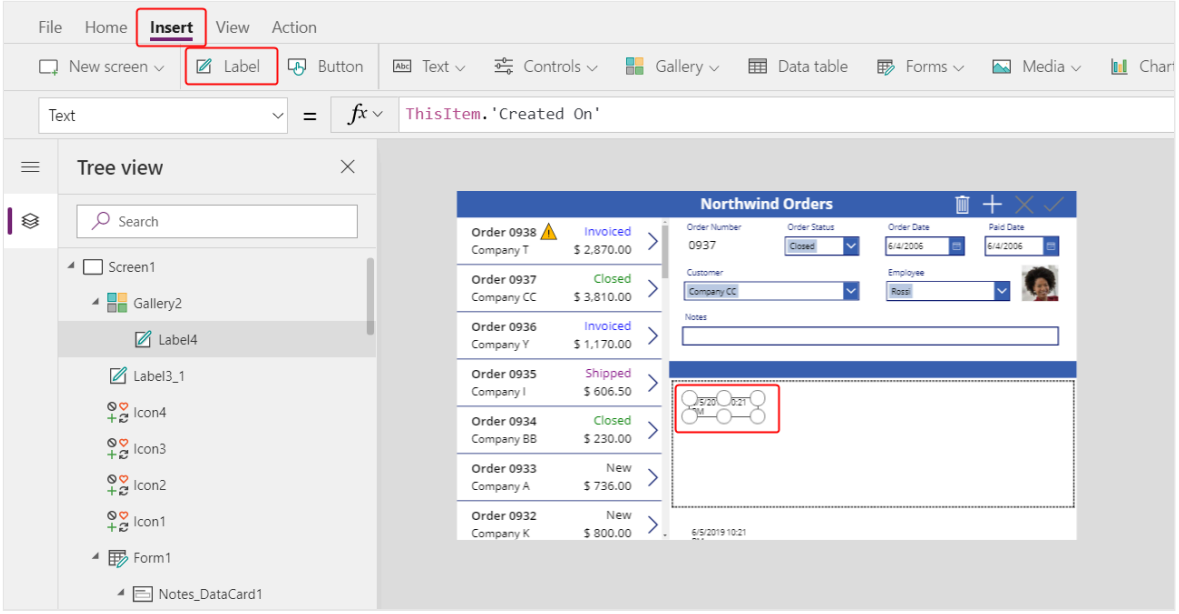
1. In the detail gallery, select **Add an item from the Insert tab** to select the gallery template:



Ensure that you've selected the gallery template instead of the gallery itself. The bounding box should be slightly inside the gallery's boundary and probably shorter than the gallery's height. As you insert controls into this template, they are repeated for each item in the gallery.

2. On the **Insert** tab, insert a label into the detail gallery.

   The label should appear within the gallery; if it doesn't, try again, but make sure to select the gallery's template before you insert the label.

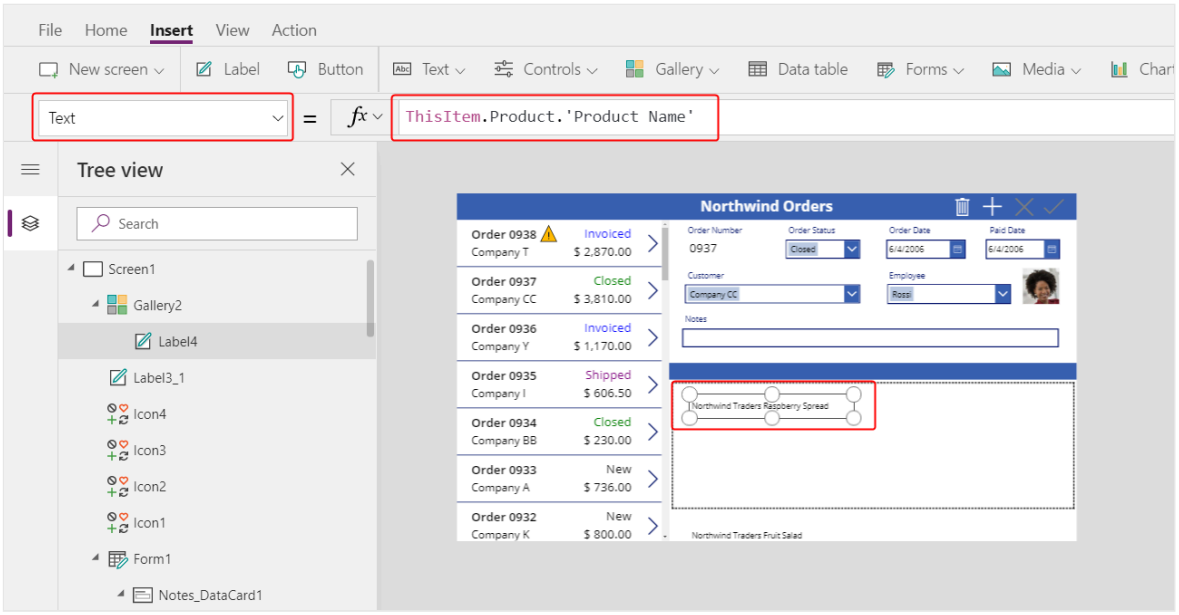


3. Set the new label's **Text** property to this formula:

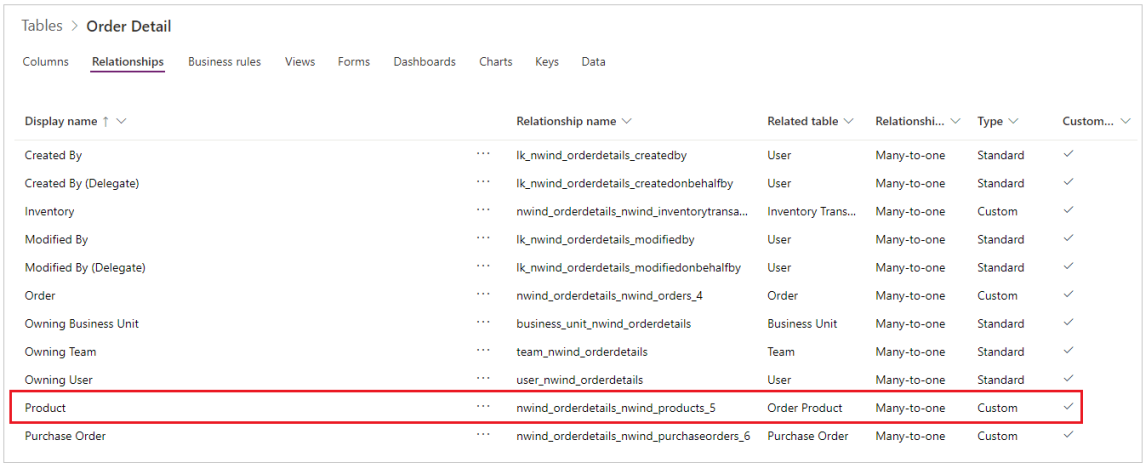
```
Power Fx

ThisItem.Product.'Product Name'
```

If no text appears, select the arrow for **Order 0901** near the bottom of the order gallery.

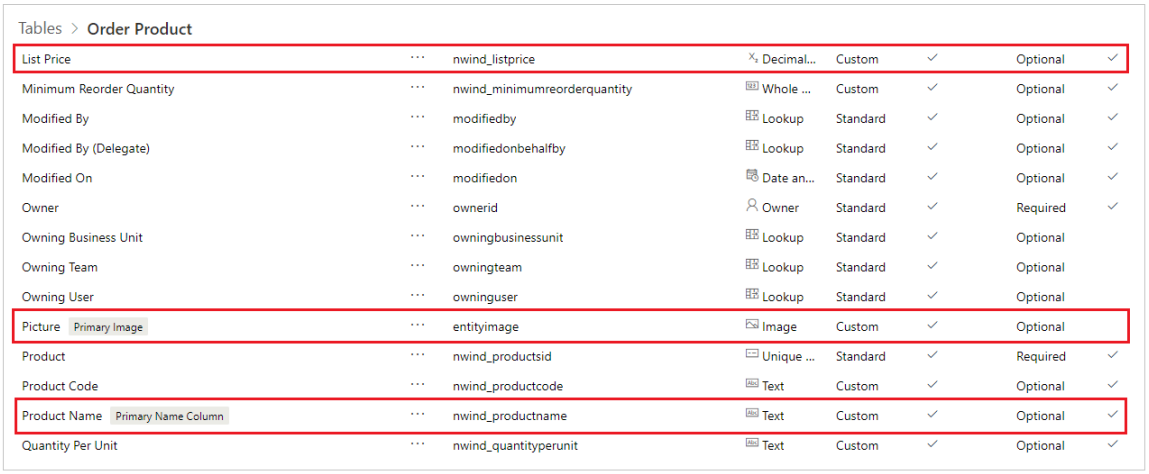
4. Resize the label so that the full text appears:



This expression walks from a record in the **Order Details** table. The record is held in **ThisItem** over to the **Order Products** table through a many-to-one relationship:

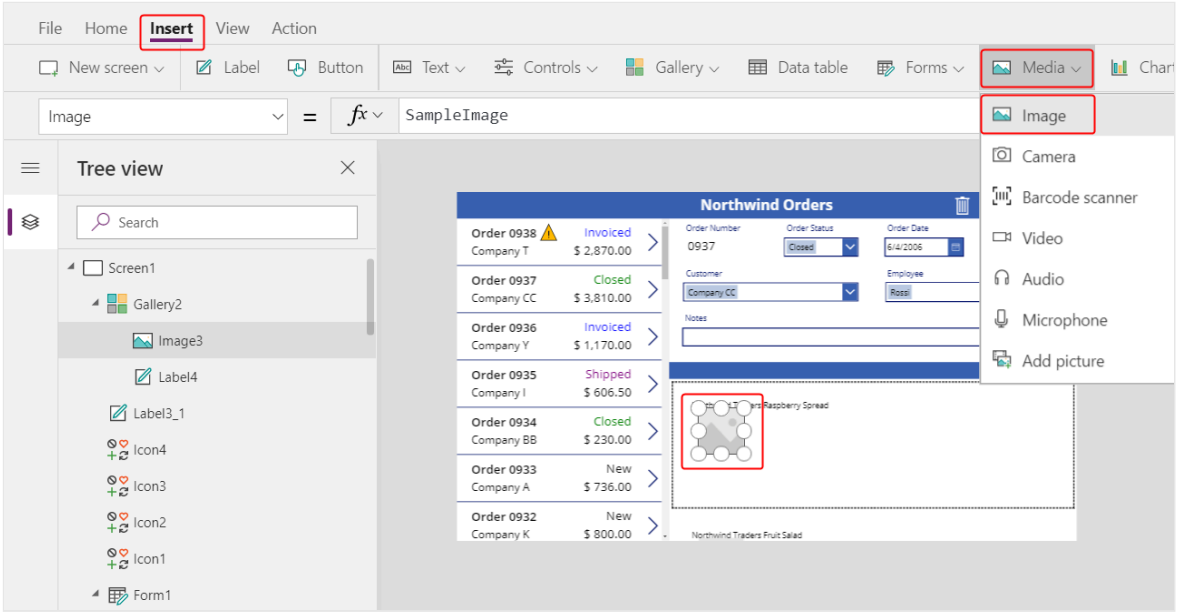


The **Product Name** column (and other columns that you're about to use) are extracted:



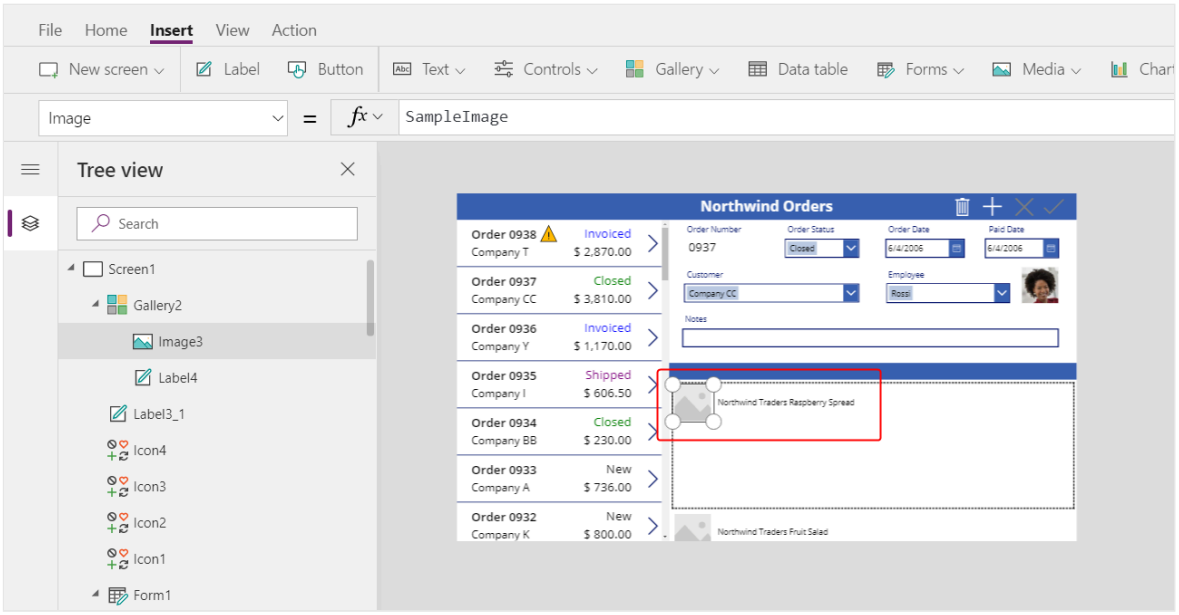
Show product images

1. On the **Insert** tab, insert an **Image** control into the detail gallery:



2. Resize and move the image and the label to be side by side.

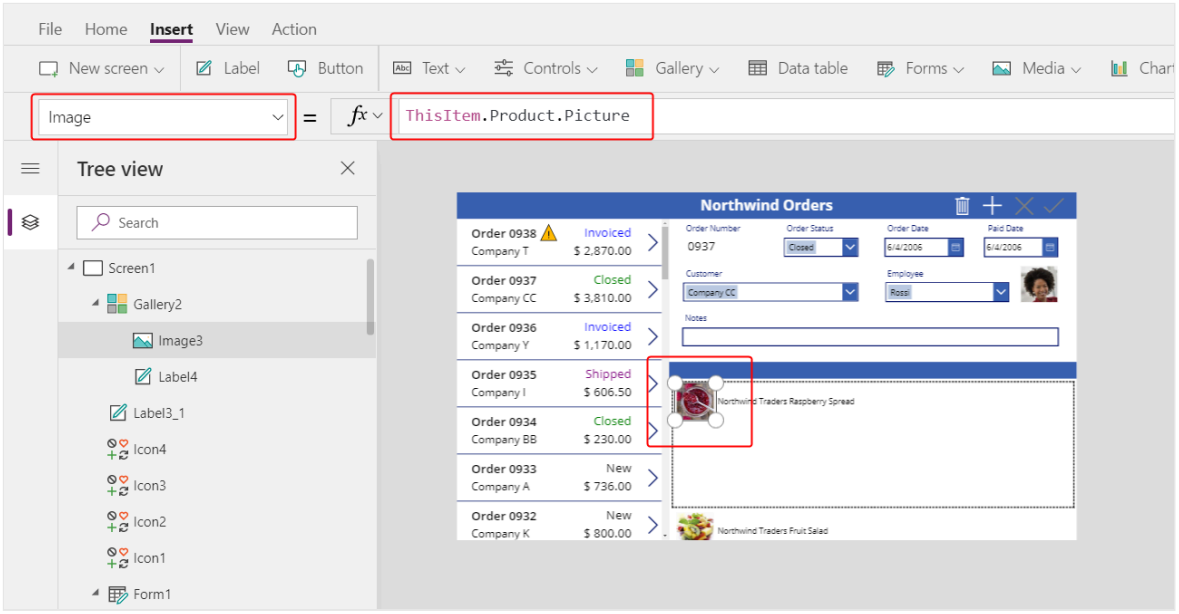
> **Tip**
>
> For fine-grained control over the size and the position of a control, start to resize or move it without pressing the Alt key, and then continue to resize or move the control while you hold down the Alt key:



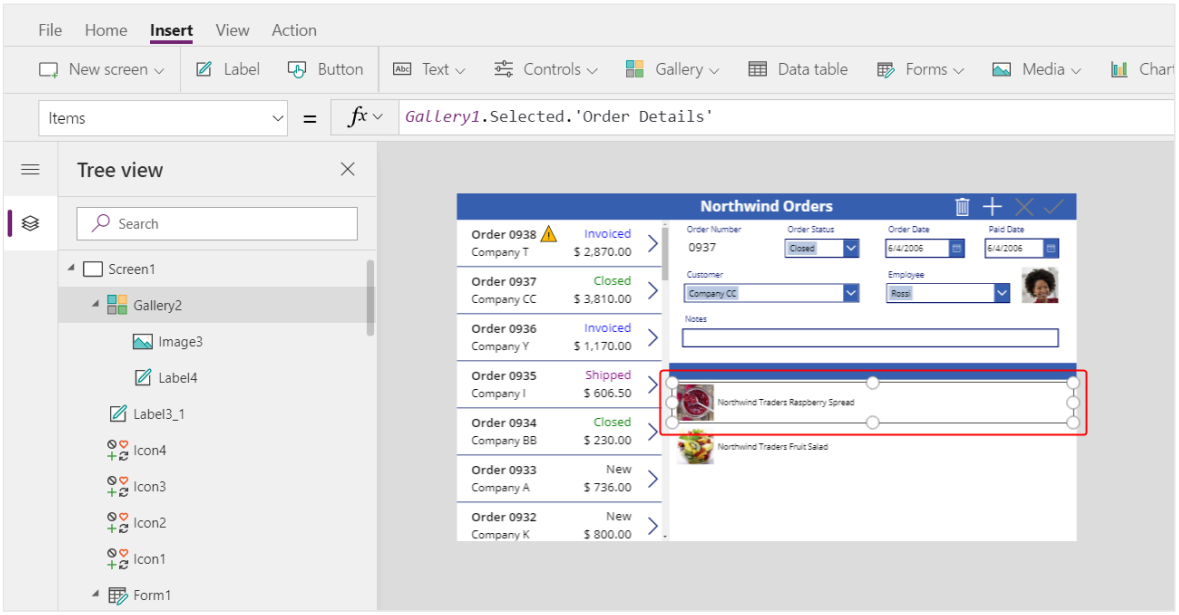
3. Set the image's **Image** property to this formula:

```
Power Fx

ThisItem.Product.Picture
```

Again, the expression references a product that's associated with this order detail and extracting the **Picture** field to display.



4. Reduce the height of the gallery's template so that more than one **Order Detail** record appears at a time:



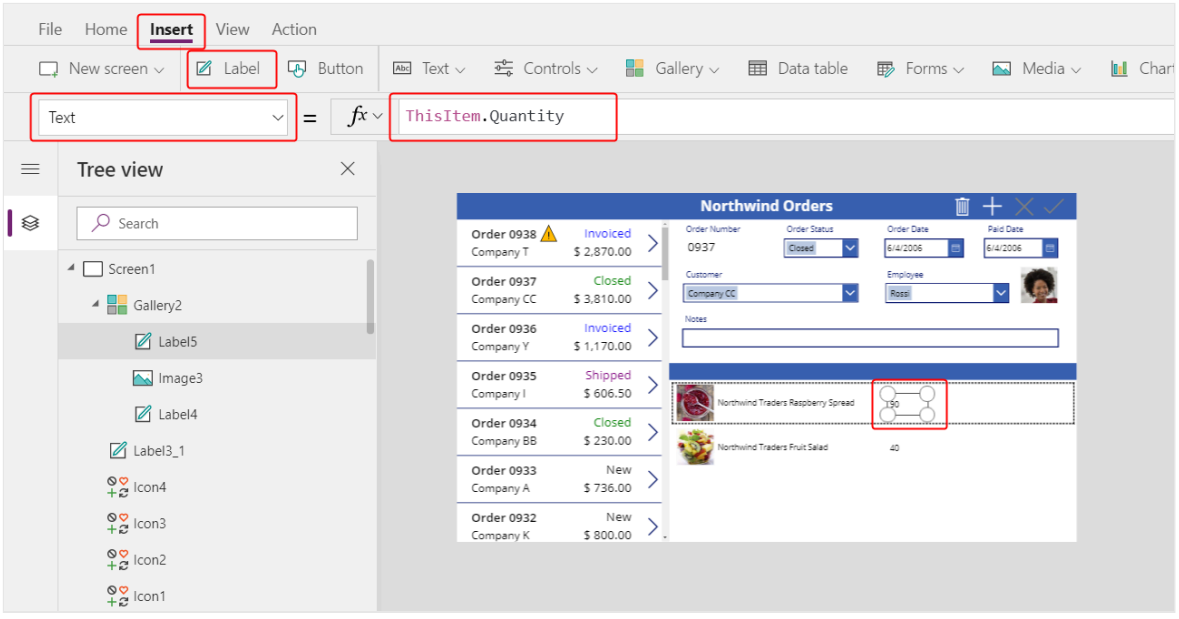
Show product quantity and cost

1. On the **Insert** tab, insert another label into the detail gallery, and then resize and move the new label to the right of the product information.

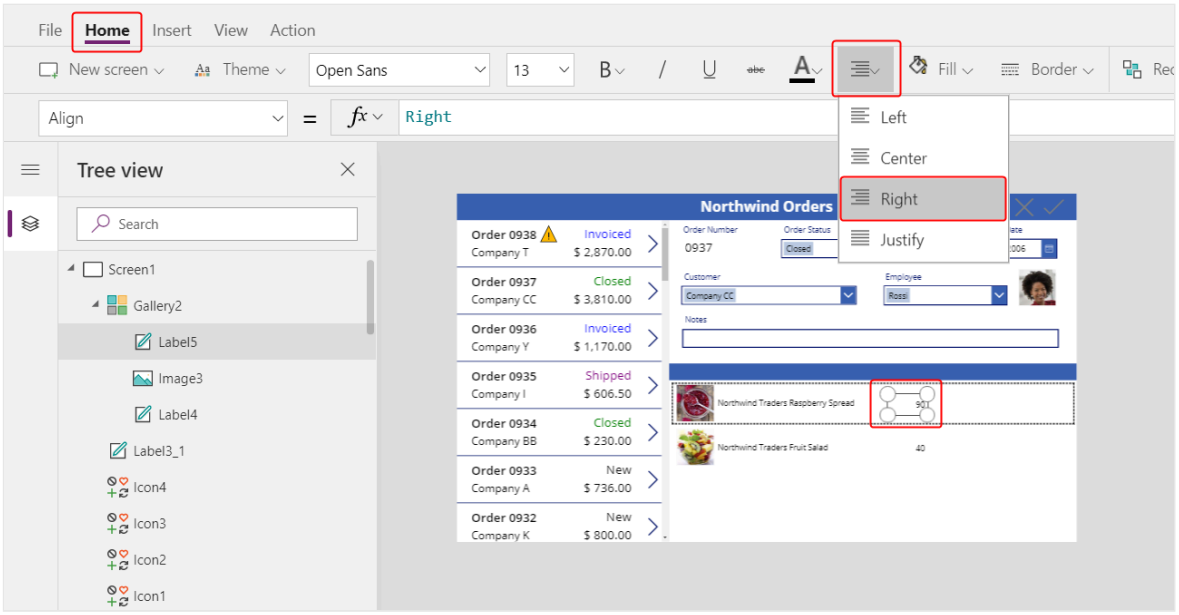
2. Set the new label's **Text** property to this expression:

```
Power Fx

ThisItem.Quantity
```

   This formula pulls information directly from the **Order Details** table (no relationship required).



3. On the **Home** tab, change the alignment of this control to **Right**:

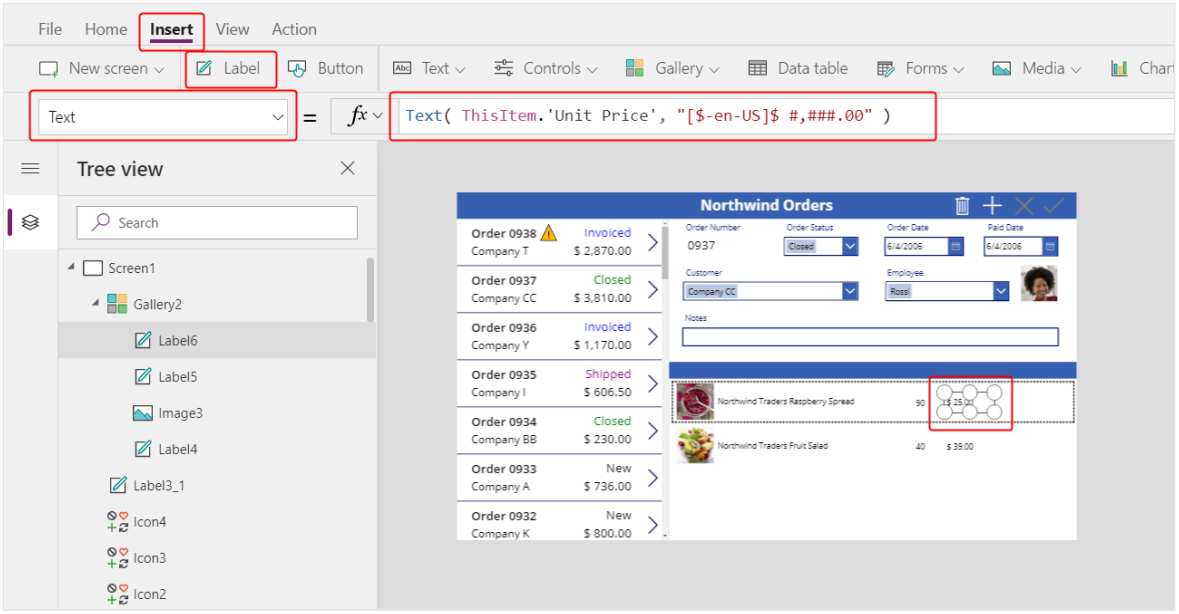


4. On the **Insert** tab, insert another label into the detail gallery, and then resize and move the label to the right of the quantity label.

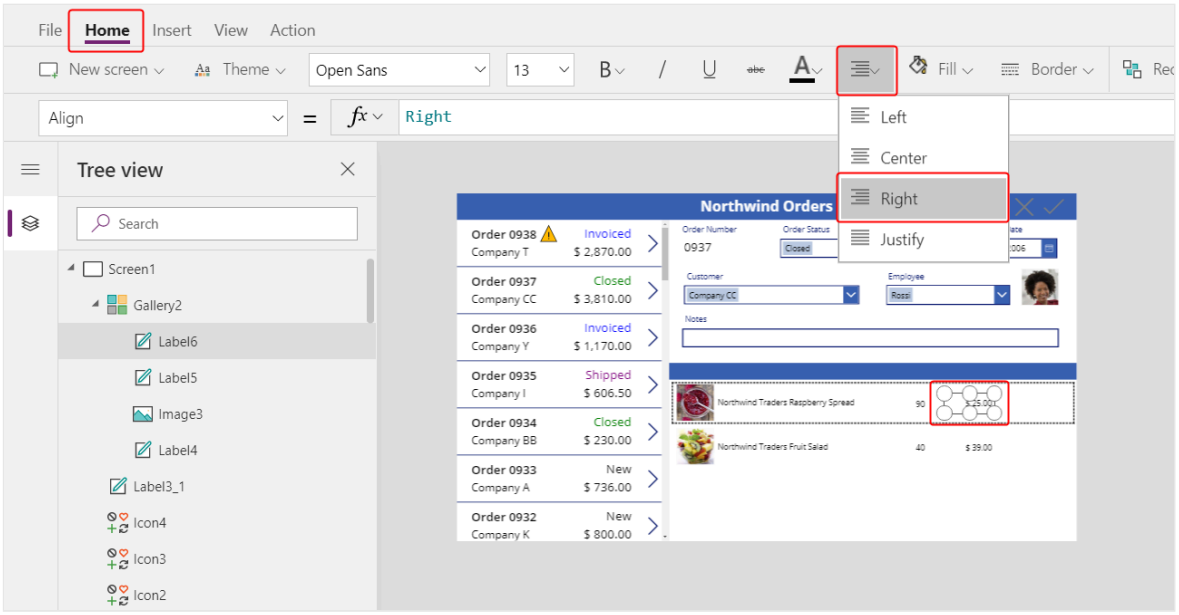
5. Set the new label's **Text** property to this formula:

```
Power Fx

Text( ThisItem.'Unit Price', "[$-en-US]$ #,###.00" )
```

If you don't include the language tag (**[$-en-US]**), it will be added for you based on your language and region. If you use a different language tag, you'll want to remove the **$** just after the close square bracket (**]**), and then add your own currency symbol in that position.



6. On the **Home** tab, change the alignment of this control to **Right**:

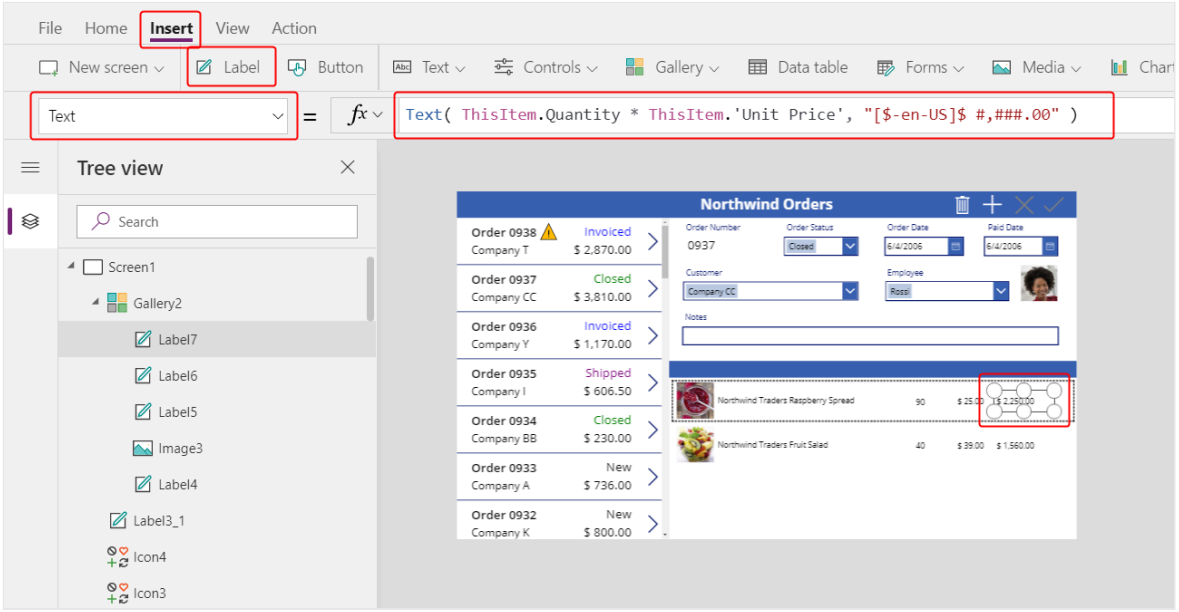


7. On the **Insert** tab, insert another label control into the detail gallery, and then resize and move the new label to the right of the unit price.

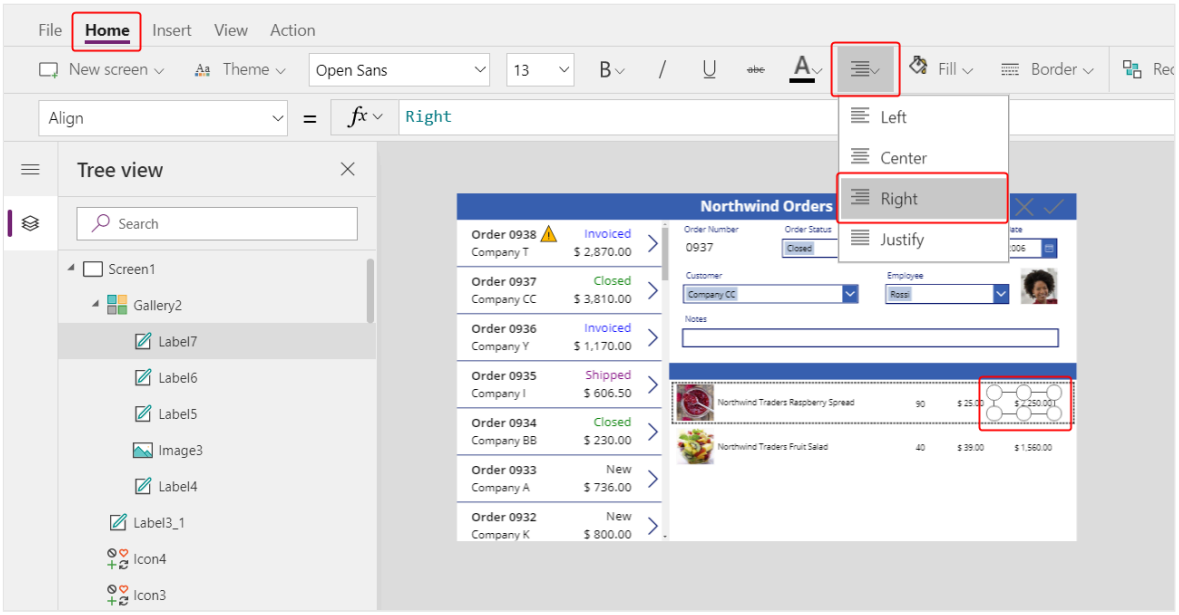
8. Set the new label's **Text** property to this formula:

```
Power Fx

Text( ThisItem.Quantity * ThisItem.'Unit Price', "[$-en-US]$ #,###.00"
)
```

Again, if you don't include the language tag (**[$-en-US]**), it will be added for you based on your language and region. If the tag is different, you'll want to use your own currency symbol instead of the **$** just after the close square bracket (**]**).



9. On the **Home** tab, change the alignment of this control to **Right**:

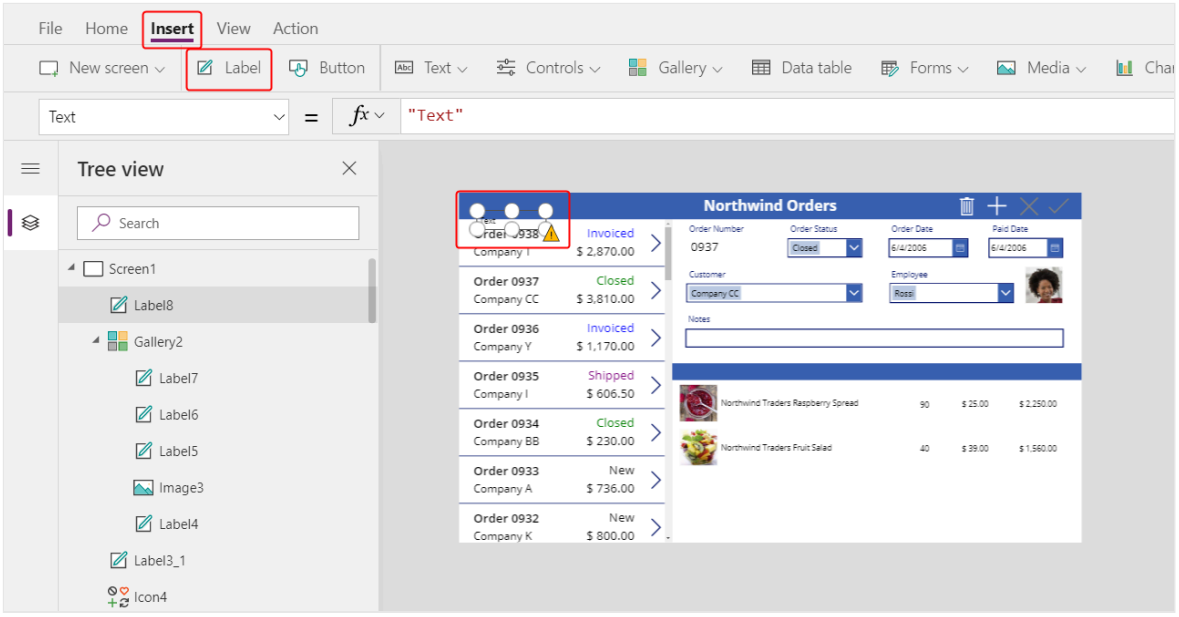


You're done adding controls to the detail gallery for now.

10. In the **Tree view** pane, select **Screen1** to ensure that the detail gallery is no longer selected.

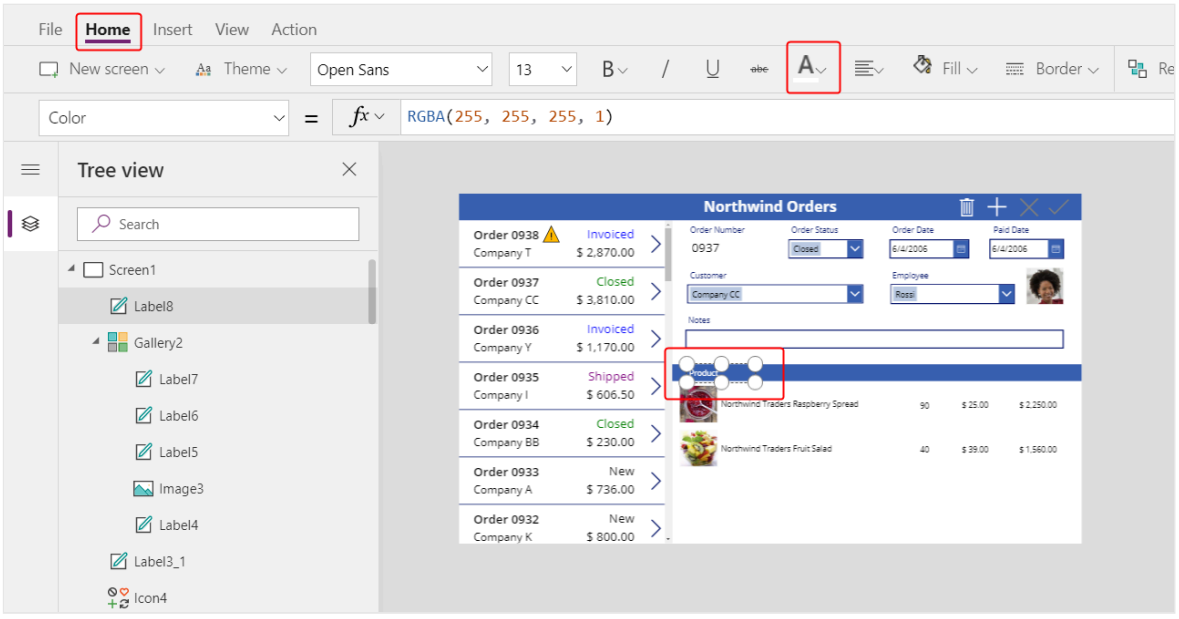
Add text to the new title bar

1. On the **Insert** tab, insert another label on to the screen:



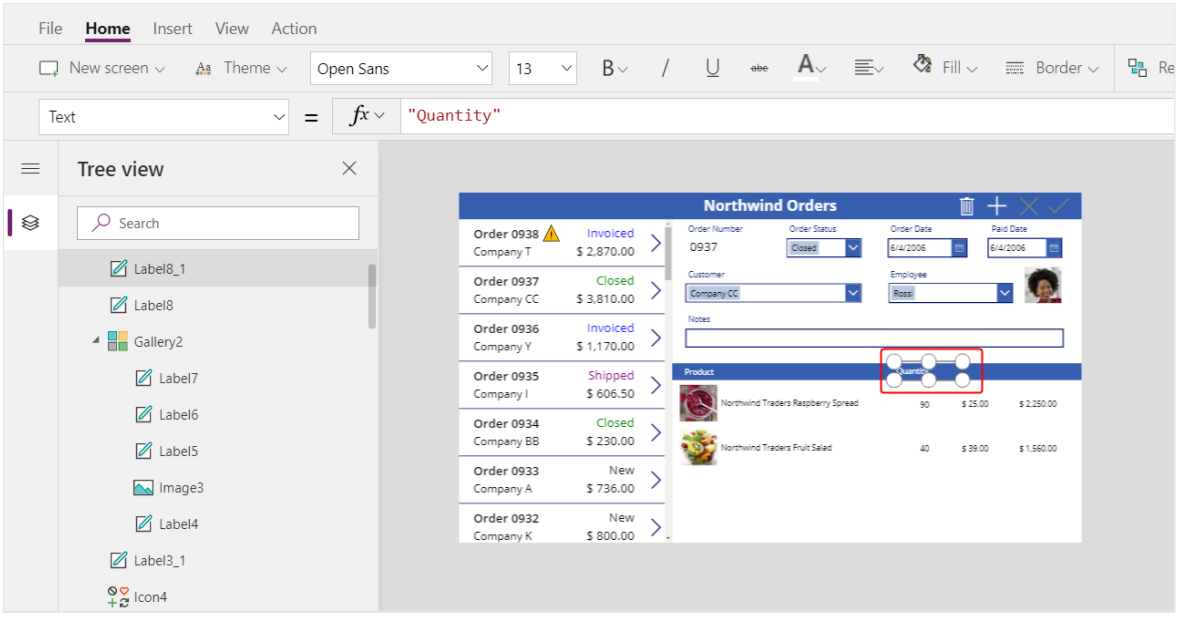
2. Resize and move the new label above the pictures of the products in the second title bar, and then change the text's color to white on the **Home** tab.

3. Double-click the label's text, and then type **Product**:



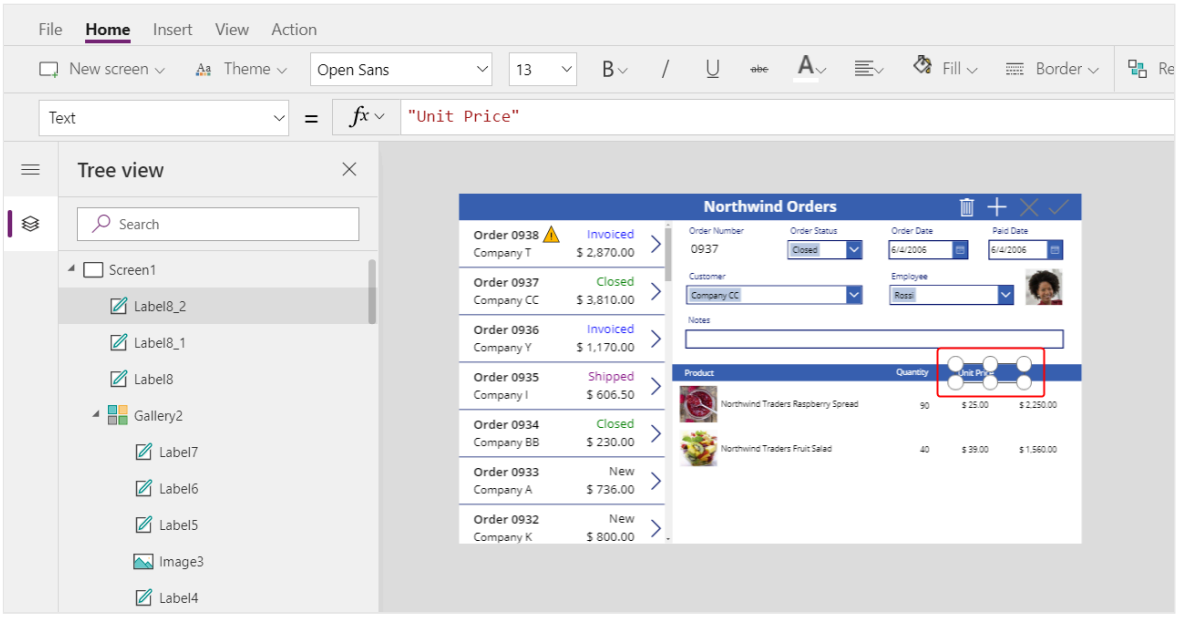
4. Copy and paste the product label, and then resize and move the copy above the quantity column.

5. Double-click the new label's text, and then type **Quantity**:



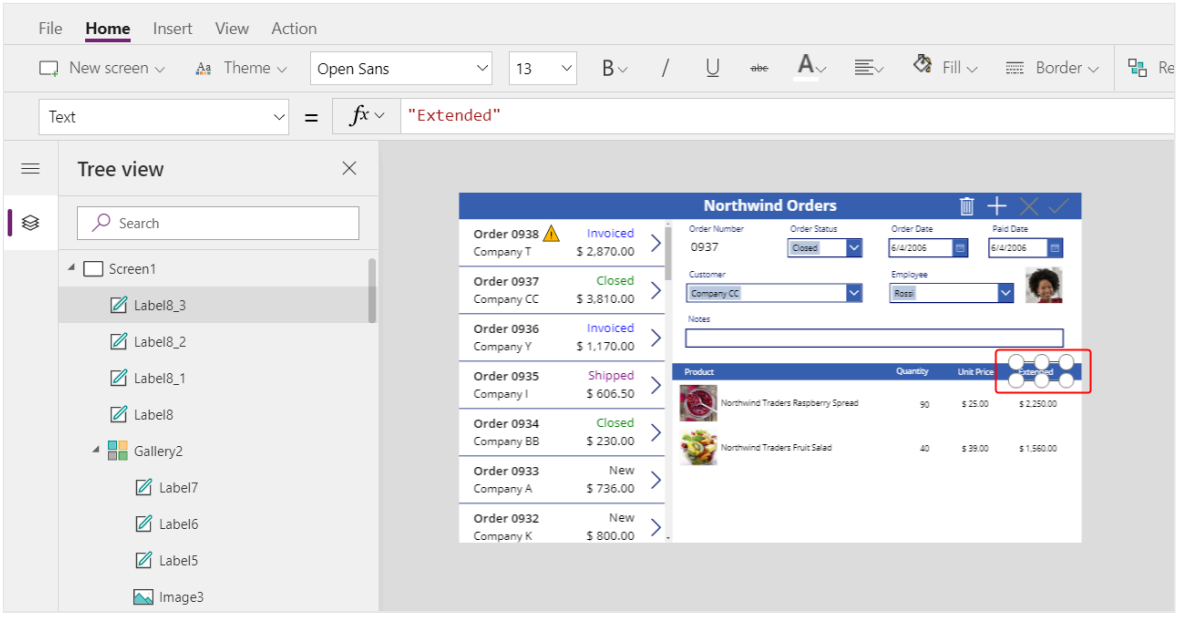
6. Copy and paste the quantity label, and then resize and move the copy above the unit-price column.

7. Double-click the new label's text, and then type **Unit Price**:



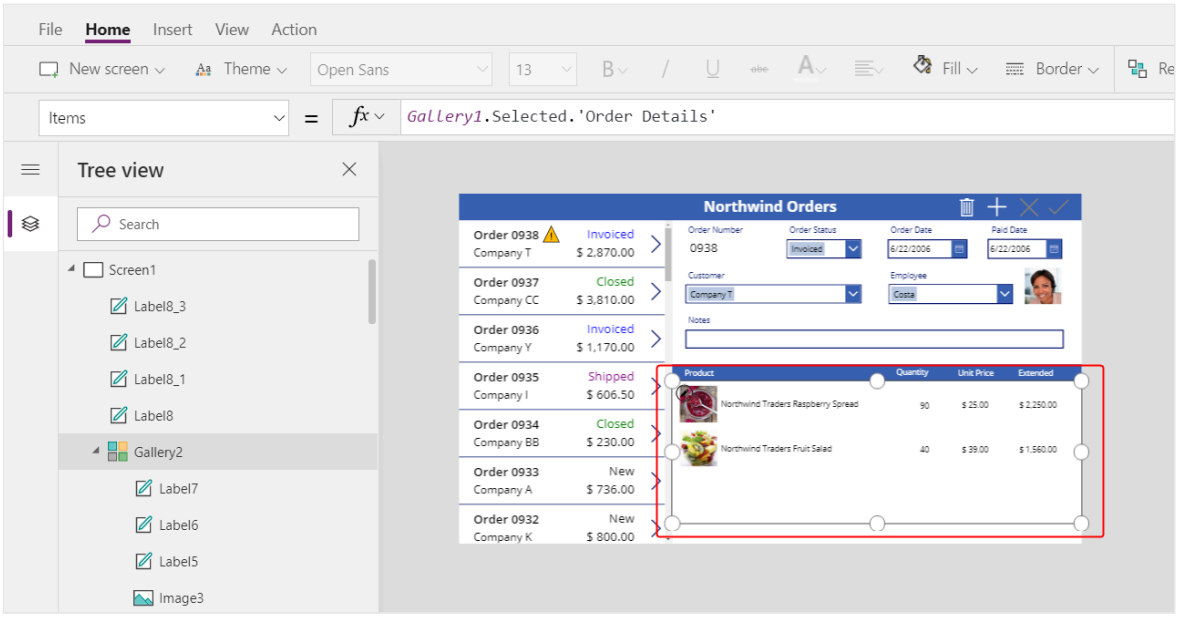
8. Copy and paste the unit-price label, and then resize and move the copy above the extended-price column.

9. Double-click the text of the new label, and then type **Extended**:

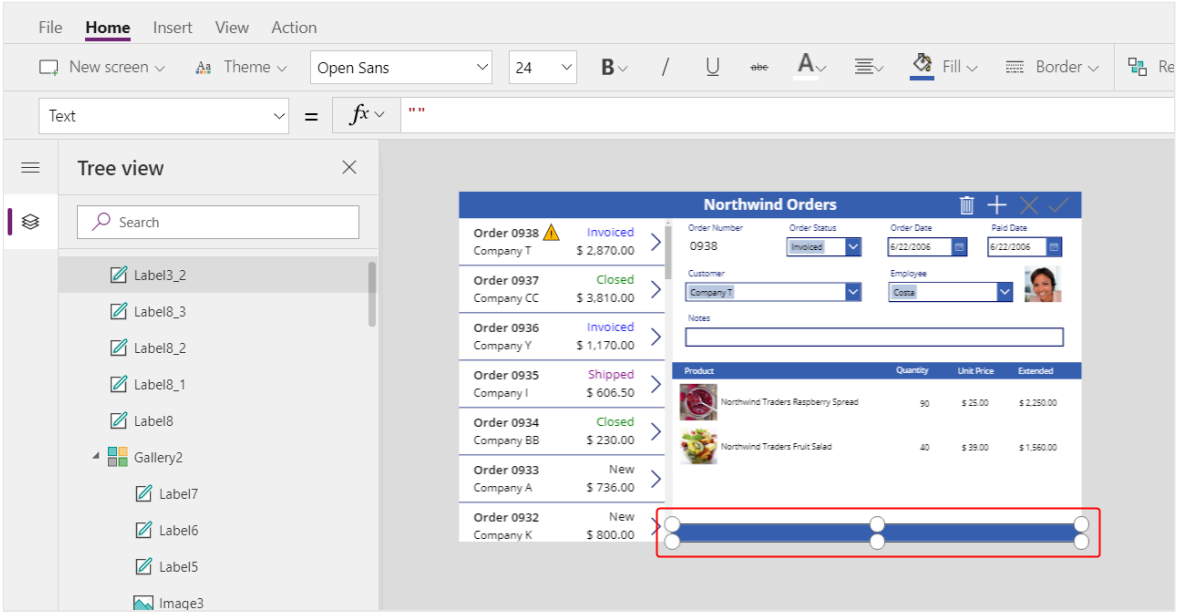


Display order totals

1. Reduce the height of the detail gallery to make room for the order totals at the bottom of the screen:

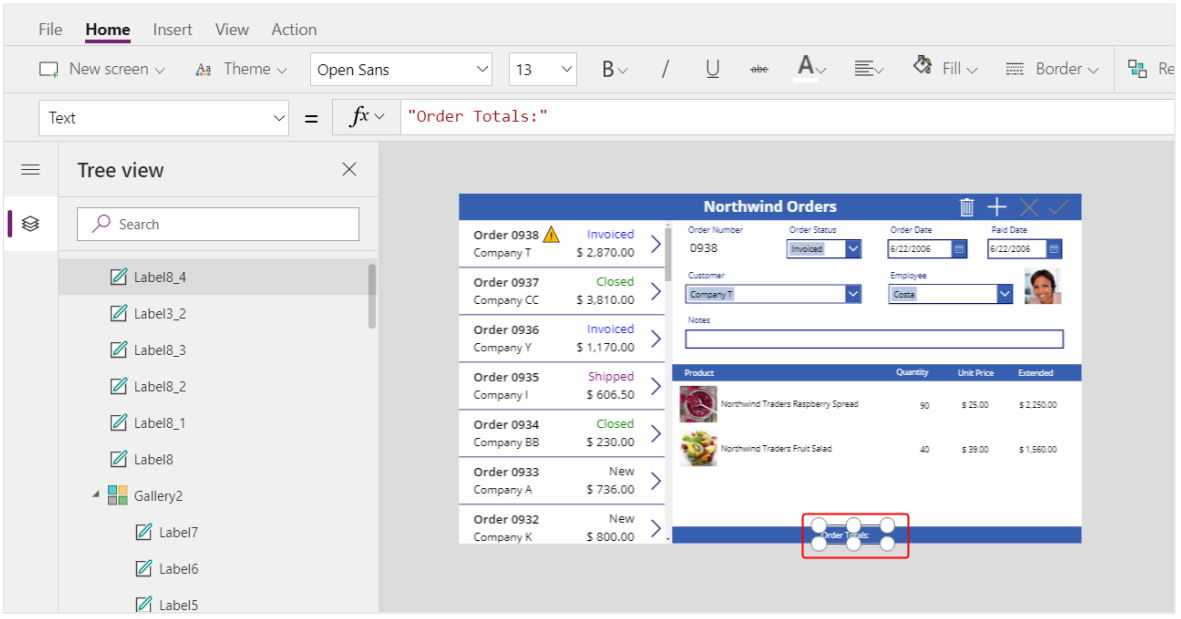


2. Copy and paste the title bar in the middle of the screen, and then move the copy to the bottom of the screen:



3. Copy and paste the product label from the middle title bar, and then move the copy to the bottom title bar, just to the left of the **Quantity** column.

4. Double-click the new label's text, and then type this text:
   Order Totals:



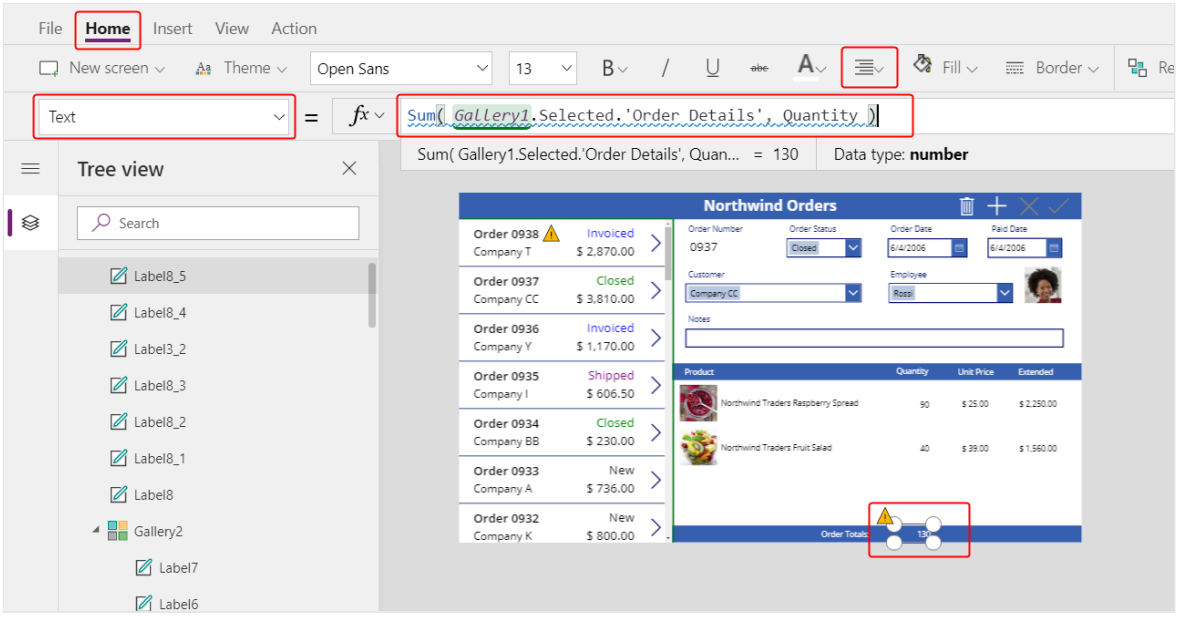
5. Copy and paste the order-totals label, and then resize and move the copy to the right of the order-totals label.

6. Set the new label's **Text** property to this formula:

```
Power Fx

Sum( Gallery1.Selected.'Order Details', Quantity )
```

   This formula shows a delegation warning, but you can ignore it because no single order will contain more than 500 products.

7. On the **Home** tab, set the new label's text alignment to **Right**:

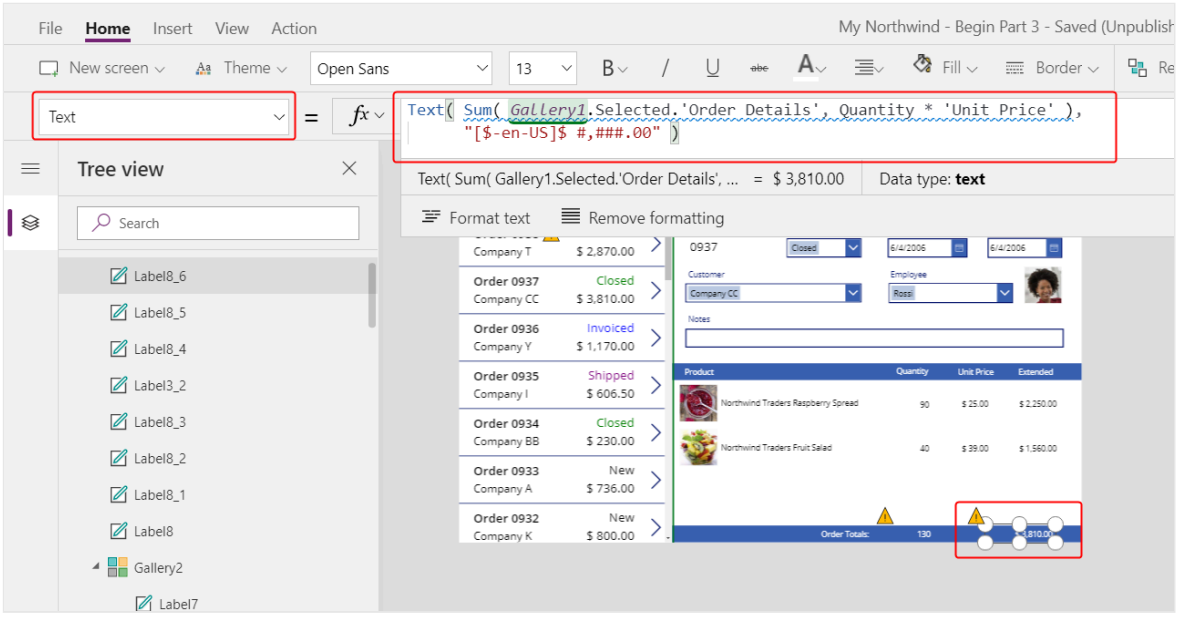


8. Copy and paste this label control, and then resize and move the copy under the **Extended** column.

9. Set the copy's **Text** property to this formula:

```
Power Fx

Text( Sum( Gallery1.Selected.'Order Details', Quantity * 'Unit Price'
), "[$-en-US]$ #,###.00" )
```

This formula shows a delegation warning, but you can ignore it because no single order will contain more than 500 products.

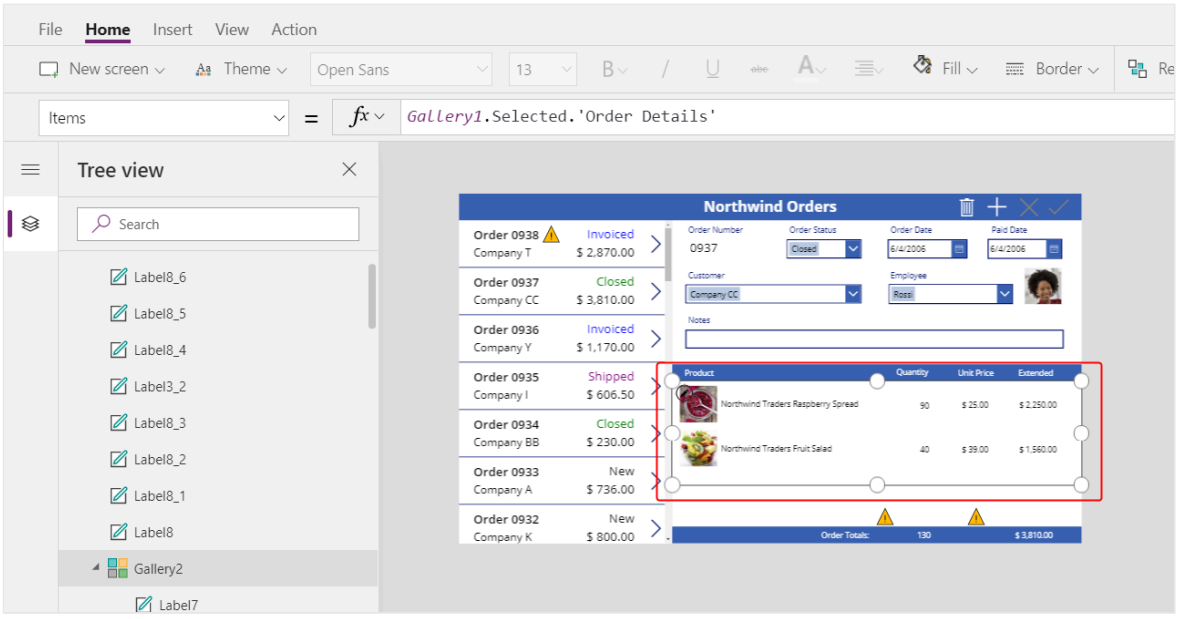


Add space for new details

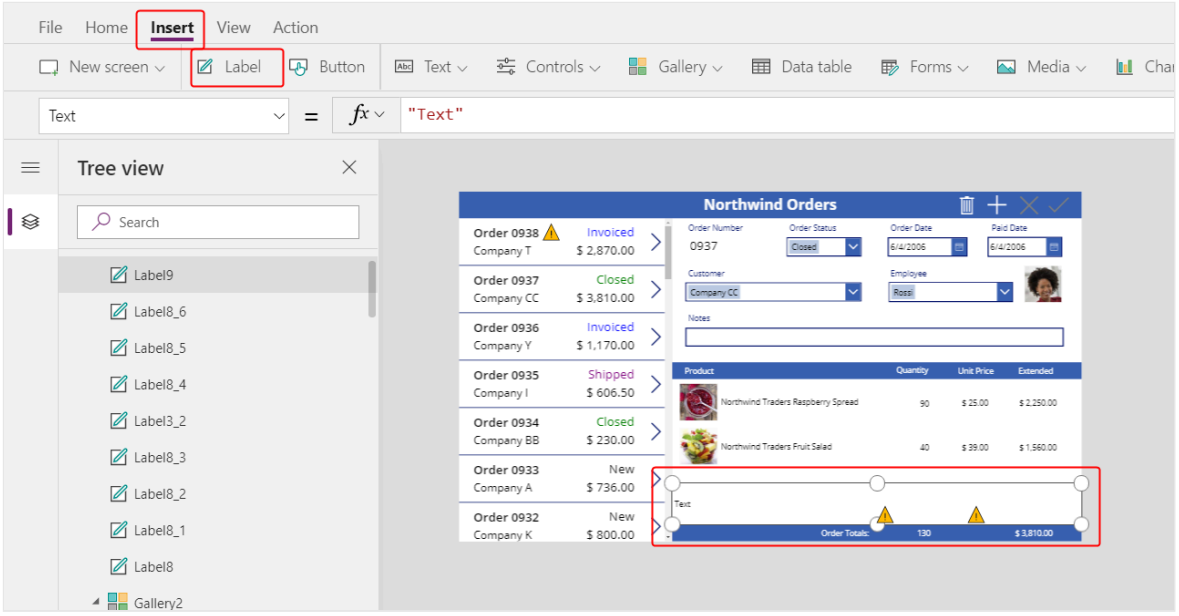
In any gallery, you can show data but you can't update it or add records. Under the detail gallery, you'll add an area where the user can configure a record in the **Order Details** table and insert that record into an order.

1. Reduce the height of the detail gallery enough to make room for a single-item editing space under that gallery.

   In this space, you'll add controls so that the user can add an order detail:

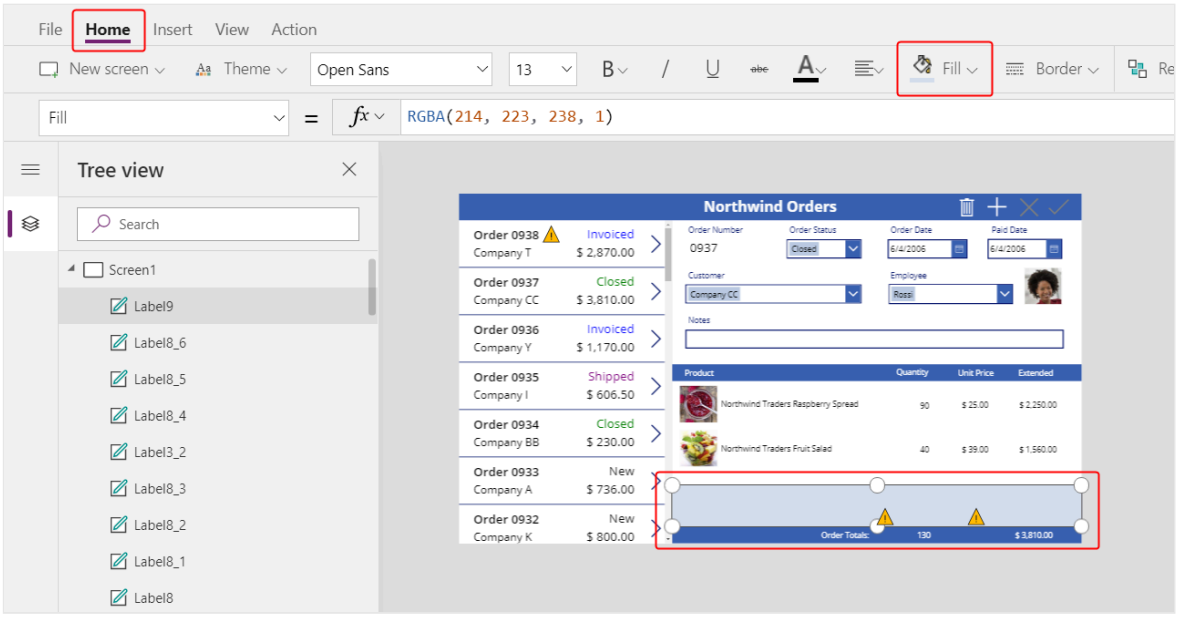


2. On the **Insert** tab, insert a label, and then resize and move it under the detail gallery.



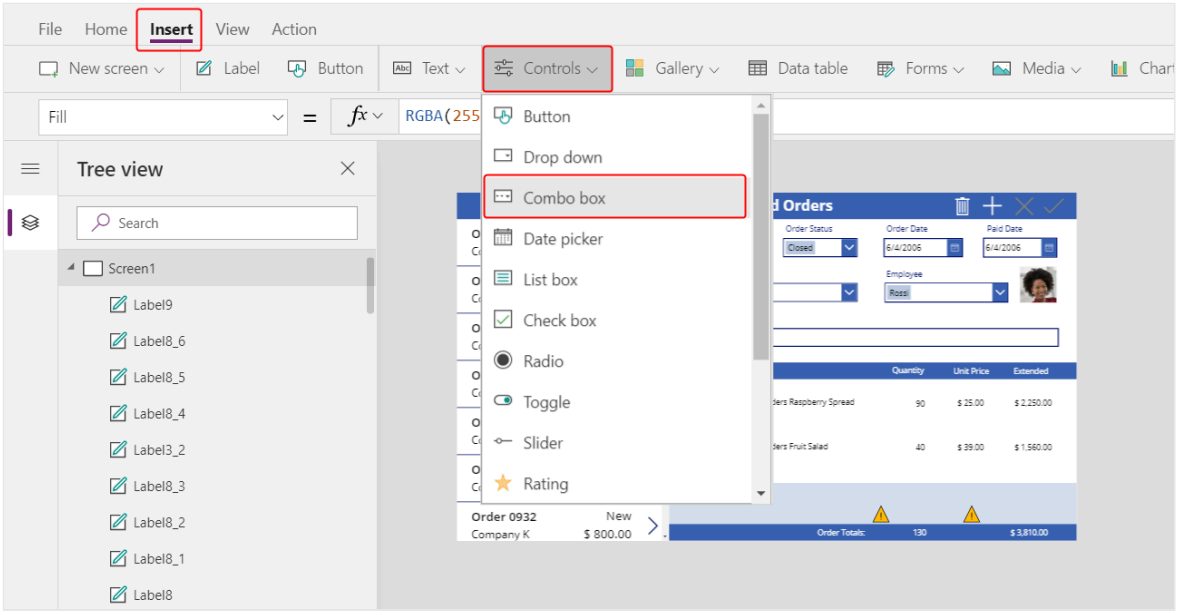
3. Double-click the text of the new label, and then press Delete.

4. On the **Home** tab, set the new label's **Fill** color to **LightBlue**:



Select a product

1. On the **Insert** tab, select **Controls** > **Combo box**:

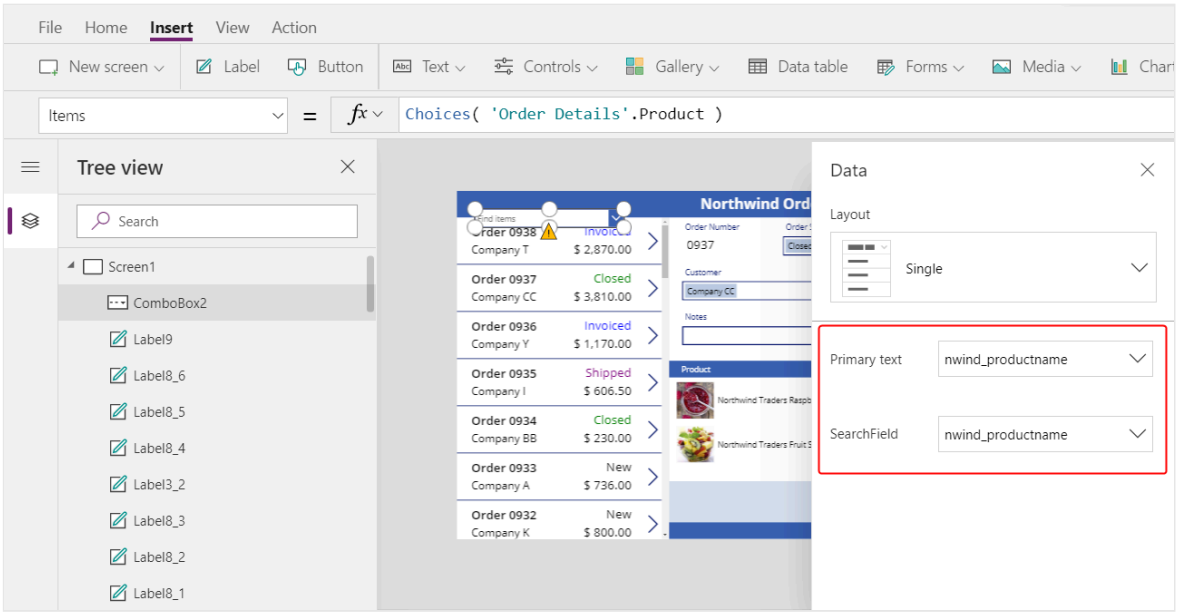


   The Combo box control appears in the upper-left corner.

2. In the fly out dialog, select the **Order Products** data source.

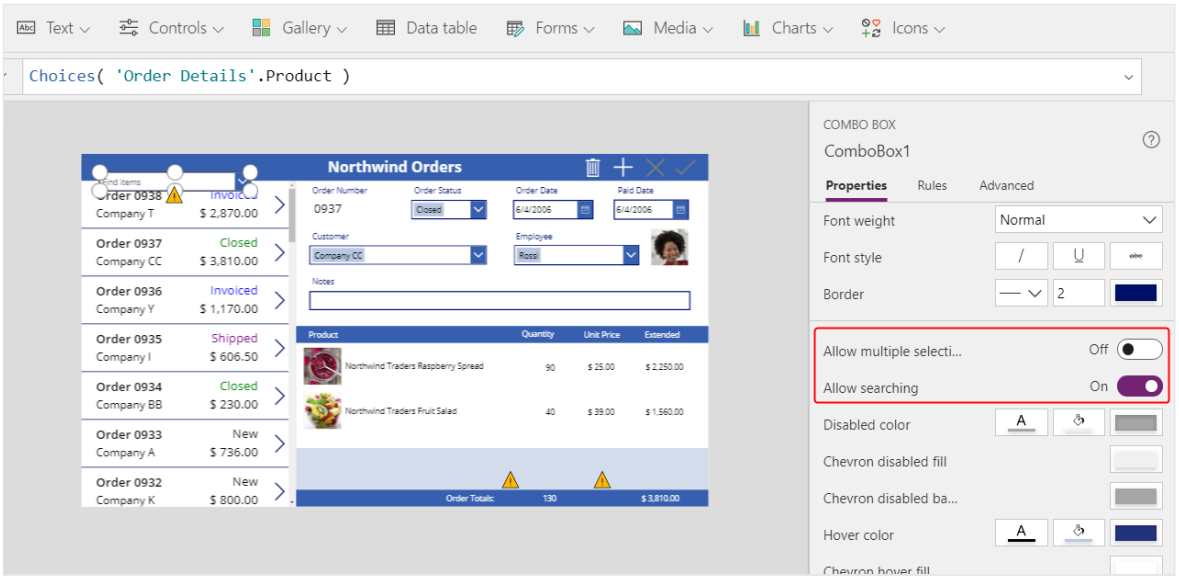
3. In the **Properties** tab for the combo box, select **Edit** (next to **Fields**) to open the **Data** pane. Ensure that the **Primary text** and **SearchField** are set to **nwind_productname**.

   You specify the logical name because the **Data** pane doesn't support display names in this case yet:

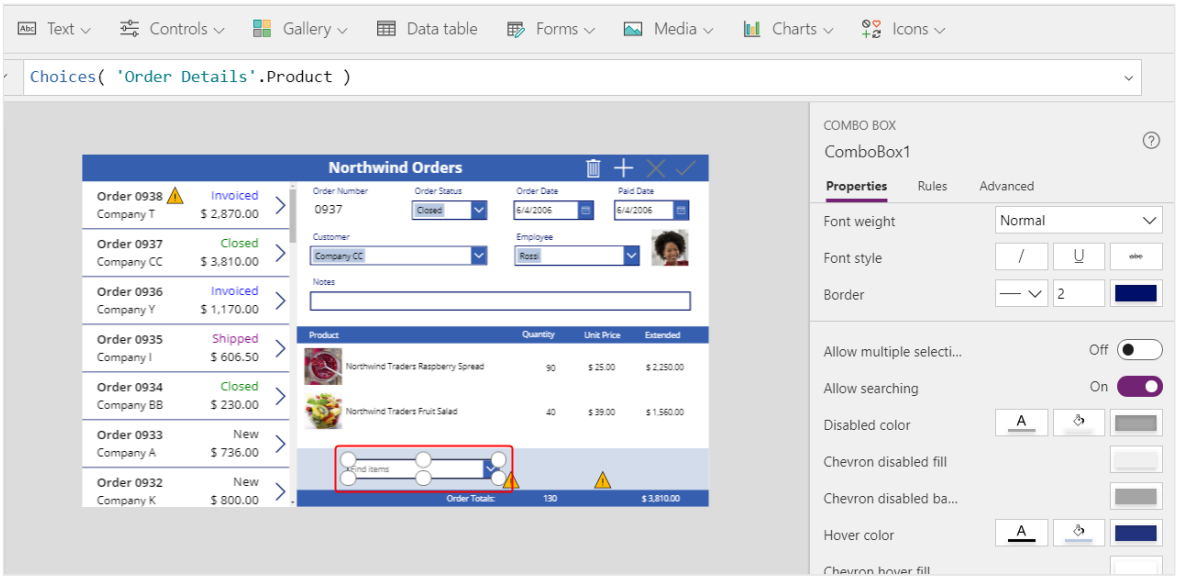


4. Close the **Data** pane.

5. In the **Properties** tab near the right edge, scroll down, turn off **Allow multiple selection**, and ensure that **Allow searching** is turned on:



6. Resize and move the combo box to the light-blue area, just under the product-name column in the detail gallery:

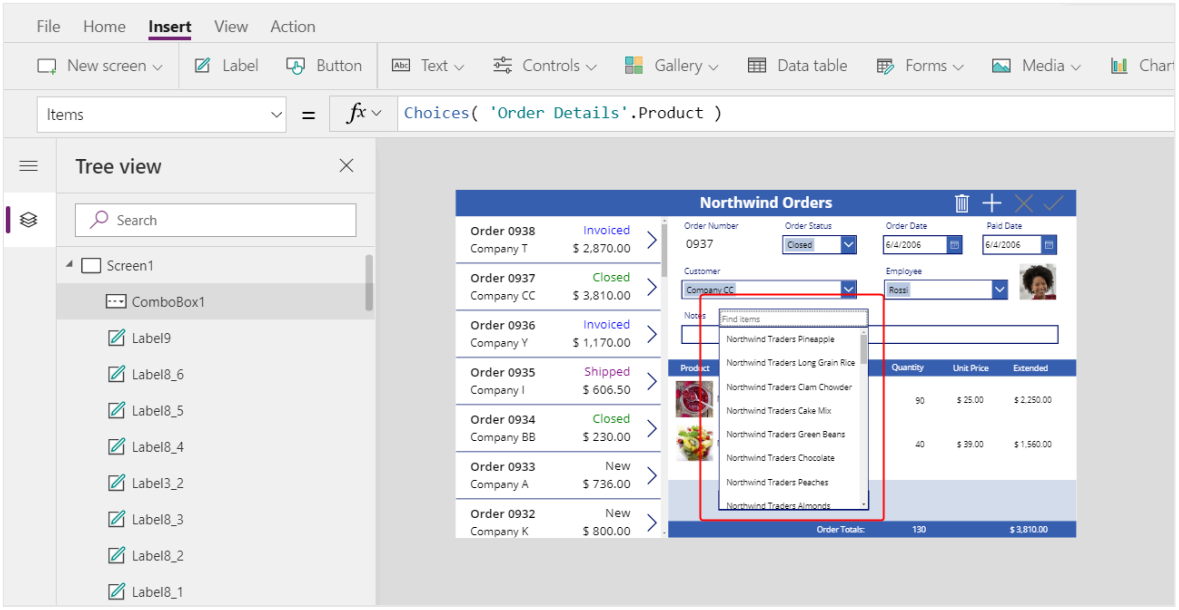


In this combo box, the user will specify a record in the **Product** table for the **Order Details** record that the app will create.

7. While holding down the Alt key, select the combo box's down arrow.

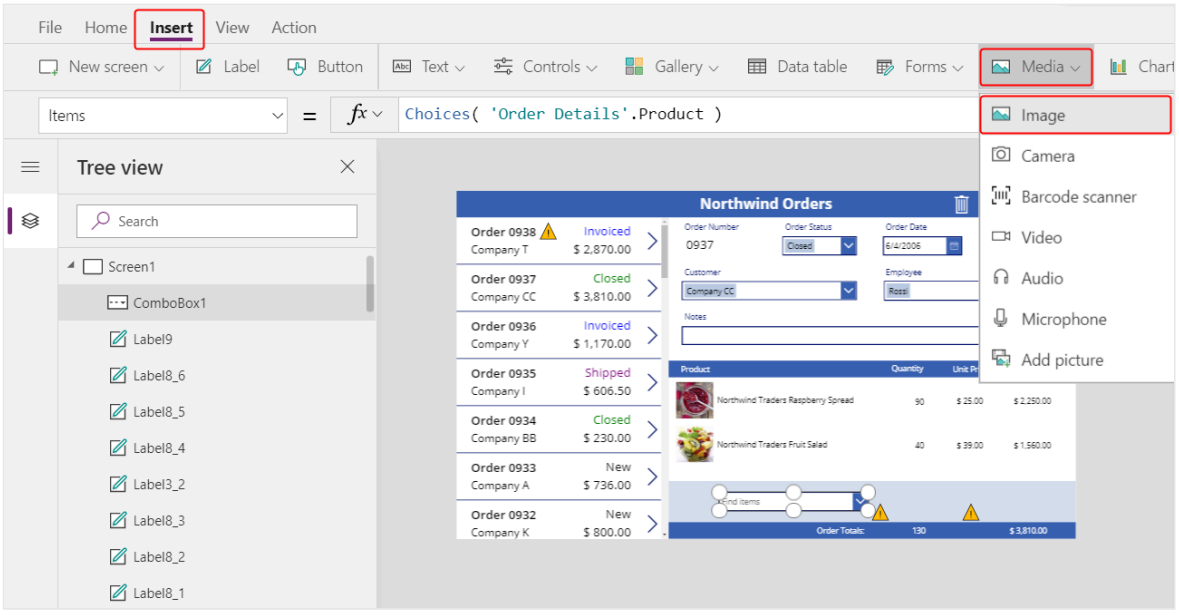
> **Tip**
>
> By holding down the Alt key, you can interact with controls in Power Apps Studio without opening Preview mode.

8. In the list of products that appears, select a product:

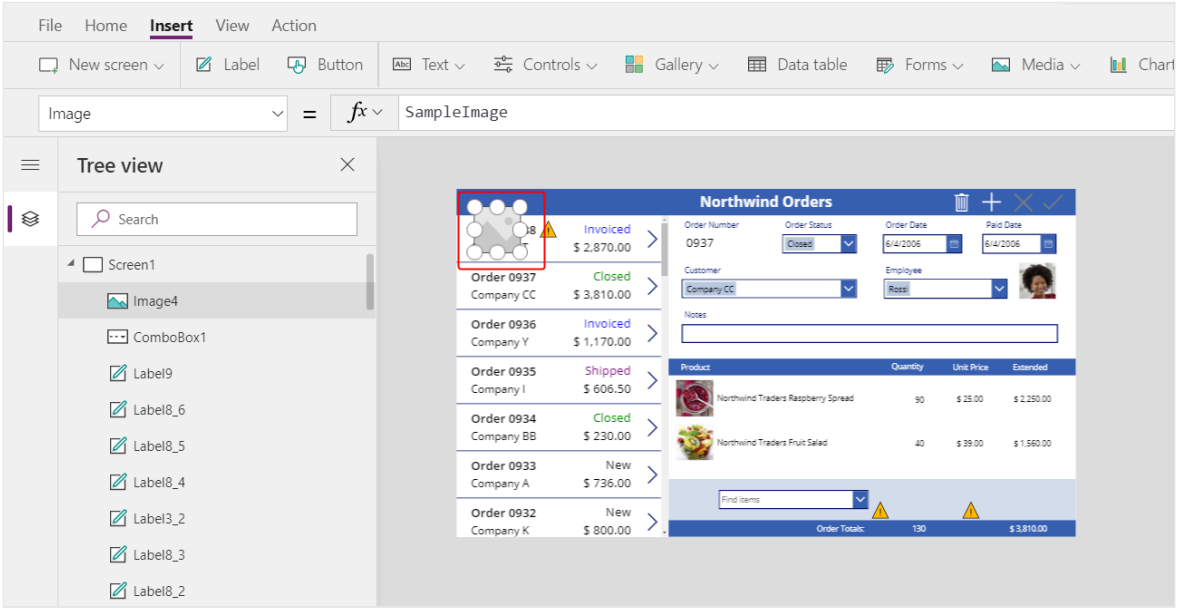


Add a product image

1. On the **Insert** tab, select **Media** > **Image**:



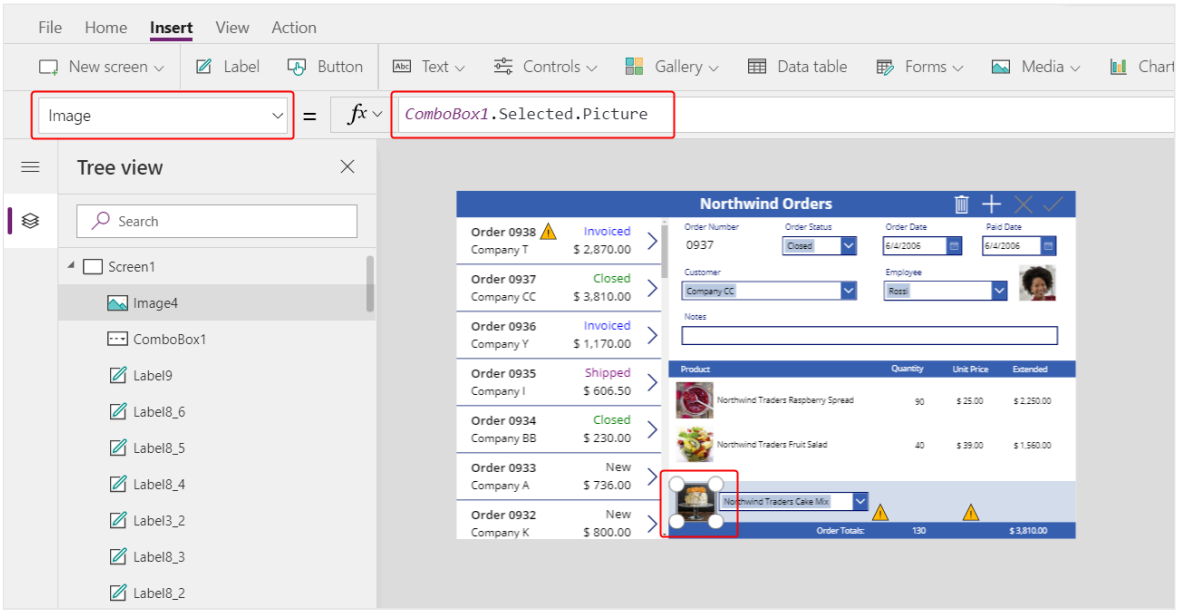
The **Image** control appears in the upper-left corner:



2. Resize and move the image to the light-blue area under the other product images and next to the combo box.

3. Set the **Image** property of the image to:

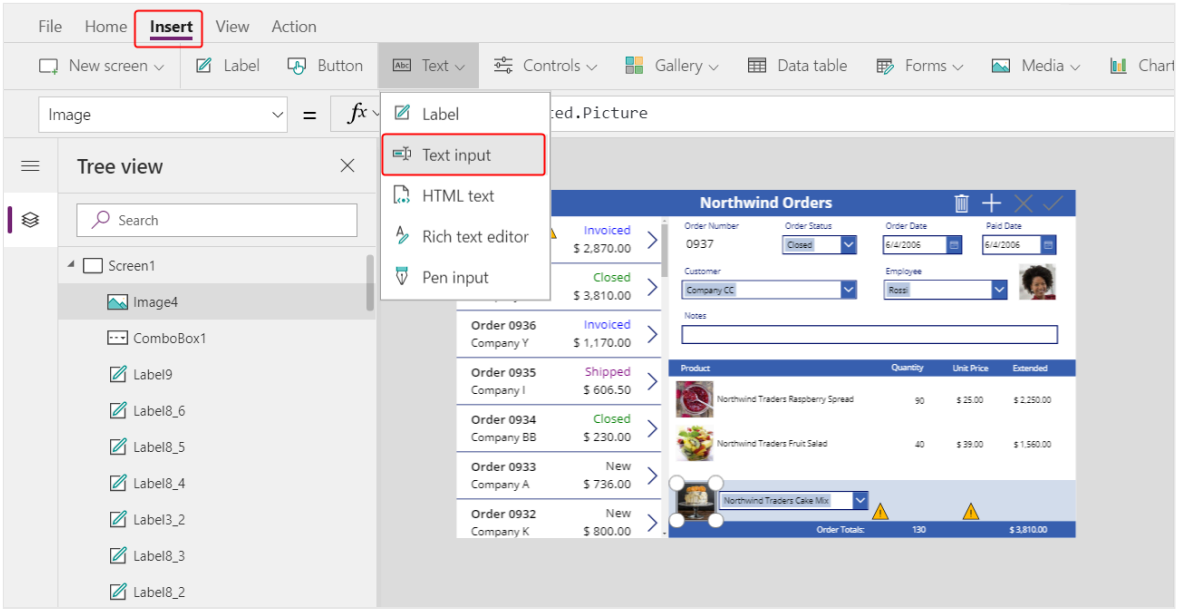
```
ComboBox1.Selected.Picture
```



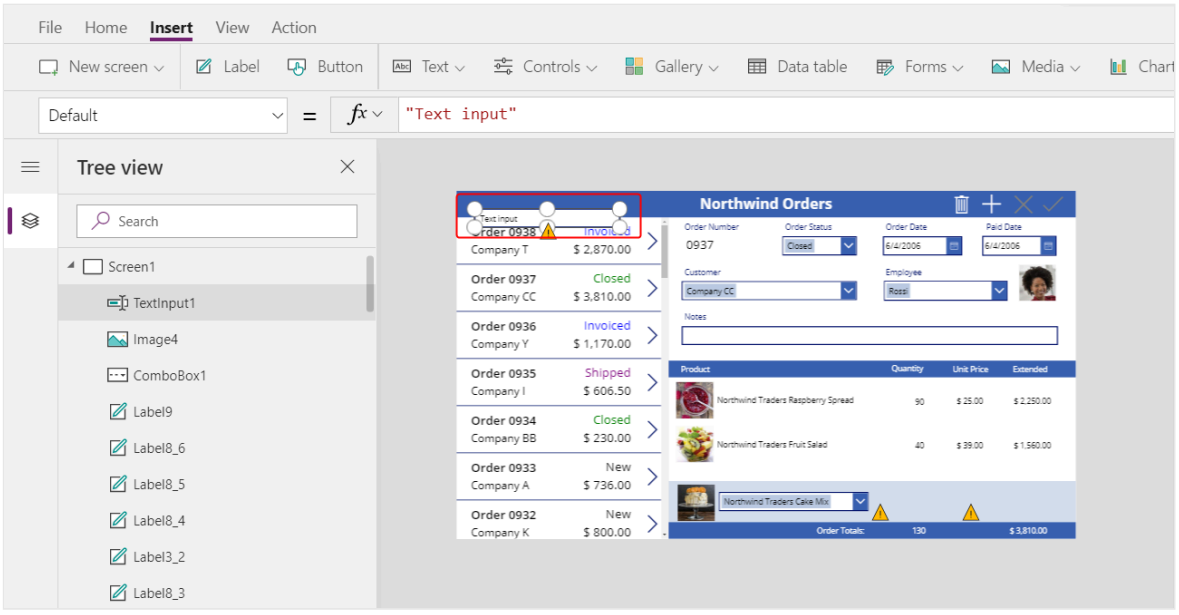
You're using the same trick as you used to show the employee picture in the summary form. The **Selected** property of the combo box returns the entire record of whatever product the user selects, including the **Picture** field.

Add a quantity box

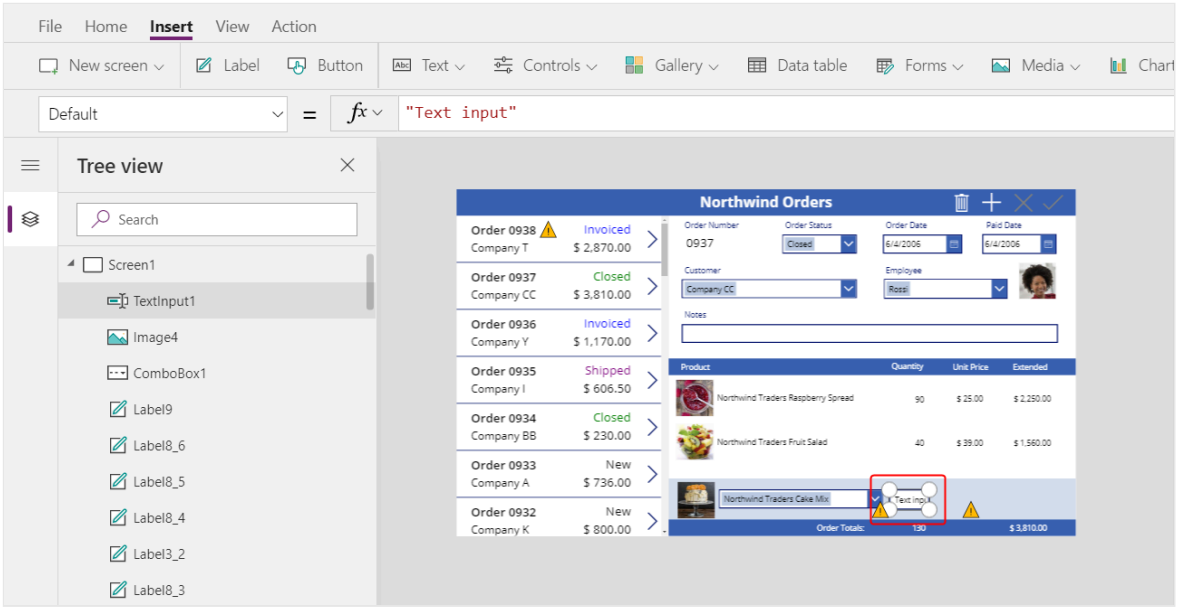
1. On the **Insert** tab, select **Text** > **Text input**:



The Text input control appears in the upper-left corner:

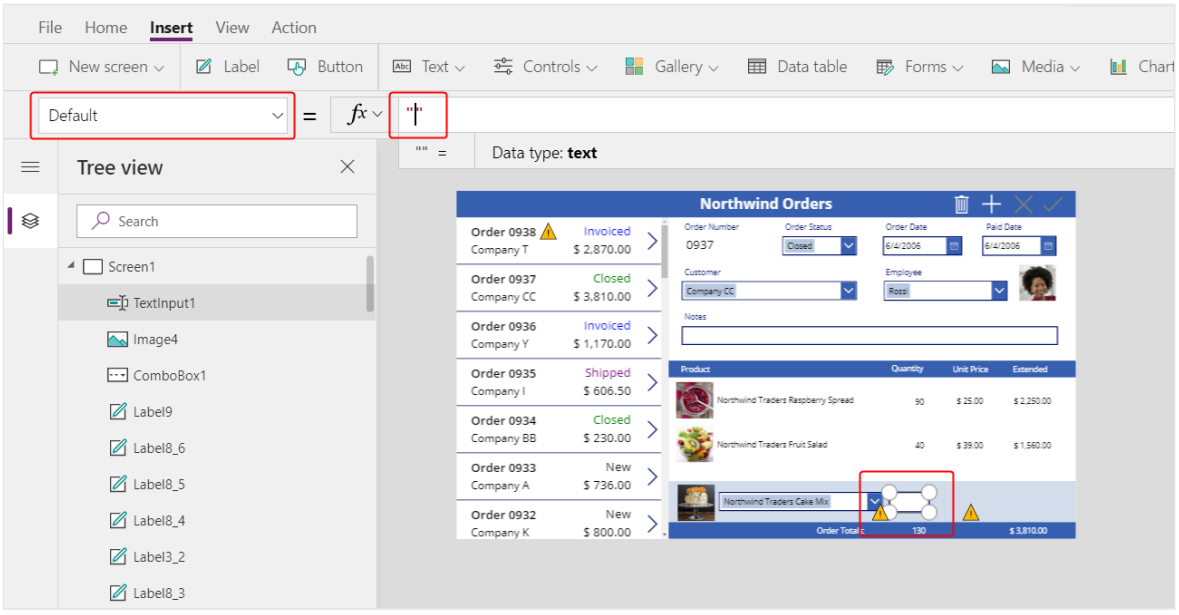


2. Resize and move the text-input box to the right of the combo box, under the quantity column in the detail gallery:

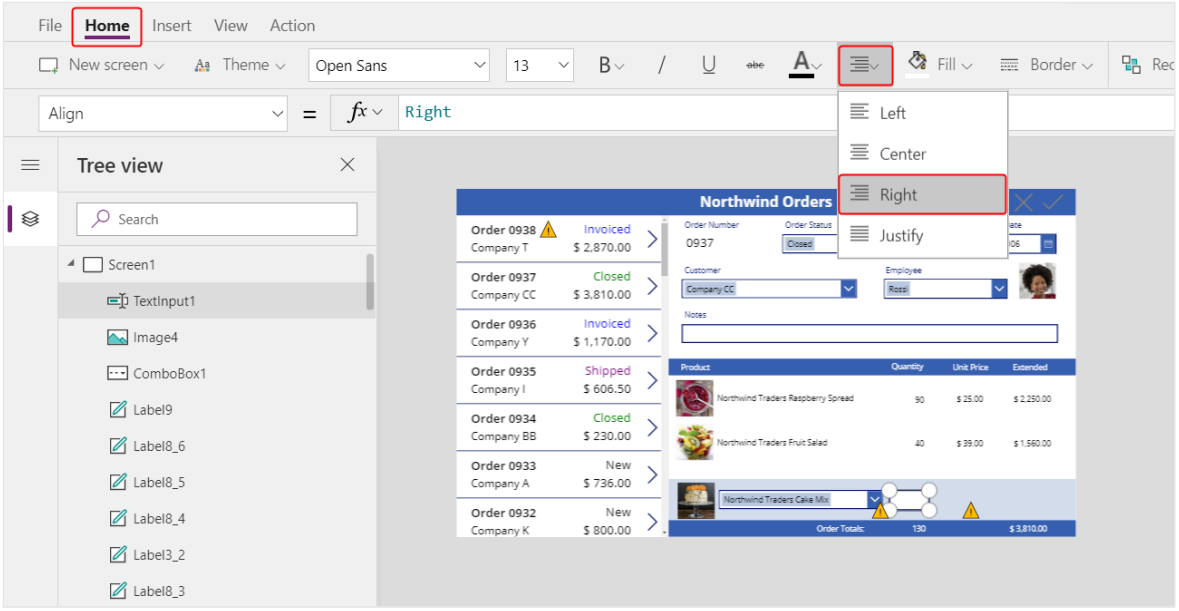


By using this text-input box, the user will specify the **Quantity** field of the **Order Details** record.

3. Set the **Default** property of this control to "":



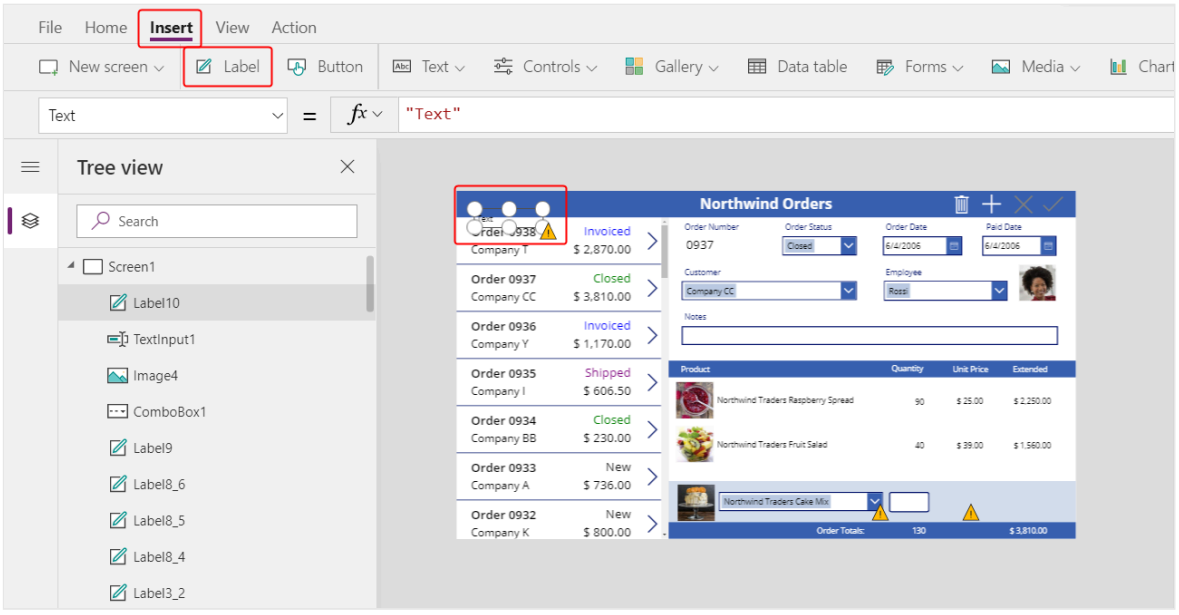
4. On the **Home** tab, set the text alignment of this control to **Right**:



Show the unit and extended prices

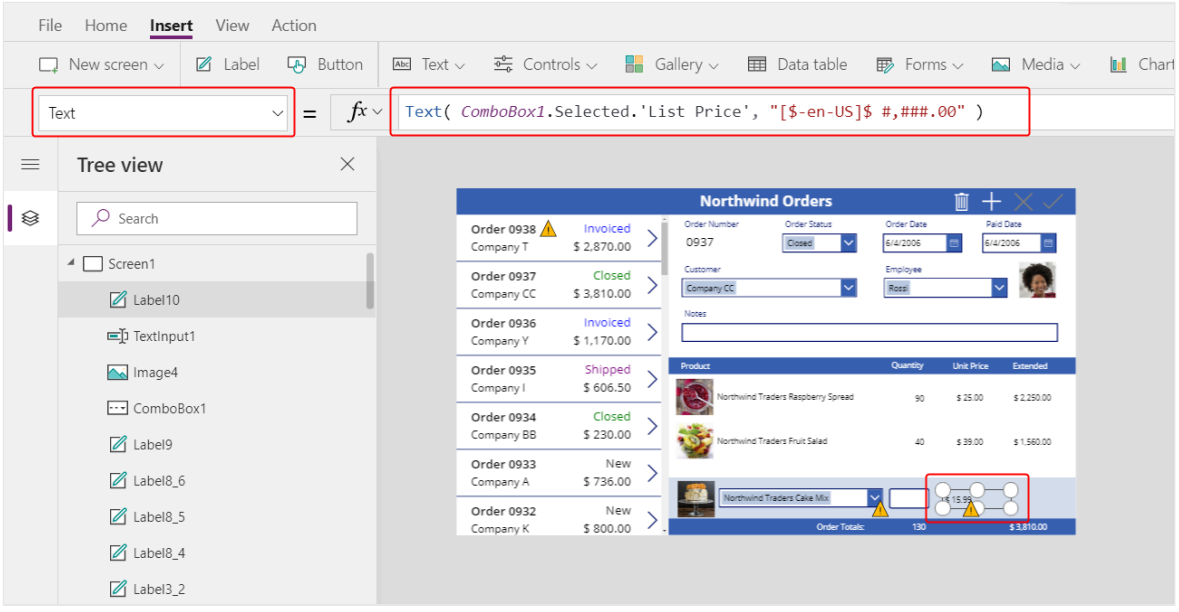
1. On the **Insert** tab, insert a **Label** control.

   The label appears in the upper-left corner of the screen:



2. Resize and move the label to the right of the text-input control, and set the label's **Text** property to this formula:

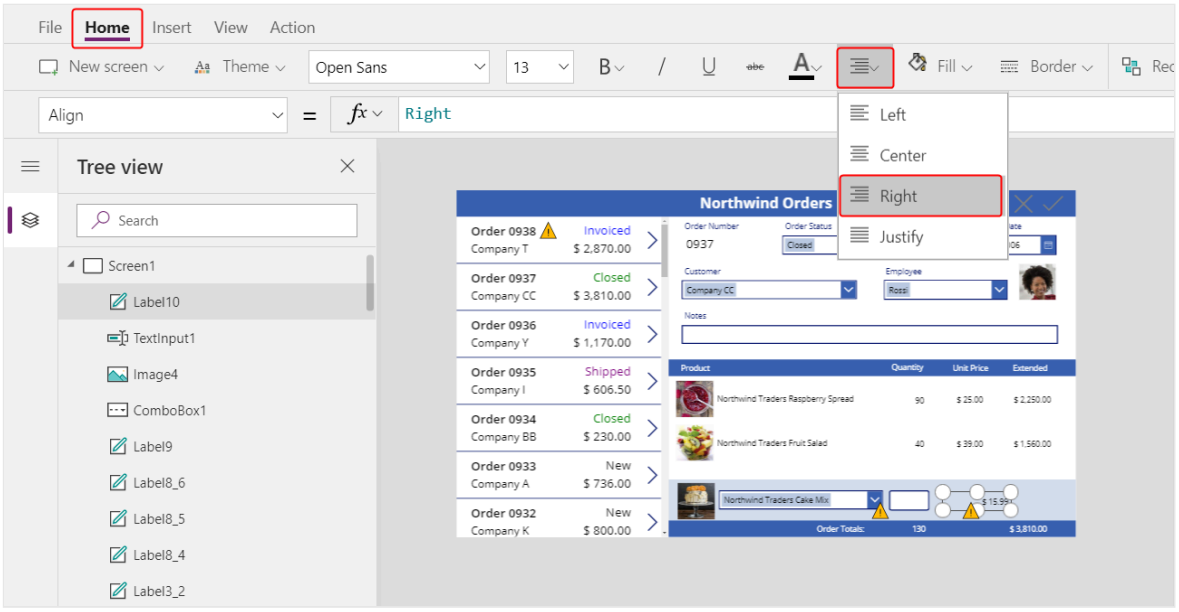
```
Power Fx

Text( ComboBox1.Selected.'List Price', "[$-en-US]$ #,###.00" )
```



This control shows the **List Price** from the **Order Products** table. This value will determine the **Unit Price** field in the **Order Details** record.

> ⓘ **Note**
>
> For this scenario, the value is read-only, but other scenarios might call for the app user to modify it. In that case, use a **Text input** control, and set its **Default** property to **List Price**.

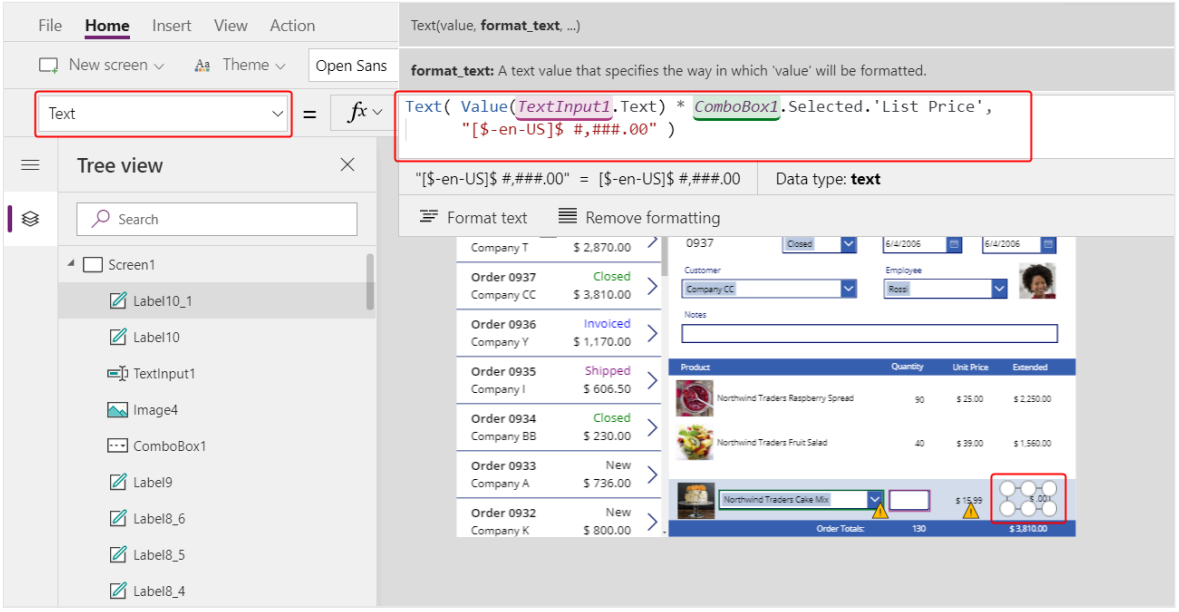
3. On the **Home** tab, set the text alignment of the list-price label to **Right**:



4. Copy and paste the list-price label, and then resize and move the copy to the right of the list-price label.

5. Set the new label's **Text** property to this formula:

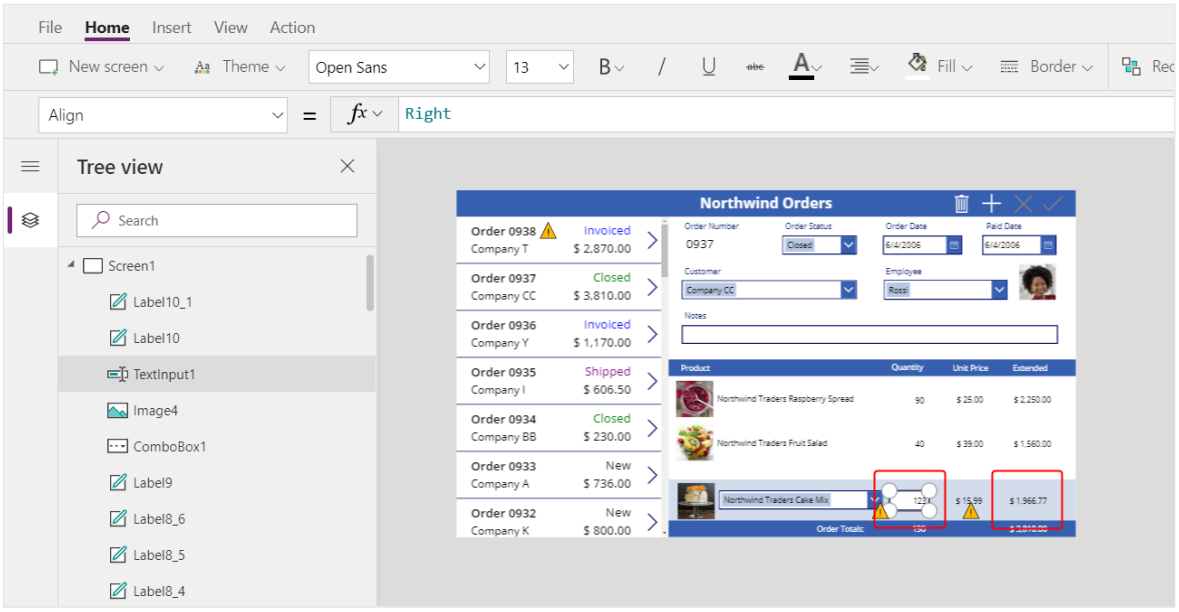
```
Power Fx

Text( Value(TextInput1.Text) * ComboBox1.Selected.'List Price', "[$-en-US]$ #,###.00" )
```



This control shows the extended price based on the quantity that the app user specified and the list price of the product that the app user selected. It's purely informational for the app user.

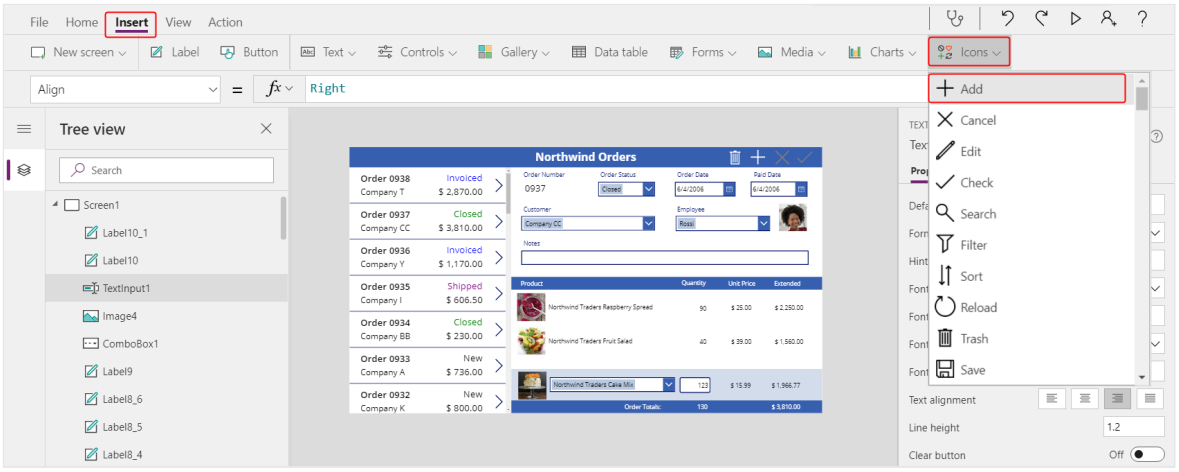
6. Double-click the text-input control for quantity, and then type a number.

The **Extended** price label recalculates to show the new value:

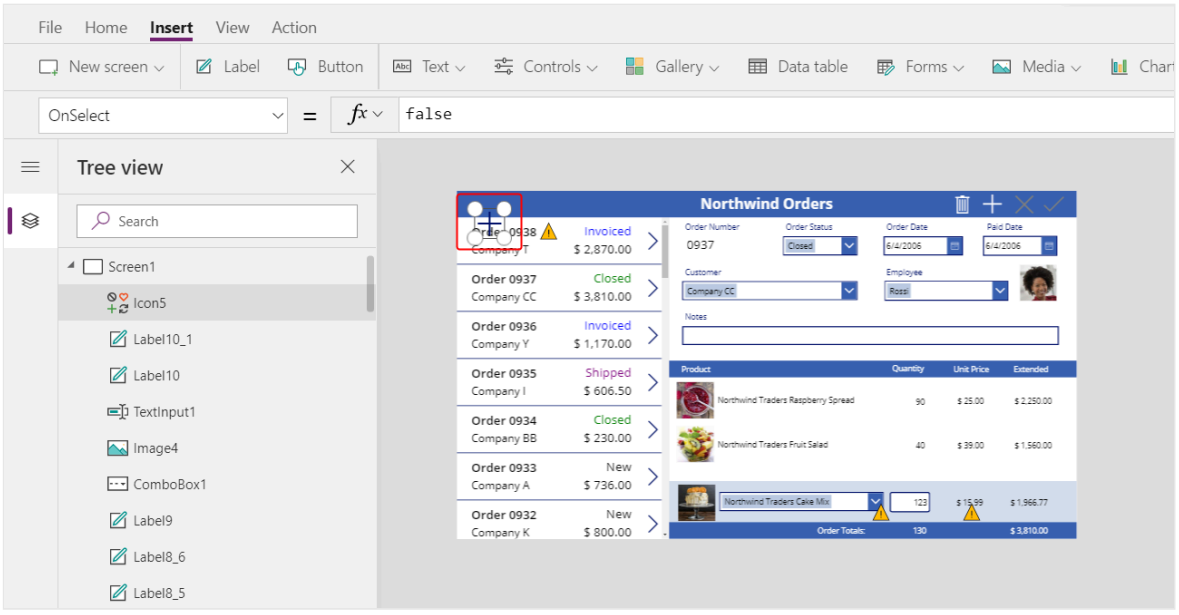


Add an Add icon

1. On the **Insert** tab, select **Icons** > **Add**:



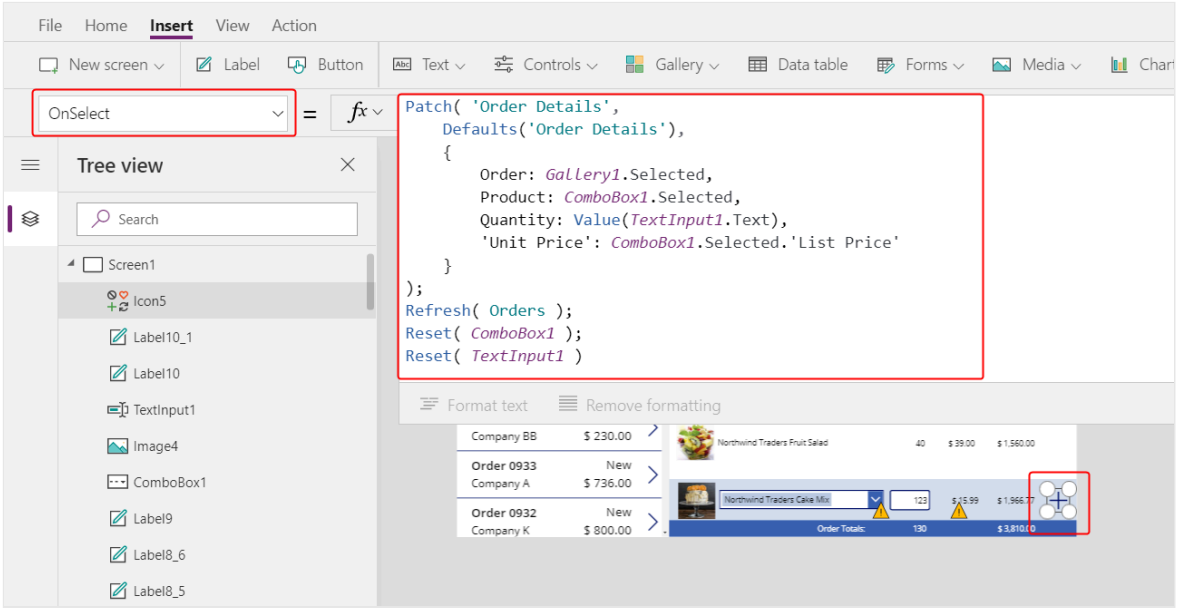
The icon appears in the upper-left corner of the screen.



2. Resize and move this icon to the right edge of the light-blue area, and then set the icon's **OnSelect** property to this formula:

Power Fx

```
Patch( 'Order Details',
    Defaults('Order Details'),
    {
        Order: Gallery1.Selected,
        Product: ComboBox1.Selected,
        Quantity: Value(TextInput1.Text),
        'Unit Price': ComboBox1.Selected.'List Price'
    }
);
Refresh( Orders );
Reset( ComboBox1 );
Reset( TextInput1 )
```



In general, the **Patch** function updates and creates records, and the specific arguments in this formula determine the exact changes that the function will make.

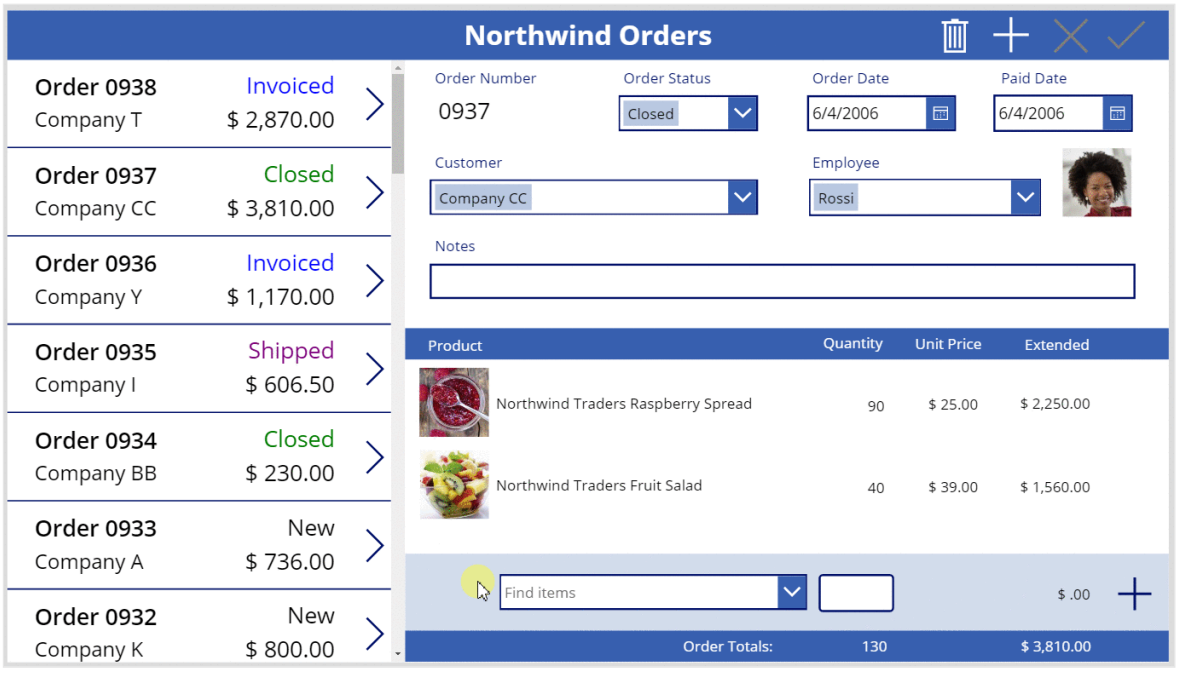
- The first argument specifies the data source (in this case, the **Order Details** table) in which the function will update or create a record.
- The second argument specifies that the function will create a record with the default values for the **Order Details** table unless otherwise specified in the third argument.
- The third argument specifies that four columns in the new record will contain values from the user.
  - The **Order** column will contain the number of the order that the user selected in order gallery.
  - The **Product** column will contain the name of the product that the user selected in the combo box that shows products.
  - The **Quantity** column will contain the value that the user specified in the text-input box.
  - The **Unit Price** column will contain the list price of the product that the user selected for this order detail.

> ⓘ Note
>
> You can build formulas that use data from any column (in the **Order Products** table) for whatever product the app user selects in the combo box that shows products. When the user selects a record in the **Order Products** table, not only does the product's name appear in that combo box but also the product's unit price appears in a label. Each lookup value in a canvas app references an entire record, not just a primary key.

The **Refresh** function ensures that the **Orders** table reflects the record that you've just added to the **Order Details** table. The **Reset** function clears the product, quantity, and unit-price data so that the user can more easily create another order detail for the same order.

3. Press F5, and then select the **Add** icon.

   The order reflects the information that you specified:

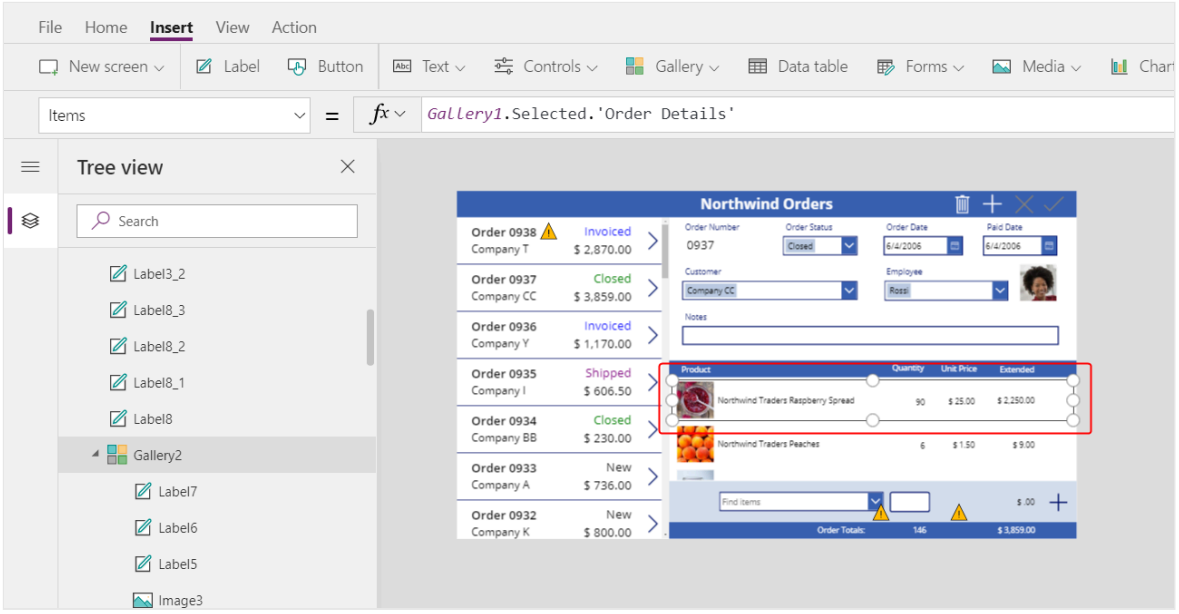


4. (optional) Add another item to the order.

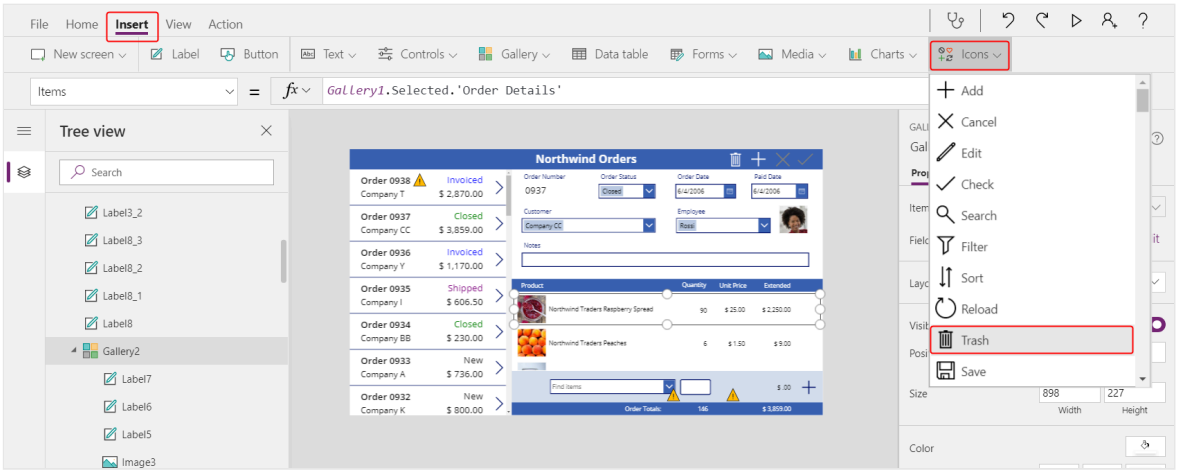
5. Press Esc to close Preview mode.

Remove an order detail

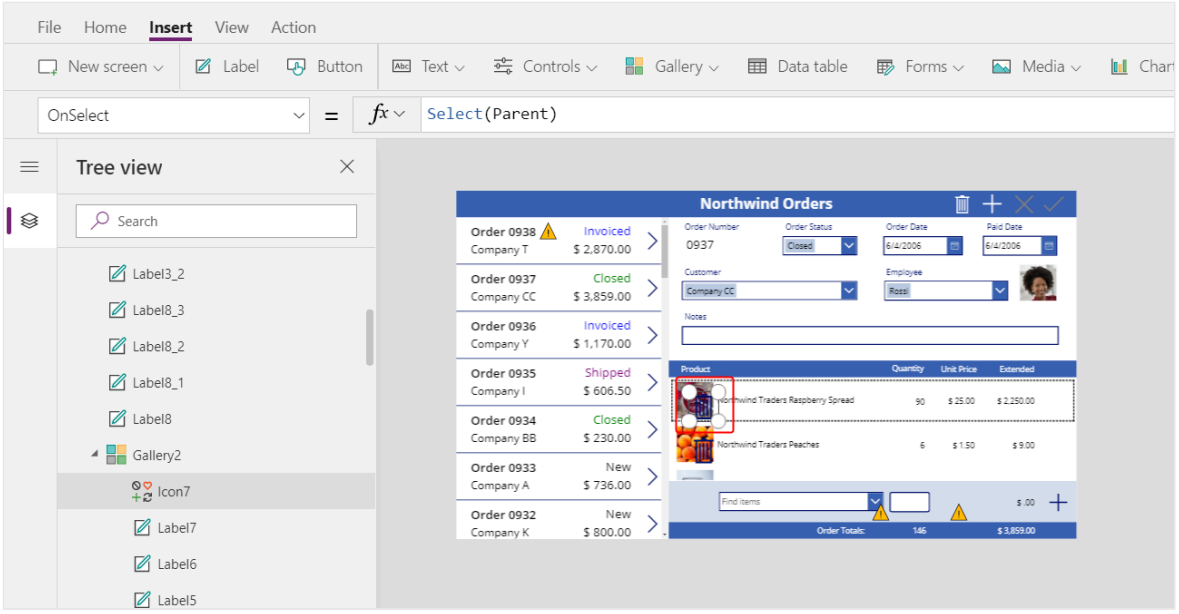
1. In the center of the screen, select the template of the detail gallery:



2. On the **Insert** tab, select **Icons** > **Trash**:



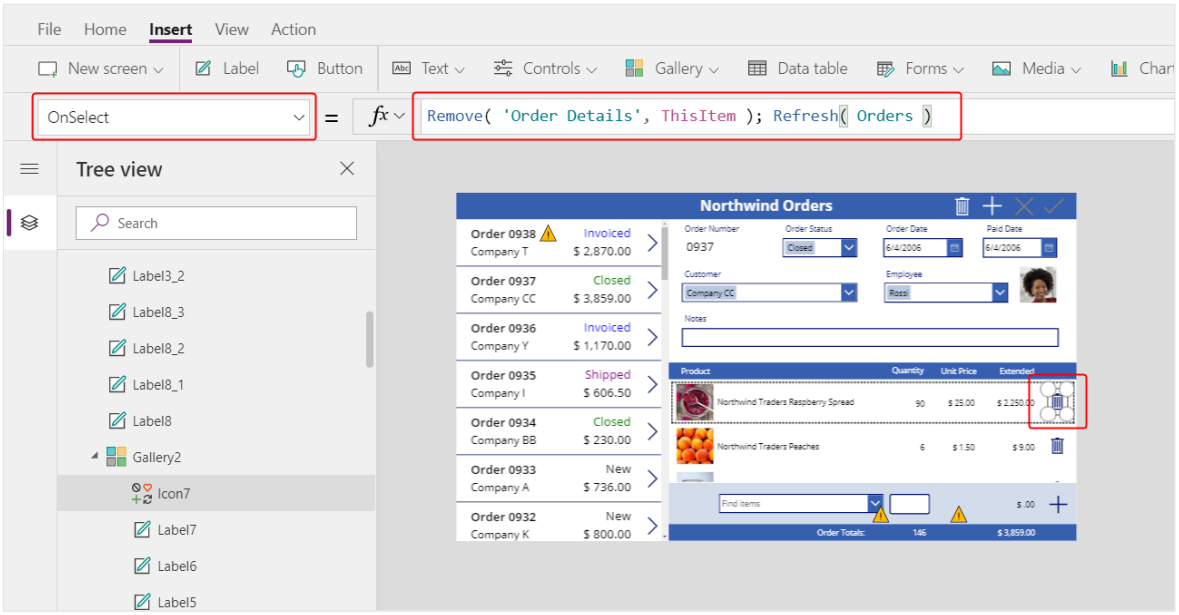
The Trash icon appears in the upper-left corner of gallery's template.



3. Resize and move the Trash icon to the right side of the detail gallery's template, and set the icon's **OnSelect** property to this formula:

Power Fx

```
Remove( 'Order Details', ThisItem ); Refresh( Orders )
```

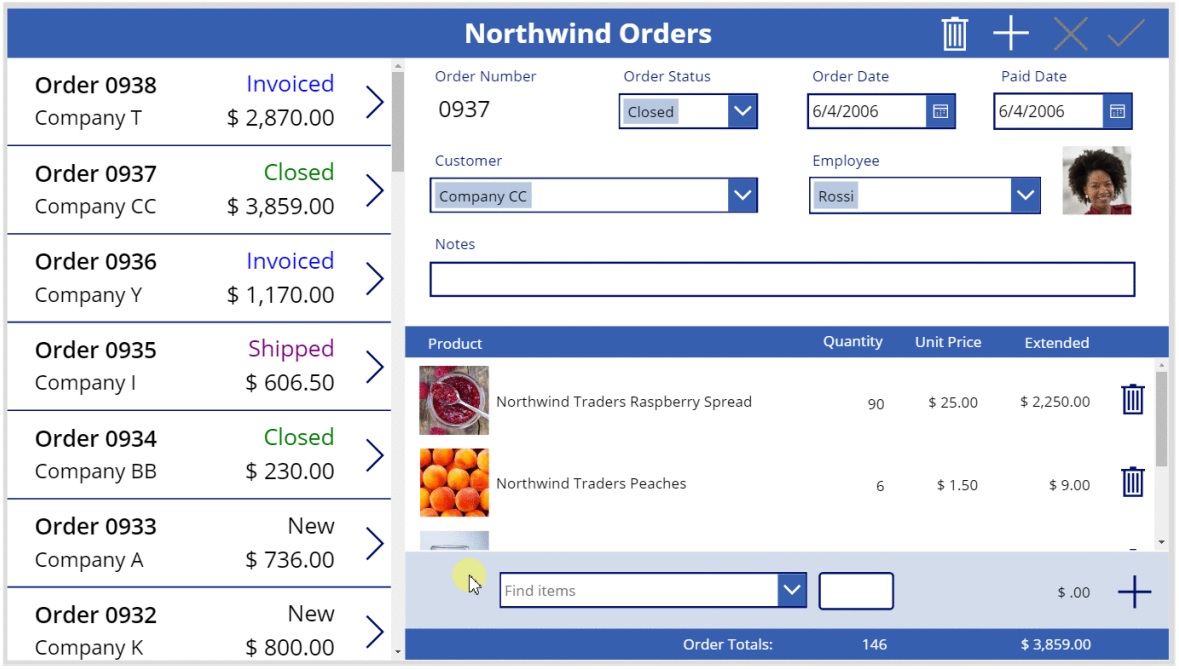


As of this writing, you can't remove a record directly from a relationship, so the **Remove** function removes a record directly from the related table. **ThisItem** specifies the record to remove, taken from the same record in the detail gallery where the Trash icon appears.

Again, the operation uses cached data, so the **Refresh** function informs the **Orders** table that the app has changed one of its related tables.

4. Press F5 to open Preview mode, and then select the Trash icon next to each **Order Details** record that you want to remove from the order.

5. Try adding and removing various order details from your orders:



In conclusion

To recap, you added another gallery to show order details and controls adding and removing an order detail in the app. You used these elements:

- A second gallery control, linked to the order gallery through a one-to-many relationship: **Gallery2.Items** = `Gallery1.Selected.'Order Details'`
- A many-to-one relationship from the **Order Details** table to the **Order Products** table: `ThisItem.Product.'Product Name'` and `ThisItem.Product.Picture`
- The **Choices** function to get a list of products: `Choices( 'Order Details'.Product' )`
- The **Selected** property of a combo box as the complete many-to-one related record: `ComboBox1.Selected.Picture` and `ComboBox1.Selected.'List Price'`
- The **Patch** function to create an **Order Details** record: `Patch( 'Order Details', Defaults( 'Order Details' ), ... )`
- The **Remove** function to delete an **Order Details** record: `Remove( 'Order Details', ThisItem )`

This series of topics has been a quick walkthrough of using Dataverse relationships and choices in a canvas app for educational purposes. Before you release any app to production, you should consider field validation, error handling, and many other factors.

Feedback

Was this page helpful? Yes No

Provide product feedback

Build global support into canvas apps

Article • 12/16/2022

Power Apps is a global product. You can build and use canvas apps in many different languages and regions.

Both while building and running apps, the text displayed by Power Apps has been translated into different kinds of languages. You'll see menu items, dialog boxes, ribbon tabs, and other text in your native language. Typing in and displaying dates and numbers is also adapted for your particular language and region. For example, some regions of the world use a **.** (dot or period) as the decimal separator while others use a **,** (comma).

The apps you create can be globally aware as well. Use the **Language**, **Text**, **Value**, **DateValue**, and other functions to adapt what is displayed and used as input in different languages.

Language settings

When using the native studio or a native player, the language used is provided by the host operating system. For Windows, this setting can be controlled under "All Settings" and then "Time & language" settings. Windows also allows you to specify the characters to use for the decimal separator, overriding the language setting.

When using the web experiences, the language used is provided by the browser. Most browser default to the host operating system's setting with some also providing a way to set the language manually.

Authoring environment

The authoring environment adapts to the language setting of the author. The app itself is stored in a language agnostic manner, so that authors using different languages can edit the same app.

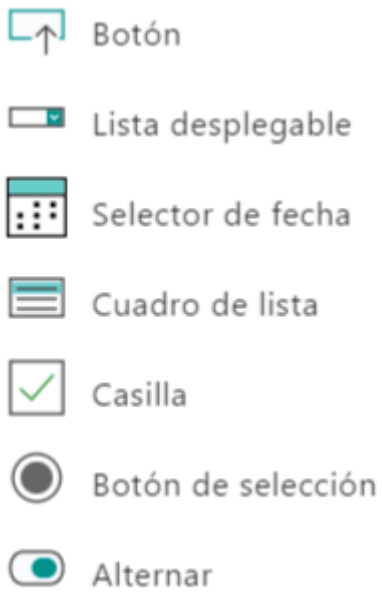
Names in formulas

Most elements in formula are always in English:

- Function names: **If**, **Navigate**, **Collect**, and so on.
- Control property names: **Screen.Fill**, **Button.OnSelect**, **Textbox.Font**, and so on.

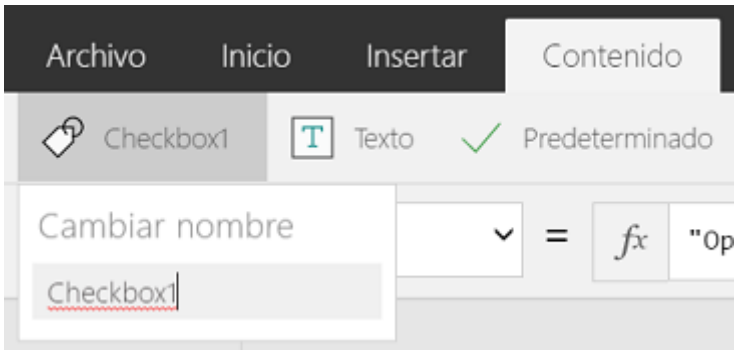
- Enumeration names: **Color.Aqua**, **DataSourceInfo.MaxValue**, **FontWeight.Bold**, and so on.
- Signal records: **Compass.Heading**, **Location. Latitude**, **App.ActiveScreen**, and so on.
- Operators: **Parent**, **in**, **exactIn**, and so on.

As the authoring experience is localized, control and other object names will appear in the native language of the author. In Spanish, some of the control names appear as:



When you insert one of these controls into your app, their name will default to English. This change is done for consistency with the control property names and the rest of the formula. For example, **Casilla** listed above is inserted as **Checkbox1**.

After a control is inserted, you can change the name to whatever you like. While selected, the far left-hand side of the "Content" ribbon displays the name of the control. Selecting this name drops down a text box where you can edit the name:



If you like, here you can rename the control to **Casilla1**. The red squiggly, in this case displayed by a browser, is because the name isn't a Spanish word and is of no concern.

You can use whatever names you like for:

- Control names
- Collection names
- Context variable names

Formula separators and chaining operator

Some separators and operators will shift based on the decimal separator of the author's language:

Expand table

Author's language decimal separator	Power Apps decimal separator	Power Apps list separator	Power Apps chaining operator
. (dot or period)	. (dot or period)	, (comma)	; (semi-colon)
, (comma)	, (comma)	; (semi-colon)	;; (double semi-colon)

The change in the Power Apps list separator is consistent with what happens to the Excel list separator. It impacts:

- Arguments in function calls.
- Fields in a record.
- Records in a table.

For example, consider the following formula expressed in a language and region that uses dot or period as the decimal separator, such as Japan or the United Kingdom:

```
If( Slider1.Value > 12.59,
      Notify( "Valid!", Success ); Navigate( "NextScreen", None ),
      Notify( "Invalid, try again", Error )
)
```

Now view this same formula in a language and region where a comma is used for the decimal separator, such as France or Spain:

```
If( Slider1.Value > 12,59;
      Notify( "Valid!"; Success );; Navigate( "NextScreen"; None );
      Notify( "Invalid, try again"; Error )
)
```

The highlight shows the operators that change between the two versions. The property selection operator . (dot or period) in **Slider1.Value** is always the same, no matter what the decimal separator is.

Internally the formula doesn't change, all that changes is how it's displayed and edited by the author. Two different authors using two different languages can view and edit the same formula, with each seeing the appropriate separators and operators for their language.

Creating a global app

The app you create can adapt to different languages, providing a great user experience for your users around the world.

Language function

The **Language** function returns the language tag of the current user. For example, this function returns **"en-GB"** for users in Great Britain and **"de-DE"** for users in Germany.

Among other things, you can use **Language** to display translated text for your users. Your app can include a table of translated values in your app:

TextID	LanguageTag	LocalizedText
"Hello"	"it"	"Ciao"
"Hello"	"fr"	"Salut"
"Hello"	*blank*	"Hello"
"Goodbye"	"it"	"Arrivederci"
"Goodbye"	"fr"	"Au revoir"
"Goodbye"	*blank*	"Goodbye"

And then use a formula such as the following to pull translated strings from the table:

Power Fx

```
LookUp( Table1, TextID = "Hello" && (LanguageTag = Left( Language(), 2 ) ||
IsBlank( LanguageTag ))).LocalizedText
```

Translated strings in other languages could be longer than they are in your language. In many cases, the labels and other elements that display the strings in your user interface will need to be wider to accommodate.

For more information, see the documentation for the **Language** function.

Formatting numbers, dates, and times

Numbers, dates, and times are written in different formats in different parts of the world. The meaning of commas, decimals, and the order of month, date, and year vary from location to location.

The Text function formats numbers and dates using the language setting of the user.

Text requires a format string to know how you want to format the number or date. This format string can take one of two forms:

- **A global aware enumeration.** For example, **Text(Now(), DateTimeFormat.LongDate)**. This formula will format the current date in a language appropriate format. This method is the preferred way to specify the format string.
- **A custom format string.** For example, **Text(Now(), "[$-en-US]dddd, mmmm dd, yyyy")** displays the same text as the enumeration when used in the language "en-US". The advantage of the custom format string is that you can specify exactly what you want.

The "[$-en-US]" on the front of the custom format string tells **Text** in which language to interpret the custom format string. This string is inserted for you and defaults to your authoring language. Normally you won't need to change this string. It's useful when authors from different languages are editing the same app.

The third argument to **Text** specifies which language to use for the result of the function. The default is the language setting of the current user.

For more information, see the documentation for the Text function.

Reading numbers, dates, and times

There are four functions for reading numbers, dates, and times provided by the user:

- Value: Converts a number in a text string to a number value.
- DateValue: Converts a date value in a text string to a date/time value. Anytime specified in the text string is ignored.
- TimeValue: Converts a time value in a text string to a date/time value. Any date specified in the text string is ignored.
- DateTimeValue: Converts a date and time value in a text string to a date/time value.

If you have used Excel, all of these functions are combined in the single **Value** function. They're broken out here since Power Apps has separate types for date/time values and numbers.

All of these functions have the same arguments:

- *String, required*: A string from the user. For example, a string types into a **Text input** control and read from the control with the **Text** property.
- *Language, optional*: The language in which to interpret the *String*. By default, the language setting of the user.

For example:

- **Value("12,345.678", "en-US")** or **Value("12,345.678")** when located where "en-US" is the user's language returns the number **12345.678**, ready for calculations.
- **DateValue("1/2/01", "es-ES")** or **DateValue("1/2/01")** when located where "es-ES" is the user's language returns the date/time value **February 1, 2001 at midnight**.
- **TimeValue("11:43:02", "fr-FR")** or **TimeValue("11:43:02")** when located where "fr-FR" is the user's language returns the date/time value **January 1, 1970 at 11:43:02**.
- **DateTimeValue("11:43:02 1/2/01", "de-DE")** or **DateTimeValue("11:43:02 1/2/01")** when located where "de-DE" is the user's language returns the date/time value **February 1, 2001 at 11:43:02**.

For more information, see the documentation for the **Value** and **DateValue, TimeValue, and DateTimeValue** functions and working with dates and times.

Calendar and Clock information

The **Calendar** and **Clock** functions provide calendar and clock information for the user's current language.

Among other things, use these functions to provide a **Dropdown** control with a list of choices.

For more information, see the documentation for the **Calendar** and **Clock** functions.

Feedback

Was this page helpful? Yes No

Provide product feedback

Build a multi-language app

Article • 03/03/2023

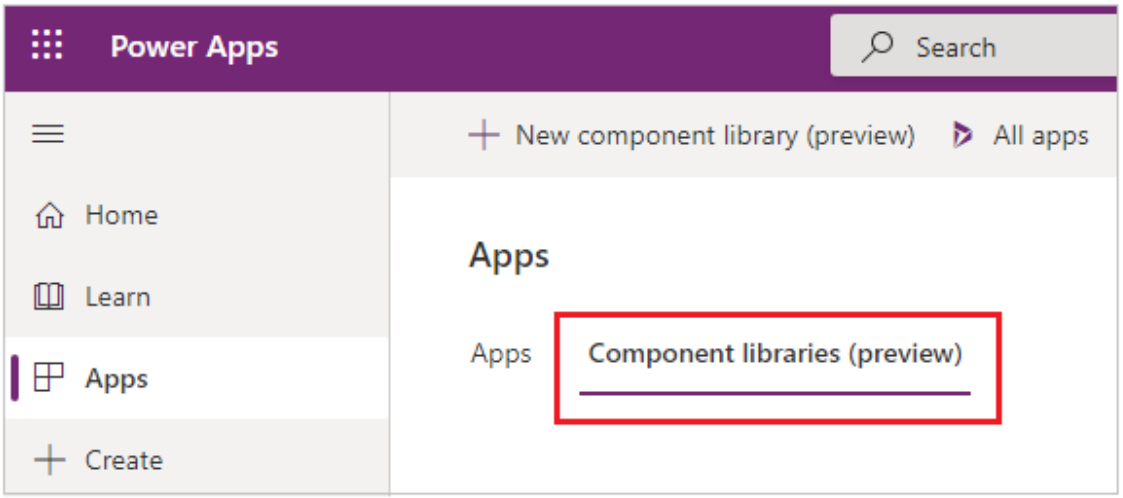
In this article, you'll learn how to create a multi-language app with localized experience. With this method, you can use canvas app components and make translations available in an app using a uniform formula syntax.

> **Important**
>
> The method described in this article use components for canvas apps which is in public preview. For more information, see Create a component for canvas apps

Create a translation component

When working with a multi-language app, the first step is to create a reusable component for translation. In this component, you'll store a table that acts as the dictionary of all translations you'll need. An output property will be used to provide output strings from this dictionary table based on the language of the user running the app that uses this component.

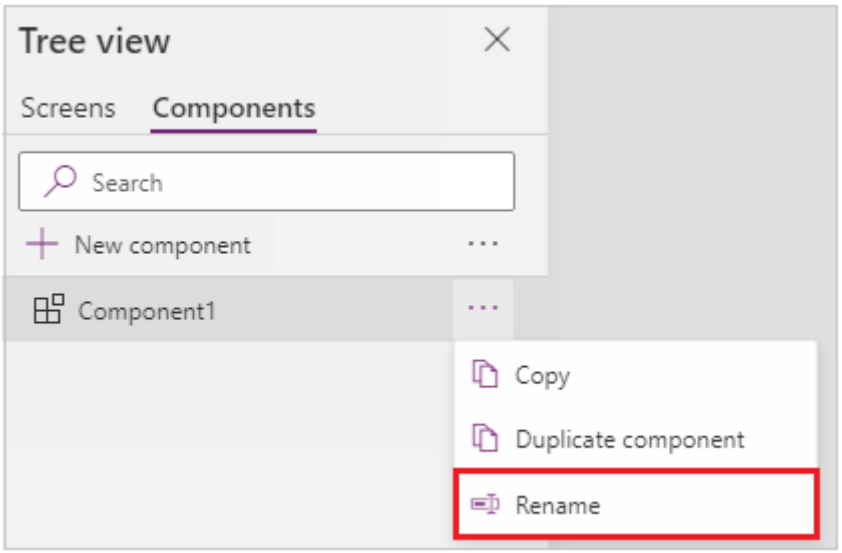
1. Sign in to Power Apps.
2. On the left pane, select **Apps**.
3. Under **Apps**, select **Component libraries (preview)**.



4. Select + **New component library (preview)**.

5. Enter a name, such as "Translation components", and select **Create** to open the component in Power Apps Studio.

6. Rename "Component1" by selecting **...** on the left pane, and then select **Rename** as "Translation component".



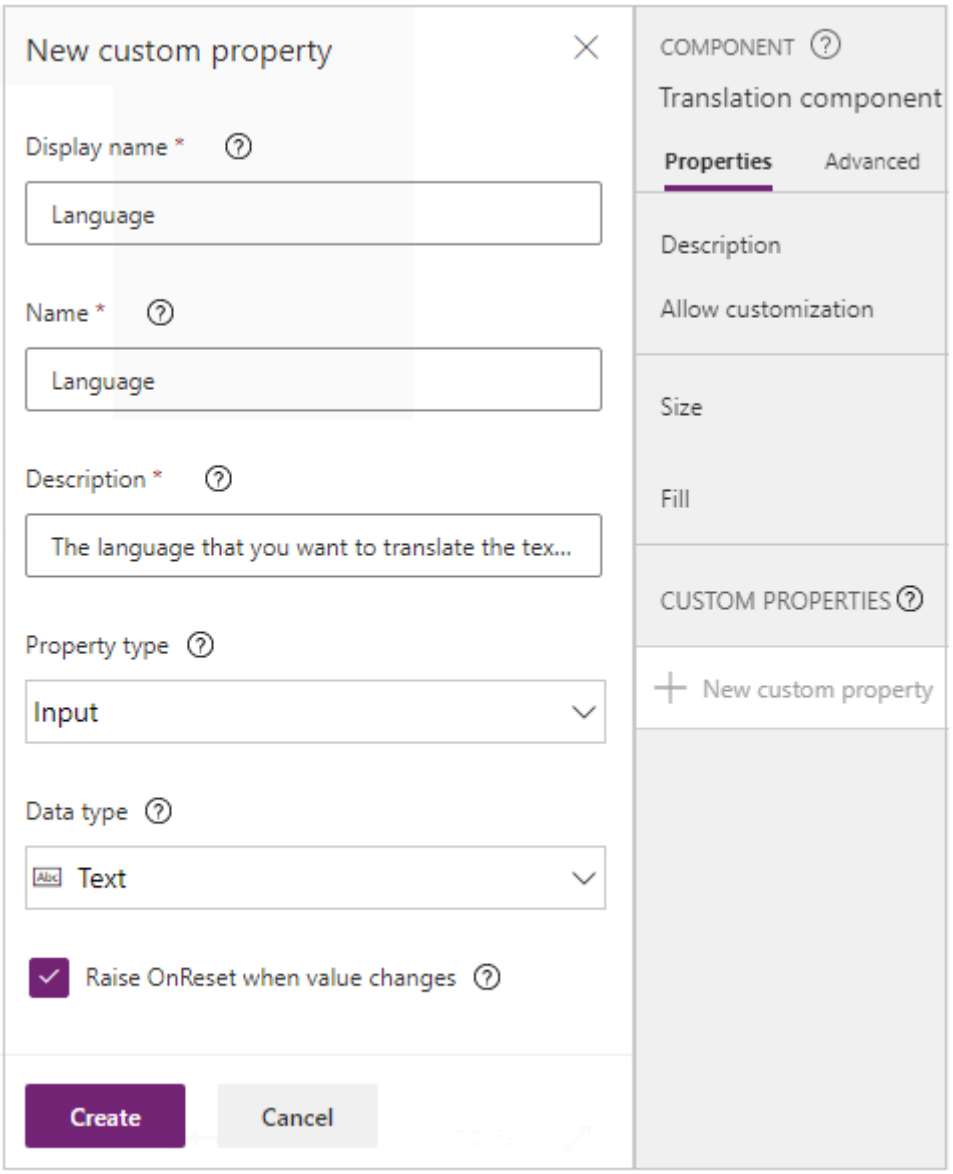
7. From the property pane on the right-side of the screen, select **+ New custom property**.

8. Set the following property values:

Expand table

Property name	Value
Display name	Language
Name	Language
Description	The language that you want to translate the text to.
Property type	Input
Data type	Text

9. Select **Raise OnReset when value changes** checkbox.



10. Select **Create**.

> **Note**
>
> This input property will receive the current locale ID (LCID) of the logged-in user.

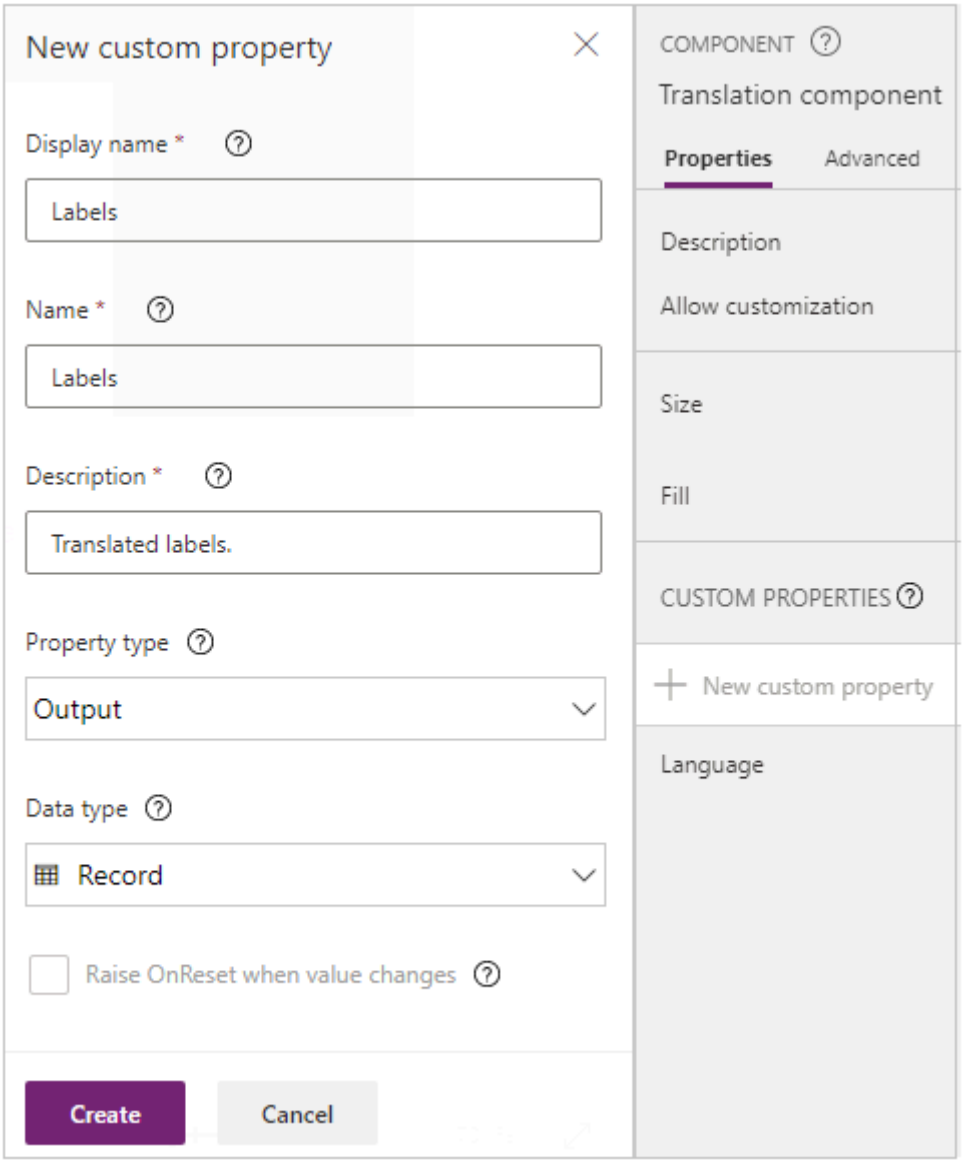
11. From the property pane on the right-side of the screen, select **+ New custom property**.

12. Set the following property values:

Expand table

Property name	Value
Display name	Labels

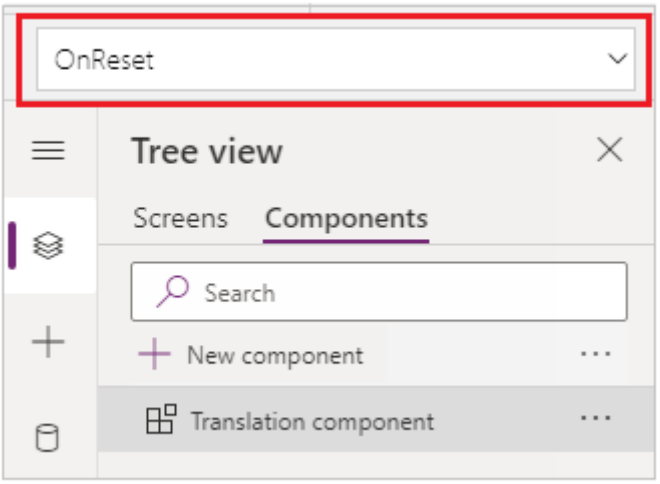
Property name	Value
Name	Labels
Description	Translated labels.
Property type	Output
Data type	Record



13. Select **Create**.

> **Note**
>
> This output property exposes the translated labels based on the input locale ID.

14. On the upper-left side of the screen, select the property list drop-down and choose **OnReset** property for the component.



15. Copy and paste the following formula in the formula bar for the **OnReset** property.

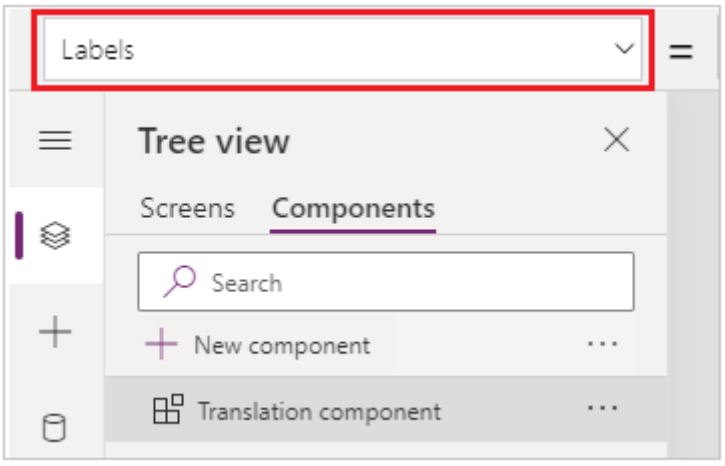
```
Power Fx

Set(
 varTranslations,
 Table(
        {
            Language: "en-us",
            Labels: {
                Title: "UI Tips for building canvas apps",
                JobTitle: "Power Platform Specialist"
            }
        },
        {
            Language: "pt-br",
            Labels: {
                Title: "Dicas de UI para construir canvas apps",
                JobTitle: "Especialista de Power Platform"
            }
        }
    )
   )
```

The formula uses Set() function to create a table with the labels in different languages as a dictionary. To customize this table as the dictionary for your translations, change the control names such as *Title*, *JobTitle*, *Platform*, *Close*, *Open*, *Cancel*, and so on. You can add, or remove translations for control names depending on your business requirement.

The table should also have an entry for each language that your app supports. Each entry will have a **Labels** property that will contain the translated content of all

possible buttons, inputs, and labels in your app.

16. On the upper-left side of the screen, select the property list drop-down and choose **Labels** property for the component.



17. Copy and paste the following formula in the formula bar for the **Labels** property.

```
Power Fx
```

```
LookUp(
 varTranslations,
 Language = Lower(
     Coalesce(
         Self.Language,
         Language()
     )
 )).Labels
```

The formula finds the right translation entry based on the input **Language** using the Lookup() function. In case where the language isn't set, the formula uses the current user's language as the filter through Coalesce function.

18. Select **File** -> **Save** to save the component library.

19. Select **Publish** to publish the component library.

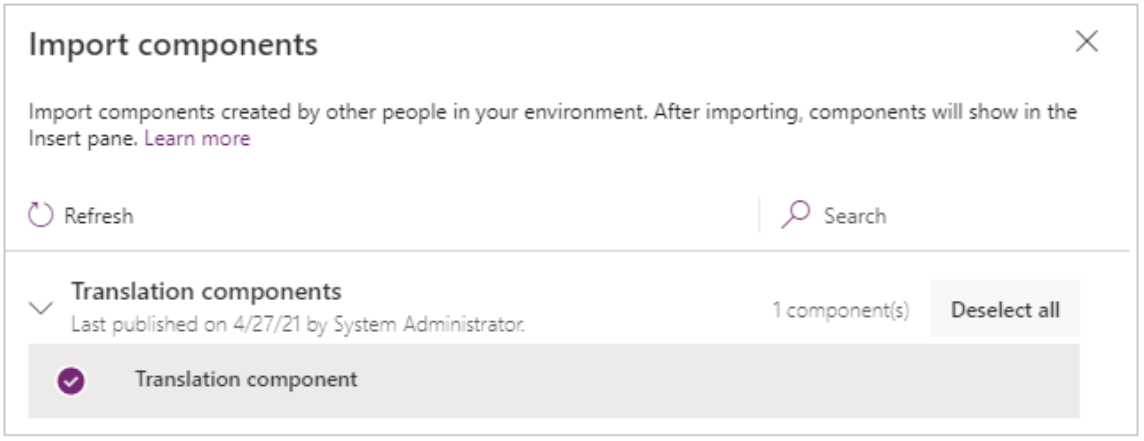
> **Tip**
>
> To learn more about creating, saving, and publishing component library, go to Create an example component library

Your translation component library is created, saved, and published for use.

Use the translation component in your app

Previously, you created the translation component library for reuse. In this section, you'll create an app that uses the translation component library and demonstrate the language translation based on the selected language.

1. Start by creating a blank canvas app with **Phone** layout.
2. On the left-side of the screen, select + (Insert).
3. On the bottom-left side of the screen, select **Get more components**.
4. Select **Translation component**.



> **Note**
>
> The name might be different if you saved the component library created earlier with a different name.

5. Select **Import**.
6. On the left-side of the screen, select + (Insert).
7. Under **Library components**, select **Translation component** to add the component to this app.
8. On the left-side of the screen, select the tree view.
9. Select the translation component.
10. On the upper-left side of the screen, select the property list drop-down and choose **Visible** property for the component.

11. Set the value of the **Visible** property to **false** to make the component invisible on the app.

12. Under **Input** in the list of controls, select **Toggle**.

13. On the upper-left side of the screen, select the property list drop-down and choose **OnChange** property for the component.

14. Update the formula for the **OnCheck** property of the toggle control to the following.

    ```
    Power Fx

        Set(varLanguage,"pt-br")
    ```

    In this formula, the toggle sets a variable called **varLanguage** with the value of "pt-br" using the function Set(), for Portuguese (BR) language code.

15. Update the formula for the **OnUncheck** property of the toggle control to the following.

    ```
    Power Fx

        Set(varLanguage,"en-us")
    ```

    In this formula, the toggle sets a variable called **varLanguage** with the value of "en-us" using the function Set(), for English (US) language code.

16. On the left-side of the screen, select the translation component.

17. On the upper-left side of the screen, select the property list drop-down and choose **Language** property for the component.

18. Set the formula value for **Language** property to `varLanguage`. The variable **varLanguage** is determined by the toggle configured earlier. When the toggle is checked, the language is set to "pt-br". When unchecked, the language is set to "en-us".

19. On the left-side of the screen, select + (Insert).

20. Select **Text label**.

21. Update the label control name to **Title** from the right-side of the screen using the property pane.

22. Select **Text label** again to add one more label.

23. Update the label control name to **JobTitle**.

24. Set the **Title**, and **JobTitle** labels below the toggle control so that both labels are visible.

25. Set the **Text** property for **Title**, and **JobTitle** labels from the upper-left side of the screen to the following.

Expand table

Label	Formula
Title	`'Translation component_1'.Labels.Title`
JobTitle	`'Translation component_1'.Labels.JobTitle`

> **Note**
>
> Replace **'Translation component_1'** in this formula to the name of the component in your app, if different.

Similarly, you can use different labels, and properties, as defined in the component that you created earlier to pass more property values. For example, in addition to *Title*, and *JobTitle*, you can create more properties such as *Description*, or *Instructions* in the component library with the translated text. And then, use such properties on the respective labels, as *'Translation component_1'.Labels.Description*, or *'Translation component_1'.Labels.Instructions*.

26. Select **File** -> **Save**, update the name for your app, and then select **Save** to save your app. More information: Save and publish an app

Test your app with language translation

Now your app is using the translation component. Go to Power Apps and select the app to run.

When the toggle is selected, the language is changed to Portuguese (BR) for the labels. When unchecked, the language is set back to English (US).



With this approach, you can now create your own component having the translations dictionary that fits your business requirement. And then, create more canvas apps that use the component that gives the business users the ability to use different languages.

See also

Add and configure controls
Understand variables
Formula reference

Feedback

Was this page helpful? Yes No

Provide product feedback

Create accessible canvas apps

Article • 12/16/2022

An accessible canvas app will allow users with vision, hearing, and other impairments to successfully use the app. In addition to being a requirement for many governments and organizations, following the below guidelines increases usability for all users, regardless of their abilities.

Use the **Accessibility Checker** to help review potential accessibility issues in your app.

Layout and color

Common sense and uncomplicated design helps apps be more accessible to all users. When doing heavy customization of apps take note of the below suggestions. Power Apps themes are designed to meet accessibility standards.

- Ensure all elements are clearly visible and text is of sufficient size. All content must be easily read and understood by the naked eye.
- Ensure input elements are labeled on the screen. **AccessibleLabel** property defines what the screen reader will announce.
- If customizing colors, ensure the contrast ratio of text to background is 4.5:1 or greater. Software tools that assist this process are readily available.
- Ensure layout follows a logical flow when read from top to bottom, left to right.
- Create a responsive app so that low-vision users can zoom in and use it without scrolling back and forth.

Keyboard

When testing your app's accessibility, ensure the app can be used by keyboard only, with or without a screen reader.

The **Tab** key should navigate to interactive elements in a logical order. You can create this with a logical app structure and by setting each control's **TabIndex** property accordingly.

- Label, Image, Icon, Shape controls: Set **TabIndex** to 0 if they are meant to be interactive. Otherwise, set **TabIndex** to -1.
- Do not set **TabIndex** greater than zero.
- Ensure that the **Simplified tab index** app setting is enabled.

Screen readers

The following screen readers have been verified to work with Power Apps:

- **JAWS**: Microsoft Edge
- **Narrator**: Microsoft Edge
- **NVDA**: Google Chrome, Firefox
- **TalkBack**: Google Chrome, Power Apps mobile
- **VoiceOver**: Power Apps mobile, Safari (macOS, iOS, iPadOS)

To ensure a satisfying experience with the screen reader it is recommended to:

- Ensure all input controls have the **AccessibleLabel** property set.
- For images, set **AccessibleLabel** to an appropriate description.
  - If a picture is not used as a button or a link (i.e. icon is there just for the decoration) and should not be read by the screen reader, make sure the **AccessibleLabel** is empty or not set.
  - If a picture or an icon is used as a button, then set **TabIndex** to 0 and **AccessibleLabel** to the link description.

Control type and structure

Using the right controls and grouping them will help screen reader users understand the structure of the app.

- Include at least one heading on each screen of the app. You can create headings by setting the **Role** property of a **Label**.
- Use a **Button** instead of a **Label** for interactive text.
- Group related content in **Containers**.
- Be aware of unsupported design patterns.

Multimedia

Ensure all videos are captioned and a transcript of all audio recordings is available to the user. **Video** control supports closed captions in WebVTT format via the **ClosedCaptionsUrl** property.

With the screen reader enabled, **Timer** does not announce button text, but how much time has passed. Announcements can't be turned off, even if timer is hidden with low opacity.

Working with signatures

If you have a signature field that uses the PenInput control, you need to enable an alternative method of signature input. The recommended way is to show a TextInput control where a user can type their name. Ensure the signing instructions are placed in the **AccessibleLabel** property and the control is placed close to the Pen input - to the right or immediately below.

Next steps

Accessible app structure

See also

- Accessible colors in Power Apps
- Show or hide content from assistive technologies for canvas apps
- Announce dynamic changes with live regions for canvas apps
- Use the Accessibility checker
- Accessibility limitations in canvas apps
- Accessibility properties

Organize controls in accessible canvas apps

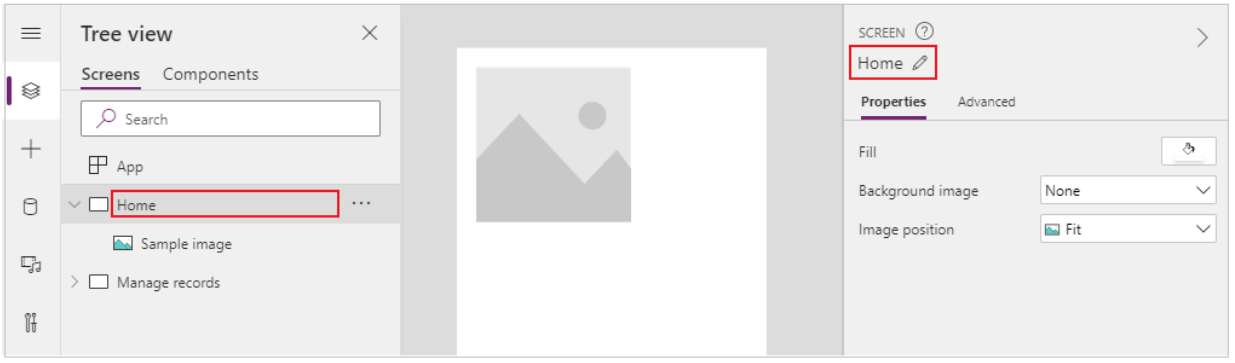
Article • 07/13/2024

Controls in an app should be organized to help screen reader users navigate sequentially. A logical order also reduces confusion for keyboard users and helps them be more productive.

Meaningful screen name

When a screen is loaded, screen readers will say its name. Pick a meaningful name to orientate users.

You can change the screen name in the controls tree or properties panel in Power Apps Studio. Select the screen, and then select to rename the screen.



The first element on a screen is its name. It's visually hidden and accessible only to screen reader users.

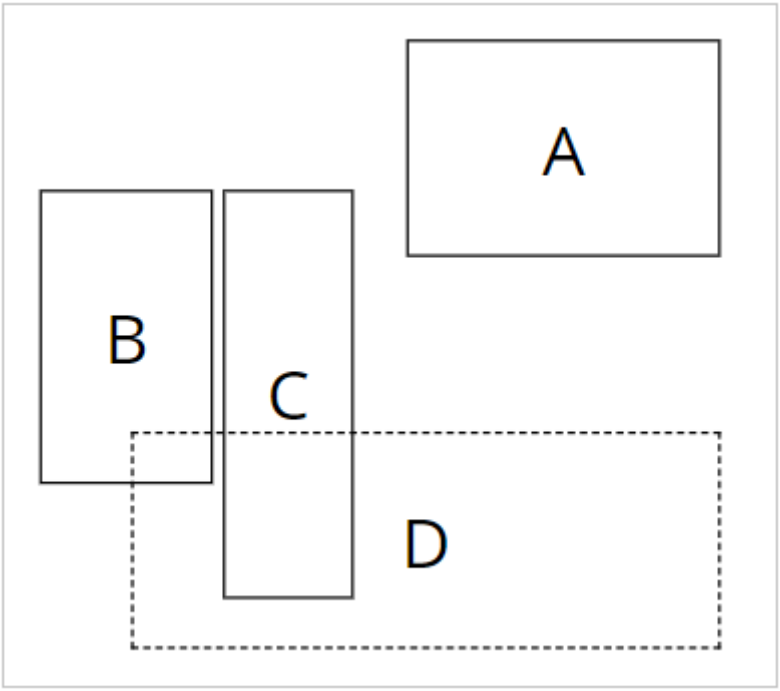
When a new screen loads, Power Apps focuses the screen name. If you use **SetFocus** immediately when the screen loads, the screen name won't be read. Consider creating a visible title and making it a live region to announce the change in context.

Logical control order

Screen reader users can navigate content sequentially. The order is determined by the position of controls, starting from top to bottom, then left to right. The size of the control doesn't matter, only its **X** and **Y** properties matter.

In this example, A appears first in the sequence since it's closest to the top. B and C have the same vertical position, but since B is closer to the left, and hence, it appears before

C. D appears last since it's furthest from the top.

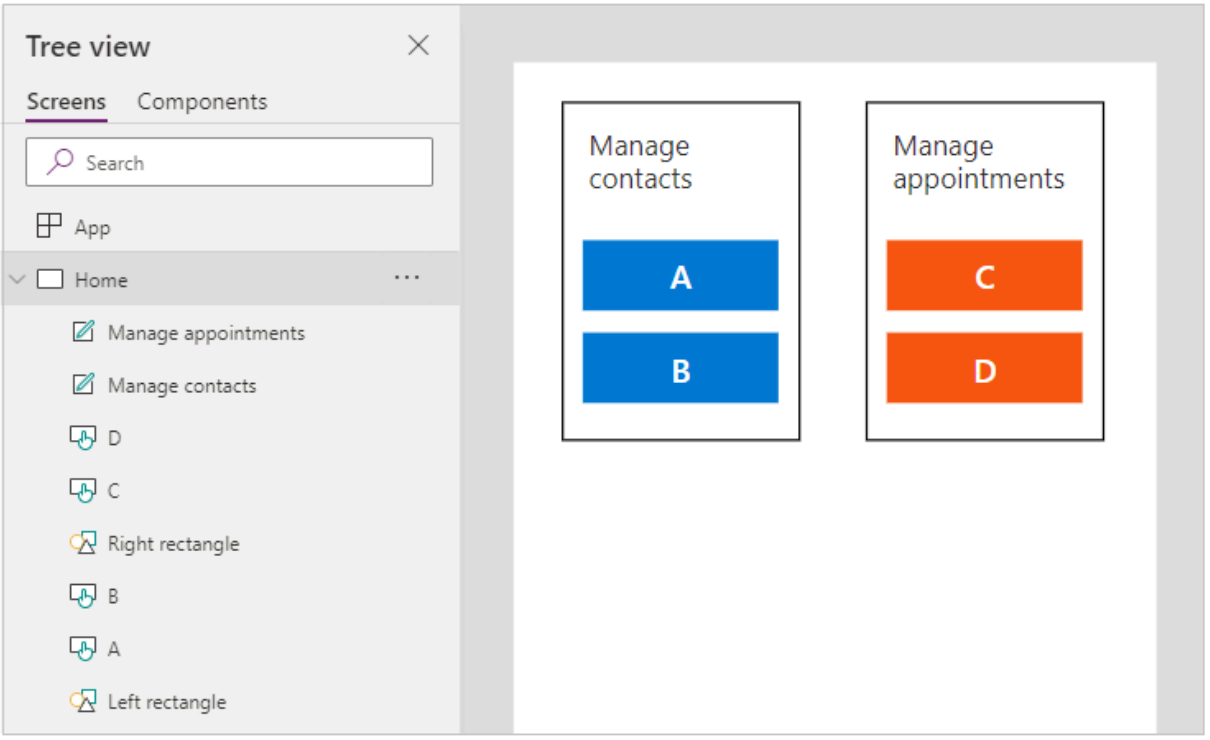


ⓘ Note

- In **Preview** mode when editing an app, the control order is not updated for performance reasons. The order will be correct when the app is published and run.
- Control order is not the same as that shown in the tree view of controls in Power Apps Studio. The tree view sorts controls according to when they were added to the app. It does not affect the order of controls when the app is run.
- When a control's **X** or **Y** value is set to an expression, the control order is not updated when the result of the expression changes. The order is computed and fixed when the app is saved, using the initial state of the app to evaluate expressions.
  - If you're changing its position because other controls are being hidden or shown, you can use auto-layout containers to manage **X** and **Y** for you.
  - You can also position all controls in a logical way regardless of expression values. For example, if control A should always be below control B and B may be hidden sometimes, set A's **Y** to be `If(B.Visible, B.Y + B.Height, B.Y + 1)`. The addition of 1 ensures that A is always below B, even if B is hidden.

Grouped controls

The default order is suitable for isolated content but not for grouped content. Consider two tiles side by side, drawn with **Rectangle** controls. Each tile has a heading. Below the heading are two buttons stacked vertically: A and B for the first tile and C and D for the other.



The default order goes from top to bottom, then left to right. Hence, the order of controls is:

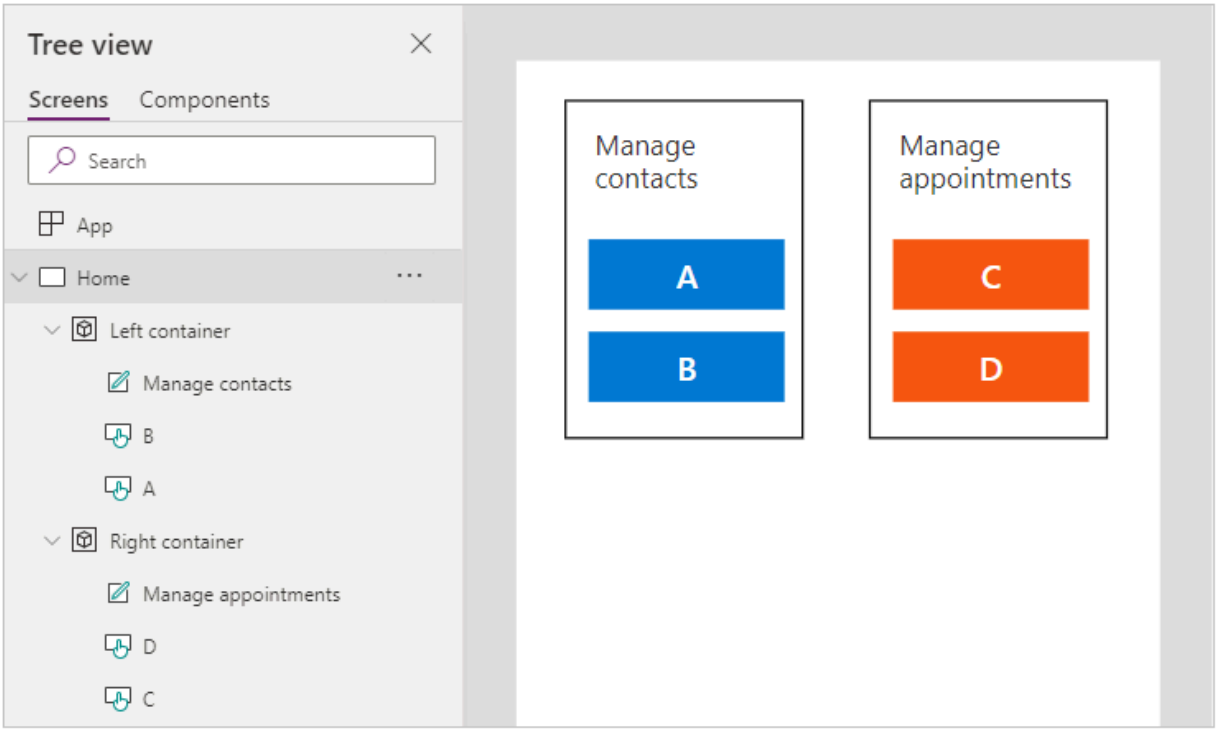
1. Left **Rectangle**
2. Right **Rectangle**
3. Left heading
4. Right heading
5. A
6. C
7. B
8. D

This structure doesn't convey that A and B are together, and similarly, C and D are together.

Use **Containers** to group related content. All controls in a **Container** will appear together in sequence. Within a container, controls are ordered with the same rule: top to bottom, then left to right.

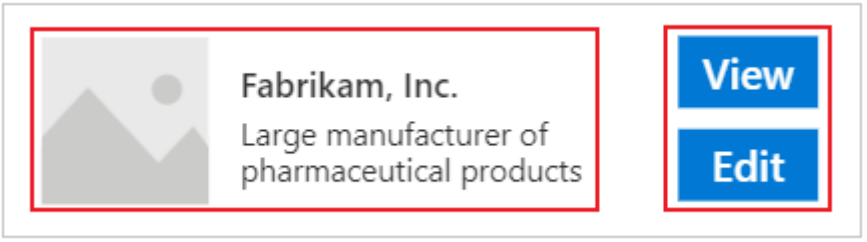
Replacing the **Rectangles** of the previous example with **Containers**, the control order is now logical for screen reader users:

1. Left **Container**
2. Left heading
3. A
4. B
5. Right **Container**
6. Right heading
7. C
8. D



All controls in a **Form Card** and **Gallery** are automatically grouped, so you don't have to use a **Container**. However, if there are subgroups, you should still use **Containers** for them.

In this example, a **Gallery** row has a thumbnail and two pieces of text on the left. On the right are two buttons. Visually and logically, the two sets of controls should be grouped. This ensures that screen reader users will encounter the left group first before the right.



Logical keyboard navigation order

Keyboard navigation is an important aspect of any app. For many, the keyboard is more efficient than using touch or a mouse. The navigation order should:

- Follow the visual flow of controls.
- Follow either an intuitive across and then down "Z" order or a down and then across "reverse-N" order.
- Only have a tab stop at controls that are interactive.

AcceptsFocus specifies whether controls can be reached by keyboard. For classic controls, the equivalent property is **TabIndex**.

Navigation order follows control order: from left-to-right, then top-to-bottom, in a "Z" pattern. You can customize it the same way as with control order. For example, controls in **Containers**, **Form Cards** and **Galleries** are automatically grouped. The Tab key will navigate through all elements inside the container before proceeding to the next control outside of the container.

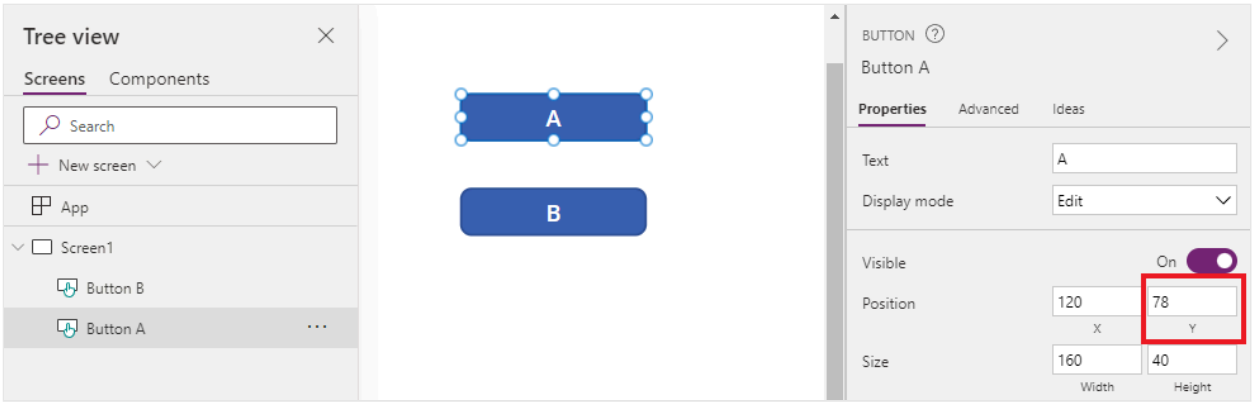
If the navigation order is unexpected, you should first check if the app structure is logical.

> ⓘ **Note**
>
> When controls are dynamically moved on the screen, for example, by having its **X** or **Y** value change according to a Power Fx expression, the navigation order will not be updated.

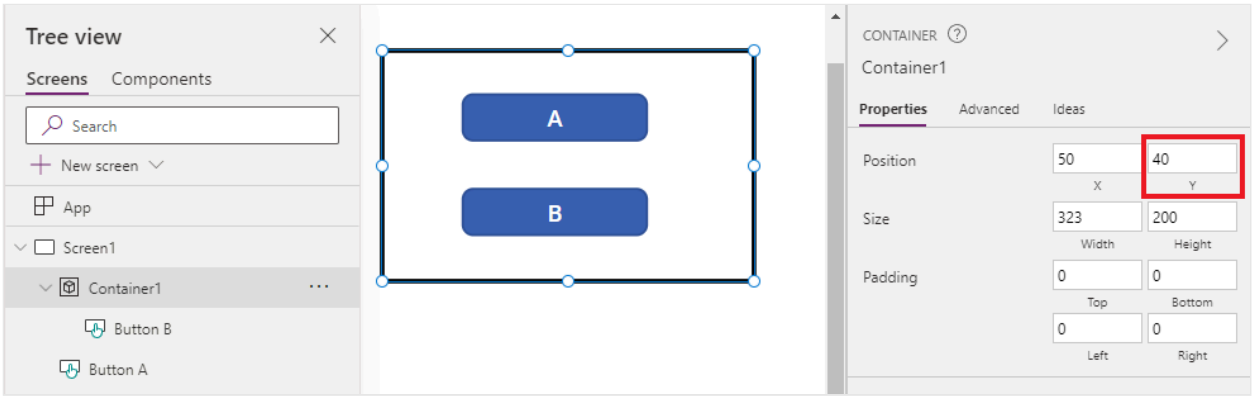
Workaround for custom tab sequence

For rare scenarios where the keyboard navigation order should be different from visual order, you can position container controls carefully to have the same effect.

In the example below, button A is above button B. The natural tab navigation order is A, then B.



To reverse the tab navigation order, put B in a Container control. Set the Y value of the Container to be above A. The app structure now has the Container (and B) before A. Hence, the tab navigation order is B, then A.



With this technique, screen reader users will also encounter B before A when navigating without the **Tab** key.

Custom tab indexes (retired feature)

Custom tab indexes are those that are greater than zero. They are no longer supported. All **TabIndex** values greater than zero will be treated as zero.

Custom tab indexes are almost always a sign of bad design. There are better alternatives like creating an appropriate app structure or using **SetFocus** to change focus.

Some problems with custom tab indexes:

Accessibility

It's a serious accessibility issue to have custom tab indexes. Screen reader users navigate an app using its logical structure. Custom tab indexes ignore that structure. Since screen reader users can also navigate using **Tab** key, they'll be confused when they get a different order from other methods of navigation.

Usability

Users can be confused when some items appear to be skipped. They can be disoriented when focus moves in an unpredictable order. This is even more problematic for users with cognitive disabilities.

Maintenance

App makers have to manually update the **TabIndex** of multiple controls whenever a new one is inserted. It's easy to miss an update or get the order wrong.

Performance

To support custom tab indexes, the Power Apps system has to examine all controls on the page and compute the appropriate order. This computation is an intensive process. Container controls like **Gallery** have complicated rules on how **TabIndex** works for child controls. The system maps the app maker's desired **TabIndex** to a different value to obey these rules. This is why even if TabIndex is set to zero for all controls, the actual HTML `tabindex` will be some positive number.

Integration with other components

Custom tab indexes only work with built-in controls. Controls that aren't integrated into Power Apps' tab index system will have an unexpected navigation order. This can be a problem for code components. Developers of these components have to keep track of interactive elements and set tab index on them. They might use third-party libraries, which may not even provide a way to customize tab indexes. On the other hand, when all tab indexes are either 0 or -1, there's no need to involve Power Apps' tab index system. Any third-party component embedded in the app will automatically get the correct tab sequence.

In the other direction, when canvas apps are embedded in another web page, custom tab indexes don't work. For example, in custom pages. Power Apps can't control the elements outside the canvas app, so the overall tab navigation order will be illogical.

Next steps

Accessible colors in Power Apps

See also

- Create accessible apps
- Show or hide content from assistive technologies for canvas apps
- Announce dynamic changes with live regions for canvas apps
- Use the Accessibility checker
- Accessibility limitations in canvas apps
- Accessibility properties

Feedback

Was this page helpful? Yes No

Provide product feedback

Improved canvas keyboard navigation in canvas apps (experimental)

Article • 11/13/2024

[This article is pre-release documentation and is subject to change.]

Important

- This is an experimental feature, and will be deprecated soon. Instead of using this experimental feature, consider structuring an app appropriately to achieve the desired tab navigation order.
- Experimental features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

When you're nesting controls such as containers and component instances, the user input value for TabIndex isn't respected sometimes. **Improved canvas keyboard navigation** experimental setting improves the navigation experience in this situation when **Tab** key is pressed on the keyboard helping focus on the desired container or control precisely and predictably.

This feature also adds support for handling accessibility for keyboard tabs inline with rest of the controls in canvas apps. And addresses manual TabIndex assignments for all controls.

Note

This feature is added as "Experimental" to maintain backward compatibility and maintain existing app functionality.

When this feature is turned on, it also enables the following boolean properties for all child controls for the selected container or component instance in canvas apps:

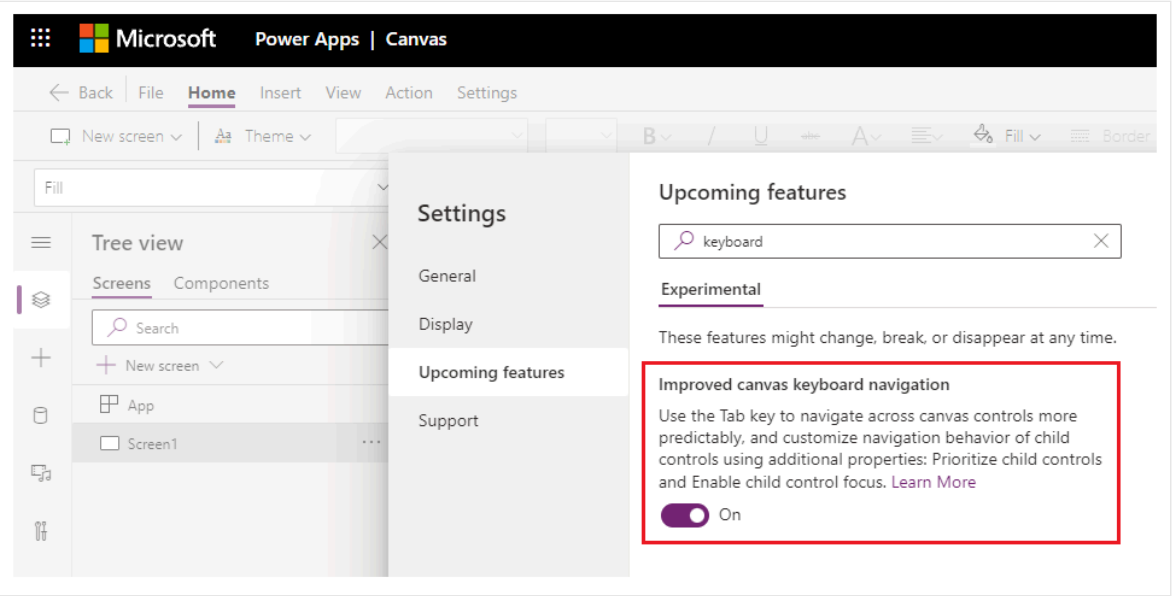
Expand table

Property Name	Description
Prioritize child controls	Determines the order of navigation (**Z-order**) for child controls on canvas when pressing tab key on the keyboard. • **On** (Default): Pressing the tab key on the keyboard will first progress through all child controls before moving the focus outside of the selected container or component instance. This option is recommended for similarly nested HTML elements. • **Off**: Pressing the tab key on the keyboard will progress through all controls only based on Z-order, ignoring parent-child relationship between controls or containers for keyboard navigation. **Note**: This property isn't applicable to responsive or autolayout containers.
Enable child control focus	Determines the value of TabIndex for child controls on canvas when pressing tab key on the keyboard. • **On** (Default): Pressing the tab key behaves as per TabIndex values defined on each control. • **Off**: Pressing the tab key doesn't move focus to any child control within the selected container or component instance. Sets the TabIndex value to **-1** for all child controls.

Configure improved canvas keyboard navigation in your app

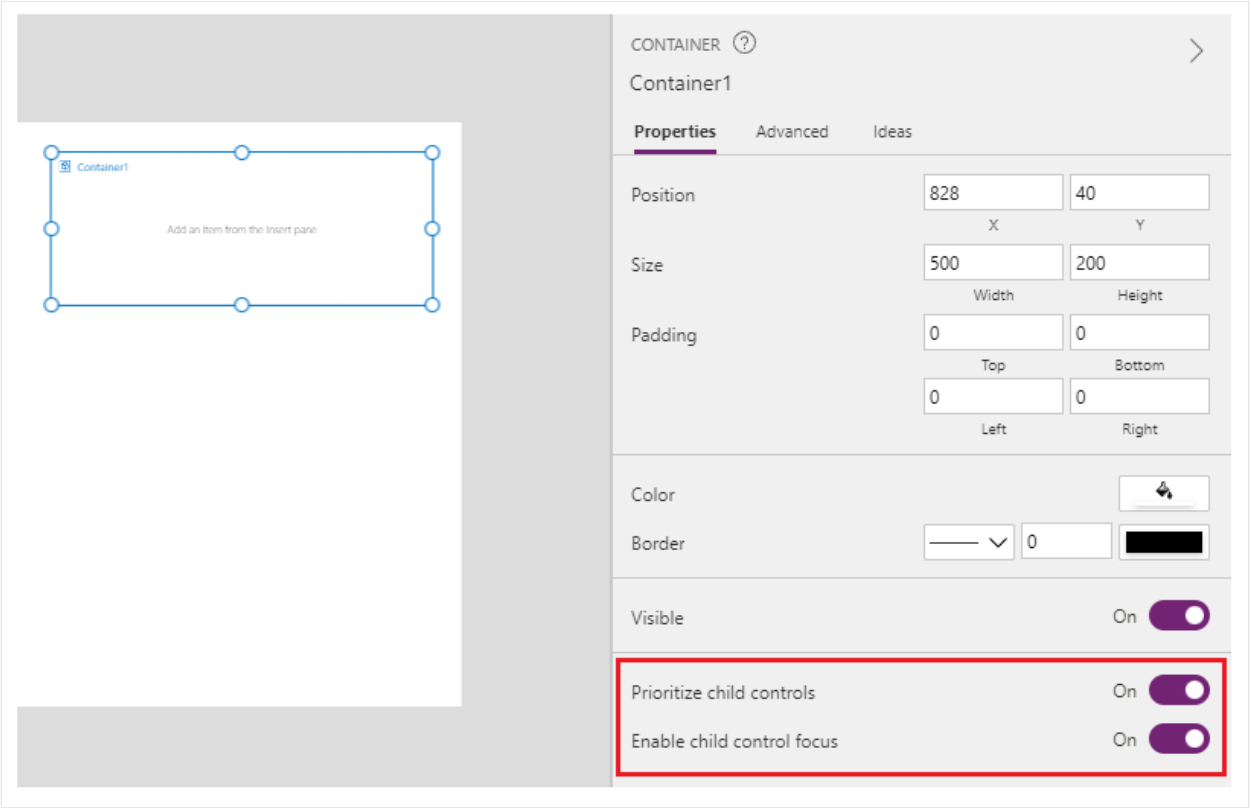
If not already enabled, follow these steps to enable this feature in your app, and set the properties mentioned earlier.

1. Sign in to Power Apps.
2. Select **Apps** from the left-pane.
3. Select your app, and then select **Edit**.
4. In Power Apps Studio, select **Settings** > **Upcoming features** > **Experimental**.
5. Select **Improved canvas keyboard navigation** to turn on the feature.



6. Close settings.

Now that you've enabled the experimental feature, you'll see two new properties for controls and containers—**Prioritize child controls**, and **Enable child control focus**. Change the property values as appropriate.



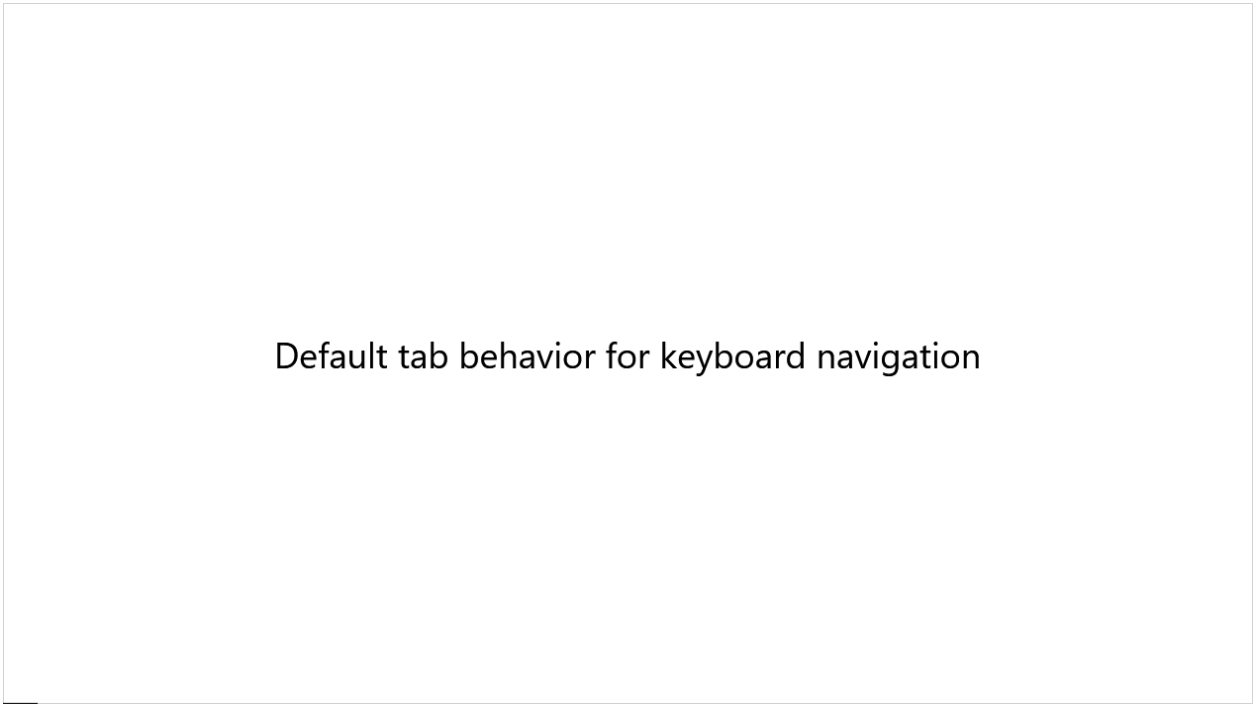
Examples

Now that you understand the new feature with improved canvas keyboard navigation, let's take a look at a few examples to understand the behavior when the tab key is pressed.

Default improved canvas keyboard navigation behavior

The following example shows multiple Text input controls, and several nesting scenarios. The number displayed in the input represents the value of the TabIndex property. There are two nested containers, and component controls overlaid on top of each other.

The default order is determined by the relative position of the controls. When focus enters a container or a component, tabs first traverse the children of the container before moving on to the next available control.



When Prioritize child controls is set to Off

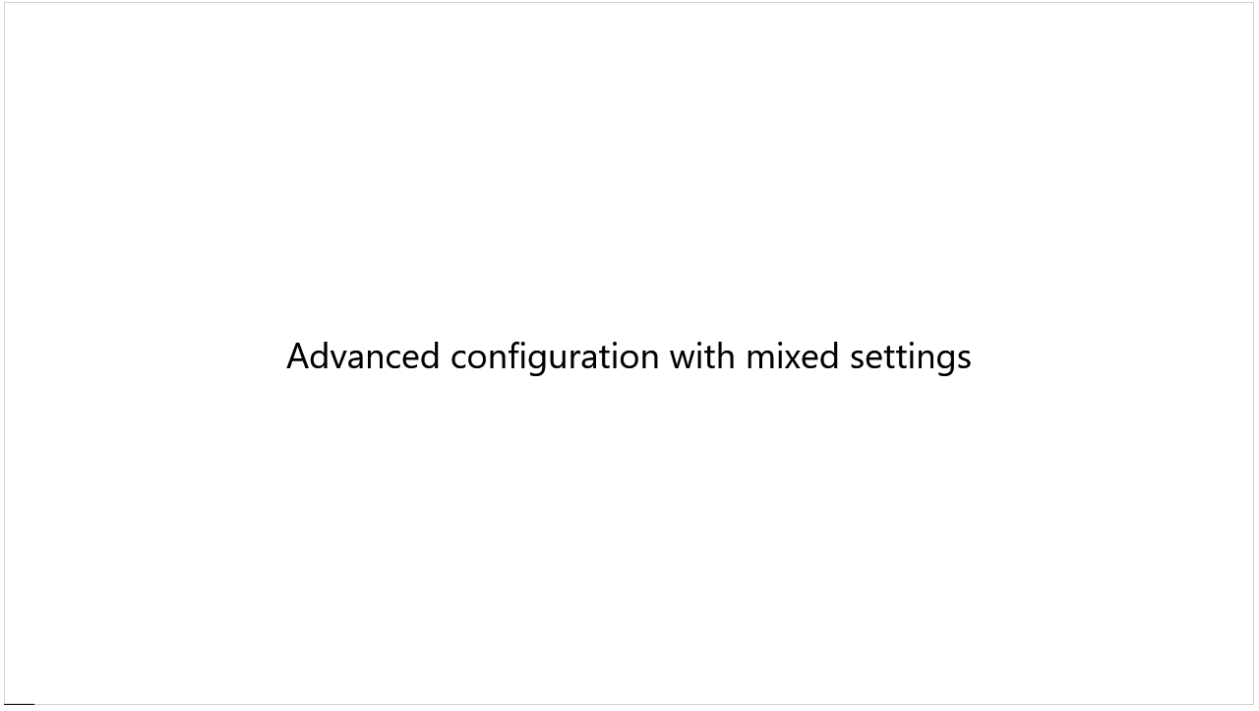
In the following example, each container and component control has the **Prioritize child controls** property set to "Off". All inputs are therefore considered to be on the same level of nesting, so the order is determined purely by their X/Y position relative to the screen.



Advanced configuration with mixed settings

In the following example, the orange outlined containers have **Prioritize child controls** property set to "Off". All others controls have this property set to "On". Also, a custom TabIndex property has been set for some inputs, indicated by the number shown in each input.

Tab order first proceeds through the containers and controls with TabIndex value of greater than 0, and then proceeds through all others with value of 0. This behavior was also the in the earlier implementation.



See also

- Create accessible apps
- Accessible app structure
- Accessible colors in Power Apps
- Use the Accessibility checker
- Accessibility limitations in canvas apps
- Accessibility properties
- Improved canvas keyboard navigation in canvas apps (video)

Feedback

Was this page helpful? Yes No

Provide product feedback

Accessible colors for canvas apps

Article • 12/16/2022

Colors used in a canvas app should be accessible to color-blind and low-vision users. All Power Apps themes are accessible by default. When modifying colors used in an app, follow these guidelines to ensure that they remain accessible. There are several tools available online which can help you identify color contrast issues.

Minimum contrast for text

- Text and its background must have a contrast ratio of at least 4.5:1
- Large text must have a contrast ratio of at least 3:1
- Disabled text has no contrast requirements

In practical terms, all interactive controls must have adequate color contrast between:

- **Color** and **Fill**
- **PressedColor** and **PressedFill**
- **HoverColor** and **HoverFill**

Minimum contrast for non-text

- Non-text components, such as icons and borders, must have a contrast ratio of at least 3:1 with the colors outside of them.
- Disabled and decorative components have no contrast requirements.

User interface components

All interactive controls must have adequate color contrast between:

- **FocusedBorderColor** and color outside the control

Additional color contrast checks apply for controls where the entire area is interactive or informative. For example, a **Button** or an **Icon** used as a button. This ensures that the boundaries of the control are clear and users know where they can click or tap.

If there is a border for such controls, there should be adequate color contrast between:

- **BorderColor** and color outside the control
- **PressedBorderColor** and color outside the control
- **HoverBorderColor** and color outside the control

If there is no border, there should be adequate color contrast between:

- **Fill** and color outside the control
- **PressedFill** and color outside the control
- **HoverFill** and color outside the control

Graphical objects

If an image conveys important information, consider checking it for contrast issues. This applies to controls where an image can be shown: **Audio**, **Image**, **Microphone**, and **Video**.

For video content, consider checking it for contrast issues. Alternatively or additionally, provide closed captions that describe the video.

Provide other visual cues

Ensure that the app does not convey information with just color. For example, users with red-green color blindness will not be able to distinguish a red error message from a green success message.

Additional cues like an **Icon** or text styles like **Italic** and **Underline** can help convey meaning.

Next steps

Show or hide content from assistive technologies for canvas apps

See also

- Create accessible apps
- Accessible app structure
- Announce dynamic changes with live regions for canvas apps
- Use the Accessibility checker
- Accessibility limitations in canvas apps
- Accessibility properties

Show or hide content from assistive technologies for canvas apps

Article • 12/16/2022

In most cases, all users should be able to access all content, but you might occasionally want to show content only to sighted users or only to screen-reader users. For example, you might want only screen-reader users to access descriptions of chart trends that are obvious to sighted users. You might also want to hide an icon from screen-reader users if an adjacent label describes it. Another description on the icon would be unnecessarily verbose.

Hide content from all users

- Set **Visible** to false.

Hide content from sighted users and show it to screen-reader users

Use one or more techniques listed below:

- Set **Size** to 0.
- Set **Width** and **Height** to 1.
- Set **X**, **Y**, or both properties such that the control is outside the screen.
- Set **Color** and related properties to transparent.
- Position a rectangle **Shape** above the content, and set **Fill** to the same color as the background color of the screen.

> ⓘ **Note**
>
> Users can still use a keyboard to access an interactive control, such as a **Button**, even if you hide it by using one of the techniques in the previous list. Set **TabIndex** to -1 if you want to prevent users from accessing the control by pressing the Tab key.

Hide content from screen-reader users and show it to sighted users

- For **Image**, **Icon**, and **Shape** controls, set **AccessibleLabel** to the empty string "".

Next steps

Announce dynamic changes with live regions for canvas apps

See also

- Create accessible apps
- Accessible app structure
- Accessible colors in Power Apps
- Use the Accessibility checker
- Accessibility limitations in canvas apps
- Accessibility properties

Announce dynamic changes with live regions for canvas apps

Article • 12/16/2022

Dynamic changes pose challenges to the visually-impaired. Users who access an app through a screen reader are focused on one part of the app. If a change happens elsewhere, those users won't be aware of it.

You can solve this problem by adding live regions, which screen readers track. If content changes in a live region, a screen reader will announce that change.

The underlying mechanism for live regions are aria-live regions, so the same guidelines apply.

Example uses of live regions

You can use live regions to notify users when events such as these occur:

- A validation error occurs in a form.
- An action triggered by a button is successful. For example, a user might select a button to add an item to a collection, and a live region could show the message "Item added".
- The user selected a different tab.
- A background timer refreshes a news feed.

Create and configure a live region

You can configure only a **Label** control as a live region. Its **Live** property determines what type of live region it is.

- **Off**: Not a live region. Screen readers don't announce changes.
- **Polite**: Screen readers announce changes after finishing speaking. Use this value for non-critical notifications that don't require immediate attention.
- **Assertive**: Screen readers interrupt themselves to announce changes immediately. Use this for critical notifications that require immediate attention.

If the text content of a live region changes, screen readers will announce the entire text content, not just the changed portion. If the value of the **Text** property is set to the empty string "", the screen reader doesn't announce anything.

To repeat a message, clear text contents by setting the value of the **Text** property to the empty string "" and then set the value to the message again.

Best practices

- Always set **Visible** to true. Some screen readers don't detect live regions that disappear and reappear.
- Avoid changing the value of **Live**. Some screen readers don't detect when a non-live region becomes live and vice-versa.
- Position the live region in a logical position in the app, even if it isn't visible. Ensure that its contents are sensible in context with the elements before and after it. Users can access a live region anytime through regular navigation with a screen reader, not just when changes happen.

Next steps

Use the Accessibility checker

See also

- Create accessible apps
- Accessible app structure
- Accessible colors in Power Apps
- Show or hide content from assistive technologies for canvas apps
- Accessibility limitations in canvas apps
- Accessibility properties

Review a canvas app for accessibility in Power Apps

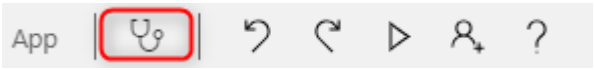
Article • 12/16/2022

Users who have vision, hearing, or other impairments can use your canvas app more easily and successfully if you consider accessibility as you design how the app looks and behaves. If you're not sure how to make your app more accessible, you can run the Accessibility checker in Power Apps Studio. This tool not only finds potential accessibility issues but also explains why each might be a potential problem for users who have a specific disability and offers suggestions on how to resolve each issue. The Accessibility checker detects screen-reader and keyboard issues for you, and you can find information about how to fix color-contrast issues by using accessible colors.

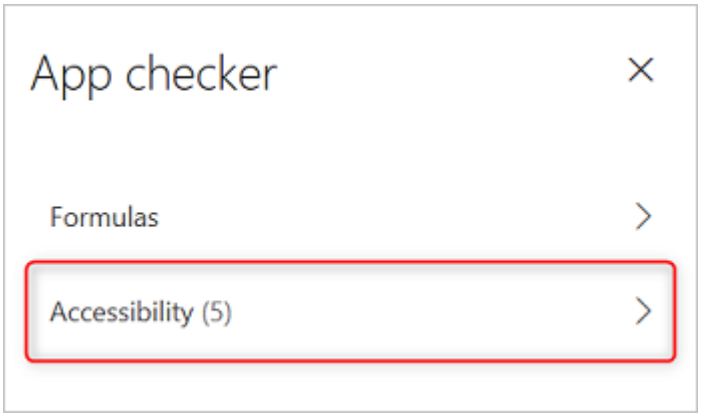
The Accessibility checker helps you identify settings that you might want to change, but you should always consider the suggestions in the context of what your app needs to do. Many suggestions may be worthwhile, but you can ignore any that might do more harm than good.

Find accessibility issues

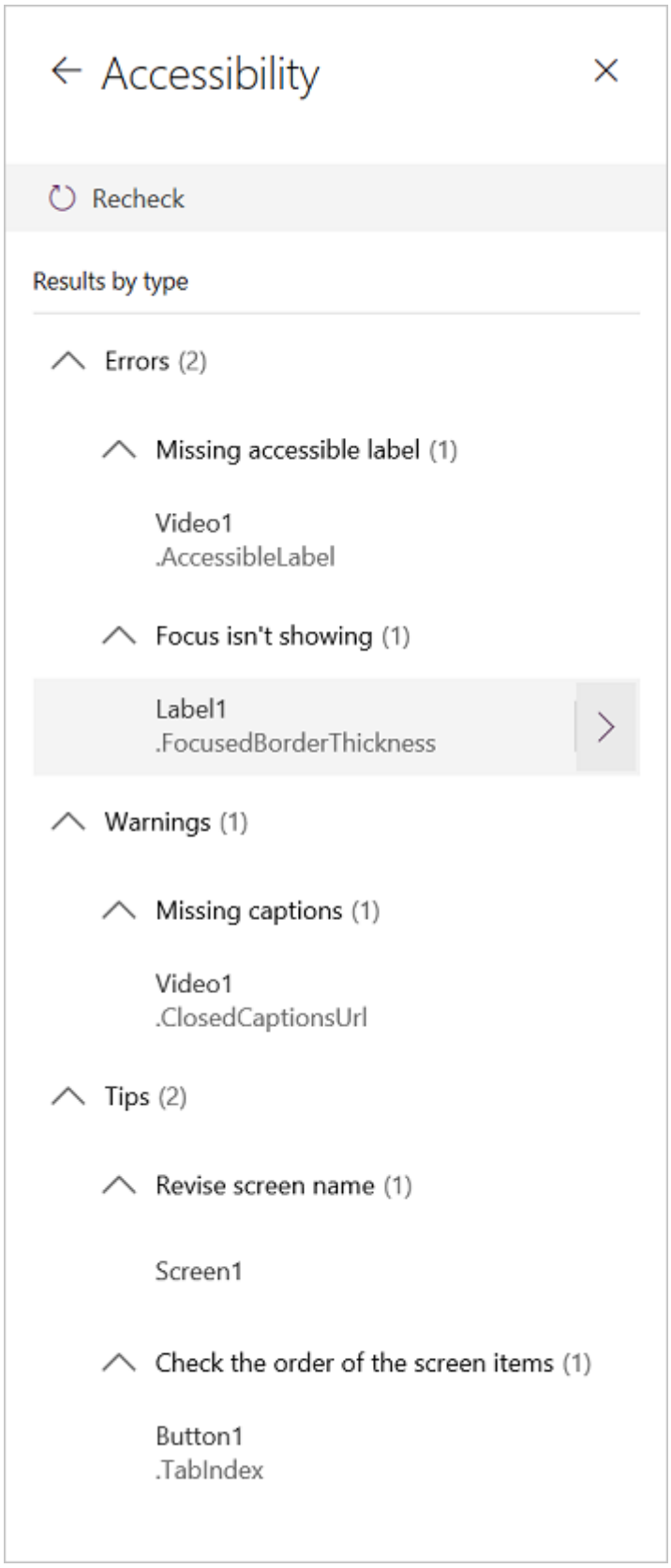
1. In the upper-right corner of Power Apps Studio, select the icon for the App checker.



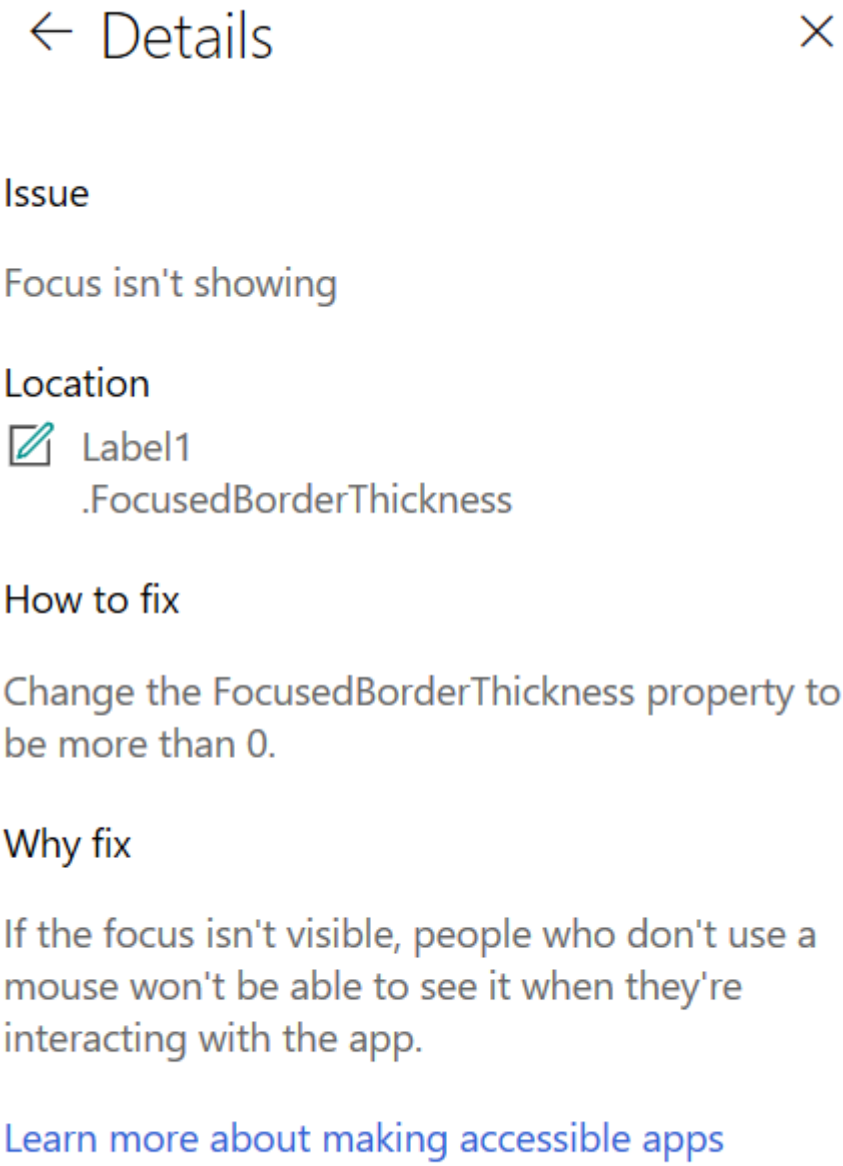
2. In the menu that appears, select **Accessibility**.



A list of issues appears, sorted first by severity and then by screen.



3. Select the arrow next to an item to show details about it.



4. Select the back arrow to return to the list of items.
5. If you decide to address an issue, select it to open the affected property.
6. After you change one or more properties, select **Re-check** to update the list of issues.

   Resolved items disappear from the list, and new items may appear.

Severity of issues

The Accessibility checker classifies each issue as an error, a warning, or a tip based the issue's severity.

- **Errors** identify issues that make the app difficult or impossible to use and understand for users who have a disability.
- **Warnings** identify issues that make the app difficult to use or understand for most but not all users who have a disability.
- **Tips** help you improve the experience of users who have a disability.

Types of issues

Issue Title	Severity	Issue Description	How to fix	Why fix
Missing accessible label	Error	When the accessible-label property of an interactive control contains no text. An interactive control can be inherently interactive, as a button is, or it has interactive properties. For example, you might have set the **OnSelect** property of an image or set its **TabIndex** property to 0 or higher.	Edit the accessible-label property to describe the item.	If the accessible-label property contains no text, people who can't see the screen won't understand what's in images and controls.
Focus isn't showing	Error	When the **FocusBorderThickness** of a control is set to 0. It is good practice to ensure a proper color-contrast ratio between the focus border and the control itself so it's clearly visible.	Change the **FocusedBorderThickness** property to a value that's higher than 0.	If the focus isn't visible, people who don't use a mouse can't see it when they interact with the app.
Missing captions	Warning	When the **ClosedCaptionsURL** property of an **Audio** or **Video** control is empty.	Set the **ClosedCaptionsURL** property to the URL for captions.	Without captions, people who have disabilities might not get any information from a video or audio segment.

Issue Title	Severity	Issue Description	How to fix	Why fix
Missing helpful control settings	Warning	When any of several settings (such as showing labels and markers for charts and showing default controls for **Audio**, **Video**, and **Pen input** controls) are turned off.	Select the warning, and then set the property to **true**.	By changing this property setting, you give the user better information about how the controls in your app function.
HTML won't be accessible	Warning	When a control other than an HTML text control contains HTML. In that case, Power Apps doesn't support accessibility of custom HTML elements.	Use a method other than HTML, or remove the HTML from this element.	Your app won't work correctly or be accessible if you add interactive HTML elements.
Turn off autostart	Warning	When an **Audio** or **Video** control's **Autostart** property is set to **true**.	Set the control's **Autostart** property to **false**.	Video and audio files that play automatically can distract users. Let them choose whether to play a clip.
Revise screen name	Tip	When a screen has a default name, which will be read out by screen readers when users navigate the app.	Give the screen a name that describes what's on the screen or what it's used for.	People who are blind, have low vision, or a reading disability rely on screen names to navigate using the screen reader.
Add State indication text	Tip	When a control that has a state, such as a toggle, but the value labels are turned off.	Set the **ShowValue** property of the control to **true** to show its current state.	Users won't get confirmation of their actions if the state of the control doesn't appear.

Issue Title	Severity	Issue Description	How to fix	Why fix
Check the order of the screen items	Tip	When the **TabIndex** property is greater than 0. App creators can set custom tab orders by setting the **TabIndex** property to a value greater than 0 but it is highly discouraged as it is hard to get right, maintain, and can break screen readers.	Set all **TabIndex** properties to 0 or -1 whenever possible. Instead of using **TabIndex**, use the **Enhanced group** control to change the navigation order from the default. If values of **TabIndex** greater than 0 must be used, make sure that your screen elements match the order in which you'd want to tab through them.	The navigation order should mirror the order in which controls appear on the screen, which is the default. If manual adjustments are made, it is difficult to maintain the correct order especially in the presence of the browser's address bar and other controls outside of the app. This can make a screen reader very difficult to use. When read by the screen reader, the controls should be presented in the same order in which they are seen on the screen, instead of an order that's less intuitive.
Add another input method	Tip	When an app contains a **Pen** control. This tip reminds you to include a separate method of input.	Add a **Text input** control in addition to the **Pen** control for an accessible experience.	Some users can't use a pen and require another way to provide information (for example, by typing a signature).

Next steps

Accessibility limitations in canvas apps

See also

- Create accessible apps

- Accessible app structure
- Accessible colors in Power Apps
- Show or hide content from assistive technologies for canvas apps
- Announce dynamic changes with live regions for canvas apps
- Accessibility properties

Accessibility limitations in canvas apps

Article • 03/05/2024

You can design and build accessible canvas apps with built-in controls. Sometimes, the combination of app design, use of various controls, and the configured control properties may result in an app to have an inaccessible user interface.

In this article, you'll learn about these accessibility limitations in canvas apps, and how can you improve the app interface to make it more accessible.

Dialogs and overlays

Dialogs and user interfaces that appear on top of other content are not supported. These overlays require focus management, hiding background content from screen readers, and appropriate control roles.

Consider the following:

- Use separate screens for each “dialog”.
- Use the **Notify** function.
- Create a code component that implements an accessible dialog.

Tabbed interfaces

A tabbed interface is made up of a list of tabs and a panel that shows content associated with the selected tab.

The only built-in control that supports tabbed interfaces is the **Tab list**. Avoid using **Galleries** to create tabs. The list of tabs should be navigable with arrow keys. Appropriate control roles and states are required.

Consider the following:

- Use the built-in **Tab list**.
- Create a code component that implements an accessible tabbed interface.

Custom tables

The only built-in control that supports two-dimensional data is the **Data Table**. Avoid using **Galleries** to present data in rows and columns. Rows and columns have to be annotated so that screen reader users can understand their structure and navigate cells.

Consider the following:

- Use the built-in **Data Table**.
- Present data in one direction only with a **Gallery**.
- Create a code component that implements an accessible table.

Custom combo boxes

It is possible to emulate a combo box by combining a **Text Input** and a **Gallery**. However, combo boxes assembled from built-in controls are not accessible. Combo boxes must handle arrow keys and set appropriate roles and states on its components.

Consider the following:

- Use the built-in **Combo box** or **Drop down**.
- Create a code component that implements an accessible combo box.

Scrollable content

Keyboard users can't scroll **Forms**, **Scrollable screens**, and **Containers** if there are no interactive controls inside them.

For **Galleries** with no interactive content, you can set **TabIndex** on the **Gallery** itself so that keyboard users can focus on it and use arrow keys to scroll it.

For **Labels**, setting **TabIndex** will make them focusable for keyboard scrolling. However, it will also turn them into buttons. Don't making labels scrollable if they're not meant to be buttons. Ensure the **Label** is large enough to fit text. The **AutoHeight** property can be used to automatically adjust the control's size when it is inside a **Form**, **Scrollable screen**, or **Gallery**.

Expandable sections

Expandable sections, also known as disclosures, contain content that is hidden until the user presses a button. There is no built-in support for these elements but there is a workaround.

Mention the expanded state in the **AccessibleLabel** of the button. For example, “Show more details. Collapsed.”.

Update the **AccessibleLabel** when the expanded state changes. Position the expanded content immediately after the button so that screen reader users can logically navigate

to it. Push other content down when the section expands.

Landmarks

You can create headings with **Label** controls. Navigation, banner, and other landmarks are not supported. Power Apps automatically sets the main landmark to the app screen.

For other landmarks, use a heading as a workaround.

Custom roles and states

There is no built-in support for custom roles and states. Hence, it's not recommended to create composite check boxes, sliders, and toggles from built-in controls.

Consider the following:

- Mention the control's role and state in its **AccessibleLabel**. For example, if an **Icon** is used as a check box, its label could be "Enable notifications. Check box. Checked.".
- Create a code component that sets WAI-ARIA roles and states as appropriate.

Custom keyboard handling

It is not possible to react to specific key presses. For example, you can't have custom behavior for arrow keys or the Escape key. Hence, it is not possible to compose list-like controls like radio buttons from built-in controls. Nor can you create overlays that can be dismissed with the Escape key.

Enter or Space key handling is supported with **OnSelect**. However, this property is also triggered by other input methods like mouse clicks. There is no way to distinguish the source of the event.

Focus management

SetFocus function can be used to change focus but it only works in limited scenarios.

It is not possible to detect when controls receive or lose focus.

Hide content from screen reader users only

There is no aria-hidden equivalent to show content to sighted users but hide it for screen reader users. Only a few scenarios are supported.

Next steps

Accessibility properties

See also

- Create accessible apps
- Accessible app structure
- Accessible colors in Power Apps
- Show or hide content from assistive technologies for canvas apps
- Announce dynamic changes with live regions for canvas apps
- Use the Accessibility checker

Send notification from an app

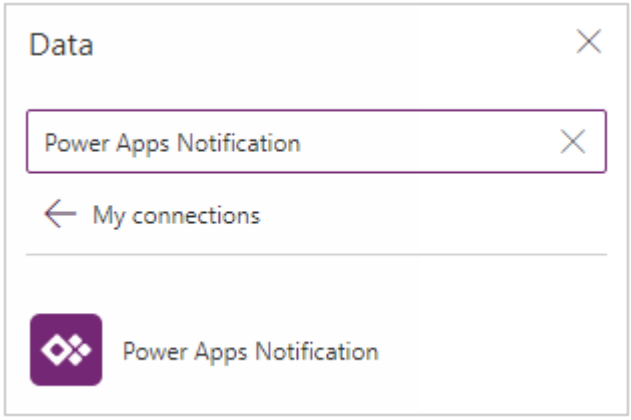
Article • 11/11/2023

You can send a push notification from one app to another or to the same app. In canvas apps, you can send notifications by using the Power Apps Notification connector.

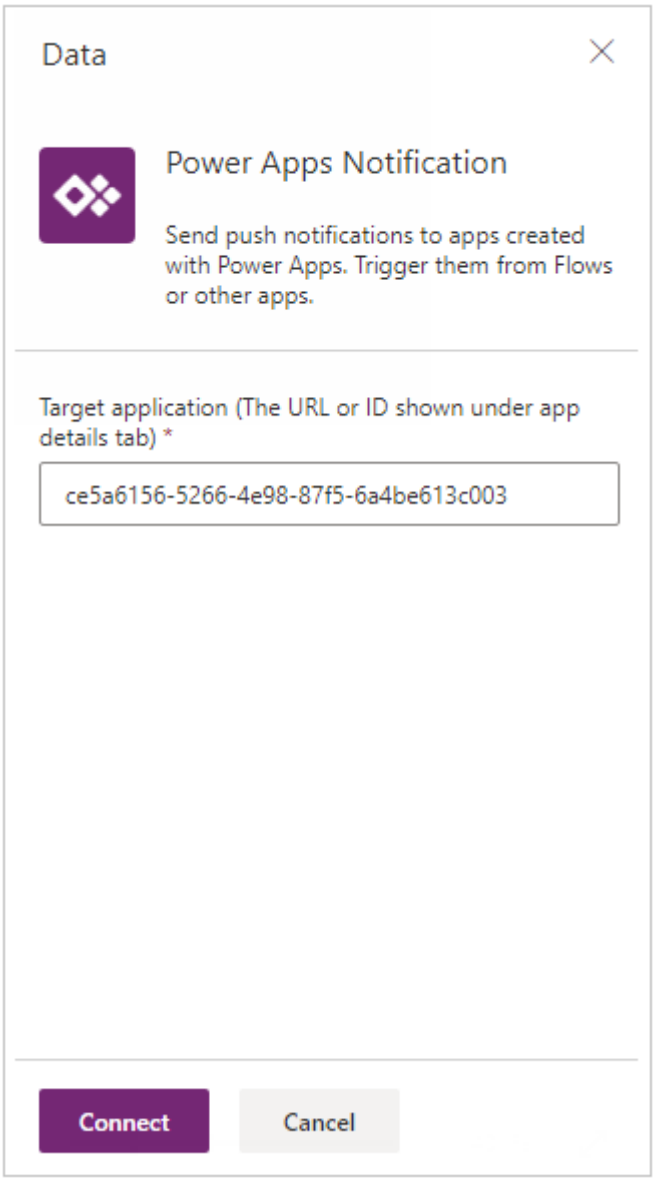
In this article, the sample app used for notifications is built from the default **Case Management** app template.

> **Note**
>
> Before you begin, create push notifications for the Power Apps mobile app. More information: Create push notifications for the Power Apps mobile app.

1. In Power Apps, go to the app to which you want to send push notifications.
2. Copy the app ID. More information: Get an app ID
3. In the left pane, select **Data** - **Connections**.
4. Edit the app. More information: Edit an app
5. Select **View** - **Data sources**.
6. Select **Add data source**.
7. Select **New Connection**.
8. Select **Power Apps Notification**.



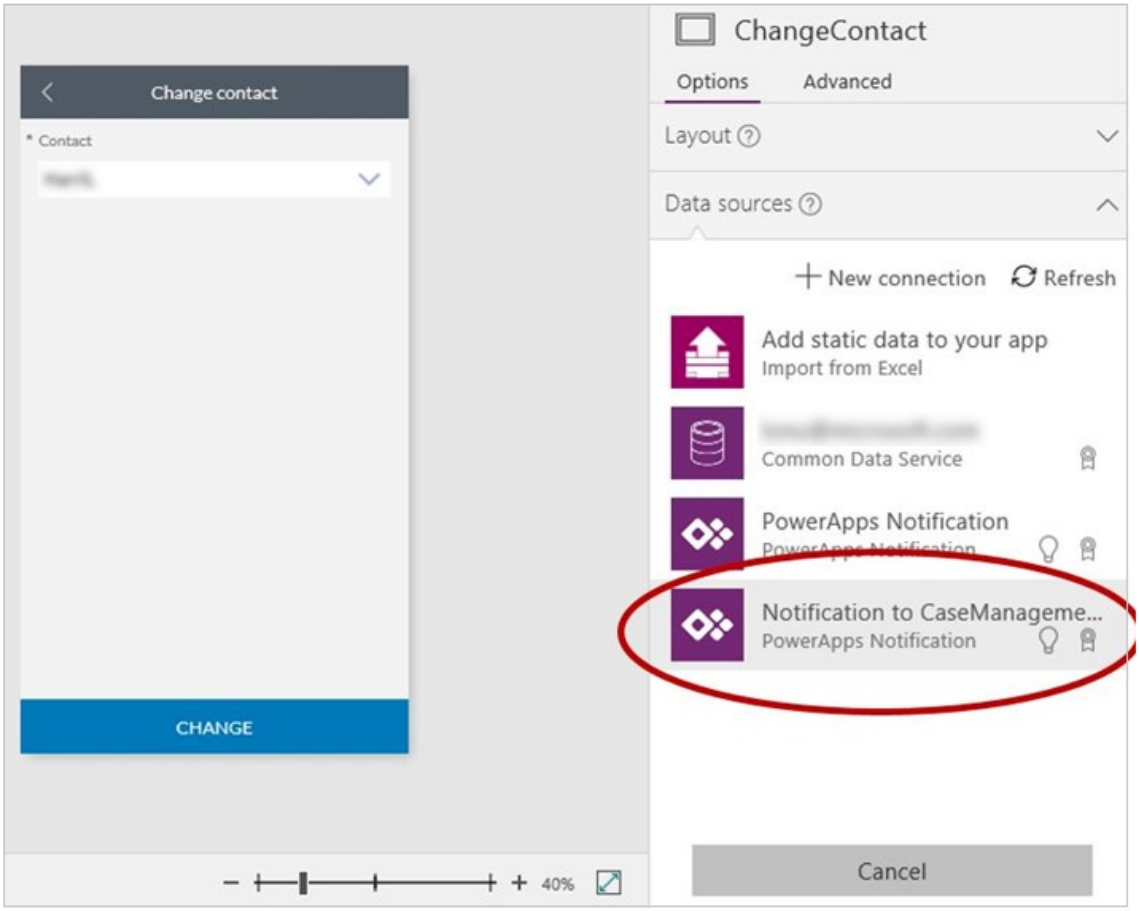
9. Paste the app ID copied from the previous step.



10. Select **Connect**.

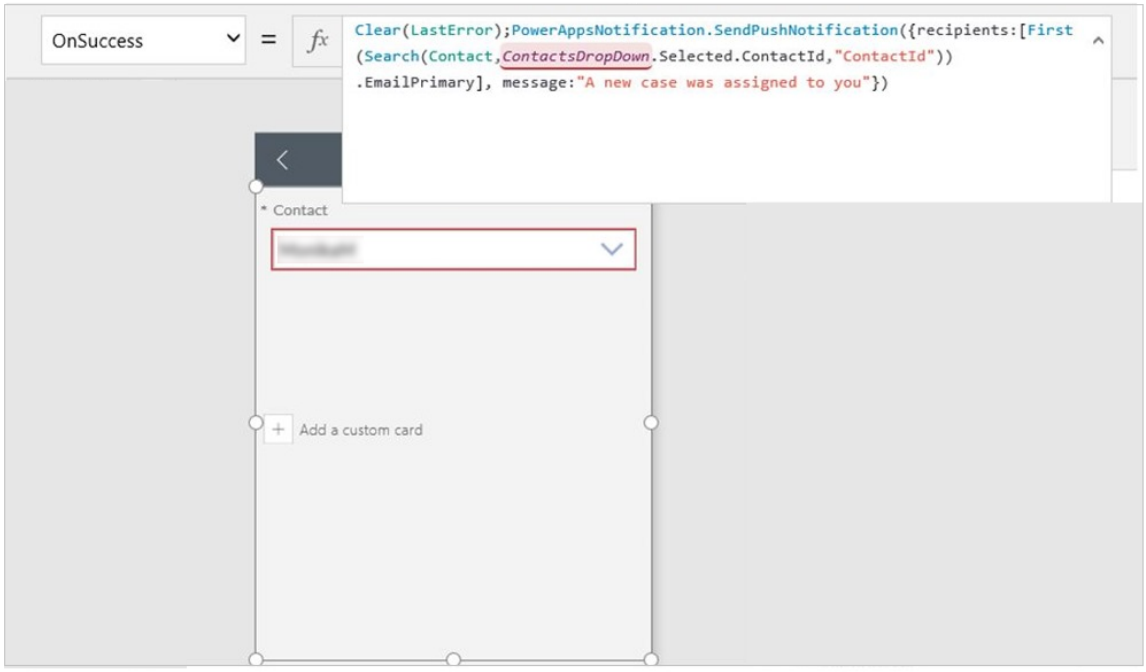
11. Add the push notification connection to the trigger app using the similar steps.

    In our example, we use the same app as the trigger app. The user who reassigns the case also triggers a push notification to the new case owner.



12. From the push notification connection, call the **SendPushNotification** method.

    In our example, we trigger this notification by using the **OnSuccess** property in a form.



Syntax

Expand table

Name	Description
SendPushNotification	Sends a push notification to the app that's specified in the connection settings for the notification.

Parameters

Expand table

Name	Type	Description
recipients	String array, required	A list of: • Email addresses for users or security groups • Object IDs for users or security groups in Microsoft Entra ID
message	String, required	The message body of the push notification.
openApp	Boolean, optional	Whether to open the app when the user taps the push notification.
params	Parameters, optional	Key-value parameters to pass with the notification. These can be further processed in the app to open a specific page and load a specific state.

Sample formulas

Send a basic notification.

Power Apps

```
PowerAppsNotification.SendPushNotification(
    {
        recipients: ["f60ccf6f-7579-4f92-967c-2920473c966b", "72f988bf-86f1-41af-91ab-2d7cd011db47"],
        message: "A new case was assigned to you."
    }
)
```

Send a notification that opens an app and passes along specific parameters.

Power Apps

```
PowerAppsNotification.SendPushNotification(
    {
```

```
        recipients: ["email1@contoso.com", "email2@contoso.com"],
        message: "message in the notification toast",
        params: Table({key:"notificationKey", value:"The value for
notificationKey"}),
        openApp: true
    }
)
```

See also

- Create push notifications for the Power Apps mobile app
- Power Apps Notification reference

Disconnect Git version control to edit canvas apps

Article • 05/14/2025

Git version control for editing apps has been removed and is no longer supported.

If your app currently uses this feature, follow these steps to disconnect it:

1. Open your canvas app for editing in Power Apps Studio. On the command bar, select **Settings**.
2. Select **Git version control** and then select **Disconnect**.

If you are using PASopa, download your app from your Git provider and pack the .zip file to an .msapp file. Once your .msapp file is uploaded to Power Apps, follow the steps above to disconnect.

> **Note**
>
> Git version control is replaced with the source control integration feature. More information: Source code integration

Understand new, preview, experimental, and retired features in canvas apps

Article • 04/29/2025

With every release, we make changes and add features to make Power Apps the best tool to fit your needs. We move the product forward.

At the same time, keeping your apps running smoothly is our top priority. Despite our best efforts, any change or improvement might unintentionally introduce a side effect, and your app might not work exactly the way it did before.

To ensure enhancements don't adversely affect current apps, we take features through a progression of stages. These stages give you a chance to try out the feature and provide us with feedback, and for us to gradually roll out the feature while watching for any problems. This article describes this process and how you can control your exposure to features that are under development.

At each stage, the number of people who use the feature increases, helping us to validate that the feature is what you need and that we're not introducing an unintended side effect.

Your feedback is critical throughout this process. Post your feedback in the Power Apps Community Forum.

Feature roll out stages

In short, the stages are:

- **New:** These features are new and generally available (GA), fully supported, and documented. Generally these features are on by default for new apps, but might take time to deploy everywhere. Enable for existing apps on your own schedule.
- **Preview:** These features are almost done and will be **New** soon. But at this stage there still might be some changes made. This stage is the last opportunity for feedback. These features are generally turned off by default and are documented. These features aren't GA, shouldn't be used in production, and are covered by the Preview terms of service.
- **Experimental:** An early stage "Preview," these features truly are experiments and might never reach GA. We're assessing the value proposition and design of the feature, and the feature might radically change or be removed completely at any time. These features are off by default and aren't generally documented. These features aren't GA, shouldn't be used in production, and are covered by the Preview terms of service.
- **Retired:** These features are GA features but are on their way to being removed from the product. They're still fully supported and document, but we either have a new, better way

to do the same thing or usage is low. Disable for existing apps on your own schedule. Generally, these features are off by default.

How long does a feature remain in each stage? It varies from feature to feature. We look at many factors, including the number of apps that use the feature, the number of issues reported, and how urgently the feature is needed. Features can remain in a stage for weeks to many months. We might also skip some stages if we don't believe it would be helpful.

New

Enable **New** features for all your new apps, and enable for existing apps when you're ready.

New features are recently added to the product. They're generally available and fully supported. You can use **New** features with confidence.

In general, **New** features are enabled by default when you create a new app. When a **New** feature is first introduced, it might not be enabled in all regions initially, allowing us time to deploy and gather information on usage and any problems. If you run into a problem with a feature, the switch can be used as a workaround to disable it until we can correct the problem.

Eventually, **New** features are no longer "new". If there's no breaking change, the switch is removed and the feature is enabled for all apps. If there's a breaking change, the old behavior is available in the app settings under **Retired** as a disable switch.

New features are fully documented in the Power Apps documentation.

Preview

Enable **Preview** features if you want to test upcoming functionalities before the feature is enabled for everyone.

Preview features are well on their way to being a valued part of the product. But, they aren't generally available yet and aren't fully supported, instead offered under the preview terms of service. **Preview** features shouldn't be used in production.

In general, **Preview** features aren't enabled by default when creating a new app. You should only enable a **Preview** feature for testing purposes and to provide feedback before the feature becomes generally available. This stage is the last stop for feedback before the feature is finalized.

Generally, **Preview** features graduate to **New**.

Preview features are documented in the Power Apps documentation. Use the Power Apps Experimental Features forum or Power Apps blog for more information. Use the "Experimental" forum as many **Preview** features start out as **Experimental** features and there may already be information or comments there that can be helpful to you.

Provide feedback on **Preview** features in the Power Apps Experimental Features forum. Again, "Experimental" also covers **Preview** features in the forum.

Experimental

Enable **Experimental** features if you're an early adopter, see something useful to you, and would like to help test the feature.

Experimental features are in an early-stage preview that is true to the name "experiment". This feature might significantly change or be removed at any time. These features are far from being generally available, aren't fully supported, and are offered under the preview terms of service. **Experimental** features shouldn't be used in production.

Experimental features are never enabled by default when creating a new app. You should only enable an **Experimental** feature for testing purposes and to provide feedback. If you don't know what the feature is referring to or its possible impact, you likely shouldn't turn it on.

Eventually **Experimental** features either graduate to **Preview** or are removed from the product.

Experimental features typically aren't documented. The short description in the **App settings** pane might be the only information about them. Use the Power Apps Experimental Features forum or Power Apps blog for more information.

Provide feedback on **Experimental** features in the Power Apps Experimental Features forum.

Retired

Enable **Retired** features only if you need an old behavior. When appropriate, disable these features in your existing apps.

Retired features, while fully supported and widely available, might be phased out in favor of newer, improved methods or due to low usage. Maintaining every feature alongside ongoing product updates incurs overhead.

When a feature moves from **New** to **Retired**, the meaning of its switch is often inverted. For example, where "Delayed load" was on by default as a **New** feature, the **Retired** version becomes "Disable delayed load" and is off by default.

Retired features are generally not enabled when creating a new app. They're provided for existing apps that still depend on the old behavior.

Retired features will eventually be removed from the product after enough time has passed and usage is low. Removals are well communicated ahead of time for breaking changes.

Retired features are fully documented in the Important changes (deprecations) coming in Power Platform.

Defaults

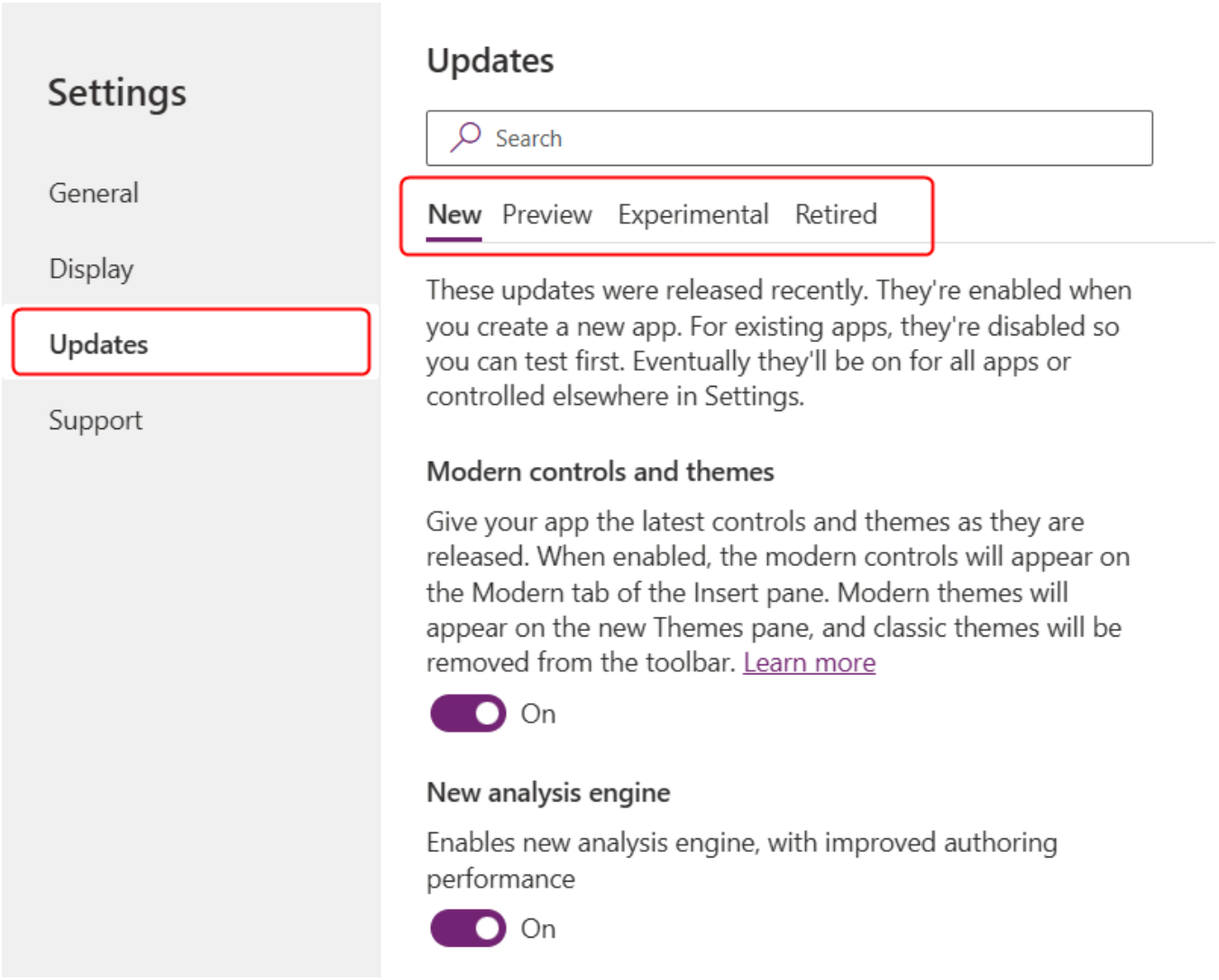
In general, when creating a *new* app:

- **New switches are On.** These features are generally available and fully supported. There might be a small delay for the switch to be turned on by default as we continue rolling out.
- **Preview switches are Off, but can be enabled for testing and feedback.** Enable the features that are interesting to you before they're enabled for all apps, for testing and feedback. Preview features aren't ready for production.
- **Experimental switches are Off.** These features are for early adopters and should only be used with caution; they can be changed or removed at any time. Experimental features aren't ready for production.
- **Retired switches are Off.** These features are generally available and fully supported, but will eventually be removed from the product. Avoid using these features.

As *existing* apps go through their lifecycle, you might want to take advantage of new features or to remove dependencies on retired features by adjusting these switches in your app. Turning **New** switches On and **Retired** switches Off brings existing apps into alignment with new apps and the future of the product.

Controlling which features are enabled

Experimental and preview features are listed in the app's **Settings** > **Updates**. From within the app, select **Settings**, and then select **Updates**. Select **New**, **Preview**, **Experimental**, or **Retired** features using the available tabs.



Each feature has a toggle switch. **Off** generally means that the feature is disabled, except in the **Retired** section where the title of the feature indicates if the switch disables the feature.

In some cases, you might need to close and reopen the app after you change a setting. The feature description should indicate when you must perform this step.

At the top of the **Settings** panel, **General** settings provide control over fully shipped, generally available features that don't belong in any of these feature categories and are a stable part of the product. We limit the number of these settings as much as possible to keep things simple and to reduce the size of our testing matrix.

These settings are specific to each app, so changing a toggle switch affects only the app that's currently open. If you create an app, these switches revert to their default settings for that app.

Preview and Experimental features without a switch

Sometimes a tag is added in the product to indicate that a particular feature is Preview or Experimental without having a switch in **Updates**. For example, a control in an **Insert** menu

might have the name **Grid (Preview)** or the entry point for the Monitor was once **Monitor (Preview)**. In these cases, the feature can't be used without first seeing this tag.

Like the **Preview** and **Experimental** feature categories previously described, these features shouldn't be used in production and are offered under the preview terms of service.

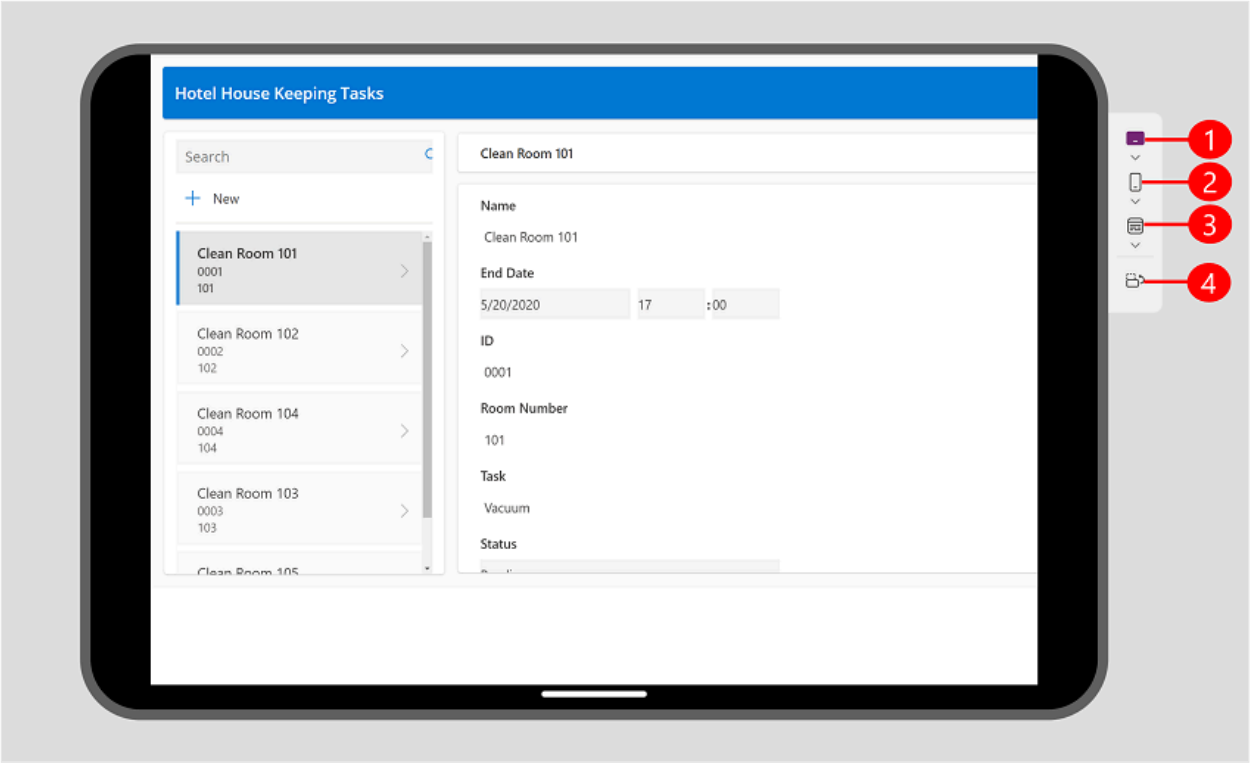
Preview an app

Article • 05/19/2023

When you create an app, it's helpful to preview your app and see how it will look for your users. Use the preview app feature to see how your app appears on various devices and in different sizes. Previewing an app is especially beneficial when you're building a responsive app.

To preview an app, open the app for editing in Power Apps Studio, in the actions menu, select **Preview the app**.

Use the preview options to see how your app looks on different devices using the device picker.



Legend:

1. **Tablet Devices**: Allows you to select from common tablet devices from several manufacturers, along with the ability to select a custom size based on your needs. This option shows a device frame and device safe zone, which is shown as the black area either on the top and bottom or sides depending on the orientation of the device. It also shows a black area with the home bar at the bottom.
2. **Mobile devices**: Lets you select from common phones from different manufacturers, along with the ability to select a custom size based on your needs. This option shows a device frame and device safe zone, which is shown as the

black area either on the top and bottom or sides depending on the orientation of the device. It also shows a black area with the home bar at the bottom.

3. **Web view**: See how your app looks like on a web browser. You can select a window size to fill the browser window, or you can select canvas size to see how the app looks while authoring without leaving the preview.
4. **Orientation Switcher**: Switch between vertical and horizontal orientation. The **Orientation Switcher** option is disabled if the **Lock Orientation** setting turned on.

If you're developing a responsive app, the content within the app adjusts to fit the screen size of the device being used.

Watch this video to learn how the device picker helps you confirm that your app is adjusting correctly.
https://learn-video.azurefd.net/vod/player?id=6fb4ee34-2393-4bf2-b3b9-749b8a7cd257&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fpreview-app

Display settings

The way your app behaves with each preview option is determined by app type and your display settings.

- If an app is created using the phone layout, you can still preview it on the web and tablet devices using the device picker. However, if an app isn't created using a responsive layout, it then appears in a phone shape on the web and selected tablet device. You get a message explaining why the preview is narrow. If you'd like an app to be responsive to phone, tablet, and web sizes then we recommend selecting the tablet option when you create an app. You can use responsive auto layout containers to determine the flow of the content for different screen sizes including phones. More information: Building responsive canvas apps

- Under Display settings:
  - **Scale To Fit**: When set to **On** your entire app is scaled to fit the available space. If a smaller space is available to the app, you may notice empty space on the sides to accommodate the scaling. When a user plays the app, they will also see the empty space.
  - **Lock Orientation**: When set to **On**, the **Switch Orientation** button in the device picker is disabled.

Known Issues

- If **Scale To Fit** setting is on and **Lock Aspect Ratio** is off, your preview won't be accurate. This configuration isn't recommended. More information: Change screen size and orientation of canvas apps
- If **Scale to Fit** is off and you resize your browser window while in preview mode, the preview won't be accurate. To address the issue, close and reopen the preview once your browser window is in the desired size.

Feedback

Was this page helpful? Yes No

Provide product feedback

Save and publish canvas apps

When you save changes to a canvas app, you save those changes for yourself and anyone else who can edit the app. To make your changes available to everyone the app is shared with, publish the app.

What you'll learn:

- How to save your app with different options
- How to turn AutoSave on or off
- How to publish your app to share changes with users
- How to identify and manage app versions

For more information about sharing an app, see Share an app.

Save your app

Open your app for editing in Power Apps. Save your work in several ways:

- Select **Save** to save any unsaved changes to your app. Each time you save, a new version appears in the app version history.

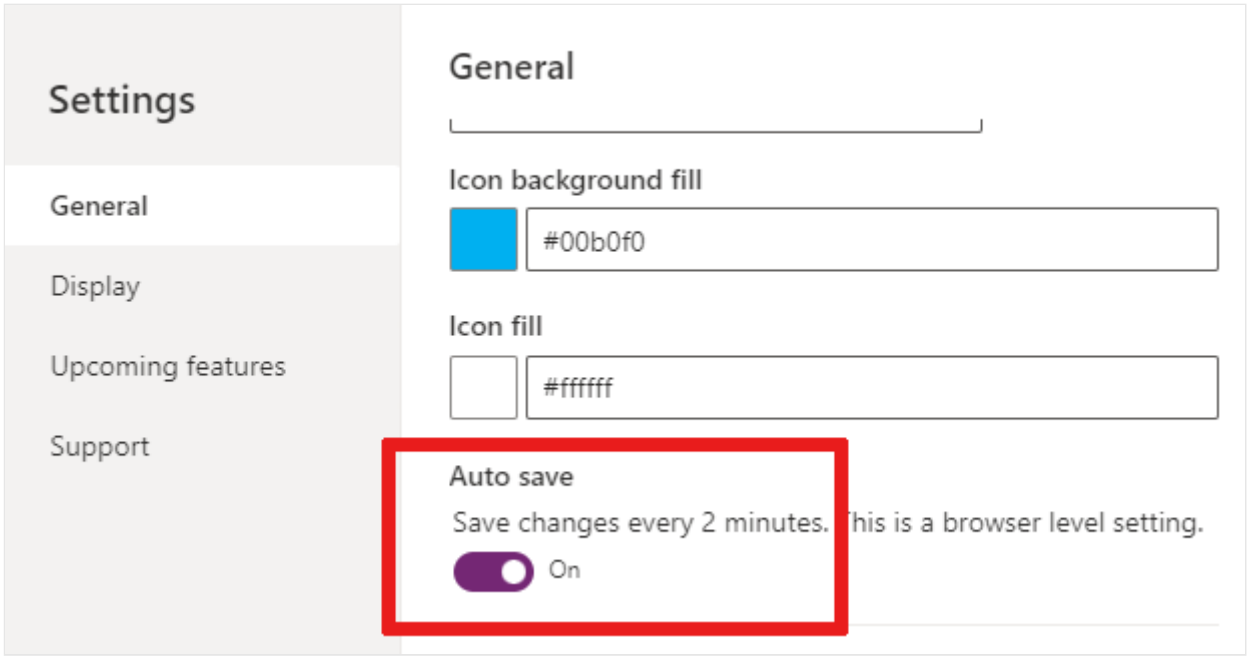
- In the dropdown menu, choose one of the following options:
  - **Save with version notes**: Save and add notes about your update.
  - **Save as**: Duplicate the app with a different name.
  - **Download a copy**: Download a local copy of the app.

Turn AutoSave on or off

Power Apps saves your work automatically every two minutes.

1. On the command bar, select **Settings**, and then go to the **General** tab.

2. In the **Auto save** section, set the **Auto save** toggle to **On** or **Off**.



> [!NOTE]
> - When you publish a canvas app, your app updates and runs on the latest version of Power Apps. Your app gets all the latest features and performance upgrades added since you last published. If you haven't published an update in several months, you see an immediate performance benefit when you republish the app.
> - To retrieve app details faster during startup, some data stores locally on user devices in the browser cache. Information stored includes app environment and connection details. This data stays in the browser based on browser storage limits. Users can clear stored data based on these instructions for each browser.

Publish your app

1. On the command bar, select **Publish**. If the app isn't saved, you're prompted to save it. After you save it, select **Publish** again.

2. Enter your app's description, and then select **Publish this version**.

Create an app description with Copilot (preview)

If your app is in a managed environment, you can use AI to help create a description for your app.

In the **Publishing** dialog box, select **Create descriptions using AI** when you save and publish your app.

You can also generate an app description by going to **Settings** > **General**. Under **Description**, select **Create descriptions using AI**.

Admins can disable this feature for managed environments in Power Platform admin. For more information, see Managed Environments panel.

> **Note**
>
> If you don't add a description before publishing the app, Copilot generates one after the app is published. Makers can edit this description if Copilot doesn't generate it correctly.

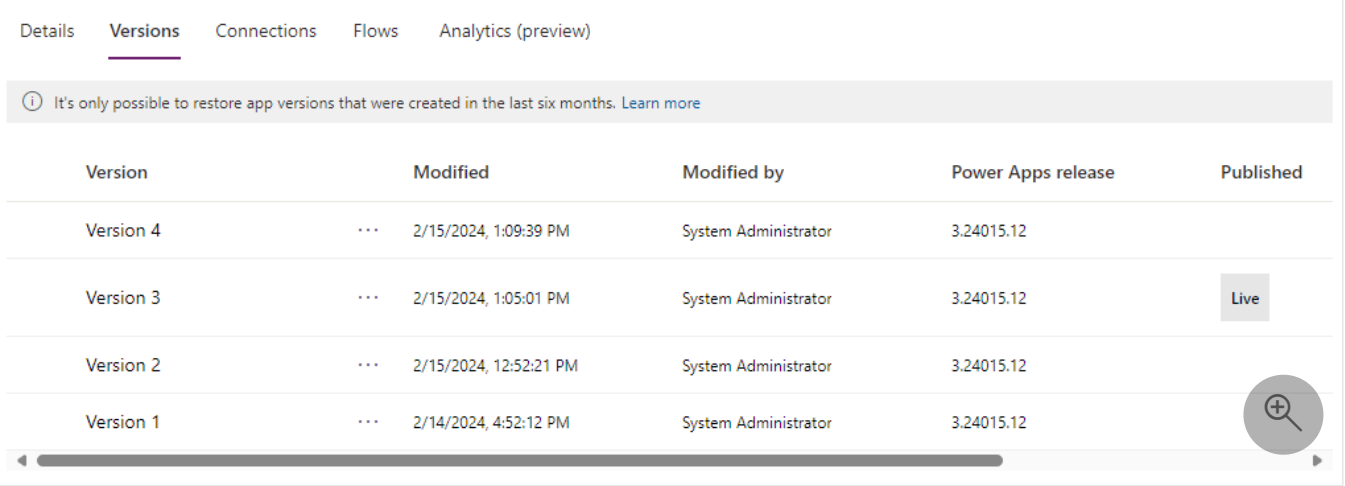
> **Important**
>
> - To use this capability, your app must be in a **managed environment**.
> - Preview features aren't meant for production use and can have restricted functionality. These features are available before an official release so customers get early access and provide feedback.
> - For more information, go to our **preview terms**.
> - This capability is powered by **Azure OpenAI Service**.
> - This capability isn't available in all regions yet and can be subject to usage limits or capacity throttling.
> - To learn about capabilities and limitations of AI-powered and Copilot features in Power Apps, see **Responsible AI FAQs for Power Apps**.

Identify the live version

To see all versions of an app:

1. Go to Power Apps, and in the navigation pane, select **Apps**.
2. Select the **Commands** icon ⋮ next to the app name.
3. Select **Details** > **Details**.
4. Go to the **Versions** tab.

The **Live** version is published for everyone the app is shared with. Only users with edit permissions can use the most recent version of the app.



To publish the most recent version, select the publish icon while editing your app.

> **Note**
>
> - New published changes to an app can take a few seconds to show when you launch the app. Publish time depends on the app's complexity, and it can take a few more minutes to finish.
> - If you already have an app open when a new version is published, reload the app to get the latest changes.
> - To reduce wait time when using your app, the app preload capability is on. You can turn it off. For more information, see Overview of creating performant apps.

In-app notifications for an updated version of the app

When users wait for an app to update, they see a notification that says **A new version of this app is coming. We'll let you know when it's available.**

When the published changes are ready, users see a notification that says **You're using an old version of this app. Refresh to use the latest version.** Users select the **Refresh** button to get the latest version of the app.

Notification availability

Expand table

Scenario	Availability
Canvas app on web	Generally available
Customized SharePoint forms	Not available
Canvas app embedded in Teams	Not available
Canvas app embedded in Power BI	Not available
Power Apps web part	Not available
Canvas app embedded in an iframe	Generally available

Next steps

- Run the app in a browser or on a phone.
- Rename an app from Power Apps.
- Restore an app if you have more than one version of the app.

Last updated on 01/14/2026

Share a canvas app with your organization

10/10/2025

When you create a canvas app, you can define which users in your organization can use the app, modify it, and even re-share it. You can specify individual users by name or designate a security group in Microsoft Entra ID.

Prerequisites

Before you share an app, you must save it (not locally) and then publish it.

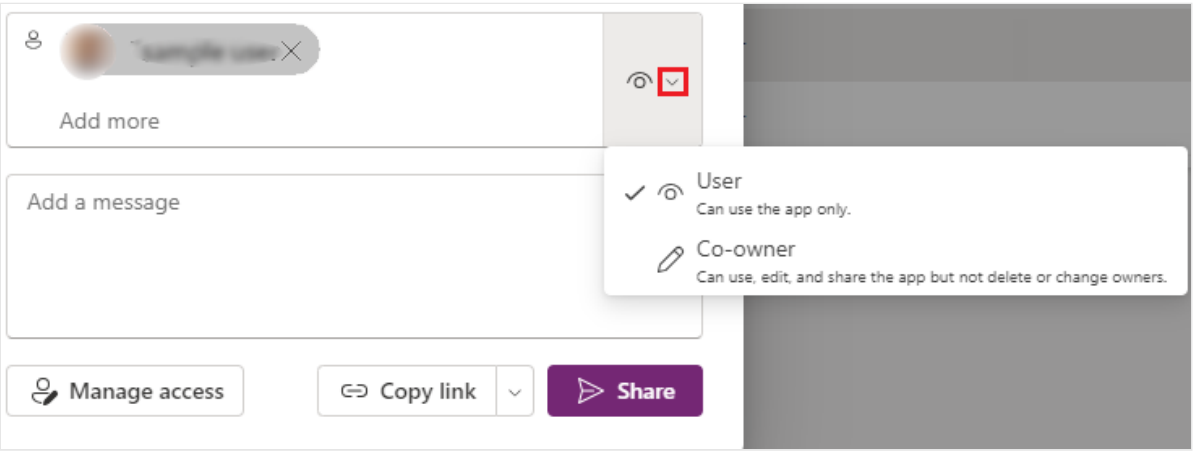
- Give your app a meaningful name and a clear description, so that people know what your app does and they can easily find it in a list. Select **Settings** > specify a name, and then enter a description.
- Whenever you make changes, you must save and publish the app again if you want others to see those changes.

> **Important**
>
> For a shared app to function as you expect, you must also manage permissions for the data source or sources on which the app is based, such as Microsoft Dataverse or Excel. You might also need to share other resources on which the app depends, such as flows, gateways, or connections.

Share an app from Power Apps

1. Sign in to Power Apps.
2. On the left navigation pane, select **Apps**.
3. Select the app that you want to share.
4. On the command bar, select **Share**. Or select the **Commands** menu next to your app name and then select **Share**.
5. In the **Share** dialog box, type the names or alias of the user or security groups in Microsoft Entra ID.
6. Select the down arrow to choose permissions for the app. Options include:
    - **User**: Can use the app only.

- **Co-Owner**: Can use, edit, and share the app but not delete or change owners.



- If your app connects to a Dataverse table, the permissions dropdown will display a **More security roles** option. Select the appropriate security roles, and those roles will be automatically assigned to users when you share the app.

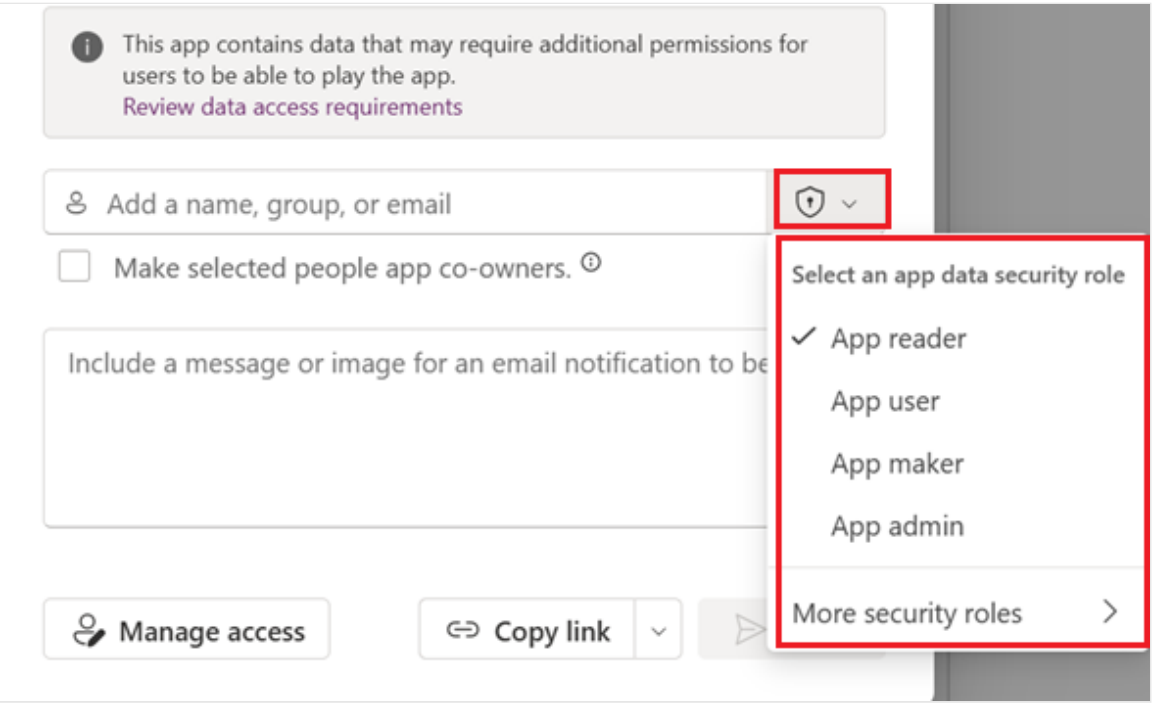
- If your admin enabled app level security roles in Power Platform Admin, makers with the **System Administrator** security role can give collaborators app-level privileges when sharing. Makers assign privileges from the security role picker, with App reader, App user, App maker, or App admin as the security role options. To edit assigned security roles, unshare and reshare the app to assign the appropriate security role under Manage Access.

    - **App reader**: This role gives read privilege for both user or team owned tables and org owned tables.

    - **App user**: This role gives full access to only their own records in user or team owned tables, and read privilege to org owned tables.

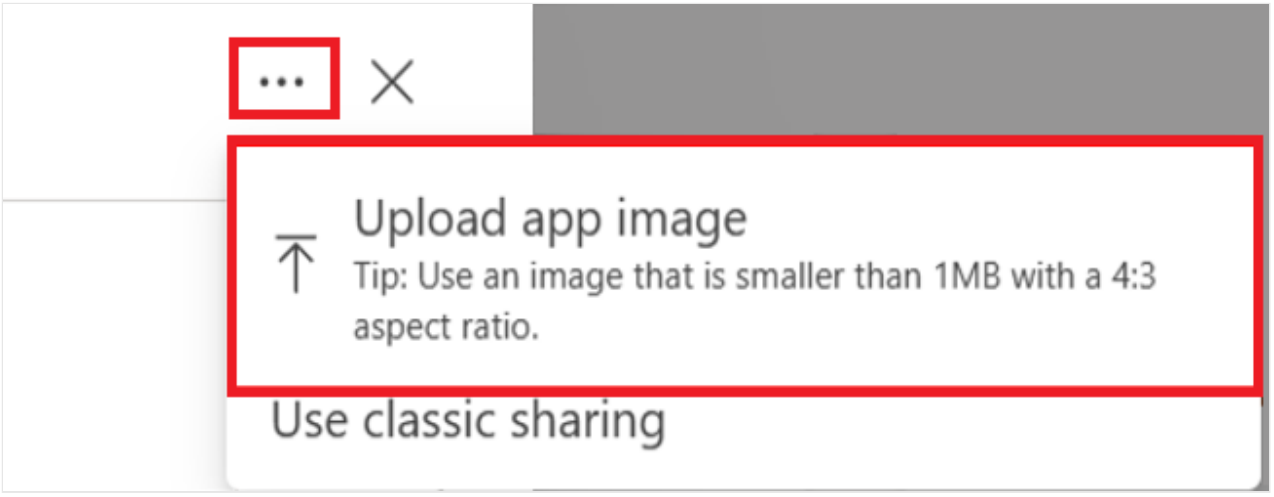
    - **App maker**: This role gives create and read privilege for all records in user or team owned tables, but only write, delete, assign, share, append, and append to privileges for their own records in user or team owned tables. Additionally, it gives read privilege to org owned tables.

    - **App admin**: This role gives full access to all records in user or team owned tables and org owned tables.

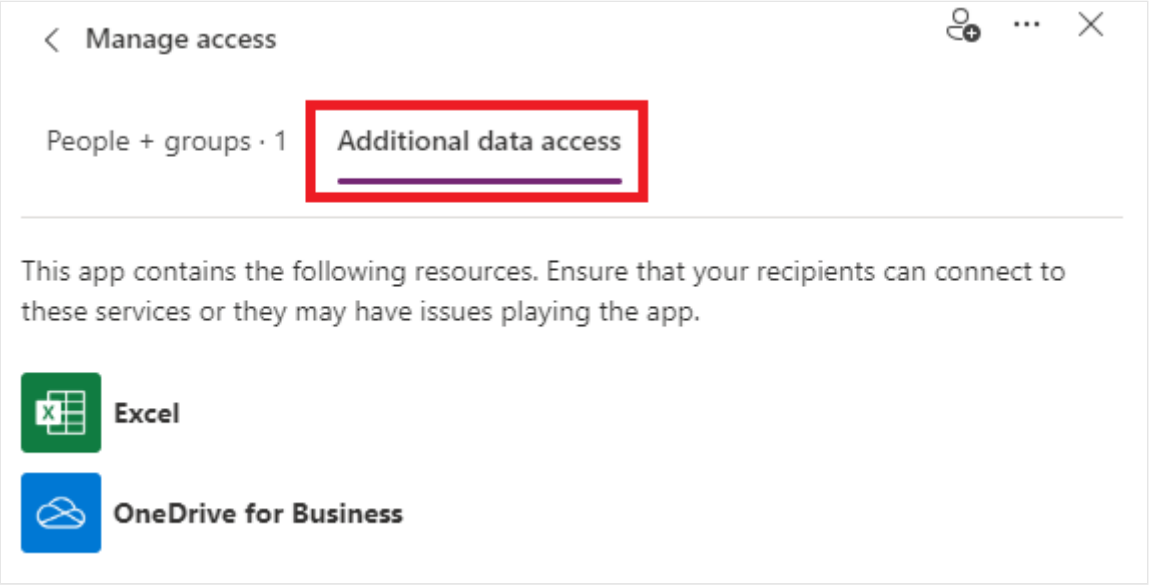


7. Optional steps:

- Select the **overflow menu (...)** at the top-right corner and then select **Upload app image** to include an image of the app in the email.



- Select **Manage access** to displays app access details, including current users and co-owners. You can also edit user access and assign security roles here. The **Additional data access** tab shows app connections to data sources like Dataverse tables or Excel files on OneDrive for Business. To manage security roles for data sources other than Dataverse, such as Excel files on OneDrive, ensure you share these data sources with the app users.



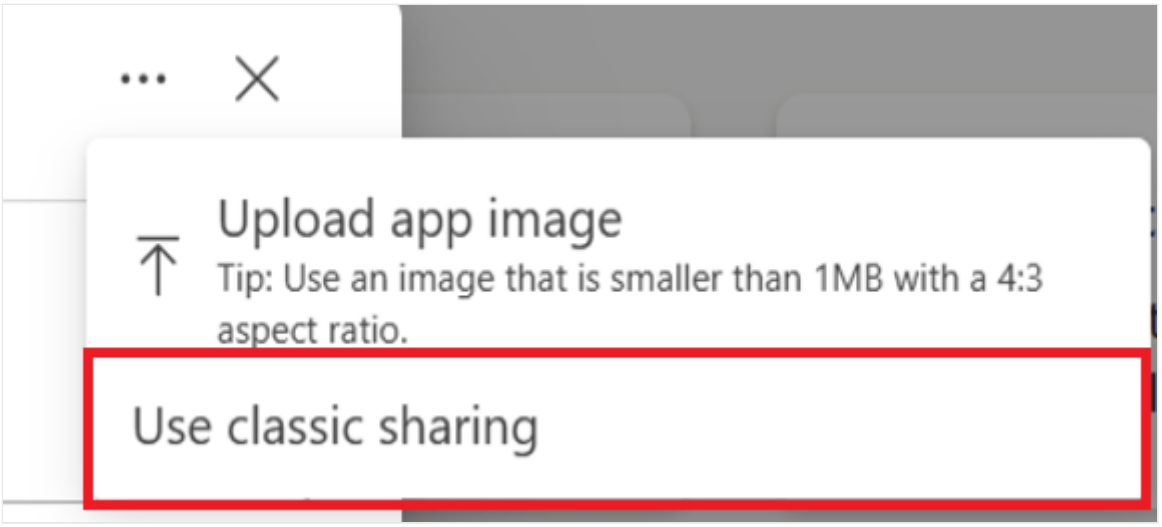
8. Add an optional message and then select **Share**.

> **Note**
>
> - To learn about sharing apps outside of your organization, see Share a canvas app with guest users.
> - To learn about sharing limits applicable to canvas apps in managed environments, see Sharing limits.

Classic app sharing experience

You can still use the classic app sharing experience by selecting the classic sharing option.

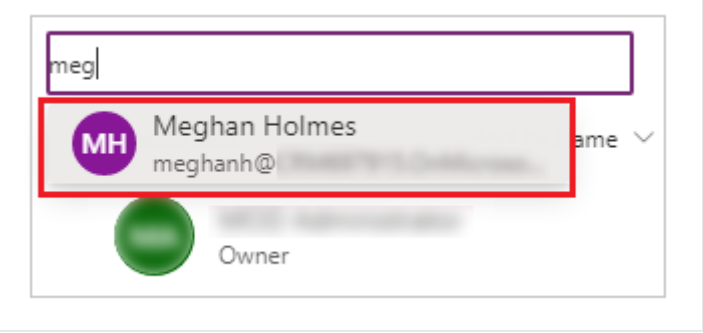
1. Sign in to Power Apps.
2. On the left navigation pane, select **Apps**.
3. Select the app that you want to share.
4. On the command bar, select **Share**. Or select the **Commands** menu next to your app name and then select **Share**
5. Select the **overflow menu (...)** at the top-right corner and then select **Use classic sharing**.



6. Specify the name or alias the users or security groups in Microsoft Entra ID with whom you want to share the app.

   You can share an app with a list of aliases, friendly names, or a combination of those (for example, **Meghan Holmes <meghan.holmes@contoso.com>**) if the items are separated by semicolons.

   If several people have the same name but different aliases, the first person found is added to the list. A tooltip appears if a name or alias already has permission or can't be resolved.

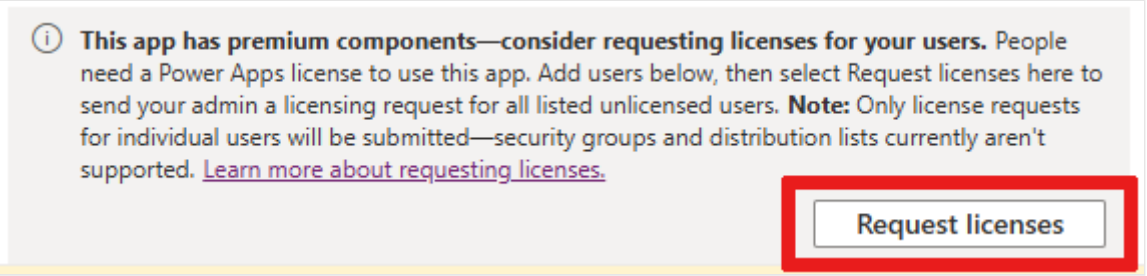


> **Note**
>
> You can't share an app with a distribution group in your organization or with a group outside your organization.

> **Important**
>
> To avoid degraded experiences, use a security group when sharing the app with over 100 users.

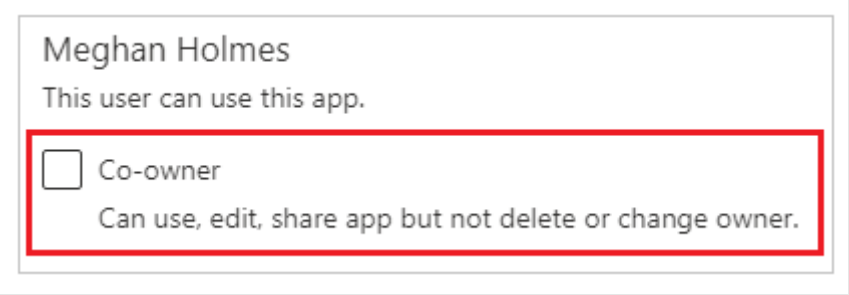
7. If your app contains premium components, such as a map or address input, users must have a Power Apps license to use the app. To request licenses for the users of your app,

select **Request licenses**, which submits the request to your admin.



> **Note**
>
> You can't request licenses for security groups or distribution lists. For more information about requesting licenses, see Request Power Apps licenses for your app users.

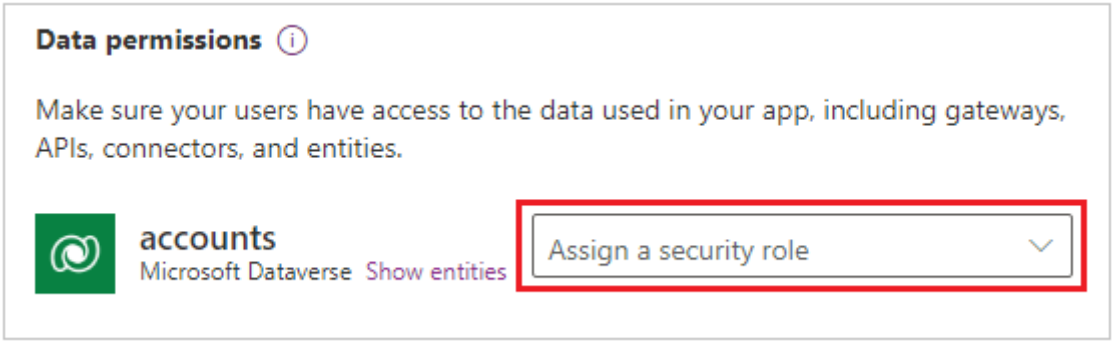
8. If you want to allow users to edit and share the app, select the **Co-owner** check box.



In the sharing interface, you can't grant Co-owner permission to a security group if you created the app from within a solution. However, its possible to grant co-owner permission to a security group for apps in a solution by using the Set-PowerAppRoleAssignment cmdlet.

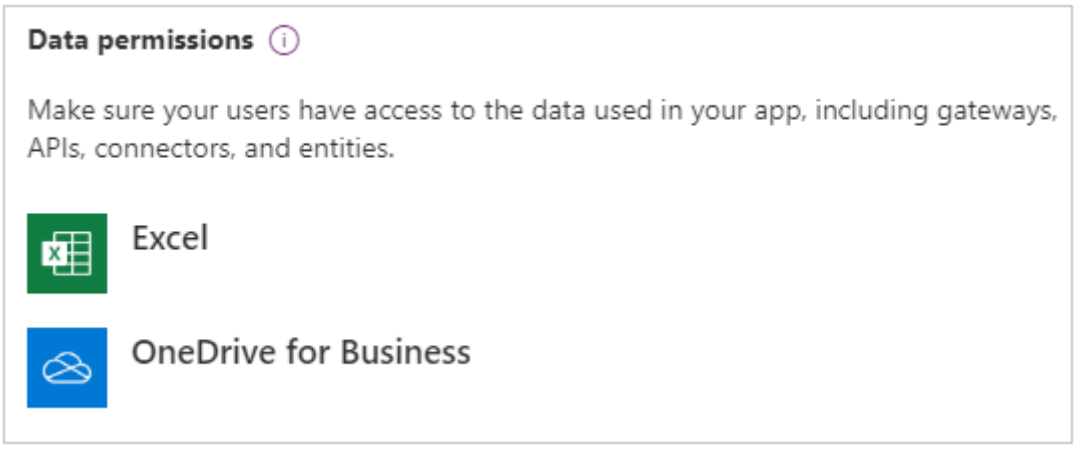
9. If your app connects to data for which users need access permissions, specify security roles as appropriate.

   For example, your app might connect to a table in a Dataverse database. When you share such an app, the sharing panel prompts you to manage security for that table.



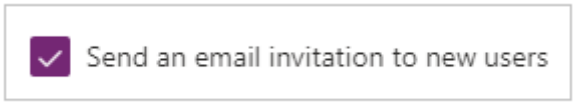
For more information about managing security for a table, go to Manage table permissions.

If your app uses connections to other data sources—such as an Excel file hosted on OneDrive for Business—ensure that you share these data sources with the users you shared the app with.



For more information about sharing canvas app resources and connections, go to Share canvas app resources.

10. If you want to help people find your app, select the **Send an email invitation to new users** check box.



11. At the bottom of the share panel, select **Share**.

Users can now run the app by using Power Apps Mobile on a mobile device or from AppSource on Microsoft 365 in a browser. Co-owners can edit and share the app in Power Apps.

If you sent an email invitation, users can also run the app by selecting the link in the invitation email:

- If a user selects the **Open the app** link on a mobile device, the app opens in Power Apps Mobile.
- If a user selects the **Open the app** link on a desktop computer, the app opens in a browser.
- If a user selects the **Microsoft Teams** link, the app opens in Microsoft Teams.

If you selected **Co-owner** while sharing, the recipients will see a link for **Power Apps Studio** that will open the app for editing using Power Apps Studio.

To change permissions for a user or a security group

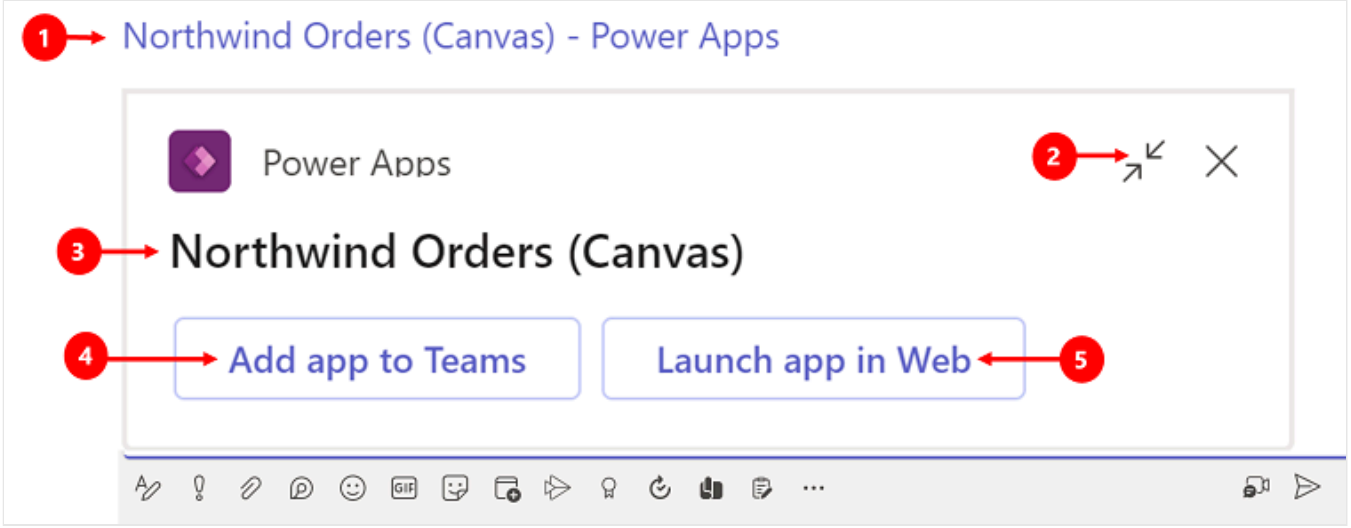
- To allow co-owners to run the app but no longer edit or share it, clear the **Co-owner** check box.
- To stop sharing the app with that user or group, select the **Remove** (x) icon.

Share apps using Teams

You can share a link to your app in a Teams chat. The user that you share the app with needs to be on the same tenant and have access to the app that you're sharing.

> ⓘ **Note**
>
> - To add canvas apps to Teams using the web player:
>   - Your organization must have **Allow interaction with custom apps** turned on. More information: Manage custom app policies and settings in Microsoft Teams
>   - Your organization must allow **Shared Power Apps**. More information: Manage Microsoft Power Platform apps in the Microsoft Teams admin center
> - If you're the sender or receiver and get a **This app cannot be found** error when you select **Add app to Teams**, this means you don't have access to the app. To get access, contact your admin.
> - If you're the sender and you don't have the Power Apps personal app installed in Teams, then a preview card will appear with a **Show Preview** button. To unfurl the full adaptive card, select **Show Preview**.

To share the app, copy the app link from your web browser and paste it into the Teams chat. Before you send the link, you'll see a preview of your message.



Legend:

1. **Link to app**: Select the web link to open the app.
2. **Collapse preview**: Select to close the preview. When you collapse the preview then the user that you send the link to will only get a web link to open the app. They won't see the buttons to add the **App to Teams** or **Launch app in web**.
3. **Name of app**: Shows the name of the app.
4. **Add app to Teams**: Select to add the app to Teams.
5. **Launch app in Web**: Open the app directly in your browser.

Request licenses for your users

When sharing an app that requires a license for use, you can request Power Apps licenses for your users. For more information, see Request Power Apps licenses for your app users.

Security group considerations

- All existing members of the security group inherit the app permissions. New users joining the security group will inherit the security group permissions on the app. Users leaving the group will no longer have access through that group, but those users can continue to have access either by having permissions assigned to them directly or through membership in another security group.

- Every member of a security group has the same permissions for an app as the overall group does. However, you can specify greater permissions for one or more members of that group to allow them greater access. For example, you can give Security Group A permission to run an app. And then, you can also give User B, who belongs to that group, Co-owner permission. Every member of the security group can run the app, but only User B can edit it. If you give Security Group A Co-owner permission and User B permission to run the app, that user can still edit the app.

- Users must explicitly be a member of the security group. If a user is an owner of the group, they must also assign themselves as a member of the group to inherit app permissions.

Share an app with Microsoft 365 groups

You can share an app with Microsoft 365 groups. However, the group must have security enabled. Enabling security ensures that the Microsoft 365 group can receive security tokens for authentication to access apps or resources.

Using Microsoft Graph:

1. Install the Microsoft Graph module.

```PowerShell
Install-Module Microsoft.Graph -Scope CurrentUser
```

2. Connect to Microsoft Graph. For information about the administrator roles required to manage groups, see Microsoft Entra roles for managing groups.

```PowerShell
Connect-MgGraph -Scopes "Group.ReadWrite.All"
```

3. Set the Object ID of Microsoft 365 group.

```PowerShell
$ObjectID = "<ObjectID>"
```

4. Enable security.

```PowerShell
Update-MgGroup -GroupId $ObjectID -BodyParameter @{ SecurityEnabled = $true }
```

5. Verify the update.

```PowerShell
Get-MgGroup -GroupId $ObjectID | Select-Object DisplayName, SecurityEnabled
```

> **Note**
>
> You must be the owner of the Microsoft 365 group to enable security. Setting the **SecurityEnabled** property to **True** doesn't affect how Power Apps and Microsoft 365 features work. This command is required because the **SecurityEnabled** property is set to **False** by default when Microsoft 365 groups are created outside of Microsoft Entra ID.

After a few minutes, you can discover this group in the Power Apps sharing panel and share apps with this group.

Manage table permissions for Dataverse

If you create an app based on Dataverse, you must also ensure that the users you share the app with have the appropriate permissions for the table or tables used by the app. Particularly, those users must belong to a security role that can do tasks such as creating, reading, writing, and deleting relevant records. In many cases, you'll want to create one or more custom security roles with the exact permissions that users need to run the app. You can then assign the role to each user as appropriate.

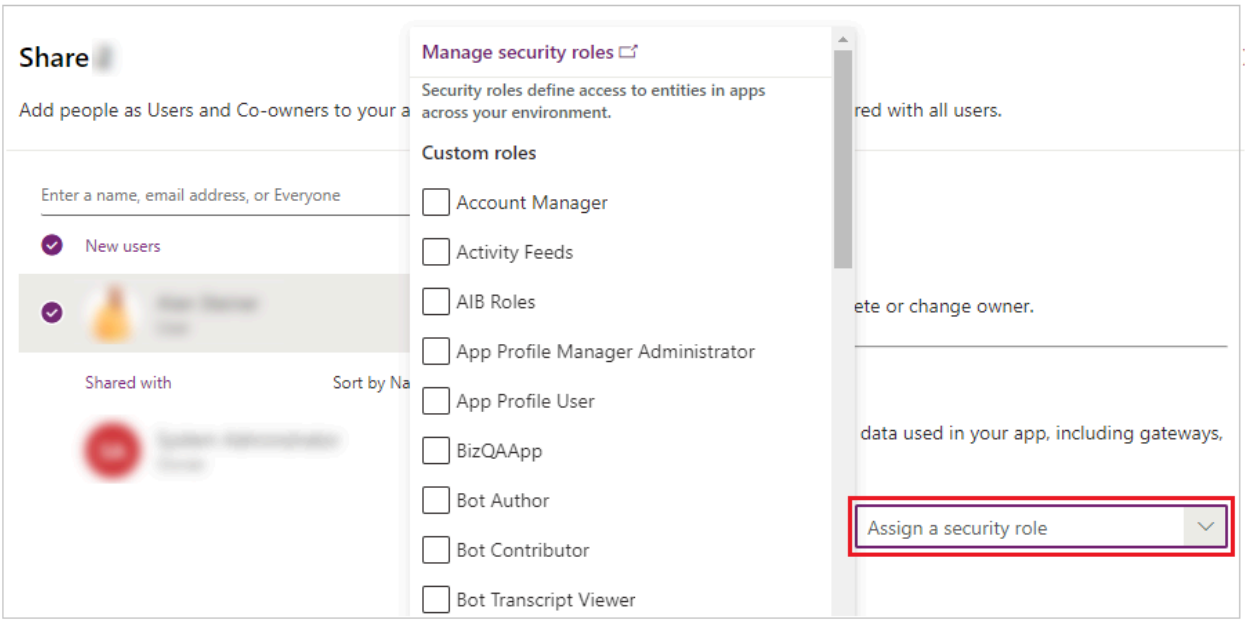
> **Note**
>
> - You can assign security roles to individual users and security groups in Microsoft Entra ID, but not to Microsoft 365 groups.
> - If a user isn't in the Dataverse root business unit, you can share the app without providing a security role, and then set the security role directly.
> - After a security role is assigned to a user or group, you can't unassigned it when you share an app. However, you can still unassign security roles through the admin portal.

Prerequisite

To assign a role, you must have **System Administrator** permissions for a Dataverse database.

To assign a security group in Microsoft Entra to a role

1. In the sharing panel under **Data permissions**, select **Assign a security role**.
2. Select the Dataverse roles that you want to apply to the selected Microsoft Entra users or groups.



> **Note**
>
> When you share an app that's based on an older version of Dataverse, you must share the runtime permission to the service separately. If you don't have permission to do this, see your environment administrator.

Next steps

Share a canvas app with guest users

See also

Edit an app
Restore an app to a previous version
Export and import an app
Delete an app

Share a canvas app with guest users

06/27/2025

Canvas apps can be shared with guest users of a Microsoft Entra tenant. You can invite external business partners, contractors, and third parties to run your company's canvas apps.

Watch this video to learn how to share an app with guests: https://learn-video.azurefd.net/vod/player?id=84cd1908-2ca9-415b-a819-3ffb61e55eca&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fshare-app-guests

Prerequisites

- In Microsoft Entra ID, enable B2B external collaboration for the tenant. More information: Configure external collaboration settings for B2B in Microsoft Entra External ID

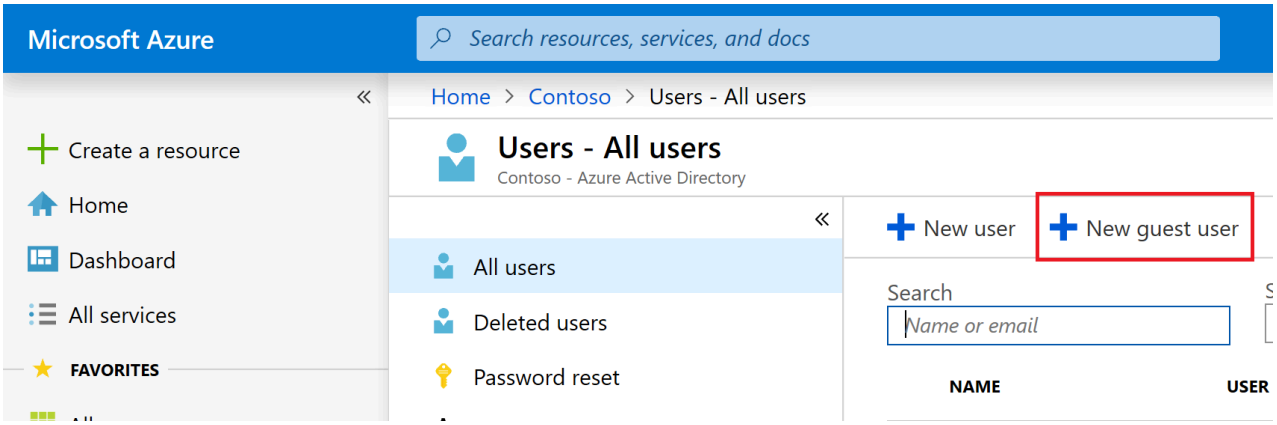
  > **Note**
  >
  > B2B external collaboration is enabled by default. However, you need to verify that the settings weren't changed by a tenant admin. For more information about Microsoft Entra B2B, see Overview: B2B collaboration with external guests for your workforce.

- Access to an account that can add guest users to a Microsoft Entra tenant. Admins and users with the Guest Inviter role can add guests to a tenant.
- To access an app that connects to Dataverse, the guest user must have a license with Power Apps use rights that matches the capability of the app. The exception to this prerequisite is when an app is hosted in a Microsoft Dataverse for Teams environment.

> **Note**
>
> Ensure that you perform these steps on the **resource tenant**, and not on the **home tenant**.
>
> - A **resource tenant** is where the app is expected to exist, and where the user is expected to create the app using Power Apps as a guest.
> - A **home tenant** is where the user's account resides and authenticates against.

Steps to grant guest access

1. In Microsoft Entra ID, select **New guest user**. More information: Quickstart: Add a guest user and send an invitation



2. If the guest user doesn't already have a license in their home tenant, assign a license to the guest user.

   - To assign guest users from admin.microsoft.com, go to Add users and assign licenses at the same time.
   - To assign guest users from portal.azure.com, go to Assign or remove licenses.

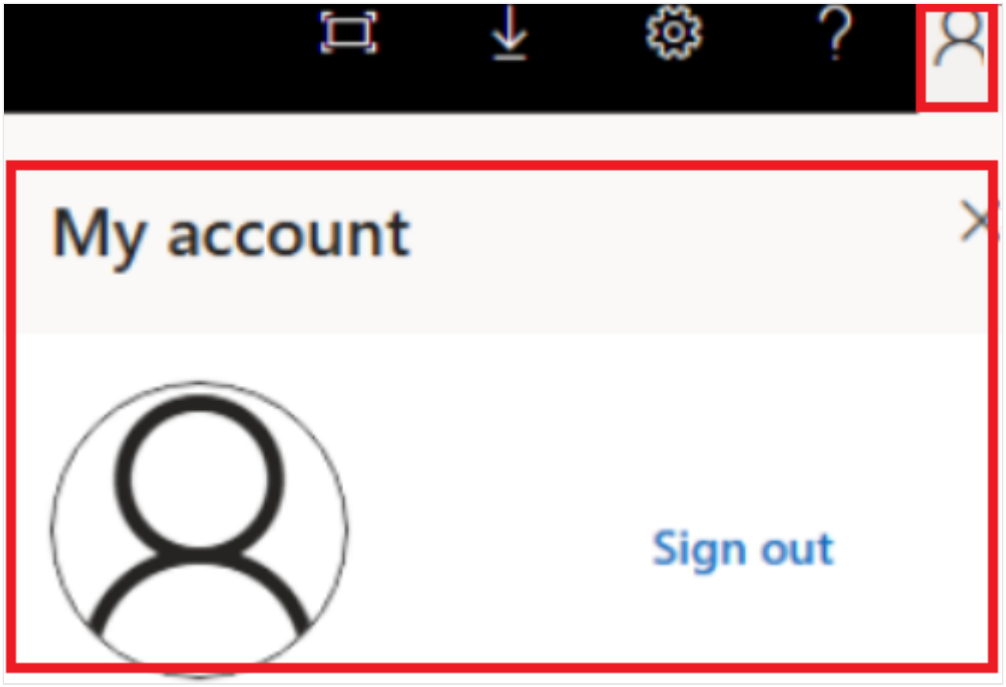
   > **Important**
   >
   > You might need to disable the Microsoft 365 admin center preview to assign a license to a guest.

3. Share an app from Power Apps

Considerations and limitations for guest access

- Users accessing web experiences in different Microsoft Entra tenants must access Power Apps in a standalone browser session (different browser or InPrivate browser session). The standalone browser session insures that Power Apps picks up the correct Azure B2B user identity for the app being accessed.
- Power Apps guest access uses Azure B2B.
- Power Apps Mobile doesn't support authentication using Microsoft Entra direct federation. More information: Sign in using Power Apps Mobile
- Power Apps per app plans are scoped to apps in a specific environment, so they can't be recognized across tenants.
- Power Apps included with Office and Power Apps per user plans have the following characteristics:

- In the Azure public cloud, they're recognized across tenants in guest scenarios because they aren't bound to a specific environment.
- In Azure national or sovereign clouds, they're recognized across tenants in guest scenarios. For more information, see National clouds, Azure geographies
- Licenses aren't recognized across tenants in difference Azure clouds.
- Not all connectors create connections in the resource tenant by default.
- A user that signs in to an app using Azure B2B, doesn't see their identity information in **My account**.



Frequently asked questions

What's the difference between canvas app guest access and Power Pages?

With canvas apps, you can build an app tailored to digitizing business processes, without writing code in a traditional programming language such as C#. Guest access for canvas apps enables teams of individuals made up of different organizations participating in a common business process to access the same app resources that might be integrated with a wide variety of Microsoft and partner sources. More information: Overview of canvas-app connectors for Power Apps

Power Pages provide you with the ability to build low-code, responsive websites that allow external users to interact with the data stored in Dataverse. With Power Pages, organizations

can create websites and share them with users external to their organization either anonymously or through the sign-in provider of their choice, such as LinkedIn, Microsoft, or other commercial sign-in provider.

The following table outlines a few core capability differences between Power Pages and canvas apps.

Expand table

Guest access in	Interface	Authentication	Accessible data sources
Power Pages	Browser-only experience	Allows anonymous and authenticated access	Dataverse
Canvas apps	Browser and mobile apps	Requires authentication via Microsoft Entra ID	Any of approximately 150 out-of-the-box connectors and any custom connector

Can guests access customized forms in SharePoint?

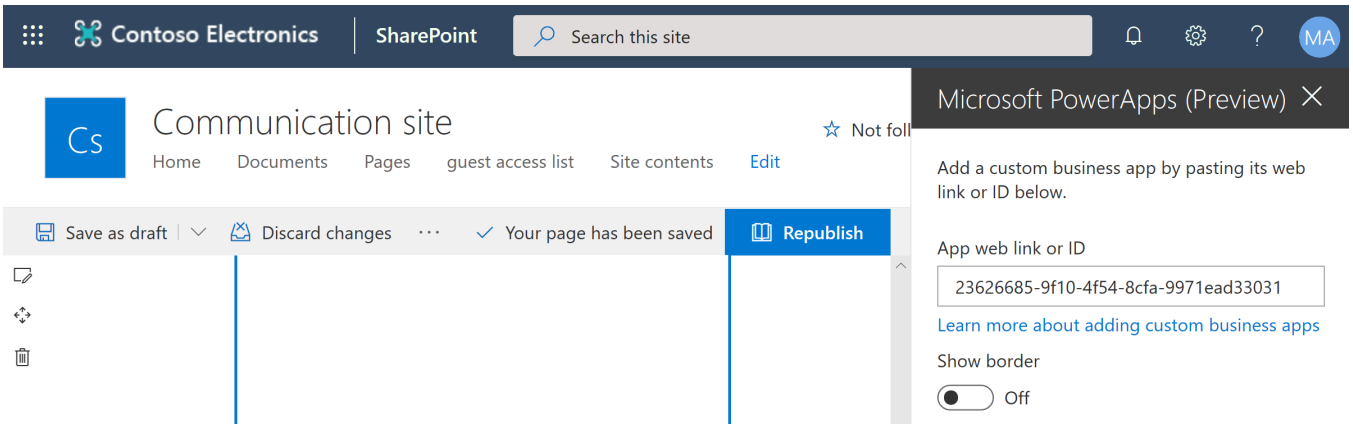
See What license must be assigned to my guest so they can run an app shared with them?

Why is a guest who accesses a customized form in SharePoint prompted for a trial?

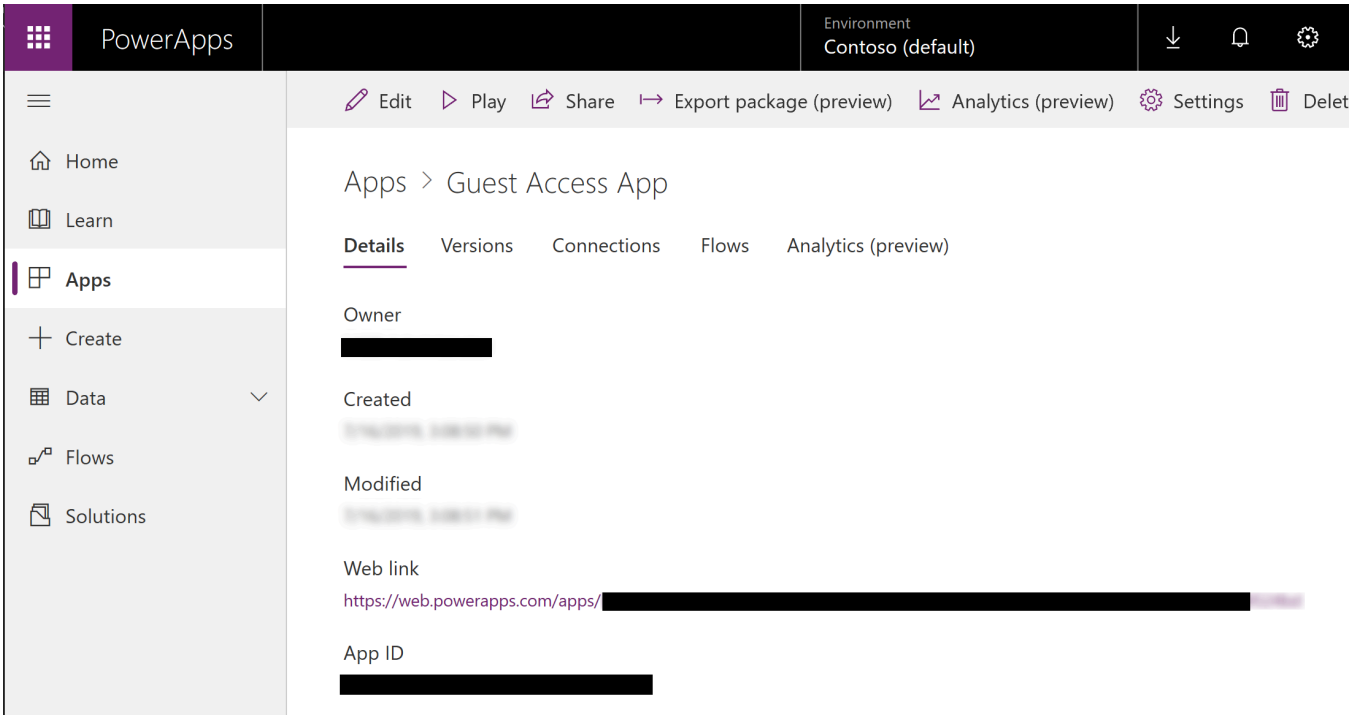
If the custom form uses a premium connector, a guest must have a Power Apps license to access the custom form. If the custom form only uses standard connectors, your tenant must allow Microsoft Power Platform internal consent plans to be assigned to users. For more details about Power Platform internal consent plans, read block trial license commands.

Can guests access apps embedded in SharePoint?

Yes. However, access to canvas standalone apps requires a user license with Power Apps user rights that matches the capability of the app, including embedded apps. When embedding a canvas app in SharePoint with the Power Apps embed control, enter the app ID in the **App web link or ID** box.



When embedding a canvas app in SharePoint via the iFrame HTML tag, reference the app by using the full web URL. To find the URL, sign in to Power Apps, select an app, and then select the **Details** tab. The URL is displayed under **Web link**.



How is it that guests can open the app shared with them, but no data connections are created?

As is the case with nonguests, the underlying data sources accessed by the app must also be made accessible to the guest.

What license must be assigned to my guest so they can run an app shared with them?

The following table explains whether the guests can run (use) customized Microsoft Lists or SharePoint library forms, canvas apps, and model-driven apps using the referenced license.

Expand table

Plan	Customized Microsoft Lists or SharePoint library form (using nonpremium connectors)	Customized Microsoft Lists or SharePoint library (using premium connectors)	Canvas app (using nonpremium connectors)	Canvas app (using premium connectors)	Model-driven app
No license	✓	✗	✗	✗	✗
SharePoint user (without Power Apps license)	✓	✗	✗	✗	✗
Power Apps included with Office	✓	✗	✓	✗	✗
Power Apps per app plan	✓	✓	✓	✓	✓
Power Apps per user plan	✓	✓	✓	✓	✓

For more information about pricing and the capabilities of various plans, go to Microsoft Power Apps and Power Automate Licensing Guide.

In Power Apps Mobile, how does a guest see apps for their home tenant?

When users access a canvas app through a mobile device and the app is published in a non-home Microsoft Entra tenant, they can switch to a different directory.

In Power Apps Mobile, how does a guest see apps in the guest tenant?

The guest user opens the email they received when an app in the guest tenant is shared and selects **Open the app**. This view applies to both Microsoft Entra and Microsoft account users. You can also create a deep link. For more information, see Use deep links with Power Apps mobile.

Must a guest accept the Microsoft Entra guest invitation before an app can be shared with them?

No. If a guest opens an app shared with them before they accept a guest invitation, the guest is prompted to accept the invitation as part of the sign-in experience while opening the app.

In which Microsoft Entra tenant are connections created for a guest user?

Connections for an app are always made in the context of the Microsoft Entra tenant the app is associated with. For example, if an app is created in the Contoso tenant, the connections made for Contoso internal and guest users are made in the context of the Contoso tenant.

Can guests use Microsoft Graph with Power Apps?

By default, Azure B2B users have limited permission to access information from Microsoft Graph. A user's permission in Microsoft Graph determines what's returned when using connectors such as Microsoft Security Graph, Office 365 Users, Office 365 Groups, and custom connectors using Microsoft Graph APIs. For more information about Microsoft Graph permissions, see Default user permissions and Working with users in Microsoft Graph.

Which Intune policies apply to guests who are using my apps?

Intune only applies the policies of a user's home tenant. For instance, if `Lesa@Contoso.com` shares an app with `Wanda@Fabrikam.com`, Intune continues to apply Fabrikam.com policies on Wanda's device, regardless of the apps Wanda runs.

Can I disable cross tenant license recognition?

No. Authorization capabilities can and should be used to control which users access a resource. For instance, Dataverse environments can be bound to a security group that excludes Azure B2B guest users.

Why can an Azure B2B user without a license access a model driven app?

The detection of a previously licensed user no longer having a license isn't immediate. If you expect a user to lose access to an app using Dataverse, then their authorization to the app or

data should be updated. For example, the app should be unshared with the user or Dataverse security roles should be removed from the user.

Which connectors create connections in the resource tenant by default?

Users relying on Azure B2B to access an app only has implications on connectors that use Microsoft Entra ID for authentication. Some Microsoft Entra ID based connectors default to creating a connection in the resource tenant, while others default to creating a connection in the home tenant. Connectors that don't use any type of Microsoft Entra ID authentication work the same for guests and members in a tenant. The following table enumerates all connectors that do use Microsoft Entra ID authentication and default creates connections in the resource tenant. For more information on each connector as there might be restriction, see List of all Power Apps connectors

Expand table

Connector	Creates connection in resource tenant by default
Microsoft Entra	Yes
Azure Automation	Yes
Azure Container Instance	Yes
Azure Data Factory	Yes
Azure Data Lake	Yes
Azure IoT Central	Yes
Azure Kusto	Yes
Azure Log Analytics	Yes
Azure Resource Manager	Yes
Microsoft Dataverse	Yes*
Dynamics 365 AI for Sales	Yes
Microsoft Teams	Yes
Office 365 Groups	Yes
Office 365 Users	Yes
Outlook Tasks	Yes

Connector	Creates connection in resource tenant by default
Power BI	Yes
SharePoint	Yes

* When using the Microsoft Dataverse as the data source, ensure that the guest user is licensed from the same tenant where you have Dataverse data located.

See also

Edit an app
Restore an app to a previous version
Export and import an app
Delete an app

Share Excel data used by your app

Article • 12/16/2022

You can share Excel data with your app users in a cloud account, such as OneDrive.

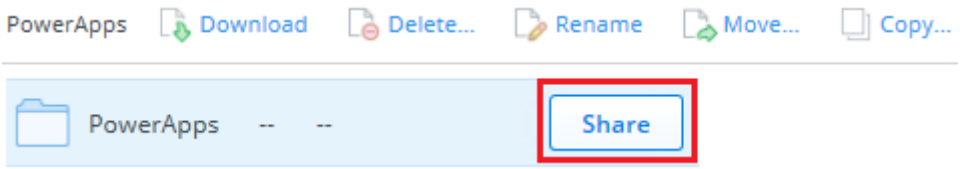
For example, you might create an app that shows the names and phone numbers of the technical-support group at your company. The information is stored in an Excel spreadsheet, which you put in a folder in Dropbox. You then share the folder with your app users so that they can see the names and phone numbers.

You must share the data so that users can run and even modify your app. Users who aren't given the sharing permissions won't see the data in the Excel file.

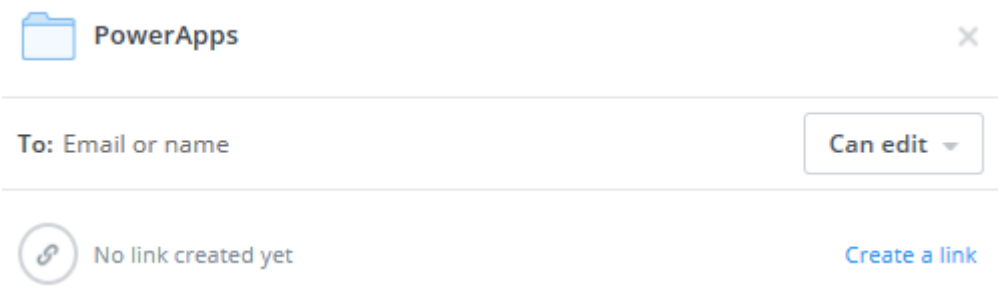
This topic shows you how to share data in an Excel spreadsheet using Dropbox, OneDrive, and Google Drive. To create an app that displays data from an Excel file, see Create an app from a set of data.

Share data in Dropbox

1. Sign in to Dropbox using the same account that you used to create a connection from Power Apps to Dropbox.
2. Select the folder that contains the Excel file, and then select **Share**:



3. In the dialog box, enter the email addresses with which your app users sign in to Dropbox.

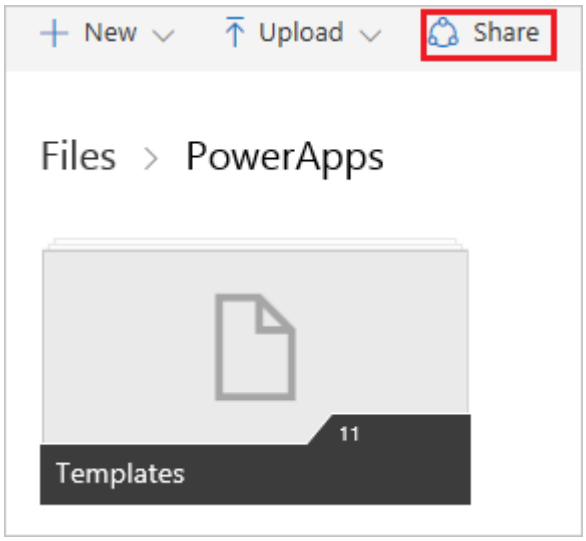


4. If your app users will add, modify, or delete data in your app, select **Can edit**. Otherwise, select **Can view**.
5. Select **Share**.

For more information, see Sharing folders on Dropbox.

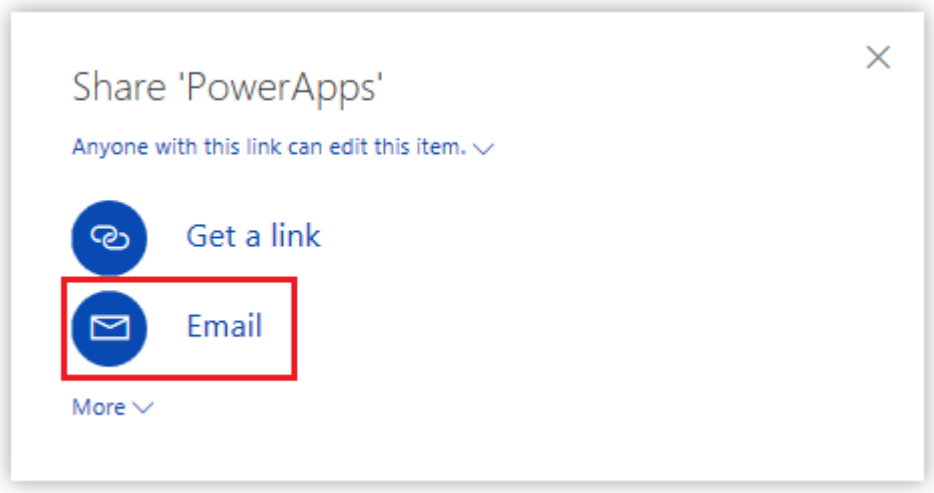
Share data in OneDrive

1. Sign in to OneDrive using the same account that you used when you created a connection from Power Apps to OneDrive.
2. Select the folder that contains the file, and then select **Share**:

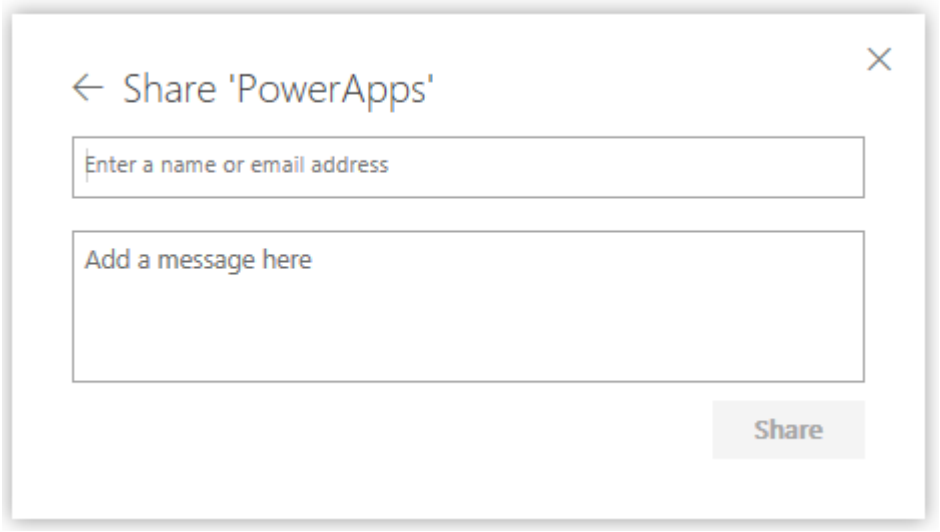


> **Note**
>
> In OneDrive for Business, share the file itself, not the folder that contains the file.

3. In the dialog box, select **Email**.



4. Specify the email addresses with which your app users sign in to OneDrive, and then select **Share**.

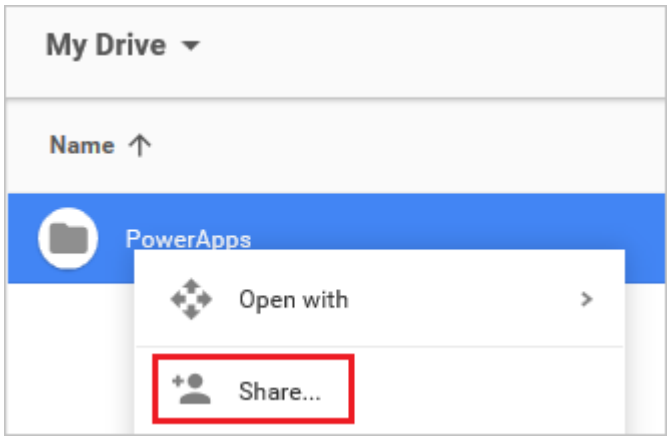


For more information, see Share OneDrive files and folders.

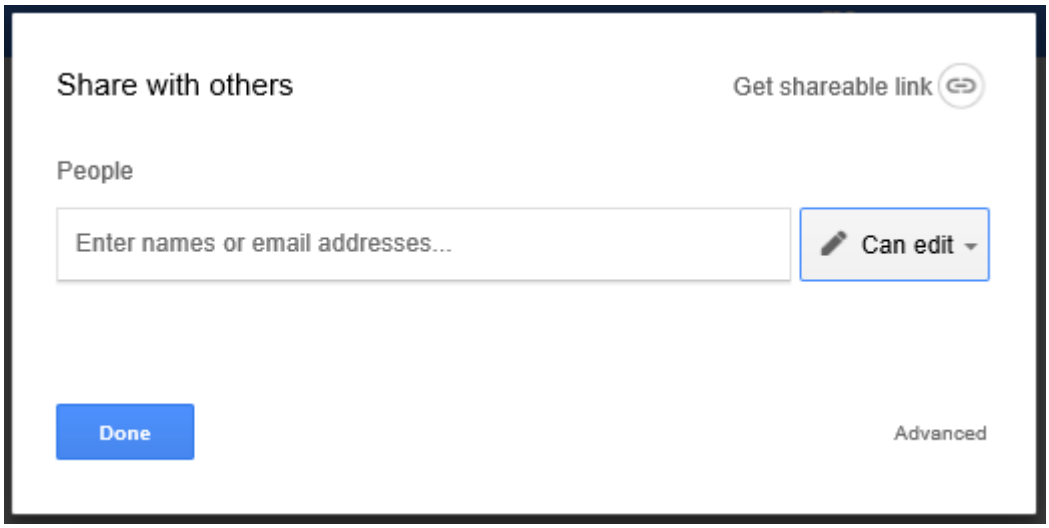
Share data in Google Drive

1. Sign in to Google Drive using the same account with which you created a connection from Power Apps to Google Drive.

2. Right-click the folder that stores your Excel file, and then select **Share**.



3. In the dialog box, enter the email addresses with which your app users sign in to Google Drive:



4. If your app users will add, modify, or delete data in your app, then select **Can edit** in the list of permissions. Otherwise, select **Can view**.

5. Select **Done**.

For more information, see Share Google Drive files and folders.

Known limitations

For information about how to share Excel data within your organization, review these limitations.

Share resources used by canvas apps

07/08/2025

Before you share a canvas app, consider the types of resources on which it relies, such as one or more of the following:

- Tables in Microsoft Dataverse.

  For information about giving users access to this data, see Manage table permissions.

- A connection to a data source.
- An on-premises data gateway.
- A custom connector.
- An Excel workbook or other service.
- A flow.

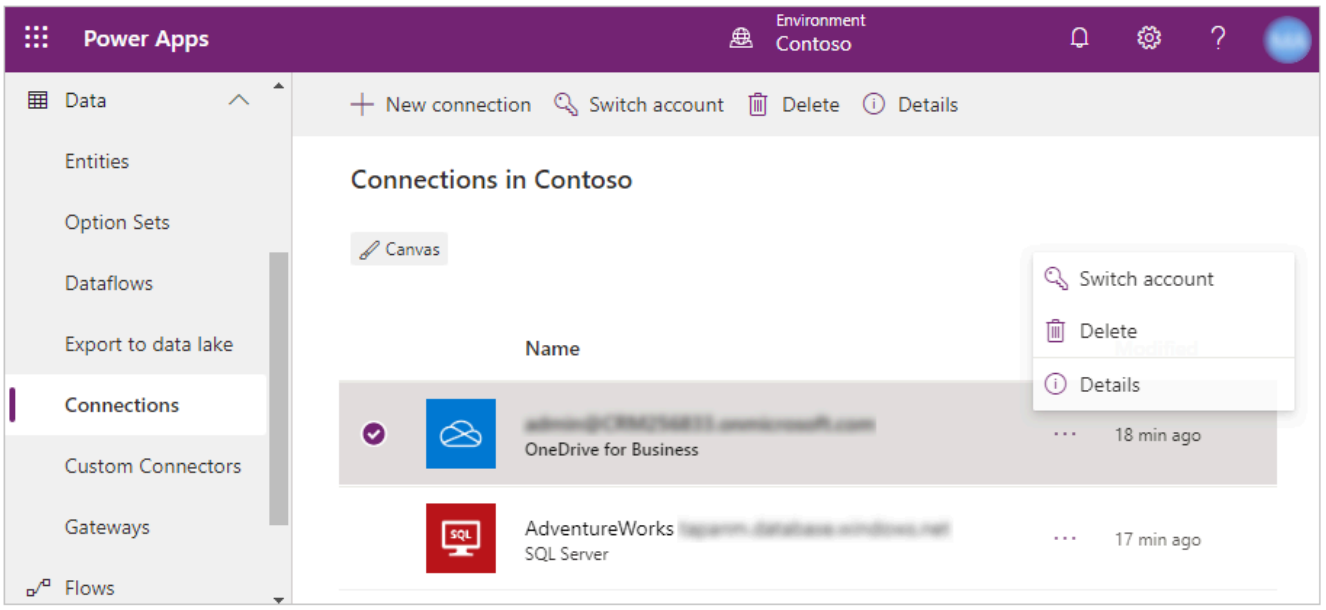
Some of these resources are shared automatically when you share the app. Other resources require you or the people with whom you share the app to take extra steps so that the app works as you expect.

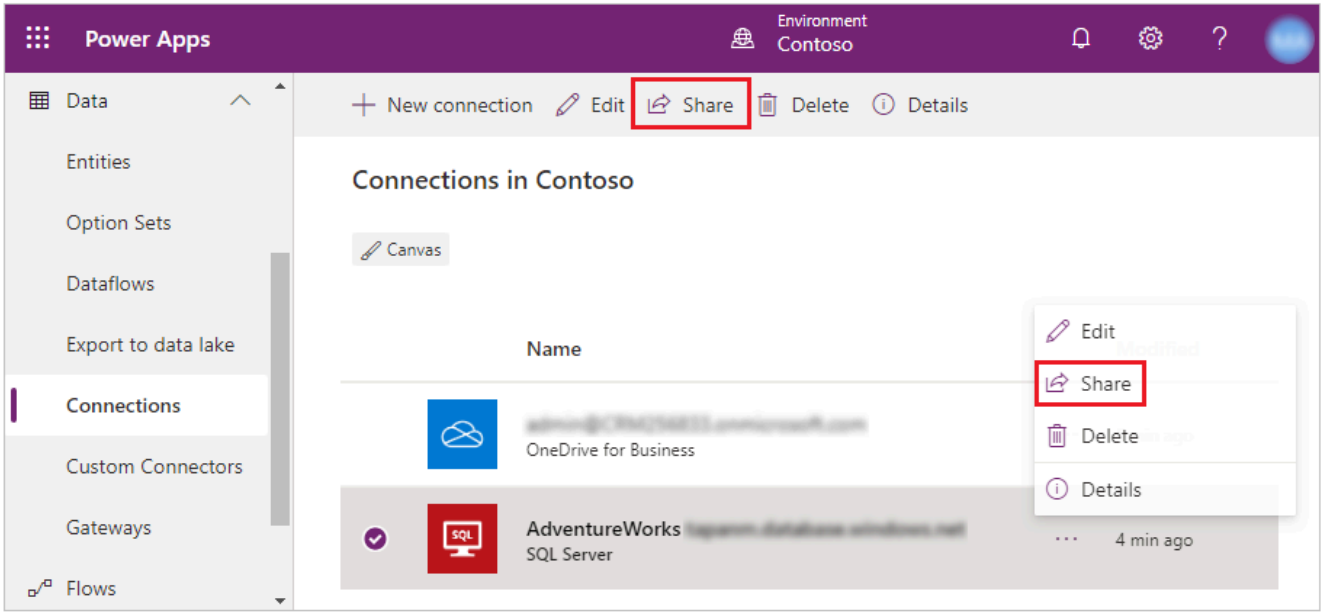
You can also share your connections, custom connectors and on-premises data gateway with your entire organization.

Connections

Some connections (such as SQL Server with SQL or Windows authentication) are implicitly shared with the app when you share the app with other users. Other connections require users to create their own connections and explictly grant security privileges (such as security roles for the Dataverse, OneDrive for Business, SQL Server with Microsoft Entra authentication).

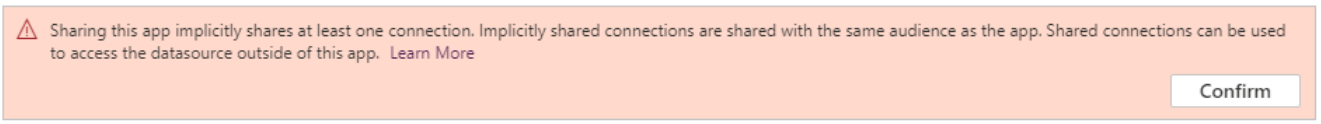
You can determine whether a connection is automatically shared as part of the app when you share the app with other users; allowing you to update sharing permissions. To do this, go to make.powerapps.com and select **Data** -> **Connections** from left navigation. Then select the required connection. If the **Share** button appears on top navigation or if the **Share** option displays when you select *More Commands* (...), the selected connection can be shared with other users.





Implicit sharing

When you share an app that uses a connection that can be shared, the app connection is **implicitly shared** along with the app. For example, following message appears when you go to make.powerapps.com, select **Apps**, choose an app that uses such connection, select *More Commands* (...) and then select **Share**:



If you select **Confirm** and share the chosen app with other users, the app connection is implicitly shared with those users along with the app.

> **Note**
>
> To ensure the app works properly, the user who is sharing the app needs to have sharing permission for all connections used by the app. Without these permissions, the user won't be able to access or use the connections correctly. This applies to all app-sharing activities, including app sharing, app publishing, and solution import.

On-premises data gateways

If you create and share an app that includes data from an on-premises source, the on-premises data gateway itself and certain types of connections to that gateway will be shared automatically. For any connection that isn't shared automatically, you can share it manually (as the previous section shows) or let the app prompt users to create their own connections. To show the connection or connections with which a gateway has been configured:

1. Open powerapps.com, click or tap **Manage** in the left navigation bar, and then click or tap **Gateways**.
2. Click or tap a gateway, and then click or tap the **Connections** tab.

> **Note**
>
> If you share one or more connections manually, you might need to reshare them under these circumstances:

- You add an on-premises data gateway to an app that you've already shared.
- You change the set of people or groups with whom you've shared an app that has an on-premises data gateway.

Custom connectors

When you share an app that uses a custom connector, it is automatically shared, but users must create their own connections to it.

On powerapps.com, you can view or update permissions for a custom connector. In the left navigation bar, click or tap **Manage**, click or tap **Connections**, and then click or tap **New connection** (in the upper-right corner). Click or tap **Custom**, and then click or tap a custom connector to display details about it.

Excel workbooks

If a shared app uses data to which not all users have access (such as an Excel workbook in a cloud-storage account), share the data.

Flows

If you share an app that includes a flow, users who run the app will be prompted to confirm or update any connections on which the flow relies. In addition, only the person who created the flow can customize its parameters. For example, you can create a flow that sends mail to an address that you specify, but other users can't change that address.

Tutorial: Embed a Power Apps visual in a Power BI report

APPLIES TO: Power BI Desktop Power BI service

In this tutorial, you use the Power Apps visual to create a new app that is embedded in a sample Power BI report. This app interacts with other visuals in that report.

If you don't have a Power Apps subscription, create a free account before you begin.

In this tutorial, you learn how to:

- Add a Power Apps visual to a Power BI report
- Work in Power Apps to create a new app that uses data from the Power BI report
- View and interact with the Power Apps visual in the report

Prerequisites

- Google Chrome or Microsoft Edge browser
- A Power BI subscription, with the Opportunity Analysis Sample installed
- An understanding of how to create apps in Power Apps and how to edit Power BI reports

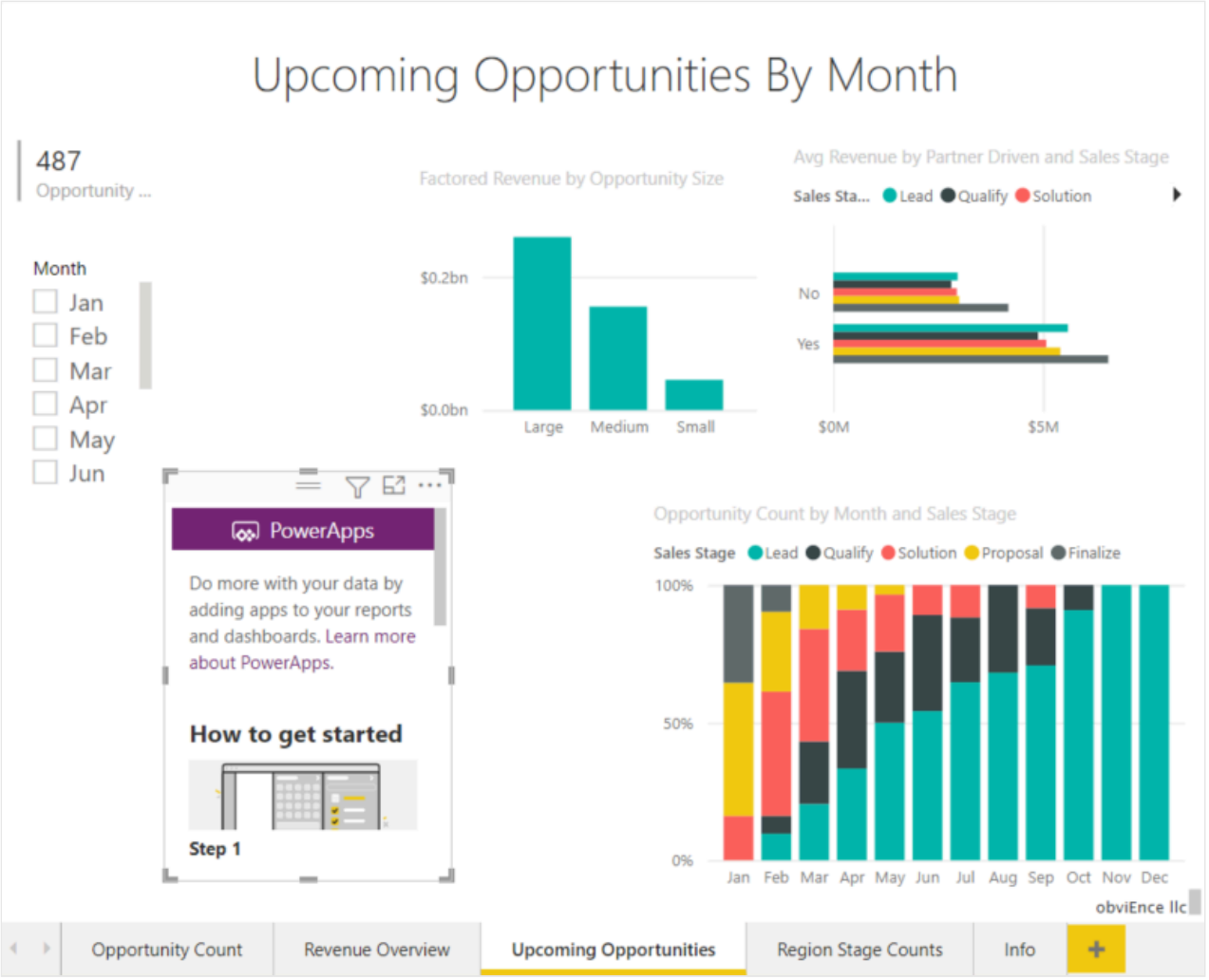
Create a new app

When you add the Power Apps visual to your report, it launches Power Apps Studio with a live data connection between Power Apps and Power BI.

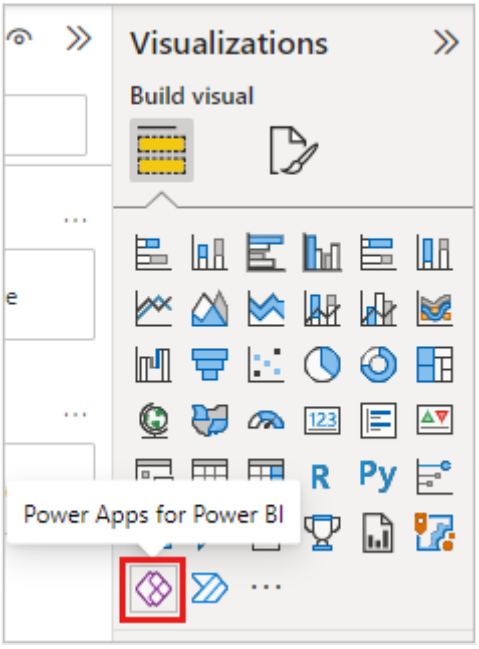
1. Open the Opportunity Analysis sample report and select the *Upcoming Opportunities* page.

2. Open the report in Editing mode by selecting **Edit** from the Options bar.

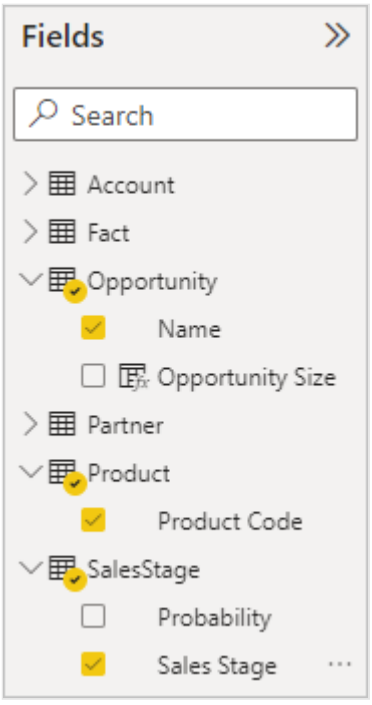
3. Move and resize some of the report tiles to make space for the new visual.



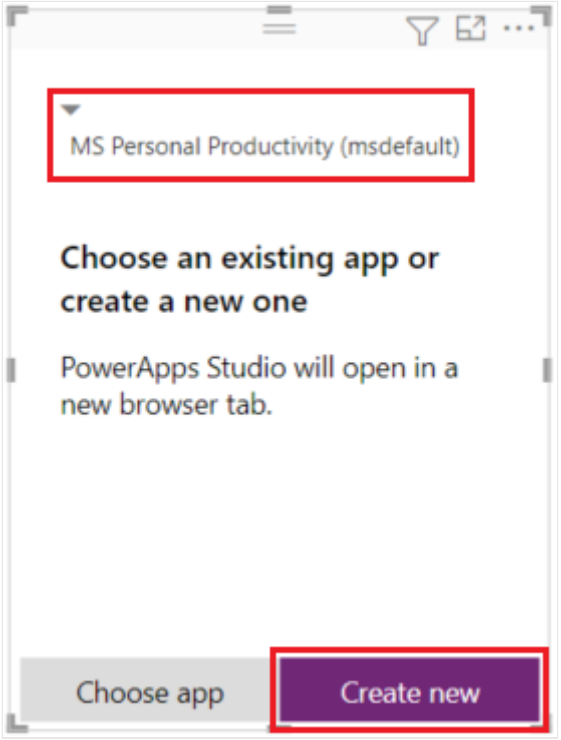
4. From the Visualizations pane, select the Power Apps icon, then resize the visual to fit the space you made.



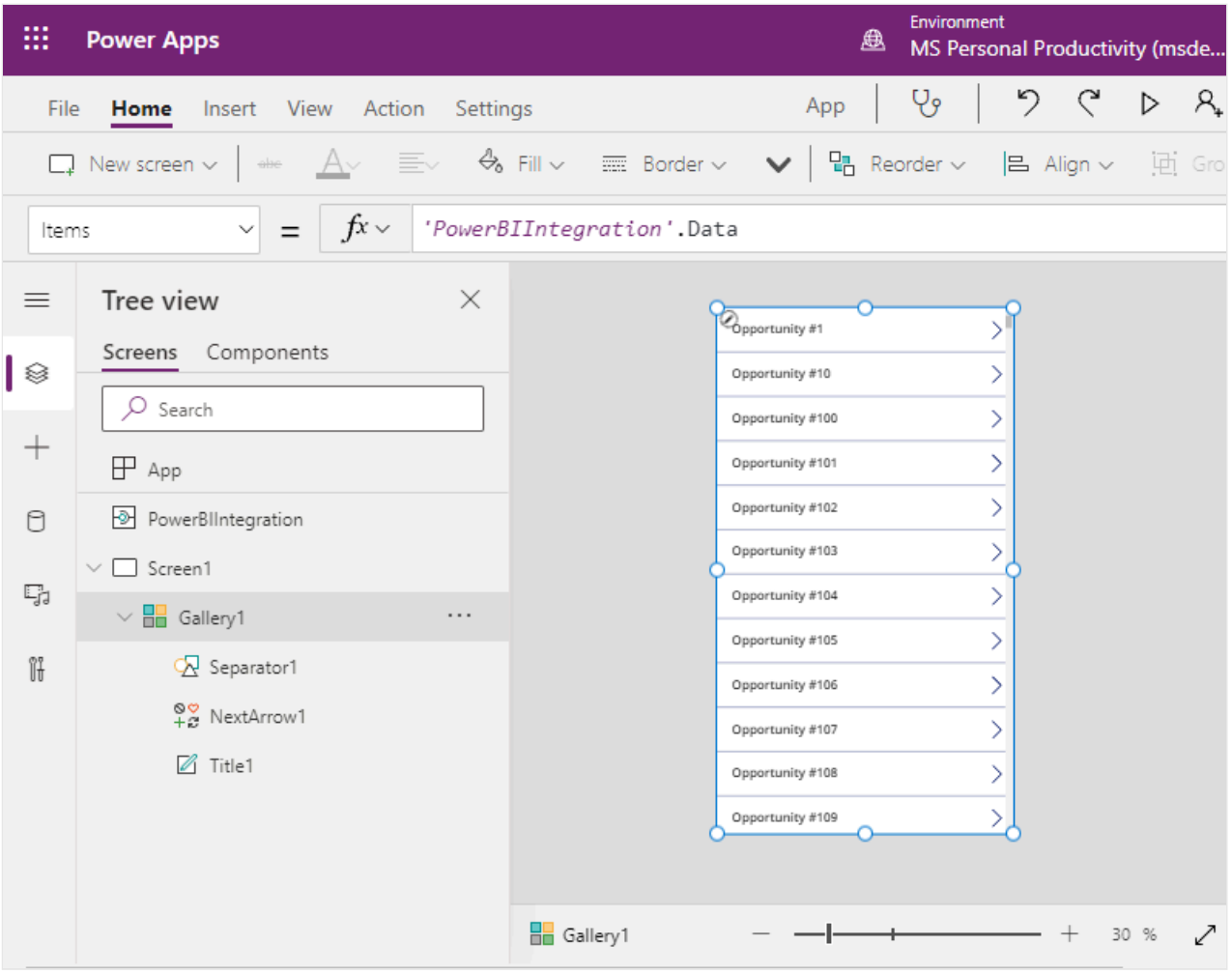
5. In the **Fields** pane, select **Opportunity** > **Name**, **Product** > **Product Code**, and **SalesStage** > **Sales Stage**.



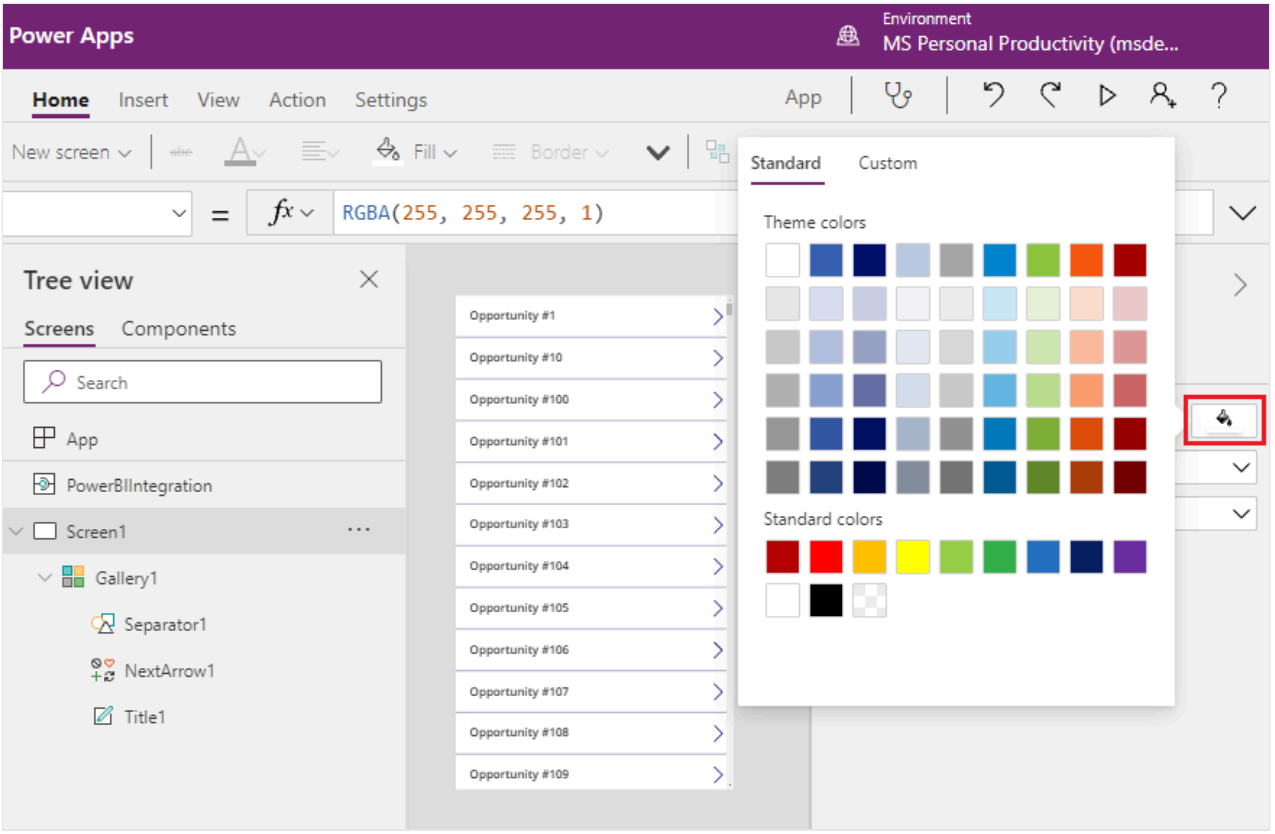
6. On the Power Apps visual, select the Power Apps environment where you want to create the app, then select **Create new**.



In Power Apps Studio, you see that a basic app is created, with a *gallery* that shows one of the fields you selected in Power BI.

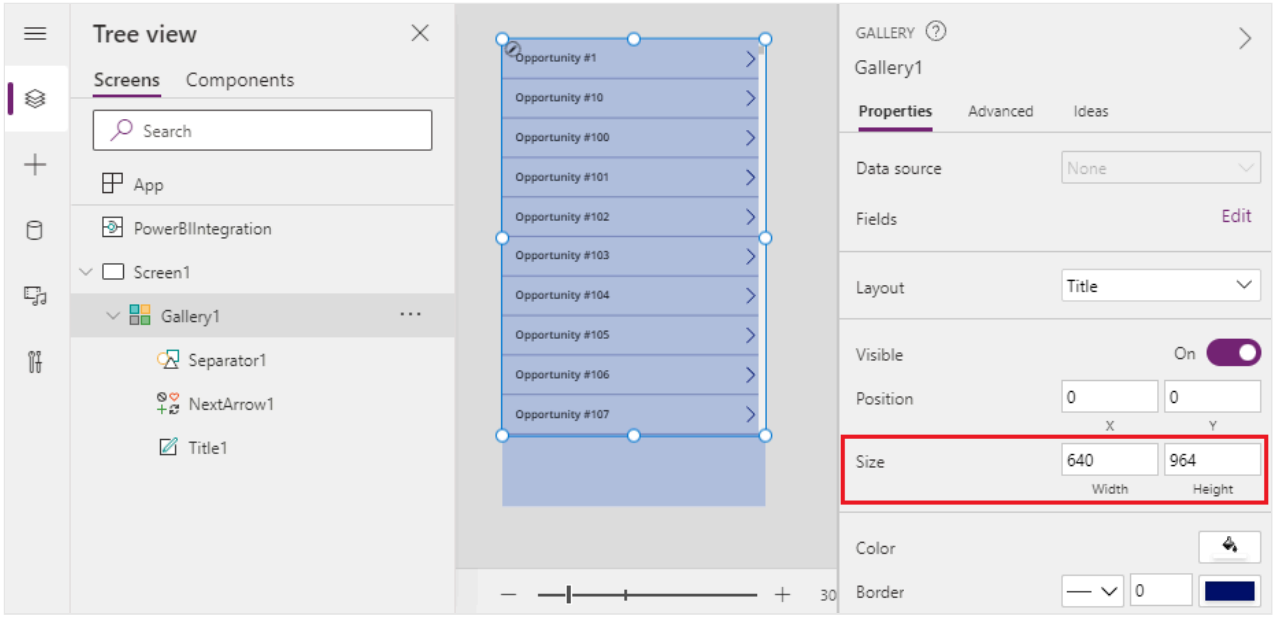


7. In the left pane, select **Screen1**, then set the screen's **Fill** property to "LightBlue" (so it shows up better in the report).

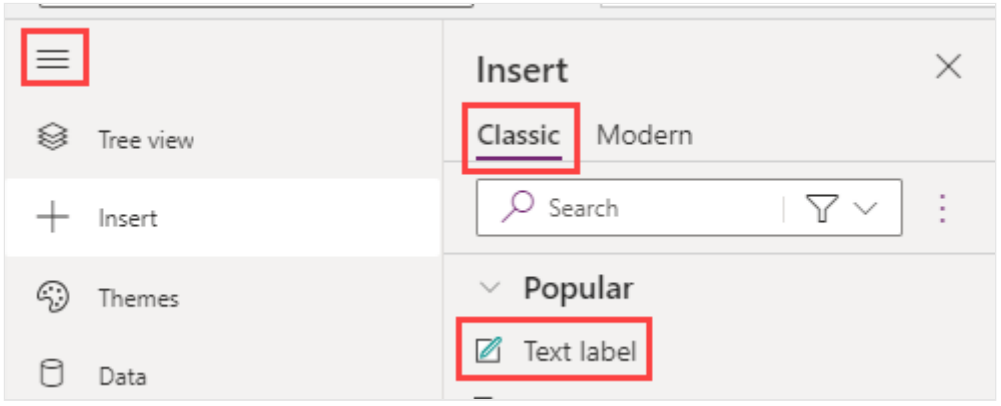


8. Select **Gallery1** and make some room for a label control by changing **Properties** > **Size** > **Height** to 964.

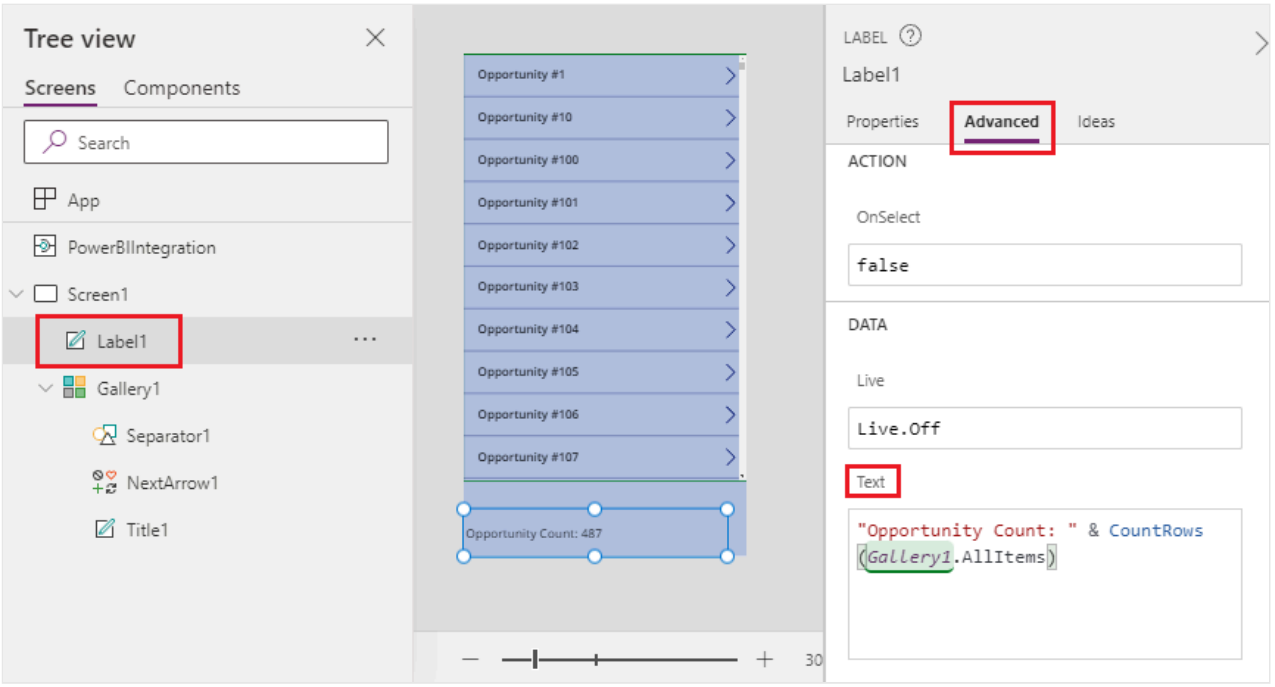
> **Note**
>
> If you can't edit the **Height** field, try adjusting the size of the visual on the canvas first. Select the visual and drag one of the handles. Now edit the **Height** field.



9. Expand the left navigation menu, choose **+ Insert**, and select **Classic** > **Text label** from the list of controls.



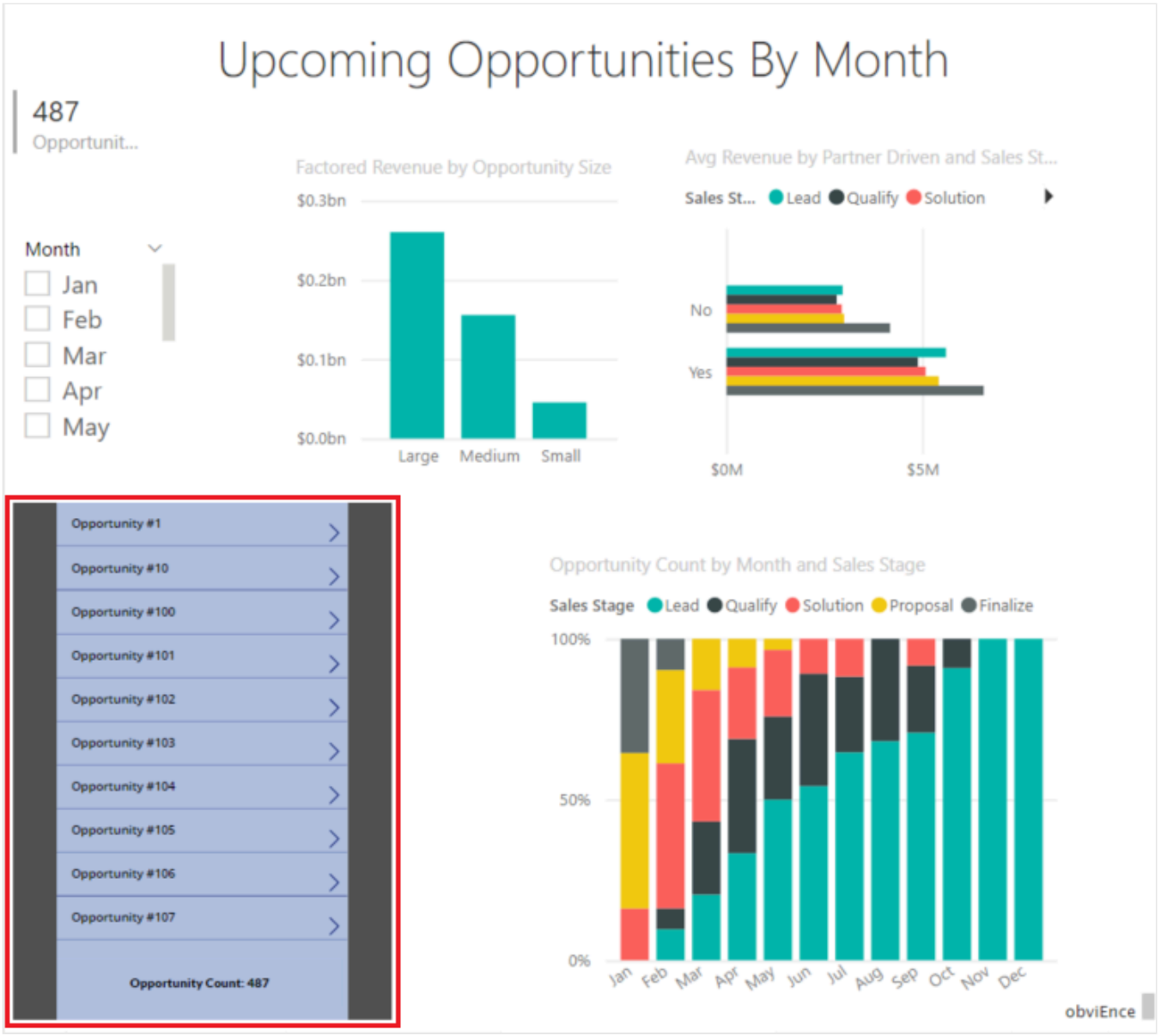
10. Drag the label to the bottom of your visual. Set the **Text** property to `"Opportunity Count: " & CountRows(Gallery1.AllItems)`. The visual on the canvas now shows a total of 487 opportunities in the dataset.



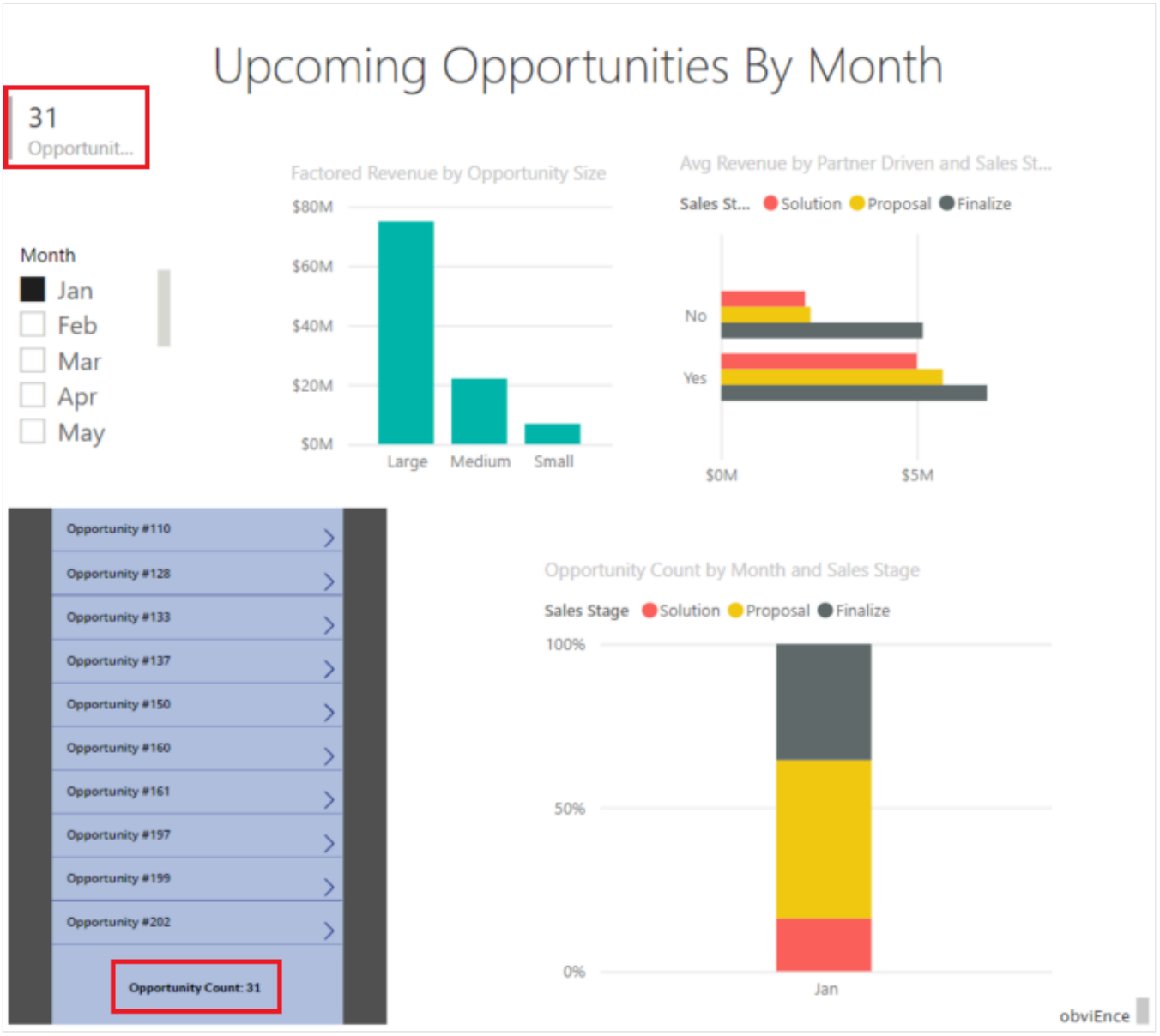
11. Select the **Save** icon from the upper right corner. Save the app with the name "Opportunities App".

View the app in the report

The app is now available in the Power BI report, and it interacts with other visuals because it shares the same data source.



In the Power BI report, select **Jan** in the slicer. This filters the whole report, including the data in the Power Apps visual.



Notice that the updated opportunity count in the Power App visual matches the count in the upper left of the report. You can select other items in the report, and the data in the app updates.

Clean up resources

If you don't want to save the Opportunity Analysis sample with the Power Apps visual, delete the dashboard, report, and semantic model.

Considerations and limitations

For troubleshooting information, see Power Apps visual for Power BI.

Related content

- Q&A visual
- Tutorial: Embed a Power Apps visual in a Power BI report

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 12/01/2025

Change app name and icon for canvas apps

Article • 12/16/2022

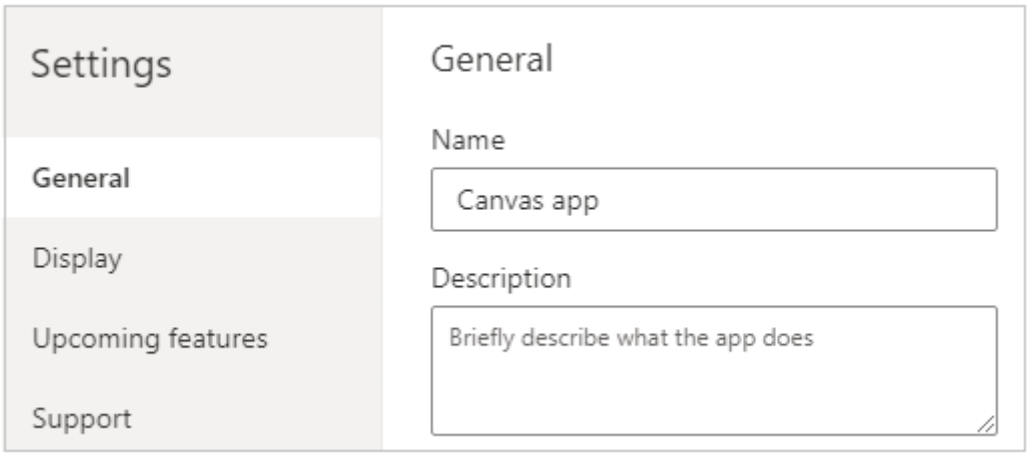
Customize a canvas app by changing its name and icon.

Prerequisites

1. Create an app, or open one for editing.
2. Select **Settings**.

Rename an app

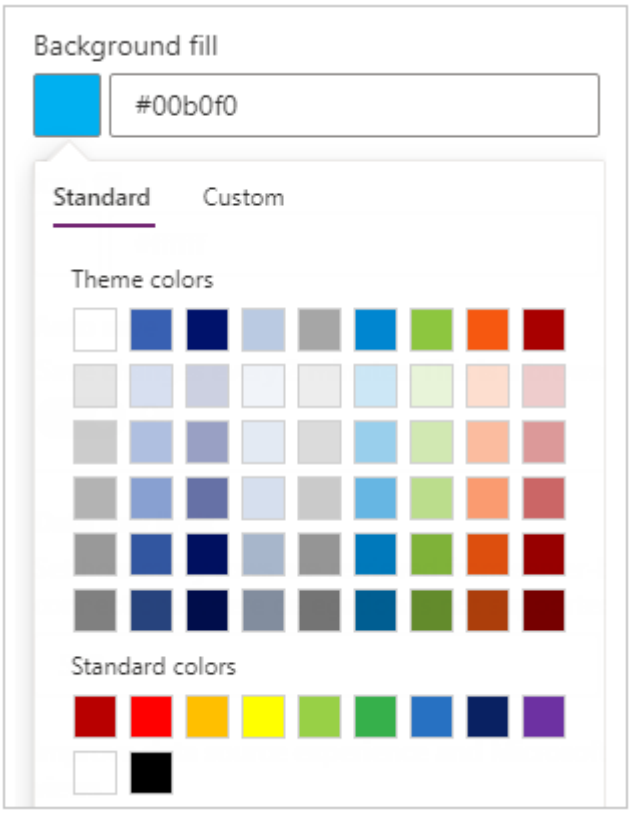
Under **General**, update the app name inside the **Name** text box.



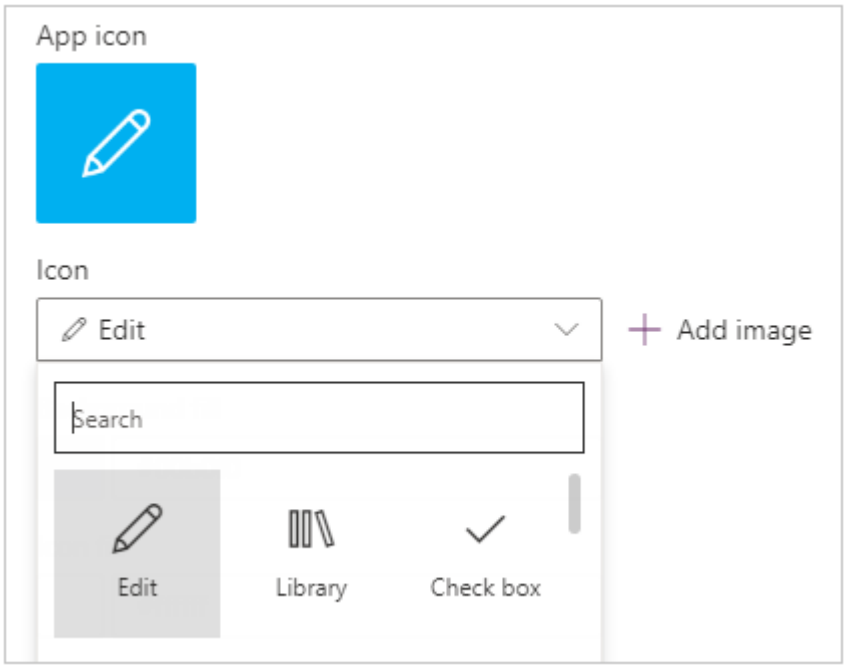
Change an app tile

Follow one or more of these steps:

- Select a different background color from the list of options.



- Select an icon from the list of default icons.



- Select **Browse file**, select the image that you want to use, and then select **Open**.

> **Note**
>
> For best results, use a square-shaped icon.

Next step

On the **File** menu, select **Save** to republish your app with the new settings.

More resources

- Share an app
- Delete an app
- Restore an app to a previous version

Edit a canvas app

Article • 09/07/2023

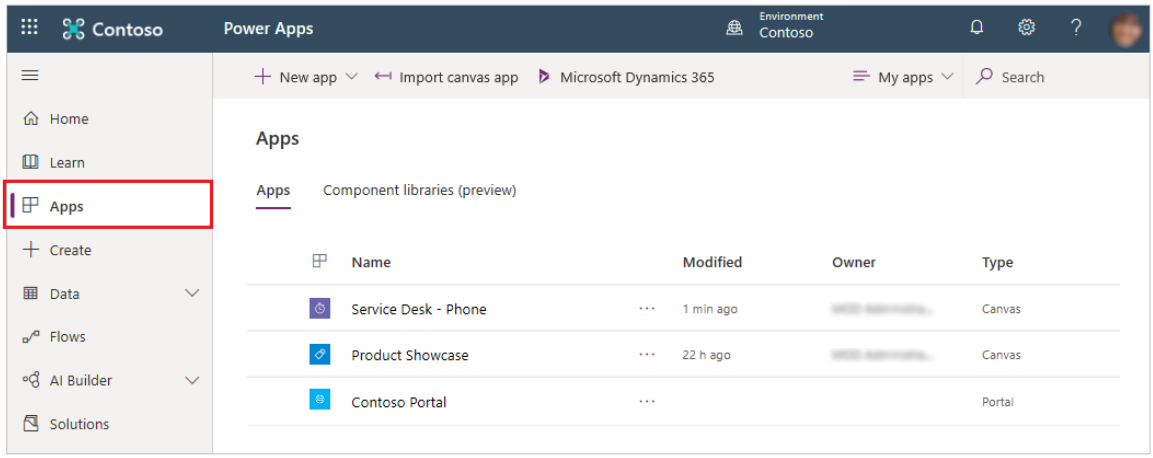
Edit any canvas app that you created, or for which you have **Co-owner** permission. You can edit an app in Power Apps Studio.

If you try to edit an app that's open for editing elsewhere, a message tells you whether you already have it open or another user does.

Edit an app

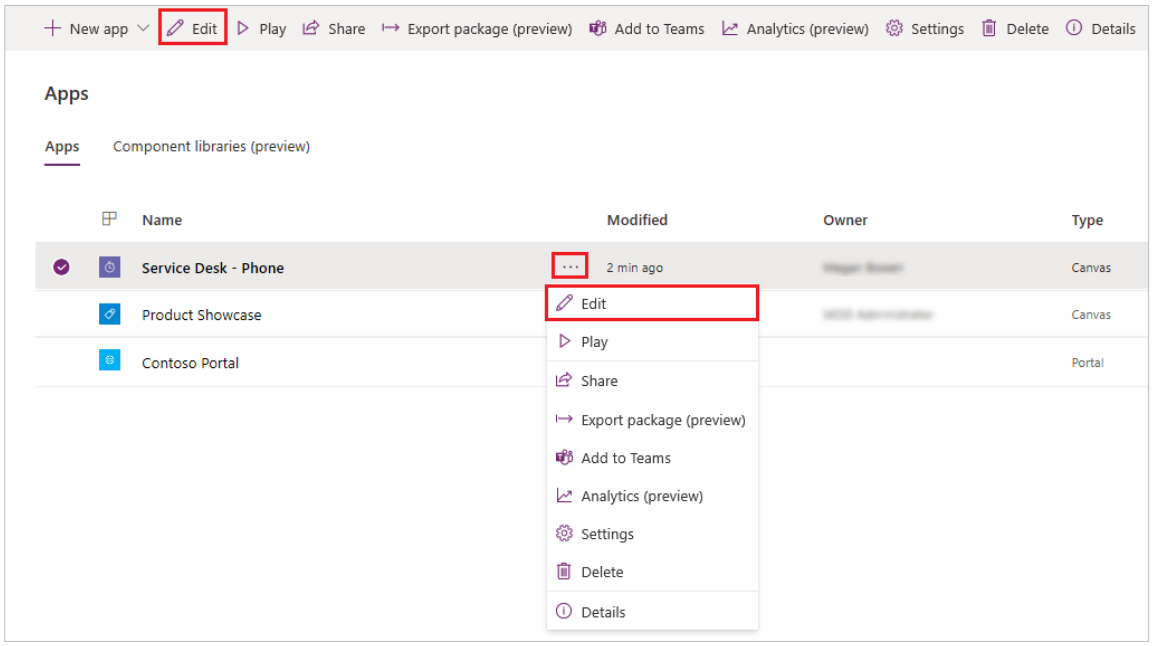
1. Sign in to Power Apps.

2. Select **Apps** in the left navigation pane.

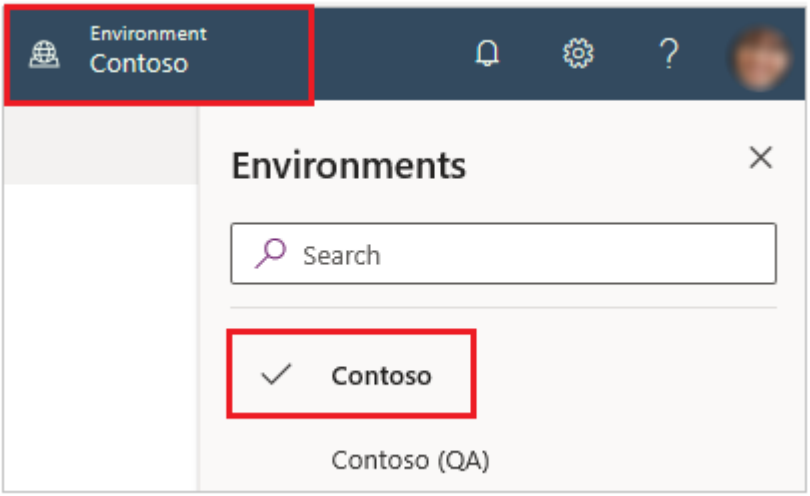


3. Select an app.

4. Select **Edit** from top menu. You can also use "**...**" (More Commands) for the app and then select **Edit** from the drop-down menu.



If you don't see the app that you want to edit, verify that you've selected the correct environment.



Collaborate on an app

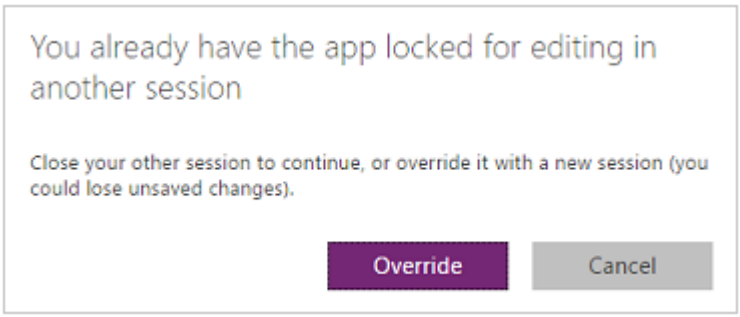
An app can be shared with others. Any **Co-owner** for an app can edit it. Consider the following scenarios when collaborating over an app.

Edit an app already being edited

Only one user can edit an app at a time. For more information, see Find out who's working on the same app.

Edit an app across multiple sessions

Consider that you already have an app open for editing. And then, you try to open the app for editing on another device, or in another browser window. In that case, you'll see the following message.



You can override the previous session, but you might lose any changes that you haven't saved.

Next Steps

Learn more about how to add a screen, a control, or a data connection.

See also

- Restore an app to a previous version
- Export and import an app

Overview of exporting and importing canvas apps

Article • 05/17/2025

In this article, you'll learn about the different options available to export and import canvas apps in the form of singular app file, app package, and considerations from the application lifecycle management (ALM) perspective. To effectively manage ALM, it's recommended to use solutions. Canvas apps packages don't support ALM and should only be used for basic import and export capabilities when Dataverse isn't accessible.

You can export and import canvas apps using one of the following options:

- Export and import canvas apps as app packages
- Export and import canvas apps as part of solutions
- Export and import a canvas app as a single app file

> **Important**
>
> - Canvas app packages can't be used with Dataverse solution packages because of the package incompatibility.
> - Canvas apps that have Dataverse dependencies, such as flows, connection references, etc. are not supported for canvas app packages. For ALM capabilities in Microsoft Power Platform environments, use Microsoft Dataverse and solutions instead of the canvas app package export and import. More information: ALM overview
> - An App Insights instrumentation key may be present in imported canvas apps. Inspect the instrumentation key after importing an app to ensure it corresponds to the desired App Insights resource.

Export and import canvas apps as app packages

You can export and import canvas apps by using packages. This feature allows you to export an app from one environment and import it to another.

Resources included in canvas apps packages

An app can consume different resources. For example, most apps use connections. Other apps might use Power Automate, have custom connectors, or connect by using gateways to on-

premises resources. Some apps might also use Dataverse customizations

The following table explains different resource types, supportability, and import options.

Expand table

Resource type	Supported	Import options
App	Yes, for canvas apps	There are two options to import an app into an environment: • **Create new**: The app is created as a new app in the environment where the package is imported. • **Update**: The app already exists in the environment and will be updated when this package is imported.
Power Automate	Yes	There are two options to import a flow into an environment: • **Create new**: The flow is created as a new flow in the environment where the package is imported. • **Update**: The flow already exists in the environment and will be updated when this package is imported. **Note:** All resources that the flow depends on will also be included in the app package that's exported and will need to be configured when the package is imported. You can also export and import flows by using solutions. More information: Power Automate solutions
Custom connectors	No	Exporting a custom connector isn't supported. You need to re-create the custom connector on the target environment.
Connections	No	Exporting a connection isn't supported. You need to re-create connections on the target environment.
Dataverse customizations	No	Exporting Dataverse customizations as a part of a canvas app package isn't supported. You need to use Dataverse solutions instead. More information: Dataverse solutions
Gateways	No	You can't export or import gateways. You need to re-create gateways on the target environment.

Permissions

Only the **Owner** or **Co-owner** of an app can export a canvas app package. To import an app, the Environment Maker security role is required on the destination environment, either directly or through a Dataverse team that is part of the **AAD Security Group** category. Custom security roles are not currently supported.

See also

- Save and publish an app
- Edit an app
- Delete an app
- Share an app

Export and import canvas apps as a package

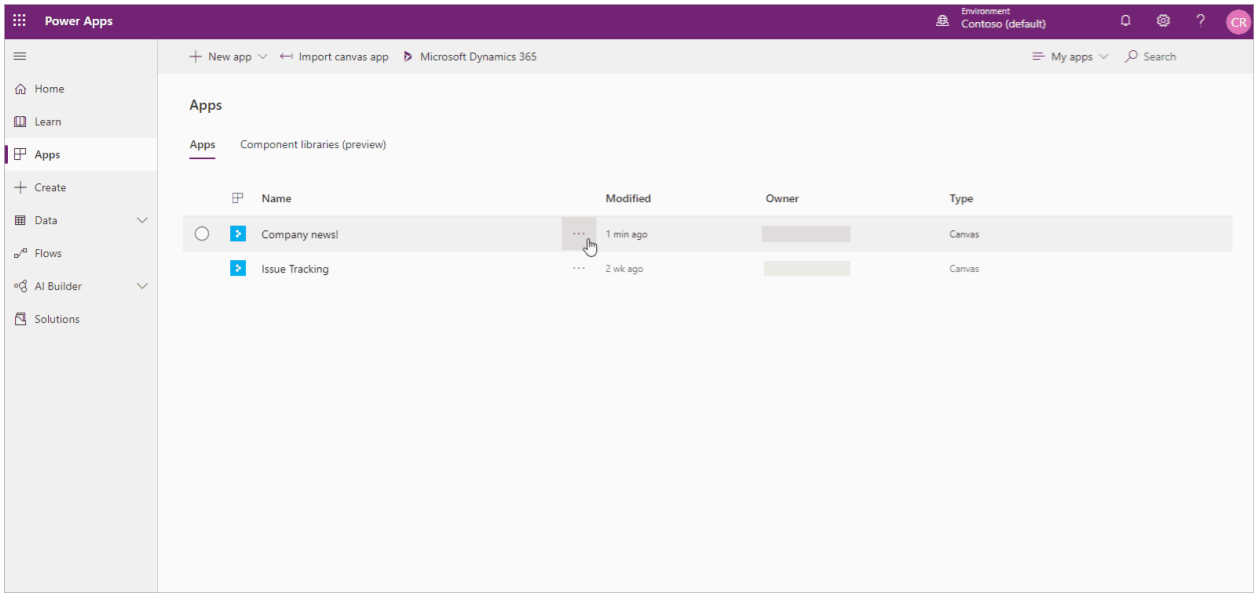
Article • 10/01/2024

This article provides a step-by-step guide for exporting and importing canvas apps. Before you begin, review the overview article to learn about the different export and import options available.

> **Important**
>
> It's recommended to use solutions for application lifecycle management (ALM) to ensure proper management and support.

Export a canvas apps package

To export a canvas apps package, select your app, and then select **Export Package**.



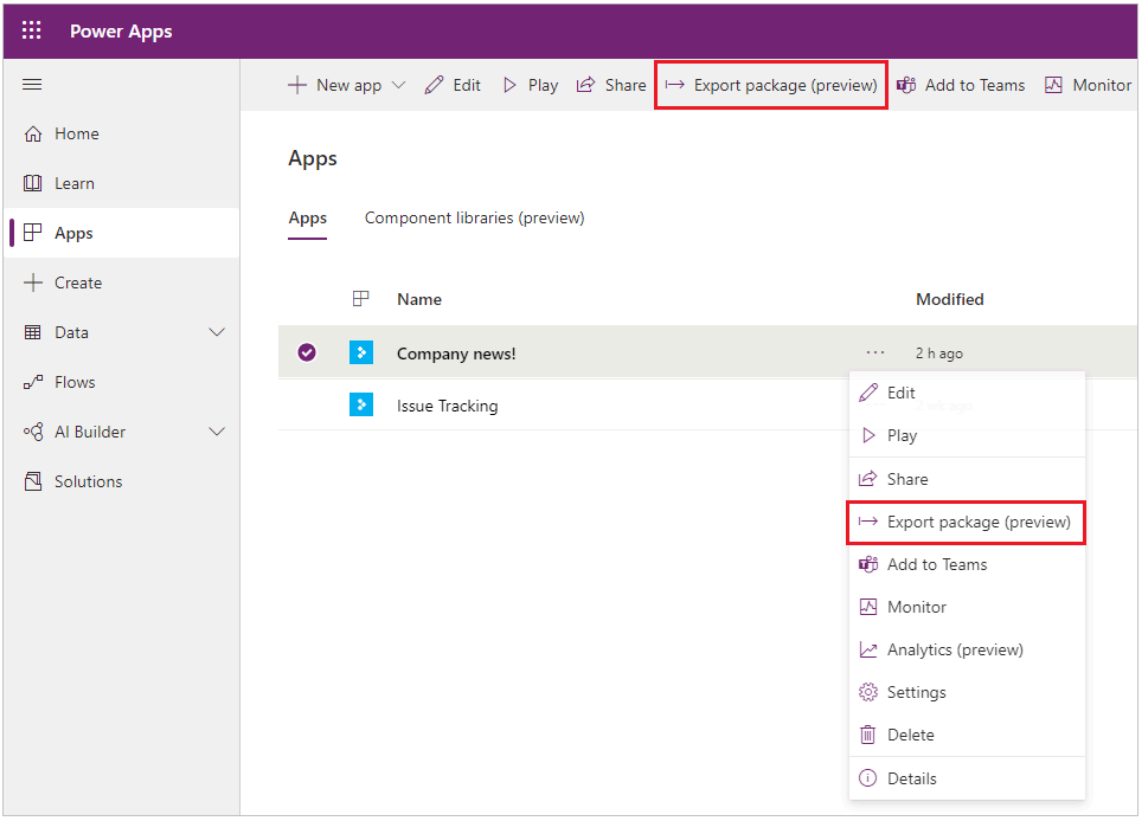
You can change the default **Import Action** for each resource used by your canvas app. By default, resources in a canvas app package, including the app itself, are set to **Update**. However, you have the option to select **Create as new** during import, which will create the resource instead of updating an existing one. The import action you select will be the default action during the app import

> **Important**

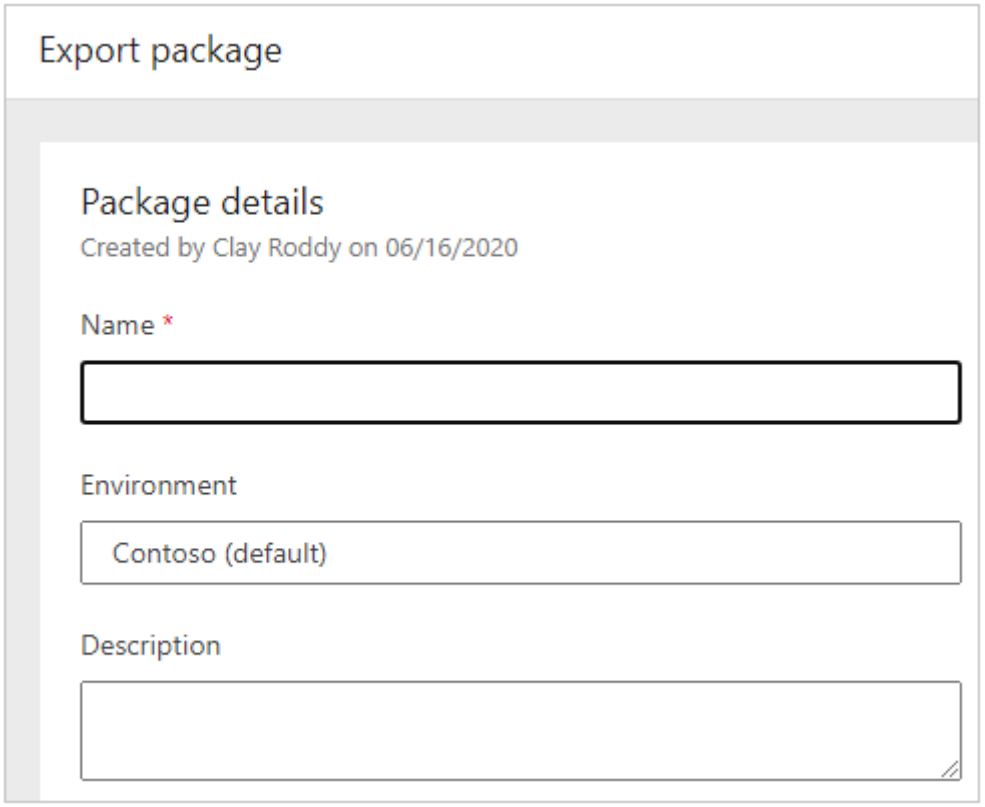
You can only import the exported package. Changes to the exported package file aren't supported.

To export a canvas app package

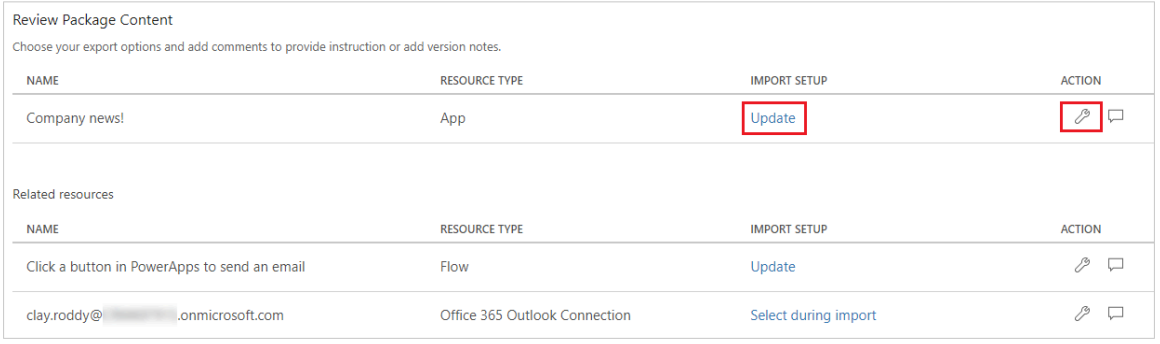
1. Sign in to Power Apps.
2. Select **Apps** from the left pane.
3. Select the app that you want to export.
4. Select **Export Package**.



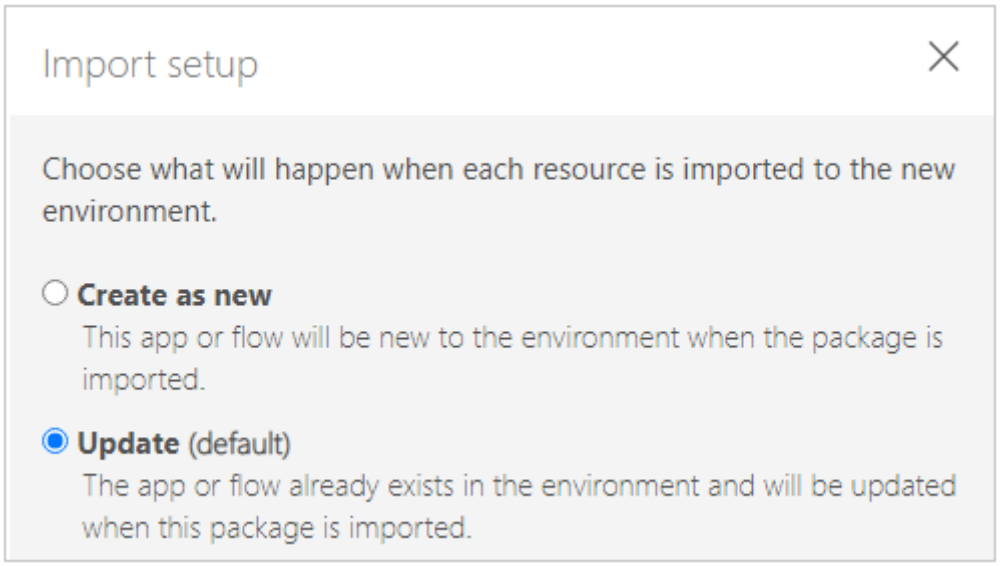
5. Enter the package **Name** and **Description**. You can also change the **Environment** name, if needed.



6. Select **Update** to choose the import action for the app package resource.

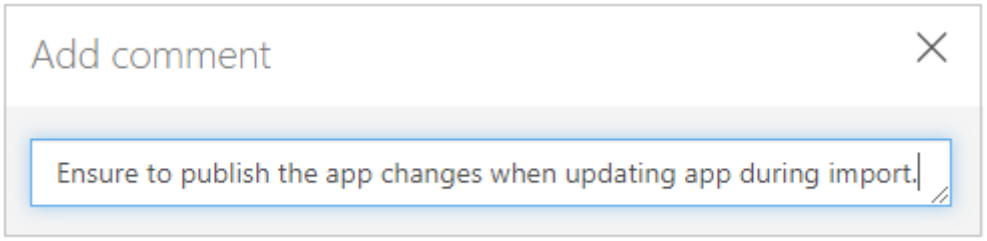


7. Select **Create as new** if the app should be created during import, or **Update** to update an existing app.



> **Note**
>
> The **Import Setup** action text changes depending on your new selection.

8. Repeat the previous step for each additional resource in the app package.
9. (Optional) Select (comment icon) to insert a comment for each resource.

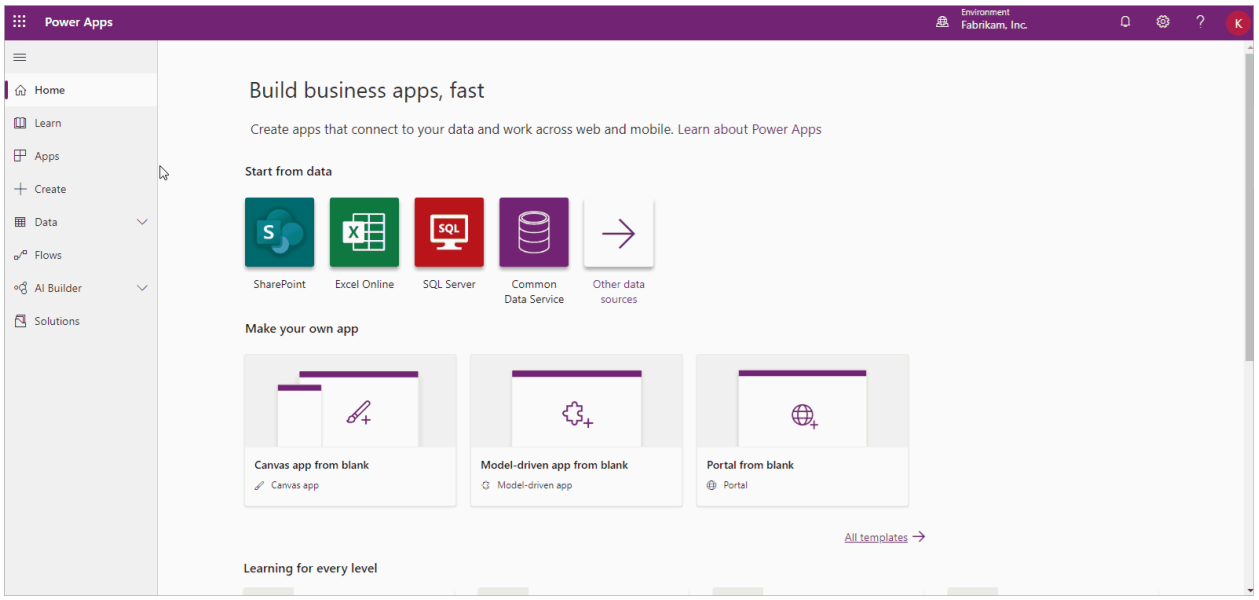


10. Select **Export** to export the package.

The package is downloaded to your browser's default downloads folder.

Importing a canvas app package

To import a canvas app package, select **Import canvas app**. Be sure to select the correct actions for each app package resource—for example, create a new instance of an app or a flow. You can also read additional information in comments that were entered while the app package was being exported.



If you're updating an existing app or resource, be sure to publish the app so users can see the changes.

Tip

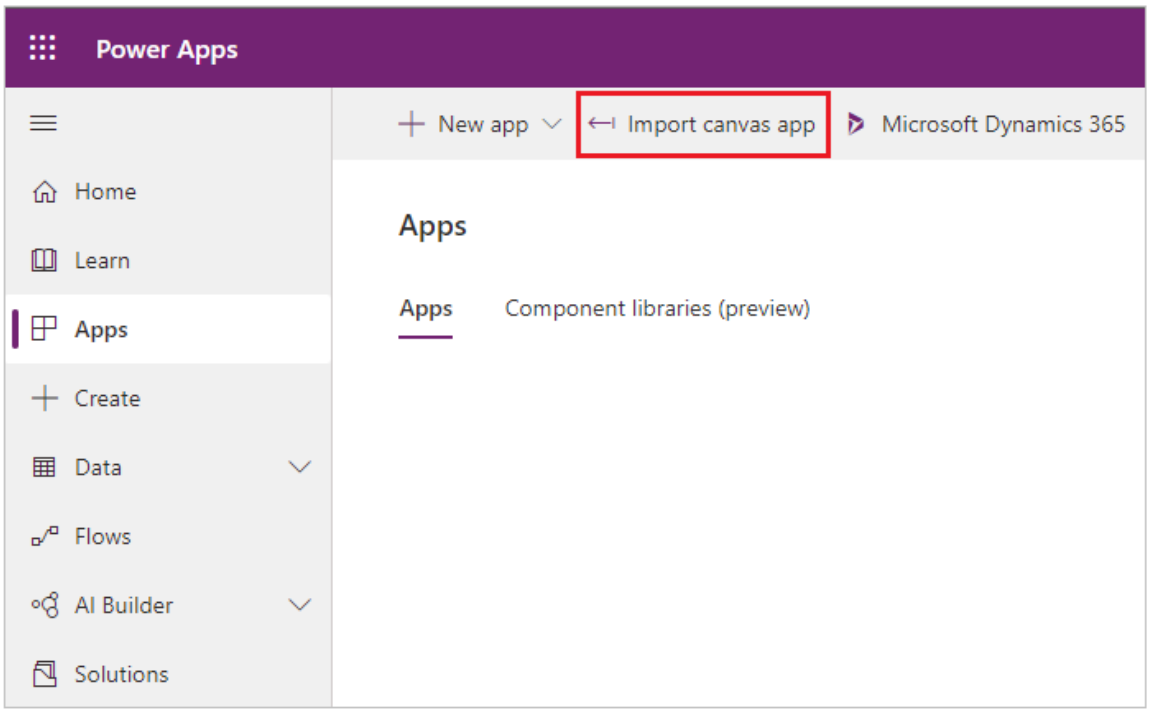
You can quickly confirm selections before importing an app package by viewing the icon changes for each resource type.

Expand table

Icon	Description
	**Create as new**: A new resource will be created.
	**Import action changed**: The import action for the resource has been modified.
	**Action required**: You must select an import action before the app package can be imported.

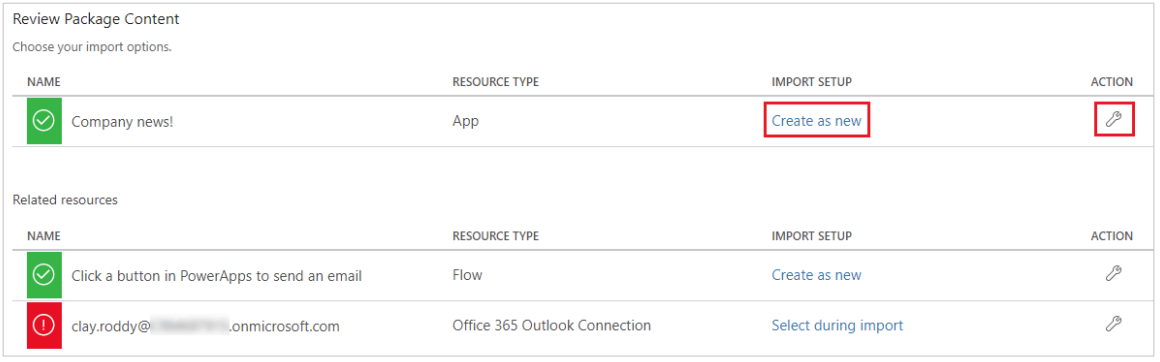
To import a canvas app package

1. Sign in to Power Apps.
2. Select **Apps** from the left pane.
3. Select **Import canvas app**.

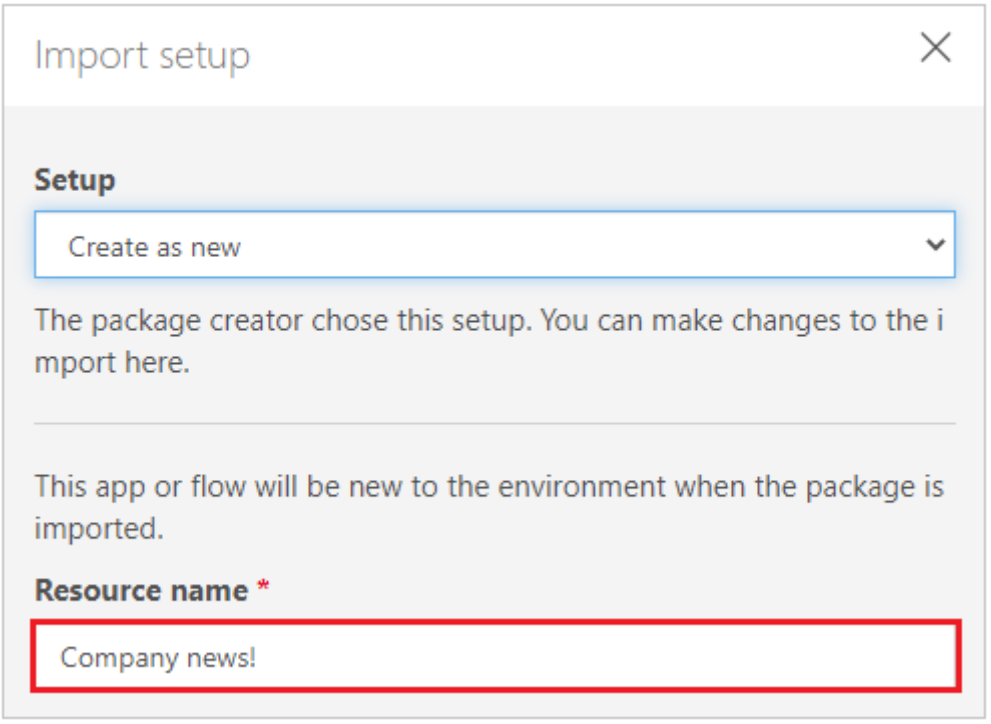


4. Select **Upload**, and select the app package file that you want to import.

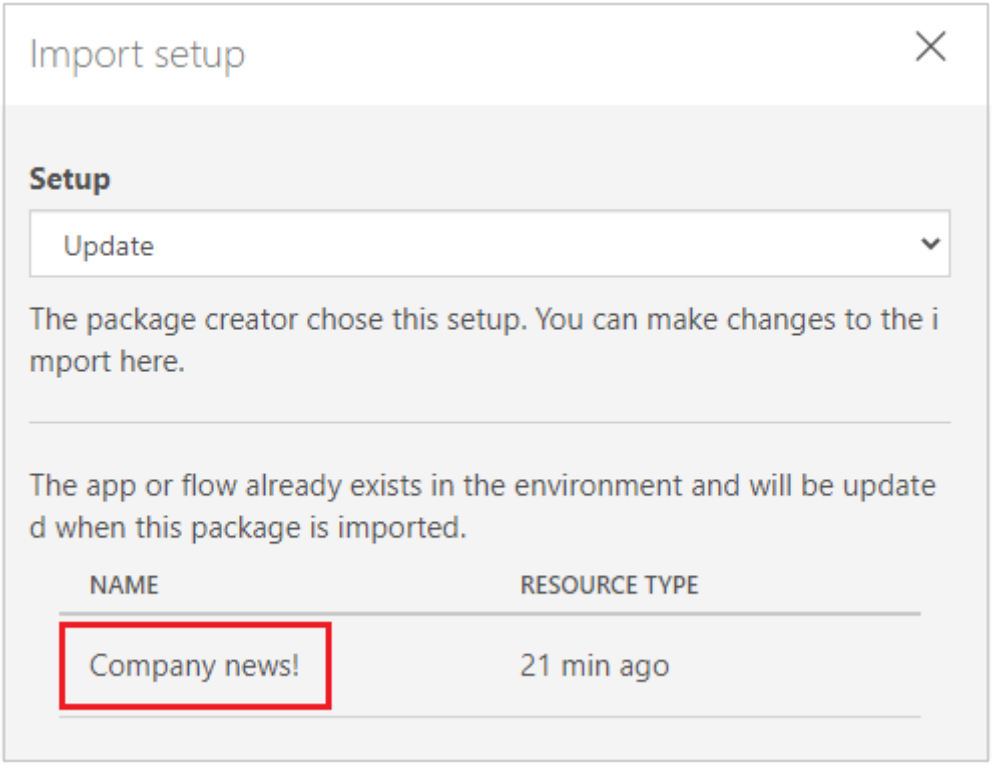
5. Select the **Import Setup** action, or  for a resource.



- If you select **Create as new**, you can change the resource name.

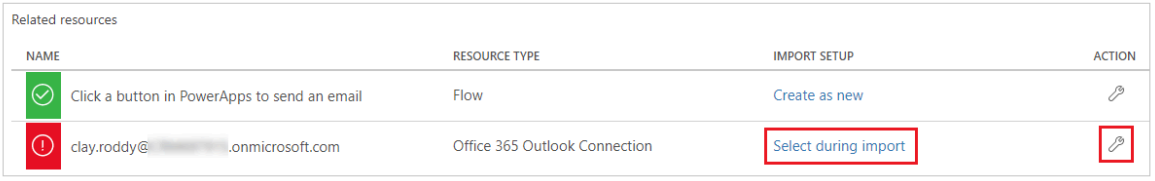


- If you select **Update**, select an app that you want to update during the import.

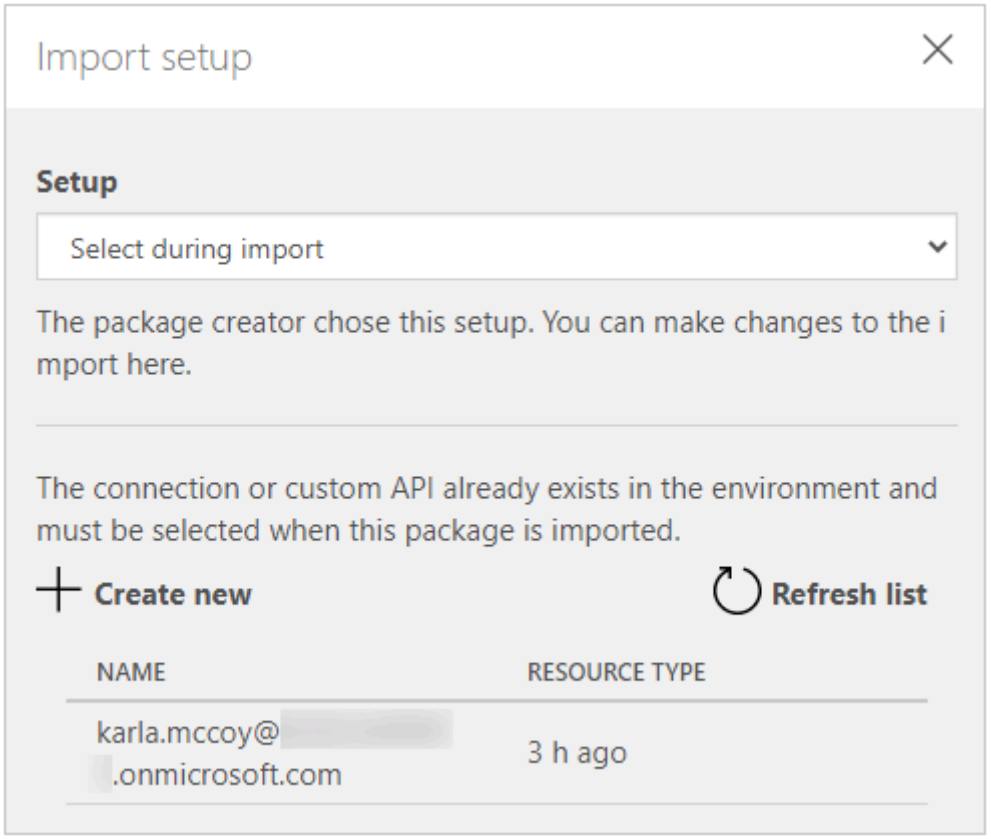


6. Repeat the previous step for each additional resource in the app package.

7. (Optional) If the app package contains references to connections, you can choose the connection from the available connections list.



Select **Select during import** to choose the appropriate connection.



> **Note**
>
> - If no connections are available, or the connection you want isn't listed, select **Create new** to create a new connection in a new browser tab. Ensure that you create the connection for the correct resource type. After creating the connection, return to the import app browser tab and select **Refresh list** to reflect and choose the newly created connection.
> - In some cases, it is necessary to reconfigure the flow association when you import a canvas app with flows. For more information, see Add an existing flow

8. (Optional) If comments were entered for a resource during the export of the app package, the comment icon will appear filled and the text **Comment yes** will appear when you hover over it. Select the comment to view it.



9. Select **Import** to import the app package.

> **Important**
>
> If you select to **Update** an existing app, the new changes will be saved as a draft of the app. You'll need to **publish** the app so users can see the changes.

Feedback

Was this page helpful? Yes No

Provide product feedback

Export and import a canvas app as a single app file

Article • 05/17/2025

In this article, you'll learn how to export and import a single canvas app. Before you begin, review the overview article to learn about the different export and import options available.

Using this option, you're able to export a single canvas app in the format of a **.msapp** file extension. This allows you to export a single file that contains the canvas app that you're currently editing and import it.

Permissions

Only the **Owner** or **Co-owner** of an app can export a canvas app package. To import an app, the Environment Maker security role is required on the destination environment, either directly or through a Dataverse team that is part of the **AAD Security Group** category. Custom security roles are not currently supported.

Export .msapp files in Power Apps

1. Sign in to Power Apps.
2. Open the canvas app that you want to export in Power Apps Studio.
3. In the Power Apps Studio, select **Save** drop-down menu and then select, **Download a copy**.

The **.msapp** file is downloaded to your device.

Import .msapp files in Power Apps

1. Sign in to Power Apps.
2. From the left navigation pane, select **Apps**.
3. Select **Import app** > **From file (.msapp)**.

When Power Apps Studio opens, you can choose the appropriate **.msapp** file to import. Power Apps Studio will reload with the **.msapp** file opened.

> **Note**
>
> For more information on resources included in the single app file, see Resources included.

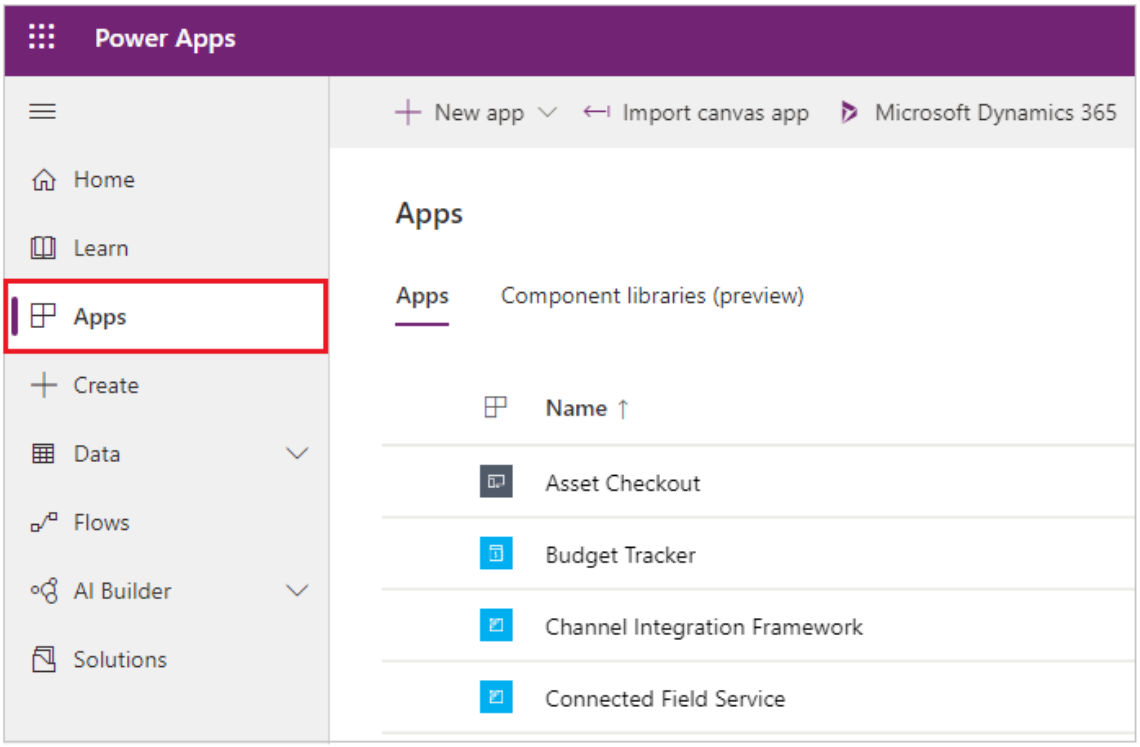
Delete a canvas app

Article • 07/13/2024

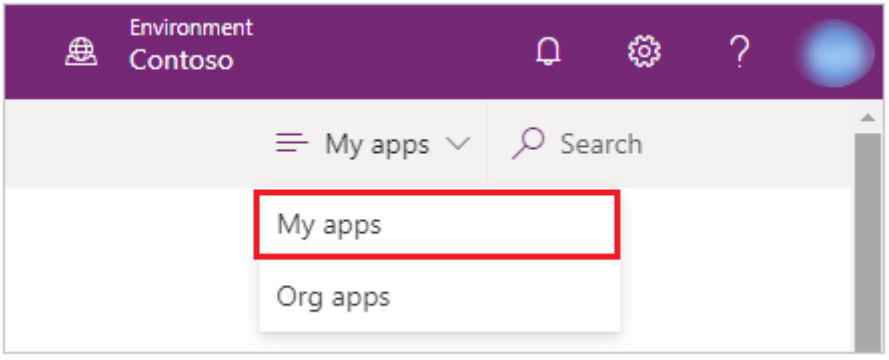
This article shows you how to delete a canvas app.

Delete an app as the owner

1. Sign in to Power Apps.
2. Select **Apps** from the left pane.



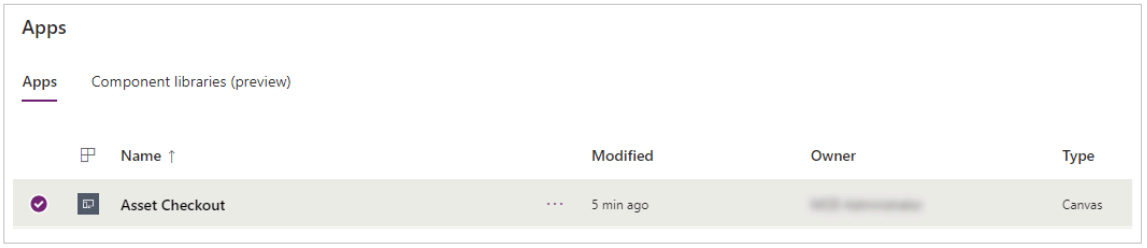
3. (optional) Near the upper-right corner, filter the list of apps to show only those apps that you own.



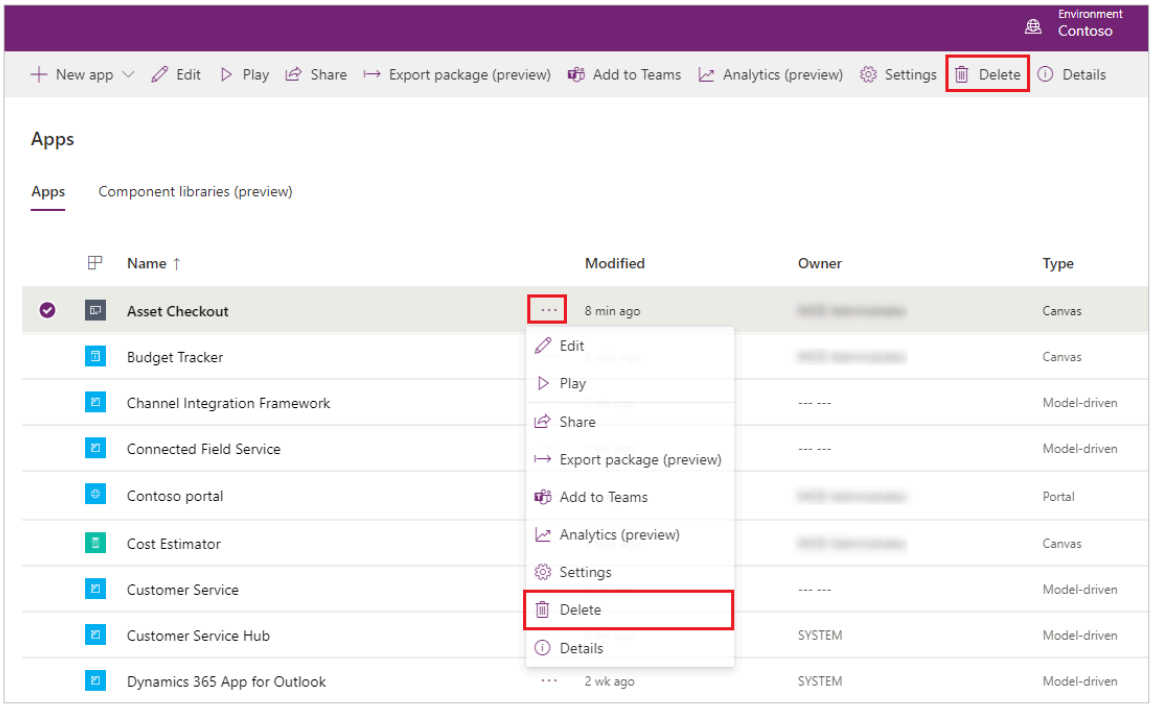
> **Note**

If the app that you want to delete doesn't appear, make sure that you're in the right environment.

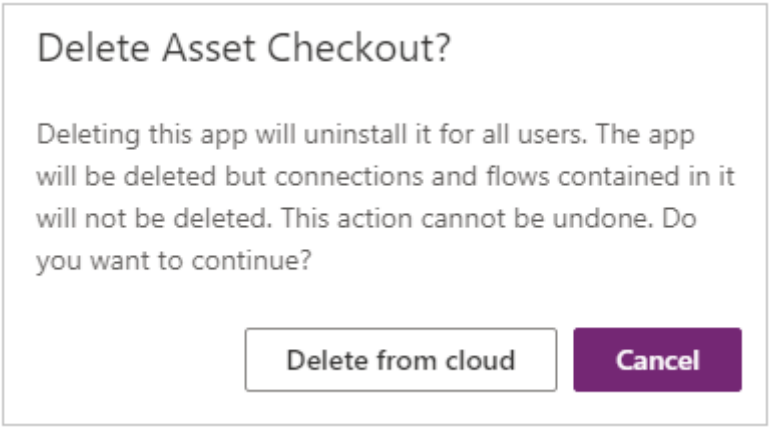
4. Select an app.



5. Select **Delete** from the top menu. You can also select **More Commands** (..) and then select **Delete** instead.



6. Select **Delete from cloud**.



> **Important**
>
> This action deletes the app not only from your account but also from the accounts of all users with whom this app was shared.

Delete an app as the administrator

If owner of an app is unavailable, a Power Platform or Dynamics 365 admin can set the owner of an app. And then, the new owner can delete the app.

To set a user as the owner of an app, use the Power Apps cmdlets for administrators.

> **Tip**
>
> You can also recover a deleted canvas app using **Power Apps cmdlets for administrators** as long as the app is **discoverable**.

See also

- Share an app
- Change app name and tile
- Restore an app to a previous version
- Export and import an app

Feedback

Was this page helpful? Yes No

Provide product feedback

How to check license designation for an app

Article • 03/03/2023

Starting October 2019, several connectors are reclassified from Standard to Premium.

Power Apps licensing FAQ outlines reclassified connectors. Apps using these connectors before the reclassification have been granted an extended timeframe allowing users without a premium license to access these apps.

The following table outlines the designations and which license an end user must have to access an app.

Designation	Definition
Standard	An app that only uses standard connectors. An end user must have a Power Apps for Office 365 plan, per app plan, or a per user plan to access this app.
Extended	An app allowed to use connectors promoted to premium on October 1, 2019. An end user must have a Power Apps for Office 365 plan, per app plan or per user plan. Power Apps licensing FAQ outlines which connectors were promoted to premium on October 1, 2019.
Premium	An app that uses at least one premium connector, a custom connector, or an on-premises gateway. An end user must have a per app plan or per user plan to access.

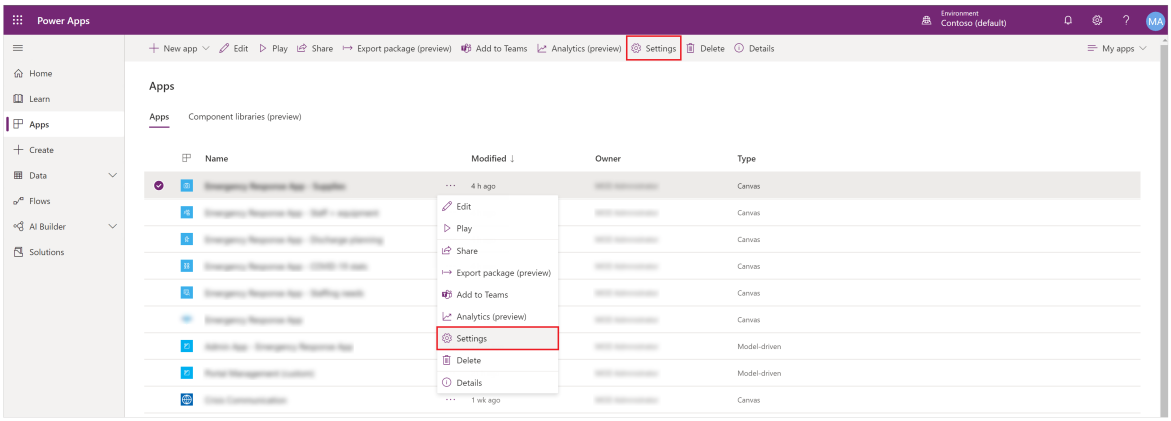
> **Note**
>
> Power Apps built in Microsoft Teams environments that connect to Dataverse for Teams, or use custom connectors have **Premium** license designation.
>
> - End-users with a **Microsoft 365 subscription including Dataverse for Teams use rights** will be able to run these apps in Microsoft Teams.
> - End-users that attempt to run these apps outside of Microsoft Teams are required to have a per app plan or per user plan.

Check app license designation from app settings

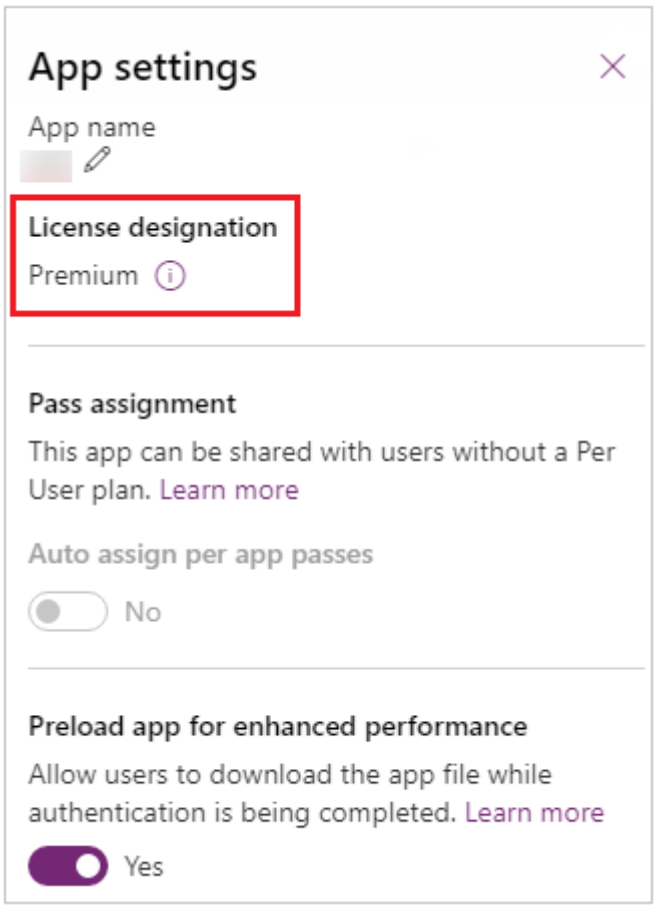
1. Sign in to Power Apps.

2. Select **Apps** from left side.

3. Select an app from the list of app. You can use the **Settings** option from top or, use the **More Commands** (...) and then **Settings** from the drop-down menu:



4. Select **Settings** to see the license designation information:

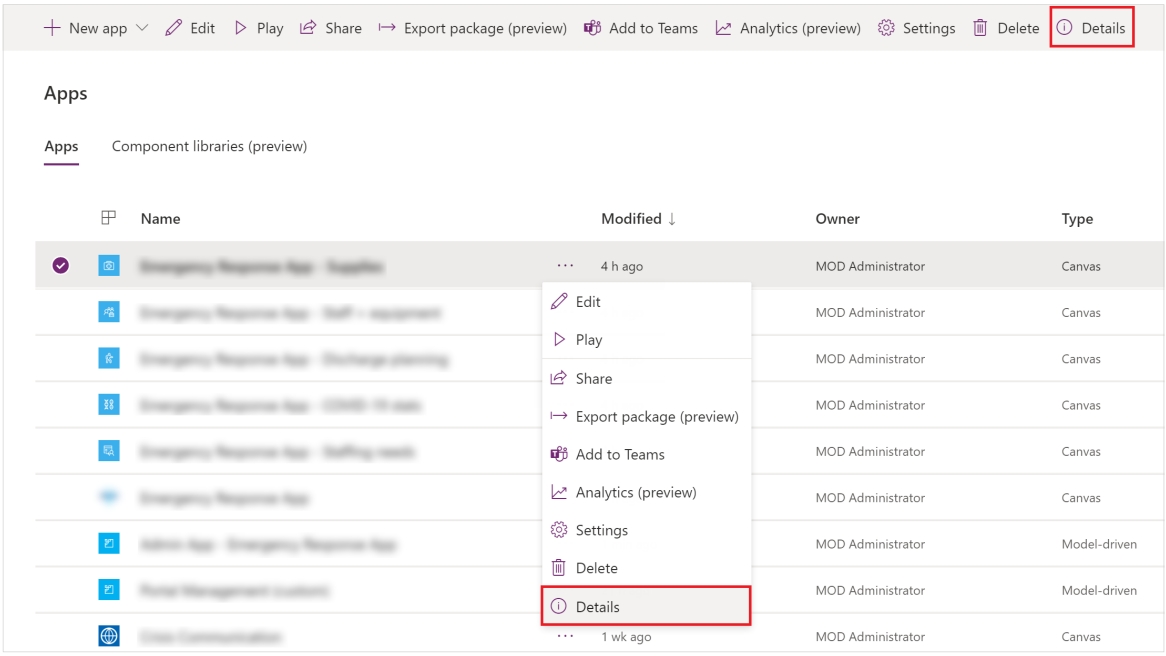


Check app license designation from app details

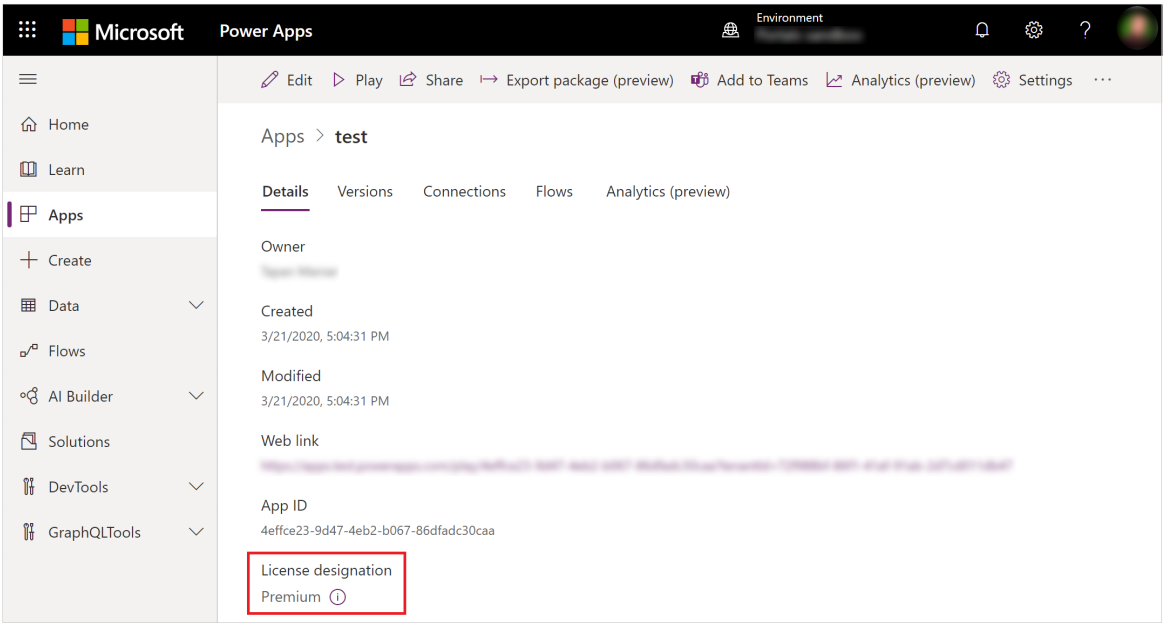
1. Sign in to Power Apps.

2. Select **Apps** from left side.

3. Select an app from the list of app. You can use the **Details** option from top or, use the **More Commands** (...) and then **Details** from the drop-down menu:



4. Select **Details**:



Known issues

- The use of a premium connector in a flow connected to an app isn't recognized by the app's license designation.

  For example, an app that only uses standard connectors and is connected to a flow using a premium connector, like the HTTP connector, will have the license designation as **Standard** although it requires the app users to have **Premium** use rights.

Pass assignment

For information about Pass assignment, read Power Apps per app plans.

Next steps

- Power Apps licensing FAQ

See also

- Edit an app
- Delete an app

Restore a canvas app to a previous version

Summarize this article for me

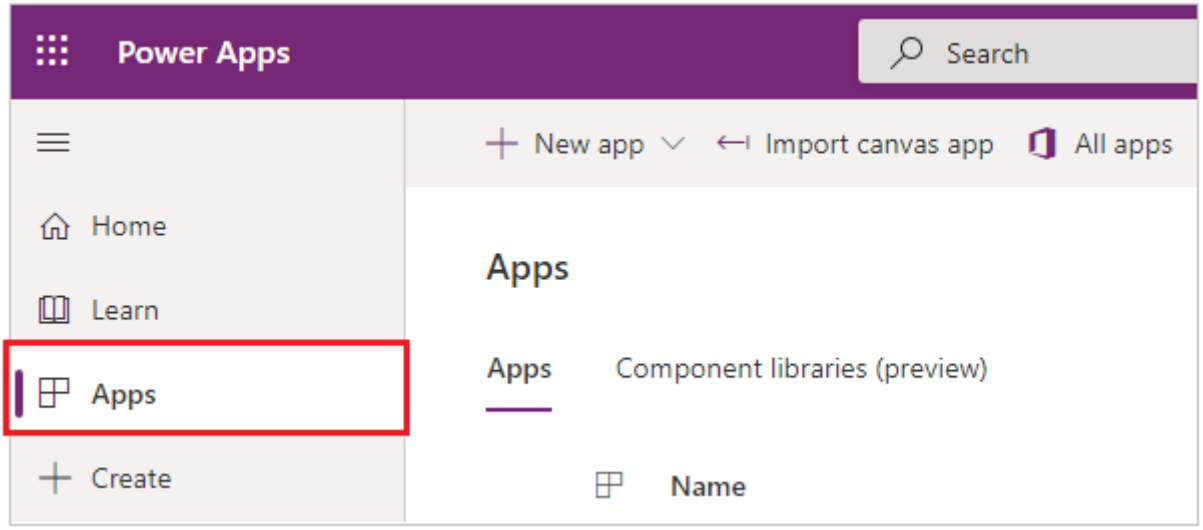
[This article is prerelease documentation and is subject to change.]

This article shows you how to restore a canvas app to a previous version that was saved to the cloud from your Power Apps account.

Restore an app from your account

1. Sign in to Power Apps.

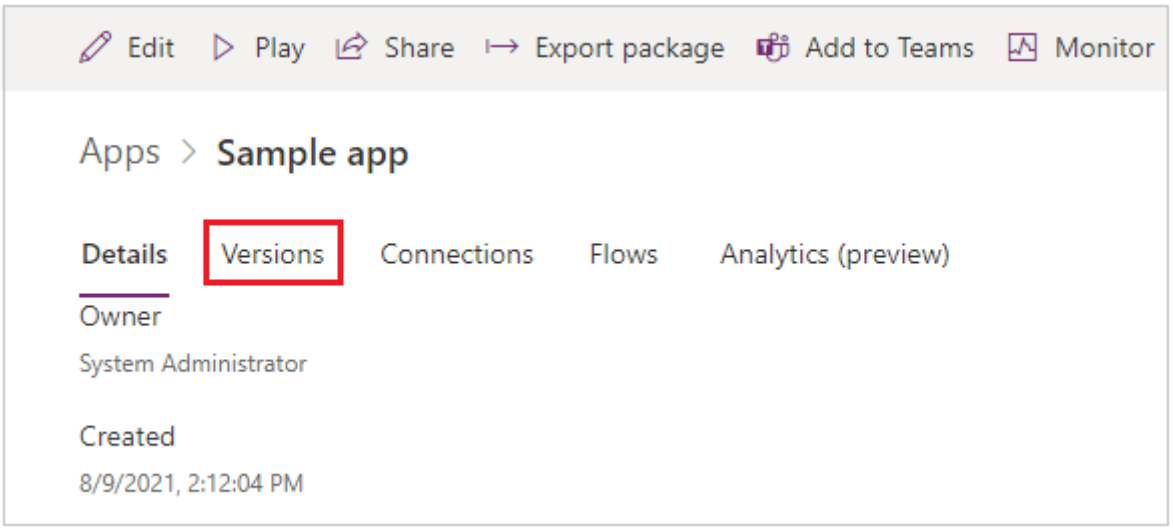
2. On the left navigation pane, select **Apps**.



3. Select your app and then select **Details** on the command bar.

> **Note**
>
> If the app that you want to restore doesn't appear, make sure that you're in the right environment.

4. Select **Versions** and then select the app version that you want to restore.

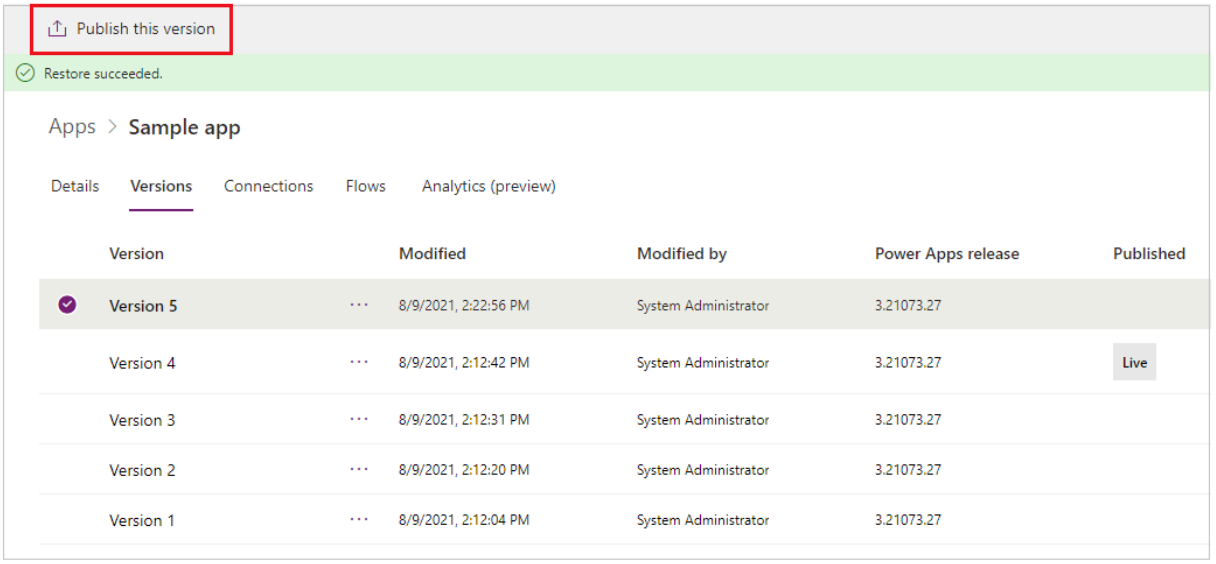


5. Select **Restore** and confirm your changes by selecting **Restore** again.

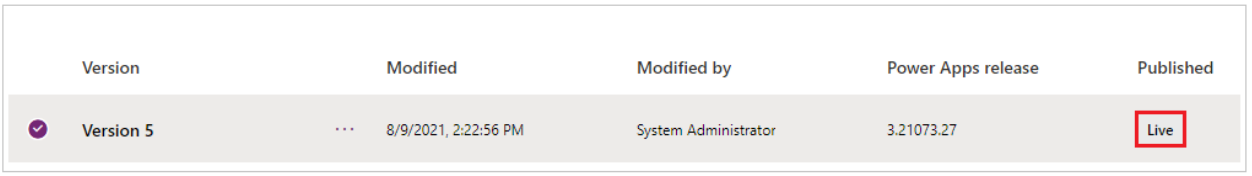
> **Note**
>
> - For apps older than 6 months, this action will repackage the app with the oldest version available. Functionality may have changed from the original packaging of the app

New version is created.

6. (Optional) If you want to publish the restored version, select **Publish this version**, and then select **Publish this version** when prompted to confirm.



Notice the published restore version that shows up as **Live**.



> **Note**
>
> **Publish this version** option only appears for the latest unpublished version of an app.

More resources

Share an app
Change app name and tile
Delete an app

Last updated on 03/11/2026

Test Studio

Article • 05/21/2025

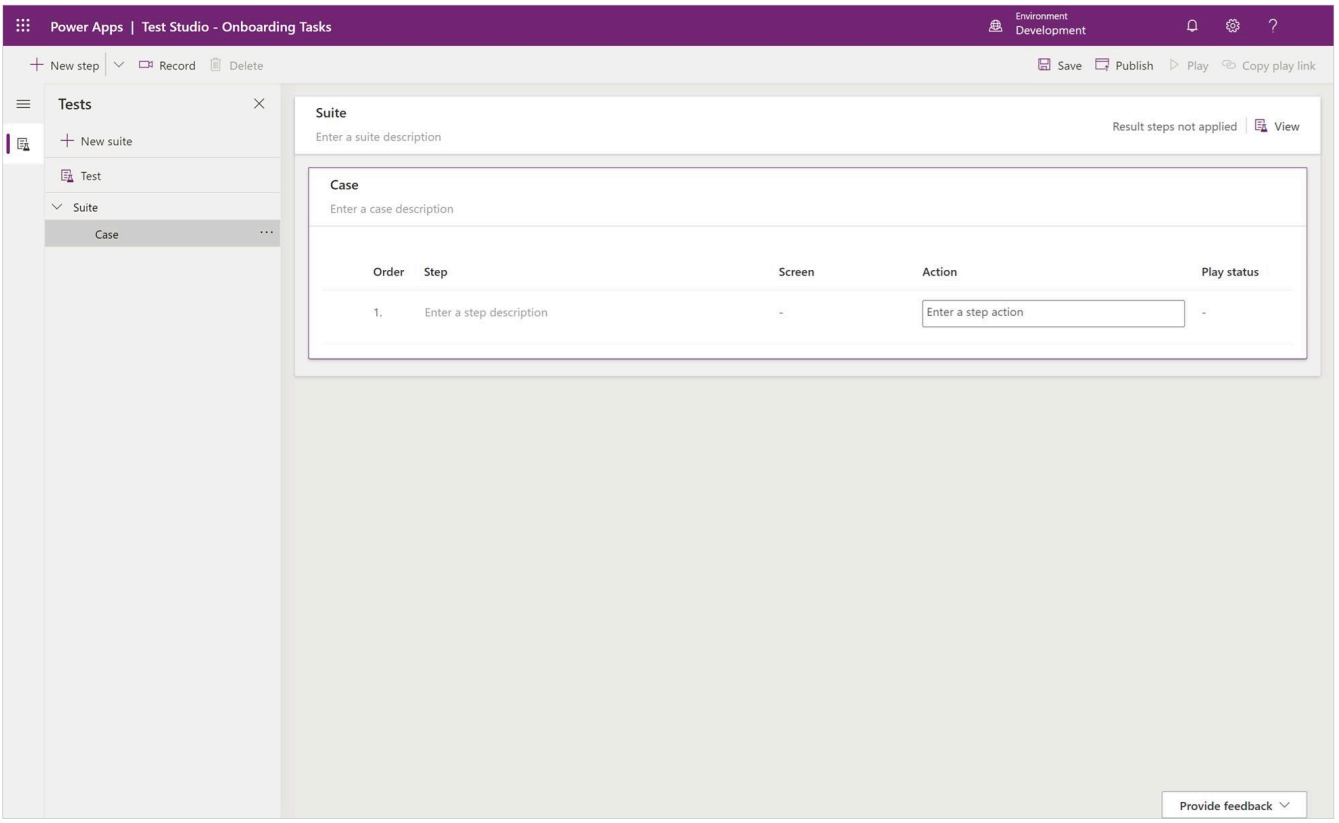
Build end-to-end UI tests for your canvas app using Test Studio. Maintain your app quality by continually validating that your app works as expected when new changes or updates are deployed.

Overview

Testing is an important part of the software development life cycle (SDLC). Testing can help ensure the quality of the app delivered to customers. It can identify issues or defects early in the release process and provides an opportunity to fix these issues to make the app more reliable before releasing changes. Depending on the size and usage of the app, manual testing of new changes might be enough. However, as the app grows in complexity and usage, you might need to consider a test strategy instead of manual testing. If the app is mission-critical, even a small mistake can have a significant impact.

Increased app changes can result in longer testing cycles. Eventually, regression testing of the app might be longer than the time spent to develop new features. In fast-paced development, thoroughly testing every feature in the app becomes a bottleneck to releasing software updates. One option to reduce the time taken during a test cycle and on regression testing is test automation. Test automation can help you test your app with minimal effort, reducing testing time and identifying critical issues before release.

Power Apps Test Studio is a low-code solution to write, organize, and automate tests for canvas apps. In Test Studio, you can write tests using Power Apps expressions or use a recorder to save app interaction to automatically generate the expressions. You can play written tests back within the Test Studio to validate app functionality, and also run the tests in a web browser and build the automated tests into your app deployment process.



Prerequisites

You must be creator or co-owner of an app to test the app with Test Studio.

Test Studio terminology

The following section explains key Test Studio terminology.

Test cases

Test cases are made up of a series of instructions or actions, called test steps. Test cases are executed to validate that your app, or specific features in your app, is working as you expect. For example, in an Expense app, you would like to ensure that only expenses with associated actual cost can be submitted. A test case can help verify that this condition or requirement is always met.

In Test Studio, test steps are written using the Power Apps expression language. Test expressions can consist of both the functions available when building your app and additional expressions to support automated testing.

Test suites

Test suites are used to organize or group test cases together. As the number of test cases in the app grows, you might consider organizing the test cases in specific features or functionality. For example, you might have one test suite with test cases to validate expense report submissions and another test suite that focuses just on expense approvals.

Test cases contained in test suites are run sequentially. The app state is persisted across all test cases in a suite. For example, if you have a test case that completes on Screen 5 in your app, the next test case in your test suite will begin running from Screen 5. It allows you to break down a complex test scenario into multiple test cases within a single suite, and the state is shared across all test cases. If your second test case expects to begin at the start screen of the app, you can navigate to the start screen as the first step in your test case. It's important to remember that the app is not reloaded at the beginning of every test case in a test suite when planning your test execution.

Test assertions

Every test case should have an expected result. To validate the expected result of a test against the actual result of your test, you can write test assertions. An assertion is an expression that evaluates to true or false in the test. If the expression returns false, the test case will fail.

In the expense app example above, you can write an assertion to validate whether an expense report is created with an expense line item having zero cost associated.

Best practices

When testing canvas apps using Test Studio, consider the following best practices to gain maximum benefits to improve your app quality:

1. **Determine which test cases should be automated.**

   It's difficult to automate all tests and we do not recommend that you completely rely on test automation. Manual testing should be performed in addition to test automation. Tests best suited to automation are:

   - Repetitive tests.
   - High business impact functionality tests.
   - Features that are stable and not undergoing significant change.
   - Features that require multiple data sets.
   - Manual testing that takes significant time and effort.

2. **Keep test cases small.**

While a single test case can support testing all functionality in your app, we recommend that you avoid writing a monolithic test case and try to divide it into multiple test cases. Each test case could test a specific feature or functionality in your app. A failed assertion in a large test case might cause other functionality to remain untested. Using multiple test cases contained in a test suite allows other functionality to get tested regardless of whether a previous test case failed. This strategy also makes it easier to isolate test failures.

3. **Keep expressions to a single test action.**

   A test action can contain multiple expressions. Large multi-action test expressions for a single step might impact your ability to debug and isolate any test failures. Consider dividing a test step with multiple actions into more test steps of single actions to identify issues faster.

4. **Every test case should have an expected result.**

   Each test case should have one or more expected results. Test assertions should be used to validate the expected outcomes of your test against the actual outcomes. Multiple assertions can be written for a single test case.

5. **Use test suites.**

   For maintenance, group or categorize similar test cases together and describe the purpose and expected results of your test.

Known limitations

While we work to provide full control coverage in Power Apps Test Studio is in progress, the following functionality is currently unavailable:

- Components.
- Code components written in the Power Apps Component Framework.
- Nested galleries.
- Media controls.
- Formula-level error management experimental feature needs to be turned on for the app.
- Support for controls not listed in the Select and SetProperty functions.
- Person-type columns.
- Test Studio is not compatible with the experimental Git version control feature, and will not work properly if that feature is enabled.

Next steps

- Working with Test Studio

See also

- Automate tests with Azure Pipelines using classic editor

Working with Test Studio

Article • 03/03/2023

In this quickstart, you'll create tests for a canvas app called Kudos. You can also explore and discover testing concepts and apply them to writing tests for your own canvas apps. The sample Kudos app is part of a suite of employee engagement apps available to download from Employee Experience Starter Kit.

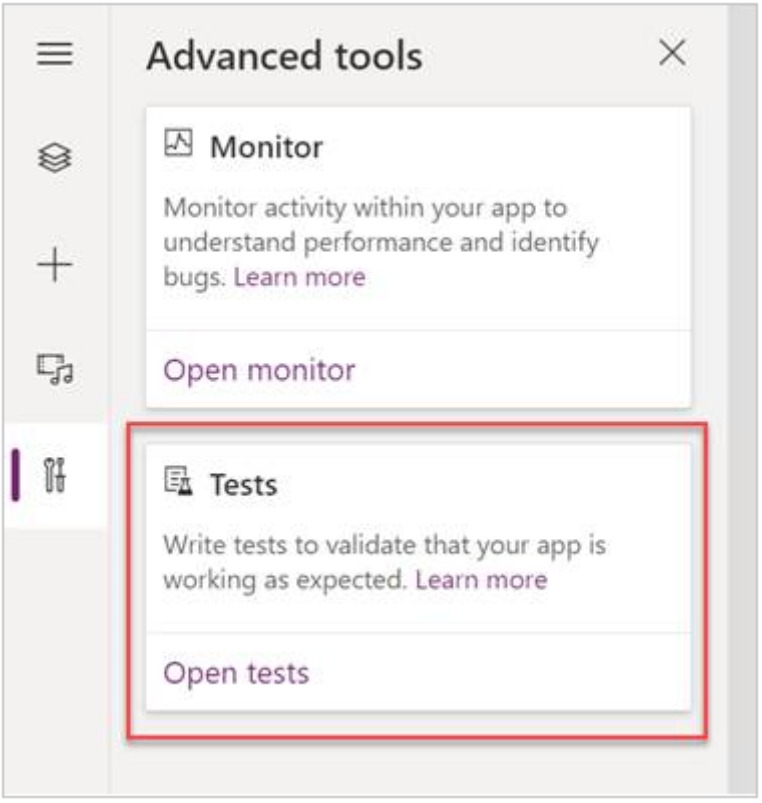
Watch this video to learn how to work with Test Studio:
https://learn-video.azurefd.net/vod/player?id=3fe85f46-745a-411c-aba8-6044cb184e78&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fworking-with-test-studio

Open Test Studio

1. Sign in to Power Apps.
2. Create a new app or edit an existing app.
3. Save your app to Power Apps to open Test Studio.

   > **Note**
   >
   > You must save an app before you can write tests for the app.

4. Select **Advanced tools** in the left navigation.
5. Select **Open tests** to open the Test Studio for this application. This action opens Test Studio in a new browser tab.

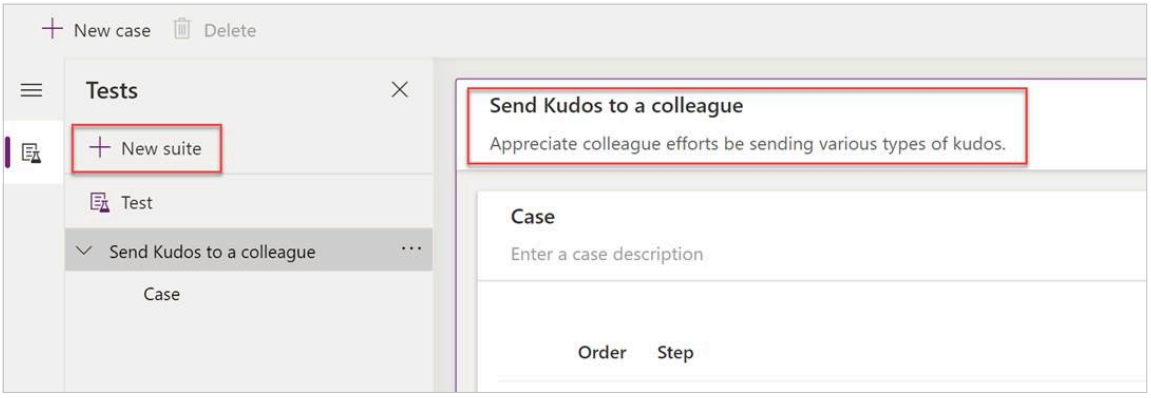


> [!NOTE]
> Tests are published and stored in the app package. Exporting and importing a canvas app package to another environment will also include all the test definitions such as test suites and test cases you have created.

Create a test suite

By default, a test suite and test case are created for you in Test Studio. Test suites are used to organize your test cases. An app can contain one or more test suites. You can use the default test suite and case to begin writing your tests immediately or create a new test suite.

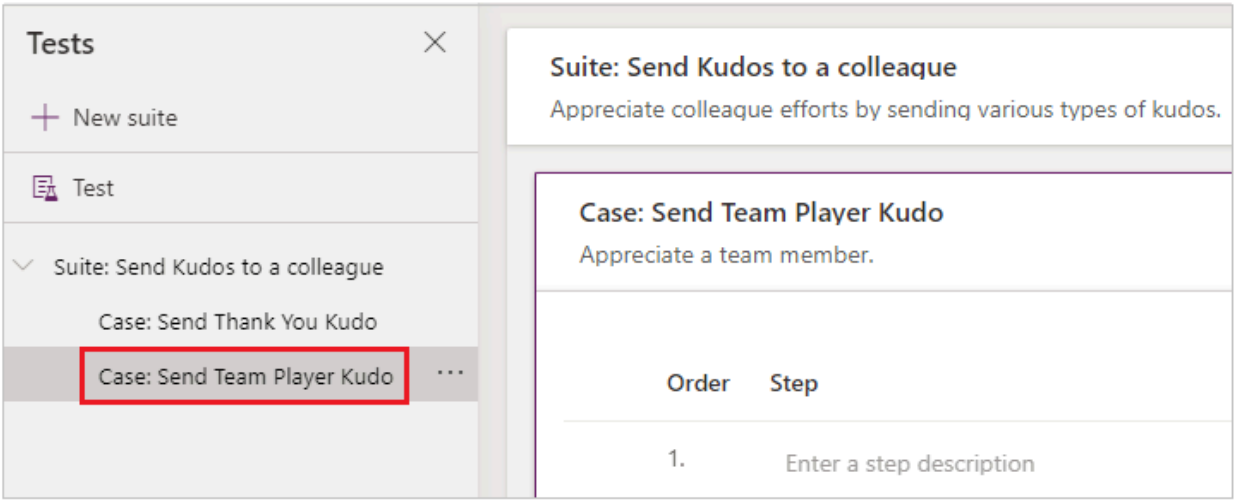
1. Select **New suite**.
2. Update the **Test suite name and description** by selecting the fields on the main grid.



Create a test case

Depending on how you want to organize or group your tests, you can create multiple test cases in a test suite. Each case can test a specific feature or a subset of functionalities in your app.

1. Select a test suite.
2. Select **New Case** in the top menu to create a new case.
3. Update the **Test case name and description** by selecting the fields on the main grid.



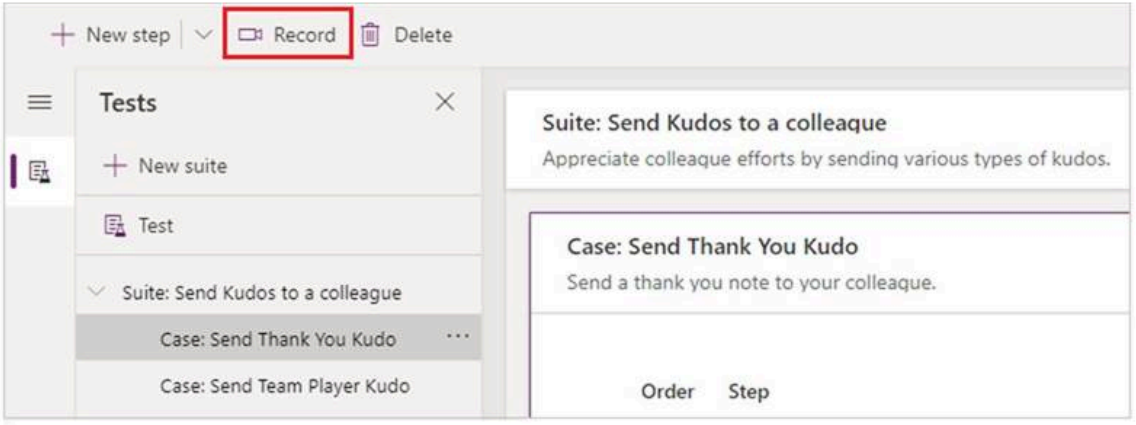
Record a test case

A test case consists of test steps that contain actions. Test actions are written using Power Apps expressions that perform a task. You can use the recorder to automatically generate the test steps as you interact with your app. After you record, you can update the test case, add new steps, delete steps, and write test assertions to validate the result of your test.

> **Note**
>
> Only a published app plays in record mode. Publish any recent changes to the app before you start recording a test case. Recording without publishing recent changes causes the last published version of the app to play in record mode.

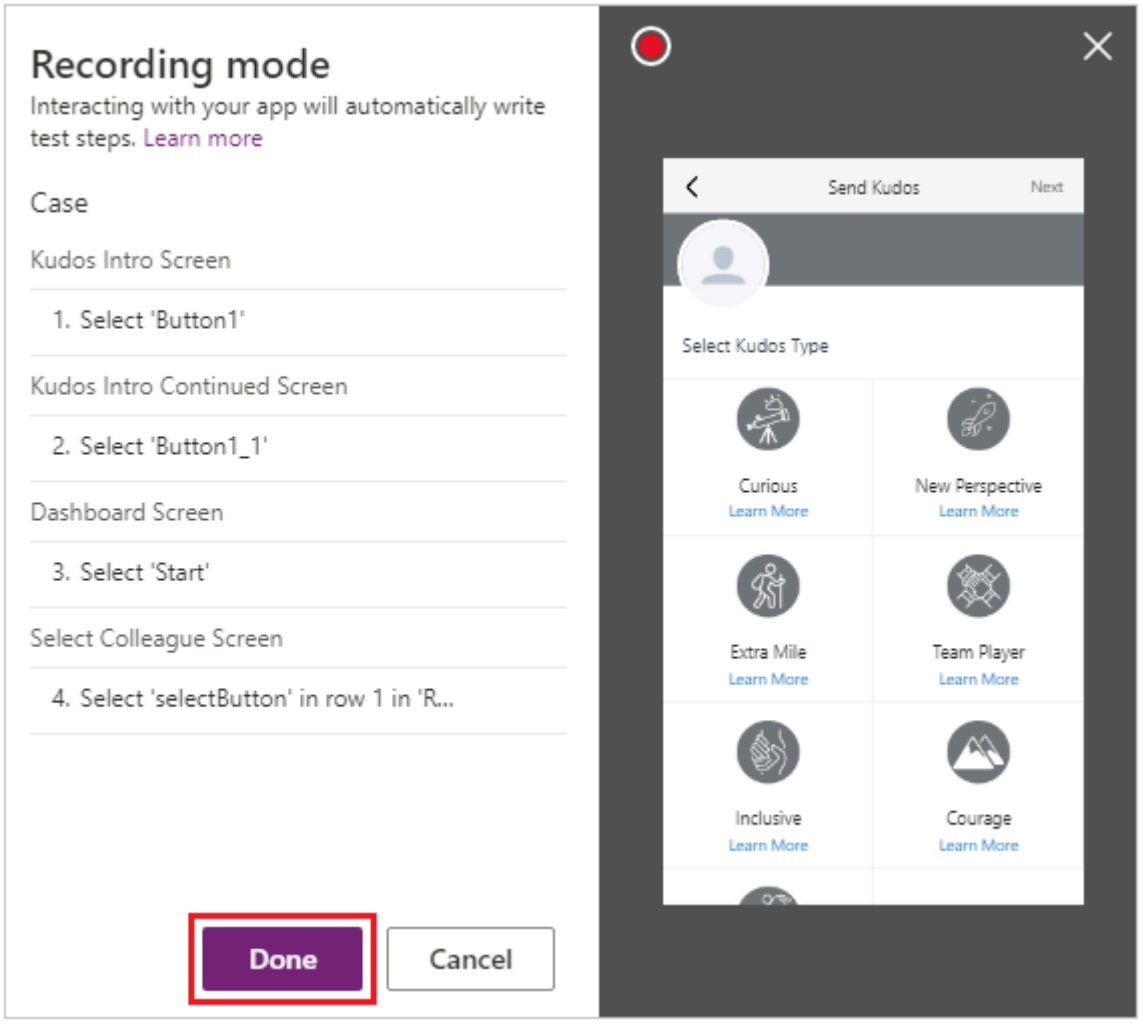
1. Select **Record** from the top menu. This action opens the published app with recording mode in a new browser tab.

   > **Important**
   >
   > Recording on an existing test case overrides any existing test steps already present.

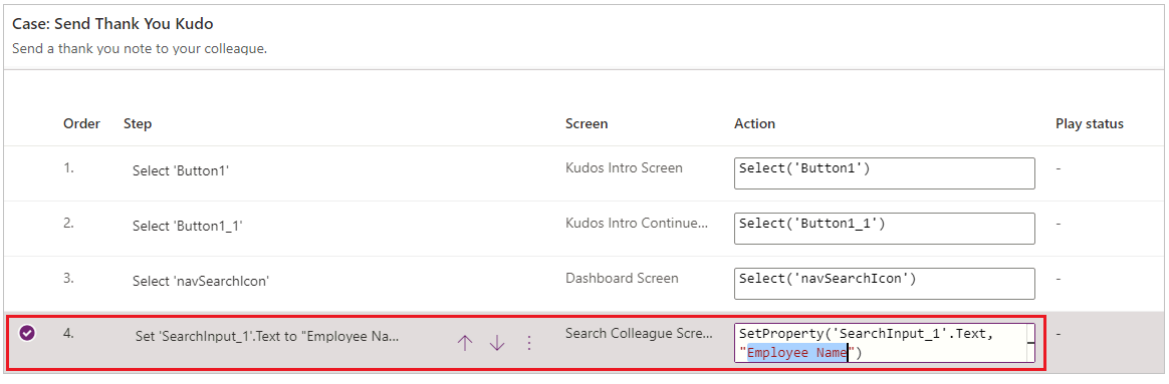
2. Interact with your app. Your actions are **recorded** in the left pane.

3. Once interaction completes, select **Done**. Optionally, you can select **Cancel** to return to Test Studio without your interactions getting recorded.



4. View the test steps and the expressions that were automatically generated for you in Test Studio.

5. Edit the step description text in the main grid if necessary. You can also update the test step actions by selecting the formula on the main grid.



Add test steps and test assertions

Every test case should have an expected result. In the Kudos example, one of the expected results of sending a kudo is creating a new record in the Microsoft Dataverse

(Dataverse) database. You'll now update the test case and add additional test steps to validate that a record was created successfully.

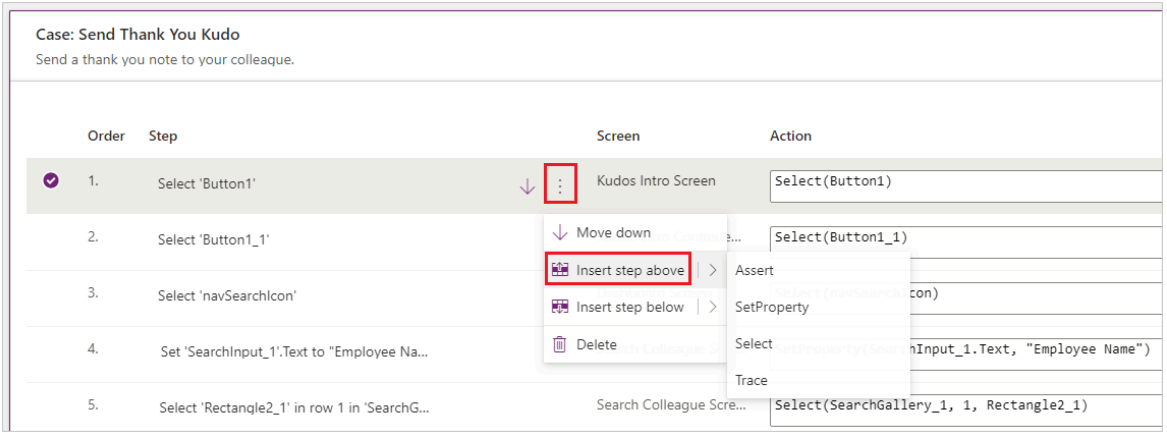
Follow these steps to verify a successful record creation:

- Initialize a variable for the kudo record count in the database at the beginning of the test case.
- Initialize a variable for the kudo record count in the database at the end of the test case.
- Write a test assertion expression to validate it incremented by one count. If the count doesn't increase by one, the test assertion fails, and your test case fails.

To add test steps and test assertions in the Kudos app:

1. Select Step 1 or the step above which you want to insert a new step.

2. Select **Insert a step above** from the top menu or by selecting the option from the active row. This action creates an empty step.



> **Note**
>
> When you select **Insert step above**, a new blank step is added above the current step. You can also use **Assert**, **SetProperty**, **Select** or **Trace** actions instead. This adds a step with a respective action formula that you can edit.

3. Update the step description. For example, "Count Kudo in database".

4. Enter an expression or formula into the action input to count the records in the database before executing the test.

   You can use any supported expression. You can also query any data sources, collections, variables, or run flows that are contained in your app, and create new global variables or collections to use in your tests.

```
Power Fx
```

```
Set(kudosBeforeTest, CountRows(Filter(Kudos, Receiver.Email =
"someone@example.com")))```
```

5. Select Step 2 or the step above which you want to insert a new step.
6. Select **Insert a step above** from the top menu or by selecting the option from the active row. This action creates an empty step.
7. Enter an expression or formula in the action input to Trace and write the *kudosBeforeTest* value to test the results record.

```
Power Fx
```

```
Trace("kudosBeforeTest : " & kudosBeforeTest);
```



8. Go to the bottom of the test case and insert a new step to count the records in the database after the test has completed.

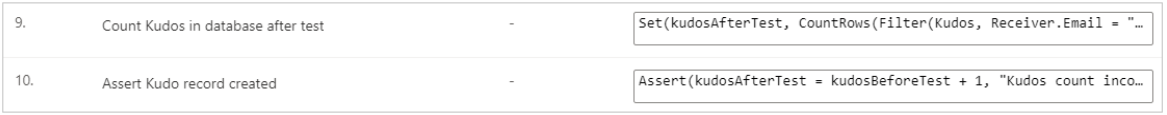
```
Power Fx
```

```
Set(kudosAfterTest, CountRows(Filter(Kudos, Receiver.Email =
"someone@example.com")))```
```

9. Add a final step to validate that the record count in the database has increased by a count of 1, and enter the following assertion action to verify:

```
Power Fx
```

```
Assert(kudosAfterTest = kudosBeforeTest + 1, "Kudos count incorrect.
Expected : " & kudosBeforeTest + 1  & " Actual :" & kudosAfterTest)
```

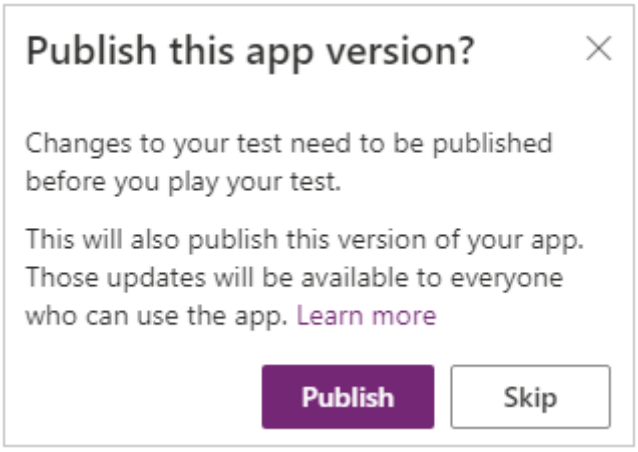


10. Save the test case from the top-right menu in Test Studio.

Play back your test

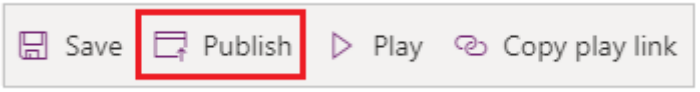
You can play back your recorded test to validate app functionality. You can play back all tests within a single test suite, or a single test case.

Before you play the recording back with recent changes, you must publish the app:

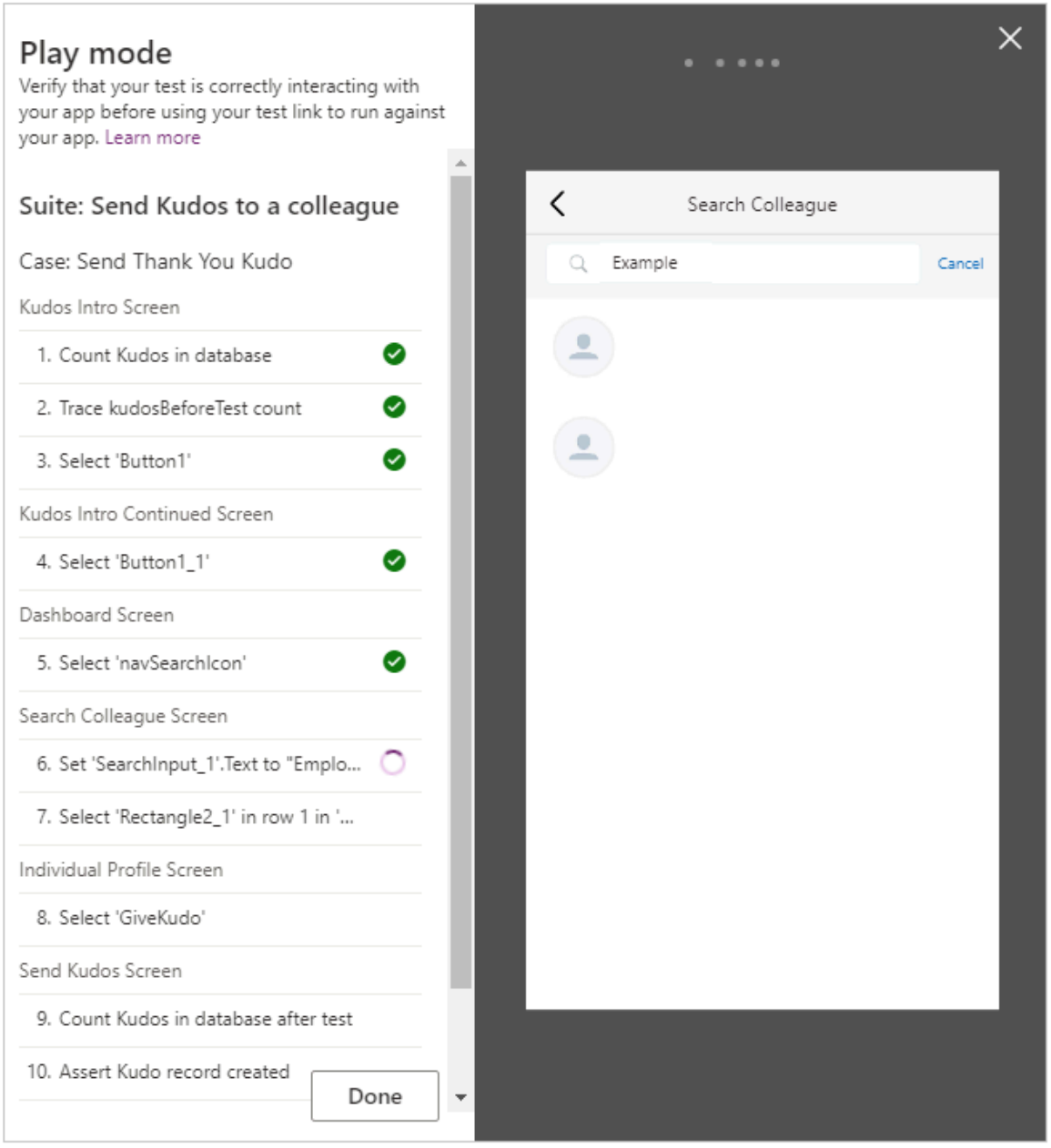


> **Important**
>
> If you skip publishing, the recording playback will not contain your recent test changes. The last published test case or suite will play against the app.

1. Select **Publish** to automatically save and publish your test.



2. Select either a test suite or a single test case.
3. Select **Play**. The published app opens in **Play** mode, and you can see your test steps playing back automatically. A green check mark indicates when a test step is executed successfully. If a step fails, a red failure indicator along with a failure message is displayed.



4. Select **Done** to return to Test Studio.

Failed assertion

In this section, you'll change the test assertion to experience a failed test:

1. Edit the assertion step by selecting the expression box.
2. Update `+ 1` to `+ 2` in test action. This update means that the test expects two records to be created, which is incorrect. If the test is successful, only one record should be created in the database.

Power Fx

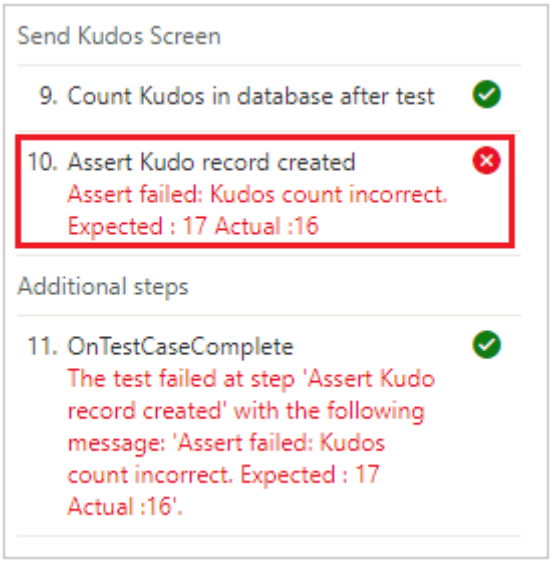
```
Assert(kudosAfterTest = kudosBeforeTest + 2, "Kudos count incorrect.
Expected : " & kudosBeforeTest + 2  & " Actual :" & kudosAfterTest)
```



3. Select **Publish**.

4. Select **Play**.

5. View the test playing back. The final step now fails and shows an error and the message you provided in the assertion step.



Playing tests in a browser

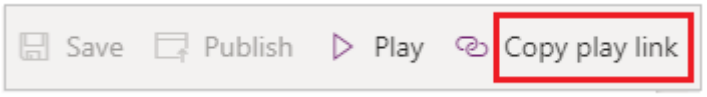
You can copy a link to play a test in separate browser outside Test Studio. It helps integrate your tests in a continuous build and release pipeline such as **Azure DevOps**.

The play link for the selected test is persisted. It doesn't change for the test suite or test case. You can update your tests without the need to modify build and release processes.

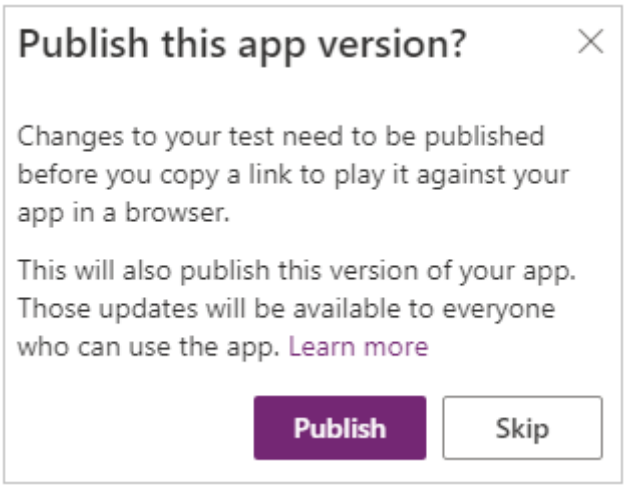
To play tests in your browser:

1. Select a test suite or test case in the right pane.

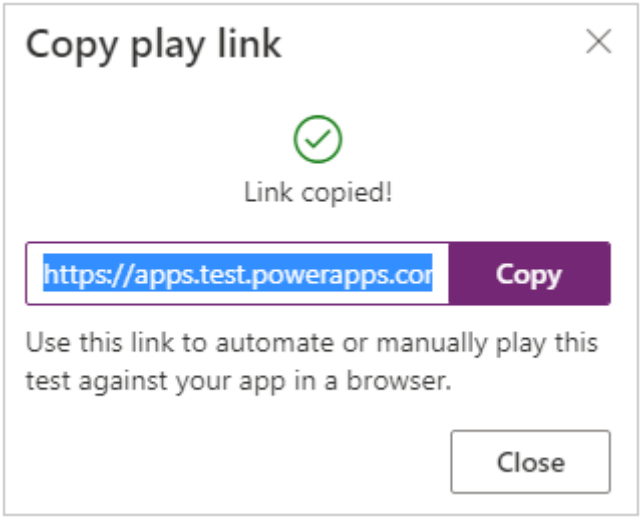
2. Select **Copy play link**.



3. You're prompted to publish your tests if there are any unpublished changes.



4. You can select to skip the publish process and copy the play link. New test changes don't play if you skip.



5. Open a browser and paste the URL into the address bar to play the test.

6. View your test playing back.

Setup your tests

The **OnTestCaseStart** property of a Test Suite can be used to setup your test. The expression entered for this property triggers for every test case in a suite before the case begins executing. **OnTestCaseStart** helps you avoid writing the same test steps at the beginning of every case. You can customize this property to run set up tasks that are common to all cases in the suite such as:

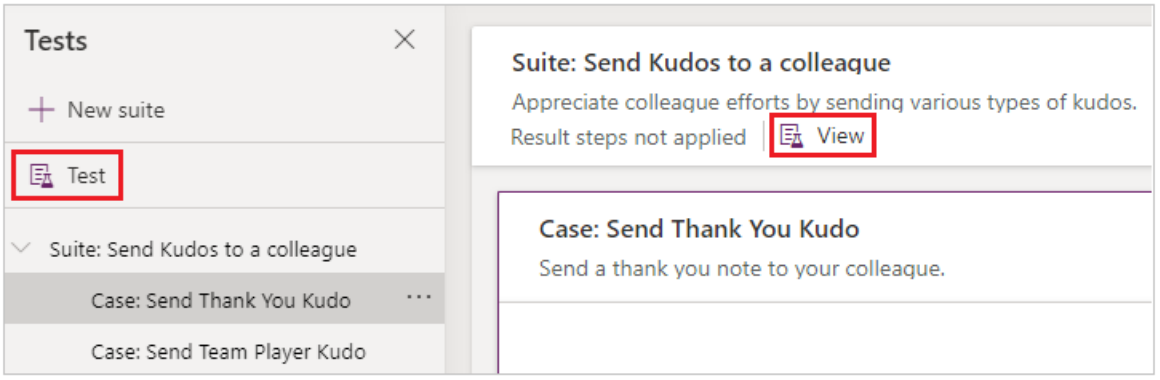
- Always starting test execution from the first screen.
- Initializing common collections or variables.
- Fetching test data from a data source for the current executing test

The **TestCaseInfo** record contains details for the current test that is executing. It contains the following properties:

- *TestCaseName* – the name of the test case.
- *TestCaseDescription* – the description of the test case.
- *TestCaseId* – the ID of the test case.
- *TestSuiteName* – the test suite name that the case belongs to.
- *TestSuiteDescription* – the description of the test suite.
- *TestSuiteId* – the test suite ID that the case belongs to.

In the below example, you'll customize the **OnTestCaseStart** property so every test case begins from the first screen in your app. You'll also fetch the test data from a data source that can be referenced in the steps for the test case.

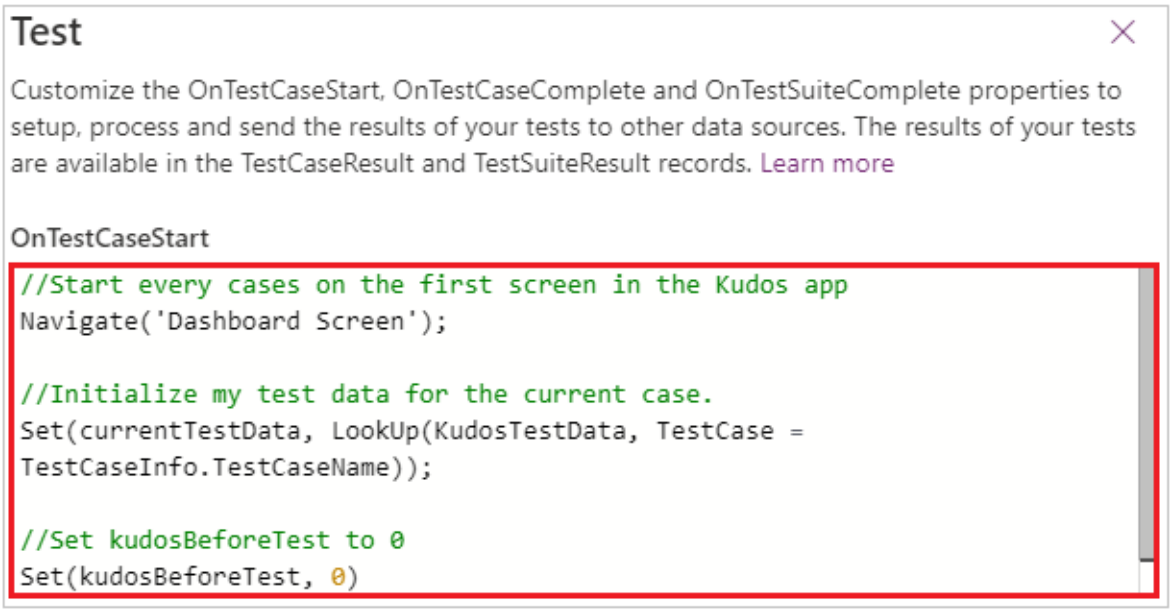
1. Select **Test** in the left pane or **View** on the suite header.



2. Select the **OnTestCaseStart** action.

3. Input an expression to Navigate to first screen and fetch the test data for your test.

```
Power Fx

//Start every cases on the first screen in the Kudos app
Navigate('Dashboard Screen');

//Initialize my test data for the current case.
Set(currentTestData, LookUp(KudosTestData, TestCase =
TestCaseInfo.TestCaseName));

//Set kudosBeforeTest to 0
Set(kudosBeforeTest, 0)
```



Processing test results

The test panel visible when playing back tests in Test Studio isn't visible when using a browser. Because of this behavior, you can't determine the specific test step that executes, or if a test passes or fails.

To determine test results outside of Test Studio, there are two properties called **OnTestCaseComplete** and **OnTestSuiteComplete** available in the test object that you can use to process the results of your tests. When integrating tests into a continuous build and release pipeline like **Azure DevOps**, these properties can be used to determine if you should proceed with the app deployment.

The expression entered for these properties triggers when each case or suite completes. You can customize these properties to process and send the results of your tests to various data sources or services such as:

- SQL Server.
- Dataverse.
- Power Automate.
- Email using Office 365.

These settings apply to every test suite or test case in your app. After each test suite or test case completes, the test results and any trace messages contained in the tests are available in the **TestCaseResult** and **TestSuiteResult** records.

The **TestCaseResult** record contains the following properties:

- *TestCaseName* – the name of the test case.
- *TestCaseDescription* – the description of the test case.

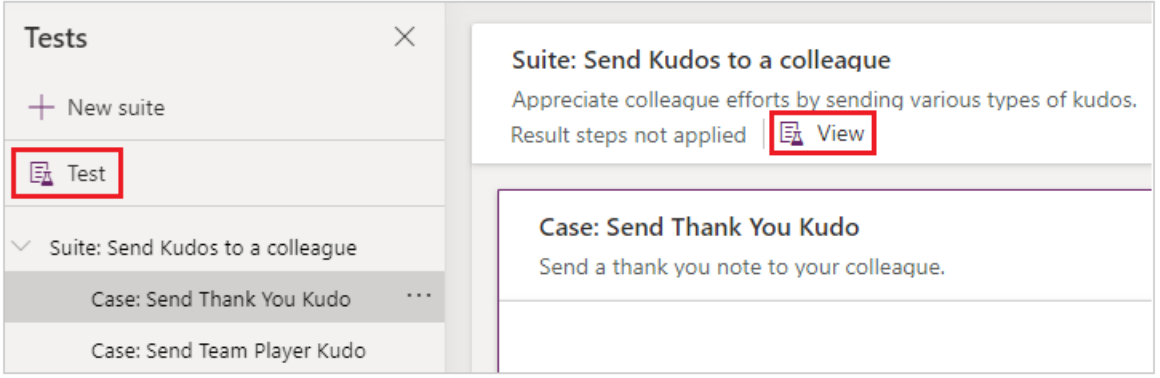
- *TestCaseId* – the ID of the test case.
- *TestSuiteName* – the test suite name that the case belongs to.
- *TestSuiteDescription* – the description of the test suite.
- *TestSuiteId* – the test suite ID that the case belongs to.
- *StartTime* – the start execution time of the test.
- *EndTime* – the end execution time of the test.
- *Traces* – the result of any test assertions and any messages from the Trace function.
- *Success* – indicates if the test case completed successfully.
- *TestFailureMessage* – if the case failed, the failure message.

The **TestSuiteResult** record contains the following properties:

- *TestSuiteName* – the test suite name.
- *TestSuiteDescription* – the description of the test suite.
- *TestSuiteId* – the test suite ID.
- *StartTime* – the start execution time of the test suite.
- *EndTime* – the end execution time of the test suite.
- *TestsPassed* – the number of test cases that completed successfully in the suite.
- *TestsFailed* - the number of test cases that failed in the suite.

In this quickstart, you'll create two custom tables in the Dataverse database to store the test results by customizing the **OnTestCaseComplete** and **OnTestSuiteComplete** properties:

1. Select **Test** in the left pane or **View** on the suite header.



2. Select the **OnTestCaseComplete** action.

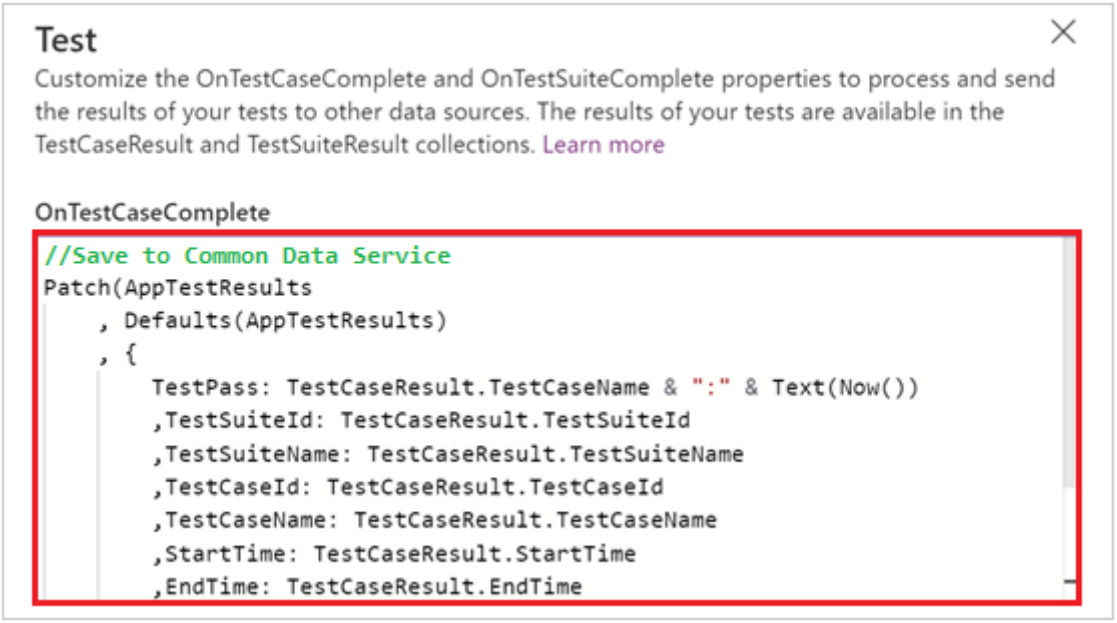
3. Input an expression to process the results of your test. The following sample saves each test case's results to the custom AppTestResults table in Dataverse. The test results can optionally be stored to SQL, SharePoint, or any other data source. You might need to set or increase the Trace field in your data source as required.

> ⓘ **Note**

The following samples connect to Microsoft Dataverse. You can create a simple app or build an app from scratch using Dataverse. Also, refer to the Patch function reference for more details to modify records of a data source used in the following samples.

Power Fx

```
//Save to Dataverse
Patch(AppTestResults
, Defaults(AppTestResults)
, {
        TestPass: TestCaseResult.TestCaseName & ":" & Text(Now())
        ,TestSuiteId: TestCaseResult.TestSuiteId
        ,TestSuiteName: TestCaseResult.TestSuiteName
        ,TestCaseId: TestCaseResult.TestCaseId
        ,TestCaseName: TestCaseResult.TestCaseName
        ,StartTime: TestCaseResult.StartTime
        ,EndTime: TestCaseResult.EndTime
        ,TestSuccess: TestCaseResult.Success
        ,TestTraces: JSON(TestCaseResult.Traces)
        ,TestFailureMessage: TestCaseResult.TestFailureMessage
}
);
```

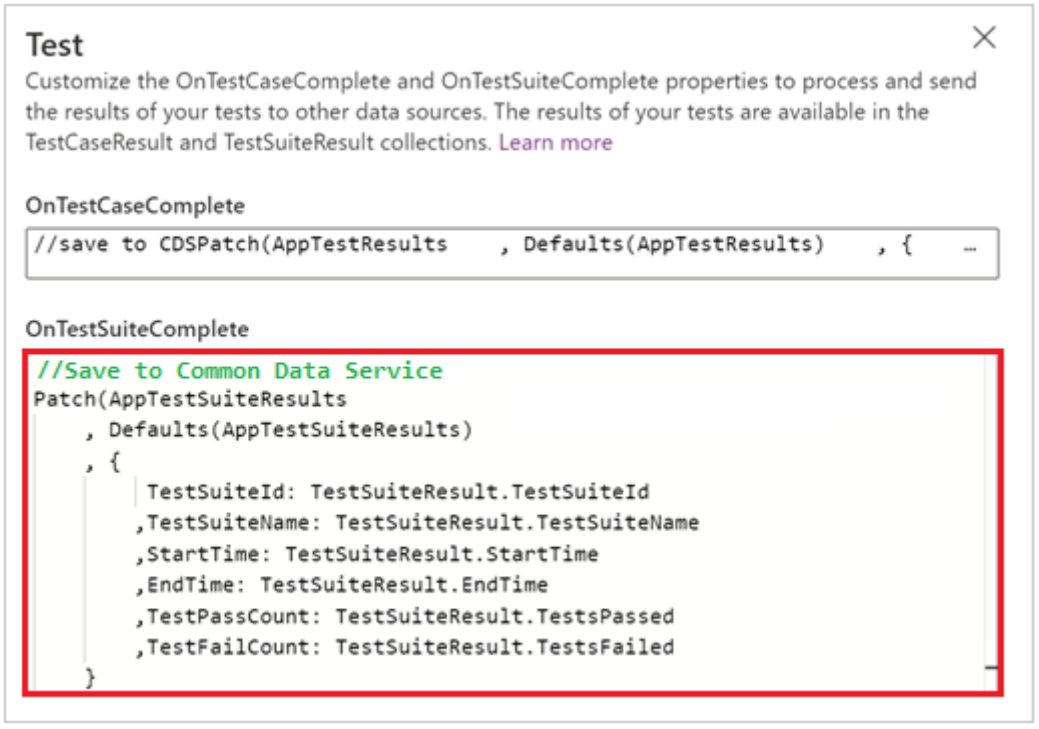


4. Select the **OnTestSuiteComplete** action.

5. Input an expression to process the results of your test. In the following sample, you'll save each test suite's results to the custom AppTestSuiteResults table in Dataverse.

Power Fx

```
//Save to Dataverse
Patch(AppTestSuiteResults
    , Defaults(AppTestSuiteResults)
    , {
        TestSuiteId: TestSuiteResult.TestSuiteId
        ,TestSuiteName: TestSuiteResult.TestSuiteName
        ,StartTime: TestSuiteResult.StartTime
        ,EndTime: TestSuiteResult.EndTime
        ,TestPassCount: TestSuiteResult.TestsPassed
        ,TestFailCount: TestSuiteResult.TestsFailed
    }
);
```



Other examples of expressions you could use in these properties are:

- Send results to a flow in Power Automate.

```
Power Fx

MyTestResultsFlow.Run(JSON(TestCaseResult))
```

- Email your results.

```
Power Fx

Office365.SendMailV2("someone@example.com", "Test case results",
JSON(TestCaseResult, JSONFormat.IndentFour))
```

- Receive an app notification of the test result.

For example, receive a notification after the test completes when playing the test in a browser, outside of Test Studio.

Power Fx

```
Notify(TestCaseResult.TestCaseName & " : "
        & If( TestCaseResult.Success
            , " Passed"
            , TestCaseResult.TestFailureMessage)
        ,If(  TestCaseResult.Success
            , NotificationType.Success
            , NotificationType.Error)
)
```

Test Functions

In addition to the functions available in Power Apps, the following are common functions that you'll typically use when authoring tests:

- Select
- SetProperty
- Assert
- Trace

Next steps

- Automate tests with Azure Pipelines using classic editor

Feedback

Was this page helpful? Yes No

Provide product feedback

Automate tests with Azure Pipelines using classic editor

Article • 12/16/2022

In this article, you'll learn how to set up and run your canvas app tests built in Test Studio using the Azure Pipelines classic editor in Azure DevOps Services.

You can use a public project on GitHub - Microsoft/PowerAppsTestAutomation to:

- Automate operations of signing in to your application.
- Open a browser on the build agent and execute a set of test cases and suites.
- View the status of the test execution in the Azure DevOps pipeline.

Prerequisites

Before you begin, you must complete the following steps:

- Fork the Microsoft/PowerAppsTestAutomation project on GitHub.

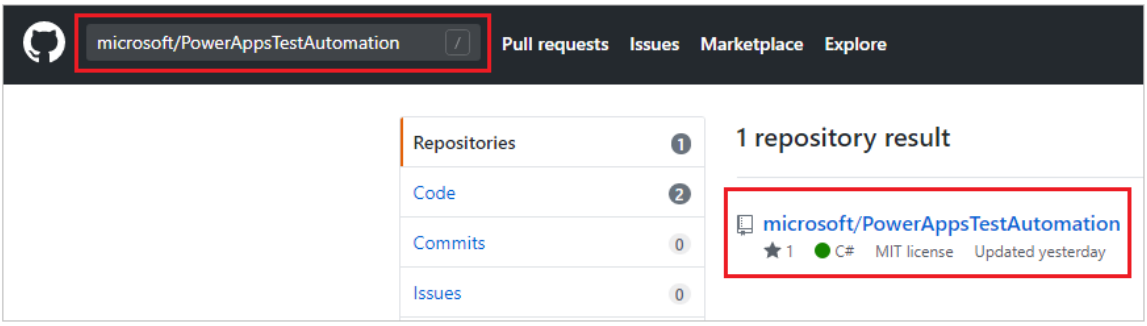
  > **Note**
  >
  > Public forks can't be made private. If you want to create a private repo, please duplicate the repository.

- Create a new *Test URL .json file* in the repo with the App Test URLs you want to run from the pipeline.

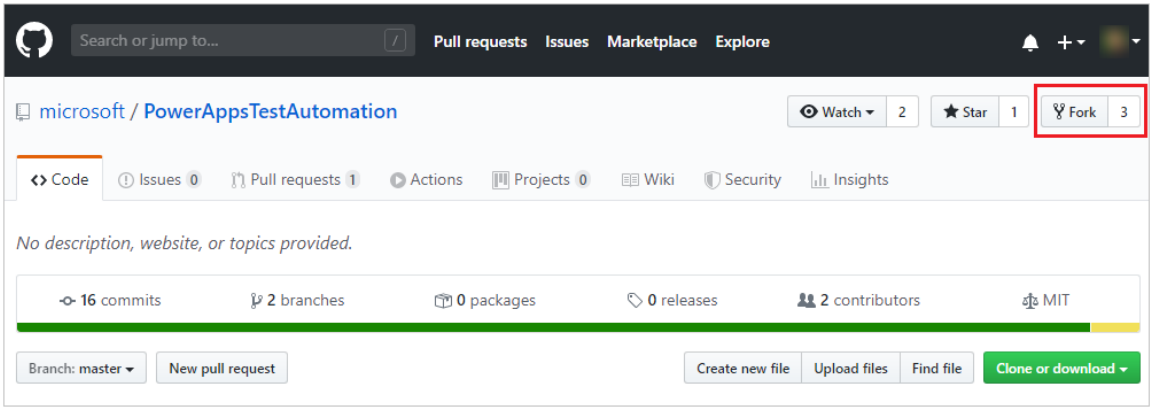
Step 1 - Fork the PowerAppsTestAutomation project

A fork is a copy of a repository. By forking a repository, you can make changes without affecting the original project.

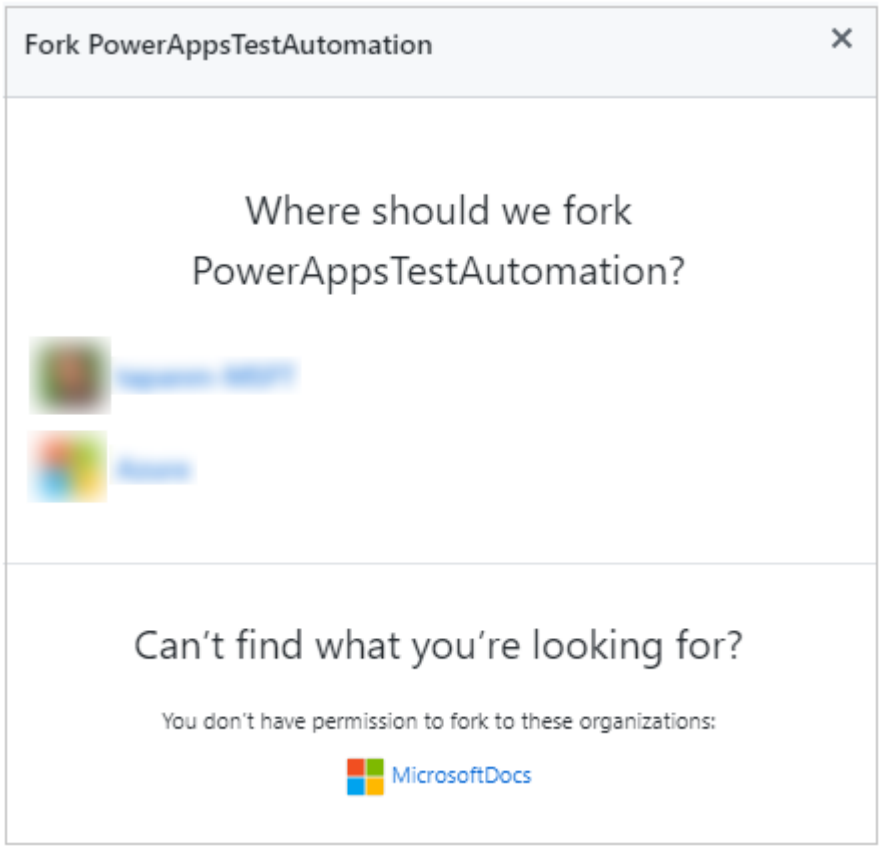
1. Sign in to GitHub.
2. Go to the microsoft/PowerAppsTestAutomation repository. You can also search for **microsoft/PowerAppsTestAutomation** instead, and then select the repository:



3. Select **Fork**:



4. Select where you want to fork:



Your forked repository will now be available.

Step 2 - Create Test URL .json file

The Test URL .json file will contain the test suite and test case URLs for validating your app. The app test suite and test case URLs can be retrieved by selecting the Copy play link in Test Studio.

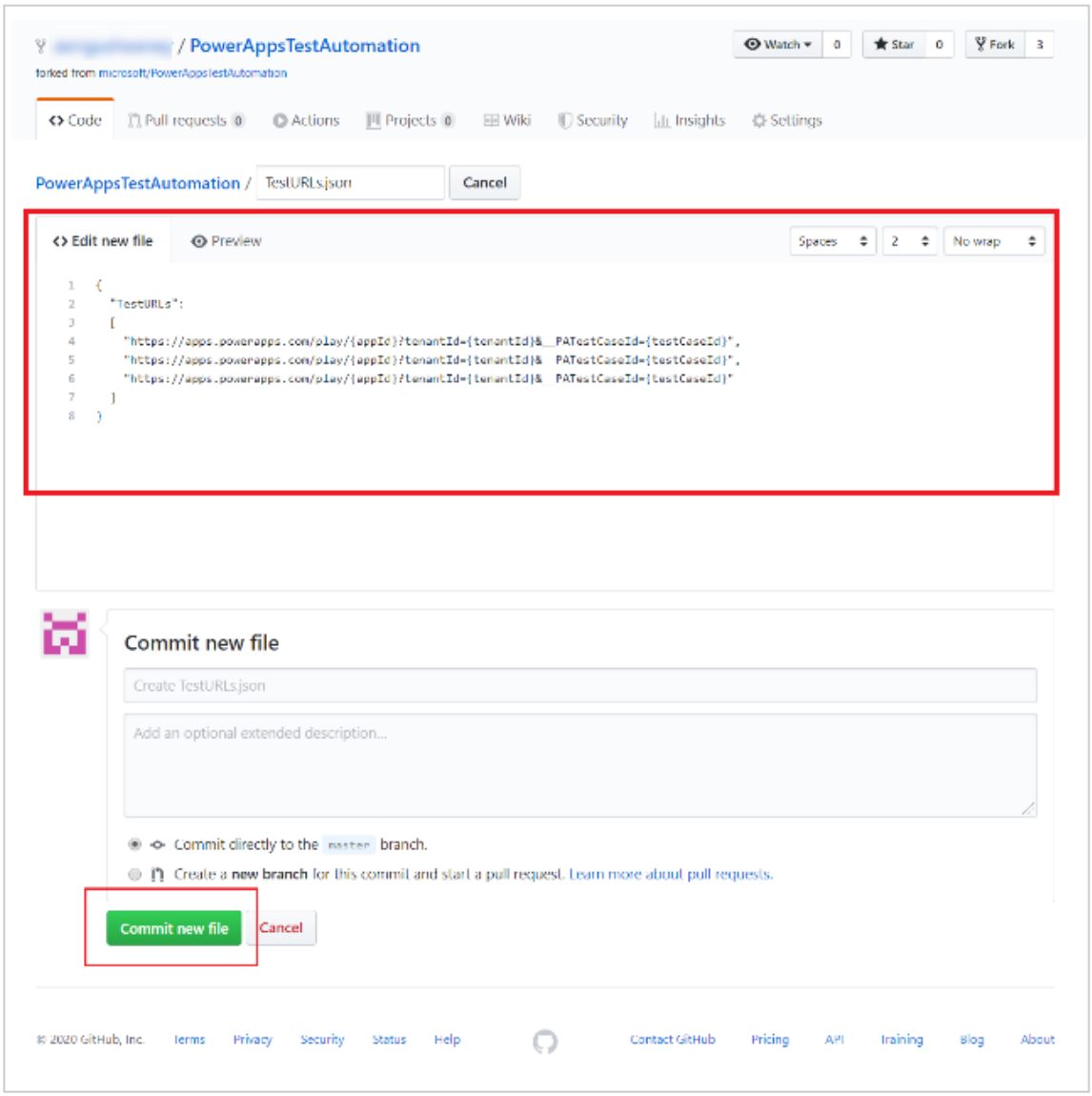
You can find a sample file `Samples/TestAutomationURLs.json` in the repo you created earlier.

1. Create a new `TestURLs.json` file in your repo, or use any other file name.
   The file name and location will be mapped in the pipeline variables later in the document.

2. Copy the format from the `Samples/TestAutomationURLs.json` file.

3. Update the Test URLs section with the tests that you want to validate in your app.

4. Commit the changes to your repo:



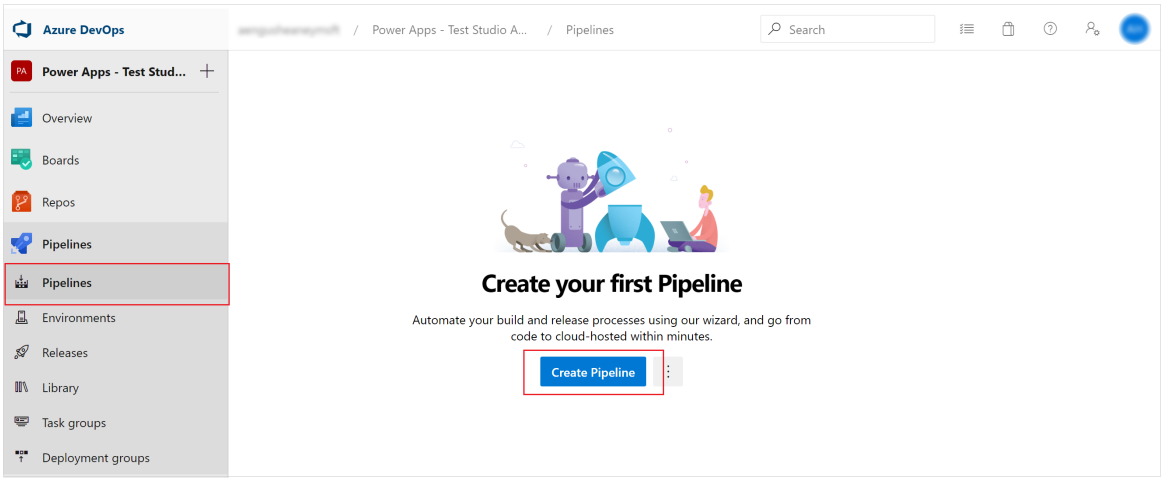
Create a pipeline

1. Sign in to your Azure DevOps instance.

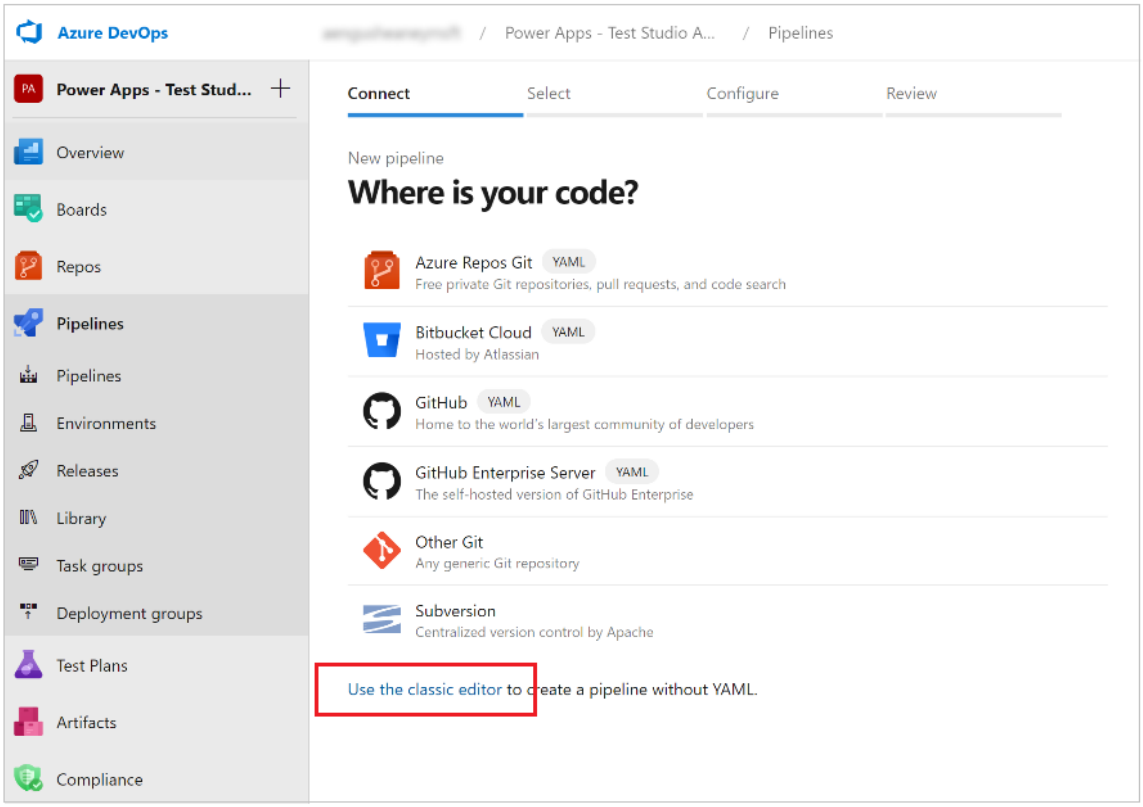
2. Select an existing project or create a new project.

3. Select **Pipelines** in the left menu.

4. Select **Create Pipeline**:

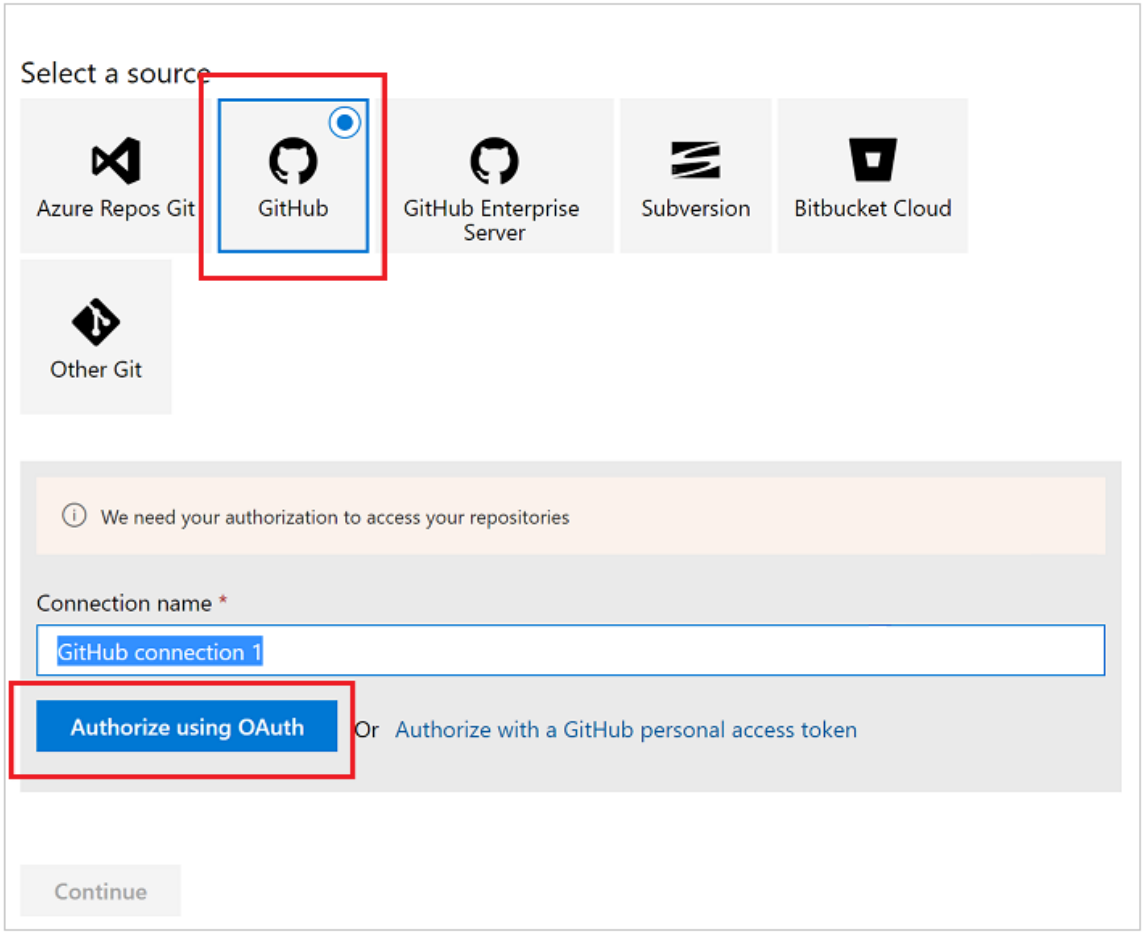


5. Select **Use the classic editor**:



6. Select GitHub as the source.

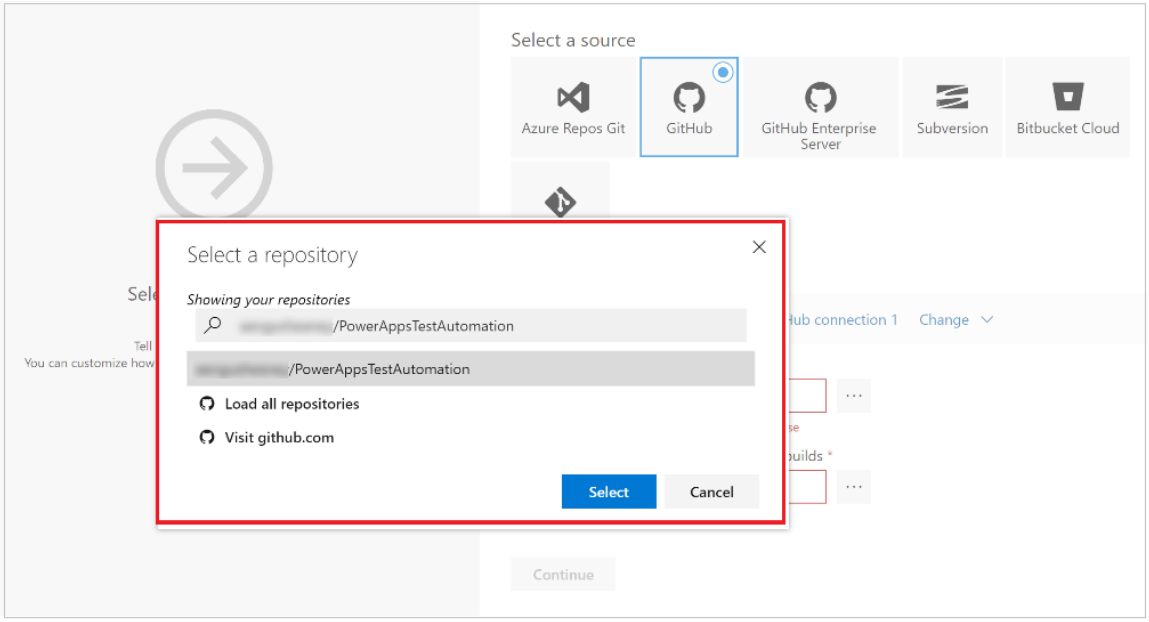
7. If necessary, authorize your GitHub connection using OAuth or using a personal access token:



8. If needed, edit the connection name.

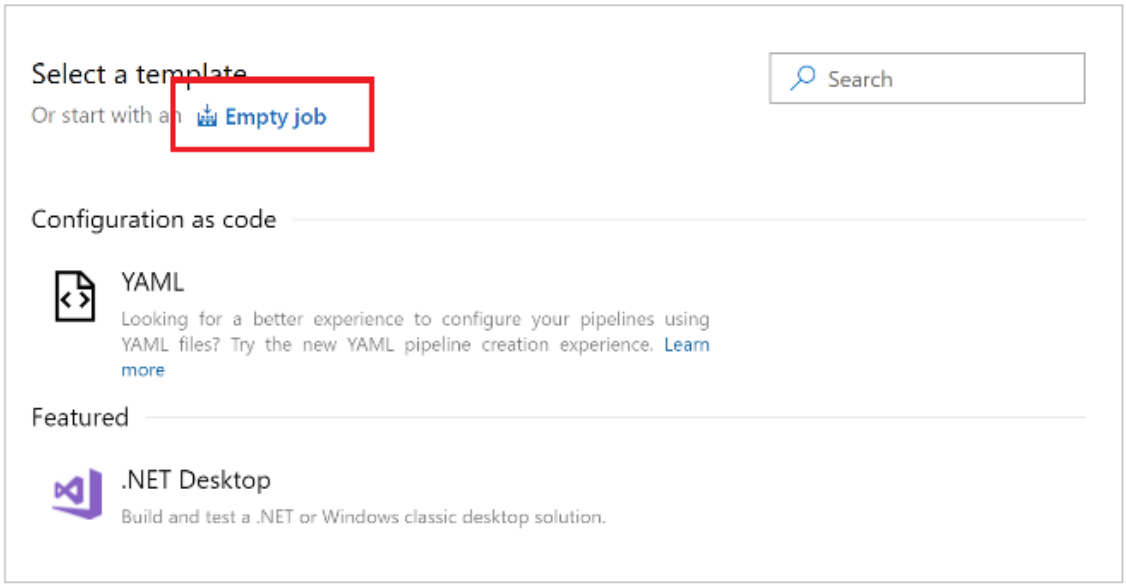
9. Select **...** (ellipsis) from the right side of **Repository** input.

10. Enter the name of your project on GitHub, and then **Select** it:



11. Select **Continue**.

12. In the Select a template screen, select **Empty job**:



13. **Save** your pipeline.

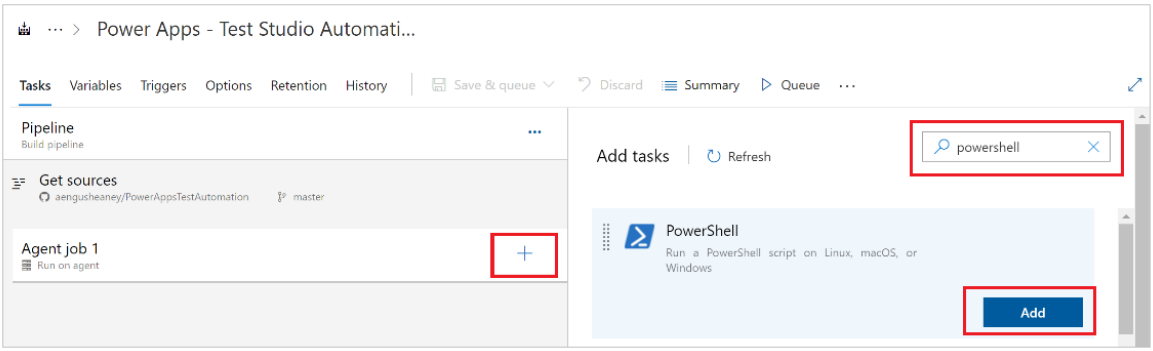
Add tasks to the pipeline

You'll now add new job tasks and configure the tasks to run the tests from the pipeline in this sequence:

1. Configure screen resolution using PowerShell.
2. Restore NuGet packages for the PowerAppsTestAutomation solution.
3. Build the PowerAppsTestAutomation solution.
4. Add Visual Studio Tests for Google Chrome.
5. Add Visual Studio Tests for Mozilla Firefox.

Step 1 - Configure screen resolution using PowerShell

1. Select + next to *Agent job 1*.
2. Search for **PowerShell**.
3. Select **Add** to add a PowerShell task to the job:

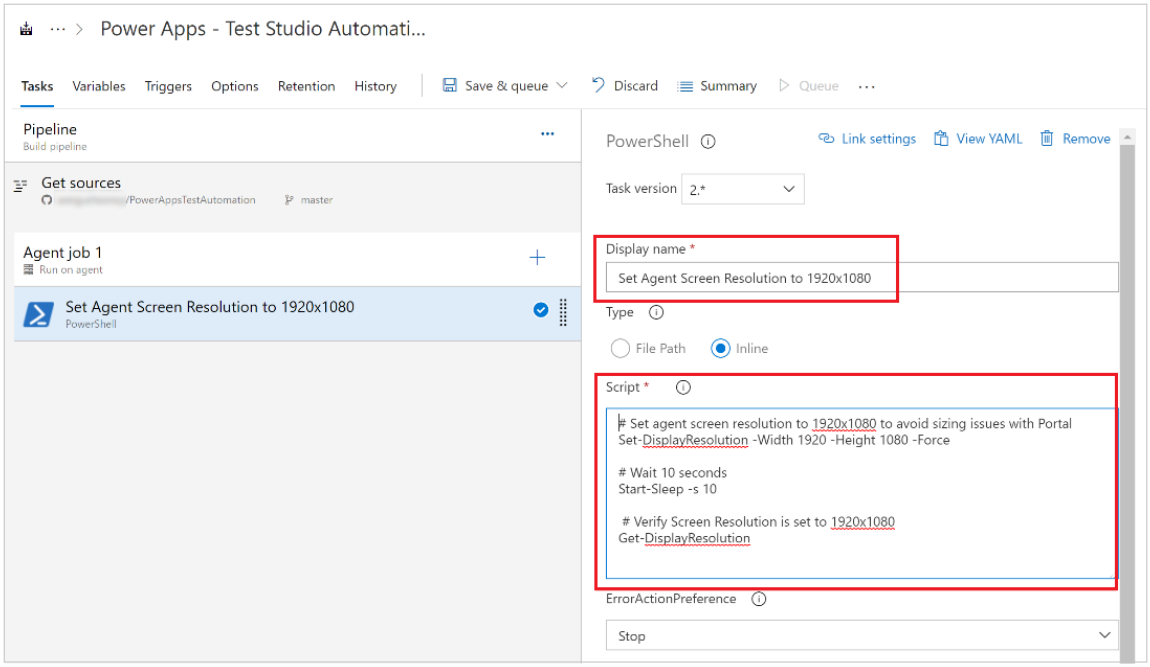


4. Select the task.

   You can also update the display name to *Set Agent Screen Resolution to 1920 x 1080* or similar.

5. Select **Inline** as the script type, and enter the following in the script window:

```PowerShell
# Set agent screen resolution to 1920x1080 to avoid sizing issues with Portal
Set-DisplayResolution -Width 1920 -Height 1080 -Force
# Wait 10 seconds
Start-Sleep -s 10
# Verify Screen Resolution is set to 1920x1080
Get-DisplayResolution
```



Step 2 - Restore NuGet packages

1. Select + next to *Agent job 1*.

2. Search for **NuGet**.

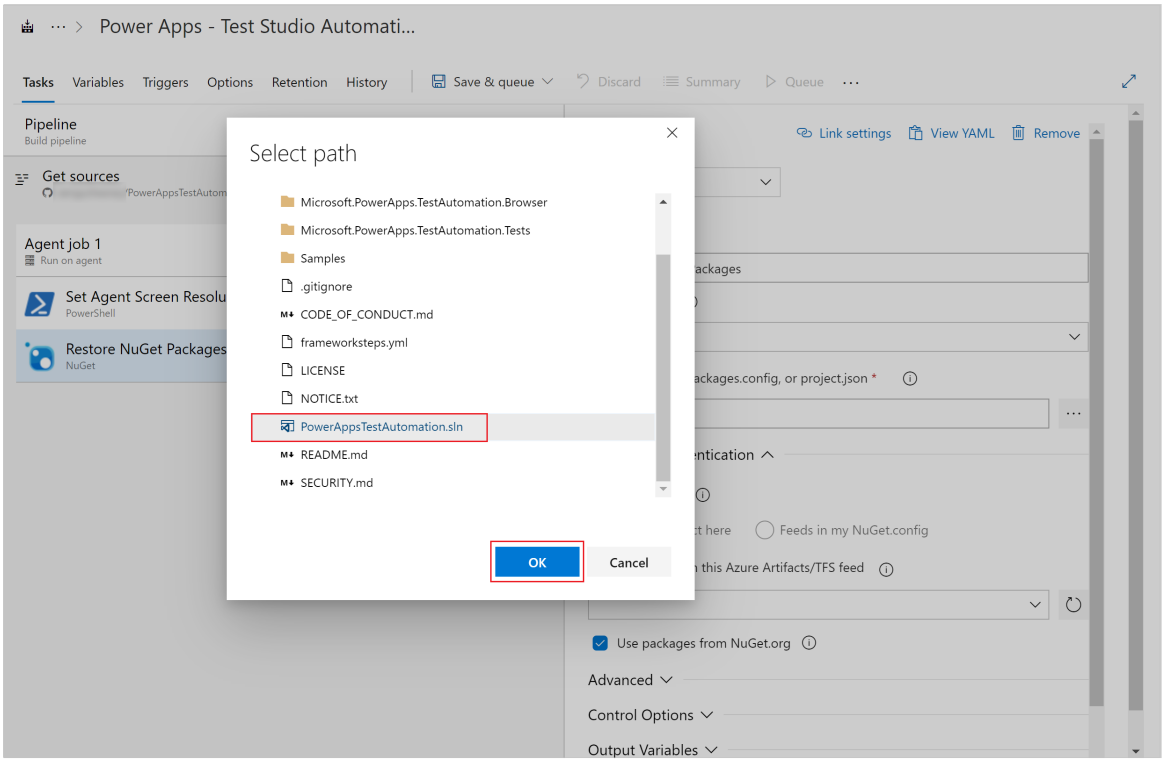
3. Select **Add** to add a NuGet task to the job.

4. Select the task.
   You can also update the display name to **Restore NuGet Packages** or similar.

5. Select **...** (ellipsis) in the **Path to solution, packages.config, or project.json** configuration field.

6. Select the PowerAppsTestAutomation.sln solution file.

7. Select **OK**:



Step 3 - Build the PowerAppsTestAutomation solution

1. Select + next to *Agent job 1*.

2. Search for **Visual Studio build**.

3. Select **Add** to add a Visual Studio build task to the job.

4. Select the task.
   You can also update the display name to **Build Power Apps Test Automation Solution** or similar.

5. Select **...** (ellipsis) in the **Solution** configuration field.

6. Select the PowerAppsTestAutomation.sln solution file.

7. Select **OK**.

Step 4 - Add Visual Studio Tests for Google Chrome

1. Select + next to *Agent job 1*.

2. Search for **Visual Studio Test**.

3. Select **Add** to add a Visual Studio Test task to the job.

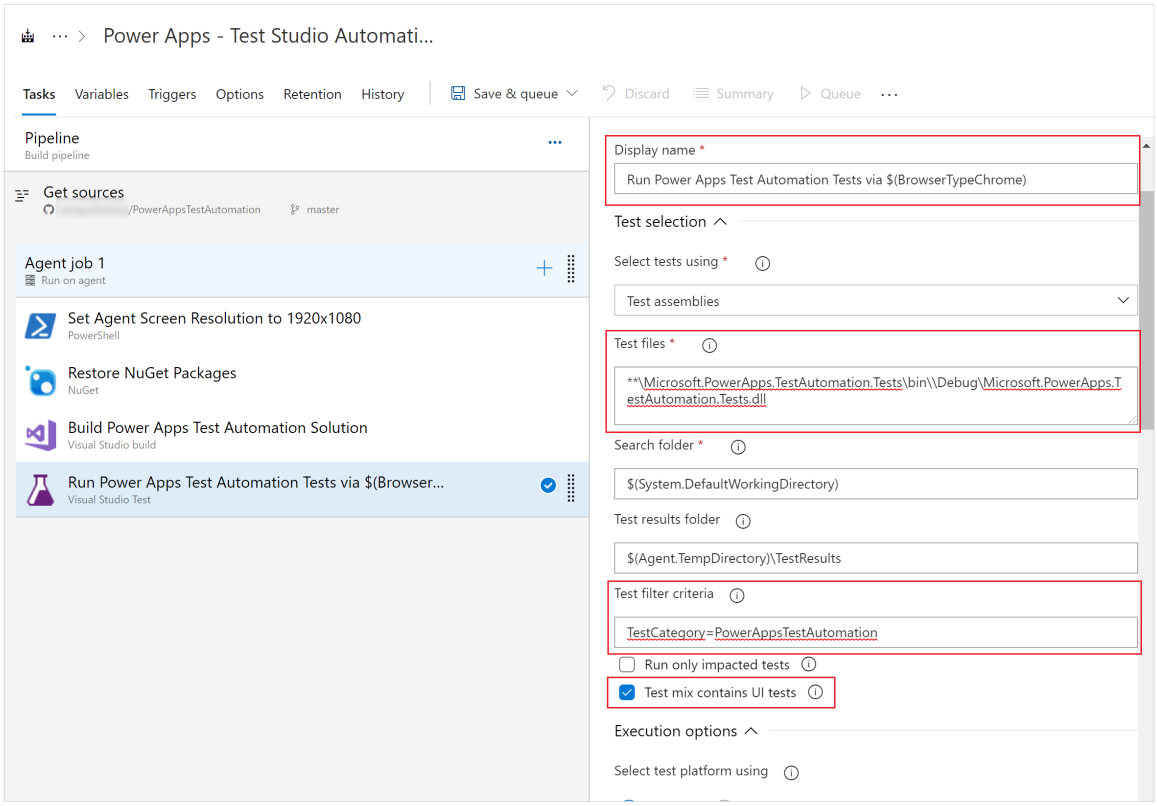
4. Select the task.
   You can also update the display name to **Run Power Apps Test Automation Tests via $(BrowserTypeChrome)** or similar.

5. Remove the default entries in the **Test files** text field and add the following:

   `**\Microsoft.PowerApps.TestAutomation.Tests\bin\\Debug\Microsoft.PowerApps.TestAutomation.Tests.dll`

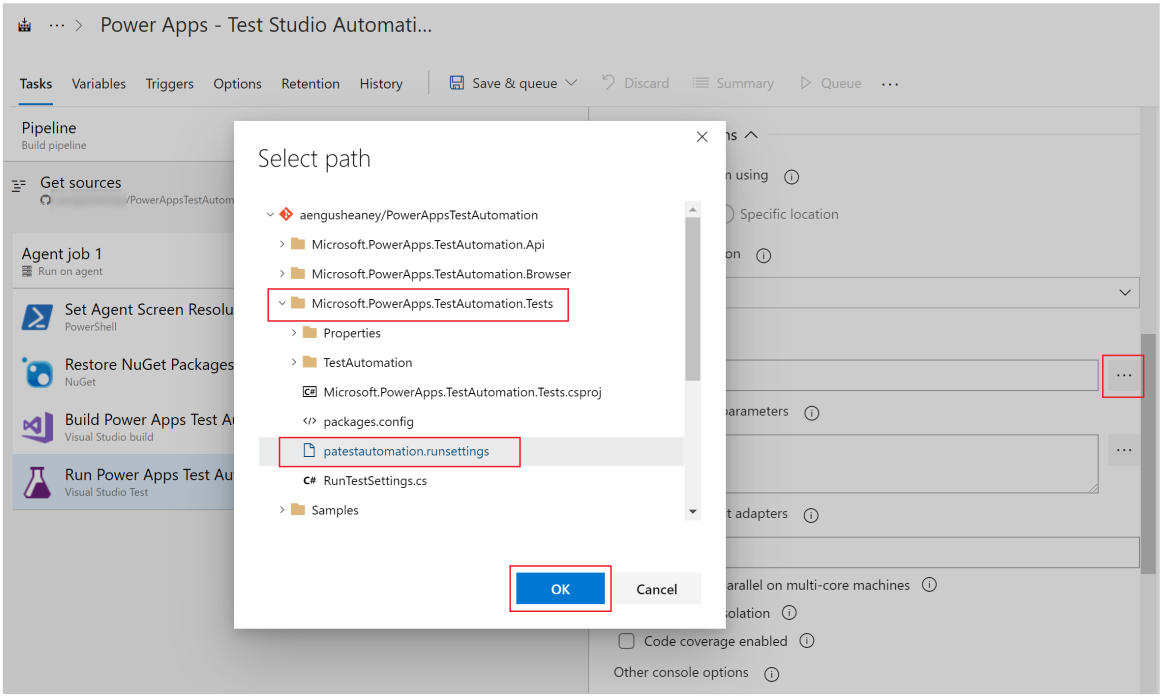
6. Enter `TestCategory=PowerAppsTestAutomation` in the **Test filter criteria** field.

7. Select **Test mix contains UI tests**.



8. Select ... (ellipsis) in the **Settings file** field.

9. Expand the **Microsoft.PowerApps.TestAutomation.Tests**, select the **patestautomation.runsettings** file, and then select **OK**:

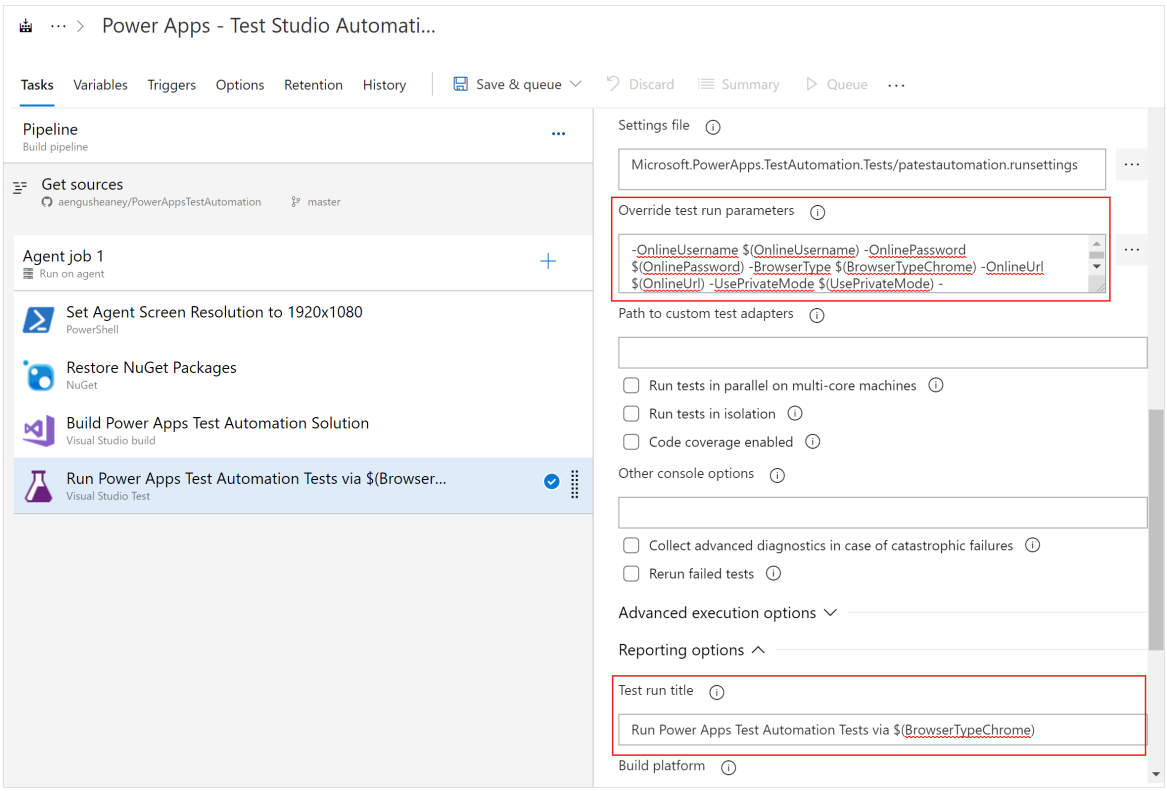


10. Copy the following in the **Override test run parameters** field.

```
-OnlineUsername "$(OnlineUsername)" -OnlinePassword "$(OnlinePassword)"
-BrowserType "$(BrowserTypeChrome)" -OnlineUrl "$(OnlineUrl)" -
UsePrivateMode "$(UsePrivateMode)" -TestAutomationURLFilePath
"$(TestAutomationURLFilePath)" -DriversPath "$(ChromeWebDriver)"
```

> **Note**
>
> This is where the variables in the pipeline are configured, represented above in the form of $(VariableName).

11. Enter **Run Power Apps Test Automation Tests via $(BrowserTypeChrome)** or similar in the **Test run title** field.



Step 5 - Add Visual Studio Tests for Mozilla Firefox

1. Right-click the **Add Visual Studio Tests for Chrome** task and select **Clone task(s)**.

2. Select the task and update the following areas:

   a. **Title**: Run Power Apps Test Automation Tests via $(BrowserTypeFirefox)

   b. **Override test run parameters**

```
-OnlineUsername "$(OnlineUsername)" -OnlinePassword
"$(OnlinePassword)" -BrowserType "$(BrowserTypeFirefox)" -OnlineUrl
"$(OnlineUrl)" -UsePrivateMode "$(UsePrivateMode)" -
TestAutomationURLFilePath "$(TestAutomationURLFilePath)" -
DriversPath "$(GeckoWebDriver)"
```

   c. **Test Run Title**: Run Power Apps Test Automation Tests via $(BrowserTypeFirefox)

Configure pipeline variables

You'll now configure the pipeline variables defined in the tasks you've added earlier.

1. Select the **Variables** tab.

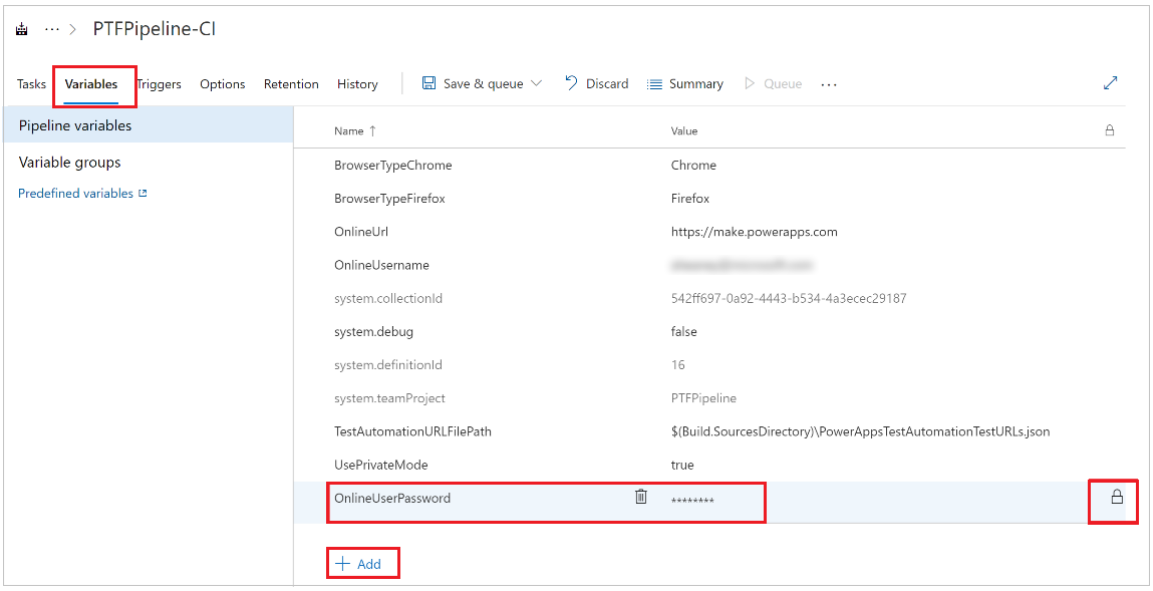
2. Select **Add** and repeat this step to configure the following variables:

Expand table

Variable name	Variable value
BrowserTypeChrome	Chrome
BrowserTypeFirefox	Firefox
OnlineUrl	https://make.powerapps.com
TestAutomationURLFilePath	`$(Build.SourcesDirectory)\<test URL file>.json` **Note:** This is the *Test URLs .json* file you created earlier.
UsePrivateMode	true
OnlineUsername	Enter the Microsoft Entra email address of the user context that will sign in to the application. Tests will run under the context of this user account. Ensure this user has the ability to run the app and connect to any data sources the app might connect to.

1. Select **Add** and enter **OnlinePassword** in the variable name.

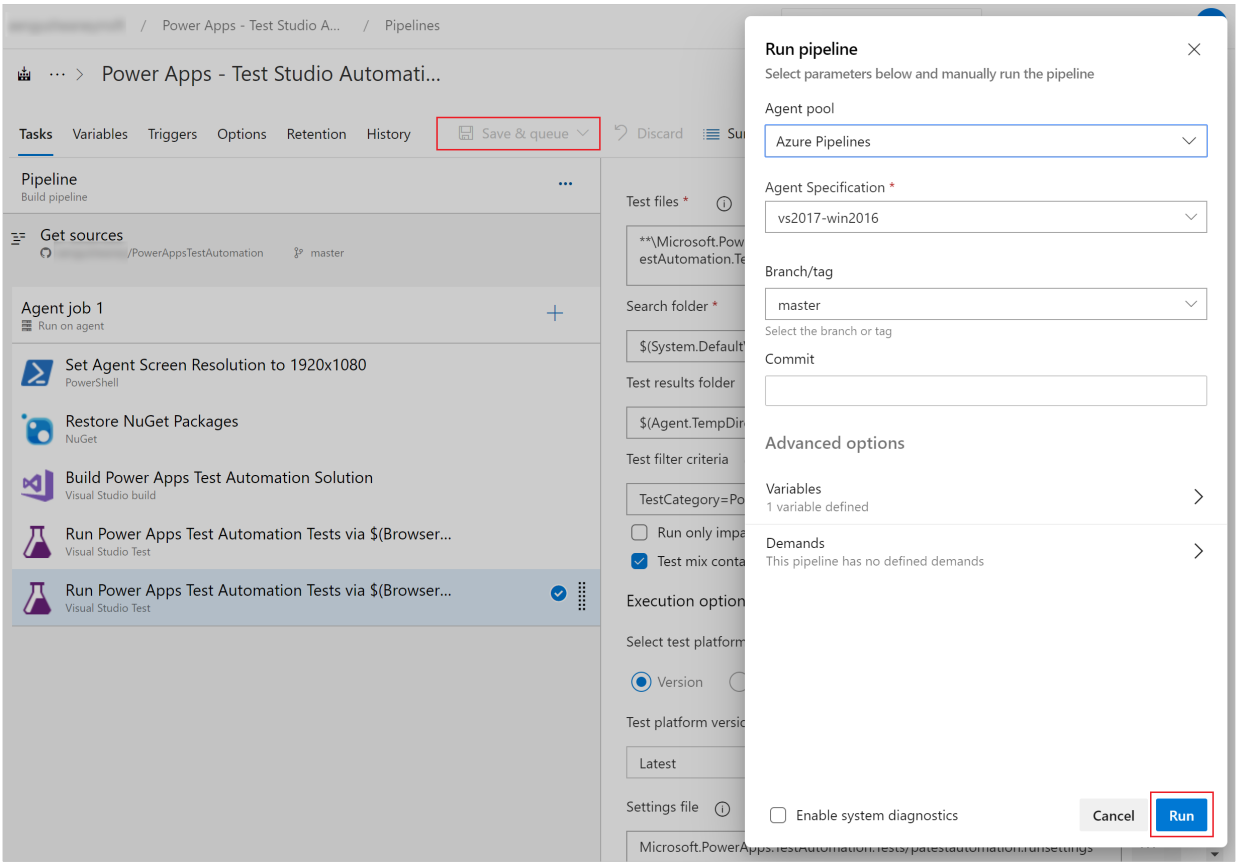
2. Check the lock image to make this variable a secret.



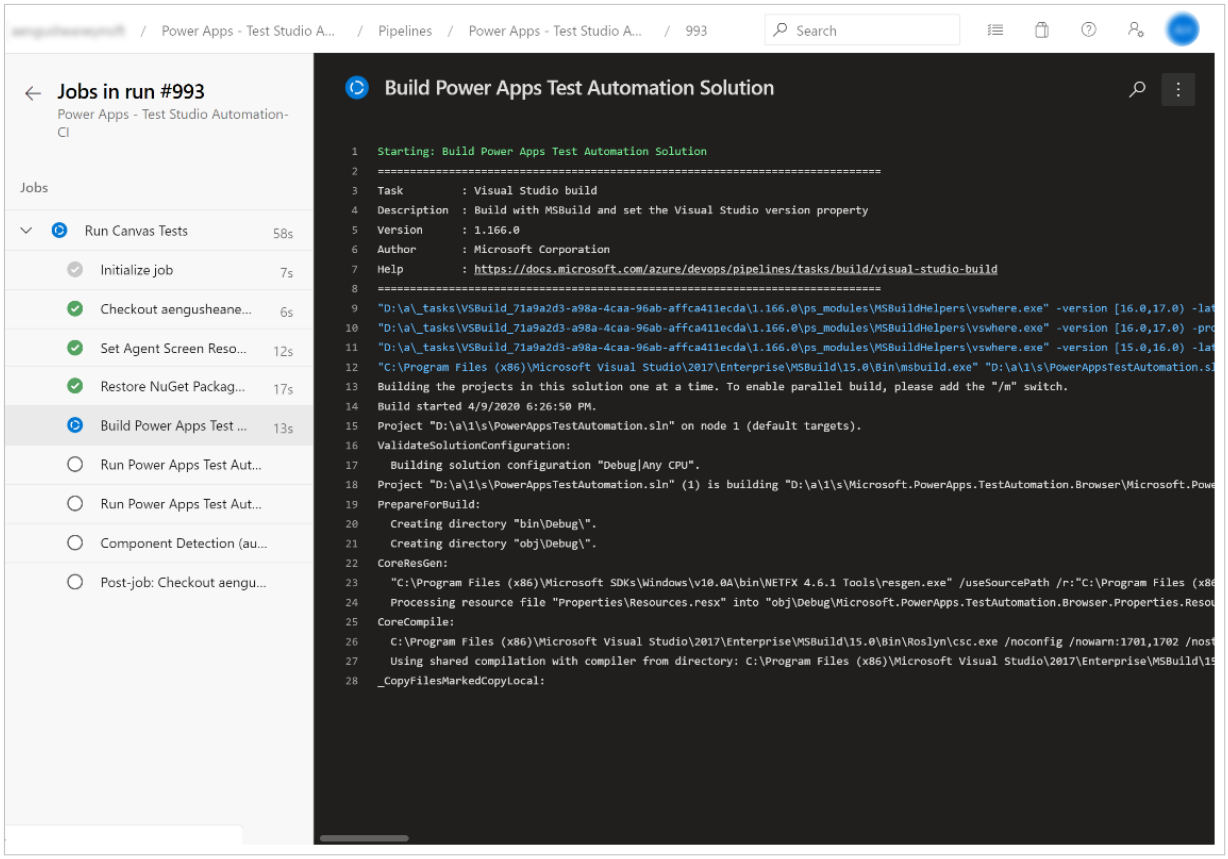
3. **Save** your pipeline configurations.

Run and analyze tests

To validate that your tests are executing successfully, select **Queue** and then select **Run**. Your job will start running.

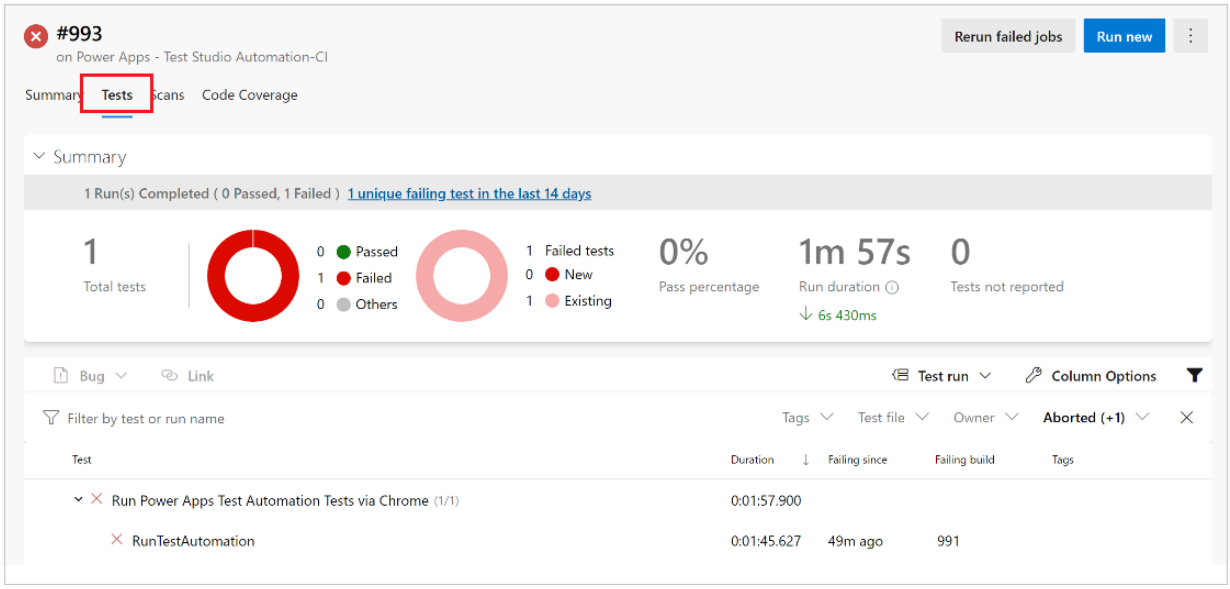


As the job runs, select the job to see a detailed status on each of the tasks running:

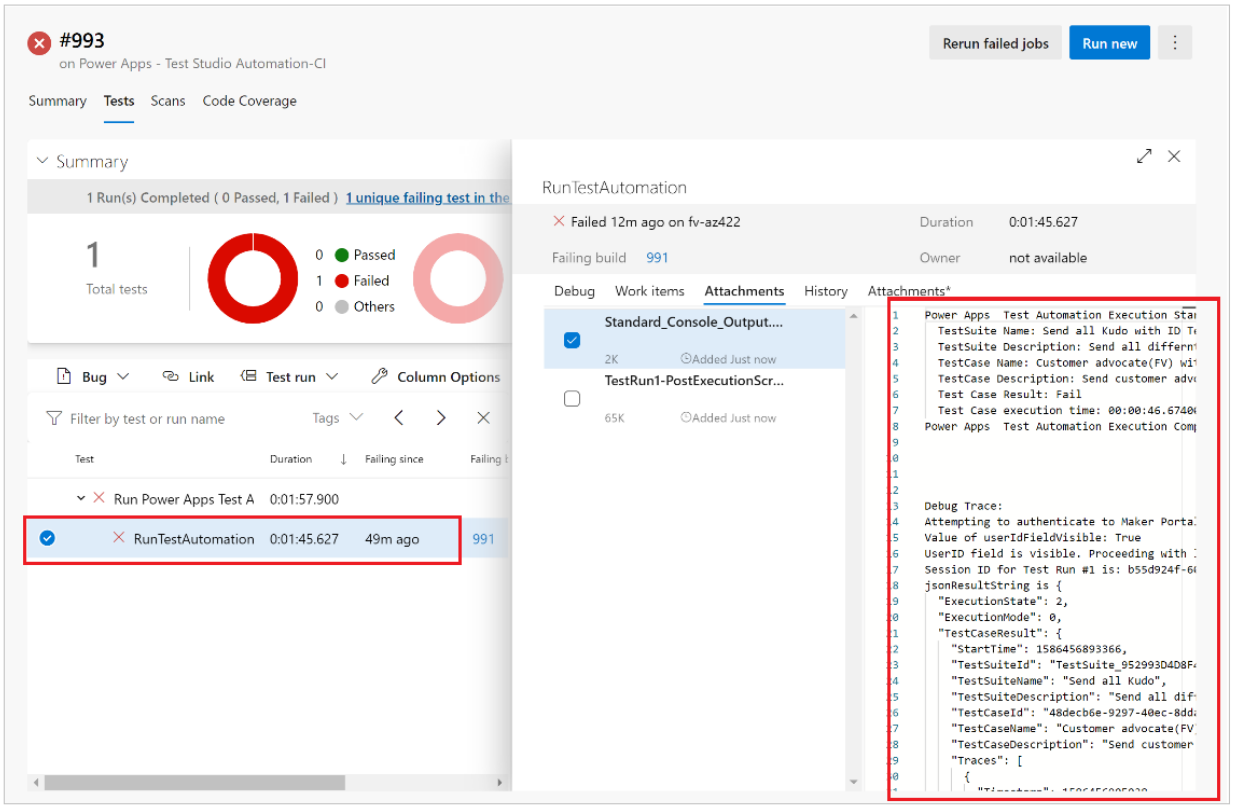


When the job completes, you can view the high-level job summary, and any errors or warnings. By selecting the **Tests** tab, you can view specific details on the test cases you've executed.

The following example indicates at least one of our test cases has failed while executing the tests using the Chrome browser:



Select **RunTestAutomation** test to drill into the details on what test case has failed. In the **Attachments** tab, you can see the summary of the test execution and which test cases have failed or passed in your test suite:



> **Note**
>
> If you execute a test suite, you'll see a summary of test cases passed and failed. If you execute a test case, you'll see specific details on the failure with any trace

information, if available.

Known limitations

- Multifactor authentication isn't supported.
- Test summary will report a single test result per browser. The test result will contain one or more test cases or test suite results.
- Any authentication process other tha Microsoft Entra sign-in flow requires customization of the sign-in process in the **PowerAppsTestAutomation** solution.

See also

- Test Studio overview
- Working with Test Studio
- Configure pipeline using YAML

Automate tests with Azure Pipelines using YAML

Article • 12/16/2022

In this article, you'll learn how to set up and run your canvas app tests built in the Test Studio by using a YAML pipeline in Azure DevOps Services.

You can use a public project on GitHub—Microsoft/PowerAppsTestAutomation —to:

- Automate the operations of signing in to your application.
- Open a browser on the build agent and execute a set of test cases and suites.
- View the status of the test execution in the Azure DevOps pipeline.

Prerequisites

Before you begin, you must complete the following steps:

- Fork the Microsoft/PowerAppsTestAutomation project on GitHub.

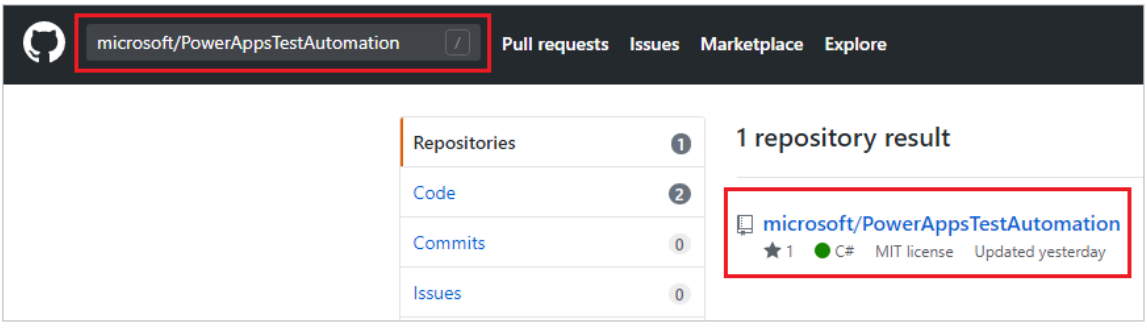
  > **Note**
  >
  > Public forks can't be made private. If you want to create a private repo, you need to duplicate the repository.

- Create a new Test URLs .json file that includes the app test URLs you want to run from the pipeline.
- Create a new Azure Pipelines YAML file.
- Create a GitHub service connection to your repo.

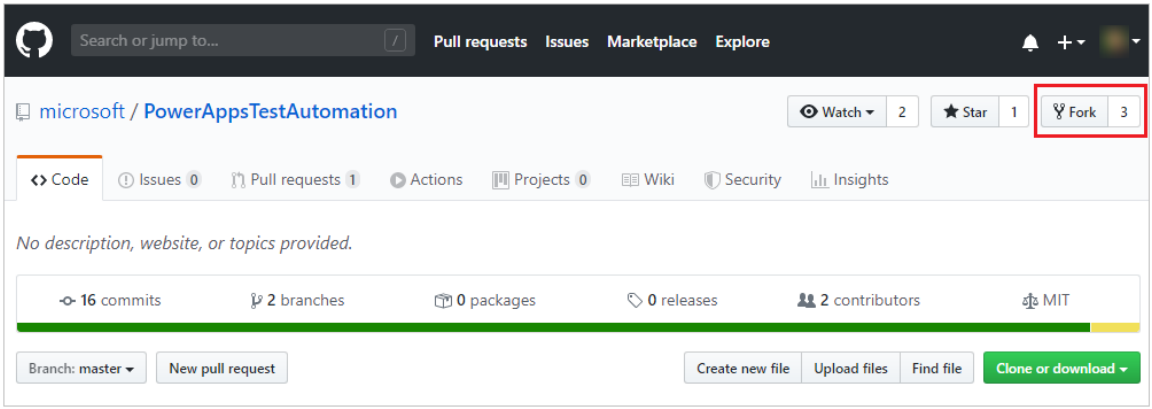
Step 1 - Fork the PowerAppsTestAutomation project

A *fork* is a copy of a repository. By forking a repository, you can make changes without affecting the original project.

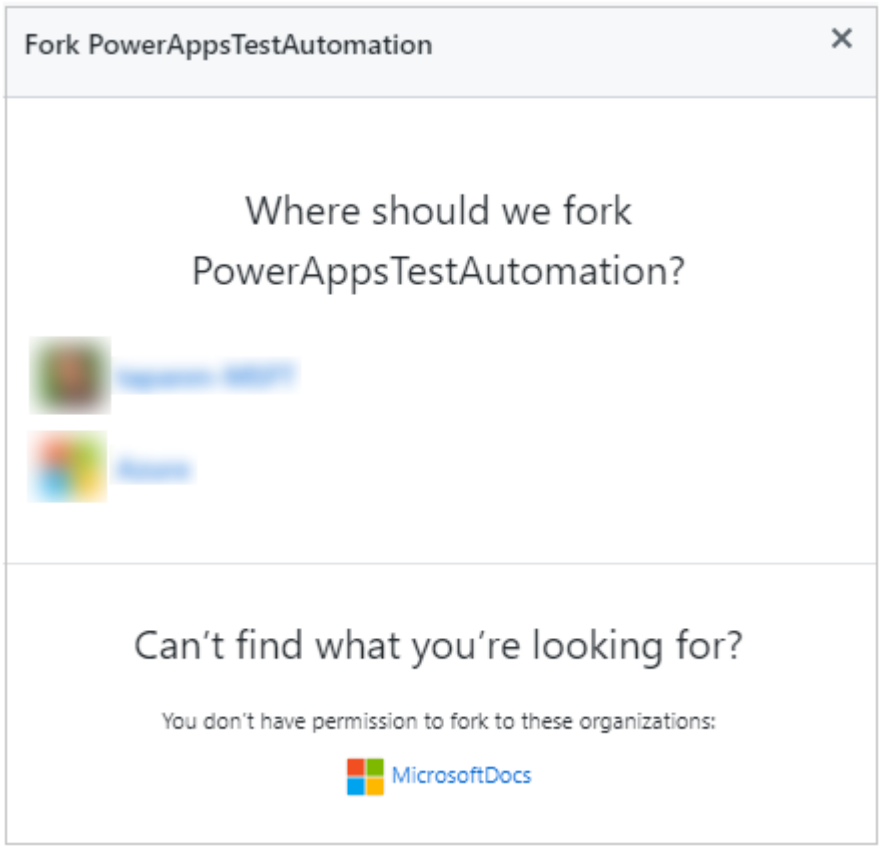
1. Sign in to GitHub.
2. Go to the microsoft/PowerAppsTestAutomation repository. You can also search for **microsoft/PowerAppsTestAutomation** instead, and then select the repository.



3. Select **Fork**.



4. Select where you want to locate the fork.



Your forked repository will now be available.

Step 2 - Create a test URL .json file

The test URL .json file will contain the test suite and test case URLs for validating your app. The app test suite and test case URLs can be retrieved by selecting the copy play link in Test Studio.

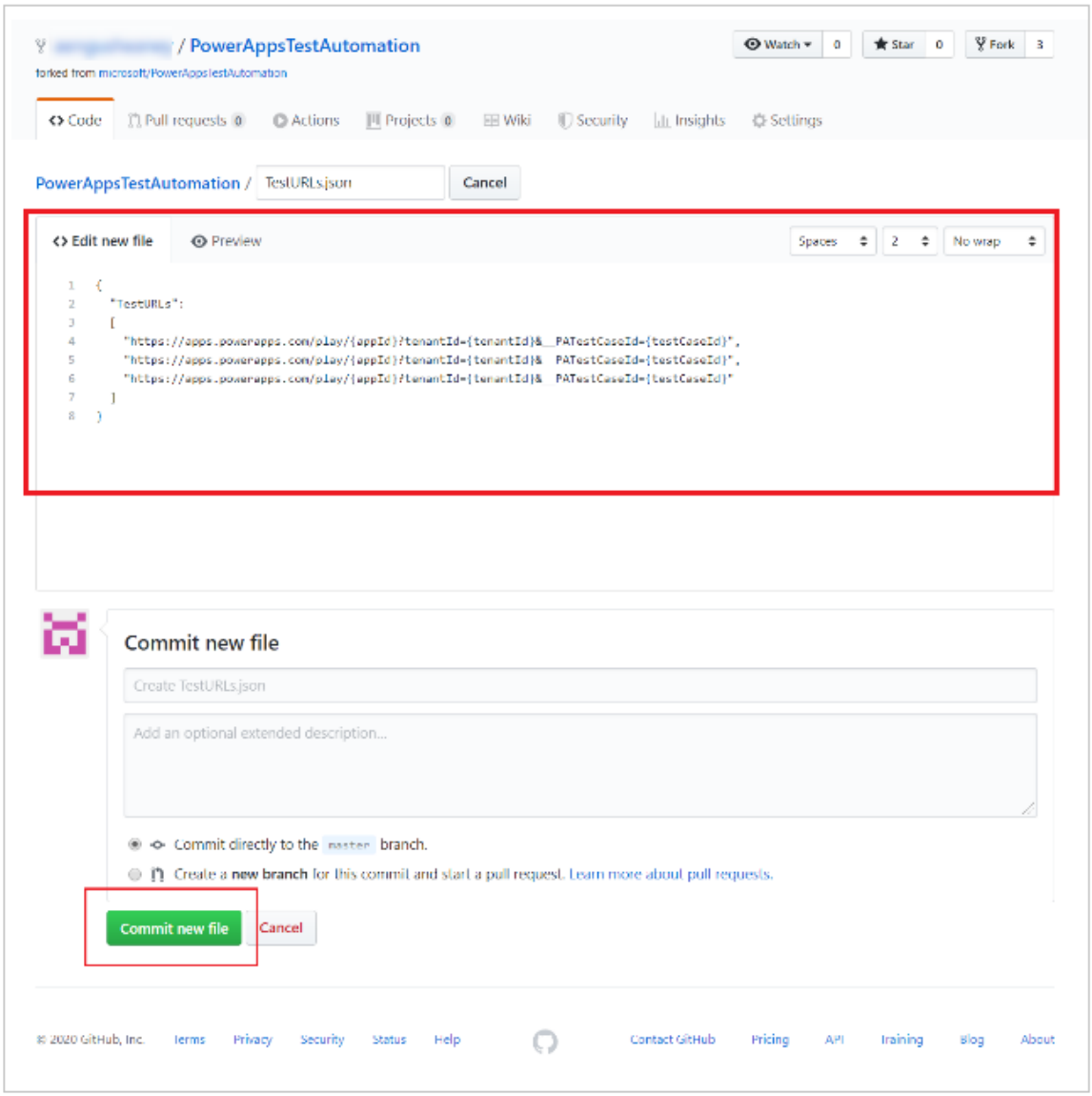
You can find a sample file, Samples/TestAutomationURLs.json, in the repo you created earlier.

1. Create a new file named TestURLs.json file in your repo, or use any file name you want. The file name and location will be mapped in the pipeline variables later in the document.

2. Copy the format from the Samples/TestAutomationURLs.json file.

3. Update the **Test URLs** section with the tests that you want to validate in your app.

4. Commit the changes to your repo.



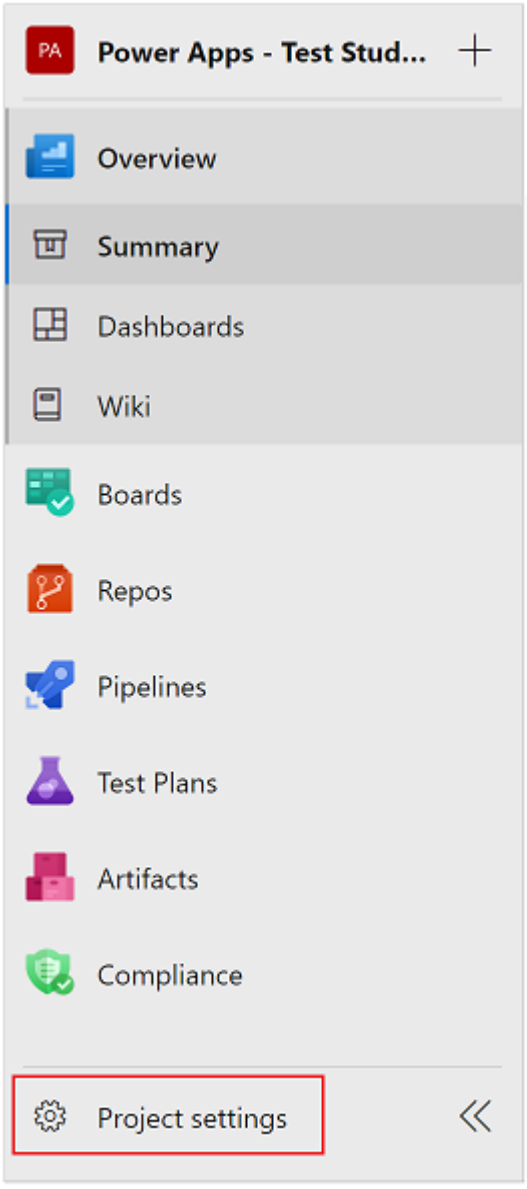
Step 3 - Create an Azure Pipelines YAML file

You can find a sample file, Samples/azure-pipelines.yml, in the repo you created earlier.

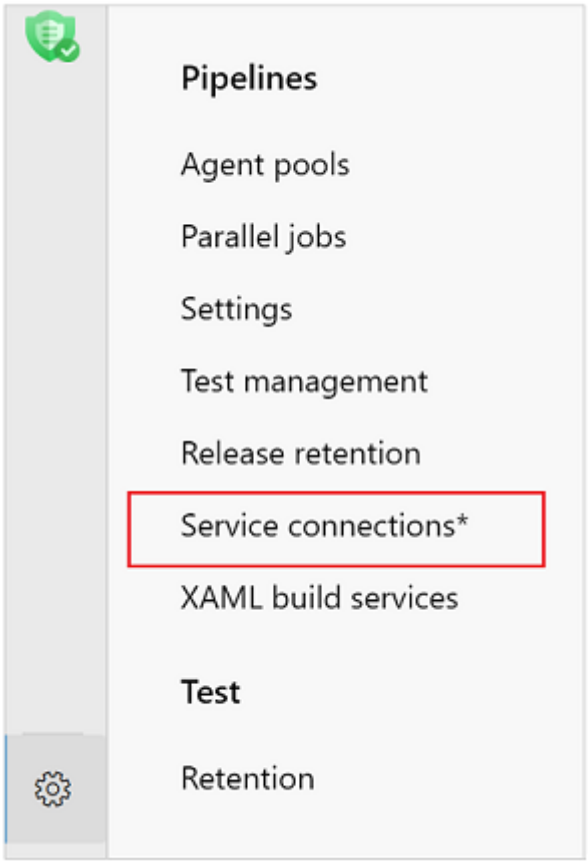
1. Create a new file named azure-pipelines.yml file in your repo.
2. Copy the content from the Samples/azure-pipelines.yml file.
3. Commit the changes to your repo. You'll reference and update the azure-pipelines.yml file when you're configuring your pipeline in Step 4.

Step 4 - Create a GitHub service connection

1. Sign in to your Azure DevOps instance.
2. Select an existing project, or create a new project.
3. Select **Project settings** at the bottom of the left navigation pane.



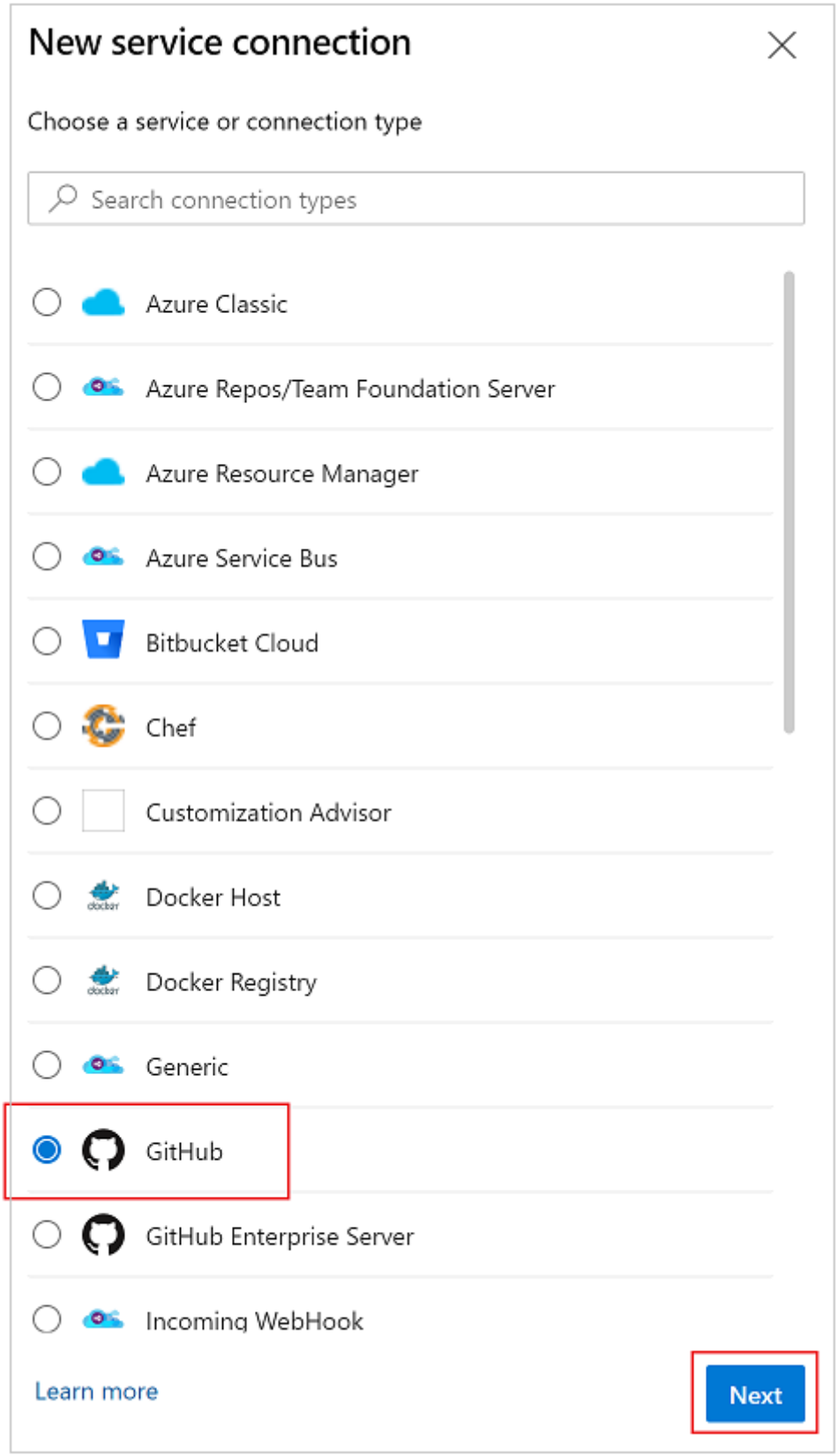
4. Under **Pipelines**, select **Service connections**.



5. Select **Create service connection**.

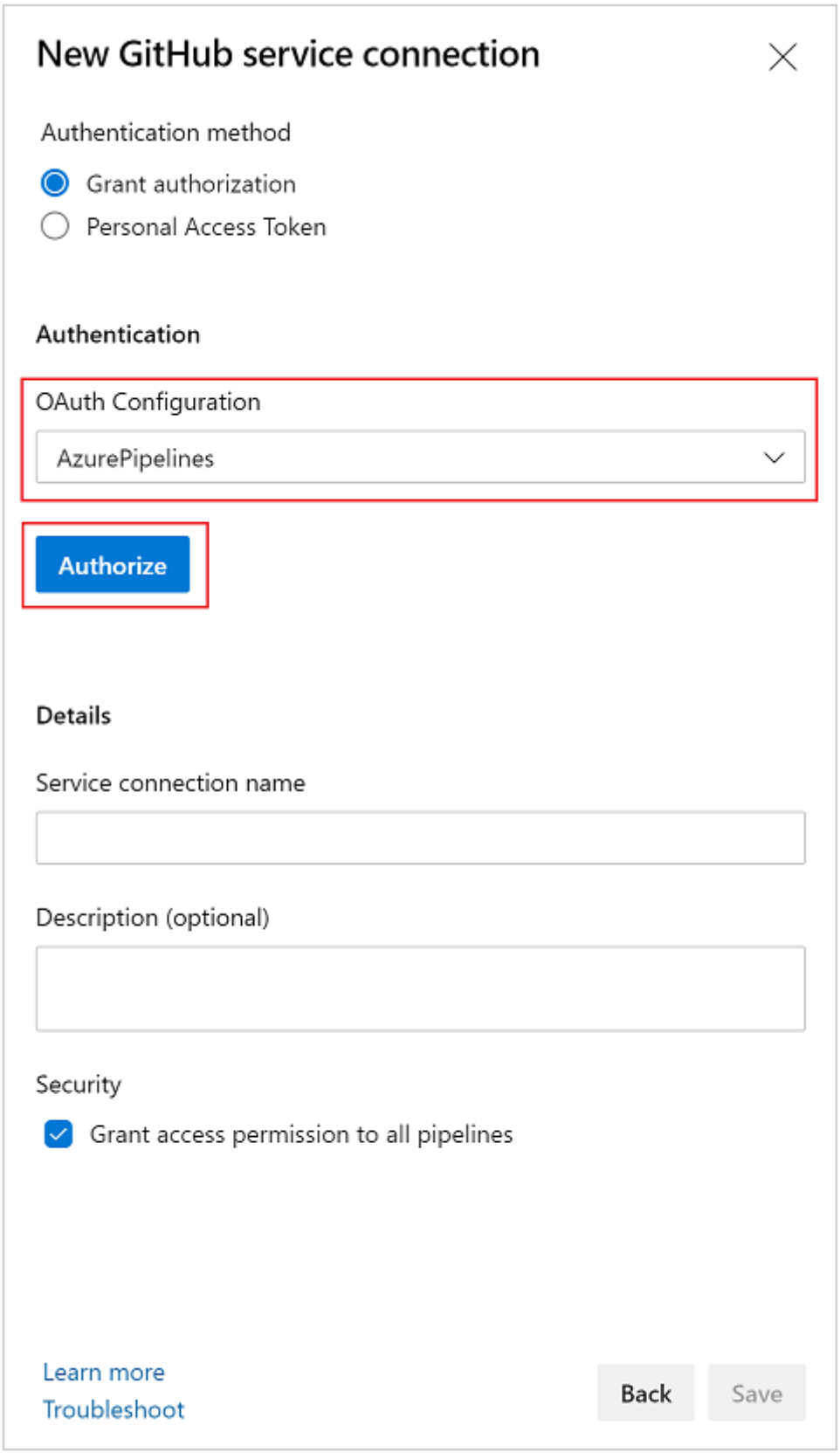
6. Select the **GitHub** service.

7. Select **Next**.



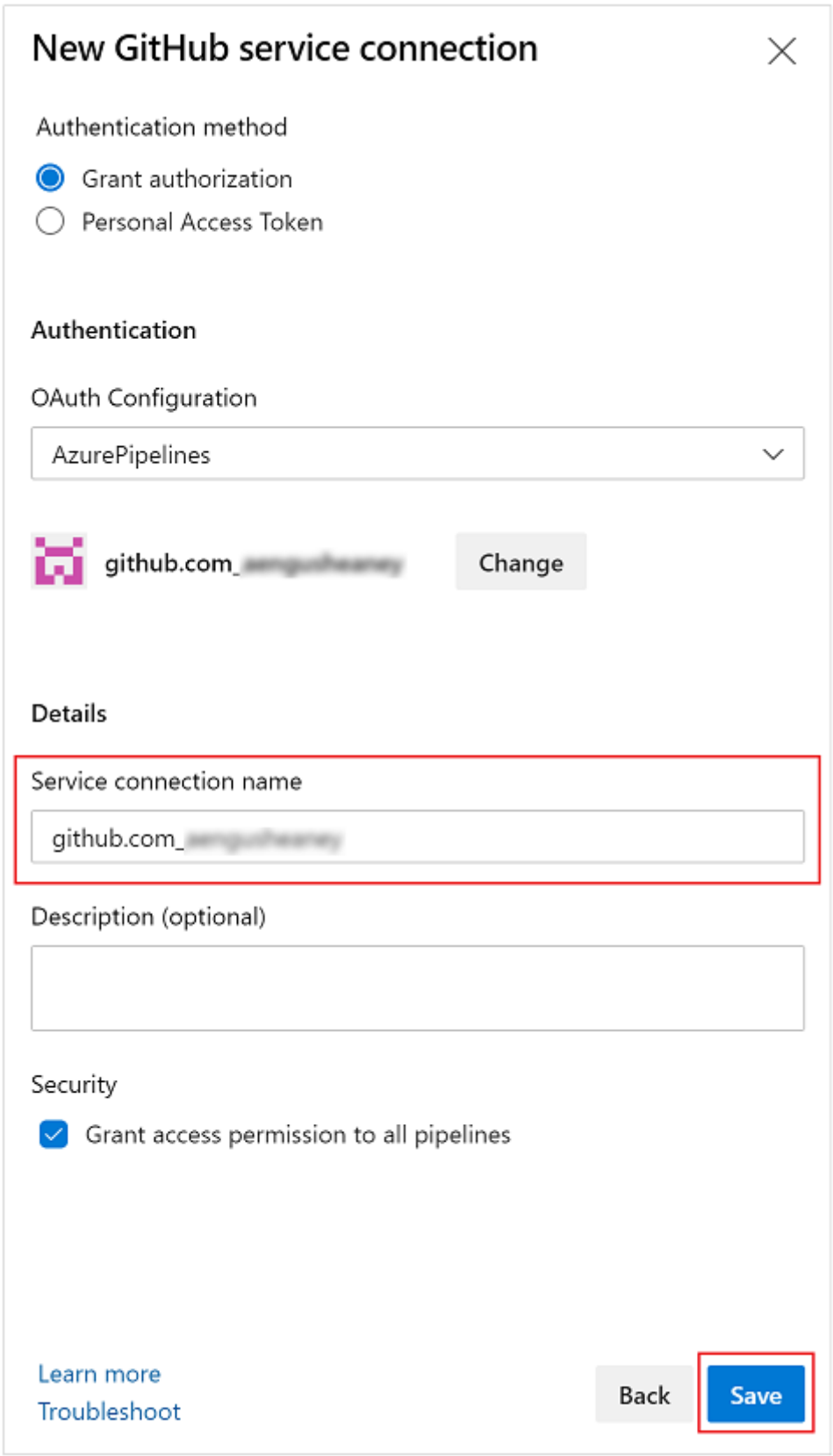
8. Under **OAuth Configuration**, select **AzurePipelines**.

9. Select **Authorize**.



10. Optionally, you can update the **Service connection name**.

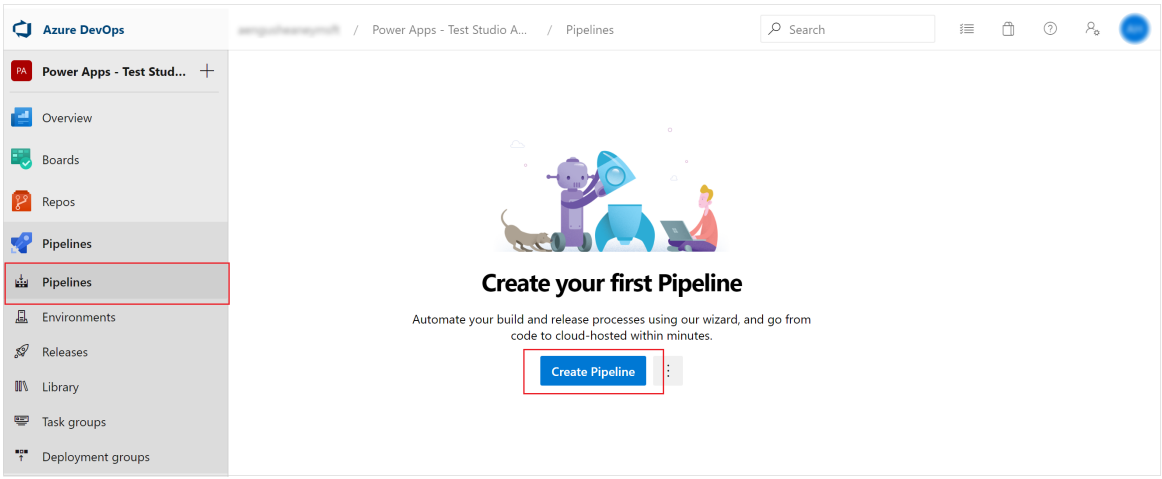
11. Select **Save**.



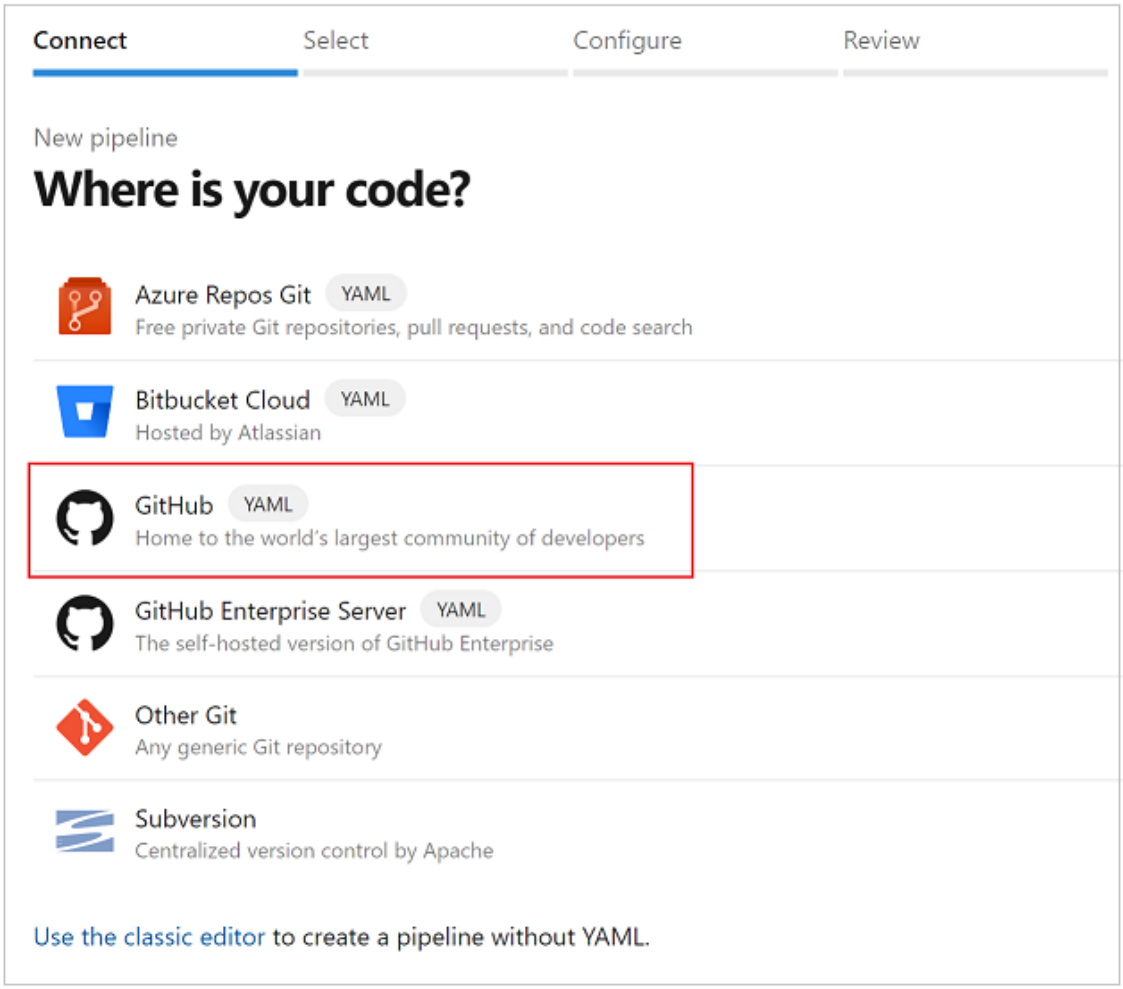
Create a pipeline

1. Select **Pipelines** in the left navigation pane.

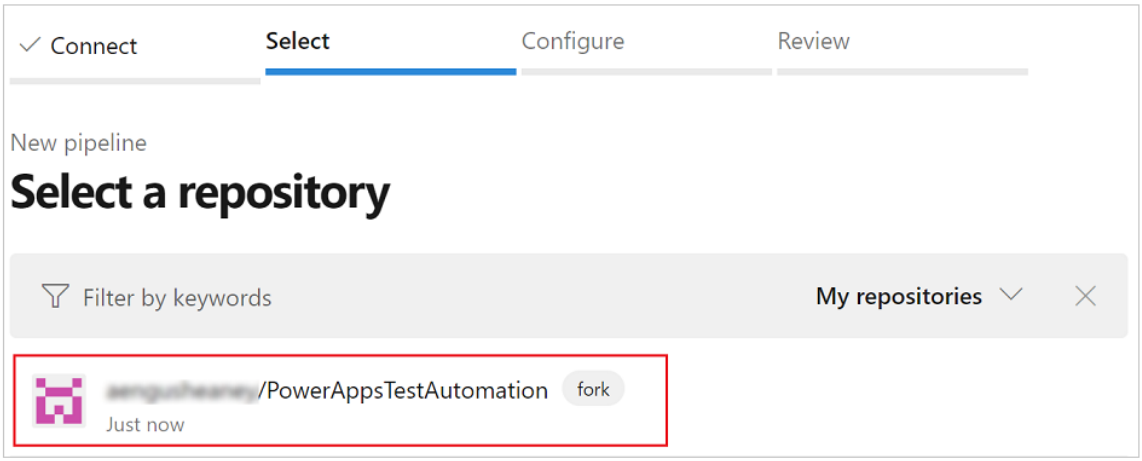
2. Select **Create Pipeline**.



3. Select **GitHub YAML**.



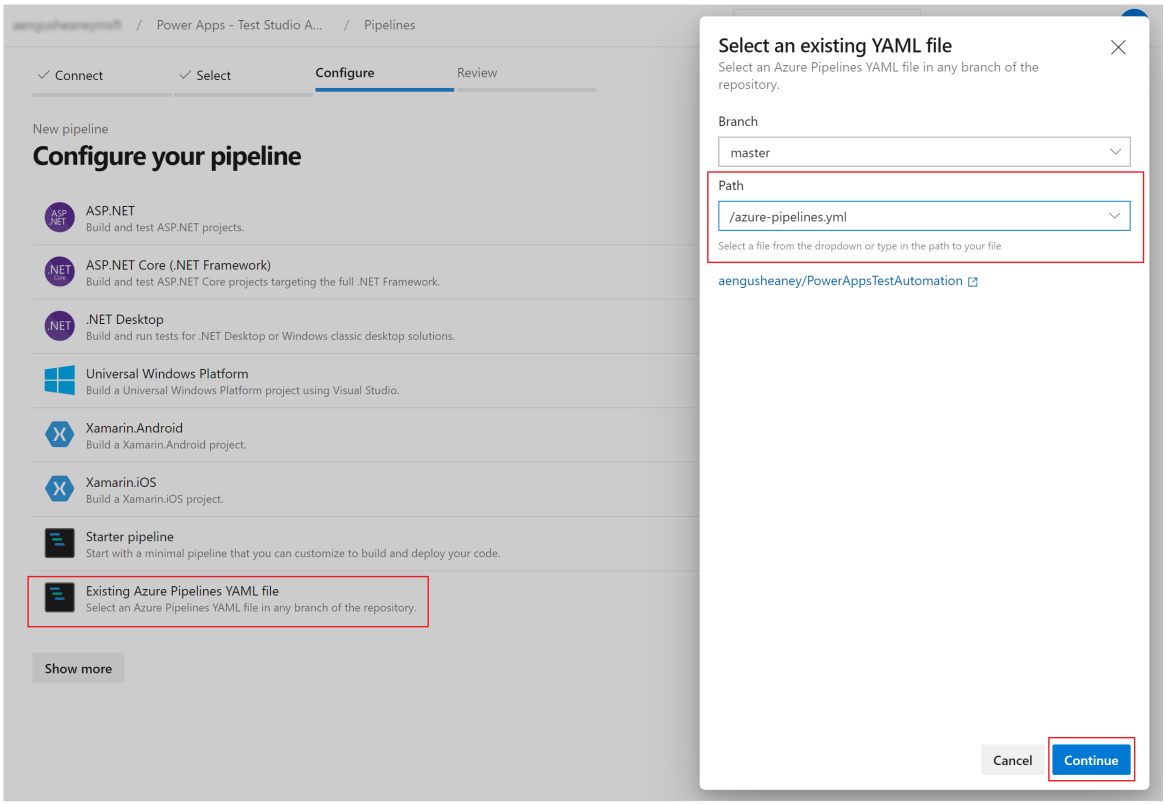
4. Search for or select your repo.



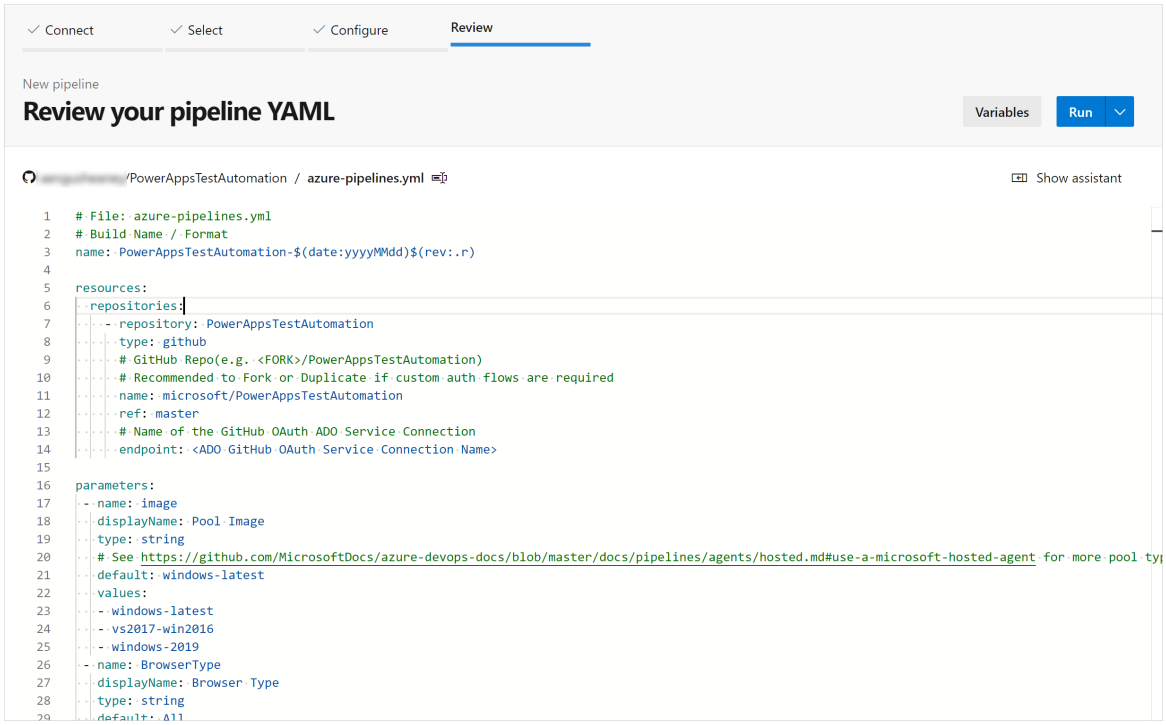
5. Select **Existing Azure Pipelines YAML file**.

6. Set the path to the Azure YAML pipeline file you created earlier.

7. Select **Continue**:

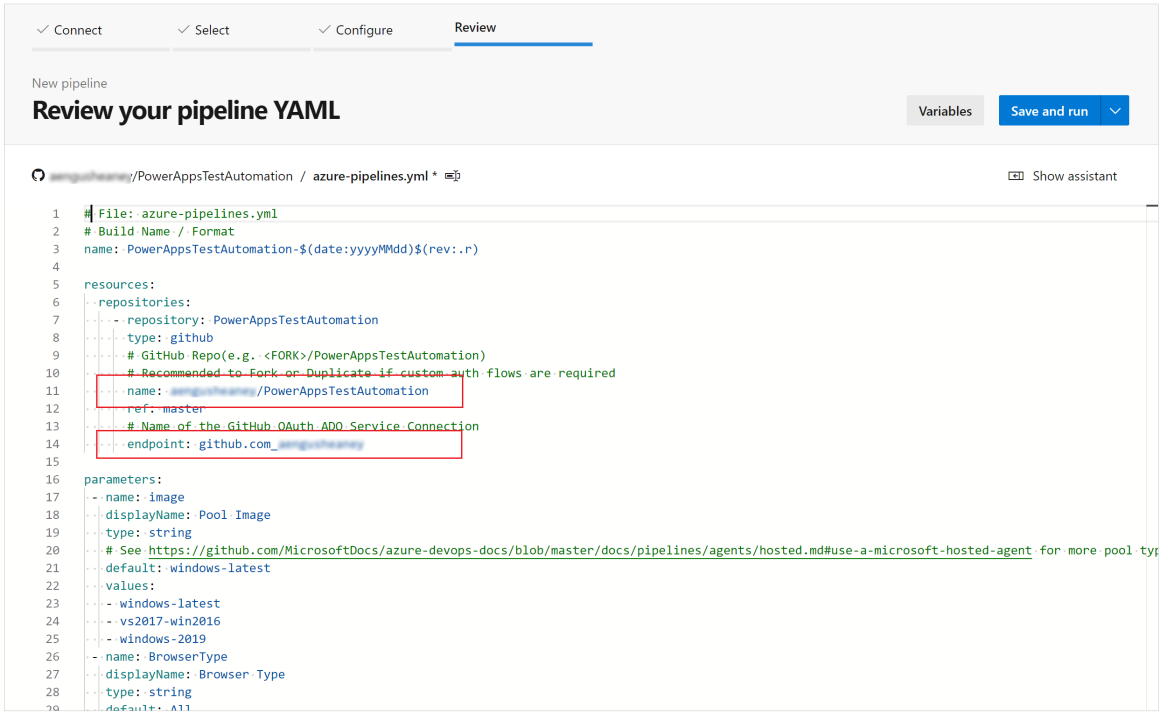


The azure-pipelines.yml file appears.



8. Update **repositories name** to your repo.

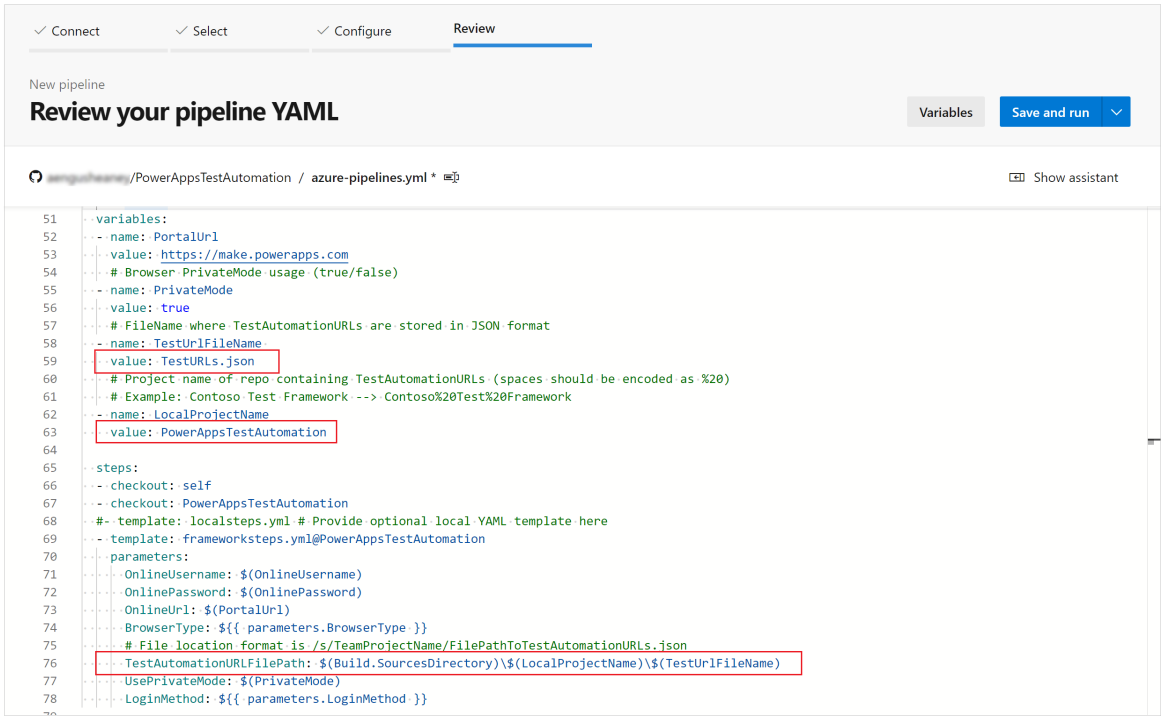
9. Update **endpoint** to the name of the GitHub service connection you created earlier.



10. Update the **TestAutomationURLs** file name. This is the test URLs .json file file you created earlier.

11. Update the **LocalProjectName** value to your repo name, if you changed it.

12. Update the **TestAutomationURLFilePath** to the location of the test URLs .json file in your repo.



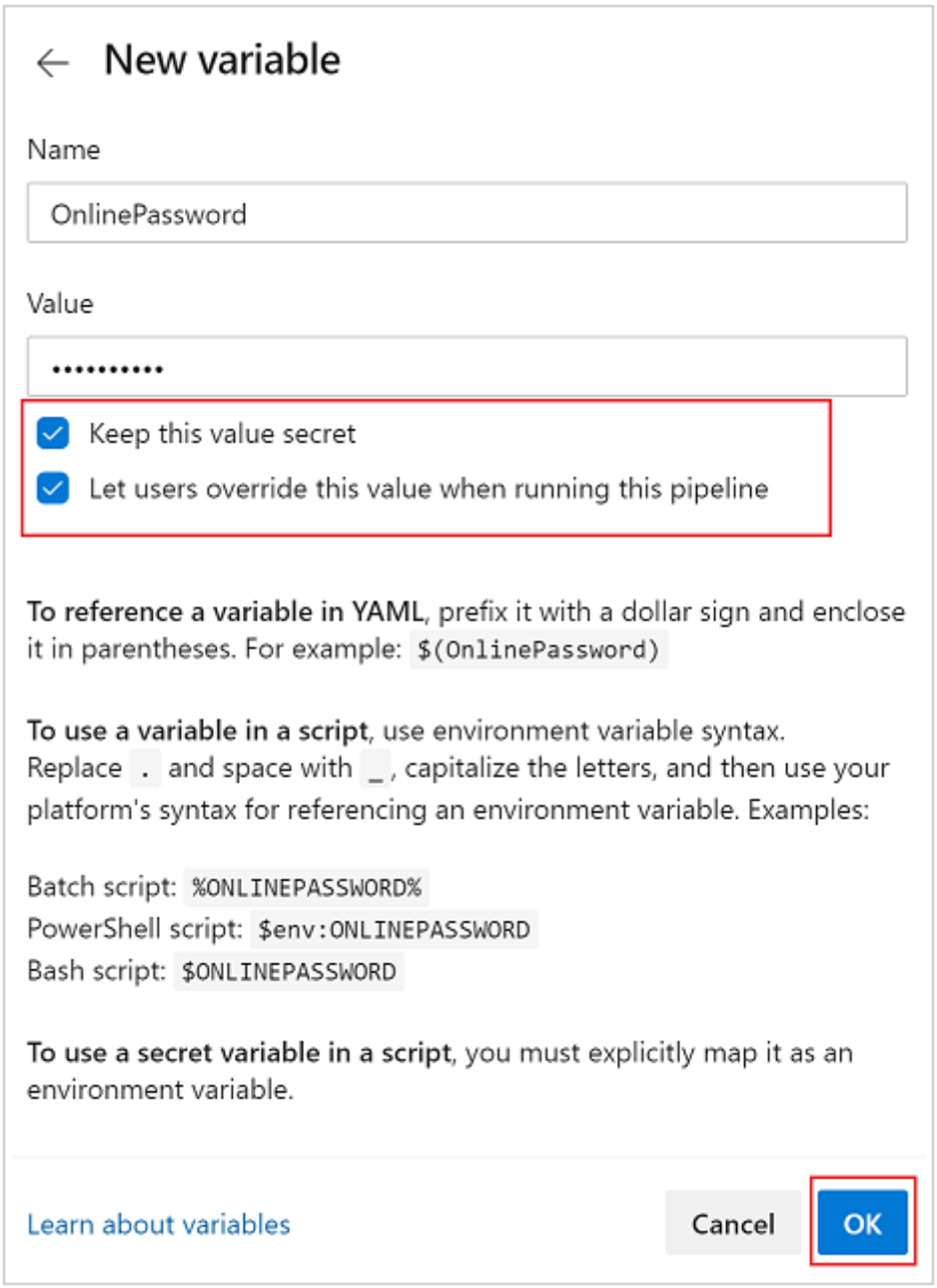
13. Select **Variables**.

14. Add a variable named **OnlineUsername**, and set the value to the Microsoft Entra email address of the user context who will sign in to the application. Tests will run under the context of this user account.

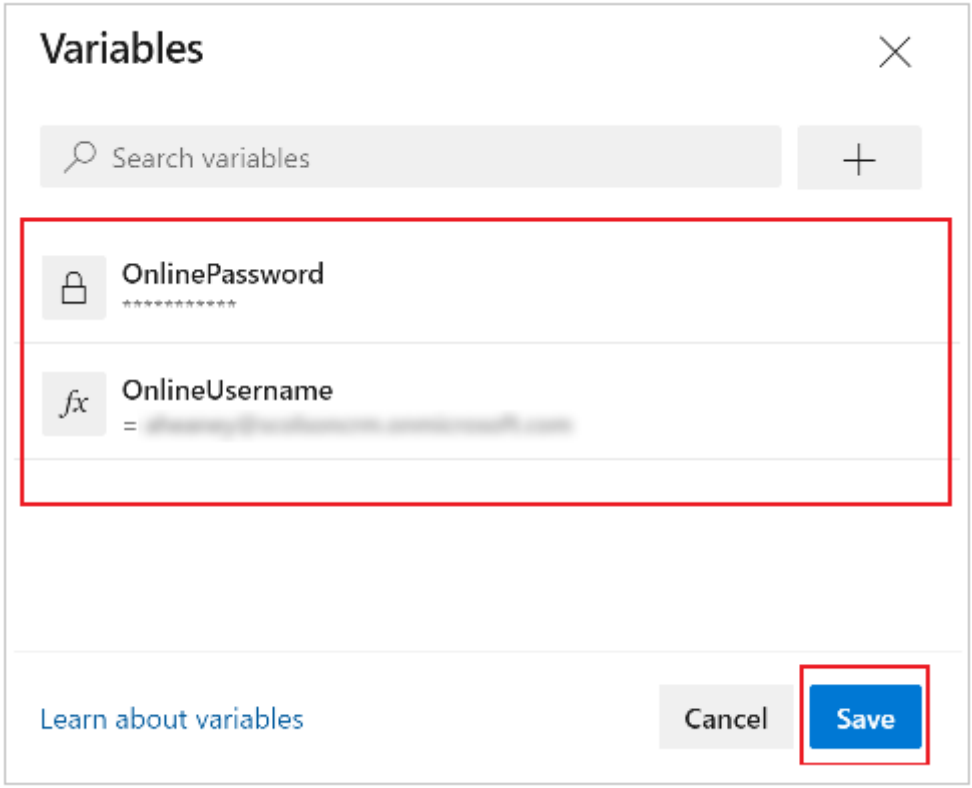
15. Select **OK**.

16. Add another variable named **OnlinePassword**. Set the value to the password of the Microsoft Entra account created earlier.

17. Select **Keep this value secret** and **Let users override this value when running this pipeline**.

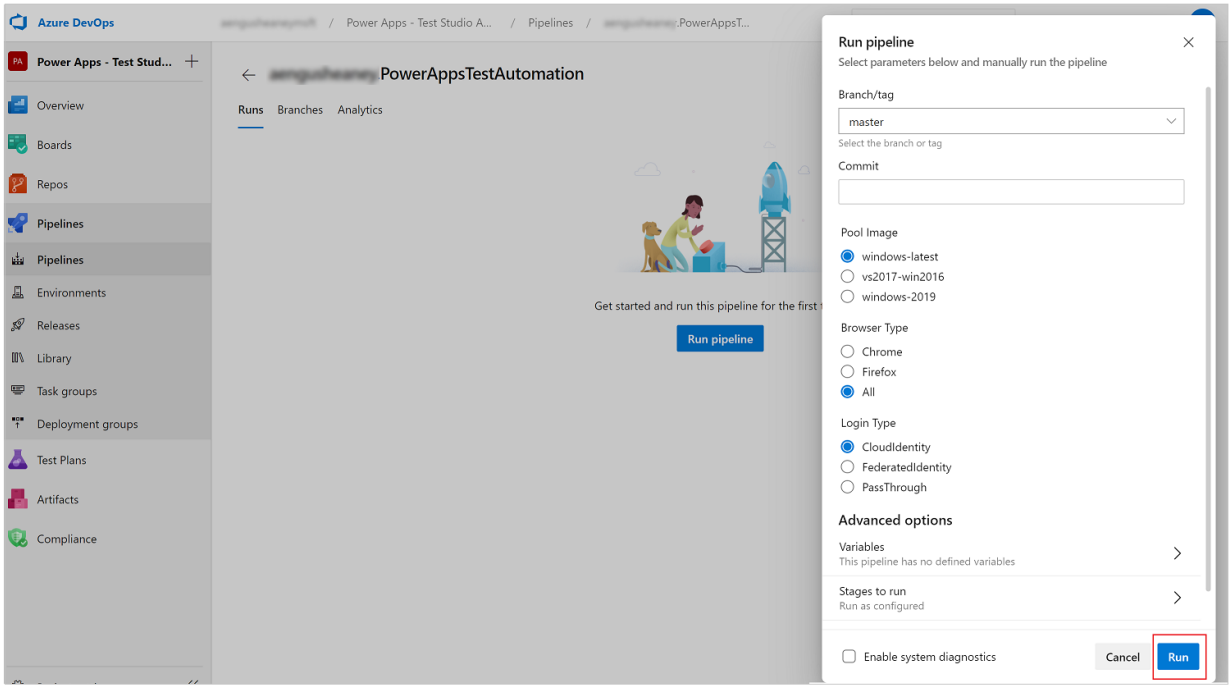


18. **Save** and **Commit** the changes to your repo.

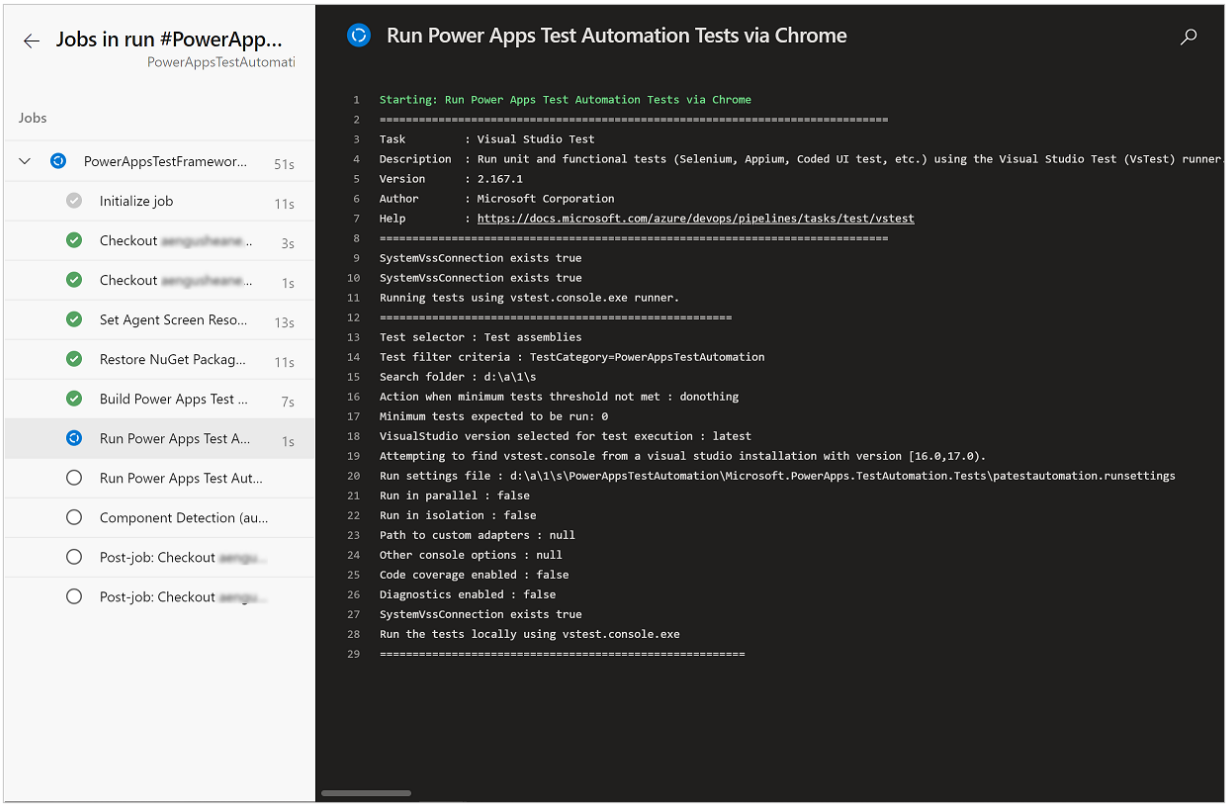


Run and analyze tests

To validate whether your tests are being successfully executed, select **Run**. Optionally, you can optionally select the server image to run your tests and also the browser types.

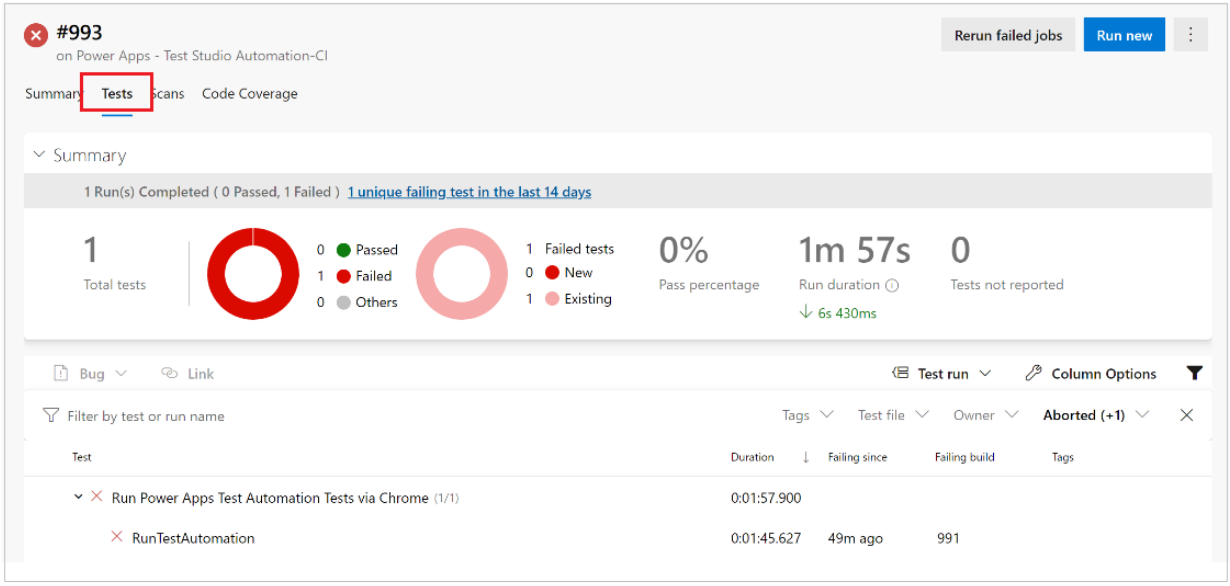


As the job runs, select it to see detailed status for each of the tasks that are running.

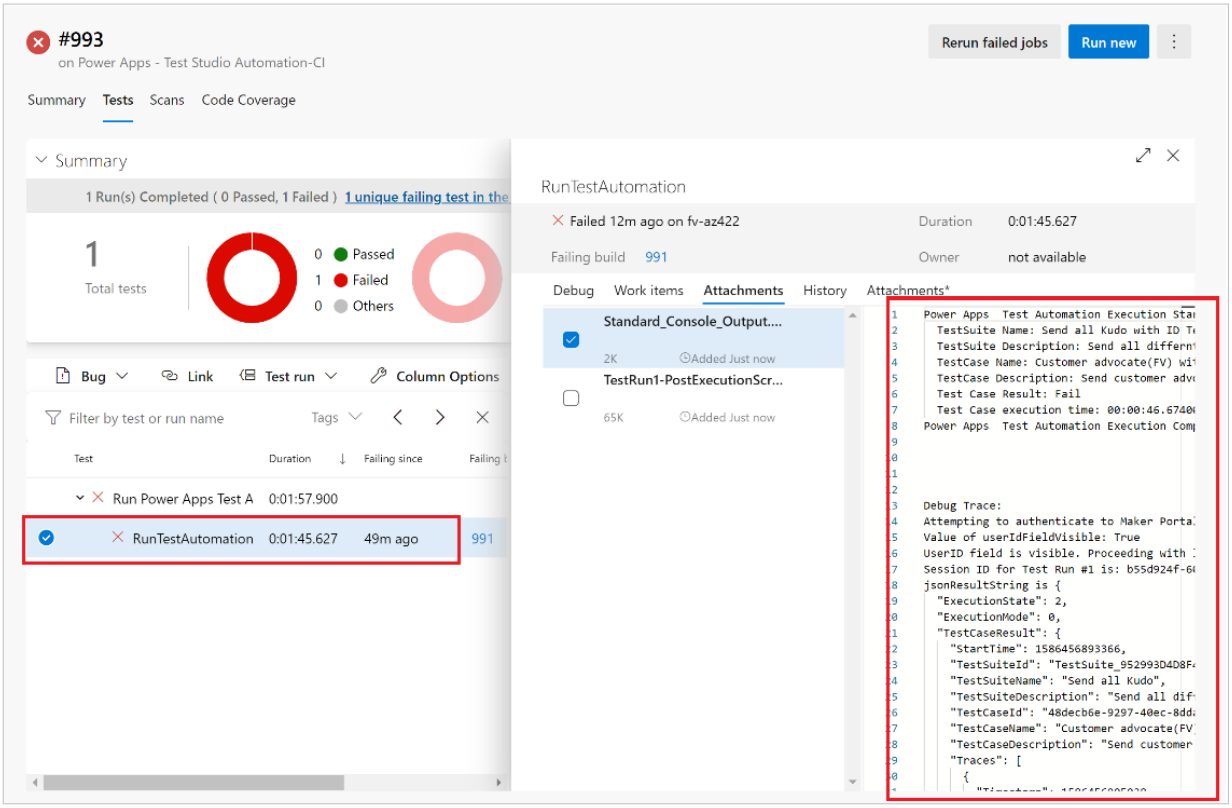


When the job is completed, you can view the high-level job summary and any errors or warnings. By selecting the **Test** tab, you can view specific details on the test cases you've executed.

The following example indicates that at least one of our test cases has failed while executing the tests by using the Chrome browser.



Select **RunTestAutomation** to drill into the details about the failed test cases. On the **Attachments** tab, you can see the summary of the test execution and which test cases have failed or passed in your test suite.



> **Note**
>
> If you execute a test suite, you'll see a summary of test cases that have passed and failed. If you execute a test case, you'll see specific details about the failure with any trace information, if available.

Known limitations

- Multi-factor authentication isn't supported.
- The test summary will report a single test result per browser. The test result will contain one or more test cases or test suite results.
- For any authentication process other tha Microsoft Entra sign-in, you'll need to customize the sign-in process in the **PowerAppsTestAutomation** solution.

See also

- Test Studio overview
- Working with Test Studio
- Automate tests with Azure Pipelines using classic editor

Test canvas applications with Test Engine (deprecated)

> **Note**
>
> **Test Engine is deprecated** and will be removed in a future release. Use Power Platform Playwright samples for test automation capabilities in Power Platform and Dynamics 365 services.

Testing canvas apps with Test Engine requires the following steps:

1. Create a test plan
2. Set up your user
3. Run the test
4. View the results

Create a test plan

Test plans are defined using YAML format. You can author your own test plan or download recorded tests from Test Studio.

Author your own test plan

Create a yaml file using the test plan schema explained in Power Apps Test Engine YAML format (deprecated).

When choosing authoring your own test plan, you can take advantage of the following features:

- **Additional Power Fx Functions**: Use the extra registered Power Fx actions to interact with Power Apps component framework (PCF) components, enhancing the capabilities of your test plans.
- **Define Custom Data Structures and Functions**: Make use of User Defined Types (UDTs) and User Defined Functions (UDFs) to create reusable and maintainable test steps.

> **Tip**

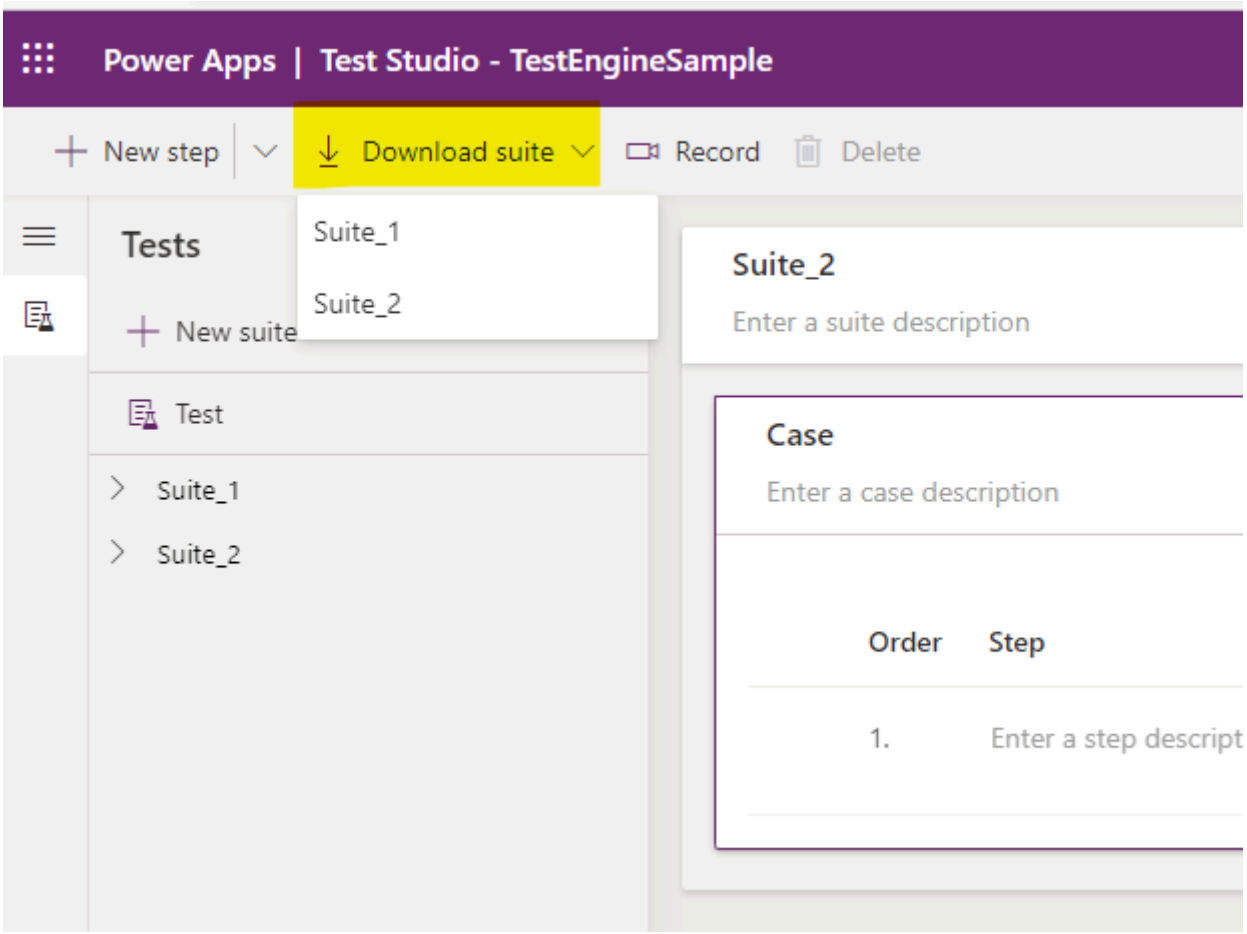
Review available sample plans in the GitHub repository.

Download recorded tests from Test Studio

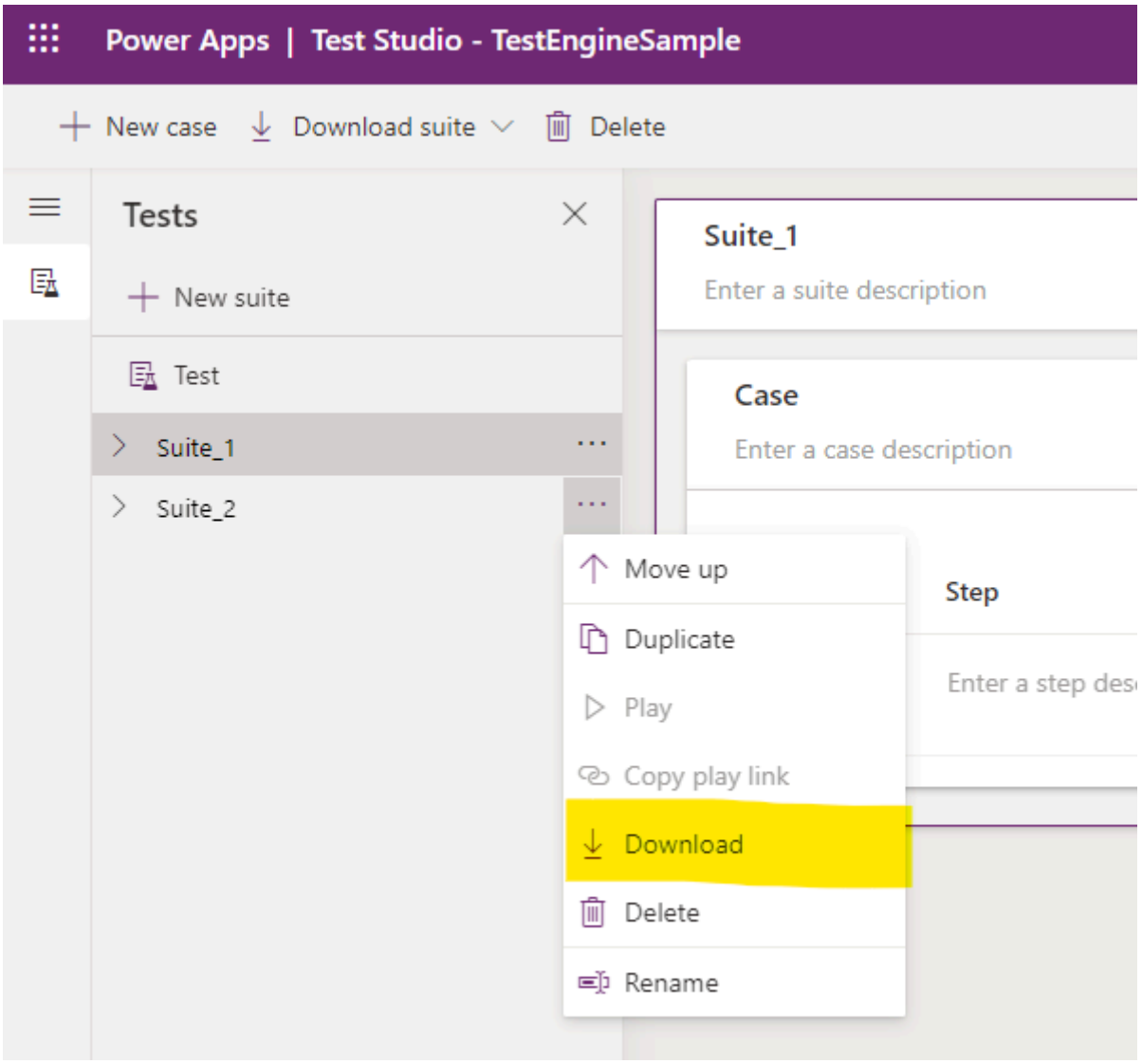
Downloading recorded tests from Test Studio allows quick capture and replay of user interactions with your application. This method is useful for creating basic test cases without writing any code. However, it has limitations in terms of flexibility and the ability to handle complex scenarios.

If you have tests that you recorded in Test Studio, you can download them from Test Studio to reuse in Test Engine.

- Use the **Download suite** button in Test Studio to download the test plan. Choose the test suite to download if you have multiple test suites.



- Alternatively, use the **Download** button available under each test suite.



- Make sure you update the config file and user configurations if you're using a different tenant or environment for this app.

Set up your user

Test Engine currently has these authentication methods:

Expand table

Method	Description
Storage State	Store authenticated user state locally in the user profile using Microsoft Windows Data Protection API
Dataverse	Store authenticated user state inside Dataverse using your own customer managed key encrypted with an X.509 certificate using ASP.NET Core Data Protection

You can't store test credentials in test plan files. You must select `Storagestate` or `Dataverse` as the secure location to store credentials. The test plan file contains references to which

environment variables are used for user persona. For example, the following YAML snippet indicates that the `user1Email` environment variables are used:

YAML

```yaml
environmentVariables:
  users:
    - personaName: User1
      emailKey: user1Email
```

View Users for more information.

Use the following PowerShell script to store the username in your environment variables.

PowerShell

```powershell
$env:user1Email = "someone@example.com"
```

> ⓘ Note
>
> The user password is no longer required in an environment variable. Review authentication for supported options.

Run the test

Use the PAC CLI pac test run command to run your test plan.

Required parameters

You need to provide the following information:

- `--test-plan-file`: Path to your test plan file
- `--tenant`: Your tenant ID
- `--environment-id`: Your environment ID

Example command

Here's an example of running a test for a canvas application:

PowerShell

```
pac test run `
   --provider canvas `
   --test-plan-file your-testplan.te.yaml `
   --tenant your-tenantid-guid-value `
   --environment-id your-environmentid-guid-value
```

You can also use the shorter parameter aliases:

PowerShell

```
pac test run `
   -p canvas `
   -test your-testplan.te.yaml `
   -t your-tenantid-guid-value `
   -env your-environmentid-guid-value
```

Dataverse integration

To enable Dataverse integration with your canvas app tests, you can add the `enableDataverseFunctions` parameter in your test settings:

YAML

```
testSettings:
  extensionModules:
    enable: true
    parameters:
      enableDataverseFunctions: true
```

When you enable Dataverse integration, you're required to open Azure CLI with a user or service principal that has rights to the Dataverse environment. You can use the az login command. More information: Sign in with Azure CLI.

The Dataverse API URL used for integration is obtained either from the host domain name of the pac test run `--domain` parameter or by defining an PowerShell environment variable named `DATAVERSE_URL`.

View the results

When the tests complete, you can view the results of your test in the `.trx` file located in the output directory. This folder contains any screenshots or videos captured by your tests when `recordVideo` is set to `true` in the test plan yaml.

Last updated on 04/18/2026

How to create a canvas app connected to Microsoft Dataverse from scratch

Article • 12/16/2022

Microsoft Dataverse is a premier data platform for Power Platform. Dataverse lets you securely store and manage data used by your apps within a set of tables.

In this article, we'll create an app connected to existing data in a Power Platform environment.

Prerequisites

To complete this lesson, you'll need a Power Apps environment with a Dataverse database with at least one table containing data. If you don't have a license, you can sign up for the Power Apps Developer Plan, or use a Power Apps trial.

Create a trial environment

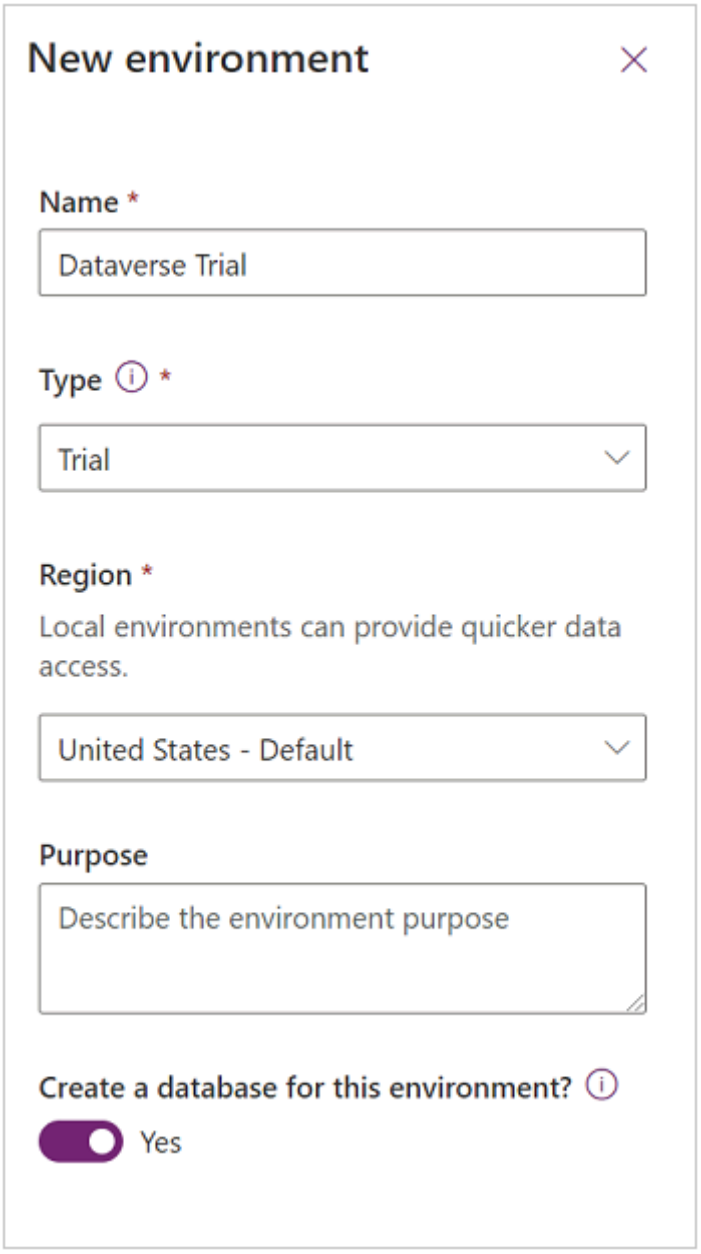
An environment in Power Apps is a space to store, manage, and share your organization's business data, apps, chatbots, and flows. It also serves as a container to separate apps that might have different roles, security requirements, or target audiences. Each environment can have one Dataverse database.

To make sure that you have what you need to complete this lesson, we're going to create a new environment that includes a Dataverse database. This environment is a 30-day trial that will go away at the end of the 30 days, but you can upgrade it to a sandbox or production environment if you want to keep it.

We're going to create a trial environment that will include a Dataverse database, and the sample data used in this article.

1. Go to Power Platform admin center.
2. Select **Environments**.
3. Select **New**.
4. Name the environment **Dataverse Trial**.
5. Select **Trial** for environment type.
6. Toggle **Create Database** to **Yes**.

7. Select **Next**.



8. Toggle **Deploy Sample Apps and Data** to **Yes**.

9. Select **Save**.

Your trial environment will now be created, including a Dataverse database and sample data.

Create a canvas app

Create a canvas app with **Accounts** table in Dataverse. When done, Power Apps will generate an app for you. This app includes the following screens:

- **Browse Screen** - a screen to search, sort, and select an account record, and launch the edit screen to create a new record.

- **Detail Screen** - a read-only screen that displays the details of the record selected on the browse screen.
- **Edit Screen** - an edit form that is used to edit the details of a selected record or create new records.

Try the app

From Power Apps Studio, you can preview your app to try it out.

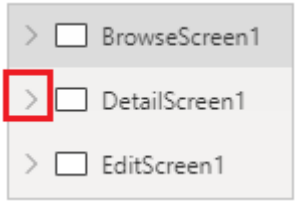
1. Select **Play** from the top-right corner of the screen.
2. From the **Account** list, select **Alpine Ski House**.
3. View the details of Alpine Ski House.
4. Select  to edit the record.
5. On edit form, remove **(sample)** from the account name field.
6. Select  to save the record changes.
7. Back on the account list screen, select + to launch a new record form.
8. Enter the desired fields, such as **Account name**.
9. Save the record.
10. Verify that the newly created record appears in the account list gallery.
11. To exit preview mode, select  in the upper-right corner of the screen.

Using the Tree view

The **Tree View** is on the left-side of the screen (if you don't see it, select ). The tree view gives you easy access to the components in your canvas app, such as screens, galleries, text input controls, and text labels.

The tree view can be collapsed or expanded by screen to make it easy to focus on the part of the app on which you're working.

Currently, the **BrowseScreen1** is expanded. To see the components for other screens, select > next to the name of the screen.



Modify the gallery of accounts

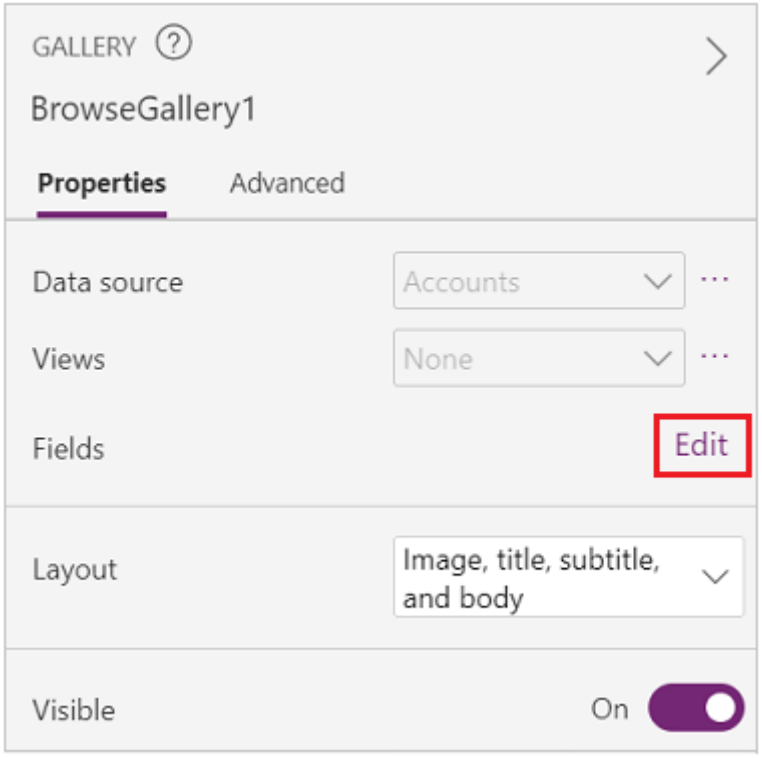
A gallery in Power Apps is used to show several related items. For example, in our app, **BrowseGallery1** displays all of our account records. When one of the items is selected, a border is placed around the item to indicate that it has been selected.

By default, this gallery includes the following three fields: Account name, phone number, and city.

However, we can change the fields included in the gallery.

1. From the tree view, select **BrowseGallery1**.

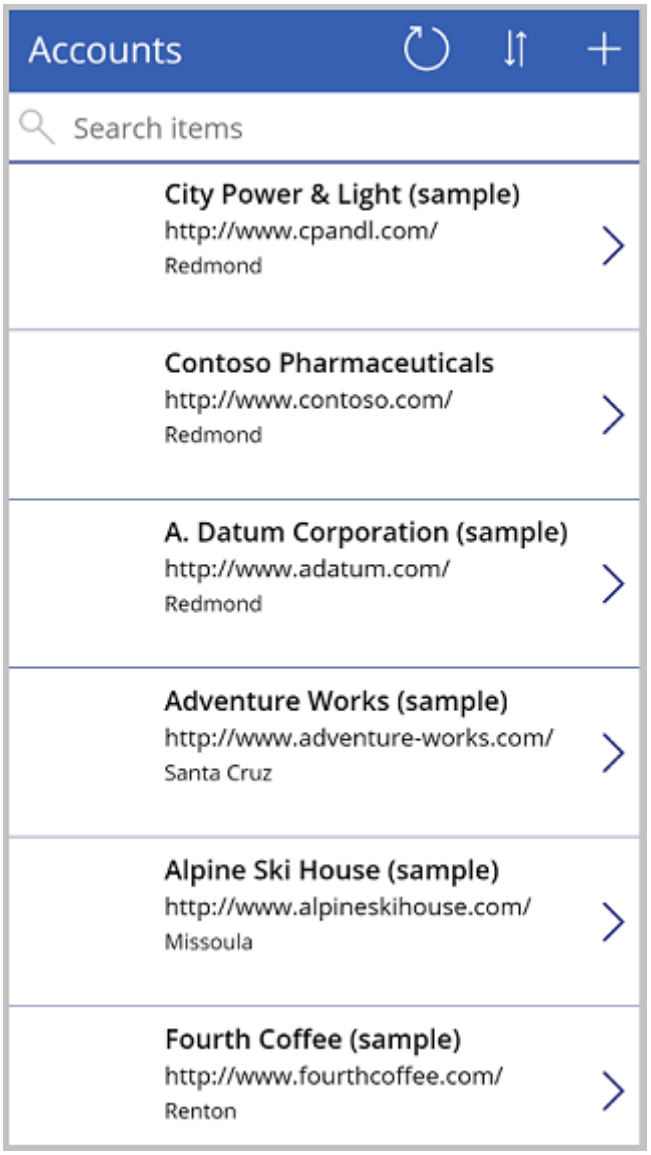
2. In the property pane on the right-side of the screen, select **Edit**.



3. The field data panel will appear. Change Subtitle to Website.

4. Select outside of the field data panel.

5. App will now display the website fields value instead of the phone number.



Update search to include the account name

When you previewed the app, you may have noticed that searching by account name didn't work. That is because the autogenerated app search formula doesn't include the **Account name** field. In this lesson, we'll update to search formula to include the Account name field.

> **Tip**
>
> The account name is stored in a column called **Name**.

1. In the tree view, select **BrowseGallery1**.
2. Verify the **Items** property is selected in the property selector.
3. In the formula bar, update the formula as follows:

```
Power Fx
```

```
SortByColumns(Search([@Accounts], TextSearchBox1.Text,
"name","address1_city","address1_composite"), "name",
If(SortDescending1, Descending, Ascending))
```

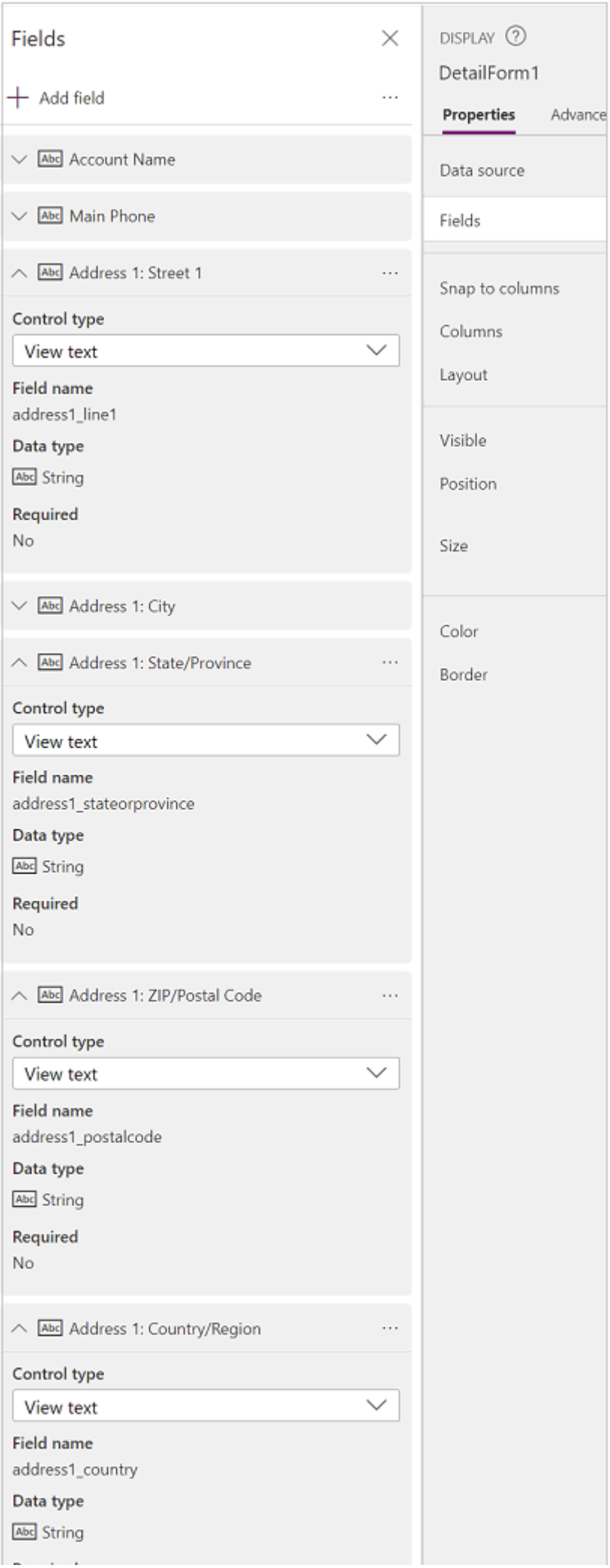
The formula consists of SortByColumns, Search, and If functions to search the **Accounts** table with the text entered in the search box that has the matching data in the name, address1_city, and address1_composite fields, and allow sorting.

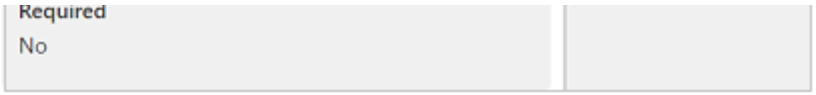
The account list can now be searched and sorted by the Account name field.

Add fields to the account detail screen

Datascreen1 includes a display form that displays the data of the record selected on the browse screen. Display form controls are read-only, intended to allow users to read data but not create or edit data.

1. In the tree view, expand **DetailScreen1**.
2. Select **DetailForm1**.
3. In the right-side properties pane, select **Edit Fields**.
4. Select **Add field**.
5. Select the following fields:
    - Address 1: Country
    - Address 1: State/Province
    - Address 1: Street 1
    - Address 1: ZIP/Postal Code
6. Select **Add**.

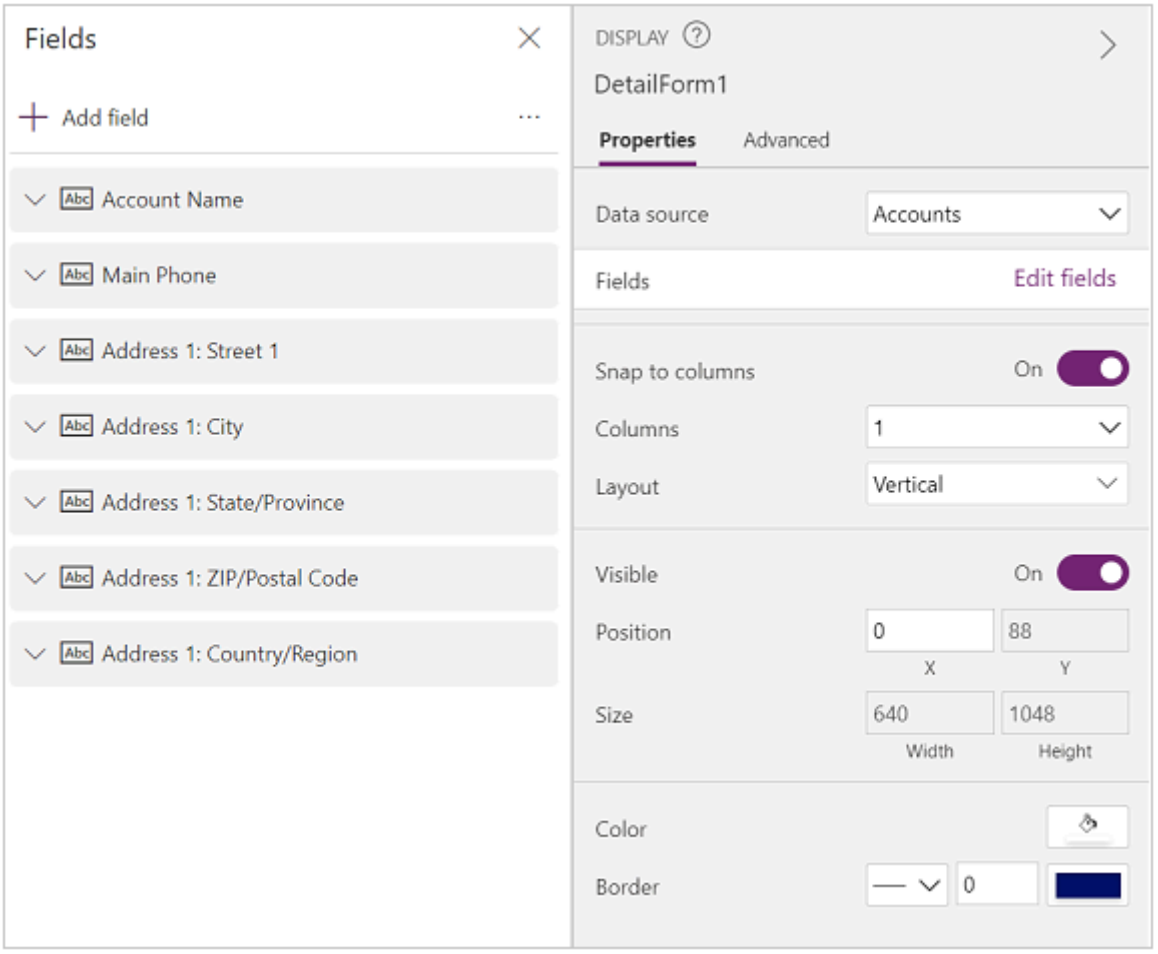

Fields
Add field
Account Name
Main Phone
Address 1: Street 1
Control type
View text
Field name
address1_line1
Data type
String
Required
No
Address 1: City
Address 1: State/Province
Control type
View text
Field name
address1_stateorprovince
Data type
String
Required
No
Address 1: ZIP/Postal Code
Control type
View text
Field name
address1_postalcode
Data type
String
Required
No
Address 1: Country/Region
Control type
View text
Field name
address1_country
Data type
String
DISPLAY
DetailForm1
Properties
Advance
Data source
Fields
Snap to columns
Columns
Layout
Visible
Position
Size
Color
Border



7. You'll now see the selected fields in the Field panel. Rearrange the fields by dragging and dropping them or by selecting ... (ellipsis) to the right of the field name, and selecting **Move up** or **Move down.**

8. Arrange the field in the following order:

    - Account Name
    - Main Phone
    - Address 1: Street 1
    - Address 1: City
    - Address 1: State/Province
    - Address 1: Zip/Postal Code
    - Address 1: Country/Region



9. Select anywhere outside of the field pane to close the pane, and return to **DetailScreen1**.

You'll now see the selected fields displayed in the correct order.

Modify account edit screen

EditScreen1 allows users to edit or create account records. This screen includes an Edit Form control called **EditForm1**.

> ⓘ **Note**
>
> Since the edit form is one of the primary methods for creating or updating records, it should include every field that you want your users to be able to submit, and shouldn't include any fields that are read-only or the ones you don't want your users to modify.

In this lesson, we'll add the same fields that we added earlier to the display form, plus the ability to attach files.

1. In the tree view, expand **EditScreen1**.
2. Select **EditForm1**.
3. In the right-side properties panel, select **Edit Fields**.
4. Select **Add field**.
5. Select the following fields:
    - Address 1: Country
    - Address 1: State/Province
    - Address 1: Street 1
    - Address 1: ZIP/Postal Code
6. You'll now see the selected fields in the field pane. Rearrange the fields by dragging and dropping them or by selecting **...** (ellipsis) to the right of the field name, and selecting **Move up** or **Move down.**
7. Arrange the field in the following order:
    - Account Name
    - Main Phone
    - Address 1: Street 1
    - Address 1: City
    - Address 1: State/Province
    - Address 1: Zip/Postal Code
    - Address 1: Country/Region

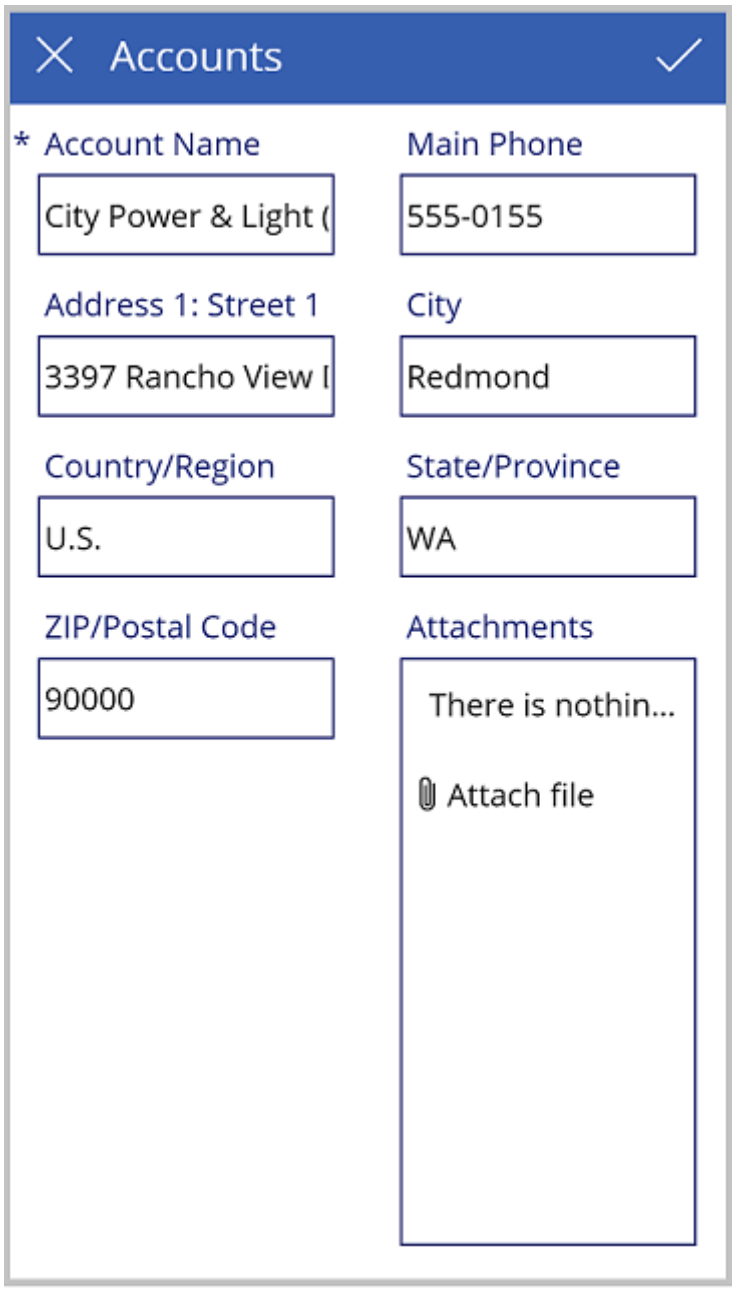
8. Select anywhere outside of the field pane to close the field pane.

9. To minimize unnecessary scrolling in the app, we want the **Edit Form** to arrange the fields into two columns. In the properties pane on the right, select 2 in the **Columns** drop down.

10. To make the form easier to read, we want to remove **Address 1:** part of the address field labels.

    - In tree view, expand **EditForm1**.
    - Select **Address 1: City_DataCard2** control.
    - In the properties pane on right, select **Advanced**.
    - Select **Unlock to change properties**.
    - Change **DisplayName** property to **City**.
    - Follow the above steps to remove **Address 1:** from the Street 1, Country/Region, State/Province, and Zip/Postal Code fields.

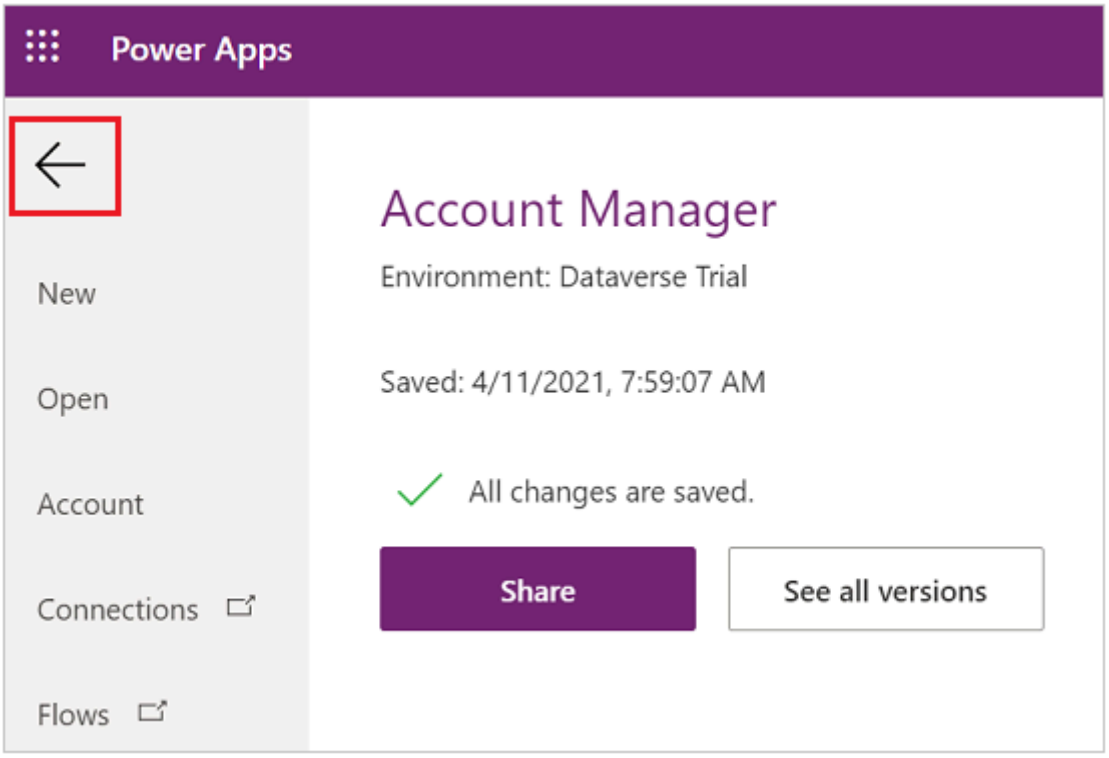
EditScreen1 should now look like the following:



Save the app

By saving the app, your app will become available to use and share with others.

1. In Power Apps Studio, select **File**.
2. In the **Name** field, enter **Account Manager**.
3. Select an icon and background color from the **Icon** area. This will be used to identify your app in the app list.
4. Select **Save**.
5. Select **Save** in the lower right corner.
6. Once app is saved, select the back arrow to return to the Power Apps Studio.



Run the app

1. Sign in to Power Apps.
2. Verify that you're in the environment you used to create the app by looking in the upper-right corner of the screen. If different, select **Environment** and select the appropriate environment.
3. Select **Apps**.
4. From the app list, select **Account Manager**.
5. When the app loads, verify the changes made.
    - Search app list for **Blue Yonder Airlines**.
    - Select the **Blue Yonder** record and verify you see the fields added to the **Display form**.
    - Select to edit the record.
    - Verify form displays two columns of fields.
    - Verify that the field names don't display **Address 1:** in the labels.
    - Select **Attach file** in the Attachment area, select a file, and select **Open**.
    - Verify uploaded file appears in the attachment list.
    - Select to save the record.

Run your app on mobile

To test your Power Apps on your mobile device:

1. Download the app from your mobile app store.
2. Sign in using your work or school account.
3. Select your app from the list.

See also

- Create a canvas app from a template
- Add and configure controls in canvas apps
- Get started with formulas in canvas apps

Feedback

Was this page helpful? Yes No

Provide product feedback

How to create a canvas app with Power BI visual

Article • 12/16/2022

Microsoft Power BI is a powerful analysis and visualization tool. Power Apps is great at enabling people to take action on the web and mobile. Canvas apps created using Power Apps can be embedded in a Power BI report. Similarly, a Power BI report tile can be added to a canvas app. The best of both worlds can be achieved at the same place and same time.

In this example, we'll create a quick Power BI report and a canvas app, and then see how we can add the report as a tile in the app.

Watch this video to learn how to create a canvas app with Power BI visual: https://learn-video.azurefd.net/vod/player?id=1624717a-18d6-4aba-ae5f-0d0503b2cccb&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fbuild-powerbi-visual

Prerequisites

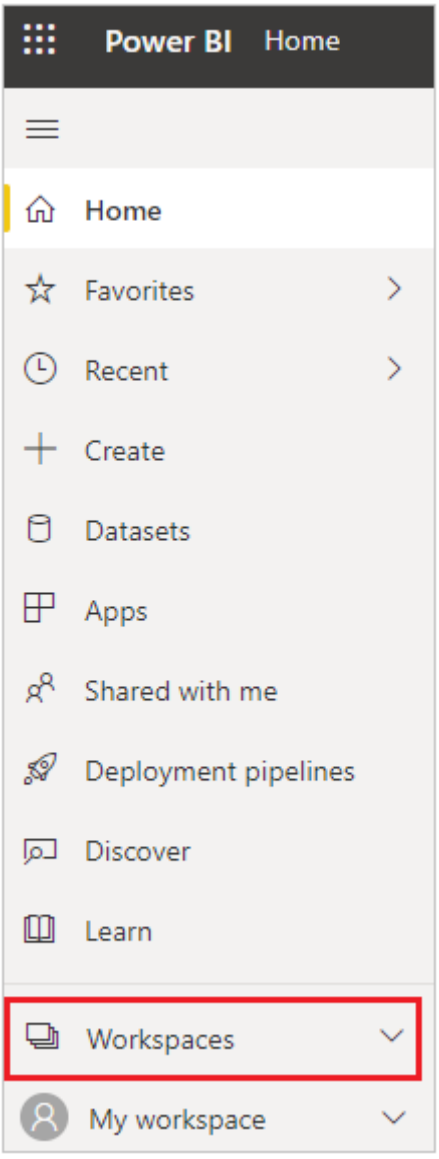
To complete this lesson, we'd need the ability to create Power BI dashboards and reports.

Sign in to Power BI

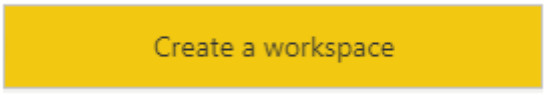
Sign in to Power BI with the same Office license as the one used for creating the Power Apps.

Create a new workspace

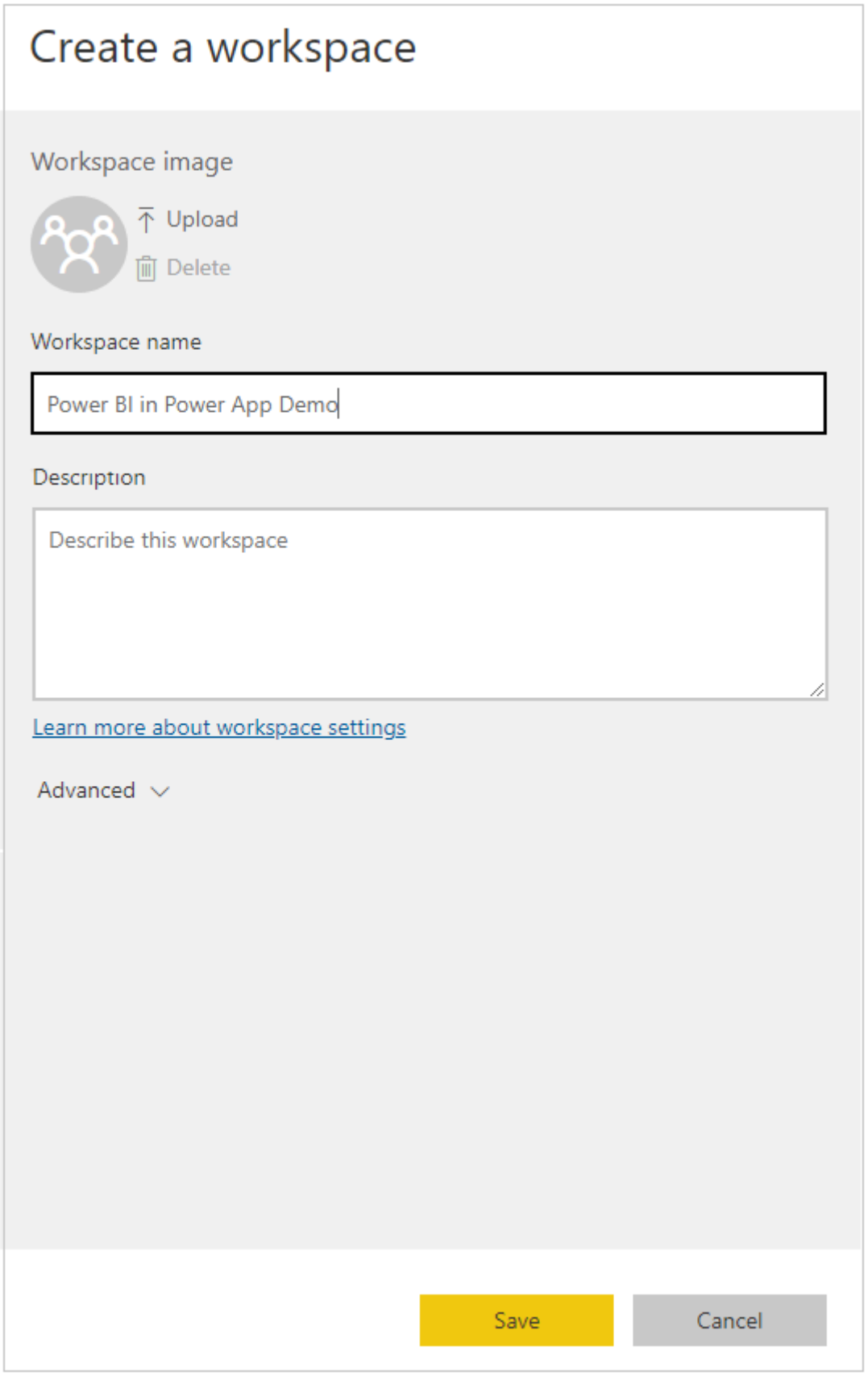
1. Select **Workspaces** from the left-pane.



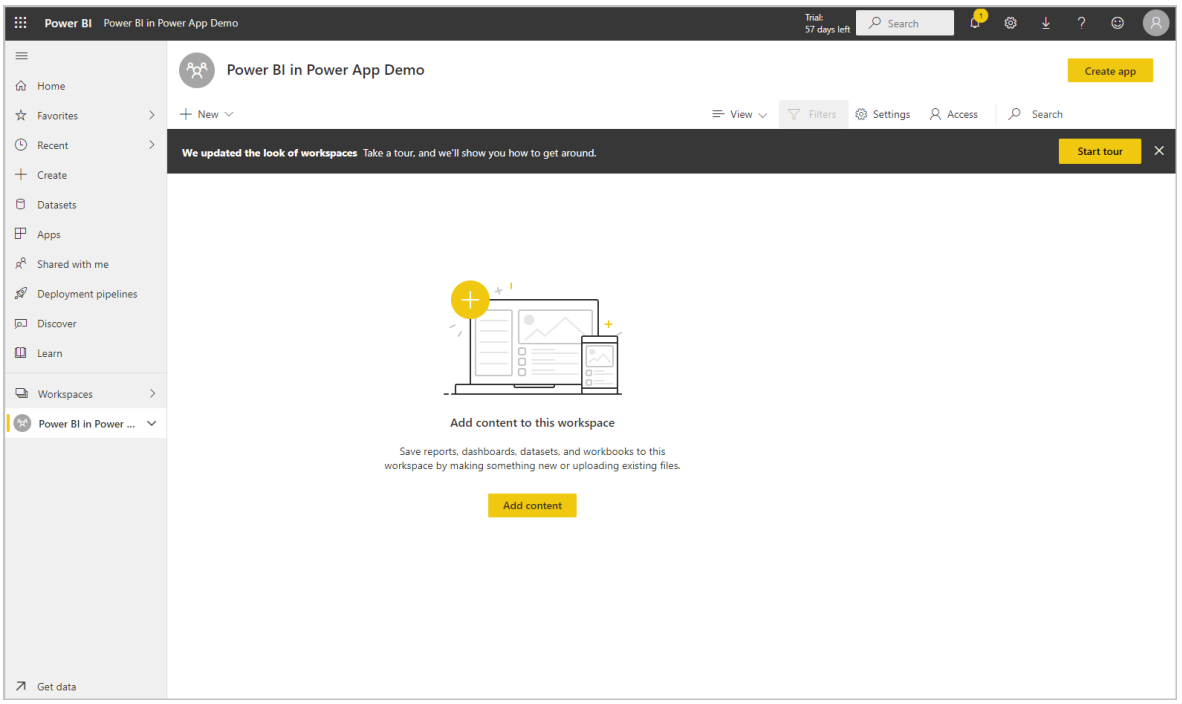
2. Select **Create a workspace**.



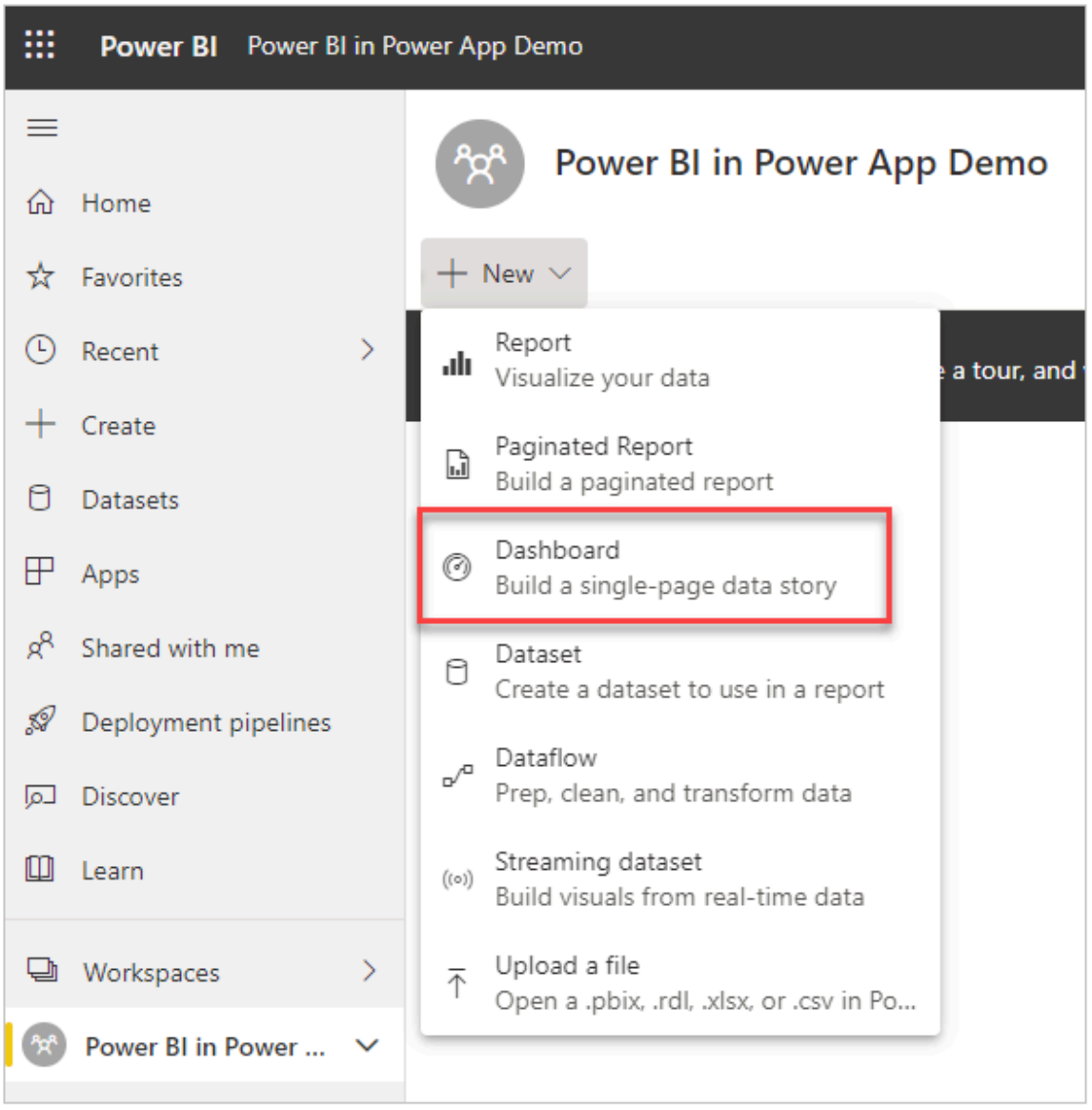
3. Enter a name for the workspace, and then select **Save**.



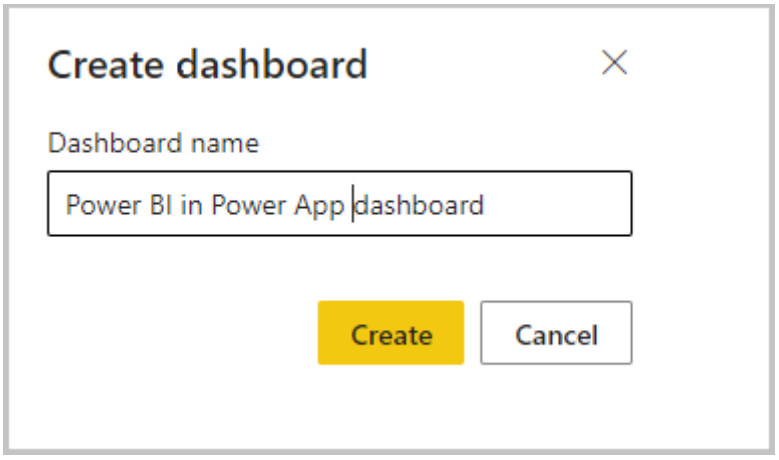
The workspace gets created



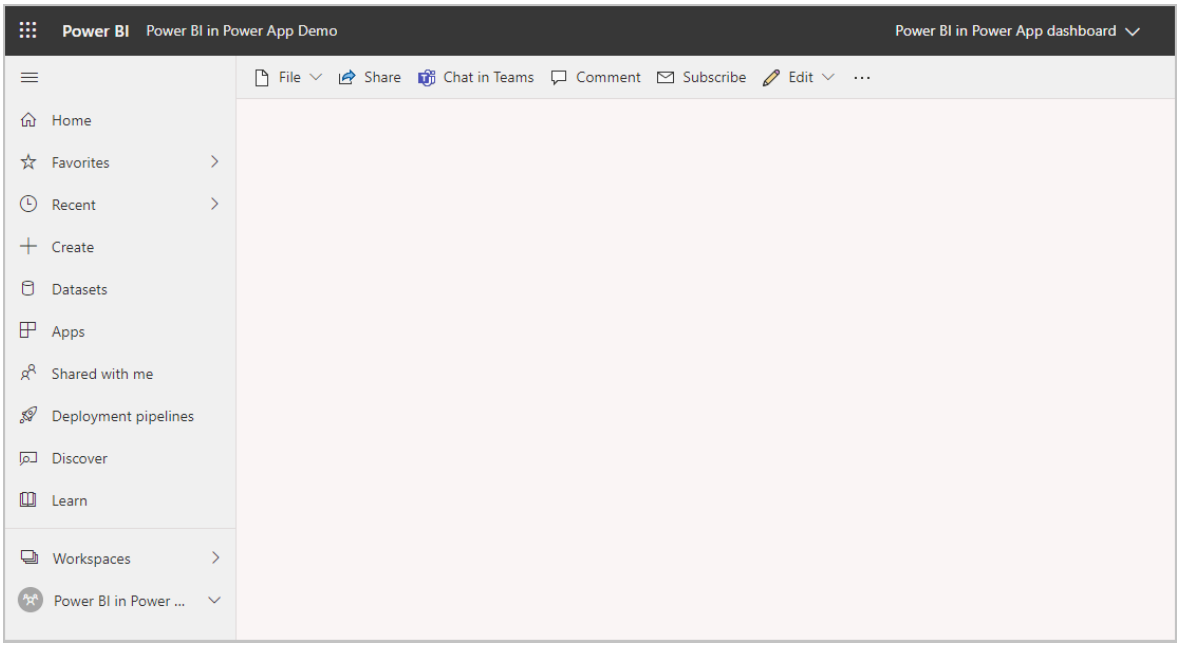
4. Select + **New**, and add a dashboard.



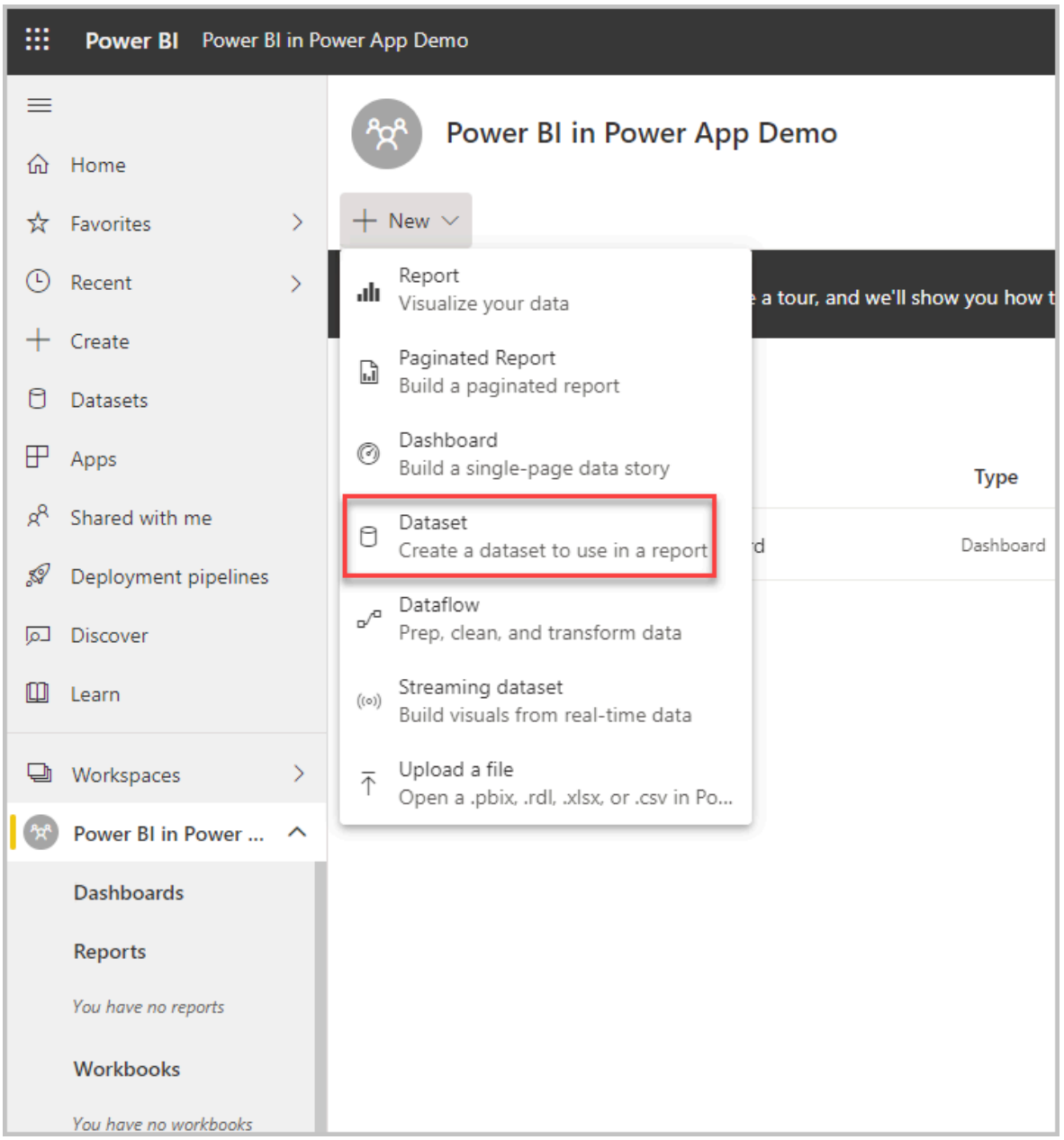
5. Add a name for the dashboard, then select **Create**.



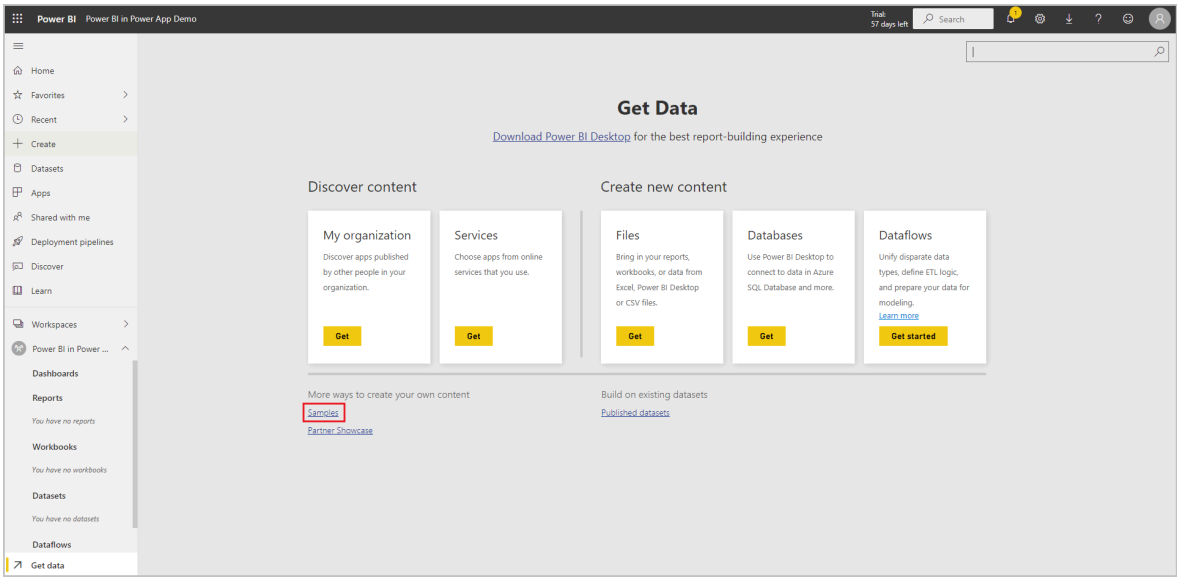
The Power BI dashboard gets created within the workspace we created



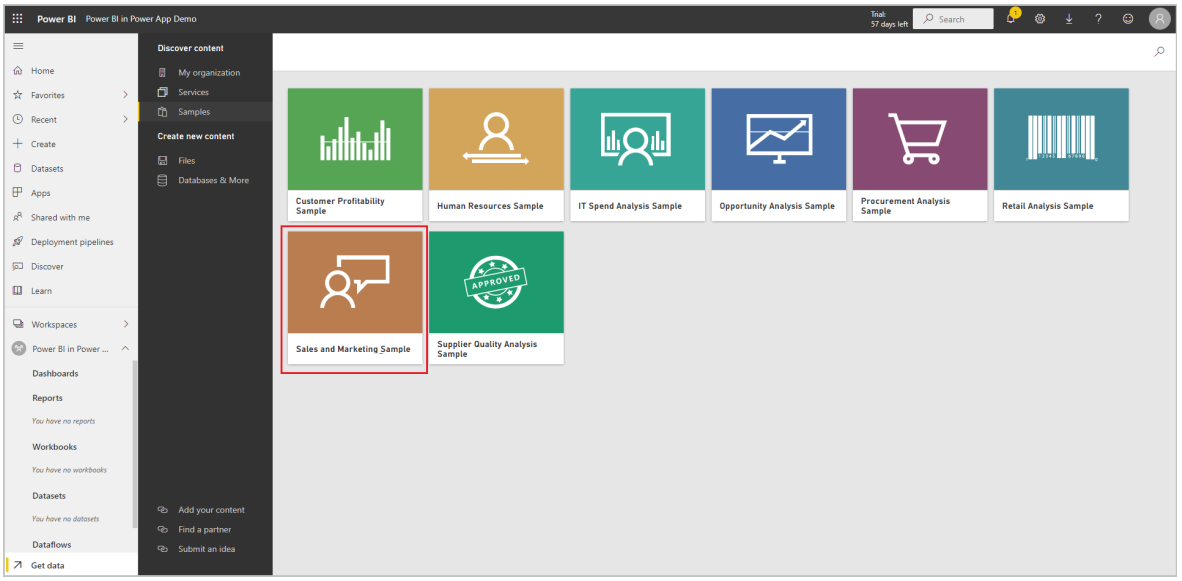
6. Open the workspace, select **+ New**, and then select **Dataset** to create a new dataset.



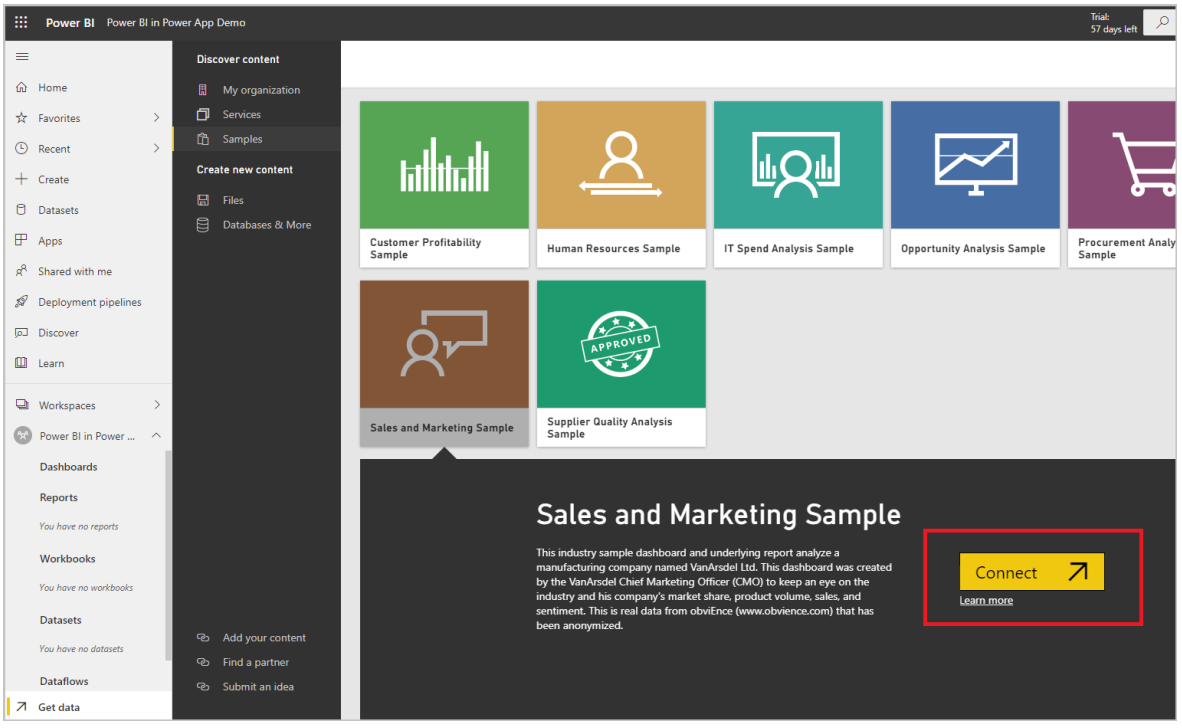
7. Select **Samples** under "More ways to create your own content" to use sample data.



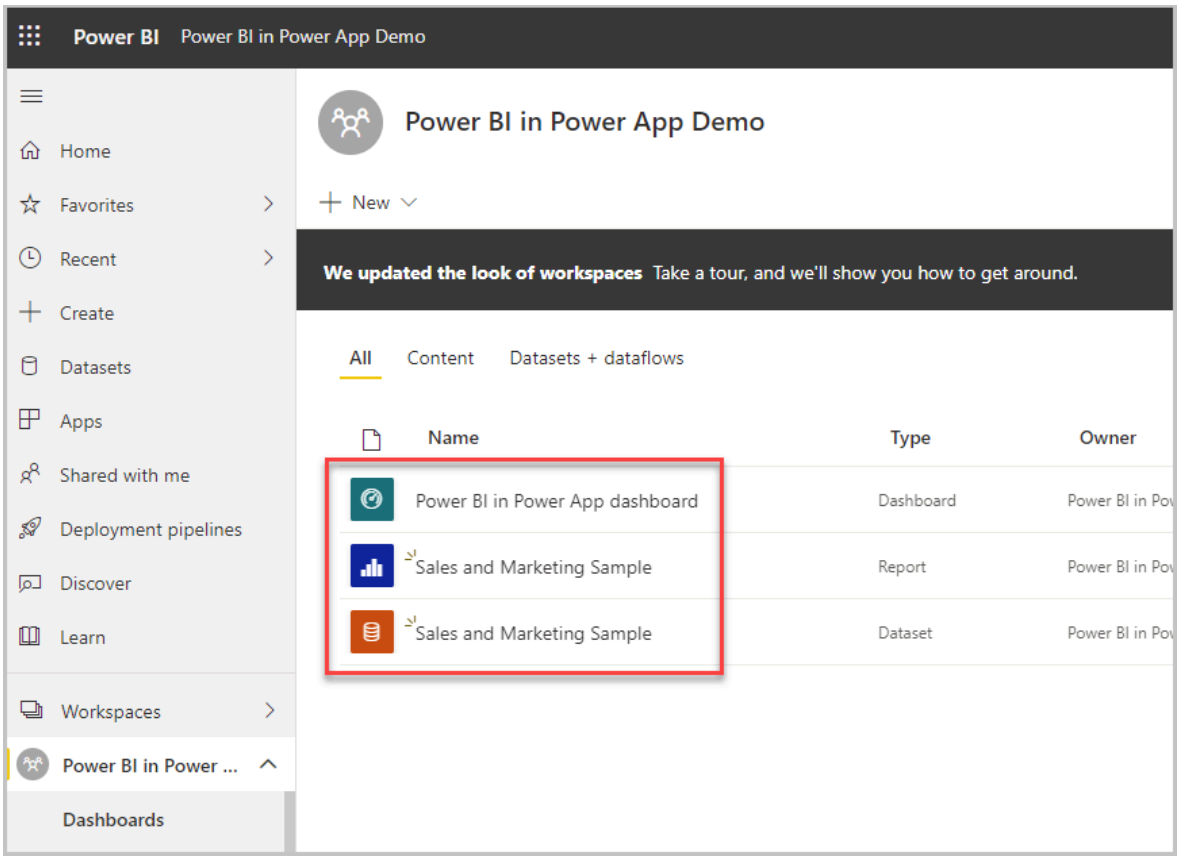
8. Select **Sales and Marketing sample**. You can also select another sample of your choice.



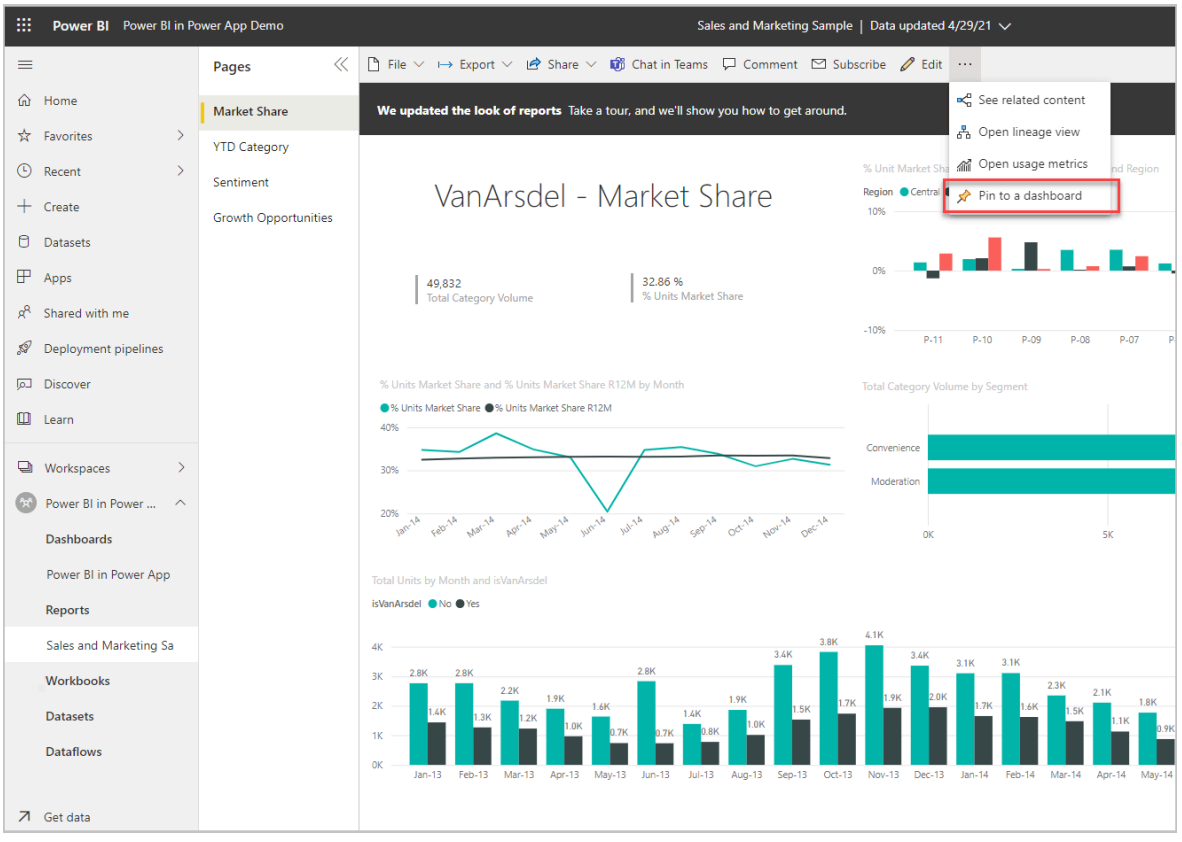
9. Select **Connect** to connect to the sample data.



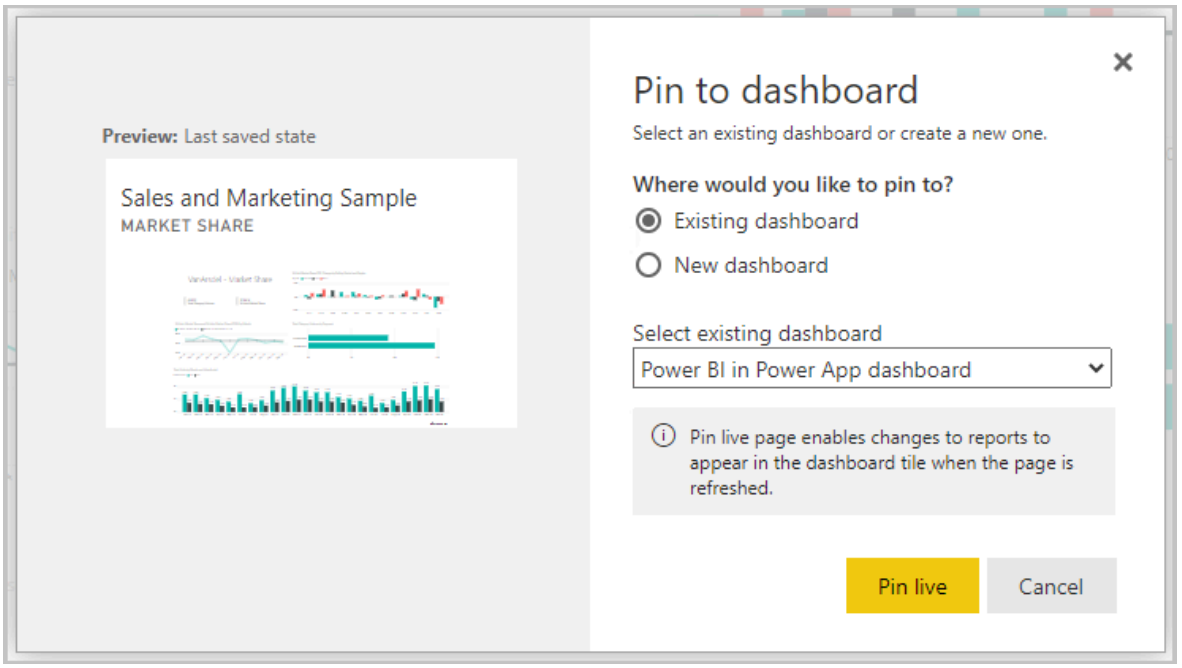
A dataset with the Sales and Marketing sample and a report gets added to the Workspace.



10. From the left-pane, select the report that was created from the dataset, select **...** (ellipsis), and select then select **Pin to a dashboard**.



11. Select **Existing dashboard**, select the dashboard, and then select **Pin live**.

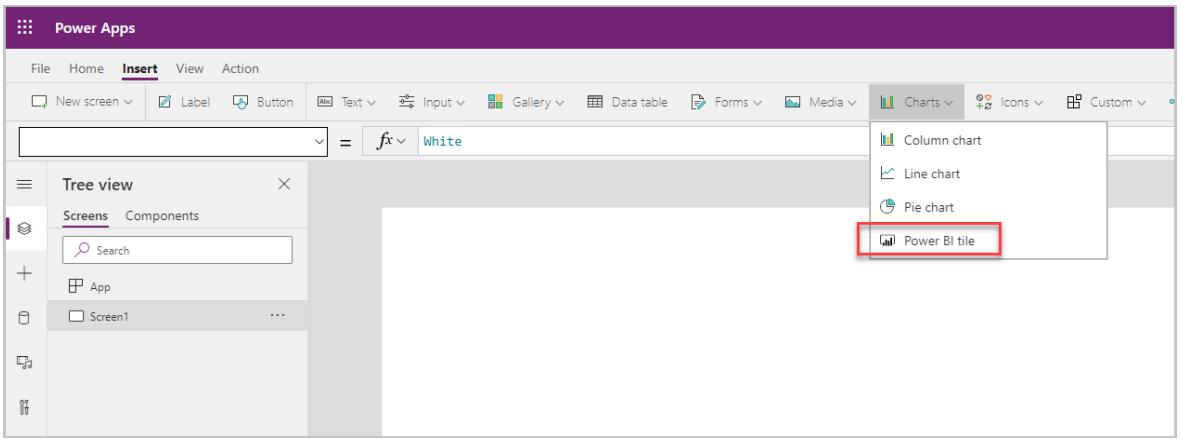


The report gets added to the dashboard, and is ready for use in Power Apps.

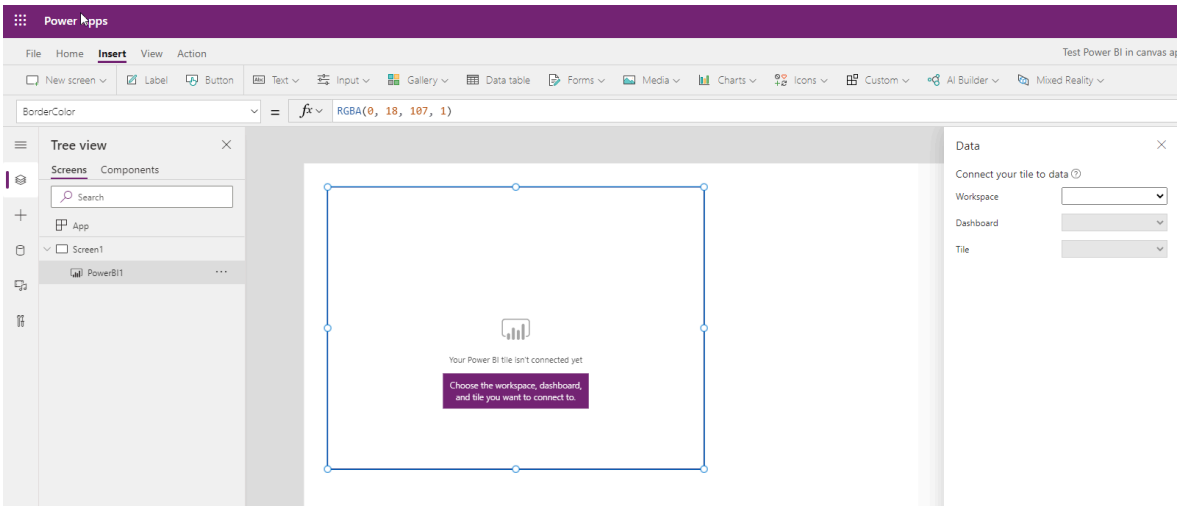
Create a new canvas app

1. Start by creating a blank canvas app.

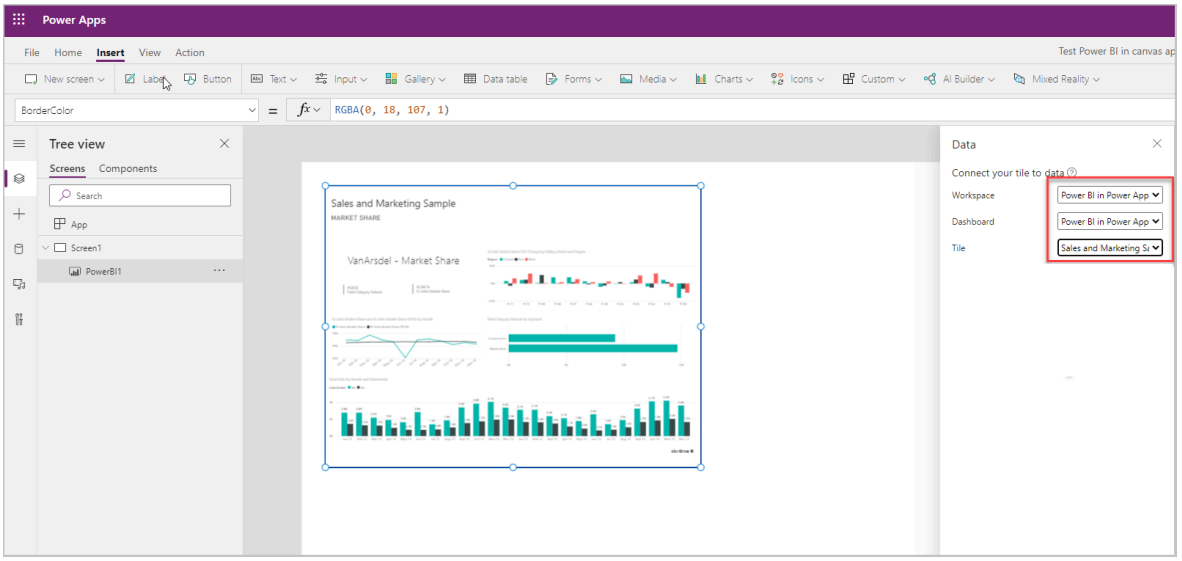
2. Select **Insert** > **Charts** > **Power BI tile**.



The Power BI Tile gets added to the screen.



3. Select the Workspace, dashboard and tile created earlier to add the Power BI tile to the canvas app.



4. Save and publish the app.

You've created a canvas app with Power BI tile.

See also

- Connect to Power BI from Power Apps
- Power Apps visual for Power BI
- Power BI tile control in Power Apps
- Tutorial: Embed a Power Apps visual in a Power BI report

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a canvas app that can trigger a Power Automate flow

Article • 12/16/2022

Power Automate can be used to create logic that performs one or more tasks when an event occurs in a canvas app. For example, a button can be configured so that, when a user selects it, an item is created in a list, an email, or meeting request is sent, a file is added to the cloud, or all of these. Any control in the app can be configured to start the flow, that continues to run even if you close the app.

In this article, we'll learn how create and trigger a flow using canvas apps.

Watch this video to learn how to create a canvas app that can trigger a flow: https://learn-video.azurefd.net/vod/player?id=97f1d303-9f0d-4817-9f9d-c899d023d9b1&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Ftrigger-flow

Prerequisites

To complete this lesson, you'll need Power Automate and Power Apps environment with a Dataverse database. If you don't have a license, you can sign up for the Power Apps Developer Plan or use Power Apps trial.

Create a trial environment

An environment in Power Apps is a space to store, manage, and share your organization's business data, apps, chatbots, and flows. It also serves as a container to separate apps that might have different roles, security requirements, or target audiences. Each environment can have one Dataverse database.

To make sure that you have what you need to complete this lesson, we're going to create a new environment that includes a Dataverse database. This is a 30-day trial that will go away at the end of the 30 days, but you can upgrade it to a sandbox or production environment if you want to keep it.

We're going to create a trial environment that will include a Dataverse database, and the sample data used in this article.

1. Go to Power Platform admin center.

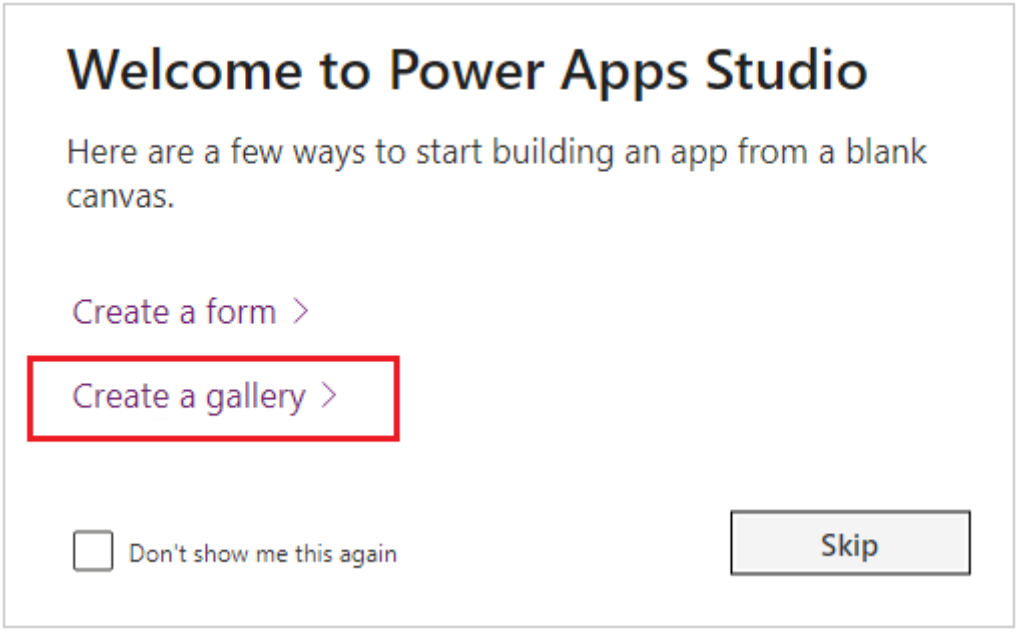
2. Select **Environments**.
3. Select **New**.
4. Name the environment **Dataverse Trial**.
5. Select **Trial** for environment type.
6. Toggle **Create Database** to **Yes**.
7. Select **Next**.
8. Toggle **Deploy Sample Apps and Data** to **Yes**.
9. Select **Save**.

Your trial environment will now be created, including a Dataverse database and sample data.

Create a canvas app

In this lesson, we'll create an app to search, view, and create customer accounts using the **Accounts** table in Dataverse.

1. Start by creating a blank canvas app.
2. Select **Create a gallery**.

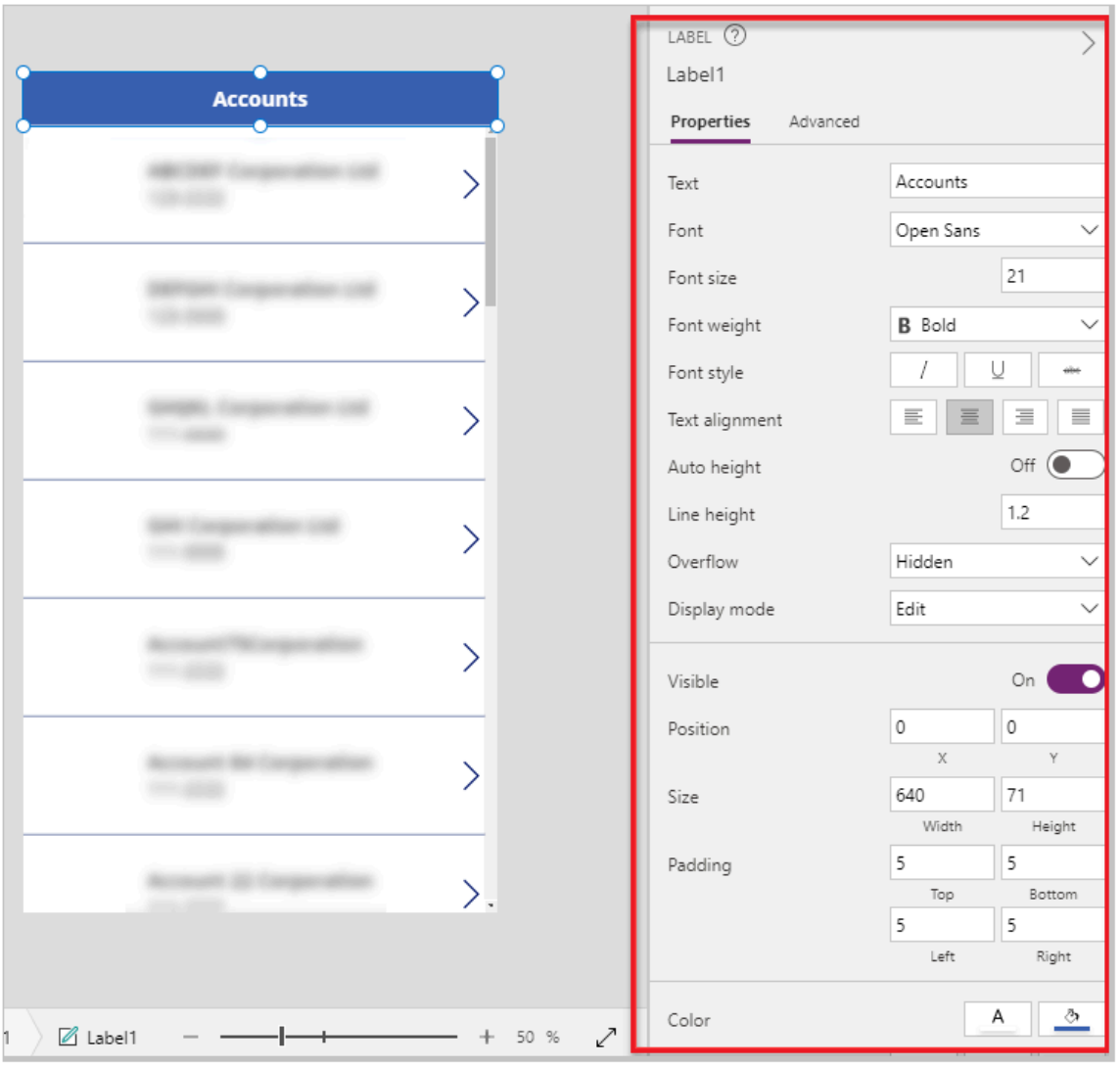


3. Select **Accounts** table.

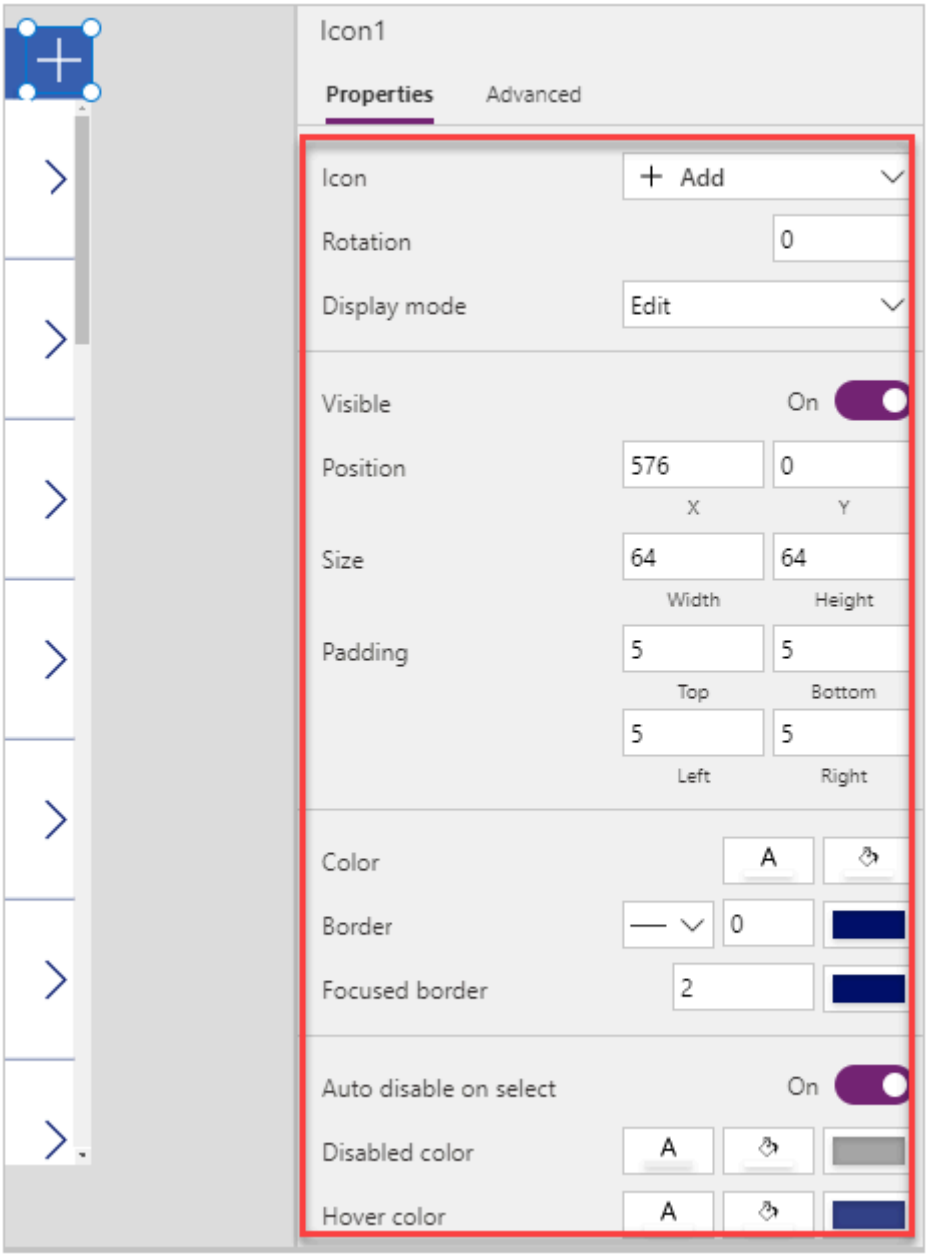
4. Stretch the gallery to the bottom of the screen, and add a label to the top with the text **Accounts**. And then, update the other properties as listed below.

Expand table

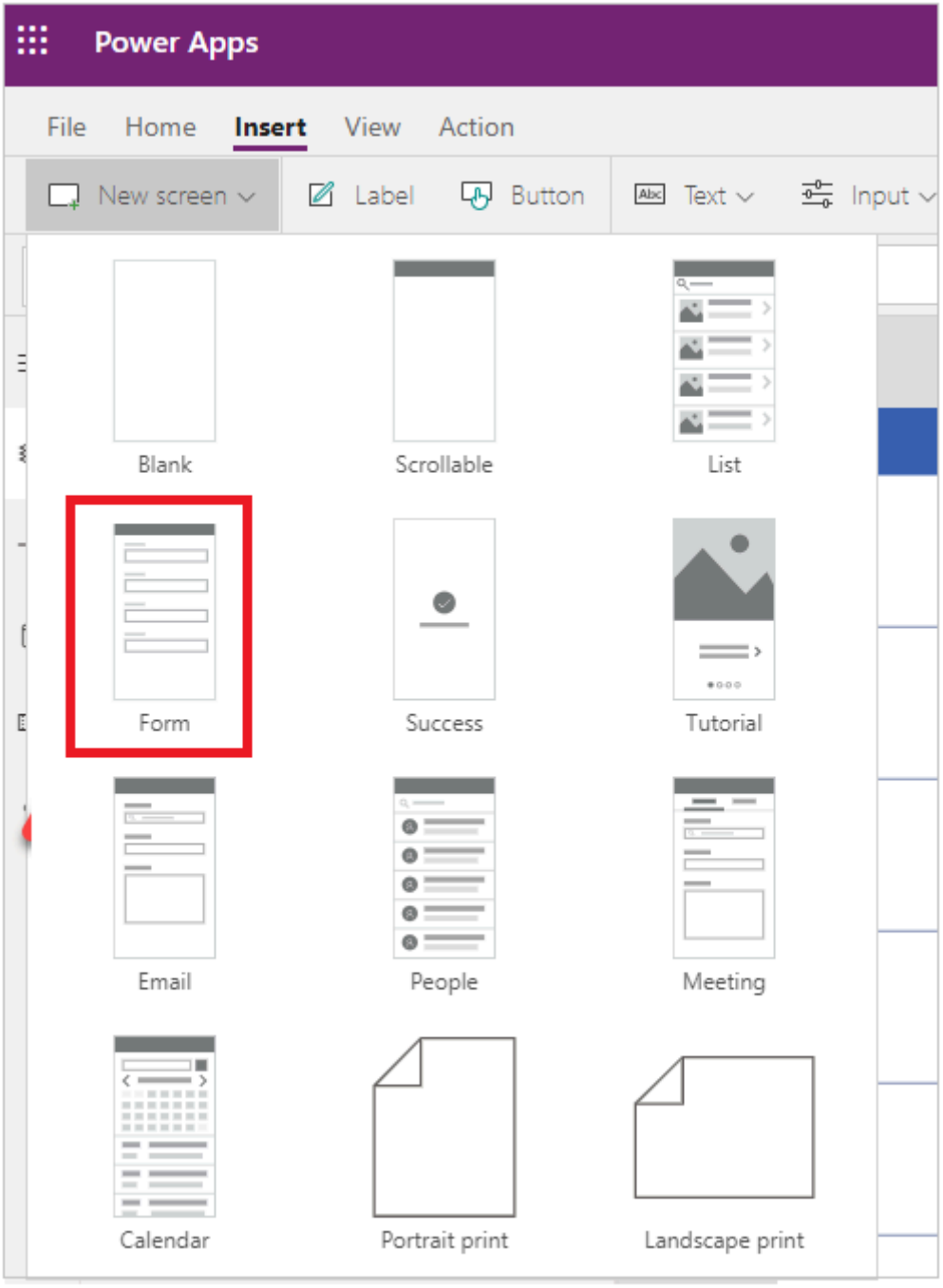
Property	Value
Font	Open Sans
Font size	21
Font weight	Bold
Text alignment	Center
Auto height	Off
Line height	1.2
Overflow	Hidden
Display mode	Edit
Visible	On
Position (X, Y)	0, 0
Size (Width, Height)	640, 71
Padding	5 Top, 5 Bottom, 5 Left, 5 Right



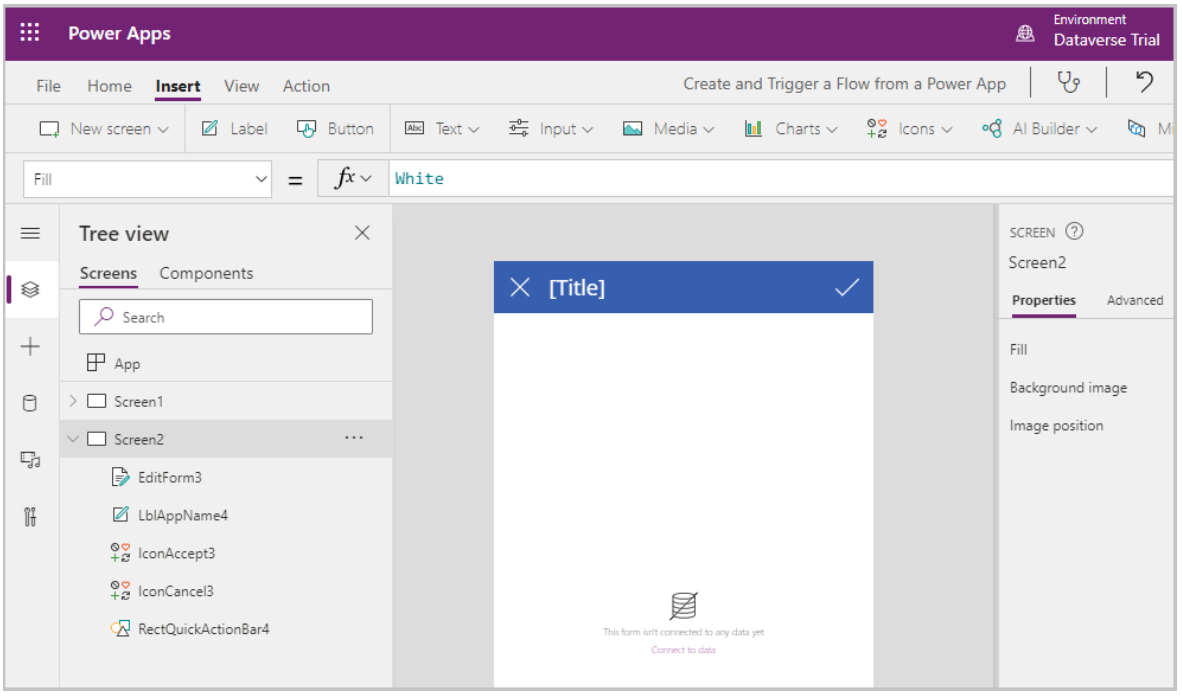
5. Add an **Add** icon to the top-right of the screen by selecting **Insert** > **Icon** > **Add**. And then, update the properties of the icon to color white and padding of 5 from all sides.



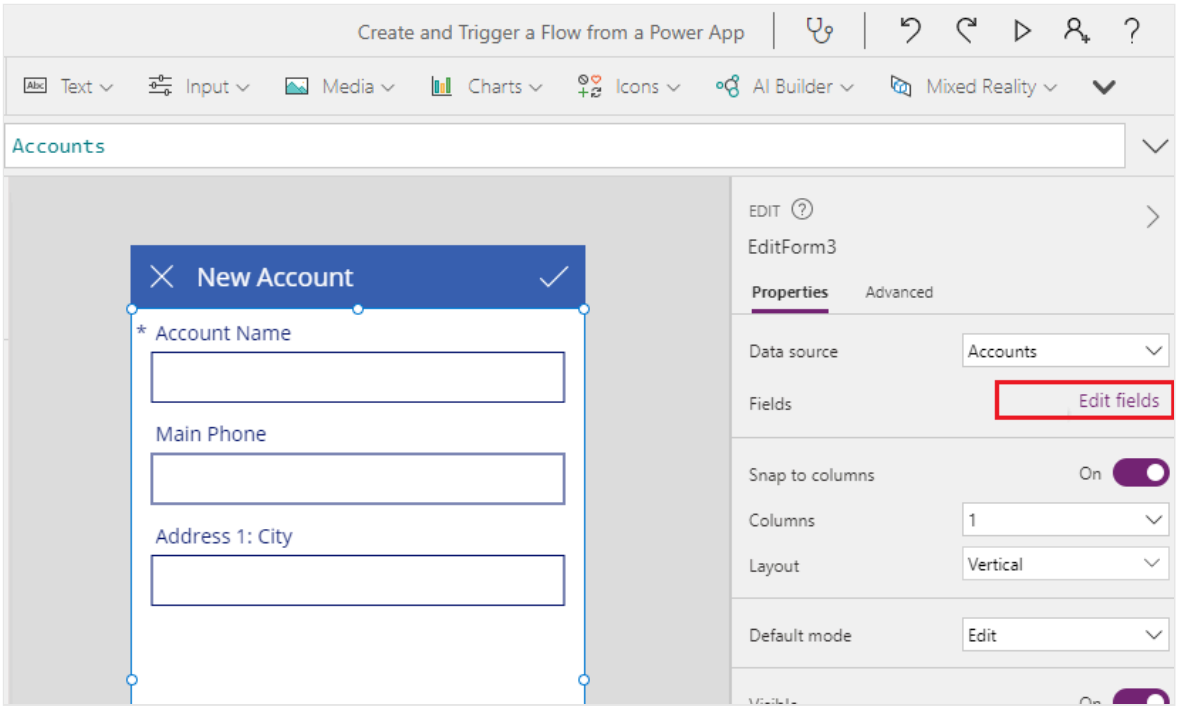
6. From the top-left side of the screen, select **New screen** > **Form**.



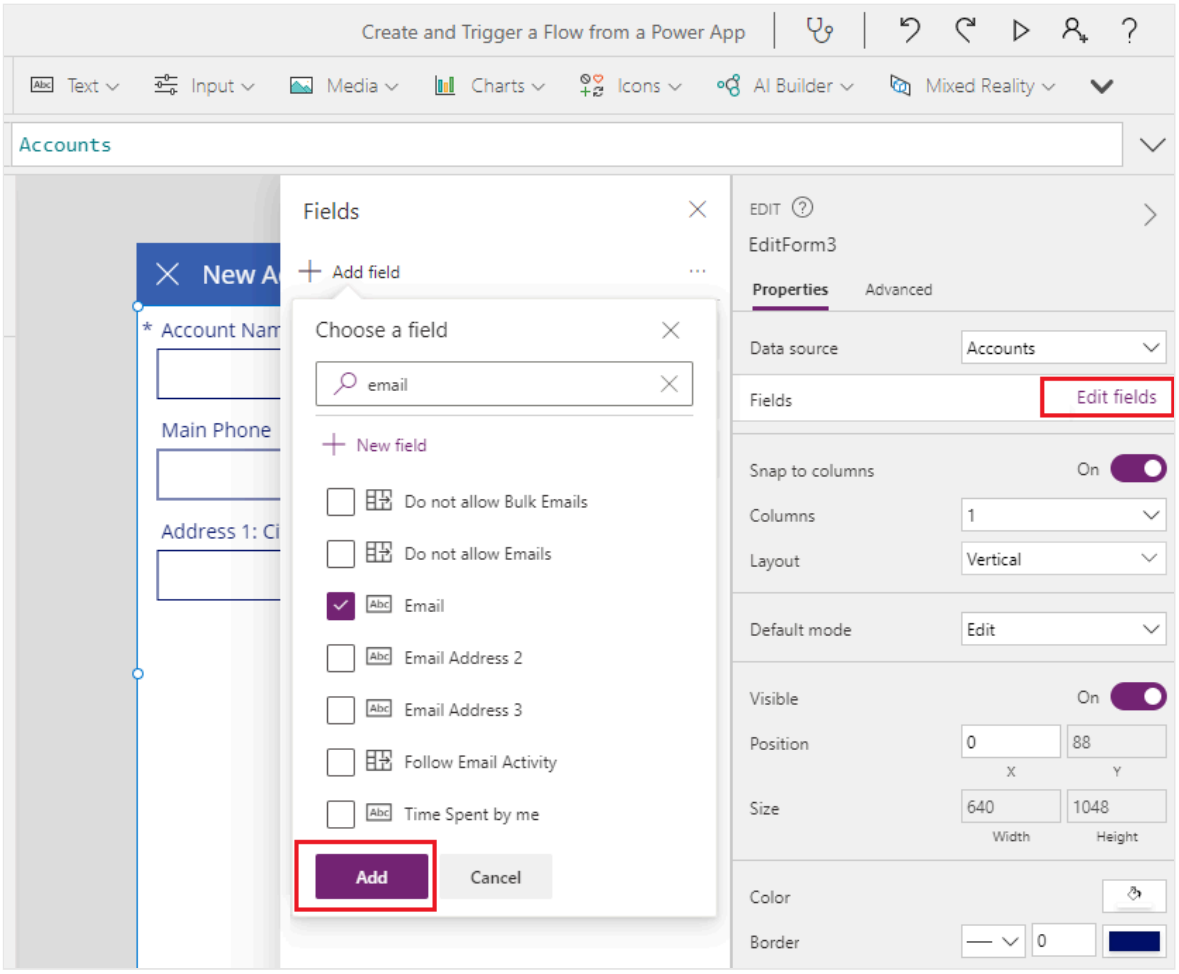
7. Update the title of the form to **New Account**, and select the data source as **Accounts** from the pane on the right-side of the screen.



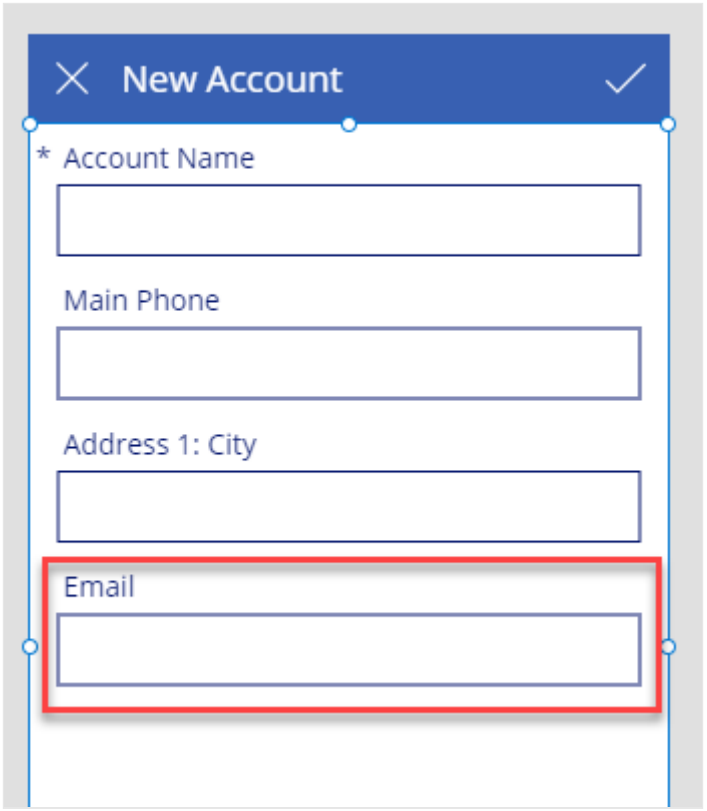
8. Select **Edit fields** on the right-pane.



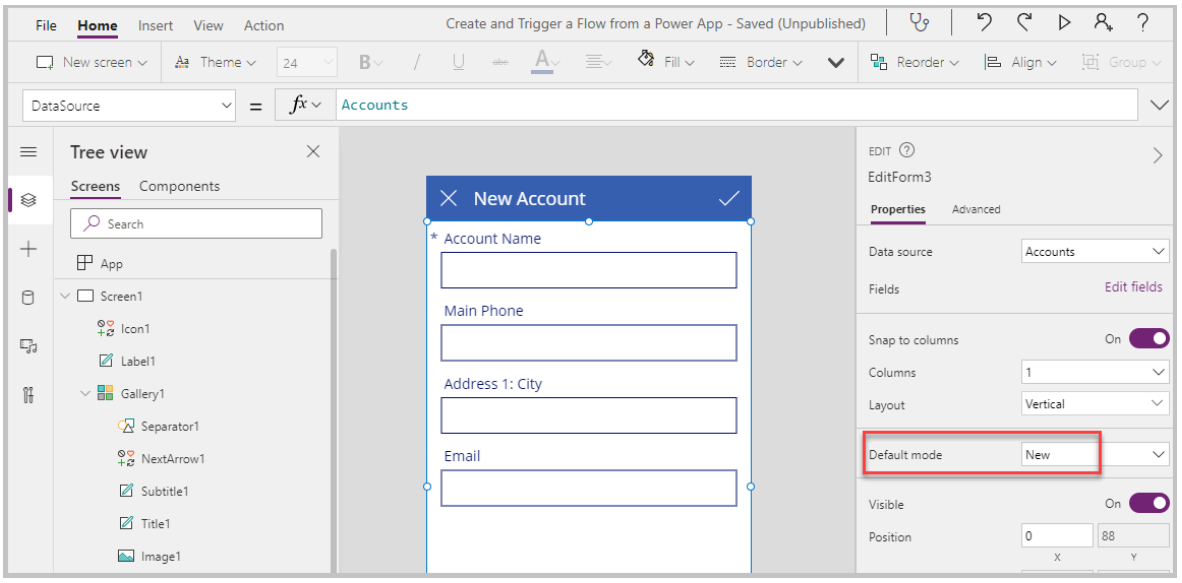
9. Select **Add field** and add **Email** to the form.



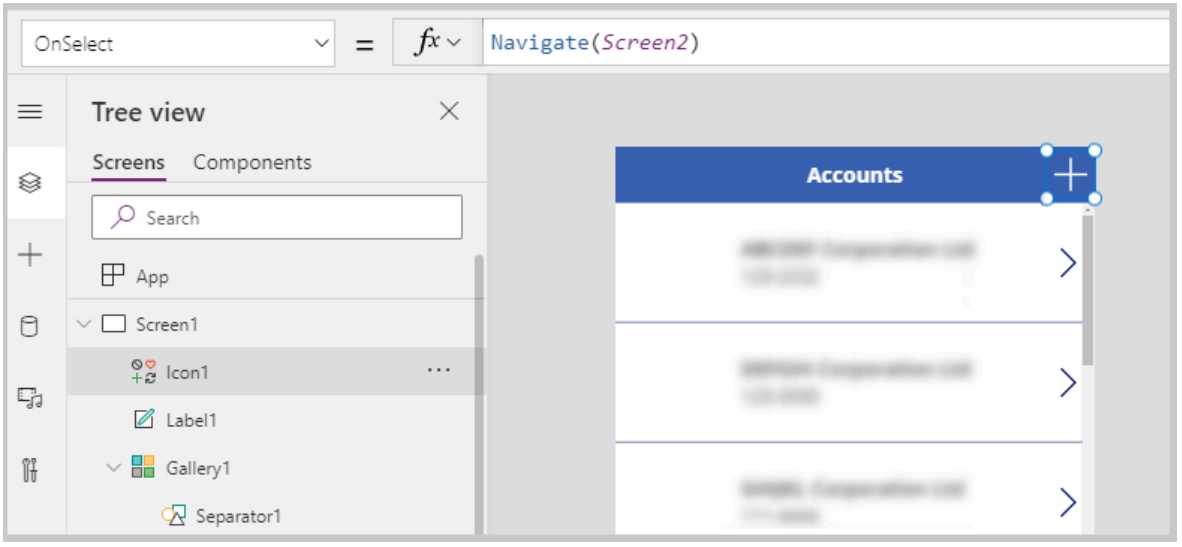
The email field gets added to the form.



10. Set the default mode of the form to **New**.

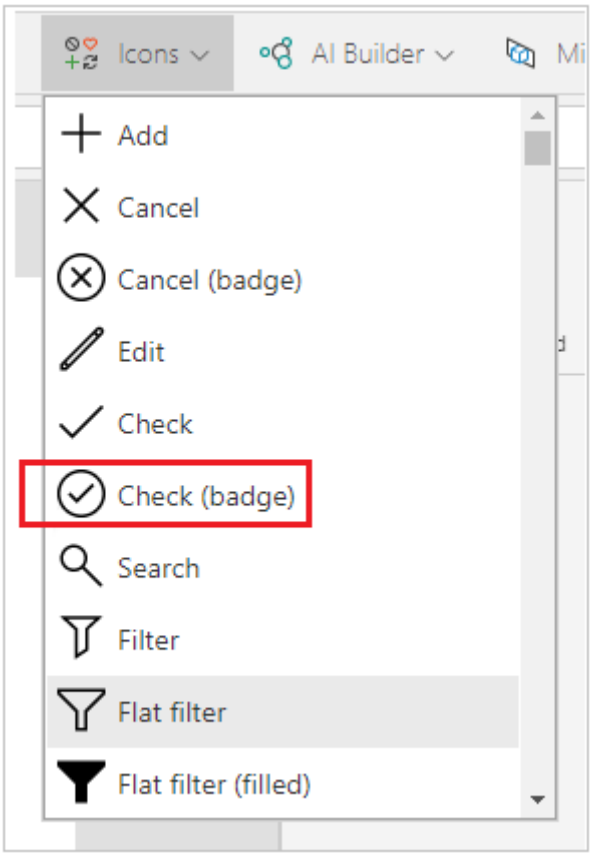


11. Select Screen1 that has the accounts gallery, and set the **OnSelect** property of the + icon to `Navigate(Screen2)`.

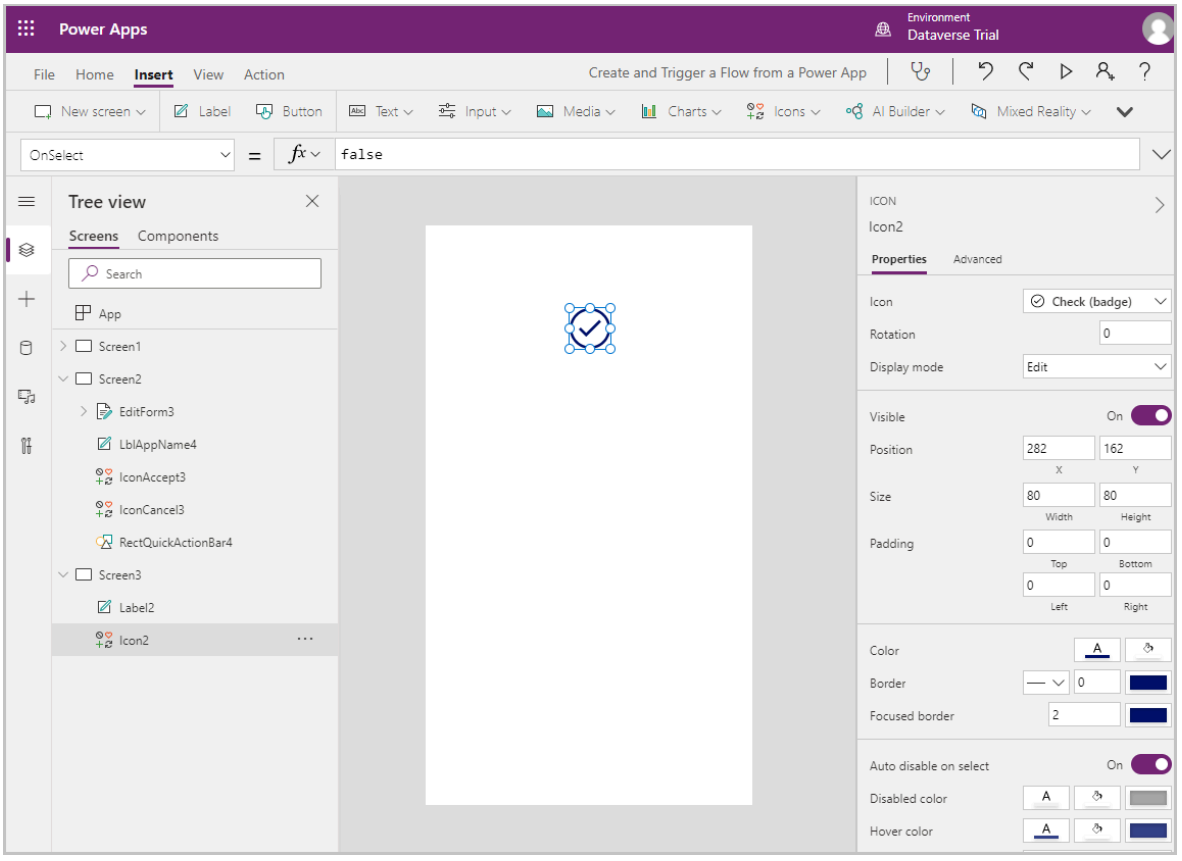


12. Select **Insert** > **New screen** > **Blank screen** to add a new screen to the app.

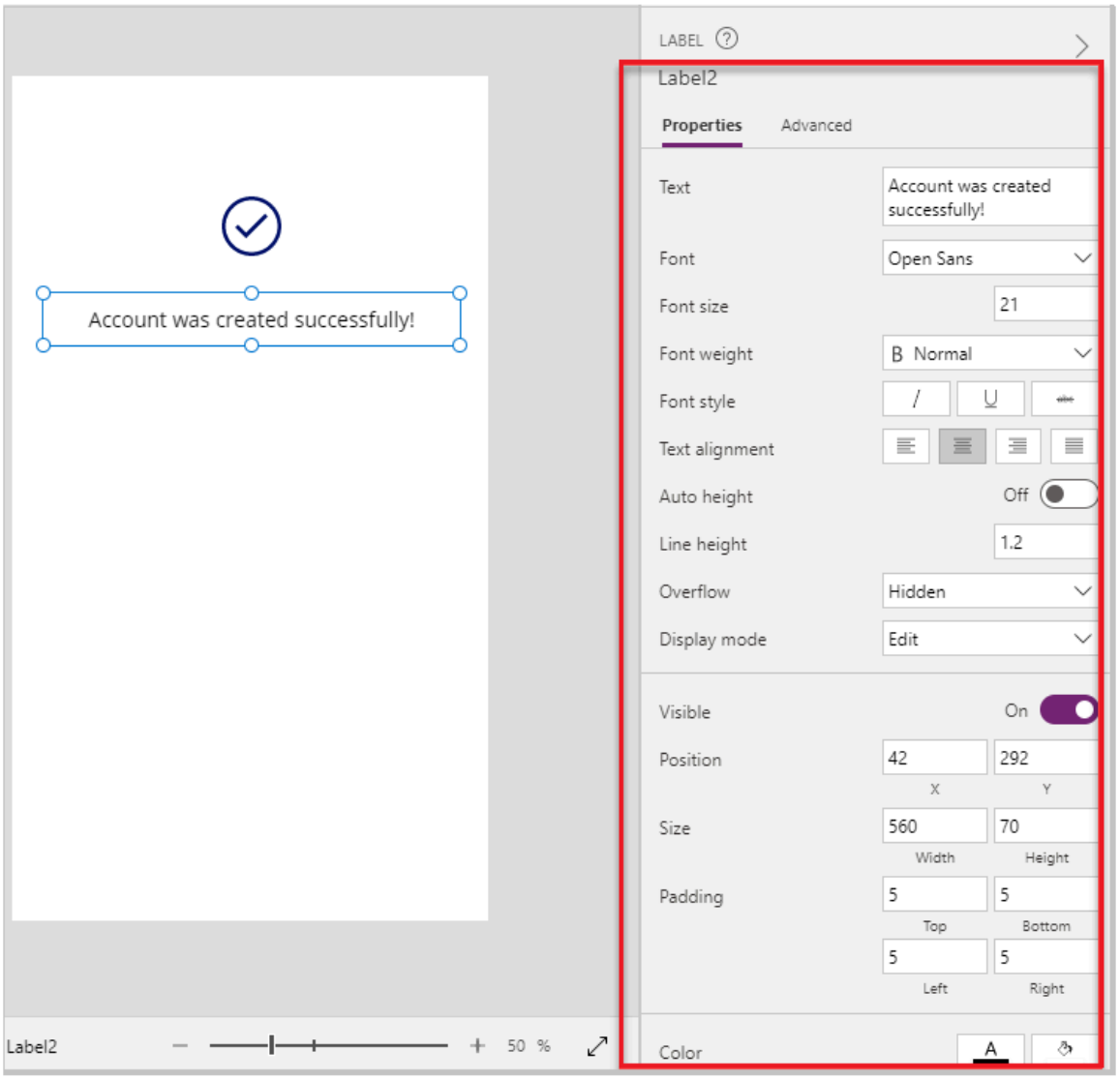
13. Select **Insert** > **Icon**, and then select the Check (badge) icon to add it to the screen.



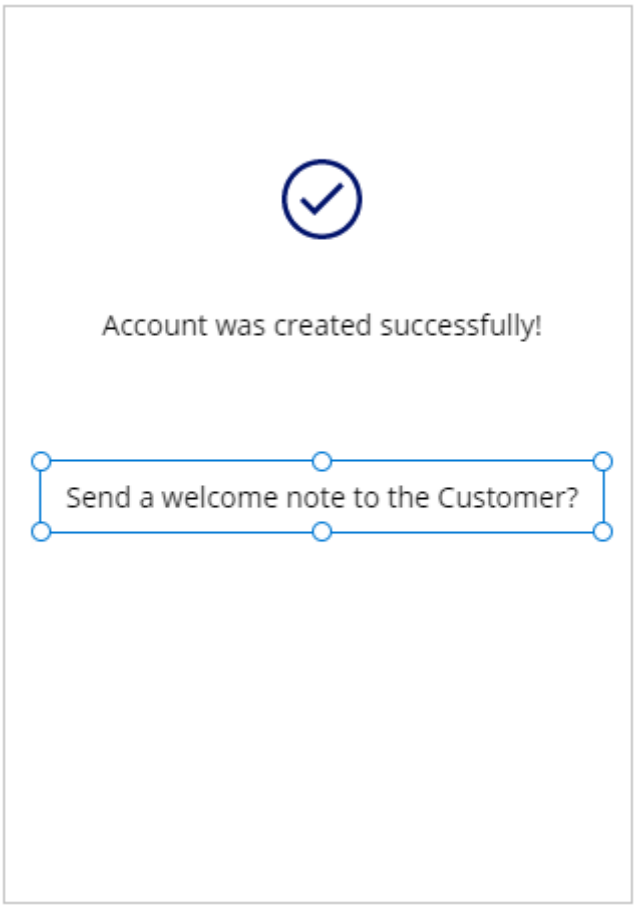
14. Move the icon to the top center of the canvas.



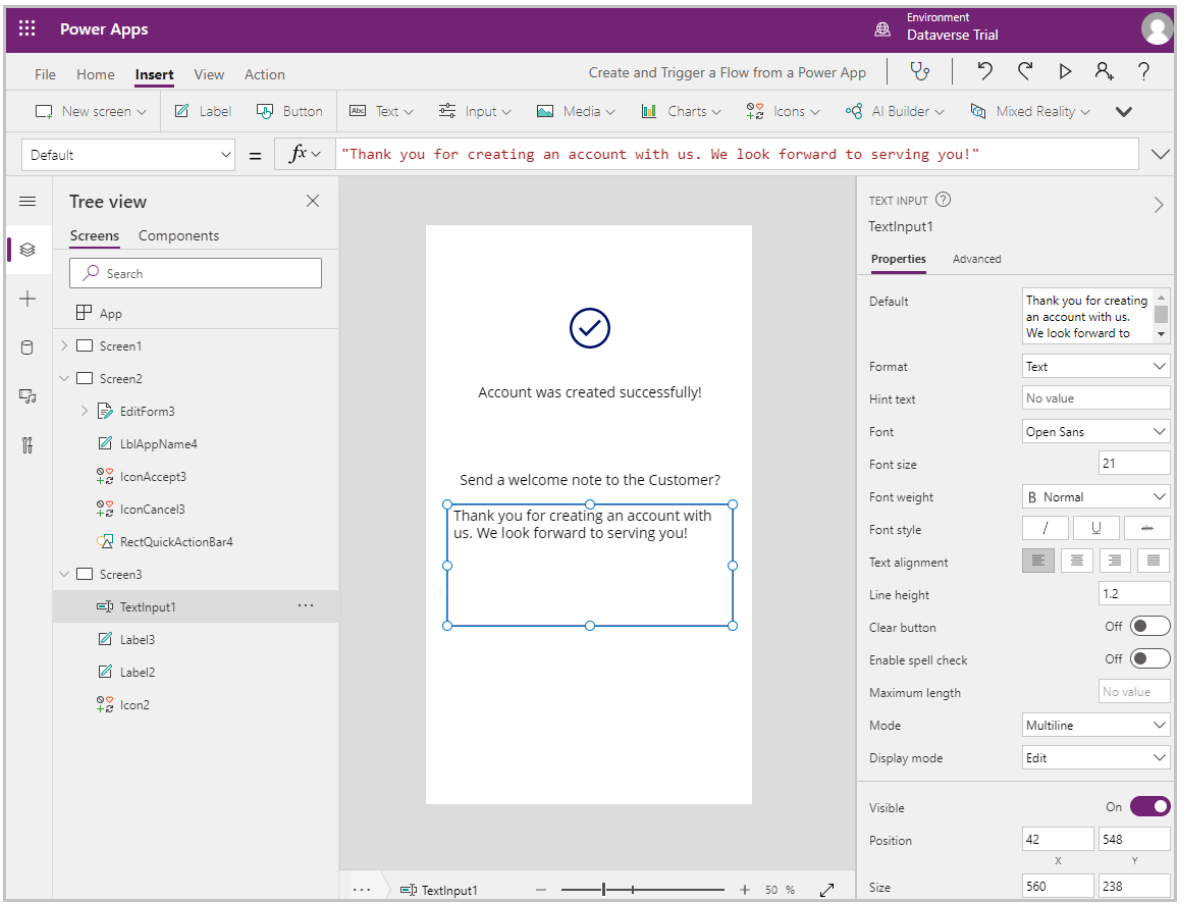
15. Add a label with text **Account was created successfully!**, and move it under the icon added in the earlier step.



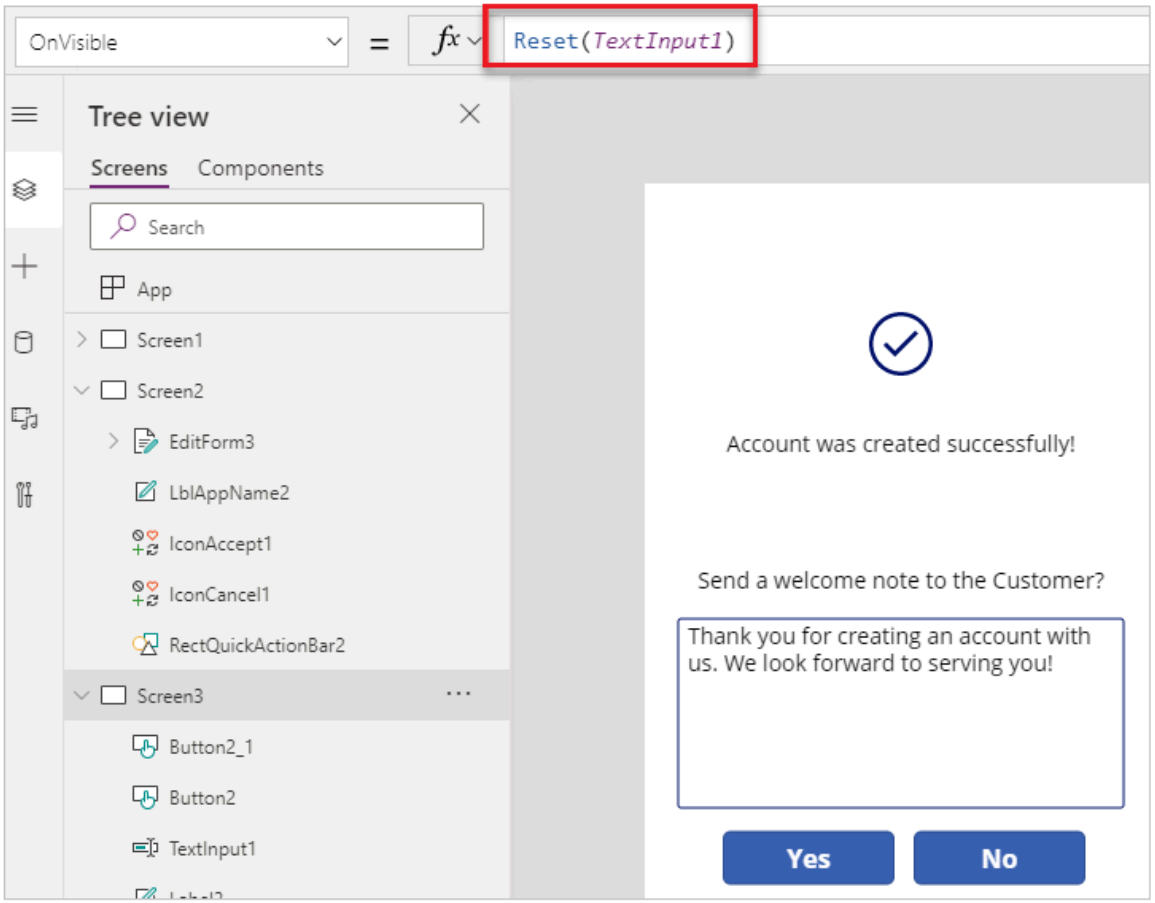
16. Add an additional label below the label added in the previous step with text **Send a welcome note to the Customer?**.



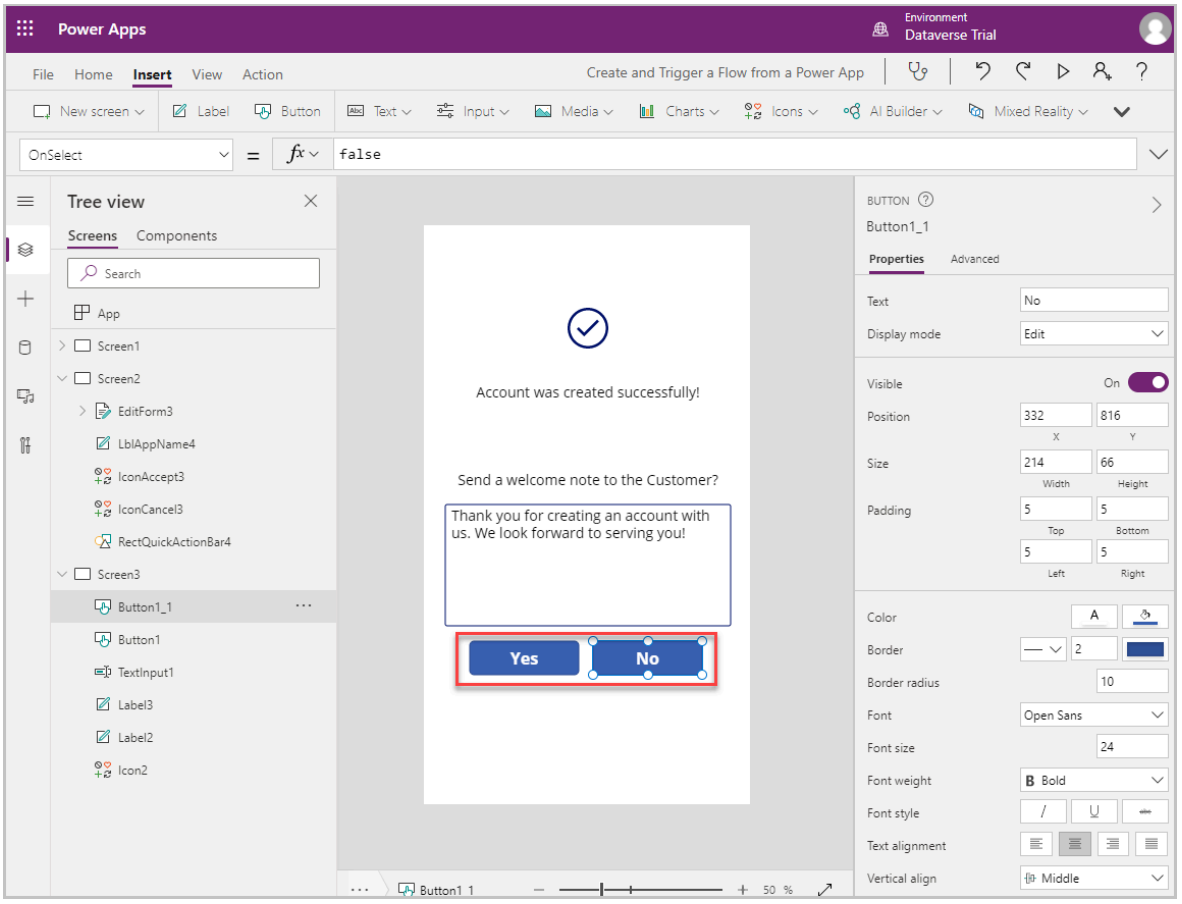
17. Add a **Text Input** box by selecting **Insert** > **Input** > **Text Input**. Set the **Mode** of the control to **Multiline**, and default text to **Thank you for creating an account with us. We look forward to serving you!**. And move the control below the label added in the previous step.



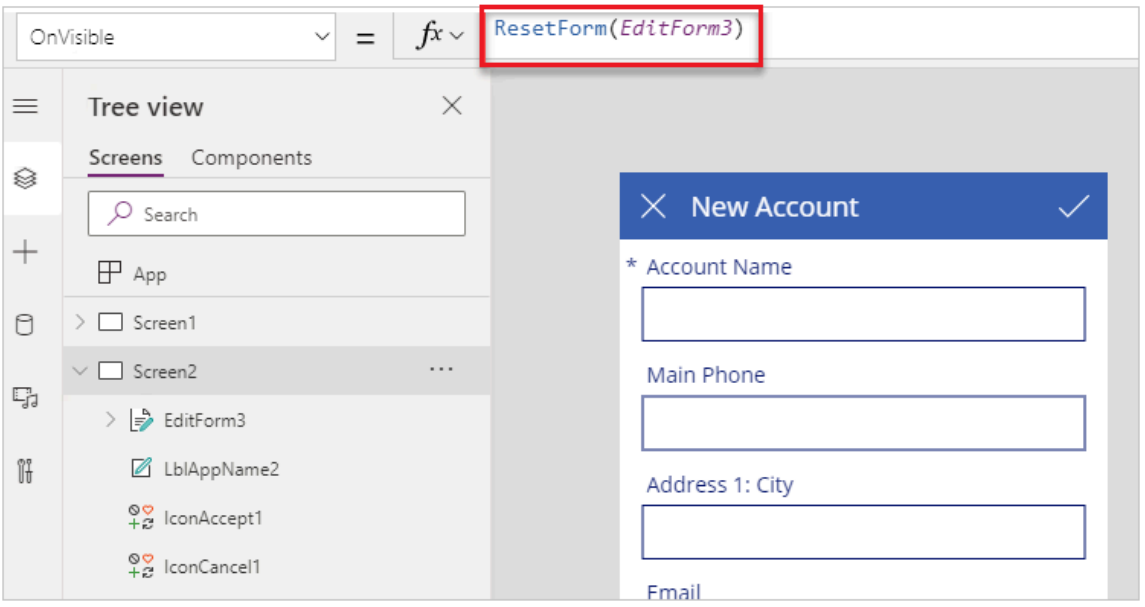
18. Set the **OnVisible** property of **Screen3** to `Reset(TextInput1)`.



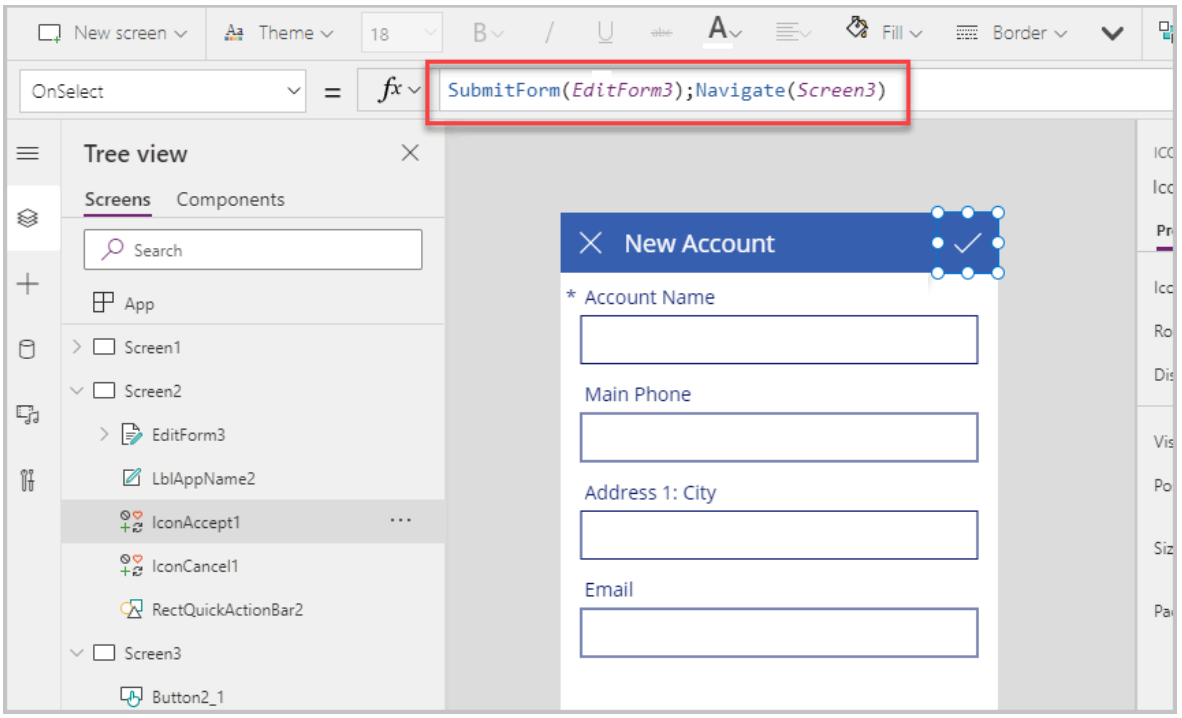
19. Add two buttons to the screen with text **Yes** and **No**, and place them below the input text box added in the previous step.



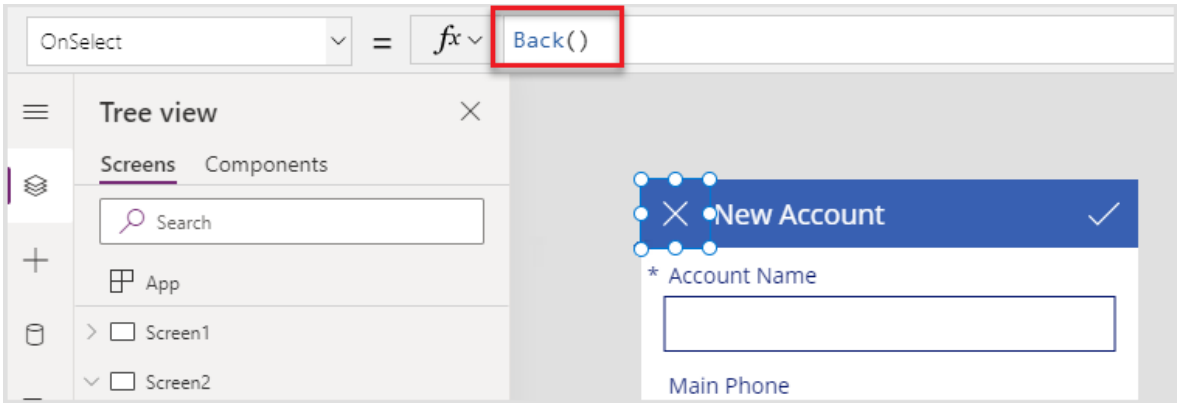
20. Go to the **New Account** form screen, and set the **OnVisible** property of the screen to `ResetForm(EditForm3)`.



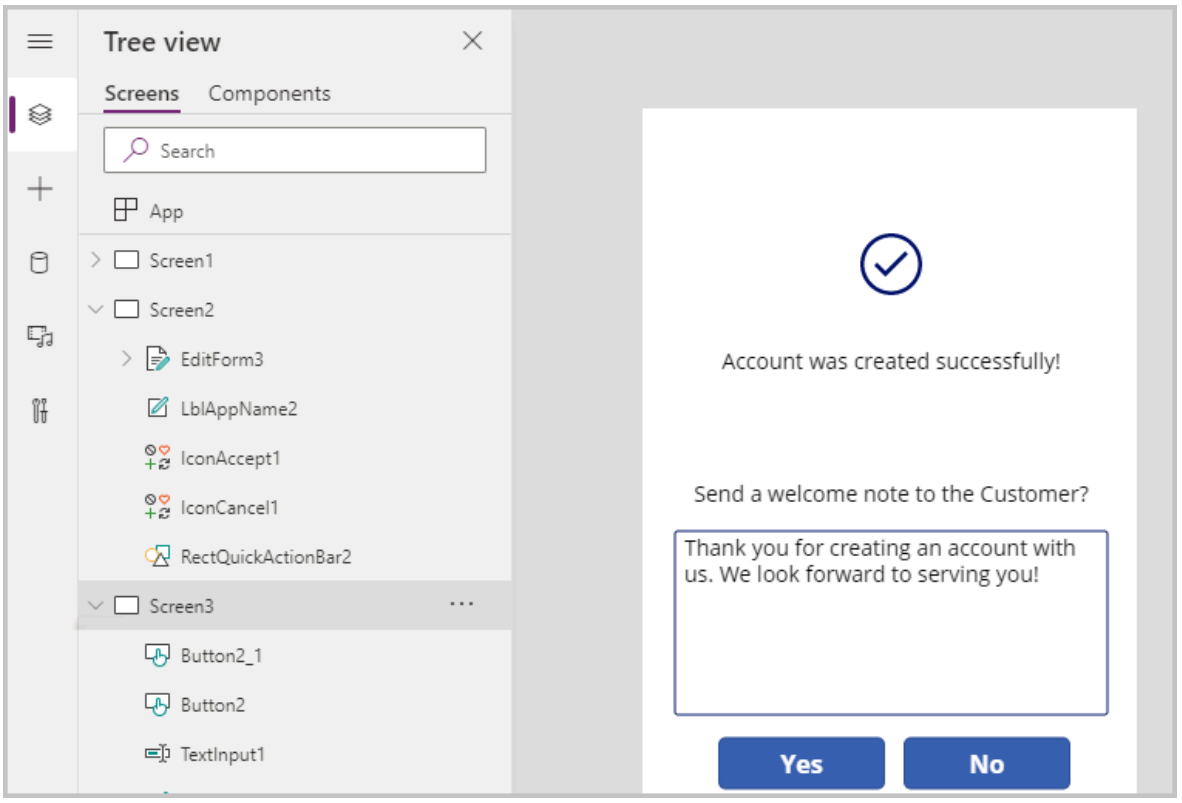
21. Select the check icon on the top-right, and set the **OnSelect** property to `SubmitForm(EditForm3);Navigate(Screen3)`.



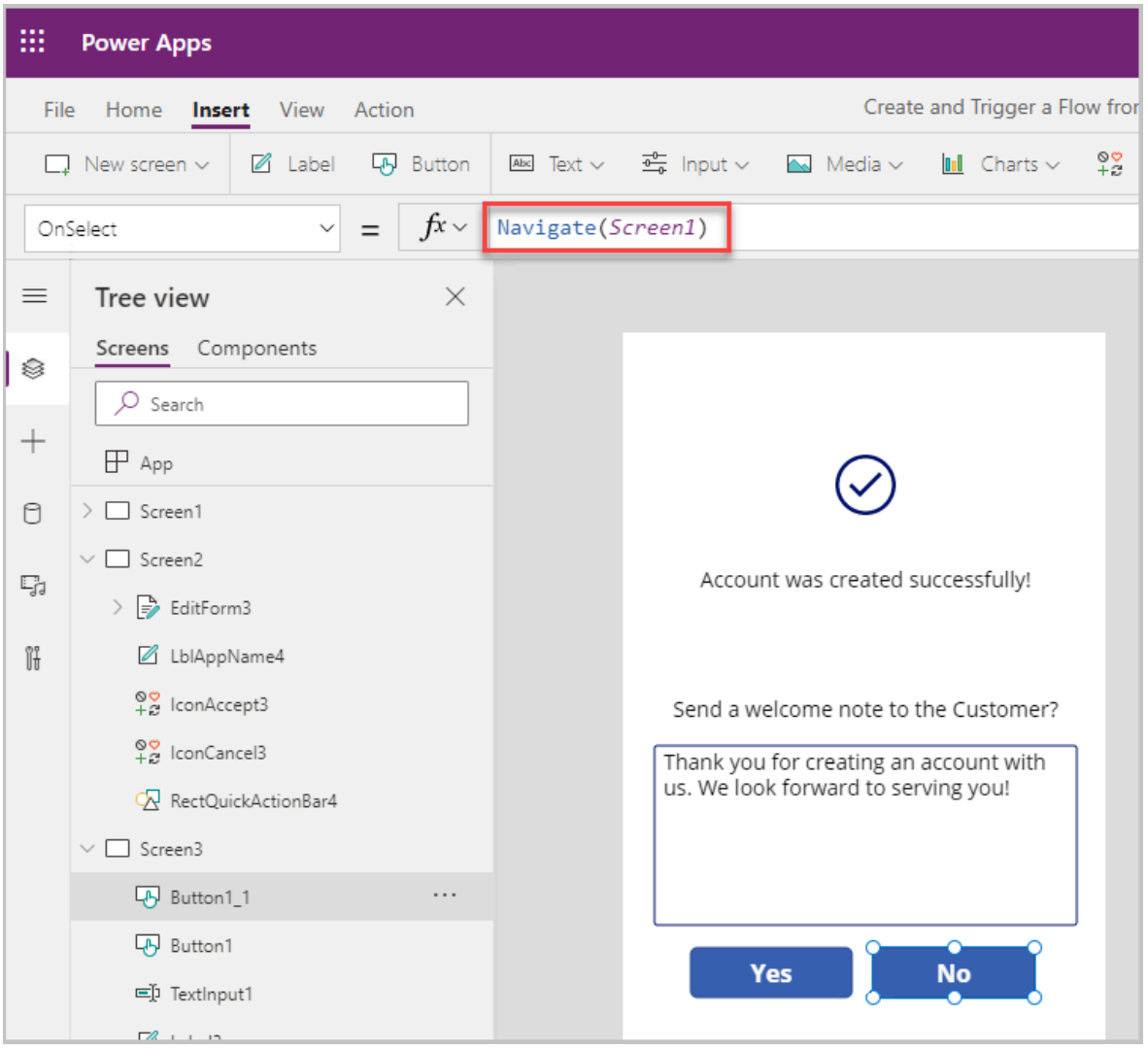
22. Select the X button on the top-right, and set the **OnSelect** property to `Back()`.



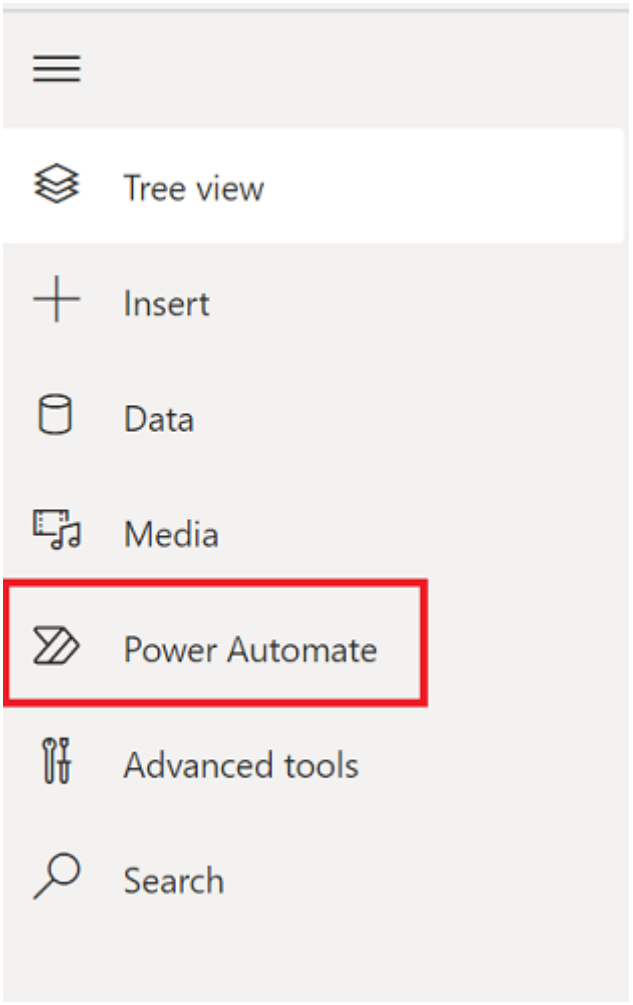
23. Go back to **Screen3** by selecting **Screen3** in the tree view.



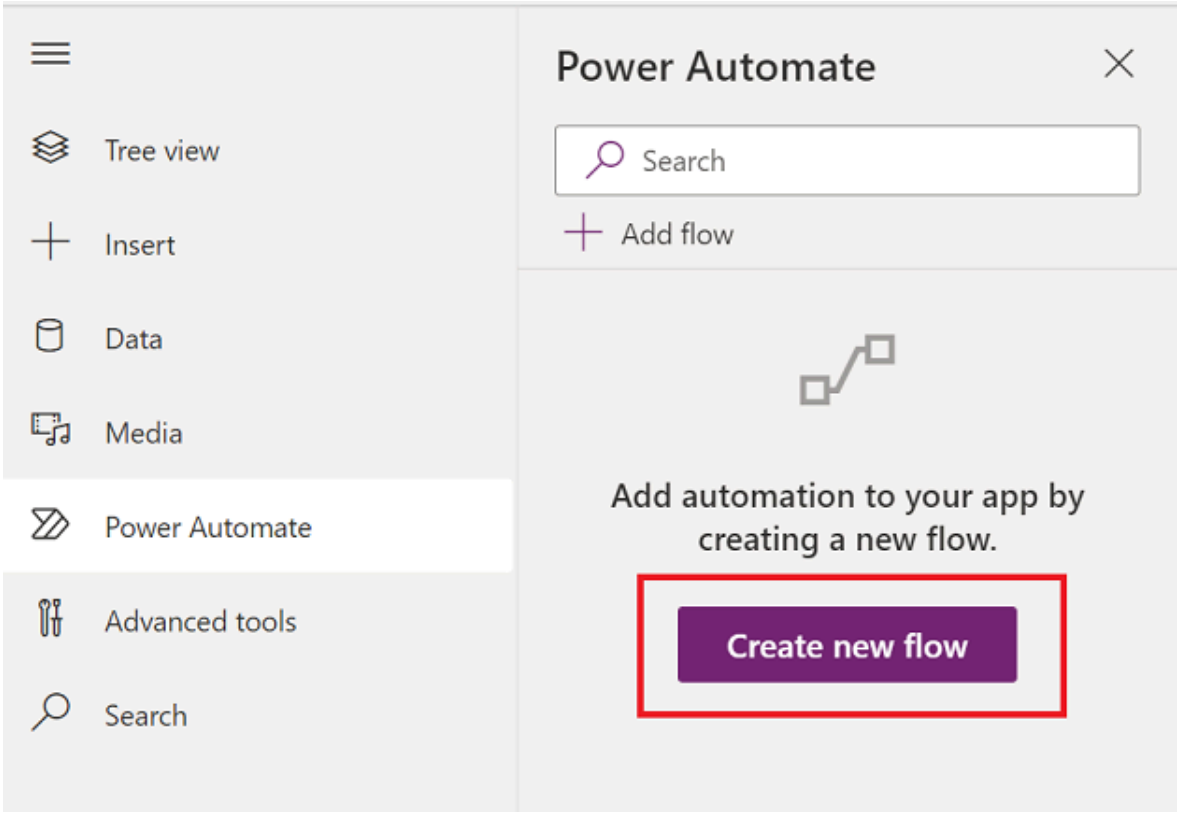
24. Select the **No** button, and set the button's **OnSelect** property to `Navigate(Screen1)`.



25. Select the **Yes** button, on the app authoring menu select **Power Automate**.

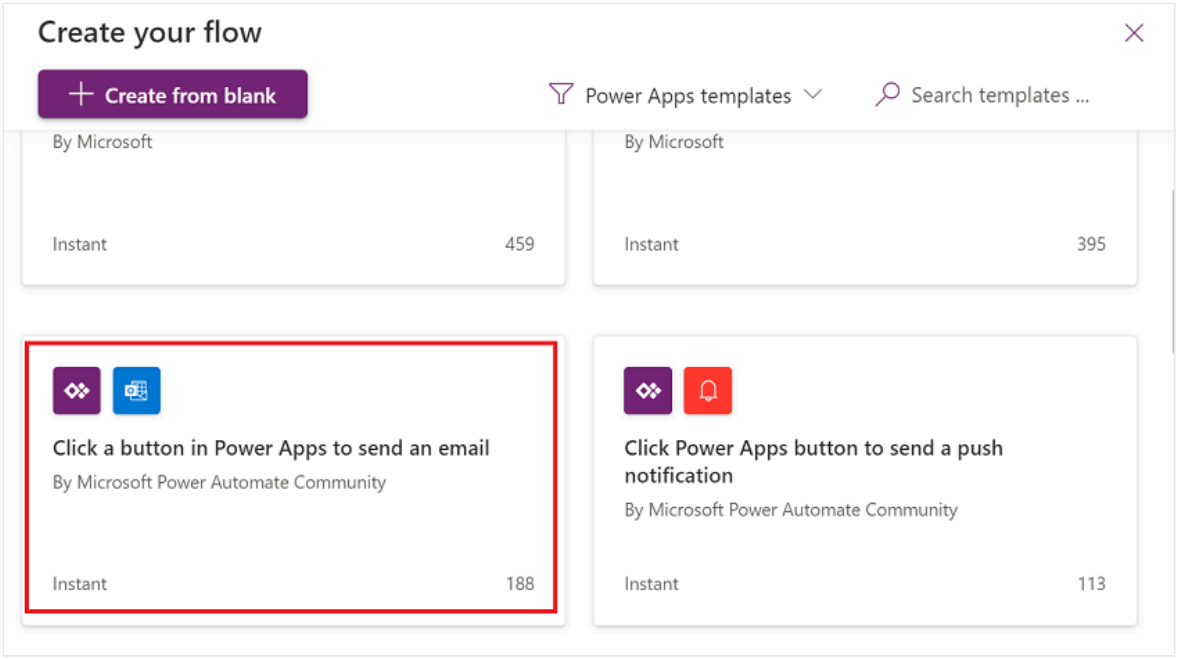


26. Select **Create a new flow** to create a new Power Automate flow.

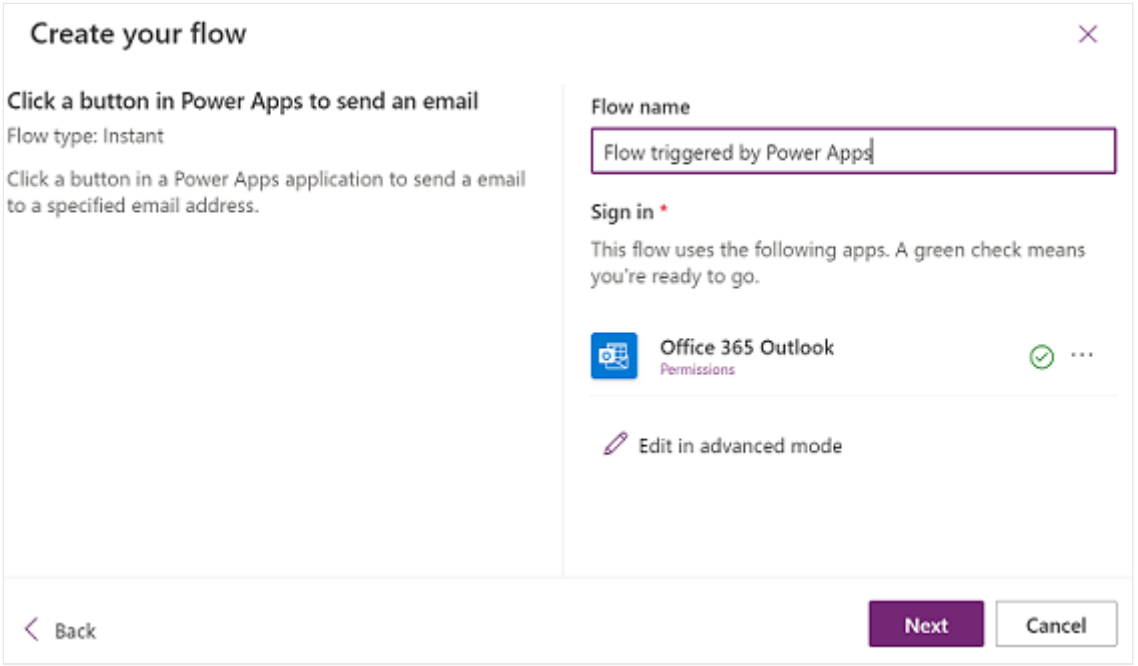


Create the Power Automate Flow

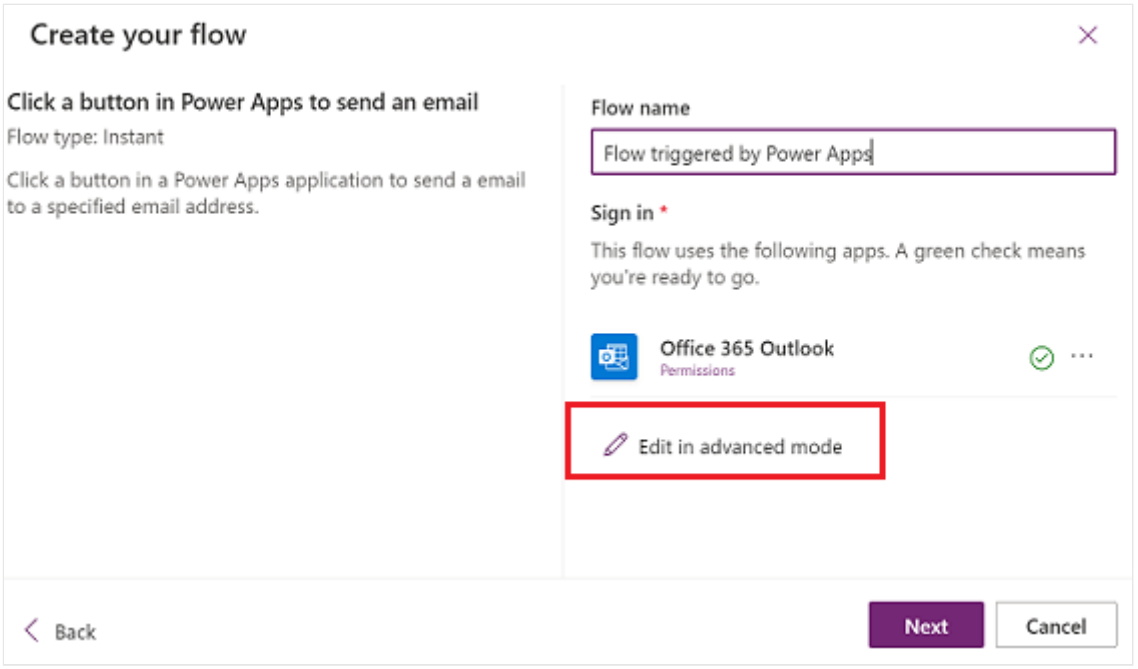
1. On the **Create your flow** screen, from the list of instant templates, select **Click a button in Power Apps to send an email**.



2. Enter a name for the flow as "Flow triggered by Power Apps", and select **Power Apps** as the trigger.



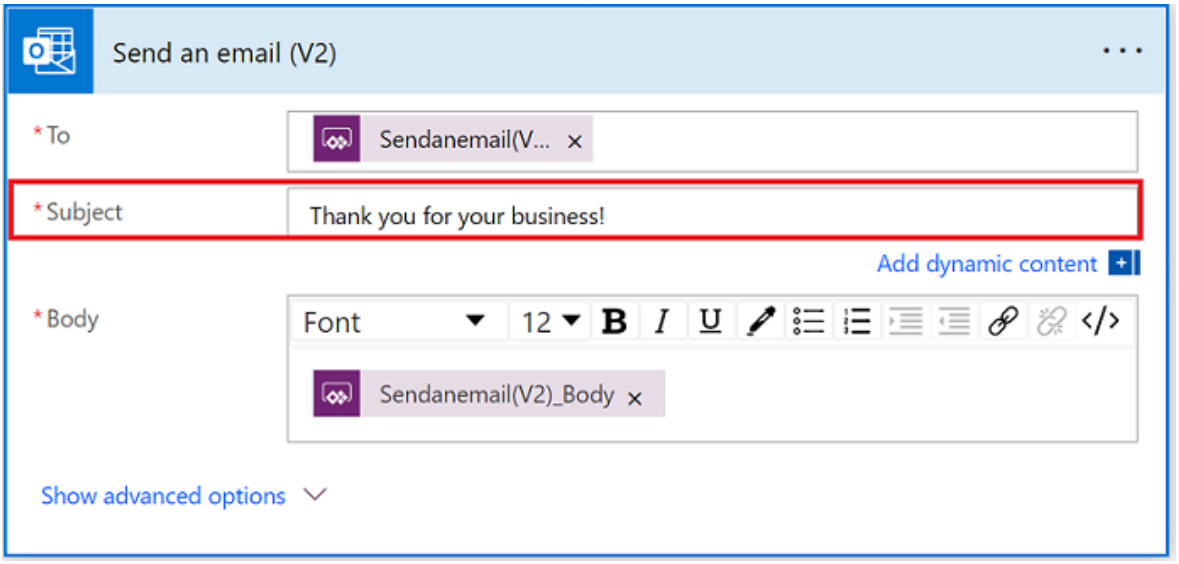
3. Select **Edit in advanced mode** and then select, **Continue**.



4. The **To** field and the **Body** field are automatically auto-populated with the following:

    - To: **Sendanemail(V2)_To**.
    - Body: **Sendanemail(V2)_Body**

5. In **Subject** field, delete **Sendanemail(V2)_To** and enten **Thank you for your business!** in the **Subject** field.



6. Select **Save** to save the flow.

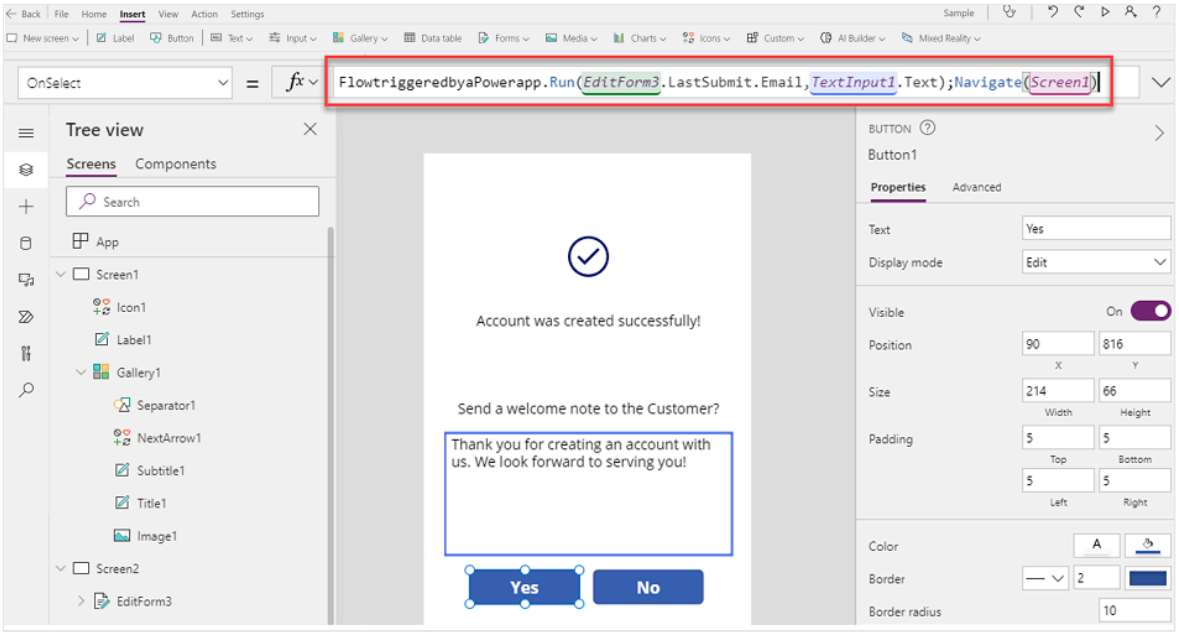
Trigger the flow from within the canvas app

1. Select the **Yes** button > in the **OnSelect** property enter the two parameters for the To email address and the Body of the email as follows:

Power Fx

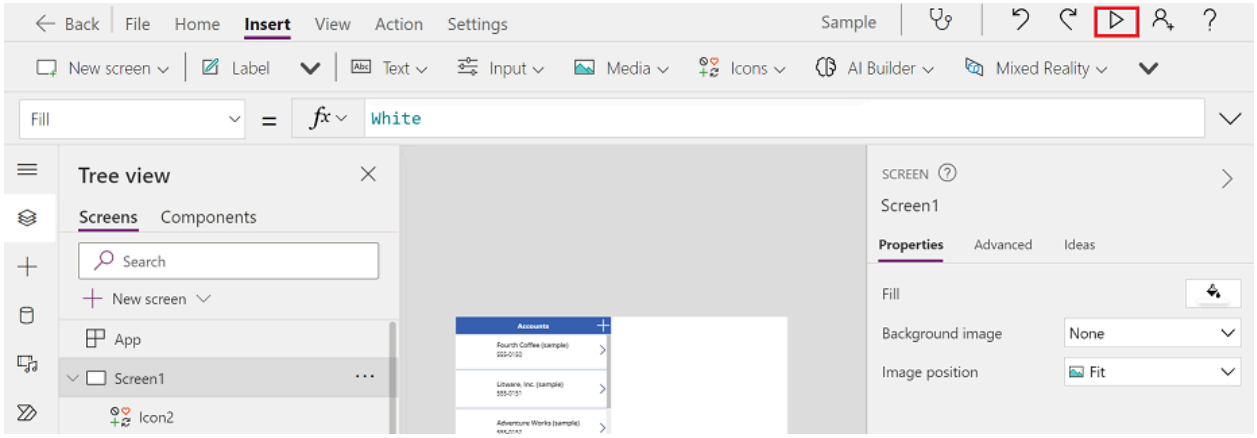
```
FlowtriggeredbyaPowerapp.Run(
   EditForm3.LastSubmit.Email,
   TextInput1.Text
);
Navigate(Screen1);
```

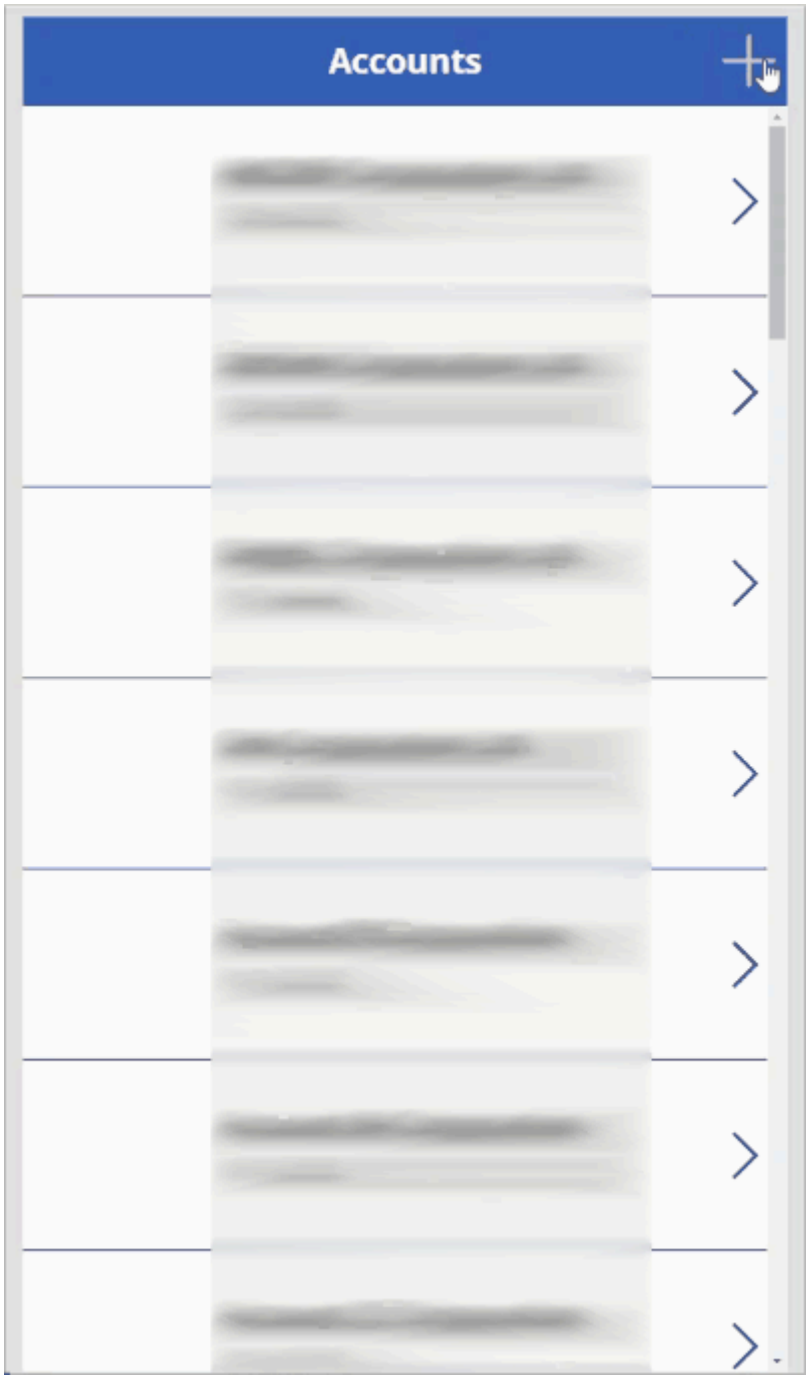
> **Note**
>
> The email address is picked up from the new account record created on the previous screen. and the body for the email is picked from the text box text captured on this screen.



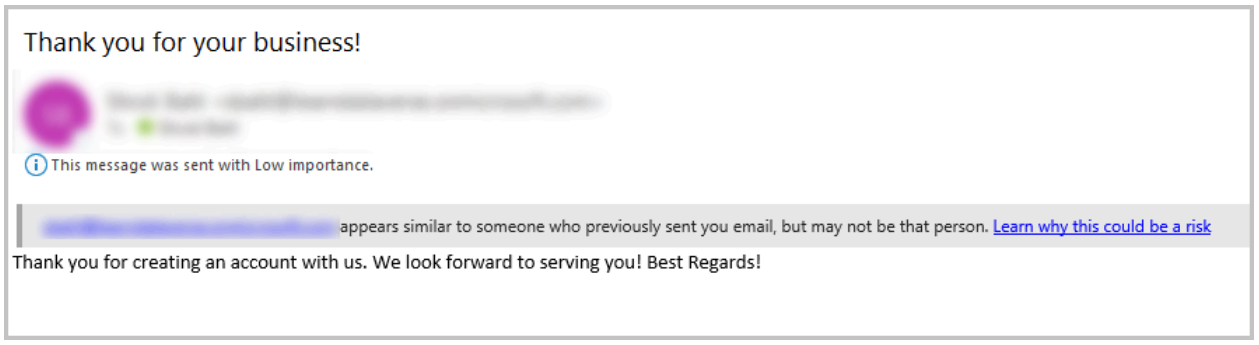
Test the app

Run the app in preview mode. In this test, we'll create a new account by entering details like account name, phone number, city, and email address on the **New Account** screen. On saving the new account details, we'll be prompted to send an email to the customer where we can update the verbiage of the email, and then select **Yes** to send the email. This will trigger the flow, and email will be sent to the email address on the account.





An email like this should appear in your inbox



See also

Use Power Automate pane

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a canvas app with the barcode reader control

Article • 03/13/2023

The barcode reader control in Power Apps lets you use your phone or mobile device to scan barcodes from various formats.

Some of the common uses of barcode reader control in a canvas app are:

- An app to check in items like books into a collection
- An inventory management app that uses barcode to identify items
- An employee app to check in and badge scanning for security

In this article, we'll create a canvas app with barcode reader and display scanned items in a gallery.

Watch this video to learn how to create an app with barcode reader: https://learn-video.azurefd.net/vod/player?id=8714cb52-ce37-452c-bc47-1af24b1f6b93&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fcreate-app-barcode-scanner

Prerequisites

- Power Apps license
- Before you create an app from scratch, familiarize yourself with Power Apps basics by generating an app and then customizing that app's controls, gallery, forms, and cards.
- To create an app, you must be assigned to the Environment Maker security role.

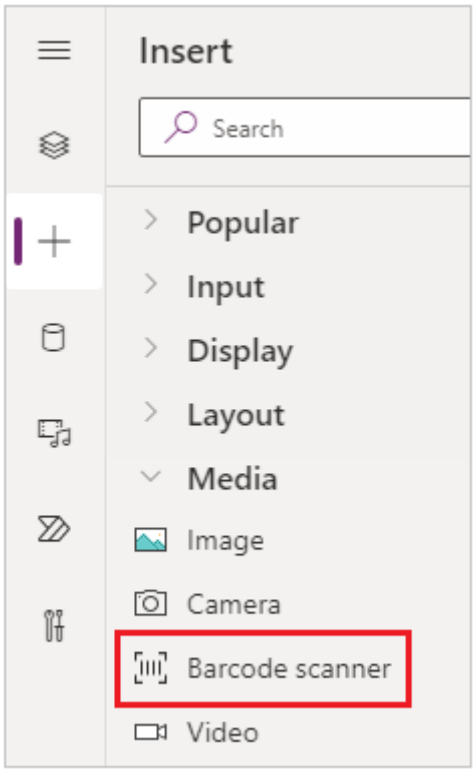
Open a blank app

Create a blank canvas app.

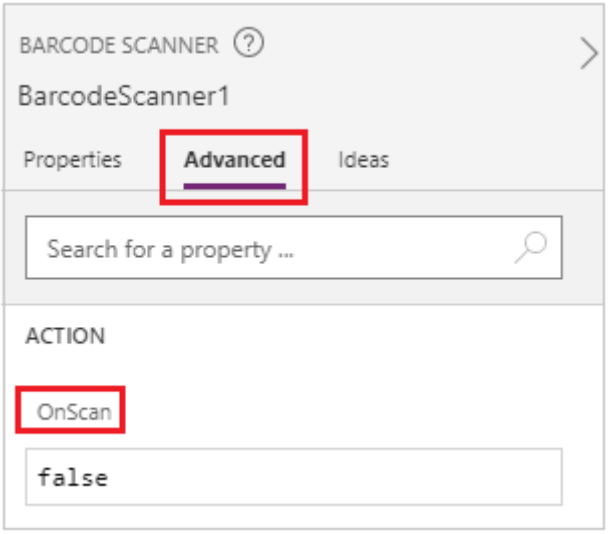
Add barcode scanner

1. From the left-pane, select **Screen1**.

2. From the left-pane, select **+ Insert** > expand **Media** > select **Barcode reader** control.



3. From the properties list on the right-side of the screen, select **Advanced** tab, and then select the **OnScan** property.

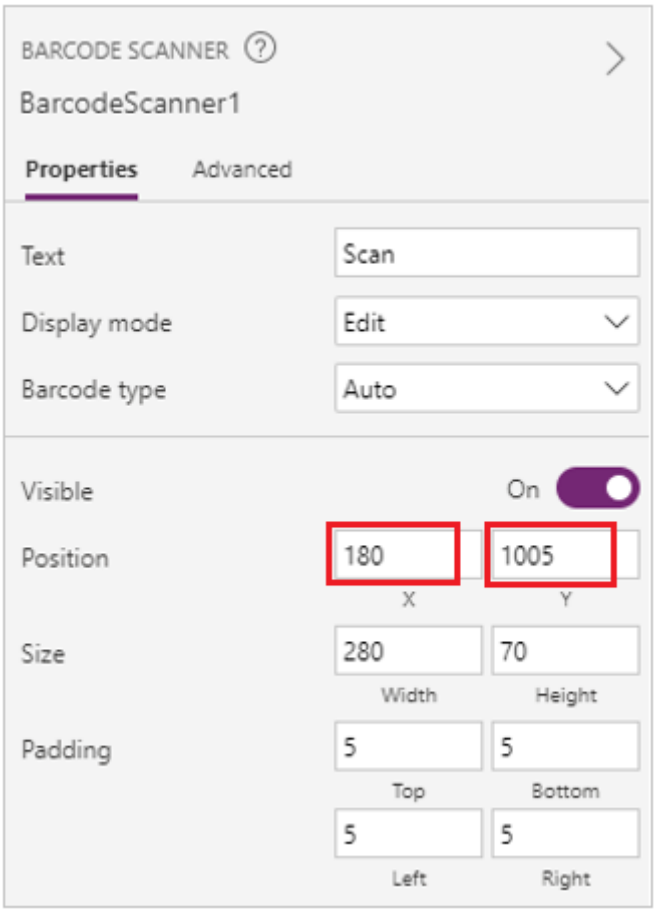


4. Set the **OnScan** property of the Barcode scanner control to this expression by typing or pasting it in the formula bar.

Power Fx

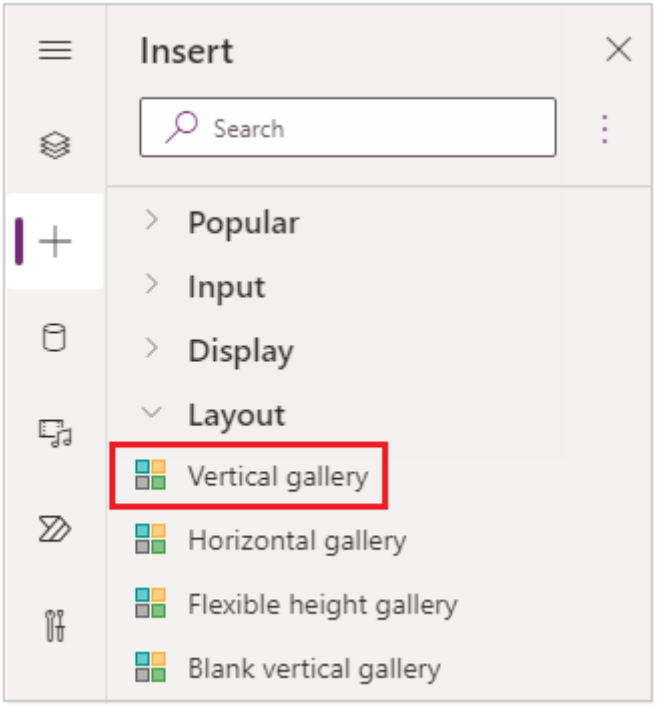
```
Collect(
    colScannedItems,
    {ScannedItem: First(BarcodeReader1.Barcodes).Value, ScannedTime:
Now()}
)
```

5. From the **Properties** pane, set **X** property to "180" and the **Y** property to "1005".

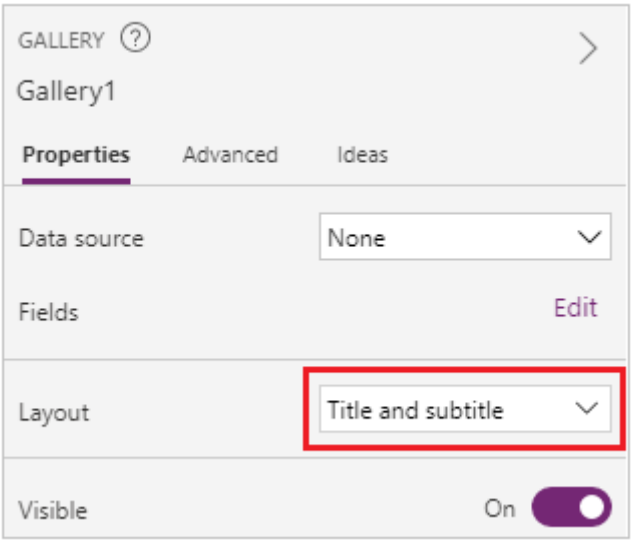


Add gallery

1. Select + **Insert** > expand **Layout** > select **Vertical gallery** control.



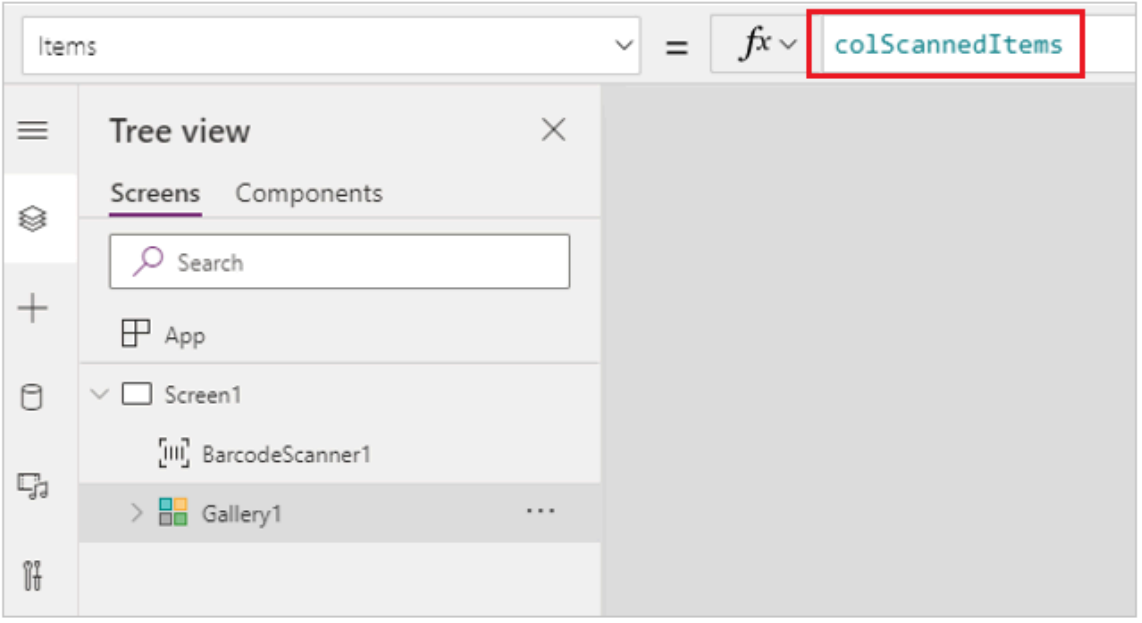
2. From the properties pane, select the layout for the gallery as **Title and subtitle**.



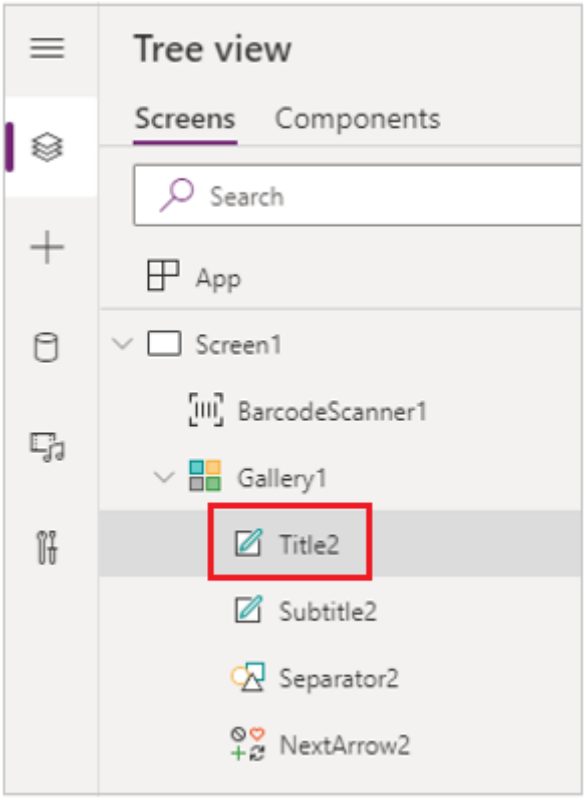
3. Select the gallery control on canvas.

4. Set the **Items** property of the gallery control to this expression by typing or pasting it in the formula bar.

```
Power Fx

colScannedItems
```



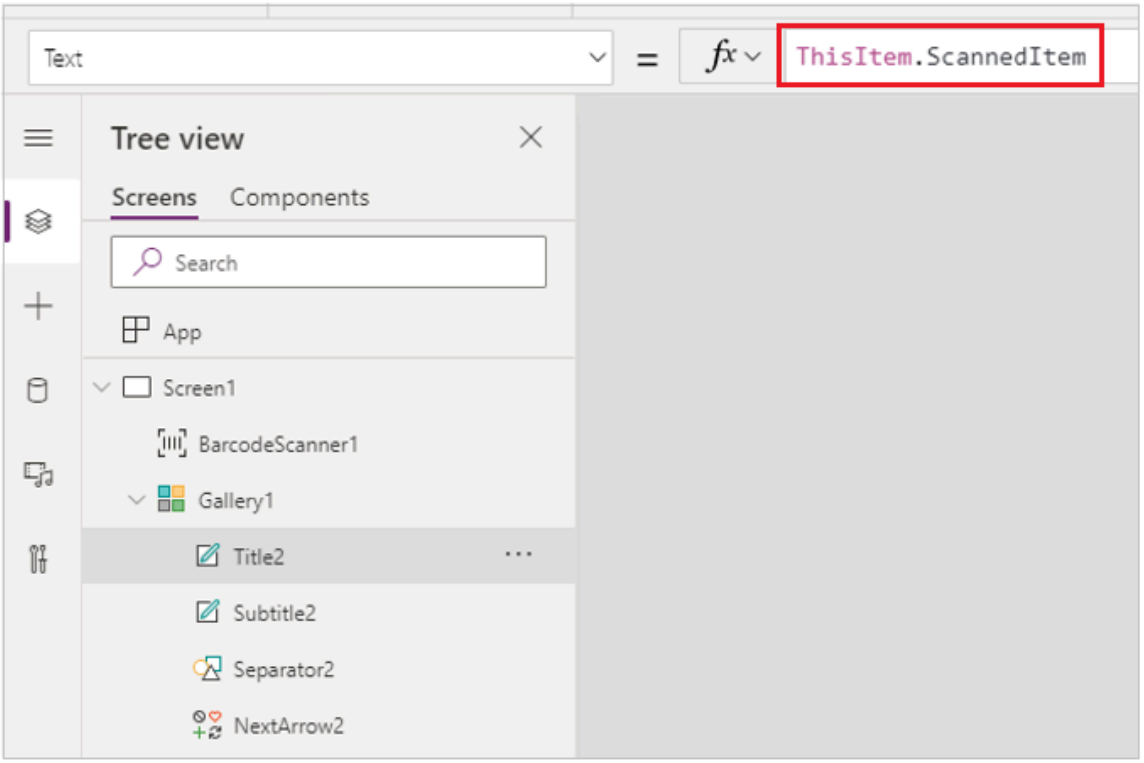
5. In the left-pane, expand the **Gallery1**, and select **Title2**.



6. Set the **Text** property of **Title2** to the following expression.

Power Fx

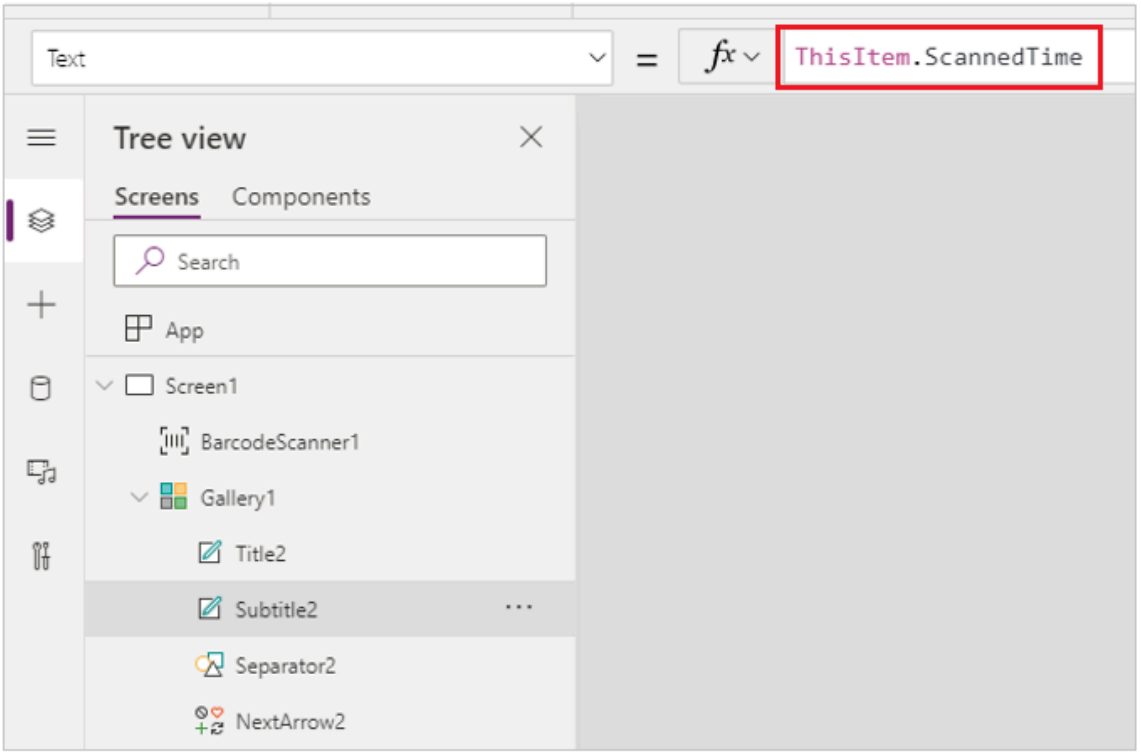
```
ThisItem.ScannedItem
```



7. Select **Subtitle2** label, and set it's **Text** property to this expression.

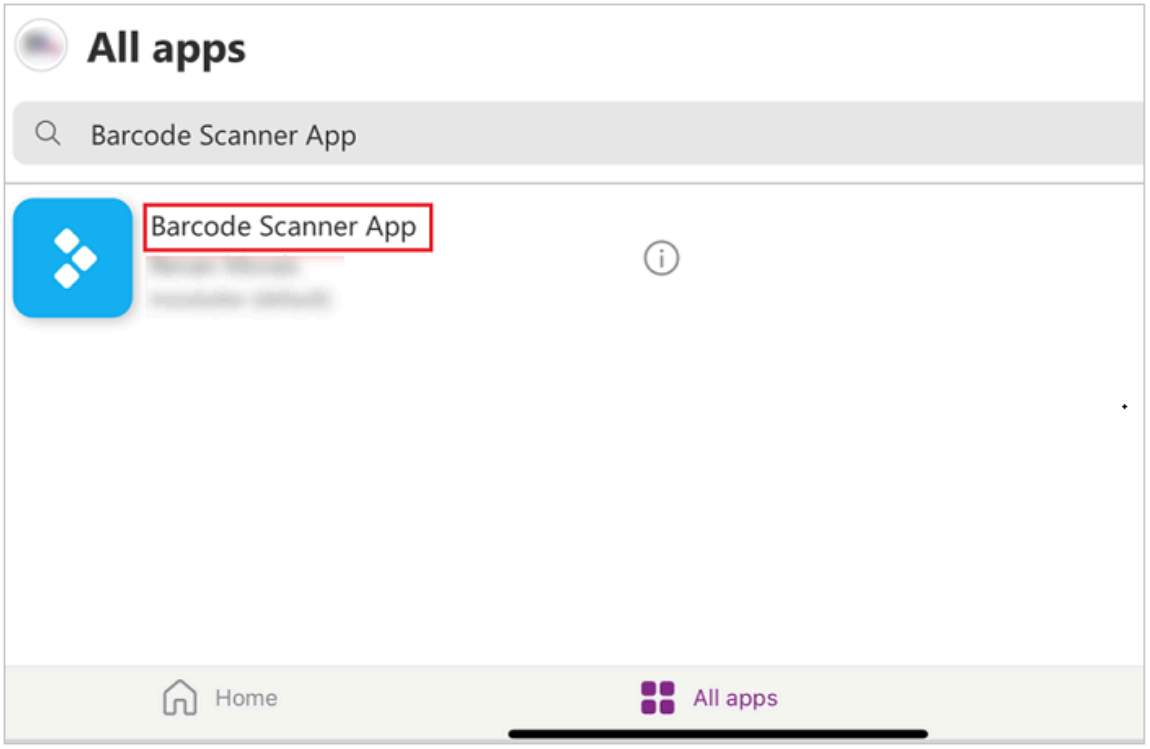
Power Fx

```
ThisItem.ScannedTime
```



Try the app

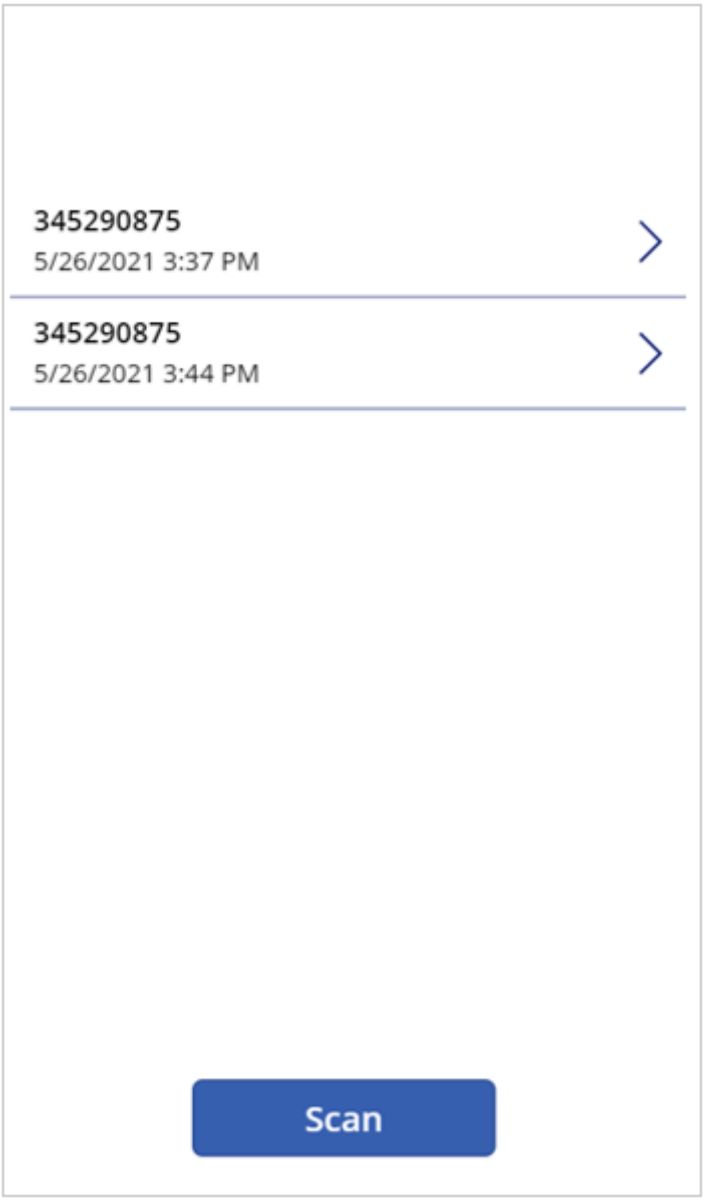
1. To test the app, select **Screen1**, and then press **F5** on the keyboard.
2. After testing, save and publish the app.
3. Download Power Apps Mobile on your phone.
4. Open the **Power Apps** app, and sign in.
5. Select and open the barcode reader app.



6. Select **Scan**, and scan any barcode label (for example, barcode label on a book).

Final Results

You'll see the scanned barcode information is saved inside the app.



See also

Barcode scanner control in Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Create an app that uses mobile sensors

Article • 12/16/2022

Sensors enable you to add richer functionality into your apps by leveraging the capabilities of your phone. In this article, you'll learn about using sensors on your mobile device within Power Apps.

To do this, you'll be making a sample app with two screens. The first screen will display the current compass heading, latitude, longitude, altitude, and can scan NFC tags to obtain text information. The second screen will showcase how to display accelerometer data to produce the angle of the device along the X and Y axes.

To set up the scenario behind the first screen, imagine that you are building an app for an NFC-based scavenger hunt. Each participant will use the app to scan an NFC tag which will provide them with a compass heading, latitude, and longitude that will point them to their next destination. At that destination, the participant will then scan another tag and repeat the process until they reach the end of the hunt.

For the second screen, imagine needing a tool to roughly measure angles. When on the second screen, the participant will be able to place or hold their device on a surface, and get the angles on the X and Y axis of the device as well as view a visual representation of those angles.

Watch this video to learn how to build an app that uses mobile sensors:
https://learn-video.azurefd.net/vod/player?id=01899fa6-7a02-4ed6-9ab1-4bc8b5bf17b9&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fmobile-sensors

Prerequisites

- Any level of Power Apps license can be used for this app, as no data connections are being used.
- A mobile device such as a mobile phone or tablet with GPS and accelerometer capabilities will be needed as most PCs might not have the necessary sensors.
- An NFC capable device is needed for the NFC scanning feature. It's also assumed that the NFC tags to be scanned have been pre-configured to produce text values in the following example format:

  ```
  ```

```
"<b>Heading: </b> 80 degrees <br>
<b>Latitude: </b> 44.4604788 <br>
<b>Longitude: </b> -110.82813759"
```

This will provide the text in an HTML-friendly format for the app. The setup of the NFC tags is beyond the scope of this article, and the NFC elements can be ignored if needed; the primary focus is using the mobile sensors in Power Apps.

Add header and HTML text for the device sensor outputs

Header label

With an app open for editing in Power Apps, add a **Text label** to the screen by dragging and dropping it from the **Insert** pane. Position it at the top-left corner of the screen, and modify the following properties in the Properties pane:

Expand table

Property	Value
Text	"Scavenger Hunt"
Font size	24
Font weight	FontWeight.Semibold
Text alignment	Align.Center
Width	Parent.Width

Then in the **Advanced** pane, modify the following properties:

Expand table

Property	Value
Color	RGBA(255, 255, 255, 1)
Fill	RGBA(56, 96, 178, 1)

This will provide a header for the screen.

HTML text for the device sensor outputs

Next, add an **HTML text** control. This will be used to display all the device sensor outputs in one place. Use this code in the **HtmlText** property:

```
Power Fx

"<b><u>Current Location</u></b><br>
<br>
<b>Compass Heading: </b>" & Round(Compass.Heading, 2) & Char(0176) &
"<br>
<br>
<b>Lat: </b>" & Location.Latitude & "<br>
<b>Long: </b>" & Location.Longitude & "<br>
<b>Alt: </b>" & Round(Location.Altitude, 2) & " m"
```

Change the following properties in the Advanced pane:

Expand table

Property	Value
Size	21
BorderStyle	BorderStyle.Solid
BorderThickness	2
Width	560
Height	576
PaddingTop	16
PaddingRight	16
PaddingBottom	16
PaddingLeft	16

If you examine what was put in the HtmlText, you may notice that there are two sensor types that are being used.

First is the **Compass** sensor, which is using the **Heading** property to provide the compass heading from the device.

Second is the **Location** sensor, which provides the **Latitude**, **Longitude**, and **Altitude**. There are two considerations about Location. First, if you use just Location without the dot operator and property, you'll get a record that has fields for Latitude, Longitude, and

Altitude. Second, the Altitude property is in meters by default. If you would like to convert it to feet, replace the following formula:

Use meters:

```
Power Fx

Round(Location.Altitude, 2) & " m"
```

Use feet:

```
Power Fx

Round(Location.Altitude \* 3.2808, 2) & " ft"
```

> **Note**
>
> Many sensor properties might not work correctly if you preview the app inside Power Apps Studio. To work with sensor properties for testing, prefer to use a mobile phone instead.

Now position the **HTML text** control near the lower half of the screen.

Add Button for the NFC scan and HTML text control

NFC scan button

Add a button to the app, positioning in below the **HTML text** control that you added in the last step, and change the following properties in the Advanced pane:

Expand table

Property	Value
OnSelect	ClearCollect(colNFCScan, ReadNFC())
Text	"Scan NFC Tag"

The code in the **OnSelect** property is utilizing the ReadNFC() function, which activates the NFC reader. It then stores what is read from the NFC tag in a collection called

colNFCScan. This collection will have four fields: **RTD**, **Text**, **TNF**, and **URI**. While a full explanation of these fields is beyond the scope of this article, some explanation may be helpful. **TNF** stands for **Type Name Format** and is used to define the structure of the **Record Type Definition**, or **RTD**, which in turn defines the type of record contained in the **Text** and/or **URI** fields. **URI** stands for **Uniform Resource Identifier**, which is essentially the address of a resource. For the NFC tags being used in this example, in addition to the **Text** field containing the example text from the introduction of this topic, they would contain a **TNF** value of *1*, an **RTD** value of *T*, and a blank **URI** value.

Add HTML text control to display the NFC scan information

Add a second **HTML text** control and use the following formula in the **HTMLText** property:

```
Power Fx

"<b><u>Next Destination</u></b><br>
<br>" &
First(colNFCScan).Text
```

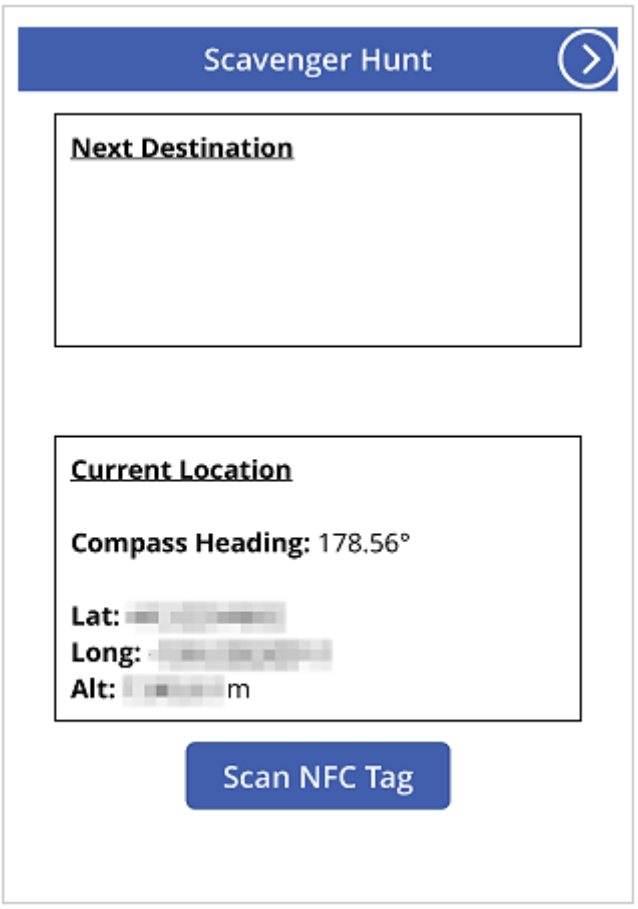
Change the following properties in the Advanced pane:

Expand table

Property	Value
Size	21
BorderStyle	BorderStyle.Solid
BorderThickness	2
Width	560
Height	248
PaddingTop	16
PaddingRight	16
PaddingBottom	16
PaddingLeft	16

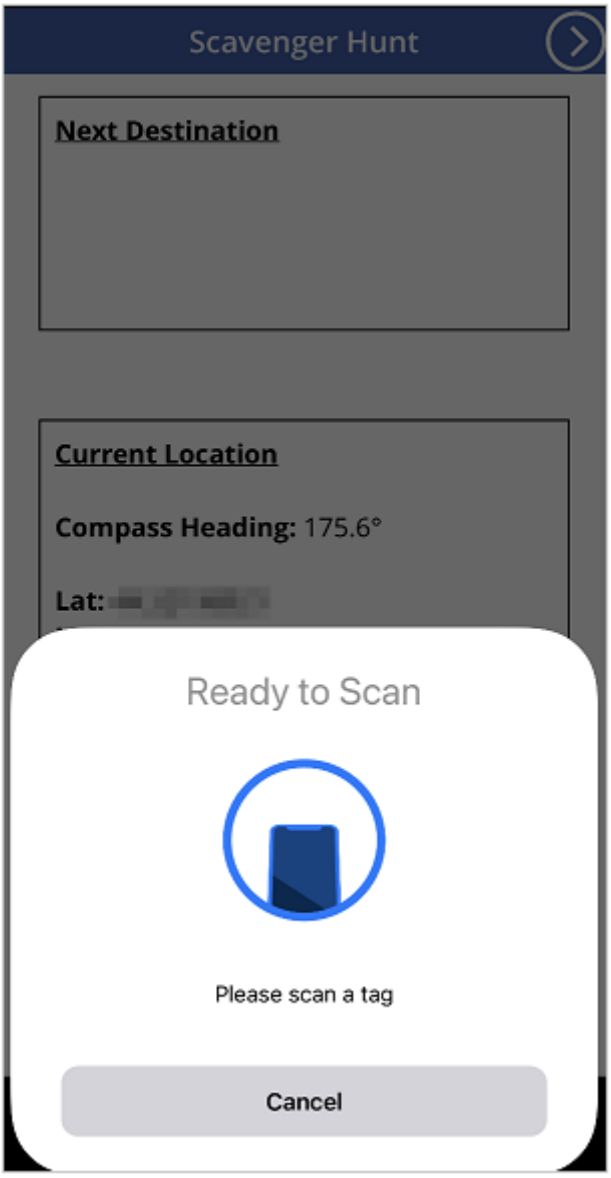
Without any NFC data present, the control will display **Next Destination**. When an NFC tag is scanned, it'll display **Next Destination**, and then the data from the **Text** field in the data collected from the scan.

Assuming the data for the NFC tags has been configured as described in the introduction, the user will see the next compass heading, latitude, and longitude values for the next scavenger hunt location after scanning one of those tags.

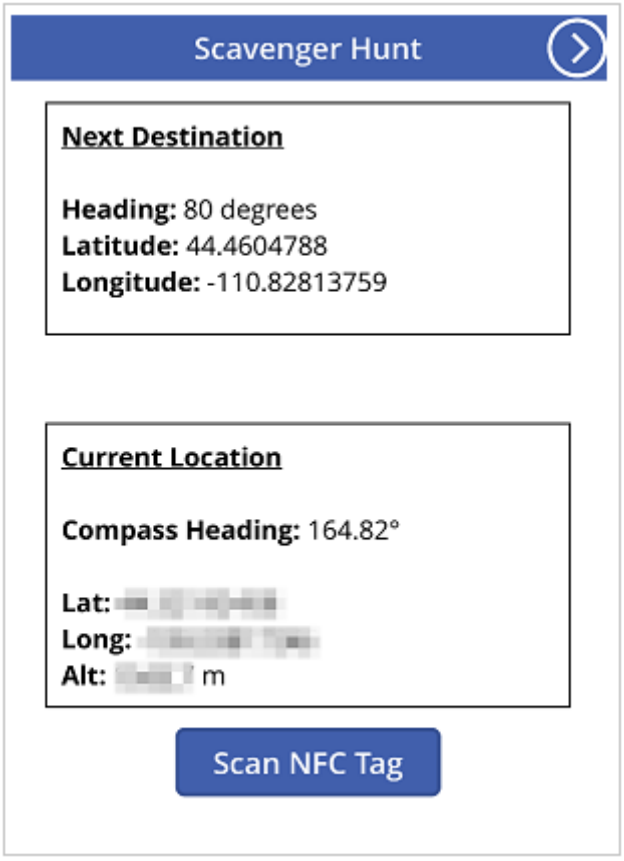


Test the app

Save and publish the app. On a mobile device with the necessary sensors, open the app and observe the compass heading, latitude, longitude, and altitude values. Try walking around while turning in various directions to see changes in the various sensor readings.



Press the **Scan NFC Tag** button to observe the NFC reader control. If you have an NFC tag that can produce a **Text** value, scan the tag to observe the text in the app. If not, select **Cancel**.



Add and configure second screen

Add a new **Blank screen** for the angle measurement portion of the app.

After adding the screen, go back to the first screen and add a **Next Arrow** icon from **+ Insert** > **Icons** > choose the **Next Arrow** icon. Position it in the top-right corner of the screen, and change the following properties in the Advanced pane:

Expand table

Property	Value
OnSelect	Navigate(Screen2)
Color	RGBA(255, 255, 255, 1)

If you have renamed the second screen that was just added, substitute that name for **Screen2** in the **Navigate** function.

Preview the app, selecting the icon that was just added to verify that it navigates you to the blank screen that was just added.

Add configuration for X and Y angles

Add a **Text label** to the screen by dragging and dropping it from the **Insert** pane. Position it at the top-left corner of the screen and modify the following properties in the **Properties** pane:

Expand table

Property	Value
Text	"2 Axis Level"
Font size	24
Font weight	FontWeight.Semibold
Text alignment	Align.Center
Width	Parent.Width

Then in the Advanced pane, modify the following properties:

Expand table

Property	Value
Color	RGBA(255, 255, 255, 1)
Fill	RGBA(56, 96, 178, 1)

This will provide a header for the second screen.

Next, add a **Back Arrow** icon from **+ Insert** > **Icons** > choose the **Back Arrow** icon. Position it in the top-left corner of the screen and change the following properties in the **Advanced** pane:

Expand table

Property	Value
OnSelect	Navigate(Screen1)
Color	RGBA(255, 255, 255, 1)

If you've renamed the first screen that was just added, substitute that name for **Screen1** in the **Navigate** function.

Finally, add an **HTML text** control. This will be used to display all the device sensor outputs in one place. Use the following formula in the HTML Text property:

Power Fx

```
"<b><u>Angles:</u></b><br>
<br>
<table width='100%'>
<tr>
<td width='50%'><b>X: </b>" & Abs(Round(Acceleration.X * (90 / 9.81),
0)) & Char(0176) & "</td>
<td width='50%'><b>Y: </b>" & Abs(Round(Acceleration.Y * (90 / 9.81),
0)) & Char(0176) & "</td>
</tr>
</table>"
```

Change the following properties in the Advanced tab:

Expand table

Property	Value
Size	21
BorderStyle	BorderStyle.Solid
BorderThickness	2
Width	560
Height	168
PaddingTop	16
PaddingRight	16
PaddingBottom	16
PaddingLeft	16

Position this control near the top of the app screen.

If you examine what was put in the **HtmlText**, you may notice that it is using the **Acceleration** sensor, with **X** and **Y** properties. There is also a **Z** property that we are not using in this scenario.

This sensor detects gravitational forces on the device over three planes. Using a mobile phone as an example, imagine a line extending out of the sides of the phone. This would be the **X** plane. Lifting the right side of the phone will produce a positive value and lifting the left side will produce a negative value. A line extending out of the top and bottom of the phone would represent the **Y** plane. Lifting the top of the phone will produce a positive value and lifting the bottom will produce a negative value. Finally, a

line coming out of the screen and back of the phone would represent the **Z** plane. Causing the screen to face upwards will produce a positive value and facing it downward will result in a negative value.

The accelerometer will measure forces acting on the device both in motion, such as if you dropped the device, and at rest, such as tilting the device on one or more of the axes described earlier. At rest, the theoretical values should be between 0 and +/- 9.81 m/s2, with 0 indicating that the relative axis is parallel to the Earth with no gravity acting on the sensor and a reading of 9.81 indicating that the relative axis is perpendicular to the Earth and the full force of gravity is acting on the sensor.

Looking at the code in the **HTML text** control you just added, notice the following section:

```
Power Fx

Abs(Round(Acceleration.X * (90 / 9.81), 0))
```

In this formula, from inside-out, first, the **Acceleration** on the **X** plane is being multiplied by **(90 / 9.81)**. This is using the theoretical maximum at-rest value to convert the sensor reading into degrees. The value of *90* is present because at the maximum theoretical value, the device would be perpendicular to the Earth on that plane, giving a measure of 90 degrees.

That value in degrees is then rounded to zero decimal places to provide a whole number degree value. Last, the absolute value of that rounded number is calculated to give a positive reading. This is so it will not matter if the angle is measured from one side or another.

> ⓘ **Note**
>
> The values shared above are approximates, and don't represent exact measurement.

Add the level visuals

In this step, we're going to use some controls in unconventional ways to achieve a visual goal.

Circular bubble level-like visual

To begin, add a **Button** control to the screen, renaming it to **btnCircleLevel** and change the following properties in the **Advanced** pane:

Expand table

Property	Value
Text	""
BorderColor	RGBA(56, 96, 178, 1)
FocusedBorderThickness	2
Fill	Transparent
DisabledFill	Self.Fill
X	(Parent.Width - Self.Width) / 2
Y	(Parent.Height - Self.Height) / 2
Width	400
Height	Self.Width

The result of these changes should produce a circular button in the center of the screen that cannot be pressed due to being disabled.

Next, add a **Circle** shape, set it's Border Radius to 400 and change the following properties in the **Advanced** pane:

Expand table

Property	Value
FocusedBorderThickness	0
X	(Parent.Width - Self.Width) / 2 + (Round(Acceleration.X / 9.81 * 90, 0) / 90 * btnCircleLevel.Width / 2)
Y	(Parent.Height - Self.Height) / 2 - (Round(Acceleration.Y / 9.81 * 90, 0) / 90 * btnCircleLevel.Width / 2)
Width	42
Height	Self.Width
Width	400
Height	Self.Width

The **X** and **Y** properties allow the **Circle** shape to move about the center of the app screen, relative change in the **Acceleration** sensor values but contained within the circular area of **btnCircleLevel**.

X and Y Axis bubble level-like visuals

Add a **Slider** control to the app, rename it to **sldXValue**, and change the following properties in the **Advanced** pane:

Expand table

Property	Value
Min	-90
Max	90
Default	Round(Acceleration.X * (90 / 9.81), 0)
ValueFill	Self.RailFill
X	(Parent.Width - Self.Width) / 2
Y	btnCircleLevel.Y + btnCircleLevel.Height + 30
Width	btnCircleLevel.Width
Height	70

This **Slider** control will display the angle similar to how a bubble level does: the **Handle** will move towards the raised side of the device, similar to the air bubble in a bubble level.

Next, copy **btnCircleLevel** by selecting it, pressing **Ctrl + C** and then **Ctrl + V**. Rename the control to **btnXValueOverlay** and change the following properties in the **Advanced** pane:

Expand table

Property	Value
X	sldXValue.X - sldXValue.HandleSize / 2
Y	sldXValue.Y
Width	sldXValue.Width + sldXValue.HandleSize

Property	Value
Height	sldXValue.Height

These changes will position it over the **sldXValue** control, preventing the user from changing its value and providing a visual border.

Copy and paste **sldXValue**, using the same method used to copy **btnCircleLevel**. Rename it to **sldYValue** and change the following properties:

Expand table

Property	Value
Layout	Layout.Vertical
X	btnCircleLevel.X - Self.Width - 30
Y	(Parent.Height - Self.Height) / 2
Width	sldXValue.Height
Height	sldXValue.Width

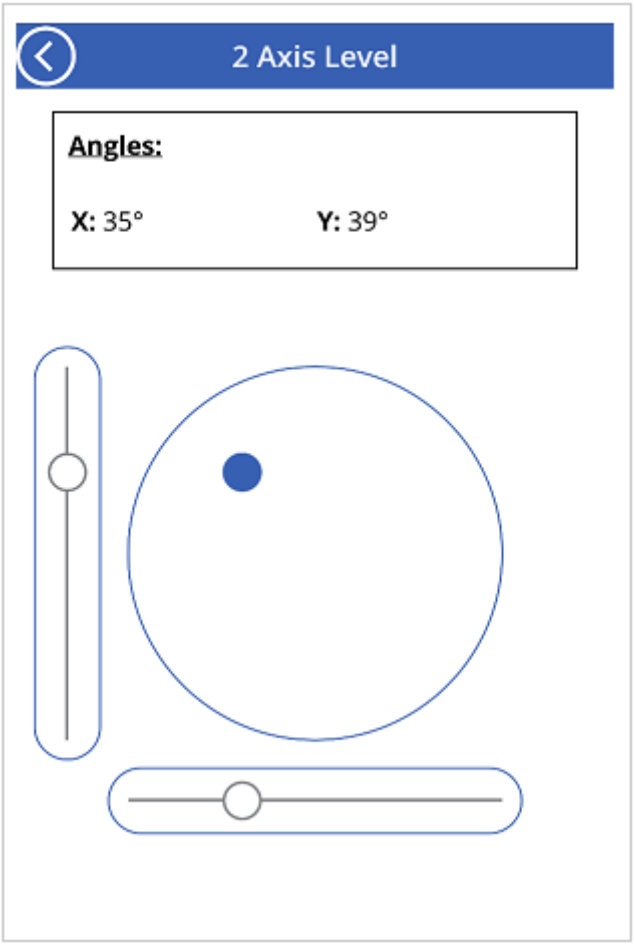
Copy and paste **btnXValueOverlay**, renaming it to **btnYValueOverlay**, changing the following properties:

Expand table

Property	Value
X	sldYValue.X
Y	sldYValue.Y - sldYValue.HandleSize / 2
Width	sldYValue.Width
Height	sldYValue.Height + sldYValue.HandleSize

This completes the bubble level-like visuals.

Test the level app



Save and publish the app. On a mobile device with the necessary sensors, open the app and navigate to screen with the bubble level visuals. Tilt the device from left to right, then up and down, and finally in all directions, noting the changes in angle properties as well as the changes in the visuals. Find an angled surface to place the device on and again note the angles and visuals.

See also

- Acceleration, App, Compass, Connection, and Location signals in Power Apps
- Mixed reality controls in Power Apps
- Geospatial controls in Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Create an app with 3D and mixed reality controls

Article • 12/16/2022

You can use the 3D object and View in MR controls to view items in 3D, or to see how a particular item might fit within a specified space.

In this article, you'll learn how to use the 3D and mixed reality controls in Power Apps.

Watch this video to learn how to build mobile apps with View in MR control: https://learn-video.azurefd.net/vod/player?id=7651a5f3-4886-4910-97b0-290116a9fa89&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fbuild-view-in-mr-3d-apps

Prerequisites

See prerequisites for mixed-reality capable devices.

Create an app with View in MR control

The **View in MR** control creates a button in your app. When app users selects the button, it overlays a selected 3D model (in the .glb file format) or image (in .jpg or .png file formats) onto the live camera feed of the device.

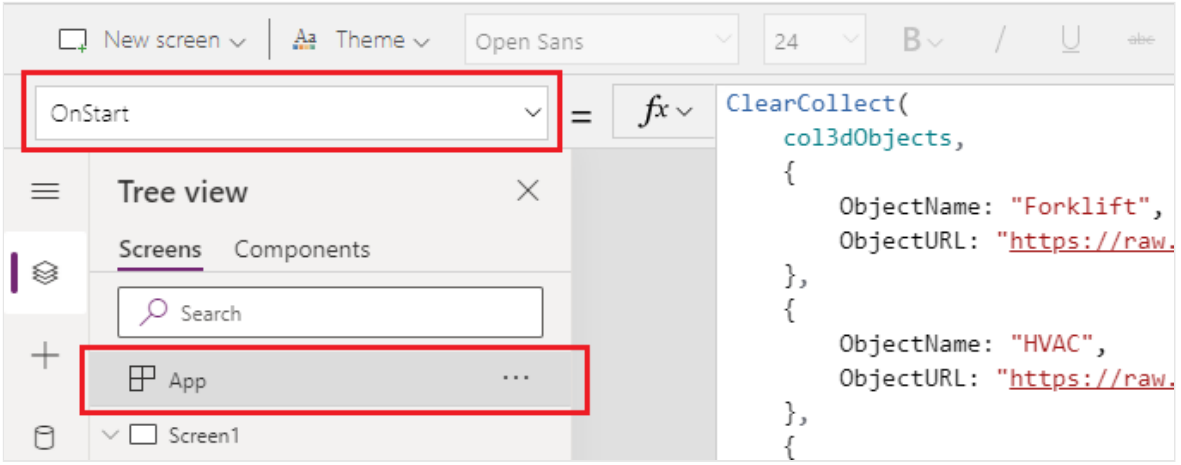
> **Tip**
>
> You can also take photos and **upload them to OneDrive**.

In this section, we will build a screen in an app that will allow a user to view the selected item in mixed reality using the **View in MR** control.

1. Open an app in Power Apps Studio.
2. Select the **OnStart** property of the app, copy and paste the following:

```
Power Fx

ClearCollect(
    col3dObjects,
    {
        ObjectName: "Forklift",
```

```
        ObjectURL:
"https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/forklift.glb"
    },
    {
        ObjectName: "HVAC",
        ObjectURL:
"https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/hvac.glb"
    },
    {
        ObjectName: "Machine",
        ObjectURL:
"https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/machine.glb"
    },
    {
        ObjectName: "Pallet w/ Boxes",
        ObjectURL:
"https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/pallet_with_boxes.glb"
    },
    {
        ObjectName: "Robot Arm",
        ObjectURL:
"https://raw.githubusercontent.com/microsoft/experimental-pcf-control-
assets/master/robot_arm.glb"
    }
)
```



This formula will create a collection containing the names and links of the 3D objects to be viewed in mixed reality.

3. Add a **Text label** to the screen by dragging and dropping it from the **Insert** tab.

4. Position the added text label at the top-left corner of the screen, and modify the following properties in the **Properties** tab:

Expand table

Property	Value
Text	"View In MR"
Font size	24
Font weight	`FontWeight.Semibold`
Text alignment	`Align.Center`
Width	640

5. In the Advanced tab, modify the following properties for the text label:

Expand table

Property	Value
Color	`RGBA(255, 255, 255, 1)`
Fill	`RGBA(56, 96, 178, 1)`

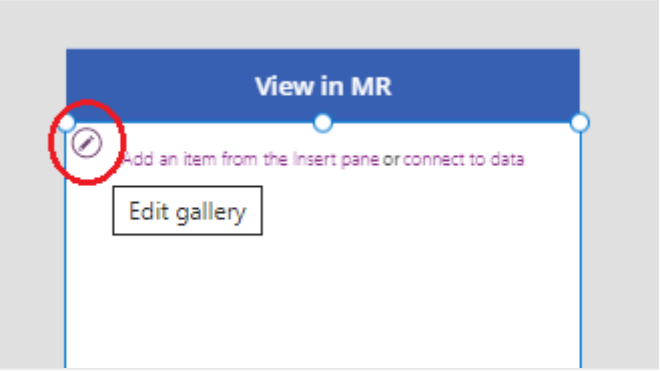
This change provides a header for the screen.

6. Insert a **Blank vertical gallery** from the **Layout** section of the **Insert** tab.
7. Position the gallery below the "View in MR" header you added earlier and expand it to use rest of the screen.
8. Change the following properties of the gallery:

Expand table

Property	Value
Data source	`col3dObjects`
Template size	80
X	0
Y	92
Width	640
Height	1044

9. Select **Edit gallery** to edit the gallery.



10. Insert a **Text label** into the gallery.

11. Change the following properties of the added text label:

Expand table

Property	Value
Text	`ThisItem.ObjectName`
X	10
Y	5
Width	540

12. Edit the gallery again.

13. From the **Mixed Reality** section of the **Insert** tab, drag and drop the **View in MR** control into the gallery.

14. Change the following properties of the **View in MR** control.

Expand table

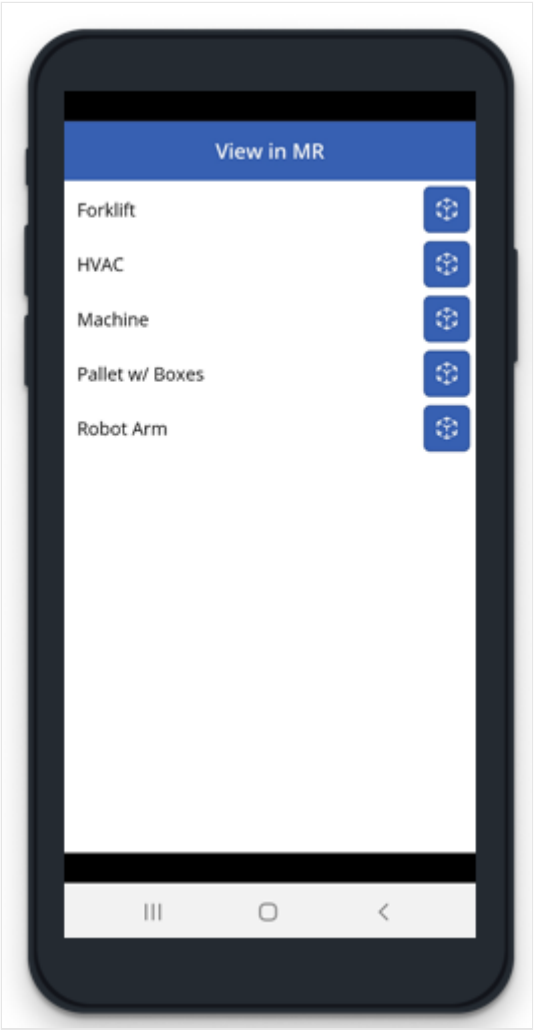
Property	Value
Text	"View In MR"
Display type	Icon
Source	`ThisItem.ObjectURL`
X	`Parent.TemplateWidth - Self.Width - 5`
Y	5

Property	Value
Width	70

This change will provide a gallery that has a list of 3D objects and a button to press for the user to view those objects in mixed reality.

Test the mixed reality control

Now that all the controls have been added, save and publish the app. On a mixed-reality capable device, open the app, and press the button that corresponds with the object you would like to view. This will open the **View in MR** experience.



Follow the on-screen instructions to calibrate the device by slowly moving it left and right while pointing the camera at a surface to be measured. Once the calibration is complete, you'll see an array of dots on the surface as well as a circle near the center of the screen. This circle shows where the 3D object can be placed initially. Once the cube has been placed it can be moved or rotated by using touch controls.

To move the object, press on it and slide it to where you would like it to be placed. To rotate the object, press one finger to the screen then swipe with another finger right or

left.

Use the back button in the top left corner to return to the app screen and select another object. Repeat this process as desired to view each of the objects.

Create an app with 3D object control

The **3D object** control allows you to view an item in 3D inside Power Apps. When app loads, it shows the 3D models (in the .glb file format) or image (in .jpg or .png file formats) inside the app through the control. You can select the 3D model to rotate, zoom in or out.

In this section, we will build a screen in an app that will allow a user to view the selected items in 3D using the **3D object** control.

1. Follow the steps 1 and 2 from the Create an app with View in MR control section to create an app with the collection of 3D objects for the app.

2. Add a **Text label** to the screen by dragging and dropping it from the **Insert** tab.

3. Position the added text label at the top-left corner of the screen, and modify the following properties in the **Properties** tab:

Expand table

Property	Value
Text	"3D object"
Font size	24
Font weight	`FontWeight.Semibold`
Text alignment	`Align.Center`
Width	640

4. In the Advanced tab, modify the following properties for the text label:

Expand table

Property	Value
Color	`RGBA(255, 255, 255, 1)`
Fill	`RGBA(56, 96, 178, 1)`

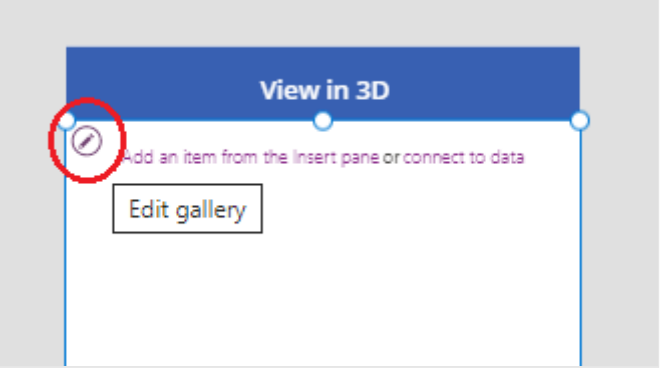
This change provides a header for the screen.

5. Insert a **Blank vertical gallery** from the **Layout** section of the **Insert** tab.
6. Position the gallery below the "View in MR" header you added earlier and expand it to use rest of the screen.
7. Change the following properties of the gallery:

Expand table

Property	Value
Data source	`col3dObjects`
Template size	80
X	0
Y	92
Width	640
Height	1044

8. Select **Edit gallery** to edit the gallery.



9. Insert a **Text label** into the gallery.
10. Change the following properties of the added text label:

Expand table

Property	Value
Text	`ThisItem.ObjectName`
Width	640

11. Edit the gallery again.
12. From the **Media** section of the **Insert** tab, drag and drop the **3D object** control into the gallery.
13. Change the following properties of the **View in MR** control.

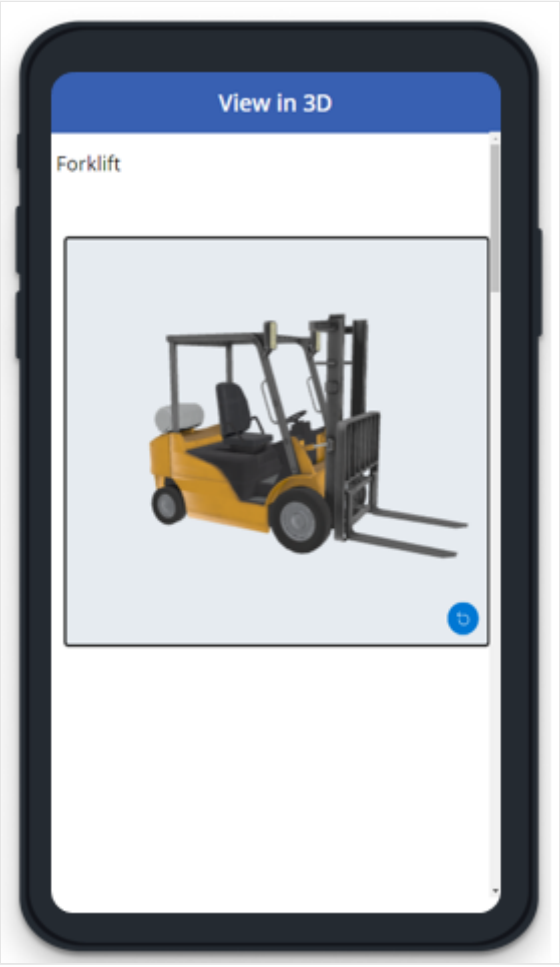
Expand table

Property	Value
Source	`ThisItem.ObjectURL`
X	18
Y	138
Width	600
Height	550

This change will provide a gallery that has a list of 3D objects that you can view in 3D.

Test the 3D control

Now that all the controls have been added, save and publish the app. On a mixed-reality capable device, open the app, touch the screen to zoom in, zoom out, or turn the object being viewed in 3D.



Since the gallery contains several objects in 3D, swipe down on the screen to see other objects and use the zoom or turn capabilities to work with the objects in 3D.

See also

- 3D object control
- Measuring Camera control
- View shape in MR control

Feedback

Was this page helpful? Yes No

Provide product feedback

Create an app to view a shape in mixed reality

Article • 12/16/2022

You can use the **View Shape in MR** control in your app to let users see if a simple cube might fit within a specified space. They might want to do this to see how a certain object that you provide would fit in their space. If you have a 3D model or a head-on picture for the object, you'd like to fit test consider using the View in MR instead.

The control creates a button in your app. When app users select the button, it overlays a cube onto the live camera feed of the device. You set up the dimensions of the cube when you edit the control in Power Apps.

Watch this video to learn how to build mobile apps with View Shape in MR control: https://learn-video.azurefd.net/vod/player?id=2b6080fd-8236-48a5-b9b2-48285eccdba6&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fmobile-apps-view-shape-in-mr

In this article, we'll create a screen in an app that will allow users to create a custom cube shape by inputting unit to be used (centimeters, feet, inches, or meters), and defining the height, width, and depth of the cube.

Prerequisites

See prerequisites for mixed-reality capable devices.

Add controls for cube's dimensions and units

Add header and labels to define cube's dimensions and units to the app screen.

Header label

With an app open for editing in Power Apps Studio, add a text label to the screen. Position it at the top-left corner of the screen and modify the following properties in the Properties tab:

Expand table

Property	Value
Text	`"Place A Cube"`
Font size	`24`
Font weight	`FontWeight.Semibold`
Text alignment	`Align.Center`
Width	`Parent.Width`

Then in the Advanced tab, modify the following properties:

Expand table

Property	Value
Color	`RGBA(255, 255, 255, 1)`
Fill	`RGBA(56, 96, 178, 1)`

This change will provide a header for the screen.

Labels and controls to specify cube properties

1. Add four text labels to the screen. These labels will be used to identify the controls we'll add later. For each label, modify the **Text** property to one of the following:

   - `Unit Type`
   - `Width`
   - `Height`
   - `Depth`

2. Set the **Width** property of these labels to 160 and position them so they're stacked vertically in the order described above, near the top of the screen and to the left side.

3. Add a dropdown control and three text entry controls. The dropdown will be used to select a unit type for the cube dimensions and the three text entry controls will be used to enter the width, height, and depth of the cube.

4. Select the drop-down, rename it to *drpUnitType*, and change the following properties:

Expand table

Property	Value
Items	`["Centimeters", "Feet", "Inches", "Meters"]`
Default	`"Feet"`
Width	`280`

5. Position the control to the right of the Unit Type label.

6. Add three text input controls, and rename them to *txtShapeHeight*, *txtShapeWidth*, and *txtShapeDepth*. Change the following properties for all three controls:

Expand table

Property	Value
Default	`2`
Format	`TextFormat.Number`
Width	`144`

7. Position the text input control named *txtShapeWidth* to the right of the Width label. Likewise, position the controls named *txtShapeHeight* and *txtShapeDepth* to the right of the Height and Depth labels, respectively.

Next, you'll add the View Shape in MR control and configure it to use the controls that were created to provide dynamic dimensions to the MR cube it'll generate.

Insert the View Shape in MR control

Insert the control into your app.

1. Open the **Insert** tab.

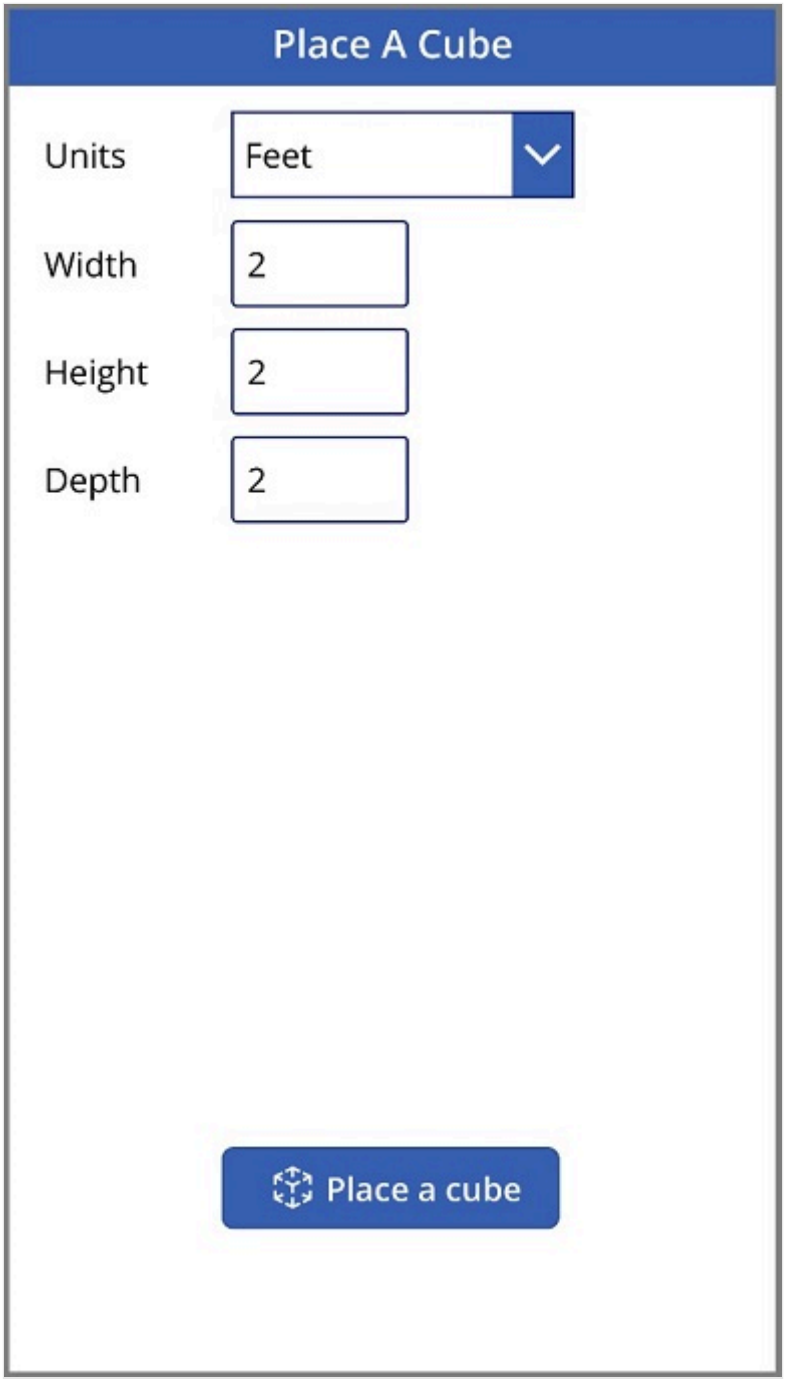
2. Expand **Mixed Reality**.

3. Select the control **View Shape in MR** to place it in the center of the app screen, or drag and drop it to position it anywhere on the screen.

4. Once the control has been added, position it near the bottom of the screen and change the following properties:

Expand table

Property	Value
Text	`"Place a cube"`
ShapeWidth	`txtShapeWidth.Text`
ShapeHeight	`txtShapeHeight.Text`
ShapeDepth	`txtShapeDepth.Text`
Units	`drpUnitType.Selected.Value`
Width	`280`



Test the app

Now that all the controls have been added, save and publish the app. On a mixed-reality capable device, open the app, and press the **Place a cube** button. This will open the **View Shape in MR** experience, allowing the user to define an area to be measured.

Follow the on-screen instructions to calibrate the device by slowly moving it left and right while pointing the camera at a surface to be measured. Once the calibration is complete, you'll see an array of dots on the surface and a circle near the center of the screen. This circle shows where the cube can be placed initially. Once the cube has been placed, it can be moved or rotated by using touch controls.

To move the cube, press on it and slide it to where you would like it to be placed. To rotate the cube, press one finger to the screen, and then swipe with another finger right or left.

Use the back button in the top-left corner to return to the app screen and change one or more of the width, height, depth, or unit type settings. Select the **Place a cube** button and place the new cube.

See also

- 3D object control
- Measuring Camera control
- View shape in MR control

Feedback

Was this page helpful? Yes No

Provide product feedback

Create an app with address input and map controls

Article • 12/16/2022

There are many scenarios where users are expected to enter addresses in an app. Historically, there have been multiple fields capturing each element of an address where the user provides the street address, city and the postal code, and selects state. Entering addresses can be frustrating and error-prone, particularly in mobile scenarios.

The address input control presents the user with dynamic address suggestions as they type. Using fuzzy matching logic, the control suggests multiple potential address matches from which the user can select—making it quicker and easier to enter accurate addresses.

The control returns the address as structured data, allowing the application to extract information like city, street, municipality, and even latitude and longitude. The data is in a format friendly to many locales and international address formats.

To use the control, you need to enable the Geospatial Services for the environment.

- Watch this video to learn how to use the address input control: https://learn-video.azurefd.net/vod/player?id=ff56e551-756f-4796-8af9-419b40b1dbfd&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fmobile-apps-address-map
- Watch this video to learn how to use the map control: https://learn-video.azurefd.net/vod/player?id=c62bc98b-cf1f-4289-af11-82618c5e6d02&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fmobile-apps-address-map

In this article, we'll learn how to use the address input control in canvas apps.

Prerequisites

To complete this lesson, you'll need a Power Apps environment with a Microsoft Dataverse database. If you don't have a license, you can sign up for the Power Apps Developer Plan or use a Power Apps trial.

Create a trial environment

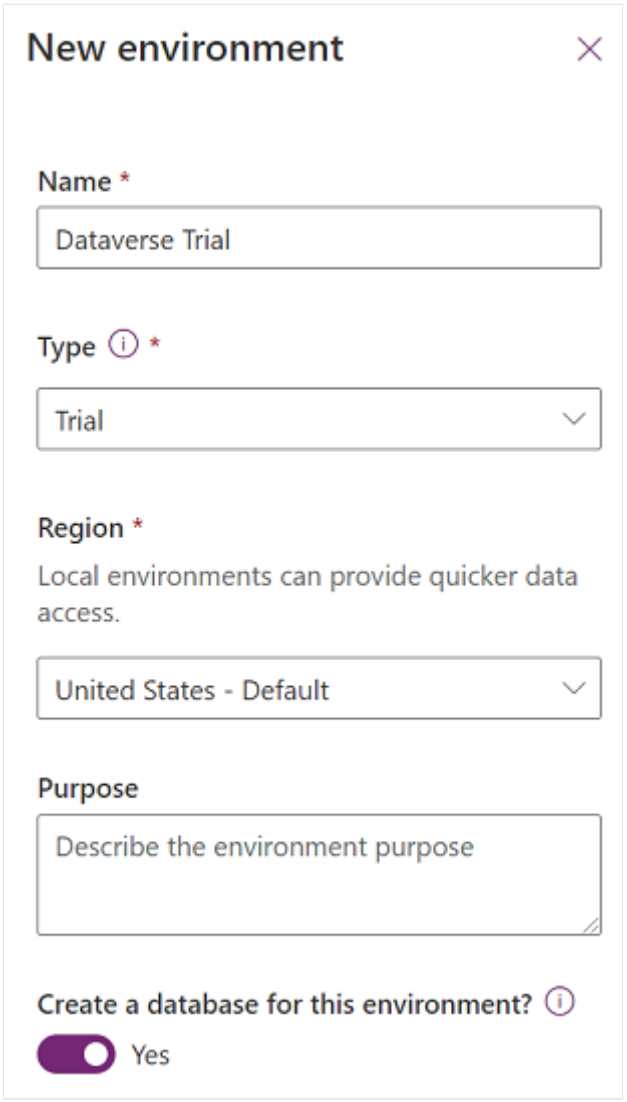
An environment in Power Apps is a space to store, manage, and share your organization's business data, apps, chatbots, and flows. It also serves as a container to separate apps that might have different roles, security requirements, or target audiences.

Each environment can have one Dataverse database.

To make sure that you have what you need to complete this lesson, we're going to create a new environment that includes a Dataverse database. This is a 30-day trial that will go away at the end of the 30 days, but you can upgrade it to a sandbox or production environment if you want to keep it.

We're going to create a trial environment that will include a Dataverse database, and the sample data used in this topic.

1. Go to Power Platform admin center in a web browser.
2. Select **Environments**.
3. Select **New**.
4. Enter a name, for example "Dataverse Trial".
5. Select **Trial** for the environment type.
6. Toggle **Create Database** to **Yes**.
7. Select **Next**.



8. Toggle **Deploy Sample Apps and Data** to **Yes**.

9. Select **Save**.

Your trial environment will now be created, including a Dataverse database and sample data in Dataverse tables.

Enable the Geospatial Services for the environment

Before we can use the Geospatial Services in canvas apps, an admin must enable access to the features for the environment where we want to create the app.

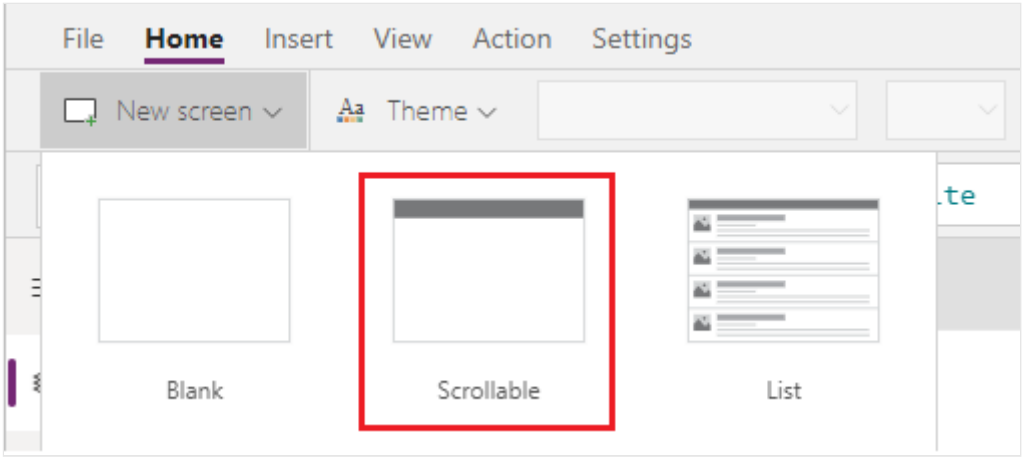
To learn about how to enable **Geospatial services** in your environment, see Enable the geospatial features for the environment.

Build a canvas app

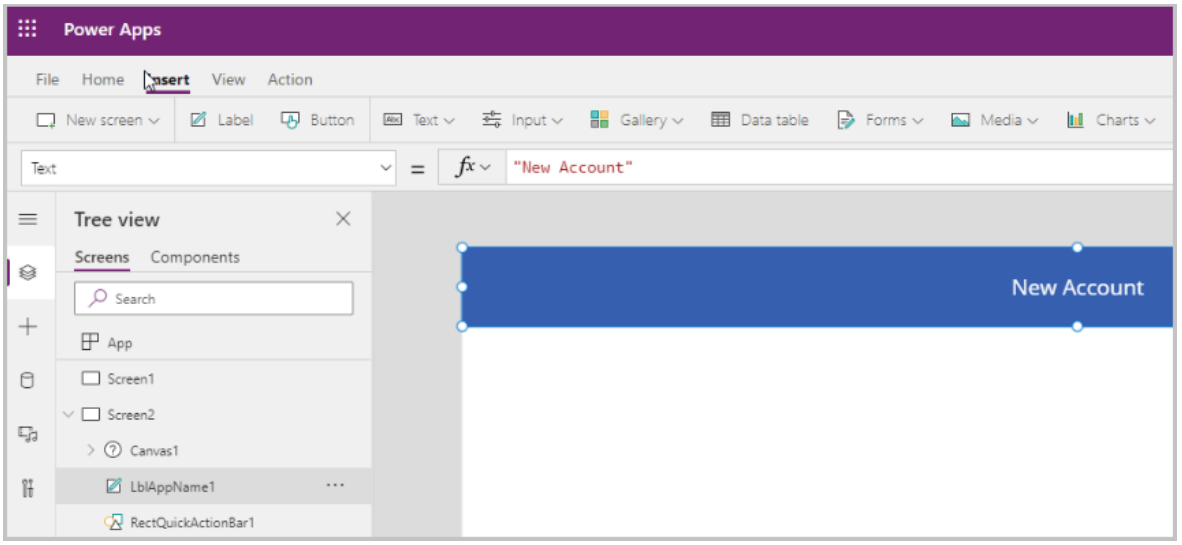
We'll start with creating an app and then, we'll create a header control to use within the app to standardize the look and feel of the header on different screens.

1. Create a blank canvas app in the environment you created earlier.

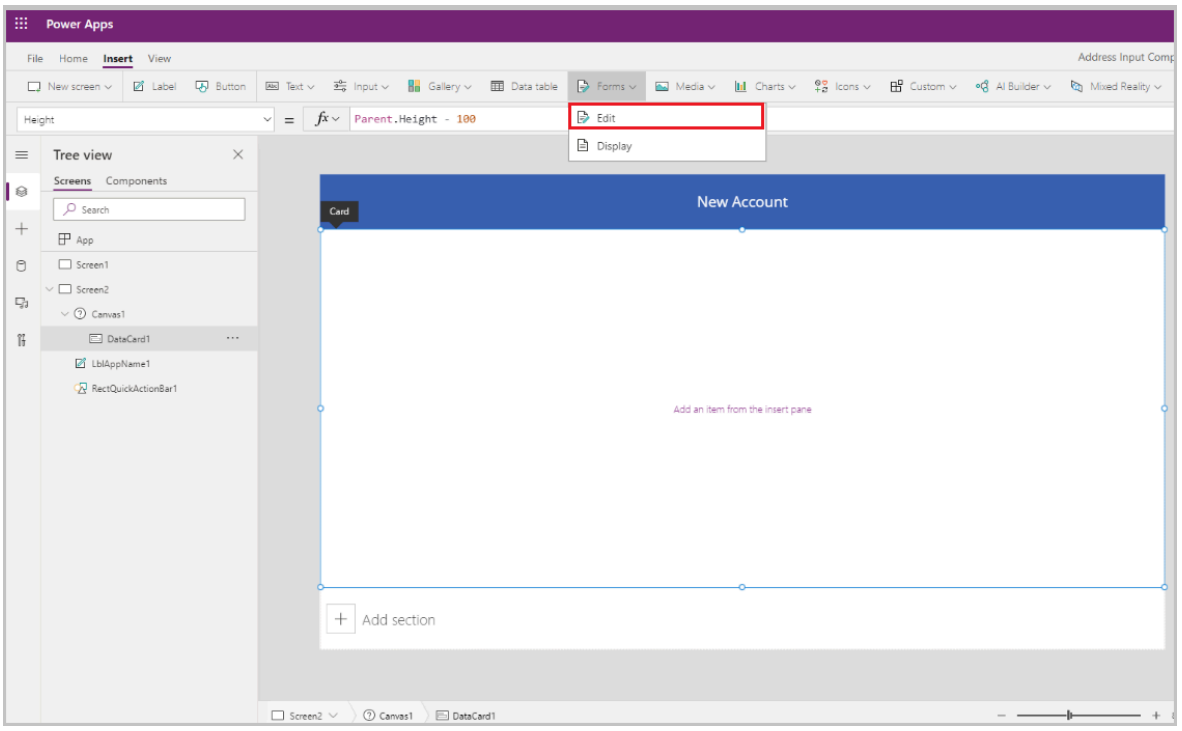
2. Select **Insert** > **New Screen** > **Scrollable** to add a new screen.



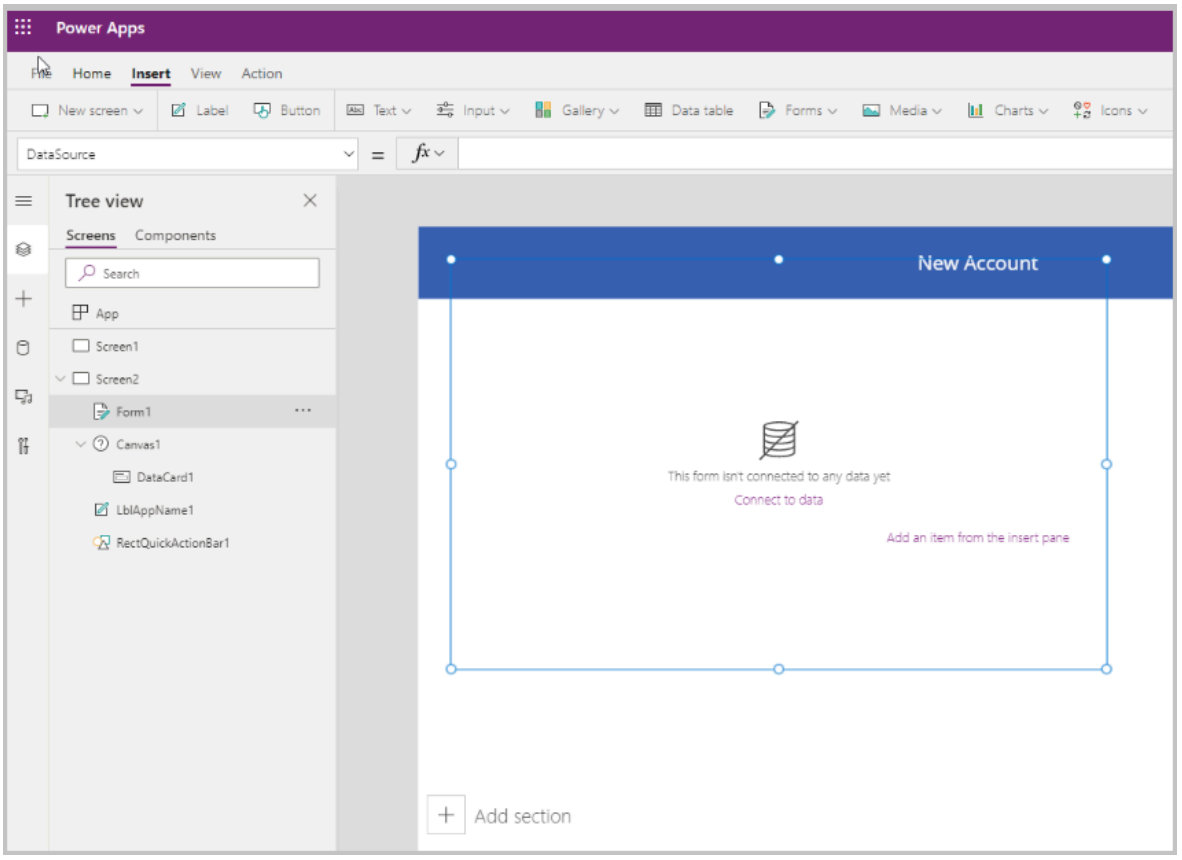
3. Change the label text to "New Account".



4. Select **Insert** > **Forms** > **Edit** to add a form to the screen.



The form control gets added



5. Update the following properties of the form:

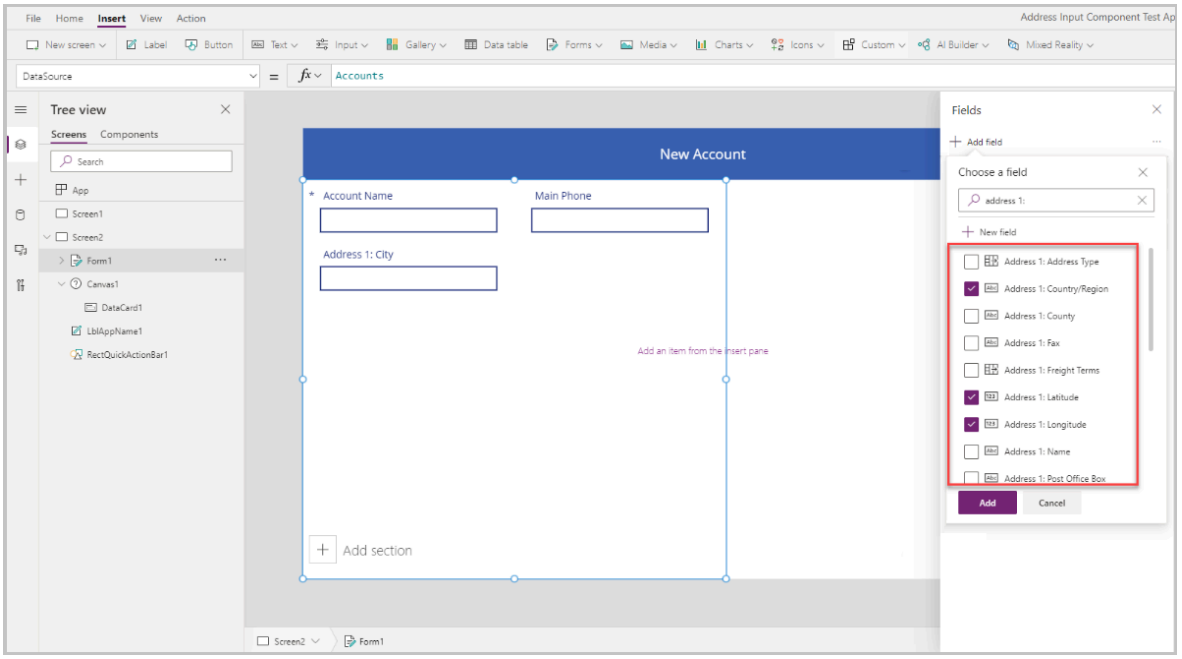
Expand table

Property	Value
Data source	Accounts

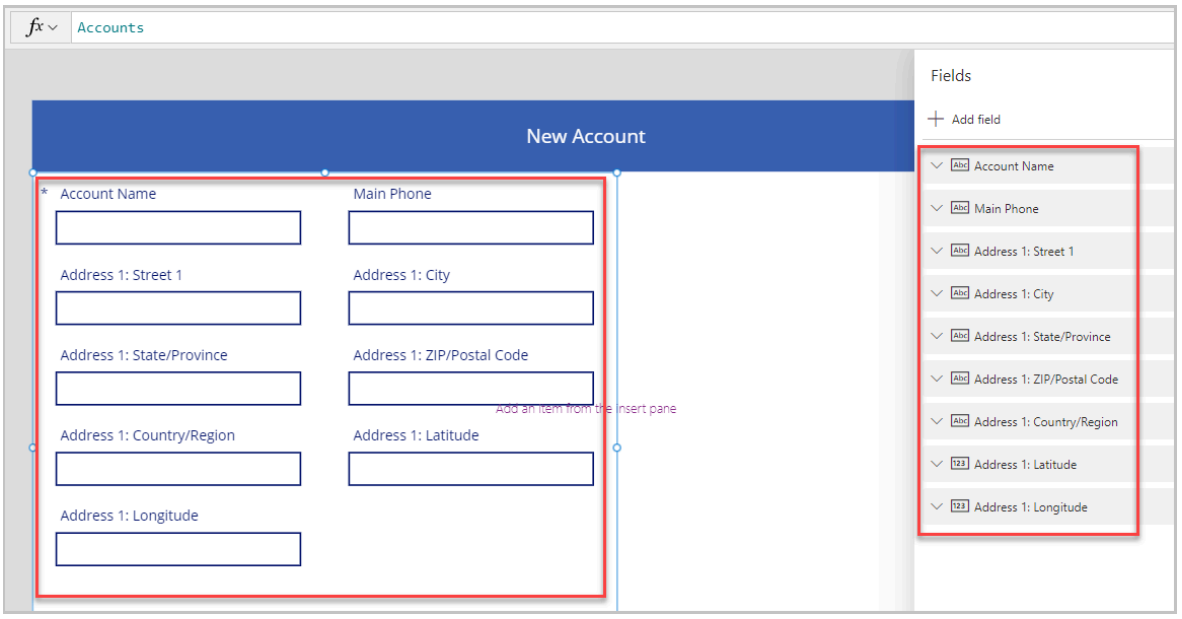
Property	Value
Columns	2
DefaultMode	`FormMode.New`

6. Select **Edit** next to fields to add more fields to the form, and select the following address fields:

    - Address 1: Street 1
    - Address 1: City
    - Address 1: State
    - Address 1: Zip/Postal Code
    - Address 1: Country/Region
    - Address 1: Latitude
    - Address 1: Longitude



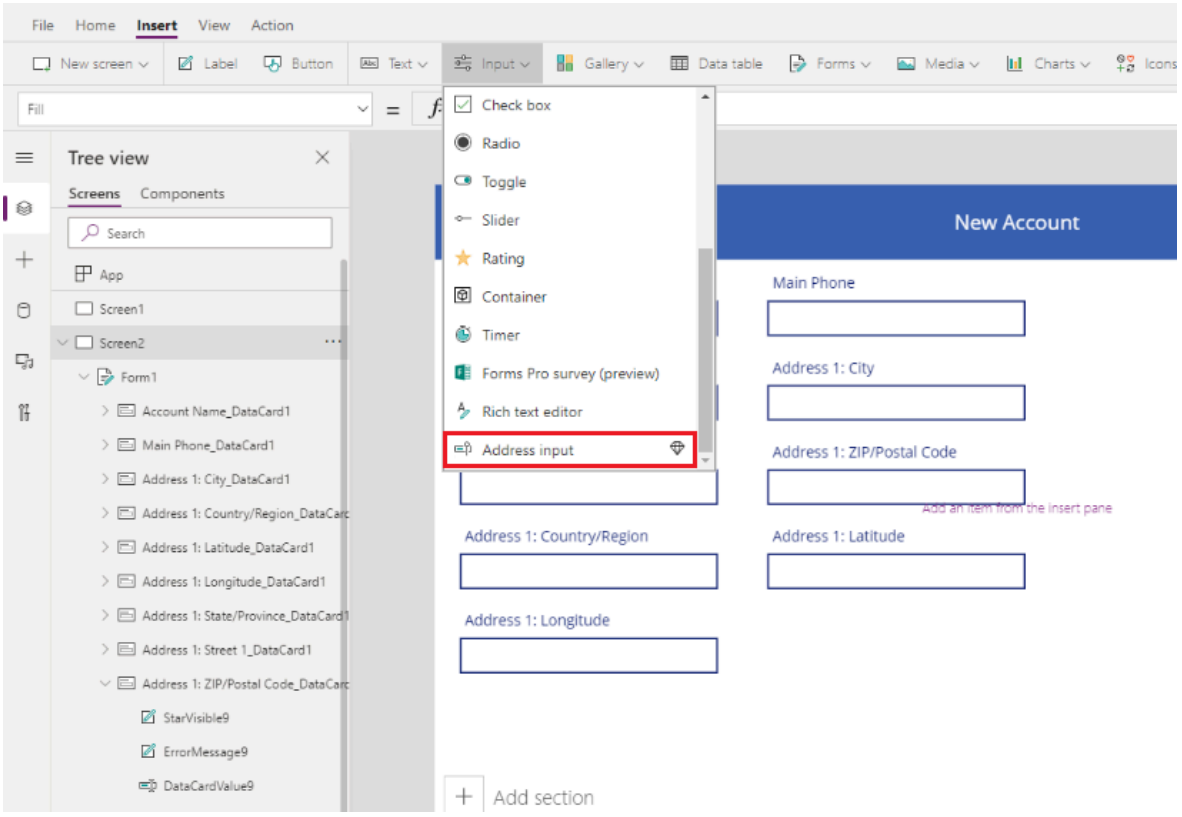
7. Adjust the order of the fields as shown below.



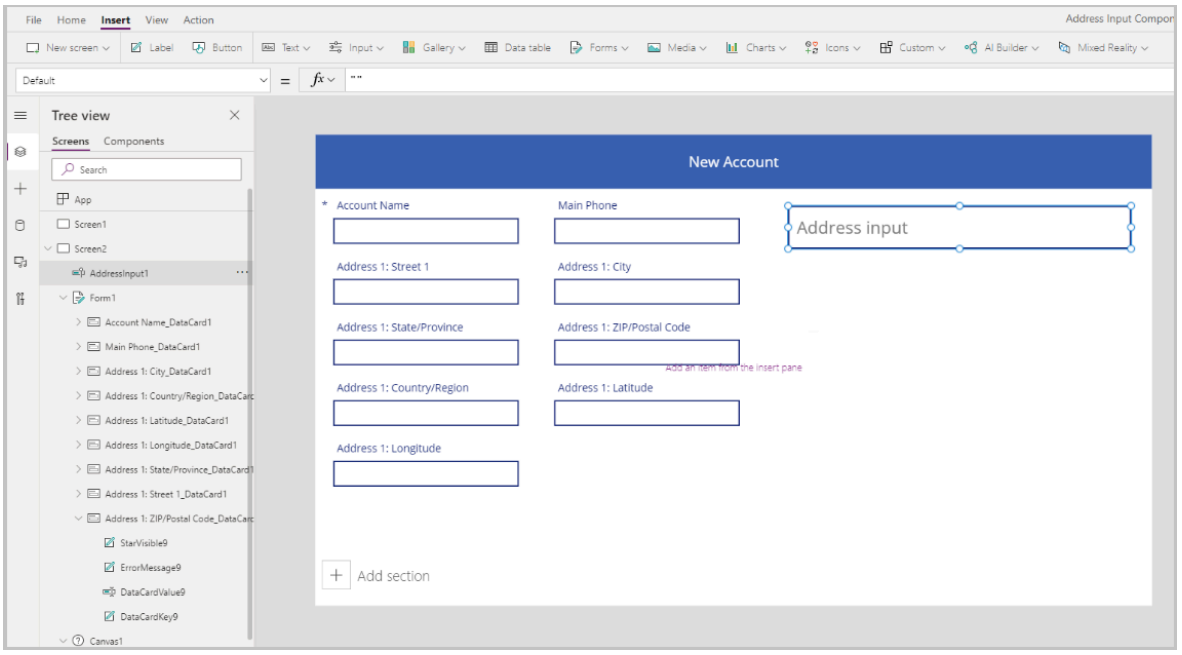
Address Input control

In this section, we'll add the **Address Input** control to the app.

1. Select **Insert** > **Input** > **Address Input** to add the control to the screen.

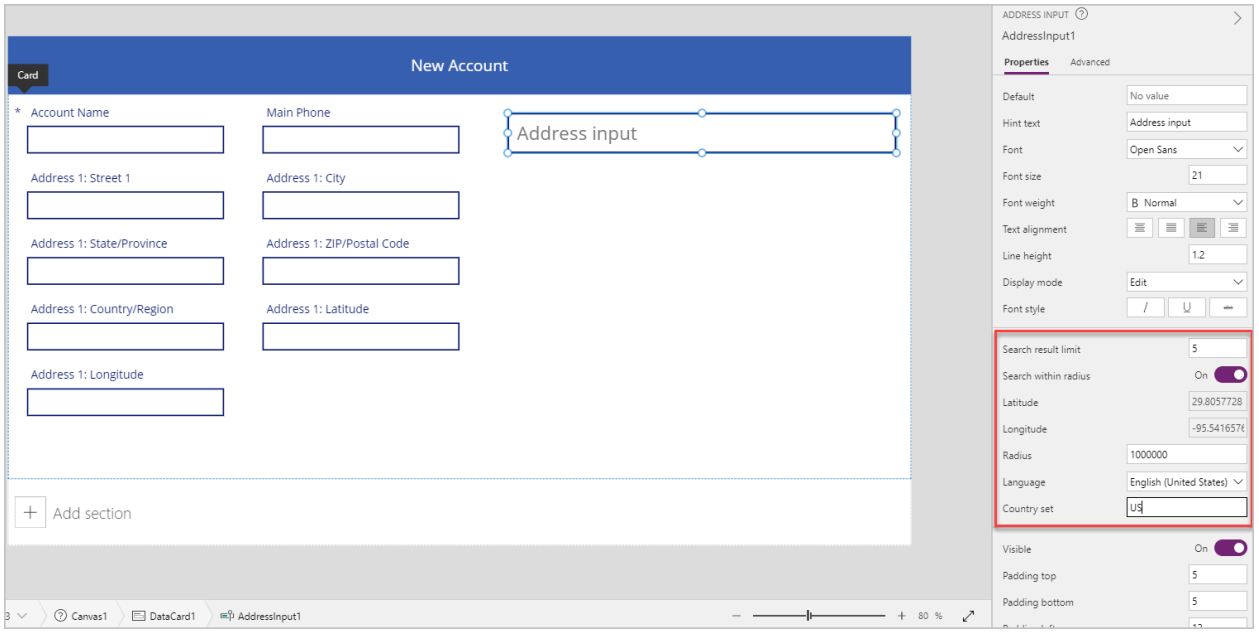


2. Move the control to the top-right side of the screen.



Configure properties for the Address Input control

In this section, we'll review and if necessary, set various properties of the address control. There are a few **Address Input** control-specific properties.



Expand table

Property	Description
Search result limit	The number of suggested addresses the control displays. We've set it to 5 in this example, so not more than five addresses will show up in search.
Search within radius	Whether the control should suggest addresses within the user-defined **Radius** of the **Latitude** and **Longitude**. We've set it to **Yes** in this example.

Property	Description
Latitude	The latitude of the center point used to geo-bias address suggestions. Requires **Search within radius** to be on. We've set it to the formula `Location.Latitude` in this example to return the latitude of the current location.
Longitude	The longitude of the center point used to geo-bias address suggestions. Requires **Search within radius** to be on. We've set it to the formula `Location.Longitude` in this example to return the latitude of the current location.
Radius	The radius, in meters, around **Latitude** and **Longitude** to constrain the address suggestions. Requires **Search within radius** to be **On**. We've set it to 100000 in this example.
Language	The language the address suggestions are returned in. We've left it with the default - "English (United States)".
Country set	Comma-separated list of countries/regions to constrain the address suggestions to, in ISO 3166 alpha-2 country codes. Examples: "US", "FR", "KW". We've it as US in this example.

Configure output properties for the Address Input control

The control outputs various properties when a user interacts with it inside an app. You can use these outputs in other controls or to customize the experience.

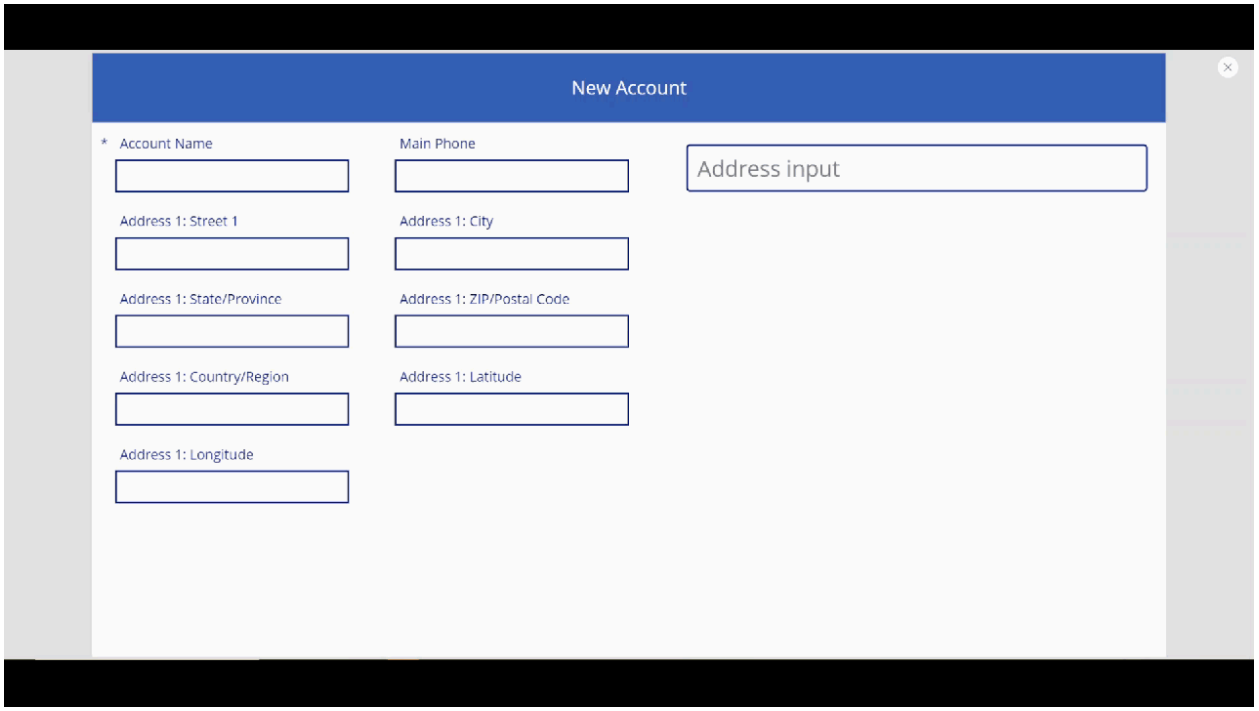
The following table lists the output properties available.

Expand table

Property	Description
UserInput	Text the user has typed in the input box.
SelectedLatitude	Latitude of the user-selected address in the input field.
SelectedLongitude	Longitude of the user-selected address in the input field.
SearchResultJson	The search result (based on the UserInput property), displayed as a string in JSON format.
FreeformAddress	Selected address from the list of suggested addresses.
LocalName	An address control that represents the name of a geographic area or locality that groups multiple addressable objects for addressing purposes, without being an administrative unit.
PostalCode	Postal code.

Property	Description
ExtendedPostalCode	Extended Postal Code.
CountryCode	Country code.
Country	Country.
CountryCodeISO3	Country code in ISO alpha-3 format.
CountrySubdivisionName	Country subdivision name.
StreetName	Street name.
StreetNumber	Street number.
Municipality	Municipality.
MunicipalitySubdivision	Municipality subdivision.
CountryTertiarySubdivision	Country tertiary subdivision.
CountrySecondarySubdivision	Country secondary subdivision.
CountrySubdivision	Country subdivision.

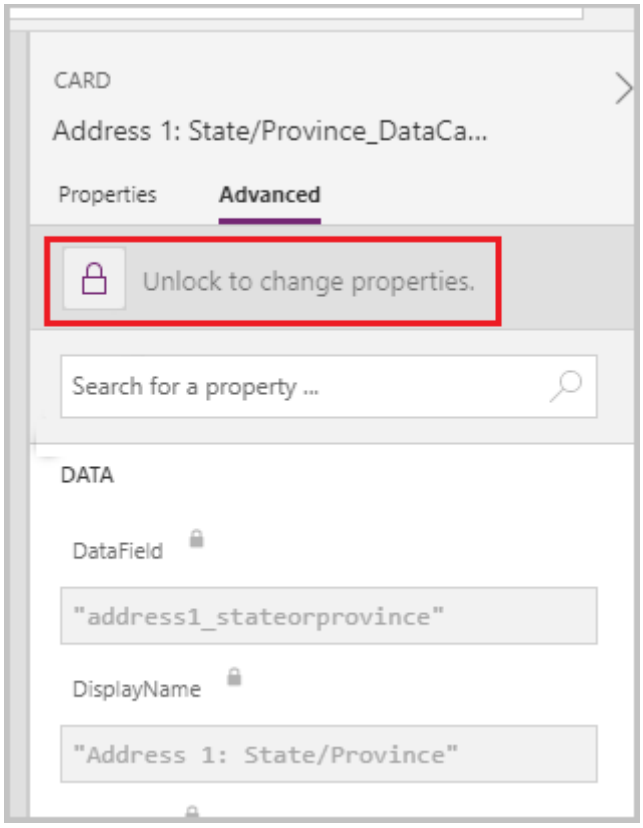
When you run the app and start typing an address, it will show addresses that match the entered value and are located only within 100,000 meters of the current location. Also, notice that it doesn't show more than five address suggestions at a time when you start typing in the address.



Using the above control, we'll capture the address in the address fields of the form to create a new account. This step is useful so that the user doesn't have to type in each of the address elements individually.

We'll set the data card values to the appropriate property value from the address input.

> **Tip**
>
> The default values of the data card might be locked. To unlock them, select the data card and go to the **Advanced properties**, and then select **Unlock to change properties**.

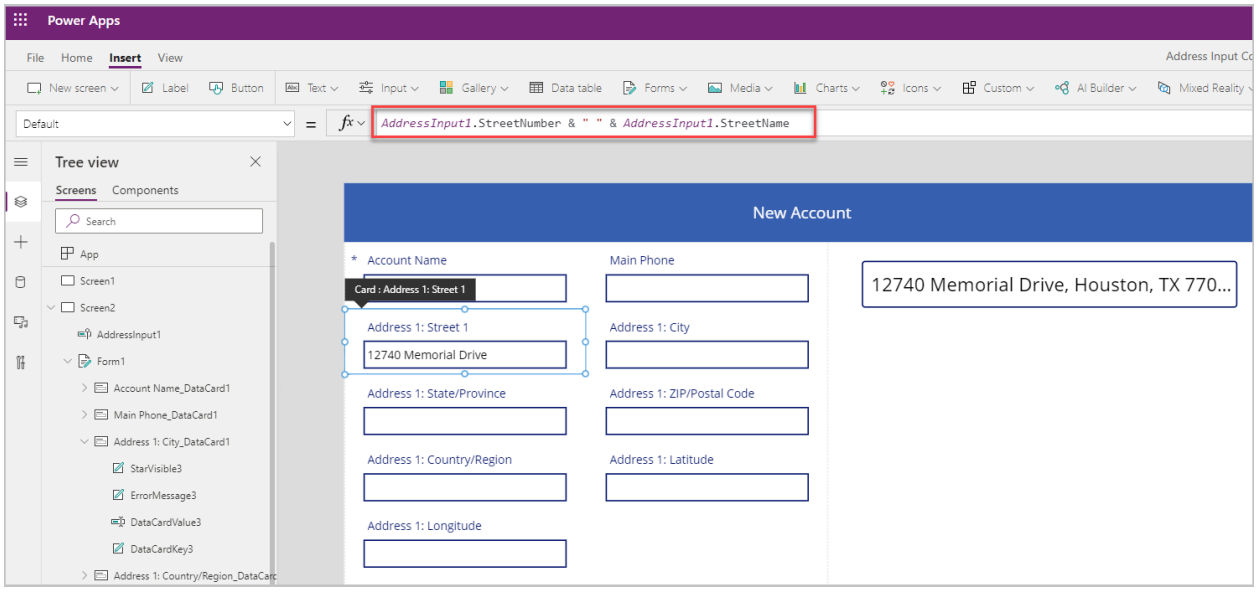


Expand table

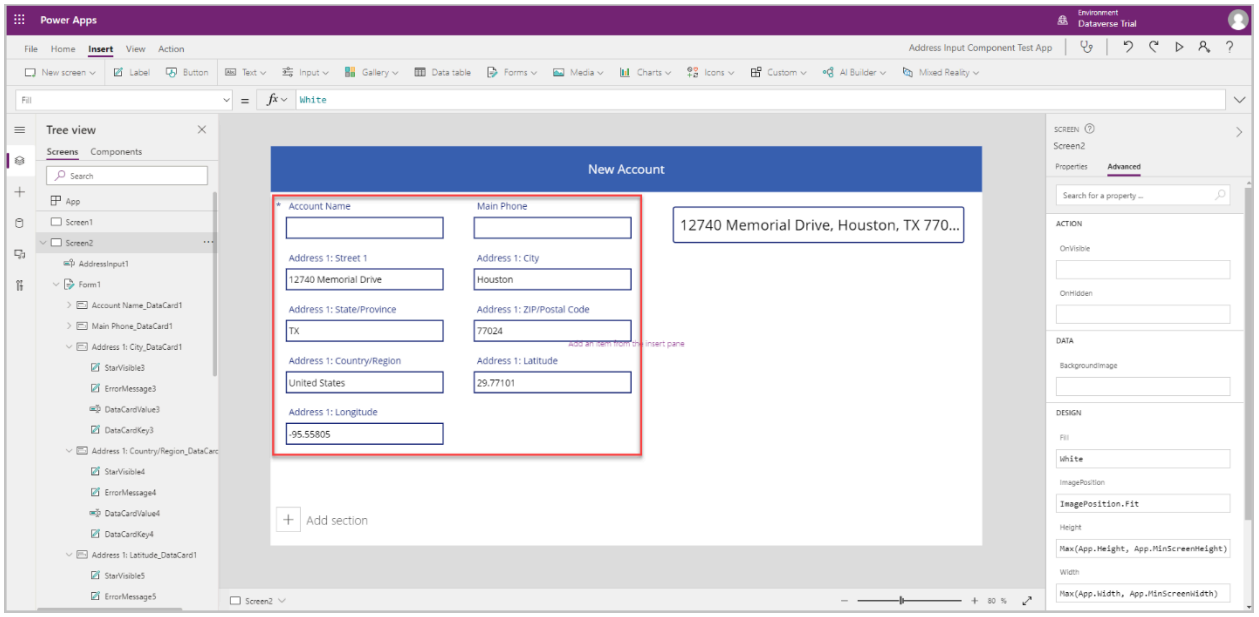
Data card	Value
Address 1: Street 1	`AddressInput1.StreetNumber & " " & AddressInput1.StreetName`
Address 1: City	`AddressInput1.Municipality`
Address 1: State/Province	`AddressInput1.CountrySubdivision`
Address 1: Zip/Postal Code	`AddressInput1.PostalCode`
Address 1: Country/Region	`AddressInput1.Country`

Data card	Value
Datacard	
Address 1: Latitude	`AddressInput1.SelectedLatitude`
Address 1: Longitude	`AddressInput1.SelectedLongitude`

Example for **Address 1: Street 1** data card:



The form gets populated with all the address fields when an address is selected in the **Address Input** control.



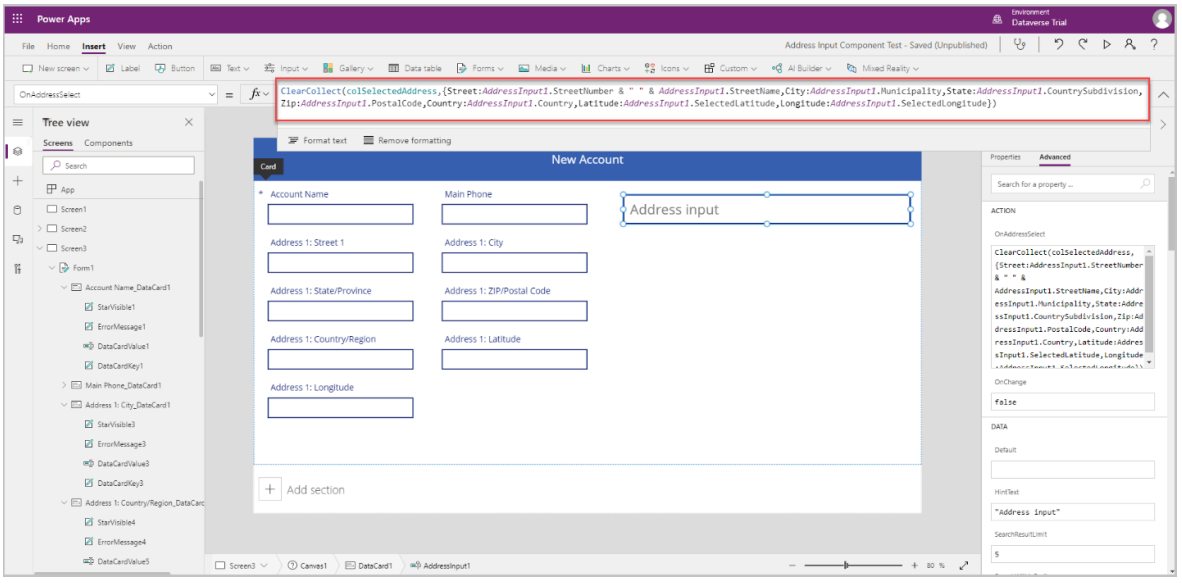
Map control

In this section, we'll see how to add the **Map** control on the app to show the selected location on the map.

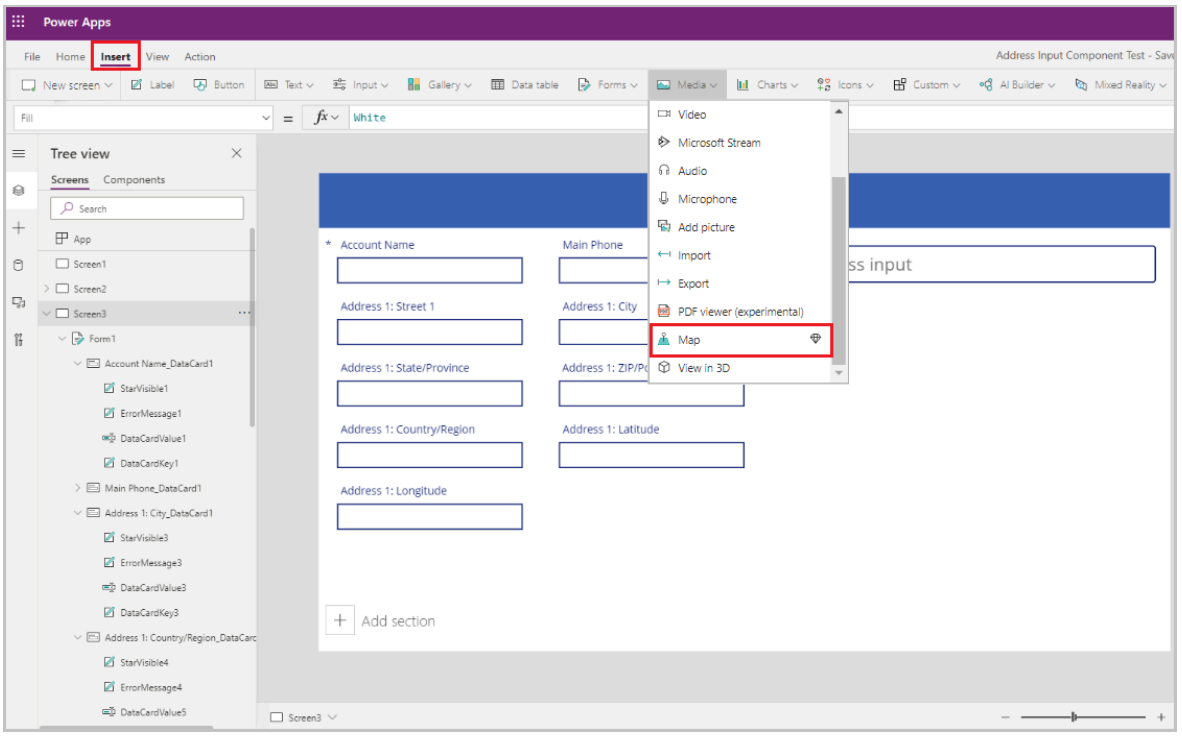
1. Before adding the **Map** control, we need to create a collection to capture the address from the address input – this collection will be used as the data source for the map.

   On the **OnAddressSelect** property of the **Address Input** control, enter the following formula to create a collection that captures the address data.

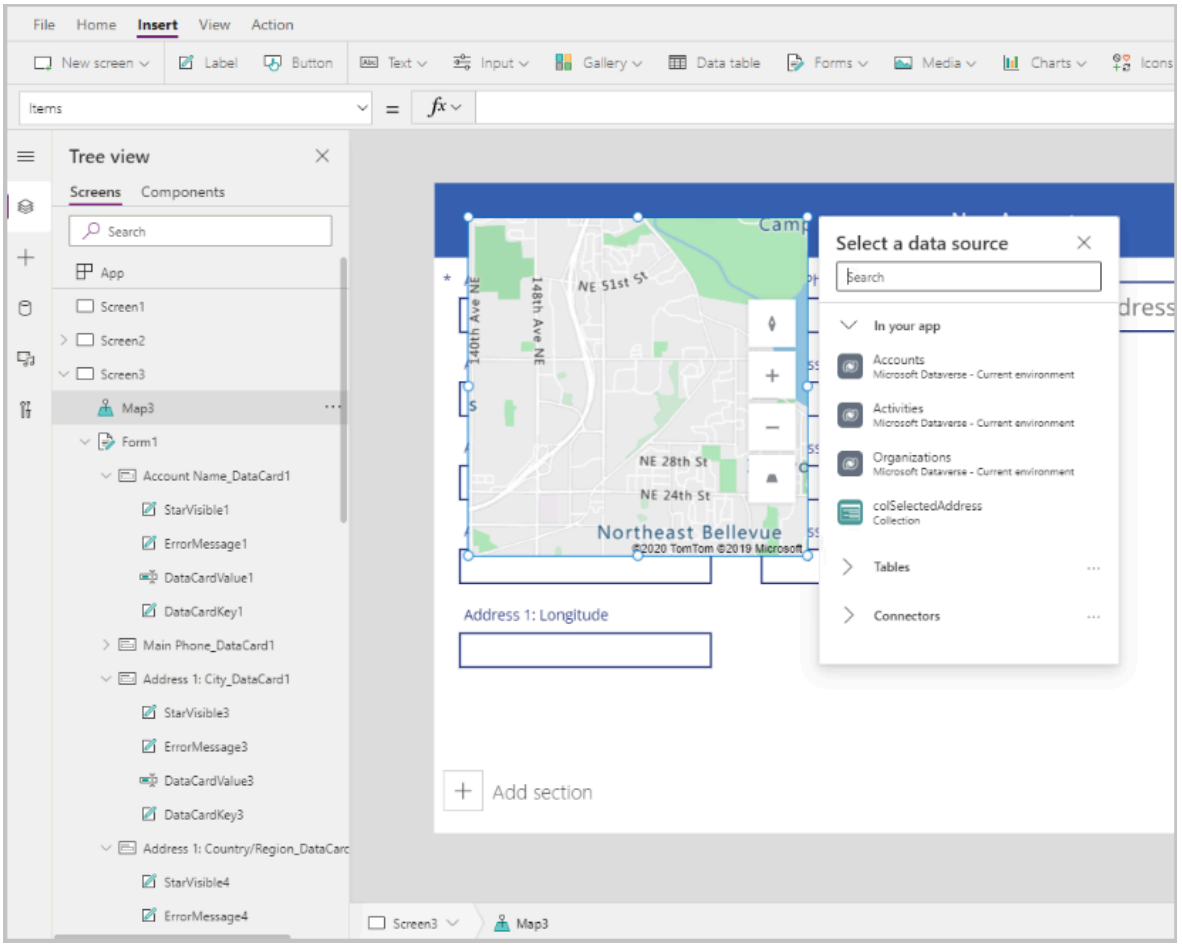
```
Power Fx

ClearCollect(
    colSelectedAddress,
    {
        Street: AddressInput1.StreetNumber & " " &
AddressInput1.StreetName,
        City: AddressInput1.Municipality,
        State: AddressInput1.CountrySubdivision,
        Zip: AddressInput1.PostalCode,
        Country: AddressInput1.Country,
        Latitude: AddressInput1.SelectedLatitude,
        Longitude: AddressInput1.SelectedLongitude
    }
)
```



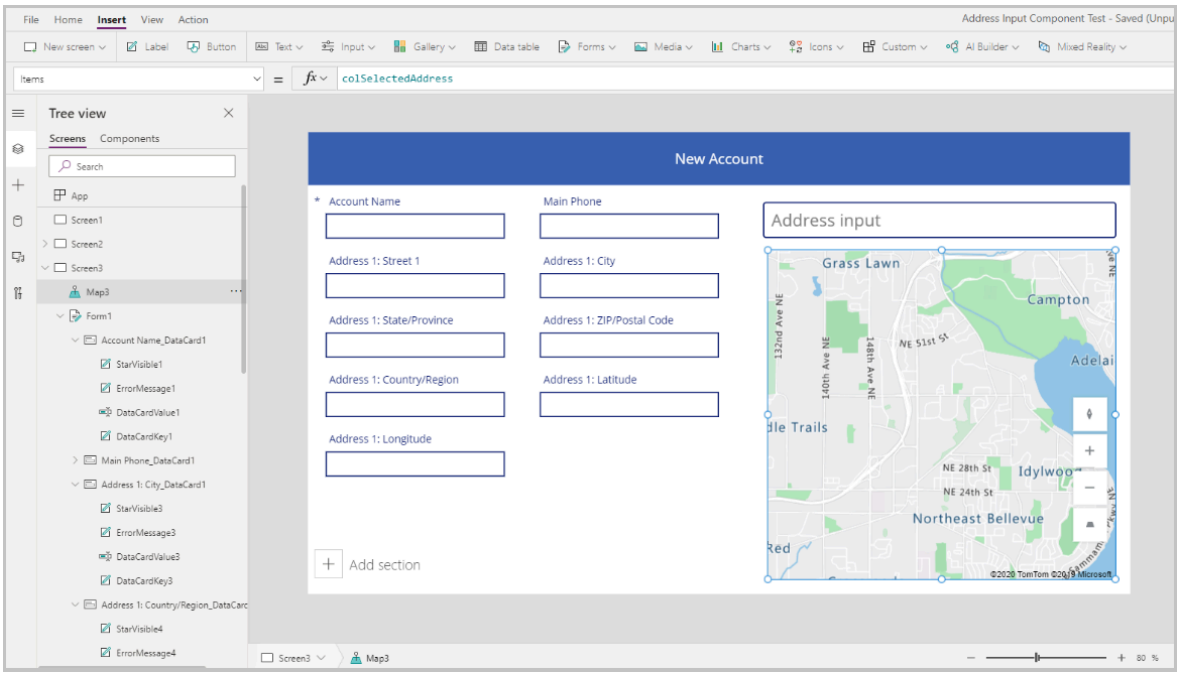
2. Select an area outside of the **Address Input** control, and select **Insert** > **Media** > **Map** to add the map control to the screen.



3. Select the collection as the data source.



4. Move the map below the address input section.



Configure properties for the Map control

In this section, we'll review and if necessary, set various properties of the map control.

The following table lists the properties available.

Expand table

Property	Description
Data source(Items)	Data source (table) that lists a predefined set of longitudes and latitudes to display as map pin on the map when it's loaded. Map each of the columns in your data by using the *ItemAddresses*, *ItemLongitudes*, *ItemLatitudes*, and *ItemLabels*.
Use default location	Whether the map initializes at a default location set by the user.
Default longitude	Longitude the map would go to when it's loaded if **Use default location** is enabled.
Default latitude	Latitude the map would go to when it's loaded if **Use default location** is enabled.
Default zoom level	Zoom level the map would be set to when it's loaded if **Use default location** is enabled.
Show current location	Whether the map should display the current location of the user.
Current location latitude	The latitude of the current location of the user if **Show Current Location** is enabled.

Property	Description
Current location longitude	The longitude of the current location of the user if **Show Current Location** is enabled.
Satellite view	Whether the style of the map is a satellite view or a road view.
Cluster pins	Whether the map pins are clustered.
Zoom control	Whether the zoom control appears on the map.
Compass control	Whether the compass control appears on the map.
Pitch control	Whether the pitch control appears on the map.
Pin color	The color of the pins.
ItemsLabels	A column in Items with the strings you want to use as labels for the pins.
ItemsAddresses	A column in Items with the strings that represent the location of the pins.
ItemsLongitudes	Name of the column in the table in your data source with floating-point numbers that represent the longitude position of the pins.
ItemsLatitudes	Name of the column in the table in your data source with floating-point numbers that represent the latitude position of the pins.
ItemsColors	Color of the pins.
ItemsIcons	Icon of the pins.
Items	Name of the table in your data source that contains all the records that you want to plot in the map by using pins. Each row must have an entry for the label, longitude, and latitude for each row.
OnMapClick	How the map responds when any location is selected.
OnSelect	How the app responds when a map pin is selected.
OnLoad	How the app responds when the map finishes loading.
OnItemsChange	How the app responds when the map pins change.
Show info cards	Whether the informational cards appear on the pins of the map.
Show shapes	Whether the shapes in **Shapes_Items** appear on the map.
Show shape labels	Whether the labels appear on the shapes of the map.
Enable shape drawing	Whether the drawing tools control appears on the map.

Property	Description
Enable shape deleting and label editing	Whether shapes can be deleted and their labels can be edited on the map.
Shapes_Items	Name of the table in your data source that contains all the records with the GeoJSON objects that you want to show in the map as shapes.
ShapeGeoJSONObjects	Name of the column in the table in your data source with strings that represent the GeoJSON objects of the shapes.
ShapeLabels	A column in **Shapes_Items** with the strings you want to use as labels for the shapes.
ShapeColors	Color of the shapes.
OnShapeSelected	How the app responds when a shape on the map is selected.
OnShapeCreated	How the app responds when a shape on the map is created.
OnShapeEdited	How the app responds when a shape on the map is edited.
OnShapeDeleted	How the app responds when a shape on the map is deleted.

Configure output properties for the Map control

The Map control outputs various properties when a user interacts with it inside an app. You can use these outputs in other controls or to customize the experience.

The following table lists the output properties available.

Expand table

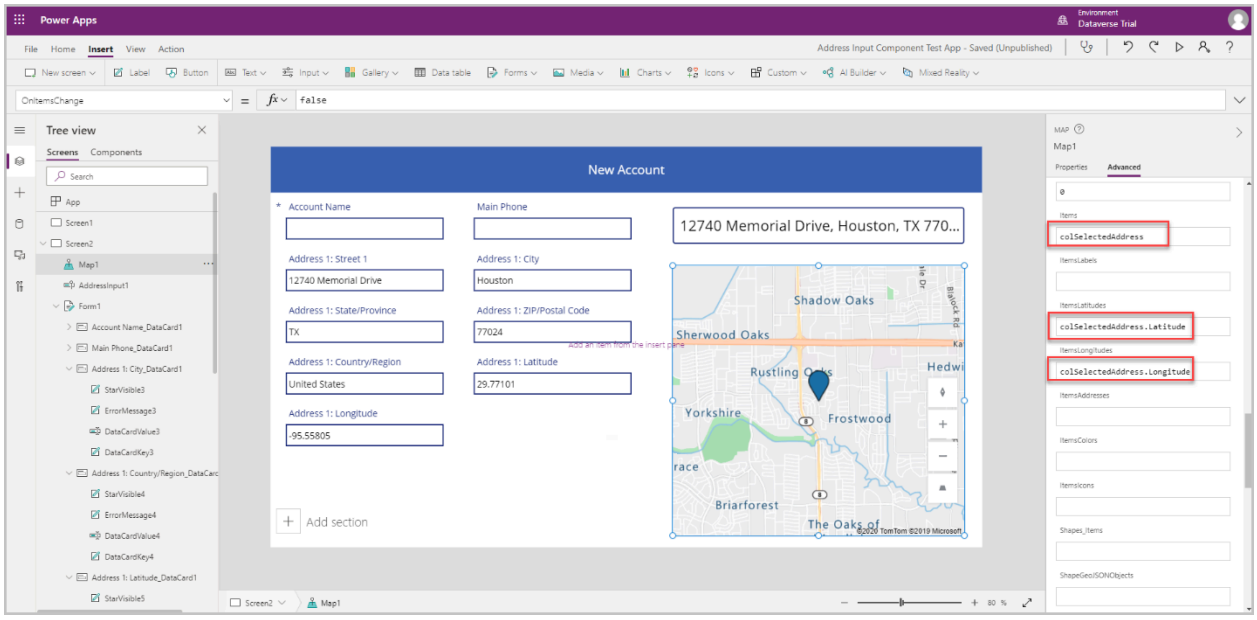
Property	Description
CenterLocation	Center location of the map as either `.Latitude` or `.Longitude`. The output will be an integer. For example, calling `Map1.CenterLocation.Latitude` will output a single integer such as "47.60357".
Selected	The selected pin on the map.
SelectedItems	The selected pin or pins of the selected cluster on the map.
GeocodedItems	The geo-coded locations of the pins on the map.
ClickedLocation	The last clicked location on the map as either `.Latitude` or `.Longitude`.

Property	Description
Shapes_Selected	The record of the selected shape from **Shapes_Items**.
Shapes_SelectedItems	The records of the selected overlapping shapes from **Shapes_Items**.
SelectedShape	The selected shape on the map with `.Perimeter` and `.Area`.
DeletedShape	The last deleted shape on the map with `.Perimeter` and `.Area`.
GeoJSON	The list of shapes on the map in Feature Collection GeoJSON format.

To drop a pin on the address entered in the address input, we'll use the collection we created on the **OnAddressSelect** property and use that in the map control.

Expand table

Property	Value
Items	`colSelectedAddress`
ItemsLatitudes	`colSelectedAddress.Latitude`
ItemsLongitude	`colSelectedAddress.Longitude`



Some of the properties of the pin can be customized per location pin are explained below:

1. **ItemsColors** - To update the color of the pin to a different color for each location, add a column to the collection **Color: Red**, and use that in the **ItemsColors** data property of the **Map** control.

2. **ItemsLabels** - To update the label of the pin to a different label showing the address for each location, add a column to the collection **Label:AddressInput1.FreeformAddress**, and use that in the **ItemsLabels** data property of the **Map** control.

3. **ItemsIcons** - To change the icon of the pin for each location, add a column to the collection and use that in the **ItemsIcons** data property of the **Map** control. In this example, we'll add a triangle icon with the formula as `Icon: triangle`.

   > **Tip**
   >
   > For the complete list of icons, go to **List of image templates**.

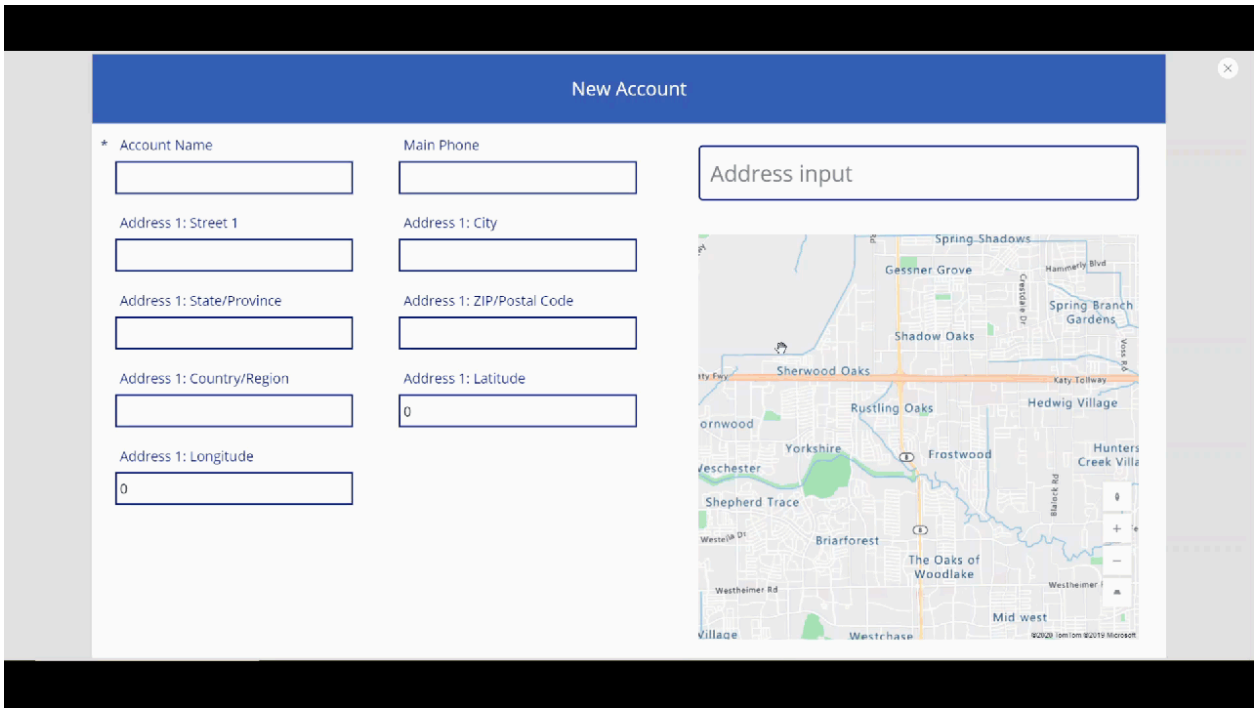
   Power Fx

   ```
   ClearCollect(
       colSelectedAddress,
       {
           Street: AddressInput1.StreetNumber & " " &
   AddressInput1.StreetName,
           City: AddressInput1.Municipality,
           State: AddressInput1.CountrySubdivision,
           Zip: AddressInput1.PostalCode,
           Country: AddressInput1.Country,
           Latitude: AddressInput1.SelectedLatitude,
           Longitude: AddressInput1.SelectedLongitude,
           Label: AddressInput1.FreeformAddress,
           Color: Red,
           Icon: Triangle
       }
   );
   ```

After adding these to the collection, go to the **Advanced** tab of the **Map** control, and update the **ItemsColors**, **ItemsLabels** and **ItemsIcons** properties.



Run the app in preview mode to test. The map shows the location selected in the address input with a triangle icon and the address shows up as a label on the map.



See also

Interactive map control

Feedback

Was this page helpful? Yes No

Provide product feedback

Create an app using the Measuring Camera control

Article • 12/16/2022

You can use the Measuring Camera control in your app to let users measure distance, area, and volume. During the measurement, you create two-dimensional and three-dimensional polygons that can be used to confirm how a certain-sized object would fit within a space.

The control creates a button in your app. When app users select the button, it shows a live camera feed of the device. App users can then specify a starting point, and one or multiple sequential endpoints to measure between. The instances of measured segments are shown directly on the live camera feed.

When the user exits the control, the measurements that were taken are captured in the **Measurements** property so they can be saved or stored.

Screenshots taken during the mixed reality experience are stored in the **Photos** property for viewing within the app.

In this article, we'll create a screen in an app to use the **Measuring camera** control to measure the area of a user-defined space as well as view a photo of the space that was measured.

Watch this video to learn how to build mobile apps with Measuring Camera control: https://learn-video.azurefd.net/vod/player?id=94d7ae60-3cb4-41e5-80bd-d8212fa3e4a8&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fhow-to%2Fbuild-apps-measure-in-mr

Prerequisites

See prerequisites for mixed-reality capable devices.

Create the app with Measuring Camera control

1. Open an app in Power Apps Studio.
2. Select the **Insert** tab.
3. Expand **Media**.

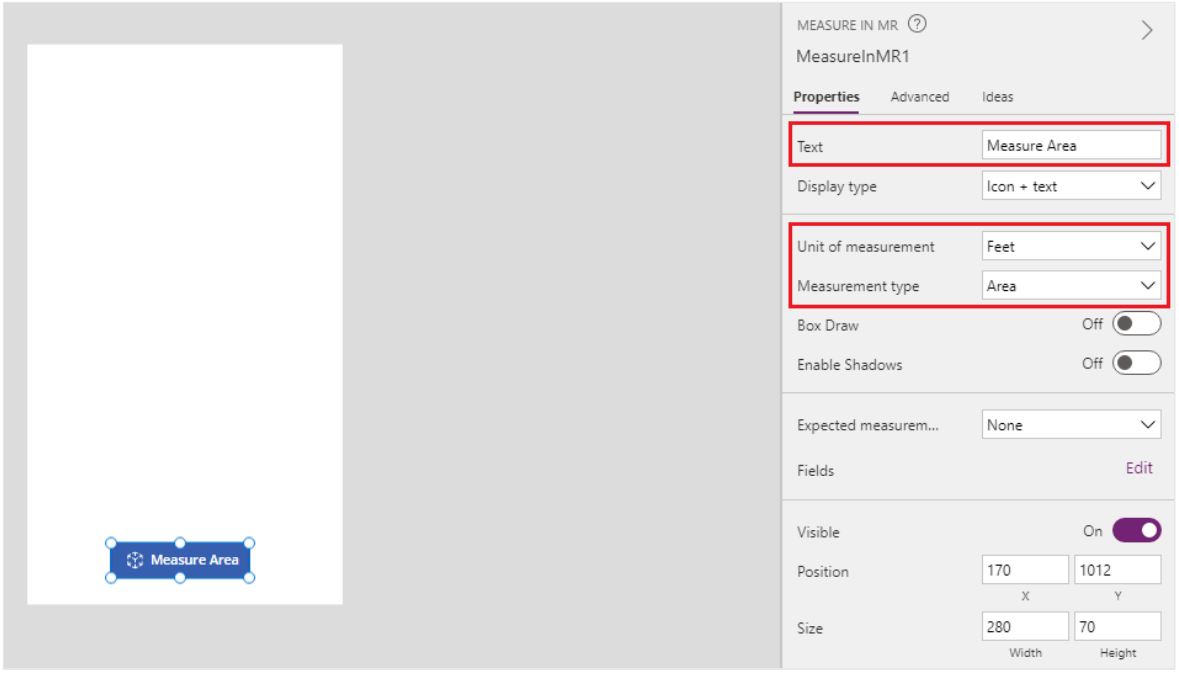
4. Select the **Measuring camera** control to insert it on the screen.

5. Position the control near the bottom of the screen.

6. Change the following properties for the control using the property pane on the right-side of the screen.

Expand table

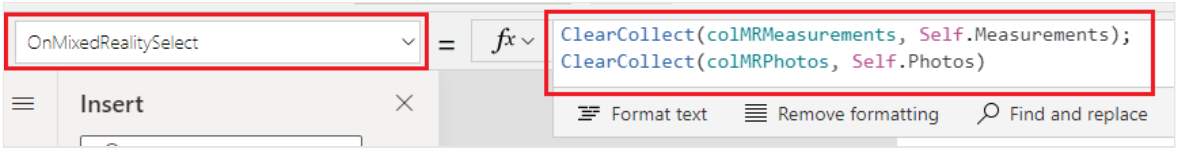
Property	Value
Text	Measure Area
Unit of measurement	Feet
Measurement type	Area



7. Select the Advanced tab and change the **OnMixedRealitySelect** property to the following formula:

Power Fx

```
ClearCollect(colMRMeasurements, Self.Measurements);
ClearCollect(colMRPhotos, Self.Photos)
```



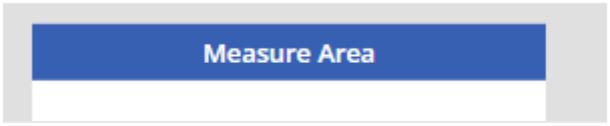
This formula will allow the user to define areas to be measured in square feet and store the most recent results of the measurements in a collection named **colMRMeasurements**, and the most recent photos in a collection named **colMRPhotos**.

8. Add a text label to the screen and move it to the top-left corner of the screen.

9. Change the following properties of the added text label:

Expand table

Property	Value
Text	Measure Area
Font size	24
Font weight	Semibold
Text alignment	Center
Width	Parent.Width
Color	`RGBA(255, 255, 255, 1)`
Fill	`RGBA(56, 96, 178, 1)`



This label provides a header for the screen.

10. Add an image control, and move it just below the header label.

11. Change the following properties for the image control:

Expand table

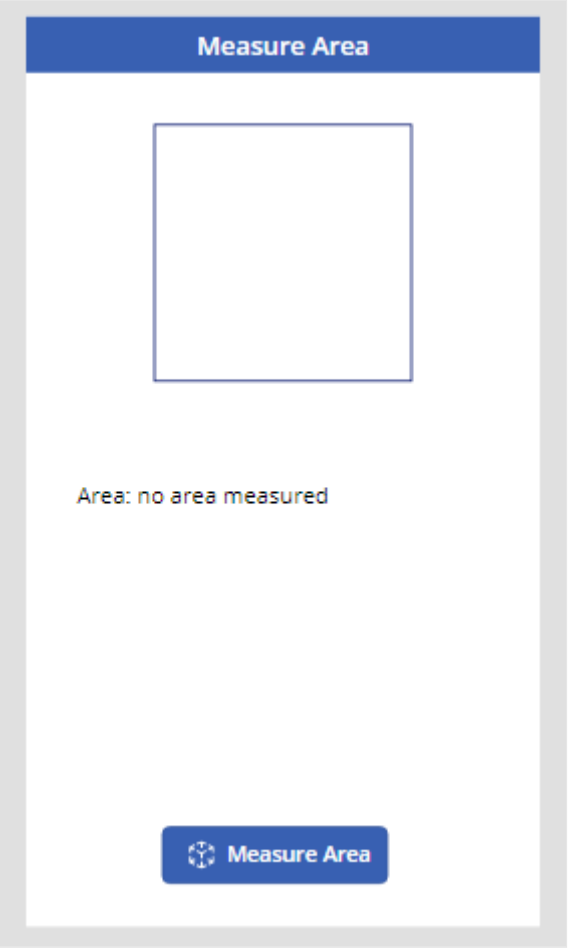
Property	Value
Image	`First(colMRPhotos).ImageURI`
Border Thickness	2

The image control will display the first image taken from the **Measuring camera** control.

12. Add another label to the screen, positioning it below the image control.

13. Change the **Text** property of the label to the following formula:

```
Power Fx

"Area: " & If(IsEmpty(colMRMeasurements), "no area measured",
First(colMRMeasurements).Area & " sq. " &
First(colMRMeasurements).Unit)
```



This label will display the first area measurement and unit type that is collected from the **Measuring camera** control. If there's no measurement to be returned, the text will show **Area: no area measured** to let the user know that no value is present.

Test the app

> **Tip**
>
> Watch the video embedded at the top of this article to see how the app works.

Now that all the controls have been added, save and publish the app. On a mixed-reality capable device (as defined in the Prerequisites section), open the app, and press the **Measure Area** button. This action will open the **Measuring camera** experience, allowing the user to define an area to be measured.

Follow the on-screen instructions to calibrate the device by slowly moving it left and right while pointing the camera at a surface to be measured. Once the calibration is complete, you'll see an array of dots on the surface as well as a circle and dot reticle in the center of the screen. This reticle is used to define lengths, areas, and volumes by using the + button to add points and line segments.

Photos can be captured using the camera button. These photos will show the lengths, areas, and volumes defined by the user.

There's an undo button to undo any placed points.

Begin by placing a point using the + button. You'll notice that an orange line and measurement now appear connected to the point you placed. Place a second point and notice how a line with measurement appears between the two points you place. Continue placing points until the desired shape has been fully defined. To close a shape, place the reticle on the first point made and use the + button. You'll notice that the center point of the reticle changes color from white to green, indicating that the shape can be closed. If two of the sides of the shape intersect, or you try to close a shape using any other point than the first one. You'll notice that the center point of the reticle changes from white to red, indicating that a point cannot be placed there.

Once an area has been fully defined, position the device so that a picture can be taken of the area you defined and use the camera button to do so. The select the **Submit** button. A confirmation dialog will appear, asking if you've finished with your measurements. If you have more measurements to enter, select **Cancel** to return to the MR environment. If you're finished, select **Submit** to be taken back to the app screen.

You'll see the first picture taken as well as the calculated area of the shape you defined as well as the units, in this case square-feet.

The measurement and photo can then be used like any other photo or text field in Power Apps, for instance to be sent in an email or stored in a table in Microsoft Dataverse.

See also

- Measuring Camera control
- Add mixed reality controls to your canvas app

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a canvas app with deep link to a specific screen

Article • 12/16/2022

A common scenario when building apps is the need to share a "deep link" to a specific screen. Deep links are useful when you want to get users straight to a specific screen and data rather than asking them to navigate from the "home" screen of your app.

To deep link into Power Apps, you'll use this URL syntax: `https://apps.powerapps.com/play/{App ID}?{Query}`.

In this syntax:

- **App ID**—ID of the app. Go to Power Apps > **Apps** > Select your app > **Details**.
- **Query**—The query text allows you to supply the data to deep link to. You'll need to make some code changes to your canvas app to use the provided parameters to open the app using the query URL.

In this article, you'll learn about how to:

- Set up a canvas app to handle the query parameter.
- Create a UI for emailing a deep link directly from a screen of your app.
- Use the app to browse to a specific account, send an email with the deep link, and use the link from the received email to open the app directly to that account.

Prerequisites

- Power Apps license
- If you're new to Power Apps, familiarize yourself with Power Apps basics by generating an app and then customizing that app's controls, gallery, forms, and cards.
- To create an app, you must be assigned to the Environment Maker security role.

Create the app

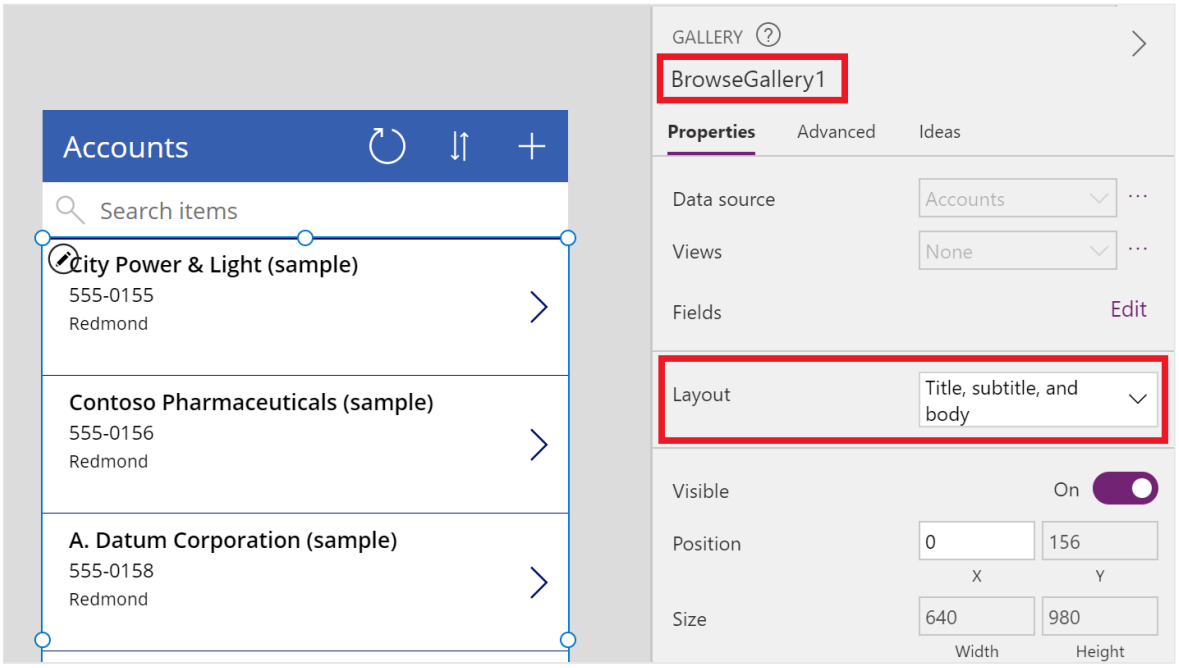
Create an app using the **Accounts** table in Microsoft Dataverse.

Your app opens to the **BrowseScreen**, that shows a list of accounts in a gallery. By default, this gallery shows an email address, a city, and an account name. The app also contains **DetailScreen** and **EditScreen**.

Update navigation

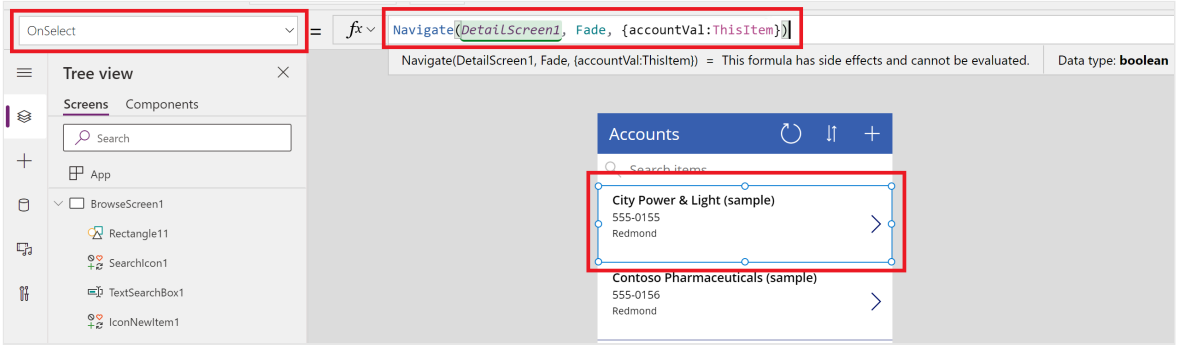
Now you'll set the navigation to use a context that we'll use to take the user to another screen.

1. On **BrowseScreen** screen, select **Layout** under gallery properties, and select **Title, subtitle and body**



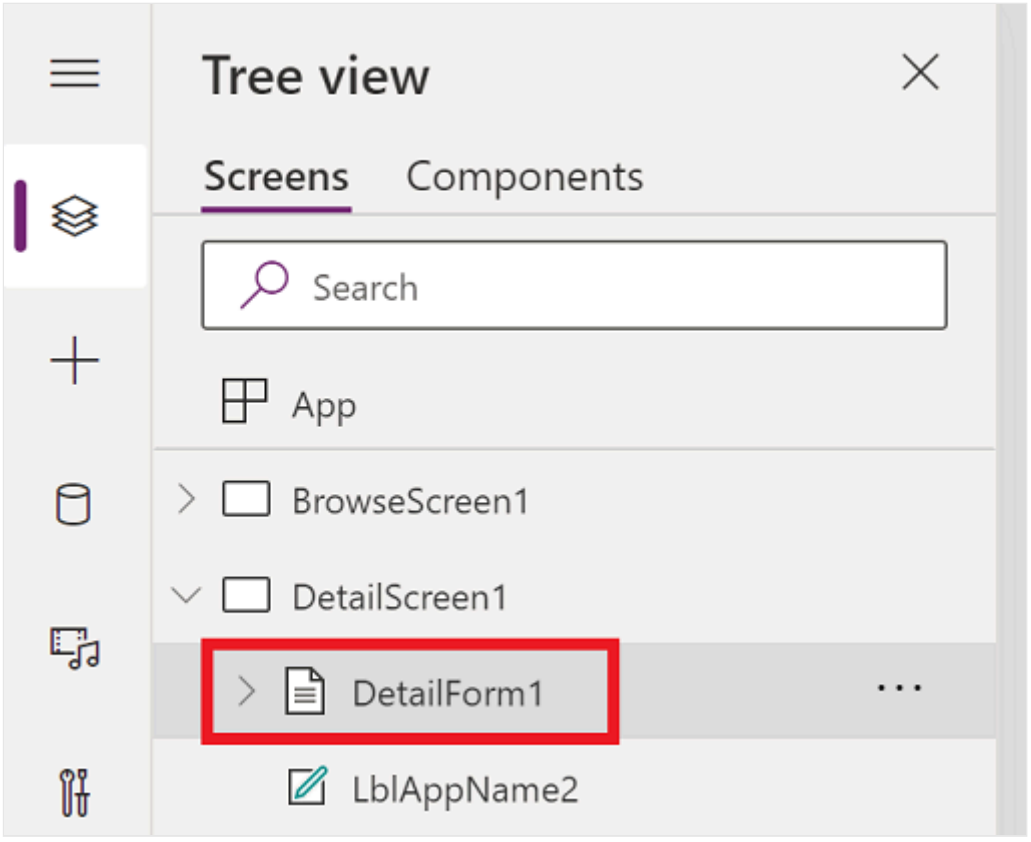
2. Select first record of **BrowseGallery** gallery. And then, under **OnSelect**, enter the following:

```
Power Fx

Navigate(DetailScreen1, Fade, {accountVal:ThisItem})
```

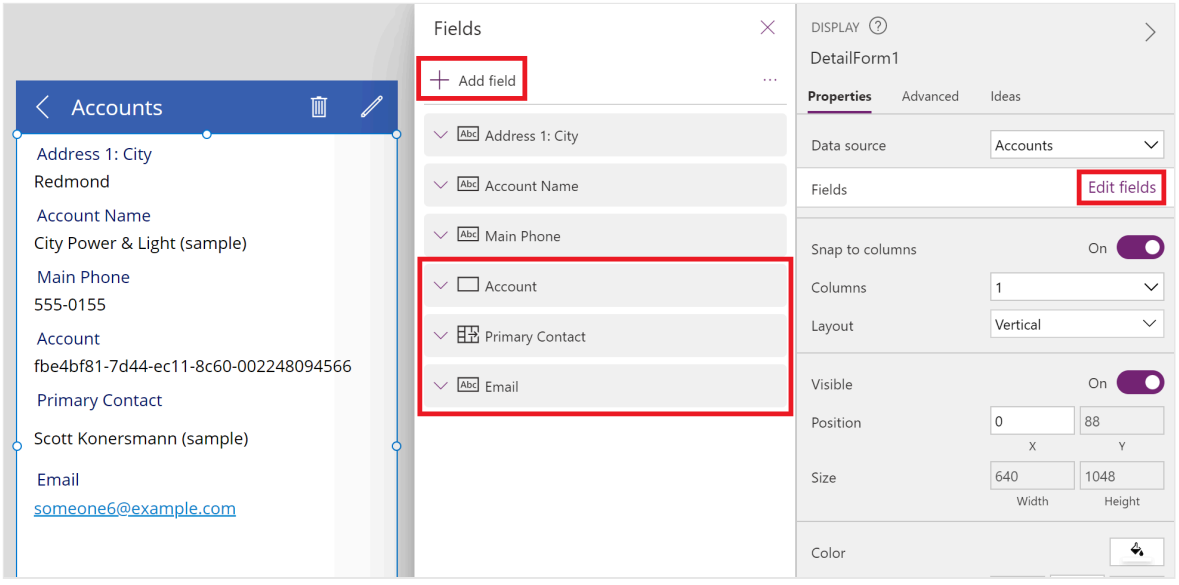


The Navigate() function creates a context variable called **accountVal**. This variable is populated whenever you select a record from the gallery. Upon selecting record, you'll be taken to the **DetailScreen** with the selected value (in this example, account) in **accountVal**.

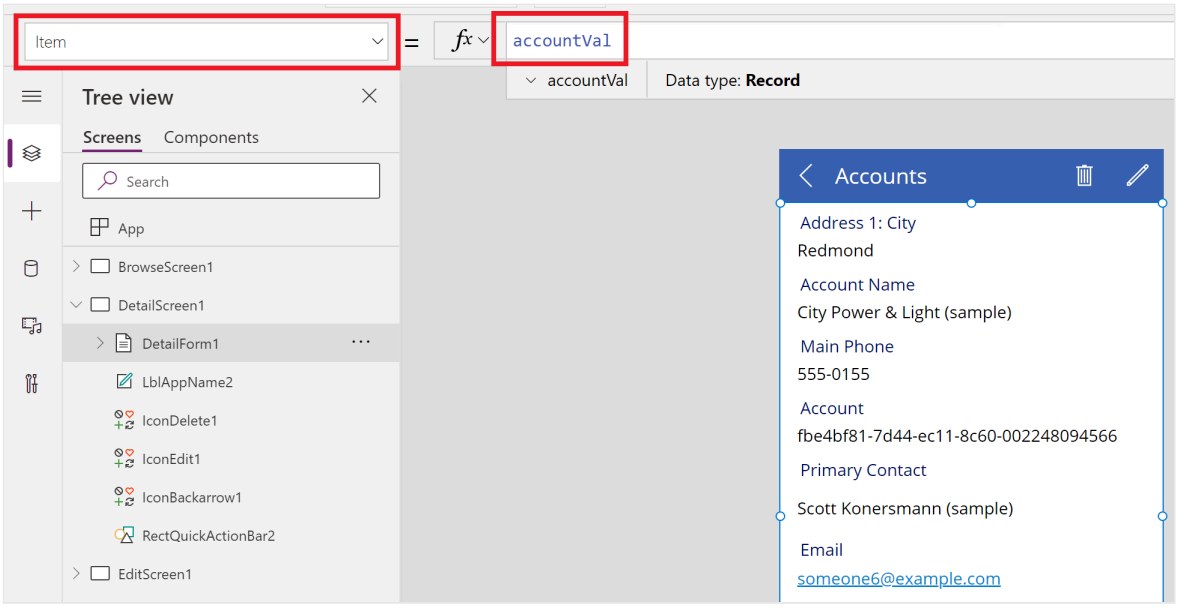
3. From the **Tree view** on the left-side of the screen, expand **DetailScreen1**, and select **DetailForm1**.



4. From the properties pane on the right-side of the screen, select **Edit fields**, and add **Account**, **Primary Contact**, and **Email** fields.



5. Select **DetailForm**, and set **Item** to **accountVal**.

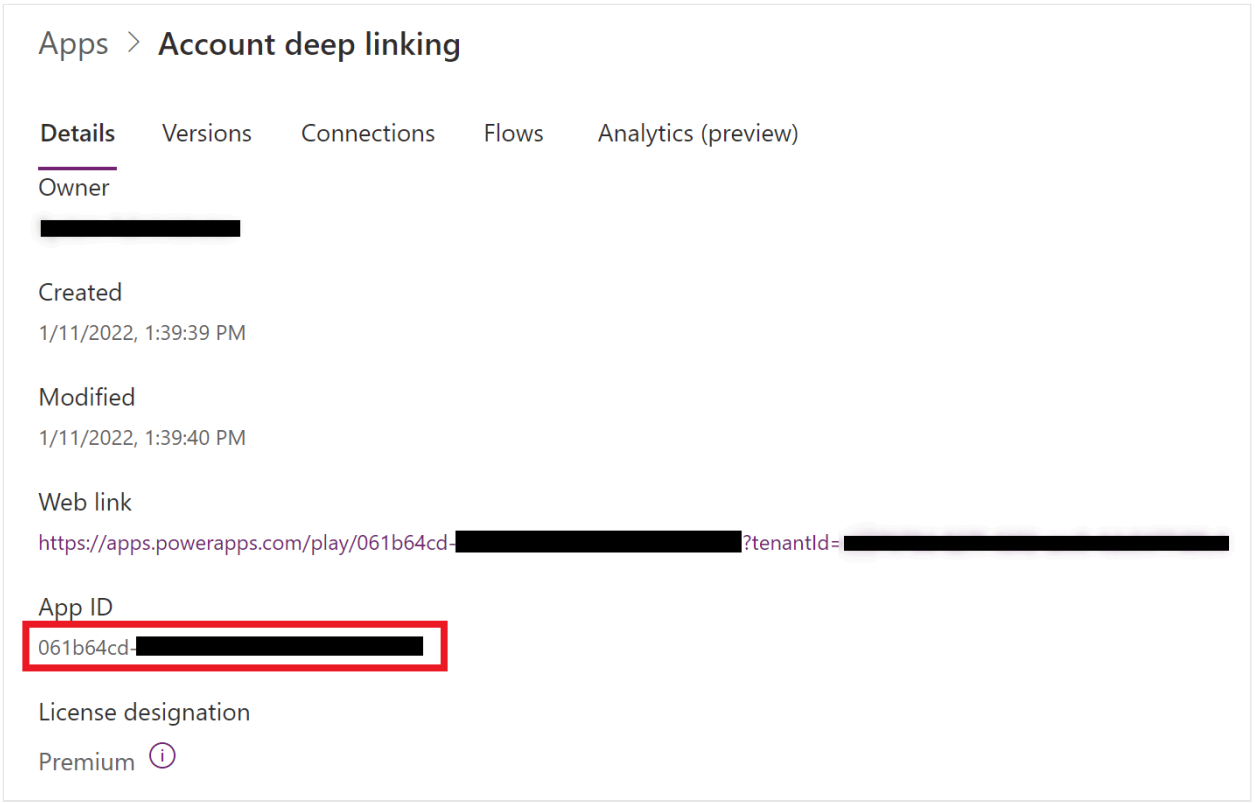


6. Save with the name "Account deep linking".

Get the App ID

App ID is a unique GUID representing a given app, and generated when you save the app to cloud.

To get the **App ID**, open a new tab and go to Power Apps > **Apps** > Select your app > **Details**. More information: Get an App ID



Enable deep link to a screen

The goal of this example is to take users to the **DetailsScreen1** when the app is launched with the **accountId** query parameter as part of the app URL. For this purpose, we'll use function Param() that retrieves the query string parameter supplied while launching the app.

1. Select **App** from the **Tree view** on the left-side of the screen.

   > **Tip**
   >
   > If you've closed Power Apps Studio, reopen the saved "Account deep linking" app by **editing** the app.

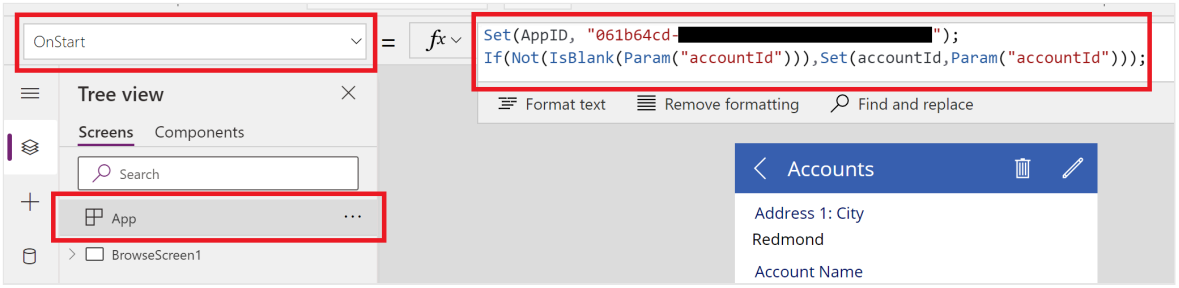
2. Select **OnStart** property from the list of properties.

3. To store the **App ID** value, update the **OnStart** property formula as below.

   ```
   Power Fx

   Set(AppID, "GUID");
   If(Not(IsBlank(Param("accountId"))),Set(accountId,Param("accountId")));
   ```

   In this formula, function Set() sets the **App ID** to the GUID of the app. And function Param() stores the value in the **accountId** variable, when not blank.

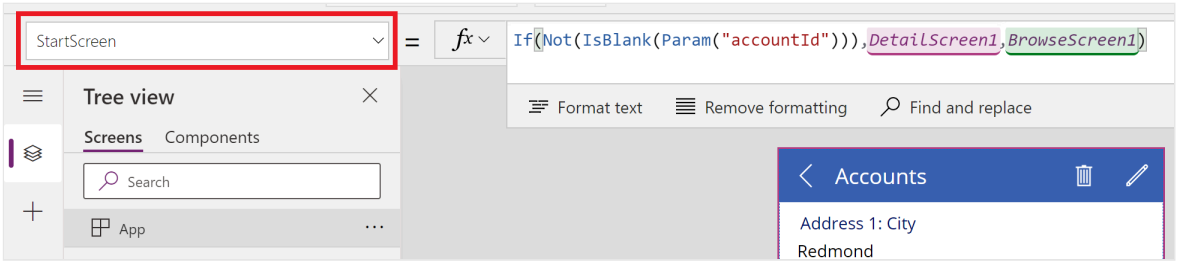
   > **Note**
   >
   > Ensure you replace "GUID" in the above formula to your App ID as appropriate.

4. Select **StartScreen** property for the app, and add the following:

   ```
   Power Fx

   If(Not(IsBlank(Param("accountId"))),DetailScreen1,BrowseScreen1)
   ```

This function checks if the Param is blank, then stay on the **BrowseScreen1**. Else, go to **DetailScreen1**.

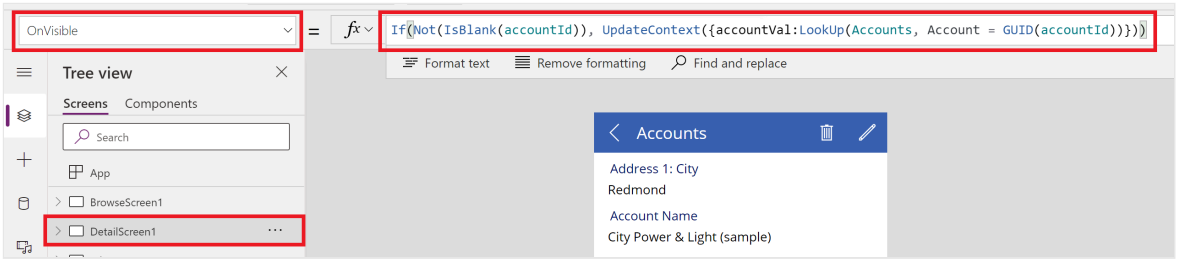


5. Select **DetailScreen1** from the **Tree view**.

6. Select **OnVisible** property, and enter the following:

```
Power Fx

If(Not(IsBlank(accountId)), UpdateContext({accountVal:LookUp(Accounts,
Account = GUID(accountId))}))
```

This function uses **accountId** stored from the previous step to set **accountVal** variable.



Create the deep link URL

Your app is now configured to receive **accountId** as the parameter that contains the GUID for an account from the **Accounts** table. And when this parameter is provided with the request to the app, the app will directly open the details screen (**DetailScreen1**) with the given **accountId**.

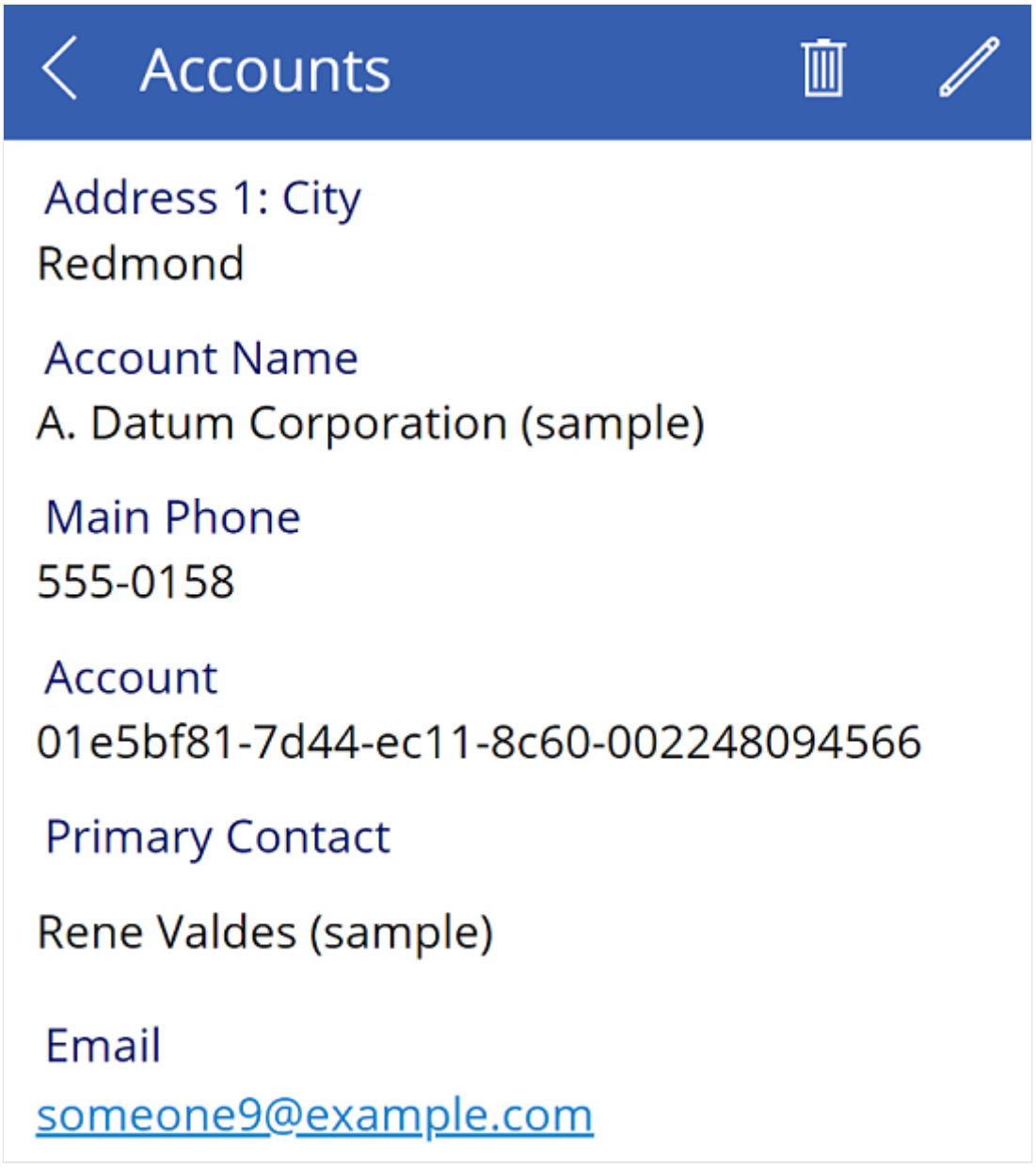
To invoke app with the parameter value, we have to use the following syntax:

`https://apps.powerapps.com/play/{App ID}?{Query}`

In the above syntax, we have to add the **App ID** and the query that contains **accountId** variable with its value.

For example, when the following URL was entered in a browser for the sample tenant, the app **Account deep linking** opened directly with the supplied **accountId** parameter

value (in this example, "A. Datum Corporation (Sample)"):
`https://apps.powerapps.com/play/061b64cd-e5a0-4a7a-a77f-b6f8586dd6c7?accountId=01e5bf81-7d44-ec11-8c60-002248094566`



Send deep link in email

We can also supply a variable **accountVal** to point to the account's ID instead of using absolute link explained above while invoking the URL for the given account from inside the app.

For this purpose, we'll create a button and add the ability to invoke an email from within the app with the deep link to the selected account's details screen from the **Account deep linking** app.

1. Select **Data** from the left-side of the screen.

2. Select **Add data** > search for and select **Office 365 Outlook** > select **Connect**.

> **Note**
>
> This action adds the Microsoft 365 Outlook connection so that we can use the next steps to invoke composing an email to share the account information.

3. Select **DetailScreen1** from the **Tree view**.

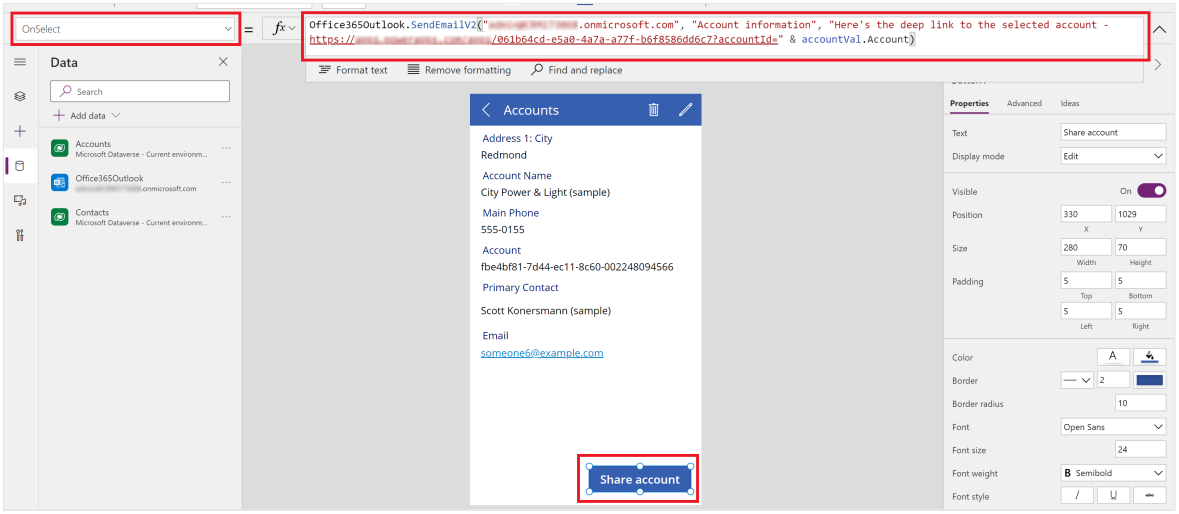
4. Insert a button at bottom-right of the screen, and rename it to **Share Account**.

5. On the button's **OnSelect** property, enter the following:

```
Power Fx

Office365Outlook.SendEmailV2("Recipient", "Subject", "Here's the deep
link to the selected account - https://apps.powerapps.com/play/{App
ID}?accountId=" & accountVal.Account)
```

This formula uses the Microsoft 365 connector for Outlook to send an email using the SendEmailV2 operation.

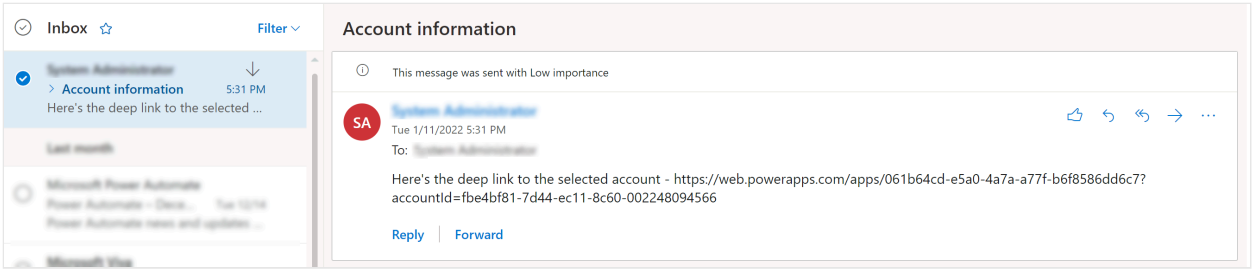
> **Note**
>
> Replace **Recipient**, **Subject**, and **AppID** in the above formula with the email address of the recipient, subject for the email, and the ID of the app.



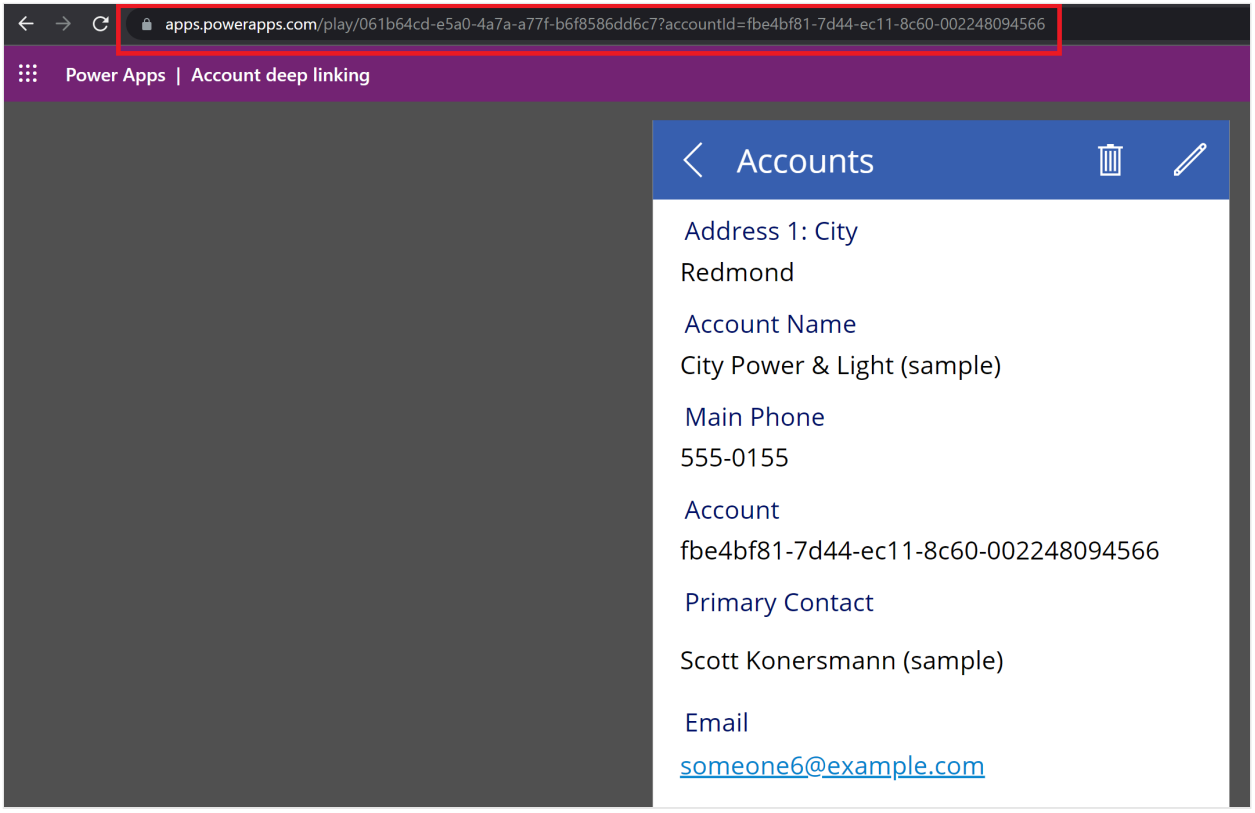
6. Save and publish the app.

Test the app

Run the app, and go to the details screen for any account. And then, select the button **Share account**. The **OnSelect** button formula triggers an email to the recipient with the subject and body configured earlier. Here's a sample email sent by the configured app:



Open the email, and copy the deep link that points to the selected account, and you'll be taken directly to the account details screen in the **Account deep linking** app, instead of the default browse screen.



See also

- Create a canvas app from a template
- Add and configure controls in canvas apps
- Get started with formulas in canvas apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Work with the PDF function (experimental)

Article • 04/11/2023

The PDF function in Power Apps generates a PDF document from the contents of a screen or certain types of controls. You can pass the generated file to an action connector—for example, to send it in an email using the Office 365 Outlook connector—or to a Power Automate flow as part of an automation scenario.

> **Important**
>
> - This is an experimental feature. Experimental features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback. **Understand experimental, preview, and retired features in canvas apps**.
> - You must turn on the PDF function in **Settings > Upcoming features > Experimental**. As an experimental feature, it's disabled by default.
> - Send your feedback in the **Power Apps experimental features community forum**.

Turn on the PDF function

Since the PDF function is an experimental feature, it's turned off by default. To use it in your apps, you need to turn it on manually.

1. Sign in to Power Apps and create a canvas app or open an existing app.
2. In Power Apps Studio, on the command bar, select **Settings**.
3. Select **Upcoming features**.
4. On the **Experimental** tab, turn on **PDF function**.

Use the PDF function to generate a PDF file

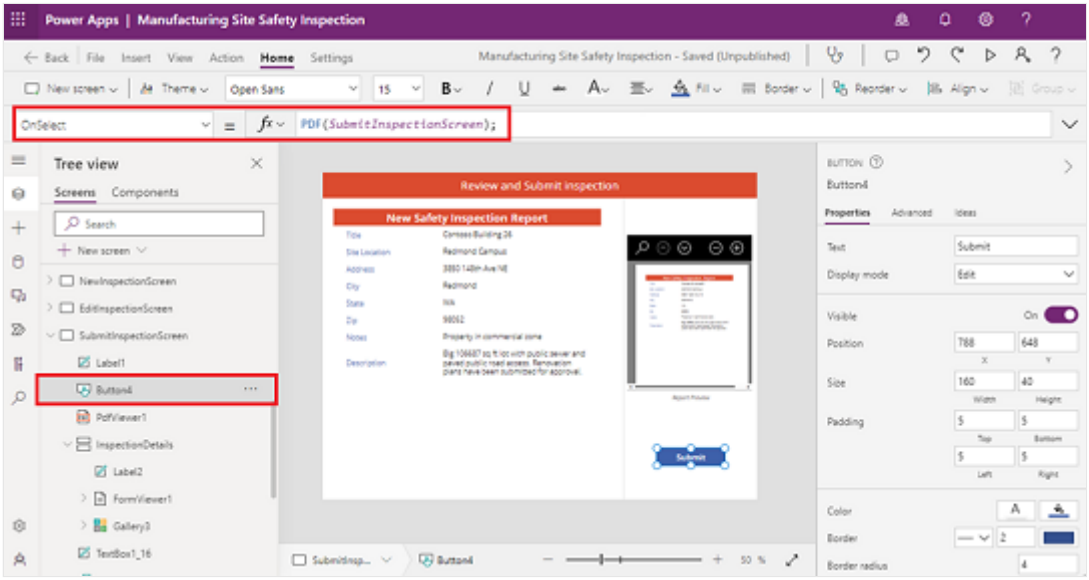
In the following example, we use the PDF function in the `OnSelect` property of a button to generate a PDF file. You can also use the PDF function in `OnVisible`, `OnHidden`, or any

other behavior property.

1. Select an existing button or add a new button to your canvas app.

2. In the button's `OnSelect` property, add the PDF function and enter the target of the function as a parameter. Enter the formula: `PDF(SubmitInspectionScreen);`

   In this example, we're targeting the screen `SubmitInspectionScreen` to generate a PDF file from the contents of our sample app's Inspection screen. Replace it with the name of a screen in your app.



3. Select **Save and publish**.

Play the app and select the button you added. The PDF file is generated and stored in memory for future use. The following section suggests a few ways to work with the PDF file in your app.

The PDF function can only target the screen that's currently visible. You can pass the entire screen, as in our example, or enter more parameters to specify a container or gallery to limit the content to pass.

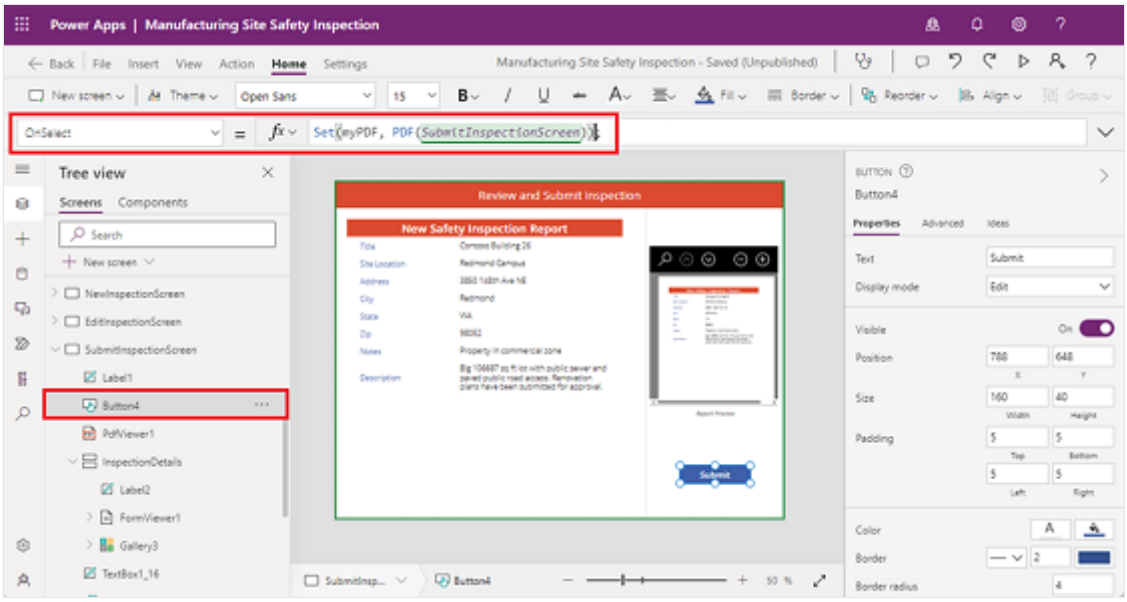
Work with the generated PDF file

You can use the generated PDF file in many ways. Here are a few, using the example in the previous section.

View the file with the PDF viewer control

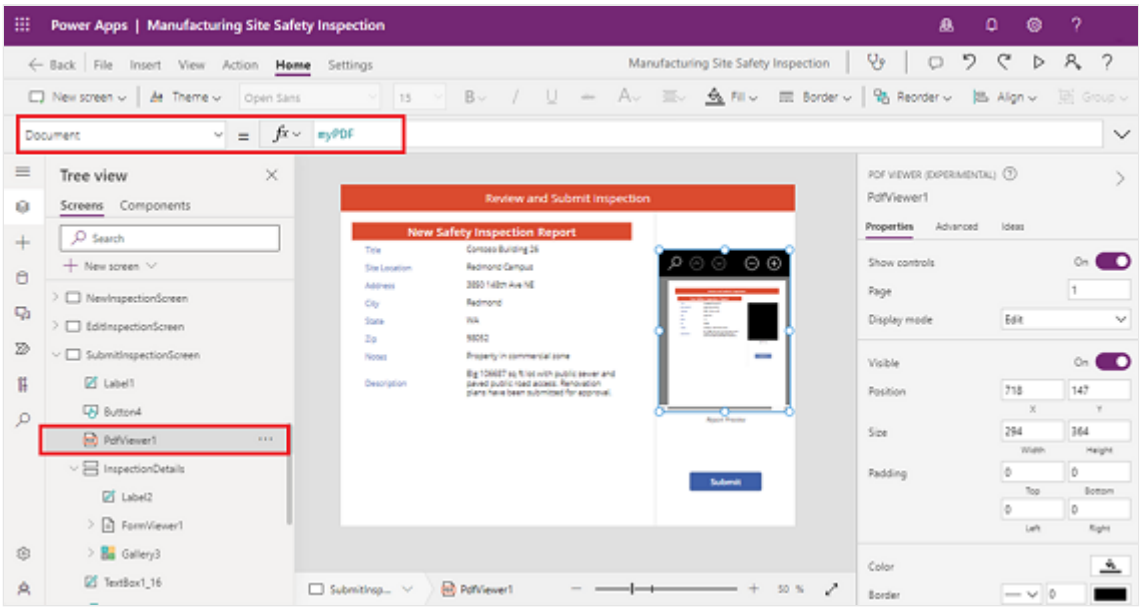
1. In your canvas app, select the button that you created in the previous example or add a new button.

2. In the button's `OnSelect` property, enter the following formula, replacing `SubmitInspectionScreen` with the name of a screen in your app: `Set(myPDF, PDF(SubmitInspectionScreen));`



3. Select **Insert** > **PDF viewer (experimental)**.

4. From the properties list, select the **Document** property and then enter **myPDF** in the formula bar.



5. Select Save and publish.

Play the app and select the button you added. The PDF file is generated, stored in the variable `myPDF`, and visible in the PDF viewer control.

The PDF viewer control appears as a black shape inside the generated PDF file. To exclude the control from the generated file, place the content you want to turn into a

PDF file inside a container or gallery and target that instead of the screen; for example, `PDF(Container1)` or `PDF(Gallery1).`

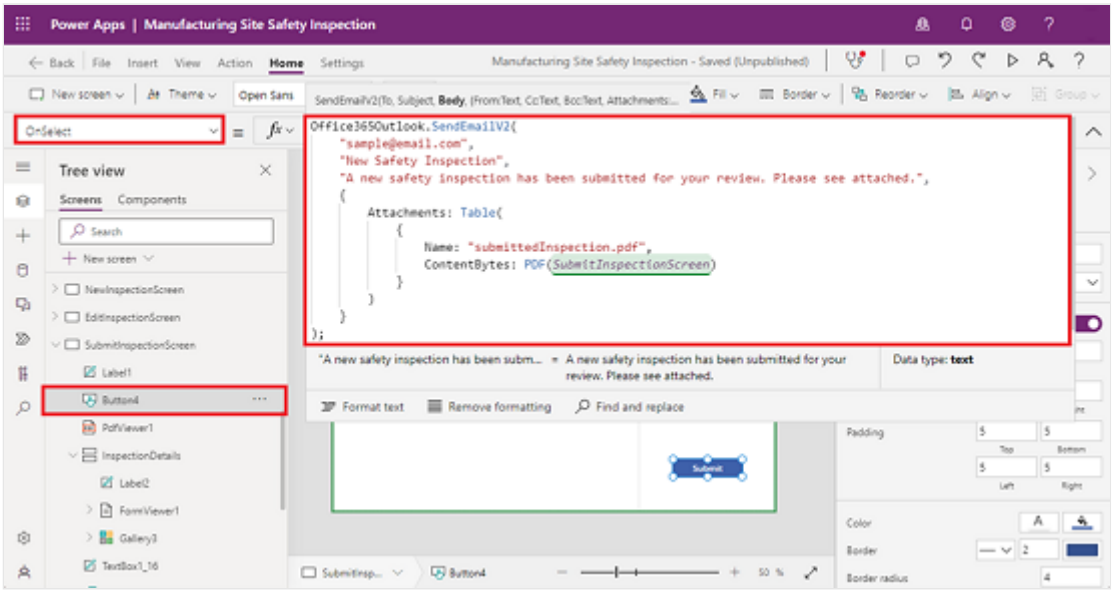
Use an action connector

In this example, we generate a PDF file that we send as an email attachment using the Office 365 Outlook connector.

1. In your canvas app, select the button that you created in the previous example or add a new button.

2. In the button's `OnSelect` property, enter the following formula:

```
Office365Outlook.SendEmailV2(
   "sample@email.com",
    "New Safety Inspection"
    "A new safety inspection has been submitted for your review. Please
see attached.",
    {
        Attachments: Table(
             {
                 Name: "submittedInspection.pdf",
                 ContentBytes: PDF(SubmitInspectionScreen)
             }
         )
    }
 );
```



3. Select Save and publish.

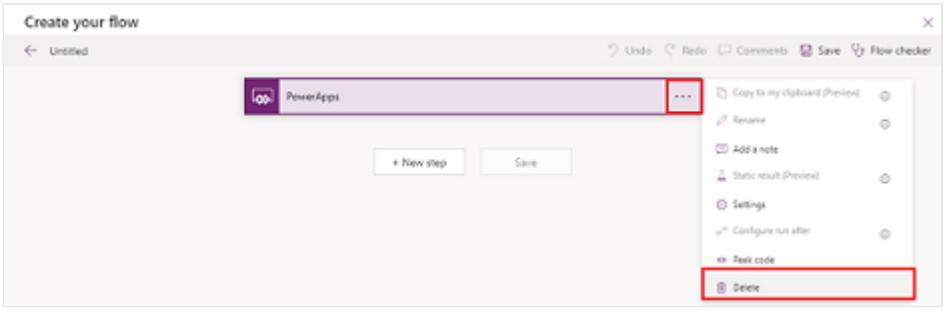
Play the app and select the button you added. The PDF file is generated, and an email is sent to the specified email address with the generated PDF file attached.

Use in a Power Automate flow

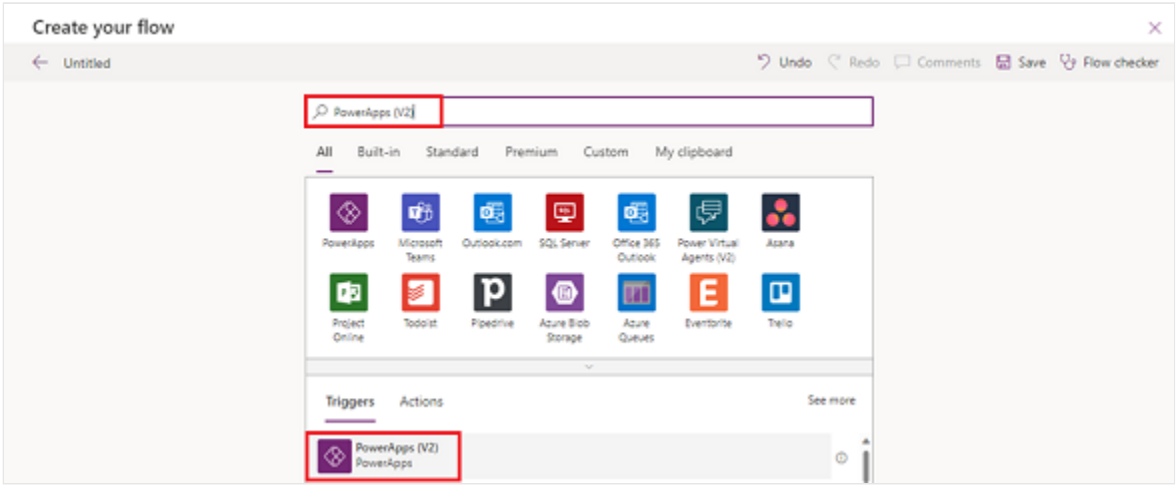
In this example, we generate a PDF file and pass it to a Power Automate flow that stores the file in a document library using the SharePoint connector. This example replaces the default PowerApps trigger in Power Automate with the PowerApps (V2) trigger.

Create a flow to use in your app

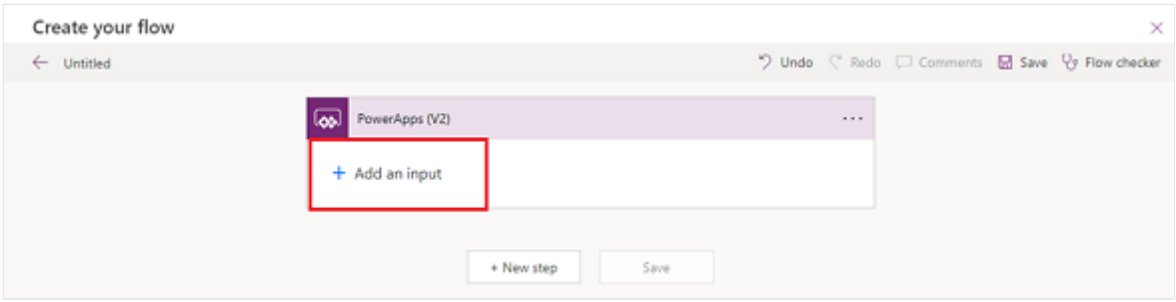
1. In your canvas app, select the button that you created in the previous example or add a new button.
2. In the app authoring menu, select **Power Automate** > **Create new flow**.
3. Select **Create from blank**.
4. To delete the default PowerApps trigger, select the trigger menu (**...**), and then select **Delete**.



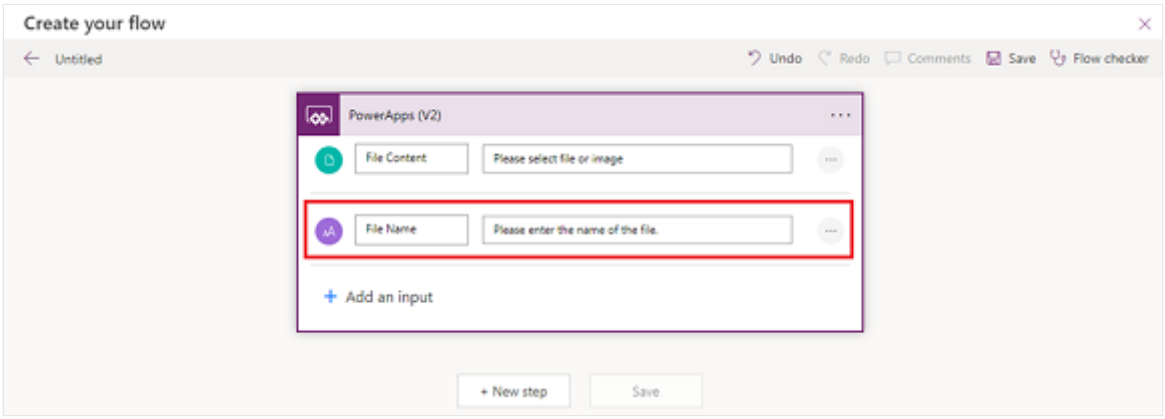
5. Search for and select **PowerApps (V2)**.



6. Select the **PowerApps V2** trigger to expand the node, and then select **Add an input**.

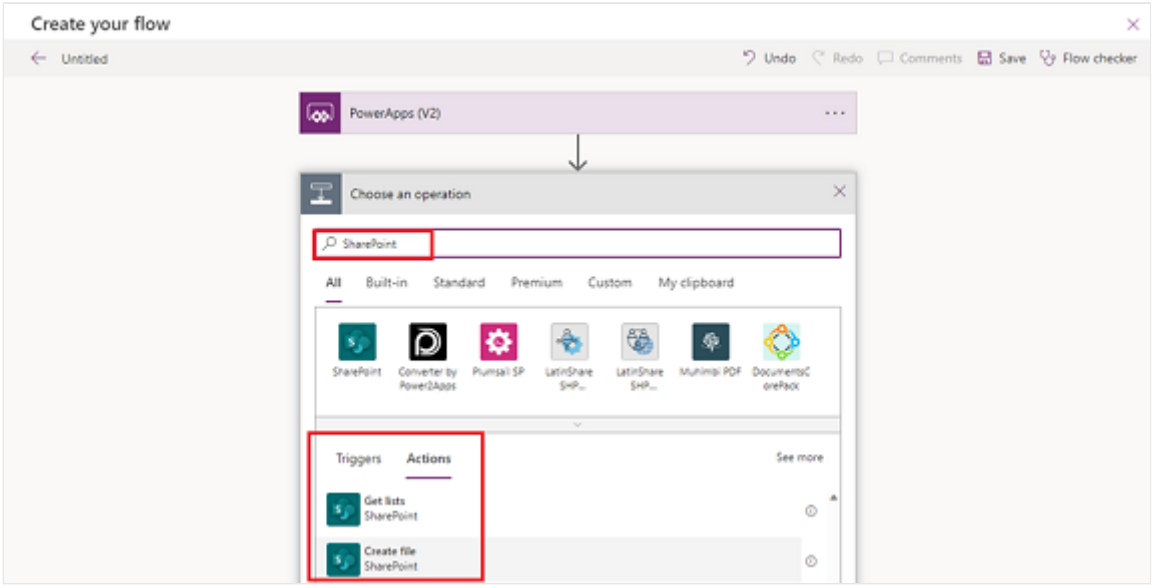


7. Select **File** as the type of user input.
8. Select **Add an input** again, and then select **Text** as the type of user input.
9. Rename the input **File Name** and change the description to **Please enter the name of the file**.



Add the SharePoint connector to the flow

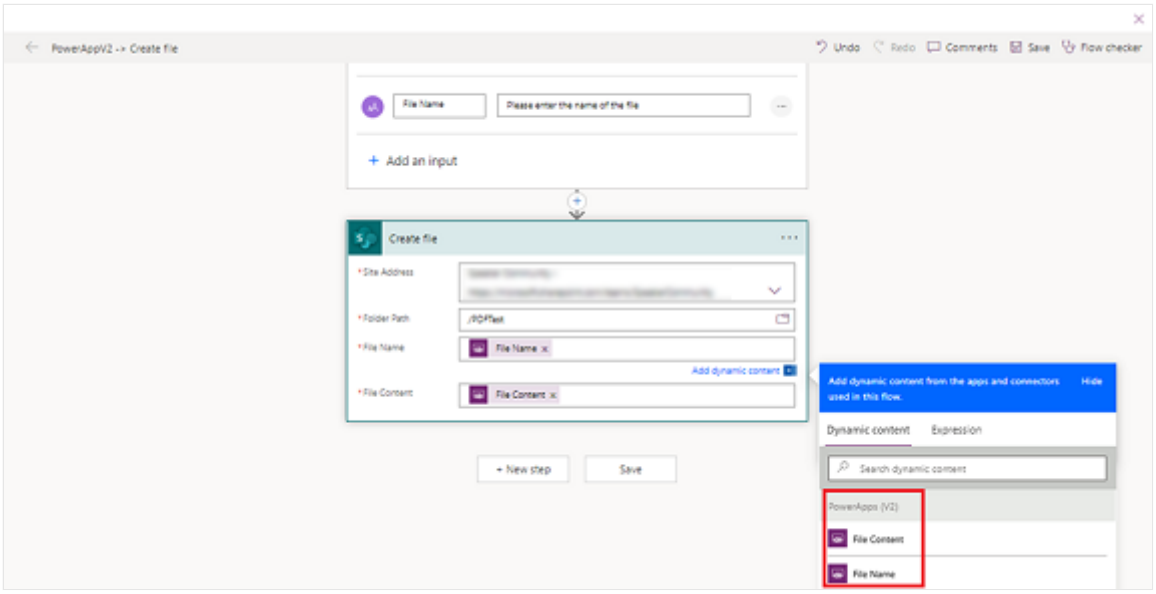
1. Select **New step** to add an action. Search for **SharePoint** and select **Create file**.



2. In **Site Address**, select a SharePoint site that you have permission to add files to.
3. In **Folder Path**, select a document library that you have permission to add files to.

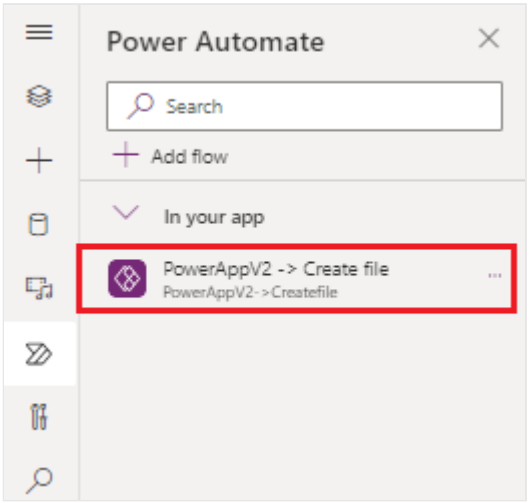
4. Select the **File Name** input. In the **Dynamic content** tab, select a file name under **PowerApps (V2)**.

5. Select the **File Content** input, and then select **File Content** in the list.



6. Select **Save**, and then select the **x** in the upper-right corner to close the Power Automate window.

7. The Power Automate pane refreshes, and the new PowerApps (V2) flow, **Create file**, appears in the list.

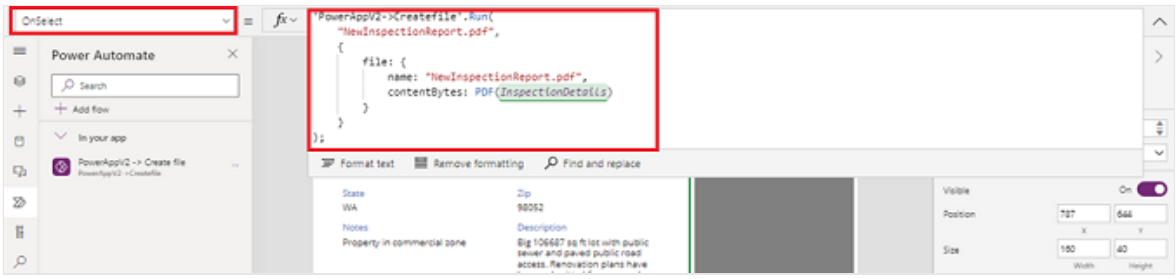


Add the flow to a button in your app

1. Select the button in your app canvas. In the button's `OnSelect` property, enter the following formula, replacing the flow name and generated PDF file name with your own:

```
```

```
'PowerAppV2->Createfile'.Run(
   "NewInspectionReport.pdf",
    {
       file: {
            name: "NewInspectionReport.pdf",
            contentBytes: PDF(InspectionDetails)
        }
    }
);
```



2. Select **Save and publish**.

Play the app and select the button you added. The Power Automate flow runs and adds the generated PDF file into the document library.

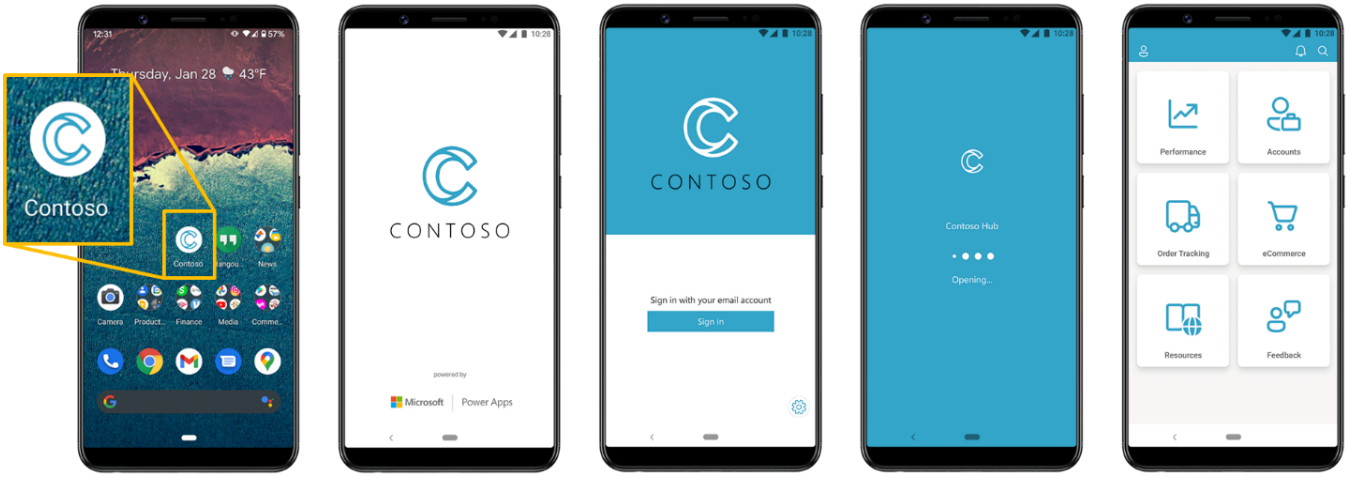
See also

PDF function in Power Apps (experimental)

Overview of wrap

06/16/2025

The **wrap** feature in Power Apps lets you package your canvas app as a custom-branded Android or iOS app for native distribution to mobile users. Distribute these wrapped native mobile apps to users through the Google Play Store, Apple Business Manager, or Microsoft Intune.



Key capabilities

With wrap, you can:

- Package one or more Power Apps canvas apps in the same native mobile app package.
- Customize your mobile app startup experience to match your organization's branding.
- Specify the app icon, splash screen image, welcome screen image, and color palette.
- Update wrapped mobile apps by publishing changes to the included canvas app through the Power Apps maker portal.
- Distribute apps using MDM.

> **Note**
>
> All published changes to the included canvas app(s) are downloaded automatically by existing, released versions of your wrapped mobile apps.

Benefits of wrap

Wrap brings native mobile application development platform (MADP) capabilities to Power Apps:

Expand table

Benefit	Description
No-code mobile app development	Create mobile apps with no previous experience
Managed mobile app builds	We generate the app for you
Seamless end-to-end branding	Use your own logo and color palette
Multiple canvas apps support	Bundle multiple apps in a single mobile app
Enterprise governance with Microsoft Intune	Protect your data with app management

Software and device requirements

Platforms supported

- **iOS**: Version 14.0 or later.
- **Android**: Version 8.0 (API level 26) or later.

Device requirements

- **iOS**: iPhone 6S or later, iPad 5th generation or later.
- **Android**: Any device that runs Android 8.0 or later.

Developer requirements

- Microsoft Power Apps account with appropriate licenses.
- Access to Microsoft Entra ID (formerly Azure AD) for app registration.
- Azure subscription (for Azure Key Vault and Blob Storage).
- For manual code signing:
  - iOS: macOS device with Xcode installed.
  - Android: Windows PC with Android Studio installed.

Wrap process overview

The **wrap** feature packages your canvas app in a native mobile app shell and produces a mobile package. You can digitally sign and distribute this mobile package as your custom-branded Android and iOS app through native distribution channels, like Google Play Store and Apple Business Manager.

Step-by-step process

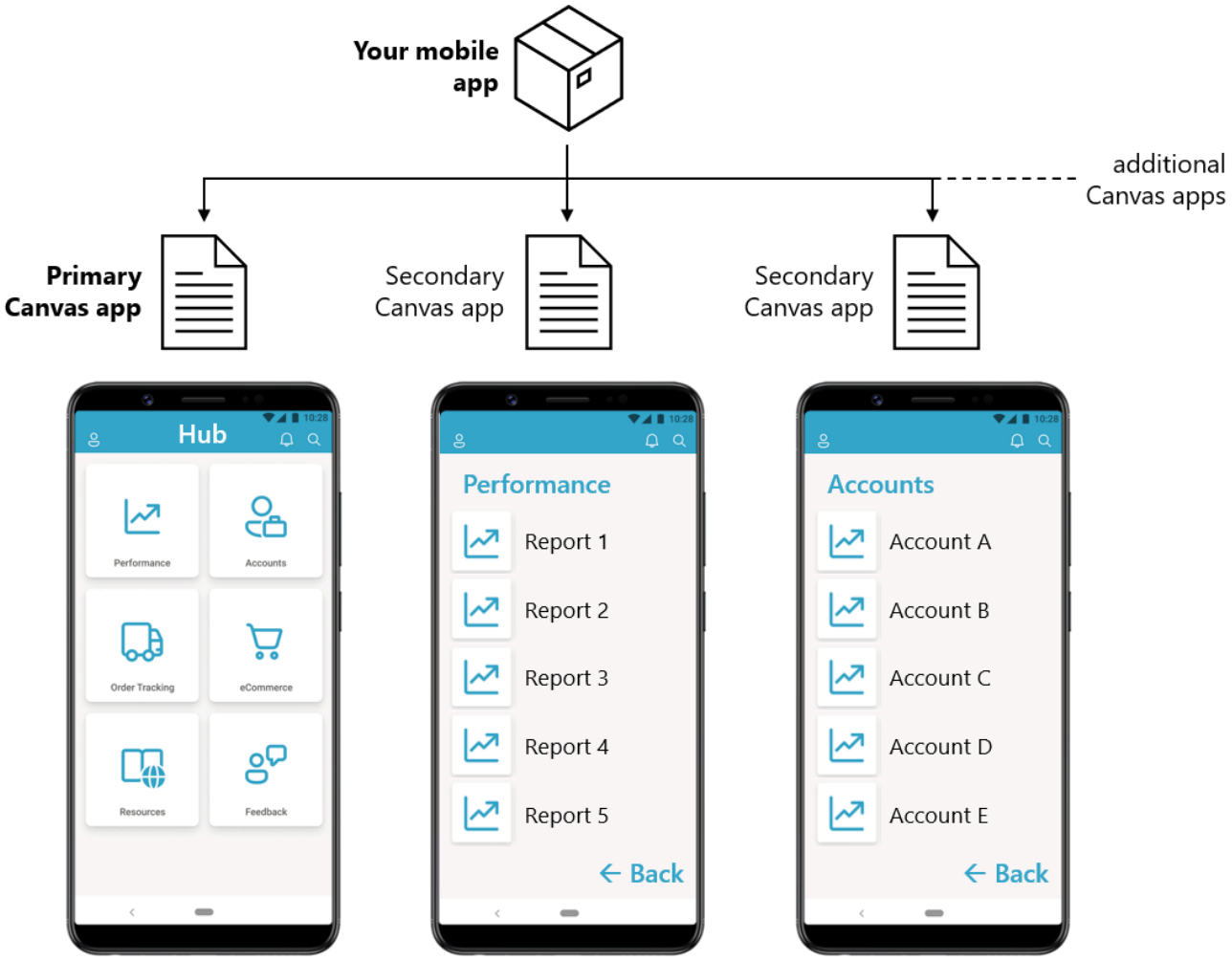
1. Select your **primary canvas app** and start the wrap wizard. A primary canvas app provides the initial experience users see when launching your mobile app. Your canvas apps must be part of a solution. More information: Add canvas app to solution.
2. Optionally, add **secondary canvas apps** to your mobile app in the wrap wizard. More information: Wrapping multiple canvas apps together.
3. Select the **target platforms** (iOS and Android) for your mobile app. Optionally, select **automatically code sign** your mobile app package.
4. Register your app. Use an existing **app registration** or create a new one in the wrap wizard.
5. Customize **app branding** with icons, images, and a color palette to personalize your mobile app.
6. Add the **Azure blob storage account name and container name**. Use an existing **Azure blob storage** or create a new one.
7. Start the build process in the **Wrap up** step to generate your custom branded mobile app.
8. Download your mobile app from the **App blob storage location**.
9. If you don't select **automatically code sign** in the wrap wizard, code sign the mobile app package manually. More about signing your mobile app package manually: Signing your mobile app package manually.
10. Test the app package.
11. Distribute the app package to mobile users.

Wrap multiple canvas apps together

You can wrap more than one canvas app into a single mobile app package. The mobile app package needs a home app, called the primary app. This app becomes the entry point for all other canvas apps included in the mobile app package, which are called secondary apps.

Secondary apps are optional. When you wrap only one canvas app, that app is considered the primary app, and the mobile app package has no secondary apps.

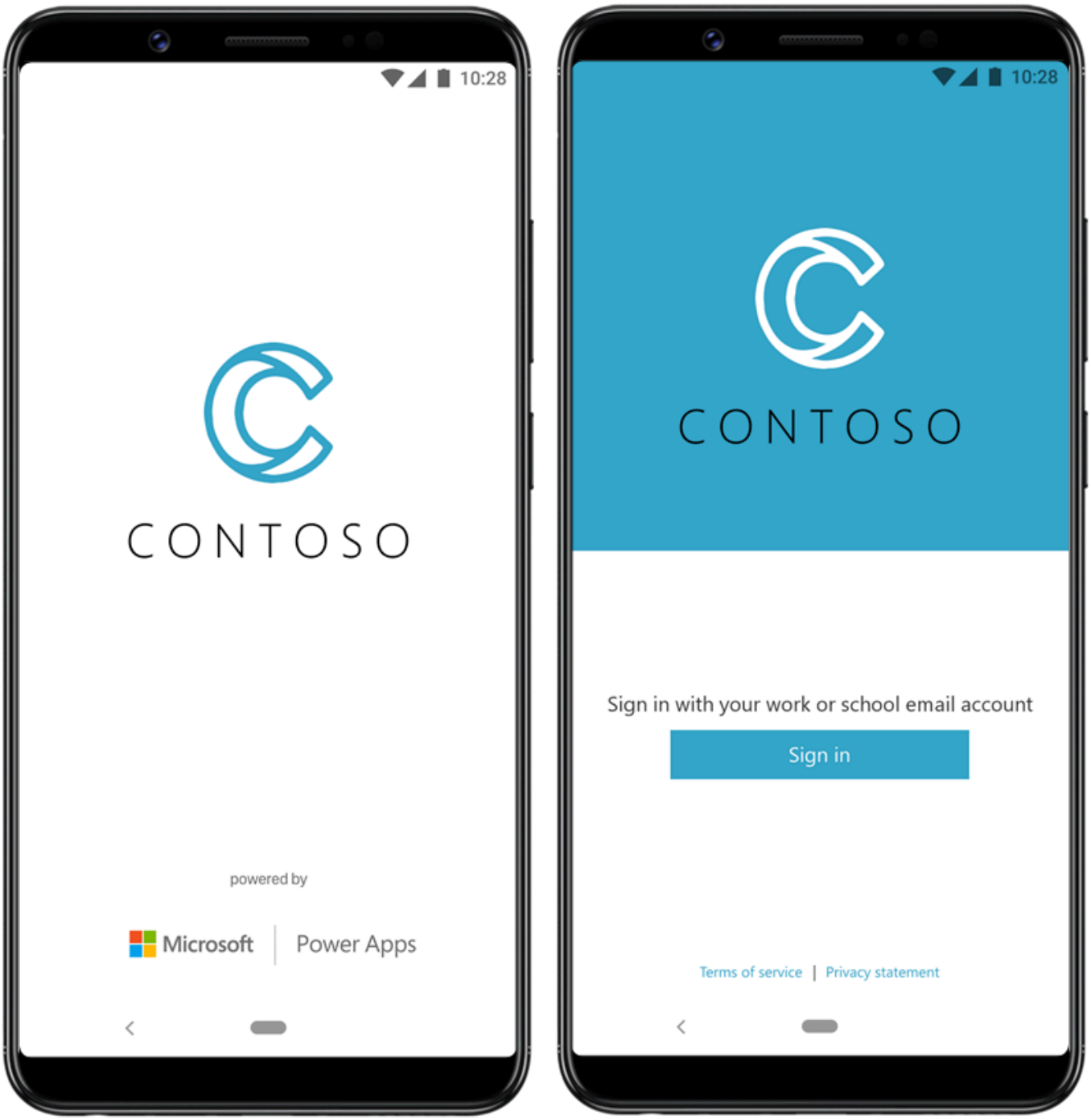
As shown in the illustration below, a primary app can have links to multiple secondary apps. You can manage navigation between primary and secondary apps using the Launch() function.



Brand your mobile app

Wrap supports customization of the mobile app bootstrap experience to match your organization's branding requirements. You can specify:

- App icon
- Splash screen image
- Welcome (sign in) screen image
- Color palette for native experiences



Branding customization options are available when building your wrap project. More information: Configure branding

---

Wrap terminology

Wrap involves multiple components across Power Apps and third-party platforms such as iOS and Android. Understanding these components is important when working with the wrap functionality.

App platform(s)

Target platforms for your app during the build process. You can create builds for:

- **iOS** — creates IPA package
- **Android** — creates APK package
- **Google Play Store** — creates AAB package for distribution

Bundle ID

The bundle ID is a unique identifier for your app that follows a reverse domain name pattern. It must contain one period (.) and no spaces. Example: `com.contoso.myapp`.

This bundle ID is used when creating the Azure key vault for wrap after iOS or Android certificates are created and uploaded. If you've already created the Azure key vault, verify the bundle ID in the **Tags** section of the Azure portal. Use this same bundle ID in Step 2: Target platform.

Code signing

Code signing completes a mobile app before distribution to end users. A code-signed app assures users it comes from a known source and the app code hasn't changed since it was last signed by the trusted source.

Primary app

A primary app is the entry point or home app for the mobile app experience when wrapping multiple canvas apps together. If only one canvas app is wrapped, it's considered the primary app.

Secondary app

Secondary apps are optional canvas apps that you wrap in the same build for mobile app distribution along with the primary app.

Redirect URI

A redirect URI (reply URL) is the location where the authorization server sends the user after successful app authorization and access token grant. The authorization server sends the code or token to the redirect URI, so registering the correct location during app registration is important. More information: Redirect URI

Common issues and limitations

Known limitations

- The wrap feature only supports canvas apps (not model-driven apps)
- All canvas apps in a wrap project must be from the same environment
- Users must have a Power Apps license to use wrapped apps

Common issues

- **Bundle ID conflicts**: Ensure your bundle ID is unique across your organization
- **Image format issues**: All images must be in PNG format
- **Signing certificate problems**: Verify certificate validity and expiration dates
- **Azure Key Vault access**: Make sure proper permissions are configured

For troubleshooting details, see Troubleshoot issues with the wrap feature in Power Apps.

Next steps

System requirements and prerequisites for Wrap

See also

- Use the wrap wizard to build your mobile app
- Manual code sign on iOS
- Manual code sign on Android
- Code sign for Google Play Store
- Create your Azure Key Vault for automated code signing
- Frequently Asked Questions
- Troubleshoot issues with the wrap feature in Power Apps
- Benefits and limitations of Wrap

System requirements and prerequisites for Wrap

07/09/2025

Before you use Wrap to create native mobile apps from your Power Apps canvas apps, check that your system meets these requirements.

Software and device requirements

- You need a Mac device for manual code signing with iOS.
- You need a Windows PC for manual code signing with Android.
- To run the wrapped mobile app:
  - Android device with version 10 or higher
  - iOS device with version 14 or higher

> **Note**
>
> You need an Apple Developer Program account to develop apps for iOS.

Permissions and access requirements

1. You need access to:

   - One or more canvas apps to build the wrap project.
   - Azure blob storage, including the account name and container name. More information: Create an Azure storage account.
   - The Azure portal to create app registration and register your app, which should be in the same environment as your Power Apps.

2. An Azure admin must grant you access to use the specific wrap app. More information: API permissions.

3. Your apps must be part of a managed or unmanaged solution. If not, add them to an existing or new solution. More information: Create a canvas app from within a solution.

4. You need an Azure key vault to perform the automatic sign-in process through Wrap wizard. More information: Create a key vault using the Azure portal.

5. Check the policies enabled for your application. More information: Conditional Access policy templates

6. If multifactor authentication (MFA) is enabled, make sure MFA is enabled for the accounts you'll use to sign in, or disable the conditional access policies.

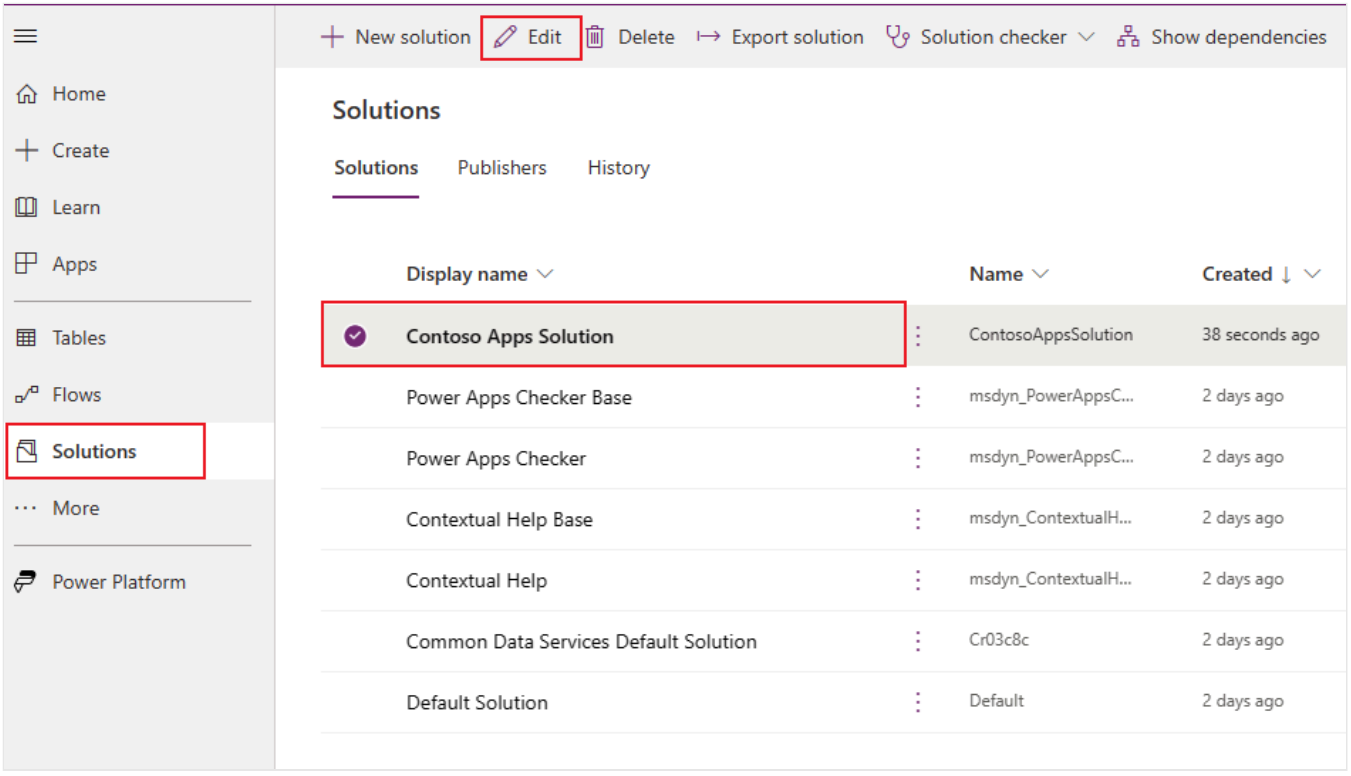
Sign-in options in Wrap

- **Manual code sign-in for Android and iOS**: This option is best for most scenarios. For Android, generate keys and generate a signature hash before you start. You need the signature hash to set up the Redirect URI. More information: manual code sign-in for Android.
- **Automatic sign-in using Wrap wizard**: This option requires an Azure key vault. More information: creating a key vault using the Azure portal.

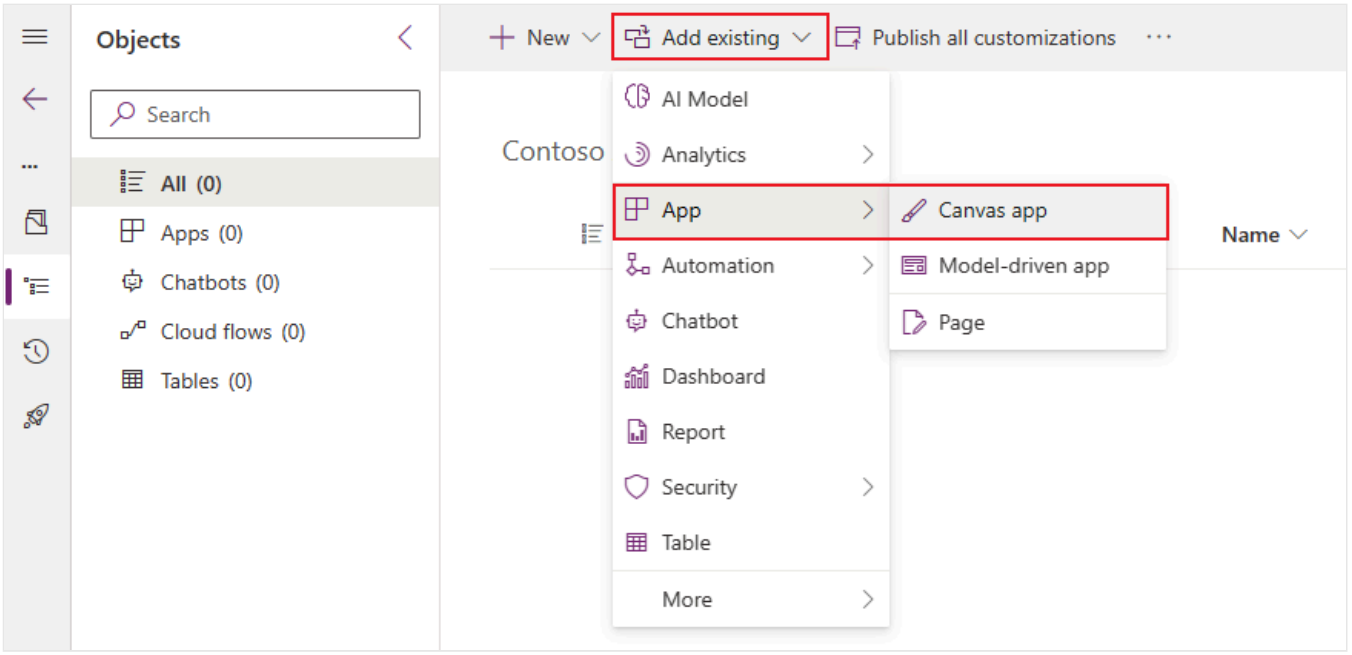
Add canvas app to a solution

Wrap requires your apps to be part of a solution. If your canvas apps aren't already in a solution, add them to an existing or new solution.

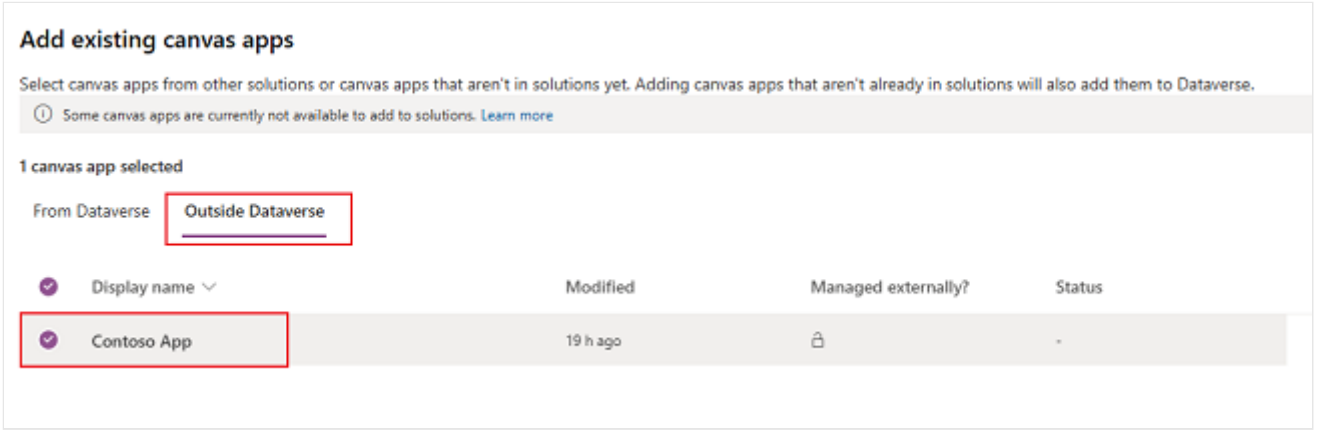
1. In the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select a solution, and then select **Edit**.



Select + **Add existing** from the top menu, and then select **App** > **Canvas app** from the dropdown.



Select the **Outside Dataverse** tab, select your app from the list, and then select **Add** to add the app to the solution.



More information: Add an app to a solution

Next steps

Use the wrap wizard to build your mobile app

See also

- Troubleshoot issues with the wrap feature in Power Apps
- Manual code sign on iOS
- Manual code sign on Android
- Code sign for Google Play Store
- Create your Azure Key Vault for automated code signing
- Frequently Asked Questions

Wrap wizard: Step-by-step guide to building your native mobile app

Use this guide to convert one or more canvas apps into a single custom-branded app package that you can deploy on Google Play and the iOS App Store.

The wrap feature in Power Apps enables you to create native mobile versions of your canvas apps as custom-branded Android and iOS mobile apps (IPA, APK, AAB packages). You can distribute these wrapped native mobile apps to end users through Google Play, Apple Business Manager, or other native distribution methods.

When you update and republish your app, the wrapped app is automatically updated for users.

Steps to create a custom-branded native app using the wrap wizard

Follow these steps to configure, build, sign, and distribute a custom-branded native mobile app package (APK, AAB, or IPA) from your canvas apps using the wrap wizard in Power Apps.

1. Sign in and start a wrap project

1. Sign in to Power Apps maker portal.
2. Select **Wrap** from the left navigation pane. If the item isn't in the side panel pane, select ... More and then select the item you want.
3. Select the app you want to wrap, then select **Wrap** on the command bar.

2. Select apps

1. On the **Select the app(s) to wrap** screen, select your primary and secondary app.

    - **Primary app**: The main app end users see when the mobile app launches.
    - **Secondary app(s)**: Optional other apps bundled in the same mobile app package.

    > **Note**
    >
    > - You can use the same primary app in multiple wrap projects.

- If the primary app name appears incorrect, proceed to the next step and return to refresh the name.

2. Select **Next**.

3. Choose target platform

1. On the **Choose mobile platform to target** screen, enter a **Bundle ID**.

> ⓘ Note
>
> The **Bundle ID** is a unique identifier for your app. It must contain one period (.) and no spaces. Use this same bundle ID when creating the Azure key vault after generating and uploading your iOS or Android certificates. If you have already created the Azure Key Vault, verify the bundle ID in the **Tags** section of the Azure portal.

2. Under **Target platform(s)**, select all the mobile platforms that your end users use on their mobile devices.

3. You need to have Azure key vault, to upload your build to Azure blob storage. If you haven't already created, create one. Create an Azure key vault right within the Wrap wizard, or follow the instructions manually.

4. Turn on the **Autosigning** toggle to automate the signing process.

> ⓘ Note
>
> Manual signing options:
>
> - Code sign for iOS
> - Code sign for Android
> - Code sign for Google Play Store You must manually sign AAB files regardless of the signing option selected in the wizard.

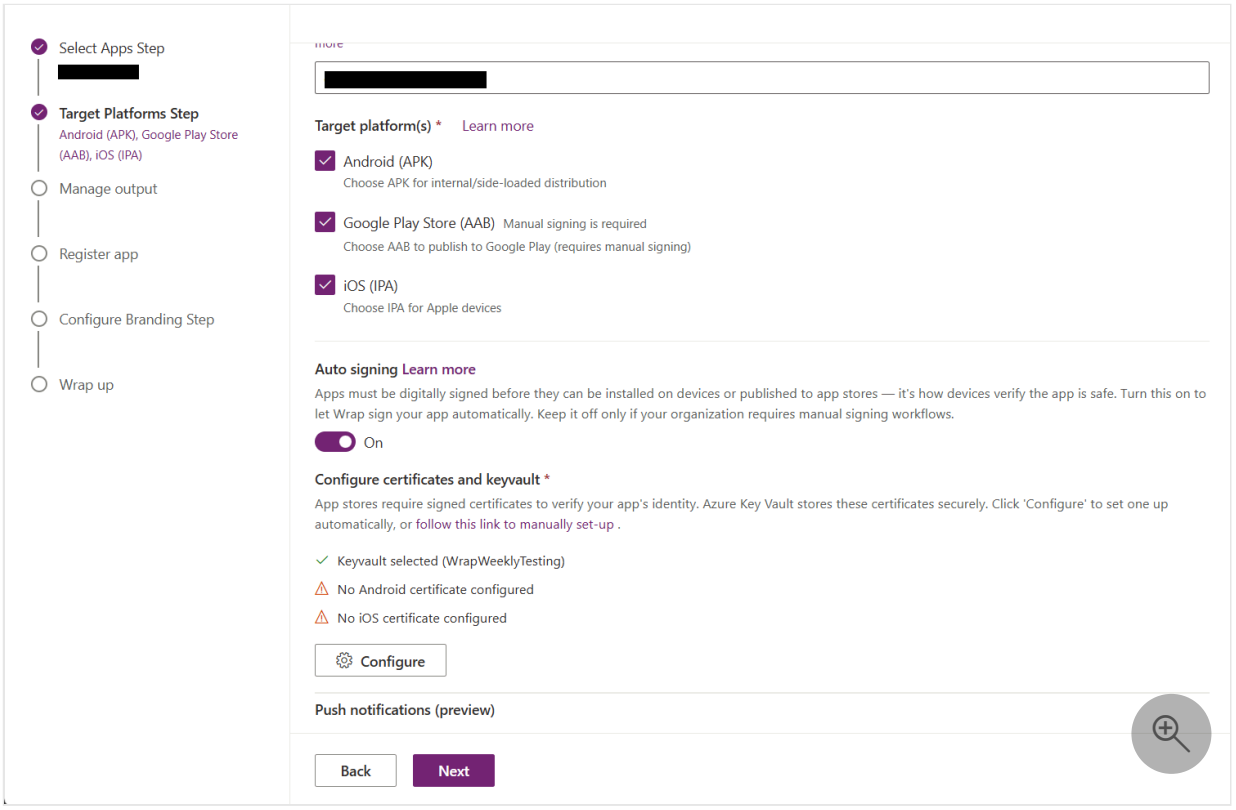
5. Set up autosigning through the wizard.

   a. In the target platform step, toggle on **Autosigning**. The service checks if key vault and certificate are already attached.

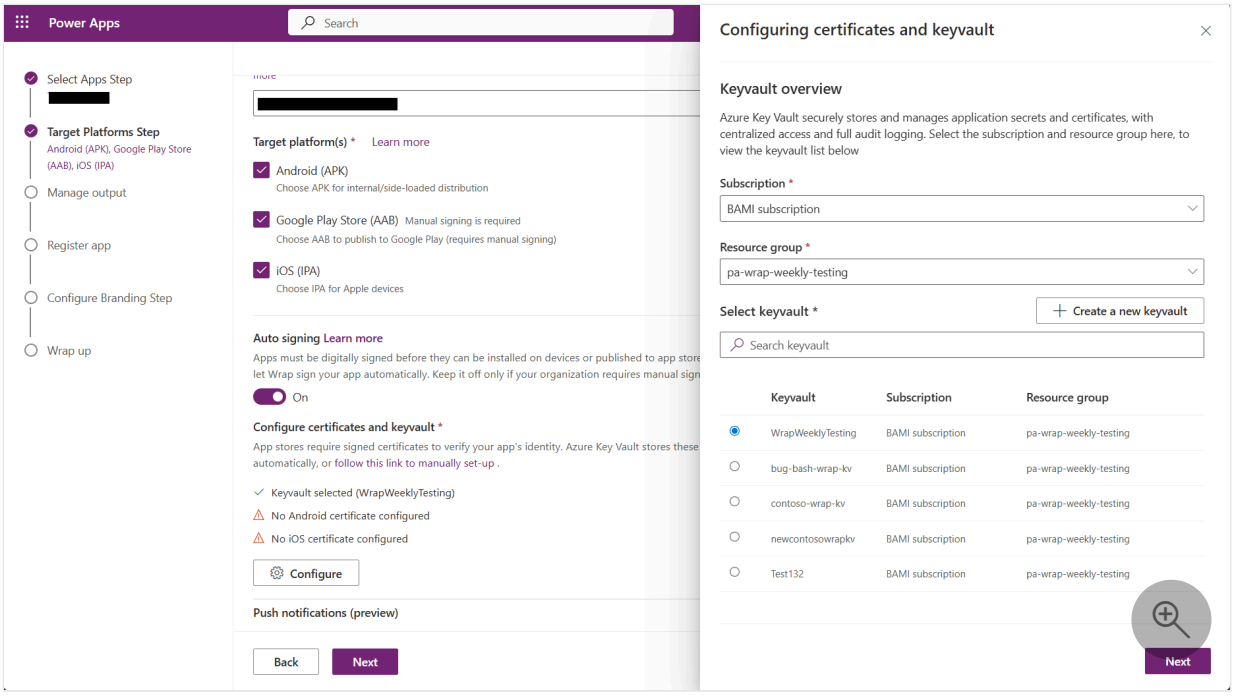
> **Note**
>
> Azure Key Vault is used to securely store and manage application secrets and certificates, with centralized access and full audit logging.

b. If you don't have key vault and certificate or want to change them, select **Configure**.

> **Note**
>
> Only Azure admins can create a key vault from the wrap wizard. Other users need to follow the manual key vault creation process.



c. On the side panel, select the subscription and resource group, and choose the key vault. To create a new key vault, select **Create a new keyvault**. After you select the key vault, select **Next**.

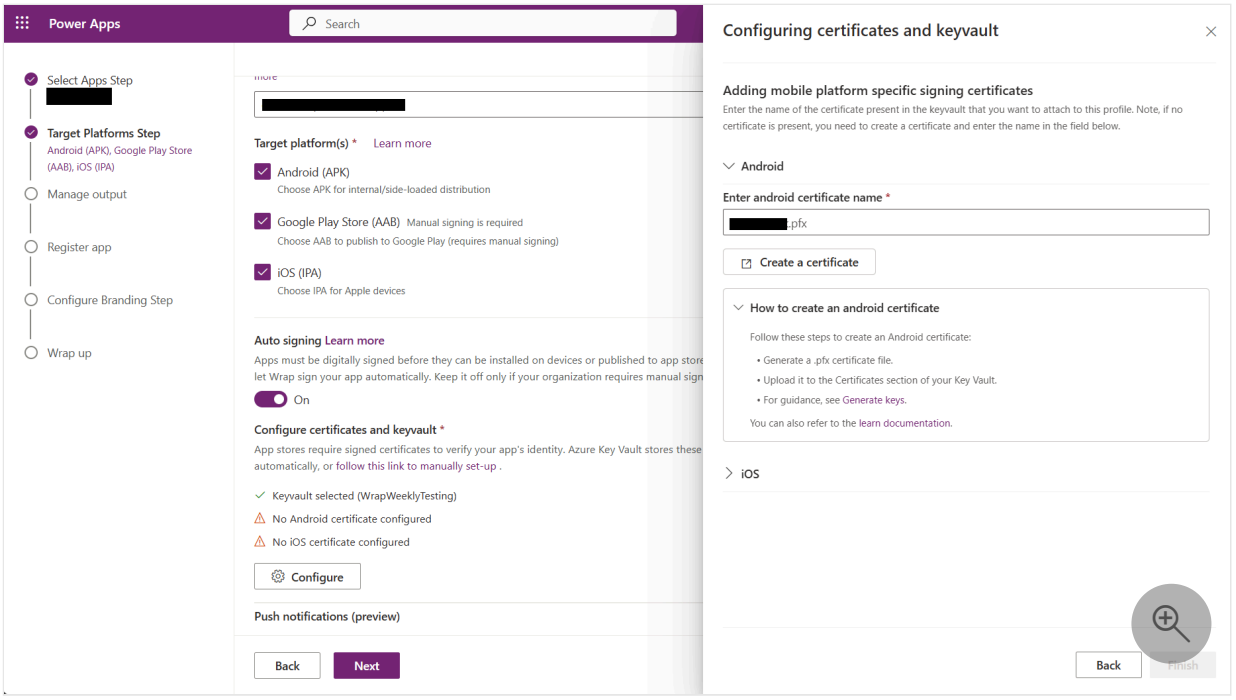


d. The service checks if the appropriate service principal, reader role, access policies, and environment variables are set up for the key vault. If errors appear, fix them. After all checks pass, select **Next**.

e. On the next screen, enter the name of the certificate in the key vault.

f. If no certificate exists or you want to change the certificate, see Generate key and signature hash to create a new certificate. Then select **Create a certificate** on this screen. This action takes you to the Azure portal key vault location. Upload the generated certificate into the configured key vault.

> **Note**
>
> Users are responsible for creating the certificate.

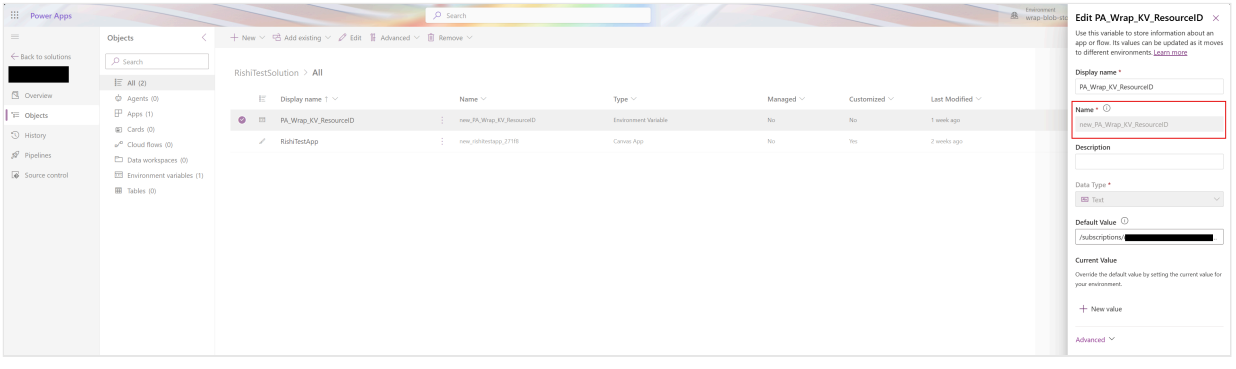


g. Select **Finish**.

h. On the main screen, the Azure Key Vault and certificate appear as confirmed.

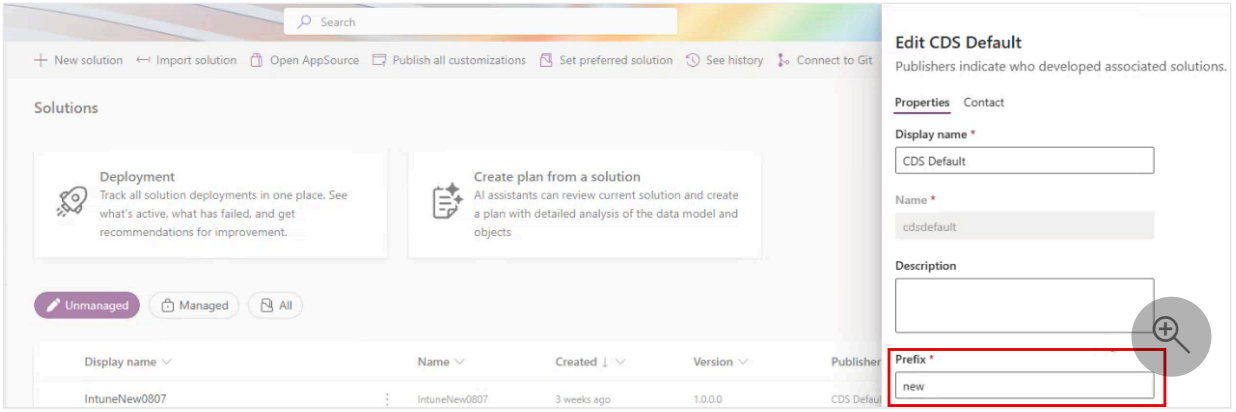
i. On the main screen, select **Next**.

6. Set up autosigning from within Azure portal.
   a. Refer to instructions in create an Azure key vault to create an Azure key vault. Once the vault is created, you need to add the required tags, secrets, and certificates. Add the environment variable if not created already.
   b. To create the environment variable, go to Power Apps > **Solutions** > **Default solution**. Then select **New** > **More** > **Environment variable**, add the display name as "PA_Wrap_KV_ResourceID".

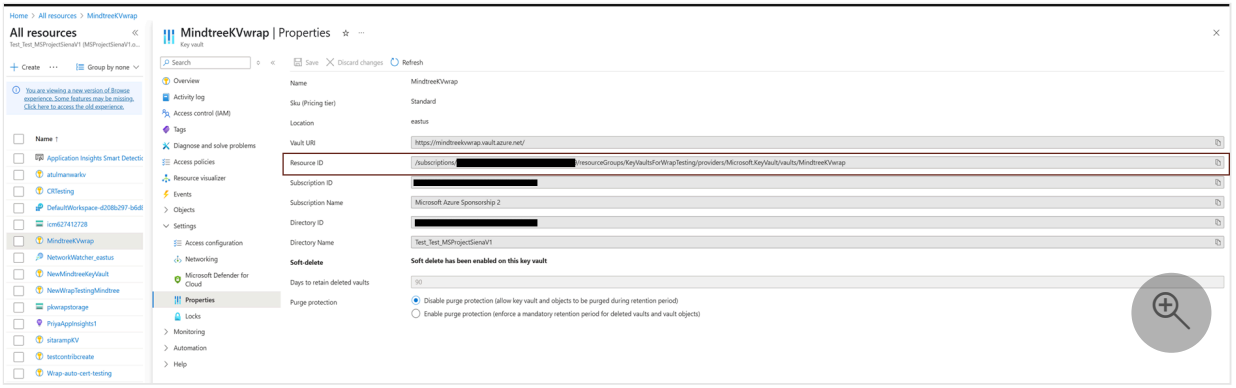


The name of the new environment variable must have the prefix "new." If it doesn't,

see Set Environment Variable Prefix for detailed steps.



c. To add vault information to your environment variables, access the **Azure** portal as an admin. Navigate to **All Resources** > **Your Key Vault** > **Properties**, and then copy the **Resource ID**.



d. To add the input to the environment variable, go to **Power Apps** > **ApplicationName** > **All** > **Environment variable**. Click the three dots, select **Edit**, add the copied value to **Default value**, and save.

e. To check whether the table value has been updated, go to **Power Apps** > **Tables** > **Environment variable definition** > **new_PA_Wrap_KV_ResourceID** . The value in **Default value** must be same as that of the resourceID for which you want to add the vault.

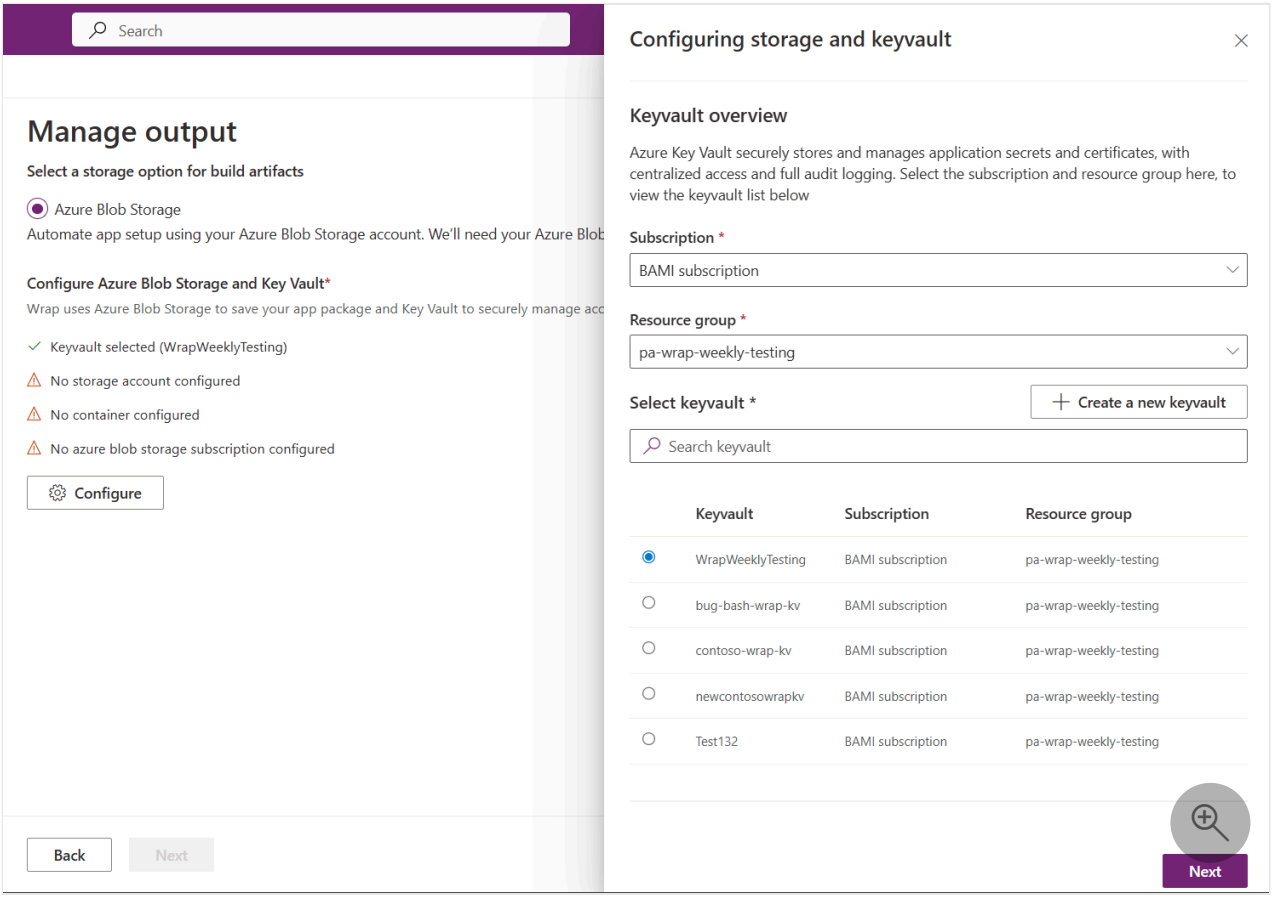
> **Note**
>
> Guidelines for adding the input behind the environment variables for Key vault information:
>
> - Environment variables must not be empty or can contain multiple entries.
> - Ensure that the resourceID added is correct (verify spelling).
> - Ensure that the resourceID added has non-empty tags and includes all the tags expected with the bundle ID used in the wrap wizard.

f. Follow the steps in Steps for automated code signing to create the tags, secrets, and certificates required during the automatic signing process.

4. Manage output

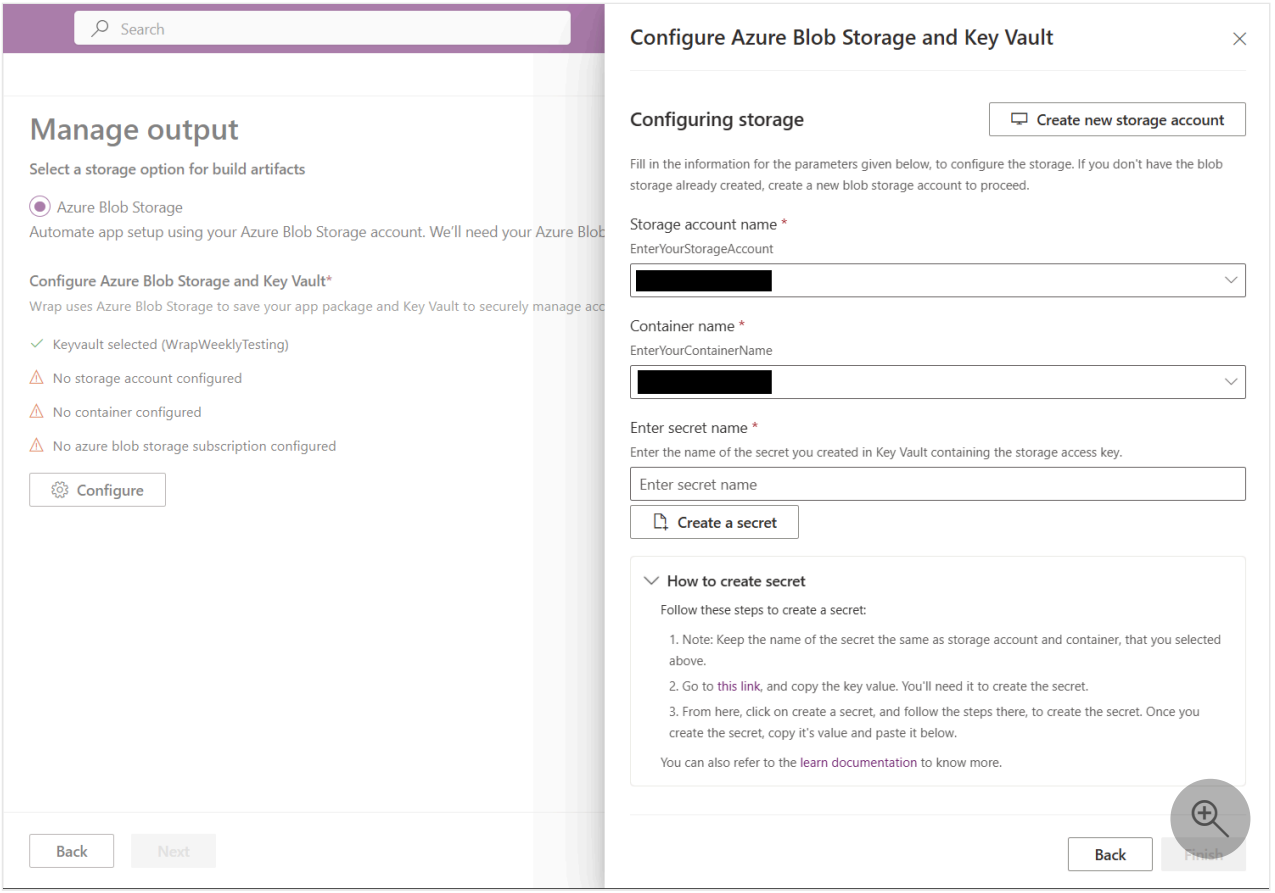
To upload your build to Azure blob storage, you need an Azure blob storage account and container. If you don't have one, create one.

1. On **Manage output**, select **Configure** to add your key vault, storage account, and container information.
2. In the **Configure keyvault** side panel, select the key vault that stores access credentials to your Azure blob storage, and then select **Next**.



3. In the **Configure storage** screen, select the storage name and container name, and then enter the name of the secret that contains access to the storage. To create a new secret,

follow the instructions in the "How to create a secret" section on the same screen.



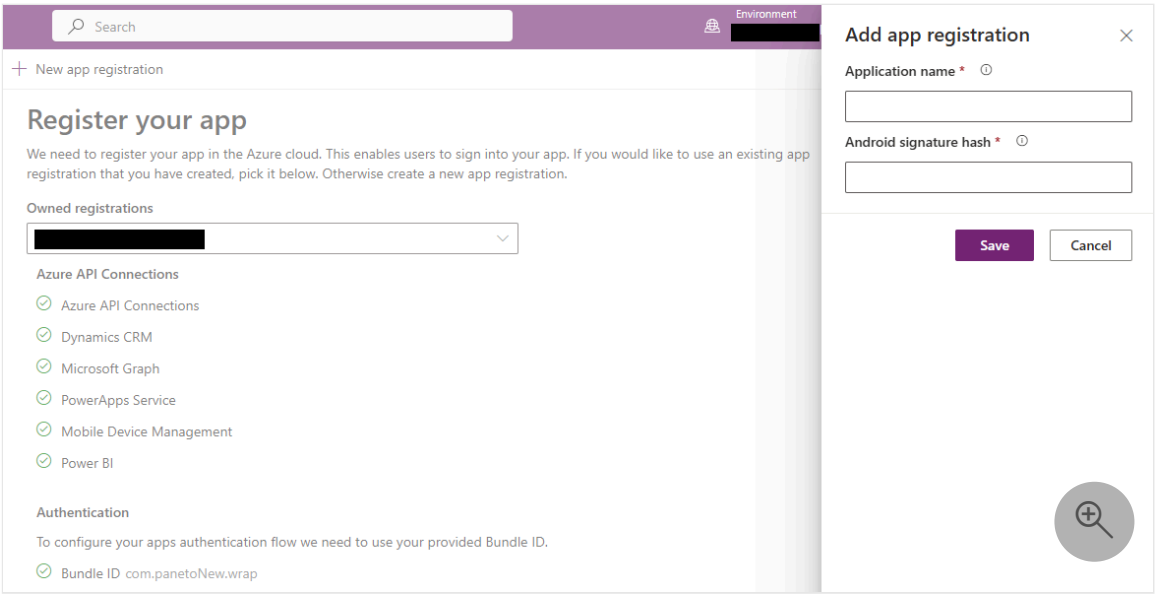
4. Select **Finish**.

5. Register your app

On the **Register your app** screen, register your application in Azure to establish trust with the Microsoft identity platform.

- **If you have already registered an app**:
  - Find your registration in the owned registration field.
- **If you do not see your registered app**:

  1. Select **New app registration** to create a new registration.
  2. Provide:
     - **Application name**: The customer-facing name of your app
     - **Android signature hash** (if targeting Android): A 28-character alphanumeric string



3. Save your changes.

Configure admin allowed third-party apps as an azure tenant admin

The registered app must be configured as a third-party app. To configure the app, the Azure tenant admin follows these steps.

For Windows

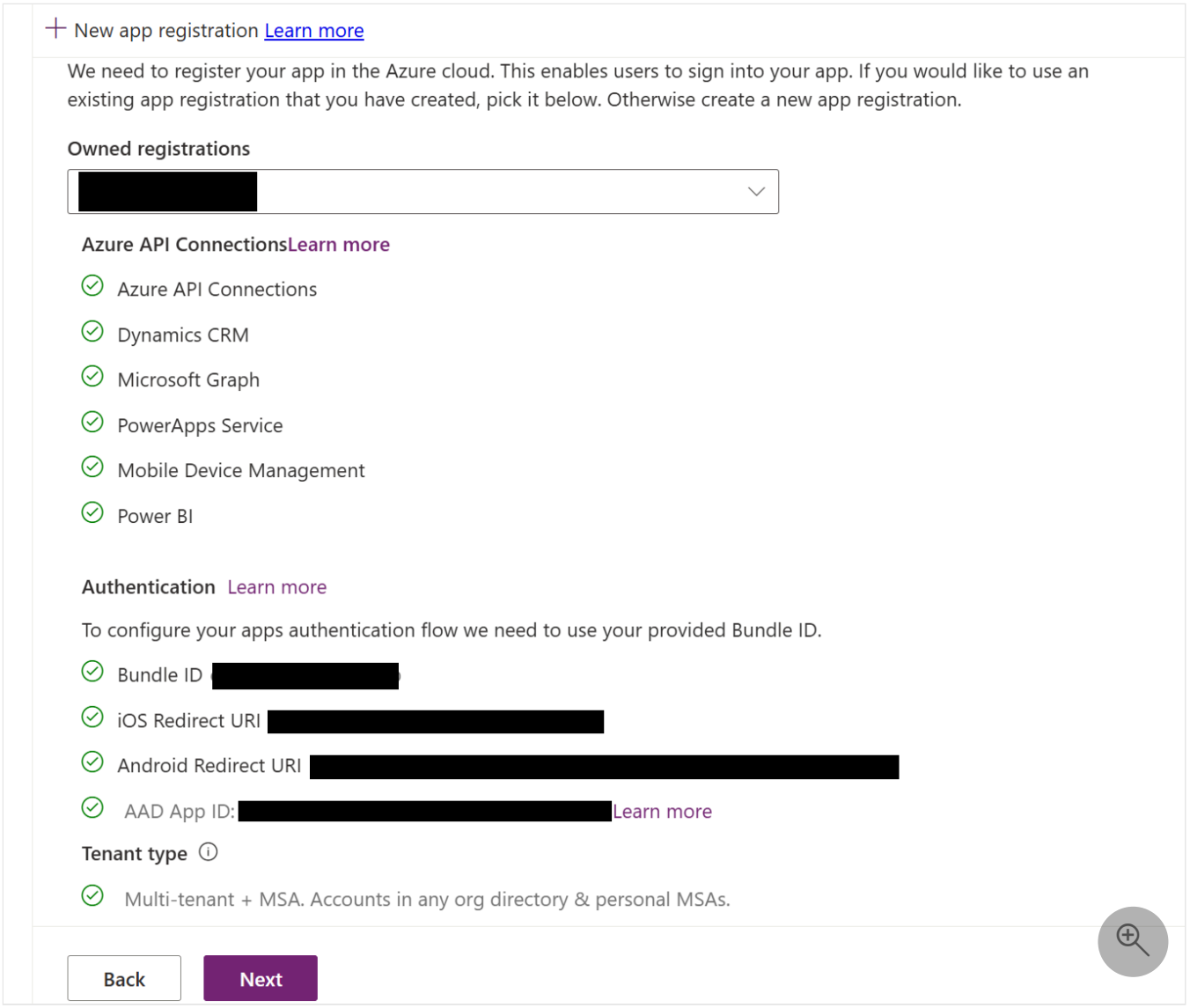
1. Open Windows PowerShell as administrator.
2. Run these commands:

PowerShell

```
Install-Module -Name Microsoft.PowerApps.Administration.PowerShell -
AllowClobber -Force
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
Import-Module -Name Microsoft.PowerApps.Administration.PowerShell
Add-AdminAllowedThirdPartyApps
Get-AdminAllowedThirdPartyApps
```

3. Provide the App ID when prompted.

After completing these steps, the registration screen will look like this:



Grant API permissions as an Azure tenant admin

The app requires the following API permissions to function correctly.

Required API permissions

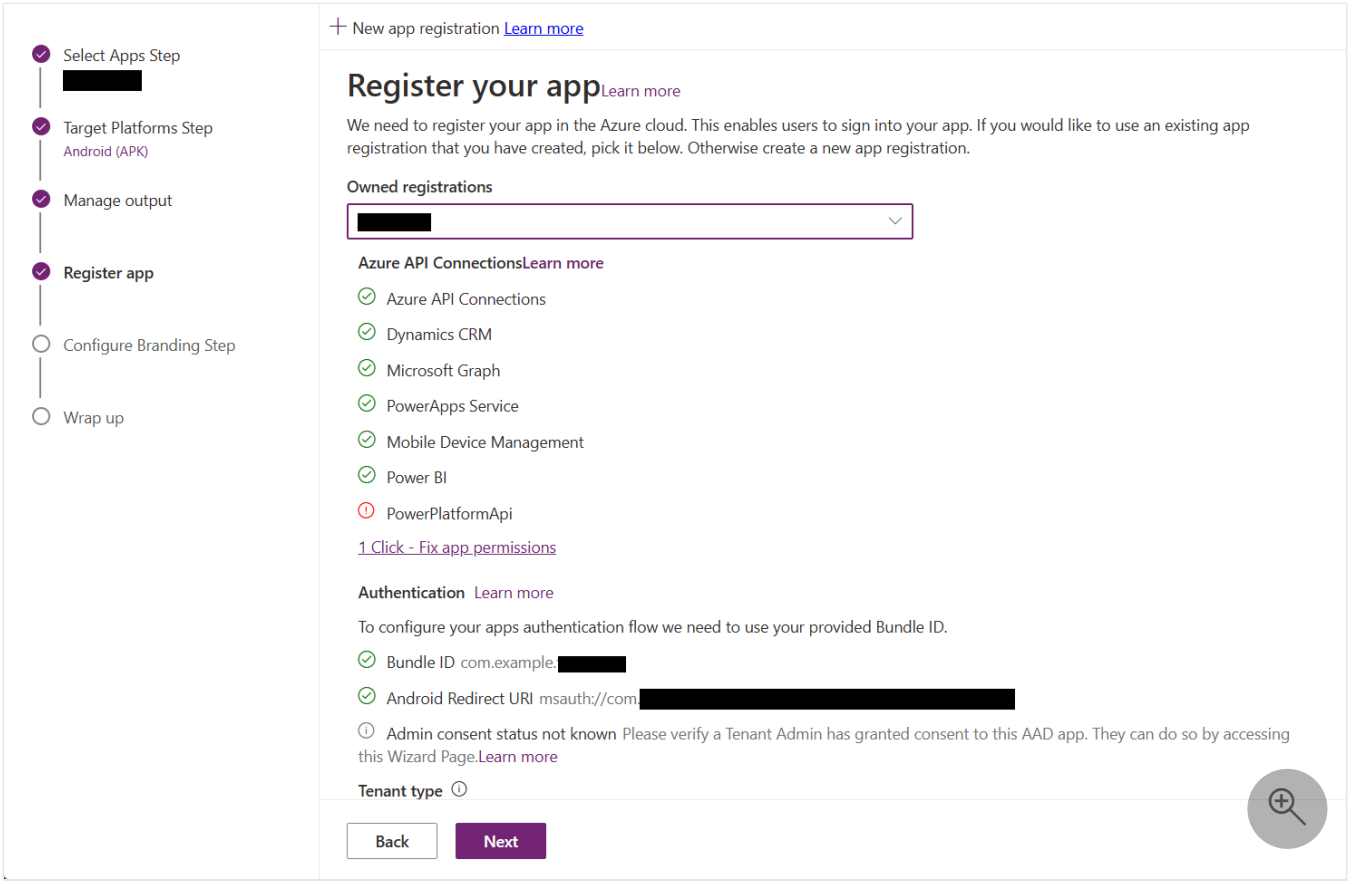
Expand table

API Type	Specific API	Reason
Microsoft APIs	Dynamics CRM	The application needs `user_impersonation` to call Dataverse for the user.
APIs my organization uses	Azure API Connections	The application needs `Runtime.All` to call any connector from Power Platform.
APIs my organization uses	PowerApps Service	The application needs the `User` permission to contact Power Apps backend services from Power Platform.
APIs my	Power BI	The application needs Power BI permissions to access

API Type	Specific API	Reason
organization uses		or embed Power BI content.
APIs my organization uses	Microsoft Mobile Application Management	The application needs this permission because Power Apps uses Intune SDK internally.

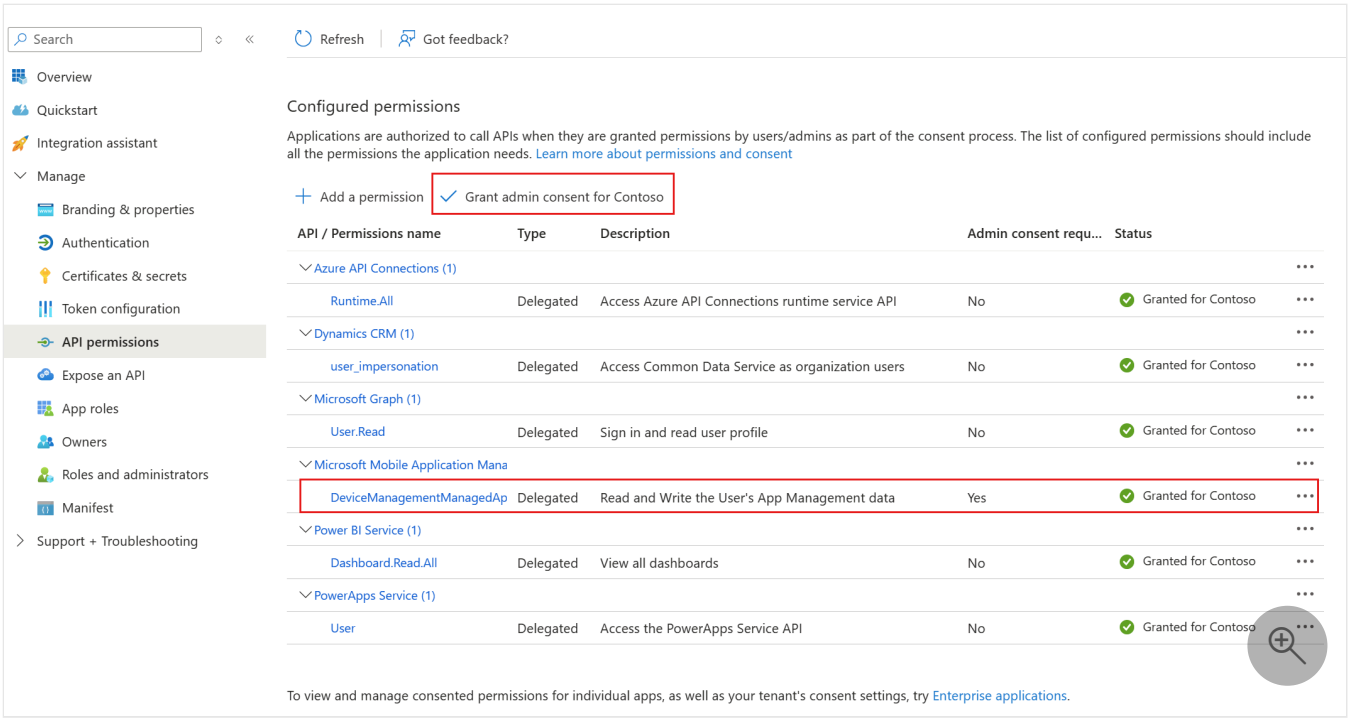
Configure permissions from within Wrap wizard

The wrap wizard checks and flags permissions that aren't available with the app. As an Azure admin, select one click access in the wrap wizard to grant the pending permissions. Alternatively, you can configure permissions through the Powershell commands below.



Configure permissions through Powershell commands:

Make sure **DeviceManagementManagedApplication** is set to **Yes** when you grant admin consent for your app.



Run these PowerShell commands as an Azure admin if you don't see permissions under **APIs my organization uses**:

1. Ensure the module Microsoft Graph is available or install it:

   PowerShell

   ```powershell
   Install-Module -Name Microsoft.Graph
   ```

2. Grant *Azure API Connections* permission for the static AppId fe053c5f-3692-4f14-aef2-ee34fc081cae:

   PowerShell

   ```powershell
   Connect-MgGraph -TenantId <your tenant ID>
   New-MgServicePrincipal -AppId fe053c5f-3692-4f14-aef2-ee34fc081cae -
   DisplayName "Azure API Connections"
   ```

3. Grant *PowerApps Service* permission for the static AppId 475226c6-020e-4fb2-8a90-7a972cbfc1d4:

   PowerShell

   ```powershell
   Connect-MgGraph -TenantId <your tenant ID>
   New-MgServicePrincipal -AppId 475226c6-020e-4fb2-8a90-7a972cbfc1d4 -
   DisplayName "PowerApps Service"
   ```

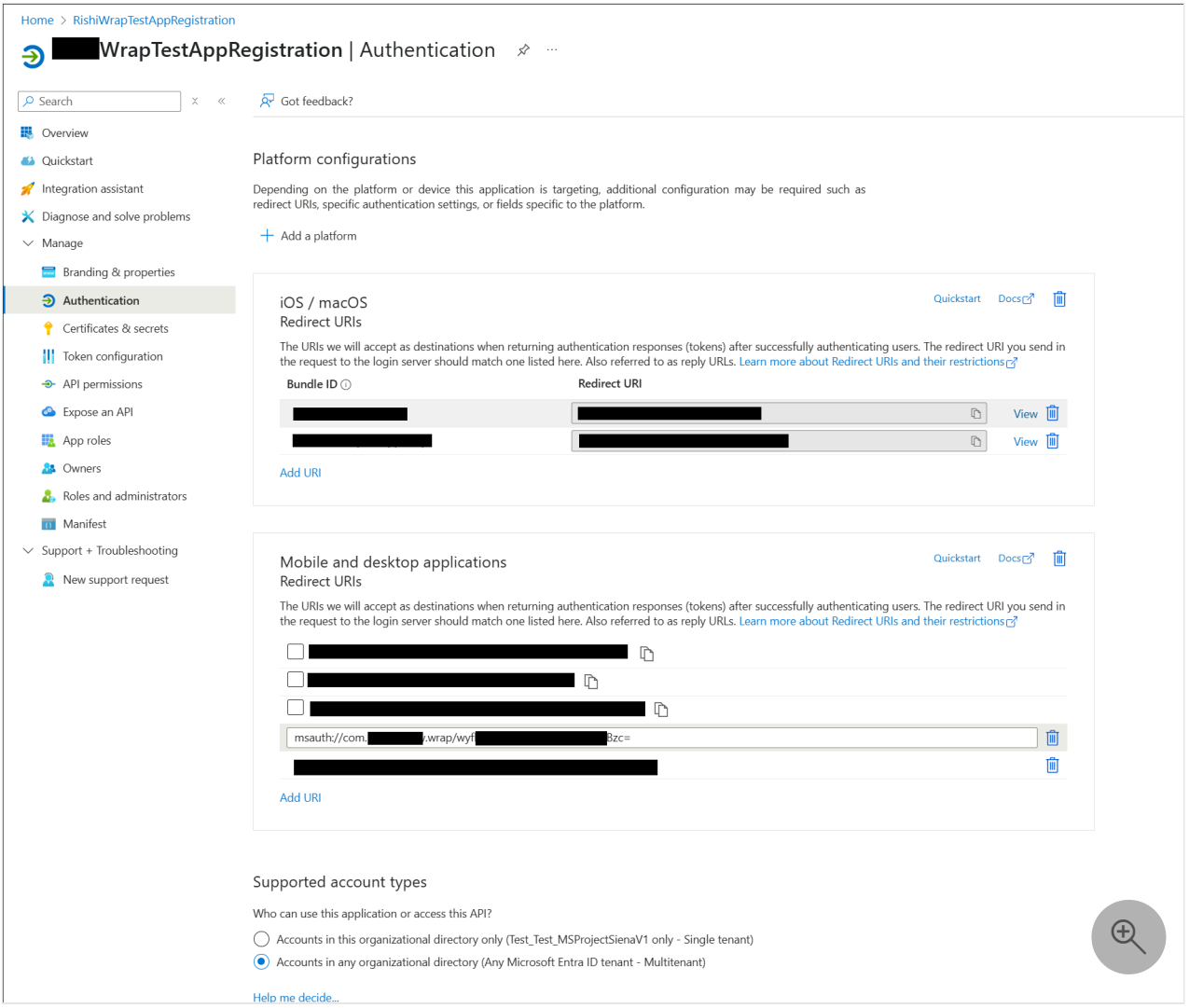
> **Note**
>
> If only the **Application name** field is visible, continue to the next steps and select **Android** as a target platform to display the signature hash field.

Configure API permissions manually (optional)

If you get errors, manually configure API permissions. For more information, see Add and configure.

Add Redirect URIs as an app admin

1. In the Azure Portal, go to your app registration > **Authentication**.
2. Select **Add a platform** and choose **iOS** or **Android**.
3. For iOS, enter the **Bundle ID**. For Android, enter both the **Bundle ID** and **Signature hash key**.



6. Configure branding

1. On the **Configure Branding** step, set the following options for your app:

> **Note**
>
> All images must be in .png format. Default images will be used if no custom images are selected.

Expand table

Setting	Description	Requirements
App icons	Icons for your app	iOS: 1024x1024 px or larger Android: 432x432 px or larger
Splash screen image	Image shown while the app loads	.png format
Welcome screen image	Image shown on the sign-in screen	.png format
Background fill color	Color for welcome screen background	Hexadecimal color code
Button fill color	Color for buttons	Hexadecimal color code
Status bar text theme	Color for the status bar text	Light or Dark

2. Select **Next**.

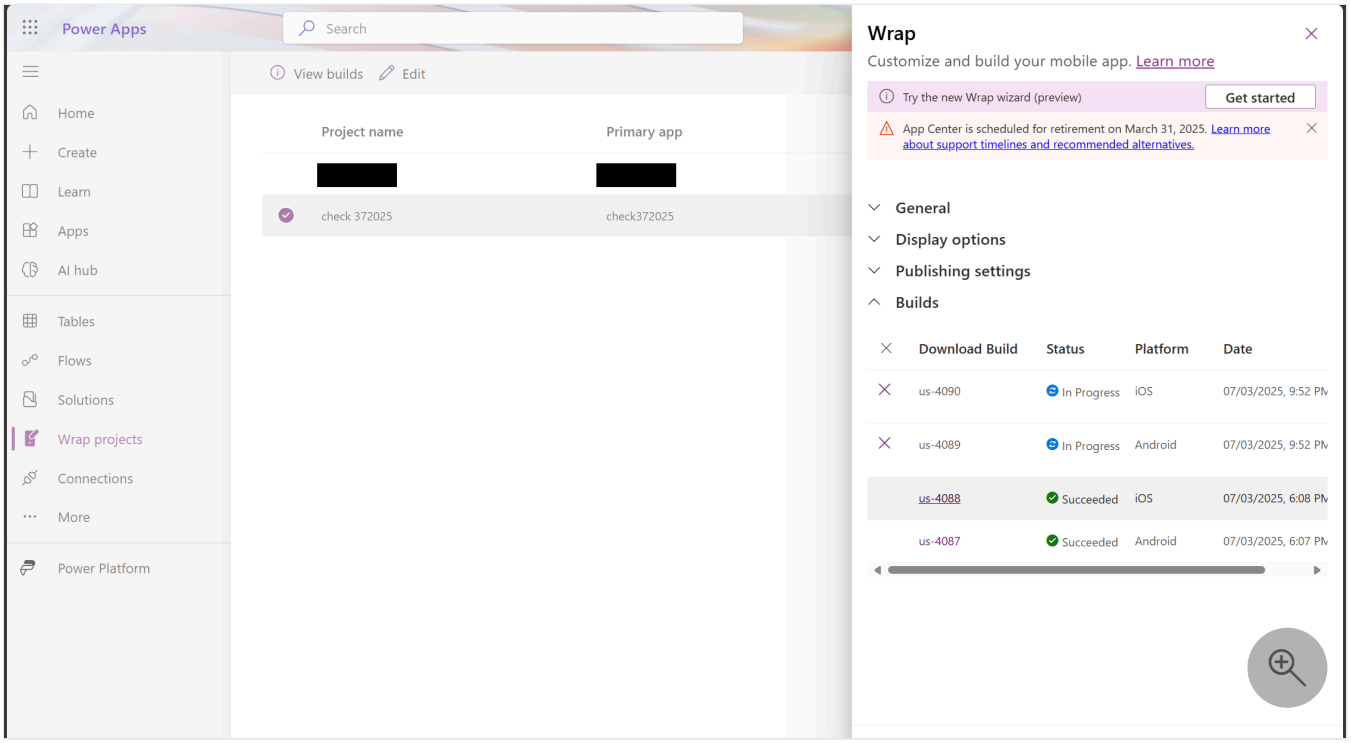
7. Wrap up and build

1. On the **Wrap up** screen, review your app details and select **Build**.
2. After a successful build, your app package will be available in the Azure blob storage you specified.

View your build

You can view your build in several ways:

- After building, select **View Builds**
- Go to **Wrap projects** in the side pane, hover over the required project, and select it
- Select the **View builds** option in the top header



> **Note**
>
> To manually code sign an iOS app, unzip the IPA file using a Mac device.

Test and distribute your app

Test your app and distribute it as needed. If you encounter issues, see the troubleshooting page.

Set environment variable prefix

The name of the new environment variable must have the prefix "new." If it doesn't, follow these steps:

1. Go to **Solutions** > **New solution**.
2. Select a **Publisher** or create one.
3. Select the edit icon next to the **Publisher** to view or change the prefix.
4. If the prefix isn't "new," change it to "new."
5. Save your changes.

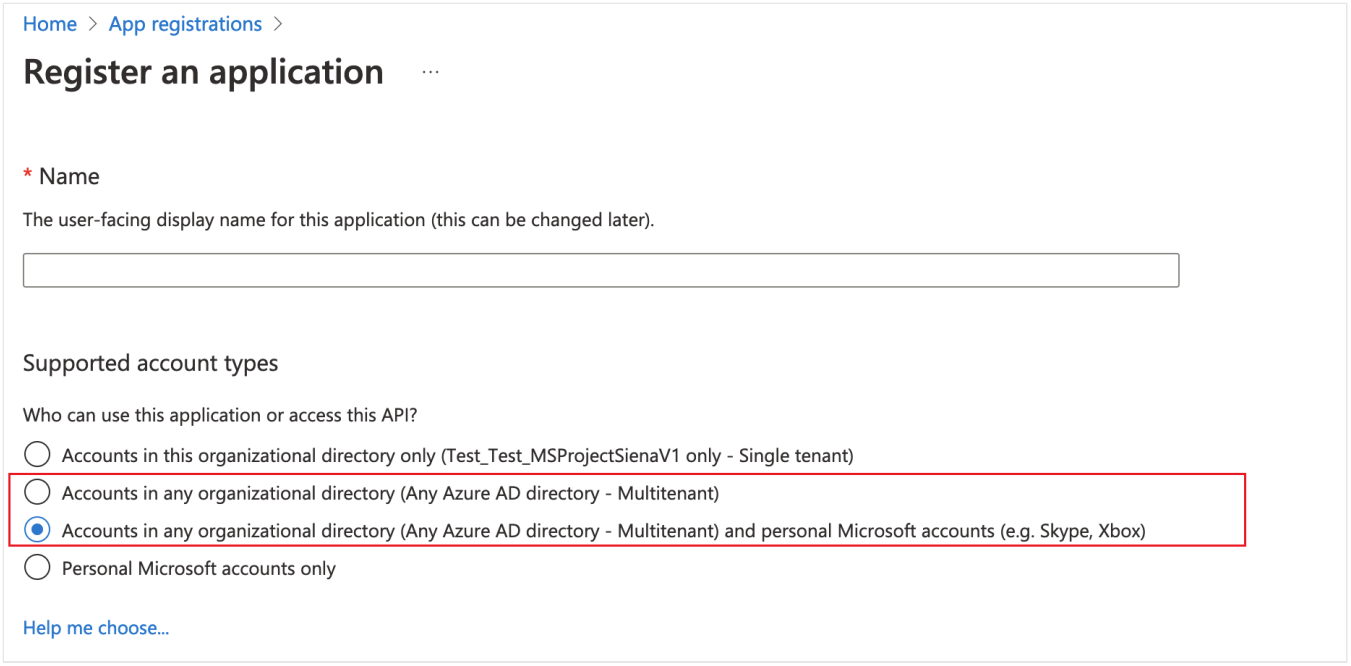
Register your app on Azure portal manually (optional)

You can create your app registration automatically in the wizard or manually in Azure. More information: Quickstart: Register an application with the Microsoft identity platform.

> **Note**
>
> Both single tenant and multitenant customers can use wrap to create native mobile apps based on their Power Apps canvas apps.

When registering, select an account type containing **Any Microsoft Entra directory - Multitenant**:

- Accounts in any organizational directory (Any Microsoft Entra directory - Multitenant)
- Accounts in any organizational directory (Any Microsoft Entra directory - Multitenant) and personal Microsoft accounts such as Skype or Xbox



> **Important**
>
> - Wrap only supports **Multitenant** account types currently. The single tenant account type is not yet supported. More information: Account types in Microsoft identity platform.
> - You must create a separate **Redirect URI** for each platform (iOS, Android).

Sign your mobile app package manually (optional)

You can sign your app automatically in **Step 2** or manually after building. Code signing is different for Android and iOS.

Advantages of automatic signing for iOS and Android (APK):

- You don't need to repeat the signing process during rewrapping.
- You don't have to wait for app developers to finish the process.
- You don't need to set up Android Studio or remember passwords.
- You don't need a Mac device for iOS signing.

Expand table

Platform	Signing Method
iOS	Manual code sign for iOS
Android	Manual code sign for Android
Google Play Store	Code signing for Google Play Store

See also

- Troubleshoot issues with the wrap feature in Power Apps
- Wrap overview
- Manual code sign for iOS
- Manual code sign for Android
- Create your Azure Key Vault for automated code signing
- Frequently asked questions for wrap

Last updated on 04/02/2026

Manual code sign for iOS

06/16/2025

This article explains how to code sign for iOS. You need to sign your app for iOS if you selected iOS as one of the platforms while creating and building your wrap project.

> **Important**
>
> Using Xcode to digitally sign your wrapped mobile apps for iOS is not supported. Follow the instructions below to sign your wrapped mobile app packages for iOS correctly.

Before you begin

Prerequisites

Before you start, make sure you have:

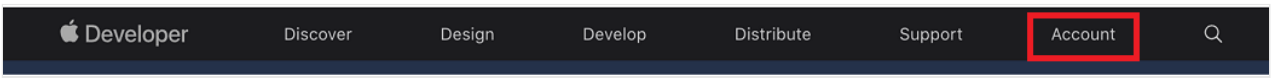
- App ID
- Device UDIDs (only for testing and development purposes)
- A macOS device to code sign for iOS

Set up your Mac

1. Install **Xcode**. More information: Xcode
2. Install PowerShell for macOS. More information: Installing PowerShell on macOS
3. Enroll in the Apple Developer Program
4. For organization distribution, sign up for the Apple Enterprise Developer Program

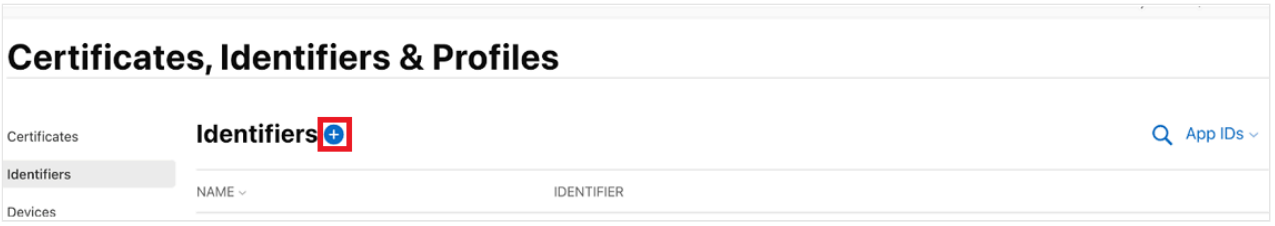
Create App ID

1. Sign in to your developer account at https://developer.apple.com and select the **Account** tab.

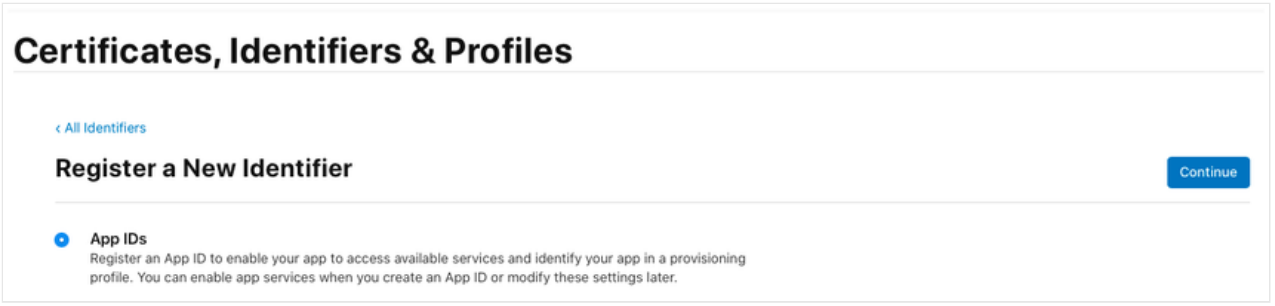


2. Go to **Certificates, IDs & Profiles > Identifiers**.

3. Select + to create a new identifier.

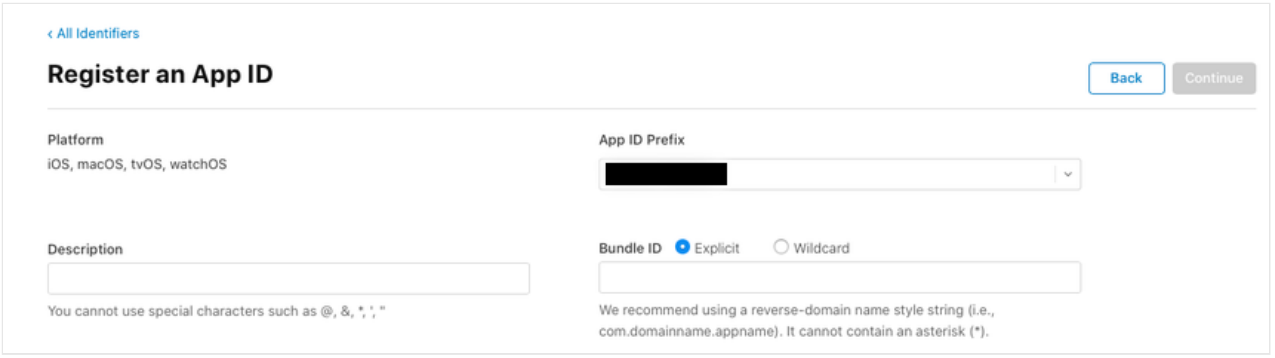


4. Select **App IDs**, then select **Continue**.



5. Select the type as **App**, then select **Continue**.

6. Register an **App ID**:



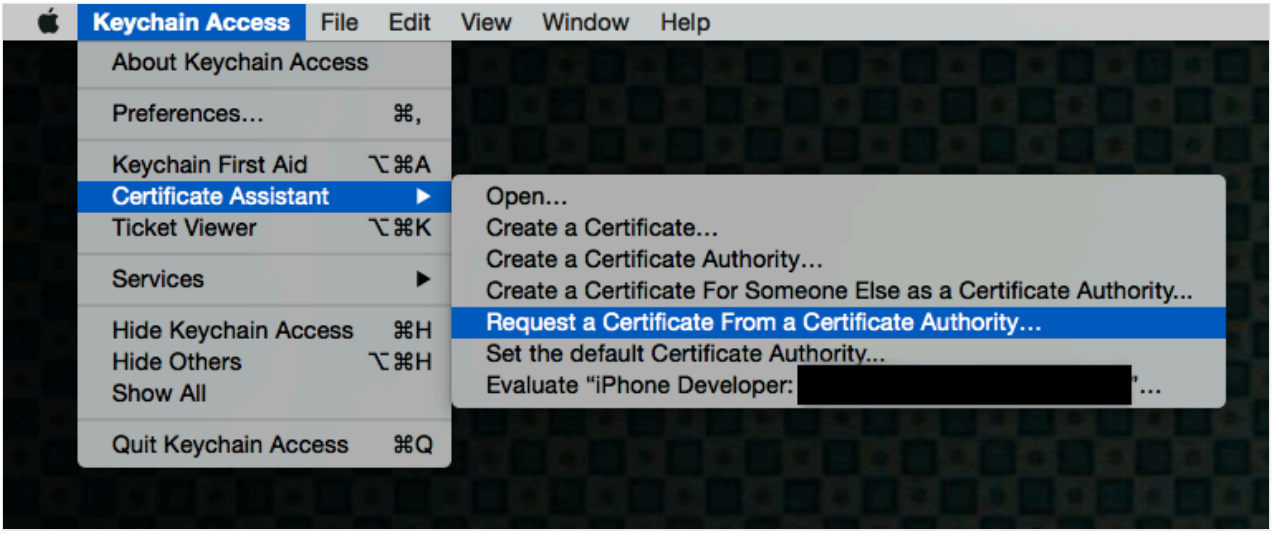
   - **Description** - Name of your app.
   - **Bundle ID** - Select **Explicit Bundle ID**.
   - Enter the bundle ID that you used while creating the wrap project. More information: Bundle ID
   - Enable these capabilities:
     - Associated Domains
     - iCloud
     - NFC Tag Reading
     - Push Notifications
   - Select **Continue**.

7. Review and register the App ID.

Create distribution certificate

Create Certificate Signing Request (CSR)

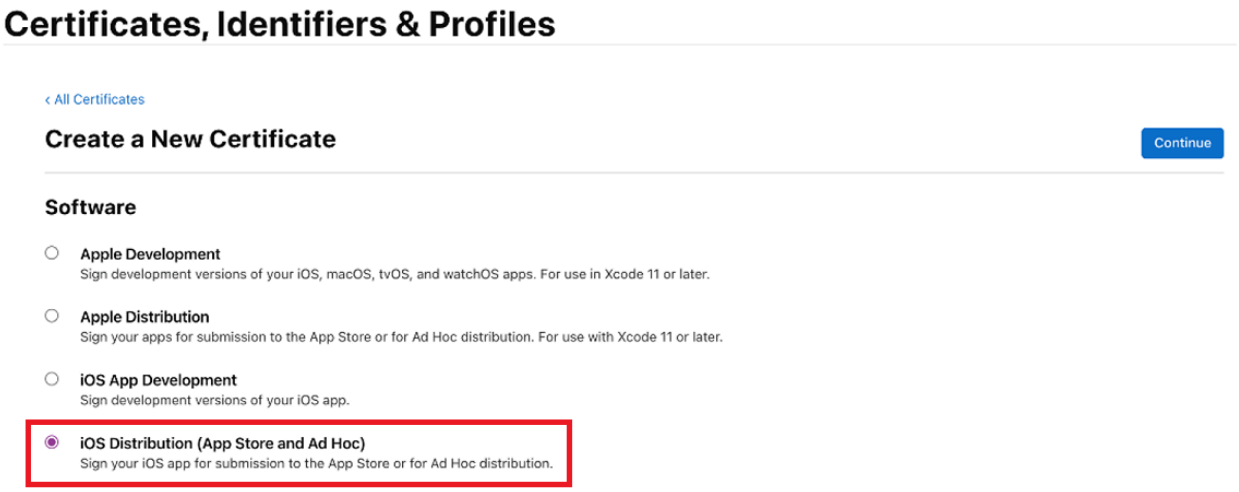
1. On your Mac, go to the **Applications** folder > **Utilities** > open **Keychain Access**.
2. Select **Keychain Access** > **Certificate Assistant** > **Request a Certificate from a Certificate Authority**.



3. Fill out the **Certificate Information**:
    - In the **User Email Address** field, enter your Apple ID email address.
    - In the **Common Name** field, enter your name.
    - In the **Request** group, select **Saved to disk**.
    - Save the file to your Mac.
    - Select **Continue**.

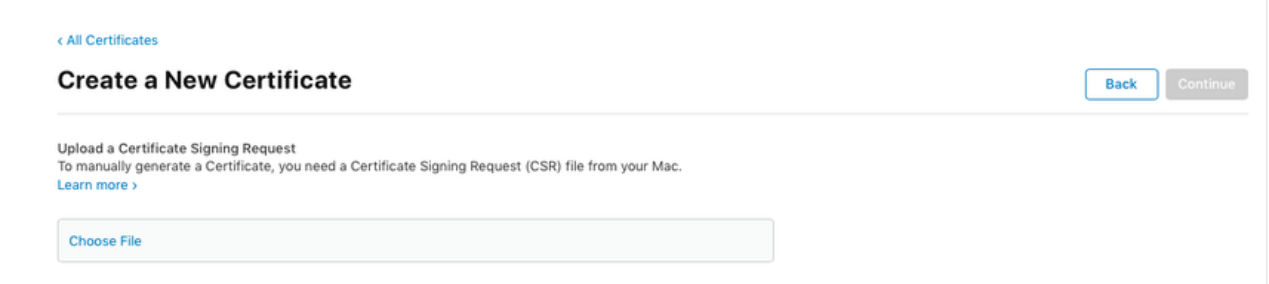
Upload and install certificate

1. Sign in to your developer account at https://developer.apple.com by selecting the **Account** tab.
2. Go to **Certificates, IDs & Profiles > Certificates**.
3. Select + to create a new certificate.
4. Select **App Store and Ad Hoc** > **Continue**.



> **Note**
>
> If you have an Enterprise Developer Account, you'll have the option to create an Enterprise Distribution certificate.

5. Upload the Certificate Signing Request (CSR) file that you generated in the previous steps.



6. Select **Continue**, and download the certificate to your Mac.
7. Double-click the downloaded .cer file to install it in Keychain Access.
8. Make note of the **Name** of the certificate (usually formatted as **iPhone Distribution: Name (Team ID)**). This value is the code signing identity needed for signing.

Register devices for testing

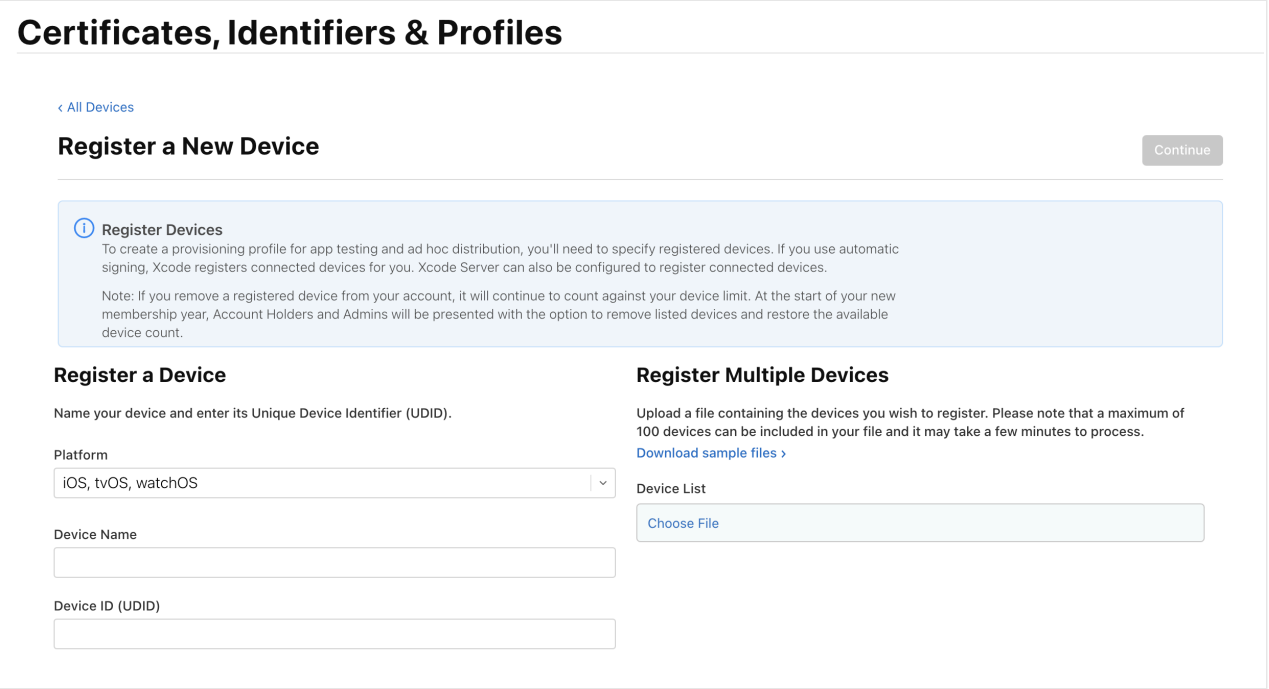
> **Note**
>
> This step is only necessary if you're distributing the app to external users. It's not required if the app is only for internal users.

1. Sign in to your developer account at https://developer.apple.com by selecting the **Account** tab.

2. Go to **Certificates, IDs & Profiles > Devices**.

3. Select + to register one or more devices.

4. Enter the **Device Name** and **Device ID (UDID)**.



5. Select **Save**.

> **Tip**
>
> You can register multiple devices at once by uploading a list of UDIDs.

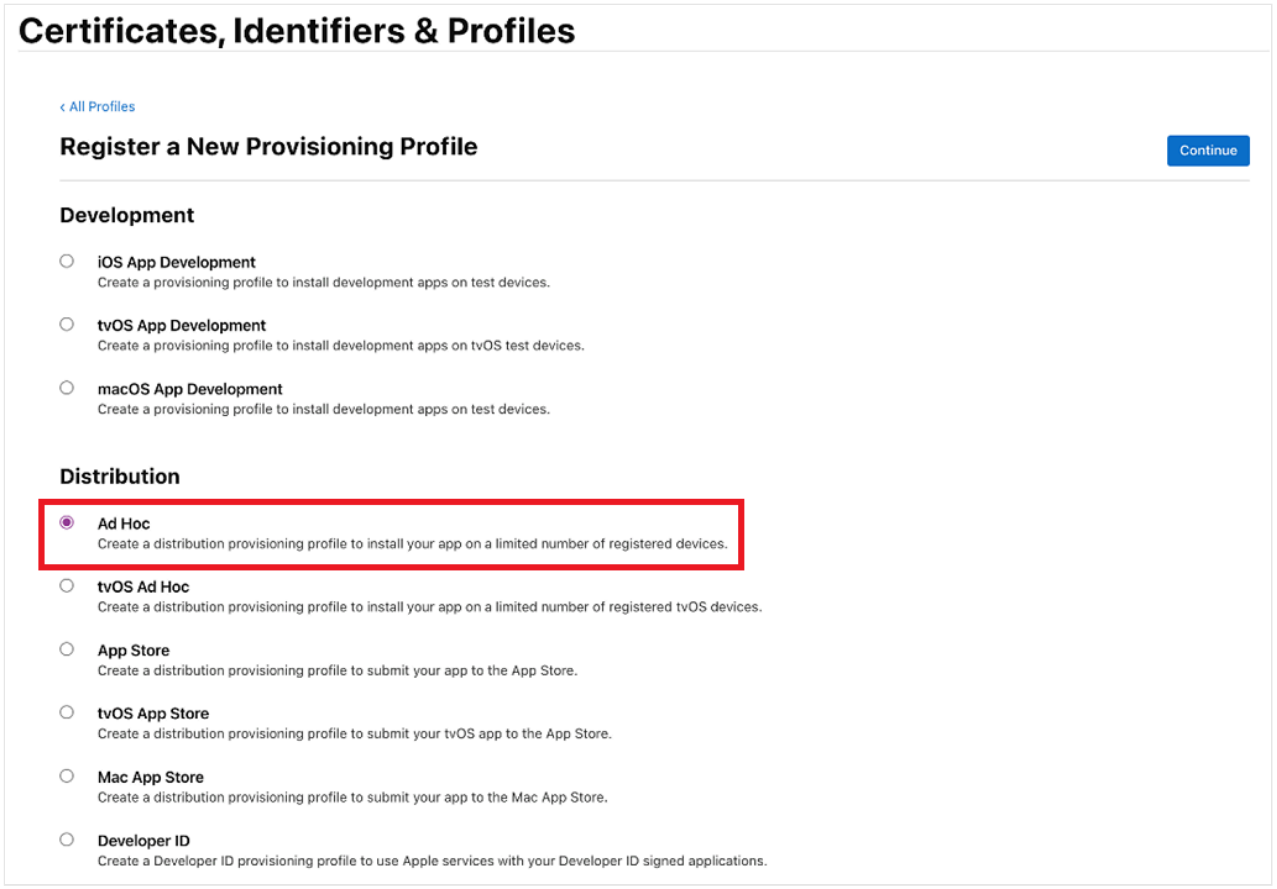
Create iOS Provisioning Profile

1. Sign in to your developer account at https://developer.apple.com and select the **Account** tab.

2. Go to **Certificates, IDs & Profiles > Profiles**.

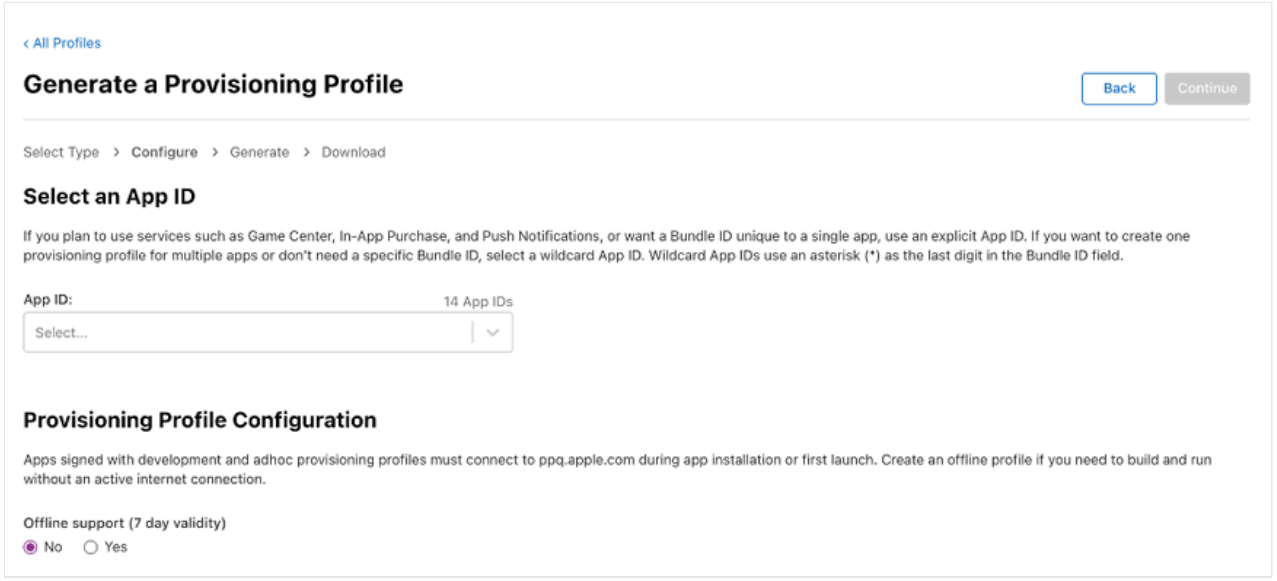
3. Select + to create a new profile.

4. Select **Ad Hoc** > **Continue**.

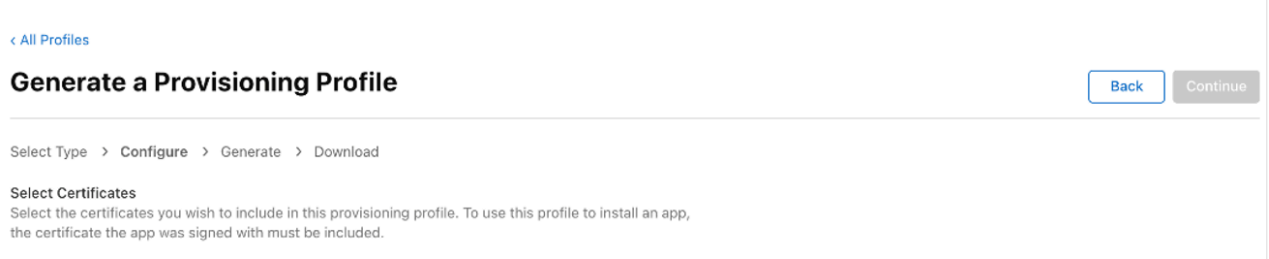


> ⓘ Note
>
> For production use, select the **App Store** distribution method.

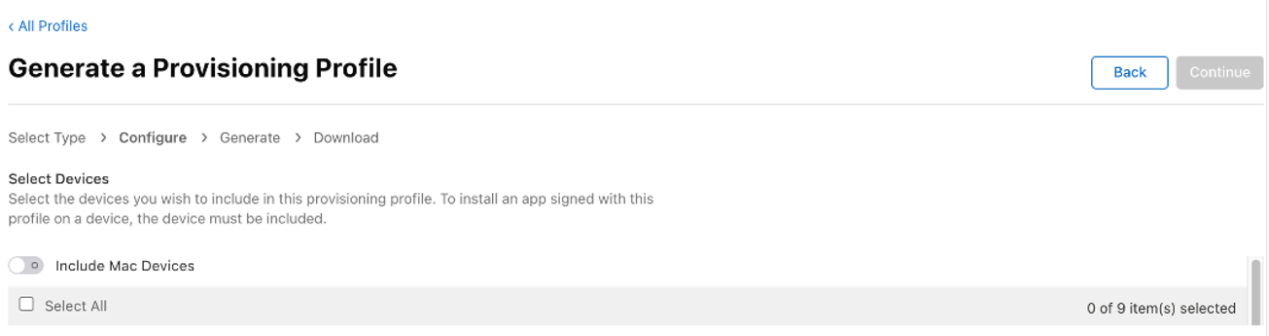
5. Select the App ID you created earlier, then select **Continue**.



6. Select the certificate you created earlier, then select **Continue**.



7. Select all the test devices where you want to install your app, then select **Continue**.



8. Name your provisioning profile and note down the name.
9. Select **Generate** and download the profile to your Mac.
10. Double-click the downloaded file (*.mobileprovision) to register it with Xcode.

Sign the iOS Archive

1. Download and unzip the **iOS-Archive.zip** file from your Azure blob storage. This creates a folder named after the Bundle ID. In the example below, the Bundle ID is **com.single.wrap**.
2. For enterprise signing only: Open the file **Distribution-exportOptions.plist** with Xcode and change the value for the **method** field to **enterprise**.

   > ⓘ **Note**
   >
   > Enterprise signing is not supported with Key Vault signing.

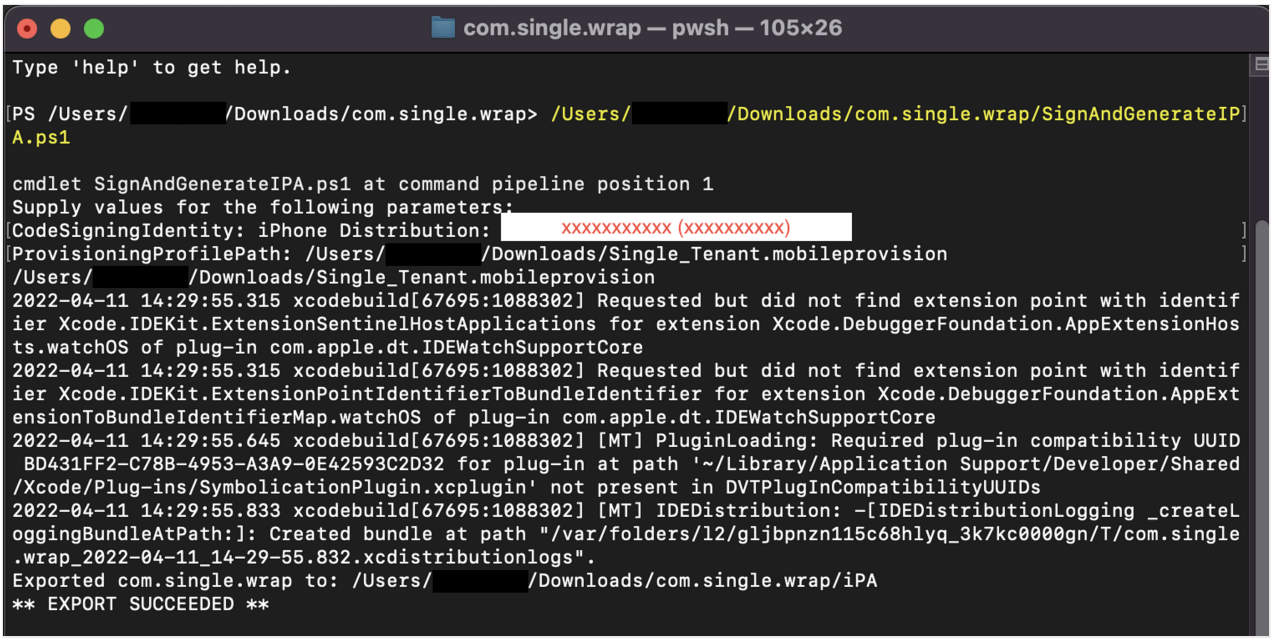
3. Open Terminal and change directory to the unzipped folder.
4. Enter `pwsh` to start PowerShell in the terminal.
5. Run the PowerShell script with your certificate and provisioning profile information:

```
```

```
./SignAndGenerateIPA.ps1 -CodeSigningIdentity "YOUR_CODE_SIGNING_IDENTITY" -
ProvisioningProfilePath "PATH_TO_PROVISIONING_PROFILE"
```

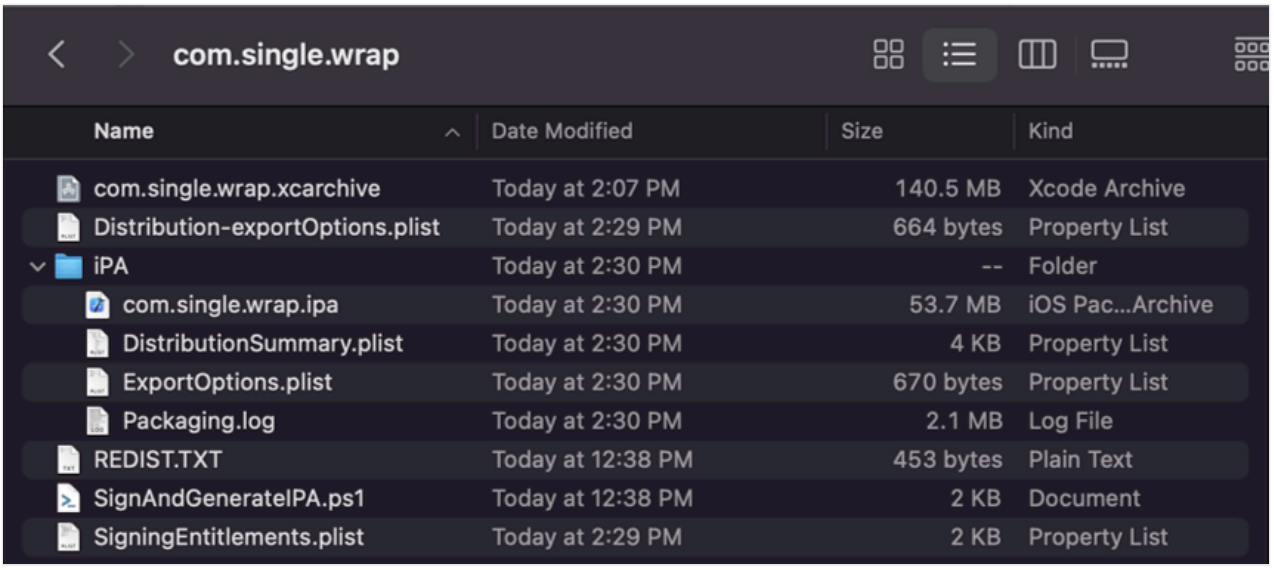
> **Note**
>
> To find your **CodeSigningIdentity** value, right-click on the **.mobileprovision** file, select **More Info** > **Preview**. Look for the **Name** field under **Certificates**.

> **Tip**
>
> The **ProvisioningProfilePath** is the path to the .mobileprovision file you downloaded. For example: `/Users/username/Downloads/MyProvisioningProfile.mobileprovision`



6. When the script completes, a **.ipa** file is created (for example, **com.single.wrap.ipa**).

> **Important**
>
> If the script fails, delete the unzipped folder and files as they might be corrupted. Unzip **iOS-Archive.zip** again before retrying.



Distribute the app

Install the **.ipa** file on registered devices using the **Finder** app.

For enterprise distribution with Microsoft Intune:

- Add an iOS/iPadOS line-of-business app to Microsoft Intune
- Give your app access to the Intune app protection service

See also

- Wrap overview
- Manual code sign on Android
- Frequently asked questions for wrap
- Troubleshoot issues with the wrap feature in Power Apps

Code sign for Android

07/30/2025

This article explains how to manually code sign your Android (APK) app for Power Apps wrap. Sign your app for Android if you select Android as one of the platforms when you create and build your wrap project.

> **Important**
>
> To sign an AAB app for Google Play distribution, see Sign your app.

Before you begin

Required software

Before you begin, install and set up:

- Android Studio
- OpenSSL
- apksigner tool (included with Android Studio)

Prerequisites

You need:

- The application name to create the hash key.
- The latest APK file to sign in.

> **Note**
>
> Use the `.jks` extension for manual signing and the `.pfx` extension for automatic signing.

Generate key and signature hash

> **Note**

Skip to Manual signing if you already generated keys and the signature hash when you created the app registration.

Set up environment variables

If you don't set environment variables:

1. Download and install Android Studio and OpenSSL.
2. Add `keytool` and `openssl` to your PATH environment variable:

    - Add `C:\Program Files\Android\Android Studio\jbr\bin` to your PATH.
    - Add the path of `openssl.exe` (for example, `C:\Program Files\OpenSSL-Win64\bin`) to your PATH.

For manual signing process

Generate signature hash key and certificate

Run this command in the command prompt:

```
keytool -genkey -alias powerappswrap -keyalg RSA -keystore powerappswrap.jks -
keysize 2048 -validity 10000
```

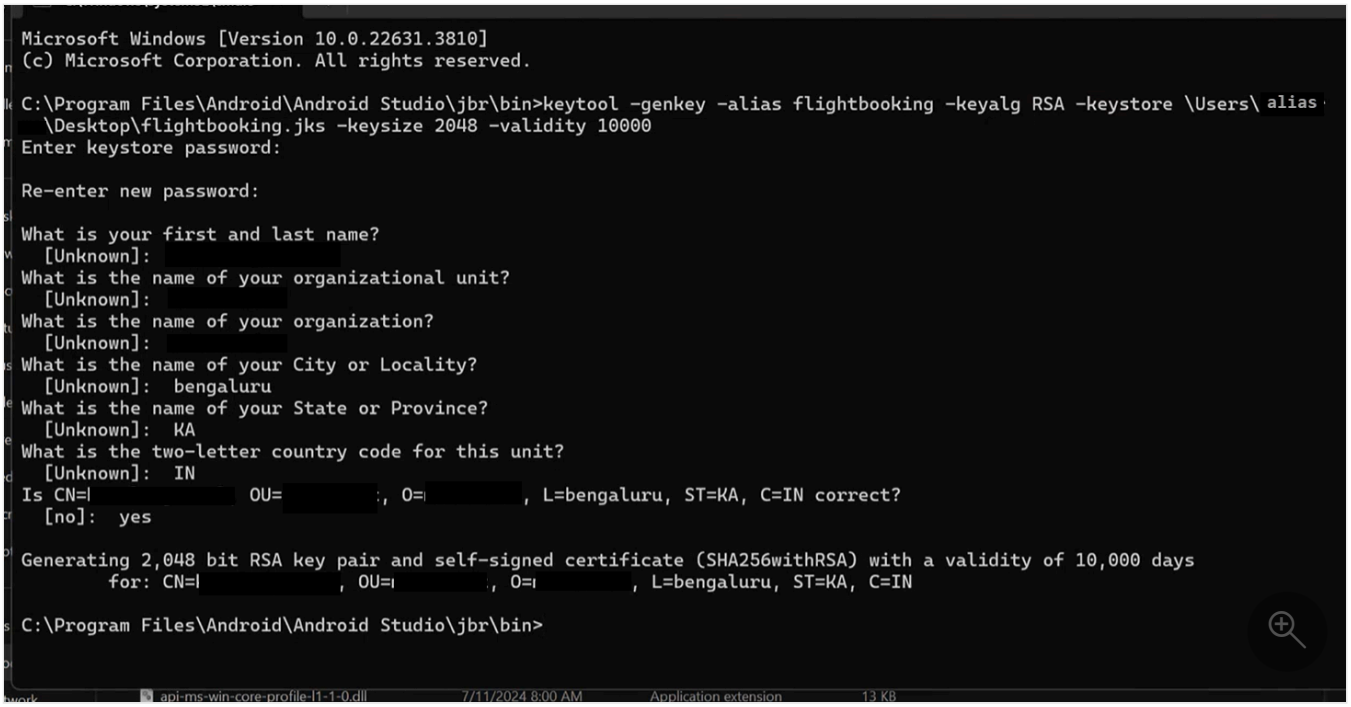
When prompted:

1. Enter a password for your keystore.
2. Enter your name, organization, location, and other required details.
3. Confirm the information.

Run this command to generate key and certificate:

```
keytool -exportcert -alias powerappswrap -keystore powerappswrap.jks | openssl
sha1 -binary | openssl base64
```

When prompted, enter the keystore password you created earlier.



Parameters explained:

Expand table

Parameter	Description
genkey	Command to generate a key
alias	Alias for the keystore entry
keyalg	Key algorithm name
keystore	Name of the keystore
keysize	Size of each key
validity	Validity of the key in days
exportcert	Reads the certificate from the keystore
openssl	Generates SHA1 key for Android

For automatic key vault signing process

Generate signature hash key and certificate

Run this command in the command prompt:

```
keytool -genkey -alias powerappswrap -keyalg RSA -keystore powerappswrap.pfx -
keysize 2048 -validity 10000
```

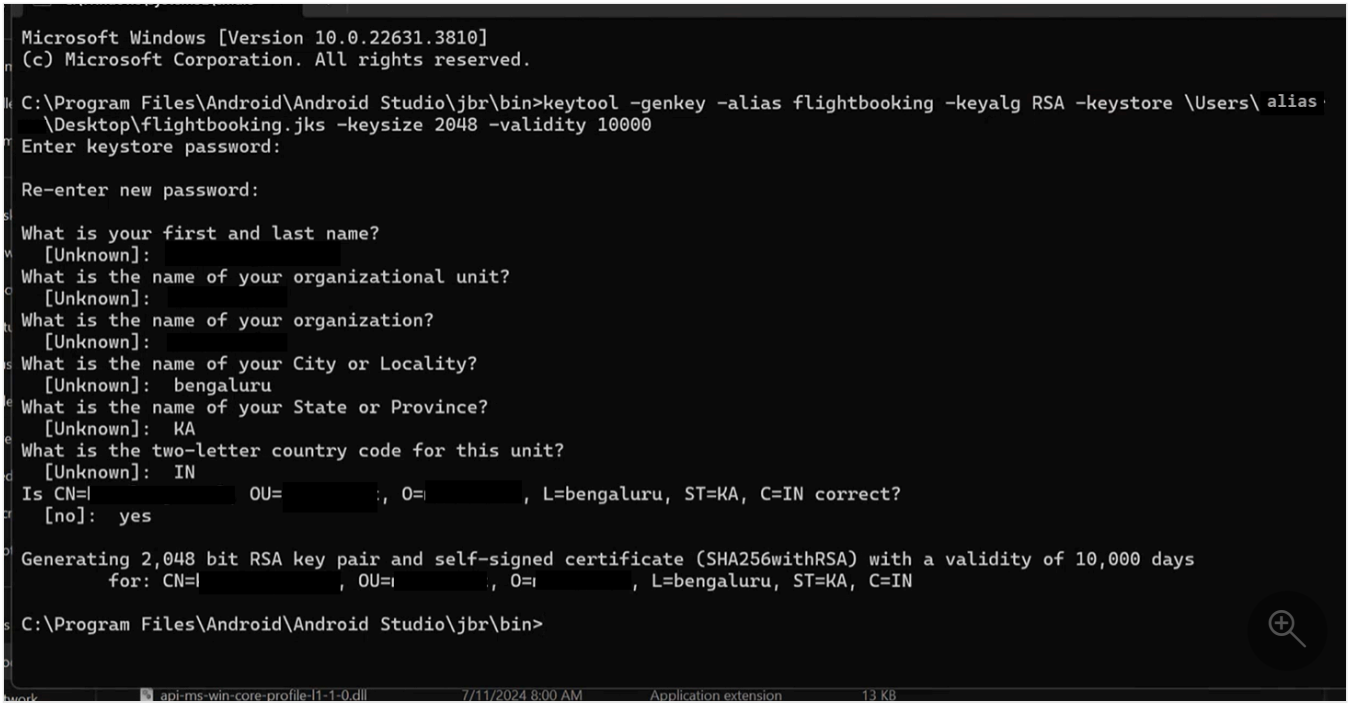
When prompted:

1. Enter a password for your keystore.
2. Enter your name, organization, location, and other required details.
3. Confirm the information.

Run this command to generate key and certificate:

```
keytool -exportcert -alias powerappswrap -keystore powerappswrap.pfx | openssl
sha1 -binary | openssl base64
```

When prompted, enter the keystore password you created earlier.



Parameters explained:

Expand table

Parameter	Description
genkey	Generates a key.
alias	Alias for the keystore entry.
keyalg	Key algorithm name.

Parameter	Description
keystore	Name of the keystore.
keysize	Size of each key.
validity	Validity of the key in days.
exportcert	Reads the certificate from the keystore
openssl	Generates SHA1 key for Android

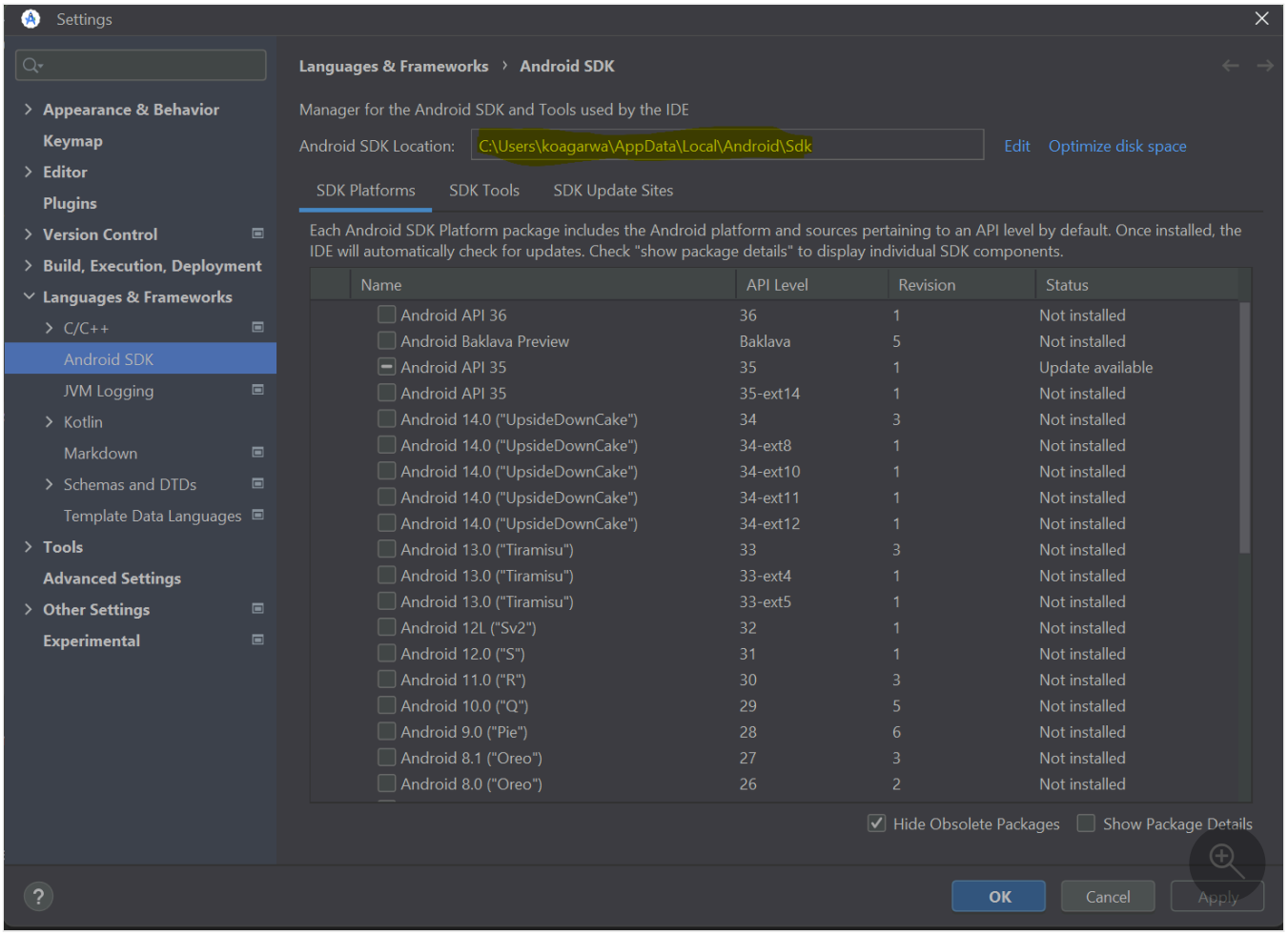
Manual signing of APK package (Not for KV signing)

Follow these steps if you don't use automatic sign-in during wrap or if you try to upload an AAB file for Play Store. To avoid repeating this process, use automatic sign-in when possible.

Locate the apksigner tool

1. Open Android Studio.
2. Go to **Tools** > **SDK Manager** > **Android SDK Location** to find your SDK path.
3. In the SDK directory, navigate to:

    - **build-tools** > **[version number]** > find **apksigner.bat** (Windows) or **apksigner** (Mac/Linux)
    - Or: **build-tools** > **[version number]** > **lib** > find **apksigner.jar**



Sign the APK file

Run this command to sign your APK:

```
apksigner.bat sign --ks PATH_TO_KEYSTORE --ks-key-alias KEY_ALIAS PATH_TO_APK
```

Parameters explained:

Expand table

Parameter	Description
ks	Path to your keystore file (for example, `C:\Users\name\Desktop\powerappswrap.jks`)
ks-key-alias	The alias you used when generating the key (for example, `powerappswrap`)
PATH_TO_APK	Full path to your APK file (for example, `C:\Users\name\Downloads\MyApp.apk`)

When prompted, enter the keystore password.

Example:

```
apksigner.bat sign --ks C:\Users\name\Desktop\powerappswrap.jks --ks-key-alias
powerappswrap C:\Users\name\Desktop\MyApp.apk
```

Verify the signature

After signing, verify the APK signature with:

```
apksigner.bat verify --verbose PATH_TO_APK
```

A successful verification confirms your APK is properly signed and ready for distribution.

For more information, see Android Studio command line tools: apksigner.

Distribute the app

After signing your app, you can distribute it using several methods:

Distribution options

- **Microsoft Intune**: To distribute using Microsoft Intune, see Add an Android line-of-business app to Microsoft Intune.
- **Intune app protection**: To give your app access to the Intune app protection service, see Give your app access to the Intune app protection service.
- **Direct distribution**: You can also distribute the signed APK directly to users for manual installation.

See also

- Wrap overview
- Manually Signing the APK - Xamarin
- Manual code sign on iOS
- Frequently asked questions for wrap
- Troubleshoot issues with the wrap feature in Power Apps

Code signing process for AAB file

06/16/2025

This article explains how to manually code sign an AAB file.

Prerequisite

1. Set up Android Studio.
2. Install bundletool for converting AAB file to APK file. You can download the latest version from the official site.
3. Install jarsigner for signing in the AAB file. You can install jarsigner by downloading and setting up the Java Development Kit (JDK).

Signing process for AAB file

> **Note**
>
> You need to sign the AAB file even if you select the automatic signing process in the wrap wizard.

To sign an AAB file, follow these steps:

1. Open a terminal or command prompt.
2. Go to the directory where `jarsigner` is installed.
3. Run the commands shown in the image.

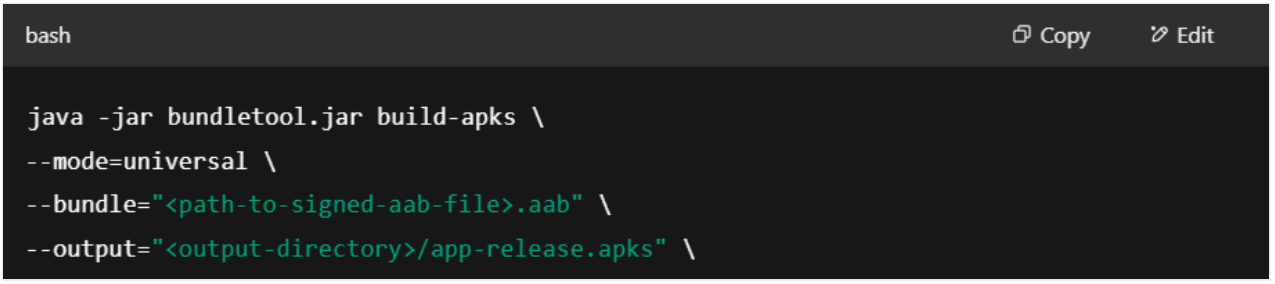
```bash
jarsigner -verbose -sigalg SHA256withRSA -digestalg SHA-256 \
-keystore "<path-to-your-keystore>.jks" \
-signedjar "<output-signed-aab-file>.aab" \
"<input-unsigned-aab-file>.aab" \
<bundle-id>
```

Replace the placeholders: - `<path-to-your-keystore>.jks` – Path to your .jks keystore file - `<output-signed-aab-file>.aab` – Output file name. - `<input-unsigned-aab-file>.aab` – Unsigned .aab file. - `<bundleid>` – Bundle ID used while wrapping.

Convert AAB to APK file for manual debugging

To convert an AAB file to an APK file, follow these steps:

1. Open a terminal or command prompt.
2. Run the commands shown in the screenshot.



   Replace the placeholders: - `<path-to-signed-aab-file>.aab` – Path to the signed AAB file. - `<output-directory>` – Output directory for the .apks file.

- Sign the APK if you use the manual signing process. For more information, see Code sign for Android - Power Apps.

See also

- Manual code sign for iOS
- Manual code sign for Android

Set up Azure Key Vault for wrap wizard code signing

08/19/2025

Azure Key Vault is a cloud service that securely stores secrets like certificates, passwords, encryption keys, and other sensitive data. Use Key Vault to manage certificates and secrets for automated code signing in wrap wizard. For more information, see Introduction to Azure Key Vault.

This guide explains how new users create and configure an Azure Key Vault.

Prerequisites

Before you begin, ensure you have the following:

- A Microsoft Entra subscription to create a key vault.
- Admin access to your Azure tenant.
- An Apple Developer account enrolled in the Apple Developer Program or Apple Enterprise Developer Program.
- An Apple distribution certificate and either an ad-hoc provisioning profile or an enterprise provisioning profile.

Step 1: Create an Azure Key Vault

If you do not already have a Key Vault, follow the steps in Create a vault using the Azure portal.

Step 2: Create a service principal for wrap Key Vault access

A service principal is required so wrap wizard can access your Key Vault.

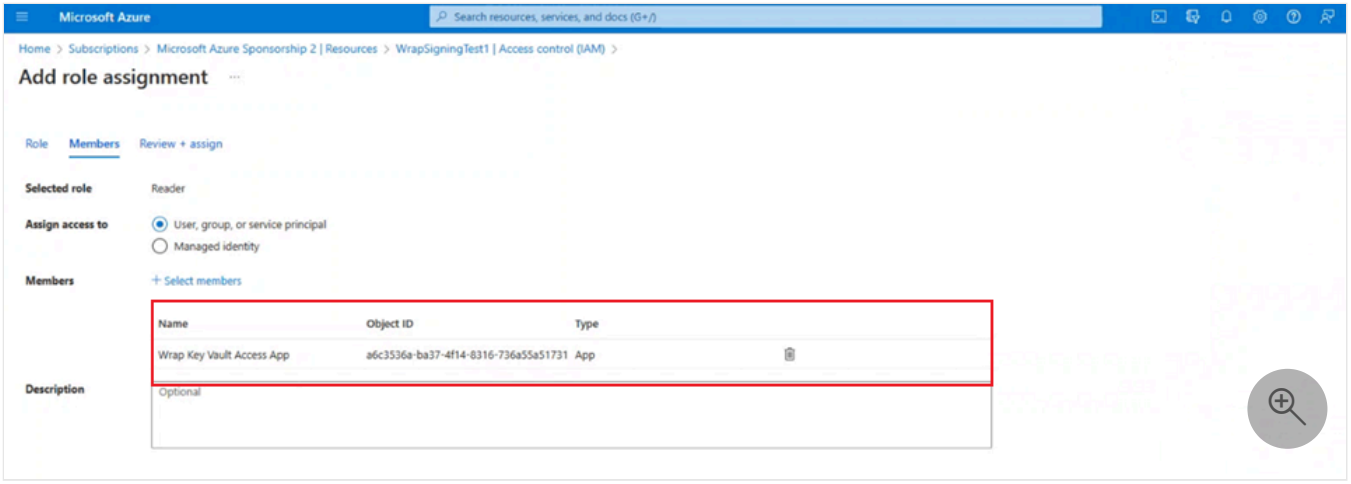
1. Sign in to your tenant as an admin.
2. Open PowerShell and run the following command:

```
Connect-AzureAD -TenantId <your tenant ID>
New-AzureADServicePrincipal -AppId 4e1f8dc5-5a42-45ce-a096-700fa485ba20 -
DisplayName "Wrap KeyVault Access App"
```

Step 3: Assign Reader Role to the Service Principal

Give the service principal permission to read your Key Vault.

1. In the Azure portal, navigate to your Key Vault.
2. Select **Access control (IAM)**.
3. Select **Add** > **Add role assignment**.
4. On the **Members** tab, select **Job function roles**, and then choose **Reader**.
5. Select **Select member**, search for "Wrap KeyVault Access App", and select it.
6. Select **Review + assign**.

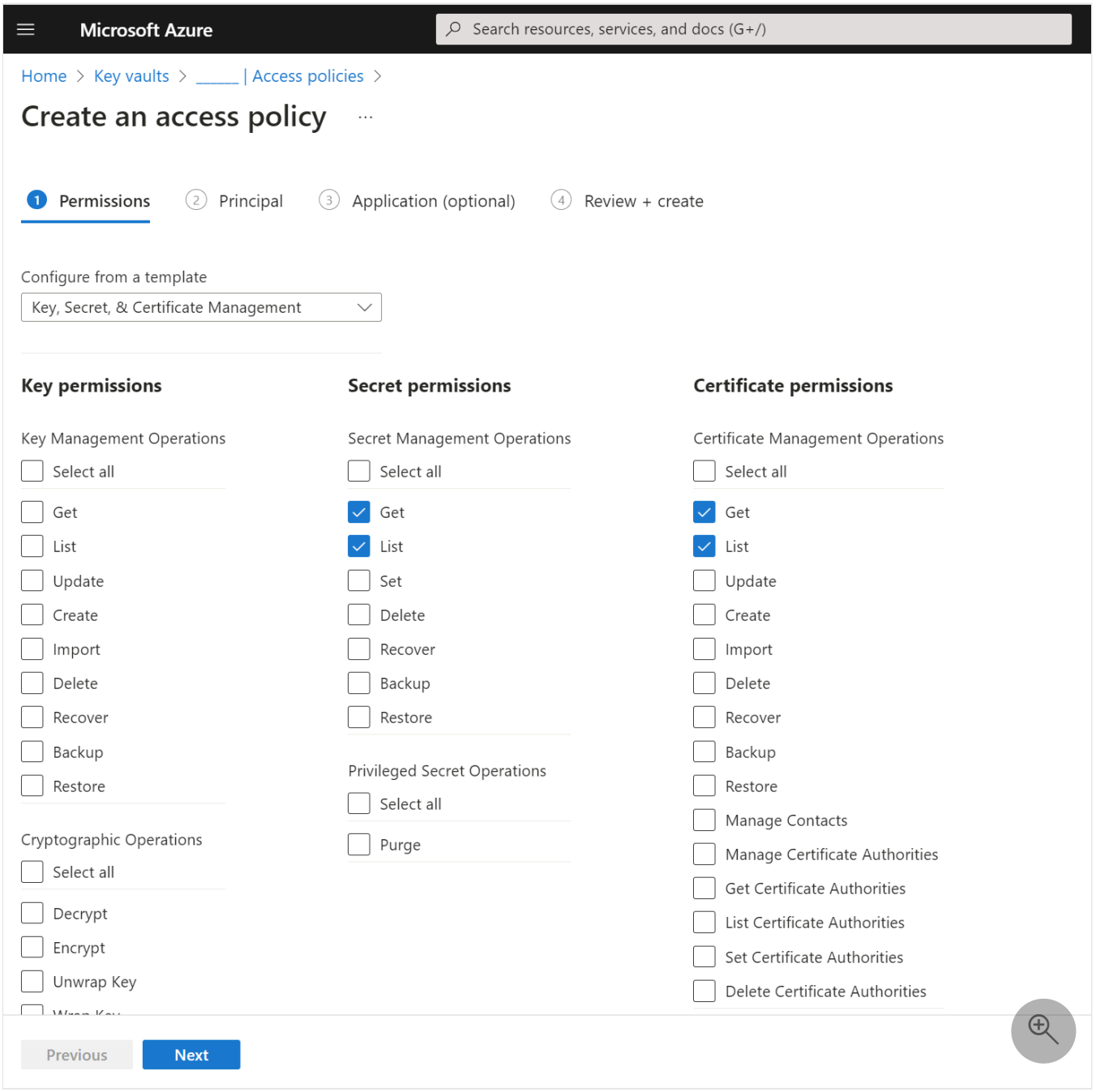


Repeat these steps to add user access permissions to your account.

Step 4: Configure access policies for Key Vault

1. Go to Key Vault settings.
2. In **Access configuration**, select **Vault access policies**.
3. Add a policy for the service principal and your user account.

    - For **Secret permissions** and **Certificate permissions**, select **Get** and **List**.

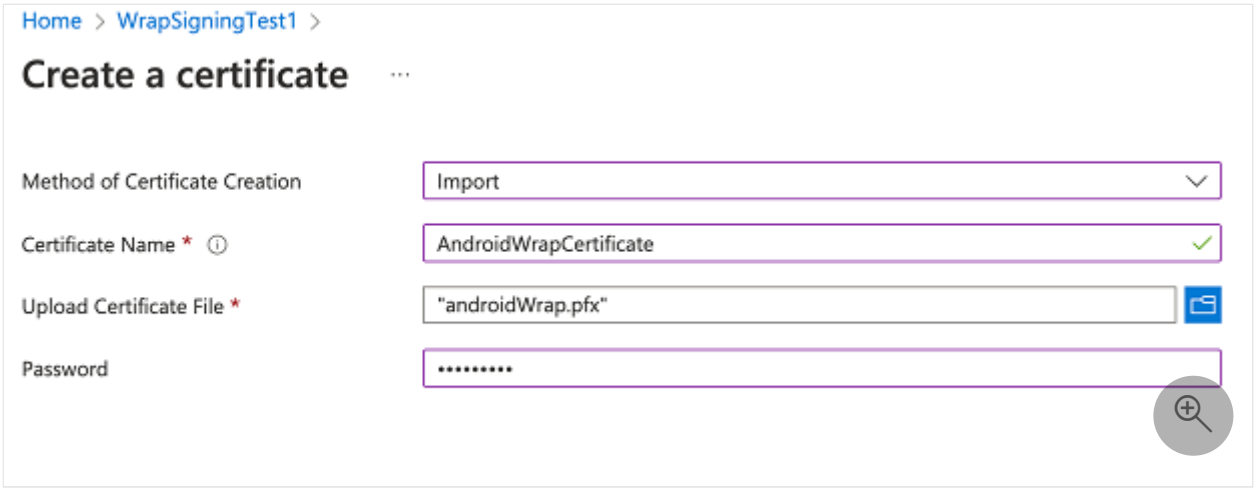


Step 5: Upload Certificates and Secrets

You need platform-specific certificates and secrets for Android and iOS apps.

For Android

- Generate a `.pfx` certificate file.
- Upload it to the **Certificates** section of your Key Vault.
- For guidance, see Generate keys.



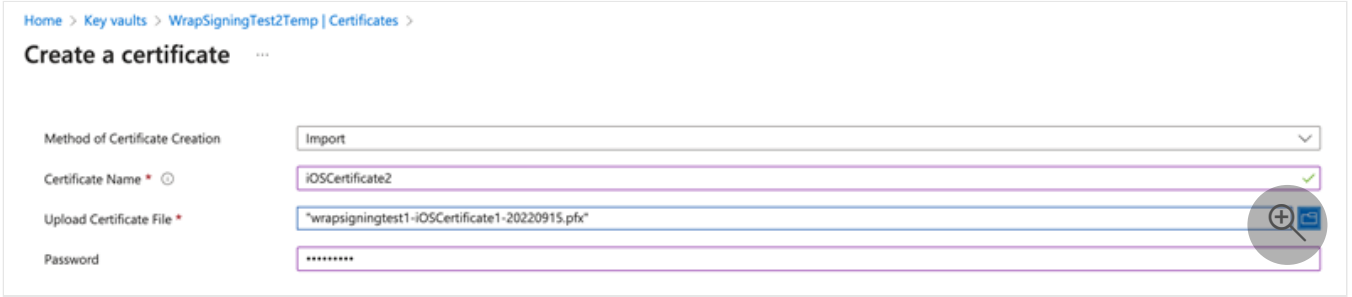
> **Note**
>
> Ensure the certificate name is included in the tag step, and its password matches the store pass parameter you set when creating the `.pfx` file.

For iOS

1. Install your `.cer` file using Keychain Access. For more information, see Create the distribution certificate.
2. Export the certificate as a `.p12` file and rename its extension to `.pfx` (required by Key Vault).
3. Upload the `.pfx` file to your Key Vault and provide the password you set for the `.p12` file.
4. Create a provisioning profile. For more information, see Create the provisioning profile.
5. Encode the provisioning profile to base64:

    - On Mac, run: `base64 -i example.mobileprovision`
    - On Windows, run: `certutil -encode data.txt tmp.b64`

6. Upload the base64 string as a secret in Key Vault and the `.pfx` file as a certificate.

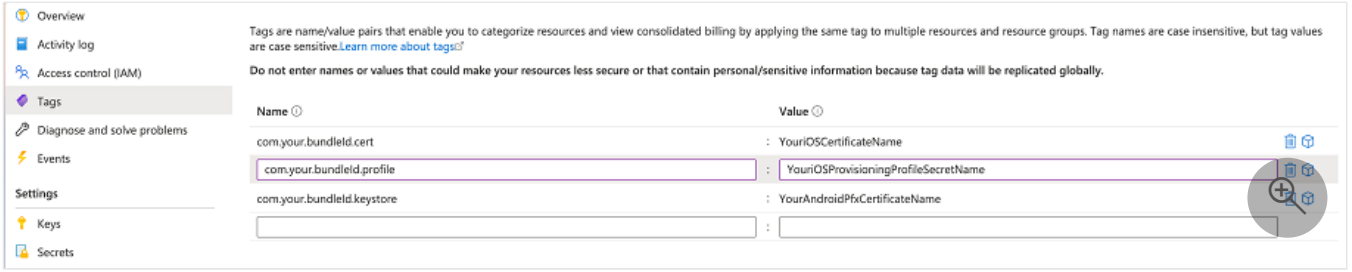


Step 6: Add Certificate Tags

After uploading your iOS or Android certificates, add three tags:

- Tag name: Use your app's bundle ID.
- Tag value: Use the name of the uploaded certificate.

Use the same bundle ID that you'll use in the wrap wizard.



Reference Video

Watch a walkthrough video: How to configure access to key vault

Troubleshooting

If you run into issues, see Troubleshoot issues with the wrap feature in Power Apps.

See also

- Wrap overview
- Code sign for iOS
- Code sign for Android
- Code sign for Google Play Store
- Frequently asked questions for wrap
- Troubleshoot issues with the wrap feature in Power Apps

Use deep links with wrapped mobile apps

Article • 12/06/2022

Deep links let users move from one application to another on computers and mobile devices. For example, a mobile app can deep link to Facebook to sign in. Deep linking an email address could open an email app to allow users to start composing a message. A website could use a deep link to an app store to start download for a specfic mobile app.

You can deep link to a wrapped native mobile app from other apps. Deep links for wrapped mobile apps should start with the following:

```
ms-mobile-apps:///providers/Microsoft.PowerApps/apps/<appID>?tenantId=<tenantId>
```

Expand table

Parameter	Description
<app-id>	Opens the correct app module
<tenantId>	Connects to the correct tenant

For information on how to deep link into other apps in Power Apps, see Use deep links with Power Apps mobile.

Feedback

Was this page helpful? Yes No

Provide product feedback

Benefits and limitations of Wrap

06/17/2025

Wrap gives app makers benefits like improved adoption, better branding, and flexible distribution options. But it also has some limitations you need to consider.

Benefits of wrap

1. **Improved app adoption**: Users easily find and use your app by searching for its name in the App Store or Play Store. This direct access streamlines the experience and increases adoption rates.
2. **Enhanced branding flexibility**: Makers customize their app's branding, improving both distribution and user engagement.
3. **Custom distribution channels**: Distribute your apps through preferred channels, so you have more control and can reach your target audience more effectively, instead of relying only on standard platforms.
4. **Power Apps mobile for frontline workers (FLWs)**: Power Apps Mobile is for frontline workers. The wrap APK lets guest accounts sign in, so frontline workers outside your organization can use it. For details, see Frequently asked questions for wrap: Why am I not able to sign into my wrapped application?.
5. **Access without MFA or Intune**: You can use the Power Apps mobile app without multifactor authentication (MFA) or Microsoft Intune if your organization's conditional access policies allow it.
6. **Offline features**: Wrap apps support offline capabilities.

Limitations of Wrap

1. **Logout button**: There's no visible sign out button.

   > ⓘ Note
   >
   > Users can select and hold the indented application to sign out. Makers should let users know about this method.

2. **Push notifications**: Push notifications aren't supported.
3. **Navigation between apps**: You can't go directly from a specific screen in one app to a specific screen in another app. Only the home screen of the other app opens.
4. **APK size limit**: APK files distributed through the Google Play Store are limited to 100 MB. To support larger apps (up to 150 MB), create an AAB file during the wrap process.

5. **Android hardware back button**: The Android hardware back button doesn't always sync with the application back button.
6. **Feedback**: You can't provide feedback in the wrap app.
7. **Surveys**: Surveys aren't available for wrap users.
8. **Sovereign cloud**: Wrap doesn't support sovereign cloud environments.
9. **Full image view (offline)**: Offline-enabled wrap apps only show image thumbnails, not full image views.
10. **VPN**: The wrap wizard doesn't support creating a wrapped app while you're connected to a VPN. Updates are in progress.

See also

- Troubleshoot issues with the wrap feature in Power Apps
- Frequently Asked Questions

Frequently asked questions for wrap

Frequently asked questions about the **wrap** feature are listed below.

Can single tenant customers use wrap with Power Apps?

Yes. Both single tenant and multitenant customers can use wrap. When registering your app manually in Azure, ensure you select the correct account type. See App registration for directions.

Does the wrap feature in Power Apps support Customer-Managed Keys (CMK) or Lockbox capabilities?

No. The wrap feature in Power Apps doesn't support Customer-Managed Keys (CMK) or Lockbox. Wrap uses Azure DevOps build pipelines, and customer assets might be exposed to Microsoft service operators during the build process.

Can I create mobile apps with Power Apps?

Yes. See Creating mobile apps with Power Apps.

If I create a mobile app with Power Apps, does Microsoft release it for me?

No. Microsoft wraps your app in a native mobile app shell for you to digitally sign and distribute.

How do I digitally sign my mobile app?

You can automatically sign your mobile app package in Step 2 of the wrap wizard if you have set up Azure Key Vault. For manual signing, see Android, iOS, and Google Play Store.

Can I use Xcode to digitally sign my mobile apps for iOS?

No. Signing your mobile app with Xcode isn't supported in wrap. To digitally sign your wrapped mobile app package for iOS, see Code sign for iOS.

How do I distribute my mobile app?

You can use Microsoft Intune, Google Play, or Apple Business Manager.

How do I configure my mobile app for distribution through Microsoft Intune?

Wrap wizard automatically configures your app for distribution with Microsoft Intune. To troubleshoot, you can manually grant all the canvas app(s) that you intend to wrap *Microsoft Mobile Application Management* API permission manually. More information: Configure the API permissions for your app manually.

Do I need to rewrap and redistribute my mobile app frequently?

Power Apps Mobile is updated periodically. To ensure compatibility and benefit from bug fixes and new features, we recommend that you rewrap and redistribute your mobile app at least monthly.

Do I need to rewrap and redistribute my mobile app after making changes to the canvas app(s) included in the package?

No. Published changes to the included canvas app(s) are downloaded automatically by existing, released versions of your mobile app. However, we recommend that you rewrap and redistribute your mobile app on a monthly basis to benefit from platform bug fixes, updates, and new features.

How do I update the icons and colors in my mobile app?

Update icons and colors by rewrapping your apps with new assets and redistributing the updated mobile app package.

How do I add or remove canvas apps in my wrapped mobile app?

Add or remove canvas apps by rewrapping your mobile app package with the desired set of apps and redistributing your mobile package.

What are Primary and Secondary apps in the context of wrap?

The Primary app provides the main experience of your mobile app. Secondary apps are bundled for performance reasons and must be linked from your primary app. See wrap multiple canvas apps together.

What are the advantages of including Secondary apps in the bundle, if I can still Launch() them without doing so?

Including Secondary apps in the bundle provides better launch performance when opening them for the first time. However, it's optional. If you have many Secondary apps, you may choose not to bundle them to reduce the total app size.

Can I add apps that use Power BI data to wrap?

Yes. Wrap wizard automatically configures your mobile app to be able to use Power BI. To troubleshoot, you can manually grant *Power BI* API permission to all your apps that use Power BI data. More information: Configure the API permissions for your app manually.

Can I deep link into a wrapped mobile app?

Yes. You can use `ms-mobile-apps:///providers/Microsoft.PowerApps/apps/<appID>?tenantId=<tenantId>` schema to deep link into your wrapped mobile apps. Refer to Using deep links with wrapped mobile apps for more information.

Why do I see this error message when I try to open my wrapped native app on a mobile device: You can't get there from here error message

Your organization might have a **Require Approved Client App** conditional access policy that restricts access to mobile apps. To fix this issue, see Edit your conditional access policy.

Can I create B2C mobile apps with Power Apps?

No. Power Apps is a platform for business applications and uses Microsoft Entra authentication. The wrap feature wraps existing canvas apps for the same set of end users.

Do I need a premium license to package the app?

No, you don't need a premium license for wrap. However, if your APK uses certain connectors, you will need to pay according to the requirements of those connectors.

Can I delete wrap projects?

No, wrap projects can't be deleted but you can delete the project APK files from your storage location.

How can I ensure that customers outside the home tenant can't access the app even

if I enable multitenant access during registration?

Users outside the home tenant can't access wrap projects unless both the Azure admin and the maker explicitly share the app with the user. The app ensures proper data separation across organizations by embedding the tenant identifier in the authentication token.

Can we allow users from outside home tenant to access wrap projects?

The Wrap App supports multi-tenancy. To enable this feature, the Azure admin must:

1. Add users from outside the home tenant as guests in Azure.
2. Include these guests in the Azure Conditional Access process.
3. The maker must share the app with these guest IDs within Power Apps.

Why am I not able to Sign into my wrapped Application?

1. Verify if the user has access to the PowerApps application. More information: Share a canvas app with your organization
2. If access is confirmed, check the app's Azure conditional policies. Admin access is required to review these policies. More information: View applied Conditional Access details in the Microsoft Entra activity logs
3. To troubleshoot sign-in errors, copy the correlation ID from the mobile screen where the sign-in is failing and refer How to troubleshoot Microsoft Entra sign-in errors to understand the error and the failing policies.
4. Check Microsoft Entra authentication and authorization error codes.
5. One common error code is 50194 (tag- 9n555). During the app registration stage, the supported account type isn't configured for multi-tenant. Configure it to support multi-tenant.

When do you need to re-wrap?

- If you're an existing customer who has already created and successfully distributed a wrapped app, you don't need to go through the wrapping process again. To ensure users

receive the latest published changes, follow the steps mentioned in Power Apps for Makers: Get App Versions.

- For new customers, we recommend re-wrapping if you have completed all the registration steps mentioned in Use the wrap wizard to build your mobile app: Register app and have either manually or automatically completed the sign-in process mentioned in Use the wrap wizard to build your mobile app: Sign your mobile app package manually.
- To ensure platform changes benefit end users, we recommend re-wrapping monthly and distributing the updated app to users.

How to find your Azure admin?

- Go to the Microsoft Entra roles and administration portal. More information: Microsoft Entra admin center
- Search for *global administration* in the search bar.
- Select the **Active assignments** tab.
- Review the list of owners for the active assignments.

Why is Azure key vault a mandatory step?

The Azure key vault securely stores the access key needed for Azure blob storage.

Do you need to pay for Azure key vault & Azure blob storage?

Yes. See Key Vault pricing .

What should I do if I don't have access to the resource group while creating the key vault?

Contact your Azure tenant admin to create a key vault for you or request permission.

How can Entra B2B guest users access the wrapped apps?

1. Add users from outside the home tenant as guests in Azure. See Quickstart: Add a guest user and send an invitation.
2. Include these guests in the Azure Conditional Access process. Share the app with these guest IDs in Power Apps. See Share a canvas app with your organization.
3. Replace the app ID and the user account in the following link: `ms-mobile-apps:///providers/Microsoft.PowerApps/apps/<appID>?tenantId=<tenantID>`
4. Share the link with your client.
5. Install the APK on Android and the IPA on iOS.
6. Open the weblink and enter the external IDs to sign in.

See also

Troubleshoot issues with the wrap feature in Power Apps

Troubleshoot common issues when using the wrap feature

08/04/2025

This guide provides solutions to common issues you might encounter when using the wrap feature in Microsoft Power Apps.

Issue 1: Wrap build fails

If your wrap build fails, try the following actions:

Verify image formats

Ensure that all images in your wrap project are in PNG format. Using other formats can cause the build to fail. Use an image converter to convert your images to `.png` format.

> **Important**
>
> Renaming a file extension to `.png` doesn't convert the image to PNG format.

Check Azure key vault setup

- Ensure you create an Azure service principal and assign the correct role.
- For more information, see Azure key vault for wrap in Power Apps.

Your key vault must contain:

- **For iOS**: two tags, one certificate, and one secret.
- **For Android**: one tag and one certificate.

Issue 2: Wrap button is disabled

Confirm you have edit permissions for the app and try again. For a list of requirements, see Permissions and access requirements for wrap.

Issue 3: Unable to save your wrap project or trigger a build

Update to the latest version of the wrap solution and try again.

Issue 4: Unable to install a wrapped mobile app

Ensure your app is properly signed by configuring a key vault during the build process or by signing the app manually.

For more information, see:

- Set up Key vault for automated signing
- Code sign for iOS
- Code sign for Android

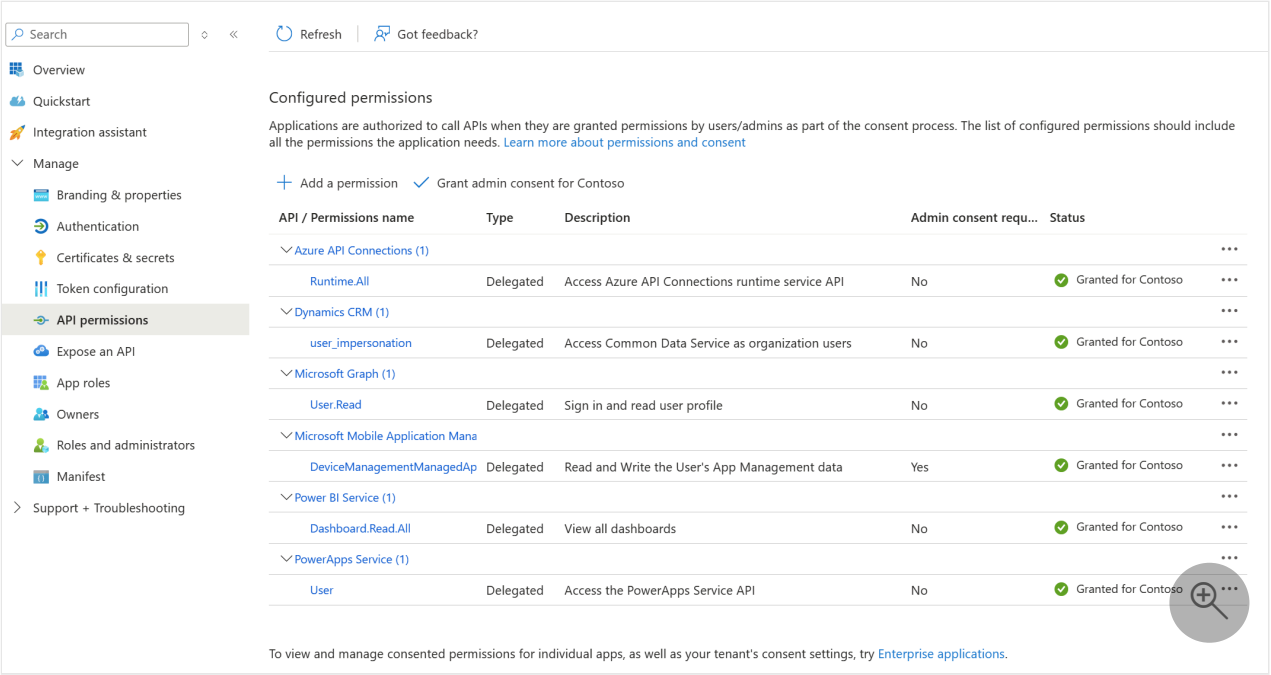
Additionally, verify your device meets the minimum requirements.

Issue 5: Can't sign in or see data in a wrapped app

If you can't sign in or see data in your wrapped app, try the following actions:

Verify API permissions and access

- Ensure all required API permissions are configured and that admin permissions are granted.



- Ensure the `Add-AdminAllowedThirdPartyApps` script runs successfully.
  For more information, see Allow registered apps in your environment.

Check account types and redirect URIs

- Verify the Microsoft Entra app type is set to **Multitenant** and that the supported account type is **Accounts in any organizational directory (Any Microsoft Entra ID tenant)**.
- Configure proper redirect URIs for iOS and Android:
  - For Android, confirm the hash is correct.
  - For more information, see Configure platform settings and redirect URIs.

Issue 6: Fail to sign in to a wrapped app

1. Ensure the user has access to the app. For more information, see Share a canvas app with your organization.
2. If the user has app access but still can't sign in, check the Conditional Access policies in the Microsoft Entra admin center.
3. To troubleshoot sign-in errors, copy the correlation ID from the mobile screen where the sign-in failed and refer to How to troubleshoot Microsoft Entra sign-in errors to understand the error and the failed policies.
4. Check Microsoft Entra authentication and authorization error codes.

Other issues

If your issue isn't covered here, or if the preceding steps don't resolve your problem, search for more support resources or contact Microsoft support with detailed steps to reproduce the problem.

Collect diagnostic information

For sign-in issues, you can collect session details and include them when you contact Microsoft support:

- For the wrap wizard: On the sign-in screen, tap the gear icon in the upper-right corner and select **Session Details**.
- For mobile devices: After opening the app, press and hold the screen, and then select **Session Details**.

Use code view for canvas app controls

Article • 05/21/2025

Use the code view feature to look at the underlying code for each control on your app's screens. The code uses an improved format that's a subset of YAML.

With code view you can:

- View the code for each control on your screen.
- Copy the code for a control and share it outside of Power Apps Studio.
- Paste the code for a control and create a new control based on it.

View, copy, and paste code

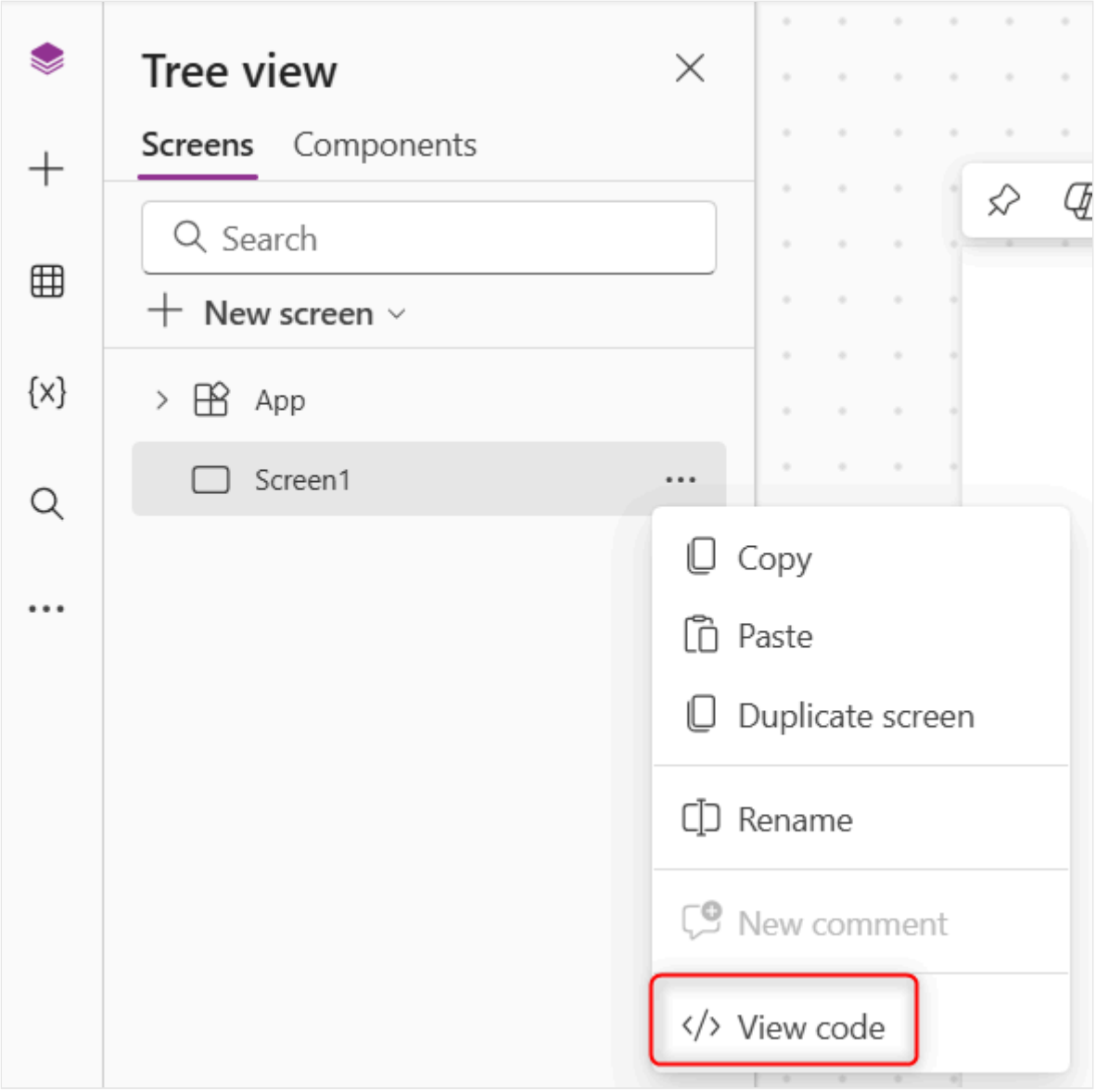
Code view shows the code for the selected control and all its child controls.

Code view lets you copy and paste code to any screen in your app or outside of Power Apps to duplicate the control. This feature is helpful when controls are highly customized and you want to share or duplicate them.

View code

1. Open your app for editing in Power Apps Studio.

2. Right-click the control in the tree view or on the screen, and then select **View code**.



> **Tip**
>
> Use the keyboard shortcut `Ctrl+F` to find a specific string in the code.

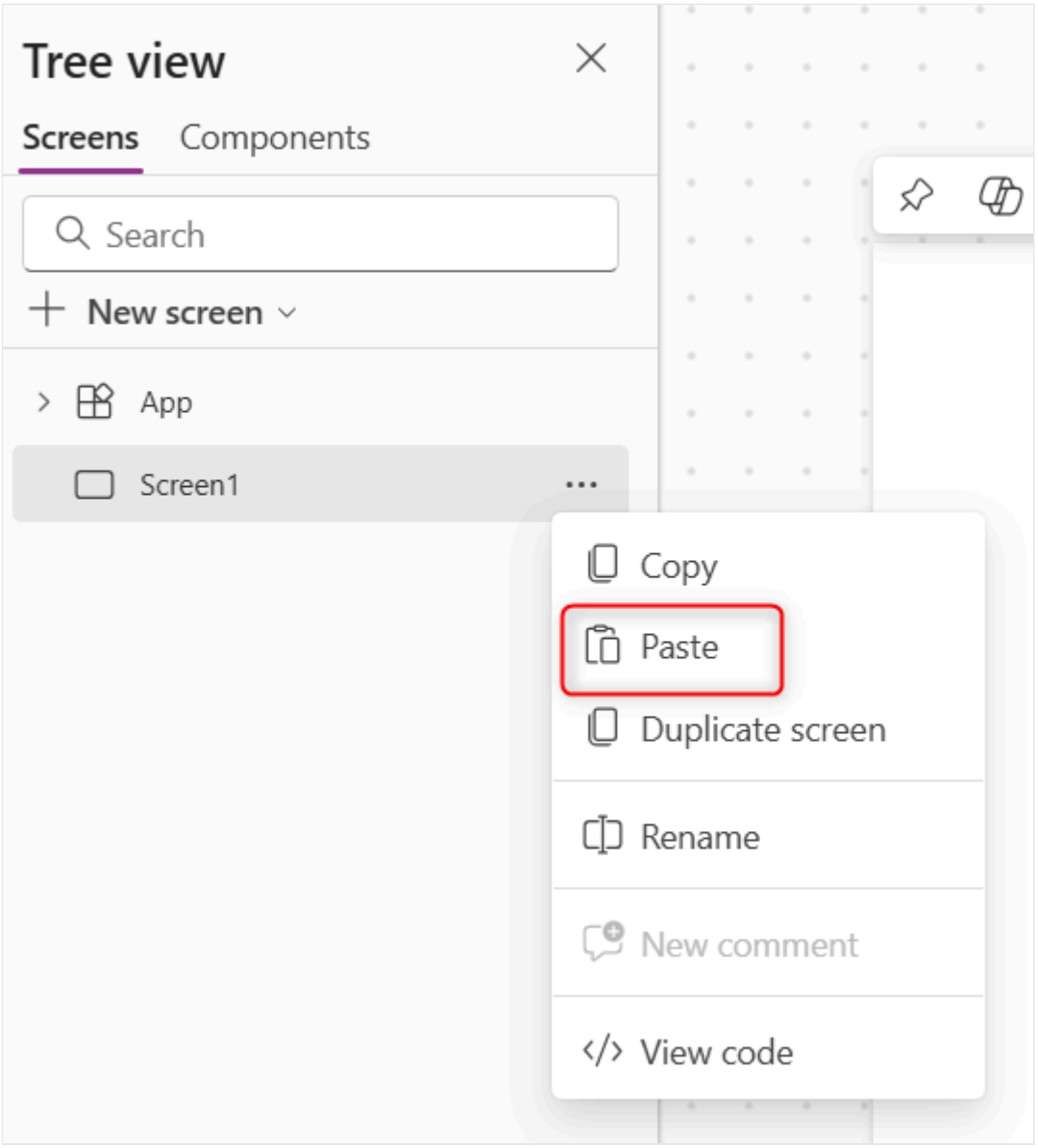
Copy and paste code

1. To copy the code of a selected control:

   - Use the menu to copy.
   - Use the shortcut `Ctrl` + `C`.
   - When viewing your code in the code view popup, select **Copy code**.

2. To add a new control from the code:

   - Use the menu to paste.
   - Use the shortcut `Ctrl` + `V`.



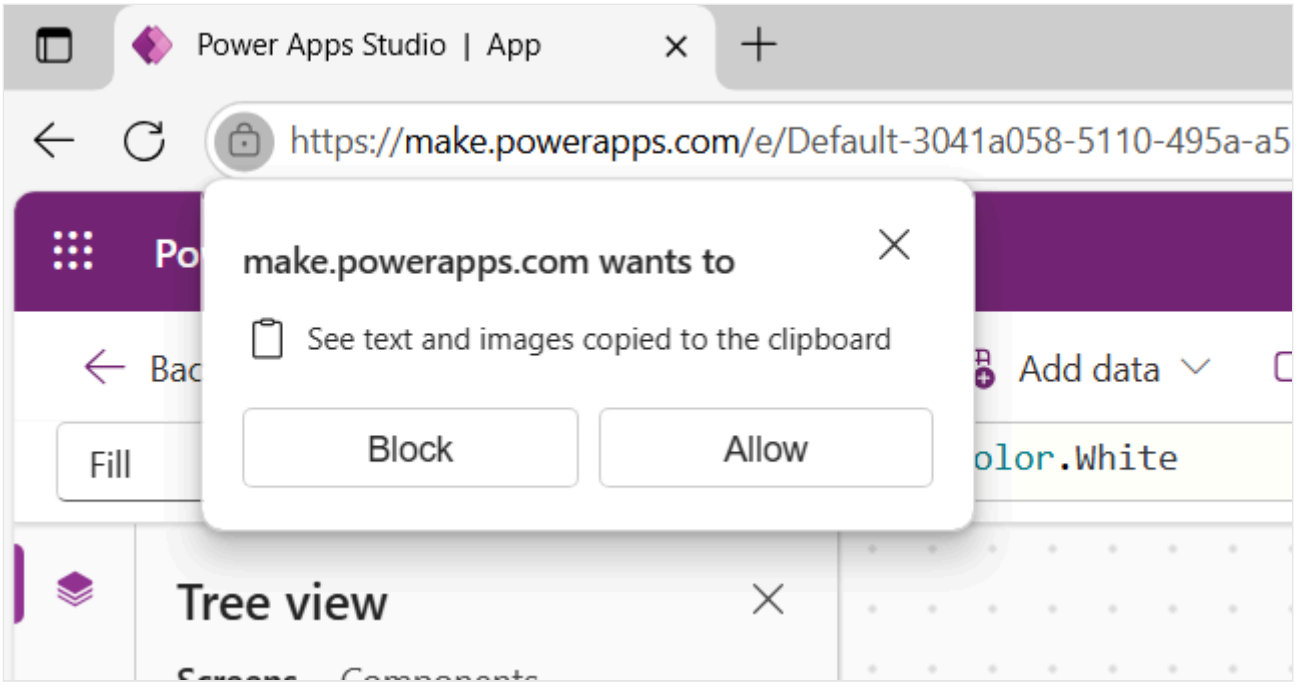
Important

Use the YAML format that Power Apps Studio generates. The code is validated before the new control is created.

Browser clipboard access required

To paste code into Power Apps Studio using code view, your browser must have access to the clipboard. If you encounter issues pasting code, check your browser settings and ensure clipboard permissions are enabled for the site.

The first time you paste code, your browser will prompt you to grant clipboard access. Approve this request to enable pasting functionality in Power Apps Studio.



> **Tip**
>
> If pasting doesn't work, check if clipboard access to Power Apps is blocked in your browser. In Microsoft Edge, you can manually add https://make.powerapps.com to the allowed sites for clipboard access in your browser settings.

Known limitations

- You can't copy and paste or view the code for the **App Object**.
- You can't edit the code in the code view.

Related information

- Get started with formulas in canvas apps
- Use Find and Replace in the formula bar
- Formula reference - canvas apps
- Power Fx YAML formula grammar
- Understand app functionality through code view (video)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

View source code files for canvas apps

10/23/2025

Use the source code for a canvas app to review changes made by makers in Power Apps Studio. The generated canvas app YAML code is read-only and can't be modified. Any changes to the file are ignored and might be lost.

> - The YAML schema is in active development, and the content might be incomplete.
> - The current static schema for ***.pa.yaml** files is available here.

Power Fx and YAML are the languages used for Power Apps source code. YAML is valued for its human-readable format and benefits from a wide array of editors, tools, and libraries for manipulation.

Access source code files

You have access to the source code files if you're using the Power Platform Git Integration, eliminating the need for **.msapp** files.

The Power Apps Studio creates the source code for canvas apps, which is stored as *.pa.yaml files within the **.msapp** file. The **.msapp** file is a binary file that contains a collection of files, including the source code.

> **Tip**
>
> To effectively use application lifecycle management (ALM), it's recommended to use solutions. Canvas apps package doesn't support ALM and should only be used for basic import and export capabilities when Dataverse isn't available.

You can also get the source code files either from the **.msapp** file or by using Power Platform CLI.

The ***.pa.yaml** files can be found in the **\Src** folder of the extracted **.msapp**.

> **Important**
>
> Only files located in the **\Src** directory of the extracted **.msapp** are intended for use with source control. The JSON files in the **.msapp** shouldn't be used as source code because those aren't stable between save and load cycles.

Download and Extract the pa.yaml files using Power Platform CLI

Connect to your environment using Power Platform CLI and then use the following commands:

1. To list the canvas apps in your current environment, use the command: pac canvas list.
2. To extract the source code files, use the command: pac canvas download with the parameter -d.

Extract source code files from a .msapp file

To extract source code files from a .msapp file, you have two options:

1. Manually unzip the **.msapp** file.
2. Alternatively, you can use the following command:

```
PowerShell

 Expand-Archive -Path "C:\path\to\yourFile.msapp" -DestinationPath
"C:\path\to\destination"
```

File structure for a .msapp file

For older **.msapp** files without the \src folder, follow these steps to generate source code files:

1. Import and resave and download the new **.msapp** in Power Apps Studio.
2. After that, you can proceed to extract the source code files.

In the **\src** folder, find the following files and folders:

- **App.pa.yaml**: Represents the App.
- **[screen Name].pa.yaml**: One file for each screen, representing the screen.
- **\Component**: A folder containing one file for each component, represented as [component Name].pa.yaml.

Only ***.pa.yaml** files within the **\src** folder can be used as source code. Any other file shouldn't be used as source code.

Power Apps YAML schema versions

Currently, the only active schema version of Power Apps source code is Source Code (*.pa.yaml). The following table shows the other historical formats:

Expand table

Format name	File extension	Description
Experimental	*.fx.yaml	Retired. Version used by the experimental Power Apps Git version control and pac canvas unpack—no longer in development.
Early preview	-	Retired. The version used by code view, copy code, and paste code. There's no version information in this schema.
Source code	*.pa.yaml files	Active. Includes enhancements and version details for source control.

Experimental format (*.fx.yaml)

This schema is retired.

This schema describes an experimental format used by the Power Platform CLI to process and convert canvas apps into a source code format. This format isn't actively developed anymore.

You can't directly convert ***.fx.yaml** files to the new formats. To convert older apps, package the canvas app as a **.msapp** file and import it into Power Apps Studio.

Preview

This schema is retired.

This version was used by Code view during the preview period. It is designed to create canvas apps in Power Apps Studio, letting you easily copy and paste controls. In this version, the source code is used in Power Apps Studio instead of being converted, like the experimental format.

The format during preview was temporary and is no longer in use. You cannot paste code from the preview format.

Here are the changes made from the experimental format:

1. **ZIndex property removal**: The ZIndex property is removed. Instead, a screen is represented as an array of controls. The order of controls determines their stacking order. Normal controls are ordered in ascending order, while responsive controls are ordered in descending order.
2. **JSON object representation replacement**: The "As" syntax is no longer used to define the control type. Instead, the left side of the control's name identifier remains unique. Only

properties that differ from the default values are serialized. Two new properties define the control type and default values:

- **Control**: Represents the control type in YAML.
- **Variant**: Identifies a variant of a control type, which might alter default property values, add or remove properties, or modify the behavior or layout of the control.

These properties are used for instantiating controls and don't accept Power Fx expressions.

Source code (*.pa.yaml)

> **Important**
>
> - The YAML source code for canvas apps is actively being developed. The content may be incomplete and subject to change.
> - The **.pa.yaml** files are read-only and should only be used to review changes made in Power Apps Studio. These files are not used when an app is loading.
> - External editing, merging, and conflict resolution is supported only in Power Platform Git Integration.

This schema is designed for source control purposes and allows the use of a single YAML file.

Updates from the previous format include:

1. Grouping of Top-Level Node Elements:
    - The top-level keywords "App" and "Screens" are examples of how elements are now grouped in a top-level node.
2. Control Version Specification:
    - You can now specify the version of a control using the Control Keyword followed by the @ operator. If no version is specified, the most current version of the control is used. The version is used to deserialize the properties of the source code. Runtime version of the control is defined by the authoring version.
3. Simplified Variant Names:
    - Not all controls require a variant.
4. Consistent ZIndex for All Controls:

- All controls now use an ascending order to determine the ZIndex value, starting from 1. This behavior aligns with the CSS 2 spec.

Disconnect Git version control to edit canvas apps

Article • 05/14/2025

Git version control for editing apps has been removed and is no longer supported.

If your app currently uses this feature, follow these steps to disconnect it:

1. Open your canvas app for editing in Power Apps Studio. On the command bar, select **Settings**.
2. Select **Git version control** and then select **Disconnect**.

If you are using PASopa, download your app from your Git provider and pack the .zip file to an .msapp file. Once your .msapp file is uploaded to Power Apps, follow the steps above to disconnect.

> **Note**
>
> Git version control is replaced with the source control integration feature. More information: Source code integration

Canvas apps for enterprise developers, partners, and ISVs

As a developer, you can extend canvas apps in Power Apps, enabling even more powerful solutions for organizations and customers.

Canvas apps for enterprise developers

As an enterprise developer, empower your organization to build robust, tailored solutions on Power Apps:

- **Build custom connectors**: Develop custom connectors to connect to your organization's data and web services. Learn more
- **Create canvas apps using Power Platform CLI**: Use Microsoft Power Platform CLI to create, pack (preview), and unpack (preview) canvas apps using a custom connector. More information: Power Platform CLI - **pac canvas**
- **Build Azure Functions**: Craft Azure Functions to extend apps with custom server-side logic. Learn more
- **Embed apps**: Embed apps directly into your website experiences to create integrated solutions, surfacing apps where people in your organization already do their work. Learn more
- **Build offline-capable apps**: Develop offline-capable apps so your users are productive whether they are online or offline. Learn more

Canvas apps for ISVs and Microsoft partners

As a Microsoft partner or Independent Software Vendor (ISV), accelerate customer adoption by extending your products to integrate with your customers' data and business processes:

- **Build and certify custom connectors**: Bring your product into the Microsoft cloud by building a connector that enables Power Apps to talk to your service. Learn more
- **List your solution on Microsoft Marketplace**: Generate new leads for your business by building an app with Power Apps and publishing it to Microsoft Marketplace for new customers to find. Learn more
- **Get started today for free**: Sign-up for the Power Apps Developer Plan for free and start building apps in a dedicated developer environment. Learn more

Last updated on 01/16/2026

Custom connectors for canvas apps

Article • 02/08/2023

Without writing any code, you can build workflows and canvas apps with Azure Logic Apps, Power Automate, and Power Apps. To help you integrate your data and business processes, these services offer 180+ connectors - for Microsoft services and products, as well as other services, such as GitHub, Salesforce, and Twitter.

Sometimes though, you might want to call APIs, services, and systems that aren't available as pre-built connectors. To support more tailored scenarios, you can build *custom connectors* with their own triggers and actions. We have a complete set of basic and advanced tutorials for custom connectors on the Connectors documentation site. We recommend that you start with the custom connector overview, but you can also go straight to the following topics for details on a specific area:

- Create a custom connector from an OpenAPI definition
- Create a custom connector from a Postman collection
- Create a custom connector from scratch
- Use a custom connector from Power Apps
- Share custom connectors in your organization
- Submit your connectors for Microsoft certification
- Custom connector FAQ

Feedback

Was this page helpful? Yes No

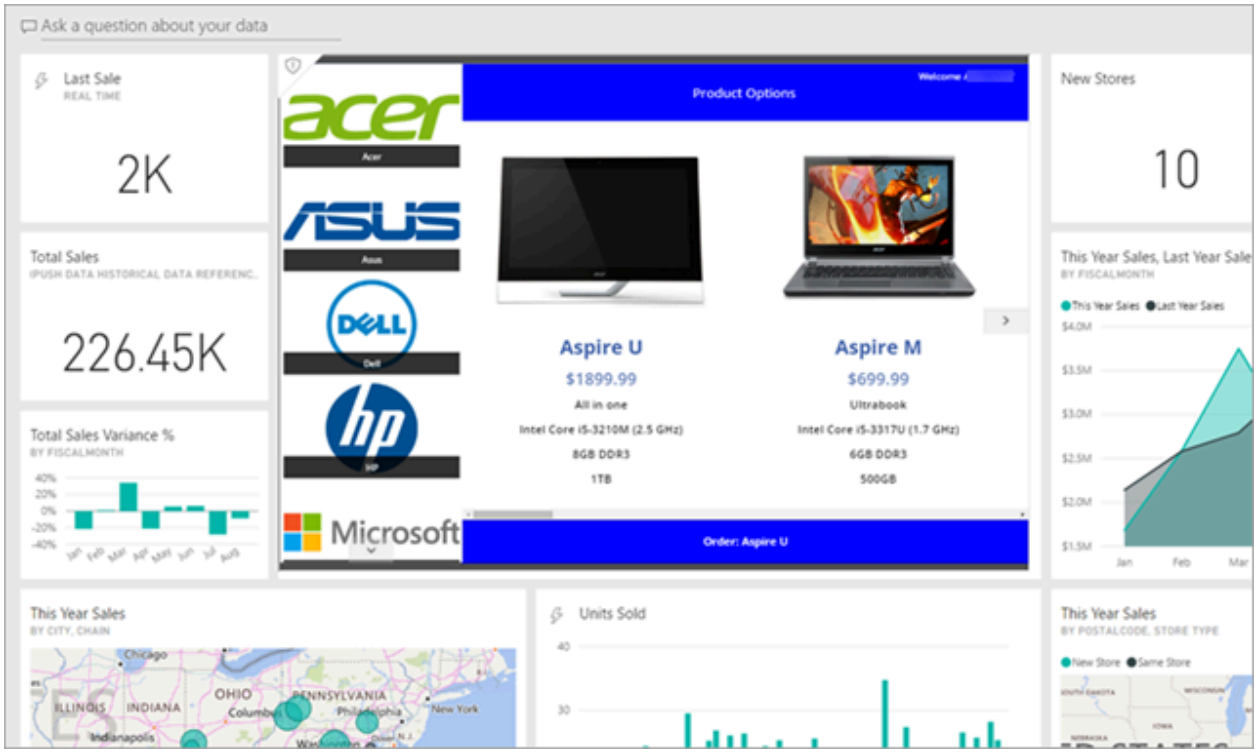
Provide product feedback

Integrate canvas apps into websites and other services

Article • 03/12/2025

The apps that you build are often most useful when they're available right where people do their work. By embedding canvas apps in an iframe, you can integrate those apps into websites and other services, such as Power BI or SharePoint.

In this topic, we'll show you how to set parameters for app embedding; then we'll embed our Asset Ordering app in a website.



> **Note**
>
> - Only Power Apps users in the same tenant can access the embedded app.
> - Embedding canvas apps in any native desktop application isn't supported, which includes Andrioid and iOS. This excludes first-party integrations such as Power Apps in Teams.

You can also integrate canvas apps into SharePoint Online without using an iframe. More information: Use the Power Apps web part.

Set URI parameters for your app

If you have an app you want to embed, the first step is to set parameters for the Uniform Resource Identifier (URI), so that the iframe knows where to find the app. The URI is in the following form:

```
https://apps.powerapps.com/play/[AppID]?source=iframe
```

For GCC users

```
https://apps.gov.powerapps.us/play/[AppID]?source=iframe
```

> **Important**
>
> As of August 2019, the URI format has changed from https://web.powerapps.com/webplayer to https://apps.powerapps.com/play. Please update any embedded iframes to use the new URI format. References to the previous format will be redirected to the new URI to ensure compatibility.
>
> Previous format:
>
> https://web.powerapps.com/webplayer/iframeapp?source=iframe&appId=/providers/Microsoft.PowerApps/apps/[AppID]

The only thing you have to do is substitute the ID of your app for [AppID] in the URI (including '[' & ']'). We'll show you how to get that value shortly, but first here are all the parameters available in the URI:

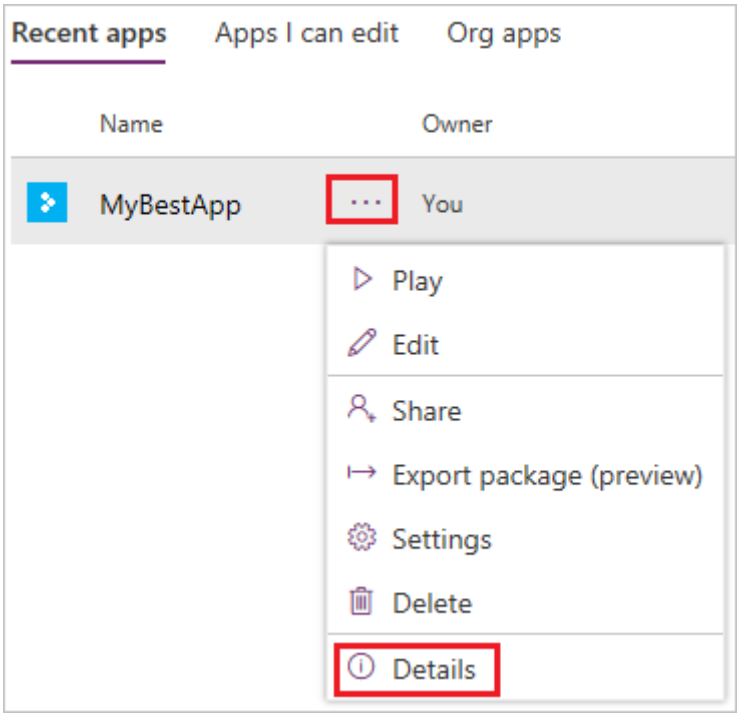
- **[appID]** - It provides the ID of the app to run.
- **tenantid** - is an optional parameter to support guest access and determines which tenant to open the app from.
- **screenColor** - is used to provide a better app loading experience for your users. This parameter is in the format RGBA (red value, green value, blue value, alpha) and controls the screen color while the app loads. It is best to set it to the same color as your app's icon.
- **source** - does not affect the app, but we suggest you add a descriptive name to refer to the source of the embedding.
- Lastly, you can add any custom parameters you want using the Param() function, and those values can be consumed by your app. They are added to the end of the URI, such as `[AppID]?source=iframe&param1=value1&param2=value2`. These

parameters are read only during launch of the app. If you need to change them, you must relaunch the app. Note that only the first item after [appid] should have a "?"; after that use the "&" as illustrated here.

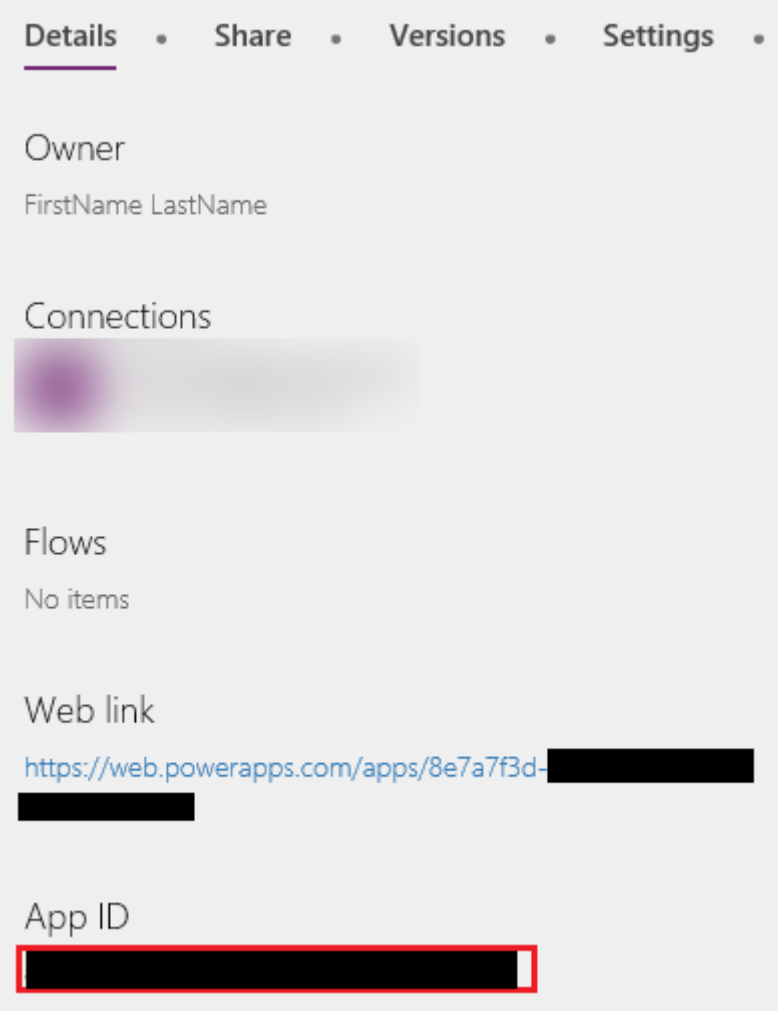
Get the App ID

The app ID is available on powerapps.com. For the app you want to embed:

1. In powerapps.com, on the **Apps** tab, click or tap the ellipsis (. . .), then **Details**.



2. Copy the **App ID**.



3. Substitute the `[AppID]` value in the URI. For our Asset Ordering app, the URI looks like this:

```
https://apps.powerapps.com/play/76897698-91a8-b2de-756e-fe2774f114f2?
source=iframe
```

You may need to allow pop-ups in browser when you embed an app in your website that uses Launch() function to launch a webpage or an app.

Embed your app in a website

Embedding your app is now as simple as adding the iframe to the HTML code for your site (or any other service that supports iframes, such as Power BI or SharePoint):

HTML

```
<iframe width="[W]" height="[H]"
src="https://apps.powerapps.com/play/[AppID]?
```

```
source=website&screenColor=rgba(165,34,55,1)" allow="geolocation;
microphone; camera"/>
```

Specify values for the iframe width and height, and substitute the ID of your app for `[AppID]`.

> **Note**
>
> Include `allow="geolocation; microphone; camera"` in your iframe HTML code to allow your apps to use these capabilities on Google Chrome.

The following image shows the Asset Ordering app embedded in a Contoso sample website.



Keep the following points in mind for authenticating users of your app:

- If your website uses Microsoft Entra ID based authentication, no additional sign-in is required.
- If your website uses any other sign-in mechanism or is not authenticated, your users see a sign-in prompt on the iframe. After they sign-in, they will be able to run the app as long as the author of the app has shared it with them.

As you can see, embedding apps is simple and powerful. Embedding enables you to bring apps right to the places you and your customers work – websites, Power BI dashboards, SharePoint pages, and more.

Feedback

Was this page helpful? Yes No

Provide product feedback

Develop offline-capable canvas apps

Article • 03/18/2024

Mobile users often need to be productive even when they have limited or no connectivity. When you build a canvas app, you can perform these tasks:

- Open Power Apps Mobile and run apps when the device is not connected to the network.
- Determine when an app is offline, online, or in a metered connection by using the Connection signal object.
- Synchronize data between your device and the server.

If your app connects to Dataverse, offline support is **built-in**. Power Apps will allows you to download data to your device and upload changes back to Dataverse. If your app does not use Dataverse, you can use collections and leverage the **LoadData** and **SaveData** functions for basic storage when offline.

> **Note**
>
> - Offline capability for canvas apps is available while running the apps using the native Power Apps Mobile players on iOS, Android, and Windows.
> - Canvas apps running in web browsers can't run offline, even when using a web browser on a mobile device.
> - Canvas apps in Teams are limited to 1 MB of data through the LoadData and SaveData functions—useful for a small number of text strings, numbers, and dates. Use of images or other media is inappropriate for this limit. More information: LoadData and SaveData function reference

Enable offline support in a Dataverse-based canvas app

For Microsoft Dataverse-based canvas apps, you must use the offline capability built-in with the offline-first experience. For more information, see Set up mobile offline for canvas apps and Working with canvas apps offline. With the flip of a switch, your app can work with Dataverse data wherever you are, with or without a network connection. Just build your app with standard Power Fx formulas and the offline feature handles all the complexity for you.

Use LoadData and SaveData for all other connectors

This section includes an example using Twitter data. An even simpler example that doesn't require a connection is included in the **LoadData** and **SaveData** function reference.

Watch this video to learn how to create offline enabled canvas apps which don't use Dataverse data:
https://learn-video.azurefd.net/vod/player?id=60fd9b60-80a9-445b-bd4e-b7f4f068431e&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Foffline-apps

Limitations

LoadData and **SaveData** combine to form a simple mechanism to store small amounts of data on a local device. By using these functions, you can add simple offline capabilities to your app.

These functions are limited by the amount of available app memory because they operate on an in-memory collection. Available memory can vary depending on the device, the operating system, the memory that Power Apps Mobile uses, and the complexity of the app in terms of screens and controls. If you store more than a few megabytes of data, test your app with expected scenarios on the devices on which you expect it to run. You'll generally have 30-70 megabytes of available memory.

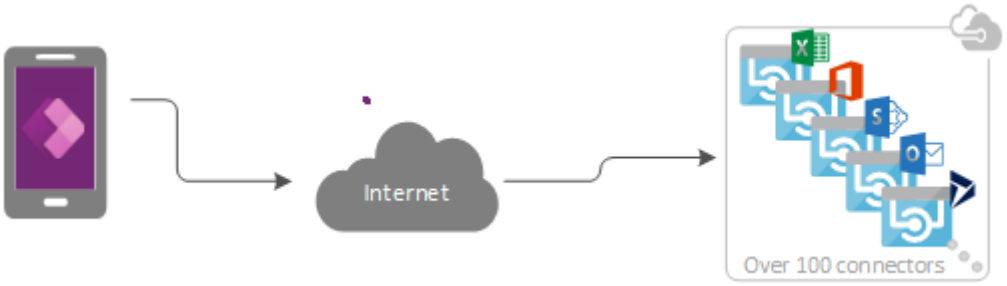
The functions also don't automatically resolve merge conflicts when a device comes online. Configuration on what data is saved and how to handle reconnection is up to the maker when writing expressions.

For updates on offline capabilities, return to this topic, and subscribe to the Power Apps blog.

Overview

When you design offline scenarios, you should first consider how your apps work with data. Apps in Power Apps primarily access data through a set of connectors that the platform provides, such as SharePoint, Office 365, and Microsoft Dataverse. You can also build custom connectors that enable apps to access any service that provides a RESTful endpoint. This could be a Web API or a service such as Azure Functions. All these

connectors use HTTPS over the Internet, which means your users must be online for them to access data and any other capabilities that a service offers.



Handling offline data

In Power Apps, you can filter, search, sort, aggregate, and manipulate data in a consistent way regardless of the data source. Sources range from in-memory collections in the app to lists created using Microsoft Lists to SQL databases and Dataverse. Because of this consistency, you can easily retarget an app to use a different data source. More importantly for offline scenarios, you can use local collections for data management with almost no changes to an app's logic. In fact, local collections are the primary mechanism for handling offline data.

Build an offline app

To keep the focus on the offline aspects of app development, this topic illustrates a simple scenario focused around Twitter. You'll build an app that enables you to read Twitter posts and submit tweets while being offline. When the app comes online, the app posts tweets and reloads the local data.

At a high level, the app performs these tasks:

- When the user opens the app:
  - If the device is online, the app fetches data through the Twitter connector and populates a collection with that data.
  - If the device is offline, the app loads the data from a local cache file by using the LoadData function.
  - The user can submit tweets. If the app is online, it posts the tweets directly to Twitter and refreshes the local cache.
- Every five minutes while the app is online:
  - The app posts any tweets in the local cache.
  - The app refreshes the local cache and saves it by using the SaveData function.

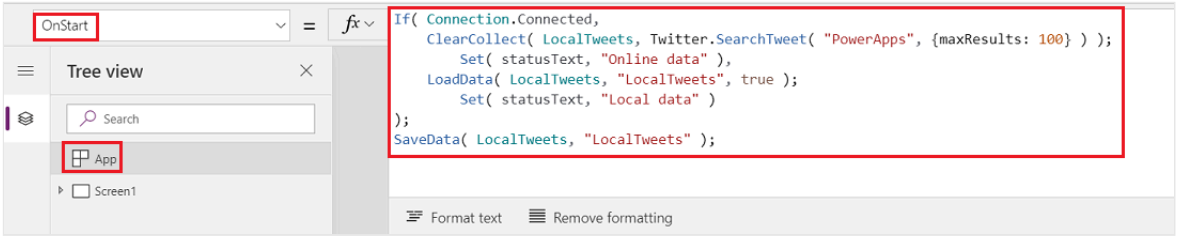
Step 1: Add Twitter to a blank phone app

1. Create a blank canvas app with **Phone** layout.
2. On the **View** tab, select **Data sources**.
3. In the **Data** pane, select **Add data source**.
4. Select **New Connection** > **Twitter** > **Create**.
5. Enter your credentials, create the connection, and then close the **Data** pane.

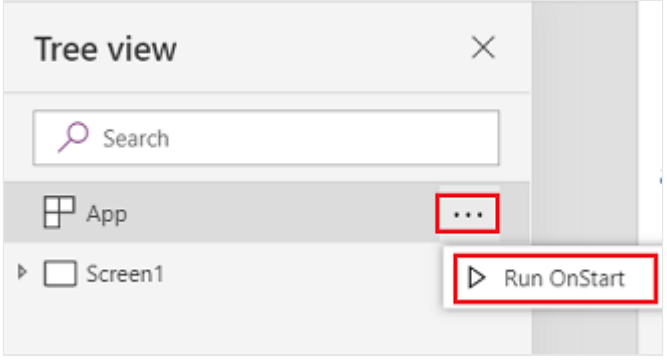
Step 2: Collect existing tweets

1. In the **Tree view** pane, select **App**, and then set its **OnStart** property to this formula:

```
Power Fx

If( Connection.Connected,
    ClearCollect( LocalTweets, Twitter.SearchTweet( "PowerApps",
{maxResults: 10} ) );
        Set( statusText, "Online data" ),
    LoadData( LocalTweets, "LocalTweets", true );
        Set( statusText, "Local data" )
);
SaveData( LocalTweets, "LocalTweets" );
```



2. In the **Tree view** pane, select the ellipsis menu for the **App** object, and then select **Run OnStart** to run that formula.



> **Note**

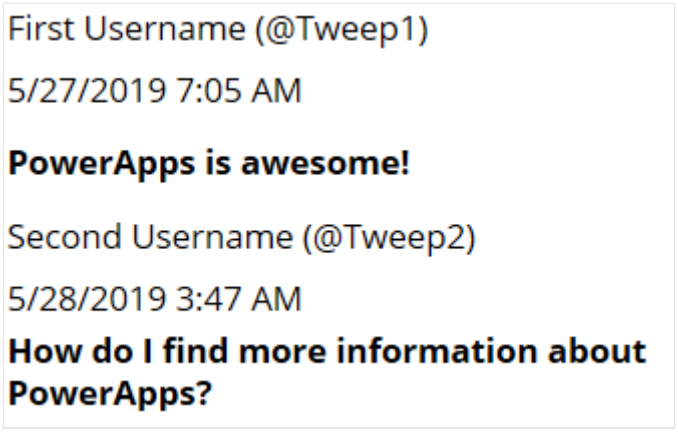
> The **LoadData** and **SaveData** functions might show an error in Power Apps Studio because browsers don't support them. However, they'll perform normally after you deploy this app to a device.

This formula checks whether the device is online:

- If the device is online, the formula loads up to 10 tweets with the search term "PowerApps" into a **LocalTweets** collection.
- If the device is offline, the formula loads the local cache from a file called "LocalTweets" if it's available.

Step 3: Show tweets in a gallery

1. On the **Insert** tab, select **Gallery** > **Blank flexible height**.
2. Set the **Items** property of the Gallery control to `LocalTweets`.
3. In the gallery template, add three Label controls, and set the **Text** property of each label to one of these values:
   - `ThisItem.UserDetails.FullName & " (@" & ThisItem.UserDetails.UserName & ")"`
   - `Text(DateTimeValue(ThisItem.CreatedAtIso), DateTimeFormat.ShortDateTime)`
   - `ThisItem.TweetText`
4. Make the text in the last label bold so that the gallery resembles this example.



Step 4: Show connection status

1. Under the gallery, insert a label, and then set its **Color** property to **Red**.
2. Set the newest label's **Text** property to this formula:

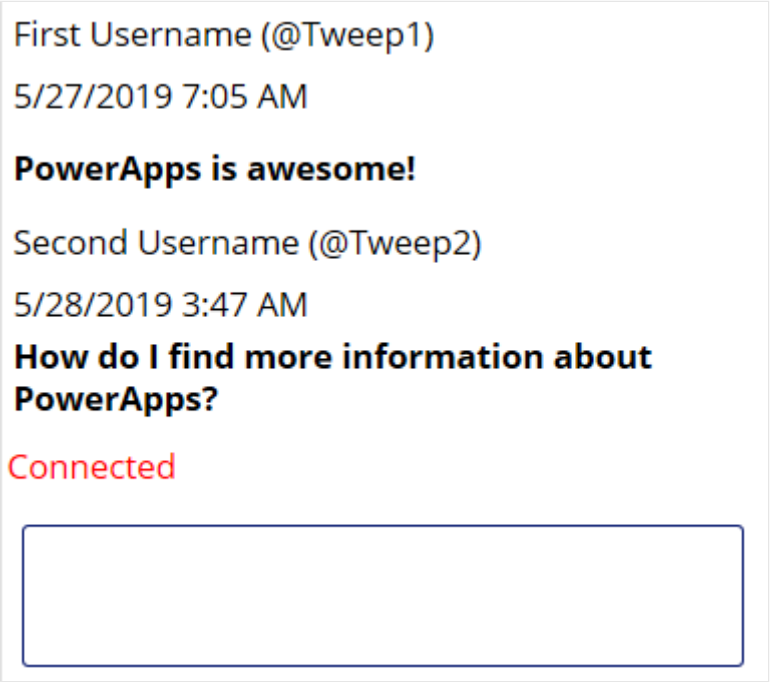
```
If( Connection.Connected, "Connected", "Offline" )
```

This formula determines whether the device is online. If it is, the label shows **Connected**; otherwise, it shows **Offline**.

Step 5: Add a box to compose tweets

1. Under the connection-status label, insert a Text input control, and rename it **NewTweetTextInput**.

2. Set the text-input box's **Default** property to `""`.



Step 6: Add a button to post the tweet

1. Under the text-input box, add a **Button** control, and set its **Text** property to this value:

   `"Tweet"`

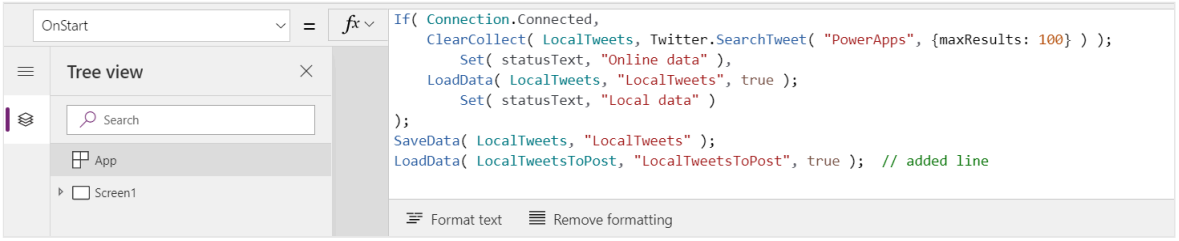
2. Set the button's **OnSelect** property to this formula:

Power Fx

```
If( Connection.Connected,
    Twitter.Tweet( "", {tweetText: NewTweetTextInput.Text} ),
    Collect( LocalTweetsToPost, {tweetText: NewTweetTextInput.Text} );
        SaveData( LocalTweetsToPost, "LocalTweetsToPost" )
);
Reset( NewTweetTextInput );
```

3. In the **OnStart** property for the **App**, add a line at the end of the formula:

```
Power Fx

If( Connection.Connected,
    ClearCollect( LocalTweets, Twitter.SearchTweet( "PowerApps",
{maxResults: 100} ) );
        Set( statusText, "Online data" ),
    LoadData( LocalTweets, "LocalTweets", true );
        Set( statusText, "Local data" )
);
SaveData( LocalTweets, "LocalTweets" );
LoadData( LocalTweetsToPost, "LocalTweetsToPost", true );  // added
line
```



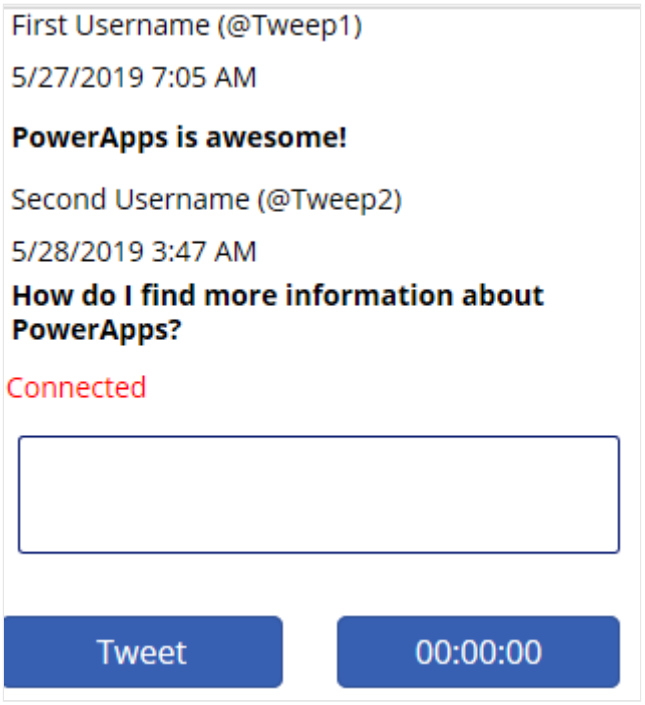
This formula determines whether the device is online:

- If the device is online, it posts the tweet immediately.
- If the device is offline, it captures the tweet in a **LocalTweetsToPost** collection and saves it to the device.

Then the formula resets the text in the text-input box.

Step 7: Check for new tweets

1. On the right side of the button, add a **Timer** control.



2. Set the timer's **Duration** property to **300000**.

3. Set the timer's **AutoStart** and **Repeat** properties to **true**.

4. Set the timer's **OnTimerEnd** to this formula:

```
Power Fx

If( Connection.Connected,
    ForAll( LocalTweetsToPost, Twitter.Tweet( "", {tweetText:
tweetText} ) );
    Clear( LocalTweetsToPost );
    ClearCollect( LocalTweets, Twitter.SearchTweet( "PowerApps",
{maxResults: 10} ) );
    SaveData( LocalTweets, "LocalTweets" );
)
```

This formula determines whether the device is online. If it is, the app tweets all the items in the **LocalTweetsToPost** collection and then clears the collection.

Test the app

1. Open the app using Power Apps Mobile on a mobile device that's connected to the Internet.

   Existing tweets appear in the gallery, and the status shows **Connected**.

2. Disconnect the device from the Internet by enabling the device's airplane mode and disabling wi-fi.

The status label shows that the app is **Offline**.

3. While the device is offline, write a tweet that includes **Power Apps**, and then select the **Tweet** button.

   The tweet is stored locally in the **LocalTweetsToPost** collection.

4. Reconnect the device to the Internet by disabling the device's airplane mode and enabling wi-fi.

   Within five minutes, the app posts the tweet, which appears in the gallery.

We hope this article gives you an idea of the capabilities that Power Apps has for building offline apps. As always, please provide feedback in our forum and share your examples of offline apps in the Power Apps community blog.

Feedback

Was this page helpful? Yes No

Provide product feedback

Canvas component overview

> **Note**
>
> - This section explains canvas components that encompass low-code UI extensibility capabilities. Professional developers can also use the Power Apps component framework to build code components.
> - You can also use canvas components in model-driven apps by using custom pages and the component library. For more information, see Add canvas components to a custom page in a model-driven app.

Components are reusable building blocks for canvas apps. App makers can create custom controls to use inside an app, or across apps by using a component library. Components can use advanced features such as custom properties and enable complex capabilities. This article introduces component concepts and some examples.

Components are useful when building larger apps that have similar control patterns. If you update a component definition inside the app, all instances in the app reflect your changes. Components also reduce duplication of effort by eliminating the need to copy and paste controls and improve performance. When you use a component library, components help create collaborative development and standardize the look and feel in an organization.

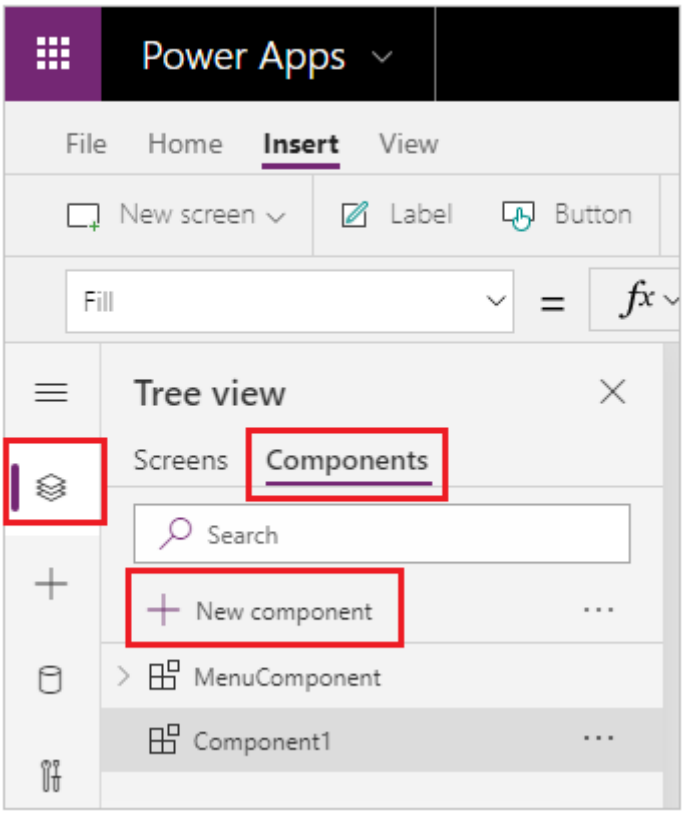
To learn how to use components in canvas apps, watch this video:

https://learn-video.azurefd.net/vod/player?id=7ec063b0-df8e-4cf8-b407-72ce6132115f&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fcanvas-apps%2Fcreate-component

Components in canvas apps

You can create a component from within an app as explained in this article, or by creating a new component inside a component library. Use a component library when you need to use components across multiple app screens. You can also copy existing components into an existing or a new component library.

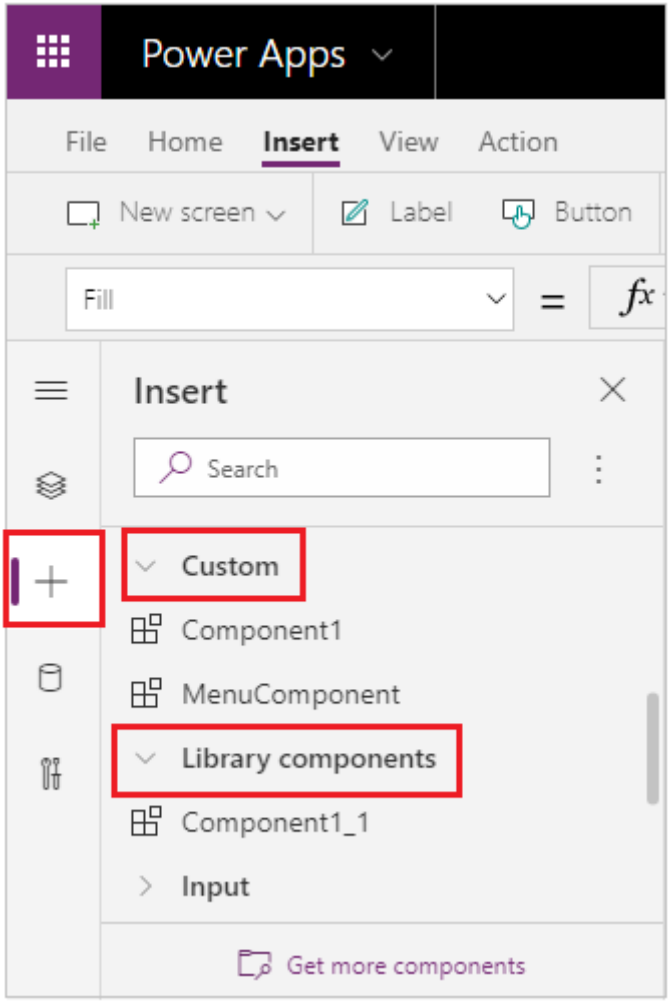
To create a component within an app, go to **Tree View**, select the **Components** tab, and then select **New component**:



Selecting **New component** opens an empty canvas. Add controls as part of the component definition on the canvas. If you edit a component in the canvas, you update instances of the same component in other app screens. Apps that reuse an already created component can also receive component updates after you publish component changes.

You can select a component from the list of existing components in the left pane after you select a screen. When you select a component, you insert an instance of that component onto the screen, just as you insert a control.

Components available inside the app appear under the **Custom** category in a list of components inside the tree view. Components imported from component libraries appear under the **Library components** category:



> **Note**
>
> The components discussed in this article are different from the Power Apps component framework that enables developers and makers to create code components for model-driven and canvas apps. For more information, see Power Apps component framework overview.

Custom properties

A component can receive input values and emit data if you create one or more custom properties. These scenarios are advanced and require you to understand formulas and binding contracts.

> **Note**
>
> An experimental feature for enhanced component properties provides even more options for properties, including functions and behavior functions. For more information, see Canvas component properties (experimental)

Input property is how a component receives data to be used in the component. Input properties appear in the **Properties** tab of the right-hand pane if an instance of the component is selected. You can configure input properties with expressions or formulas, just as you configure standard properties in other controls. Other controls have input properties, such as the **Default** property of a **Text input** control.

Output property is used to emit data or component state. For example, the **Selected** property on a **Gallery** control is an output property. When you create an output property, you can determine what other controls can refer to the component state.

The following walk-through further explains these concepts.

Create an example component

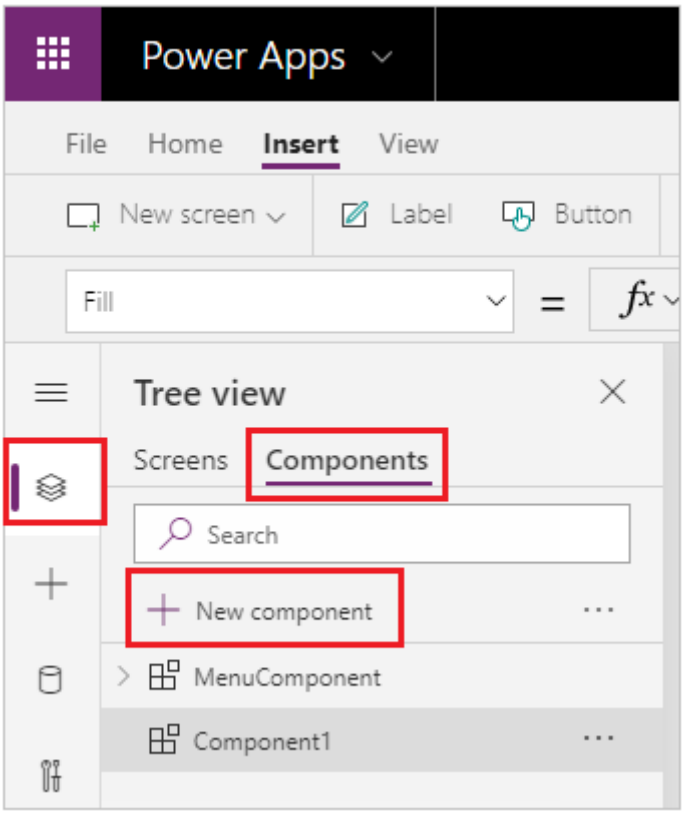
In this example, you'll create a menu component that resembles the following graphic. And you can change the text later to use it in multiple screens, apps, or both:



> **Note**
>
> We recommend that you use a component library when creating components for reuse. Updating components inside an app only makes the component updates available inside the app. When using a component library, you get prompted to update components if components inside a library are updated and published.

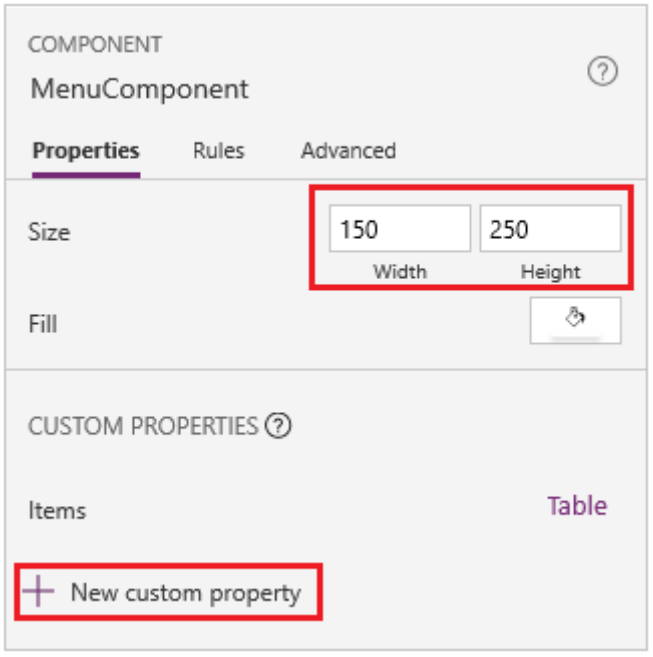
Create a new component

1. Create a blank canvas app.
2. In the **Tree View**, select **Components** and then select **New component** to create a new component.

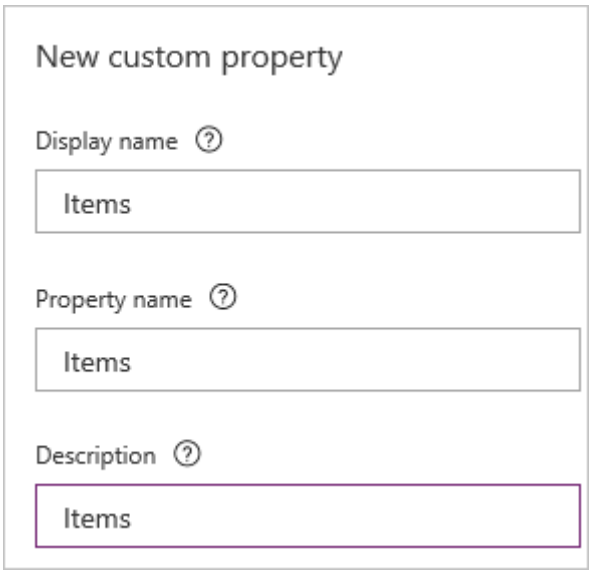


3. Select the new component in the left pane, select the ellipsis (...), and then select **Rename**. Type or paste the name as **MenuComponent**.

4. In the right-hand pane, set the component's width as **150** and its height as **250**, and then select **New custom property**. You can also set the height and width to any other value as appropriate.



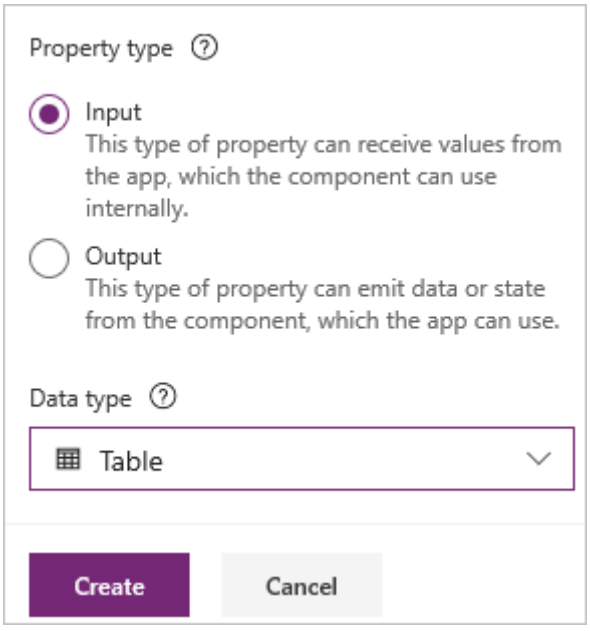
5. In the **Display name**, **Property name**, and **Description** boxes, type or paste text as *Items*.



Don't include spaces in the property name because you'll refer to the component by this name when you write a formula. For example, **ComponentName.PropertyName**.

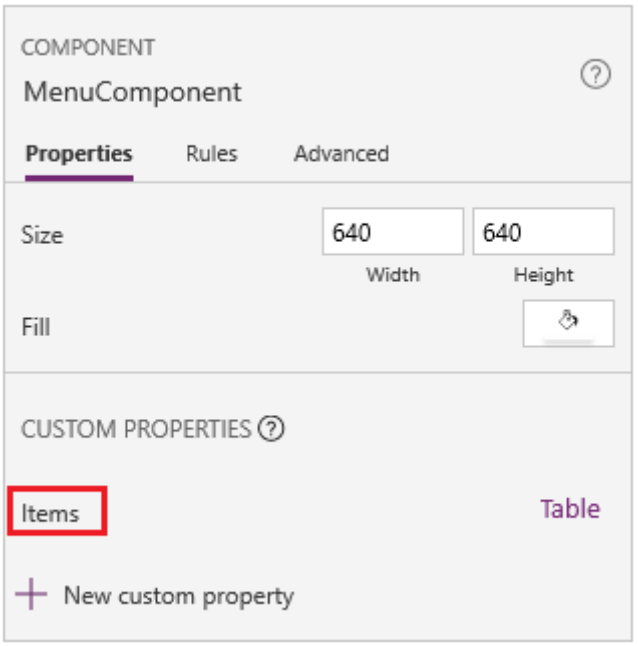
The display name appears on the **Properties** tab of the right-hand pane if you select the component. A descriptive display name helps you and other makers understand the purpose of this property. The **Description** appears in a tooltip if you hover over the display name of this property in the **Properties** tab.

6. In the **Data type** list, select **Table**, and then select **Create**.



The **Items** property is set to a default value based on the data type that you specified. You can set it to a value that suits your needs. If you specified a data type of **Table** or **Record**, you might want to change the value of the **Items** property to match the data schema that you want to input to the component. In this case, you'll change it to a list of strings.

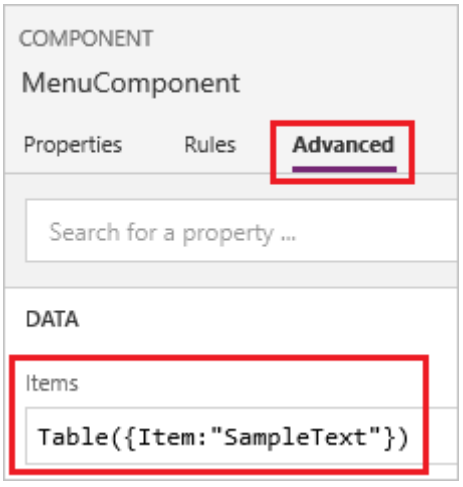
You can set the property's value in the formula bar if you select the name of the property on the **Properties** tab of the right-hand pane.



As the next graphic shows, you can also edit the property's value on the **Advanced** tab of the right-hand pane.

7. Set the component's **Items** property to this formula:

```
Power Fx
Table({Item:"SampleText"})
```



8. In the component, insert a blank vertical **Gallery** control and select **Layout** on the property pane as **Title**.

9. Make sure that the property list shows the **Items** property (as it does by default). Then set the value of that property to this expression:

```
Power Fx
MenuComponent.Items
```

This way, the **Items** property of the **Gallery** control reads and depends on the **Items** input property of the component.

10. (Optional) Set the **Gallery** control's **BorderThickness** property to **1** and its **TemplateSize** property to **50**. You can also update values for border thickness and template size to any other value as appropriate.

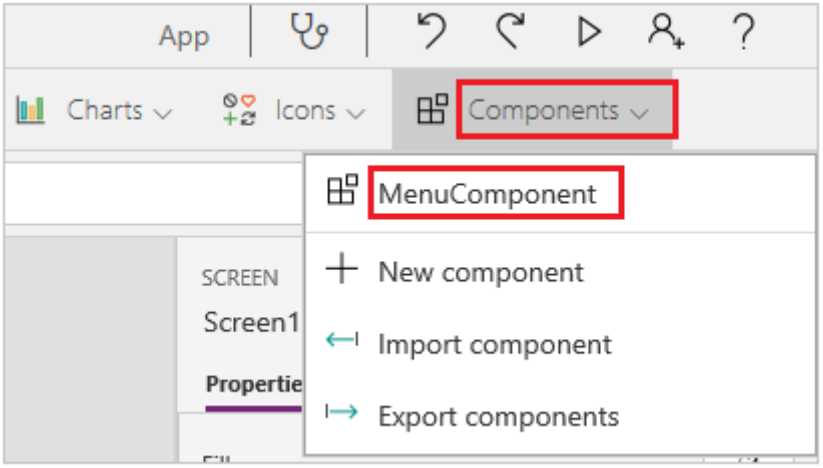
Add component to a screen

Next, add the component to a screen and specify a table of strings for the component to show.

1. In the left pane, select the list of screens, and then select the default screen.



2. On the **Insert** tab, open the **Components** menu, and then select **MenuComponent**.



The new component is named **MenuComponent_1** by default.

3. Set the **Items** property of **MenuComponent_1** to this formula:

```
Power Fx
Table({Item:"Home"}, {Item:"Admin"}, {Item:"About"}, {Item:"Help"})
```

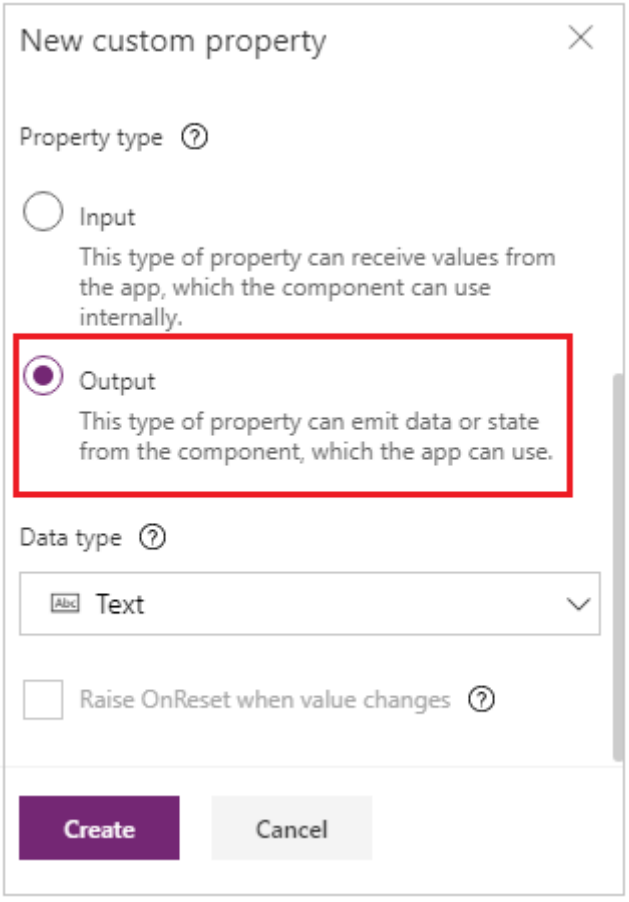
This instance resembles this graphic, but you can customize the text and other properties of each instance.



Create and use output property

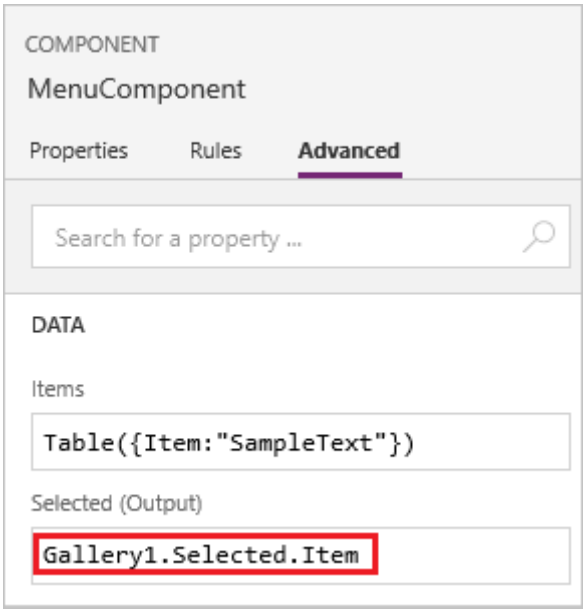
So far, you created a component and added it to an app. Next, create an output property that reflects the item that the user selects in the menu.

1. Open the list of components, and then select **MenuComponent**.
2. In the right-hand pane, select the **Properties** tab, and then select **New custom property**.
3. In the **Display name**, **Property name**, and **Description** boxes, type or paste **Selected**.
4. Under **Property type**, select **Output**, and then select **Create**.



5. On the **Advanced** tab, set the value of the **Selected** property to this expression, adjusting the numeral in the gallery name if necessary:

```
Power Fx
Gallery1.Selected.Item
```



6. On the default screen of the app, add a label, and set its **Text** property to this expression, adjusting the numeral in the component name if necessary:

   ```
   Power Fx
   MenuComponent_1.Selected
   ```

   MenuComponent_1 is the default name of an instance, not the name of the component definition. You can rename any instance.

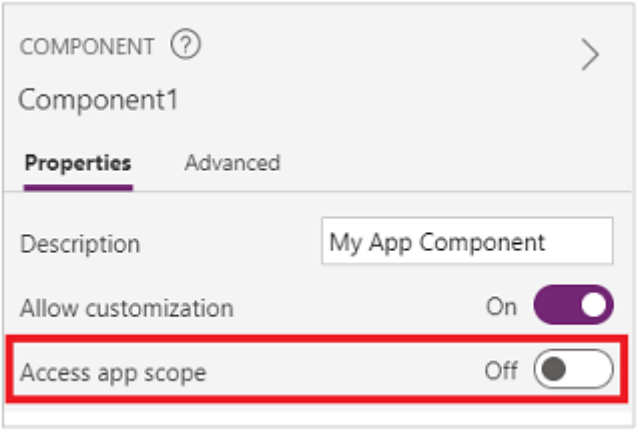
7. While holding down the Alt key, select each item in the menu.

   The **Label** control reflects the menu item that you selected most recently.

Scope

Input and output properties clearly define the interface between a component and its host app. By default, the component is encapsulated so that it's easier to reuse the component across apps, requiring the use of the properties to pass the information in and out of the component. Scope restrictions keep the data contract of a component simple and cohesive, and it helps enable component-definition updates—especially across apps with component libraries.

But there are times when a component might want to share a data source or a variable with its host. This sharing is especially useful when the component is only intended for use in one particular app. For these cases, you can directly access app level information by turning on the **Access app scope** switch in the component's property pane:



When you turn on **Access app scope**, the following items are accessible from within a component:

- Global variables
- Collections
- Controls and components on screens, such as a TextInput control
- Tabular data sources, such as Dataverse tables

When you turn this setting off, the component can't access any of the preceding items. Set and Collect functions are still available but the resulting variables and collections are scoped to the component instance and aren't shared with the app.

Non-tabular data sources, such as Azure Blob Storage or a custom connector, are available whether this setting is turned on or off. Think of these data sources more like referencing an environment resource rather than an app resource. When a component is brought into an app from a component library, these data sources from the environment are also brought in.

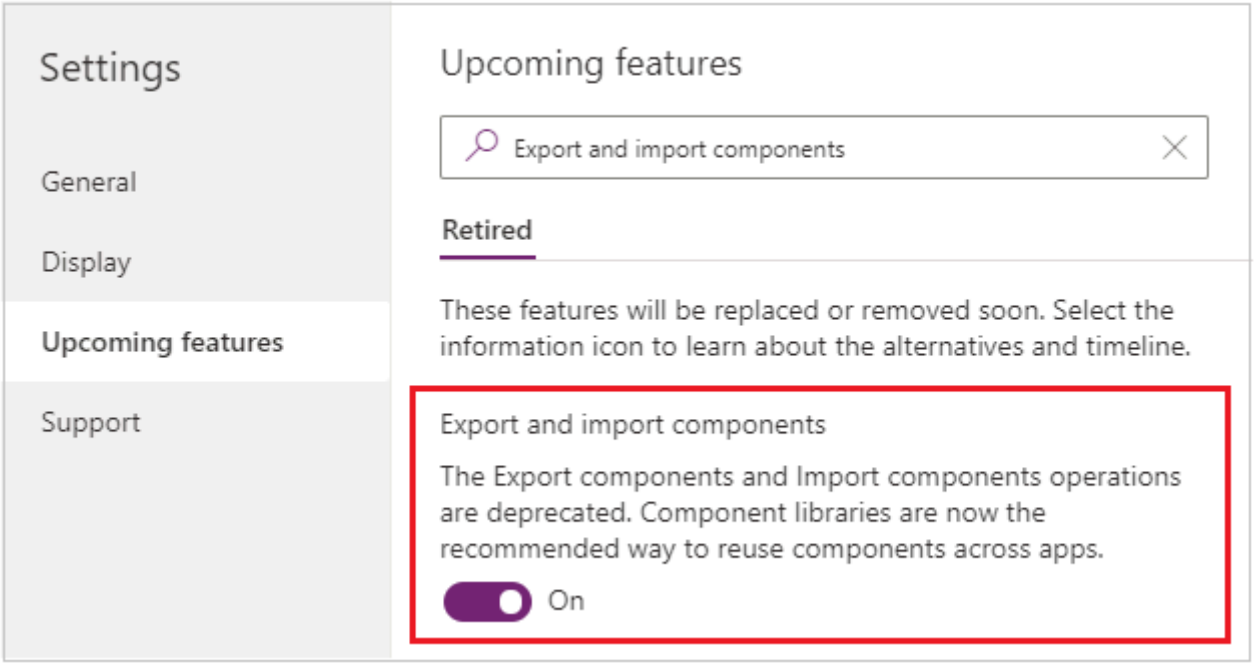
Components in a component library can never access app scope, as there's no single app scope to refer to. So, this setting isn't available in this context, and is effectively off. Once imported into an app, and if customization is allowed by the component maker, the switch can be enabled, and the component can be modified to use the app scope.

> **Note**
>
> - You can insert instances of components into a screen within a component library, and preview that screen for testing purposes.
> - Component library doesn't display when using Power Apps Mobile.

Import and export components (retired)

> **Note**
>
> This feature is retired. Component libraries are the recommended way to reuse the components across the apps. When using component library, an app maintains dependencies on the components it uses. The app maker is alerted when the updates to dependent components become available. Hence, all new reusable components should be created within the component libraries instead.

The ability to import and export components is disabled by default since this feature is retired. While the recommended method to work with components is to use component libraries, you can still enable this feature on a per-app basis as an exception until the feature is removed. To do this, edit your app in Power Apps Studio and then, go to **Settings** > **Upcoming features** > **Retired** > Set **Export and import components** to On.



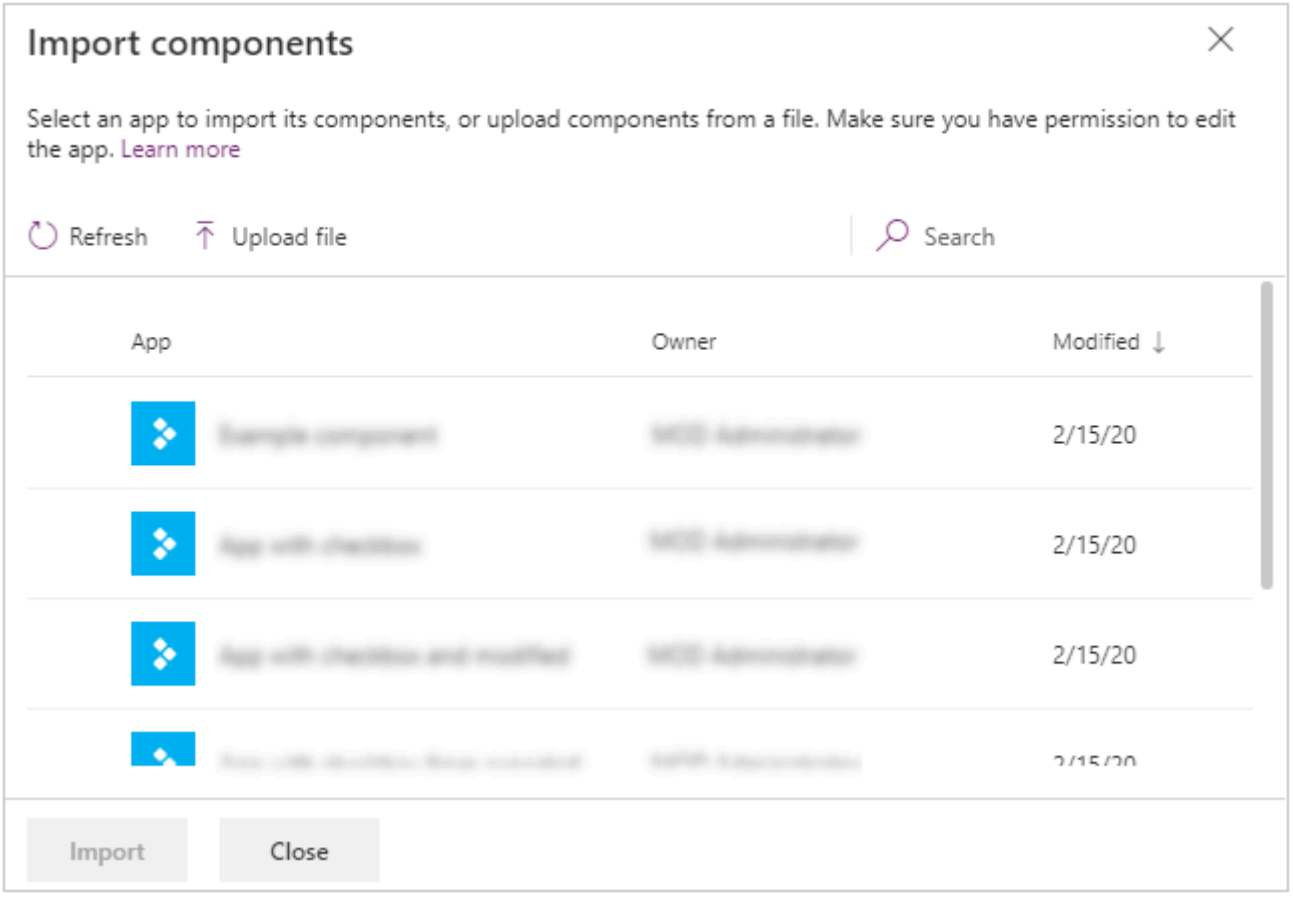
After you enable this feature, you can use the following capabilities to import and export components.

Import components from another app

To import one or more components from one app into another app, select **Import components** from the **Insert** menu and then use the **Custom** drop-down menu. Or use **Components** in the tree view on the left pane.

A dialog box lists all apps that contain components that you have permission to edit. Select an app, and then select **Import** to import the most recent published version of all of the

components in that app. After you import at least one component, you can edit your copy and delete any that you don't need.



You can save an app with existing components to a file locally and then reuse the file by importing it. You can use the file to import components to another app.

If the app contains a modified version of the same component, you're prompted to decide whether to replace the modified version or cancel the import.

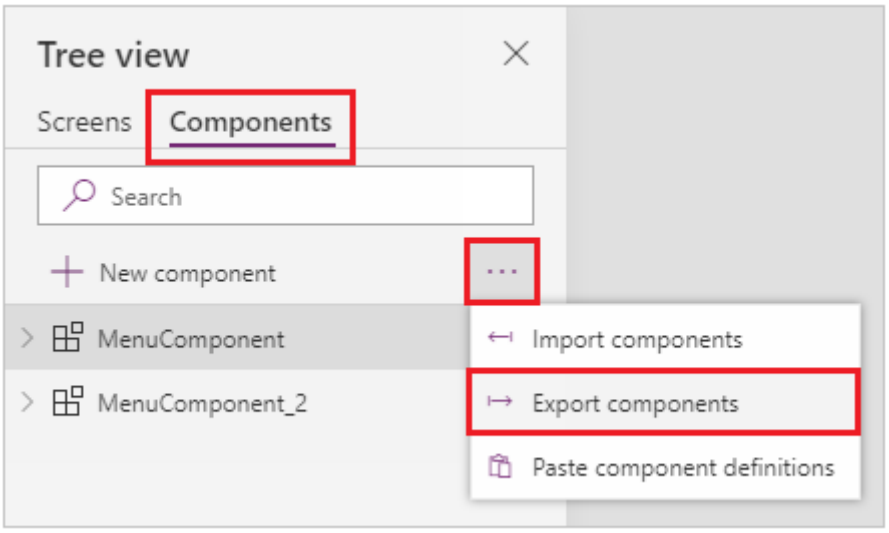
After you create components in an app, other apps can consume the components from this app by importing them.

> **Note**
>
> If a component that you imported from another app is modified in the original app, you must manually import the component again in the consuming app to receive latest component changes. Use component libraries instead to work with component updates more efficiently.

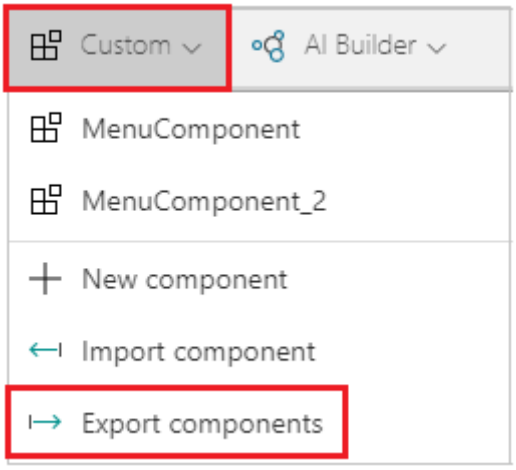
Export components from your app

You can export components to a file and download them for import to another app.

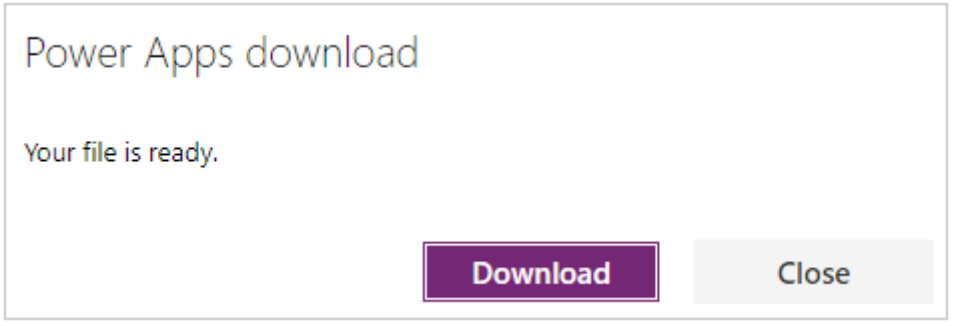
Select the **Export components** option from the **Components** section in the tree view on the left pane:



You can also use the **Insert** menu and then select the **Custom** drop-down menu instead.



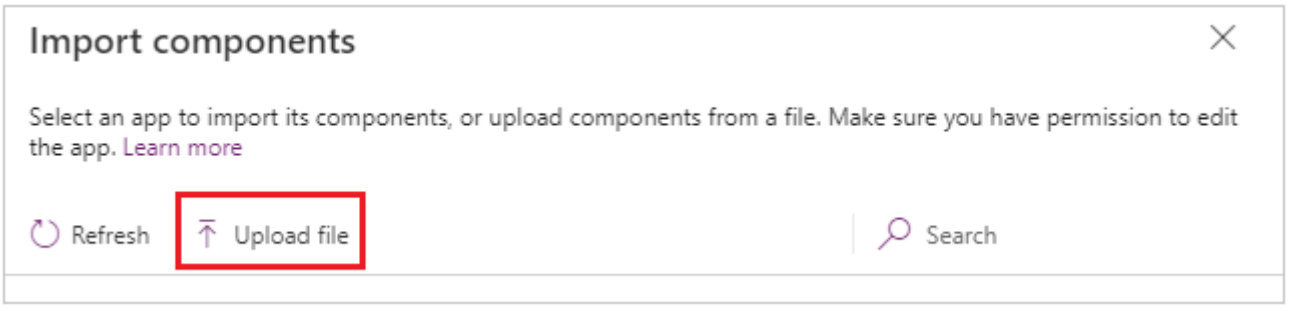
Selecting **Export components** downloads the components to a file:



The downloaded component file uses the *.msapp* file name extension.

Import components from exported components file

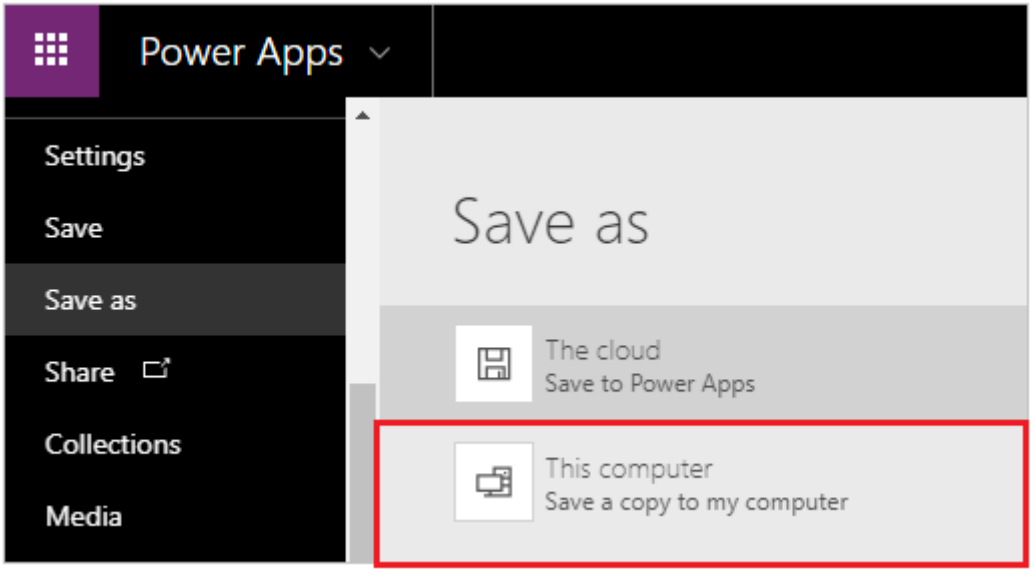
To import components from an exported components file, select **Import components** from either **Insert** menu and then use the **Custom** drop-down menu or use **Components** in the tree view on the left pane. From the components dialog box, select **Upload file** instead of selecting any other components or apps:



From the **Open** dialog box, browse to the location of the component file and select **Open** to import components inside the app.

Import components from exported app

You can save an app locally by selecting **File** > **Save As**:



After you save the app, you can reuse its components by importing them from a file. For more information, see the previous section on importing components from an exported components file.

Known limitations

- When you have two or more instances of the same component in an app, you can't configure a custom input property to a custom output property value across same or

different instances. This action results in a circular reference warning message. To work around this limitation, create a copy of the component inside your app.
- Adding and running Power Automate flows in component libraries isn't supported.
- You can't save data sources or controls that include data from those data sources (such as forms, fluid grids, or data tables) with components.
- You can't insert a component into a gallery or a form (including SharePoint form).
- Components don't support the **UpdateContext** function, but you can create and update variables in a component by using the **Set** function. The scope of these variables is limited to the component, but you can access them from outside the component through custom output properties.

Next steps

Learn to use a component library to create a repository of reusable components.

See also

- Component library
- Component library application lifecycle management (ALM)
- Map input fields of a component
- Add multimedia to a component
- Behavior formulas for components
- Power Apps component framework
- Add canvas components to a custom page in a model-driven app

---

Last updated on 01/14/2026

Component library

Article • 12/19/2023

In the overview article for creating components, you're introduced to components inside canvas app. As you create components inside an app, you can also create a library of components that can be reused. By creating a component library, app makers easily share and update one or more components with other makers.

Component libraries are containers of component definitions that make it easy to:

- Discover and search components.
- Publish updates.
- Notify app makers of available component updates.

> **Note**
>
> Component libraries are the recommended way to reuse components across apps. When using a component library, an app maintains dependencies on the components it uses. The app maker will be alerted when the updates to dependent components become available. Hence, all new reusable components should be created within the component libraries instead. An earlier Power Apps feature that allowed importing components from one canvas app to another is retired.

Difference between an app and a component library

A component library provides a centralized and managed repository of components for reusability.

The **Insert** pane on the left navigation defaults to a components tab if you create a component library. When you create an app, this view shows screens instead of components.

The screens inside a component library are available for testing only. It provides library creators a way to quickly test the created components on an actual screen and also validate the update behavior as components are enhanced over time. To use the components from the component library, you must create an app that uses the component library.

You can preview component library components using the screens inside the library with the play option. When you select the component tab, the play option is disabled. The component library doesn't display when using Power Apps Mobile.

> **Note**
>
> The component library discussed in this article is different from the Power Apps component framework that enables developers and makers to create code components for model-driven and canvas apps. For more information, go to Power Apps component framework overview.

Working with component library

You can create a new component library or edit an existing component library from the same interface. Go to Power Apps > select **More** from left navigation > select **Discover all** > select **Component libraries** from the **App enhancements** section. You can also pin **Component libraries** option to your Power Apps interface. More information: Left navigation in Power Apps

Create an example component library

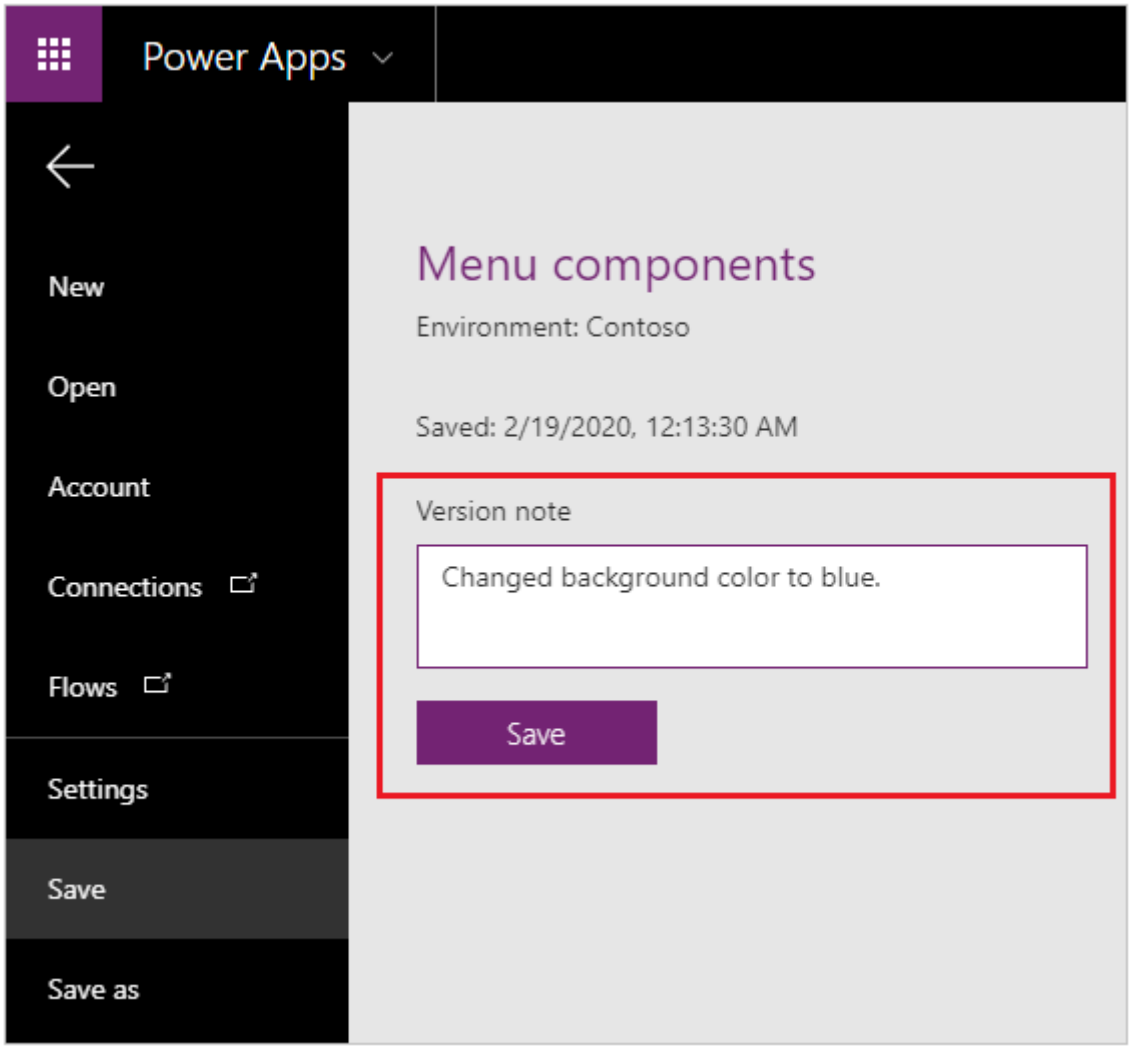
The steps to create components inside a component library are the same as creating components inside an app. You'll create a component library and then reuse the steps for creating components from components overview example. Then you'll use the component library to provide the reusable components in a new app.

1. Sign in to Power Apps.
2. Select **Component libraries** in the left navigation, and then select **New component library**. Alternatively, select **More** from left navigation > select **Discover all** > select **Component libraries** from the **App enhancements** section.
3. Name the component library as *Menu components*; you can also provide a different name of your choice.
4. Follow the steps to create components from components overview example. You don't have to open Power Apps Studio or create a new blank app, since you already have created a new component library. Start with step 2.

   After following the steps to create components, follow the next set of steps to also add components to a screen and the steps to create output property.

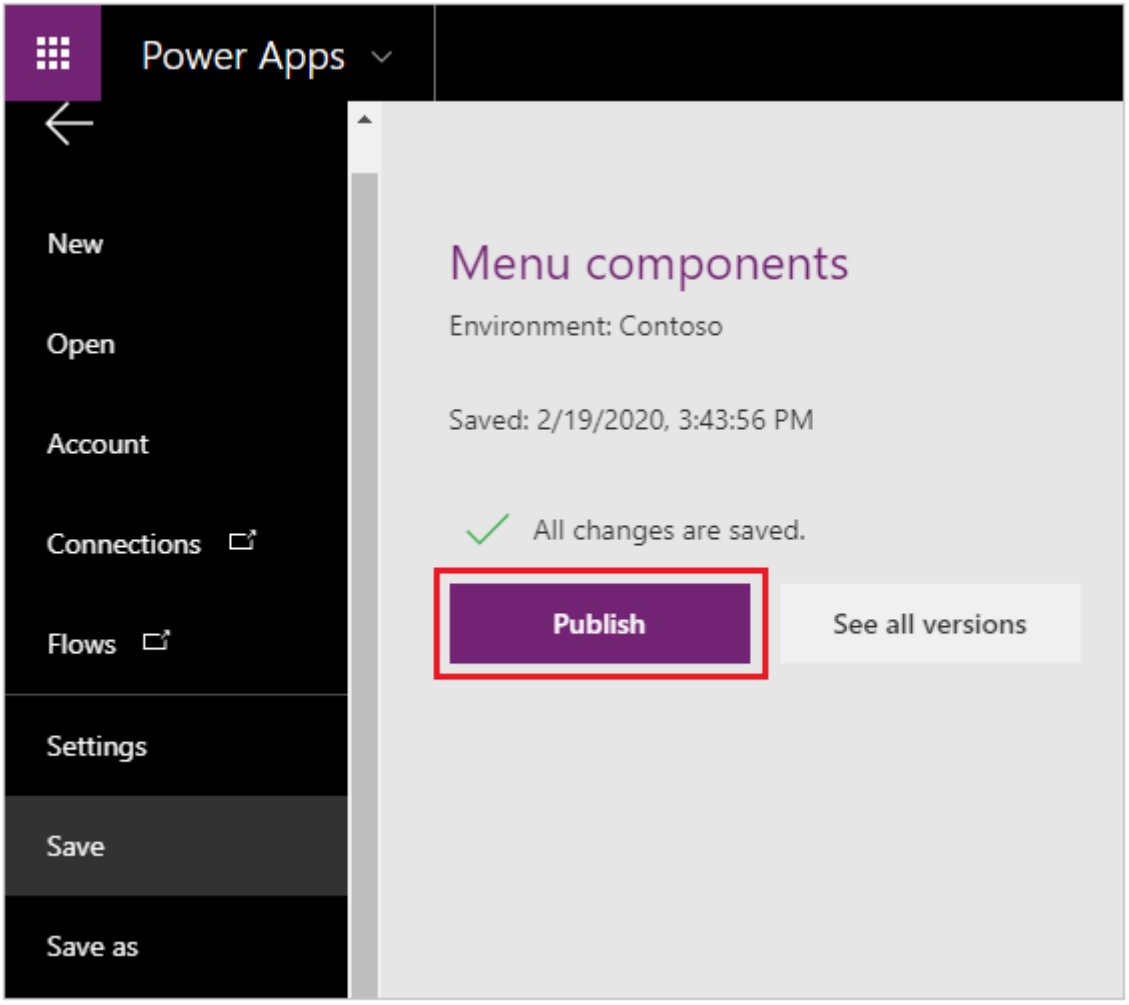
5. After you complete the components creation and testing, save the component library by selecting the **File** menu and then selecting **Save**.

   You also have an option to save a **version note**. A version note is useful to retrieve versions of a component library and for upgrading the components used in apps from this component library.



> **Tip**
>
> A version note is useful when reviewing versions of a component library and for the app makers using your component library to review changes and update apps consuming these components as needed. Go to update a component library for more details.

6. A saved component library can be published. Only published component library updates are available for apps that consume a component library. Select **Publish** to publish the component library version:

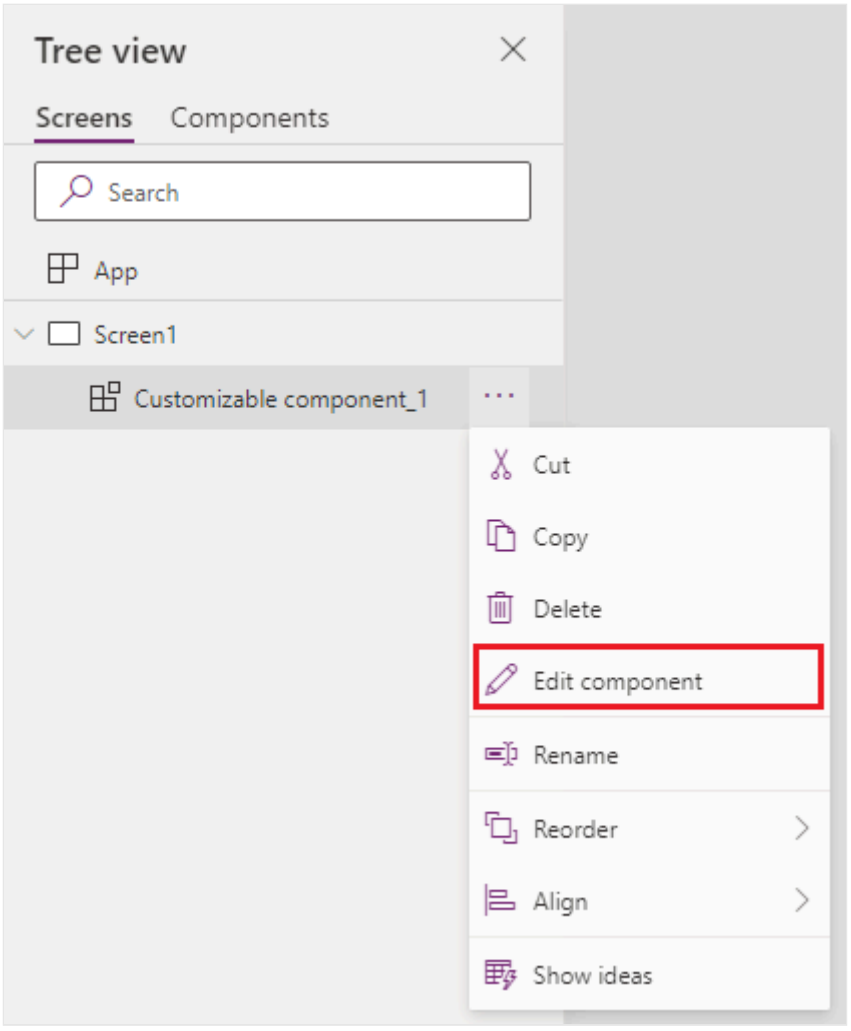


Library component customization

When creating component inside a component library, you have the option to allow the component to be customized when used by apps consuming the library components. By default, this setting is turned **On**. Turn this setting to **Off** to remove the ability to customize component while using in other apps.

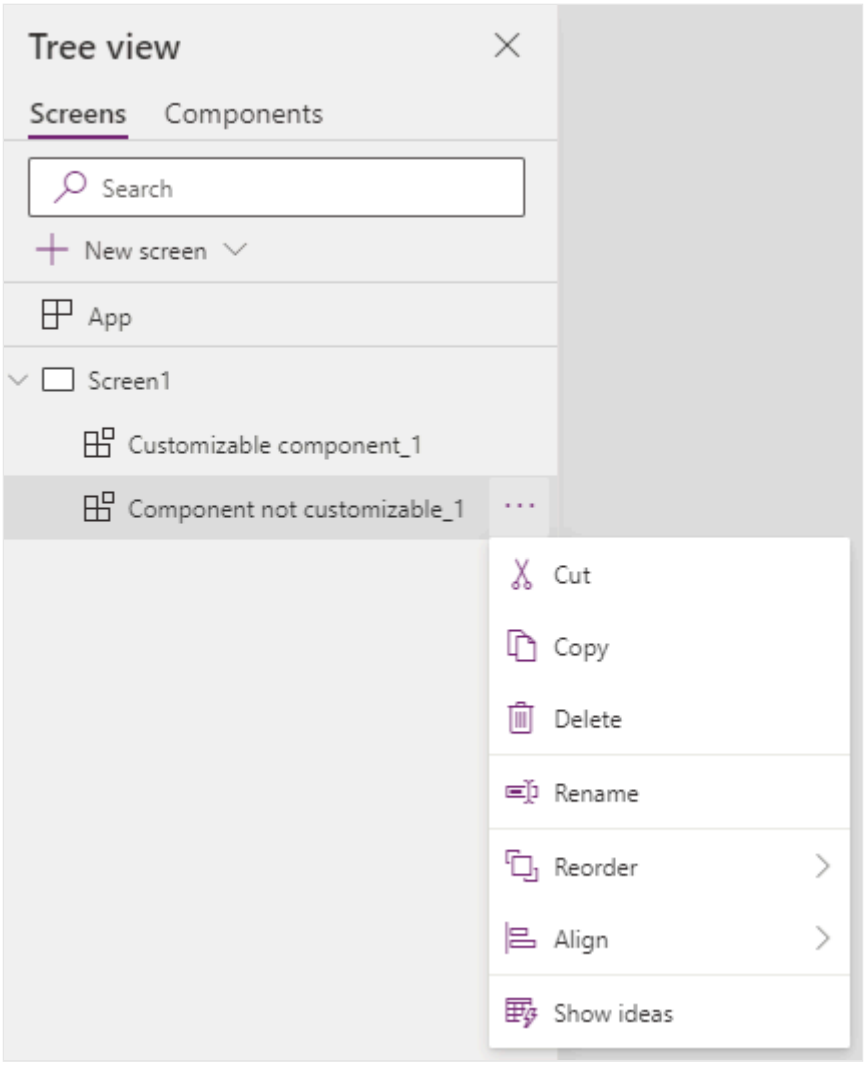


When you add a component that allows customization, you'll see an option to edit this component for more customization. App makes a local copy and you can make changes to local component copy. The association with the component library is removed once the component is edited.



Selecting **Edit component** prompts you to copy the component locally for customization.

However, when you don't allow the components to be customized, the consuming apps won't show the option to edit the component.

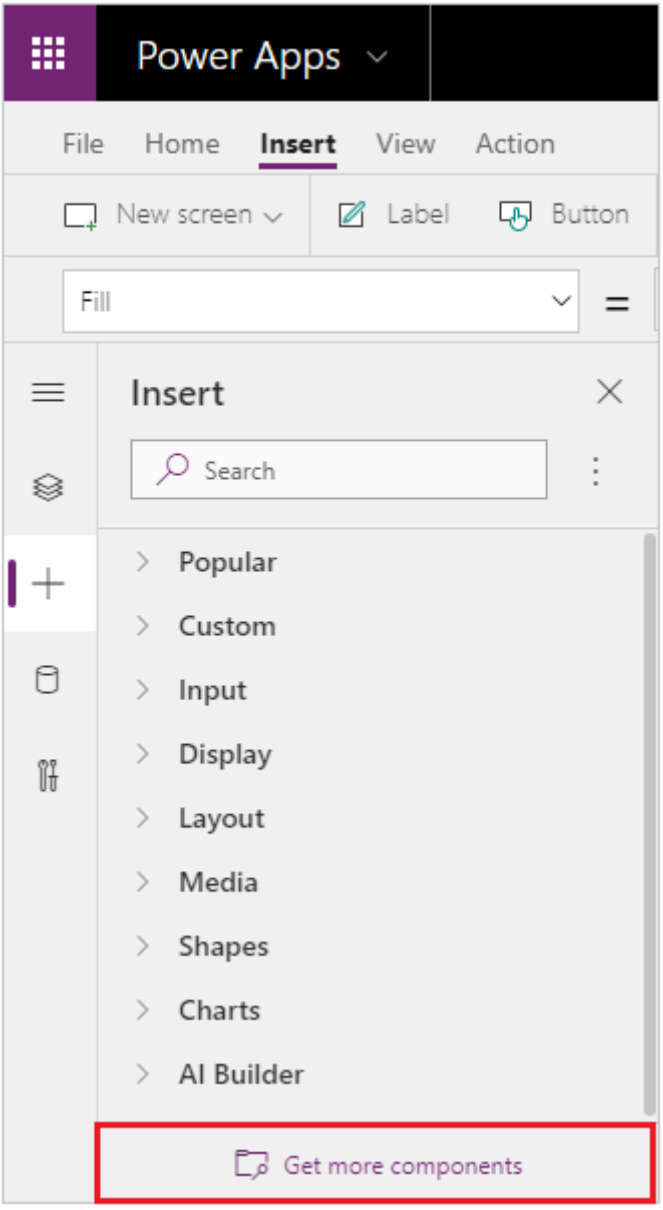


In this case, the only way to edit the component is to edit the component library where the component is stored.

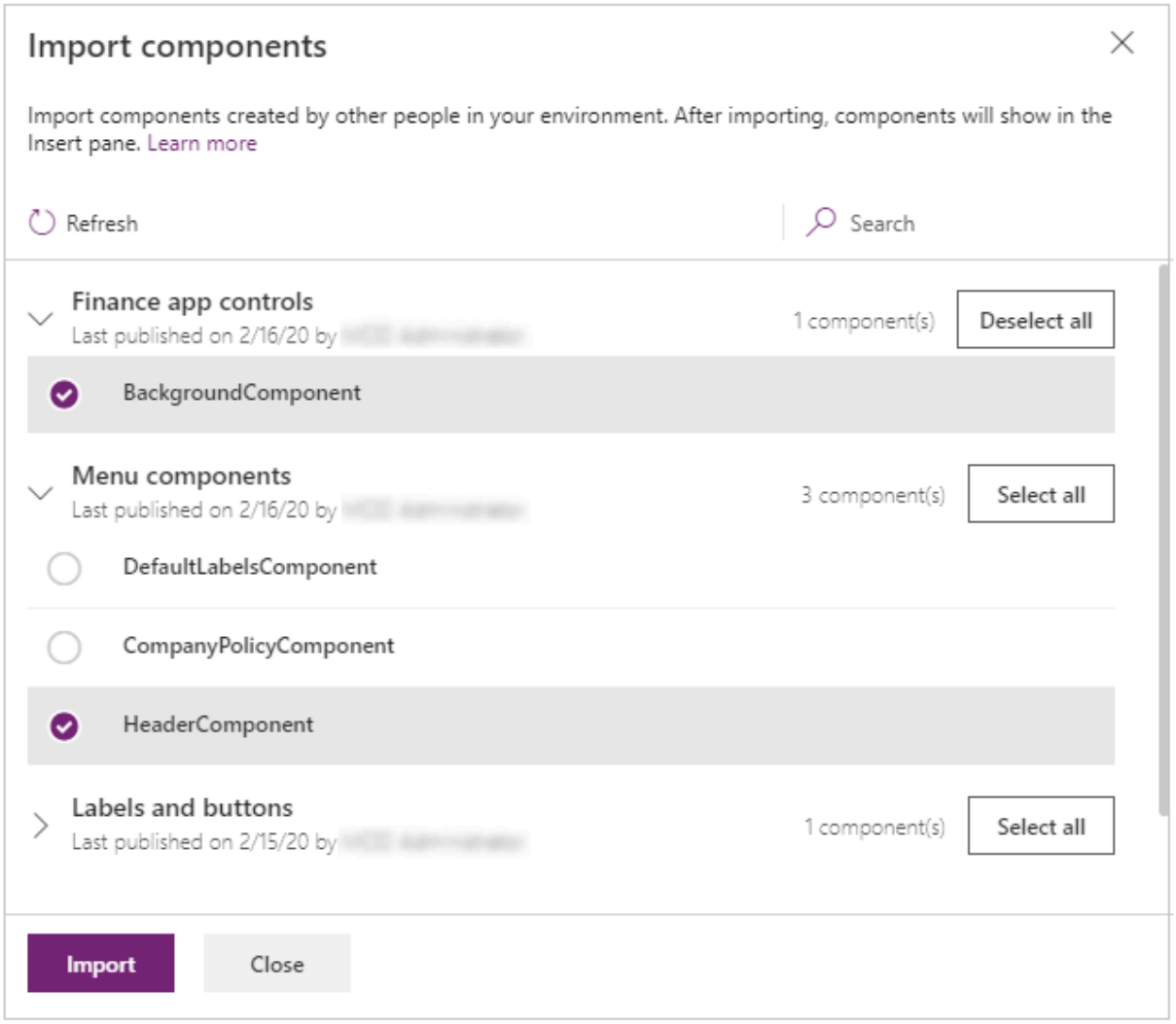
Import from a component library

After you create a component library and publish, apps can consume the components from this component library by importing the library. You can also share a component library.

To import from a component library, edit an existing app or create a new app. After the app opens in canvas app studio, select **Insert** or the + on the left navigation. Then select **Get more components** to list the component libraries available in the current environment:



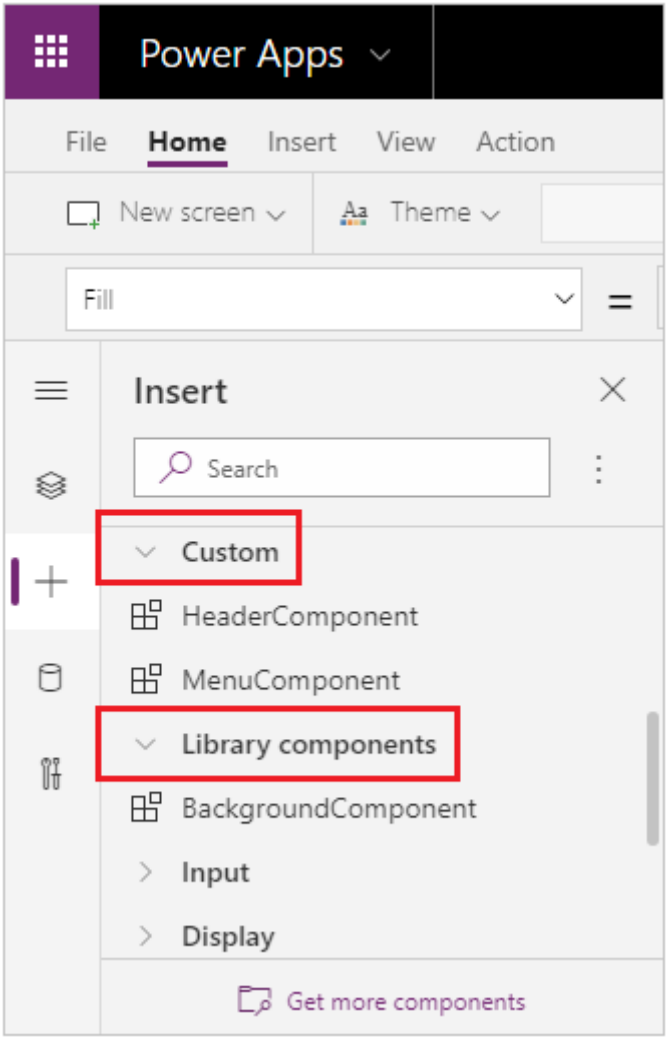
You'll see the list of component libraries available in the current environment on the right side of the screen. Select an individual component from a component library. Or use **Select all** to import all of the components from the library at once:



> **Note**
>
> If a maker doesn't see the component library listed in the import section, ensure the component library is shared with the maker. For more details, go to component library permissions.

Notice you can select and import more than one component and across different component libraries.

Components available inside the app are listed under the **Custom** category in the list of components in the **Insert** pane. Components available from imported component libraries are listed under the **Library components** category:



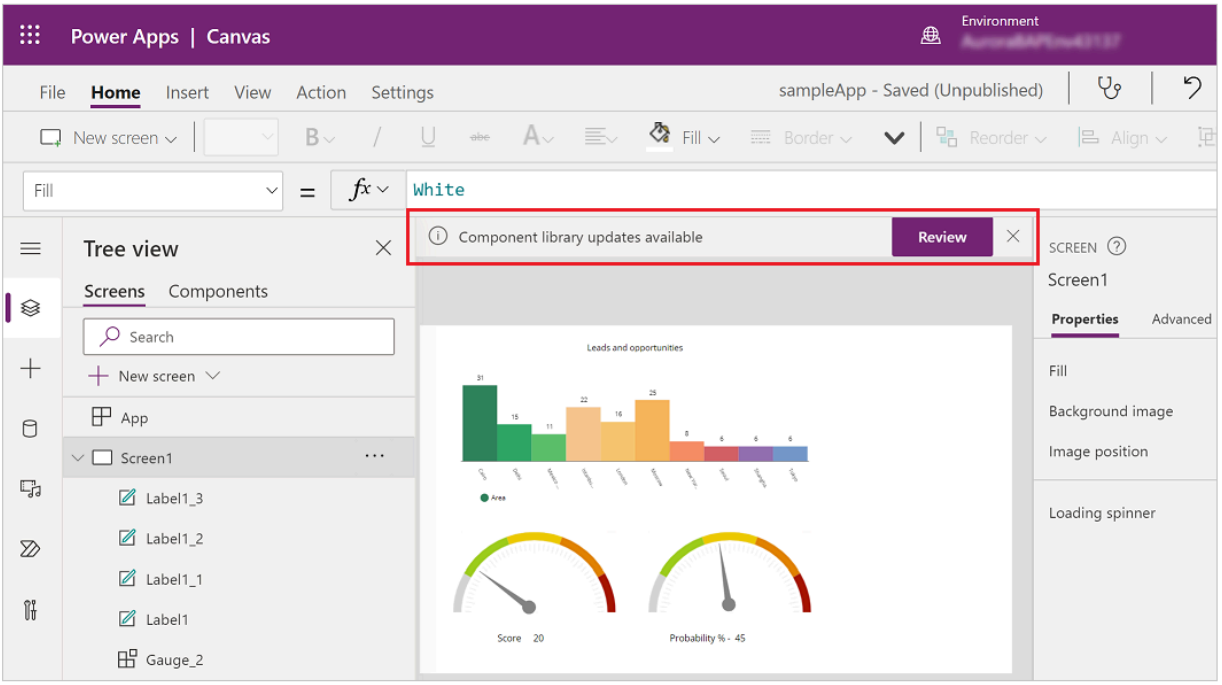
Update a component library

You can modify an existing component library and save any changes with other version notes. However, the updated component library version must be published for use in existing apps that use the component library. The example component library steps above explain how to publish a component library after saving it.

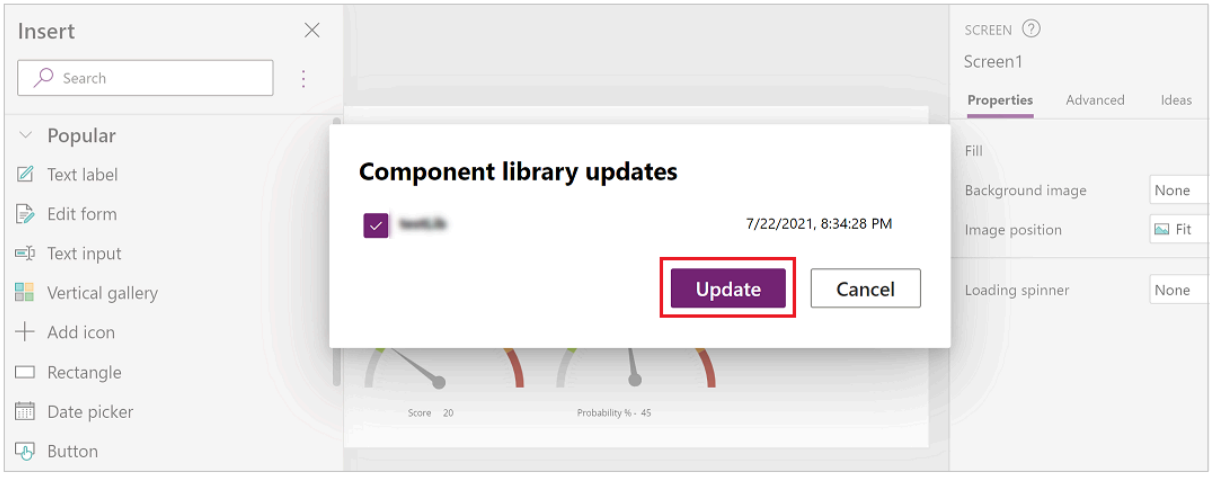
There are two ways apps using the components from library can get the latest changes.

Method 1: Component update notification on app edit

Makers of other apps are notified of updated components being available. The notification appears when makers edit the apps in canvas app studio. They can choose to update the components:

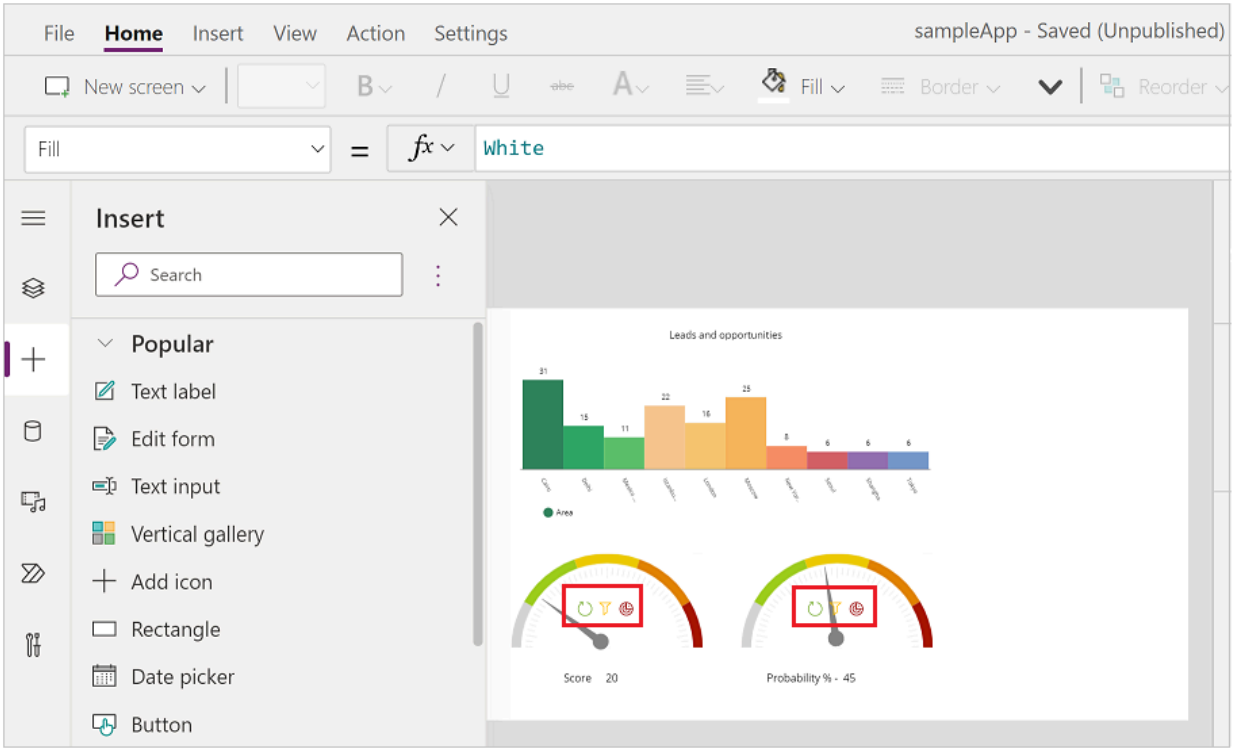


Select **Review**, and you'll see the option to update the component:



Notice that the version note you added when publishing the component library version shows up here.

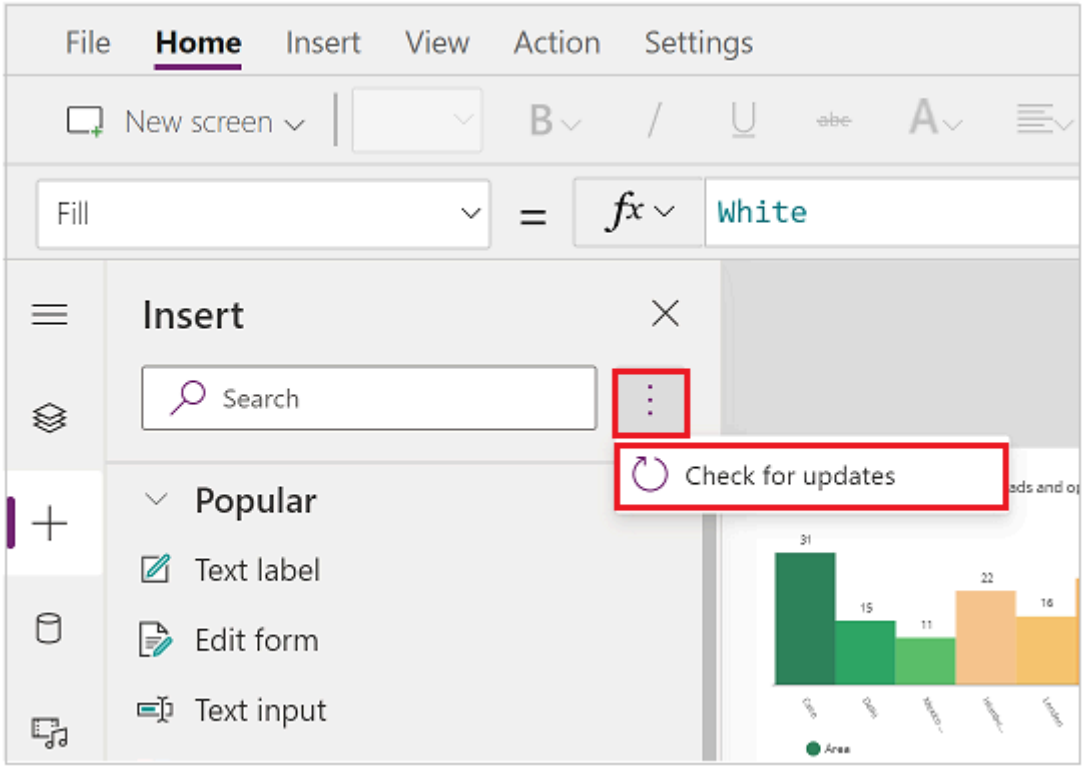
Select **Update** to update the components.



Method 2: Proactive check for updates

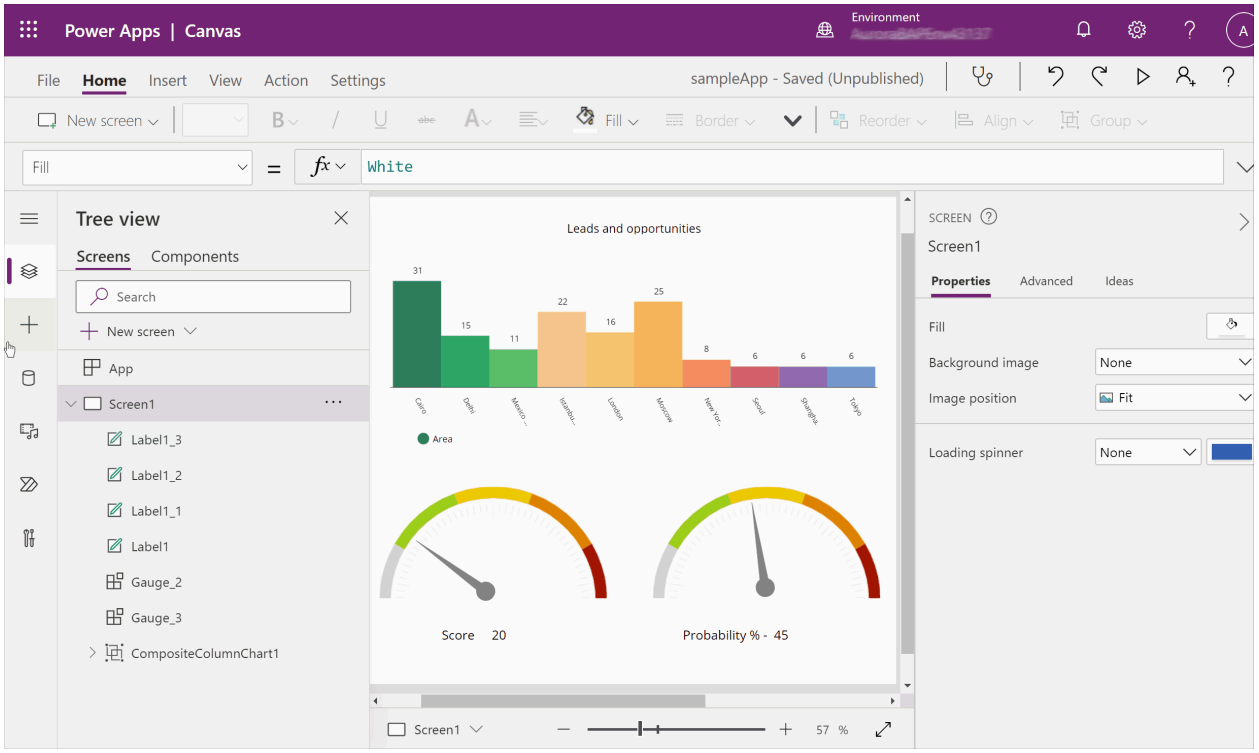
This method is to check for the app update proactively while working on the app in Studio.

To do this check, select Insert component panel on the left-side, and then select ... (ellipsis) next to the search button. And then, select **Check for updates**.



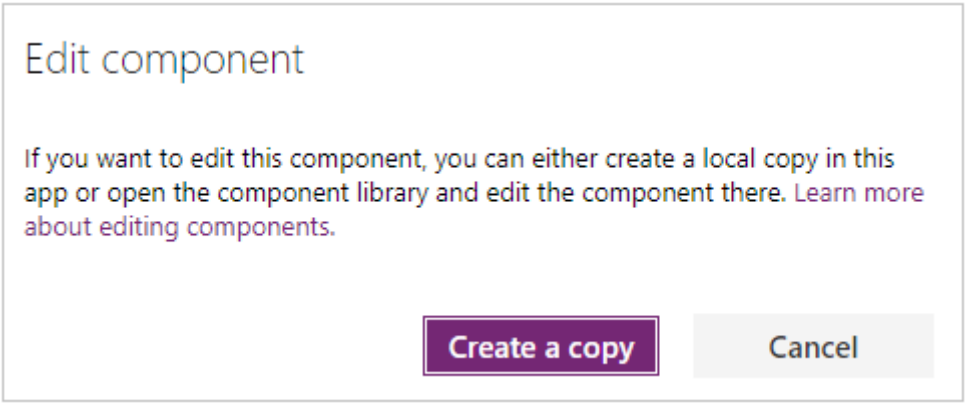
If there are updates in the components used from library, maker will get the message to review and get the latest changes. Upon review and update, the app gets updated with the latest components.

The following animation shows the process to get the latest components.



Updating component from a component library in a consuming app

If you import a component from a component library, you can't edit it inside the consuming app. If you select **Edit component**, you'll see an option to create a copy of the component inside the current app for you to make changes:



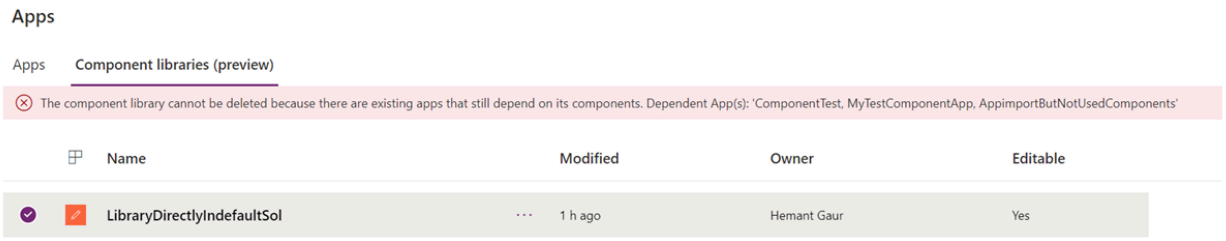
If you select **Create a copy**, the component is copied to the local app. The local copy of the component appears under the **Custom** category in the **Insert** pane. This local copy

of the component won't receive updates if a new version of the originating component library is published later.

> **Note**
>
> The in-use instances of components from the component library won't be affected after you create a local copy of a component, and will remain linked to the component library definition. Only instances of the locally copied components (from **Custom** category) are considered to be the local app components.

Deleting a component library

If a component library is referenced by a canvas app, it can't be deleted. This is true for both Power Platform and non-Power Platform environments. Remove the component from the app before deleting the library. You can find information about the apps using the component library in either solution dependencies view or in the message, which is shown when you try to delete library.



Component library permissions

Sharing a component library works the same way you share a canvas app. When you share a component library, you allow others to reuse the component library. Once shared, others can edit the component library and import components from this shared component library for creating and editing apps. If shared as a co-owner, a user can use, edit, and share a component library but not delete or change the owner.

Additional considerations when sharing component libraries to security groups:

- You can't use the sharing UI when sharing component libraries created as part of a solution. Instead, use Set-PowerAppRoleAssignment cmdlet with **CanEdit** permission level to share the component libraries with the selected security group.
- Component libraries created in a Power Platform environment without a Dataverse database shared as **Co-owner** automatically grant **CanEdit** permission to the selected security group.

Known limitations

- Known limitations applicable to components also applies to component libraries.
- You can't import components from a locally saved component library. It's possible to save a component library using **File** > **Save As** > **This Computer**. But if you try to import such saved component library, you'll see the following error message:



- If you're importing an app and a component library in an environment for the first time, ensure the app uses the latest version of the component included with the component library. Otherwise, the app won't receive updates after the import. The app will recognize and prompt for the updated version upon subsequent component library imports to the same environment.
- Code components aren't supported inside component libraries.

See also

- Canvas components
- Component library application lifecycle management (ALM)
- Map input fields of a component
- Add multimedia to a component
- Behavior formulas for components
- Power Apps component framework
- Add canvas components to a custom page in a model-driven app

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Component library application lifecycle management (ALM)

Article • 01/24/2023

A component library is a special type of canvas app that can contain one or more canvas components. These library components can then be used by all the other canvas apps in the environment. This capability enables you to create reusable shared components across all apps in an environment, unlike app-level components that are restricted to one app.

To use a component from a component library, you have to first import the component into the canvas app. Once imported, you can then add the component to any app screen. Any updates in the component definition from the component library will prompt you to review and incorporate the change on demand or when the app is opened for editing. Component libraries and dependent apps can also be moved to another environment using standard Microsoft Dataverse solutions.

> **Note**
>
> In this article, the term "import" refers to importing a component from a component library to an app, and shouldn't be confused with importing a solution into Dataverse.

When a component from a component library is imported into a canvas app, that component's definition is copied into the definition of the canvas app. Once a component definition has been imported, the app is "self-contained" as far as that component definition is concerned. The app maker can choose to edit the component, and create local instances of the component within the app. At this point there's no direct link to the component library from where the component originated. This self-containment characteristic also applies if the canvas app is then migrated to a different environment where the component library isn't present. You can continue to create instances of the imported component definition within the apps in the target environment, and the apps can still be published and played. No new updates will be prompted or received in the consuming app in this case.

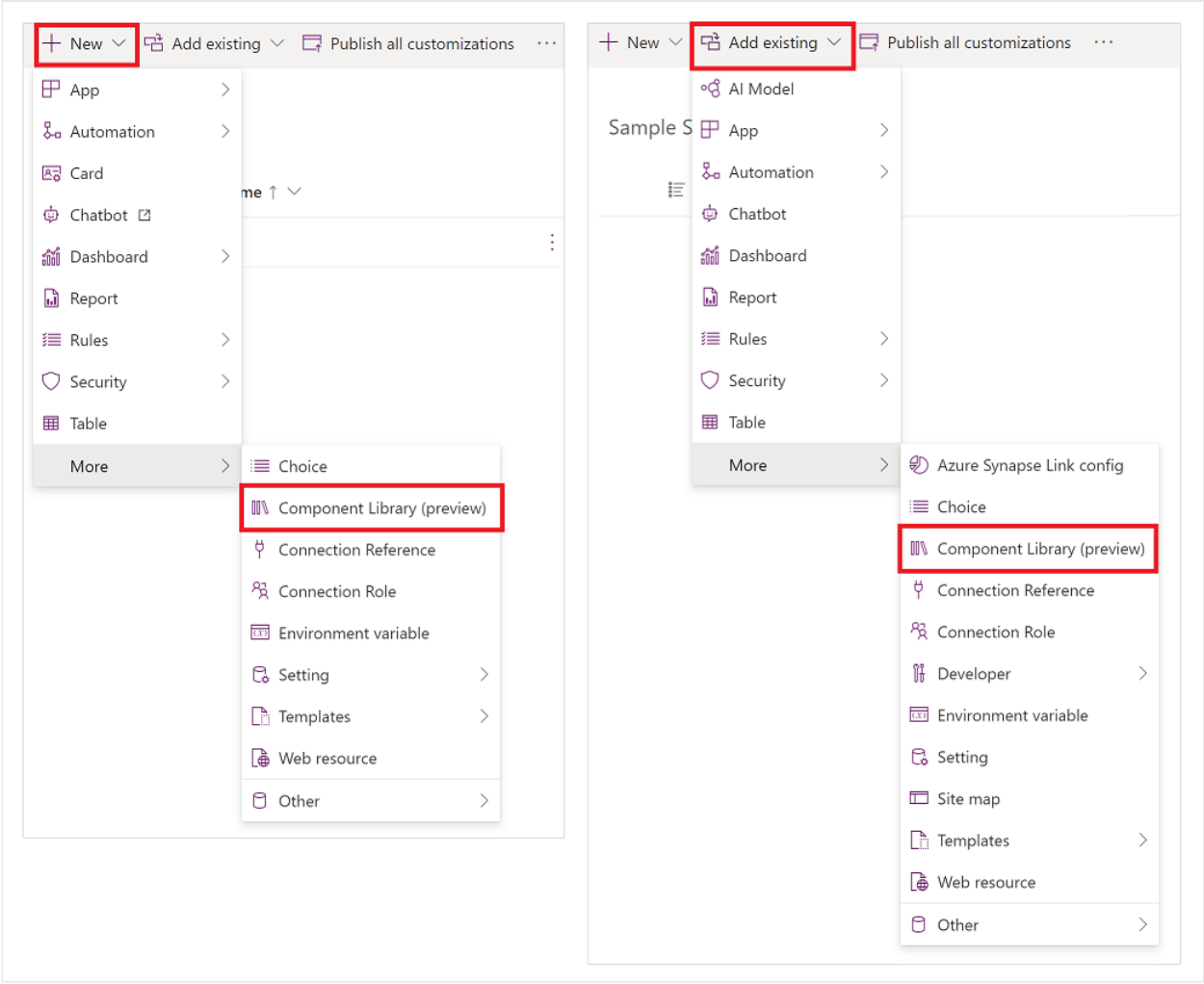
In order to maintain the relationship from the app to the component library, ensure that you use the component library to make any changes to the component, instead of editing the component within the consuming app.

Canvas apps and component libraries solution support

Consistent with the other solution object dependencies, if a canvas app imports a canvas component from a component library, it will have a dependency on that component library. In order to move an app to the new environment, you'll need to either package the component library inside the same solution or install it as a pre-requisite. App to component library dependency is maintained in the target environment. At a later point, when a component library with the updated component is imported using a solution into the target environment, existing apps will get the new component definitions using the regular component update flow.

Creating and exporting component library in a solution

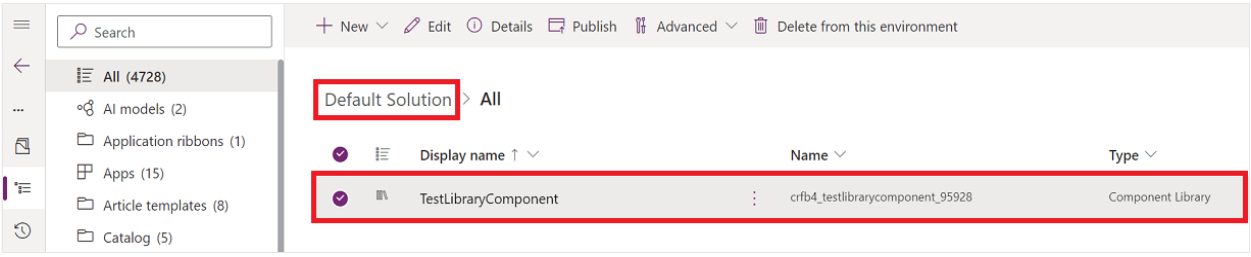
You can either create a component library directly from within the solution, or add it to an existing solution.



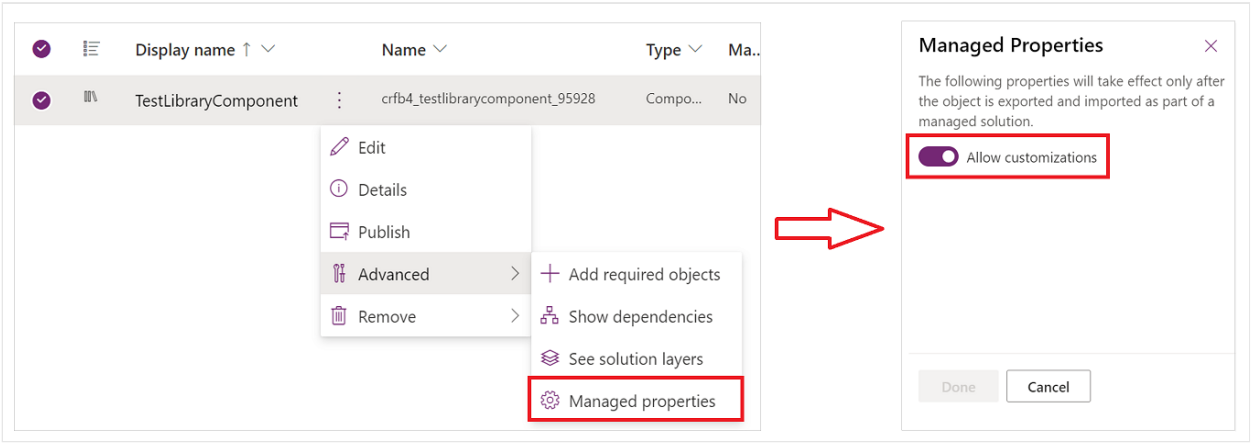
When a component library is saved in an environment that has Dataverse available, component library is automatically added to the default solution. A unique logical name

is generated for the component library with the **Default CDS Publisher** prefix. This behavior is to ensure that the solution system is aware of its presence, and can link the dependencies from the apps that use the component library's logical name.

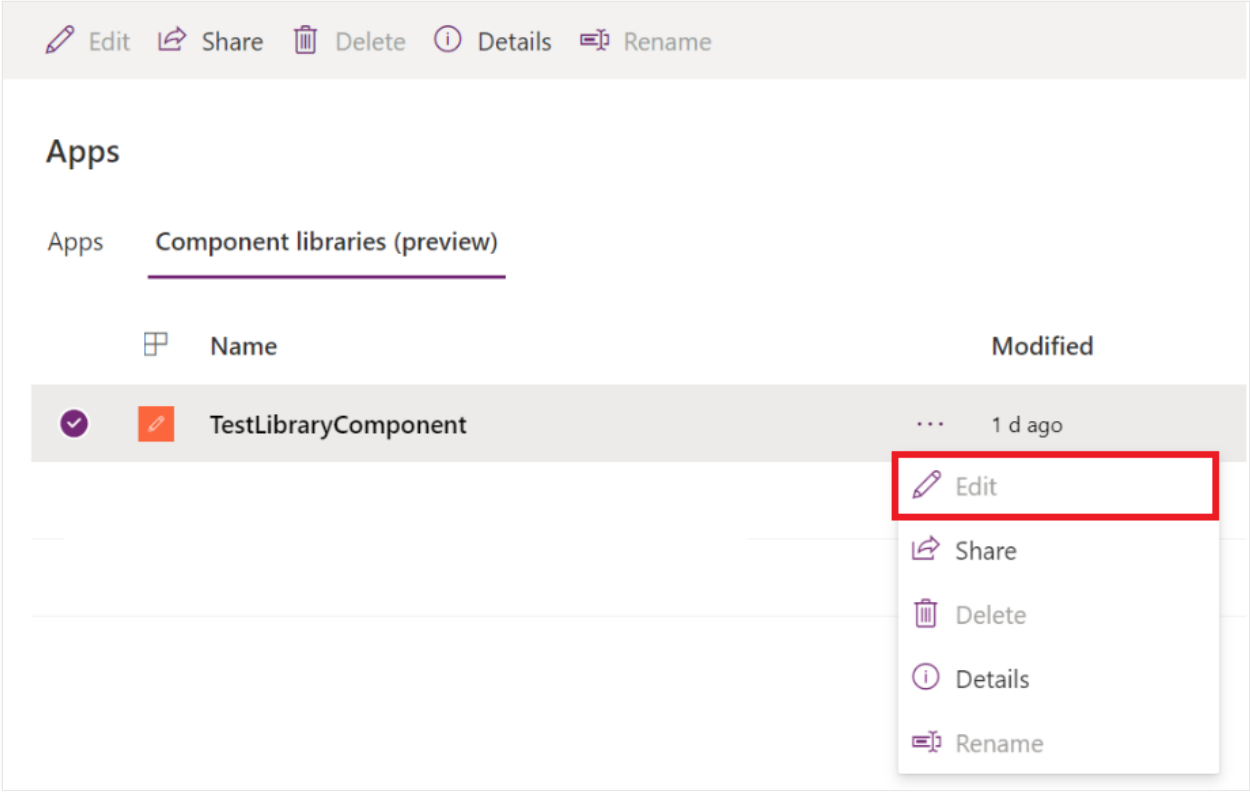
> **Note**
>
> Component libraries created before the rollout of the component library ALM feature need to be edited, published, and the editor must be closed explicitly before they're enabled for the ALM capabilities. You can check the component library ALM readiness by its presence in the default solution.



Component libraries inside a solution also support **Allow customizations** managed properties that govern the behavior of the component library in the target environment.

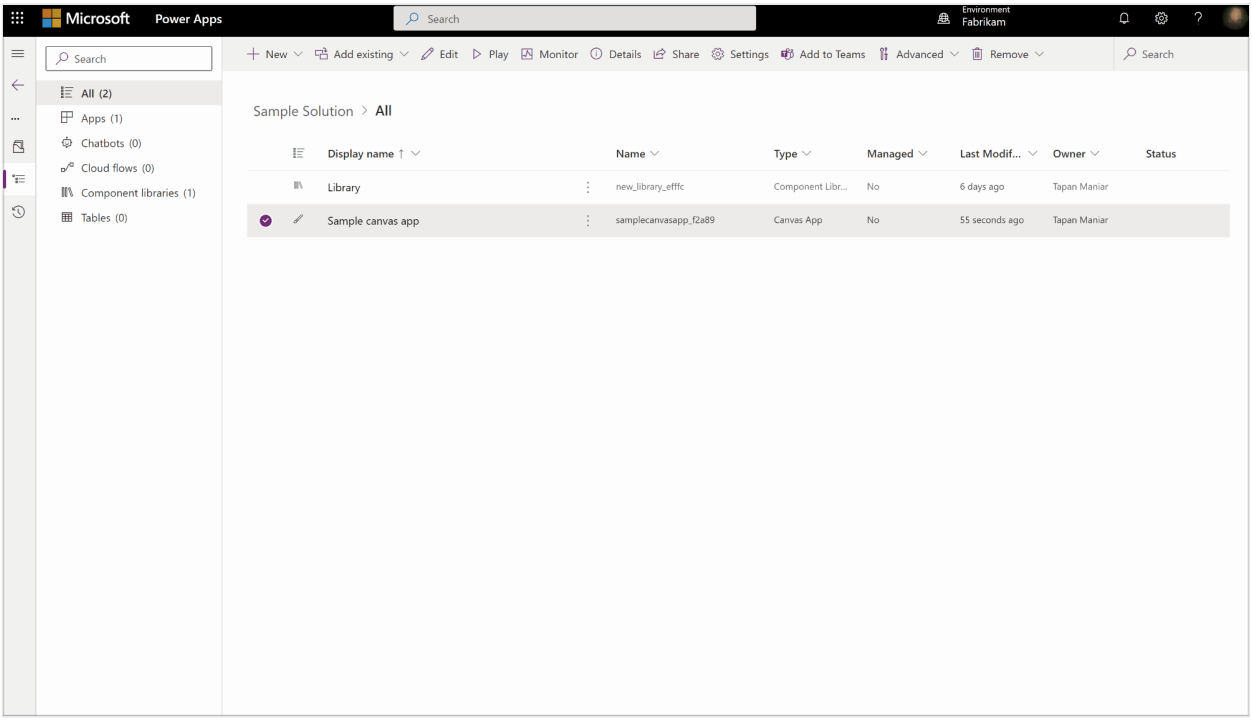


If you turn off this setting, and export the solution to a target environment, you won't be able to edit the component library.



Component library dependencies

Apps that use components from the component library will be marked as dependent in the solutions infrastructure. This behavior applies to all apps that are added to any Dataverse solution in a given environment. You can still create apps outside of solutions, but those apps won't have any solution dependencies. You can later add these apps to solutions to make them part of the solution ALM.



If you import a solution that only has an app that uses a component from a component library, but excludes the component library, you'll see the following message:

"Import failed due to missing dependencies for <app name>".

In this case, you can choose to install the component library solution first, or bundle the component library with the solution that contains the canvas app. Either of the steps would ensure the app will have the dependency created in the target environment.

When the library is updated and a newer component version is imported through the solutions, the app will get a notification and receive the updates when the app is opened for editing.

> **Note**
>
> If the component library managed property **Allow customizations** is turned off, the component library can't be edited in the target environment.

Dependencies are calculated based on the latest published state of an app. If you restore an older version of the app that doesn't use a library component, the dependency will be removed from the app, and the solution. Importing a component into an app from a component library without actually using it also creates a dependency since the unused library component remains available within the app for a future use.

> **Tip**
>
> Editing a component from a component library inside the consuming app creates a local copy. At this point, the library component is still available for use through the **Insert** pane. In order to remove the dependency completely, delete the component from **Insert** > **Library components** > **...** (ellipsis) > **Remove from app**.

Best practices and troubleshooting

- Limit the number of components in a library to 20 to get optimal performance. Plan and create multiple component libraries in advance as the number of components in them will likely grow over time. This approach will also reduce the solution payload as apps are moved across the environment.
- There's a delay from when the component library is published to when it's available to the application, and can take up to 5 minutes.

- If the app isn't able to receive the update from the library component in the target environment where the solution is installed, check using the below actions:
  - Changes made to a component library in a target environment reside at the top layer of a component. Subsequently, these customizations define the runtime behavior of the component. To remove these unmanaged customizations, see Remove an unmanaged layer.
  - Determine the component library logical name from the solution view. Use the default solution if library isn't explicitly added to solution.
  - Download app using the library component to local computer using **File** > **Save as** > **This computer**. Rename the downloaded file to have a .zip extension, and unzip the package. Open the **Properties.json** file, and then search for the keyword "LibraryDependencies". You should see a matching library logical name.
  - If you're consuming the solution, check that the canvas app has properly identified the component libraries as solution dependencies. If the solution doesn't properly identify the component libraries as solution dependencies, that means the app dependency to the component library link hasn't been created properly. In that case, check with the solution provider to resolve the issue.
  - If you're the solution publisher, check that the component libraries are saved with the library logical name in the solution, and that it's same as the one referenced in the component library .msapp package.

- Solution export always exports the latest version of the component library. Hence, always update the apps with the latest component version before exporting them through solutions. This action ensures that the apps have the same component version as available in the latest version of the component library. Apps and library are considered to be synchronized with each other when they're moved to a target environment for the first time; and hence, you're not prompted for any update being available when editing the app.

See also

- Canvas components
- Component library
- Map input fields of a component
- Add multimedia to a component
- Behavior formulas for components
- Power Apps component framework
- Add canvas components to a custom page in a model-driven app

Map input fields of a component

Article • 01/24/2023

A component can receive input values to emit or process data using custom input properties. In this article, you'll learn about working with such components expecting one or more input properties with a specific schema for the given table or record, and how to map the component input fields to the columns of the data source.

> **Tip**
>
> To learn about what the custom input and output properties in components are, see **Custom properties in components**.

Map columns

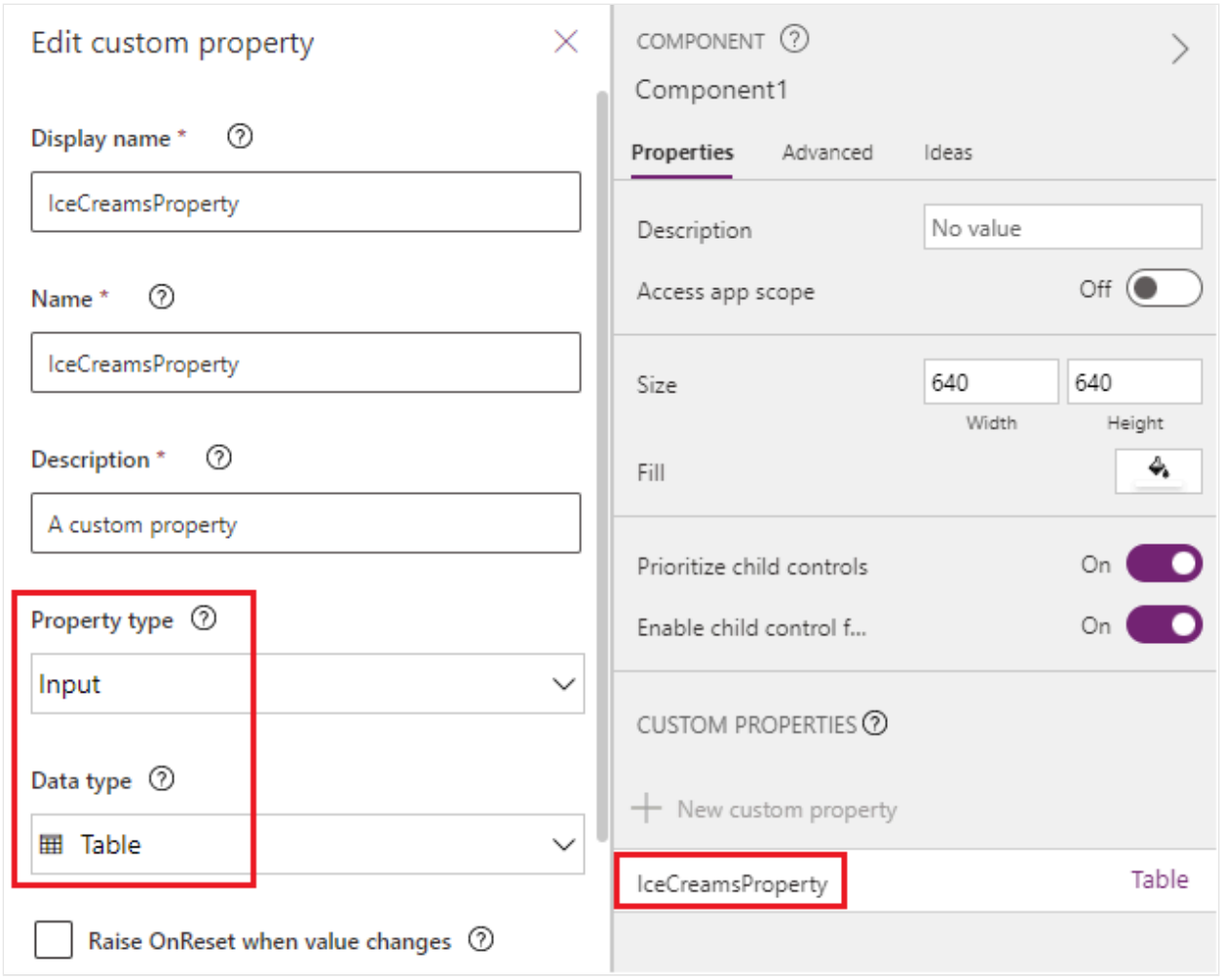
Use the RenameColumns() function to rename one or more columns of a table to match the input property schema for input column selection.

For example, consider a component that expects a table input with the following format:

Expand table

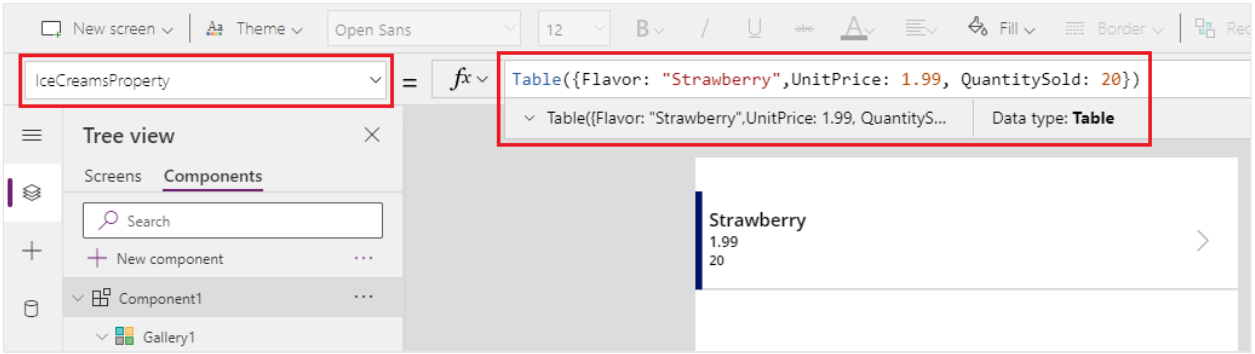
Flavor	UnitPrice	QuantitySold
Strawberry	1.99	20
Chocolate	2.99	45

The input property expects table data type:



The schema of the input property looks like the following formula:

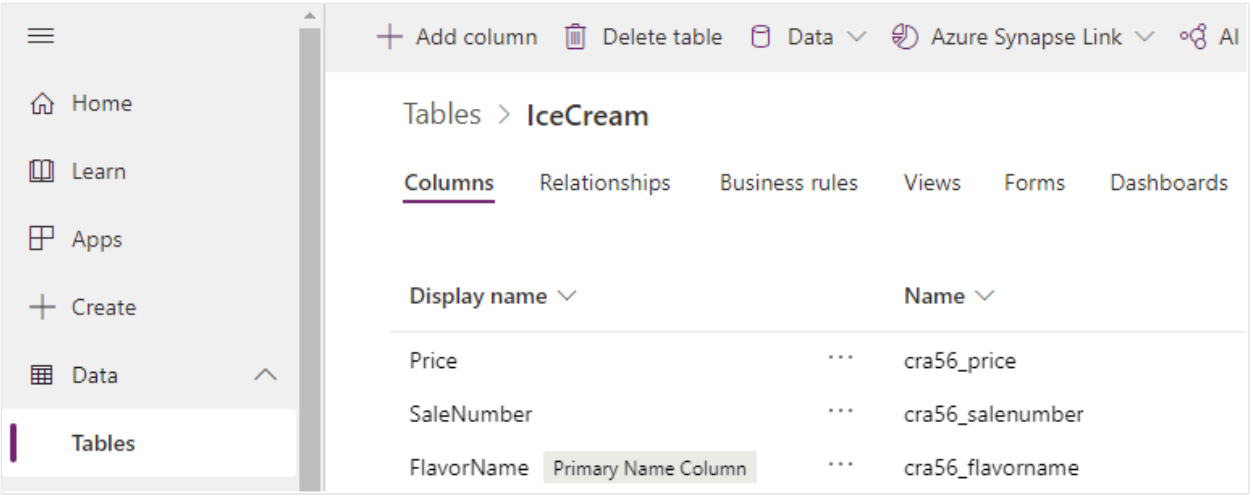
```
Power Fx

Table({Flavor: "Strawberry",UnitPrice: 1.99, QuantitySold:20})
```



The app consuming this component has the following **IceCreams** table that doesn't match with the component schema:

Expand table

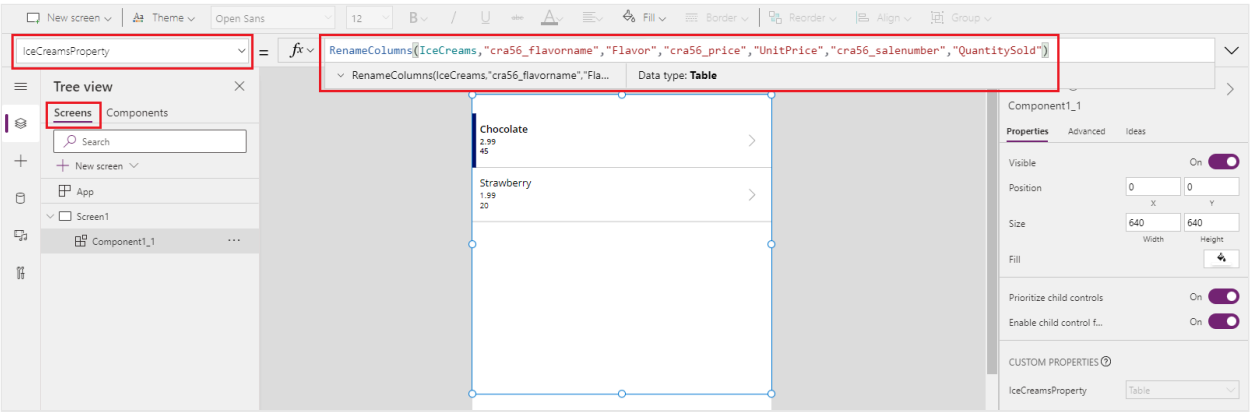
FlavorName	Price	SaleNumber
Strawberry	1.99	20
Chocolate	2.99	45



To map the correct fields, use **RenameColumn()** function to rename expected columns.

Power Fx

```
RenameColumns(IceCreams,"cra56_flavorname","Flavor","cra56_price","UnitPrice","cra56_salenumber","QuantitySold")
```



The input fields that the component expects are now mapped with the matching columns from the data source.

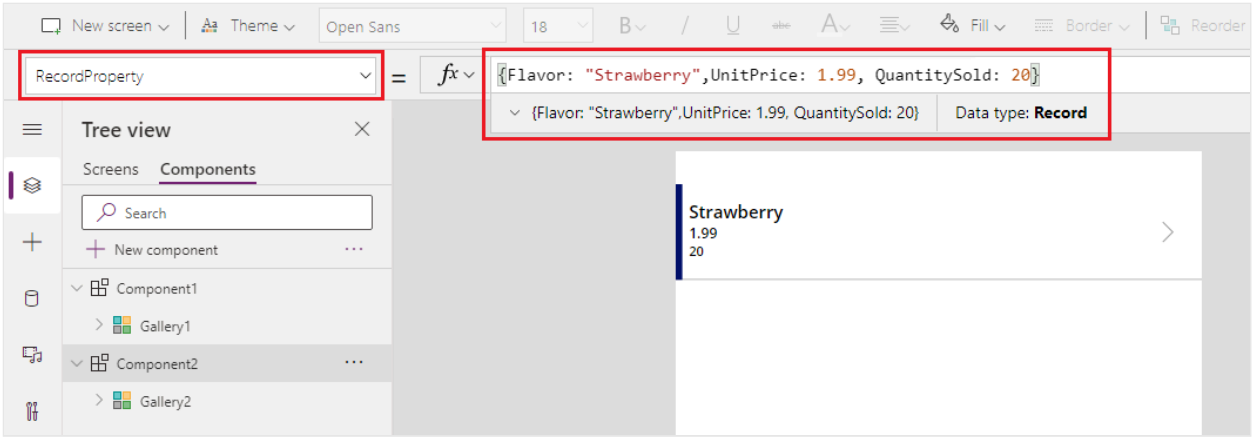
Map records

Use With() function to map a single record.

For example, continuing from the earlier example for mapping columns, the custom input property of a component inside an app expects a record type with the following schema:

Power Fx

```
{Flavor: "Strawberry",UnitPrice: 1.99, QuantitySold: 20}
```

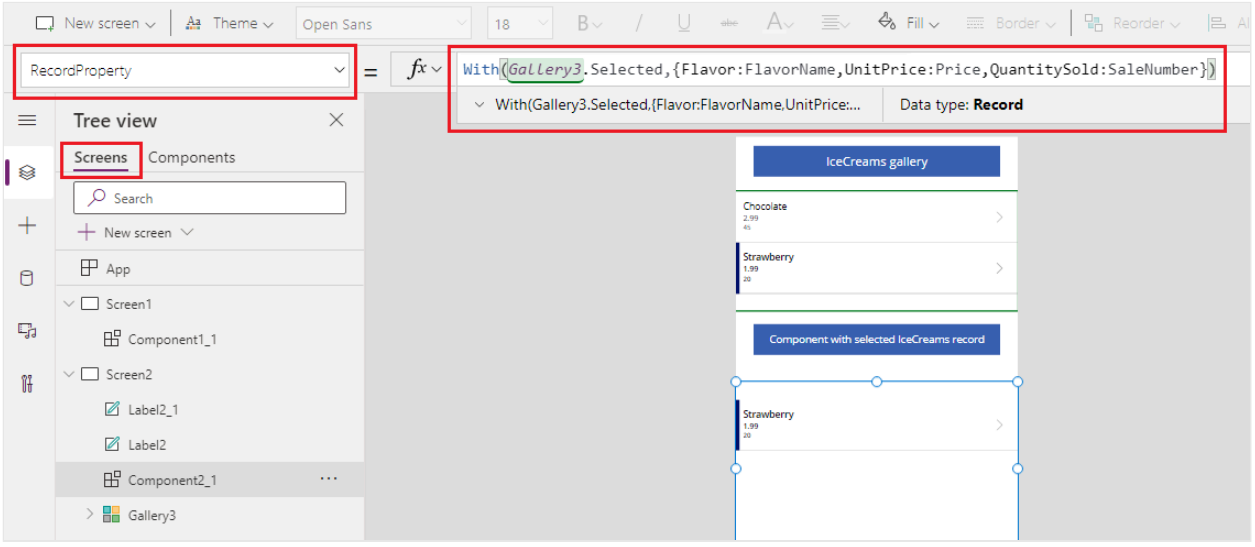


Since the **IceCreams** data source expects column names as **FlavorName**, **Price**, and **SaleNumber**, we'll need to change the mapping for the record once the component is added to the app.

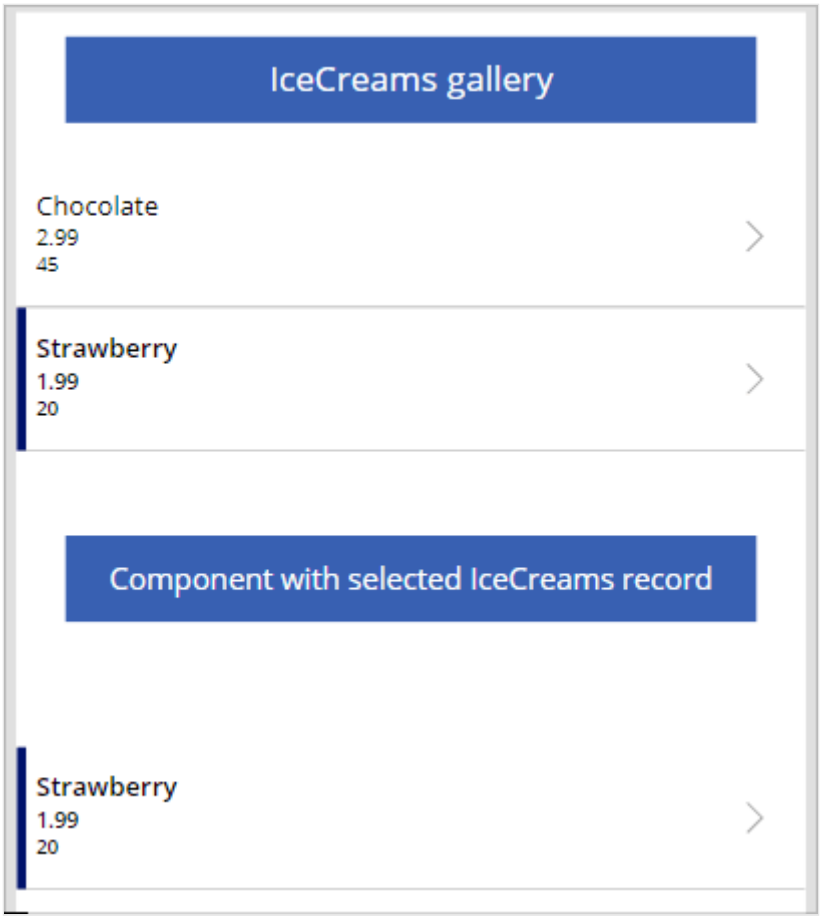
Use **With()** function to select the columns of the **IceCreams** table, and map them to the input fields of the component:

Power Fx

```
With(Gallery3.Selected,
{Flavor:FlavorName,UnitPrice:Price,QuantitySold:SaleNumber})
```



The following animation shows the example of a component added to the app that shows the selected record from the gallery (above the component):



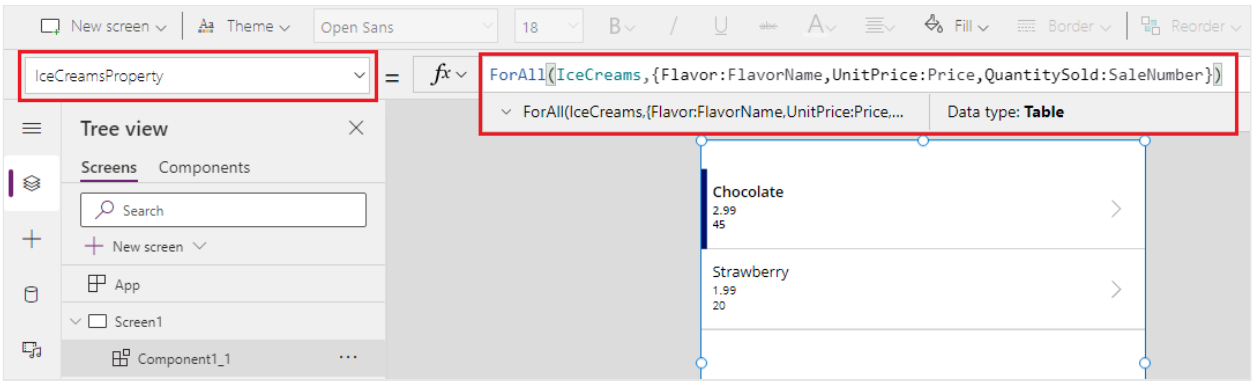
Map tables

Use ForAll() function to map a table of records with the records expected by the component added to the app.

For example, at the end of the map columns example, you can use the **ForAll()** function to map the entire table with the specific columns to point to the fields from the component for each row:

Power Fx

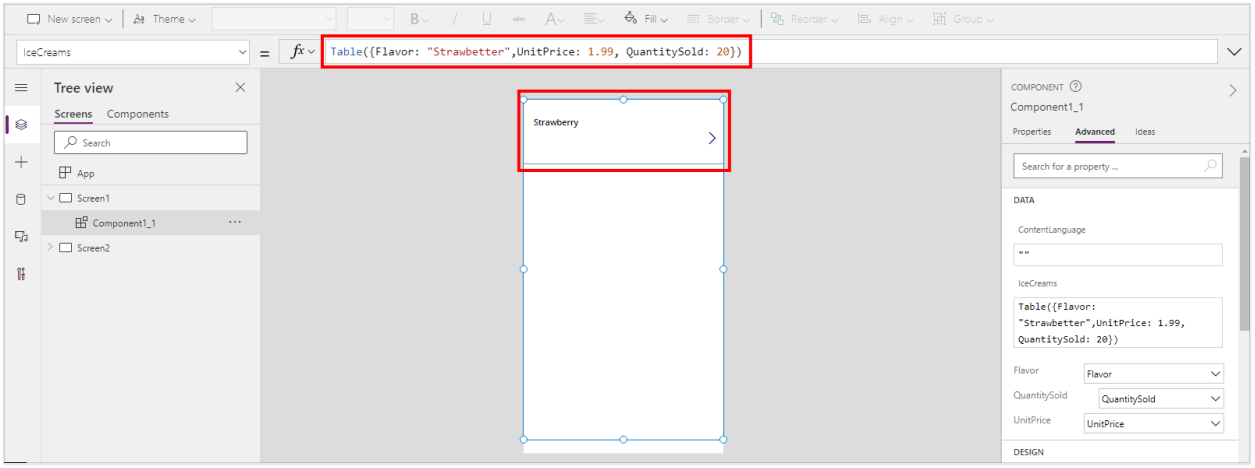
```
ForAll(IceCreams,
{Flavor:FlavorName,UnitPrice:Price,QuantitySold:SaleNumber})
```



Map using dropdown (deprecated)

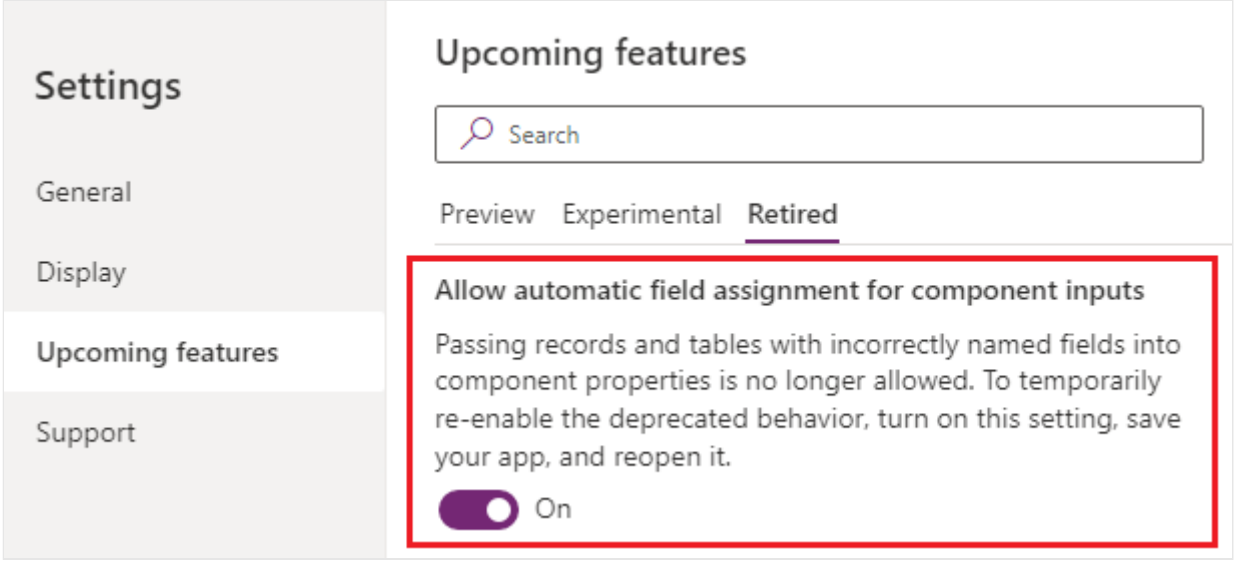
You can use the **Advanced** tab for a component added to an app that expects input property of table or record type, and select the field mappings using the dropdown option. This mapping is assigned by default, and unless you manually select the appropriate mapping fields, the mapping might not yield expected results.

For example, the following animation shows the formula being updated to rename columns. The component on screen doesn't change the field mapping since the default field selection using this dropdown needs to be updated manually.



This method of selecting the mapping using the dropdown has been deprecated. Instead, use the column, record, or table mapping methods as described earlier in this article.

You can still use the dropdown option to choose mapping for existing apps, though not recommended. For new apps, this option will be unavailable by default. To turn this deprecated capability on or off, go to **Settings** > **Upcoming features** > **Retired**, and choose **Allow automatic field assignment for component inputs**.



When turning the setting on, ensure you save and reopen the app in Power Apps Studio. Once reopened, update the input property formula to trigger formula validation so that the dropdown options start to appear in the Advanced properties pane.

See also

- Behavior formulas for components
- Deprecated: Map component fields using dropdown option
- Canvas components
- Component library
- Component library application lifecycle management (ALM)
- Add multimedia to a component
- Behavior formulas for components
- Power Apps component framework
- Add canvas components to a custom page in a model-driven app

---

Feedback

Was this page helpful? Yes No

Provide product feedback

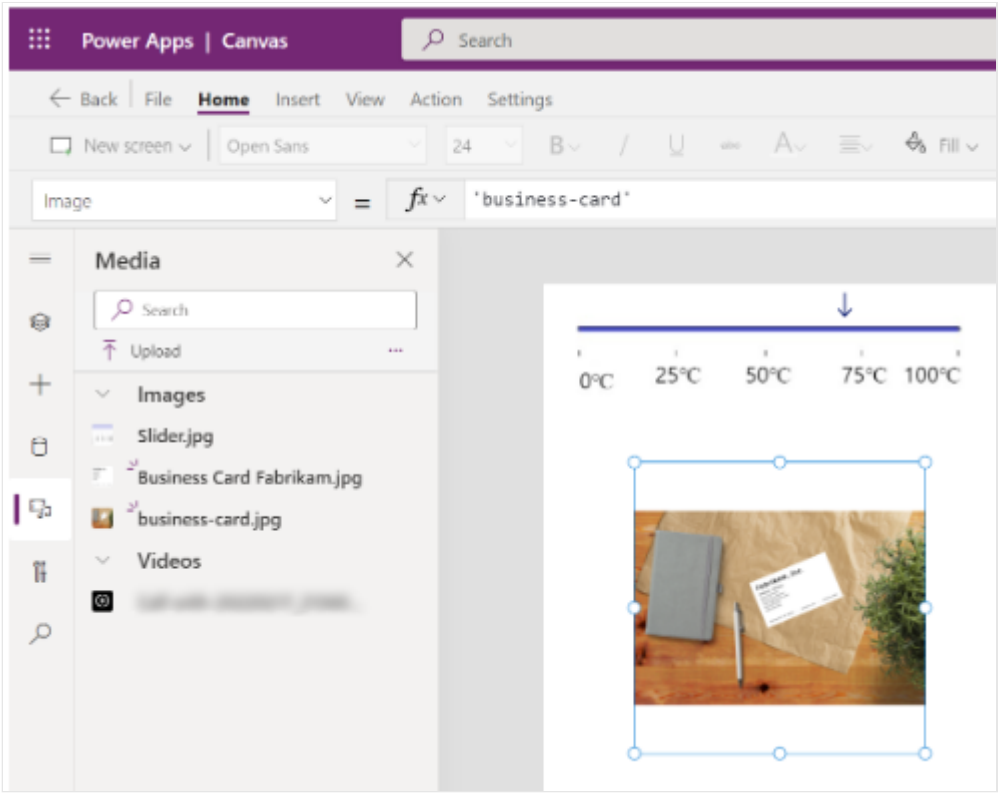
Add multimedia to a component

Article • 01/24/2023

You can add multimedia files to the components inside a library. These components can then be used by all apps in an environment and can be moved across environments using standard component application lifecycle management (ALM) through solutions in Microsoft Dataverse.

To add multimedia to a component, use the same steps that you'd follow when adding multimedia files in a canvas app. More information: Using multimedia files in canvas apps

After adding multimedia files to a component inside the component library, the files will appear under the media pane as they do for canvas apps:



Media resources in a component library are defined at the library level, and not tied to individual components. Importing any component from the component library will import all media resources from the library to the app. However, you can remove the unused media when app customization is complete. More information: Remove unused media

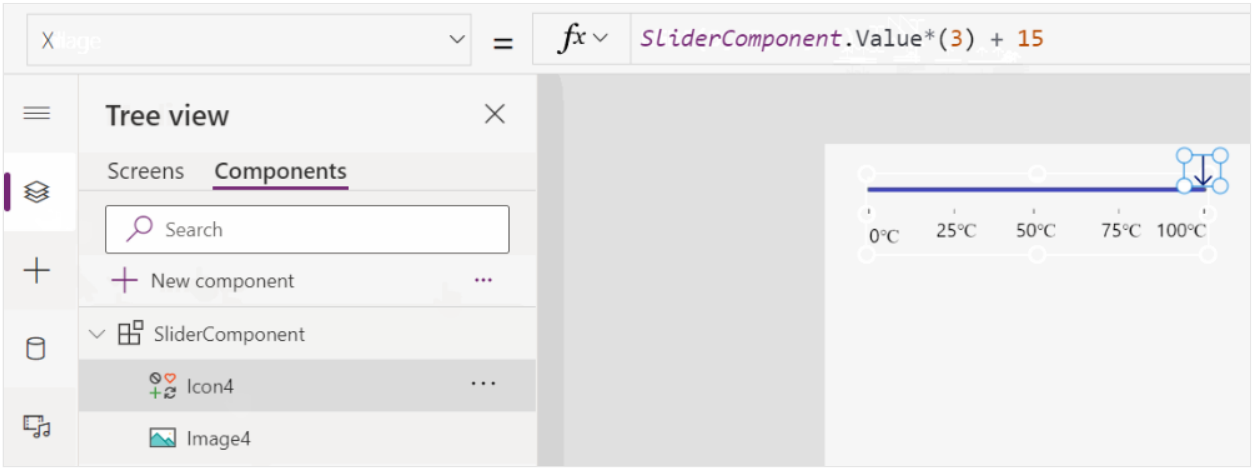
Tip

Prefix the media resources in a component library with a unique namespace or library name so that those names don't conflict with existing resources in an app during import.

During the component import to your app, if the component library multimedia has a resource with the same name as the multimedia that already exists inside your app, the import process adds an underscore ("_") and a random number to enforce name uniqueness.

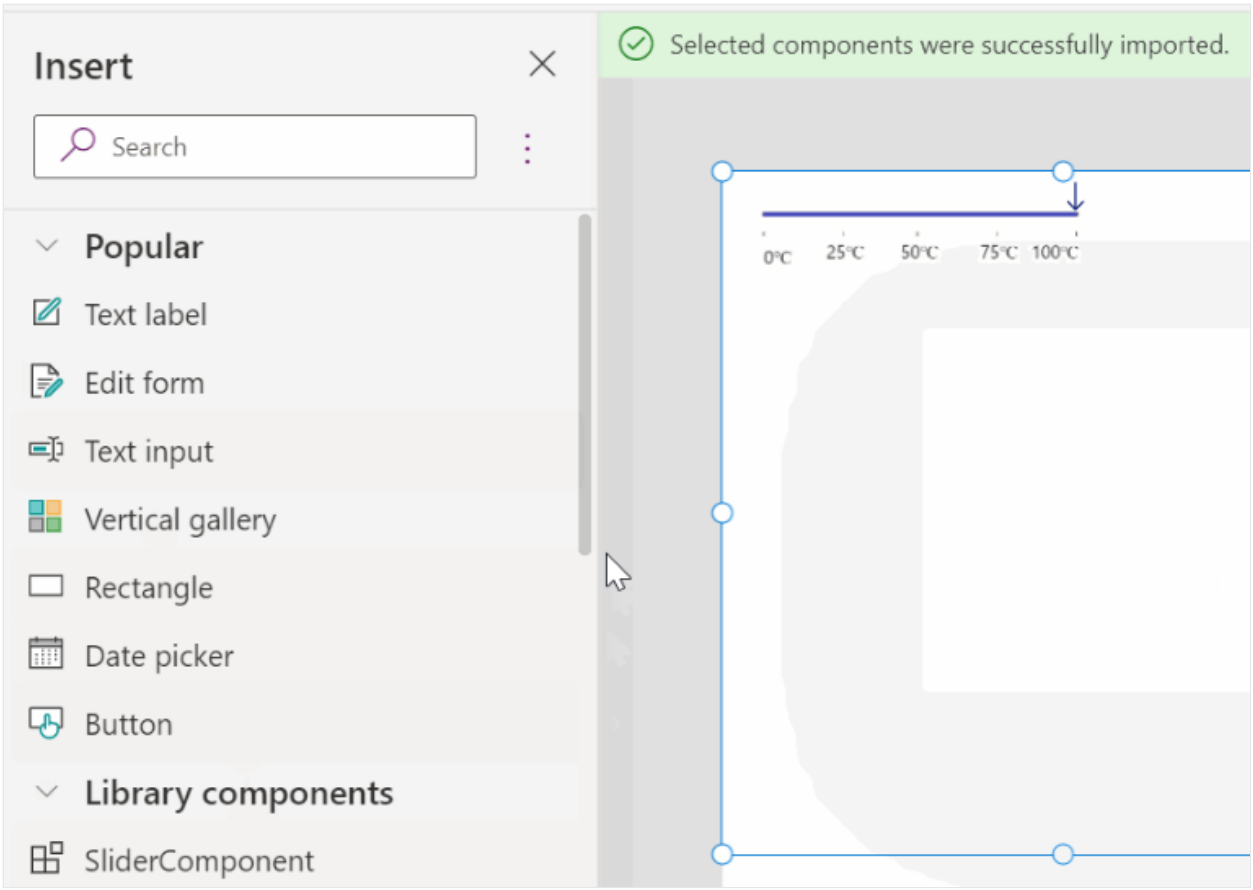
Example

Consider that you created a component library with a slider image that uses an icon with an input property for the component named "value" to set it's `X` property.

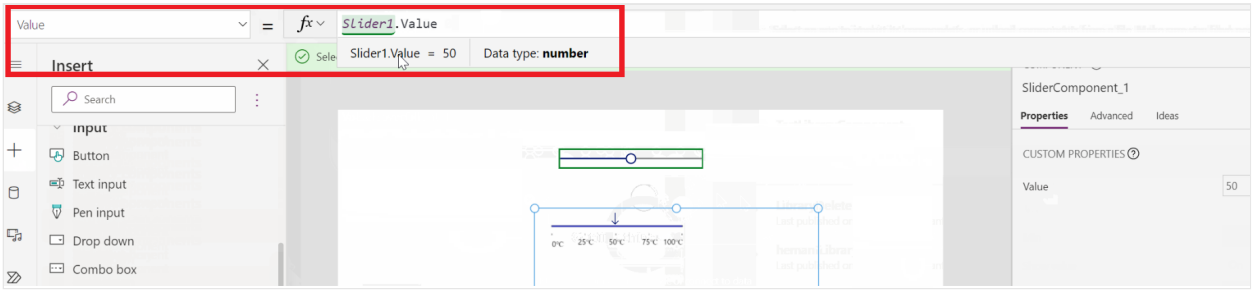


To reuse this multimedia component in your app, all you have to do is import the component, and link the component's input property with the value you'll supply from within the app.

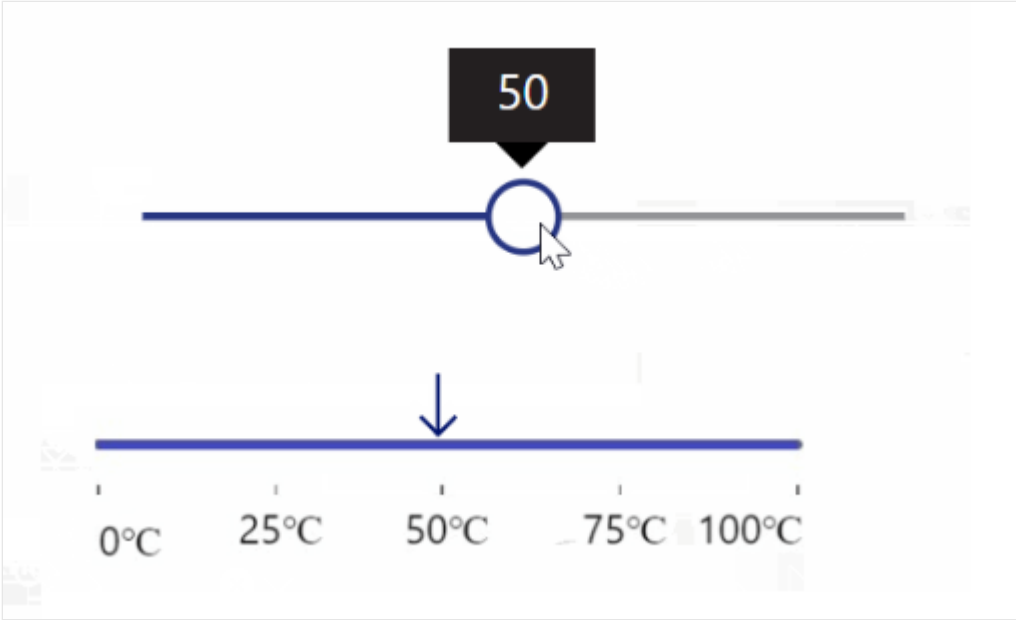
Start with importing the component to your app:



After the import, you can insert a slider control, and set the imported component's value to the slider's value:



When you play this app, the slider movement in the app would then move the imported slider component:



See also

- Canvas components
- Component library
- Component library application lifecycle management (ALM)
- Behavior formulas for components
- Power Apps component framework
- Add canvas components to a custom page in a model-driven app

Behavior formulas for components (experimental)

Article • 01/24/2023

[This article is pre-release documentation and is subject to change.]

> **Important**
>
> This is an experimental feature. For more information, see **Experimental and preview features**.

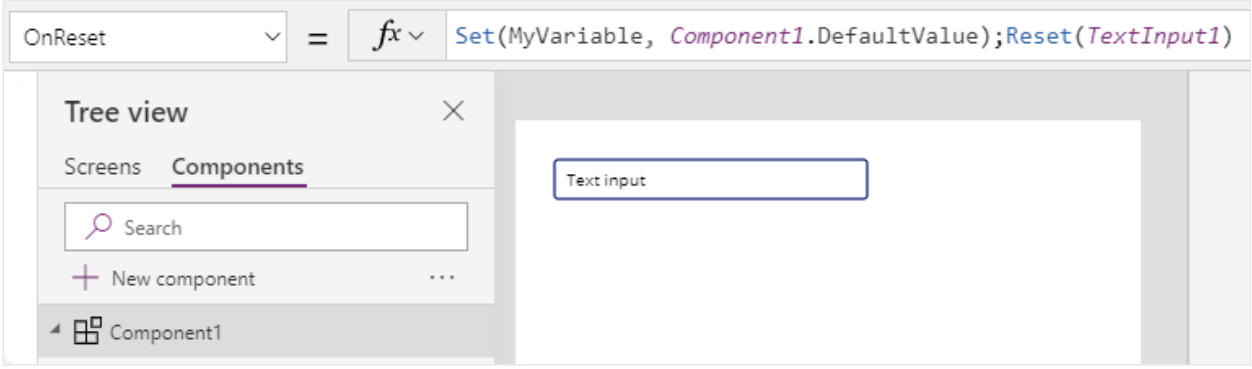
Specify one or more behavior formulas that run when an event triggers a change in component instances.

For example, set a component's **OnReset** property to one or more formulas that do initialization and clear input. Reset values when the **Reset** function runs on the component instances.

> **Note**
>
> The feature to create custom behavioral properties is currently experimental. However, you can use the default **OnReset** property that is available on all components in your production environment.

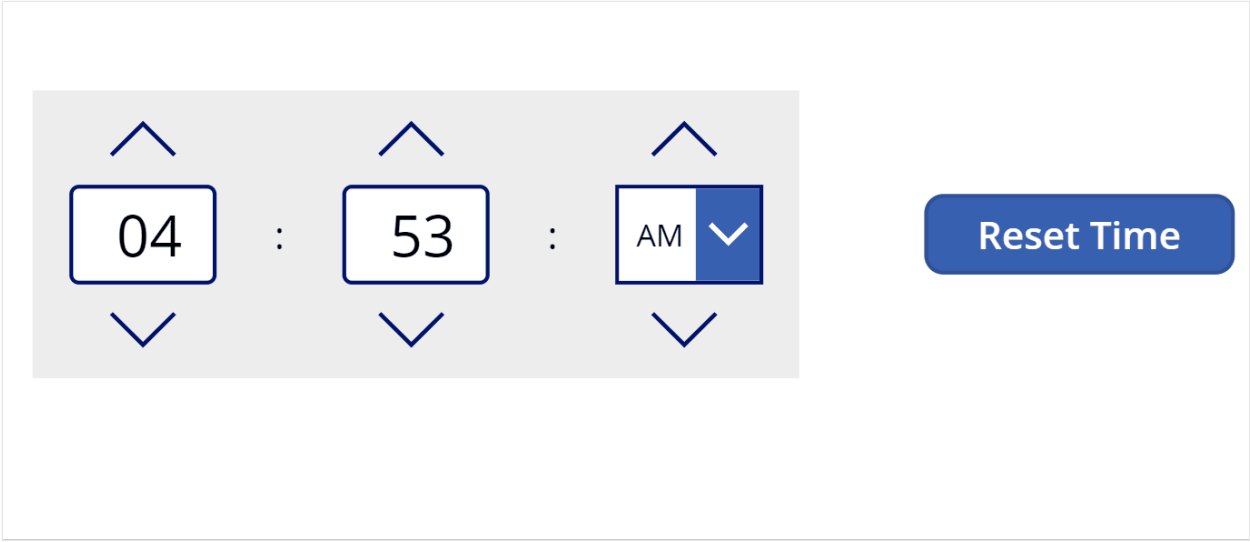
OnReset

With a component master selected, select **OnReset** in the drop-down list of properties (on the left side of the formula bar), and then enter one or more formulas.



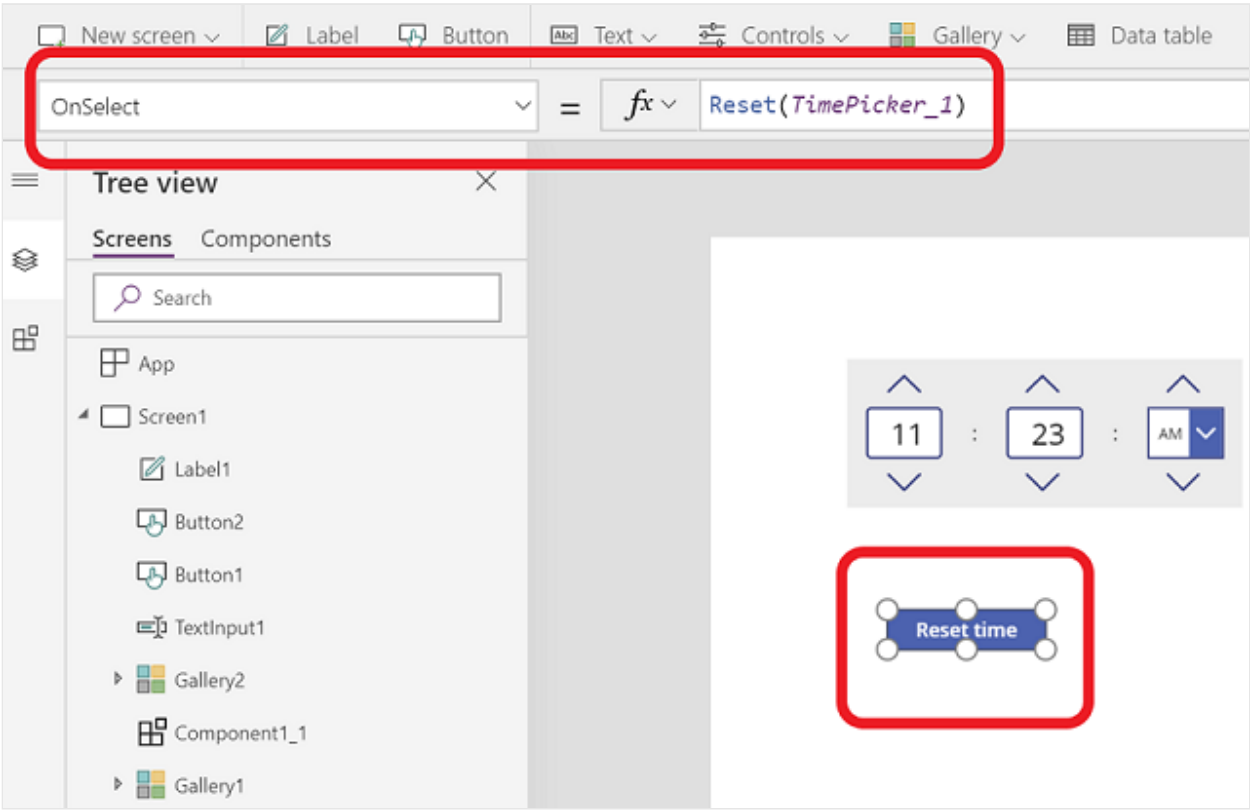
To test **OnReset**, configure a control to reset the component. For example, set the **OnSelect** property of a button to this formula: **Reset**(*ComponentName*).

Example - Reset timer



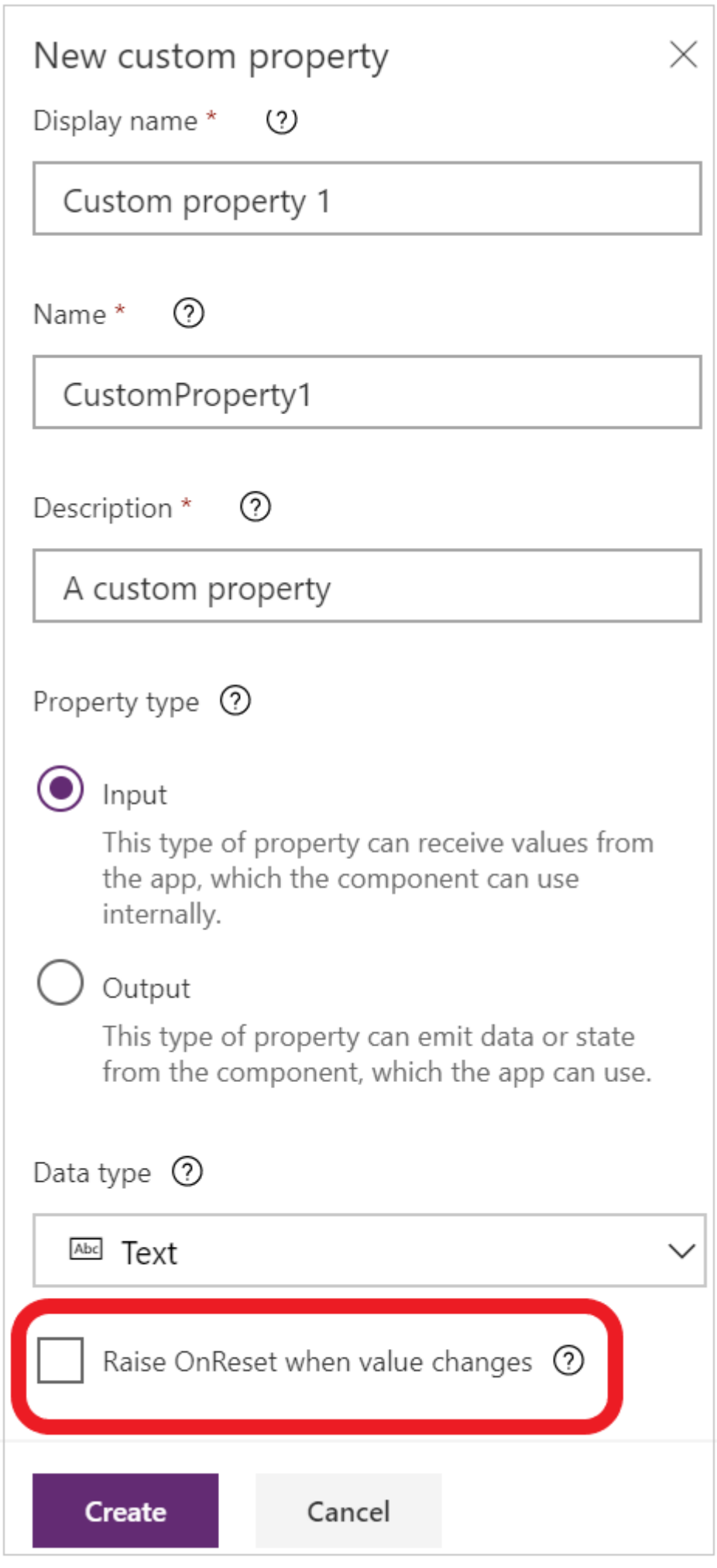
In this time picker component, two variables are used to display the time _selectedHour and _selectedMinute. When the picker gets reset, these variables should be reset to a default value, say 12: 12. The OnReset property for the component has the following formula: **Set(_selectedHour,12); Set(_selectedMinute,12)**

To trigger reset, go to a screen and insert an instance of the component. Add a button and configure OnSelect of the button to call **Reset(TimerComponent_instance)** to trigger OnReset.

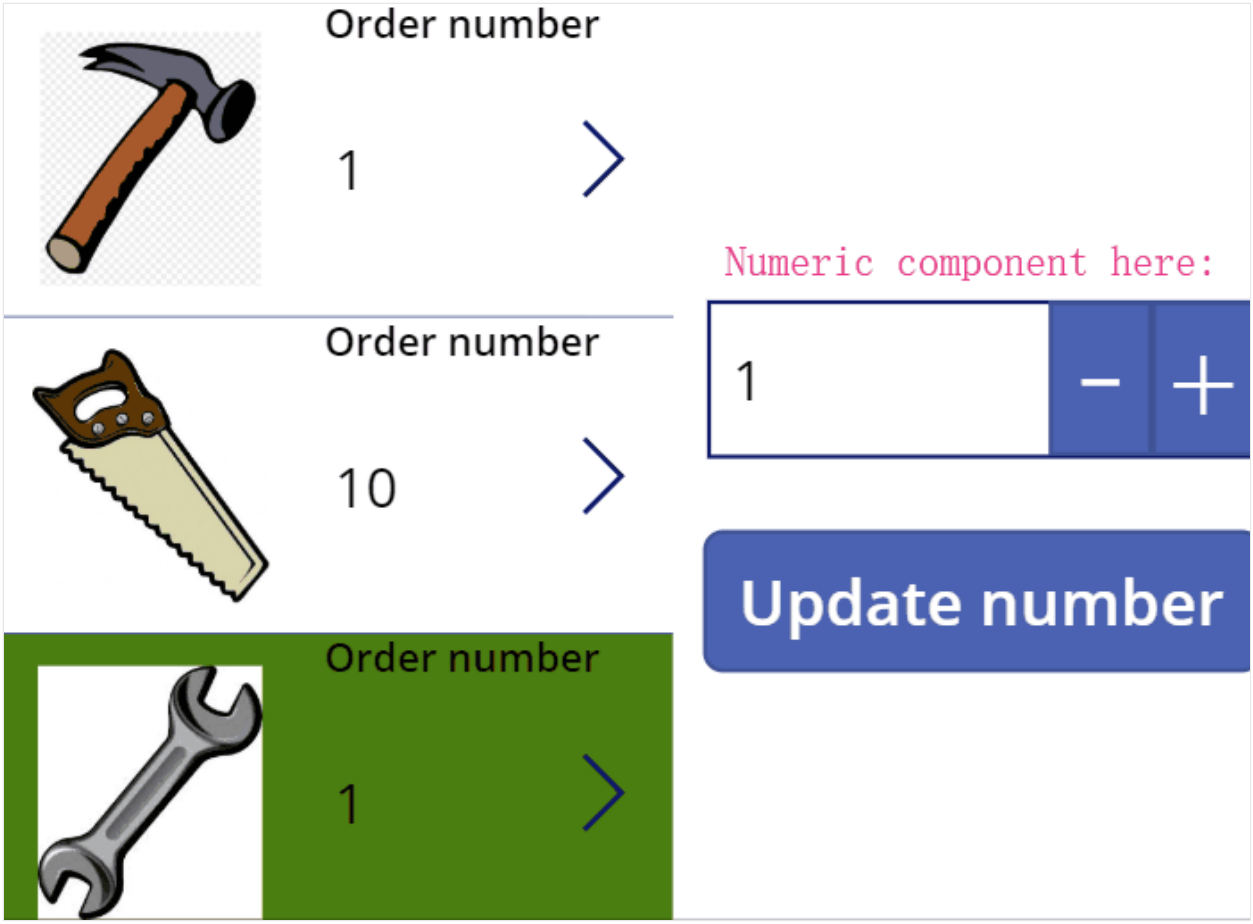


Update OnReset using custom property

Besides resetting a component instance from outside of the component, there's another method to trigger the OnReset from the inside. "**Raise OnReset when value changes**" is an option when creating a custom input property. It allows the value changes of this property to trigger OnReset of the component. This method is designed to set and reset the default value easily.


New custom property
Display name *
Custom property 1
Name *
CustomProperty1
Description *
A custom property
Property type
Input
This type of property can receive values from the app, which the component can use internally.
Output
This type of property can emit data or state from the component, which the app can use.
Data type
Text
Raise OnReset when value changes
Create
Cancel

Example



The example above shows reviewing order numbers and updating the numbers. The numeric up and down component is used to increase or decrease number of orders. When selecting the gallery on the left, the default number of the numeric up and down component is reset to display the order number of the selected tool. **Raise OnReset when value changes** made it possible to reset the default value when the input changes.

To do so, check **Raise OnReset when value changes** of the default input property. **OnReset** of the component is set to **Set(_numericValue,'Numeric up down'.DefaultValue)**. _numericValue is the variable to store the value of the current order value. Set the **Default** of the text input control to **If(IsBlank(_numericValue), 'Numeric up down'.DefaultValue, _numericValue)**.

See also

- Canvas components
- Component library
- Component library application lifecycle management (ALM)
- Behavior formulas for components
- Power Apps component framework
- Add canvas components to a custom page in a model-driven app

Canvas component properties

06/07/2025

Canvas components let makers create different types of properties to pass values or logic between the component and the app that hosts the component. Properties are essential for building interactive, reusable components.

Prerequisites

This feature is on by default for new apps. For existing apps, you need to turn it on in your app settings.

1. Open your canvas app for editing in Power Apps Studio. On the command bar, select **Settings** > **Updates**.
2. On the **New** tab, find and turn on the **Enhanced component properties** setting.

Types of properties

There are four types of properties available to makers.

1. Data properties: Data properties pertain to data, like a color or text value. A **Data** property can be set to be **Input** or **Output**, which indicates if the component provides data to the app (**Output**) or the app provides data to the component (**Input**). **Data** properties are the only properties that participate in app data flow.
2. Function properties: Function properties are related to logic, such as performing a calculation based on specific parameters or altering text. A **Function** property can be set to be **Input** or **Output**, which indicates if the component provides a function the app can call (**Output**), or the app provides a function the component can call (**Input**). **Function** properties do not participate in an app's data flow, and cannot use component or app variables.
3. Action properties: Action properties are a type of property that deals with logic and behaves like an **Output Function**. Thus, the component has the logic defined, and that logic can use chained expressions and manipulate collections or variables ("behavior"). For example, a `Clear()` **Action** property could provide functionality the app can call to clear out some values in the component, or a `Save()` **Action** property that updates a datasource.
4. Event properties: Event properties are a category of property that involves logic and functions as an **Input Function**. Thus, the app defines the logic, which the component can call and that logic can use chained expressions and manipulate collections or variables

("behavior"). Typically these properties' names reflect an event such as `OnSelect` or `OnChanged`.

Data property

A data property's usage is easy to imagine. Standard controls in apps typically have several data properties, to set default values, text color, size, etc. Let's look at a simple example where you want to make properties of a control within the component available to the hosting app. In this example, the component is named `Component1`. Put a slider control inside the component. Add an **Input** property to specify the color from the consuming app, and an **Output** property so the component can tell the app the current value of the slider. This example assumes the component has a **Slider** control named `Slider`.

1. In the property pane of the component, select **New custom property**.
2. On the **New custom property** pane, enter display name `Slider Color`.
3. Select **Property type** of **Data** and select **Property definition** value **Input**.
4. Finally, from the **Data type** dropdown, select **Color**.
5. Click **Create**.

Next, select the `Slider1` **Slider** control. Find its **ValueFill** property and in the formula bar, enter `Component1.SliderColor`. `Component1` is the name of the component, and `SliderColor` is the name of the property you added.

Now the component has a `SliderColor` property that can be set in the consuming app to pass a color into the component and set the slider's **ValueFill** property.

You might also want to provide the value of the slider to the consuming app. To do this, add an **Output** property.

1. In the property pane of the component, select **New custom property**.
2. On the **New custom property** pane, enter display name `Slider Value`.
3. Select **Property type** of **Data** and select **Property definition** value **Output**.
4. Finally, from the **Data type** dropdown, select **Number**.
5. Click **Create**.

When a **Data** property is **Output**, the component provides the value to the consuming app. To set the formula, set the new `SliderValue` property in the component to the **Value** property of the slider.

1. In the property pane of the component, click on the `Slider Value` property.
2. In the formula bar, replace the default value of `100` with the following formula: `Slider1.Value`.

Now the component's `SliderValue` property reflects the value of the slider inside the component, which the consuming app can read.

Function property

A function property contains an expression that returns a value. Typically, the function takes arguments that it uses to calculate the value to return.

> ⓘ **Note**
>
> Function properties currently cannot access variables or component values and properties, and cannot trigger data flow. Any required values have to be passed in as arguments.

An **Output** function is a simple way to create a custom function for Power Apps. The component defines an **Output** function with an expression that takes some arguments and returns a value. This function can then be used in an app using the component's name (say `Component1`) by calling `Component1.MyFunction(arg1, arg2)`.

An **Input** function lets a consuming app provide logic to a component, similar to a function pointer or callback function. For example, if the component works with people's names, it can have an input function with arguments `firstname` and `lastname` and expects a string back. The app can define the function expression to return `$"{firstname} {lastname}"` or `$"{lastname}, {firstname}"`. The component calls the function the app maker provides and uses the returned string.

Action property

Action properties are similar to function properties of type **Output**, but they allow side-effect formulas and expression chaining. A component can have an action property named `AddRecord` that lets the app add a record to a collection inside the component, or a `Reset` action that clears variables or collections inside the component.

In the earlier slider example for the **Data property**, you can add an action property called `ResetValue` to set the slider back to its default value. Use the formula `Reset( Slider1 )` for this. Now, instances of the component in the app can call `Component1.ResetValue()` to set the slider back to the default value.

Event property

Many standard controls have common **Event**-type properties, which are input behavior function properties. **OnSelect** on the button control is a common example. A component can define event properties and call these events like a function. For example, a component with a button control can have an event property named `OnButtonClicked`. In the button control's **OnSelect**, the component can call its `Component1.OnButtonClicked()` property. A consuming app can define its own logic for this property to act when the button inside the component is pressed.

Default values for properties or arguments

Default values give an argument or property a value if none is set. This doesn't make sense for some types of properties. In some cases, you can use default values to show the expected schema for a record. By providing a default record, you set the record type as the expected schema.

For example, an **Action** property named `AddRecord` takes a record to add to a local collection. To show the schema of the expected record, add a default value for the `AddRecord` property's argument.

Overview of creating performant apps

06/27/2025

Creating performant Power Apps ensures your apps run efficiently and providing a better user experience. This article explains key patterns, anti-patterns to avoid, and design principles to help you build high-performing Power Apps for your organization.

- **App patterns and Power Apps productivity gain**: Power Apps makes it easier to build enterprise-grade apps by using established app patterns. Patterns are groups of Power Apps elements that work together, like data sources, collections, controls, Power Automate, and pages. Power Apps includes key performant patterns by default, so low-code developers quickly build functional enterprise apps. Deployment and administration tasks are also straightforward. With Power Apps, your team is more productive because many elements don't need your attention.

- **Power Apps steers towards performant patterns**: Power Apps guides you toward well known performant patterns by default. These patterns include streamlined data loading at launch, automatic incremental paging, caching data for collections, and loading only essential data for each page. These proven patterns work well for data-heavy enterprise apps. Many successful Power Apps implementations use more than 100 tables and over 50 screens while keeping excellent performance.

- **Falling into anti-patterns**: When you build an app on any development platform, you risk making it perform poorly because of anti-patterns. These patterns can cause slow loading, slow page transitions, and make it hard to update or get data. Common anti-patterns include loading too much data, turning everything into collections, and overloading OnStart. People often use these patterns to work around real or perceived Power Apps limitations. Even with guidance, you might still use a bad pattern and end up with a slow app.

Key performance design principles

When building your app, consider these key performance principles to ensure it runs efficiently. These principles cover most aspects needed to enhance your app's speed. Some performance suggestions might appear in multiple sections due to their interrelated nature.

- Optimize page loads: Optimize your apps for app and page load speed. Minimize, delay, or eliminate actions that prevent fast app or page load.
- Small data payloads: Keep the amount of data that is bulk retrieved small.
- Optimize query data patterns: Do data mashups on the server, not in your app.
- Fast calculations: Work with Power Fx, not against it.

For a deeper understanding, also see Execution phases of a Power App article.

Additional performance guidance

Other performance considerations can affect your app. For more information, see these articles:

- Other performance considerations: Discusses other factors that might affect performance.
- Build large complex apps: Lists key factors to consider when building a large app.
- Canvas app coding standards and guidelines : Lists general coding and development guidelines that help you build an app.

Use Monitor to get insights and recommendations for your app

Summarize this article for me

Monitor in Power Apps offers insights and recommendations to enhance app performance. It provides makers with detailed information on their app's performance and actionable steps to improve their apps.

Prerequisites

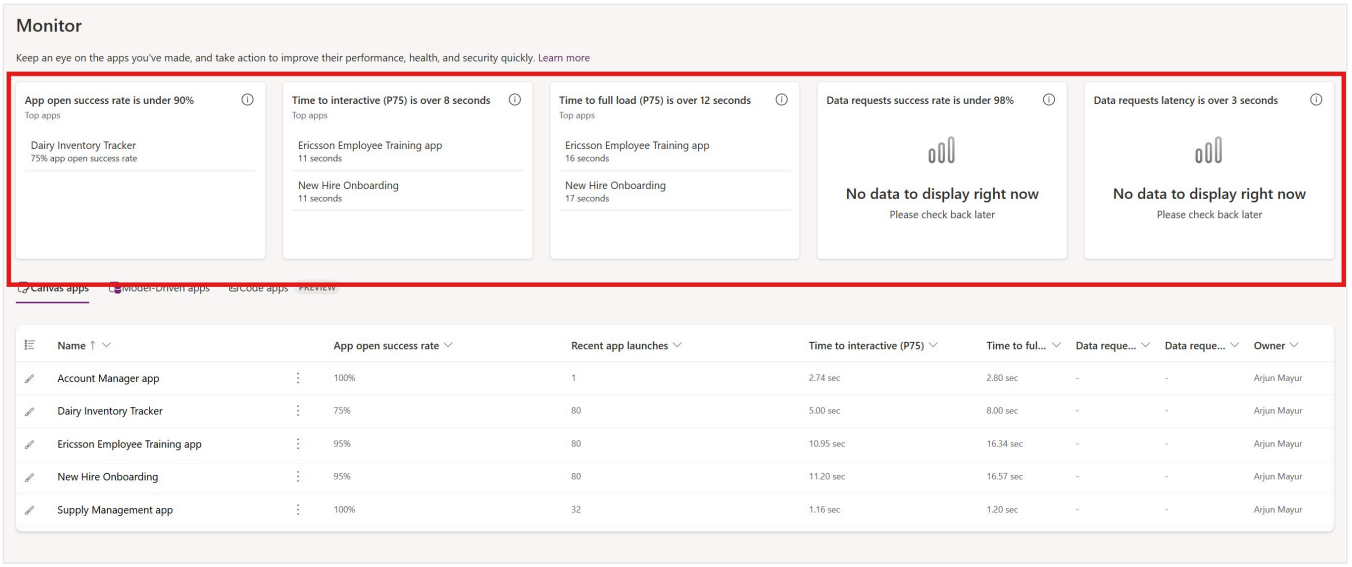
To use this feature, your administrator must enable certain analytics and monitoring settings in the Power Platform Admin Center. For more information, see Monitoring experiences for makers. When you sign in to Power Apps and open Monitor, make sure you've selected the correct environment and that you're a member of that environment with an assigned role (for example, Basic User). Otherwise, you won't be able to access Monitor.

Open and pin Monitor

1. Sign in to Power Apps.
2. On the left navigation, select **Apps**.
3. On the command bar, select **Monitor**.
4. On the left navigation pane, select the  pin button next to **Monitor**.

Use Monitor

The **Monitor** page displays metric cards at the top, highlighting the top three underperforming apps for each metric. If all apps perform well on a specific metric, the card doesn't list any apps. Use the tabs under the cards to switch between viewing your model-driven, canvas, and code apps.



Expand table

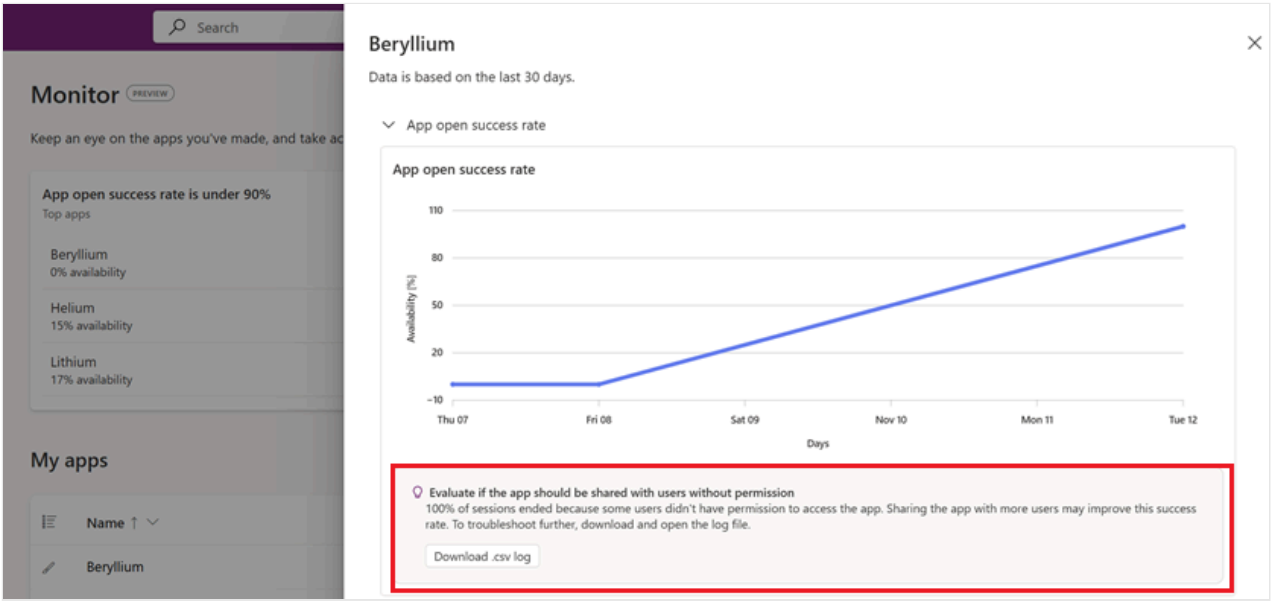
Metric	Definition	Available for Canvas apps	Available for Model-driven apps	Available for Code apps
App open success rate	Indicates the percentage of app launch attempts that successfully allowed end-users to access the app. This metric is calculated once every 24 hours.	Yes	Yes	Yes
App session count	The number of distinct user sessions in an app in one day. A session begins when a user opens the app and ends after a period of inactivity or when the app is closed.	Yes	Yes	Yes
Time to interactive (TTI)	Measures the time end-users wait to access the first screen of an app. It doesn't include the time users wait to see all data presented on the first screen, such as filling in rows of a gallery. This metric is calculated every 24 hours and is measured using percentiles, with the 75th percentile being reported.	Yes	No	Yes
Time to full load (TTFL)	Measures the time end-users wait to see all data displayed on the first screen of an app. In the app launch lifecycle, this metric follows the **Time to interactive** phase. This metric is calculated every 24 hours and is measured using percentiles, with the 75th percentile being reported.	Yes	No	No
Data request success rate	A percentage that describes how often data requests coming from an app are successful	Yes	No	No

Metric	Definition	Available for Canvas apps	Available for Model-driven apps	Available for Code apps
Data request latency	The average time (in seconds) it takes for all data requests made by the app to complete during a session.	Yes	No	No
Row summary dwell time	The time (in seconds) that end users spend on the expanded AI row summary of a main form.	No	Yes	No

View metrics and recommendations for an app

When you select an app on the **Monitor** page, a side pane opens and displays more data for each metric, including a data chart for the last 30 days.

1. On **Monitor** page, select an app.
2. To view recommendations for an app, select the app's name to open the side panel. The chart shows a callout with a recommended action for improving the metric.



> **Important**
>
> Recommendations are only available in Managed Environments. Contact your admin to enable Managed Environments for access to recommendations.

Last updated on 03/05/2026

Analyze system-generated logs using Application Insights

Article • 04/09/2025

Connect your canvas apps to Application Insights, which is a feature of Azure Monitor. Application Insights includes analytics tools that help diagnose issues and understand user behavior in apps. Use the collected information to make better business decisions and improve app quality.

Prerequisites

- You need access to the Azure portal.
- You need permission to create Azure resources.

> **Note**
>
> To view telemetry information, your tenant admin needs to enable **Canvas app insights**. Sign in as an admin at the Power Platform admin center. Select **Settings** > **Tenant settings** > **Canvas app insights**. In the **Canvas app insights** pane, turn the toggle **On** and save your changes. Learn more in Tenant settings.

Create an Application Insights resource

Send system-generated logs from an app by creating an Application Insights resource to store events.

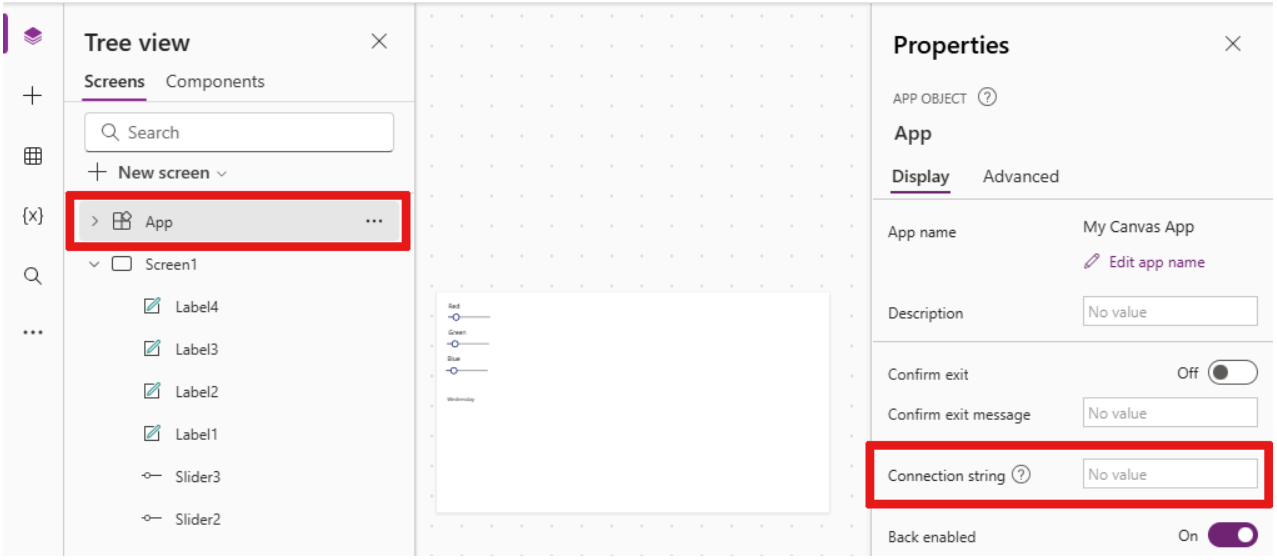
Learn how to create a workspace-based resource for Application Insights in the Azure portal.

Connect your app to Application Insights

> **Note**
>
> - When you specify a connection string, remember that data can be sent across tenants. Trace events are sent to the App Insights resource that corresponds to the connection string you set for your app, even if the target App Insights instance is in a different tenant than the app.

- Be cautious when importing existing .msapp files because connection strings for App Insights might be present. Manually open the app after import to check that the correct App Insights connection string is used.

1. Sign in to Power Apps.
2. Open an app to edit it.
3. Select the **App** object in the left navigation tree view, and paste the **Connection string** from your Application Insights resource.



4. **Save** and **Publish** your app.
5. **Play** the published app and browse the different screens.

When you browse the app screens, events are automatically logged to Application Insights. These include usage details like:

- Where users access the app from
- Which devices users use
- The types of browsers users use

> **Important**
>
> Play the published app to send events to Application Insights. Events don't send to Application Insights when you preview the app in Power Apps Studio.

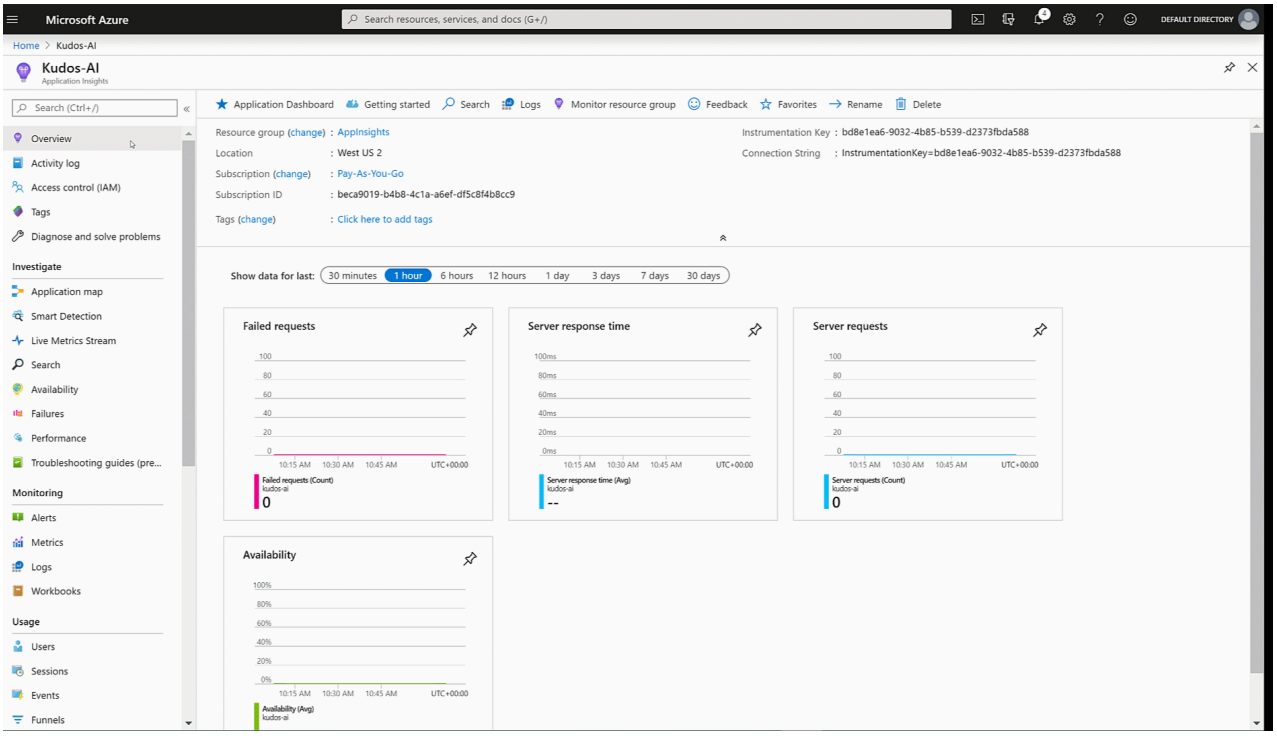
View events in Application Insights

1. Sign in to the Azure portal and open the Application Insights resource you created previously.

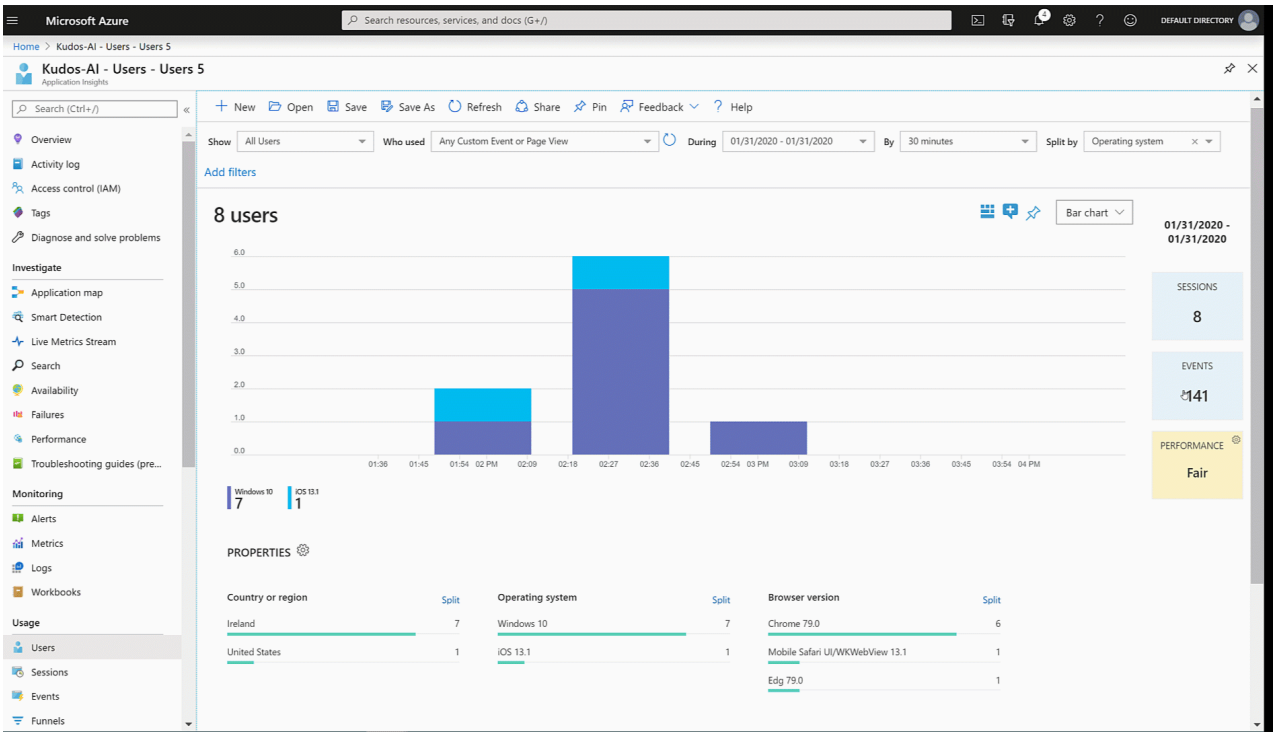
2. In the left navigation pane, select **Users** under the **Usage** section.

> **Note**
>
> The **Users** view shows app usage details, such as:
>
> - Number of users who viewed the app
> - Number of user sessions
> - Number of events logged
> - Users' operating systems and browser version details
> - Users' region and location
>
> Learn more about users, sessions, and events analysis in Application Insights.

3. Select a user session to view specific details, such as session length and screens visited.



4. In the left navigation pane, select **Events** under the **Usage** section. You can view a summary of all the screens viewed across all app sessions.



> **Tip**
>
> Explore more Application Insights features, such as:
>
> - Funnels
> - Cohorts
> - Impact analysis
> - Retention analysis
> - Usage flows

Create custom trace events

Write custom traces to Application Insights to analyze information specific to your app. The Trace function allows you to collect:

- Detailed usage information for controls on the screens
- Which specific users are accessing your app
- What errors occur

Tracing helps you diagnose issues by sending a trail of information as users browse your app and take actions. Trace messages sent to Application Insights have one of three severities:

- Information
- Warning
- Error

Send a trace message with the appropriate severity based on the situation. You can query the data and take specific actions based on the severity.

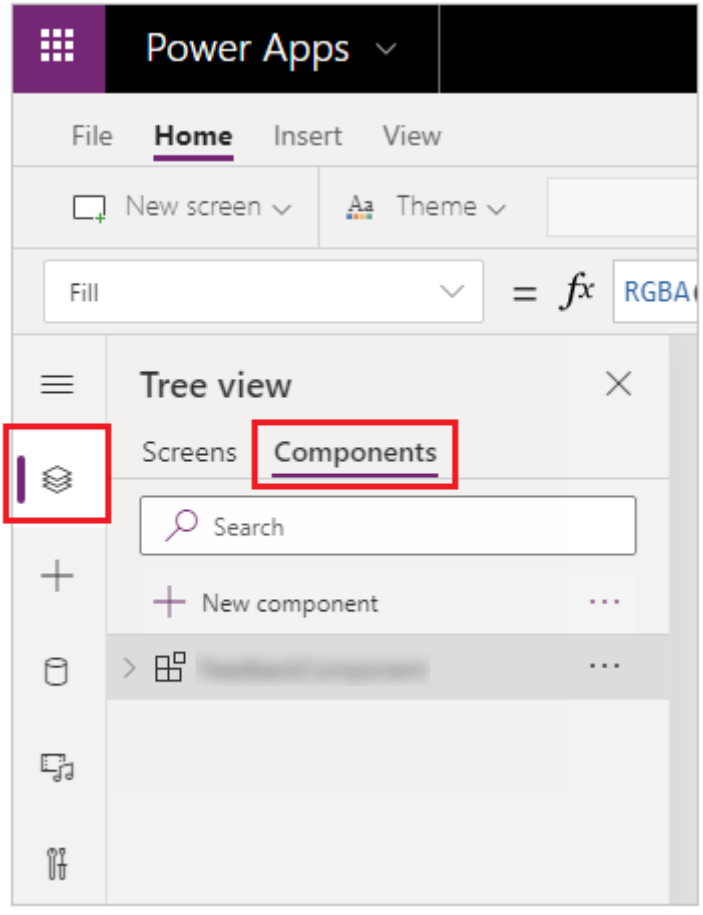
> **Note**
>
> If you are logging any personal data, be aware of your obligations with respect to various privacy laws and regulations. Refer to the Microsoft Trust Center and the Service Trust Portal for more information.

Create a component in your app to collect feedback on each screen and log events to Application Insights.

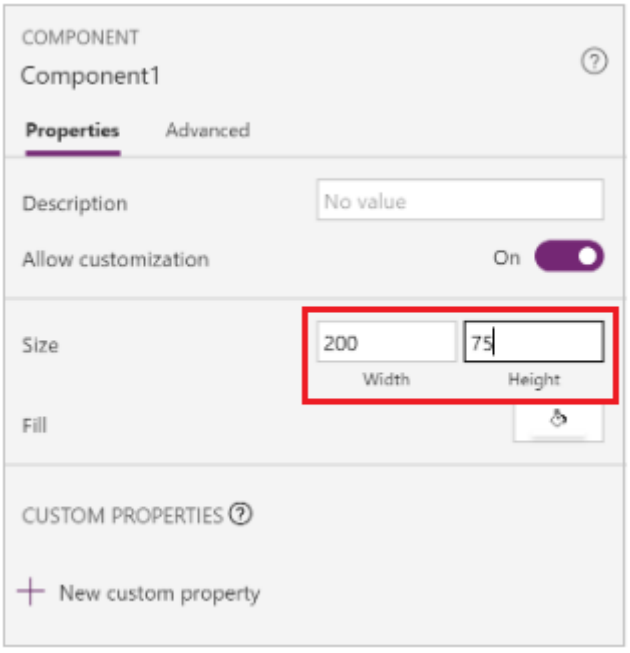
1. Sign in to Power Apps.
2. Select **Apps** in the left navigation pane. From the list of apps, select the **Kudos** app, and then select **Edit**.

   > **Note**
   >
   > You can also create a new app or edit an existing app instead.

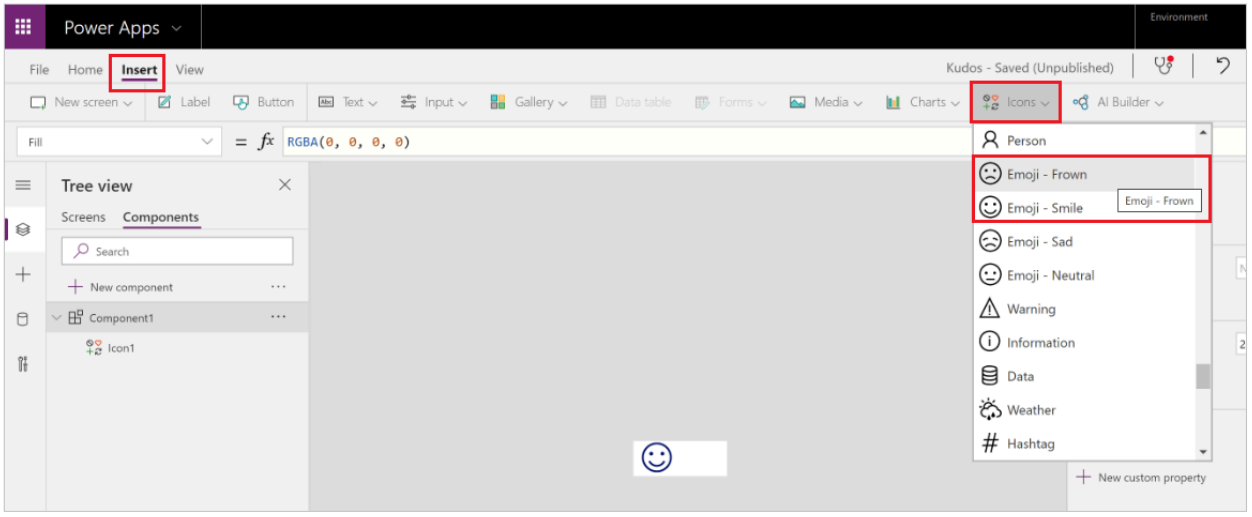
3. Select **Components** in the **Tree view**:



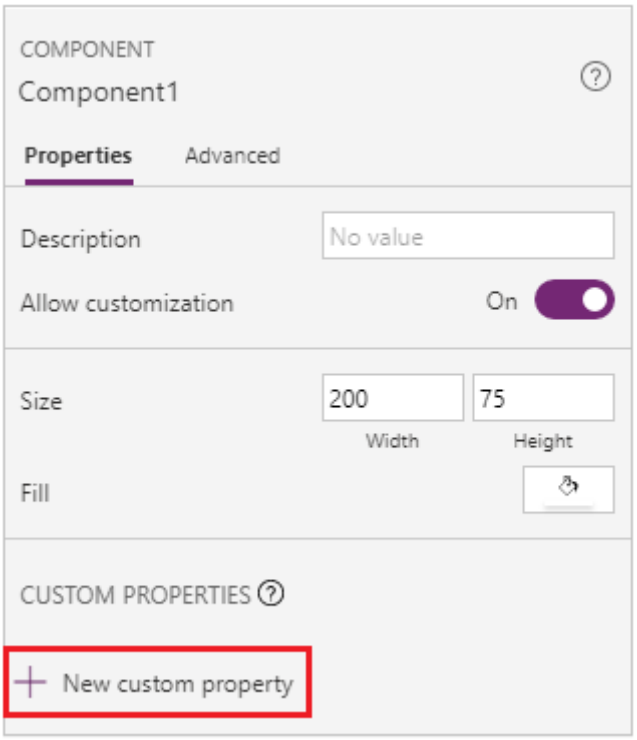
4. Select **New component**, then resize the width to 200 and the height to 75:



5. Select **Insert** from the menu and then select **Icons** to add *Emoji - Frown* and *Emoji - Smile*:



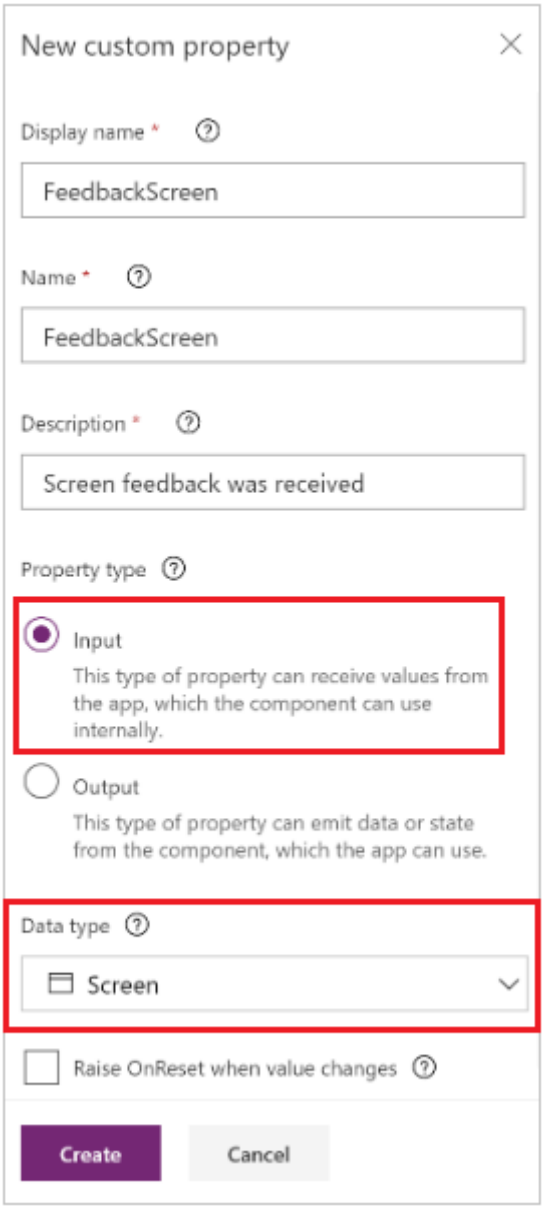
6. Select **New custom property** to create a custom property:



7. Enter the property *Name* and *Display name* (for example, *FeedbackScreen*).

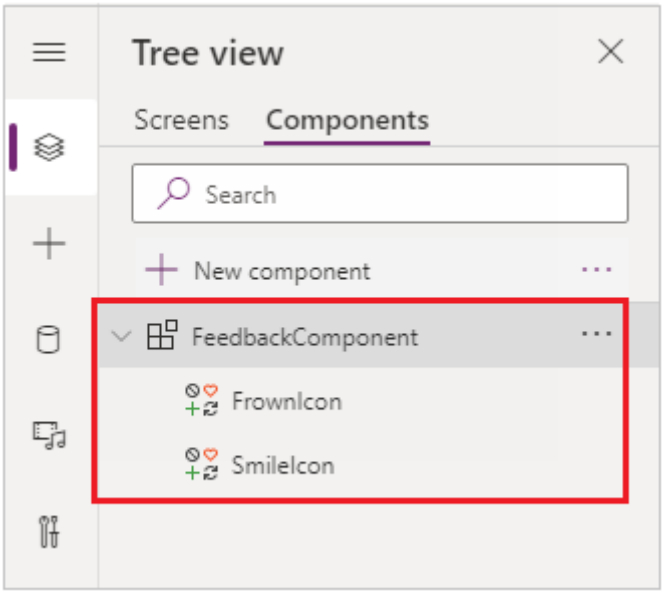
8. Enter a property *Description*.

9. Select **Property type** as **Input** and **Data type** as **Screen**:



> **Note**
>
> An input property lets you capture the screen name and its component to log this information to Application Insights.

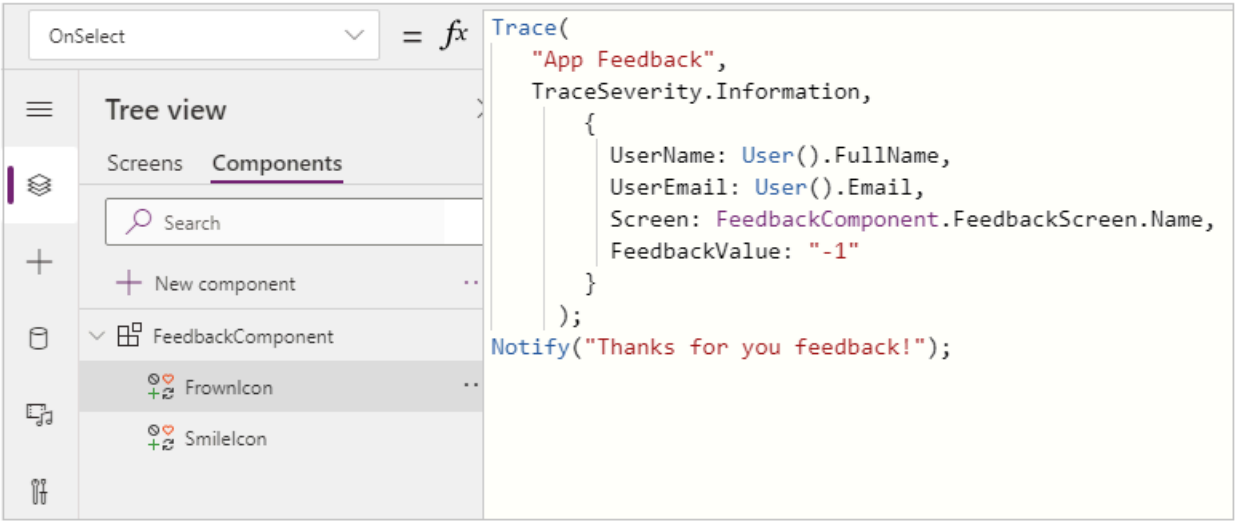
10. Select the component in the **Tree view**, select **More actions** (...), and select **Rename** to give the component a meaningful name, such as *FeedbackComponent*.



11. Select the icons, select **More actions** (...), and then select **Rename** to rename the icons with meaningful names, such as *FrownIcon* and *SmileIcon*.

12. Select **FrownIcon**, select the **OnSelect** property, and enter the following expression in the formula bar:

```
Power Fx

Trace(
   "App Feedback",
   TraceSeverity.Information,
       {
         UserName: User().FullName,
         UserEmail: User().Email,
         Screen: FeedbackComponent.FeedbackScreen.Name,
         FeedbackValue: "-1"
       }
     );
Notify("Thanks for your feedback!");
```

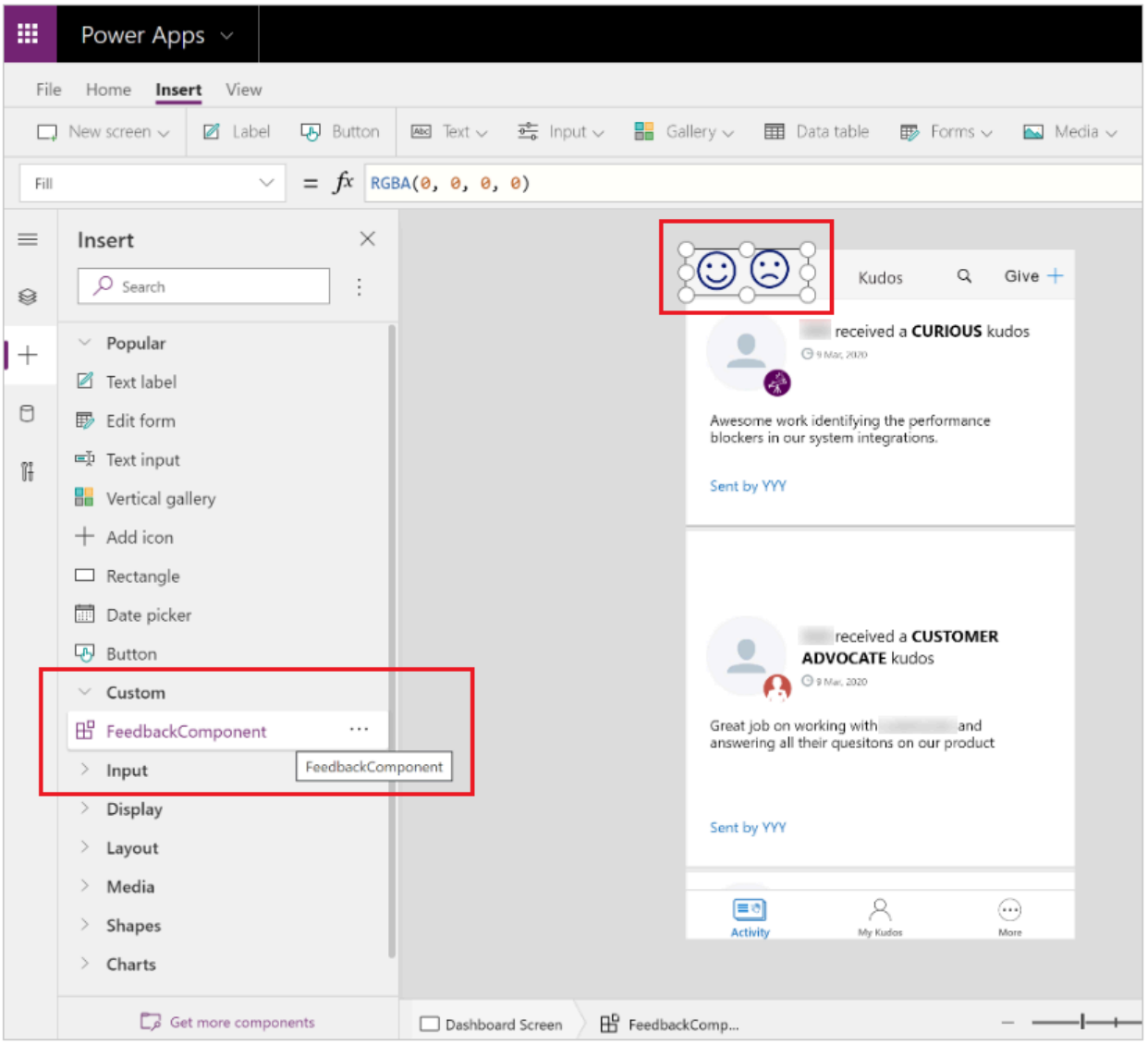


> **Note**
>
> The formula sends *UserName*, *UserEmail*, *Screen*, and *Feedback* (with the value *-1*) to Application Insights.

13. Select **SmileIcon**, select the **OnSelect** property, and then enter the following expression in the formula bar:

```
Power Fx

Trace(
   "App Feedback",
   TraceSeverity.Information,
       {
         UserName: User().FullName,
         UserEmail: User().Email,
         Screen: FeedbackComponent.FeedbackScreen.Name,
         FeebackValue: "1"
       }
     );
Notify("Thanks for your feedback!");
```

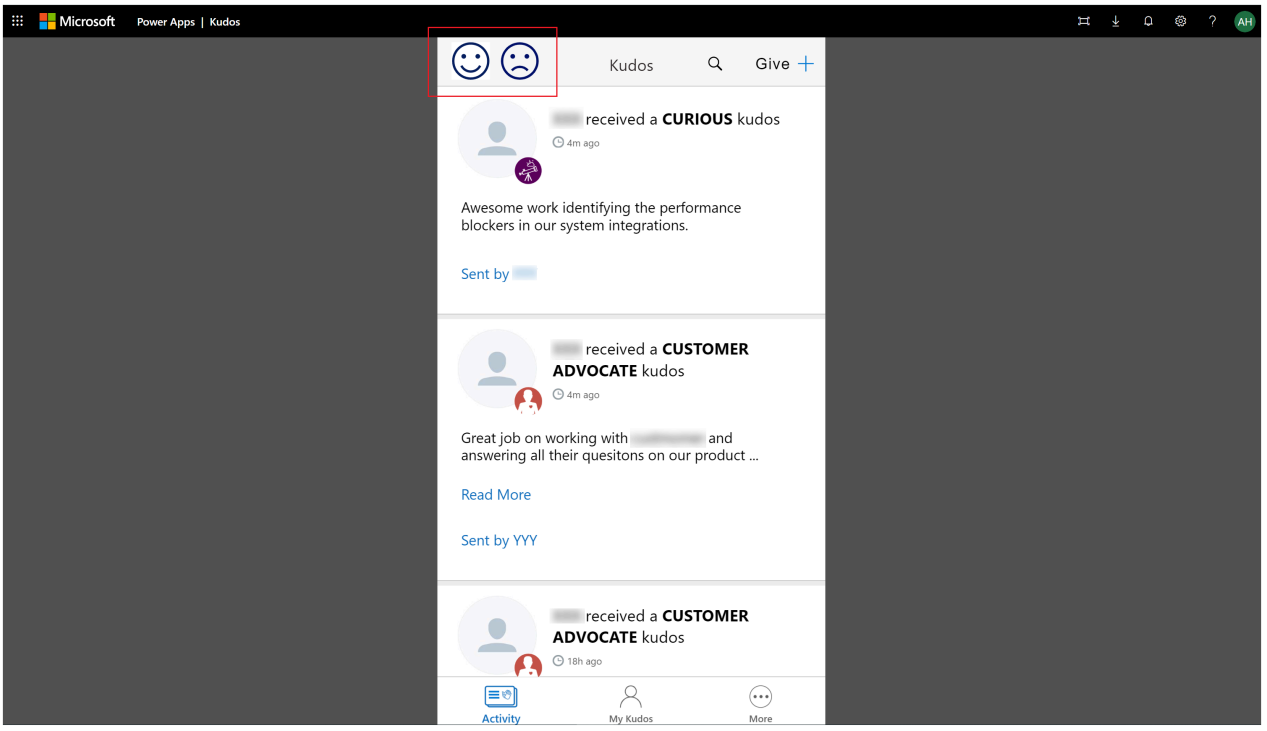
14. Add the component to one of the screens in your app:



15. Select **Save** and then select **Publish** to save and publish your app.

16. Play the published app and send smile or frown feedback from its screens.

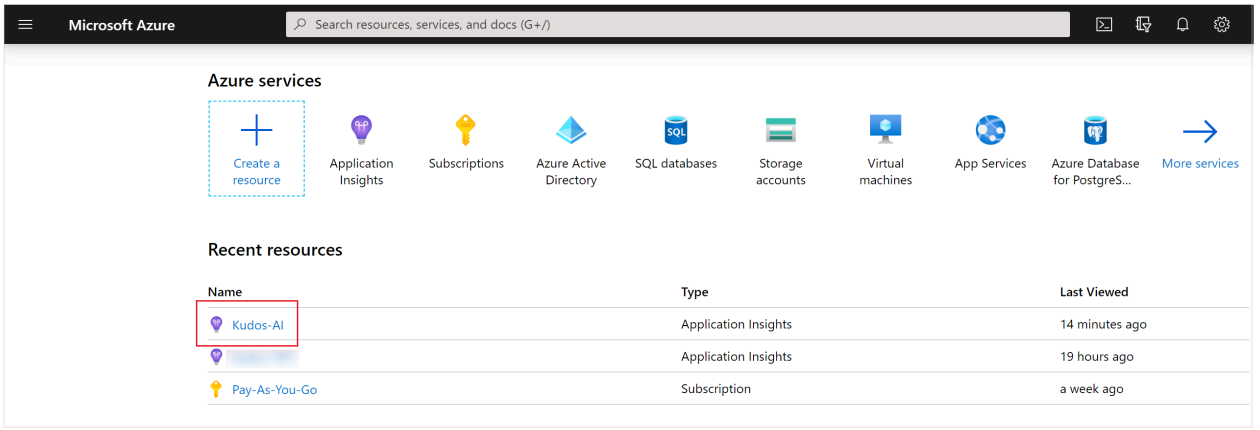
> **ⓘ Important**
>
> You must play the published app to send events to Application Insights. Events are not sent to Application Insights when you preview the app in Power Apps Studio.



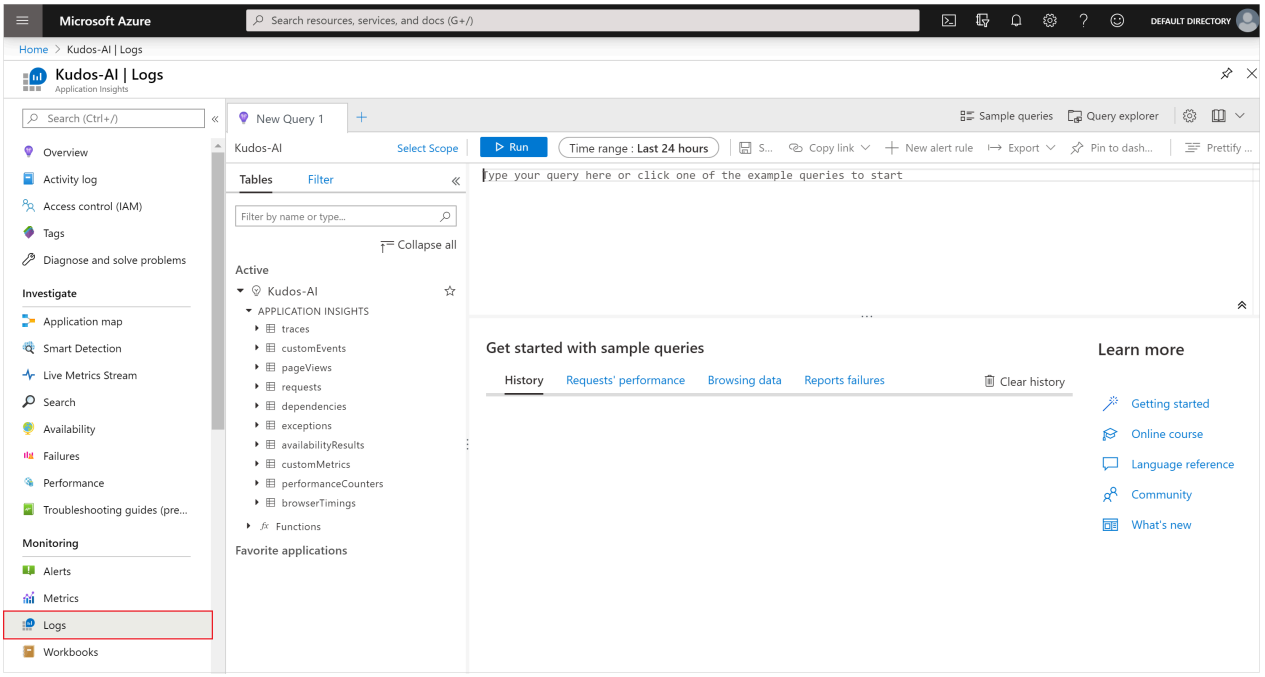
Analyze custom data in Application Insights

You can now begin to analyze the data you sent using the Trace function from your app in Application Insights.

1. Sign in to the Azure portal and open the application insights resource you created earlier.

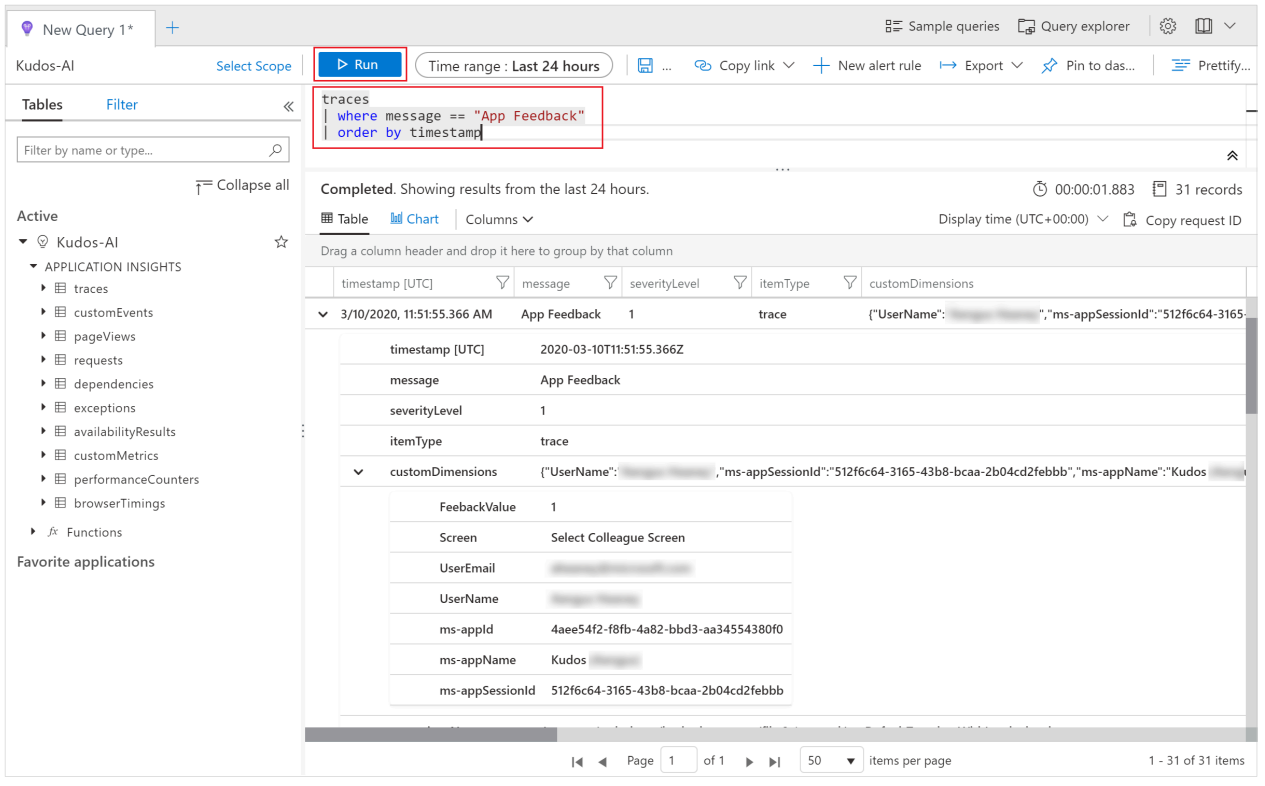


2. Select **Logs** under **Monitoring** in the left navigation pane.



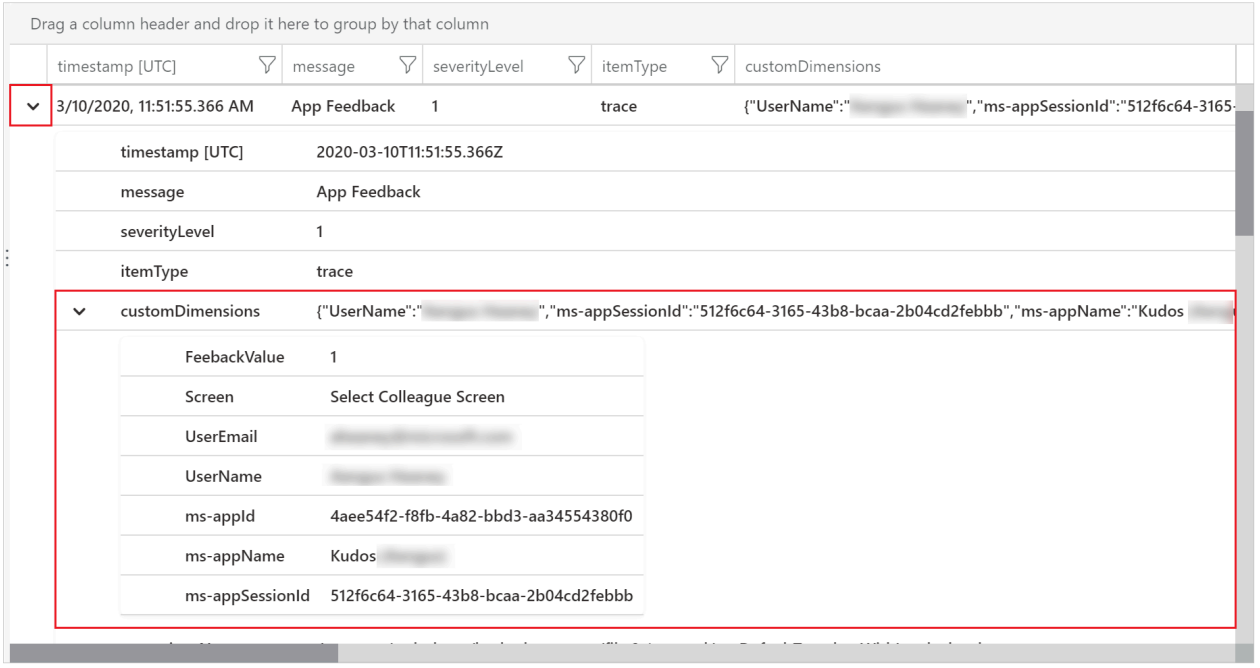
3. Enter the following query and select **Run** to view the feedback from your app:

```
Kusto

traces
| where message == "App Feedback"
| order by timestamp
```



4. Select a row in the results and expand the *customDimensions* field.

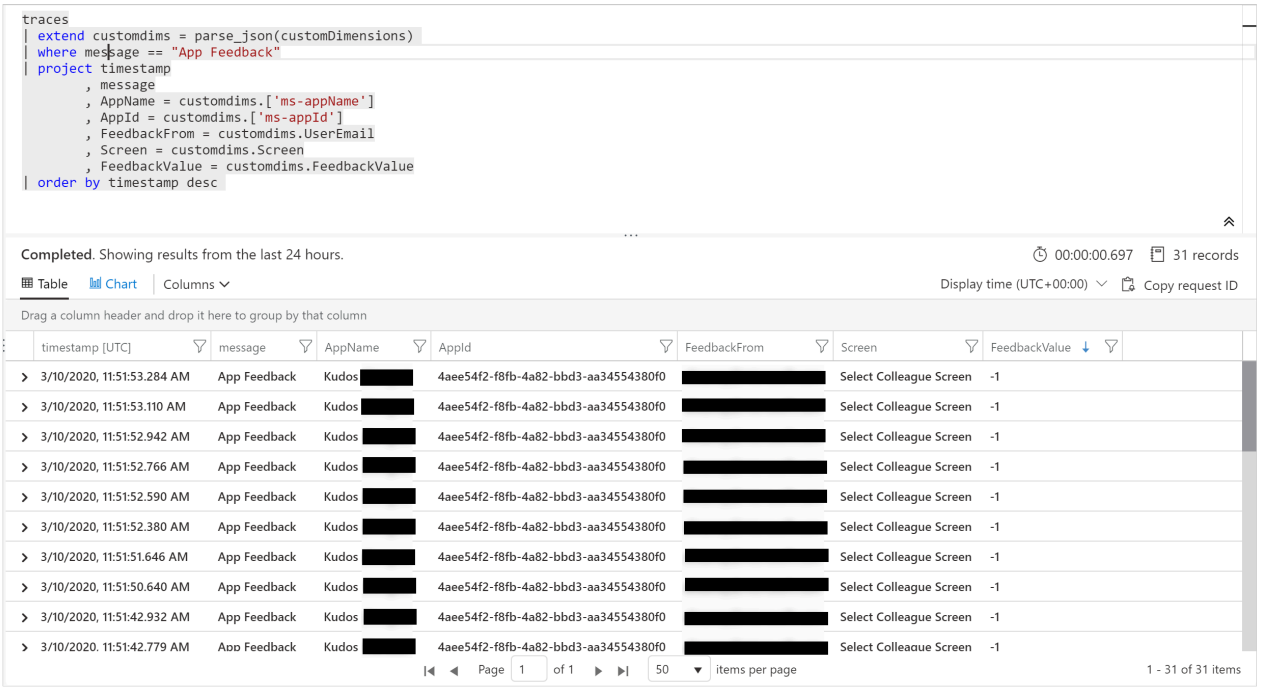
The values for **Screen**, **UserName**, **UserEmail**, and **FeedbackValue** for the **OnSelect** event of the smile or frown icon in your component are recorded. Values are also recorded for each event sent to Application Insights, such as the **appId**, **appName**, and **appSessionId**.



5. Use the following example query to extend the properties of the JSON custom dimensions and project the columns in the results view.

Kusto

```
traces
    | extend customdims = parse_json(customDimensions)
    | where message == "App Feedback"
    | project timestamp
        , message
        , AppName = customdims.['ms-appName']
        , AppId = customdims.['ms-appId']
        , FeedbackFrom = customdims.UserEmail
        , Screen = customdims.Screen
        , FeedbackValue = customdims.FeedbackValue
    | order by timestamp desc
```



> **Tip**
>
> *Log queries* are powerful. Use them to join multiple tables, aggregate large amounts of data, and run complex operations. Learn more about log queries.

Analyze app lifecycle data in Application Insights

The session summary event is logged once per session and contains information on app open success, app open optimal vs non-optimal sessions, and app open performance metrics.

Limits

These events are accurate for canvas apps running in a web browser. They aren't available for apps running in Power Apps mobile, and their values might not always be available or accurate for custom pages.

Here is an example query showing how to access the session summary event and all available fields:

Kusto

```
customEvents
| where name == "PowerAppsClient.PublishedApp.SessionLoadSummary"
//
| extend cd = parse_json(customDimensions)
//
| extend sessionSummary = parse_json(tostring(cd["ms-sessionSummary"]))
```

```
| extend successfulAppLaunch = tobool(sessionSummary["successfulAppLaunch"])
| extend unsuccessfulReason = tostring(sessionSummary["unsuccessfulReason"])
| extend appLoadResult = tostring(sessionSummary["appLoadResult"])
| extend appLoadNonOptimalReason =
tostring(sessionSummary["appLoadNonOptimalReason"])
//
| extend timeToAppInteractive = todouble(sessionSummary["timeToAppInteractive"])
| extend timeToAppFullLoad = todouble(sessionSummary["timeToAppFullLoad"])
//
| project
    timestamp,
    session_Id,
    successfulAppLaunch,
    unsuccessfulReason,
    appLoadResult,
    appLoadNonOptimalReason,
    timeToAppInteractive,
    timeToAppFullLoad
| limit 5
```

The following fields help measure app open success and latency of performance markers tied to end-user experiences.

Expand table

Field	Description
successfulAppLaunch	Boolean value indicating whether the session successfully launched the app
unsuccessfulReason	If the session failed to launch the app, this indicates the reason / error. This field will be an empty string if the session was successful.
appLoadResult	Indicates if the session was optimal or not. Possible values: optimal, other
appLoadNonOptimalReason	If the session was not optimal, this indicates the reason. Possible values: interaction-required, throttled, screen-navigated-away, other
timeToAppInteractive	Duration in milliseconds for the app session to reach an interactive state. In this state, users can start to interact with the first screen, but data may not be fully loaded.
timeToAppFullLoad	Duration in milliseconds for the app session to reach a fully loaded state, where all data requests for the first screen have finished loading.

Sample queries

App Open Success rates

This query will show the app open success rate by day. This can be used to evaluate spikes or trends in issues that users may be experiencing

Kusto

```
customEvents
| where name == "PowerAppsClient.PublishedApp.SessionLoadSummary"
| extend cd = parse_json(customDimensions)
| extend sessionSummary = parse_json(tostring(cd["ms-sessionSummary"]))
| extend successfulAppLaunch = tobool(sessionSummary["successfulAppLaunch"])
| summarize
sessions_total = dcount(session_Id),
sessions_success = dcountif(session_Id, successfulAppLaunch == true)
by bin(timestamp, 1d)
| extend successRate =
100.0 * (todouble(sessions_success) / todouble(sessions_total))
| project timestamp, successRate
| render timechart
```

Count of Unsuccessful Sessions by Reason

This query will show counts of unsuccessful sessions by reason / error. This can be used to debug app open failures or evaluate trends in issues that users may be experiencing.

Kusto

```
customEvents
| where name == "PowerAppsClient.PublishedApp.SessionLoadSummary"
| extend cd = parse_json(customDimensions)
| extend sessionSummary = parse_json(tostring(cd["ms-sessionSummary"]))
| extend successfulAppLaunch = tobool(sessionSummary["successfulAppLaunch"])
| extend unsuccessfulReason = tostring(sessionSummary["unsuccessfulReason"])
| where successfulAppLaunch == false
| summarize
count()
by unsuccessfulReason, bin(timestamp, 1d)
| render timechart
```

App Open Performance

This query will show app open performance metrics by day. This can be used to evaluate performance trends over time or after making changes. We recommend:

1. Using the 75th percentile of the timeToAppInteractive and timeToAppFullLoad fields to avoid noise caused by outliers.

2. Filtering to only optimal sessions to avoid noise in the data caused by expected cases like sessions with user interaction, sessions where the app was loaded in a background tab, etc.

Kusto

```
customEvents
| where name == "PowerAppsClient.PublishedApp.SessionLoadSummary"
| extend cd = parse_json(customDimensions)
| extend sessionSummary = parse_json(tostring(cd["ms-sessionSummary"]))
| extend appLoadResult = tostring(sessionSummary["appLoadResult"])
| extend timeToAppInteractive = todouble(sessionSummary["timeToAppInteractive"])
| extend timeToAppFullLoad = todouble(sessionSummary["timeToAppFullLoad"])
| where appLoadResult == "optimal"
| summarize
percentile(timeToAppInteractive, 75),
percentile(timeToAppFullLoad, 75)
by bin(timestamp, 1d)
| render timechart
```

App Load State on HTTP Calls

There is a new request header x-ms-app-load-state that indicates if an HTTP call contributed to app startup. Specifically, this can be used to determine which HTTP calls impacted the timeToAppFullLoad above.

The header can be one of two values:

Expand table

Value	Description
TTFL	indicates that the request contributed to timeToAppFullLoad
PostTTFL	indicates that the request did not contribute to timeToAppFullLoad

Here is an example query showing how to access the header value and projecting it in the appLoadState column:

Kusto

```
dependencies
| extend cd = parse_json(customDimensions)
| extend requestHeaders = parse_json(tostring(cd["requestHeaders"]))
| extend appLoadState = tostring(requestHeaders["x-ms-app-load-state"])
| project timestamp, session_Id, appLoadState, name, duration
| limit 5
```

Count of HTTP calls contributing to Full Load

This query will show the average count of HTTP calls that are contributing to timeToAppFullLoad by day. This can be used to evaluate the number of calls the app is making at startup that may be contributing to poor performance.

Kusto

```
dependencies
| extend cd = parse_json(customDimensions)
| extend requestHeaders = parse_json(tostring(cd["requestHeaders"]))
| extend appLoadState = tostring(requestHeaders["x-ms-app-load-state"])
| where appLoadState == "TTFL"
| summarize httpCalls = count() by session_Id, bin(timestamp, 1d)
| summarize avg(httpCalls) by timestamp
| render timechart
```

Duration of HTTP calls contributing to Full Load

This query will show the total duration of HTTP calls that are contributing to timeToAppFullLoad by day. This can be used to evaluate the overall impact of HTTP calls to app startup performance.

Kusto

```
dependencies
| extend cd = parse_json(customDimensions)
| extend requestHeaders = parse_json(tostring(cd["requestHeaders"]))
| extend appLoadState = tostring(requestHeaders["x-ms-app-load-state"])
| where appLoadState == "TTFL"
| summarize httpCallDuration = sum(duration) by session_Id, bin(timestamp, 1d)
| summarize percentile(httpCallDuration, 80) by timestamp
| render timechart
```

Duration of HTTP calls contributing to Full Load by URL

Similar to above, this query will show the count and duration of HTTP calls contributing to timeToAppFulLoad by URL. This can be used to identify specific slow HTTP calls that are impacting app startup performance.

Kusto

```
dependencies
| extend cd = parse_json(customDimensions)
| extend requestHeaders = parse_json(tostring(cd["requestHeaders"]))
| extend appLoadState = tostring(requestHeaders["x-ms-app-load-state"])
```

```
| where appLoadState == "TTFL"
| summarize
count(), percentile(duration, 80) by name
```

Monitor unhandled errors (experimental)

[This section contains pre-release documentation and is subject to change.]

> **ⓘ Important**
>
> - This is an experimental feature.
> - Experimental features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

You can't always anticipate or plan for every error that might occur while your app runs. Unhandled Power Fx formula errors are reported to users as banner messages. They can also be reported to Application Insights to show their frequency and severity without relying on your app's users to report issues. You can also set up real-time alerts when runtime errors occur to take a more proactive approach.

Enable error passing to Application Insights

Enable the setting that lets Power Apps pass unhandled runtime errors to Azure Application Insights.

> **⚠ Warning**
>
> Enabling the **Pass errors to Azure Application Insights** setting might incur additional costs for storing Application Insights logs.

1. Open the canvas app for editing.
2. Select **Settings** > **Updates** > **Experimental**, and turn on **Pass errors to Azure Application Insights**.
3. Save and then publish your app.

Error events in Application Insights

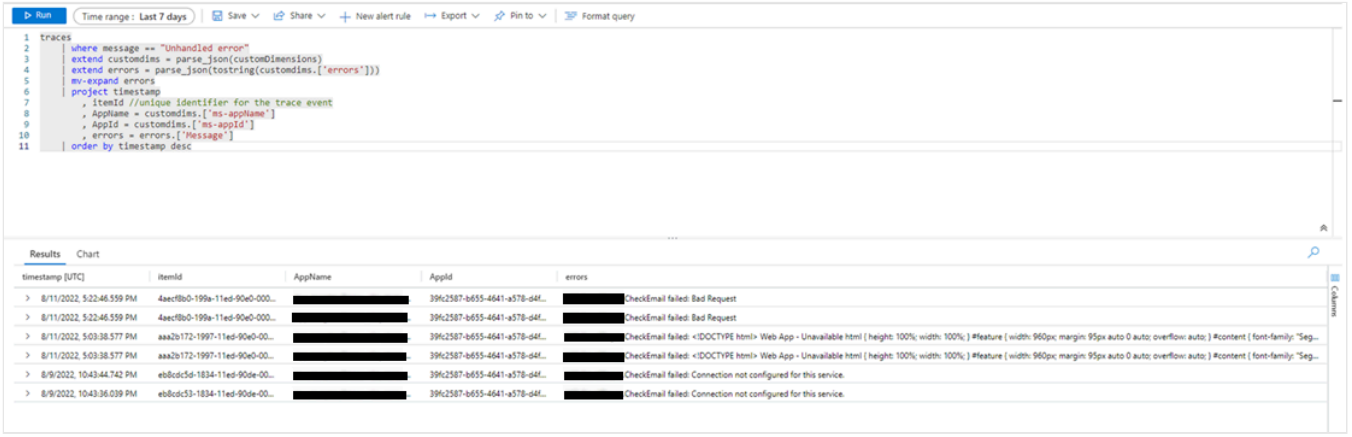
Unhandled Power Fx errors that users encounter at app runtime are reported to the **traces** table. Unhandled errors can be identified and distinguished from other error events by the event message "Unhandled error." The "severityLevel" dimension of these events is 3 (TraceSeverity.Error).

Detailed error messages are provided in the "errors" dimension of the *customDimension* property. In situations where multiple errors occurred during the same operation, the errors are consolidated in the "errors" dimension of a single trace event. The error messages match those reported in Monitor during a live debug session.

This example query identifies unhandled errors and expands all error messages in the trace event:

Kusto

```
traces
    | where message == "Unhandled error"
    | extend customdims = parse_json(customDimensions)
    | extend errors = parse_json(tostring(customdims.['errors']))
    | mv-expand errors
    | project timestamp
        , itemId //unique identifier for the trace event
        , AppName = customdims.['ms-appName']
        , AppId = customdims.['ms-appId']
        , errors = errors.['Message']
    | order by timestamp desc
```



Correlation tracing (experimental)

[This section contains pre-release documentation and is subject to change.]

> **Important**
>
> - This is an experimental feature.

- Experimental features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

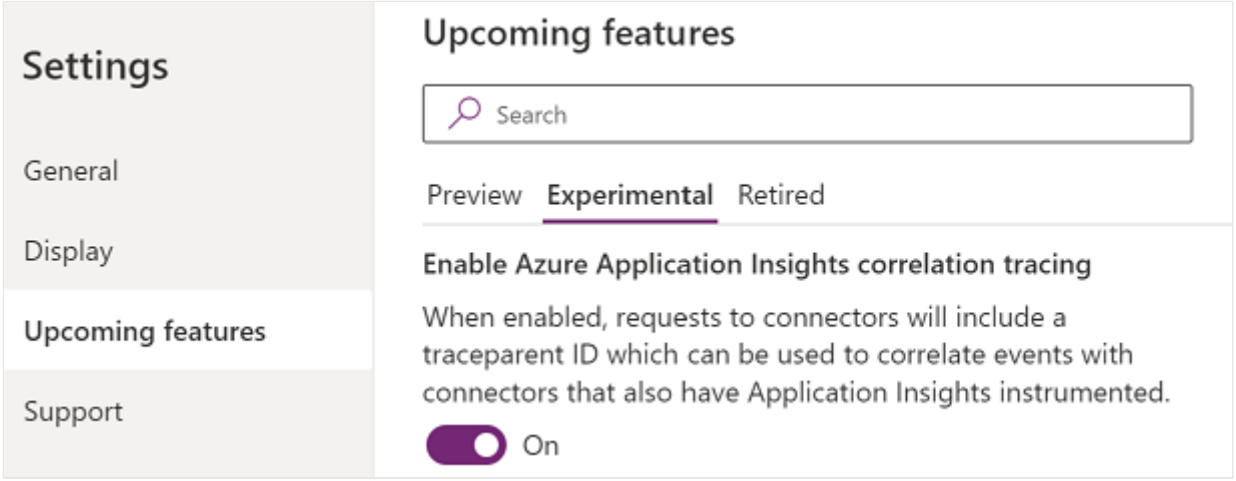
Connecting to external data and services is essential for most apps. Correlation tracing generates and propagates context information to link system-generated logs across a canvas app and its connections, subject to certain limitations. For example, your app might call a custom connector that then calls an Azure Function or another REST API. Correlation tracing allows you to correlate actions taken in the app with the underlying API calls across tiers. This is useful for troubleshooting.

Canvas app correlation tracing implements context tracing and follows the W3C specification.

Enable correlation tracing

> ⚠ **Warning**
>
> Enabling this setting might incur additional costs related to storing Application Insights logs.

1. To enable correlation tracing, open your cavas app for editing.
2. Select **Settings** > **Udates** > **Experimental** > and turn on **Enable Azure Application Insights correlation tracing**.
3. Save and publish the app.

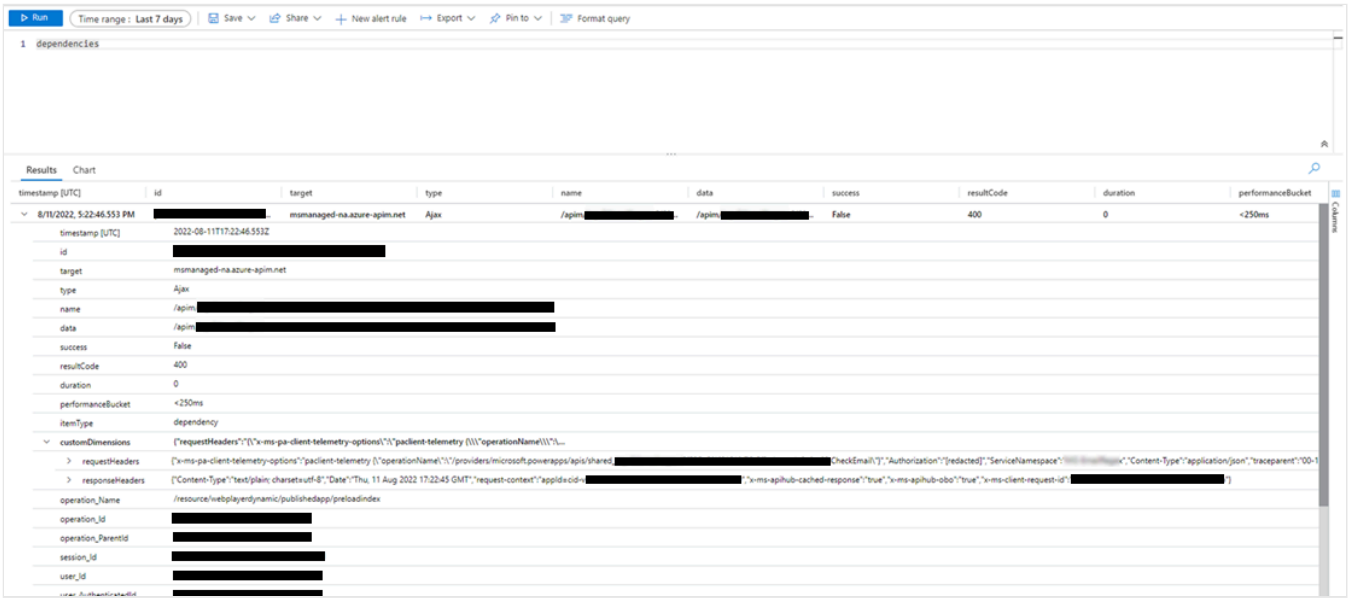


Limitations

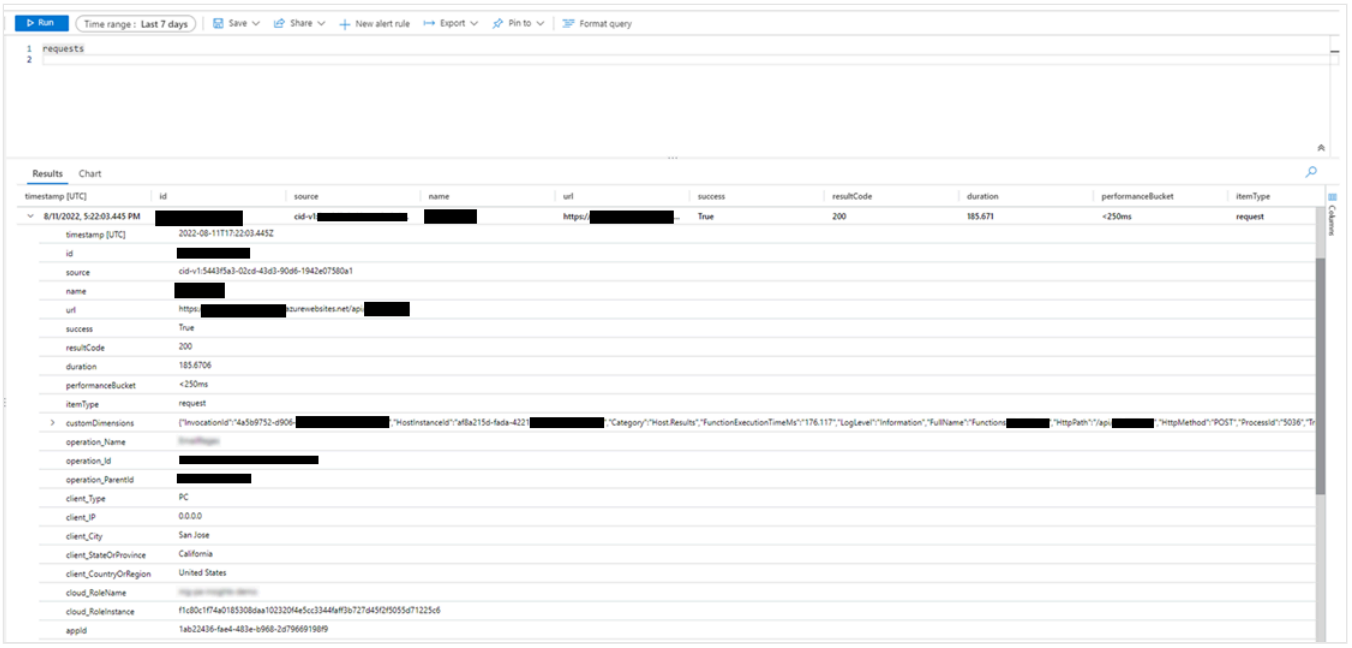
- Correlation tracing works only with custom connectors. Other connector types aren't supported.
- HTTP requests are captured in Application Insights only if the connected service is also connected to Application Insights.

Using correlation tracing

When enabled, correlation tracing adds a new system-generated log event in the **dependencies** table of the canvas app's Application Insights instance. This event is recorded at the time a response from a network call is received. Dependency events capture details of the network call, including the request and response headers, response status code, and duration of the call.



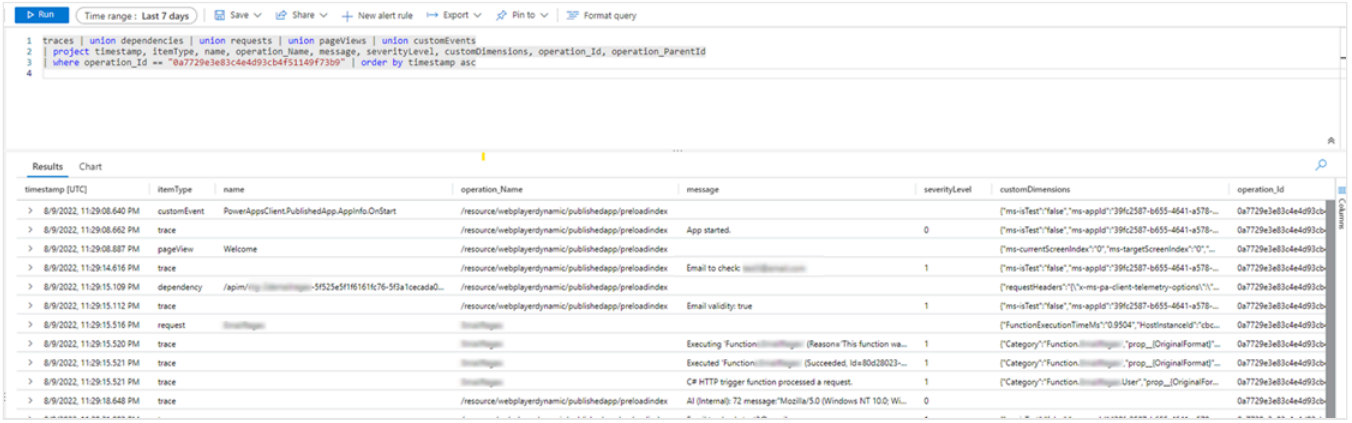
If the connected service is also connected to Application Insights, an additional system-generated log event capturing the request is generated in the **requests** table of the service's Application Insights instance. Some Azure services, such as Azure Functions, can be connected without any coding from the Azure portal. Both the canvas app or multiple apps and connected services can be connected to the same Application Insights instance.



Join network calls for supported connectors with other system-generated logs on the "operation_Id" dimension. The following query shows a network call alongside trace events emitted during an app session.

Kusto

```
traces | union dependencies | union requests | union pageViews | union
customEvents
| project timestamp
    , itemType
    , name
    , operation_Name
    , message
    , severityLevel
    , customDimensions
    , operation_Id
    , operation_ParentId
| where operation_Id == "0a7729e3e83c4e4d93cb4f51149f73b9" //placeholder
operation_Id, replace
| order by timestamp asc
```



Export data to Power BI

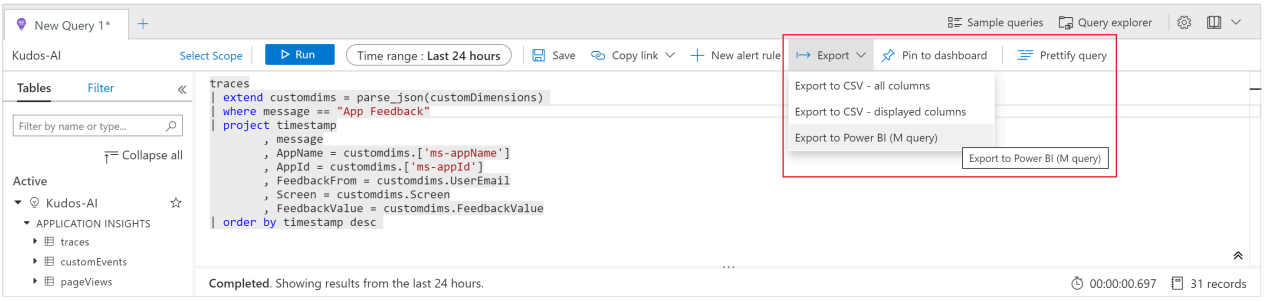
You can export your Application Insights data and query results to Power BI for analysis and data presentation.

1. Sign in to the Azure portal and open the Application Insights resource you set up earlier.

2. Select **Logs** under **Monitoring** in the left navigation pane.

3. In the log analytics query window, select the **Export** menu.

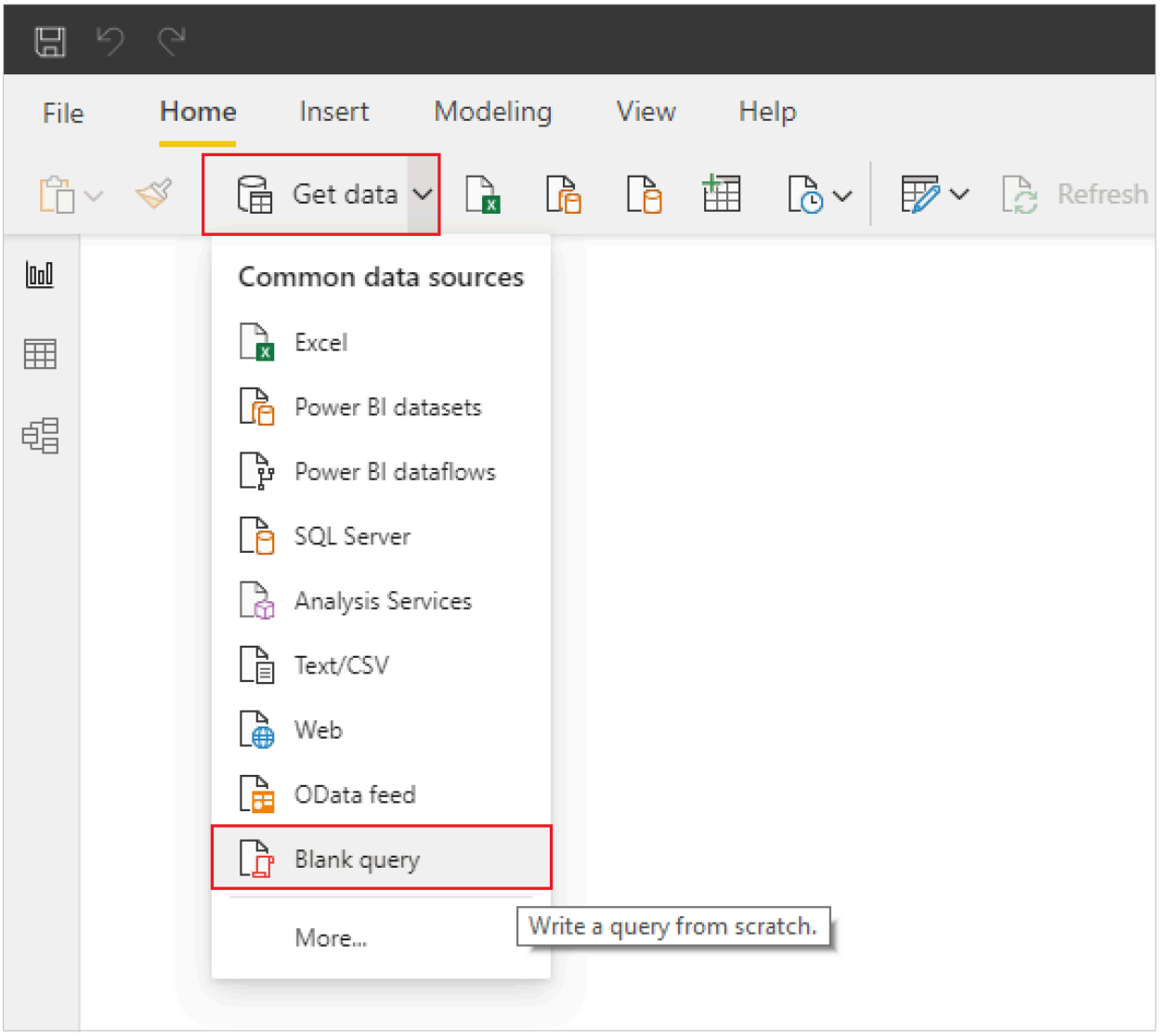
4. Select **Export to Power BI (M query)** to download a Power BI query file.



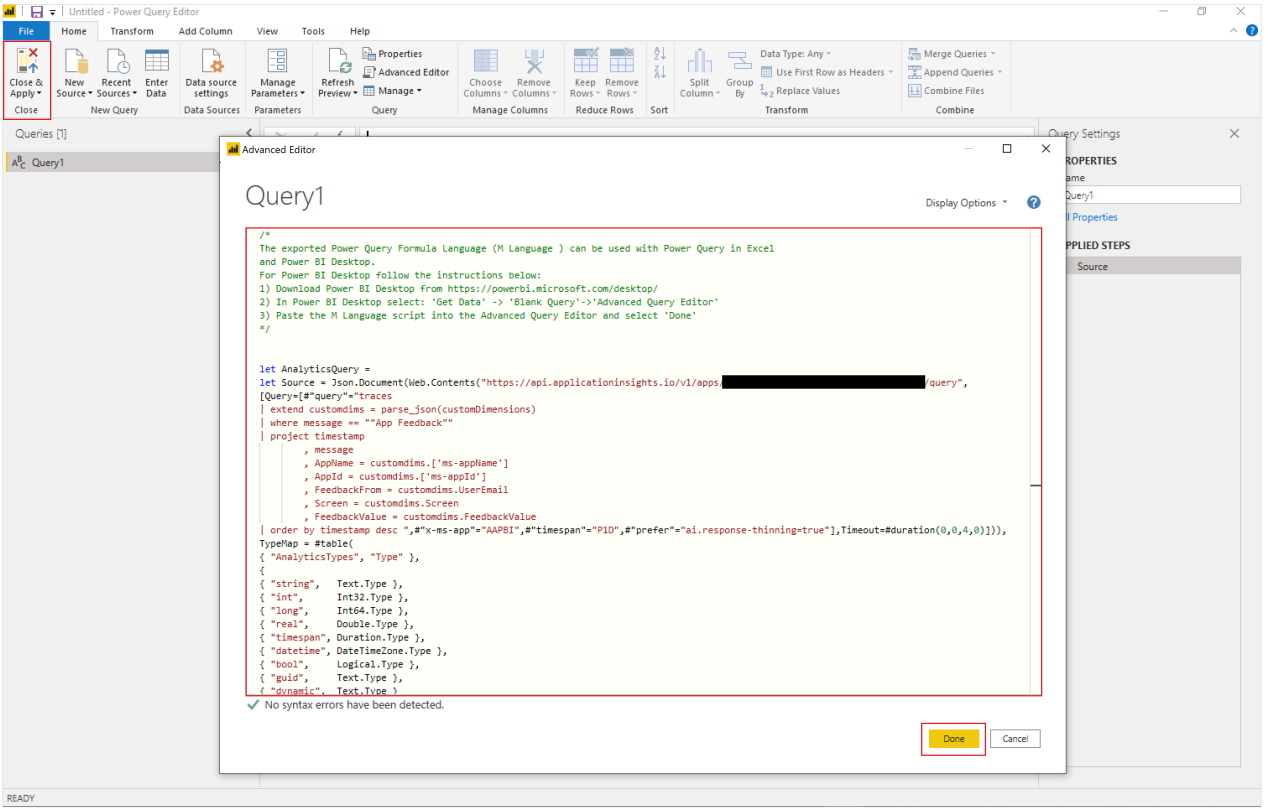
5. Open the downloaded file in a text editor, and copy the query to the clipboard.

6. Open Power BI.

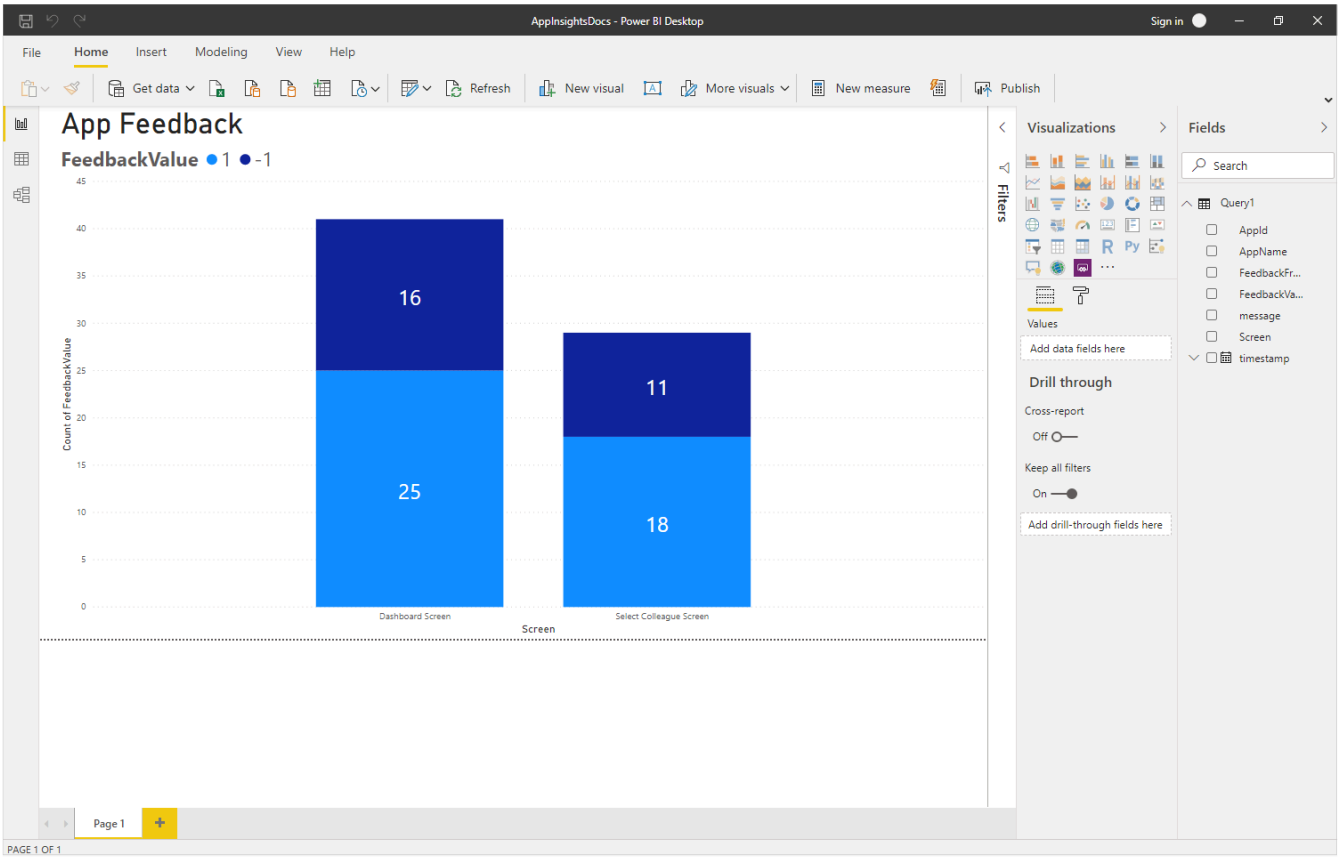
7. Select **Get Data** in the **Home** ribbon, and then select **Blank query**.



8. In the query window, select **Advanced Editor**, paste the query into the window, select **Done**, and then select **Close & Apply**.



Create charts and visualizations in Power BI to represent feedback from your app and make data-driven decisions.



Default Trace event context and dimensions

Default dimensions are added to the *customDimensions* property on each Trace event. These dimensions identify the application and application sessions where the events occur. If you log additional custom data using the Trace function, it also appears in the custom dimensions.

Expand table

Dimension Name	Represents
ms-appId	The Application ID of the app that sent the event.
ms-appname	The Application name of the app that sent the event.
ms-appSessionId	The application session ID. This value might not be populated in some scenarios. When available, this value overrides the standard Application Insights sessionID dimension.
ms-tenantID	The unique identifier of the tenant where the application is published.
ms-environmentId	The name of the environment where the application is published.
userId	A unique identifier for the user associated with the session.
ms-duration	An imputed value measuring the time it takes for a user to navigate from one screen to another. This value overrides the standard Application Insights PageView duration dimension.
sessionId	A session ID that can be used to correlate all events associated with a single application session. This value is always present and is recommended for understanding unique session count. This value is taken from the player's session ID and is shown when viewing the session details while playing the app. Session ID might sometimes get a default, random, unique Application Insights-generated value. This default value isn't reliable and doesn't correlate with any app-specific parameters.
Duration	An imputed value measuring the time it takes for a user to navigate from one screen to another. This value matches the duration reported by the ms-duration dimension.
ms-isTest	Indicates whether the session is associated with the Test Studio test runner.
ms-currentScreenName	The name of the page the user navigates from (available for page navigation events).
ms-targetScreenName	The name of the page a user is navigating to (present for page navigation events).

Unsupported scenarios

Application Insights doesn't support these scenarios.

- Offline player events aren't recorded.
- Events from mobile apps (iOS and Android) aren't recorded when the app is suspended.

Understand canvas app execution phases, data call flow, and performance monitoring

Article • 12/12/2023

When a user opens a canvas app, the app goes through several phases of execution before showing any user interface. While the app loads, it connects to different data sources—such as SharePoint, Microsoft Dataverse, SQL Server (on-premises), Azure SQL Database (online), Excel, and Oracle.

In this article, you'll learn about these different phases of execution and how an app connects to data sources and about tools you can use to monitor performance.

Execution phases in canvas apps

A canvas app goes through the following phases of execution before showing the interface to a user:

1. **Authenticate the user**: Prompts the first-time user to sign in with credentials for whatever connections the app needs. If that user opens the app again, that person might be prompted again, depending on the organization's security policies.

2. **Get metadata**: Retrieves metadata, such as the version of the Power Apps platform on which the app runs and the sources from which it must retrieve data.

3. **Initialize the app**: Performs any tasks specified in the OnStart property.

4. **Render the screens**: Renders the first screen with controls that the app populates with data. If the user opens other screens, the app renders them by using the same process.

Data call flow in canvas apps

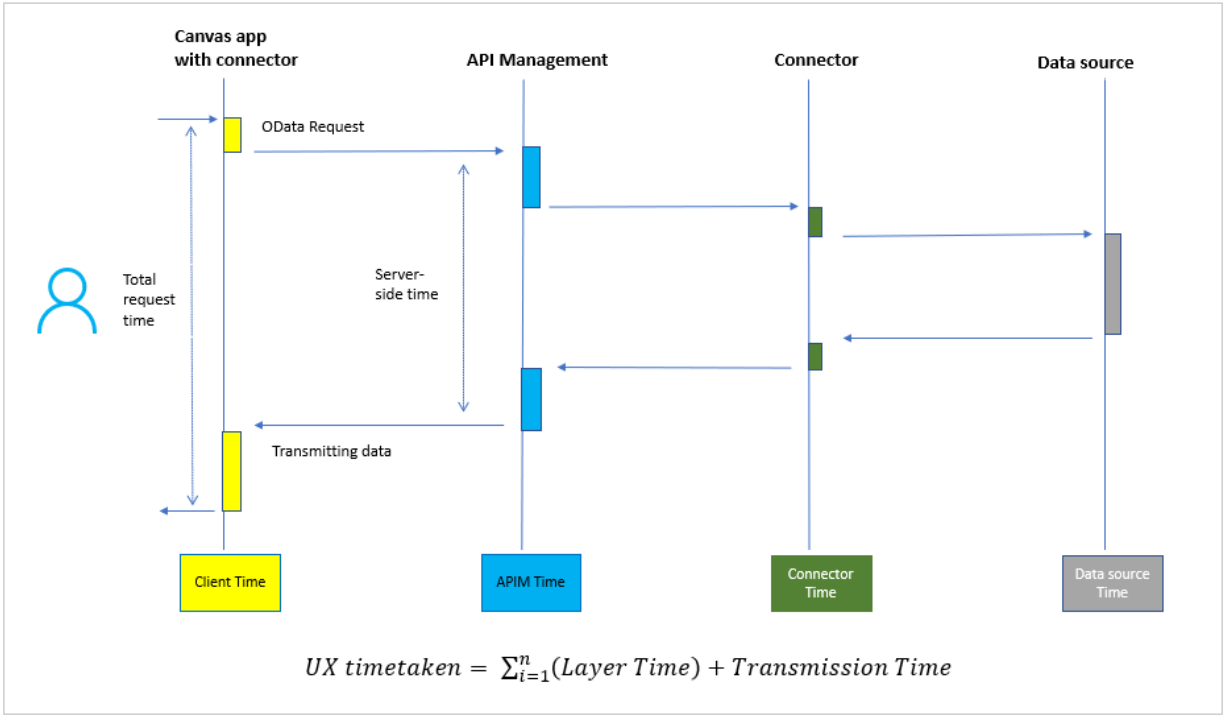
Data calls from canvas apps send data to tabular data sources by using connectors over the OData protocol.OData requests flow to back-end layers to reach out to the target data source and retrieve data for the client, or commit data to the data source. Action based connectors that enable APIs work in the same way.

Understanding how OData and API requests travel in canvas apps can help you to optimize your canvas app performance and your back-end data sources.

In this section, you'll learn about how the data call flows in canvas apps with different data source types.

Data call flow with online data sources

The following diagram shows how a typical data request in a canvas app (on the left side) travels server-side layers, reaches out to the target data source (on the right side), and then returns the data to the client.



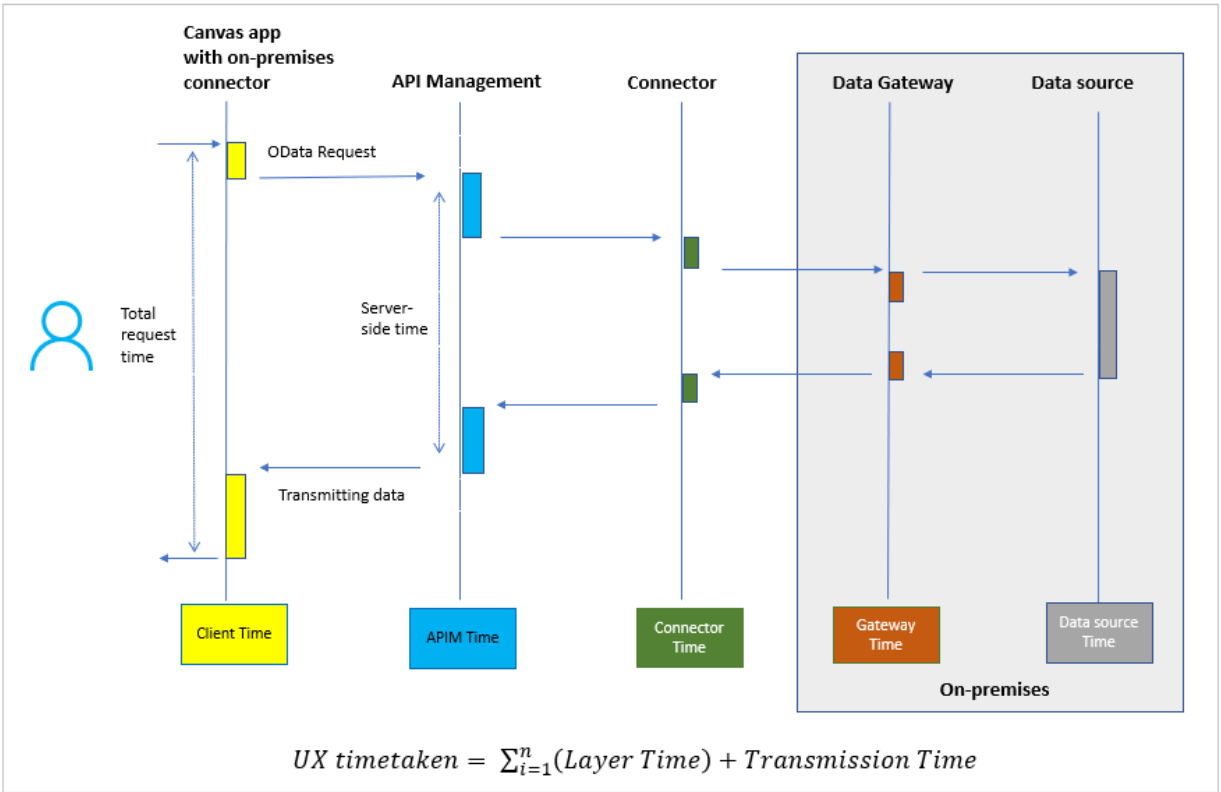
Each layer in the preceding diagram can perform quickly or encounter some overhead while processing the request. In many apps, two particular spots commonly present noticeable overhead:

- **Back-end data source** while processing the request.
- **Client** while sending the request—or while manipulating the received data on the heap memory and executing the associated JavaScript functions to process data to show in screens.

Data call flow with on-premises data gateway

If a canvas app connects to an on-premises data source like SQL Server, you need to have another layer called an *on-premises data gateway*. This gateway is mandatory for accessing on-premises data sources. It takes charge of converting OData protocol requests to SQL Data Manipulation Language (DML) statements.

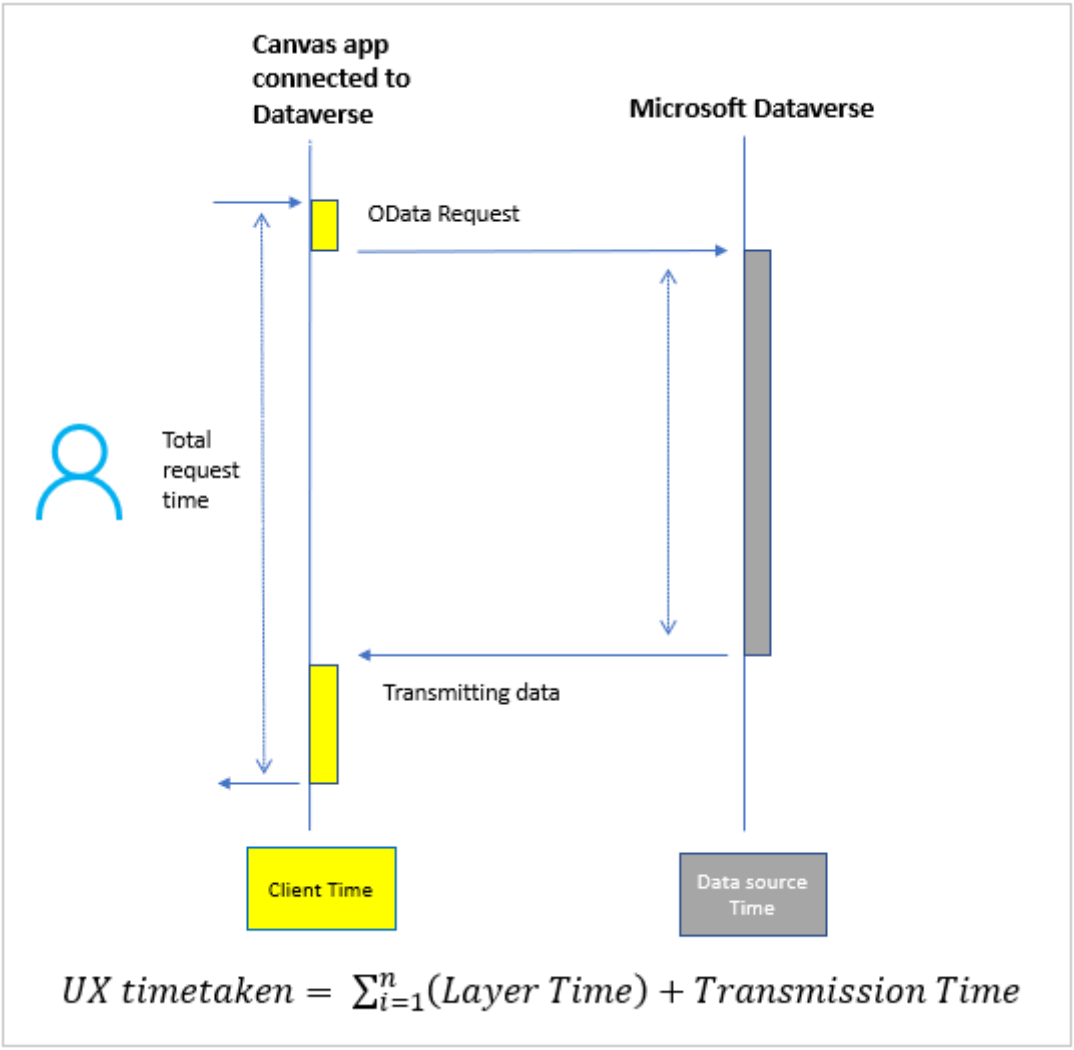
The following diagram shows where and how the on-premises data gateway is put in place to process data requests.



If the app uses a data source on-premises, the location and the specification of the data gateway also affect the performance of data calls.

Data call flow with Microsoft Dataverse

When you use Microsoft Dataverse as the data source, data requests go to the environment instance directly—without passing through Azure API Management. Because of this, the performance of data calls is faster compared to the rest of the data sources. The app is by default connected to Microsoft Dataverse when you create a new canvas app.



With the understanding of this high-level concept of how data calls travel, you can get into the details of reviewing the performance of your app. In summary, performance overhead can happen at any of the layers—from client, API Management, connector, on-premises data gateway, or back-end data sources.

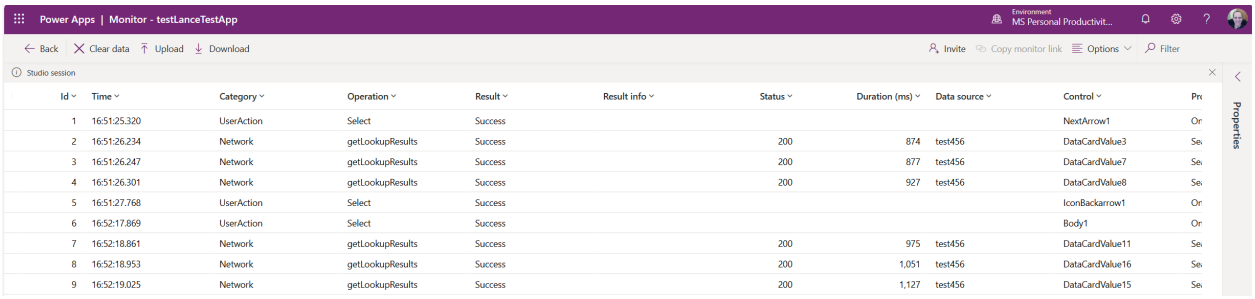
Measuring performance

Power Apps Monitoring tool

While you can use the browser's developer tools to see performance, Power Apps subsets the set of calls in the Monitoring tool to just those that are Power Apps.

The Power Apps monitoring tool can help you track what is actually sent to the data source and timestamps for when requests are sent and responses come from the server.

You can learn more about the monitoring tool in this article: Debugging canvas apps with Monitor .



Id	Time	Category	Operation	Result	Result info	Status	Duration (ms)	Data source	Control	Pr
1	16:51:25.320	UserAction	Select	Success					NextArrow1	On
2	16:51:26.234	Network	getLookupResults	Success		200	874	test456	DataCardValue3	Se
3	16:51:26.247	Network	getLookupResults	Success		200	877	test456	DataCardValue7	Se
4	16:51:26.301	Network	getLookupResults	Success		200	927	test456	DataCardValue8	Se
5	16:51:27.768	UserAction	Select	Success					IconBackarrow1	On
6	16:52:17.869	UserAction	Select	Success					Body1	On
7	16:52:18.861	Network	getLookupResults	Success		200	975	test456	DataCardValue11	Se
8	16:52:18.953	Network	getLookupResults	Success		200	1,051	test456	DataCardValue16	Se
9	16:52:19.025	Network	getLookupResults	Success		200	1,127	test456	DataCardValue15	Se

Measuring memory pressure on the client

To see memory consumption graphically, you can use the developer tools for your browser to profile memory. It helps you visualize heap size, documents, nodes, and listeners. Profile the app's performance by using a browser, as described in Microsoft Edge (Chromium) Developer Tools overview. Check the scenarios that exceed the memory threshold of the JS heap. More information: Fix memory problems



Next steps

Small data payloads

See also

Troubleshooting issues for Power Apps

Small data payloads - limit the amount of data you get

Article • 12/12/2023

One of the most important patterns for enterprise data apps is to limit the size of the data you fetch into Power Apps. If the data payloads are small, nearly everything else gets easier. The Gallery and Table controls do this for you automatically when you connect directly to the data source in the Items property. For instance, when connected directly to a remote data source, a Gallery control pages in data in small increments, for example, 100 records. This default leverages the fact that an end user rarely really needs more than a hundred records for a user task.

Use delegation

One key way to keep data payloads small is to rely on the data source to do work for you before it gets to Power Apps. **Delegation** is when Power Fx can translate a Power Fx expression into a query expression that a data source such as Dataverse, SQL Server, SharePoint, and Excel can handle on its own. And, then Power Fx delegates that query, or gives the responsibility of doing the query work, the data mashup, to the data source. The data source produces the correct data and returns it to Power Apps.

When the data source is bound directly to a gallery or table, then the data is paged or handed back the data to Power Apps in small performant increments of 100 records. Different data sources have different capabilities. Dataverse, for example, has many more capabilities to filter data on the server than Excel. A good example is CountRows and CountIf. Dataverse supports CountRows in a limited way. Dataverse calculates the size of the table periodically and keep that value around. When CountRows is called, you're given that value. That way it doesn't have to perform a full table scan to get the exact number for every CountRows call. But Dataverse also supports an exact count with CountIf up to 50,000 rows. It supplies these two different capabilities as a way to help preserve good performance of Dataverse server. In contrast, SharePoint doesn't support this function. So, a Power Fx expression with CountRows or CountIf for SharePoint isn't delegated. Instead, Power Apps downloads a limited number of rows, 500 – 2000. Power Fx works on the 500/2000 records locally and returns a result. If your data is always less than 500/2000 records this approach can work. But if it's greater than 500/2000 records you might get incorrect results.

Avoid too many columns

By default, Power Apps computes the actual columns you need for a given query using a feature call **Explicit Column Selection**. This feature is on by default for all new apps. To turn on the explicit column selection feature on the canvas app, go to **Settings** > **Upcoming features** > **Preview**> turn on **Explicit column selection** toggle.

Suggestions

To achieve the goal of a small data payload, consider the following questions and suggestions:

1. Aim for the default query for a gallery or table to only return approximately 100 – 200 records. If you use a delegable query, this happens automatically. If you're querying an API, or other source that doesn't support delegation, use parameters to refine the results.

    - Consider using a data source based view that automatically filters the data. Most enterprise-grade apps make heavy use of views on the data source.
    - Consider using default lookup or filter values to scope the data.
    - Consider requiring search arguments in the UI before you show data.
    - Consider these questions about your app:
        - What is necessary on a given screen by default?
        - What data does the end user really need to take a given business action?
        - Users generally don’t need thousands of records on an initial screen to accomplish a task.

2. Start building your query for a gallery or table by using the delegation tables for your data source. Only choose the functions supported for your query. If your query isn't delegable, what can you do to make it delegable?

Optimized query data patterns

Article • 12/13/2023

The simplest and fastest data query pattern is:

1. A single table or view
2. Prefiltered on the server to what you need
3. Columns are indexed correctly for the expected queries

When you design your app, you need to think about how to query the data quickly. The best way to query data is to use a single table or view that has all the information you need, and filter it on the server before you display it in your app. You also need to make sure that the columns you use to filter or sort the data are indexed properly. This makes your app faster and smoother.

For example, suppose you have a gallery that shows a list of customers and their salespersons. If you store the customer and salesperson information in separate tables, you need to use lookups to get the salesperson name for each customer. This slows down your app because it needs to run many queries to the other table. A better way is to create a view that combines the customer and salesperson information in one table, and use that view as the data source for your gallery. Then your app only needs to run one query to get all the data it needs.

There's a trade-off between query speed and data normalization. Data normalization means that you store the data only once, and avoid duplication. This helps to keep the data consistent and accurate. However, sometimes you need to duplicate some data to make the queries faster and easier. You need to balance these two goals in your app design and table structure. Otherwise your app will be slow and laggy because it needs to do numerous work to filter and join the data from different tables.

Use server-side views

Views are probably the most common tool to help balance these goals. They present a single table structure for queries, prefilter data for what you need in the query, and enable lookups and joins to other tables. Because the filters, lookups and joins for the view are computed on the server, both the payload and client-side compute are minimized.

Avoid too many lookups on a gallery

A gallery can display many records from a data source. But sometimes, you need to show additional information from another data source that is related to the original one. For example, you have a gallery that shows a list of customers, and you want to show the name of the salesperson who is assigned to each customer. The salesperson's name is stored in a different data source than the customer's information. To show the salesperson's name, you need to use a lookup function that finds the matching record in the other data source. This expands the original table with the lookup values.

However, expanding the table can be very slow if you have many records and many lookups. For each record in the gallery, the app needs to run a separate query to the other data source and get the lookup value. This means that the app may need to run many queries for each record, which can take a long time and affect the app performance. This anti-pattern is sometimes known as "N squared, (n^2)" or an "N+1" problem.

Use StartsWith or Filter

Power Fx provides several ways to search data. In general, use an expression that leverages an index like **StartsWith** or **Filter** instead of one that reads the entire table like **In**. The In operator is fine for in-memory collections or if the external data source table is very small.

Consider duplicating data

Sometimes data is slow to access in a query because it is stored in a different location or format. To make the query faster, you can copy the slow data and store it locally in a table that is fast and easy to query. However, this means that the local data may not be the most updated version of the original data. Then run another process to update the local data periodically. This process can be a Power Automate flow, a plugin, a stored procedure, or any other method that can move data from one place to another.

The frequency requirement of updating the local data depends on your business needs. How fresh does the data need to be for your app? For example, suppose you work for Contoso, a company that sells bicycles. The list of available bicycles is stored in a Products database that you can access through an API in a custom connector. But say the API call is slow, and so you decide to copy the product data and store it locally in a table. Then you create a view that combines your table with other relevant data for your app. You also create a Power Automate flow that runs every day and updates your table with the latest product data from the API. Then your app can query the local data faster, and the data is only one day old at most.

Duplicating data is a common type of technique in enterprise-grade applications to ensure good performance. You can use Dataverse plugins, stored procedures, or data movement to duplicate data into a single table that is optimized for querying. The key question is: how up to date must this data be? If you can afford some delay, you can use this technique to speed up your app.

Suggestions

To achieve this goal, consider the following questions and suggestions:

1. How important is it for a customer to see the data value in a gallery or data grid? Would it be acceptable to first select a record and then show the data in a form?
2. Can a view do the prework necessary to see data in the right format?
3. Are you using an "IN" operator where a "StartsWith" will work?
4. How current does your data need to be? Is there a data duplication strategy you can use to get your query to work over a single table by default?

Optimize app or page load for peak performance

Article • 01/05/2024

One of the key factors that shape a user's perception of an app is how quickly it opens and becomes functional. Therefore, prioritizing this goal is crucial in building a performant app. To achieve optimal app performance, there's three main areas that require attention:

1. Load data fast
2. Efficient calculations
3. Minimizing required resources

Each of these areas has several common anti-patterns.

Load data fast

Follow these guidelines to achieve fast data loading apps.

Avoid directly populating a collection with large amounts of data

Sometimes authors use ClearCollect() to copy data from a server to a collection in their app. This practice is a work-around for delegation limitations in the source or because they plan to use collections in the app for other purposes. Using ClearCollect() can potentially increase the speed of the app when the collection is later utilized. However, it's important to exercise caution while implementing it. Using ClearCollect in this manner can lead to slower app load times or when navigating between pages. ClearCollect() must finish before you can see the data in a gallery or a table. This step can take a long time if there's lots of data or if you use this approach for too many data sources. Collections are best used for situations where the data is small and you need to do many calculations on the collection. For example, a good use of collections is an Online order basket. The customer may update and remove items several times before choosing to commit the order. Additionally, you may augment the collection with more data items like potential discounts, highlights, etc. Data that is 'read-only' should be accessed natively - without bringing it into a collection.

Consider avoiding calling Power Automate to populate a collection

This issue is a slight variation of the previous section. Sometimes authors also use Power Automate to populate their collection in Power Apps. There's approximately a 0.6-second performance cost to instantiate Power Automate. Power Automate must be independently launched each time it's called. It must allocate memory, get situated with the right components, and be ready to run. As with the above advice, one or two calls to Power Automate might not be an issue depending on your app. But, almost universally, the worst performing apps overuse this approach. The performance cost can add up quickly and ruin the performance of your app.

Avoid using SaveData() and LoadData() as a full offline scenario

Some authors use ClearCollect() and then SaveData() to store data for general purpose offline use. You can use SaveData() to save the collection to your device, and LoadData() to load it when you're offline. However, this approach isn't recommended for instances where there's a large amount of data or if the data is complex. It makes your app slower because it has to wait for ClearCollect() to finish before it can show the data. You should only use SaveData() and LoadData() for small and simple data scenarios such as preferences and short lists. A better way to work with large amounts of offline data is to use the Power Apps offline feature that works with Dataverse. This feature can handle larger and more complex data more efficiently.

Use Explicit column selection

Explicit column selection is on by default. However, some authors turn this feature off. The issue is that with Explicit column selection (ECS) turned on, columns are sometimes not retrieved from the data source if the data is first retrieved into a collection. As some tables may have many columns, ECS automatically computes which columns are necessary to retrieve based on their usage in controls (for example, galleries and forms.) As some tables can many columns, reducing the size of the download can speed up performance. Some tables can have a hundred columns or more. If your app only needs to use 10 columns, and you bring down all the columns from a 100 column table, you are bringing down ten times as much data as you actually need.

Most issues arise when bringing data into collections. If a column is explicitly referenced in a control then ECS works well. And, ECS generally does work for collections. However, the column lineage is occasionally lost when the data moves through collections and

variables. And, and so Power Apps may lose track of the column it should retrieve. To resolve this issue, you can force Power Apps to "remember" the column by using the `ShowColumns` function. For example:

```
Power Fx

    ClearCollect(
        MyColTable,
        ShowColumns(Filter(BankAccounts, AcountNo = 32),
        "Col1",
        "Col2",
        "Col3")
    );
```

Where `Col1`, `Col2`, and `Col3` are columns you expect to be retrieved from the data source (for example, the `Account` data source).

Alternatively, you can add a hidden control to your form that references the column. This forces Power Apps to "remember" the column.

Fast calculations

Fast (or efficient) calculations are a performance goal in its own right. For more information, see Fast (efficient) calculations. However there are some common anti-patterns that can affect app load as well and so we discuss them here. Below is a list of optimizations you should consider that might affect app launch or page navigation.

Use App.Formulas

Historically many authors have put a large number of calculations in OnStart. When you place an expression in OnStart, it forces Power Apps to run that expression precisely when the app starts and before everything else. However, with the introduction of App.Formulas you can let Power Apps decide when to run an expression. Power Apps can run the formula 'Just-in-time' before it's needed. For more information, see App formulas. Use App.Formulas to split up your formula into pieces that Power Apps can arrange more efficiently for execution.

Use Concurrent

Use the Concurrent function to allow formulas to be executed at the same time. It's common to use this function to populate collections as it allows for parallel execution.

While this can provide some modest speed ups, adding many data sources can cause timing and throttling issues.

Use Enhanced performance for hidden controls

When enabled by default in all new apps created since December 2022, Power Apps doesn't render any control that isn't initially visible.

Minimize required resources

Minimize the resources required to launch your app or screen. This effort includes carefully scoping or partitioning the resources your app or screen needs. Below are several approaches to help you.

Use a low dependency start screen and eliminate unused screens.

Use a low-dependency first screen such as a welcome on your application. Create a screen that doesn't load a gallery or table or reference data. This manages the user's perception of speed and allow Power Fx to properly delay some calculations to later. If you have any unused screens, remove them.

Keep cross screen dependencies between screens low

Cross page references force the load of additional pages via references, for example, referencing controls on pages and putting into collections. Some references might be unavoidable. Centralize common references to a single page if possible so that only page is loaded.

Consider embedded media

Authors sometimes embed media in their apps to ensure fast loading. If you have embedded media consider whether or not, you're using it. If not, delete it. If you have embedded a .png file, consider substituting a .svg file which is smaller. Note that if you use a .svg, the font for the .svg must be on the client machine. Consider the embedded media resolution. Is it too high for the device it will used on?

Don't forget App.StartScreen

If you use App.StartScreen, ensure the first screen is a blank screen. Due to current packaging of the app, the first logical screen is always bundled with the app init logic and will get initialized, regardless of whether we ever navigate to it.

Consider splitting up the app

If your app is large, consider partitioning it into smaller apps. If the functionality is separate enough in different parts of your app, this approach might work. In this scenario, you create an actual separate app that you launch with parameters that include context from the first or parent app.

Suggestions

To achieve the goal of a fast app and page start, consider the following questions and suggestions:

1. Are you loading lots of data in a first screen? Can you use a different first screen?
2. Are you running many commands or Power Fx expressions at the beginning of the app load? Can you defer these commands and expressions to a later point in the application? Only get the data you actually need to start the app? 1 Can you convert expressions in App.OnStart to named formulas with App.Formulas? This allows Power Fx to decide when to actually execute the formula rather than forcing it to happen on load or navigate events.
3. Can you use a separate Power Automate flow to create a temporary table in a local data store such as Dataverse that combines data from different sources? And then access that data in your Power App?
4. For business process kick offs, can you use Server triggered actions rather than calling a Power Automate Flow?
5. Can you create a view on the server that joins the data for you?
6. If you want to use offline data in your app, can you use the Power Apps offline feature that works with Dataverse? If your data isn't in Dataverse, can you move it there?

Feedback

Was this page helpful? Yes No

Provide product feedback

Fast calculations

Article • 12/13/2023

Power Fx expressions are powerful and do many background calculations for you automatically. While Power Fx automates many things for you, it's possible to fine-tune your calculations so there as fast as possible.

Data retrieval

Use Explicit column selection

The Explicit Column Selection (ECS) feature is enabled by default for all new apps. If it isn't enabled for your app, you should enable it. ECS automatically reduces the number of columns retrieved to only the ones that are used in the application. If ECS isn't enabled, you might be retrieving more data than you need, which can affect performance. Occasionally, when an app pulls data in through collections, the original lineage or source of a column can be lost. We don't know if it's being used and we drop it using ECS. You can usually force ECS to work for a missing column by using the PowerFx expression **ShowColumns** after a collection reference or by using it in a control.

Use images on galleries, tables, and forms carefully

Use Dataverse thumbnail versions of images for galleries and tables. Dataverse thumbnails are small about 1 kilobyte and are stored inline as part of the record, and are useful and fast for display on controls that show collections. All other image references including all SharePoint images require a separate call and shouldn't be placed on a gallery or table. Place images that require a separate call on detail forms. Consider never showing a full image by default. Full detailed images can be useful and important to users. However, you can make these images available to users through an explicit user action, for example, a button or navigation to a separate page. SharePoint has a range of intermediate sized images you can use for use on a form short of the full image. Dataverse only has two sizes: Thumbnail and Full.

Calculations

Split up formulas with App.formula

The use of App.formulas and named formulas can help with the speed of app load and page navigation because it allows Power Fx to decide when to evaluate a formula. That means it doesn't have to necessarily evaluate it in OnStart. In addition, named formulas can generally help speed as well. In particular if you have a long script, breaking it up into named formulas allow for more efficient calculations as Power Fx can schedule the work and it enables reuse. For more information, see App formulas.

Use Concurrent

Use the Concurrent function to allow formulas to be executed at the same time. Carefully choose where concurrent is used. It can provide some modest speed ups but if you're running items that depend on each other it can cause timing and throttling issues.

Defer significant updates to a nonblocking UI step

Large updates to a data source may take a while to complete. But users expect the UI to return control to them quickly. Tasks may be either sequential where the update must finish before the user can take another actions or asynchronous (the update can finish separate of user actions.)

An example of a time-consuming synchronous task is confirming a seat at a concert. For most time-consuming synchronous tasks like this, it’s common to put up a progress bar. This UI actually blocks the user but ensures that the task is complete before other UI elements are updated. This approach may not work for your application. Businesses normally handle longer sequential steps as an explicit business step. You're approved to go the next step via a business process signal. An example is an approval. An approval may come quickly or may be delayed. In the UI you can signal that the process is complete several ways. You can enable a button, show a message, send e-mail, or enable a part of the UI such as a menu item.

An example of an asynchronous task is the completion of an order. A customer updates an order basket, works through their order and then places the order. But several updates must happen before the customer is given a confirmed ship date. In this example, the Ship Date isn't something you can easily provide right away. , you can defer this part of the update to an e-mail that is sent to the customer later.

For synchronous tasks, you'll often use the code in Power Apps for UI blocking tasks. For instance, the code waits until it gets a return value and then releases the progress bar. It is best to minimize these types of situations. But for tasks that use an explicit business

step or asynchronous tasks it's common to use an external service to complete the task such as a Dataverse action, stored procedure, or a Power Automate flow.

Place 'ForAll' appropriately

If you have an expression with ForAll and collect that looks like this:

```
Power Fx

ForAll(x, Collect(y, { … }))
```

Then, invert this to

```
Power Fx

Collect(y, ForAll(x, { … }))
```

In the first pattern, any dependent rule on collection y is notified of changes and evaluated for each iteration of x. In the second pattern, these rules are only evaluated once.

Consider avoiding references to Gallery.AllItems

When working with Power Apps, it’s best to avoid referencing Gallery.AllItems unless you need out user values. This is because every time AllItems is read, a new output table is generated. Instead, use Gallery.AllItemsCount if you just want to know how many items are loaded.

Carefully use Gallery.TemplateSize

To ensure that flexible-height galleries are rendered correctly, it’s important to set a reasonable default size. Initially, we try to render as many rows as possible based on this value. If you set the default size to 0, we try to render everything, except for a few edge cases. If you’re using a formula, make sure to set a minimum value for when the formula might evaluate to 0. For example, you could use Max (20, varFoo + rectBar.Height). This way, if varFoo and rectBar.Height aren’t yet available, we at least use a reasonable value of 20.

Note DelayOutput especially for search

There's one second delay before the changes is detected, which allows you to complete the typing instead of detecting the changes for every typing.

Feedback

Was this page helpful? Yes No

Provide product feedback

Other performance considerations

Article • 12/11/2024

In addition to the four key performance principles, there are several other possible reasons for poor performance typically due to external factors.

Consider the differences in Client browsers, devices, and locations

Canvas apps can be used on different devices, browsers, and locations with varying network conditions. As the Power Apps client is executed, be sure to use modern, updated, and supported browsers. The performance of an app might vary when loading large sets of data on different platforms like iOS or Android. This variation happens because of different network request limitations on each platform. For example, the number of concurrent network requests allowed differs by platform. These differences can have a major impact act on the data load time for large datasets.

Consider the differences in Geographic location of the on-premises data gateway and environment

Users can access canvas apps globally. However, we recommend that you locate the data source near most of your users. For example, when your app accesses your on-premises data gateway, it's best to place the gateway near the users who is accessing the app most frequently.

General Server-side issues

Poor performance might be caused by problems at the server source of the data. This can happen for various reasons. You can use the monitoring tool to assess the specific issue by measuring the data call timings.

Possible bottlenecks issues in the data source

There are many possible causes of bottlenecks in the data source. Usually, a few tables in the data source are at the center of activity for many queries. Queries might be slow if:

- The data source is missing or has incorrect indexes.
- The query is joining extra-ordinary large amounts of data on the server.
- The query requires a table SCAN, for example, **In** operator instead of using an index like **StartsWith**.
- The back-end machine hosting the data source is low on resources.
- The back-end SQL instance has blockings, deadlocks, or resource contention.
- The on-premises data gateway is unhealthy.
- The on-premises data gateway should be scaled out.

When these problems occur, tune the back-end data source to avoid slowing the app's performance.

Specific data sources

Azure SQL Database

It's important to select the right tier for your business requirements. For more information, see Azure SQL Database documentation. A lower tier has some limitations and constraints. From a performance perspective, CPU, I/O throughput, and latency are important. Therefore, we recommend that you check the performance of the SQL database periodically, and check whether resource usage exceeds the threshold. For example, on-premises SQL Server normally sets the threshold of CPU usage to around 75 percent.

SharePoint

The SharePoint connector can be used to create apps that use data from SharePoint Lists. Here are some common performance problems and resolutions for SharePoint:

Avoid too many dynamic lookup columns: SharePoint supports various data types, including dynamic lookups such as Person, Group, and Calculated. If a list defines too many dynamic columns, it takes more time to manipulate these dynamic columns within SharePoint before returning data to the client running the canvas app. To avoid this, don't overuse the dynamic lookup columns in SharePoint. For example, use static columns to keep email aliases or people's names.

Carefully use the picture column and attachment: The size of an image and an attached file can contribute to a slow response while retrieving to the client. Review your list, and ensure that only necessary columns have been defined. The number of columns in the list affects the performance of the data requests. This is because the matched records, or the records up to the defined data row limits, are retrieved and transmitted

back to the client with all the columns defined in the list—even if the app doesn't use all of them.

Consider breaking up large lists: If you have a large list with hundreds of thousands of records, consider partitioning the list or splitting it into several lists based on parameters such as categories, or date and time. For instance, your data might be stored in different lists on a yearly or monthly basis. In such a case, you can design the app to let a user select a time window and retrieve the data within that range.

Dataverse

When you use Microsoft Dataverse as the data source, data requests go directly to the environment instance without passing through Azure API Management. So, it tends to be faster than other data sources. For more information, see Data call flow when connecting to Microsoft Dataverse.

Check custom table configurations: If custom tables are used in Dataverse, additional security configuration might be required for users to view the records with canvas apps. For more information, see Security concepts in Dataverse, Configure user security to resources in an environment, and Security roles and privileges.

Excel

The Excel connector allows a canvas app to connect to a table in an Excel file. However, this connector has limitations compared to other data sources. For example, it restricts the canvas app to loading data from the table only up to 2,000 records due to limited delegable functions. To load more than 2,000 records, partition your data in different data tables as other data sources.

Use the new Excel connector: Be sure to use the new Excel connector - Excel business online. It allows for multi-user access and handles contention issues better.

Only use the columns you need from large data lists in Excel: An app can perform slowly if the Excel file that has too many data tables or data tables that contain an immense amount of data over several columns. To ensure that your app isn't affected by this problem, define only the columns you need on the data table in an Excel file.

Note the limitations of Excel as a database. Excel isn't a relational database system : Any changes from an app are managed by Excel in the same way as if a user were changing data in an Excel file directly. If the app has a high number of reads, but fewer update operations, it might perform well. However, if the app requires heavy transactions, it can adversely affect the performance of the app. There's no specific

threshold value for the number of transactions. It also depends on the data being manipulated. Several other aspects also affect the app's performance, such as network overhead or the user's device.

Consider the differences in geographic location: The geographic location of the data and its distance from customer locations can be a performance issue. This issue can be amplified if a mobile client has limited bandwidth.

Enable Preload app for enhanced performance

You can optionally preload your app to increase performance.

1. Sign in to Power Apps.
2. On the left navigation pane, select **Apps**.
3. Select the app and then select **Settings** on the command bar.
4. In the **App settings** pance, set the **Preload app for enhanced performance** to **Yes**. The app will then pre-load.
5. For the changes to take effect for apps embedded in Teams, remove and add your app into Teams again.

> **Note**
>
> This makes the compiled app assets accessible via unauthenticated endpoints to enable loading them before authentication. However, users can still only use your app to access data via connectors only after authentication and authorization completes. This behavior ensures that the data an app retrieves from data sources won't be available to unauthorized users. Compiled app assets include a collection of JavaScript files containing text authored in app controls (such as PCF controls), media assets (such as images), the app name, and the environment URL the app resides in.
>
> In general, apps should retrieve media and information from data sources, through connections. If media and information must be added to the app, without coming from a connection, and it is considered sensitive you may want to disable this setting. Note, disabling this setting will result in users waiting a bit longer to access an app.

Feedback

Was this page helpful? Yes No

Provide product feedback

Build large and complex canvas apps

Article • 04/11/2023

Most of the articles in this section of the documentation cover the runtime performance of apps as experienced by the people who use them. This article covers app performance as experienced by the people who make them.

As apps become larger and more complex, Power Apps Studio needs to load and manage larger numbers of controls, formulas, and data sources, all with interdependencies that grow exponentially. Power Apps Studio can take longer to load, and features such as IntelliSense and color coding can lag. Use the recommendations that follow to better work with large and complex apps in Power Apps Studio. They can also help to improve your apps' runtime performance.

The examples in this article use the Hospital Emergency Response sample solution.

Use App.Formulas instead of App.OnStart

> **Tip**
>
> You can use **With** function and canvas component custom output properties as an alternative to named formulas.

The best way to reduce loading time for both Power Apps Studio and your app is to replace variable and collection initialization in **App.OnStart** with named formulas in **App.Formulas**.

Let's look at the following example, which uses App.OnStart.

Power Fx

```
// Get the color of text on a dark background.
Set(varColorOnDark,RGBA(0, 0, 0, 1));

// Get the color of the menu icons.
Set(varColorMenuIcon,"#0070a9");

// Get the styles for a form.
Set(varFormStyle,
    {
        DataCard: { Height: 50 },
        Title: { Height: 50, Size: 21, Color: varColorOnDark },
        Control: { Height: 50, Size: 18 },
        Label: { Size: 18, Color: varColorOnDark }
```

```
    }
);

ClearCollect(
    FacilitiesList,
    ForAll(
        Facilities,
        { Name: 'Facility Name', Id: Facility }
    )
);
If(
    Not IsBlank(Param("FacilityID")),
    Set(ParamFacility,
        LookUp(
            FacilitiesList,
            Id = GUID(Param("FacilityID"))
        ).Name
    );
);
```

Because they're a sequence of statements, your app must evaluate these **Set** and **Collect** calls in order before it can display the first screen, which makes the app load more slowly. And because the entire **App.OnStart** must be considered as a whole, order preserved, and errors aggregated before returning the final result, the formula is complex for Power Apps Studio to analyze.

There's a better way. Use **App.Formulas** instead and define these variables and collections as named formulas, as in the following example.

Power Fx

```
// Get the color of text on a dark background.
varColorOnDark = RGBA(0, 0, 0, 1);

// Get the color of the menu icons.
varColorMenuIcon = "#0070a9";

// Get the styles for a form.
varFormStyle =
    {
        DataCard: { Height: 50 },
        Title: { Height: 50, Size: 21, Color: varColorOnDark },
        Control: { Height: 50, Size: 18 },
        Label: { Size: 18, Color: varColorOnDark }
    };

FacilitiesList =
    ForAll(
        Facilities,
        { Name: 'Facility Name', Id: Facility }
    );
```

```
ParamFacility =
    If( Not IsBlank(Param("FacilityID")),
        LookUp(
            FacilitiesList,
            Id = GUID(Param("FacilityID"))
        ).Name,
        Blank()
    );
```

This change may seem small, but it can have a huge impact. Because each named formula is independent of the others, Power Apps Studio can analyze them independently, effectively splitting a large **App.OnStart** into smaller pieces. We've seen Power Apps Studio load time drop by as much as 80% with this change alone.

Your app also loads faster because it doesn't have to evaluate these formulas until it needs the result. The first screen of the app is displayed immediately.

Named formulas can't be used in all situations because you can't modify them or use them with **Set**. Some situations require the use of a state variable that can be modified. **Set** is perfect for these situations and you should continue to use it. But, more often than not, you're using global variables in **OnStart** to set up static values that don't change. In those cases, a named formula is the better choice.

Since named formulas are immutable, the prefix `var` (short for "variable") as a naming convention is no longer appropriate. We didn't change the names in this example because it would require changes to the rest of the app to match.

It's tempting to place a named formula in **App.OnStart**, but don't. They don't belong there. As an **On** behavior property, **App.OnStart** evaluates each of its statements in order, creating global variables, and talking to databases *only once, when the app is loaded*. Named formulas are formulas that define how to calculate something *whenever needed* and are always true. It's this formula nature that allows them to be independent and allows the app to finish loading before they're evaluated.

Split up long formulas

App.OnStart is one of the worst offenders for long formulas and definitely where you should start, but it isn't the only case.

Our studies have shown that nearly all apps with a long load time for Power Apps Studio have at least one formula of more than 256,000 characters. Some apps with the longest load times have formulas of more than 1 million characters. Formulas that long put a significant strain on Power Apps Studio.

To make matters worse, copying and pasting a control with a long formula duplicates the formula in the control's properties without it being realized. Power Apps is modeled after Excel, where multiple copies of a formula are common. However, in Excel formulas are limited to one expression and are capped at 8,000 characters. Power Apps formulas can grow much longer with the introduction of imperative logic and the chaining operator (`;` or `;;`, depending on locale).

The general solution is to split long formulas into smaller parts and reuse the parts, as we did in the previous section when we changed **Set**/**Collect** statements in **App.OnStart** to named formulas in **App.Formulas**. In other programming languages, reusable parts are often referred to as subroutines or user-defined functions. You can think of named formulas as a simple form of user-defined function with no parameters or side-effects.

Use named formulas everywhere

In the earlier example, we used named formulas as a replacement for **App.OnStart**. However, you can use them to replace a calculation anywhere in an app.

For example, one of the screens in the Hospital Emergency Response sample solution includes this logic in **Screen.OnVisible**:

Power Fx

```
ClearCollect(
    MySplashSelectionsCollection,
    {
        MySystemCol: First(
            Filter(
                Regions,
                Region = MyParamRegion
            )
        ).System.'System Name',
        MyRegionCol: First(
            Filter(
                Regions,
                Region = MyParamRegion
            )
        ).'Region Name',
        MyFacilityCol: ParamFacility,
          MyFacilityColID:  LookUp(
            FacilitiesList,
            Id = GUID(Param("FacilityID"))
        ).Id
    }
);
```

This formula can be split into a set of named formulas. It also makes the formula easier to read.

Power Fx

```
MyRegion = LookUp(
                    Regions,
                    Region = MyParamRegion
           );

MyFacility = LookUp(
                    FacilitiesList,
                    Id = GUID(Param("FacilityID")
            );

MySplashSelectionsCollection =
    {
        MySystemCol: MyRegion.System.'System Name',
        MyRegionCol: MyRegion.'Region Name',
        MyFacilityCol: ParamFacility,
        MyFacilityColID:  MyFacility.Id
    };
```

We extracted **ParamFacility** as a named formula earlier when we moved most of the **Set** calls from **App.OnStart** to named formulas in **App.Formulas**.

Named formulas are evaluated only when their values are needed. If the original intent of using **Screen.OnVisible** was to defer work until the screen is shown, then the work is still deferred as global named formulas in **App.Formulas**.

Use the With function

You can also use the **With** function in a formula to split up logic. Create a record in the first parameter with the values you want to use as fields, and then use those fields in the second parameter to calculate the return value from **With**. For example, the previous example can be written as just one named formula:

Power Fx

```
MySplashSelectionsCollection =
    With( { MyRegion: LookUp(
                            Regions,
                            Region = MyParamRegion
                        ),
            MyFacility: LookUp(
                            FacilitiesList,
                            Id = GUID(Param("FacilityID")
                        )
```

```
        },
        {
            MySystemCol: MyRegion.System.'System Name',
            MyRegionCol: MyRegion.'Region Name',
            MyFacilityCol: ParamFacility,
            MyFacilityColID:  MyFacility.Id
        }
    )
```

One downside to using **With** this way is that `MyFacility` can't use `MyRegion` because they're defined in the same **With** function, a problem that's not present with named formulas. One solution is to nest **With** functions and use the **As** keyword to name the record for each to give easy access to all the **With** variables.

Use canvas components

Canvas components are most often used to create a UI control that can be placed on the canvas just like a control. You can also use them without placing them in the UI to perform calculations with custom output properties as an alternative to named formulas. Canvas components are easy to share across apps with component libraries and, unlike named formulas, are fully supported. However, they're harder to configure and use than named formulas.

To split logic:

1. In Power Apps Studio, switch to the **Components** tab in the **Tree view**.
2. Create a new component.
3. In the **Properties** pane, turn on **Access app scope**.
4. Add a custom property.
5. Set the **Property type** to **Output** and the **Data type** as appropriate.
6. Select **Create**.
7. In the property picker next to the formula bar at the top of the screen, select the new property.
8. Write the formula for the logic to split and reuse.

To use the logic:

1. Switch to the **Screens** tab in the **Tree view**.
2. In the **Insert** pane, expand **Custom** and insert your component.
3. To calculate a value with the property, use *ComponentName.PropertyName*.

Use Select with a hidden control for imperative logic

Imperative logic is used to modify state with **Set** and **Collect**, notify the user with **Notify**, navigate to another screen or app with **Navigate** and **Launch**, and write values to a database with **Patch**, **SubmitForm**, or **RemoveIf**.

Named formulas and canvas component custom output properties don't support imperative logic. A common way to split up imperative logic is to use the **OnSelect** property of a hidden control.

1. Add a **Button** control to a screen.
2. Set the **OnSelect** property to the imperative logic you want to execute.
3. Set the **Visible** property to false, since there's no need for the user to see or interact with it.
4. Call `Select( Button )` when you want to execute the imperative logic.

For example, one of the screens in our sample has the following **OnSelect** property on a **Button** control. (This simple example is for illustration purposes only. Normally, you would only use this technique for longer formulas.)

Power Fx

```
btnAction_17.OnSelect =
    Trace("Feedback Screen: Submit Button",TraceSeverity.Information);
    If(
        // Proceed if all forms are validated.
        And(
            FormFeedback.Valid
        ),

        // Set the updates to static variables.
        Set(updatesFeedback,Patch(Defaults('App Feedbacks'),
FormFeedback.Updates));
        // Submit the first form. Subsequent actions can be found in the
OnSuccess.
        SubmitForm(FormFeedback);
        ,

        Notify("Please complete all fields before proceeding",
               NotificationType.Warning,2000)
    );
```

To split this logic into parts, we can put portions onto separate **Button** controls and **Select** them from the original:

Power Fx

```
btnTrace.OnSelect =
    Trace("Feedback Screen: Submit Button",TraceSeverity.Information);
```

```
btnSubmit.OnSelect =
    If(
        // Proceed if all forms are validated.
        And(
            FormFeedback.Valid
        ),

        // Set the updates to static variables.
        Set(updatesFeedback,Patch(Defaults('App Feedbacks'),
FormFeedback.Updates));
        // Submit the first form. Subsequent actions can be found in
OnSuccess.
        SubmitForm(FormFeedback);
        ,

        Notify("Please complete all fields before proceeding",
               NotificationType.Warning,2000)
    );

btnAction_17.OnSelect =
    Select( btnTrace );
    Select( btnSubmit );
```

This technique only works on the same screen. Other techniques that are slightly more complicated work across screens, like using the **Toggle** control, setting **OnCheck** to the logic you want to run, and setting **Default** to a global variable and then toggling the global variable with `Set( global, true ); Set( global, false )` at the point you want to run the logic.

In this example, some logic splitting had already been done. The comment mentions that "Subsequent actions can be found in OnSuccess." This event runs imperative logic after the record has been successfully submitted, a solution specific to the **SubmitForm** function.

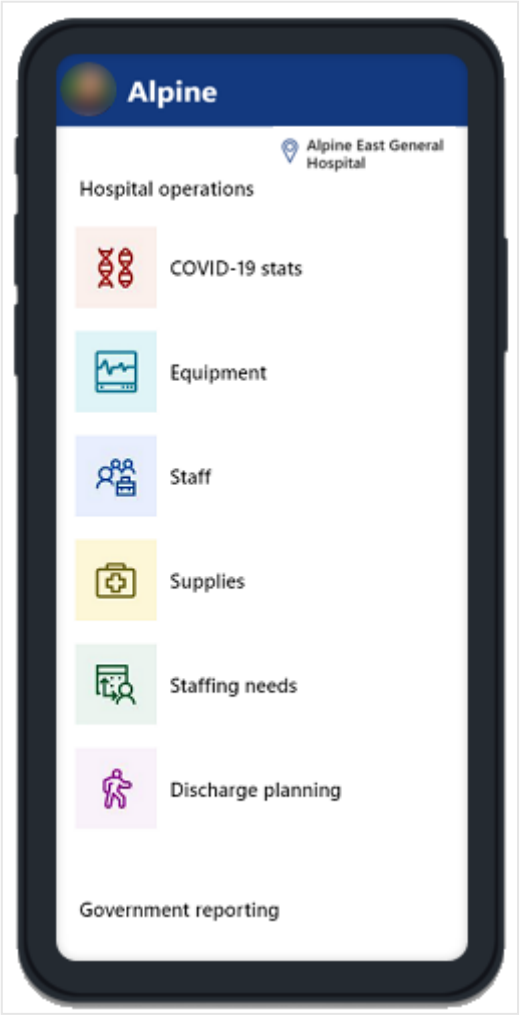
Partition the app

Some apps grow to thousands of controls and hundreds of data sources, which slows Power Apps Studio. As with long formulas, large apps can be split into smaller sections that work together to create one user experience.

Separate canvas apps

One approach is to implement sections in separate canvas apps and use the **Launch** function to navigate between the separate apps and pass needed context.

This approach was used in the Hospital Emergency Response sample solution. Separate apps manage each of the major areas of the overall app. The apps share a common switchboard component through a component library that each app shows on its startup screen:



When the user selects an area, the component uses metadata about the apps available and which app is hosting the component. If the desired screen is in this app (that is, **ThisItem.Screen** is not blank), then a **Navigate** call is made. But if the desired screen is in a different app (that is, **ThisItem.PowerAppID** is not blank), then a **Launch** function is used with the App ID of the target and the FacilityID context:

Power Fx

```
If(
    IsBlank(ThisItem.Screen),
    If(IsBlank(ThisItem.PowerAppID),
        Launch(ThisItem.URL),
        Launch("/providers/Microsoft.PowerApps/apps/" & ThisItem.PowerAppID,
               "FacilityID", Home_Facility_DD.Selected.Id)
    ),
    Navigate(
        ThisItem.Screen,
        Fade
```

```
    )
);
```

State in the original app is lost when another app is launched. Be sure to save any state before you call the **Launch** function. Write it to a database, call **SaveData**, or pass state to the target app with parameters that are read with the **Param** function.

Model-driven app with custom pages

Sections can also be implemented as custom pages. Custom pages act as a mini canvas app with a model-driven app container for navigation.

Feedback

Was this page helpful? Yes No

Provide product feedback

Build business logic

Article • 10/22/2024

Power Platform offers various capabilities for makers to build business logic for Power Apps. In this article, you find an overview of the key methods and offer guidance to help determine the most suitable approach building business logic for Power Apps.

Use Power Fx in your app

Power Fx is the low code functional programming language shared by Excel and Power Platform. When used with Power Platform connectors, makers can access data and create custom app logic. Power Fx supports the following characteristics:

- **Live** – Canvas apps recalculate just as Excel spreadsheets do. As users interact with the app, Power Fx is in the background making data requests and keeping the state of the app up to date with Dataverse.
- **Delegation** – Power Fx optimizes data handling by delegating operations to the server whenever possible. Functions like Filter(), Lookup(), and Search() enable server-side data filtering, ensuring that only the necessary data is retrieved to support the app's functionality and user experience. In cases where delegation isn't possible, functions run locally within the JavaScript environment of the browser.
- **Optimized for Dataverse** – Dataverse serves as the primary data storage option for Power Apps, offering direct, and low-latency access due to fewer intermediary layers. Power Fx is also compatible with Dataverse's diverse relationship types, including many-to-one and one-to-many associations.
- **Offline** – The Power Apps mobile app enables offline use of Dataverse data, allowing users to use the app in the field allowing a seamless connection. Any changes made offline are synchronized when internet connection is available. The app's predefined business logic remains operational, even without an internet connection.

Use cloud flows in your app

Power Automate is a low code workflow service built on top of the Power Platform connector ecosystem. Power Automate adds the following capabilities when building Power Apps:

- **Asynchronous** – A Power Automate cloud flow is inherently asynchronous. This means that when a flow is initiated, it's using a queuing system to manage the various subtasks. The asynchronous nature of Power Automate means makes it well-suited for longer running complex sequences of logic.
- **Detailed logging** – All flows create a record of what happened when they executed in the form of Run History providing traceability and ensuring auditing record for what happened and why.
- **Multi-connector** – Although it's possible to create multi-connector Power Apps logic, due to the live nature of Power Apps, as you increase the number of connectors performance degrades. These complex multi-connector scenarios are a great place to use Power Automate which can offload these cases from live execution in the app.

Choose between Power Fx and cloud flows for your app

We recommend using Power Fx for developing business logic in Power Apps. While it's a powerful tool, there are scenarios where integrating other tools might be more appropriate.

Low latency use cases

Power Fx excels in delivering low latency for Power Apps, making it the go to choice for developing responsive business logic. However, the actual latency experienced can vary based on the complexity of the operations and the volume of data being processed. It's crucial to understand that while Power Fx can facilitate swift interactions, the design, and objectives of your application play a pivotal role in its overall performance. For more information, see performance and optimization articles earlier in this section.

When it comes to accessing data efficiently, the combination of Dataverse and Power Fx offers the quickest solution.

For more information, see Data call flow with Microsoft Dataverse.

Complex sequences and use of multiple connectors

When dealing with complex sequences of action that span various connectors, Power Automate serves as an effective asynchronous solution to delegate processing away

from Power Apps. This asynchronous capability lets Power Apps kick off a workflow and proceed with other tasks without having to pause for a response.

Design app experience around inherent task latency

When designing user experiences, it's essential to consider the latency that comes with executing complex tasks.

There are two strategies:

1. Pause the user experience until the task finishes, providing a visual indicator of progress.
2. Offload the complex task to Power Automate, allowing the user experience to continue uninterrupted.

Related content

Overview of creating performant apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Power Apps coding guidelines

Summarize this article for me

Creating a simple canvas app is straightforward, but as the application's complexity grows, maintaining high code quality becomes essential. These guidelines describe standards for naming objects, collections, and variables, and provide recommendations for developing consistent, performant, and easily maintainable apps.

The guidelines offer essential guidance for developers who design, build, test, deploy, and maintain Power Apps apps in various environments, including small businesses, corporations, and government entities.

> **Tip**
>
> Use the Power CAT Toolkit to perform code reviews. The toolkit incorporates many of the coding guidelines from this documentation, ensuring that your solutions are efficient and maintainable. The toolkit automatically flags any patterns that don't adhere to these best practices, so you can address potential issues early in the development process. This proactive approach enhances the quality of your code and helps streamline the review process, making it easier to maintain high standards across your projects. Integrating the Power CAT Toolkit into your development workflow ensures that your solutions are built on a solid foundation and leads to more reliable and scalable solutions.

Next step

App design guidelines

Last updated on 02/28/2026

Power Apps and Microsoft Teams integration

09/03/2025

Microsoft Teams enables people around the world to create, collaborate, and communicate. With more than 75 million daily active users, Teams is how people are getting work done.

Power Apps can be used in four scenarios with Teams.

Expand table

Scenario	Description
Embed a canvas app as a tab app	Use the **Power Apps tab** for Teams to embed a canvas app as a tab app in Teams. The canvas app integrates business data from a wide variety of Microsoft and third-party data sources, including Dataverse.
Embed a canvas app as a personal app	Embed a canvas app as a first-class Teams personal app. The canvas app integrates business data from a wide variety of Microsoft and third-party data sources, including Dataverse.
Embed a model-driven app as a tab app (preview)	Use the **Power Apps tab** for Teams to embed a model-driven app as a tab app in Teams.
Embed a model-driven app as a personal app (preview)	Embed a model-driven app as a first-class Teams personal app.
Create apps in Teams backed by Microsoft Dataverse for Teams	Use the new Power Apps app in Teams to create canvas apps within Teams backed by Dataverse for Teams, the new built-in, low-code data platform for Teams. Dataverse for Teams empowers users to build custom apps, bots, and workflows—by using Power Apps, Microsoft Copilot Studio, and Power Automate, respectively—without ever leaving the Teams interface.

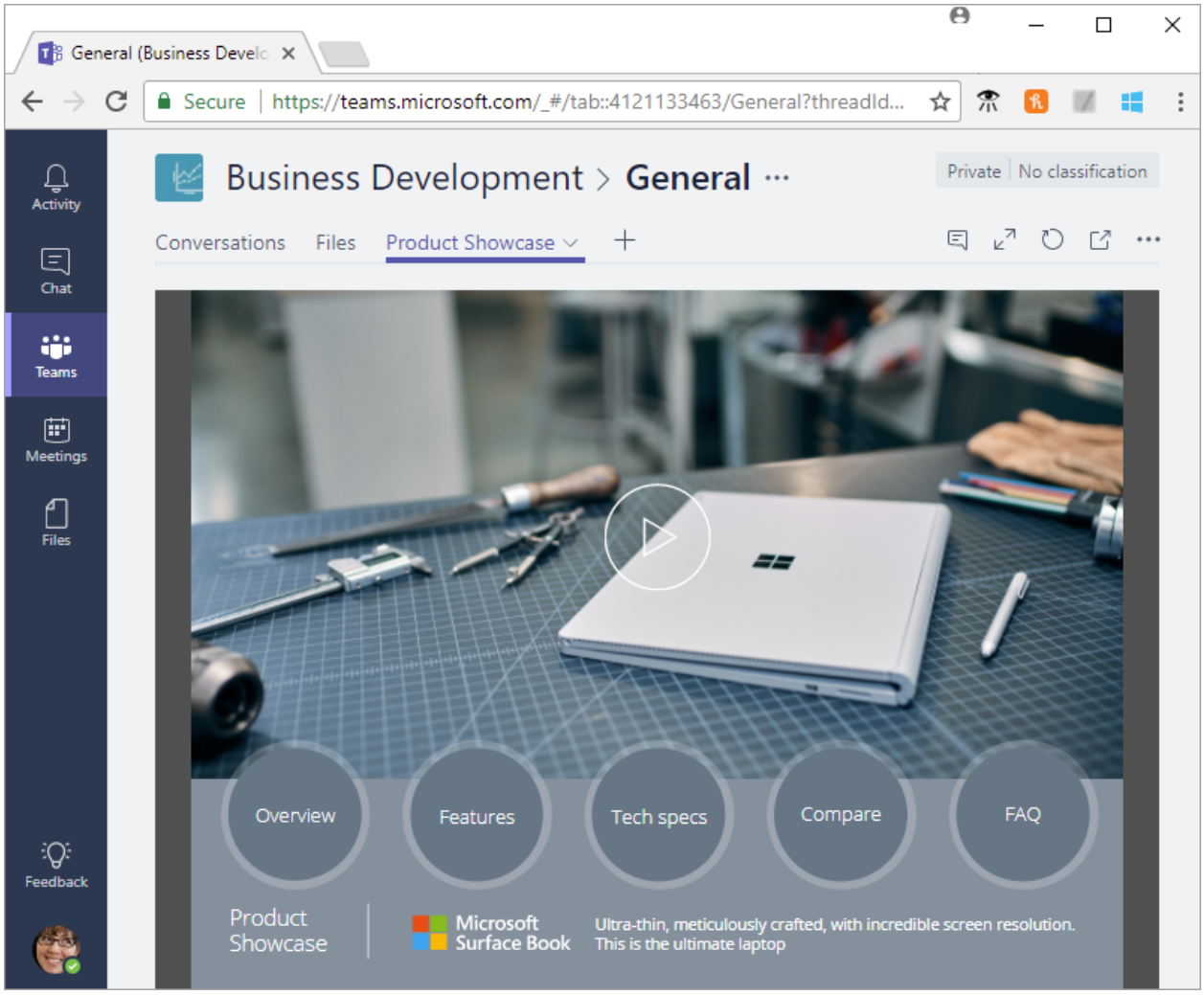
Related articles

What is Dataverse for Teams?
Create your first app in Teams
Use flows in Microsoft Teams
Microsoft Copilot Studio in Teams

Embed a canvas app as tab app in Teams

Article • 01/30/2024

You can customize the Teams experience by adding Power Apps canvas apps to your channels in Teams using the **Power Apps** tab. In this topic, you learn how to add the Product Showcase sample app template to a Teams channel, and then open the app from that channel.



If you're not signed up for Power Apps, sign up for free before you begin.

Prerequisites

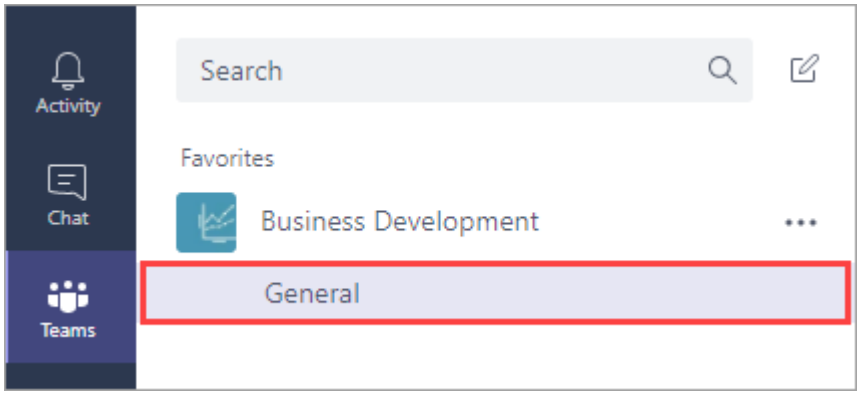
To follow this procedure, you need an Office 365 subscription and a channel in Teams.

Sign in to Power Apps

Sign into Power Apps at https://make.powerapps.com.

Add an app

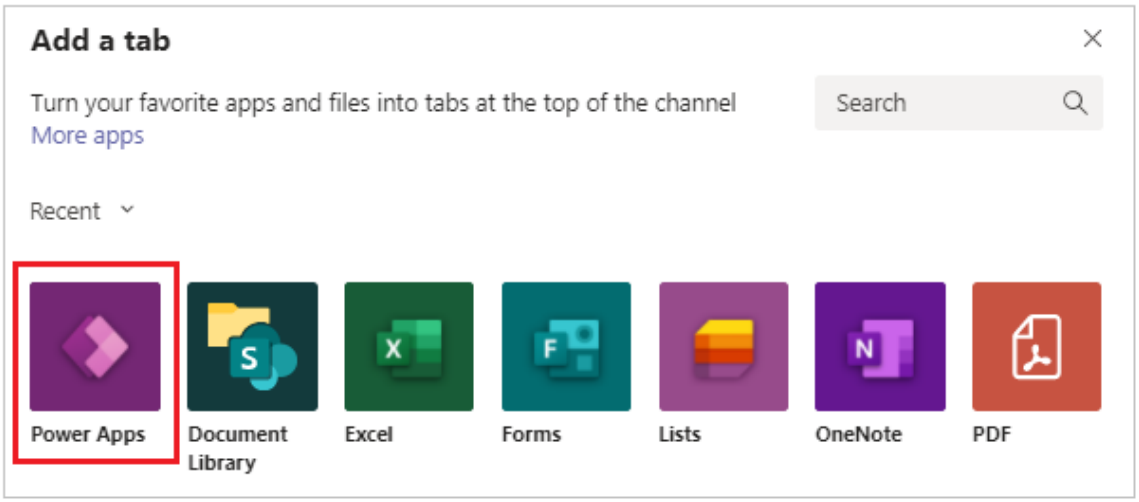
1. In Microsoft Teams, select a team, and a channel under that team. In this example, it's the **General** channel under the **Business Development** team.



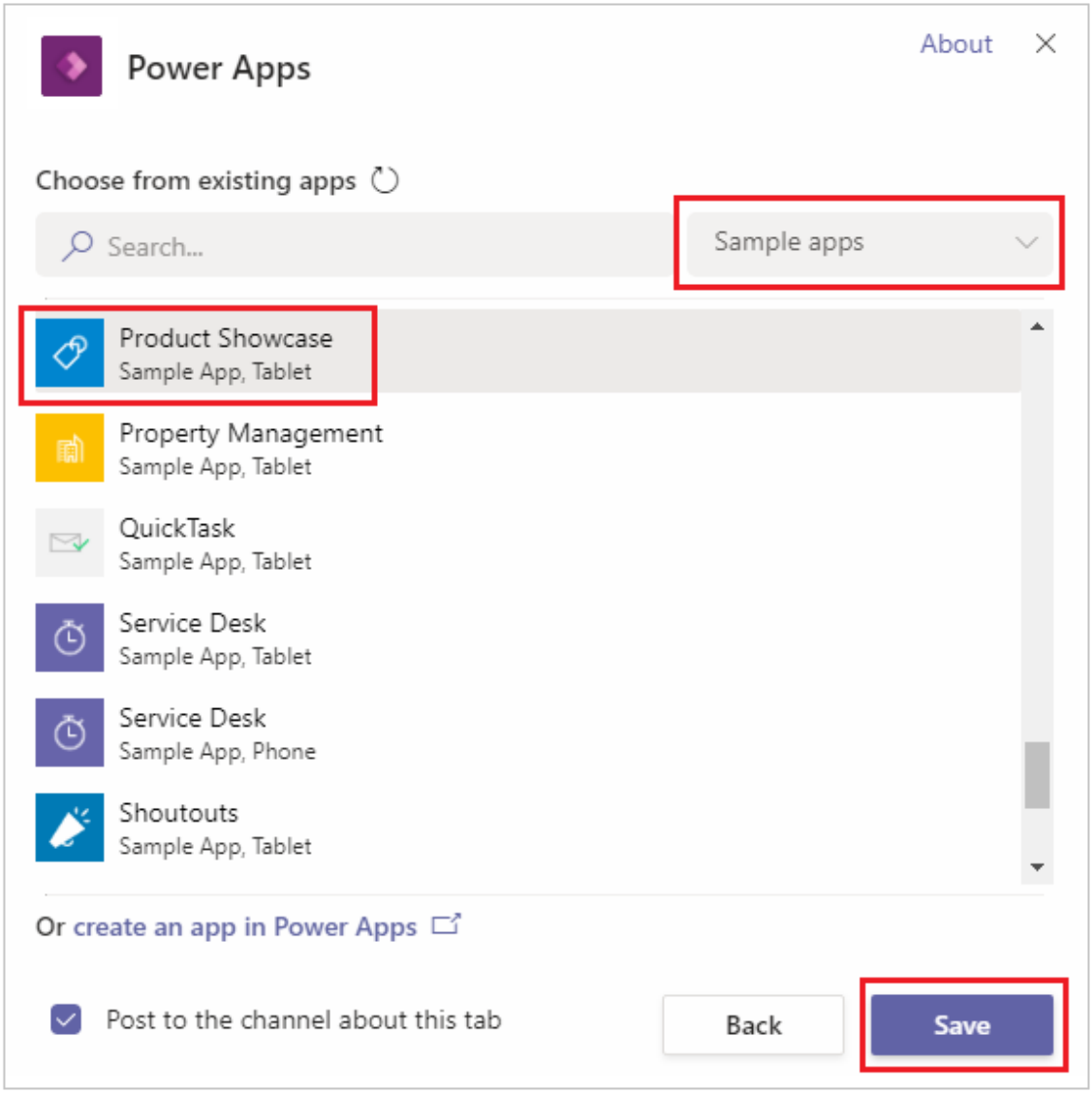
2. Choose + to add a tab.



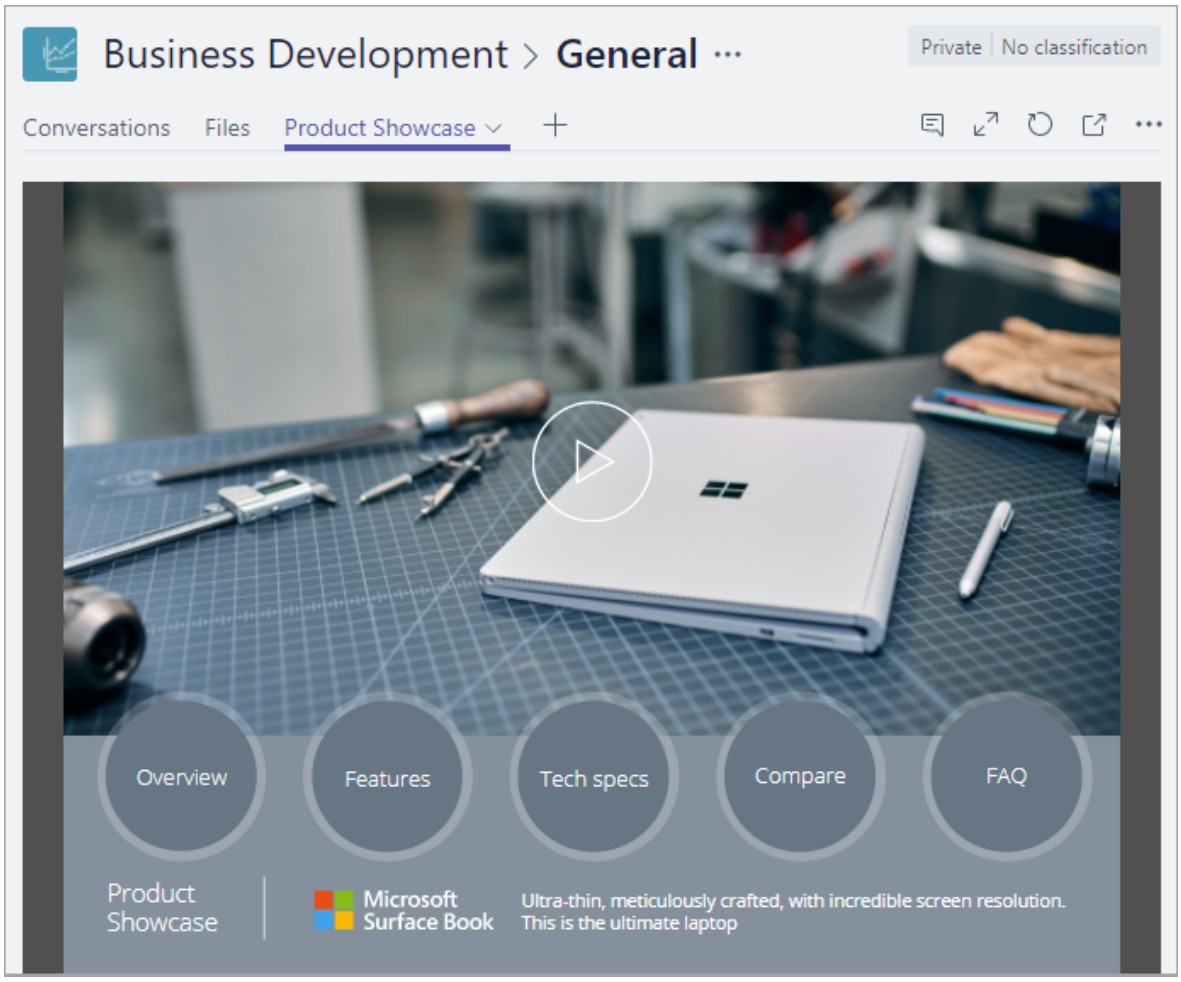
3. In the **Add a tab** dialog box, choose **Power Apps**.



4. Choose **Sample app templates** > **Product Showcase** > **Save**.



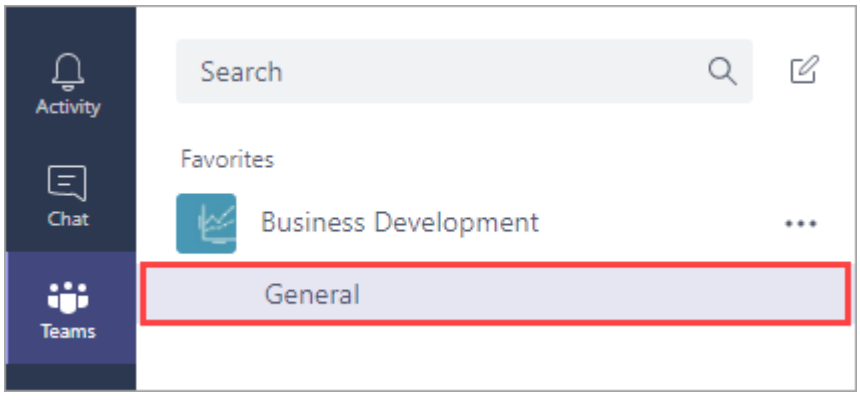
The app is now available to use in the channel.



> ⓘ **Note**
>
> You must **share** your own apps before you add them to Teams (sample app templates are shared by default).

Open an app

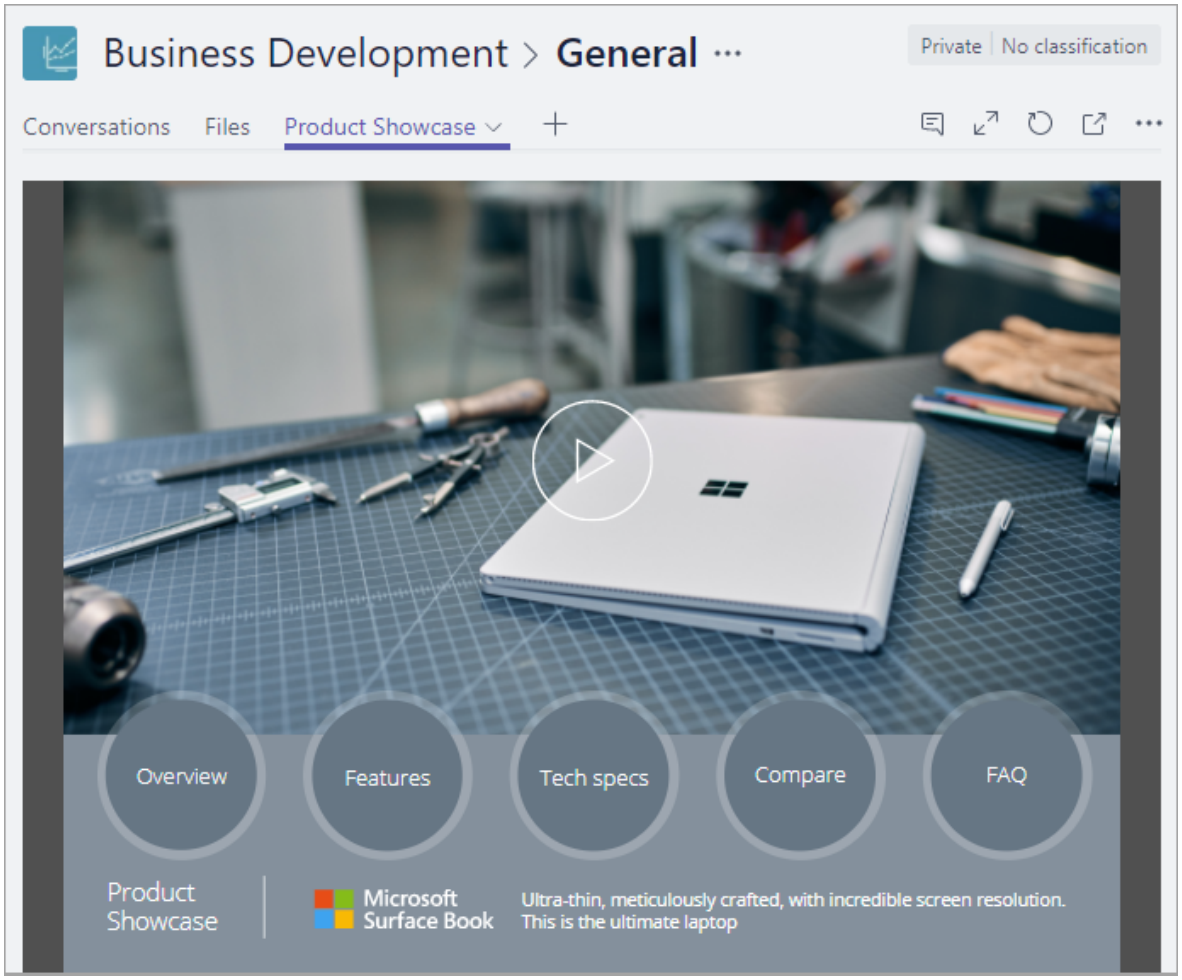
1. In Microsoft Teams, choose the team and the channel that contains the app.



2. Choose the **Product Showcase** tab.



The app opens in the channel.



Known issues

In the desktop app for Microsoft Teams:

- Apps must load content such as images and .pdf files over a secure (https) connection.
- Not all sensors, such as **Acceleration**, **Compass**, and **Location**, are supported.
- Only these audio formats are supported: AAC, H264, OGG Vorbis, and WAV.

In the mobile app for Microsoft Teams:

- End users can't download attachments from canvas apps. As a workaround, use Microsoft Teams desktop to access the canvas app and download the attachment.

Clean up resources

To remove the app from the channel, choose the **Product Showcase** tab > **Remove**.

Embed a canvas app as personal app in Teams

08/19/2025

You can share an app you've created by embedding it directly into Microsoft Teams. When completed, users can select + to add your app to any of **your** team channels or conversations in the team you are in. The app appears as a tile under **Tabs for your team**.

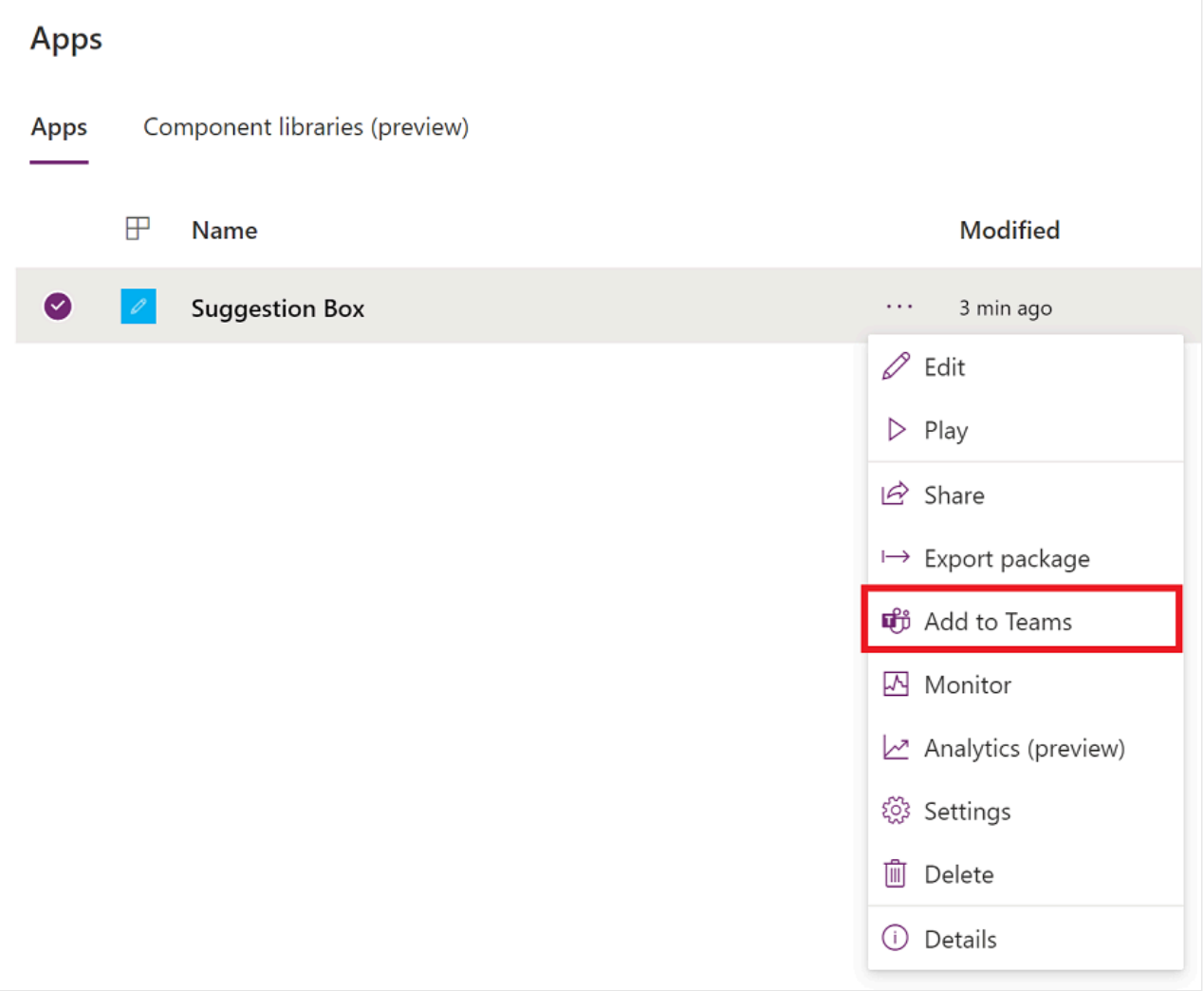
> **Note**
>
> Team custom app policies must be set to allow uploading custom apps. If you are unable to embed your app in Teams, check with your administrator to see if they've setup custom app settings.

Prerequisites

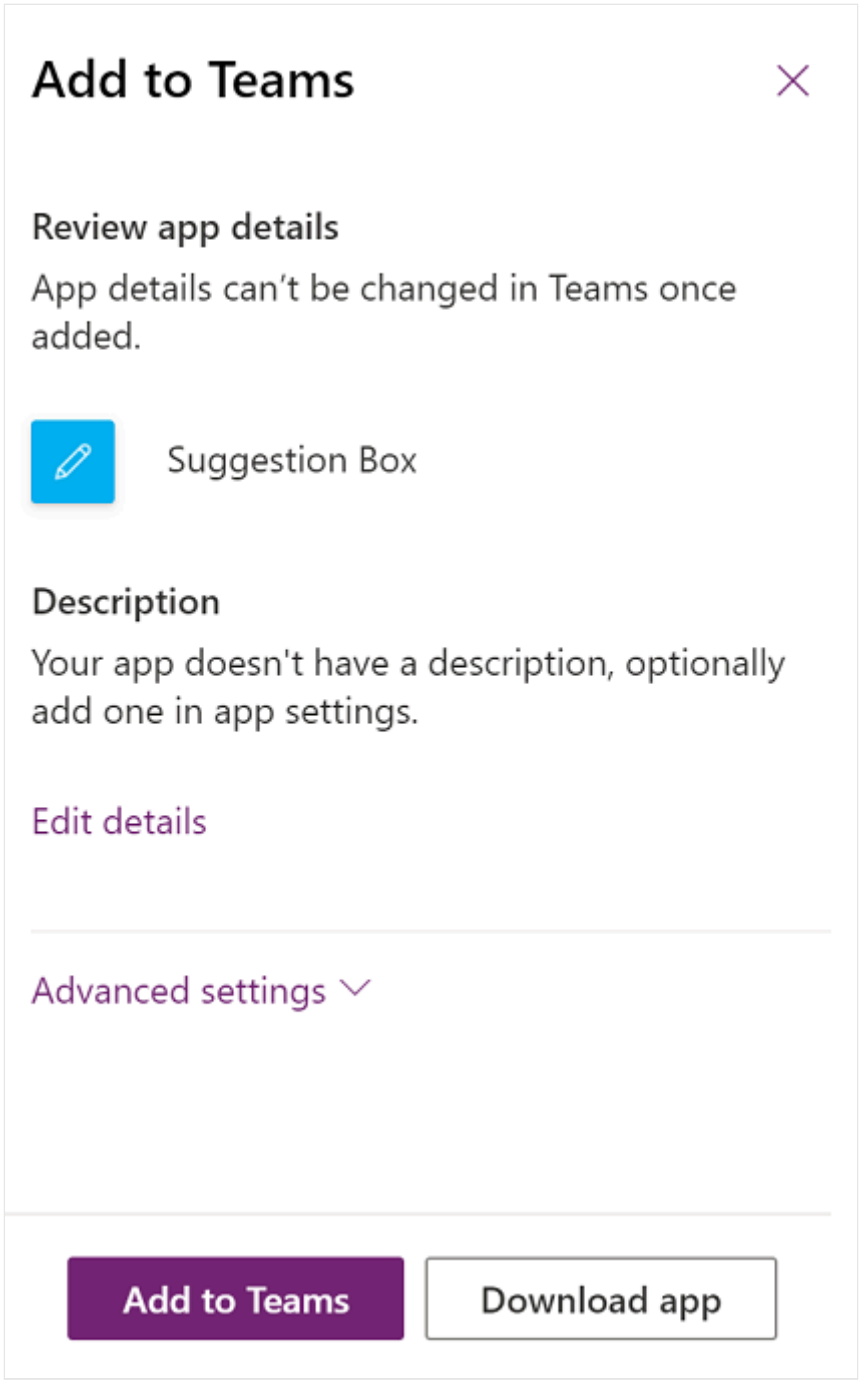
- You need a valid Power Apps license.
- To embed an app into Teams, you need an existing app created using Power Apps.

Add to Teams

1. Sign in to Power Apps, and then select **Apps**.
2. Select **More actions** (...) for the app you want to share in Teams, and then select **Add to Teams**.

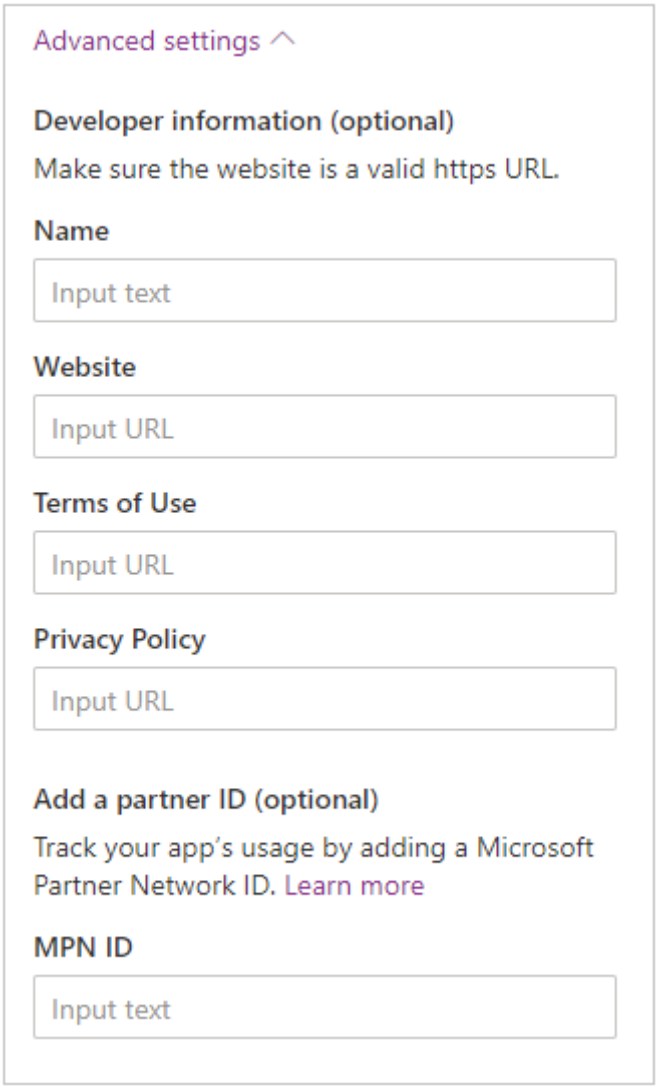


Add to Teams panel opens on the right-side of the screen.

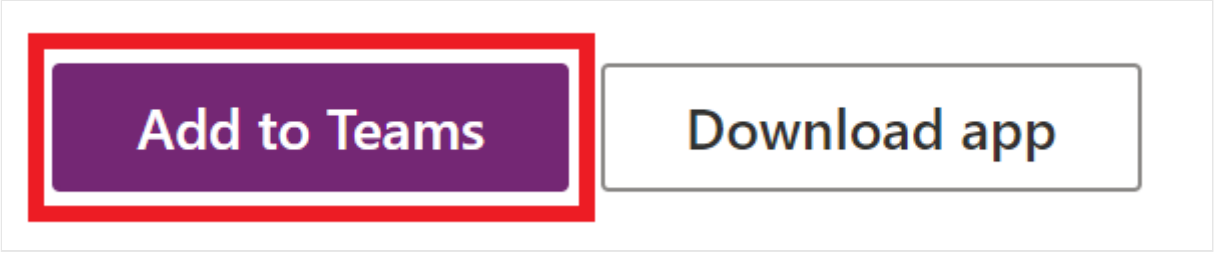


3. (Optional) If the app doesn't have any description, select **Edit details** to open the app in Power Apps Studio to add.

4. (Optional) Select **Advanced settings** to add additional details such as *Name*, *Website*, *Terms of Use*, *Privacy Policy*, *MPN ID* (Microsoft Partner Network ID).



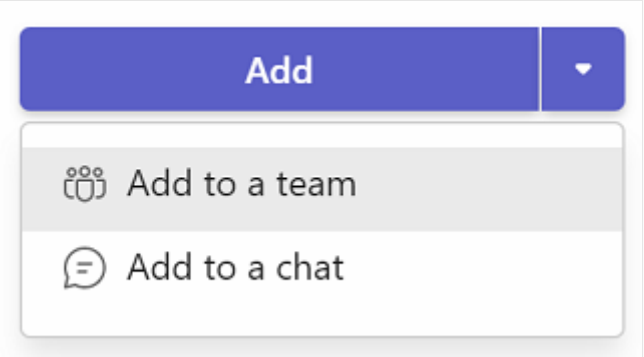
5. Select **Add to Teams** and you'll be asked to open Teams.



> **Tip**
>
> If you want, you can also choose **Download app** to download the app, and then use the steps described in Upload your app in Microsoft Teams article to upload the app to Teams.

6. Select **Open Microsoft Teams** to open Teams app, or select **Cancel** followed by **Use the web app instead** to open Teams web.

7. Select **Add**. You can also choose to add the app to a specific team using **Add to team**, or a specific chat using **Add to chat**.



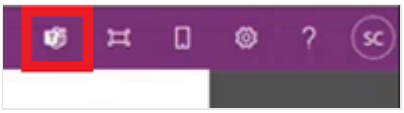
The app is now added to Teams.

> **Tip**
>
> You can also pin the app for easy access.

Add to Teams using the web player

You can also add canvas apps to Teams using the web player.

1. Go to Power Apps > **Apps** > select the app to launch in a new browser tab.
2. Edit the app > select ▷ or press **F5** to play the app.
3. After the app is open using the web player, choose the **Add to Teams** button from the top-right side of the screen.



4. When prompted, select **Cancel**.
5. Select **Launch it now** to open the Teams app. Alternatively, you can also choose **Use the web app instead** to open the app in Teams web.

> **Note**
>
> - To add canvas apps to Teams using the web player:
>   - Your organization must have **Allow interaction with custom apps** turned on.
>
>   More information: Manage custom app policies and settings in Microsoft Teams

- Your organization must allow **Shared Power Apps**. More information: Manage Microsoft Power Platform apps in the Microsoft Teams admin center
- You can only add apps to Teams that belong to the same tenant as your Teams account. Otherwise, you'll see an error "App was not found in tenant".

Publish the app to the Teams catalog

If you're an admin, you can also publish the app to the Microsoft Teams catalog.

Use context from Teams

To build deeply integrated apps with Teams, you can use Team's context variables with the `Param()` function. For example, use the following formula in screen's **Fill** property to change the background of app based on user's theme within Teams:

```
Switch(
        Param("theme"),
        "dark",
        RGBA(
            32,
            31,
            31,
            1
        ),
        "contrast",
        RGBA(
            0,
            0,
            0,
            1
        ),
        RGBA(
            243,
            242,
            241,
            1
        )
    )
```

To test the app, publish it and then play it within Teams.

The following context variables from Teams are supported:

- locale
- channelId
- channelType
- chatId
- groupId (not supported with private channels)
- hostClientType
- subEntityId
- teamId
- teamType
- theme
- userTeamRole

> **Note**
>
> This feature was added in March, 2020. If you embedded your app within Teams before this, you may need to re-add your app to Teams to use this functionality.

Improve the performance of your app

You can optionally preload your app within Teams to increase performance. For mor information: Enable Preload app for enhanced performance.

See also

Welcome to Microsoft Teams

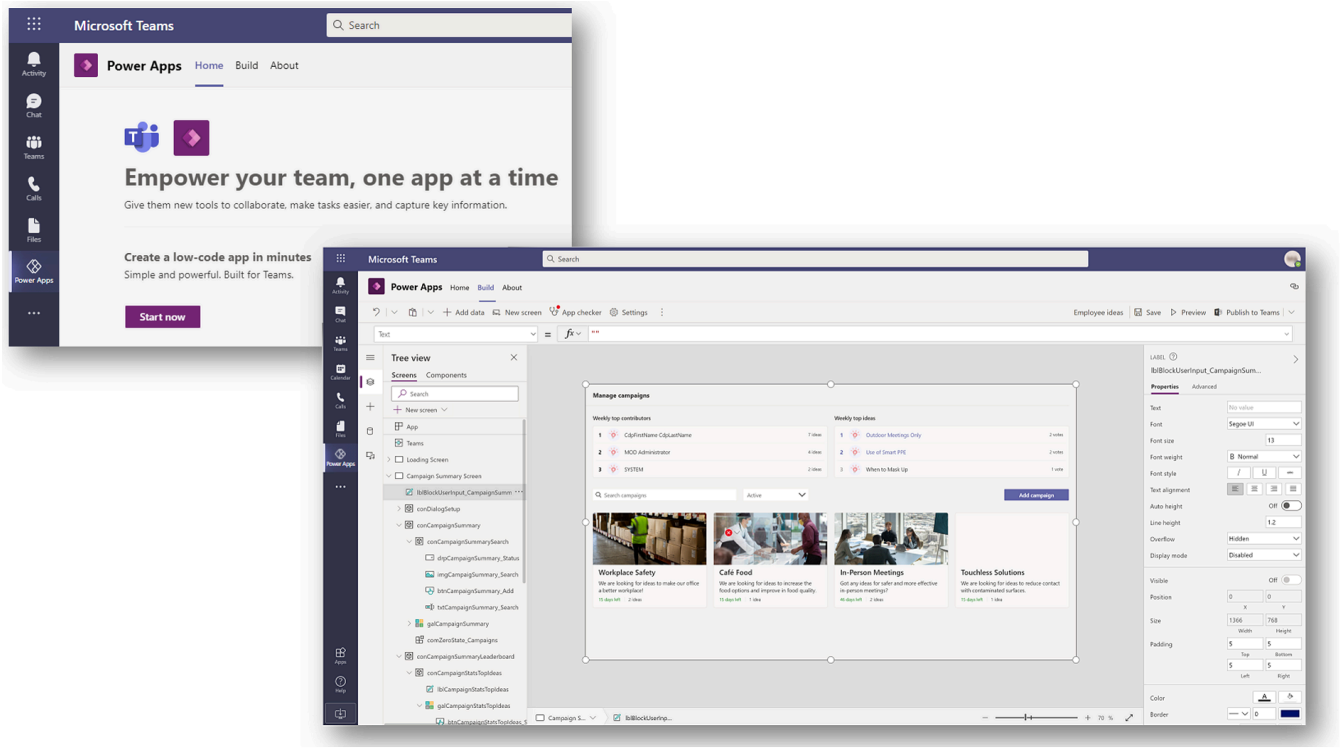
Create apps in Microsoft Teams by using Power Apps

09/03/2025

With the significant number of enterprise employees working remotely, and millions of them meeting and collaborating through Microsoft Teams, there's significant interest in building low-code and no-code applications that can make remote work easier and more productive.

Introducing Microsoft Dataverse for Teams (formerly called Project Oakdale), a built-in, low-code data platform for Teams that empowers you to build custom apps and workflows in Teams by using Power Apps and Power Automate. Dataverse for Teams, built on Microsoft Dataverse (formerly called Common Data Service), provides relational data storage, rich data types, enterprise-grade governance, and one-click solution deployment to the Teams app store.

The new Power Apps app in Teams provides an integrated experience for app makers to create and edit apps and workflows within Teams, and quickly publish and share them for anyone on the team to use, without having to switch between multiple apps and services. With Power Apps Studio embedded in the Power Apps app in Teams and the new built-in data platform providing an easy-to-use, editable data table, you can quickly build apps based on custom data tables that are Teams-specific and scenario-specific.



To help organizations realize value faster, you can use *templates*, which are purpose-built apps for common application patterns that can be easily discovered and installed by members of the

team. Although these templates can work without modification, members of the team can customize and extend them to meet the specific needs of their business.

Access and authorization align with the Microsoft Teams security model and include security groups that enable access for owners, members, and guests of the team that your app is associated with.

Licensing requirements

The ability to create apps within Microsoft Teams will be available as part of select Microsoft 365 subscriptions. For detailed information, see Licensing and restrictions in the Power Platform admin guide.

Get started with creating apps in Teams

There are two ways to use the apps created by using Power Apps in Teams:

- You can use the Power Apps app from the default Teams catalog as a personal app and create apps to share with teams in Teams. More information: Install the Power Apps personal app

When you create an app in Teams with the Power Apps app for the first time, or install an app created with Power Apps from the app catalog for the first time, a new Dataverse for Teams environment is created for the selected team. The Dataverse for Teams environment is used to store, manage, and share team-specific data, apps, flows, and chatbots. Each team can have one Dataverse for Teams environment, and all data, apps, and flows created with the Power Apps app inside a team are available from that environment. More information: About the Dataverse for Teams environment

When you create an app in Teams with the Power Apps app for the first time, or install an app created with Power Apps from the app catalog for the first time, you get to choose which team is responsible for customizing, maintaining, and sharing the app.

After selecting a team, we will create a new Dataverse for Teams environment if one doesn't already exist. This will provide the necessary workspace for your team to collaborate on all your data, apps, flows, and bots. More information: About the Dataverse for Teams environment

Get started with creating flows and chatbots in Teams

You can create:

- Power Automate flows by using the new Power Apps app in Teams. More information: Create flows using the Power Apps app in Teams
- Chatbots by using the new Microsoft Copilot Studio app in Teams. More information: Microsoft Copilot Studio in Teams

Next steps

Install the Power Apps personal app
Create your first app in Teams

Related articles

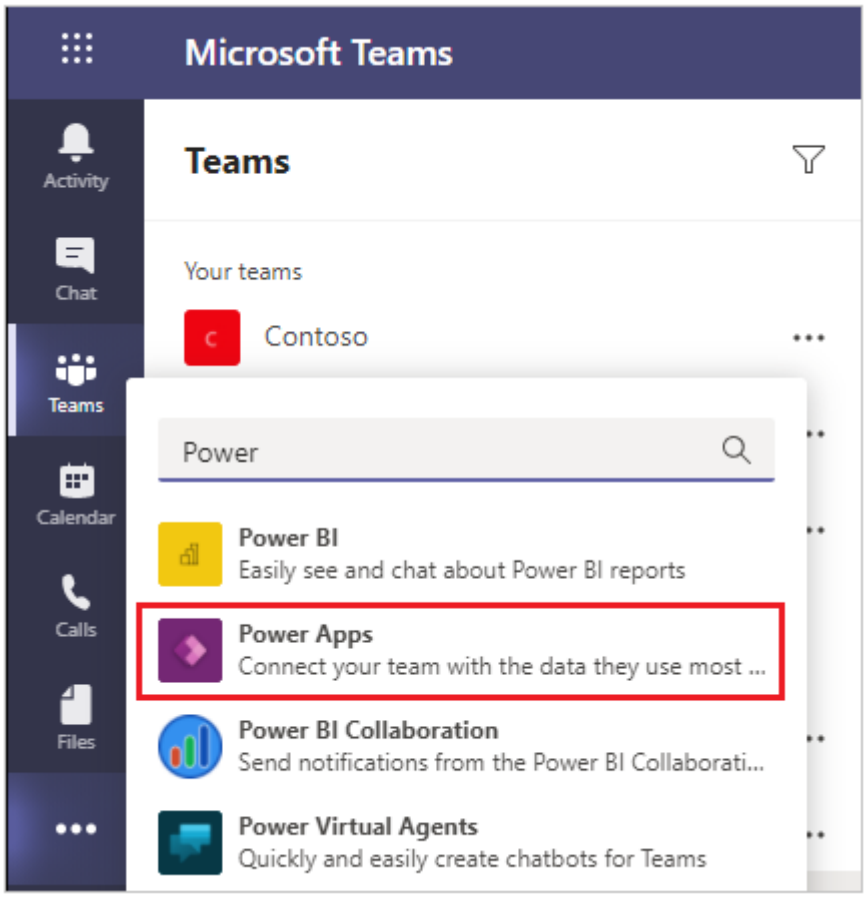
Administer Dataverse for Teams environment

Install the Power Apps personal app in Microsoft Teams

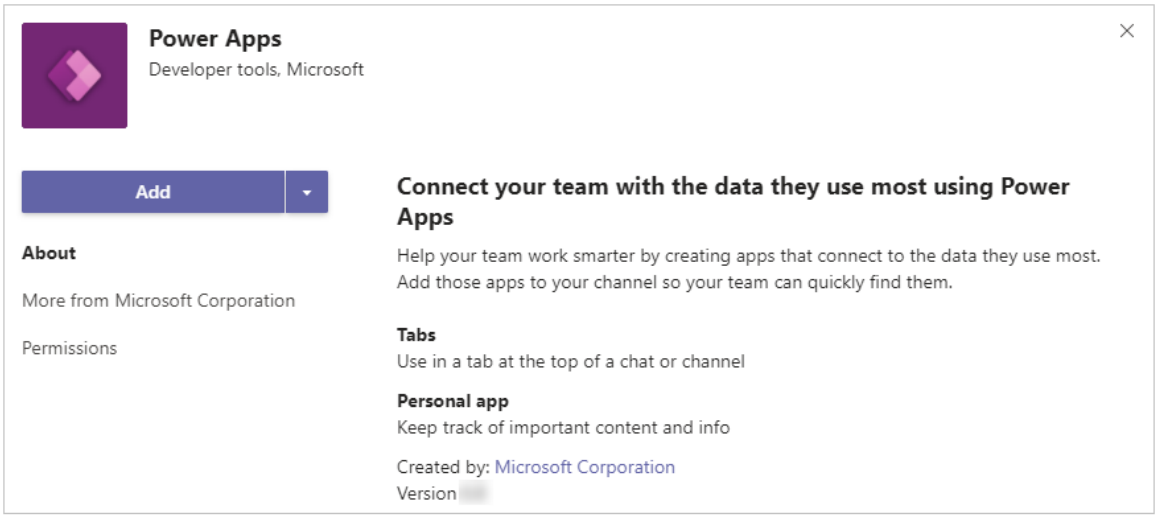
Article • 03/03/2023

A *personal app* is a Teams application that has a personal scope. Power Apps is by default available for you to select and add to a tab in the Teams channel of your choice. More information: Personal apps

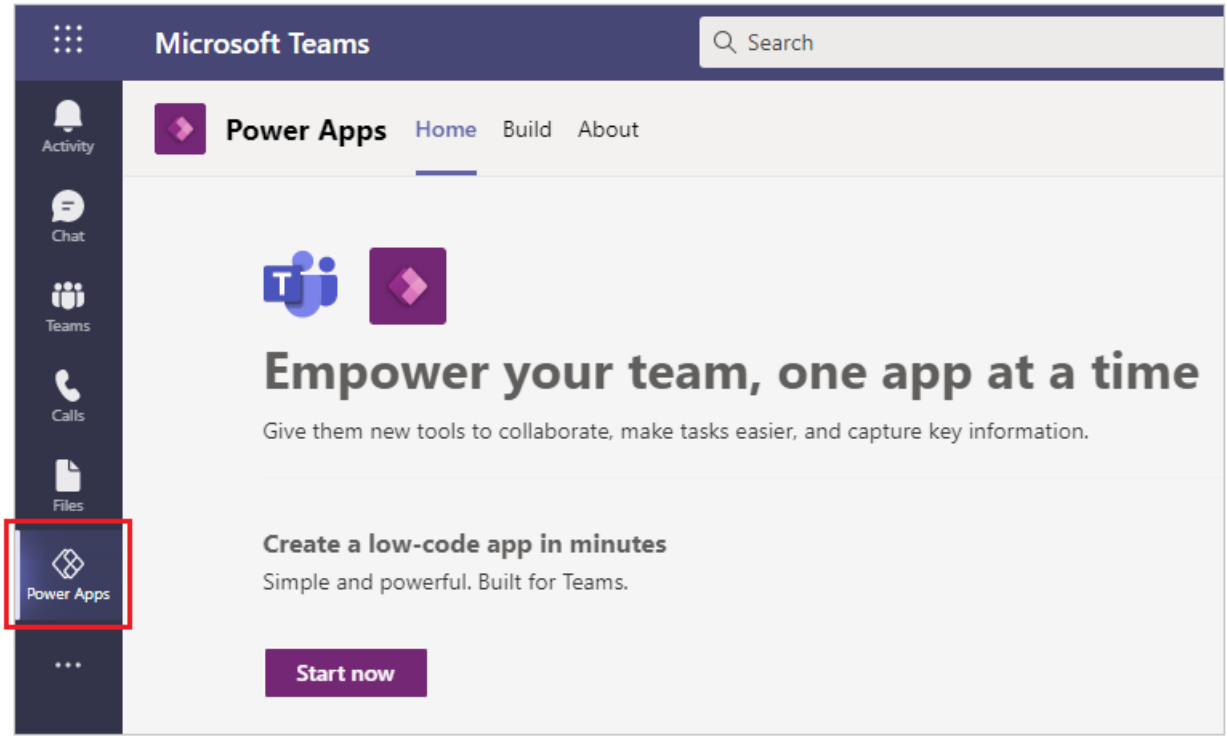
1. Open Microsoft Teams, and then select **More added apps** (...).
2. Search for **Power Apps**.



3. Select **Add**.



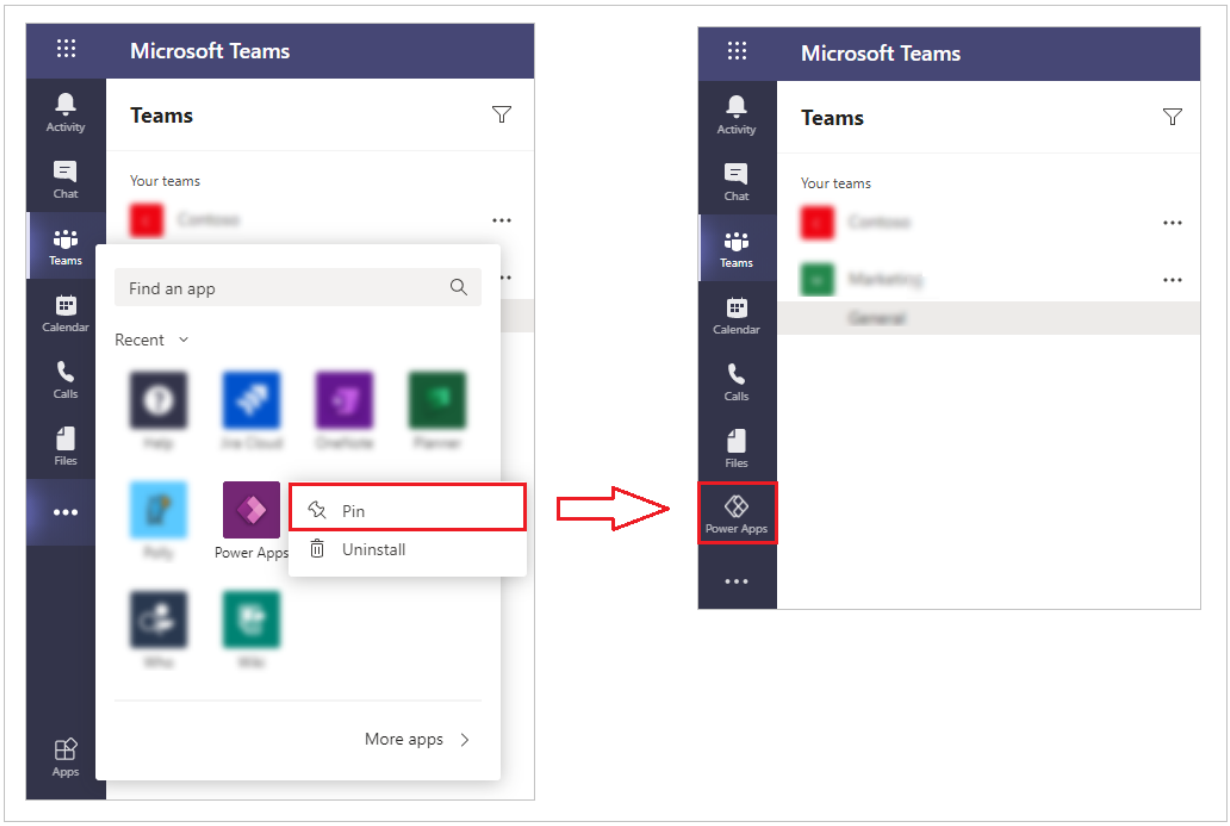
Power Apps is now added to Teams. You can access the app in Teams from the left pane.



> **Tip**
>
> As an alternative to using the flyout menu, you can select **Apps** from the lower-left corner in Teams, and then search for **Power Apps** to install the app.

Pin Power Apps to Teams

To easily access Power Apps app, **right click** the Power Apps app from ..., and then select **Pin**.

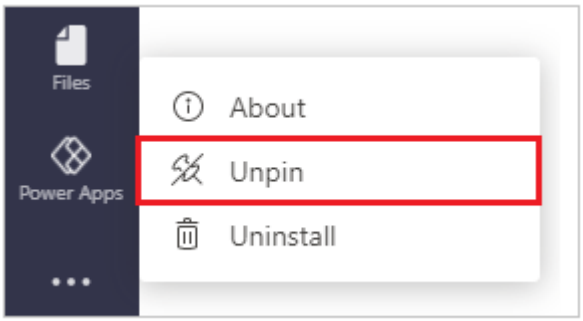


This action adds Power Apps to the Teams left rail for quick access.

When you pin Power Apps app, the text of the app icon in the Teams rail changes from italicized to normal to reflect the pinned app.

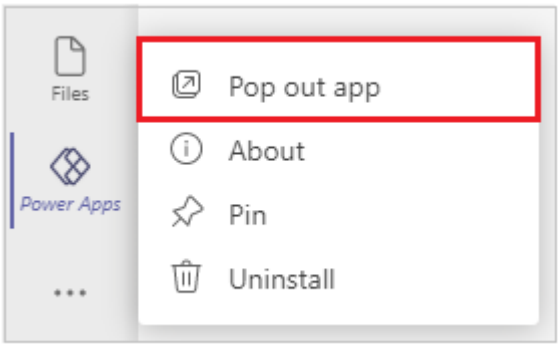


You can unpin Power Apps app anytime with a **right click** on the Power Apps app from the left rail, and then by selecting **Unpin**.



Pop out app

If you're using Microsoft Teams desktop app, you can use the **Pop up out** option to open Power Apps in a separate window for ease of use.



This way, you can continue to use Microsoft Teams for day-to-day interactions and still be able to interact with Power Apps separately.

Next steps

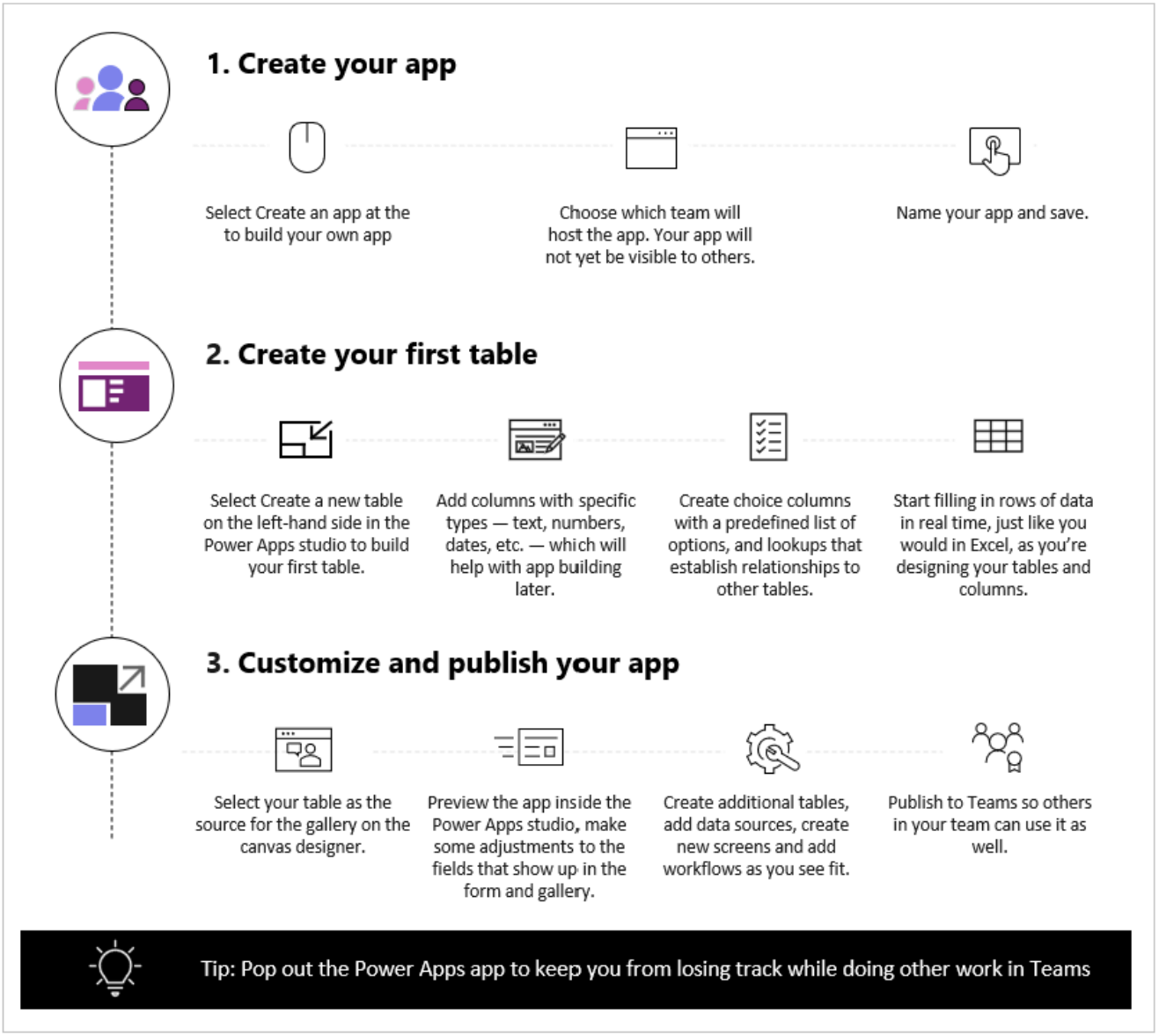
Create your first app in Teams
Overview of the Power Apps app

Create your first app in Microsoft Teams

You can create, edit, and delete canvas apps in Teams. If you're creating an app by using Power Apps in a Teams team for the first time, you also create a new Dataverse for Teams environment. For more information, see About the Dataverse for Teams environment.

Creating an app in Microsoft Teams is a quick and easy three-step process.

1. Create your app
2. Create your first table
3. Customize and publish your app



Watch this video to learn how to create your first app.

https://learn-video.azurefd.net/vod/player?id=3e63d41c-d4eb-411e-b99a-7cc87bb2992d&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fcreate-first-app

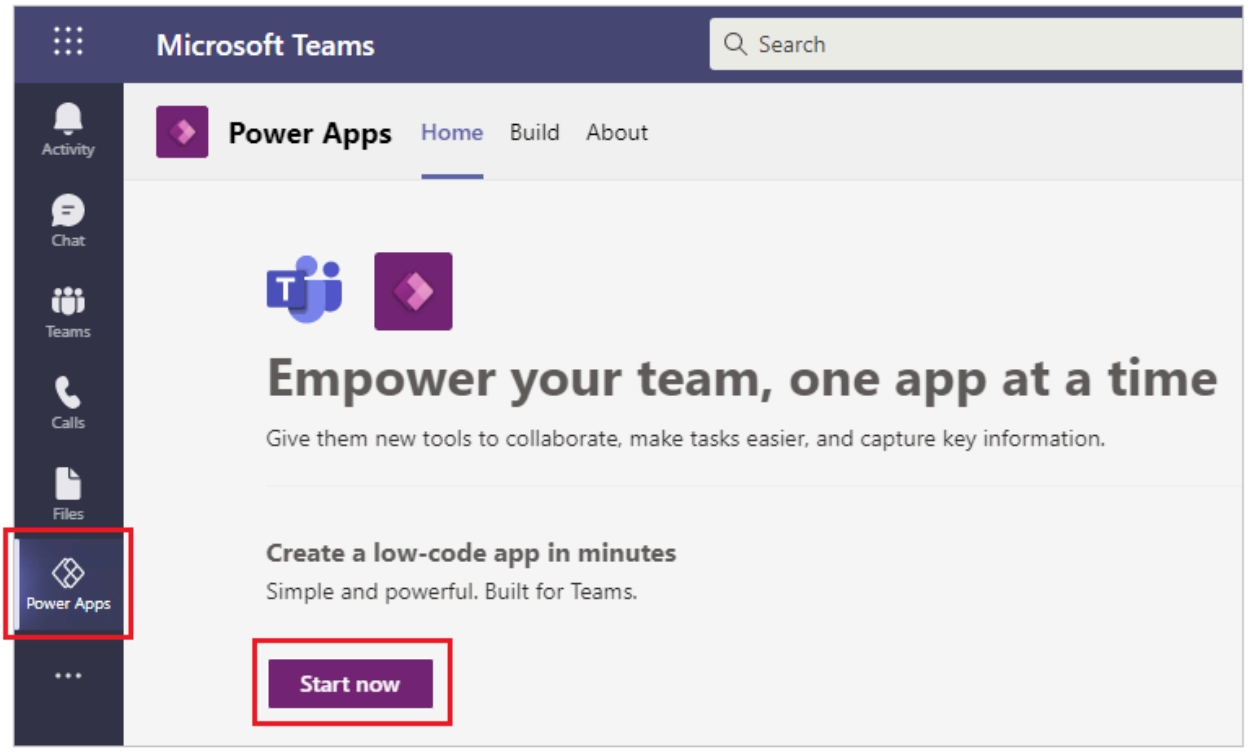
Ensure you refer to the latest steps when creating the table.

Now let's go through each step in detail.

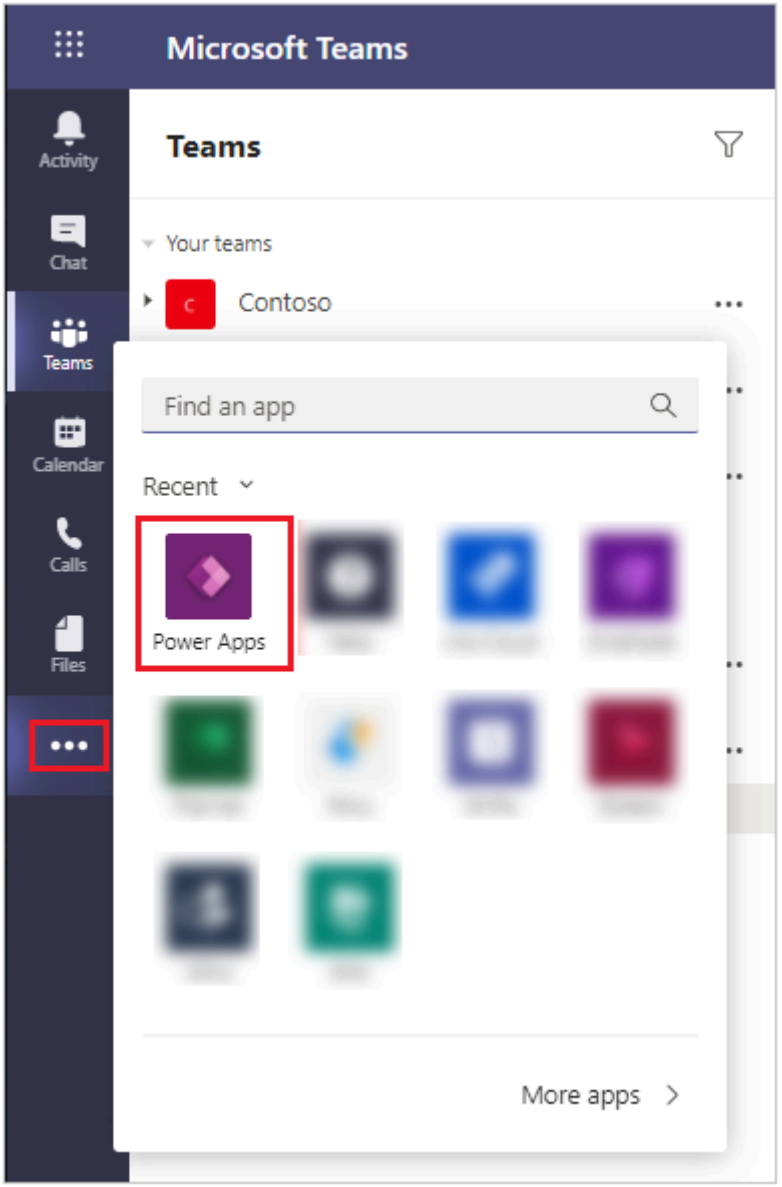
Create your app

After installing the Power Apps app template:

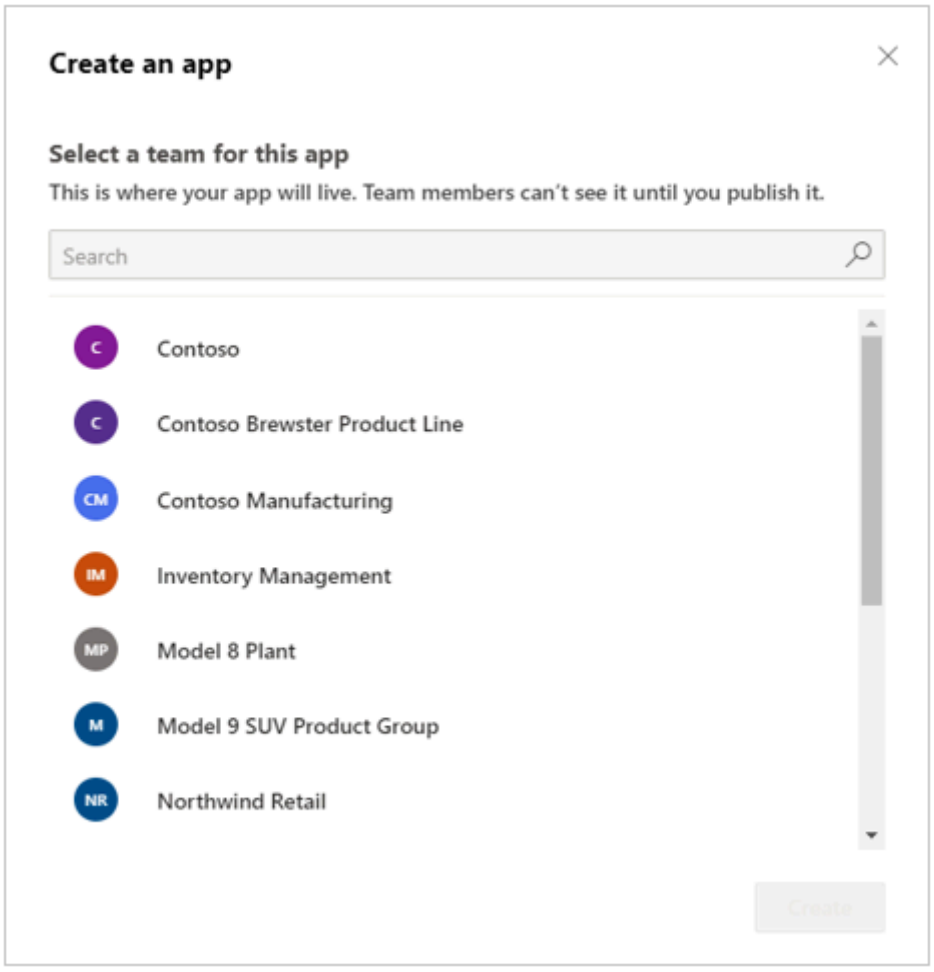
1. Start Teams, and sign in.
2. In the left pane, select **Power Apps**, and then select **Start now**.



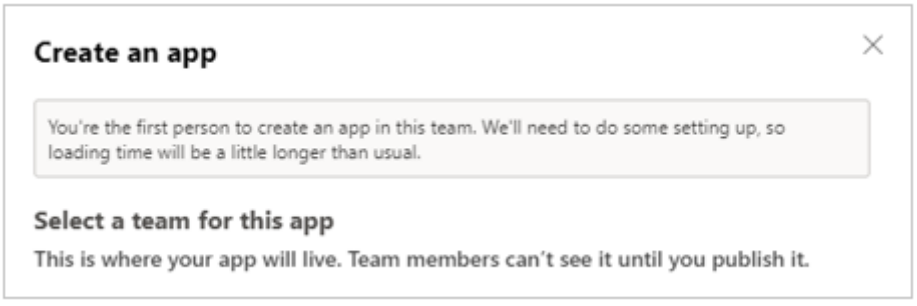
> **Tip**
>
> If you don't see the **Power Apps** app in the left pane, select ... (*More added apps*), and then select **Power Apps**.



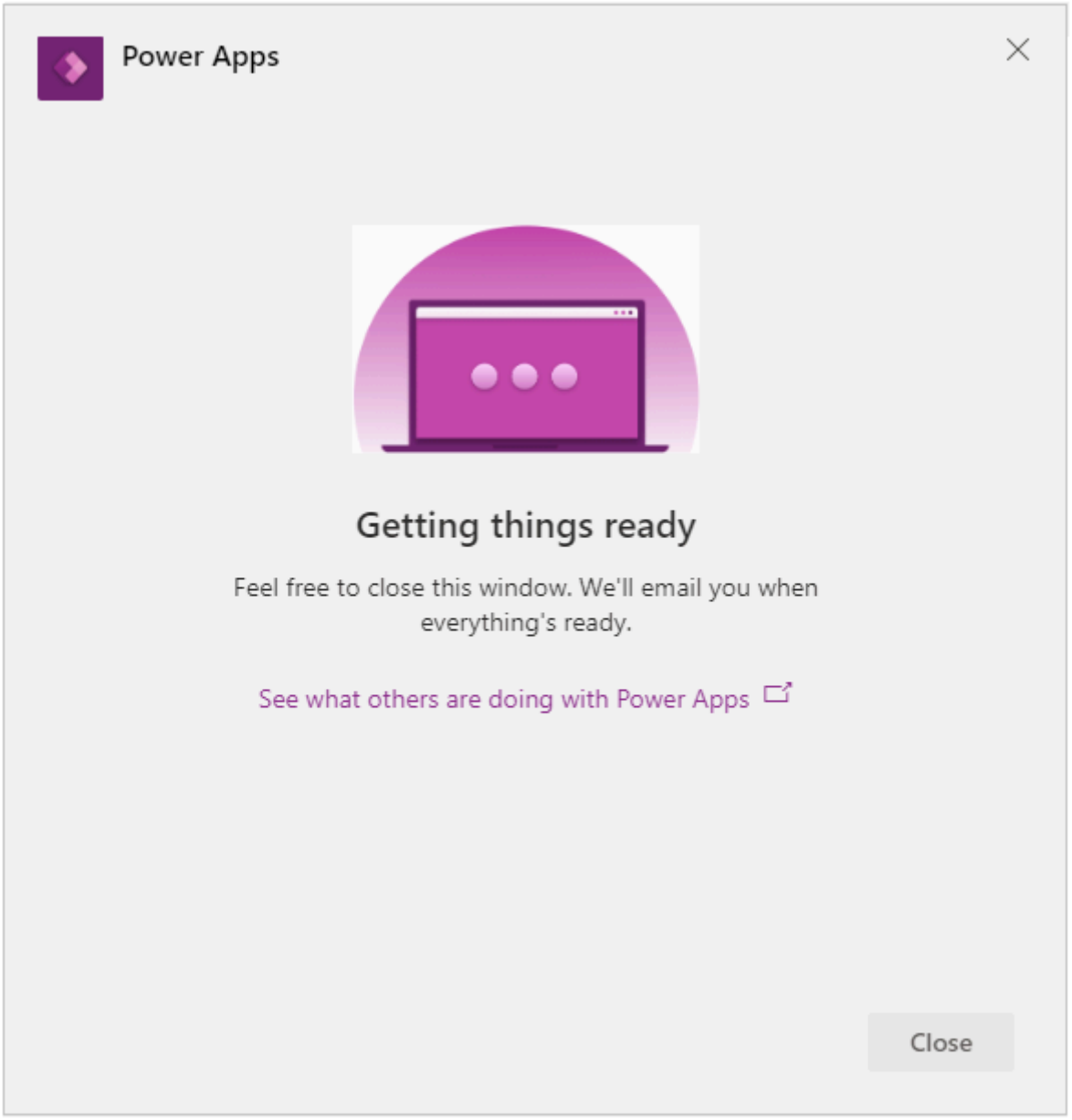
3. Select a team where you want to create the app. This is where the app lives. Team members can't see the app until you publish the app. You can also share the app with colleagues within your company that are not part of the team. For more information, see Set permission and share apps with colleagues.



If this is the first app you're creating in the selected team, a new Dataverse for Teams environment is created for the selected team. You see a message about this when you select the team. For more information, see About the Dataverse for Teams environment.

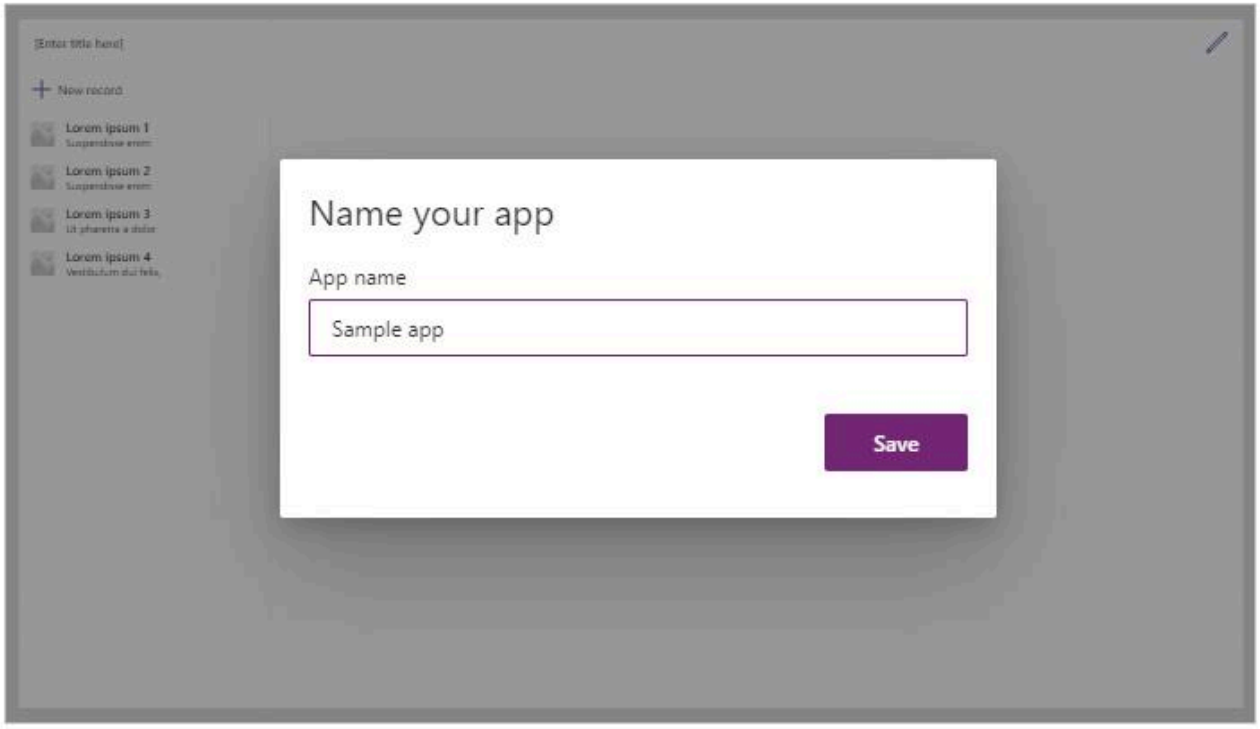


4. Select **Create**. You see a confirmation about the Dataverse for Teams environment being created, which might take some time. You're notified when the environment is available for you to create your first app by using an email and a Teams notification.



> **Note**
>
> Environment creation fails if the Teams team that you selected has *Hiddenmembership* enabled. If this happens, try creating the app in a different team. For more information, see Hidden membership groups.

5. Enter an app name, and then select **Save**.



> **Tip**
>
> An app template is included with the app by default, so that you can get started with creating your table and using the app. For more information, see Understand Power Apps Studio.

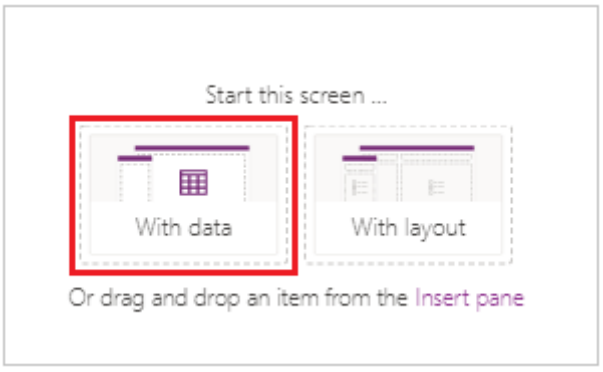
Create your first table

Data for apps built in Teams lives in Dataverse for Teams environments as tables. Like other data experiences, you can add extra columns to track different attributes of each record before filling the table with multiple rows of data.

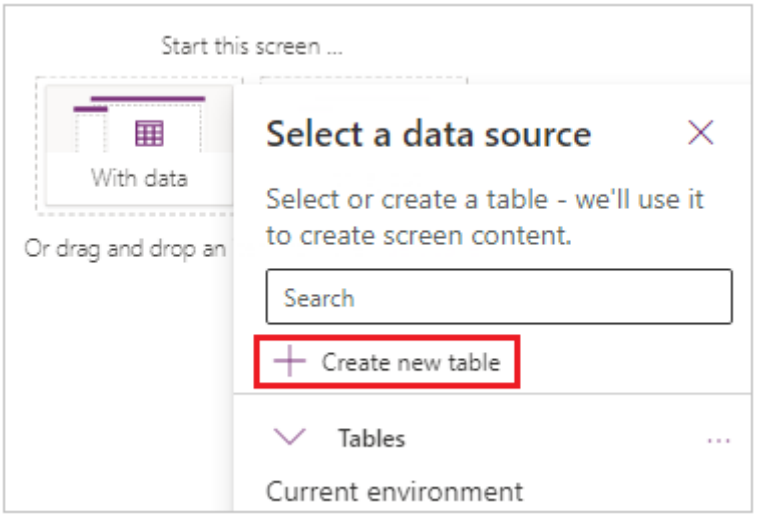
> **Tip**
>
> For more information about Dataverse for Teams tables, see Overview of Dataverse for Teams.

To create your first table:

1. Select **With data**.



2. Select **Create new table** > **Start with blank table**.



3. In the **Create a table** dialog box, select **Edit table properties** and type a meaningful name for your new table that describes this dataset, and then select **Save**.

4. After the table is created, add columns of different types, add rows, and work with data in the visual editor.



Add columns to the table

To add a new column to track new data:

1. Select **New column.**
2. Enter a name that best describes your new column.
3. Select the type for your column.

> **Tip**
>
> For more information about column types supported by the visual editor, go to Supported column types. To add columns that aren't supported by the visual editor, create a table by using the solution explorer. More information: Create tables

4. If required, change **Advanced options** for the selected column type.
5. Select **Save**.
6. Repeat the previous steps to add the rest of the data columns.

Add rows to the table

Select **Add row** to add rows.

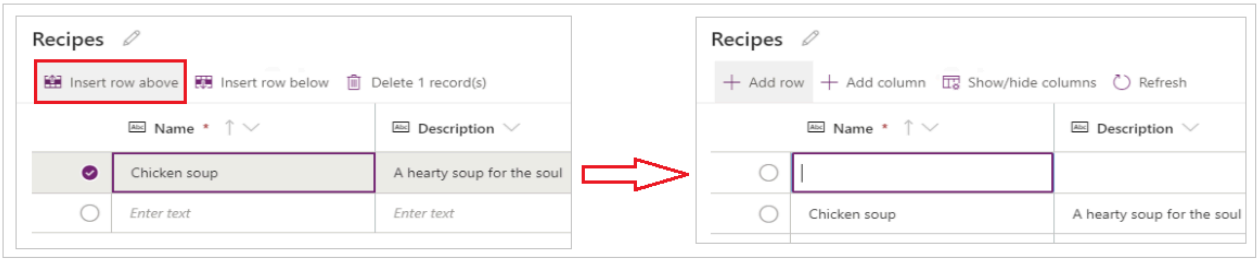
Paste the copied data into the table

You can paste up to **50 rows** of data into a table. To create new rows from the copied data, add empty rows in the table first. Adding new empty rows ensures pasting data from clipboard doesn't accidentally overwrite your existing data.

> **⊗ Caution**
>
> Paste operations are automatically saved, and there's no undo option. Use caution when pasting data over existing rows since this action overwrites your existing data in the table.

To paste the data from clipboard into the table:

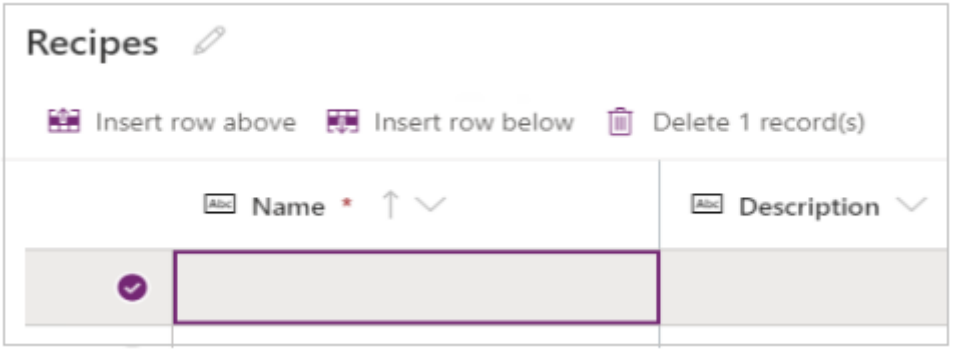
1. Select the record that you want to add new records above or below.
2. Select **Insert row above**, or **Insert row below** to insert rows before or after existing rows. For example, inserting row above the selected row:



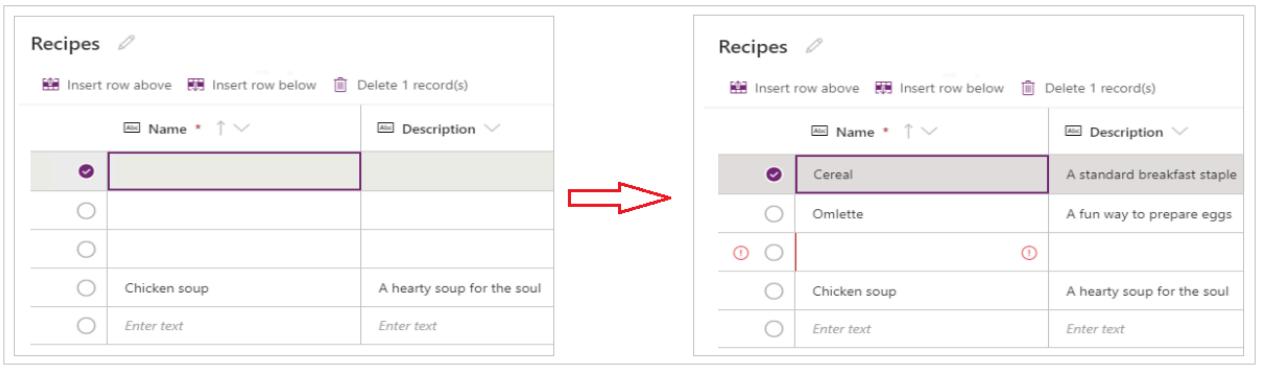
3. Repeat the preceding step to add more empty rows, as needed.

> **Tip**
>
> If you're unsure of how many rows you need, create additional empty rows before pasting data. You can delete extra empty rows later.

4. Select the top-left cell from the newly created rows.



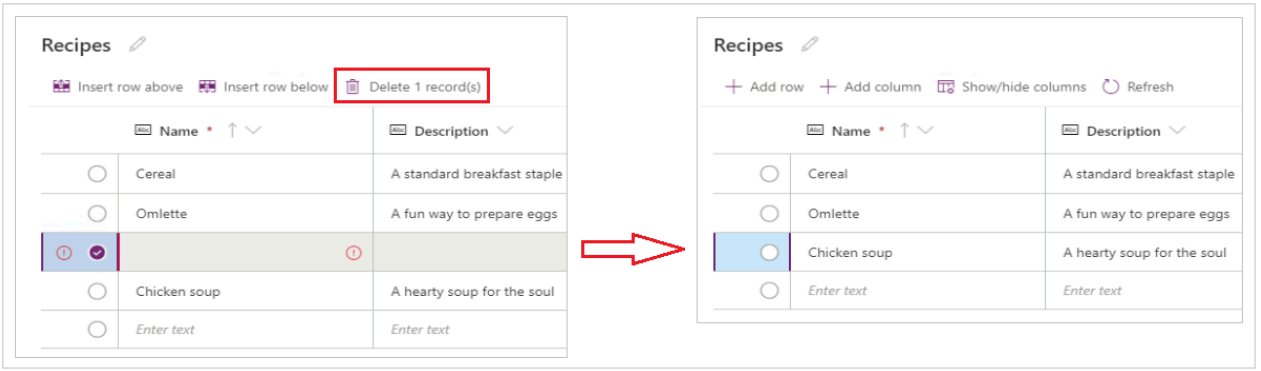
5. Paste the copied data.



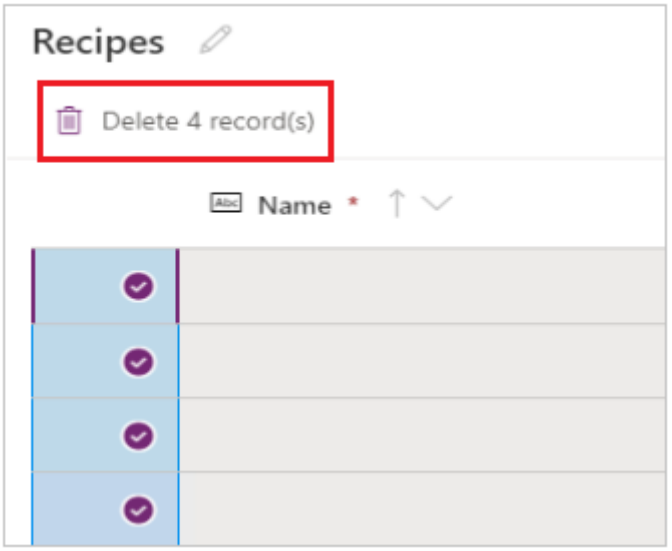
> **Note**
>
> If you paste data when the cell is in edit mode (if you *double-clicked* top-left cell), the data is pasted inside the cell. If this problem happens, delete the incorrectly pasted data, select any other cell, select the top-left cell, and try to paste again.



6. Remove any extra empty rows.



> **Tip**
>
> To delete multiple rows that are together, select the top row, and press the **Shift** key on the keyboard. Keep the key pressed, select the last row in the chunk of the rows that you want to delete. Then, select **Delete X record(s)**.

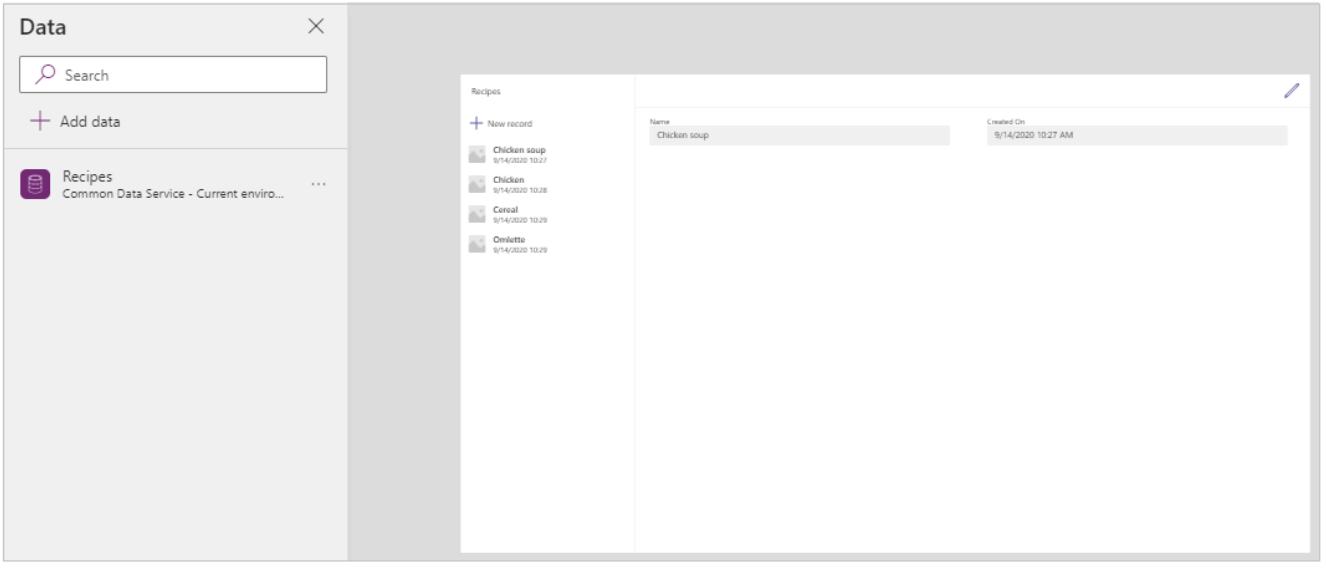


Changes to the table are automatically saved. However, check for errors, and fix them before you close the visual editor.

Customize and publish your app

When you close the table, the app template that you added to the app screen by default automatically refreshes to use the new table.

> **Note**
>
> To add new data sources, such as a new table, or to reopen the app, see Manually connect data to the app for steps to manually refresh the app with the new data.



Depending on the number of fields you add to your table, you might need to update the form and the gallery in the app template.

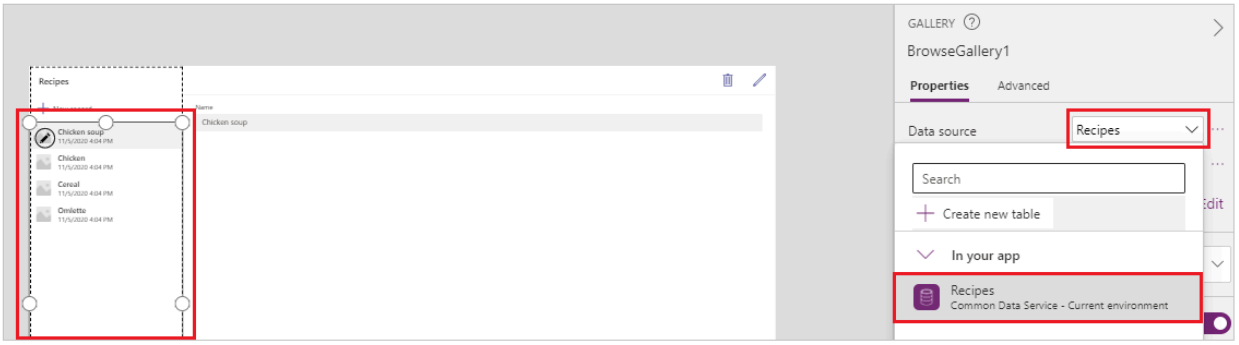
Manually connect data to the app

The gallery and the edit form components on the screen automatically refresh with the data only for the first data source. If you add extra data sources, such as a new table, or if you reopen the app, you need to manually connect the data to the template gallery and form.

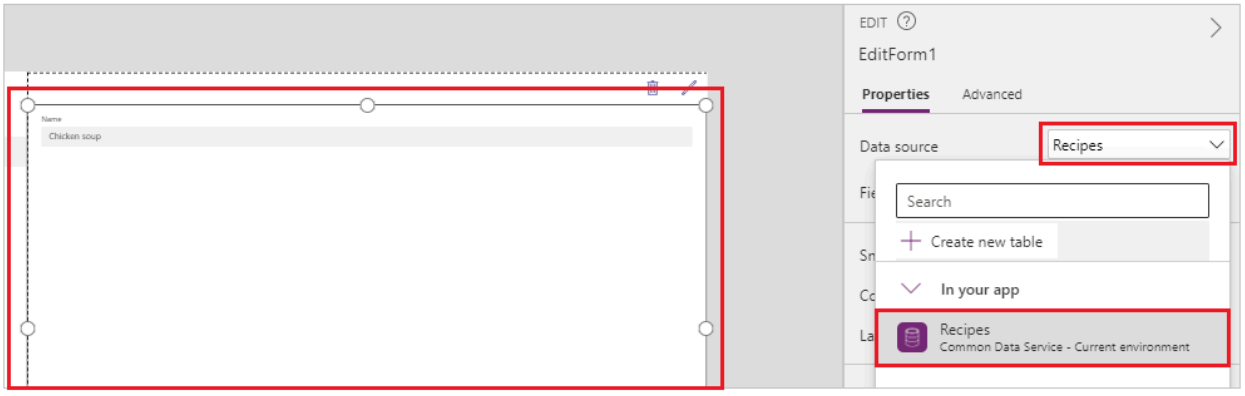
> **Tip**
>
> Skip these steps if the app screen is already connected to the table that you created.

To manually connect to the new data:

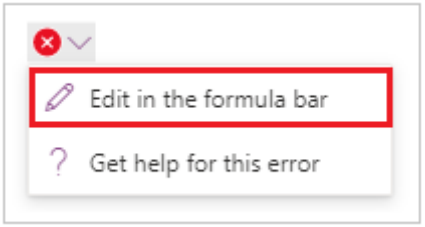
1. Select the gallery, and then select the correct data source.



2. Select the form, and then select the correct data source.



3. If you see any formula error, select **Edit in the formula bar**, and update the formula with the data source name.



For example, to change from using *Instructions* as the data source name for the edit form to *Recipes*, replace the name of the data source.

Change from:

```
Power Fx
Remove(Instructions, selectedRecord); If(IsEmpty(Errors(Instructions,
selectedRecord)),UpdateContext( {itemSelected:false, editMode:false,
newMode:false, deleteMode:false}); Set(CurrentItem,First(Instructions)););
```

Change to:

```
Power Fx
Remove(Recipes, selectedRecord); If(IsEmpty(Errors(Recipes,
selectedRecord)),UpdateContext( {itemSelected:false, editMode:false,
newMode:false, deleteMode:false}); Set(CurrentItem,First(Recipes)););
```

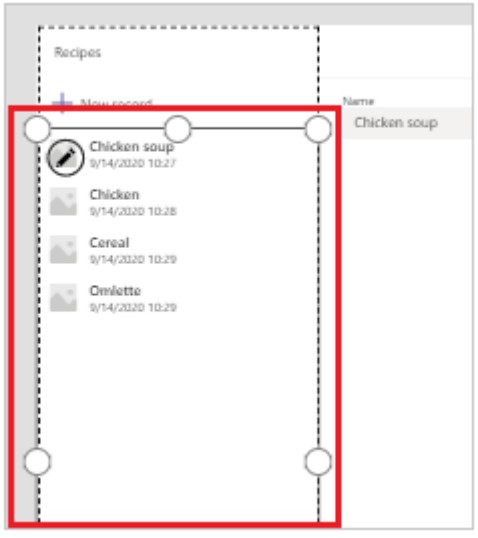
4. Repeat the earlier step to fix any extra formula errors.

Update background color for the selected gallery record

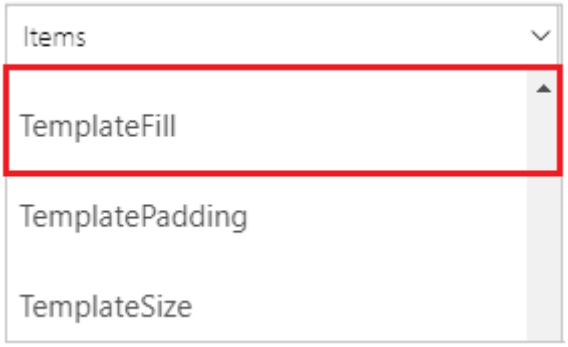
The gallery shows records you created earlier. However, you need to change the background color for a record when you select it from the gallery to show the selection.

To update the selected item background color:

1. Select the gallery.



2. Select *TemplateFill* property from the property drop-down on the top-left.



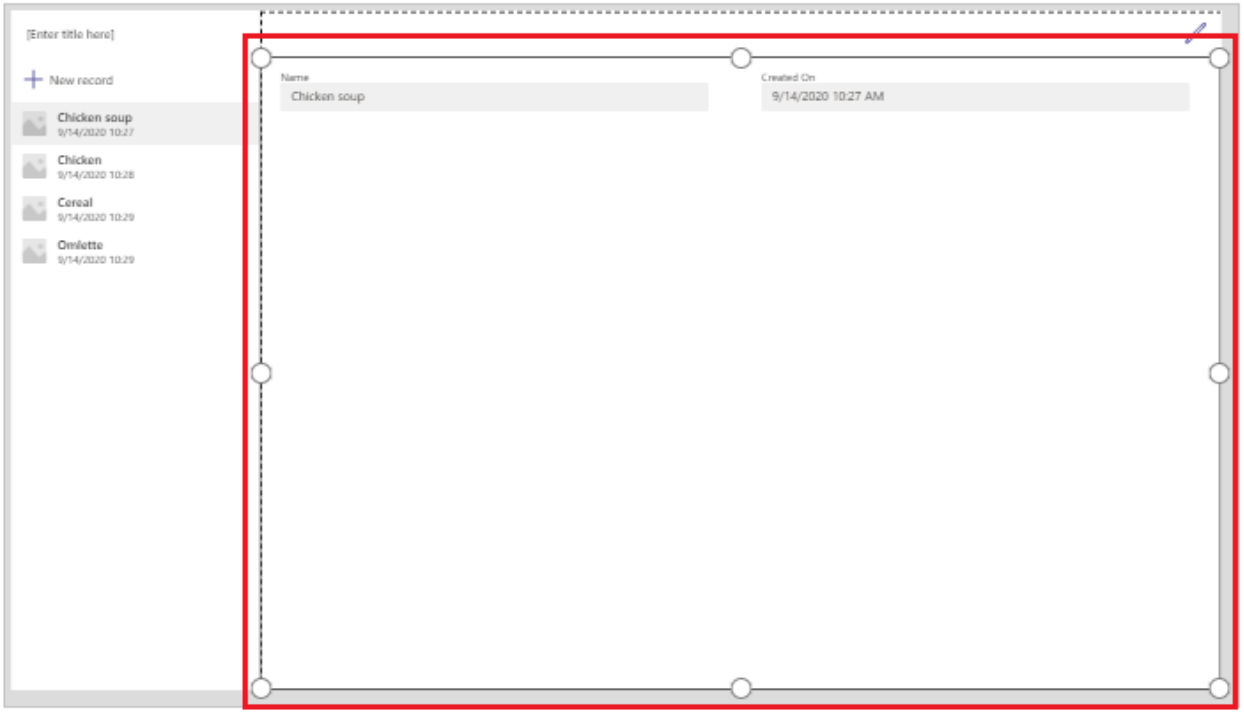
3. Update the *TemplateFill* property value in the formula bar to the following formula:

```
Power Fx
If(ThisItem.IsSelected, RGBA(0,0,0,.05), RGBA(0,0,0,0))
```

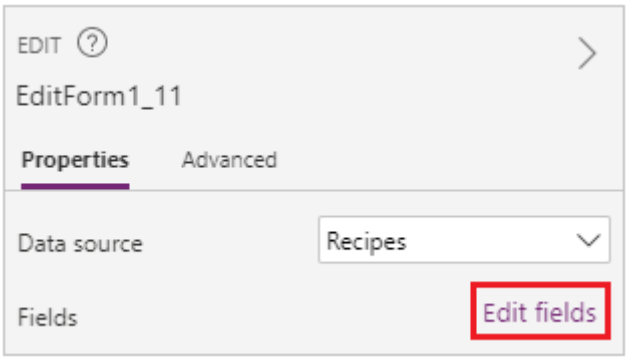
Update fields on the form

The template shows two fields inside the form by default. To add more fields to the form:

1. Select the form.



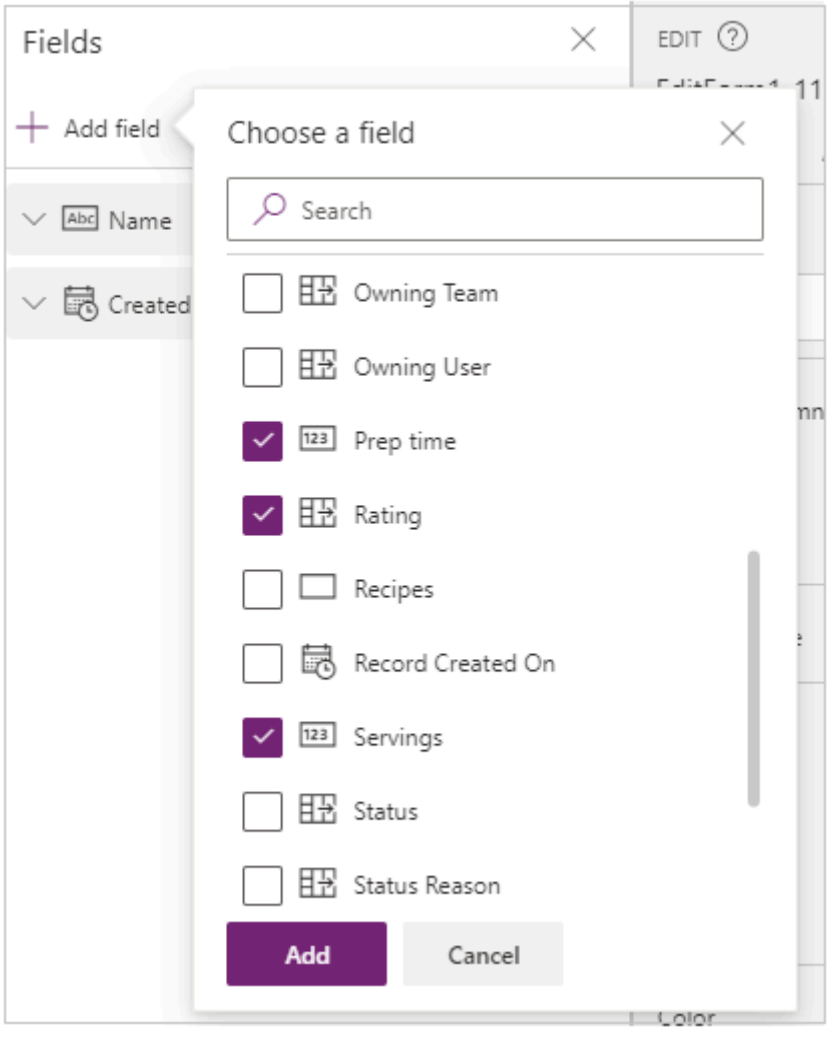
2. Select **Edit fields** from the property pane on the right.



3. Select **Add fields**.

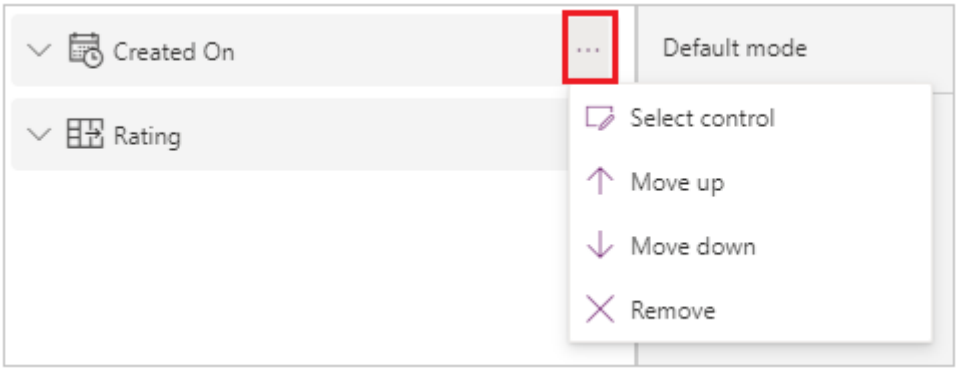


4. Select the fields that you want to add on the form.



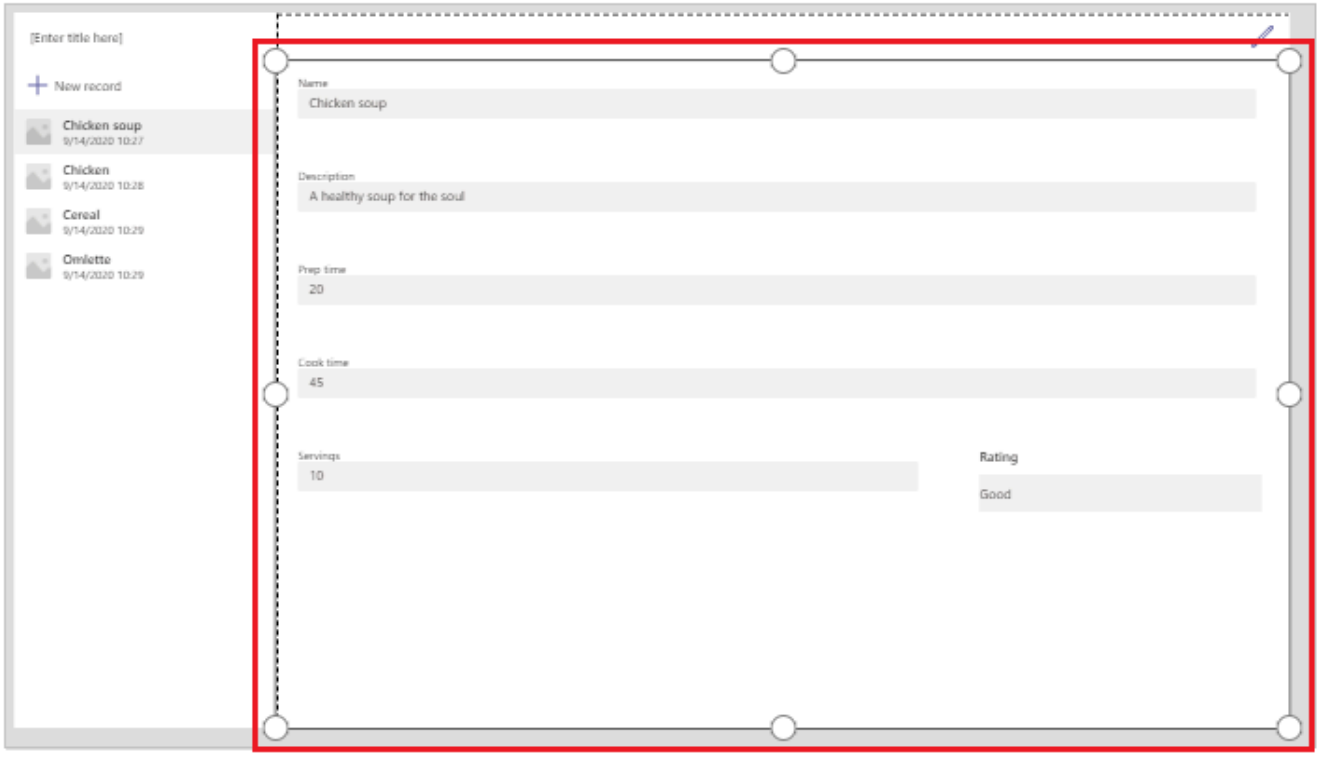
5. Select **Add**.

6. (Optional) Select fields to move up or down, or remove unwanted fields by selecting **Remove**.



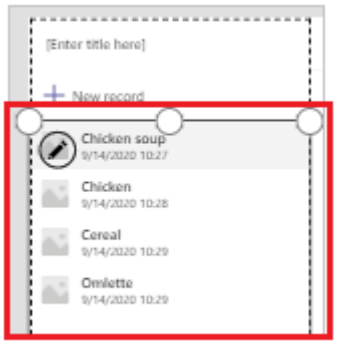
7. Close the fields pane.

When you add, remove, or reorder fields, the form updates automatically.



Update fields in the gallery

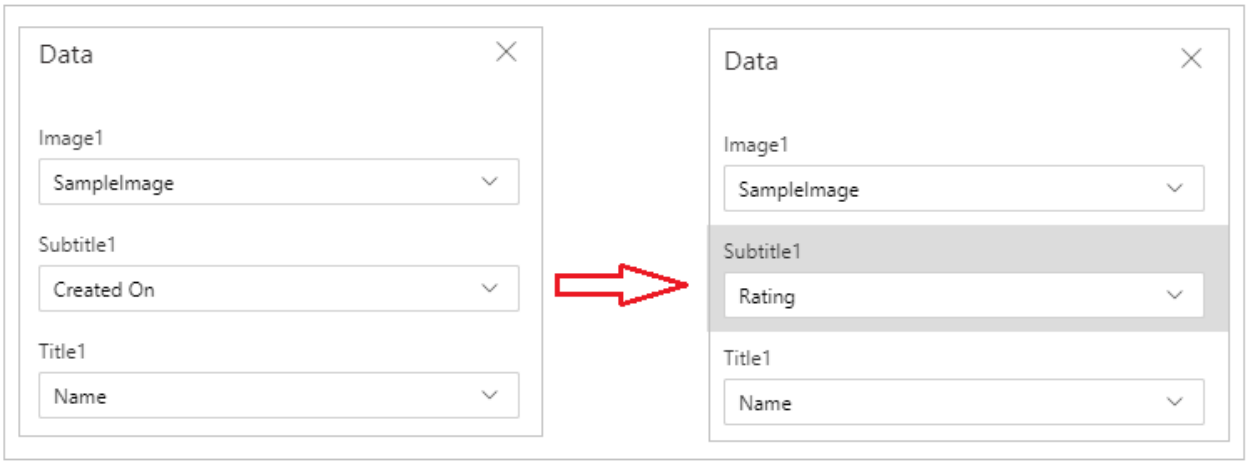
You might need to update the gallery on the default app template to reflect the fields from the table you created.



To update the fields in the gallery:

1. Select **Edit fields** from the property pane on the right.

2. Select the drop-down menu for a gallery component (such as image, title, or subtitle) and change the selection to the field you want.



3. Close the data pane.

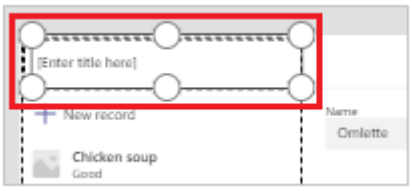
The gallery updates with the selected changes.



Update the app title

To update the app title:

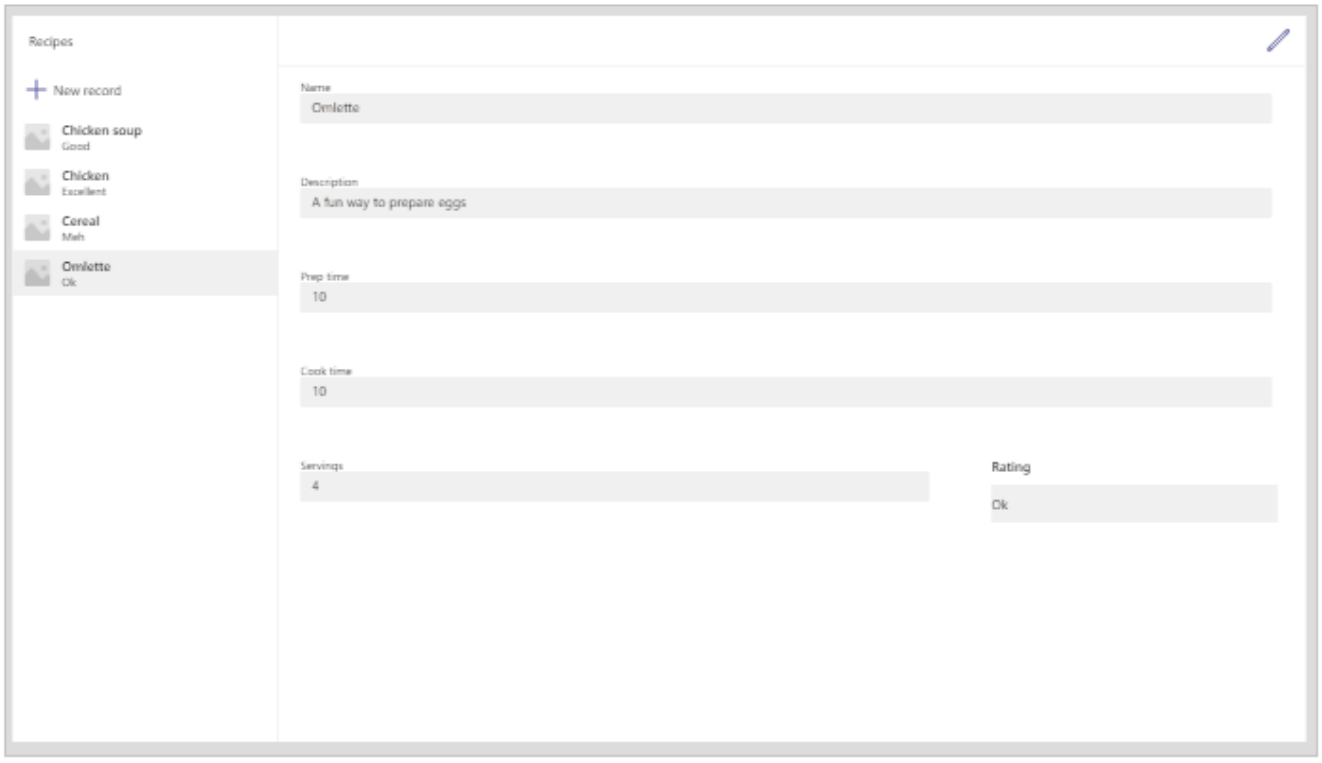
1. Select the label from the upper-left corner of the screen.



2. Change the *Text* property value from the properties pane on the right.

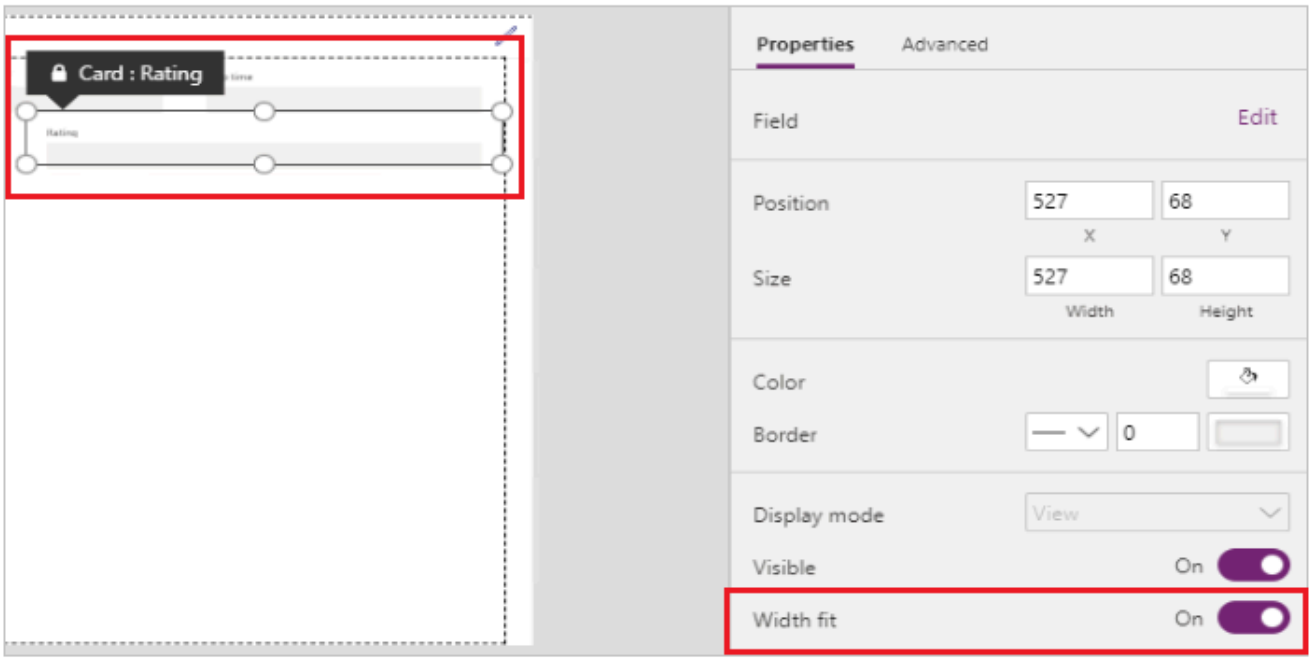


Verify all the changes you made to the app.



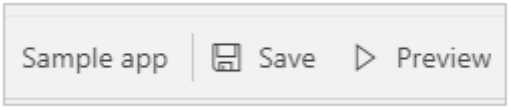
Tip

For better responsiveness, the data cards by default have the Width fit property set to *On*. If you want to resize the data cards manually, select the data cards and then set the **Width fit** property to *Off*.



Save and preview the app

Now that you've created your first app with data, you can save your changes and preview the app.



1. Select **Save**.
2. Select **Preview**.
    - To add a new record, select **New record** and fill in the form.
    - To edit a record, select the record from the left pane, and edit the values.
    - To close the preview, select **Close** or the **Esc** key.

Publish and add app to Teams channel

Select **Publish to Teams** to publish the app so others in your team can use the app, and add the app to Teams channel. More information: Publish and add an app to Teams

Now that you've created your first app from a blank table, let's go through the Power Apps interface, Studio, controls and how to create additional apps in detail.

Additional considerations for Microsoft 365 Group

When you select **Create** to create an app in the selected team, Power Apps automatically enables the selected team's Microsoft 365 Group for security. However, if the group can't be enabled for security, you see the following error:

`The request failed with error: 'Insufficient privileges to complete the operation.' The tracking Id is '[xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx]'`.

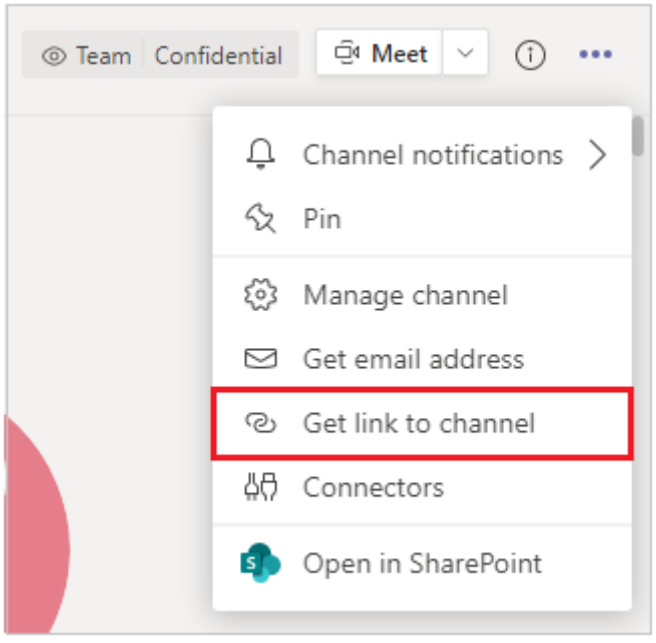
To fix this issue, manually update the Microsoft 365 Group setting *SecurityEnabled* from *False* to *True* by using PowerShell. For detailed steps, see Share an app with Microsoft 365 Groups. For more information, see Microsoft 365 Groups and Microsoft Teams.

Enabling security for teams with same names

Team names aren't always unique. It's possible for more than one team in Teams to share the same display name. For example, there might be more than one team with the name "Inspection" in a tenant. However, the Microsoft 365 Group associated with each team has a different *ObjectId* in Azure Active Directory (Azure AD), even when the team names are the same. The *ObjectId* in Azure AD for a Microsoft 365 Group is the same as the *GroupId* in Teams for the associated team.

To find the **GroupId** (ObjectID in Azure AD) of a team:

1. Go to the team in Teams.
2. Select any channel in the team.
3. Select **More options** (...) from the upper-right corner, and then select **Get link to channel**.



4. Copy the *GroupID* from the link.



You can also copy the channel link by selecting **Copy**, and then pasting it into a text editor to get the *GroupID* value.

```
https://teams.microsoft.com/l/channel/19%3a06f038959c9f4387a35f5d776484cff2%40thread.tacv2/General?groupId=40b7e373-a788-4e29-a590-5a86db7550e3&tenantId=00000000-0000-0000-0000-000000000000
```

5. Now that you have the correct team's **GroupID** (ObjectID in Azure AD), use the instructions in Share an app with Microsoft 365 Groups to enable security for the team's Microsoft 365 Group.

Next steps

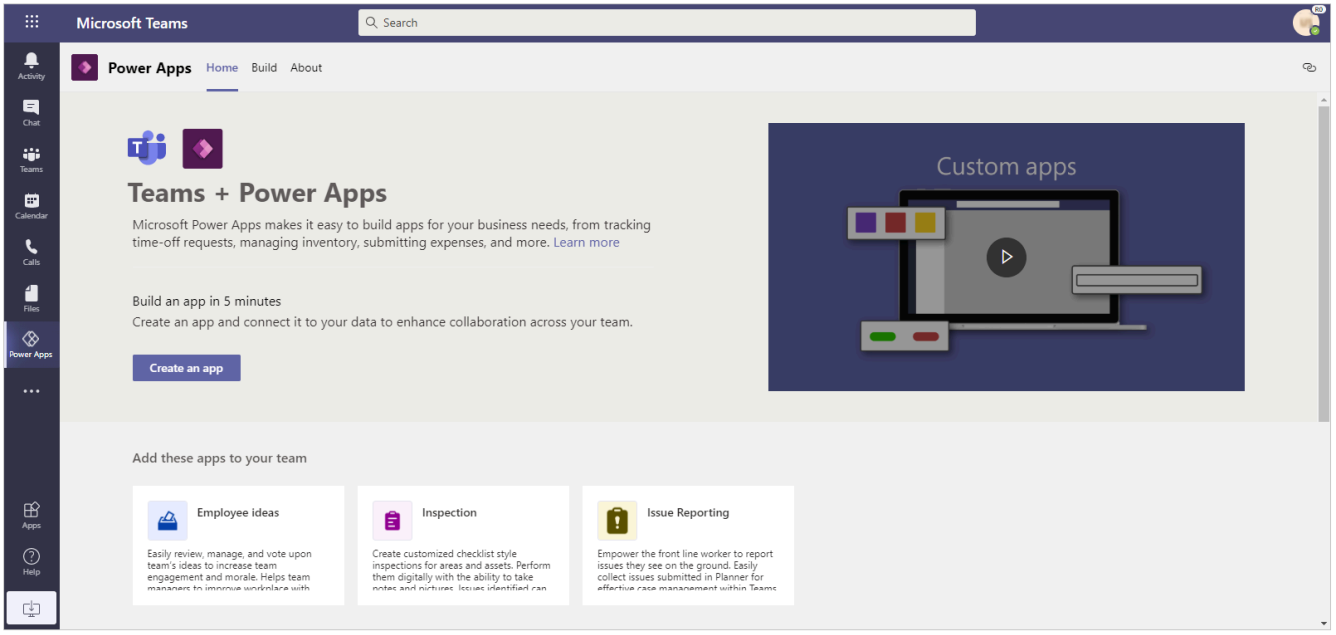
Overview of the Power Apps app
Understand Power Apps Studio
Use the Fluent UI controls
Create additional apps

Last updated on 01/14/2026

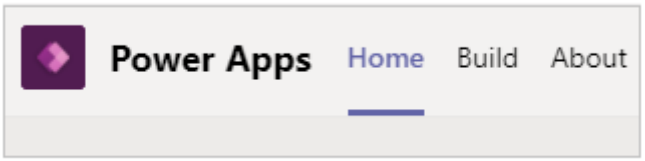
Overview of the Power Apps app

Use the Power Apps app to create and manage apps across the available environments for your teams. Select the Power Apps app from the left pane in Microsoft Teams to open the Power Apps interface.

> **Note**
>
> If **Power Apps** doesn't appear on the left pane, you can install it. After you install the app, you need to create your first app to create an environment. For more information, see About Teams environments.



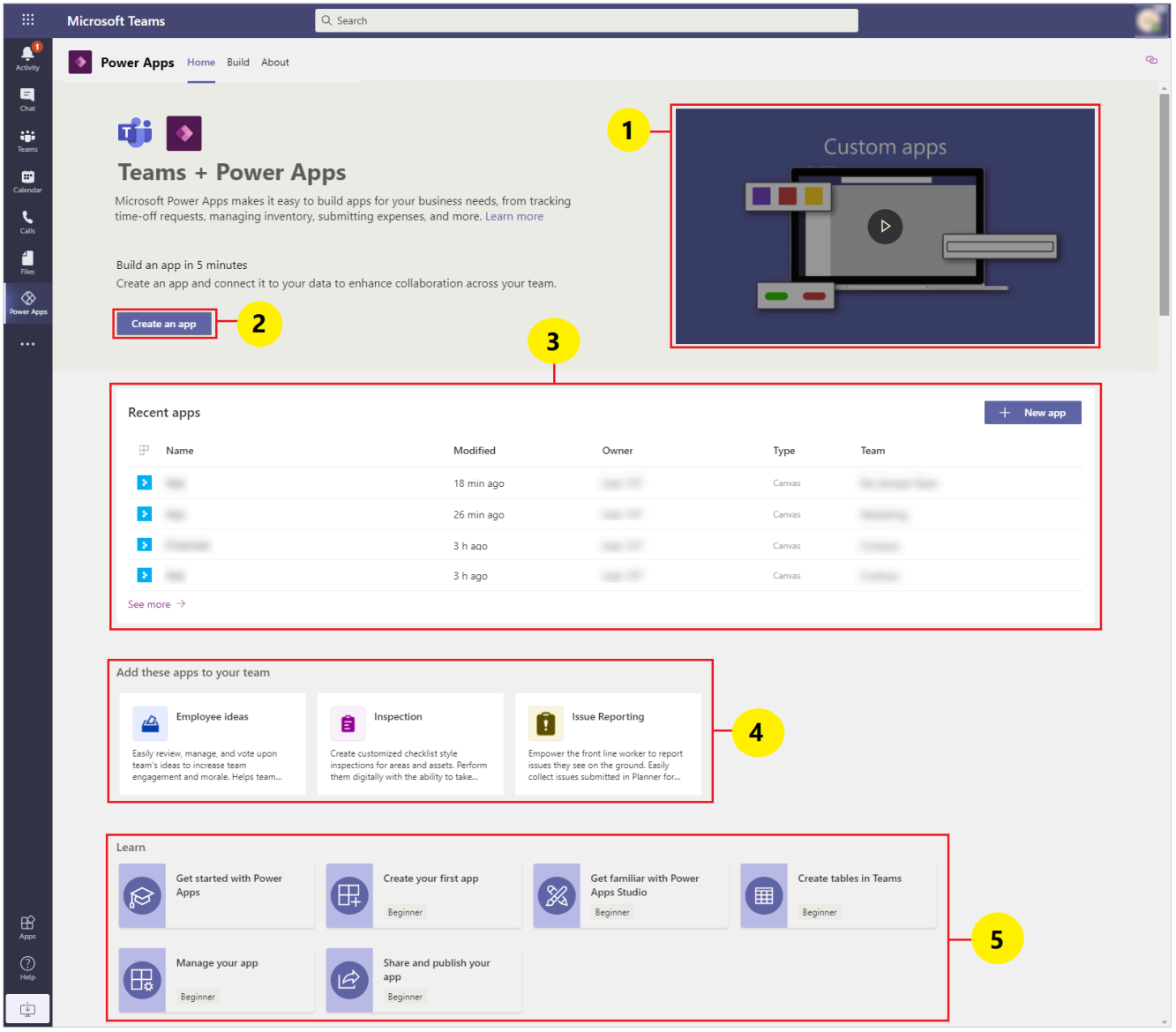
Three tabs are available: **Home**, **Build**, and **About**. You can start creating a new app by selecting **Create an app** on the **Home** tab.



Let's understand each tab in detail.

Home

Your **Power Apps** experience in Teams starts with the **Home** tab. Use the **Home** tab to get started creating new apps with Power Apps.



1. **Introduction Video**: Watch a one-minute video to find out what Power Apps in Teams is all about.

2. **Create an app**: Select **Create an app** to enter the Power Apps Studio integrated inside Teams and start building a Power Apps app. As you learned from Create your first app in Teams, a new environment is created for your team when you first create an app. Creating additional apps in the same team uses the environment that you already created for the team.

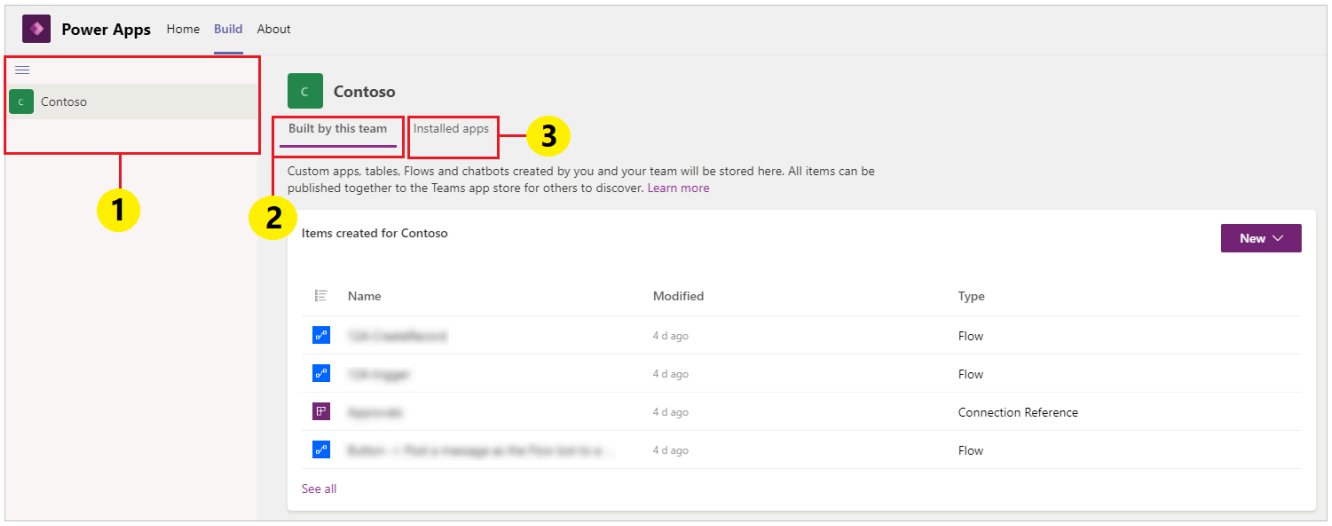
3. **Recent apps**: Find the apps that you recently edited in this section. Select an app from this section to directly open the app inside the Studio. This section is empty if you didn't edit any app yet.

4. **Sample app templates**: This section shows available sample app templates.

5. **Learn**: Find helpful articles in this section to get you started with Power Apps in Teams.

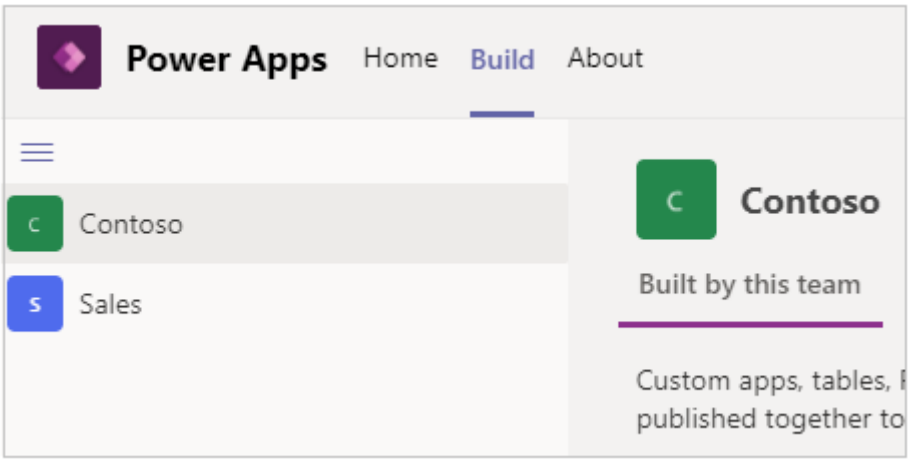
Build hub

Select the **Build** hub to see the list of environments for each team that you're a member of, and the list of apps or objects that the team created or installed.



1. **Environments**: Each Microsoft Teams team with an environment that you have access to is listed here.

   If you have more than one team with environments, select the team or environment you want from the left pane.



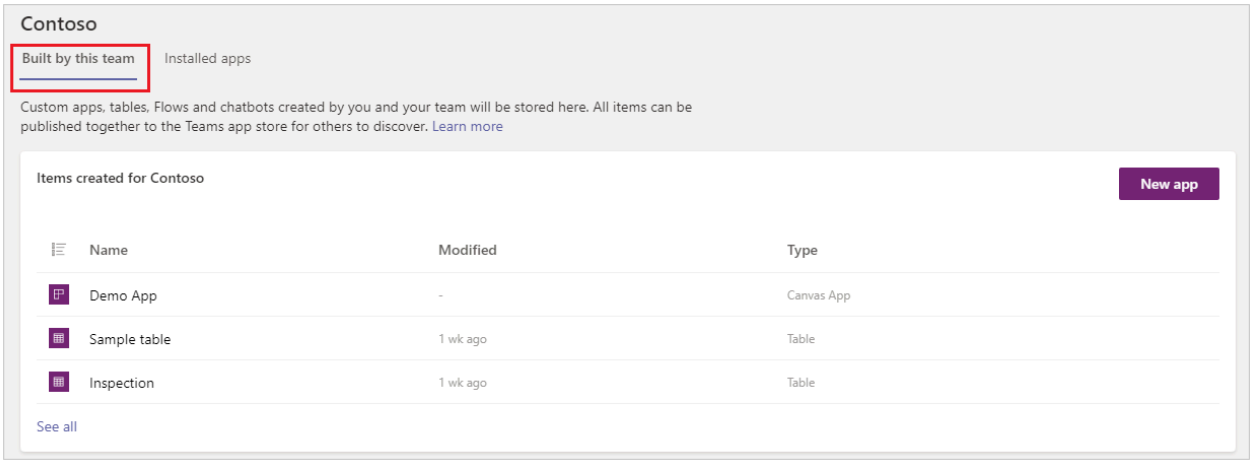
   Select  to collapse or expand the environment list.



When you select an environment from the list, you see the recent resources built by the selected environment's team, or the apps installed in the selected environment on the right side of the screen.

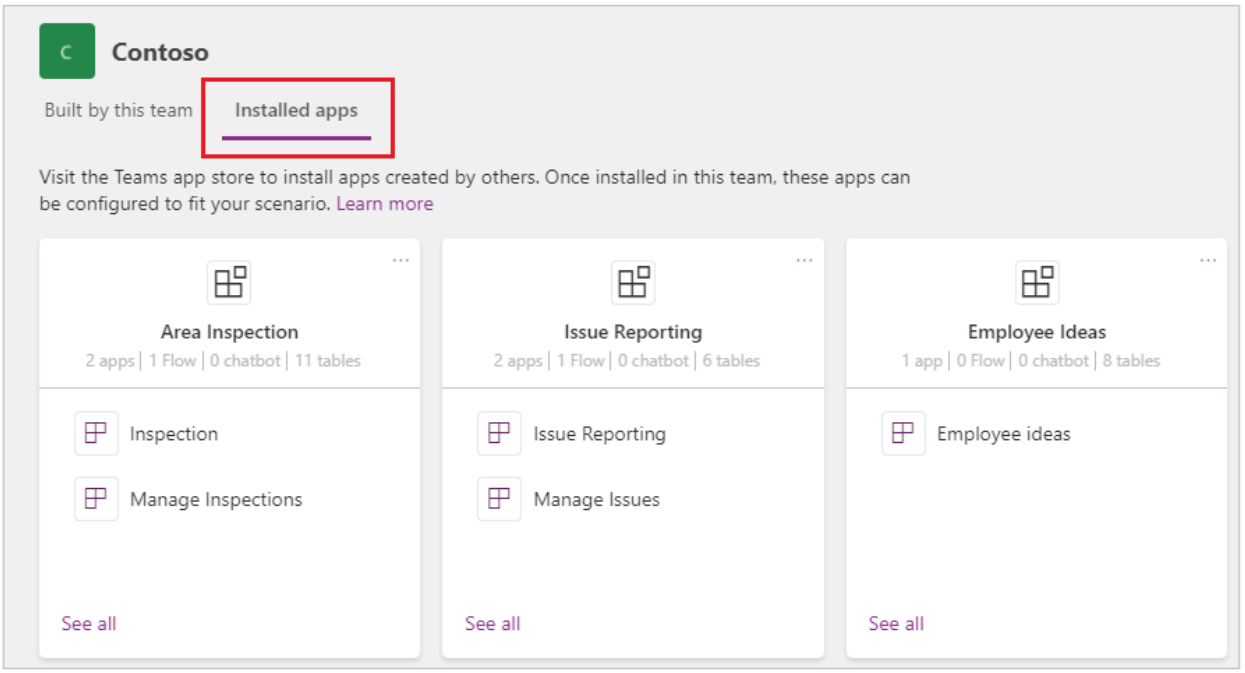
2. **Built by this team**: This list shows all Power Apps objects such as apps, flows, and tables created in the selected environment (the Teams team). Select **New app** to create a new app. Select **See all** to open the solution explorer for the environment. More information: Solution explorer

   Select **See all** to view all the components in the selected environment, and to create new apps or components.



   In the *Items created for* list, select an app, or a table to directly open the selection in the respective editor.

3. **Installed apps**: Shows a list of installed apps in the environment (Teams team) from the Teams app store.



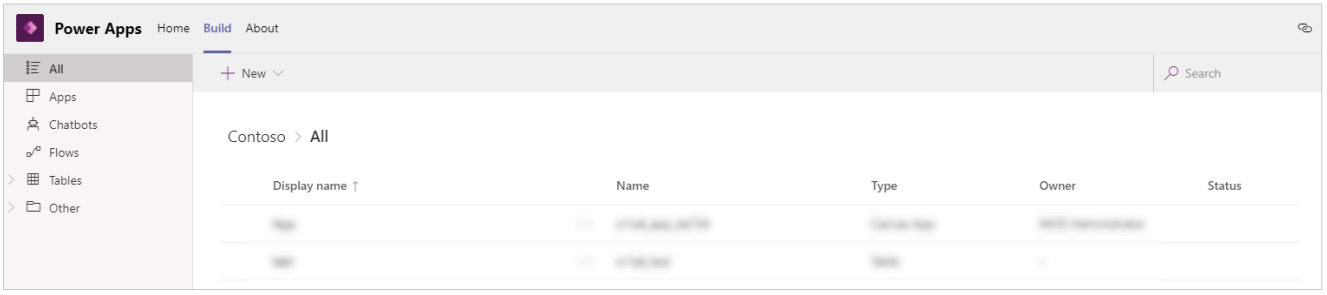
Select an app, or a table to directly open the selection in the respective editor.

Solution explorer

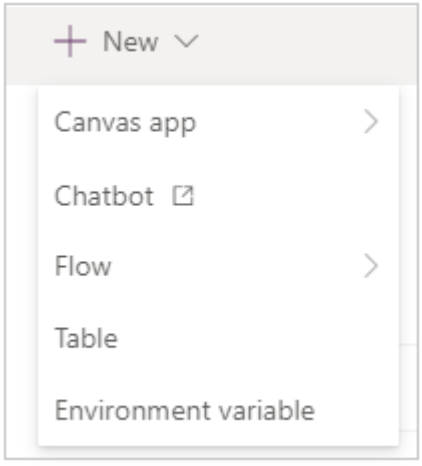
When you select **See all** from the environment list on the **Build** hub, you see the Power Apps solution explorer with a tree view of the pages for the environment. To get started, select a page from the solution explorer on the left.

All

Shows all available Power Apps objects such as apps, flows, and chatbots in the selected environment.

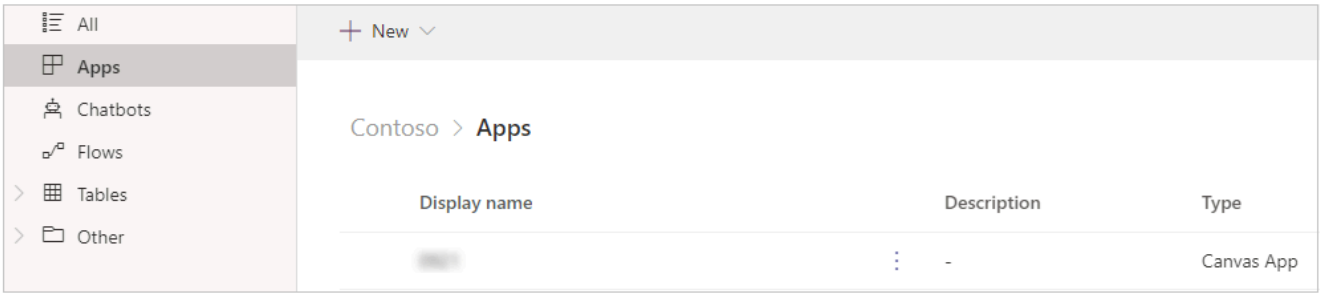


Select **New** to create new apps, flows, or tables.



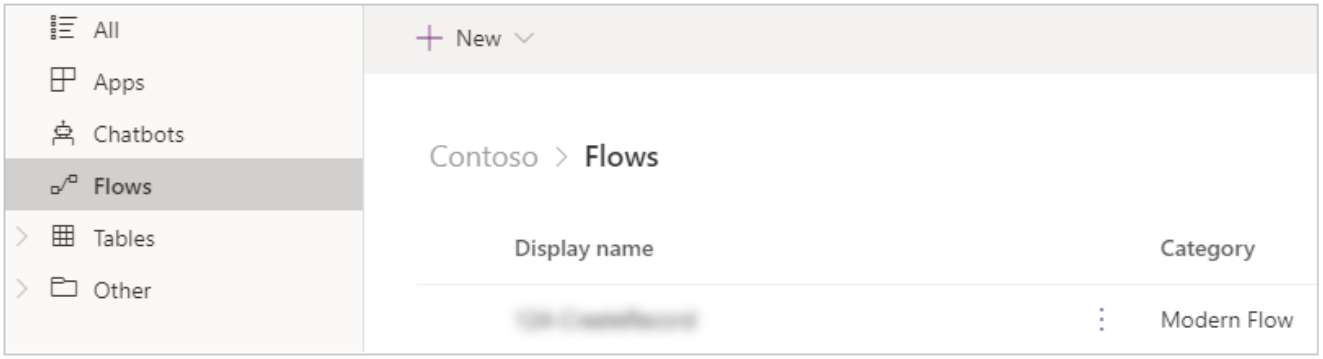
Apps

Shows the apps created with Power Apps. More information: Manage your apps



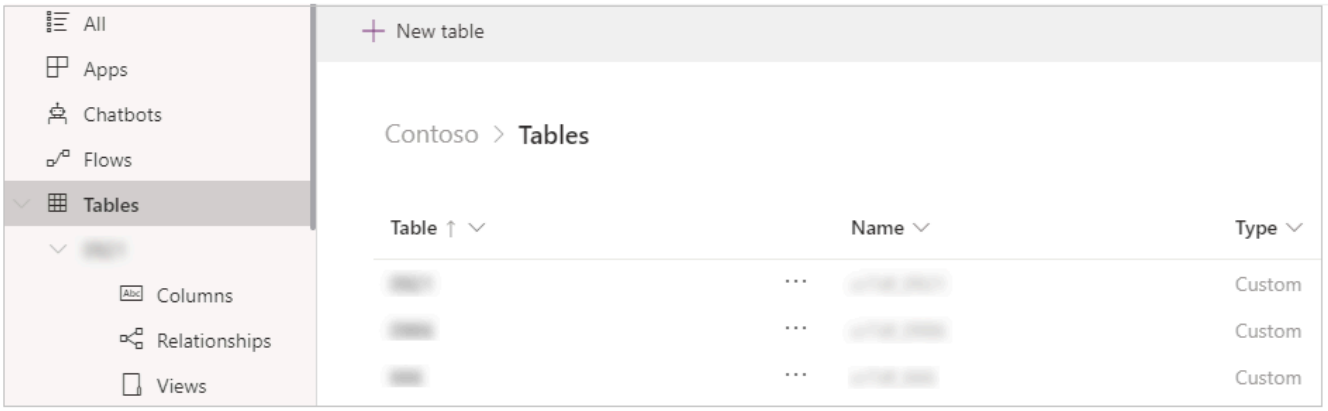
Flows

Shows all the Power Automate flows in the selected team's environment. Power Automate is a service that helps you create automated workflows between your favorite apps and services to synchronize files, get notifications, collect data, and more. Different types of flows are available. More information: Get started with Power Automate



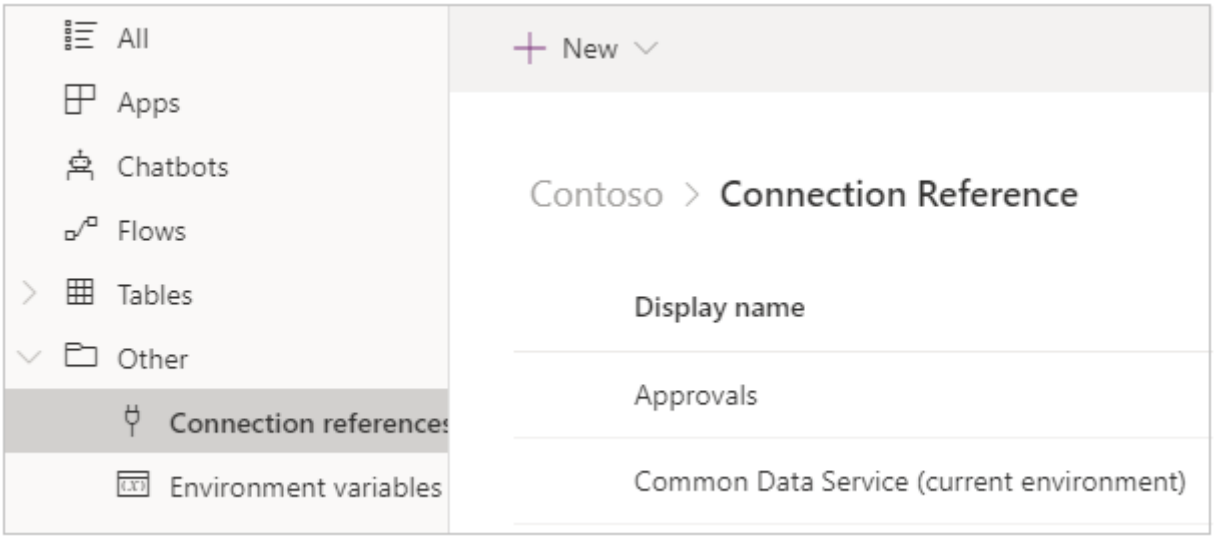
Tables

In Power Apps, a *table* defines information that you want to track in the form of records, which typically include properties such as company name, location, products, email, and phone. You can then surface that data by creating an app that refers to the table. For more information about tables, go to Tables overview.



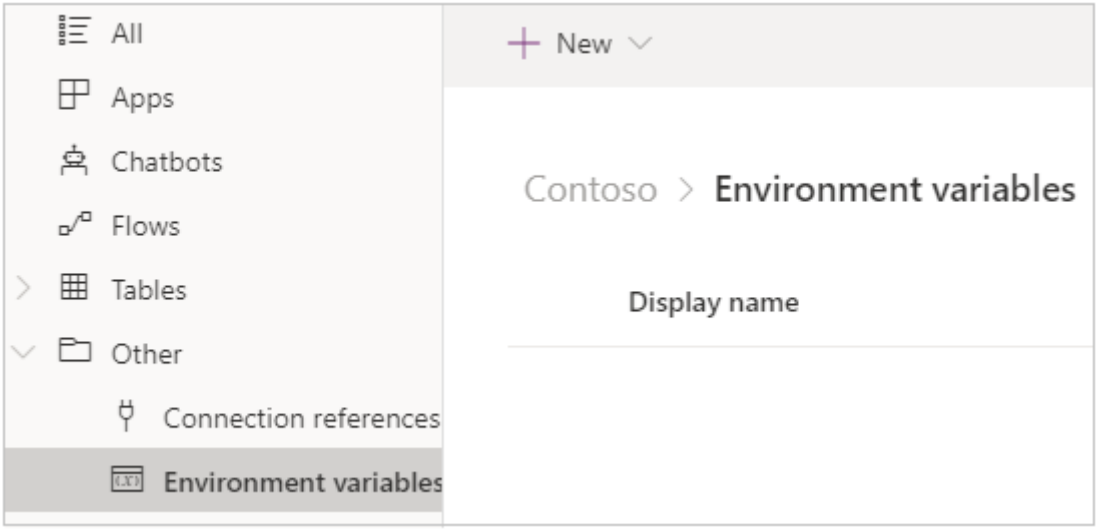
Other - Connection references

Lists connections available to the environment. Data is stored in a data source, and you bring that data into your app by creating a connection. A Power Apps connection connects your app to such a data source. You can use several other types of connections to connect to the Dataverse for Teams environment within the team. More information: Overview of canvas-app connectors



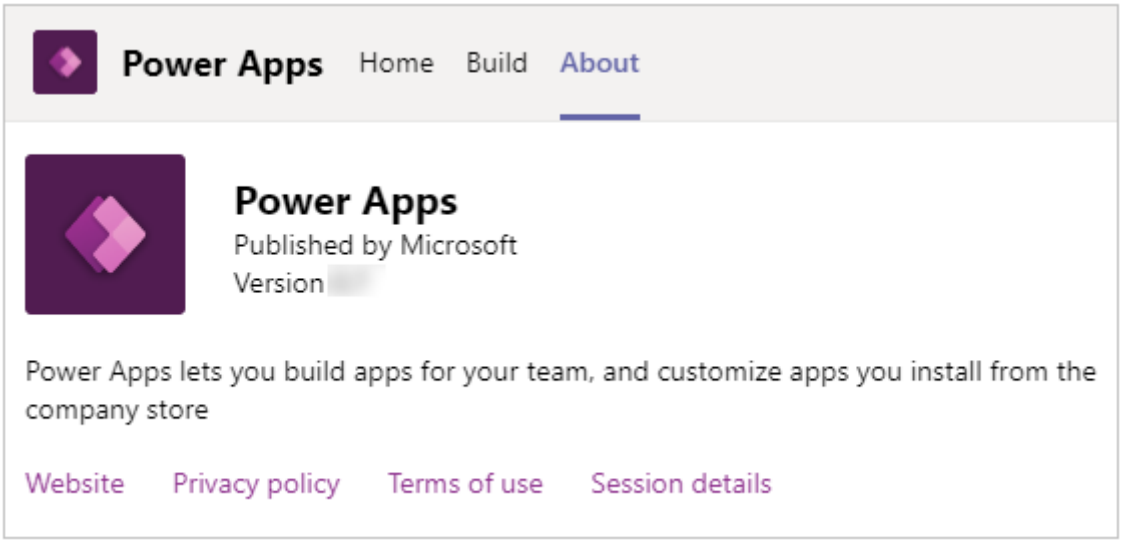
Other - Environment variables

Lists the environment variables for your selected team's environment. Apps and flows often require different configuration settings across environments. You can use environment variables as configurable input parameters separately to manage data, rather than hard-coding values in your customization or using additional tools. More information: Environment variables overview.

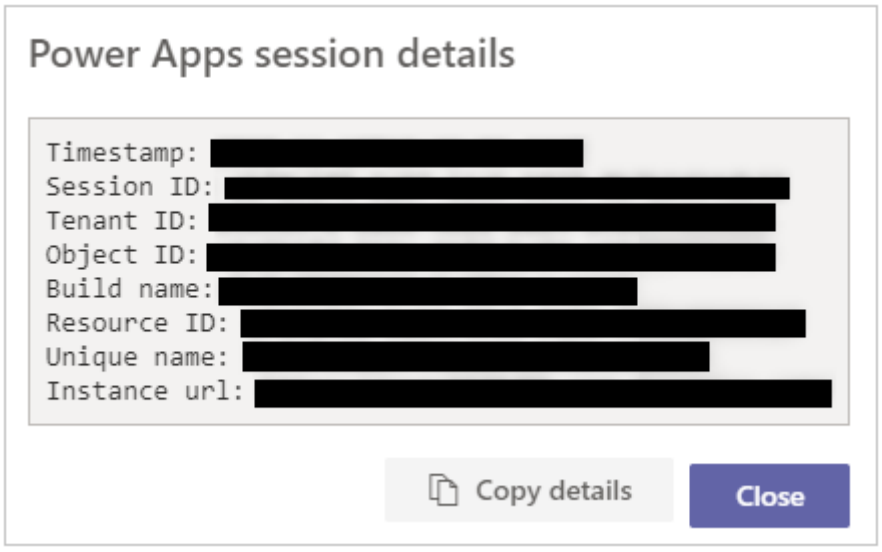


About

Select the **About** tab to view the version of the installed Power Apps app.



You can get your current session details by selecting the **About** tab and then selecting **Session details**. The session details include useful information that you can share with others when investigating an issue or working with support teams.



- **Timestamp**: Date and time in Coordinated Universal Time (UTC).
- **Session ID**: Unique GUID representing the current Power Apps session.
- **Tenant ID**: Tenant ID of Teams and Power Apps.
- **Object ID**: Object ID of the user account in Azure Active Directory.
- **Build name**: Build and version details of the Power Apps app.
- **Resource ID**: ID of the resource.
- **Unique name**: Unique name of the Dataverse organization.
- **Instance url**: URL of the Dataverse instance.

> **Note**
>
> Use session details only when you contact Microsoft Support to investigate an issue.

Canvas apps terminology reference and definitions

When you create canvas apps by using Power Apps from the Teams interface, you need to understand the terms and definitions for various objects, options, and actions in an app. The following table defines these canvas app terms.

Expand table

Component or action	Description
Power Apps Studio	An application used to make or author a canvas app.
Canvas app	An app that you can create from scratch or by using any available template through Power Apps Studio.
Model-driven app	An application based on Dataverse forms, views, tables, columns, and other components.
Connector	A connection object that connects an app with a data source. For example, a connector for SharePoint or a connector for OneDrive.
Control	An object that you add on the canvas to provide certain app functionality.
Property	Behaviors representing the controls available in an app.
Function	Readily available functionality to extend the behavior of a control or app.
Formula	A combination of one or more functions to accomplish a certain task or action.
Screen	The visible area of an app enclosing available visible controls. An app can have multiple screens.
Component	Reusable combinations of controls that you can define, save, and use for new apps.
Save an app	Saving an app for the first time, or saving new changes to an app.
Publish an app	Making an app available to the app store for general consumption.
Versions	Each time you save an app, you create a new version that you can restore.
Share an app	Allowing users or groups to use the app, as consumers or co-owners.

See also

Understand Power Apps Studio
Create your first app

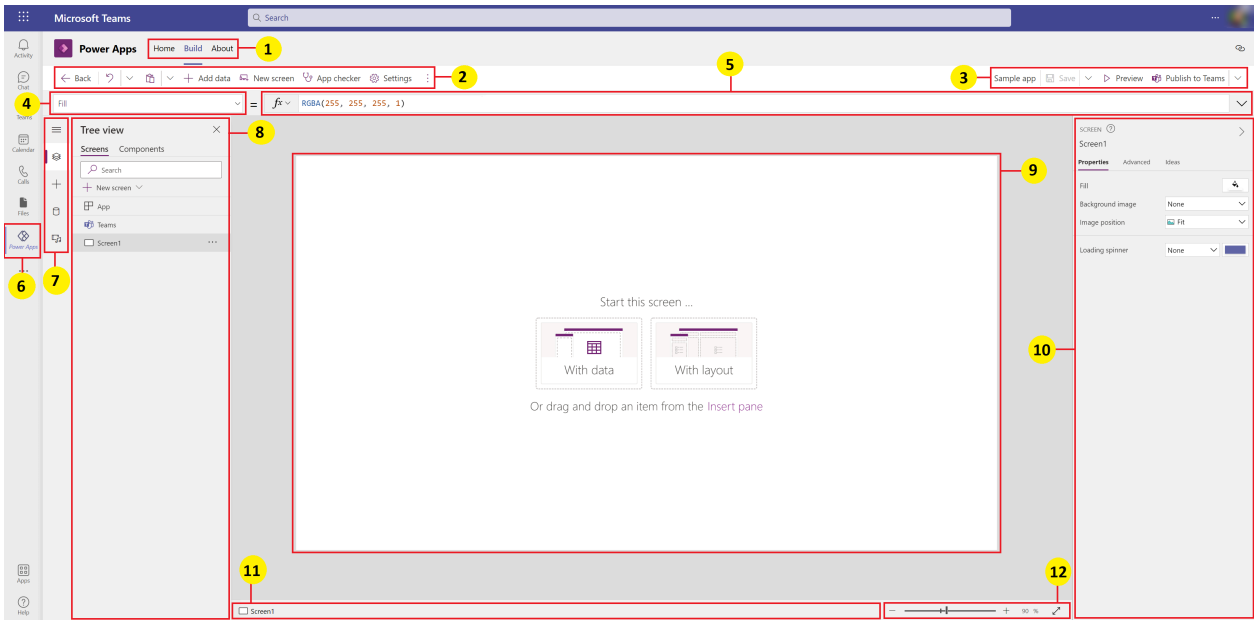
Last updated on 01/13/2026

Understand Power Apps Studio in Dataverse in a Teams environment

Article • 12/16/2022

When you create a canvas app by using any method, you're taken to the canvas app builder called Power Apps Studio. You can use Power Apps Studio to design, build, and manage your canvas app.

Overview of Power Apps Studio



1. Build hub – select different pages in the Power Apps app.
2. Power Apps Studio options – options relevant to the settings in a Power Apps Studio session.
3. App actions - options to rename, save, preview, or publish the app.
4. Properties list - list of properties for the selected object.
5. Formula bar - compose or edit a formula for the selected property with one or more functions.
6. Power Apps app – Power Apps app in Microsoft Teams.
7. App authoring menu - selection pane to switch between data sources and insert options.

8. App authoring options - details pane with options relevant to the selected menu item for authoring the app.
9. Canvas/screen - primary canvas for composing the app structure.
10. Properties pane - properties list for the selected object in UI format.
11. Screen selector - switch between different screens in an app.
12. Change canvas screen size - change the size of the canvas during an authoring experience in Power Apps Studio.

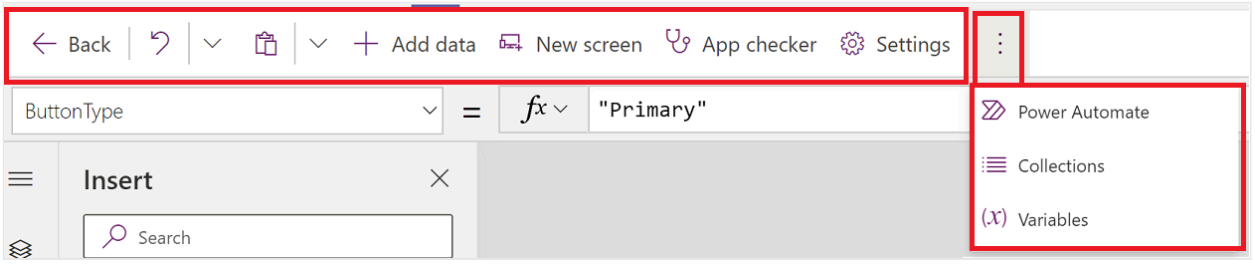
Let's understand each option in Power Apps Studio in detail.

1 - Build hub

Opening Power Apps Studio from the build hub opens an app authoring experience that inherits the Teams interface and options. The **Home**, **Build**, and **About** tabs are described in Overview of the Power Apps app.

2 – Power Apps Studio options

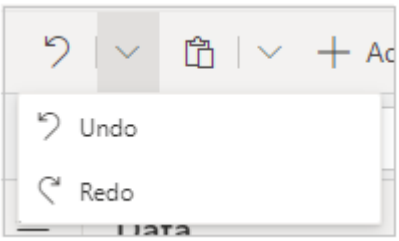
Power Apps Studio options are available on the menu in the upper-left corner. The options are relevant to the current session and app-related settings.



Back

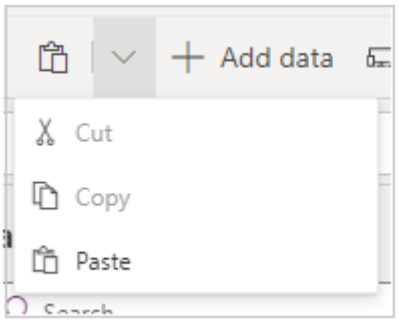
Takes you back to the build hub, closing the current Power Apps Studio session.

Undo and redo



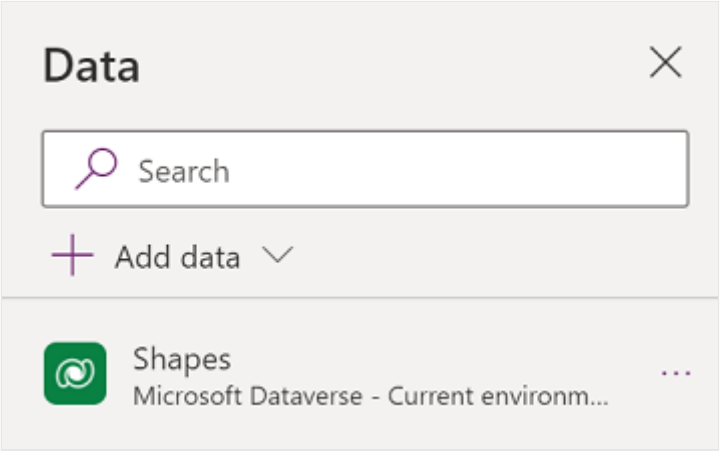
- **Undo** – undo the last action.
- **Redo** – repeat the last action.

Cut, copy, and paste



- **Cut** – Cut the selection, and store it in the clipboard.
- **Copy** – Copy the selection, and store it in the clipboard.
- **Paste** - Paste the last cut, or copied selection from the clipboard.

Add data



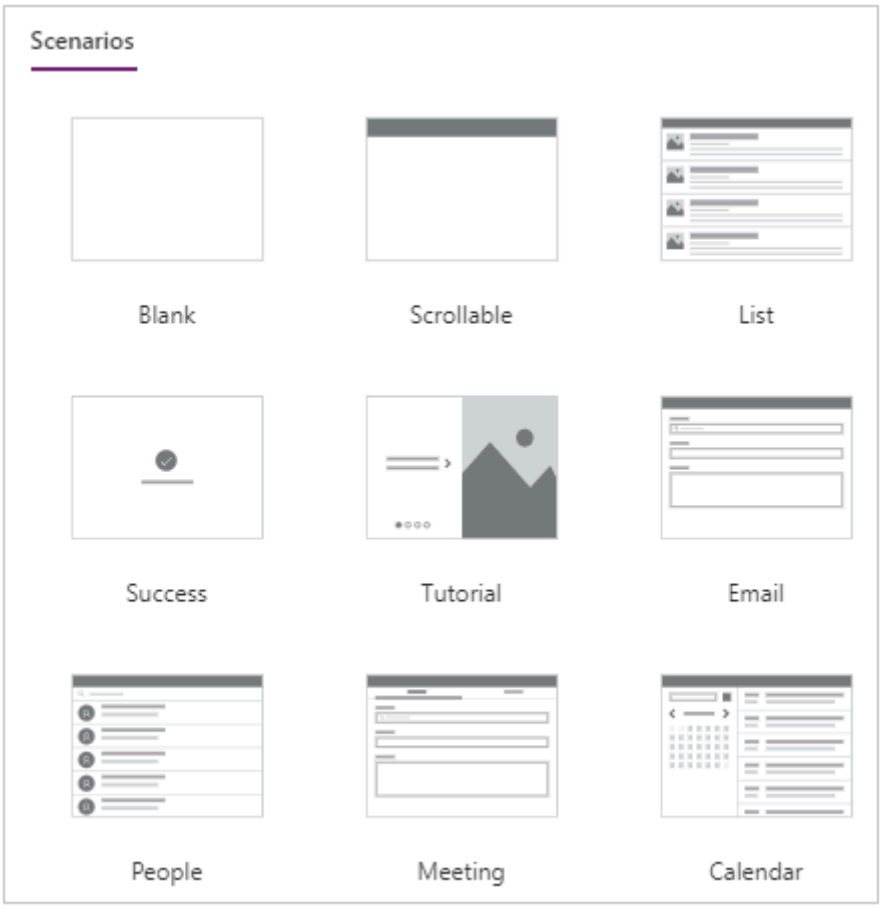
- Create a new table by selecting **Create new table**.
- Select any other existing tables from the current environment.
- Search and select a connector, such as **SharePoint** or **SQL Server**.

New screen

Add screens based on the available layouts.

Layouts

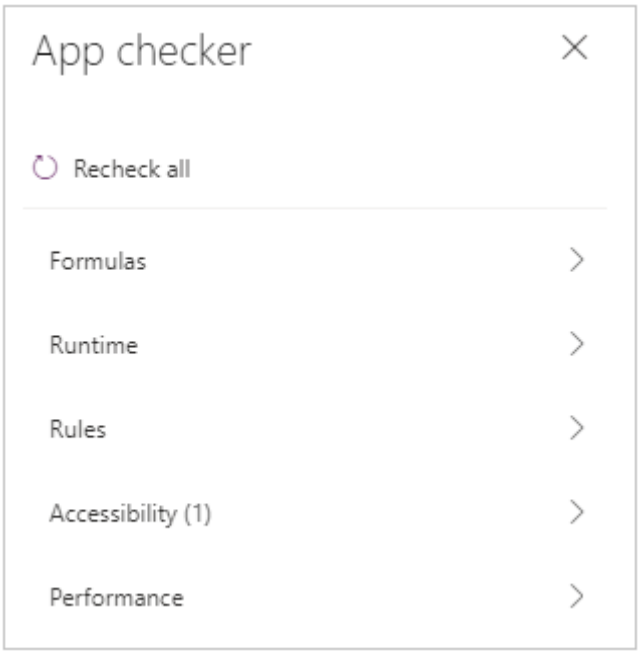
Select a new screen to add to the app based on the layout of the screen.



Select a screen type based on the available scenarios, such as **Blank**, **Scrollable**, **List**, **Success**, **Tutorial**, **Email**, **People**, **Meeting**, or **Calendar**.

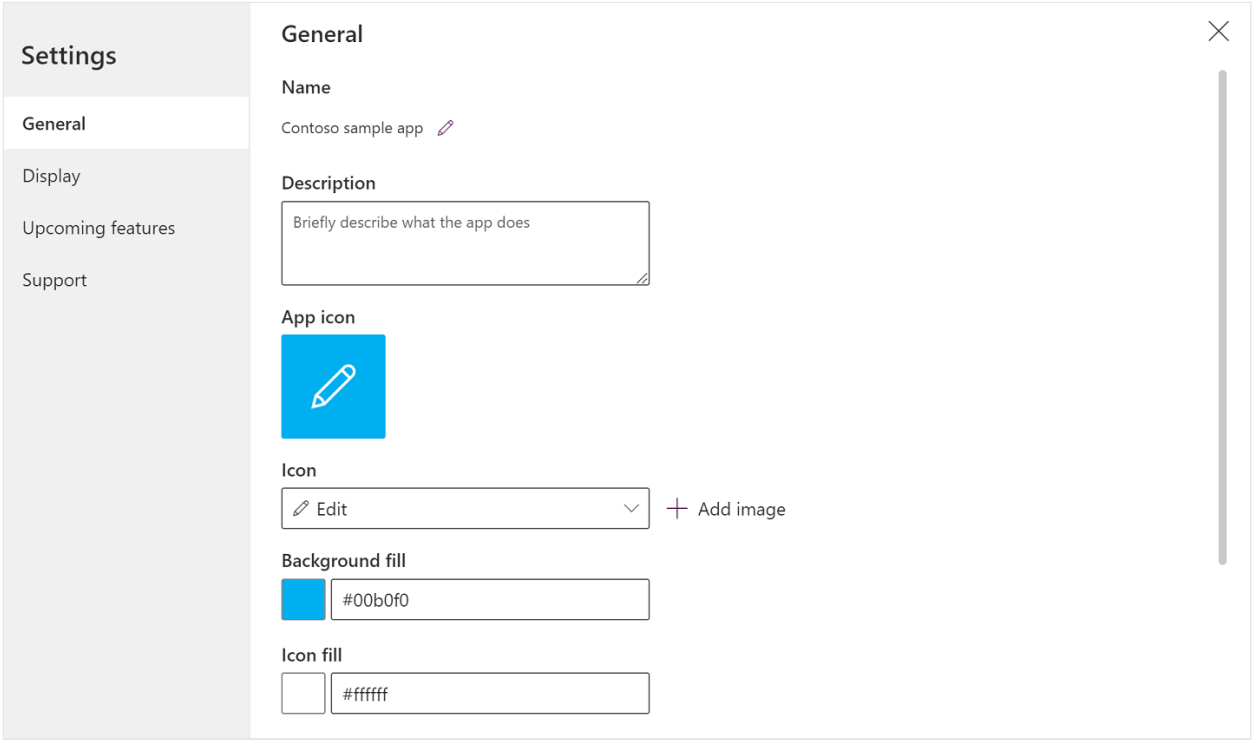
App checker

Runs the App checker with available rules and shows the results.



Settings

Configure the app's general settings.



General

Shows app name, and allows changing the app icon. To update the icon, select a new icon or background color. To add or update description, enter text in the text box. You can also upload a custom icon for the app using the **Browse** option.

Display

Shows the screen size and orientation. To change, select the radio buttons for **Orientation** and **Size**.

Advanced settings allow you to further customize the app screen configuration.

- **Scale to it**: Scales the app to fit available space.
- **Lock aspect ratio**: Locks the height and width ratio.
- **Lock orientation**: Maintains app orientation when device rotates.

To change the setting, toggle the switch.

More information: Change screen size and orientation

Upcoming features

Allows you to configure advanced settings for the app that include features under preview, experimental or retired features.

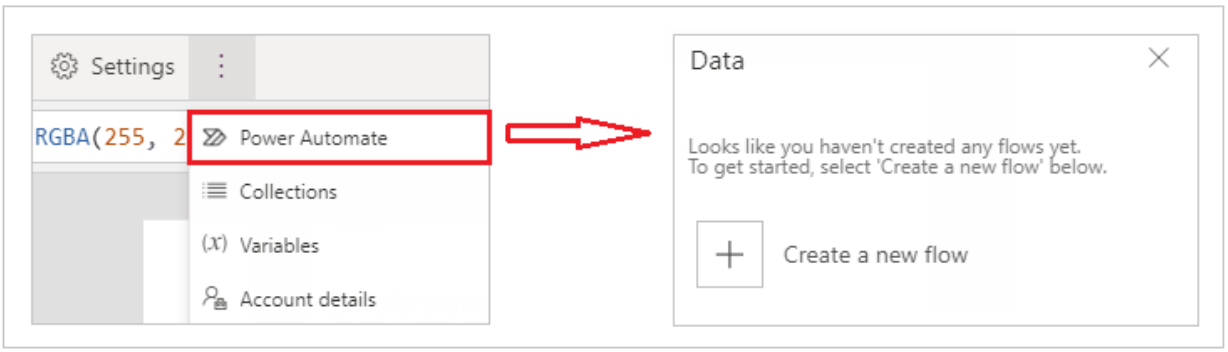
More information: Understand experimental, preview, and deprecated features in Power Apps

Support

Shows current Power Apps Studio session details, version, and other information useful when working with Microsoft support.

Power Automate

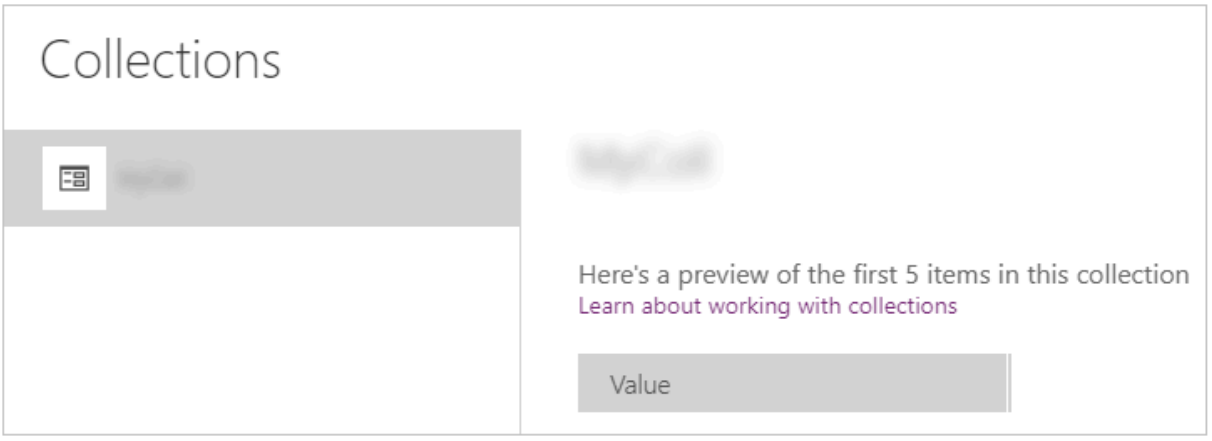
Create a new flow with Power Automate, or select any available flow.



More information: Create flows using the Power Apps app in Teams

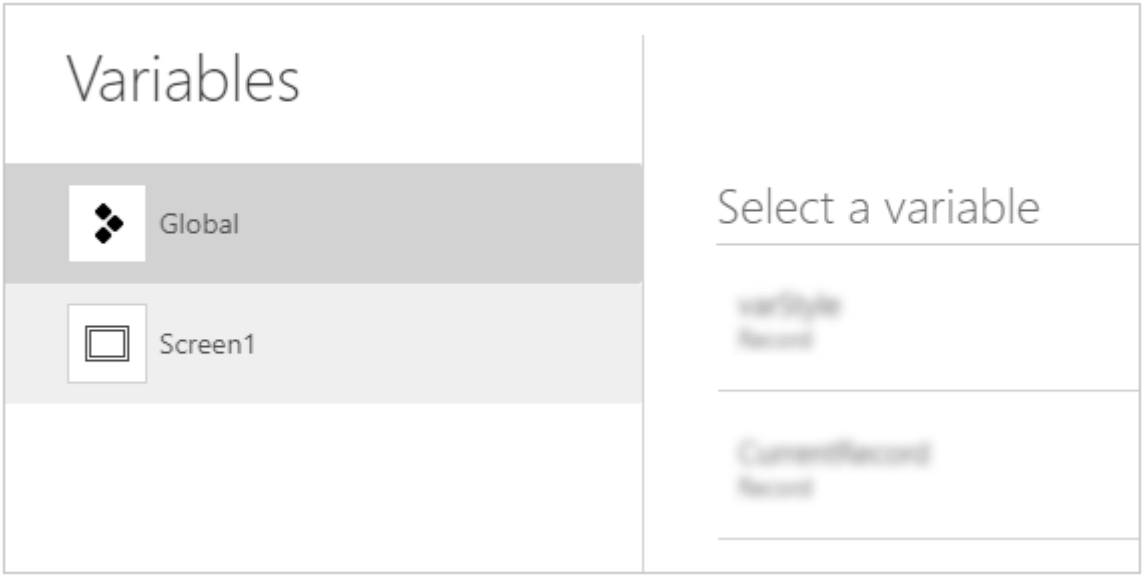
Collections

A collection is a group of items that are similar, such as products in a product list. This section lists the collections used by the current app. More information: Collections in canvas apps used by the app.



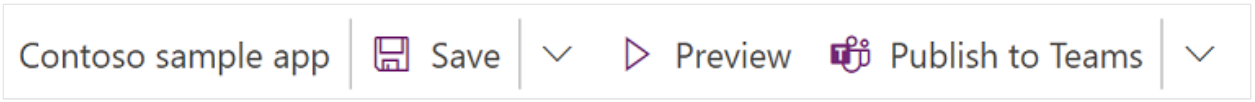
Variables

You can save data such as the result values from a data set into temporary storage by using variables. This section lists variables used by the current app. More information: Variables in canvas apps used by the app.



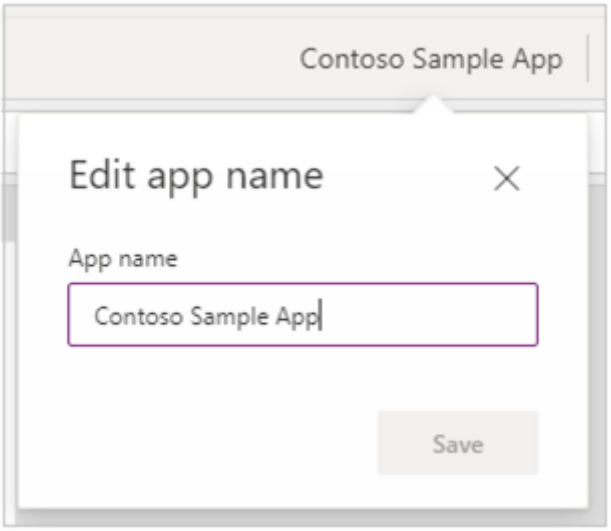
3 – App actions

Use the options in the upper-right corner of Power Apps Studio to work with app-specific actions.

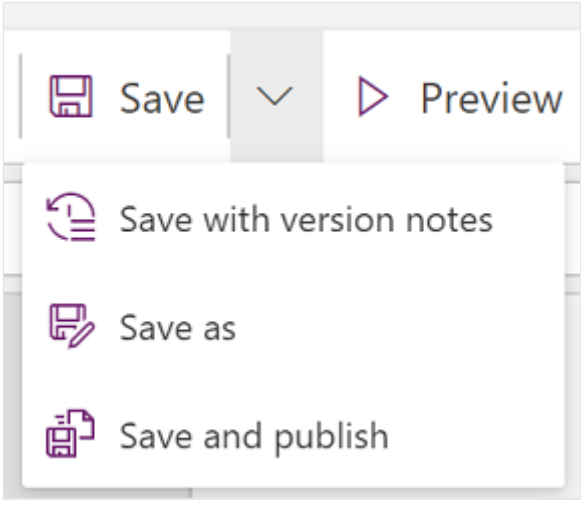


App name editor

Select the name of the app to edit it.



Save



Save allows you to perform the following actions:

- **Save** - Saves recent and unsaved changes you made to the app in Power Apps Studio. Each time you save changes, a new version is created.
- **Save as** - Duplicate the app by saving the app with a different name.
- **Save and publish** - Allows you to both save the app, and publish it at the same time.

Preview

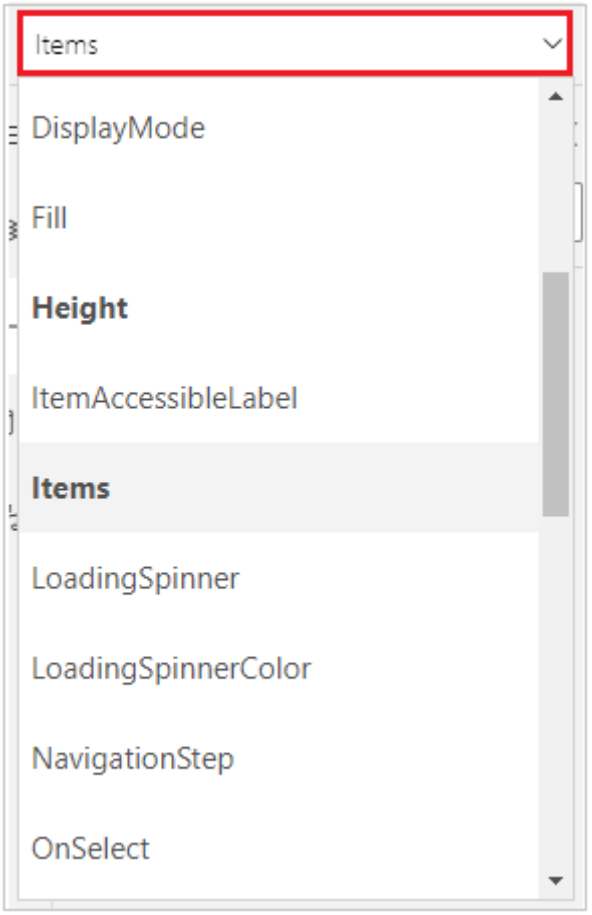
This will show a preview version of the app in Microsoft Teams that you can interact with.

Publish to Teams

Publishes the app's current version to a channel within Microsoft Teams. For more information about publishing an app, go to Publish an app.

4 – Properties list

Shows the list of available properties for the selected object on the canvas. The properties list changes based on your selection. For a complete list of all properties, go to All properties.

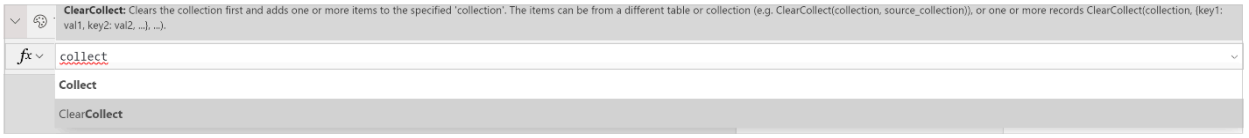


5 – Formula bar

Use the formula bar to add, edit, or remove functions relevant to the selected object and the property selected from the properties list. For example, select the screen to update the background by using the RGBA function.

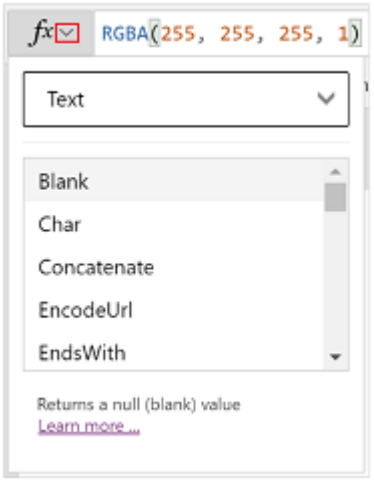


The formula bar is IntelliSense-enabled, and provides tips as you enter text to help you with the function syntax. If a formula returns an error, tips relevant to the syntax error and mitigation steps are displayed. When you start entering text that matches one or more functions, the formula bar shows inline function help and highlights help text relevant to the cursor position.

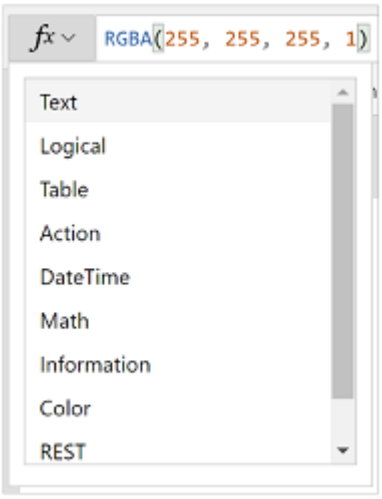


Similarly, you'll find help when working with complex functions, nested functions, or when correcting a formula syntax.

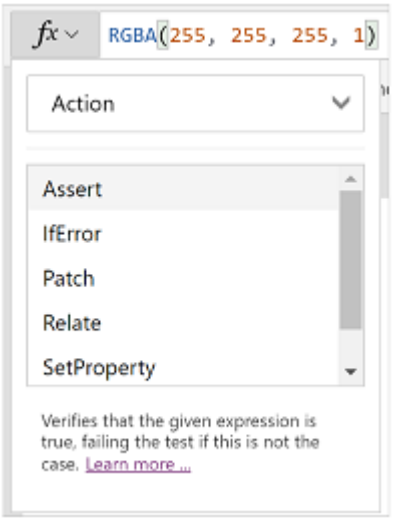
For a quick and easy function reference, you can also select the formula drop-down menu.



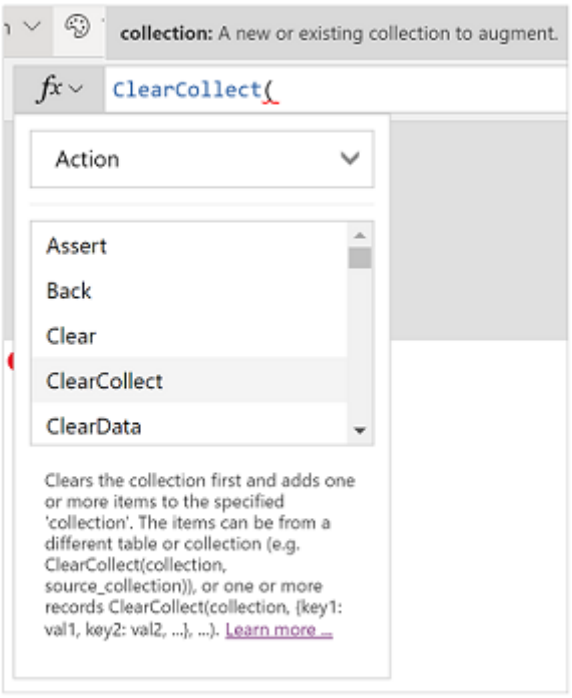
Select an event type from the drop-down menu at the top of the dialog box, such as **Action** instead of **Text**.



Select an action that you want to add a function for.



The available functions for the selected event type are dynamically updated depending on the object you select. For example, if you selected a button on the canvas, the available **Action** functions also include the function *ClearCollect()*.



You can read the description of the selected **Action** function. Double-clicking a function name adds it into the formula bar.

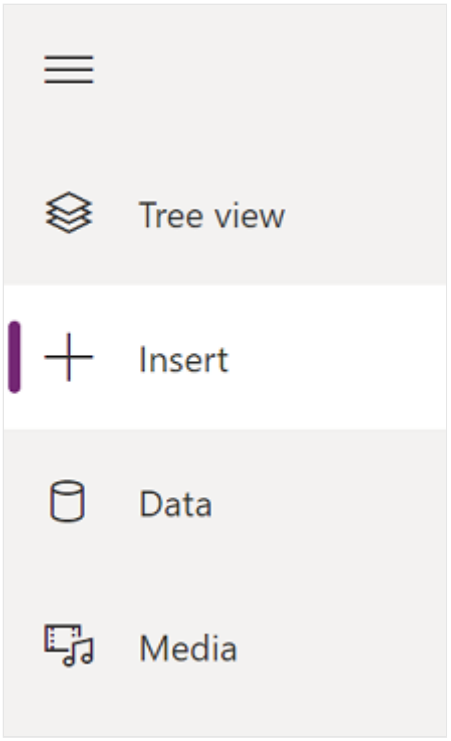
For a complete list of all canvas app functions, go to Formula reference.

6 – Power Apps app

The Power Apps app in Teams is described in detail in Overview of the Power Apps app.

7 – App authoring menu

Switch between various authoring options while working with the app.



> **Tip**
>
> You can also select the expand button to expand the list to include names instead of just icons.

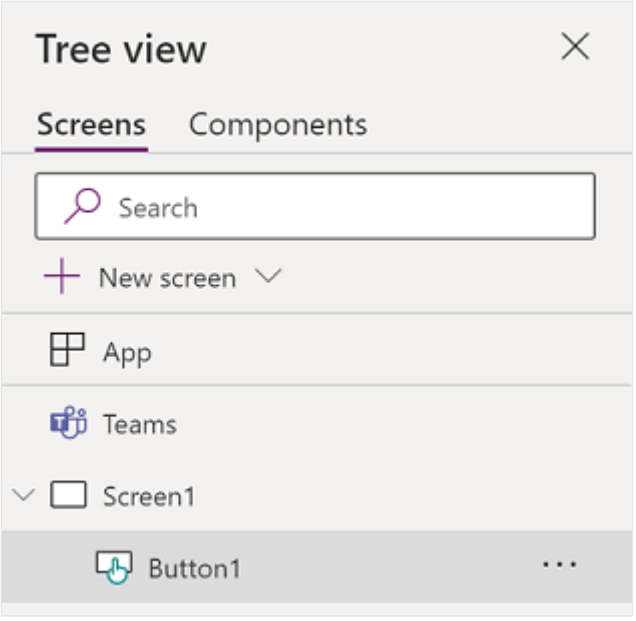
- **Tree view** - Shows a tree view of all screens and controls in the current app.
- **Insert** - Allows you to add different controls to the screen.
- **Data** - Add or remove data such as tables that the app connects to.
- **Media** - Insert or remove media from the app.

8 – App authoring options

The options for working with canvas apps change depending on the selection on the left pane.

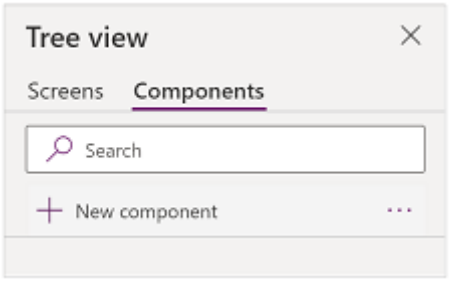
Tree view

Select the tree view to show the screens available in the app,



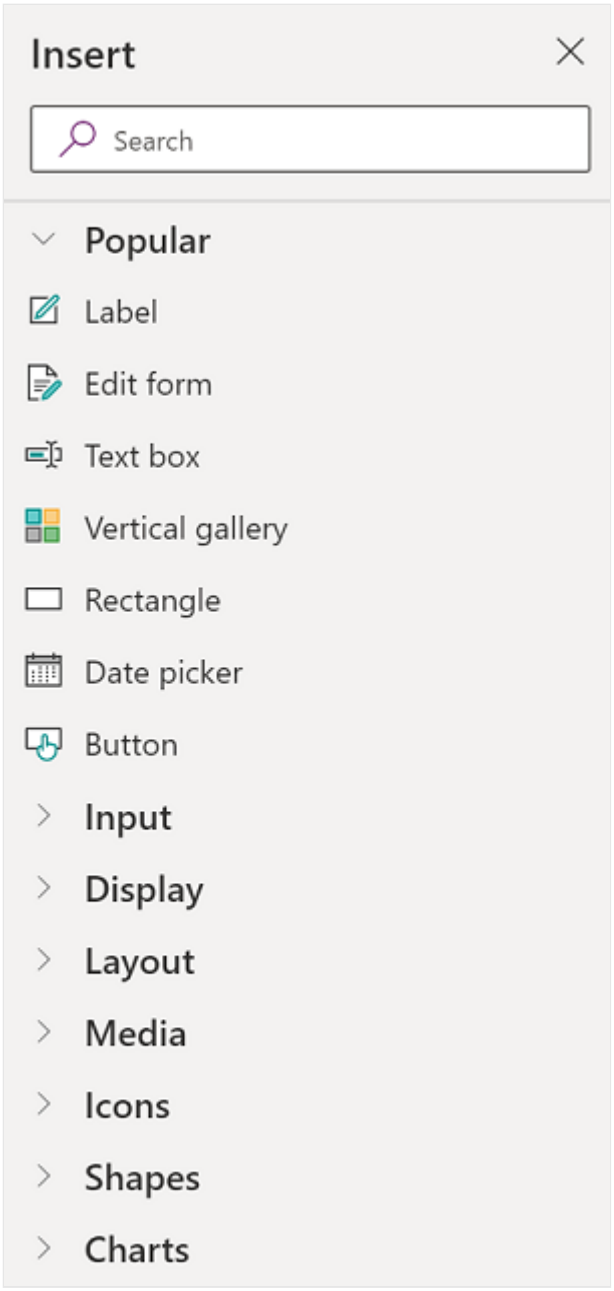
> **Tip**
>
> Select **App** in the tree view to work with app-specific controls or to change app behavior, such as adding a formula on *OnStart* event of the app.

Switch to the **Components** tab to work with component library features. You can add new components or reuse those that were already published from published component libraries. More information: Component library



Insert

Insert shows all the popular objects or controls that you can add on the selected screen in your canvas app. You can also expand other choices or use the components option to insert controls from a component library.



To insert controls on the canvas, you can drag the control to the canvas, select the control, or select **(...)** and then select **Add to canvas**.

> **Tip**
>
> Dataverse for Teams provides new components built on the Fluent UI framework. More information: Fluent UI controls.

Popular controls

Expand table

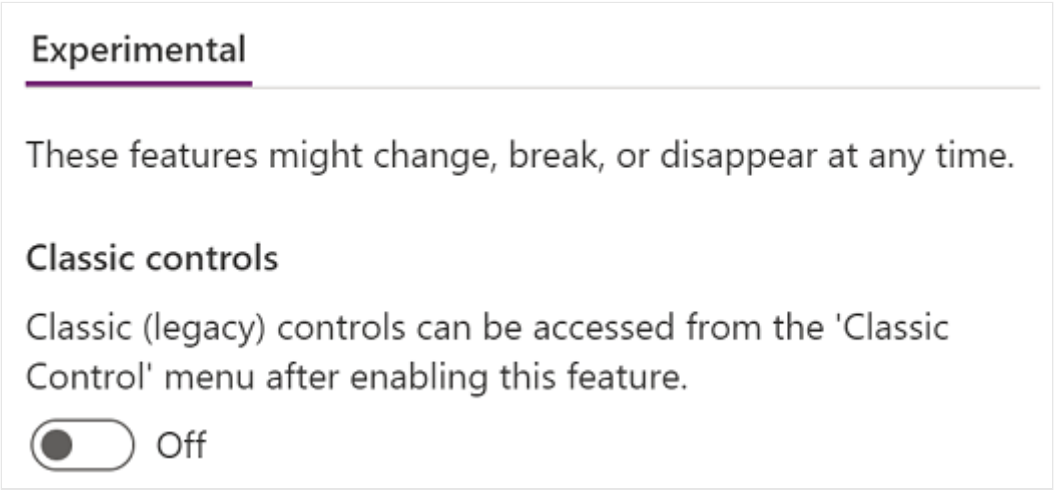
Name	Description
Label	A box that shows data such as text, numbers, dates, or currency.
Edit form	Display, edit, or create a record in a data source.
Text	A box that shows text.
Text box	A box in which the user can enter text, numbers, and other data.
Vertical gallery	A control that contains other controls and shows a set of data.
Add icon	Graphics for which you can configure appearance and behavior properties.
Rectangle	A rectangular shape to configure the canvas appearance.
Date Picker	A control that the user can select to specify a date.
Button	A control that the user can select to interact with the app.

For more information about the controls that you can insert, and their properties and definitions, go to Controls and properties in Power Apps.

Classic controls

Dataverse for Teams uses Fluent UI controls by default. If necessary, you can enable classic controls by using the experimental feature setting. To do this, go to **Settings** > **Upcoming features** > **Experimental** > toggle **Classic controls** to **On**.

After you enable classic controls, you'll be able to see and add them from the **Classic** category.

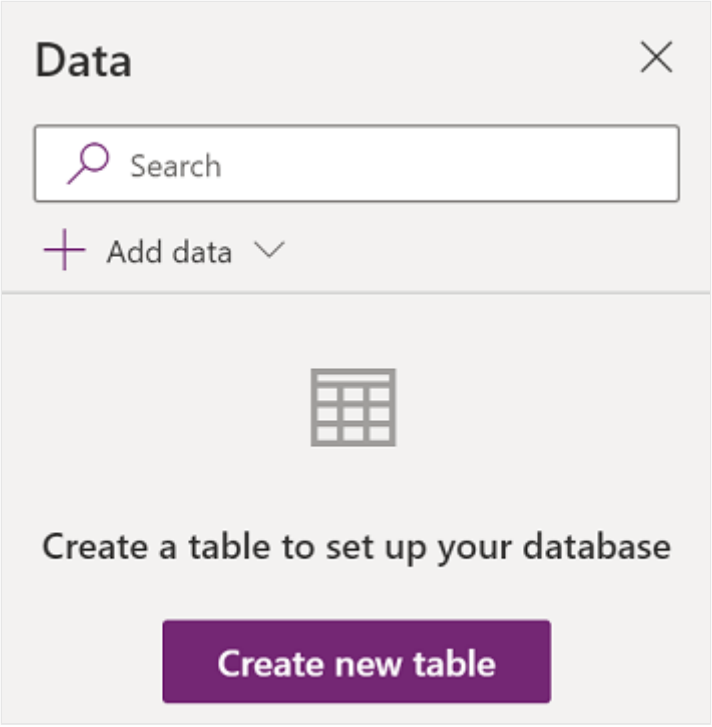


For example, you may need to enable classic controls when using dependent drop-down lists when using Dataverse for Teams.

Data

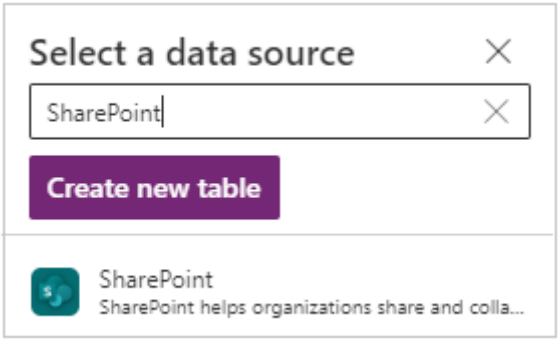
Add, refresh, or remove data sources from your canvas app. You can add one or more connections by using data sources.

In addition to data stored within tables, there are many connectors available to interact with data in popular SaaS, services, and systems.



Select **Create new table** to create a new table in the Dataverse for Teams environment.

To choose other connectors such as SharePoint, OneDrive, or SQL Server, you can enter text in the data source search box or select from the list of connectors.

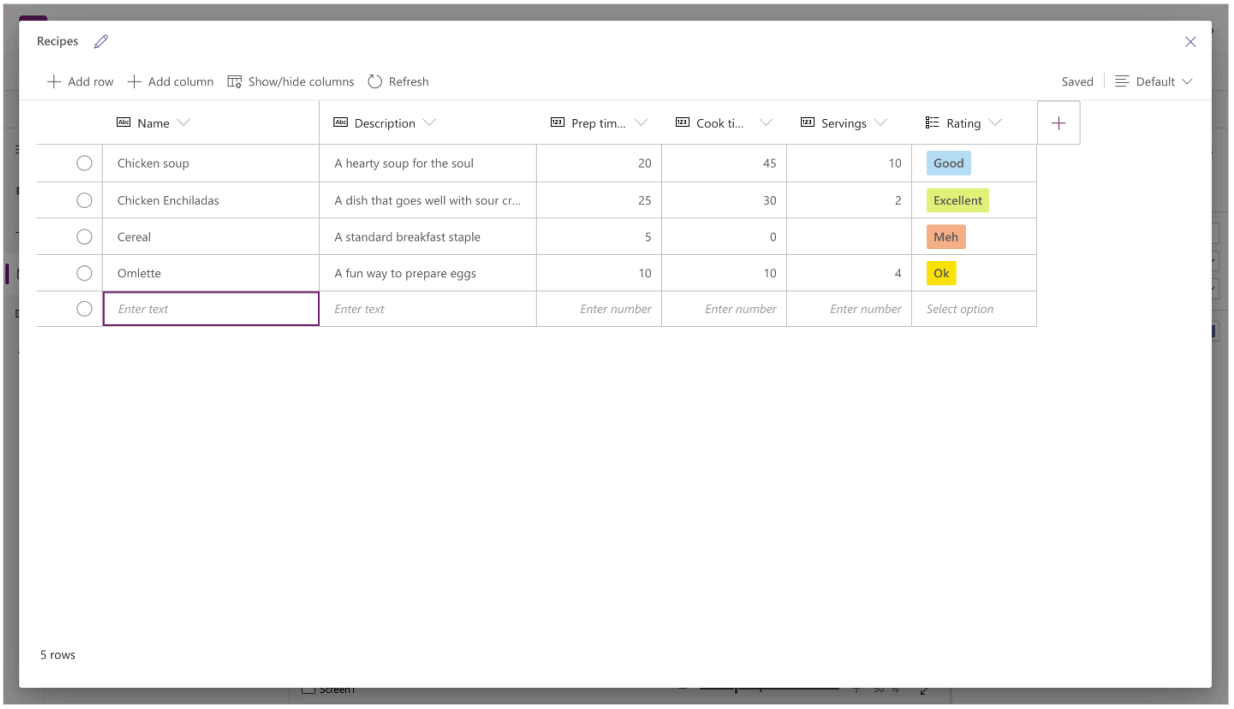


More information: List of all Power Apps connectors

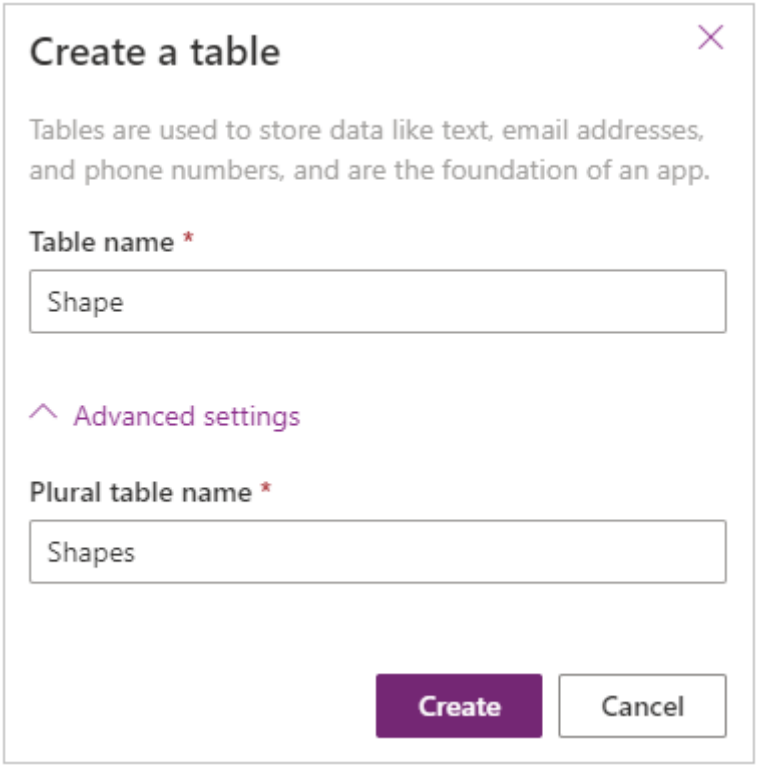
Visual editor

When you select **Create new table** and enter a name for the new table, you open visual editor where you can design the table, add columns of different data types, enter data in

rows, and save the changes.



To get started with visual editor, select **Create a table** and enter the table name. You can also expand the *Advanced settings* to update the table's plural name. For example, a table name can be *Shape*, and the plural table name can be *Shapes*.



Watch this video that shows you how to quickly create table and columns.
https://learn-video.azurefd.net/vod/player?id=40f4f29f-77df-420a-a182-cb71b39f4613&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Funderstand-power-apps-studio

> **Note**
>
> You can use visual editor in Power Apps Studio to quickly create a table while authoring an app. However, you can also use the Build tab to edit or delete the tables you create by using Power Apps Studio. Go to Create tables in Microsoft Teams for more information about creating tables by using the **Build** tab.

Understand visual editor

Visual editor allows you to work with table rows, columns, and data.

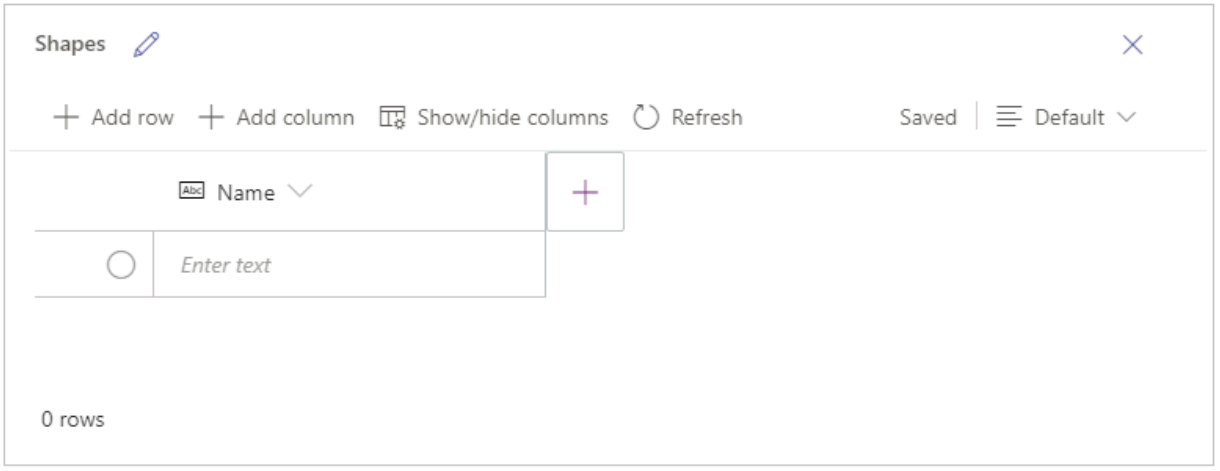
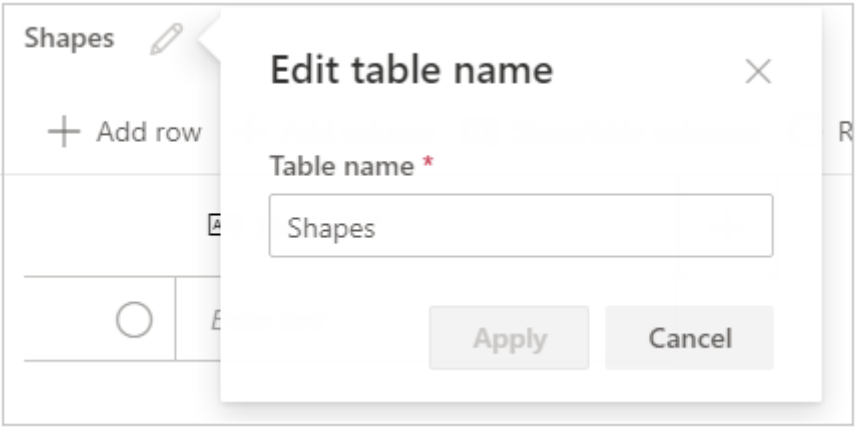


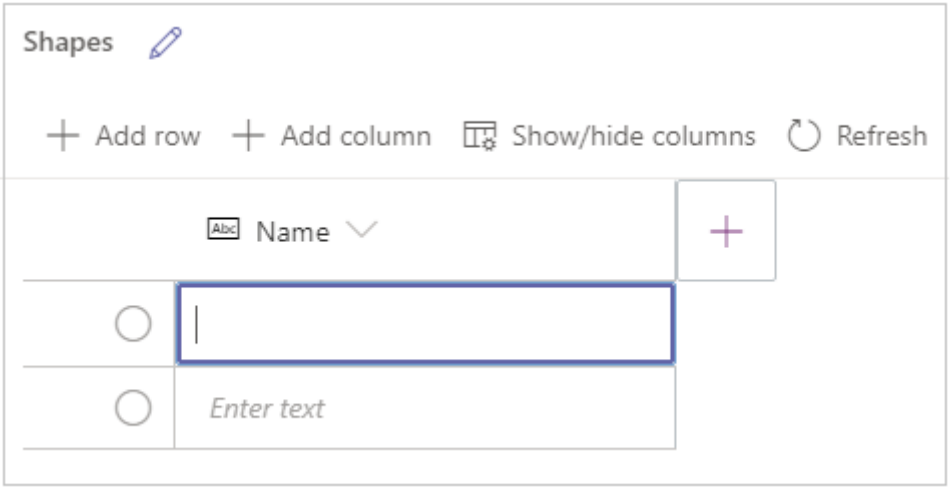
Table name

Select **Edit** to edit the name of the table.



Add row

Select **Add row** to add a new row to the table.



Add columns

Select **Add columns** to add new columns of the available column types supported by visual editor.

Supported column types

Visual editor supports specific data types as columns. The following options are available when creating a new column using visual editor inside Power Apps Studio:

- Text
- Email
- URL
- Phone
- Auto number
- Number
- Date (Only dates without the user locale (time zone) settings are supported.)
- Decimal
- Lookup
- Choice
- Yes/No

> **Important**
>
> To add columns of types that aren't supported by visual editor, such as file or image, create a table by using the solution explorer instead.

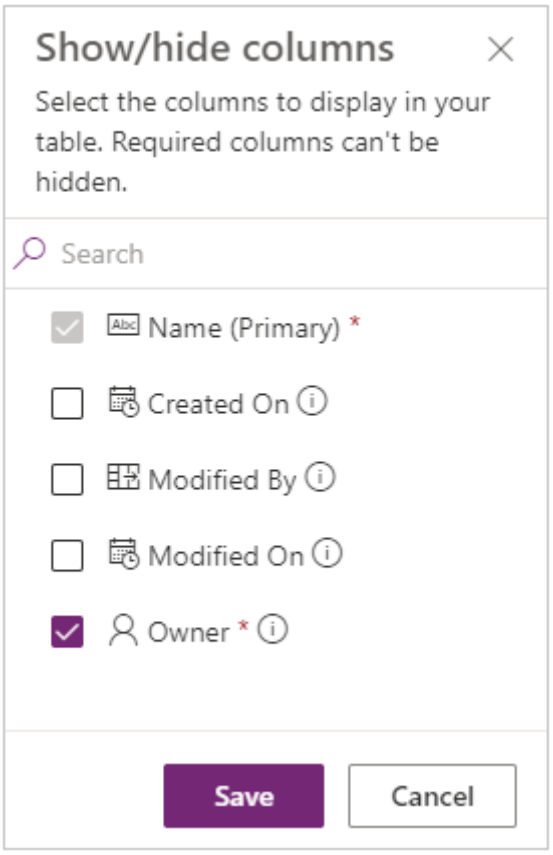
Advanced options

The advanced options for columns change depending on the type of column. For example, a **Text** column type has an advanced option for **Max length**. By contrast, an **Auto number** column type has options such as the type of autonumbering, prefixes, and maximum number of digits. More information: Types of fields

Show/hide columns

Use the **Show/hide columns** option to show or hide available columns, including columns automatically created as part of the table metadata.

For example, you can add an *Owner* column created by default to the existing table.

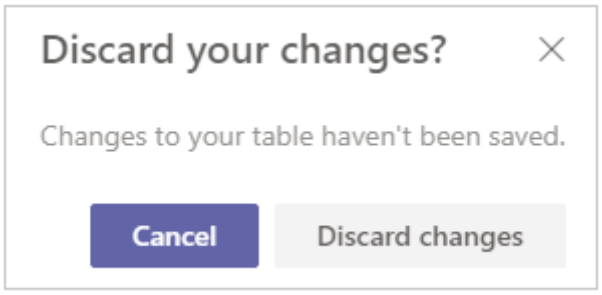


Refresh

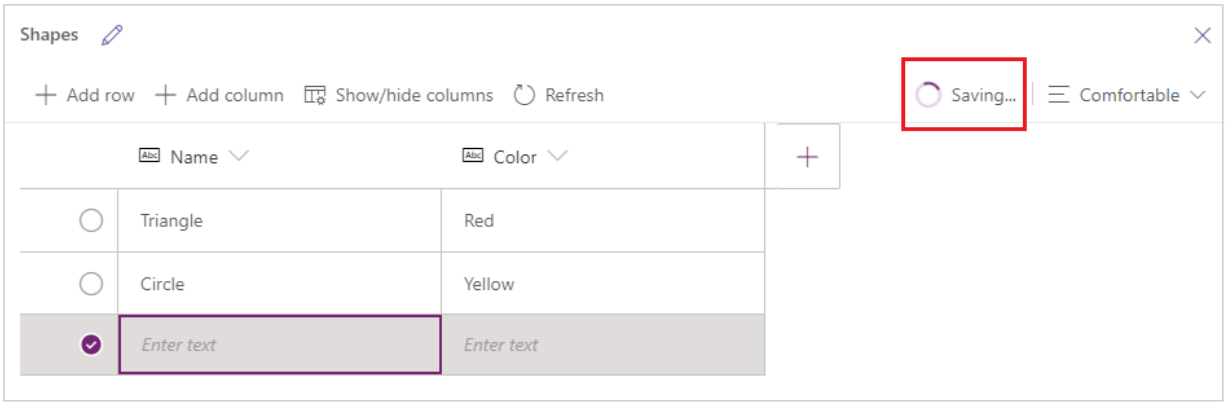
Refreshes the current table with data.

Save

To save changes to a table, select **Save.** When you change a table and try to close it without saving changes, you're prompted to discard changes.



To ensure that changes to the table are saved, you can select the next row inside the table, or select any other cell after editing a cell to trigger the auto save functionality.



After you close a saved table, you'll see the table added to the list of available data sources in **Data** on the left pane.

Visual editor view

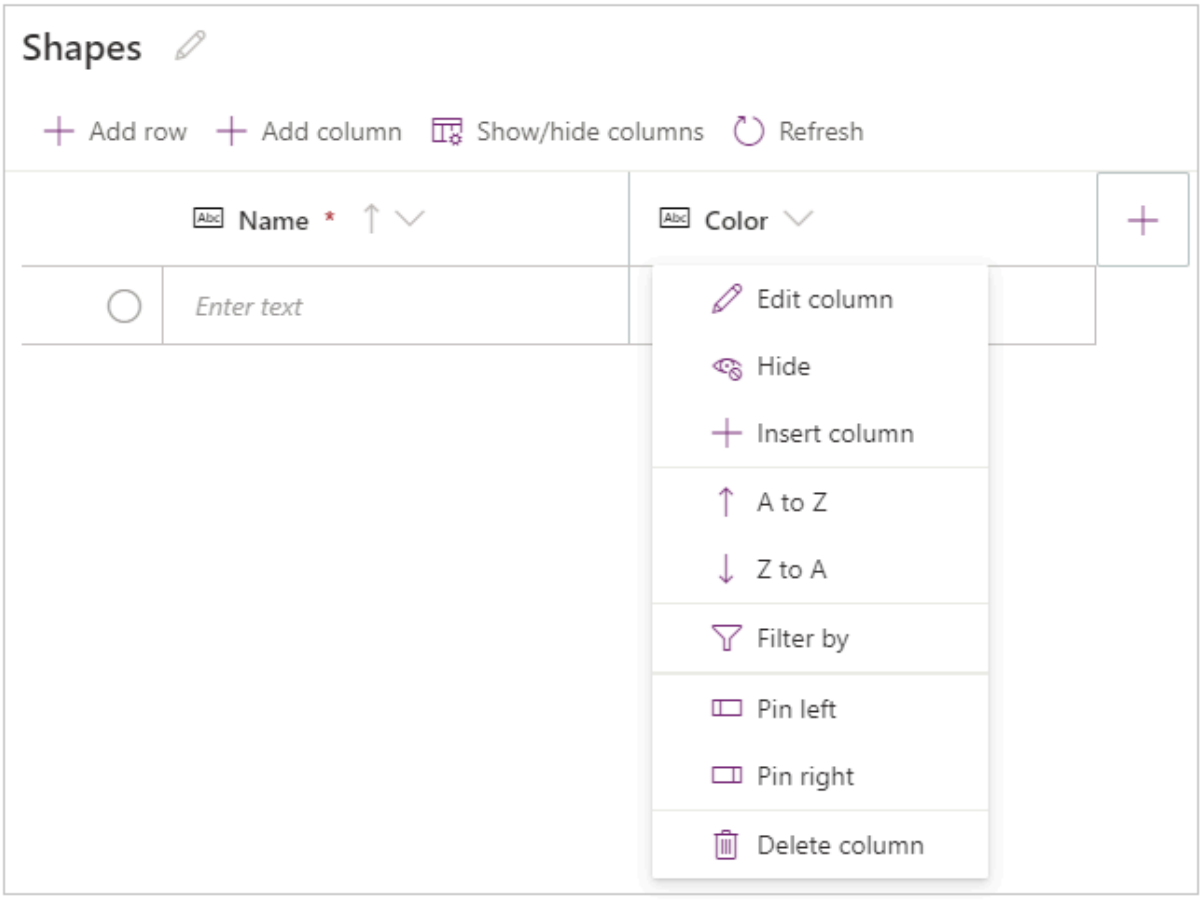
Select from the options of **Compact**, **Default**, or **Comfortable** layouts to switch the view with columns and rows spacing to change visual editor layout.

Row count

Shows the row counts in the table.

Column options

Select the drop-down menu next to the column heading to view column-related options.



Expand table

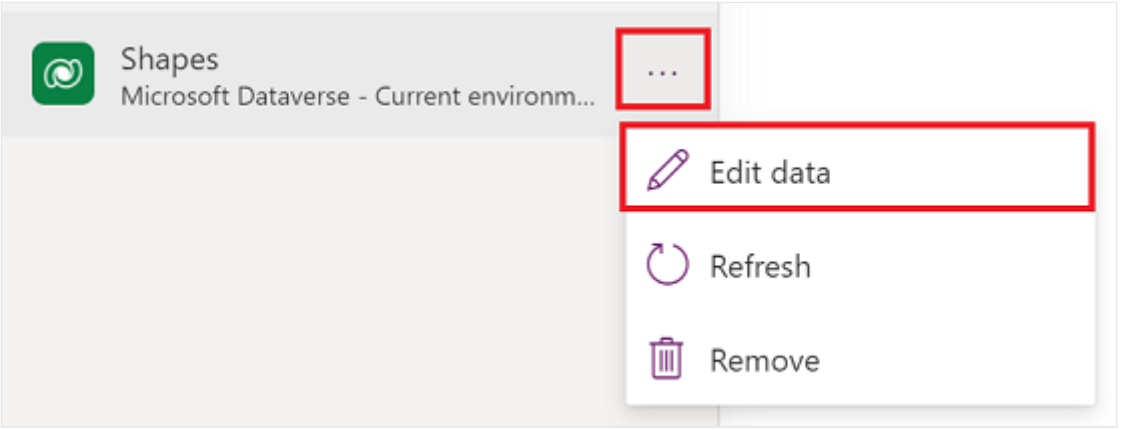
Option	Description
Edit column	Edit the column name or advanced options. After it's created, you can't change the name of the column.
Hide	Hide or unhide the column.
Insert column	Insert a new column at the selected column location.
A to Z	Sort records in ascending order.
Z to A	Sort records in descending order.
Filter by	Filter column data based on the filter criteria you define.
Move left	Move the column to the left from the current position.
Move right	Move the column to the right from the current position.
Pin left	Pin the column to the left side of the table.
Pin right	Pin the column to the right side of the table.

Option	Description
Delete column	Delete the column.

Edit existing table

After you add rows and columns, and add data, you can close the table and use it as the data source in your app. To edit content in the table, you can use the app controls or go back to visual editor.

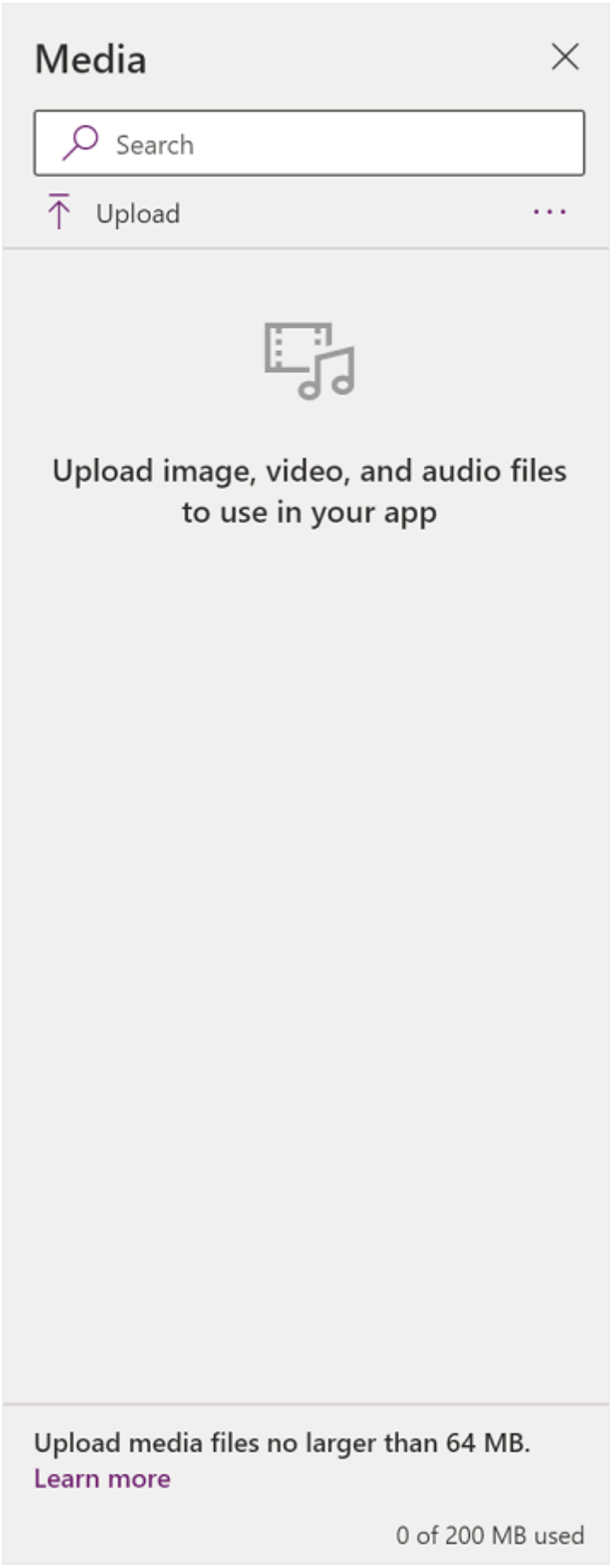
To edit the table, select the table data source, and then select **Edit data** to open the table in visual editor.



After you close visual editor, the data source is automatically refreshed to reflect the updated data in Power Apps Studio. You can also select **Refresh** to manually refresh Power Apps Studio to reflect the data in the controls you added on the canvas.

Media

Select **Media** to add images, video, or audio files to your app. Adding media directly to your app uploads the files to the app and uses the app storage. Each file uploaded to the app as media must be 64 MB or smaller, and the size of all media files uploaded to an app can't exceed 200 MB.



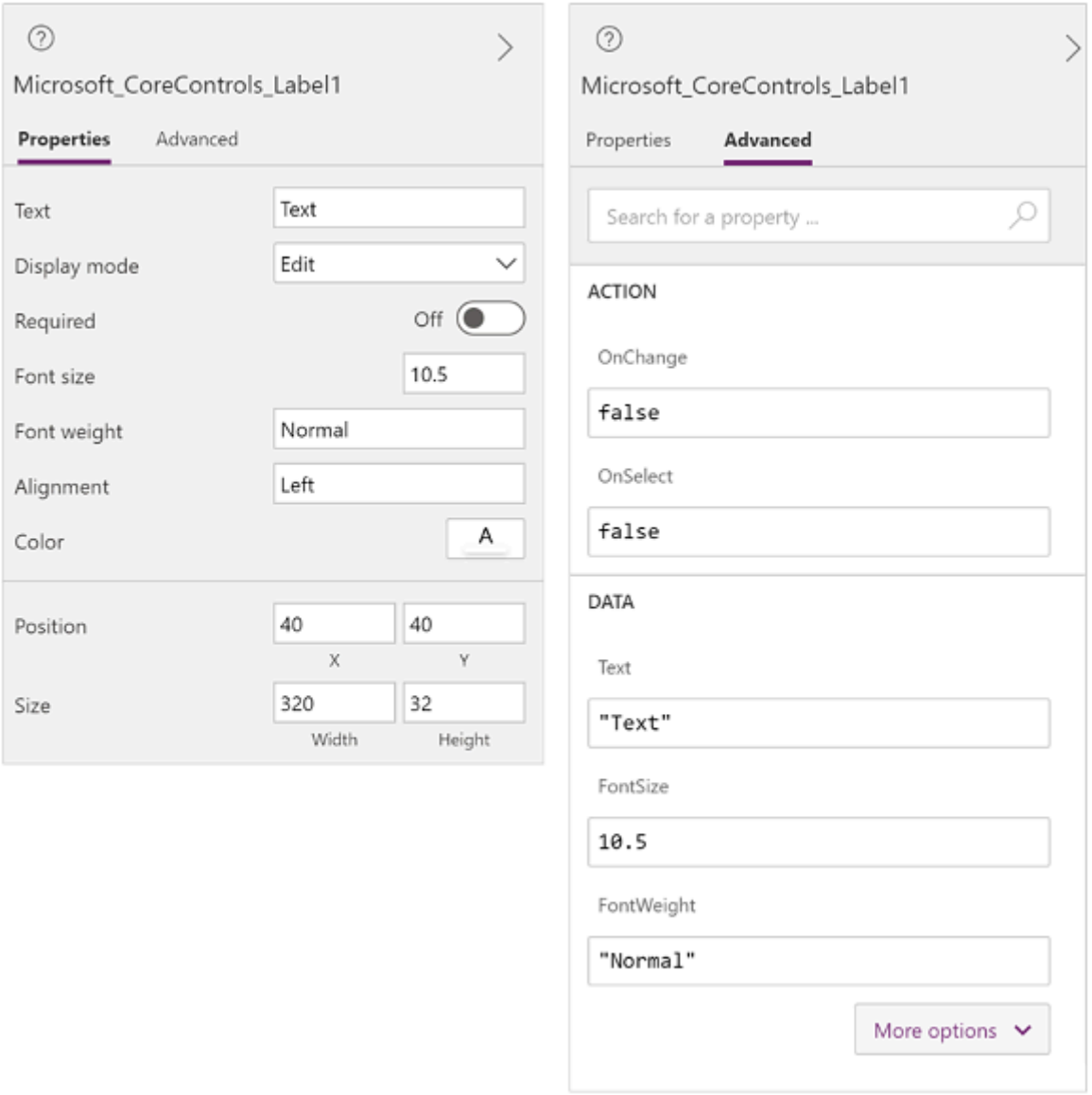
If you want to reference more media, consider using audio and video controls with URLs, using media from Azure Media Services, or from Microsoft Stream. More information: Using multimedia files in Power Apps

9 – Canvas/screen

The canvas shows the currently selected screen from the left pane.

10 – Properties pane

The properties pane shows properties and options available for the currently selected object on the canvas. The **Properties** tab shows generic options such as the name, color, size, or position. The **Advanced** tab shows more options for advanced customization. The advanced properties might sometimes be locked for editing, such as when working with data cards. You can select Unlock to change properties in such situations.



11 – Screen selector

Use the screen selector to switch between screens when your canvas app has multiple screens. You can also select a screen from the left pane by selecting the tree view. If the

current selection is inside a container, or inside an individual cell in a gallery, the selector shows the breadcrumbs for the parent elements at each level.

12 – Change canvas screen size

You can zoom in or zoom out while authoring the canvas app. Select **Ctrl+0 Fit to window** to fit the screen size based on the current authoring window size. The zoom percentage or screen size you use while authoring a canvas app has no impact on the aspect ratio configured for the app when you preview your app or play a published app.

See also

Use the Fluent UI controls
Overview of the Power Apps app

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Fluent UI controls

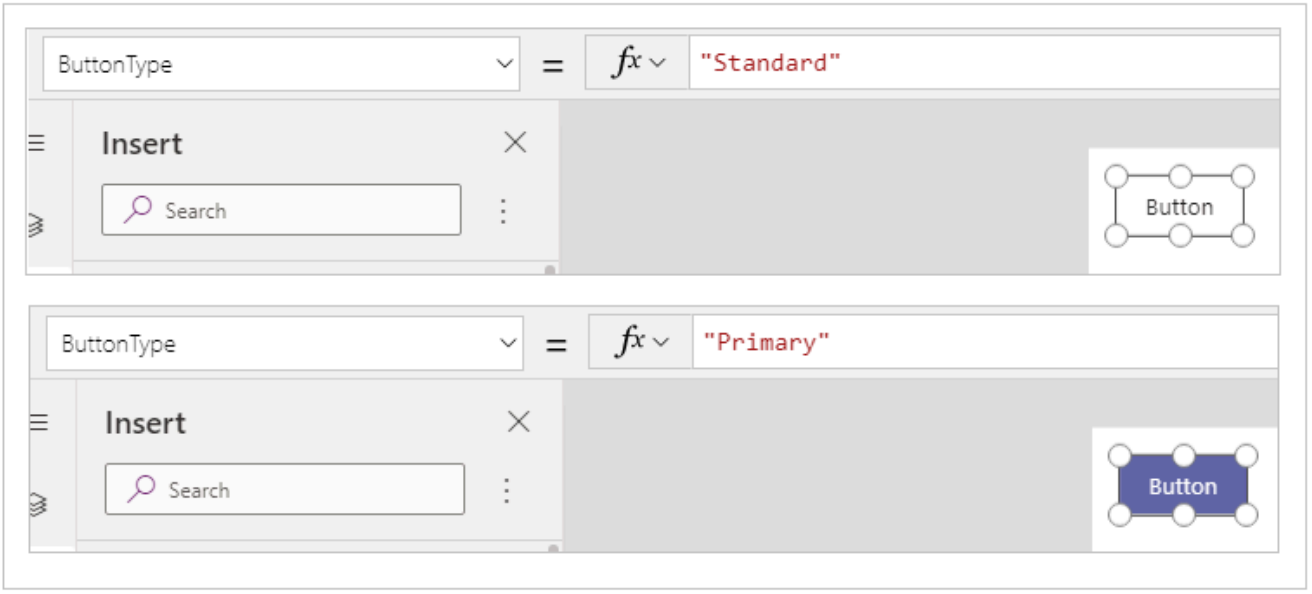
Article • 05/23/2025

Creating apps that look great in Microsoft Teams will be easier with our new components. Built on the Fluent UI framework, the new components will look great with Teams styles and will automatically adjust to the Teams default theme. The new controls are Button, Check box, Combo box, Date picker, Label, Radio group, Rating, Slider, Text box, and Toggle.

Let's take a look at each Fluent UI control and its most useful properties. For a complete list of controls and properties in Power Apps, go to Controls and properties in Power Apps.

Button

A control that the user can select to interact with the app.



Description

Configure the OnSelect property of a **Button** control to run one or more formulas when the user selects the control.

Key properties

ButtonType – The style of button to show, Standard, or Primary. Default value: **Standard**.

OnSelect – How the app responds when the user selects a control.

Text – Text that appears on a control or that the user enters into a control.

Check box

A control that the user can select or clear to set its value to **true** or **false**.



Description

The user can specify a Boolean value by using this familiar control, which has been widely used in the user interface.

Key properties

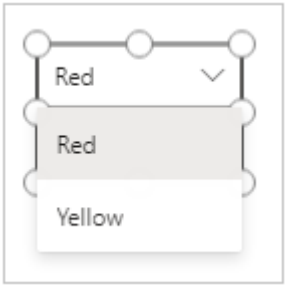
Box side – The side of the control on which the check box is displayed.

Label – Text that appears on a control.

Checked – Whether the control is checked or not.

Combo box

A control that allows users to make selections from provided choices. Supports search and multiple selections.



Description

A **Combo box** control allows a user to search for items to select.

Single-select or multiple-select mode is configured via the *SelectMultiple* property.

Key properties

Items – The source of data from which selections can be made.

DefaultSelectedItems – The items that are initially selected before the user interacts with the control.

SelectMultiple – Whether the user can select a single item or multiple items.

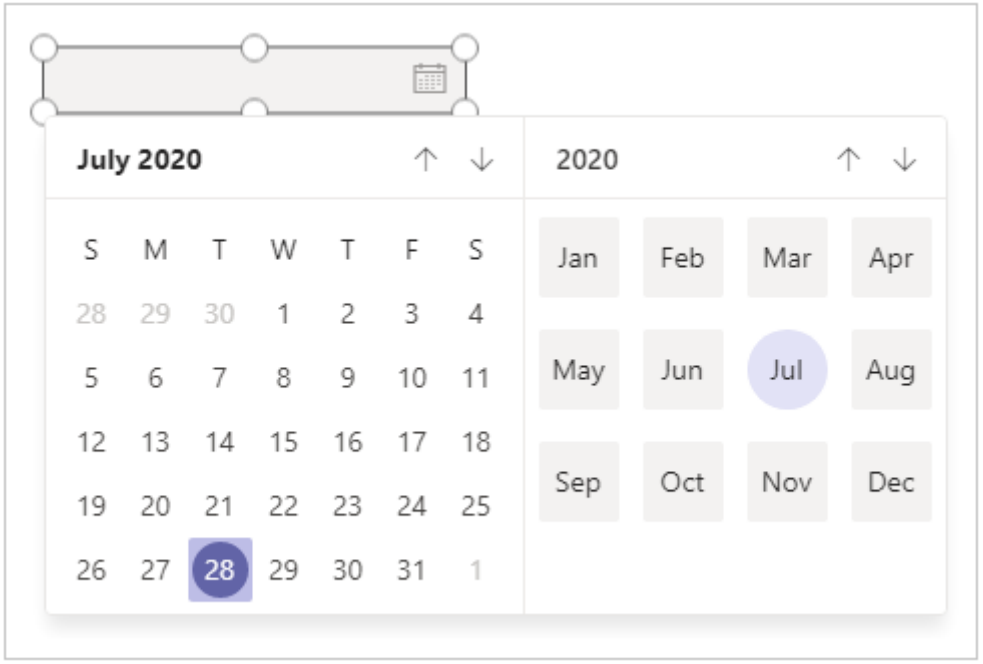
IsSearchable – Whether the user can search for items before selecting.

Limitations

1. The list of Options doesn't support Data Paging. It's limited to show the first 500 records.
2. Filtering the list of Options by typing in the Text Box area of the Control works inside the browser, not on the Server. This, coupled with the previous limitation, means that Filtering will not match any Option(s) beyond the first 500 entries, even when the Text matches.

Date picker

A control that the user can select to specify a date.



Description

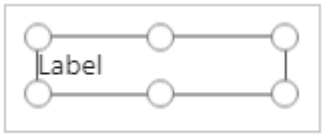
If you add a **Date picker** control instead of a text input control, you help ensure that the user specifies a date in the correct format.

Key properties

Value – The date currently selected in a date control. This date is represented in local time.

Label

A box that shows data such as text, numbers, dates, or currency.



Description

A label shows data that you specify as a literal string of text or as a formula that evaluates to a string of text. Labels often appear outside of any other control (such as a banner that identifies a screen), as a label that identifies another control (such as a rating or audio control), or in a gallery to show a specific type of information about an item.

Key properties

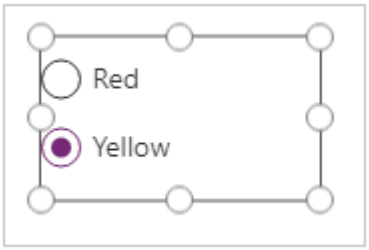
Color – The color of text in a control.

Font – The name of the family of fonts in which text appears.

Text – The text that appears on a control.

Radio group

An input control that shows multiple options, only one of which users can select at a time.



Description

A **Radio group** control, a standard HTML input control, is best used with only a few, mutually exclusive options.

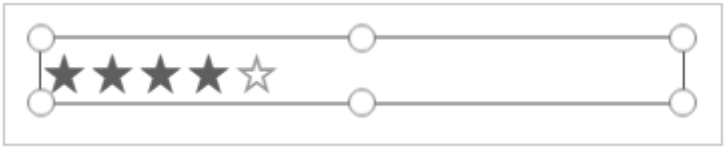
Key properties

Items – The source of data that appears in a control such as a gallery, a list, or a chart.

Selected – The data record that represents the selected item.

Rating

A control with which users can indicate a value from 0 through a maximum number that you specify.



Description

In this control, the user can indicate, for example, how much they liked something by selecting a certain number of stars.

Key properties

Value – The initial value of a control before it's changed by the user.

Max – The maximum value to which the user can set a slider or a rating.

Slider

A control with which the user can specify a value by dragging a handle.



Description

The user can indicate a value, between a minimum and a maximum value that you specify, by dragging the handle of a slider left to right or up and down, depending on the direction that you choose.

Key properties

Max – The maximum value to which the user can set a slider or a rating.

Min – The minimum value to which the user can set a slider.

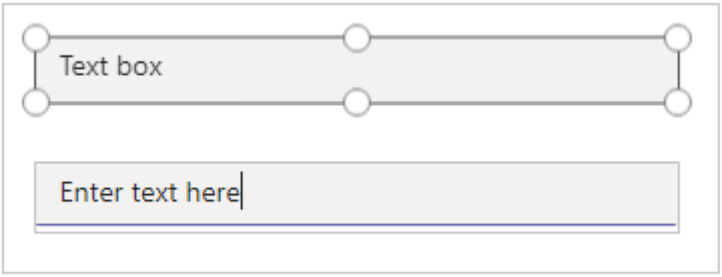
Value – The value of an input control.

Layout – Whether a control is displayed horizontally or vertically.

Show value – Whether a control should display the value.

Text box

A box in which the user can enter text, numbers, and other data.



Description

The user can specify data by typing into a **Text box** control. Depending on how you configure the app, that data might be added to a data source, used to calculate a temporary value, or incorporated in some other way.

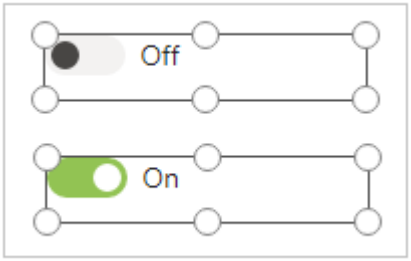
Key properties

Font – The name of the family of fonts in which text appears.

Text – Text that appears on a control or that the user enters into a control.

Toggle

A control that the user can turn on or off by moving its handle.



Description

A **Toggle** control is designed for the modern user interface, but behaves the same way as a check box.

Key properties

Checked – Whether the control is checked or not.

OffText – The text of the off state.

OnText – The text of the on state.

Difference between Fluent UI and classic controls

Properties of controls have been simplified for the ease of use. The following table lists differences between Fluent UI controls and classic control property names.

Expand table

Control type	Classic	Fluent UI
Button	Fill Background TabIndex	ColorFill ColorBackground AcceptsFocus
Check box	Default TabIndex	Checked AcceptsFocus
Combo box	InputTextPlaceholder TabIndex	Text AcceptsFocus
Date picker	SelectedDate TabIndex	Value AcceptsFocus
Label	Size	FontSize

Control type	Classic	Fluent UI
Radio group	TabIndex	AcceptsFocus
Rating	Default TabIndex	Value AcceptsFocus
Slider	Default TabIndex	Value AcceptsFocus
Text box	Color Default Fill SpellCheck TabIndex	ColorText Value ColorBackground EnableSpellCheck AcceptsFocus
Toggle	Size Default TabIndex	FontSize Checked AcceptsFocus

See also

Create additional apps
Understand Power Apps Studio

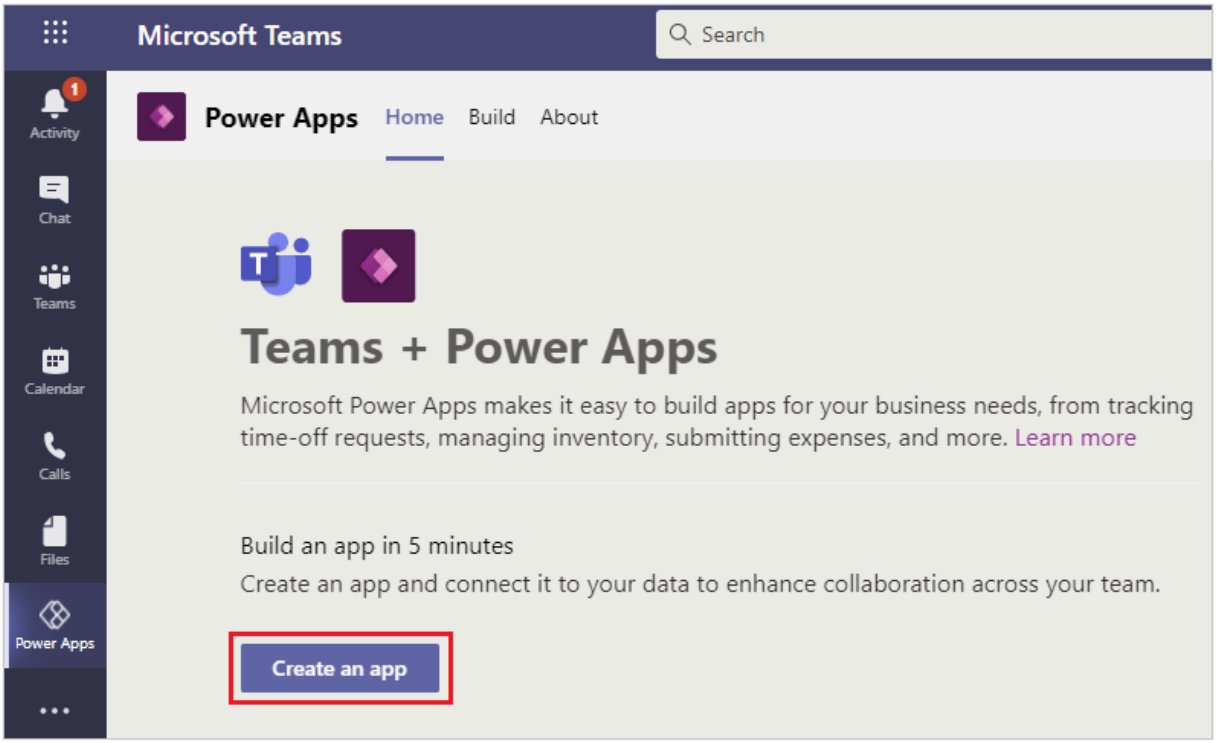
Create additional apps from Microsoft Teams

Article • 12/16/2022

Using the Power Apps app to create the first app in a team creates an environment. After an environment is available, you can create additional apps in the team by using several methods. Let's take a look at them.

Create an app from the Power Apps home

Select **Create an app** from the Power Apps app to get started with creating an app.

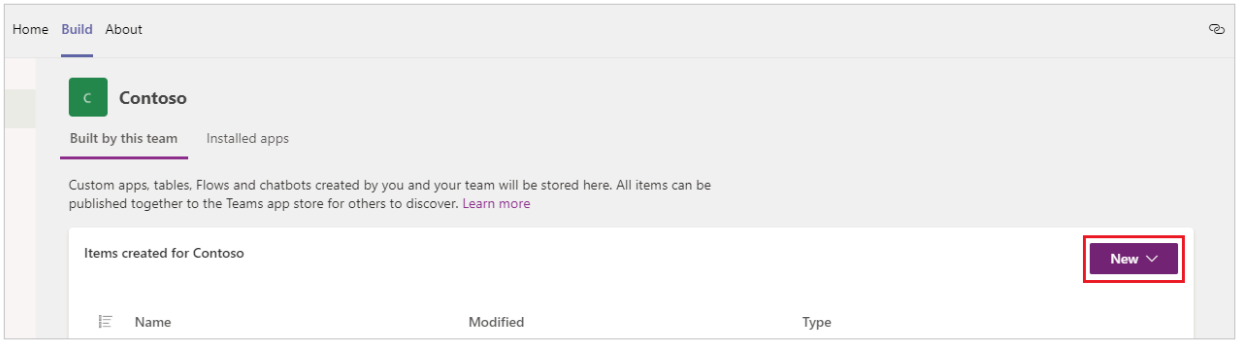


When the team that you select already has an environment available, you're not prompted to install an environment. Instead, you can select **Create** to create the app.

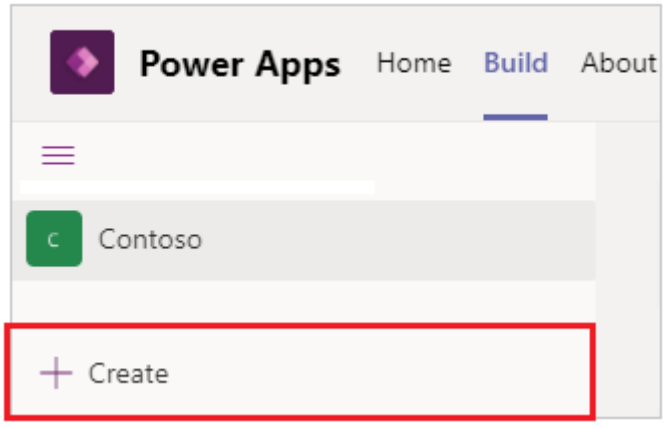


Create an app from the environment home

To create an app for the environment, you can also select the environment on the left pane, and then select **New**.



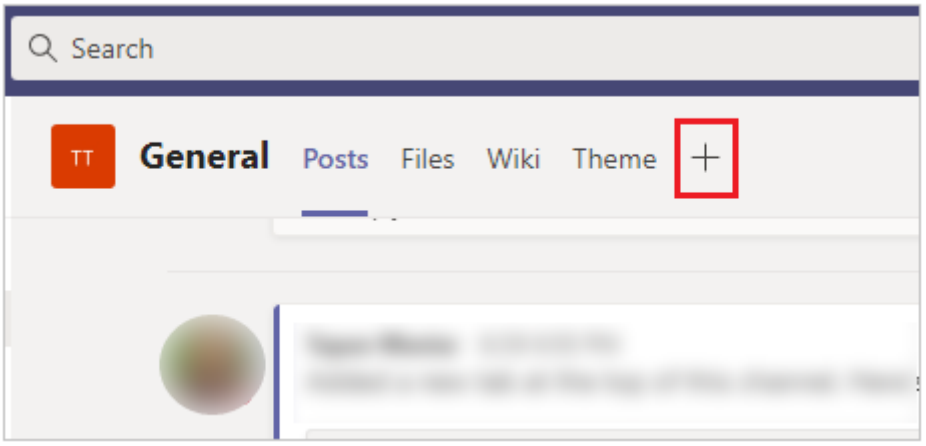
You can also select **Create** at the bottom of the environment list.



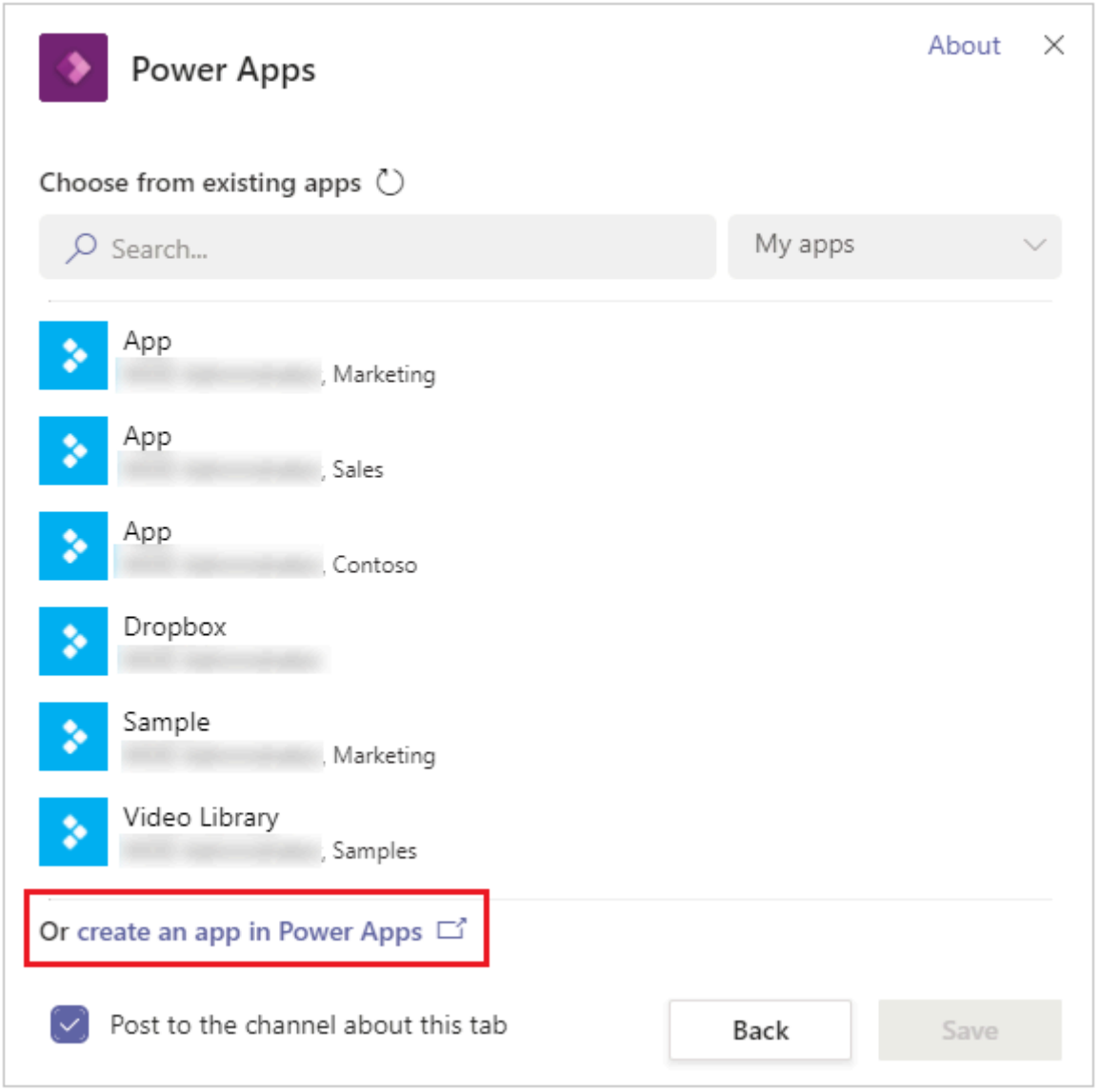
Create an app while inside a team

To create an app while you're inside a team, go to the team and select the channel.

Select **Add a tab** +.



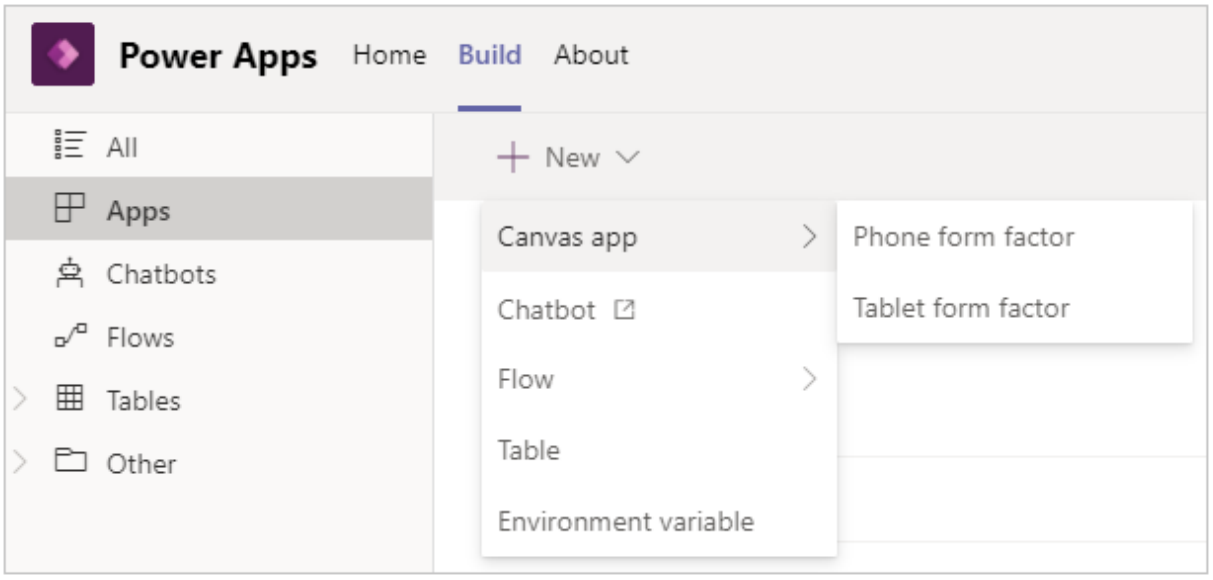
In the dialog box, select **Power Apps**, and then select **Create an app in Power Apps**.



Create an app from the solution explorer

You can also select **New** > **Canvas app**, and then select an app layout to create an app. A canvas app is an app that you can create from scratch or using any available template

through Power Apps Studio. More information: Canvas apps terminology reference and definitions



Tip

Creating a canvas app using different methods doesn't affect which app features are available. If you want to change the app orientation and size after you selected a layout, go to **Change screen size and orientation of a canvas app**.

See also

- Microsoft Teams integration object
- Use the Fluent UI controls
- Create apps from lists in Teams

Feedback

Was this page helpful? Yes No

Provide product feedback

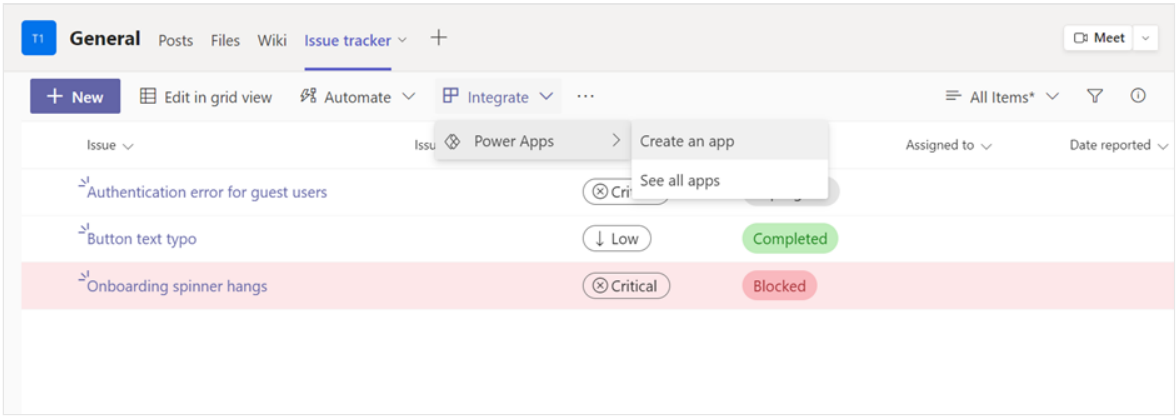
Create an app from a list in Microsoft Teams

Article • 12/16/2022

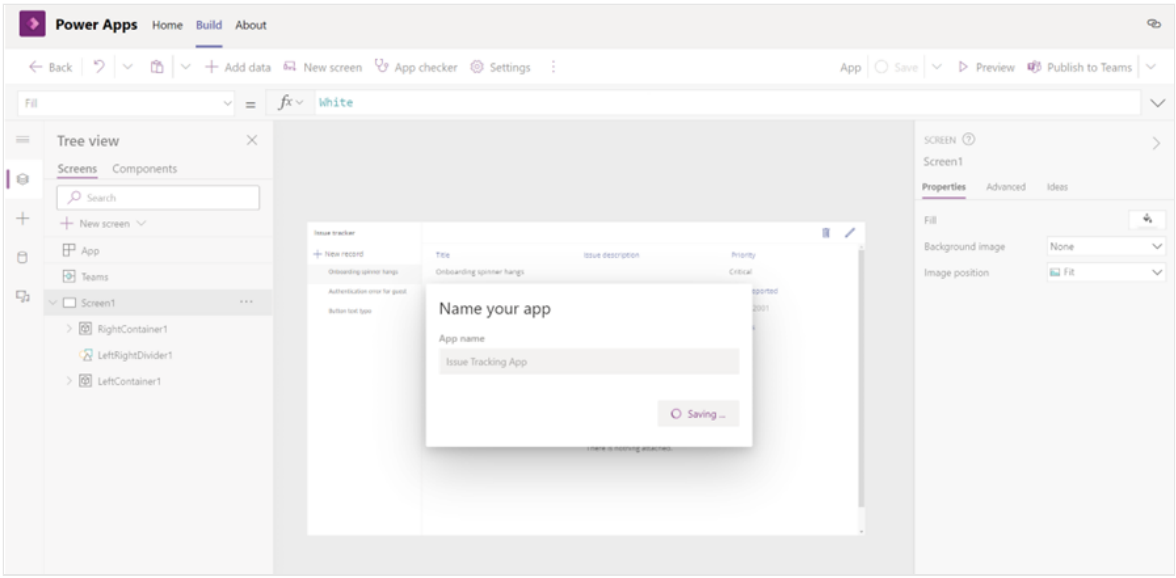
You can create an app from within a list in a Microsoft Teams channel.

> **Tip**
>
> To learn more about the **Lists** app in Teams, go to **Overview of the Lists app in Microsoft Teams**.

1. Create a list in a Teams channel.
2. From the list, select **Integrate** > **Power Apps** > **Create an app**.



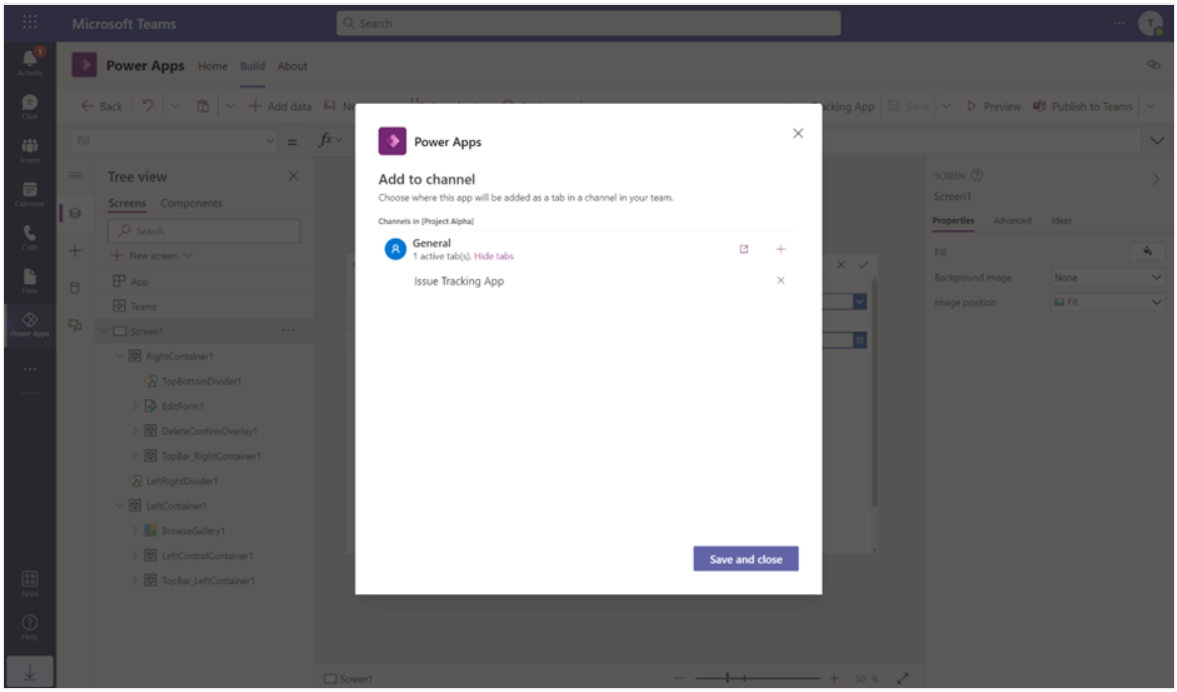
3. If you have never used Power Apps in Teams before, you'll be prompted to add the Power Apps app to Teams. Select **Add** to install the Power Apps app in Teams. More information: Install the Power Apps personal app in Microsoft Teams
4. In the modal that appears, enter a name for your app and then select **Save**.



The app will be created using the data from your list.

5. (Optional) Customize your app by adding and configuring controls.

6. (Optional) Select **Publish to Teams** to publish your app to a channel of any team that you have access to, including the team where your Microsoft list is located.



> **Note**
>
> Apps created in Teams are only accessible within Teams. You may view your apps created in Teams by selecting **Integrate** > **Power Apps** > **See all apps** from the command bar of a list in Teams or by going directly to the **Build** tab in Power Apps in Teams. In the **Build** tab, a link is available to navigate to the rest of your apps in Power Apps web.

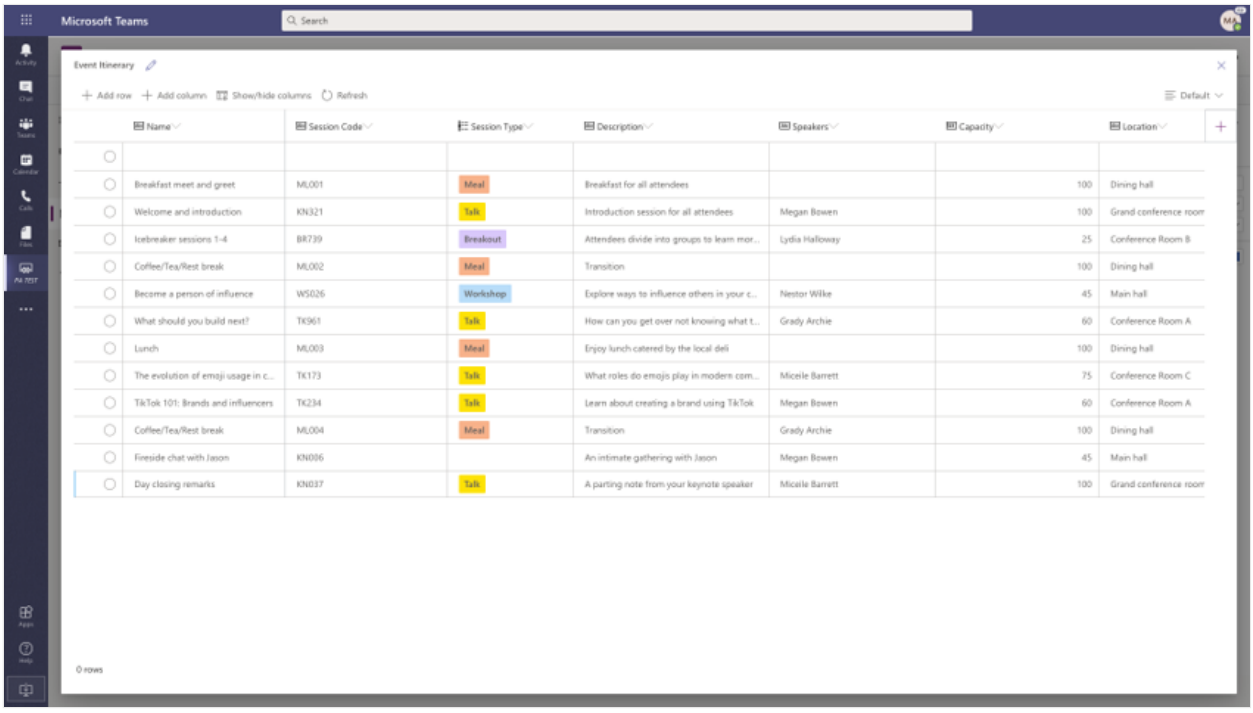
See also

- Create your first app in Microsoft Teams
- Create additional apps from Microsoft Teams

Overview of Microsoft Dataverse for Teams

Article • 02/16/2022

Microsoft Dataverse for Teams delivers a built-in, low-code data platform for Microsoft Teams. It provides relational data storage, rich data types, enterprise-grade governance, and one-click solution deployment. Dataverse for Teams enables everyone to easily build and deploy apps.



Some of the benefits of Dataverse for Teams include:

- Support to build low-code and no-code apps, flows, and chatbots for and within Teams.
- Core data capabilities from the same platform behind Microsoft Power Platform and Dynamics 365.
- Storage, rich data types with enterprise capabilities, and one-click solution deployment.
- A new visual editor that makes it even easier to define and populate table data.
- Enterprise security that's easy to use and aligned with the approach used in Teams.
- Inclusion in most existing Teams licenses.
- Storage of 2 GB per team, and support for up to 1 million rows.

- Support for 5 teams plus 1 additional team for every 20 licensed users.
- The capability to be upgraded to Dataverse.

See also: Dataverse for Teams FAQs

Tables in Dataverse for Teams

Dataverse for Teams tables provide the ability to create, populate, and query data within Dataverse for Teams. Tables represent different types of entities important to an organization, such as a table to store products or another to store orders.

Each of these tables includes columns that contain data about the subject of the table. For example, a table named *Product* might have columns that contain a product name, product identifier, manufacturer identifier, and price. Each of these columns can contain different types of data. For example, the type of data for a product name is text, identifiers could be numbers, and so on.

A solution often has multiple tables that are used together in an application. For example, an Order table might reference multiple other tables, such as Customer, Product, and Country. These tables are "related" to one another, and tables provide the way to create those relationships.

You can create and populate these tables with a new visual editor that makes it even easier to work with these tables.

> **Note**
>
> All the capabilities found in tables are powered by Dataverse for Teams. Although this will satisfy many situations, in some situations an organization might want to have additional capacity, capabilities, or control over their solution. In these scenarios, Dataverse for Teams environments can be upgraded to Microsoft Dataverse. The upgrade process has several considerations discussed in **Upgrade process** in the Microsoft Power Platform admin guide.

Security in Dataverse for Teams

These tables and the applications that use them are secured with enterprise-grade security that's easy to understand and use for low-code and no-code developers. Security in Dataverse for Teams aligns to how security is handled in Teams, with a focus on Owners, Members, and Guests.

Administer Dataverse for Teams environment

You can administer and manage Dataverse for Teams environment using the Power Platform admin center. More information: About the Dataverse for Teams environment in the Power Platform admin guide.

See also

How are Dataverse for Teams and Dataverse different?
Create tables
Work with table relationships
Dataverse for Teams FAQs

How are Dataverse for Teams and Dataverse different?

Article • 05/28/2025

Applications within a team in Microsoft Teams have access to a Dataverse for Teams environment. Dataverse for Teams is a common platform that allows all these interfaces to have a unified understanding of how the data is modeled and consumed. Dataverse for Teams delivers a targeted set of the features commonly needed for creating apps, flows, and more within Teams. If your organization requires more capabilities, such as more granular control for security and governance, or capacity beyond the approximately 1 million rows a Dataverse for Teams environment can contain, Dataverse for Teams can be upgraded to Dataverse.

Table features

Following are the differences between a Dataverse for Teams and Dataverse table features.

Expand table

Feature	Dataverse for Teams	Dataverse
Basic data types	Yes	Yes
Advanced data types (customer, multiple transaction currencies)	No	Yes
Relational storage	Yes	Yes
Non-relational storage (logs)	No	Yes
Managed data lake	No	Yes
File and image support	Yes	Yes
Find, filter, sort	Yes	Yes
Advanced and Dataverse search	No	Yes
Mobile offline	No	Yes

Most of the standard tables that are provided with a Power Platform environment won't be present as part of Dataverse for Teams.

Business intelligence, professional developer, and maker features

This table describes the differences between Dataverse for Teams and Dataverse business intelligence and professional developer features.

Expand table

Area	Feature	Dataverse for Teams	Dataverse
Business intelligence	Data visualization	Yes	Yes
	Paginated reports (SQL Server Reporting Services)	No	Yes
Professional developer	API access	No	Yes
	Plug-ins	No	Yes
	Power Apps component framework	No	Yes
Maker	Create canvas apps	Yes	Yes
	Create model-driven apps	No	Yes

Environments

In Dataverse for Teams and Dataverse, data is stored within an environment. Dataverse for Teams creates a single environment for each team in Teams where you create data, apps, chatbots, and workflows. Environments support backups, point-in-time restore, and disaster recovery. With Dataverse for Teams, capacity is measured with relational, image, and file data. The 2-GB capacity provided to a team can typically store up to 1 million rows of data.

To make management easier, the lifecycle of the Dataverse for Teams environment is connected to that of the associated team. For example, when a team is deleted, the associated environment is also deleted.

Whereas Dataverse for Teams focuses on one environment per team for up to 10,000 teams, Dataverse supports unlimited environments in addition to capabilities that are relevant for multiple environments, such as copy and reset.

Expand table

Environment lifecycle	Dataverse for Teams	Dataverse
Environments	1 per Team	Unlimited

Environment lifecycle	Dataverse for Teams	Dataverse
Maximum size	1 million rows or 2 GB	Unlimited
Upgrade to Dataverse	Yes	N/A

Security

Because collaboration in Teams happens with people inside and outside your organization, the security model needed to support this is easy to use. In Dataverse for Teams, access is preconfigured with a security role based on membership type such as owners, members, or guests.

Because Dataverse isn't specific to the Teams environment, it delivers more options for admin and user roles. It also includes many other security capabilities such as customer-managed keys, field-level security, hierarchical security, sharing, and support for legacy authentication.

> **Note**
>
> Because Dataverse for Teams environments are specific to a team, they only contain one business unit.

Expand table

Security feature	Dataverse for Teams	Dataverse
Admin roles	System Administrator and System Customizer	System Administrator and System Customizer and other service admin roles. More information: Use service admin roles to manage your tenant
User roles	Team owners, members, and guests	Several standard security roles and custom roles can also be created. More information: Security roles and privileges
Activity logging	No	Yes
Auditing	No	Yes
Business units	One	Unlimited
Customer-managed environment encryption key	No	Yes
Field-level security	No	Yes

Security feature	Dataverse for Teams	Dataverse
Hierarchical security	No	Yes
Record sharing	No	Yes
Create Owner Teams**	Yes	Yes
Create Microsoft Entra group Teams	No	Yes
Record sharing to Group Teams	No	Yes
Assign Teams Roles to Owner Teams[1]	Yes	Yes
Change record ownership**	Yes	Yes

[1] *Can be done via a custom app in Power Apps*

Integration

Integration with Dataverse for Teams is delivered primarily through connectors. Support for both standard connectors and the ability to use the Dataverse connector to connect to a Dataverse for Teams for a Teams environment is included. Users with Premium licenses have access to the full set of over 350 standard and premium connectors available. This allows you to bring data into or retrieve data from tables in Dataverse for Teams, execute workflows when data in those tables change, and also use data from tables in workflow logic.

Dataverse can also use connectors. Additionally, Dataverse includes other built-in integration capabilities that can export data to a data lake or publish events to an event hub, service bus, or by using webhooks. Dataverse also supports the TDS protocol, which provides integration with SQL Server.

Dataverse can also be used with server-side sync to synchronize with Exchange or POP3, and with Data Export Service, which synchronizes data to Azure SQL Database.

Expand table

Integration feature	Dataverse for Teams	Dataverse
Power Automate	Yes	Yes
Azure Synapse Link for Dataverse	No	Yes

Integration feature	Dataverse for Teams	Dataverse
Data Export Service	No	Yes
Events to Azure Event Hubs	No	Yes
Events to Azure Service Bus	No	Yes
Webhooks	No	Yes
Server-side sync	No	Yes
SQL Server Management Studio	No	Yes

See also

Create a table

Comparing Microsoft Lists, Dataverse for Teams, and Dataverse

Article • 10/26/2022

Whether you're using Power Apps to build a small app for your team or a mission critical app for your business, there are many great options for data.

This article focuses on three of the most popular data technologies used in Microsoft Power Platform: Microsoft Lists, Microsoft Dataverse for Teams, and Microsoft Dataverse. With the answers to the questions below, you can quickly understand and apply the key considerations that will help you pick the correct one for your application.

The questions are broken out into four categories:

Expand table

Category	Questions
Data	What types of data (and how much of it) will your application require? How do you want to search the data?
Application	How will the app be made available? For example, will it be a Teams app, custom code, or something else? Will guests be accessing your application? Who will build the app; low-code or pro developers? What special capabilities does your application need?
Integration	What do you want to integrate the system with? For example, databases, services, and so on?
Admin and governance	What are your organizations requirements related to security and compliance? Are there special requirements for backing up and restoring the data?

Key considerations and differences between Lists, Dataverse for Teams, and Dataverse

Using the answers to the questions above, use the table below to help identify the right technology for your application.

Expand table

Considerations	Lists	Dataverse for Teams	Dataverse
Types of data	Lists, File, Image	Relational, File, or Image	Relational, File, Image, Lake, Log, Dataverse Search Virtual tables
Number of data types	15	23 (Currency is basic version)	24 (Currency is advanced version)
Common Data Model	N/A	User table only	Full support
Capacity	Up to 30M rows (considerations for lists > 100k)	Up to 1M rows Small number of files or images For limits on API requests per day, go to the entitlements available for Microsoft 365 apps with Power Platform access in the article: Request limits based on user licenses	No specified limit on rows. No specified limit of files or images. For limits on API requests per day, with option of capacity add-ons, go to this article: Request limits based on user licenses
Data movement	Create from/Export to Excel	Dataflows In	Dataflows In/Out Server-side sync Synapse Integration (Bring Your Own Data Lake, Data Factory)
Security	Owners, Members, Visitors, Designers, Approvers roles Customizable permissions	Owner, Member, Guest roles Share app with Azure AD group	Robust options to satisfy complex enterprise scenario requirements, such as roles, business units, auditing, CMK, hierarchical/field-Level security, and so on
Clients	Lists, Teams[1], custom code	Teams	Teams, Power Apps, Power Pages, Dynamics 365, custom code
Guest limitations	Can't create or delete a list	Can't make, install, or edit apps	Must be in Azure AD using Azure B2B
Pro developer capability	REST API Graph API	N/A	REST API Software Development Kit (SDK) Plug-in Support Integration (Event Hub,

Considerations	Lists	Dataverse for Teams	Dataverse
			Service Bus, Webhook, Export to Lake) SQL Server Management Studio Integration
Package and deploy	Package and deploy Lists	Single unmanaged solution per environment	Unlimited
Additional capabilities	Calculations and rollups	N/A	Business workflows Business rules Calculations and rollups Mobile offline

[1]You can't create or pin lists on Power Apps mobile.

See also

Dataverse and Dataverse for Teams licensing
How are Dataverse for Teams and Dataverse different?
What is Dataverse?

Create tables in Microsoft Teams

Article • 07/02/2024

In Teams, data is defined by *tables*. To create a table, you must be a member of a team.

Types of tables

Whereas Dataverse has two types of tables (a standard table with ownership defined and an activity table), you can only create standard tables in Dataverse for Teams.

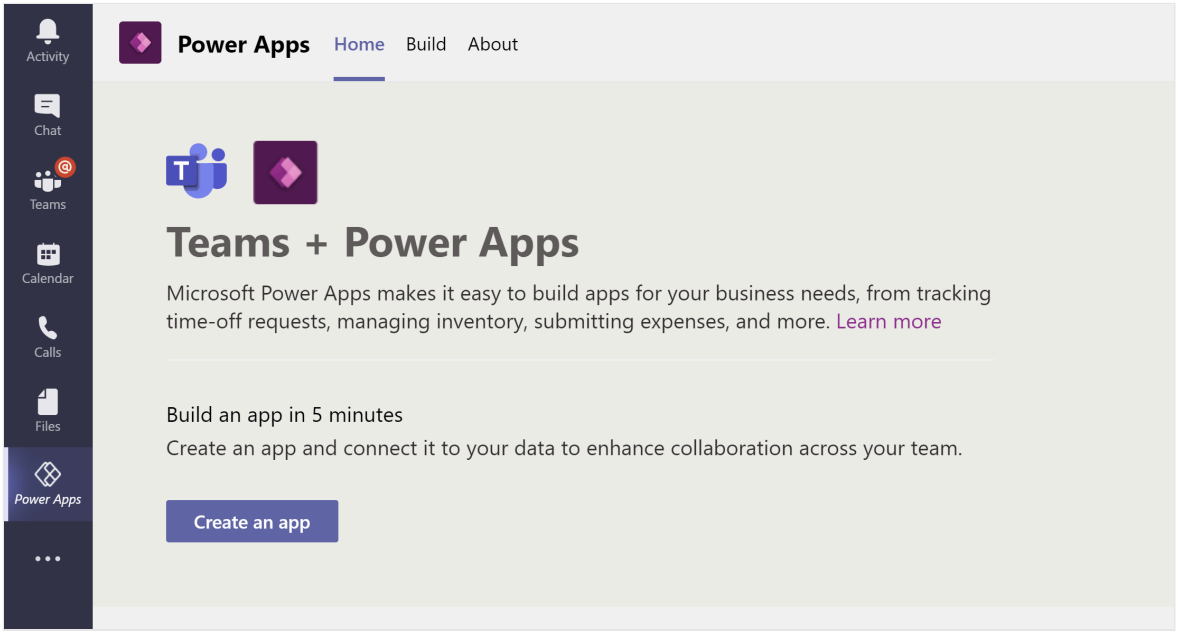
You define the type of table by selecting **More settings** when you create a table.

There are four different types of table ownership. When you create a table, the only options are **user**, **team owned**, or **organization-owned**, but you should be aware that other tables have different ownership types. More information: Types of tables in Dataverse

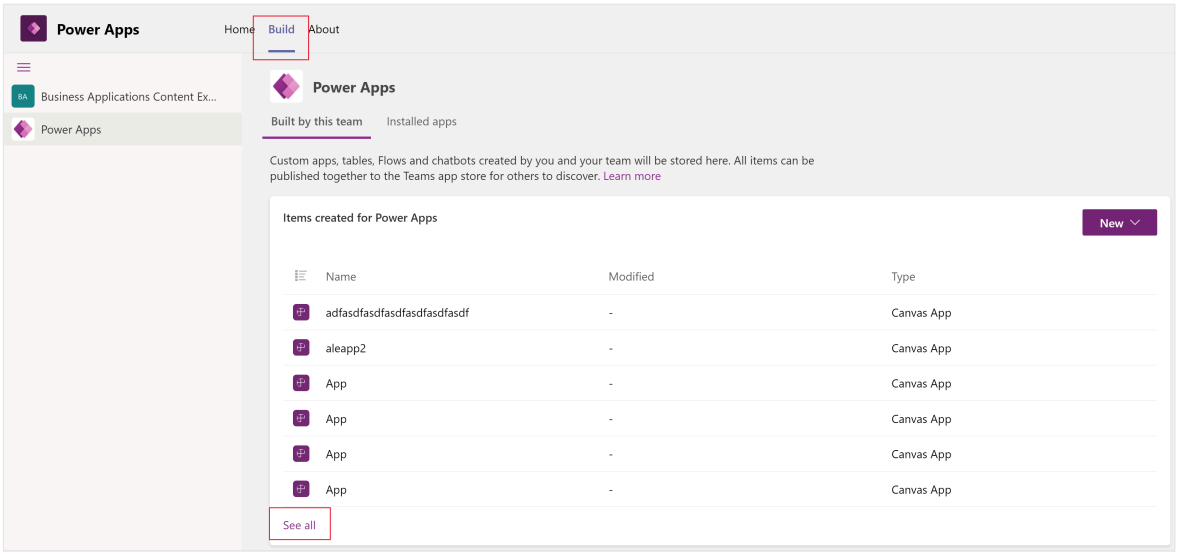
Create a table

Watch this video for a quick overview about how to create a table in Dataverse for Teams:
https://learn-video.azurefd.net/vod/player?id=a7b07bc4-1e08-4ed9-8675-b700a8e1a6a3&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fcreate-table

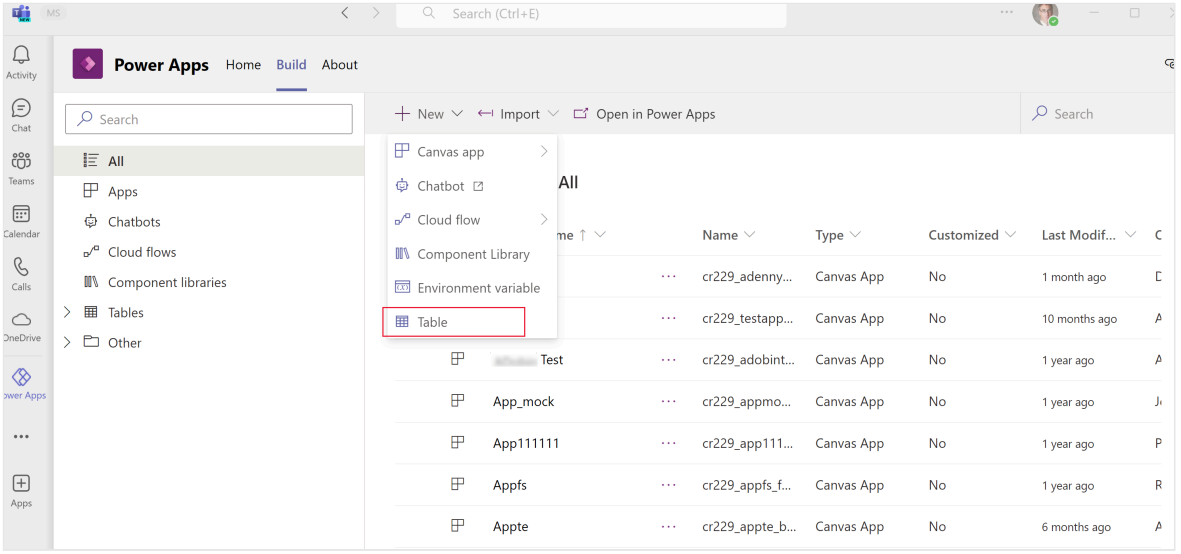
1. Sign in to Teams, and then select the link for **Power Apps**.



2. Select the **Build** tab, and then select **See all**.



3. Select **New** on the command bar, and then select **Table**.

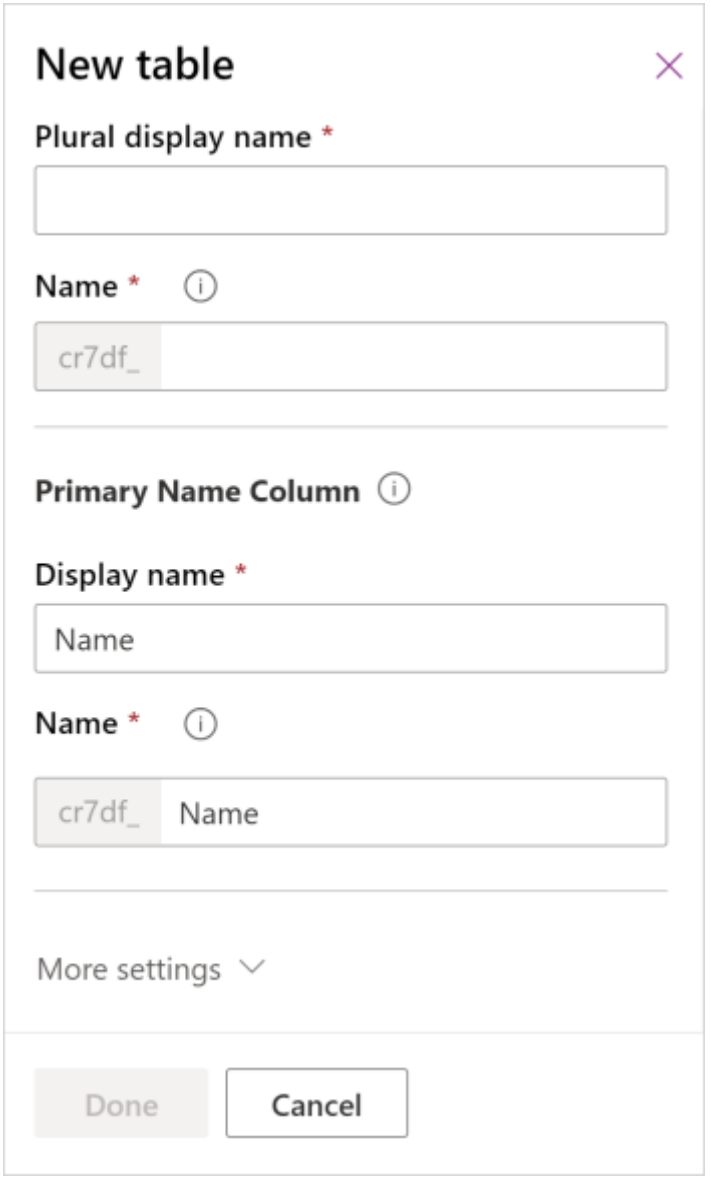


4. In the **New table** pane, enter the following information. Required columns are designated with an asterisk (*) next to the column label.

   - **Display name**: The name of the table to be displayed. For example, **Product**.
   - **Plural display name**: The name used to reflect multiples of what the table contains. For example, **Products**.
   - **Name**: A unique internal name for the table. Unlike the display name, after it's saved this value can't be changed.

   > ⓘ Note
   >
   > Each Dataverse for Teams environment is assigned a prefix when it's created, such as **cr628**. The name for every table and column you create will be prefixed with this value. This value can't be changed.

- **Display name** (Primary name column): Every table contains a primary name column, which is used by lookup fields when establishing relationships with other tables. Typically the primary name column stores the name or primary description of the data stored in the entity.
- **Name** (Primary name column): This column is pre-populated based on the primary name column name you enter. It includes the prefix. You cannot change this after the table is saved.
- **More settings**: This area provides additional settings that are available. More information: Create a custom table.



5. Select **Done**.

See also

Edit or delete a table

Feedback

Was this page helpful? Yes No

Provide product feedback

Edit or delete a table

Article • 02/15/2022

With Dataverse for Teams, you can perform the following types of operations on a table:

- Create table columns.
- Delete a table.
- Create, edit, and delete table relationships. More information: Work with table relationships

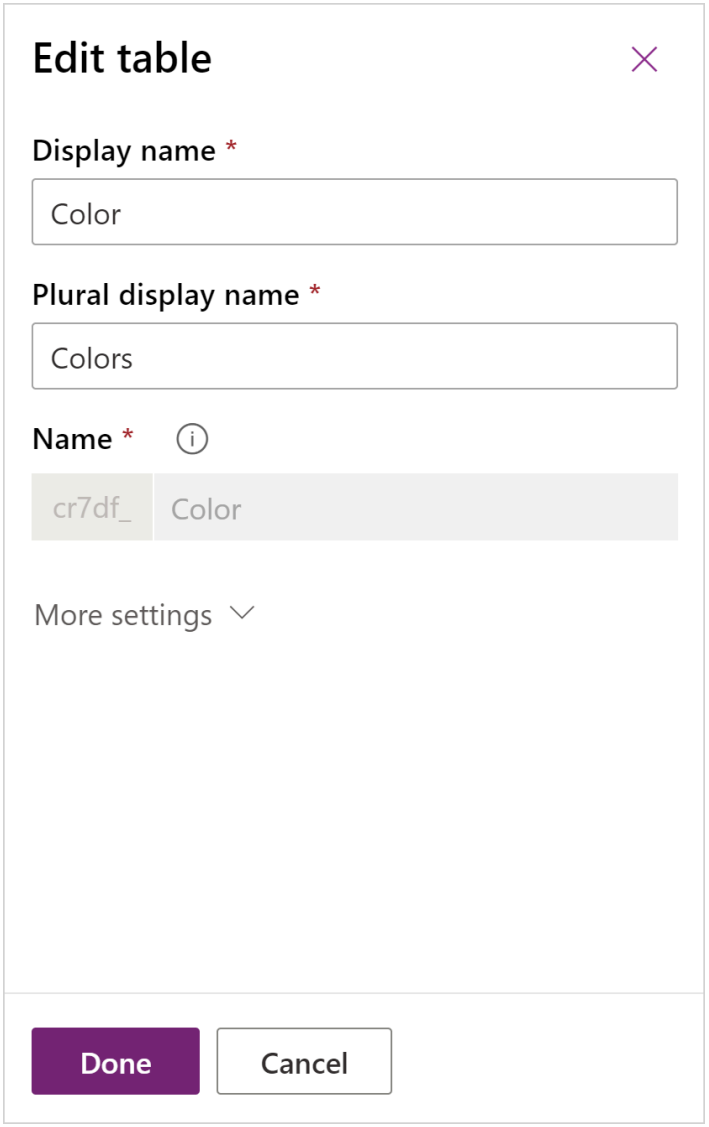
Edit table properties

In Microsoft Teams, you can edit four table properties:

- **Display name**: The name you want to display for the table.
- **Plural display name**: The plural of the table name, when needed for display.
- **Description**: A brief description of the purpose of the table.
- **Quick create settings**: This enables quick creation of records for client apps that support this feature.

> ⓘ Note
>
> - You can also edit a table using the visual editor. More information: Create your first table
> - Each Dataverse for Teams environment is assigned a prefix when it's created, such as **cr1a3**. The name for every table and column you create will be prefixed with this value. This value can't be changed.

1. On the left pane, select **Tables**.
2. Select the table you want and then select **Settings** on the command bar to display the **Edit table** dialog box.



For more information about these settings, see Create a table.

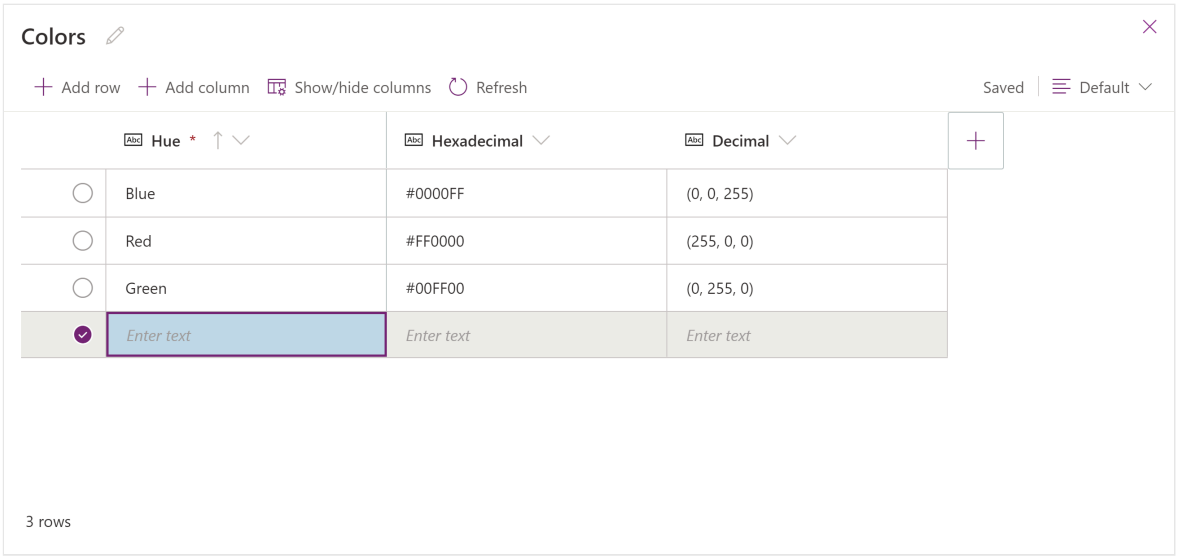
3. Make the changes you want, and then select **Done**.

Edit table columns

You can edit table columns using the visual editor.

1. On the left pane, select **Tables**.

2. Select the table you want, select **Edit data** on the command bar to display the visual editor.



More information: Create your first table

Delete a table

Anyone who's a member of the team can delete tables that aren't part of a managed solution.

In the **Tables** list, select **...** next to the table you want to delete, and then select **Delete table**. Alternatively, while viewing a table, select **Delete table** on the command bar.

See also

Work with table relationships

Feedback

Was this page helpful? Yes No

Provide product feedback

Work with table relationships

Article • 02/15/2022

Table relationships define the ways that table records can be associated with records from other tables or the same table. There are three types of table relationships. Dataverse for Teams allows for the creation of the same relationship types as Dataverse does:

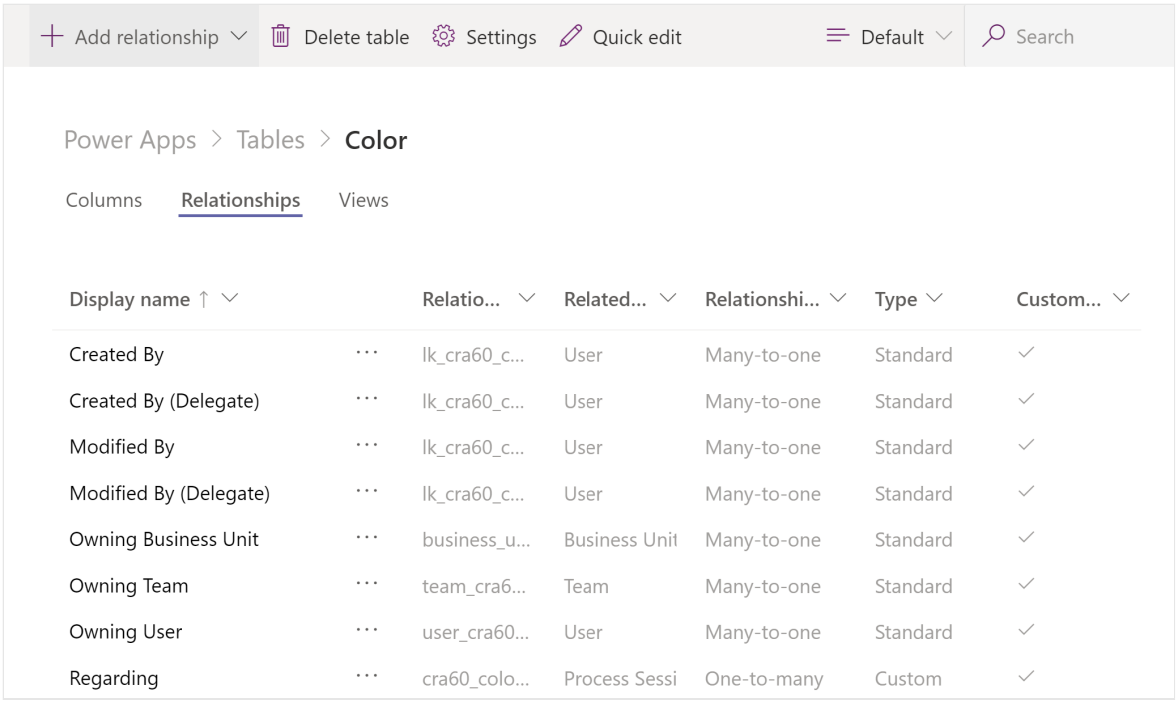
- **One-to-many relationships**. In this type of relationship, each row in table A can match more than one row in table B, but each row in table B can match only one row in table A. For example, a class of students have a single classroom.
- **Many-to-one relationships**. In this type of relationship, each row in table B can match more than one row in table A, but each row in table A can match only one row in table B. For example, one single teacher teaches many classes.
- **Many-to-many relationships**. In this type of relationship, each row in table A can match more than one row in table B, and vice versa. For example, students attend many classes, and each class can have multiple students.

More information: Entity relationships

Create a table relationship

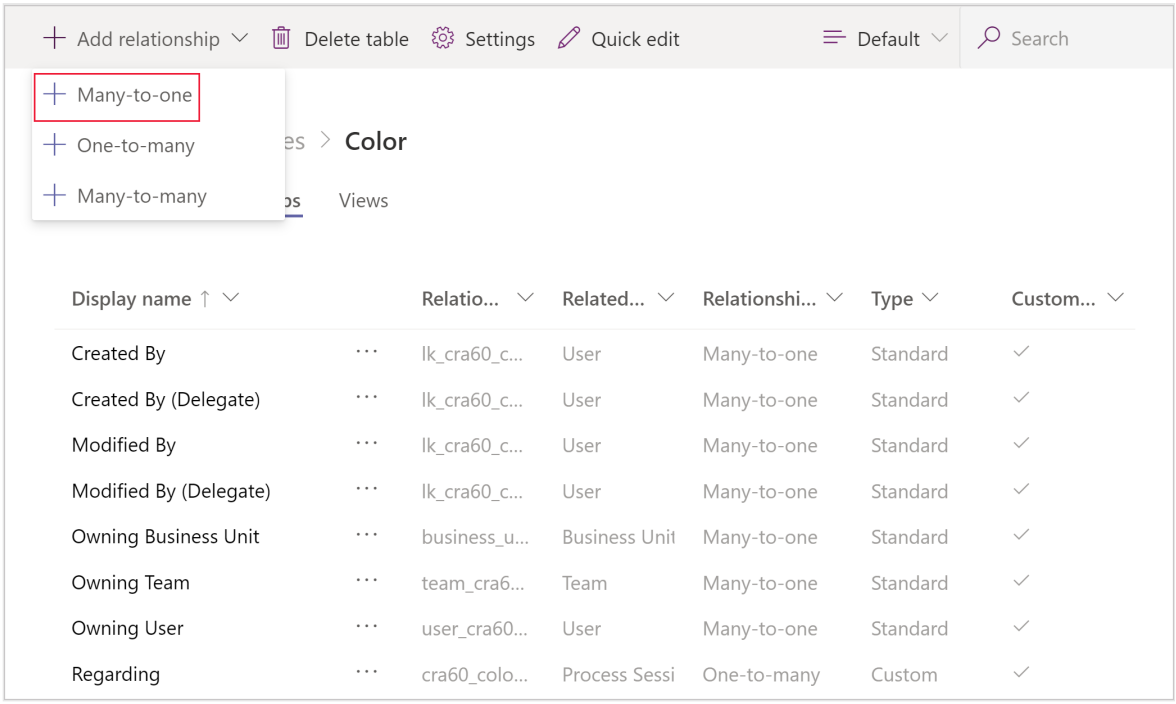
To add a lookup relation to a table, create a relation on the **Relationships** tab and specify the table with which you want to create a relationship.

1. In the **Power Apps** section of Microsoft Teams, select the table to create a relationship. This can be done by opening the table and selecting the **Relationships** tab.



Or, on the left pane, expand **Tables**, expand the table you want, and then select **Relationships**.

2. When the relationships page is open, select **Add relationship**, and then select the type of relationship you want to create (for example, **Many-to-one**).



More information: Create a relationship between entities

> **Note**
>
> When defining columns of type lookup within a table, a many-to-one relationship is created automatically.

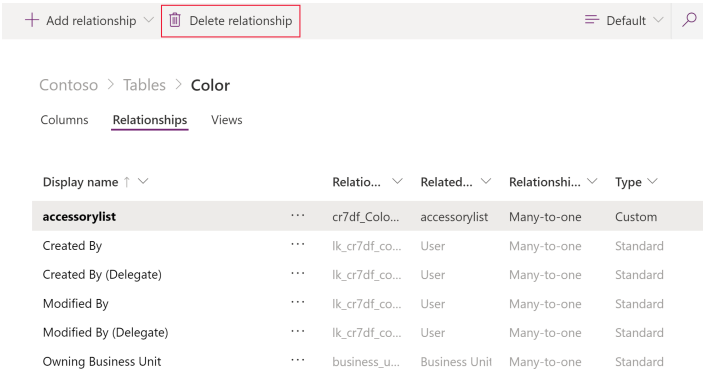
Edit a table relationship

Editing relationships in Dataverse for Teams works the same as Dataverse except that you can't manage collaboration functions, such as allowing feedback or enabling SharePoint document management. More information: Relationship actions

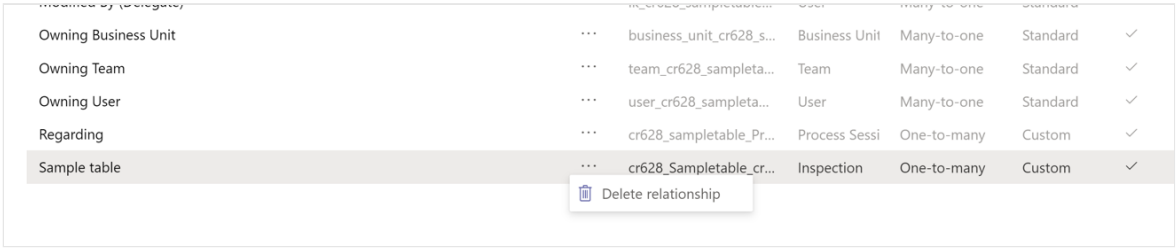
Delete a table relationship

While viewing table relationships, select the table relationship you want to delete, and then do one of the following:

- Select **Delete relationship** on the command bar.



- Select **...** next to the relationship, and then select **Delete relationship**.



See also

Work with table columns

Work with table columns

Article • 02/15/2022

With the exception of the Customer column, all column types in Dataverse are available in Dataverse for Teams as *columns*. This article covers the content you'll need for working with table columns in Dataverse for Teams.

Watch this video for a quick overview about column data types in Dataverse for Teams: https://learn-video.azurefd.net/vod/player?id=416d5037-a701-436d-bd77-e7fcb5329ead&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Ftable-columns

Notice that the currency feature will always use the default currency for the country/region that was selected during Dataverse for Teams environment creation. This can't be changed, and additional transaction currencies or exchange rates can't be added. However, you can upgrade from Dataverse for Teams to Dataverse for full currency functionality.

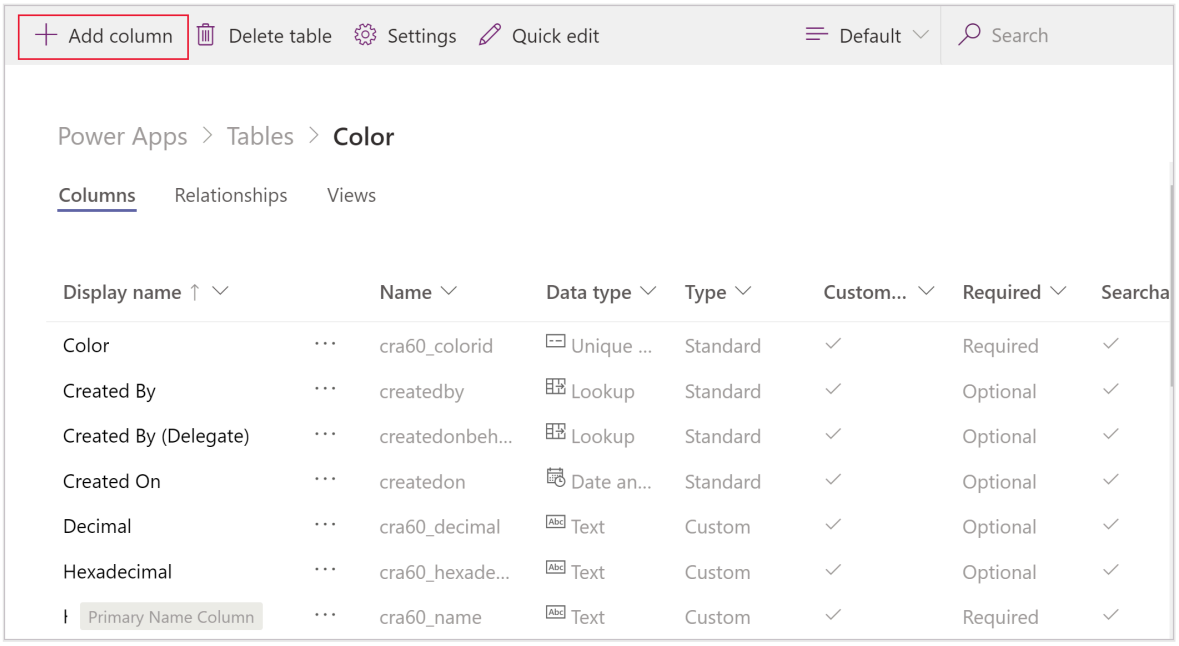
For more information about the columns available, see these Dataverse articles:

- Columns overview
- Create and edit global option sets overview
- Autonumber columns
- Set managed properties for columns
- Behavior and format of the Date and Time column

Create a column

Watch this video to learn how to create columns in Dataverse for Teams: https://learn-video.azurefd.net/vod/player?id=9a5cecc2-2fce-4ad5-adc9-0d68afb3426c&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Ftable-columns

1. On the **Build** tab, select **See all**, and then expand **Tables**.
2. Select the table you want to add a column to, and then select **Add column** on the command bar.



The rest of the process to add and manage columns is the same as in Dataverse, which is documented in these articles:

- Create and edit columns for Dataverse using Power Apps portal
- Manage custom columns in an entity

Choice columns

In Dataverse for Teams, choices can only be created as a column within a table. Creating choices is otherwise the same as creating an option set in Dataverse. More information: Create an Option set

Calculated and rollup columns

Calculated columns and rollup columns are not currently supported in Dataverse for Teams.

See also

Create a table

Feedback

Was this page helpful? Yes No

Provide product feedback

Work with formula table columns (preview)

Article • 01/01/2023

[This topic is pre-release documentation and is subject to change.] More information: Power Apps preview program

Formula columns are a data type in Microsoft Dataverse for Teams that are built on Power Fx. You can add a formula column to a table in real time. The Dataverse table stores the logic and gives you the values during fetch operations. Formula columns use the Power Fx syntax that's similar to Office Excel. As you enter the formula, Intellisense helps you with recommendations for formula, syntax, and errors.

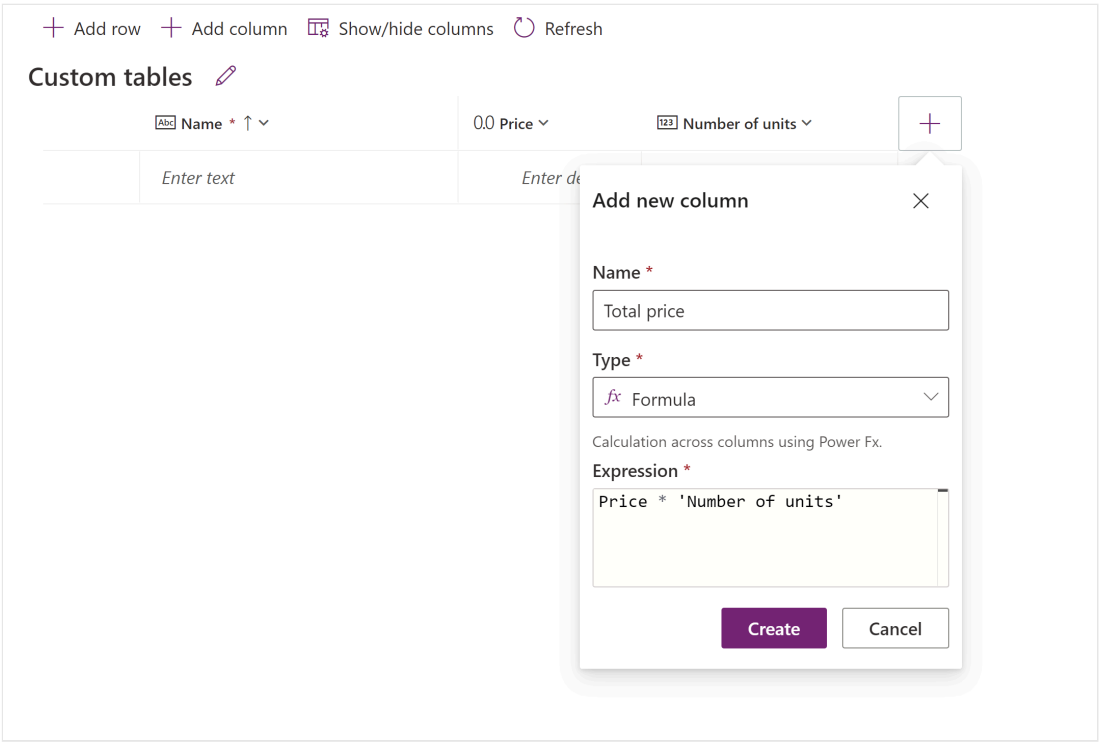
> **Note**
>
> Formula columns can be added as a calculated field. Currently, formula columns can't be used in roll-up fields or with plugins.

Add a formula column

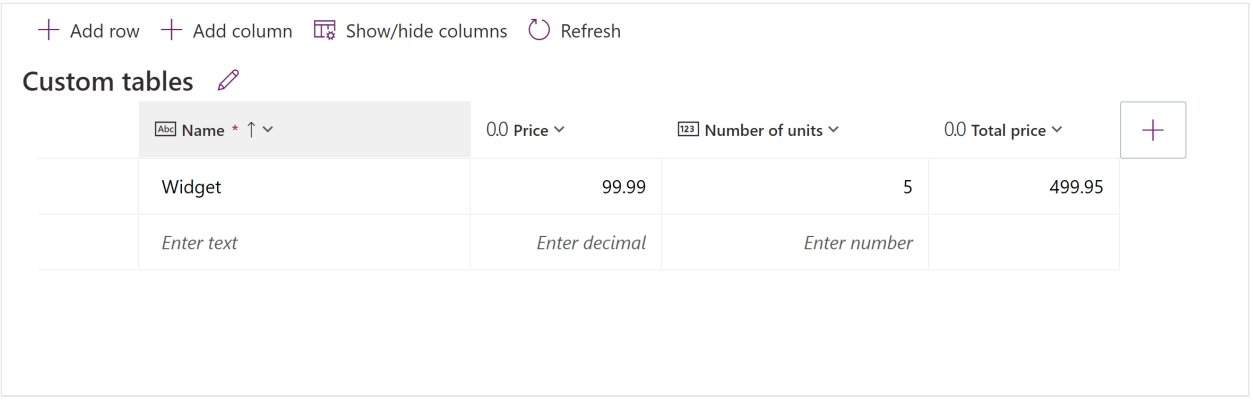
1. On the **Build** tab, select **See all**, and then expand **Tables**.
2. From the list of tables in Power Apps app for Teams, select the table you want.
3. Select **Add column** or select **Edit data** > **Add column**. When you select **Edit data**, you can also select + on the column where you want to add the formula column.
4. In the **Add new column** pane:

    - Enter a **Name** for the column, such as *Total price*.
    - Select **Formula** as the **Type**.
    - Enter the formula in the **Expression** box. In this example, the *Price* column (Decimal data type) is multiplied by the *Number of units* column (Number

data type). Select **Create**.



When you create a record, the formula column executes the formula and displays the data for the record. If the formula column value for a record doesn't update, select **Refresh** on the command bar to execute the formula.



Notice that the column type is determined by the formula. You can change the formula after you've created it if it doesn't change the column type. For example, the formula *price * discount* creates a column type of number, whereas the formula *First & " " & Last* creates a column type of string. You can change a *price * discount* formula to *price * (discount +10%)* because that doesn't change the column type. However, once saved, you can't change the *price * discount* formula to *Text(price * discount)* because that would require changing the column type to string.

Data types

The following data types can be displayed in a formula column:

- Text
- Decimal Number
- Yes/No (boolean)
- Date

> **Note**
>
> The Currency data type is not currently supported.

Function types

The following function types are supported with the formulas used in a formula column:

- Decimal
- String
- Boolean
- Option Set
- DateTime (TZI)
- DateTime (User local) (limited to comparisons with other user local values and the DateAdd function)
- DateTime (Date only) (limited to comparisons with other date-only values, and the DateAdd function)
- Currency
- Whole Number, promoted to Decimal

Operators

These operators are supported with the formulas used in a formula column:
+, -, *, /, %, ^, in, exactin, &

More information: Operators in Power Apps

Available functions

The following scalar functions are available with formula columns.

Abs

And

Average

Blank

Char

Concatenate

DateAdd

DateDiff

Day

EndsWith

Exp

Hour

If

IfError

Int

IsBlank

IsError

ISOWeekNum

IsUTCToday

Left

Len

Ln

Lower

Max

Mid

Min

Minute

Mod

Month

Not

Or

Power

Replace

Right

Round

RoundDown

RoundUp

Second

Sqrt

StartsWith

Substitute

Sum

Switch

Text *

Trim

Trunc

TrimEnds

Upper

UTCNow

UTCToday

Value *

Weekday

WeekNum

Year

* The **Text** and **Value** functions only work with whole numbers, where no decimal separator is involved. Since the decimal separator varies across locales, and the formula columns are evaluated without locale knowledge, there's no way to properly interpret or generate the decimal separator.

See also

Work with table columns
Formula reference for Power Apps

Feedback

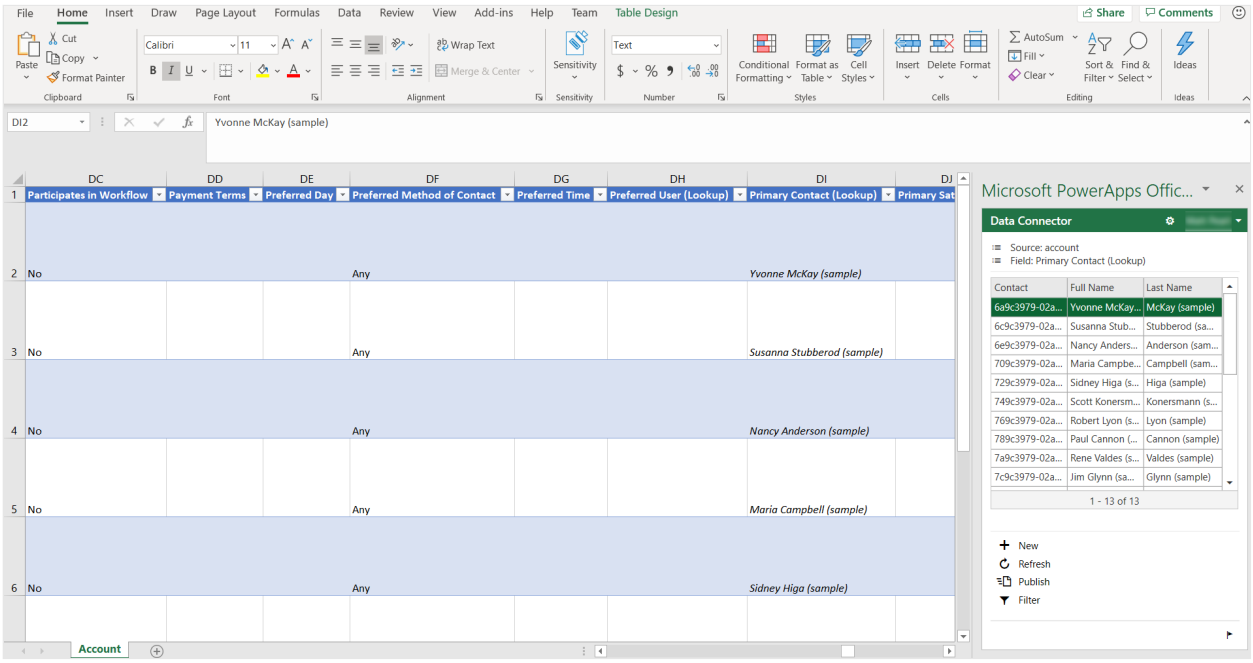
Was this page helpful? Yes No

Provide product feedback

Edit table data in Excel and publish it back to Dataverse for Teams

Article • 02/15/2022

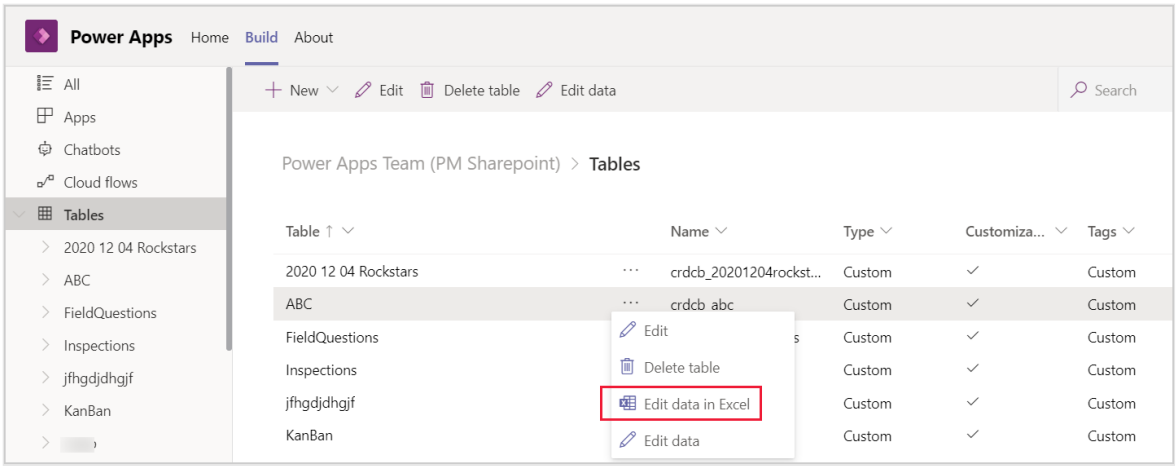
By opening Dataverse for Teams table data in Microsoft Excel, you can quickly and easily view and edit the data by using the Microsoft Power Apps Office Add-in.



To install the Power Apps Excel Add-in, see Microsoft PowerApps Office Add-in. For more information about how to add or remove an Office Excel Add-in, see Add or remove add-ins in Excel.

Open table data in Excel

1. Sign in to Teams, and then in the left pane, select the **Power Apps**.
2. Select the **Build** tab, and then select **See all**.
3. In the left navigation pane, select **Tables**, next to the table you want, select **...**, and then select **Edit data in Excel**.



4. Open the Excel worksheet that is downloaded to your browser's default download folder named similar to *crdcb_table-name (1591125669213).xlsx*.

5. In Excel, select **Enable editing** to enable the Power Apps Excel Add-in to run. The Excel Add-in runs in a pane on the right side of the Excel window.

> **Important**
>
> - If the pane displays an error message, see **Office Store Add-in download disabling**.
> - If this is the first time that you've run the Power Apps Excel Add-in, you must **Trust this Add-in** to allow the Excel Add-in to run.
> - The maximum table size when editing in Excel is one million cells. If there are too many rows or columns, not all data will be read or published.

Next steps

- View and refresh data in Excel
- Edit data in Excel
- Configure the add-in to adjust tables and columns
- Troubleshooting

Move customizations between environments

Article • 06/17/2022

Microsoft Dataverse for Teams provides a simplified way to transport customizations between Dataverse for Teams environments. Additionally, customizations can also be moved between Dataverse and Dataverse for Teams environments.

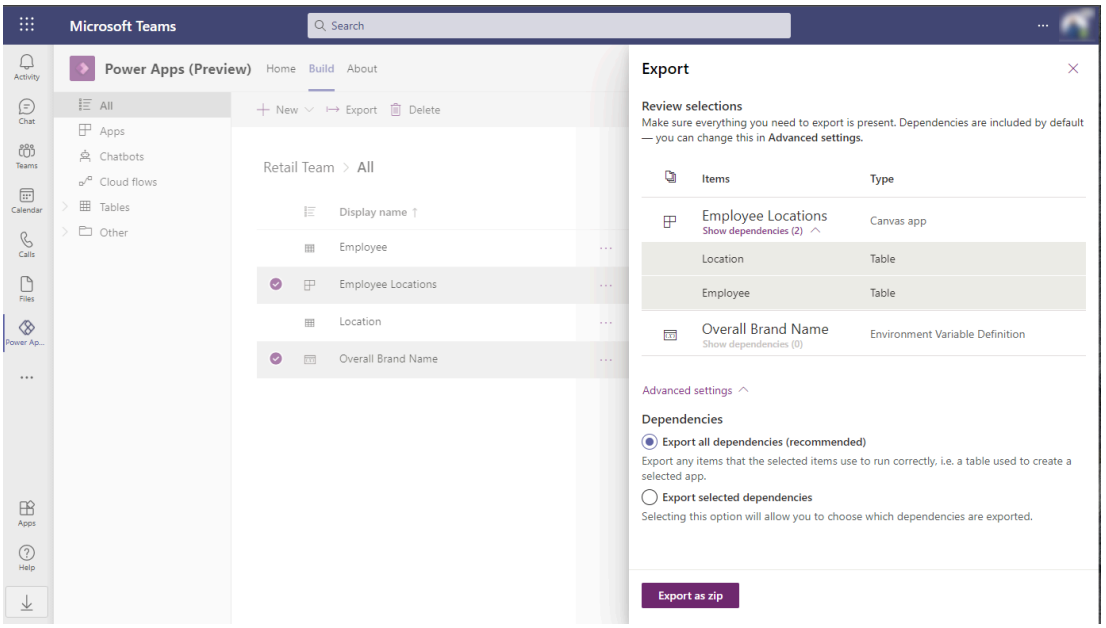
Export customizations from Dataverse for Teams

Take the following steps to export your customizations from Dataverse for Teams. Notice that the solution packages you export from Dataverse for Teams are unmanaged.

1. From the Power Apps personal app in Teams, select the **Build** tab of the team where you want to export your customizations. More information: Install the Power Apps personal app in Microsoft Teams
2. Select **See all**, and select the items you want to export.
3. On the command bar, select **Export**.
4. You can now review your selected items. By default, all dependencies of the items you selected are added too.

    - We recommend that you keep the option **Export all dependencies** under **Advanced settings** selected. By including all dependencies, you're optimizing your success when importing your customizations into another environment.

    - If you prefer to manually select the dependencies of your items, you can change the value to **Export selected dependencies** under **Advanced settings**. Importing customizations that don't include all dependencies will fail if those dependencies aren't already in the destination environment.



5. Select **Export as zip**. The selected items are exported as a compressed solution file (.zip). The solution file is automatically offered for download when the export completes.

    - Files you download from Microsoft Teams will show up in your device's downloads folder. In Microsoft Teams desktop, you can navigate to the **Downloads** section by select **Files** on the left side of the Teams client.

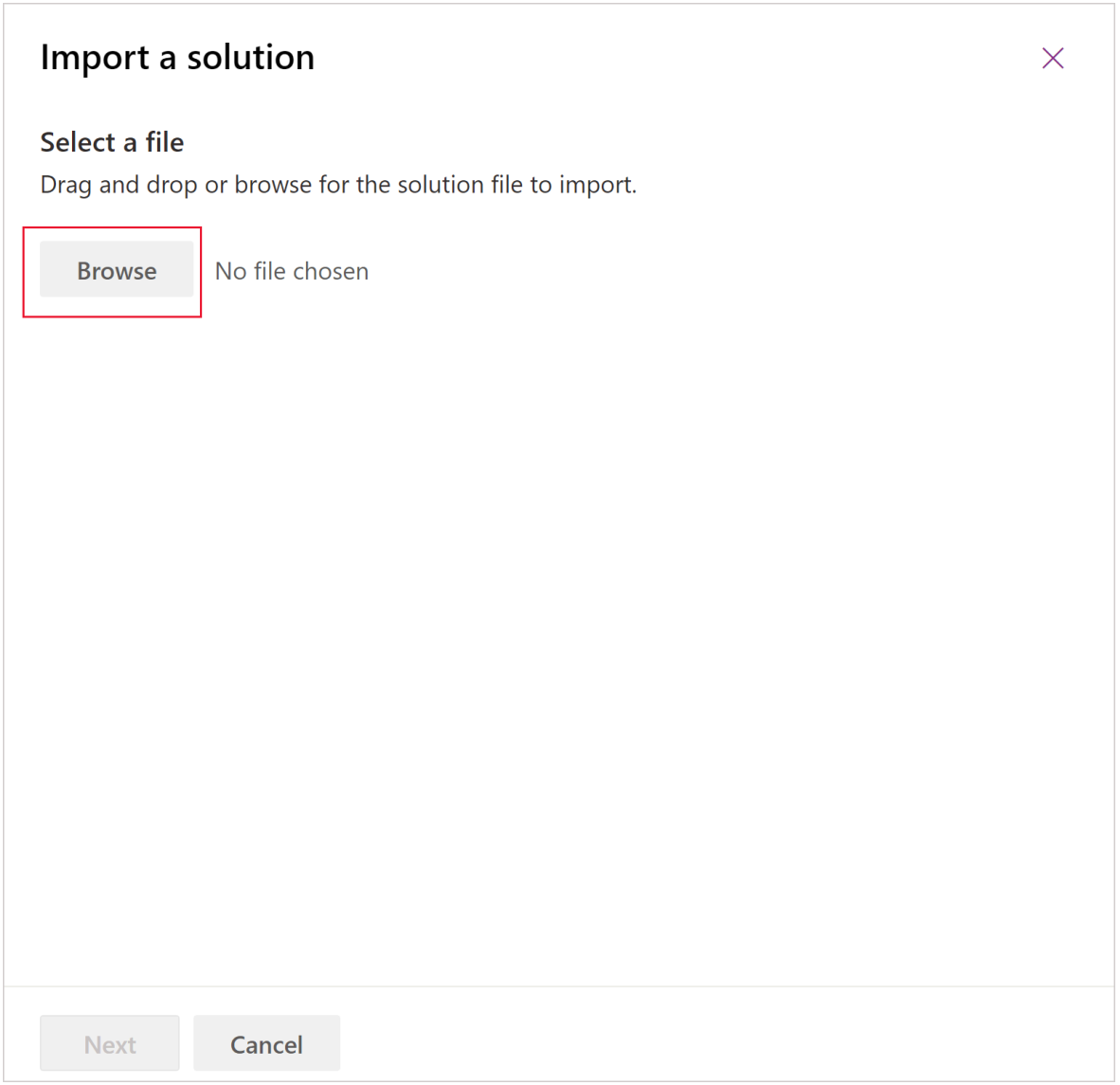
Import customizations into Dataverse for Teams

Take the following steps to import your customizations into Dataverse for Teams:

1. From the Power Apps personal app in Teams, select the **Build** tab of the team where you want to import your customizations. More information: Install the Power Apps personal app in Microsoft Teams

2. Select **See all**, and then on the command bar, select **Import**.

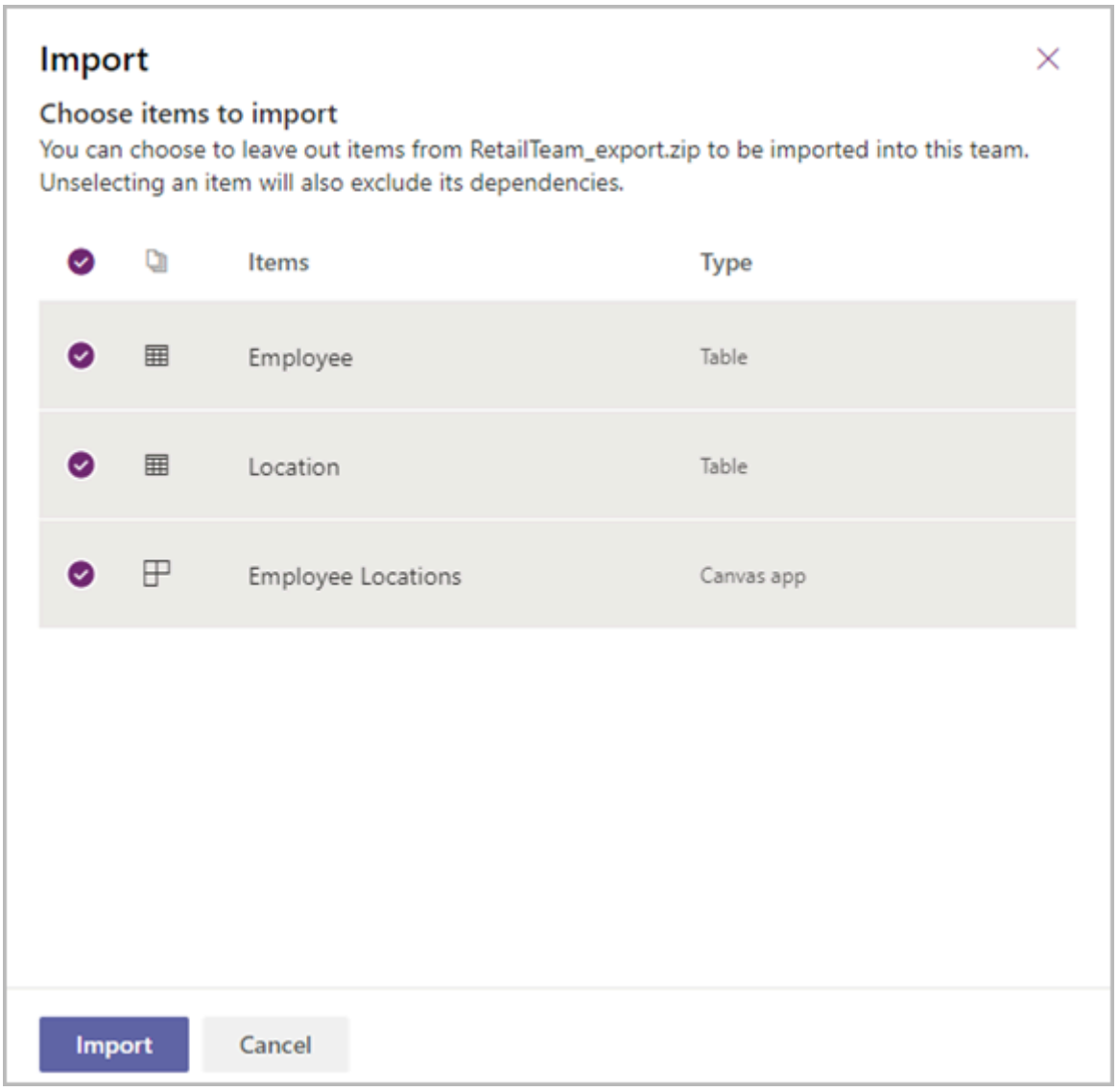
3. On the **Import a solution** page, select **Browse**.



4. Locate the compressed (.zip) file that contains the solution you want to import, select **Open**, and then select **Next**.

5. Review the items that you want to import.

    - If you're importing an unmanaged solution, you can select the item that you want to import into your team. We recommend that you keep the default selection and import all of the items. This helps ensure that all items and their dependencies are imported. Import will fail when you exclude an item that isn't already in the team and another item in the import is dependent on it.
    - If you're importing a managed solution, all the items will be imported.

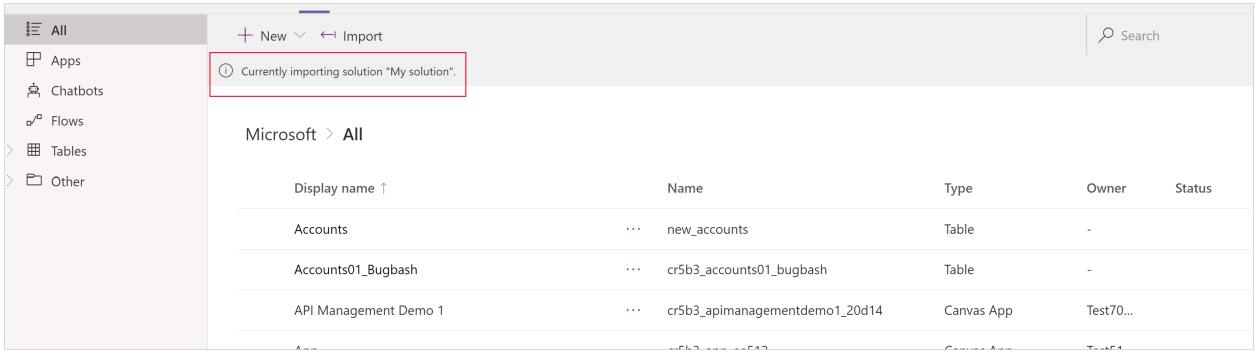


6. Depending on the components in the solution, you might be prompted for additional information when you select **Next**.

    - If your solution contains connection references, you'll be prompted to select the connections you want. If a connection doesn't already exist, create a new one.
    - If your solution contains environment variables, you'll be prompted to enter values. You won't see this page if values are already present in your solution or the target environment.

7. Select **Import**.

Your solution imports in the background and might take a while. A notification appears on the **Build** tab as the solution is imported.



After the import completes, a success or failure notification is displayed.

The location of your solution components is different depending on whether you imported a managed or unmanaged solution:

- With a managed solution, all your customizations appear on the **Build** tab under **Installed apps** using the name of the solution.
- With an unmanaged solution, all your customizations appear on the **Build** tab under **Built by this team**.

Move customizations from Microsoft Dataverse to Dataverse for Teams

Create a solution in a Power Platform environment

The first step to move your customizations from Microsoft Dataverse to Dataverse for Teams is to create a solution in Dataverse. Include only components that are fully supported for Dataverse for Teams. Solutions that include components that aren't supported by Dataverse for Teams won’t import.

The following custom components can be imported into Dataverse for Teams:

- Canvas apps
- Tables
- Flows
- Bots
- Connection references
- Environment variables

> **Note**

Only include items that you or your team have created. Solutions that contain any of the standard Dataverse or Dynamics 365 tables, such as account or contact, won't import.

For information about the differences between Dataverse for Teams and Dataverse, go to How are Dataverse for Teams and Dataverse different?

For information about how to create a solution with Dataverse, go to Create a solution.

Export the solution from Dataverse

Typically, you want to export your solution from Dataverse as managed. However, if your goal is to transport your customizations to Dataverse for Teams and continue your development from there, export your solution as an unmanaged solution. More information: Export solutions

You can now import your solution into Dataverse for Teams. More information: import customizations into Dataverse for teams.

Troubleshooting import failures

If the import isn't successful, you'll see a notification on the solutions page that displays error or warning messages that were captured. Select **Download Log File** to capture details about what caused the import to fail. The most common cause for an import to fail is that the solution didn't contain some required components.

When you download the log file, you'll find an XML file in your browser's default download folder that you can open using an XML reader, such as Microsoft Excel, to view the contents.

Known issues

When you're working with a new Dataverse for Teams environment, the **Import** command is missing because the **See all** link located on the **Build** tab isn't displayed for that environment.

To work around this issue, create and save an app in the environment. When an item exists in the environment, the **See all** link is available and you'll be able to browse to the **Build** tab.

See also

Export solutions

Feedback

Was this page helpful? Yes No

Provide product feedback

View Dataverse for Teams table data in Power BI Desktop

Article • 02/28/2023

You can use Power BI Desktop to view data from tables in Dataverse for Teams. This enables you to create sophisticated reports to analyze data from your Dataverse for Teams tables. This is possible using a new connector that can use direct query mode for real-time data while enforcing the Dataverse for Teams security model when accessing data. Additionally, the connector automatically maps relationships.

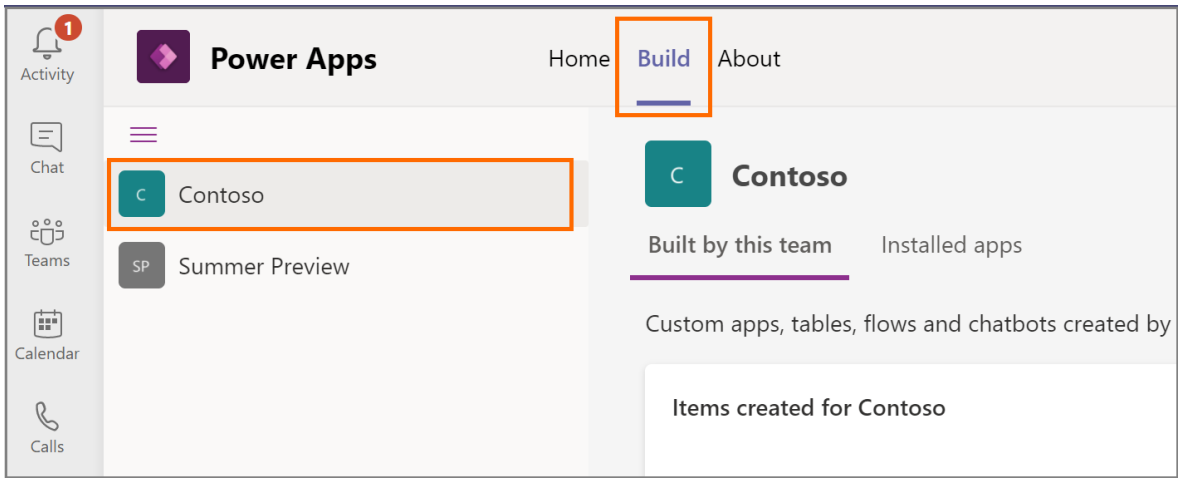
Prerequisites

You must have the appropriate Power BI license to build and share Power BI reports.

Step 1: Get the environment URL

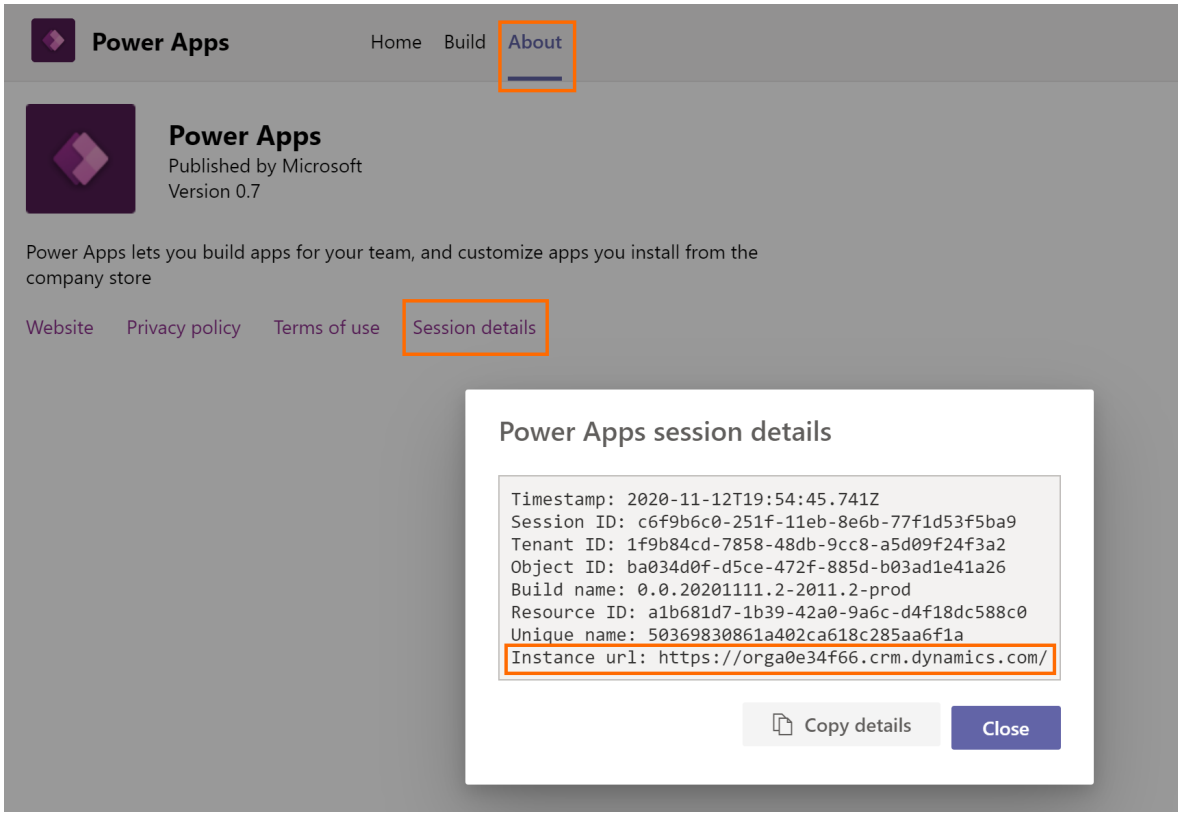
Follow the steps to get the URL of your Dataverse for Teams environment, which is required to connect from the Power BI Desktop.

1. Start Teams, and sign in.
2. In the left pane, select **Power Apps**, and then select the **Build** tab.
3. Select the team in the left pane that contains the data you want to view in Power BI.



4. Select the **About** tab, and then select **Session Details** to display the session details information.
   Copy the Instance url value to notepad. This is the environment URL of your

Dataverse for Team environment. For example:
`https://orga0e34f66.crm.dynamics.com/`



Step 2: View the table data in Power BI

1. Open Power BI Desktop, and sign in. Don't have Power BI Desktop? Get it now.

2. Select **Get data**.

3. On the **Get Data** page, select **Power Platform** > **Dataverse**. Select **Connect**.

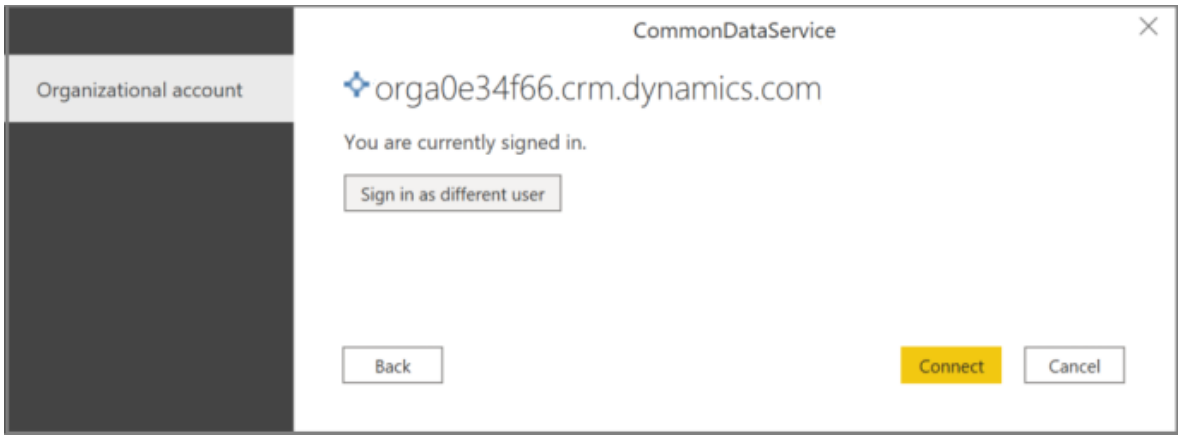
4. Read the preview disclaimer and select **Continue**.

5. On the next page, specify the environment URL that you copied earlier. Make sure you remove `https://` and the trailing `/` from the URL. So, in this case you will add `orga0e34f66.crm.dynamics.com`.
   Select the data connectivity mode and select **OK**.

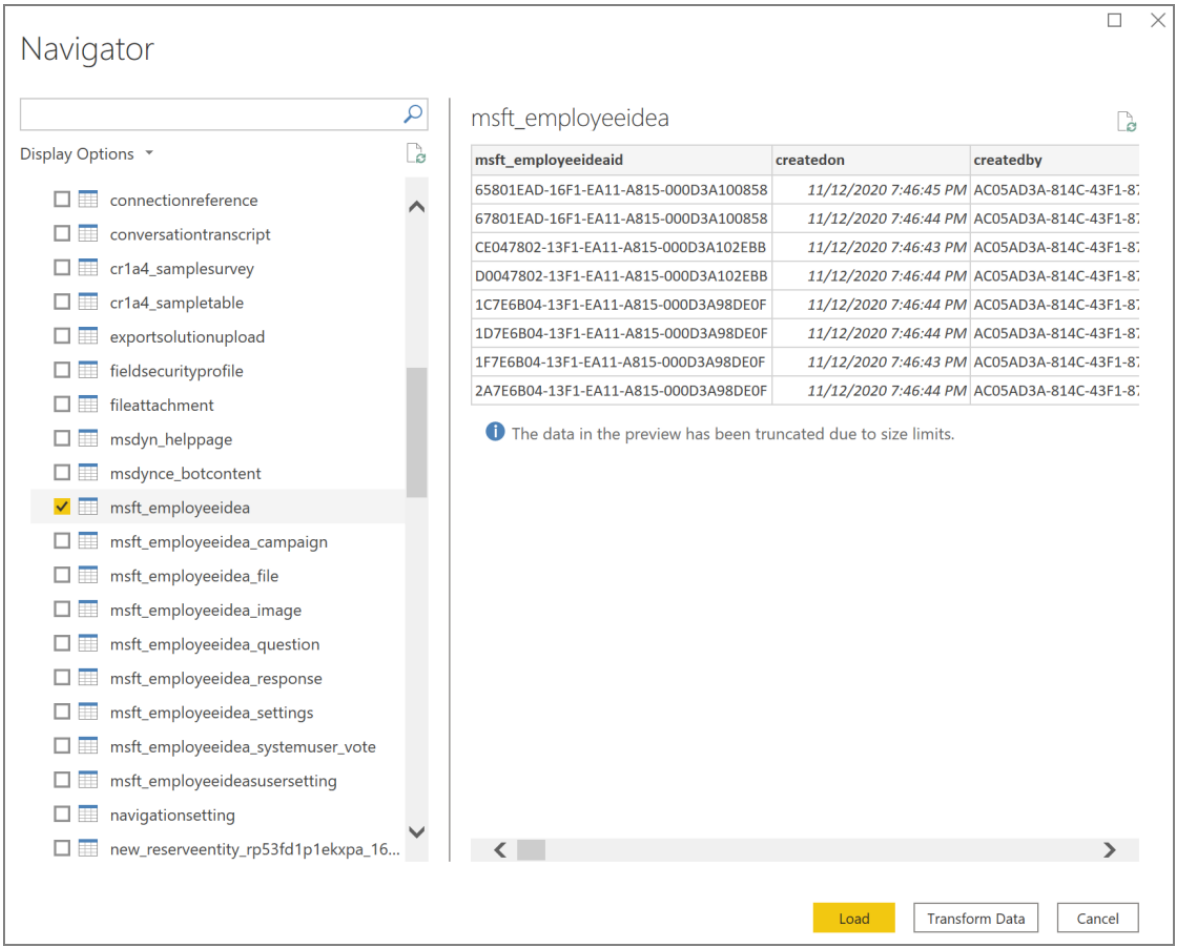


6. On the next page, select **Sign in**. You'll be prompted to specify your credentials to sign in to your Dataverse for Teams environment. Specify your credentials and submit.

7. On successful sign in, the **Connect** button becomes available. Select **Connect**.



The environment appears in the Power BI Desktop **Navigator** window. Expand it to view the tables available to analyze. Select a table to view its data.



For more information about working with Power BI Desktop, see Get started with Power BI Desktop.

See also

View Dataverse table data in Power BI Desktop (preview)
Dataverse for Teams FAQs

Dataverse for Teams environment language

Article • 02/15/2022

A Dataverse for Teams environment is automatically created when a Teams user *either* creates an app in a team using the Power Apps app for the first time *or* installs an app (created using Power Apps app) in a team for the first time from the Teams app catalog. During the environment creation, the environment's language is set to the tenant default language.

The language settings for the environment define the language, sorting, and search formats of the tables in the environment. The *default* language set during the automatic creation of the environment can't be changed thereafter.

When a user accesses the Power Apps app within a team in Microsoft Teams, the user's language setting is compared against the default language of the Dataverse for Teams environment for the team. Depending on the match of the languages, the following happens:

- If the user's language setting matches the default language of the environment, the default language is used to display existing information and for creating new tables and columns.
- If the user's language setting is different from the default language of the environment and the environment:
    - Doesn't have the required language pack installed, the new language pack matching the user's language setting will automatically be installed in the environment. It might take up to 15 minutes for the language pack installation to complete. This is a one-time installation, and will only occur the first time a new language is used.
    - Already has the required language pack installed, the environment will use the user's language.

    Also, this does not change the default language of the environment, but provides a localized user experience for the user when viewing the existing information in the team, and allows all new tables and columns created by the user to have localized labels that match their language.

> **ⓘ Important**

- Any existing tables or columns at the time of the language change will not be translated to another language. They will continue to use the existing names.
- In some cases, a language supported by Teams might not be available in Dataverse for Teams and other Power Platform features. When this happens, the nearest fallback language is used for the interface. More information: **Supported languages and fallbacks** later in this topic.

Example

User A's language is English (United States) in Teams and they create an app in a team called Team 1.

User B's language is set to French in Teams and they access the Power Apps app in Team 1.

- If the Dataverse for Teams environment doesn't have the French language pack installed, it will be automatically installed in the environment for Team 1. The user interface for User B immediately changes to French while the language pack is installed. After the language pack is installed, any new tables and columns will be created in the French language.
- If the Dataverse for Teams environment for Team 1 already has the French language pack installed, no additional installation is required.

If the User B changes their language in Teams again and accesses the Power Apps app in the team next time, the new language pack will be installed if the language isn't already installed in Dataverse for Teams environment.

Supported languages and fallbacks

The languages supported across Teams, Dataverse for Teams, and other Power Platform features vary slightly. When a language is not supported by Dataverse for Teams and other Power Platform features, these will use an identified fallback language. Here is a list of the languages supported across Teams, Dataverse for Teams, and other Power Platform features and their fallback languages.

The fallback languages are denoted in *italics*.

Teams	Dataverse for Teams	Power Apps Studio	Power Automate	Power Virtual Agents	Power BI
Arabic	Arabic	*English US*	*English US*	*English US*	Arabic

Teams	Dataverse for Teams	Power Apps Studio	Power Automate	Power Virtual Agents	Power BI
Brazilian Portuguese	Brazilian Portuguese	Brazilian Portuguese	Brazilian Portuguese	Brazilian Portuguese	Brazilian Portuguese
Bulgarian	Bulgarian	Bulgarian	Bulgarian	*English US*	Bulgarian
Catalan	Catalan	Catalan	Catalan	*Spanish*	Catalan
Croatian	Croatian	Croatian	Croatian	*English US*	Croatian
Czech	Czech	Czech	Czech	*English US*	Czech
Danish	Danish	Danish	Danish	Danish	Danish
Dutch	Dutch	Dutch	Dutch	Dutch	Dutch
English (Canada)	*English US*	*English US*	*English US*	*English US*	*English US*
English UK	*English US*	*English US*	*English US*	*English US*	*English US*
English US	English US	English US	English US	English US	English US
Estonian	Estonian	Estonian	Estonian	*English US*	Estonian
Flipino	*English US*	*English US*	*English US*	*English US*	*English UK*
Finnish	Finnish	Finnish	Finnish	*Swedish*	Finnish
French	French	French	French	French	French
French (Canada)	*French*	*French*	*French*	*French*	*French*
German Standard	German Standard	German Standard	German Standard	German Standard	German Standard
Greek	Greek	Greek	Greek	*English US*	Greek
Hebrew	Hebrew	*English US*	*English US*	*English US*	Hebrew
Hindi	Hindi	Hindi	Hindi	Hindi	Hindi
Hungarian	Hungarian	Hungarian	Hungarian	*English US*	Hungarian
Icelandic	*English US*	*English US*	*English US*	*English US*	English UK
Indonesian	Indonesian	Indonesian	Indonesian	*English US*	Indonesian
Italian	Italian	Italian	Italian	Italian	Italian

Teams	Dataverse for Teams	Power Apps Studio	Power Automate	Power Virtual Agents	Power BI
Japanese	Japanese	Japanese	Japanese	Japanese	Japanese
Korean	Korean	Korean	Korean	Korean	Korean
Latvian	Latvian	Latvian	Latvian	*English US*	Latvian
Lithuanian	Lithuanian	Lithuanian	Lithuanian	*English US*	Lithuanian
Norwegian - NB	Norwegian - NB	Norwegian - NB	Norwegian - NB	Norwegian - NB	Norwegian - NB
Norwegian - NN	*Norwegian - NB*	*Norwegian - NB*	*Norwegian - NB*	*Norwegian - NB*	*Norwegian - NB*
Polish	Polish	Polish	Polish	Polish	Polish
Portuguese (Portugal)	Portuguese (Portugal)	Portuguese (Portugal)	Portuguese (Portugal)	*Brazilian Portuguese*	Portuguese (Portugal)
Romanian	Romanian	Romanian	Romanian	*English US*	Romanian
Russian (cyrillic)	Russian (cyrillic)	Russian (cyrillic)	Russian (cyrillic)	Russian (cyrillic)	Russian (cyrillic)
Serbian (Latin)	Serbian (Latin)	Serbian (Latin)	Serbian (Latin)	*English US*	Serbian (Latin)
Simplified Chinese	Simplified Chinese	Simplified Chinese	Simplified Chinese	Simplified Chinese	Simplified Chinese
Slovak	Slovak	Slovak	Slovak	*English US*	Slovak
Slovenian	Slovenian	Slovenian	Slovenian	*English US*	Slovenian
Spanish	Spanish	Spanish	Spanish	Spanish	Spanish
Spanish (Mexico)	*Spanish*	*Spanish*	*Spanish*	*Spanish*	*Spanish*
Swedish	Swedish	Swedish	Swedish	Swedish	Swedish
Thai	Thai	Thai	Thai	English US	Thai
Traditional Chinese	Traditional Chinese	Traditional Chinese	Traditional Chinese	Traditional Chinese	Traditional Chinese
Turkish	Turkish	Turkish	Turkish	Turkish	Turkish
Ukrainian	Ukrainian	Ukrainian	Ukrainian	*English US*	Ukrainian

Teams	Dataverse for Teams	Power Apps Studio	Power Automate	Power Virtual Agents	Power BI
Vietnamese	Vietnamese	Vietnamese	*English US*	*English US*	Vietnamese
Welsh	*English US*	*English US*	*English US*	*English US*	*English UK*

See also

What is Dataverse for Teams?
Create tables
About the Dataverse for Teams environment

FAQs for Dataverse for Teams

Article • 12/16/2022

Here is a list of frequently asked (FAQs) for Dataverse for Teams; also see Dataverse for Teams licensing FAQs in the Power Platform admin guide.

What does Dataverse for Teams enable and how does this impact Microsoft Power Platform and Teams users?

Power Apps developers and Power Virtual Agents chatbot creators will now be able to make and manage their apps and bots directly in Teams with embedded Power Apps and Power Virtual Agents apps. This enables a streamlined end-to-end user experience and allows makers to deploy to Teams with one click through the Teams app store. These new features are powered by Power Platform enhancement in Teams that provide enterprise datastores with rich data types to Microsoft 365 users and is now included in Microsoft 365 and Office 365 licenses.

Also, apps built with Power Apps and used in Teams will be responsive to the form factor that they're loaded on, meaning a creator can build an app once for users to view full screen on both mobile devices and desktops.

What value does Dataverse for Teams offer for professional developers?

Dataverse includes a robust set of capabilities for professional developers, including API access, the ability to create plug-ins, and so on.

Dataverse for Teams is initially focused on the low-code and no-code developer and does not provide access to these capabilities at launch. However, Dataverse for Teams capability along with premium functionality through Power apps provides professional developers the ability to create robust solutions with far less code, including extending the out-of-the-box capabilities of Power Automate with their own custom logic in Azure functions, connect their own web services, etc.

When should customers use Lists versus Dataverse for Teams?

Lists is great for quickly tracking data in Teams. All your lists are instantly usable inside of Teams. Additionally, you can build canvas apps and flows on top of your Lists just like

you can today in SharePoint.

Dataverse for Teams is great for building Microsoft Power Platform solutions in Teams. With support for files, images, and multiple related tables, Dataverse for Teams can meet your needs today and be promoted to the full Dataverse if you need additional capacity or capabilities in the future.

How does security and governance differ between Dataverse and Microsoft Dataverse for Teams?

Dataverse for Teams is enterprise grade and designed to work within Teams. Focused on Teams, it aligns with the core roles identified in the environment: Owners, Members, and Guests.

For Dataverse for Teams, access to the environment is restricted just to the Teams owners, members, and guests. Granting access to the environment is managed by Teams Owner by adding or removing the Teams members. Key benefits include -

- Supports Teams Azure AD security group with runtime (JIT) user access and privilege checks.
- Auto assignment of System Administrator security role to Teams Owners.
- Ability to assign different security roles to Teams Owners and Members.
- No security settings management for Business Unit and Teams is required.

Dataverse is designed to be used in any application (not just Teams) and includes additional security features such as auditing, sharing, field level and hierarchical security.

How do users import data into tables in Dataverse for Teams?

Makers have the opportunity to bring in data through both the apps they develop as well as via connectors in Power Apps and Power Automate.

Which tables are available to developers in Dataverse for Teams?

Dataverse for Teams makes it easy for users to create custom tables for all of their scenarios.

It also includes a User table that represents the Common Data Model's User entity. Data stored in the User table corresponds to a user in Azure Active Directory (Azure AD).

Does Dataverse for Teams include support for Common Data Model?

The User table is included in Dataverse for Teams. A broader set of Common Data Model entity support is available out-of-the-box in Dataverse.

Can I use tables from other environments?

Applications can include data from environments that the maker has access to within the current tenant.

Where can I use the new visual editor?

In addition to the table designer experience previously found in Dataverse, Dataverse for Teams includes a new, easy-to-use editable grid that helps user be even more productive. You can also use tables from other environments subject to the normal permission model. The new visual editor is currently available only in Dataverse for Teams.

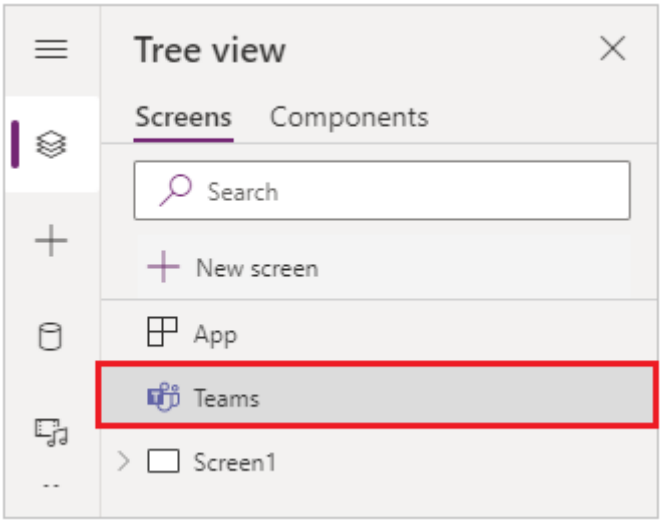
See also

Dataverse for Teams overview
How are Dataverse for Teams and Dataverse different?
Create tables
Work with table relationships

Microsoft Teams integration object

Article • 12/16/2022

You can use the Teams integration object to easily find and access the Teams integration features and use the improved usability of expressions while integrating your canvas app with Teams. The availability of a *Teams integration object* can be validated by checking the presence of **Teams** control from the tree view.



There are two features available with the integration of Power Apps with Teams. You can get a Teams theme inside a canvas app, and you can filter data depending on the team or channel context.

> **Note**
>
> To further integrate with Teams, you can use the Microsoft Teams connector with actions such as post a message, create a Teams meeting, list channels, and more.

Get the Teams theme

Teams uses three themes: *Default*, *Dark*, and *High contrast*. When using the expressions to retrieve theme names, you can use the names as *Default*, *Dark*, and *Contrast* in the Power Apps formula bar.

To retrieve a theme name, use the following formula.

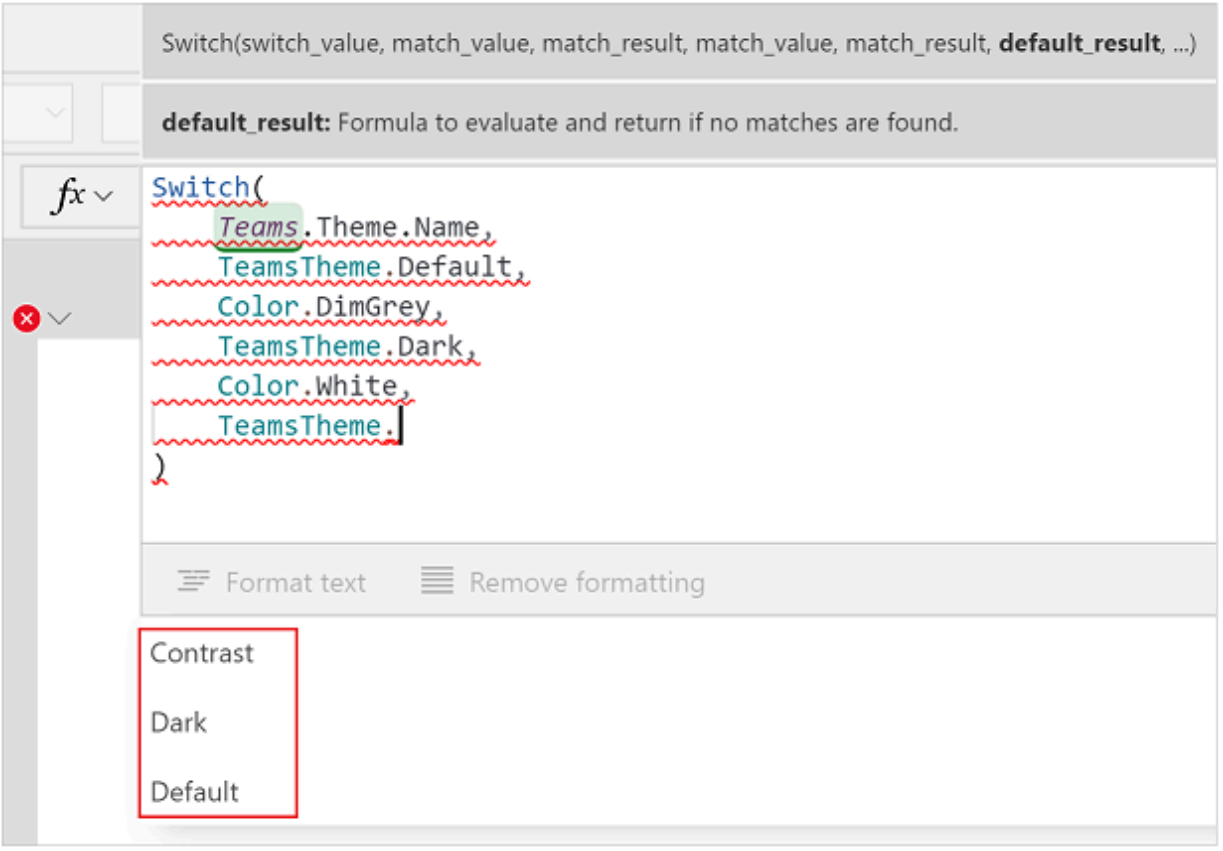
Power Fx

```
Switch(
    Teams.Theme.Name,
```

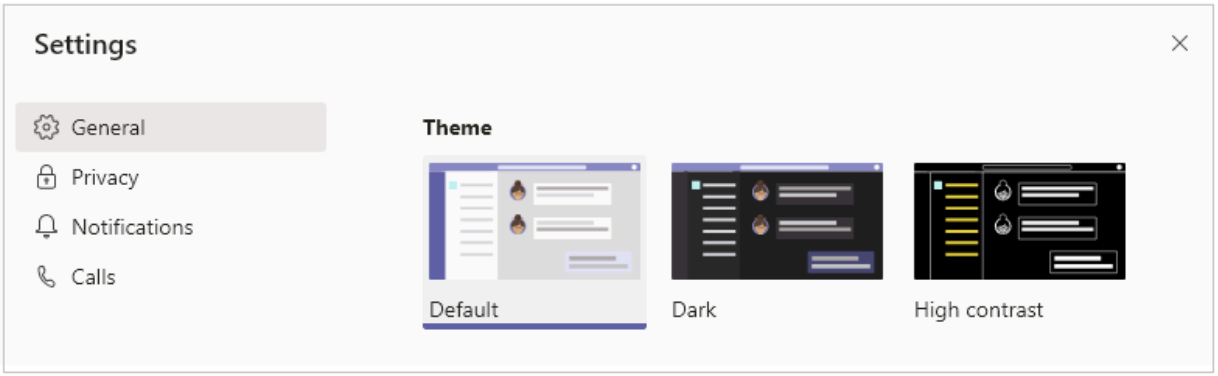
```
    TeamsTheme.Default,
    Color.DimGrey,
    TeamsTheme.Dark,
    Color.White,
    TeamsTheme.Contrast,
    Color.Black
    )
```

In the above formula, the expression *Teams.Theme.Name* is used to evaluate theme names of *Default*, *Dark*, or *Contrast* and the color of the selected canvas app object is changed depending on the color entered in the formula.

The Switch() function evaluates a formula and determines whether the result matches any value in a sequence that you specify.

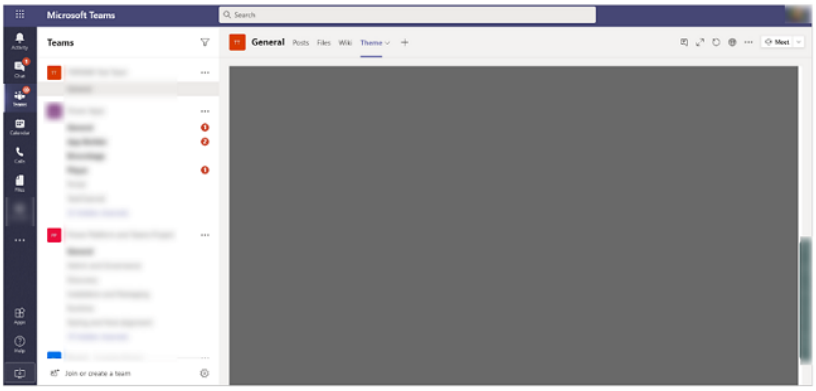
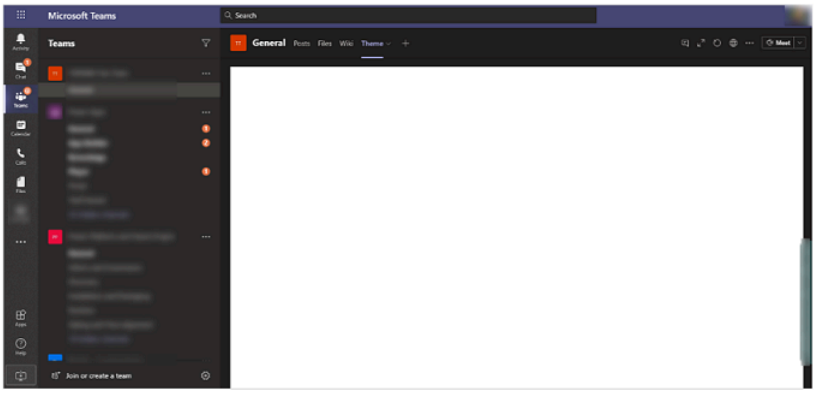
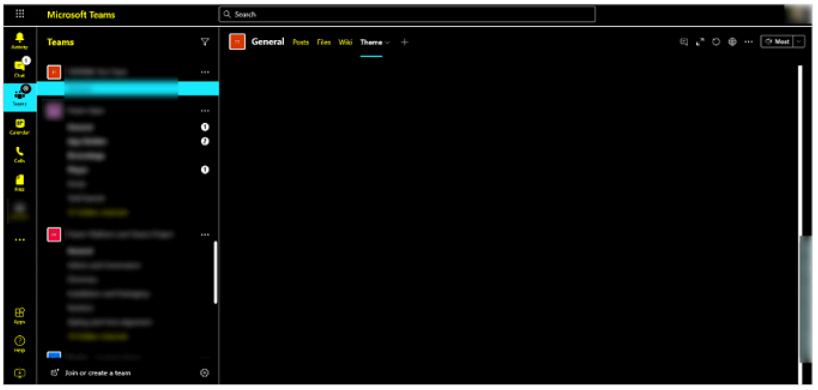


This function results in changing the background of canvas app, depending on the change in the theme selection in Teams.



For example, use this formula in the *Fill* property of a control (such as, a rectangle) to change the fill color based on the selected Teams theme.

Expand table

Teams theme	App background
Default (App background color: DimGrey)	
Dark (App background color: White)	
High contrast (App background color: Black)	

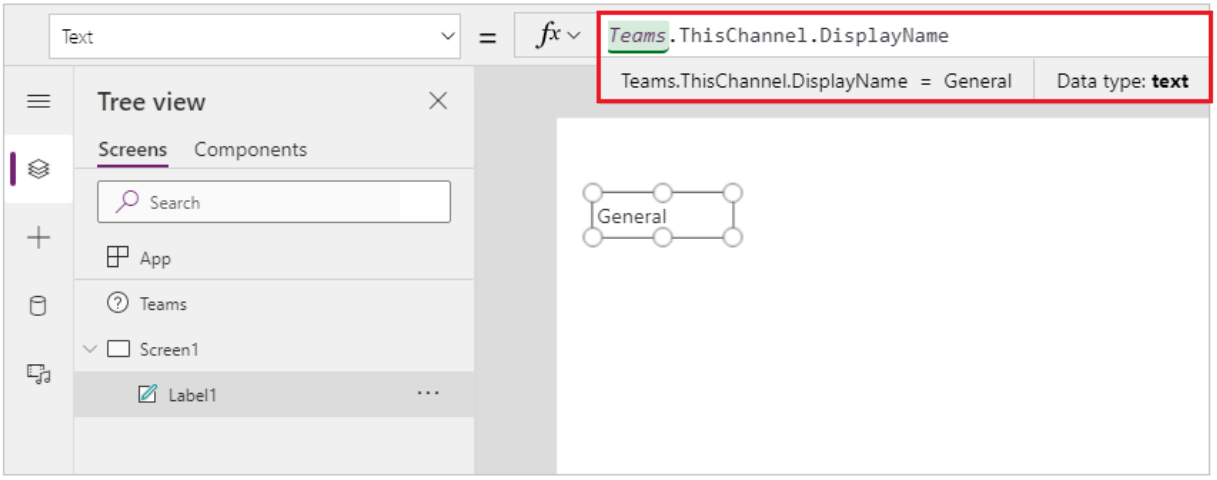
Get the Teams channel name to filter app data

You can get the Teams channel name in canvas app functions, such as function Filter(). To get a Teams channel name, use the following syntax:

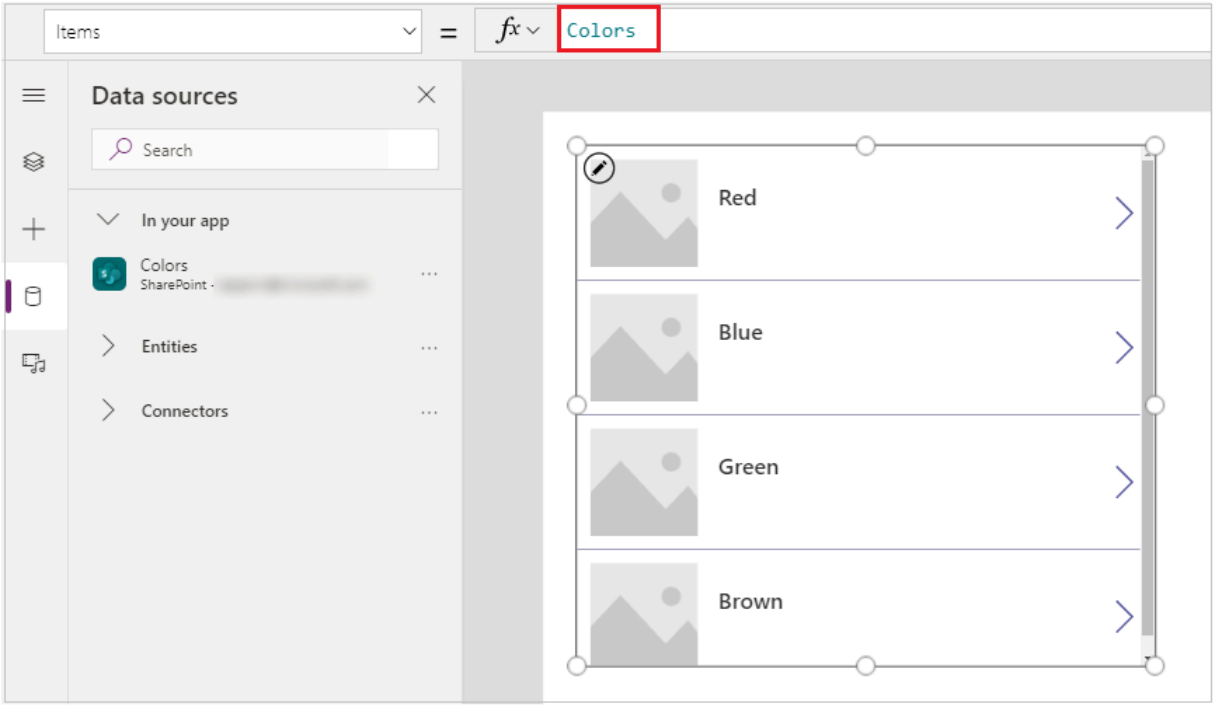
Power Apps

```
Teams.ThisChannel.DisplayName
```

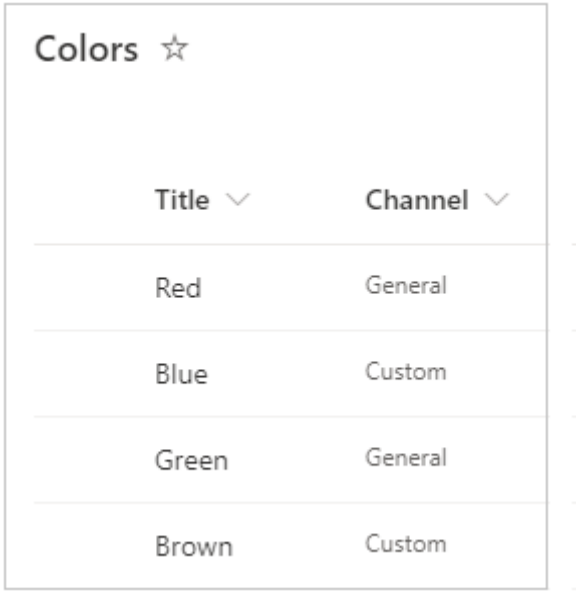
For example, you can add a label and set the *Text* property to display the name of the Teams channel.



For example, connecting to a list hosted on a SharePoint site, you can list the items in a default gallery.

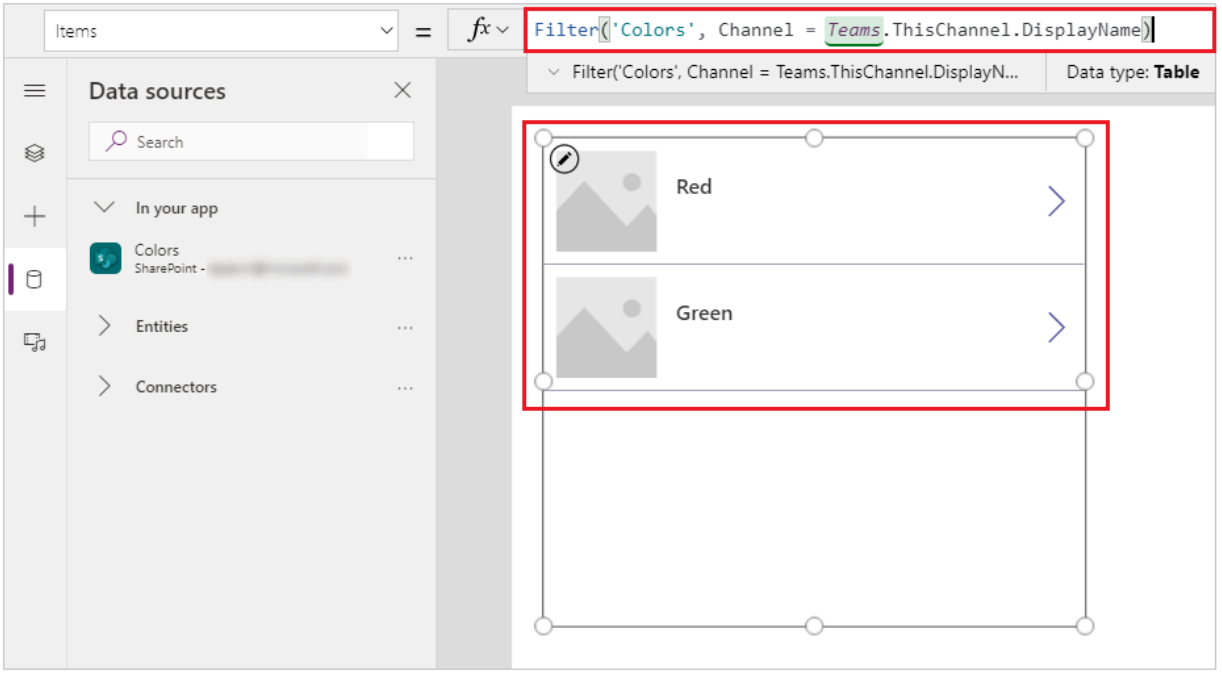


The list shows all items and a column representing the Teams channel name.



Considering the Teams channel name as *General*, you can use the following formula to filter the gallery content depending on the name of the channel.

Filter('Colors', Channel = Teams.ThisChannel.DisplayName)



For more information about the Teams client SDK, go to Context interface. For more information about using context from Teams, go to Use context from teams.

See also

Manage your apps
Understand Power Apps Studio

Feedback

Was this page helpful? Yes No

Provide product feedback

How to build apps for Microsoft Teams

Article • 02/04/2023

The sample app templates for Microsoft Teams are designed, and built with the following goals:

- Provide fully functional apps for Teams that have relevance across multiple industries.
- Provide examples of deeper integration of line-of-business data from Power Apps into collaborative elements of Teams to drive better more productive workloads.
- Provide examples of using good UX design with Teams and Power Apps.
- Provide extensible apps which users can modify and extend to suit their business requirements.

In this article, we'll review the process that we used to build these apps, and the steps that you can follow when building your own Power Apps.

> **Tip**
>
> This article highlights some of the approaches taken when designing the Teams template apps. For additional resources on bet practices for building Power Apps, please refer to **Power Apps guidance documentation** section of the Power Apps docs.

Watch this video to learn how the Power Apps templates for Microsoft Teams were made:
https://learn-video.azurefd.net/vod/player?id=a16dd643-8d23-4c54-a619-a4fe0d38c07d&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fhow-templates-built

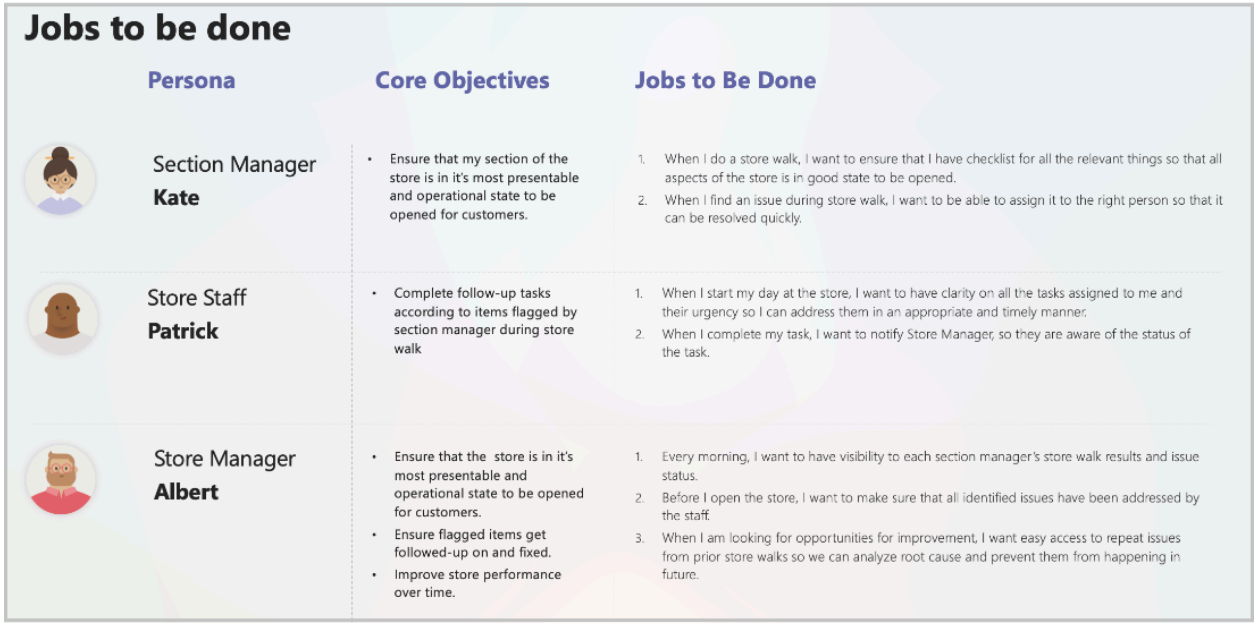
Define

Before designing and building, you need to have an idea—can be an idea for a solution to a business problem, an idea for modernizing a legacy or manual process, or an idea for an app that will make the life of your users easier.

Look to see if there are any existing solutions for the problem. Talk to the stakeholders, customers, or potential users of the app and get their input. Capture their requirements or user stories and prioritize based on what is the highest priority need or opportunity.

Identify “personas” or roles of users who will use your application. At this point, you want to focus on defining the user’s key problems—what are their unmet needs? How important is it for them to solve this problem?

Many teams tend to dive into solutioning before they've fully understood the user’s problem. One way to avoid this trap is to use the “Jobs to be done” framework. This framework helps us identify and understand the needs and goals of the users that are the original drivers of their actions or decisions. This artifact documents each of the persona’s core objective and jobs they want to accomplish in their language.



The above example shows personas and jobs to be done for Inspection sample app template for store walk scenario.

Design

Once you've identified and validated your idea, personas, and their jobs to be done, now is time to design your app. It can be tempting to jump into development at this point, but a bit more design planning will serve you well.

Having a plan will ensure a consistent user experience while enabling you to make the best design decisions for the app.

Design user experience

Make your design plan by creating user flow and wireframes (blueprints) for each job statement. This process helps to ensure the process your solution takes are logical, efficient, and tackles the core of the problem at hand.

Playing with the flow can help you better understand the various pros and cons of different processes. This understanding allows you to select the flow and screen that will enable users to achieve their end goal smoothly. There are many tools available to help you achieve the design outcome. For example, Microsoft applications such as PowerPoint, Visio, or 3rd-party applications such as Figma, Adobe XD, Sketch, or even simple pen and paper.

This is also a good check point to get the feedback from end users. Share these high-level flows with your app users and get their feedback on the end to end experience. The iteration at this stage is lower risk as you can quickly make changes and compare options.

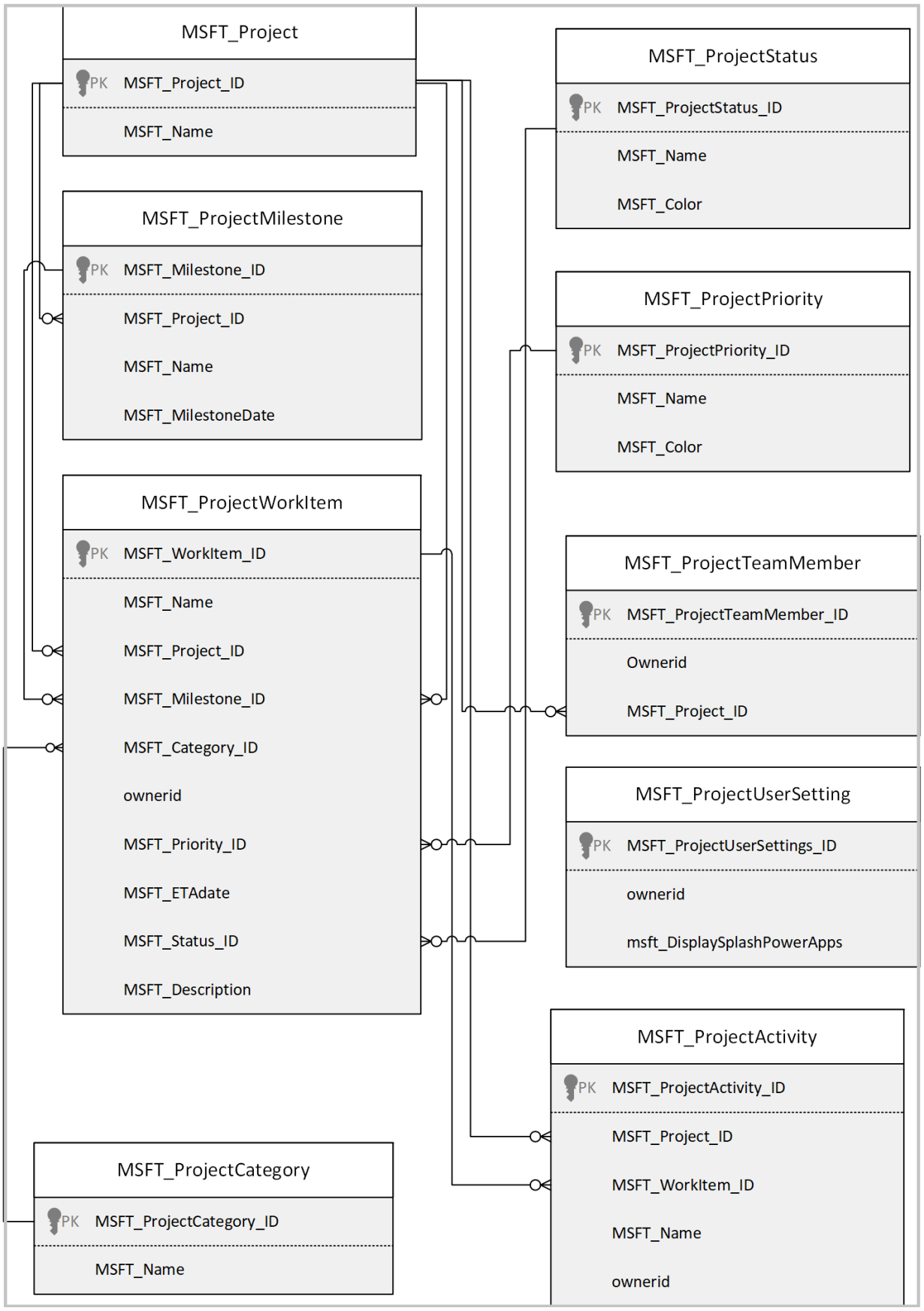
Once the base layout is in, now you can have some fun. Do you want a clean and crisp app? Or something more playful. Think about the state of mind of the user and what would work best for them. Wireframing the app screens also helps you to plan out what data connectors will be needed, and giving you an artifact you can review with the stakeholders to get feedback before building the app to verify that the proposed design addresses the requirements.



Once you have identified the requirements and personas and prioritized the backlog, identify the set of features that will be needed to satisfy the "must have" items in the backlog. This is called the Minimum Viable Product (MVP).

Design data model

Along with the user experience, designing the data model of the app is also important. While Power Apps can connect to data from anywhere, you want to think through the data that is needed in the app and how it will be used. In Visio, or some other data diagraming tool, plan out the tables, fields, and relationships that will be needed to support your desired app functionality. Here's an example of the data model for Milestones app tables, and their relationships.



Think through how users will need to interact with the data as you plan what field data types to use. For example, drop-downs for Status, Priority, and Category are updatable in Milestones—that's why we created a table for these options and used a lookup to link them to the work item. For other infrequently changing list options that aren't editable

by users in the app, we used choice data type fields. Make sure your data design fits the purpose and usage patterns of your app and users.

Develop

Now that you've got an idea and a design, start building the app in Power Apps. If you're building an app using Power Apps in Teams, we recommend creating a private team and limiting the members of the team to only those involved in the app development process.

Be sure to periodically review progress with stakeholders to get feedback. Once they see the app coming together, it will build excitement for the finished product, and they'll also provide feedback that you can use to iterate and improve the app experience.

Validate

Once the app functionality is complete, thoroughly test the app:

- Test as each Persona
- Test in Teams Desktop
- Test in Teams Mobile
- Test as user who isn't a member of the team if your app will be shared for broad distribution (sharing with colleagues outside of the team).
- Verify **App Checker** to determine if there are any issues or accessibility problems with your app and address them.
- Run the app through the Power Apps Code Review Tool. This tool has been purpose built to highlight specific areas of potential improvement in the app across a range of areas such as performance, maintainability, and security.

Deploy

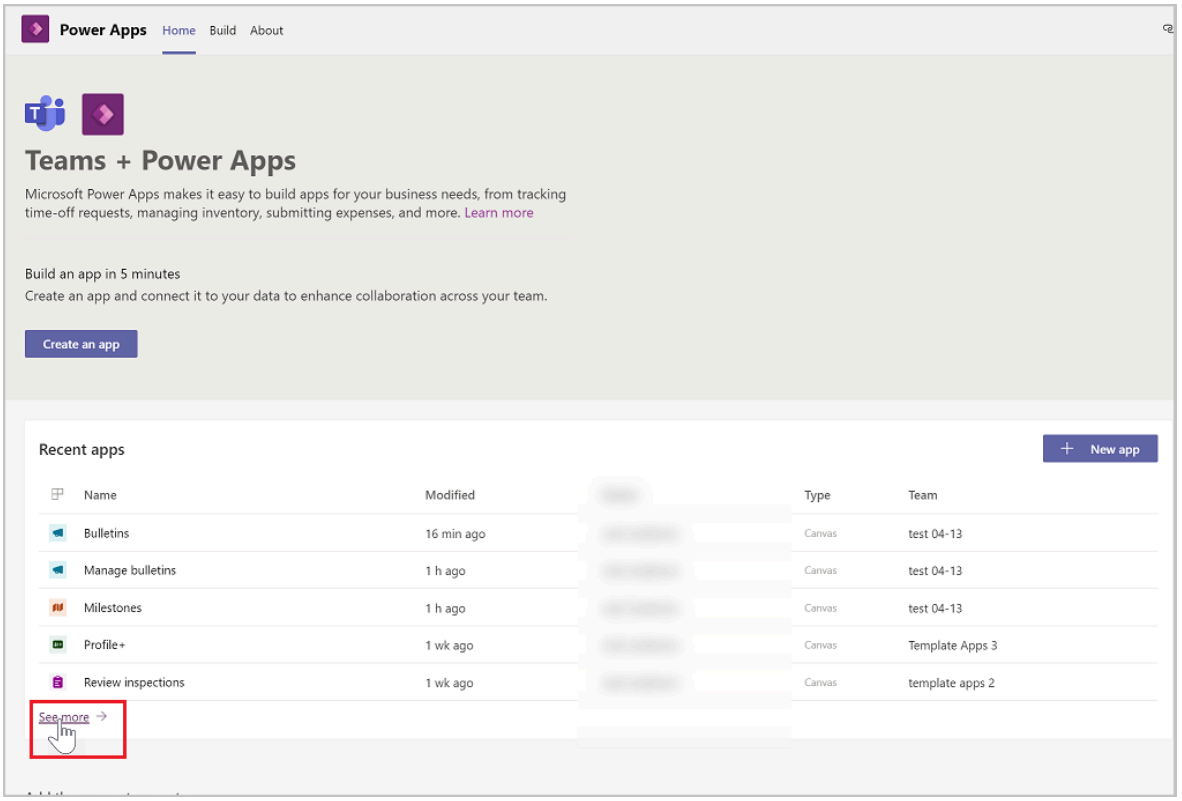
Your app is now complete and fully tested. Now it's time to onboard users. With Teams, you've several options that you can use to give users access to your app:

- Add members to the team in which you developed the app.
- Create a solution and move the app and underlying tables to a different Team environment.

- Share the app for broad distribution.

To share an app for broad distribution, follow these steps:

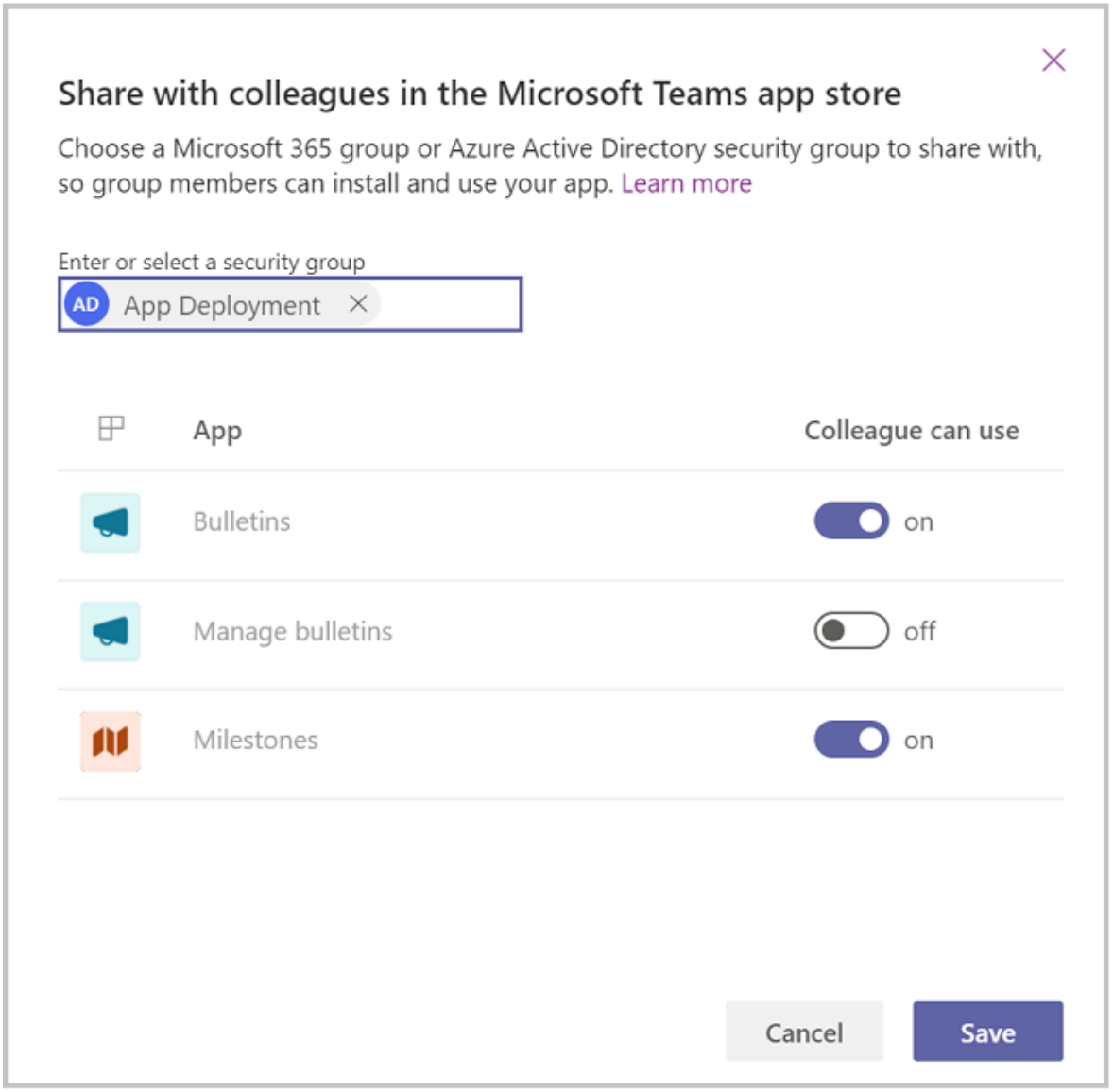
1. In Teams, open the Power Apps app.
2. Under **Recent apps** select **See more**.



3. Select the team where your app is installed.
4. Select **Share with colleagues**.



5. In the **Enter or select a security group** field, enter a security group or team with which you want to share access to your app.
6. Toggle **Colleague can use** to **on** for the app that you wish to share.



7. Select **Save**.

> **Note**
>
> You'll need to set table permissions for the Microsoft Dataverse tables included in your app before users will be able to use it. For more information on table permissions, see Set permission and share apps with colleagues.

Feedback

Was this page helpful? Yes No

Provide product feedback

Add notifications to make your apps collaborative

Article • 01/20/2023

Apps in Microsoft Teams are designed to enable collaboration between people in your organization. When you have an app that has a hand-off of information, such as group A doing something that group B then needs to process, if everything happens in the app, users may miss important updates, and processes might stall.

Using notifications for critical events in your apps in Teams enhances collaboration by proactively letting users know that something needs their attention or needs to be done.

Teams and Power Platform give you many options in the ways that you can notify people.

- Email notifications
- SMS/text notifications
- Teams messages
- Adaptive cards
- Channel posts

Watch this video to learn how to add notifications to make your apps collaborative: https://learn-video.azurefd.net/vod/player?id=c17e7bd1-aa38-434e-8120-e26b90cb3618&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fadd-app-notifications

Choose the right notification type

How should you decide what type of notification to use? Many people default to email notifications, but many people also complain that they receive too many emails. Email is an acceptable type of notification, but it should be used sparingly. Especially for urgent notifications, as frequent emails are only seen after a period of time.

Here are some questions to ask when defining the appropriate type of notification:

1. **Is the notification urgent?**

   If you're creating an app to track safety issues, and there's an urgent issue that requires immediate attention, a notification like SMS text, push notification, or high priority Teams message should be used so the notification is immediately noticed.

But don’t over-use these types of notifications for non-urgent notifications, or you risk irritating your users.

2. **Is the notification informational or actionable? If you're creating a notification, what do you want to have the notification recipient do with the notification?**

   Informational notifications are to notify users, but they're for information purposes only. For example, in the Employee Ideas sample app template for Teams, when a new idea is created, a message is posted in a Teams channel notifying people that a new idea has been created.

   Actionable notification enable one or more follow-up activities by the recipient. In the extending Employee Ideas article, we show how to change the notification to an adaptive card so users can open the submitted idea and vote for it.

   If the notification requires that the user be able to do something with the notification, such as open a record or respond, an actionable notification should be used—such as an interactive card, or add a hyperlink to deep link to the app.

   Also, Teams makes other types of notifications more actionable by enabling conversations around the notification—when you post a message to a Teams channel, users can respond to it in the channel, facilitating the ability for multiple people to discuss the notification.

3. **Is the notification personal, or to a group?**

   Is there a specific person that should be notified? Or, is there a group of people that needs to know about the notification? For personal notifications, consider sending a message, email, or text message so that other users aren't overloaded with non-applicable notifications. If the message should be seen by multiple people in a group, a post In a Teams channel is a better choice, as the users in the group can all see the message and collaborate on it.

4. **What is the frequency of notification?**

   In the Inspection sample app template for Teams, by default, a message is posted to a Teams channel when an inspection is completed. However, if you've a high frequency of inspections, many messages will be posted to the channel. And when many notifications are posted to a channel, individual notifications can get lost. Think of your phone notification center—when you've a few apps giving you notifications, they're helpful, but if you've many apps posting updates, notifications become noise. In this case, you'd want to reevaluate your notifications, and either change the notification to only notify on urgent inspections, or issues.

5. **What is the impact on the users?**

   Your colleagues are probably busy, and each interruption can add stress to their lives—especially in high quantities. Notifications are good if they make important information more available. But they can also have negative impact if too frequent or unnecessary. Also, arbitrary notifications can diminish a user's sense of control—if many messages start arriving that a user has no control over. And it can be a frustrating experience. Consider giving users an option in the app to control their notifications—this can be a settings toggle to opt in to notifications, or a preference to select what type of notification to receive. See extending bulletins for an example.

Notifications: Power Apps or Power Automate

When creating a notification, you have two options:

- Have your app create the notification directly.
- Or, trigger a Power Apps flow to post the notification.

The decision as to which approach you should use depends on what type of notification you're using, and how the notification should be triggered. Some notification types—such as adaptive cards, require Power Automate. Others, like sending emails, text messages, or Teams channel messages can be sent from a Power Automate formula, or a Power Automate flow. While there isn't a "right or wrong" answer to this question, here are some rules to consider:

- Notifications sent directly from an app without flow will be in the context of the user making the change in the app. That means that the user will need access to the service sending the message, and some types of messages like emails may come from their personal account. If you want to send messages from a service account or general notification mailbox, a Power Automate flow can be owned by a different user account, and send a message when triggered by a data condition (like a record being created).

- If a flow sends a notification, and is triggered directly by the app, the flow needs to be shared or assigned with the user triggering the flow. If a flow sends a notification, and is triggered by a data condition (like a record creation or update), the flow doesn't need to be shared or owned by the triggering user.

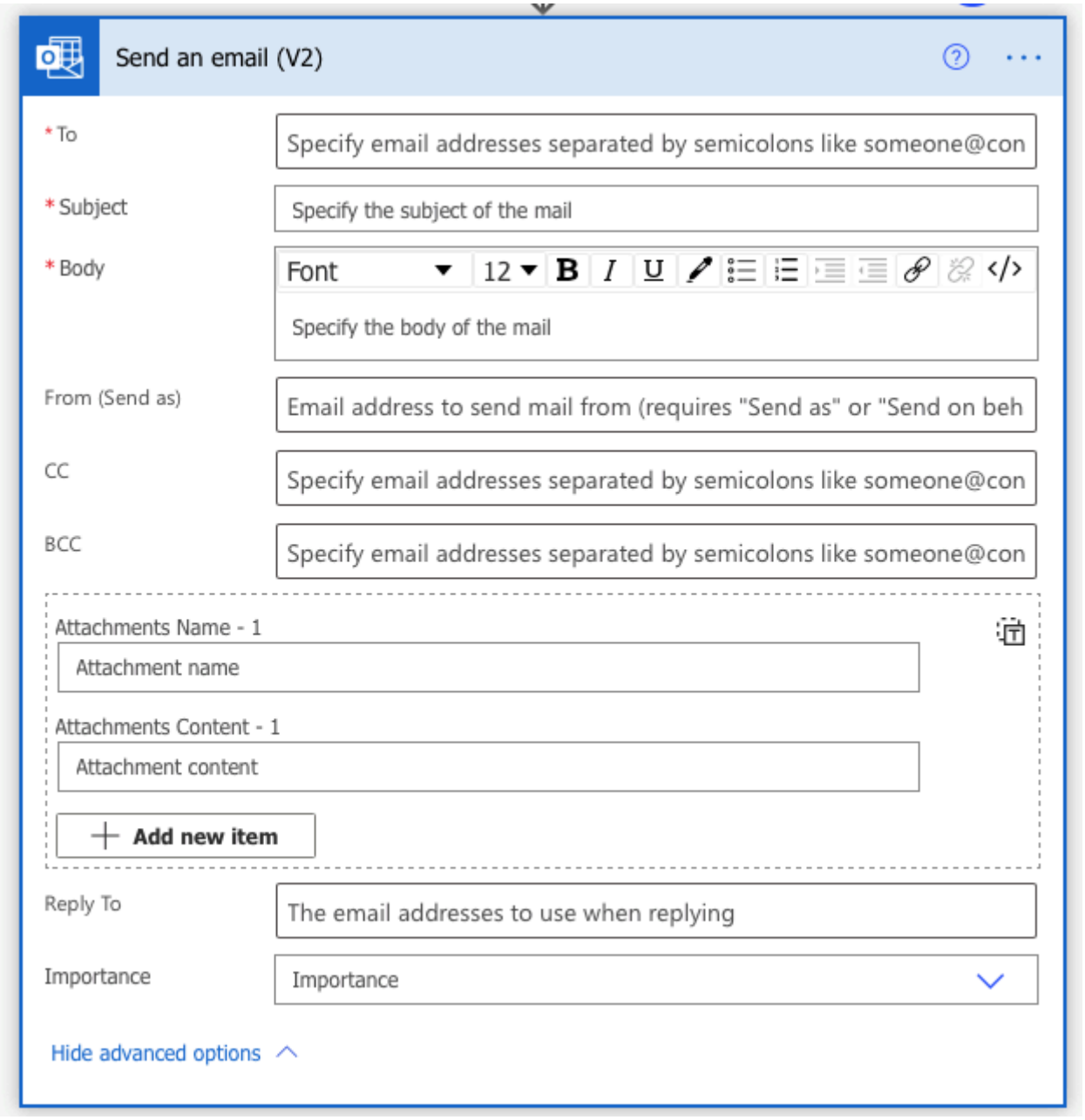
- By using Power Automate to send notifications, you can update your notifications without having to republish your app. If your notification details frequently change, or users other than the maker of the app need to modify them, having the

notification sent by Power Automate can segment your workloads and enable other users to work on the notifications while you work on the app.

- Power Apps sending notifications directly use formulas to define the notification logic, Power Automate flow uses a more graphical interface to set the properties of a notification. For example, from a Power Apps app, you can use the following formula to send email with the Outlook connector:

```
Power Fx

Microsoft365Outlook.SendEmail("mailbox@contoso.com", Summary, Description)
```

In Power Automate instead, the send an email action is presented in a graphical manner, allowing users to populate the fields in a manner similar to how they send an email in Outlook.



Sending an email from Power Apps directly is faster for experienced makers, but the more details are required—such as attachment files and formatted text. For the less experienced makers, sending the email with Power Automate flow might be easier.

Feedback

Was this page helpful? Yes No

Provide product feedback

Integrate with Microsoft Planner

Article • 01/20/2023

In this article, we'll learn how to integrate Power Apps with Planner to create and assign tasks to users within the organization.

Power Apps in Microsoft Teams enables collaboration between people and groups. Employees already use Teams to work, communicate, and collaborate. Apps in Teams have deep integration with other Teams capabilities. For example, in Teams, you can assign tasks to team members using Planner in your Teams channel. Apps that require follow-up by others in your team, such as to address issues reported in the Issue Reporting app, can leverage Planner in the Teams channel and create Tasks where users already manage their tasks.

Watch this video to learn how to integrate Power Apps with Planner: https://learn-video.azurefd.net/vod/player?id=37312c36-393d-44fd-ba8a-9b3dbdf8d60e&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fintegrate-planner

Prerequisites

To complete this lesson, we'd need the ability to create apps within Teams that will be available as part of select Microsoft 365 subscriptions. We'll also need the ability to connect to Planner.

Login into Teams

Login into Teams using either the Desktop app or the web app.

Create a new Team

In this section, we'll create a new Teams team and then create an app within that team. Skip this section if you already have a team that you want to use instead.

1. To create a new team, select the **Teams** tab > select **Join or create a team** > select **Create Team** > select **From Scratch** > select **Public**, and give the team a name such as "Planner Integration". And then, select **Create**.

2. Select and add any members in your organization that you'd like to add to the team for testing purposes, and select **Add**.

3. Select **Close**.

   The new team gets created and is listed under the Teams tab.

Create a new app

We'll create an app with five fields capturing details that will be used to create a Task in Planner.

1. Open Teams.
2. Select Power Apps from the left-pane.
3. Select + **New app** under **Recent apps**.
4. Select the team created earlier, and select **Create** to open Power Apps Studio.
5. Enter a name for the app, such as "Integrate with Planner", and select **Save**.

   The app shows a default gallery
6. Select the database option from the left-pane to connect to the Planner.
7. Select + **Add data**, and then select **Connectors**.
8. Enter "Planner" in the search box to look for the Planner connector.
9. Select **Planner**.
10. If prompted, sign in.

    Planner gets added as a data connection to the app
11. Select to open the Tree view.
12. Select + **New screen** > **Blank layout**.
13. Add the following controls.

    a. Title (text input)

Expand table

Property	Value
Name	Title
Type	Text box

Property	Value
X	Screen2.Width/2
Y	135
Width	320
Height	35

b. From Date (date picker)

Expand table

Property	Value
Name	FromDate
Type	Date picker
X	Screen2.Width/2
Y	235
Width	320
Height	35

c. Due Date (date picker)

Expand table

Property	Value
Name	DueDate
Type	Date picker
X	Screen2.Width/2
Y	335
Width	320
Height	35

d. Assigned To (combo box)

Expand table

Property	Value
Name	AssignedTo
Type	Combo box
X	Screen2.Width/2
Y	435
Width	320
Height	35

e. Planner ID (combo box)

Expand table

Property	Value
Name	PlannerID
Type	Combo box
Items	Planner.ListGroupPlans(Param("groupID")).value
X	Screen2.Width/2
Y	535
Width	320
Height	35

14. Add the following labels.

a. Title

Expand table

Property	Value
Name	Label_Title
X	365
Y	135
Width	320

Property	Value
Height	35

b. From Date

Expand table

Property	Value
Name	Label_FromDate
X	365
Y	235
Width	320
Height	35

c. Due Date

Expand table

Property	Value
Name	Label_DueDate
X	365
Y	335
Width	320
Height	35

d. Assigned To

Expand table

Property	Value
Name	Label_AssignedTo
X	365
Y	435
Width	320

Property	Value
Height	35

e. Planner ID

Expand table

Property	Value
Name	Label_PlannerID
X	365
Y	535
Width	320
Height	35

15. Add a button to the bottom of the screen with the following properties.

Expand table

Property	Value
Name	Button_CreateTask
X	526
Y	635
Width	320
Height	35

16. Copy the following formula in the **OnSelect** property of **Button_CreateTask**.

Power Fx

```
Planner.CreateTaskV3(
    Param("groupId"),
    PlannerID.Selected.id,
    Title.Value,
    {
        startDateTime: FromDate.Value,
        dueDateTime: DueDate.Value,
        assignments: AssignedTo.Selected.'User Name'
```

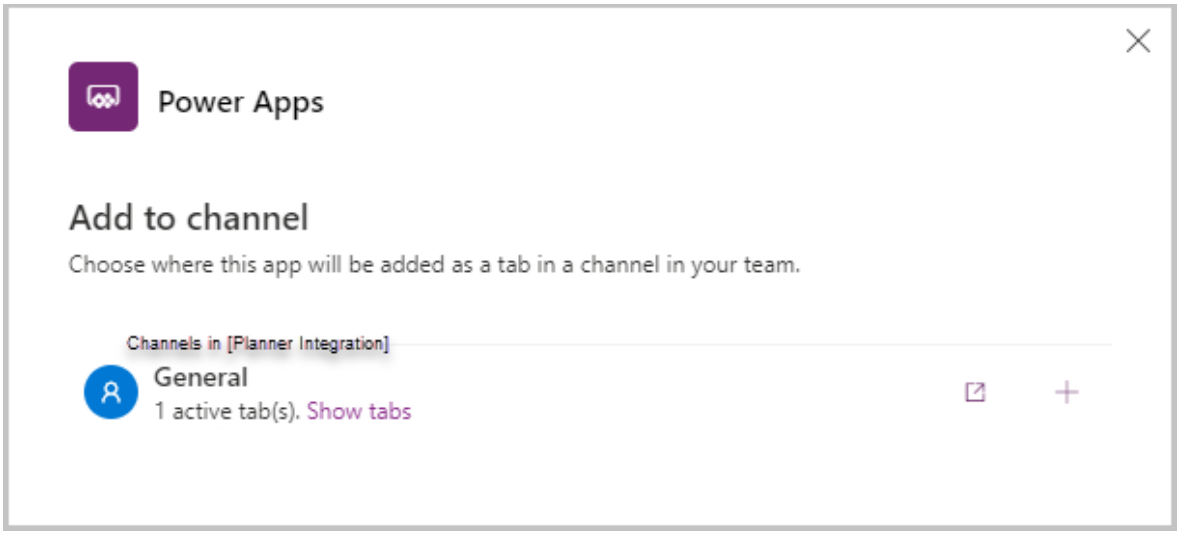
```
    }
)
```

> ⓘ Note
>
> Upon selecting the **Create Task** button, it'll use the parameters provided in the formula to figure out the team for which the task is being created, the planner within the team the task is to be created, the Title from the text box captured on the screen, and the From and Due dates captured on the screen.

17. Move **Screen2** up above **Screen1**. If you want, you can also delete **Screen1** instead.

Save and publish the app

1. Select **Save** on the top-right side of the screen to save the app.
2. Select **Publish**.
3. Select **Next**.
4. Under **Add to Channel**, make sure the channel under which the app should be shown is listed and shows as at least "1 active tab(s)" and not "0 active tab(s)".



5. Select **Save and Close**.

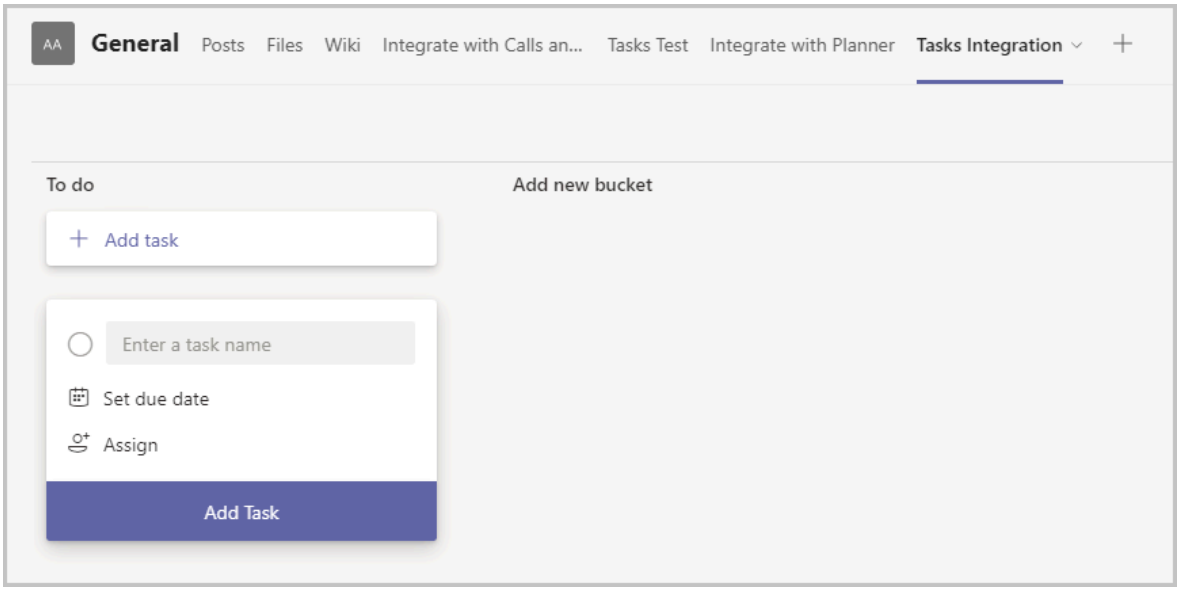
Test the app

1. Open the team under which the app is created.
2. Select **+ Add a tab**.

3. Search for tasks by Planner to add a planner tab to the team.

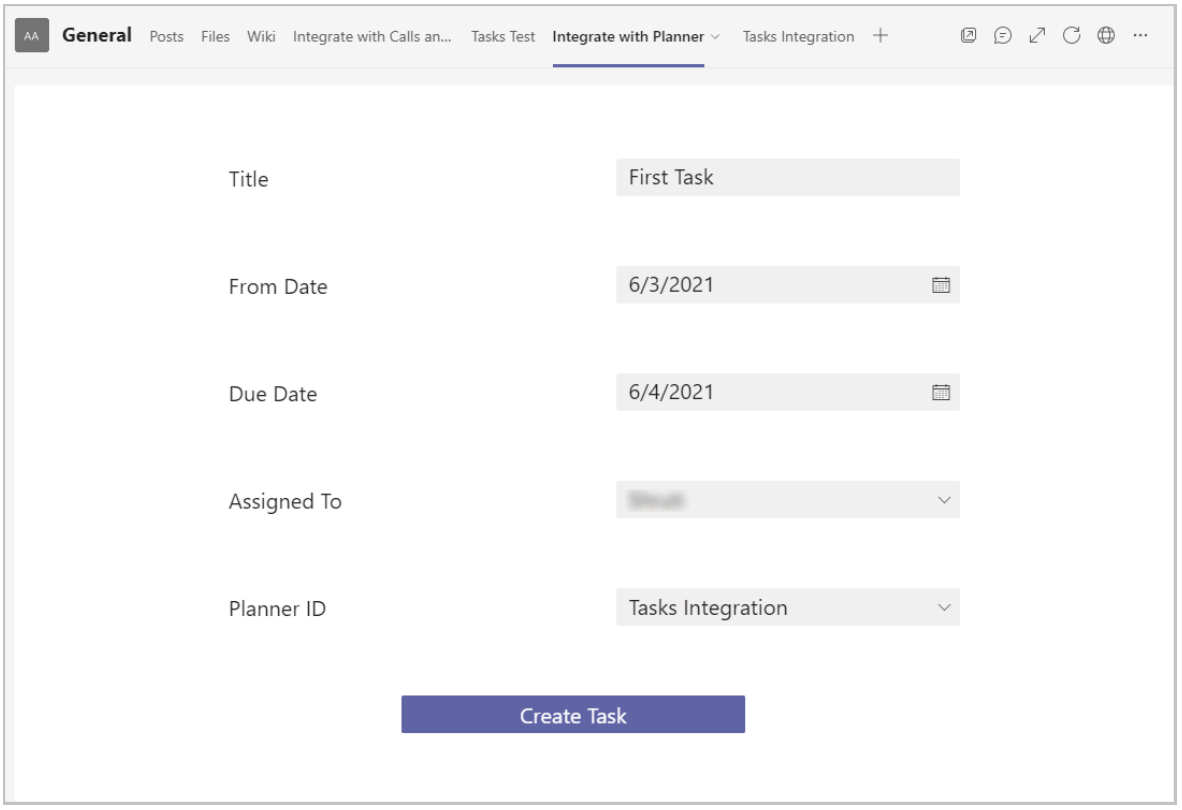
4. Enter tab name under **Create a new plan**, for example "Tasks Integration". And then, select **Save**.

   The tab gets added to the Team



5. Run the app in preview mode or from within the team.

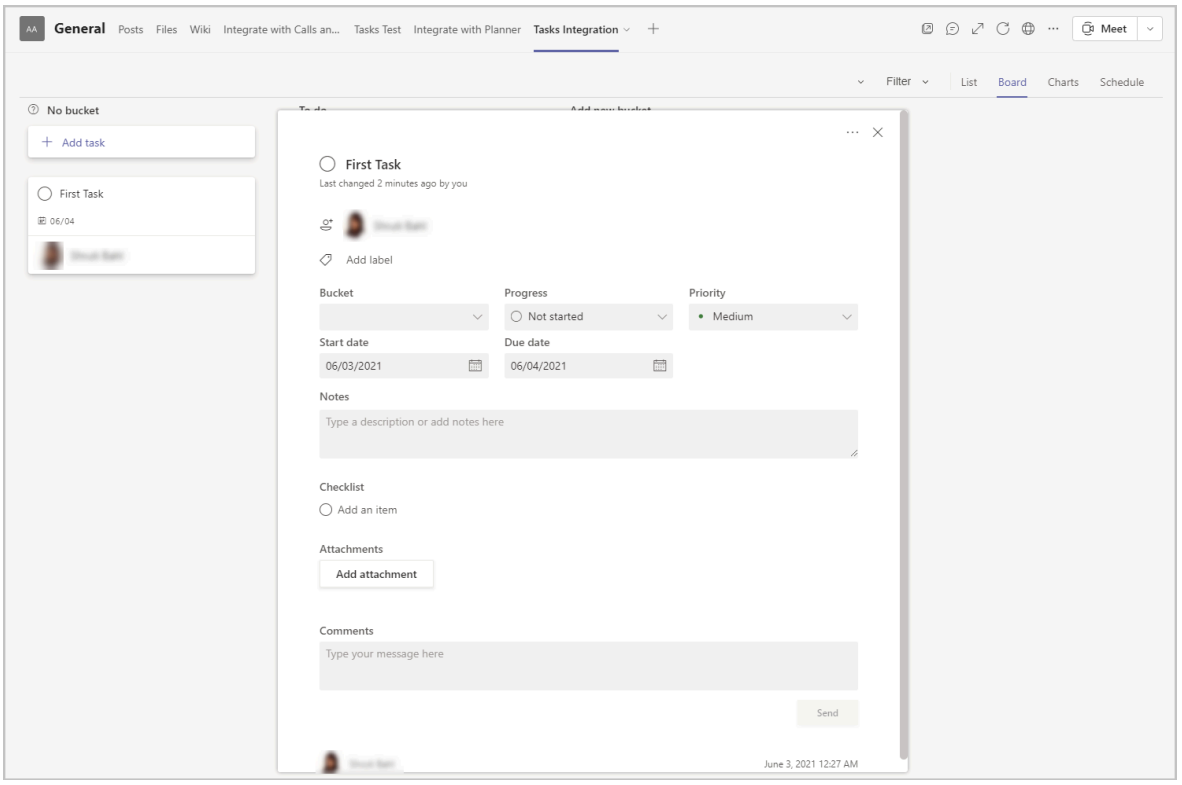
6. Populate the columns with appropriate values.



7. Select **Create Task**.

8. Go to the Planner tab for which the task is being created—in our example, "Tasks Integration Planner".

   A new task with the details should be added to the list.



See Planner integration in action

To see an example of Planner integration, check out the Issue Reporting sample app template for Teams. Issue Reporting makes it easy for users to report issues. These issues are created in Planner and assigned for follow-up.

This integration can be easily extended to add logic to automatically assign the Planner Tasks or provide notifications for issue tasks.

Feedback

Was this page helpful? Yes No

Provide product feedback

Integrate with Azure Active Directory

Article • 01/20/2023

In this article, we'll learn about how to integrate apps created using Power Apps in Microsoft Teams with Azure Active Directory (AD).

Integration with Azure AD enables many deep capabilities within your organization hierarchy, the people in your organizations, and groups. For example, you can get information about a person, their job title, department, or photo. You can also get information about the user's organization hierarchy, such as their manager.

This integration can also be used to grant access to app functionality based on their role or group membership, and therefore, be used to control security to sensitive data or functionality.

In this article, we'll create a Power Apps app with a button with administrative access on it. The visibility of the button will depend on whether the user is a member of the admin group or not.

Watch this video to learn how to integrate Power Apps app with Azure Active Directory: https://learn-video.azurefd.net/vod/player?id=9fabda9a-e9b5-43af-addd-a27d3f381f18&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fintegrate-azure-ad

Prerequisites

To complete this lesson, we'd need the ability to create apps within Teams that will be available as part of select Microsoft 365 subscriptions. We'll also need access to Azure AD to create the admin group and assign members to it.

Sign in to Teams

Login into Teams using either the Desktop app or the web app.

Create a new Team

In this section, we'll create a new team, and then create an app within that team. Skip this section if you already have a team that you want to use instead of creating a new one.

1. To create a new team, select the Teams tab, and then select **Join or create a team** on the left bottom of the screen.
2. Select **Create Team** > **From Scratch** > **Public**, give the team a name such as "Azure AD Integration", and select **Create**.
3. (Optional) If you have any members in your organization that you'd like to add to the team for testing purposes, select them, and add them in the popup for adding members.
4. Select **Close**.

The new team gets created, and is listed under the Teams tab.

Create a new Azure AD Group

1. Sign in to Azure portal with the same account that you logged into Teams with earlier.
2. On the top search bar, search for and select **Groups**.
3. Select **+ New Group**.
4. Set up the group with the following values:

Expand table

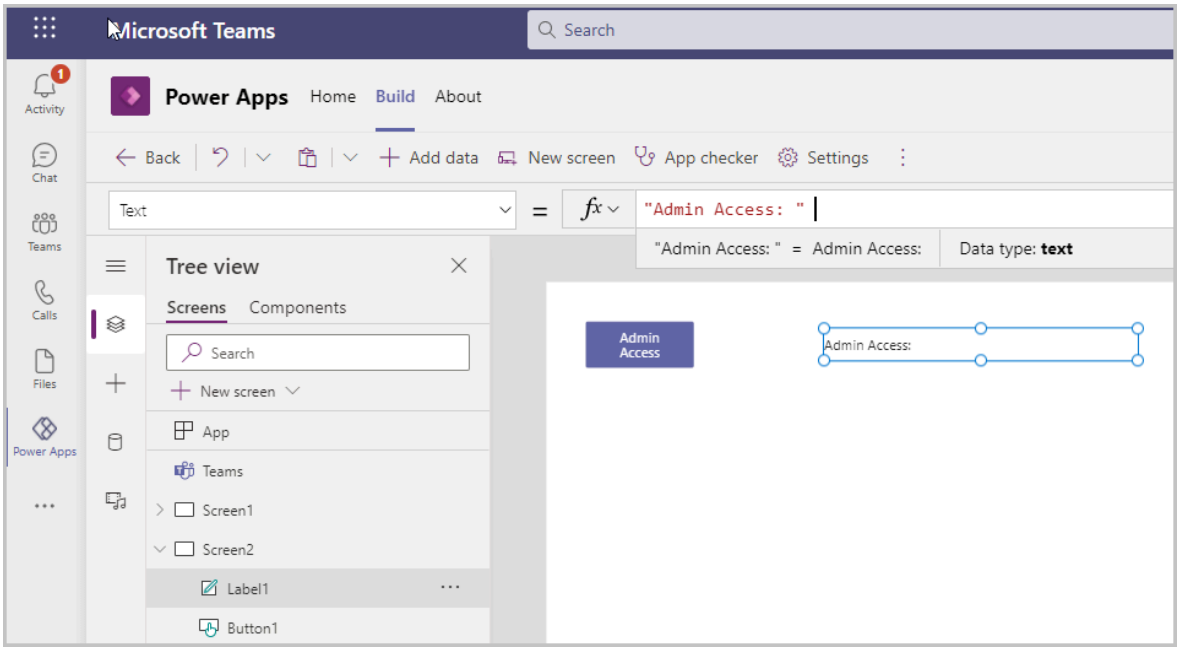
Option	Value
Group type	Microsoft 365
Group name	Admin Access
Group email address	Automatically updated
Group description	Group for users with Admin access

5. Select **Create**.
6. Copy the group's object ID from the group overview screen for use later.

Create a new app

We'll create an app with a button that will only show up if the user is a member of the team we created above.

1. Open the Teams app.
2. Select **Power Apps**.
3. Select **+ New app** under **Recent apps**.
4. Select the team created earlier, and then select **Create**.
5. The app gets created, and the Power Apps Studio opens to allow editing the app.
6. Enter a name for the app, such as "Integrate with Azure AD", and then select **Save**.
7. Select **Data** > **Add data**.
8. In the connectors list, search for Azure AD, and select it to add that as a connection.
9. Select **Connect**.
10. Sign in, and select **Accept** to connect.
11. In the **Tree view**, select **+ New screen** to add a new screen to the app.
12. Select **+** (Insert) icon on the left-pane.
13. Select **Button**.
14. Set the "Text" property of the button to “Admin access”.
15. Select **+** (Insert) again, and then select label control to add.
16. Update the label text to "Admin level: ".



17. Select the app from the tree view, and open the **OnStart** property from the top-left of the screen.

18. Enter the following formula for the app OnStart:

```
Power Fx

If(
    !IsEmpty(
        AzureAD.CheckMemberGroups(
            User().Email,
            ["Enter your Azure group ID here"]
        ).Value
    ),
    Set(varAdmin,true),
    Set(varAdmin,false)
)
```

> **Note**
>
> Update the formula with the group ID (GUID) for the group you copied earlier.

19. Select ... (ellipsis) next to the app, and then select **Run OnStart**.

20. Select ... (ellipsis) next to **Settings** on the top ribbon, and select to view variables. **varAdmin** is set to False.

21. Select Screen 2 from the Tree view, and update the Text formula for the label to `"Admin Level: " & varAdmin`. The label will currently show as "Admin level: false" on the screen.

    We'll now control the visibility of the Admin Access button so that it only shows up when the user is an admin (for example, belongs to the Admin access group in Azure AD)

22. Select the button, and set the **Visible** property to "varAdmin".

    The button doesn't show up anymore since we don't have admin access.

Testing the integration

1. Sign in to Azure portal.

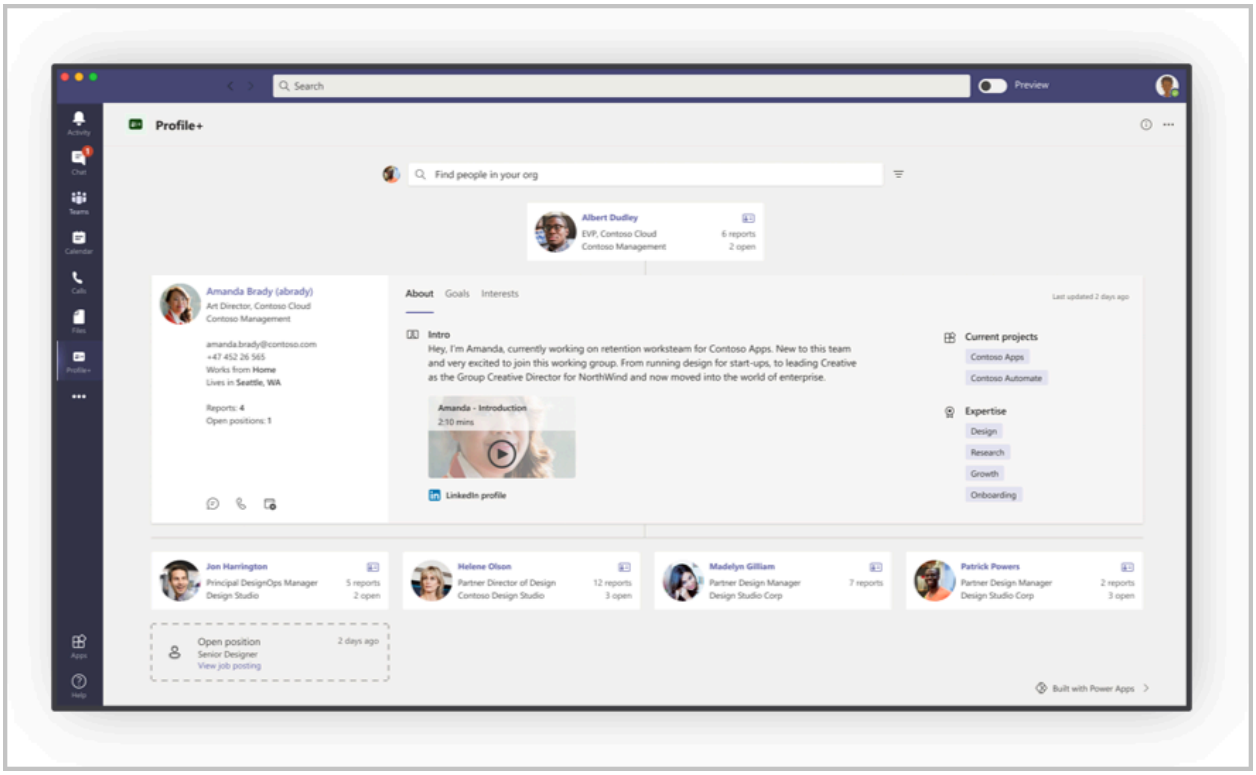
2. Select the group created earlier.

3. Select **Members** under **Manage** from the left-pane.
4. Select + **Add members** to add members to the group/
5. Select users from the list of users under **Add members** to add to the group.
6. Select **Select**. Selected users get added to the group.
7. Go back to Teams.
8. Select **Tree view**.
9. Select the app > **Run OnStart**.

The admin access button is now visible on Screen 2, and the label shows admin level as **true** since we're now members of the Admin access group.

See Azure AD integration in action

You can see an example of Azure AD integration in the **Profile+** sample app template for Teams.



- **Office 365 Users** connector is used to retrieve the user profile, including name, photo, manager, and other details.
- **Office 365 Groups** connector is used to retrieve the people who report to a selected user.

Feedback

Was this page helpful? Yes No

Provide product feedback

Integrate with calls and meetings

Article • 01/20/2023

Apps in Microsoft Teams drive collaboration by enabling users to do work where they communicate and collaborate. Many apps require a “back and forth” communication or passing items between people. For example, in the Inspection sample app template for Teams, managers create inspection forms, users complete the inspection, and reviewers review the inspections.

Power Apps in Teams can leverage deep integration with other Teams capabilities to make apps collaborative. In this article, we'll learn how we can integrate Power Apps with Teams to make calls or schedule meetings with other users within the organization.

Watch this video to learn how to integrate Power Apps with calls and meetings: https://learn-video.azurefd.net/vod/player?id=022e5ad1-7a6d-49bb-99de-93dd4041bebd&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fintegrate-calls-and-meetings

Prerequisites

To complete this lesson, we'd need the ability to create apps within Teams that will be available as part of select Microsoft 365 subscriptions.

Log in into Teams

Log in into Teams using either the Desktop app or the web app

Create a new team

In this section, we'll create a new Teams team, and then create an app within that team. Skip this section if you already have a team that you want to use.

1. To create a new team, select the Teams tab > select **Join or create a team** on the bottom-left side of the screen > select **Create Team** > select **From Scratch** > select **Public**, and give the team a name such as "Calls and Meetings Integration". And then, select **Create**.

2. If you have any members in your organization you'd like to add to the team for testing purposes, add them in the pop-up, and then select **Add**.

3. Select **Close**.

The new team gets created and is listed under the Teams tab

Create a new app

Well create an app with a gallery of users and buttons on each of the display forms to either call, chat or schedule a meeting with a user from the list.

1. Open Teams app.
2. Select **Power Apps** from the left-pane.
3. Select **+ New app** under **Recent apps**.
4. Select the team created earlier, and select **Create** to open the app in Power Apps Studio.
5. Enter a name for the app, such as "Integrate with calls and meetings".
6. Select **Save**.

   The app shows a default gallery
7. Select the gallery, and search for users in the **Add a data source** search box.
8. Select **Users** from the list of data sources.

   The list of users shows in the gallery on the left-side of the screen. The right-side of the screen shows details of the selected user record.
9. Select the app from the left-pane, and update the **OnStart** property of the app with the following formula:

   ```
   Power Fx
   ```

   ```
   Set(gblIsHostClientWeb,Param("hostClientType")="web")
   ```

   > ⓘ **Note**
   >
   > This formula checks if the user is accessing Teams from a web browser, desktop, or a mobile app.
10. Select **Screen1** from the Tree view.

11. Select + (Insert) > **Button** to add a button.

12. Move the button to the details pane on the right-side of the screen.

13. Set the following properties of the button.

Expand table

Property	Value
Text	"Call"
Width	150
Height	45
X	500
Y	300

Copy the following formula in the **OnSelect** property of the button:

Power Fx

```
If(
    gblIsHostClientWeb,
    Launch(
        "https://teams.microsoft.com/l/call/0/0?
users="&BrowseGallery1.Selected.'Primary Email',
        {},
        LaunchTarget.New
    ),
    Launch(
        "msteams://teams.microsoft.com/l/call/0/0?
users="&BrowseGallery1.Selected.'Primary Email',
        {},
        LaunchTarget.New
    )
)
```

Note

- If the user is accessing Teams using a web browser, this will launch the call window in the browser itself because of `https://` protocol identifier. If the user is accessing Teams using the Teams desktop or mobile app, then it launches the call window in the Teams app itself because of `msteams://` instead.

- The format to launch a call through a deep-link is:
  `https://teams.microsoft.com/l/call/0/0?users=\<user1\>,\<user2]`.
  More information: Generate a deep link to a call

14. Add another button with the following properties:

Expand table

Property	Value
Text	"Schedule a Meeting"
Width	150
Height	45
X	Button1.X + 200
Y	Button1.Y

15. Copy the following formula in the **OnSelect** property of the new button:

Power Fx

```
If(
    gblIsHostClientWeb,
    Launch(
        "https://teams.microsoft.com/l/meeting/new?
subject=New%20Meeting&attendees="&BrowseGallery1.Selected.'Primary
Email',
        {},
        LaunchTarget.New
    ),
    Launch(
        "msteams://teams.microsoft.com/l/meeting/new?
subject=New%20Meeting&attendees="&BrowseGallery1.Selected.'Primary
Email',
        {},
        LaunchTarget.New
    )
)
```

16. Add a third button with the following properties

Expand table

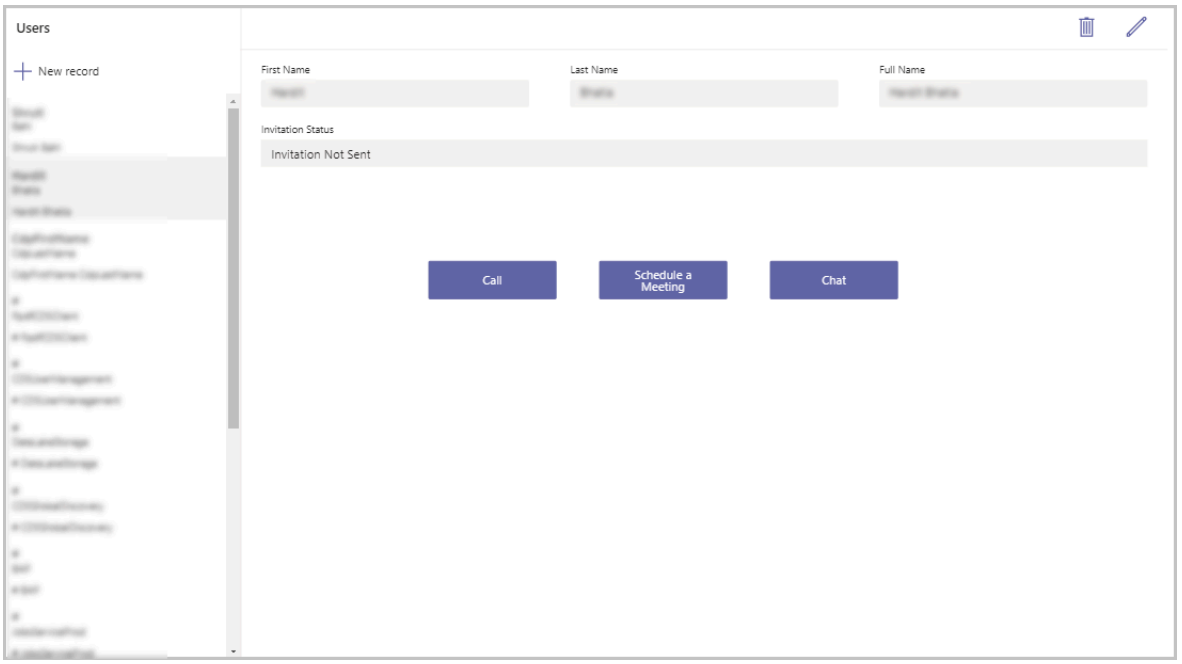
Property	Value
Text	"Chat"
Width	150
Height	45
X	Button2.X + 200
Y	Button1.Y

17. Copy the following formula in the **OnSelect** property of the new button:

```
Power Fx
```

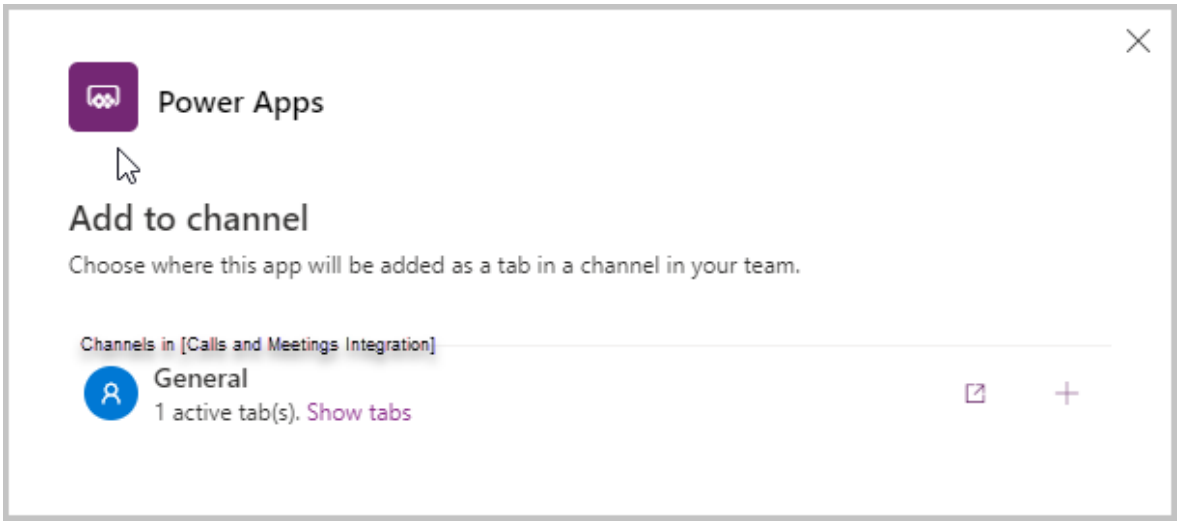
```
If(
    gblIsHostClientWeb,
    Launch(
        "https://teams.microsoft.com/l/chat/0/0?
users="&BrowseGallery1.Selected.'Primary
Email'&"&topicName=&message=Hi!",
        {},
        LaunchTarget.New
    ),
    Launch(
        "msteams://teams.microsoft.com/l/chat/0/0?
users="&BrowseGallery1.Selected.'Primary
Email'&"&topicName=&message=Hi!",
        {},
        LaunchTarget.New
    )
)
```

The screen should appear as shown below



Save and publish the app

1. Select **Save** on the top-right side of the screen to save the app.
2. Select **Publish** to publish the app.
3. Select **Next** in the pop-up.
4. Under **Add to Channel**, make sure the channel under which the app should be shown is listed and shows as "1 active tab(s)" and not "0 active tab(s)".



5. Select **Save and Close**.

Test the app

1. Run the app in the preview mode.

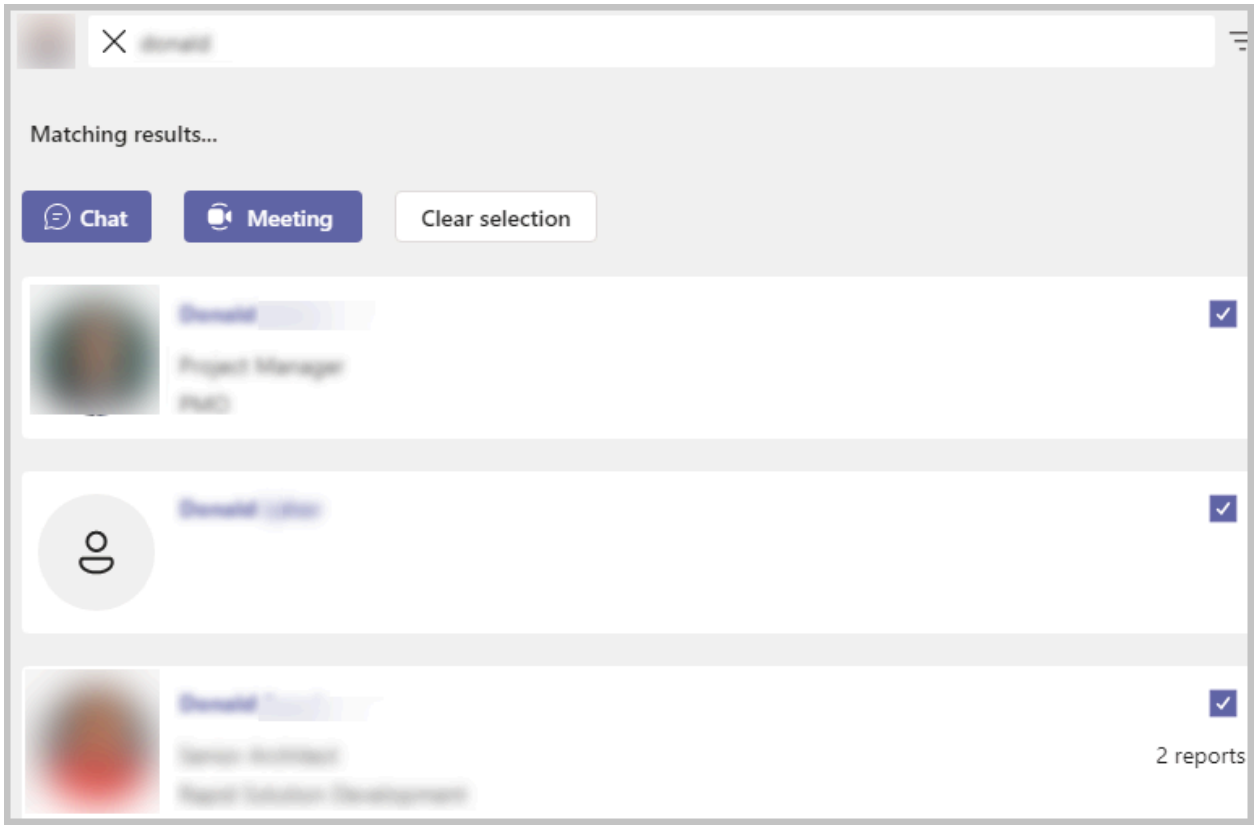
2. Select a user from the list on the left.
3. If you don't have a user showing up in the list, share the app with another colleague/user. Once that user logs in into the app, they'll show up in this list of users.
4. Select the user from the list.
5. Test the options.

The call button launches a new call. The meeting button creates a meeting invitation with the selected person

See chat, call, and meeting integration in action

For an example of chat, call, and meeting integration in action, check out the Profile + sample app for template Teams.

When opened from within a Team, the app displays the other members of the team. You can select the **Select people to connect with** and select one or more people, then start a call or chat with the selected people, or schedule a meeting.



Feedback

Was this page helpful? Yes No

Provide product feedback

Enable consistent experience across sessions using LoadData and SaveData functions

Article • 01/20/2023

Microsoft Dataverse for Microsoft Teams delivers a built-in, low-code data platform for Teams. It provides relational data storage, rich data types, enterprise-grade governance, and one-click solution deployment. Dataverse for Teams enables everyone to easily build and deploy apps.

While there are numerous advantages of using Power Apps in Teams, one of the biggest design considerations is the ability to keep consistent experience across sessions.

While using an app, if we switch a tab in Teams to look at Wiki or Planner, or even chat with someone, the session data of the app is lost and we'd have to start the session from scratch. For example, if we're in the middle of an inspection using the Inspections app and we switch tabs even for a moment, all the data that was captured on the inspection would be lost and we'll have to recapture all that information.

In this article, we'll learn how to use the LoadData() and SaveData() functions to store the session data within the app to avoid any loss of work because of switching tabs in Teams.

Watch this video to learn how to enable consistent experience across sessions: https://learn-video.azurefd.net/vod/player?id=e669c7fe-3f11-43ad-8233-0dbbb374bec5&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fconsistent-experience-across-sessions

Prerequisites

To complete this lesson, we'd need the ability to create apps within Teams that is available as part of select Microsoft 365 subscriptions.

Log in into Teams

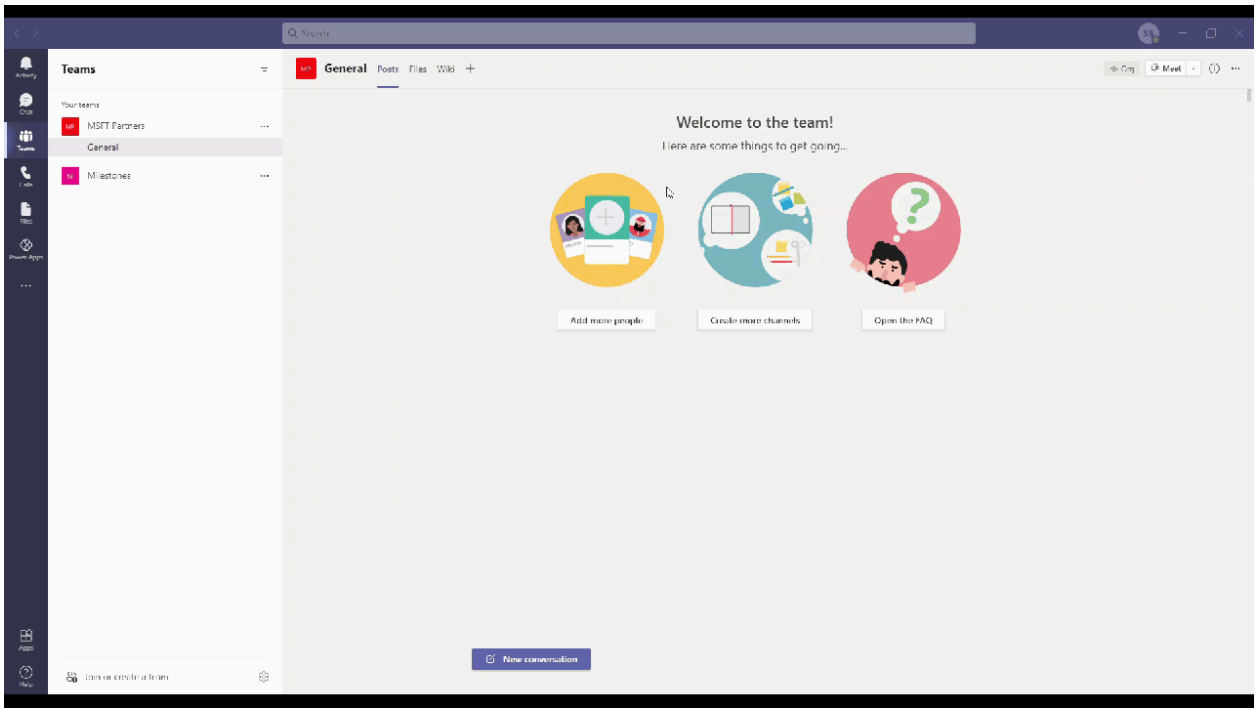
Log in into Teams using Teams Desktop.

Create a new Team

To create a new team:

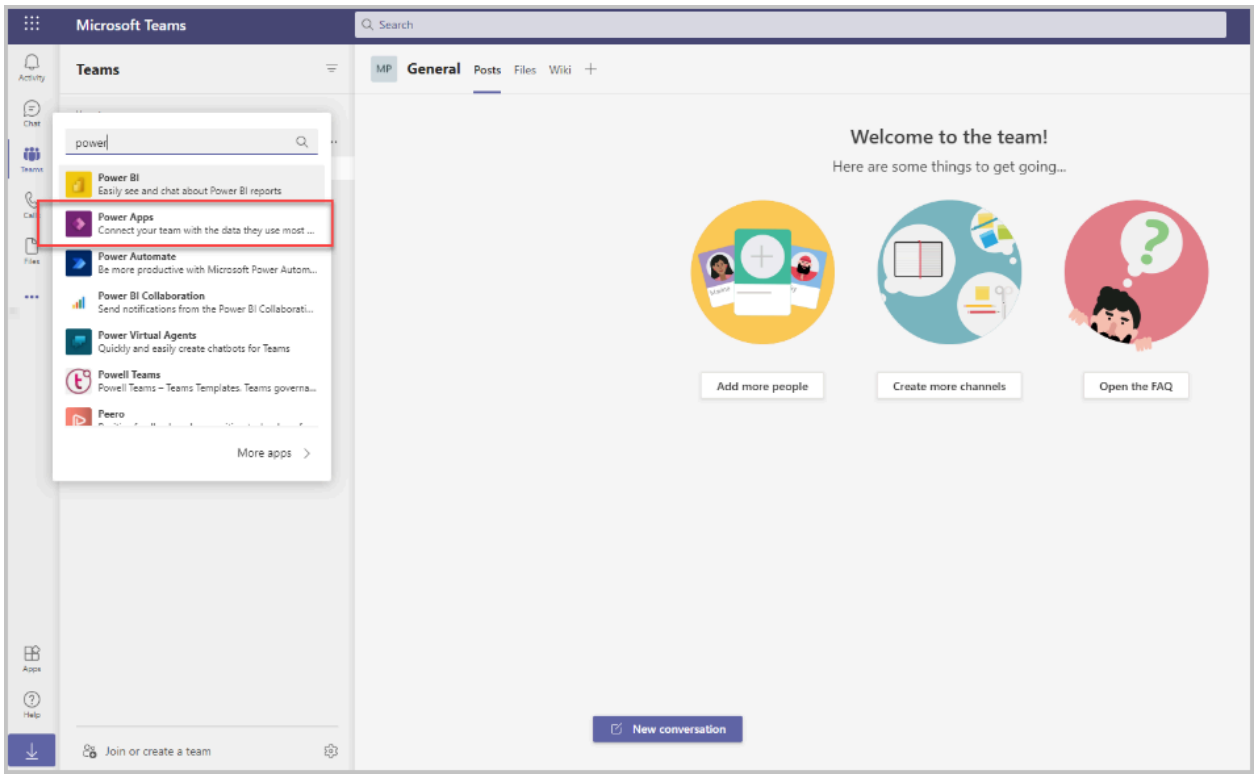
1. Select the Teams tab.
2. Select **Join** or create a team on the bottom-left of the screen.
3. Select **Create a team from scratch**.
4. Select **Public** and give the team a name. In this example, we'd use "MSFT Partners".
5. Select **Create**.

The new team gets created and is listed under the Teams tab.

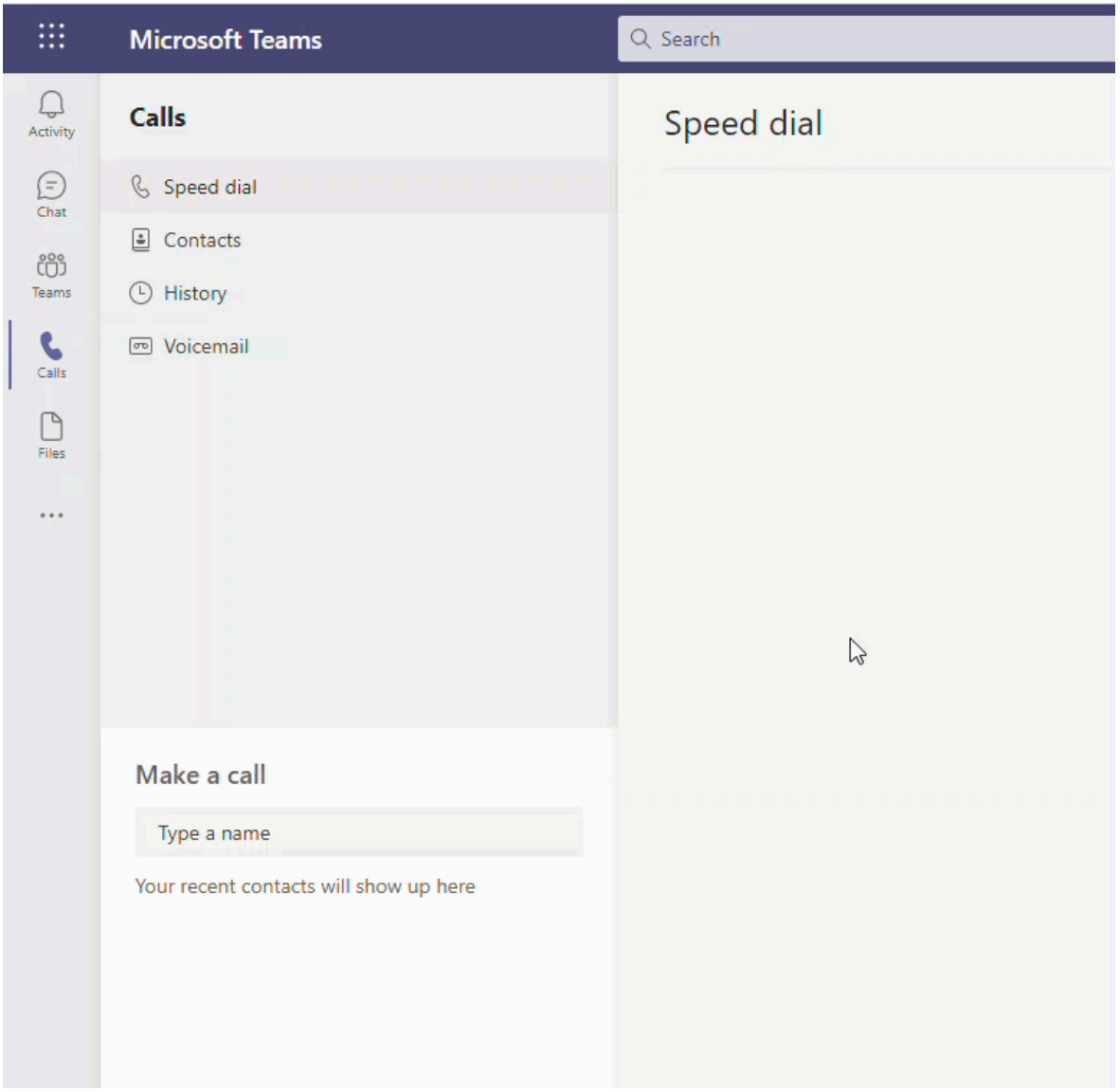


Create a Power Apps app in Teams

To create a Power Apps app in teams, we need to open Power Apps Studio in Teams as shown below.



1. Select **Power Apps** from the list of apps and pin it to the left-pane, and then select it.



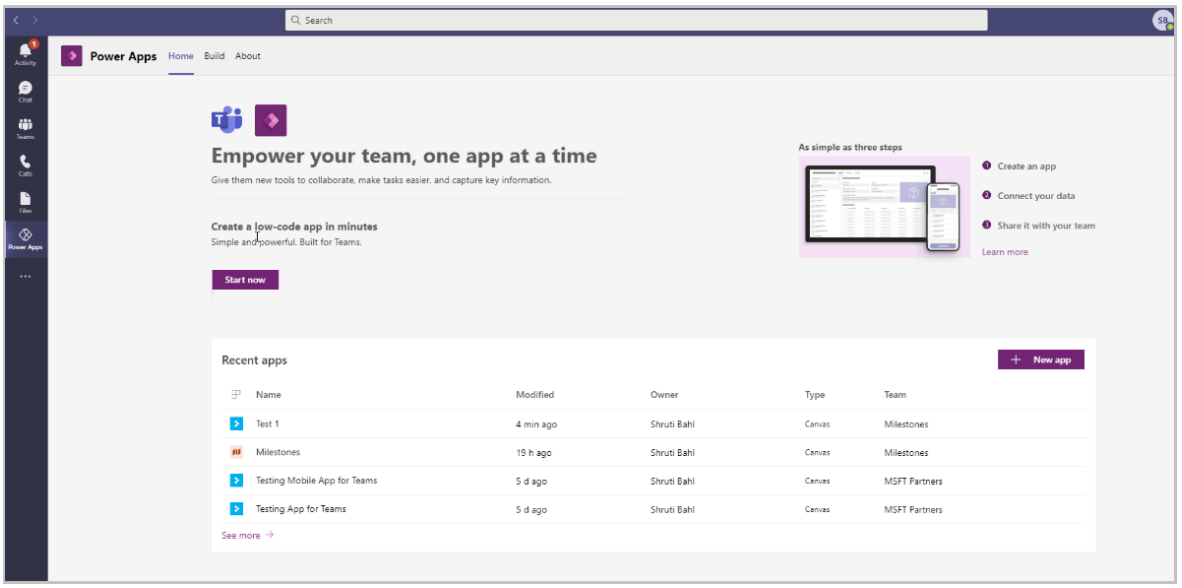
2. Select **Start now** to create the first app.

3. Select the team for which this app is to be created—a Power Apps environment is created for that team (there's one Power Apps environment per team).

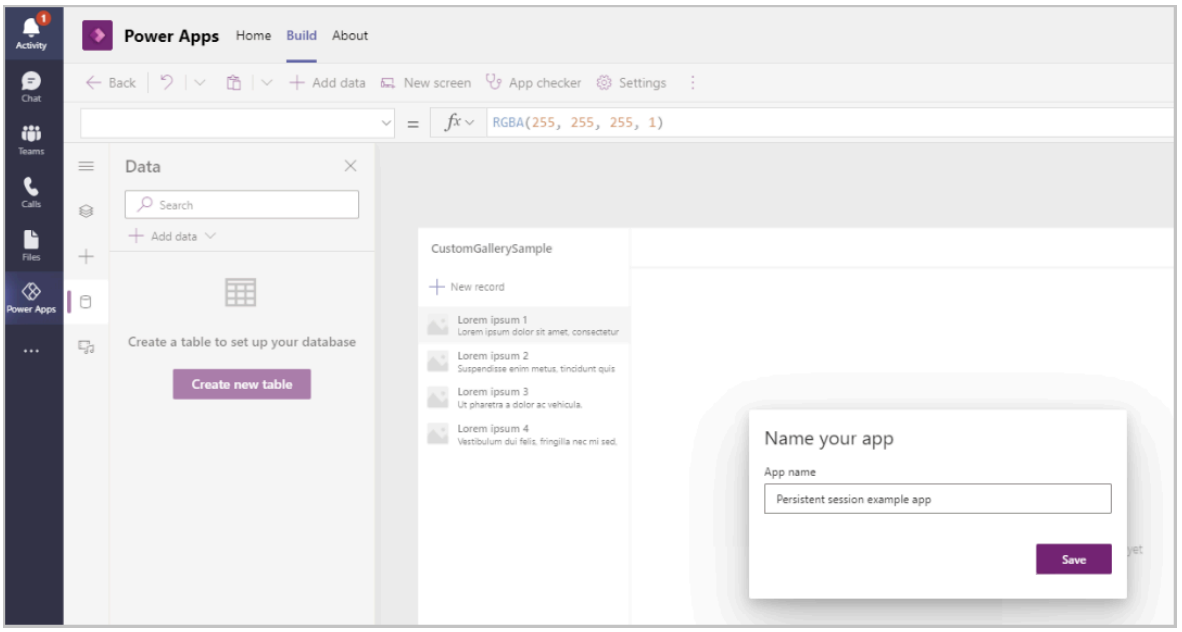
4. While the environment is being created, close the dialog box that says **Getting things ready**.

5. Select the **Build** tab on top, and in a few seconds the created app should appear in the list.

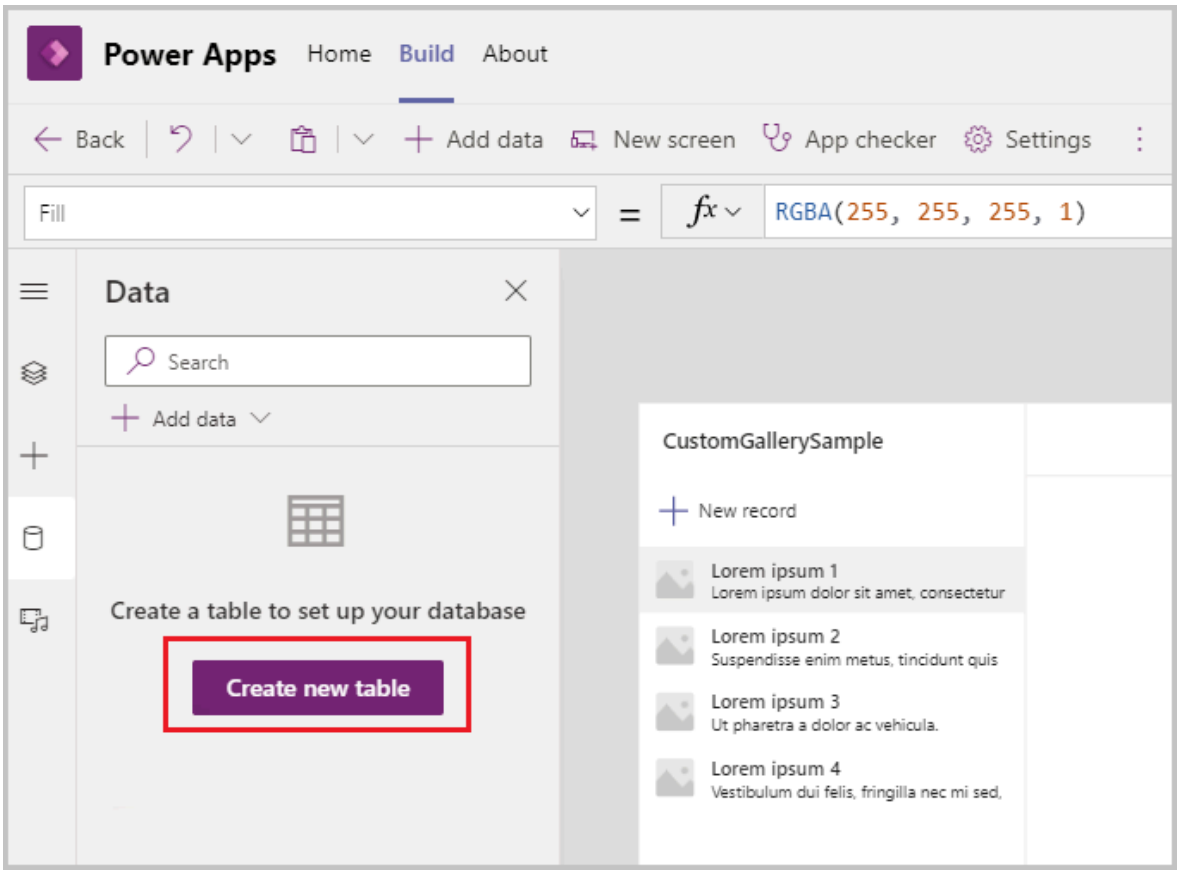
6. Open the app, and notice it gets created in the tablet layout/form factor by default.



7. Enter a name for the app **Persistent session example** app, and select **Save**.

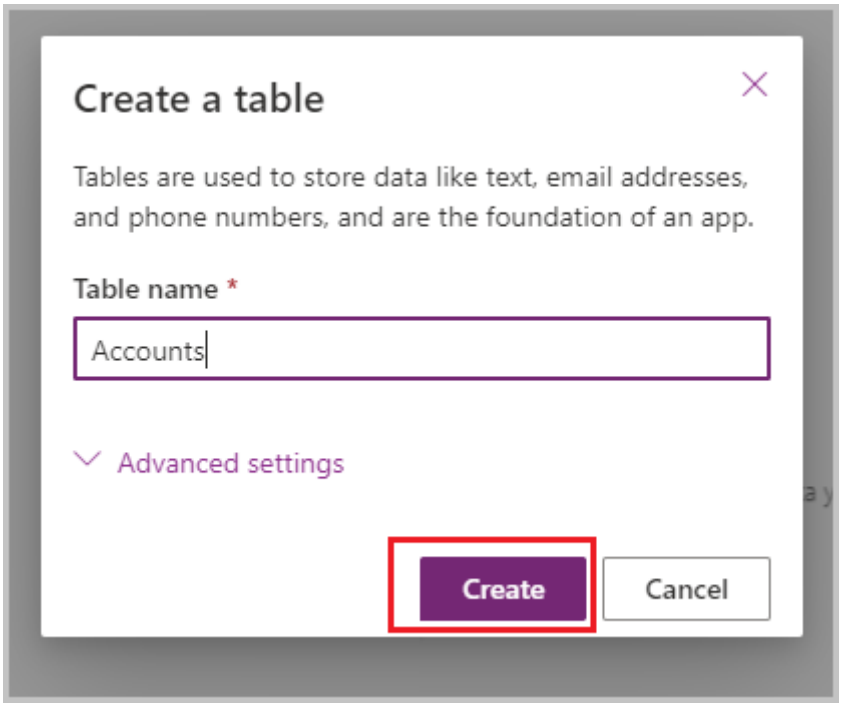


The app gets created with a gallery.

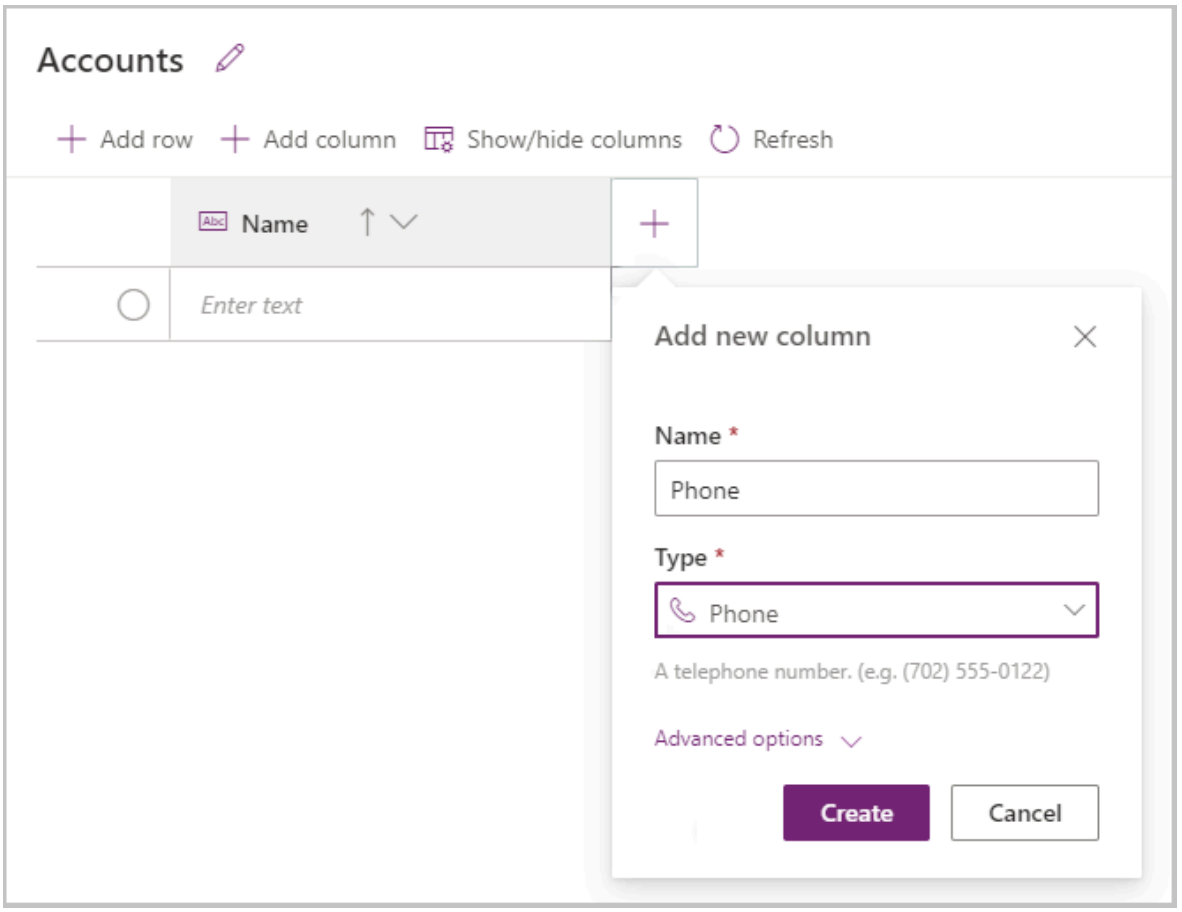


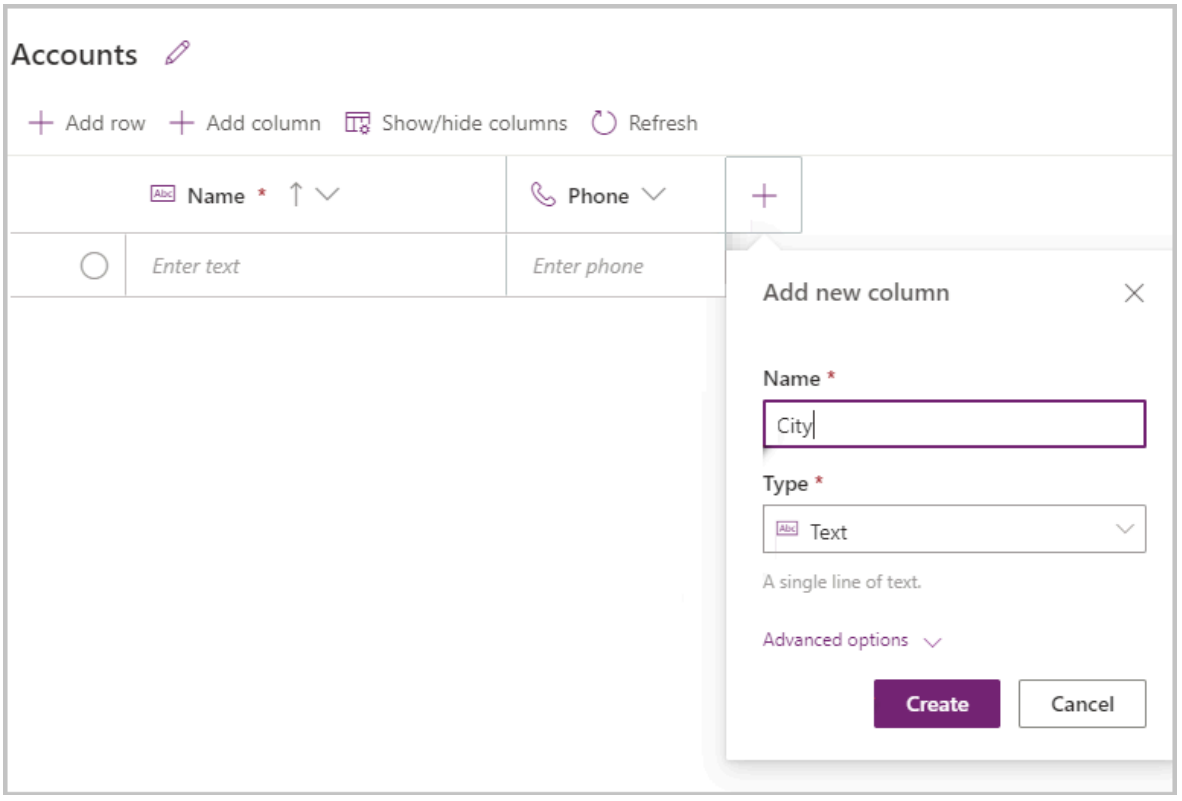
8. Select **Create new table** to create a new table.

9. Enter **Accounts**, and select **Create**.



10. Add two columns—**Phone** with data type of Phone, and **City** with data type of Text.

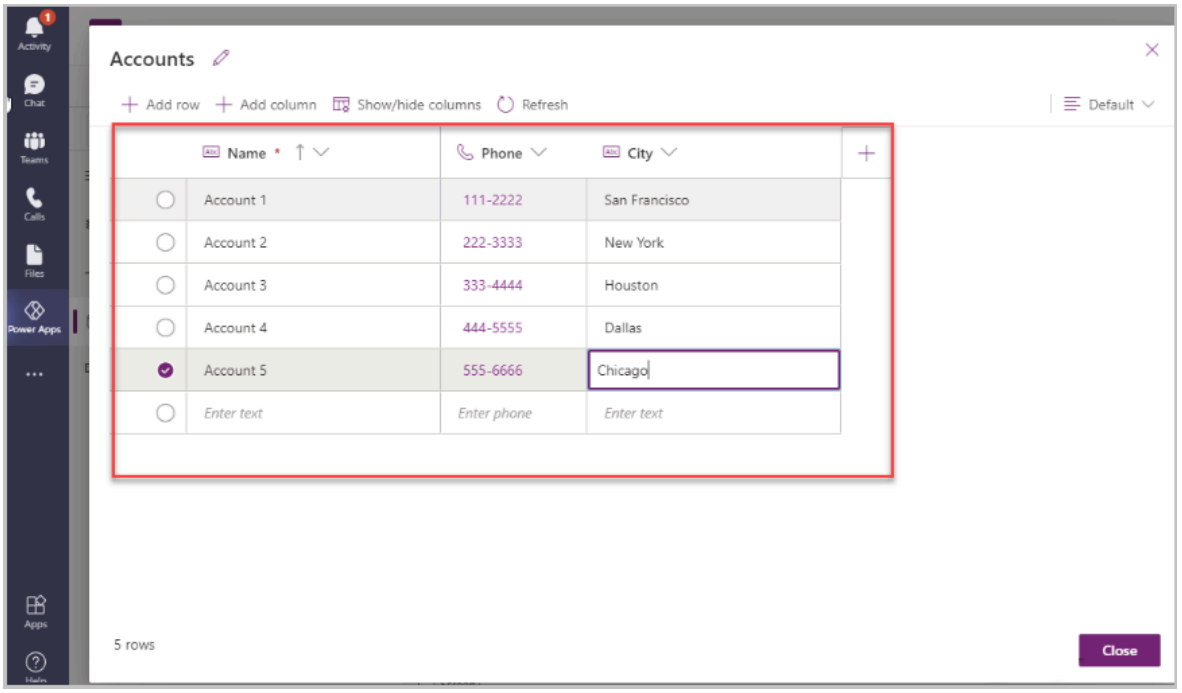




11. Add sample data to the three columns.

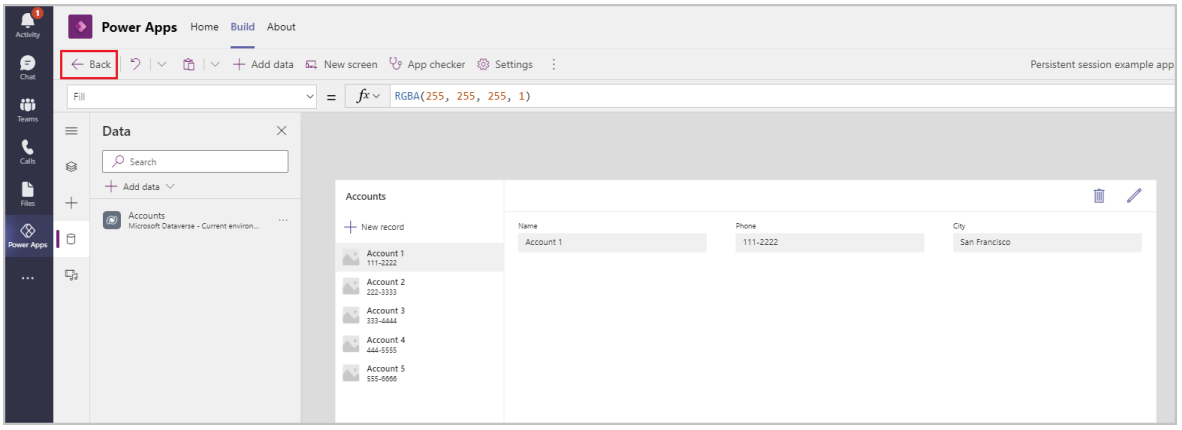
12. Repeat to add at least five records.

13. Select **Close**.



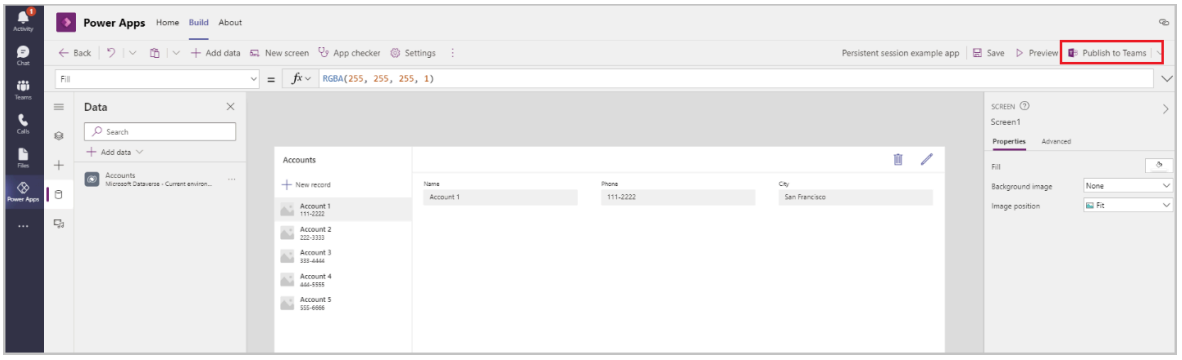
The accounts table gets added as a data table

14. Select **Tree View** > Select **BrowseGallery1** > Select **Accounts** as the data source.

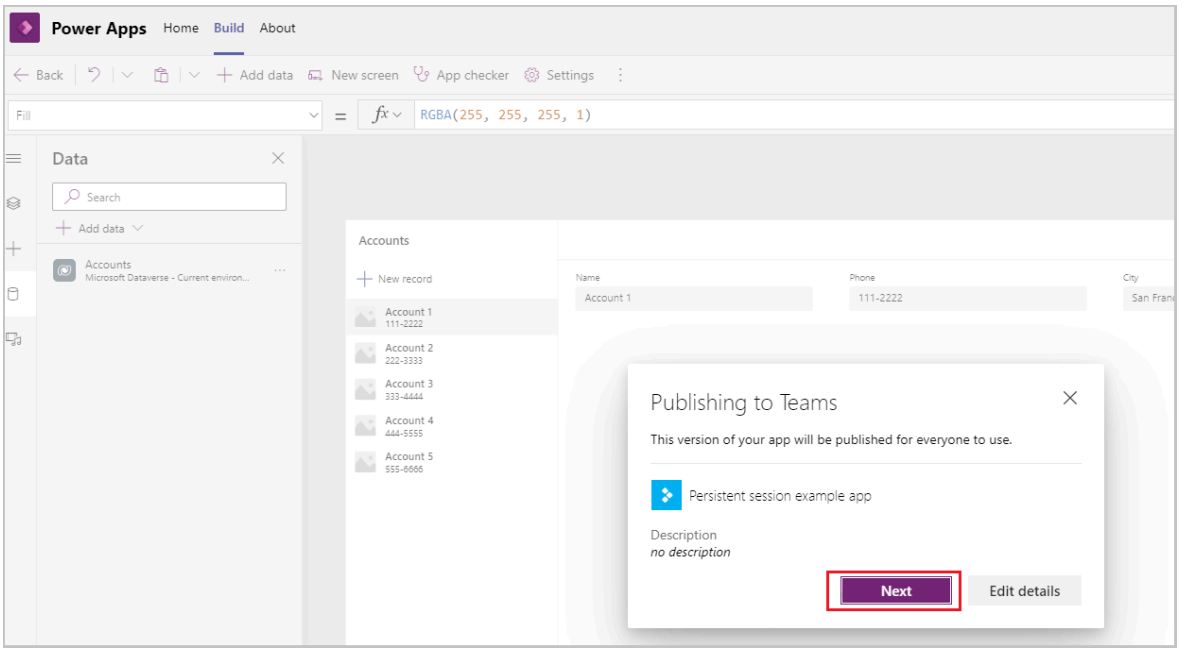


Publish the app in Teams

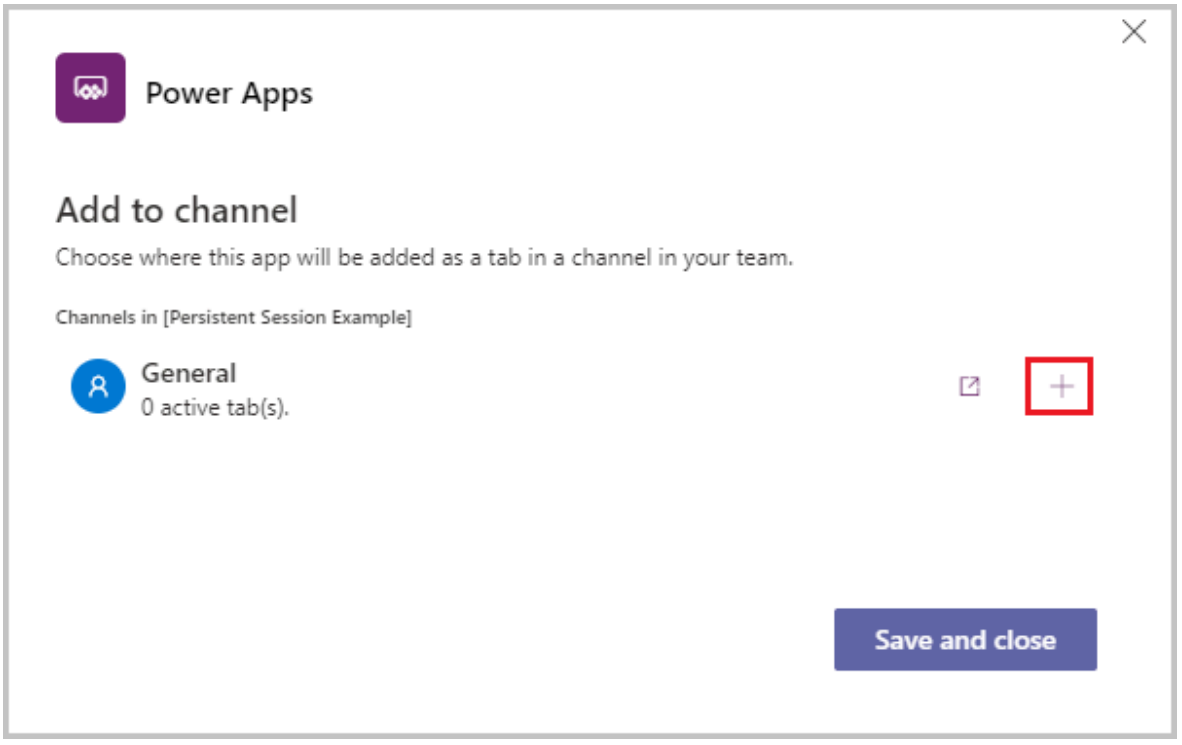
1. Select **Publish to Teams**.

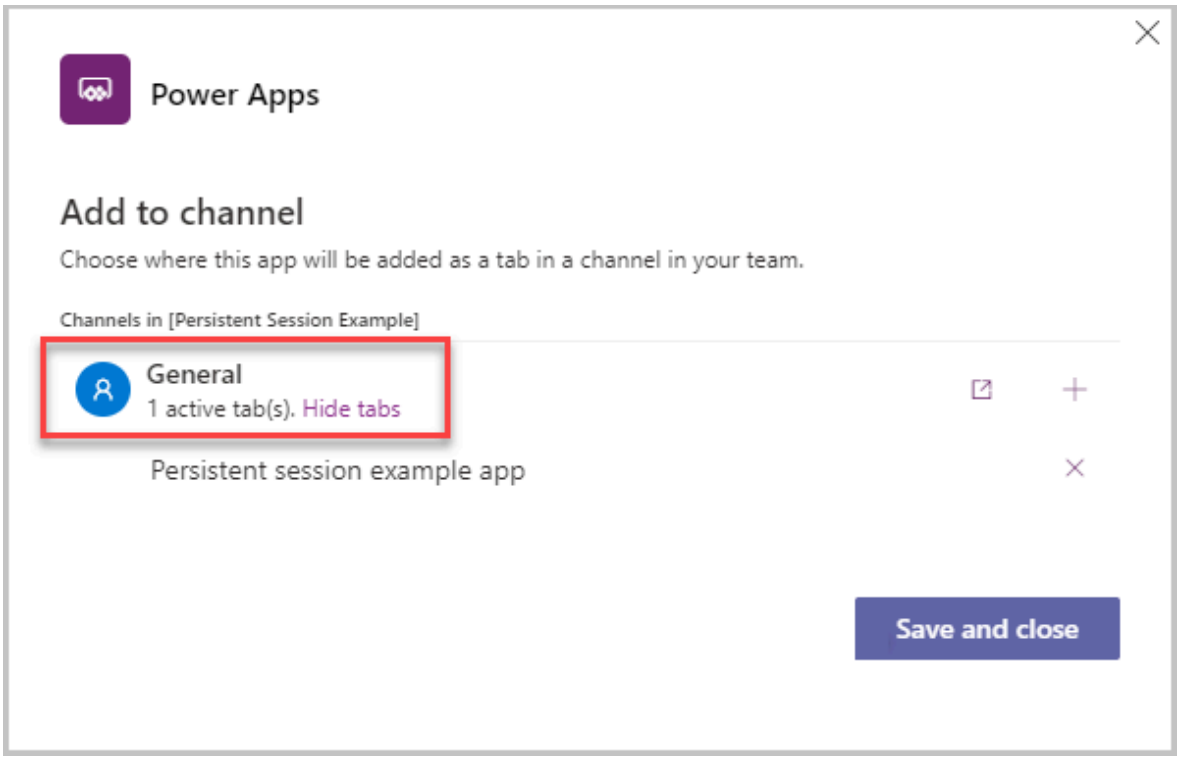


2. Select **Next**.

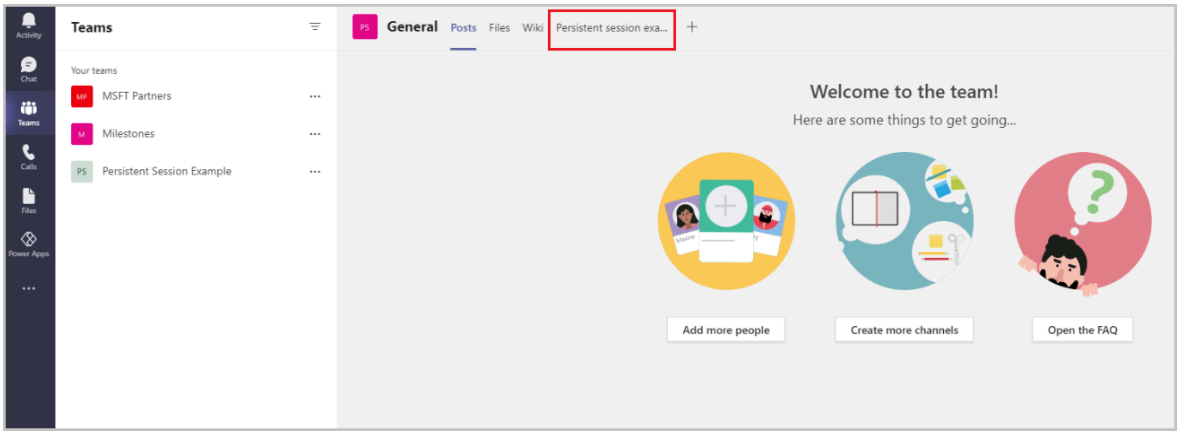


3. Select + to add the app to the "General" channel.



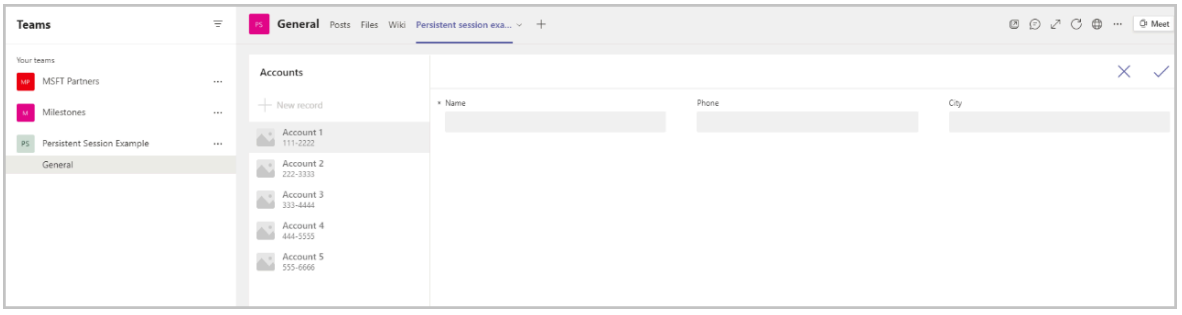


4. Go to the **Persistent Session Example** team, and notice the **Persistent session example** app shows up on the top as a tab.



Testing the issue

1. Open the **Persistent session example** app from the **Persistent Session Example** team.

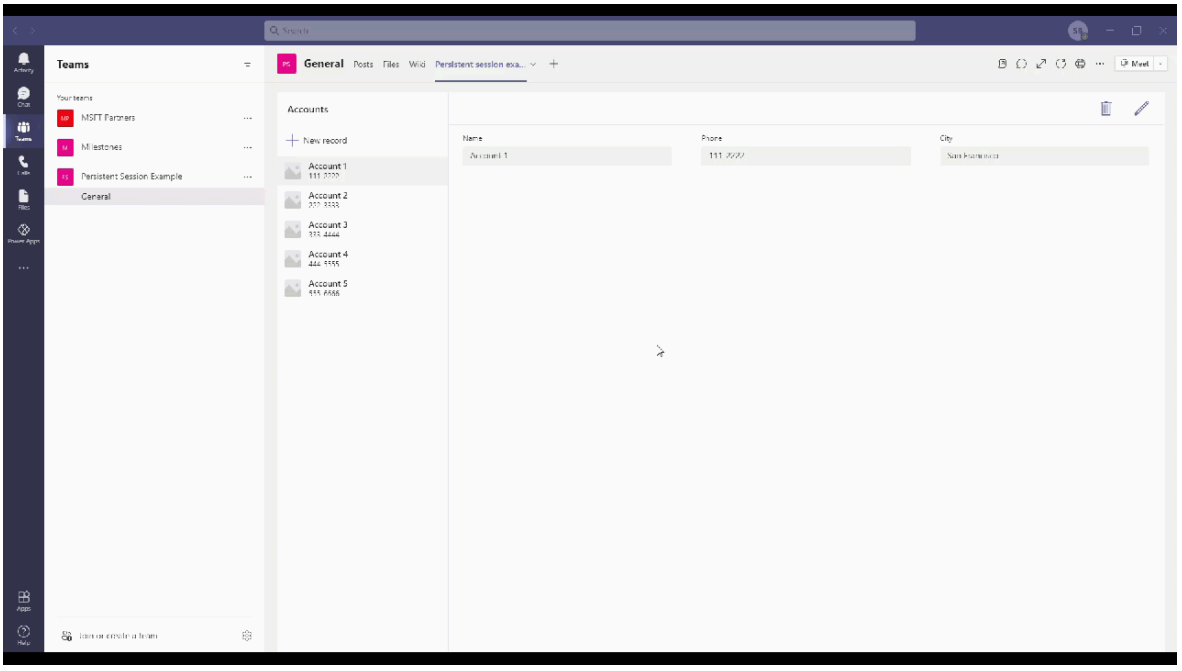


2. Select **New Record** to create a new account in the app.

3. Enter the name and phone number

4. Before you enter **City** and save the record, switch to a different tab (like Files or Wiki on top or a different team), and then come back to the app.

5. Notice that the data entered in the two columns is no longer available on the screen and is lost



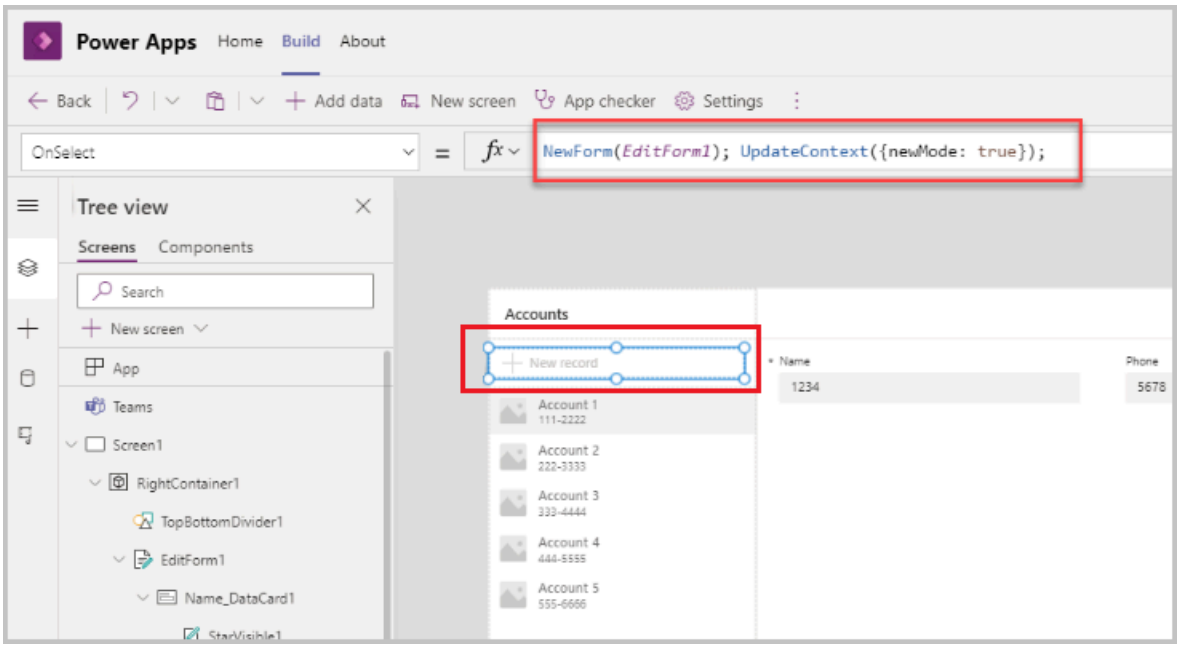
Solution: Use SaveData() and LoadData() functions

To resolve the above issue, we'll use the LoadData() and SaveData() functions. We'll create a collection for the data entered on a form as and when data is entered, and keep saving the data in the collection until the record has been committed or saved. This way, if the record hasn't been saved and we must tab out of the screen for some reason, the collection would have the saved data that would get loaded once we're back in the app. We'll also clear the collection once the record has been saved so that the collection is ready to capture new data when the next new record is being created.

1. Start with making sure that the changes we're applying are applicable only in the case of a new record. So, we're updating the variable **newMode** that will be used as a condition on the edit form to figure out what data needs to be displayed. Select **New record** and add the following formula on the **OnSelect** property of the button:

```
Power Fx
```

```
NewForm(EditForm1);
UpdateContext({newMode: true})
```

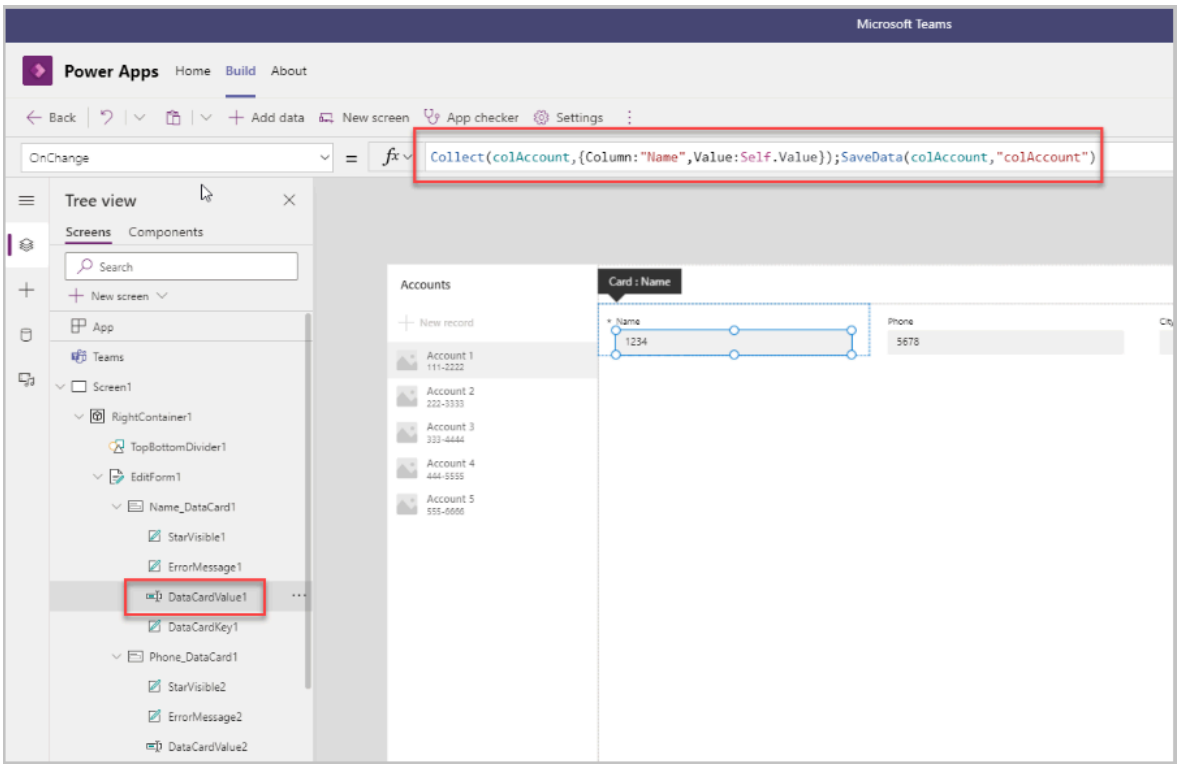


2. Next, we'll add the data from the individual input controls to a collection, and then we'll save the data to a local file from that collection. Select the text input box—in our example it's the "DataCardValue1" of the "Name_DataCard1" on "EditForm1" for the **Name** field—enter the following formula in the **OnChange** property of the input field:

Power Fx

```
Collect(
    colAccount,
    { Column:"Name",Value:Self.Value}
);
SaveData(colAccount,"colAccount").
```

With this change, we're adding the data to a local cache so that if the user leaves the app, their data won't be lost.



Similarly, enter the following code in the **OnChange** property for **Phone** (DataCardValue2):

```
Power Fx

Collect(colAccount, {Column:"Phone",Value:Self.Value});
SaveData(colAccount,"colAccount");
```

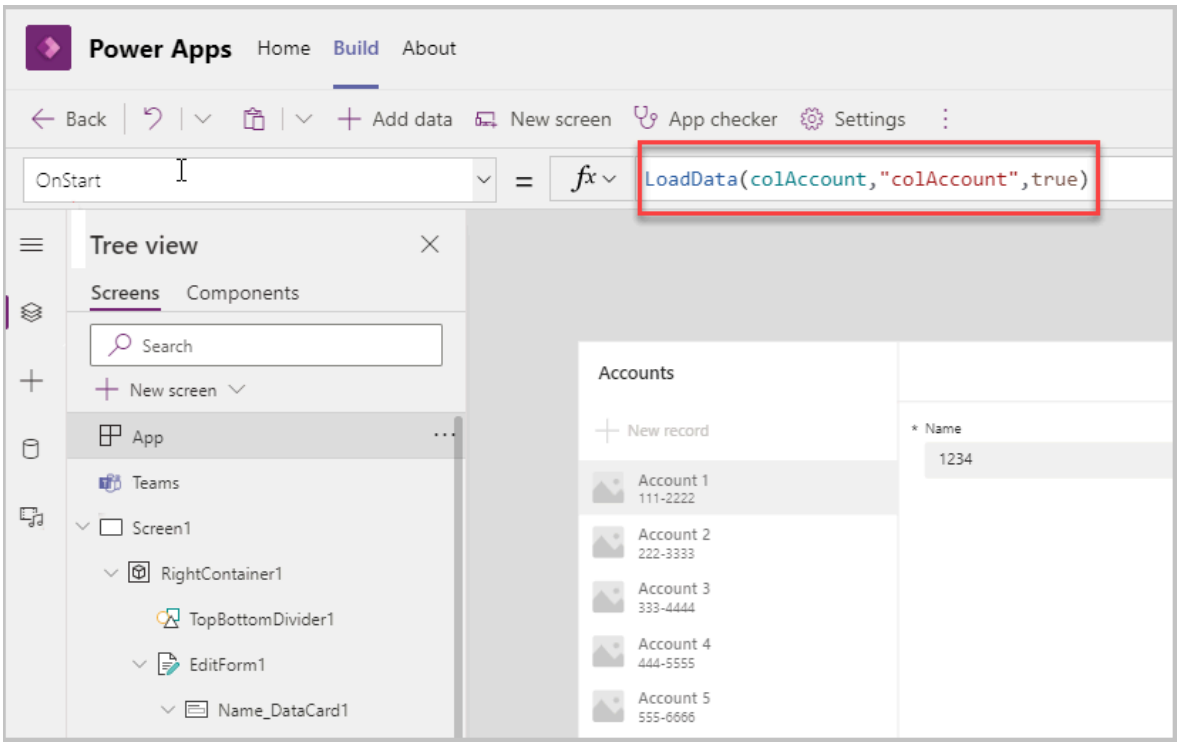
And enter the following code in **OnChange** property for **City** (DataCardValue3):

```
Power Fx

Collect(colAccount {Column:"City", Value:Self.Value});
SaveData(colAccount, "colAccount");
```

3. Now we'll update the App OnStart so that if there's any data saved in the collection, that would get loaded on the start of the app.

   Go to the **App** > **OnStart** property, and add the following formula:

```
Power Fx

LoadData(colAccount,"colAccount", true)
```



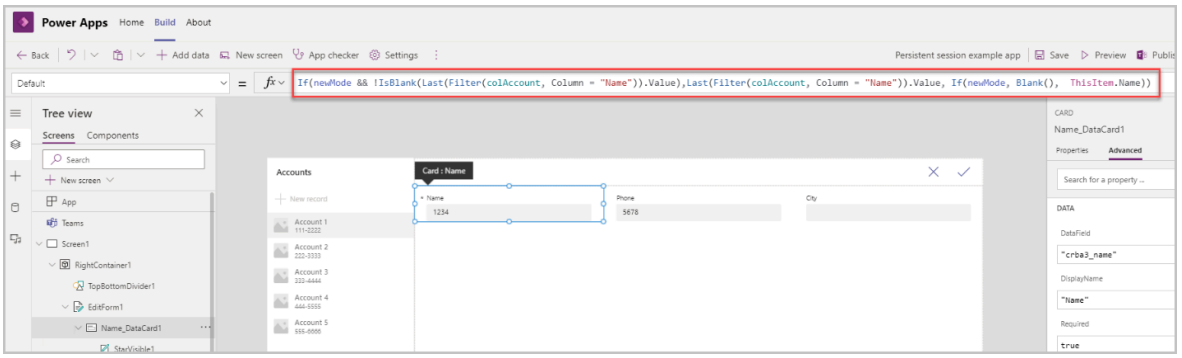
4. Next, we need to update the **Default** property of the data fields so that when the page loads once we come back to the app, if data exists in the collection, it should show the data from the collection by default.

   Select the field **DataCard** and update the **Default** property of the data card to the following formula for the listed fields.

   For **Name** field, enter **Name_DataCard1:**

```
Power Fx
```

```
If(
    newMode && !IsBlank(
        Last(
            Filter(colAccount, Column = "Name")
        ).Value
    ),
    Last(
        Filter(colAccount, Column = "Name")
    ).Value,
    If(newMode, Blank(), ThisItem.Name)
)
```



Similarly, enter the following formula in the **Default** property formula for **Phone** (Phone_DataCard1):

```
Power Fx
```

```
If(
    newMode && !IsBlank(
        Last(
            Filter(colAccount, Column ="Phone")
        ).Value
    ),
    Last(
        Filter(colAccount, Column = "Phone")
    ).Value,
    If(newMode, Blank(), ThisItem.Phone)
)
```

And use the following formula for **City** (City_DataCard1):

```
Power Fx
```

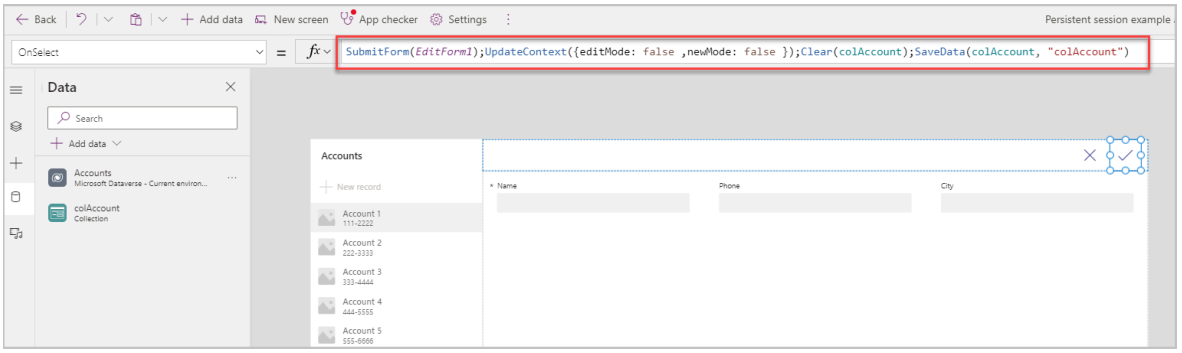
```
If(
    newMode && !IsBlank(
        Last(
            Filter(colAccount, Column = "City")
        ).Value
    ),
    Last(
        Filter(colAccount, Column = "City")
    ).Value,
    If(newMode,Blank(), ThisItem.City)
)
```

5. Now, we need to clear the collection in two scenarios:
   a. When the user submits the form, and the changes get saved.
   b. When the user selects the cancel button to cancel the changes.

Use the following formula in the **OnSelect** property of the **Submit** button.

Power Fx

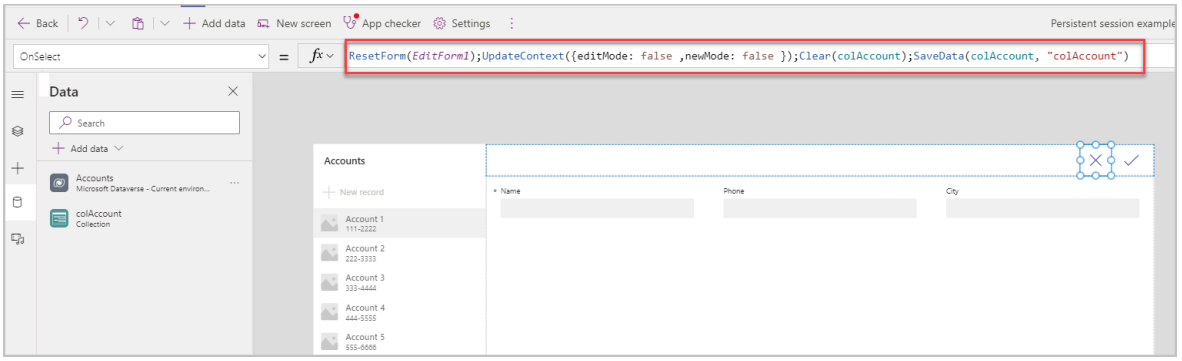
```
SubmitForm(EditForm1);
UpdateContext({editMode: false, newMode: false});
Clear(colAccount);
SaveData(colAccount, "colAccount");
```



Select the **Cancel** button, and enter the following formula:

Power Fx

```
ResetForm(EditForm1);
UpdateContext({editMode: false, newMode: false});
Clear(colAccount);
SaveData(colAccount, "colAccount");
```

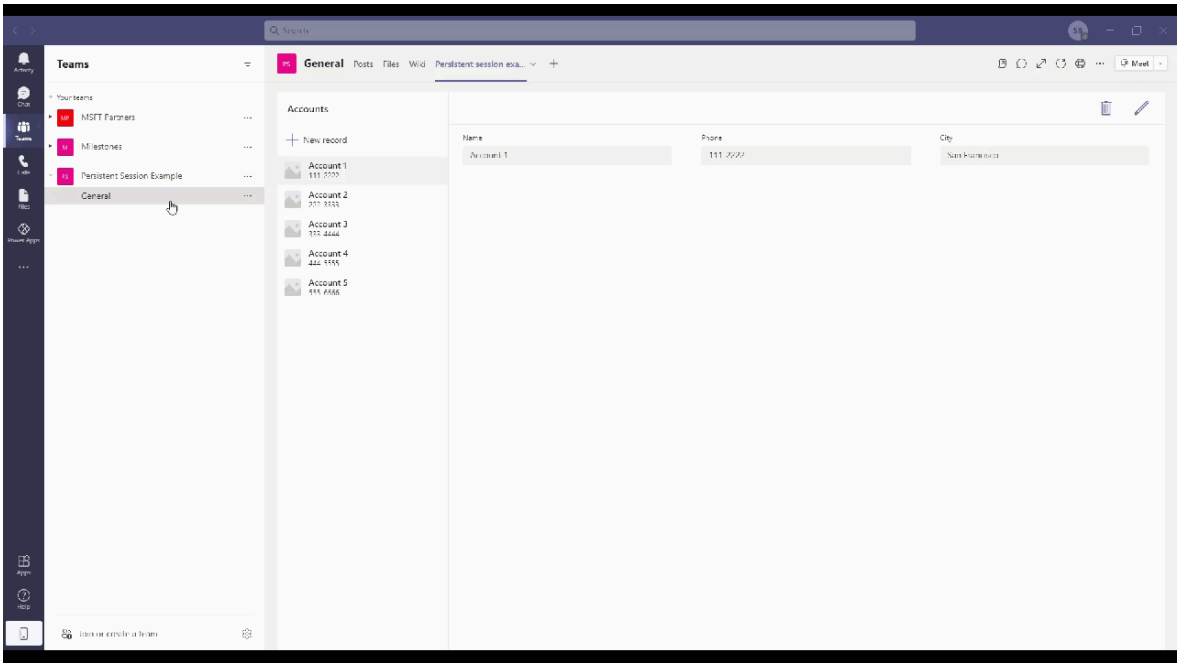


6. Publish the apps after making all the above changes by selecting **Publish to Teams**.

Test the app again

1. Go to the team and run the app.
2. Select **+ New** record, and add the name and phone details for the new record.
3. Select another tab to move out of the app.
4. Come back to the app—it shows details of the first account.

5. Now, when we select + **New** again, we'll see the Name and Phone details populated in those text input fields already. And we can resume adding the account we were adding before we had to tab out of the app.



How we use SaveData and LoadData in the Inspection app

In the Inspection sample app template, we use load data and save data on the inspection form—if a user is in the middle of an inspection, and they switch away from the apps in Teams (like to send someone a message), we don't want them to lose their place in the inspection. When they return to the app tab in Teams, the app gives them the option to resume the in-progress inspection.

Considerations

In Dataverse for Teams, there are some limits to the SaveData() and LoadData() capabilities.

- 1-MB limit on Teams Desktop.
- Doesn't work in a browser.
- Apps in mobile are limited by the amount of local storage available to the app.

Feedback

Was this page helpful? Yes No

Provide product feedback

Have a conversation about your business data in a Teams app

Article • 01/20/2023

Apps in Microsoft Teams drive collaboration by enabling users to do work where they communicate and collaborate. One way to make your apps collaborative is by adding the ability to have a conversation about your app or the data in your app. Teams is your go-to place to have such conversations.

For example, in the Profile+ sample app template for Teams, when you select one or more profile cards in the app, you can start a chat with the selected people directly in Teams.

You might also want to use this capability for your apps. For example, if you're building a sales opportunity tracker for the sales manager to review upcoming deals, you might want to add a button for the user to chat with the sales person about the opportunity.

This way, Power Apps in Teams can use deep integration with other Teams capabilities to make apps collaborative. In this article, we'll learn how we can integrate Power Apps with Teams to facilitate conversations.

Watch this video to learn how to configure having conversations about business data in Teams app: https://learn-video.azurefd.net/vod/player?id=aeacd56d-232b-4187-83f0-e15dc4103f43&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fbusiness-data

Prerequisites

To complete this lesson, we'd need the ability to create apps within Teams that will be available as part of select Microsoft 365 subscriptions.

Log in into Teams

Log in into Teams using either the Desktop app or the web app

Create a new Team

In this section, we'll create a new Teams team and then create an app within that team. Skip this section if you already have a team you want to use instead.

1. To create a new team, select the Teams tab, and then select **Join or create a team** on the left bottom of the screen.
2. Select **Create Team** > **From Scratch** > **Public**, give the team a name such as "Calls and Meetings Integration", and select **Create**.
3. (Optional) If you have any members in your organization that you'd like to add to the team for testing purposes, select them, and add them in the popup for adding members.
4. Select **Close**.

The new team gets created, and is listed under the Teams tab.

Create a new app

We'll create an app where the user can start a conversation for a particular team and Channel by selecting a button and typing in a message in a text box. The user can then select the **Submit** button to post the message in the particular team and channel in Teams, then other user can join the conversation.

1. Open Teams.
2. Select **Power Apps** from the left-pane.
3. Select **+ New app** under **Recent apps**.
4. Select the team created earlier, and then select **Create**.

   The app gets created and Power Apps Studio opens to allow editing the app.
5. Enter a name for the app, such as "Conversation app", and select **Save**.

   The app is created with a default gallery on Screen 1.

Add Teams as a connector

We need to add Teams as a connector to the app since we want to be able to access the teams and channels from the logged in user's account.

1. Select **Data** from the left-pane.
2. Select **+ Add data**, and then select **Connectors**.
3. Scroll down and select **See all connectors**.
4. Search for and select Teams connector.

Add a new table to capture company record

We need to add a table to maintain a list of companies we'll use as the company record, and to start a conversation about it.

1. Select **Data** from the left-pane.
2. Select **+ Add data** and then create a new table with the name "Company", and plural name as "Companies".
3. Add a few company names to the table, and then select **Close**.

Add a new screen to select the customer

Next, we'll add a screen to the app so that users can select the customer that they want to have the conversation about.

1. In the tree view, select **+ New screen** > select **Scenarios** tab > select **List**.

   A new screen gets added with a gallery list.
2. Update the title of the gallery to "Companies List".
3. Select **Companies** as the data source for the gallery.

The list of companies that was added shows up in the gallery **TemplateGalleryList1**.

4. Delete **Screen1** from the tree view, and rename the new screen to **Screen1**.

Add a new table to capture the conversation details

We need to add another table to capture the details such as the Teams conversation ID, team, and channel related to a conversation started in the app.

1. Select **Data** > **+ Add data** > **Create new table**.
2. Enter table name as **Conversation** and plural name as **Conversations**.
3. Change the default **Name** column name to **ID**.
4. Add the following columns to the conversation table:

Expand table

Column	Type
Team	Text
Team Channel	Text
Team Name	Text
Channel Name	Text
Company	Lookup, related table= Company

5. Save and close the table.

Add a new screen to start or join a conversation

Now, we'll add a new screen where the app user can start or join a conversation.

1. In the tree view, select **+ New screen** > **Blank layout**.
2. Update the **Fill** property of the screen to something other than white color.

   In our example, we've set the Fill property of the screen to `RGBA(224, 224, 237, 1)`.
3. Select **+** (Insert) > **Input** > **Button**.
4. Update the following properties of the button.

Expand table

Property	Value
Name	startaconversation_Button
Text	"Start a conversation"
Width	200
Height	32

Property	Value
Font size	12
X	`Parent.Width/2 - startaconversation_Button.Width/2`
Y	100
OnSelect	`Set(enterMessage,true)`

5. Select + (Insert) > **Input** > **Combo box**.

6. Update the following properties of the Combo box.

Expand table

Property	Value
Name	team_Combobox
Items	`MicrosoftTeams.GetAllTeams().value`
Width	320
Height	32
X	`Parent.Width/2 - team_ComboBox.Width/2`
Y	200
Text	"Team"
Tooltip	"Team"
Visible	enterMessage

7. Add another combo box and update the following properties.

Expand table

Property	Value
Name	channel_Combobox
Items	`If(!IsBlank(team_ComboBox.Selected.id),MicrosoftTeams.GetChannelsForGroup(team_ComboBox.Selected.id).value)`
Width	320
Height	32
X	`Parent.Width/2 - channel_ComboBox.Width/2`
Y	200
Text	"Channel"
Tooltip	"Channel"
Visible	enterMessage

8. Select + (Insert) > **Input** > **Text box**.

9. Update the following properties of the Textbox.

Expand table

Property	Value
Name	message_TextBox
Value	""
Width	500
Height	180
X	`Parent.Width/2 - message_TextBox.Width/2`
Y	300
Placeholder	Type message here
Visible	enterMessage

10. Select + (Insert) > **Input** > **Button**.

11. Update the following properties of the button.

Expand table

Property	Value
Name	submit_Button
Value	"Submit"
Width	96
Height	32
Font size	12
X	`Parent.Width/2 - submit_Button.Width/2`
Y	500
Visible	enterMessage

12. Copy the following formula in the **OnSelect** event of the button.

Power Fx

```
Patch(Conversations,Defaults(Conversations),
{ID:MicrosoftTeams.PostMessageToChannelV3(team_ComboBox.Selected.id,channel_ComboBox.Selecte
d.id,{content:message_TextBox.Value,contentType: "text"},{subject:"New
conversation"}).id,Team:team_ComboBox.Selected.id,'Team
Channel':channel_ComboBox.Selected.id, 'Team
Name':team_ComboBox.Selected.displayName,'Channel
Name':channel_ComboBox.Selected.displayName, Company: TemplateGalleryList1.Selected});
Set(enterMessage,false); Reset(team_ComboBox);Reset(channel_ComboBox);Reset(message_TextBox)
```

> **Note**

- In the above formula, we're using the **Patch** function to create a row in the **Conversations** table to capture the conversation ID along with the team, team name, channel and channel came. At the same time, we're also posting a message in the appropriate team and channel using the **PostMessageToChannelV3** function. We then set the **enterMessage** variable back to "false" to hide some of the fields and buttons whose visibility depends on this variable.
- The subject is hardcoded to “New conversation" to keep the app simple ,and not add another text box control to capture. But a text box can be added for subject to the screen and can be captured by the user. This formula will need to be adjusted accordingly to reference that control. The **Reset** function doesn't work for the **Fluent UI Combobox** control. This is needed to not have any team or channel selected once a message is posted.

13. Select + (Insert) > **Input** > **Button**.

14. Update the following properties of the button.

Expand table

Property	Value
Name	joinconversation_Button
Text	"Join conversation"
Width	200
Height	32
Font size	12
X	`Parent.Width/2 - submit_Button.Width/2`
Y	500
Visible	enterMessage

15. Copy the following formula in the **OnSelect** event of the button.

Power Fx

```
Launch(Concatenate("msteams://teams.microsoft.com/l/message/",Last(Sort(Conversations,
'Created On', Ascending)).'Team Channel',"/",Gallery1.Selected.etag,"?
tenantId=",Param("tenandId"),"&groupId=",Last(Sort(Conversations, 'Created On',
Ascending)).Team,"&parentMessageId=",LookUp(MicrosoftTeams.GetMessagesFromChannel(Last(Sort(
Conversations, 'Created On', Ascending)).Team,Last(Sort(Conversations, 'Created On',
Ascending)).'Team Channel').value,id = Last(Sort(Conversations, 'Created On',
Ascending)).Team).etag,"&teamName=",Last(Sort(Conversations, 'Created  On',
Ascending)).'Team Name',"&channelName=",Last(Sort(Conversations, 'Created On',
Ascending)).'Channel Name'),{},LaunchTarget.New)
```

Note

- In the above formula, we're using the **Launch** function to launch Teams using the team and channel parameters of the last record that was created in the **Conversations** table. We're doing this so that when a user starts a conversation and submits a message to post to a channel, the user can then select the **Join Conversation** button to go to the Teams post directly.

- We're using the last function here to keep it simple. If you anticipate using this app more extensively and anticipate having multiple conversations, you could get creative by using a gallery of conversations, and letting the user select which conversation they'd want to join.*

Update the gallery OnSelect Property

Next, we'll update the **OnSelect** property of the company gallery so that when a user selects a company from the list, the app will go to the second screen.

1. Select **Screen1** from the tree view to open the **Companies** gallery.
2. Set the **OnSelect** property of the gallery item to `Navigate(Screen2)`.

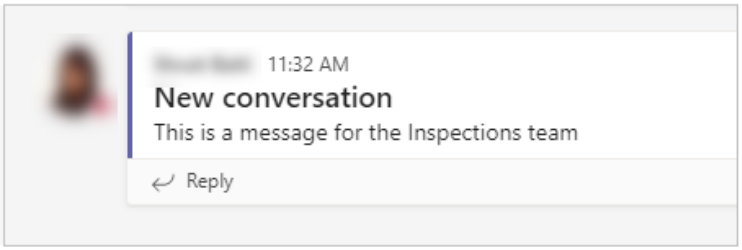
Save and publish the app

1. Select **Save** on the top-right to save the app.
2. Select **Publish** on the top-right to publish the app.
3. Select **Next**.
4. Under **Add to Channel**, make sure the channel under which the app should be shown is listed and shows as "1 active tab(s)" and not "0 active tab(s)".
5. Select **Save and Close** to complete the publishing of the app.

Testing the app

Run the app in preview mode or go to the team in which the app is created.

1. The Companies gallery should show up as the first screen.
2. Select one of the companies.
3. You should only see two buttons: **Start a conversation** and **Join a conversation**.
4. Select **Start a conversation**.
5. Additional fields should show up:
    - Team (dropdown with a list of teams)
    - Channel (dropdown list of channels within the selected team)
    - Message box (text box to type in the message to be sent to the team)
    - Submit button (to submit the message)
6. Select a team.
7. Select a channel within the team.
8. Enter message.
9. Select **Submit**. All the additional fields/controls get hidden.
10. Select **Join conversation** to be taken to the team and channel chat where the last message was sent.



Feedback

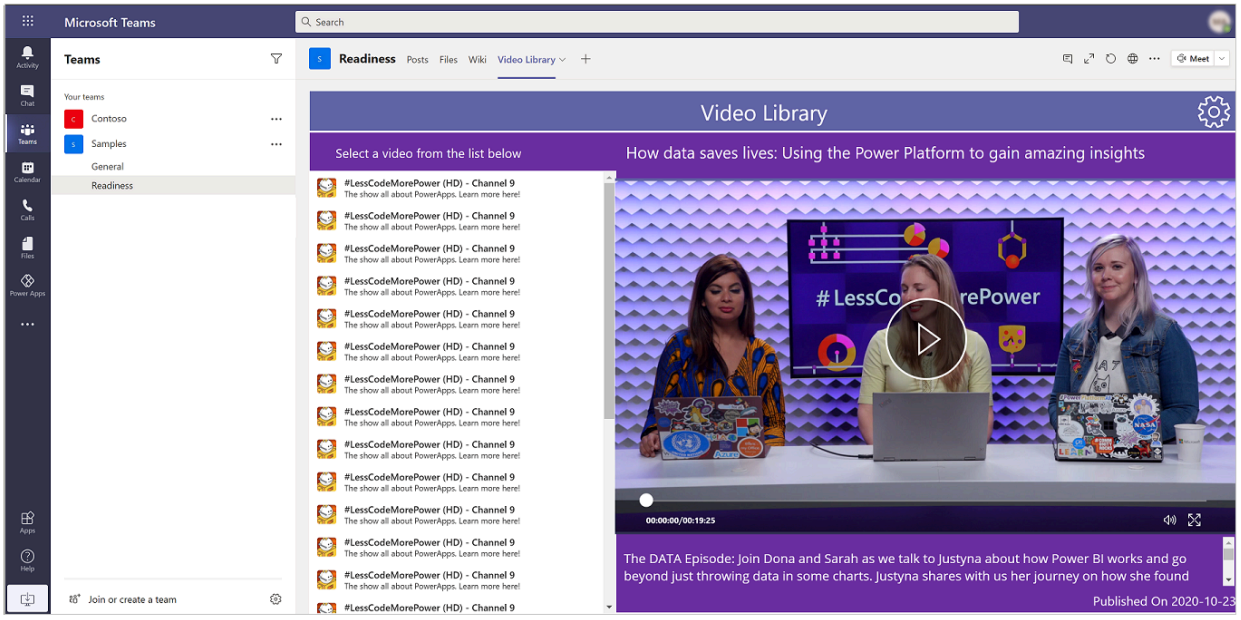
Was this page helpful? Yes No

Provide product feedback

Build an app to retrieve Power Platform videos for readiness

Article • 02/04/2023

In this tutorial, you'll build a solution that retrieves, lists, and displays content that can be used for readiness.



Objective

The focus of this tutorial will be on finding, retrieving, browsing, and playing video content available on YouTube or RSS Feeds with MP4 video files. It will continually monitor these channels and feeds for new content, and send a message to a specified Microsoft Team when new content is available.

The channels and feeds in the tutorial will focus on Power Platform content but can easily be used for other topics.

In this tutorial, you'll use:

- Power Apps to build the application to specify the YouTube channels and RSS feeds.
- Power Automate to retrieve information about new and existing videos.
- Microsoft Dataverse for Teams to create tables to store the feed and video information.
- Power Automate to send a message to Microsoft Teams when a new video is delivered.

Prerequisites

- Identify an existing Microsoft Team or create a new Microsoft Team that you'll build your app for.
- Identify an existing channel or create a new channel within Microsoft Teams to post the messages sent, to specify that a new video has been retrieved.

If you're unfamiliar with how to create a Team or a channel in a Team, see Create a channel in Teams.

In this tutorial, we'll use a channel named "Readiness".

Step 1: Install and pin Power Apps in Teams

In this section, you'll install the Power Apps app in Teams and pin it to the left rail so you can easily access it. To learn how to do install and pin the Power Apps app, see Install the Power Apps personal app in Microsoft Teams.

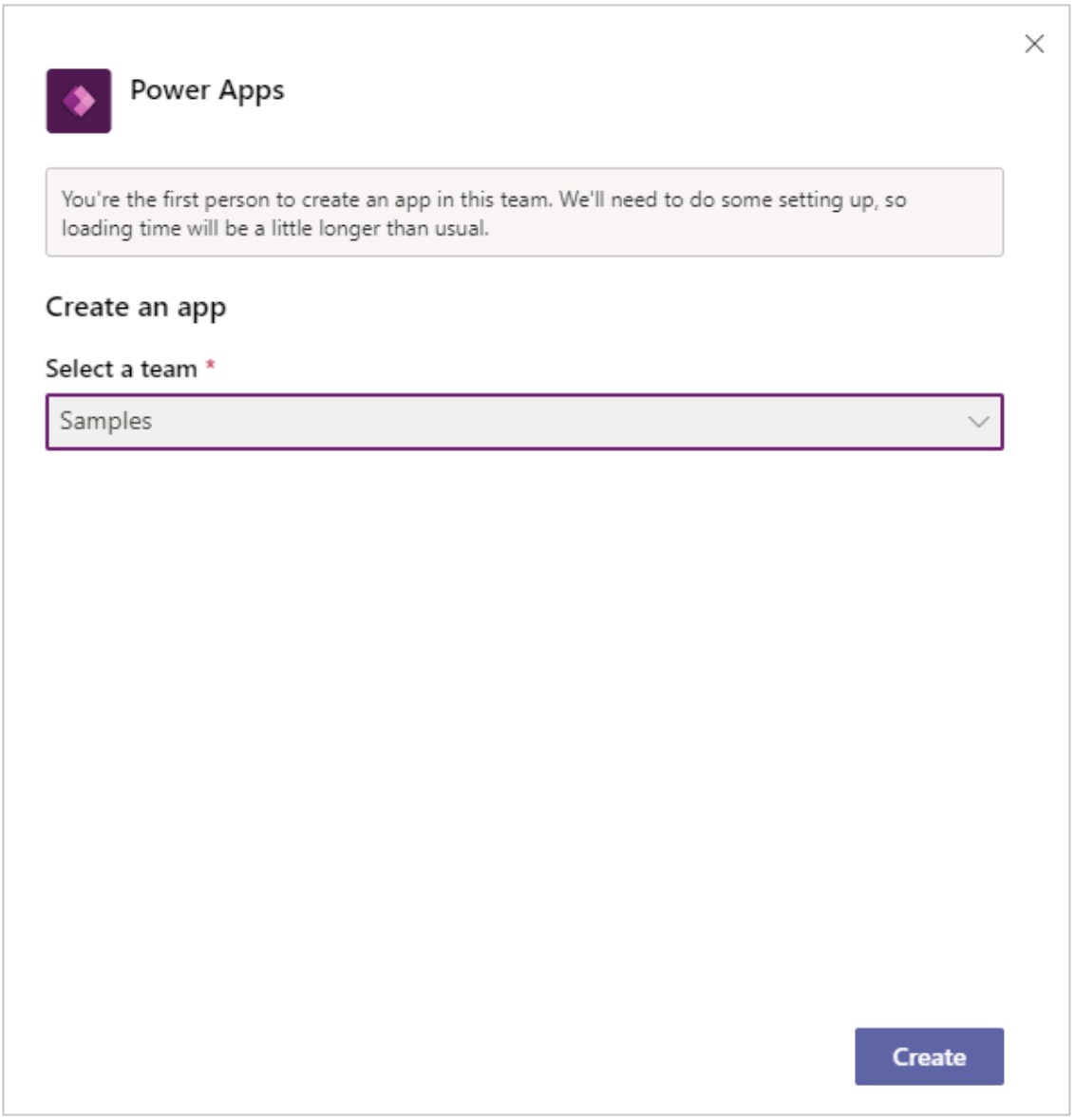
If you've already done installed and pinned the Power Apps app, you can skip to the next step.

Step 2: Create the app and the tables

In this section, you'll take the initial steps to create the application.

1. Select the Power Apps app.
2. Select the **Build** tab at the top of the screen.
3. Select **Create** shown in the lower left of the screen.

   This action will display the **Create an app** dialog box.
4. Select the name of the team from the drop-down list and select **Create**.



> **Note**
>
> If this is the first time you've tried to create an app in this Microsoft Teams team, it will take a moment to set up.

5. When prompted for the name of the app, enter **Video Library** and select **Save**.

Step 2.1: Create the table for the feeds

In this section, you'll create a table to store the names and details for the feeds that will be used by the application.

1. Select **Create new table** on the left side of the screen.

2. When prompted for the name of the table, enter **Feeds**.

3. Select **Create**.

Step 2.1.1: Edit the Name column

1. In the visual editor, select the drop-down menu for the **Name** column and then **Edit column**.
2. Change the name to *Title*.
3. Select **Advanced options** and then change the value for **Max length** to *255*.
4. Select **Save**.

Step 2.1.2: Create the Description column

1. Select + to create a new column.
2. Specify the name as *Description*.
3. For **Type**, select **Text** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *255*.
5. Select **Create**.

Step 2.1.3: Create the Image Link column

1. Select + to create a new column.
2. Specify the name as *Image Link*.
3. For **Type**, select **URL** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *255*.
5. Select **Create**.

Step 2.1.4: Create the Image Title column

1. Select + to create a new column.
2. Specify the name as *Image Title*.
3. For **Type**, select **Text** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *255*.

5. Select **Create**.

Step 2.1.5: Create the Last Retrieved column

1. Select + to create a new column.
2. Specify the name as *Last Retrieved*.
3. For **Type**, select **Text** from the list.
4. Select **Create**.

Step 2.1.6: Create the Link column

1. Select + to create a new column.
2. Specify the name as *Link*.
3. For **Type**, select **URL** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *255*.
5. Select **Create**.

Step 2.1.7: Create the Published On column

1. Select + to create a new column.
2. Specify the name as *Published On*.
3. For **Type**, select **Text** from the list.
4. Select **Create**

Step 2.1.8: Add a row

In the visual editor, add a row for the Microsoft show "#LessCodeMorePower" that will be used in testing.

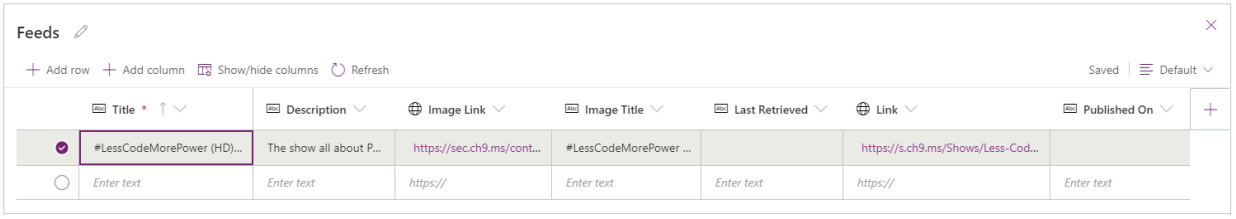
Expand table

Title	**#LessCodeMorePower (HD) - Channel 9**
Description	The show all about Power Apps. Learn more here!

Title	**#LessCodeMorePower (HD) - Channel 9**
Image Link	https://sec.ch9.ms/content/feedimage.png
Image Title	#LessCodeMorePower (HD) - Channel 9
Link	https://s.ch9.ms/Shows/Less-Code-More-Power/feed/mp4high

> **Note**
>
> Leave the **Published On** column empty.

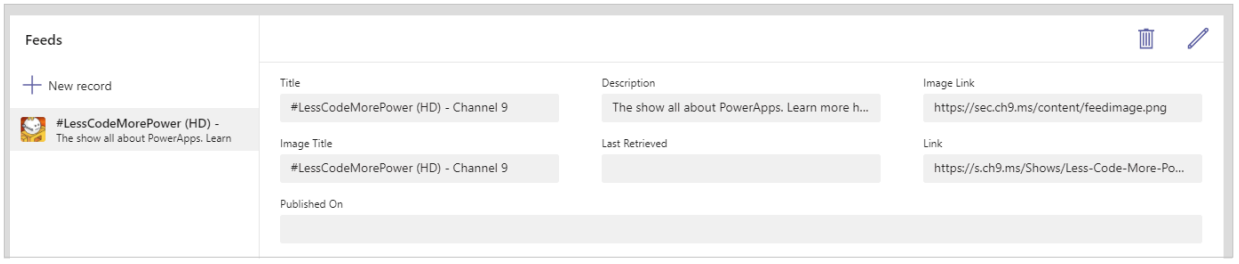
The visual editor should resemble the image below.



Select **Close** to close the visual editor.

The screen created automatically in the app, named **Screen1**, should automatically bind to the table, displaying the fields in the table and the record you added.

The screen should resemble the image below.

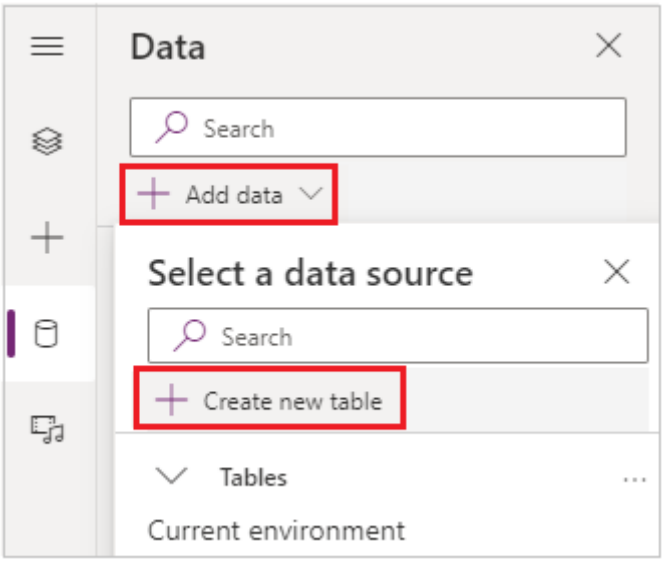


Step 2.2: Create the table for the feed items

In this section, you'll create a table to store the details for the videos stored in the Feed Items for the YouTube channel or RSS feed.

1. Select **Add data** on the left side of the screen.

2. Select **Create new table**.



3. When prompted for the name of the table, enter *Feed Items*.

Step 2.2.1: Edit the Name column

1. In the visual editor, select the drop-down for the **Name** column, and then select **Edit column**.
2. Change the name to *Feed Item ID*.
3. Select **Advanced options** and then change the value for **Max length** to *255*.
4. Select **Save**.

Step 2.2.2: Create the Feed column

1. Select + to create a new column.
2. Specify the name as *Feed*.
3. For **Type**, select **Lookup** from the list.
4. For the table, select **Feeds** from the list
5. Select **Create**.

Step 2.2.3: Create the Title column

1. Select + to create a new column.
2. Specify the name as *Title*.

3. For **Type**, select **Text** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *255*.
5. Select **Create**.

Step 2.2.4: Create the Description column

1. Select + to create a new column.
2. Specify the name as *Description*.
3. For **Type**, select **Text** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *2500*.
5. Select **Create**.

Step 2.2.5: Create the Primary Feed Link column

1. Select + to create a new column.
2. Specify the name as *Primary Feed Link*.
3. For **Type**, select **URL** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *255*.
5. Select **Create**.

Step 2.2.6: Create the Video Link column

1. Select + to create a new column.
2. Specify the name as *Video Link*.
3. For **Type**, select **URL** from the list.
4. Select **Advanced options** and then change the value for **Max length** to *255*.
5. Select **Create**.

Step 2.2.7: Create the Published On column

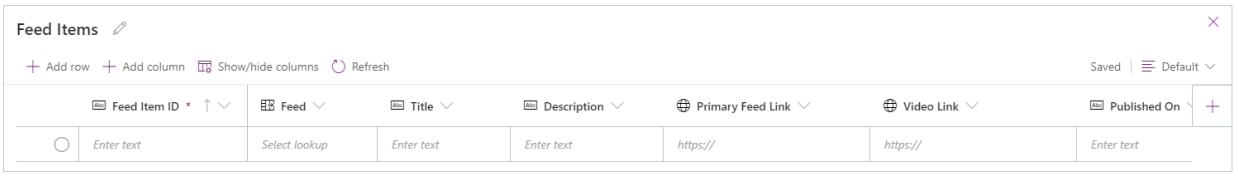
1. Select + to create a new column.

2. Specify the name as *Published On*

3. For **Type**, select **Text** from the list.

4. Select **Create**.

The visual editor should resemble the image below.



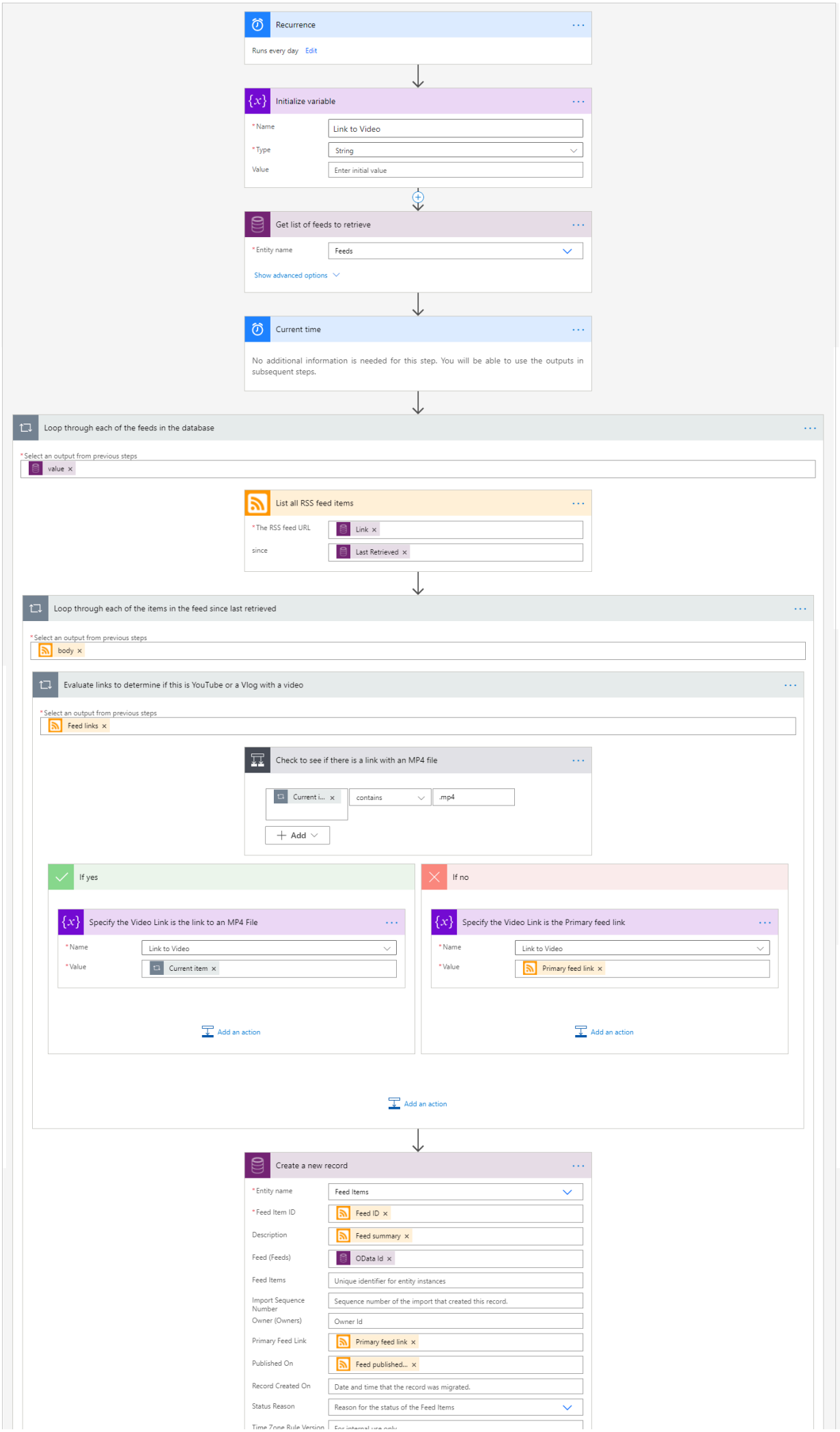
Select **Close** to close the visual editor.

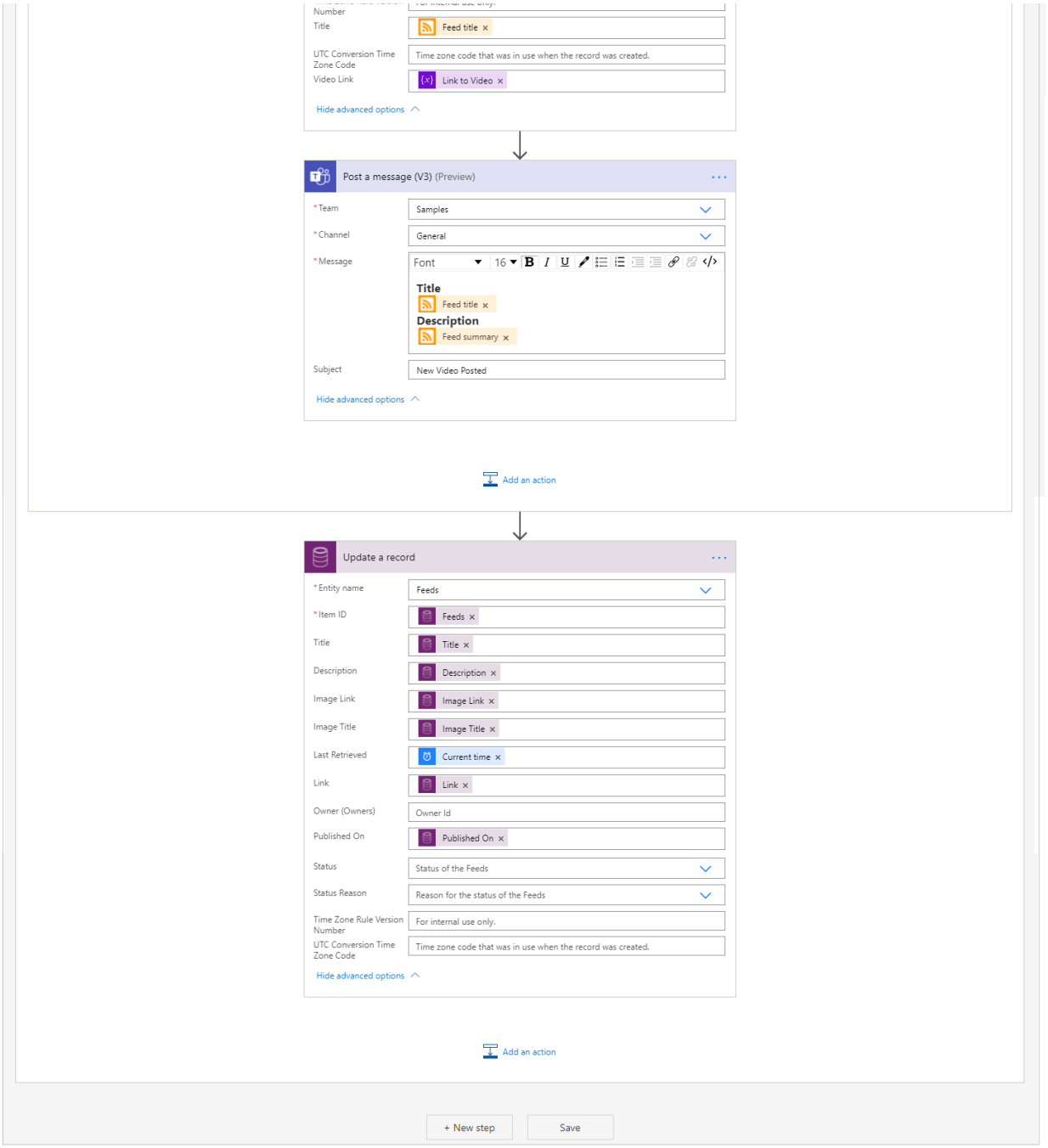
Now that you've created the required tables, select **Save** on the upper-right corner to save the app and close the Power Apps Studio.

Step 3: Create flows to get the video details for configured feeds

In this section, you'll create flows in Power Automate to retrieve videos for each of the configured feeds. This flow will monitor the RSS feeds for the addition of new videos, retrieve details for the videos, and save those details in the database.

When you've completed this section, your flow should resemble the diagram below.


Recurrence
Runs every day Edit
Initialize variable
*Name
Link to Video
*Type
String
Value
Enter initial value
Get list of feeds to retrieve
*Entity name
Feeds
Show advanced options
Current time
No additional information is needed for this step. You will be able to use the outputs in subsequent steps.
Loop through each of the feeds in the database
*Select an output from previous steps
value
List all RSS feed items
*The RSS feed URL
Link
since
Last Retrieved
Loop through each of the items in the feed since last retrieved
*Select an output from previous steps
body
Evaluate links to determine if this is YouTube or a Vlog with a video
*Select an output from previous steps
Feed links
Check to see if there is a link with an MP4 file
Current i...
contains
.mp4
Add
If yes
Specify the Video Link is the link to an MP4 File
*Name
Link to Video
*Value
Current item
Add an action
If no
Specify the Video Link is the Primary feed link
*Name
Link to Video
*Value
Primary feed link
Add an action
Add an action
Create a new record
*Entity name
Feed Items
*Feed Item ID
Feed ID
Description
Feed summary
Feed (Feeds)
OData Id
Feed Items
Unique identifier for entity instances
Import Sequence Number
Sequence number of the import that created this record.
Owner (Owners)
Owner Id
Primary Feed Link
Primary feed link
Published On
Feed published...
Record Created On
Date and time that the record was migrated.
Status Reason
Reason for the status of the Feed Items



> **Note**
>
> Power Automate has a trigger that allows you to detect when a new item is added to a feed for an individual feed. When tracking multiple feeds, this would require multiple feeds and would require the same number of flows to match the number of feeds. The approach taken in this lab is more complex but with the benefit of having a single flow that supports any number of configured feeds in a single flow.

Step 3.1: Create the flow

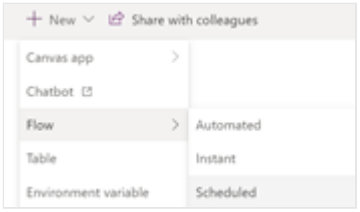
1. Select the **Build** tab.

2. From the list on the left, select the name of your team in which you created the tables earlier.

3. Select **See all** under the *Built by this Team* tab.

4. Select **New**.

5. Select **Flow** > **Scheduled**.

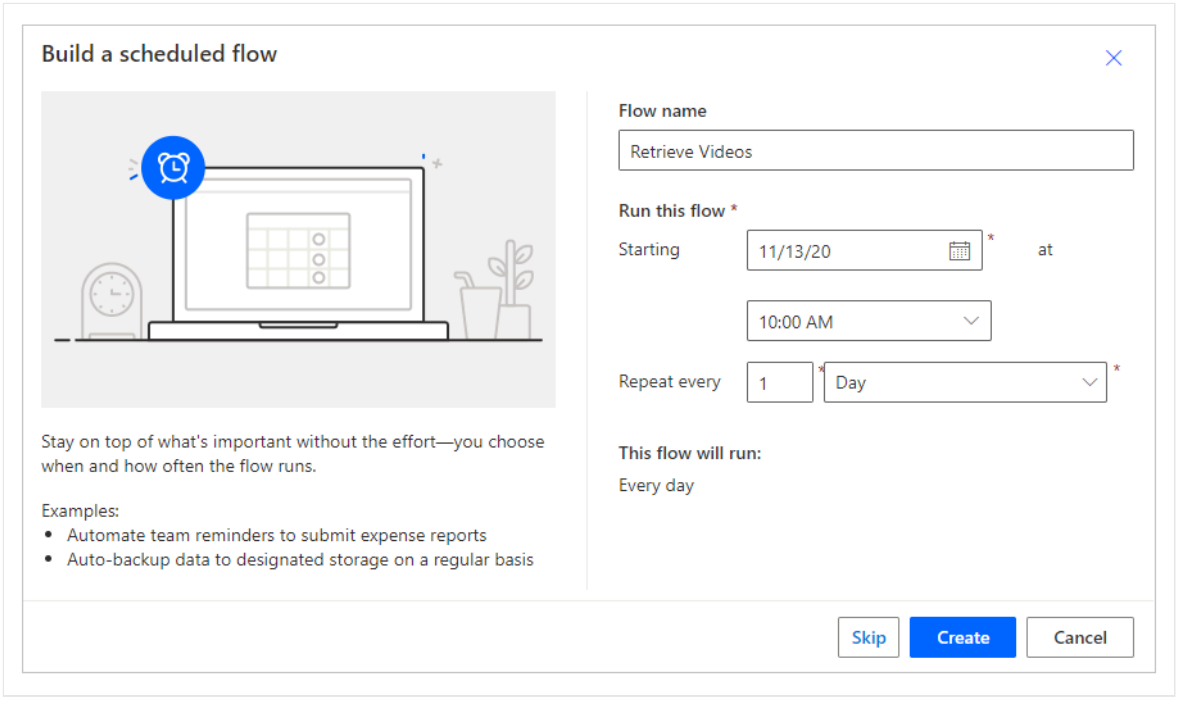


6. In the **Build a scheduled flow** dialog,

   a. Enter the name as "Retrieve Videos".

   b. Select **Day** as the frequency.

   c. Select **Create**.



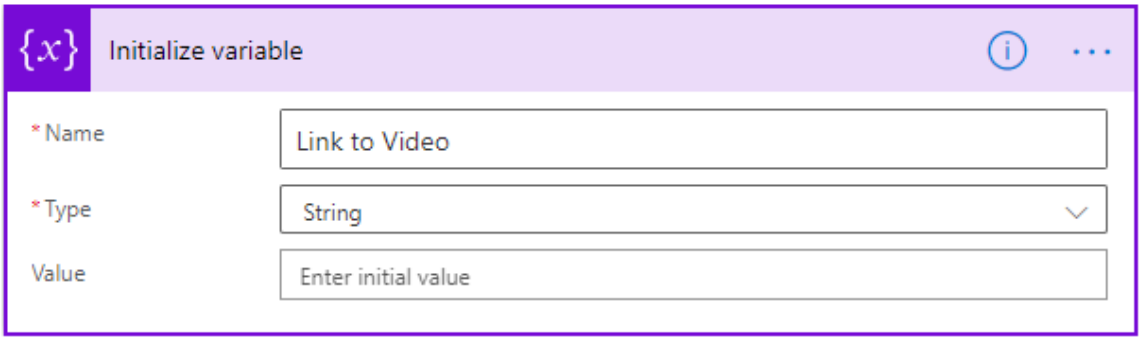
Step 3.2: Add the initialize variable action

1. Select **New step**.

2. Enter *Initialize variable* in the search box.

3. Select the **Initialize variable** action.

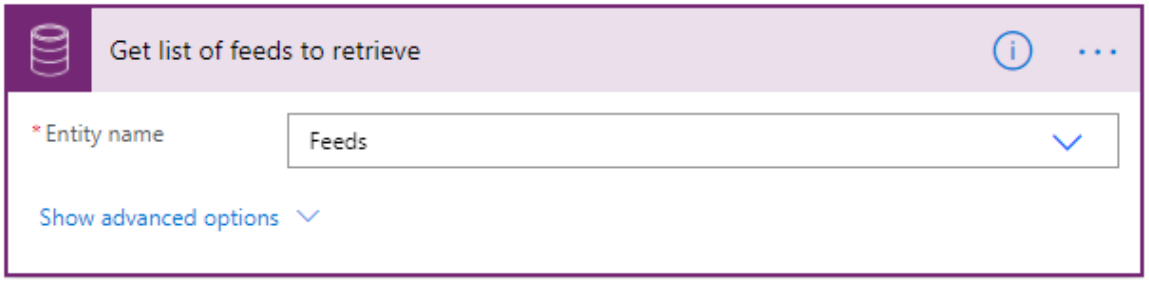
4. Enter *Link to Video* in the **Name** field.
5. Select **String** in the drop-down for **Type**.



> **Note**
>
> Don't enter any value in the **Value** field.

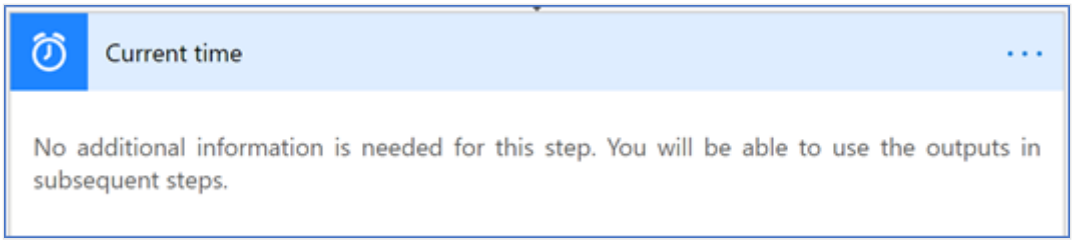
Step 3.3: Add the List Records action

1. Select **New Step**.
2. Enter *Microsoft Dataverse* in the search box and select the action named **List records**.
3. Select **Feeds** in the **Entity name** property.
4. Select ellipsis (...) in the header of the action and select **Rename**.
5. Rename the action to *Get list of feeds to retrieve*.



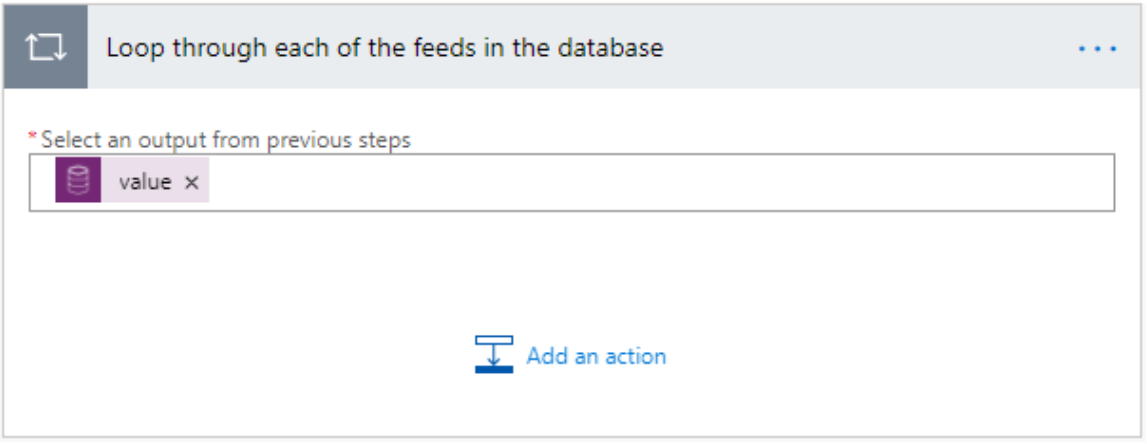
Step 3.4: Add the Current time action

1. Select **New Step**.
2. Enter *Date Time* in the search box and select the action named **Current time**.



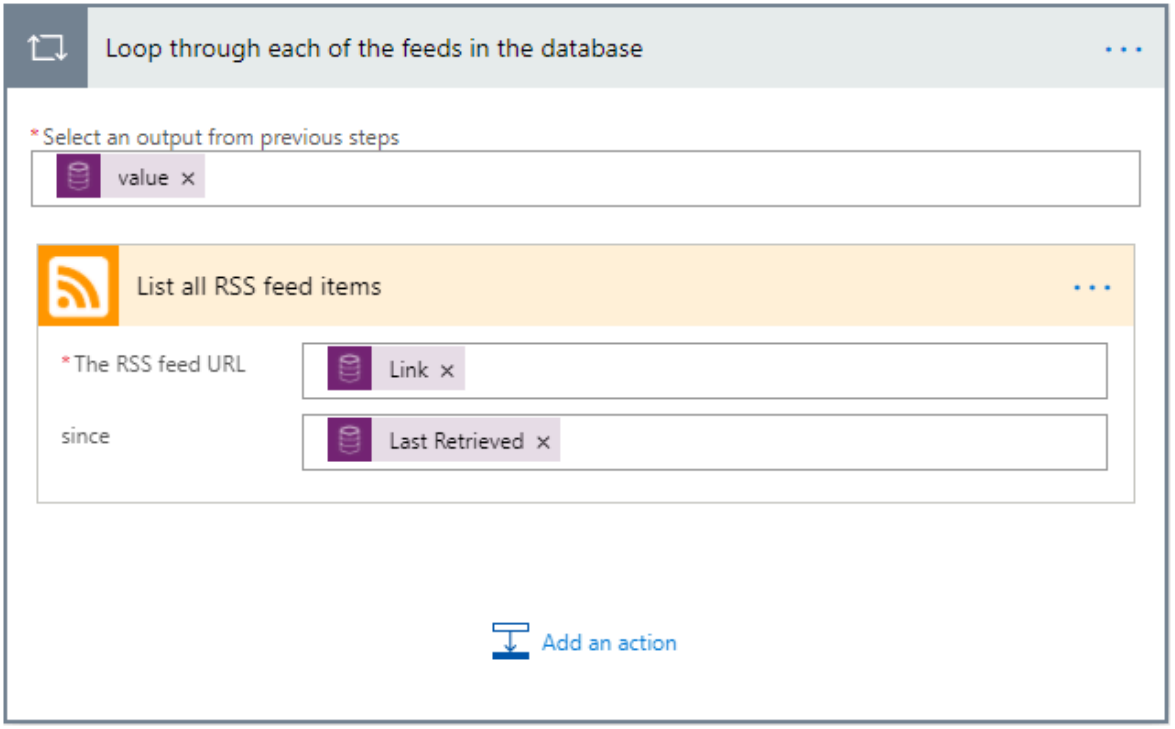
Step 3.5: Add the Apply to each action

1. Select **New Step**.
2. Enter *Apply to each* in the search box, and then select the action named **Apply to each**.
3. Select the text box under **Select output from previous steps**.
4. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **value**.
5. Select ellipsis (...) in the header of the action and select **Rename**.
6. Rename the action to *Loop through each of the feeds in the database*.



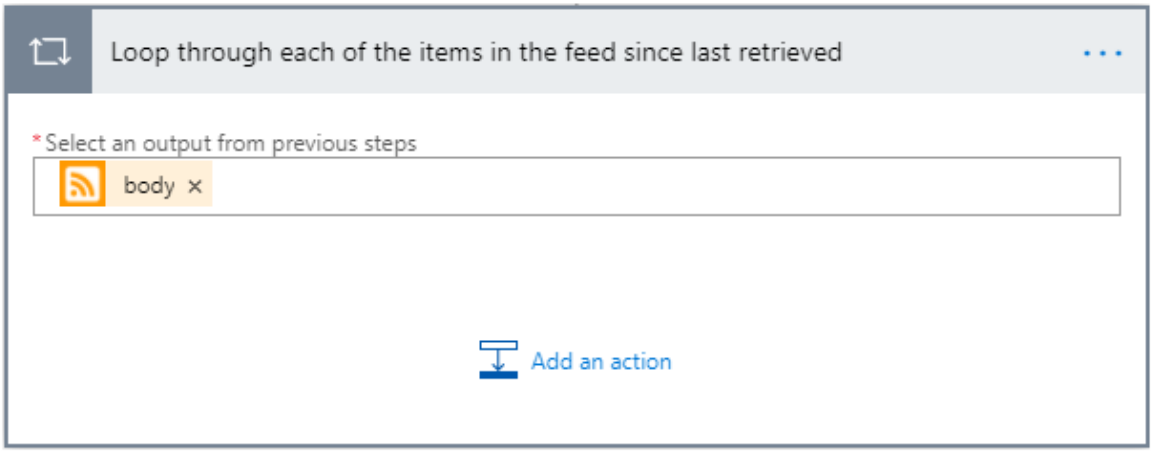
Step 3.6: Add the List all RSS Feed Items action

1. Select **Add an action**.
2. Enter *RSS* in the search box and select the action called **List all RSS feed items**.
3. Select the box for the property **The RSS feed URL**.
4. In the **Dynamic content**, scroll down to the section **Get list of feeds to retrieve**, and then select **Link**.
5. Select in the box for the property **since** and select **Last Retrieved**.



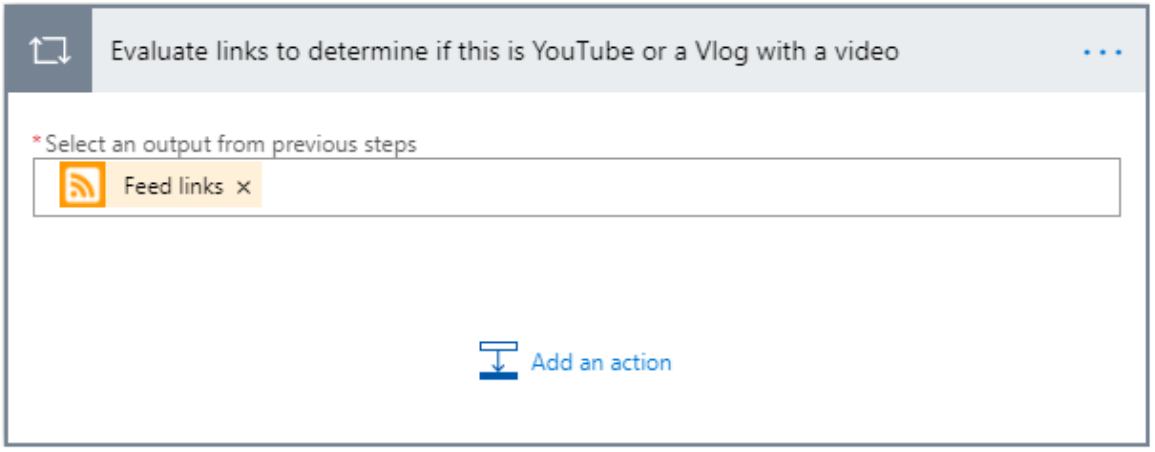
Step 3.7: Add the Apply to Each Action

1. Select **Add an action**.
2. Enter *Apply to each* in the search box and select the action named **Apply to each**.
3. Select the text box under **Select output from previous steps**.
4. In the **Dynamic content** list, scroll down to the section called **List all RSS feed items**, and then select **body**.
5. Select ellipsis (...) in the header of the action, and then select **Rename**.
6. Rename the action to *Loop through each of the items in the feed since last retrieved*.



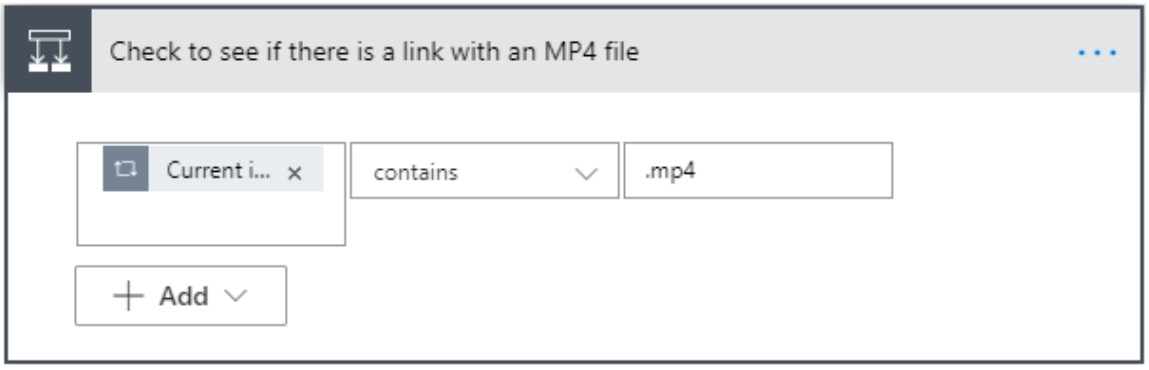
Step 3.8: Add the Apply to Each action

1. Select **Add an action** inside *Loop through each of the items in the feed since last retrieved*.
2. Enter *Apply to each* in the search box and select the action named **Apply to each**.
3. Select the text box under **Select output from previous steps**.
4. In the **Dynamic content** list, scroll down to the section **List all RSS feed items**, and then select **Feed links**.
5. Select ellipsis (...) in the head of the action, and then select **Rename**.
6. Rename the action to *Evaluate links to determine if this is YouTube or a Vlog with a video*.



Step 3.9: Add the Condition action

1. Select **Add an action**.
2. Enter *Condition* in the search box, and then select the action named **Condition**.
3. Select the text box containing text **Choose a value**.
4. In the **Dynamic content** list that is displayed, scroll down to the section called **Evaluate links to determine if this is YouTube or a Vlog with a video**, and then select **Current item**.
5. Select the drop-down that shows **is equal to**, and then select **contains**.
6. Select the text box text **Choose a value** and enter the value **.mp4**.
7. Select ellipsis (...) in the header of the action, and then select **Rename**.
8. Rename the action to *Check to see if there is a link with an MP4 file*.



9. In the **If yes** path on the left, select **Add an action**.

    a. Enter *Set variable* in the search box, and then select the action named **Set Variable**.

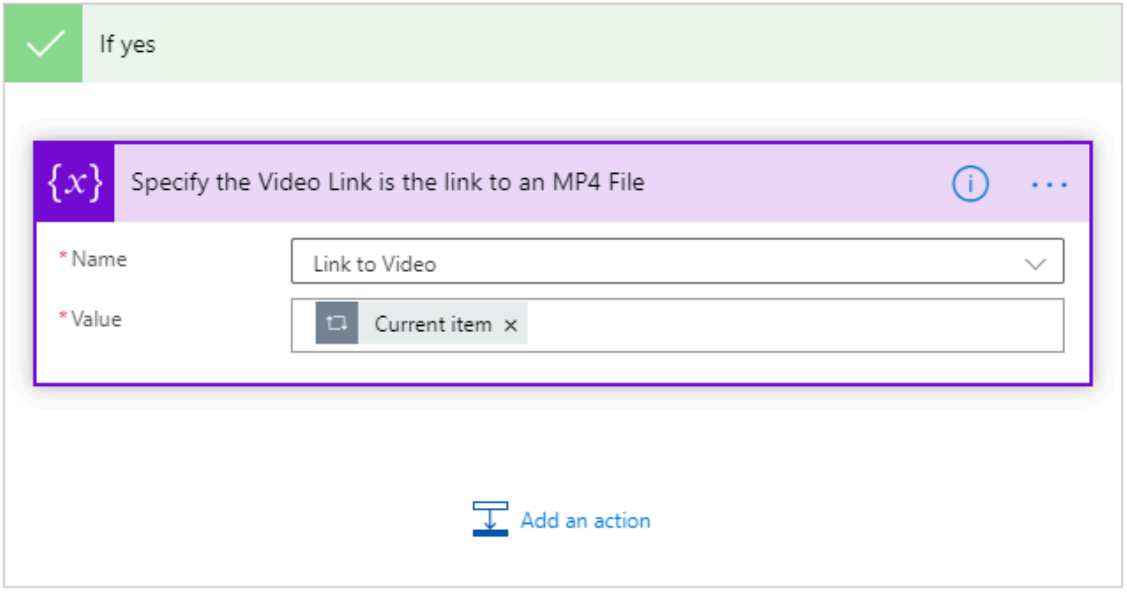
    b. Select **Link to Video** in the drop-down for the *Name* property.

    c. Select the textbox for the **Value** property.

    d. In the **Dynamic content** list, scroll down to the section called **Evaluate links to determine if this is YouTube or a Vlog with a video** and select **Current item**.

    e. Select ellipsis (...) in the header of the action, and then select **Rename**

    f. Rename the action to *Specify the Video Link is the link to an MP4 File*.



10. In the **If No** path on the left, select **Add an action**.

    a. Enter *Set variable* in the search box, and then select the action named **Set Variable**.

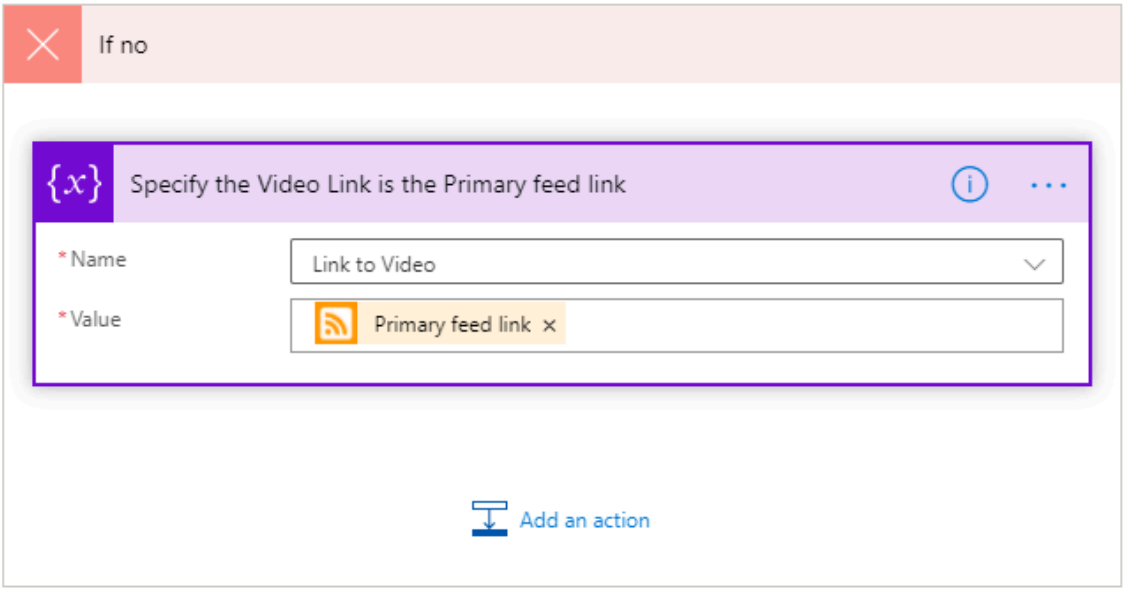
    b. Select **Video Link** in the drop-down for the **Name** property.

c. Select the textbox for the **Value** property.

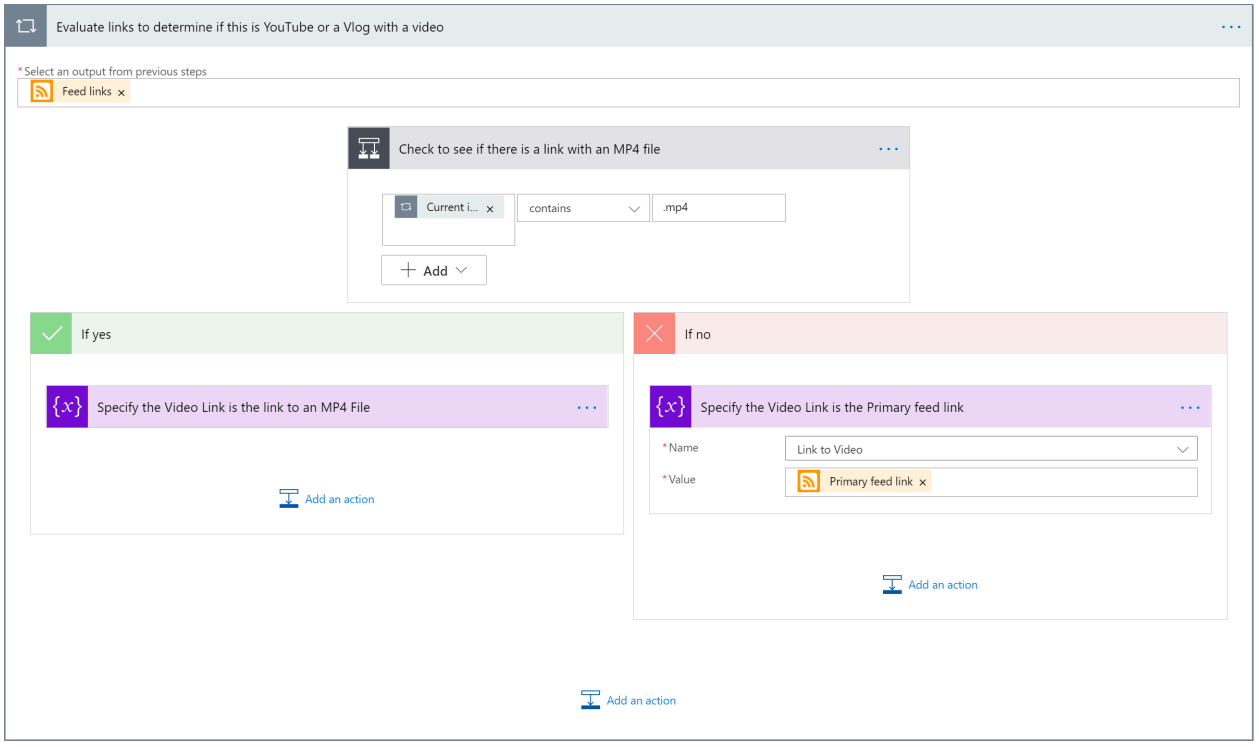
d. In the **Dynamic content** list, scroll down to the section **List all RSS feed items**, and then select **Primary feed link**.

e. Select ellipsis (...) in the header of the action, and then select **Rename**.

f. Rename the action to *Specify the Video Link is the Primary feed link*.

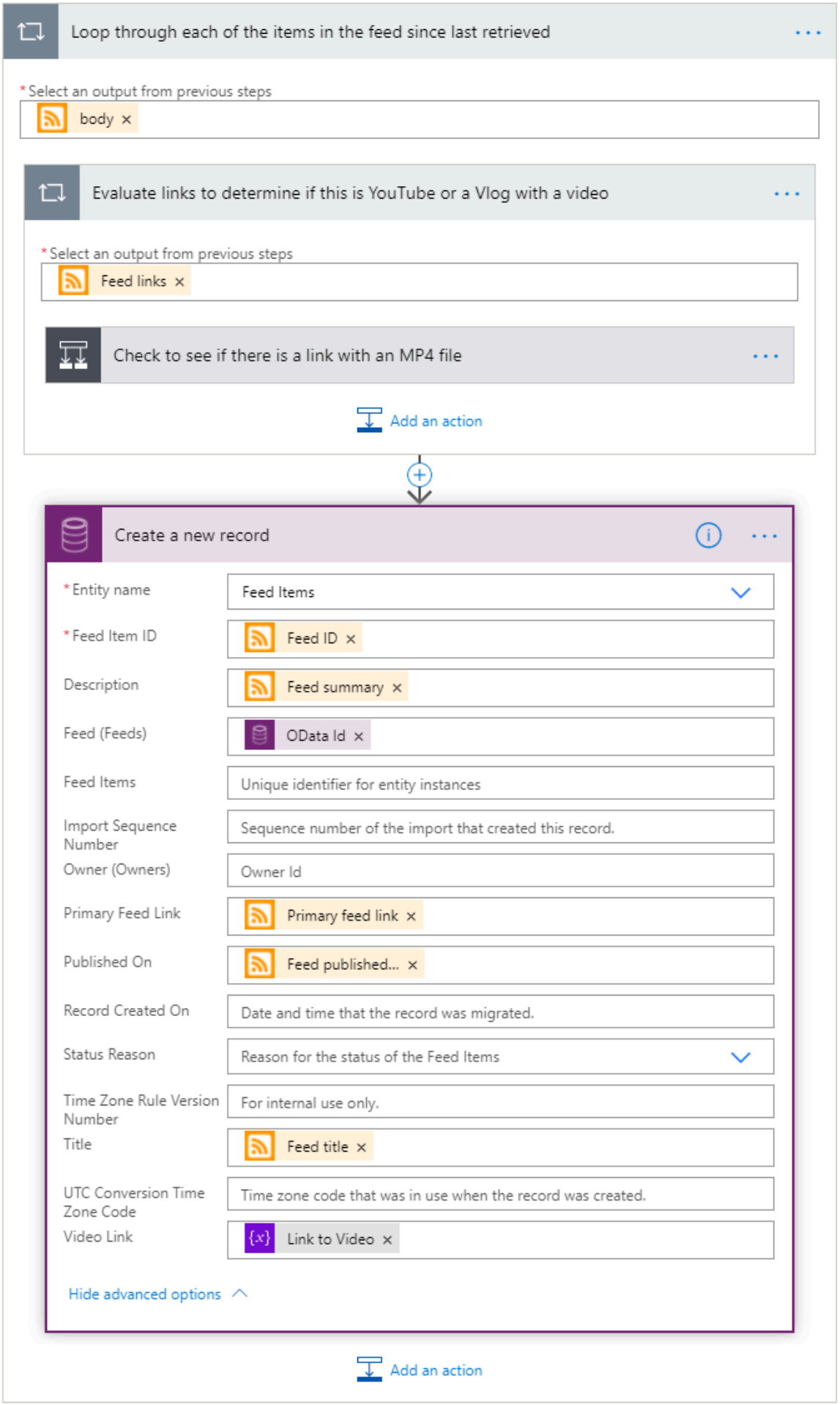


Select the header of the action to show the title of **Evaluate links to determine if this is YouTube or a Vlog with a video**.



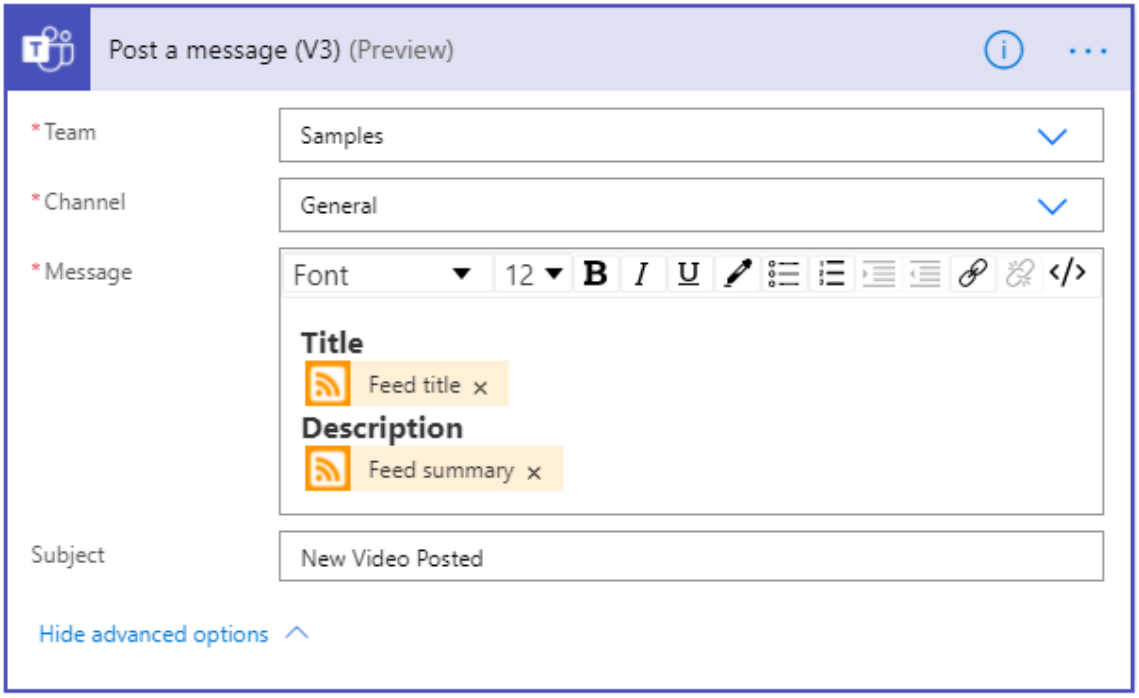
Step 3.10: Add the Create a New Record action

1. Inside *Loop through each of the items in the feed since last retrieved* action, select **Add an action**.
2. Enter *Microsoft Dataverse* in the search box, and then select the action called **Create a new record**.
3. Select **Feed Items** in the *Entity name* property.
4. In the **Dynamic content** list scroll down to the section **List all RSS feed items**, and then select **Feed ID**.
5. Select the textbox next to *Field Item ID*.
6. Select **Show advanced options** link in the action to display all the fields in the table.
7. Select the textbox next to *Description*.
8. In the **Dynamic content** list, scroll down to the section called **List all RSS feed items** and select **Feed summary**.
9. Select the textbox next to *Feed (Feeds)*.
10. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **OData Id**.
11. Select the textbox next to *Primary Feed Link*.
12. In the **Dynamic content** list scroll down to the section **List all RSS feed items**, and then select **Primary feed link**.
13. Select the textbox next to *Published On*.
14. In the **Dynamic content** list, scroll down to the section **List all RSS feed items**, and then select **Feed published on**.
15. Select the textbox next to *Title*.
16. In the **Dynamic content** list, scroll down to the section **List all RSS feed items**, and then select **Feed title**.
17. Select the textbox next to *Video Link*.
18. In the **Dynamic content** list, scroll down to the section **Variables**, and then select **Link to Video**.



Step 3.11: Add the Set Variable action

1. Select **Add action** after *Create new record*.
2. Enter **Teams** in the search box, and then select the **Post a message (V3) (preview)** action.
3. In the drop-down for the **Team** property, select the team to send the message.
4. In the drop-down for the **Channel** property, select the channel to send the message.
5. Select **Show advanced options** link.
6. In the *Message* property, create a message with details about the item. An example message is shown in the image below.

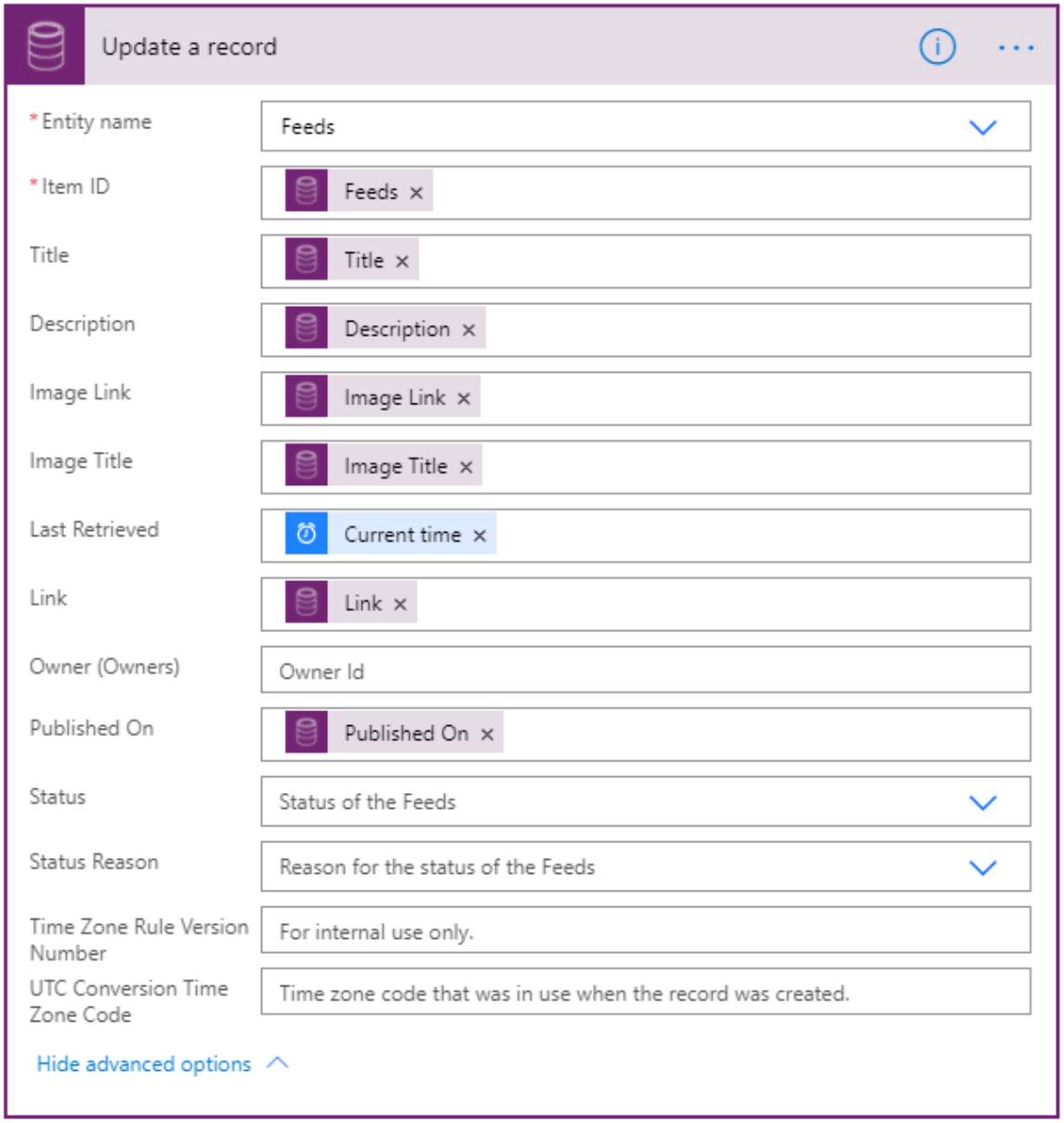


Select the header of the action to show the title of **Loop through each of the items since last retrieved**.

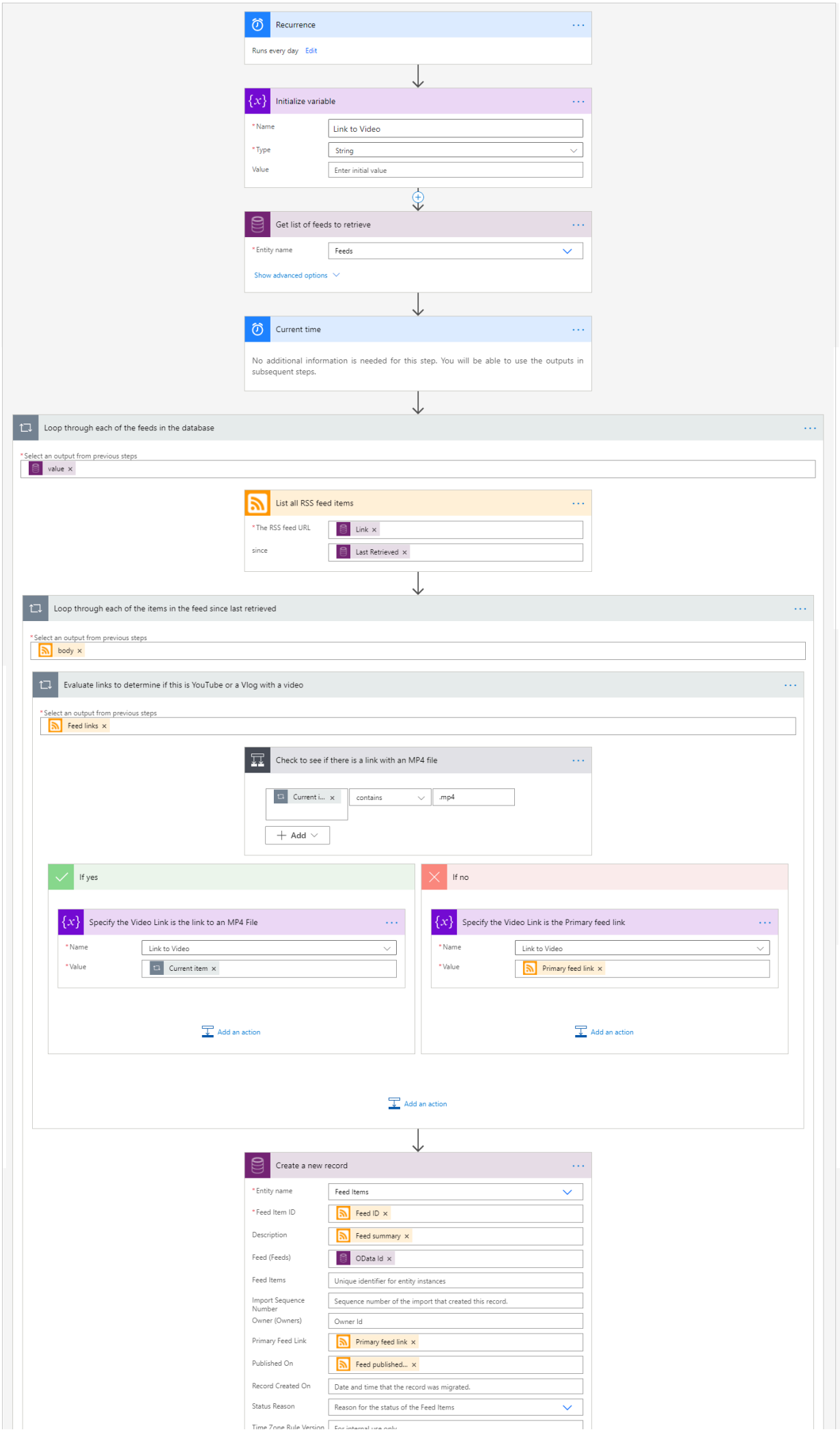
Step 3.12: Add the Update Record action

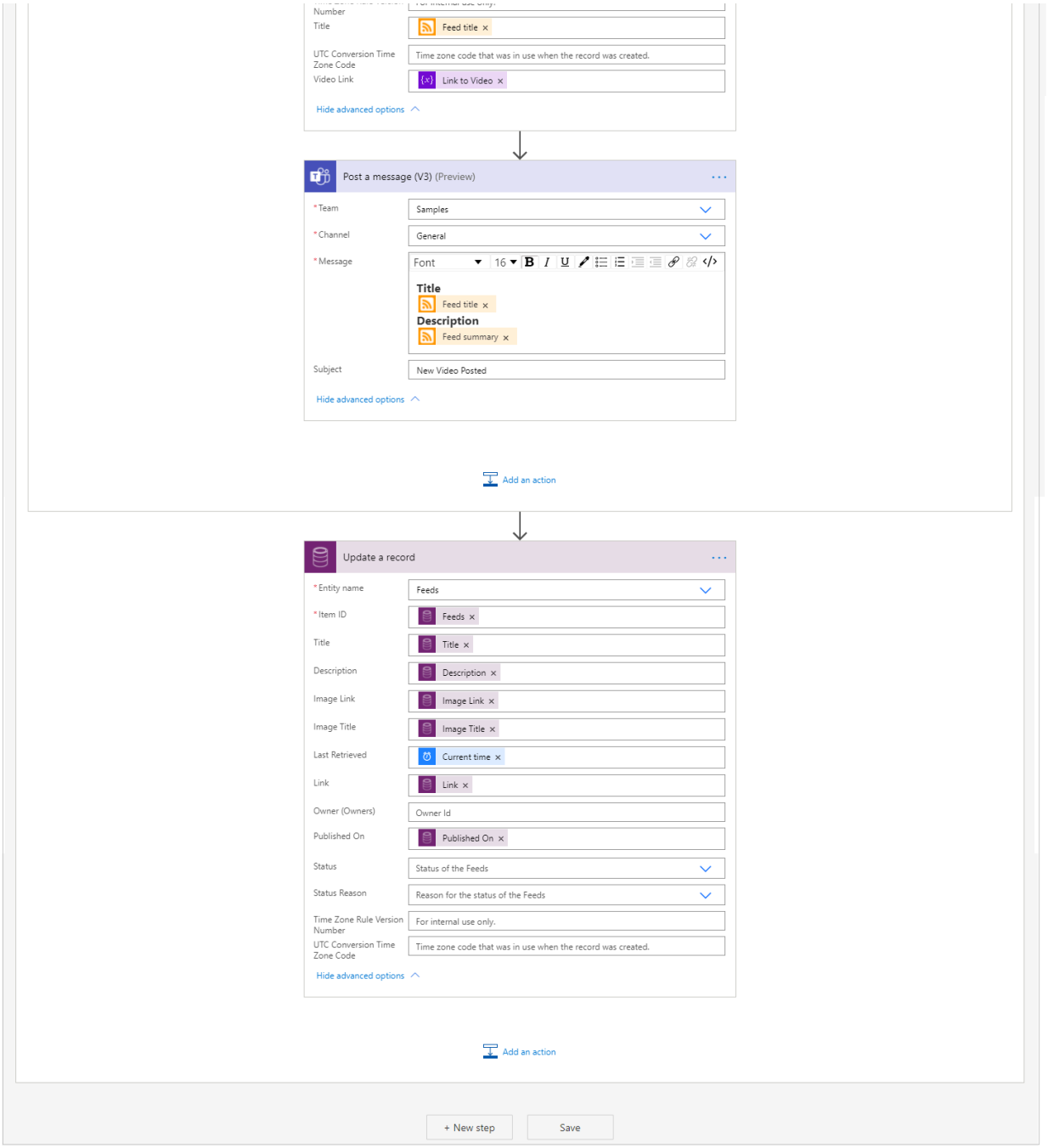
1. Inside *Loop through each of the feeds in the database* action, select **Add an action**.
2. Enter *Microsoft Dataverse* in the search box, and then select the action **Update a record**.
3. Select **Feeds** for the *Entity Name*.
4. Select textbox next to *Item ID*.

5. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **Feeds**.
6. Select textbox next to *Title*.
7. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **Title**.
8. Select **Show advanced options** link in the action to display all the fields in the table.
9. Select textbox next to *Description*.
10. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **Description**.
11. Select textbox next to *Image Link*.
12. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **Image Link**.
13. Select textbox next to *Image Title*.
14. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **Image Title**.
15. Select textbox next to *Last Retrieved*.
16. In the **Dynamic content** list, scroll down to the section **Current time** and select **Current time**.
17. Select textbox next to *Link*.
18. In the **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **Link**
19. Select textbox next to *Published On*.
20. In **Dynamic content** list, scroll down to the section **Get list of feeds to retrieve**, and then select **Published On**.



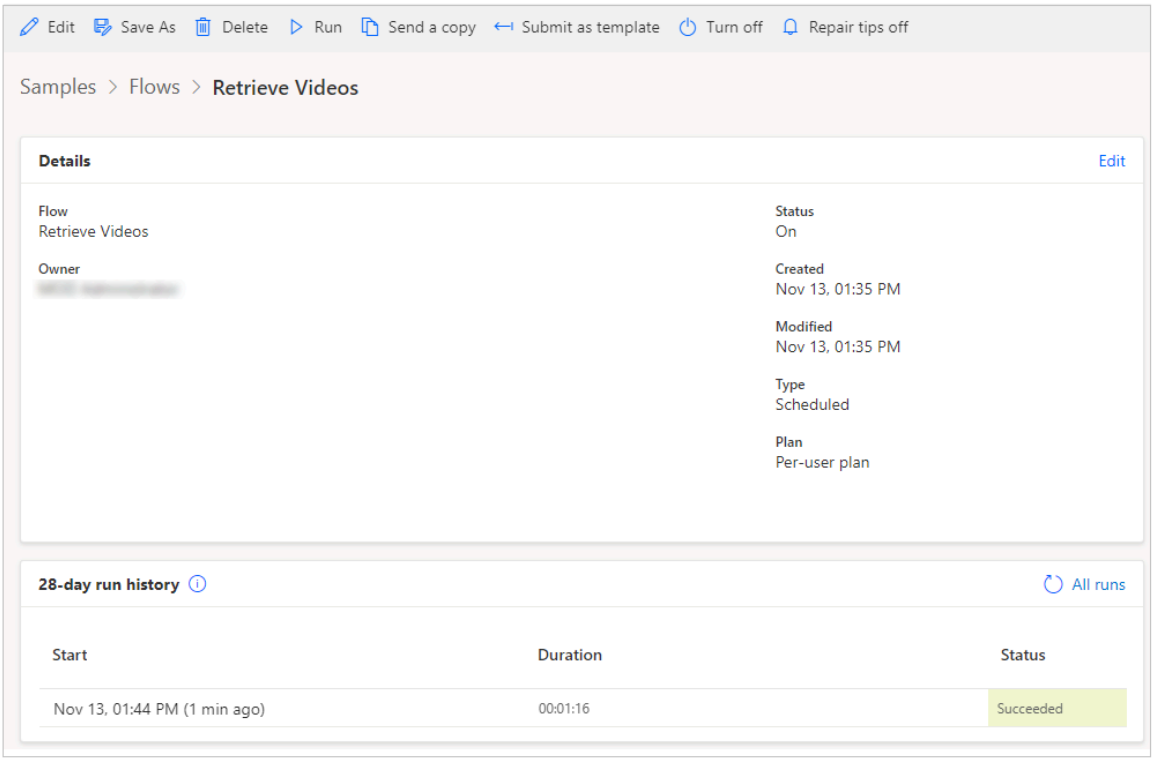
Your complete flow should look like the following image:


Recurrence
Runs every day Edit
Initialize variable
*Name
Link to Video
*Type
String
Value
Enter initial value
Get list of feeds to retrieve
*Entity name
Feeds
Show advanced options
Current time
No additional information is needed for this step. You will be able to use the outputs in subsequent steps.
Loop through each of the feeds in the database
*Select an output from previous steps
value
List all RSS feed items
*The RSS feed URL
Link
since
Last Retrieved
Loop through each of the items in the feed since last retrieved
*Select an output from previous steps
body
Evaluate links to determine if this is YouTube or a Vlog with a video
*Select an output from previous steps
Feed links
Check to see if there is a link with an MP4 file
Current i...
contains
.mp4
Add
If yes
Specify the Video Link is the link to an MP4 File
*Name
Link to Video
*Value
Current item
Add an action
If no
Specify the Video Link is the Primary feed link
*Name
Link to Video
*Value
Primary feed link
Add an action
Add an action
Create a new record
*Entity name
Feed Items
*Feed Item ID
Feed ID
Description
Feed summary
Feed (Feeds)
OData Id
Feed Items
Unique identifier for entity instances
Import Sequence Number
Sequence number of the import that created this record.
Owner (Owners)
Owner Id
Primary Feed Link
Primary feed link
Published On
Feed published...
Record Created On
Date and time that the record was migrated.
Status Reason
Reason for the status of the Feed Items



Step 3.13: Save the Flow

1. From the upper-right side of the screen, select **Save**.
2. Select the back arrow in the upper left.
3. Select the flow **Retrieve Videos**.
4. Select **Run** in the command bar.
5. From the panel on the right side of the screen, select **Run flow**.
6. Select **Done**.
7. Select the refresh button in **28-day run history** to show the details of the flow run.



If the flow doesn't show as succeeded, select **Run** and it will display the failed action with any associated error message.

Step 4: Edit the app to add the video library interface

In this section, you'll create a new screen in the app that enables you and other members of the team to browse through videos, see the video details, and play them directly within Teams.

To start editing the app:

1. Select Power Apps from the left rail in Teams.
2. Select the **Build** tab.
3. From the left pane, select the team environment where the app is created.
4. Under *Built by this team**, select the **Video Library** app.

Now create a new screen and configure the screen controls using the following steps.

Step 4.1: Create new screen

1. Select **Tree view**.

2. Select **New screen**.
3. Select **Blank** layout.
4. Select + to begin adding controls to screen.

Step 4.2: Add the header

1. Enter **Rectangle** in the search box.
2. Drag the rectangle control onto the screen.
3. Set the following properties with these values:

Expand table

Property	Value
X	0
Y	0
Width	1365
Height	60
Color	Purple

Step 4.3: Add the header label

1. Enter **Label** in the search box.
2. Drag the label onto the screen.
3. Set the following properties with these values:

Expand table

Property	Value
Text	Video Library
Display Mode	View
Font size	24
X	576

Property	Value
Y	0
Width	191
Height	61
Color	White

Step 4.4: Add a rectangle for the Video List Header

1. Enter **Rectangle** in the search box.
2. Drag the rectangle control onto the screen.
3. Set the following properties with these values:

Expand table

Property	Value
X	0
Y	61
Width	1365
Height	58
Color	Purple

Step 4.5: Add a label for the Video List Header

1. Enter **Label** in the search box.
2. Drag the label onto the screen.
3. Set the following properties with these values:

Expand table

Property	Value
Text	Select a video from the list below
Display Mode	View

Property	Value
Font size	14
X	39
Y	74
Width	320
Height	32
Color	White

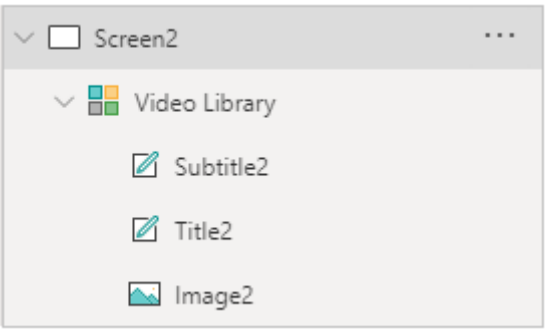
Step 4.6: Add a Vertical Gallery

1. Enter **Vertical Gallery** in the search box.
2. Drag the label onto the screen.
3. Set the following properties with these values:

Expand table

Property	Value
Name	Video Library
Data Source	Feed Items
Layout	Image, title, and subtitle
X	0
Y	119
Width	450
Height	649

In the tree view on the left side of the screen, there are three controls underneath the Video Gallery with names starting with Subtitle, Title, and Image.



Step 4.6.1: Set the Subtitle

1. Select the control that starts with **Subtitle**.
2. Select the **Text** property from the property list on top-left.
3. Update the property value to the following formula:

```
Power Fx

ThisItem.Feed.Description
```

Step 4.6.2: Set the Title

1. Select the control that starts with **Title**.
2. Select the **Text** property from the property list on top-left.
3. Update the property value to the following formula:

```
Power Fx

ThisItem.Feed.Title
```

Step 4.6.3: Set the Image

1. Select the control that starts with **Image**.
2. Select the **Text** property from the property list on top-left.
3. Update the property value to the following formula:

```
Power Fx
```

```
ThisItem.Feed.'Image Link'
```

Step 4.7: Add the background for the feed item

1. Enter **Rectangle** in the search box.
2. Drag the rectangle control onto the screen.
3. Set the following properties with these values:

Expand table

Property	Value
X	451
Y	61
Width	915
Height	707
Color	Purple

Step 4.8: Add the feed item title header label

1. Enter **Label** in the search box.
2. Drag the label onto the screen.
3. Set the following properties with these values:

Expand table

Property	Value
Font size	18
X	466
Y	51
Width	883
Height	78

Property	Value
Color	White

4. Select the **Text** property from the property list on top-left.

5. Set the **Text** property value to the following formula:

```
Power Fx
'Video Library'.Selected.Title
```

Step 4.9: Add a Video Control

1. Enter **Video** in the search box.

2. Drag the video control onto the screen.

3. Set the following properties with these values:

Expand table

Property	Value
X	450
Y	129
Width	916
Height	523

4. Select the **Media** property from the property list on top-left.

5. Set the **Media** property value to the following formula:

```
Power Fx
'Video Library'.Selected.'Video Link'
```

Step 4.10: Add the feed description HTML Text control

1. Enter **HTML text** in the search box.

2. Drag the HTML text control onto the screen.

3. Set the following properties with these values:

Expand table

Property	Value
Font size	14
X	458
Y	651
Width	908
Height	82
Color	White

4. Select the **HtmlText** property from the property list on top-left.

5. Set the **HtmlText** property value to the following formula:

```
Power Fx
'Video Library'.Selected.Description
```

Step 4.11: Add the Published On label

1. Enter **Label** in the search box.

2. Drag the Label onto the screen.

3. Set the following properties with these values:

Expand table

Property	Value
Font size	14
X	1153
Y	732
Width	213
Height	36

Property	Value
Color	White

4. Select the **Text** property from the property list on top-left.

5. Set the **Text** property value to the following formula:

```
Power Fx

Concatenate("Published On ",Text('Video Library'.Selected.'Published
On'))
```

Step 4.12: Enable toggle between screens

Currently there are two separate screens, but no way to toggle between them. The ability to toggle between the screens will be important to enable the addition of more feeds.

Step 4.12.1: Add the gear icon

1. Select **Screen2**.

2. Select **Add icon** from the properties pane.

3. Select the **Settings** icon type.

4. Set the following properties with these values:

Expand table

Property	Value
X	1307
Y	6
Width	48
Height	51
Color	White

5. Select the **OnSelect** property from the property list on top-left.

6. Set the **OnSelect** property value to the following formula:

```
Power Fx

Navigate(Screen1)
```

Step 4.12.2: Configure the back button

1. Select **Screen1**.
2. Select **Add icon** from the properties pane.
3. Select the **Back** icon type.
4. Set the following properties with these values:

Expand table

Property	Value
X	225
Y	12
Width	32
Height	32

5. Select the **OnSelect** property from the property list on top-left.
6. Set the **OnSelect** property value to the following formula:

```
Power Fx

Navigate(Screen2)
```

When you select the gear icon, the app will now show the form to edit the list of feeds. Selecting the back button will return to the list of videos.

Step 5: Test the app

Before publishing the application to teams, it's important to test core functionality.

Step 5.1: Test the Power Automate Flow

1. Run the Power Automate flow.

2. Confirm that new items have been added to the **Feed Items** table.
3. Confirm that the flow has posted new messages to the Channel you specified when configuring the application.

Step 5.2: Test the app functionality

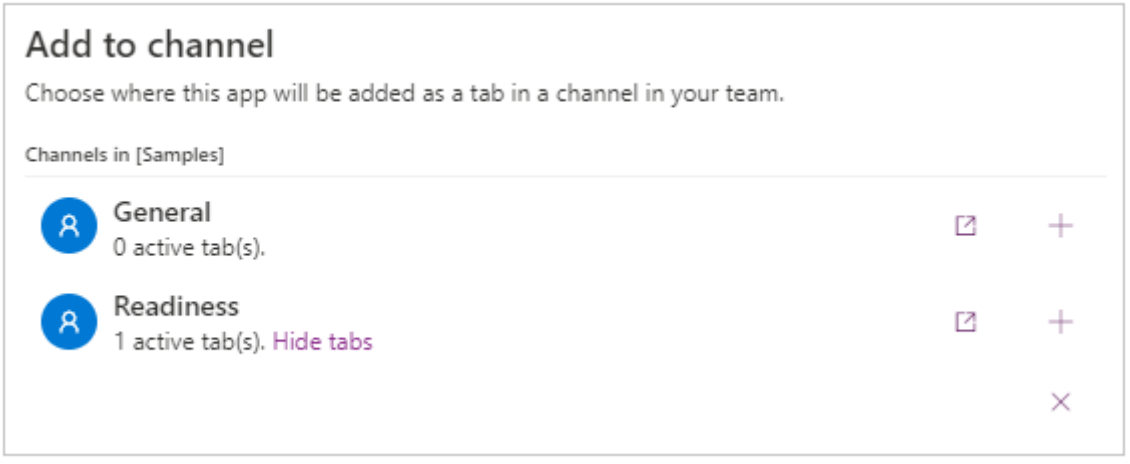
1. Open the app in Power Apps Studio.
2. From upper-right side of the screen, select **Preview**.
3. Select the list of feed items in the gallery on the left.
4. When you select an item, confirm that the *Title*, *Description*, *Publish Date*, and the Video details are displayed.
5. Select the video and confirm the video plays.

Step 5.3: Test additions to the Feeds table

1. Add another feed to the **Feeds** table.
2. Rerun the Power Automate flow.
3. Repeat the above steps.

Step 6: Publish the app

1. Select the **Publish to Teams** icon in the upper right.
2. Select **Next**.
3. Select + next to the channel, such as **Readiness** to add the app to.



4. Select **Save and close**.

See also

Use sample app templates in Microsoft Teams

Feedback

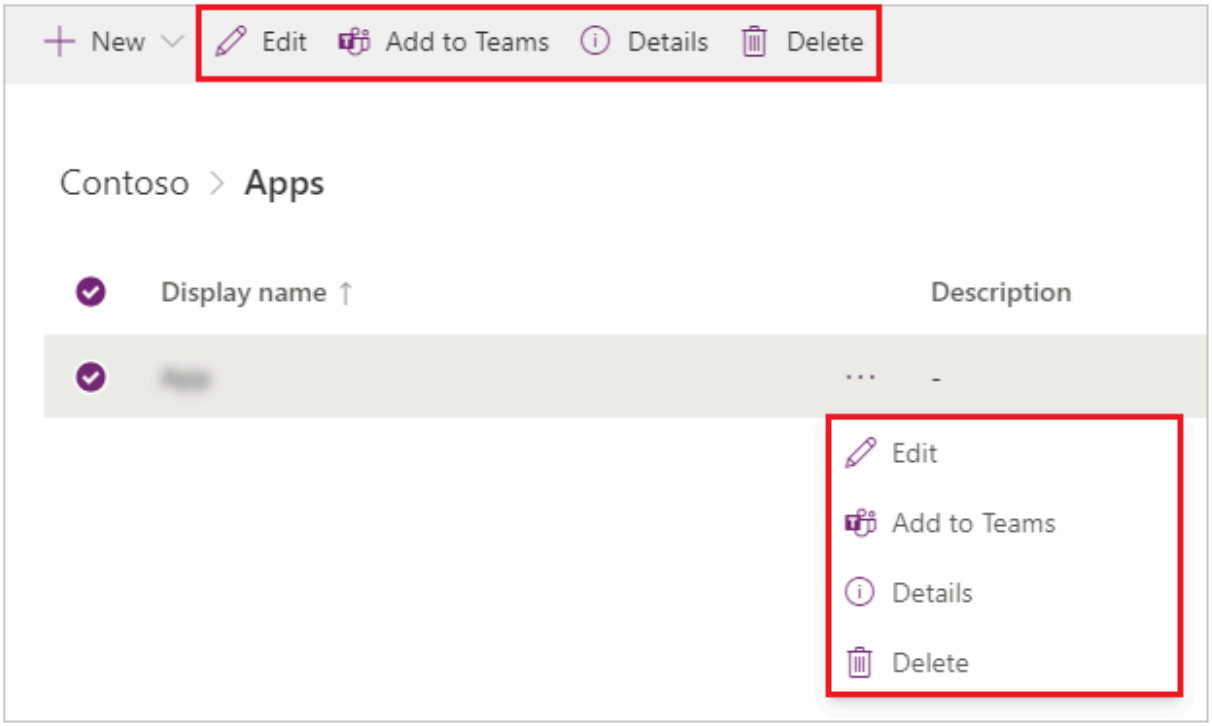
Was this page helpful? Yes No

Provide product feedback

Manage your apps in Teams

Article • 12/16/2022

To manage an app, select **Apps** from the solution explorer. Then select the app that you want to manage, or select **Commands** (...) next to the app name, and then select commands from the drop-down menu.



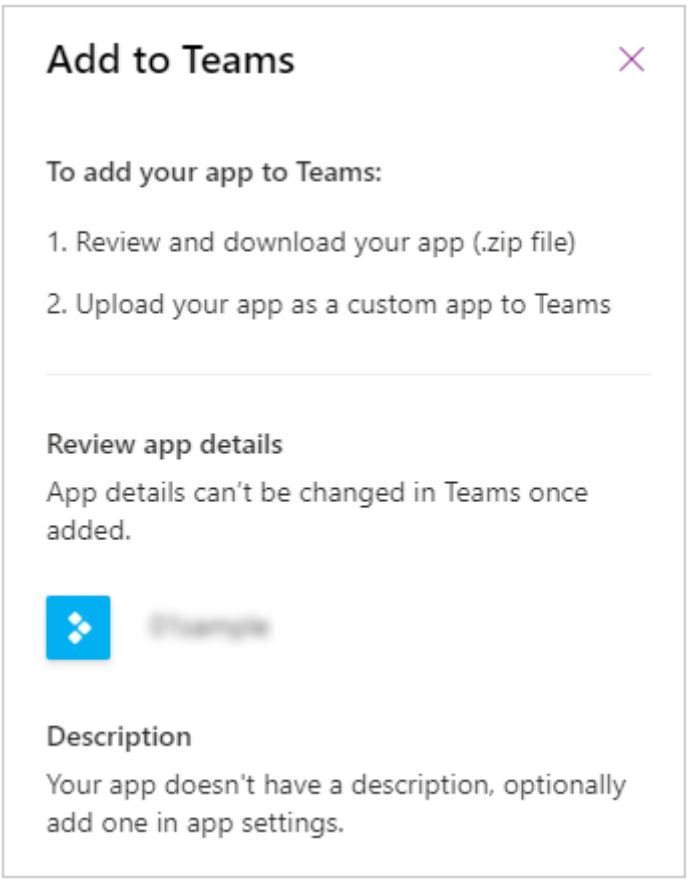
Edit an app

Select **Edit** to edit the app in Power Apps Studio. More information: Edit a canvas app in Power Apps and Understanding Power Apps Studio

> **Tip**
>
> Use this option if you want to rename an app, or change app icon and description. For detailed steps, go to rename an app or change app icon and description.

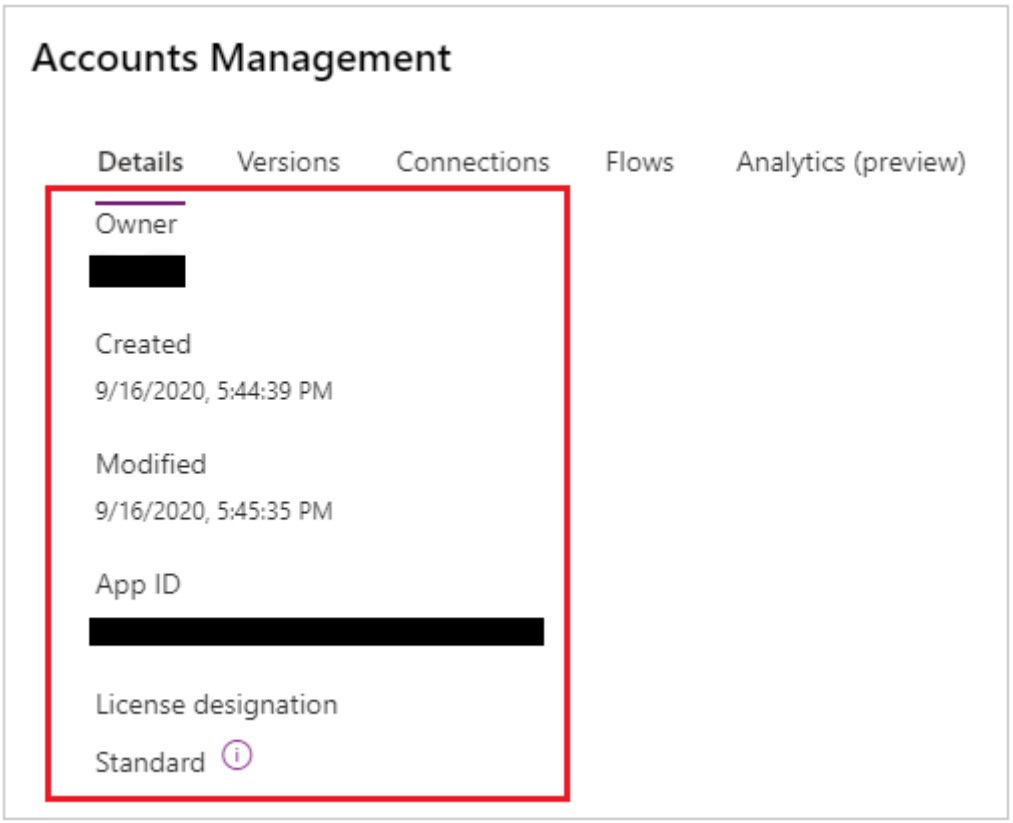
Add to Teams

Allows you to download the app, and then upload to Teams. To learn how to add an app to Teams, go to Embed a canvas app in Teams.



App details

The **Details** option shows the owner of the app, when the app was created and last modified, app ID and license designation.

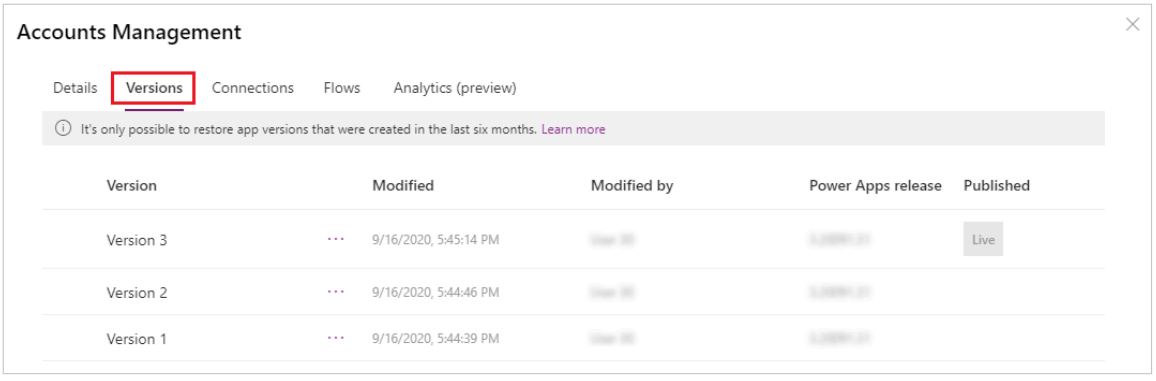


Restore an app

A canvas app can have multiple versions. You can restore an app to an available published version.

To restore an app to a specific version:

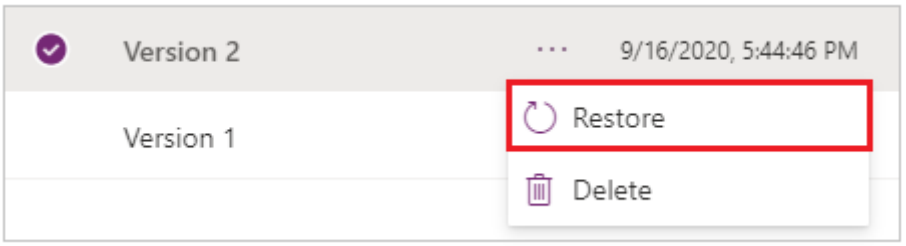
1. Select **Versions** from the app details screen.



2. Select a version other than *Live* that you want to restore.

3. Select ···.

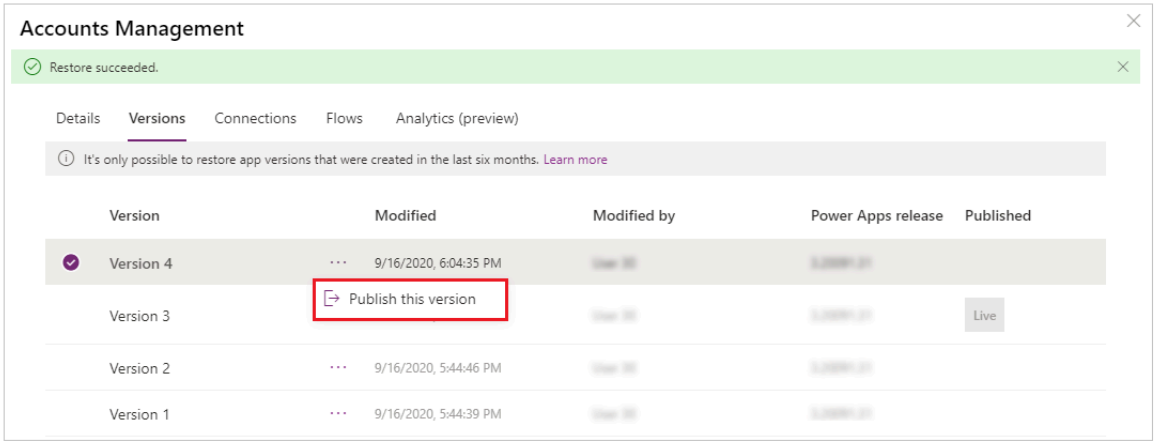
4. Select **Restore**.



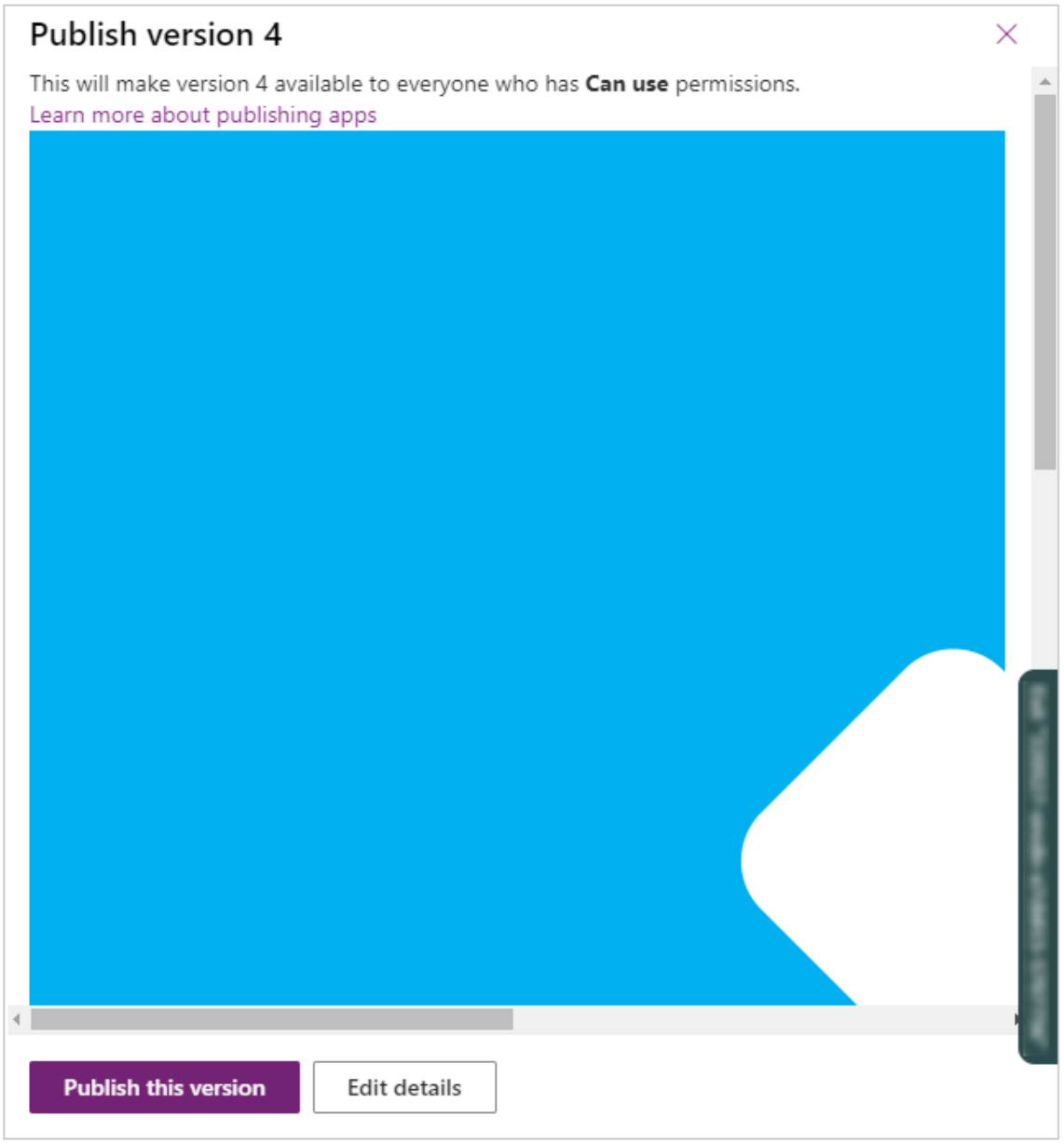
> **Note**
>
> The restored app becomes the latest version of the app. However, the latest version is not yet live. You must publish the restored version to make the restored version available to all users.

5. Select ··· for the restored version.

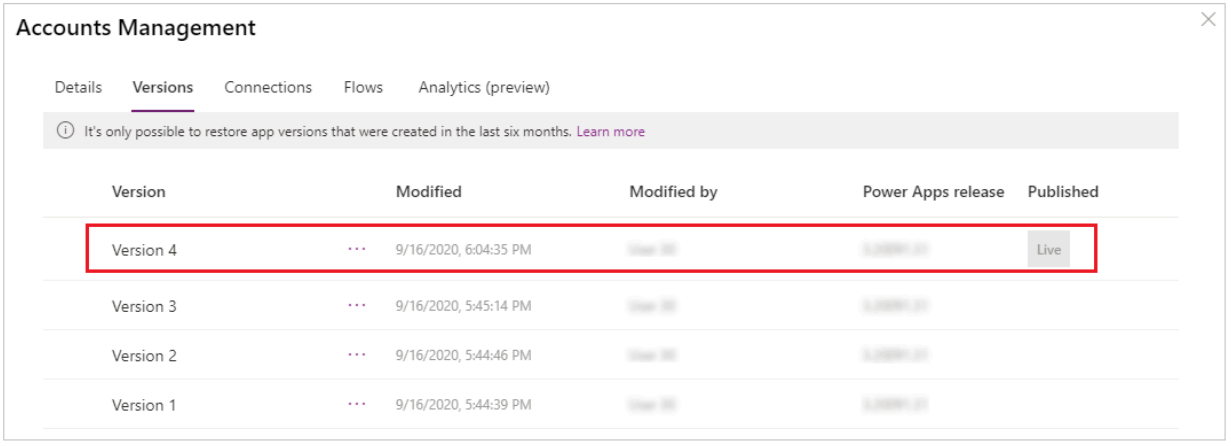
6. Select **Publish this version**.



7. Review the app preview and select **Publish this version** to confirm.



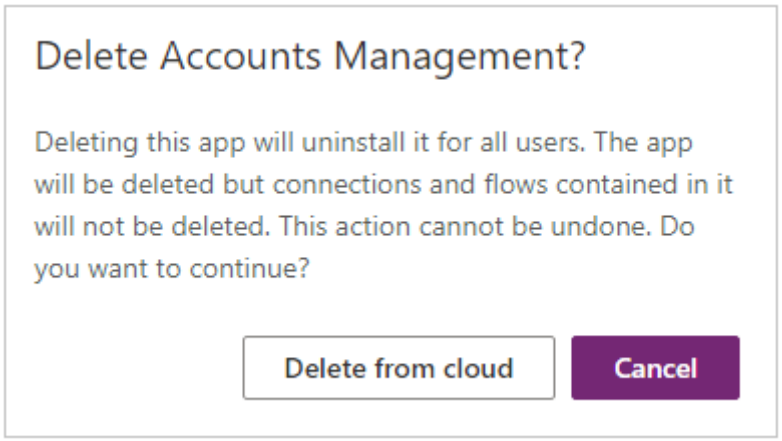
The **Versions** tab now shows the restored version as *Live*:



More information: Restore an app

Delete an app

Select **Delete** to delete an app. When prompted, select **Delete from cloud** to delete the app.

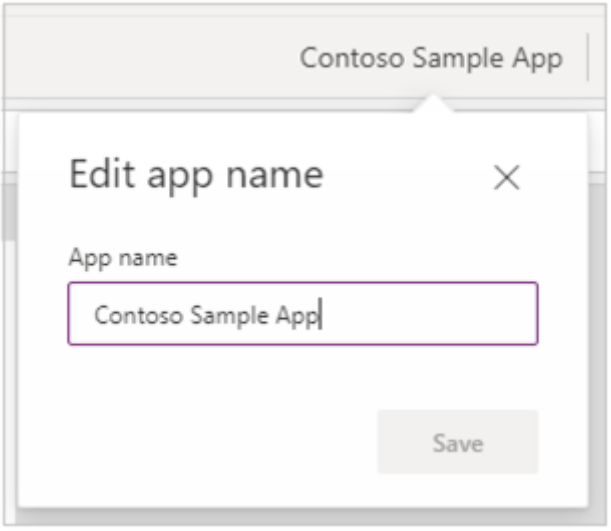


For more information about deleting an app, go to Delete an app.

Rename an app

To rename an app:

1. Edit the app in Power Apps Studio.
2. Select the app name from the top-right.
3. Update the name.



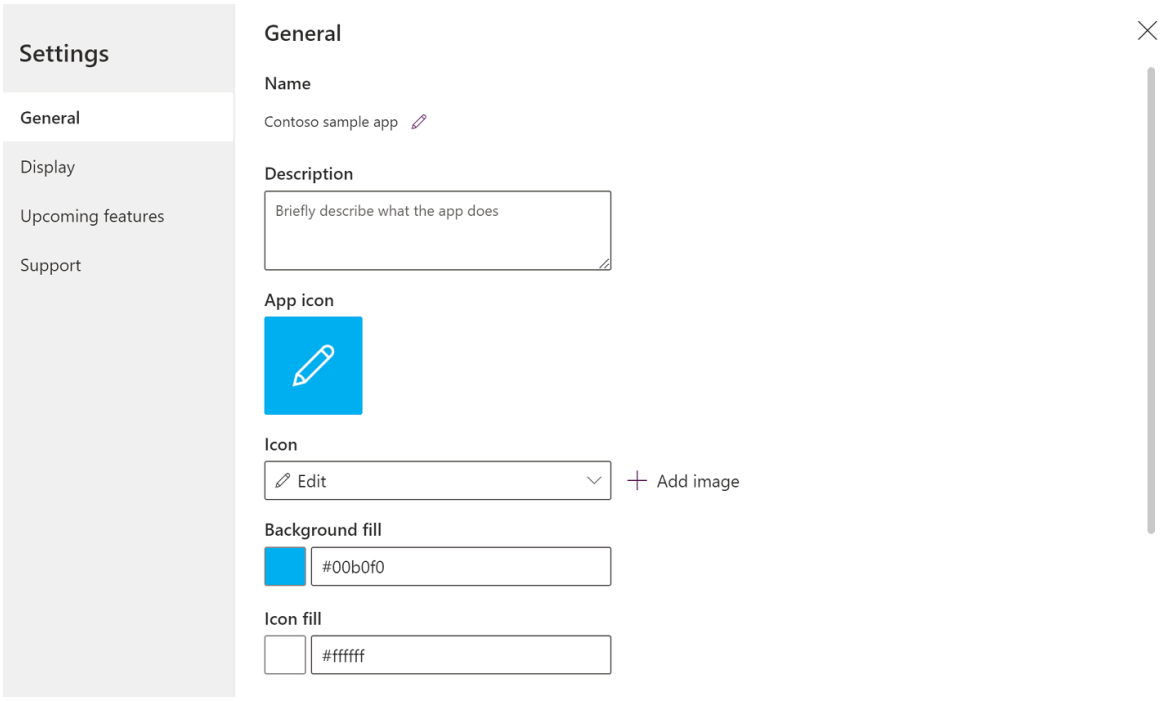
4. Select **Save**.

More information: Edit app editor

Change app icon and description

To change the app icon and description:

1. Edit the app in Power Apps Studio.
2. Select **Settings**.
3. Update the icon and description.



4. Select **Save**.

More information: Change app settings

See also

Publish and share your apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Set permission and share apps with colleagues

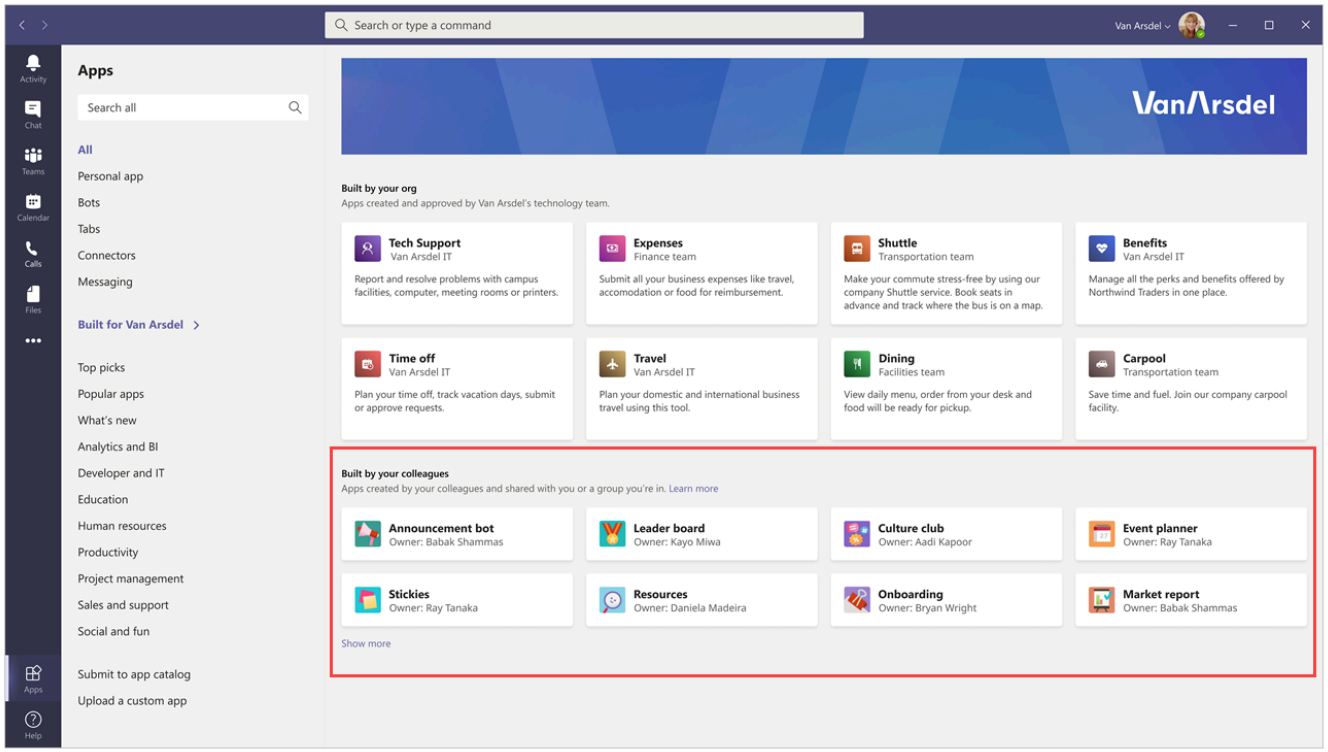
Article • 05/12/2025

After collaboration with your team to build an app, as an owner of the team, you can share the app and its underlying data with other colleagues within your company that don't belong to your development team.

To share an app:

1. Assign the correct permissions for the tables in your app for the **Colleagues with access** role.
2. Select a security group you would like to assign to the **Colleagues with access** role.
3. Choose which apps the **Colleagues with access** role should have access to.

Once you're done sharing your apps, they'll appear in the Built for your colleagues section within the Teams app store. If you're a Teams admin, you can also pin apps for users in the Teams app bar for even easier discovery.



> **Note**
>
> - If your app doesn't appear in the **Built by your colleagues** section, your Teams app might be caching information. If this happens, try signing out, and sign back in if

you're using the Teams desktop app. If you're using Teams web app, try refreshing your browser.

- You need to be an owner within the team to share apps and edit table permissions. If you are not an owner, the **Manage permission** and **Share with colleagues** options won't appear.

Assign table permissions

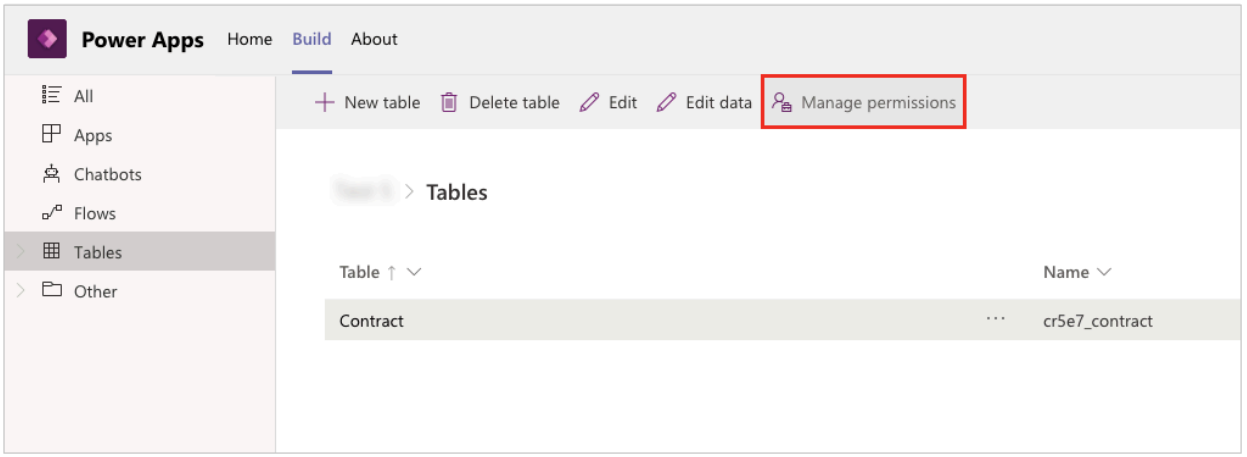
By default, the **Colleagues with access** role has no access to the data inside any of the custom tables you built using Dataverse for Teams. If you would like users outside of your team to access this data, you'll need to change the default access to one of these four permission sets:

- **Full access** – Allows end users to see and edit all records in the table.
- **Collaborate** – Allows end users to see all records, but they can only edit their own records.
- **Reference** – Provides a read-only view of data for end users.
- **Private** – Allows end users to only view and edit their own data.

> ⓘ **Note**
>
> You can also use this experience to alter the default access rights for members and guests within your team. By default, Team members are given **Full access** and guests are given **Private** access to new custom tables.

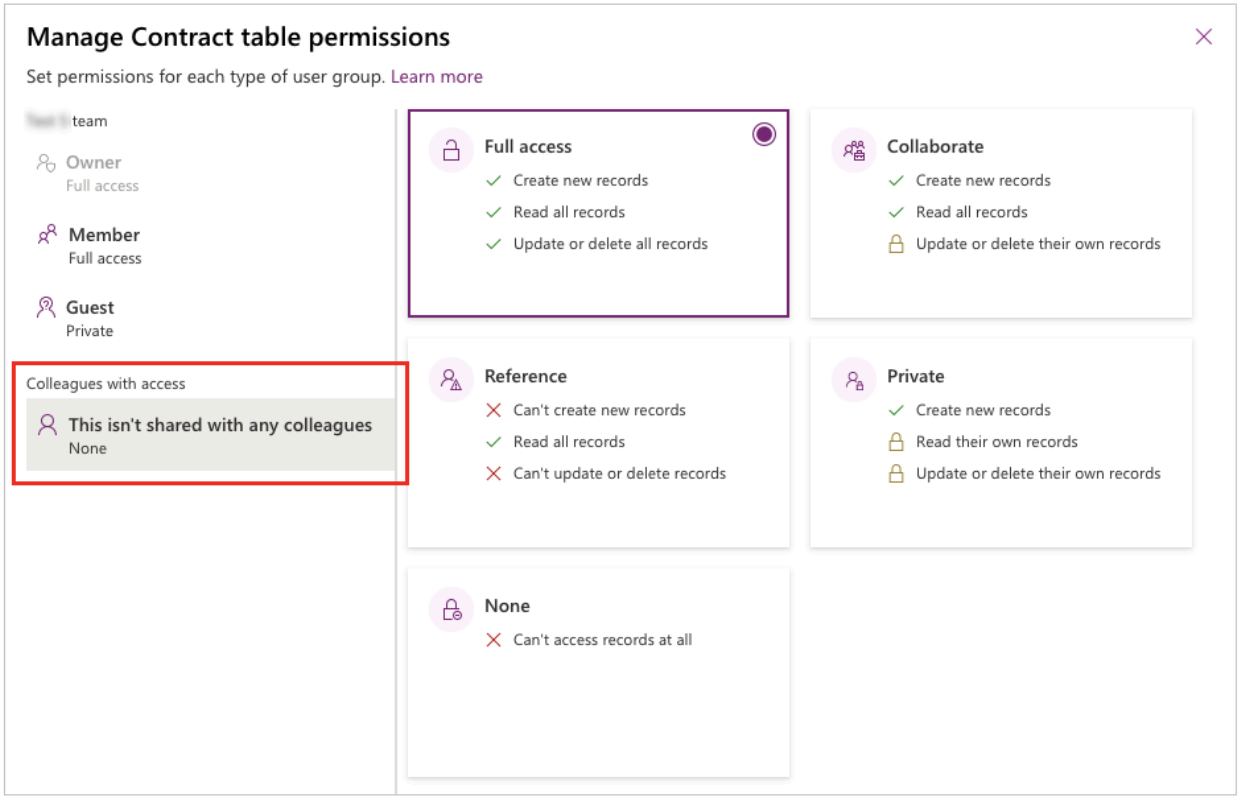
To set the permissions of a table:

1. Select **Build** tab.
2. Select the team that contains the app and tables you want to share.
3. Select **See all** under the list of resources in the team.
4. Select **Tables** from the left pane.
5. Select the table you want to share.



6. Select the **Manage permissions** in the command bar.

7. Under the **Colleagues with access** role, and select the permission set that you want to grant this role.



> **Tip**
>
> After you assign a security group to the **Colleagues with access** role, you'll see the name of the security group reflected in the table permission panel.

8. Select **Save**.

9. Repeat the above steps for the remaining tables in your app.

> **Important**
>
> When using **Share app with Colleagues in the Microsoft Teams app store** and the security group doesn't exist in the Dataverse organization, the Teams app will be created and assigned the **Basic User** role.

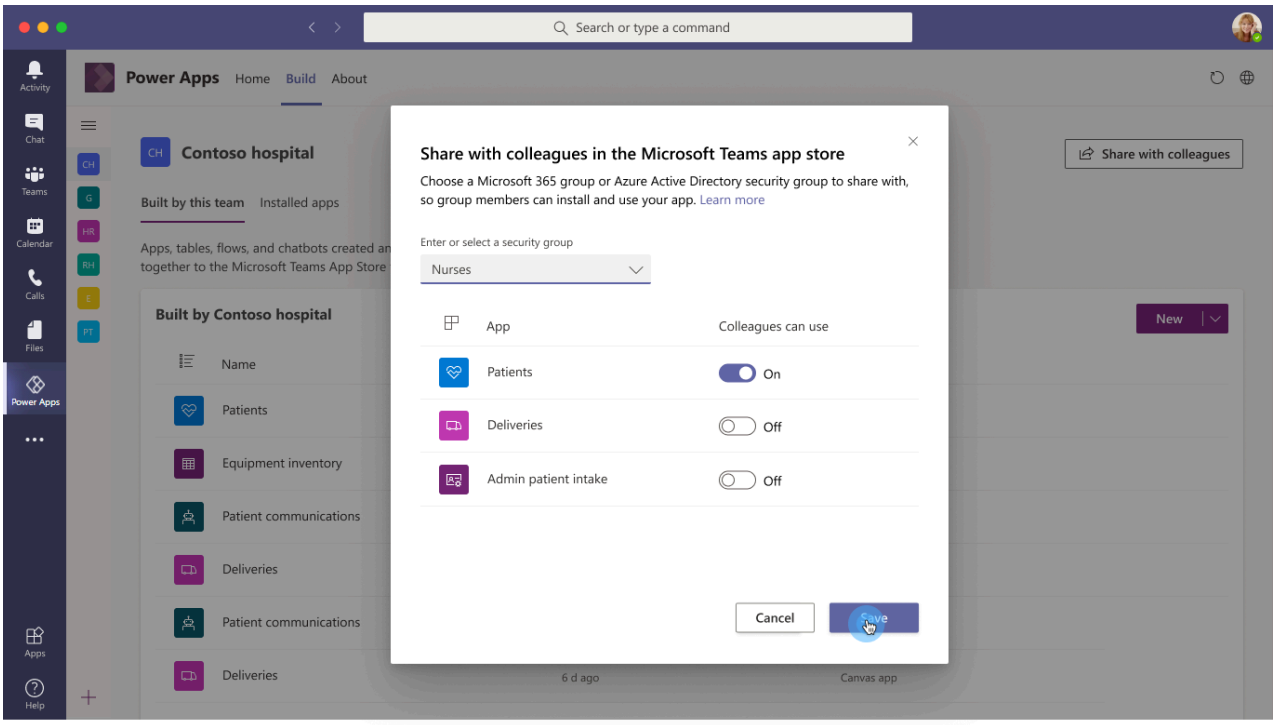
Assign the colleagues with access role to a security group and share the app

After you've completed setting the permissions to all your tables, you're now ready to share the app with an existing security group.

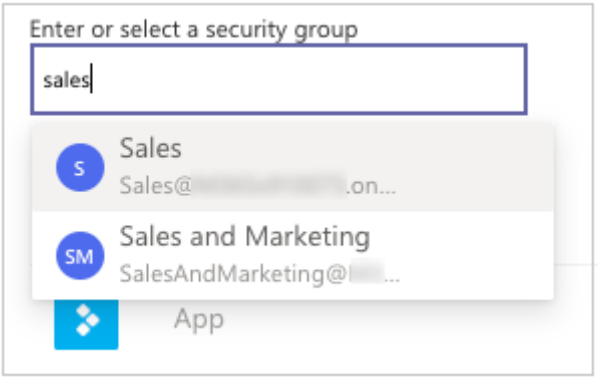
> **Note**
>
> - You can share an app to a single security group.
> - If you want to share the app to a Microsoft 365 group, it must be security enabled.
> - The security group's membership type must be **Assigned**. More information: Group membership types in Azure Active Directory
> - Your Power Apps administrator may have applied the limit to the maximum size of the security group. By default, this limit is 10,000 members. More information: List tenant settings (preview) - powerPlatform.teamsIntegration.shareWithColleaguesUserLimit, Set-TenantSettings

To share an app:

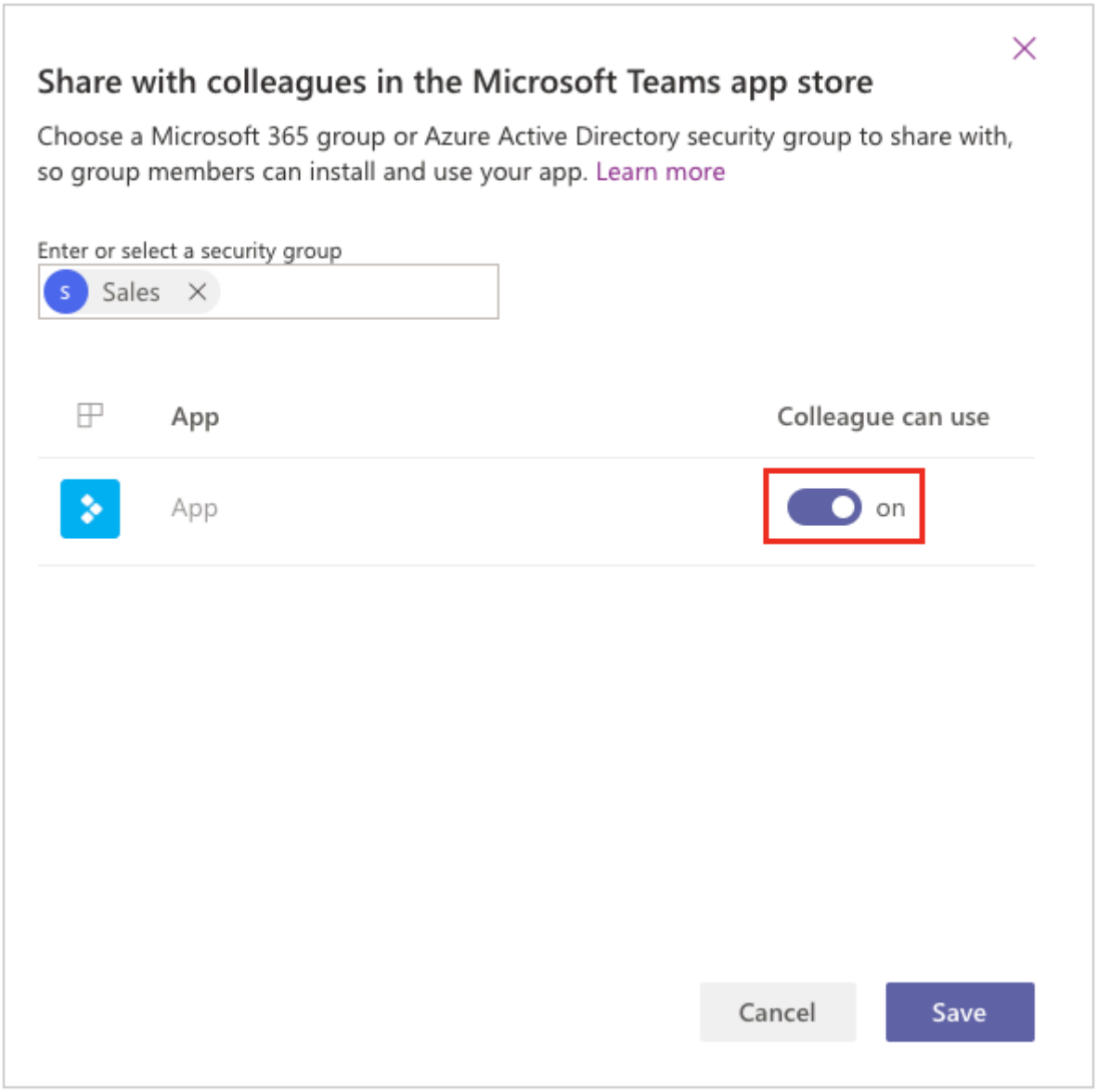
1. Select **Build** tab.
2. Select the team that contains the app you want to share.
3. In the top-right, select the **Share with colleagues**.



4. Search for, and select the security group you want to share the apps and tables with.



5. Set **Colleague can use** toggle to **On** for the apps and tables that you want to share with the security group selected in the previous step.



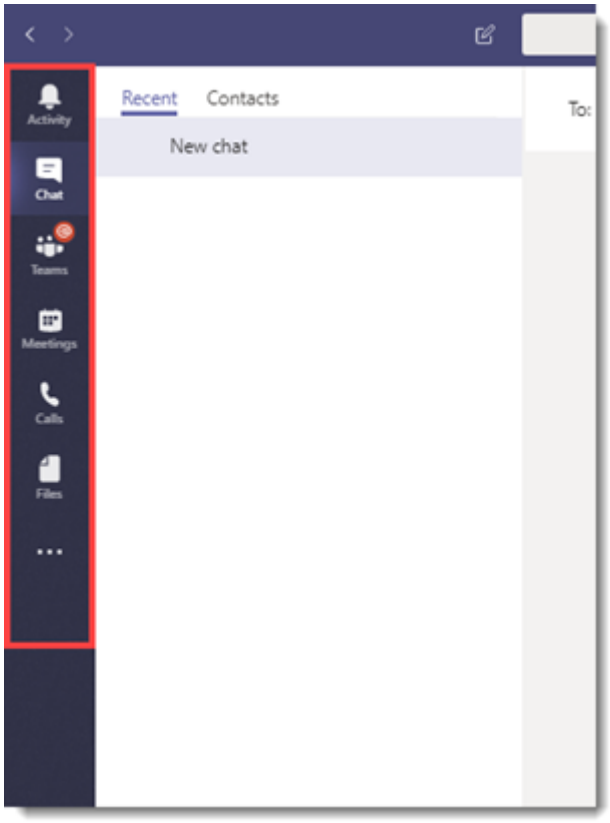
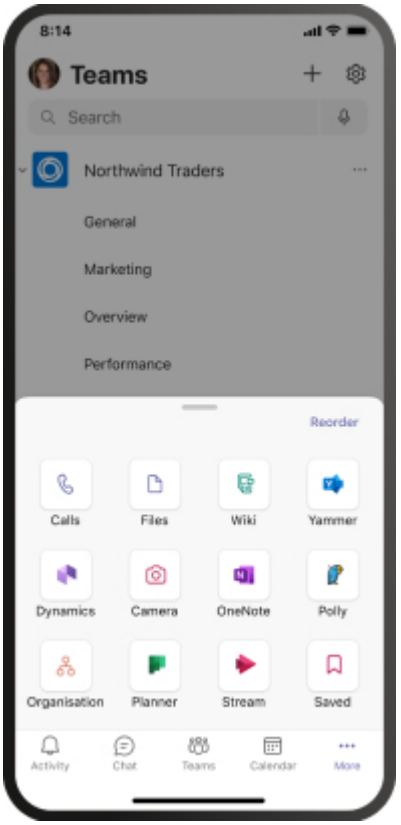
6. Select **Save**.

The apps you selected will now appear in the Built for your colleagues section within the Teams app store.

Optional: Pin the app to the app bar as a tenant administrator

If you're a tenant administrator, you can go one step forward and pin the apps for end users in Teams. To pin apps in Teams, follow the steps to add an app to the app catalog, and then, edit or create a new Teams app setup policy with your app.

Expand table

Teams desktop client	Teams mobile client
	

See also

Publish your app in Teams

Publish your app in Teams

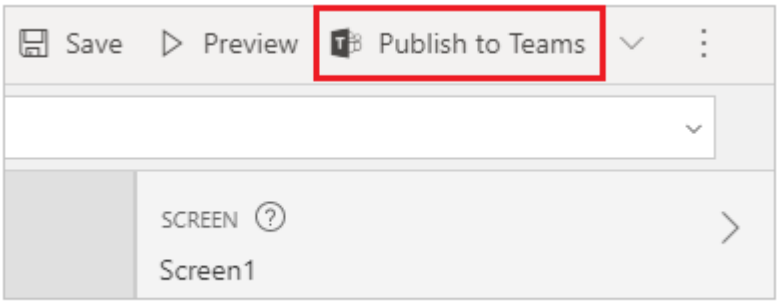
Article • 12/16/2022

In this article, learn about how you can publish and add apps to Microsoft Teams team.

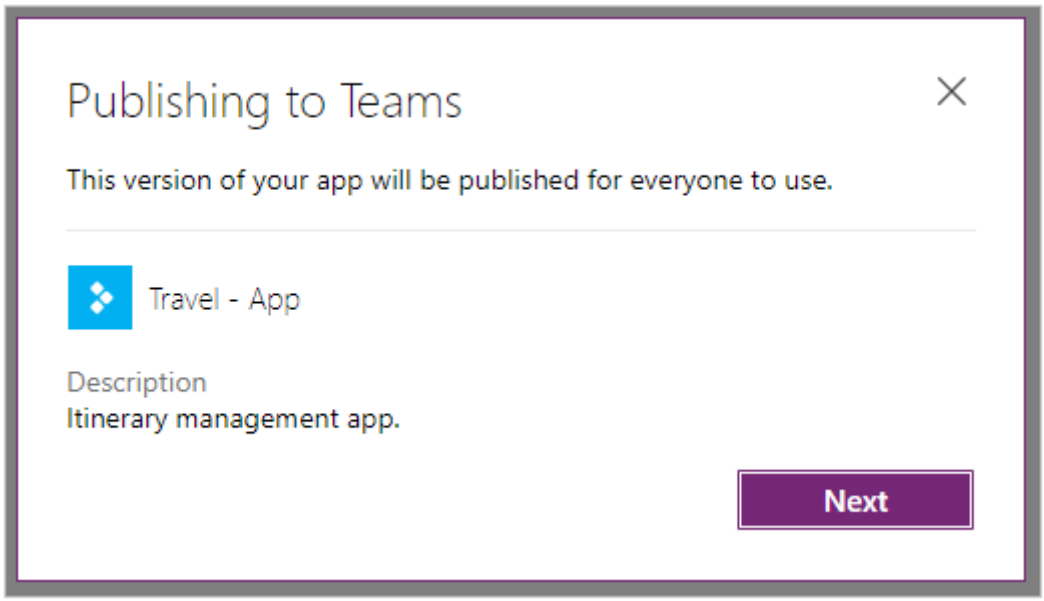
Publish and add an app to Teams

Whenever you save changes to a canvas app, you automatically publish them only for yourself and anyone else who has permissions to edit the app. When you finish making changes, you must explicitly publish them to make them available to everyone the app is shared with.

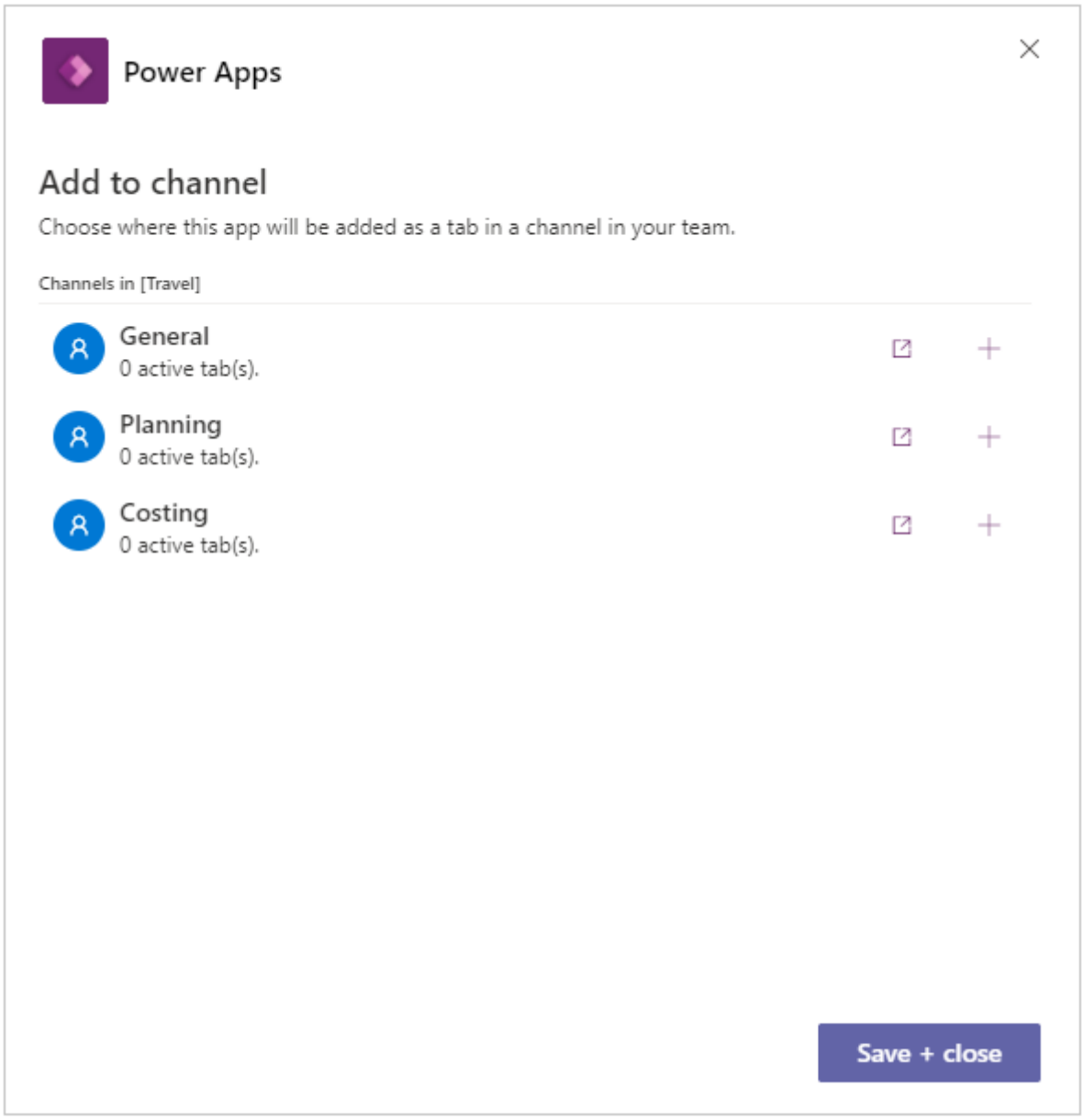
To publish an app, select the **Publish to Teams** button from the top-right side of Power Apps Studio.



Select **Next.** Publishing an app creates a new version of the app and makes the published version of the app available to everyone else the app is shared with.

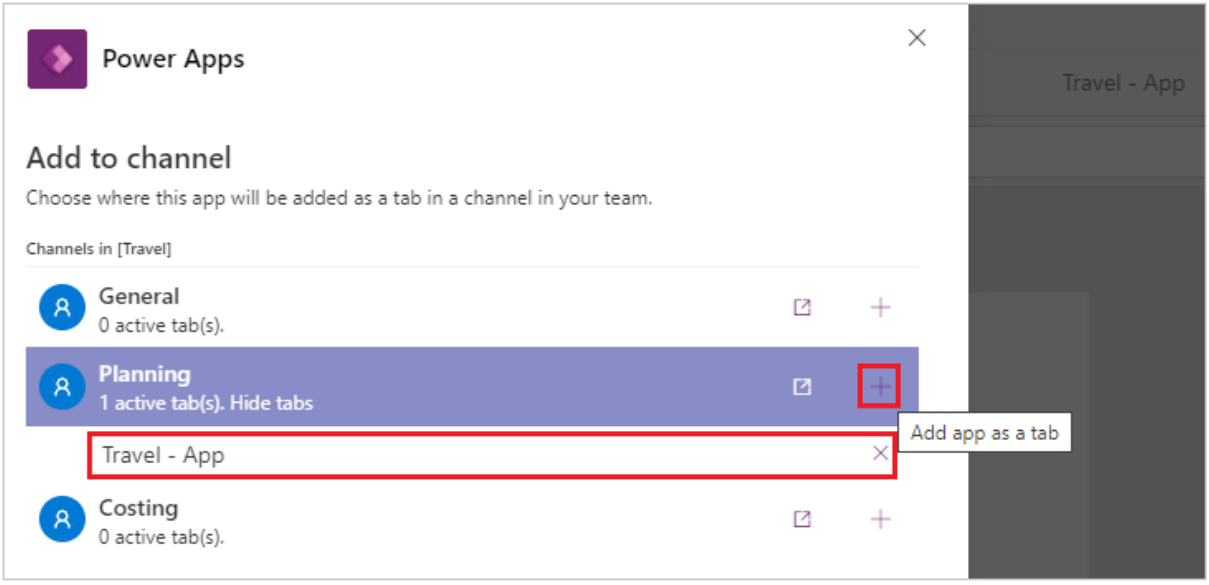


A list of all channels is displayed. You can add the app to one or more channels as a tab in the team.



Select **Add a tab** +, and then select **Save + close**.

> **Tip**
>
> If the app is already published to a Teams channel as a tab, you'll have the option to **Show tabs**.
> To publish updated version of an app to an existing active tab, select **Save + close** without adding a new tab.
> To add published version of an app to a new tab in a different channel, Select **Add a tab** + for the channel you want, and then select **Save + close**.



More information: Publish an app

Move an app to another Teams channel in the same team

Moving an app to another Teams channel in the same team requires publishing the app to the new channel, and then, removing the app from the old channel.

- To publish an app to the new channel: Follow the steps to publish and add an app to Teams. But before you select **Save + close**, select **Add a tab** + for the new channel you want. And then, select **Save + close** to publish the app to the new channel.
- To remove the app from a channel: Follow the steps to remove an app.

See also

Set permission and share apps with colleagues

Manage your apps in Teams

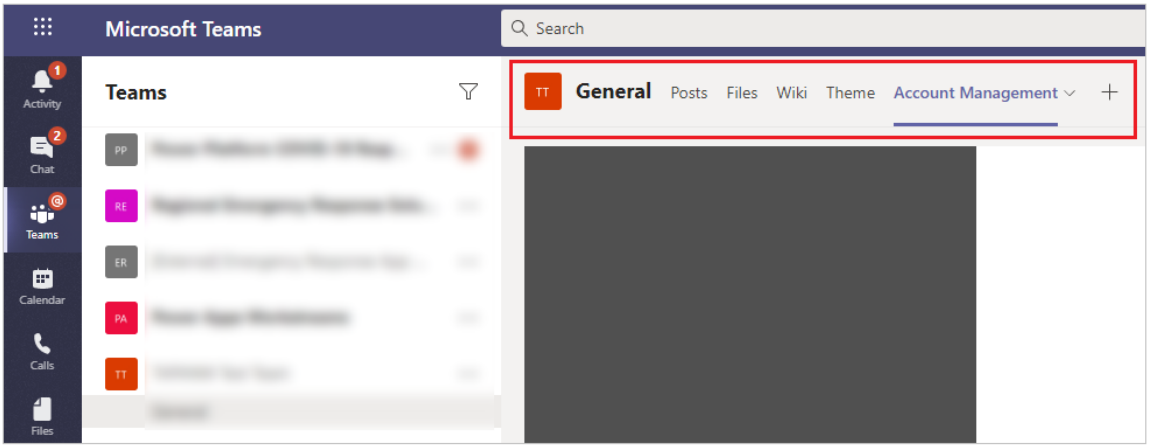
Remove an app from your team

Article • 12/16/2022

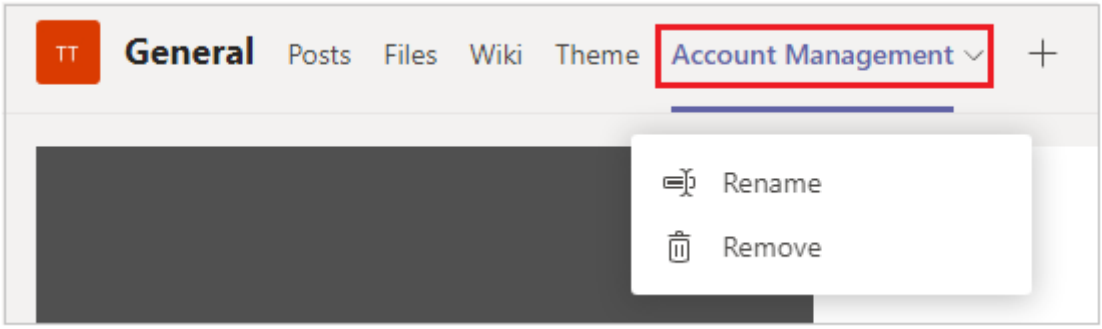
Removing an app that was added as a tab from a Microsoft Teams channel makes it inaccessible from the Teams tab for all members of the Teams channel.

To remove an app from your team:

1. Go to the team that you added the app to.

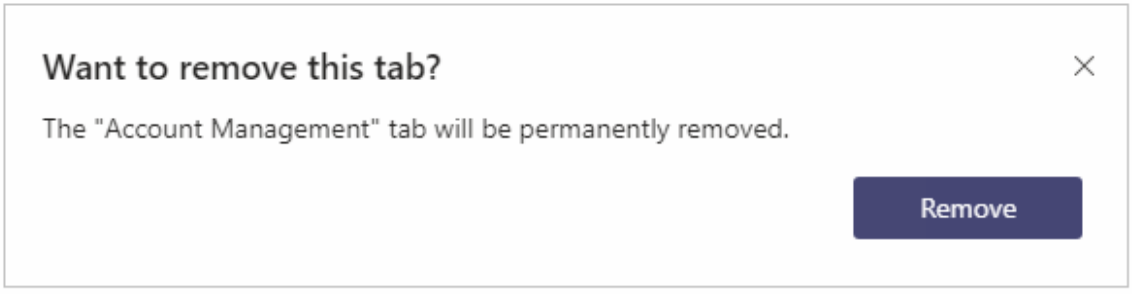


2. Select the drop-down for the tab.

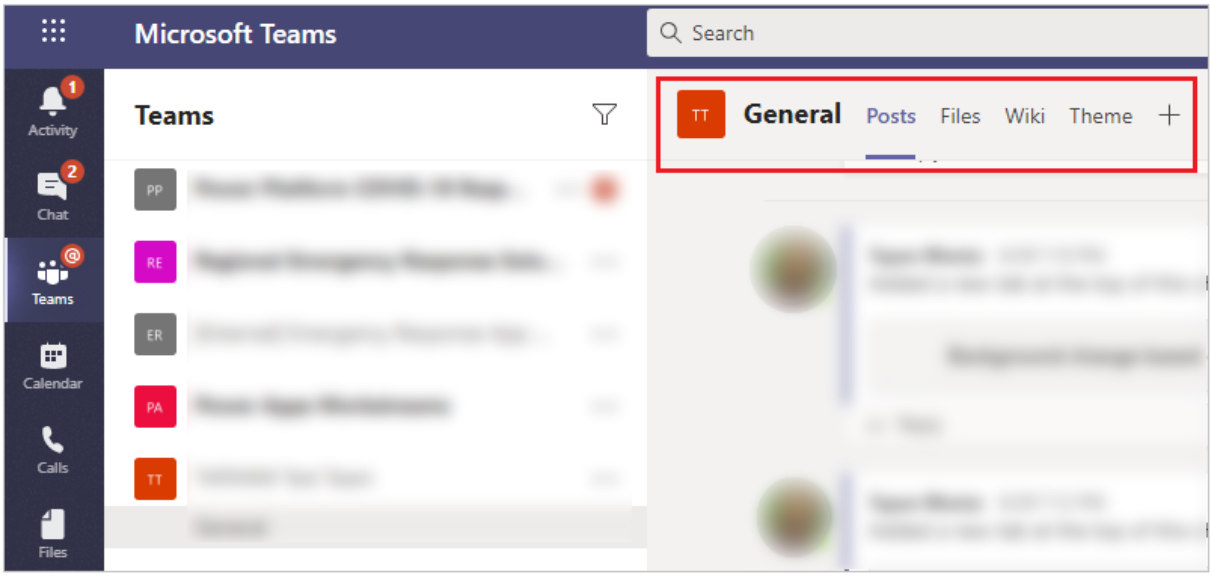


3. Select **Remove**.

4. Confirm **Remove**.



The app is now removed from your Teams channel.



See also

Uninstall the Power Apps personal app

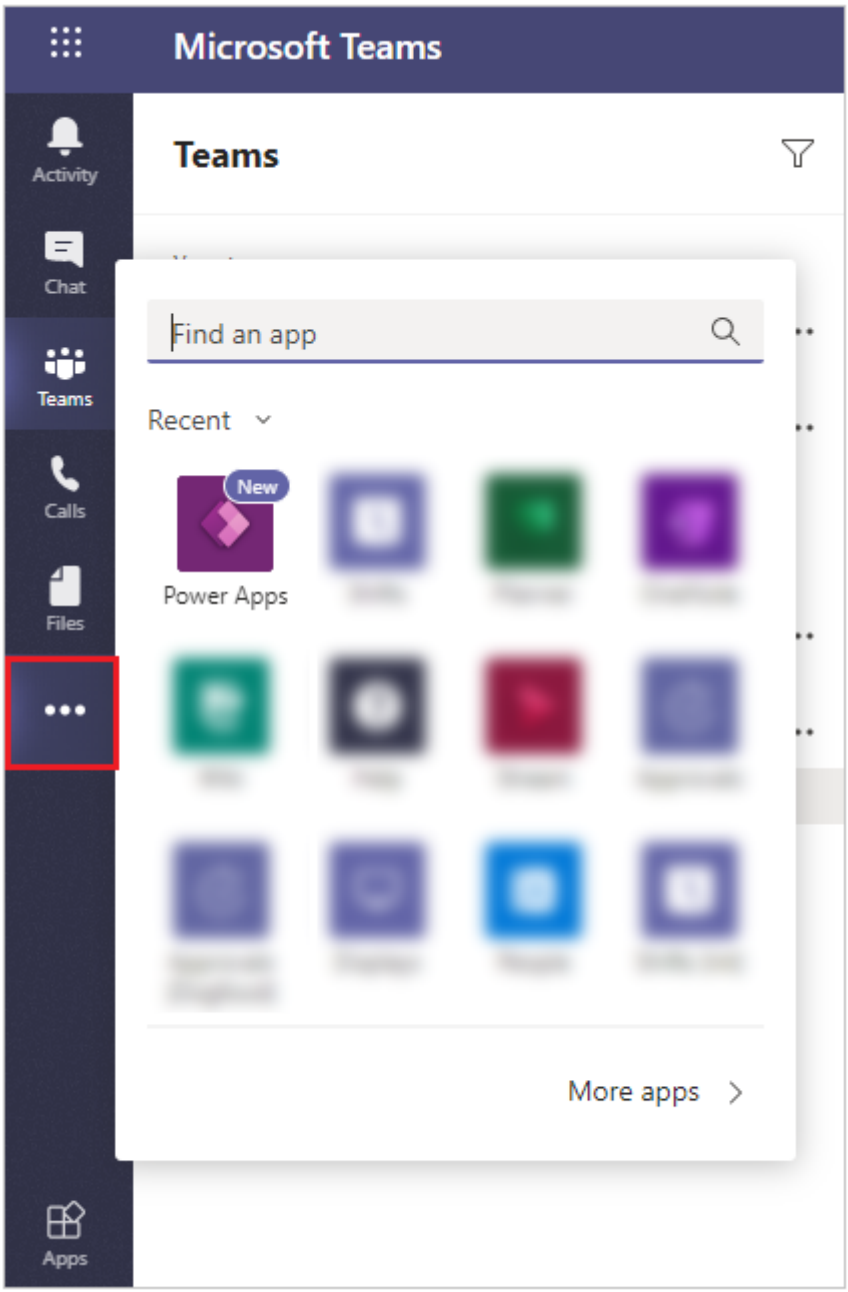
Uninstall the Power Apps personal app

Article • 03/03/2023

Uninstalling the Power Apps app removes the app as a personal app available to your account. Uninstalling the Power Apps app doesn't uninstall or remove the apps created by using Power Apps from the respective Microsoft Teams channels.

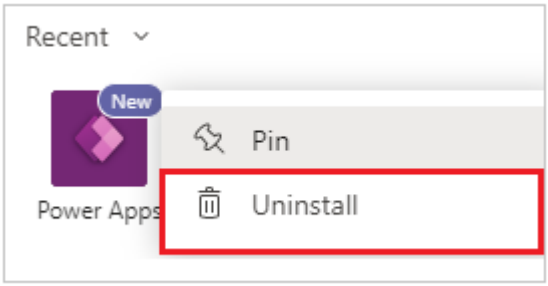
To uninstall the Power Apps personal app:

1. Select ••• on the left pane in Teams.

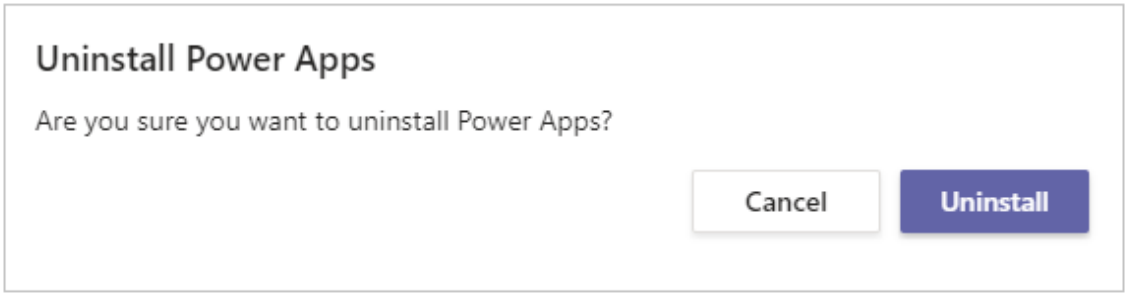


2. Find the Power Apps app by using search, if it's not visible in your most recent view.

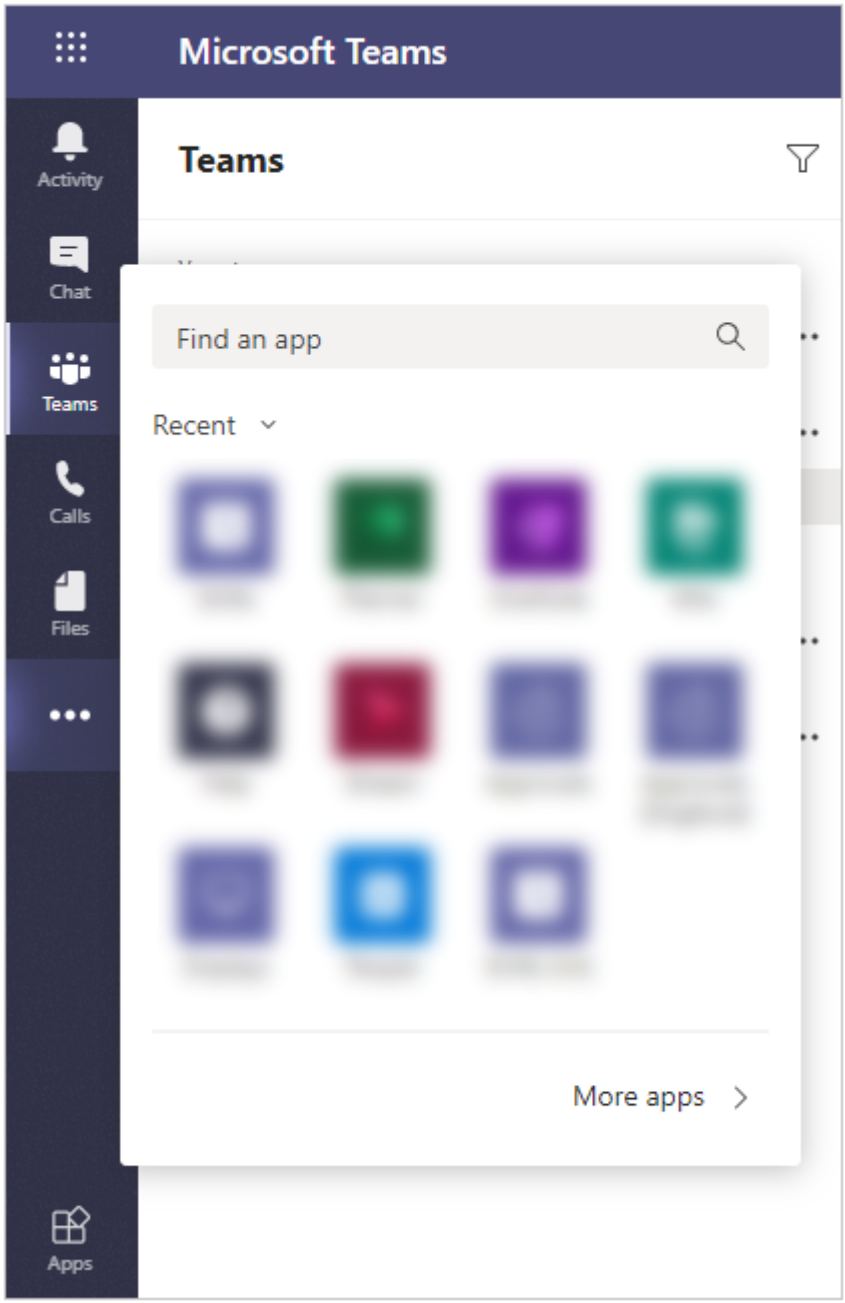
3. Right-click to select **Power Apps** app, and then select **Uninstall** from the shortcut menu.



4. In the window that appears, select **Uninstall**.



The Power Apps app is now uninstalled from Teams.



See also

Install the Power Apps personal app

Use sample app templates in Microsoft Teams

Article • 02/15/2023

Several sample app templates are available for use in Microsoft Teams that you can download from here. You can choose the sample app template that best fits your business requirement and quickly install it to get started. Sample app templates created with Power Apps typically consist of multiple components such as apps, flows, and tables.

> **Important**
>
> Sample app templates available for **download** are provided as-is. For any issues, feedback, support, or changes related to sample app templates, go to **https://github.com/microsoft/teams-powerapps-app-templates/issues** and submit an issue or a pull request.

Available sample app templates

The following table lists sample app templates available for download.



Boards

A simple way to connect and share with people in your organization with similar interests.



Bulletins

Manager and user apps for company communications.



Employee ideas

App for campaigns and ideas.



How to

Learn how to be a Power Apps maker.



Inspection

Manager and user apps for area inspections.



Issue reporting

Manager and user apps for issue reporting.



Milestones

App to keep track of projects, and initiatives.



Perspectives

A simple way to add topics and extend the topics with Q&A for discussions.



Profile+

Quickly find out about people in your organization.



Get connected
Connect with people in your organization with similar skills.

The following tutorial shows how to install the Employee ideas app. You can use similar steps to install other sample app templates.

Install the sample app template

To learn about installing sample app templates, go to https://github.com/microsoft/teams-powerapps-app-templates.

> **Note**
>
> - Installing sample app templates automatically enables the selected team's Microsoft 365 Group for security. More information: **Enable security for the selected team's Microsoft 365 Group**
> - If you can't upload or install uploaded sample app templates, check your **app permission policies in Microsoft Teams**.

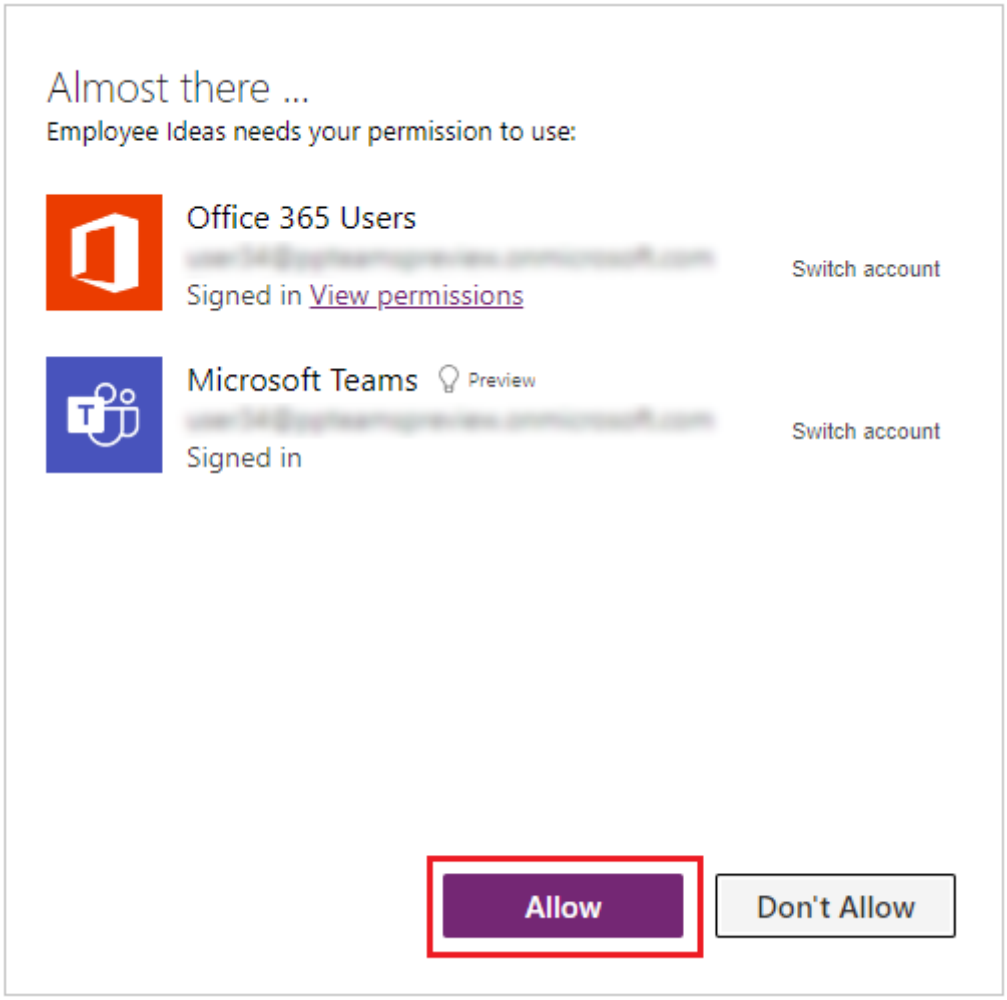
Run the sample app template

To run the installed app, select the app's tab from the team channel. In this example, we'll use **Employee ideas**.

Step 1 - Allow connections

Before you can use the app, it might ask your permission to use connections.

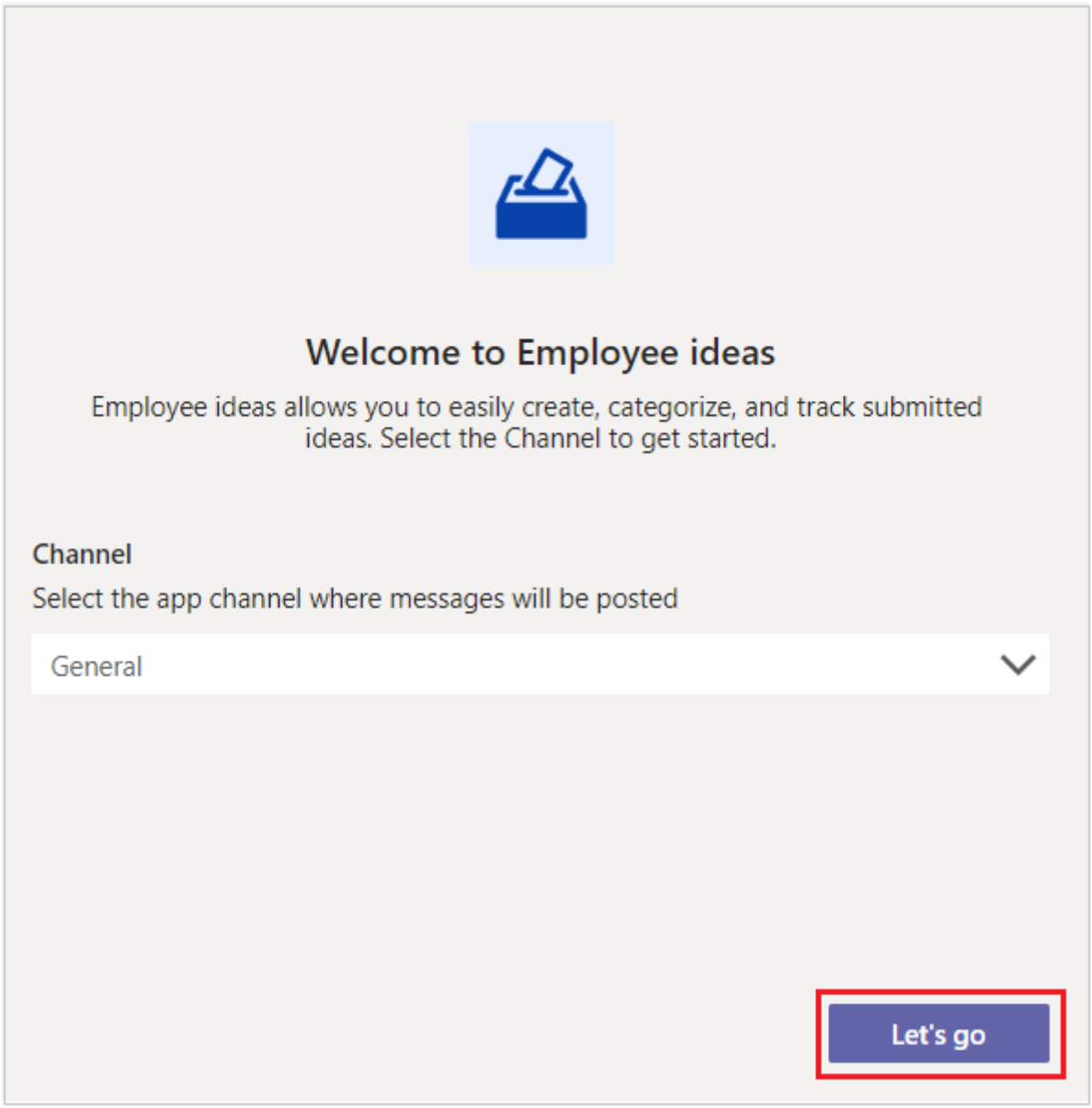
> **Note**
>
> The list of connections you see below is an example. The number and types of connections asking for your permissions may vary depending on the app you install.



Select **Allow**, so the app can use connections.

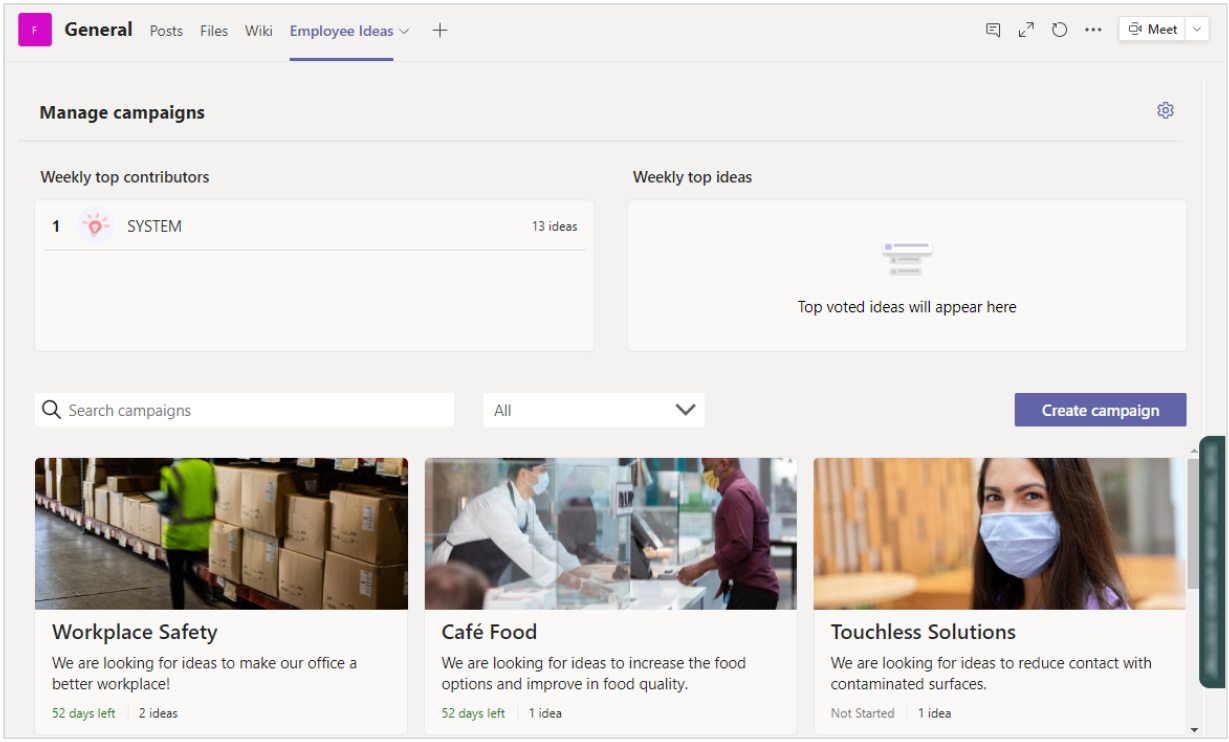
Step 2 - Select the channel

Select the channel (in this example, the channel is populated automatically), and then select **Let's go**.



Step 3 - Use the app

The **Employee ideas** app is now open and ready for your use.



For more details about using the **Employee ideas** app, go to Employee ideas.

Edit the sample app template

You can further customize and edit the components of an installed Power Apps app in Teams. More information: Manage your apps

Report installation errors

If you get any errors during the installation process, you can help us troubleshoot the problem much more effectively by selecting **Ctrl**+**Alt**+**A** on the keyboard to get the session details. For more information about session details, go to About tab.

For issues related to the sample app templates, report an issue at https://github.com/microsoft/teams-powerapps-app-templates.

Sample app template updates

Updates to the sample app templates are published immediately. If you already have a sample app template in use when this happens, a new version of the app is created with these updates and published as the live version. If you customized a sample app, you can restore the app to a previous version with your customizations, or recreate the customizations on top of the updated live version. When restoring the app to preserve

your customizations, ensure to check the app functionality for compatibility with the updates.

Sample app templates FAQs

For Frequently Asked Questions about sample app templates, go to Sample app templates FAQs.

Customize sample app templates

Article • 02/08/2023

In this article, you'll learn about customizing sample app templates. For example, you may need to change the company logo, update the data inside the sample app templates to fit your business requirement, add or remove controls on the screens to extend the app.

Prerequisites

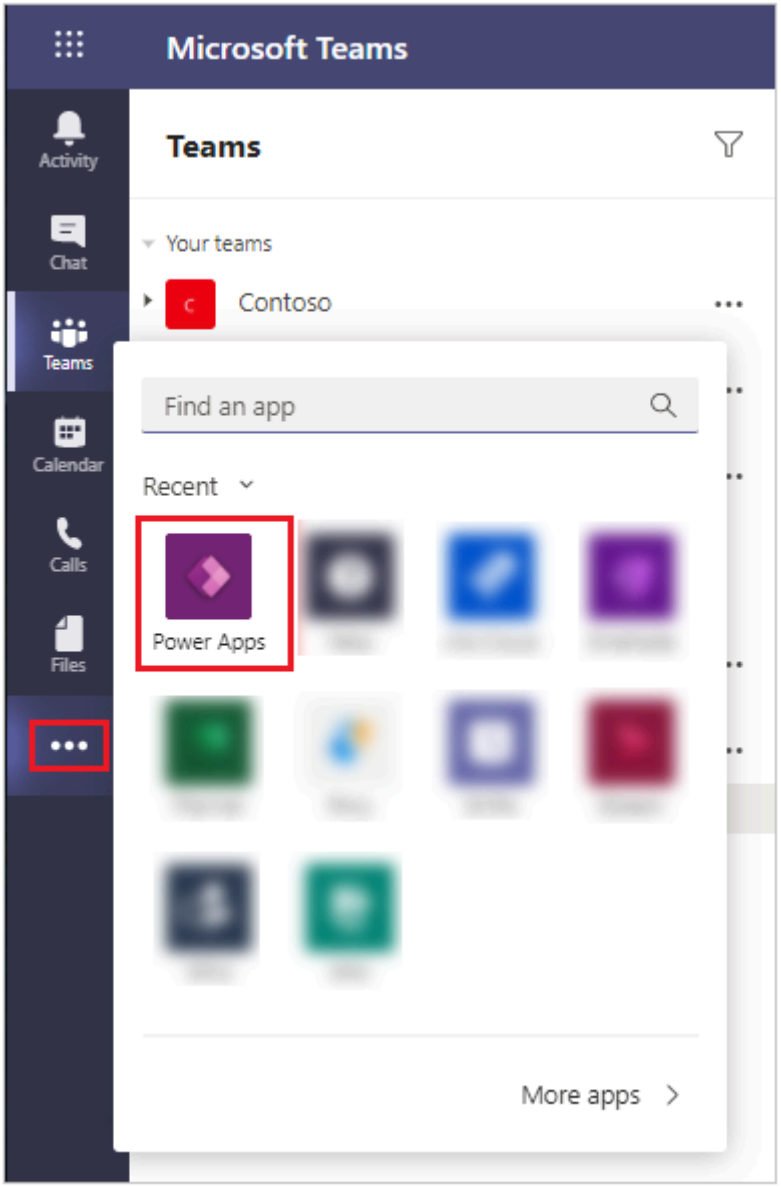
Install a sample app template. More details: Use sample app templates in Microsoft Teams.

> **Note**
>
> - Sections of this article uses one of the available sample app templates as an example. The tables, controls, screens and other components of the apps may vary depending on the app you installed. Use discretion when customizing the app that you select.
> - Read **Sample app template updates** before you customize sample app templates to understand the impact of new releases to the sample app template versions.

Open the sample app templates in Power Apps Studio

To start with customizing the sample app templates, the first step is to open the app inside Power Apps Studio.

1. Sign in to Teams.
2. Select  (*More added apps*).
3. Select Power Apps.



4. Select Build tab.
5. Select the team environment where you installed the sample app template.
6. Select **Installed apps**.
7. Select the sample app template that you installed.
8. If prompted, select the location.

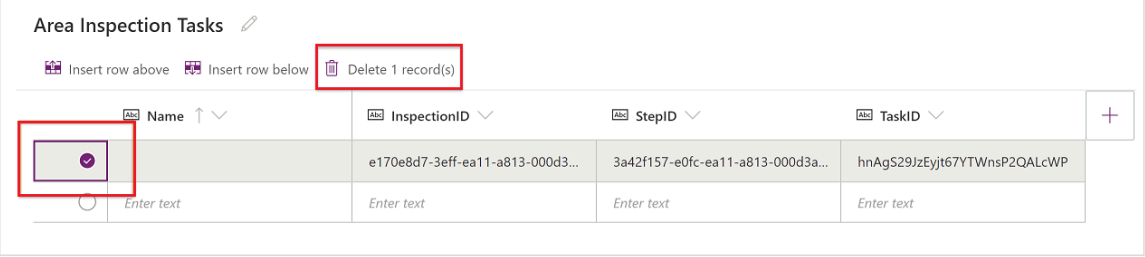
More information: Edit an app in Power Apps Studio

Remove sample data

When you install sample app templates, the tables might be pre-populated with sample data. The following table lists the sample app templates and the list of tables with the sample data to be removed:

To remove data from the sample app templates:

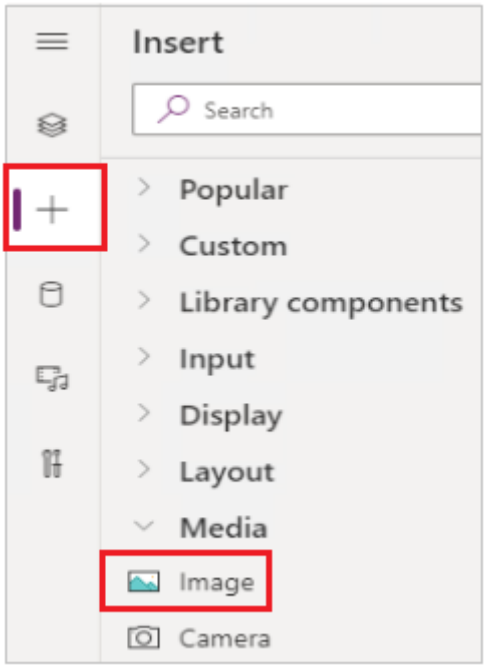
1. Open the sample app template (for example, **Manage inspections**) in Power Apps Studio.
2. Select **Data** from the left pane.
3. Select ... next to the table name (for example, *Area Inspection Task* for **Manage inspections** app).
4. Select **Edit data**.
5. Delete all rows containing sample data.



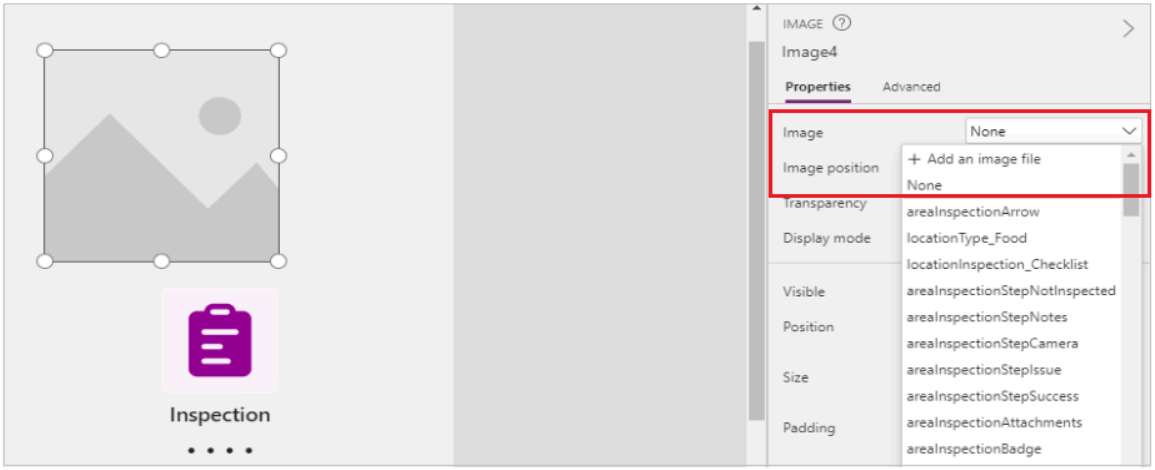
6. Close the visual editor.
7. Repeat the previous steps for the remaining tables.

Add your logo to the loading screen

1. Open the sample app template (for example, **Inspection**) in Power Apps Studio.
2. Select Tree view from the left pane.
3. If not already selected, select **Landing Screen**.
4. Select **Insert**.
5. Select **Image** control in *Media*.

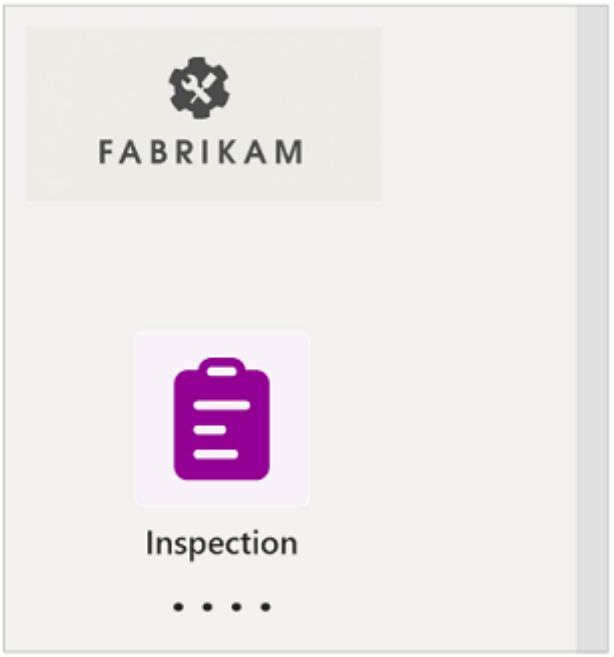


6. From the properties pane on the right side, select the **Image** drop-down, and then select **+ Add an image file**.



7. Select image for your company logo, and then select **Open**.

8. Position and size image control to the position you want on loading screen.



Change the welcome screen text

> **Note**
>
> The Inspection app welcome screen greeting text uses a formula containing global variables to correctly display the desired terminology for inspections and items being inspected. Use caution when changing this formula.

1. Open the sample app template (for example, **Inspection**) in Power Apps Studio.
2. Select the text label control with the greeting text **Glad to have you here**.

   The formula for the greeting text is in this format:

```
Power Fx

"Glad to have you here, we are ready for you to " &
If(Lower(gblWorkType)="inspection", "inspect", Lower(gblWorkType)) &
Switch(
Left(
    Lower(areaLabel),
    1
),
"a",
" an ",
"e",
" an ",
"i",
" an ",
"o",
```

```
" an ",
"u",
" an ",
" a "
) & Lower(areaLabel)
```

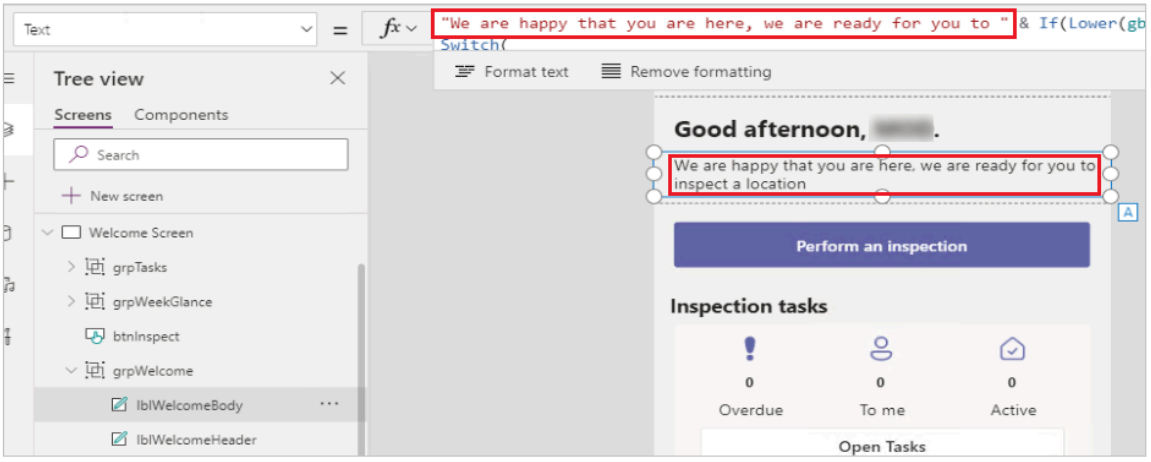
3. Select the formula bar.

4. Select the phrase **Glad to have you here**.

5. Update the greeting text, such as **We are happy that you are here**.

Power Fx

```
"We are happy that you are here, we are ready for you to " &
If(Lower(gblWorkType)="inspection", "inspect", Lower(gblWorkType)) &
Switch(
Left(
    Lower(areaLabel),
    1
),
"a",
" an ",
"e",
" an ",
"i",
" an ",
"o",
" an ",
"u",
" an ",
" a "
) & Lower(areaLabel)
```

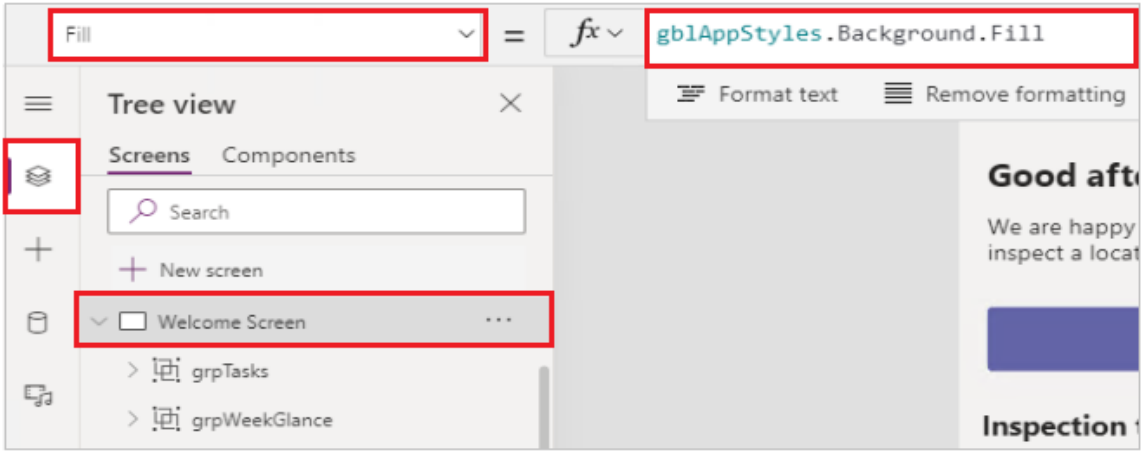


Change the screen background color

> **Note**
>
> The Inspection app leverages global theme variables to ensure consistent user experience. If you modify a screen fill, the modified screen will no longer use the standard app theme.

1. Open the sample app template (for example, **Inspection**) in Power Apps Studio.
2. Select Tree view from the left pane.
3. Select **Welcome Screen** from Tree view.
4. Select Fill from the property list on the top-left.
5. In formula bar, replace the formula with the color that you want.

   Screen background fill color will be set to the selected background color.



Bulk edit data

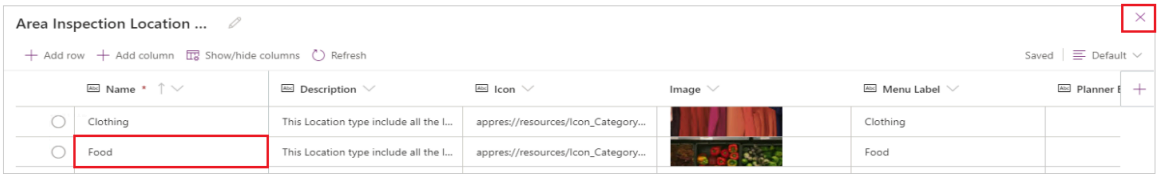
If you want to quickly edit the data in the app, you can open the table and modify or delete data.

> **Caution**
>
> Sample app templates uses several related tables, such as Area Inspection Locations and Area Inspection Location Types for Inspection app. Use caution when deleting records from the tables without checking data relationship and dependencies.

1. Open the sample app template (for example, **Inspection**) in Power Apps Studio.

2. Select **Data** from the left pane.
3. Select ... next to the table name (for example, *Area Inspection Location Types* for Inspection app).
4. Select the cell inside the Visual Editor, for example **Food** in the **Name** column.
5. Replace the text with the value you want.
6. Close the visual editor.

   Data in the app should reflect the updated value, such as the location name in this example.



Add a column to the app

> **Important**
>
> The Inspection app customization steps in this section covers some of the advanced configuration options. This example requires that you have familiarity with the Power Apps Studio, are comfortable with modifying formulas, and work with controls such as containers.

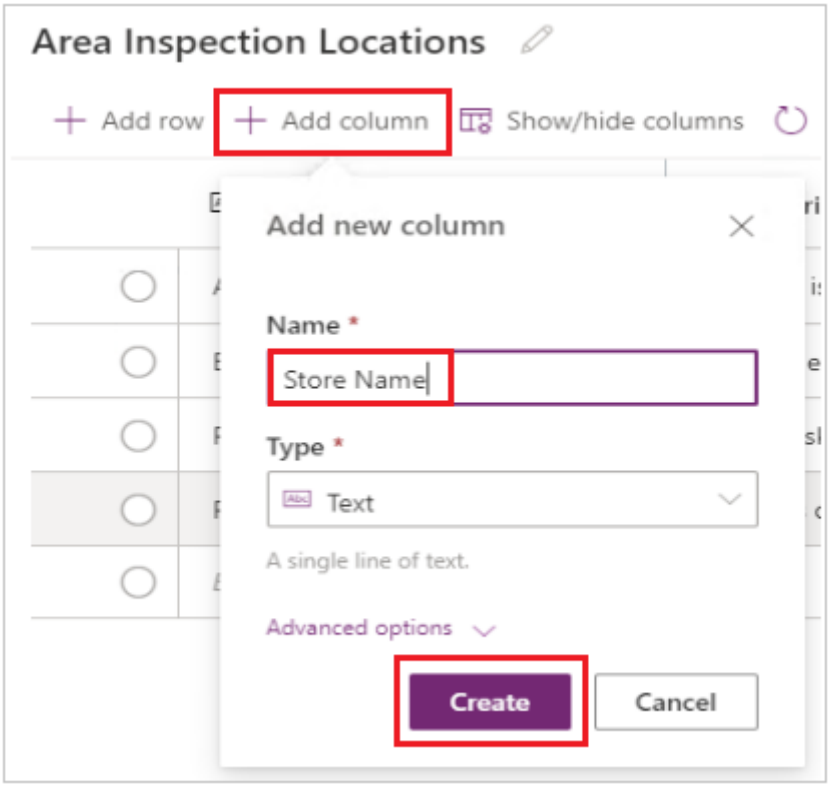
Consider a scenario where you have multiple stores, and you want to associate locations and inspections with specific stores.

Add Store Name text column to the Location table

1. Open the **Manage inspections** app in Power Apps Studio.
2. Select **Data** from the left pane.
3. Select ... next to the table name (for example, *Area Inspection Locations* for **Manage inspections** app).
4. Select **Add column**.
5. Enter **Name** as "Store Name".

6. Select **Create**.

7. Close the visual editor.



Add Store Name column to the location form

1. Select the **Locations Screen**.

> **Tip**
>
> You can ignore the error about the use of the preview connectors in this app.

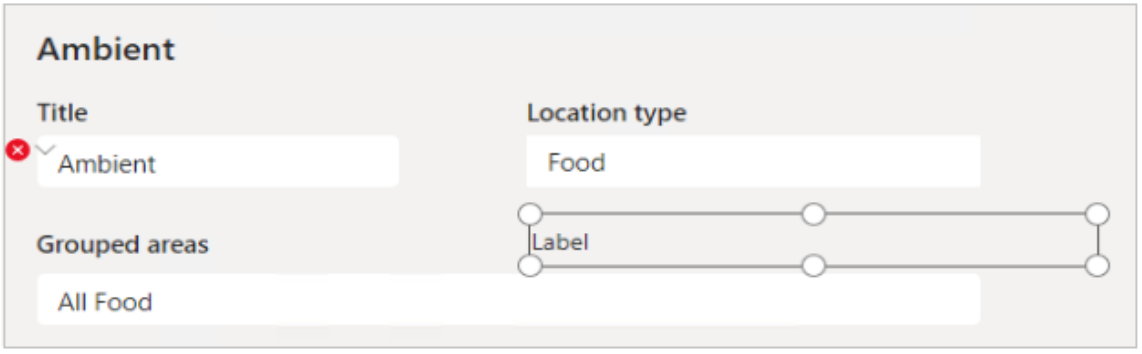
2. In Tree view, search for `btnArea_GroupedAreas`.

3. In the **Grouped Areas** on the form, drag the left edge to be parallel with the edge of the **Title** field.

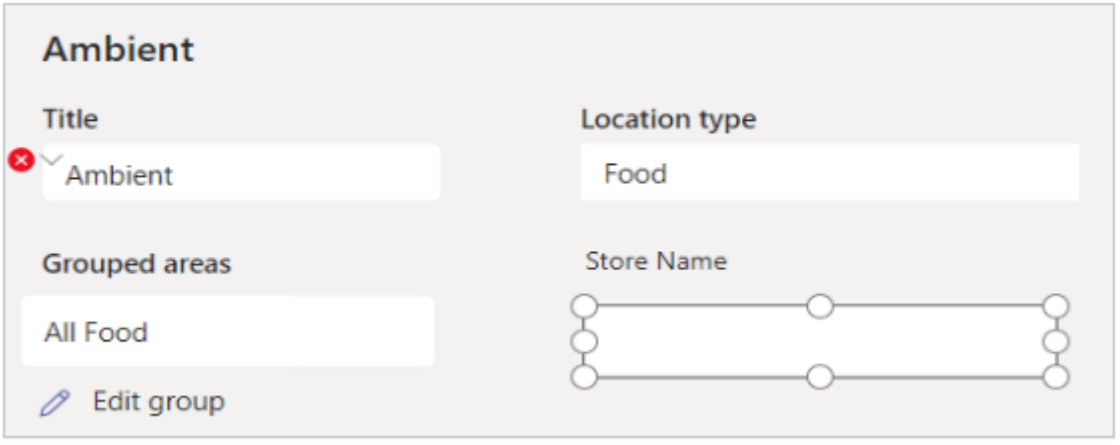
> **Note**
>
> The Grouped Areas field control is in a container. Hence, the selected area may appear in a different part of the screen than where you see the column. Dragging the selected area resizes the column.

4. Select **Insert** from the left pane.

5. Select **Label** control.
6. Position the **Label** so that it's aligned horizontally with **Grouped areas**, and vertically with the **Area type**.



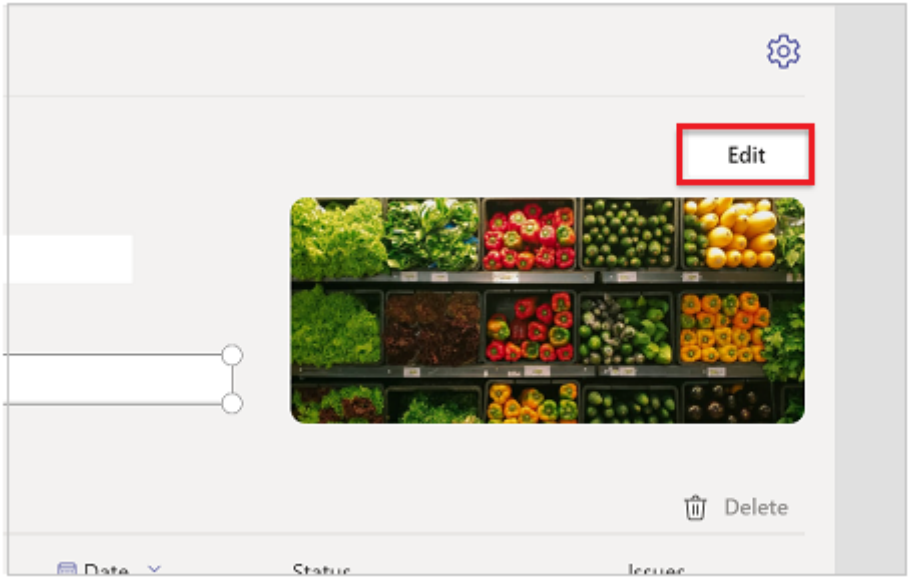
7. Select **Text** from the property list on the top-left.
8. Select the formula bar.
9. Change the formula to **Store Name**.
10. Select **Insert** from the left pane.
11. Select **Text box** control.
12. Drag the text box control to be aligned horizontally with **Grouped area** and vertically with **Area type**.
13. Select **Value** from the property list on the top-left.
14. In the formula bar, replace the current value with `gblLocation.'Store Name'`
15. Select **ColorBackground** from the property list on the top-left.
16. In the formula bar, enter "White".



17. Select **Display mode** from the property list on the top-left.

18. In the formula bar, enter "Edit" if not already present.

19. Hold the **ALT key** on your keyboard, and select the **Edit**.



20. Select **Tree view** from the left pane.

21. Search for `btnAreaDetails_Save` and select the returned control.

22. Expand the formula bar.

23. At the end of the 17th line, add a comma and this formula:

```
Power Fx
'Store Name':Microsoft_CoreControls_TextBox1.Value
```

This section of the formula should now look like the following:

```
Power Fx
{
    msft_name: txtArea_EditTitle.Text,
    'Location Type': cmbAreaDetails_AreaType.Selected,
    'Primary Image': picArea_UploadedImage.Image, 'Store
Name':Microsoft_CoreControls_TextBox1.Value
}
```

> **Note**

> If the updated formula shows error, verify the text box name for the store name. For example, it may be `Microsoft_CoreControls_TextBox2` instead of `Microsoft_CoreControls_TextBox1`, depending on your changes.

24. Select the label control added in step 5.
25. Select the **Y** property from the property list on the top-left.
26. Select the formula bar.
27. Change the formula to:

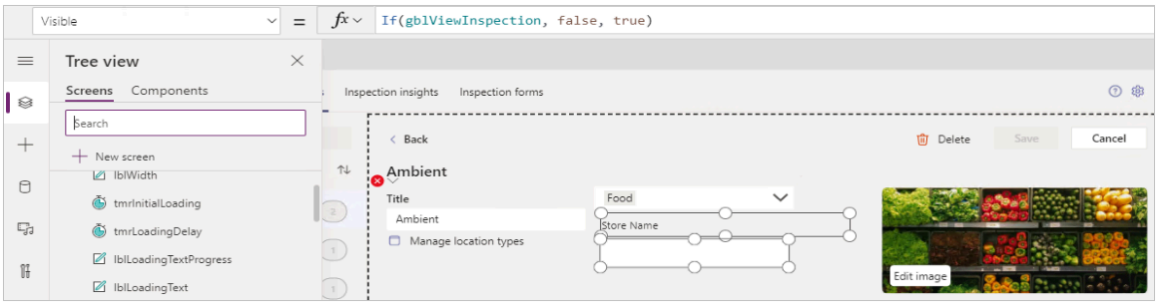
```
Power Fx
If(gblEditLocation,250,200)
```

28. Select text box added in step 11.
29. Select the **Y** property from the property list on the top-left.
30. Select the formula bar.
31. Change the formula to:

```
Power Fx
If(gblEditLocation,282,232)
```

32. Hold the **CTRL key** on your keyboard, and select both the store name label and the text box.
33. Select Visible property from the property list on the top-left.
34. Select the formula bar.
35. Change the formula to:

```
Power Fx
If(gblViewInspection, false, true)
```



Publish app updates to Teams

To save and publish the changes to the sample app:

1. Select **Save** from the top-right.
2. Select **Publish to Teams**
3. Select **Next**.
4. To publish updated version of an app to an existing active tab, select **Save + close**.

   To add published version of an app to a new tab in a different channel, Select + (Add a tab) for the channel you want, and then select **Save + close**.

More information: Publish and add an app to Teams

Feedback

Was this page helpful? Yes No

Provide product feedback

Localize sample app templates

Article • 02/08/2023

The sample app templates for Microsoft Teams have been localized to support eight different languages so that they can be used by people with different languages. Depending on the user's language, you can see the text in an app in the appropriate language, if that language has been included in the app localization.

For example, consider that you use the Russian language. Russian isn't a language included in the standard localization; however, you can still use the app in Russian—if you add that language and localize the app labels.

This article covers the approach used for localization of the sample app templates for Teams, and how to extend it to add support for another language (such as Russian). In this example, we'll use the Milestones app; but this same approach is used consistently across all of the Teams template Power Apps, and you can follow the same approach for localization of your own canvas apps.

Watch this video to learn how to localize sample app templates:
https://learn-video.azurefd.net/vod/player?id=2b55d792-7cde-4948-9072-5d38d33e844f&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Flocalizing-sample-apps

Prerequisites

To complete this lesson, we'd need the ability to create Power Apps within Teams, which is available as part of select Microsoft 365 subscriptions.

Understanding localization in Power Apps

In canvas apps, text labels and other visual text components can be localized by loading a static Excel data source to your app containing the localized value for each text string in each language. For more information about this process, see Language function.

For example, Milestones app contains an Excel data source called **StaticLocalizationsjon** with three columns—"OOBTextID", "LanguageTag", and "LocalizedText".

- **OOBTextID** is the identifier of the text that is to be localized.
- **LanguageTag** is the identifier of the language in which the text is to be localized. Language tag is a standard abbreviated text defining a language and the

country/region its being used in (for example, "en-US" means "English-United States").

- **LocalizedText** is the text in the appropriate language that needs to be displayed.

Here's an example with localized text for each text ID.

Expand table

OOBTextID	LanguageTag	LocalizedText
lblLoadingText_AppLoading__locText	en-US	Getting things ready ...
lblLoadingText_AppLoading__locText	de-DE	Dinge fertig machen ...
lblLoadingText_AppLoading__locText	es-ES	Preparando las cosas ...
lblLoadingText_AppLoading__locText	fr-FR	Préparer les choses ...
lblLoadingText_AppLoading__locText	it-IT	Preparare le cose ...
lblLoadingText_AppLoading__locText	ja-JP	準備をする..。
lblLoadingText_AppLoading__locText	nl-NL	Dingen klaarmaken ...
lblLoadingText_AppLoading__locText	pt-BR	Preparando as coisas ...
lblLoadingText_AppName__locText	en-US	Milestones
lblLoadingText_AppName__locText	de-DE	Meilensteine
lblLoadingText_AppName__locText	es-ES	Hitos
lblLoadingText_AppName__locText	fr-FR	Jalons
lblLoadingText_AppName__locText	it-IT	Passaggi fondamentali
lblLoadingText_AppName__locText	ja-JP	マイルストーン
lblLoadingText_AppName__locText	nl-NL	Mijlpalen
lblLoadingText_AppName__locText	pt-BR	Etapas
lblLoadingText_BuiltWith__locText	en-US	Built with Microsoft Power Platform
lblLoadingText_BuiltWith__locText	de-DE	Erstellt mit Microsoft Power Platform
lblLoadingText_BuiltWith__locText	es-ES	Creada con Microsoft Power Platform
lblLoadingText_BuiltWith__locText	fr-FR	Créé avec Microsoft Power Platform
lblLoadingText_BuiltWith__locText	it-IT	Creata con Microsoft Power Platform

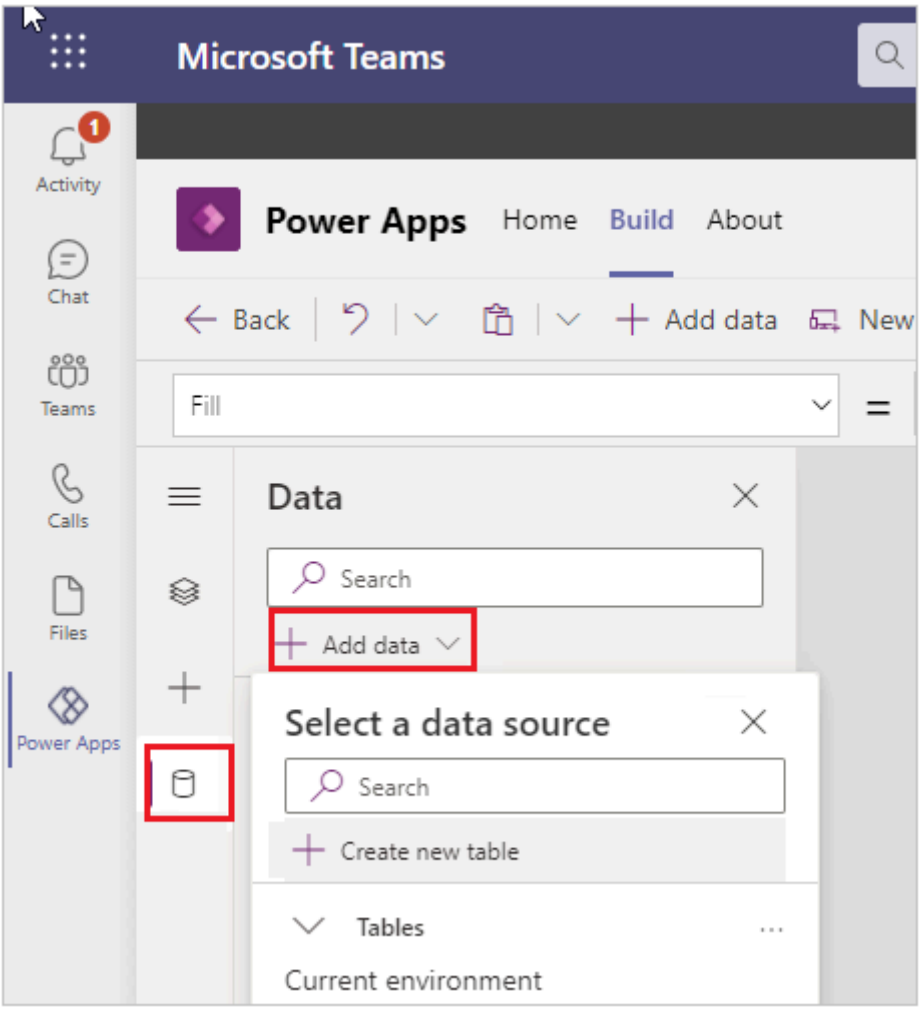
OOBTextID	LanguageTag	LocalizedText
lblLoadingText_BuiltWith__locText	ja-JP	Microsoft Power Platform でビルド
lblLoadingText_BuiltWith__locText	nl-NL	Gebouwd met Microsoft Power Platform
lblLoadingText_BuiltWith__locText	pt-BR	Criado com o Microsoft Power Platform

> **Note**
>
> The above table is from the master file that contains localized text for all the screens in the app. However, it's been filtered to show only the rows relevant to earlier discussions.

How to use Excel as a static data source in your app

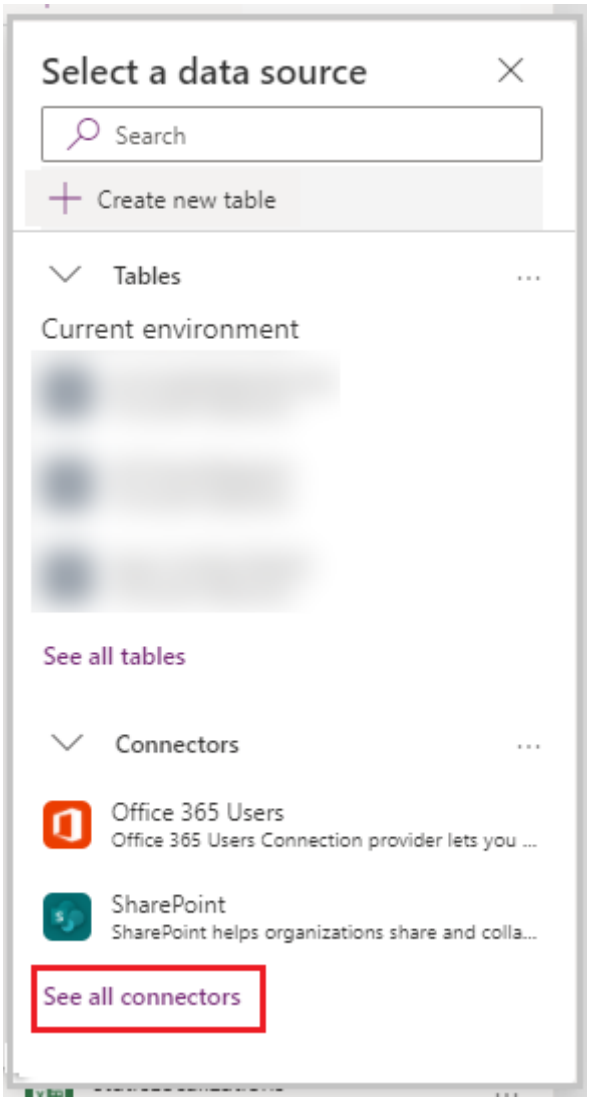
Static data is data loaded into Power Apps from Excel. This data isn't editable in the app. If you want to add a static data source to your app, perform the following steps.

1. In Teams, right-click on **Power Apps**, and pop out the app.
2. Select the **Build** tab.
3. Open the desired app to edit in Studio. In this example, we'll use **Milestones** app.
4. Select the **Data** tab.
5. Select **Add data**.



6. Select **Connectors**.

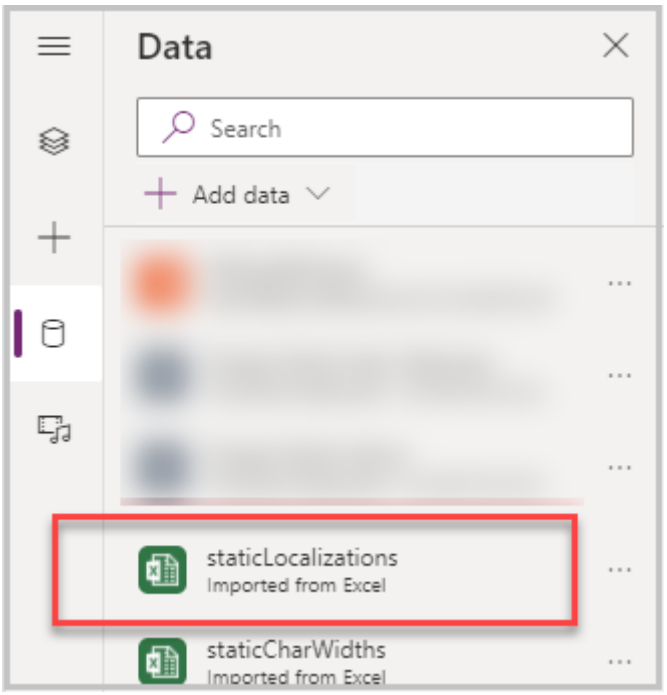
7. Select **See all connectors**.



8. Select **Import from Excel**.

9. Select the **LocalizationMilestones_AllLanguages** file, and select **Open**.

10. On the right of the screen, you're asked to choose a table. Select the table **staticLocalizations**, and select **Connect**. The table gets added as a data source and can be used in the app.



Adding a new language to your app

Now that you understand how localization is handled in Power Apps, if you want to add an additional language to the app, you don't have to update every text control in the app. The text formulas are built to dynamically display the appropriate localized copy of text strings based on the user's language.

To add Russian, you would first add Russian translation to the spreadsheet for each **OOBTextID**. In this example, we'll show how we can add localized text for the text on the first page of the app.

1. Open **LocalizationMilestoes_AllLanguages.xlsx** file.

2. Add the following rows to the Excel:

Expand table

OOBTextID	LanguageTag	LocalizedText
lblLoadingText_AppName__locText	ru-RU	Вехи
lblLoadingText_BuiltWith__locText	ru-RU	Создано на платформе Microsoft Power Platform
lblLoadingText_AppLoading__locText	ru-RU	Готовимся ...

3. To add Russian text for all pages, we'd have to add rows for all the OOBTextIDs as they exist for any other language, then add **ru-RU** as the Language tag. and then add the Russian translation for each of those OOBTextIDs.

4. Save Excel file.

> **Note**
>
> Since the Excel file already exists as a data source in the data sources list for the app, we'll have to remove the excel from the list of data sources and then readd it as shown above.

5. Update the app OnStart function **gblUserLanguage** to include the language that you added to the Excel file. In the example below we've added Russian:

```
Power Fx

//user language
Set(gblUserLanguage,Switch(Left(Language(),2),"de","de-DE","en","en-
US","es","es-ES","fr","fr-FR","it","it-IT","ja","ja-JP","nl","nl-NL",
"pt","pt-BR","ru-RU",
//default
"en-US"));
//build localization collection, with user's language
ClearCollect(colLocalization,Filter(staticLocalizations,LanguageTag =
gblUserLanguage));Collect(colLocalization,Filter(staticLocalizationsNew
Messages,LanguageTag = gblUserLanguage));
```

6. Save and publish the app.

Since the formulas dynamically retrieve the localized content, there's no need to update the formulas to reflect the newly added language.

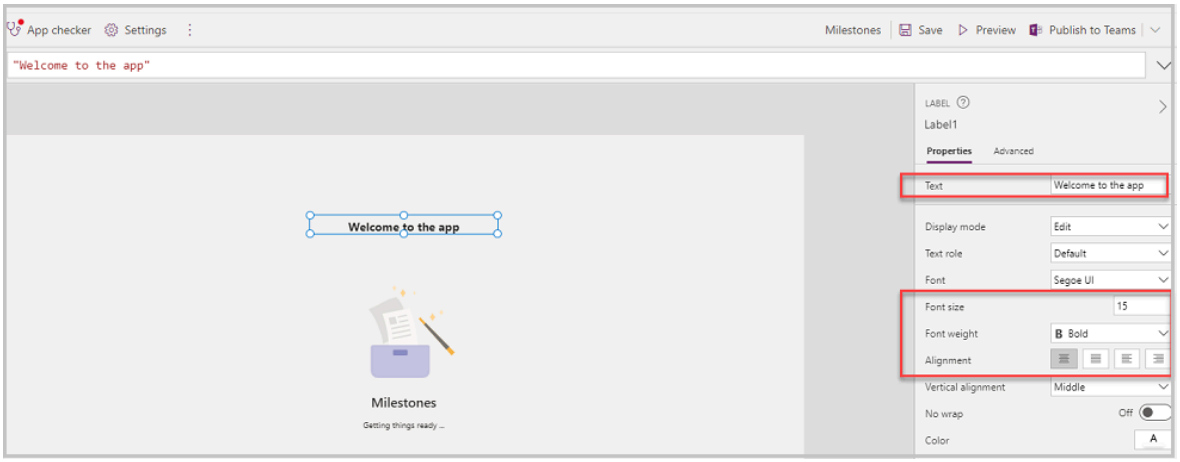
Adding and localizing a new label

As an extension to the default Milestones app, we'll add a new label and learn how to localize it. Let us add a label to the Loading screen:

1. Select **Insert**, and select label control.



2. Update the Text and properties of the label.



For now, we'll set the text to "Welcome to the app" until it's ready for localization.

3. Open the Excel file to add rows for this text.

4. Create eight rows in the Excel file with **OOBTextID** value **lblLoadingText_Welcome_locText**, and **LanguageTag** for the 8 rows:

Expand table

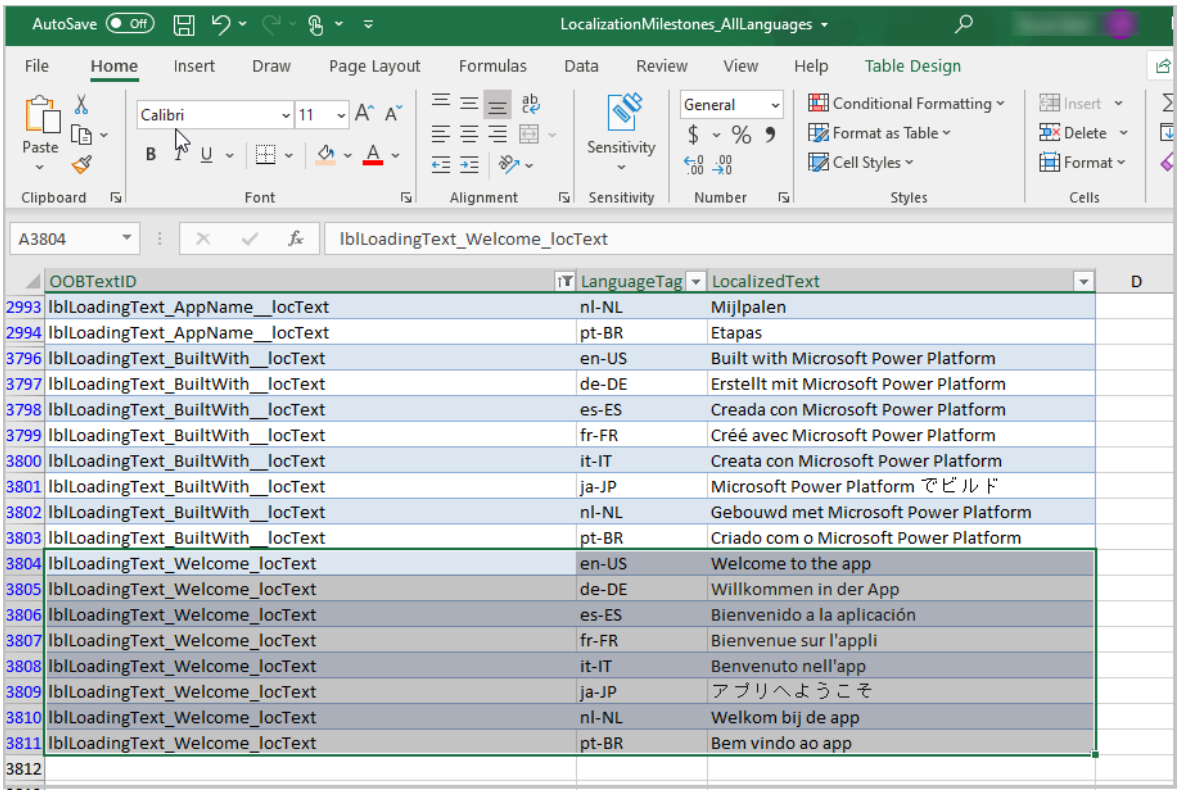
en-US
de-DE
es-ES
fr-FR
it-IT
ja-JP
nl-NL
pt-BR

5. For the third column—**LocalizedText**, translate the text "Welcome to the app" in the 8 different languages:

   - en-US (US English)
   - de-DE (German)
   - es-ES (Spanish)
   - fr-FR (French)
   - it-IT (Italian)
   - ja-JP (Japanese)

- nl-NL (Dutch)
- pt-BR (Portuguese)

6. Update the spreadsheet with all the translated text as shown below.

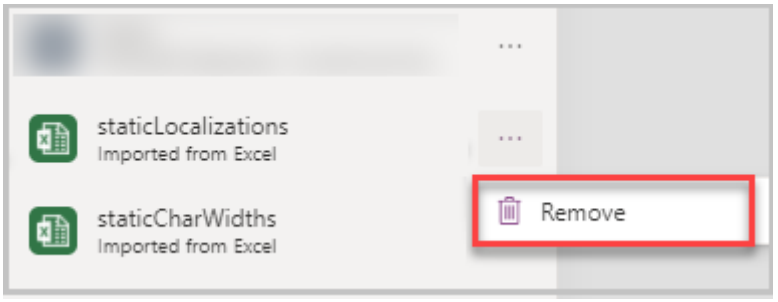


	OOBTextID	LanguageTag	LocalizedText	D
2993	lblLoadingText_AppName__locText	nl-NL	Mijlpalen	
2994	lblLoadingText_AppName__locText	pt-BR	Etapas	
3796	lblLoadingText_BuiltWith__locText	en-US	Built with Microsoft Power Platform	
3797	lblLoadingText_BuiltWith__locText	de-DE	Erstellt mit Microsoft Power Platform	
3798	lblLoadingText_BuiltWith__locText	es-ES	Creada con Microsoft Power Platform	
3799	lblLoadingText_BuiltWith__locText	fr-FR	Créé avec Microsoft Power Platform	
3800	lblLoadingText_BuiltWith__locText	it-IT	Creata con Microsoft Power Platform	
3801	lblLoadingText_BuiltWith__locText	ja-JP	Microsoft Power Platform でビルド	
3802	lblLoadingText_BuiltWith__locText	nl-NL	Gebouwd met Microsoft Power Platform	
3803	lblLoadingText_BuiltWith__locText	pt-BR	Criado com o Microsoft Power Platform	
3804	lblLoadingText_Welcome_locText	en-US	Welcome to the app	
3805	lblLoadingText_Welcome_locText	de-DE	Willkommen in der App	
3806	lblLoadingText_Welcome_locText	es-ES	Bienvenido a la aplicación	
3807	lblLoadingText_Welcome_locText	fr-FR	Bienvenue sur l'appli	
3808	lblLoadingText_Welcome_locText	it-IT	Benvenuto nell'app	
3809	lblLoadingText_Welcome_locText	ja-JP	アプリへようこそ	
3810	lblLoadingText_Welcome_locText	nl-NL	Welkom bij de app	
3811	lblLoadingText_Welcome_locText	pt-BR	Bem vindo ao app	
3812				

7. Save the spreadsheet.

8. Go back to the app editor in Teams.

9. To refresh the Excel data source (since we updated it), select **Data** > **...** (ellipsis), and then select **Remove** next to the **staticLocalizations** Excel spreadsheet.



Removing the table will cause some errors in the app. However, we'll readd the updated Excel file in the next steps.

10. Readd the spreadsheet as a connector, and select **staticLocalizations** table as the data source (make sure to select the correct excel file that was updated).

11. Select the label with the text "Welcome to the app".

12. Open the **Text** property of the label and paste the following formula:

Power Fx

```
With({varDefault: "Welcome to the app", varOOBTextId:
"lblLoadingText_Welcome_locText"}, With({varLocalizedText:
LookUp(colLocalization;OOBTextID = varOOBTextId, LocalizedText)},
Coalesce(varLocalizedText;varDefault)))
```

13. To test the localization of the label we created, change the language of our user in Teams to one of the languages that is in our localization Excel spreadsheet. You can change your Teams language by selecting your photo in the upper-right corner, and then selecting settings.

14. Select Milestones to open the app and the **Welcome to the app** label has been localized to the language that was selected in the previous step. You can verify the same by going to the specific Team channel where the app was installed and playing the published app.

Feedback

Was this page helpful? Yes No

Provide product feedback

Theming in sample app templates

Article • 02/08/2023

The most attractive feature about canvas apps is the ability to customize the user interface per the user or organization's branding and liking. Theming strategy provides us with the ability to create apps that match the organizational brand. A **theme** contains a collection of styles that define multiple design properties for controls and components such as colors, fonts, and borders.

In this article, we'll learn how to use global variables to define themes, and how the colors and properties can be easily changed throughout the app by just updating the global variables.

Watch this video to learn about theming in sample app templates: https://learn-video.azurefd.net/vod/player?id=c8ddd966-a7fc-431b-8b2c-375e85b862ba&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fsample-app-theming

Prerequisites

To complete this lesson, we'd need the ability to create apps within Microsoft Teams that will be available as part of select Microsoft 365 subscriptions. You'll also need the Milestones sample app template for Microsoft Teams installed in your Team. You can install it in your team at https://aka.ms/TeamsMilestones.

Defining the theming strategy

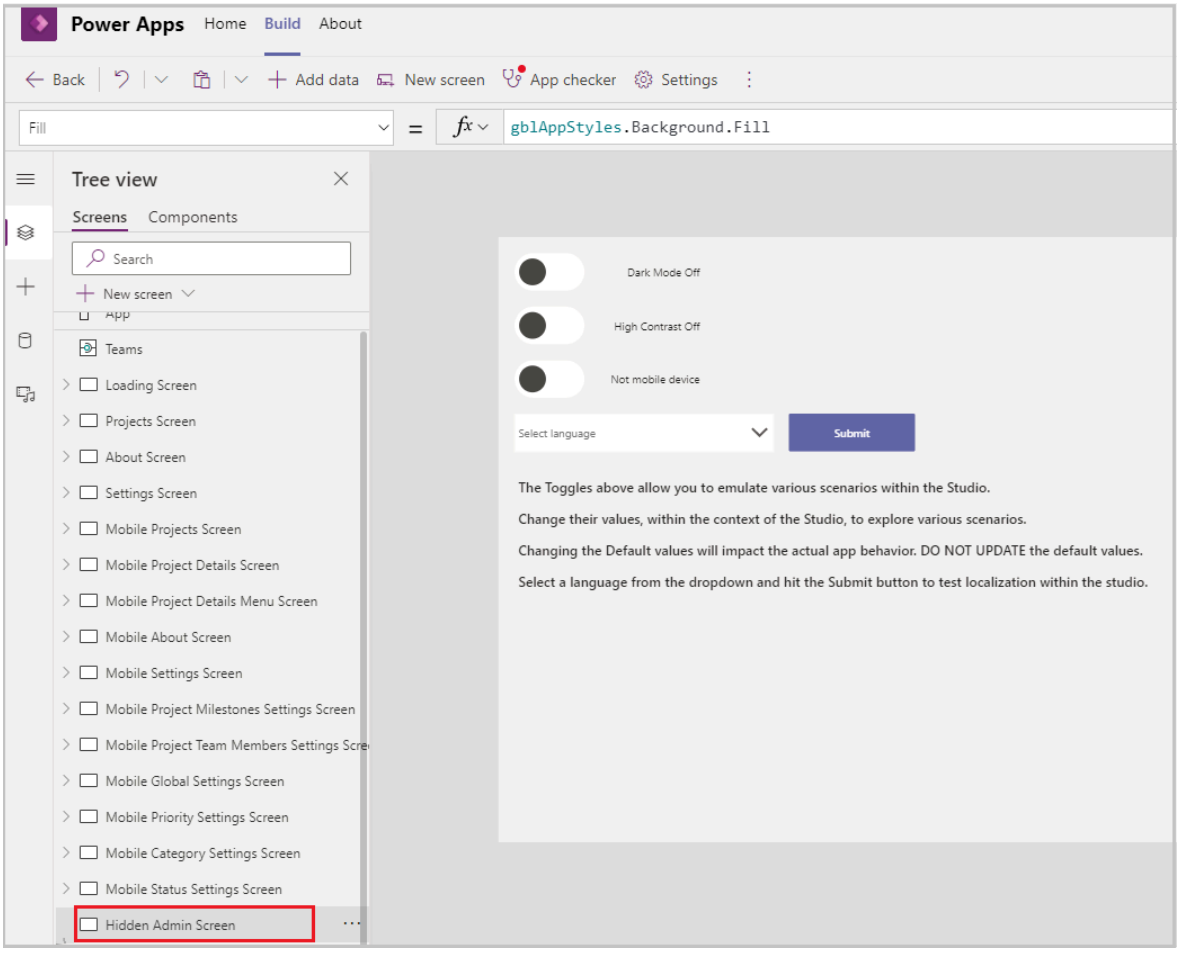
In this example, we'll see how global variables have been used to define colors and styles for theming in the Milestones app. The following global variables have been defined in the **OnVisible** property of the **Loading screen**.

- **gblAppColors** to define colors so they can be referenced easily
- **gblAppSizes** to define different sizes of fonts for mobile and web apps
- **gblAppStyles** properties of every single control using **gblAppColors**
- **gblThemeDark** set to true when the Teams theme is dark so all controls adapt to the dark theme
- **gblThemeHiCo** set to true when the Teams theme is high contrast so all controls adapt to the high contrast high contrast

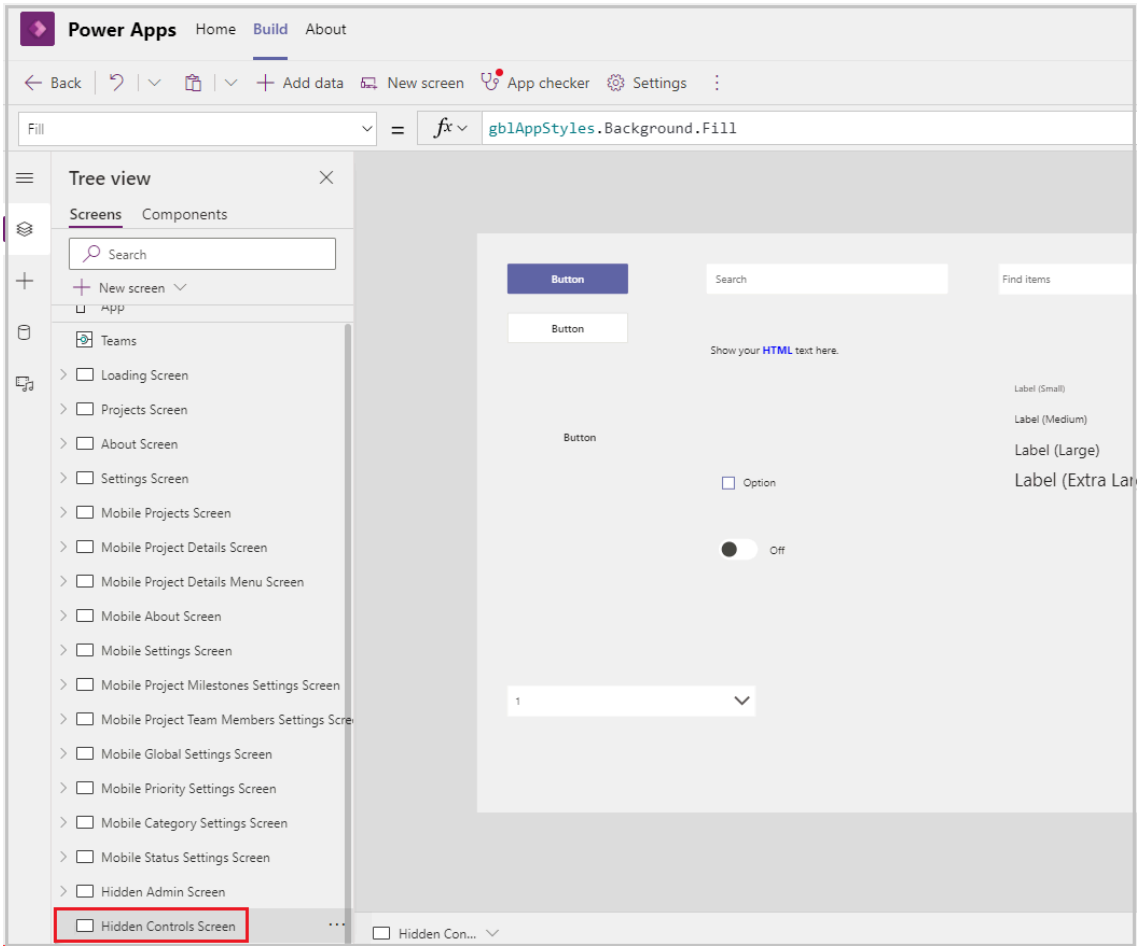
Testing the theming Strategy

Two hidden screens have been defined for testing the theming strategy:

- Hidden Admin Screen – This screen has two toggles defined **tglAdmin_DarkMode** and **tglAdmin_ContrastMode**. This screen helps with quick testing of dark and contrast modes without having to change the theme of teams for testing



- Hidden Controls Screen – This screen has every single control with properties set using gblAppStyles.
    - Helps in testing out theme changes.
    - Helps in copying a particular control and pasting it on the required screen so that the formula for the control doesn't need to be typed repeatedly.



Using the theming variables for a sample app templates

1. To understand the usage of theming in the Milestones app, select **Projects Screen** in the Tree view to open the screen and then select **+New Project**.

2. Select the **Color** property of **+New Project**.

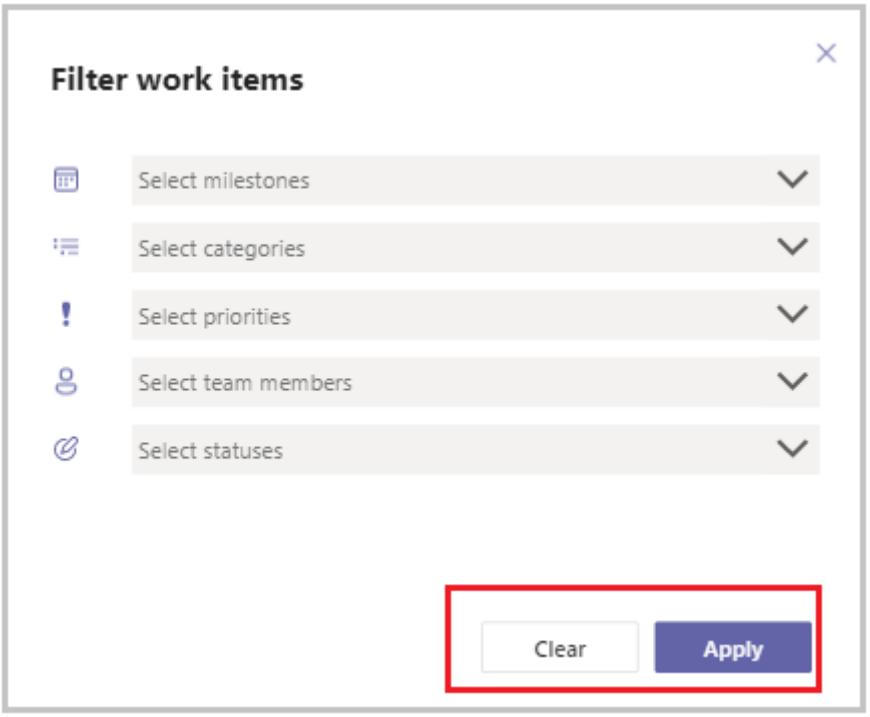
   Notice that the formula references **gblAppStyles** > **ButtonTransparent** > **Color** formula from the **OnVisible** property of the Loading screen.

   So now, if the theme of the app is High Contrast, then the color defined in **gblAppColors** > **HighContrastHyperlinks** formula will be used.

3. Another example would be to look at the Primary and Secondary buttons.

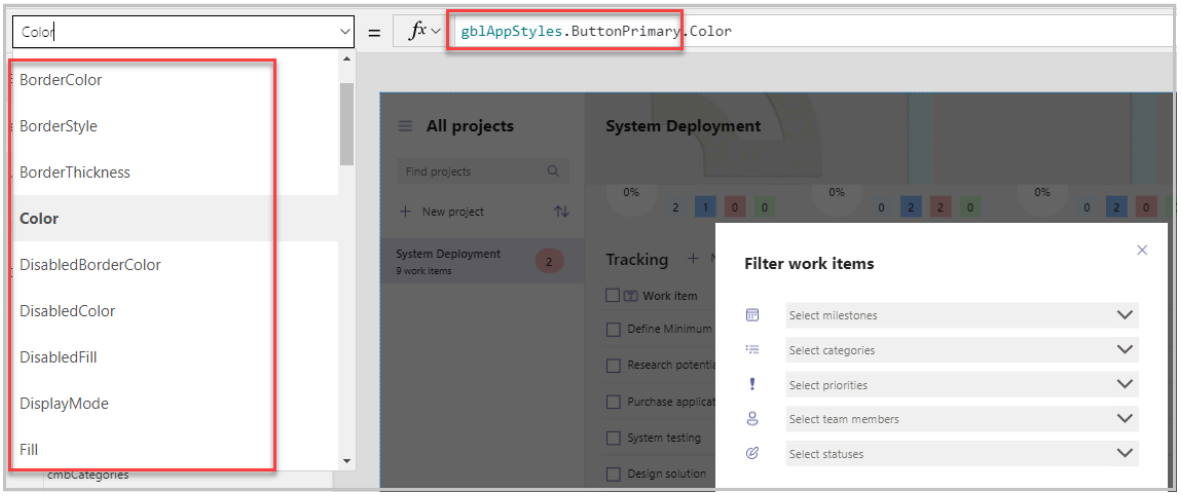
   - Select the Project Screen in the editor.
   - Hold down the Alt key and select the Filter icon on the Projects Screen.

     The Filter work items pop-up opens up. Notice the two buttons: **Clear** and **Apply**.



Apply is an example of a primary button and **Clear** is an example of a secondary button.

4. Select the **Apply** button, and select any Border/Thickness/Color related property and notice the formula defined.



5. The **Apply** button is defined as the Primary button and uses all the properties defined on **Loading Screen** > **OnVisible**> **ButtonPrimary** formula.

   Depending on the theme and color variables selected in the ButtonPrimary formula, the styling properties of the button will get updated.

   Similarly, **Clear** is defined as a Secondary button and uses all properties defined under **gblAppStyles** > **ButtonSecondary**.

> ⓘ Note

Another possible extension to the app would be to add Font as a Property, either as one variable to be used across all controls, or once per control (like how the other styling properties are defined separately for primary buttons, or secondary buttons). The font throughout the Milestones app is Segoe UI and there was no need identified for theming of the font.

Updating the theme in the sample app template

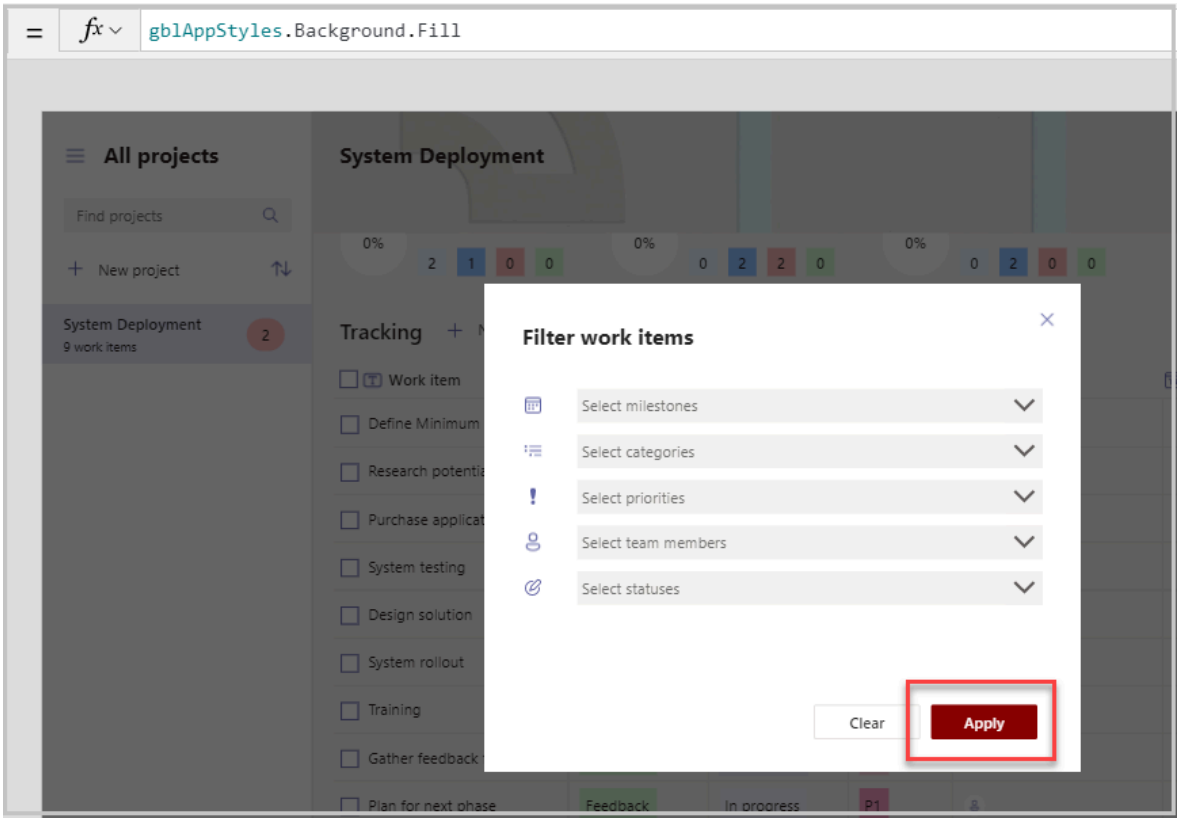
In this section, we'll see how we can change a property of a particular control and how the control gets updated across various screens. In our example, we'll update the Fill property of the Primary button and see how the button color changes in the Dark mode, and the Default mode.

1. Go to the **Loading Screen** and select the OnVisible property in the dropdown.
2. Expand the formula bar and scroll down to the **ButtonPrimary** > **Fill** function.
3. Update the last two parameters to Red and DarkRed.

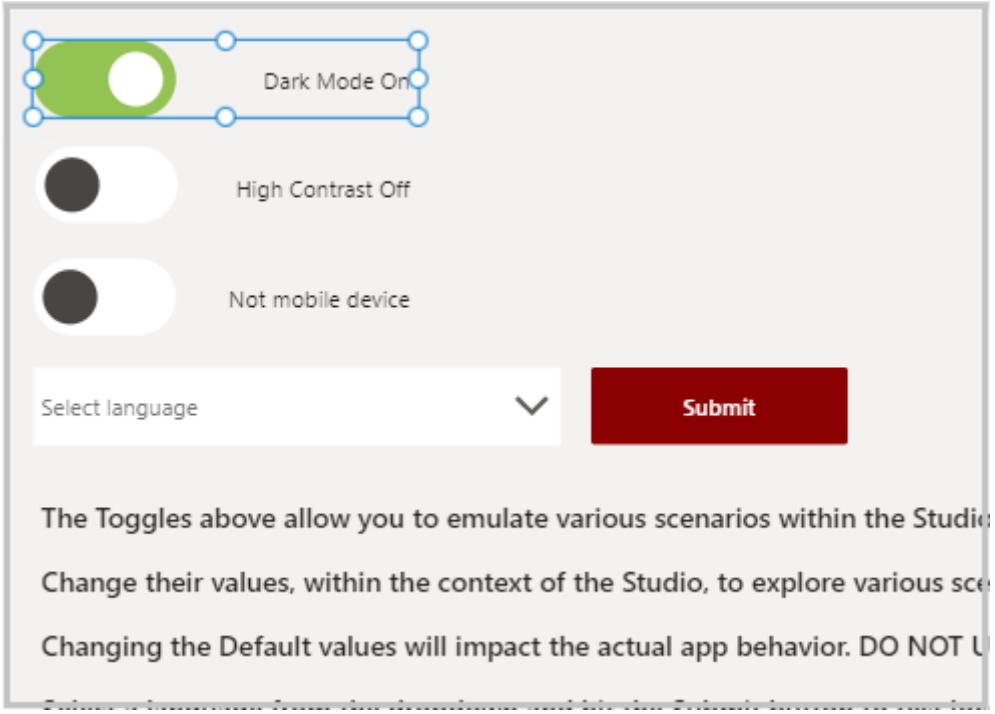


4. Go to another screen and then back to Loading Screen for the **OnVisible** function to run.

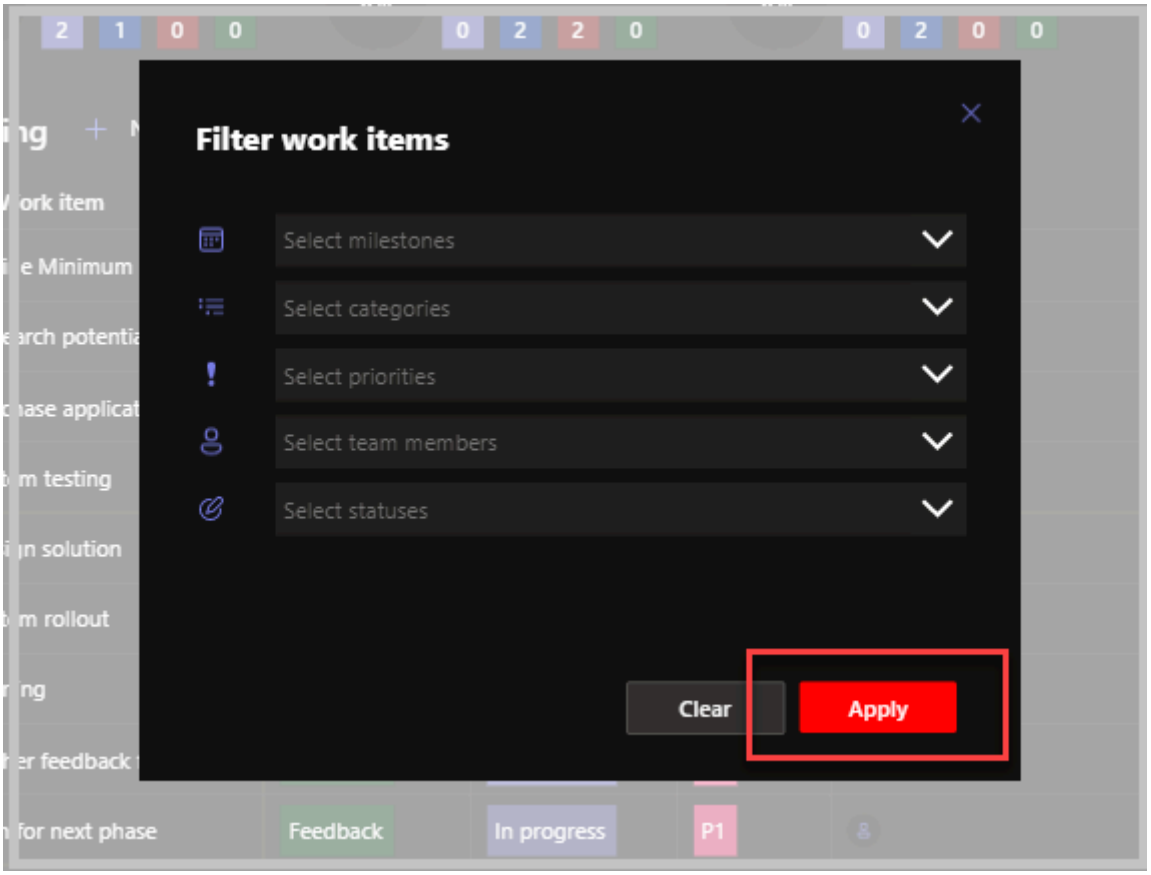
5. Go to the **Projects** screen, select the **Filter** button to open the Filter pop-up and notice the color of the Apply button is now DarkRed.



6. Go to the **Hidden Admin Screen**, hold **Alt** key on the keyboard, and select the **Dark Mode On** toggle.



7. Go to the Loading Screen again for the function to load and then come back to the Projects screen. Notice the screen is now in dark mode (with the background dark), and the Apply button color has now changed to **Red**.



8. Similarly, changing properties of the **SecondaryButton** would have changed colors for the secondary button, such as the **Clear** button.

> **Note**
>
> Updating the fill function directly is a quick way of testing theming as shown earlier. If the color needs to be changed for not just the Primary Button control but for all other controls where it is being used as well, then the color palette under the variable gblAppColors should be updated with the formula of the desired colors and names that makes the most sense to your organization. For example, `companynameDefaultHeader: ColorValue("#30D5C8")` would be the formula for turquoise blue.

Feedback

Was this page helpful? Yes No

Provide product feedback

Granting permission to tables in Dataverse for Microsoft Teams

Article • 01/20/2023

When building Power Apps, one important consideration is data security. Your apps drive important business processes, and if users can see sensitive data or delete records that they shouldn't have access to, it can disrupt your business.

For example, if you've an app with sensitive customer data, unauthorized access to the data could put your company at risk of compliance with privacy laws.

You may think "I'll just obscure access to sensitive data in my app!" However, if the underlying data source is shared with users, they can still get to the data, or build new apps on the data source, giving them access to the restricted records.

One of the advantages of building Power Apps in Teams with Dataverse for Teams is that it provides the ability to set the table permissions to your specific business needs that map the security roles in Teams—such as Owners, Members, Guests, and App Users.

For example, if there's a table that the users should only be able to modify for the records that they own, Dataverse for Teams offers a table permission called **Collaborate** that gives users the ability to only edit records that they own. In this article, we'll review the table permissions in Dataverse for Teams, and learn about how you can use them to secure your important business applications.

> ⓘ **Note**
>
> Only team owners have the ability to edit table permissions currently.

Watch this video to learn how to configure table permissions:
https://learn-video.azurefd.net/vod/player?id=23f33cc0-7db3-40fc-85e9-d6c3057b2489&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fdataverse-for-teams-table-permissions

Understanding team roles

Permission to apps and the data that they use in Teams are based on the following roles in Teams team.

- **Owner** - A team member with the owner role. This includes the creator of the team, and anyone who has been given the owner role by another users. Owners get full access to the data used in apps in the team.
- **Member** - A team member who isn't an owner. By default, team members get full access like owners, but you can also change the permission for team members. You might want to restrict members from being able to edit certain tables.
- **Guest** - A guest is someone outside of your organization who is added to the team. By default, guests can create records, but not view records created by others.
- **Colleagues with access**: With Power Apps for Teams, you can share an app with Azure AD security group whose members need not be part of the Teams team where the app was built. This enables you to add users to the application without having to add them to the specific team, and opens up “Broad Distribution” scenarios. For example, you may want to build an app that is enabled for every accountant in the organization, or even every employee in that organization.

Understanding table permissions

When assigning security to a table, the following are the permissions that you can assign:

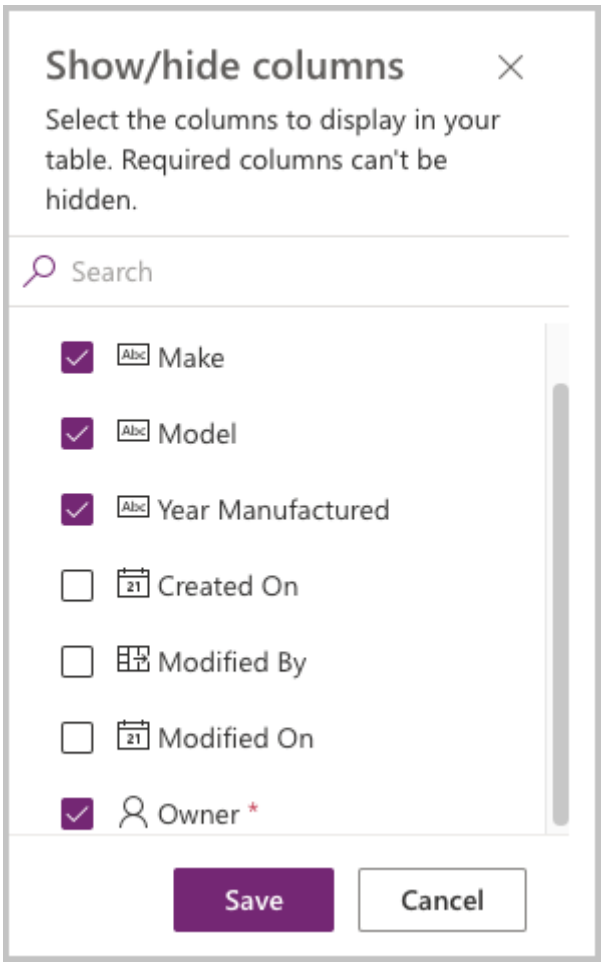
- **Full Access** – Allows end users to see and edit all records in the table.
- **Collaborate** – Allows end users to see all records and create new records. But they can only edit or delete their own records. An example for when you want to use collaborate is "up votes" in the Ideas the Perspectives app. You want each user of the app to view the up-votes of other users. But they shouldn't be able to change the votes for any users other than their own.
- **Reference** – Provides a read-only view of data. Users can't create, edit, or delete any records. An example is in Bulletins app—if you want users to read bulletin records but not edit them, you'd give them **Reference** permission on the **Bulletin** table.
- **Private** – Users can create new records. Additionally, they can view, edit, or delete their own records only. An example is in the Issue reporting app—if you want users to only be able to create and see their own issue records in Dataverse, you'd give them **Private** permission on the **Issue Report** table.

Table permissions in real-world scenario

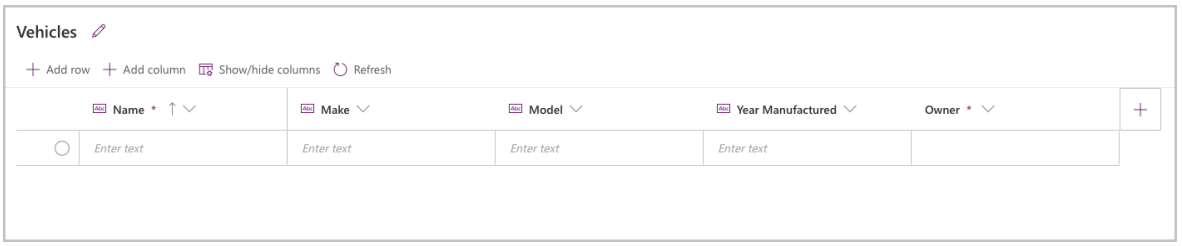
Let’s take a look at how table permissions work in practice. We'll create a table, and then show what two different users see.

> **Note**
>
> To complete this part of the lesson, you'll need an environment with two user logins. One of them having the team owner role and another with team member (a member of the team but not granted owner role)

1. Sign in to Teams as the Team owner.
2. Create an app and the table in team with a Dataverse for Teams database.
3. Right-click on the Power Apps icon and pop out the app.
4. Create a new app.
   a. App name: "Vehicle Manager"
   b. Select **Create new table** and name it "Vehicle".
   c. Add columns: "Make", "Model", "Year Manufactured".
   d. Select **Show/hide columns**.
   e. From Show/hide column dialog, select the **Owner** field and select **Save**.



Your table should now look like this:



5. Add a record to the table:
   a. Select **Add row**.
   b. Enter record details.
   c. The owner field of the created record will reflect your logged-in user account.

6. Select the area outside of the table definition screen to return to the app.

7. Save and Publish the app. You'll need to select a channel in which the app will be displayed—select + next to the desired channel name to select the channel.

Configure and verify table permissions

We're going to set the permissions for the vehicles table for Team members:

1. In Teams, open the **Power Apps** app. Right-click on the Power Apps icon in Teams, and select **Pop out app**. This will pop out the app in a new window, so if you move somewhere else in Teams, you won't lose your changes.

2. Select the **Build** tab.

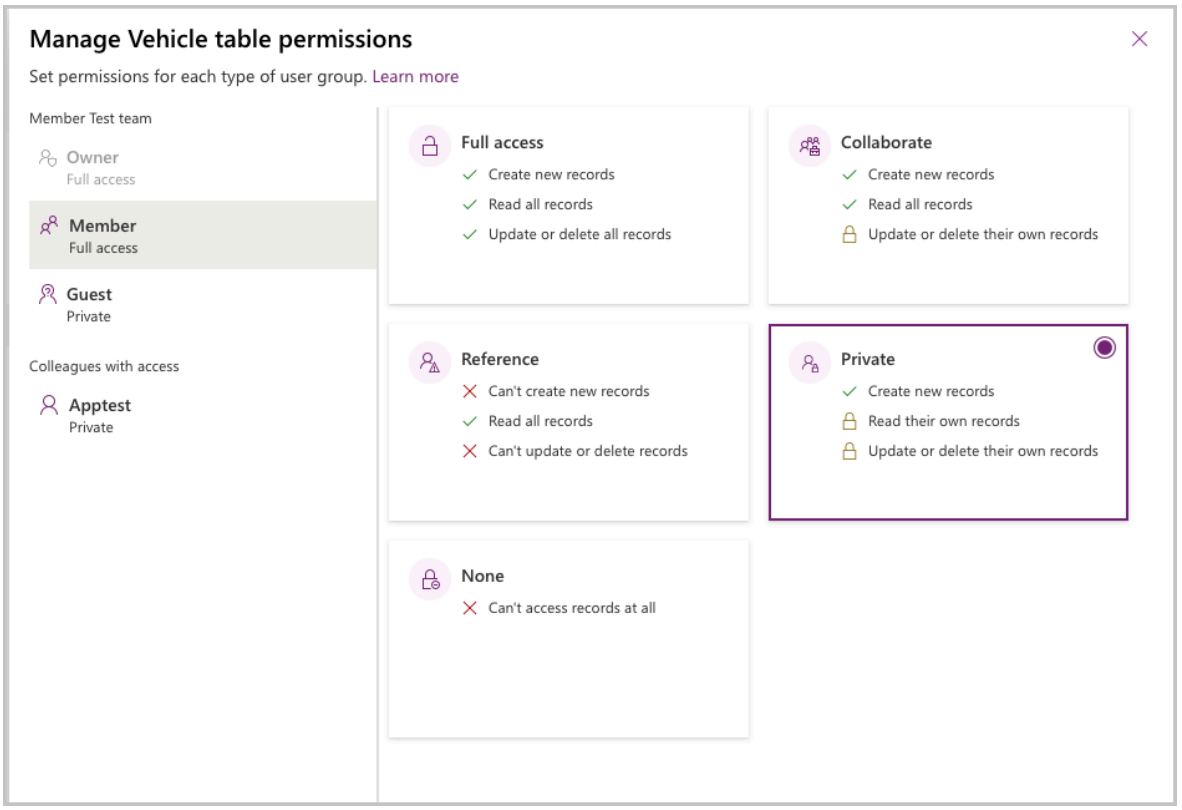
3. Select the team that contains the Vehicle Manager app.

4. In the **Items created for...** area, select **See all**.

5. Select **Tables** from the solution components bar.

6. Select the Vehicle table and then select **Manage permissions**.

7. Select the **Member** node. The initial permission will show Full access. Select the permission **Private** and select **Save**.



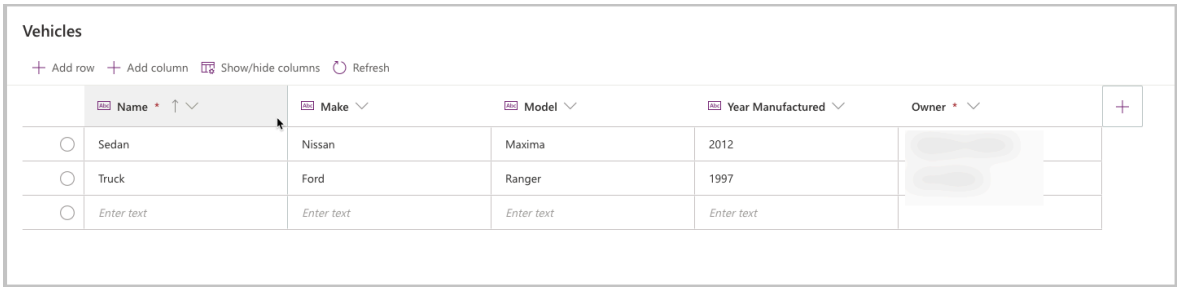
Next, we'll log in to Teams as a user who is a non-owner Team member to observe their experience.

1. Log into Teams using a user who is a member of the team in which the app is installed.
2. Select the channel where the app is installed.
3. Select the tab for the Vehicle Manager app.
4. The app should open for the user, but no records will show in the gallery, as the user only has visibility to their own records.
5. Select **New record** to create a new vehicle record, select the check mark to save the record.
6. You should now see one record in the gallery.

Log out of Teams and log back in as the Team owner. We'll now go to the table and see the created data.

1. Log in to Teams using the Team owner personal.
2. Right-click on the Power Apps icon and pop out the app.
3. Select the **Build** tab.
4. Select the team in which the Vehicle Manager app is installed
5. Select the Vehicle table
6. Select **Edit**.

7. You'll now see two records in the Vehicle table, each with a different owner.



So how would this apply to one of the sample app templates for Teams?

For example, with Bulletins we might set the following permissions for the tables in the app:

Expand table

Table	Permission	Reason
Bulletin	Reference	Users should read all bulletins, but not edit or create bulletins.
Bulletin Bookmark	Private	Users should create their own bulletins, but not view or edit other users' bookmarks.
Bulletin Category	Reference	Users should read all categories but not edit or create categories.
Bulletin Category Preference	Private	Users should create their own category preferences but shouldn't be able to read or edit other users category preferences.
Bulletin Contact	Reference	Users should read all contact records but not create or edit contacts.
Bulletin FAQ	Reference	Users should read all FAQ records but not create or edit existing FAQ records.
Bulletin FAQ Category	Reference	Users should read all FAQ categories but not create or edit existing FAQ records
Bulletin Link	Reference	Users should read all links but not create or edit existing link records.
Bulletin Link Category	Reference	Users should read all link categories but not create or edit existing link categories.
Bulletin Read Receipts	Private	Users should be able to create new read receipts when they view a bulletin, but not read or edit other users read receipts.
Bulletin User Settings	Private	Users should be able to create new user setting records but not read or edit other users setting records.

What permissions should I share?

Before assigning permissions, you should plan your security based on the personas of users of your app. Who will administer the app? Who will use the app? How will they use it?

For example, the Inspection sample app template for Teams is designed to address the following personas:

- **Inspection manager** - Creates Inspection forms and manages the list of locations or assets.
- **Inspection reviewer** - Reviews inspection data and approves inspections.
- **Inspection User** - Completes the inspections.

Once we have the personas defined, relate these personas to the team role to which they should be assigned:

- **Inspection manager** - Since inspection managers will need full access to create inspection forms and other system table records, we'll make them owners in the team.
- **Inspection reviewer** - Since inspection reviewers should be able to edit all inspections, but they won't create inspection forms, we'll make them team members. This will enable us to give them reference access to the Area **Inspection Checklist** table (where inspection form data is stored), but full access to **Area Inspection** table (where inspection records are stored).
- **Inspection user** - Inspection users should only be able to submit new inspections, but not create new inspection forms or see other user's inspections. So we won't make them members of the team in which the app is installed—we'll share the app with them as colleagues, and give them **Reference** permission to **Area Inspection Checklist** and **Private** permission to **Area Inspection**. This will allow users to view inspection forms and create their own inspection records.

For more details about the recommended table permissions for the sample app templates for Teams, see articles about deploying sample app templates broadly, such as Deploy Profile + as broad distribution app.

Feedback

Was this page helpful? Yes No

Provide product feedback

Frequently Asked Questions (FAQs) for sample app templates

Article • 02/08/2023

In this article, learn about the frequently asked questions and the answers about using the sample app templates.

> **Important**
>
> For any issues, feedback, support, or changes related to sample app templates, go to **https://github.com/microsoft/teams-powerapps-app-templates/issues** and submit an issue or a pull request.

When I try to install a sample app template, I get an error that the app is already in the team.

You may see the following error when you try to install a sample app template in a team that already has the app installed, but you're not the app owner, or the team owner.

```
This app is already in this team
To update it, you must be its owner or a team owner.
```

When this happens, use the installed app, or install the app in a different team.

How can I move an app to another channel in the same Team?

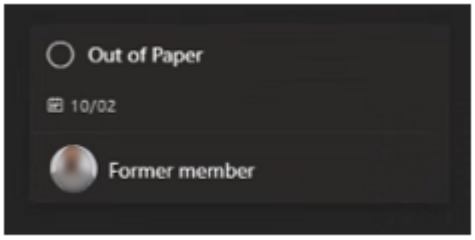
For the steps to move an app, go to Move an app to another Teams channel in the same team.

How do I delete the sample data from the sample app templates?

For the steps to delete sample data from the sample app templates, go to Remove sample data.

Planner tasks are created with “Former Member” instead of the name of the assigned user

When you assign a task in the Issue reporting, or Inspection apps who isn't a member of the Teams team, the user's name on the task will show as "Former member". To correctly display the user's name, add the user to the Team where the app is installed.



Why can't I create tasks in Inspection app?

If a user isn't allowed to create tasks using a mobile device in the Inspection app:

1. Open **Manage inspections** in Power Apps Studio.
2. Select **Settings**.
3. In the Planner drop-down, select the Planner that you want to use.
4. **Save**

Inspection app forms are slow

If your inspection forms are loading slowly, reduce the size of images used in the inspection steps. Users can't interact with the form before all images on the form are loaded.

Known issues and limitations

09/03/2025

This article provides details about the known issues and limitations when using **Dataverse for Teams environments**.

Additional components

The following components aren't supported:

- Model-driven apps
- AI Builder

> **Note**
>
> Custom connectors are supported only if both of these conditions are met:
>
> - The app or flow is running in the context of Microsoft Teams.
> - And, the custom connector points at an APIM endpoint in Azure.

Controls

To learn about the limitations of controls in Dataverse for Teams environments, go to Limitations of controls in Power Apps.

Region availability

Dataverse for Teams currently isn't available in:

- Power Apps operated by 21Vianet in China

Hidden membership groups

Dataverse for Teams doesn't support hidden membership groups. You'll receive the following error when you try to create an app in a team that has the group visibility set to *Hiddenmembership*.

```
Existing limits on this team
```

```
You're unable to continue due to an administrative setting on this team. You can try
```

`again with another team.`

The group may have hidden membership enabled by your administrator. Try using Dataverse for Teams on a new Teams team, an existing team that doesn't have hidden membership enabled, or contact your administrator.

Localization

Localization of Power Apps Studio isn't supported.

Power Apps component framework

Power Apps component framework isn't supported.

Required fields

There's no enforcement of user-created required table fields. Rows that have empty values in those fields can be saved successfully. System required fields are enforced and cannot be saved without a value.

Studio

Classic controls

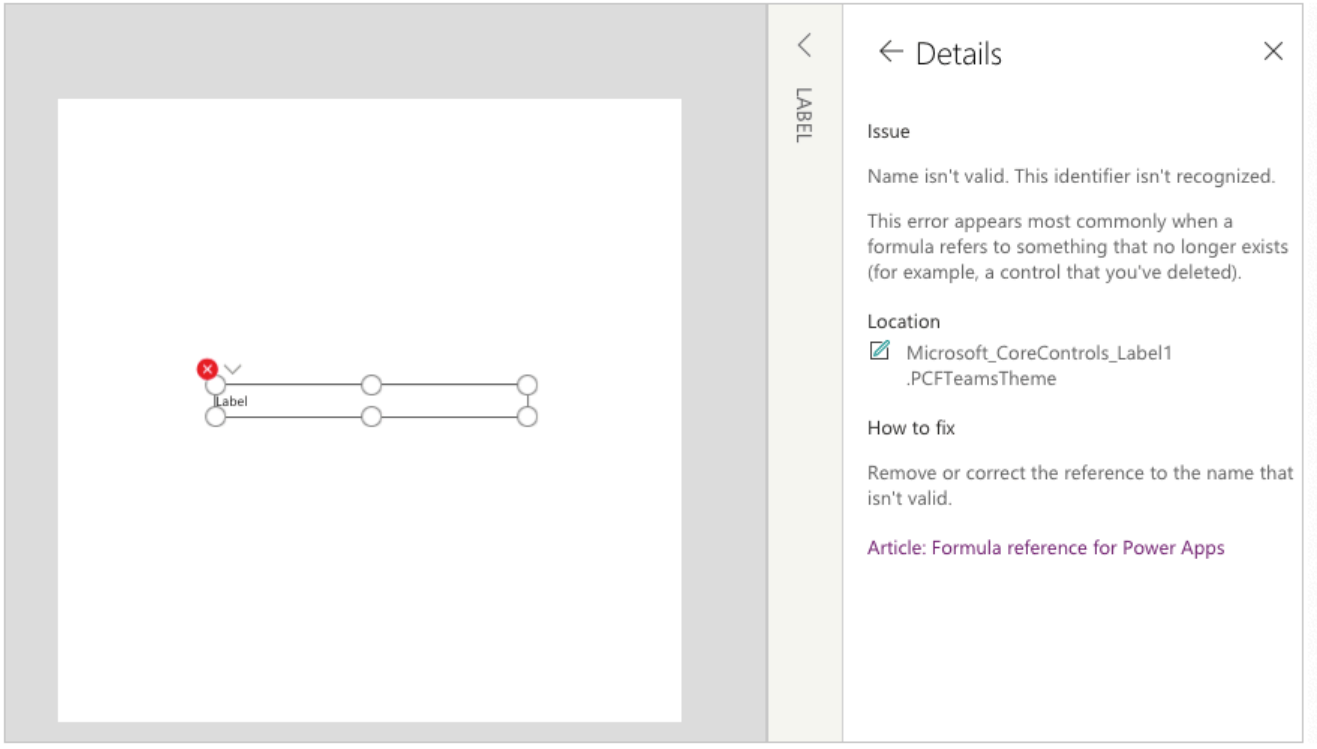
Enabling classic controls requires a refresh of Power Apps Studio.

Fluent UI date picker control

The Fluent UI date picker control doesn't yet support format string. If you need to support special formatting, you'll need to enable classic controls and use the classic Date Picker.

Canvas components

You may see red errors while using the modern controls in canvas components. These errors won't functionally impact your app, and you can ignore them.



New connections

Connections in the Power Apps Studio that require an authentication dialog fails in the Teams desktop client. Open the Studio in the Teams web client to add these connectors.

Studio and visual editor synchronization

Power Apps Studio doesn't reflect the changes to tables after:

1. Renaming tables.
2. Adding relationships within the inline visual editor.

To see the new changes, refresh Power Apps Studio.

Studio version

It's currently not possible to change your Studio version.

Visual editor

Currency, Duration, Language, Ticker, and Timezone fields can't be added using visual editor. To create these fields, use the experience to work with tables as available using solution explorer.

Load time limit

Apps embedded in Teams are required to load within 30 seconds as per Teams requirements. If an app exceeds this 30 seconds limit, Teams assumes that the app has timed-out, and will display an error screen. If your app is encountering this 30 second timeout, review tips to improve performance for boosting your app's performance.

Theme

Power Apps Studio and apps currently don't support **Dark theme**.

Others

- If a user is an owner of the Azure Active Directory (Azure AD) group associated with a team but isn't also a member of that group, they may not be able to see that team in the Power Apps and Microsoft Copilot Studio apps.
- It may take up to 2 hours for deleting, renaming, or restoring a team to reflect correctly within the Power Apps and Microsoft Copilot Studio apps.
- It may take up to 15 minutes for new team users to be able to see the team within the Power Apps and Microsoft Copilot Studio apps.
- Users with the **Teams Guest** role con't trigger flows embedded in canvas apps.

Migrate Microsoft Access data to Microsoft Dataverse for Teams (preview)

Article • 02/15/2022

[This topic is pre-release documentation and is subject to change.]

Microsoft Access users can now choose to migrate their data into Dataverse or Dataverse for Teams to make use of Azure Cloud security and Power Platform functionality. Migrating Access data to Dataverse provides many new opportunities for interaction and management of data.

Watch this short video about migrating Access data to Dataverse for Teams.
https://learn-video.azurefd.net/vod/player?id=de255c2b-7a20-47a7-a7fe-f0cbdf19ee99&locale=en-us&embedUrl=%2Fpower-apps%2Fteams%2Fmigrate-access-to-dvft

Next steps

- Migrate Microsoft Access data to Microsoft Dataverse (preview)
- Data types and sizes for Access data migration to Dataverse (preview)
- System columns and tables in Dataverse and Dataverse for Teams (preview)

Feedback

Was this page helpful? Yes No

Provide product feedback

Important upcoming changes (deprecations) in canvas apps

The announcements for changes (deprecations) described in this article apply to canvas apps.

Makers, developers, and IT professionals can use this information to prepare for future releases.

> **Important**
>
> "Deprecated" means we intend to remove the feature or capability from a future major release. The feature or capability will continue to work and is fully supported until it is officially removed. This deprecation notification can span a few months or years. After removal, the feature or capability no longer work. This notice is to allow you sufficient time to plan and update your code before the feature or capability is removed.

Enhanced deep link creation requirement

Starting May 1, 2026, for canvas apps, all deep links for Power Apps mobile must include the environment ID as a required parameter. Existing deep links without these parameters will stop working after the deprecation date. You also need to create new shortcuts. Update your links now to avoid disruptions. Old links have slower performance until support ends. Learn more in Use deep links with the Power Apps mobile app.

Deprecated endpoints

For information about deprecated endpoints, see Deprecated endpoints

New addition for restrictive firewall rules to filter subdomains

Effective February 14, 2022, if you're using restrictive firewall rules to filter subdomains instead of `*.powerapps.com`, you'll also need to allow access to `*gateway.prod.island.powerapps.com` for a continued Power Apps authoring and runtime experience. To see the complete list of the subdomains, go to Required services.

Rules feature in canvas apps is deprecated

Effective October 14, 2019, the rules feature in canvas apps in Power Apps is deprecated. Few people have made use of rules in their canvas apps. From feedback collected through research and discussions with makers of Power Apps, the rules feature was confusing, and expressions were easier to learn, use, and debug. For more information about the rules feature deprecation, see Blog: Canvas Rules feature deprecation .

Map component fields using dropdown option

The capability to map component fields using dropdown option available through the **Advanced** tab for the selected component that's added to an app has been deprecated. To learn more about mapping component fields, alternatives, and this deprecation, go to Map input fields of a component.

See also

Important changes (deprecations) coming in Power Apps, Power Automate, and customer engagement apps

---

Last updated on 01/09/2026

Power Apps model-driven apps documentation

Create powerful apps with easy to use tools that run across the spectrum of devices.

Get started

OVERVIEW

What are model-driven apps?

Reasons for building model-driven apps

Meet the app editors

Licensing guide

Create

QUICKSTART

Create a simple app

Steps to building a model-driven app

Building a modern app

App development

CONCEPT

Model-driven app components

Data modelling

Tables

Forms

Views

Relationships

Advanced modeling

HOW-TO GUIDE

Business rules

Business process flows

Embedding canvas apps

Advanced app making and customization

Apply data visualizations

HOW-TO GUIDE

Reporting overview

Create a system chart for a table

Create or edit dashboards

Work with Power BI

Distribute, share, and use

DEPLOY

Use a model-driven app

Share an app

Distribute an app

Use solutions for ALM

Developer zone

HOW-TO GUIDE

Developer guide

Customize table forms using code

Apply data visualizations using code

Best practice

HOW-TO GUIDE

Planning a Power Apps project

Performance monitoring

Designing productive forms

Optimize forms for performance

Resources

DEPLOY

Glossary of terms

Share an app

Learn module

CONCEPT

Environments

Solutions

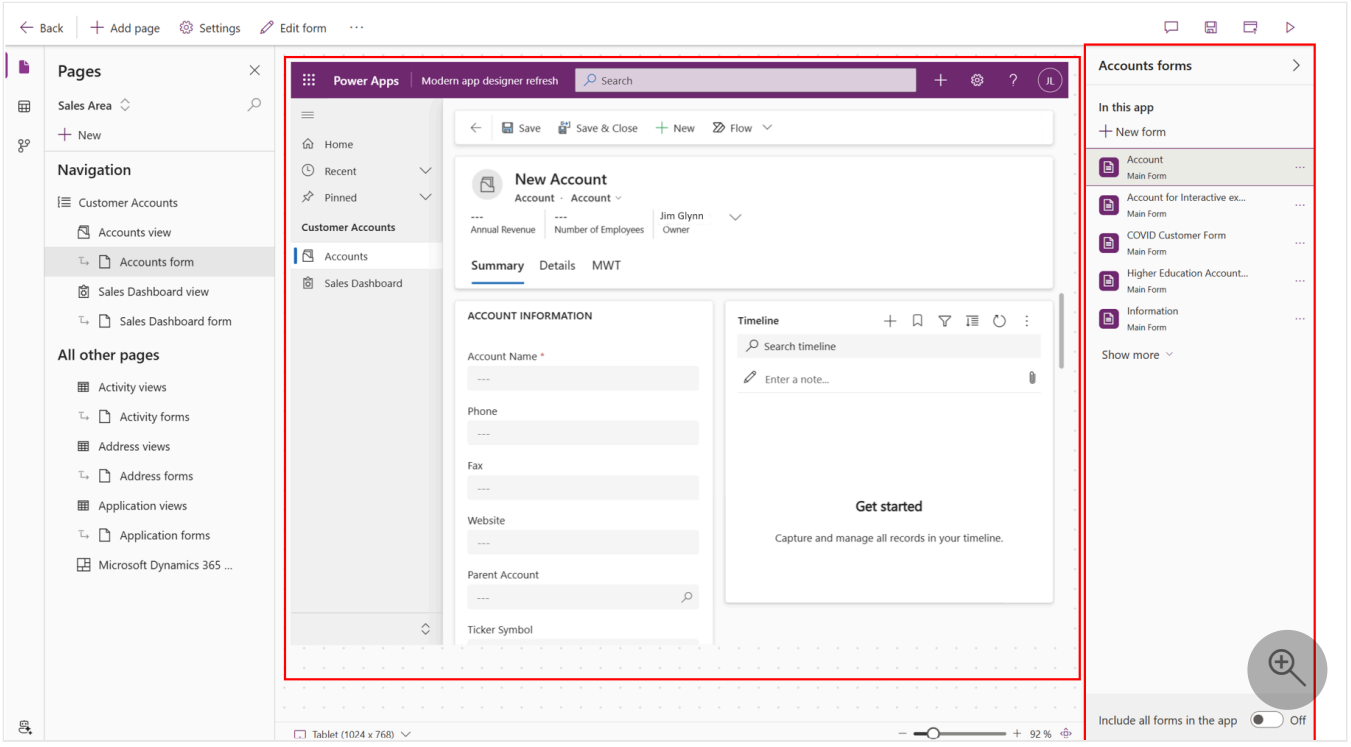
Dataverse

What are model-driven apps in Power Apps?

Model-driven apps overview

Model-driven app design is an approach that focuses on adding components such as forms, views, charts, and dashboards to tables using an app designer tool. Additionally, relationships connect tables together in a way that permits navigation between them and ensures that data isn't repeated unnecessarily.

Using the app designer with little or no code, you can build apps that are simple or complex.



Process driven apps

Model-driven apps are especially well suited to process driven apps that are data dense and make it easy for users to move between related records. For example, if you're building an app to manage a complex process, such as onboarding new employees, managing a sales process, or member relationships in an organization such as a bank, a model-driven app is a great choice.

Data modeling

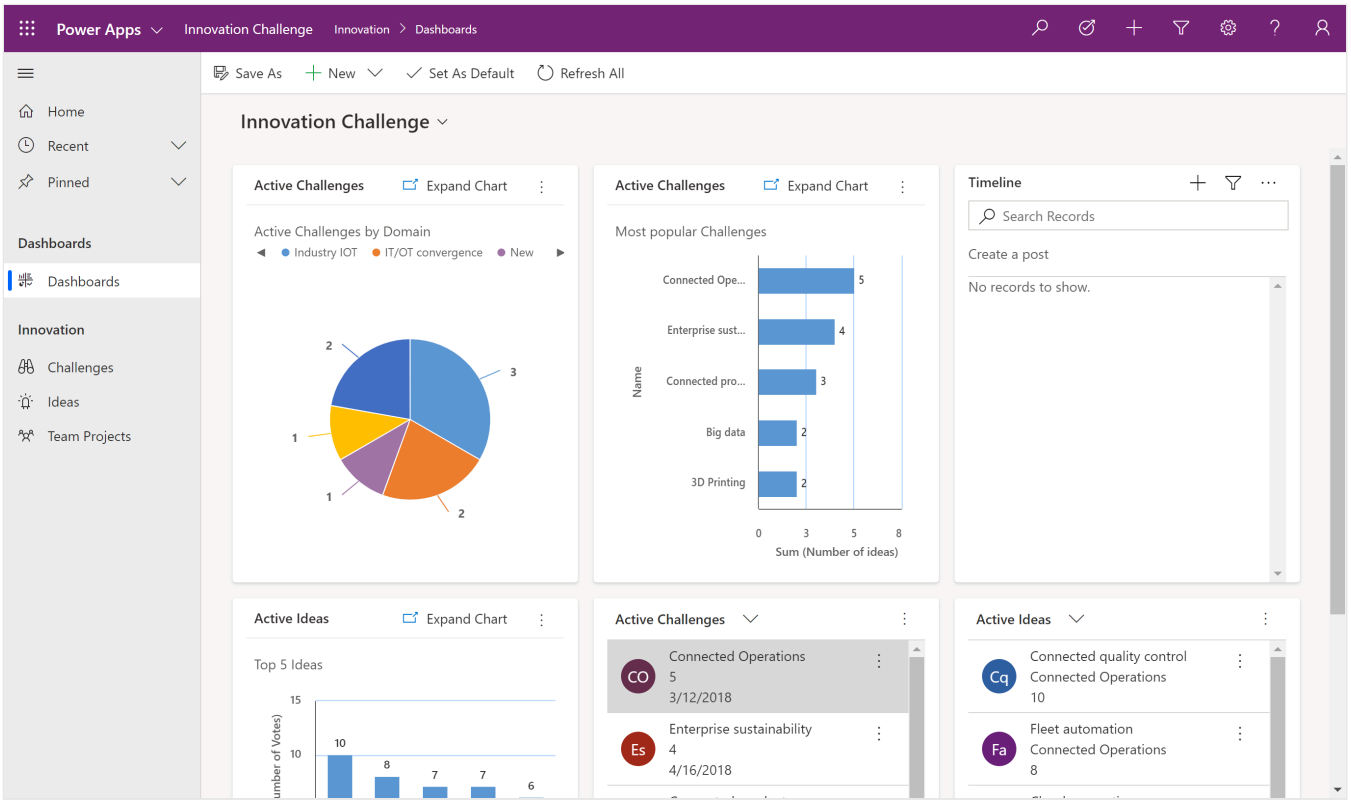
Although we call them *model-driven apps*, it's often easier to think of them as data model driven apps. This is because, without a data model housed within Microsoft Dataverse, you

can't create a model-driven app.

User experience

From the user's perspective, all model-driven apps offer a similar experience, which is both accessible to many users and to the device used. The experience is similar to the diagram shown here.

In this example, the app contains three tables (challenges, ideas, team projects), one dashboard, and multiple charts and views. Users navigate between the tables using the left pane or via the dashboard.



Benefits of the model-driven approach

Unlike canvas app development where the designer has complete control over app layout, with model-driven apps much of the user interface is determined for you and is largely designated by the components you add to the app.

There are some notable advantages to this method of application development.

- Once the data model and relationships are created, the build process is relatively rapid due to rich component-focused no-code designers.
- Apps have a similar user interface across various devices from desktop to mobile.
- The apps are **accessible** and **responsive** automatically.

- The user experience is consistent across all model-driven apps. Once a user is confident with one model-driven app, later apps are easier to adopt within an organization.
- Migrating apps between development, test, and production environments is relatively straightforward by using solutions.

Learn more about the benefits of the model-driven approach

Model-driven and canvas apps compared

In canvas apps, the app maker has total control over the app layout. With model-driven apps, much of the layout is determined by the components you add. The emphasis is more on quickly viewing your business data and making decisions instead of on intricate app design.

Expand table

Category	Model-driven apps	Canvas apps
Data platform	Dataverse only	Dataverse + many others using connectors
Design experience	No-code component focused design	Manipulation of control properties using Power Fx expressions
UI control	Limited, predominantly customization	Full control
App consistency	High – differs predominantly based on the tables and views chosen	Often low, given the significant control the designers have of the user experience
Migration between environments	Simple	Potentially complex given that the datasources might need to be updated
Speed of creation	Rapid	Relative to the complexity of the design
Responsive	Automatically responsive	Only responsive if designed in this way
Navigation through relationships	Automatic, provided relationships exist	Only where designed and applied using Power Fx formulas
Accessibility features	Built in	Designed into the app: Create accessible canvas apps

Steps to building and sharing a model-driven app

At a fundamental level, model-driven app making consists of the following areas.

- Modeling business data
- Defining business processes
- Composing the app
- Configuring security roles
- Sharing your app

A large part of the time spent building the app is dedicated to modeling the business data and in some case to defining the business processes. Customizing security roles depend on the needs of the organization.

Learn more about the steps to building a model-driven app

Using a model-driven app

Documentation has been developed that helps users successfully navigate around and interact with model-driven apps in a way that helps make them more productive.

Learn more about using model-driven apps

You can use model-driven apps on a mobile device and run them in offline mode.

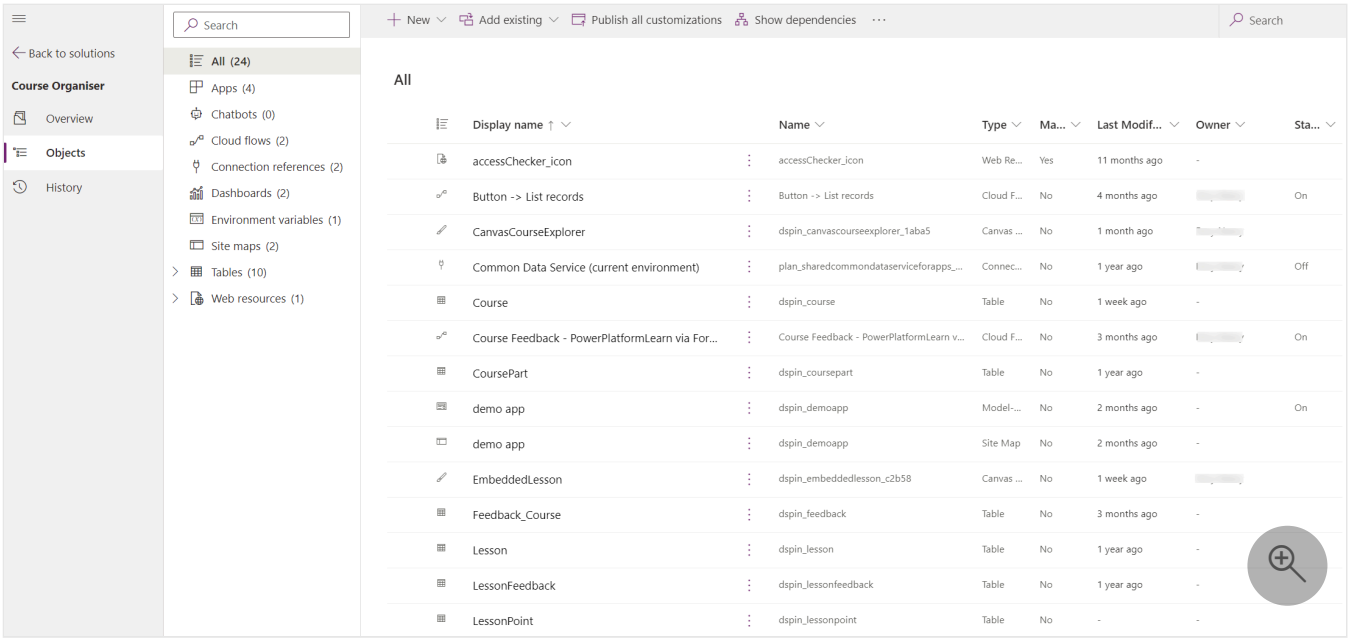
- Get started with Power Apps mobile
- Mobile offline overview (preview)

Using solutions to assist with application lifecycle management

Application lifecycle management (ALM) is the way in which we develop an app from conception to end of life.

Solutions are used to act as wrappers for all the elements required to deliver a product for use within a business.

As a minimum, a model-driven app requires a single table, a site map, plus the app itself. However, model-driven apps generally include a considerable amount more, and this includes canvas apps, security roles, environment variables, and much more.



Solutions offer the means by which these objects can be migrated between environments. Consequently, solutions are an important part of any application lifecycle management strategy.

When you create a model-driven app, you should create it within a solution.

- Learn more about solutions
- Create a solution

Minimum requirements for building a model-driven app

Model-driven apps can be complex. However, the minimum requirements for building them are relatively straightforward.

- A Microsoft Power Apps license.
- Capacity to create a Power Platform environment. This is a feature of the Power Apps and Dynamics 365 licenses held.
- A Power Platform environment.
- A Dataverse database within the environment. This is selected when you create an environment. More information: Create an environment with a database
- Privileges for the maker to work as a system customizer or environment maker in the environment where the model-driven app is created. More information: About predefined security roles
- A table. (There are many standard tables, including account, which exist by default.)
- One table column. (Every table has at least one column, and many system columns.)
- One table view, to view records. (There are a number by default.)
- One table form, to enter data. (There will be one by default.)

Using the app designer, a model-driven app is created by adding a page, which adds navigation to the app, and is typically based on a table or custom page.

Next steps

Learn about the value of model-driven apps

Building a simple model-driven app is a great way to start. You might wish to Create a solution before you do so.

To go deeper into creating model-driven apps, go to Steps to building a model-driven app.

If you're new to Power Apps, and want to learn about how to convert your ideas into a fully working solution using Power Apps, start with Planning a Power Apps project.

Understand model-driven app components helps you understand some of the elements that make up a model-driven app.

Learn about licensing for the Power Platform

Last updated on 01/09/2026

The benefits of the model-driven approach

Article • 02/15/2022

Model-driven apps are familiar to anyone who has worked with Dynamics 365 apps. To others, they're a new business productivity tool.

Dynamics 365 apps, such as Dynamics 365 Sales, Dynamics 365 Marketing, or Dynamics 365 Customer Service, are essentially a set of line-of-business apps, built on Dataverse. The utility of the Dynamics 365 model was recognized as being powerful and by making Dataverse fully customizable, has enabled more bespoke solutions to be created.

Model-driven apps are worth considering in your organization for the following reasons.

Security

- The platform is reliable. It is the same as is used by Dynamics 365 apps, to cover complex business processes and govern millions of transactions in the biggest businesses in the world.
- Model-driven apps have tried and tested application lifecycle management options.
- Dataverse supports a strong and flexible data security model affecting the degree to which users can interact with table records and columns.

Extendability

- Any Dataverse table component can be used and reused, such as contacts, accounts, or any custom tables created.
- Model-driven apps integrate with Office 365.
- Industry accelerators exist to enable a quicker entry into markets such as insurance or healthcare.
- Dataverse is extendable through use of Azure services, such as Azure Synapse Link for Dataverse.
- Canvas app integration possibilities are integral to the platform. More information: Design a custom page for your model-driven app (preview)
- Extensible with code to allow for a user experience that fits the organization more tightly.
- There is a pool of competence worldwide covering the platform through Dynamics consultants in addition to Power Platform practices.

Usability for all

- Model-driven apps are responsive by design. This means they render better on a wide range of devices.
- Model driven apps are accessible to users with disabilities including those that require screen readers.
- Offline capabilities are native to model-driven apps.
- Multilingual label support to ensure that your apps can be used worldwide.

Empowers citizen developers

- Minimal user interface maintenance as much of the experience is delivered via configuration versus outright coding.
- Predominantly a low code development platform.

Business logic

- Logic can be introduced to ensure that columns contain the data that adheres to the business rules. More information: Create business rules and recommendations to apply logic in a model-driven app form
- Business process flows to guide users to update records in the correct order and in the correct way.
- Power Automate native integration exists and flows exist as separable solution components.

Next steps

Build your first model-driven app

Release channels for your model-driven app

Microsoft regularly provides new and updated features for your model-driven apps. You can control how often the users in your organization get these new features by specifying the release channel. The concept of multiple release channels originated from Microsoft 365 channels and provides Power Platform admins with a choice of how often end user impacting features are released.

The existing cadence of twice-yearly release waves is called the *Semi-annual channel* and is currently the default for all existing apps and environments. The *monthly channel* option enables GA-ready features to be turned on each month with a four week preview notification period to allow customer validation with existing customizations.

This table provides a comparison of monthly channel and semi-annual channel.

Expand table

Category	Monthly channel	Semi-annual channel
Recommended use	Provide your users with new end user impacting features only once a month and on a predictable schedule.	For users in your organization where more training and extensive testing are needed before rollout out of new end user impacting features.
Release frequency	Once a month, on the first release for the month and following the gradual weekly rollout.	Twice a year, starting early April and early October and following the gradual weekly rollout.

Security and non-end user impacting changes continue to be delivered with the weekly releases and changes outlined in Microsoft Unified Interface versions.

The key idea is that GA-ready end user impacting features release monthly instead of waiting for the twice-yearly release cadence. The same number of features are delivered in 12 smaller releases instead of two larger releases.

Microsoft 365 products experience consistently higher user satisfaction for monthly channel compared to the semi-annual channel. Currently, we observe a user satisfaction drop with the twice-yearly releases, and we're looking to improve this experience by following the Microsoft 365 channel approach.

Configuring release channel

Release channel can be configured on the environment, app, or user level. The release channel is primarily managed on the environment or app. The user level is an override, which can be used for gradual rollout or to revert specific users.

Environment admins use the Power Platform admin center to select the channel for the environment. Makers use app designer or the solutions area to select the channel for one or more apps.

To allow gradually switching the default from semi-annual channel to monthly channel, the default value for both environment release channel and app release channel were renamed to **Auto**. There's an explicit choice for **Semi-annual** and **Monthly**. More information: Changing release channels.

With 2024 release wave 1, the Power Platform environments were changed to treat **Auto** app release channel as **Monthly** and newly created model-driven apps default to **Monthly**. With 2024 release wave 2, the Dynamics 365 environments changed to treat **Auto** app release channel as **Monthly**. Admins and makers can select **Semi-annual** if they need the slower cadence. More information: Keeping semi-annual release channel.

Monthly release schedule

Model-driven apps release updates every week, which is gradually rolled out to groups of regions over several weeks. These are the region groups with each group happening in successive weekend updates.

1. First release environment (FRE)
2. Canada, South America, India, France, South Africa, Germany, Switzerland, Norway, Korea, Singapore
3. United Arab Emirates, Japan, Asia Pacific, Great Britain, Australia
4. Europe
5. United States

The monthly release is the first weekly release of each month. The release typically matches `YYMM.1` where `YY` is the release year and `MM` is the release month. The monthly release notes are published to Released versions for Power Apps four weeks before the release reaches the region group 2. Region group 2 is used because it's the first region group for production customer environments and region group 1 is used for customer early validation.

Expand table

Monthly release	Doc published	Feature release	Group 1	Group 2	Group 3	Group 4	Group 5
Aug 2024	Jul 19	2408.2	Aug 16	Aug 23	Aug 30	Sep 6	Sep 13
Sep 2024	Aug 16	2409.1	Sep 6	Sep 13	Sep 20	Sep 27	Oct 4
Oct 2024	Sep 20	2410.1	Oct 11	Oct 18	Oct 25	Nov 1	Nov 8
Nov 2024	Oct 18	2411.1	Nov 8	Nov 15	Nov 22	Dec 6	Dec 13
Dec 2024	Nov 15	2412.2	Dec 13	Dec 20	Jan 10	Jan 17	Jan 24

See also

Changing release channels
User about dialog - channel
Power Platform admin center - Environment behavior settings
Power Platform admin center - User settings

Last updated on 03/30/2026

Change release channels for model-driven apps

The release channel affects the features that impact users. When the monthly channel is enabled for an environment, makers need to validate that their customizations work with each monthly release. This article describes different approaches for a maker to change channels and validate an upcoming release.

Flexible channel configuration

The release channel for model-driven apps can be changed in two primary ways.

- App channel
- Environment channel

In addition, the release channel can be overridden with either of these options.

- User channel override
- Browser session channel override

> **Note**
>
> When you change the release channel at the environment level, a user must refresh the browser tab twice to update the release channel information. The first refresh triggers a background update of feature configuration to a local cache. The second refresh uses the feature configuration local cache.

Ordering of channel release usage

The order of checks to determine which setting controls the channel is:

1. URL override.
2. User channel override.
3. App release channel value other than Auto (for example, Monthly, Semi-annual).
4. Environment release channel value other than Auto (for example, Monthly, Semi-annual).
5. App release channel and environment release channel are both Auto.
   a. With 2024 release wave 1, Power Apps environments use monthly channel.
   b. Starting with 2024 release wave 2, Dynamics 365 environments use monthly channel.

Dynamics 365 app release channel settings

With 2024 release wave 1, the following Dynamics 365 apps changed their app release channel to **Monthly**:

- Customer Service workspace
- Customer Service Hub
- Customer Service admin center
- Field Service
- Field Resource Hub
- Field Service Mobile
- Resource Scheduling
- Resource Scheduling Optimization
- Customer Insights - Journeys
- Business performance analytics
- Project Operations Team Member
- Power Platform environment settings

With 2024 release wave 2, this app is changing the app release channel to **Monthly**:

- Sales Hub

Keep semi-annual release channel

When an environment needs to have a slower release cadence, these configurations are recommended.

Change apps with explicit app release channel

Model-driven apps with the app release channel set to a value other than **Auto** need to be changed to **Semi-annual**. These apps can be deployed to all environments as a managed solution.

1. Customize any out of box Dynamics 365 app.
2. Update any custom app created after 2024 release wave 1 when the new app started defaulting to **Monthly**.

The preferred approach is to follow Change app channel in app designer and then deploy the app's solution to all environments.

> **Note**
>
> Model-driven apps in Power Platform environments use **Monthly** when the app release channel is **Auto** regardless of the environment release channel. All model-driven apps in

Power Platform environments need to set the app release channel explicitly.

Override apps when no explicit app release channel

Model-driven apps in a Dynamics 365 environment that have **Auto** for app release channel can choose either option:

- Change each environment release change to **Semi-annual**.
- Change each app to **Semi-annual**.

Changing the environment channel

The environment channel can be set using the Power Platform admin center or with code.

Expand table

Environment release channel	Environment setting value	Behavior
Auto	0	Environment default value is currently **Semi-annual**, but changes to **Monthly** with 2024 release wave 2.
Monthly	1	Environment explicitly set to **Monthly Channel**.
Semi-annual	3	Environment explicitly set to **Semi-Annual Channel**.

Power Platform admins can change the release channel using the environment's behavior settings. More information: Manage behavior settings.

Developers can change the environment channel by updating the ReleaseChannel column value for the row in the Organization table. There's always a single row in the organization table. More information:

- Update a record using Web API
- Update a record using the SDK for .NET

Changing the app channel

Use the app channel to override the release channel for a model-driven app by using Power Apps app designer or the **Solutions** area.

Expand table

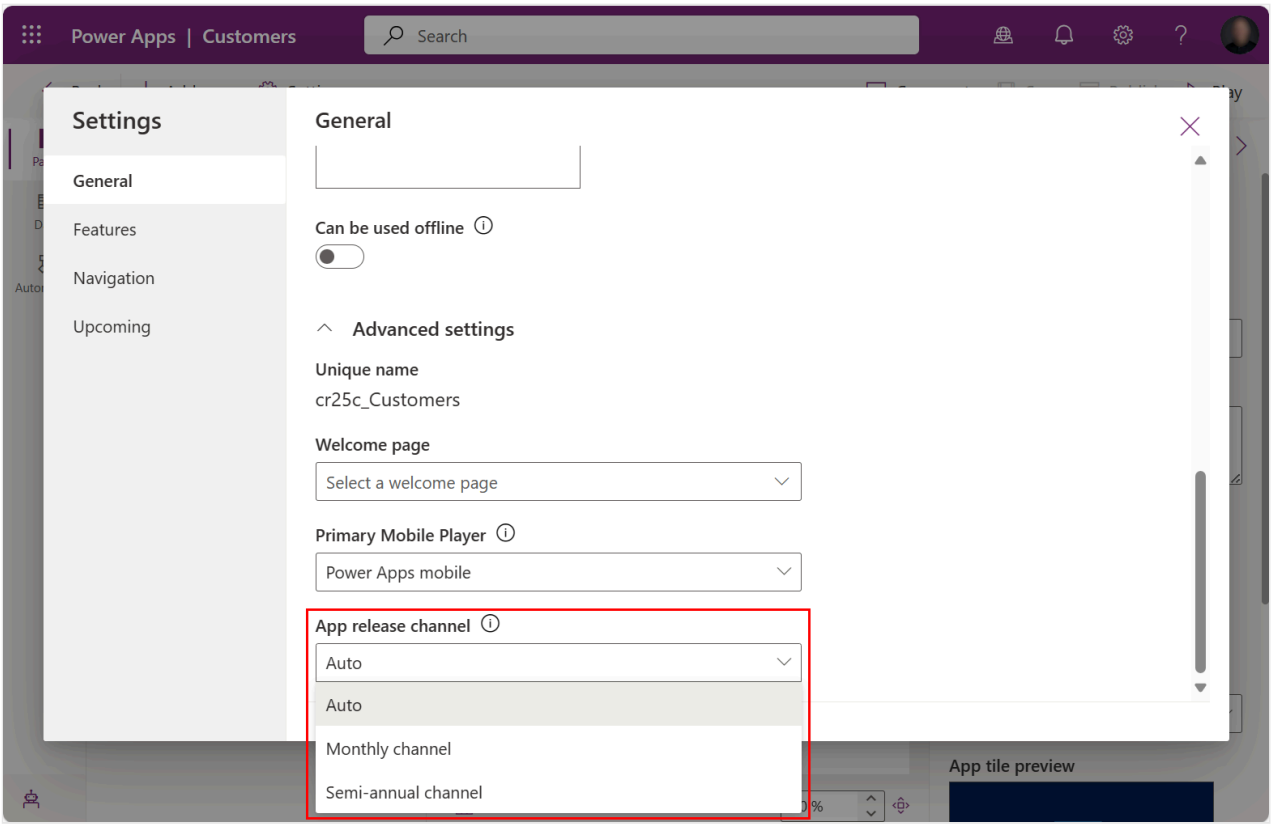
App release channel	App setting value	Behavior
Auto	0	App default value is **Semi-annual** for Dynamics 365 environments and **Monthly** for Power Apps environments. With 2024 release wave 2, the default changes to **Monthly**.
Monthly	1	App explicitly set to **Monthly Channel**.
Semi-annual	3	App explicitly set to **Semi-Annual Channel**.

> **Note**
>
> App release value `2` is used for internal testing and receives updates faster than monthly channel so isn't supported for production usage.

Change app channel in app designer

A maker can use the app designer to explicitly set the release channel for an app, which overrides the environment channel.

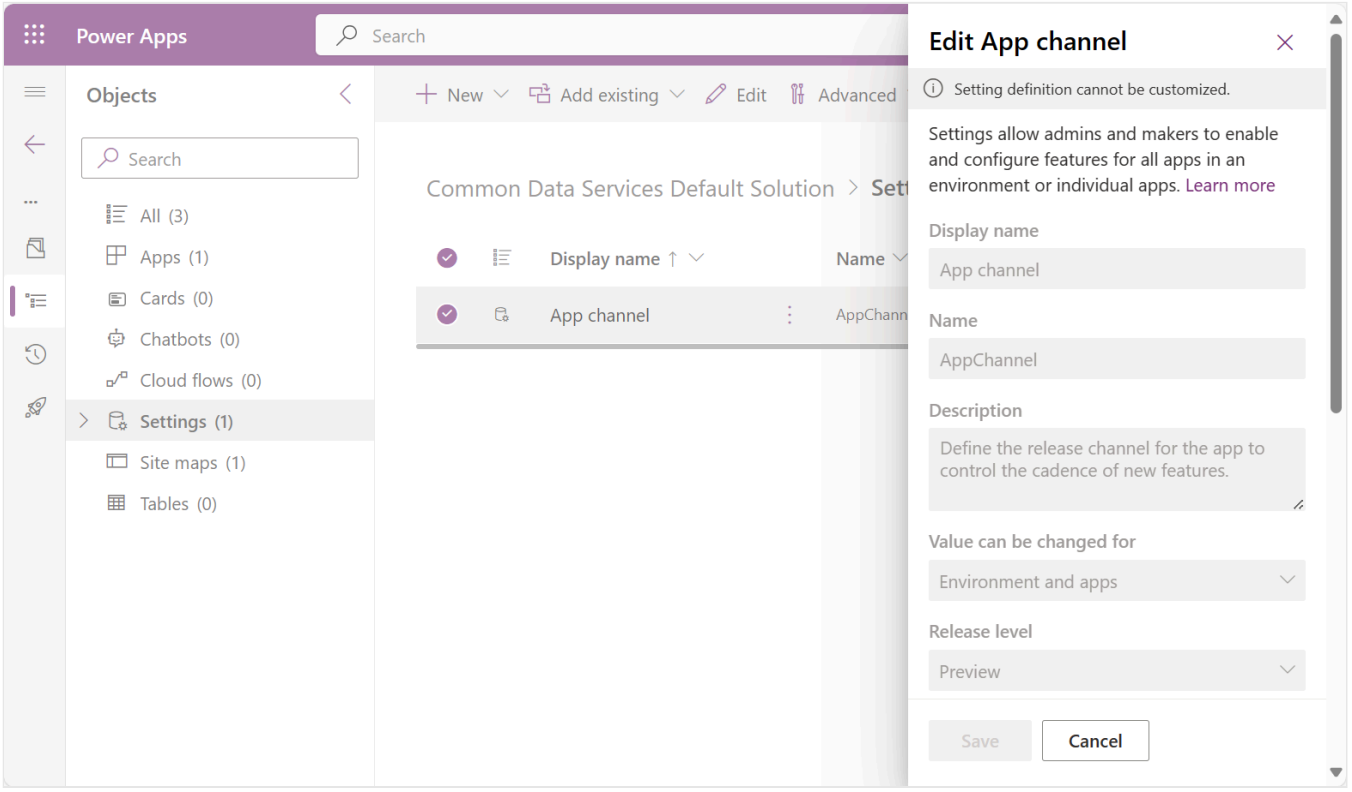
1. Open https://make.powerapps.com/ .
2. Under **Solutions** open an existing solution containing a model-driven app.
3. Open the app in the app designer.
4. Open **Settings** dialog.
5. Under the General tab, expand **Advanced settings**.
6. Use **App release channel** to change the app release value. More information: Changing the app channel



7. Save and publish the app.

Change app channel in the solutions area

As an alternative, a maker can use the **Solutions** area to explicitly set the release channel for all apps within the environment.



1. Open https://make.powerapps.com/ .

2. Under **Solutions** open an existing solution with one or more model-driven apps:
3. Add the existing app setting **Allow new app channel default** into the solution:
    a. Select **Add existing** > **More** > **Setting**.
    b. Search for *app channel*.
    c. Select the item **App channel**.
    d. Select **Add**.
4. To change the app channel for all apps in the environment:
    a. Edit the setting **App channel**.
    b. Find the section **Setting environment values**.
    c. Select **New environment value** and then enter the integer for the channel.
    d. Select **Save** which publishes the Setting Definition.

> ⓘ **Note**
>
> When the setting definition is used to change one or more apps as part of a single solution, you must republish the apps for the change to take effect. Also, making this change causes the apps to have dependencies on the solution with the customization.

Set the default for new apps to monthly channel

As part of the gradual migration to default all apps to use monthly channel, newly created model-driven apps gradually start seeing the app channel defaulted. Admins or makers can control the release channel default for new apps using an app setting. The app setting **Allow new app channel default** defaults to **Yes**, which means a newly created app is set to **Monthly**.

To override this behavior, change the app setting to **No** in the environment or apply to multiple environments by including in a solution that is imported into the environments.

Prevent new app default to monthly channel

The new app default can be prevented by switching **Allow new app channel default** to **No**. This change causes the new app to be created with release channel **Auto** value.

The following steps change the default for all new apps within an environment. This app setting override can also be put into a solution and imported into environments to prevent new apps from having a default set.

1. Go to **Solutions** and open an existing or create a new solution.
2. Add the existing app setting **Allow new app channel default** into the solution:
    a. Select **Add existing** > **More** > **Setting**.
    b. Search for *app channel default*.

c. Select the item **Allow new app channel default**.
d. Select **Add**.

3. Edit the setting **Allow new app channel default**:
   a. Select **Allow new app channel default**.
   b. Select **New environment value**.
   c. Change the environment value to **No**.
   d. Select **Save**.

Changing the user channel

The user channel can be used to override both the environment channel and the app channel using the Power Platform admin center or with code.

Power Platform admins can change the release channel using the settings in the environment's user list. More information: Manage user channel override.

Developers can change the user release channel by updating the ReleaseChannel column value for the row in the UserSettings table. The user channel can be programmatically updated with the same approach as the environment channel.

Changing the browser session channel

A single browser session can be changed by adding the URL parameter `&channel=<channelname>` as a temporary override. This URL parameter is used for all navigation within the browser tab. It might not be copied to a new browser tab.

Expand table

Channel	URL parameter
Semi-annual	`&channel=semiannual`
Monthly	`&channel=monthly`

When the channel is monthly, the monthly release can be changed using the URL parameter `&channelrelease=<releasename>`. The release name format is the two digit year and two digit month, like *YYMM*. Additionally the next monthly release can be set using `&channelrelease=next`.

Expand table

Monthly Release	Release Name Parameter
October 2023	`&channelrelease=2310`
November 2023	`&channelrelease=2311`
December 2023	`&channelrelease=2312`
January 2024	`&channelrelease=2401`

> ⓘ Note
>
> Any valid `YYMM` release can be entered but future dates might not have any features defined.

Validating the next monthly release

Validation should be done for each monthly channel release before it is automatically enabled for users. Users can test when the validation build version reaches the environment.

The easiest way to validate is by appending `&channelrelease=next` that automatically sets the release channel to the next upcoming monthly release.

1. Find the current monthly release for a model-driven app by selecting **Settings** > **About**. The release version follows **Channel: Monthly** and is a date like *July 2023*.
2. Append `&channelrelease=next` to the URL.
3. Repeat the first step to observe an updated release version.

To validate against a specific monthly release, use the following steps:

1. Find the current monthly release for a model-driven app by selecting **Settings** > **About**. The release version follows **Channel: Monthly** and is a date like *July 2023*.
2. Find the monthly release short name by opening Unified Interface monthly channel releases.
3. A specific release can be set with the URL parameter ```&channelrelease=`` with the next release short name like *2308*.

Comparing features across channels and releases

When a user running the monthly channel observes unexpected behavior, the following steps can help investigate where the behavior occurred.

- Check if the unexpected behavior exists in the semi-annual channel by using the URL parameter `&channel=semiannual`. If the behavior also exists in the semi-annual channel, it's unrelated to the monthly channel and should follow normal support processes.
- Check if the unexpected behavior exists in the previous monthly release by using the URL parameter `&channelrelease=` with the prior release short name like *2308*. If the two monthly releases behave the same, then it's likely unrelated to a specific monthly channel release and should follow normal support processes.
- When a change is noticed between monthly releases, review the changed features in Unified Interface monthly channel releases to learn more.

See also

Release channel overview
User About dialog - channel
Power Platform admin center - Manage behavior settings
Power Platform admin center - Manage user channel override.

Last updated on 03/30/2026

Meet the model-driven app designers

Article • 02/15/2022

In order to fully develop a model-driven app, multiple designers are required at different stages in the development process.

Broadly speaking, app development breaks down into the following stages:

- Data model development
- App development
- Data security

Assets created, such as tables, cloud flows, or model-driven apps, have the capacity to be held inside solutions to enable secure application lifecycle management.

This article focuses on the first two stages. For more information about data security, go to Security in Microsoft Dataverse

> **Note**
>
> It isn't necessary to use all the editors to create a model-driven app. As a minimum, concentrate on the table column editor, form designer, and view designer in addition to adding any required relationships. Then use the app designer to pull the components of the app together.

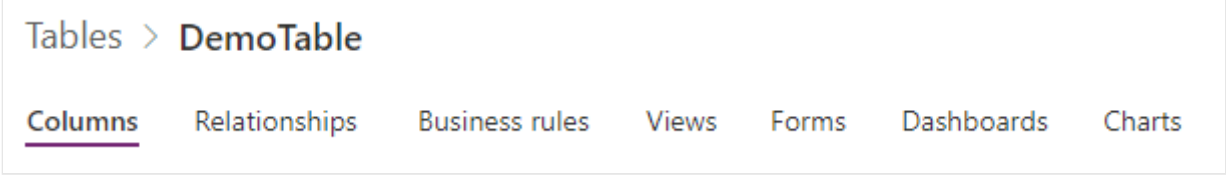
Data-model development

The value of model-driven apps centers around having a strong, secure, reusable data-model. Once the data-model is in place, the process of developing the app is straightforward.

The table designer, and related designers

Development of the data-model starts with using the table designer. A table is a significant artifact of a model-driven app. There is a table designer, which defines the table structure, and includes access to the other designers for relationships, views, forms, and so on.

The table designer is shown below.



The following table describes the designers, what they do, and provides links about how to open them.

Editor	Description	Link
Table designer	Tables hold record metadata in columns for a wide range of data types such as text, email, image, currency, and more. Many standard tables are available. You can customize a non-system standard table. You can also create a new custom table from scratch. The table designer is effectively an entry point to the other designers, such as view and form. In many cases, the designers are native to the table designer and in others a new browser tab opens where editing can take place. The columns area of the table designer allows the developer to define the table columns.	Create a custom table
Relationships	Part of the table designer. Relationships are fundamental to the operation of any model-driven app. Without them all tables are isolated.	Create a relationship
Business rule designer	A separate designer accessed via the table designer. Business rules apply rules or recommendation logic to a form to set column requirements, hide columns, validate data, and more. App designers use a simple interface to implement and maintain fast-changing and commonly used rules.	Create a business rule for a table
View designer	Part of the table designer. Views are tied to the table and present table data in columns by selecting, positioning, and filtering them.	Create a view
Form designer	A separate designer accessed via the table designer. Forms allow users to interact with data held in table records.	Create a form
Dashboard designer	A separate designer accessed via the table designer. Used to create and edit dashboards presenting different data visualizations, such as embedded Power BI reports, data charts, and views.	Create a dashboard
Chart designer	A separate designer used to create table charts.	Create a system chart

Editor	Description	Link
Business process flow designer	Business process flows walk users through a standard business process. While they are a part of the solution, they are created using Power Automate.	Create a business process flow

Business logic development

Business logic can be introduced in the form of business rules and business process flows.

In addition, Power Automate cloud flows can be used in a model-driven app that leverage cloud flow functionality, which can access hundreds of data connectors, not just Dataverse.

Learn more about using Power Automate with Dataverse

App designer

The app designer is the tool that is used to create the app and to configure the tables used in the app. It is important to understand that the quality of the app has much less to do with the edits performed at this stage than those using the other designers.

Makers can choose the views, forms, charts, and dashboards relevant to the app in development.

Create an app using the app designer

Next steps

Steps to building a model-driven app

Build your first model-driven app

Steps to building a model-driven app

This article describes the steps for building a model-driven app in Power Apps.

At a fundamental level, model-driven app making consists of the following stages.

Expand table

Stage	Purpose
Modeling business data	To ensure that the data is constructed in the correct way to address the business problem.
Defining business processes	To help users update the relevant tables and complete their work efficiently and accurately.
Composing the app	To create the app and select tables, and elements of tables, that are relevant to the app.
Configuring security roles	To ensure that app users can only interact with data relevant to their roles.
Sharing the app	App distribution.

Each stage requires a range of designers and editors to ensure that the application can be created, whether the app maker is updating the data model or going through the process of composing the app.

While it might seem strange to have multiple designers or editors, these tools reflect the various technologies that have been brought to bear within Microsoft Dataverse. Often makers move seamlessly through the editors to construct the app.

For a simple walk-through of building a first app, go to building a first model-driven app.

Modeling business data

To model business data, you determine what data your app needs and how that data relates to other data. Model-driven design uses a metadata-driven architecture so that designers can customize the application without writing code. Metadata means “data about data” and it defines the structure of the data stored in the system.

> **Note**

With the exception of model-driven apps that have a custom page, model-driven apps can't be created without Dataverse tables. However, Dataverse tables can be used by many Power Platform services, including canvas apps and Power Automate.

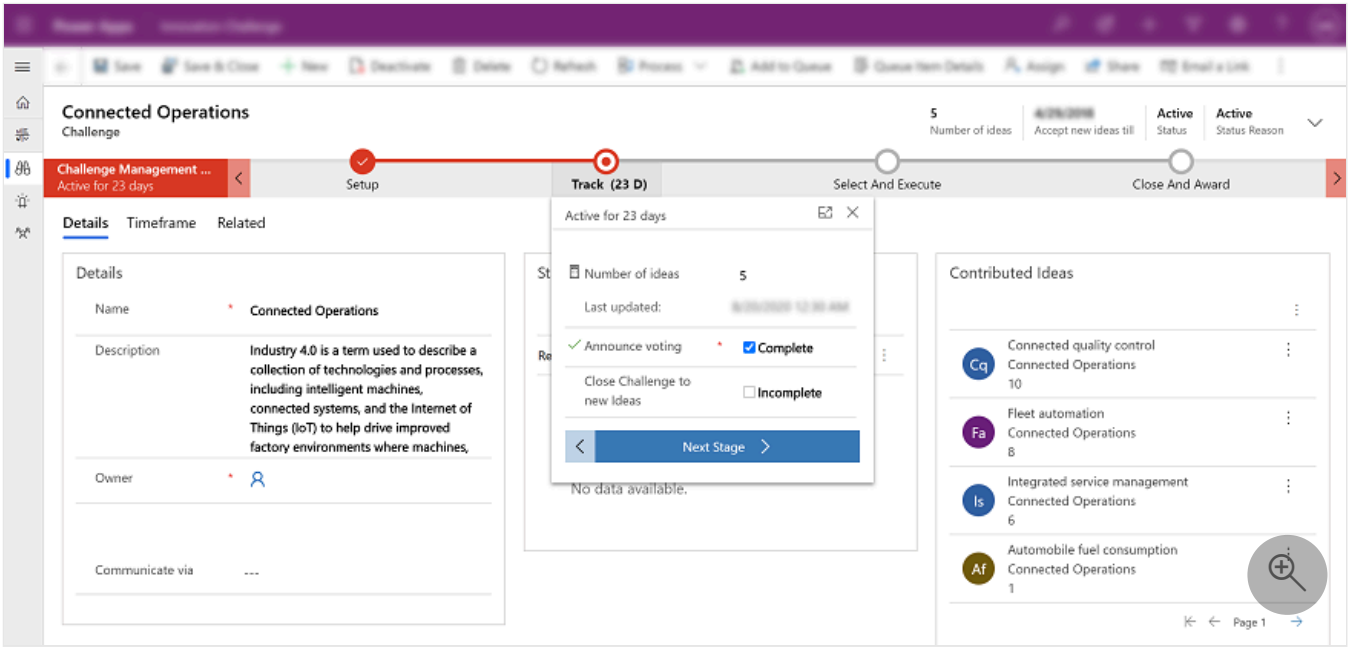
Tutorial: Create a custom table that has components in Power Apps

Defining business processes

Defining and enforcing consistent business processes is an important aspect of model-driven app design. However, you can create a model-driven app without a business process configured around it.

Consistent processes help ensure your app users focus on their work and not on remembering to perform a set of manual steps. These processes can be simple or complex and can contain operations on multiple tables.

The screenshot here illustrates the impact of having a business process flow in place.



Business process flows are created and configured using the business process flow designer.

More information: Business process flows overview and Apply business logic with Microsoft Dataverse.

Composing the model-driven app

After you create a data model and define business processes where necessary, you can build the app.

This is done by building an app using the modern designers and editors.

With the modern designers and editors, makers can see the effects of the changes that they make while designing, whereas with the legacy interfaces there was a level of abstraction between the final user experience and the design experience. More information: Learn how to build a modern app

When you build an app, a site map is created, defining the navigation experience for users. With the modern app designer, this is a part of the app design process.

More information: Building a site map

Playing the app

Play your app through the app designer.

Once these stages are complete, you can move onto the final phases associated with distributing your app.

Configuring security roles

Access to tables is defined using security roles and these roles govern the actions that users can perform with the tables within Dataverse. Without this, users have no meaningful access to the app.

These actions cover Create, Read, Write, Delete, Append, Append To, Assign, and Share. Security roles need to be first configured and then users are assigned to roles at the point of sharing.

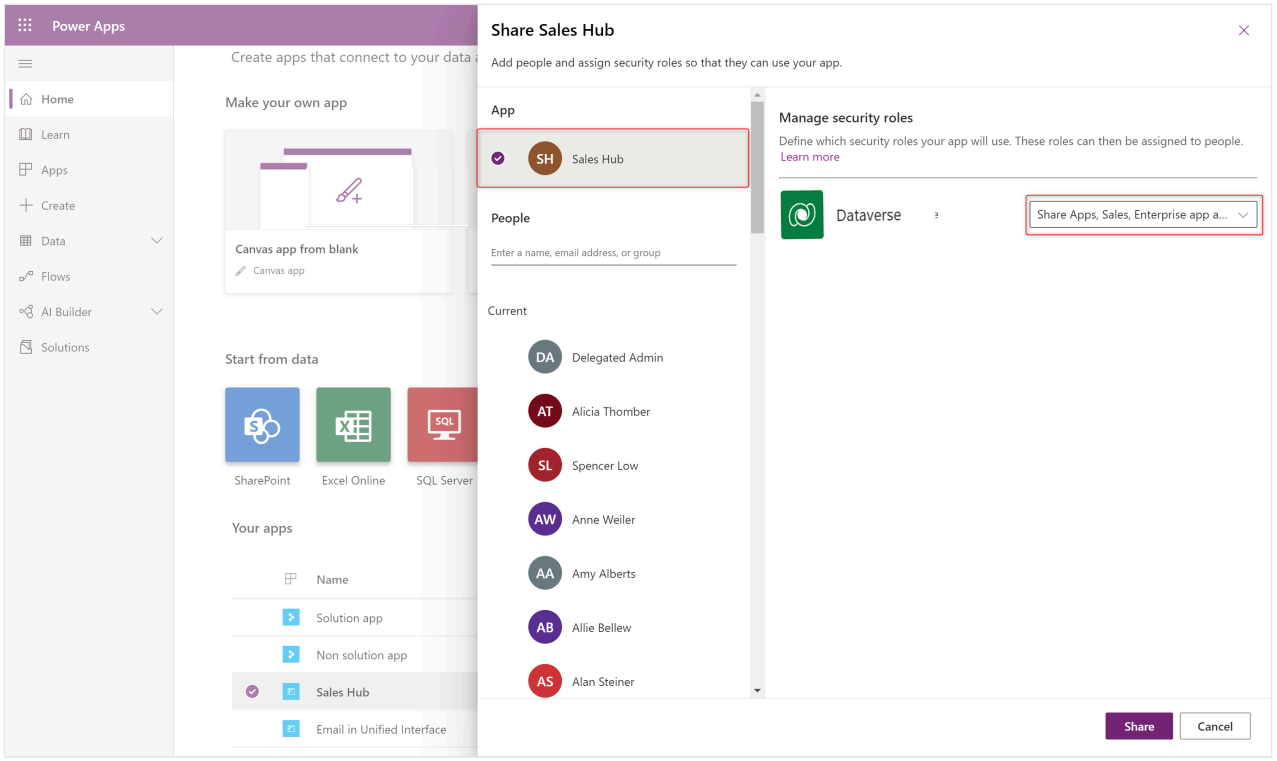
For more information on understanding, creating and configuring security roles go to:

- Power Platform documentation
- Get started with security roles in Dataverse

Sharing the app

To share an app, two actions are required:

- Assign the user a security role. This means that they have permission to see the data.



- Share a link to the app. To get the link:

1. Sign in to Power Apps, and then select **Apps** on the left navigation pane.
2. Select the model-driven app you want, and then select **Details** on the command bar.
3. Copy the **Web link**. Alternatively, you can make a copy of the **Mobile QR code** for mobile users.
4. Paste the app URL in a location so that your users can access it, such as by posting it on a SharePoint site or send via email.

Sharing an app and security roles are intrinsically linked. To properly share an app, you need to have a strong understanding of both. Discover more about sharing apps and establishing security

Using a model-driven app

Documentation is available that helps users of model-driven apps successfully navigate around and interact in ways that help make them more productive. Learn more about using model-driven apps

Next steps

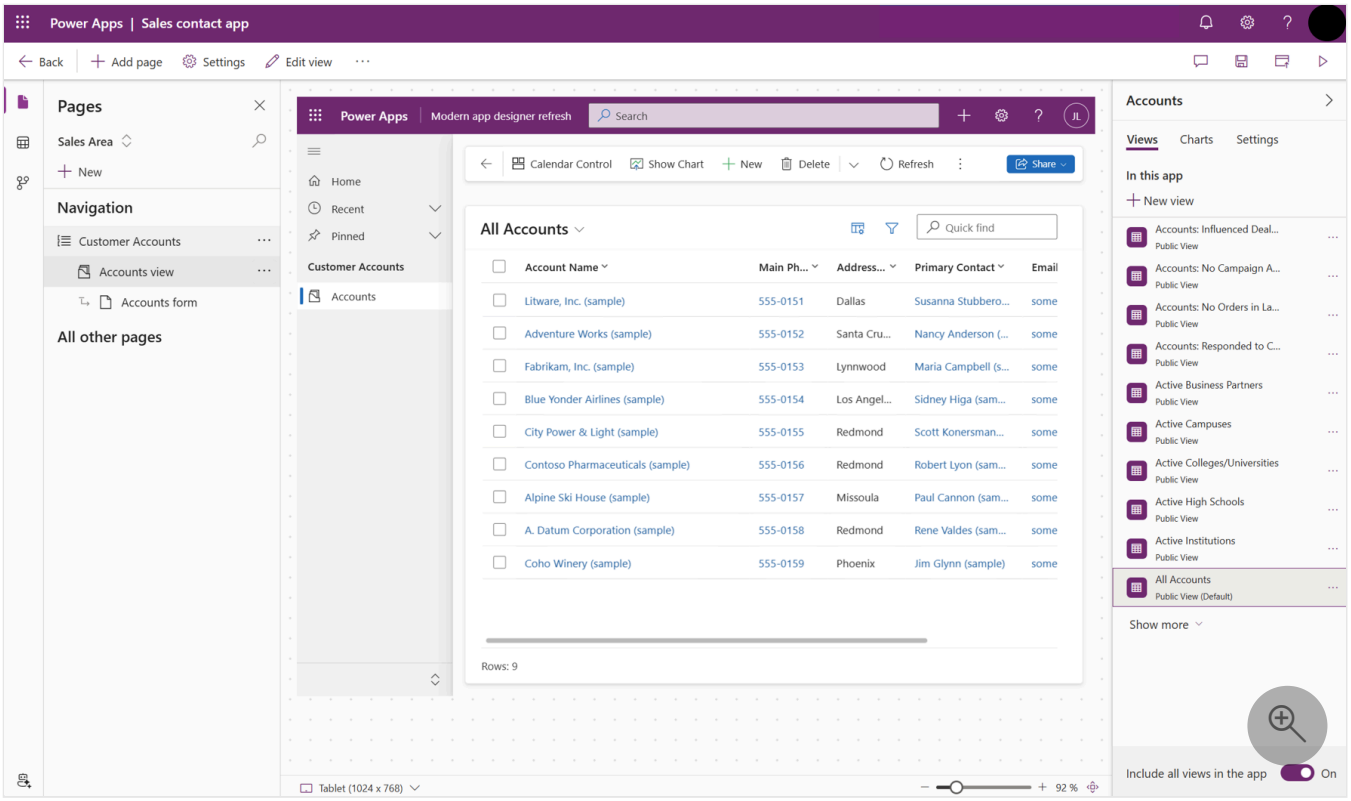
Build your first modern app

Meet the model-driven app designers

Last updated on 04/08/2026

Build your first model-driven app

In this tutorial, you create a model-driven app by using one of the standard tables that is available in Microsoft Dataverse, the account table.



1. Sign in to Power Apps. If you don't already have a Power Apps account, select the **Get started free** link.

2. Select an environment. An environment in Power Apps is a space to store, manage, and share your organization's business data, apps, chatbots, and flows. It also serves as a container to separate apps that might have different roles, security requirements, or target audiences.

   Each environment can have one Dataverse database.

   Select the environment you want. If your tenant allows you to create your own environment, select the current environment from the top right, and then select **Try it now**. More information: Get your developer environment (preview)

3. We recommend creating your model-driven app from a solution. A solution is a package that can contain Dataverse tables, forms, views, apps, flows, and other components. By building your model driven-app in a solution, you can easily move it to other environments or preserve a copy in your source control repository.

   Select **Solutions** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

4. Select **New solution**, enter a **Display name** for your solution, such as *Power Apps Training*, select a publisher (the **CDS Default Publisher** is okay for this tutorial), and then select **Create**. For more information about creating a solution, go to Create a solution.

5. In your solution, select **Add existing**, and then select **Table.**

6. In the **Add existing tables** list, select the **Account** table, and then select **Next**.

7. Since this is a standard table, leave the default setting for account table objects and metadata and select **Add.**

   Now that you have a solution and have added the account table to it, you're ready to create a model-driven app.

8. In your solution, select **New**, select **App**, and then select **Model-driven app.**

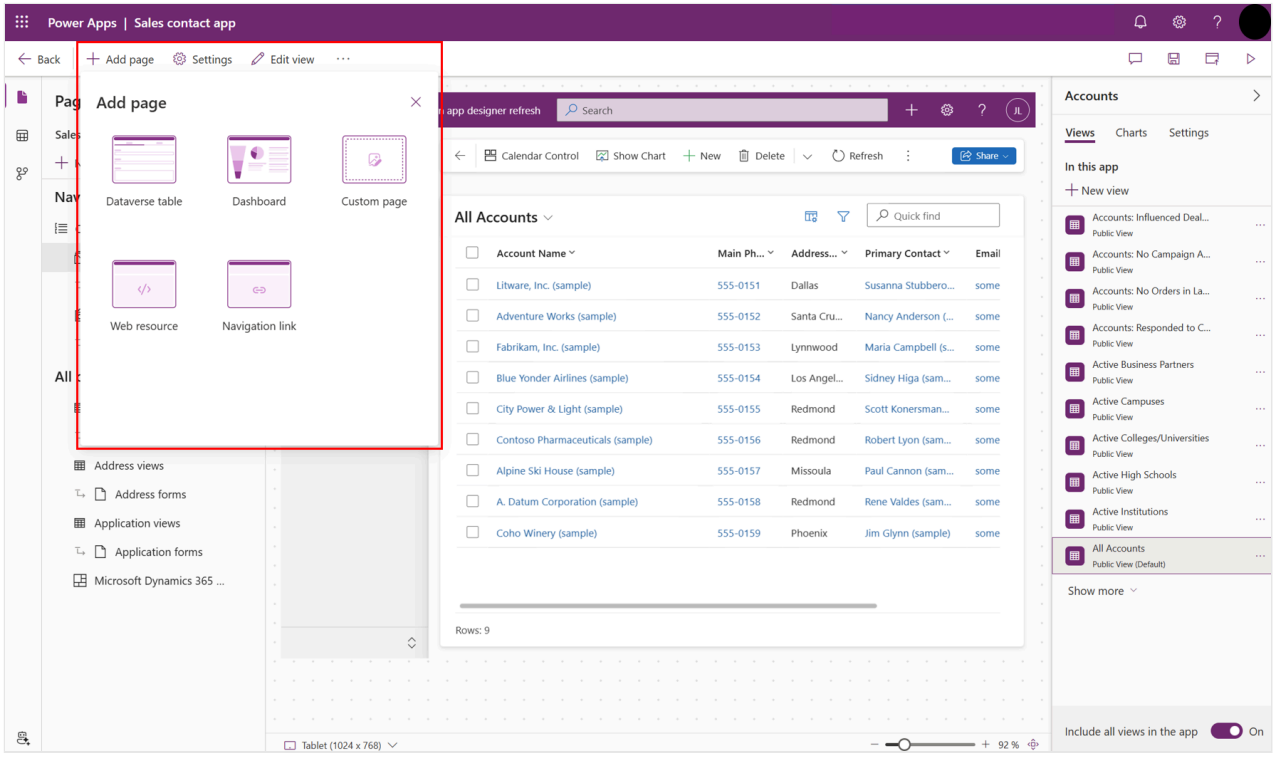
9. Enter a name for the app, such as *My new custom app*, and then select **Create**.

   Your app loads in the model-driven app designer.

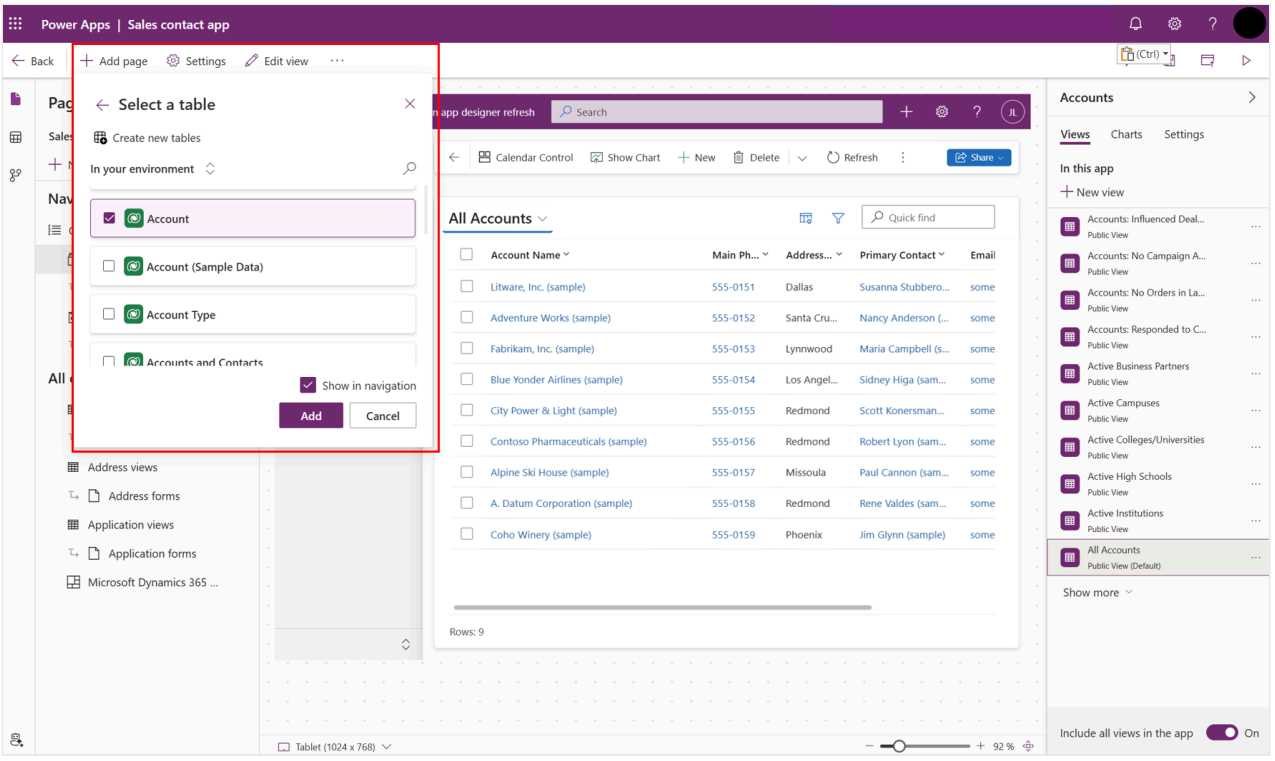
Add a page to your app

Next, you add a new page to the model-driven app in the app designer.

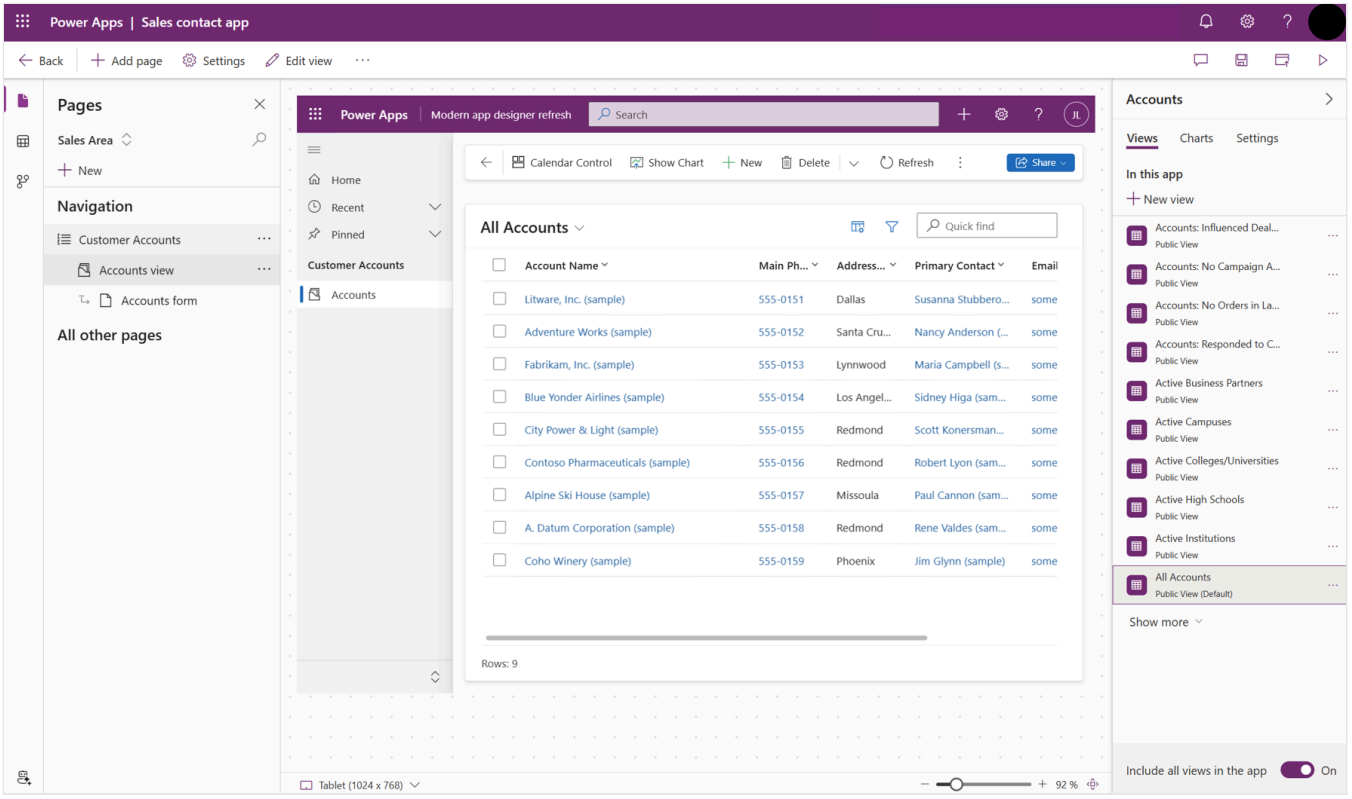
1. Select **Add page** on the command bar, and then select **Dataverse table**.



2. Select **Account**, and then select **Add**.



Your app with the account table is displayed similarly to how it appears to users at run time when published. If your environment has sample data, the sample list of active accounts might appear in the **All Accounts** view.



Save and publish your app

Save your app so that your app and changes aren't lost. To make it available to users, publish it.

To save without publishing, on the app designer command bar, select **Save**.

To save and publish, on the app designer command bar, select **Publish**.

Create an app description with Copilot

If your app is in a managed environment, you can use AI to help you create a description for your app.

> **Important**
>
> - To use this capability your app must be in a **managed environment**.
> - This capability is powered by **Azure OpenAI Service**.
> - This capability might be subject to usage limits or capacity throttling.
> - To understand capabilities and limitations of AI-powered and Copilot features in Power Apps, go to **Responsible AI FAQs for Power Apps**

When you save and publish your app, on the **Publish** dialog box select, **Create descriptions using AI** to replace your current description or, if your app doesn't have a description, Copilot generates one for you.

You can also generate an app description by going to, **Settings** > **General**. Under **Description** select, **Create descriptions using AI**.

For information that describes the AI impact of Power Apps generate app descriptions with Copilot feature, go to FAQ for generate app descriptions with Copilot.

Enable or disable AI-generated app descriptions

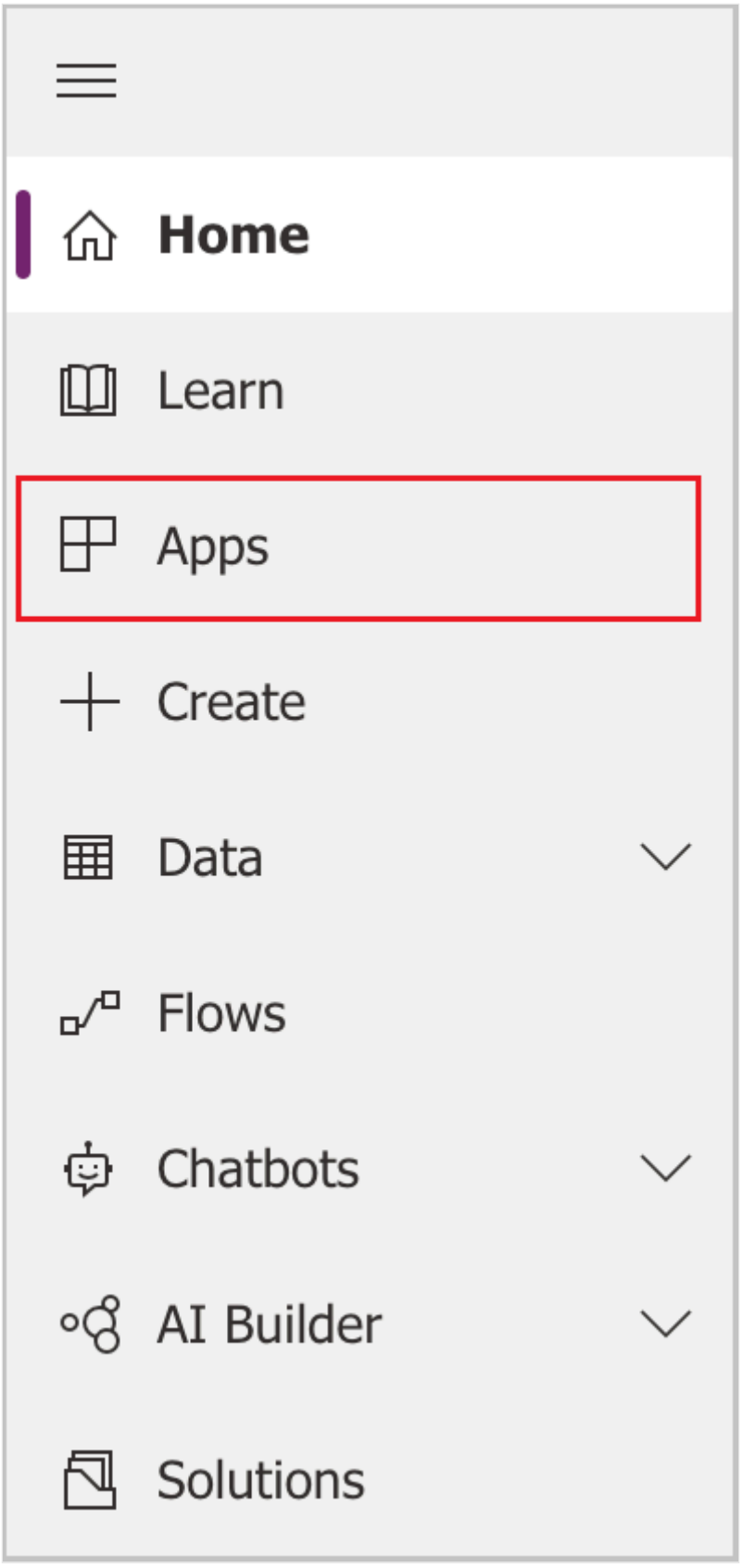
App AI-generated descriptions are managed by a setting in the Power Platform admin center.

1. Open https://admin.powerplatform.microsoft.com/environments
2. Select an environment, and then select **Edit managed environments**.
3. Under the section titled **Enable AI-generated app descriptions**, select or clear **AI-generated app descriptions**
4. Select **Save**.

Clear the browser cache and close any active Power Apps (make.powerapps.com) browser windows. The feature is now be enabled or disabled.

Run your app on a desktop computer

1. To run your app, sign in to Power Apps, and on the left navigation pane, select **Apps**.



2. Select the app from the app list. The app opens in your browser. If you want to show the chart, use the button **Show Chart**.

Run your app on mobile

To run your app on your mobile device, follow these steps:

1. Download the app from your device's mobile app store:
    - IOS: Power Apps on the App Store (apple.com)
    - Android: Power Apps - Apps on Google Play
2. Open the app, tap **Sign in**, and then enter your Microsoft work or school account.
3. Select your app from the list to run it.

Next steps

Add additional pages and configurations to make your app do more!

- Add a custom page where a canvas app runs from your model-driven app. More information: Overview of custom pages for model-driven apps
- Create Dataverse custom tables that you can add to your app the same way as you added the account table in this tutorial. More information: Create and edit tables using Power Apps
- Identify the URL of your model-driven app. More information: Run a model-driven app in a browser
- Share your app. More information Share a model-driven app.
- Learn more about building model-driven apps. More information: Understand model-driven app components

> **Tip**
>
> Ready to convert your ideas into an app? Start here: **Planning a Power Apps project**.

Last updated on 01/09/2026

Understand model-driven app components

Article • 01/22/2025

A well designed model-driven app consists of several components you select using the app designer to build the appearance and functionality of the finished app. The components and component properties that makers use to create an app become the metadata.

To understand how each of these components relates to app design, they're separated here into *data*, *UI*, *logic*, and *visualization* categories.

Data components

These components determine what data the app is based upon and what designer is used to create or edit the component.

Data components are either designed at a table level within an environment, or within a solution contained within the environment.

Expand table

Component	Description	Designer
Table	A container for records with properties that you track, such as a contact or account. Many standard tables are available. You can customize a nonsystem standard table (production table) or create a custom table from scratch.	Power Apps table designer
Relationship	Table relationships define how tables can be related to each other. There are 1:N (one-to-many), N:1 (many-to-one), and N:N (many-to-many) types of relationships. For example, adding a lookup column to a table creates a new 1:N relationship between the two tables and lets you put that lookup column on a form.	Power Apps table designer
Column	A property that is associated with a table. A column is defined by a data type, which determines the type of data that can be entered or selected. Examples include text, number, date and time, currency, or lookup (creates a relationship with another table). Columns typically are used with forms, views, and searches.	Power Apps table designer
Choice column	This is a special type of column, which provides the user a set of predetermined options. Each option has a number value and label. When added to a form, this column displays a control for the user	Power Apps choice designer

Component	Description	Designer
	to select an option. There are two kinds of choices; choices, where the user can only select one option, and multi-select choices, which allow more than one selection.	

More information: Define data for your model-driven app

UI components

These components determine how users interact with the app.

Expand table

Component	Description	Designer
App	Determines the application fundamentals such as components, properties, client type, and URL for your app.	App designer
Site map	Specifies the navigation for your app.	App designer
Form	A set of data-entry columns for a given table that matches the items that your organization tracks for the table. For example, a set of data-entry columns where users input relevant information to track a customer's previous orders along with specific requested reorder dates.	Form designer
View	Views define how a list of rows for a specific table is displayed in your application. A view defines the columns to display, width of each column, sort behavior, and the default filters.	View designer or app designer
Custom page (preview)	A canvas based page which allows flexible layout, low-code Power Fx functions, and Power Apps connector data. For more information, go to Model-driven app custom page overview (preview)	Canvas designer

Logic components

Determines the business processes, rules, and automation the app has. Power Apps makers use a designer that is specific to the type of process or rule.

Expand table

Type of logic	Description	Designer
Business process flow	An online process that walks users through a standard business process. For example, use a business process flow if you want everyone to handle customer service requests the same way, or to require staff to gain approval for an invoice before submitting an order.	Business process flow designer
Workflow	Workflows automate business processes without a user interface. Designers use workflows to initiate automation that doesn't require any user interaction.	Workflow designer
Actions	Actions are a type of process that let you manually invoke actions, including custom actions, directly from a workflow.	Process designer
Business rule	Used to apply rule or recommendation logic to a form, such as to set column requirements, hide columns, or validate data. App designers use a simple interface to implement and maintain fast-changing and commonly used rules.	Business rule designer
Power Automate Flow	Power Automate is a cloud-based service that lets you create automated flows between apps and services to get notifications, synchronize files, collect data, and more.	Power Automate

More information: Apply business logic in your model-driven app

Additional options for adding custom business logic

Use plug-ins to extend business processes
Workflow extensions

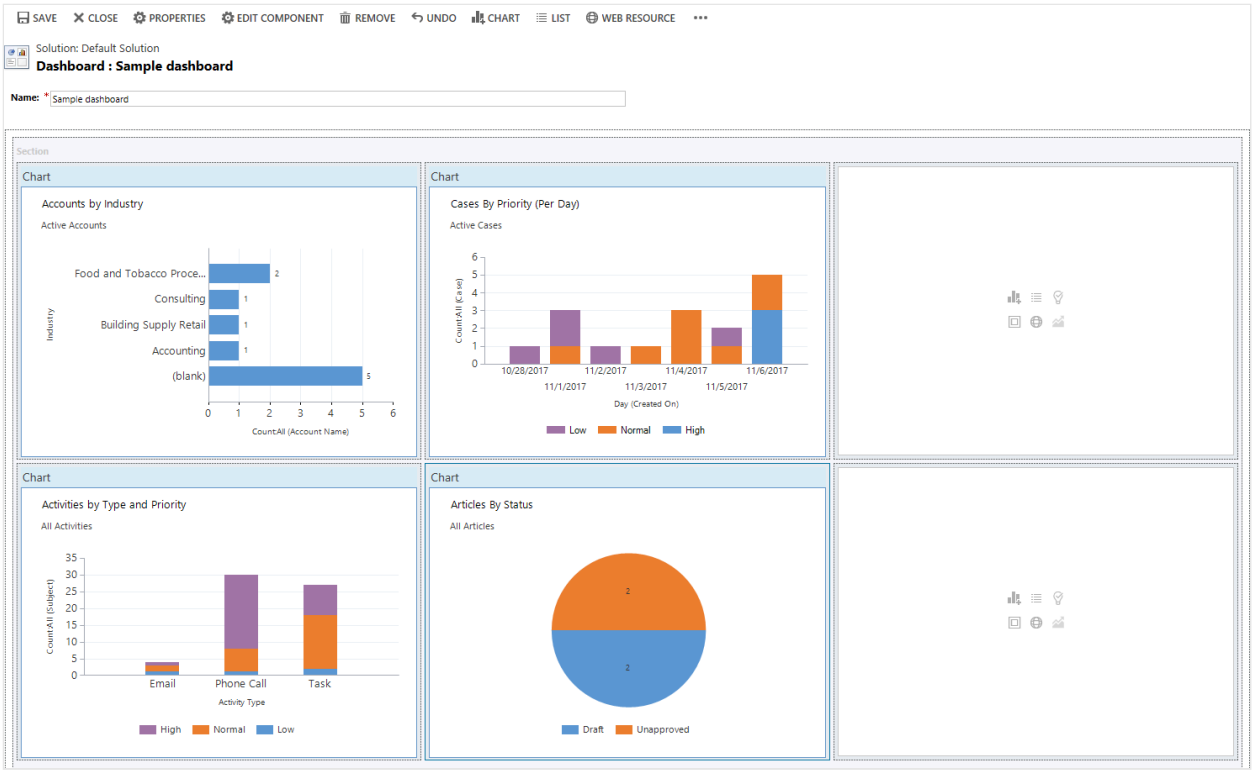
Visualizations

Determines what type of data visualizations and reporting the app has available.

Expand table

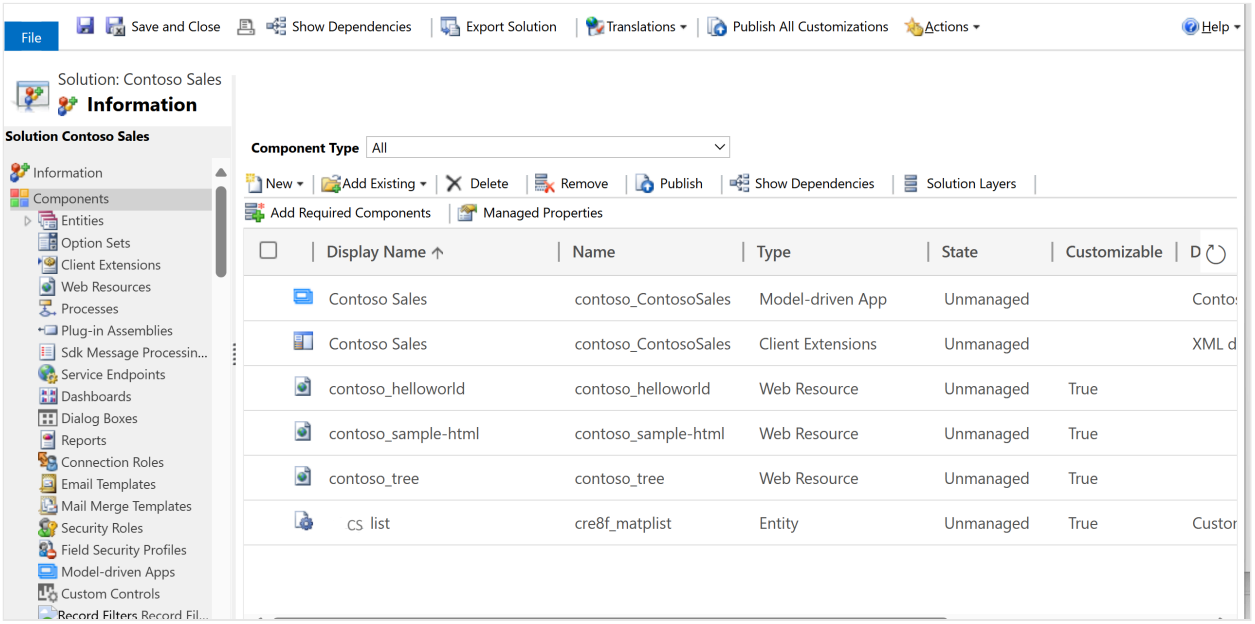
Component	Description	Designer
Chart	A single graphic visualization that can be displayed within a view, on a form, or be added to a dashboard.	Chart designer
Dashboard	Functions as a palate for one or more graphic visualizations that provide an overview of actionable business data.	Dashboard designer

Component	Description	Designer
Embedded Power BI	Add embedded Power BI tiles and dashboards to your app. Power BI is a cloud-based service that provides business intelligence insight.	Combination of chart designer, dashboard designer, and Power BI



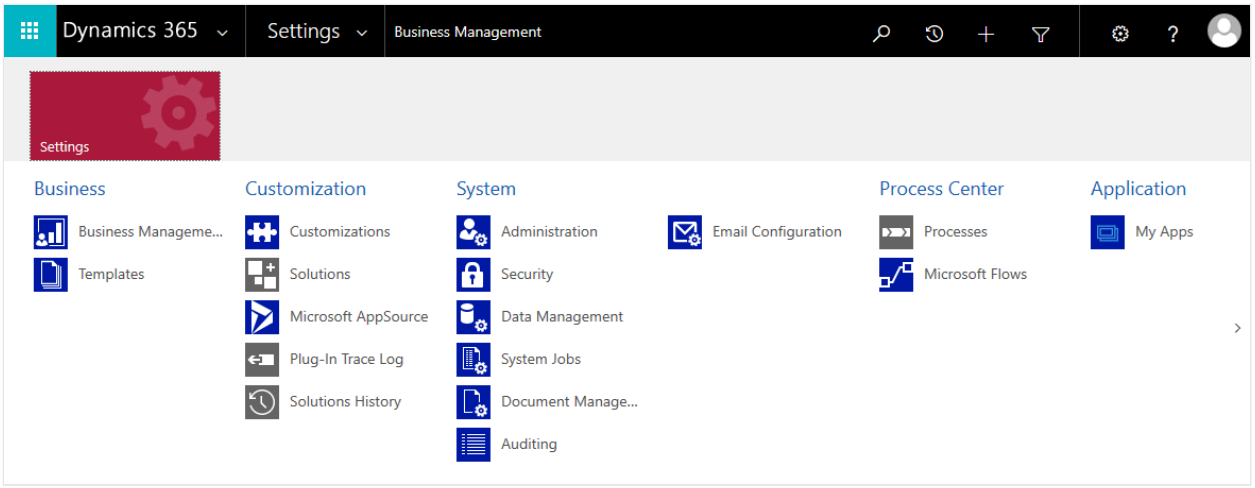
Advanced model-driven app making

The solution explorer is a comprehensive tool used for advanced model-driven app building. Within the solution explorer, you can navigate through a hierarchy that consists of all app components using the navigation pane on the left side of the tool.



To open solution explorer:

1. On the Power Apps Home page, select **Settings**, and then select **Advanced Settings**.
2. On the Dynamics 365 Business Management page, select **Settings**, select **Customizations**, and then select **Customize the System**.



More information: Advanced app making and customization

Next steps

Data model development overview

Feedback

Was this page helpful? Yes No

Provide product feedback

Define data for your model-driven app

Summarize this article for me

Model-driven apps behave according to the nature of the data model that underpins them. App users observe and interact through views of data housed within tables and the forms used to enter data into records based on those tables.

In addition to editing table views and forms, charts and dashboards are also defined at the level of the table.

Finally, business rules ensure consistent business logic is applied to table records.

While it is possible to create a table without a business rule, chart, or dashboard, it isn't possible to design a meaningful table without a view or a form.

Finally, while business process flows are highly relevant to tables and the app user experience, they aren't authored at the level of the table. Business process flows can be created and managed in Power Apps (make.powerapps.com).

The process of designing an app using the app designer in most instances comes *after* the data model is prepared. However, an iterative approach is possible using the platform.

How to define a data model

The data model for a model-driven app is defined in Microsoft Dataverse. Power Apps model-driven apps can only be defined using Dataverse.

The apps data-model is defined using the components: *table*, *column*, and *relationship*. Once these are created, then the forms and views can be developed further to meet the needs of the organization.

For detailed information about working with these components in Dataverse to define data for your model-driven app, go to the following articles:

Expand table

Component	Article	Note
Table	Work with tables	
Column	Work with columns	
Relationships	Work with relationships	

Component	Article	Note
Business rule	Create a business rule	
Chart	Create a chart	
Dashboard	Build a dashboard	
Business process flow	Create a business process flow	Uses Power Automate designer

Next steps

Learn about common column properties

Learn about configuring tables

Learn about creating and designing forms

Discover out more about views here

Build your first app

Last updated on 02/13/2026

Model-driven app common column properties

Summarize this article for me

Common properties of table columns for a model-driven app can be viewed using Power Apps (make.powerapps.com). Power Apps provides an easy way to create and edit table columns with Microsoft Dataverse.

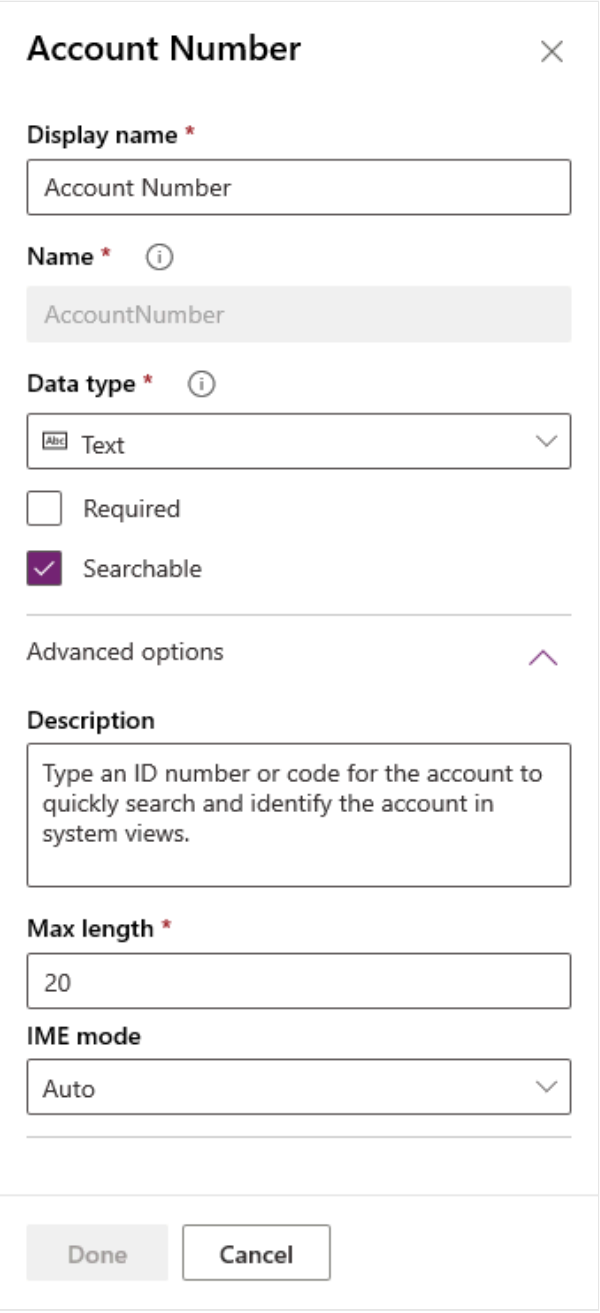
In addition to configuring aspects of the Dataverse column, such as its type and whether or not it's required, it's possible to configure aspects of how the column is used in the context of a form. This approach allows for different forms to permit users to have different interactions with the data column.

Common column properties in Power Apps

1. From Power Apps make.powerapps.com, select **Tables**, and then select the table that has the columns you want to view. If the item isn't in the side panel pane, select ...More and then select the item you want.

   > **Note**
   >
   > Customizations to a table take should place within a solution. To update a table within a solution, open your **solution**, open the **table**, and then select the **Columns** area.

2. Select the column that you want to view.



The following table describes the common properties of columns. Certain types of columns have special properties. These are described in Create and edit columns for Dataverse.

Expand table

Property	Description
Display Name	The text to display for the column in the user interface.
Name	The unique name across your environment. Power Apps generates a name based on the display name that you enter. However, you can edit this name before saving. Once you create a column, you can't change the name as your applications or code might reference it. The **Dataverse Default Publisher** customization prefix is prepended to the name.
Data type	Controls how values are stored and how they're formatted in some applications. Once you save a column, you can't change the data type except for converting text columns to

Property	Description
	autonumber columns.
Required	A row can't be saved without data in this column.
Searchable	This column appears in Advanced Find and is available when customizing views.
Calculated or Rollup	Use to automate manual calculations. Use values, dates, or text.
Advanced Options	Add a description, and specify a maximum length and IME mode for the column.

There are many different types of columns, but you can only create some of them. For more information about all types of columns, see Types of columns and column data types.

Common column properties in solution explorer (classic)

In addition to using Power Apps (make.powerapps.com), you can also review columns by using the solution explorer.

Open an unmanaged solution

1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. In the **All Solutions** list open the unmanaged solution you want.

Navigate to a column for a table using the solution explorer

To update the column, follow these steps.

1. Sign in to Power Apps.

2. On the left navigation pane, select **Tables**. If the item isn't in the side panel pane, select ... More and then select the item you want.

   > **Note**

Customizations to a table should take place within a solution. To update a column within a solution, open your **Solution**, and then open the **Table**.

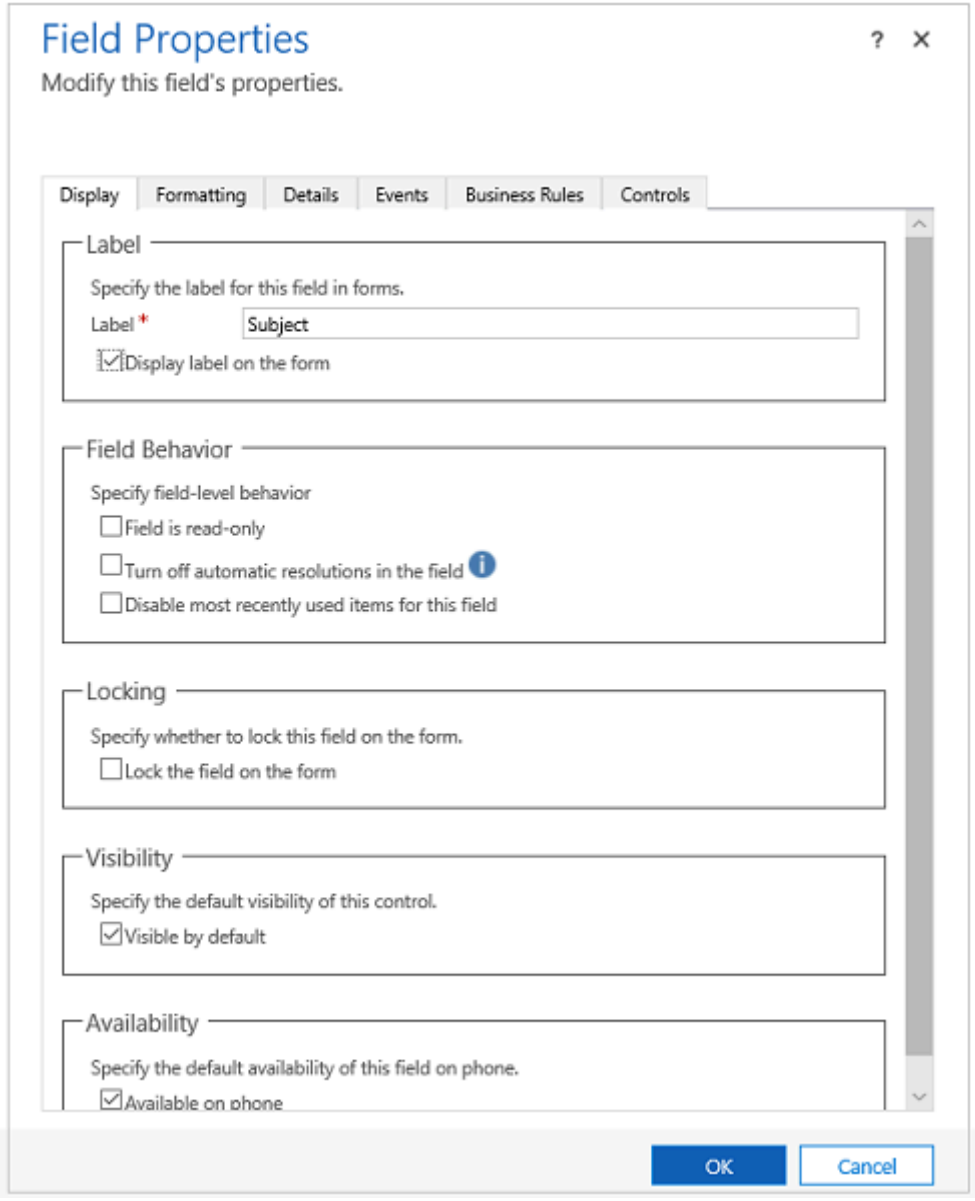
3. Select a table, such as the account table.

4. Select **...** on the command bar, and then select **Switch to classic**.

> **Note**
>
> In the solution explorer, tables are known as entities and columns are known as fields.

5. Select the column that you want.

6. Select **Change properties** from the command bar, or double-click the column.



7. Make the necessary property changes (see the classic editor field properties options), and then select **OK**.

8. **Save** and **Publish** the form.

Field properties within the classic editor

There are six tabs shown for a field. **Display**, **Formatting**, **Details**, **Events**, **Business Rules and Components**.

The following table describes properties that all columns have. Certain types of columns have special properties. These are described in Special column properties.

Expand table

Tab	Property	Description
Display	**Label**	**Required**: By default the label will match the display name of the column. The name for the form can be overridden by entering a different label here.
	Display label on the form	This contains the option not to display the label at all.
	Field Behavior	Specify the column level behavior using the check boxes.
	Locking	This will prevent the column from being removed from the form accidentally. This will prevent any configuration applied to the column, such as event handlers, from being cleared if the column were removed. To remove this column, a customizer would need to clear this setting first.
	Visibility	Showing the column is optional and can be controlled using scripts. More information: Visibility options
	Availability	Choose if you want the field to be available on the phone.
Formatting	**Select the number of fields the control occupies**	When the section containing the columns has more than one column this can set the column to occupy up to the number of columns that the section has.
Details	**Display Name**, **Name**, and **Description**	These read-only columns are for reference. Select **Edit** for access to the column definition if you want to edit it. Each instance of a column in the form has a name property so that they can be referenced in form scripts, but this name is managed by the application. The first instance of the column is the name of the column specified when it was created. More information: Create and

Tab	Property	Description
		edit columns For each additional time that a column is included in a form, the name appends a number starting with 1 to the end. So if the column name is 'new_cost', the first instance is 'new_cost', the second is 'new_cost1', and so on, for each instance of the column in the form. **Note:** The column **Description** value provides tooltip text for the column when people place their cursor over it.
Events	**Form Libraries**	Specify any JavaScript web resources that will be used in the column `OnChange` event handler.
	Event Handlers	Configure the functions from the form libraries that should be called for the column `OnChange` event. More information: Configure Event Handlers
Business Rules	**Business Rules**	View and manage any business rules that reference this column. More information: Create business rules and recommendations
Controls	**Controls**	Add controls and specify their availability on **Web**, **Phone**, and **Tablet**.

Editing form level column properties

In some scenarios, it is desirable to render a data column within a form field in one form in a way that differs in the way it is presented in another. This allows the user experience to be tailored to the individual using the form.

1. Sign in to Power Apps.
2. On the left navigation pane, select **Tables**. If the item isn't in the side panel pane, select ... More and then select the item you want.

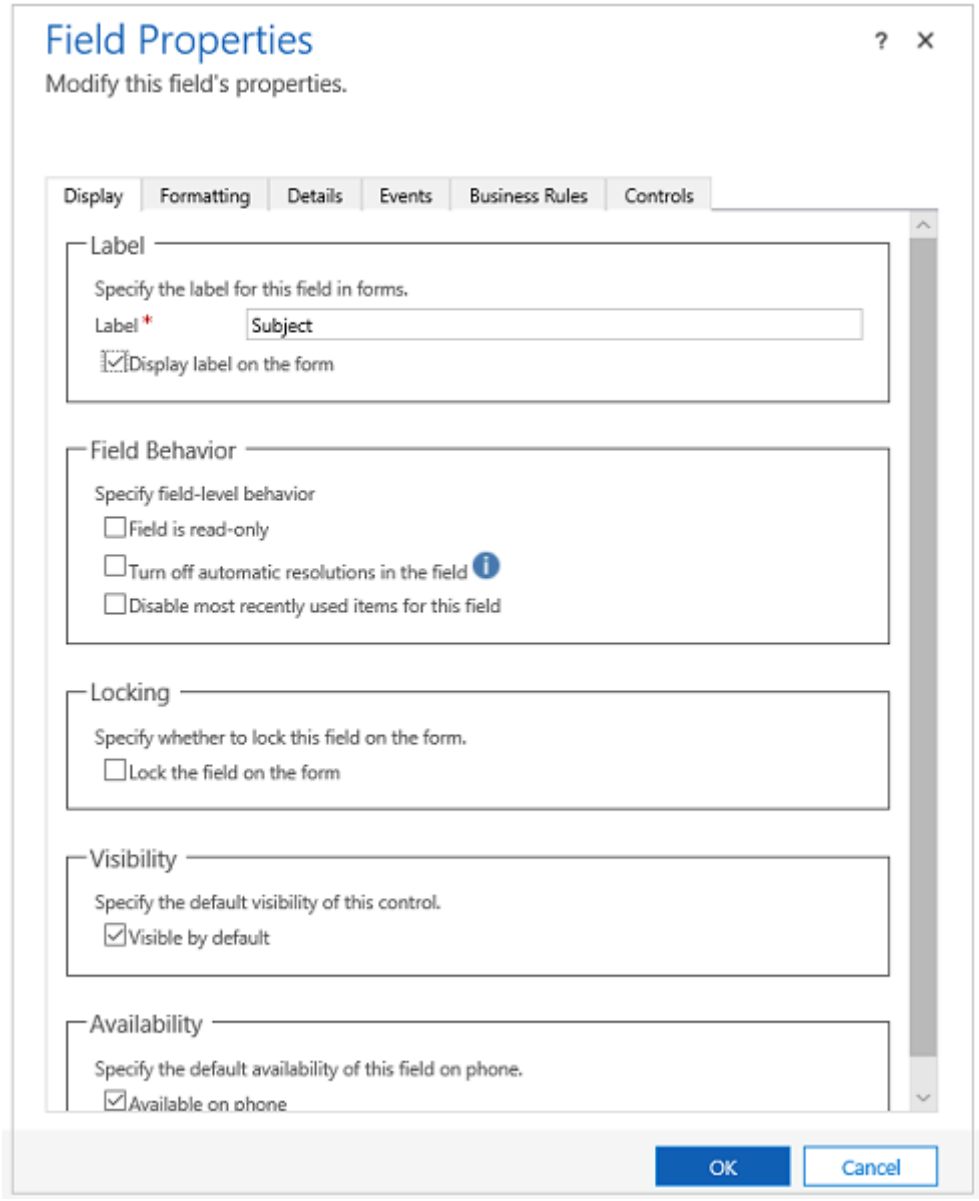
   > **Note**
   >
   > Customizations to a table should take place within a **solution**. To update a form within a solution, first navigate to your **Solution**, open the **Table**, and then select the **Forms** area.

3. Select a table, such as the account table, and then select the **Forms** area.
4. Select the form where customizations are required.

5. Select **Switch to classic**. The form opens in the classic experience.

6. Select the form field where an updated experience is required.

7. Select **Change properties** from the command bar or double-click the field.



8. Make the necessary changes, and then select **OK**.
9. **Save** and **Publish** the form.

Next steps

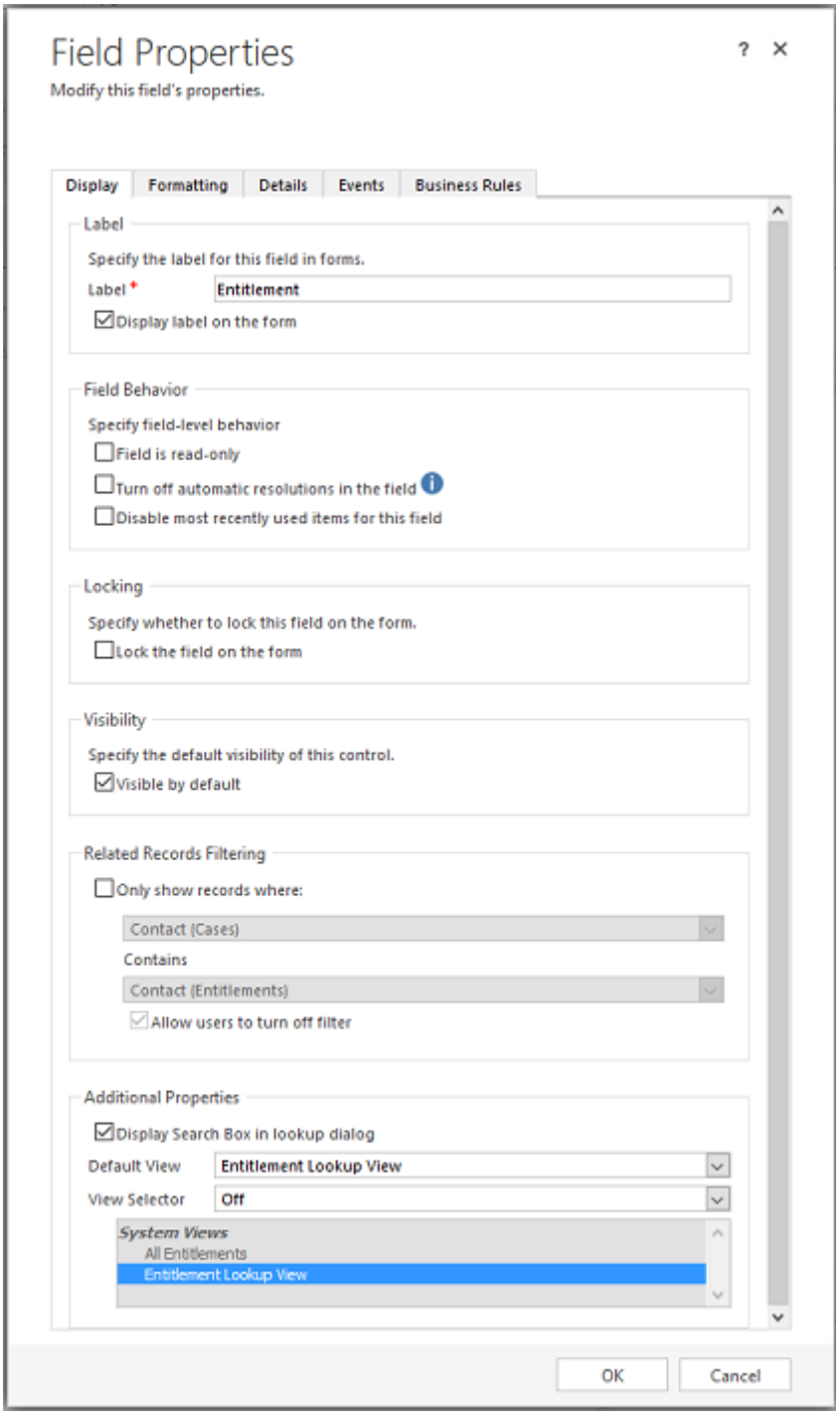
Overview of special column properties

Last updated on 03/04/2026

Overview of model-driven app special column properties

Summarize this article for me

All columns have the properties listed in Common column properties, but certain columns have additional properties, such as this entitlement column that can be opened from the main form for the case table.



> **Note**

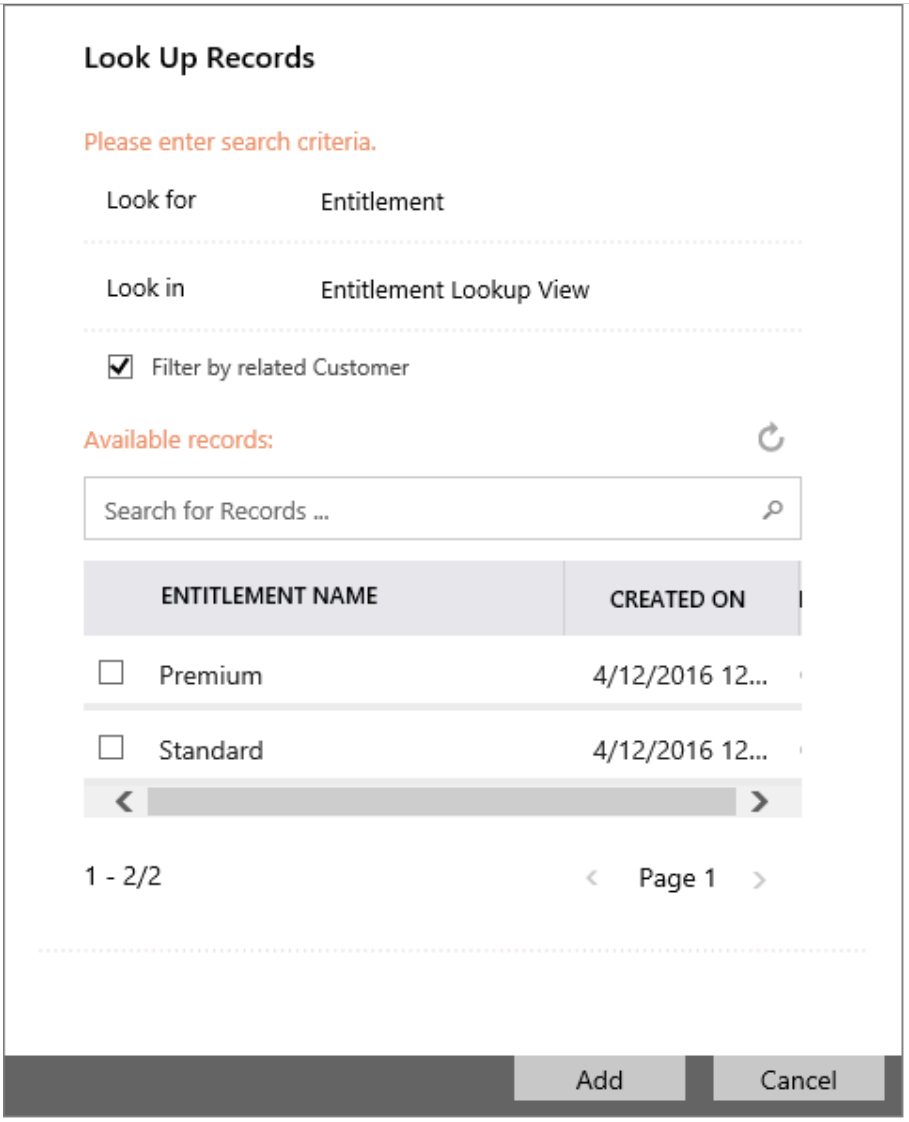
This article describes properties for the classic experience. For information about lookup column properties in the modern for designer, go to Configure a lookup component on a form.

Lookup column properties

> **Note**
>
> The options described in the table below are available only for single-table lookup columns.

Expand table

Section	Property	Description
Related Records Filtering	**Only show records where**	When this is enabled, the records that display when users search for a record will have additional filtering applied. This helps provide more relevant searches when setting the value of the lookup. By default, this is turned off. The relationship combinations that are possible when you filter related records are listed in the table following this one. The first list is populated with all the potential relationships you can use to filter this lookup. Select one. The second list is then populated with all relationships that connect the related table (selected in first list) to the target table. Select one. Select the **Allow users to turn off filter** check box to give users the option to turn off the filter you define here. When users select the **Look Up More Records** option while setting the value for a lookup, they see this dialog box.

Section	Property	Description
		 If you've selected the **Allow users to turn off filter** option while configuring the lookup column, users will see the check box to turn off the filter. This makes it possible for them to see a wider range of records. If you want to make sure that users only see a limited range of records defined by this filter, clear the **Allow users to turn off filter** check box.
Additional Properties	**Display Search Box in lookup dialog**	You can choose not to display the search box in the lookup dialog.
	Default View	This view is used to filter the results of the inline search and set the default view shown in the lookup dialog when users select the **Look Up More Records** option. The default view also controls which columns are included in the inline lookup. For lookups that only allow selection of a single table type, the columns displayed in the inline lookup are set to be the first two columns

Section	Property	Description
		included in the default view. In this example, **Main Phone** and **Email** are the first two columns in the default view configured for an account lookup. For system lookups that allow for multiple table types, the first two columns of the table lookup view are shown.
	View Selector	You can choose from three options: - **Off**: Don't allow users to choose a different view. - **Show All Views**: All views are available. - **Show Selected Views**: When you choose this option you can use the Ctrl key and your cursor to choose which views to show. The Lookup view for the table can't be de-selected.

Possible relationship combinations

Expand table

First list relationship	Second list relationship	Available?
N:1	1:N	Yes
N:1	N:1	Yes
N:1	N:N	Yes
1:N	1:N	Yes
1:N	N:1	No
1:N	N:N	No
N:N	1:N	Yes
N:N	N:1	No
N:N	N:N	No

Yes/No column properties

On the formatting tab, Yes/No columns have the following formatting choices:

- **Two radio buttons**: Two labeled controls with labels. Only one may be selected.
- **Checkbox**: A single checkbox to set the true value, otherwise false.

- **List**: A drop-down list containing both values.

Multiple lines of text column properties

Multiple lines of text and single line of text columns that use the `Text Area` format have a **Record Layout** property. With this property you can specify a value for **Number of Records** or select **Automatically expand to use available space**.

Last updated on 02/12/2026

Translate localizable text for model-driven apps

Summarize this article for me

If you customize table or column text, such as column labels or drop-down list values, you can provide users in your environment who aren't working with the base language version of your environment with this customized text in their preferred languages.

The process has the following steps:

1. Enable other languages for your environment
2. Export the localizable text
3. Get the localizable text translated
4. Import the localized text

Enable other languages for your environment

If you haven't already enabled the languages for your environment, use the steps described in Enable the language to enable them.

> **Important**
>
> Each language can take several minutes to enable. During this time, other users of the environment might not be able to use your app. Enable languages at a time that is least disruptive to users.

> **Tip**
>
> While you enable the languages, note the LCID values used for each language. This value represents the language in the exported data for the localizable text. Language codes are four-digit or five-digit locale IDs. Valid locale ID values can be found at Locale ID (LCID) Chart).

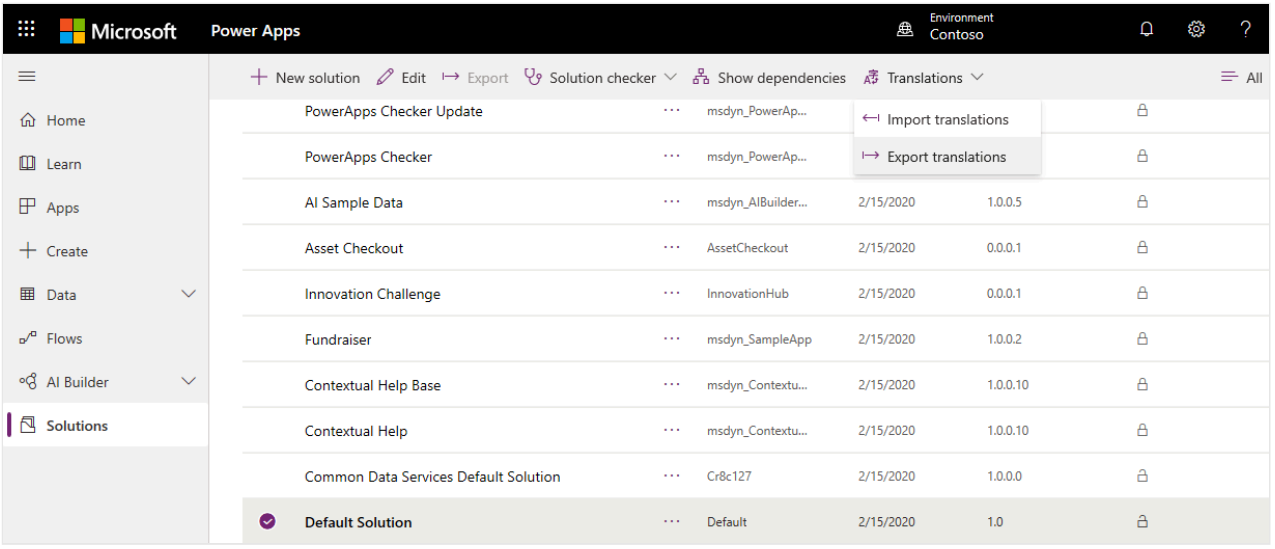
Export the localizable text

The localizable text that you export comes from the unmanaged solution that contains the localizable text.

1. From Power Apps, select **Solutions**. If the item isn't in the side panel pane, select ... More and then select the item you want.

2. In the **All Solutions** list, select the unmanaged solution that contains the localizable text you want.

3. On the command bar, select **Translations** > **Export Translations**.



You could see an alert:

> Exporting customized labels for translation can take several minutes. Do not click the export link again until the first export has finished. Are you sure that you want to export now?
>
> Select **OK** if you want to continue.

When the export is completed, save the translations zip file. The file is named something like `CrmTranslations_{0}_{1}.zip`, where `{0}` is the unique name of the solution and `{1}` is the version number of the solution.

Get the localizable text translated

You can send this file to a linguistic expert, translation agency, or localization firm.

If you have the knowledge to translate the text, or if you just want to see the format, you can extract the zip file that you exported you see that it contains two XML files.

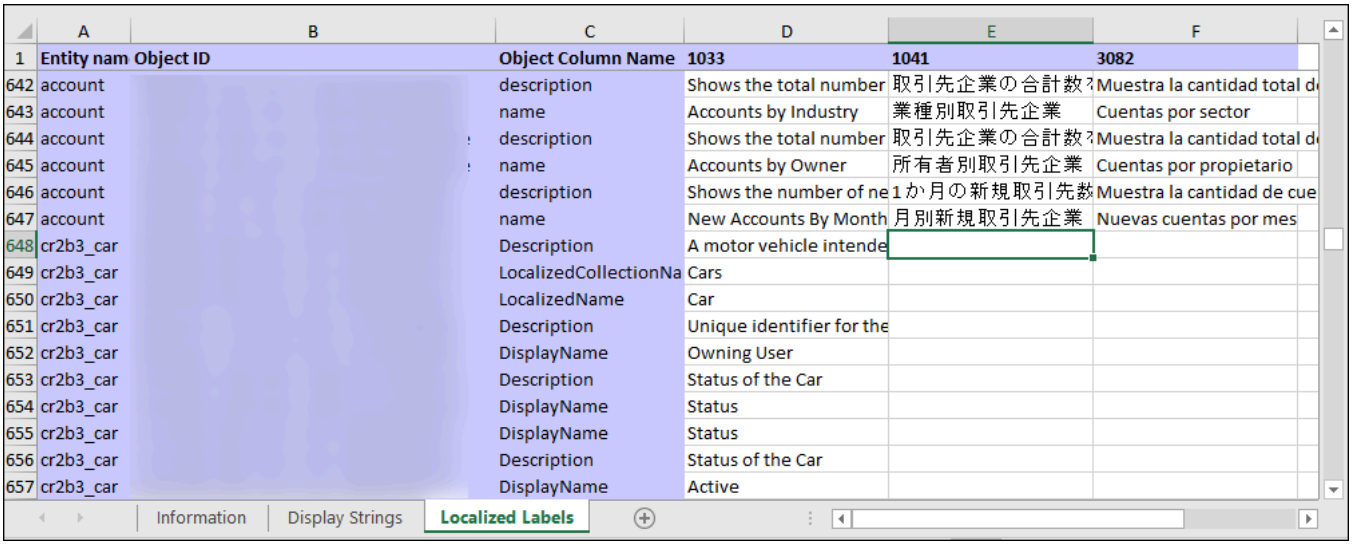
- `[Content_Types].xml`
- `CrmTranslations.xml`

You can open the CrmTranslations.xml file with Microsoft Office Excel.

> **Tip**
>
> Unless you normally open XML files with Excel, it might be easier to open Excel, then choose to open the file by pasting in the path to the extracted CrmTranslations.xml file.

> **Important**
>
> Don't change the file format. If you save the file in another format, you can't import it back.

When you view the data in Excel, look at the **Localized Labels** tab.



	A	B	C	D	E	F
1	Entity nam	Object ID	Object Column Name	1033	1041	3082
642	account		description	Shows the total number	取引先企業の合計数	Muestra la cantidad total d
643	account		name	Accounts by Industry	業種別取引先企業	Cuentas por sector
644	account		description	Shows the total number	取引先企業の合計数	Muestra la cantidad total d
645	account		name	Accounts by Owner	所有者別取引先企業	Cuentas por propietario
646	account		description	Shows the number of ne	1か月の新規取引先数	Muestra la cantidad de cue
647	account		name	New Accounts By Month	月別新規取引先企業	Nuevas cuentas por mes
648	cr2b3_car		Description	A motor vehicle intende		
649	cr2b3_car		LocalizedCollectionNa	Cars		
650	cr2b3_car		LocalizedName	Car		
651	cr2b3_car		Description	Unique identifier for the		
652	cr2b3_car		DisplayName	Owning User		
653	cr2b3_car		Description	Status of the Car		
654	cr2b3_car		DisplayName	Status		
655	cr2b3_car		DisplayName	Status		
656	cr2b3_car		Description	Status of the Car		
657	cr2b3_car		DisplayName	Active		

Information | Display Strings | **Localized Labels**

Any custom tables or columns have empty cells for the localizable text. Add the localized values for those items.

> **Note**
>
> If you have changed the display name or description for any standard table or table column, the localized strings will still reflect the translations for the original value. Those should be localized to reflect the new value.

The **Display Strings** tab contains text that is displayed for other UI elements such as ribbon commands, error messages, and form labels.

Update localizable text in the base language

If you change the display name for any standard table or table column, which is included in any special message, update the information in the **Display Strings** tab to use the customized

name.

> **Tip**
>
> Although the UI exposed to edit system table messages includes many references to table names, it doesn't include all of them. Using this technique might find more. More information: Edit system table messages

For example, if you change the display name for the Account table to *Company*, search through the base language column in the **Display Strings** for the following matches: `account`, `accounts`, `Account`, and `Accounts` then make appropriate replacements to `company`, `companies`, `Company`, and `Companies` respectively.

> **Important**
>
> Don't do a general find and replace in the file for this. Make sure that the matching text actually refers to the names that you changed.

Import the localized text

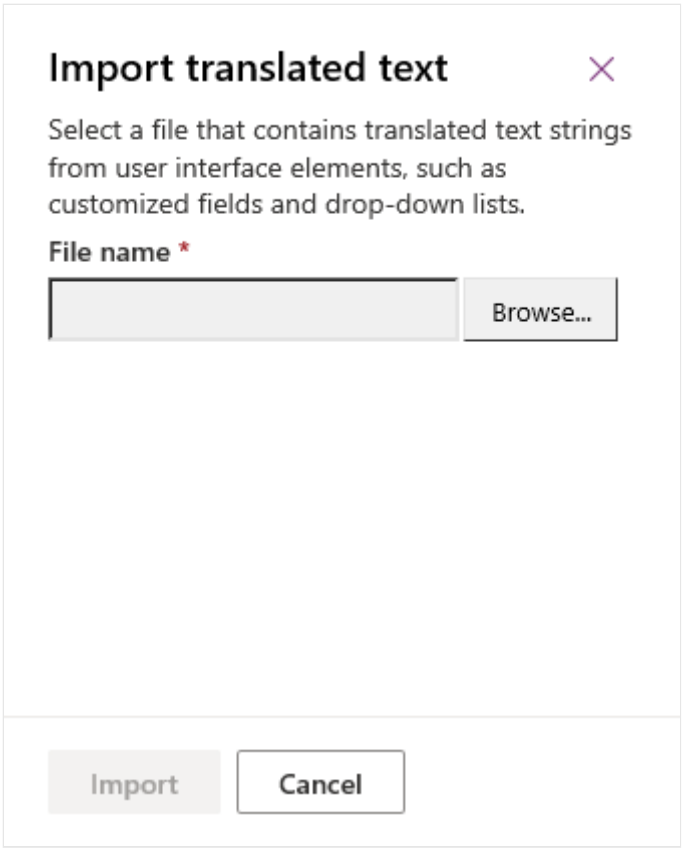
Importing the text requires compressing the files and importing them into the system.

Compress the files

After you make changes to the `CrmTranslations.xml` file, compress the file together with the `[Content_Types].xml` file into the zip format. Select *both files* and then right-click to open the context menu. In the context menu, choose **Send to** > **Compressed (zipped) folder**.

Import the files

From the same unmanaged solution that you exported the translations from, in the menu choose **Translations** > **Import Translations**.



Select the file that contains the compressed translated text and select **Import**.

After you import the translated text, publish all customizations to see the changes in your apps.

Next steps

Regional and language options for your organization

Edit system table messages

Last updated on 03/04/2026

Model-driven app interface design overview

Article • 04/23/2025

Model-driven app interface design includes working with table views and forms. To work with these components in your app, you create pages.

Creating pages in a model-driven app builds the app site-map that describes the navigation experience within the app.

App interface design can include changing the command bar menu in an app.

This article mentions the classic approach to developing apps. This includes the creation of apps in addition to working with forms, views, and other elements.

Model-driven app design

Here's an overview of the model-driven app design process.

Designing a model-driven app requires a range of skills. There are many design tools that enable a model-driven app to be created. At times the app maker plays the role of a database designer, a user experience customizer, and even a process designer.

Model-driven apps are essentially a selection of Microsoft Dataverse components (tables, charts, dashboards) that work alongside each other to achieve a business outcome. These components are delivered in conjunction with the Dataverse security roles to create a user experience that meets the needs of members of different parts of an organization.

This means that the same app can be used by colleagues in different departments. In some cases, users only want to review data and might only have interest in specific areas in order to complete their daily objectives.

The design of tables and table columns is covered under a documentation dedicated to this article. Learn more about creating Dataverse tables.

App interface design resources

The following table describes the resources in this section.

Expand table

Title	Detail	Link
Use the modern app designer	An overview of the most recent experience for building model-driven apps.	Learn more
Use the classic app designer	How to create an app using the classic method.	Learn more
Working with views	How to create or edit table views.	Learn more
Working with forms	How to create or edit forms.	Learn more
Use custom pages with model-driven apps	How to understand and work with custom pages.	Learn more
Customize app commands	Discover how to customize the command bar menu located near the top of a model-driven app.	Learn more

See also

Learn more about Dataverse security roles

Overview of the model-driven app designer

Article • 01/27/2025

The model-driven app designer provides a modern WYSIWYG authoring experience when you work with model-driven apps.

Changes to the app are instantly reflected in the preview, enabling you to see exactly how the app will appear to users when published.

The app designer interface has the following areas:

1. Command bar – Displays the available actions:

    - **Back**. Closes the model-driven app designer and returns you to the Power Apps website (make.powerapps.com).
    - **Add page**. Creates a new page, which can include table forms and views, table dashboards, or a custom page for the app.
    - **Settings**. Opens the app properties such as name and description.
    - **Edit form**: Open the form designer to edit the default form shown in app preview.
    - **Comments**. Add a comment to the app. More information: Add comments in the model-driven app designer
    - **Save**. Saves the app.
    - **Publish**. Makes the changes made available to other users.
    - **Play**. Opens the app in a new tab in run mode.

2. App preview – Displays a real-time preview of the form as it will appear to users when published.

3. Panes - The left navigation pane consists of the following areas:

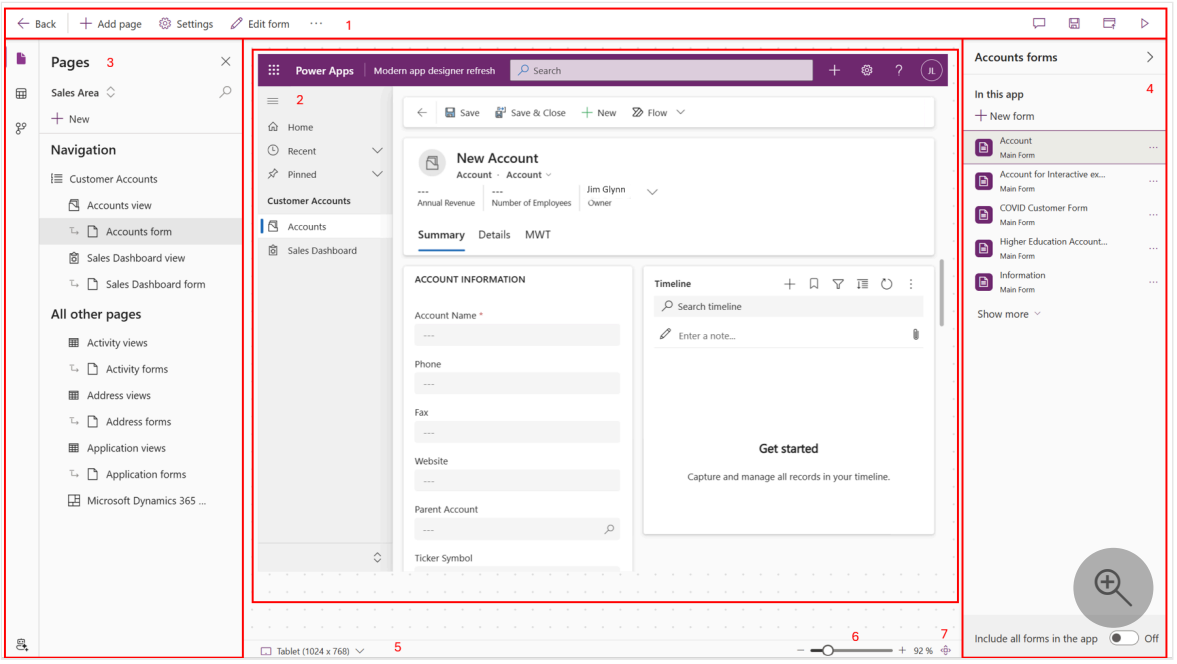
    - **Pages**. The **Navigation** section displays a layout of your app that is formed using areas, groups, and subareas. You can add or remove groups and subareas to the navigation. The **All other pages** portion displays the components in your app. From this section, you can choose to add or remove forms, views, and dashboards for each table.
    - **Data**. Provides a view of all available tables that are currently used within your app and a view of all tables that are available in your environment.
    - **Automation**. Displays business process flows that are a part of this app. You can add, remove, or create new business process flows to the app.

4. Property pane – Displays properties of the selected component and associated forms and views with the selected table. Selecting the pencil or ellipsis opens the form or view designer.

5. Preview size switcher - Changes the size of the form preview helping you to see how the form will appear on various screen sizes.

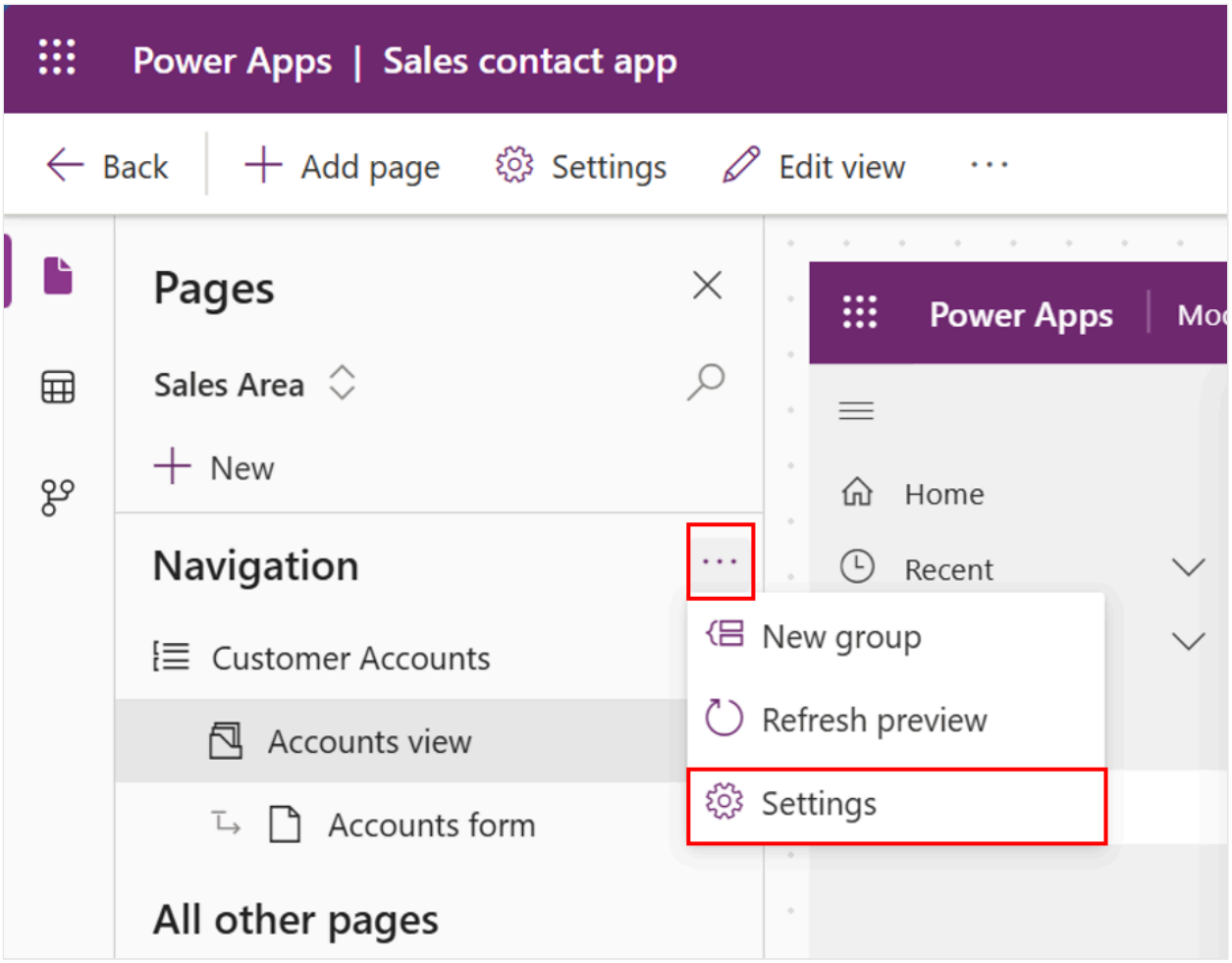
6. Zoom slider - Zooms in or out of the app preview helping you take a closer look.

7. Fit to screen - Quick action to fit the app preview to the available screen size.



Navigation pane options

To edit the app's navigation, select ... next to **Navigation**, and then select **Settings**.



The following options are available:

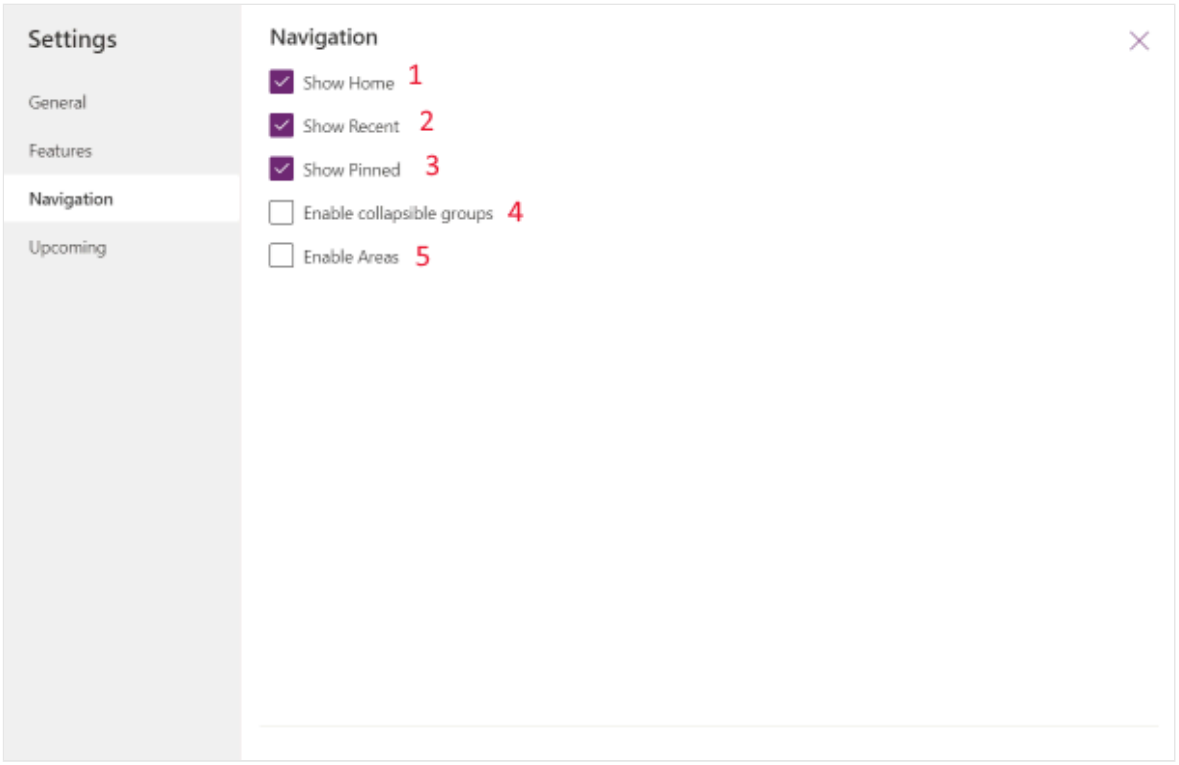
1. **Show Home**. Enabled by default. When selected, displays the **Home** page link for the app.

2. **Show Recent**. Enabled by default. When selected, displays the recently viewed pages link. Selecting the link displays all recently viewed pages.

3. **Show Pinned**. Enabled by default. When selected, displays the pages that have been pinned. App users select the push-pin icon next to a record listed under **Recent** to add it to their pinned rows.

4. **Enable collapsible groups**. Disabled by default. When selected, subareas displayed under groups in the site map can be expanded or collapsed.

5. **Enable Areas**. Disabled by default. When selected, new areas can be added to the app. For apps with existing multiple areas, this setting is enabled by default. You can't disable this setting while the app has multiple areas.



Known limitation

- The app's URL can't be specified.

Next steps

Create a model-driven app using the app designer

Working with custom pages

Understanding app navigation

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a model-driven app with the app designer

In this article, you learn the basics of how to create a model-driven app that can be shared and distributed to other environments.

Prerequisites

Verify the availability of the following prerequisites before you start to create a model-driven app:

- A Power Platform environment used for app development.
- An environment maker, system administrator, or system customizer security role.

For more information, go to the following articles:

- Create an environment
- Environment strategy for ALM
- About predefined security roles

Create an app

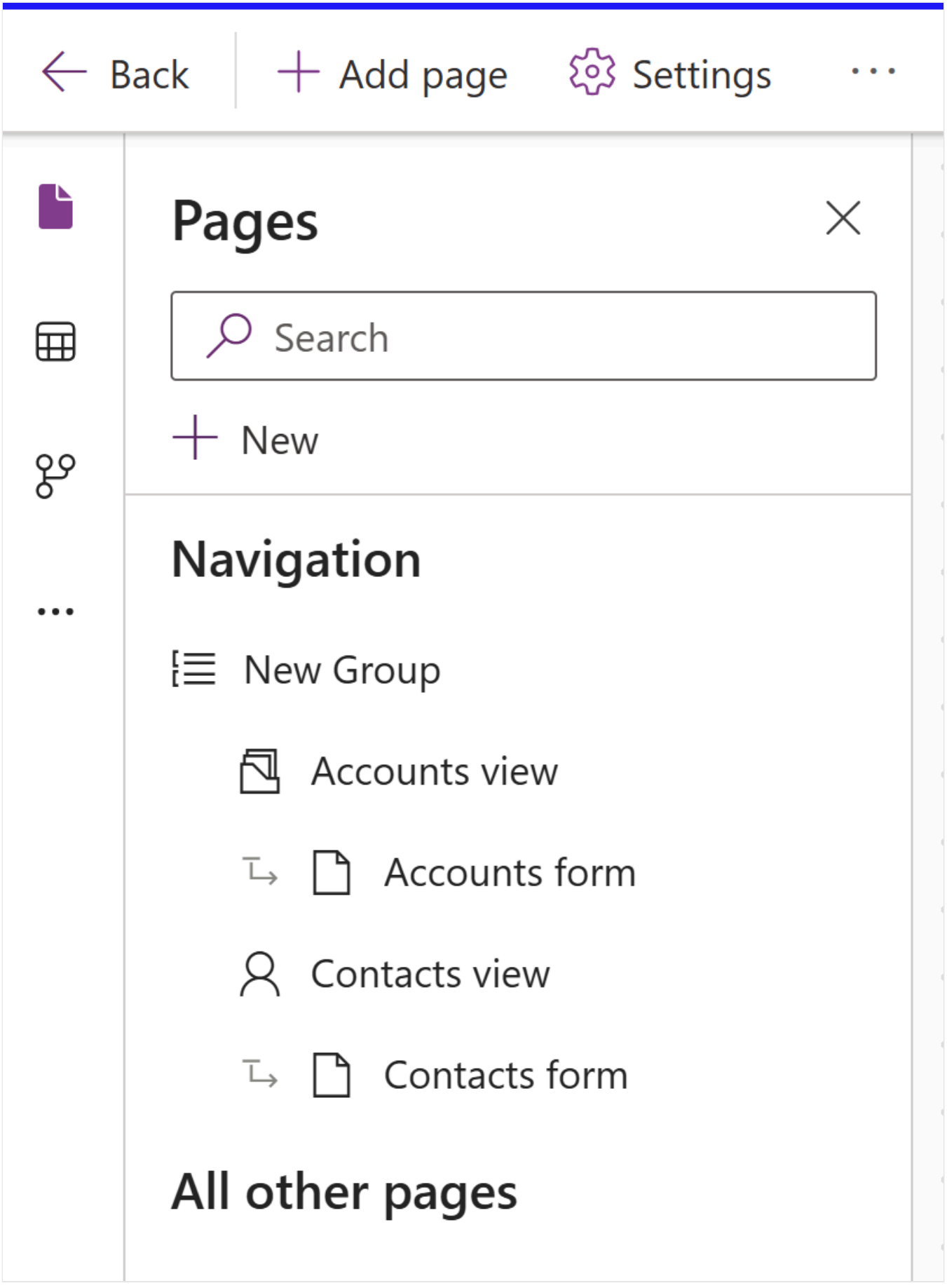
1. On the Power Apps **Home** page, select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open an unmanaged solution or create a new solution.

   > **Note**
   >
   > Unmanaged solutions are displayed when the **Unmanaged** filter is selected in the **Solutions** area.
   >
   > Although the **Default Solution** is an unmanaged solution, it is in most circumstances not recommended to create or edit customizations in the default solution.

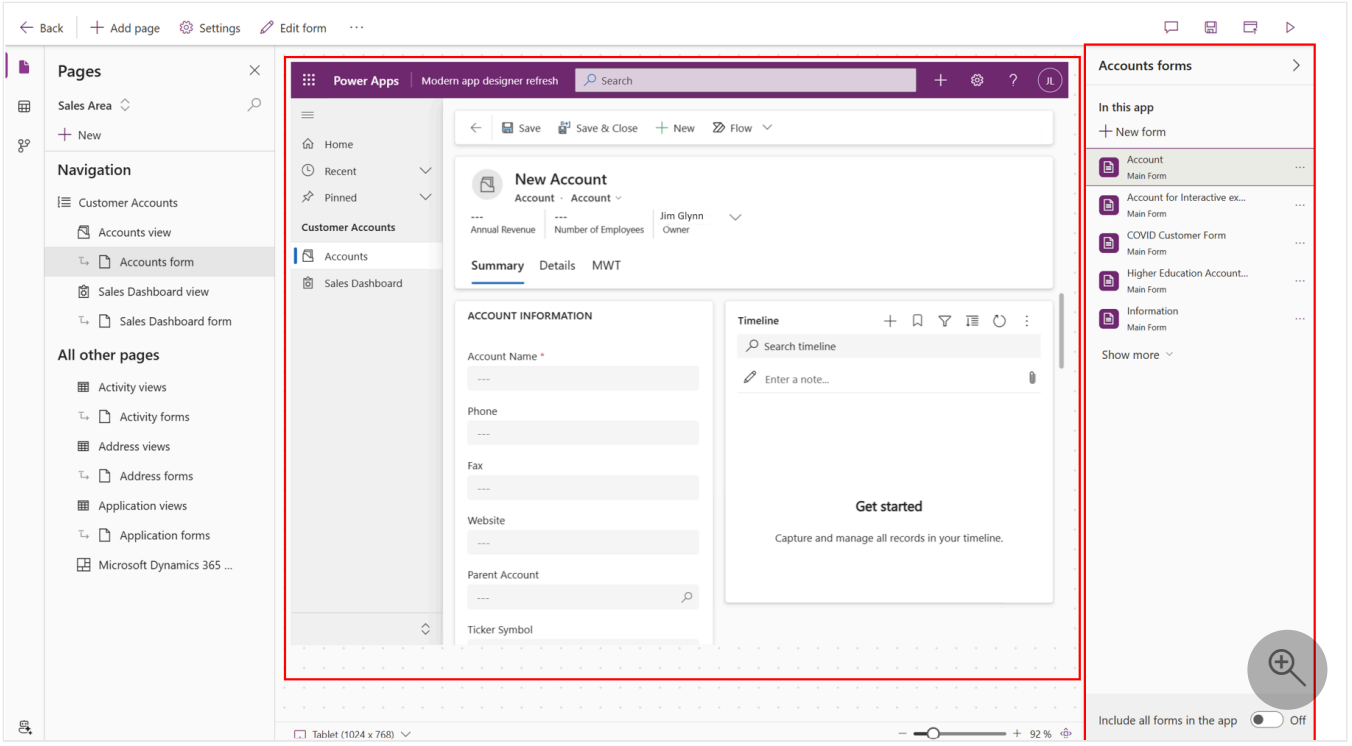
3. Select **New** > **App** > **Model-driven app**.
4. On the **New model-driven app** dialog box, enter a **Name** and optionally, a **Description**, and then select **Create**.
5. On the command bar, select **Add page** > **Dataverse table**.

6. From the list of tables, select **Account**, select **Contact**, and then select **Add**.

On the **Pages** left pane, under **Navigation** the layout for the app is displayed.



Notice the preview pane displays your app. When you select a component, such as the **Accounts form**, it affects what's displayed in the preview and property panes. If your environment includes data that you have access to view, that also appears in the preview.



To save and publish the app, select **Save and Publish**.

To see how the app runs in a full browser window, on the command bar, select **Play**.

Create an app description with Copilot

If your app is in a managed environment, you can use AI to help create a description for your app.

> **Important**
>
> - To use this capability your app must be in a managed environment.
> - This capability is powered by Azure OpenAI Service.
> - This capability is in process of rolling out, and might not be available in your region yet.
> - This capability might be subject to usage limits or capacity throttling.
> - To understand capabilities and limitations of AI-powered and Copilot features in Power Apps, go to Responsible AI FAQs for Power Apps

When you save and publish your app, on the **Publish** dialog box select **Create descriptions using AI** to replace your current description or, if your app doesn't have a description, Copilot

generates one for you.

You can also generate an app description in app designer by going to, **Settings** > **General** tab. Under **Description** select, **Create descriptions using AI**.

For information that describes the AI impact of Power Apps generate app descriptions with Copilot feature, go to FAQ for generate app descriptions with Copilot.

Enable or disable AI-generated app descriptions

The app AI-generated descriptions are managed by a setting in the Power Platform admin center.

1. Go to **Environments** in the Power Platform admin center.
2. Select an environment, and then select **Edit managed environments**.
3. Under the section titled **Enable AI-generated app descriptions**, select or clear **AI-generated app descriptions**
4. Select **Save**.

Clear the browser cache and close any active Power Apps (make.powerapps.com) browser windows. The feature is now be enabled or disabled.

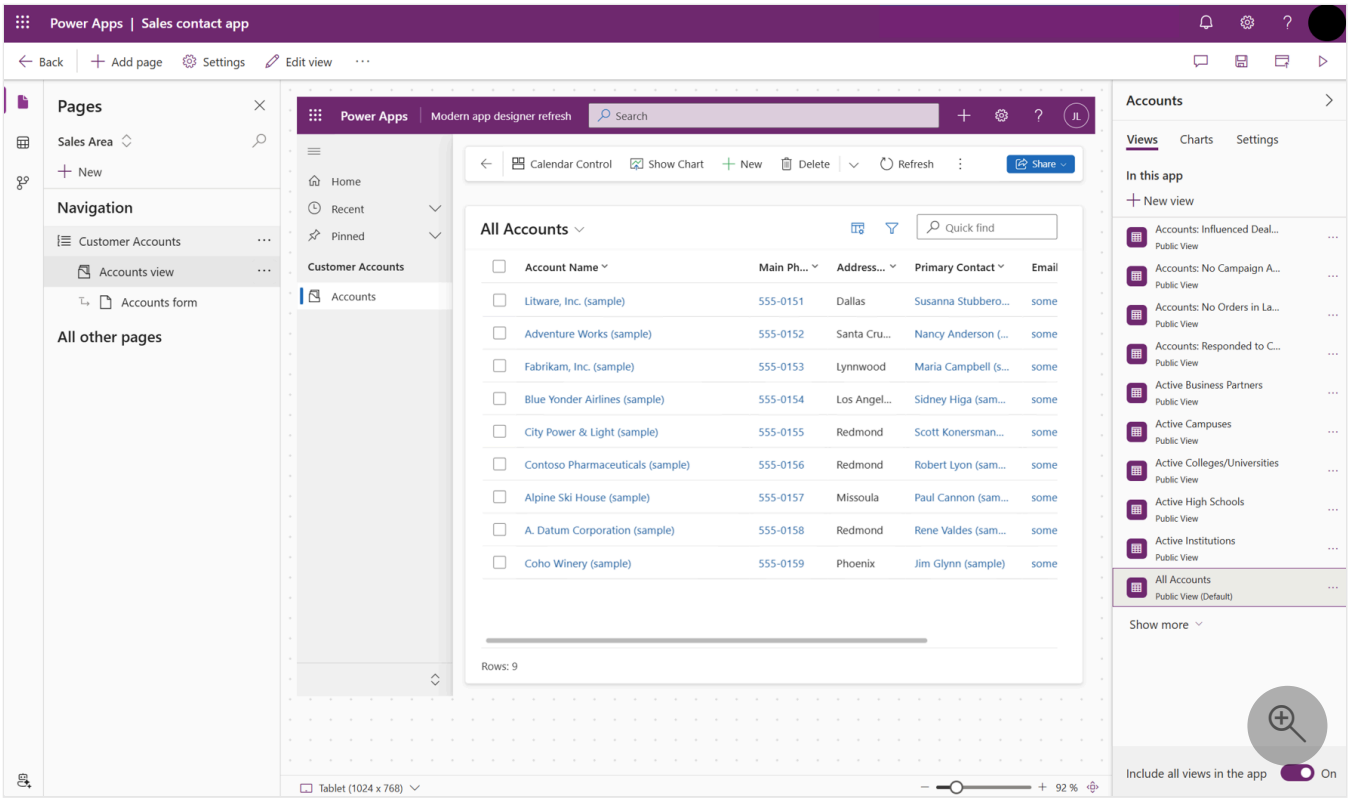
Next steps

- Add areas, groups, and subareas for app navigation
- Create, add and remove forms, views or dashboards
- Create and remove pages
- Configure app properties (Preview)
- Configure app properties

Last updated on 01/09/2026

Build your first model-driven app

In this tutorial, you create a model-driven app by using one of the standard tables that is available in Microsoft Dataverse, the account table.



1. Sign in to Power Apps. If you don't already have a Power Apps account, select the **Get started free** link.

2. Select an environment. An environment in Power Apps is a space to store, manage, and share your organization's business data, apps, chatbots, and flows. It also serves as a container to separate apps that might have different roles, security requirements, or target audiences.

   Each environment can have one Dataverse database.

   Select the environment you want. If your tenant allows you to create your own environment, select the current environment from the top right, and then select **Try it now**. More information: Get your developer environment (preview)

3. We recommend creating your model-driven app from a solution. A solution is a package that can contain Dataverse tables, forms, views, apps, flows, and other components. By building your model driven-app in a solution, you can easily move it to other environments or preserve a copy in your source control repository.

   Select **Solutions** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

4. Select **New solution**, enter a **Display name** for your solution, such as *Power Apps Training*, select a publisher (the **CDS Default Publisher** is okay for this tutorial), and then select **Create**. For more information about creating a solution, go to Create a solution.

5. In your solution, select **Add existing**, and then select **Table.**

6. In the **Add existing tables** list, select the **Account** table, and then select **Next**.

7. Since this is a standard table, leave the default setting for account table objects and metadata and select **Add.**

   Now that you have a solution and have added the account table to it, you're ready to create a model-driven app.

8. In your solution, select **New**, select **App**, and then select **Model-driven app.**

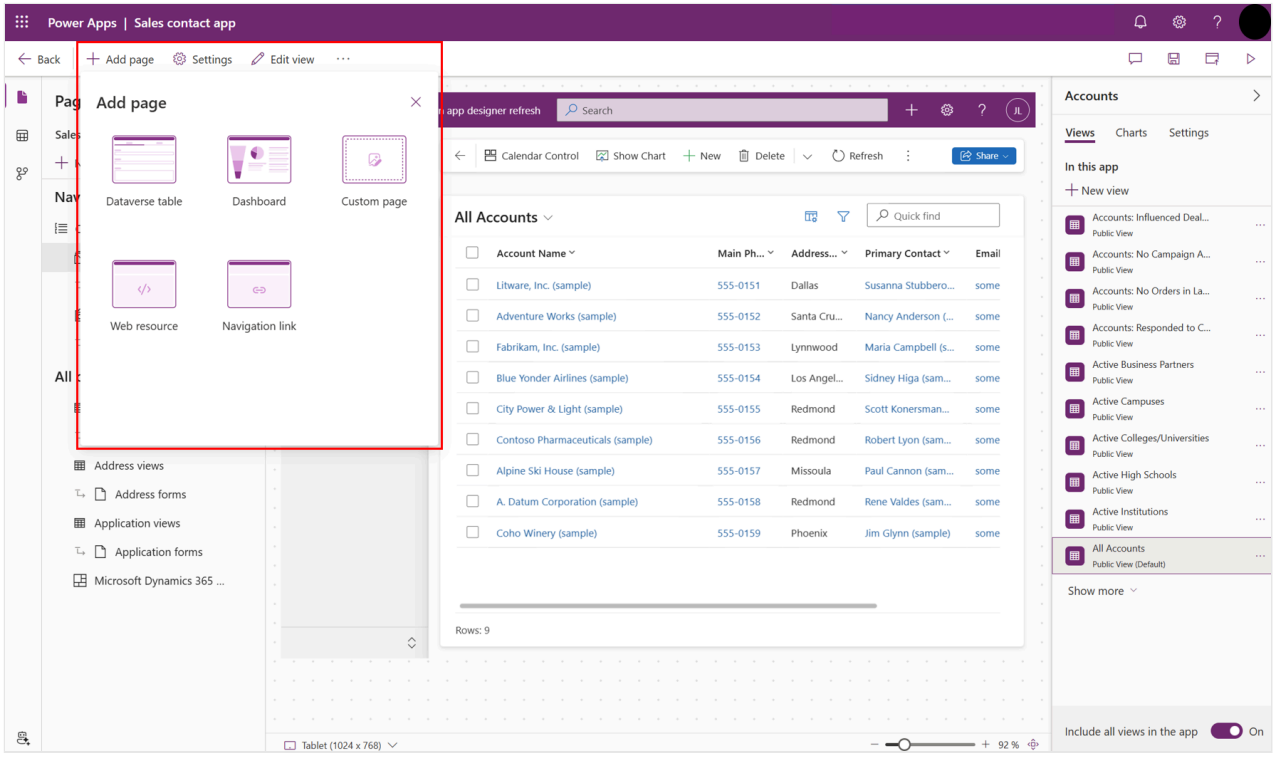
9. Enter a name for the app, such as *My new custom app*, and then select **Create**.

   Your app loads in the model-driven app designer.

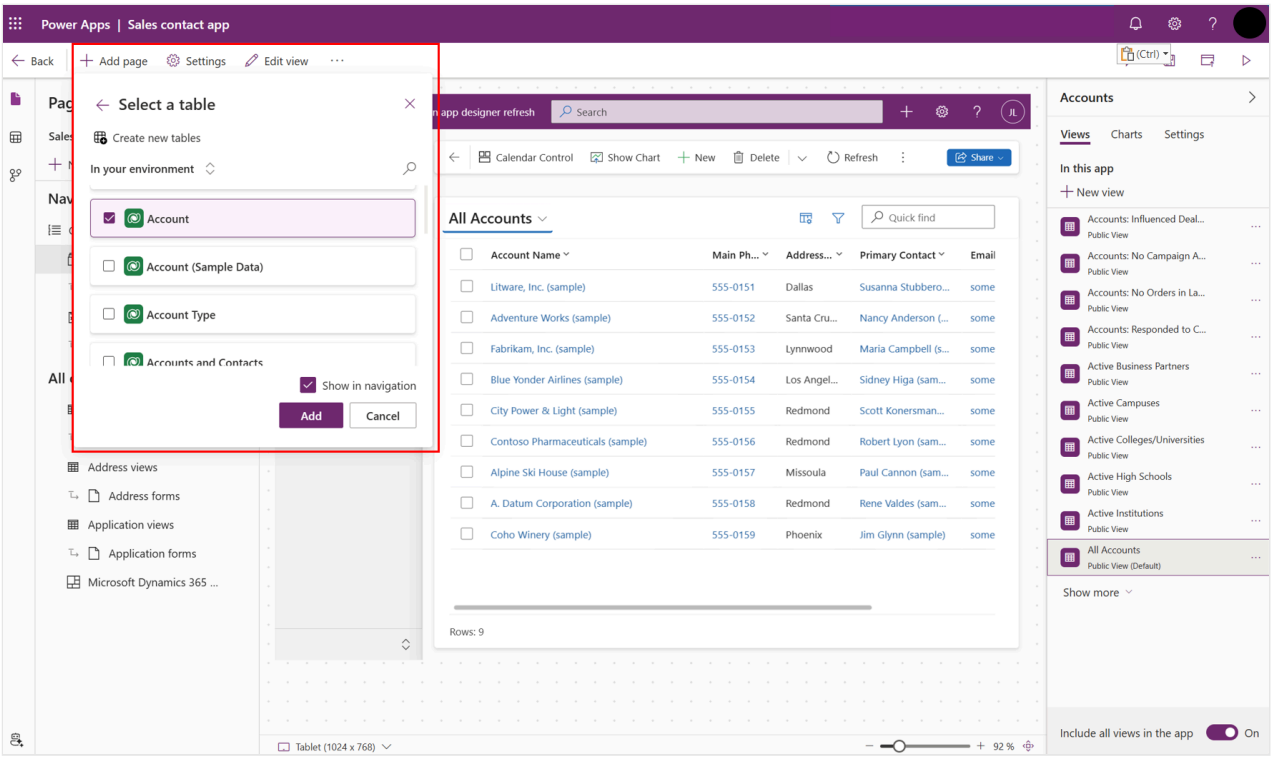
Add a page to your app

Next, you add a new page to the model-driven app in the app designer.

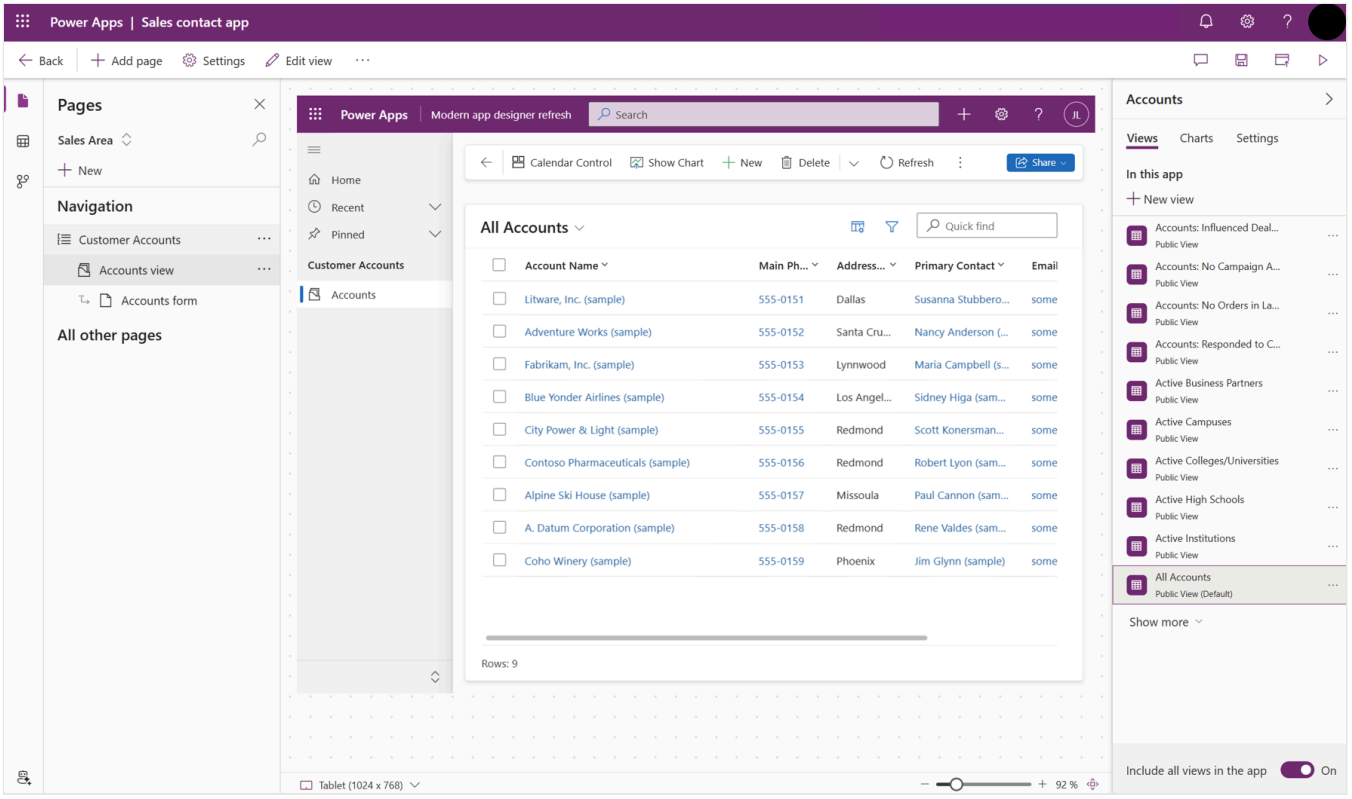
1. Select **Add page** on the command bar, and then select **Dataverse table**.



2. Select **Account**, and then select **Add**.



Your app with the account table is displayed similarly to how it appears to users at run time when published. If your environment has sample data, the sample list of active accounts might appear in the **All Accounts** view.



Save and publish your app

Save your app so that your app and changes aren't lost. To make it available to users, publish it.

To save without publishing, on the app designer command bar, select **Save**.

To save and publish, on the app designer command bar, select **Publish**.

Create an app description with Copilot

If your app is in a managed environment, you can use AI to help you create a description for your app.

> **Important**
>
> - To use this capability your app must be in a **managed environment**.
> - This capability is powered by **Azure OpenAI Service**.
> - This capability might be subject to usage limits or capacity throttling.
> - To understand capabilities and limitations of AI-powered and Copilot features in Power Apps, go to **Responsible AI FAQs for Power Apps**

When you save and publish your app, on the **Publish** dialog box select, **Create descriptions using AI** to replace your current description or, if your app doesn't have a description, Copilot generates one for you.

You can also generate an app description by going to, **Settings** > **General**. Under **Description** select, **Create descriptions using AI**.

For information that describes the AI impact of Power Apps generate app descriptions with Copilot feature, go to FAQ for generate app descriptions with Copilot.

Enable or disable AI-generated app descriptions

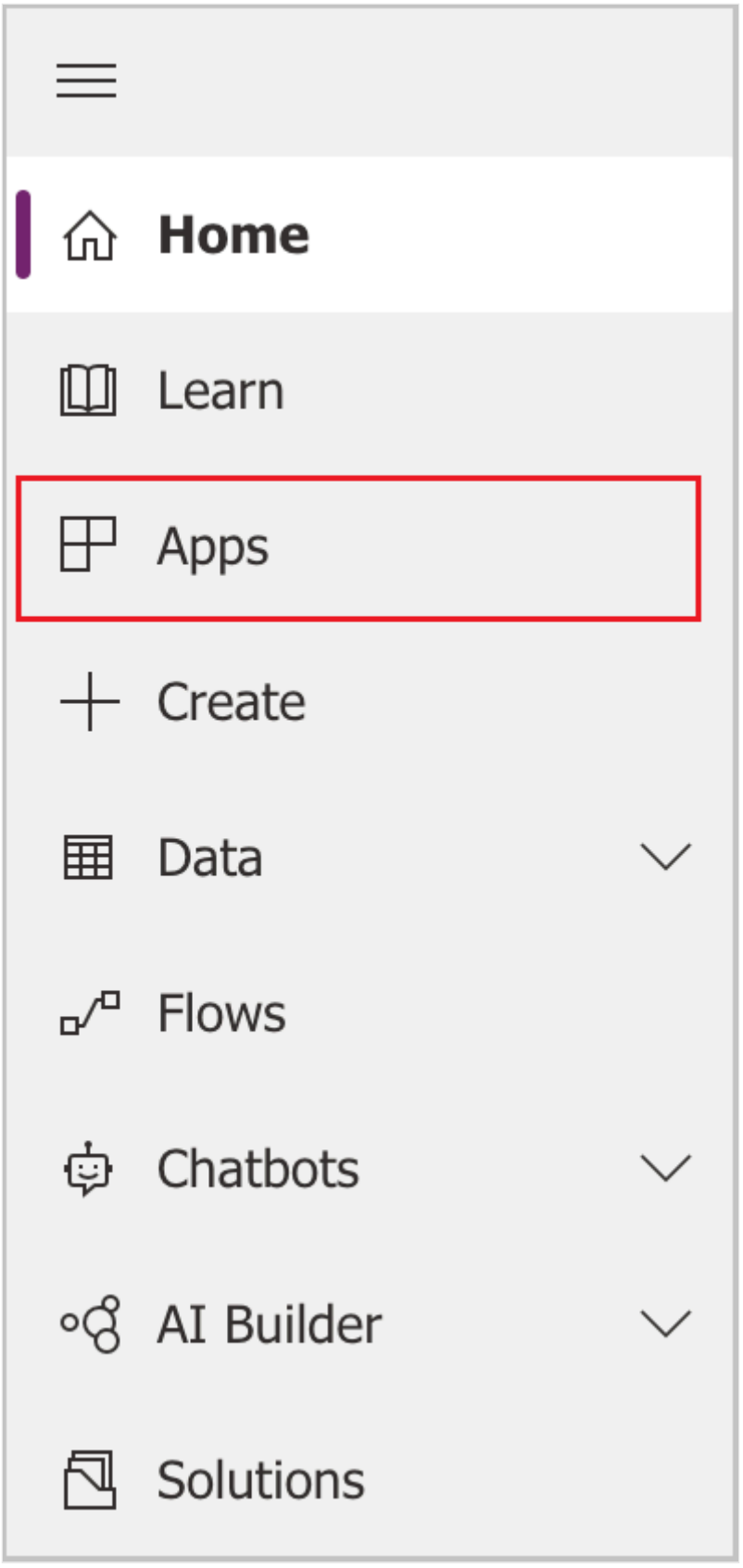
App AI-generated descriptions are managed by a setting in the Power Platform admin center.

1. Open https://admin.powerplatform.microsoft.com/environments
2. Select an environment, and then select **Edit managed environments**.
3. Under the section titled **Enable AI-generated app descriptions**, select or clear **AI-generated app descriptions**
4. Select **Save**.

Clear the browser cache and close any active Power Apps (make.powerapps.com) browser windows. The feature is now be enabled or disabled.

Run your app on a desktop computer

1. To run your app, sign in to Power Apps, and on the left navigation pane, select **Apps**.



2. Select the app from the app list. The app opens in your browser. If you want to show the chart, use the button **Show Chart**.

Run your app on mobile

To run your app on your mobile device, follow these steps:

1. Download the app from your device's mobile app store:
    - IOS: Power Apps on the App Store (apple.com)
    - Android: Power Apps - Apps on Google Play
2. Open the app, tap **Sign in**, and then enter your Microsoft work or school account.
3. Select your app from the list to run it.

Next steps

Add additional pages and configurations to make your app do more!

- Add a custom page where a canvas app runs from your model-driven app. More information: Overview of custom pages for model-driven apps
- Create Dataverse custom tables that you can add to your app the same way as you added the account table in this tutorial. More information: Create and edit tables using Power Apps
- Identify the URL of your model-driven app. More information: Run a model-driven app in a browser
- Share your app. More information Share a model-driven app.
- Learn more about building model-driven apps. More information: Understand model-driven app components

> **Tip**
>
> Ready to convert your ideas into an app? Start here: **Planning a Power Apps project**.

Last updated on 01/09/2026

App navigation in model-driven apps

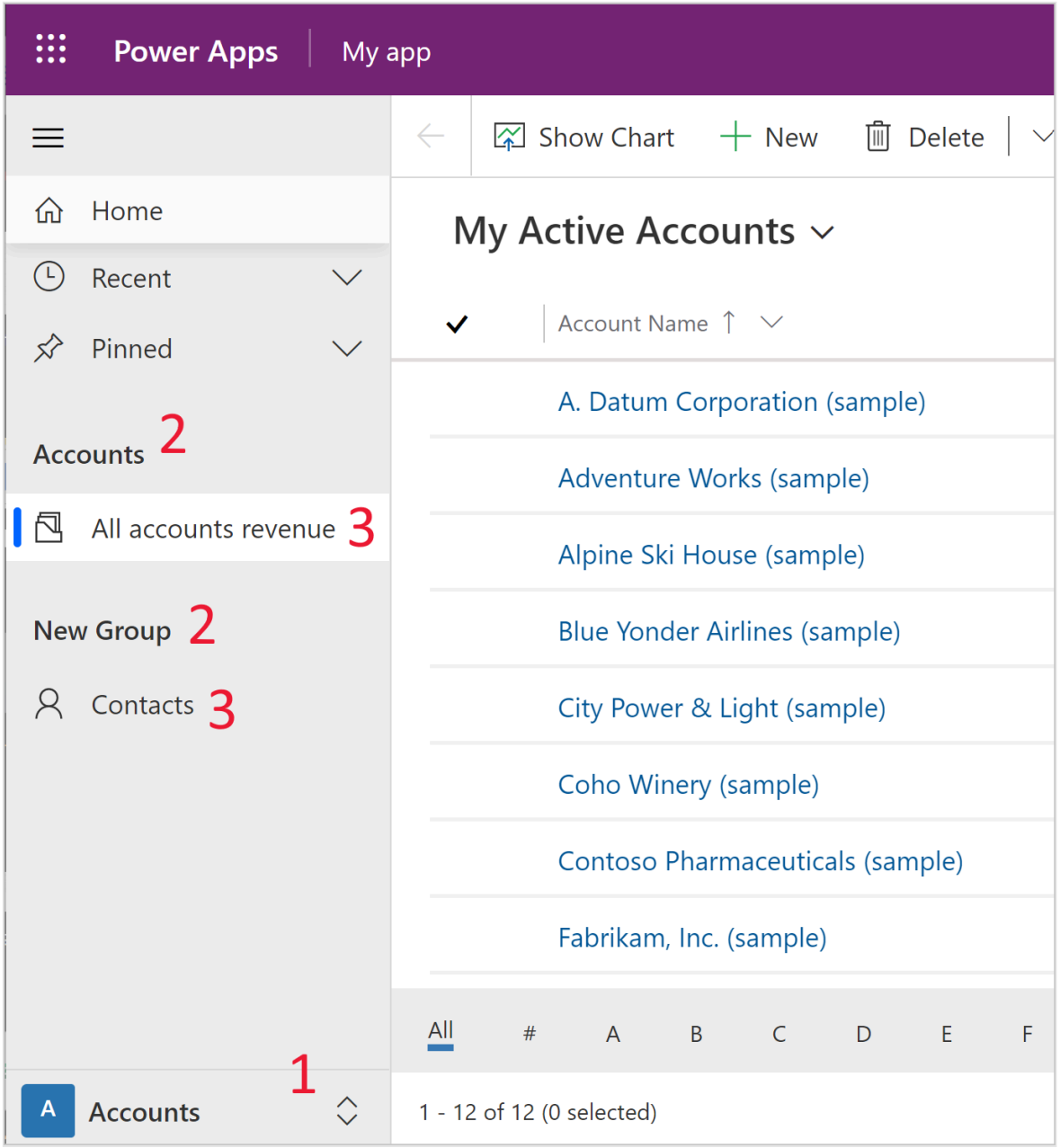
Article • 02/05/2025

In a model-driven app, there are the three main app runtime navigation components.

1. *Areas* - For apps with more than one area, a switch control is displayed in the lower left navigation pane. In the screenshot below, the current area is named *Accounts*.

2. *Groups* - Group names appear as a navigation element in an app with the subarea names within the group listed beneath it. In the screenshot below, one group is named *Accounts* and one is named *New Group*.

3. *Pages* - Pages appear under the group that they're configured within in the app designer. In the screenshot below, one page is named *All accounts revenue* and another page is named *Contacts*.



Create an area

By default, the ability to create additional areas is disabled. Areas are enabled by default for apps with existing multiple areas.

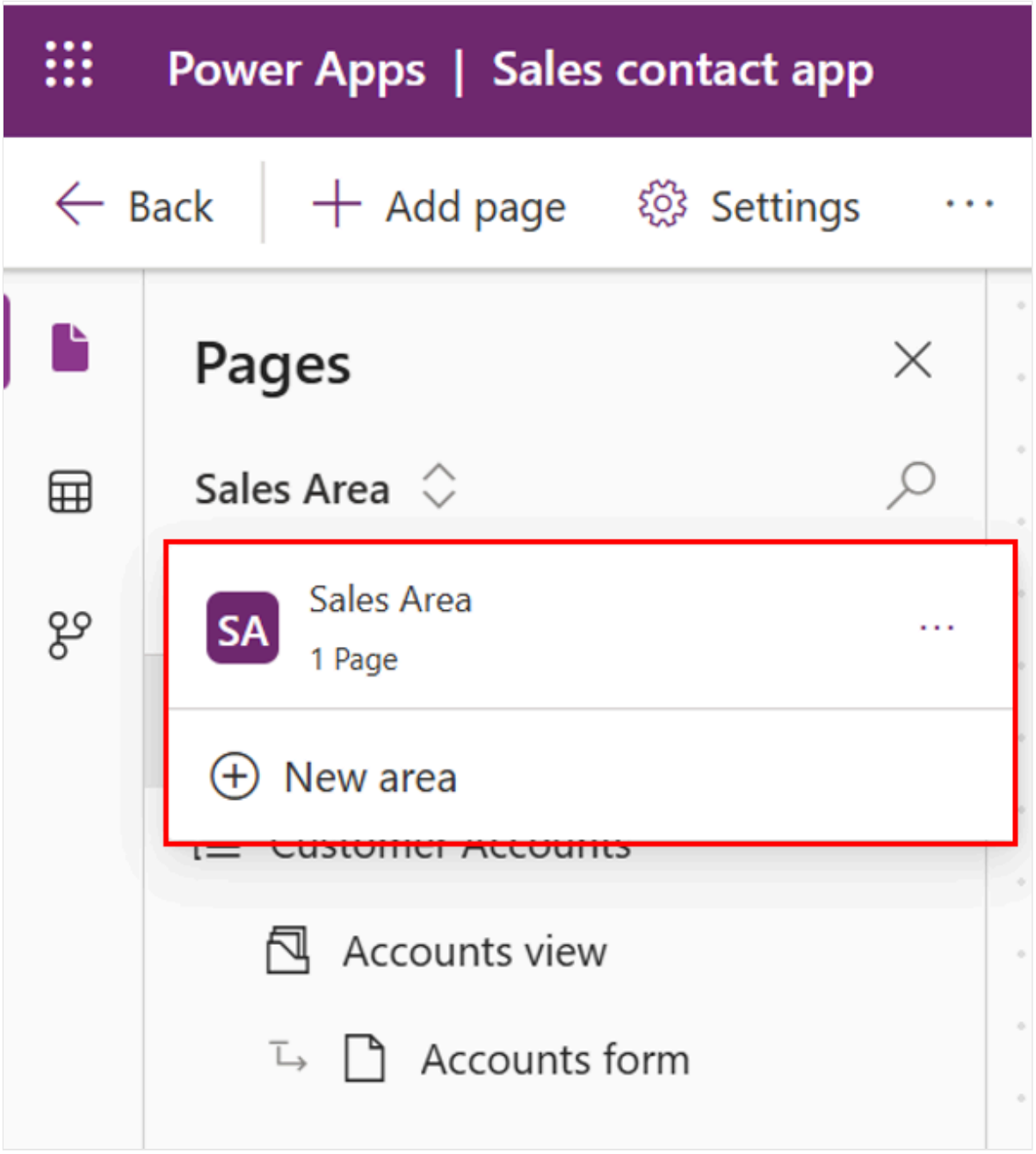
Enable areas

1. On the command bar select **Settings**.
2. Select **Navigation** on the left pane, and then select **Enable Areas**.
3. Close the **Settings** dialog.

Add a new area

1. On the left navigation pane, select **Pages**.

2. Select the area switcher under **Pages**, and then select **New area**. If the area switcher is missing, you need to enable areas.



3. Complete the properties pane for the area:

   - **Title**: Enter a title used to describe the area.
   - **Icon**: Optionally set an icon for the area. More information: Create or edit model-driven app web resources to extend an app
   - **ID**: The system generates an identifier for the area. Accept the system generated ID or enter a new one.

4. To save your app navigation changes select **Save**.

5. To publish the changes and make them available to other users, select **Publish**.

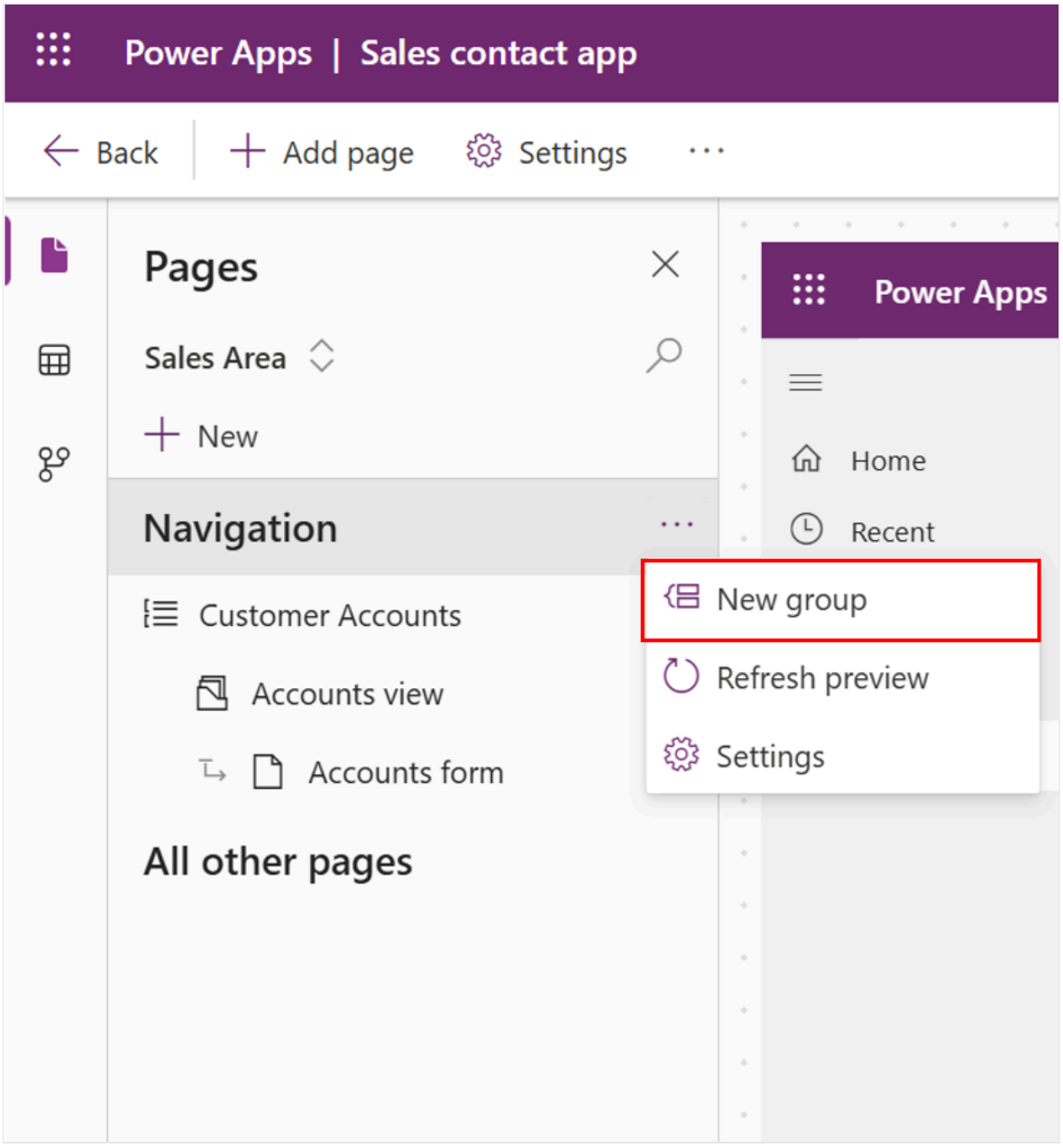
Create a group

To create a new group, complete the following steps:

1. On the left navigation pane, select **Pages**.

2. Select **...** next to **Navigation**.

3. Select **New group**.



4. Complete the properties pane for the group:

   - **Title**: Enter a title used to describe the group.
   - **ID**: The system generates an identifier for the group. Accept the system generated ID or enter a new one.
   - **Advanced Settings**:
     - **Localized titles**: Add localized titles for the group title. More information: Translate localizable text for model-driven apps
     - **Localized descriptions**: Add localized descriptions for the group description. More information: Translate localizable text for model-driven apps

5. To save your app navigation changes select **Save**.

6. To publish the changes and make them available to other users, select **Publish**.

Create a page

Pages were formerly called *subareas* in the app designer. To create a new page, complete the following steps:

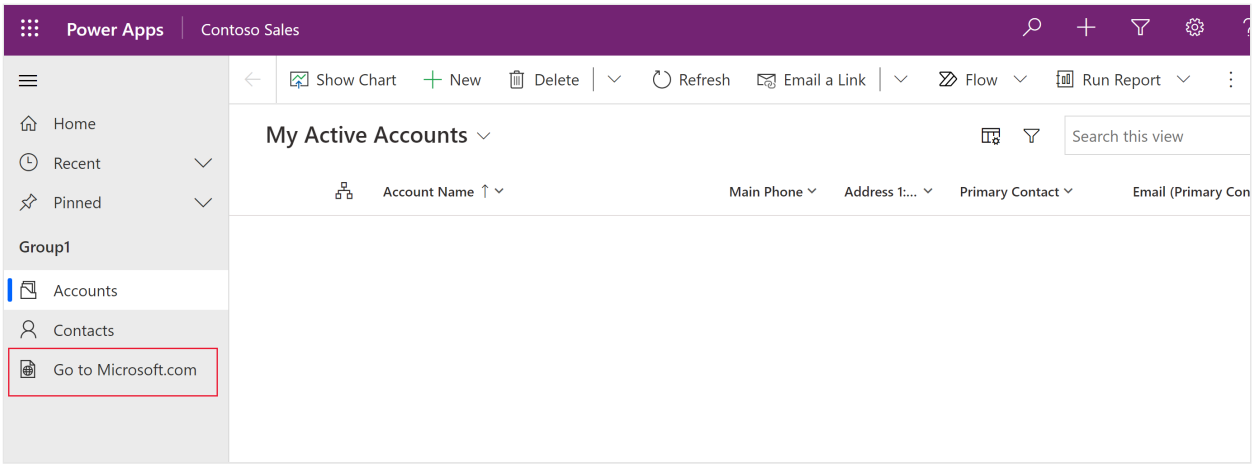
1. On the command bar, select **Add page**.
2. Select the content type you want.

    - **Dataverse table**. Select an existing table or create a new one. More information: Create a custom table
    - **Dashboard**. Select a dashboard. More information: Create or edit model-driven app dashboards
    - **Custom page**. Add a custom page, such as a canvas app page. More information: [Overview of custom pages for model-driven apps]
    - **Web resource**. Web resources represent files that can be used to extend an application such as HTML files, JavaScript, and CSS, and several image formats.
    - **Navigation link**. Add a URL, such as a website address. More information: Add a URL to an app (model-app-page-overview.md)

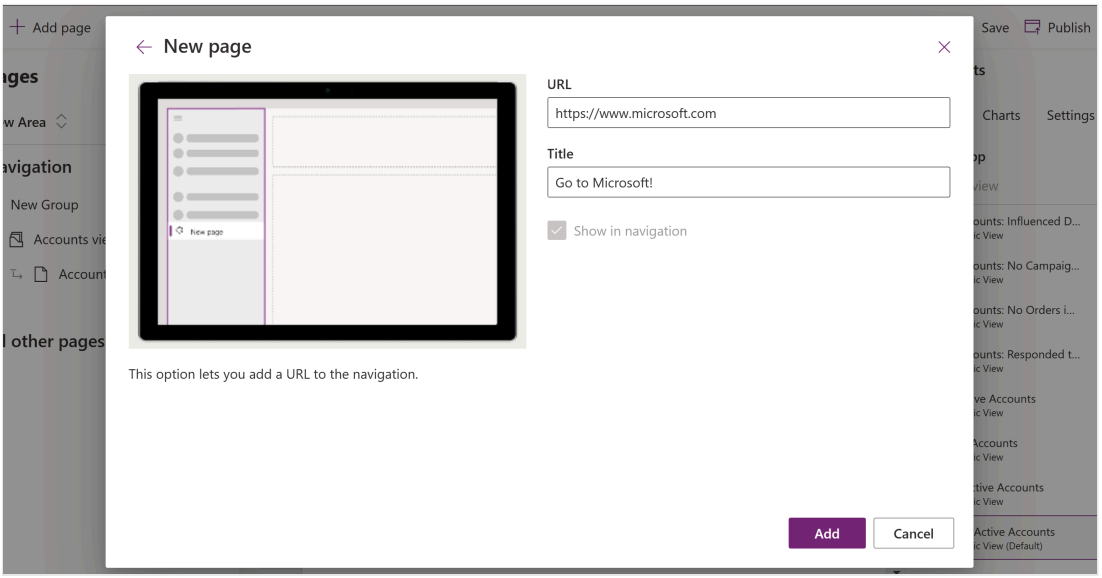
3. Complete the choices that are on you screen to create the page, and then select **Add**.
4. To save your app navigation changes select **Save**.
5. To publish the changes and make them available to other users, select **Publish**.

Add a URL to an app

A URL is a type of page in the app navigation. When the user selects the page, the URL opens in a new tab in the web browser.



1. In the model-driven app designer, select **Add page** on the command bar.
2. On the **New page** screen, select **URL**, and then select **Next**.
3. In the properties pane for the URL, enter the following information, and then select **Add**:
   - **URL**: Enter the full URL, such as *https://www.microsoft.com*.
   - **Title**: Enter a title for the URL component.



4. To save your app navigation changes select **Save**.
5. To publish the changes and make them available to other users, select **Publish**.

Remove a group or page

1. On the left navigation pane, select **Pages**.
2. Select the group or page you want, select the ellipses (...) and then select **Remove from navigation**.

Remove an area

1. On the left navigation pane, select **Pages**.
2. Select the area switcher under **Pages**, select **...** next to the area you want to remove, and then select **Remove from navigation**.

See also

Overview of the model-driven app designer (preview)

Feedback

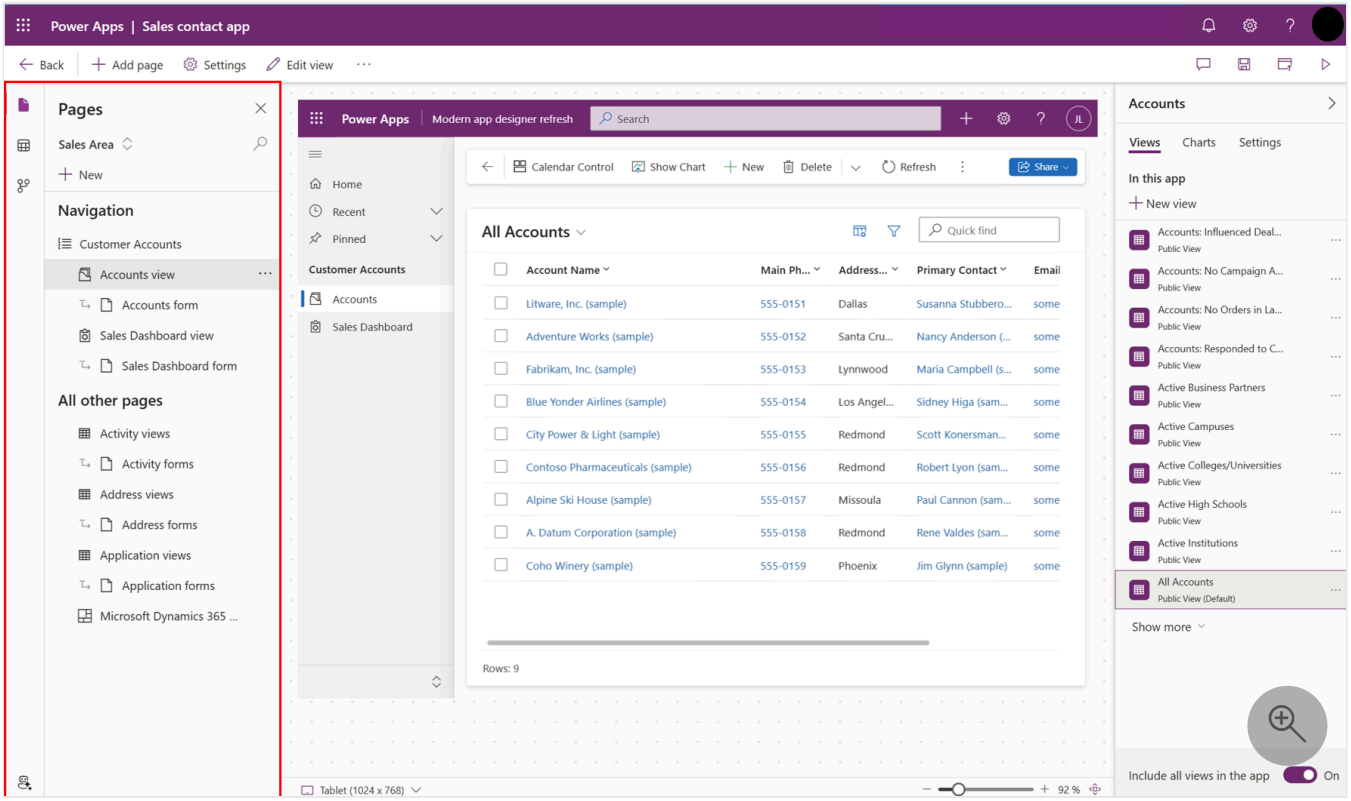
Was this page helpful? Yes No

Provide product feedback

Work with model-driven app pages

07/17/2025

App designer pages are containers for tables, dashboards, custom pages, or web resources. Pages determine your app's assets and navigation. You can choose whether to have the page appear in the app navigation (site map) or not.



Create a page

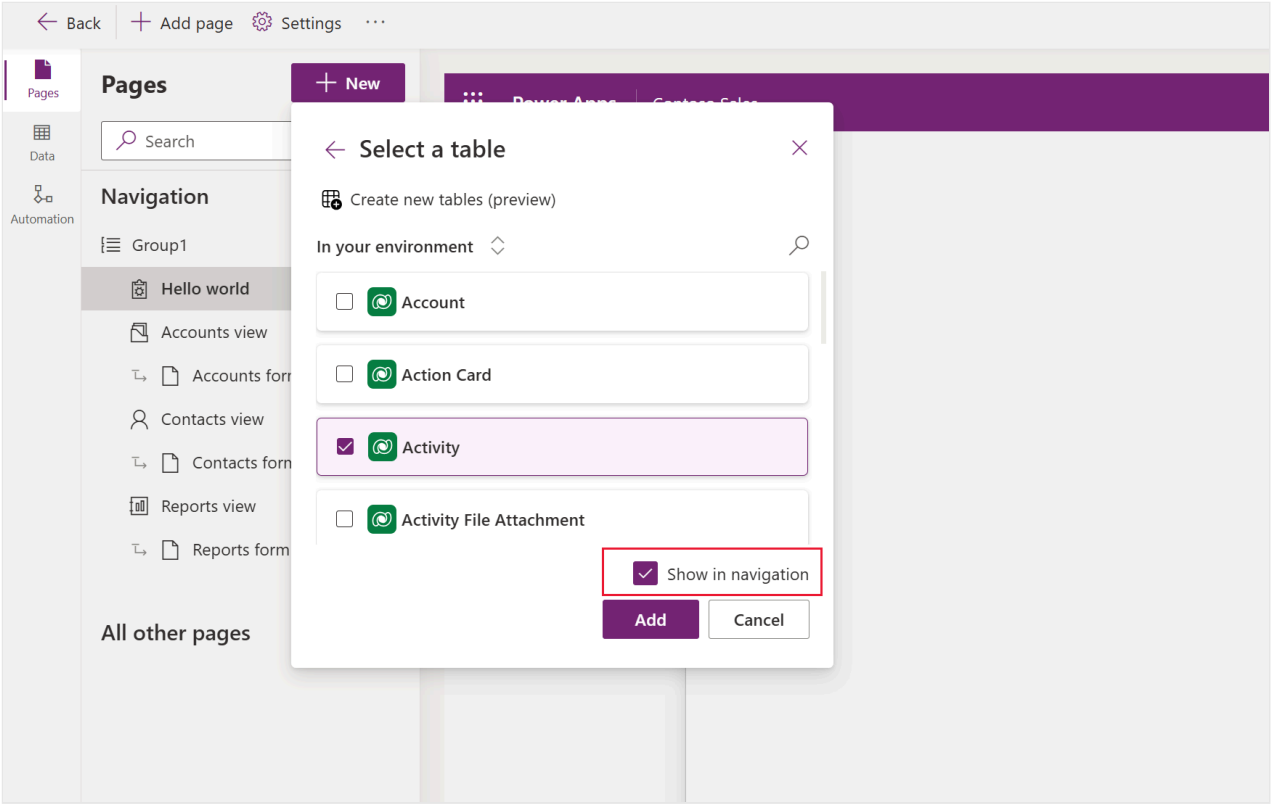
To create a page, follow these steps:

1. Sign in to Power Apps, select **Apps**, and then **Edit** the app you want.
2. In the app designer on the **Pages** tab, select **New**.
3. Select the page type, and then select **Next**:

   - **Describe a page**. Use natural language to create a generative page that's generated using AI. More information: Describe a page using natural language (preview)
   - **Dataverse table**: Display records of a standard or custom table in a full-page list view. Creating a data view page also adds an associated form page for viewing and editing data on a selected record. More information: Tables in Dataverse
   - **Dashboard**: Display charts and tables from multiple entities to visualize data on a single page. Select one or more system, interactive, or Power BI dashboards. More information: Create or edit model-driven app dashboards

- **Custom page**: Design and build a page that's based on a canvas app. More information: Overview of custom pages for model-driven apps
- **Web resource**: Select a web resource from a list of web resources. More information: Create or edit model-driven app web resources to extend an app
- **Navigation link**: Enter a direct link to a web resource. Create or edit model-driven app web resources to extend an app

4. If you don't want your page to appear in the app's site map, clear the **Show in navigation** option.



5. Select **Add**.

Work with tables on a page

Create and edit tables directly from the app designer.

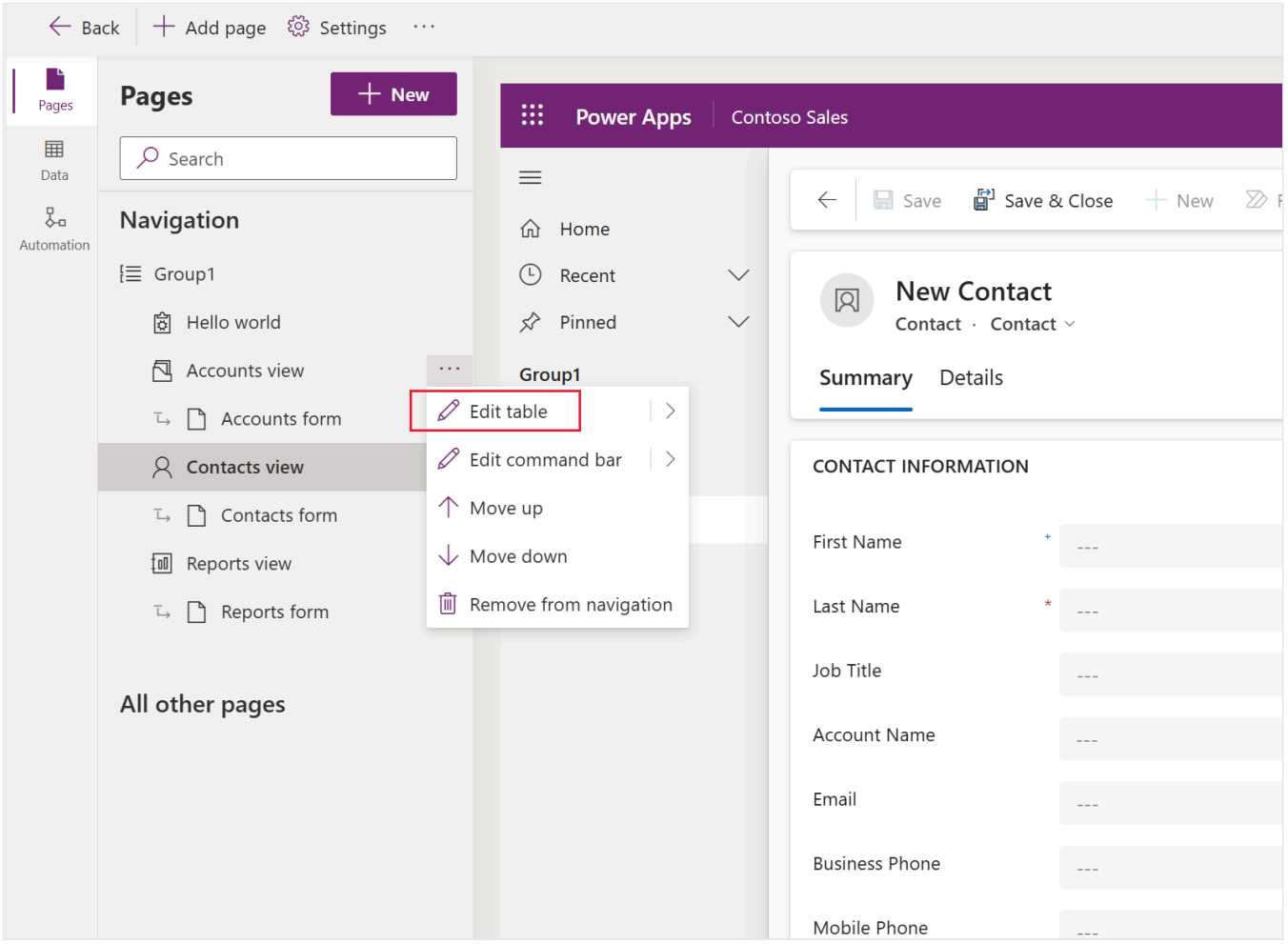
Create a table for a page

1. In the model-driven app designer, select **Add page**, select **Dataverse table** as the page type.
2. Select **Create new tables**. Otherwise, if your table is already created select it from the list.
3. If you selected **Create a new tables**:

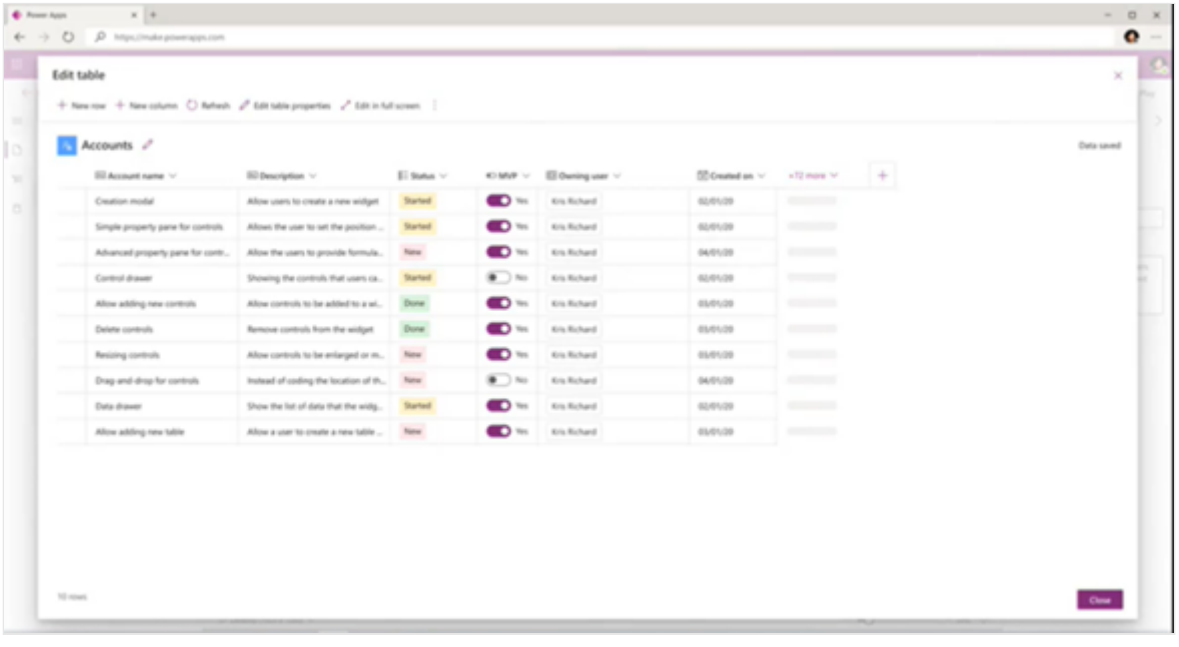
   - Select the option you want to create a table and follow the instructions on your screen.
   - More information: Create new tables

Edit a table for a page

From the **Pages** tab in app designer, select **...** to the right of the table name, and then select **Edit table**.



The table designer opens for you to make changes to the table.



To delete the table, **Close** the table hub, select **...** to the right of the table name, and then select **Remove**.

Edit a table view

To edit a view for a table that's added to your app in a page, complete these steps:

1. In the model-driven app designer, select a page that is based on a table.
2. On the command bar, select **Edit view**.

The view designer is displayed where you can edit the table view. More information: Edit public views from a table

Remove a page

To remove a page, complete these steps:

1. In the model-driven app designer, select a page.
2. Select the ellipses (**...**), and then select **Remove from navigation**.

Next steps

Create and customize a model-driven app form

Create or edit model-driven app dashboards

Add a custom page to your model-driven app

Manage model-driven app settings in the app designer

App settings define important details about the app, like its title. You define app settings when you create an app. If you want to change those settings later, you can do that in the app designer.

You must select **Publish** to activate an app settings change.

1. In the app designer select **Settings** on the command bar.

2. View and change the information, as required:

Expand table

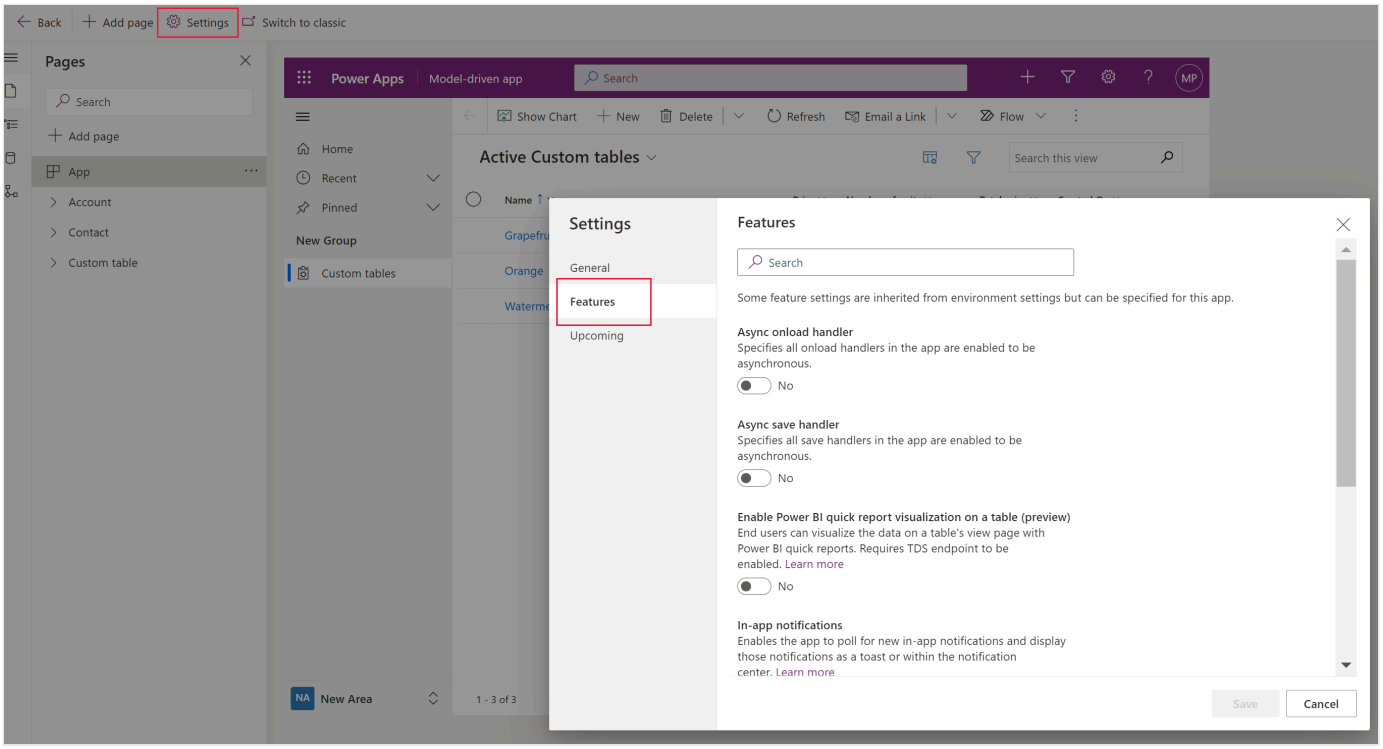
Area	Setting	Description
General	**Solution**	Read-only property that displays the solution name where the app is located.
General	**App name**	The friendly name for the app.
General	**Description**	The description of the app (optional). If your app is in a managed environment, you can select **Create description using AI** to have Copilot generate one for you. More information: Create an app description with Copilot
General	**Icon**	Change the app icon by selecting **Select icon**, to browse and select an image web resource. This icon is displayed on the preview tile of the app. More information: Create or edit model-driven app web resources to extend an app
Advanced settings	**Unique name**	Read-only property that displays the app unique name including the publisher prefix.
Advanced settings	**Welcome page**	This option allows a maker to select from the web resources available in your organization. The welcome pages created contain information that's useful to users, such as links to videos, upgrade instructions, or getting started information. The welcome page is displayed when an app is opened. Users can select **Do not show this Welcome Screen next time** on the welcome page to disable the page so it doesn't appear the next time the app starts. Notice that the **Do not show this Welcome Screen next time** option is a user-level setting and can't be controlled by administrators or app makers. More information: Create or edit model-driven app web resources to extend an app

Area	Setting	Description
Advanced settings	**Primary mobile player**	Determines which mobile player you want the model-driven app to be available from. By default, **Power Apps mobile** is selected. For example, if you want the model-driven app only available from the Dynamics 365 Sales mobile app, select **Dynamics 365 Sales**. - **Power Apps Mobile**. More information: Use model-driven apps on Power Apps mobile - **Dynamics 365 Sales**. More information: Overview of Dynamics 365 Sales mobile app - **Field Service (Dynamics 365)**. More information: Overview of the Field Service (Dynamics 365) mobile app

3. **Save** and **Publish** the app.

Features

When you select **Settings** from the command bar, select **Features** to enable or disable model-driven app features for the app.



Here are a few of the features available to app makers:

- **Enable Power BI quick report visualizations on a table**. Enabled by default, this feature allows app users to create a Power BI quick report from a table view by selecting the **Visualize this view** command on the app command bar. More information: Visualize data in a view with Power BI service

- **In app notifications**. When enabled, the app polls for new in-app notifications and displays them. More information: Send in-app notifications within model-driven apps
- **Lock tabs at the top of forms on mobile and tablets**. When enabled, tabs remain visible at the top of a form while users scroll through the data on the form. More information: Lock tabs at the top of forms
- **Mobile commanding improvements**. When enabled, optimizes the command bar for phone and tablet providing easy access to contextual commands to users helping increase productivity and satisfaction. More information: Mobile commanding improvements
- **Enable offline classic**. Enable mobile offline classoc
- **Tablet optimization for command bar**. When enabled, replaces the native command bar at the bottom of the screen with the web command bar located at the top of the screen on tablets. More information: Tablet optimization for command bar
- **New look for model-driven apps**. Enabled by default, this feature determines whether end users see the modern refresh app experience or the classic look. This setting is hidden and ignored with the **2026 Wave 1** release when the new look becomes mandatory. More information: Modern, refreshed look for model-driven apps
- **Form fill assist toolbar**. When enabled, the form fill assist toolbar is visible at the top of a form, providing easy access to Copilot form fill assistance capabilities like smart paste and files. For more information about AI form fill assistance environment settings, go to AI form fill assistance. For more information about using this feature, go to Use Copilot's form fill assistance feature in model-driven apps.

Navigation

Controls navigation features for the app.

- **Home**. Displays a link to the home page for the model-driven app.
- **Recent**. Displays the list of links for the model-driven app that were recently visited.
- **Pinned**. Pin items from the **Recent** list to locate them easily.
- **Enable collapsibile groups**. Disabled by default. Allows groups to be collapsed or expanded. More information: Create a group
- **Enable Areas**. Disabled by default. Areas can contain multiple groups. For apps with multiple areas, a user must select the area switch located at the bottom left in the app to change to a different area. Add a new area

Upcoming

[This topic is pre-release documentation and is subject to change.]

The **Upcoming** tab displays preview features currently available:

> **Important**
>
> Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

- **Microsoft 365 Copilot in model-driven apps**. Microsoft 365 Copilot chat makes it easier for users to work with model-driven apps by offering AI-powered insights through natural language conversations. With this feature, users can quickly find information, navigate apps more easily, and get help to boost their productivity. More information: Add Microsoft 365 Copilot chat for app users in model-driven apps (preview)

Known limitations

The following app properties can't be edited in the app designer:

- App Url Suffix
- Manage App

See also

Create or edit an app

Last updated on 04/22/2026

Overview of custom pages for model-driven apps

The custom page is a new page type within a model-driven app, which brings the power of canvas apps into model-driven apps. Custom pages increase the convergence of model-driven and canvas apps and can be used to add full pages, dialogs, or panes with the flexibility of Power Apps Studio. It also includes a low-code page authoring experience with expressions and custom Power Apps component framework controls.

A custom page is more flexible than a model-driven app form, view, or dashboard page. You can include one or more tables. Then, you define the data and component interactions. The custom page is a separate solution element, which means one maker can edit one custom page at a time. Like other model-driven app pages, the page state comes from the parameters passed or is retrieved from persisted tables.

> **Important**
>
> Custom pages are a feature with significant product changes and currently have a few known limitations outlined in Custom Page Known Issues.

Expand table

Capability	Status	Notes
Runtime for custom pages	General Availability	
Solution and ALM for custom pages	General Availability	
Connectors in custom pages	General Availability	List of all Power Apps connectors
Modern controls in custom pages	General Availability	List of supported controls
Code components in custom pages	General Availability	
Monitor support for custom pages	General Availability	
Authoring custom pages	General Availability	Use the modern app designer and Power Apps Studio to author custom pages that are supported at runtime.

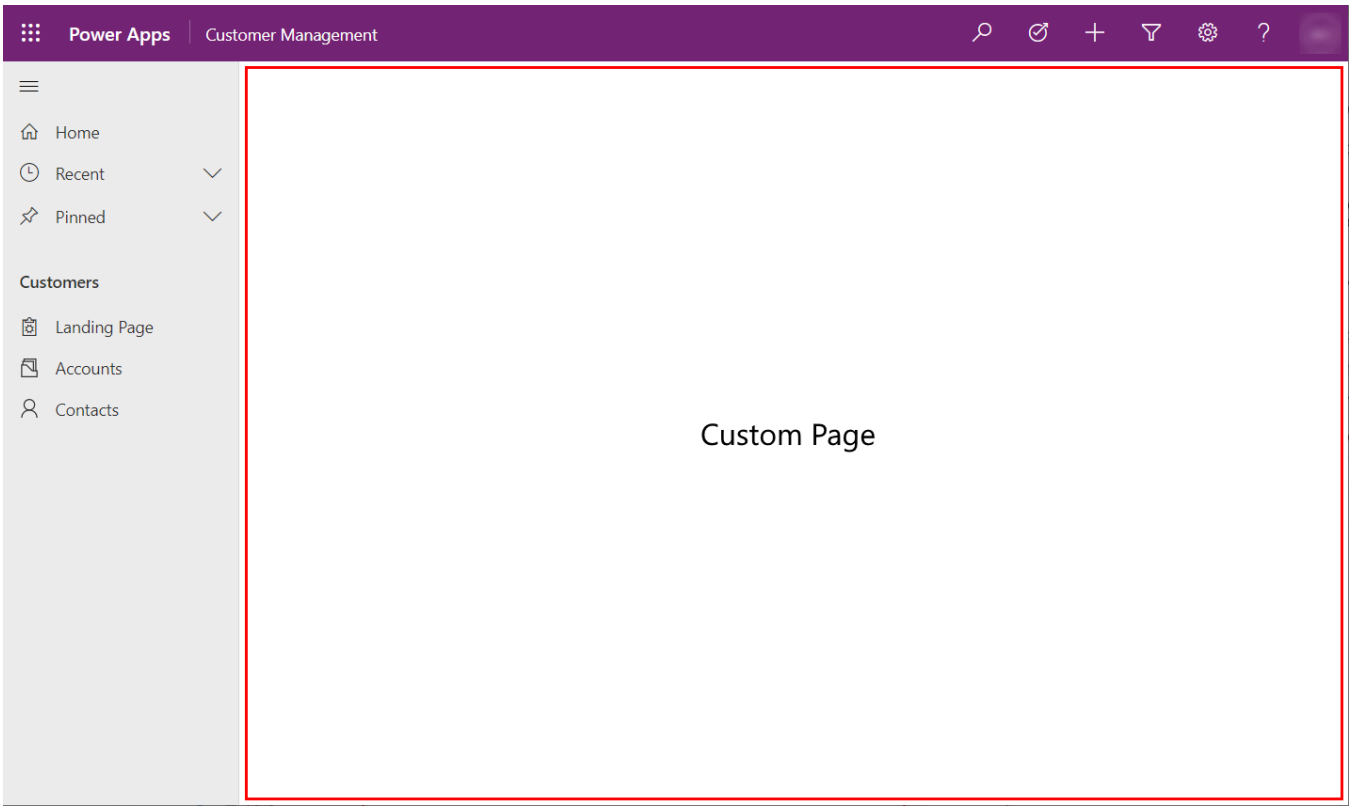
Capability	Status	Notes
Canvas components in custom pages	General Availability	
Custom page in Teams model-driven app	Public preview	
Custom page in mobile online	Public preview	iOS must allow enabling “Allow cross site tracking” that can be prevented by device management

Examples of custom pages

These images show the custom page inline within the model-driven app. The model-driven app has the full-page space in the images without the header and navigation.

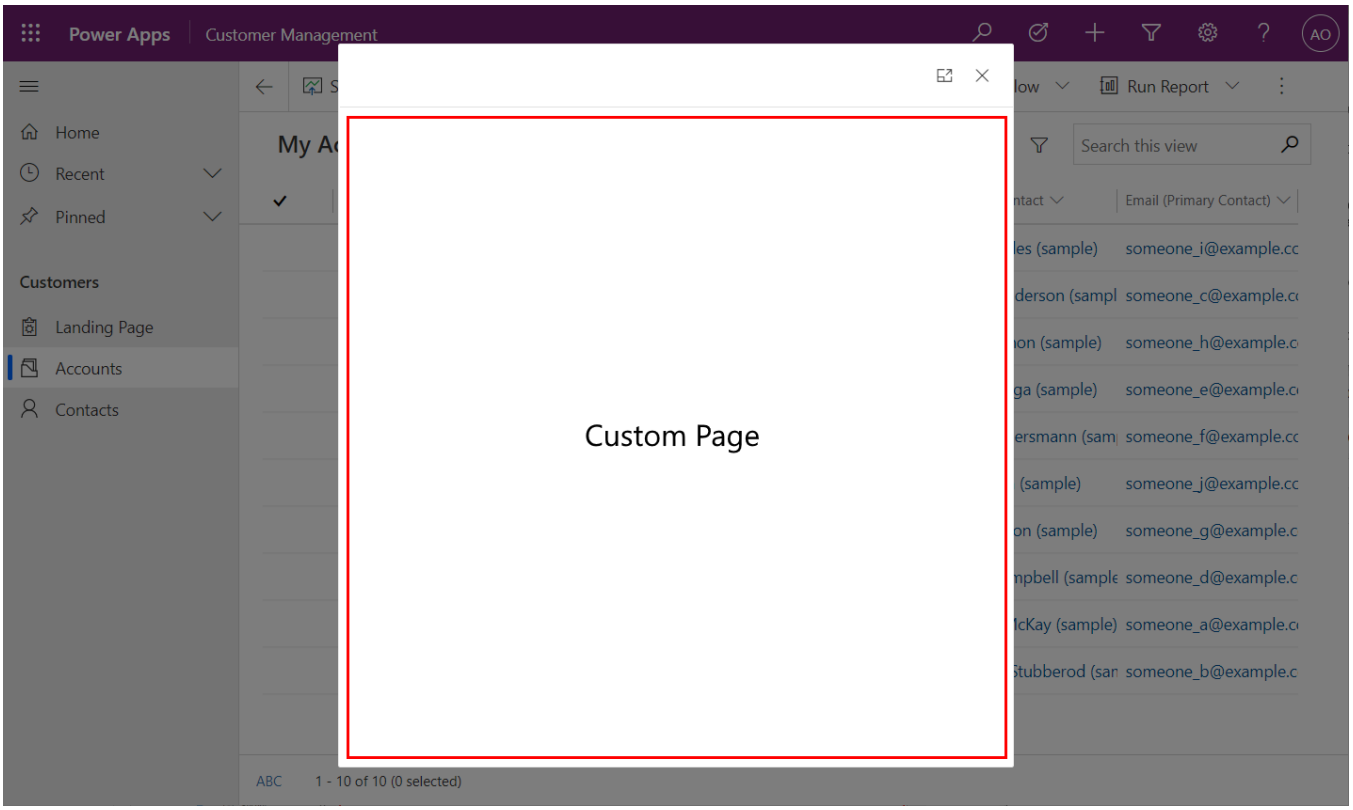
Custom page as main page.

[!div class="mx-imgBorder"]



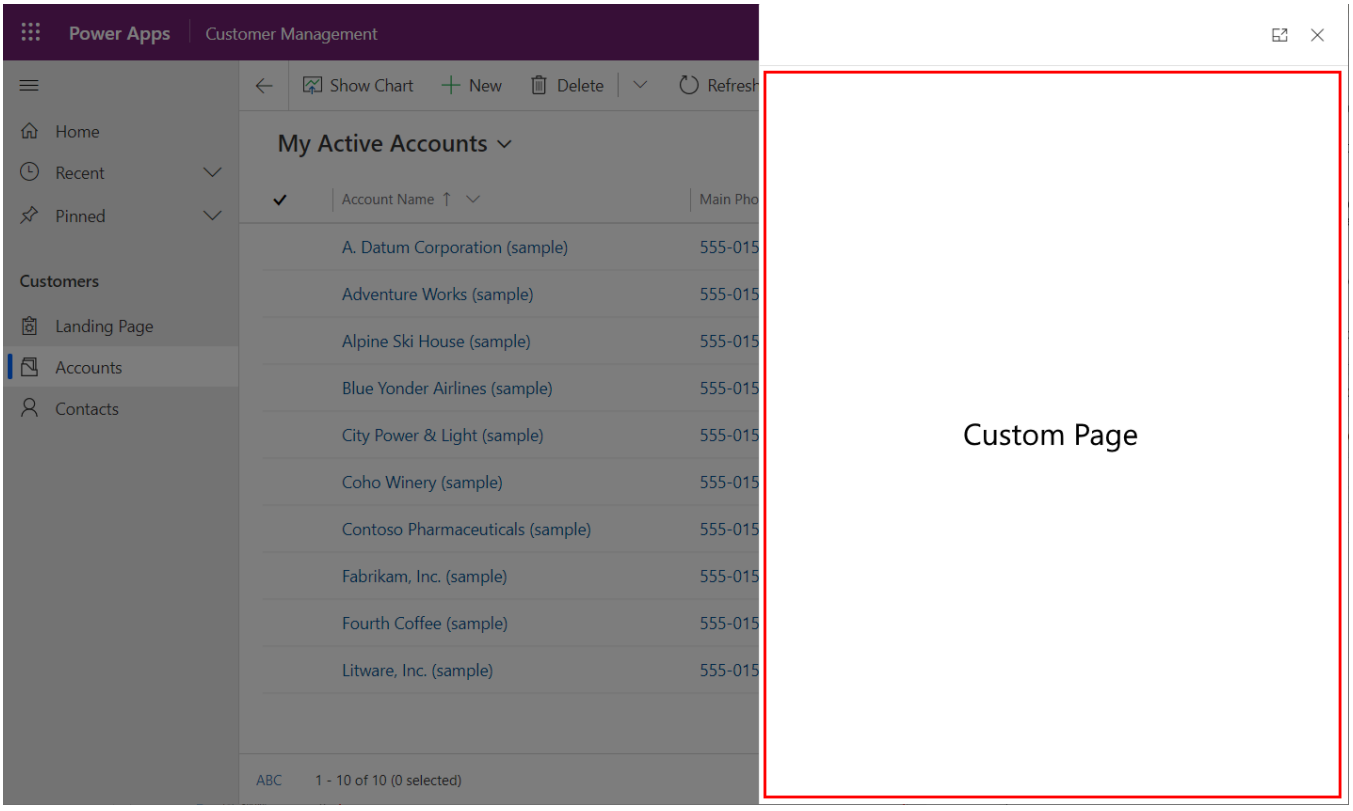
Custom page as a center dialog.

[!div class="mx-imgBorder"]



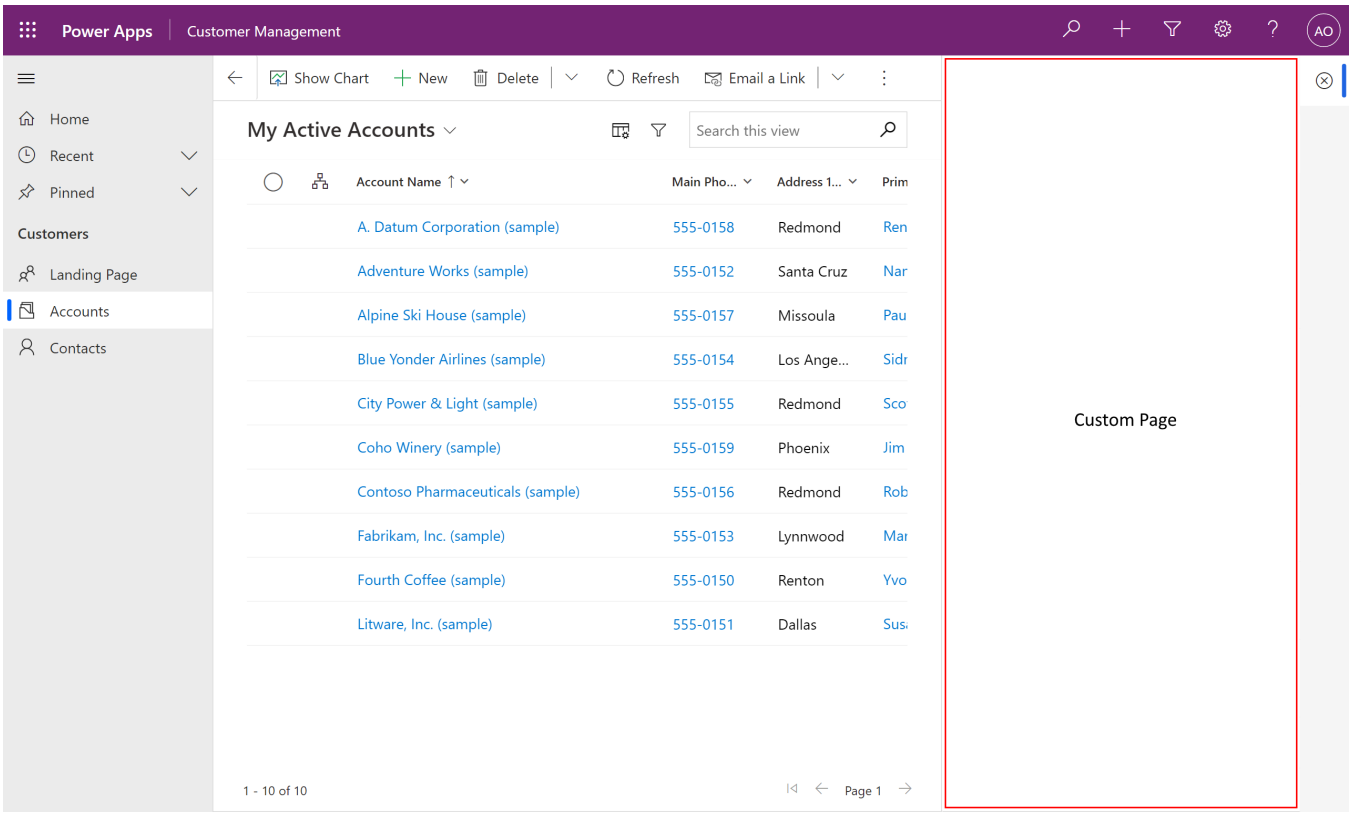
Custom page as a side dialog.

[!div class="mx-imgBorder"]



Custom page as an app side pane allows opening a custom page within the new app side pane on the right side of the app.

[!div class="mx-imgBorder"]



Custom pages must be created from a solution either from the modern app designer or the **Solution** area in Power Apps using **New** > **Page**. More information: Add a custom page to your model-driven app

Custom pages are different than embedded canvas apps

By using a custom page, you can create a new page experience by using the canvas app capabilities. This approach provides a low-code authoring experience with more flexible layouts, more control with styling options, the ability to add connector data, use expressions, and more. You create custom pages in Power Apps Studio with increasing context of the model-driven app that the page runs in.

Embedded canvas apps also use the canvas capabilities with a hosting approach that isn't as integrated as a custom page. The simpler integration of an embedded canvas app means the current limitation on number of embedded canvas apps hasn't changed. The advanced integration of the custom page addresses those limits. The embedded canvas app can only be placed on a model-driven form acting like a low-code component. More information: Embed a canvas app on a model-driven form.

In most cases, use custom pages instead of embedded canvas apps for tighter integration and better performance.

> **Note**
>
> Don't exceed 25 custom pages in a model-driven app. If you add more custom pages, it can increase the wait time that an app user experiences when they play an app with custom pages, after the last app publish. The increased user wait time affects only the first app launch, by the first user, after the last publish of the model-driven app.

Migrate standalone canvas app content to custom pages

Existing standalone canvas apps aren't supported for use as a custom page and the expected app structure is different. A standalone canvas app often has many screens with global access to all controls and variables. The custom page is expected to typically be a single screen with loose coupling to provide performance and co-development capabilities.

To migrate an existing standalone canvas app, start by identifying a mapping of screens to separate custom pages. For each separate custom page, complete these steps:

1. Create a blank custom page from the model-driven app designer. More information: Add a custom page to your model-driven app
2. Add a canvas app data source for data used by the screen.
3. Copy the screen from the original canvas app in Power Apps Studio.
4. Paste the screen into the blank custom page in Power Apps Studio.
5. Change the navigate calls to use the custom page name instead of the screen name.
6. Add the custom page into the model-driven app designer site map.

Frequently asked questions

- What data can the custom page use?

  A custom page can use Microsoft Dataverse and all of the connectors for Power Apps. More information: List of all Power Apps connectors.

- What interactions can the custom page have with the model-driven app?

  Custom pages can be added to the site map for direct navigation using Add custom page to sitemap. Model-driven app pages can open a custom page using the `navigateTo` Client API. Custom pages can navigate to other custom pages or to a model-driven app page such as a form, view, or dashboard with the Power Fx navigate function. More information: Navigating to a custom page

- How is the custom page made responsive?

  The responsive container controls enable building a responsive app page without formulas. More information: Building responsive pages. More custom page design guidance can also be found in Design a custom page for your model-driven app.

- How is the custom page managed in a solution?

  Each custom page is a separate component in the solution, which allows one maker to edit one custom page at a time. Most custom pages will have a single screen. Instead of multiple screens, they use the custom page's navigation functions to move to another custom page or model-driven app page. When a custom page has multiple screens, it's still a single solution component so only one maker can be working on the contained set of screens.

- What licenses are allowed to use a custom page and does a custom page impact app counts?

  The custom page uses a special canvas app type, which allows it to be managed differently. The custom page is considered part of the model-driven app infrastructure and can only be used within a model-driven app. So, it follows the license for the model-driven app. Also, custom pages don't count toward the app limits because they're treated as a page instead of an app.

- Do custom pages need to be shared like standalone canvas apps?

  The custom page aligns with the model-driven app page sharing, which relies on the model-driven app sharing without sharing individual pages for app users. Makers might need to share the custom page to allow editing.

See also

Add a custom page to your model-driven app

Design a custom page for your model-driven app

Using PowerFx in custom page

Add connectors into custom pages

Use Monitor to troubleshoot custom page

Model-driven app custom page known issues

Last updated on 03/31/2026

Add a custom page to your model-driven app

This article shows you how to create and edit a custom page for a model-driven app by using the modern app designer.

This article walks you through opening a model-driven app in the app designer, which you use to add a custom page to a model-driven app. If you need to create a new model-driven app, go to Create a model-driven app with the app designer.

For information about the limitations and known issues with custom pages, see Custom page known issues.

Create or edit a custom page

Custom pages can be created from two places. The first is while authoring a model-driven app in the modern app designer. The other is from the **Solutions** area in Power Apps. Custom pages can be edited from the **Solutions** area but won't appear in the **Home** or **Apps** areas of make.powerapps.com.

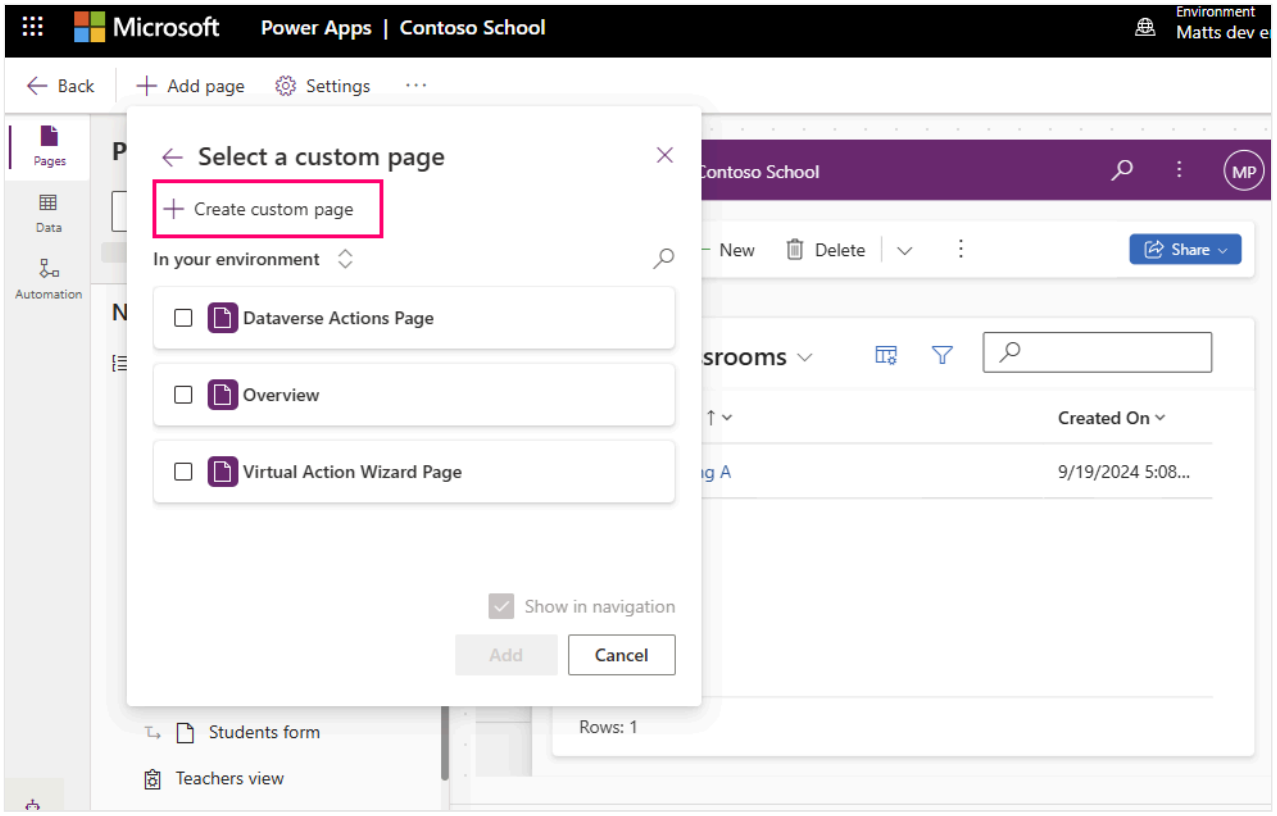
> **Note**
>
> Custom pages must be created from a solution either from the modern app designer or the **Solutions** area in Power Apps using **New** > **App** > **Page**. The custom page is a different canvas app type than the normal standalone canvas app.

Create new custom page from modern app designer

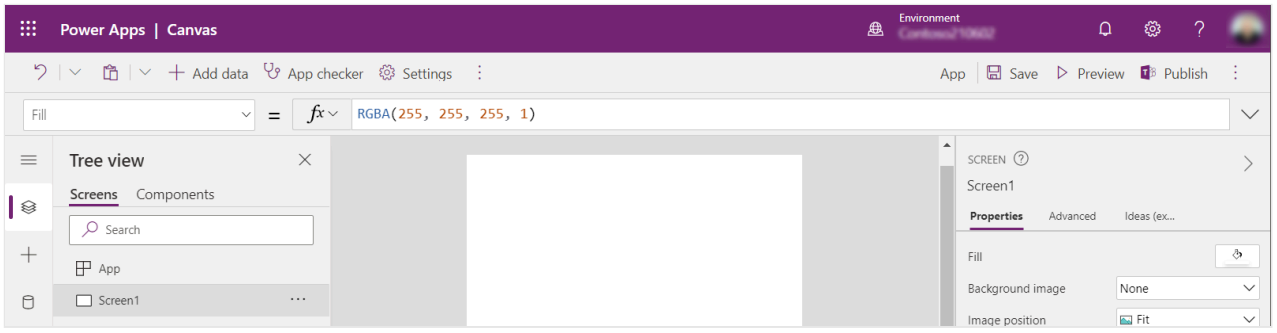
1. Open make.powerapps.com.
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution containing the existing model-driven app.
4. Select the model-driven app where you want to add a custom page, and then select **Edit** on the command bar.

5. In the app designer, select **Add page** > **Custom page** on the command bar.

6. Select **Create custom page**.



7. Power Apps Studio opens for page authoring.



8. When you're finished creating your canvas app custom page, **Save**, **Publish**, and then close the Power Apps Studio browser tab to return to the model-driven app designer.

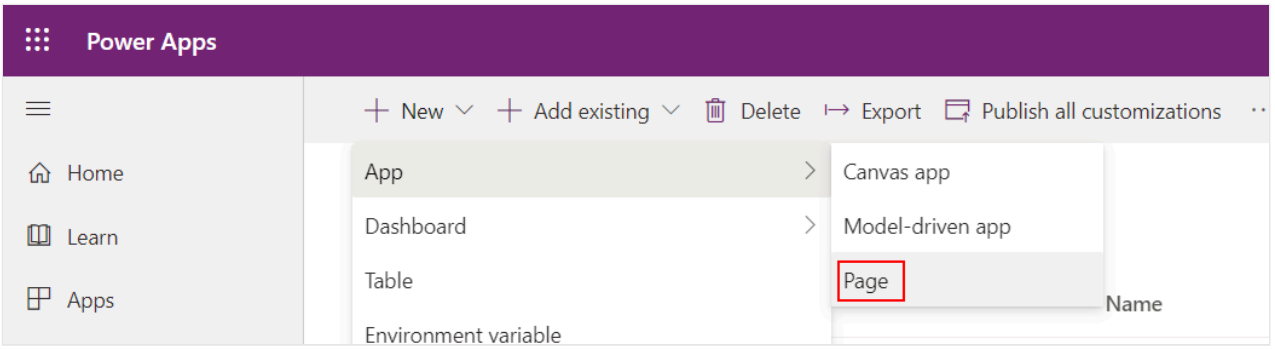
Create new custom page from the solutions area

1. Sign in to Power Apps.

2. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. Open or create a solution to contain the new custom page.

4. Select **New** > **App** > **Page**.



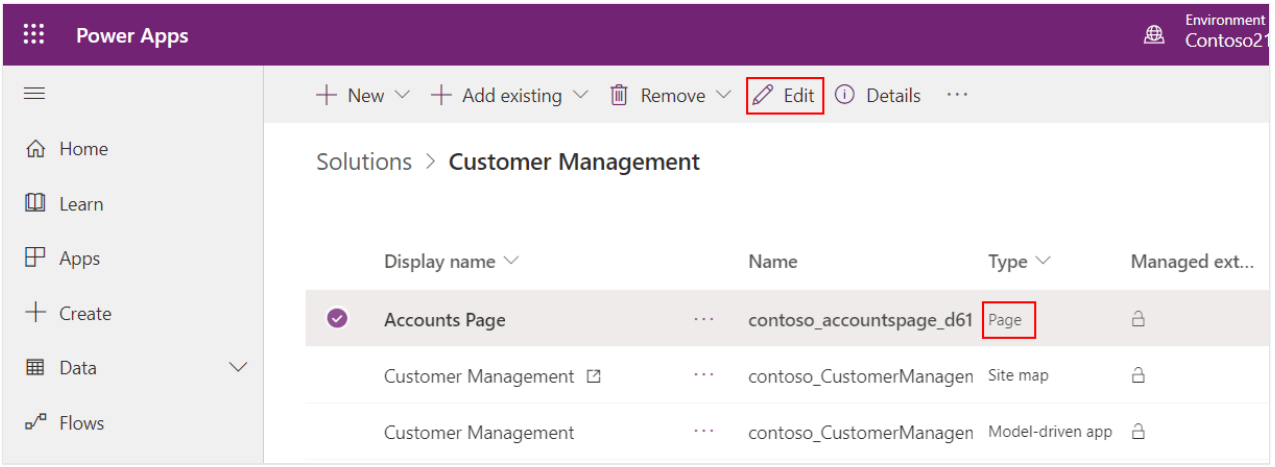
Edit an existing custom page

1. Open make.powerapps.com.

2. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. Open or create a solution to contain the new custom page.

4. Select the custom page with **Page** type, and then select **Edit**.



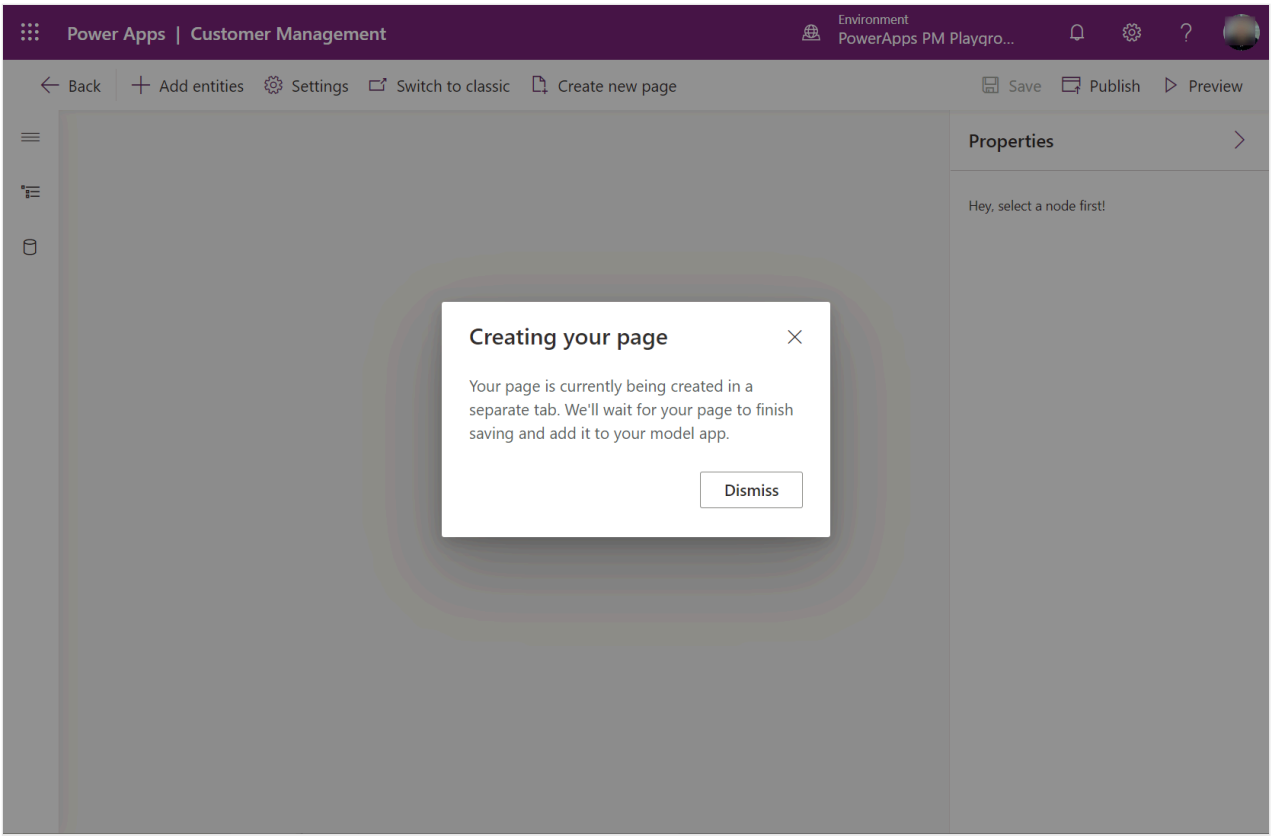
Author custom page content

1. Design the custom page content. More information: Design a custom page for your model-driven app

2. Save and publish the custom page.

3. Close Power Apps Studio.

4. Return to the app designer browser tab and refresh the app designer by selecting **Dismiss**.



5. Select **Publish** in the app designer to add the changed custom page into the model-driven app.

6. Select **Preview** to play the app in a new browser tab.

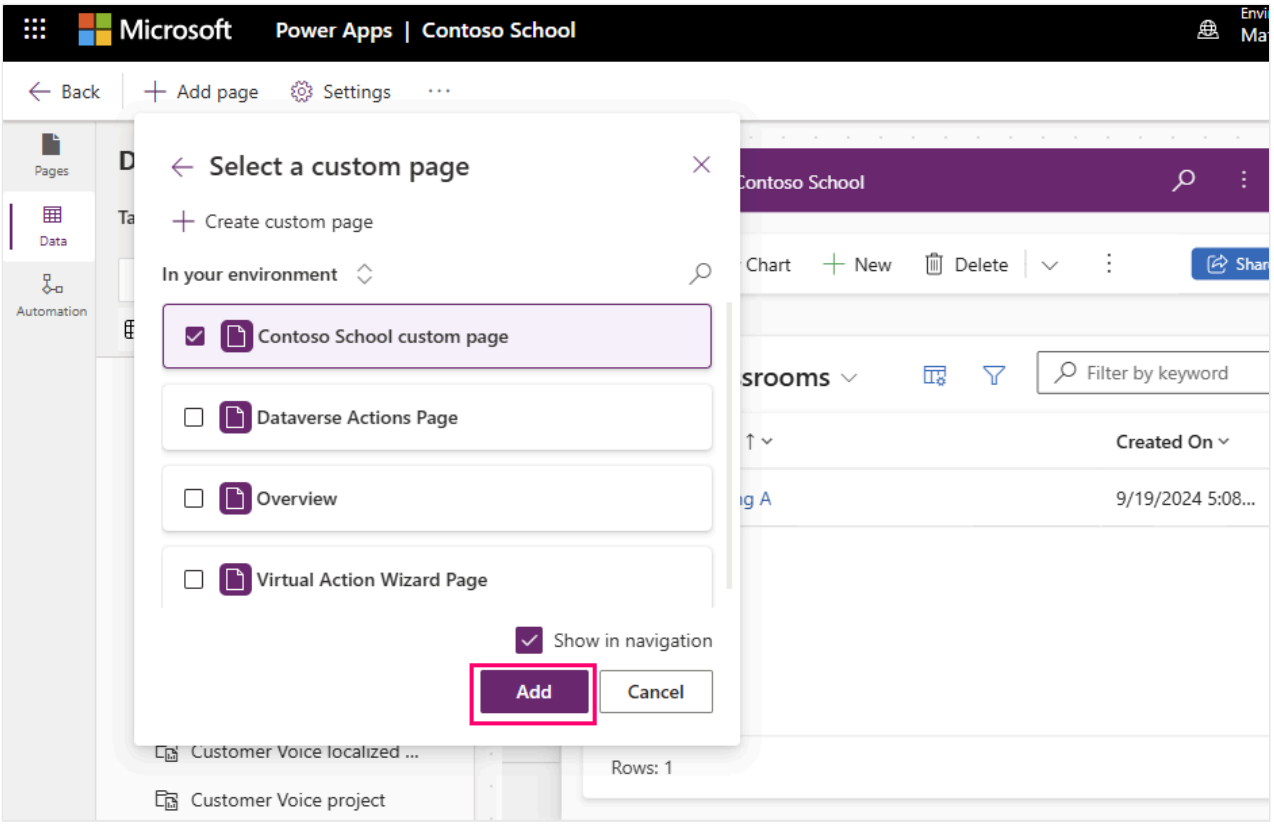
Add an existing custom page into a site map

1. Sign in to make.powerapps.com.

2. Open an existing model-driven app using modern app designer.

3. Select **Add page** > **Custom page** on the command bar.

4. Select the custom page in the list of custom pages available in the environment, and then select **Add**.



5. Select **Publish**, which also saves the app if there are changes.

6. Select **Play** to run the app in a new browser tab.

7. To close the app designer select **Back** and return to the solution.

Publishing a custom page

> **Important**
>
> Currently, model-driven apps must be re-published after a custom page is published. Otherwise the model-driven app continues to use the previous published custom page.

After saving changes to a custom page in Power Apps Studio, the custom page must be first published by Power Apps Studio. Then all model-driven apps referencing that custom page need to be published.

1. From Power Apps Studio, select **Publish**.

2. From app designer or solutions area, select **Publish** on each model-driven app referencing the custom page.

See also

Model-driven app custom page overview

Design a custom page for your model-driven app

Using PowerFx in custom page

Navigating to a custom page using client API

Code components for custom page designer

Last updated on 04/08/2026

Design a custom page for your model-driven app

Article • 01/29/2025

This article provides tips for designing a custom page for use in a model-driven app. A custom page lets you create a page that's similar to a canvas app in your model-driven app.

> **Important**
>
> Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in Custom page known issues.

Supported controls in a custom page

Custom page authoring currently supports a subset of canvas app controls. The table here lists the currently supported controls.

Expand table

Control	Control Type	Notes
Label[1]	Display	
Text Box[1]	Input	
Date Picker[1]	Input	
Button[1]	Input	
Combo Box[1]	Input	
Check Box[1]	Input	
Toggle[1]	Input	
Radio Group[1]	Input	
Slider[1]	Input	
Rating[1]	Input	
Vertical Container	Layout	New responsive horizontal layout container

Control	Control Type	Notes
Horizontal Container	Layout	New responsive horizontal layout container
Rich Text Editor	Input	
Gallery	List	
Icon	Media	
Image	Media	
Edit Form	Input	
Display Form	Input	
Code components	Custom	Add code components to a custom page
Canvas components (preview)	Custom	Add canvas components to a custom page

[1] Control is a new modern control. The control was introduced for canvas apps in Teams. The control is based on Fluent UI library wrapped with Power Apps Component Framework.

Custom components support for custom page

You can add both low-code (canvas components) and pro-code (code components) custom UX components to your environment and make them available for all makers. For custom page specific UX extensibility articles, go to add canvas components to a custom page for your model-driven app and add code components to a custom page for your model-driven app.

In general, the low-code extensibility approach is simpler to build, test, and has a lower maintenance cost. We recommend evaluating canvas components first and then use code components only if there's a need for more complex and advanced customization.

More information:

- Canvas component gallery
- Code components samples
- Code components community resources

Enable responsive layout with container control

Responsive custom page layouts are defined by building a hierarchy of **Horizontal layout container** and **Vertical layout container** controls. These controls are found in the canvas app designer under **Layout** on the **Insert** tab.

Set the minimum screen height and width on the **App** object to prevent page level scroll bars and use a vertical body scroll bar.

PowerApps Formula

```
MinScreenHeight=200
MinScreenWidth=200
```

Optionally, the custom page design size can be adjusted in **Settings** > **Display** with **Size** set to **Custom**. Then set the **Width** and **Height** to a more typical desktop custom page size like width 1080 and height 768. Changing this setting after controls are added to the screen might cause some layout properties to reset.

Set the topmost container to fill the entire space and resize based on available space.

PowerApps Formula

```
X=0
Y=0
Width=Parent.Width
Height=Parent.Height
```

Horizontal wrapping of a flexible height container

To support pages adjusting from desktop down to a narrow width, enable these properties on a horizontal container with flexible height. Without these settings, the page clips the control when the page is narrow.

PowerApps Formula

```
Direction=Horizontal
FlexibleHeight=true
Justify=Stretch
Align=Stretch
VerticalOverflow=Scroll
Wrap=True
```

Child containers or controls directly under this container should be set to have a minimum width that allows the page to fit within a 300 pixel width. Consider the padding on the container or control as well as parent containers.

Vertical wrapping of a flexible width container

To support pages adjusting from desktop down to a narrow width, enable these properties on a vertical container with flexible width. Without these settings, the page clips the control when the page is narrow.

PowerApps Formula

```
Direction=Vertical
FlexibleWidth=true
Justify=Stretch
Align=Stretch
HorizontalOverflow=Scroll
Wrap=True
```

Child containers or controls directly under this container should be set to have a minimum height that allows the page to fit within a 300 pixel width. Consider the padding on the container or control as well as parent containers.

More information: Building responsive layout.

Vertical container with fixed header, flexible body, fixed footer

1. On the **Vertical Container**, set **Align (horizontal)** to **Stretch**
2. Insert three **Horizontal Container** controls within the parent **Vertical Container**.
3. On the first and third child horizontal container controls, set **Stretch height** off and reduce height to the amount of space needed, such as *Height=80*.

Horizontal container with two even child containers

1. On the parent horizontal container, set **Align (vertical)** to **Stretch**.
2. Insert two **Vertical Container** controls within the parent **Horizontal Container**.

Styling custom page controls to align to model-driven app controls

By creating the custom page from the modern app designer, these features use the default values.

- Theme for the custom page. Theme values for the controls used in a custom page are automatically set to match the default blue theme of the Unified Interface. This default theme is used both in the studio and at application runtime. Explicit theme selecting is removed from custom page authoring experience.
- Controls need to use a different font size, which is based on their position in the page hierarchy.

  > **Note**
  >
  > Custom page text has an upscaling of 1.33 so you need to divide the target **FontSize** by 1.33 to get the desired size.

  Expand table

Label Type	Target FontSize	FontSize to use
Page title	17	12.75
Normal labels	14	10.52
Small labels	12	9.02

- Primary and secondary button controls need the following styling changes:

  Primary buttons

  PowerApps Formula

  ```
  Color=RGBA(255, 255, 255, 1)
  Fill=RGBA(41,114,182,1)
  Height=35
  FontWeight=Normal
  ```

  Secondary buttons

  PowerApps Formula

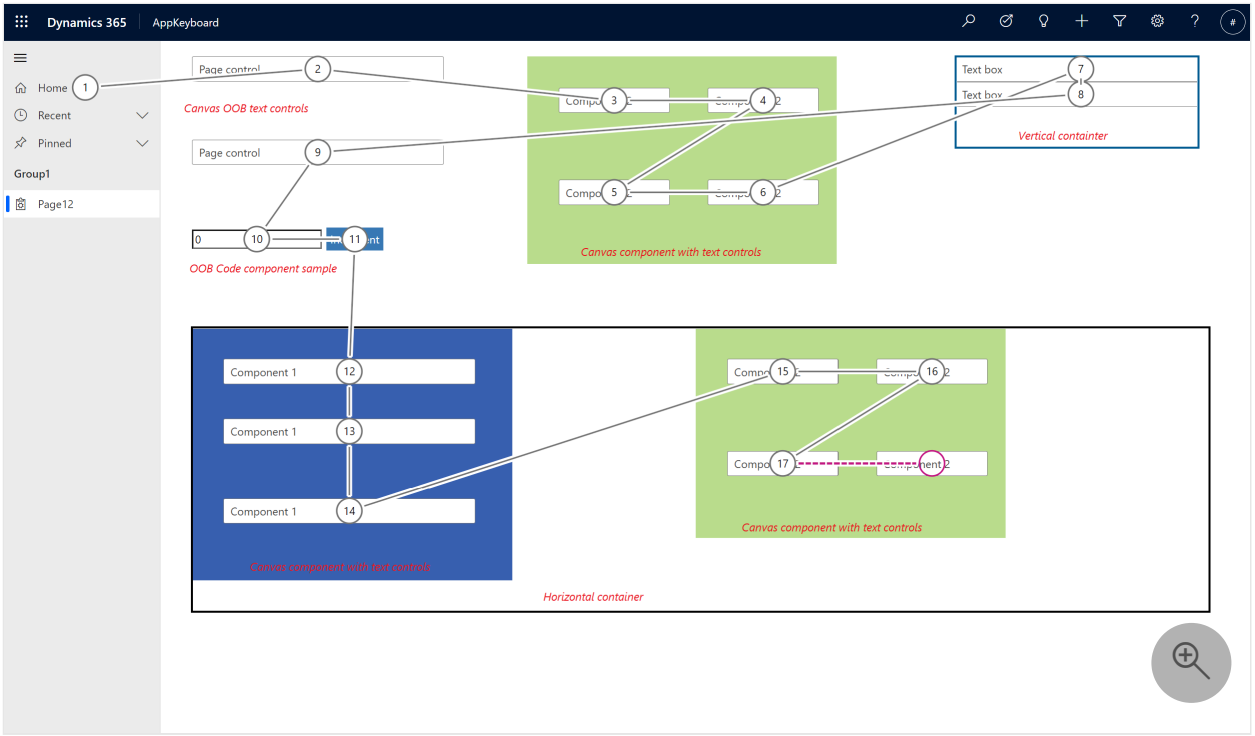
  ```
  Color=RGBA(41,114,182,1)
  Fill=RGBA(255, 255, 255, 1)
  BorderColor=RGBA(41,114,182,1)
  Height=35
  FontWeight=Normal
  ```

Tab navigation and keyboard accessibility for custom pages

Custom pages follow the same tab navigation design that's used by the hosting model-driven app. Visually aligned semantic HTML structure helps users navigate the custom pages seamlessly when using a keyboard or a screen reader. Unlike stand-alone canvas apps, custom page controls, and other UX elements don't need explicit tab numbers assignments. Modern controls don't have a `TabIndex` property and utilize the semantic HTML structure for navigation.

Various elements like controls, canvas and code components, containers, and so on, can be tabbed based on their position in the custom page layout. The tab navigation follows Z order navigation. Individual tab-stops inside larger grouping elements like components, containers are navigated first before the tab moves out to the next element in the document object model (DOM) tree.

Here's an example navigation with the page containing controls, code, and canvas components and containers.



> **Note**
>
> Overlapping controls and elements on the custom page don't have their DOM merged so tab stops can be out of sync from the visual layout. The same is true for the dynamic element positioning using formulas.

See also

Model-driven app custom page overview

Using PowerFx in custom page

Building responsive layout

Add a custom page to your model-driven app

Feedback

Was this page helpful? Yes No

Provide product feedback

Navigate to a custom page using client API

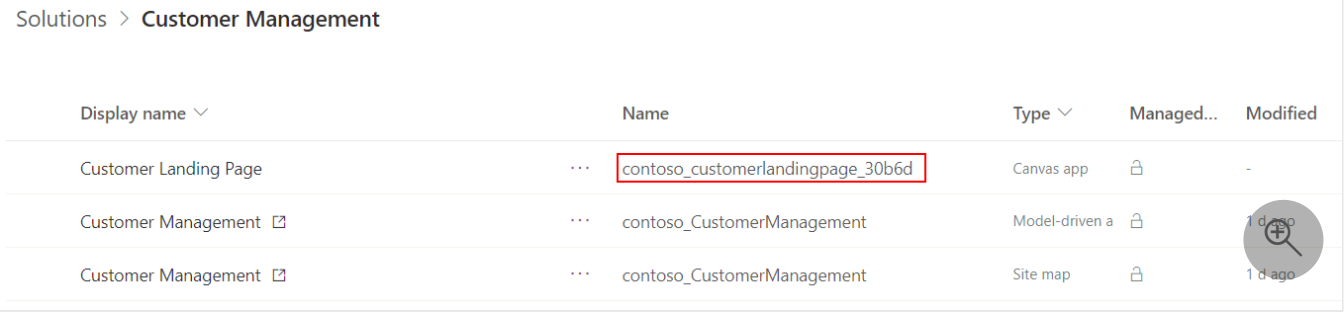
This article provides examples of navigating from a model-driven app page to a custom page using Client API. Learn how to open custom pages as full-page, dialog, or pane views in model-driven apps.

This article outlines the steps to use client API to open a custom page as a full-page, dialog, or pane. It provides examples of **custom** as a `pageType` value in navigateTo (Client API reference).

> **Important**
>
> Custom pages are a new feature with significant product changes. They currently have a number of known limitations outlined in Custom Page Known Issues.

Find the logical name

Each of the following client API examples takes the logical name of the custom page as the required parameter. The logical name is the **Name** value of the page in the solution explorer.



Open as an inline full page without context

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the page.

JavaScript

```
// Inline Page
var pageInput = {
    pageType: "custom",
    name: "<logical name of the custom page>",
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
```

```
    .then(
        function () {
            // Called when page opens
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open as an inline full page with a record context

This example uses the `recordId` parameter within the NavigateTo function to provide the custom page with the record to use.

JavaScript

```
// Inline Page
var pageInput = {
    pageType: "custom",
    name: "<logical name of the custom page>",
    entityName: "<logical name of the table>",
    recordId: "<record id>",
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when page opens
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

The `Param` function within the custom page retrieves the value and uses the Lookup function to retrieve the record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

> **ⓘ Important**
>
> The `recordId` parameter must be a GUID because it updates the URL and an app start from the URL validates the `recordId` is a GUID.

Open as a centered dialog

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the custom page. This mode supports the sizing parameters similar to the Main form dialogs.

JavaScript

```
// Centered Dialog
var pageInput = {
    pageType: "custom",
    name: "<logical custom page name>",
};
var navigationOptions = {
    target: 2,
    position: 1,
    width: {value: 50, unit:"%"},
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open as a side dialog

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the custom page.

JavaScript

```
// Side Dialog
var pageInput = {
    pageType: "custom",
```

```
    name: "<logical page name>",
};
var navigationOptions = {
    target: 2,
    position: 2,
    width: {value: 500, unit: "px"},
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open from a grid primary field link as a full page with record ID

This example uses the `recordId` parameter within the navigateTo function to provide the custom page with the record to use. The `Param` function within the custom page retrieves the value and uses the `Lookup` function to retrieve the record.

You can find a more complete example at Override the default open behavior of data rows in an entity-bound grid.

1. Create a web resource of type **JScript** and update the **name** parameter to be the logical page name. Add the following code to the web resource.

   JavaScript

   ```
   function run(selectedItems)
   {
       let selectedItem = selectedItems[0];

       if (selectedItem) {
           let pageInput = {
               pageType: "custom",
               name: "<logical page name>",
               entityName: selectedItem.TypeName,
               recordId: selectedItem.Id,
           };
           let navigationOptions = {
               target: 1
           };
           Xrm.Navigation.navigateTo(pageInput, navigationOptions)
   ```

```
            .then(
                function () {
                    // Handle success
                }
            ).catch(
                function (error) {
                    // Handle error
                }
            );
    }
}
```

2. Customize the table ribbon **CommandDefinition** for **OpenRecordItem** to call the function earlier and include the **CrmParameter** with the value **SelectedControlSelectedItemReferences**.

XML

```
    <CommandDefinition Id="Mscrm.OpenRecordItem">
        <Actions>
            <JavaScriptFunction FunctionName="run"
Library="$webresource:cr62c_OpenCustomPage">
                <CrmParameter Value="SelectedControlSelectedItemReferences" />
            </JavaScriptFunction>
        </Actions>
    </CommandDefinition>
```

3. In the custom page, override the **App**'s **OnStart** property to use the `Param` function to get the `recordId` and `lookup` record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

> **Note**
>
> After changing the `OnStart` property, you need to run `OnStart` from the App context menu. This function performs only once within a session.

4. Then, the custom page can use the **RecordItem** parameter as a record. The following example shows how to use it in an EditForm.

PowerApps Formula

```
EditForm.Datasource=<datasource name>
EditForm.Item=RecordItem
```

Open from a selected record in editable grid as a centered dialog with record ID

Use an editable grid to trigger the OnRecordSelect event when you want to perform an action after selecting a particular record in a view. This example uses the `recordId` parameter within the navigateTo function to provide the custom page with the record to use. The `getId` method in the GridEntity object retrieves the record ID. The `Param` function within the custom page retrieves the value and uses the `lookup` function to retrieve the record.

1. Enable editable grid control in the table.
2. Create a web resource of type **JScript** and update the **name** parameter to be the logical page name. Add the following code to the web resource.

JavaScript

```
function RunOnSelected(executionContext) {
// Retrieves the record selected in the editable grid
var selectedRecord = executionContext.getFormContext().data.entity;
var Id = selectedRecord.getId().replace(/[{}]/g, "");

// Centered Dialog
var pageInput = {
    pageType: "custom",
    name: "<logical page name>",
    recordId: Id,
};
var navigationOptions = {
    target: 2,
    position: 1,
    width: { value: 50, unit: "%" },
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
```

```
    );
}
```

3. In the custom page, override the **App**'s **OnStart** property to use the `Param` function to get the `recordId` and `lookup` record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

> **Note**
>
> After changing the `OnStart` property, you need to run `OnStart` from the App context menu. This function performs only once within a session.

4. Then, the custom page can use the **RecordItem** parameter as a record. The following example shows how to use it in an EditForm.

PowerApps Formula

```
EditForm.Datasource=<datasource name>
EditForm.Item=RecordItem
```

Related articles

Model-driven app custom page overview
Add a custom page to your model-driven app
navigateTo (client API reference)

Last updated on 03/28/2026

Use Power Fx in a custom page for your model-driven app

Article • 01/23/2025

This article outlines how the common Microsoft Power Fx functions work differently between a standalone canvas apps and a custom page. Functions work differently because a custom page is a component within the model-driven app. Other Microsoft Power Fx formulas continue to behave in the same way.

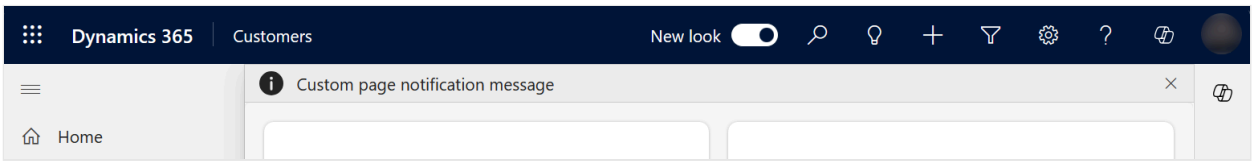
> **Important**
>
> Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in Custom Page Known Issues.

Add notifications to a custom page

A notification can be shown to the user in a custom page by calling the Notify function. When the notification messages appear, they're docked above the default page to stay visible until disabled. If a time-out interval is provided, the notification message disappears after the time out interval. We recommend that you don't use a time-out interval of 10, as this is no longer considered as a time-out interval. More information: Notify function.

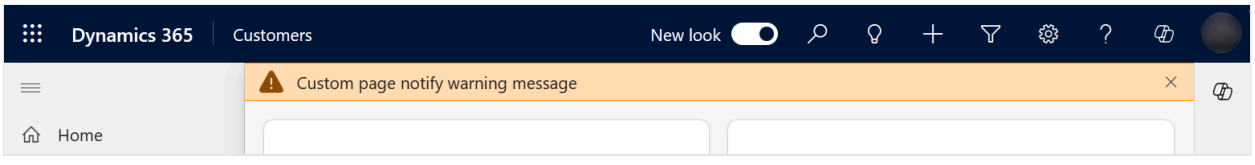
Power Fx

```
Notify( "Custom page notification message" )
```



Power Fx

```
Notify( "Custom page notify warning message", NotificationType.Warning )
```



Navigating to a custom page

This section provides examples of navigating from a model-driven app form to a custom page, navigating from a custom page to other custom pages or model-driven app form using Power Fx.

Navigating from a custom page

The Navigate function allows users to navigate either from model-driven app forms or custom pages. This function is only applicable when the custom page is running within a model-driven app. During custom page authoring or previewing in canvas designer, this function has no effect.

Navigate to another custom page

To navigate from one custom page to another, pass the display name of the custom page as the first parameter.

Power Fx

```
Navigate( CustomPage2  )
```

Navigate to the default view of the table

To navigate to the default view of the table, pass table name as the first parameter.

Power Fx

```
Navigate( Accounts )
```

> ⓘ **Important**
>
> Make sure you add the accounts Microsoft Dataverse table to the custom page before publishing and testing.

Navigate to specific system view of the table

To navigate to a specific system view of the table, pass the GUID of the view.

Power Fx

```
Navigate( 'Accounts (Views)'.'My Active Accounts' )
```

Navigate to the default form of the table

To navigate to the default form of the table, pass the record as the first parameter.

Power Fx

```
Navigate( Gallery1.Selected )
```

Navigate to a specific form of a table

To pass a Dataverse record to a specific form, pass the form name in the second parameter's `Page` attribute.

Power Fx

```
Navigate(
  AccountGallery.Selected,
  { Page: 'Accounts (Forms)'.Account  } )
```

Navigate to a specific custom page with a record input

To pass a Dataverse record to a specific custom page, pass the custom page name in the second parameter's `Page` attribute.

Power Fx

```
Navigate(
  AccountGallery.Selected,
  { Page: 'Account Record Page'  } )
```

In the target custom page, the record is retrieved using `Param` function to get the `etn` and `id` values.

Here's an example of loading the record into an `EditForm` control.

Power Fx

```
AccountEditForm.DataSource = Accounts
AccountEditForm.Item =
```

```
  LookUp( Accounts, accountid = GUID( Param("id") ) )
```

Navigate to the default form of the table in create mode

To navigate to the default form of the table in create mode, pass a Dataverse record created from the Defaults function. Defaults opens the default form with the record as a new record. The `Defaults` function takes the table name to create the record.

Power Fx

```
Navigate( Defaults( Accounts ) )
```

Navigate to the default form of the table in create mode with field defaulted

To navigate to a new record with some fields defaulted, use the `Patch` function to set fields on the default record for the table.

Power Fx

```
Navigate(
    Patch(
        Defaults(Accounts), { 'Account Name': "My company", Phone: "555-
3423" } )
  )
```

Navigate back to the prior page or close a dialog

To navigate back to the last page or to close a dialog, the Back function is called in a custom page. The `Back` function closes the current page and returns to the last model-driven app or custom page in the model-driven app. If the custom page has multiple screens, go to the article Navigating back when custom page has multiple screens.

Power Fx

```
Back()
```

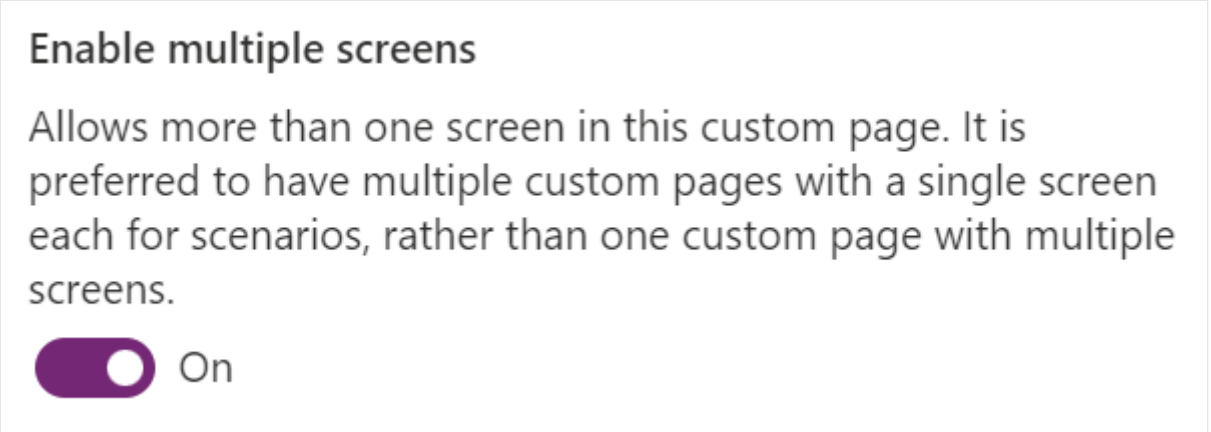
Navigating back when custom page has multiple screens

The default configuration for a custom page is to have one screen. In this case, the `Back` function call closes the custom page unless the custom page is the last in the page stack in model-driven app. The last page is kept open.

An app maker can enable multiple screens in a custom page. These should be considered like full page controls within the custom page that can be stacked. Opening a custom page has no means of specifying the screen to use. When a custom page contains multiple screens, the maker is responsible for managing the screen stacking. Calling the `Navigate` function to a screen adds to the screen stack with the custom page. Each `Back` function call removes a screen from the screen stack. When there's only one screen on the screen stack, the custom page is closed.

Enabling multiple screens

By default a custom page uses a single screen to encourage separation of the app into a screen per page. Single screen can be switched by enabling **Settings** > **Display** > **Enable multiple screens**.



Known issues

- The `Navigate` function doesn't have support for opening a model-driven app or custom page to a dialog. All navigation from a custom page opens inline.
- Navigate function doesn't support opening:
  - A dashboard collection or a specific dashboard.
  - A specific model-driven app form.
- A custom page can only open into the current session's current app tab in a multi-session model-driven app.

See also

Navigating to a custom page using client API

Model-driven app custom page overview

Feedback

Was this page helpful? Yes No

Provide product feedback

Add canvas components to a custom page for your model-driven app

Article • 06/01/2022

This article outlines the use of low-code canvas components to build custom UX for a custom page by utilizing shared canvas component library capability. For code first custom UX extensibility, see Add code components to a custom page for your model-driven app.

> **Important**
>
> Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in **Custom Page Known Issues**.

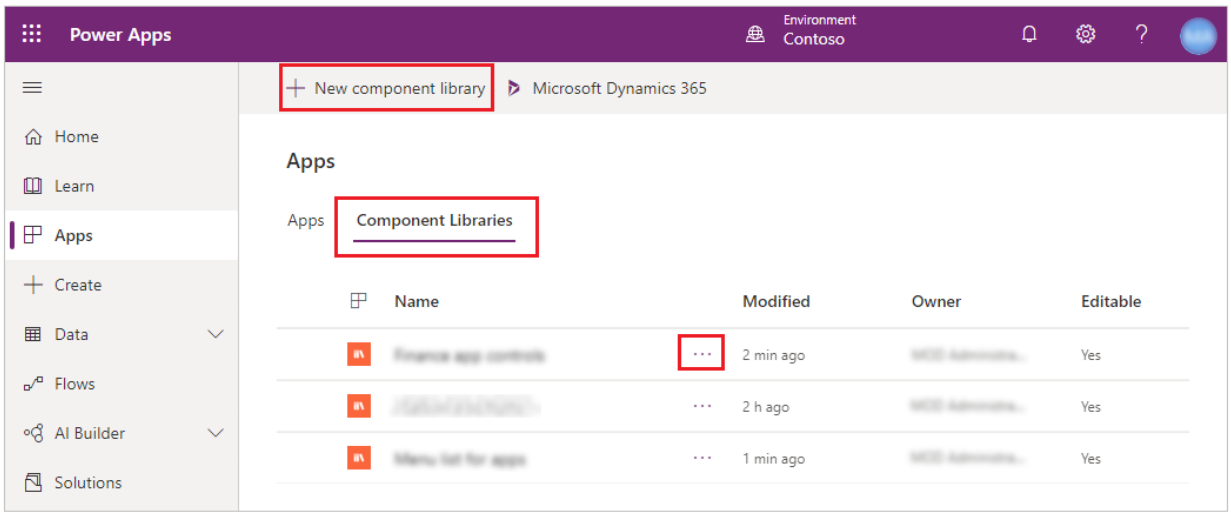
> **Note**
>
> - Custom page currently supports a **limited set of controls** and only the currently supported controls should be used to create canvas components for the custom page.
> - **Modern controls** are currently only supported with custom pages and not with standalone canvas apps. Make sure that the modern control experimental setting is used only for component libraries used with custom pages.

Canvas components provide app makers the ability to create custom components in a low-code fashion. These components cannot only then be reused across custom pages and applications, but also can be centrally updated, packaged, and moved using Microsoft Dataverse solutions. More information: Create a component for canvas apps

Since custom page authoring is limited to only one page, canvas components can only be authored inside a component library. This is different from the standalone canvas apps, which have the ability to create components at the app level.

Create a canvas component using the component library

You can create a new component library or edit an existing one either from the **Solutions** area or the **Component Libraries** tab in the **Apps** area.

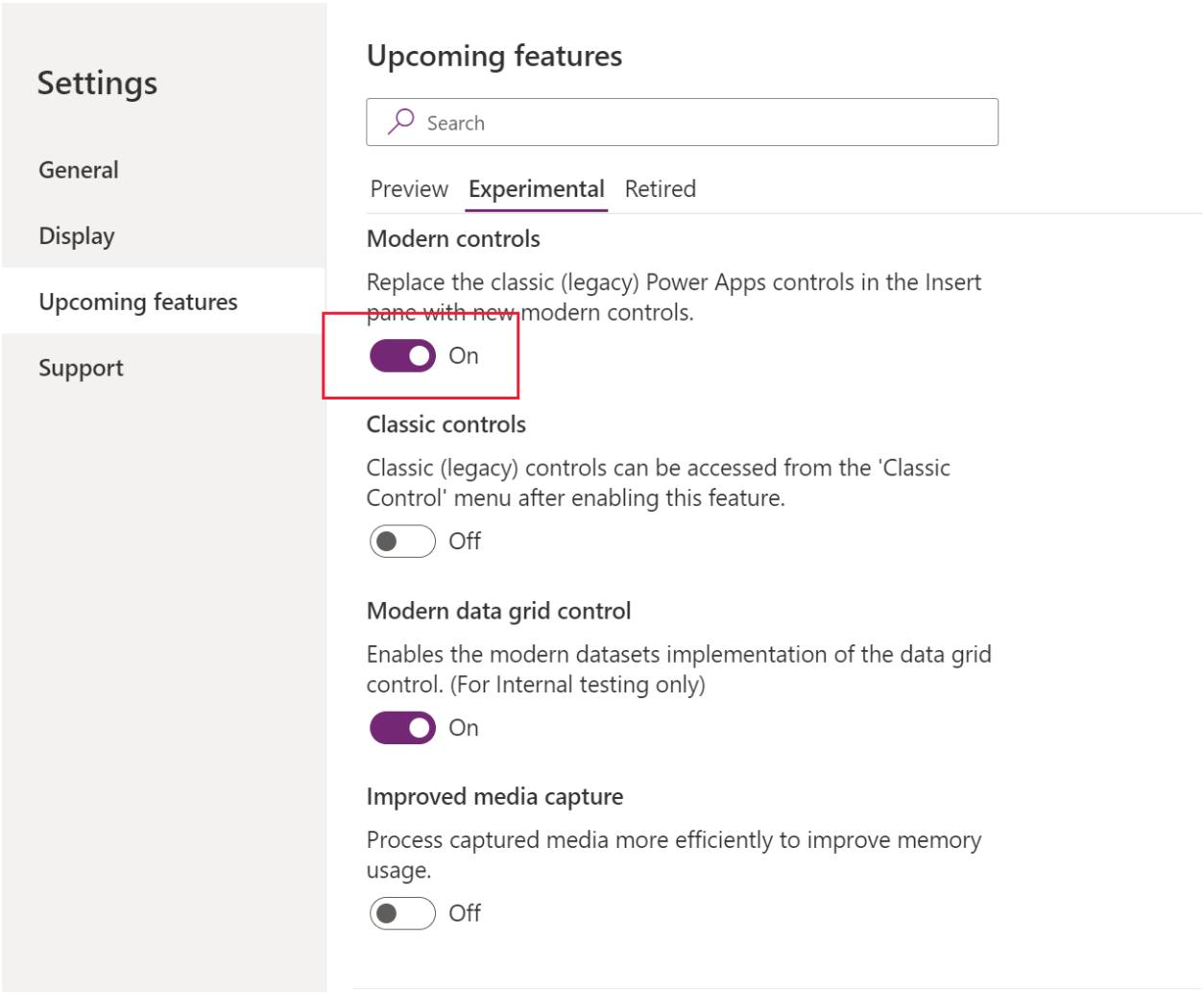


Create a canvas component for use in a custom page

This section explains how to create a canvas component that you can use in a custom page.

Enable modern controls for the component library

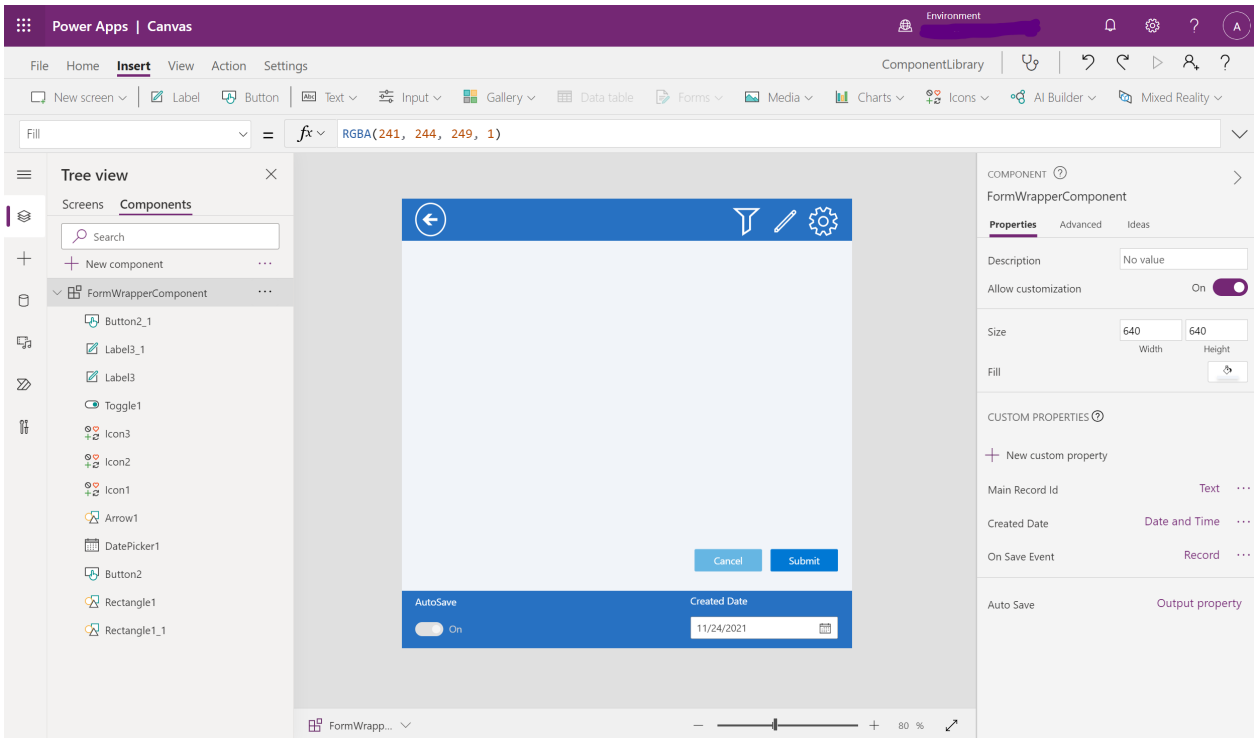
First, enable the **Modern controls** setting in the canvas app designer. To do this, go to **File** > **Settings** > **Upcoming features** > **Experimental**.



Make sure that only the supported set of controls are used to create components for custom page.

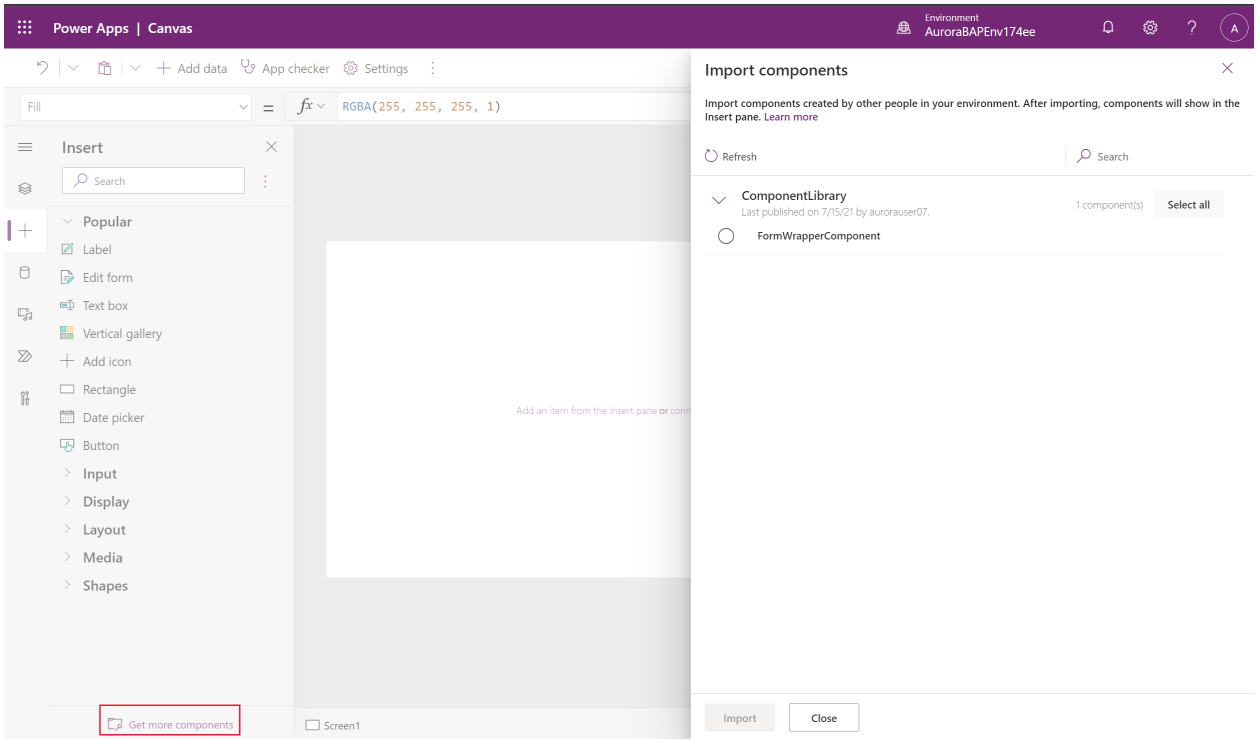
Create a canvas component

Once the modern controls are enabled, you can now create desired components inside the library and publish them once the components are ready.

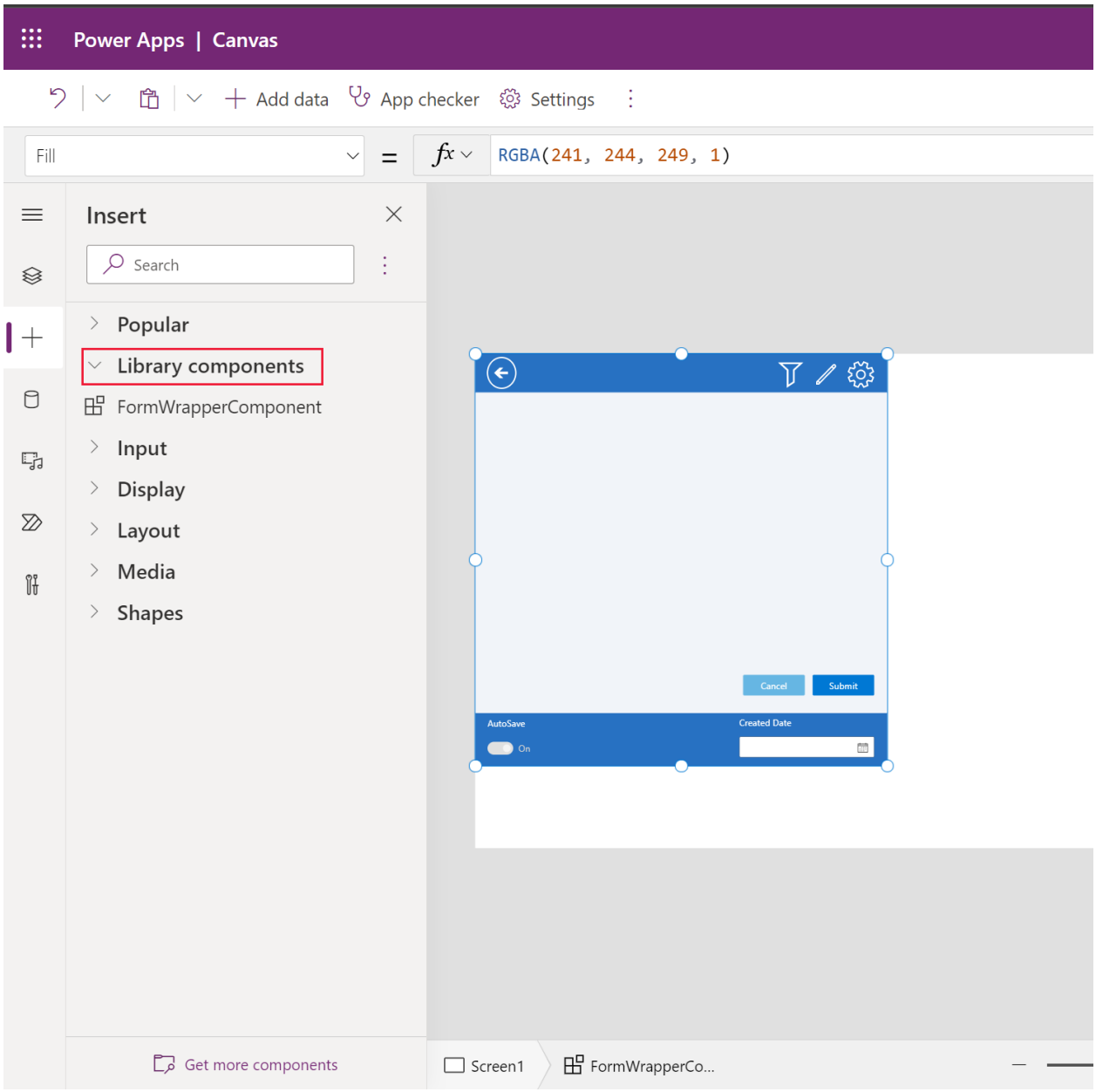


Import and use a canvas component in a custom page

Custom pages can use the components from the canvas component library created or imported in the current environment. You can select **Get more components** at the bottom of the add control left navigation area.

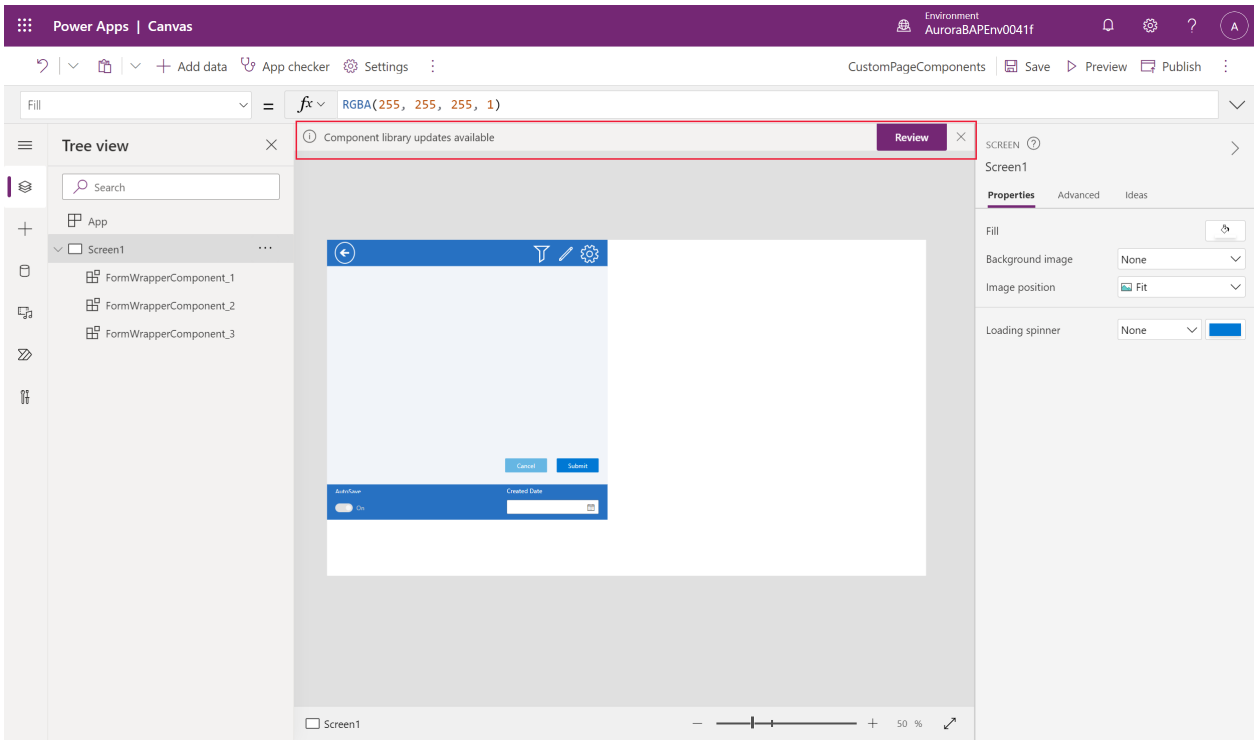


The canvas component is available under the **Library components** section and can be added to the custom page.



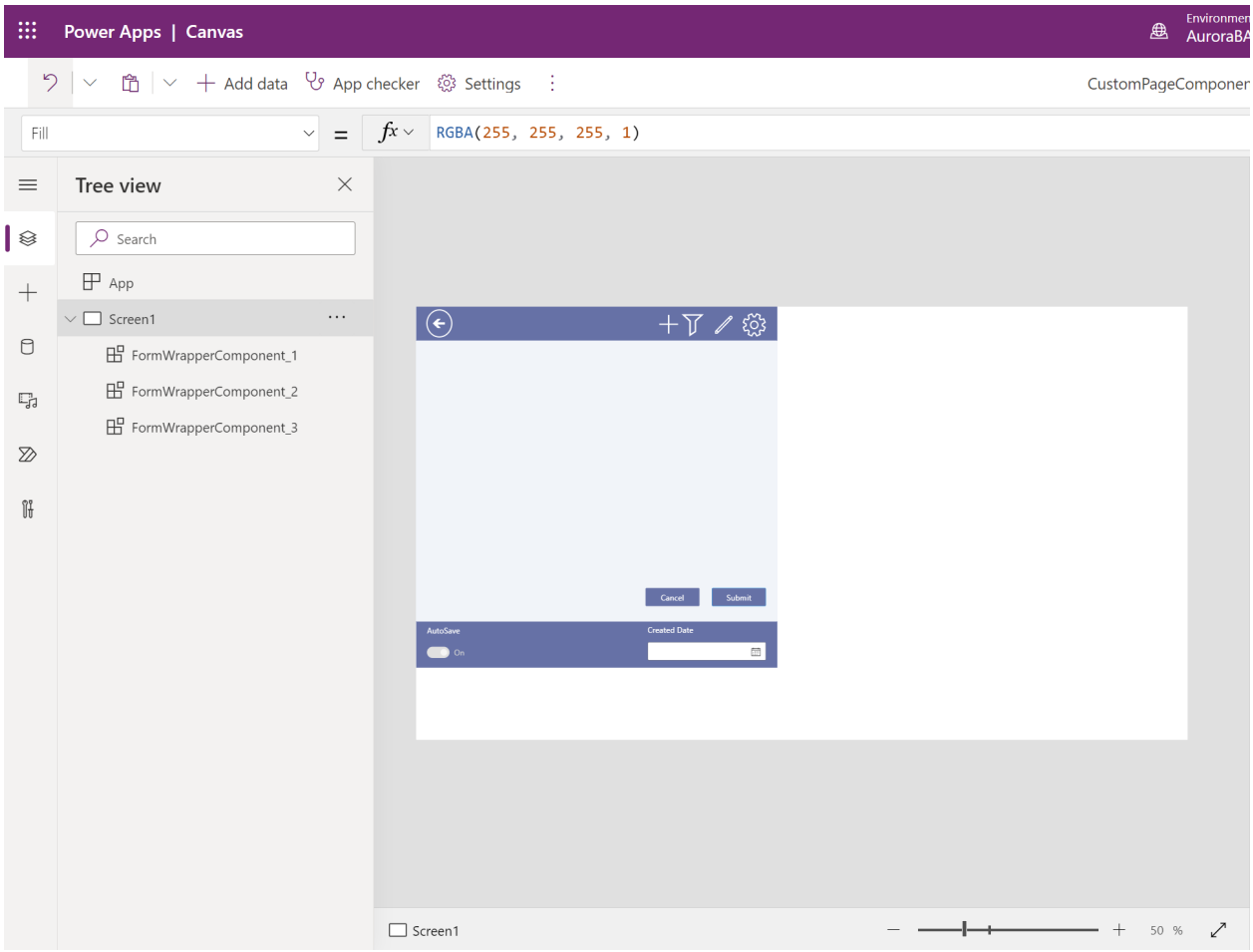
Component updates to the custom page

When the component is added to the custom page via the component library, it maintains the reference to the library. When the library owner publishes the newer library version with updated components, the custom page inline with the canvas app displays an update available message. This message is displayed when the custom page is opened for editing.



Select **Review** to review the updates, and then select **ok** to get the component refreshed with the latest changes from the canvas library.

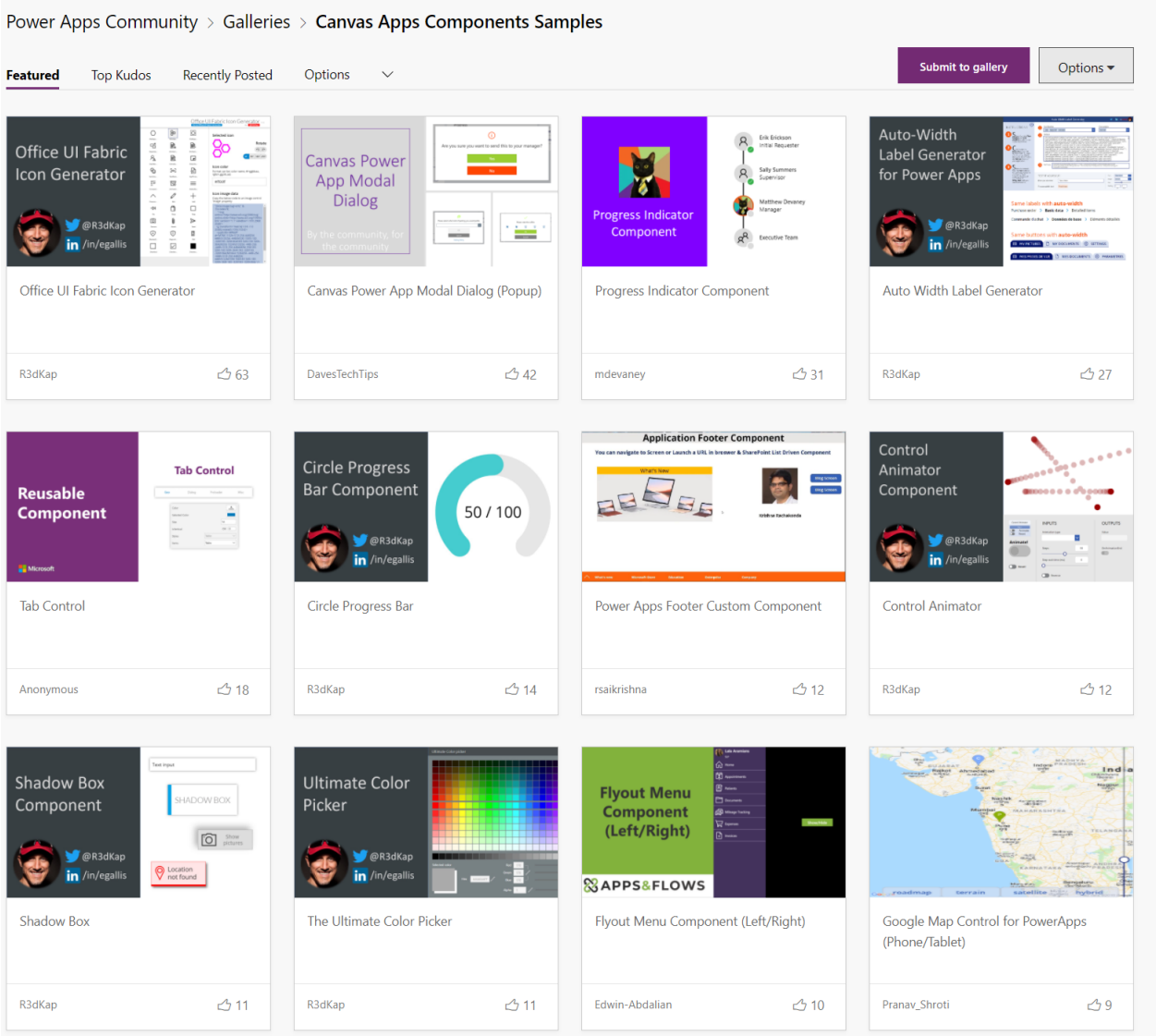
> **Note**
>
> Individual custom pages present in a model-driven app should be opened individually for editing inside canvas app studio to get the latest updates from the shared component library.



You can now publish the custom page and model-driven app to get the latest changes reflected.

Additional canvas component resources

You can evaluate and use canvas component samples from Microsoft and other Power Apps community developers, which are hosted on the Canvas app components gallery.



See also

[Model-driven app custom page overview]

[Add a custom page to your model-driven app]

[Using PowerFx in custom page]

[Overview of Power Apps connectors]

[Add data connection in canvas designer]

Add code components to a custom page for your model-driven app

Article • 06/03/2022

This article outlines the use of code components built by professional developers using Power Apps component framework within a custom page. For low-code custom UX extensibility, see add canvas components to a custom page for your model-driven app.

Important

- Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in **Custom Page Known Issues**.

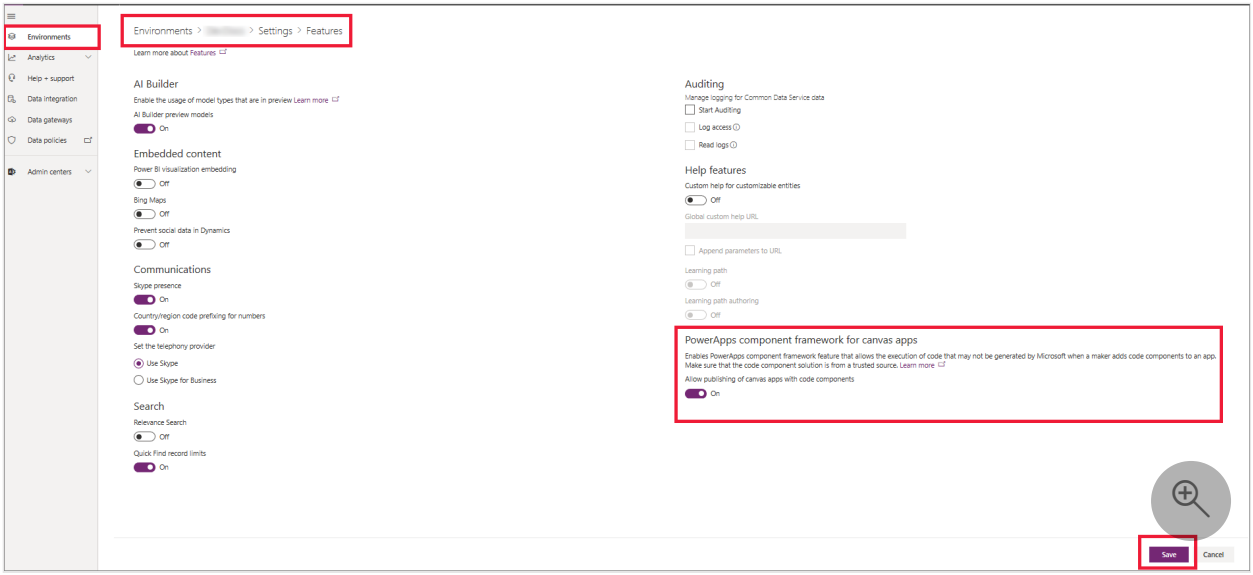
Note

- Custom pages support all the component framework APIs that are currently supported for canvas apps.
- Certain model-driven app specific APIs like **WebAPI** and **Navigation** have published app support for custom pages. These APIs can be used in the code component for custom pages and works on the final published app.
- For individual API support status on each platform, see **Power Apps component framework API reference**

Code components provide professional developers the ability to create custom code components for use within an app. This pro-code extensibility mechanism provides first-class application lifecycle management (ALM) to seamlessly extend the components available to all the app makers across the organization. Code components can be reused across custom pages, canvas, and model-driven apps. They can be centrally updated, packaged, and moved using standard Microsoft Dataverse solutions. More information: Power Apps component framework overview

Enabling Power Apps component framework for custom pages

To use code components inside a custom page, you need to enable Power Apps component framework feature. More information: Enable the Power Apps component framework feature



Create code components for use in a custom page

This section outlines how to create, import, and test code components.

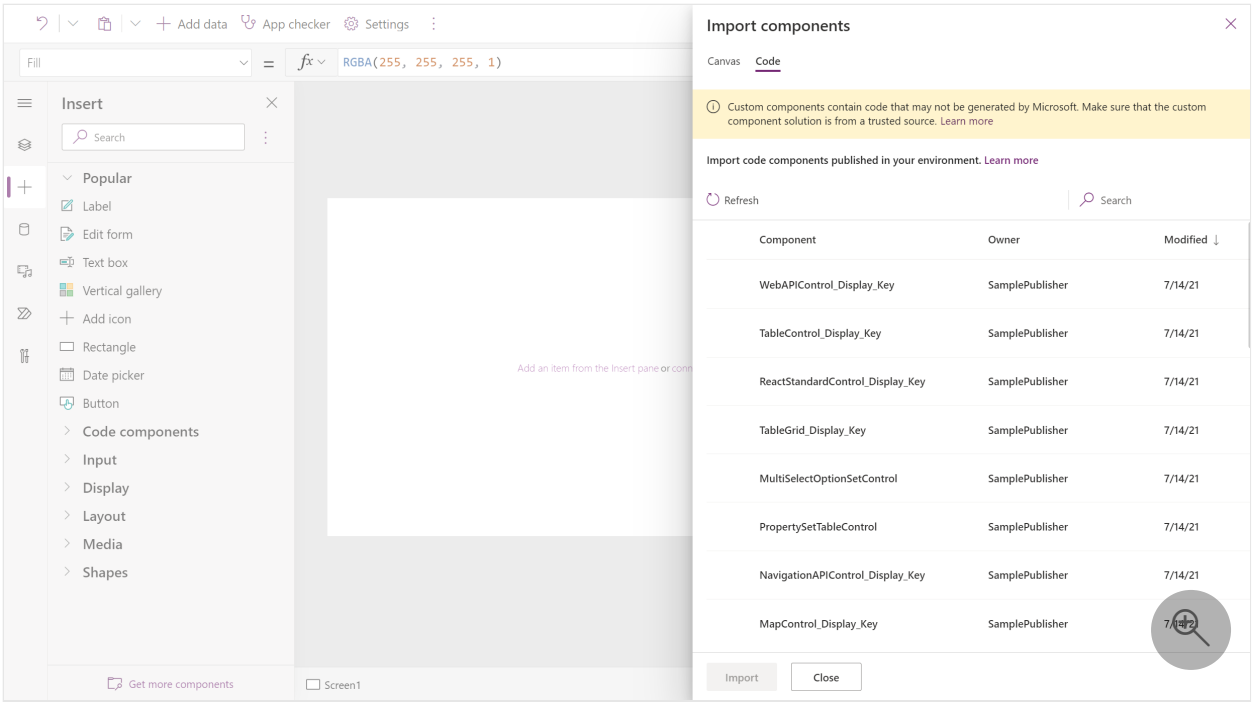
Creating and importing code components in Dataverse

Code components for custom pages follow the same pattern as they do with canvas app. Code components need to be implemented first before they can be added to a custom page. To create a code component, see Create your first component.

You can also try OOB sample components to jump-start. Once you are done implementing your code component, it can be packaged into a solution and added to Dataverse, making it available for use in all the custom pages in line with model-driven and canvas apps. More information: Code components application lifecycle management (ALM).

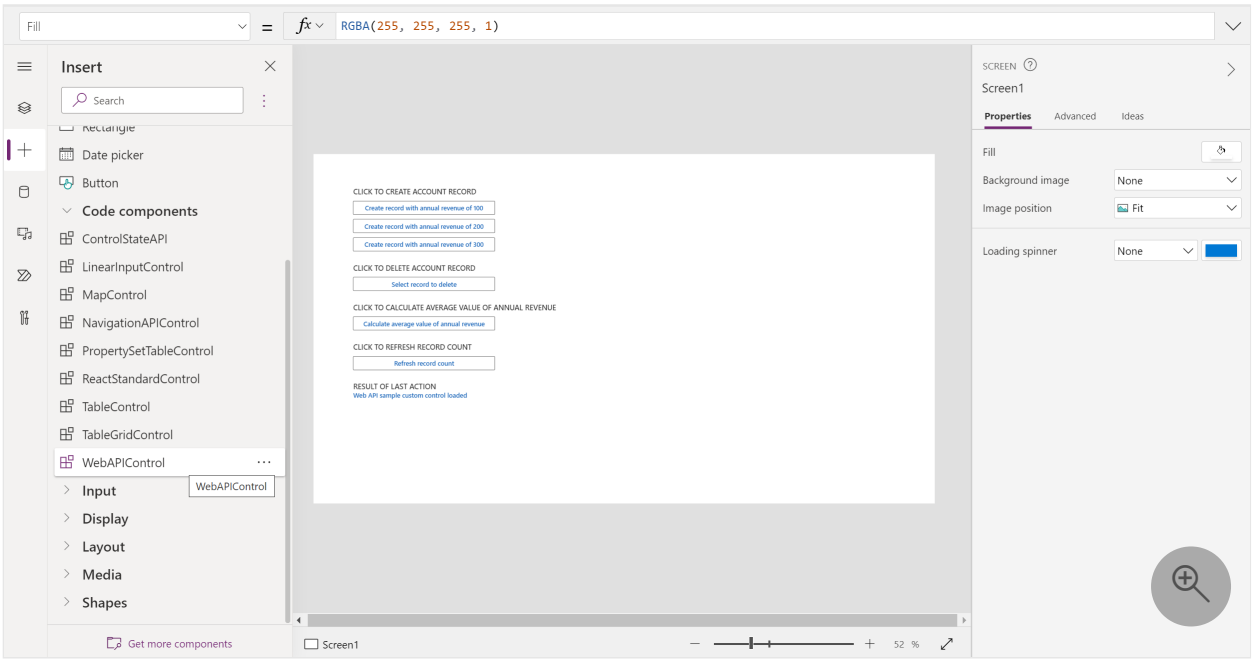
Importing and using code component in a custom page

In an environment, custom pages can use all the code components previously imported into Dataverse using solutions. On the left pane, select **Add (+)**, and then select **Get more components** at the bottom of the page. You will see a code tab on the **Import components** pane showing all code components present in the environment.



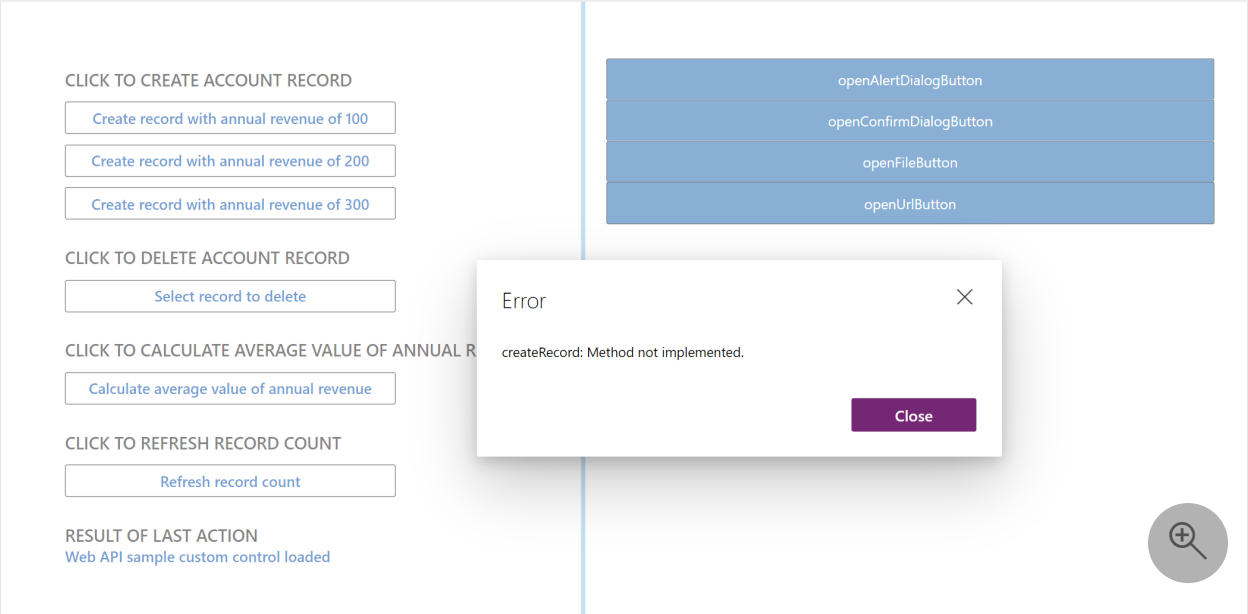
> **Note**
>
> - If the code tab doesn't show up in the **Import component** pane, verify that the Power Apps component framework feature setting for canvas apps is enabled. More information: **Enable Power Apps component framework feature**
> - Also, ensure that you are working on the latest canvas app studio authoring version.

The newly added code components are now available under the **Code components** section. It can be added to a custom page.



Testing the code component inside the studio and published app

Like canvas apps, code components are interactive and can be tested in the custom page studio authoring environment. However, specific APIs like `Web APIs` and `Navigation`, which only have the custom page runtime support when invoked, will show the error message "Method not implemented".

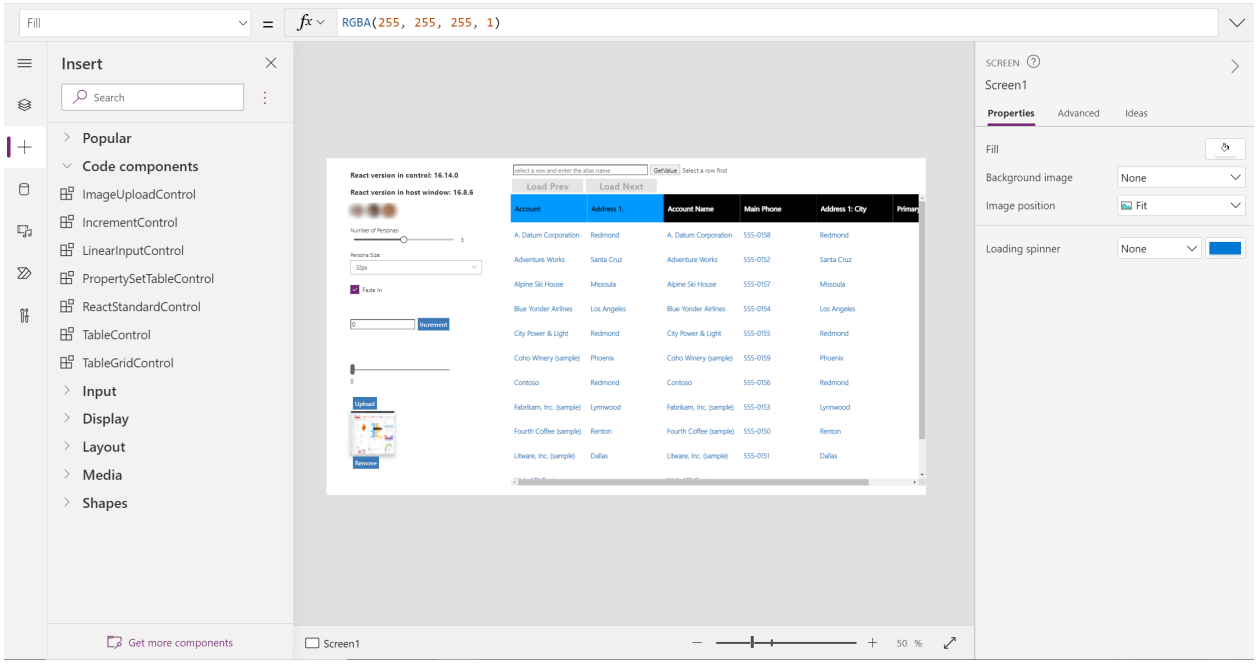


You can dismiss this error and publish the custom page. Then, add this custom page to the model-driven app and publish the app to see the custom page web API in action.

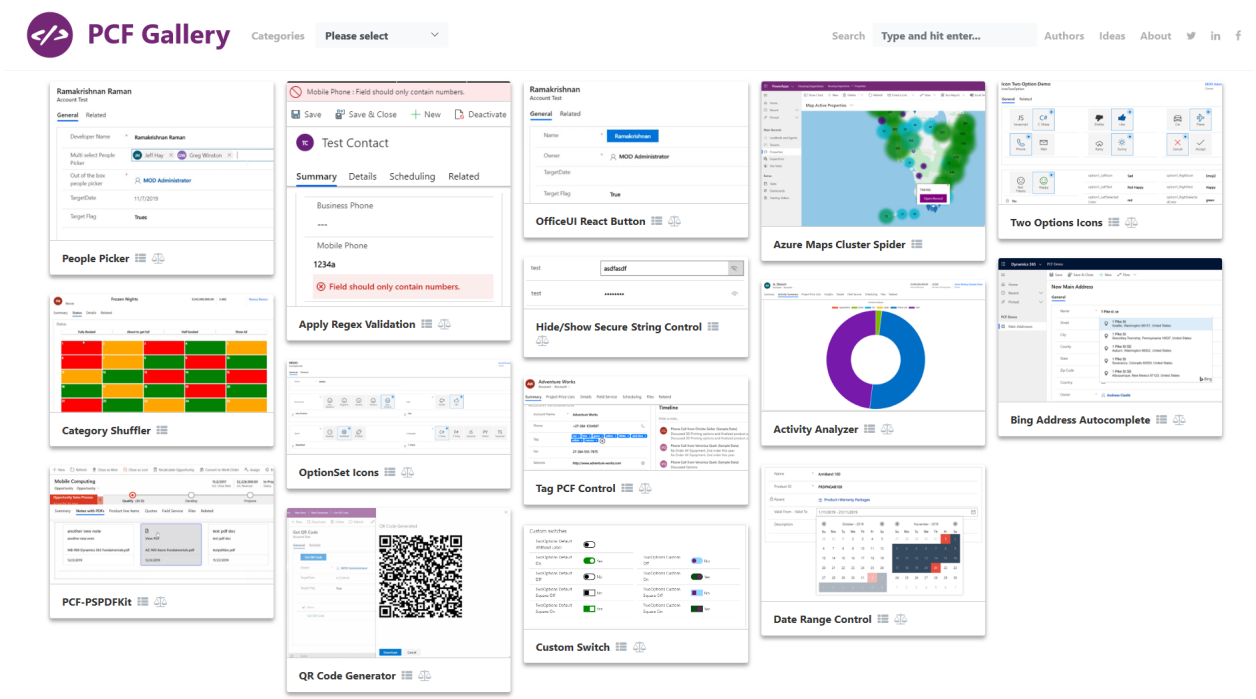
The image below shows the standard Web API sample control and Navigation API control added to custom page working inside a published model-driven app.

Additional code component resources

You can also use other sample components from Microsoft.



Or try some from the Power Apps community gallery.



See also

Model-driven app custom page overview

Add a custom page to your model-driven app

Using PowerFx in custom page

Overview of Power Apps connectors

Add data connection in canvas designer

Overview of connectors for canvas apps

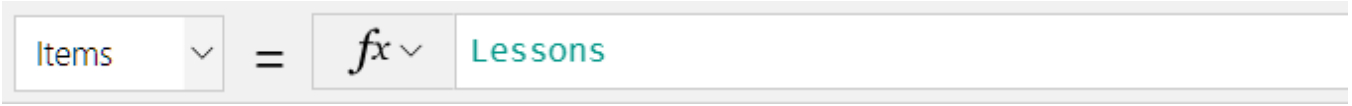
Data is at the core of most apps, including apps you build in Power Apps. You store data in a *data source*, and you bring that data into your app by creating a *connection*. The connection uses a specific *connector* to talk to the data source. Power Apps has connectors for many popular services and on-premises data sources, including SharePoint, SQL Server, Office 365, Salesforce, and Twitter. To get started adding data to a canvas app, see Add a data connection in Power Apps.

A connector might provide **tables** of data or **actions**. Some connectors provide only tables, some provide only actions, and some provide both. Also your connector might be either a standard or custom connector.

> **Note**
>
> Keep the number of connectors in a canvas app to a maximum of 10, and connection references to no more than 20. Going beyond these limits can lead to longer loading times for users when launching the app and can cause problems when saving the app.

Tables

If your connector provides tables, add your data source, and then select the table in the data source that you want to manage. Power Apps retrieves table data into your app and updates data in your data source automatically. For example, add a data source that has a table named **Lessons**, and then set the **Items** property of a control, such as a gallery or a form, to this value in the formula bar:



Specify the data that your app retrieves by customizing the **Items** property of the control that shows your data. Continuing the previous example, sort or filter the data in the **Lessons** table by using that name as an argument for the **Search** and **SortByColumn** functions. In this graphic, the formula set for the **Items** property specifies that the data is sorted and filtered based on the text in **TextSearchBox1**.



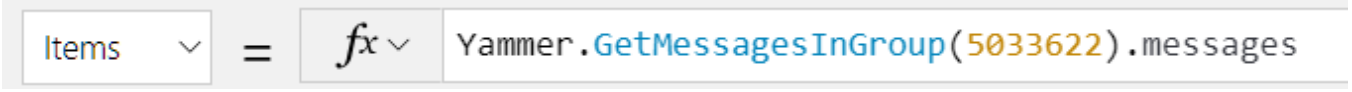
For more information about customizing your formula with tables, see these articles:

Understand data sources in Power Apps
Generate an app from Excel data
Create an app from scratch
Understand tables and records in Power Apps

> **Note**
>
> To connect to data in an Excel workbook, host it in a cloud storage service like OneDrive. For more information, see Connect to cloud-storage from Power Apps.

Actions

If your connector provides actions, select your data source as you did before. Instead of selecting a table as the next step, manually connect a control to an action by editing the **Items** property of the control that shows your data. The formula you set for the **Items** property specifies the action that retrieves data. For example, the app doesn't retrieve any data if you connect to Yammer and then set the **Items** property to the name of the data source. To populate a control with data, specify an action such as **GetMessagesInGroup(5033622).messages**.



To handle custom data updates for action connectors, build a formula that includes the **Patch** function. In the formula, identify the action and the fields that bind to the action.

> **Note**
>
> For action-based connectors, galleries and other controls don't automatically page in more data like they do for tabular connectors. For example, if you bind a tabular data source to a gallery, it retrieves the first set or page of records (for example, 100 records), and then it pages in more data as the control requests it. For an action-based connector, it retrieves a "page" of data, but if the data requested exceeds the page size, the control doesn't automatically get the next page.

For more information about how to customize your formula for custom updates, see these articles:

Patch
Collect
Update

Dynamic schema is a common type of result for action based connectors. Dynamic schema refers to the possibility that the same action might return a table with different columns depending on how it's called. Conditions that might cause the columns in the table to differ include input parameters, the user or role executing the action, and the group in which the user is working, among others. For example, SQL Server stored procedures might return different columns if run with different inputs, or an Azure DevOps instance might use custom fields that aren't available by default.

> **Note**
>
> The connector documentation shows dynamic schema results with this message **"The outputs of this operation are dynamic."** as the return value.

For more information about how to work with dynamic schema in Power Apps, see Working with Dynamic values for an overview and Connect to Azure DevOps from Power Apps for a detailed example.

Popular connectors

This table links to more information about popular connectors. For a complete list, see All connectors.

Expand table

Microsoft Dataverse	Cloud storage **
Dynamics AX	Excel
Microsoft Translator	Office 365 Outlook
Office 365 Users	Oracle
Power BI	SharePoint
SQL Server	Twitter

** Applies to Azure Blob, Box, Dropbox, Google Drive, and OneDrive.

Standard and custom connectors

Power Apps provides *standard* connectors for many commonly used data sources. If Power Apps has a standard connector for the type of data source you want to use, use that connector. To connect to other types of data sources, like a service you built, see Register and use custom connectors.

All standard connectors

Standard connectors don't require special licensing. For more information, see Power Apps plans.

Ask questions about a specific connector in the Power Apps forums, and suggest connectors you want to add or other improvements in Power Apps Ideas.

Security and types of authentication

As you author your app and create a connection to a data source, you might see that your choice of connector offers different ways to authenticate. For instance, the SQL Server connector allows you to use Microsoft Entra Integrated, SQL Server Authentication, and Windows Authentication. Each type of authentication has different levels of security. Understand what information and rights you share with users who use your application. The primary example in this article is SQL Server. However, the principles apply to all types of connections.

> **Note**
>
> - For detailed information about security considerations when using a relational database server, such as Microsoft SQL Server or Oracle, as the data source for an app, see Use Microsoft SQL Server securely with Power Apps.
> - Power Apps doesn't support **External member** identities. For more information, see Properties of a Microsoft Entra B2B collaboration user.
> - Gateway selection isn't supported for custom connectors that use the anonymous authentication type.

Microsoft Entra ID

This authentication is a secure type of connection. For example, SharePoint uses this type of authentication. SQL Server also allows for this type of authentication. When you connect, the Microsoft Entra service identifies you separately to SharePoint on your behalf. You don't have to supply a username or password. As an author, you can create and work with the data source

by using your credentials. When you publish your application, your application user signs in by using their credentials. If the data is appropriately secured on a back-end, your users can only see what they're authorized to see based on their credentials. This type of security allows you to change rights for specific application users on the back-end data source after the application is published. For instance you can grant access, deny access, or refine what a user or set of users can see all on the back-end data source.

Open-standard authorization (OAuth)

This type of connection is also secure. For example, Twitter uses this type of authentication. When you connect, **you must supply your user name and password.** As an author, you can create and work with the data source by using your credentials. When you publish your application and your application user signs in, they must also supply their credentials. Therefore this type of connection is secure as your users must use their own credentials to access the data source service.

Shared connections and secure implicit connections

In a shared connection, the Power Apps author provides the user name and password for the connection when creating the data source in the application. The connection authentication to the data source is then **implicitly shared** with end users. Once the application is published, the connection is also published and available to your users.

Before January 2024, your end users could take the connection that you shared with them and create separate new applications. Your users can't see the user name or password, but the connection is available to them. However, **after January 2024, all newly created shared connections are secured.** To secure existing apps, republish them. The connection is no longer shared with end users. The published Power App talks to a connection proxy. The connection proxy only talks to the specific Power App for which it's linked. The connection proxy limits the actions that are sent to the connections to the ones in the Power App **{Get, Put/Patch, Delete}** for a given data source. If you have an app using the connections published before January 2024, republish your application and unshare any connections with end users that shouldn't have them.

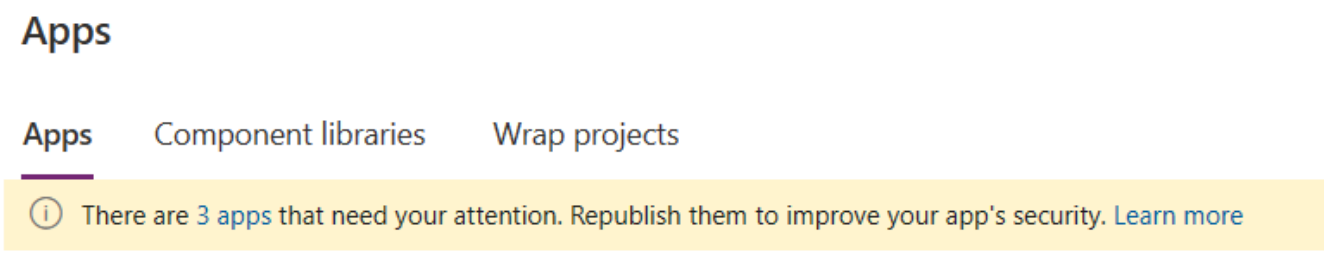
In SQL Server, an example of this type of connection is **SQL Server Authentication**. Many other database data sources provide a similar capability. When you publish your application, your users don't need to supply a unique user name and password.

> ⓘ **Note**

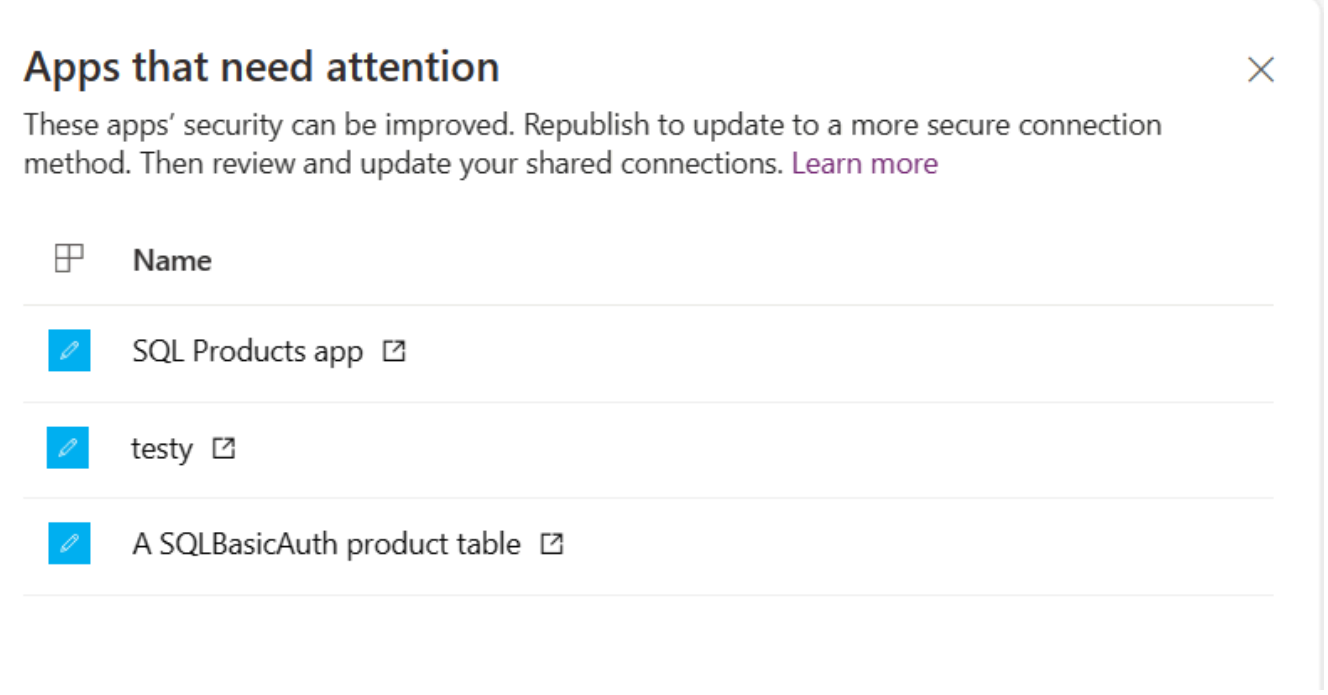
Your end-users might encounter the error message **You do not have correct permissions to use this connection** in the consent dialog. Two situations might cause this error. First, the application might have a shared implicit connection that **isn't** a secure implicit connection. Sharing the connection with the end user resolves this issue but isn't recommended because all shared connections should be secure implicit connections. The author should convert all connections in the application to be secure implicit connections to resolve this issue. Second, the connection might already be a secure implicit connection. Republishing might resolve this issue. If it doesn't, then a product bug should be filed.

Notification to update your apps (secure implicit connections)

If you have applications that might be upgraded to use this feature, you see a message on the **Apps** page. It indicates the number of apps that need your attention.



Select the link and it opens a side panel that lists all of the apps that need attention.



Select the **open** icon to the right of the app name to open and republish it. Continue with the following directions.

Enable secure implicit connections for an existing app

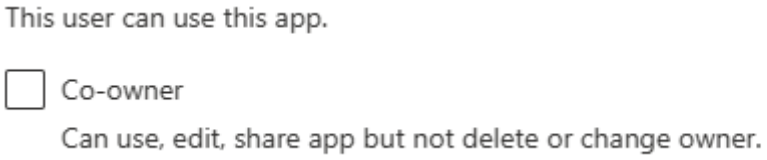
Open an existing app open for editing with implicitly shared connections already published:

1. On the command bar, select **Settings** and search for "**Secure**".
2. Update the feature switch appropriately to enable secure implicit connections.
3. Save and publish the app.

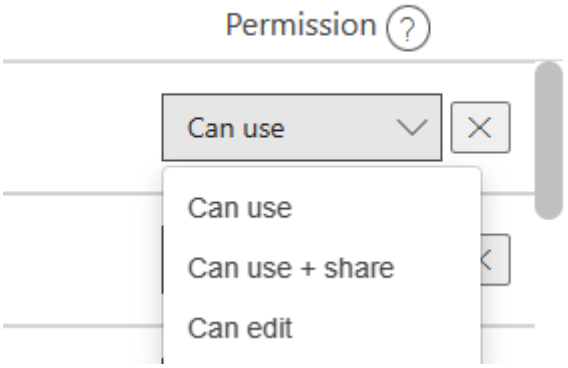
Unsharing

Once you publish the app, follow these steps to verify that sharing works correctly:

- Check if connections are shared with co-owners. If you don't want an end-user to get a connection, then uncheck the **Co-owner** checkbox.



- To verify the feature works correctly, share the app with a different user who isn't an owner. Once you share the app, check the **Connections** list in the **Dataverse** tab in Power Apps for that user. Verify that the user doesn't have a connection available.
- Open the **Sharing** panel to change the end-user's right to the connection. Choose the **X** to remove the user's access to the connection.



Use apps with a new secure implicit connection

When you republish and share your app, end-users don't have access to the connection but work with the hidden proxy connection. Users can't create a new app based on your original connection.

Limitations

1. All types of implicitly shared connections work, such as action and tabular.
2. Server and database names are hidden in network traces but visible in the consent dialog. Column names aren't hidden.
3. For tabular connectors, the feature only limits CRUD actions such as Get, Post, Put, or Delete. If you have permissions to **Put**, then you have access to **Post**.
4. Action based connectors limit based on the specific API being used in the application.
5. Warnings are still enabled in sharing. The warning around implicitly shared connections still warns while in preview. However, your connection with this feature is secure – despite the warning.
6. Publishing to an entire tenant, as opposed to specific groups or individuals isn't supported.
7. There's a known issue when importing an implicitly shared secure connection via a connection reference. The security isn't set properly in the target environment.
8. There's a known issue importing a solution using a service principal, causing import failure. A workaround is to share the connection with the service principal.

Windows Authentication

This type of connection isn't secure because it doesn't rely on end-user authentication. Use Windows authentication when you need to connect to a data source that is **on-premises**. An example of this type of connection is to an on-premises server that has a SQL Server. The connection must go through a gateway. Since it goes through a gateway, the connector has access to all of the data on that data source. As a result, any information that you can access with the Windows credentials you supply is available to the connector. When you publish the application, you also publish the connection and make it available to your users. This behavior means that your end users can create applications by using this same connection and access the data on that machine. Connections to the data source are also **Implicitly Shared** with users that the app is shared with. This type of connection might be valid when your data source only lives on an on-premises server and the data on that source is freely shareable.

Data sources in solutions

Solutions help with application lifecycle management and offer other ways to manage the lifecycle of **data sources**. If a canvas app is in a solution, you can create connection references and environment variables to store information about the data sources. This setup makes it easy to change or reconnect data sources when you move solutions to different environments.

Rename data sources in apps

Learn how to rename data sources in an app, and understand the difference between tabular and action-based data sources. For more information, see Rename Power Apps action-based data sources.

Connection consent dialog

When users open an app that uses connectors for the first time, they see a connection consent dialog for the following purposes.

1. To inform users about the data sources accessed by the app.
2. To outline the actions a connector might or might not perform in an app. For example, for apps using the **Office 365 Users** connector:

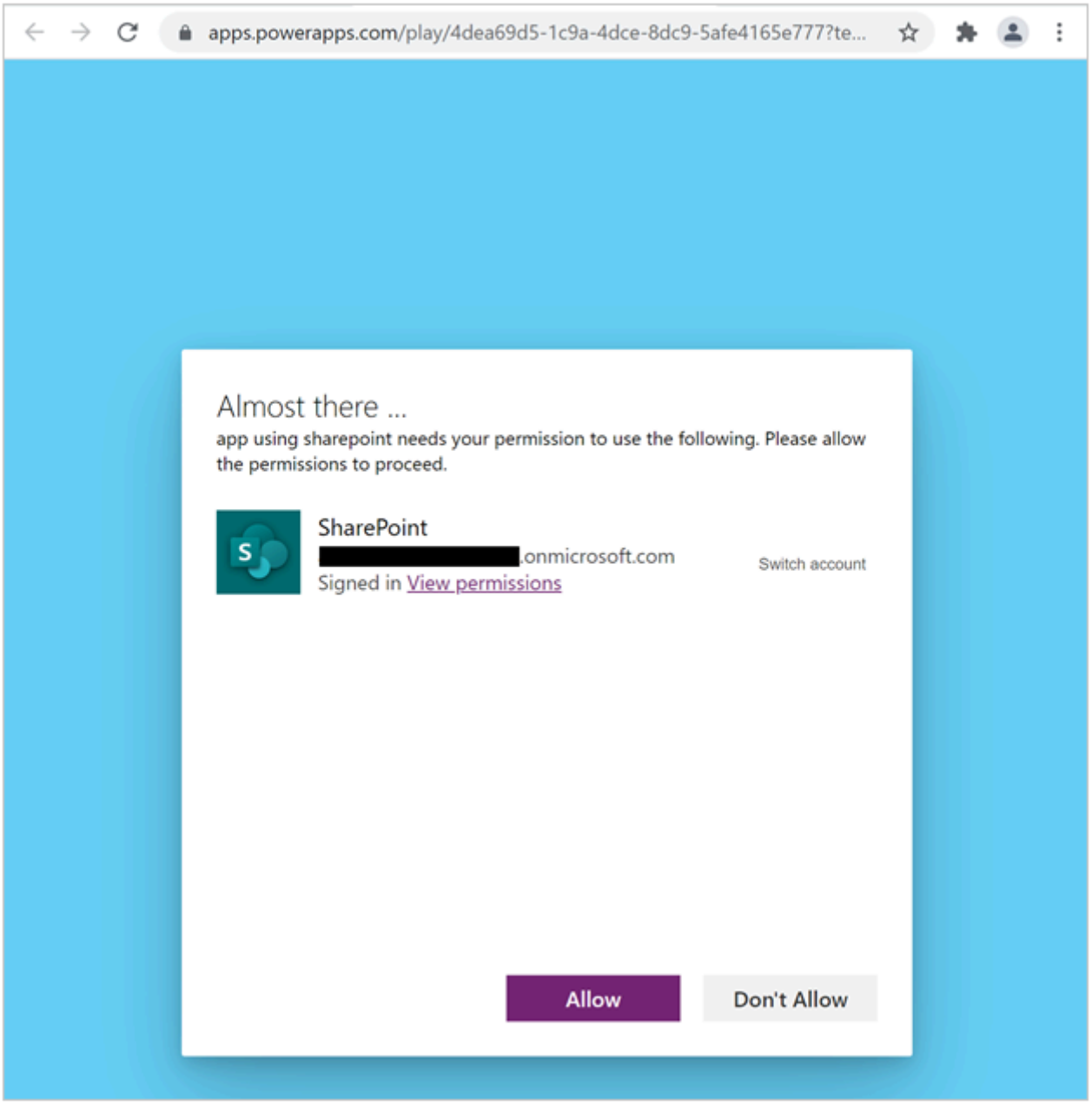
- This app is able to:
  - Read your full user profile
  - Read the full profile of all users
- The app can't:
  - Modify or delete any user-profile information

1. To capture end-user consent to connect to the data sources that the app uses.
2. To facilitate manual end-user authentication, when needed.

For some connections, Power Platform can automatically authenticate a user to access a data source. However, if the automatic sign-in fails, this dialog prompts users to fix a connection by manually signing in. Power Platform can only attempt automatic sign-in for a connection when a data source preauthorizes Microsoft's Azure API connections service principal, granting it permission to perform single sign-on for a user when a connection is created. For more information on single sign-on, see What is single sign-on (SSO)?

For model-driven apps that use custom pages, when there are multiple custom pages in an app, the consent dialog asks for data permissions for all of the connectors in all the custom pages even if users don't open them.

The following image is an example of the connection consent dialog for an app connecting to a SharePoint site.



For select connectors, admins can suppress this dialog and consent on behalf of end users to connect to a data source. The following table explains which types of connectors the consent dialog might be suppressed for an app.

> **Note**
>
> If an admin suppresses the consent dialog but the platform can't perform single sign-on for an end-user, the dialog is presented to the user when they launch the app.

Expand table

Connector type	Consent dialog suppressible?	Reference
Microsoft connectors that support single sign-on (such as SharePoint, Office 365 users)	Yes	Power Apps admin cmdlet
Connector accessing a non-Microsoft, partner service, such as Salesforce	No	Not applicable
Custom connectors using OAuth with Microsoft Entra ID as the identity provider. These custom connectors are built by organizations, and are only accessible by the users within the organization (for example, built by Contoso for only Contoso users)	Yes	Manage Connections

Microsoft Power Platform can only suppress the consent dialog for connections to data sources where all the following conditions are true:

1. The data source doesn't require an explicit consent UI.
2. The data source preauthorizes Microsoft's Azure API connections service principal to enable single sign-on.
3. An admin configures an app to suppress the consent for the preceding connections.

Microsoft's Azure API connections service principal preauthorization exists for Microsoft's first-party data sources. Custom applications registered in a Microsoft Entra tenant that are used by custom connectors might configure this preauthorization. An admin manages consent suppression on a per-app basis (as opposed to connector basis), so suppression is managed at the most granular app experience level. This level of granularity prevents consent suppression for an organization's "approved apps" from inadvertently suppressing consent for apps that aren't approved or reviewed.

Last updated on 01/13/2026

Localize labels, messages, and tooltips on a custom page

Article • 03/13/2023

[This topic is pre-release documentation and is subject to change.]

This article outlines how to add localized content to a custom page by uploading web resource RESX files for each language into your solution.

This article shows you how to use Power Fx to set the content of a control on a custom page. You can use similar steps for tooltips and any messages that use a text control on the custom page.

Localizing labels on a custom page

Providing localized content on a custom page in a model-driven app is vital when building apps that support global users or a multilingual organization. Translation can be done by uploading text files in a .resx format as a web resource. The files contain the translated text that you want to use for labels on your custom pages.

The process has the following steps:

1. Enable languages for your environment.
2. Add localized RESX web resource files into your solution.
3. Add the localized files to your custom page.
4. Manage localized web resources.
5. Add the localized strings to your controls.
6. Run your app with localized content.

Enable languages for your environment

To add languages for a custom page, you need to first enable the languages you want to support in the model-driven app where your custom page will run. Make sure you have enabled languages from the Power Platform admin center. More information: Enable the language

After completing the steps to enable languages in the environment, create a RESX file for each language in the proper format for the strings you want localized on your custom page for your model-driven app.

After you have the created the RESX files with the localized strings, add them into your solution as a web resource. There are two ways to accomplish this task, the first is to open your solution in make.powerapps.com and select the option to add a web resource to your solution in your environment or you can add a web resource while working on a custom page. More information: RESX web resources

Adding a web resource while working on a custom page is covered later in this article in the section Add localized files to your custom page.

Localized content will be based on the user or organizational locale for the application the custom page is running. The language for your application is based on user or admin defined language preferences in the user app settings or admin settings in the Power Platform admin center. If neither has been set, the base language set when the environment was created is used.

> **Note**
>
> Content shown in a Power Apps Control Framework (PCF) control will localize to the applications locale language without additional changes needed on the custom page. More Information **Localizing PCF Controls**

Add localized RESX Web resource files to your solution

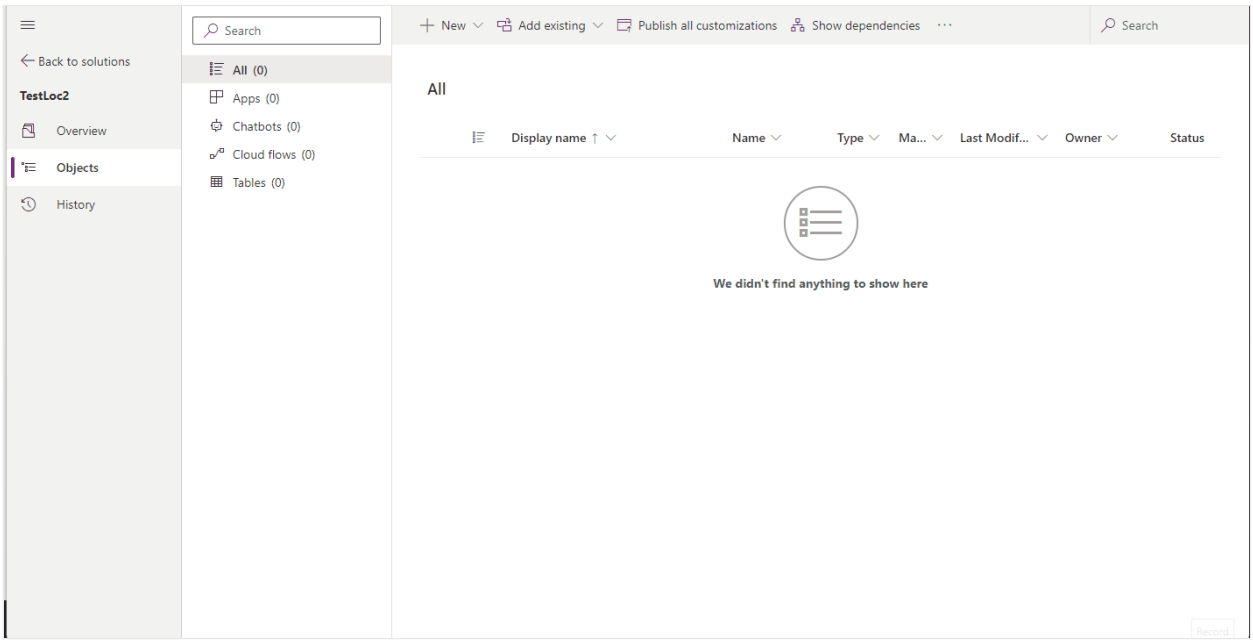
1. Go to make.powerapps.com, select the **Solutions** area, and then open the solution you want.
2. On the command bar, select **New** > **More** > **Web Resource**. This opens a panel to add your RESX web resource file.

All files added to your solution need to follow a specific format that includes **{filename}.LanguageID.resx**, where the language ID is the numeric value for that language. This format is necessary because the framework relies on the naming convention to identify which resource file should be used to associate the appropriate localized string to a label control.

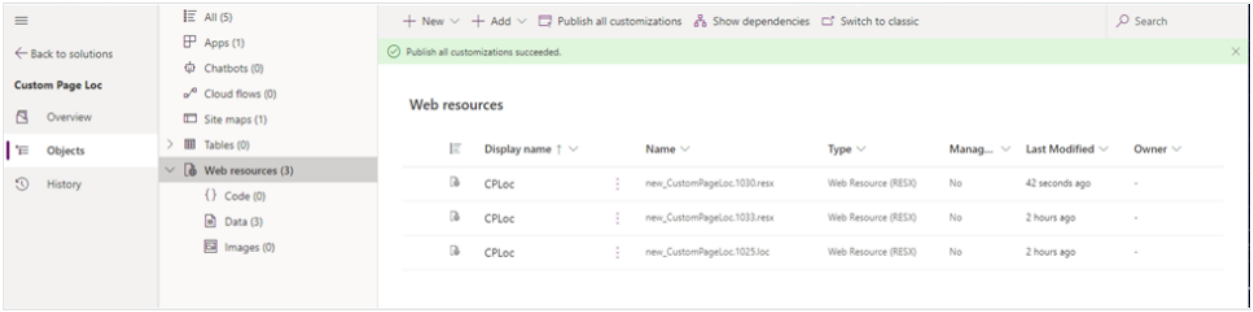
> **Important**
>
> The RESX {filename} must be the same for all your files.

For example, all localized string files have the name `CustomPageLoc.LanguageID.resx`, which means for English it is `CustomPageLoc.1033.resx`, for French `CustomPageLoc.1036.resx`, and for Arabic `CustomPageLoc.1025.resx`. The display name for every file added must also be the same name.

In the example below all files have `CPLoc` as the display name. *The display name must be the same to ensure all localized content is used properly on a custom page for each language*. We recommend that you add the corresponding name for the language in the **Description** field to help identify the content when you view the files in the custom page designer. Also, you'll need to add the same language's RESX file that you're working in. For example, if you are authoring in English, be sure to add the English RESX file.



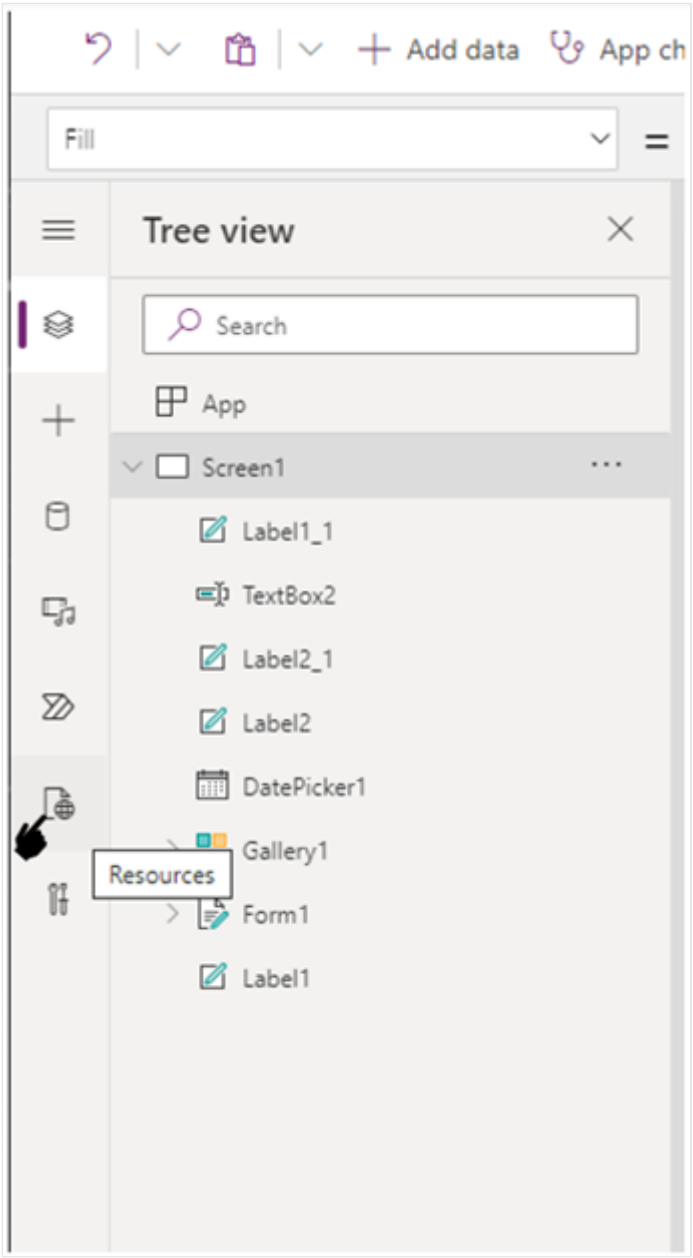
After you have added your web resource to your solution, make sure to publish the changes. Publishing ensures the file is available to add to your custom page. Here's an example of the web resource RESX files that have been added to a solution and are now available to add to a custom page.



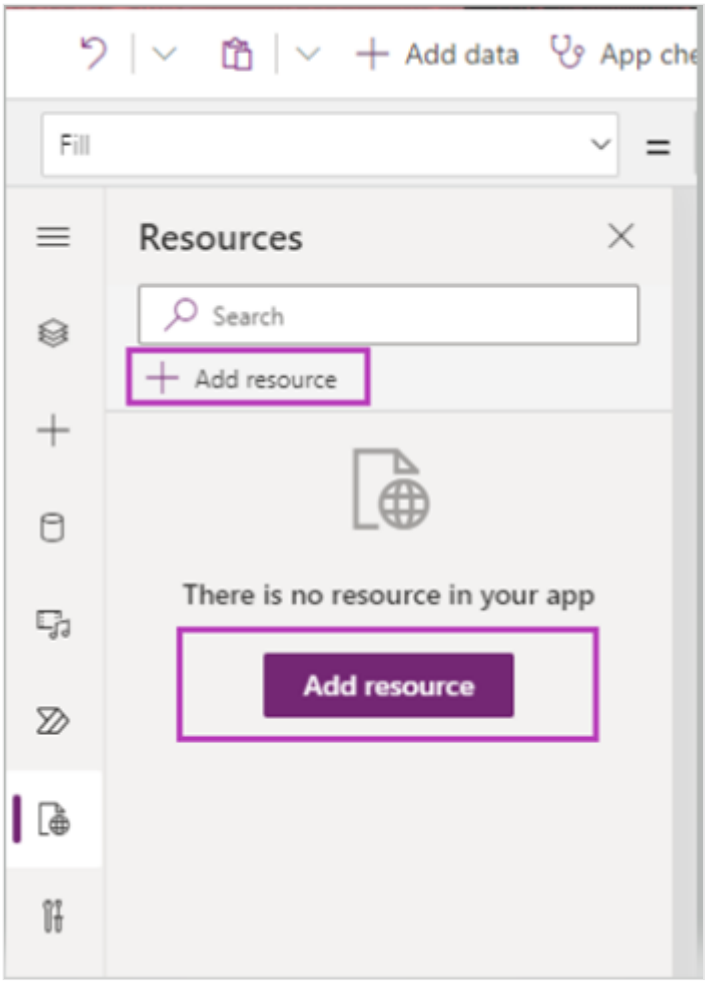
Add localized files to your custom page

Add localized files to your custom page when you're creating or editing a page.

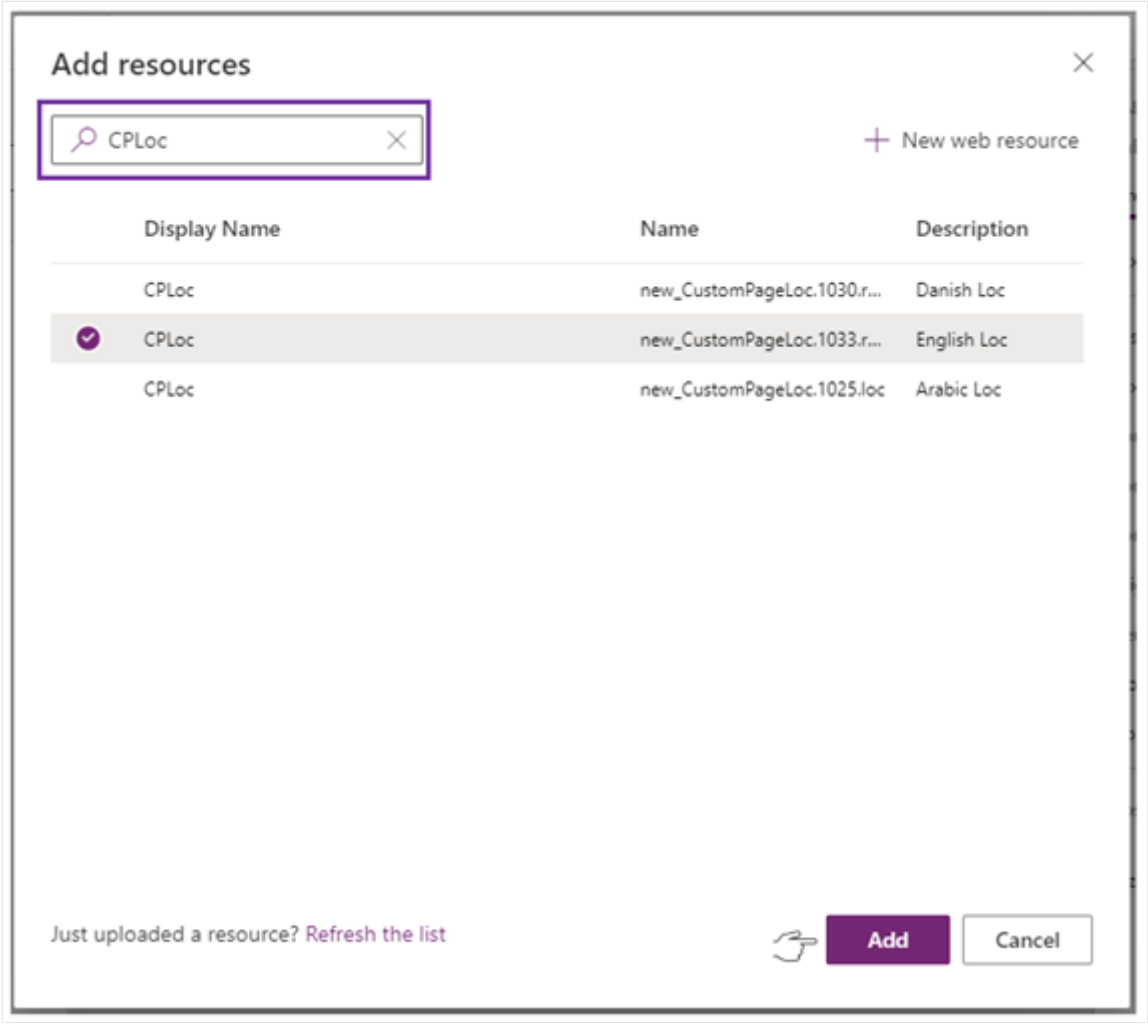
To add a language to your page, select the **Resources** icon on the left to open the **Resources** pane.



Select **Add resource** in either one of two places.



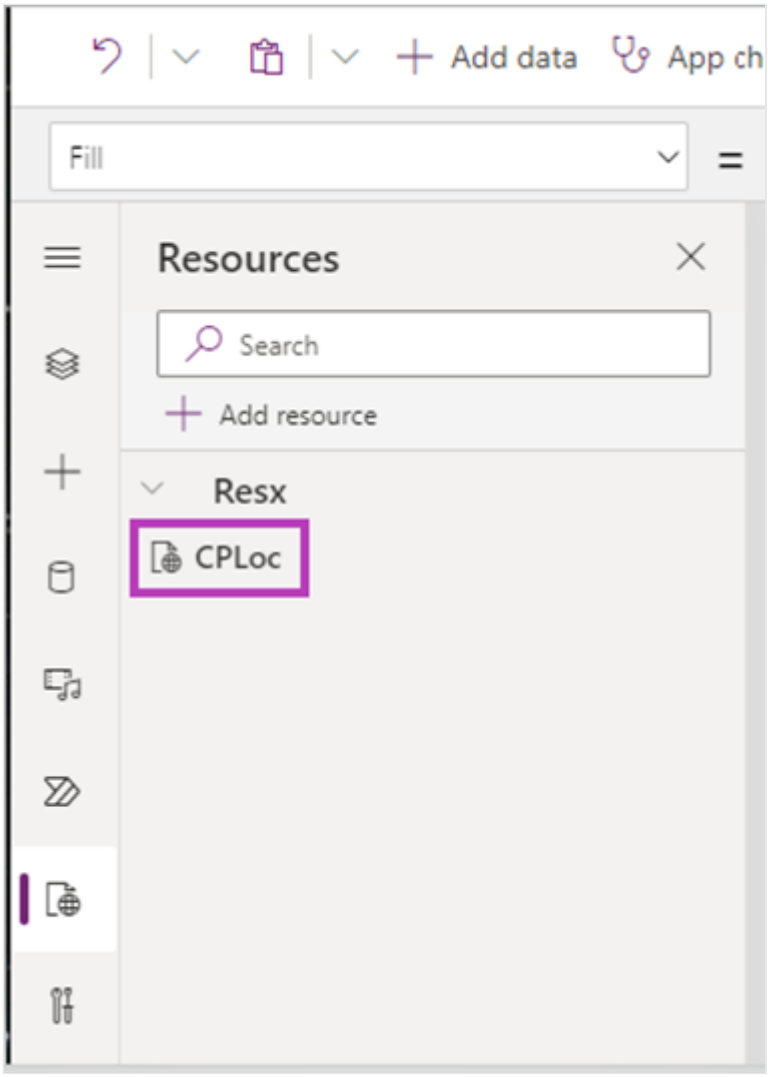
A dialog opens that lists all the available web resources in your solution. To find your files quickly, you can use the search at the top of the dialog to search for the **Display name** that displays all your localized RESX files. You only need to select and add the file that matches the language that you are using to author your page. In this example, the English file is the only file you need to add.



> **Note**
>
> It isn't necessary to add every file-you only need to add one. The solution and platform maps the uploaded languages based on the display name to the language web resource files that you added to your solution. When you add multiple files, a dialog is displayed with an error message, which states that files have already been loaded to your page.

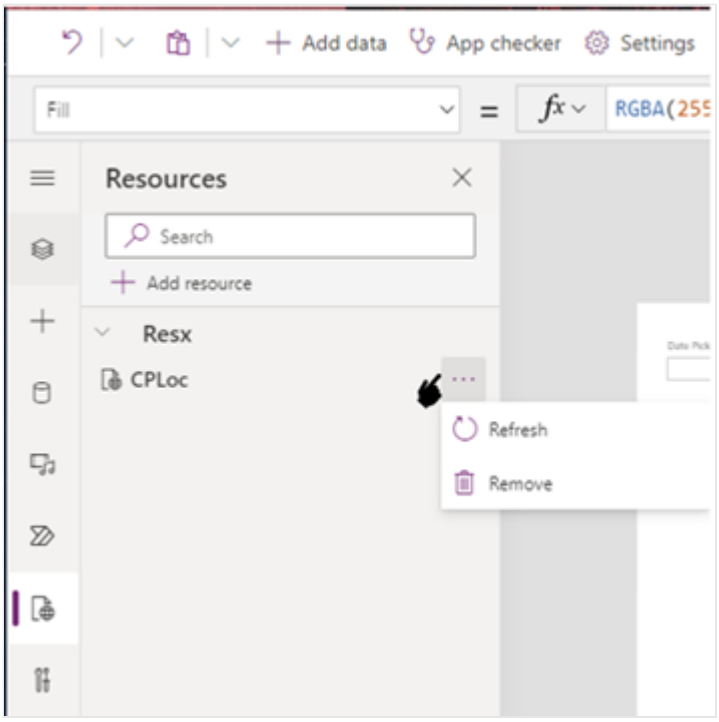
If you don't see the RESX web resource with the language you want to add in the dialog, you'll need to add the file to your solution using the steps described earlier in this article.

After selecting **Add**, the panel closes, and the web resource is listed in the **Resources** pane.



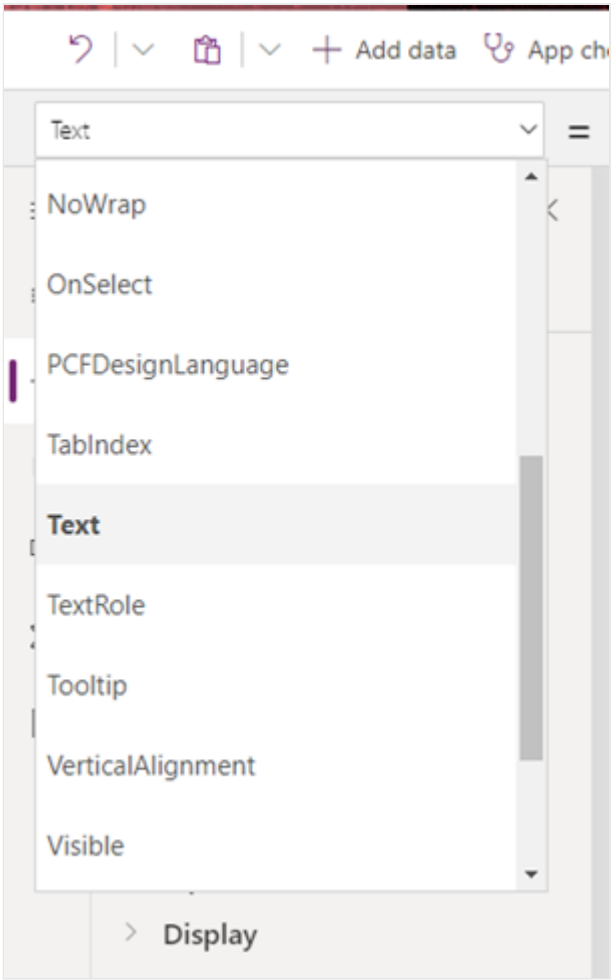
Manage localized web resources

After a web resource language has been added to your page, you can manage the file by selecting the ... ellipses next to the file. A flyout opens where you can refresh the web resource if it has been updated. You can also remove the resource from your page. You'll need to refresh the RESX resources on your page if the display name has changed or any of the string keys have been updated in the RESX file that's been published to your solution. Note that updating the translations in the RESX file and publishing the solution doesn't require the RESX on your page to be refreshed.

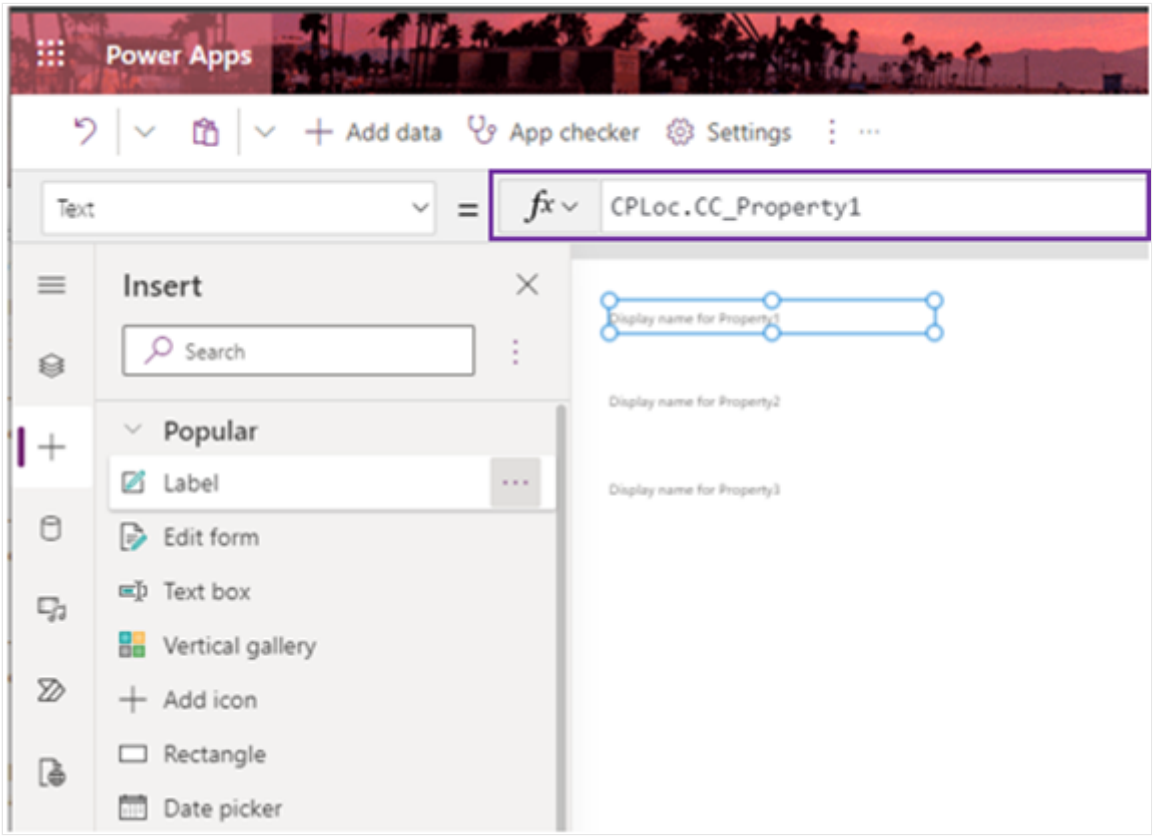


Add the localized string to your label controls

To add localized content to your label control, select the label and set the function to the resource using the Power Fx text function of the control. To do this, select the label control, and then select the **Text** function.



Using the text function, set the value to the web resource and the value you want that is in your web resource RESX file. After setting the expression, the content will appear in the preview for the page.

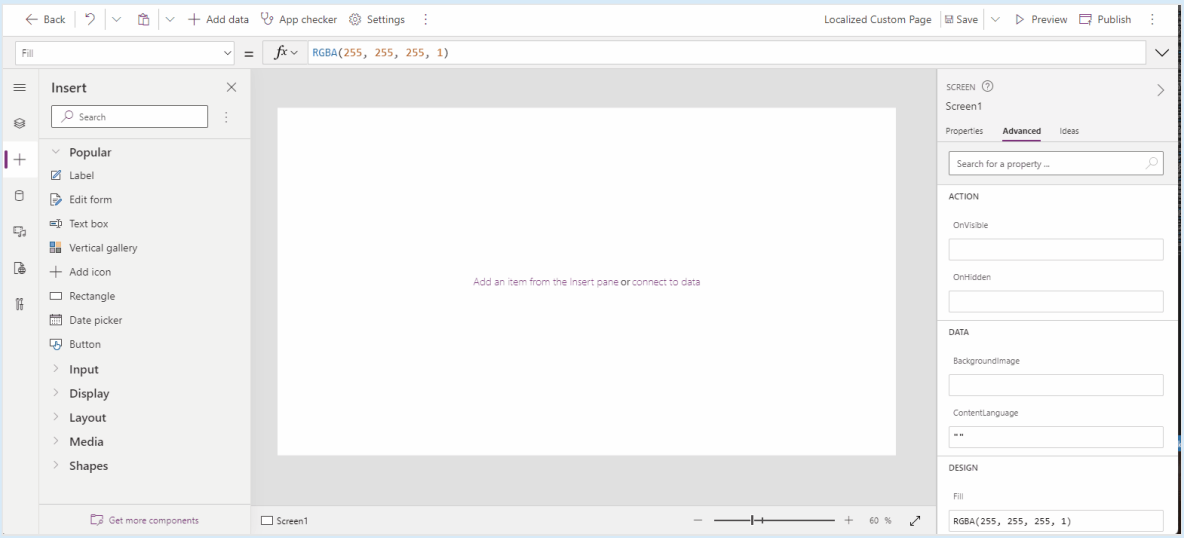


Save and publish your changes to the page.

> **Note**
>
> Make sure you also save and publish your changes to the app. Saving and publishing both the page and app are needed to ensure your localized labels render with the appropriate strings at runtime.

> **Important**
>
> When you add form controls, make sure you localize the labels. They do not automatically pull localized labels when you bind the control to a Microsoft Dataverse table. Currently, you'll need to add strings for the labels to your resx files, unlock the form control, and then set the `displayName` property to your localized

string using the same expression for your custom labels, text, tooltips, etc.



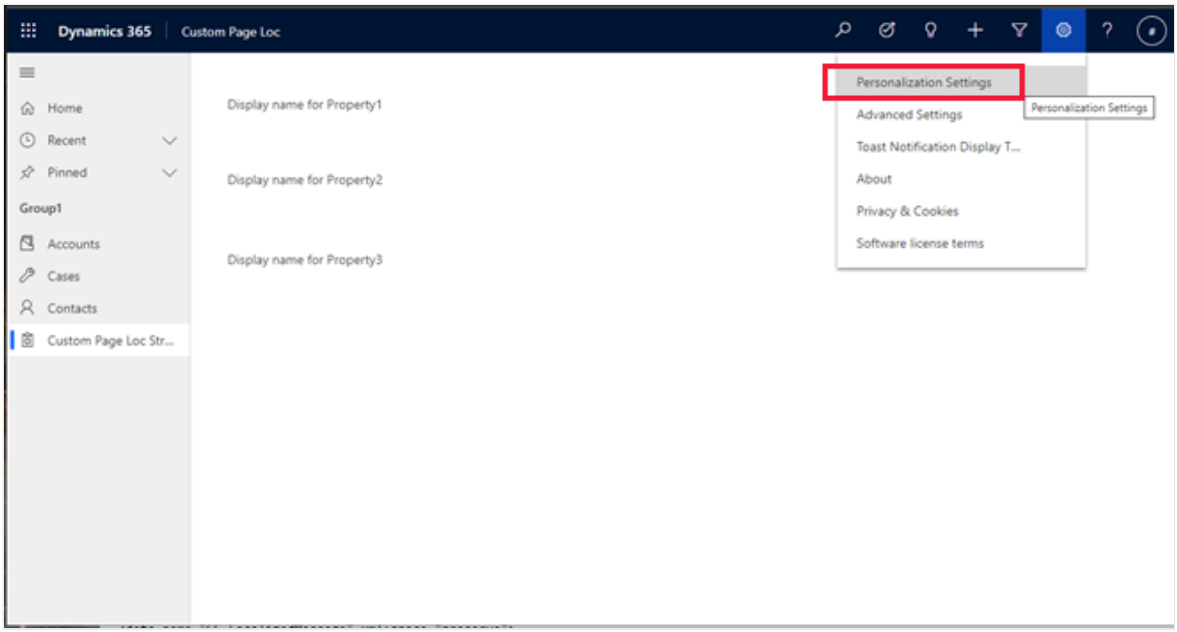
Run your app with the localized content

1. After publishing your page and your app, select **Play** in the app designer to open the app in a separate browser tab.

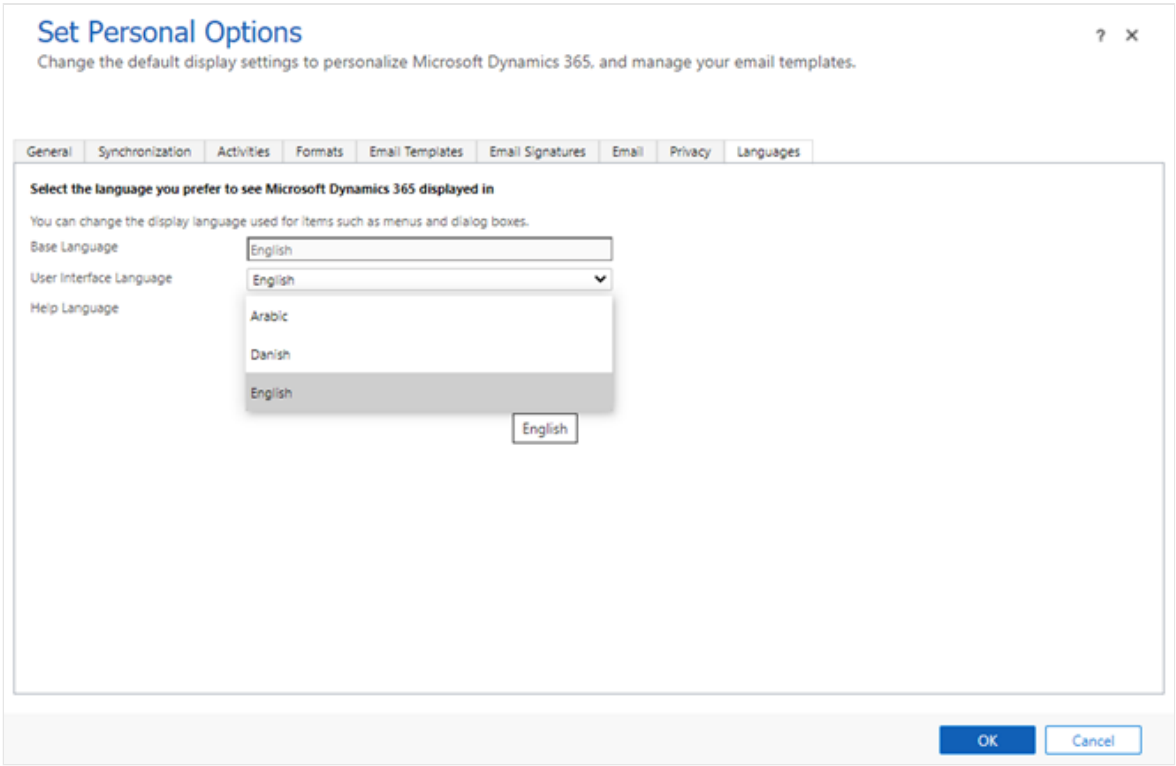
2. Select your custom page on the site map.



3. The labels for your controls display the localized content. Test each language added to your app and pages by selecting **Personalization Settings** under **Settings** (gear icon) at the top of the app.



4. Then, on the **Languages** tab, set the **User Interface Language** to the language you would like to test.



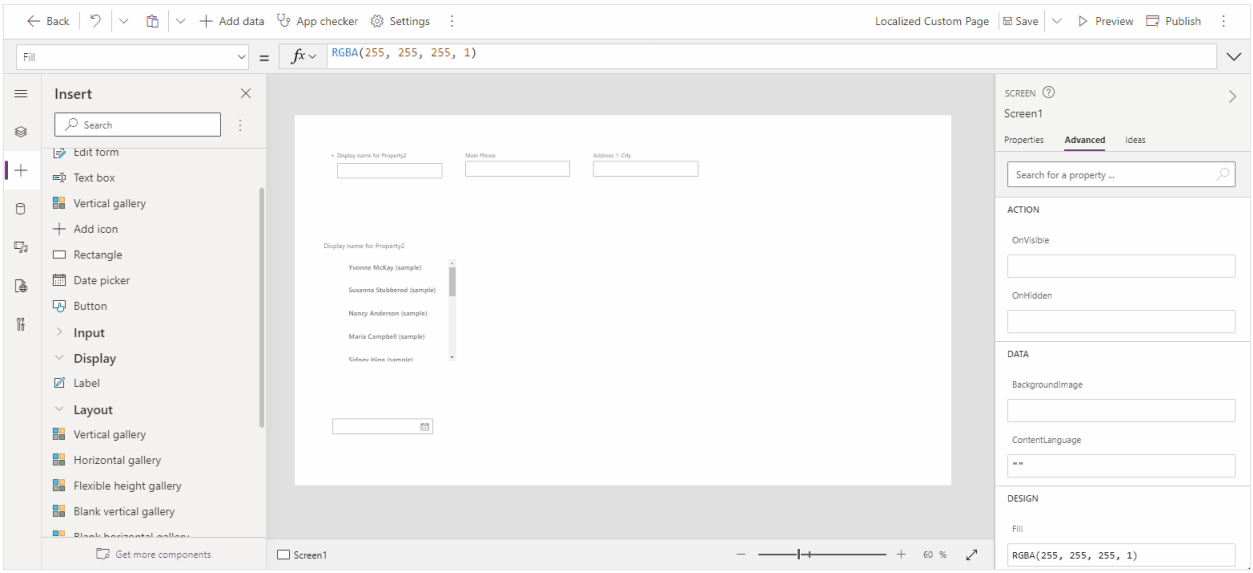
Right-to-left support for custom pages

With localized content, there are several languages that require a right-to-left orientation for content rendered on your page. Right-To-Left (RTL) is fully supported for languages like Arabic and Hebrew and the page will change at runtime automatically for these languages.

> ⓘ **Important**

Right-to-left orientation for icons, shapes, and images aren't currently supported in custom pages. If you add an icon, shape, or image it may not behave as expected, so please be careful when using these types of controls on your custom page.

To review RTL when building a custom page, you will need to go to the page settings, select upcoming features, search for RTL, and enable the setting. Setting this option shows an RTL or LTR button in the preview window where you can switch between orientation without having to run your app. If you would like to see your custom page running RTL, you can add &`flags=isRTL=true` to the end of the URL in your browser when running your app to test RTL for your page.



Known limitation for exporting solutions with resx files

Resx files don't automatically get added when you export your solution and import it into a new environment. This is a known limitation and will require you to upload your resx files into your new environment. So, if you have a developer environment, and you export your solution and import into a production environment, you'll need to add your resx localization files into your production environment solution.

See also

Create or edit model-driven app web resources to extend an app

Overview of custom pages for model-driven apps

Known issues with custom pages in a model-driven app

Article • 04/16/2025

The custom page is a new page type within model-driven apps. Custom pages bring the power of canvas apps into model-driven apps. Below are the known issues to be aware of.

> **Important**
>
> Using custom pages with mobile devices is currently in public preview. Offline and device capability controls like barcode scanning, capturing photos from device, or attaching files isn't supported.
>
> Custom pages aren't supported within **App for Outlook** and return an error on loading.

Maker-related issues

- When a custom page is modified, such as saved and published, the model-driven app isn't aware of the change. The model-driven app continues to use the last version of the custom page when the model-driven app was published. A model-driven app published through app designer, solution explorer, or **Publish all** updates all custom pages in the model-driven app.
- Images, icons, and shapes aren't currently supported with Right-To-Left (RTL) languages.
- The App.ConfirmExit property isn't supported in custom pages.
- The ability to get the current data formats from users settings including date, time, numbers, and currency isn't supported.
- Custom pages use a canvas app hosting session that can time out after 8 hours. However, the Unified Interface session has a longer timeout. When the timeout happens, an error message bar appears that prompts the user to refresh the browser.



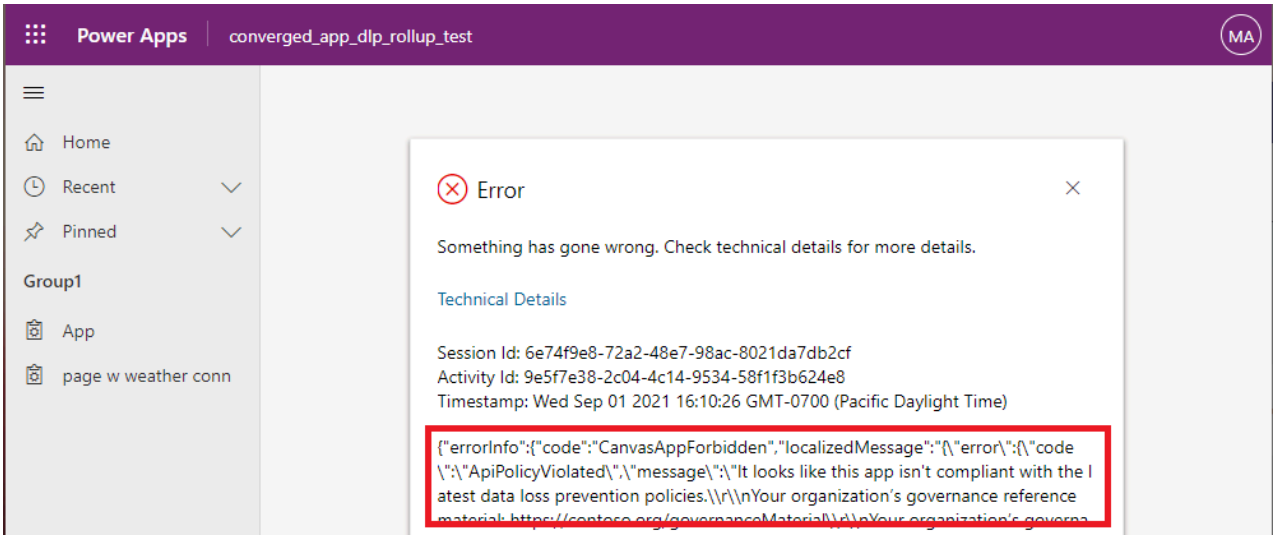
- When a custom page with code component is opened for editing, a security dialog is shown. Selecting **Go back** on the security dialog doesn't navigate back to the parent context. The user can close the browser tab to leave Power Apps Studio.

- Not all canvas app controls are available with custom pages. However, custom pages support the most common canvas app controls and custom pro-dev components. For more information about what is available, see Design a custom page for your model-driven app
- Makers need to share custom pages to allow another maker to make changes, which is a different behavior than the typical model-driven app components. If a custom page can't be shared from the **Solutions** area, open the environment in Power Platform admin center, and then open **Resources** > **Power Apps** > **Page** > **Share**. Similarly, to reuse the canvas app components inside the custom page, the corresponding canvas app component library also needs to be shared with the custom page makers.
- The maker experience for the custom page doesn't have support for certain Power Apps component framework APIs like `Navigation` and Web APIs, which is in line with the stand-alone canvas apps. However, these APIs are available in the published app where the custom page is added to model-driven apps. More information: Add code components to a custom page for your model-driven app.
- The maker experience for the custom page is currently not enabled in sovereign clouds. If you would like to enable it for a maker session, append "powerappsPortalApps.enableEditInShellAppDesigner=true" as a query parameter to the https://make.powerapps.com/ url.
- Makers can't use cross-environment Dataverse references in a custom page.

User-related issues

- When a user with no Power Apps user privileges opens a custom page in the model-driven app, they see an error mentioning no active entitlements to use Power Apps. More information: Licensing overview for Microsoft Power Platform and the associated licensing guide.
- Custom pages require third-party cookies to be enabled, which is required by the canvas app runtime.
- When a user is prompted for consent with connectors and selects **Don't allow**, the custom page renders but without data. The user doesn't get notified that data retrieval is skipped.
- After a model-driven app or custom page is changed and published, loading a custom page can take longer than normal, and no page loading spinner is shown.

- Native player support is available for iOS, Android, and Windows in online-only mode. Offline support currently isn't supported.
- When you go back to a custom page from another page, the page state isn't restored so the page appears like a new navigation. State is also not persisted when switching between multi-session tabs in multi-session apps. For more information about multi-session support with model-driven apps, go to Customer Service workspace sessions and tabs.
- While attempting to sign in, the current behavior caused by a user selecting anywhere away from the sign in box causes the pop out window for sign in to shift behind the app browser.
- When a user runs an app that isn't compliant with their organization's Data Loss Prevention (DLP) policies, they see an error dialog and the 'Technical details' reflects the app isn't DLP compliant.



- When there are multiple custom pages in an app, the consent dialog asks for data permissions for all of the connectors in all the custom pages even if they haven't yet been opened.

Connections

- All custom pages share the connection infrastructure, as if they were part of the same canvas app. For instance, if a model-driven app has two custom pages and one is connected to SQL Server while the other isn't, the consent dialog will still appear and require permission for SQL access even if the page being opened doesn't require it.
- Custom pages in model-driven apps don't support connection detection through the *Connected.connected* property.

- The number of connectors in a model-driven app, across all custom pages, shouldn't exceed 10. The number of connection references, across all custom pages, shouldn't exceed 20. The number of connectors and connection references can increase the wait time an app user experiences when the user plays an app with custom pages.
- The **App Permissions** dialog is not opened when a custom page is first opened by a user in the Power Apps mobile app. This behavior can cause the page to appear frozen, but is expected behavior. To resolve this issue, open the app in a browser to approve the permissions before playing it in the Power Apps mobile app. You can also run the AdminPowerAppApisToBypassConsent PowerShell command to bypass the need for the permissions dialog by a Power Platform administrator.

See also

Model-driven app custom page overview

Use live monitor to troubleshoot page behavior in model-driven apps

07/19/2025

Live monitor can help you debug and diagnose problems, which help you build faster, more reliable model-driven apps. Live monitor provides a deep view into how an app runs by providing a log of all activities in your app as it runs.

> **Important**
>
> Live monitor must be run on the model-driven app and not directly on a custom page.

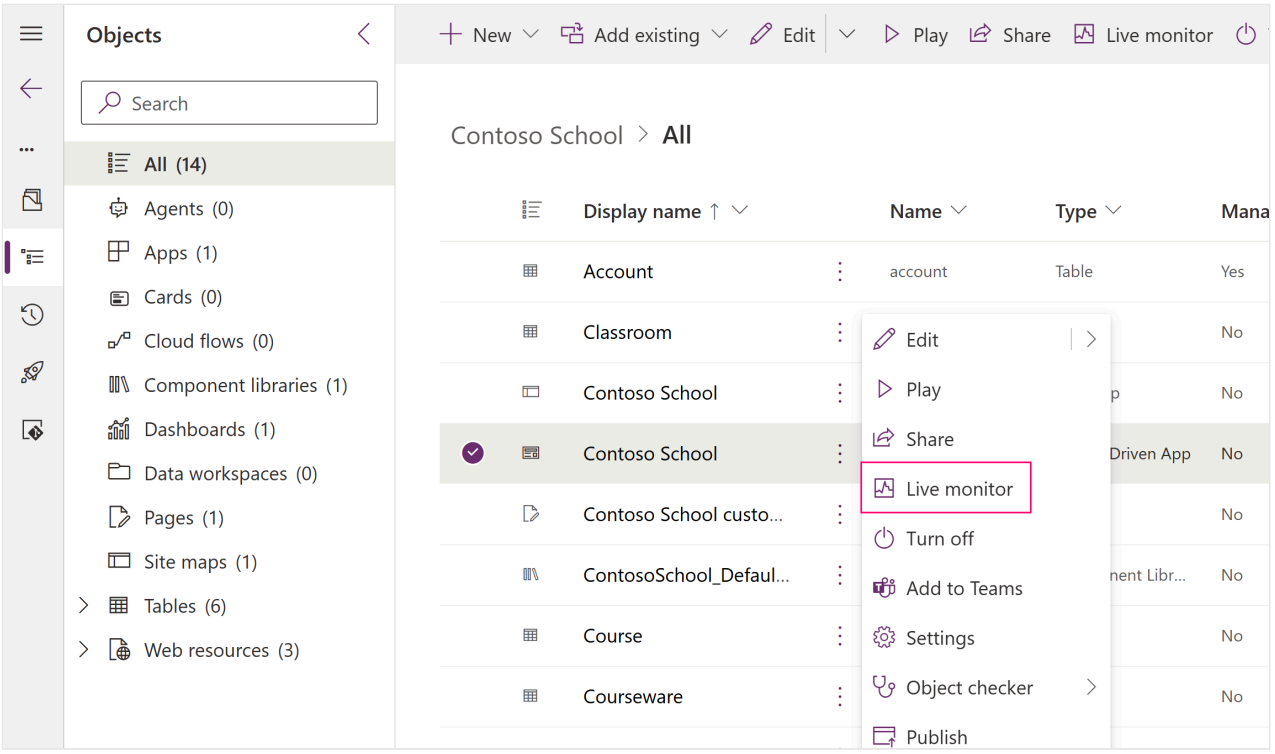
Start a live monitor session

There are two ways to open a live monitor session.

- From Power Apps
- From a model-driven app

From Power Apps

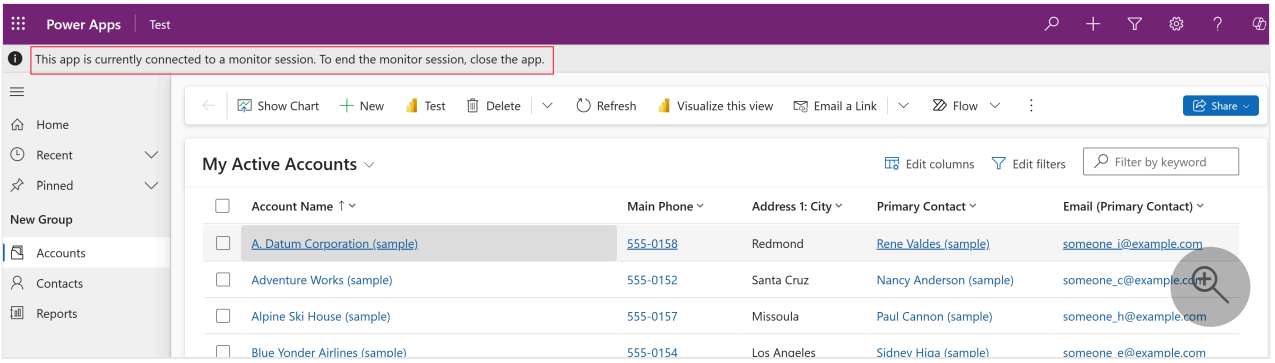
1. Sign in to Power Apps, select **Apps**.
2. Select **...** next to the model-driven app or on the global command bar, and then select **Live monitor**.



3. On the **Live monitor** browser tab that opens, select **Play model-driven app** on the command bar.

4. When prompted, select **Join**.

5. The app loads in a new browser tab with a message indicating the monitoring session. Switch back to the live monitor tab to view activity.

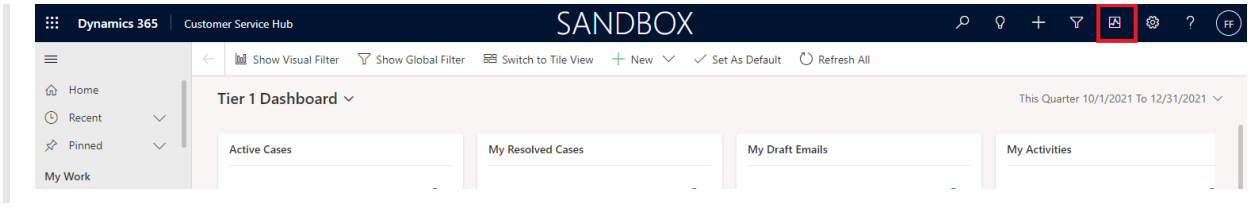


From a model-driven app

1. Play your model-driven app.

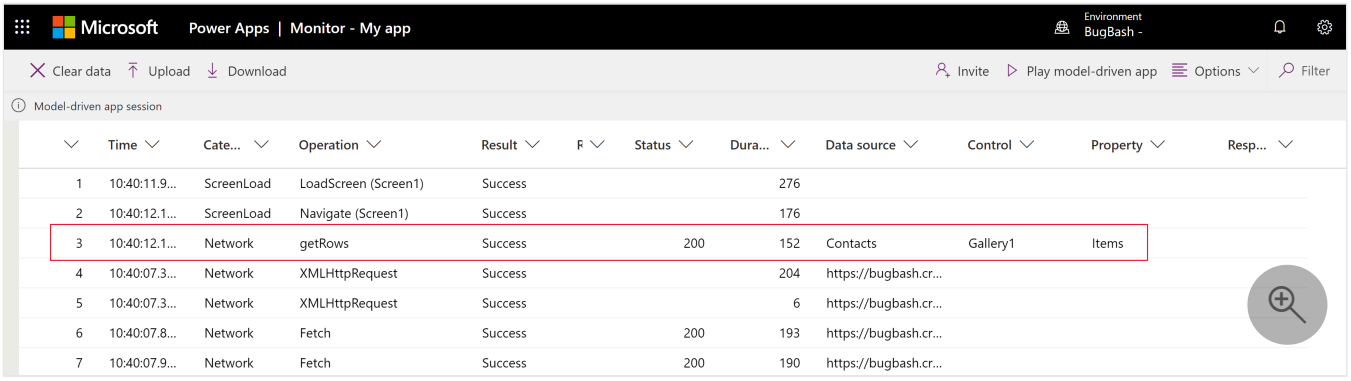
2. Add `&monitor=true` to the end of the URL in your web browser, and then refresh the page.

3. Select **Live monitor** on the command bar.



Filter monitored activity

When you filter on a model-driven app custom page-related events in live monitor, you can get information about related tables, tables, controls, and components, such as on a custom page, in live monitor as your app runs.



	Time	Cate...	Operation	Result	R	Status	Dura...	Data source	Control	Property	Resp...
1	10:40:11.9...	ScreenLoad	LoadScreen (Screen1)	Success			276				
2	10:40:12.1...	ScreenLoad	Navigate (Screen1)	Success			176				
3	10:40:12.1...	Network	getRows	Success		200	152	Contacts	Gallery1	Items	
4	10:40:07.3...	Network	XMLHttpRequest	Success			204	https://bugbash.cr...			
5	10:40:07.3...	Network	XMLHttpRequest	Success			6	https://bugbash.cr...			
6	10:40:07.8...	Network	Fetch	Success		200	193	https://bugbash.cr...			
7	10:40:07.9...	Network	Fetch	Success		200	190	https://bugbash.cr...			

For example, to filter on the custom page that uses the `Gallery1` control, select the **Control** column header > **Filter by**, enter *Gallery1*, and then select **Apply**.

For more information about model-driven app monitoring, go to Debug a model-driven app with live monitor

Close a live monitor session

To close the monitoring session, close the browser tab where the monitored model-driven app is playing.

Next steps

Use live monitor to troubleshoot model-driven app form behavior

Generate a page using natural language

Use natural language to build a *generative page* that's created using AI. Generative pages are an AI-driven experience designed to simplify, accelerate, and improve your app design process in model-driven apps. By interacting with the app agent, you create fully functional pages in your model-driven apps by describing what you need in natural language and specifying which Microsoft Dataverse tables to reference. You can even attach an image of what you'd like the page to look like.

After you describe the page, the system then processes your requirements and specifications and intelligently generates React code that covers both the front-end user experience by selecting the right components and determining the best layout, and the corresponding business logic. Through an interactive, conversational experience, you can refine the page design in real time, adjusting elements, layout, and functionality to perfectly match your vision.

Development approaches

You can create generative pages using two approaches:

- Power Apps (make.powerapps.com): As described in this article, makers can create and iterate on generative pages directly in the browser by using a conversational UI experience. Available in the United States, Great Britain, Australia, and Singapore.
- AI code generation tools: Developers who prefer working with local development tools and direct access to TypeScript and React code can use external tools like Claude Code to develop generative pages locally with a code-first approach and CLI-based deployment. Available worldwide on public clouds. More information: Create and edit generative pages with AI code generation tools

Prerequisites

- The Power Platform environment must be located in one of the following regions: United States, Great Britain, Australia, or Singapore.

Create a generative page in model-driven apps

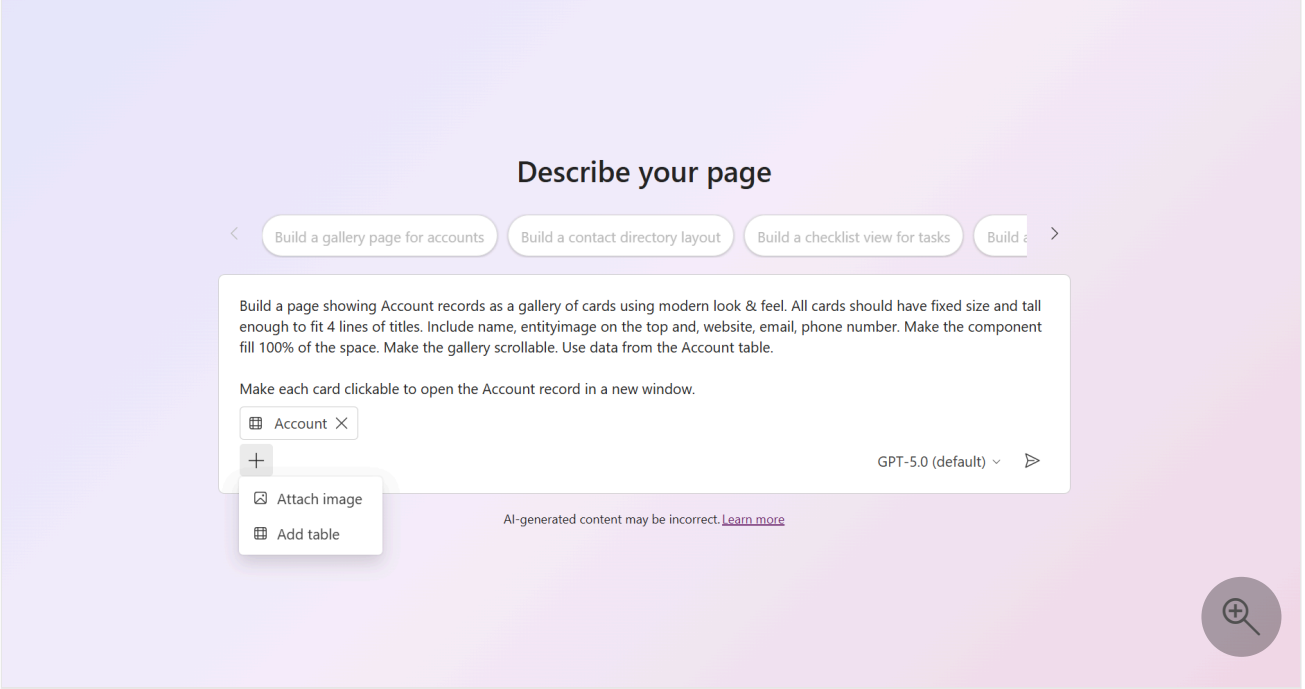
1. Sign in to Power Apps.
2. Open a model-driven app for editing.

3. In the app designer, select **Add a page** > **Describe a page**.

   A full-page generative page experience opens.

4. In the text box, type a description of the type of page you want to create. Include functional requirements and optionally any UX specifications. For example, you could enter *Build a page showing Account records as a gallery of cards using a modern look and feel. Include name, entityimage on the top, and website, email, phone number. Make the gallery scrollable by using data from the Account table*. If you want the page to support multiple languages, follow the guidance in Localize a generative page.

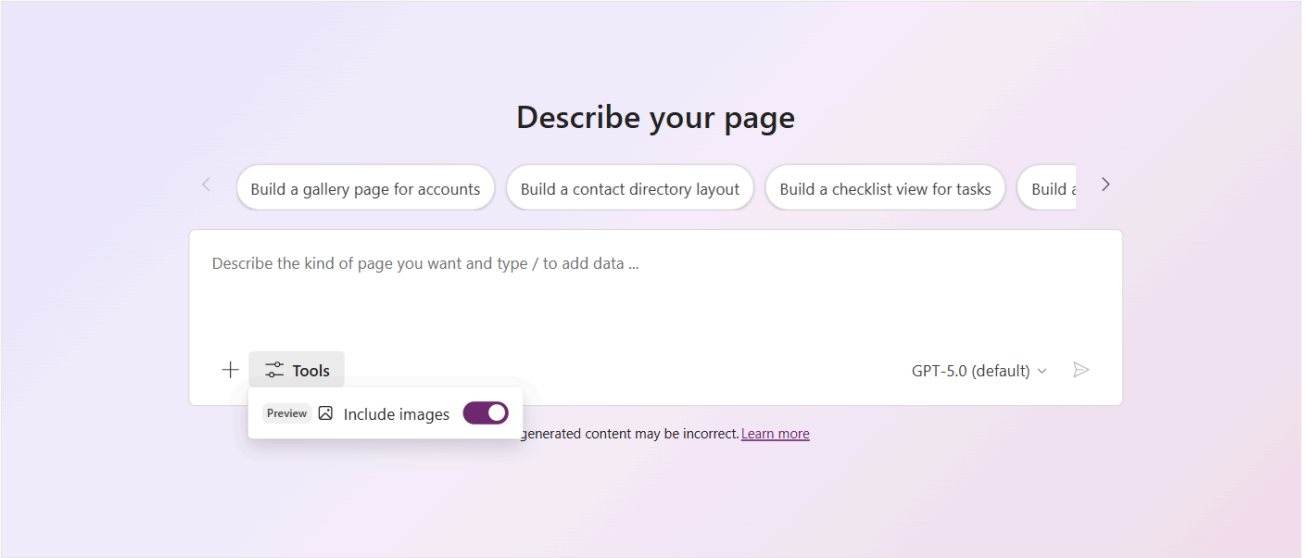
5. Add tables and images as appropriate by selecting **Add data** > **Add table**. You can link up to six Dataverse tables. In the screenshot, the account table is added.



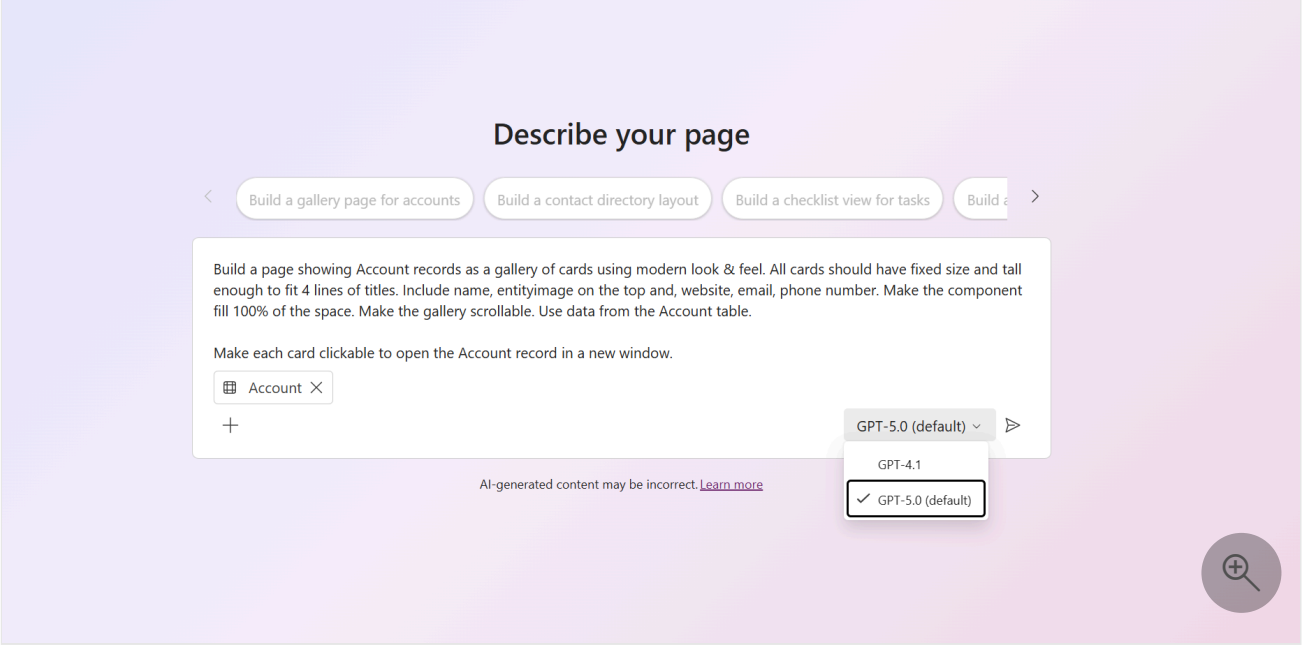
6. Optionally upload one or more images to guide the UI of your generated page by selecting **Add data** > **Attach image**. This can be a rough napkin sketch or a higher resolution image. The image can represent the structure or layout of the entire page you want to create or it can be visuals for a certain section or component you want to mimic.

> **Note**
>
> If you want the generated page to display a particular image, instead of attaching the image to the prompt, you can (1) include the image in a referenced Datatable table and have the agent retrieve the image from there, (2) host the image publicly and give the agent the URL, or (3) add the image as a web resource and give the agent the URL for the web resource.

7. Optionally enable or disable the **Include images** (preview) tool to allow the agent to utilize images from a curated, safe library of 25,000 stock images to support use cases like default/placeholder images, decorative backgrounds, static content blocks, and richer empty or error states.



8. Optionally choose the AI model you want to use to generate the page.



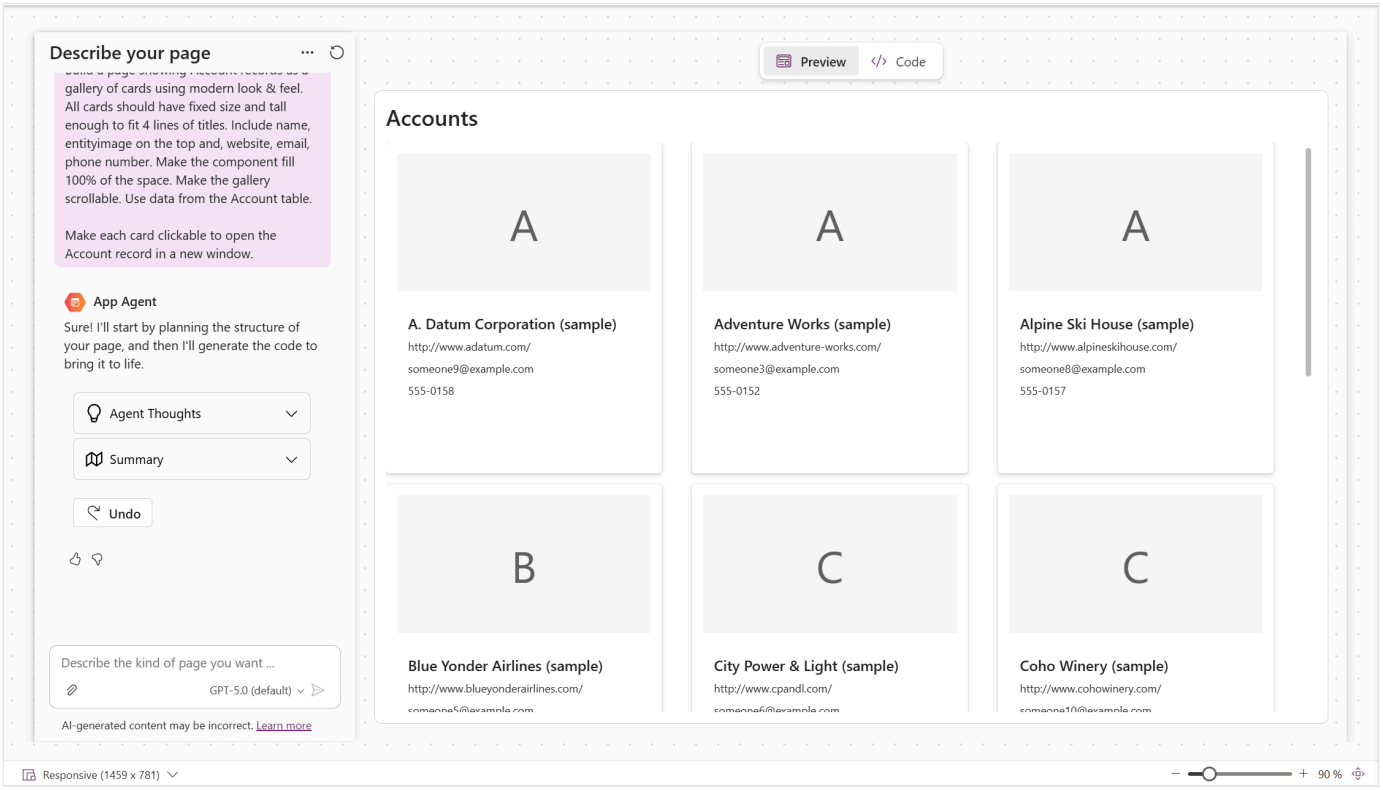
> **Note**
>
> GPT-4.1 is currently the only available model for generative pages.

9. When you're finished describing the page, select **Generate page**.

The agent begins a multistep build process that you can observe in real time:

- **Thought streaming**: The agent first outlines its interpretation of your prompt, listing requirements, assumptions, and an execution plan.
- **Code generation**: Next, it writes the underlying code for your page based on the plan.
- **Transpilation**: The generated code is then transpiled to ensure compatibility and proper rendering.
- **Final rendering**: Finally, the completed user experience is displayed.

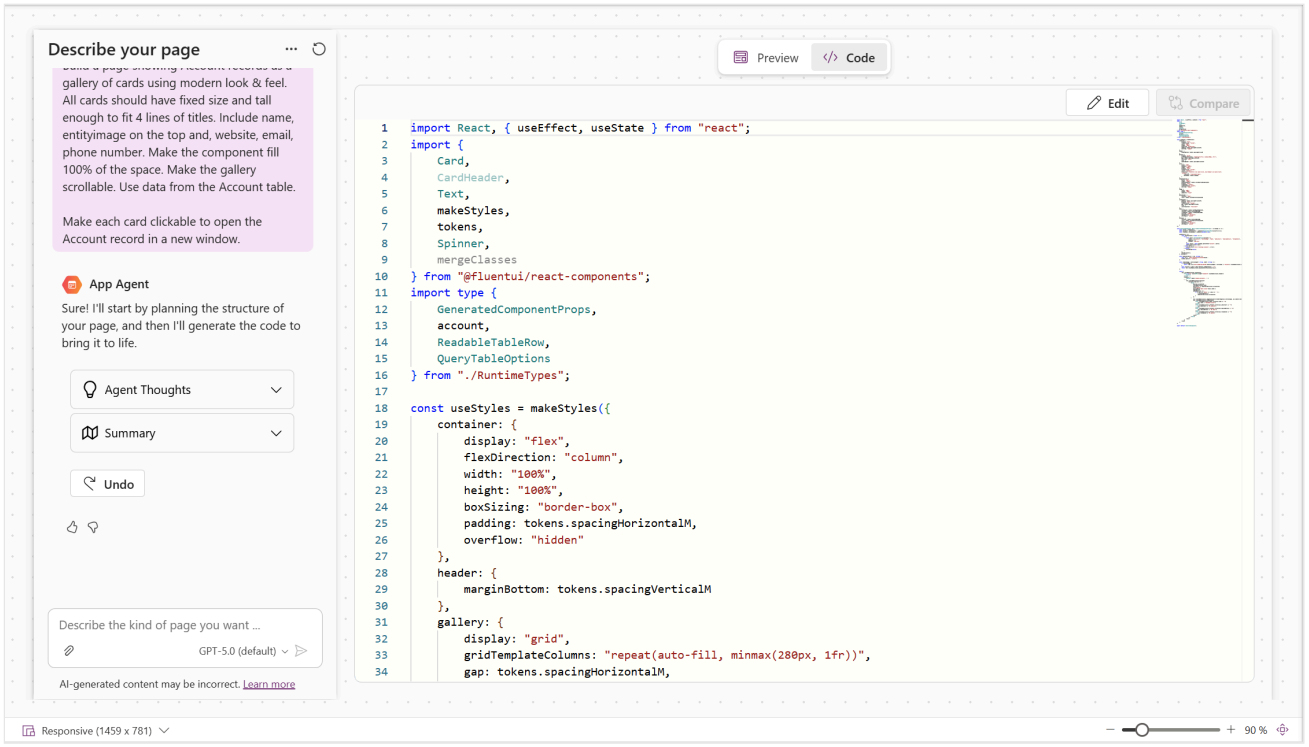
If the user experience isn't displayed at the end of this process, you can view it by selecting the **Preview** tab.



View the generated code, iterate, and publish

After generating your page, you have several options to refine and finalize it:

1. **View and edit the generated code** Select the **Code** tab to view the code generated by the app agent.



You can refine the output in two ways:

- Iteratively chat with the app agent to fix errors, adjust layout, or add functionality.
- Manually edit the code by selecting **Edit** on the Code tab. After making some edits, you can select **Save** to commit your changes as a new iteration, or **Cancel** to discard your changes.

Learn about the dataApi object methods used for data operations

2. **Compare iterations** After completing two or more iterations with the agent, you can select **Compare** on the Code tab to view a code diff between the current and previous iteration.

> **Note**
>
> This capability is currently only available starting with the second iteration in the current session.

3. **Attach a screenshot for quick reference** In the chat experience, select **Attach** > **Add screenshot** to include a screenshot of the current preview with your next interaction with the app agent. A screenshot is useful for helping to adjust the visuals of the page. You can alternatively attach other images to help refine the visuals of the entire page or a particular piece of the page.

4. **Check for accessibility issues (new)** After each iteration of code generation, the **Accessibility assistant** at the bottom of the screen scans the generated code assessing it

with respect to accessibility. You can see a high level status update directly on screen and can choose to open the detailed results in a panel. Select **Auto fix** to pass any violations directly to the agent for it to try to automatically resolve.

5. **Save and publish** In the command bar, select **Save** to avoid losing progress on your page. When you're satisfied with your page, select **Save and Publish** to publish all pending changes for the app, including any generative pages.

> **ⓘ Important**
>
> While the agent makes a best-effort attempt to generate complete, production-ready code, including considerations for accessibility and security best practices, you're ultimately responsible for validating the code. Ensure the generated code meets your organization's standards and compliance requirements.

Common tasks with generative pages

This section covers common scenarios and tasks when working with generative pages in your model-driven apps.

Set up a page to accept input parameters

Generative pages can accept the input parameters `recordId`, `entityName`, and `data` enabling them to receive contextual data when navigated to. To configure a page to accept parameters, describe what the page should accept in your prompt and the agent wires up the initialization code automatically. For example:

```
Set up the page to accept an account recordId. When the page loads, use these to fetch and display the corresponding account details.
```

```
Configure this page to accept a data parameter containing a custom filter object. Use it to filter the displayed records when the page loads.
```

Navigate to a generative page

You can navigate to a generative page programmatically using `Xrm.Navigation.navigateTo`, passing input parameters if the target page is set up to receive them. For examples and full API details, see Navigate to and from a generative page using Client API.

Use specific images in a page

There are three approaches to using specific images in your generative pages.

Store images in a Dataverse table

Include the image in a Dataverse table that's referenced by your generative page, and instruct the agent to retrieve the image from there. When creating your page, tell the agent how to get the right image from your table.

```
Build a product catalog page showing products from the Product table. Display the product image from the ProductImage column, along with the name, description, and price.
```

Host images externally and reference by URL

Host your images on an external server or content delivery network (CDN) and provide the agent with the URL to the image:

```
Display the company logo using this URL: https://example.com/images/logo.png at the top of the page.
```

Add images as web resources

Upload the image as a web resource in your solution and reference it by its web resource URL:

1. Add the image as a web resource in your solution.
2. Note the web resource name (for example, `new_/images/banner.png`).
3. Reference it in your prompt or code using the web resource URL format described here:

```
Use the banner image from web resource new_/images/banner.png as the page header background.
```

Adding generative pages to solutions

Generative pages are solution-aware and can be added to a solution (via an app) to facilitate moving between environments. To add a generative page to a solution, do the following:

> **Important**
>
> If your generative page was created during the preview phase, you must load the generative page in the model app designer to initiate a one-time migration to the new solution-aware data model. When the page loads in the designer, you observe an "Upgrading your page" progress message. Don't close the window until migration completes.

1. Add the app containing your generative pages to a solution

    - In Power Apps, select **Solutions** on the left navigation pane, and open the desired solution.
    - Select **Add existing** > **App** > **Model-driven app**.
    - Select the app that contains the generative pages.

2. **Export the solution** (as managed or unmanaged)

    - Ensure the app sitemap is included in the solution. If not, it should be requested during dependency checks.
    - Generative pages (shown as **UX Agent Project** rows) are also requested based on their dependency on the sitemap.

> **Note**
>
> Check these items if no generative pages are requested during dependency checks:
>
> - Pages created during preview will only appear if they've been migrated. Load them in the designer to trigger migration.
> - If the sitemap is included in the solution and your generative pages have been migrated, try making a minor change to the sitemap, such as reordering or renaming a page, republishing the app, and then trying the export again.

After export, you can import the app and generative pages into another environment. When opened in the designer in the target environment, only the first prompt and published code are available. The full agent conversation doesn't transfer with the page.

Localize a generative page

To localize a generative page, ensure your prompt covers the follow items:

- Specify your languages. Tell the agent which languages you want the page to support, ideally language name and LCID.
- Request translated text. Ask the agent to create a translation dictionary for all user-visible labels and text in the page for each language.
- Enable RTL support. If your environment includes an RTL language like Arabic or Hebrew, ask the agent to implement the appropriate right-to-left layout support.
- Use regional formatting. Ask the agent to format dates, numbers, and currency using each user's preferences from their Dataverse user settings, rather than using hardcoded formats or locale codes.

For detailed implementation guidance and code patterns, go to the localization instructions in the Power Platform Skills repository.

> **Note**
>
> The sitemap entry for a generative page isn't localized by default. Sitemap localization needs to be handled separately in the app designer.

Limitations

These are the current limitations of generative pages:

- Your page can connect to only Dataverse tables (up to 6 for a single page). No other data sources can be used.
- Your prompt can have a maximum of 50,000 characters.
- When you create generative pages through in Power Apps (make.powerapps.com), only US English is a supported prompting language.
- Collaboration isn't supported. Ensure only one maker is working on a generative page at a time to avoid unintended conflicts.
- Only these data types are supported:
  - Choice
  - Currency
  - Customer
  - Date and Time

- Date Only
- Decimal Number
- Floating Point Number
- Image
- Lookup
- Multiline Text
- Status
- Status Reason
- Text
- Whole Number
- Yes/No
- Unique Identifier

Frequently asked questions

Can I start designing my app in Plan Designer and then use generative pages?

Yes, while there's currently no direct integration between plans and generative pages, you can use plans to define your tables and apps, and then switch to the model-driven app designer to create any desired generative pages.

Are there any additional fees or credits needed to use generative pages?

No, the maker experience for generative pages (and plans) don't require any additional AI or message credits.

What kind of images work best for guiding the UI generation?

It depends on your goal. If you know exactly how you want the final page to look, clear and detailed sketches or wireframes (whether hand-drawn or digital) are most effective. If you only have a general concept and want the agent to fill in the details, quick whiteboard sketches or even napkin drawings can work well.

Can I use generative pages with canvas apps or other app types?

No, generative pages are currently only supported in model-driven apps.

Is there a way to reuse or clone a generative page across environments?

Yes, generative pages are solution-aware elements that can be moved across environments. Only the last published version of the code and the first prompt are maintained with the solution; the full revision history remains on the original environment.

Can I edit the generated code manually?

Yes. You can manually edit the generated code by selecting **Edit** on the **Code** tab. Your changes are saved as a new iteration. Select **Cancel** to cancel your edits if you choose not to keep them.

Is the code generated by generative pages guaranteed to be production-ready and compliant with my organization's standards?

No. While the agent makes a best-effort attempt to produce high-quality, production-ready code with accessibility and security best practices in mind, it's your responsibility to validate the output. Always review the generated code to ensure it aligns with your organization's standards, policies, and compliance requirements.

How do I give feedback on the feature?

The best way to give feedback on generative pages is to use the thumbs up/down buttons in the chat experience that are available after each iteration. We also strongly encourage you to select **Yes** on **Share relevant content samples and additional log files** so that we can debug or attempt to recreate any issues you encountered with the feature.

What if I don't see generative pages enabled in my environment?

Confirm that your environment is in one of the supported regions for the Power Apps maker studio experience: United States, Great Britain, Australia, or Singapore. If your region isn't listed, you can still create and edit generative pages using AI code generation tools, which are available worldwide on public clouds. More information: Create and edit generative pages with AI code generation tools

Related articles

- Create and edit generative pages with AI code generation tools
- FAQ about generative pages in model-driven apps
- Legal terms

Last updated on 04/20/2026

Create and edit generative pages with AI code generation tools

This article describes how to use AI code generation tools, such as GitHub Copilot CLI and Claude Code, to create and edit generative pages for model-driven apps in Power Apps. This approach allows you to integrate advanced code generation capabilities directly into your development workflow, allowing you to create new generative pages or iterate on existing ones using natural language instructions.

Using AI code generation tools with generative pages provides an alternative development approach that complements the UI-based experience in Power Apps (make.powerapps.com). This code-first approach is designed for developers and technical makers who prefer working with local development tools and CLI-based workflows.

What you can do with code generation tools

- *Create new generative pages* using plain-language requirements
- *Update existing generative pages* by requesting changes or enhancements through your AI tool
- *Deploy directly* to your Power Apps environment using PAC CLI commands
- *Work locally* with your preferred IDE and development tools

How it works

1. You describe what you want to build in natural language, for example, "Create a generative page dashboard showing top accounts by revenue."
2. The AI code generation tool uses installed generative page skills or other context about generative pages to ask clarifying questions about your requirements.
3. The tool generates production-ready TypeScript and React code for your generative page.
4. The tool deploys the code to your environment using generative page PAC CLI commands.
5. Your generative page appears in your model-driven app.

Prerequisites

Before you start, ensure you have the required software and permissions described here.

Software requirements

Expand table

Component	Minimum version	More information
Node.js	18.0 or later	Download Node.js
Power Platform CLI (PAC CLI)	Latest	Install PAC CLI
GitHub Copilot CLI, Claude Code, or other code generation tool	Latest	GitHub Copilot CLI or Claude Code

Additional requirements

- A Power Platform environment with a model-driven app to deploy pages.
- An *authenticated PAC CLI session* connected to your target environment.
  - Go to Authenticate Power Platform CLI for more details on getting connected.

> **Note**
>
> This capability is available worldwide in public clouds.

Install the plugin

Run the installer to set up all Power Platform plugins in either PowerShell or a Windows command window.

PowerShell

```
iwr https://raw.githubusercontent.com/microsoft/power-platform-skills/main/scripts/install.js -OutFile install.js; node install.js; del install.js
```

dos

```
curl -fsSL https://raw.githubusercontent.com/microsoft/power-platform-skills/main/scripts/install.js | node
```

The installer automatically:

- Detects available tools (Claude Code, GitHub Copilot CLI)
- Registers the plugin marketplace and installs all plugins

- Enables auto-update so plugins stay current

After installation, restart your AI tool if needed.

Install only the generative page plugin

To install only the generative page plugin for GitHub Copilot CLI or Claude Code:

1. Add the Power Platform Skills marketplace plugin: `/plugin marketplace add microsoft/power-platform-skills`
2. Install the Power Apps plugin: `/plugin install model-apps@power-platform-skills`

> **Note**
>
> For Claude Code, you can install the plugin with different scopes, such as global, local, or user. Depending on the scope, you need to be in the correct directory for Claude Code to use the plugin. Go to Extend Claude with skills

Once installed, you can use the plugin by either:

- Running the `/genpage` command explicitly.
- Describing the page you want to create. The tool automatically detects and uses the plugin.

> **Tip**
>
> Turn on auto-update to automatically receive updates to the marketplace and skills. Use the `/plugin` command, navigate to **Marketplaces**, choose the marketplace, and turn on auto-update.

Using other AI code generation tools

For other AI code generation tools, ensure your tool has access to the generative page resources from the Power Platform skills GitHub repository. The model-apps plugin folder includes component documentation, sample code, PAC CLI command reference, and workflow instructions necessary to create code adhering to generative page requirements, including how to create pages that support multiple languages and regions. Consult the repository readme for information about accessing and using these resources with your preferred tool.

Skills overview

The Power Apps plugin provides this skill for working with generative pages.

Expand table

Skill	Command	Description
Generative pages	`/genpage`	Create code for generative pages (for creation or editing scenarios)

This skill enables you to describe what you want to build and have the AI tool generate complete TypeScript and React code for your generative page, then deploy it directly to your Power Apps environment.

Create a new generative page

Follow this workflow when building a new page from scratch.

1. Start a conversation with your AI tool. Describe what you want to create, including what data you want to include (which Dataverse tables or whether to create sample mock data). Be as specific as you want—the more vague you are with the request, the more details the agent tries to fill in itself. You can also attach or provide an image or other materials to help guide the visuals, theming, and layout. For example:

    - "Create a generative page dashboard showing our top 10 accounts by revenue using the Account table"
    - "Build a generative page form for creating and editing contact records with sample data"
    - "Make a generative page for displaying incident reports on a map using the Incident table"
    - "Create a generative page for a sales pipeline visualization with opportunities using the modern blue theme"

2. Answer clarifying questions. The AI tool asks questions to understand your requirements. Be specific about business needs and data requirements, identify mobile requirements early, and mention any specific UI components or layout preferences.

3. Review the implementation plan. The AI tool presents a plan describing the components to be built, Dataverse tables and columns to be used, key features and interactions, and data retrieval approach. Confirm the plan meets your requirements or request changes.

4. Review code and deploy. The AI tool generates complete TypeScript code. Ask the tool to publish or deploy your page when you're ready, optionally specifying a sitemap name (the tool generates a meaningful name by default).

5. Test and iterate. Open your model-driven app in Power Apps and navigate to the new page using the sitemap. If you need to make changes, return to your AI tool and describe the updates using natural language.

> **Note**
>
> You can change the name or position of the generative page in the sitemap at any time from the model-driven app designer.

Edit an existing generative page

Use this workflow to update a page that already exists in your environment.

1. Retrieve the existing page. In your AI code generation tool, request to retrieve the existing generative page by providing the page ID (GUID) or the page name in the sitemap and the app it's in. For example, "I want to update the Pet Adoption generative page from the Demo app."

2. Describe your updates. Tell the AI tool what changes you want to make. For example:

   - "Add a filter to show only active records"
   - "Change the layout to display cards in a grid instead of a list"
   - "Add a chart showing adoption trends over time"
   - "Update the form to include the new custom field for pet temperament"

3. Review, publish, test, and iterate. The AI tool generates updated TypeScript code based on your requested changes. Follow the same review, publish, and test process described in the "Create a new generative page" section. Continue iterating with natural language instructions until the page meets your requirements.

Set up a page to accept input parameters

Generative pages can accept the input parameters `recordId`, `entityName`, and `data`, enabling them to receive contextual data when navigated to from other pages or code. When you

instruct the AI tool to configure input parameters, it generates the appropriate initialization code so the page reads and uses those values when it loads.

Describe the parameters you want in your prompt:

- "Set up the page to accept an Account recordId and entityName. When the page loads, use these parameters to fetch and display the corresponding account details."
- "Configure this page to accept a data parameter containing a custom filter object. Use it to filter the displayed records when the page loads."

To navigate to the page and pass these parameters, see Navigate to and from a generative page using Client API.

Localization

When you create a generative page using the Power Apps plugin for Claude Code or GitHub Copilot CLI, localization is handled automatically. The agent detects all languages enabled in your environment and generates code so that the page works with all of those languages. The page respects each user's preferred language and regional formatting preferences for dates, numbers, and currency.

If you want to target a different set of languages than those enabled in your environment, you can ask the agent to adjust, for example:

> "Update this page to support English, French, and Spanish only."

> **Note**
>
> The sitemap entry for a generative page isn't localized by default. To localize sitemap entries, update them separately in the app designer.

For more information, go to the localization instructions in the Power Platform Skills repository.

Troubleshooting

Page fails to load in Power Apps

If you navigate to your generative page and observe an error message or blank screen:

1. Open browser developer tools (F12 in most browsers).
2. Select the **Console** tab.
3. Copy the complete error message, including the stack trace.
4. Return to your AI code generation tool and paste the error with context:

   "I'm getting this error when opening the page: [paste error here]. Please fix the issue."

   The AI tool analyzes the error, identifies the root cause, and generates a fix.
5. Review the fix and ask the tool to republish the page.

Reverting to a working version

If recent changes broke your page or made issues worse, you can ask the AI tool to roll back to a previous working version:

"The recent changes broke the page. Please revert to the last working version."

The AI tool then:

1. Identifies the changes that were made
2. Restores the previous working code
3. Redeploys the stable version

Best practices

- Start simple. Begin with a basic version of your page and iterate to add complexity.
- Test frequently. Deploy and test your page after each significant change.
- Be specific. Provide detailed requirements to get better initial results.
- Use existing patterns. Reference similar pages or UI patterns when describing your requirements.
- Validate generated code. Always review the generated code to ensure it meets your organization's standards and compliance requirements.

> **ⓘ Important**
>
> While AI code generation tools make a best-effort attempt to generate complete, production-ready code with accessibility and security best practices, you're ultimately

responsible for validating the code. Ensure the generated code meets your organization's standards, policies, and compliance requirements.

Limitations

The limitations for generative pages created with AI code generation tools are the same as those for generative pages created in the Power Apps maker portal:

- Your page can connect to only Dataverse tables.
- Collaboration isn't supported—ensure only one maker is working on a generative page at a time.
- Only these data types are supported: Choice, Currency, Customer, Date and Time, Date Only, Decimal Number, Floating Point Number, Image, Lookup, Multiline Text, Status, Status Reason, Text, Whole Number, Yes/No, Unique Identifier.

Related content

- Generate a page using natural language with model-driven apps
- Navigate to and from a generative page using Client API
- dataApi object methods for generative pages
- Power Platform CLI reference

Last updated on 04/10/2026

Navigate to and from a generative page using client API

This article provides examples of navigating to generative pages in model-driven apps by using the client API navigateTo method. Learn how to open generative pages inline or in a dialog, and how to pass input parameters such as a record ID or custom data.

> ⓘ Note
>
> This method is supported only on Unified Interface.

Find the page ID

Each of the following examples requires the ID of the target generative page. To find the page ID:

1. Open the model-driven app containing the generative page in the app designer.
2. Select the generative page in the pages list.
3. In the properties pane, copy the GUID shown in the **Generative page** field.

Open a generative page inline without parameters

Opens a generative page as a full-page inline view with no input parameters.

JavaScript

```
var pageInput = {
    pageType: "generative",
    pageId: "<genPageID>"
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when page opens
        }
    ).catch(
        function (error) {
            // Handle error
```

```
        }
    );
```

Open a generative page inline with record context

Passes a `recordId` and `entityName` to the generative page so the page can load and display a specific record. The target generative page must be set up to accept these parameters.

JavaScript

```
var pageInput = {
    pageType: "generative",
    pageId: "<genPageID>",
    entityName: "account",
    recordId: "00aa00aa-bb11-cc22-dd33-44ee44ee44ee" // replace with actual record
GUID
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when page opens
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open a generative page inline with custom data

Passes a `data` object containing custom key-value pairs to the generative page. The target generative page must be set up to accept these parameters.

JavaScript

```
var pageInput = {
    pageType: "generative",
    pageId: "<genPageID>",
    data: { status: "active", category: "premium" }
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
```

```
        function () {
            // Called when page opens
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open a generative page as a centered dialog

Opens a generative page in a centered dialog, passing both record context and custom data. Adjust `width` and `height` as needed.

JavaScript

```
var pageInput = {
    pageType: "generative",
    pageId: "<genPageID>",
    entityName: "account",
    recordId: "00aa00aa-bb11-cc22-dd33-44ee44ee44ee", // replace with actual record
GUID
    data: { view: "summary" }
};
var navigationOptions = {
    target: 2,
    position: 1,
    width: { value: 70, unit: "%" },
    height: { value: 80, unit: "%" },
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open a generative page as a side dialog

Opens a generative page as a side dialog using `position: 2`.

JavaScript

```
var pageInput = {
    pageType: "generative",
    pageId: "<genPageID>"
};
var navigationOptions = {
    target: 2,
    position: 2,
    width: { value: 500, unit: "px" },
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Unsupported: Open a generative page in a side pane

> **ⓘ Important**
>
> Opening a generative page by using `Xrm.App.sidePanes.createPane()` isn't currently supported.

JavaScript

```
// Not supported — page content doesn't render
const pane = await Xrm.App.sidePanes.createPane({
    title: "My Generative Page",
    paneId: "GenPage",
    canClose: false,
    width: 400
});
pane.navigate({ pageType: "generative", pageId: "<genPageID>" });
```

Use a centered or side dialog via `Xrm.Navigation.navigateTo` instead.

Navigate from within a generative page using client API

When you navigate from inside a generative page component, use `(window as any).Xrm` to access the Xrm object, since the React component scope doesn't provide direct access to it.

Navigate to another generative page with record context and custom data

TypeScript

```
const xrm = (window as any).Xrm;
xrm.Navigation.navigateTo({
    pageType: "generative",
    pageId: targetPageId,
    entityName: "account",
    recordId: selectedRecordId,
    data: { view: "summary" }
});
```

> **Note**
>
> When you navigate within model-driven apps, avoid constructing raw URLs or manipulating `window.location`.

Navigate to a generative page via URL (external callers only)

You can navigate to a generative page by constructing a URL with the following structure:

```
https://<your-org>.crm.dynamics.com/main.aspx?appid={app-id}&pagetype=genux&id={page-id}&recordid={recordId}&entityname={entityName}&data={encoded-json}
```

You must URL-encode the `data` parameter as JSON. For example, to pass a custom filter object:

```
https://<your-org>.crm.dynamics.com/main.aspx?appid={app-id}&pagetype=genux&id={page-id}&data=%7B%22status%22%3A%22active%22%7D
```

You must set up the target generative page to accept these parameters.

Related articles

- Generate a page using natural language
- navigateTo (Client API reference)

Last updated on 04/10/2026

Add or remove forms, views, or charts

Article • 07/11/2024

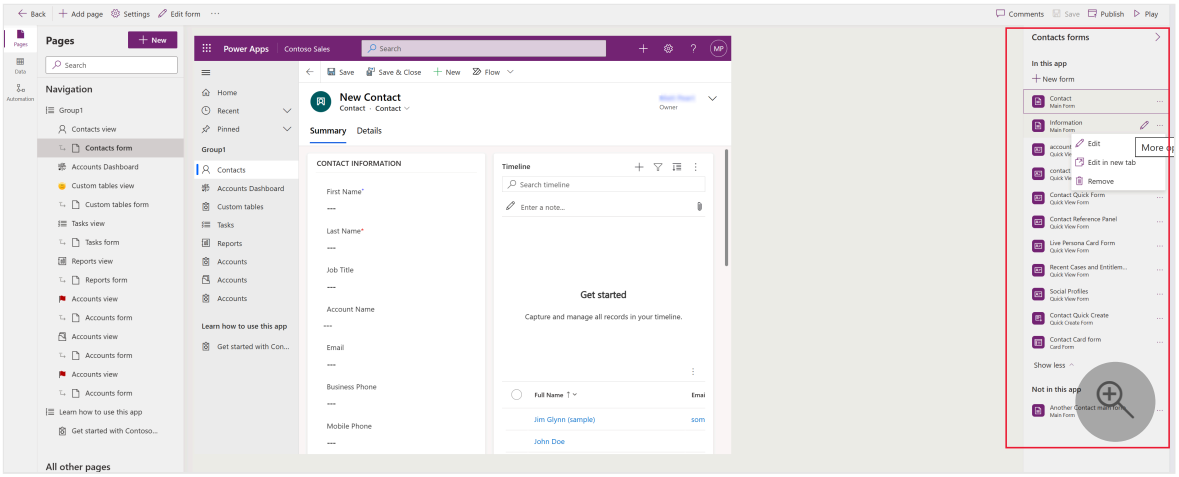
Add or remove one or more forms, views, or charts for a model-driven app.

Add or remove forms from an app

To add or remove forms for a model-driven app, follow these steps:

1. In the app designer, on the left navigation pane select **Pages** and then select a table form, such as the **Contact form**.

2. On the right forms pane, the list of available forms is displayed. Forms that are available in the app appear under **In this app**. Forms available to add but not displayed in the app appear under **Not in this app**. To remove a form that's displayed, select **...** > **Remove**. To add a form that isn't displayed, select **...** > **Add**.



> **Note**
>
> By default, all forms are selected and made available in the app. Select specific forms to enable only those forms you want available in the app.

3. Select **Publish** to make the changes available to other users.

Manage views and charts

To add or remove one or more views or charts for a model-driven app, follow these steps:

1. In the app designer, on the left navigation pane select **Pages** and then select the table you want to update, such as the **Accounts view**.
2. On the right properties pane, select the **Views** or **Charts** tab.
3. Next to the views or charts you want to add or remove, select ..., and then select **Remove**, or if the item is under **Not in this app**, select **Add**.
4. Select **Save**.
5. Select **Publish** to make the changes available to app users.

> **Note**
>
> By default, all views and charts are selected and made available in the app. Select specific views or charts to enable only those components you want available in the app.

See also

Create and design model-driven app forms

Understand model-driven app views

Create a model-driven app system chart

Feedback

Was this page helpful? Yes No

Provide product feedback

Enable your app and custom tools in Microsoft 365 Copilot

[This topic is pre-release documentation and is subject to change.]

Power Apps in Microsoft 365 Copilot lets users interact with your model-driven app directly from Copilot. When a user talks to the agent, it can surface your app's data as interactive widgets, a grid for browsing records or a form for viewing, editing, or creating them, all without leaving the Copilot experience. Additionally, you can add custom MCP tools and interactive UI to the app's declarative agent. This functionality is achieved via MCP apps which is an extension to MCP that enables MCP servers to deliver interactive user interfaces to hosts. The feature works by generating an MCP server and declarative agent from your app.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

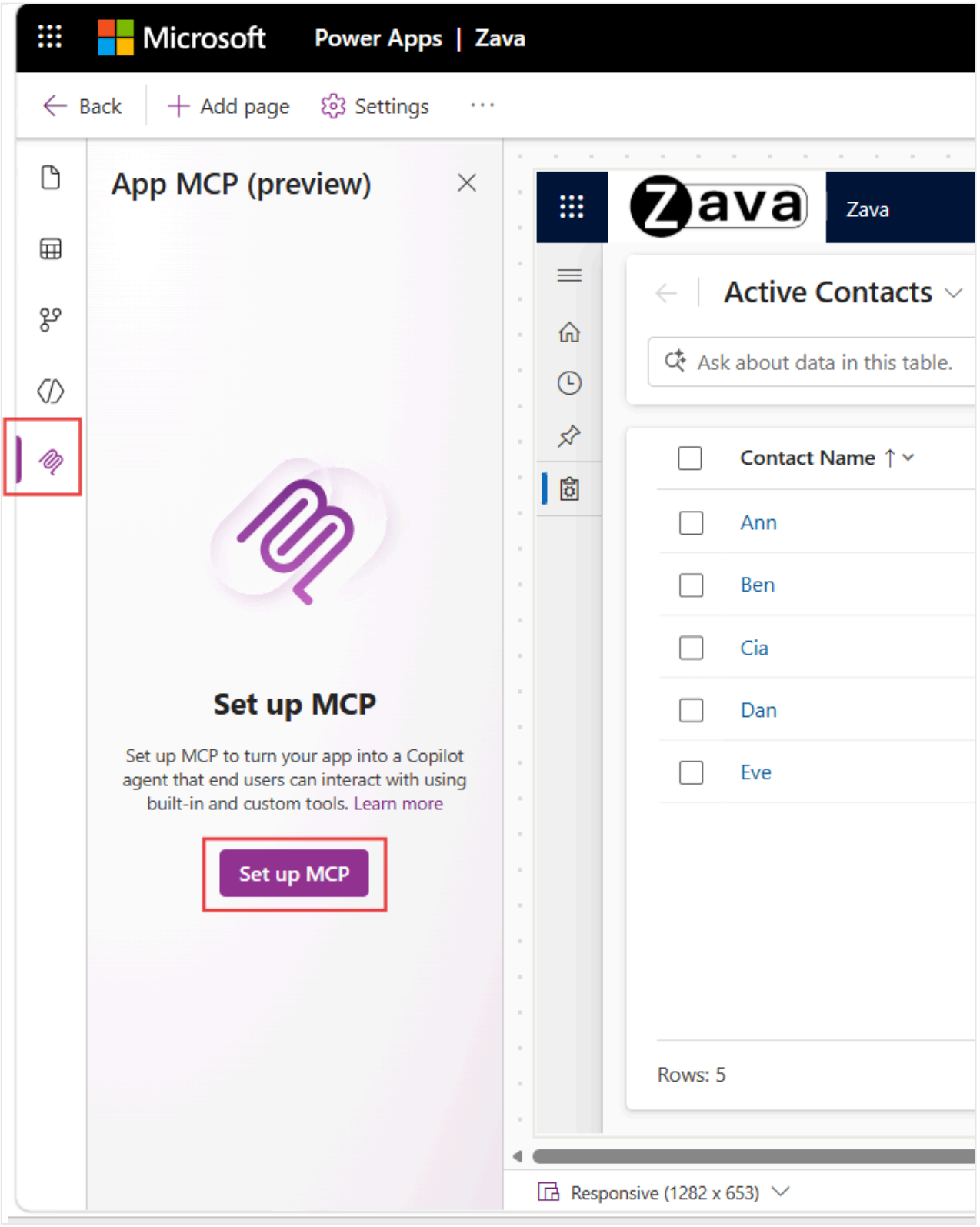
Prerequisites

- A model-driven app. Currently, this feature is only available with model-driven apps.
- Microsoft 365 Copilot license. Required for both the maker deploying the agent and end users interacting with it.
- Permission to upload custom apps in Microsoft Teams. Your Microsoft 365 admin might need to enable this. Check your Teams admin settings if the upload option is not visible.

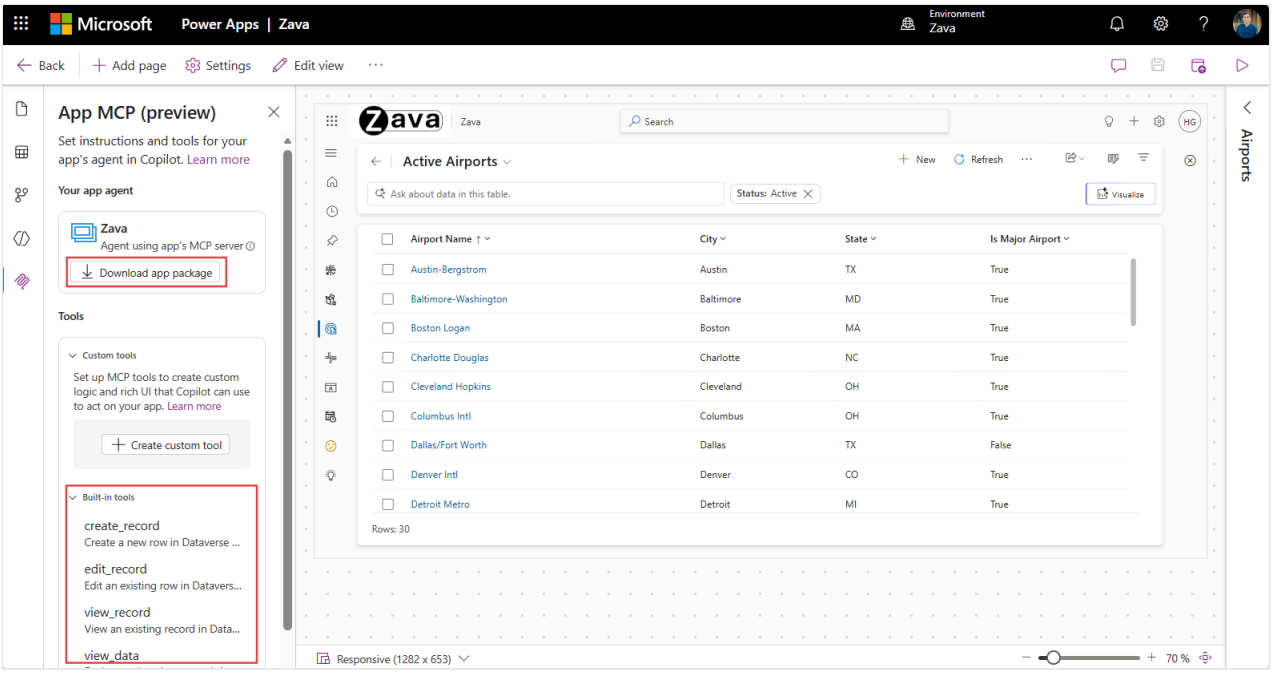
Set up Power Apps in Copilot

1. Go to Power Apps, select the environment you want, and then open the desired model-driven app.

2. Select the **App MCP** icon on the left navigation bar, and then select **Set up MCP**. This action sets up the MCP server for the app, which provides MCP tools for Copilot conversations. This is a one-time step and may take a few seconds to complete.



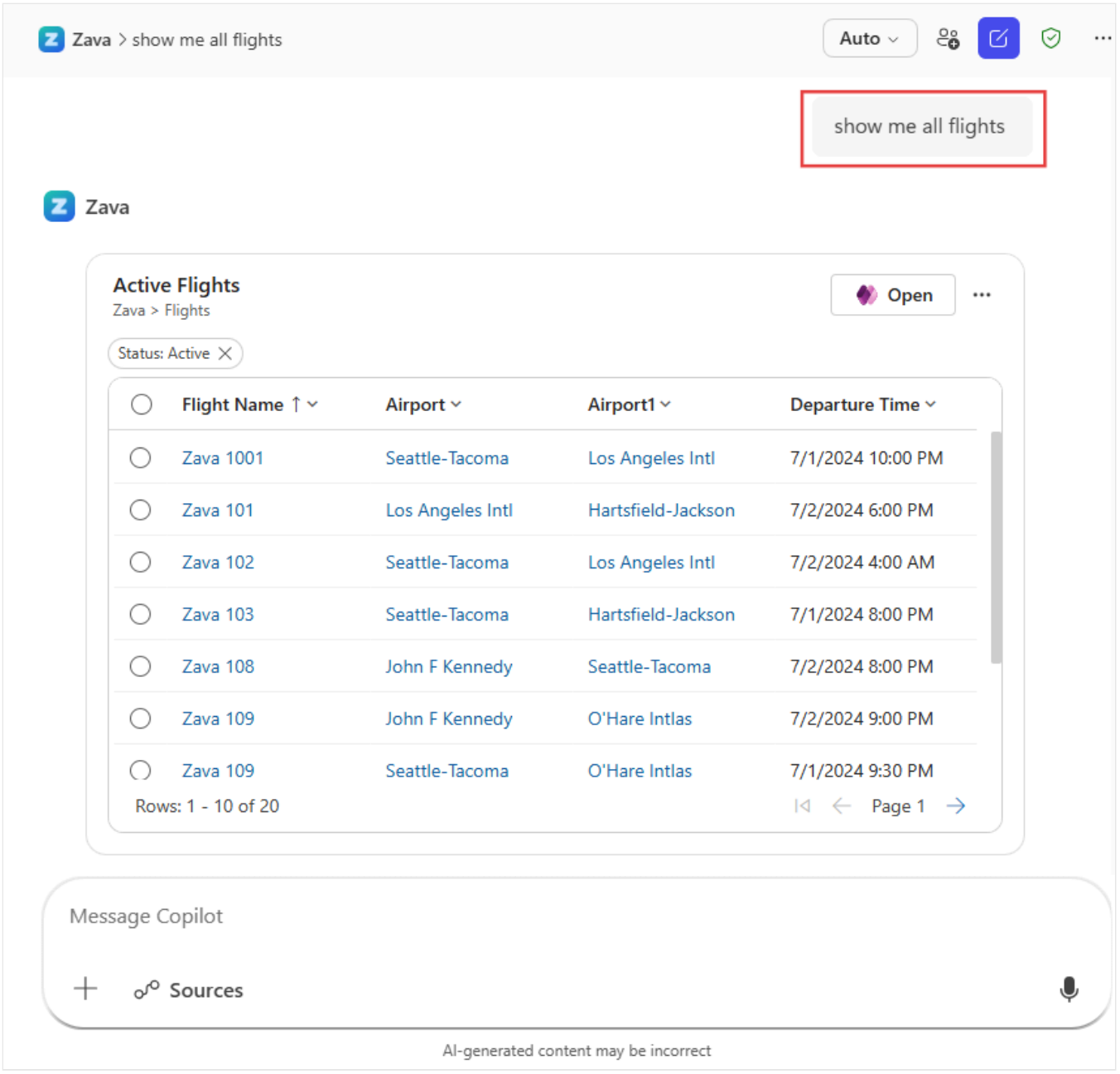
3. Select **Download app package** to download declarative agent generated by your app's MCP. This package contains the agent definition, built-in tools for record and data along with the configuration needed to deploy your app's experience to Microsoft 365 copilot.



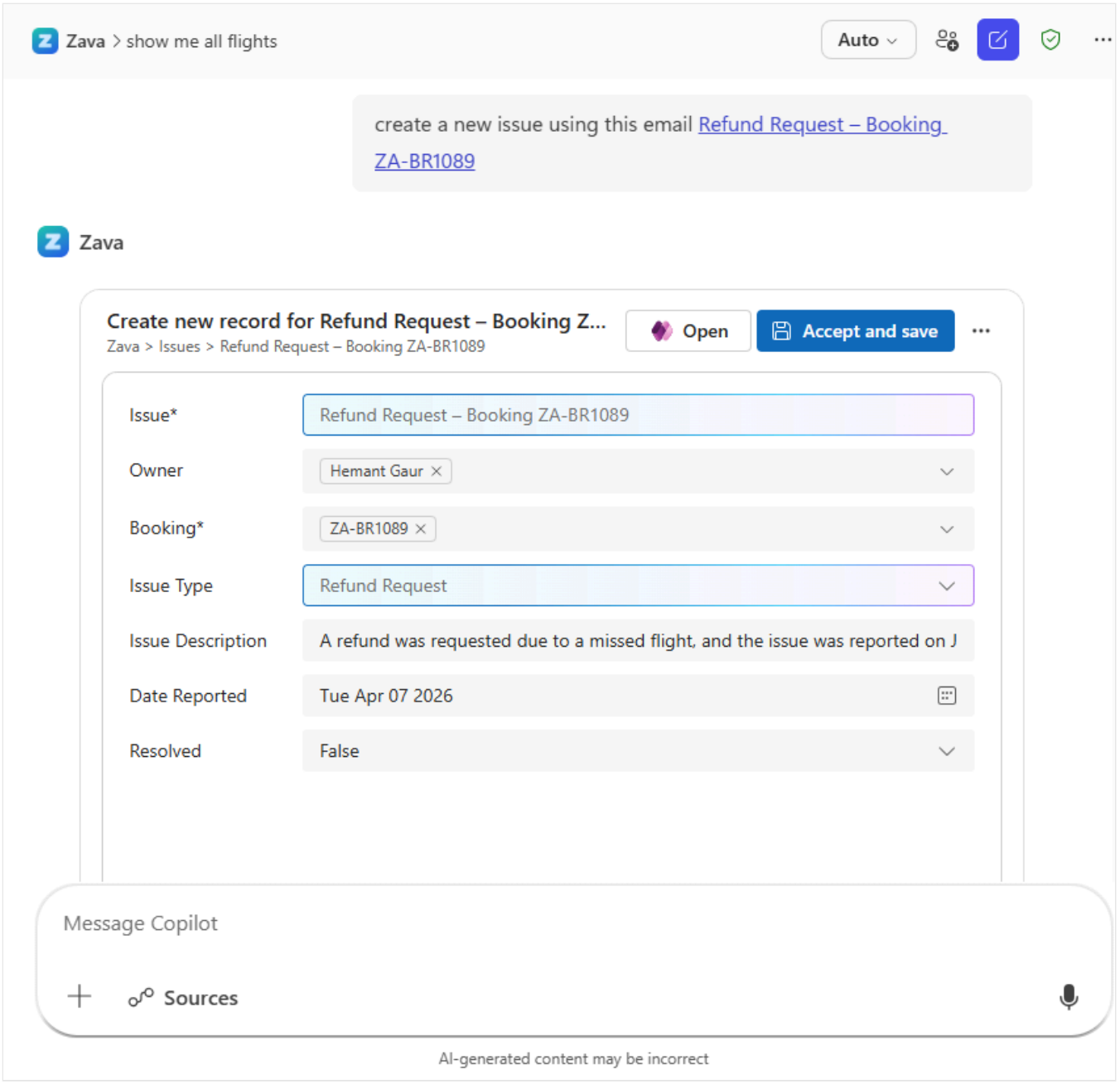
4. Upload the compressed file that is named similar to *declarative-agent-app name.zip* to Microsoft Teams or Microsoft 365 Agents:

    - Teams: Upload for personal use as a custom app in Teams. More information Upload your app in Teams
    - Microsoft 365 Agents: Publish the package for a team or group from the Microsoft 365 admin center. More information: Publish agents

You can now ask questions about the tables in the app from this declarative agent. Built-in tools can show an interactive grid for the dataset queries.



Additionally, intelligent forms can be used to create, view, and edit records within the Microsoft 365 Copilot context.



> **Note**
>
> You can customize the declarative agent included in the app package at you own discretion. To reduce the possibility of unexpected behavior when using AI agents, use caution when you customize the declarative agent code. More information: Declarative Agents for Microsoft 365 Copilot.

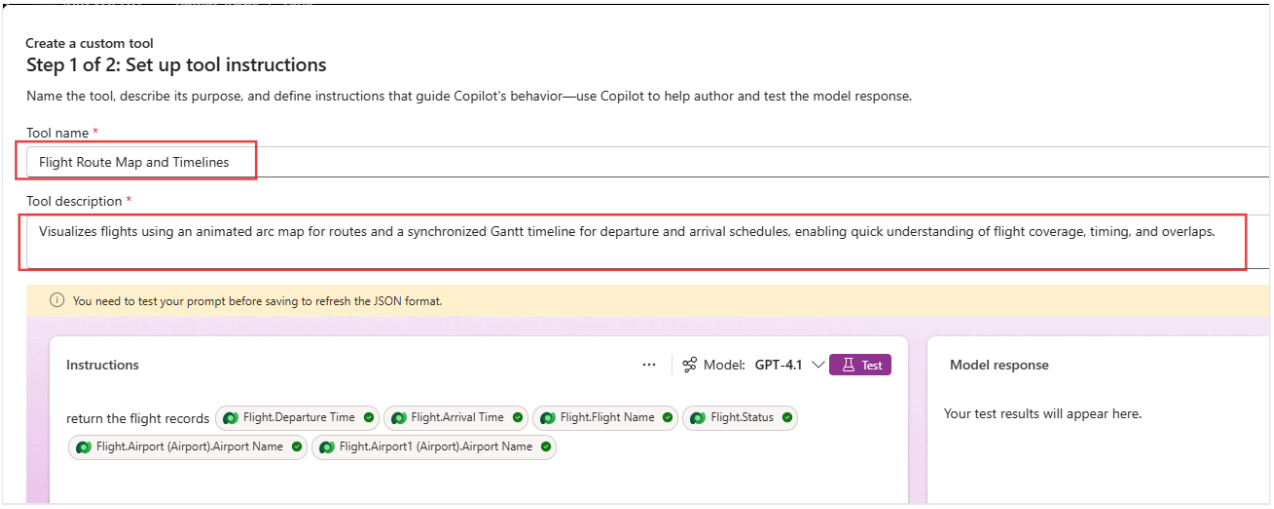
Create custom tools

In addition to the built-in tools available within the app's declarative agent, its functionality can be extended by incorporating custom tools and widgets. A prompt builder–based designer can be used to integrate custom MCP tools and interactive UI components into the app's

declarative agent. For example, in the Zava Airlines app, a custom tool can be added to display the flight map alongside the flight timeline as a Gantt chart in a side-by-side view.

> **Note**
>
> Custom tools and UX is based on Microsoft 365 Copilot extensibility for declarative agents. More information: Microsoft 365 Copilot Extensibility and Supported capabilities

1. Select **Create custom tool** under the **Tools** section of the **App MCP** tab.
2. Provide a clear, descriptive **name** and **description** for the tool, as Microsoft 365 Copilot uses this information to determine when to invoke it.
3. Add the instructions to output data for the tool that defaults to JSON format. You can change it to text if the tool doesn't have UX associated with it. Instructions can include queries for specific records of the table in the app or it could be an input parameter to the tool to be used in tool-chaining. Make sure to experiment and choose the right prompt AI model.

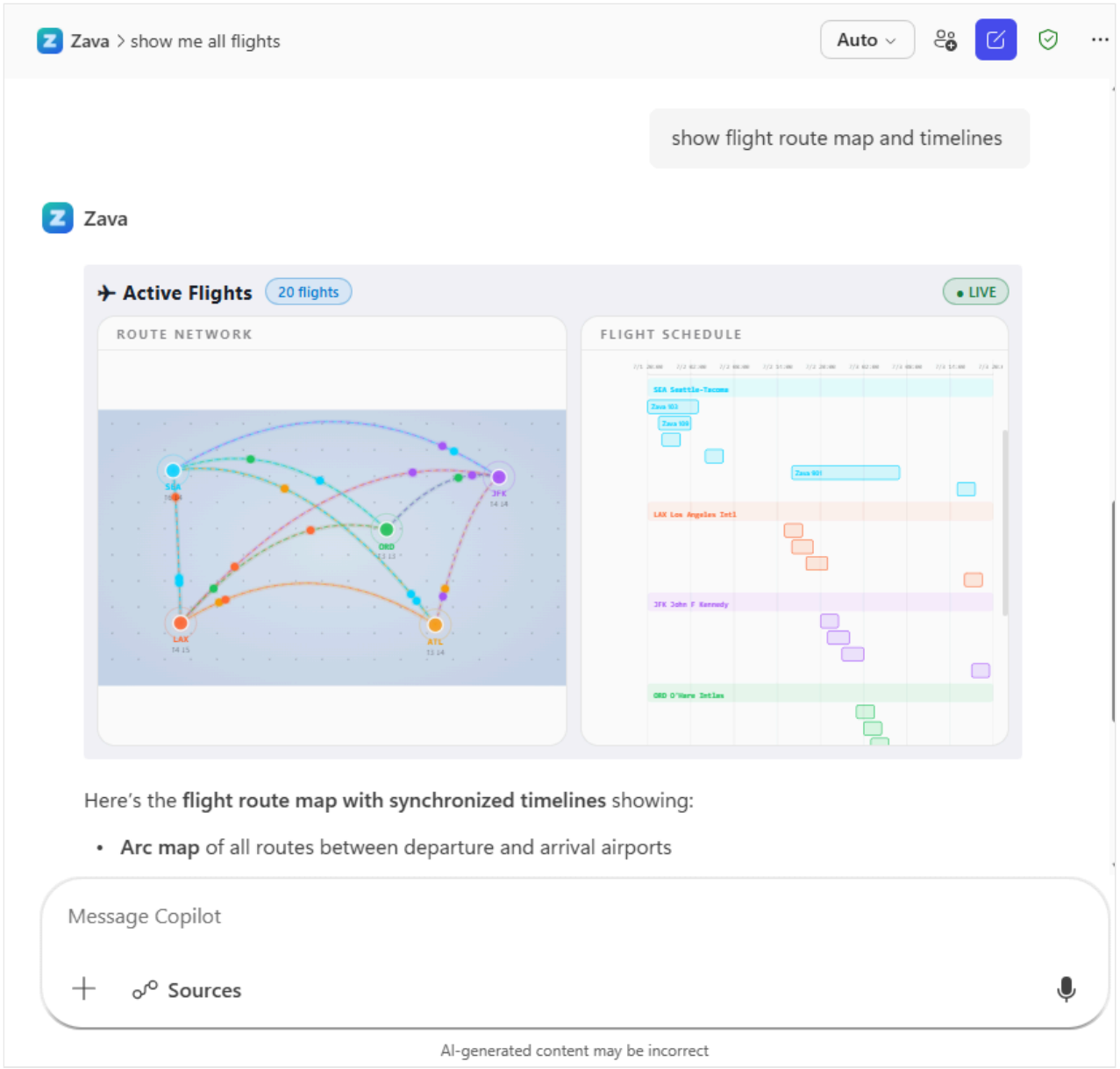


4. Select **Test** to check and validate tool's output and then select **Next**. You can now create and paste optional custom HTML specific to this tool. This HTML can be created using the `/generate-mcp-apps-ui skill` or your own development tool. Select **Save**.



5. Download the updated app package zip and upload the latest version to Microsoft Teams or Microsoft 365 agents.

6. Invoke the tool by issuing the relevant query in the app agent to see the custom UI in action.



> **Note**
>
> You can issue '-developer on' command to Microsoft 365 Copilot to see agent debug info and tools invoked.

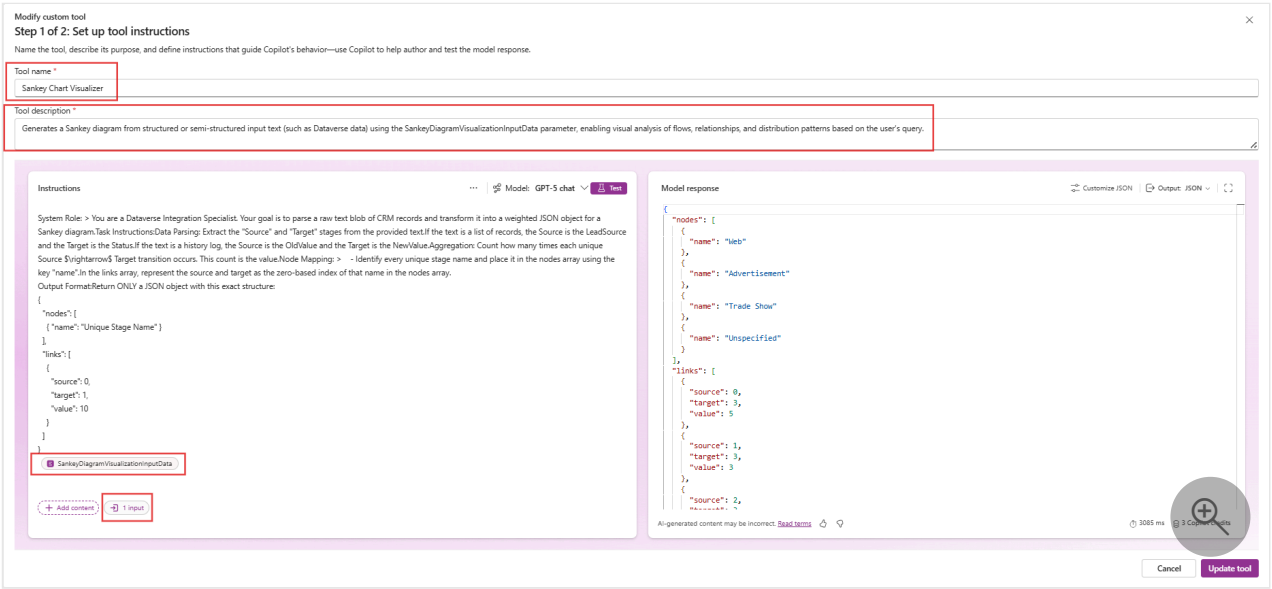
Dynamic tool chaining

Tool chaining in Power Apps declarative agents uses the Microsoft 365 Copilot orchestrator to reason over user intent and dynamically invoke multiple MCP-powered actions (tools) in sequence to complete a task. The orchestrator selects, parameterizes, and executes the right combination of MCP app actions based on context and function descriptions, enabling end-to-end workflows across Power Apps and enterprise systems. Here's how to create a tool with parameterized input.

1. Select **Create custom tool** under the **Tools** section of the **App MCP** tab.

2. Provide a clear, descriptive **name** and **description** for the tool, as Microsoft 365 Copilot uses this information to determine when to invoke it.

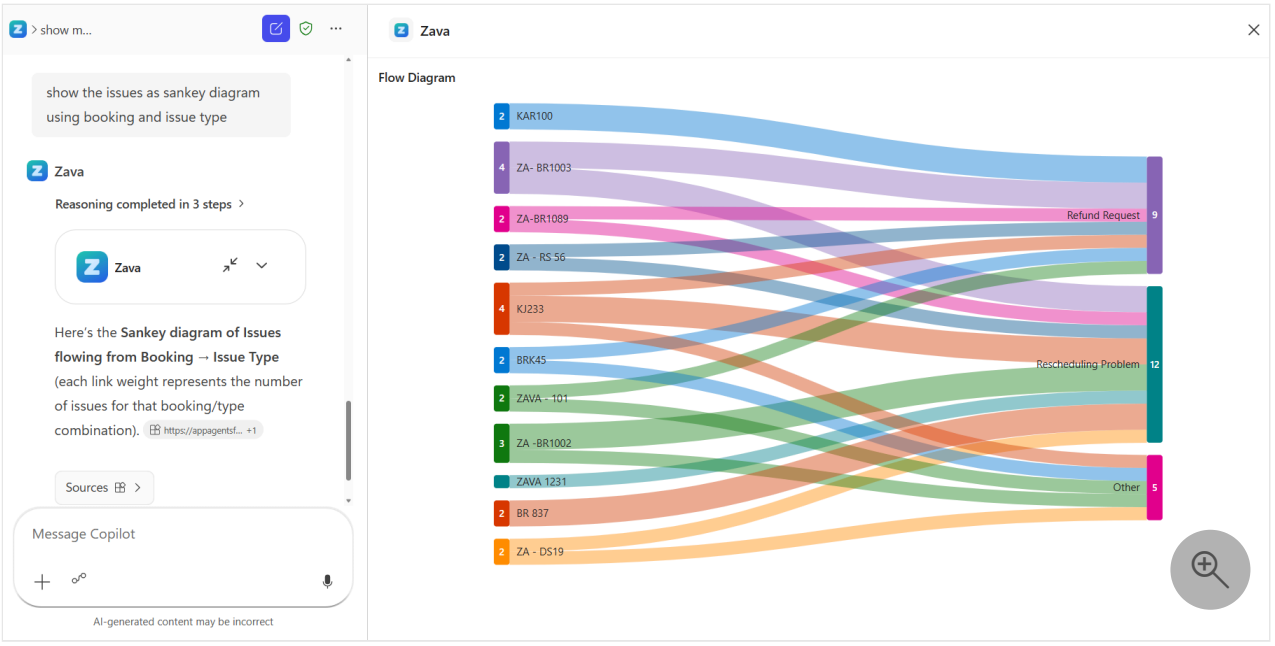
3. In this example, we use a Sankey chart visualizer. First, define the JSON input schema required to generate a Sankey chart. Next, create a prompt for the Sankey chart visualizer tool. The tool exposes an input parameter named `SankeyDiagramVisualizationInputData`. This parameter is interpreted by the tool and transformed into the JSON structure expected by the visualizer. Choose a clear and descriptive input parameter name so the LLM can correctly identify and invoke the tool.



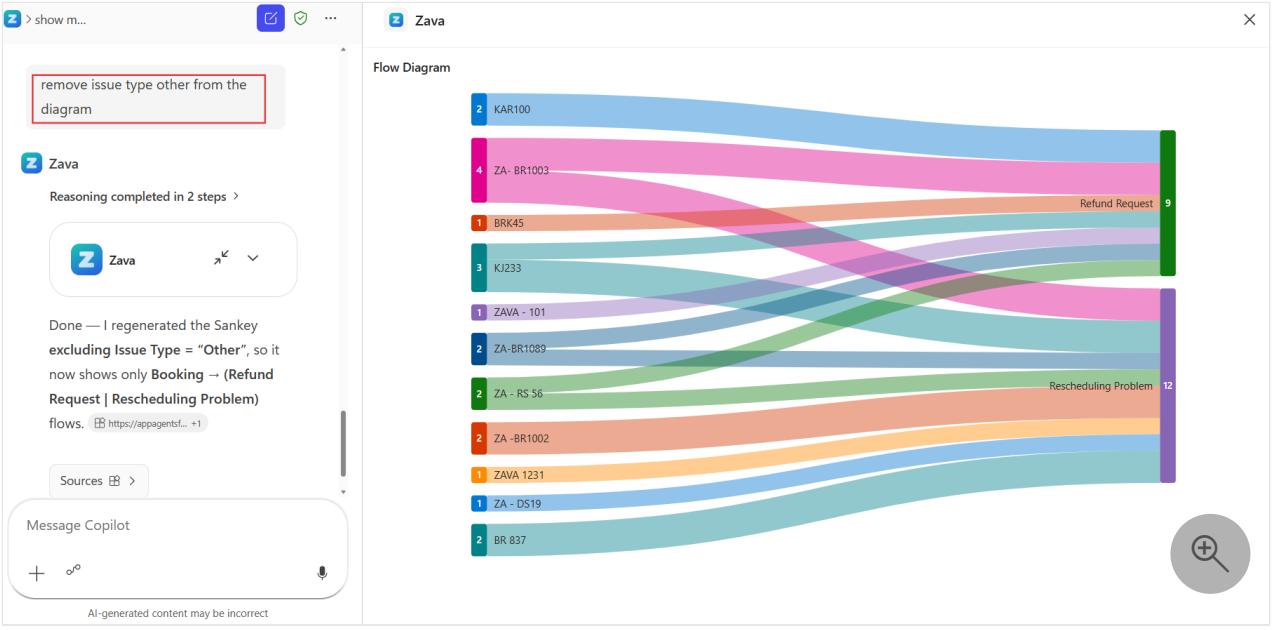
4. You can now create and paste custom HTML specific to this tool. This HTML can be created using the `/generate-mcp-apps-ui skill` or your own development tool. Select **Save**.

5. Download the updated app package zip and upload the latest version to Microsoft Teams or Microsoft 365 Agents.

6. Copilot can now use natural language queries to fetch the data and pass it to the tool for visualization. For example you can use a complex query to dynamically get inputs from the 'issues' table data and visualize them using Sankey Chart Visualizer tool.



7. You can further refine the Sankey chart by continuing the conversation.



Full screen example

In this example, an expand button is rendered only if the host supports fullscreen mode.

js

```
function renderExpandButton(app) {
  const ctx = app.getHostContext();

  // Don't render the button if the host doesn't support fullscreen
  if (!ctx?.availableDisplayModes?.includes('fullscreen')) {
    return null;
  }

  const btn = document.createElement('button');
```

```
  btn.addEventListener('click', async () => {
    const result = await app.requestDisplayMode({ mode: 'fullscreen' });
    // Always use result.mode — the host may grant a different mode than requested
    console.log('Granted mode:', result.mode);
  });
  return btn;
}
```

The same guard applies to any host capability. Check `getHostContext()` before use:

js

```
const ctx = app.getHostContext();

// Theme
if (ctx?.theme) applyTheme(ctx.theme);

// Current display mode
if (ctx?.displayMode === 'fullscreen') { /* adjust layout */ }

// Available modes
if (ctx?.availableDisplayModes?.includes('inline')) { /* show collapse option */ }
```

React to capability changes at runtime via `onhostcontextchanged` — the host context can update after `connect()`:

js

```
app.onhostcontextchanged = (ctx) => {
  if (ctx.theme) applyTheme(ctx.theme);
  if (ctx.displayMode) updateLayout(ctx.displayMode);
};
```

Using fullscreen mode with tool results

A common pattern is to switch to fullscreen automatically when the widget receives data, giving the content more space to render.

js

```
app.ontoolresult = async (result) => {
  const data = result.structuredContent;
  if (!data) return;

  // Expand to fullscreen when data arrives, if the host supports it
  const ctx = app.getHostContext();
  if (ctx?.availableDisplayModes?.includes('fullscreen') && currentMode !==
'fullscreen') {
```

```
    const granted = await app.requestDisplayMode({ mode: 'fullscreen' });
    currentMode = granted.mode;
    document.body.classList.toggle('mode-fullscreen', currentMode ===
'fullscreen');
  }

  renderData(data);
};
```

You can also let the user decide — render into the current mode first, then offer the expand button:

js

```
app.ontoolresult = (result) => {
  const data = result.structuredContent;
  if (!data) return;

  renderData(data);         // render in whatever mode is active
  updateExpandBtn();        // show the expand button now that there is content
};
```

Related articles

Generate MCP app widgets with AI code generation tools

FAQ for Power Apps in agents

Last updated on 04/22/2026

Generate MCP app widgets with AI code generation tools

[This topic is pre-release documentation and is subject to change.]

This article explains how to use AI code generation tools like GitHub Copilot CLI or Claude Code to generate interactive model context protocol (MCP) apps for your model-driven Power Apps MCP tools. MCP apps are self-contained HTML files that render a tool's JSON output visually as cards, charts, dashboards, or maps inside any MCP Apps compatible host, including Microsoft 365 Copilot, Claude, and Visual Studio Code.

If you have an MCP tool that returns JSON data, the `generate-mcp-app-ui` skill can produce a polished, theme-aware widget that displays that data in a compact visual format directly inside a chat conversation.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.
> - MCP apps support in Microsoft 365 Copilot Chat is generally available as of March 2026. Power Apps support for MCP apps in declarative agents is currently in public preview. For the full announcement, see MCP Apps now available in Copilot Chat.

What you can do with the generate-mcp-app-ui skill

- *Create visual widgets* for any MCP tool by describing what you want and pasting the tool's JSON output.
- Choose the right visual for your data, such as charts for numeric trends, cards for structured records, tables for comparisons, maps for coordinates, and so on.
- Support light and dark themes automatically through Fluent UI design tokens.
- Add interactivity so widgets can call your tool again at runtime (for example, a refresh button).
- Refine the UX iteratively by describing changes in natural language. For example, "make the gallery compact," "add a chart," or "use a card layout."

Prerequisites

Software requirements

Expand table

Component	Minimum version	More information
GitHub Copilot CLI, Claude Code, or other code generation tool	Latest	Claude Code, GitHub Copilot CLI
A modern browser	Any	For previewing generated widgets locally

Additional requirements

- An MCP tool that returns JSON output. Your tool's output type must be set to **JSON**.
- A working internet connection. Widgets load Fluent UI and other libraries from the Content Delivery Network (CDN) at runtime.

Install the plugin

Run the following installer command from GitHub Copilot CLI or Claude Code. The installer automatically detects available tools and installs all Power Platform plugins, including `generate-mcp-app-ui`.

```
/plugin marketplace add microsoft/power-platform-skills
```

To install only the MCP App widget skill:

```
/plugin install mcp-apps@power-platform-skills
```

> **Tip**
>
> Turn on auto-update to automatically receive skill updates. Use the `/plugin` command, navigate to **Marketplaces**, choose the marketplace, and turn on auto-update.

Skills overview

Expand table

Skill	Command	Description
MCP Apps widget generator	`/generate-mcp-app-ui`	Generate a self-contained MCP app widget (HTML file) for an MCP tool's JSON output

The skill is also triggered by natural language phrases such as "create a widget," "build a widget for my tool," or "make an MCP app."

Generate a widget

Follow these steps to create a new widget for an MCP tool.

1. **Create and test a custom tool** from model-driven app designers and copy the full JSON output. Make sure the tool's output type is set to JSON. More information: Create custom tools

2. **Invoke the skill** and describe what you want displayed, pasting the JSON output into the conversation:

   ```
   /generate-mcp-app-ui Visualizes flights using an animated arc map for routes and a synchronized Gantt timeline for departure and arrival schedules, enabling quick understanding of flight coverage, timing, and overlaps. Here's an example of the tool's output: {"flight_records":[{"Departure Time":"2024-07-02T05:00:00Z","Arrival Time":"2024-07-02T07:30:00Z","Flight Name":"Zava 1001","Status":"Active","Airport":"Seattle-Tacoma","Airport1":"Los Angeles Intl"},{"Departure Time":"2024-07-02T03:00:00Z","Arrival Time":"2024-07-02T10:00:00Z","Flight Name":"Zava 103","Status":"Active","Airport":"Seattle-Tacoma","Airport1":"Hartsfield-Jackson"}]}
   ```

3. **Review the generated HTML file.** The skill writes a self-contained HTML file, for example, `flight-map.html`, to your working directory.

4. **Preview in a browser.** Open the HTML file locally as widget has the fallback option for testing. You can ask chat agent to add standalone HTML preview if missing.

5. **Iterate.** Describe any changes directly in the chat:

   - "Make the map larger"
   - "Add tool tips on the Chart"
   - "Decrease the height and fit in 250 pixels with responsive layout and no scroll bars"

> ⓘ Note

The skill requires actual JSON from your tool—not sample or mock data. The data shape drives the widget generation. If you paste mock data, the generated widget might not work correctly when connected to the real tool.

Deploy your widget

Once your widget is ready, copy the HTML file to the UX input for corresponding tool and it will be returned as the tool's UI response. Refer to your create custom tools documentation for details.

Add interactivity with callServerTool

If you also provide your tool's name when invoking the skill, the generated widget can include interactive tool-call integration. This allows the widget to call your tool again at runtime. For example a refresh button on the tool UX can call itself.

```
/generate-mcp-app-ui Show the current weather conditions with a refresh button.
Tool name: get_weather. Tool output:
{"city":"Seattle","temp_f":54,"condition":"Overcast","humidity":78,"forecast":
[...]}
```

The skill wires up `app.callServerTool` in the widget so that when users select **Refresh**, the widget fetches updated data directly from your tool. If you don't provide a tool name, the widget is read-only and renders only the data delivered through the `ontoolresult` callback.

- **Microsoft 365 Copilot chat**: See MCP apps in Copilot Chat for deployment paths including sideloading for testing, deploying through the Microsoft 365 admin center for organizational use, and publishing to the Microsoft 365 agent store.
- **Power Apps declarative agents**: See Power Apps MCP declarative agent documentation for how to connect MCP tools with model-driven apps.
- **Other MCP hosts**: Consult your host's documentation for the MCP apps widget registration process.

Widget technical details

MCP apps protocol

Widgets communicate with the chat host using the `App` class from the `@modelcontextprotocol/ext-apps` package. The protocol manages these callbacks and methods.

Expand table

Callback / method	Description
`app.ontoolresult`	Fires when the host delivers tool data. *Your data is always at* `result.structuredContent` —not `result.data` or `result` itself.
`app.onhostcontextchanged`	Fires when the host context changes, including theme (`ctx.theme` is `'light'` or `'dark'`).
`app.onteardown`	Fires when the widget is removed from the conversation.
`app.connect()`	Establishes communication with the host. All event handlers must be registered *before* calling `connect()`.
`app.getHostContext()`	Returns the current host context (including initial theme) after `connect()` completes.
`app.callServerTool({ name, arguments })`	Calls a tool interactively. Returns `result.isError` and `result.structuredContent`.

CDN imports

Widgets load all dependencies from CDN. No build step or local installation is required. Dependencies come in two formats:

- **ECMAScript Modules (ESM)** — imported inside `<script type="module">` using a URL ending in `/+esm`
- **Universal Module Definition (UMD)** — loaded via a plain `<script src>` tag; registers itself globally as a side effect

Expand table

Library	Format	URL	Purpose
`@modelcontextprotocol/ext-apps`	ESM	`cdn.jsdelivr.net/npm/@modelcontextprotocol/ext-apps/+esm`	MCP apps `App` class
`@fluentui/tokens`	ESM	`cdn.jsdelivr.net/npm/@fluentui/tokens/+esm`	`webLightTheme` / `webDarkTheme` token sets
`@fluentui/web-components@beta`	UMD	`unpkg.com/@fluentui/web-components@beta/dist/web-components.min.js`	Fluent UI custom elements

Visual states

Every widget handles three states:

Expand table

State	Guidance
Loading	Show a `<fluent-spinner>` with a contextual message ("Finding attractions..." not just "Loading...").
Loaded	Render the content compactly. Use the full available width.
Error	Show a friendly message and a "Try again" button. If the widget uses `callServerTool`, the button re-invokes the tool.

Fluent UI components

The following Fluent UI web components are available in widgets:

`<fluent-card>`, `<fluent-button>`, `<fluent-text-input>`, `<fluent-textarea>`, `<fluent-dropdown>`, `<fluent-listbox>`, `<fluent-option>`, `<fluent-checkbox>`, `<fluent-spinner>`, `<fluent-divider>`, `<fluent-badge>`, `<fluent-switch>`, `<fluent-tooltip>`

Theme support

Widgets support light and dark themes through Fluent UI design tokens. The widget applies the correct token values when the host's theme changes via `onhostcontextchanged`. Always use token variables, for example, `var(--colorNeutralForeground1)`, rather than hardcoded color values to ensure correct rendering in both themes.

Color tokens

Expand table

Use	Token
Primary text	`var(--colorNeutralForeground1)`
Secondary text	`var(--colorNeutralForeground2)`
Primary background	`var(--colorNeutralBackground1)`
Card/hover background	`var(--colorNeutralBackground2)`
Brand/accent	`var(--colorBrandBackground)`
Text on brand surface	`var(--colorNeutralForegroundOnBrand)`

Use	Token
Borders	`var(--colorNeutralStroke1)`
Error text	`var(--colorStatusDangerForeground1)`
Success text	`var(--colorStatusSuccessForeground1)`

Never use hardcoded hexidecimal or RGB values. Don't invent token names not listed here.

Best practices

- Provide real test data. The skill analyzes the actual JSON structure to select the right visual. Mock data produces widgets that break when connected to the real tool.
- Be specific about the visual. Describe the format you want, such as map, chart, table, or card layout. Vague descriptions lead to generic results.
- Start with one view. Widgets are compact conversation cards, not full applications. No tabs, page navigation, or search bars that duplicate the chat input.
- Test with both themes. Preview in light and dark mode to verify contrast and readability.
- Match the visual to the data. Maps for coordinates, charts for numeric or trend data, cards for structured records, tables for comparisons.

Limitations

- Widgets must load all external libraries from CDN. An internet connection is required at runtime.
- Full screen display mode requires additional implementation beyond what the skill generates.
- The skill doesn't handle MCP server registration or deployment to the Microsoft 365 admin center. You must complete those steps separately.
- Authentication (OAuth 2.1, Microsoft Entra SSO) is handled by the MCP host environment, not the widget HTML itself.

Related documentation

Microsoft 365 developer documentation

- MCP Apps now available in Copilot Chat
- Microsoft 365 Agents Toolkit

Power Platform documentation

- Create and edit generative pages with AI code generation tools
- Power Platform CLI reference

External references

- Model Context Protocol (MCP) specification
- Fluent UI Web Components
- Power Platform Skills repository

Last updated on 04/22/2026

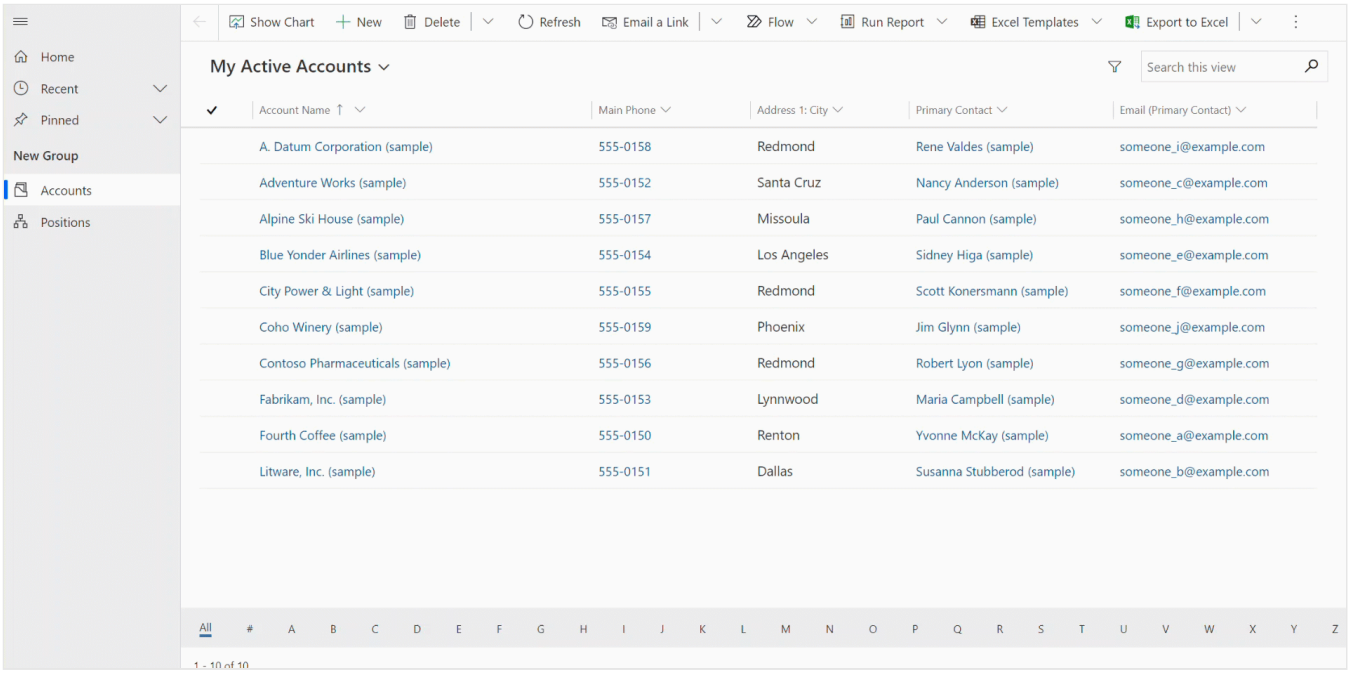
Understand model-driven app views

Model-driven apps use views to define how a list of records for a specific table are displayed in an application.

A view defines:

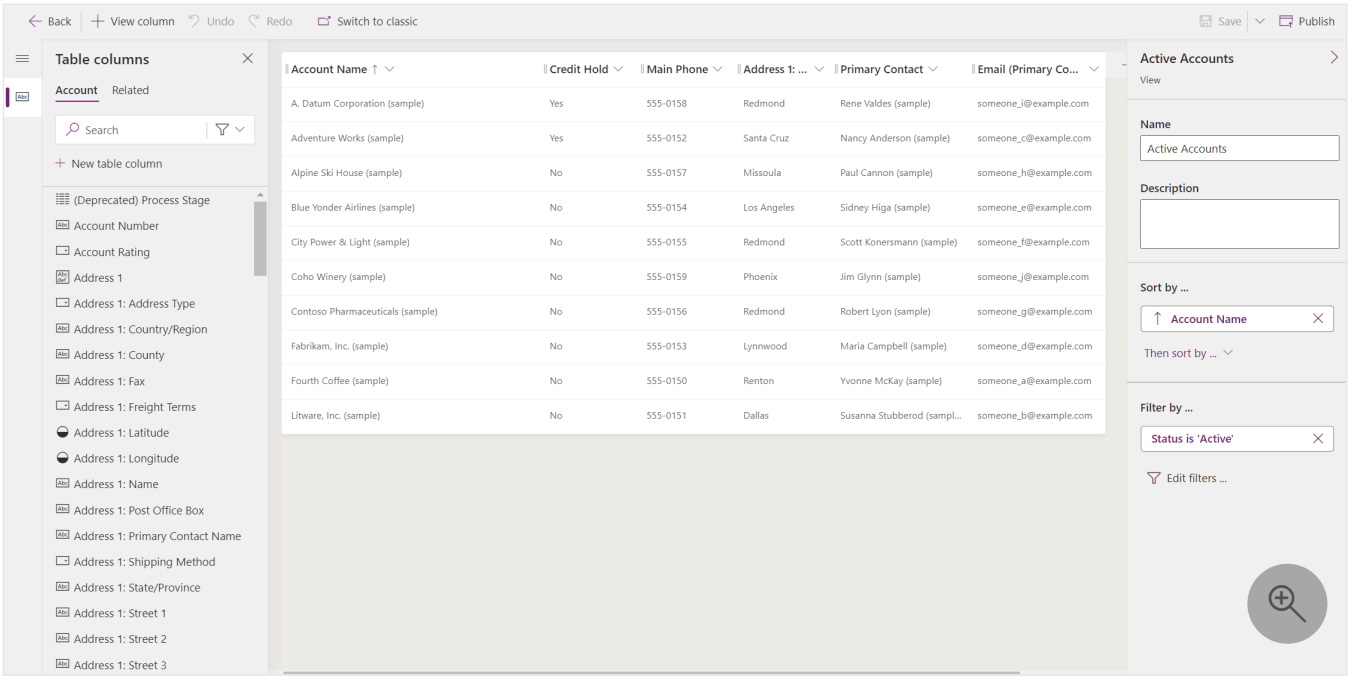
- The columns to display.
- The order of the columns.
- How wide each column should be.
- How the list of records should be sorted by default.
- The default filters applied to restrict the records that appear.

Once a view is available to the app, the user can select it.



When makers design an app, they decide which of the public views to make available to app users. These decisions are typically based on the type of user, such as sales or marketing, that use the app.

Views can be developed through the table designer or the app designer.



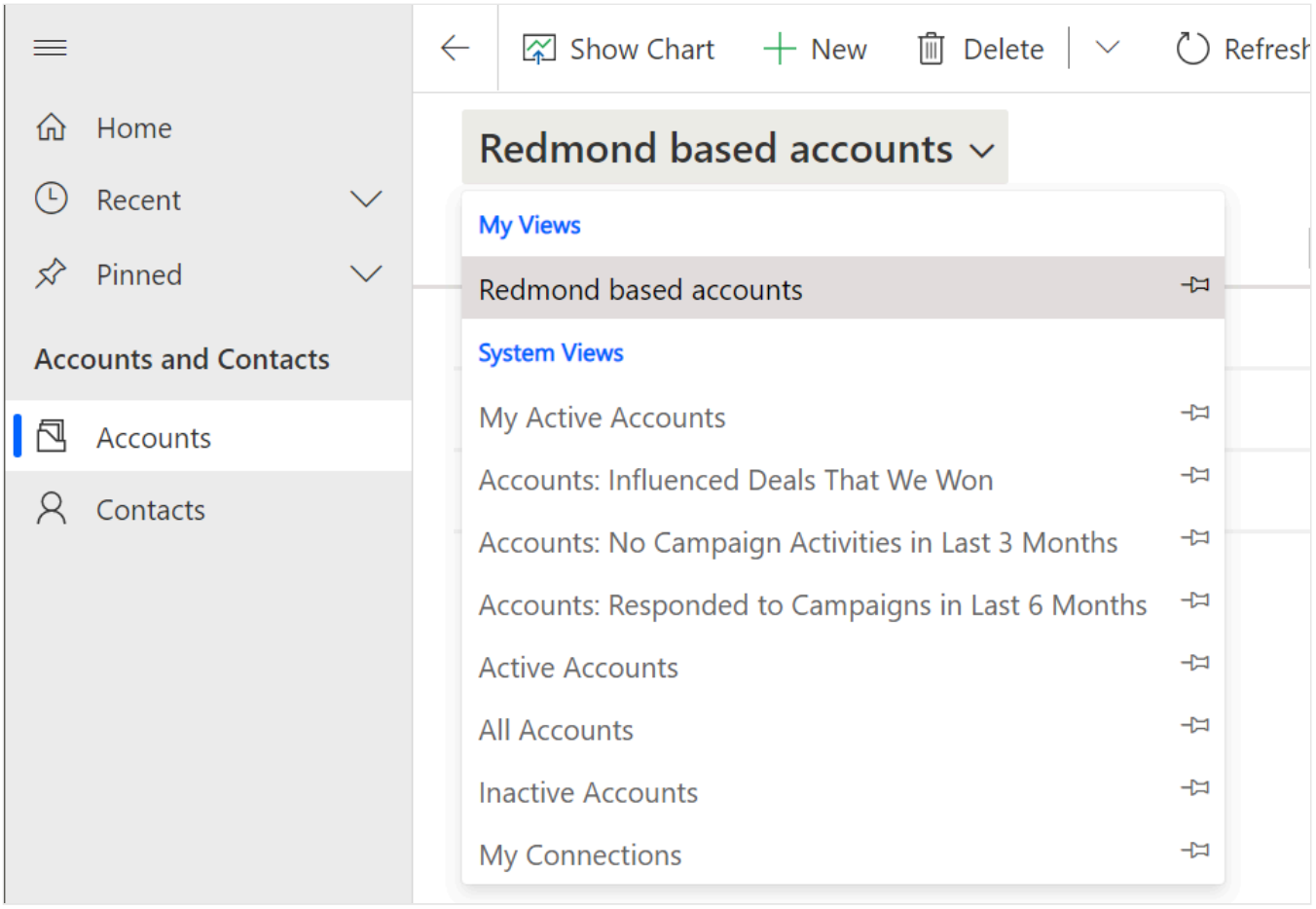
Types of views

There are three types of views: *personal*, *system*, and *public*.

- **Personal view** - Personal views are owned by individuals and only visible to that person unless they share their personal views with others.
- **System view** - As a system administrator or system customizer, you can edit system views. System views are special views the application depends on, which exist for system tables or are automatically created when you create custom tables. These views have specific purposes and some additional capabilities.
- **Public view** - Public views are general purpose views that you can customize as you see fit. They're important because all app users can access them, when they're made available, by using the view selector. It's possible to use public views in subgrids on a form or as a list in a dashboard.

Views within model-driven apps

Users might want to view data in relation to a table in a range of ways. A drop-down list of views is frequently displayed in the application so these can be selected.



Personal views are included above the list of system or public views that are available in the app. This makes it easier for users to find the data that is important to them.

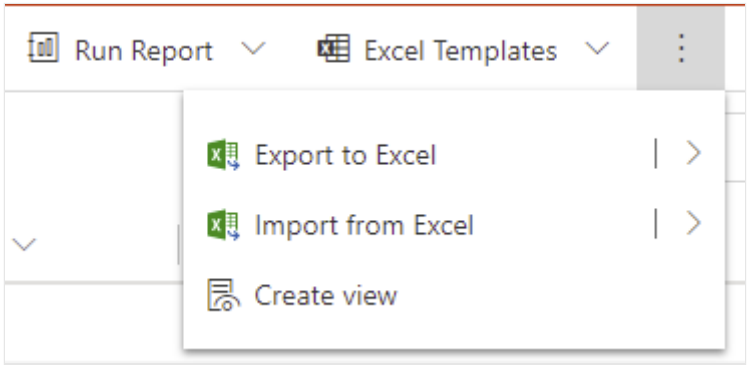
The records that are visible in views are displayed in a list. Views frequently provide options for users to change the default sorting, column widths, and filters to more easily find the data that's important to them.

Views aren't only used by users within model-driven apps. Views can also be used to define the data source in, for example, charts that are used in the application.

Personal views

Personal views can be created by following these steps:

1. Select **Create view** from the command bar in your model-driven app.



2. Define the view filters.
3. Select **Save**.

> **Note**
>
> While you can create a new personal view based on a system or public view, you can't create a system or public view based on a personal view.

Personal views can be created by users who have at least User level access to actions for the Saved View table.

As system administrator, you can modify the access level for each action in the security role to control the depth to which people can create, read, write, delete, assign, or share personal views.

System views

Expand table

System views	Description
Quick Find	The default view used when searches are performed using Quick Find. This view also defines which columns are searched when using search capabilities of Quick Find and Lookup column views.
Advanced Find	The default view used to display results when using Advanced Find. This view also defines the columns used by default when new custom public views or personal views are created without defining a view to use as a template.
Associated	The default view that lists the related tables for a record.
Lookup	The view that is displayed when you select a record to set for a Lookup column.

System views aren't shown in the view selector and you can't use them in sublists in a form or as a list in a dashboard. You can't delete or deactivate these views. However, Power Platform admins can control whether system views are available or not to users with security roles. More information: Manage access to public system views (preview)

For information about removing views, go to Remove views.

System views are owned by the organization so that everyone can view them. For example, everyone has organization-level access to read records for the view (`savedquery`) table. These

views are associated with specific tables and are visible within the Solutions area in Power Apps (make.powerapps.com). You can include these views in solutions because they're associated with the table.

> **Note**
>
> System views are cached for performance optimization purposes and therefore plug-ins on the `savedquery` table aren't supported.

Public views

Some public views exist by default for system tables and for any custom table. For example, when you create a new custom table, it has the following combination of public and system views.

Expand table

Name	Type
Active *table plural name*	Public
Inactive *table plural name*	Public
Quick Find Active *table plural name*	Quick Find
Table name Advanced Find View	Advanced Find
Table name Associated View	Associated
Table name Lookup View	Lookup

You can create custom public views. You can delete any custom public views you create in an unmanaged solution.

You can't delete any system-defined public views.

Custom public views added by a managed solution should only be deleted by uninstalling or updating the managed solution.

How to access the view editor to create or edit views

- App designer: If you're working in an app, you might want to use the app designer, which provides a simple and intuitive user interface with drag-and-drop capabilities for creating views. More information: Tutorial: Create and edit public or system views by using the app designer
- Solution area: In Power Apps, go to **Solutions** > open the solution you want > open the table you want, and then select the **Views** area. Selecting a view opens it in the view designer.

Customize views

As a system customizer you can customize views and subgrids through controls by making them read-only or editable as well as display an alphabetic list at the bottom of views or subgrids (jump bar).

The following controls are available:

Expand table

Grid control name	Read-only or editable?	Description
Power Apps grid control	Read-only or editable	This grid control includes accessibility enhancements and will become the default grid control used in views and subgrids. This control will eventually replace all other grid controls. More information: Power Apps grid control
Power Apps read-only grid (deprecated)	Read-only	Modern grid with accessibility enhancements released in October 2021 and became the default read-only grid experience in April 2022. More information: Power Apps read-only grid control
Editable grid (deprecated)	Editable only	Legacy grid control. More information: Make model-driven app views editable using the editable grid control
Read-only grid (deprecated)	Read-only	Legacy grid control that is deprecated. More information: The legacy read-only grid in model-driven apps is deprecated

Next steps

Opening the view designer

Manage access to system views (preview)

Last updated on 04/03/2026

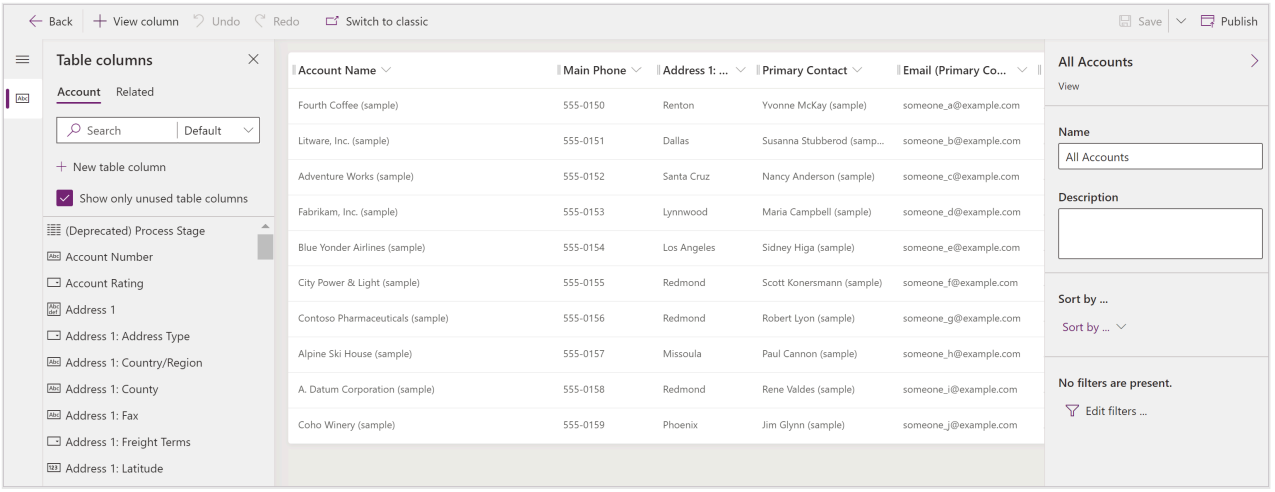
Open the view designer

09/09/2025

In this article, a view definition is opened that displays properties and options to configure a table view using the view designer for use with model-driven apps.

Open a view for editing in Power Apps

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, and then open the solution that needs to be modified. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the table that contains the required view, such as the **Account** table.
4. Select the **Views** area.
5. Select the view, such as the account table **All Accounts** view.



6. From the view editor, the following tasks can be performed:

- Sort rows in a view
- Choose and configure columns in views
- Filtering data within views

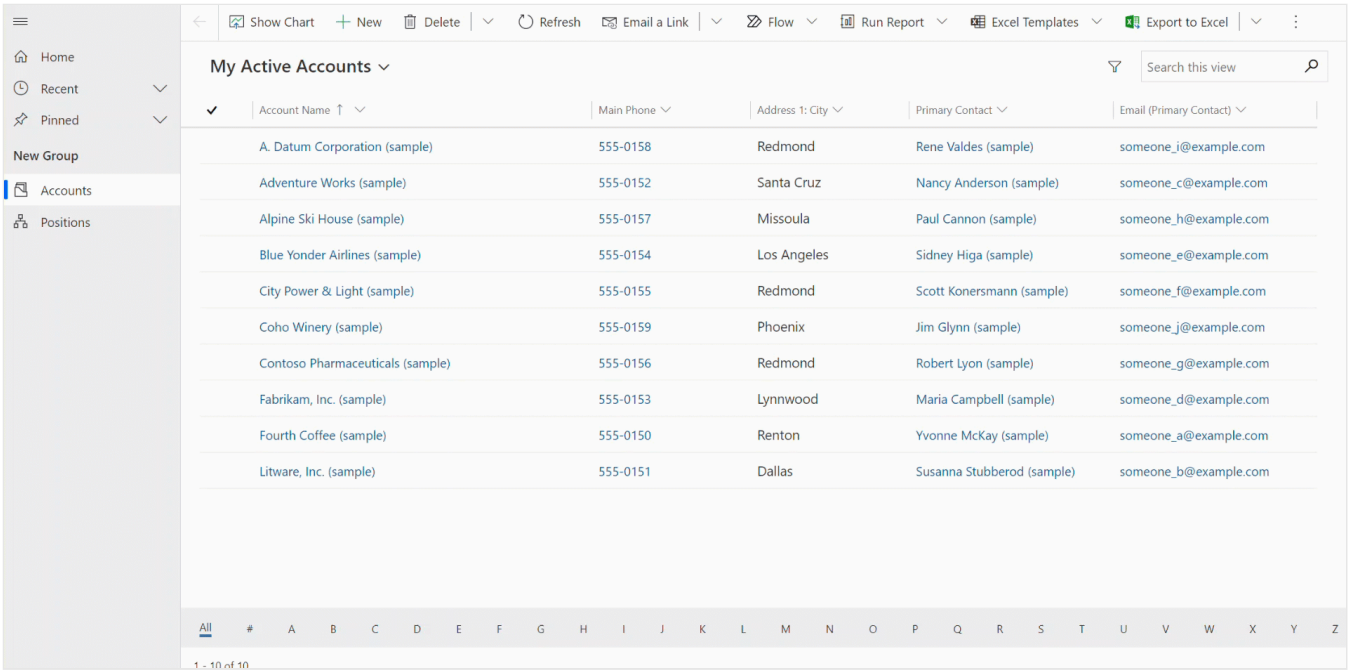
Next steps

Creating and editing public views

Create and edit public or system views

Model-driven apps can contain a range of views, so that app users can open the most suitably presented and filtered version of the table. Views can be **Personal**, **System, or **Public**.

> **Note**
>
> Views are tied to a given table. However, you can select views during the process of creating a model-driven app. By default, ALL views are made available in an app.



> **Tip**
>
> For a general introduction into views, go to Understand views in model-driven apps

Create a public view in Power Apps

Public views are available for users to display table records. App makers can create and configure public views using Power Apps.

1. Sign in to Power Apps and select an environment
2. On the left navigation pane, select **Solutions**, and open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open a table, such as the account table, and then select the **Views** area.
4. Select **New view** on the command bar, and then enter a **View name**, and optionally enter a **Description**.

5. If you want to have suggestions from Copilot for the columns to create for the form based on the **View name** and **Description** values, select **Get AI-generated column suggestions**. For more information about using this AI feature, go to Column suggestions by Copilot.
6. Select **Create**.

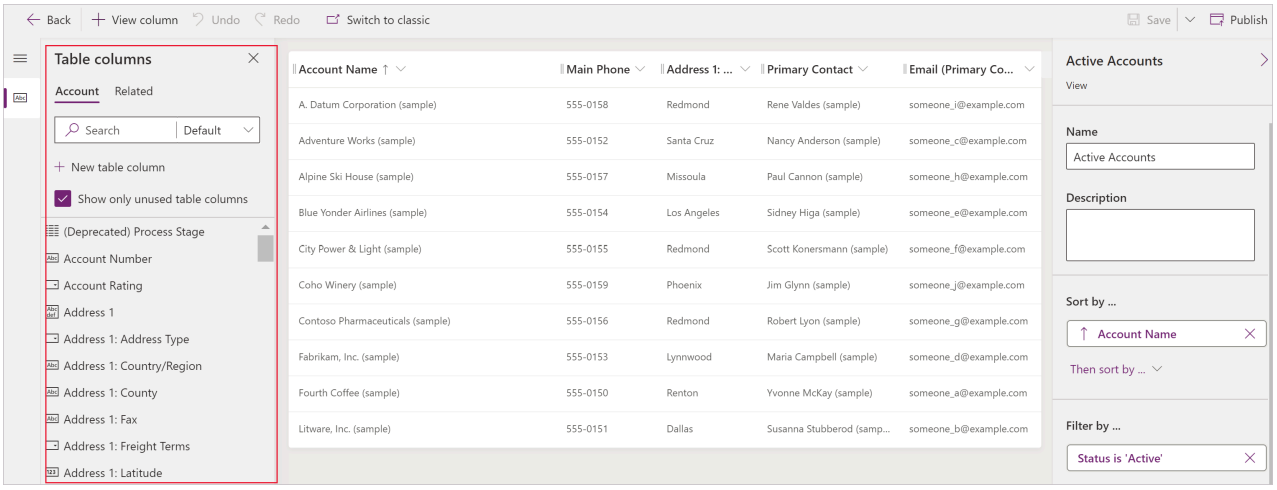
> **Note**
>
> Public views created in Power Apps that include **Contains data** or **Does not contain data** filters don't appear in the list of saved views in Advanced Find.

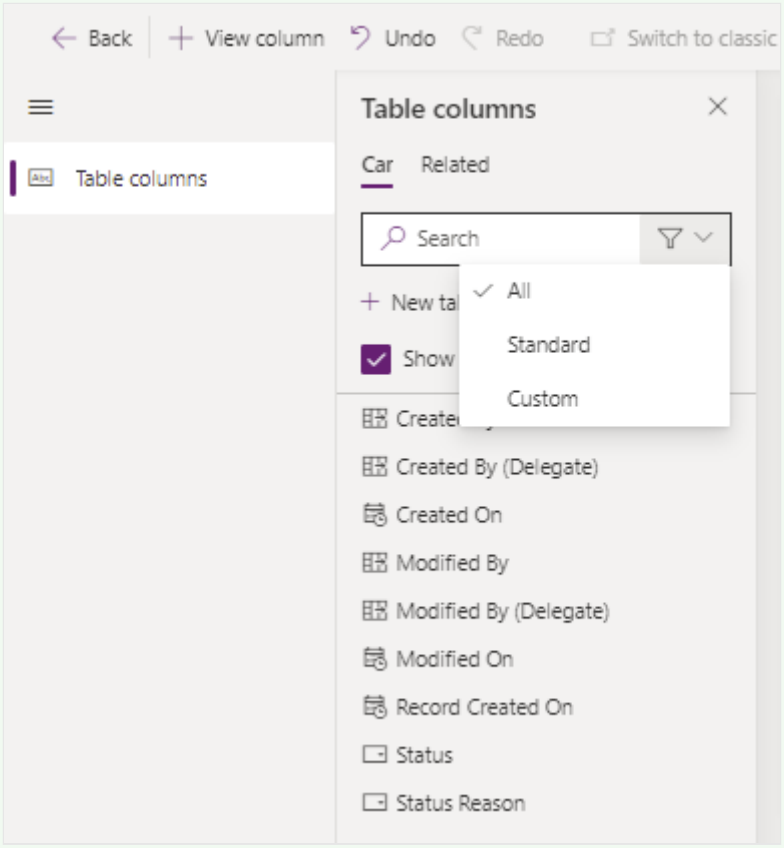
Edit public views from a table

In the solution that you opened earlier, create a new table or find an existing table where the public views need to be edited.

1. Sign in to Power Apps and select an environment
2. On the left navigation pane, select **Solutions**, and open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open a table, such as the account table, and then select the **Views** area, and then open the view you that want to edit.
4. In the view designer, select **+ View column** to add additional columns needed within the view. Or, in the **Table columns** left navigation pane drag the table columns onto your view.

   > **Tip**
   >
   > Only include the specific columns you need for the view, as larger numbers of columns in a view can cause degradation of grid performance.



> **Tip**
>
> The **Default** view in the **Add column** pane displays all columns. If preferred, select **Standard** or **Custom** to display a subset of columns.
>
> 
>

5. In the view designer, the following tasks can be performed:

   - To change the column filtering select the header of the column where the filter is required, and then in the dropdown list select **Filter by**.
   - To change the column sorting select the header of the column where sorting is needed then select **Sort A-Z**, **Sort Z-A**, **Sort descending**, or **Sort ascending**.
     - You can configure multi-column sorts by holding Shift when selecting a sort option to add a sort condition to the set of the existing sorts.

- Configure column width by selecting and dragging the column to the desired position.
- Reorder columns by dragging a column to the desired position.

> **Note**
>
> Depending on the data type of the column the term **Sort A-Z/Sort Z-A** or **Sort ascending/Sort descending** is used

> **Tip**
>
> It is also possible to change column order by selecting the column header and then selecting **Move Right** or **Move Left**.

6. Select **Save and publish** to save the view and make it available for other users in your organization.

Next steps

How to configure view columns

Learn about sorting within views

Learn about filtering views

Make views editable using sub-grids

Last updated on 01/09/2026

Choose and configure columns in model-driven app views

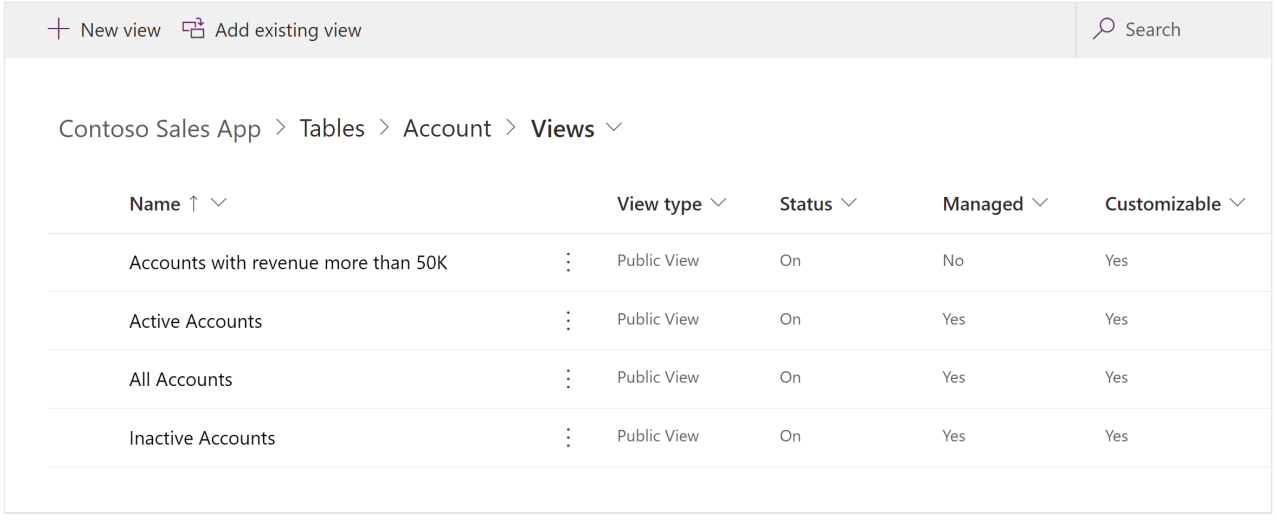
Summarize this article for me

Along with the filter criteria, the columns visible in a Power Apps view are very important to the value provided by the view. In this article, you create or edit views by performing the following tasks:

> **Important**
>
> The latest version of the view designer is currently in preview. Some features, like enabling or disabling presence for a column and adding a find column, aren't yet supported. To accomplish these tasks, open the view in the classic view designer.
>
> - Enable or disable presence for a column
> - Add find columns

Opening the view editor

1. Sign in to Power Apps.
2. Select **Tables**, and then open the table that you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Views**. The list of views for the table is displayed. You can also navigate to the table and views via solutions as shown below.



4. Select an existing view to open it or on the toolbar select **New view**.

Adding columns

You can include columns from the current table or any of the related tables that have a 1:N table relationship with the current table.

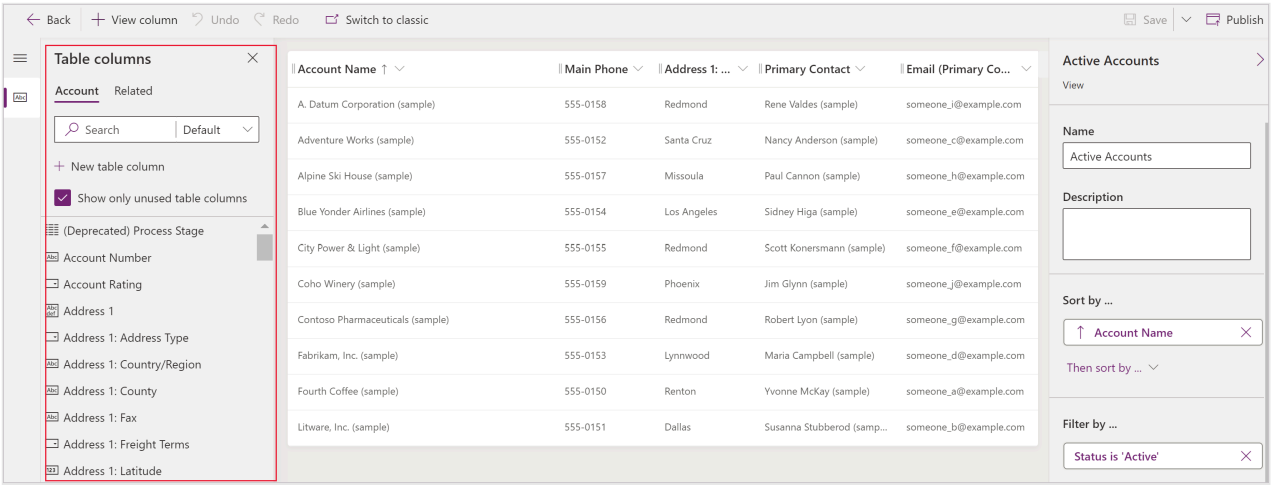
For example, you might want to display the owner of a user-owned table in a column. You can choose the **Owner** column of the current table to display the name of the owner. This appears as a link to open the **User** row for the person who is the owner.

If you want to display the phone number for the owner of the row, you must select **Owning User (User)** from the **Row type** drop-down and then select the **Main Phone** column.

> **Tip**
>
> Only include the specific columns you need for the view. A larger numbers of columns in a view can cause degradation of view and grid performance.

Add columns to views

1. While creating and editing views, ensure that the **Table columns** panel is open. If it isn't, select **View column** on the toolbar.



2. Select the columns you want to add to the view designer. This adds the column as a column on the right-hand of the view.

3. Select the **Related** tab to see related tables and their corresponding columns.

As you add columns, you increase the width of the view. If the width of the view exceeds the space available to show it in the page, horizontal scrollbars appear to allow people to scroll and view the hidden columns.

> **Tip**
>
> If your view filters on data for a certain column so that only rows with a certain value are shown, don't include that column in the view. For example, if you are only showing active rows, don't include the status column in the view. Instead, name the view to indicate that all the rows shown in the view are active.

> **Note**
>
> When you add columns to Lookup views for updated tables, only the first three columns are displayed.

Removing columns

1. Select the header of the column you want to remove.
2. In the dropdown, select **Remove**.

Changing column widths

1. Hover over the area between columns in the view.
2. A line appears and your cursor becomes a double sided arrow.
3. Drag the column to the appropriate width.

Moving a column

Click and drag the column header to the correct position.

> **Tip**
>
> You can also select the header of the column you want to move and from the dropdown select **Move Right** or **Move Left**.

Next steps

Create or edit view filters

Last updated on 02/21/2026

Create or edit filters in model-driven app views

Article • 01/29/2025

The filters in a Power Apps view define the records that a user is able to see when selecting the view for the current table. You can add or edit a filter for the columns that you include in a view by selecting the column and selecting **Filter by** in the view designer.

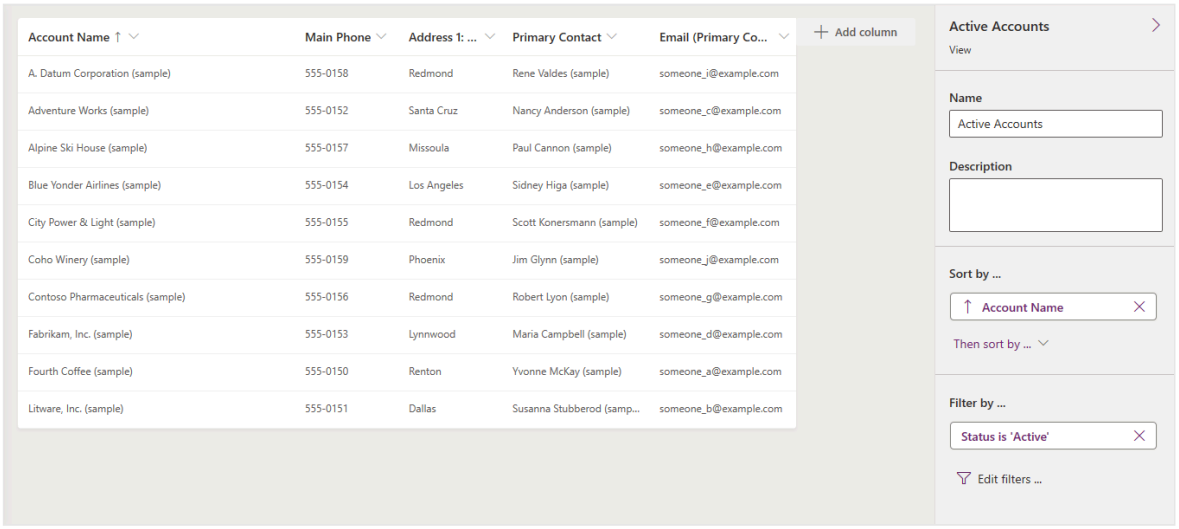
You can also use the expression builder in the view designer. Use the expression builder to add or edit filters for any columns of the table in the current view or any columns in a related table.

In this article, you create or edit filters by performing the following tasks:

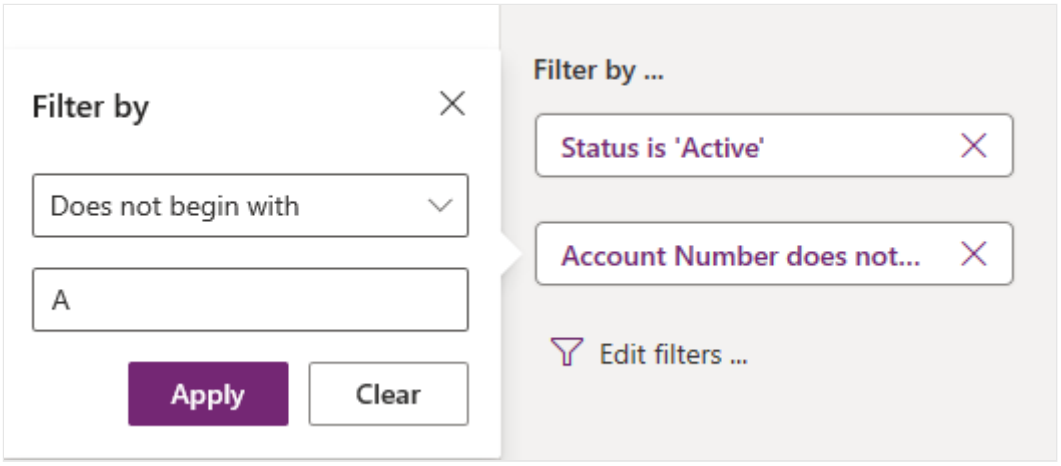
- Edit or remove a filter condition
- Open the expression builder
- Add conditions to a filter
- Add a group condition to a filter
- Add a related table to a condition
- Group conditions of a filter

Edit or remove a filter condition

1. Sign in to Power Apps.
2. On the left pane, select **Solutions**, and then open the solution you want.
3. Open the table that you want, and then select the **Views** area.
4. Open the view that you want to edit. The view properties panel lists existing filters.



5. On the view properties panel, select a filter condition.

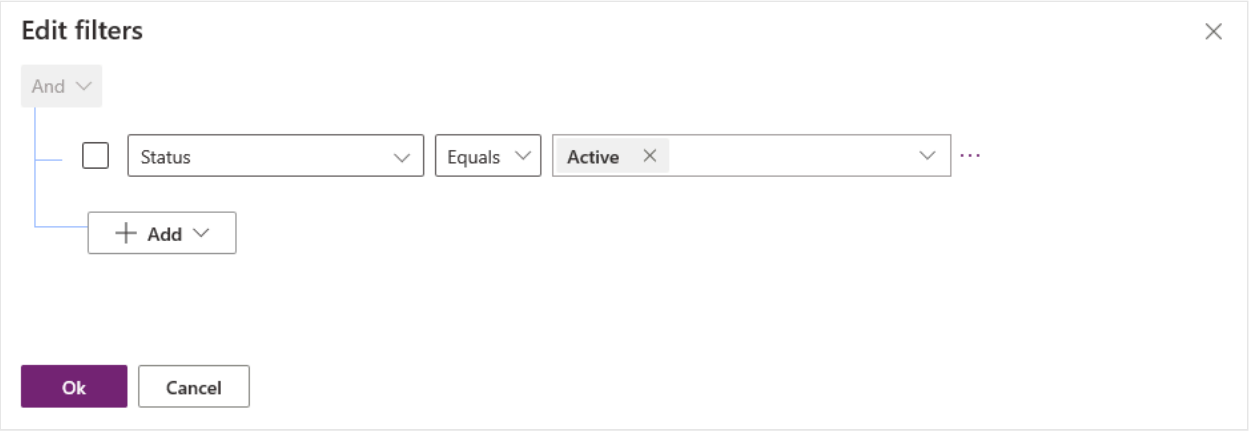


6. Select the conditional operator that you want to use.
7. Type or select the comparison value for the condition.
8. Select **Apply**.
9. To remove a condition, select **Close**. The condition is removed without confirmation.

Open the expression builder

Use the expression builder to redefine existing filters and add additional clauses.

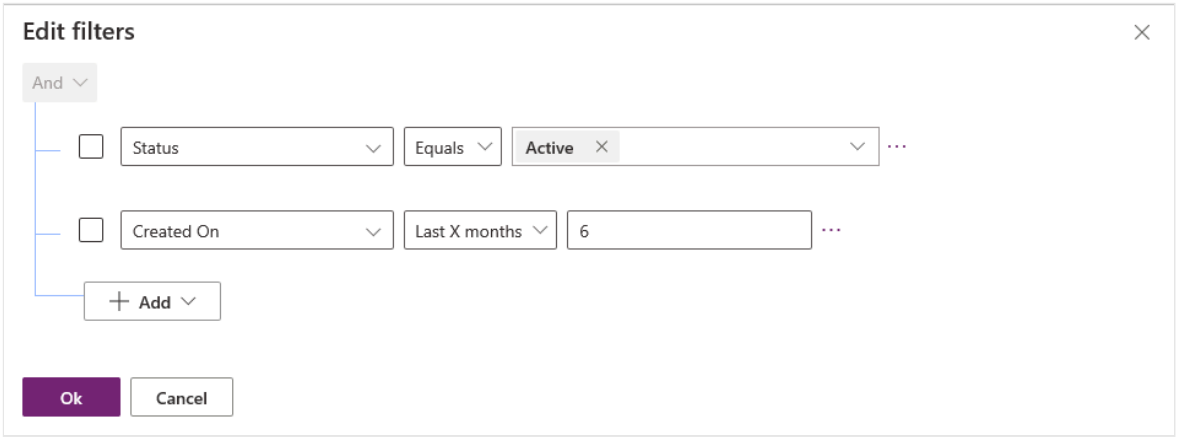
On the view properties right panel in the view designer, select **Edit filters**.



Add conditions to a filter

1. In the expression builder, select **Add** > **Add row**.
2. Select a column for the condition.
3. Select a conditional operator.
4. Select a comparison value.

   Some filter conditions don't require a comparison value for the condition. For example, the operator **Contains data** doesn't require a comparison value. With other filter conditions, you choose the comparison value from an option set. For example, the **Status** column has the values **Active** and **Inactive**.

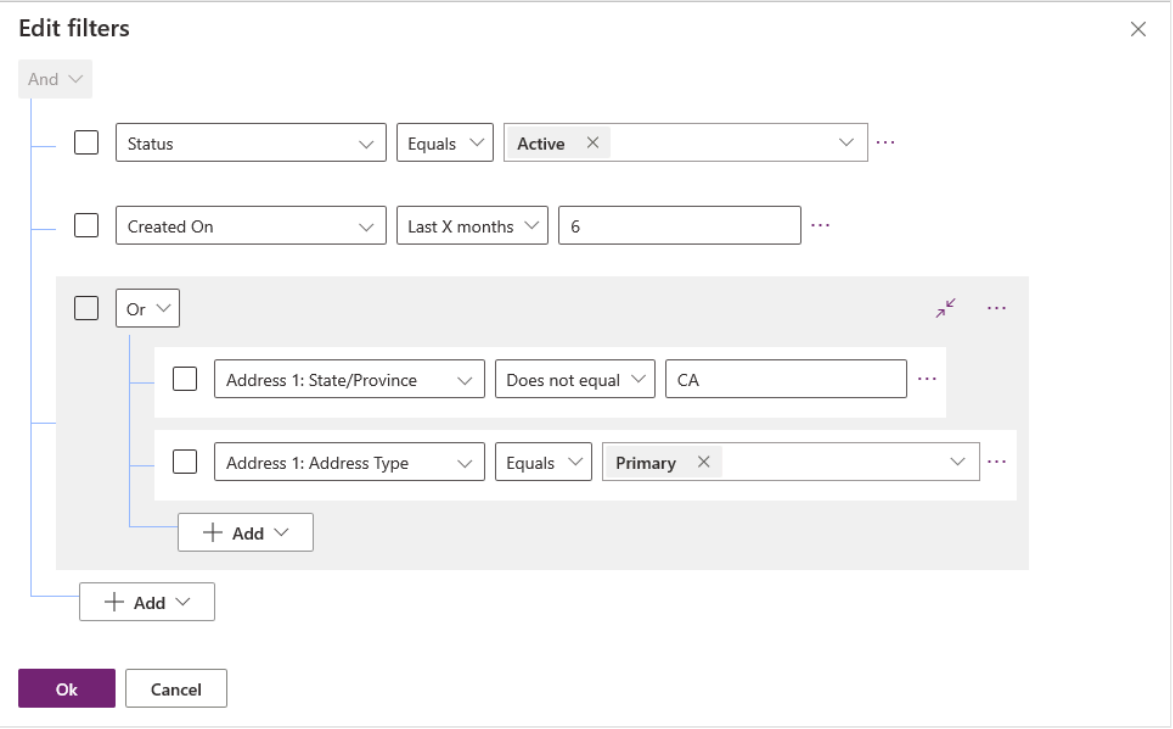
   

5. Select **OK**.

Add a group condition to a filter

1. In the expression builder, select **Add** > **Add group**.
2. Select the relational operator **Or** for the group. **And** is the default relational operator.

3. Specify the first clause of the grouped condition. Select the column, conditional operator, and comparison value.
4. Select **Add** > **Add group**.
5. Specify the second clause of the grouped condition.



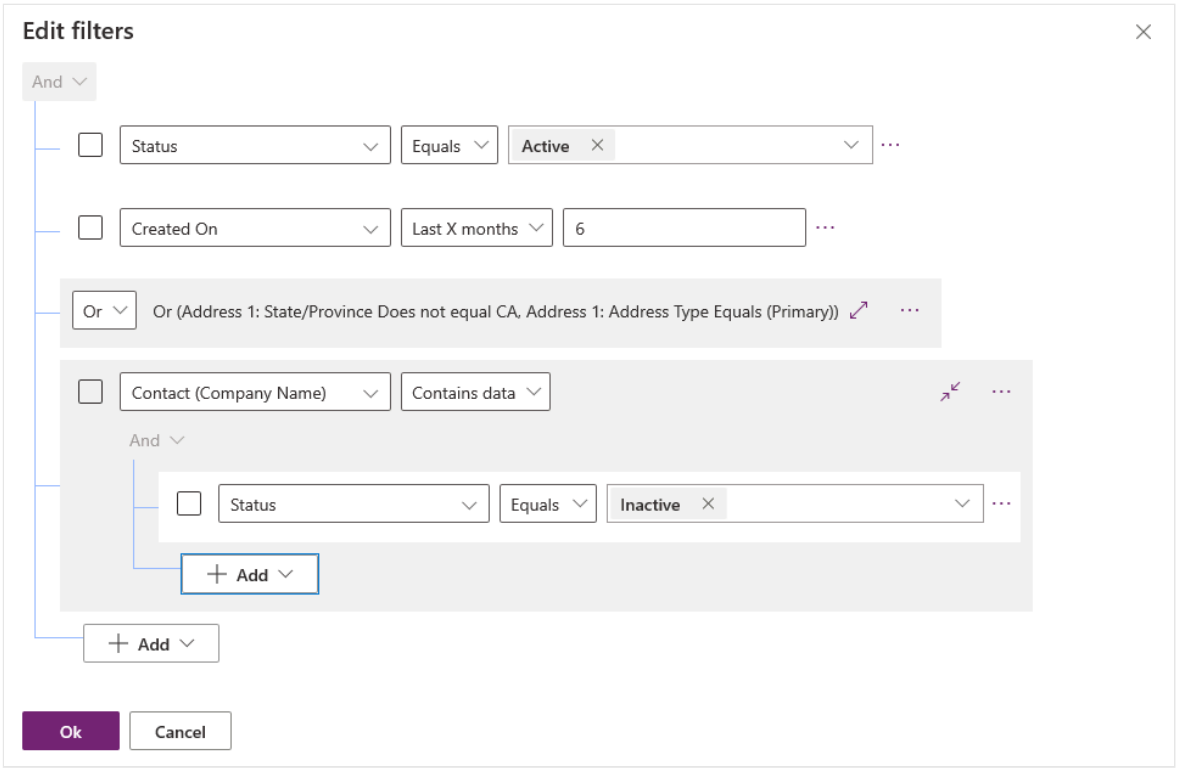
You can select **Collapse** to display the group as a conditional expression.

Add a related table to a condition

1. In the expression builder, select **Add** > **Add related table**.
2. Select a column from the current table that is related to another table. The table related to the column is shown in parenthesis. You can select columns that have a many-to-one, one-to-many, or many-to-many relationship with the related table.
3. Select a column of the related table for the condition.
4. Select a conditional operator.

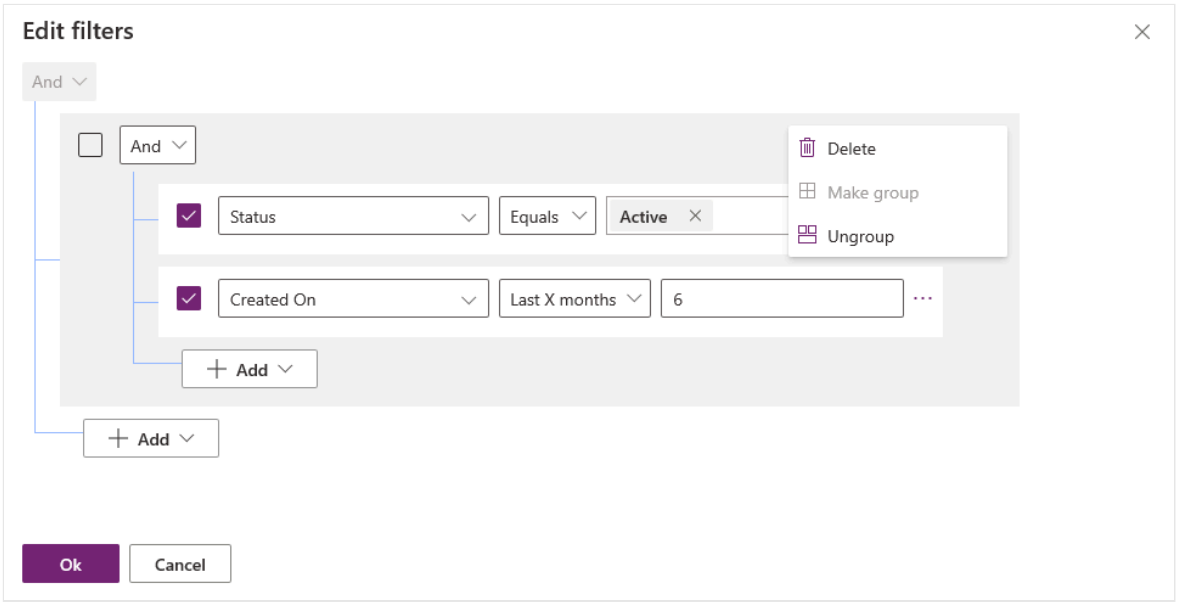
> **Note**
>
> **Contains data** is the only conditional operator that can be used with a related table in a view filter.

5. Select or enter a comparison value.



Group conditions of a filter

1. In the expression builder, select the check box for the conditions that you want to group.
2. Select **More commands** (...) for one of the conditions, and then select **Make group**.
3. To ungroup a group, select **More commands** (...) for the group, and then select **Ungroup**



Next steps

Configure sorting in views
Converting views into editable grids

Feedback

Was this page helpful? Yes No

Provide product feedback

Sort rows in a model-driven app view

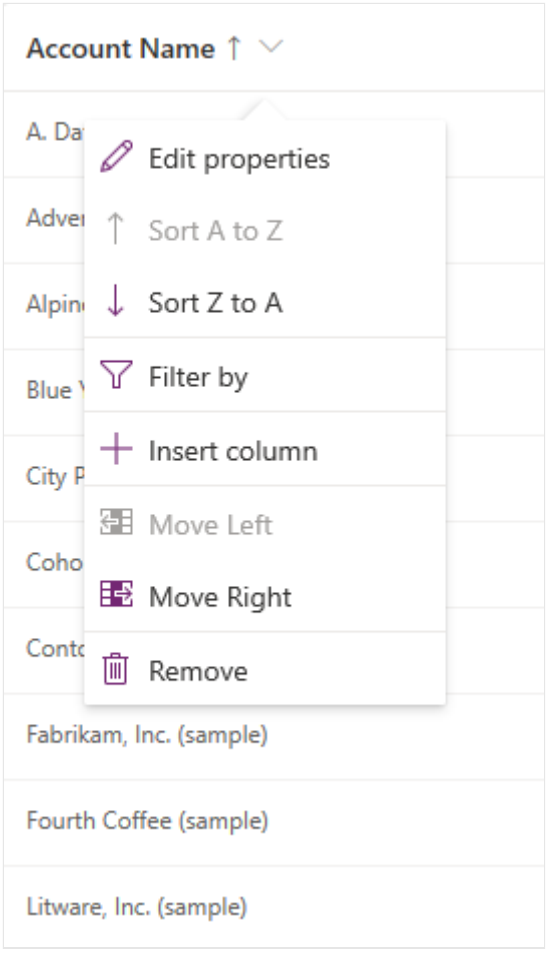
Article • 11/12/2024

When you create or edit a view, you can configure the sort order for either ascending or descending.

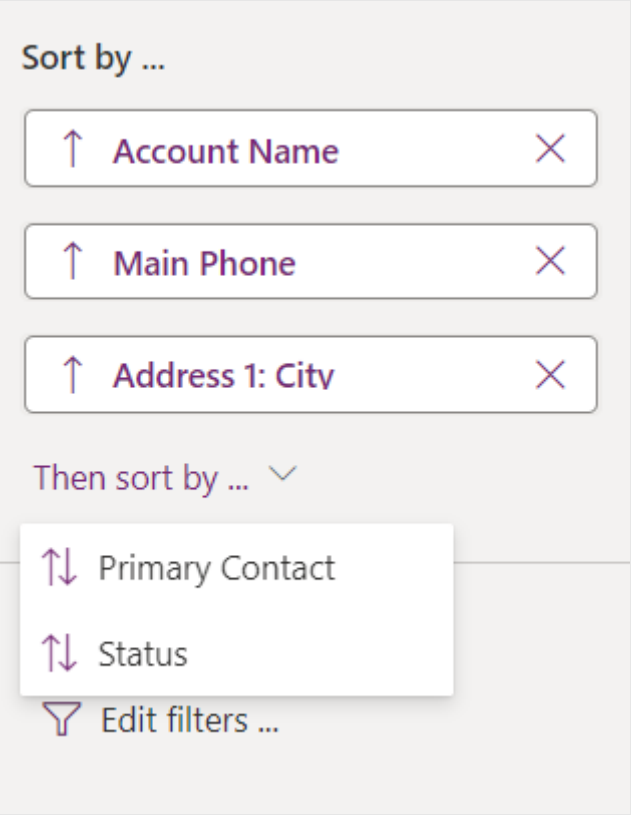
To change the sort order in the view designer, go to Create a public view in Power Apps.

Change the sort order of a view

1. Sign in to Power Apps.
2. Select **Solutions**, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Tables**, open the table you want, and then select the **Views** area.
4. Select a view to open it in the view designer.
5. Select the leftmost column head, such as **Account Name** in the example here, and from the column menu, select **Sort A to Z** or **Sort Z to A**. The sort order is indicated in the column head with an up arrow or a down arrow.



6. If a sort order hasn't been set for the view, select **Sort by**, and then select the primary sort by column.
7. To sort the view by additional columns, select **Then sort by** on the view properties pane on the right, and then select a sort by column for the view.



8. To remove a filter expression, under **Filter by** in the view properties pane, select **X** next to the filter expression you want to remove.
9. Select **Save and publish** to save the view and make it available to apps that include the table.

Next steps

Table views overview

Setting managed properties for views

Feedback

Was this page helpful? Yes No

Provide product feedback

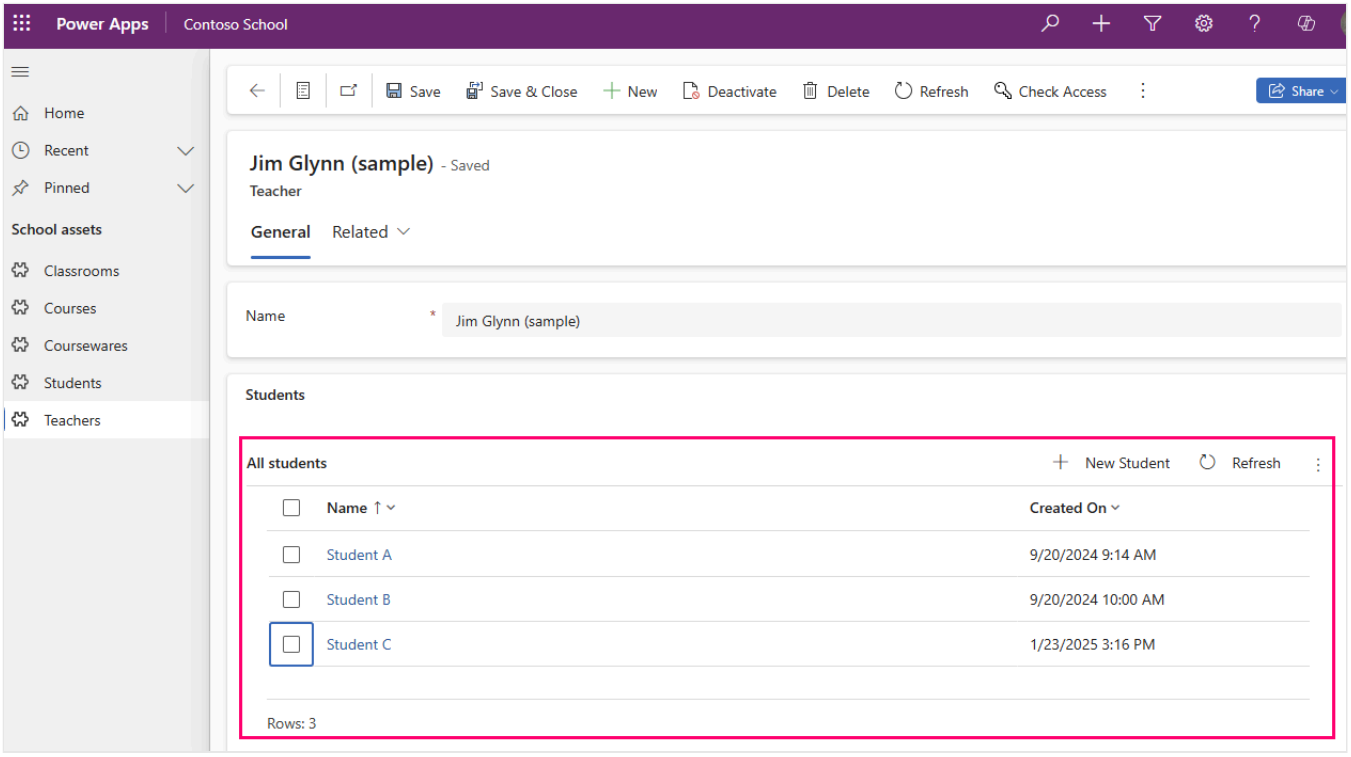
Power Apps read-only grid control

Summarize this article for me

> **Important**
>
> The Read-only Grid control is deprecated. Use the Power Apps grid control instead. More information: Deprecation of Editable Grid and Power Apps Read-Only Grid controls

The Power Apps read-only grid control lets users view and open records from views and subgrids. The control supports the latest Microsoft accessibility standards and is built to be performant and easily extensible for the future. The Power Apps read-only grid control also aligns with the latest Microsoft design guidance for fonts, sizing, and styling.

> **Note**
>
> As of April 2022, the Power Apps read-only grid control is the default control for all read-only views and subgrids.



Add the Power Apps read-only grid as a subgrid on a form

When you add a subgrid to a form, the subgrid only appears on that form and not other forms or views.

More information: Add and configure a subgrid component on a form

Add the Power Apps read-only grid control to views for a table

When you configure the Power Apps read-only grid control for a table, *all* views for that table display using that grid.

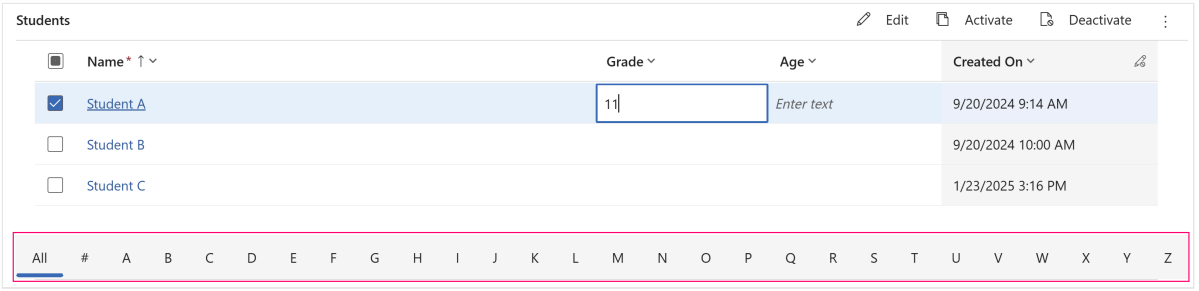
1. Open solution explorer.
2. Expand **Entities**, in the list of tables, open the table you want, select the **Controls** tab, and then select **Add Control**.
3. In the **Add Control** dialog box, select **Power Apps Read-only Grid**, and then select **Add**.

Configure the Power Apps read-only grid control

1. In the **Power Apps Read-Only Grid** row, select the app types you want to apply the grid to. This selection makes the control the default control for the selected clients.

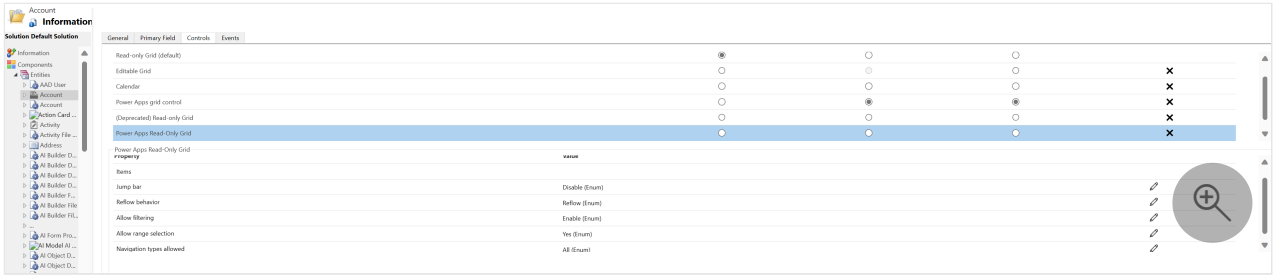
2. Select the pencil icon next to the following properties to make the changes you want:

   - **Jump bar**: By default, the jump bar is disabled. To display an alphabetic list at the bottom of views or subgrids (the jump bar), select the pencil icon in the **Jump bar** row. Under **Bind to static options**, select **Enable** in the dropdown list. Below is a screenshot of the jump bar enabled for the contact table in a model-driven app.



   - **Reflow behavior**: Use this parameter to specify when the grid reflows into a list format or a grid format. Reflowing the control into a list is often better suited for small displays such as a mobile device. The default value is Reflow.
     - **Reflow**: Allows the grid to render into list mode when there's not enough display space.
     - **Grid only**: Displays only as a grid even on smaller displays such as a mobile device.

    - **List only**: Displays only as a list even when there's enough space to display as grid.
- **Allow filtering**: Determines whether filtering options are available to users in the grid column header dropdowns. The default value is Enable.
    - **Enable**: Filtering options are available to users.
    - **Disable**: Filtering options aren't available to users.
- **Allow range selection**: Controls whether users can select a subset of the grid and copy that data to another application like Excel. The default value is Yes.
    - **Yes**: Users can select and copy data from the grid.
    - **No**: Users can't select and copy data from the grid.
- **Navigation types allowed**: Determines which lookup controls in the grid render as hyperlinks. The default value is All.
    - **All**: All table lookup columns render in the grid as a hyperlink.
    - **Primary only**: Suppresses hyperlinks on all lookup fields except the primary column for the selected table.



3. To save your changes, select **Save** on the command bar. When you're ready to make the changes available to app users, select **Publish** on the command bar.

Add the Power Apps read-only grid control to a subgrid

1. Open solution explorer.
2. Open the form that contains the subgrid.
3. Select the subgrid where you want to configure this grid, and then select **Change Properties** on the command bar.
4. In the **Set Properties** dialog box, select the **Controls** tab, select **Add Control**, and then select **Power Apps Read-only Grid**.
5. To continue with configuring and publishing the control, see Configure the Power Apps read-only grid control.

Opt out of the Power Apps read-only grid control

While the Power Apps read-only grid control became the default grid experience in April 2022, you can opt your organization out of the automatic switch to this grid by following the steps here.

1. Go to the Power Platform Admin Center **Manage** > **Environments**. Select **Settings** on the command bar, and then select **Product** > **Features**.
2. Turn off the **Enable the modern read-only grid experience** option in the **Grids and views** section.

> **Note**
>
> As the legacy read-only grid is deprecated, this toggle to opt out of the automatic switch to the Power Apps read-only grid is eventually be removed.

Known issues and limitations

Grid rendering performance

In general, grid performance can be increased by limiting the number of records fetched at once and limiting the number of columns in a view.

If the page size is sufficiently large, the grid might take a long time to render. The current recommendation is to ensure the page size isn't too large, ideally no more than 25 rows.

UI limitations and differences

- The grid doesn't currently support any theme customizations
- Cells with no data are blank instead of displaying three dashes ---.
- The owner column doesn't show online status and a people card for the user.
- Reordering columns from the grid column headers isn't supported.
- If the dataset displayed in the grid contains duplicate rows, the duplicates might not display in the grid. This can lead to the reported record count showing more records than are actually in the grid, or more records appearing when exporting the data to Excel or viewing the data in legacy Advanced Find. This behavior applies to all grid controls, not just the Power Apps read-only grid.

Limitations with card form lists

When a view or subgrid is narrow, it changes into a card list format that is better for small screens, such as mobile devices. The Power Apps read-only grid control displays the following

behavior in a card list:

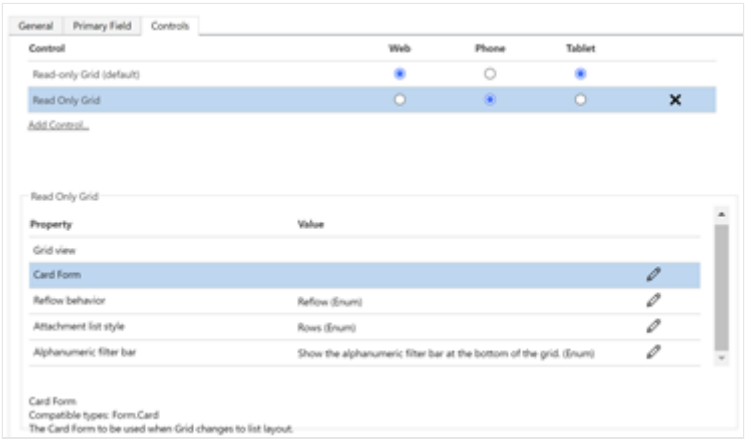
- Sort isn't currently available in a card list view. To work around this, create views with the sort order you want.
- **Select all** and **Clear all** aren't available in a card list view.
- The jump bar isn't available in a card list view. To work around this, select **Search** to filter records. Type the desired letter, and then press Enter.
- Images on a record in the list don't display in a card list view. This includes table icons, web resource images, custom images, and conditional images. The only image displayed for a record in the list are the initials of the record.
- There are no icons for context menu commands in a card list view.
- When there are no records, some lists have a prompt to create a new item. This button isn't displayed in a card list view that uses the Power Apps read-only grid control.

The list view doesn’t display custom cards for the table

To work around this issue, display a custom card by configuring the table to use the card form of the legacy read-only grid control.

1. Go to Power Apps (make.powerapps.com) > **Settings** > **Advanced Settings** > **Settings** > **Customizations** > **Customize the System** > expand **Entities** > select the entity you want to customize > **Controls** tab > **Add Control**.

2. Select **Read Only Grid**, and then select **Add**. Choose to display that grid on one or more client form factors.



3. Select **Configure property** (pencil icon) for the **Card Form** property, and select the custom card form from the drop-down list.

4. Select **OK**, and then **Publish** the customization.

See also

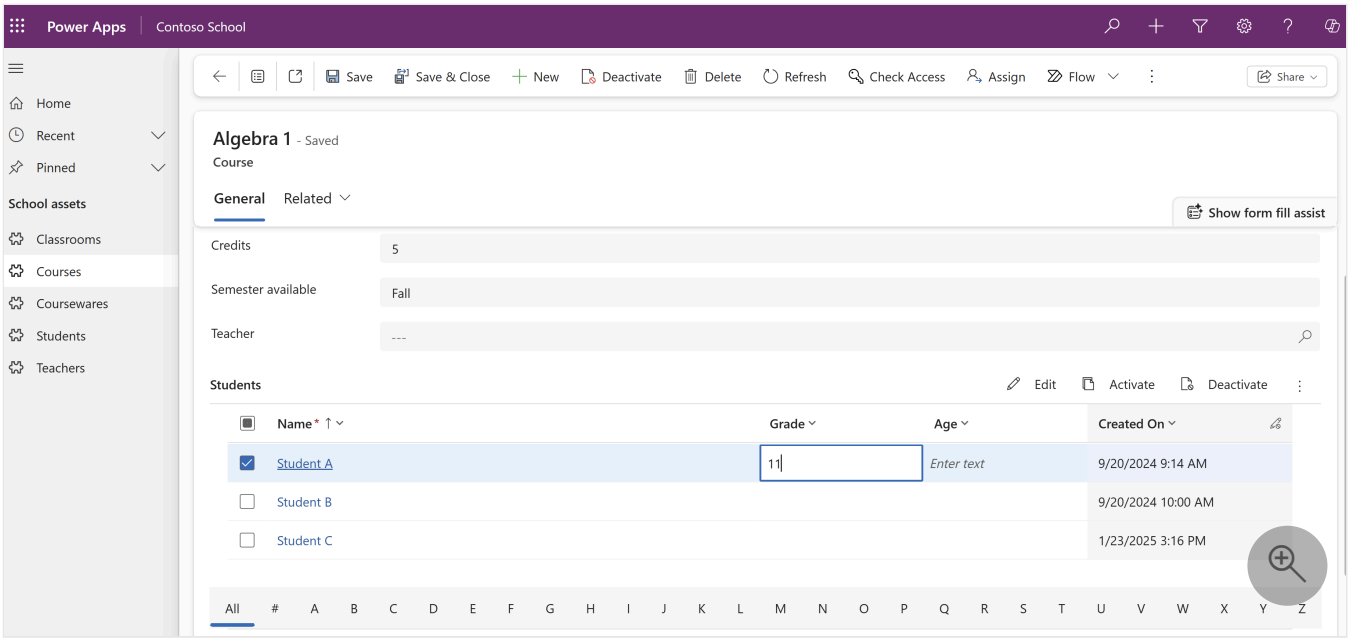
Explore data in a view on a grid page

Last updated on 02/28/2026

Power Apps grid control

The Power Apps grid control represents the next evolution of the classic Power Apps read-only grid control, allowing users to view, open, and edit records from views and subgrids. In addition to inline editing, the control provides a number of other powerful capabilities including infinite scrolling, nested grids, grouping, aggregation, and is customizable. Like the read-only grid control, this control supports the latest Microsoft accessibility standards and aligns with the latest Microsoft design guidance. This control eventually replaces all read-only and editable grids in model-driven apps.

- **Infinite scroll**: A modern data browsing experience in which users can scroll indefinitely through data until they find the records they're interested in. Since there are no page boundaries, users can select more rows at once (up to 1,000 rows) to facilitate taking action against a set of records or perform bulk editing.
- **Inline editing**: This singular grid control can be utilized for read-only or edit scenarios.
- **Nested grids**: Makers can configure the grid to show surface related records. When configured, users can expand a row to see a subgrid of related records. Multiple rows in the grid can be expanded simultaneously. This functionality is an improvement over older grid experiences.
- **Grouping**: Users can organize tabular data in interesting ways by grouping the data based on a single column. Each group is collapsed initially, but users can expand and collapse the groups as desired. Grouping is exposed on most data types with the notable exception of lookup columns. Grouping isn't supported when paging on the grid is enabled.
- **Aggregation**: Users can surface meaningful insights from their data by enabling aggregation on one or more numeric columns in a grid. For each column, users can choose to see the total, minimum value, maximum value, or average of values from that column. If grouping is applied, this aggregate value is also displayed for each group.
- **Customizable**: Makers have the ability to modify various experiences inside the grid component.

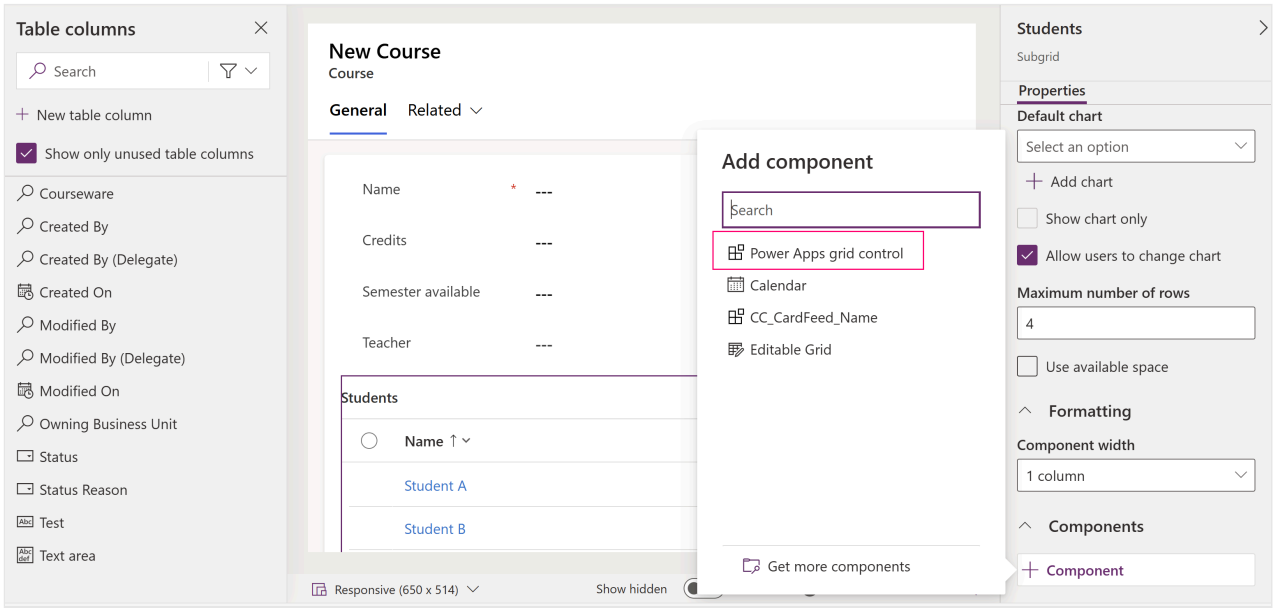


Add the Power Apps grid control using form designer

1. Sign in to Power Apps, select **Solutions** on the left navigation pane, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Select the **Table** within the solution, and then select the **Forms** area. Open the **Main Form** that contains the subgrid for editing.

3. In the form designer, select **Components** on the left navigation pane, expand **Grid**, and then select **Power Apps grid control**. If the component isn't listed, select the subgrid on the form canvas, expand **Components** on the right property pane, select **+ Component**, and then select **Power Apps grid control** from the **Add component** page.



4. Configure the Power Apps grid control

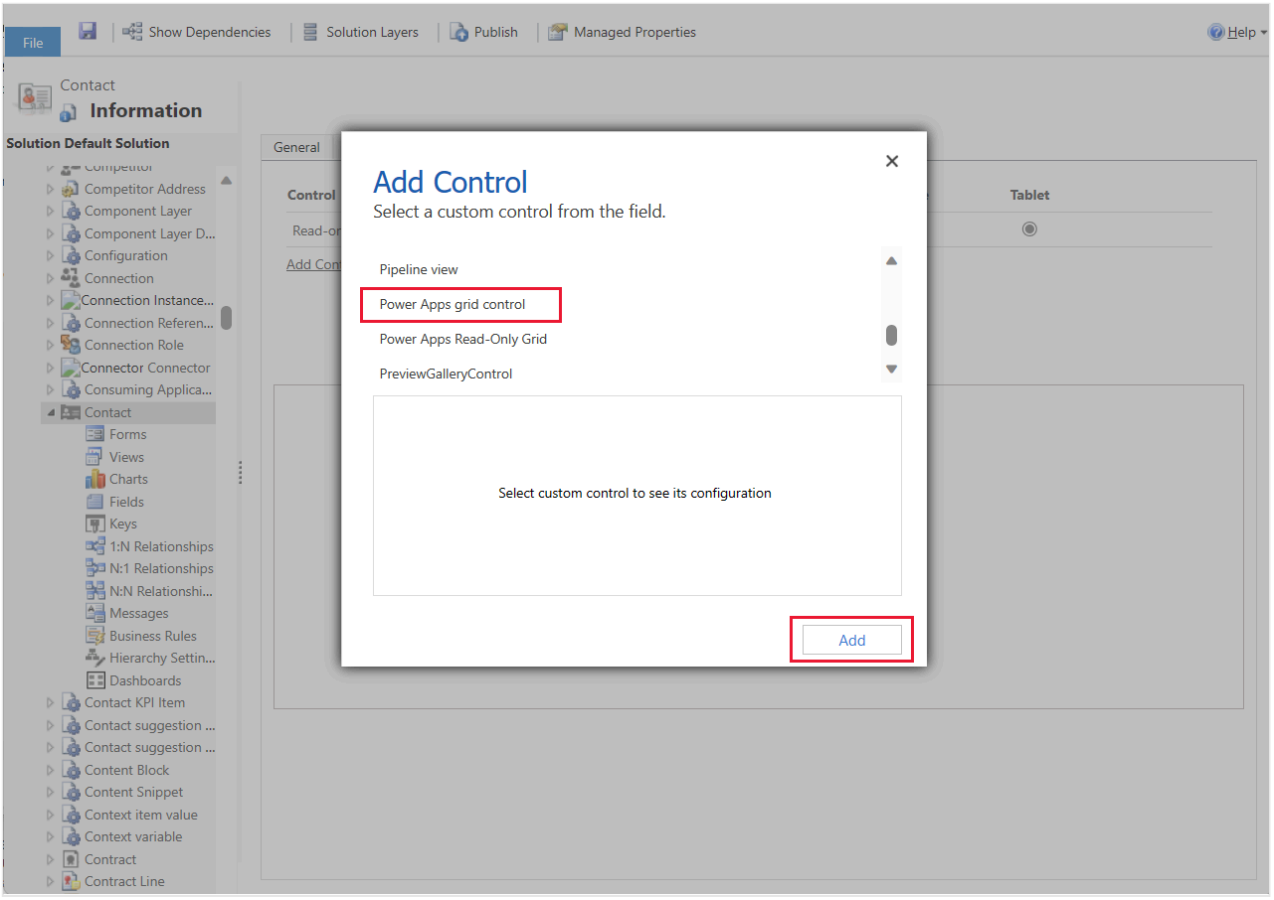
5. Select **Done**.

6. **Save and publish** the form to save and make it available to app users.

Add the Power Apps grid control to views using classic solution explorer

When you configure the Power Apps grid control for a table, all views for that table display using that grid.

1. Open the solution explorer.
2. Expand **Entities**. Browse to the table you want and select it. On the **Controls** tab, select **Add control**.
3. In the **Add control** dialog box, select **Power Apps grid control**, and then select **Add**.



4. Configure the Power Apps grid control
5. After configuring the Power Apps grid control to meet your needs, select **Save** on the command bar to save your changes. When you're ready to make the changes available to app users, select **Publish** on the command bar.

Add the Power Apps grid control to a subgrid using classic solution explorer

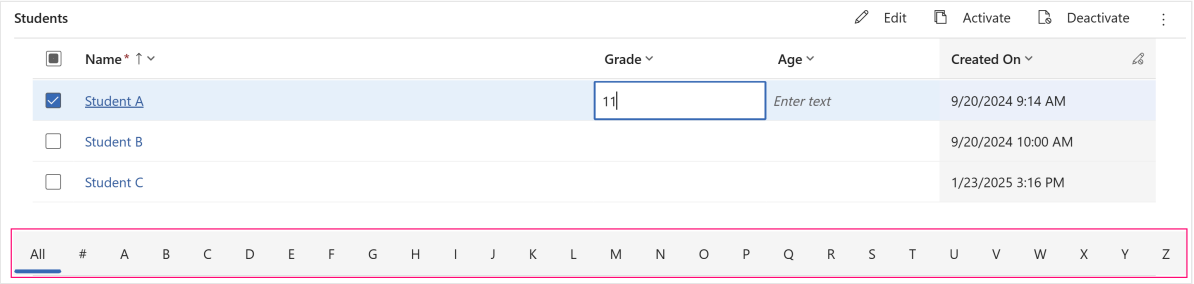
1. Open the solution explorer.
2. Expand **Entities**, expand the table you want, and then select **Forms**.
3. Open the form where you want to add the Power Apps grid control.
4. On the form canvas, select the subgrid where you want to configure this grid (usually located on a form section), and then select **Change properties** on the command bar.
5. In the **Set properties** dialog box, select the **Controls** tab, select **Add control**, and then select **Power Apps grid control**.
6. To configure the control, go to Configure the Power Apps grid control.
7. Select **OK** to close the **Set properties** dialog box.
8. To save and make available to apps, select **Save**, and then select **Publish**.
9. After configuring the Power Apps grid control to meet your needs, select **Save** on the command bar to save your changes. When you're ready to make the changes available to app users, select **Publish** on the command bar.

Configure the Power Apps grid control

1. Select the **Power Apps grid control** components. More information: Configure components on a form
2. There are several properties included with the control to let you tailor the grid experience for that table. To modify any of these properties, select **Edit** in the corresponding row, and then change the value using the dropdown list under **Bind to static options**.

    - The **Enable editing** property determines whether the grid is read-only or editable. The default value is **No**. Select **Yes** to make the grid editable. Editable grids have subtle visual differences to read-only grids including boolean columns showing toggle switches and dropdown and date fields displaying chevrons and date picker icons on hover or focus.
    - The **Enable filtering** property determines if filtering options are available to users in the grid column header dropdowns. The default value is **Yes**.
    - The **Enable sorting** property determines if sorting options are available to users in the grid column header dropdowns. The default value is **Yes**.
    - The **Enable grouping** property determines if grouping options are available to users in the grid column header dropdowns. The default value is **No**.

- The **Enable aggregation** property determines if aggregation options (sum, minimum, maximum, average) are available to users in numeric columns of the grid. The default value is **No**.
- The **Allow column reordering** property controls whether users can reorder columns from directly within the grid. This control includes drag-and-drop of the column headers or using **Move left** and **Move right** actions from the column header dropdowns. The default value is **No**.
- The **Enable multiselect** property determines whether users can select multiple rows in the grid simultaneously or just a single row. The default value is **Yes**.
- The **Allow range selection** property controls whether users can select a subset of the grid and copy that data to another application like Excel. The default value is **Yes**.
- The **Enable jump bar** property can be used to display an alphabetic list at the bottom of views or subgrids. The default value is **No**. Below is a screenshot of the jump bar enabled for a subgrid in a model-driven app.



- The **Enable pagination** property can be used to decide between modern data browsing (infinite scroll) and paging buttons. The default value is **No**. Select **Yes** to disable infinite scrolling and surface paging buttons. Note that the **Select all** action isn't currently available when using infinite scroll, but users can still perform range selection.
- The **Enable OptionSet colors** property can be used to increase the visual appeal of choice columns by showing each value with its configured background color. The default value is **No**. Be sure to verify the configured color for each choice column to ensure readability and accessibility before enabling this property for a table.
- The **Show row status icons** property controls whether users see status icons (invalid, save in progress, saving, and so on) at the beginning of a grid row during the editing experience. The default value is **Yes**.
- The **Show data type icons** property determines whether column headers display an icon corresponding to the data type next to the column name. The default value is **No**.
- The **Navigation types allowed** property determines which lookup controls in the grid render as hyperlinks. The default value is **All**. Select **Primary only** to suppress

hyperlinks on all lookup fields except the primary column for the selected entity, or **None** to suppress all hyperlinks in the grid.

- The **Reflow behavior** property determines whether the grid components render always in a tabular fashion (**Grid only**), always as a list (**List only**), or switches between tabular and list depending on the available width (**Reflow**). The default value is **Reflow**.
- The **Show avatar** property determines if the avatar is shown while the grid is in a list view. This avatar is typically represented as a colored icon with text showing the first letters of the row value. The default value is **Yes**.
- The **Number of columns for list view** property allows you to define the number of columns to render while the grid is in list view. The default number of columns for a list is 3, the maximum number of columns is 10.
- The **Enable banded rows for list view** property applies alternating color band to rows in the list view to improve record differentiation. The default value is **No**.
- The **Show column labels for list view** property displays column labels before values in a list view. The default value is **No**.
- The **Child items** property is used to configure the nested grid capability, which allows users to expand rows to see related records in a subgrid. To set up this property, specify the **Entity** where the related records come from and the **View** that defines the filters, sorts, and columns to show in the subgrid. The default value is no nested grid configured.
    - The **Child items parent ID** property is also needed to configure a nested grid and specifies the column (and relationship) to use to link the parent and child tables together.
    - The **Disable editing in child items grid** property provides some control over the editability of the nested grid. By default, the nested grid inherits the editability of the parent grid, but this property can be used to suppress editing in the nested grid when the parent grid is editable. The default value is **No**.
- The **Customizer control** property allows the maker to link to a single customizer Power Apps component framework control with definitions for changing the visuals or interactions for one or more columns in the grid or the empty grid visual.
- The **Use first column for lookup edits** property allows makers to select the column used for editing lookup cells. When set to **Yes** the cell shows and filters by the first column of the target table's lookup view. When set to **No** the cell shows and filters by the primary column. The default value is **No**.

Customization

For information about how to customize the Power Apps grid control using extensibility APIs, go to Customize the editable grid control.

Known issues and limitations

Limitations with card form lists

When a view or subgrid is narrow, it changes into a card list format that's better for small screens, such as mobile devices. The Power Apps grid control displays the following behavior in a card list:

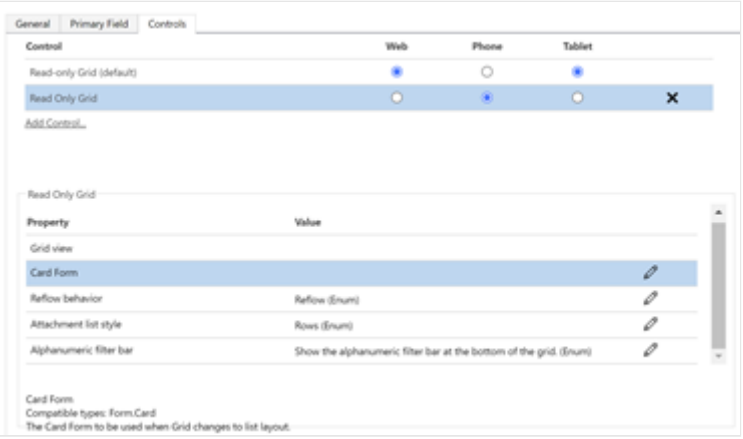
- Sort isn't currently available in a card list view. To work around this limitation, create views with the sort order you want.
- **Select all** and **Clear all** aren't available in a card list view.
- The jump bar isn't available in a card list view. To work around this limitation, select **Search** to filter records. Type the desired letter, and then press Enter.
- Images on a record in the list don't display in a card list view. This limitation includes table icons, web resource images, custom images, and conditional images. The only image displayed for a record in the list are the initials of the record.
- There are no icons for context menu commands in a card list view.
- When there are no records, some lists have a prompt to create a new item. This button isn't displayed in a card list view that uses the Power Apps read-only grid control.

The list view doesn't display custom cards for the table

To work around this, you can display a custom card by configuring the table to use the card form of the legacy read-only grid control.

1. Go to Power Apps (make.powerapps.com) > **Settings** > **Advanced Settings** > **Settings** > **Customizations** > **Customize the System** > expand **Entities** > select the table you want to customize > **Controls** tab > **Add Control**.

2. Select **Read Only Grid**, and then select **Add**. Choose to display that grid on one or more client form factors.



3. Select **Configure property** (pencil icon) for the **Card Form** property, and select the custom card form from the drop-down list.

4. Select **OK**, and then **Publish** the customization.

UI limitations and differences from the read-only grid control

- The grid doesn't currently support legacy theme customizations.
- Cells with no data are blank instead of displaying three dashes ---.
- The owner column doesn't show online status and a people card for the user.
- Columns lookup filters, when using the 'Equals' or 'Does not equal' operator, only show lookup options available within the current set of applied filters.
    - For example, if the contact table has users 'Person A', 'Person B', and 'Person C' but account table data only has rows referencing 'Person B' and 'Person C', then the column filter lookup for the contact column in the account grid only shows the options for 'Person B' and 'Person C'. This is a change in behavior from previous grids.

Known issues

- Column grouping isn't supported when paging on the grid is enabled.
- If the dataset displayed in the grid contains duplicate rows, the duplicates might not display in the grid. This can lead to the reported record count showing more records than are actually in the grid, or more records appearing when exporting the data to Excel or viewing the data in legacy Advanced Find. This behavior applies to all grid controls, not just the Power Apps grid control.
- If a user has read-only privileges to the main list and then navigates through a form to a subgrid representing an entity to which they have write privileges, the subgrid still won't allow the user to make changes as access is determined first by the main entity, then by the related entity.

- When using the Power Apps grid control in editable mode, lookup field cell customization options aren't supported.
- When the Power Apps grid control is configured as read-only at the table level but editable for a subset of views, events might not fire uniformly across the different configuration states.

Related articles

Explore data in a view on a grid page

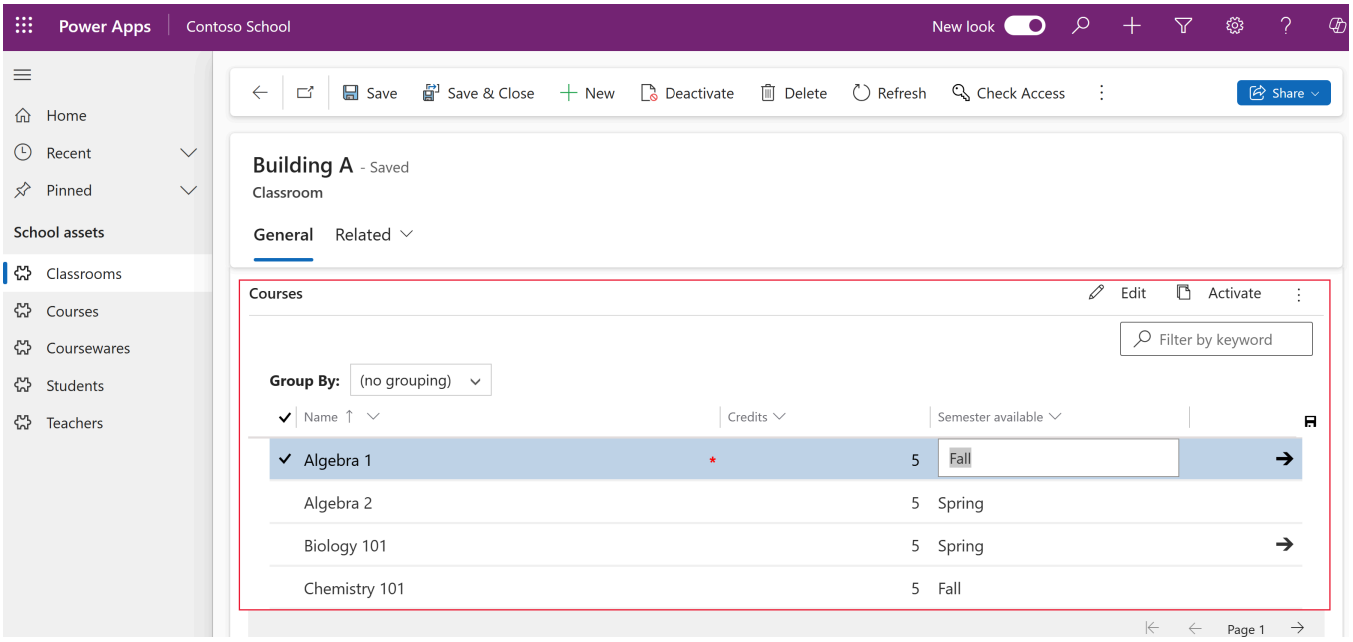
Last updated on 04/07/2026

Make model-driven app views editable using the editable grid control

Summarize this article for me

> **Important**
>
> The Editable Grid control is deprecated. Use the Power Apps grid control instead. More information: Deprecation of Editable Grid and Power Apps Read-Only Grid controls

By default, users can't enter data directly using the read-only view control for views and subgrids on forms. Users select the row in the grid to open a form, edit the data, and then save, which requires multiple steps. With editable grids, users can do rich in-line editing directly from views and subgrids whether they're using a web app or tablet. This editing experience isn't available on phones.



When editable grids are enabled through the editable grid control, users can edit the data inside most types of columns, including basic lookup columns and choices columns. Editable grids avoid the need to open a form.

Add an editable grid to a main form

1. Sign in to Power Apps, select **Solutions** on the left navigation pane, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Select the **Table** within the solution, and then select the **Forms** area. Open the **Main Form** that contains the subgrid for editing.
3. In the form designer, select **Components** on the left navigation pane, expand **Grid**, and then select **Editable Grid**.
4. Select the following for the grid.

Expand table

Area	Property	Description
Grid	Table	Select the table you want that will display records in the grid.
Grid	View	Select the table view you want displayed in the grid.
Grid	Lookup view	Select to add a lookup. Select the lookup column to add (for example, select **Primary Contact**) and in the **Default view** list, select the data source for the lookup column.
Subgrid	Table	If you have a nested grid select the **Table** and **View** for the nested grid (phones and tablets only).
Subgrid	Subgrid parent id	For the subgrid parent id, select the relationship for the tables. For example, the subgrid parent id table column connects the **Account** and **Contact** tables.
Group by	Enable or Disable	If you don't want to allow the user to group data by any column in the view (you want to save space, for example), in the **Group by** value select **Disable**.
Layout	Vertical or Horizontal	Determines how the grid displays on the form.
Allow filtering	Enable or Disable	Disable if you don't want users the ability to filter the grid by keyword.
Hide nested grid column header	Show column header or Hide column header	Determines whether the grid header displays.
Alphanumeric filter bar	Show the alphanumeric filter bar or Hide the alphanumeric filter bar	Determines whether the filter bar located at the bottom of the grid is displayed.
Show component	Web, Mobile, Tablet	Determines the client type that can use the

Area	Property	Description
on		editable grid control.

5. Select **Done**.

6. **Save and publish** the form to save and make it available to app users.

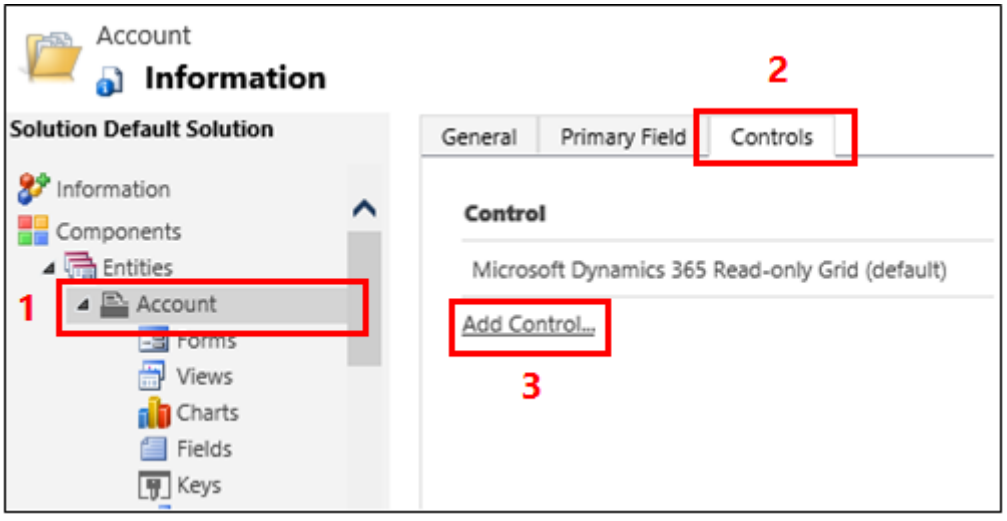
For information about editing the properties of an existing subgrid on a form, go to Configure a subgrid component.

Make main grids editable for views using the classic solution explorer

1. Sign in to Power Apps, select **Solutions** on the left navigation pane, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. On the toolbar select **...** > **Switch to classic**. This opens solution explorer in a new browser window.

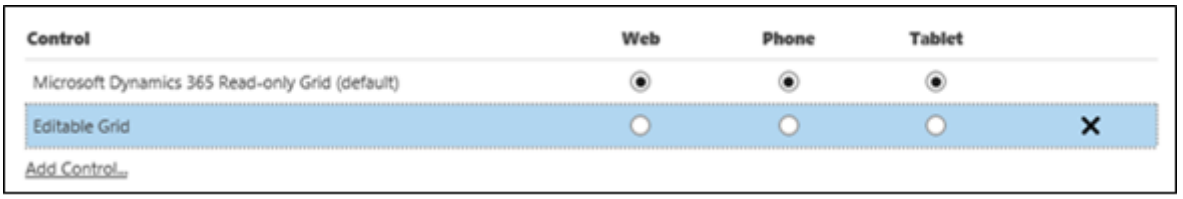
3. In the **Entities** list (these represent Dataverse tables), open the table you want, select the **Controls** tab, and then select **Add Control**.



4. In the **Add Control** dialog box, select **Editable Grid**, and then select **Add**.

5. In the **Editable Grid** row that's added, select the client type you want to apply the grid to, web, phone, and tablet. This makes the editable grid control the default control for the selected client type. Since the control doesn't work with phones, you should select **Web** for desktop app users. At runtime, users are able to toggle between editable grids and read-only grids.

> ⓘ **Note**
>
> Since the editing experience is not available on the **Phone** form factor, if this control is configured for phones, you will see a read-only version of the list control.



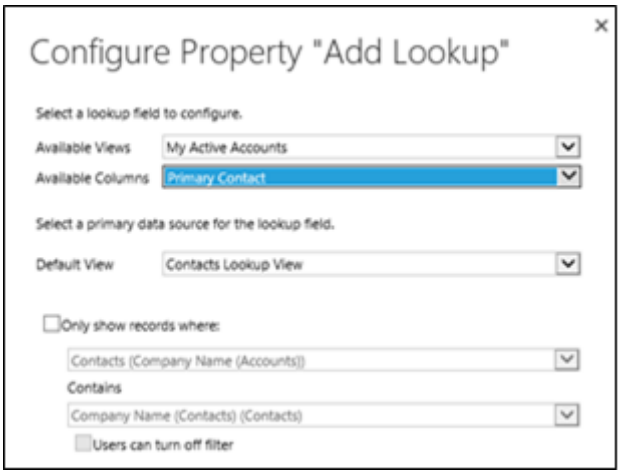
1. To add a lookup, in the **Editable Grid** area, select **Add Lookup**, and then in the **Configure Property "Add Lookup"** dialog box:

   a. In the **Available Views** list, select the view to add the lookup to (for example, select **My Active Accounts**).

   b. In the **Available Columns** list, select the lookup column to add (for example, select **Primary Contact**).

   c. In the **Default View** list, select the data source for the lookup column.

   d. If you want to limit the rows displayed, select the **Only show rows where** check box, and then select your criteria from the list, and then select **OK**.

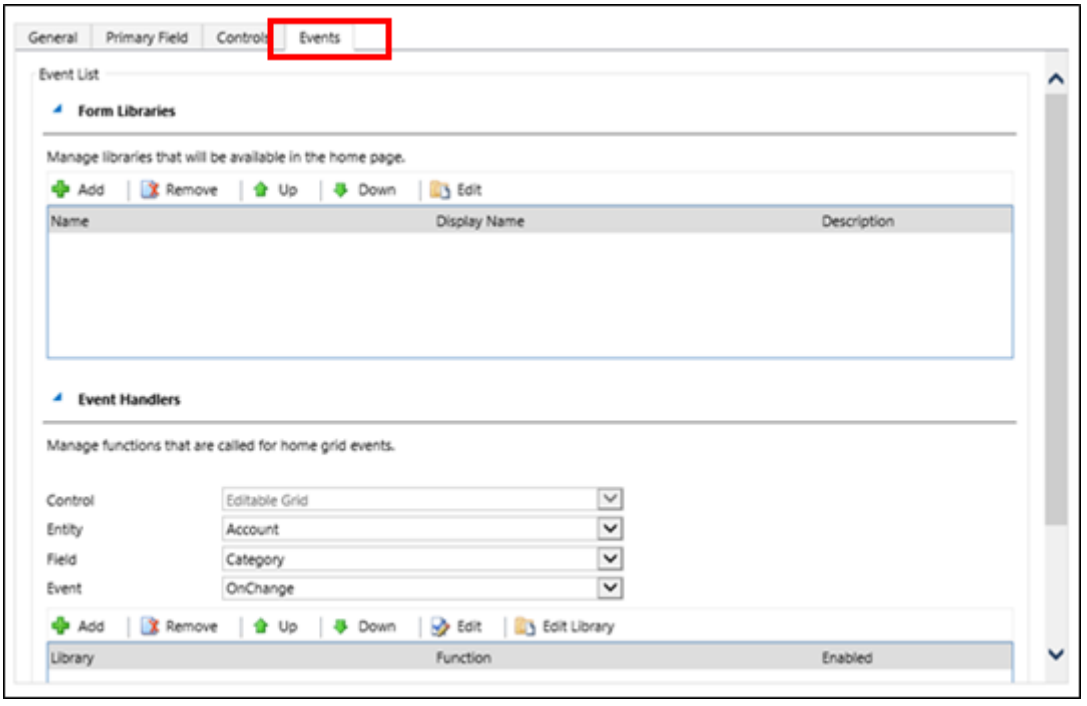
2. If you have a nested grid, select the pencil button for **Nested grid view**, and then select the table and view for the nested grid. For the **Nested grid parent ID** select the relationship for the tables. For example, the **ParentAccountID** column connects the **Account** and **Contact** tables.

   > ⓘ **Note**
   >
   > Nested grids are only available for phones and tablets, not the web.

3. If you don't want to allow the user to group data by any column in the view (you want to save space, for example), in the **Group by Column** row, select the pencil button, and then in the **Configure Property "Group by Column"** dialog box, select **Disabled**, and then select **OK**.

   > **Tip**
   >
   > This is mostly useful for sub-grids on forms.

4. If you want to add JavaScript events, select the **Events** tab, and then select the appropriate tables, columns, and events. More information: Developer Documentation: Use editable grids



5. To save your changes, select **Save** on the action bar.

6. When you're ready to make changes available to your team, select **Publish** on the action bar.

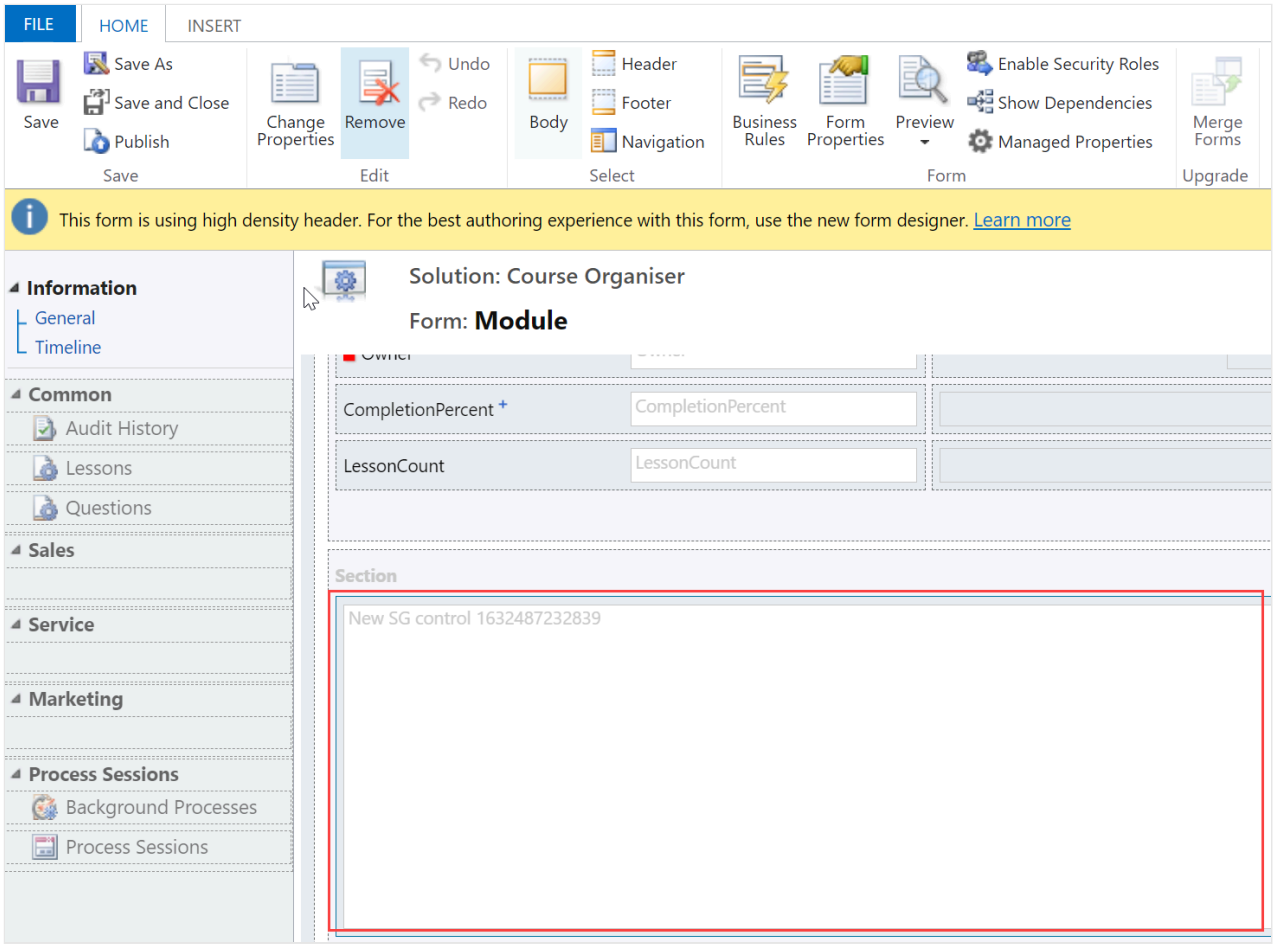
7. To test your changes, go to the view you specified in the previous step, and then make some in-line editing changes.

Make a subgrid on a form editable using classic solution explorer

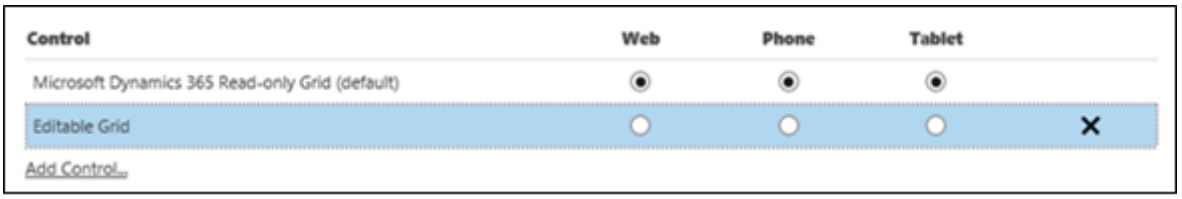
> **Note**

To save an editable grid change within a sub-grid, the user must explicitly save before navigating out of the form.

1. Sign in to Power Apps.
2. Select **Solutions**, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the **Table** within the solution, and then select the **Forms** area. Open the **Main Form** that contains the subgrid.
4. On the form designer command bar, select **...** > **Switch to classic**. This action opens a new tab in your browser.
5. Select the appropriate control, and then select **Change Properties** on the ribbon.



6. In the **Set Properties** dialog box, select **Controls**.
7. Select **Add Control**, select **Editable Grid**, and then select **Add**.
8. In the **Editable Grid** row that's added, select the client types you want to apply the grid to. This makes the editable grid control the default control for the selected form factor. For most instances, select **Web**.



9. Select **Save**, and then select **Publish**.

Editable grids support

- In-line editing of rows at the table or subgrid level (includes custom tables).
- System views and personal views.
- Web and mobile clients.
- Navigation with a keyboard or mouse.
- Grouping and sorting (you can group by/sort rows on the client-side by any column in the current view).
- Filtering.
- Moving and resizing columns.
- Pagination.
- Lookup configuration.
- Calculated columns and rollup columns.
- Business rules (Show error message, Set column value, Set business required, Set default value, Lock, or unlock column).
- JavaScript events.
- Enabling or disabling of cells based on security role.
- Users can continue to use search and charts, and can access the action bar as with read-only grids.

Supported standard tables

Expand table

Web/tablet/phone	Tablet/phone only	Web only
Account	Activity	Campaign
Appointment	Attachment	Campaign Activity
Bookable Resource	Channel Access Profile Rule Item	Campaign Response
Bookable Resource Booking	Competitor Address	Channel Access Profile
Bookable Resource Booking Header	Connection	Channel Access Profile Rule
Bookable Resource Category	Connection Role	Contract
Bookable Resource Category Assn	Email Signature	Entitlement Template
Bookable Resource Characteristic	Email Template	External Party
Bookable Resource Group	Expired Process	Fax
Booking Status	Invoice Product	Letter
Case	Knowledge Article Incident	Marketing List
Category	Lead To Opportunity Sales	Position
Characteristic	Process	Quick Campaign
Competitor	Mailbox	Recurring Appointment
Contact	New Process	Sales Literature
Email	Note	SLA
Entitlement	Opportunity Product	
Feedback	Opportunity Sales Process	
Invoice	Order Product	
Knowledge Article	Organization	
Knowledge Article Views	Phone to Case Process	
Knowledge Base Record	Price List Item	
Lead	Queue Item	
Opportunity	Quote Product	

Web/tablet/phone	Tablet/phone only	Web only
Order	Sharepoint Document	
Phone Call	Translation Process	
Price List		
Product		
Queue		
Quote		
Rating Model		
Rating Value		
SLA KPI Instance		
Social Activity		
Social Profile		
Sync Error		
Task		
Team		
User		

Limitations

Data types that aren't editable in an editable grid

The following data types aren't editable in editable grids: Customer and Partylist Lookup columns; Composite (address) columns; State/Status columns; Lookup table-related columns (for example, the Account table includes a contact lookup, where the Contact column is editable but the EmailAddress(Contact) column isn't editable).

Group by views work on client side only

Grouping behavior works only on the client side and doesn't span pages. Group by is a client only function and works only on one page of data. Group by doesn't show you all options based on your complete data set on the server. Group by shows grouping only on the current page. You can disable the grouping by using the property on custom control configuration. More information: Make a subgrid on a form editable using classic solution explorer

Inline grid modifications aren't persisted

Changes made to the structure of the editable grid from within the grid, such as column resizing, column reordering, grouping, filtering, and sorting, will be reset the next time the user visits the page. These types of changes aren't saved across sessions or within views.

Business rules work only if conditional column is a column on the grid

Business rules on an editable grid are supported only if the conditional column is also a column on the grid. If the column isn't a column the business rules won't work. Verify that each column referenced in the business rule is also included on the form. Note that business rules on an editable grid don't fire if the editable grid is configured on a dashboard.

Editable grids don't work on phones

Based on customer feedback, we have removed the editable grid experience from phones. When using an editable grid on a phone, you see a read-only version of the list control.

Duplicate rows in a dataset might not be displayed in the grid

If the dataset displayed in the grid contains duplicate rows, the duplicates might not display in the grid. This can lead to the reported record count showing more records than are actually in the grid, or more records appearing when exporting the data to Excel or viewing the data in legacy Advanced Find.

Next steps

Setting managed properties for views

Keyboard shortcuts for editable grids (views)

Last updated on 02/27/2026

Model-driven app managed properties for views

Summarize this article for me

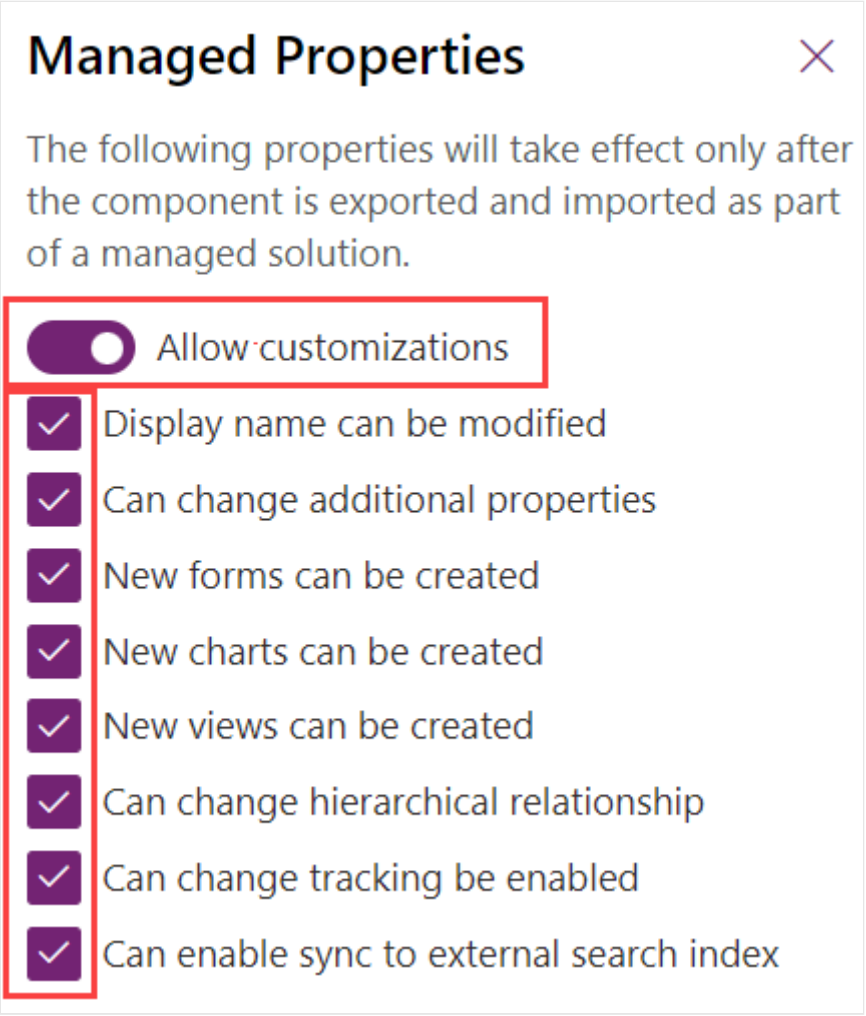
When you create a custom public view for a table in Power Apps that's included in a managed solution, there's the option to limit the ability of anyone who is installing the solution from customizing the view.

Additionally, there are high-level options that you can set on the table that apply to all views.

By default, most views have their **Customizable** managed property set to true so that people can customize them. Unless there's a good reason to change this, we recommend you allow people to customize views in the table.

Set managed properties for all views

1. Sign in to Power Apps.
2. Select **Solutions**, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the table you want. Select **Advanced**, and then select **Managed properties** on the command bar.
4. There are a range of options relevant to the current table to either prevent any customizations or to prevent the creation of new views.



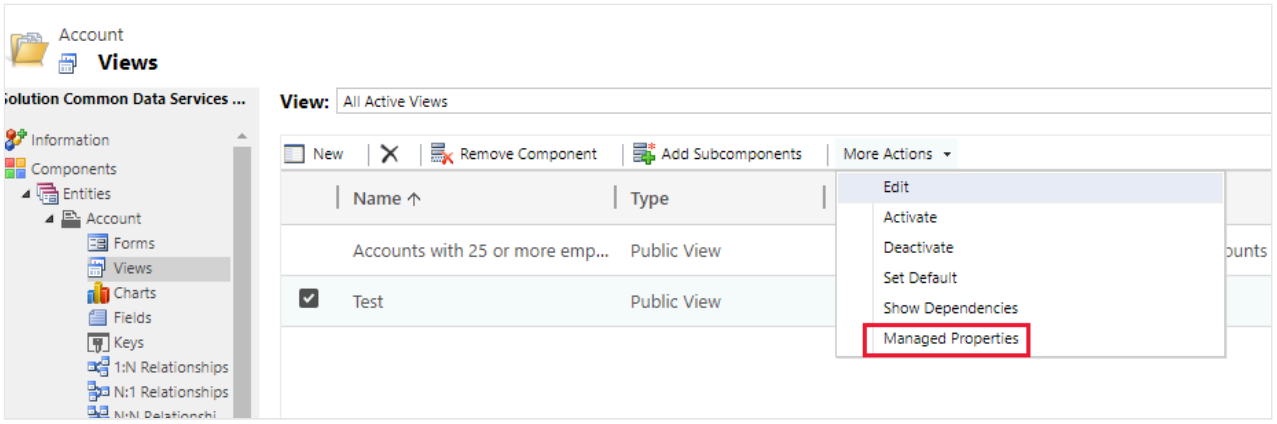
5. Make any necessary changes and when finished, select **Done**.

> **Note**
>
> The setting doesn't take effect until the solution is exported from the development environment and imported as a managed solution into another environment.

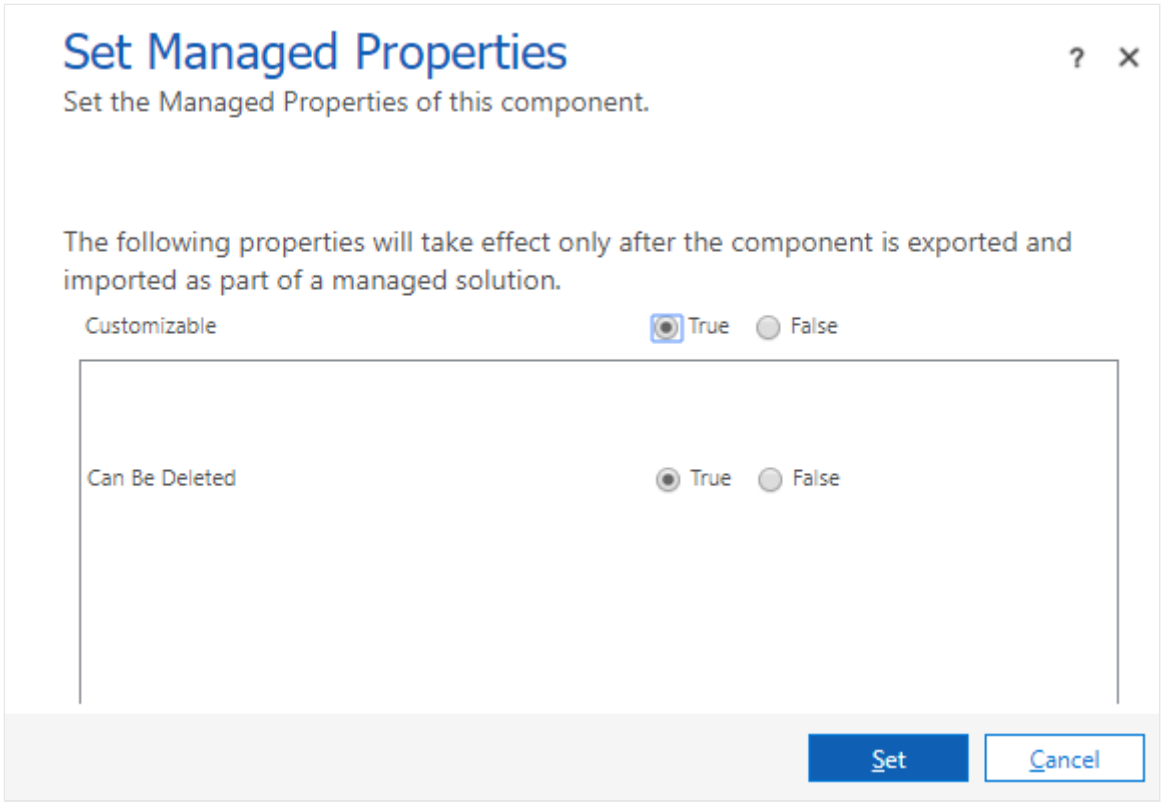
Set managed properties for a view (classic)

With this approach, the ability for the view to be changed can be set at the level of the individual view.

1. Open solution explorer, expand **Entities**, select the table required, and then select **Views**.
2. Select a custom public view.
3. On the command bar, select **More Actions** > **Managed Properties**.



4. Set the **Customizable** or **Can Be Deleted** options to **True** or **False**.



Next steps

[Specifying a default view]

Last updated on 02/13/2026

Specify read only grid properties

Article • 04/19/2023

Model-driven apps use responsive design principles to provide an optimal viewing and interaction experience for any screen size or orientation. As the size of the screen or the container decreases—for example, on phones and smaller viewports—the grid is transformed into a list.

The Read Only Grid control specifies how a grid should respond to different screen sizes. As an app maker working with a model-driven app, you can configure the Read Only Grid control and its properties for custom grids and lists.

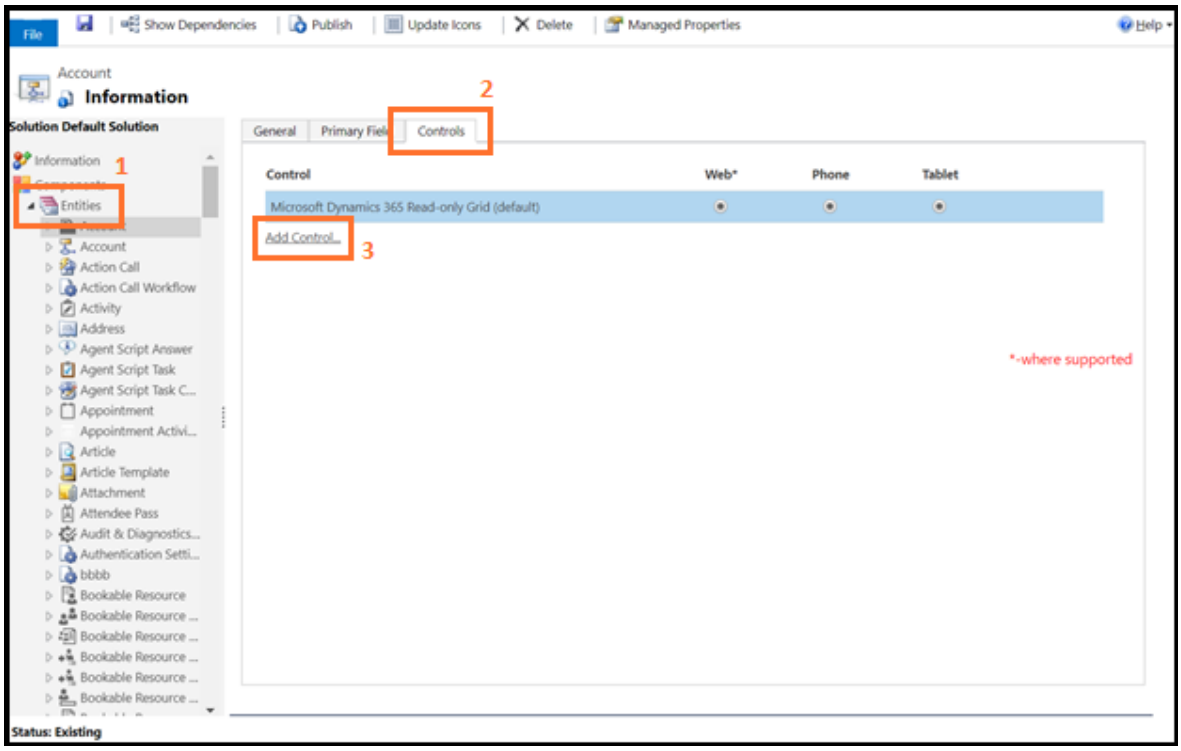
- **Card Form** property: Use a card form for lists instead of the default list template. Card forms provide more information for list items than the default list template.
- **Reflow behavior** property: Use this parameter to specify a grid to reflow in to a list or not.

Allow grid to reflow into list

Configure the following features after you add the Read Only Grid control to a table:

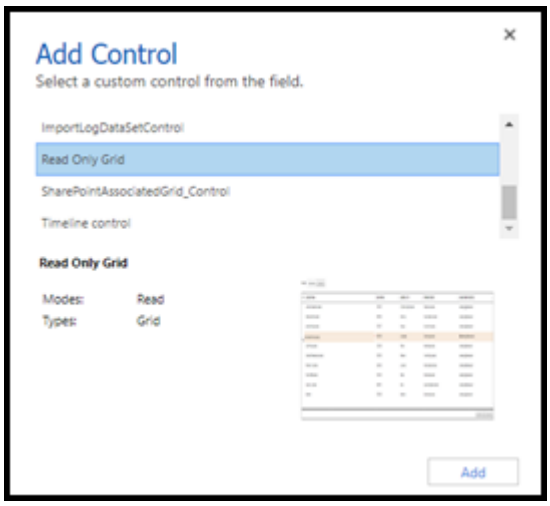
- Allow a grid to reflow into a list on small displays such as mobile.
- Specify the rendering mode as grid-only or list-only.

1. Go to Power Apps
2. Select **Tables** on the left navigation pane. Alternatively, select **Solutions**, and then open the solution. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. Open the table, such as **Account**.
4. Select the **Forms** area, and select the form that you want to edit.
5. Select **Switch to classic**. The classic form editor opens as a new tab in your browser. (You may need to select **...** before selecting **Switch to classic**.)
6. In the navigation pane expand **Entities**, select the table (such as **Account** or **Contact**), and then on the **Controls** tab, select **Add Control**.

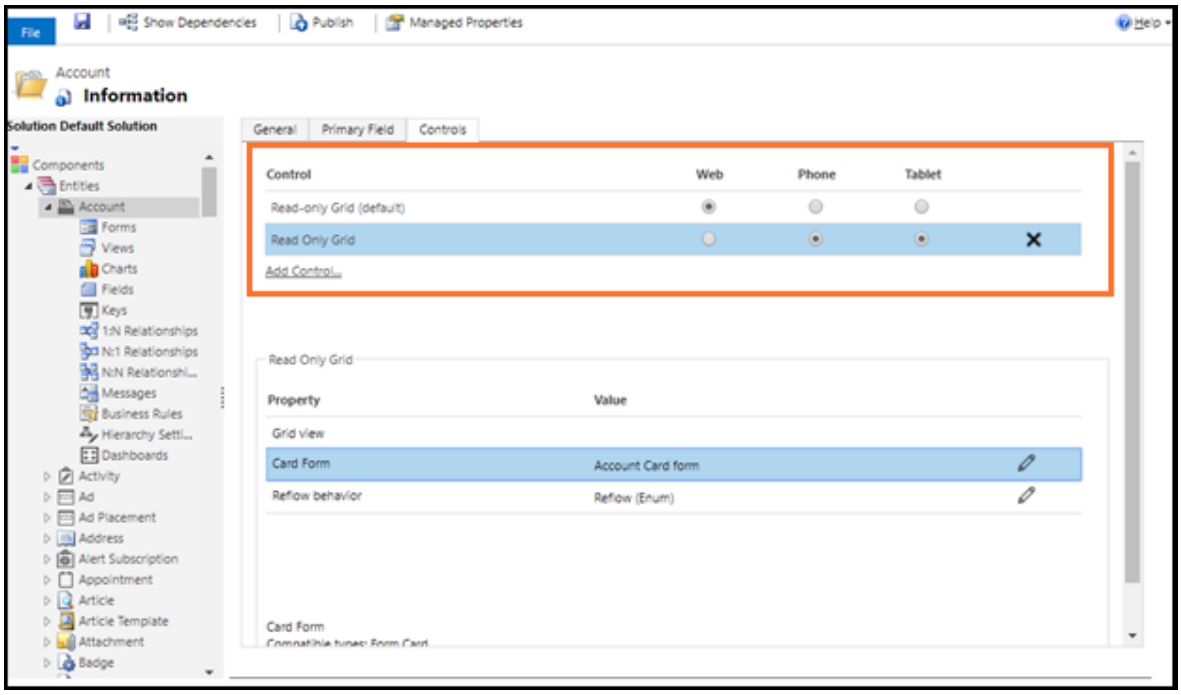


7. Select **Read Only Grid** from the list of controls, and then choose **Add**.

   The control is added to the list of available controls.



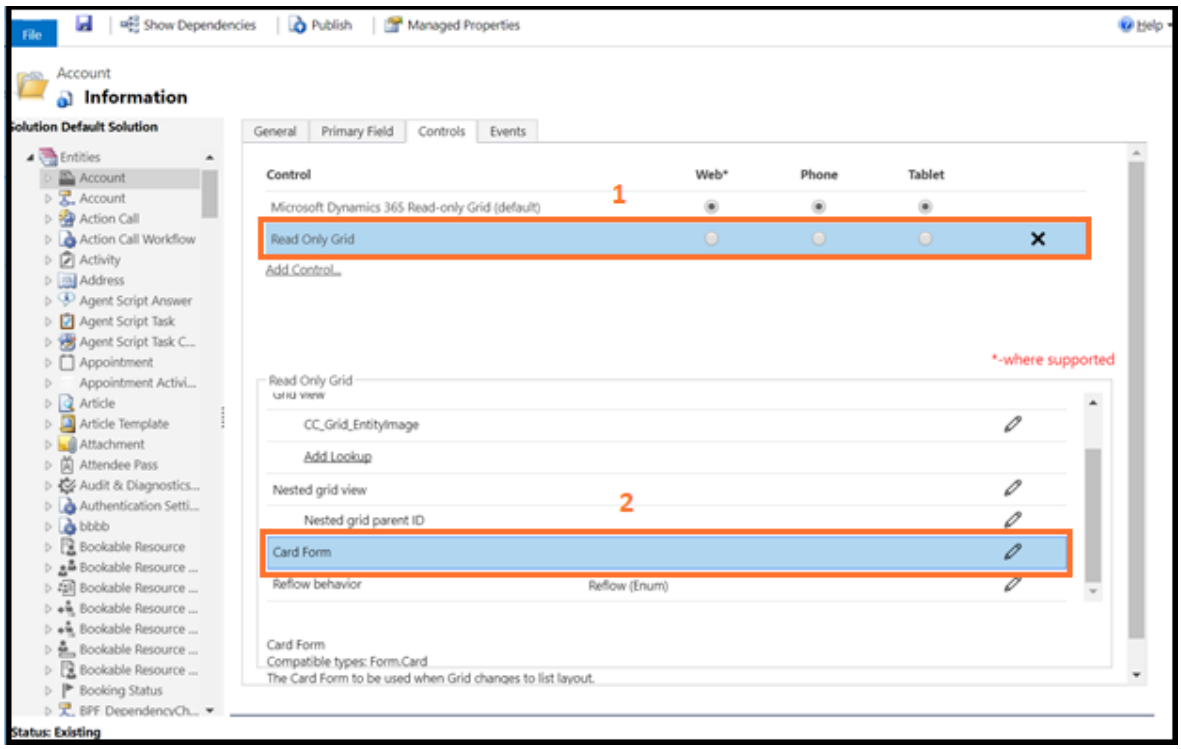
8. Select the devices (**Web**, **Phone**, or **Tablet**) for which you want to make the grid read-only.



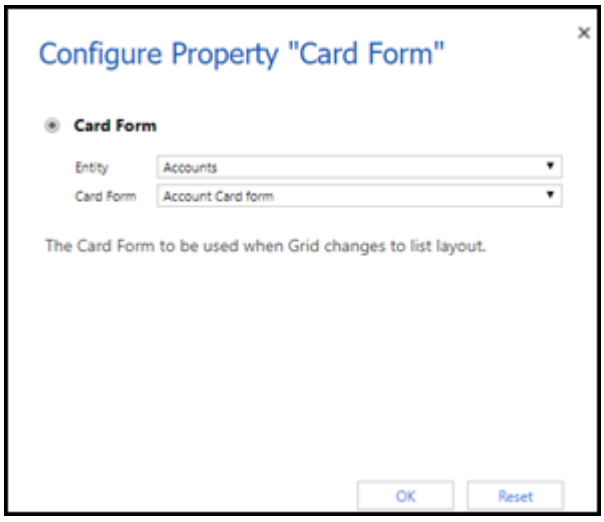
9. Configure the **Card Form** property.

   You can use the Card Form property to show list items instead of the default list template. Card forms provide more information for list items than the default list template does.

   a. Choose the pencil icon next to **Card Form**.



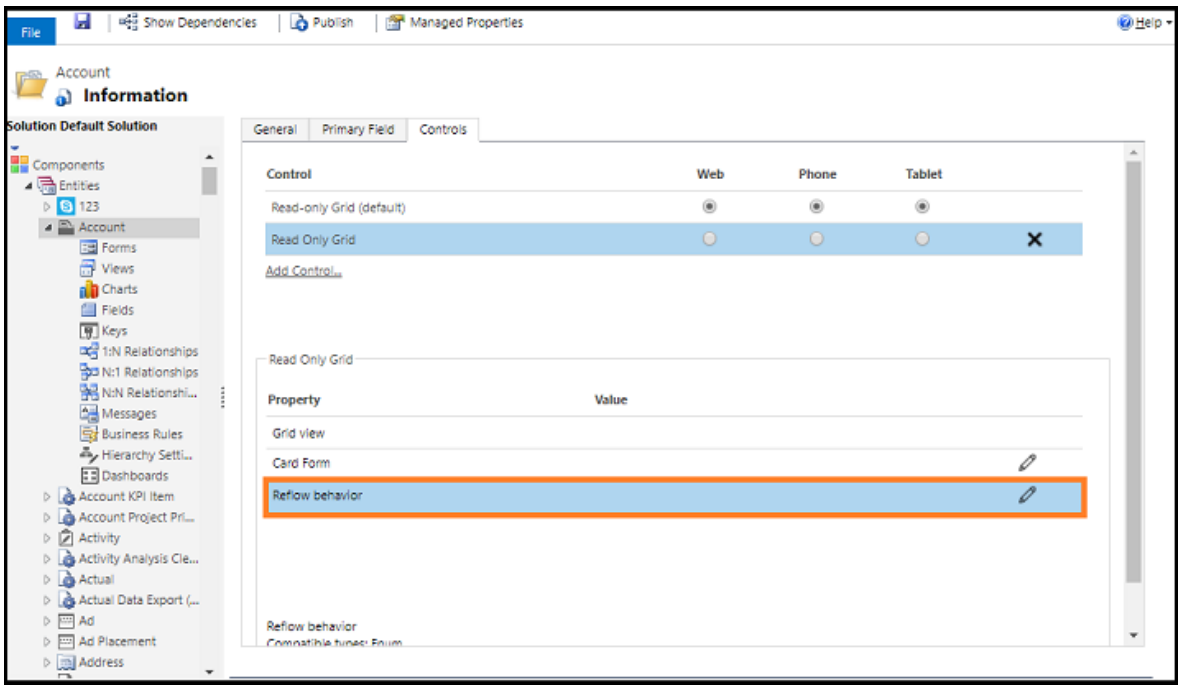
   b. Select the **Entity** and **Card Form** types.



c. Choose **OK**.

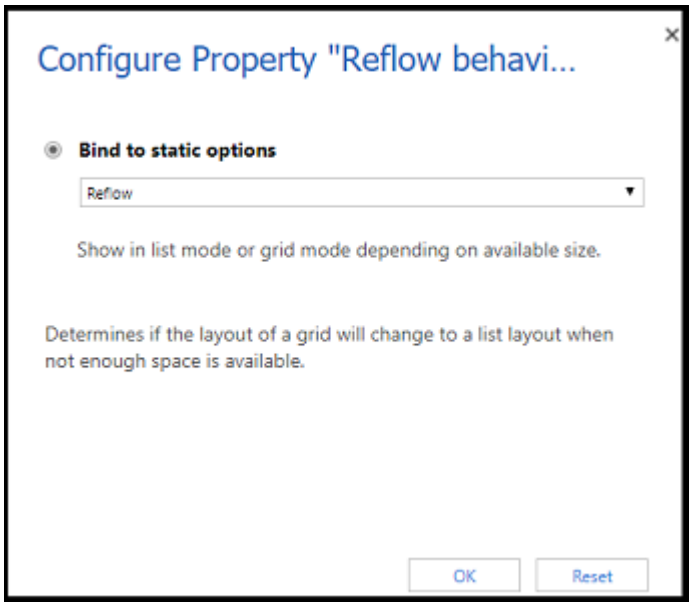
10. Configure the **Reflow behavior** property.

    a. Choose the pencil icon next to **Reflow behavior**.



    b. Select the grid flow type from **Bind to static options** drop down.

Flow Type	Description
Reflow	Allows the grid to render into list mode depending when there is not enough display space.
Grid Only	Restricts the grid to reflow into list even when there is not enough display space.
List Only	Displays only as a list even when there is enough space to display as grid.



c. Choose **OK**.

11. Save and publish the changes.

Conditional image

You can display a custom icon instead of a value in a list and establish the logic used to select them based on a column's values by using JavaScript. For more information about conditional images, see Display custom icons instead of values in list views.

Known issue

If the dataset displayed in the grid contains duplicate rows, the duplicates might not display in the grid. This can lead to the reported record count showing more records than are actually in the grid, or more records appearing when exporting the data to Excel or viewing the data in legacy Advanced Find. This behavior applies to all grid controls, not just the Read-only grid control.

Next steps

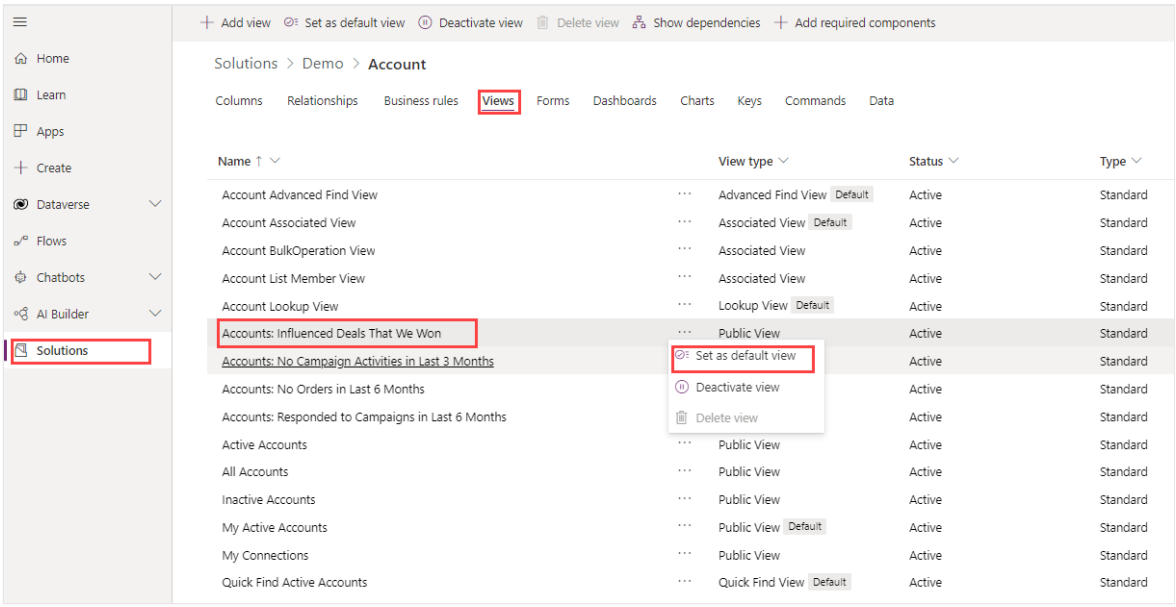
Model-driven app views overview

Specify a default view for a table

Article • 02/23/2023

Unless someone has 'pinned' a different view in your app as their personal default, they will see the default view specified by the app maker. Any of the public views can be set as the default view for a table.

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. Open the solution where the table is located.
4. Open the table, and then select the **Views** area.
5. Select **More commands** ••• next to the view, and then select **Set as default view**. This can also be set on the command bar.



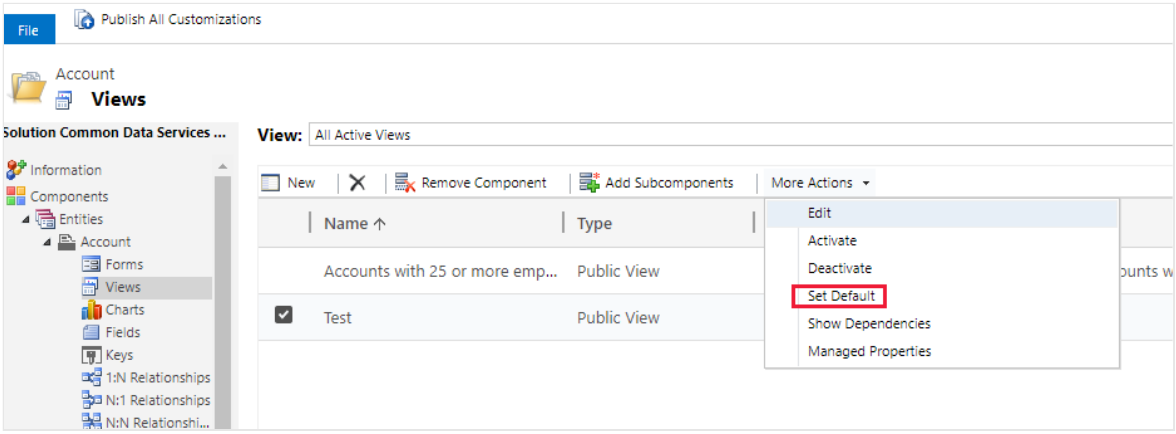
> **Note**
>
> You can also specify the views that can be displayed in a model-driven app using the model-driven app designer. More information: **Manage views and charts**

Set the default view for a table in solution explorer

1. Open solution explorer, expand **Entities**, select the table that you want, and then select **Views**.

2. Select one of the public views.

3. On the menu bar, select **More Actions** > **Set Default**.



4. On the solution explorer toolbar, select **Publish All Customizations**.

Next steps

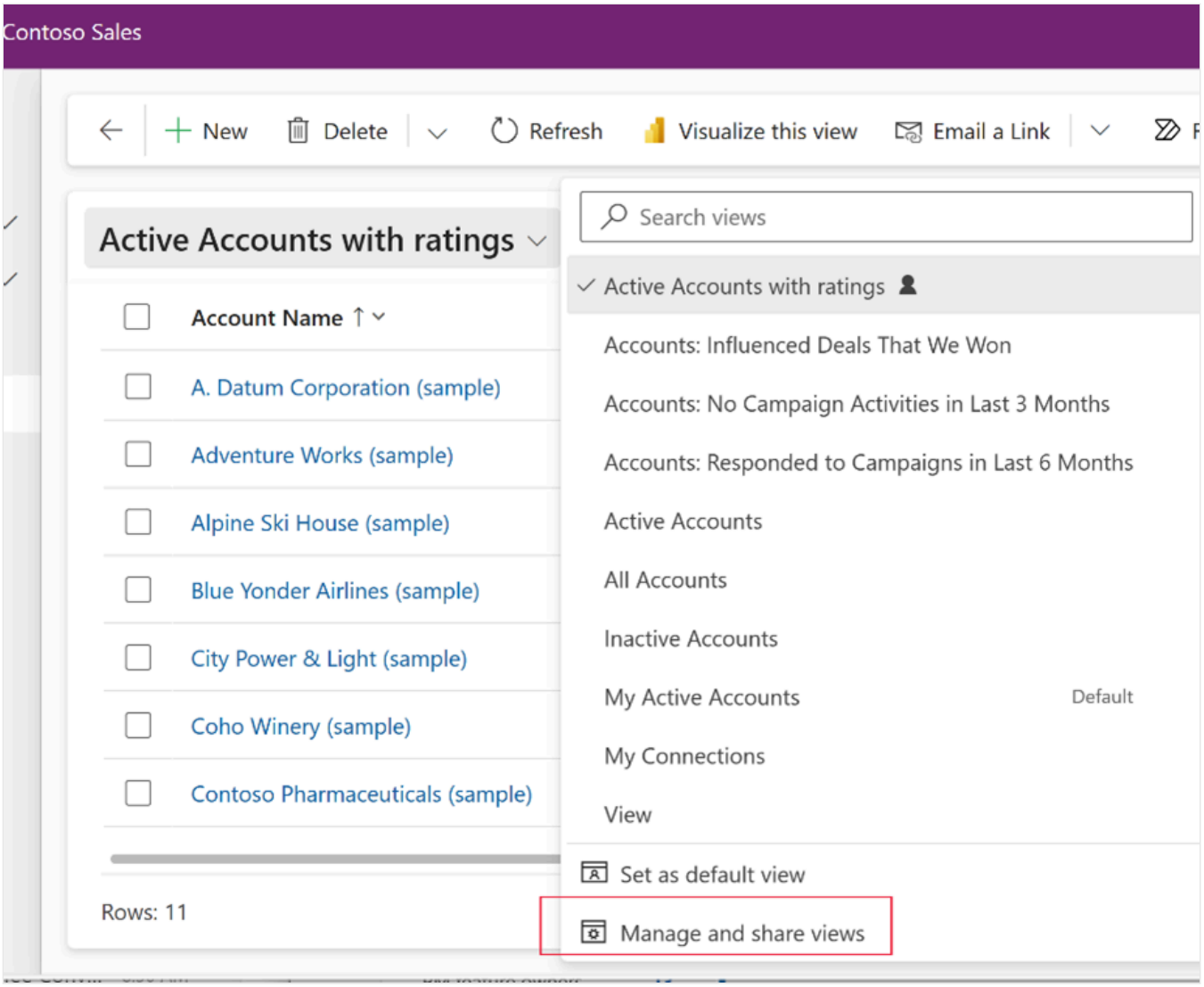
Delete or deactivate a view

Manage access to public system views

Admins can manage the views that users can access in model-driven apps by using security roles. When a user plays a model-driven app, they only have access to the system views that apply to the security roles they're assigned.

System views are special views that model-driven apps depend on, which exist for system tables or are automatically created when you create custom tables. These views have specific purposes and some additional capabilities. By default, all system views are for **everyone**. When a Power Platform admin manages a view with a security role, only the users who are assigned with the selected security role are able to see the view in the view selector. The other system views aren't filtered from the view selector dropdown list. Users can still access all the views when they navigate to the **Manage and share views** option.

After a Power Platform admin turns on the manage table list views feature, users can set their own default view from the list of views that the admin manages and their own personal views from the **Manage and share views** option in a model-driven app.



Data access continues to be secured with security role privileges, which means only records that a user has access to are displayed by the views. Having access to a particular view doesn't mean you also have access to the data that's available in the view.

> **Note**
>
> This feature filters views that appear in the table list view selector. All system and associated views continue to display in subgrids and associated grids that have a view selector.

Prerequisites

- Confirm that the orgdbsettings `EnableRoleBasedSystemViews' property is set to **true** using the OrganizationSettingsEditor tool.
- System administrator security role membership in the Microsoft Dataverse environment.
- Turning on auditing is recommended, but not required.

System views in a Dataverse environment

System views are predefined views that exist for all system tables. These views can include Public, Quick Find, Advanced Find, Associated, or Lookup. When an admin or maker creates a view from Power Apps (make.powerapps.com), the view becomes a public view. You can manage public views with security roles.

Commonly used public view examples

You can manage the table list of views with your business users based on the users' security role assignment. Admins select the applicable system views and manage these views with security roles. When the user navigates to a table list form, they can access the system views that were managed with the security roles that are assigned to them. The selected system views with security role aren't visible to users who don't have the security role assigned.

Examples of public views:

Expand table

Table	Public view
Account	My Active Accounts
Account	Accounts: No Campaign Activities in Last 3 months
Lead	My Open Leads
Lead	All Leads
Case	All Cases
Case	Cases I follow
Case	My Cases
Case	My Resolved cases
Contact	All Contacts
Contact	Contact I follow
Contact	Inactive contacts

Turn on manage table list views

1. Make sure you enable the table list views feature in your environment. More information: Prerequisites
2. You manage public views with security roles. If you need to create new security roles, go to Security roles and privileges for more information about security roles.
3. Create a solution and add the tables and their corresponding views that you manage.
4. Export the solution.
5. Import the solution into your production environment.

> **Tip**
>
> Inform your users that their system views are now filtered based on their security role assignment.

Manage public views with security roles

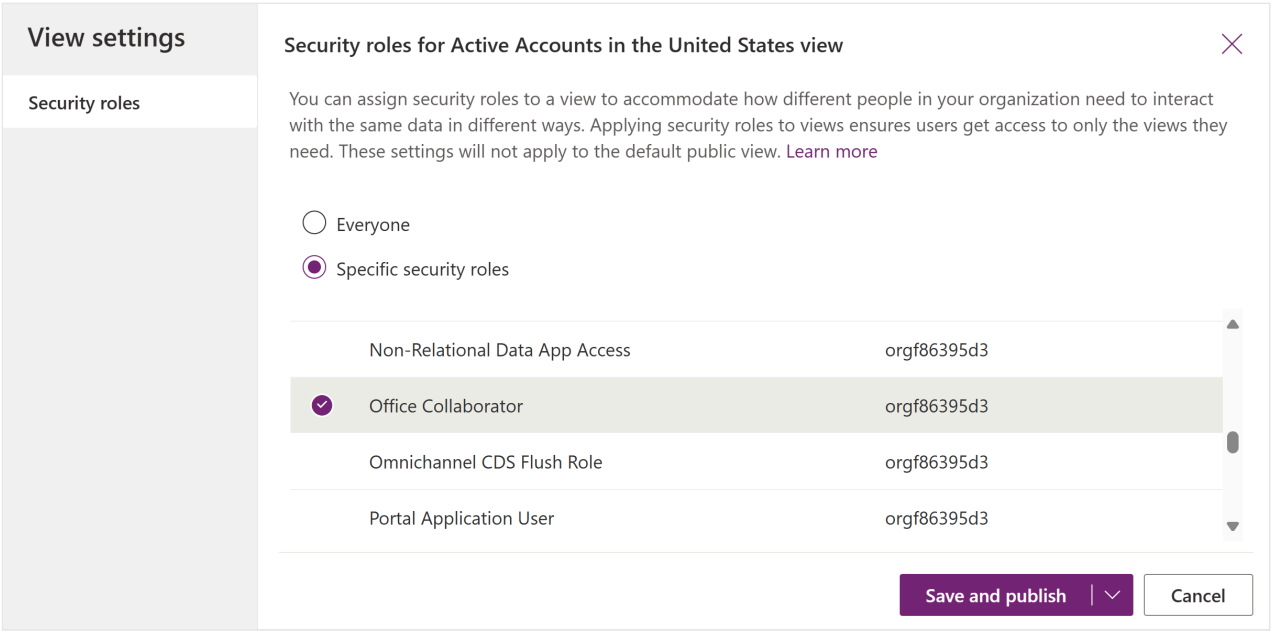
All system views can be managed with security roles. Once they're managed, users with the assigned security roles only view these views in the view selector.

> **Note**
>
> System views are denoted as **Public** view type in Power Apps (make.powerapps.com). Only public views can be managed. Default public views can't be assigned security roles for access.

Manage system views with security roles

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**, and then open the solution that has the table with the view you want to manage. If the item isn't in the side panel, select **...More** and then select the item you want.
3. Open a table, such as the account table, and then select the **Views** area.
4. Select the nondefault public view that you want, and then on the command bar, select **View settings**.
5. Select the **Specific security roles** option.
6. Select the security roles that you want to assign to the public view.

7. When you're done selecting the security roles, select **Save and Publish**.



> **Note**
>
> When you make changes to the 'EnableRoleBasedSystemViews' and **View settings**:
>
> 1. Setting the OrganizationSettingsEditor tool `EnableRoleBasedSystemViews` property to **true** is effective immediately.
> 2. Setting a view with security roles is effective immediately after you select **Save and publish**.
> 3. Changing a view setting from 'Specify security role' to 'Everyone' can take up to 24 hours to be effective or until the user signs out and back in.
>
> If you select multiple views and go to the view settings, only the first selected Views can be updated. You're required to select individual view to update the view settings.

> **Important**
>
> The security roles as listed under **View settings** are from the *root business unit*. Business unit level security roles are inherited from the root business unit. When you select the security roles from **View settings**, the system automatically applies filtering to users who are assigned the corresponding business unit level security roles.

Related articles

Create or edit a model-driven app view in Power Apps

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 04/06/2026

Delete or turn off a model-driven app view

Article • 10/09/2024

Views allow the data in Microsoft Dataverse tables to be presented in a way that meets a business need at a given time.

The views that are required in an organization might change over time. It's good practice to maintain views in order to keep your model-driven apps as simple as possible to use and maintain.

System views versus public views

Depending on the type of view, you can either turn it off or delete it. Turned off views can be turned on again, while the deletion of a view is permanent.

- Custom table public views can both be deleted and turned off/on. Public views that are standard with Dataverse, such as the active accounts and inactive accounts standard public views can be turned off/on.
- System views can't be deleted or turned off/on. This includes public views created by the system.

Turn on or turn off views

Custom public views can be turned off and these turned off views can be turned on. While a view is turned off, the view isn't available in the associated model-driven apps for users to select. Public views set as the default view for the table can't be turned off.

1. Sign in to Power Apps, open the environment you want, select **Solutions** on the left navigation pane, and then open the unmanaged solution you want.
2. Open the table you want, and then select the **Views** area under **Data experiences**.
3. Select the view you want to turn on or off, and then select either **Turn off** or **Turn on** on the command bar.

Delete a view

Deleting a view permanently removes it from the environment.

1. Sign in to Power Apps, open the environment you want, select **Solutions** on the left navigation pane, and then open the unmanaged solution you want.
2. Open the table you want, and then select the **Views** area under **Data experiences**.
3. Select the view you want to remove, and then select **Remove** > **Delete from this environment** on the command bar.

> **Note**
>
> - Where a model-driven app specifies a view this creates a dependency between the app and the view. If this is the case the view will need to first be removed from the app using the app designer.
> - If the view has been added to more than one solution, you can select **Remove** > **Remove from this solution** to remove from the solution but keep available in the environment.

Delete or activate and deactivate views in classic solution explorer

Delete a view in classic solution explorer

You can delete any custom public view. Use the steps in Access view definitions to find the view you want to delete and use the **Delete** command. Once you verify that you really want to delete it, the view is permanently deleted.

Deactivate or activate views in classic solution explorer

1. Navigate to **System Views** as described in Access view definitions.
2. Select a public view. To see inactive views, use the **Inactive Public Views** view.
3. On the menu bar, select **More Actions**, and then select either **Deactivate** or **Activate**.
4. Select **Publish All Customizations**.

Next steps

Views overview

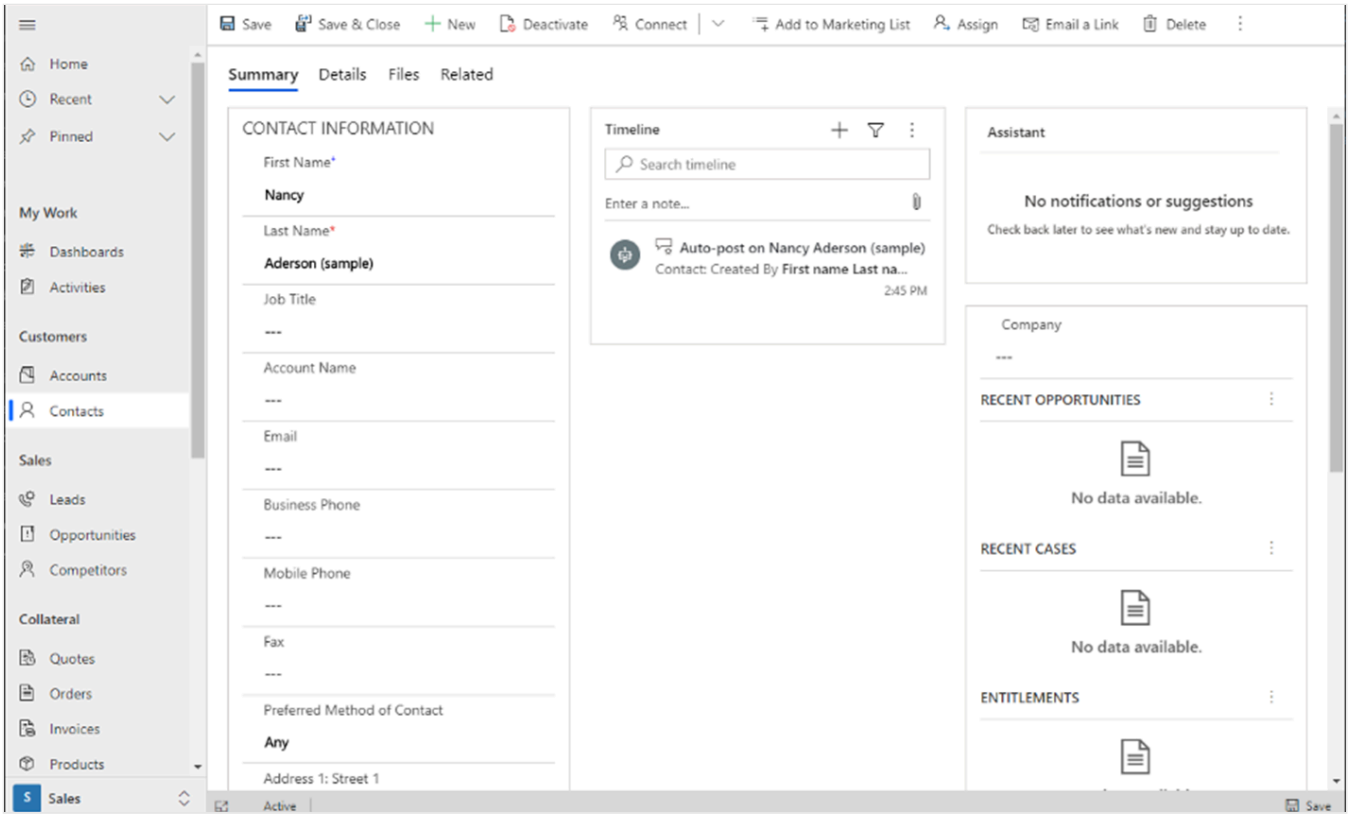
Feedback

Was this page helpful? Yes No

Provide product feedback

Create and design model-driven app forms

With Power Apps model-driven apps, forms provide the user interface that people use to interact with the data they need to do their work. It's important that the forms people use are designed to allow them to find or enter the information they need efficiently.

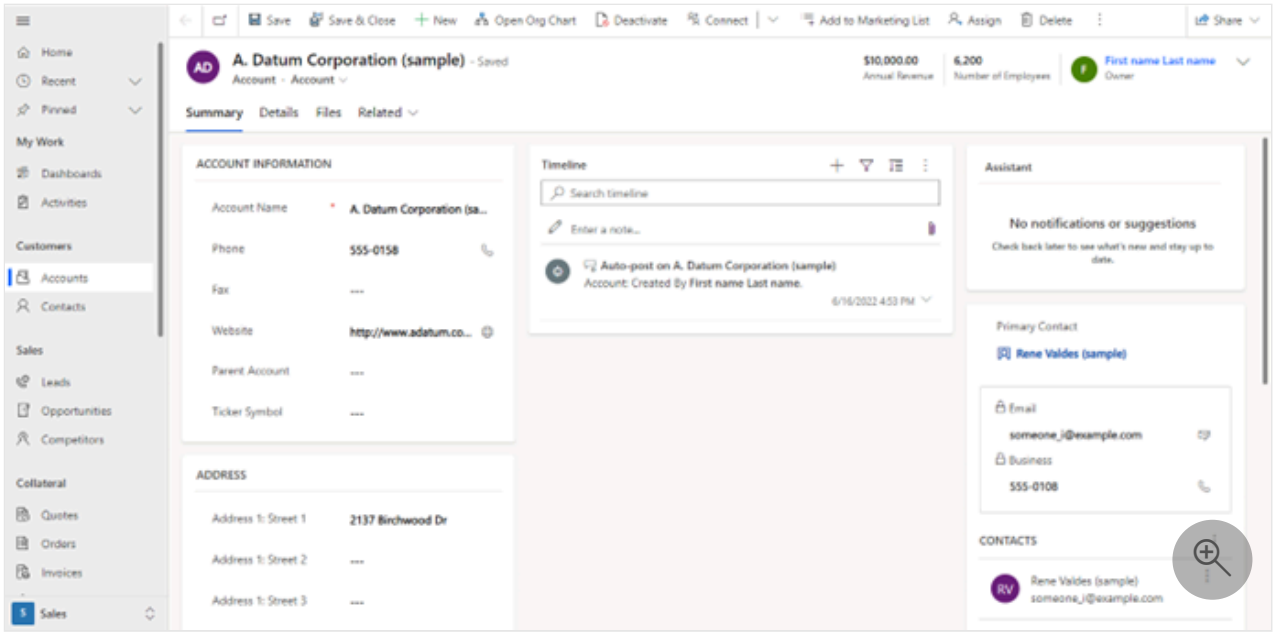


In the default solution or an unmanaged solution, it's possible to create new forms or edit existing forms for all tables that allow form customization.

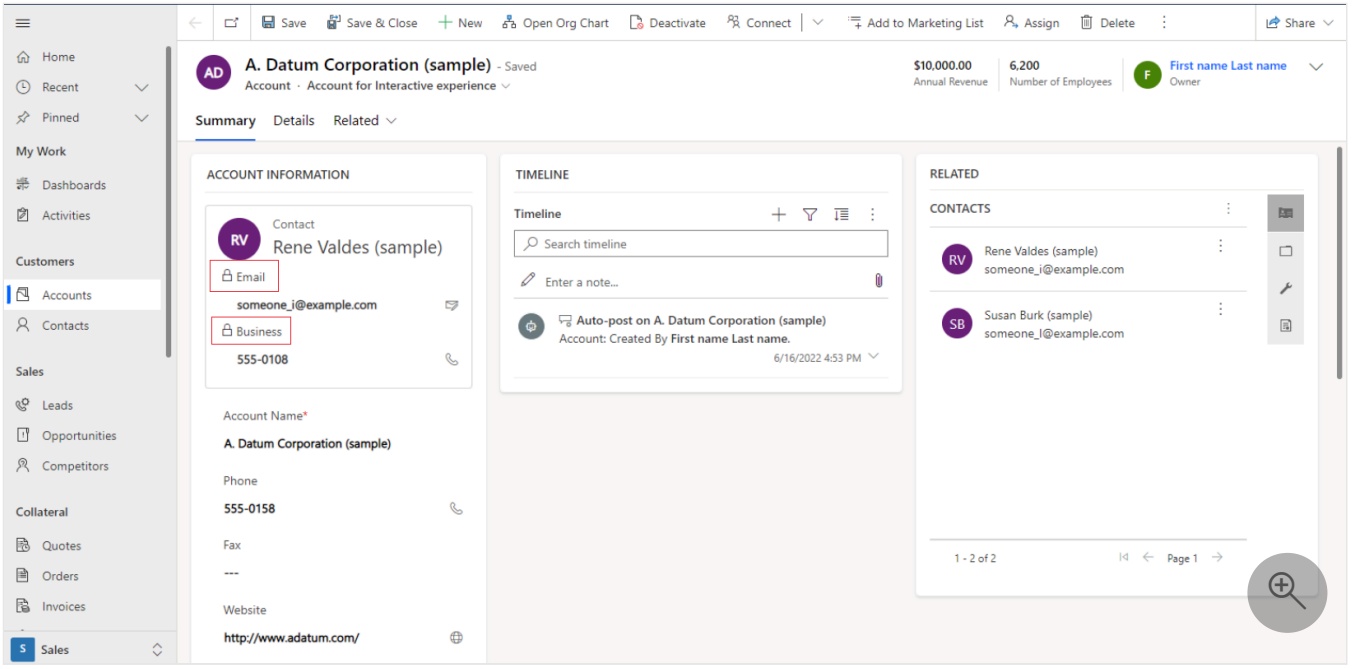
Model-driven app form UI enhancements

Form UI enhancements consist of multiple small changes to improve data presentation and usability of model-driven app forms:

- Forms have a light gray background, which removes the white-on-white section treatment that helps users by providing easier visual navigation.
- There's a better delineation between sections as white space is reduced and shadow and rounded corners to borders are added.
- Form field dividers are removed and consistent font styling is used across section labels.
- The **Related** tab has an added chevron to indicate the dropdown flyout, which opens-ups when you select it.



Also, quick view form labels are displayed on top instead of icons.



Forms and solutions

This section describes form customization when the form is in an unmanaged or managed solution.

Forms inside unmanaged solutions

In an unmanaged solution, it's possible to edit the properties, including columns, views, and forms, for an unmanaged custom table that was created for the solution. Unmanaged solutions are where app makers and developers go to author changes that they require. Unmanaged solutions typically exist in a development environment.

Forms inside managed solutions

In a managed solution, typically in a production environment, it isn't possible by default to create new forms or edit existing forms for tables. However, if the managed properties for a table in the managed solution are set to allow customization, you can add or edit forms for that table.

Learn more about solutions

Form types

There are different types of forms, and each type has a specific functionality or use. Here are the form types:

- Main. The main user interface.
- Quick create. For rapid data entry.
- Quick view. Used to see related data.
- Card form. Shows a compact display of the form.

More information: Types of forms in Power Apps.

Create or edit a form

The fundamentals of building a model-driven app include creating a table, configuring data views, and creating and editing forms.

Before you begin to build forms, it's worth determining whether you have all the columns necessary to solve your business problem. Additionally, you should have an understanding how to organize columns by using sections and tabs. Broadly speaking, the complexity of your forms are a reflection of the number of columns in your table in addition to the complexity of your business process.

Learn to create, edit, or configure forms using the form designer

Delete a form

To delete a form, sign in to Power Apps and go to **Solutions** > Open the solution you want > select the table that you want > **Forms** area. Select the form, and then select **Delete** on the command bar.

There are a couple of reasons you might not be able to delete a form.

Expand table

Reason	Work around
Every table requires at least one main form and it's the only main form for the table.	Create a new main form for the table. Then delete the main form you tried earlier. More information: Create a form
Every table requires one designated fallback form and it's the only fallback form.	Create a new form for the table and set it as the fallback. Or designate another existing form as the fallback form. Then delete the form you tried earlier. More information: Set the fallback form for a table

Controlling the display of a form within an app

Tables can have multiple forms. The use of a form is controlled through the model-driven app. This control allows makers to use the same table in different ways depending on the audience, by controlling the display.

By default all forms are made available to users. However, through the app designer a specific form can be restricted.

Change the forms displayed and form order

Makers change the forms displayed in an app from the app designer. More information: Add or remove forms from an app

Makers can also change the form order, fallback forms, and the default form for a table from the Forms area. More information: Set the form order

Updated versus classic tables

Power Apps provides many options for designing forms using the form designer.

There are still a number of tables, referred to here as classic tables, that retain the appearance and capabilities from earlier versions. These tables are used less often. They're listed here:

Address

Article

Article Comment

Bulk Delete Operation

Connection

Discount

Discount List

Document Location

Email Attachment

Follow

Goal

Goal Metric

Import Source File

Invoice Product

Order Product

Price List

Queue Item

Quote Product

Rollup Field

Rollup Query

Saved View

Service

Service Activity

SharePoint Site

Site

Territory

Unit

Unit Group

Main form dialogs

With the client API, you can use main form dialogs so users can open a related row table on a parent or base form without navigating away from the form. More information: Open main form in a dialog using client API

Next steps

Understanding the form types available

Assign form order
Control access to forms
How main forms appear in different clients

Last updated on 03/12/2026

Types of model-driven app forms in Power Apps

Summarize this article for me

There are different types of forms and each type has a specific functionality or use. This table describes the types of forms available.

Expand table

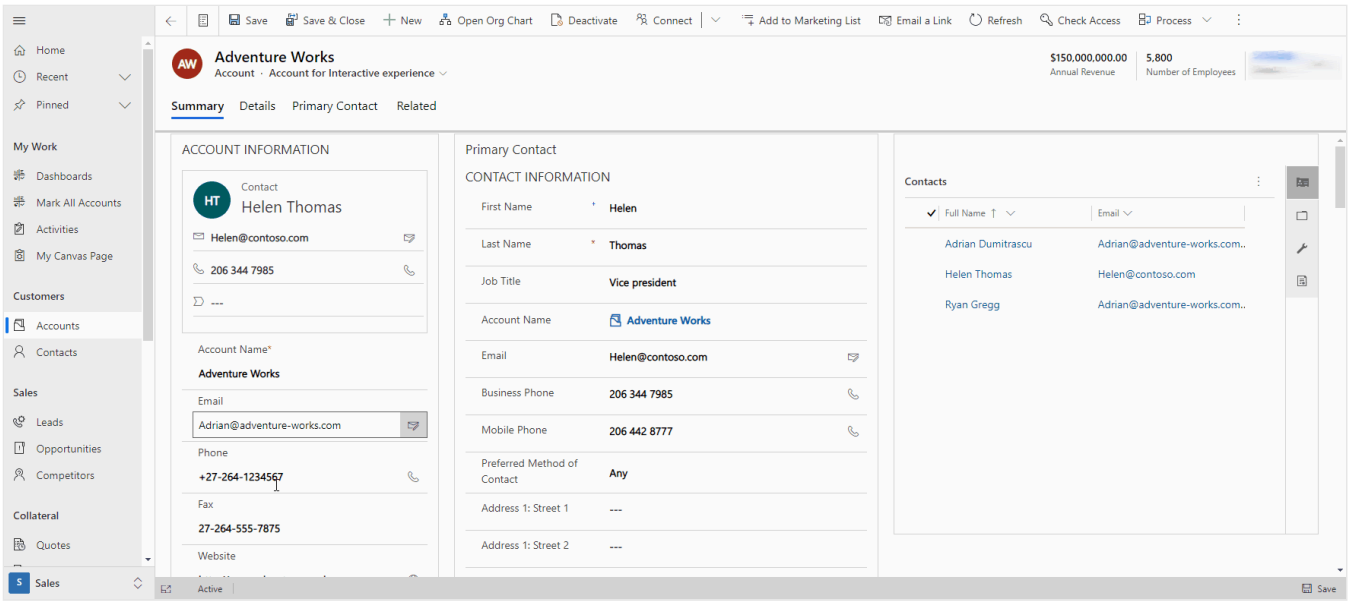
Form type	Description	More information
Main	Used in model-driven apps, Dynamics 365 for tablets, and Dynamics 365 for Outlook. These forms provide the main user interface for viewing and interacting with table data.	Design considerations for main forms
Quick Create	Used in model-driven apps, Dynamics 365 for tablets, and Dynamics 365 for Outlook. These forms provide a basic form optimized for creating new records.	Create and edit quick create forms
Quick View	Used in model-driven apps, Dynamics 365 for tablets, and Dynamics 365 for Outlook. These forms appear within the main form to display additional data for a row that is referenced by a lookup column in the form. Users can view data from related tables without having to leave the form.	Create and edit quick view forms
Card	Used in views for model-driven apps. Card forms are designed to present information in a compact format that is suitable for mobile devices.	Create a card form

While each form type has specific needs, when designing forms you use the form editor. More information: Overview of the form editor user interface

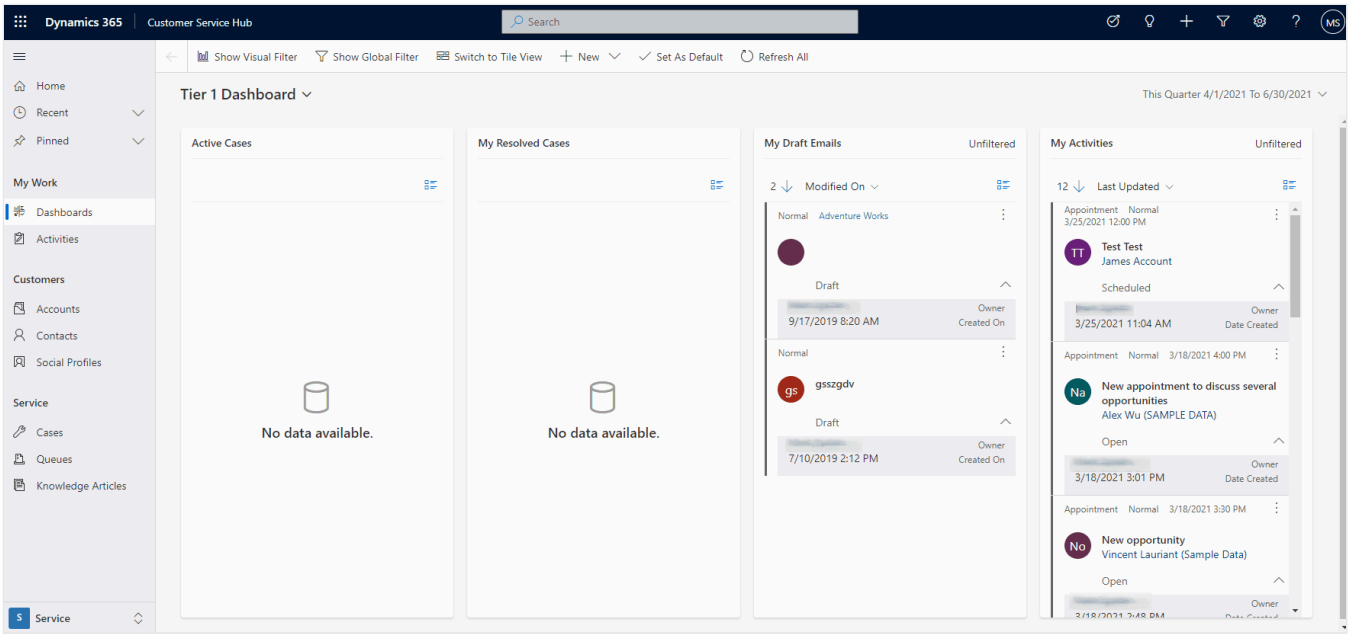
Form examples

Here are examples of each form type.

Main form



Quick create



Quick view

PP

Parent Properties

Lesson

General Related

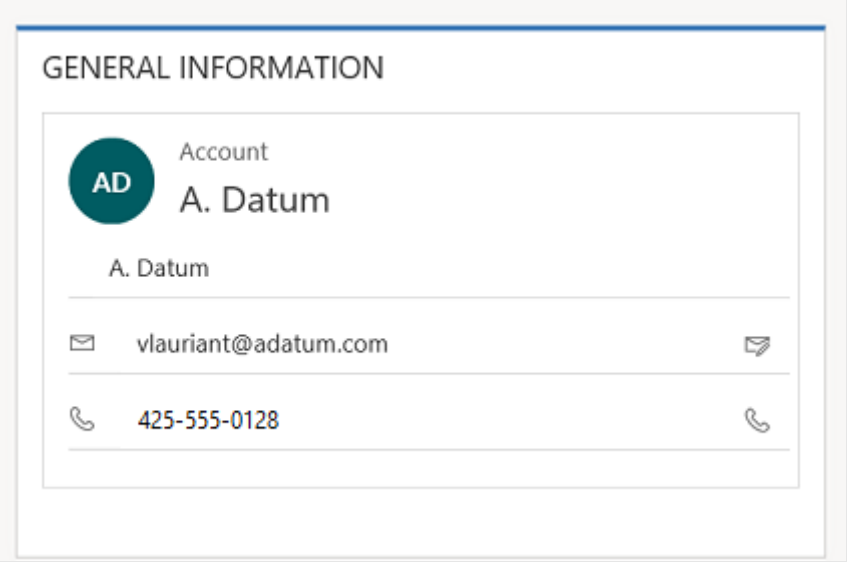
LessonName	*	Parent Properties
isPremium		No
Owner	*	[redacted]
CompletionPercent		100
LessonOrder	*	60
Module	*	Object Properties

Quick View Section

CompletionPercent	*	100
ModuleName	*	Object Properties
ModuleImage		 Open
Course		[redacted]

Detail	---
URLResource	---

Card



Next steps

Overview of the form editor user interface

Last updated on 02/13/2026

How model-driven app main forms appear on different devices

Article • 04/20/2022

There are four types of form. Main, quick view, quick create, and card.

The main form is a fundamental building block of any model-driven app and is used by all devices used to run them.

A main form can be rendered using a web browser, Dynamics 365 for phones, Dynamics 365 for tablets, or Dynamics 365 for Outlook, and its design adjusts to accommodate the device. This responsive design is an important feature of model-driven apps.

Additionally, the main form is an intrinsic part of the table and it travels with the table as part of any solution. This helps with application lifecycle management.

Learn more about all the form types

Notes on Microsoft Dynamics 365

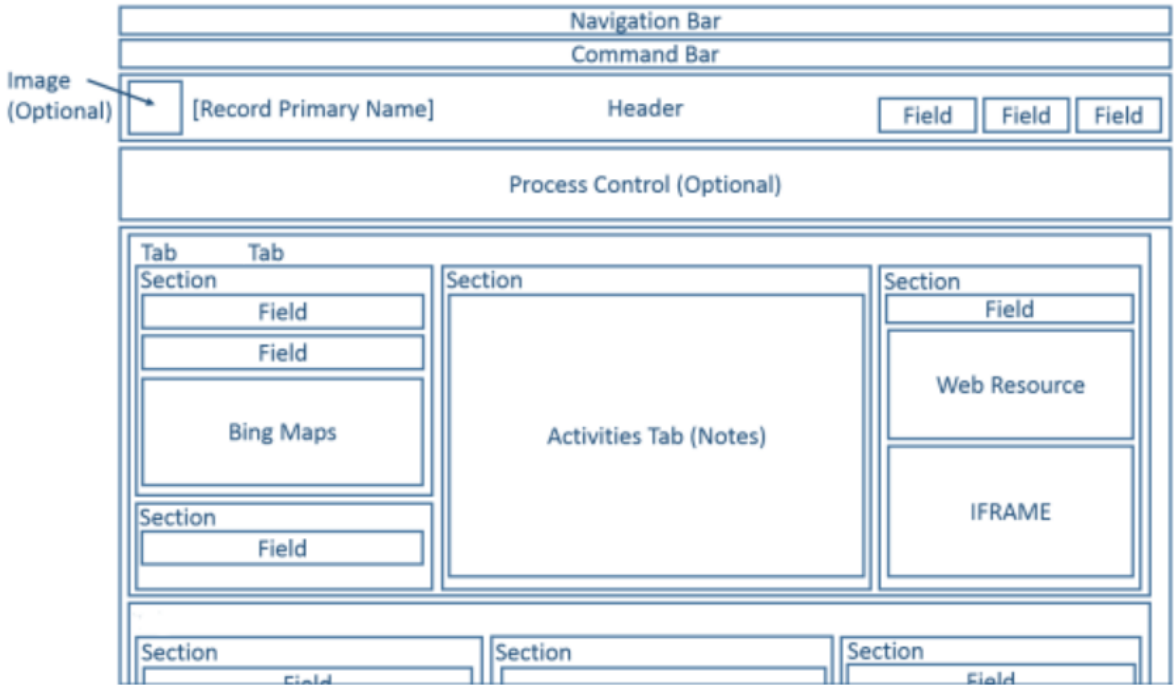
Main form presentation options

Any main forms that exist for a table may be displayed differently depending on the factors in the following table below. When you design a main form, consider how it works in each presentation tool.

Presentation	Description
Current	Most of the standard tables and all custom tables created use the current form, which provides a modern user experience. The current forms have a layout with an improved command bar design, and support features such as AutoSave and business process flows.
Dynamics 365 for tablets	Dynamics 365 for tablets presents the content of the main form in a manner optimized for a tablet.
Dynamics 365 for phones	Dynamics 365 for phones presents the content of the main form in a manner optimized for a phone.

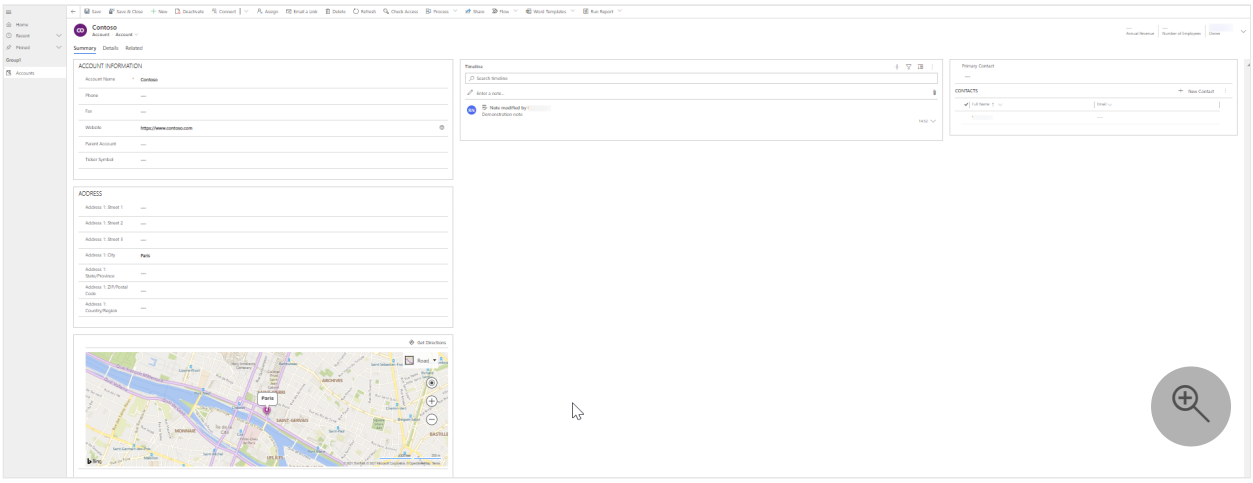
Presentation	Description
Classic	Classic forms are from earlier versions of Dynamics 365 Customer Engagement apps. They use the ribbon rather than the command bar and the navigation pane on the left side of the form. These forms have a two-column layout. Notice that there are a few tables that still use the classic forms. More information: Updated verses classic tables

This diagram represents common components found in main table forms.



The layout of the form works with a wide range of displays and window sizes. As the width of window decreases, section columns move down so that you can scroll down to work with them instead of being compressed or requiring you to scroll to the right.

The image below illustrate the accounts table main form as it would be viewed through a web browser.



The following table summarizes available components of the main form for tables.

Component	Summary
Navigation bar	In the context of the table, the navigation bar provides access to managing records, such as search, create, and advanced find.
Command bar	The first several commands are displayed followed by a vertical ellipsis that provides a menu to choose additional commands.
Image	When a table has an image column and the table **Primary Image** option is set to **Default Image**, an image can be displayed in the header when the form is configured to show the image.
Header	Columns placed in the header remain visible when people scroll down through the body of the form. Up to four columns can be placed in the header. Multiple lines of text, web resources, or iFrames aren't allowed in the header. The header shares some properties with sections.
Process Control	When a table has active business process flows, the process control displays below the header. More information: Business process flows
Body	The body is the scrollable part of the form that contains the sections.
Tabs	In the body of the form, tabs provide horizontal separation. Tabs have a label that can be displayed. Tabs contain up to three columns and the width of each column can be set to a percentage of the total width. When you create a new tab, each column is prepopulated with a section.
Sections	A section occupies the space available in a tab column. Sections have a label that can be displayed and a line may be shown below the label. Sections can have up to four columns and include options for displaying how labels for columns in the section are displayed.
Columns	Section columns display fields and controls people use to view or edit data in a table record. A section can be formatted to occupy up to four columns within the section.
Spacer	A spacer allows for an empty space to be added to a section column.
Sub-grids	Subgrids allow for the display of a list within the form.
Quick View Form	A quick view form displays data from a record referenced by a lookup column on the form. The table that is the target of the lookup must have a quick view form before one can be added to the form. More information: Create and edit quick view forms

Component	Summary
Web Resources	HTML and Microsoft Silverlight web resources can be added to main forms but they won't be displayed when using Dynamics 365 for phones and tablets.
iFrame	An inline-frame that you configure to show a webpage from another website. **Important:** • When the page displayed in an iFrame is on another domain, browsers apply a higher level of security. This may complicate the requirements for the contents of an iFrame to interact with data in the form. • Displaying a table form within an iFrame embedded in another table form isn't supported.
Bing Maps	When this control is present in a form for a table and the system setting **Enable Bing Maps** is enabled with a valid Bing Maps key, this control can be used one time in a form to show the location for one of the addresses in a table. More information: Configuring Bing maps

Dynamics 365 for phones and tablets forms

Most system tables and custom tables are available for Dynamics 365 for phones and tablets. The main form for these tables is transformed to a presentation optimized for phones or tablets.

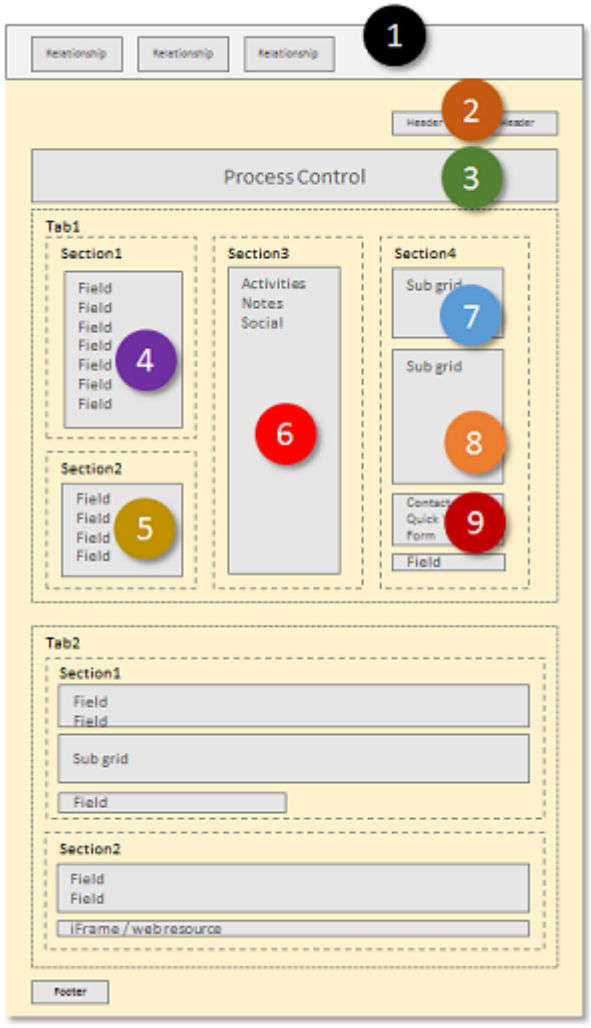
Tables enabled for Dynamics 365 for phones and tablets

Only tables that are enabled for Dynamics 365 for phones and tablets use this presentation of the main form. More information: Entities displayed in Dynamics 365 for phones and tablets

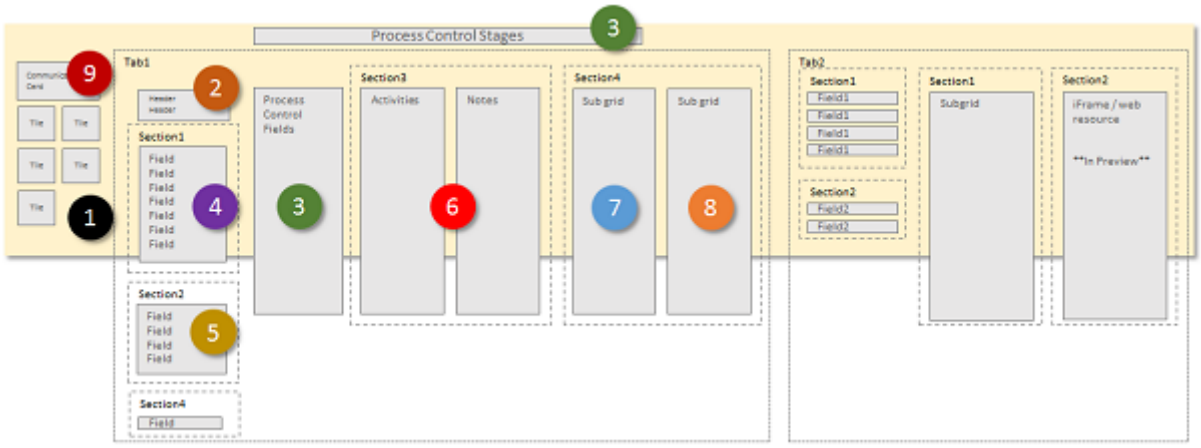
Form design

Dynamics 365 for phones and tablets takes many of the main form elements and presents them in a way optimized for phones or tablets. The following diagrams show the reflow from the web app to the tablet and phone apps.

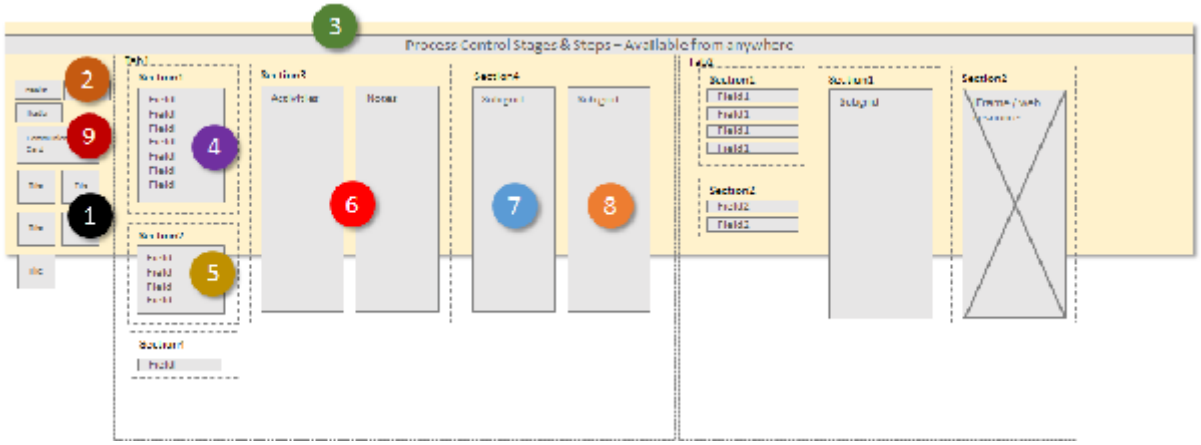
Web app



Tablet app



Phone app



The form elements are transformed to a wide panorama layout in Dynamics 365 for tablets, where users swipe the screen to change elements visible within a view port. In Dynamics 365 for phones, users swipe the screen to see a different column, or pane of elements, and the process control appears over every column.

View port element

The following items are always visible within the view port in the context of a form:

Nav bar
The nav bar is a presentation of the sitemap that is optimized for touch. More information: Change navigation options

Home
The home button takes users to the dashboard that is the starting page for Dynamics 365 for phones and tablets.

Process Control
If the table has a business process enabled, it will appear in the top-right corner next to the search control in Dynamics 365 for tablets, and at the top of the screen in Dynamics 365 for phones.

Search
People can tap the search control to open the screen to search for records.

Command Bar
By default, some of the commands that appear in the app running in a web browser don't appear in the Dynamics 365 for phones and tablets apps. Similar to the web application, the command bar is context-sensitive, so the available commands change depending on what is currently viewed or selected. More information Change commands

Form elements

The form elements displayed are taken from the main form and presented as a series of panels that users see through the view port.

In Dynamics 365 for tablets, the first panel displays contact information about relationships that exist for the record. In Dynamics 365 for phones, the first panel also displays header columns from the form above the relationship tiles.



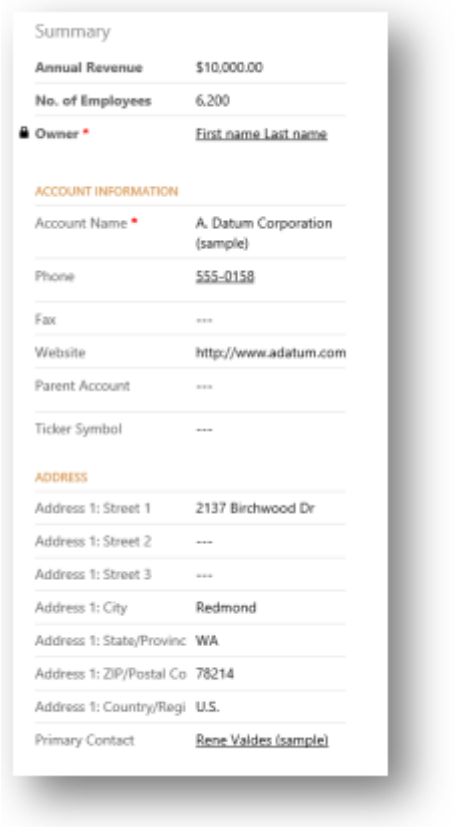
For Contact and User forms, the top item displays a communication card for the record. The communication card provides buttons to initiate communication with the person. For other tables, a communication card is displayed if there's a Contact quick view form embedded in the main form.

You can show additional tiles based on table relationships, but you can't customize the tiles for the following tables:

Entity	**Tiles**
Account	Owner
Contact	Company Name, Owner
Lead	Owner
Opportunity	Account, Owner

You can customize the remaining tiles with the form editor. The order is fixed, but you can set which elements are visible in the relationship panel.

In Dynamics 365 for tablets, the second panel begins with the name of the first tab on the form. Any columns that are included within the header are included and then the contents of the first tab. In Dynamics 365 for phones, headers appear in the first column.



If there is a process flow active for the form, the third tab displays tasks for the current stage of the process in Dynamics 365 for tablets. In Dynamics 365 for phones, the process control floats above the panes, expands over the user's current pane when it's selected, and is always visible and actionable.

The remaining panels of the form contain the contents of the tabs in the form. Any subgrids found display as a separate panel.

The Dynamics 365 for phones and tablets form always displays the labels for tabs and subgrids. The **Display Label on the Form** setting isn't applied.

> ⓘ **Note**
>
> To optimize performance on mobile devices, the number of objects is limited to 5 tabs or 75 columns and 10 subgrids.

Forms for Dynamics 365 for phones and tablets don't support the following features:

- Bing maps
- Yammer

- Activity feeds
- Theming

In addition, table images are visible in list views and contact cards, but not within the actual form.

Multiple forms

Dynamics 365 for phones and tablets supports multiple forms but doesn't provide a way for people to switch between forms if they can access more than one. People will see the first form in the form order that they have access to.

For example, if you have the following main forms for the opportunity table and have assigned the following security roles for each one, you'll see the form order shown in the following table.

Form Order	Form Name	Security roles
1	**Sales Form One**	Salesperson
2	**Sales Form Two**	Salesperson and Sales Manager
3	**Sales Form Three**	Sales Manager
4	**Sales Form Four**	Vice President of Sales

- People with the Salesperson role will always see **Sales Form One**.
- People with the Sales Manager role will always see **Sales Form Two**.
- People with the Vice President of Sales role will always see **Sales Form Four**.

Classic forms

For information about the classic forms available with Dynamics 365 Customer Engagement (on-premises), see Classic forms.

Next steps

Create or edit a main form overview

Design considerations for model-driven app main forms

Summarize this article for me

Main forms are the primary user interface where people view and interact with their data. Main forms provide the widest range of presentation options that are available for model-driven apps.

Other form types include Quick view, Quick Create, and Card.

A fundamental quality of model-driven apps is that they're responsive to the size and type of device used when interacting with them. This responsiveness affects the position of the controls on the form and the way in which they behave. This quality is most notable with main forms.

One of the main design objectives for main forms is that they are designed once per table and deployed everywhere required.

There are several important factors to consider in designing forms for model-driven apps.

Custom forms for different groups

Because multiple main forms can be created and assigned to different security roles, it's possible to present different groups in an organization with a form that is optimized for how the group uses the application. It's also possible to provide each group with different options so that they have different forms to choose from. More information: Control access to forms

Managers and decisions makers want forms that are optimized to provide quick reference to key data points. They often like to see charts more than lists and they might not perform a lot of data entry.

People who interact directly with customers need forms tailored to tasks they perform most frequently. They often want forms that allow for the most efficient data entry.

Form creation might be an iterative process where input is gathered and the user interface is developed. Keep in mind that there are a variety of tools available to developers and that not everything has to be done within the form. Use business rules, workflow processes, dialogs, and business process flows together with main forms to provide a solution that works for the organization.

Creating and editing forms is relatively easy, but as more forms are created more forms need to be maintained.

Form presentation differences

Although it isn't necessary to manage multiple forms for each delivery mechanism (web, tablet, phone), a maker must consider how differences in the presentation can be accounted for in the main form.

Main form appearance describes the different ways that the main form can be presented. The primary things to take into consideration are:

- Dynamics 365 for tablets doesn’t support image, HTML, or Silverlight web resources to be added to forms.
- The layout of Dynamics 365 for tablets forms is autogenerated based on the main form. There is no special form editor for Dynamics 365 for tablets forms. It is necessary to verify that the form presentation works well for both clients.
- If there are unsupported scripts that interact with DOM elements found in the web application, those scripts won’t work in the Dynamics 365 for tablets forms because the same DOM elements aren’t available.
- Dynamics 365 for Outlook Reading Pane forms don’t allow for scripting. The visibility of form elements depend on the default settings and can’t be changed at runtime using scripts.

Form performance

Forms that load slowly or don’t respond quickly might affect productivity and user adoption of the app. Design forms for performance in model-driven apps provides a number of recommendations to consider when you design forms so that customizations don’t adversely affect form performance.

Design forms for efficiency

Form layout and design are important to building better forms. Designing forms where tasks can be completed quickly and effectively is crucial to user satisfaction. More information: Design productive main forms in model-driven apps

Next steps

Create or edit a main form overview

Last updated on 02/13/2026

Create or edit a model-driven app main form for a table

Summarize this article for me

This article describes how to create or edit a main form for a table.

When a new form is created for a table, its form type is **Main**. When the new form opens, it is identical to the form named **Information**. Columns, sections, tabs, navigation, and properties associated with the form can be edited and the form can then be saved.

Each main form is composed of one or more tabs. Each tab can have one or more sections. Each section contains one or more columns, also called form fields. Forms can be cloned to provide a simpler starting point for form development. To clone a form, open the form you want to copy in the form editor, on the command bar select **Save a copy**, enter a **Display Name** and optionally a **Description**, and then select **Save**.

> **Note**
>
> Make sure that you have the System Administrator or System Customizer security role or equivalent permissions within your environment to perform this task.

How to create or edit a main form

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane, select the table you want, and then select the **Forms** area. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. To create a new main form, on the toolbar select **Newform** > **Main form** > **Create**.
4. Change the form design in any of the following ways, as needed:
   - Add a column to or remove a column from a form
   - Add a section to or remove a section from a form
   - Add a tab to or remove a tab from a form
   - Add or edit a subgrid on a form
   - Edit form headers
   - Edit form footers
5. Edit the properties for parts of the form, as needed:

- Edit form properties
- Edit form column properties
- Edit tab properties
- Edit section properties

6. When you finish editing the form and are ready to make it available in apps, select **Save and publish**.

Next steps

Create and edit a main form walkthrough

Last updated on 02/13/2026

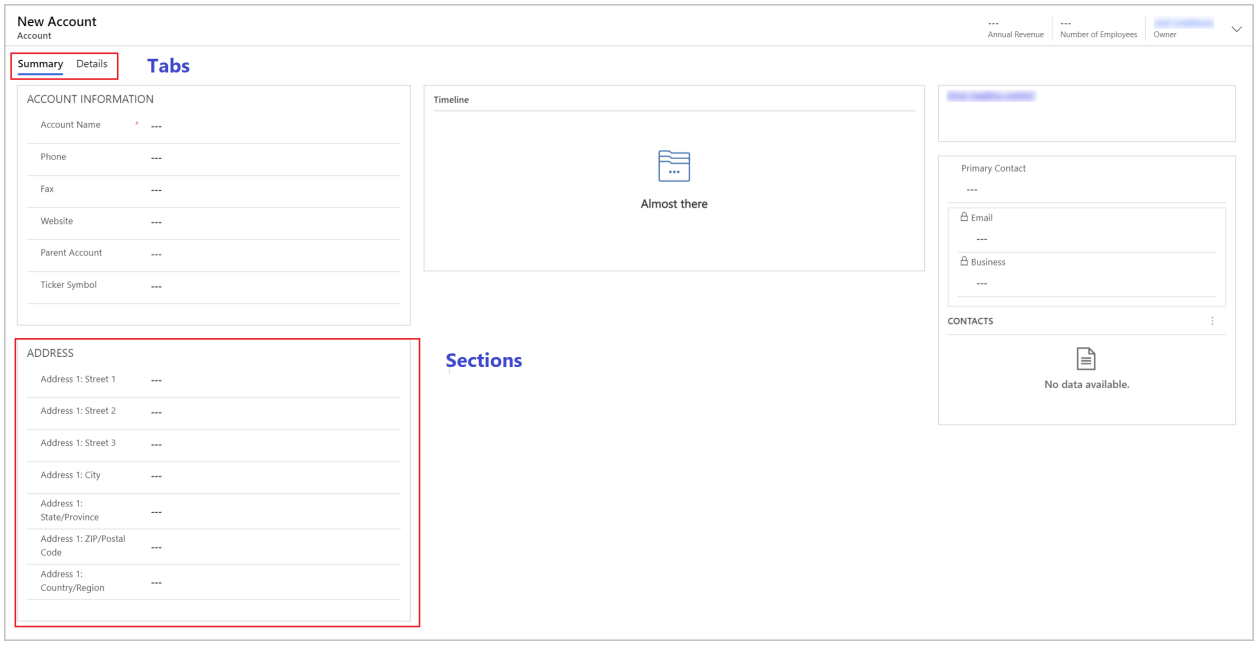
Create and customize a model-driven app form

Article • 03/13/2023

In model-driven Power Apps, forms provide the user interface that people use to interact with the data with which they need to work. A table can have one or more forms, and forms come in different types. The different types are main, quick create, quick view, and card. For more information about form types, see Type of model-driven app forms in Power Apps.

In this tutorial, you create a main type form and then add the form to a model-driven app. The main form is the primary form type that users see when they open records in model-driven apps.

Main forms have a default three-column layout. Model-driven form fields and components are grouped into tabs (primary grouping) and sections (secondary grouping). Tabs separate the form into logical pages, while sections separate the content within a tab.

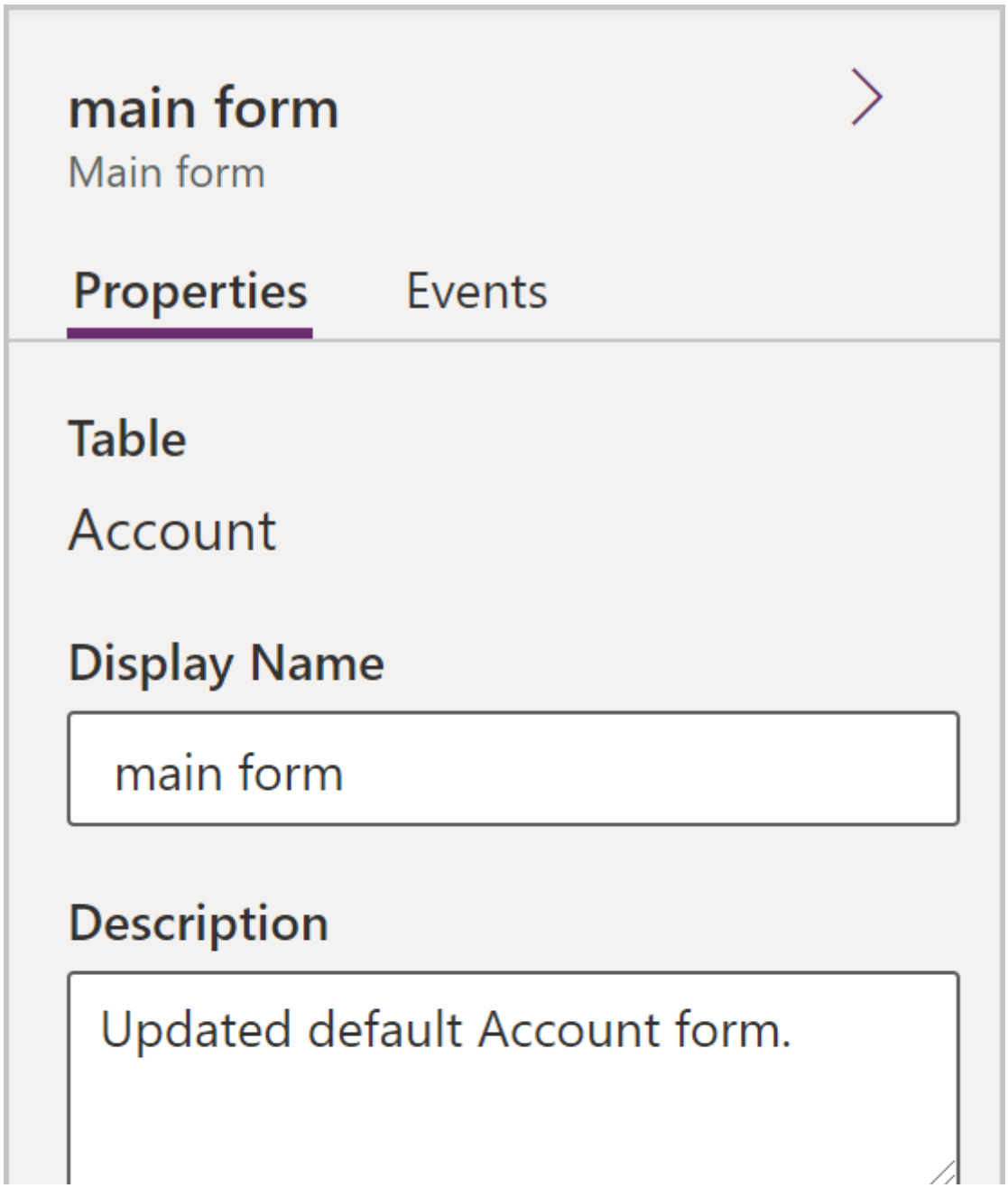


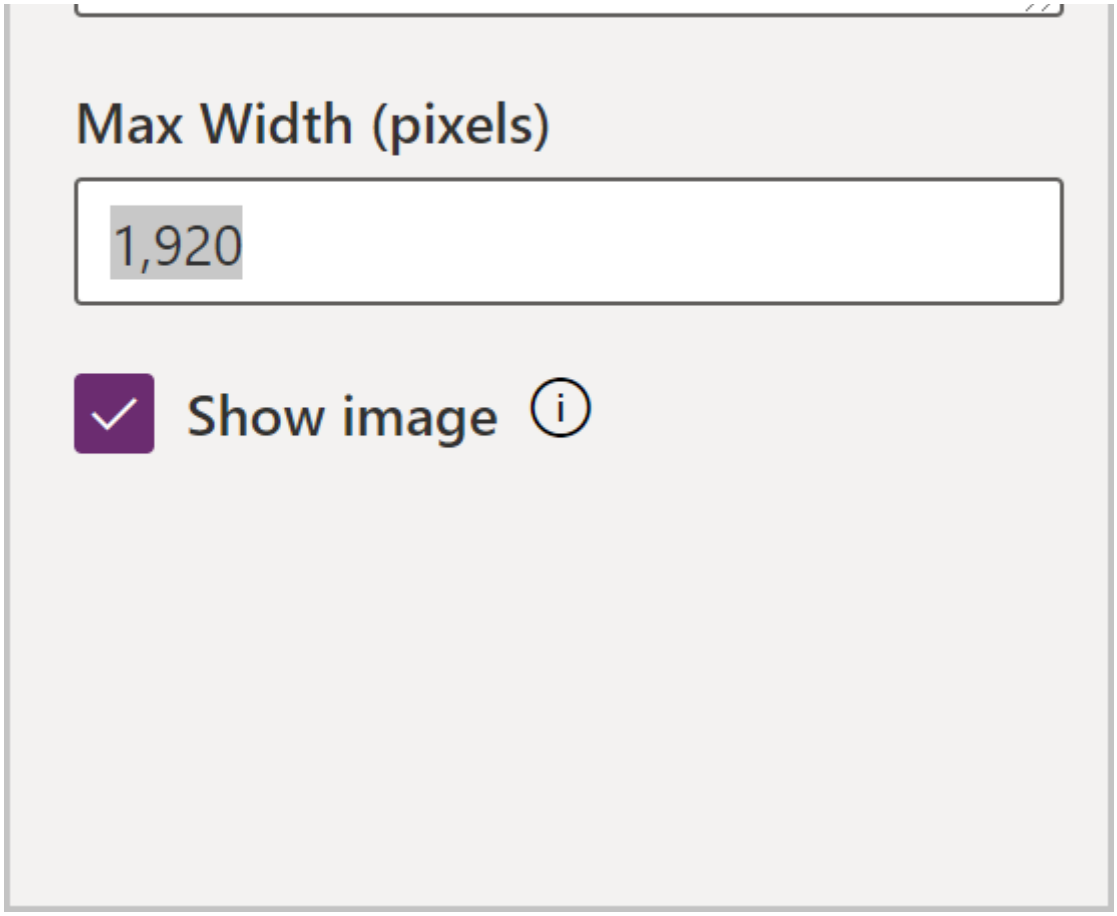
Create and configure a form

1. Sign in to Power Apps.
2. Select **Tables**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. Open the **Account** table, and then select the **Forms** area.
4. Select **Add Form** from the command bar, and then select **Main Form**.

5. The form designer opens to enable you to configure the properties for the form.

Property	Description
Title	Enter a name that is meaningful to other makers and app users. This name is shown to app users. If users have access to multiple forms for a table, they will use this name to differentiate between the available forms. This property is required.
Description	Enter a description that explains how the form is different from other main forms. This description is only shown to makers in the form designer and in the list of forms for a table in the solution explorer.
Max Width	Set a maximum width (in pixels) to limit the width of the form. The default value is 1900. This property is required.
Show image	Show the table's **Primary Image** if it has one set. This setting will enable showing the image column in the header of the form. For more information about table options, see Enable or disable table options .

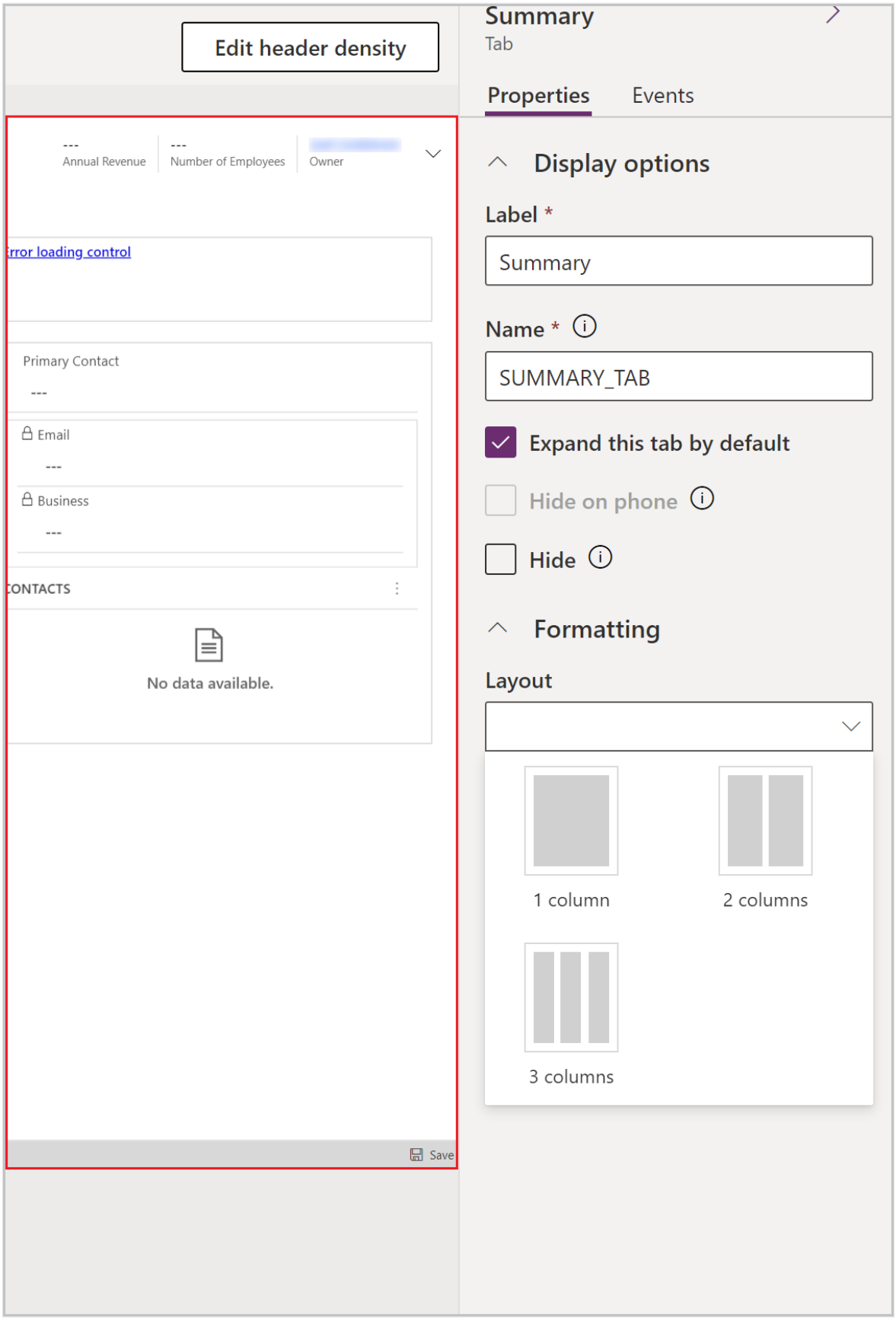




Change the tab layout

If you wish to use a different layout than the default three columns in a tab, follow these steps:

1. Select the tab name, such as **Summary**.
2. In the **Properties** pane on right, select the **Formatting** area to expand it.
3. Select the **Layout** drop-down list, and then select your desired layout, such as **1 column** or **2 columns**.



Add items to the form

Using the left side panel in the form designer, you can add different items to the form depending on which tab you select.

Components

From this tab you can add new form tabs, sections, or visual controls. More information: Add, configure, move, or delete components on a form

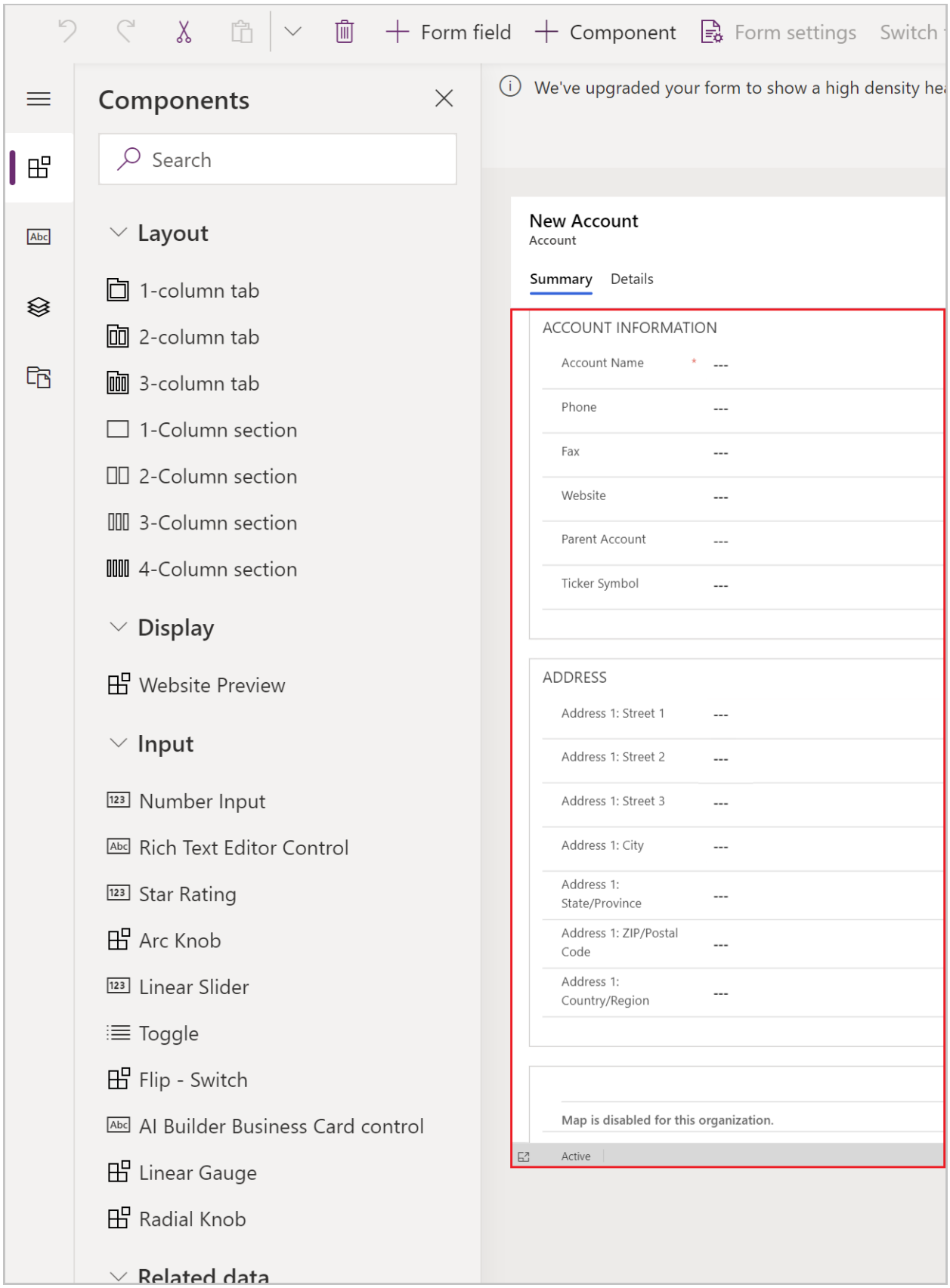
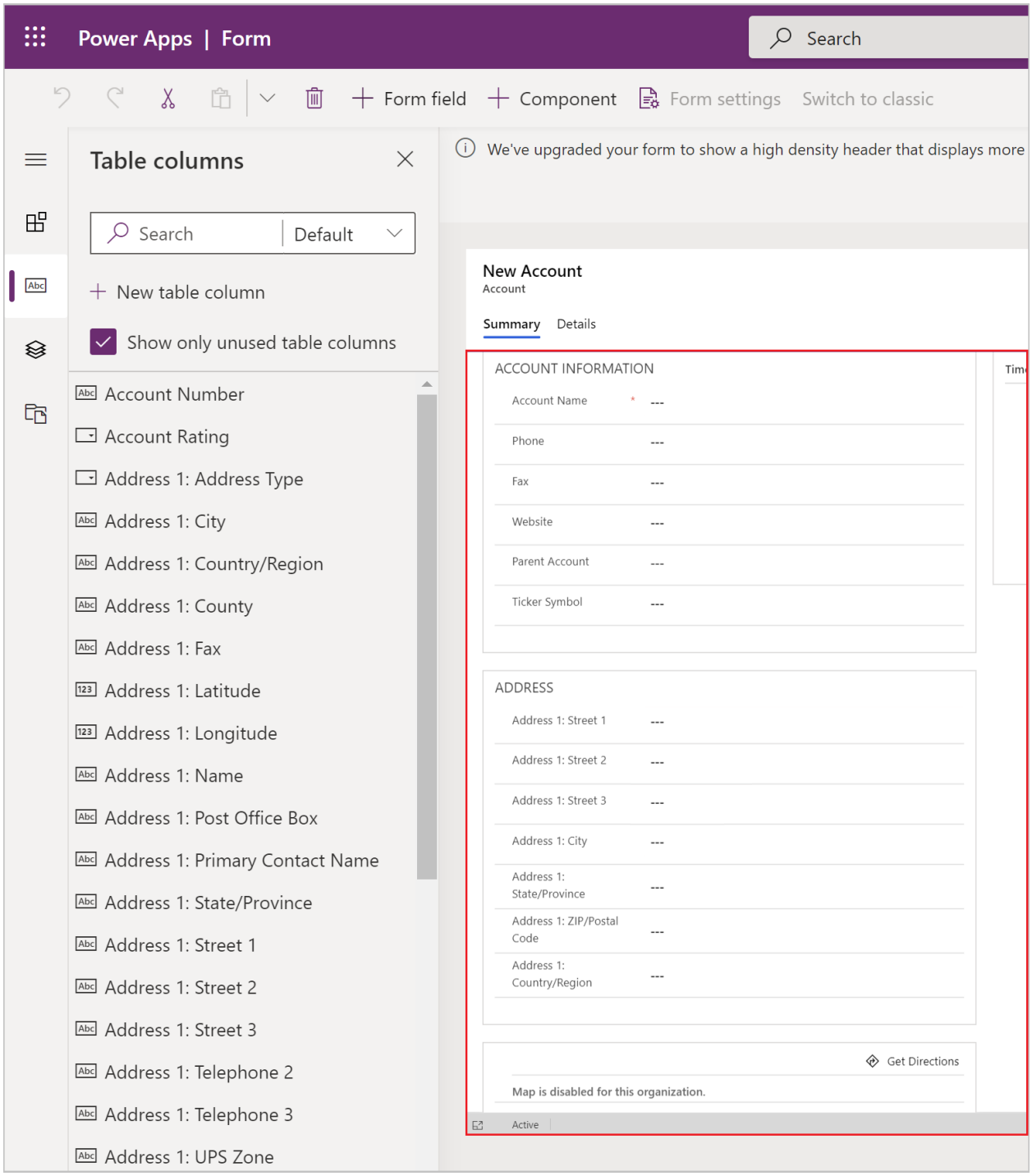


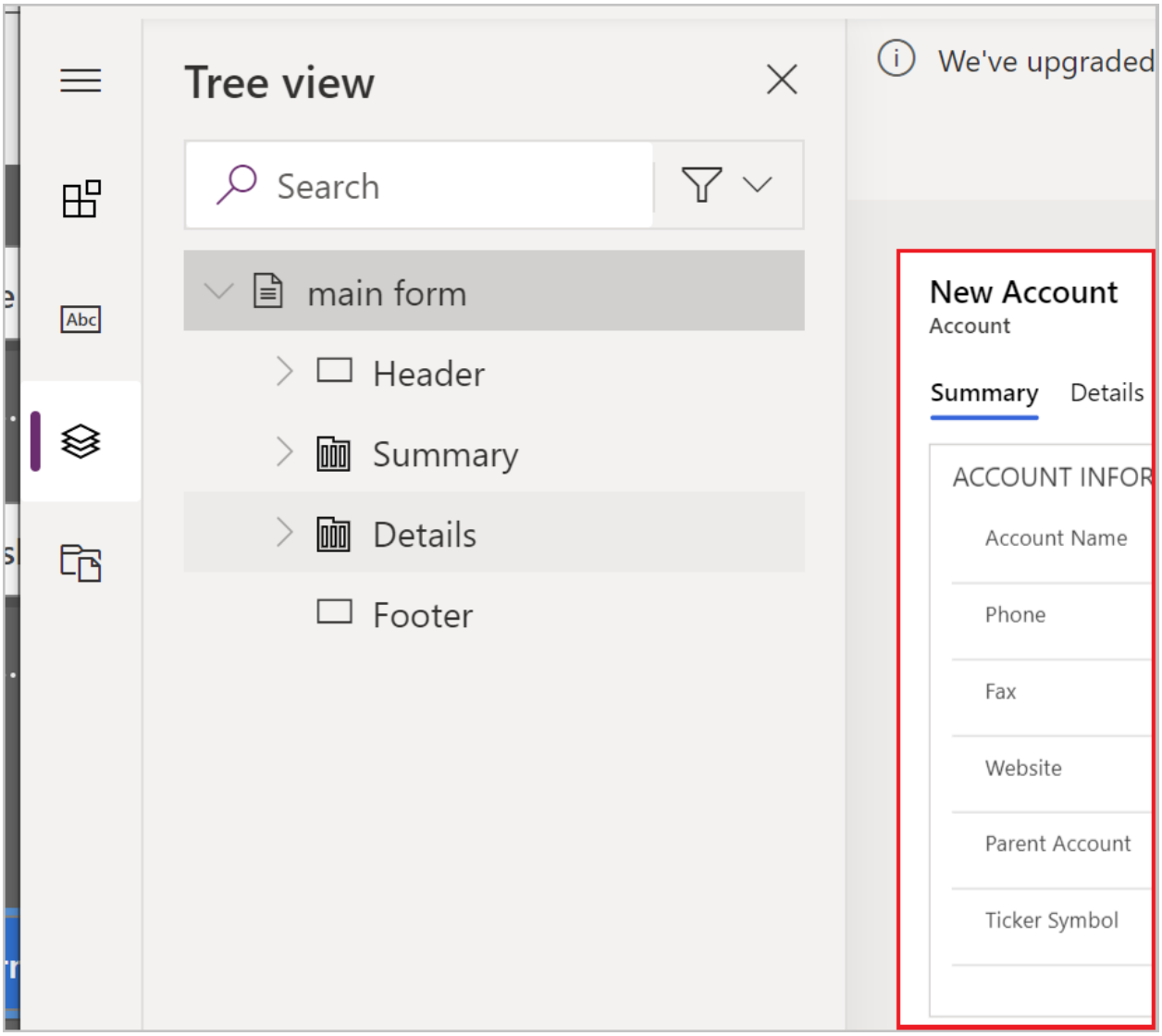
Table columns

From this tab you can add form fields, also called table columns. More information: Add, configure, move, or delete columns on a form



Tree view

From this tab you can select to navigate to the form header, footer, or any tab or section on the form. This is useful when configuring forms with many tabs. More information: Using the tree view in the form designer



Form libraries

Model-driven forms allow you to add JavaScript libraries to further extend your form behavior.

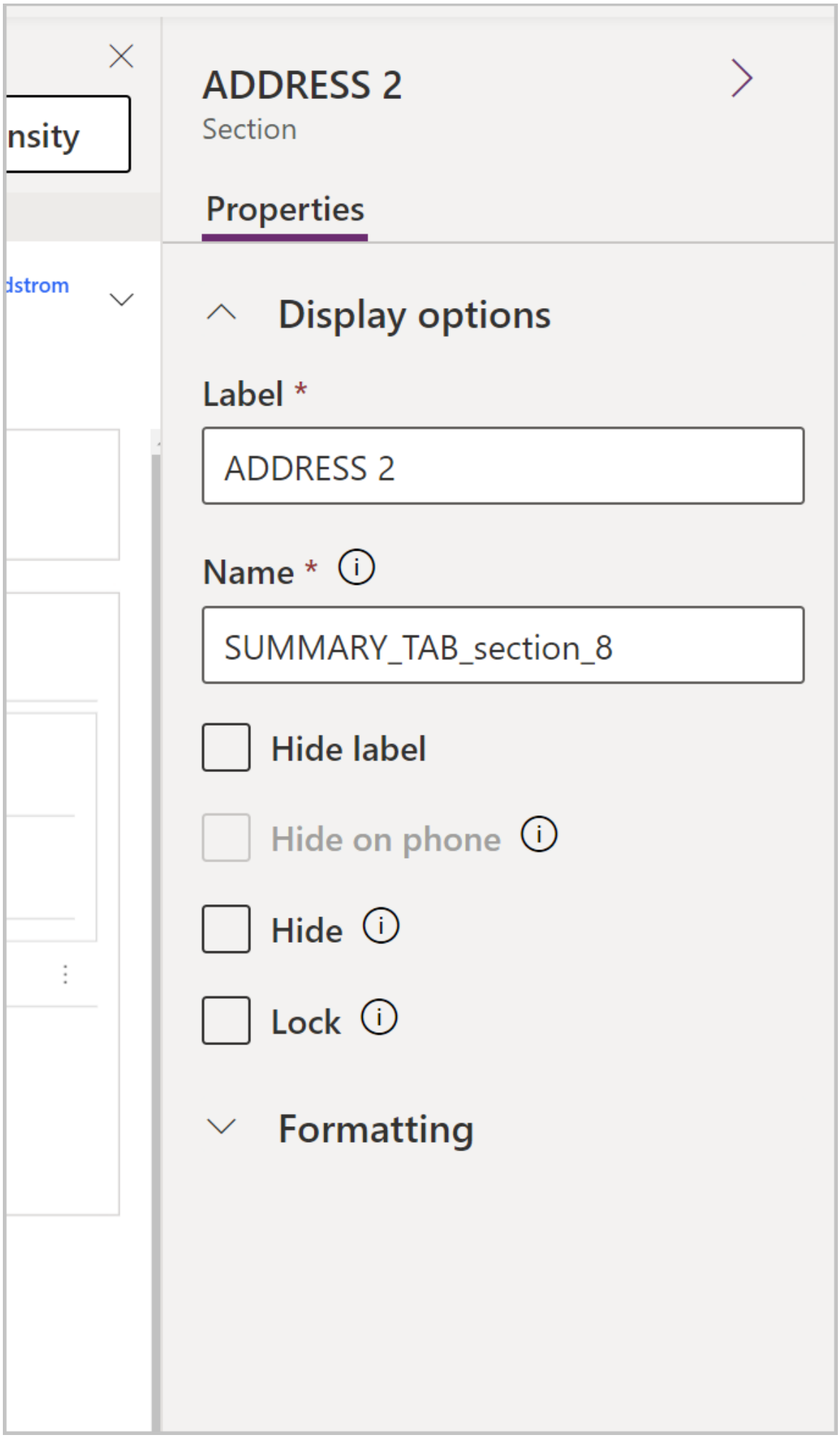
Exercise: Add a new section and fields to the form

By default, the account form includes a single address section. However, the account table also includes a second address set. In cases where you need to track shipping and billing addresses, it's common to add the second address set to the form. Follow these steps to add a new **ADDRESS 2** section and associated columns to the form.

1. Select the **Address** section of the form in the form designer.

2. On the left pane of the form designer select the **Components** tab, and then select **1 Column Section**. A new section is added below the **ADDRESS** section.

3. In the right **Properties** pane, change the **Label** from **New Section** to *ADDRESS 2*.

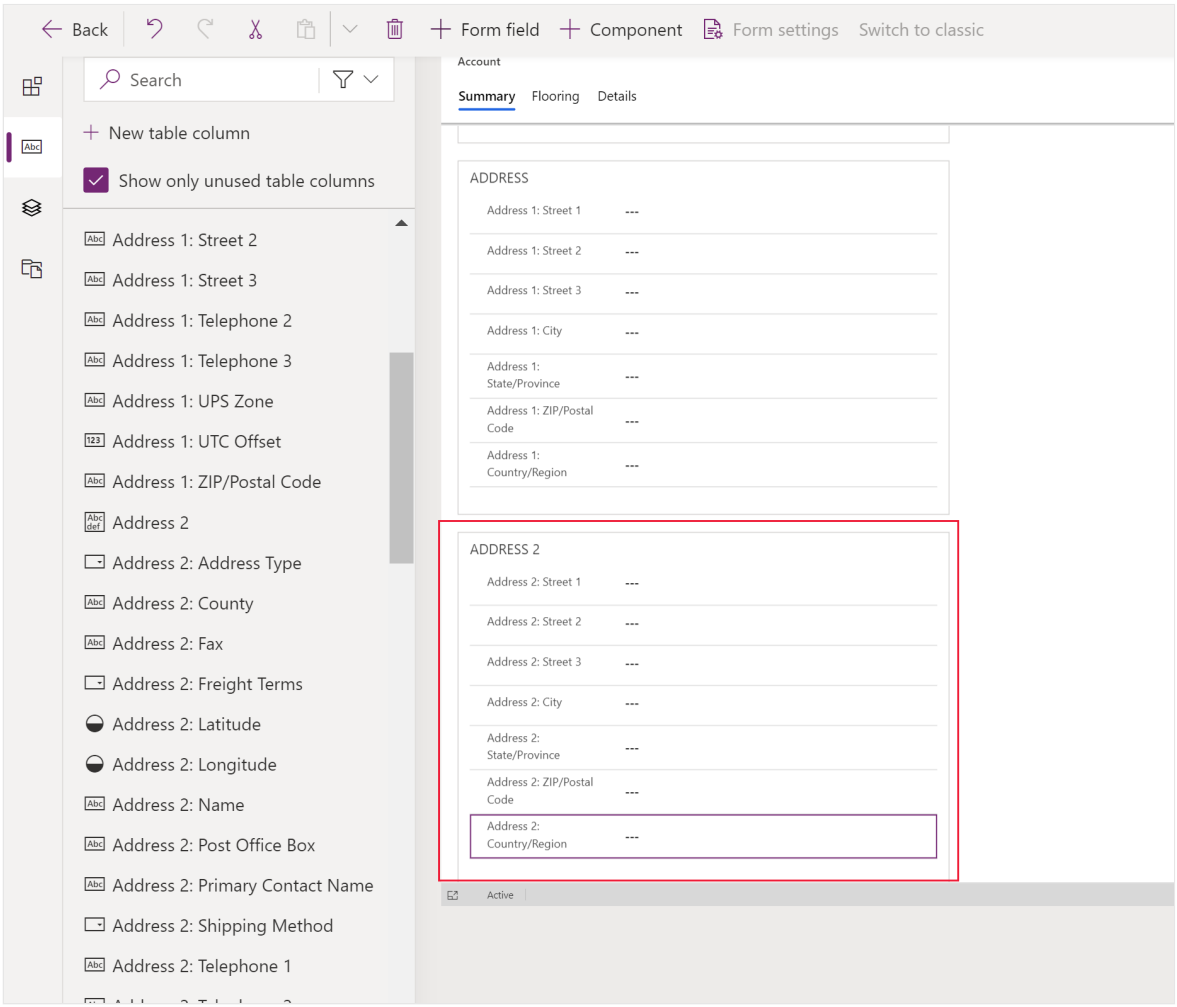


4. On left pane, select the **Table columns** tab.

5. With the *ADDRESS 2* section still selected, select **Address 2: Street 1** from the table column list to add the field to the **ADDRESS 2** section.

6. Repeat the previous step to add the following fields:

    - **Address 2: Street 2**
    - **Address 2: Street 3**
    - **Address 2: City**
    - **Address 2: State/Province**
    - **Address 2: Zip/Postal Code**
    - **Address 2: Country/Region**



> **Note**
>
> You can also drag and drop fields. This is especially useful when you want to move columns to a different area or section.

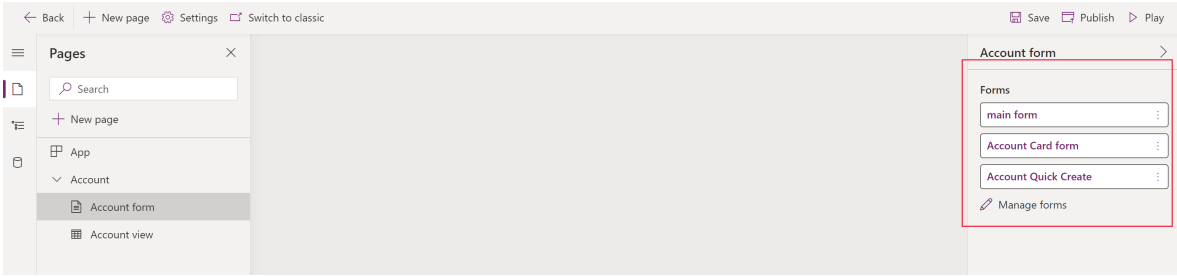
7. On the form designer command bar select **Save**, and then select **Publish**.

8. Close the form designer tab in your browser.

Control which form is visible in your model-driven app

If your app includes multiple model-driven main forms, users can select which form to view. If you want to restrict users of your app to only view records with your newly configured form, follow these steps.

1. Sign in to Power Apps.
2. Go to **Apps**, select **...** next to a model-driven app, such as the Account Tracking app you can create following this article: Build your first model-driven app, and then select **Edit**.
3. In app designer, select a form, such as the account main form.
4. On the right pane, select **Manage forms**.
5. Select the forms that you want to display in the app, such as **main form**, **Account Card form**, and **Account Quick Create**. Clear the other forms, such as **Account**.



6. Select **Save**, and then select **Publish** to reflect the changes in the app.

Controlling access to your forms

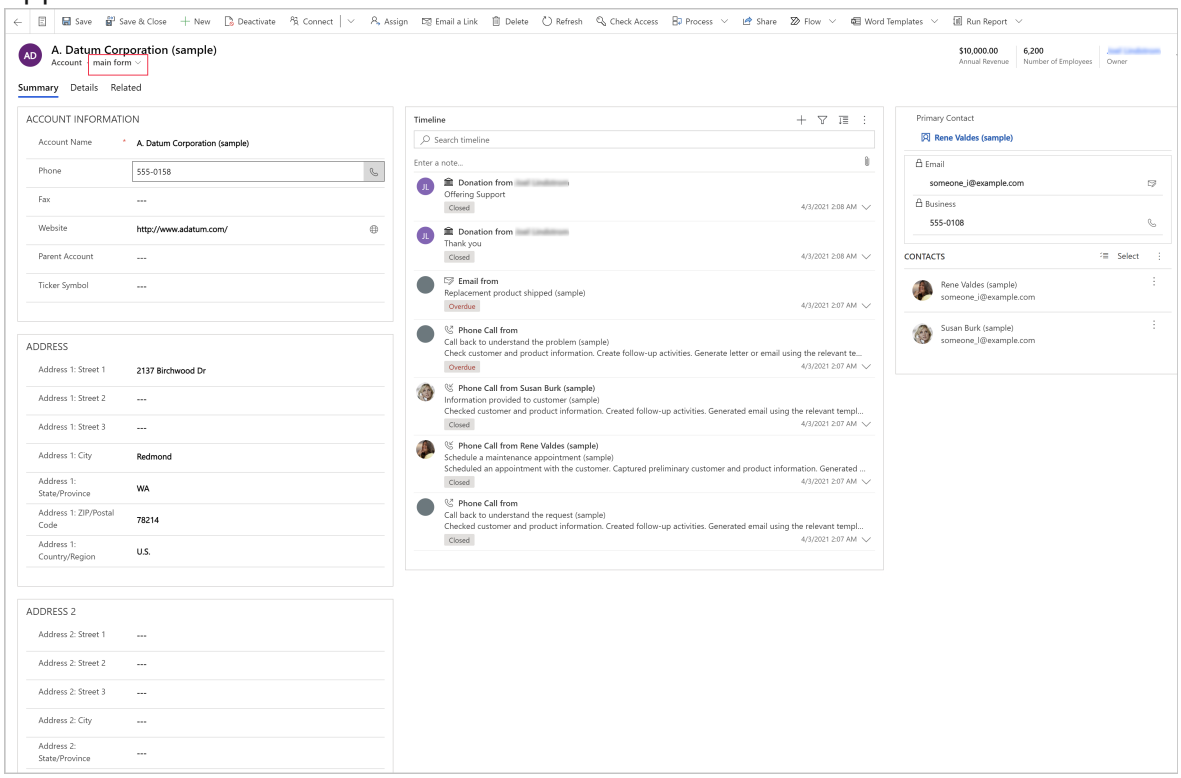
We recommend that you select all quick view forms and quick create type forms, unless you have specific reasons to not check them. Only select the main form or forms that you want users to see. You must also ensure that your new form is enabled for a security role that users of the app have been assigned. You can further control how users will access forms in an app by configuring the form order and fallback form. For more information, see these articles:

- Set security roles for a form
- Set the form order
- Set the fallback form for a table

View your new form

Now that you have created and modified a new form, let's test it out. Follow these steps to view your new form:

1. In app designer, select **Play**.
2. Open an account record.
3. The new main form with the section for **ADDRESS 2** is displayed. If your new account **main form** isn't displayed, select **main form** from the form selector in the upper left of the form.



Next steps

Create or edit quick create forms

Create or edit model-driven app quick create forms

Article • 01/23/2025

With quick create forms, an app can have a streamlined data entry experience with full support for logic defined by form scripts and business rules. In a Power Apps model-driven app, quick create forms appear when the **Create** button is selected in the navigation bar or when **+ New** is chosen when creating a new row from a **lookup** or **subgrid**.

Model-driven apps and Dynamics 365 mobile apps use quick create forms for creating new rows. If a table already has a quick create form configured for it, the mobile apps use that form. If a table doesn't have a configured quick create form, Power Apps generates a quick create form for creating rows in the mobile apps based on the main form definition.

Here's the quick create form for the standard Microsoft Dataverse account table.

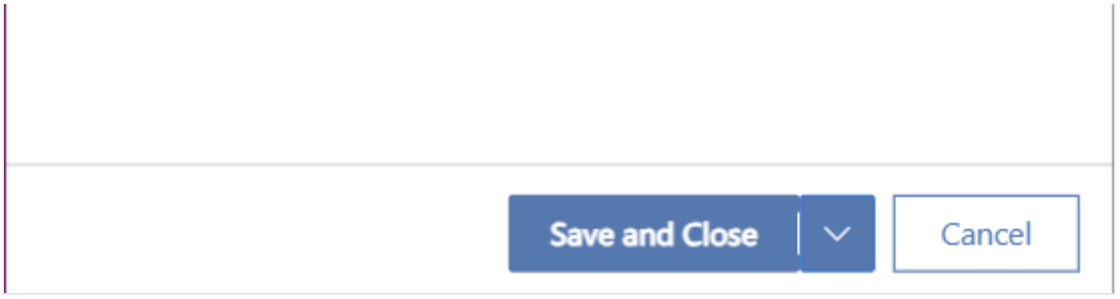
Quick Create: Account

Details

Account Name	*	---
Main Phone		---
Primary Contact		---
Annual Revenue		---
Number of Employees		---
Description		---

Address

Street 1	---
Street 2	---
City	---
ZIP/Postal Code	---



Tables with quick create forms

By default only these system tables have quick create forms: account, [1]campaign response, [1]case, [1]competitor, contact, [1]lead, [1]opportunity.

[1]Requires a customer engagement app.

Although it's possible to create quick create forms for system activity tables (except for the appointment table), system activity tables don't support quick create forms. Currently, the option to disable the quick create form for the appointment table isn't supported. Any of the other updated tables and any custom tables can be enabled to support these forms by selecting **Enable quick create forms** in the table definition and creating a quick create form for the table.

Custom activity tables that support quick create forms can be enabled, and it's possible to create quick create forms for those tables. However, the quick create form for custom activity tables aren't used when app users select **Create** on the navigation bar. These quick create forms can be used only when users add a new row for a subgrid that displays that specific custom activity table.

Create a quick create form

Although you can define multiple quick create forms, only one quick create form can be used by everyone. The form everyone uses is set using the form order. Quick create forms can't be assigned to security roles and they don't provide the capability for the user to switch forms.

> **Note**
>
> - The table must have the **Enable quick create forms** option enabled for the quick create form to be available as an option for the table.
> - The table must be added to your app.

- Some columns, such as the `CREATEDON` column, aren't available to add to a quick create form.

How to create a quick create form

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Tables**, select the table, and then select the **Forms** area.
4. On the toolbar, select **Add form** > **Quick create form**.
5. In the form designer, drag any columns from the **Table columns** left pane into the sections on the form.
6. When you're finished, select **Save and publish** to display the new form in the application.

Edit a quick create form

While quick create forms support form scripts and business rules, their purpose is different from main forms and they don't support all the capabilities of main forms. Quick create forms always have one section with three columns. You can't add more sections or columns.

The following controls can't be added to quick create forms:

- Subgrids
- Quick view forms
- Web resources
- iFrames
- Notes
- Bing Maps

If you add a composite column to a quick create form, it's displayed as separate columns.

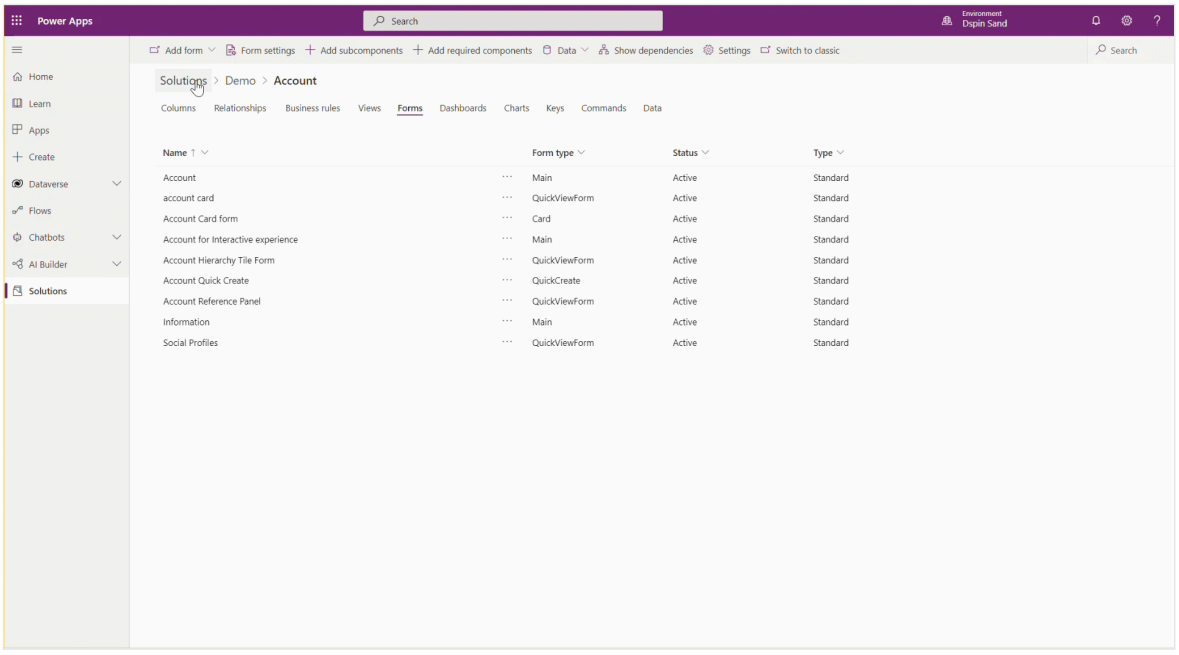
To edit a quick create form

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you

want.

3. Open the table, and then select the **Forms** area.
4. In the form list, select a form where the form **Type** is **Quick create**.
5. Drag any columns from the **Table columns** left pane into the sections in the form.

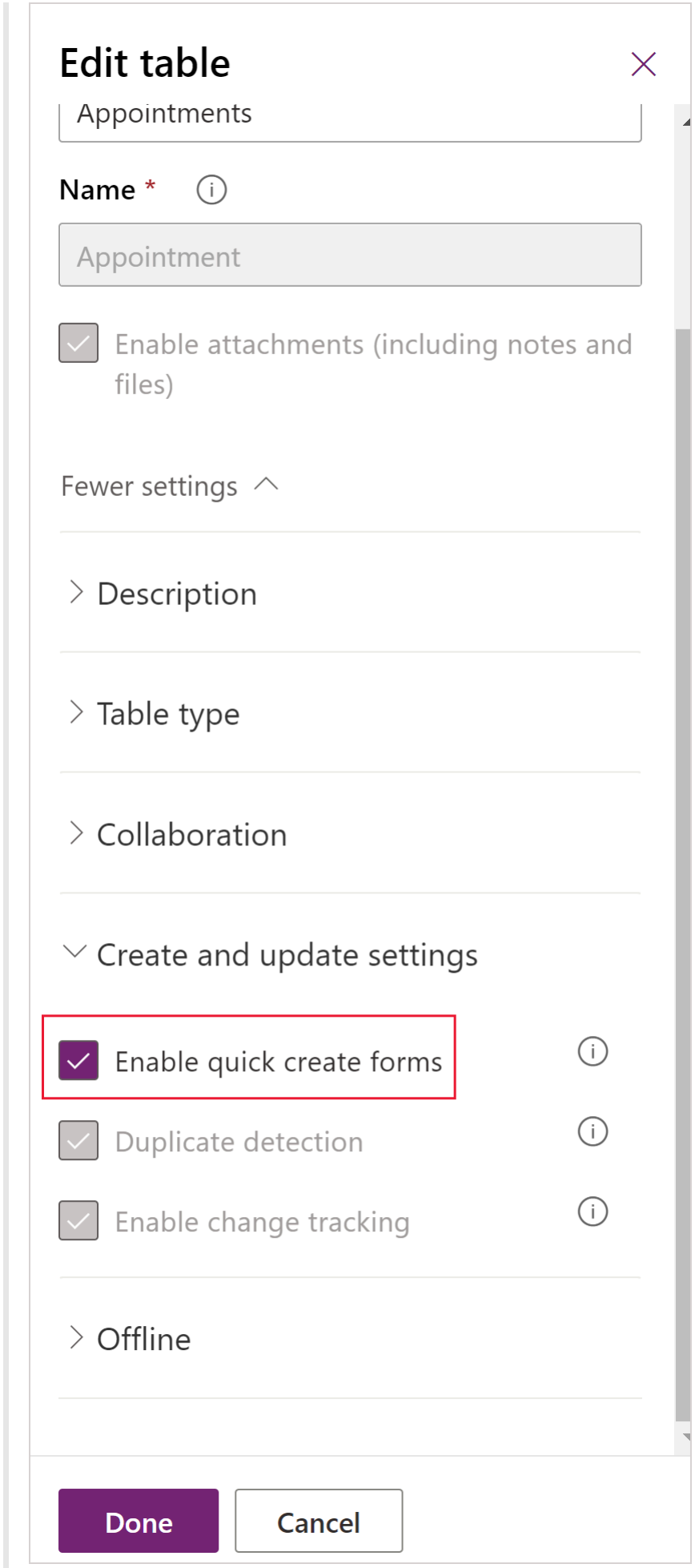
   See Configure event handlers for information about editing event handlers for form scripts.



6. When you're finished, select **Save and publish** to display the modified form in the application.

Enable quick create form property behavior for activities

The **Enable quick create forms** property can be enabled or disabled for all standard activities except recurring appointments. This property allows the form that is displayed by default for most activities to display. By default, the **Enable quick create forms** property is enabled and the quick create form is the form displayed in the app areas and activity tables that support it.



Unified Interface client form display behavior

The following table indicates what form is displayed by default when the **Enable quick create forms** property is *enabled* in the Unified Interface client.

Expand table

Location where form is accessed	Form displayed
Specific activity associated grid	Quick create
Specific activity sub grid	Quick create
Activities (`activitypointer`) grid	Quick create
Activities (`activitypointer`) associated grid	Quick create
Activities (`activitypointer`) sub grid	Quick create
Global command bar + button [1]	Quick create
Timeline wall	Quick create
Activities (`activitypointer`) grid	Main
Specific activity grid	Main

[1]Activities appear in the global **Create** or **+ New** buttons when the **Enable quick create forms** property is enabled. In this case, the quick create form is used if it exists or the main form if it doesn't. If **Enable quick create forms** is disabled, the entry for the table won't appear.

Classic web client form display behavior

The following table indicates what form is displayed by default when the **Enable quick create forms** property is *enabled* in the classic web client.

Expand table

Location where form is accessed	Form displayed
Specific activity associated grid	Quick create
Specific activity sub grid	Quick create
Activities (`activitypointer`) grid	Main
Activities (`activitypointer`) associated grid	Main
Activities (`activitypointer`) sub grid	Main

Location where form is accessed	Form displayed
Global command bar + button	Main
Specific activity grid	Main

Classic web client social pane behavior

The social pane is a special case because it doesn't use the **Enable quick create forms** property but uses different forms for different activity tables as indicated here.

Expand table

Activity	Form displayed
Task	Quick create
Phone Call	Quick create
Email	Main
Appointment	Main
Custom activity	Main

Solution import Allow Quick Create value behavior

When a solution from version 8.2 is imported, regardless of the value of the **Enable quick create forms** property in the solution, the following tables are reset to the default form display value. Also, the main form displays: task, phone call, email, and appointment. In this situation, you'll need to reset the **Enable quick create forms** option back to *enabled* for those activity tables after the import.

If there's a customization made in a version 9.0 solution to tables where **Enable quick create forms** is enabled, the value won't change after import. However, if the **Enable quick create forms** option is set to *disabled* for the task, phone call, email, and appointment tables, the value is overwritten to enabled. In this situation, the **Enable quick create forms** option needs to be reset back to disabled for those activity tables after the import.

Important

- If a quick create form exists for a table but isn't included in the app, the Unified Interface runtime will still open the quick create form. This behavior

occurs with lookup controls and subgrids. If you don’t want a quick create form to appear in your app you must delete it.

- Be aware of the behavior that occurs when a parent relationship is created with the same table. For example, if Account has a relationship to Account and lookup is created that is used by a quick create form that creates a parent row, the first row won't be saved with the lookup that has the parent row value. This is because of the circular reference introduced by using the same table. If you experience this issue, you can resolve it by removing the parent row ID on the quick create form before saving the row.

Next steps

Create quick view form

Form types overview

Feedback

Was this page helpful? Yes No

Provide product feedback

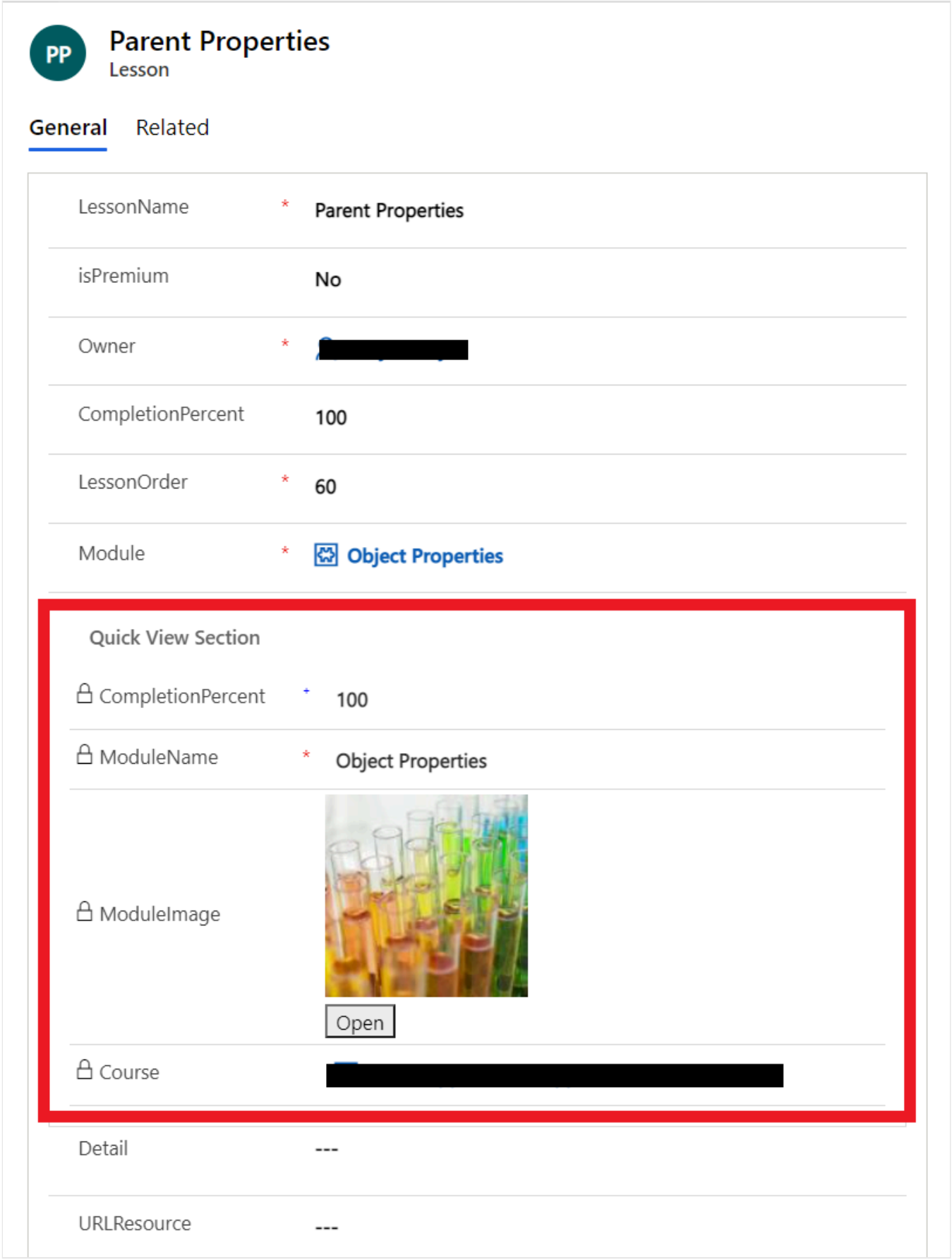
Create a quick view form to view information about a related table

This article describes how to create a quick view form and how to add a quick view control to a main form.

You can add a quick view form to another form as a quick view control. It provides a template to view information about a related table row within a form for another table row. This means app users don't need to navigate to a different row to see the information they need to do their work.

Note that while a quick view form can appear over a view or a form, it is authored as a form, rather than a view.

In the example here, notice the module that relates to a given lesson within the quick view form. A one-to-many relationship exists between the two tables for this to be possible. You can see additional columns of metadata, including the image associated with the module. The form then continues with the metadata associated with the lesson table record.



Quick view controls are associated with a lookup column that is included in a form. If the lookup column value isn't set, the quick view control is not visible.

> **Note**

- You can't edit data in quick view form controls and they don't support form scripts.
- Because quick view forms are viewed using a quick view control in a form, they don't include header, footer, or navigation areas.
- Security roles can't be assigned to quick view forms and they can't be activated or deactivated.
- Custom controls aren't supported in quick view forms, and there is limited customization support for complex controls such as subgrids. If this is required, consider using a Form component control

Create a quick view form

You create quick view forms by using the form editor, similar to how you create other forms. Quick view forms are read-only. Use them to create forms that are for reading purposes only.

1. Sign in to Power Apps and then select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the table where you want to create a quick view form, select the table that you want, and then select the **Forms** area.
3. On the toolbar, select **New form** > **Quick view form**.
4. Enter a **Form name** and **Form description** to differentiate this quick view form from others.
5. Optionally, clear **Get AI-generated column suggestions** if you don't want suggestions from Copilot for the columns to create for the form based on the **Form name** and **Form description** values.
6. Select **Create**.
7. The form designer opens. Drag columns you want to add from the left **Table columns** pane into the form canvas. To remove columns you don't want, select the column, and then select **Delete** on the command bar.

   > **Important**
   >
   > Required columns can't be removed from a form. If you add a required column to the form and want to remove it, you must delete the form and then re-create it. When you set the required property for a column, a row can't be saved without data in the column.

8. Select **Save and publish** to make your changes available to users of the apps that use the form.

Edit a quick view form

Quick view forms have a simplified layout because they are designed to be viewed within a form section. Only one single column tab is available with quick view forms. You can add only additional single column sections, columns, subgrids, and spacers.

1. Sign in to Power Apps.
2. Open the solution that has the table where you want to edit the form, and open the table you want, and then select the **Forms** area.
3. Open the quick view form.
4. Add or remove the table columns you want.

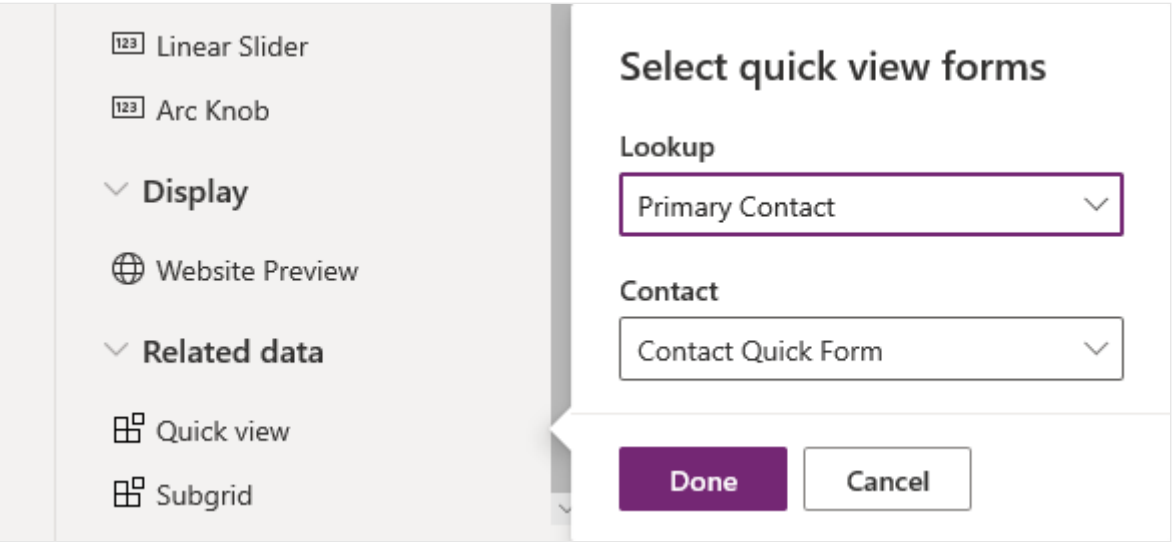
   > **Important**
   >
   > You can't delete required columns. If you add a required column to the form, you can't delete it. If you don't want the column in the form, you must delete the form and then re-create it.

5. When you're done editing a quick view form, select **Save and publish** to publish your changes so they're visible in the application.

Add a quick view control to a main form

You can add quick view forms to a main form only when a lookup column exists that targets the table of the quick view form.

1. Sign in to Power Apps.
2. Open the solution that has the table where you want to edit the form, and open the table you want, and then select the **Forms** area.
3. Open a form, which **Type** is **Main**
4. In the form designer, select **Quick view** from the **Components** pane.
5. In the **Select quick view forms** dialog box, select the **Lookup** column, and then select the lookup column value. More information: Quick view control properties.



6. Select **Done** to close the **Select quick View forms** dialog box. The quick view form appears on the form.

7. To save and publish the form select **Save and publish**.

> **Important**
>
> Be aware of the behavior that occurs when you create a parent relationship with the same table. For example, if Account has a relationship to Account and a lookup is created that is used by a quick create form that creates a parent row, the first row will not be saved with the lookup that has the parent row value. This is because of the circular reference introduced by using the same table. If this issue is experienced it can be resolved by removing the parent row ID on the quick create form before saving the row.

Quick view form properties

When the quick view form is used in a main form the following properties can be set or updated.

Expand table

Property	Description
Name	**Required**: Enter a unique name for the quick view form. Use this name when referencing the form in scripts.
Label	**Required**: Enter a label to display for the quick view form.
Display label on the Form	Choose whether to display the label on the form.

Property	Description
Lookup Column	Choose one of the lookup columns included in the form.
Related table	This value depends on the **Lookup Column** chosen. It's usually the primary table for the 1:N table relationship for the lookup. If the table includes a **Potential Customer** lookup that can accept either an account or contact, in the **Quick View Form** column you can choose a quick view form for both account and contact by changing this value and then choosing another quick view form.
Quick View Form	If the **Related table** has any quick view forms these can be selected here. Otherwise, select **New** to create one.

Next steps

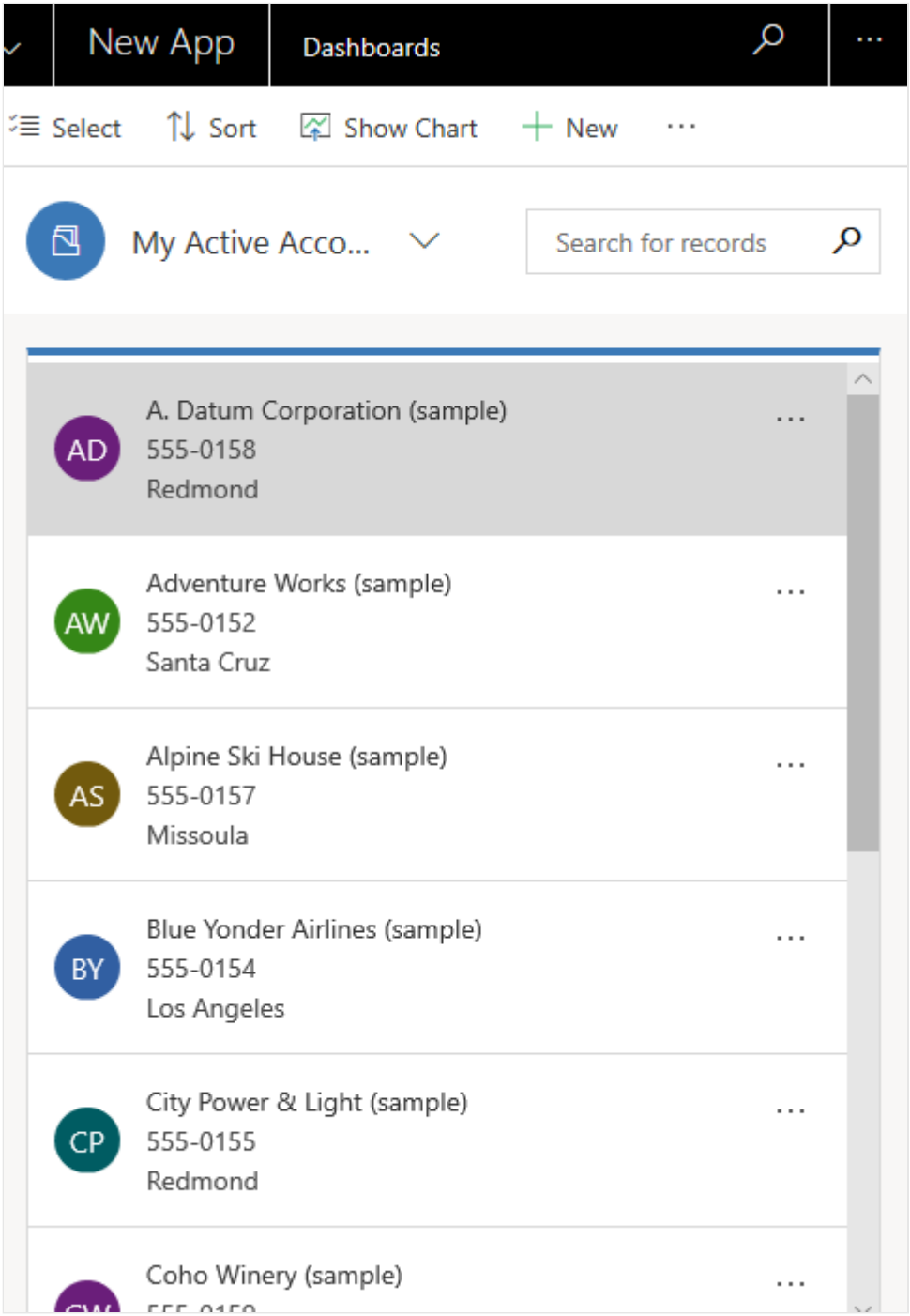
Create and edit a card form

Last updated on 03/31/2026

Create a card form

Article • 03/13/2023

Card forms are used in views for model-driven apps. Card forms are designed to present information in a compact format that is suitable for **mobile devices**. For example, the default card form for the My Active Accounts view defines the information displayed for each account row.



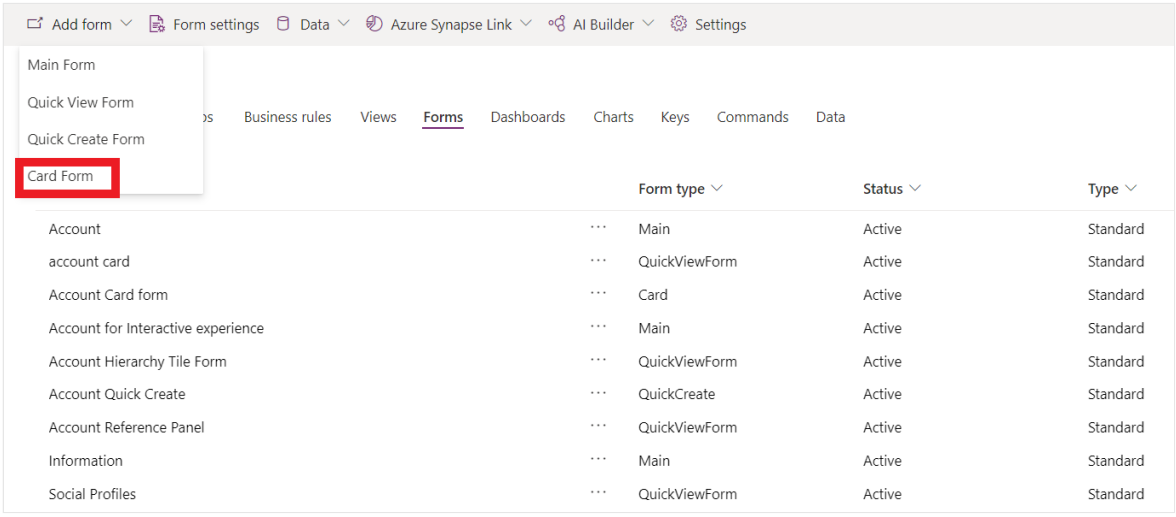
Although card forms can be created and edited in the same way as other form types, card forms are added to apps differently. Instead of adding a form as an app component, custom card forms are added to views by using the **Read-only Grid** control.

Create a card form

1. To create a card form, sign in to Power Apps.

2. Select **Tables**, select the table you want, and then select the **Forms** area. If the item isn't in the left navigation pane, select ...More and then select the item you want.

3. On the toolbar select **Add form**, and then select **Card form**. Alternatively, you can open an existing **Form type** that's a **Card** form to edit it.



4. Add the columns that you want. We recommend that you limit the number of columns so that the form displays well on small screens.

5. Select **Save**, and then select **Publish**.

Add a card form to a view

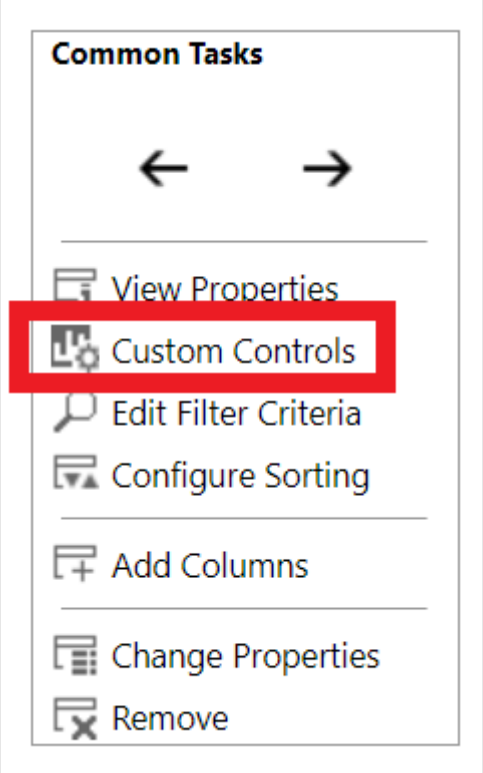
> **Important**
>
> To add a custom card form to a view requires the deprecated **Read-only Grid** control. The **Read-only Grid** has been deprecated so it is no longer the default grid control for tables in Dataverse. However, while the control remains available you can select it for a table to use to create a card form view. More information: **Configure a grid control for a table**

1. Sign in to Power Apps.

2. Select **Tables**, select the table you want, and then select the **Views** area. If the item isn't in the left navigation pane, select ...More and then select the item you want.

3. Select the view that you want, and then on the view designer toolbar, select **Switch to classic**.

4. Select **Custom Controls** from the **Common Tasks** pane.

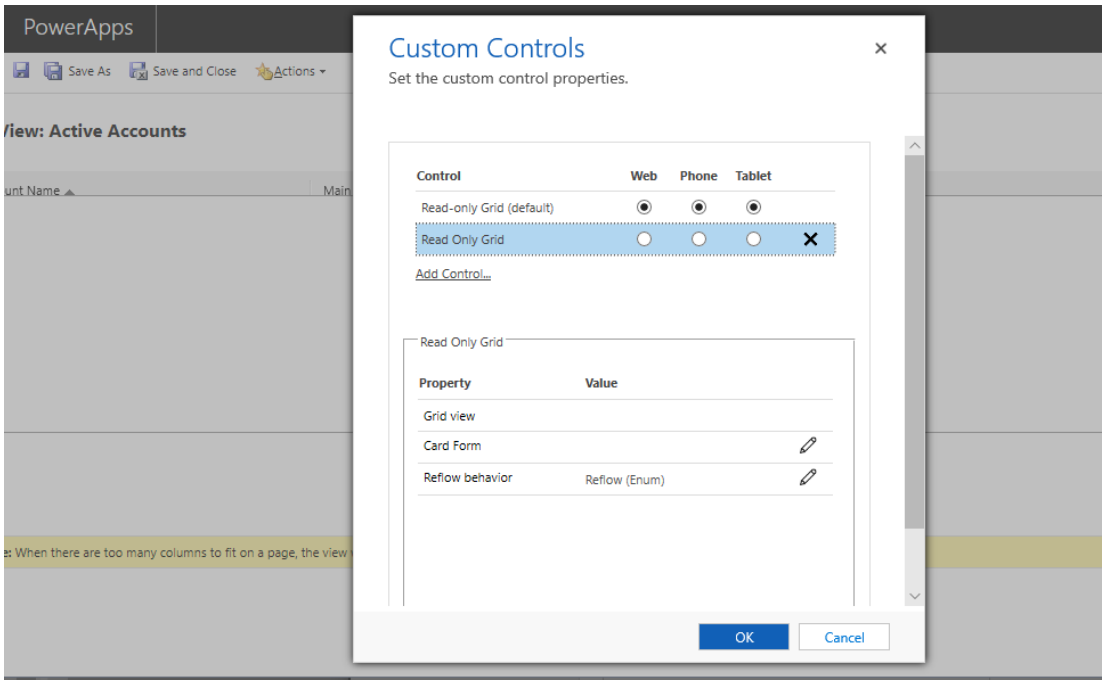


5. Select **Add Control**, from the list of controls select **Read Only Grid**, and then select **Add**.

> ⓘ **Note**
>
> There are two read only grid controls. The default read only grid control that is named **Read-only Grid (default)** doesn't support custom card forms.

6. From the **Read Only Grid** properties page configure the following properties, and then select **OK**.

   - **Card Form**. Select the pencil icon and then select the card form you want to display in the view. By default, the primary table associated with the view is already selected, but you can change it.
   - **Reflow behavior**. If you want to change whether the card form displays when resized, select the pencil icon . More information: Allow grid to reflow into list
   - Select the client types, **Web**, **Phone**, or **Tablet**, where you want the **Read Only Grid** control to display.



7. Select **OK** to close the **Custom Controls** properties page.

8. On the classic view designer toolbar, select **Save and Close**.

See also

Form types overview

Assign model-driven app form order

Summarize this article for me

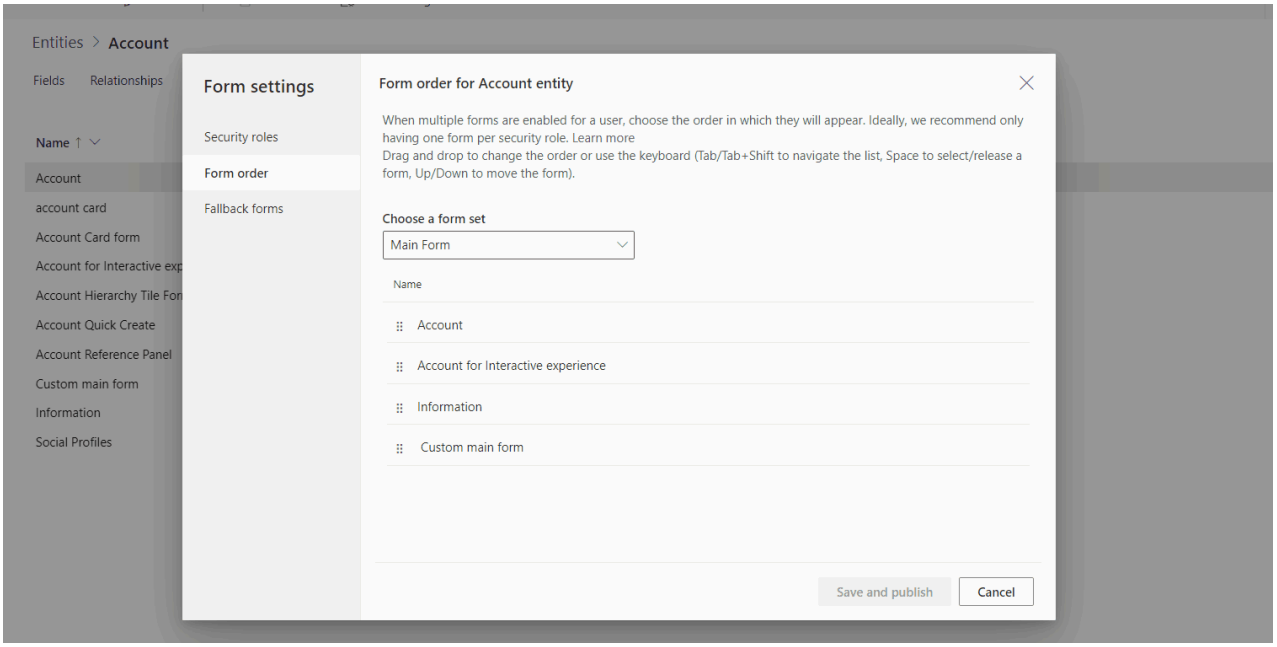
When there are multiple main, quick create, quick view, or card forms for a table, a form order can be assigned. The form order determines which of the available forms are shown by default.

The available main forms can be further controlled by assigning security roles to the form. More information: Control access to forms

It isn't possible to assign security roles to quick create, quick view, or card forms, so the only form that is used by everyone is the one at the top of the form order.

Set the form order

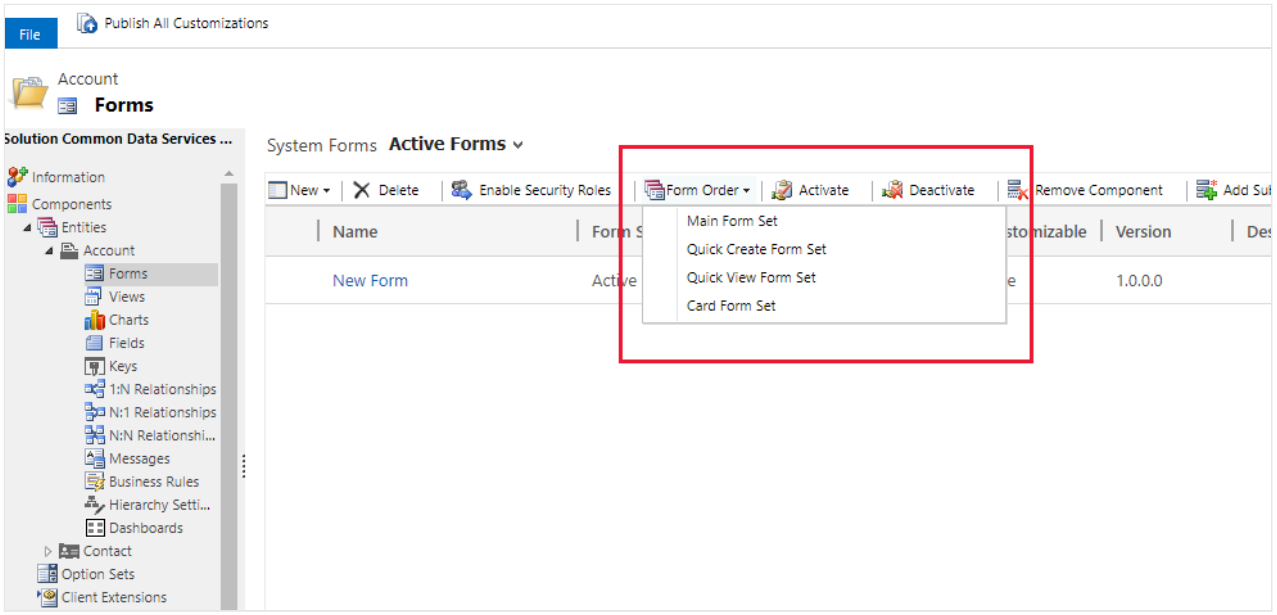
1. Sign into Power Apps, select **Solutions**, open the solution you want, select a table on the left navigation pane, and then select the **Forms** area. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Next to the main form you want to configure the fallback form for, select ..., and then select **Form settings**.
3. In the left navigation on the **Form settings** page, select **Form order**.
4. Select the form set that is required, such as **Main Form**, **Quick Create Form**, **Quick View Form**, or **Card Form**.
5. Change the order by dragging and dropping the form up and down the list in the order best suited to the users. It is also possible to use these keyboard shortcuts: Tab/Tab + Shift to move the focus, Space to select or release a form, and Down/Up arrow keys to move a form. Select **Save and publish**.



Using the classic editor to assign the form order

> **Note**
>
> To set form order using the new experience, go to Set the form order.

1. Open the solution explorer, expand the table that you want, and then select **Forms**.
2. In the form list toolbar, select **Form Order**.



3. Choose either **Main Form Set**, **Quick Create Form Set**, **Quick View Form Set**, or **Card Form Set** depending on the type of forms you want to work with. More information: Type of forms.
4. The **Form Order** dialog is a simple list where a selected form can be moved up or down in the form order.
5. After the order has been set, select **OK** to close the dialog.

Next steps

Form designer overview

Last updated on 02/21/2026

Overview of the model-driven form designer

Article • 02/15/2022

The new model-driven form designer provides a modern WYSIWYG authoring experience when you work with model-driven forms. It includes several improvements not available in the classic model-driven form designer.

The designer shows a real-time WYSIWYG preview (Unified Interface only) while you author a form. Changes to the form are instantly reflected in the preview, enabling you to see exactly how the form will appear to users when published. An always available property pane makes the common task of updating properties quick and easy. Updates to properties are also instantly reflected in the form preview. The columns pane with searching and filtering capabilities helps makers quickly find and add columns to the form. The components pane makes it easy to discover and use components and provide a rich end-user experience. The tree view helps visualize the hierarchy of the form's columns and components and to quickly find and select a particular column or component on the form.

The form designer interface has the following areas:

1. Command bar – Displays available actions such as **Save**, **Publish**, **Undo**, and **Redo**.

2. Form preview – Displays a real-time preview of the form as it will appear to users when published.

3. Panes – Displays panes with specific functions:

    - **Columns** pane to add columns to a form.
    - **Components** pane to add components to a form.
    - **Tree view** to see the hierarchy of components on the form and select them.
    - **Form libraries** pane is where form libraries can be viewed and added. More information: Configure model-driven app form event handlers

4. Property pane – Displays properties of the selected element, and also allows you to make changes.

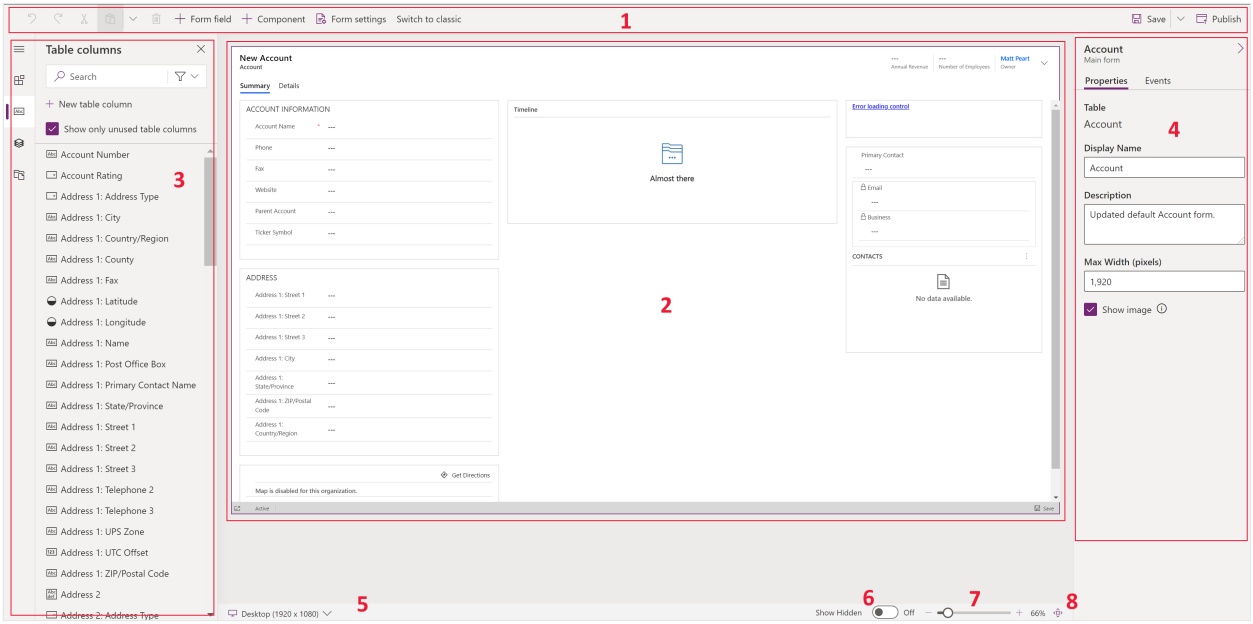
5. Preview size switcher - Changes the size of the form preview helping you to see how the form will appear on various screen sizes.

6. Show hidden - Displays hidden columns in the form preview area. By default, this option is turned off and hidden columns don't appear on the form preview and are

visible only from the tree view pane. When enabled, columns that are hidden are indicated in the form preview area with  .

7. Zoom slider - Zooms in or out of the form preview helping you take a closer look.

8. Fit to width - Quick action to fit the form preview to the available width.



Next steps

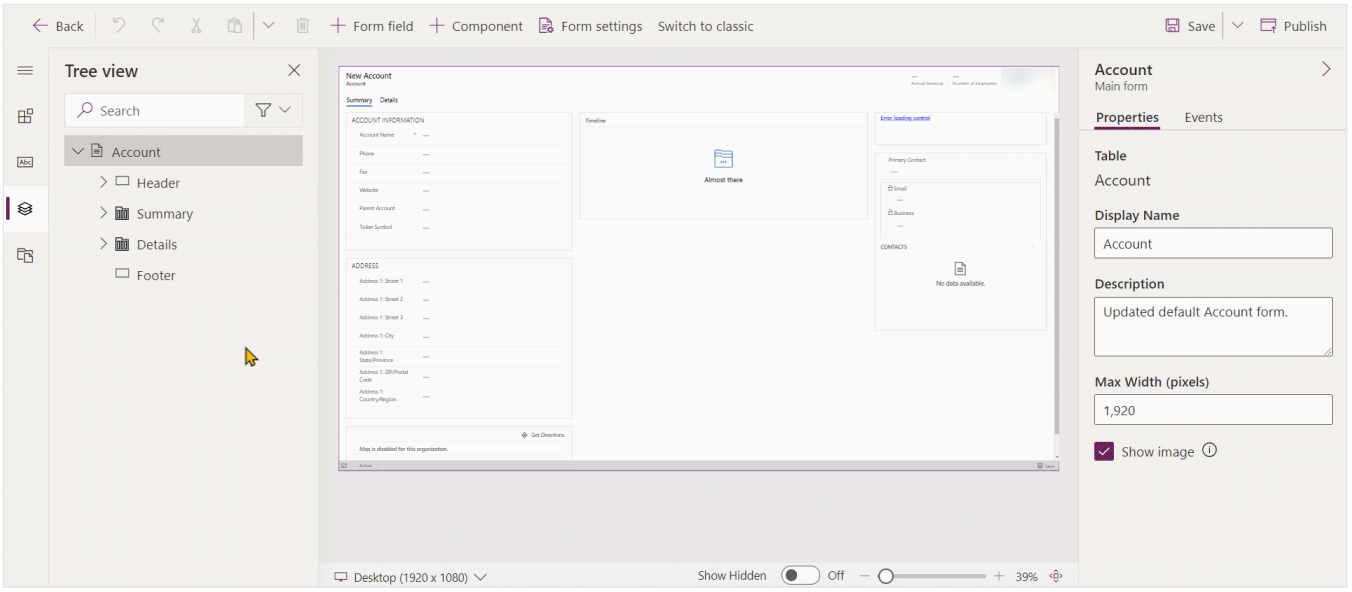
How to open the form editor

Open the model-driven app form editor

Summarize this article for me

Model-driven app forms let users update data within columns, or fields, associated with a given table record.

The form editor is where you design forms by dropping components such as sections, tabs, columns, and controls onto the form editor canvas. In this article, you learn how several different ways to access the form editor.



If you create any new solution components in the process of editing the form, for example web resources, the names of the components will use the solution publisher customization prefix for the default solution and these components will only be included in the default solution. If you want any new solution components to be included in a specific unmanaged solution, you should open the form editor through that unmanaged solution.

Access the form editor from the Power Apps site

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation, and then open the table you want, such as the account table. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the **Forms** area and then open the form you want, such as the **Account** main form.

Access the form editor for an unmanaged solution

1. Select **Solutions**, on the left naviation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the unmanaged solution you want to work with.
3. Open the table with the form you want to edit. If the table isn't there, you'll need to add it. More information: Add a table to an unmanaged solution
4. Select the **Forms** area, and then open the form you want to edit.

Add a table to an unmanaged solution

1. While in an unmanaged solution, on the toolbar above the list, select **Add Existing** > **Table**.
2. In the **Add existing tables** pane, select the table you want to add, and then select **Next**.
3. Choose the correct option for the table objects:
   - If this is a custom table that doesn't exist in other environments, select **Include all objects**.
   - If this table has been customized but already was exported or otherwise exists in target environments, select **Include table metadata** if only properties for the table have changed, or select **Edit objects** and select specific objects, such as columns, forms, or views that have been changed or added.
   - If no changes have been made, such as if you adding an out-of-box table, don't select any options.
4. Select **Add**.

Access the form editor from the classic solution explorer

1. Open solution explorer.
2. Under **Components**, expand **Tables**, and then the table you want, and select **Forms**.
3. In the list of forms, select the form you want to edit.

Next steps

Create or edit forms

Last updated on 02/13/2026

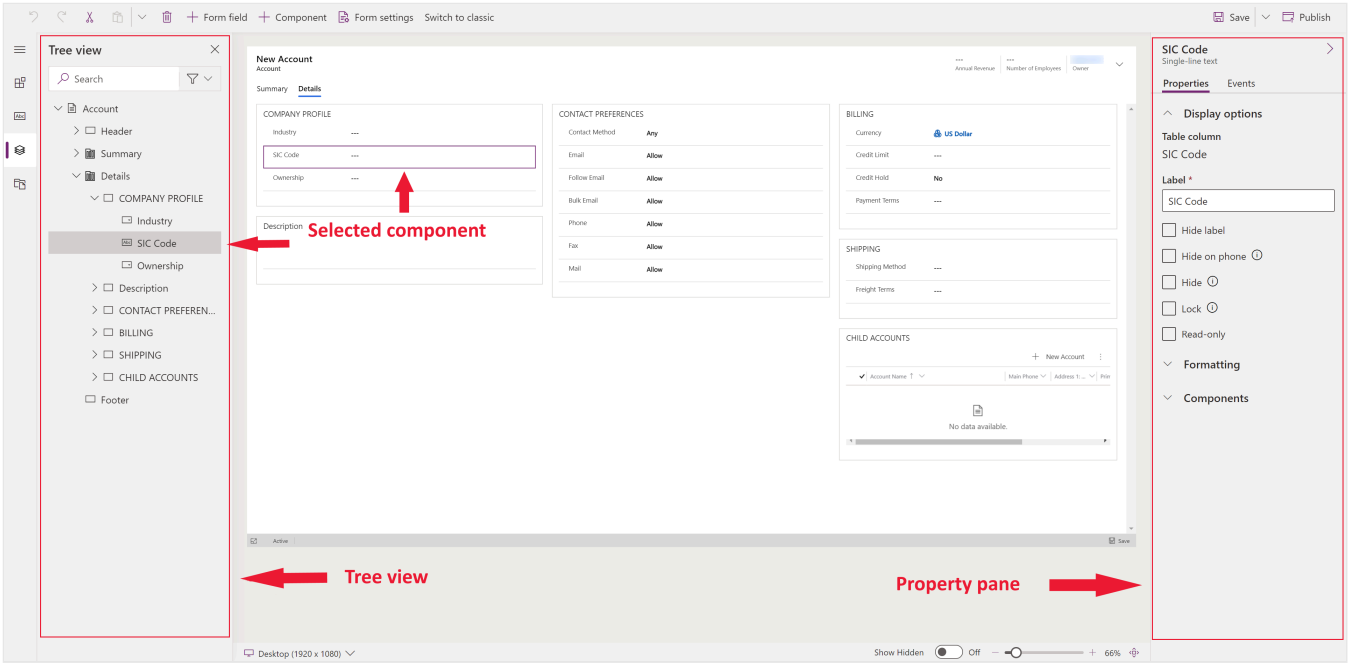
Using the tree view in the form designer

Summarize this article for me

The **Tree view** pane displays a visual hierarchy of the components on the form. The icons in the tree view help an app maker to quickly identify the type of properties of the column, section, tab, or component.

Use the tree view to select columns and other components present on the form. The tree view is helpful when you want to select hidden elements that aren't visible on the form preview.

You can expand or collapse nodes in the tree view to show or hide the elements within a node. When you select an element in the tree view, it becomes highlighted in the form preview, and the property pane displays the properties for the element.



Open the tree view

1. Sign in to Power Apps.
2. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution you want, select the table you want, and then select the **Forms** area.
4. Open the form you want. The form designer opens to enable creation or editing of a form.
5. In the left pane, select **Tree view**.

See also

Overview of the model-driven form designer
Create, edit, or configure forms using the form designer
Add, configure, move, or delete columns on a form
Add, configure, move, or delete components on a form
Add, configure, move, or delete sections on a form
Add, configure, move, or delete tabs on a form
Configure header properties in the form designer
Add and configure a sub-grid component on a form
Add and configure a quick view component on a form
Configure a lookup component on a form
Create and edit columns

Last updated on 03/04/2026

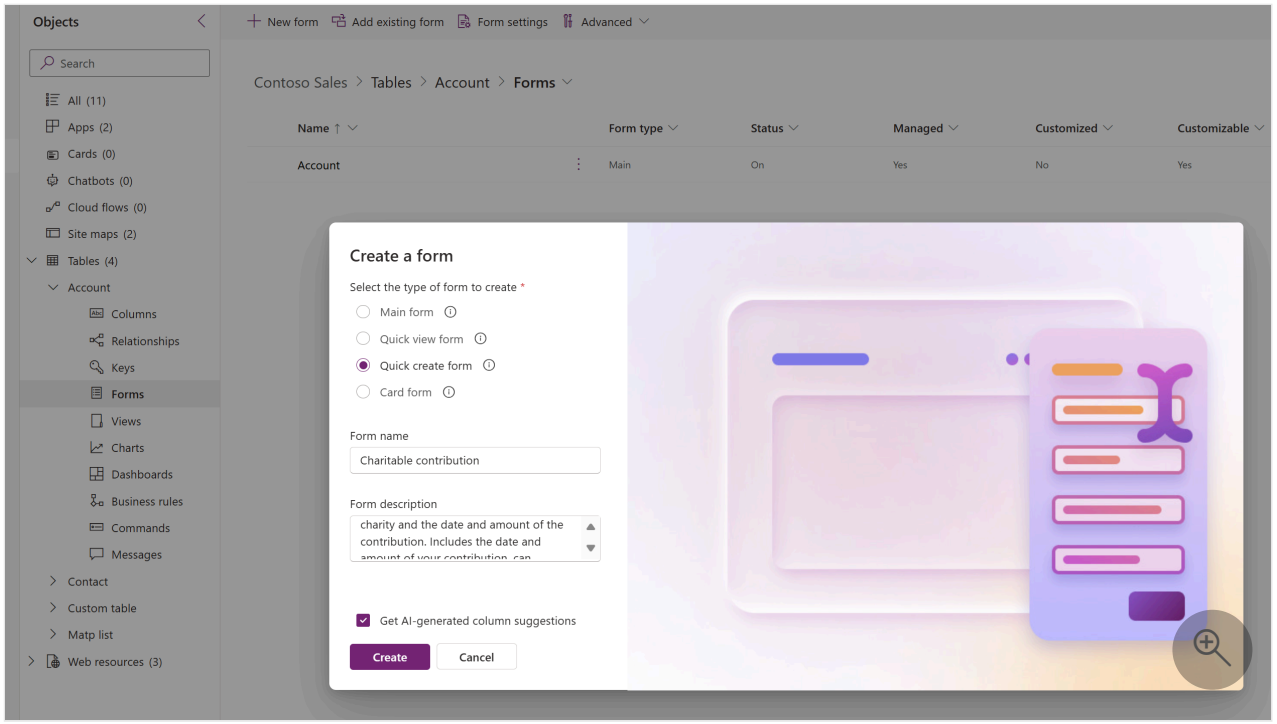
Create, edit, or configure forms using the form designer

Use the form designer to create, edit, or configure forms for model-driven apps.

Create a form

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**, and open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open a table, such as the account table, and then select the **Forms** area.
4. Select **New form**, and then select one of the following form types:
    - **Main form** The primary form type for interaction by users with record data. The contents of the new form are filled using the existing main form definition. If multiple main forms exist, the form at the top of the list in the form order for your app is used to fill the new form.
    - **Quick view form** Quick view forms appear within a main form to display additional read-only data referencing a lookup column. Learn more about main forms
    - **Quick create form** Best for creating new records quickly where only essential columns are required. Appears in a side panel so users don't navigate away from the current screen.
    - **Card form** For displaying compact data in a layout that is good for small screens or areas in a subgrid or view.
5. Enter a **Form name**, and optionally enter a **Description**.
6. If you want to have suggestions from Copilot for the columns to create for the form based on the **Form name** and **Description** values, select **Get AI generated column suggestions**. For more information about using this AI feature, go to Column suggestions

by Copilot.



7. Select **Create**.

Edit a form

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**, and open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open a table, such as the account table, and then select the **Forms** area.
4. Select the form that you want, and then on the command bar, select **Edit**.
5. Make changes such as adding columns or configuring properties.
6. When you're done making changes to the form, select **Save only** to save the form, or select **Save and Publish** to save and have your changes available to app users.

Column suggestions by Copilot

Use column suggestions by Copilot to select the best columns to display in your model-driven app form. Instead of using the default fields that Power Apps selects, makers can view suggestions from Copilot. The column suggestions are based on the form name and description. Makers review the suggestions and make adjustments as needed, saving time and improving the quality of the app.

Prerequisites

- Prerequisites for Copilot in Power Apps features: Copilot in Power Apps overview (preview)
- See if this feature is available in your region: Product availability report
- Learn how to turn on Copilots in your region: Turn on copilots and generative AI features

How column suggestions work

Column suggestions are available when you create a new form or view for a Microsoft Dataverse table.

Column suggestions are available for the following form types:

- Quick view
- Quick create

To understand the AI impact of field suggestions by Copilot in Power Apps, go to FAQs for field suggestions by Copilot.

Form properties

These are the properties available to configure a form when you create or edit a form using the form designer.

Expand table

Name	Description
Title	Enter a name that is meaningful to other makers and app users. This name is shown to app users. If users have access to multiple forms for a table, they will use this name to differentiate between the available forms. This property is required.
Description	Enter a description that explains how the form is different from other main forms. This description is only shown to makers in the list of forms for a table in the solution explorer.
Max Width	Set a maximum width (in pixels) to limit the width of the form. The default value is 1900. This property is required.
Show image	Show the table's **Primary Image** if it has one set. This setting enables showing the image column in the header of the form. More information: Enable or disable table options

See also

Overview of the model-driven form designer
Add, configure, move, or delete columns on a form
Add, configure, move, or delete components on a form
Add, configure, move, or delete sections on a form
Add, configure, move, or delete tabs on a form
Configure header properties in the form designer
Add and configure a subgrid component on a form Learn more about quick create forms Learn more about quick view forms Add and configure a quick view component on a form
Configure a lookup component on a form

Last updated on 01/09/2026

Add, move, configure, or delete columns on a form

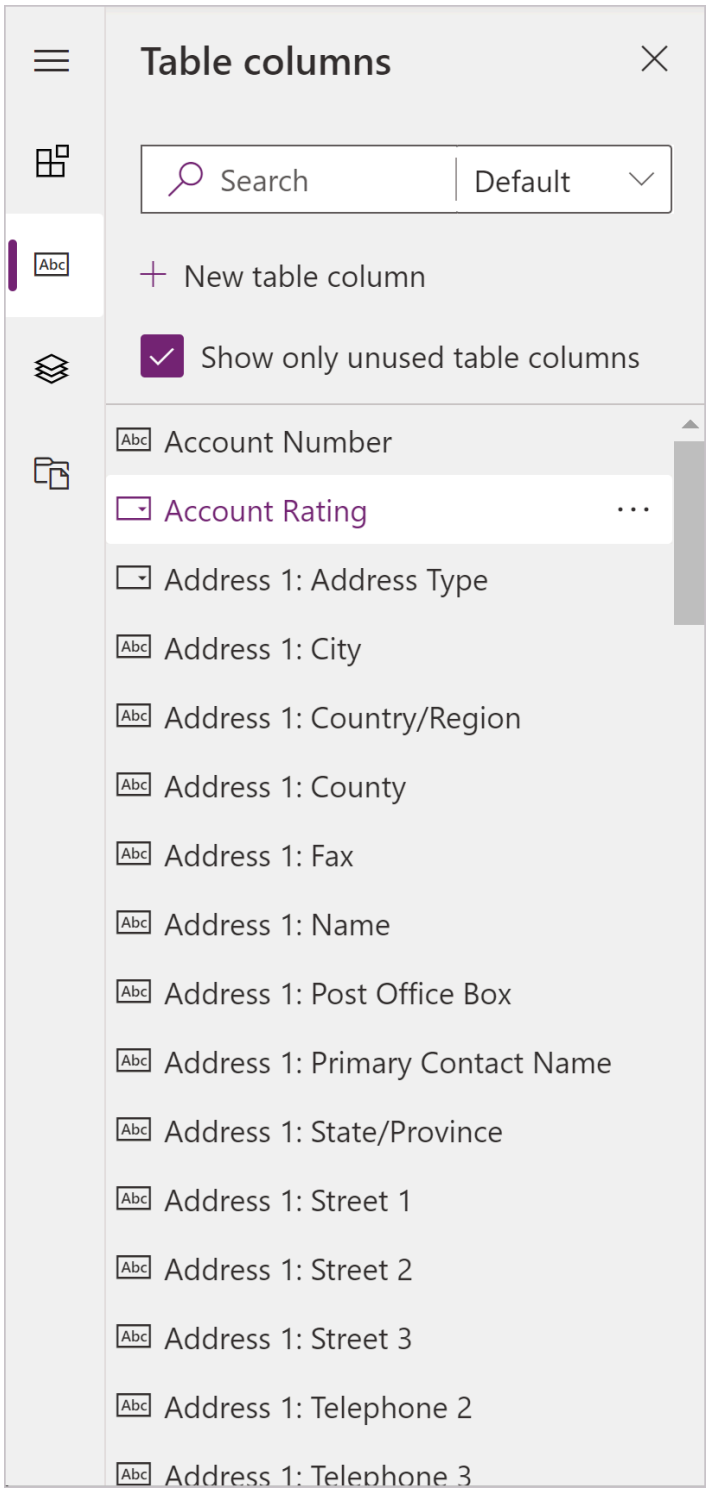
Article • 01/28/2025

Columns define the individual data items that can be used to store information in a table. Create new columns to capture data when existing standard tables don’t have columns that meet your requirements. After you create a new column, be sure to include it on the appropriate forms and views for the table so that they're available in your app. For information about the different column types, go to Types of columns.

Add, configure, move, or delete columns using the form designer.

Add columns to a form

To add columns to a form, use the **Columns** pane. The **Columns** pane lets you search and filter to help you quickly find columns. It also includes the option to show only unused columns.



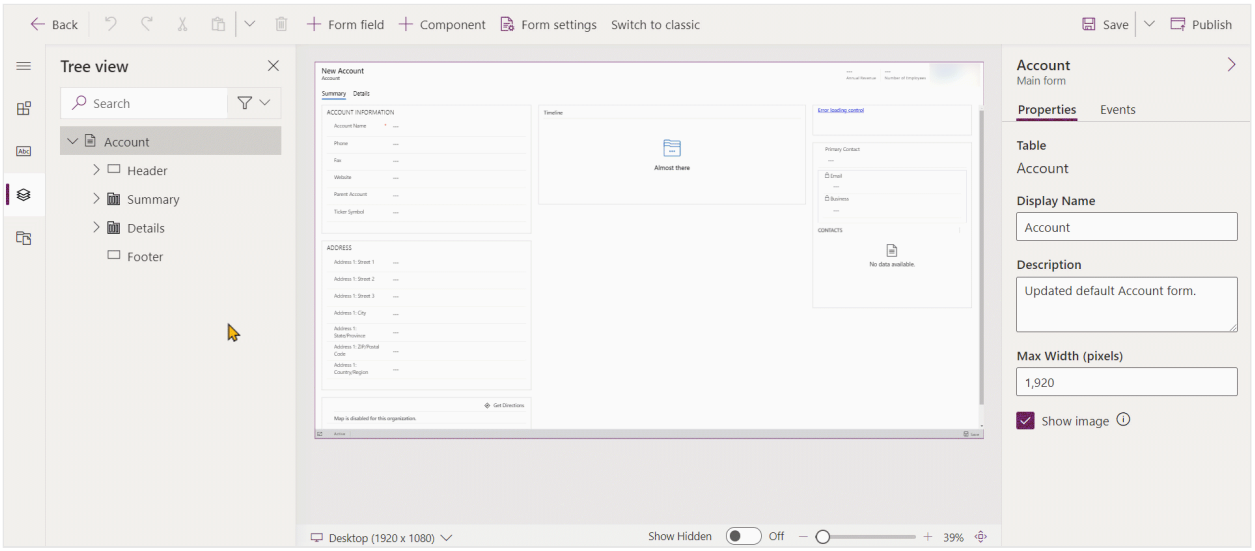
Add columns to a form using drag and drop

> **Note**
>
> When adding or moving columns using drag and drop be aware that the form preview is responsive and might be rendering multiple section columns as stacked. To ensure that the column being added or moved is in the correct section column, drop or paste it anchored to another column that is already in that section.

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. On the command bar, select **Add column**, and then select **Columns**. The **Columns** pane is open by default when the form designer is opened.
3. In the **Columns** pane, search, filter, or scroll to find the column you want to add. If you can't find a column, it might already be on the form. Clear **Show only unused columns** to view all columns, including those already added to the form.
4. In the **Columns** pane, select a column and drag it onto the form preview. As you drag the column on the form preview, you notice drop targets where you can add the column.
5. Drop the column in the location you want. Note the following behavior:

    - Columns can be dropped before or after any existing column or component.
    - Columns can also be dropped in the empty area within a section. In this case, the column is added in an available space so as to evenly distribute columns and components across the section columns.
    - Hovering over a tab header when dragging a column changes the currently selected tab, allowing you to add the column to a different tab.

6. Repeat steps 3-5 above if you want to add more columns.
7. On the command bar, select **Save and Publish** if you want to save and make your changes visible to users.

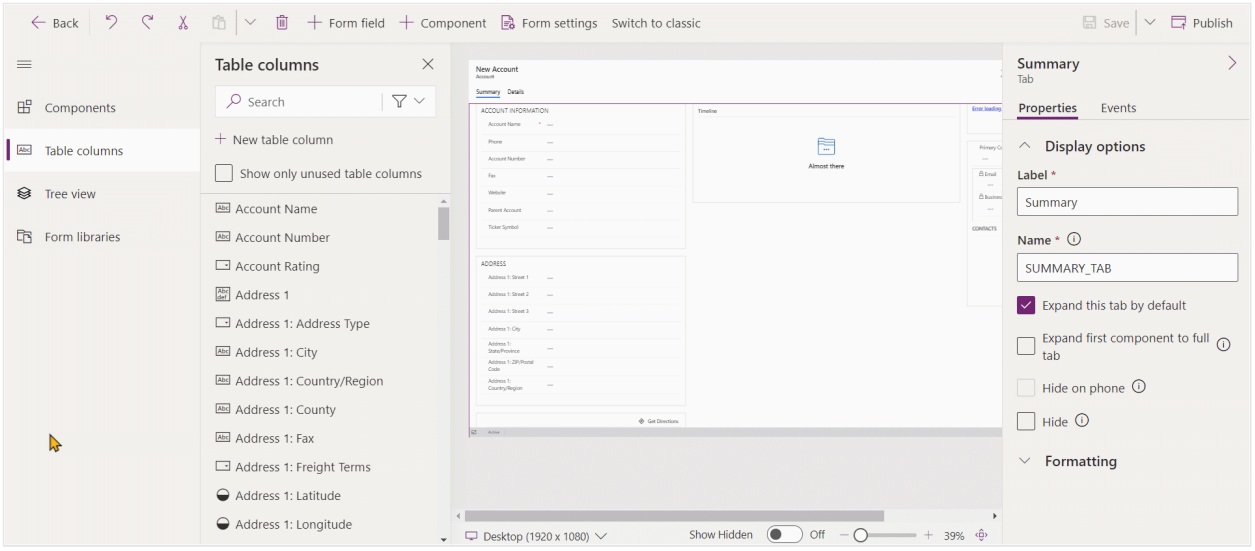


Add columns to a form using selection

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select another existing column or section. Note the following behavior:

- When you select an existing column, the new column is added after the existing column.
- When you select a section, the new column is added in an available space so as to evenly distribute columns across the section columns.

3. In the command bar, select **Add column**, or in the left pane, select **Columns**. The **Columns** pane is open by default when the form designer is opened.
4. In the **Columns** pane, search, filter, or scroll to find the column you want to add. If you can't find a column, it might already be on the form. Clear **Show only unused columns** to view all columns, including those already added to the form.
5. In the **Columns** pane, select a column to add it to the form. Alternatively, select **...** next to the column you want, and then select **Add to selected section**.
6. Repeat steps 2-5 above if you want to add more columns.
7. In the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.



Create a choice column

1. In the form designer on the command bar, select **Form field**.
2. Select **New table column**.
3. In the **New column** pane, enter or select the values for the following complete the properties for the choice column. More information: Configure properties for a local choice

Configure column properties on a form

The properties and events can be configured for column when you create or edit a form using the form designer.

Column properties

The **Properties** tab displays the following properties.

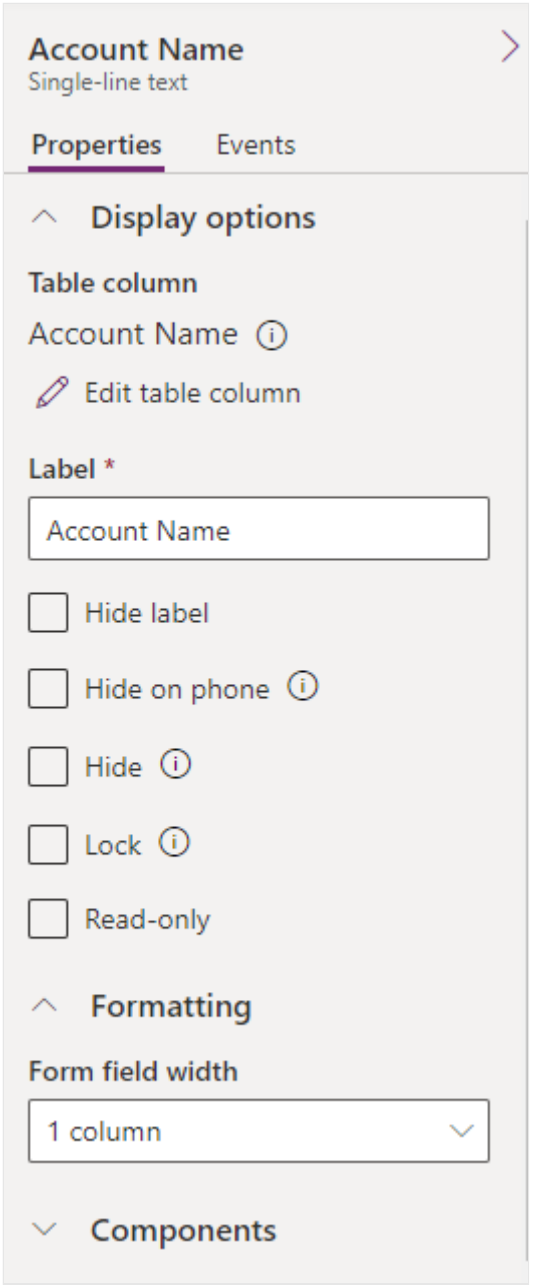
Expand table

Area	Name	Description
Display options	**Table column**	The name of the column. This comes from the column properties on the table and is read-only.
Display options	**Label**	By default the label matches the display name of the column. You can override that name for the form by entering a different label here. This property is required.
Display options	**Hide label**	When selected, the column label is hidden.
Display options	**Hide on phone**	The column can be hidden to render a condensed version of the form on phone screens.
Display options	**Hide**	When selected, the column is hidden by default and can be shown using code.
Display options	**Lock**	Lock this column so it can't be removed from the form.
Display options	**Read-only**	When selected, the column value isn't editable on the form.
Formatting	**Form field width**	When the section containing the columns has more than one column, you can set the column to occupy up to the number of columns that the section has.
Formatting	**Form field height**	Some columns, such as the multiline text column, let you control the height of the column by specifying the number of rows the column occupies.
Formatting	**Use all available vertical space**	Instead of setting the height by a number of rows, some columns, such as the multiline text column, let you set the column height to expand to all the available space on the form.
Components	**+ Component**	Adds a control to the column such as the canvas app control or the AI business card control.

⚠ Warning

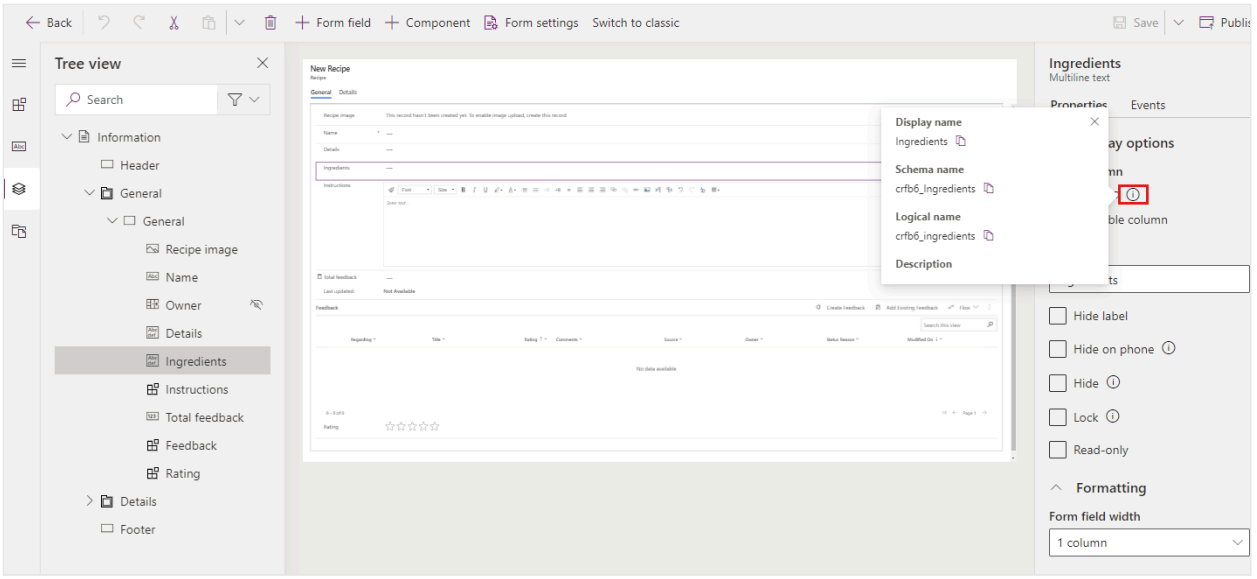
The **Hide** and **Read-only** properties should never be used as a secure way to prevent users from viewing or editing column values. These properties only apply to a form and don't have an effect on a user's data privileges. When a column is hidden on a form, users can still access data in other ways like viewing other forms or by making Web API calls. To secure columns, use column-level security to control access.

> **Note**
>
> In Unified Interface, the label size setting is ignored because the form adjusts width for all columns to ensure proper rendering for screen sizes from very small to extra-large.



Edit a column and view column names

To view column names, such as schema name or to change additional column properties, such as required, searchable, description, and size, select the column, and then in the right column properties pane select the information icon ⓘ next to the column name.



To edit the column data type format, required, and advanced options, follow these steps.

1. On the column **Properties** tab, select **Edit table column** to display the **Edit Table Column** pane where you can make the changes you want.
2. When you're finished making your changes, select **Done**.
3. Select **Save and Publish** to make your changes available to app users.

Column events

The **Events** tab is where event handlers for model-driven app forms can be configured for a column. An event handler consists of a reference to a JavaScript web resource and a function defined within that web resource that will execute when the event occurs, such as when the value for the column changes. More information: Configure model-driven app form event handlers

Move columns on a form

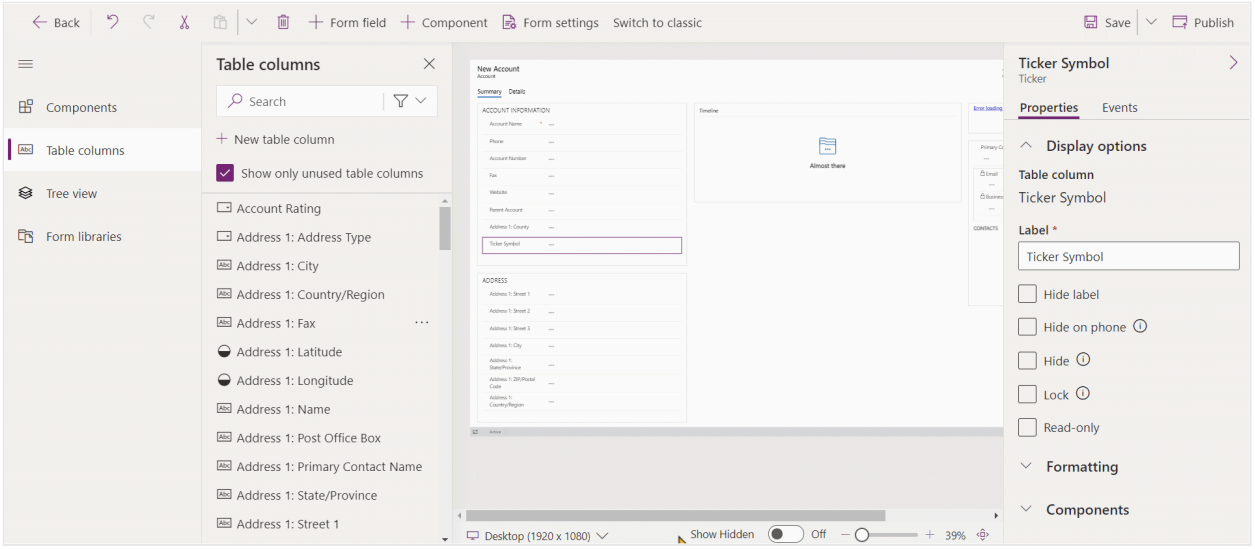
You can move a column on a form using drag and drop or cut and paste actions.

Move columns on a form using drag and drop

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the column that you want to move and drag and drop it. As you drag the column on the form preview, you'll see drop targets where you can move the column to. Note the following:

    - Columns can be dropped before or after any existing column or component.
    - Columns can also be dropped in the empty area within a section. In this case, the column is added in an available space so as to evenly distribute columns and components across the section columns.
    - Hovering over a tab header when dragging a column changes the currently selected tab, allowing you to add the column to a different tab.

3. Repeat step 2 above if you want to move more columns.
4. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.



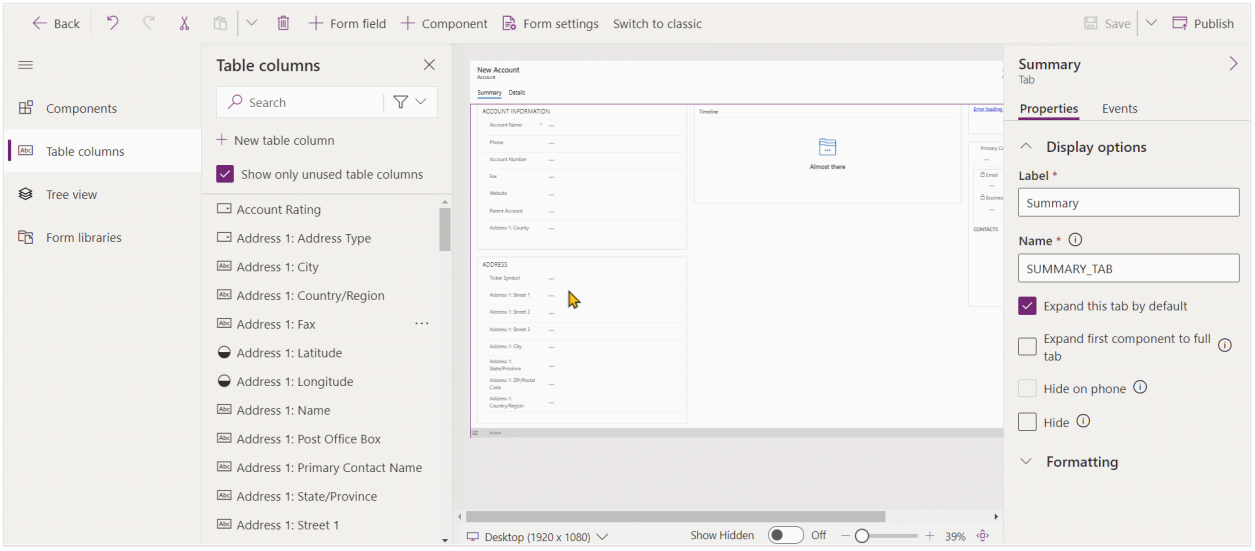
Move columns on a form using cut and paste

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the column that you want to move.
3. On the command bar, select **Cut**.
4. In the form preview, select another existing column, component, or section. You can also switch to a different tab if needed.
5. On the command bar, select **Paste** or select the chevron, and then select **Paste before**. Note the following:

    - When you select **Paste**, the column that is moved is pasted after the existing column or component.

- When you select **Paste before**, the column that is moved is pasted before the existing column or component.
- When you select a section, the column that is moved is added in an available space so as to evenly distribute columns and components across the section columns. The **Paste before** action isn't applicable and therefore not available in this case.

6. Repeat steps 2-5 above if you want to move more columns.
7. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.



Delete columns on a form

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the column that you want to delete from the form.
3. On the command bar, select **Delete**.
4. Repeat steps 2-3 if you want to delete more columns.
5. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

> **Note**
>
> - If you delete a column by mistake, on the command bar, select **Undo** to revert the form to its previous state.

- You can't delete a column that is locked or is required and not present anywhere else on the form.

Create a new column on the table when editing a form (preview)

[This topic is pre-release documentation and is subject to change.]

Create new columns from the form designer. When a column is created on a table, it isn't limited to the current form and is available for use in other forms and views for the table.

ⓘ Important

- This is a preview feature.
- You can use the modern form designer to create any available column type.

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. On the left navigation pane, select the **Table columns** pane, and then select **New table column**.
3. In the **New column** pane, provide the **Display name** for the column.
4. Select the **Data type** and configure any other required properties of the column.
5. Select **Save** to create a new column on the table. The column appears in the **Table columns** pane.
6. If you want to add the newly created column to the form, follow the steps outlined in the **Add columns to a form** section.

See also

Overview of the model-driven form designer
Create, edit, or configure forms using the form designer
Add, configure, move, or delete components on a form
Add, configure, move, or delete sections on a form
Add, configure, move, or delete tabs on a form
Configure header properties in the form designer
Add and configure a subgrid component on a form
Add and configure a quick view component on a form
Configure a lookup component on a form

Using the tree view in the form designer
Create and edit columns

Feedback

Was this page helpful? Yes No

Provide product feedback

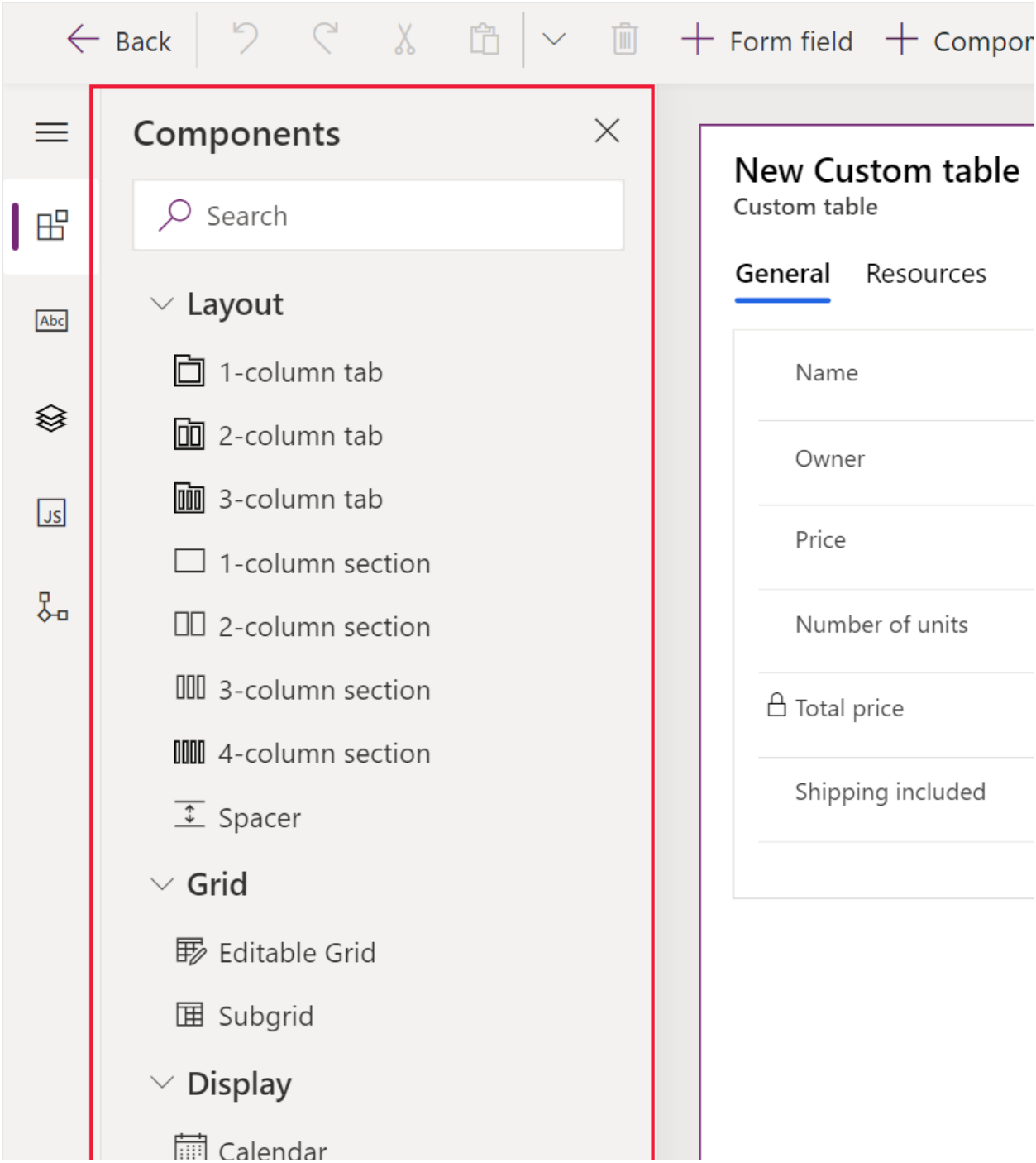
Add, configure, move, or delete components on a form

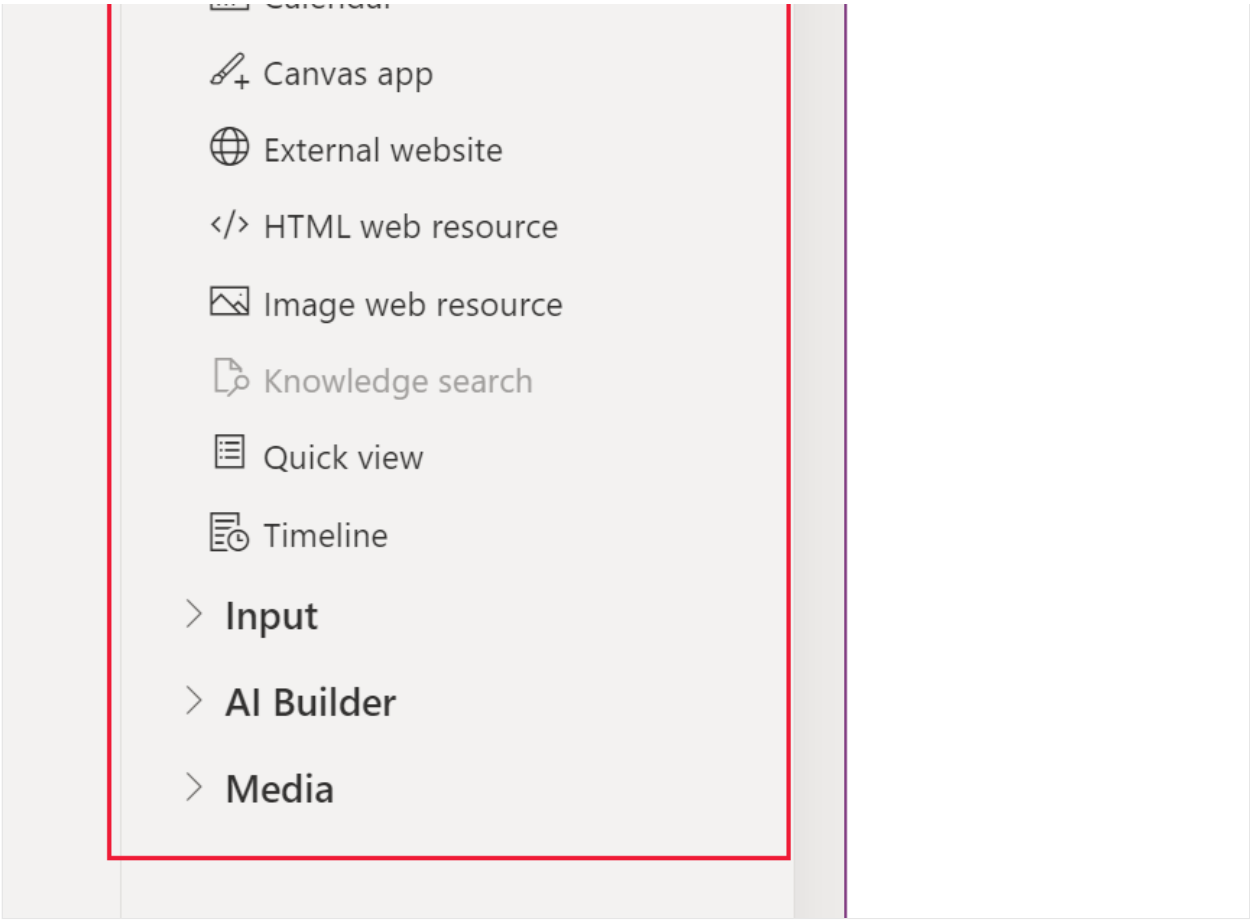
Article • 06/29/2022

By using the form designer, makers can easily add and configure popular components such as subgrid, quick view, web resources, pen input, and more.

Add components to a form

To add components to a form, use the **Components** pane. Components can be found easily using the search in the **Components** pane.





Add components to a form using drag and drop

> **Note**
>
> When adding or moving components using drag and drop, be aware that the form preview is responsive and might render multiple section columns as stacked. To ensure that the component that's added or moved is in the correct section column, drop or paste it anchored to another column or component that is already in that section column.

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. On the command bar, select **Add component**, or in the left pane, select **Components** to see a list of available components. Hover over a component in the list to see a preview image, description, and other details of that component.
3. In the **Components** pane, search or scroll to find the required component then select to add the component.
4. Drag and drop the component onto the form preview. As the component is dragged on the form preview, drop targets appear.

> **Note**
>
> - Components can be dropped before or after any existing component or column.
> - Components can also be dropped in the empty area within a section. In this case the component will be added in an available space so as to evenly distribute columns and components across the section columns.
> - Hovering over a tab header when dragging a component changes the currently selected tab, allowing the component to be added to a different tab.
> - When the component is dropped, in most cases, a dialog to configure the properties of the component will appear. Ensure that all the required properties of the component are configured.

5. In the dialog to configure the properties of the component, under **Show component on**, the **Web**, **Mobile**, and **Tablet** options are selected by default to ensure the component is used when the form is displayed on the web, mobile app, and tablet app. Based on you app's requirements, you might want to clear some of the client types for the component.

6. Select **Done**.

7. Repeat steps 3-6 above to add more components.

8. On the command bar, select **Save** to save the form, or select **Publish** to make the changes visible to users.

Add components for a column on the form

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select an existing column.
3. In the property pane, under the **Components** area, select **+ Component**.
4. The **Add component** dialog displays a list of components that are available for the current column type. Hover over a component in the list to see a preview image, description, and other details of that component.
5. In the **Add component** dialog, search, or scroll to find the required component to add and then select it. In most cases, a dialog is displayed to configure the properties of the component. Ensure that all the required properties of the component have been configured.

6. In the dialog to configure the properties of the component, under **Show component on**, the **Web**, **Mobile**, and **Tablet** options are selected by default to ensure the component is used when the form is displayed on the web, mobile app, and tablet app. Based on the requirements some of these options to limit the usage of the component can be cleared.
7. Select **Done**.
8. Repeat steps 2-7 above to add more components to the same or another column.
9. On the command bar, select **Save** to save the form, or select **Publish** to save and make the changes visible to users.

Configure components on a form

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select an existing column.
3. In the property pane, under the **Components** area, select the component that you want configured.
4. A dialog might appear to configure the properties of the component. Change the properties of the component as needed, and then select **Done**.
5. Repeat steps 2-4 to configure more components on the same or another column.
6. On the command bar, select **Save** to save the form, or select **Publish** to save and make the changes visible to users.

Move components on a form

To move components on a form, either use drag-and-drop or cut-and-paste actions.

Move components on a form using drag and drop

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form

2. In the form preview, select the component to move and drag and drop it. As the component is dragged onto the form preview, drop targets appear where it can be moved.

   > ⓘ **Note**
   >
   > - Components can be dropped before or after any existing component or column.

- Components can also be dropped in the empty area within a section. In this case the component will be added in an available space to evenly distribute components and columns across the section columns.
- Hover over a tab header while using drag and drop to add the component to a different tab.

3. Repeat steps 2-3 above to move more components.
4. On the command bar, select **Save** to save the form, or select **Publish** to save and make the changes available to users.

Move components on a form using cut and paste

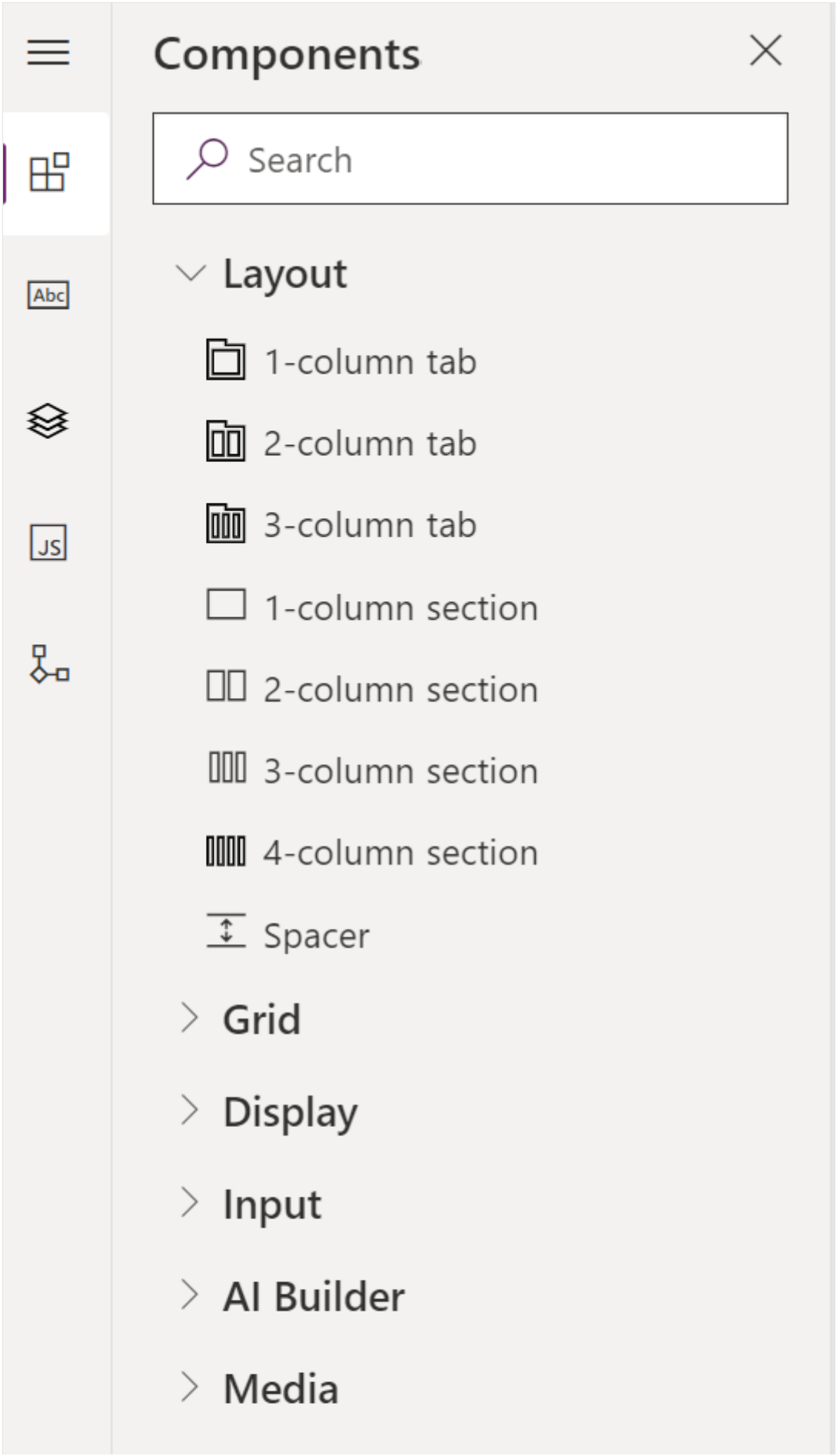
1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the required component to move.
3. On the command bar, select **Cut**.
4. In the form preview, select another existing component, column, or section. It is possible to switch to a different tab if needed.
5. On the command bar, select **Paste** or select the chevron, and then select **Paste before**.

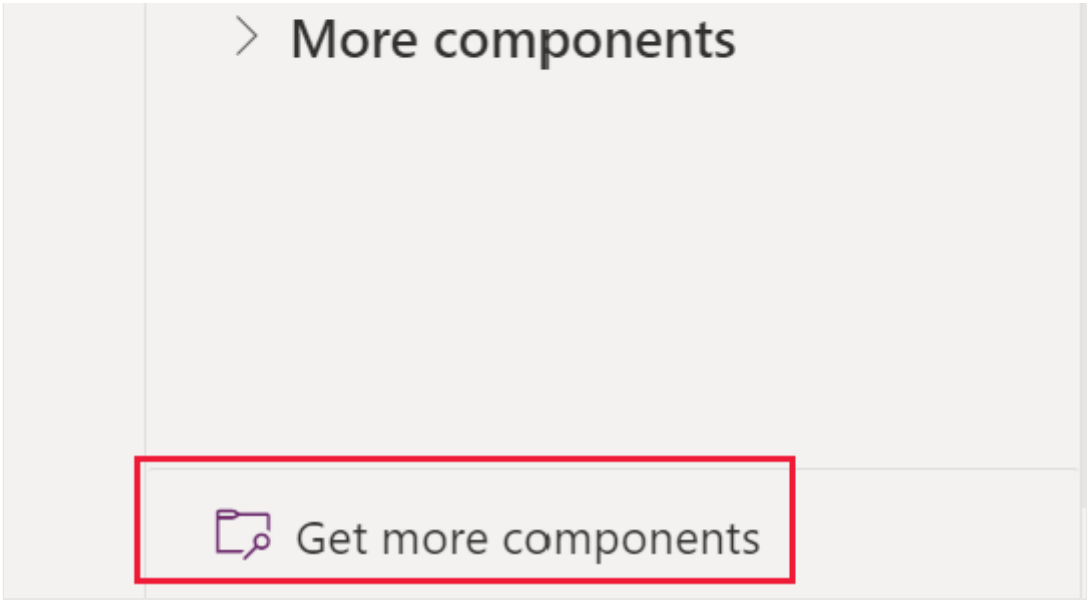
> ⓘ Note
>
> - When **Paste** is selected, the component being moved is pasted after the existing component or column.
> - When **Paste before** is selected, the component being moved is pasted before the existing component or column.
> - When a section is selected, the component being moved is added in an available space so as to evenly distribute components and columns across the section columns.

6. Repeat steps 2-5 above if to move more components.
7. On the command bar, select **Save** to save the form, or select **Publish** to save and make the changes visible to users.

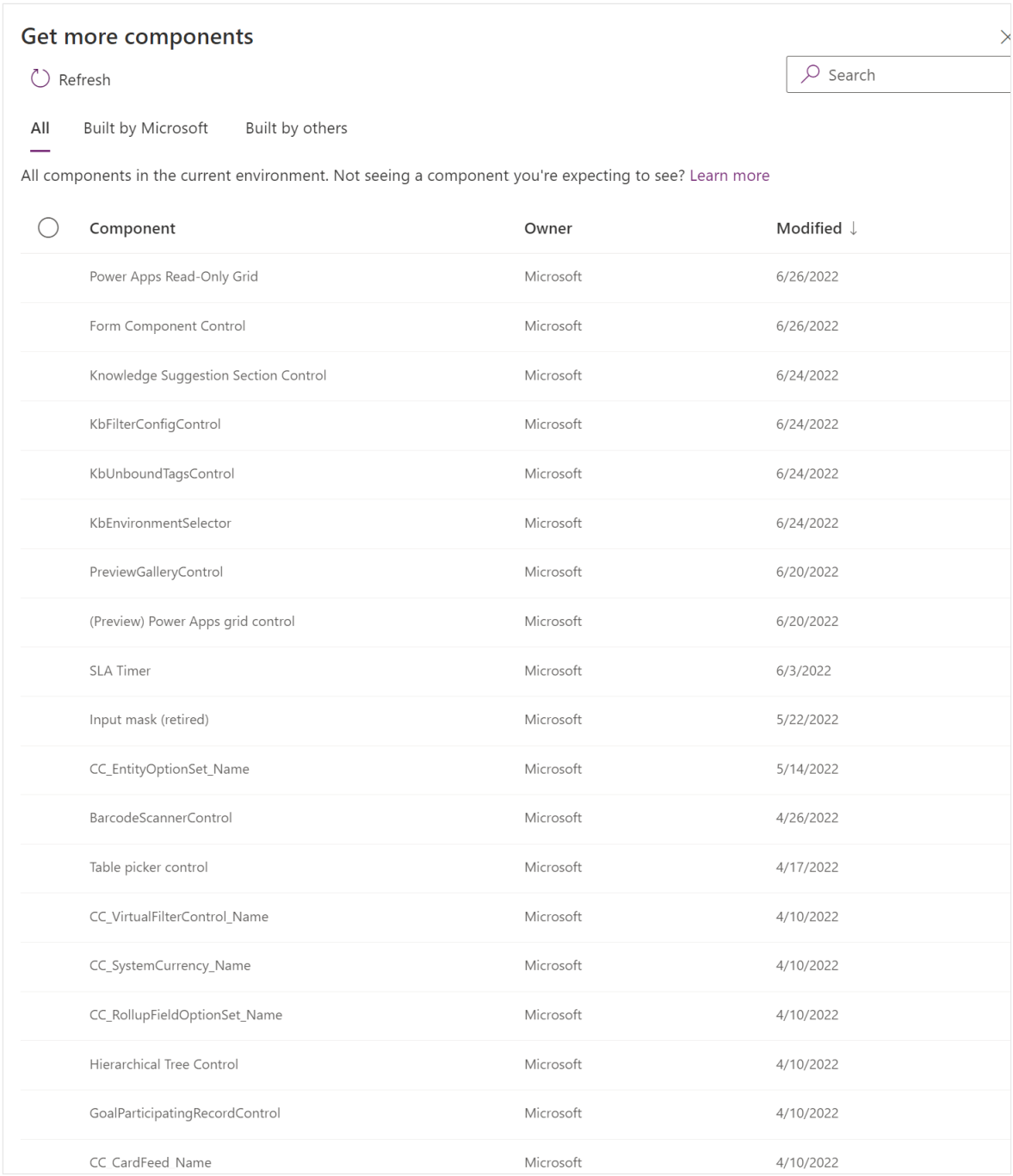
Get more components

1. Select **Get more components** to find and add other Power Apps components as well as custom controls that aren't available by default in the **Component** pane.





2. Select the tab you want, either **Built by Microsoft** or **Built by others**, and then select the component.



3. Select **Add**. The component appears under the **More components** section of the **Component** pane.

Delete components on a form

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form

2. In the form preview, select the component that you want to delete from the form, and then on the command bar, select **Delete**.

3. Repeat step 2 to delete more components.

4. On the command bar, select **Save** to save the form, or select **Publish** to save and make the changes visible to users.

> **Note**
>
> - If a component is deleted by mistake, select **Undo** on the command bar to revert the form to its previous state.
> - A component can't be deleted when it is locked or is using a required column that is not present anywhere else on the form.

See also

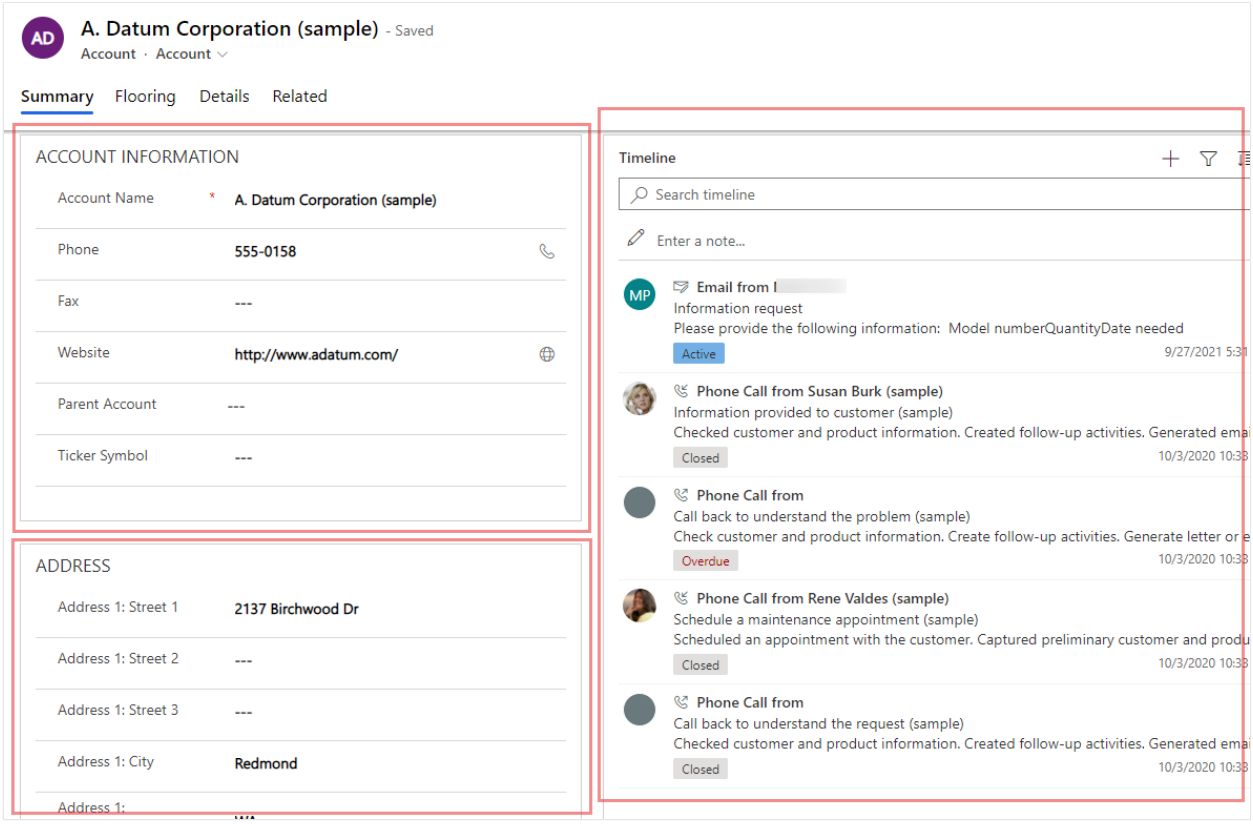
Overview of the model-driven form designer
Create, edit, or configure forms using the form designer
Add, configure, move, or delete columns on a form
Add, configure, move, or delete sections on a form
Add, configure, move, or delete tabs on a form
Configure header properties in the form designer
Add and configure a subgrid component on a form
Add and configure a quick view component on a form
Configure a lookup component on a form
Using the tree view in the form designer
Create and edit columns

Add, configure, move, or delete sections on a form

Article • 05/05/2022

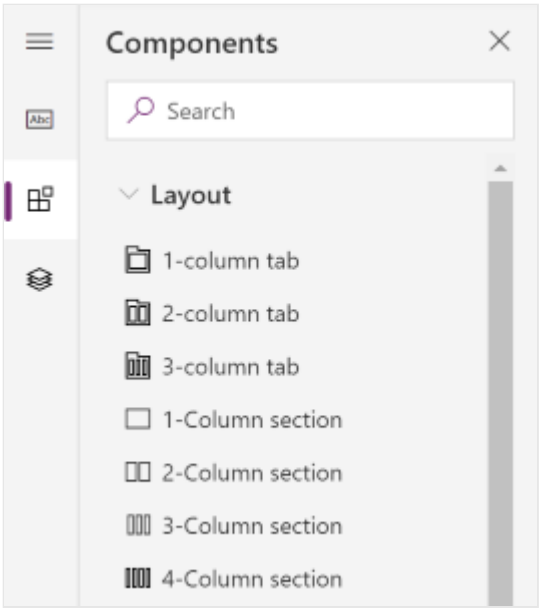
A section on a form occupies the space available in a tab. Sections have several configurable properties including a label that can be displayed.

Sections help you group similar components in an area of a form tab, such as the ACCOUNT INFORMATION, ADDRESS, and Timeline single column sections on the main account form Summary tab. Sections can have up to four columns.



Add sections to a form

To add sections to a form, use the **Components** pane.



> **Note**
>
> Sections can only be added on main forms and quick view forms. More information: **Form types**. In Unified Interface, empty sections will not render in runtime and will be automatically hidden.

Add sections to a form using drag and drop

> **Note**
>
> When adding or moving sections using drag and drop be aware that the form preview is responsive and may be rendering multiple tab columns as stacked. To ensure that the section being added or moved is in the correct tab column, drop or paste it anchored to another section that is already in that tab column.

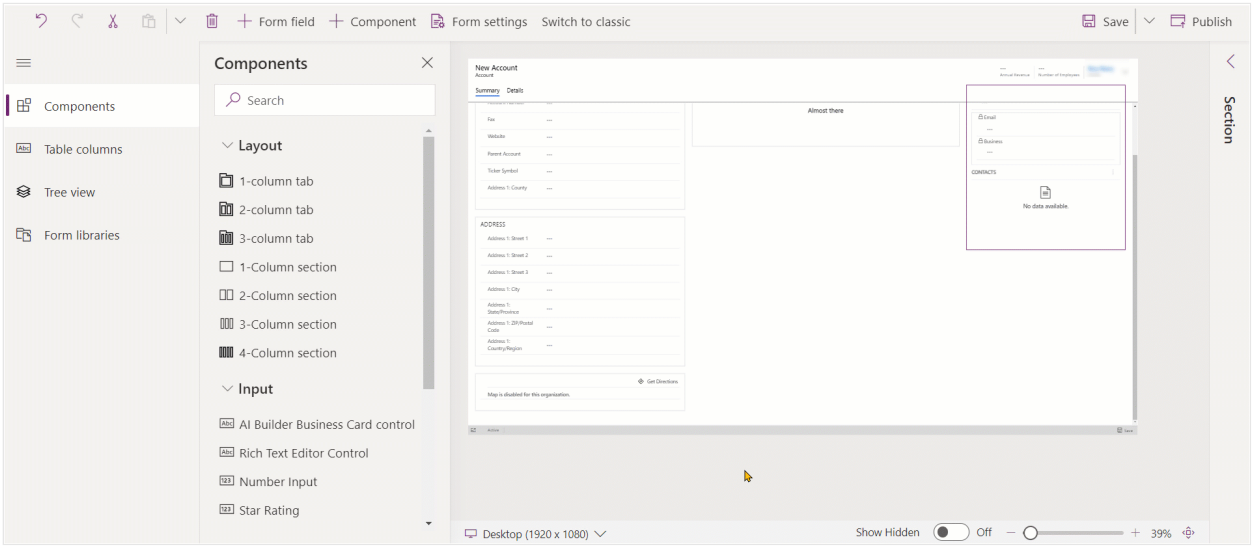
1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. On the command bar, select **Add component**, or in the left pane, select **Components**.
3. In the **Components** pane, select a section component and drag it onto the form preview. As you drag the section on the form preview, you will see drop targets where you can add the section.
4. Drop the section in the location you want. Note the following behavior:

    - Sections can be dropped before or after any existing section.
    - Sections can also be dropped in the empty area within a tab. In this case, the section will be added in an available space so as to evenly distribute sections

across the tab columns.

- Hovering over a tab header when dragging a section changes the currently selected tab, allowing you to add the section to a different tab.

5. Repeat steps 3-4 above if you want to add more sections.
6. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

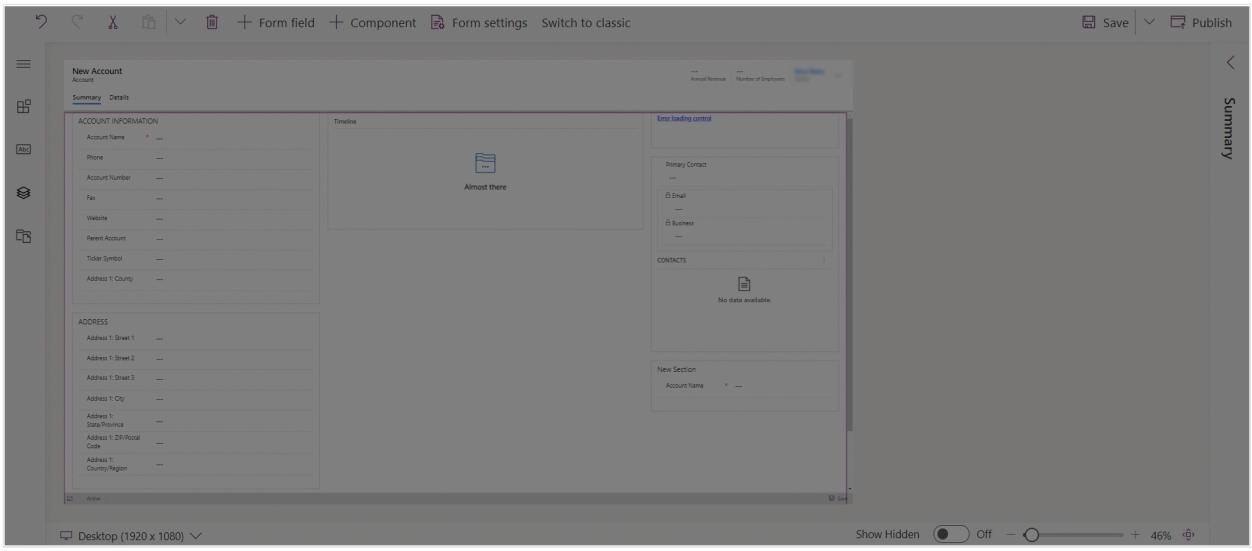


Add sections to a form using selection

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select another existing section or tab. Note the following behavior:

   - When you select an existing section, the new section is added after the existing section.
   - When you select a tab, the new section is added in an available space so as to evenly distribute sections across the tab columns.

3. On the command bar, select **Add component**, or in the left pane, select **Components**.
4. In the **Components** pane, select a section component to add it to the form. Alternatively, select **...** next to the section component you want, and then select **Add to selected tab**.
5. Repeat steps 2-4 above if you want to add more sections.
6. On the command bar, select **Save** to save the form. Select **Publish** to save and make your changes visible to users.



Configure sections on a form

These are the properties available to configure a section when you create or edit a form using the form designer.

Area	Name	Description
Display options	**Label**	The localizable label for the section visible to users. This property is required.
Display options	**Name**	The unique name for the section that is used when referencing it in scripts. The name can contain only alphanumeric characters and underscores. This property is required.
Display options	**Hide label**	When selected, the section label is hidden.
Display options	**Hide on phone**	The section can be hidden to render a condensed version of this form on phone screens.
Display options	**Hide**	When selected, the section is hidden by default and can be shown using code.
Display options	**Lock**	Lock this section to keep it from being removed.
Formatting	**Columns**	Specify up to four columns for the section.
Formatting	**Component label position**	Determines the position for the component label. Select **Left** to display the label to the left of the value or **Top** to place the label above the value.

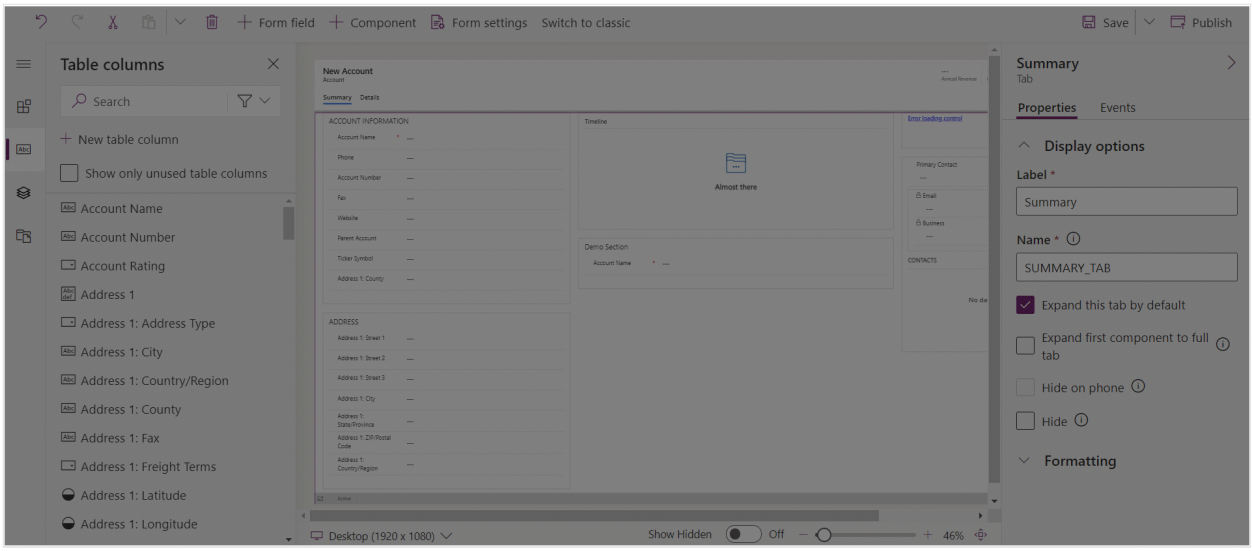
> **Note**
>
> We recommend limiting the length of section label text to about 50 characters. Section label text doesn't wrap and can truncate on devices with small screens.
>
> Sections that don't have any columns or controls will be hidden at runtime in Unified Interface.

Move sections on a form

You can move sections using drag and drop or cut and paste actions.

Move sections on a form using drag and drop

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the section label or empty space within the section that you want to drag and drop. As you drag the section on the form preview, you will see drop targets where you can move the section to. Note the following behavior:
    - Sections can be dropped before or after any existing section.
    - Sections can also be dropped in the empty area within a tab. In this case, the section will be added in an available space to evenly distribute sections across the tab columns.
    - Hovering over a tab header when dragging a section changes the currently selected tab, allowing you to add the section to a different tab.
3. Repeat step 2 above if you want to move more sections.
4. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

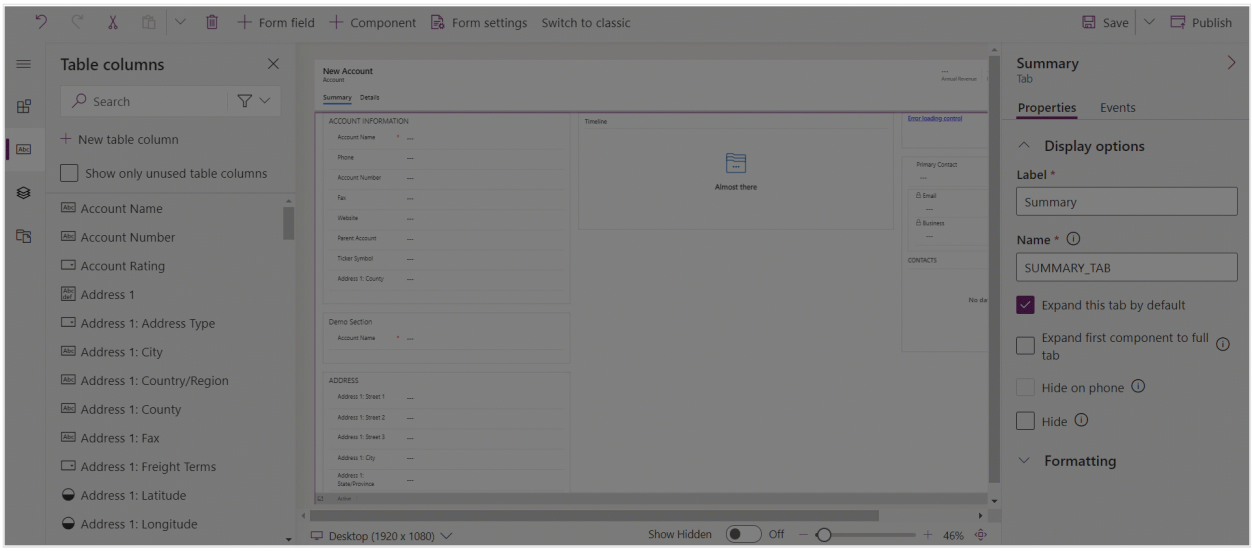


Move sections on a form using cut and paste

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the section that you want to move.
3. On the command bar, select **Cut**.
4. In the form preview, select another existing section or tab. You can also switch to a different tab if needed.
5. On the command bar, select **Paste** or select the chevron, and then select **Paste before**. Note the following behavior:

    - When you select **Paste**, the section being moved is pasted after the existing section.
    - When you select **Paste before**, the section being moved is pasted before the existing section.
    - When you select a tab, the section moved is added in an available space to evenly distribute sections across the tab columns. The **Paste before** action is not applicable and therefore not available in this case.

6. Repeat steps 2-5 above if you want to move more sections.
7. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.



Delete sections on a form

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the section that you want to delete from the form.
3. On the command bar, select **Delete**.
4. Repeat steps 2-3 above if you want to delete more sections.
5. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

> **Note**
>
> - Sections can only be deleted on main forms and quick view forms. More information: **Form types**
> - If you delete a section by mistake, on the command bar, select **Undo** to revert the form to its previous state.
> - You can't delete a section that contains a column that is required or locked.
> - You can't delete a section that is locked.
> - A tab needs to have at least one section in each tab column. If you delete the last remaining section in a tab column a new section will be automatically added.
> - In Unified Interface, sections won't be rendered if you have selected to hide them including the title and the border. This is different than the

legacy web client, and is by design to ensure proper rendering of the form across differing view ports from extra wide to narrow.

See also

Overview of the model-driven form designer

Create, edit, or configure forms using the form designer

Add, configure, move, or delete columns on a form

Add, configure, move, or delete components on a form

Add, configure, move, or delete tabs on a form

Configure header properties in the form designer

Add and configure a subgrid component on a form

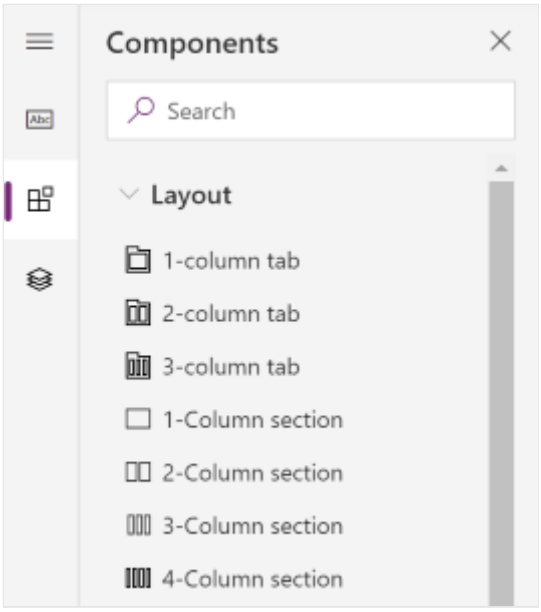
Add, configure, move, or delete tabs on a form

Summarize this article for me

Add, move, or delete tabs on a form using the form designer.

Add tabs to a form

To add tabs to a form, use the **Components** pane.



> **Note**
>
> Tabs can only be added on main forms. More information: Form types

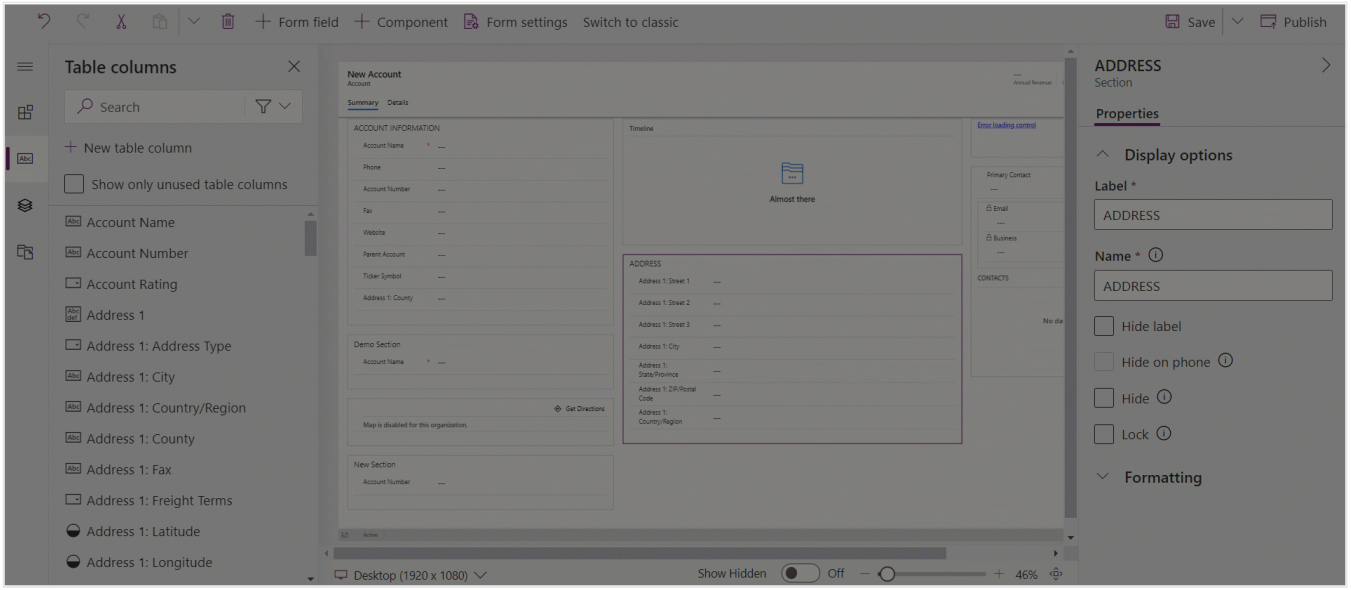
Add tabs to a form using drag and drop

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. On the command bar, select **Add component**, or in the left pane, select **Components**.
3. In the **Components** pane, select a tab component, such as a *1-column tab*, and drag and drop it onto the form preview. As you drag the tab on the form preview, you notice drop targets where you can add the tab. Note the following behavior:

    - Tabs can be dropped before or after any existing tabs by hovering over the tab headers.

- Tabs can also be dropped before or after the current tab by hovering over the left or right edge of the current tab.

4. Repeat step 3 if you want to add more tabs.
5. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

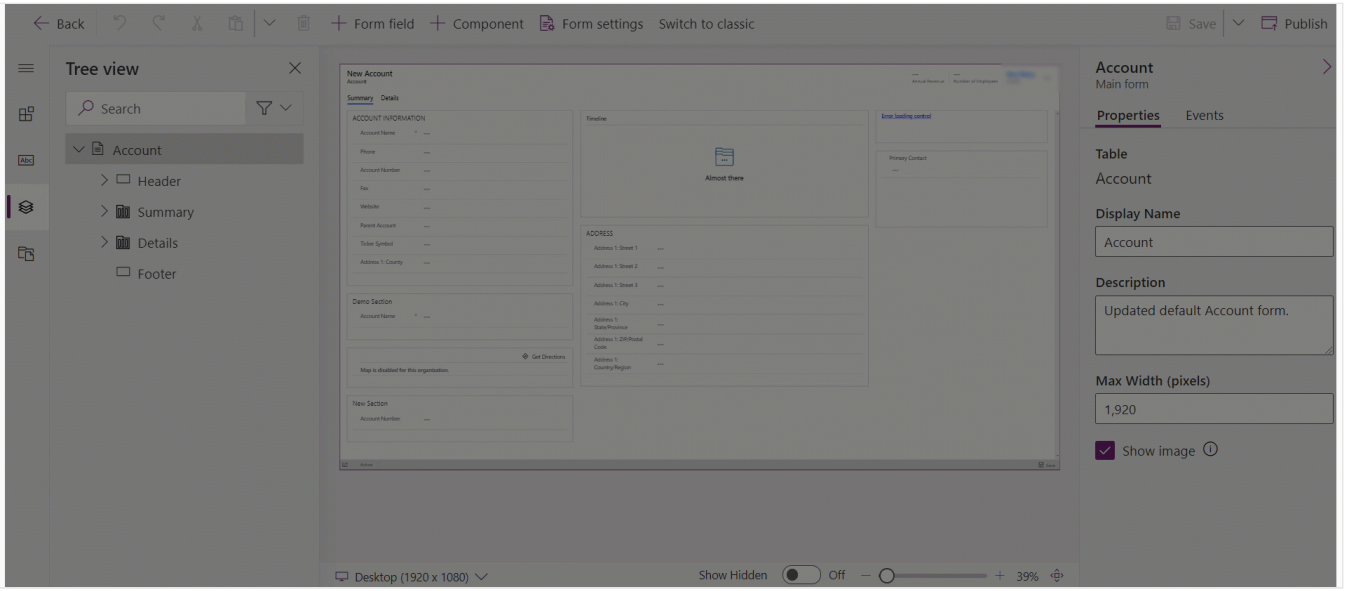


Add tabs to a form using selection

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select another existing tab or the form canvas. Note the following behavior:

   - When you select an existing tab, the new tab is added after the existing tab.
   - When you select the form canvas, the new tab is added as the last tab on the form.

3. On the command bar, select **Component**, or in the left pane, select **Components**.
4. In the **Components** pane, select a tab component to add it to the form. Alternatively, select **...** next to the tab component you want, and then select **Add to form**.
5. Repeat steps 2-4 if you want to add more tabs.
6. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.



Configure tabs on a form

These properties are available to configure a tab when you create or edit a form using the form designer.

Expand table

Area	Name	Description
Display options	**Label**	The localizable label for the tab visible to users. This property is required.
Display options	**Name**	The unique name for the tab that is used when referencing it in scripts. The name can contain only alphanumeric characters and underscores. This property is required.
Display options	**Expand this tab by default**	The tab state can toggle between expanded or collapsed using form scripts or by people selecting the label. Choose the default state for the tab.
Display options	**Hide**	When selected, tab is hidden by default and can be shown using code.
Display options	**Hide on phone**	The tab can be hidden to render a condensed version of this form on phone screens.
Formatting	**Layout**	Tabs can have up to three columns. Use these options to set the number of tabs and what percentage of the total width they should fill.
Formatting	**Column Width**	Enter the column width. Valid values are 1 to 100. The default is 100.

Move tabs on a form

You can move a tab on a form using drag and drop or cut and paste actions.

Move tabs on a form using drag and drop

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the tab header of the tab that you want to move and drag and drop it. As you drag the tab on the form preview, you see drop targets where you can move the tab. Note the following behavior:

    - Tabs can be dropped before or after any existing tabs by hovering over the tab headers.
    - Tabs can also be dropped before or after the current tab by hovering over the left or right edge of the current tab.

3. Repeat step 2 if you want to move more tabs.
4. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

Move tabs on a form using cut and paste

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the tab that you want to move.
3. On the command bar, select **Cut**.
4. In the form preview, select another existing tab or the form.
5. On the command bar, select **Paste** or select the chevron, and then select **Paste before**. Note the following behavior:

    - When you select **Paste**, the tab moved is pasted after the existing tab.
    - When you select **Paste before**, the tab moved is pasted before the existing tab.
    - When you select the form, the tab moved is added as the last tab on the form. The **Paste before** action is not applicable and therefore not available in this case.

6. Repeat steps 2-5 if you want to move more tabs.
7. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

Delete tabs on a form

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form preview, select the tab that you want to delete from the form.
3. On the command bar, select **Delete**.
4. Repeat steps 2-3 above if you want to delete more tabs.
5. On the command bar, select **Save** to save the form, or select **Publish** if you want to save and make your changes visible to users.

> **Note**
>
> - Tabs can only be deleted on main forms. More information: Form types
> - If you delete a tab by mistake, on the command bar, select **Undo** to revert the form to its previous state.
>   - You can't delete a tab that contains sections with required or locked columns.
> - You can't delete a tab that has locked sections.
> - A form must have at least one tab. You can't delete the last remaining tab on the form.

See also

Overview of the model-driven form designer
Create, edit, or configure forms using the form designer
Add, configure, move, or delete columns on a form
Add, configure, move, or delete components on a form
Add, configure, move, or delete sections on a form
Configure header properties in the form designer
Add and configure a subgrid component on a form
Add and configure a quick view component on a form
Configure a lookup component on a form
Using the tree view in the form designer
Create and edit columns

Last updated on 03/02/2026

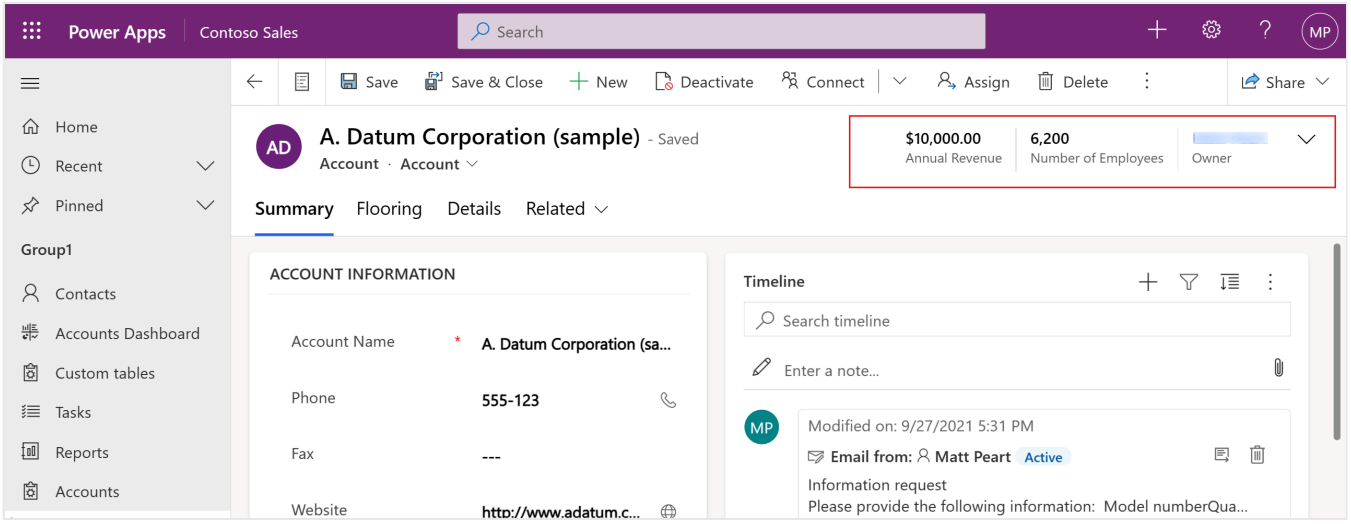
Configure header properties in the form designer

Makers can choose the columns displayed in the form header. Form headers have two main visual components, the form header and the flyout for the form header.

Form header

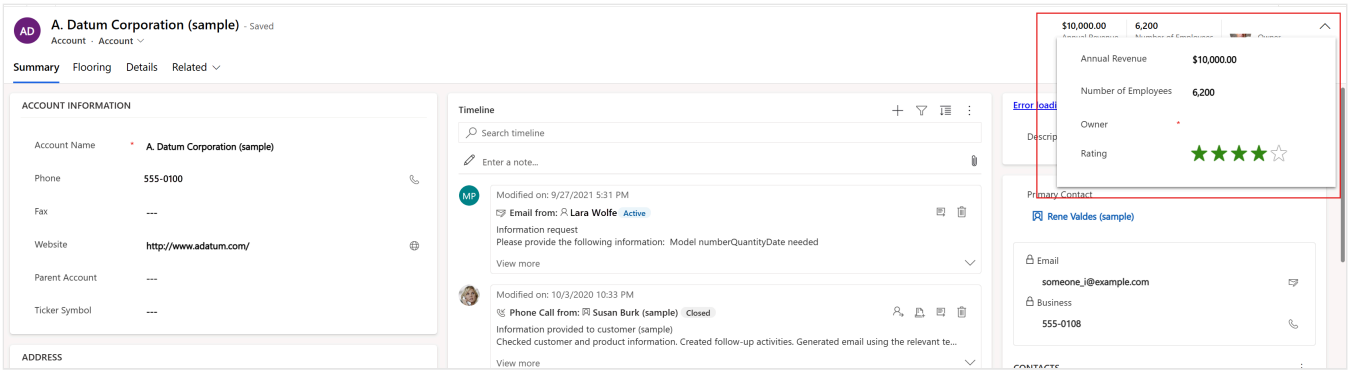
The form header is used to display common columns that are important for the table. Form headers can't display columns that use unsupported controls. More information: Controls that can be displayed on the form header

Most controls that can't be displayed in form header can be displayed in the form header flyout.



Form header flyout

The form header flyout displays when a user selects the chevron located to the right of the form header, which can display more information otherwise hidden in the form header. The header flyout enables users to edit column values.



The header flyout displays all header columns including the four columns that are directly displayed in the header. By default, the header flyout is available for display, but makers can hide it if needed.

Show or hide the header flyout

To show or hide the header flyout for a model-driven form, follow these steps:

1. Open the form designer to create or edit a form.
2. Select the form header in the form preview or use the tree view to select it.
3. In the right property pane, select **Show header flyout** to make the header flyout visible or clear it to hide the header flyout.
4. In the command bar, select **Save** to save the form, or select **Publish** to save and make your changes visible to users.

> **Note**
>
> - Use the new form designer. The classic form designer doesn't provide the ability to show or hide the header flyout.
> - An image for a table is displayed in the header only when the **Primary Image** attribute is defined for the table and the form property **Show image in the form** is enabled. More information: Image columns.
> - Developers can specify an image for a table by using the EntityMetadata.PrimaryImageAttribute attribute.

High-density header

Model-driven app forms use a high-density form header to ensure that key information is always visible to users. High-density headers also make sure the record title never truncates and long record titles are displayed using multiple lines. Similarly, the high-density header also

ensures that up to four column values are directly visible in the header and never truncated or hidden.

To ensure that key information is always visible, the framework displays up to four read-only column values and users can't directly edit the column values in the header. Visualizations such as custom components or web resources don't render on the header, but are available in the flyout.

Controls that can be displayed on the form header

Columns set to these controls can be displayed in the main body of a high-density header:

- `checkbox`
- `datetime`
- `decimal`
- `duration`
- `emailaddress`
- `float`
- `integer`
- `languagepicker`
- `lookup`
- `money`
- `phonenumber`
- `picklist`
- `regarding`
- `textbox`
- `ticker`
- `timezonepicklist`

These controls are replaced with a simplified version in the header. Thus, not all control features are available. For example, for lookup controls, just a link is rendered and control options like **Use Main Form Dialog for Edit** aren't supported.

Any other type of control, including custom Power Apps Component Framework controls or web resources, aren't displayed.

To show these unsupported controls or have full functionality of controls, add them to the form header flyout instead of the form header.

Form designer messages related to form headers

When you edit forms using the new or classic form designer, you might see messages related to form headers. Below, you can find details on each message and why you're seeing it.

We've upgraded your form to show a high density header that displays more data. You can edit this setting in the properties of the header

This message is displayed in the form designer when a maker creates a new main form (including via the Save-as action) or edits a main form that hasn't previously been configured for header density.

The framework defaults to high-density header and this message helps makers become aware of that behavior. Makers can override the framework default at any time by manually configuring the form header density as outlined earlier.

This form isn't using high density header, access the setting in the header properties. High density header helps display more data

This message is displayed in the form designer when a maker opens a main form for editing that is configured to use low-density header.

The message helps increase awareness about the high-density header and its benefits.

Column moved to header flyout: The header supports displaying up to four read-only column values. The column *[field display name]* will now only be available from the flyout

This message is displayed in the form designer when a form is using high-density header with the header flyout visible.

High-density header displays read-only values of the first four columns in the header. When makers add a column in the header in the top four positions, it causes an existing column that was directly displayed in the header to become extended and visible only in the header flyout.

The message informs the maker of the change and confirms whether to proceed with the action.

Header column limit exceeded: The header supports displaying up to four read-only column values. Remove unused columns to add more

This message is displayed in the form designer when a form is using high-density header with the header flyout hidden.

High-density header displays read-only values of up to four columns in the header. Because the header flyout is hidden, users are unable to see the additional columns.

The message informs the maker that there are already four columns in the header and prevents adding additional columns in the header that users won't be able to see.

Header doesn't display custom components: The header supports displaying up to four read-only column values. Remove the custom component from the column before adding it to the header

This message is displayed in the form designer when a form is using high-density header with the header flyout hidden.

High-density header displays read-only values of columns in the header. Because the header flyout is hidden, users are unable to see any custom components associated with the columns in the header.

The message informs the maker that they're trying to add a column with an associated custom component to the header and they must remove the custom component before adding the column to the header. This is because users won't be able to see the custom component in the header.

Header displays read-only column values: The header supports displaying up to four read-only column values. To provide editability to the user, add the column to a section in the form

This message is displayed in the form designer when a form is using high density header with the header flyout hidden.

High-density header displays read-only values of columns in the header. Because the header flyout is hidden, users will be unable to edit column values.

The message informs the maker that any columns added to the header will be read-only and that any columns they want users to edit should also be added to a section in the form.

Header column values aren't editable: Moving a column from the body to the header will display as a read-only value. To maintain editability, copy the column to the header

This message is displayed in the form designer only for forms using high-density header with the header flyout hidden.

High-density header displays read-only values of columns in the header. Because the header flyout is hidden, users will be unable to edit column values.

The message informs the maker that they're trying to move a column from the form body to the form header. Doing so will make it read-only. It gives the maker the choice of moving the column to the header or adding a copy of the column to the header. Moving the column to the header will make the column unavailable in the original location on the form body for users to edit. Adding a copy of the column to the header will leave the column in the original location as-is, ensuring that users can continue to edit it within the form body.

This form is using high density header. For the best authoring experience with this form, use the new form designer

This message is displayed in the classic form designer when a maker opens a main form for editing and it's configured to use high-density header.

The classic form designer doesn't provide a WYSIWYG authoring experience. It also doesn't detect and prevent or warn makers about the implications of changes they make to the form header. For example, when you edit a form that's using high-density header with the header flyout hidden, the classic form designer won't prevent makers from adding more than four columns to the header even though these columns won't be available to users.

The message informs the maker that when editing a form that's using high-density header, they should use the new form designer. This helps ensure the maker is aware of the impact of their changes to the form header.

See also

Overview of the model-driven form designer

Create, edit, or configure forms using the form designer

Add, configure, move, or delete columns on a form

Add, configure, move, or delete components on a form

Add, configure, move, or delete sections on a form

Add, configure, move, or delete tabs on a form

Add and configure a subgrid component on a form

Add and configure a quick view component on a form

Configure a lookup component on a form

Using the tree view in the form designer

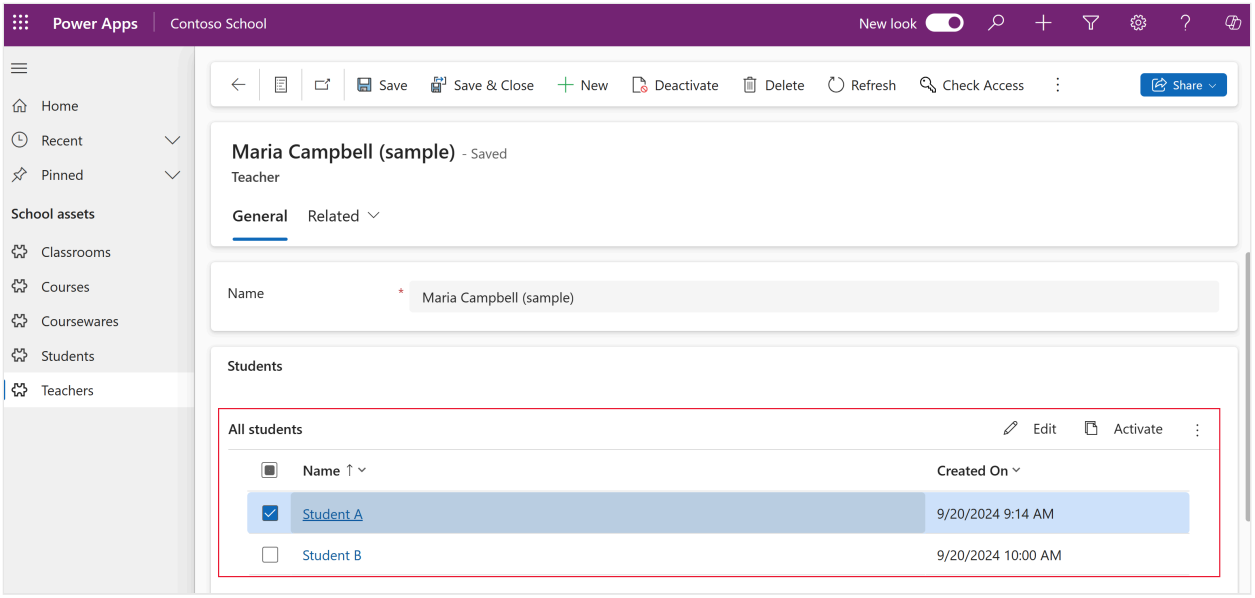
Create and edit columns

Last updated on 04/03/2026

Add and configure a subgrid component on a form

Article • 09/20/2024

A form that displays the details of a table record can use a subgrid component to display a read-only list of related or unrelated records in a tabular format. From the subgrid, app users can create new rows or open an existing row to edit. Makers add and configure a subgrid component using the form designer.



This article explains how makers can add a subgrid component to a model-driven app form. To learn how to add an editable grid to a model-driven app form, go to Make model-driven app views editable using the editable grid control.

Add a subgrid component

1. Sign in to Power Apps, and select the environment you want.
2. Select **Solutions** in the left navigation pane, and then open the solution that contains the table you want.
3. Select **Tables**, and then open the table that you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
4. Select the **Forms** area, and then open the form you want to edit.
5. In the form designer, select **Component** on the command bar.
6. Drag a **1-column section** on the left **Components** pane onto the form.
7. With the **New Section** selected, select **Subgrid** from the left **Components** pane.
8. On the **Select subgrid views** pane:

- Typically app users want to view related records. When **Show related records** is selected, only tables with a relationship are displayed.
- Select the **Table** you want.
- Select the **Default view** from the table. When an existing view isn't suitable for the app, create or edit a view for the table.

9. Select **Done**.
10. Select **Save**, and then select **Publish**.

Configure a subgrid component

These are the properties available to configure when using a subgrid component on a form using the form designer.

Expand table

Area	Name	Description
Display options	**Label**	The localizable label for the subgrid visible to users. This property is required.
Display options	**Name**	The unique name for the subgrid that is used when referencing it in scripts. The name can contain only alphanumeric characters and underscores. This property is required.
Display options	**Hide label**	The label can be hidden on the form.
Display options	**Hide on phone**	The subgrid can be hidden to render a condensed version of the form on phone screens.
Display options	**Show related records**	When selected, the subgrid displays only records related to the current record that is displayed on the form. The **Table** drop-down list is also filtered to only list tables that are related to the current table.
Display options	**Table**	The table whose records you want to display in the subgrid. When **Show related records** is selected, the list of tables is filtered to show only tables that are related to the current table. In addition to the table name, the name of the lookup column is also displayed in parentheses.

Area	Name	Description
Display options	Default view	The view of the table selected in the **Table** property that will be used to get and display the list of records in the subgrid.
Display options	Team template	Note: This option is only available with the user table when the **Default view** is **Associated Record Team Members**. When available, sets the team template to use for the subgrid. More information: Create a team template to control access
Display options	Allow users to change view	When selected, app users can change from the **Default view** to another view of the table selected in the **Table** property.
Display options	Show all views	When selected, app users can change from the **Default view** to all other views of the table selected in the **Table** property. This property is only available when **Allow users to change view** is selected.
Display options	Hide search box	When selected, the **Search** box won't appear on the upper right of the subgrid.
Display options	Selected views	A list of views of the table selected in the **Table** property that app users can change from the **Default view**. This property is only available when **Allow users to change view** is selected and **Show all views** is cleared.
Display options	Default chart	Select which chart to show if **Show chart only** is selected.
Display options	Show chart only	Rather than a list of records a chart is displayed.
Display options	Allow users to change chart	When **Show chart only** is selected, app users can change the chart displayed in the subgrid.
Display options	Maximum number of rows	Determines the maximum number of records to display in the subgrid. The minimum number of rows displayed is 2 and the maximum is 250.
Display options	Use available space	Note: This property only works with the legacy web client. It has no effect on Unified Interface. Determines whether the form allows space for two records and will expand the space as the number of records increases. If the number exceeds the **Maximum number of rows**, people can navigate to additional pages to view the records. If **Use available space** isn't chosen the form provides space for the number of

Area	Name	Description
		records defined by **Maximum number of rows** and people can navigate to additional pages to view any additional records.
Formatting	**Component width**	When the section containing the subgrid has more than one column, you can set the column to occupy up to the number of columns that the section has.

See also

Overview of the model-driven form designer

Create, edit, or configure forms using the form designer

Add, configure, move, or delete components on a form

Add, configure, move, or delete sections on a form

Feedback

Was this page helpful? Yes No

Provide product feedback

Add and configure a quick view component on a form

Summarize this article for me

A main form that displays the details of a record can use a quick view component to display read-only details of a related record. This view of a related record is known as a lookup. The data displayed by the quick view component is defined by the quick view form of the related table. When there is no related record, such as a lookup, the quick view component is automatically hidden.

Add a quick view component

You add a quick view component the same way you add any other component, which is from the **Components** tab in the form designer. More information: Add, configure, move, or delete components on a form

Configure a quick view component

These are the properties you can configure when using a quick view component on a form using the form designer.

Expand table

Area	Name	Description
Display options	**Label**	The localizable label for the quick view visible to users. This property is required.
Display options	**Name**	The unique name for the quick view that is used when referencing it in scripts. The name can contain only alphanumeric characters and underscores. This property is required.
Display options	**Hide label**	When selected, the quick view label is hidden.
Display options	**Quick view forms**	A list of quick view forms that are displayed to app users. To configure the list of quick view forms Select **Select forms ...**, and then in the **Lookup** drop-down list select a lookup column where you want to display a quick view form.

Area	Name	Description
		Depending on the lookup column you select in the **Lookup** drop-down list, you see lists that let you select quick view forms for one or more tables.

See also

Overview of the model-driven form designer
Create, edit, or configure forms using the form designer
Add, configure, move, or delete columns on a form
Add, configure, move, or delete components on a form
Add, configure, move, or delete sections on a form
Add, configure, move, or delete tabs on a form
Configure header properties in the form designer
Add and configure a sub-grid component on a form
Configure a lookup component on a form
Using the tree view in the form designer
Create and edit columns

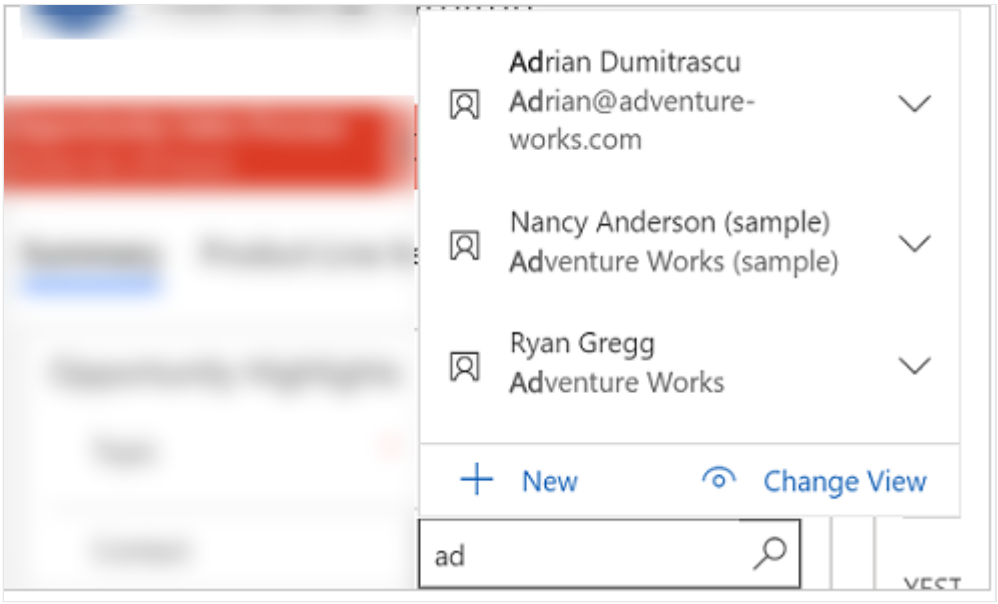
Last updated on 03/04/2026

Configure a lookup component on a form

Article • 08/05/2022

Lookups help a user choose records from a **related** table. A lookup component is automatically added when a lookup column is added to a form.

For example, using a lookup component, you can open an account record that relates to a sales invoice record.



Learn more about the lookup field user experience

Configure a lookup component

Makers configure a lookup component using the form designer.

These are the properties available to configure when using a lookup component on a form using the form designer.

Area	Name	Description
Display options	**Label**	The label text that is displayed next to the lookup column.
Display options	**Hide label**	When enabled, the label isn't displayed.
Display options	**Hide on phone**	When enabled, the lookup column won't display when running the app on a phone.

Area	Name	Description
Display options	**Hide**	Don't display the lookup column. The lookup column can then be displayed using code.
Display options	**Lock**	Make this column so that it can't be removed by you or other makers until the **Lock** property is disabled.
Display options	**Read-only**	When this property is enabled, users can't change the value in the lookup column.
Display options	**Disable most recently used items**	Disabling this option allows users to view recently used rows in the lookup dropdown list. Enabling this option disables the recent items.
Display options	**Use Main Form Dialog for Create**	Enabling this option allows users to create rows in a pop out dialog for this lookup field, instead of closing the current form and navigating away.
Display options	**Use Main Form Dialog for Edit**	Enabling this option allows users to edit rows in a pop out dialog for this lookup field, instead of closing the current form and navigating away.
Display options	**Default view**	The view of the table selected in the **Table** property that can be used to get and display the list of rows that app users can select in the lookup drop-down list.
Display options	**Allow users to change view**	When selected, app users can change from the **Default view** to another view of the table.
Display options	**Show all views**	When selected, app users can change from the **Default view** to all other views of the table selected in the **Table** property. This property is only available when **Allow users to change view** is selected.
Display options	**Selected views**	A list of views of the table selected in the **Table** property that app users can change to from the **Default view**. This property is only available when **Allow users to change view** is selected and **Show all views** is unselected.
Formatting	**Form field width**	Determines the width in number of columns for the lookup column.

Area	Name	Description
Filtering	**Filter by related rows**	When this is enabled, you can filter this lookup based on a row related to both the current table and this lookup. The rows that display in this lookup when users search for a row will then have additional filtering applied. This helps provide more relevant searches when setting the value of the lookup. By default, this is turned off.
Filtering	**Relationship to current table**	This dropdown list defines the relationship from the table of the related row you want to filter by to the target lookup's table. The possible relationship combinations will be listed in the table following this one.
Filtering	**Relationship to this lookup's table**	This dropdown list defines the relationship from the table of the related row you want to filter by to the target lookup's table. For the possible relationship combinations, go to Table relationships available for a lookup column.
Filtering	**Allow users to turn off filter**	When this is enabled, users will have the option to turn off the filter you define here.
Components	**+Component**	Configure a control for the lookup column, such as the form component control.

Possible relationship combinations for a lookup column

Relationship to current table (Related table > Current table)	Relationship to this lookup's table (Related table > Lookup's table)	Available?
1:N	1:N	Yes
1:N	N:1	Yes
1:N	N:N	Yes
N:1	1:N	No
N:1	N:1	No
N:1	N:N	No
N:1	Self	Yes

Relationship to current table (Related table > Current table)	Relationship to this lookup's table (Related table > Lookup's table)	Available?
N:N	1:N	No
N:N	N:1	No
N:N	N:N	No
N:N	Self	Yes

See also

Overview of the model-driven form designer

Table relationships

Add agent response with the form designer (preview)

08/06/2025

[This article is prerelease documentation and is subject to change.]

Add the agent response component to a model-driven app form, which can directly call a Microsoft Copilot Studio topic and display the response to the user. This component builds on the Agent API executeEvent and simplifies usage by not requiring a new custom component. The implicit context of the app, page, and record are available in Copilot Studio.

This component supports the following responses from Copilot Studio:

- Markdown
- Adaptive card
- Image
- Video

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Prerequisites

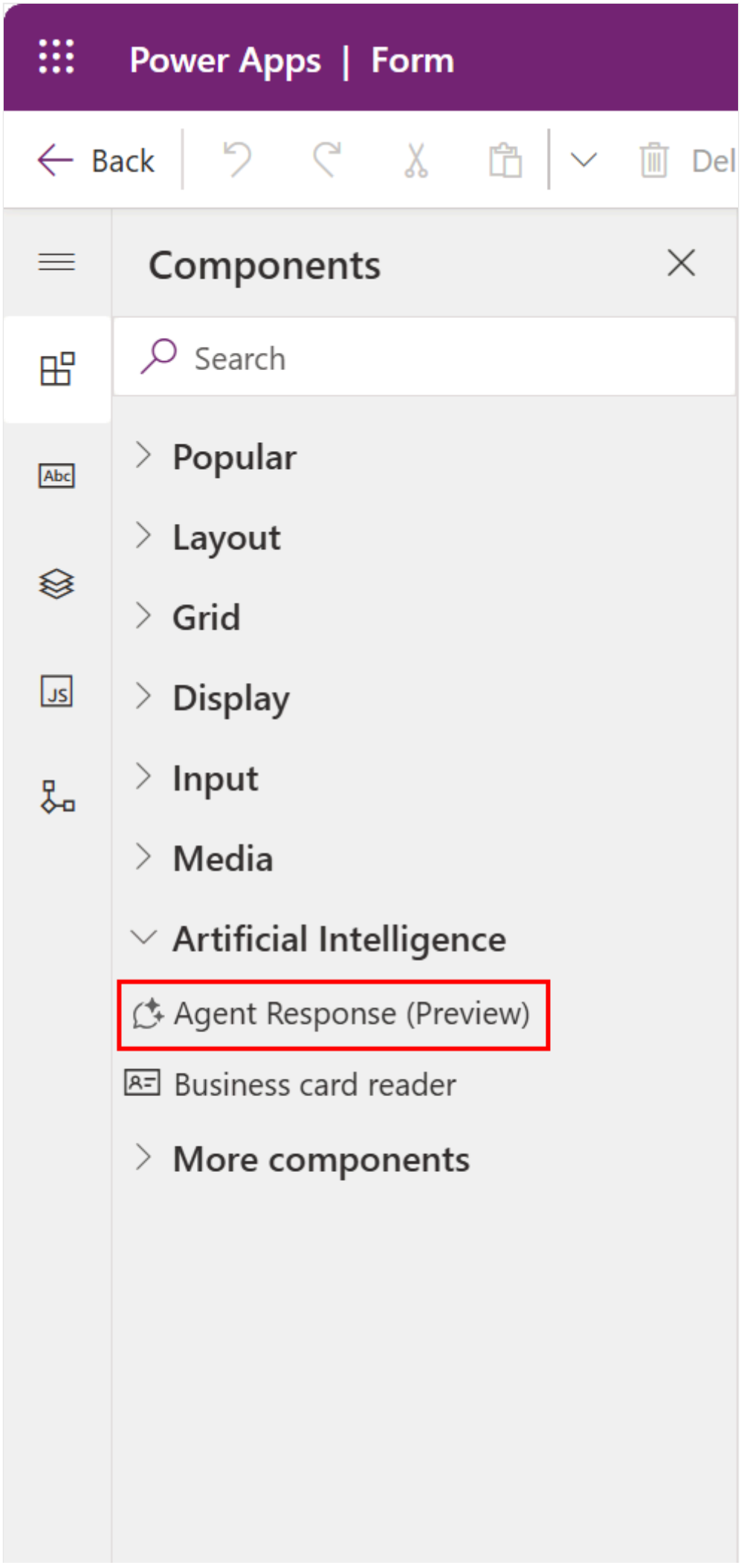
Requires the **Allow users to analyze data using an AI-powered chat experience in canvas and model-driven apps** environment setting enabled for model-driven apps. More information: Copilot environment settings

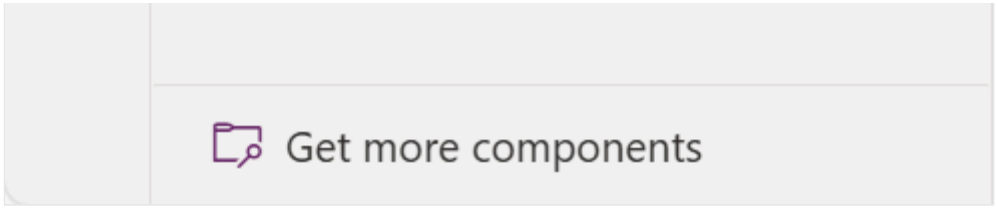
Add an agent response component to a form

1. Open the form designer to create or edit a form. More information: Create a form or Edit a form
2. In the form designer, select **Component** on the command bar.

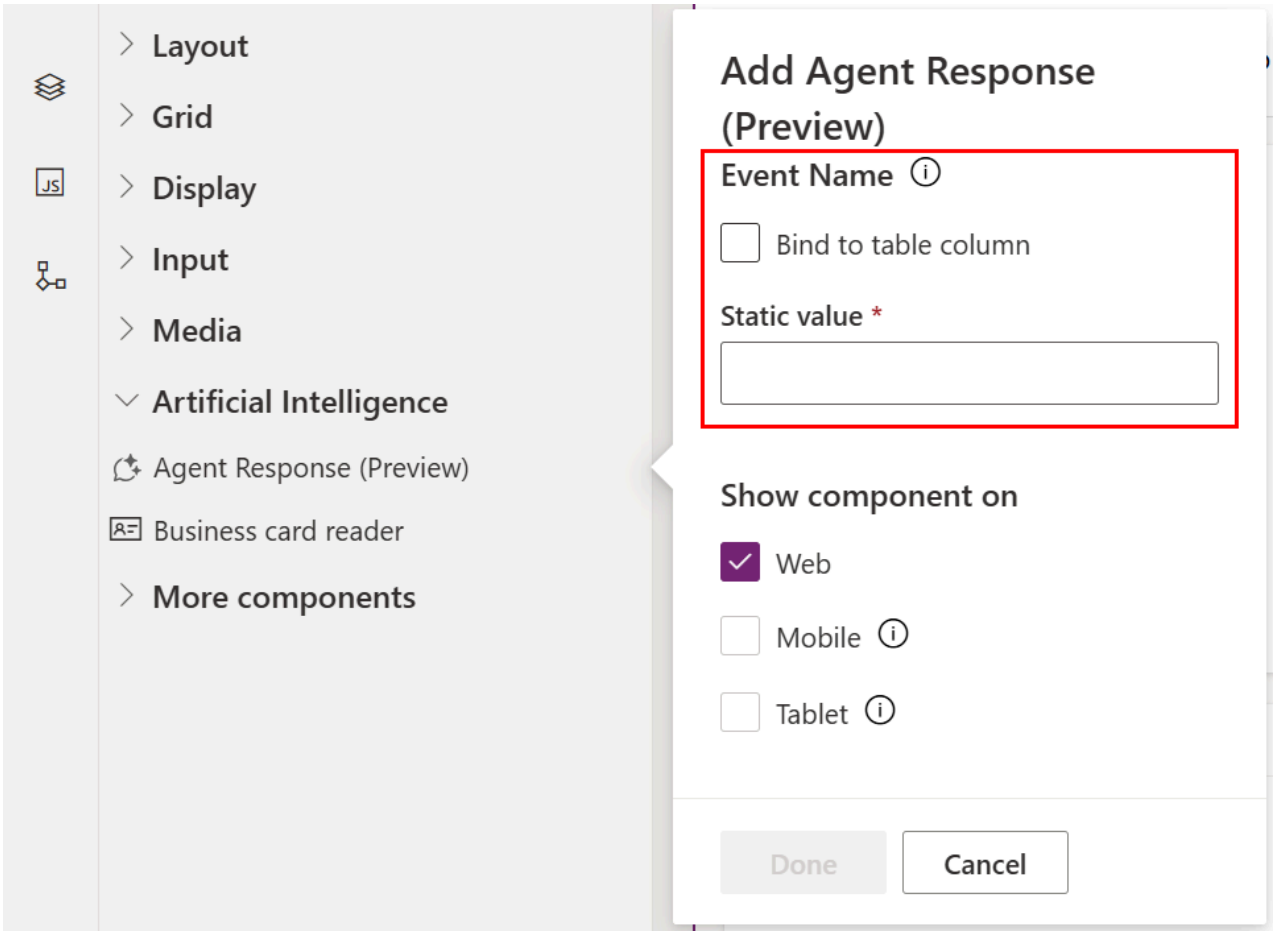
3. In the Components pane, expand **Artificial Intelligence**

4. Drag **Agent Response** onto the form canvas.


Power Apps | Form
Back
Del
Components
Search
Popular
Layout
Grid
Display
Input
Media
Artificial Intelligence
Agent Response (Preview)
Business card reader
More components



5. In the **Add Agent Response** dialog, enter the **Event name** from the topic in Copilot Studio in the **Static value** box. More information: Find the event name for a topic in Copilot Studio



> **Note**
>
> The **Bind to table column** option isn't supported for this component.

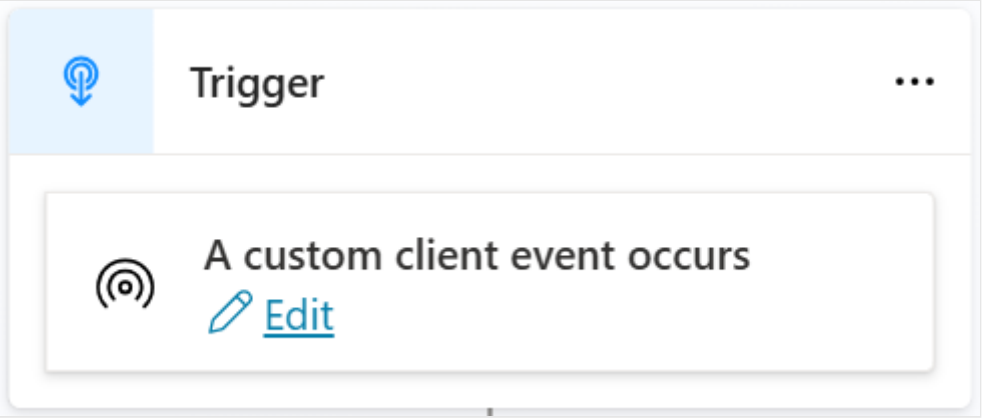
6. Select **Save and publish** on the command bar.

7. Open the form in a model-driven app to test the component.

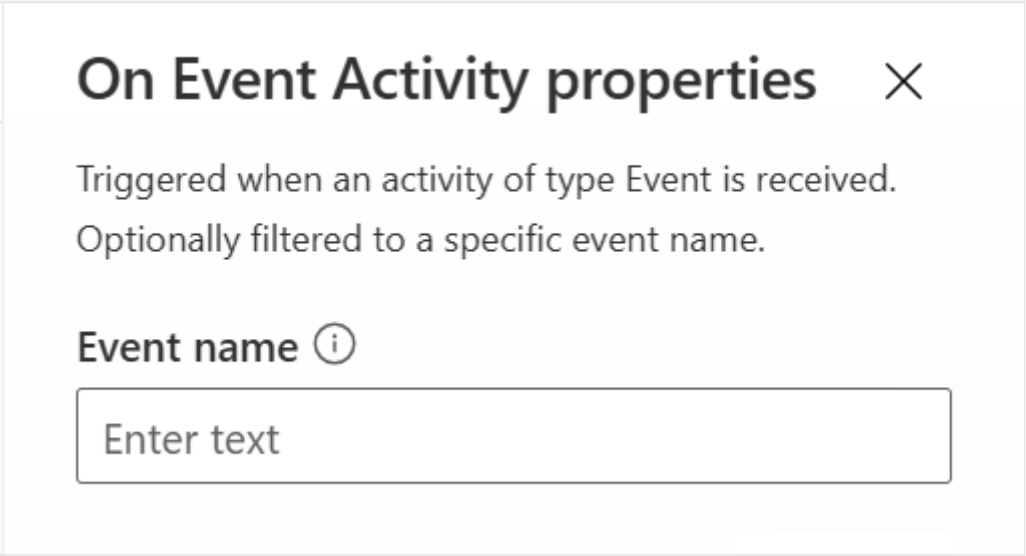
> **Note**
>
> The form designer preview displays the warning message "Agent Response is only available when you play the app" since live preview isn't yet supported.

Find the event name for a topic in Copilot Studio

1. Open the topic in Copilot Studio and find the **Trigger** node.
2. Verify the trigger type is **A custom client event occurs** or **Event received**.
3. Within the **Trigger** node, select **Edit**.



4. Within the **On Event Activity properties** pane, copy the **Event name** property string value to be passed from the agent response component. More information: Change the trigger for a topic



Accessing app context

When the agent response component calls the Copilot Studio topic, it passes context for the app through a set of variables. The following are context variables available as Copilot Studio global variables.

Expand table

Variable	Description
`Global.PA__Copilot_Model_PageContext.pageContext.id.guid`	ID of the table record on the main form
`Global.PA__Copilot_Model_PageContext.pageContext.entityTypeName`	Logical name of the table in the main page
`Global.PA__Copilot_Model_PageContext.pageContext.pageName`	Name of the main page
`Global.PA__Copilot_Model_PageContext.pageContext.pageType`	Type of the main page
`Global.PA__Copilot_Model_AppUniqueNameContext.appUniqueNameContext.appUniqueName`	Unique name of the model-driven app

Accessibility

When the Copilot Studio response is an Adaptive Card, ensure the following best practices are followed to have an accessible experience.

- Design best practices - Adaptive Cards
- Image - Adaptive Cards

Limitations

- Agent response component properties currently not supported:
  - Label
  - Hide label
  - Hide on phone
  - Lock
  - Read-only
  - Component width
  - Component height
  - Bind to table column
- Agent response component doesn't support passing additional context to Copilot Studio. Create a custom component using Agent API executeEvent to support passing more context.

- Agent response component only supports a read-only adaptive card. Create a custom component using Agent API executeEvent to support client side logic.
- Agent response component isn't supported for mobile or tablets.

Related articles

FAQ for Agent APIs and Agent response component

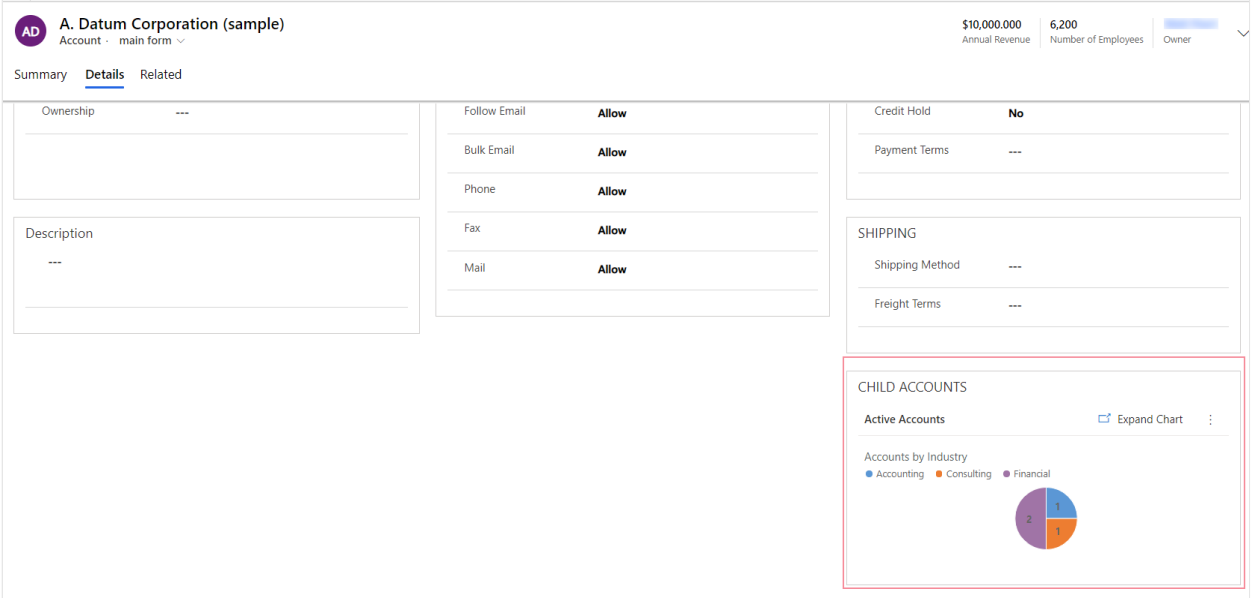
Agent API executeEvent

Overview of the model-driven form designer
Add, configure, move, or delete components on a form

Add a chart to a form

Article • 03/13/2023

A chart can be added to a form or an existing list to a chart can be updated. For example, the Child Accounts filtered sub-grid can be changed to display a chart instead of a list on the account main form.



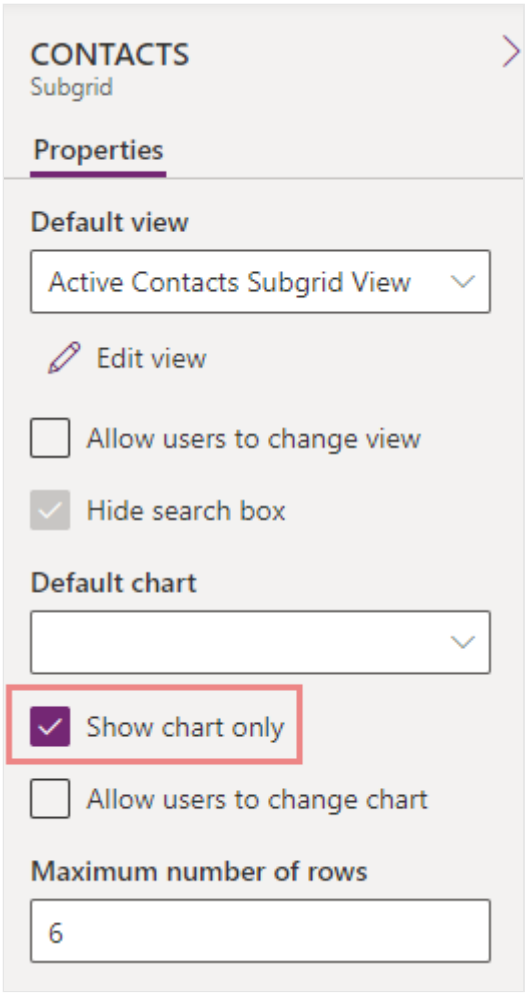
1. Sign into Power Apps, and then select **Solutions** on the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. Open the solution and then open the table.

3. Select the **Forms** area, and then open the form where you want to add a chart.

4. On the form canvas, select an existing subgrid or add one. More information: Add and configure a subgrid component on a form

5. On the right properties pane for the subgrid, select the **Show chart only** option. When enabled, this option displays the view in a chart format.



6. Configure the subgrid properties for the chart, such as the default chart to display and whether users can change the chart displayed. More information: Configure a subgrid component

7. **Save** and **Publish** the form to make the customization available to users.

Add a chart to a form (classic)

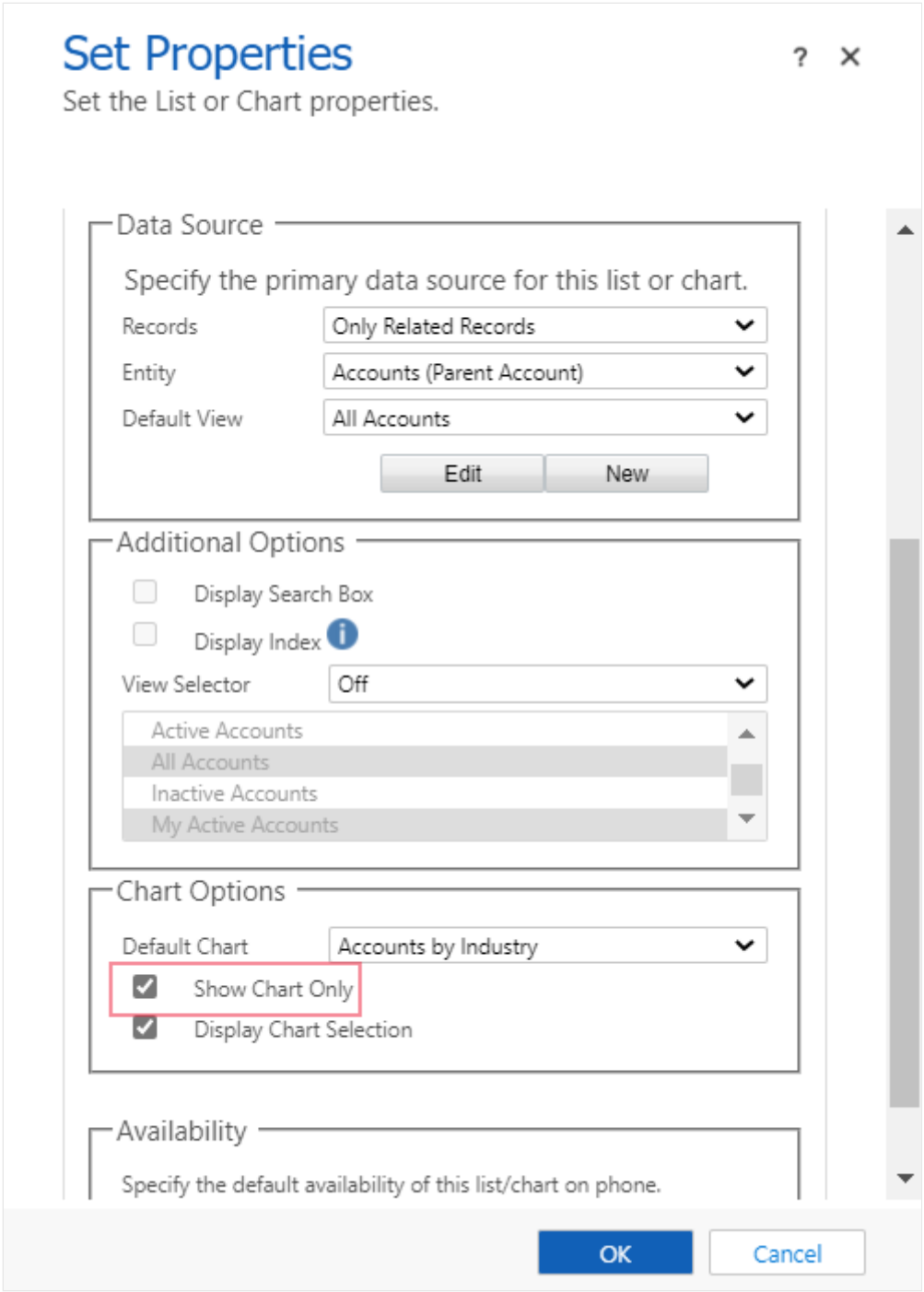
1. Sign into Power Apps, and then select **Solutions** on the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. Open the solution with the required table and open this table. Then select the **Forms** area.

3. Open the required form. In the form designer select **Switch to classic** from the command bar.

4. Add a sub-grid to the form, or to change an existing list in a sub-grid, double-click a sub-grid on the form.

5. On the **Set Properties** page for the sub-grid, select the **Show Chart Only** option. When enabled, this option displays the view in a chart format.



6. On the **Set Properties** page for the sub-grid, consider from the following options that affect how the chart is displayed, and then select **OK**. For more information about these properties, see Set properties for a model-driven app chart or list included in a dashboard.

   - **Records**:
     - **Only Related Records**: Shows the rows based on the selected view, but only with rows related to the table row.
     - **All Row Types**: Shows all the rows based on the selected view.

- **View Selector**: When enabled, users can change to a different view, which displays the chart with different data and filtering that are associated with the view:
  - **Off**. Don't display the view selector. Users won't be able to change views at runtime.
  - **Show All Views**. Provide a full list of views associated with the value set in the **Table** property.
  - **Show Selected Views**. Select this setting to limit the list of views available at runtime. To select the specific views to be displayed, hold down the Ctrl key and tap or select each view that needs to be included.

- **Display Chart Selection**: When enabled, users can change the type of chart, such as column, bar, or pie, on the form. If the user changes the type of chart, the settings aren't saved. The chart type reverts to the **Default Chart** setting when the form is closed.

7. Publish your customization.More information: Publish changes

See also

Create a model-driven app system chart

Add a Power BI Visual to a model-driven app

Configure a map on a form

Article • 08/31/2023

By default, the map control is configured on the main form for both the account and contact tables, which provides the ability to display a map on table records. Although not configured by default, the map control can be added to the system user table. The map control can also be used with some Dataverse standard tables and Dynamics 365 apps tables. The map control can't be used with custom tables. More information: Tables that can be used with the map control

When enabled, the map displays the location specified in the address composite columns for the given record.

AD

A. Datum Corporation (sample) - Saved

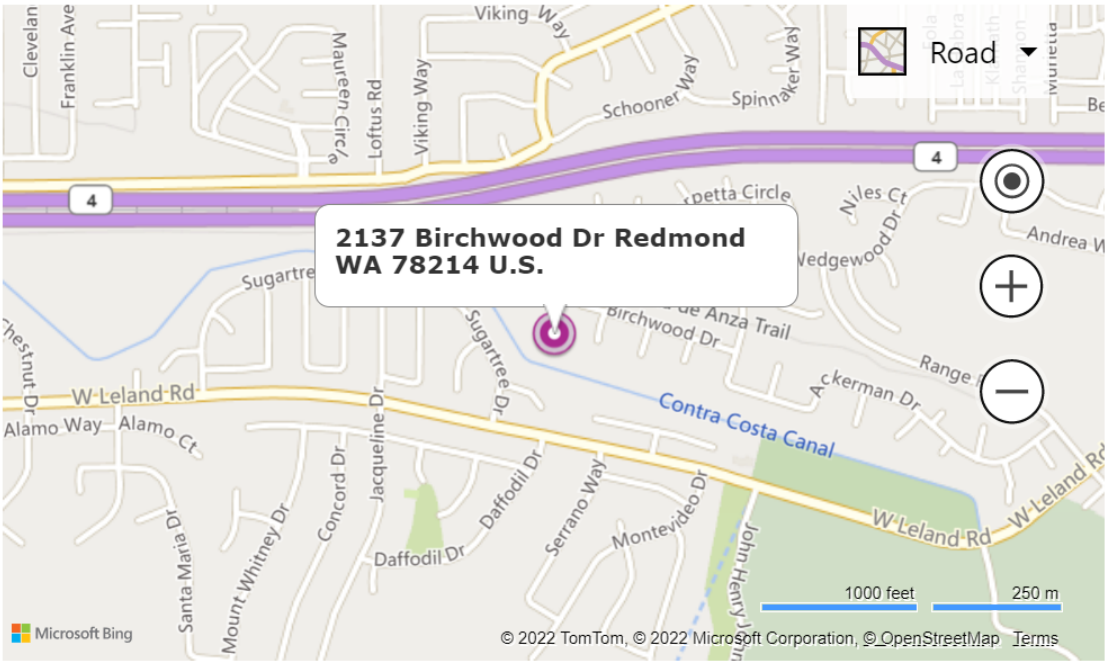
Account · Custom main form

Summary | Details | Related

ADDRESS

Address 1: Street 1	2137 Birchwood Dr
Address 1: Street 2	---
Address 1: Street 3	---
Address 1: City	Redmond
Address 1: State/Province	WA
Address 1: ZIP/Postal Code	78214
Address 1: Country/Region	U.S.

Get Directions



Important

- To use the map component the **Bing Maps** setting must be **On** under the **Embedded content** section of environment settings in the Power Platform admin center. More information: **Manage Bing Maps for your organization**
- A form can only have one map component.

Tables that can be used with the map control

- Contact
- Account
- System User
- Competitor[1]
- Invoice[1]
- Quote[1]
- SalesOrder[1]
- Lead[1]

[1]Requires a customer engagement app.

Add and configure a map for a system form

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane, and then select the table where you want to configure a map on the main form. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. Select the **Forms** area, and then open the system main form you want.
4. On the **Components** left pane, expand **Display**, and then select **Map**. Or, if the map component is already on the form, select it on the form designer canvas.

View and edit map properties

Area	Property	Description
Display options	**Label**	**Required**: A label to display for the map.
Display options	**Hide label**	Whether the label should be displayed.

Area	Property	Description
Display options	**Hide**	Showing the map is optional and can be controlled using business rules or scripts. More information: Visibility options
Display options	**Bing Maps address**	Choose which address should be used to provide data for the map.
Formatting	**Component width**	When the section containing the map has more than one column, you can set the column to occupy up to the number of columns that the section has.
Formatting	**Use all available vertical space**	You can allow the map height to expand to available space.

Configure a map using the classic form editor

You can remove the maps area in the form editor or add it back by using the **Bing Maps** button on the **Insert** tab of the classic form editor. More information: Configure Bing Maps to be displayed on forms

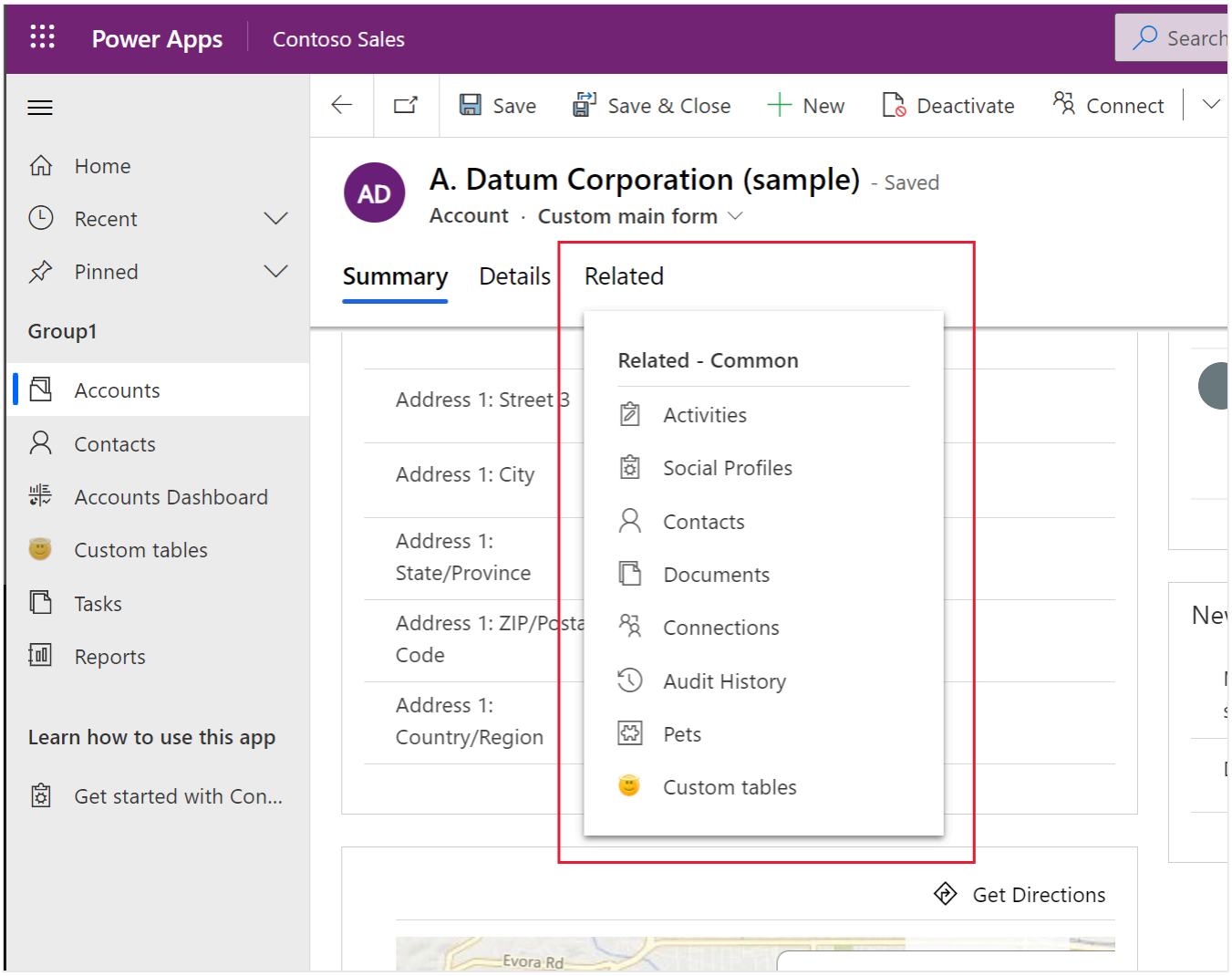
See also

Create and design model-driven app forms

Add table relationships to a form

09/25/2025

Creating a table relationship adds the navigation within a form that allows app users to view a list of related records from the **Related** tab in a model-driven app.



Each table relationship has properties to control the relationship behavior. More information: Table relationship behavior

Table relationships that are configured to be displayed can be hidden using the form designer.

Add, edit, or create a related table in form designer (preview)

[This topic is pre-release documentation and is subject to change.]

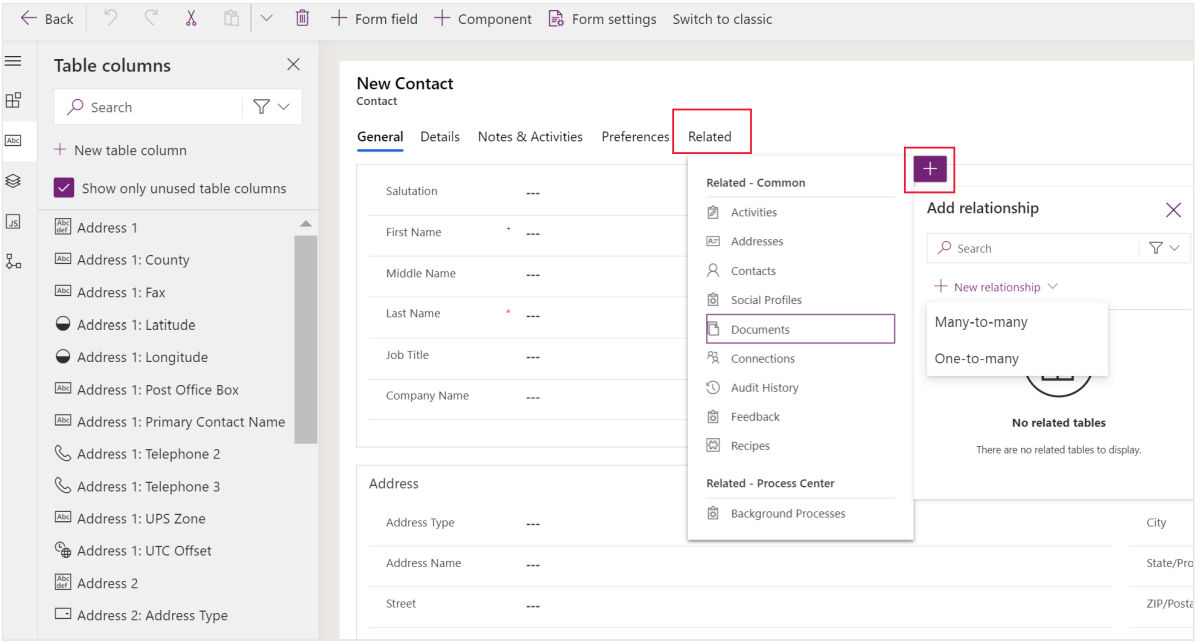
> **Important**

This is a preview feature.

1. Sign into Power Apps
2. Select **Solutions** on the left navigation pane, open the solution that has the table you want, and then open the table. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Forms**, and then open the main form where you want to add a related table.
4. Select the **Related** tab. The **Related** menu appears:

   - Add and edit an existing relationship. Select a table in the list to display the table properties in the right pane. Select **Edit relationship** to make changes to the relationship. More information: Add advanced relationship behavior

   - Add a new relationship. Select + to **Search** for another relationship or select **New relationship** to create either a **One-to-many relationship** or **Many-to-many** relationship. More information: Add a One-to-many relationship and Add a Many-to-many relationship



5. **Save** and **Publish** the change.

Hide a relationship

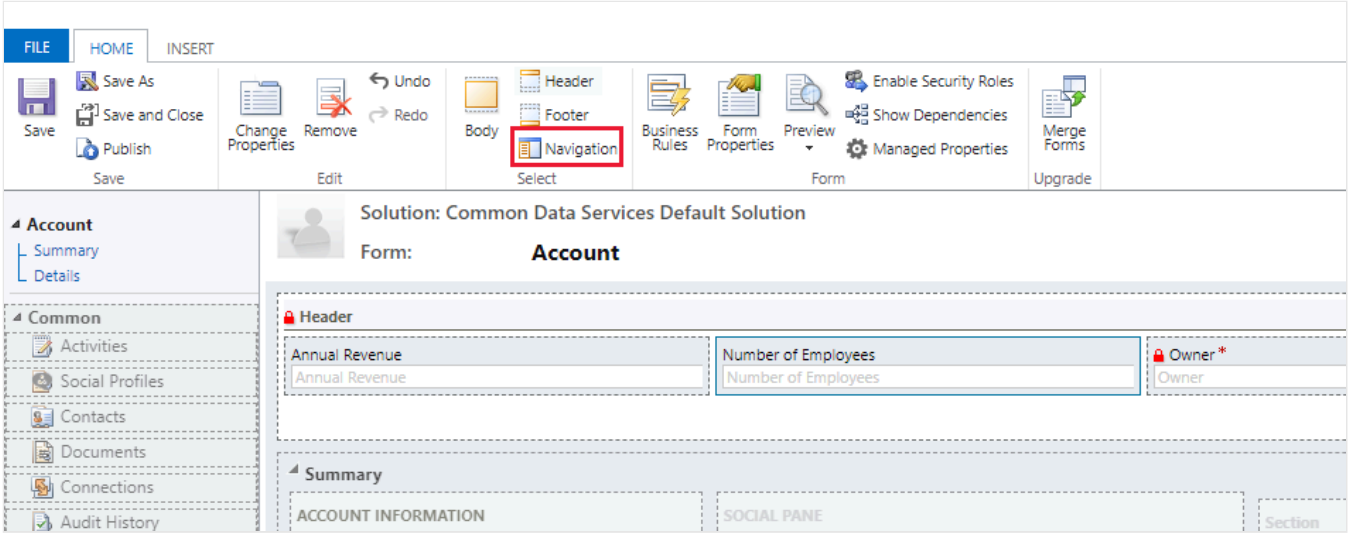
1. In form designer, select the **Related** tab, and then select the table where you want to hide a relationship.
2. On the right properties pane, under **Display options** select **Hide**.
3. **Save and Publish** the change.

Relationship management limitations in form designer

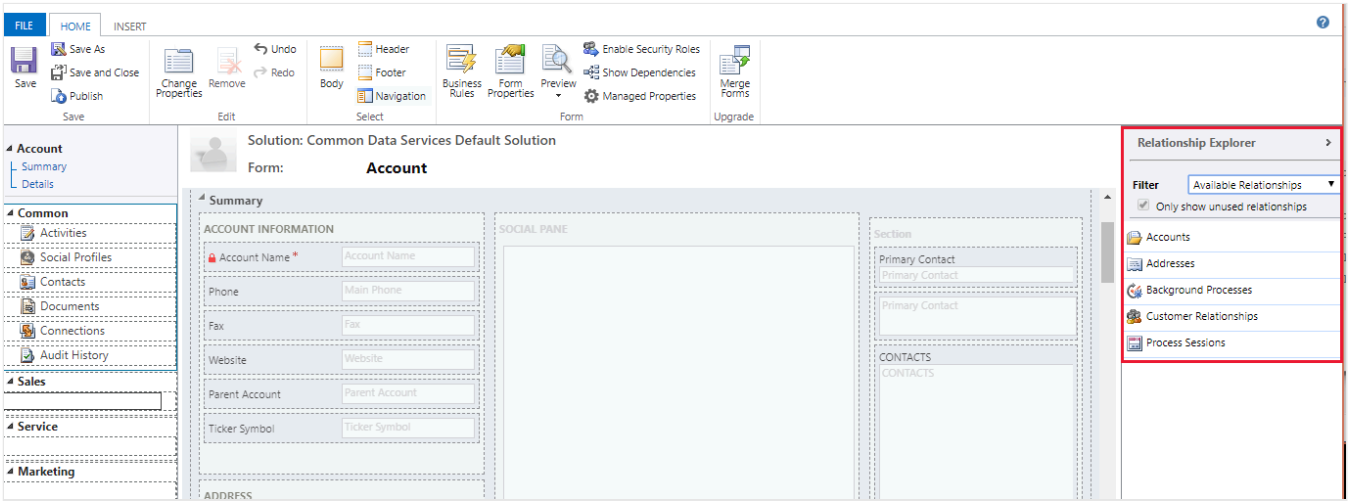
- You can edit display options for system relationships but you can't remove the relationship from the related menu. However, you can hide the relationship. More information: Hide a relationship
- You can't edit or remove self many-to-many relationships.
- You can't add or edit related web resources or external URLs.

Create navigation using the legacy experience

To enable editing navigation first select **Navigation** from the **Select** group on the **Home** tab of the form designer.



In the right pane, from **Relationship Explorer** you can filter by 1:N (one-to-many) or N:N (many-to-many) relationships, or view all available relationships. The **Only show unused relationships checkbox** is disabled and selected. Relationships are added one at a time.

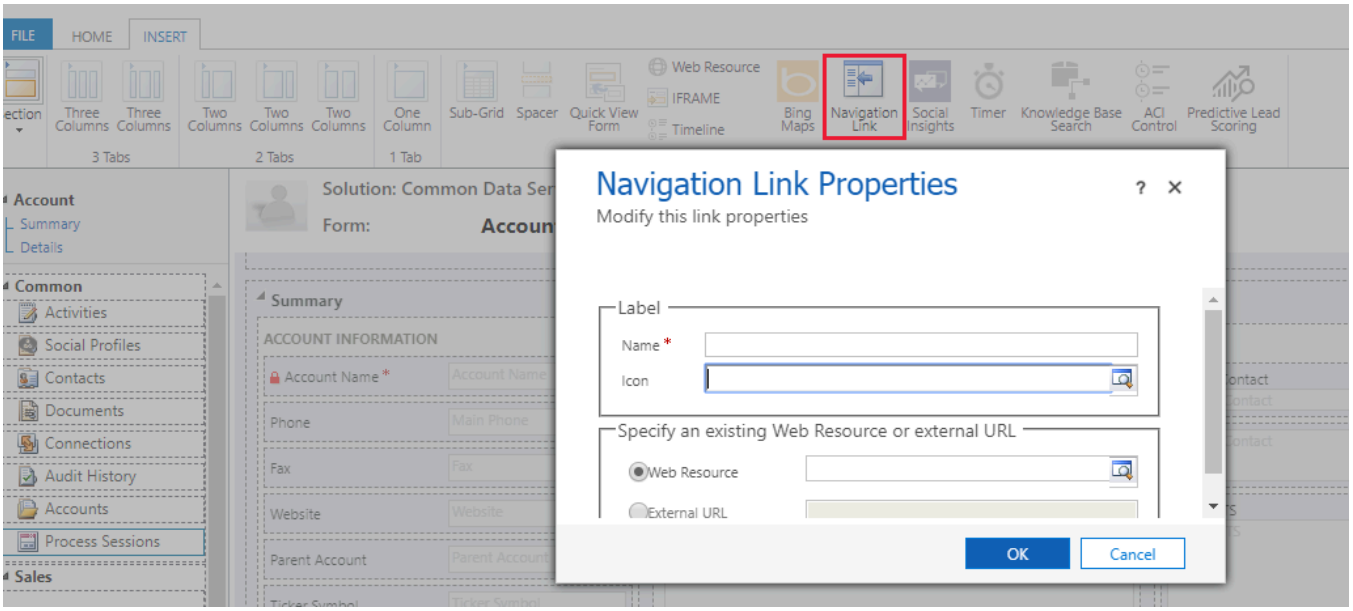


To add a relationship from the **Relationship Explorer** just double-click the relationship and it will be added below the currently selected relationship in the navigation area. Double-click a relationship in the navigation area and you can change the label on the **Display** tab. On the

Name tab you can see information about the relationship. Use the **Edit** button to open the definition of the table.

There are five groups in the navigation area. You can drag them to reposition them and double-click them to change the label, but you can't remove them. These groups will only display when there is something in them. So if you don't want a group to appear, just don't add anything to it.

Use the **Navigation Link** button in the **Control** group of the **Insert** tab to add a link to a web resource or external URL.



Navigation link properties

Navigation links have the following properties:

Expand table

Property	Description
Name	**Required**: Text to display as a label.
Icon	Use a 32x32 pixel web resource. Use a PNG image with a transparent background is recommended.
Web Resource	Specify a web resource to display in the main pane of the form.
External URL	Specify the URL of a page to display in the main pane of the form.

See also

Create and edit table relationships for Microsoft Dataverse

Manage model-driven app forms settings

Article • 01/24/2025

Power Apps forms settings are a set of capabilities that make it easy for model-driven app makers to configure and verify how users access forms in a solution or application. From **Form settings**, there are four different areas a maker can choose from:

- Set security roles for a form.
- Set the form order.
- Set the fallback form for a table.
- Form access checker.

You can access form settings from the **Forms** area in Power Apps or command bar in the form designer.

Set security roles for a form

Different people in your organization might interact with the same data in different ways. Managers might depend on being able to quickly scan information in a row and service people might require a form that streamlines data entry. You can accommodate different requirements by assigning forms to the security roles that different groups of people belong to.

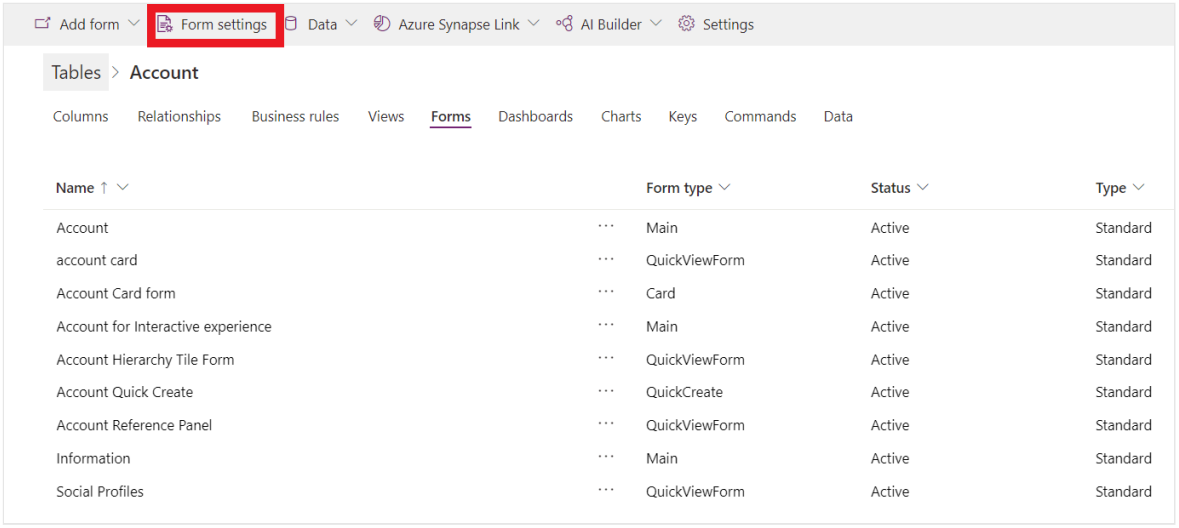
When you have more than one main or other form type defined for a table, you can select which forms users are able to use based on their security roles. Because each table must be able to display a form for any user, at least one form must be designated as a ”fallback” form – a form visible to users whose security roles don't have any forms explicitly assigned to them.

> ⓘ **Note**
>
> - Quick Create, Quick View, and Card forms can't be assigned to security roles.
> - Another option makers have to control form access is to make a main form inactive. You can set an active or inactive state to main forms. This feature was included primarily to manage new forms included when Microsoft Dataverse environments upgrade but you can use it to prevent users from being able to use any main form.

Assign a security role to a form

1. Sign into Power Apps, select **Solutions**, open the solution you want, open a table, and then select the **Forms** area. If the item isn't in the side panel pane, select ... More and then select the item you want.

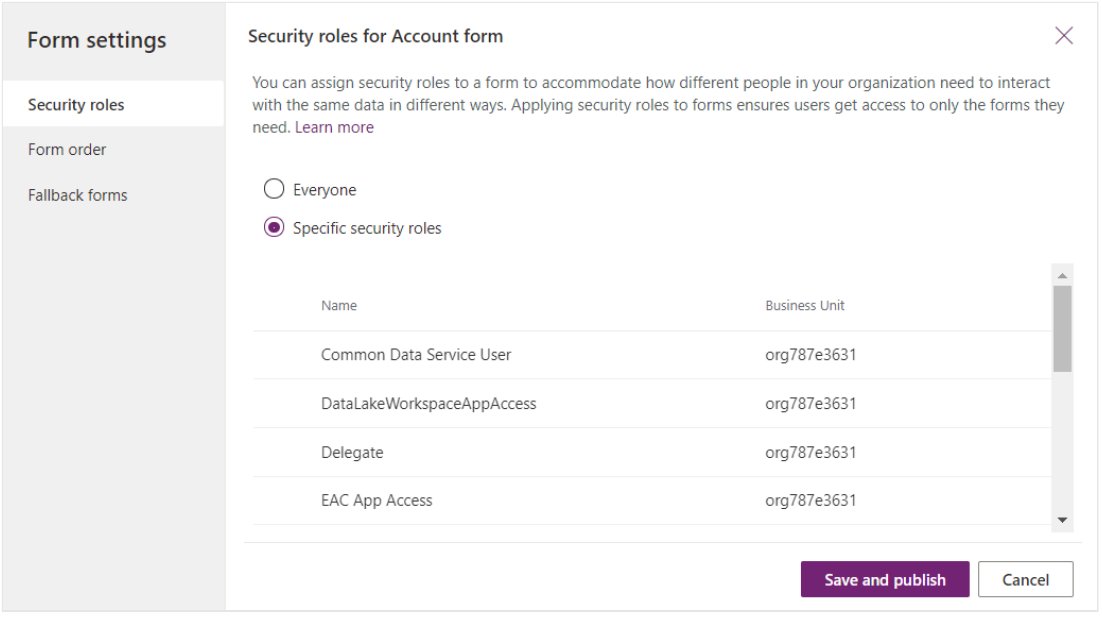
2. Select the main form you want to configure the security roles for a form, and then on the command bar select **Form settings**.



3. In the left navigation on the **Form settings** page, select **Security role**.

4. Choose from the following options:

   - To set the security roles for every user select **Everyone**.
   - To set the access to the form for one or more specific security roles, select **Specific security roles**. Then, select the security roles you want to grant access from the list of security roles that are set up for your environment.



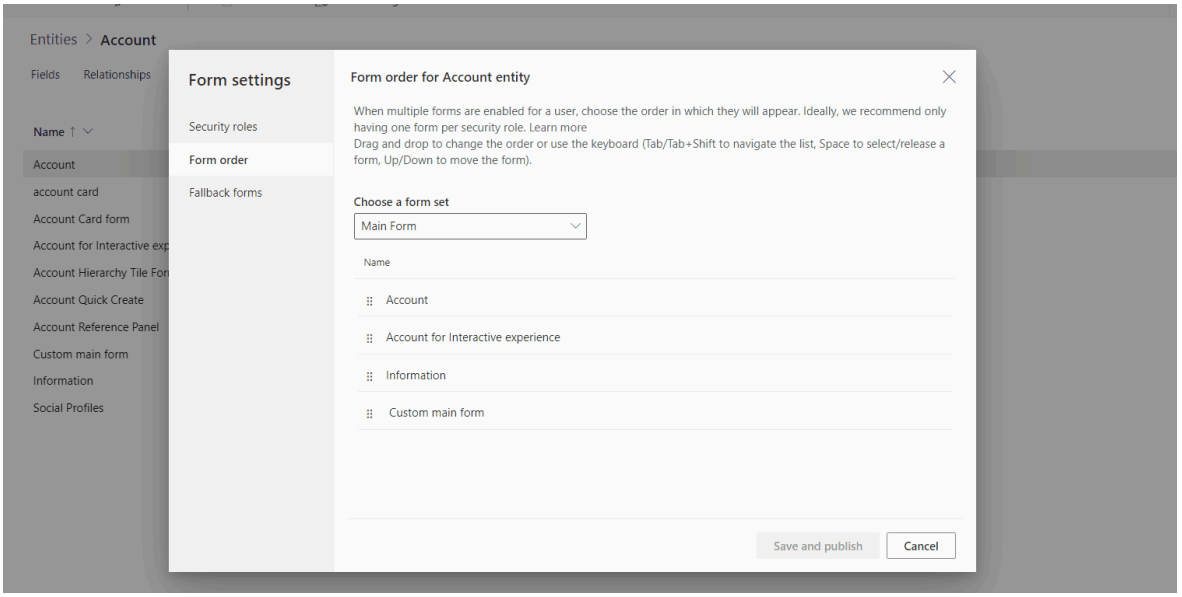
5. Select **Save and Publish**.

Set the form order

When you have multiple main, quick create, quick view, or card forms for a table, you can assign a form order. The form order determines which of the available forms is shown by default. The available main forms can be further controlled by assigning security roles to the form.

Notice that the form order doesn't set who can view the form that is managed by setting security roles. More information: Set security roles for a form

1. Sign into Power Apps, select **Solutions**, open the solution you want, select a table, and then select the **Forms** area. If the item isn't in the side panel pane, select ... More and then select the item you want.
2. Select the main form you want to configure the fallback form for, and then on the command bar select **Form settings**.
3. In the left navigation on the **Form settings** page, select **Form order**.
4. Select the form set that you want, such as **Main Form**, **Quick Create Form**, **Quick View Form, or **Card Form**.
5. Change the order by dragging and dropping the form up and down the list in the order you want it to be shown to your users. You can also use these keyboard shortcuts: Tab/Tab + Shift to move the focus, Space to select or release a form, and Down/Up arrow keys to move a form.
6. Select **Save and publish**.



To set the form order using solution explorer, see Assign model-driven app form order.

Set the fallback form for a table

Setting the fallback form for a table is important to ensure that if a user doesn't have any assigned security roles or there are no assigned security roles for a main form type there's always a “default” form that shows for that table. Fallback forms are only available with main forms and don't apply to quick create, quick view, and card forms.

There are a few different strategies to consider when you manage the fallback form for a table.

All users view the same form

If you don't require multiple forms for a table you don't need a fallback form.

Create a contingency form

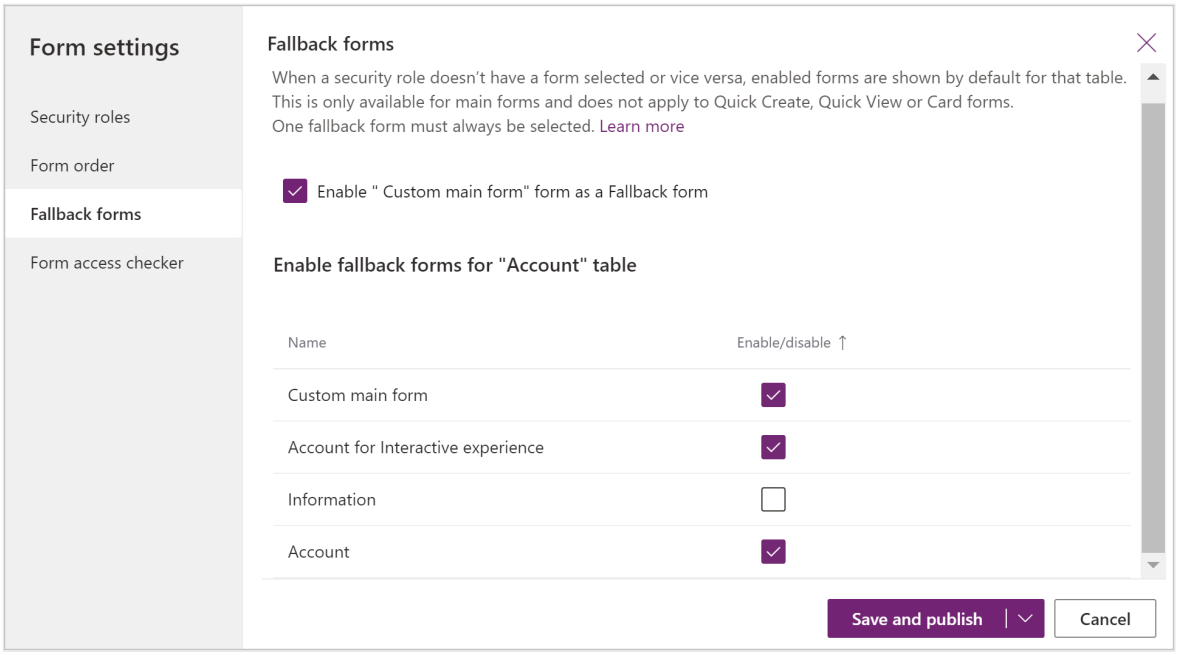
If you're using role-based forms because you want to restrict the information people might view or edit, consider creating a form that has a minimum of information displayed. Then, select specific security roles, but don't select any roles except system administrator. The result is that this form will never be seen by anyone except users with the system administrator security role and anyone whose security roles haven't been associated with a specific form. You could include an HTML web resource in the form with information about why little information is visible in the form and a link to information about how to request being added to a security role that is associated with a from or to include a new security role for a form.

Create a generic form

If you use role-based forms to provide a customized experience based on a user’s role, you can set your least specialized form as the fallback form and configure it to display for everyone. Then, create customized forms for specific security roles and configure those forms to only display for security roles that require them. Don't enable these forms for fallback. Finally, in the **Form settings** configure the **Form order** to specify which forms to display ranking them from most exclusive to least exclusive. Your fallback form is at the bottom of the list. This strategy causes people seeing the form that has been customized for their role as the default form, yet they can use the form selector to select the most common form if they want. Whatever form they select remains their default form until they select a different form.

1. Sign into Power Apps, select **Solutions**, open the solution you want, select a table, and then select the **Forms** area. If the item isn’t in the side panel pane, select ... More and then select the item you want.

2. Select the main form you want to configure the fallback form for, and then on the command bar select **Form settings**.
3. In the left navigation on the **Form settings** page, select **Fallback forms**.
4. To disable a form from the fallback order, clear the checkbox. In this example the **Information** form is removed from the fallback order.



   To enable a form for fallback select the checkbox next to the form to enable it.
5. Select **Save and publish**.

Use form scripting

The Client API form context (formContext) provides a reference to the form or to an item on the form, such as, a quick view control or a row in an editable grid, against which the current code is executed. More information: Client API form context

> **Important**
>
> The Xrm.Page object is **deprecated**, and you should use the **getFormContext** method of the passed in execution context object to return reference to the appropriate form or an item on the form.

See also

Overview of the model-driven form designer

Feedback

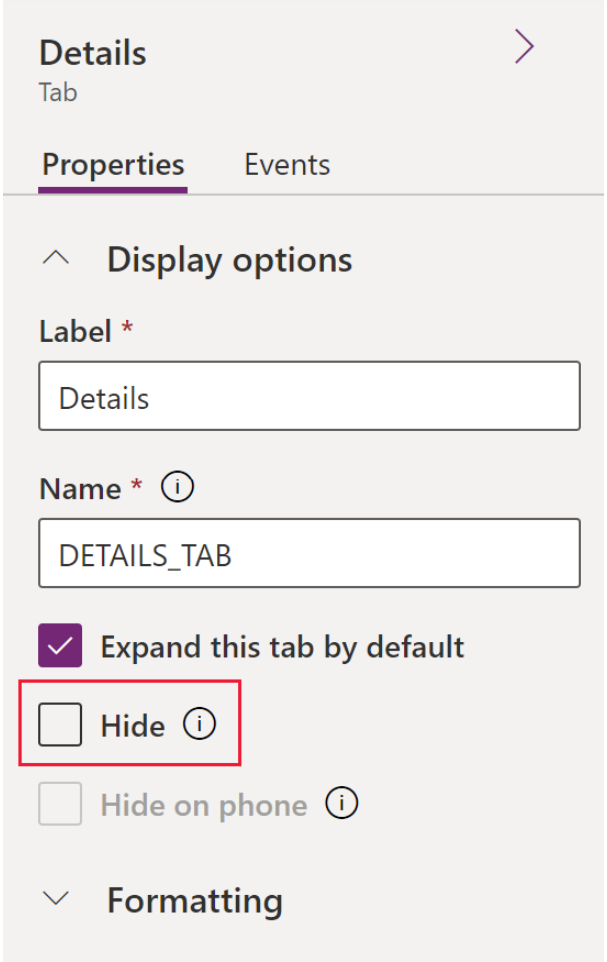
Was this page helpful? Yes No

Provide product feedback

Show or hide model-driven app form elements

Article • 04/08/2022

Several types of form elements have the option to be shown or hidden at app runtime. Tabs, sections, columns, iFrames, and web resources all provide this option. You can change the default setting to view a form element in the form designer by selecting the **Hide** property.



Using form scripts or business rules, the visibility of these elements can be controlled to create a dynamic form to provide a user interface that adapts to conditions in the form.

> **Important**
>
> Hiding form elements is not a recommended way to enforce security. There are several ways people can view all the elements and data in the form when elements are hidden.
>
> Disabling, hiding, or locking a column, control, or form component using the form designer, manually in the form XML, or using a Client API only works to control data access in a Unified Interface or legacy web client app. It doesn't strictly enforce

data access by itself in the apps. Moreover, it doesn't control data access using other ways to connect to Microsoft Dataverse such as using the Web service or a canvas app. It is also important to know that the app will not display an error message or block a save if a required column is null and has been disabled. This ensures that users are never blocked from completing a task. To control access, see **Set security roles for a form** or use **Field-level security**.

Rather than designing forms that depend on scripts to control visibility of options, consider whether a business process flow, a dialog, or switching to a different form may be better suited to meet your requirements. If you do use scripts, make sure that any element that might be hidden is hidden by default. Only show it with scripts when your logic calls for it. This way it will not be displayed in presentations that do not support scripts.

In Unified Interface, for sections where columns don't span more than one column, hiding a column in the section moves the column below up on the form. If a column spans more than two columns in a section, hiding a column in the section that has a control across from it will not move the column below it up on the form. You will see additional white space where the hidden column is in the section.

Client API methods for controlling the visibility of form elements

For information about how to change the form element show or hide status for the header, command bar, and tabs at app runtime, see these client API reference articles:

- formContext.ui.headerSection item (Client API reference)
- formContext.ui.tabs (Client API reference)

Next steps

Overview of the form editor interface

Use the model-driven app main form and its components

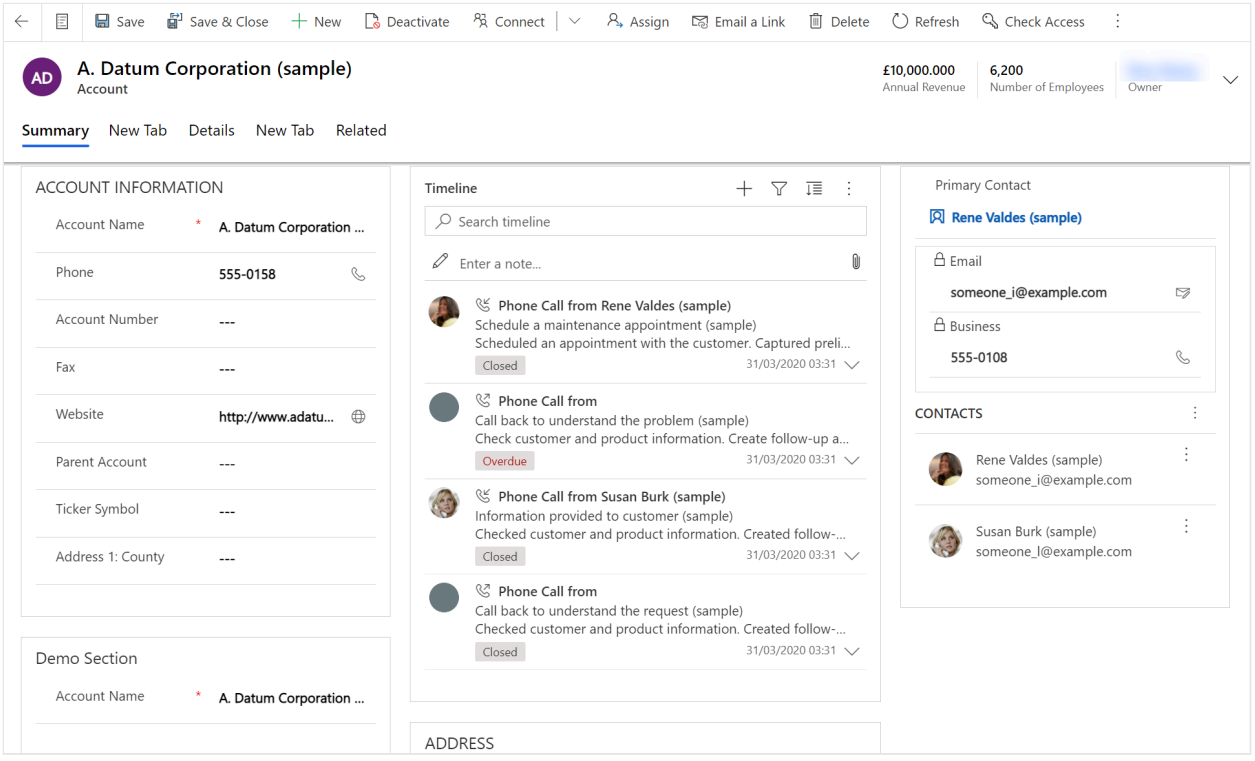
Article • 03/04/2025

Forms in model-driven apps provide the primary means by which users can update records. Forms associated with a table can be accessed via the table designer.

The most versatile form type of the new form is **Main**. However, in addition to main forms, app makers can build quick create, quick view, or card forms. Learn more about other types of forms

This article explains how to edit a main form, and add or change various elements of the form.

Here's an example of a main form within an app. It has several tabs, in addition to exposing the timeline component and a quick view form providing more details associated with the primary contact.



Open the form designer

To edit a form, such as to add or change elements, use the form designer. The form designer lets a maker edit forms for all model-driven apps.

Follow the procedures given below to access the form designer.

> **Note**
>
> If you create any new solution components in the process of editing the form, the names of the components use the solution publisher customization prefix for the default solution and these components are only be included in the **default solution**. If you want any new solution components to be included in a **specific unmanaged solution**, open the form designer through that unmanaged solution. This helps with the application management lifecycle.

Access the form designer for an unmanaged solution

1. Sign in to Power Apps.
2. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the unmanaged solution you want to work with. The solution type, managed or unmanaged, is displayed in the **Managed externally?** column.
4. In the list of components, locate the table with the form you want to edit. If the table isn't there, you need to add it.

Add a table to an unmanaged solution

1. With the unmanaged solution opened, on the command bar select **Add Existing** and then select **Table**.
2. All the available tables are listed. Select the table, and then select **Next**.
3. Select the option that's most appropriate, and then select **Add**:
   - For a custom table, select **Include all components**.
   - For a standard table choose **Select components**, and then browse to and select the main form you want to customize, and then select **Add**.
4. The table is added to the solution. In the list of components, open the table, select the **Forms** area, and then open form of type **Main**.
5. In the form designer, make the changes you want.
6. **Save** the form.

Publish the changes for use in the app

Certain customizations that make changes to the user interface require that they be published before people can use them in the application. To publish your customization, on the solution **Objects** area command bar, select **Publish all customizations**.

Access the form designer through the default solution

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane.
3. Select the table that you want, and then select the **Forms** area.
4. In the list of forms, open the form of type **Main**.

> **Note**
>
> If you have made other changes to the app, publish them using the app level publish option. Go to Validate and publish an app using the app designer for more information.

Form designer user interface

To understand in detail the form designer user interface, go to Overview of the model-driven form designer.

Form properties

To learn more about the form properties, go to Form properties.

Visibility options

Several types of form elements have the option to be shown or hidden by default. Tabs, sections, and columns all provide this option. When you use form scripts or business rules, the visibility of these elements can be controlled to create a dynamic form to provide a user interface that adapts to conditions in the form.

> **Note**
>
> Hiding form elements isn't a recommended way to enforce security. There are several ways people can view all the elements and data in the form when elements are hidden. To learn more, go to Show or hide form elements.

Tab properties

In the body of a form, tabs provide horizontal separation. Tabs have a label that can be displayed. If the label is displayed, tabs can be expanded or collapsed to show or hide their content by choosing the label. To know in detail about the tab properties, go to Configure tabs on a form.

Section properties

A section in a form occupies the space available in a tab column. Sections have a label that can be displayed and a line might be shown below the label. To know in detail about the section properties, go to Configure sections on a form.

Timeline

The timeline shows related activities for a specific table.

The types of activities supported are task, appointment, phone call, email, social activity, and custom activity.

The timeline also shows notes and, system, and user posts. Timelines show those activities that have the **Regarding** column set to the table you're viewing. For notes, the **Regarding** column isn't shown to the user; It's implicit when created from the timeline.

Each activity that's shown in the timeline, has the same quick actions that are available on the activity's command bar.

Common column properties

To know in detail about the common column properties, go to Configure column properties on a form.

Special column properties

All columns have the properties listed in Common column properties - legacy, but certain columns have additional properties. To know more, go to Special Column Properties - legacy.

Subgrid properties

You can configure a subgrid on a form to display a list of rows or a chart. To know in detail about the subgrid properties, go to Add and configure a subgrid component on a form.

Quick view control properties

A quick view control on a form displays data from a row that is selected in a lookup on the form. To explore the quick view control properties, go to Add and configure a quick view component on a form.

Web resource properties

You can add or edit web resources on a form to make it more appealing or useful to app users. Form enabled web resources are images or HTML files. These items are under found on the **Components** tab under **Display** in the form designer.

External website

You can add website content as an inline frame (iframe) to a form to integrate content from another website within the form. More information: Add an iframe to a model-driven app main form

Edit navigation

Navigation within the form allows people to view lists of related rows. Each table relationship has properties to control whether it should be shown. More information: Table relationships advanced options

Any table relationships that are configured to display can be overridden within the form editor.

For step-by-step instructions, go to Add form navigation for related tables.

To enable editing navigation, you must first select **Navigation** from the **Select** group on the **Home** tab.

In the Relationship Explorer, you can filter by 1:N (one-to-many) or N:N (many-to-many) relationships, or view all available relationships. The **Only show unused relationships checkbox** is disabled and selected. So you can only add each relationship one time.

To add a relationship from the **Relationship Explorer** just double-click it and it will be added below the currently selected relationship in the navigation area. Double-click a relationship in the navigation area and you can change the label on the **Display** tab. On the **Name** tab, you can see information about the relationship. Use the **Edit** button to open the definition of the table.

There are five groups in the navigation area. You can drag them to reposition them and double-click them to change the label, but you can't remove them. These groups are displayed only when there's something in them. If you don't want a group to appear, just don't add anything to it.

Configure event handlers

An event handler consists of a reference to a JavaScript web resource and a function defined within that web resource that will execute when the event occurs. To know more about configuring event handlers, go to Configure event handlers.

Next steps

Create and design forms

Create and edit quick create forms

Create and edit quick view forms

Feedback

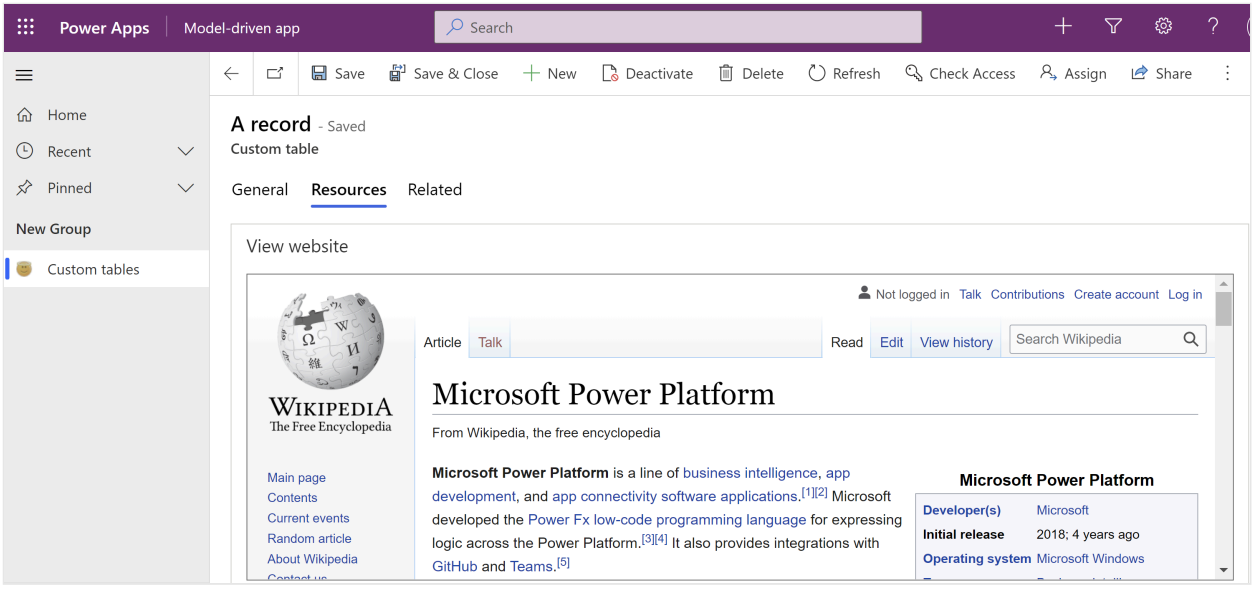
Was this page helpful? Yes No

Provide product feedback

Add an iframe to a model-driven app main form

Article • 03/04/2025

With the external website component, you can add an inline frame (iframe) to a form to integrate content from another website within the form.



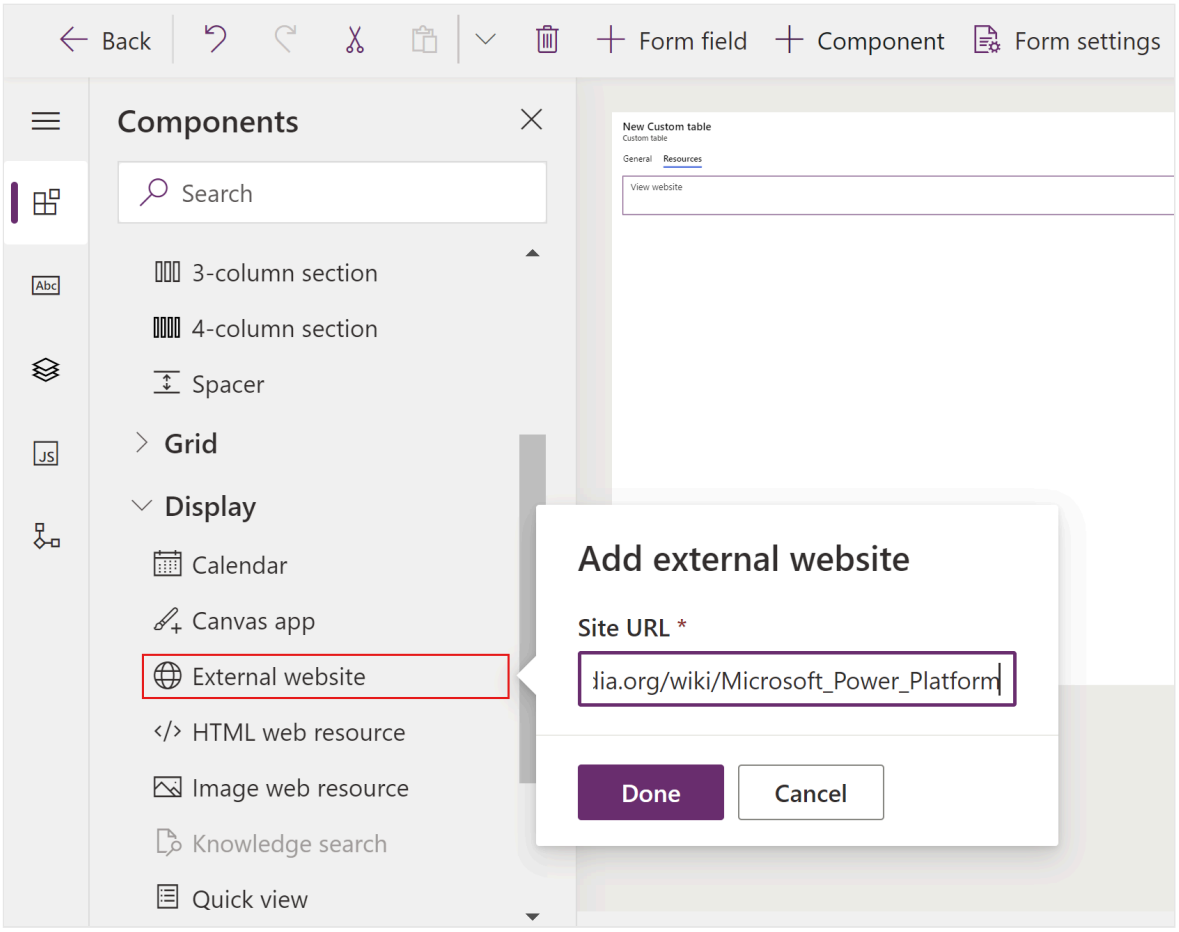
1. Select **Solutions** on the left navigation pane, open the solution you want, and then open the table that you want to edit. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Select **Forms** from the **Data experiences** area.

3. In the list of forms, open a form of type **Main**.

4. In the form designer, select the section of the canvas where you want to add the iframe.

5. On the **Components** left pane, expand **Display**, select **External website**, enter the **Site URL**, and then select **Done**.



Expand table

Tab	Property	Description
Display options	Label	**Required**: A label to display for the iframe.
Display options	Name	**Required**: A unique name for the iframe. The name can contain only alphanumeric characters and underscores.
Display options	Hide label	Select if you want the label hidden.
Display options	Hide	You can hide the iframe so that it can be made visible by using scripts. More information: Visibility options
Display options	URL	**Required**: The URL for the page to display in the iframe.
Formatting	Column width	When the section containing the iframe has more than one column, you can set the column to occupy up to the number of columns that the section has.

Tab	Property	Description
Formatting	**Component height**	You can control the height of the iframe by specifying a number of rows the control occupies.
Formatting	**Use all available vertical space**	Instead of setting the height by a number of rows, you can allow the iframe height to expand to available space.
Formatting	**Scrolling**	You have three options for scrolling behavior: - **As Necessary**: Display scrollbars when the size of the iframe is larger than the available space. - **Always**: Always display scrollbars. - **Never**: Never display scrollbars.
Formatting	**Display border**	Display a border around the iframe.
Dependencies	**Table column dependencies**	An iframe might interact with columns in the form using script. If a column is removed from the form the script in the iframe might break. Add any columns referenced by scripts in the iframes to the **Table column dependencies** so that they can't be removed accidentally.
Advanced	**Restrict cross-frame scripting, where supported**	It's considered a security risk allowing pages from a different web site to interact with the app using scripts. Use this option to restrict cross frame scripting for pages you don't have control over.
Advanced	**Pass row object-type code and unique identifiers as parameters**	Data about the organization, user, and the record can be passed to the iframe. More information: Pass parameters to iframes

6. **Save** and **Publish** the form.

> **Note**
>
> - If the web page doesn't appear in the iframe, it might be because the website doesn't allow iframe rendering. When this occurs, the message *URL* **refused to connect** is displayed in the iframe at runtime.
> - Forms aren't designed to be displayed within an iframe.
> - Authentication inside an iframe, either through a redirection or popups, isn't supported on mobile.

Pass parameters to iframes

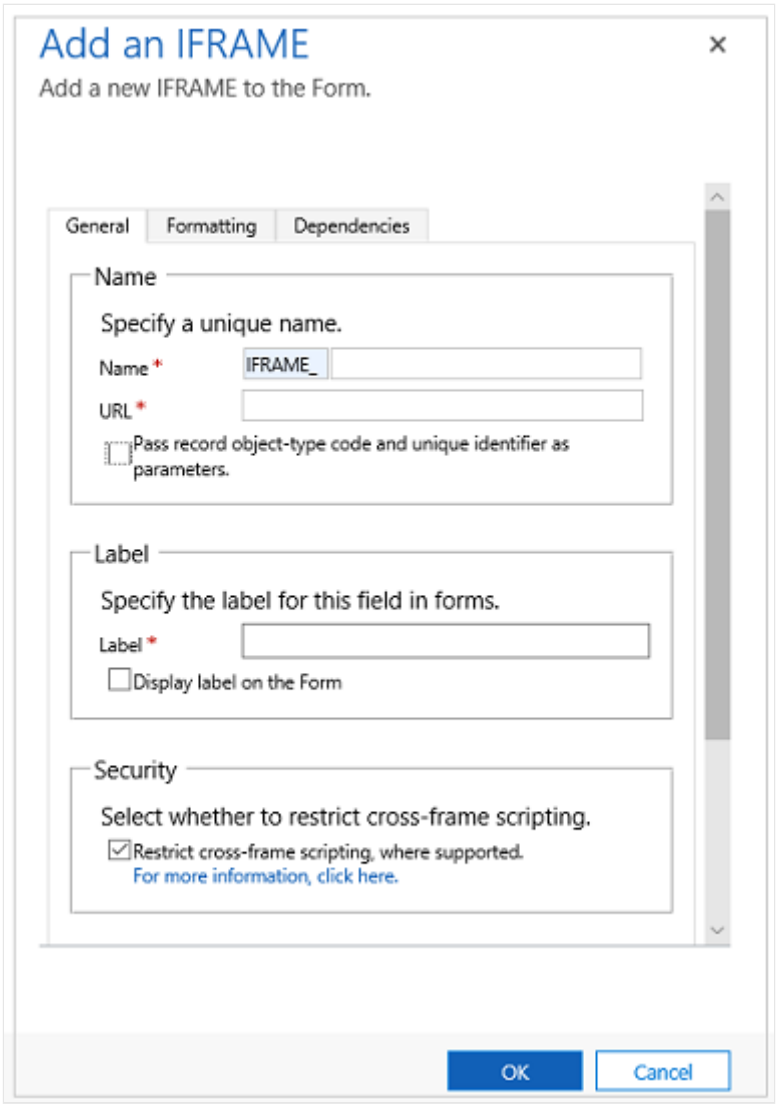
Information about the row can be passed by enabling the **Pass row object-type code and unique identifiers as parameters** option. The values passed are:

Expand table

Parameter	Description
`orglcid`	The organization default language LCID.
`orgname`	The name of the organization.
`userlcid`	The user's preferred language LCID
`type`	The table type code. This value can be different for custom tables in different organizations. Use `typename` instead.
`typename`	The table type name.
`id`	The ID value of the row. this parameter has no value until the table row is saved.

Add an iframe using the classic form designer

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane, and then open the table that you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the **Forms** area.
4. In the list of forms, open a form of type **Main**.
5. Select **Switch to classic** to edit the form in the classic form designer.
6. On the **Insert** tab, select **IFRAME** to view iframe properties.



Expand table

Tab	Property	Description
General	**Name**	**Required**: A unique name for the iframe. The name can contain only alphanumeric characters and underscores.
	URL	**Required**: The URL for the page to display in the iframe.
	Pass row object-type code and unique identifiers as parameters	Data about the organization, user, and the row can be passed to the iframe. More information: Pass parameters to iframes
	Label	**Required**: A label to display for the iframe.
	Display label on the Form	Whether the label should be displayed.
	Restrict cross-frame scripting, where supported	It's considered a security risk allowing pages from a different web site to interact with the Dynamics 365 application using scripts. Use this option to restrict cross frame scripting for pages you don't have control

Tab	Property	Description
		over.
	Visible by default	Showing the iframe is optional and can be controlled using scripts. More information: Visibility options
	Enable for mobile	Select the checkbox to enable the iframe for mobile.
Formatting	**Select the number of columns the control occupies**	When the section containing the iframe has more than one column, you can set the column to occupy up to the number of columns that the section has.
	Select the number of rows the control occupies	You can control the height of the iframe by specifying a number of rows the control occupies.
	Automatically expand to use available space	Instead of setting the height by a number of rows, you can allow the iframe height to expand to available space.
	Select the scrolling type for the iFrame	You have three options: - **As Necessary**: Show scrollbars when the size of the iframe is larger than the available space. - **Always**: Always show scrollbars. - **Never**: Never show scrollbars.
	Display border	Display a border around the iframe.
Dependencies	**Dependent columns**	An iframe might interact with columns in the form using script. If a column is removed from the form the script in the iframe might break. Add any columns referenced by scripts in the iframes to the **Dependent columns** so that they can't be removed accidentally.

Next steps

Use the Main form and its components

Feedback

Was this page helpful? Yes No

Provide product feedback

Design productive main forms in model-driven apps

Article • 04/08/2022

Building experiences where tasks can be completed quickly and effectively is crucial to user satisfaction.

Model-driven apps provide many options to build good user experiences, including:

- Application-level options.
- Site map organization options.
- Many form-creation options.

This article shows how to design highly efficient and productive forms in a model-driven app, including how to work with:

- Main forms.
- Main form dialogs.
- Form component controls.
- Quick create forms.
- Quick view forms.

Form layout and design are important to building better forms. However, it's also important to build forms that load quickly and allow for fast navigation within the form and across tabs.

We also recommend an understanding of best practices for optimizing form performance and how to make forms load fast in your model-driven app. More information: Design forms for performance in model-driven apps

Working with main forms

Main forms are used in a model-driven app when working with and editing a specific record's data for a table to ensure that users interacting with that record stay within the constraints of the table structure.

Building around table data

A form allows an app maker to build around the table data, and often requires that you work with and interact with related records from other tables. One important consideration is how a model-driven form is bound to data. The out-of-the-box controls

for a form are bound to fields from the table. These controls allow rapid development of a form for creating and editing data, but without more customization they aren't as flexible when using a form for multiple tasks that need user input, or where additional action is required before saving information.

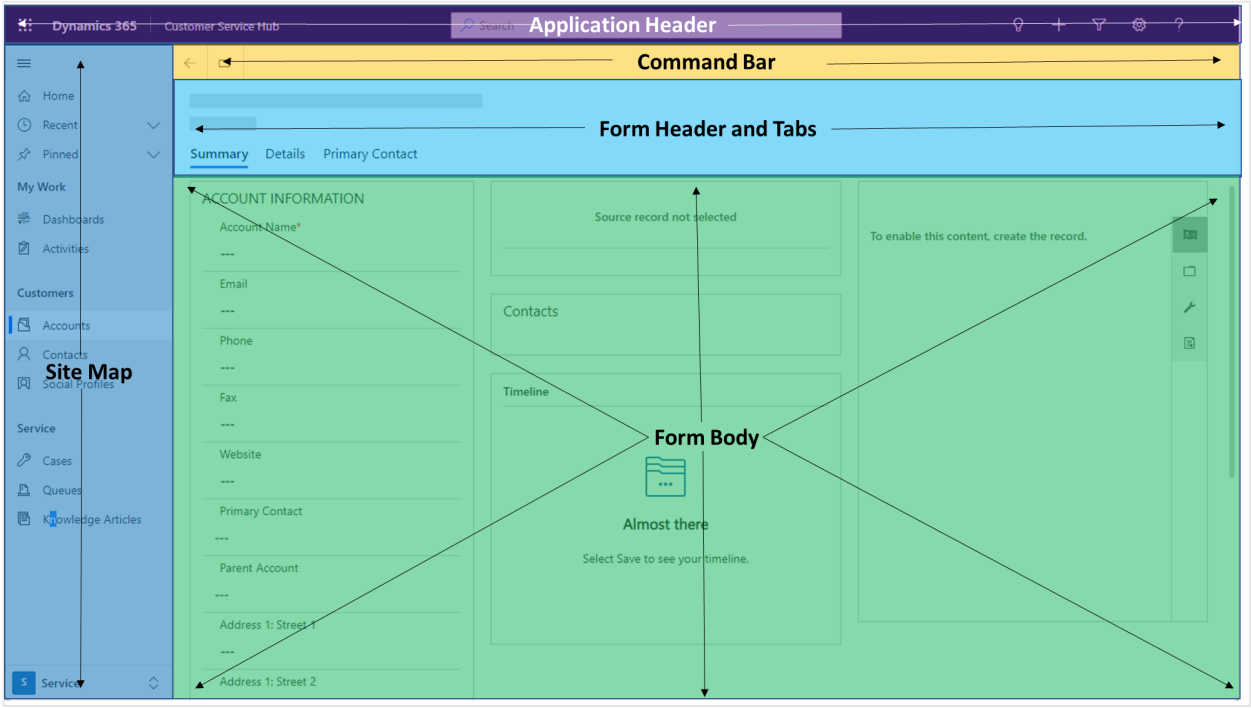
Use of related data

One of the strengths of model-driven forms is the ease of use with standard (out-of-the-box) controls like lookups, subgrids, and reference panels to quickly view or edit **related** records. Forms also support custom Power Apps component framework controls to extend functionality beyond the standard controls. Forms primarily support child or secondary data relationships for a record; they don't easily support tertiary or higher-order relationships. For example, if there is an account record that has a one-to-one or one-to-many relationship with contacts, interactions with both the parent and child data can be quickly added with a form by using lookups that interact with other forms, including a main form dialog or embedded forms like a quick view or form component control. Each of these form types will be covered in more detail later in this article.

Working with related data requires configuration, including using a main form dialog to work with related records of the main form dialog or using a custom canvas page. Multiple-entity forms can be created by using main form dialogs or form component controls. However, a main form doesn't directly support tertiary-related records on the main form.

Understanding the main form layout

The following image shows the layout of a Unified Interface application.



Visual elements that are not a part of the form

- The application header is part of the overall page structure of a model-driven app. The app header contains the application name and app-level actions, which include search, help, assistance, quick create forms, settings (personal and app level for admins), and so on.
- The application site map contains navigation across the application to pages that are included in your application. The navigation experience provided by the site map offers a quick way to access other areas of your application.
- The form command bar is also not part of the main form. It can be configured and set up by using the Ribbon Workbench community plug-in for XrmToolBox. Navigation and interaction on the form can be customized. Several out-of-the-box options include saving, creating a new record, sharing, merging, editing multiple records, and so on.

> **Note**
>
> Resources created by the community aren't supported by Microsoft. If you have questions or issues with community resources, contact the publisher of the resource.

Visual elements that are a part of the form

- The form header is part of the form. It includes the table and record name, the ability to switch to additional forms for the table, four read-only fields of the

record, and the tabs included on the form. Forms only appear in the form selector if they've been enabled and added to the app. More information: Create and design model-driven app forms.

- Form headers also have a mode where more than four fields can be added and provide a simple navigation to edit the fields in the header and any additional field you want a user to easily access from the form header. Additional tabs should be used to access information that isn't primary to the task at hand, or to focus the user on data or information that's specific to a given task but isn't primary to the job. Information that's necessary and is primary to working with data should be on the first tab and should not be hidden.
- The form body is the area on the form that should be used to interact with data of the record and any related records that are essential to completing a task. We recommend that you limit the data to the top tasks that need to be done on the first tab and move secondary tasks to additional tabs. building forms based on specific business needs and roles should also be considered when designing your application.

If some data is rarely used you should consider building another form that can be accessed when a user needs to reference that data or information.

> ⓘ **Important**
>
> **Form footers are deprecated** with the 2021 Release Wave 2 (August 2021 Preview / October 2021 GA). Remove any controls you have added out of the footer.

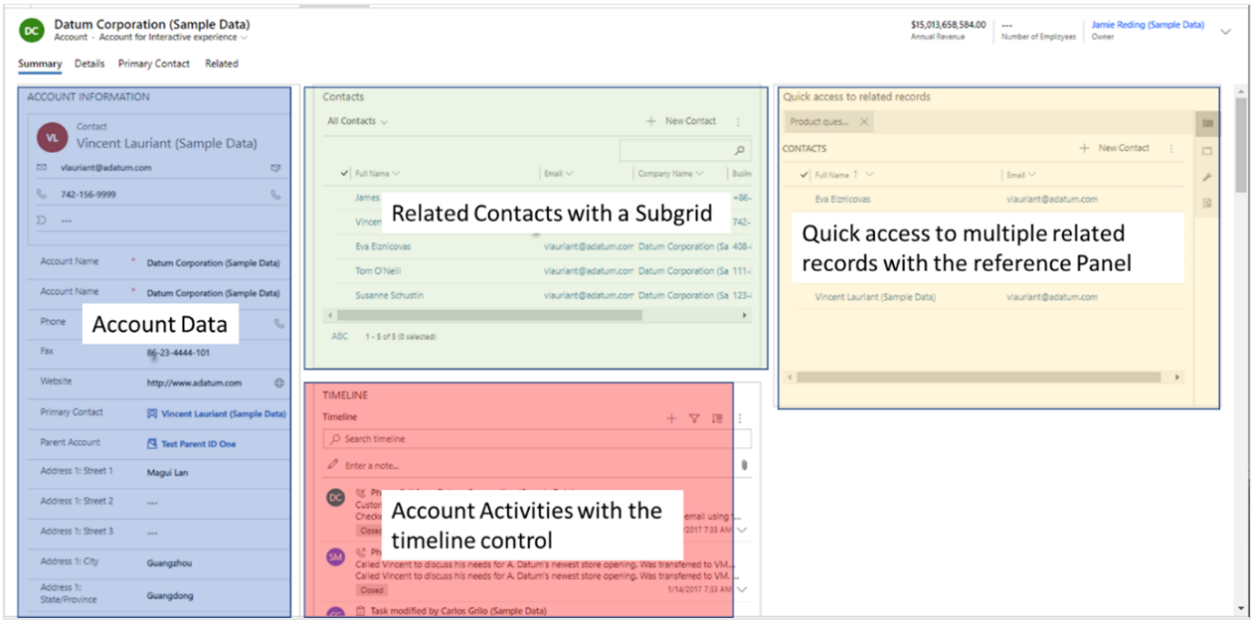
Example scenario of productive form design

The following scenario demonstrates the best use of a model-driven main form over data that uses the available controls.

Imagine a scenario where a salesperson needs to maintain data for an account that also has the following form components:

- A primary contact (lookup).
- Additional contacts (subgrid).
- A reference panel where a user can select to see lists of related records with access to read-only views (quick view) in the reference panel. The reference panel includes a timeline control that's used to quickly access and create activities for that record.

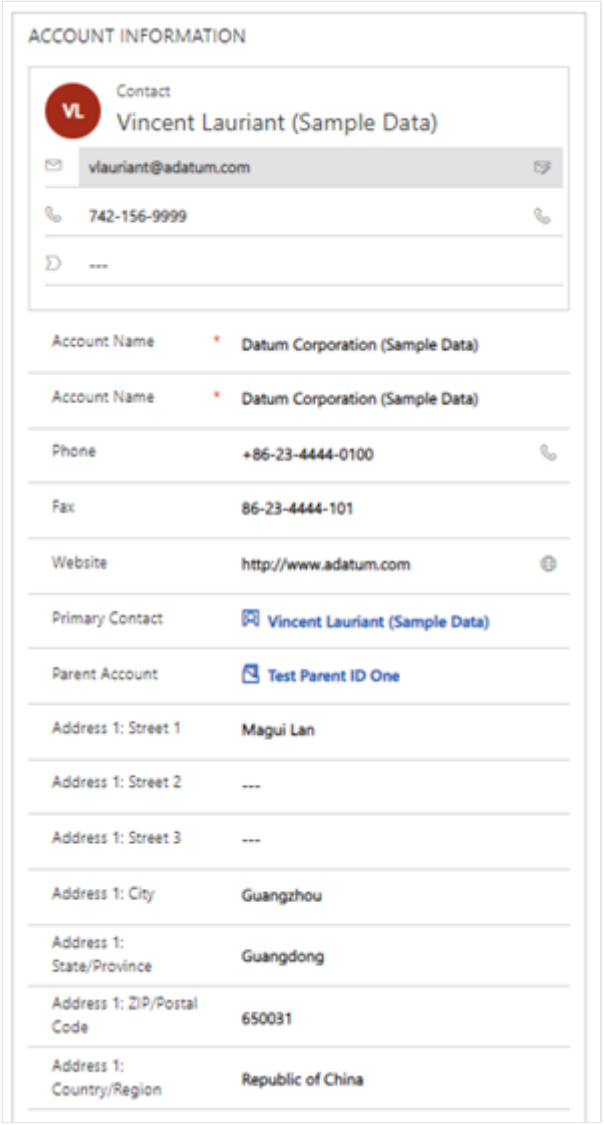
This form is in regular use, so it is worth spending time on the layout to ensure it is simple and efficient to operate.



Account data

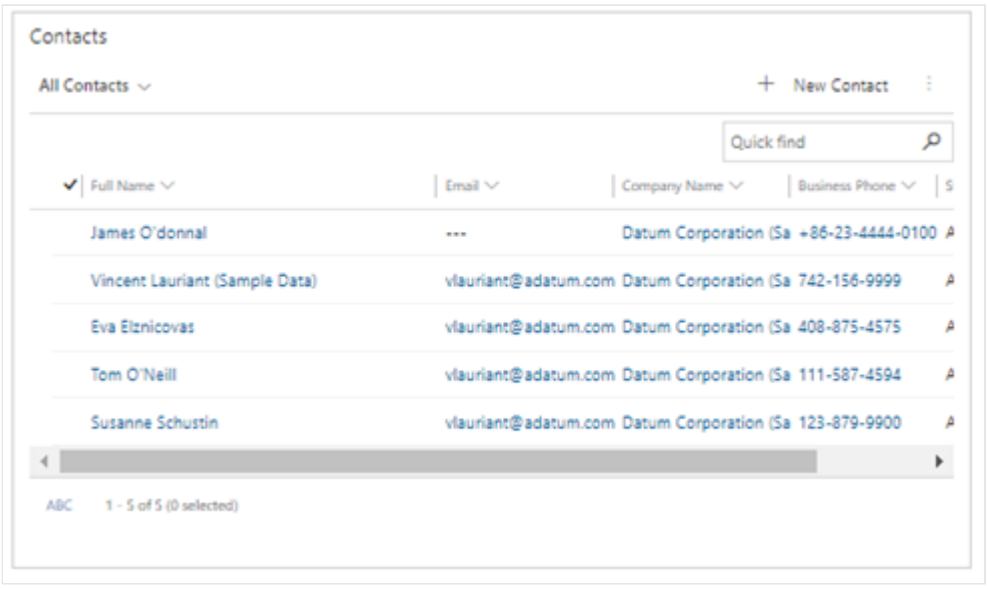
The first section on the form contains the fields that are used to manage the data and information for that record. Here, a user can quickly review and edit information they use throughout their workday. Points to note:

- Required fields and most-used fields are at the top of this section.
- A quick view form of the primary contact is added at the top so users can quickly view and use the information to contact the individual who's important for their daily tasks.



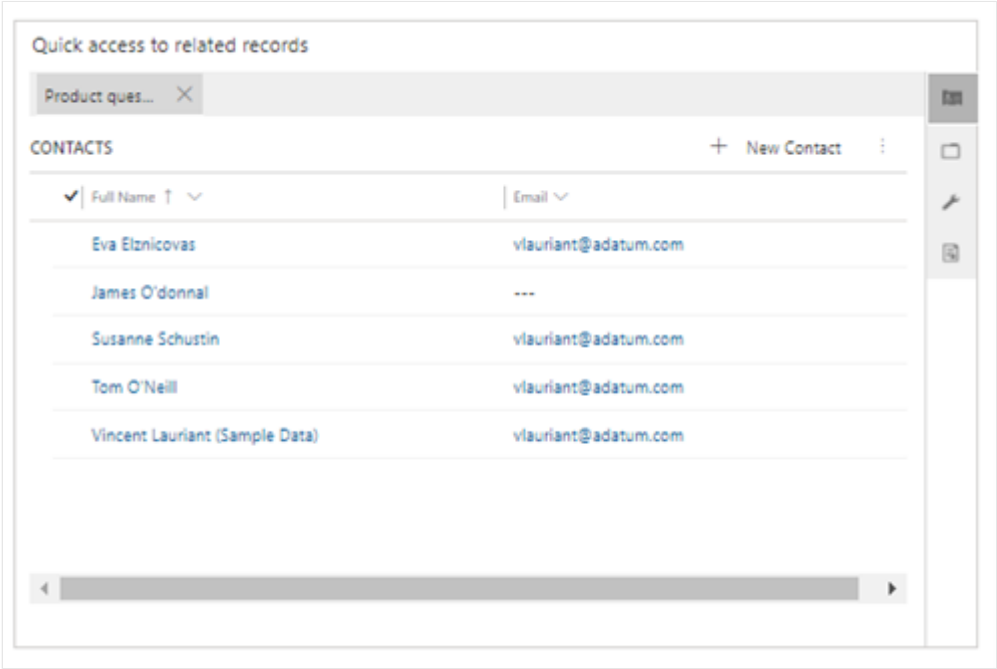
Related contacts with a subgrid

To the right of the account information section is the contact table subgrid. It shows all the contact records that are related to the main record. This allows the user speedy interaction with the contact record related to the account.

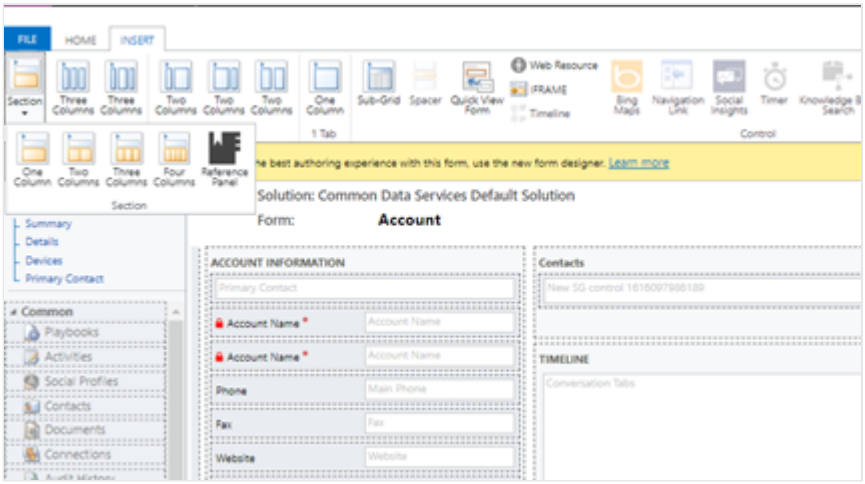


Quick access to multiple related records with the reference panel

Next to the contact subgrid is the reference panel. This control can give even further access to related records that go beyond just the contacts. A subgrid helps drive data density and efficiencies when working across all related records of the main form. In this example, the reference panel includes contacts, opportunities, leads, and products. One limitation of the reference panel is that it only provides read-only views of records. It doesn't support a main form of a related record like a main form dialog or a form component does. Reference panels are best used when you want to quickly view information of a related record.

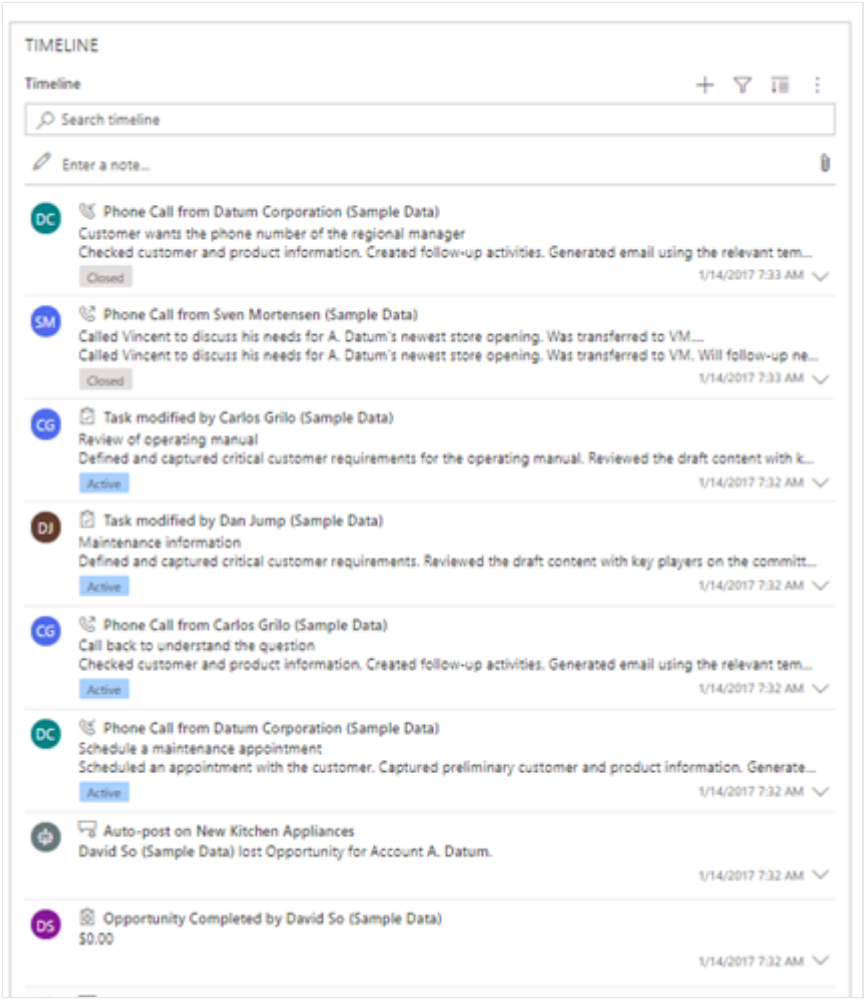


Currently, you add a reference panel control by using the classic form designer. On the **Insert** tab, expand **Sections**, and then select **Reference panel**.



Account activities with the timeline control

Below the contacts subgrid is the activities timeline wall. The timeline control allows a user to quickly view recent information about activities that have an activity record associated to the account. Available activities found in a timeline are emails, phone calls, notes, appointments, and tasks. Users can also send new emails, log new phone calls, and add notes quickly and easily from the control. More information: Set up the timeline control



Working with data on a main form best practices

- If additional requirements are added to data on the form that isn't configured on the table, it's important to make sure that the data that's required is always visible and editable—for example, a required field for a user that's based on a business rule. If a field is required and made read-only, it will block users from completing tasks and can create frustration and dissatisfaction. Although hidden and read-only components are supported by the Power Apps platform, it's important to know that if a user saves data on a form that includes a required field that isn't editable, the save will be processed. This ensures that a user is never blocked from completing a task based on a bad design pattern.

- Hidden fields can be used like read-only fields and are handy when building data-dense forms. However, the same caution applies as when working with required fields: A required field may not be hidden if it can be null when editing a record. Business rules or events can be used to hide or show fields on a form; however, the form will behave the same as a read-only field when it's being saved. Model-driven forms will always allow for the save if a field is required but hidden on the form.

- If data needs to always be available, regardless of the tab a user is on, it's vital to add that data to the header. It's important to know that the high-density headers in Unified Interface will only display up to four read-only fields. However, you can use the high-density headers with a flyout to add additional fields that also allow users to edit fields. This streamlines the experience to quickly access and edit information that you always want to make available without multiple clicks or additional navigation.

Building multiple-entity forms

The previous section provided good examples of how to work with data that's directly related to a record. However, what can drive even further efficiencies is working with related data directly from the main form, without navigating away. This keeps the user in context, and reduces friction and improves satisfaction by streamlining experiences using a main form dialog or a form component control.

The following sections walk through several scenarios that show how to build highly performant multiple-entity forms. The goal is to streamline user experiences that reduce clicks and unnecessary navigation, and speed up daily tasks that often require a user to interact with related records on a main form. Form components used for efficiency described here include:

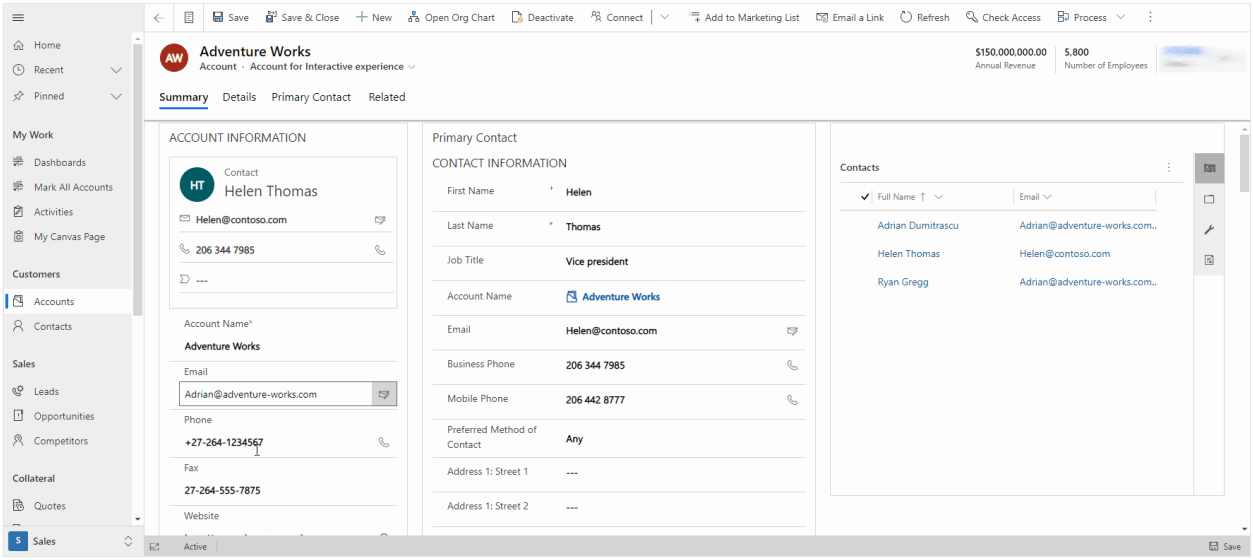
- Main form dialog.
- Main form component control.
- Quick create form.
- Quick view form.

Using a main form dialog

Using a main form dialog is powerful when you need to have users access, and—more importantly—interact with, all the information of a related record. This includes when users need to work with a business process flow, access all the tabs, or access all the related records on the form. This is important when there are business processes that drive requirements on a related record that need to be followed to ensure data integrity.

Users need access to timeline activities or related records of the child record without additional navigation. Note that a main form dialog can be configured directly from the lookup control and can be used on the command bars or with JavaScript by using the `navigateTo` client API. The dialog can be positioned on the left, middle, or right side of the page as a modal dialog overlay. More information: Open main form in a dialog using client API

In this example, a user needs to access the primary contact record, update the phone number, add a note in the activities timeline, and create a new appointment from the contact form. To streamline this experience by reducing clicks and unnecessary navigation, the lookup control is configured to open the contact main form as a dialog. This example shows how the main form dialog can use quick create forms from the related record, incorporate duplicate records, and allow the user to fully interact with data on the related record.

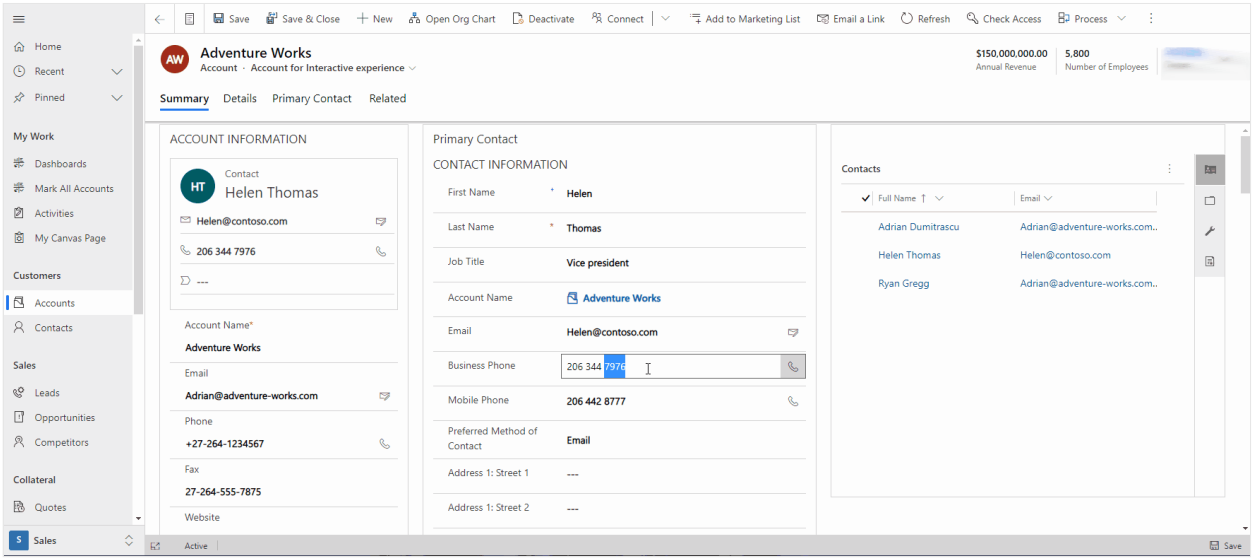


Using the form component control

Often you only need to edit information on a related record that doesn't require the user to interact with the entire form and doesn't require an additional business process flow, but you don't want to give users the ability to navigate to the related form. To accomplish this scenario and streamline managing data inline on the main form, you can use a form component control. The form component control provides the user with access to the related record, including all the sections, controls, and fields on the related record form. Form components can also be used for specific actions with forms designed for only that task. An example of this is the Microsoft Dynamics Field Service Mobile experience. Form components are used to manage work orders and tasks related to those work orders in a streamlined and more efficient process.

Note that a form component control will use any form that you've created for that record and it will honor any event, including loading a form, saving data, or on change information. It will also honor any business rules and actions on the form, including opening a main form dialog from a lookup (if you've configured the form to work in that manner). It will also include any duplicate detection rules and parent/child relationships when working with data in the related record form component. The form control component promotes unsaved changes to the main form, and notifies users when a field is required and if there are data entry errors on the related form component. More information: Edit related table records directly from another table's main form

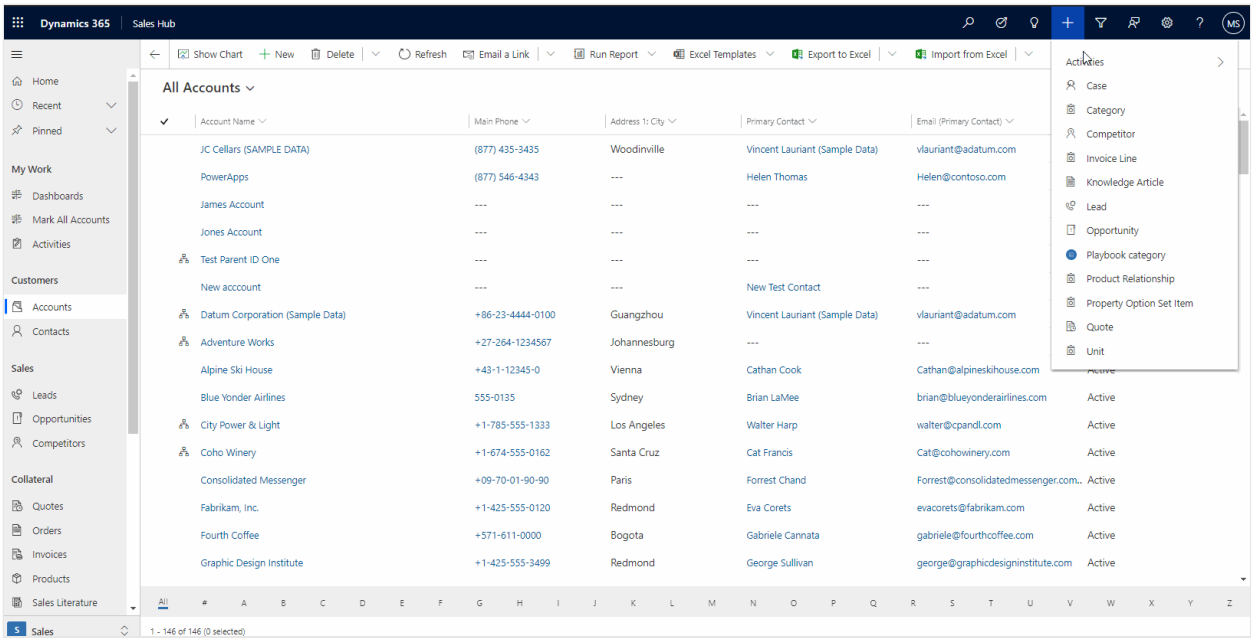
In this example, a user is editing the primary contact information directly on the account main form and scrolls down the form to create a new appointment for the primary contact without leaving the account form.



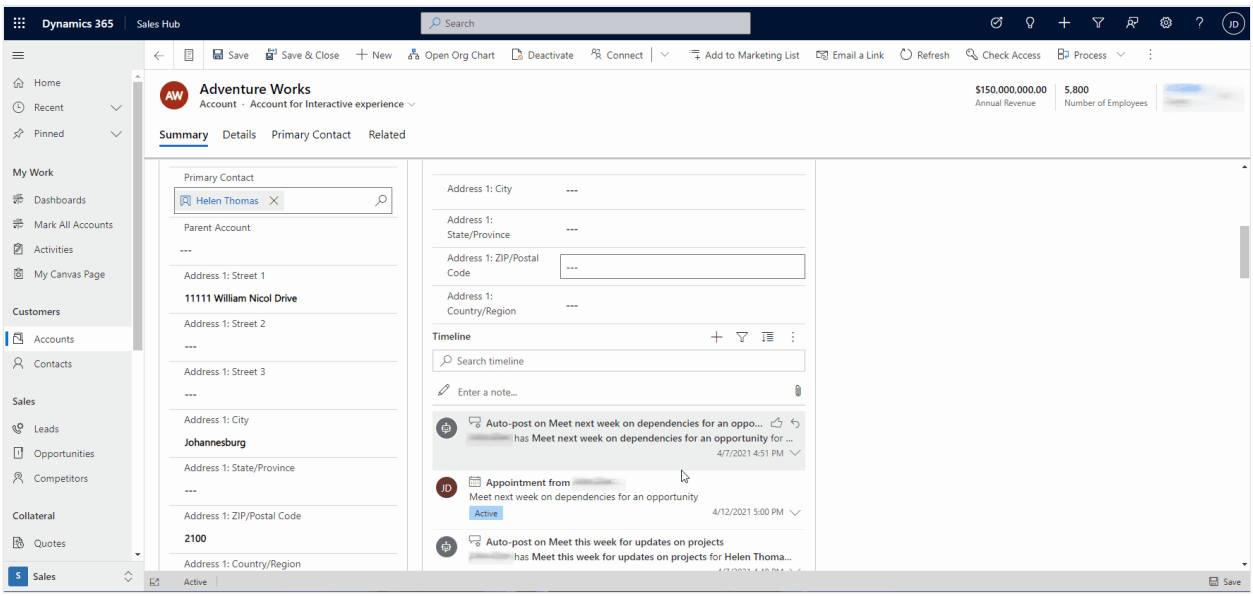
Using quick create forms

There will be times when you want to create an experience where users can quickly create records without additional navigation or clicks. A quick create form is a good solution for this. It can be used in multiple scenarios, which include creating appointments from an activity timeline wall, creating a contact from a lookup if a main form dialog isn't needed, or quickly creating a record—like an account or contact that doesn't require additional relationships for data integrity—from a view page (app menu + option). More information: Create or edit model-driven app quick create forms for a streamlined data entry experience

In the following example, a user needs to quickly create a lead by using the + menu option in the app header.

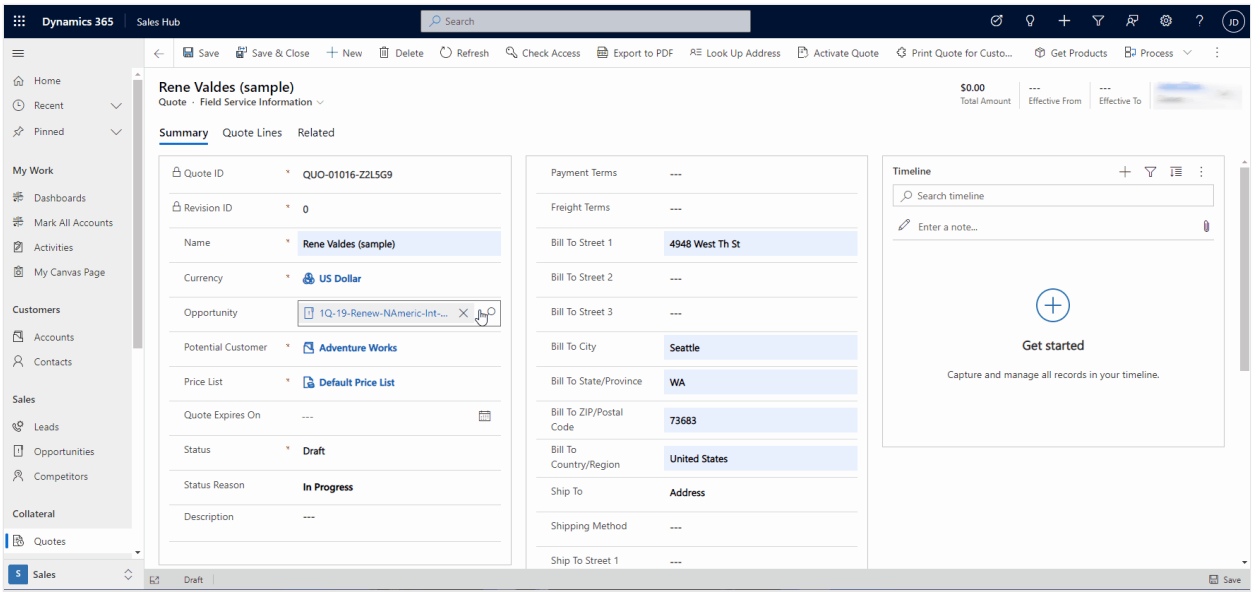


In this example, the quick create form is used to create an appointment for a contact from the timeline wall.



In this example, a user can quickly create an opportunity from a quote record form by using a lookup control.

> **Note**
>
> Opportunity and quote tables require a Dynamics 365 customer engagement app, such as Dynamics 365 Sales.

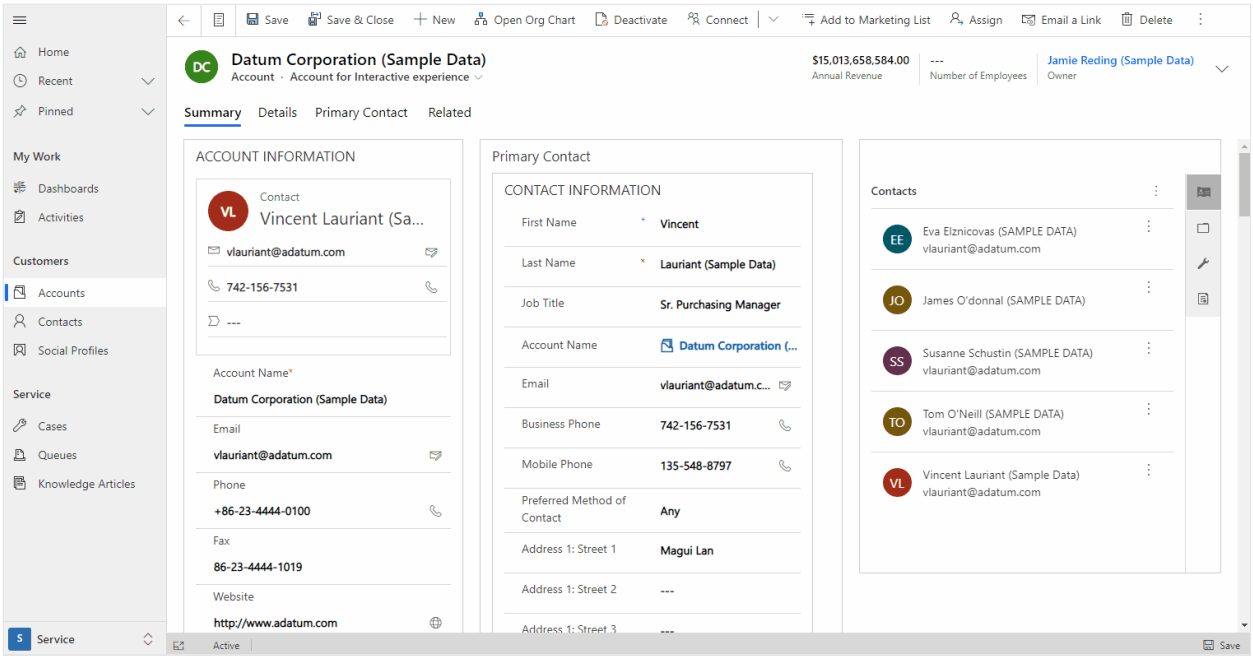


Using quick view forms to see data from related tables

In many scenarios, a user just needs to quickly view information from a related table without having to interact with it. To facilitate these scenarios, you can use quick view forms. An example of where a quick view form is used is in the reference panel of a main record. This demonstrates the power of accessing multiple related records without having to navigate away.

> **Note**
>
> While it is called a quick view form, it is in principle, a read only view of data. Users can't use this control to edit data.

More information: Create a model-driven app quick view form to view information about a related table



Using form display options to simplify forms

There are many times when you need to build an experience that focuses the user on the task at hand without additional distraction that might cause inefficiency. This is especially important when you're working with tables and records that have global components, which are created and automatically added to forms regardless of the task you're trying to build. Although this provides a shareable and easy-to-build infrastructure that drives consistency across an app, it does at times provide unnecessary distraction.

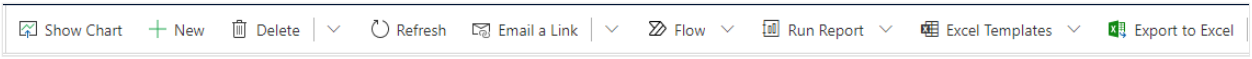
Form display options should be used when you want to lessen the number of actions, focus a user to complete a task, and reduce clutter by removing unnecessary components. Components that can be removed from a form include the command bar, the header body, and the form tabs.

All these types of options are only available via the client API and manual formxml updates. Manual formxml manipulation can cause issues importing solutions if done incorrectly. Remove global form components carefully when designing your form.

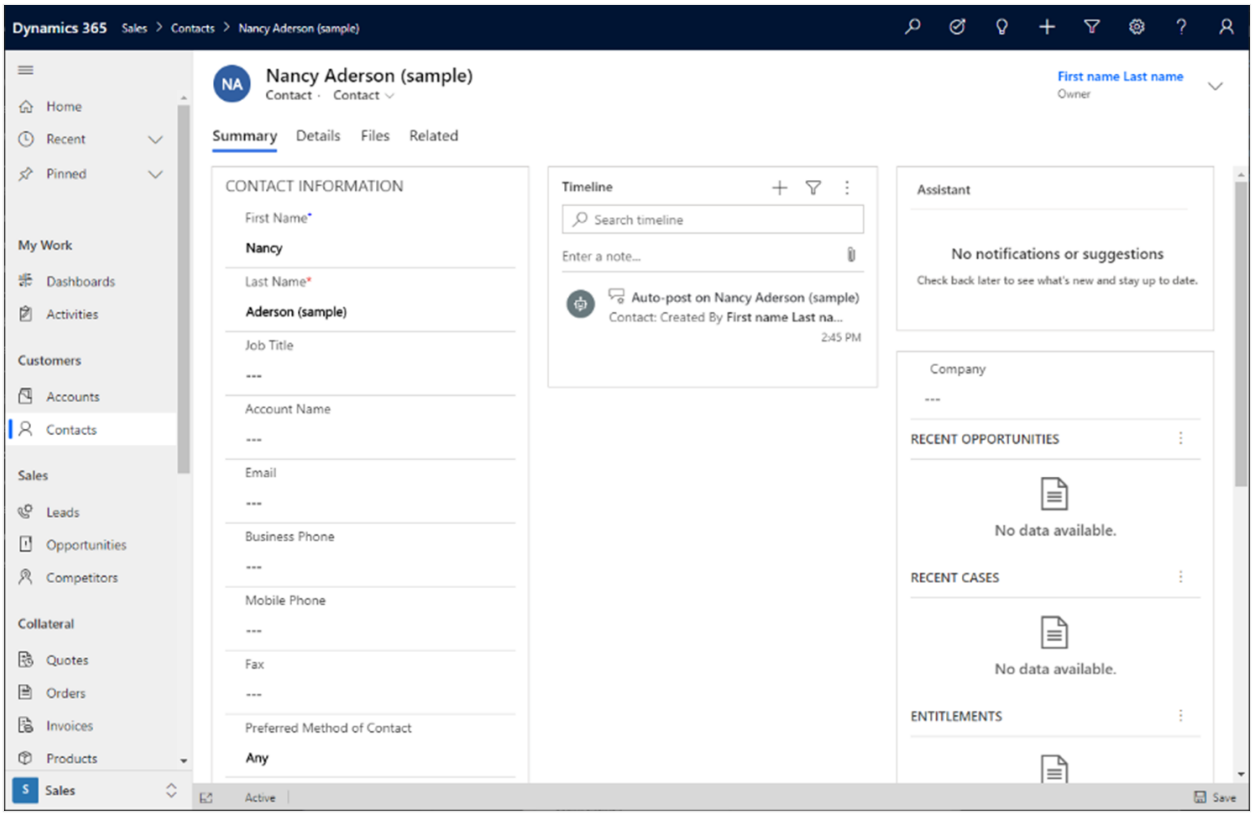
Navigation considerations

When certain components are hidden like the command bar or the tab list, this may remove navigation that might be necessary to complete a task. Be sure to account for those requirements in your form design by using additional controls created with the Power Apps component framework.

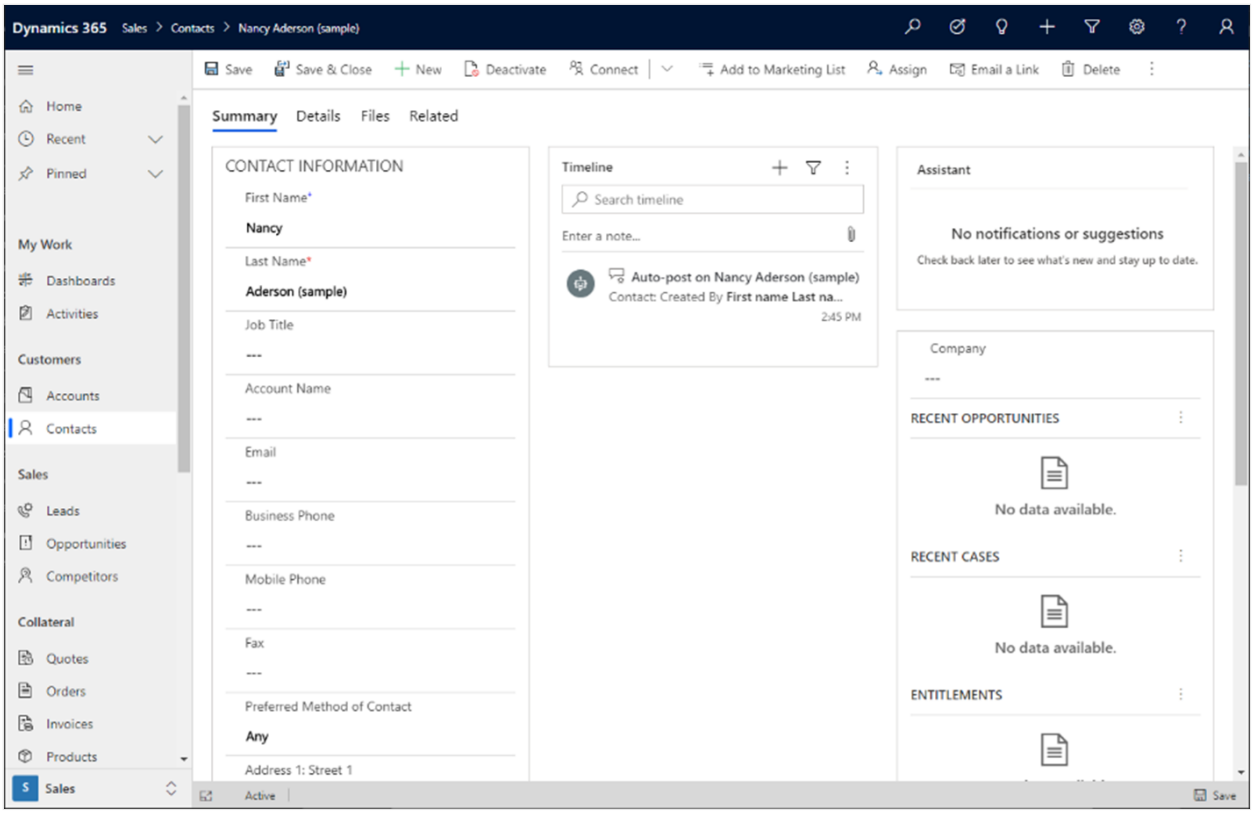
In this example, the **command bar** has been removed to reduce any unnecessary actions when working with an account. The command bar would ordinarily be shown as follows.



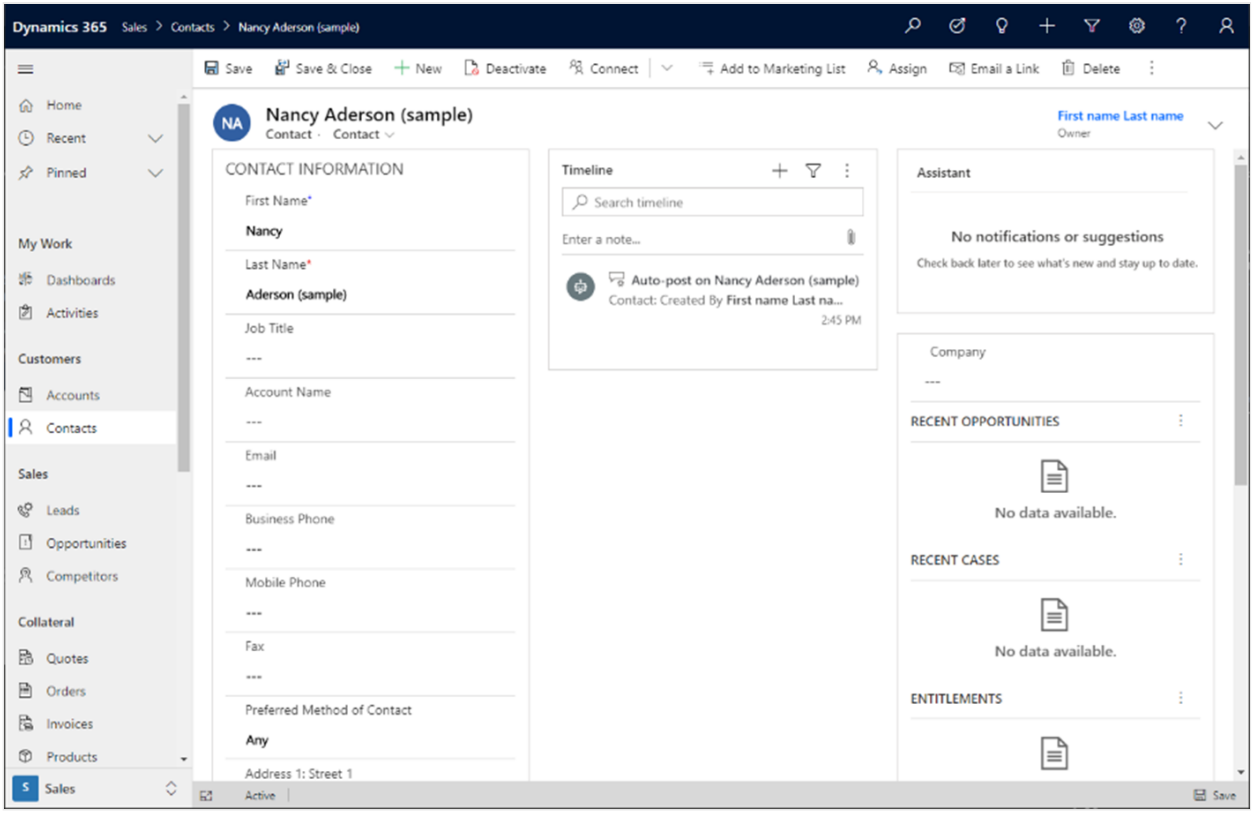
This form has been designed to not need the command bar, so the maker has removed it to keep the user on task. More information: setCommandBarVisible (Client API reference)



In this example, the header has been removed to promote a dense form that focuses the user on record details. This should only be used if the header doesn't provide additional value or displays information that doesn't always need to be available to a user. More information: setBodyVisible (Client API reference)

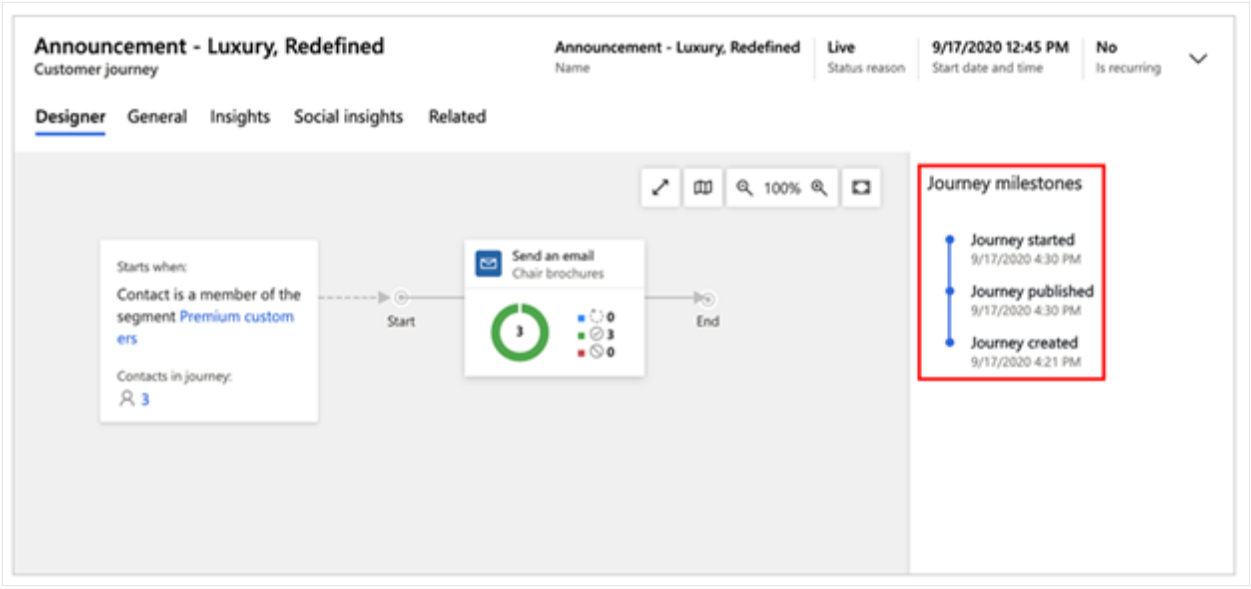


In this example, the tabs have been removed. This can be helpful when you build a form that only has a single tab or you want to focus the user on the first tab of a form without the distractions of additional tabs that can lead to loss in productivity for a specific task. More information: setTabNavigatorVisible (Client API reference)



Form display option to use the entire space of a tab

In many scenarios, a form might exist with a tab and a single control, and you want that control to take up the entire available space within the body of the form on that tab. This can be done by using the **form display option** that allows the first control on a page in a tab to expand the full height and width of the form. All existing reflow rendering across view port sizes from large to small is honored, and will behave like a regular form except that the control will use the entire space. In this example, the Dynamics 365 Marketing application supports a full tab layout for the customer journey experience. The command bar is removed to further reduce clutter and ensure that the user is focused on the task of managing customer journeys in the application. More information: setContentType (Client API reference)



Next steps

Design forms for performance

Design forms for performance in model-driven apps

Article • 03/04/2022

Building experiences where tasks can be completed quickly and efficiently is crucial to user satisfaction. Model-driven apps can be highly customized to create experiences that meet the needs of your users, but it is important to know how to effectively code, build, and run model-driven apps that load quickly when a user opens and navigates in your app while working on daily tasks. Performance has been shown to be a key driver of dissatisfaction of an app when it is not optimized for performance.

Intelligent customizations and performant forms are important aspects to building highly efficient and productive forms. It's also important to make sure you're building highly productive forms with the best practices in user interface design and layout. For information about designing forms for efficiency and productivity, see Design productive main forms in model-driven apps.

It is also important to ensure users are on recommended and supported devices and minimum required specifications. More information: Supported web browsers and mobile devices

Working with data and tabs

This section covers how controls that display data and tabs impact form performance.

Significance of the default tab

The default tab is the first expanded tab on a form. It plays a special role in the loading of a form page. By design, the controls of the default tab are always rendered when opening a record. Specifically, the control initialization logic, such as data retrieval, is invoked for every control on the tab.

In contrast, a secondary tab does not perform this initialization on its controls when the form is initially loaded. Instead, the control initialization occurs at the time the secondary tab is opened either through user interaction or calling the `setFocus` client API method. This provides an opportunity to shelter the initial form load from excessive control processing by placing certain controls in secondary tabs instead of the default tab. Thus, the control placement strategy can have a significant effect on the responsiveness of the initial form load. A more responsive default tab provides a better

overall experience for modifying important fields, interacting with the command bar, and exploring other tabs and sections.

Always put controls that are used the most at the top of your default tab. Layout and information architecture is not only important for performance but also to improve productivity when users interact with data on the form. More information: Design productive main forms in model-driven apps

Data-driven controls

Controls that require extra data beyond the primary record produce the most strain on form responsiveness and loading speed. These controls fetch data over the network and often involve a waiting period (seen as progress indicators) because it can take time to transmit the data.

Some of the data-driven controls include:

- Quick view form
- Subgrid
- Timeline
- Assistant (requires Dynamics 365 Sales Insights)

Keep only the most frequently used of these controls on the default tab. The remaining data-driven controls should be distributed into secondary tabs to allow the default tab to load quickly. Furthermore, this layout strategy reduces the chance of fetching data that ends up being unused.

There are other controls that are less impactful than the data-driven controls but can still participate in the above layout strategy in order to achieve the best performance. These controls include:

- Lookup
- iFrame
- Web resource

Web browser

This section covers good practices to use with web browsers.

Don't open new windows

The `openForm` client API method allows a parameter option to display a form in a new window. Don't use this parameter or set it to false. Setting it to false will ensure the `openForm` method performs the default behavior of displaying the form using the existing window. It is also possible to directly call the `window.open` JavaScript function from a custom script or another application; however, this should also be avoided. Opening a new window means that all of the page resources need to be fetched and loaded from scratch since the page is unable to leverage the in-memory data caching capabilities between a previously loaded form and the form in a new window. As an alternative to opening new windows, consider using the multisession experience that allows records to be opened in multiple tabs while still maximizing the performance benefits of client caching.

Use modern browsers

Using the most up-to-date web browser is key to ensuring your model-driven app runs as fast as possible. The reason for this is that many of the performance improvements can only be used in the newer modern browsers.

For example, if your organization has older versions of Firefox, non-Chromium-based browsers, and so on, many of the performance gains that are built into a model-driven app will not be available in the older browser versions because they don't support features that the app depends on to run quickly and smoothly.

In most cases, you can expect to see page load improvements by just switching to Microsoft Edge, updating to the latest current browser version from an older version, or moving to a modern Chromium-based browser.

JavaScript customization

This section covers how to make intelligent customizations when you use JavaScript that help you build performant forms and pages in a model-driven app.

Using JavaScript with forms

The ability for forms to be customized by JavaScript provides professional developers great flexibility over how a form looks and behaves. The improper use of this flexibility can negatively impact form performance. Developers should use the following strategies to maximize form performance when implementing JavaScript customizations.

Use asynchronous network requests when requesting data

Request data asynchronously rather than synchronously when extra data is necessary for customizations. For events that support waiting for asynchronous code like the form `OnLoad` and form `OnSave` events, event handlers should return a `Promise` in order for the platform to wait until the `Promise` is settled. The platform will show an appropriate UI while the user waits for the event to complete.

For events that do not support waiting for asynchronous code, like the form `OnChange` event, you can use a workaround to stop interaction with a form while the code is doing an asynchronous request by using `showProgressIndicator`. This is better than using synchronous requests because users will still be able to interact with other parts of the application as a progress indicator is displayed.

Here's an example using asynchronous code in synchronous extension points.

JavaScript

```
//Only do this if an extension point does not yet support asynchronous code
try {
    await Xrm.WebApi.retrieveRecord("settings_entity", "7333e80e-9b0f-49b5-
92c8-9b48d621c37c");
    //do other logic with data here
} catch (error) {
    //do other logic with error here
} finally {
    Xrm.Utility.closeProgressIndicator();
}

// Or using .then/.finally
Xrm.Utility.showProgressIndicator("Checking settings...");
Xrm.WebApi.retrieveRecord("settings_entity", "7333e80e-9b0f-49b5-92c8-
9b48d621c37c")
    .then(
        (data) => {
            //do other logic with data here
        },
        (error) => {
            //do other logic with error here
        }
    )
    .finally(Xrm.Utility.closeProgressIndicator);
```

You should be careful when using asynchronous code in an event handler that does not support waiting for asynchronous code. This is particularly true for code that needs an action to be taken or handled on the resolution of the asynchronous code. Asynchronous code can cause issues if the resolution handler expects the application context to remain the same as it was when the asynchronous code was started. Your

code should check that the user is in the same context after each asynchronous continuation point.

For example, there may be code in an event handler to make a network request and change a control to be disabled based on the response data. Before the response from the request is received, the user may have interacted with the control or navigated to a different page. Because the user is on a different page, the form context may not be available, which might lead to errors, or there might be other undesired behavior.

Async support in form OnLoad and form OnSave events

The form `OnLoad` and `OnSave` events support handlers that return promises. The events will wait for any promises returned by a handler to resolve, up to a timeout period. This support can be enabled via app settings.

More information:

- Form OnLoad
- Form OnSave
- Interact with HTTP and HTTPS resources asynchronously

Limit the amount of data requested during form load

Only request the minimum amount of data that is necessary to perform business logic on a form. Cache the data that is requested as much as possible, especially for data that doesn't change often or doesn't need to be fresh. For example, imagine there is a form that requests data from a *setting* table. Based on data in the setting table, the form might choose to hide a section of the form. In this case, the JavaScript can cache data in `sessionStorage` so that data is only requested once per session (`onLoad1`). A stale-while-revalidate strategy might also be used where the JavaScript uses the data from `sessionStorage` while requesting data for the next navigation to the form (`onLoad2`). Finally, a deduplication strategy could be used in case a handler is called multiple times in a row (`onLoad3`).

JavaScript

```
const SETTING_ENTITY_NAME = "settings_entity";
const SETTING_FIELD_NAME = "settingField1";
const SETTING_VALUE_SESSION_STORAGE_KEY =
`${SETTING_ENTITY_NAME}_${SETTING_FIELD_NAME}`;

// Retrieve setting value once per session
async function onLoad1(executionContext) {
    let settingValue =
```

```
sessionStorage.getItem(SETTING_VALUE_SESSION_STORAGE_KEY);

    // Ensure there is a stored setting value to use
    if (settingValue === null || settingValue === undefined) {
        settingValue = await requestSettingValue();
    }

    // Do logic with setting value here
}

// Retrieve setting value with stale-while-revalidate strategy
async function onLoad2(executionContext) {
    let settingValue =
sessionStorage.getItem(SETTING_VALUE_SESSION_STORAGE_KEY);

    // Revalidate, but only await if session storage value is not present
    const requestPromise = requestSettingValue();

    // Ensure there is a stored setting value to use the first time in a
session
    if (settingValue === null || settingValue === undefined) {
        settingValue = await requestPromise;
    }

    // Do logic with setting value here
}

// Retrieve setting value with stale-while-revalidate and deduplication
strategy
let requestPromise;
async function onLoad3(executionContext) {
    let settingValue =
sessionStorage.getItem(SETTING_VALUE_SESSION_STORAGE_KEY);

    // Request setting value again but don't wait on it
    // In case this handler fires twice, don’t make the same request again
if it is already in flight
    // Additional logic can be added so that this is done less than once per
page
    if (!requestPromise) {
        requestPromise = requestSettingValue().finally(() => {
            requestPromise = undefined;
        });
    }

    // Ensure there is a stored setting value to use the first time in a
session
    if (settingValue === null || settingValue === undefined) {
        settingValue = await requestPromise;
    }

    // Do logic with setting value here
}

async function requestSettingValue() {
```

```
    try {
        const data = await Xrm.WebApi.retrieveRecord(
            SETTING_ENTITY_NAME,
            "7333e80e-9b0f-49b5-92c8-9b48d621c37c",
            `?$select=${SETTING_FIELD_NAME}`);
        try {
            sessionStorage.setItem(SETTING_VALUE_SESSION_STORAGE_KEY,
data[SETTING_FIELD_NAME]);
        } catch (error) {
            // Handle sessionStorage error
        } finally {
            return data[SETTING_FIELD_NAME];
        }
    } catch (error) {
        // Handle retrieveRecord error
    }
}
```

Use information available in the client API rather than make requests. For example, rather than request a user's security roles on form load, you can use getGlobalContext.userSettings.roles.

Load code only when it's needed

Load as much code as needed for events for a particular form. If you have code that is only for *form A* and *form B*, it shouldn't be included in a library that is loaded for *form C*. It should be in its own library.

Avoid loading libraries in the `OnLoad` event if they are only used for the `OnChange` or `OnSave` events. Instead, load them in those events. This way the platform can defer loading them until after the form loads. More information: Optimize form performance

Remove usage of console APIs in production code

Don't use the console API methods such as `console.log` in production code. Logging data to the console can significantly increase memory demand and might prevent data from being cleaned up in memory. This can lead to the app becoming slower over time and eventually crashing.

Avoid memory leaks

Memory leaks in your code can lead to slower performance over time and eventually cause your app to crash. Memory leaks occur when the application fails to release

memory when no longer needed. With all customizations and code components on your form, you should:

- Thoroughly consider and test scenarios for anything responsible for cleaning up memory, like classes responsible for managing lifecycle of objects.
- Cleanup all event listeners and subscriptions, especially if it's on the `window` object.
- Cleanup all timers like `setInterval`.
- Avoid, limit, and cleanup references to global or static objects.

For custom control components, cleanup can be done in the destroy method.

For more information on fixing memory problems, go to this Edge developer documentation.

Tools you can use to help make apps performant

This section describes the tools that can help you understand performance issues and offer recommendations about how to optimize your customizations in model-driven apps.

Performance insights

Performance insights is a self-service tool for enterprise app makers that analyzes runtime telemetry data and provides a prioritized list of recommendations to help improve the performance of model-driven apps. This feature provides a daily set of analytic insights related to the performance of a Power Apps model-driven or customer engagement app, such as Dynamics 365 Sales or Dynamics 365 Service, with recommendations and actionable items. Enterprise app makers can view detailed performance insights at an app-level in Power Apps. More information: What are performance insights? (preview)

Solution checker

Solution checker is a powerful tool that can analyze client and server customizations for performance or reliability issues. It can parse client-side JavaScript, form XML, and .NET server-side plug-ins and give targeted insights into what may slow end users down. We recommend that you run solution checker each time you publish changes in a development environment, so that any performance concerns are surfaced before

reaching end users. More information: Use solution checker to validate your model-driven apps in Power Apps

Some examples of performance-related issues found with solution checker:

- il-specify-column. Avoid selecting all columns via Dataverse query APIs.
- web-use-async. Interact with HTTP and HTTPS resources asynchronously.
- web-avoid-ui-refreshribbon. Avoid using `refreshRibbon` in form `OnLoad` and `EnableRule`.

Object checker

Object checker runs real-time diagnostics on component objects within your solution. If issues are detected, a recommendation is returned that describes how to fix the issue. More information: Use object checker to diagnose a solution component (preview)

Next steps

Design productive main forms in model-driven apps

Embed a Power BI report in a model-driven app main form

You can use Power BI reports in Power Apps model-driven apps to bring rich reporting and analytics to your main forms and empower your users to accomplish more. This unlocks the power to aggregate data across systems, and tailor it down to the context of a single record.

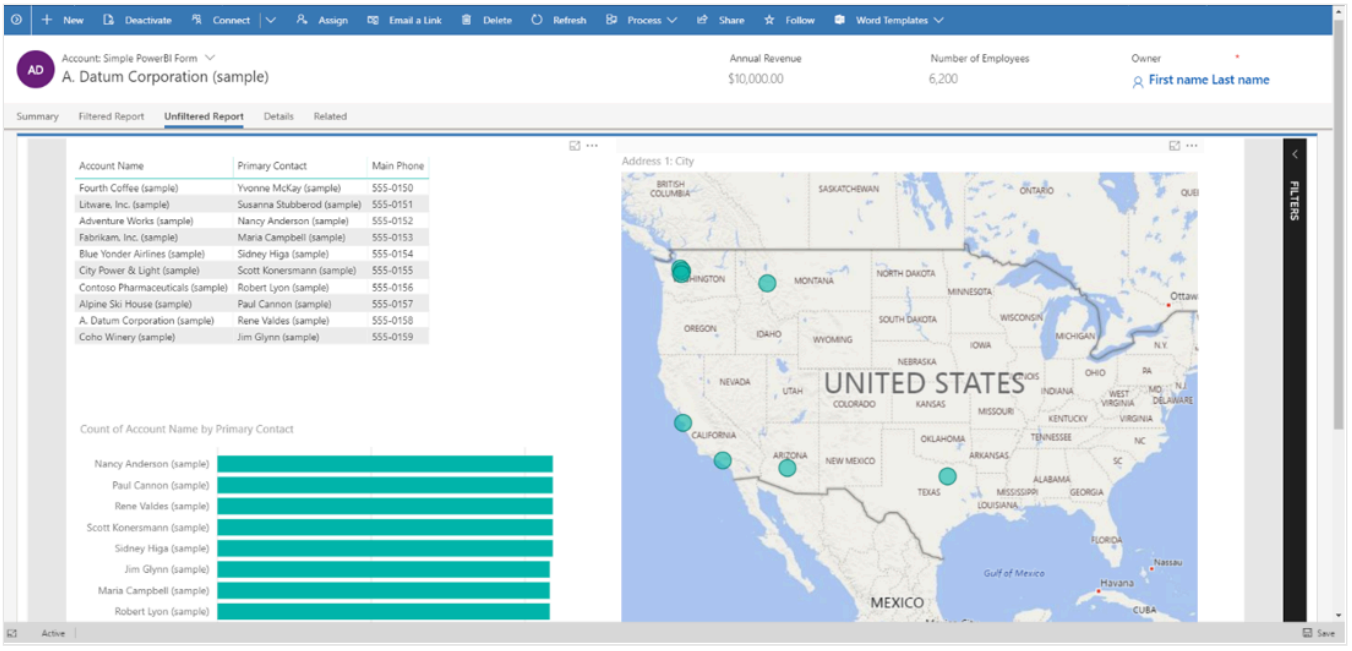
Prerequisites

Embedding Power BI content is an optional feature and is disabled on all environments by default. You must enable it before you can embed Power BI content. More information: Enable Power BI visualizations in the organization.

This feature requires exporting a solution, modifying it to replace the XML snippet, and then importing back into the environment. Be sure to import the changes on your development environment via an unmanaged solution only. Go to Import, update, and export solutions for guidance on installing an update to an existing unmanaged solution.

Embed without contextual filtering

You can use Power BI reports by embedding them, and get the exact same report. This doesn't involve contextualizing them to the current model-driven form, and hence you get the same report on all records of the table. For example, the following report shows the geographic location of all accounts at once, and is useful to show summary information.

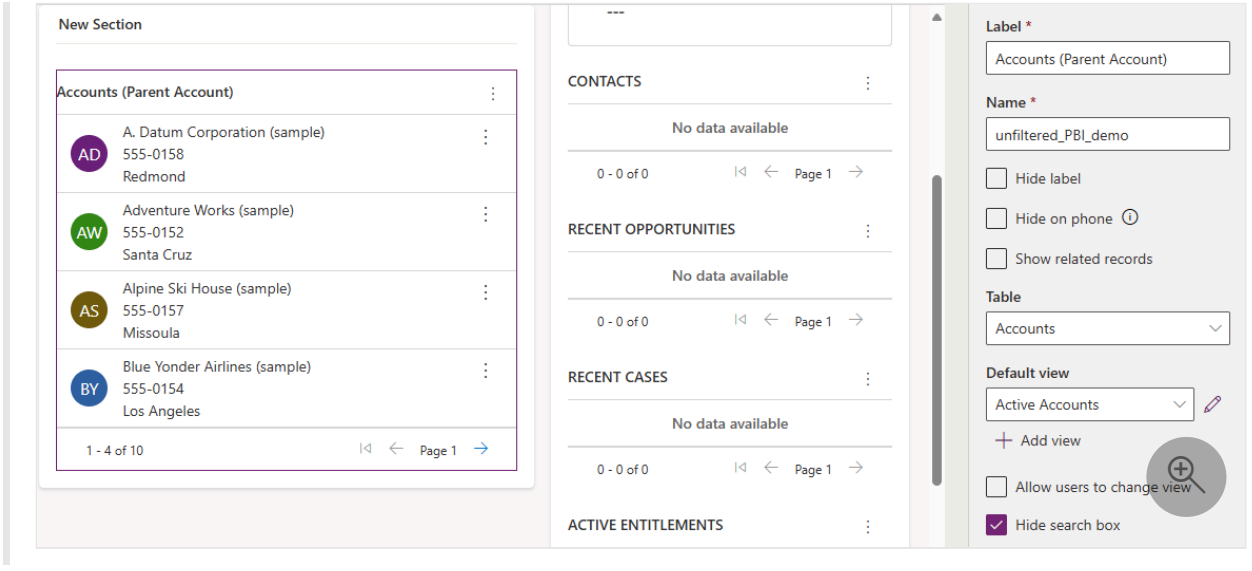


You can customize a main form XML `controls` node so that it can host an embedded Power BI report and tile by following these steps:

1. In your development environment, create a solution and add the table that contains the main form where you want the embedded Power BI report to display. Choose the **Select components** option when you add the table to the solution. Then, add just the table's main form.

2. Edit the table's main form in your solution and add a placeholder subgrid where you want to embed your Power BI report.

    - Update the label and name of your subgrid in the form designer.



3. Export the solution as unmanaged and extract all files in the solution package zip file. Then, edit the customizations.xml file and find the name of your subgrid. Replace the XML

code provided inside the `<control>` block in the customizations.xml file. The `rowspan` in the parent `cell` node can be adjusted to change the size of the embedded Power BI report.

XML

```
<cell id="{7d18b61c-c588-136c-aee7-03e5e74a09a1}" showlabel="true"
rowspan="20" colspan="1" auto="false" solutionaction="Added">
 <labels>
    <label languagecode="1033" description="Accounts (Parent Account)"/>
 </labels>
 <control id="unfilteredreport" classid="{8C54228C-1B25-4909-A12A-
F2B968BB0D62}">
    <parameters>
        <PowerBIGroupId>00000000-0000-0000-0000-000000000000</PowerBIGroupId>
        <PowerBIReportId>544c4162-6773-4944-900c-
abfd075f6081</PowerBIReportId>
        <TileUrl>https://app.powerbi.com/reportEmbed?reportId=544c4162-6773-
4944-900c-abfd075f6081</TileUrl>
    </parameters>
 </control>
</cell>
```

> **Important**
>
> Be sure to use the control `classid="{8C54228C-1B25-4909-A12A-F2B968BB0D62}"` as indicated in the XML sample.

- Make the following changes to the form XML that you copied into the customizations.xml file for your Power BI report. This table describes the values you need to change to the elements in the previous XML sample.

Expand table

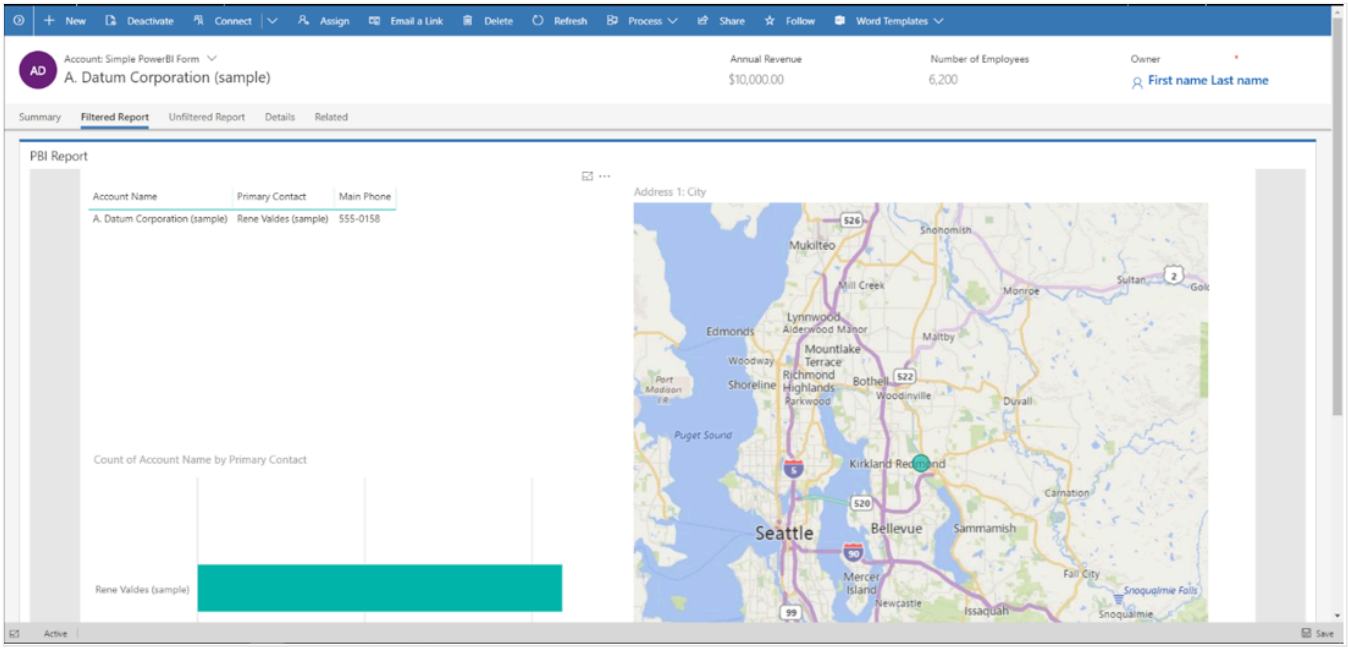
Property	Description
PowerBIGroupId	The Power BI workspace ID. If your report is in **My workspace**, then the workspace ID is 00000000-0000-0000-0000-000000000000. Otherwise, add the workspace ID. You can find the ID of the workspace in the Power BI service URL. More information: Find the Power BI workspace and report Ids.
PowerBIReportId	The Power BI report ID. Replace this with the report that you want to embed. You can find the ID of your report in the Power BI service URL. More information: Find the Power BI workspace and report Ids

Property	Description
TileUrl	The Power BI report URL that you want to embed. Be sure to use the correct Power BI subdomain name (you might need to replace app.powerbi.com with your own) and report ID (replace *reportId=544c4162-6773-4944-900c-abfd075f6081* with your own). For example, `https://app.powerbi.com/reportEmbed?reportId=544c4162-6773-4944-900c-abfd075f6081`.

4. Create a zip file out of all the solution files previously extracted. Then, import the solution into the development environment.

Embed with contextual filtering

You can make Power BI reports more meaningful by applying contextual filters to the current model-driven form, so that the report is filtered based on attributes of the current row. For example, the following report shows the geographic location of an account, by filtering the Power BI report using the account name. This allows a single report to show contextualized information for all rows of the table.



The filtering is done by adding a `<PowerBIFilter>` element in the `<parameter>` block as shown here. You can use any attribute of the form's table to construct the filter expression. More information: Constructing Filters to understand how to create your own filters.

XML

```
<control id="filteredreport" classid="{8C54228C-1B25-4909-A12A-F2B968BB0D62}">
    <parameters>
        <PowerBIGroupId>00000000-0000-0000-0000-000000000000</PowerBIGroupId>
```

```
        <PowerBIReportId>544c4162-6773-4944-900c-abfd075f6081</PowerBIReportId>
        <TileUrl>https://xyz.powerbi.com/reportEmbed?reportId=544c4162-6773-4944-
900c-abfd075f6081</TileUrl>
        <PowerBIFilter>{"Filter": "[{\"$schema\":\"basic\",\"target\":
{\"table\":\"table_name_of_power_bi_dataset\",\"column\":\"power_bi_field\"},\"oper
ator\":\"In\",\"values\":[$a],\"filterType\":1}]", "Alias": {"$a":
"field_name_in_powerapps_to_filter"}}</PowerBIFilter>
    </parameters>
</control>
```

Note that this uses the same control as the unfiltered report embedding, and hence the control class ID remains unchanged.

This table describes any additional properties used in the previous XML sample.

Expand table

Property	Description
PowerBIFilter	The filter expression that contextualizes the Power BI report by passing the form attributes as parameters. To make it more readable, the filter is constructed as shown here. The filter expression can't be a comma separated list of values.

JSON

```
{
        "Filter": "[{
                \"$schema\":\"basic\",
                \"target\":{
                        \"table\":\"table_name_of_power_bi_dataset\",
                        \"column\":\"power_bi_field\"
                },
                \"operator\":\"In\",
                \"values\":[$a, $b],
                \"filterType\":1
        }]",
        "Alias": {
                "$a": "field1_name_in_powerapps_to_filter",
                "$b":"field2_name_in_powerapps_to_filter"
        }
}
```

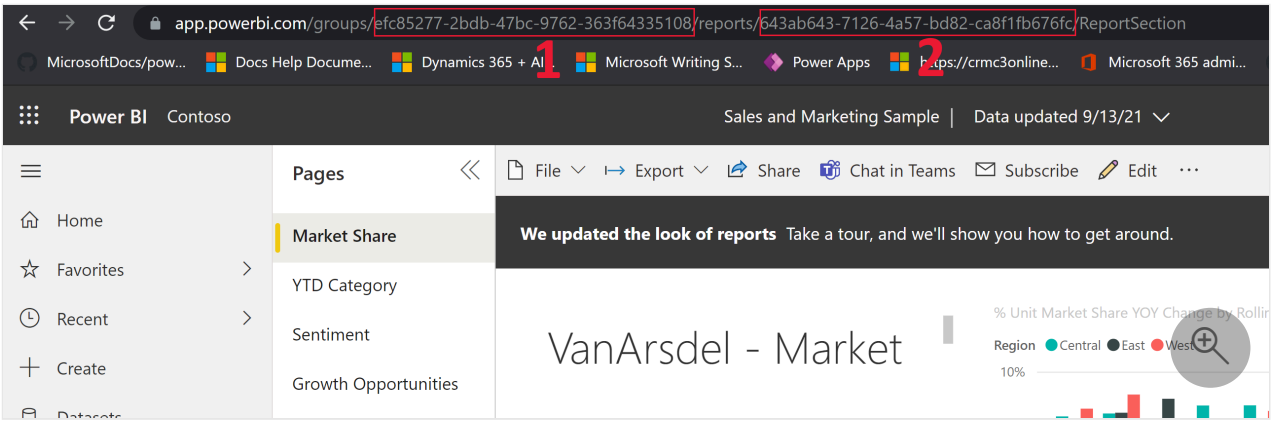
The target part of the previous expression identifies the table and the column to apply the filters on. The operator identifies the logic and values identify the data passed from the Power Apps model-driven app. To parameterize in a generic way, the values are constructed by aliasing. In the previous expression, the value of an account's **firstname** and **lastname** are passed, and either of them is searched in the **Account Name** column in the Power BI report.

Note that **firstname** and **lastname** are the unique names of attributes of the account table, whose value will be passed here.

You can create more complex filter expressions by looking at examples from Constructing Filters and providing the appropriate values for $schema and filterType. Be sure to escape every literal in the filter part using ", so that the JSON is generated correctly.

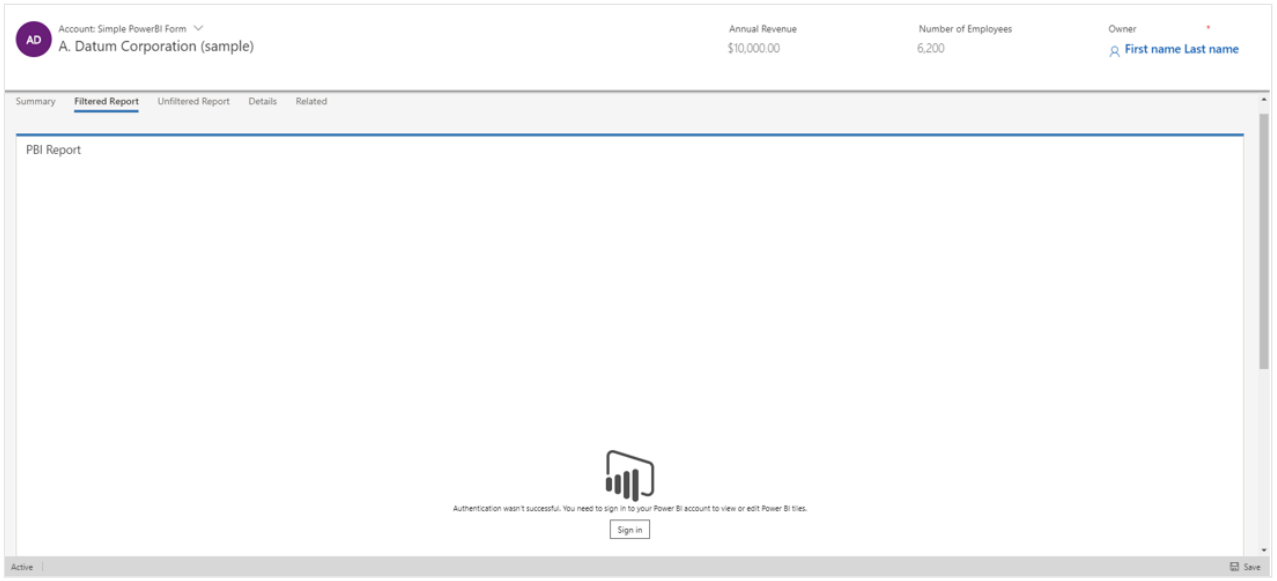
Find the Power BI workspace and report Ids

1. The workspace ID for this example is efc85277-2bdb-47bc-9762-363f64335108.
2. The report ID for this example is 643ab643-7126-4a57-bd82-ca8f1fb676fc.

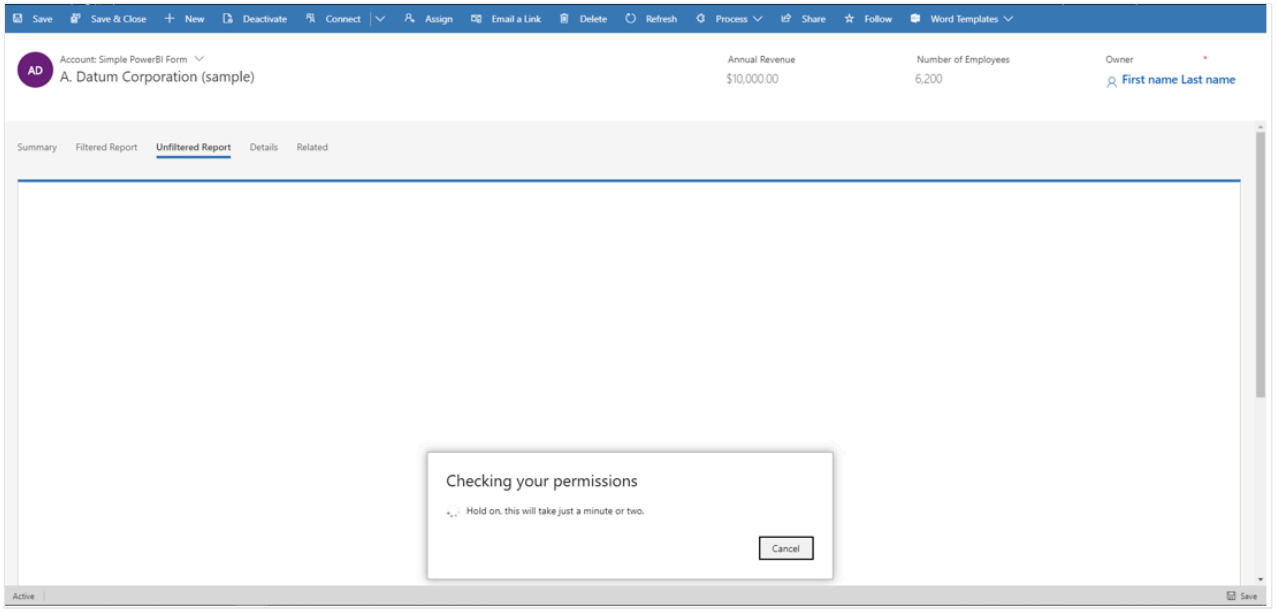


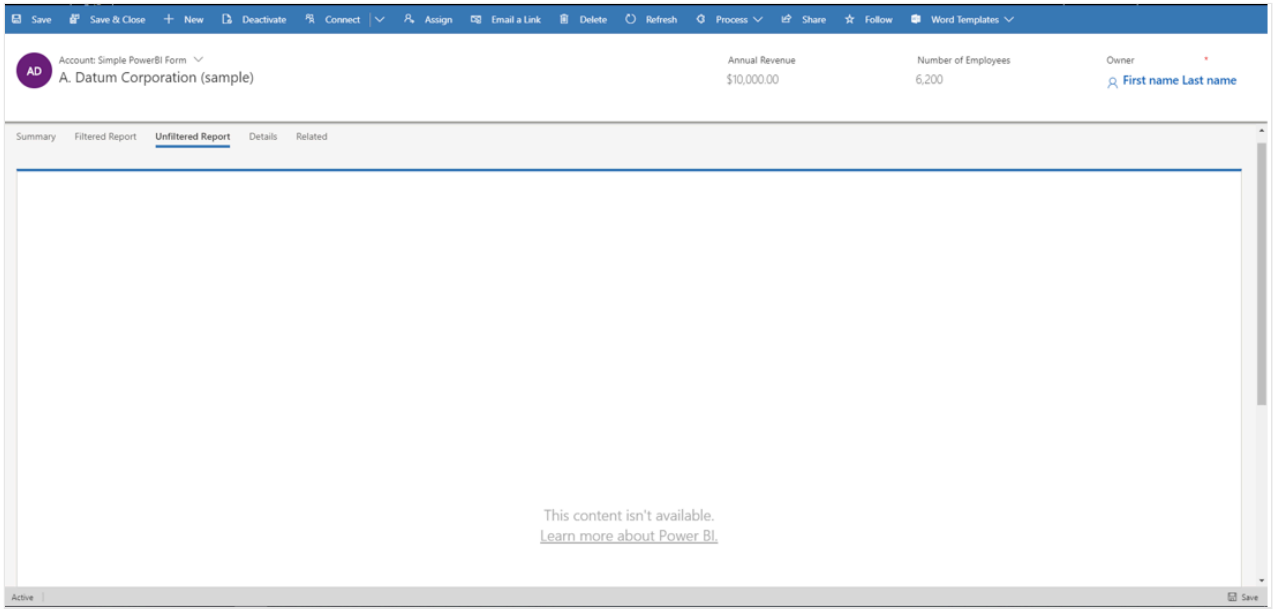
Known issues and limitations

- Opening this form in the Power Apps form designer doesn't show the control in a meaningful way. This is because the control is customized outside of the form designer.
- Users are authenticated into Power BI automatically with their Power Apps username and password. If a Power BI account with matching credentials doesn't exist, a sign-in prompt is displayed as illustrated here.



No data displays if an incorrect account is used to log into Power BI. To sign in with the correct credentials, sign out, and then sign in again.





- The view of the report data shown inside Power Apps is the same as Power BI, and Power Apps security roles and privileges don't affect the data that is displayed. Hence, the data is essentially the same as what the creator of the Power BI dataset would see. To apply data access restrictions similar to Power Apps security roles and teams, use Row-level security (RLS) with Power BI.
- If the form doesn’t show the Power BI report after importing the solution and publishing customizations, open it in the model-driven form designer and save it, so that the form JSON is regenerated.
- Embedding a Power BI tile on a form is only supported without contextual filtering.
- Embedding a Power BI report doesn't support cross-tenant references. For example, trying to embed a Power BI report belonging to the Contoso tenant in a model-driven app belonging to the Fabrikam tenant isn't currently supported, even if the report being embedded has been shared with users in the Fabrikam tenant via Power BI.
- Pinned reports as part of Power BI dashboards aren't meant to be interactive. If you want to have interactive Power BI reports, embed them directly instead of pinning them to a dashboard.

Common issues

- The group ID isn't specified in the `TileUrl` node of the control parameters when it might need to be. This example includes a group ID.

XML

```
<parameters>
    <PowerBIGroupId>fd266a4c-9a02-4553-9310-80e05ee844f3</PowerBIGroupId>
    <PowerBIReportId>544c4162-6773-4944-900c-abfd075f6081</PowerBIReportId>
    <TileUrl>https://xyz.powerbi.com/reportEmbed?reportId=544c4162-6773-4944-900c-
abfd075f6081&amp;groupId=fd266a4c-9a02-4553-9310-80e05ee844f3</TileUrl>
</parameters>
```

- Fields have different data types in Power BI and Dataverse. They need to be the same type, such as string in Power BI and string in Dataverse.
- String fields don't have escaped quotes in the Power BI filter. Notice `values` has `[\"$a\"]` rather than `[$a]`.

JSON

```
{
        "Filter": "[{
                \"$schema\":\"basic\",
                \"target\":{
                        \"table\":\"table_name_of_power_bi_dataset\",
                        \"column\":\"power_bi_field\"
                },
                \"operator\":\"In\",
                \"values\":[\"$a\"],
                \"filterType\":1
        }]",
        "Alias": {
                "$a": "field_name_in_powerapps_to_filter",
        }
}
```

See also

Embed a Power BI dashboard in a Power Apps model-driven personal dashboard

Use Power BI with Dynamics 365 apps

Last updated on 04/03/2026

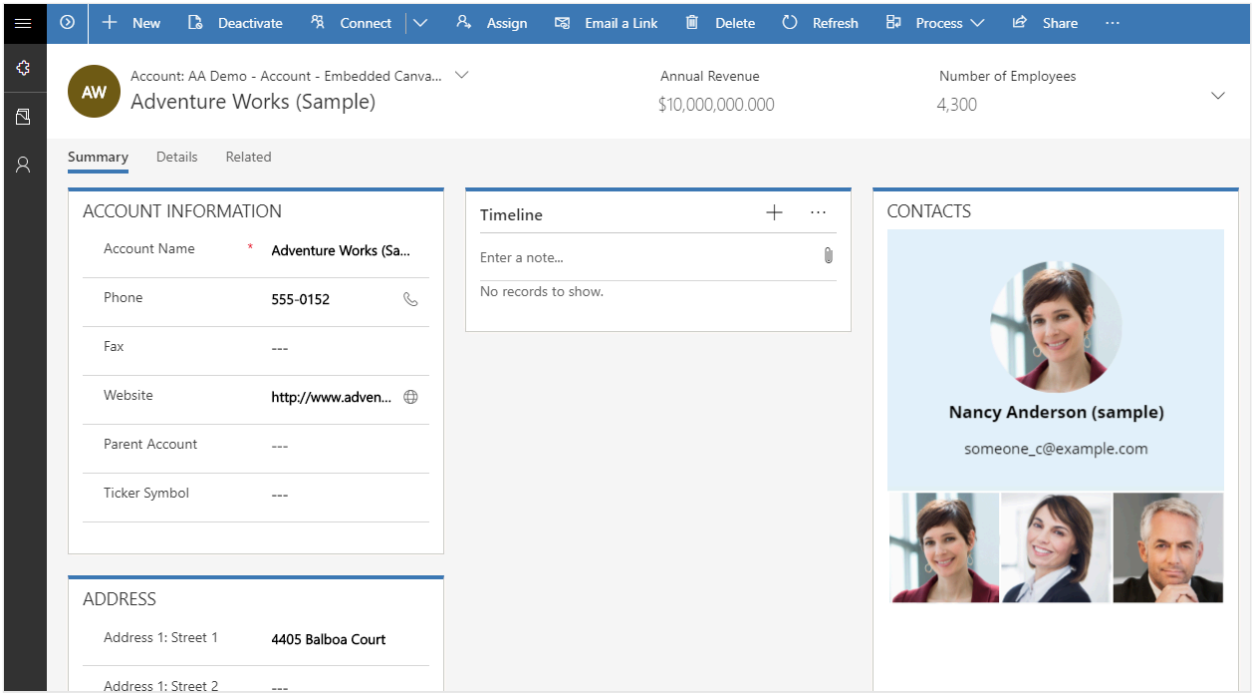
Embed a canvas app on a model-driven form

Article • 01/22/2025

Canvas apps enable makers to easily design and create custom layouts using the low-code, WYSIWYG canvas app designer. Canvas apps also enable makers to connect and display data from over 400 data sources.

Embedded canvas apps can help you build better solutions

With embedded canvas apps, makers can bring the power of canvas apps to their model-driven app forms. Using embedded canvas apps, it's possible to create rich visual areas on a form and display data from a variety of sources right next to data from Microsoft Dataverse.



Canvas apps are embedded in model-driven app forms in the same way other custom controls are added. An embedded canvas app includes rich data integration capabilities that bring in contextual data from the host model-driven form to the embedded canvas app.

Integrate canvas apps using custom pages in the modern app designer

With the modern app designer, canvas apps can be accessed through a model-driven app by using a custom page.

The modern app designer introduced the concept of pages, which can contain either canvas apps or Dataverse components, such as tables, dashboards, and web resources.

Embed using the modern designer

Select the **Canvas app** control to add a canvas app to a main form using the form designer. More information: Add an embedded canvas app on a model-driven form

Embed using the classic designer

Add canvas apps from an environment into a model-driven app that can either be contextually aware, or simply render the app within the model-driven experience. More information: Embed a canvas app using the classic experience.

Sharing embedded canvas apps

After an embedded canvas app is added to your model-driven form, learn how to share the embedded canvas app with other users. More information: Learn how to share an embedded canvas app.

For guidelines on working with embedded canvas apps as well as helpful tips to troubleshoot any issues, go to: Guidelines on working with embedded canvas apps.

See also

What are canvas apps in Power Apps?
Add and configure a canvas-app control in Power Apps
Overview of canvas-app connectors for Power Apps
Add an embedded canvas app on a model-driven form
Edit a canvas app embedded on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Guidelines on working with embedded canvas apps

Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

Feedback

Was this page helpful? Yes No

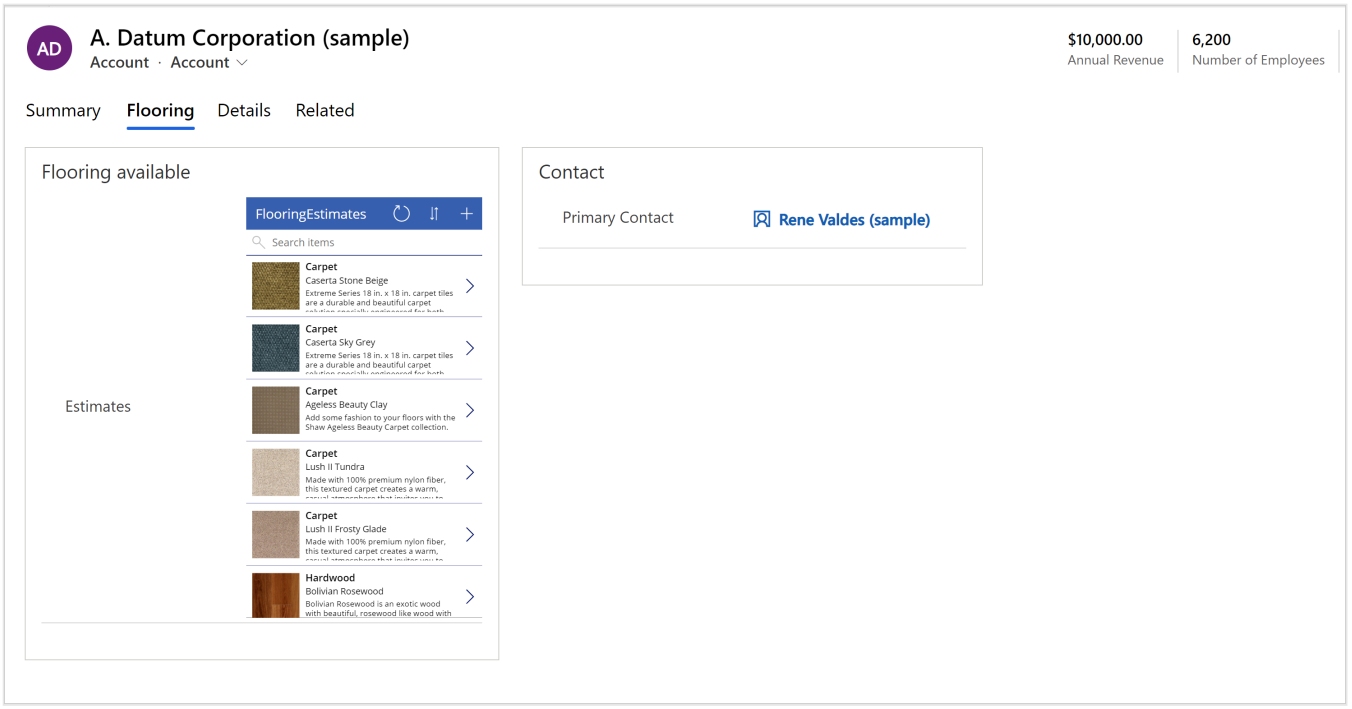
Provide product feedback

Add an embedded canvas app on a model-driven form

This article explains how to embed a canvas app on a model-driven form. Canvas apps can be connected to a wide range of data sources and offer designers greater control around the overall user experience.

Two methods of integration are described in this article. First, using the modern Unified Interface experience, and second, using the classic experience.

> **ⓘ Important**
>
> - There is a third and more recent method of embedding canvas apps inside a modern app, which uses custom pages for better integration with the canvas app. More information: Overview of custom pages for model-driven apps.
> - Both the modern form and classic form designers allow you to add an existing canvas app to embed in a model-driven form. However, currently only the classic form designer allows you to create a new canvas app to embed. More information: Embed a canvas app using the classic experience
> - To enable access to form data through the `ModelDrivenFormIntegration` control, the embedded canvas app *must be created using the classic form designer*. Apps *added* through the modern or classic experience don't currently support this integration. For more information about how to create a canvas app to embed on a model-driven form using the classic form designer go to Embed a canvas app using the classic experience. For more information about the `ModelDrivenFormIntegration` control go to ModelDrivenFormIntegration control properties and actions.



Embed a canvas app using the modern experience

To embed a canvas app on a model-driven app main form, such as the main form for the Accounts table, follow these steps:

1. Sign in to Power Apps.
2. Choose the environment, and then select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution that has the table where you want to add a canvas app to a main form.
4. Open the table, select **Forms**, and then open the **Main** form you want.

   > **Tip**
   >
   > By default, the canvas app component label is **New canvas app**. You might want to change the canvas app label name to something more meaningful, such as the name of the canvas app or a short description of the canvas app that you're embedding onto the form.

5. Select the area on the form where you want the canvas app to display or create a new tab to display the canvas app.
6. On the left navigation pane select **Components**, expand **Display**, and then select the **Canvas app** component.

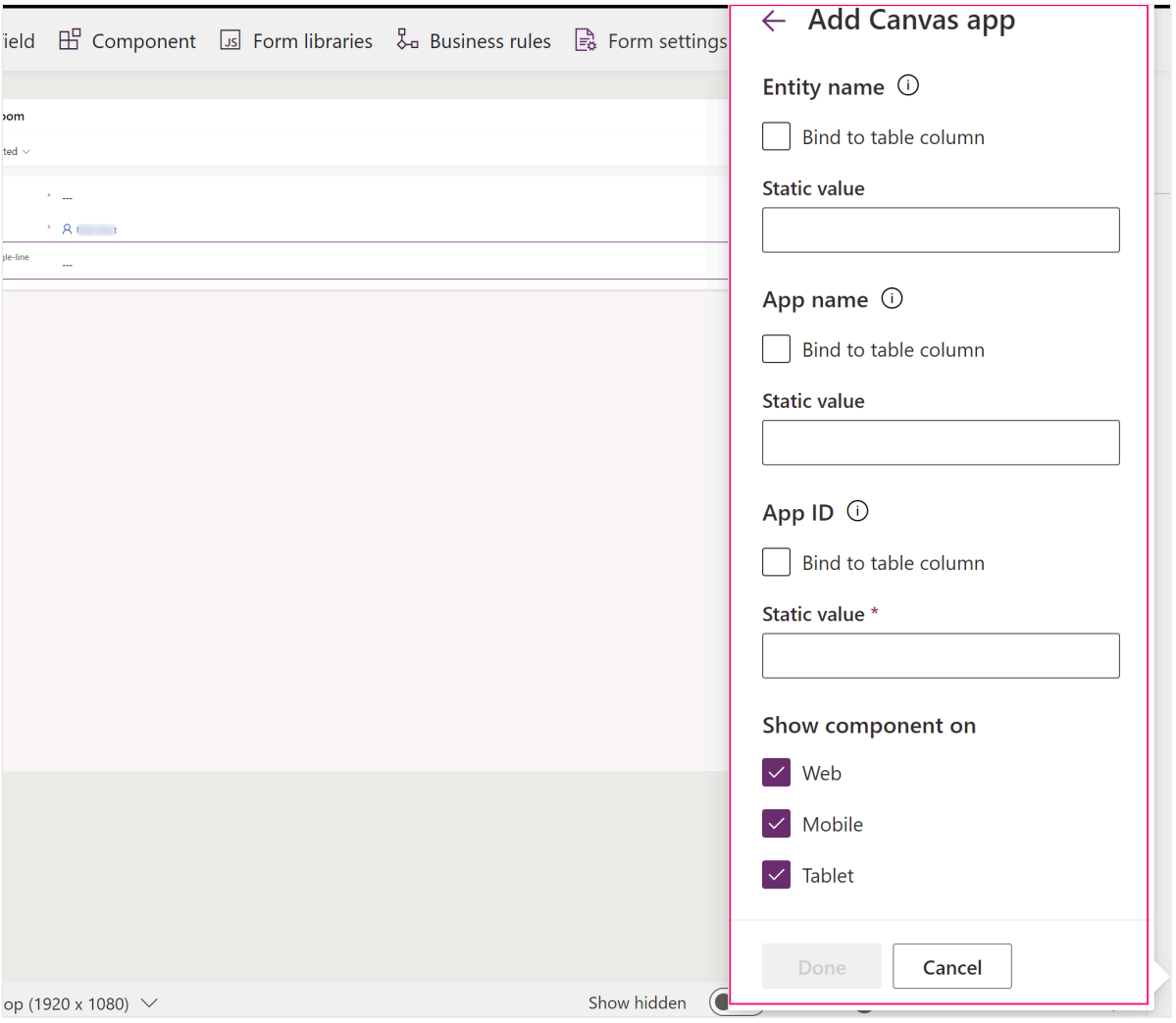
7. The following properties can be configured for the control.

- The **Entity name** area lets you specify the following properties:
  - **Bind to table column**: When the **Bind to table column** is selected, the **Static value** property changes to allow you to select what table column to bind to. When you choose to bind to a table column, the column will be defined by the table column value. For example, when **Bind to table column** is selected for the **Entity name**, at app runtime the value of the table name takes the value of the **Table column** specified.
  - **Static value**: Use this to add a name for you to refer to the canvas app. Notice that this value has no effect on the embedded canvas app. It's meant only to serve as a reference point.
- The **App name** area specifies the unique name of the canvas app.
  - **Bind to table column**: When you select the **Bind to table column**, the **Static value** property changes to allow you to select what table column to bind to. When you choose to bind to a table column, the column will be defined by the table column value. For example, when you select **Bind to table column** for the **App Name**, at app runtime the value of the app name takes the value of the Table column, such as the value from a custom text column.
  - **Static value**: Specifies the unique name of the canvas app to embed. Enter the canvas app's unique name for an existing canvas app. More information: How to find the unique name and app ID for a canvas app
- The required **App ID** area specifies the app ID for the canvas app.
  - **Bind to table column**: When you select **Bind to table column**, the list changes to allow you to select what table column to bind to. When you choose to bind to a table column, the column will be defined by the table column value. For example, when you select **Bind to table column** for the **App Name**, at app runtime the value of the app name takes the value of the column, such as the value from a custom text column.

    > **Important**
    >
    > For most situations, we don't recommend using the **Bind to table column** option for the App ID as this would require the correct app ID to be shown in a field within the record.

  - **Static value**: Enter the app ID for an existing canvas app. More information: How to find the unique name and app ID for a canvas app

- **Show component on**. Specifies the client type to display the embedded canvas app. Choose from **Web**, **Mobile**, and **Tablet**.



8. Select **Done**. Then, to publish your app to make it available to users, select **Save and publish**.

After you have published an embedded canvas app to your model-driven form, share your embedded canvas app with other users. More information: Share an embedded canvas app.

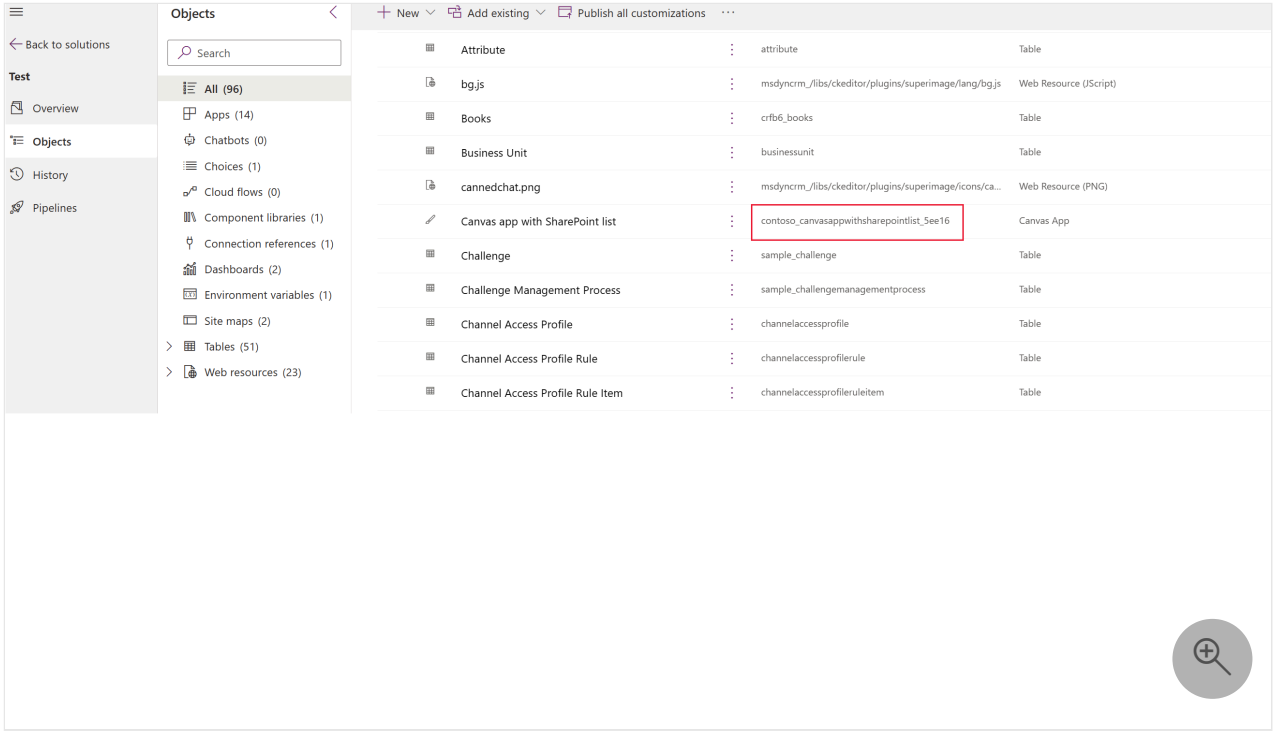
When users open a model-driven app that includes the form you have modified, they see the embedded canvas app on the form. Changing the record displayed on the main form changes the data context that is passed to the form, and the embedded app refreshes to show the relevant data.

This article showed you how to get started with embedding a canvas app in a model-driven form. You can further customize the embedded canvas app to connect and bring in data from various data sources. Use the `Filter`, `Search`, and `LookUp` functions and the context passed in from the host model-driven form to filter or find specific records in those data sources. Use the WYSIWYG canvas app editor to easily design the interface to match your requirements.

How to find the unique name and app ID for a canvas app

The canvas app control looks for the canvas app first by using the *unique* name found in the **App name** field. If a canvas app with that unique name can't be found in the environment, the canvas app control next uses the specified value in the **App ID** field to load the canvas app. Although you must enter values in both the App name and App ID fields, only one of the values needs to be accurate for the embedded canvas app to load in the control. Note that the canvas app must be in the current environment.

- The **App name** field is the unique name of the canvas app. To find the unique name, add the canvas app to a solution. The canvas app unique name is displayed in the **Name** column and appears in the form *solutionpublisherprefix_appnamenospaces_appendix*. For example, *contoso_canvasappwithsharepointlist_5ee16*.



- The **App ID** property specifies the ID of the embedded canvas app. Enter the App ID for an existing canvas app. The canvas App ID can be found on Power Apps, go to **Apps**, select the canvas app, and then select **Details** on the command bar.

Embed a canvas app using the classic experience

1. Sign in to Power Apps.
2. Create or edit the main form of a table, Accounts table in our example.
3. In the command bar, select **Switch to classic** to open the form in the classic form designer.

4. In the classic form designer, select the section on the form where you want the embedded canvas app to appear.

5. Using the column pane, add a required column, such as **Account Name**.

   > **Important**
   >
   > Always use a required column that is guaranteed to have a value. If your column doesn't have a value, your embedded canvas app won't refresh in response to any change in data on the host model-driven form.

6. With the column selected, on the **Home** tab in the **Edit** group, select **Change Properties**.

7. On the **Column Properties** dialog box, select the **Controls** tab.

8. On the **Controls** tab, select **Add Control**.

9. On the **Add Control** dialog box, in the list of available controls, select **Canvas app** and then select **Add**.

10. On the **Column Properties** dialog box, in the list of controls select **Canvas app**, and then select the **Web** option.

11. If you want to create a new canvas app, go to the next step. To add an existing canvas app, in the section below the controls list, the lists of properties available to the canvas app control are displayed.

    - The **Table name** property specifies the table that provides the data to your embedded canvas app. It is set to the table that contains the column you added in an earlier step.
      - Notice that, even though this property appears changeable, changing it has no effect on the embedded canvas app. It's meant only to serve as a reference for you.
    - The **App Name** property specifies the name of the canvas app to embed. The model-driven form looks for the canvas app with the specified name in the current environment. If a canvas app with that name can't be found in the environment, the canvas app control uses the App ID to load the canvas app. Enter the App Name for an existing canvas app.
    - The **App ID** property specifies the ID of the embedded canvas app. You can enter an App ID for an existing canvas app or an App ID will be automatically generated and filled-in for you when you create or edit a canvas app by selecting **Customize**. The canvas App ID can be found on Power Apps, go to **Apps**, select the canvas app, and then select **Details** on the command bar.

> ⓘ Note
>
> Notice that any change to the **App ID** value breaks the link from the model-driven form to the embedded canvas app.
>
> If neither the App Name or App ID matches an existing canvas app, the form displays the message "Sorry, we didn't find that app."

12. Go to step 20 if you're adding a completed canvas app. If you're creating a new or editing an existing canvas app, continue to follow these steps. Select **Customize** to create or edit a canvas app. This opens Power Apps Studio in a new tab.

> ⓘ Note
>
> If opening Power Apps Studio is blocked due to a web browser pop-up blocker you must enable the make.powerapps.com site or temporarily disable the pop-up blocker and then select **Customize** again. 13. Observe the following:

- In Power Apps Studio notice that there's a special `ModelDrivenFormIntegration` control in the left pane. This control is responsible for bringing contextual data from the host model-driven form to the embedded canvas app.
- Observe that a canvas app form control was automatically added to your embedded canvas app and displays the data being passed to it from the host model-driven form via the ModelDrivenFormIntegration control.
- Select the **View** tab, and then select **Data sources**. Notice that a data source for the parent table of your host model-driven form, Accounts in this case, was automatically added to your embedded canvas app. - Select the **Form1** control and observe that the **DataSource** property is set to **Accounts**.
  - With the **Form1** control still selected, observe that the **Item** property is set to **ModelDrivenFormIntegration.Item**.

> ⓘ Note
>
> The embedded canvas app has full access to read from the host model-driven form via ModelDrivenFormIntegration.Item. > As an example, to get the value of a column with the name **accountnumber** and display name **Account Number**, you can use **ModelDrivenFormIntegration.Item.accountnumber** or **ModelDrivenFormIntegration.Item.'Account Number'**. 14. In the property pane on the right, next to **Columns**, select **Edit columns**. 15. Select **+ Add column** to add another column to the canvas app form or reorder existing columns using drag and drop. Close

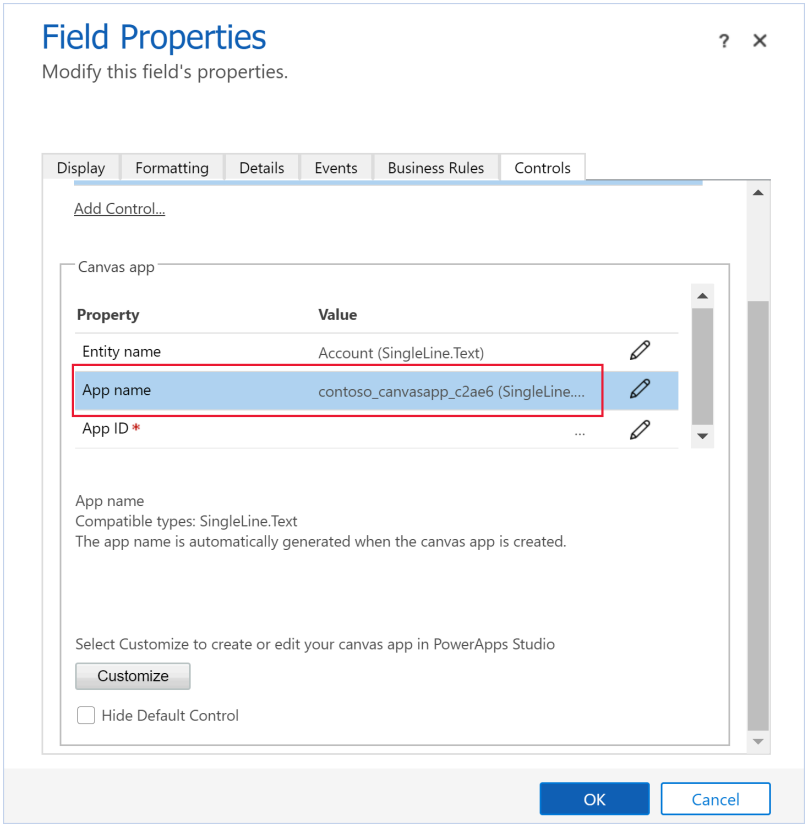
the data pane when you're done adding and reordering columns. 16. Select the **File** tab, and then select **Save**. 17. Select the **The cloud** tab. Provide a unique name for the app and then select **Save** located on the lower right. Note the following behavior:

- Saving an app for the first time automatically publishes the app.
- On subsequent saves, select **Publish** and then select **Publish this version** to make your changes available.

18. On the menu, select **Back**.
19. Select the browser tab that has the classic form designer open. Observe that the **App ID** and **App Name** properties of the canvas app control now have a value automatically filled in.

ⓘ Important

- The form designer has a direct link with Power Apps Studio that was opened in another browser tab in an earlier step.
- Make sure the App name has a value similar to the screenshot here. The App name property specifies the unique name that is used to reference the canvas app at runtime. If this name isn't set, the App ID property is used to reference the canvas app, which might cause issues when you import the canvas app as part of a solution. We recommend that you set this name to ensure proper canvas app referencing between environments. To set it, go back to step 12 and follow the steps to create the canvas app to embed.



20. On the **Column Properties** dialog box, select the **Display** tab.

21. Clear **Display label** on the form and then select **OK**. If you already have a canvas app embedded on this form, a message is displayed that "Only one canvas app can be enabled on a form." To add the new canvas app, you must first disable the current embedded canvas app. Then, enable the new embedded canvas app.

22. On the **Home** tab, select **Save**, and then select **Publish**.

See also

Edit a canvas app embedded on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Guidelines on working with embedded canvas apps

Last updated on 11/17/2025

Edit a canvas app embedded on a model-driven form

Summarize this article for me

This article explains how to edit a canvas app embedded on a model-driven form by using the classic designer.

Edit the canvas app directly

You can edit a canvas app embedded on a model-driven form just like any other canvas app. For steps on editing a canvas app go to: Edit a canvas app

Edit the canvas app via the host model-driven form

An alternate option is to edit the embedded canvas app via the host model-driven form.

Imagine that you want to edit a canvas app embedded on a form named Account Main Form for the Accounts table. To do this, follow these steps:

1. Sign in to Power Apps.
2. Edit the form named **Account Main Form** for the **Accounts** table.
3. In the command bar, select **Switch to classic** to open the form in the classic form designer.
4. In the classic form designer, select the column that is customized to display the embedded canvas app.
5. With the column selected, on the **Home** tab in the **Edit** group, select **Change Properties**.
6. On the **Column Properties** dialog box, select the **Controls** tab.
7. In the **Column Properties** dialog box, in the list of controls select **Canvas app**.
8. In the section below the controls list, select **Customize** to edit the canvas app. This opens the canvas app for editing, in Power Apps Studio, in a new tab.

> **Note**
>
> If opening Power Apps Studio is blocked due to a web browser pop-up blocker you must enable the make.powerapps.com site or temporarily disable the pop-up

blocker and then select **Customize** again.

9. When you finish making changes, select the **File** tab, and then select **Save**.
10. To make your changes available to users select **Publish** and then select **Publish this version**.

Related articles

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Guidelines on working with embedded canvas apps
Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

---

Last updated on 02/21/2026

Customize the screen size and orientation of a canvas app embedded on a model-driven form

Article • 05/22/2025

You can customize the screen size (tablet layout only) and orientation of a canvas app embedded on a model-driven form.

1. Follow the steps to add or edit an embedded canvas app on a model-driven form.

    - Add an embedded canvas app on a model-driven form
    - Edit a canvas app embedded on a model-driven form

2. With the canvas app open in Power Apps Studio, select the **File** tab, select **Settings**, and then select the **Screen size + orientation** tab to set the orientation screen and orientation of the canvas app. More information: Change screen size and orientation of a canvas app in Power Apps

See also

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Guidelines on working with embedded canvas apps
Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

Perform predefined actions on the host model-driven form from within an embedded canvas app

Summarize this article for me

Embedded canvas apps provide the ability to perform predefined actions on the host model-driven form. These actions enable makers to navigate, refresh and save the host model-driven form. Using these actions, an embedded canvas app can act as a more integral part of the model-driven form and the model-driven app.

The **ModelDrivenFormIntegration** object includes the following new methods to enable makers to perform actions on the host model-driven form.

> **Important**
>
> To be able to interact with the `ModelDrivenFormIntegration` control, the canvas app must be *created* by selecting **Customize** on the **Canvas app** control properties for the model-driven form using the classic form designer. More information: Embed a canvas app using the classic experience

NavigateToMainForm(entityName, mainFormName, recordId)

Navigates the host model-driven form to a main form and displays the specified row.

- **entityName** - A required string parameter that specifies the parent table of the main form.
- **formName** - A required string parameter that specifies the name of the main form to navigate to.
- **recordId** - A required string parameter, that specifies the ID of the row to display in the main form.

Calling the NavigateToMainForm method can return the following error messages.

Expand table

Error message	Troubleshooting guidance
Table not found: *[EntityName]*	Please check the value of the *entityName* parameter and ensure it is a valid table name and that the user has access to it.

Error message	Troubleshooting guidance
Form not found: *[FormName]*	Please check the value of the *mainFormName* parameter and ensure it is a valid main form name and that the user has access to it.
There was a problem loading the row.	Please check the value of the *recordId* parameter and ensure it is a valid row ID and that the user has access to it.

NavigateToView(entityName, viewName)

Navigates the host model-driven form to a view.

- **entityName** - A required string parameter that specifies the parent table of the view.
- **viewName** - A required string parameter that specifies the name of the main form to navigate to.

Calling the NavigateToView method can return the following error messages.

Expand table

Error message	Troubleshooting guidance
Table not found: *[EntityName]*	Please check the value of the *entityName* parameter and ensure it is a valid table name and that the user has access to it.
View not found: *[ViewName]*	Please check the value of the *viewName* parameter and ensure it is a valid view name and that the user has access to it.

OpenQuickCreateForm(entityName)

Opens the default quick create form for a table.

- **entityName** - A required string parameter that specifies the parent table of the quick create form.

Calling the OpenQuickCreateForm method can return the following error messages.

Expand table

Error message	Troubleshooting guidance
Table not found: *[EntityName]*	Please check the value of the *entityName* parameter and ensure it is a valid table name and that the user has access to it.

RefreshForm(showPrompt)

Refreshes the data on the host model-driven form.

- **showPrompt** - A required boolean parameter that indicates if a confirmation prompt should be displayed to the user before saving any unsaved data on the host model-driven form. Values should be "true" or "false".

Calling the RefreshForm method can return the following error messages.

Expand table

Error message	Troubleshooting guidance
Please use "true" or "false" as the parameter value.	Please check the value of the *showPrompt* parameter and ensure that it is either "true" or "false".

SaveForm()

Saves the data on the host model-driven form.

> **Note**
>
> If you don't observe the IntelliSense for the methods to perform predefined actions in embedded canvas apps that were created prior to the functionality being made available; save, close and re-open the app.

Related articles

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Edit a canvas app embedded on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-driven form
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Guidelines on working with embedded canvas apps
Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

Last updated on 02/21/2026

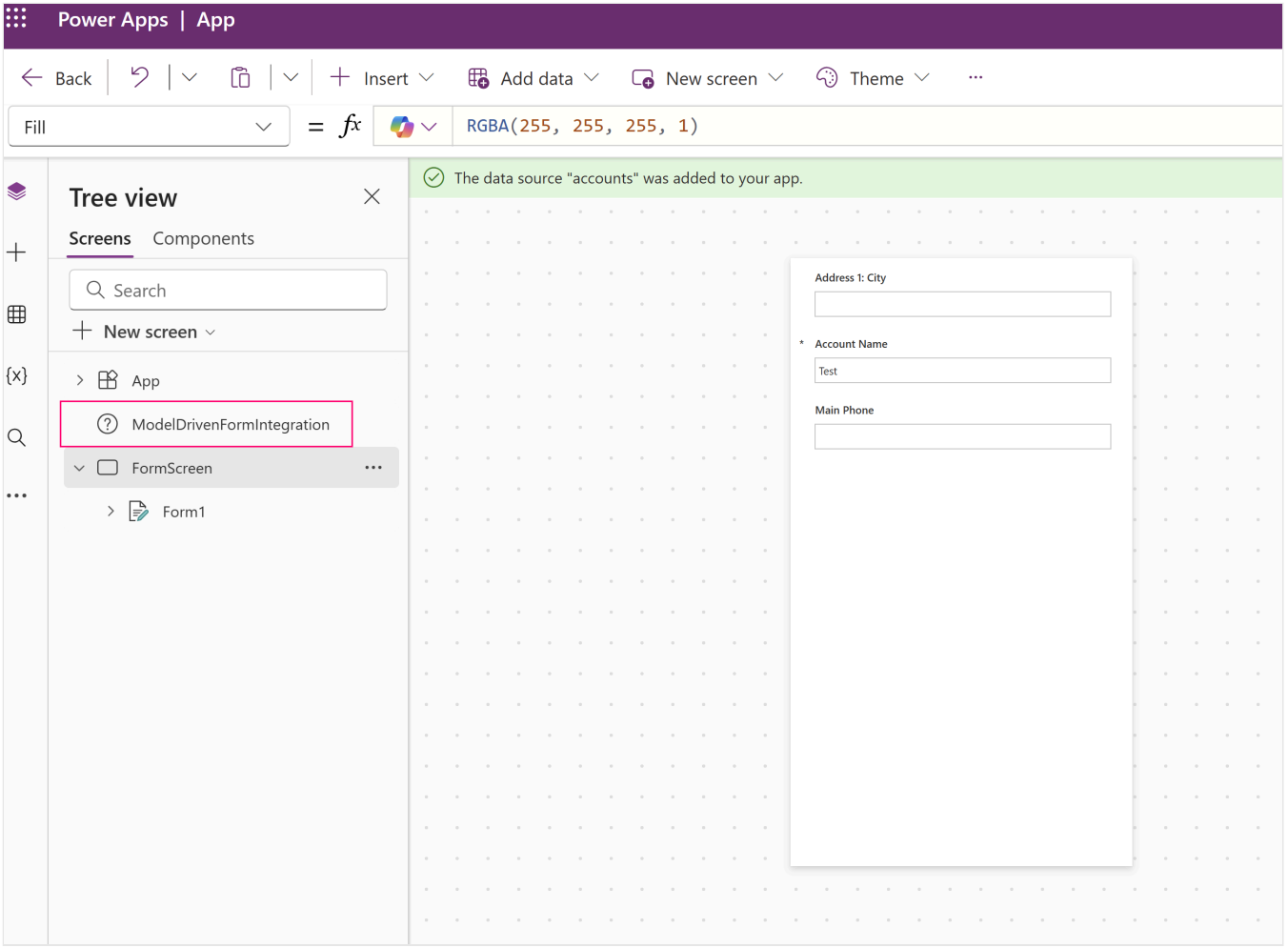
ModelDrivenFormIntegration control properties and actions

Canvas apps embedded on model-driven forms contain a special control named **ModelDrivenFormIntegration**. This control is responsible for bringing contextual data from the host model-driven form to the embedded canvas app.

> **Important**
>
> To be able to interact with the `ModelDrivenFormIntegration` control, the canvas app must be *created* by selecting **Customize** on the **Canvas app** control properties for the model-driven form using the classic form designer. More information: Embed a canvas app using the classic experience

This article explains the properties and actions available on the `ModelDrivenFormIntegration` control. This control is available to customize in Power Apps Studio when you create the canvas app using the Canvas app control properties UI in the classic form designer.

mod



Expand table

Property or action	Description
DataSource	Should be set to the data source connected to the parent table of the host model-driven form. Automatically set when embedding a new canvas app. When embedding an existing canvas app, ModelDrivenFormIntegration data source can be configured from the canvas app tree view. The value for DataSource should always refer to the table of the form the app is embedded in. The value shouldn't be filtered or modified in any way.
Item	Read-only property that enables the embedded canvas app to access the row from the host model-driven form. As an example, to get the value of a column with the name accountnumber and display name Account Number, you can use ModelDrivenFormIntegration.Item.accountnumber or ModelDrivenFormIntegration.Item.'Account Number'.
OnDataRefresh	The formula in this property is evaluated when the host model-driven form saves data. Use it to refresh the data source connected to the parent table of the host model-driven form and to perform other actions like setting or updating variables. As an example, setting it to the formula below will refresh the Accounts data source and set a variable named CurrentAccountNumber to the value of the Account Number column of the current row. Refresh(Accounts); Set(CurrentAccountNumber, ModelDrivenFormIntegration.Item.'Account Number') The OnDataRefresh action should only be used to refresh data within the embedded canvas app. We don't recommend that you use the OnDataRefresh action to save changes within the embedded canvas app. More information: Saving data in an embedded canvas app
RefreshForm	Refreshes the data on the host model-driven form. See Perform predefined actions on the host form for details.
SaveForm	Saves the data on the host model-driven form. See Perform predefined actions on the host form for details.
NavigateToMainForm	Navigates the host model-driven form to a main form and displays the specified row. See Perform predefined actions on the host form for details.
NavigateToView	Navigates the host model-driven form to a view. See Perform predefined actions on the host form for details.
OpenQuickCreateForm	Opens the default quick create form for an table. See Perform predefined actions on the host form for details.

Property or action	Description
Data	Read-only property used by the framework to send some key data from the host model-driven form to the embedded canvas app. Do not use this property. Use Item to access the row from the host model-driven form.

ModelDrivenFormIntegration control data is not available on app start

The `ModelDrivenFormIntegration.Item` property isn't guaranteed to be populated with data when the app starts. The app should either be designed to wait for this data using an expression like `!IsBlank(ModelDrivenFormIntegration.Item)`, or the app should explicitly retrieve the record using an expression like `LookUp(Accounts, Account = GUID(First(ModelDrivenFormIntegration.Data).ItemId))`.

See also

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
Share an embedded canvas app
Guidelines on working with embedded canvas apps

Last updated on 11/18/2025

Pass the current record as data context to an embedded canvas app

Summarize this article for me

> **Important**
>
> Canvas apps embedded on model-driven forms are generally available. The steps listed below are outdated and applicable only to the public preview release of canvas apps embedded on model-driven forms. For the updated list of steps for the latest release, go to: Add an embedded canvas app on a model-driven form

This article explains how to add an embedded canvas app and pass the current (main form) record as a data context to the embedded canvas app.

Imagine that you want to add an embedded canvas app on an account main form and pass the current account record to the embedded canvas app. To do this, follow these steps:

1. Sign in to Power Apps and open the form editor for a main form of a table, such as the account table.
2. Select the section on the form where you want the embedded canvas app to appear.
3. Using the column explorer pane, add a required column, such as **Account Name**.

   > **Important**
   >
   > Always use a required column that is guaranteed to have a value. If your column does not have a value, your embedded canvas app will not refresh in response to any change in data on the host model-driven form.

4. With the column selected, on the **Home** tab in the **Edit** group, select **Change Properties**.
5. On the **Column Properties** dialog box, select the **Controls** tab.
6. On the **Controls** tab, select **Add Control...**.
7. On the **Add Control** dialog box, in the list of available controls, select **Canvas app** and then select **Add**.
8. In the **Column Properties** dialog box, in the list of controls select **Canvas app**, and then select the **Web** option.
9. In the section below the controls list, the list of properties available to the canvas app control are displayed.

   - The **Table name** property specifies the table that will provide the data to your embedded canvas app. It will be set to the table that contains the column you

added in an earlier step.
    - Notice that, even though this property appears changeable, changing it has no effect on the embedded canvas app. It is meant only to serve as a reference for you.
- The **App ID** property specifies the ID of the embedded canvas app. It will be automatically generated and filled-in for you when the canvas app is created.
    - Notice that any change to the **App ID** value breaks the link from the model-driven form to the embedded canvas app.

10. Select **Customize** to create or edit the canvas app. This opens Power Apps Studio in a new tab.

    > ⓘ **Note**
    >
    > - The **Customize** option is currently only available by using the classic form designer.
    > - If opening Power Apps Studio is blocked due to a web browser pop-up blocker you must enable the make.powerapps.com site or temporarily disable the pop-up blocker and then select **Customize** again.

11. In Power Apps Studio notice that there is a special **ModelDrivenFormIntegration** control in the left pane. This control is responsible for bringing contextual data from the host model-driven form to the embedded canvas app.
12. Select the **Gallery1** control and observe that the **Items** property is set to **ModelDrivenFormIntegration.Data**.

    > ⓘ **Note**
    >
    > ModelDrivenFormIntegration.Data is a list of records. In this example it has only one record. To directly reference the record you can use the First function. For example, *First(ModelDrivenFormIntegration.Data).Name*.

13. In the property pane on the right, next to **Columns**, select **Edit**.
14. In the data pane, change the column mapped to the **Title1** control to **Name** or another column that has data.
15. Observe that the gallery displays the data being passed to it from the host model-driven form via the ModelDrivenFormIntegration control. Close the data pane.
16. Select the **File** tab, and then select **Settings**.
17. On the **Upcoming features** tab, in the **Experimental features** section, set **Optimize embedding appearance** to **On**.
18. Select the **General** tab. Provide a unique name for the app and then select **Save**. Note the following:

- Saving an app for the first time automatically publishes the app.
- On subsequent saves, select **Publish** and then select **Publish this version** to make your changes available.

19. Close the **Settings** dialog.
20. On the menu, select **Back** and then select the browser tab that has the form editor open. Observe that the **App ID** property of the canvas app control now has a value automatically filled in. Note the following:

    - The form editor has a direct link with Power Apps Studio that was opened in another browser tab in an earlier step.
    - The form editor "listens" for the **App ID** to be sent to it.
    - The **App ID** is sent to the form editor when the app is saved.

21. On the **Column Properties** dialog box, select the **Display** tab.
22. Clear **Display label** on the form and then select **OK**.

    - If you already have a canvas app embedded on this form, a message is displayed that “Only one canvas app can be enabled on a form.” To add the new canvas app you must first disable the current embedded canvas app. Then, enable the new embedded canvas app.

23. On the **Home** tab, select **Save**, and then select **Publish**.

After you have added an embedded canvas app to your model-driven form, share your embedded canvas app with other users. More information: Share an embedded canvas app.

When users open a model-driven app (Unified Interface only) that includes the form you have modified, they see the embedded canvas app on the form. Changing the row displayed on the main form changes the data context that is passed to the form, and the embedded app refreshes to show the relevant data.

This topic showed you how to get started with embedding a canvas app in a model-driven form. You can further customize the embedded canvas app to connect and bring in data from a variety of data sources. Use the Filter, Search, and LookUp functions and the context passed in from the host model-driven form to filter or find specific records in those data sources. Use the WYSIWYG canvas app editor to easily design the interface to match your requirements.

See also

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Edit a canvas app embedded on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-driven form

Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Guidelines on working with embedded canvas apps
Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

---

Last updated on 02/20/2026

Pass a list of related records as data context to an embedded canvas app

Article • 02/15/2022

> **Important**
>
> Canvas apps embedded on model-driven forms are now out of preview and generally available. The steps listed below are outdated and applicable only to the public preview release of canvas apps embedded on model-driven forms. For the updated list of steps for the latest release, go to: **Add an embedded canvas app on a model-driven form**

Overview

This article explains how to add an embedded canvas app and pass a list of records related to the current (main form) record as a data context to the embedded canvas app.

Procedure

Let's say you want to add an embedded canvas app on an account main form and pass a list of contacts related to the current account record to the embedded canvas app. To do this, follow these steps:

1. Sign in to Power Apps and open the form editor for a main form of a table, such as the account table.
2. Select the section on the form where you want the embedded canvas app to appear.
3. With the section selected, on the **Insert** tab, in the **Control** group, select **Sub-Grid**.
4. In the **Set Properties** dialog box, select the **Display** tab, and then in the **Name** box enter a name for the grid control.
5. In the **Data Source** section, select an **Table** and **Default View** that corresponds to the list of records that you want to pass as data context to the embedded canvas app.

6. Select the **Controls** tab, and then select **Add Control...**

7. In the **Add Control** dialog box, in the list of available controls, select **Canvas app** and then select **Add**.

8. In the **Set Properties** dialog box, in the list of controls select **Canvas app** and then select the **Web** option.

9. In the section below the controls list, see the list of properties corresponding to the Canvas app control and note the following:

    - The **Table name** property specifies the table that will provide the data to your embedded canvas app. It will be set to the table that you selected earlier.
        - Even though this property appears changeable, changing it has no effect on the embedded canvas app. It is meant only to serve as a reference for you.
    - The **View name** property specifies the view of the table that will be used to filter the data provided to your embedded canvas app. It will be set to the **Default View** you selected earlier.
        - The data (fields and values) sent to the embedded canvas app at runtime are determined by this view. Only use columns in your canvas app that are included in the view or add them to the view if needed. Any columns that are not included in the view display as empty values at runtime.
        - The filter criteria for a view are not used at authoring time. Therefore, the data that you see when authoring embedded canvas apps is not filtered, it is simply a list of top few records that you have access to. At runtime, the filter criteria for the view are applied as expected so users only see relevant data.
    - The **App ID** property specifies the ID of the embedded canvas app. It is automatically generated and filled-in for you when the canvas app is created.
        - Notice that any change to the App ID value breaks the link from the model-driven form to the embedded canvas app.

10. Select the **Customize** button to create or edit the canvas app. This opens Power Apps Studio in a new browser tab.

    > **ⓘ Important**
    >
    > - The **Customize** option is currently only available by using the classic form designer.
    > - If opening Power Apps Studio is blocked due to a web browser pop-up blocker, you must enable the make.powerapps.com site or temporarily

disable the pop-up blocker and then select **Customize** again.

11. In Power Apps Studio, notice that there is a **ModelDrivenFormIntegration** control located in the left pane. This control is responsible for bringing contextual data from the host model-driven form to the embedded canvas app.

12. Select the **Gallery1** control and observe that the **Items** property is set to **ModelDrivenFormIntegration.Data**.

13. In the property pane on the right, next to **Columns**, select **Edit**.

14. In the data pane, change the column mapped to the **Title1** control to **FullName** or another column that has data.

15. Observe that the gallery displays the data being passed to it from the host model-driven form via the **ModelDrivenFormIntegration** control. Close the data pane.

16. Select the **File** tab, and select **Settings**.

17. On the **Upcoming features** tab, in the **Experimental features** section, set **Optimize embedding appearance** to **On**.

18. Select the **General** tab. Provide a unique name for the app and then select **Save**. Note the following:

    - Saving an app for the first time automatically publishes the app.
    - On subsequent saves, select **Publish** and then select **Publish this version** to make your changes available.

19. Close the **Settings** dialog.

20. Select **Back**, and then select the browser tab that has the form editor open.

21. Observe that the **App ID** property of the **Canvas app** control now has a value automatically filled in. Note the following:

    - The form editor has a direct link with Power Apps Studio that was opened in another browser tab in an earlier step.
    - The form editor has been listening for the App ID to be sent to it.
    - The App ID was sent to it when the app was saved.

22. In the **Set Properties** dialog box, select the **Display** tab, clear **Display label on the form**, and then select **OK**.

- If you already have a canvas app embedded on this form, a message is displayed that “Only one canvas app can be enabled on a form.” To add the new canvas app, you must first disable the current embedded canvas app. Then, enable the new embedded canvas app.

23. On the **Home** tab, select **Save**, and then select **Publish**.

After you have added an embedded canvas app to your model-driven form, share your embedded canvas app with other users. More information: Share an embedded canvas app.

When users open a model-driven app (Unified Interface only) that includes the form you have modified, they see the embedded canvas app on the form. Changing the record displayed on the main form changes the data context that is passed to the form and the embedded app refreshes to show the relevant data.

This article showed you how to get started with the embedding a canvas app in a model-driven form. You can further customize the embedded canvas app to connect and bring in data from various data sources. Use the Filter, Search, and LookUp functions and the context passed in from the host model-driven form to filter or find specific records in those data sources. Use the WYSIWYG canvas app editor to easily design the interface to match your requirements.

See also

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Edit a canvas app embedded on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Guidelines on working with embedded canvas apps
Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

Share an embedded canvas app

Article • 10/14/2022

This article explains what's needed to share an embedded canvas app that you have already created.

After you have created and added an embedded canvas app to a model-driven form you will need to take steps to ensure that all users that have access to the model-driven form also have access to the canvas app and the data that it uses. Please refer to the following guidelines:

- Sharing an embedded canvas app with Everyone in your organization is not supported with Microsoft Dataverse. Doing so might cause issues when rendering the app. Instead, share your embedded canvas app with a security group or specific users. More information: Share an app
- Ensure that users have appropriate permissions for any Microsoft Dataverse tables that your embedded canvas app uses. Specifically add read privilege for the **Canvas App** table located on the **Customization** tab of the security role editor. More information: Manage table permissions
- Ensure that users have appropriate permission for data on any cloud services that your embedded canvas app uses, such as SharePoint or OneDrive. The steps to share are specific to each cloud service and beyond the scope of Power Apps.

> ⓘ **Note**
>
> Currently, you must explicitly share canvas apps through app sharing. The **Canvas App** privilege in a security role does not share the embedded or standalone application.

Embedded canvas apps are also solution aware. By default embedded canvas apps are created in the same solution as the host model-driven form. To move the embedded canvas app from one environment to another export and import embedded canvas apps as a part of a solution just like any other component.

See also

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Edit a canvas app embedded on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-

driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Guidelines on working with embedded canvas apps
Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Guidelines and troubleshooting for embedded canvas apps

Article • 01/10/2024

Guidance on embedding canvas apps

This article provides guidance on working with embedded canvas apps as well as helpful tips to troubleshoot any issues faced.

- Embedded canvas apps are only supported with Unified Interface model-driven apps.
- Only three embedded canvas apps can be enabled for each form with Web, and one for Tablet and Phone client types.
- Multiple embedded canvas apps can be added to the form, but can only enable three at a time for Web and one at a time for Tablet and Phone client types.
- If more than three embedded canvas apps are enabled with the Web client type on a model-driven app form the error message shows as follows "You have more than three canvas apps with Web form factor, the maximum is three for this form factor. The number of canvas apps are limited to three for Web and one for Tablet and Phone form factors."
- To enable or disable an embedded canvas app see Enable an embedded canvas app and Disable an embedded canvas app.
- We recommend that you have a single embedded canvas app for each form tab.
- When adding an embedded canvas app to a model-driven form always use a required column that is guaranteed to have a value. If your column doesn't have a value your embedded canvas app won't refresh in response to any change in data on the host model-driven form.
- Publishing a model-driven form doesn't also publish the embedded canvas app.
  - Embedded canvas apps must be published independent of the host model-driven form. More information: Publish an app.
- If opening Power Apps Studio to create or edit an embedded canvas app via the **Customize** button in the canvas app control properties is blocked due to a web browser pop-up blocker, you must enable the make.powerapps.com site or temporarily disable the pop-up blocker and then select **Customize** again.
- Embedded canvas apps aren't displayed when creating a new row since they need a row context to be passed to them.
- The ModelDrivenFormIntegration.Item object is read-only.

    - To write back data, you must use the Dataverse connector. More information: Microsoft Dataverse
- Embedded canvas apps can only be created via the host model-driven form.
- When viewing a model-driven form with an embedded canvas app, if an error message reads "It looks like you don't have access to this app" ask its owner to share it with you" make sure that the author has shared the embedded canvas app with you. More information: Share an embedded canvas app.
- Adding a canvas app on the subgrid control is no longer available.
    - In the preview release, makers were able to add a canvas app on a subgrid control. With canvas app embedding on model-driven forms now generally available, adding an embedded canvas app on a model-driven form is streamlined to the column.
    - This makes it easier for makers since they don't have to decide up front whether to pass the current (main form) row as data context or a list of rows related to the current (main form) row.
    - Makers always start with a column and can access both the current (main form) row or a list of rows related to the current (main form) row.
    - To access the list of related rows in the canvas app, makers can use the Dataverse connector and Filter function with the Improve data sources experience and Dataverse views capability enabled in the canvas app. For example, to access the *Active Contacts* view of the *Contacts* table, makers can use: *Filter(Contacts, 'Contacts (Views)'.'Active Contacts')*.
    - Existing canvas apps that use the subgrid control will continue to work. However, we recommend that you migrate these apps to use a column instead. More information: Migrating embedded canvas apps on model-driven forms that use a list of rows related to the current (main form) row for details.

Enable an embedded canvas app

1. Select the column that is customized to display as an embedded canvas app.
2. In the **Column Properties** dialog, select the **Controls** tab.
3. In the list of controls select **Canvas app** and then select the **Web** option.
4. Select **OK**.

Disable an embedded canvas app

1. Select the Column that is customized to display as an embedded canvas app.
2. In the **Column Properties** dialog, select the **Controls** tab.
3. In the list of controls, select the default control and then select the **Web** option.
4. Select **OK**.

Saving data in an embedded canvas app

- A save event made from a model-driven app, such as selecting the Save button on the main form command bar, doesn't save changes made in the embedded canvas app.
- To save changes made in an embedded canvas app, use the Dataverse connector.
- The ModelDrivenFormIntegration control OnDataRefresh action should only be used to refresh data within the embedded canvas app. We don't recommend that the OnDataRefresh action is used to save changes within the embedded canvas app.

Known issues and limitations with embedded canvas apps

Limitations

- The canvas app custom control is only supported for use with the **Web** client type. Currently, the **Phone** and **Tablet** client types aren't supported.
- The **Canvas App** privilege in a security role can't be used to grant app users access to either an embedded or standalone canvas app. For more information on sharing an embedded canvas app, go to: Share an embedded canvas app.
- If you write back the same data that is being displayed in the host model-driven form, the form will continue to display old data until it's refreshed. An easy way to do that is to use the RefreshForm method.
- Offline and device capability controls like barcode scanning, capturing photos from device, or attaching files aren't supported in embedded canvas apps.

The ModelDrivenFormIntegration control doesn't provide a value for columns of a related table

For example, when the ModelDrivenFormIntegration control is connected to the Accounts table, using *ModelDrivenFormIntegration.Item.'Primary Contact'.'Full Name'* won't return a value.

To access columns of a related table, makers can use either of the expressions listed here: - *LookUp(Accounts, Account = GUID(First(ModelDrivenFormIntegration.Data).ItemId)).'Primary Contact'.'Full Name'* - *ItemId* is empty at authoring time but will have a value at runtime. - *LookUp(Accounts,*

Account = ModelDrivenFormIntegration.Item.Account).'Primary Contact'.'Full Name' (This expression is easier to read, but the previous expression will perform slightly better.)

Embedded canvas app doesn’t render correctly

You can build your canvas app to be responsive, which refers to the ability of an app to automatically align to different screen sizes and form factors to use the available screen space sensibly. Depending on whether your app is built to be responsive or not, we recommend different settings to ensure that your canvas app might render correctly within the field on the model-driven app form. If you're experiencing additional whitespace or scroll bars around your canvas app, we recommend checking the following in your app.

For responsive apps:

- Disable the **Scale to fit** option. This allows your app to scale according to the dimensions and properties you have set for your app. An embedded canvas app might not render correctly within the field on the model-driven app form, such as additional whitespace or scroll bars around the canvas app.

For nonresponsive apps:

- We recommend that you enable the **Scale to fit** option. This helps the app to resize to fit the available space.

In both scenarios, ensure that the **App Name** property value is set and correctly defined in the embedded canvas app control.

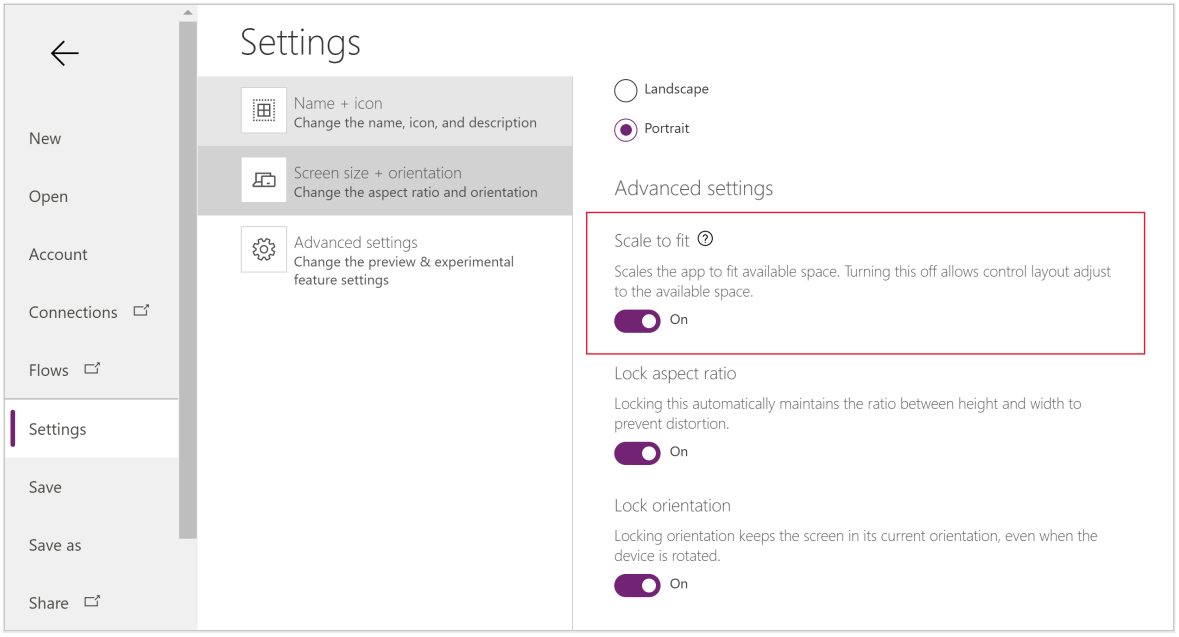
Embedded canvas app doesn't respect height

The **Form field height** property isn't respected by the canvas app component. When embedded, the dimensions of the canvas app component respect the aspect ratio set on the canvas app. Because the aspect ratio is fixed, the height of the canvas app component is calculated relative to the width of the app. The width of the app is determined based on the horizontal space available in the model-driven app. If you would like to make adjustments to the height of the canvas app component, we recommend customizing the aspect ratio on your canvas app. Note that to customize the aspect ratio, the app must be in tablet layout.

Enable scale to fit

By default, canvas apps have the scale to fit option enabled.

1. Open the canvas app that you're embedding on a model-driven app form for editing.
2. In Power Apps studio, select **File** > **Settings** > **Screen size + orientation**.
3. Under **Advanced Settings**, set **Scale to fit** to **On**.



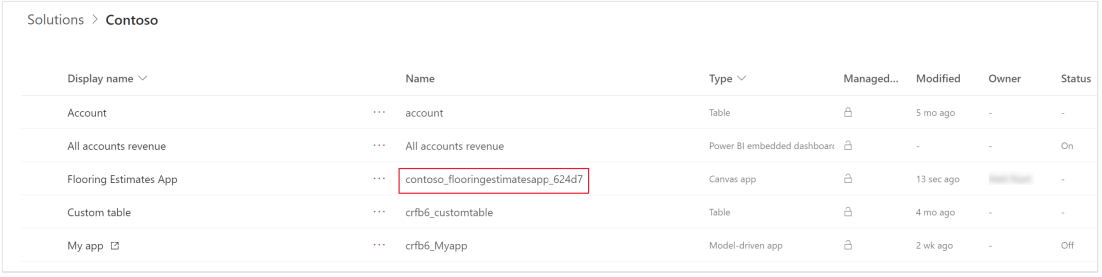
The App Name property value is missing or is incorrectly defined

To resolve this issue, choose the correct option:

- Managed solutions: If the solution was imported into an environment as a managed solution, follow these steps:

    1. Sign into Power Apps and go to the development environment where you originally created your embedded canvas app. This is also the environment where the solution was exported.

    2. Open the unmanaged solution that includes the canvas app, and then find the canvas app in the list of solution components. Copy the canvas app **Name** exactly as it appears in the components list. For example, *contoso_flooringestimatesapp_624d7*.



    3. In the same solution, edit the model-driven app that has the embedded canvas app control, and then set the canvas **App Name** to the embedded

canvas app control using the value from the previous step. More information: Add an embedded canvas app on a model-driven form

4. Export the solution from the development environment and then import the solution into your target environment.

- Unmanaged solutions: If the solution was imported into an environment as an unmanaged solution, edit the model-driven app that has the embedded canvas app control, and then follow similar steps as described for a managed solution to set the canvas **App Name** property.

Embedded canvas app control loads the canvas app from a different environment

When a solution that contains a model-driven app with an embedded canvas app is imported into a target environment, the control loads the canvas app from the source environment (where the solution was exported). The control should load the canvas app from the target (current) environment.

This issue occurs because the app user doesn't have read access to the CanvasApp Extended Metadata table in the target (current) environment. To resolve this issue, add the user to a security role used for the app that has read access to the CanvasApp Extended Metadata table. More information: Grant read privileges for the CanvasApp Extended Metadata table

Error message: "You don’t have read privileges for the Canvas App entity. Please contact your administrator"

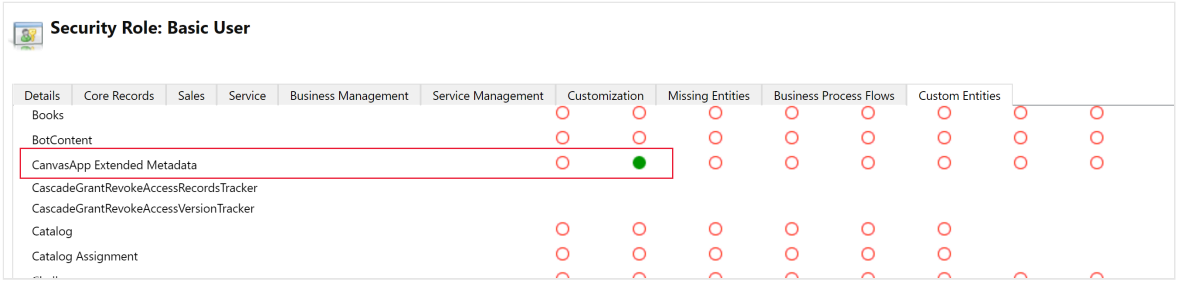
The error message is displayed on the model-driven app form where the embedded canvas app should appear.

This issue occurs because the app user doesn't have read access to the CanvasApp Extended Metadata table. To resolve this issue, add the user to a security role used for the app that has read access to the CanvasApp Extended Metadata table.

Grant read privileges for the CanvasApp Extended Metadata table

1. In Power Apps, select the environment, and then on the left navigation pane select **Apps**.
2. Select the app you want, select **...**, and then select **Share**.
3. On the left pane, select the app, and then select **Manage security roles**.
4. Open the security role assigned to the app, such as the Basic User security role.

5. Select the **Custom Entities** tab, and set organization scope read privileges for the **CanvasApp Extended Metadata** table.



6. Select **Save and Close** to close the security role window.

See also

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Edit a canvas app embedded on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Migrating embedded canvas apps on model-driven forms created using the public preview release to latest

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Migrate embedded canvas apps on model-driven forms

Summarize this article for me

> **Important**
>
> With the latest release, embedded canvas apps on model-driven forms are generally available. Any embedded canvas apps on model-driven forms created using the public preview release should be migrated to new embedded canvas apps created using the latest release. Support for embedded canvas apps on model-driven forms created using the public preview release will soon be deprecated.

To migrate an embedded canvas app on a model-driven form created using the public preview release to the latest, makers first need to create a new embedded canvas app using the latest release. Makers can then copy over the controls from the existing embedded canvas app to the new one, add required data sources and update broken references, if any. Detailed steps are provided below.

1. Sign in to Power Apps.
2. Open the embedded canvas app created using the public preview release for editing in Power Apps Studio. For steps on editing a canvas app, go to: Edit a canvas app.
3. In a new browser tab, follow the steps to add a new embedded canvas app on a model-driven form.
4. Copy the controls from the embedded canvas app created using the public preview release to the new embedded canvas app, one screen at a time using these steps:
   a. Select the browser tab from Step 2, that has the embedded canvas app, created using the public preview release, open in Power Apps Studio.
   b. Select a screen to copy controls from.
   c. Use **Ctrl + A** to select all controls on the screen.
   d. Use **Ctrl + C** to copy all selected controls.
   e. Select the browser tab from Step 3, that has the new embedded canvas app created using the latest release.
   f. Select a screen or add a new one.
   g. Use **Ctrl + V** to paste the controls on the selected screen.
   h. Repeat the steps to copy each screen.
5. When you finish copying all the screens, select the browser tab from Step 3, that has the new embedded canvas app created using the latest release.
6. Update all the places where the row from the host model-driven form is being accessed. Replace **First(ModelDrivenFormIntegration.Data)** with

ModelDrivenFormIntegration.Item.

7. Add any missing data sources in the new embedded canvas app.
8. Update all broken references in the new embedded canvas app.
9. When you finish making changes, select the **File** tab, and then select **Save**.
10. To make your changes available to users, select **Publish** and then select **Publish this version**.

Migrating embedded canvas apps on model-driven forms that use a list of rows related to the current (main form) row

In the preview release, to embed a canvas app on a model-driven form, makers had to decide up front if they wanted to pass the current (main form) row as data context or a list of rows related to the current (main form) row. They then had to add the canvas app control to either the column or sub-grid control.

With the latest release, adding an embedded canvas app on a model-driven form is simplified and streamlined to the column only. Makers can still easily access the list of related rows directly in the canvas app using the Common Data Service connector.

To migrate an embedded canvas app on a model-driven form that uses a list of rows related to the current (main form) row, follow these steps.

1. Follow the steps in the section above to migrate embedded canvas apps on model-driven forms created using the public preview release to the latest.
2. Using the Common Data Service connector, add a data source for the related table to the app. To learn how add a data source in a canvas app refer to Add a data connection to a canvas app in Power Apps.
3. When using the data source of the related table for a control such as Gallery or Data table, use the **Filter** function to filter the rows to the ones that are related to the current (main form) row. The current (main form) row is available via **ModelDrivenFormIntegration.Item**.

   > ⓘ **Note**
   >
   > The embedded canvas app has full access to row from the host model-driven form via `ModelDrivenFormIntegration.Item`. As an example, to get the value of a column with the name **accountnumber** and display name **Account Number**, you can use **ModelDrivenFormIntegration.Item.accountnumber** or **ModelDrivenFormIntegration.Item.'Account Number'**.

4. With recent updates Microsoft Dataverse now also provides support to use table views as a filter. Go to this blog post for details: Improved data source selection and Dataverse views.

Related articles

Embed a canvas app on a model-driven form
Add an embedded canvas app on a model-driven form
Edit a canvas app embedded on a model-driven form
Customize the screen size and orientation of a canvas app embedded on a model-driven form
Perform predefined actions on the host form from within an embedded canvas app
ModelDrivenFormIntegration control's properties and actions
Share an embedded canvas app
Guidelines on working with embedded canvas apps

Last updated on 02/21/2026

List of controls available for model-driven apps

Controls allow app users to visualize data within table columns in addition to enabling interaction with them.

Most controls are configured for a table column, but some controls, like the grid control, can be configured for the table.

Using controls in the form designer

To use these controls in the form designer:

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane. If the item isn't in the side panel pane, select ... More and then select the item you want.
3. Select the required table and then select the **Forms** area.
4. Select the main form to be edited.
5. Select the column or section where you want to add the control.
6. In the right properties pane, select **+ Component** to display the available controls, and then select the control you want.

   > **Note**
   >
   > Different controls are available depending on the column or grid type. For example, star rating controls are only available for whole number columns.

7. Select the devices (web, tablet, and phone) you want the control to appear on.
8. Configure the values for each property.
9. Select **Done** when you're done configuring the control.

The following descriptions explain each control you can use on forms.

Grid controls

Grid controls can be configured at the following levels:

- Table. When you configure a grid control for a table, all views for the table use the grid control. For more information, see Configure a grid control for a table.
- Form. Subgrids added to a form use a grid control to control the behavior of the subgrid.

Subgrid

The read-only subgrid allows you to present a view of data related to the current record where users can view and open the record from within the grid. More information: Add and configure a subgrid component on a form

Editable grid

With editable grids, users can do rich in-line editing of records directly from views and subgrids whether they're using a web app, tablet, or phone. More information: Make model-driven app grids (lists) editable using the editable grid control

Power Apps grid control

The Power Apps grid control represents the next evolution of the Power Apps grid control, allowing users to view, open, and edit records from views and subgrids. By default, the Power Apps grid control is read-only but you can configure it to be editable.

More information: Power Apps grid control

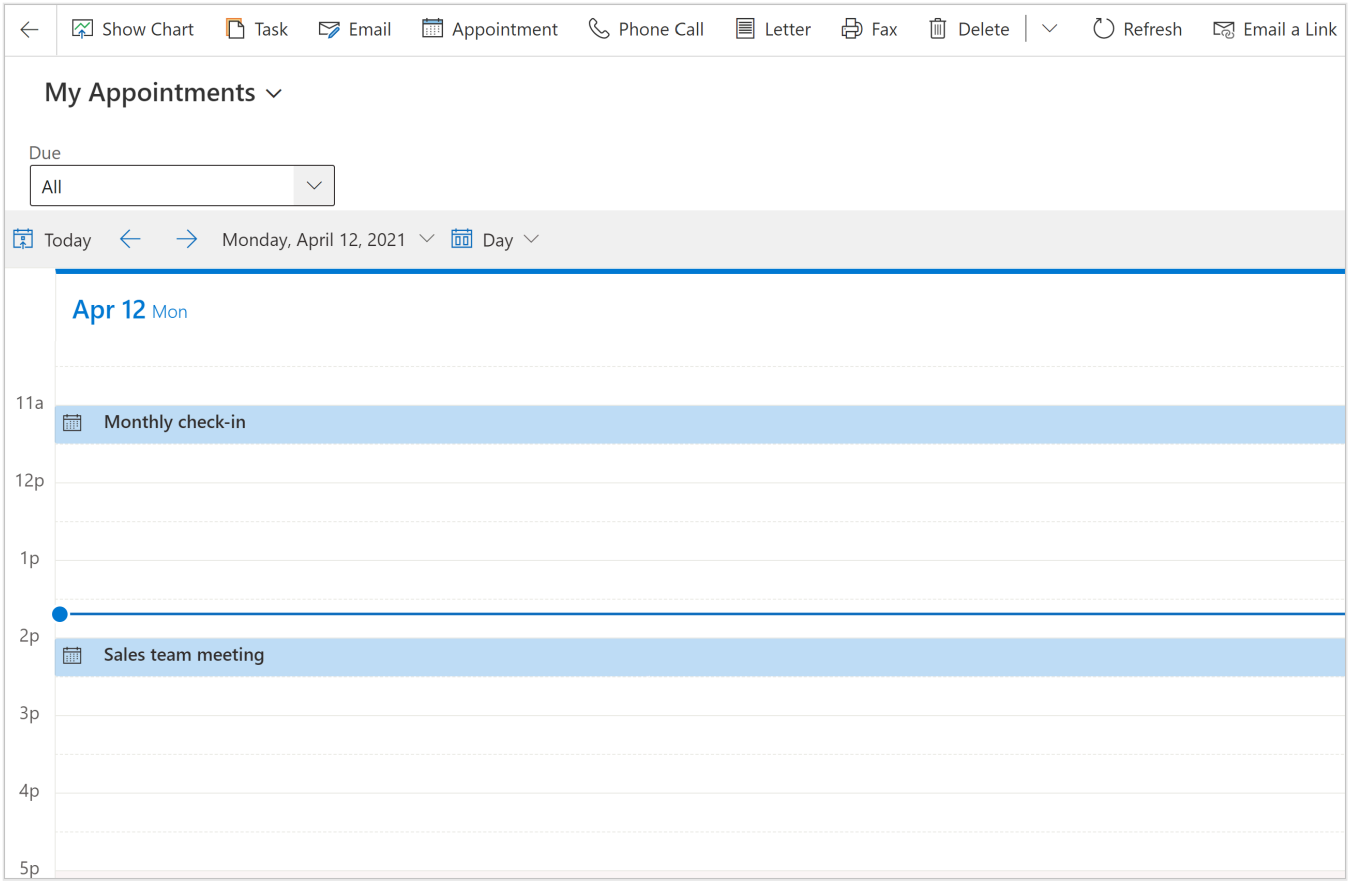
Configure a grid control for a table

1. Sign in to Power Apps.
2. Open solution explorer, expand **Entities**, and then select the table you want, such as the account table.
3. Select the **Controls** tab, and then select the grid you want for the table.
4. Depending on the grid control, specify the values you want, **Save** and then **Publish** the customization.

Display controls

Calendar

The Calendar control control displays scheduled activities and their associated details in a calendar. You can view, create, and delete your activities in a day, week, or month view. More information: Add the calendar control to tables



Canvas app

An embedded canvas app includes rich data integration capabilities that bring in contextual data from the host model-driven form to the embedded canvas app. Display the data you want from a variety of sources right next to data from Microsoft Dataverse. More information: Add an embedded canvas app on a model-driven form

External website

Add inline frames (iframes) to a form to integrate content from another website within the form. More information: Add an iframe to a model-driven app main form

HTML and image web resource

Display an HTML page or image file on a form via a web resource. More information: View and edit a web resource on a form

Knowledge search

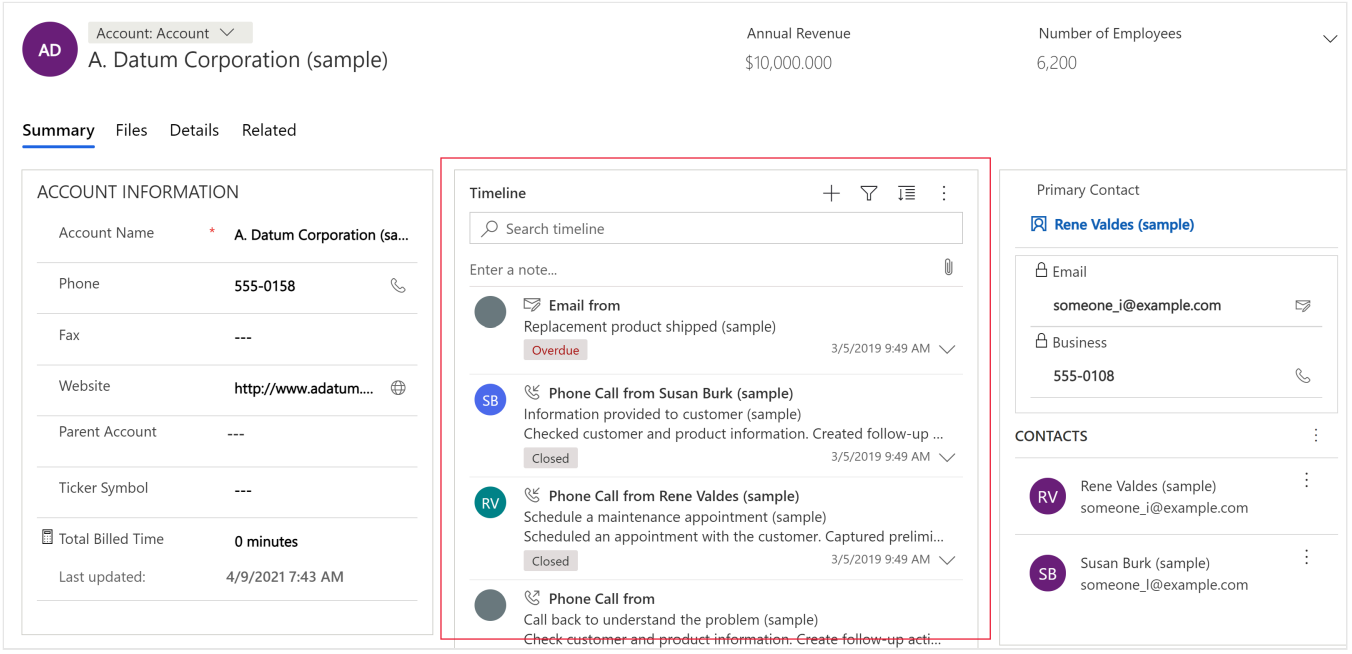
Requires Dynamics 365 Customer Service app. Add a Knowledge Base Search control to a form to make it easy for users in your organization to find knowledge articles so they can answer common customer questions and resolve their issues right from the records, without having to switch to a different application. More information: Add the Knowledge Base Search control to forms

Quick view

The quick view control displays data from a row that you select in a lookup on the form. You define the data displayed in the control by using a quick view form. The data displayed isn't editable, but when the primary column is included in the quick view form, it becomes a link to open the related row. For more information, see Model-driven app quick view control properties.

Timeline control

Provide a timeline of recent, relevant news articles and Twitter tweets for an account. More information: Set up the timeline control



Expand table

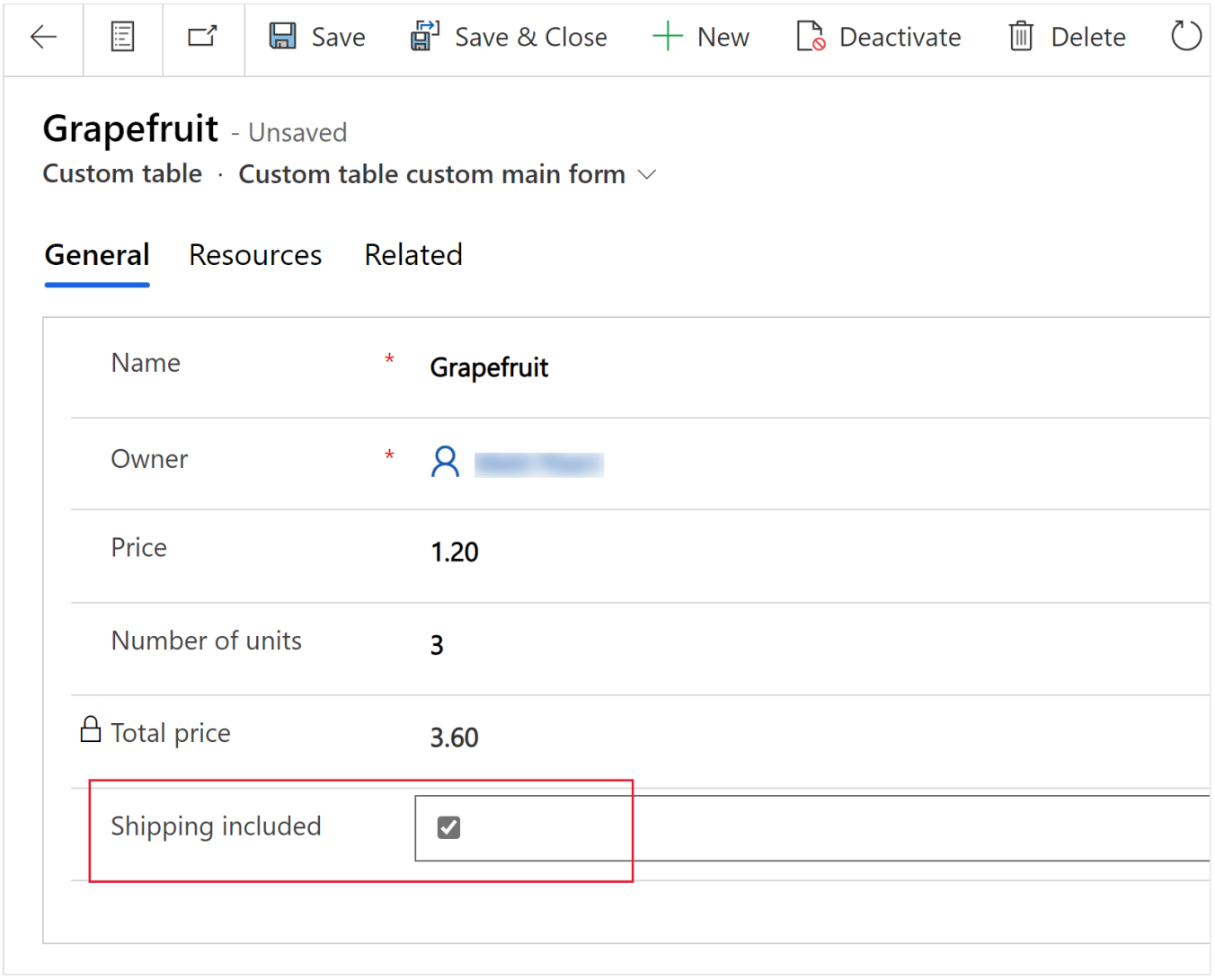
Property	Description
CC_Timeline_Title	Property to map for the title of each timeline item.
CC_Timeline_Title_Desc	Description for Title.
CC_Timeline_Label1	Column to be displayed below the title of timeline item.

Property	Description
CC_Timeline_Label1_Desc	Description for Label 1.
CC_Timeline_Label2	Column to be displayed after Label 1.
CC_Timeline_Label2_Desc	Description for Label 2.
CC_Timeline_Label3	Column to be displayed after Label 2.
CC_Timeline_Label3_Desc	Description for Label 3.
CC_Timeline_Label4	Column to be displayed after Label 3.
CC_Timeline_Label4_Desc	Description for Label 4.
CC_Timeline_Label5	Column to be displayed after Label 4.
CC_Timeline_Label5_Desc	Description for Label 5.
CC_Timeline_Timestamp	Column to use for sorting timeline in reverse chronological order.
CC_Timeline_Timestamp_Desc	Description for Timestamp.
CC_Timeline_Group	Column to map for grouping timeline.
CC_Timeline_Group_Desc	Description for Group column.
CC_Timeline_GroupOrder	Order of the group the item belongs to relative to other groups (assign values 1, 2, 3, and so on for groups to be displayed). The group will be displayed in ascending value of group values assigned.
CC_Timeline_GroupOrder_Desc	Description for Group Order column.
CC_Timeline_URL	URL column to map for displaying the URL of each timeline item.
CC_Timeline_URL_Desc	Description for URL column.
CC_Timeline_ThumbnailURL	Column to map for thumbnail of image/icon to display for each item.
CC_Timeline_ThumbnailURL_Desc	Description for the `ThumbnailURL` column.
CC_Timeline_Filter	Column to map for timeline filter.
CC_Timeline_Filter_Desc	Description for Filter.
CC_Timeline_Footer	Web resource to display as the footer of the timeline.
CC_Timeline_Footer_Desc	Description for Footer column.

Input controls

Checkbox

Use the checkbox control to let users choose between two values by checking or clearing a check box. Configure it by using yes/no choice columns.

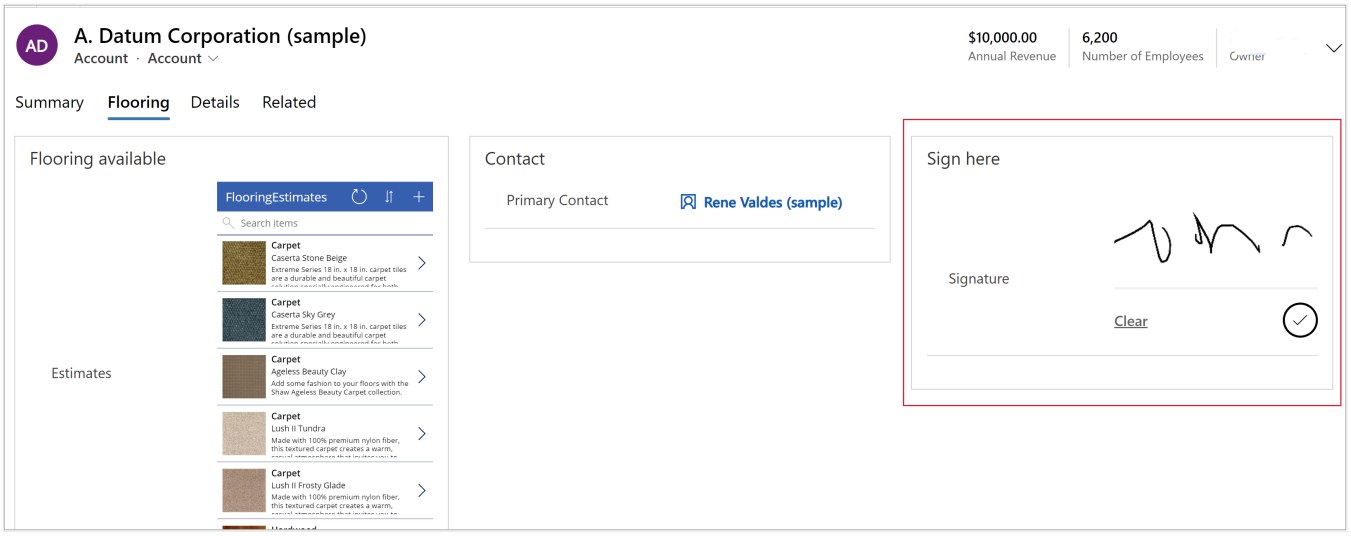


Option set

The option set control presents a set of choices for your users to choose from when entering data. It is made available by default when introducing a choice column into a form. More information: Choices

Pen input

Use the pen input control to capture written input such as signatures. The pen input control can be configured for multiline text columns for use with web, tablet, and phone clients.



> [!IMPORTANT]
> The minimum recommended **Maximum Length** specified for the column this control maps to is 15000.

Expand table

Property	Description
PenMode	Specify **PenMode!Draw**, **PenMode!Erase**, or **PenMode!Select** to determine what happens when a user drags a pointing device in a pen control.

Rich text editor

The rich text editor control provides the app user a WYSIWYG editing area for formatting text. The control's input and output format is HTML. The control allows copied rich text, such as from a web browser or Word, to be pasted into the control. It can be configured with text columns. More information: Add the rich text editor control to a model-driven app

Toggle

The toggle control allows users to choose between binary values, such as on/off and yes/no, by toggling the button. It can be configured with yes/no choice columns.

AI Builder

Business card reader

Use the AI Builder business card reader control to detect business cards and extract their information. You can take photos directly in the component or load images that you've taken. More information: Use the business card reader component in model-driven apps

Power BI

Power BI report

Embed a Power BI report in a model-driven app dashboard. More information: Create or edit a Power BI embedded system dashboard

More components controls

Some components aren't displayed in the default **Components** list in the form designer but can be added to the list by selecting Get more components.

Form component

The form component control lets users edit information of a related table record directly from another table's form. More information: Edit related table records directly from another table's main form

Collaboration controls

Collaboration controls give you the power to simplify your user's workflow collaboration. Build model-driven apps that allow users to work with approvals, files, meetings, notes, and tasks from Microsoft 365 and Microsoft Teams without switching the context from app to app. More information: Collaboration controls

Legacy controls

Timer control

The timer control shows your users how much time is available to complete an action in the resolution of an active row or how much time has passed since the time to complete the action has passed. More information: Model-driven app timer control overview

News control

Only available in classic.

Gain valuable insights from the latest news about your customers, competitors, and contacts. The news control delivers relevant news from Bing News. More information: Set up and use the news control

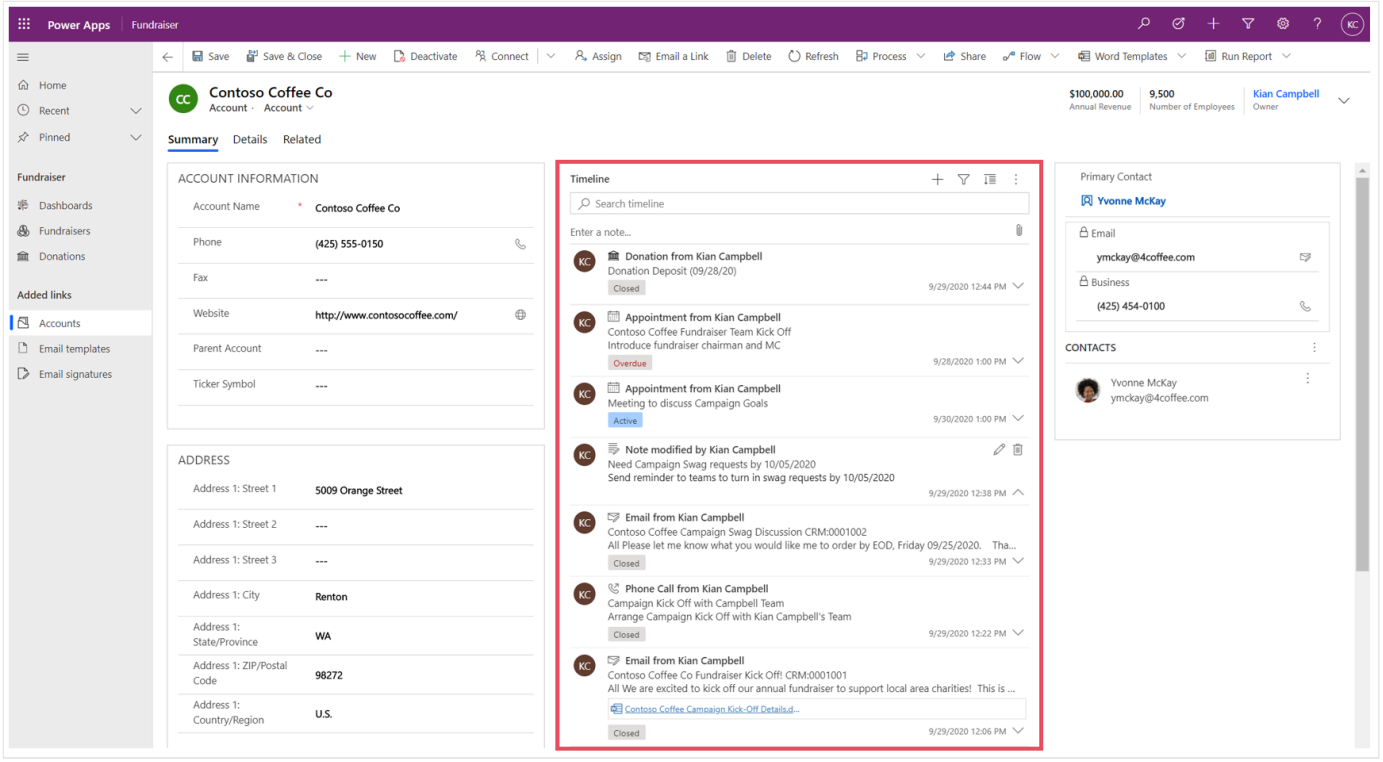
Next steps

Tutorial: Use custom controls for data visualizations

Last updated on 04/07/2026

Set up the timeline control

The timeline helps app users review all activity history. The Timeline control captures activities such as notes, appointments, emails, phone calls, and tasks. It tracks interactions for a related table over time. Use the timeline to quickly catch up on all of the latest activity details.



The timeline control provides an easy way to view information related to a table, such as accounts or contacts. This immersive experience helps give users a better understanding of their customer interactions, and helps them deliver more personalized service in an efficient and effective manner.

App makers can configure the information that appears on the timeline. Users can then access information and create activity records, such as emails and tasks, directly from the timeline.

Add a timeline

A timeline control is located on a form within a table. Any timeline control on a form can be configured. For example, an account table has three standard forms, and each of those forms can have a timeline that can be configured.

Display a custom table in a timeline

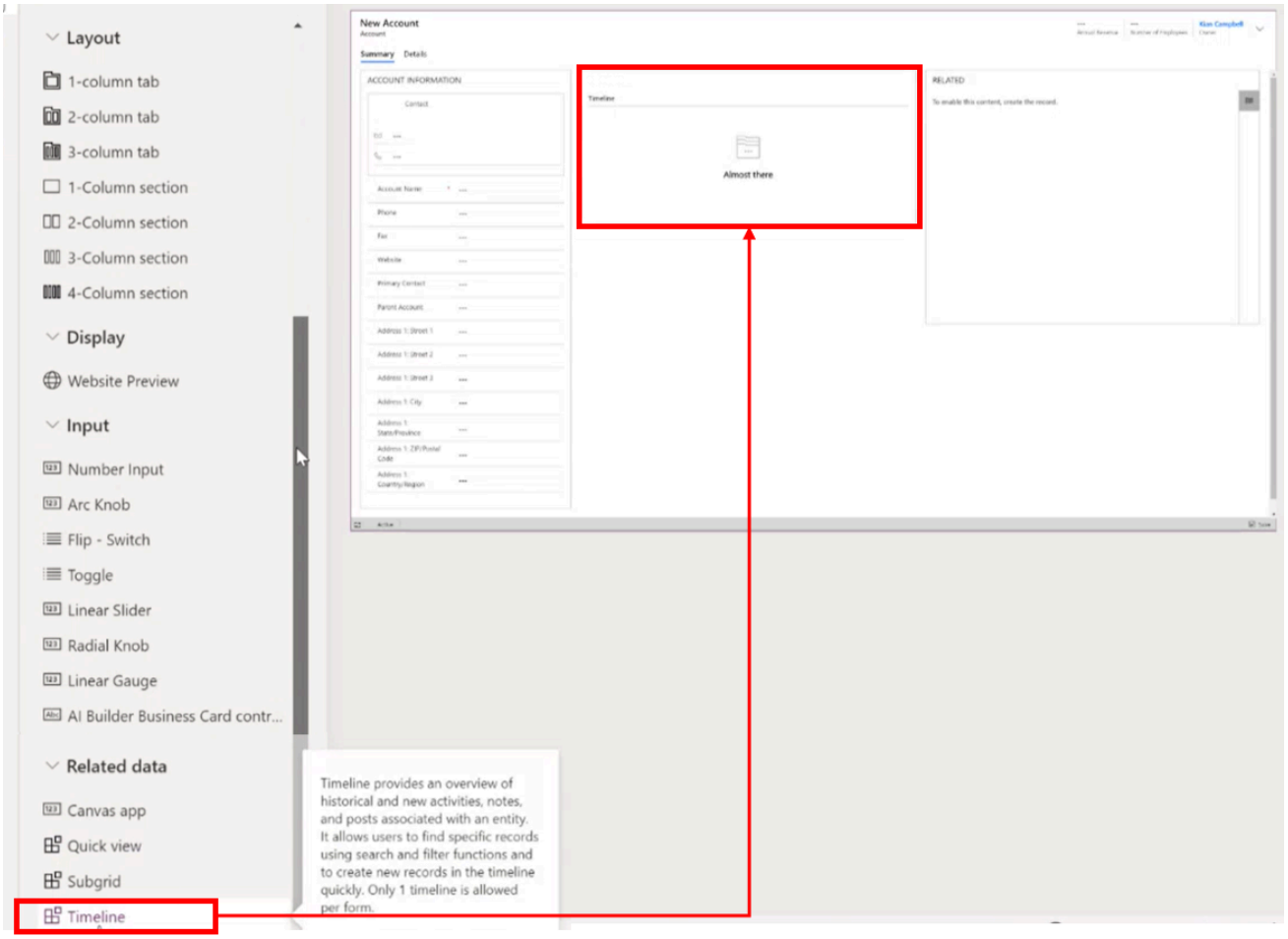
To add custom tables as activities in a timeline, configure the following settings when you create the table:

- Set the table **Type** as **Activity**.
- Check the **Enable attachments** setting if you want app users to be able to add attachments and notes to the custom activity record.

Learn more in Create a custom table.

Add the timeline component to a form

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane, and then open the table you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the **Forms** area, and then open the form where you want to add or configure a timeline.
4. In the form designer, select **Components** from the left navigation, and then scroll down to the **Timeline** component on the left side. Drag and drop it into a section on the form.



5. Make the changes you want to the timeline settings. Learn more in Configure the timeline component.

6. Before you can view any configuration changes on the table form, you must save and publish your updates on the timeline component. Select **Save**, and then select **Publish** to make your form changes available on the table form for the environment.

Note the following when you work with the timeline control in the form designer.

- You can add multiple timelines to a form.
- To remove the timeline component from a form, select the **Timeline** component area, and then press the Delete key.
- Since the timeline component relies exclusively on underlying related data, **Almost there** is displayed in the timeline section.
- Because you are in a create and/or edit state on the form, there's no underlying data, so the timeline section is blank in the form designer.

Configure the timeline component

The timeline component is rich in features and functionality that can be configured and tailored to support specific business needs. The timeline is composed of features and functionality that you configure in the timeline component, which is then displayed in the table form.

Display options

This section describes the settings available in the Display options area of the timeline control settings.

Timeline component name

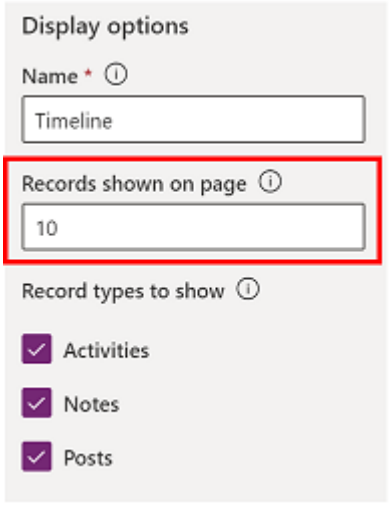
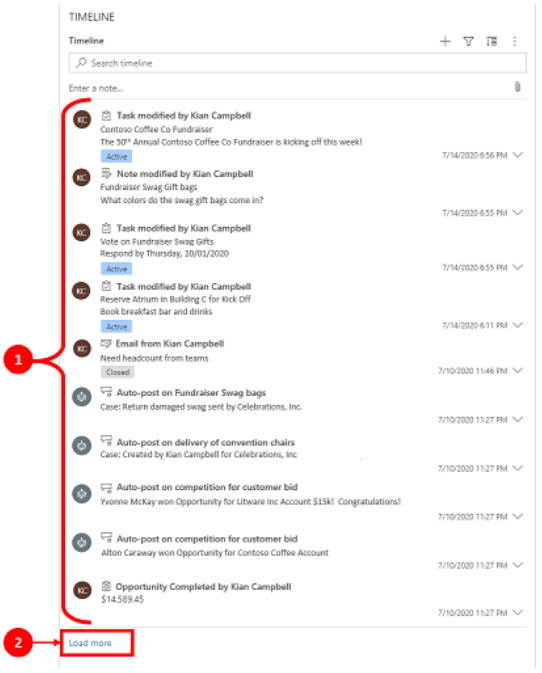
The **Name** column under the **Properties** tab in **Display options** is the unique name of the timeline control and is only used by the app maker to reference.

> ⓘ Note
>
> - The **Name** column has limitations. For example, you can't use spaces between words. Instead, you must use an underscore (_)
> - You can't change the default heading label that is displayed as **Timeline** on the form during app runtime.

Records shown on page

This section allows you to control the number of records that appear before displaying **Load more** at the bottom of the section.

Expand table

Form designer configuration View	Form designer display View
	
The default setting on the **Records shown on page** column is set to 10 records, but you can change it to display up to 50 records.	1. Per the default setting shown in this example, the form displays a maximum of 10 records. 2. Once records exceed 10, the **Load more** option appears at the bottom of the form.

Record types to show

There are three primary record types: Activities, notes, and posts. All record types are enabled by default.

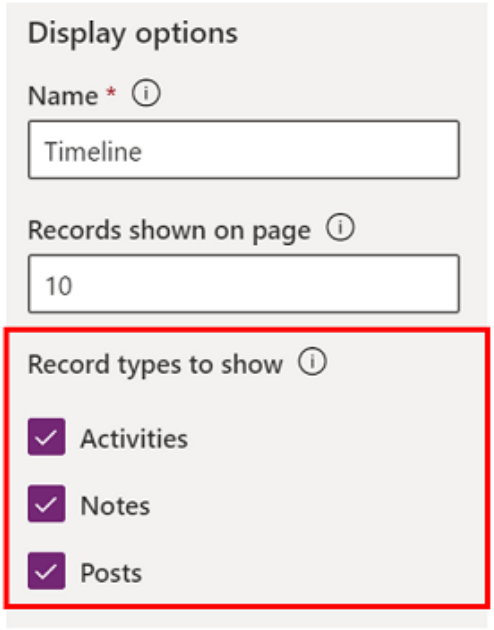
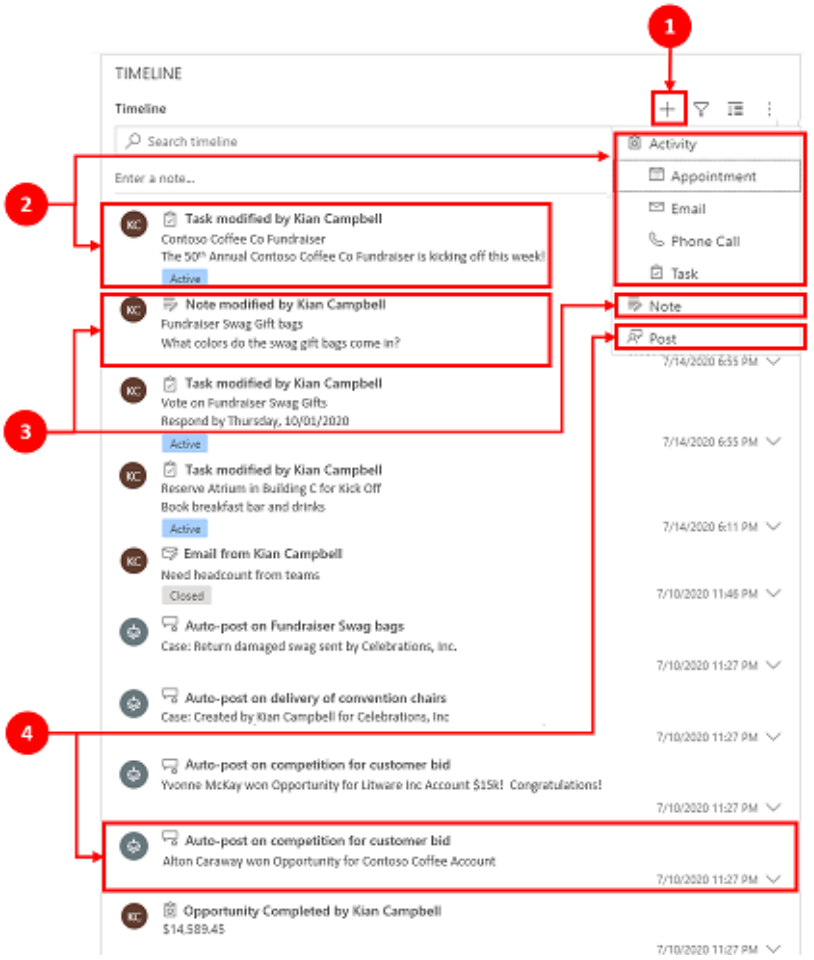
- Activities. Activities can have a large number of customizable subactivity record types to support business needs. Depending on what you install, the administrator can create, add, and display many different customized subactivity record types under the **Activity** section of the dropdown menu on the timeline.
- Notes. Notes allow you to capture details related to the table record. For example, you can use notes to capture thoughts, summarize information, and provide feedback on a case, and then update the case details later.
- Posts There are two types of posts: auto and user:
  - **Auto Posts** are system-generated posts that notify you of account activity.

- **User Posts** allow you to leave a message for another user on a record.

> **Note**
>
> Posts require a Dynamics 365 app, such as an app for Dynamics 365 Customer Service.

Enable these record types to be displayed in the timeline

Expand table

Form designer configuration view	Form designer display view
	
To enable **Record types shown** for **Activities**, **Notes**, and **Posts** on a form, check the box next to the record type.	1. To confirm that the record type is displaying on the form, select **Create a timeline record** + in the top-right nav in the timeline, and a dropdown menu that displays the list of enabled record types appears. 2. When **Activities** are enabled, the system displays **Activity** record types for appointments, email, phone calls, and tasks. 3. When **Notes** are enabled, the system displays the note record types on the form. 4. When **Posts** are enabled, the post record types display on the form.

> **Note**
>
> Activities and notes are standard entities. If additional solutions have been imported into the environment, other tables may also be displayed.

Activity, notes, post icons, and date / timestamp display on timelines

The timeline shows a simple icon before the activity, note, post, and custom table, making it easy for you to identify the record type.

Dates and timestamps always appear on each record on the bottom-right side of the preview, and are always viewable.

> **Note**
>
> When a custom table has a custom icon, activities related to the custom table are displayed using the default custom table icon not the custom icon.

Advanced

This section describes the settings available in the Advanced area of the timeline control settings. The **Advanced** settings apply to all record types.

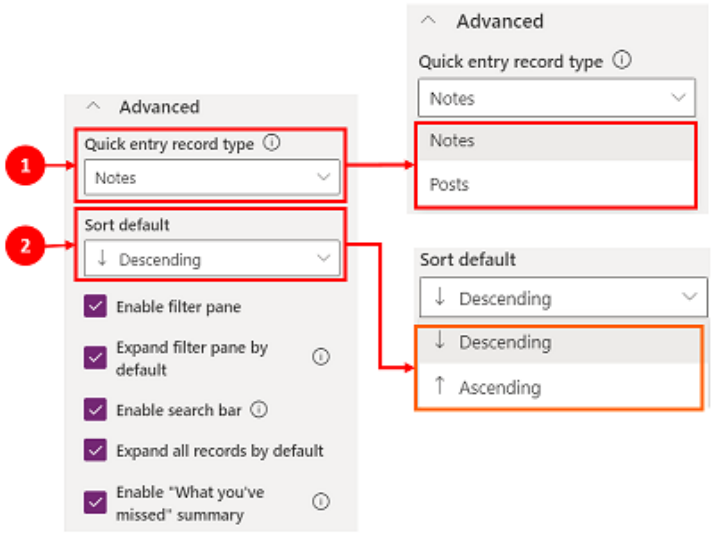
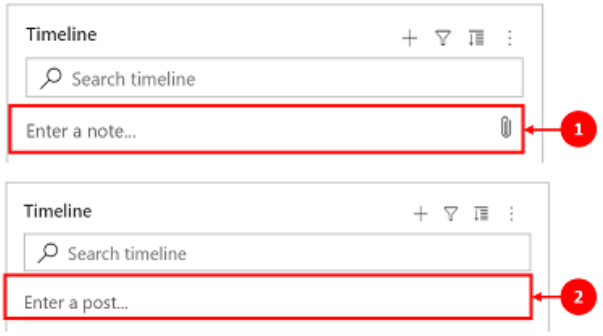
When enabling and disabling **Advanced** features, you aren't able to view updates (even though it might appear that there's activity occurring on the timeline component) until you save and publish your configuration updates and refresh the table form.

Advanced covers common settings that span across an activity, note, or post, which are the three primary record types that appear in the timeline section.

Quick entry record type and sort order default

The following is a detailed overview of the **Advanced** configuration options, starting with selecting your **Quick entry record type**:

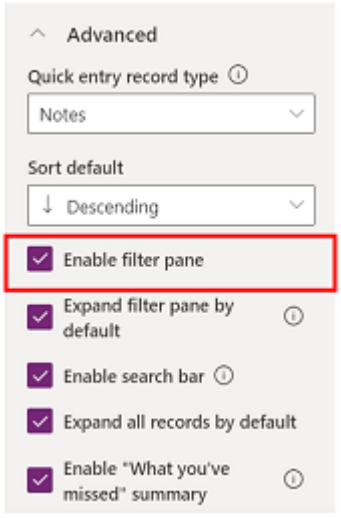
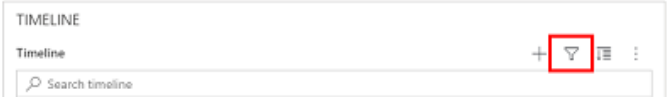
Expand table

Form designer configuration View	Form designer display View
	
1. Select **Advanced** to expand this area in the configuration view. The **Quick entry record** type provides you with quick access to create either a **Note** or a **Post**. The default setting for this setting is set to **Notes**. 2. The **Sort order default** setting controls the order of how all data is sorted on the timeline. The default setting for this setting is set to **Descending**.	1. When you select **Notes**, it displays under the **Search** bar in the timeline. Also, the system displays the paperclip icon that allows you to attach files. You can only attach files to notes. 2. If you select **Posts**, it also displays under the **Search** bar in the timeline.

Enable filter pane

Filters provide a quick option to sort and look for specific data quickly.

Expand table

Form designer configuration View	Form designer display View
	

Form designer configuration View	Form designer display View
The **filter pane** setting allows you to enable or disable filter functionality on timeline. It's enabled by default.	Enable the **filter pane** by checking the box next to the setting. This setting enables the filter icon to display on the timeline. To disable the **filter pane**, clear the box next to the setting and the filter icon no longer appears on the timeline.

Filter records on timeline

Filters are valuable for sorting data. You can quickly filter activities, notes, and posts with multiple options to display what matters to you. The filter is available for the activities, notes, posts, and custom tables that are present in the timeline. The timeline filters and displays the records and the count that are present in the timeline.

When you select filters based on an activity status then those activities, notes, and posts are displayed in your timeline. You can customize data using data filters and either choose to keep filters in place or clear them when you're done.

- When the **Filter** icon is transparent on the table form, it means no items were selected, so the filter pane is empty.
- When the **Filter** icon is dark, it means filters are set. To view which filters are set, select the **Filter** icon. The filter pane displays the selected filters.
- Choose how you want to filter data by selecting the box next to the filter.
- Clear filters by selecting the **Clear all filters** icon on the filter pane.

The following category and subcategory options are available on the filer menu:

Expand table

Category	Subcategory
Record type	• Notes • Activities • Posts
Activity type	• Appointment • Campaign Activity[1] • Campaign Response[1] • Email

Category	Subcategory
	• Fax • Case Resolution[1] • Letter • Opportunity Case[1] • Order Case[1] • Phone Call • Quote Close[1] • Recurring Appointment • Social Activity • Task • Project Service Approval[1] • Booking Alert[1] • Conversation • Session • Customer Voice survey invite[1] • Customer Voice survey response[1] • Custom activities (made available by imported solutions)
Activity status	• Active • Overdue • Closed
Activity status reason	Allows you to filter using specific status reasons. The values are a unique list of all of the status reasons from the activities in the timeline. The status reasons change depending on the activity. If multiple activities on the timeline have the same status reason, it appears once. The number next to it shows how many times the status reason occurs on the timeline.
Activity due date (active)	• Next 30 days • Next seven days • Next 24 hours • Last 24 hours • Last seven days • Last 30 days
Posts by	• Auto post • Users
Modified date	• Last 24 hours • Last seven days • Last 30 days

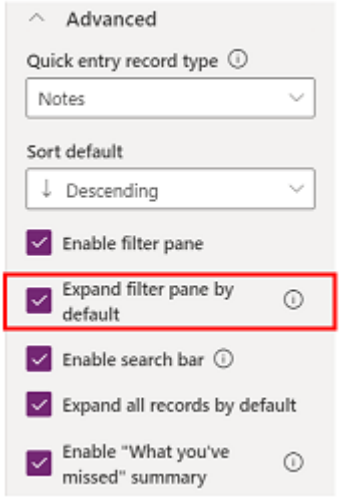
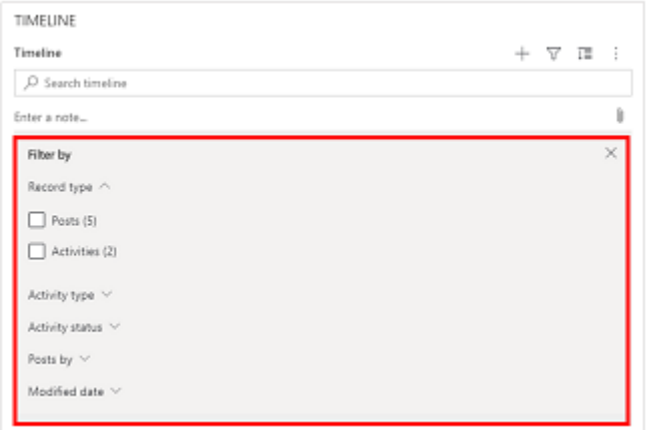
> **Note**
>
> By default, the fax and letter records are disabled on the timeline.

[1] Requires a Dynamics 365 app

Expand filter pane

The **Expand filter pane** setting provides quick access to sorting options within the timeline. It's enabled by default.

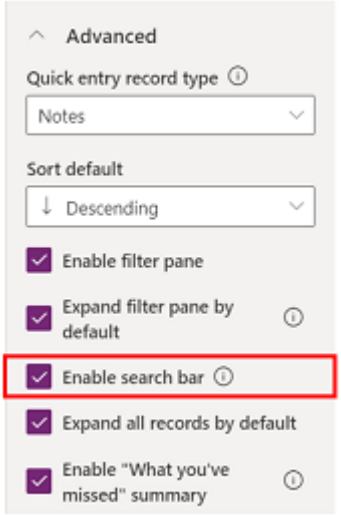
Expand table

Form designer configure view	Form designer display view
	
The **Expand filter pane by default** setting displays an expanded filter pane at the top of the form each time the form opens or refreshes. It's disabled by default.	Enable the **Expand filter pane** by checking the box next to the setting. This setting enables the filter pane to appear at the top of the form anytime the form is opened and refreshed. To disable the **Expand filter pane**, clear the box next to the setting and the filter pane doesn't appear on the timeline.

Enable search bar

You can search for records in the timeline. When you search for a phrase in the timeline, it searches in the title of the record or body and description column of the record then displays the record for you.

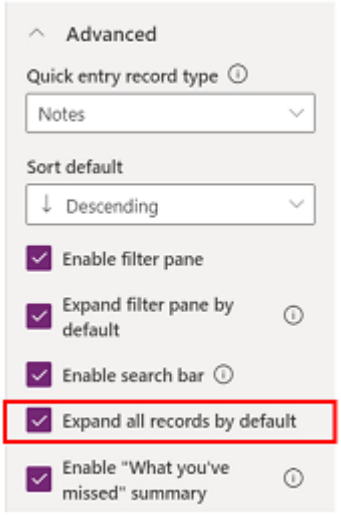
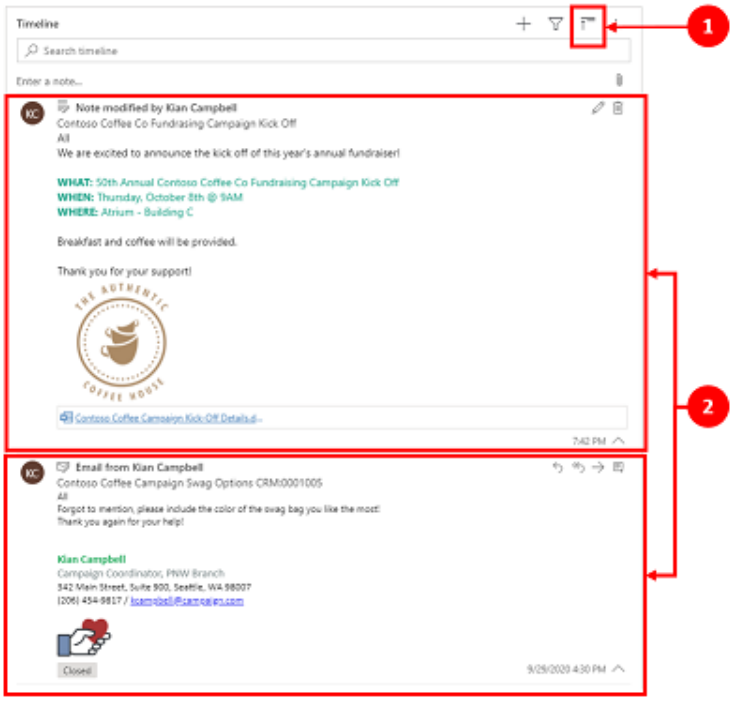
Expand table

Form designer configure view	Form designer display view
	
Select **Enable search bar** to enable the **Search timeline** bar functionality. It's enabled by default.	Enable the search bar setting to display a **search bar** at the top of the timeline. Disable the search bar by clearing the box next to the setting and the search bar doesn't appear on the timeline.

Expand all records in timeline

Expand all records by default displays all activities in an expanded view in timeline.

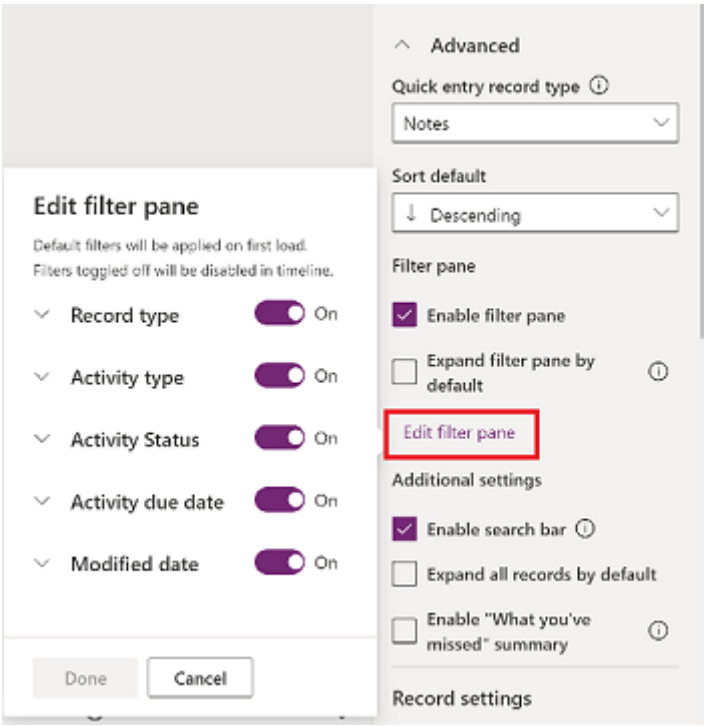
Expand table

Form designer configure view	Form designer display view
	

Form designer configure view	Form designer display view
Select **Expand all records** to set the default view to display all records in the expanded view format in the form each time the timeline is opened. **Expand all records** is disabled by default.	1. When enabled, the **Expand all records** icon is displayed in the top-right corner of the timeline nav. 2. Records can be expanded or collapsed by using the **Expand all records** icon. When expanded, all records display in the expanded view each time the form opens. When you clear **Expand all records**, it no longer displays activities in an expanded view. When disabled, the **Expand all records** icon doesn't display in the top-right nav of the timeline. Records are always be displayed in a collapsed view.

Edit filter pane

You can configure the default filters that are applied when a form loads or is refreshed using **Edit filter pane**. Remove filter groups by turning the setting **Off**. Users can remove the default filters to display all the records unless **Enable filter pane** is disabled.

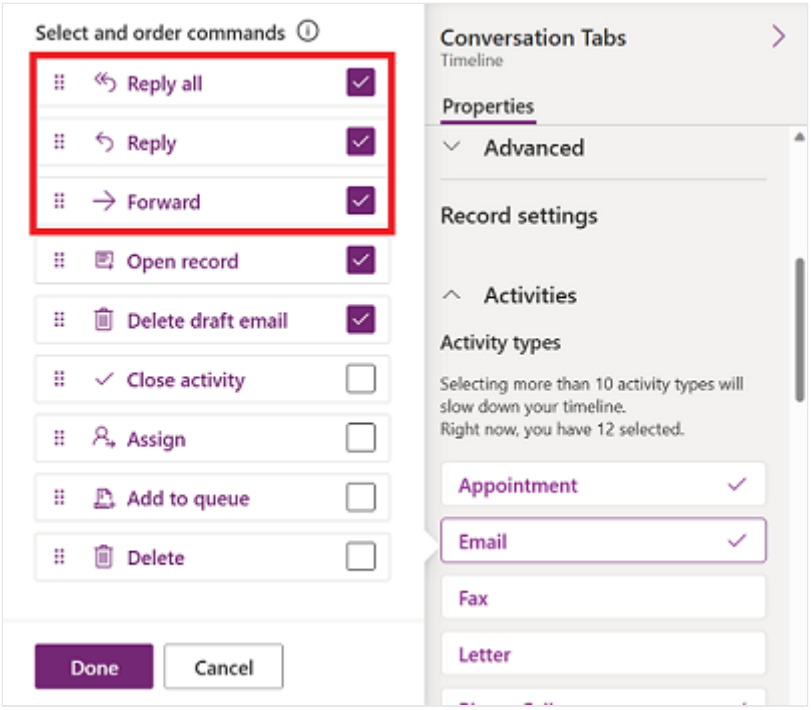


Configure email commands for timeline

You can configure email options for the timeline so that users can reply, reply-all, and forward messages. You can select the order in which the options are displayed for users, and exclude them if they're not needed. Reply-all is the default order setting.

1. Select the form for which you want to apply the email settings, and then in **Properties** in **Conversation Tabs**, select the **Email** activity type. The **Email** properties pane is displayed.

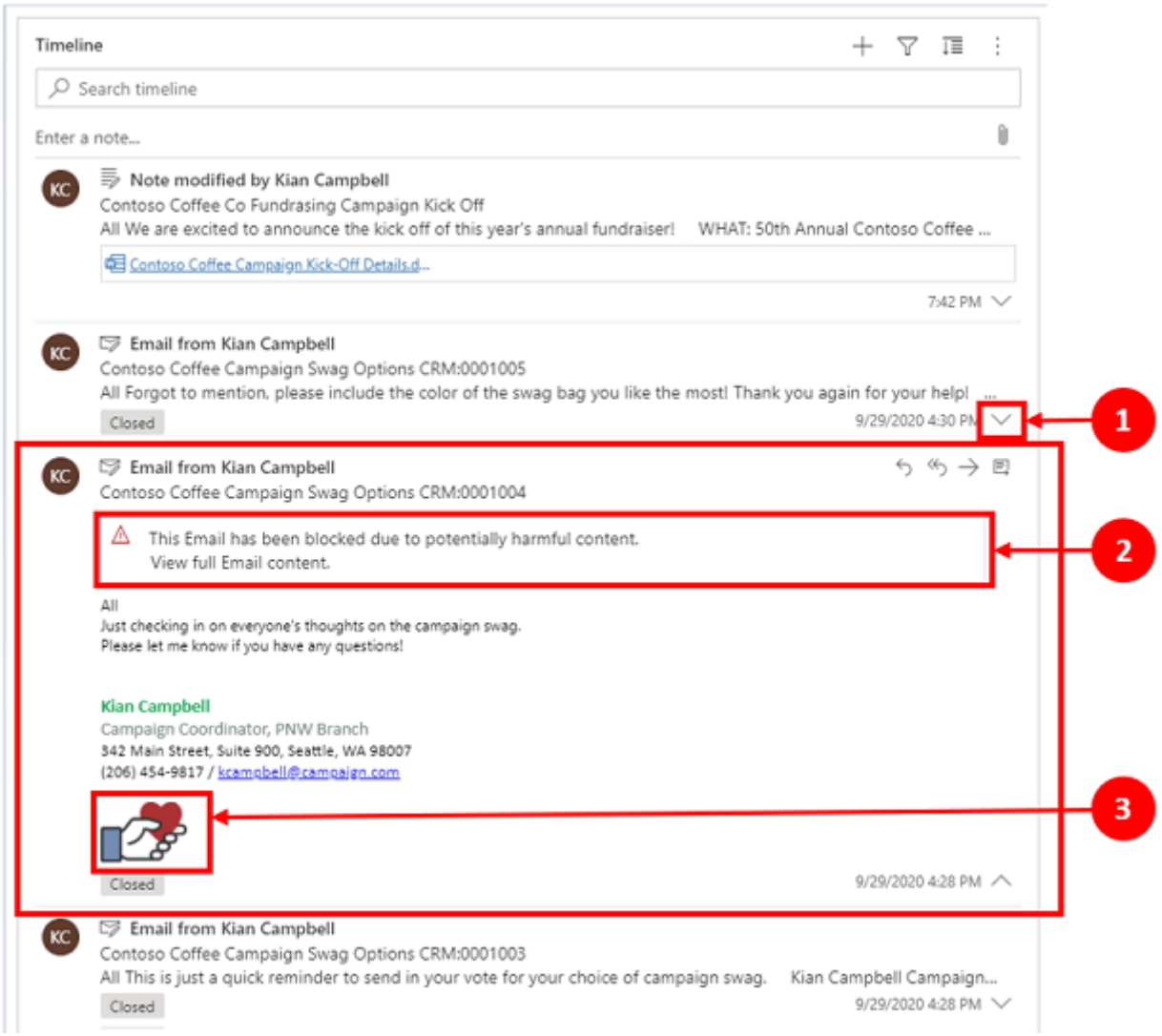
2. In **Select and order commands**, check the boxes for the email commands you want to configure for users. You can drag and drop the options to choose the order in which you want each option to display.



3. Select **Done**, and then save and publish your changes.

Expand records with images in timeline

You can send and receive records with images, but they don't display when the record is collapsed. To view the images, you must expand the records.



1. Records when collapsed provide a visual summary. To expand an individual record, select anywhere on the timeline record to expand and collapse a record view. In the bottom-right corner of the record, there's a caret:

    - When the caret is facing downward (v), the record is collapsed.
    - When the caret is facing upward (^), the record is expanded.

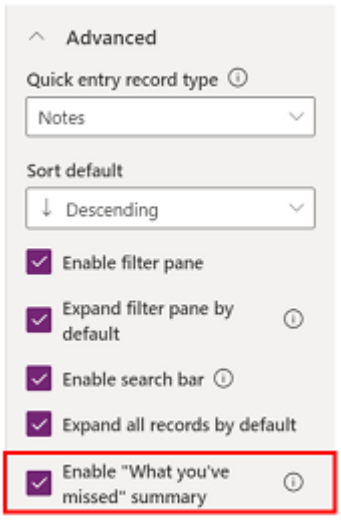
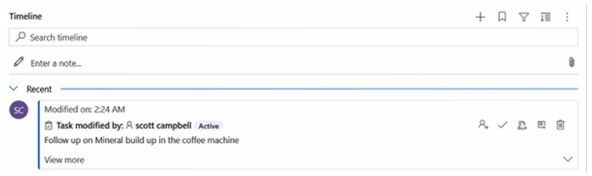
2. Records with images might display the following notice:
   This email has been blocked due to potentially harmful content. View full email content.

3. When you select the message, the warning goes away, and the image appears.

Enable "What you've missed" summary

The **What you've missed** summary helps you stay on top of updates and changes made to records by displaying updates at the top of the timeline when you access a record.

Expand table

Form designer configure view	Form designer display view
	
The **What you've missed** setting displays new records that are available for review. It's enabled by default. To disable it, clear the box next to the setting.	When enabled, when you view an account record, the newest activities appear at the top of the list.

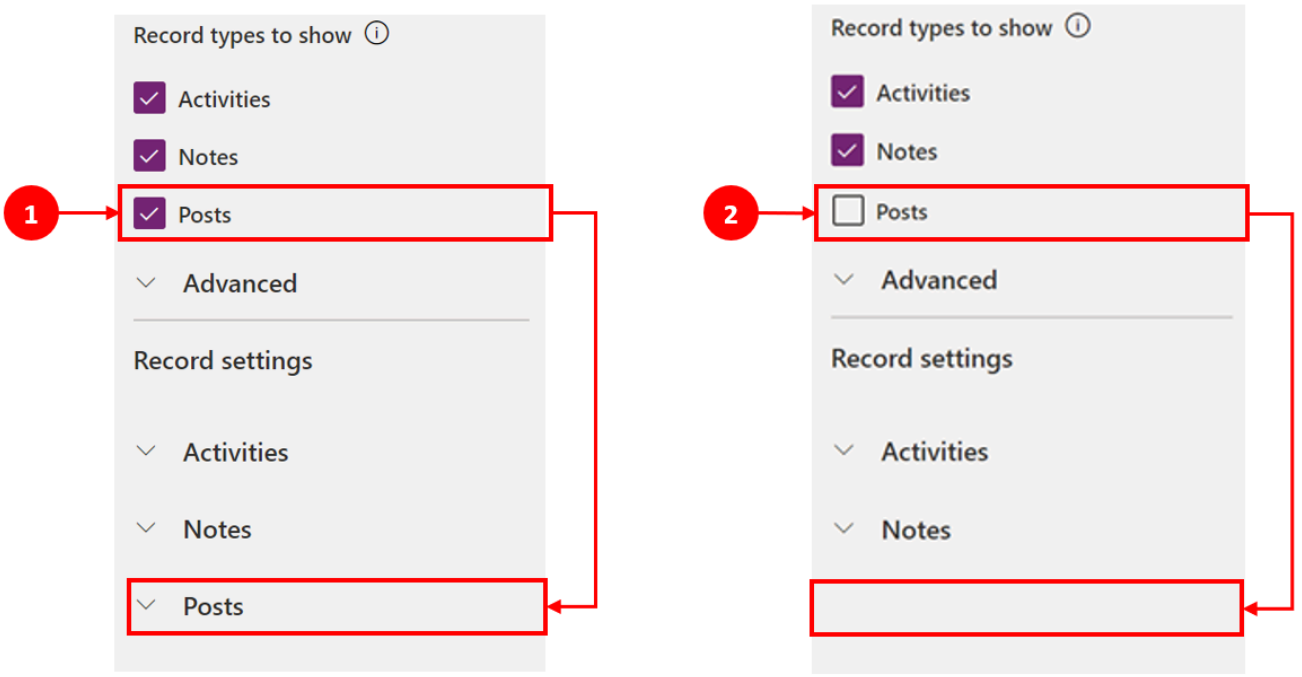
Record settings

This section describes the settings available in the Record settings area of the timeline control settings. The **Record types to show** setting determines the record types that are affected by the **Record settings** described here.

Record settings let you manage the settings within the record types.

- The **Activities** record type is tied to **Activities** in record settings.
- **Notes** record type is tied to **Notes** in record settings.
- **Posts** record type is tied to **Posts** in record settings.

To enable or disable a record type, select or clear the checkbox. Then, that record type either displays or no longer appears in the **Record settings** section.

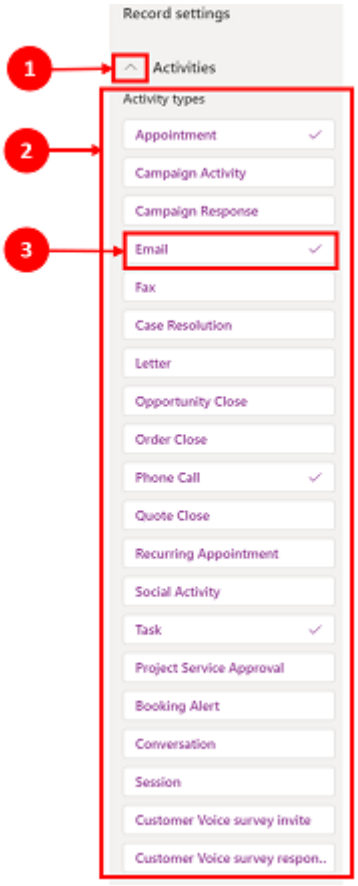
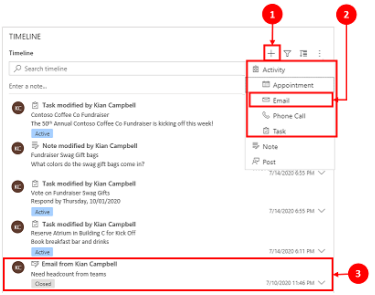


1. When **Posts** is checked in the **Record types to show** section, it appears in the **Record settings** section.
2. When **Posts** is cleared in the **Record types to show** section, it no longer shows in the **Record settings** section.

Configure activity record types

When you expand the **Activities record settings** on the timeline component section, a list is displayed that shows all the activity types that can be either enabled or disabled on the table form.

Expand table

Form designer configuration view	Form designer display view
	
1. Expand and view **Activities** under the **Record settings** section using the caret (^). 2. A list of **Activity types** is displayed in the expanded view. 3. You can enable or disable an activity by selecting an activity type. For example, enable **Email**. To enable an **Activity type**, check the box next to **Enable** and select **Done**. To disable an **Activity type**, clear the box next to **Enable**, and then select **Done**. This action disables all other items in the box and disables the activity type from displaying on the timeline. This action also disables the activity type from being created or viewed in the timeline.	1. When enabled, an **Activity type** appears under the **Create a timeline record** +. 2. The activity type is displayed as an option the user can choose from the dropdown menu. 3. Also, the **Activity type** record is displayed in the body of the timeline.

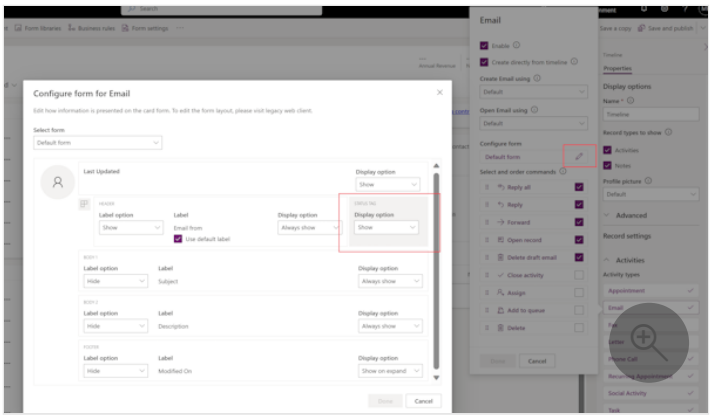
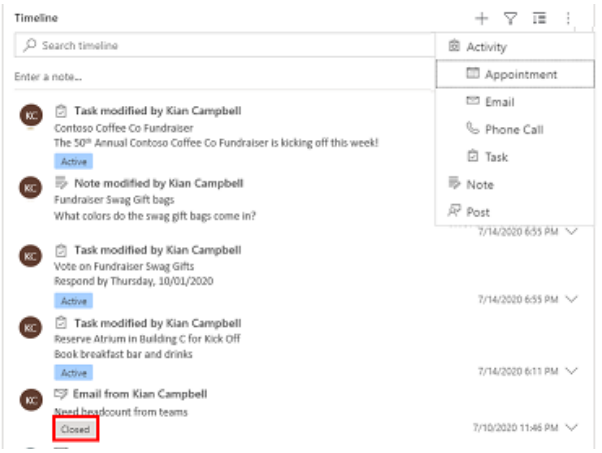
> **Note**
>
> A checkmark appears next to enabled **Activity types**. Other record type settings remain disabled until you enable them for that record type.

Show status tags on activity record types

Status tags match the status filters that display in the timeline to help you be able to quickly review whether the state of an activity record is **Active**, **Overdue**, or **Closed**, such as on a task,

appointment, or email. Makers enable or disable status tags for any **Activity type** in the default form configuration. Status tags are enabled by default.

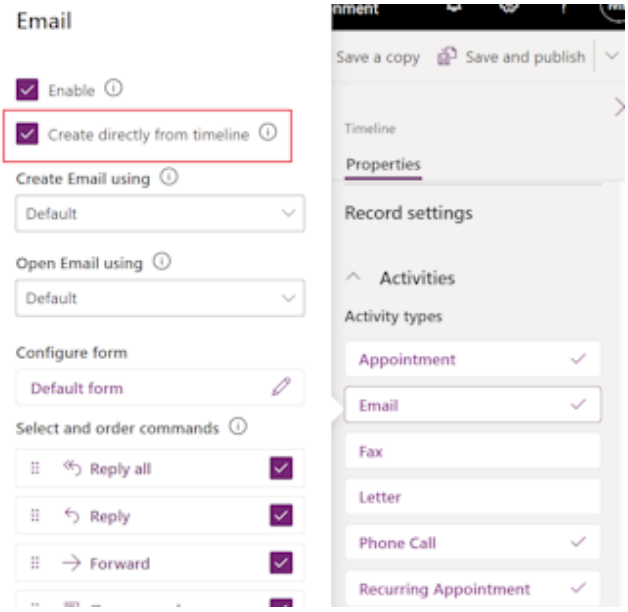
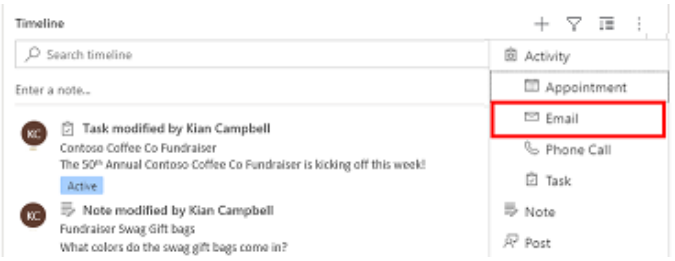
Expand table

Form designer configuration view	Form designer display view
	
To display email status tags, select **Default form** (pencil icon) > **Show** in the **Display option** dropdown list under **STATUS TAG**.	When enabled, status tags such as **Active**, **Overdue**, or **Closed** appear in the timeline next to that activity record.

Enable the ability to create directly from timeline

App makers have the ability to enable activity types so they can be created directly on the timeline. Having the ability to quickly select and create an activity, such as email, tasks, and appointments, helps to streamline productivity.

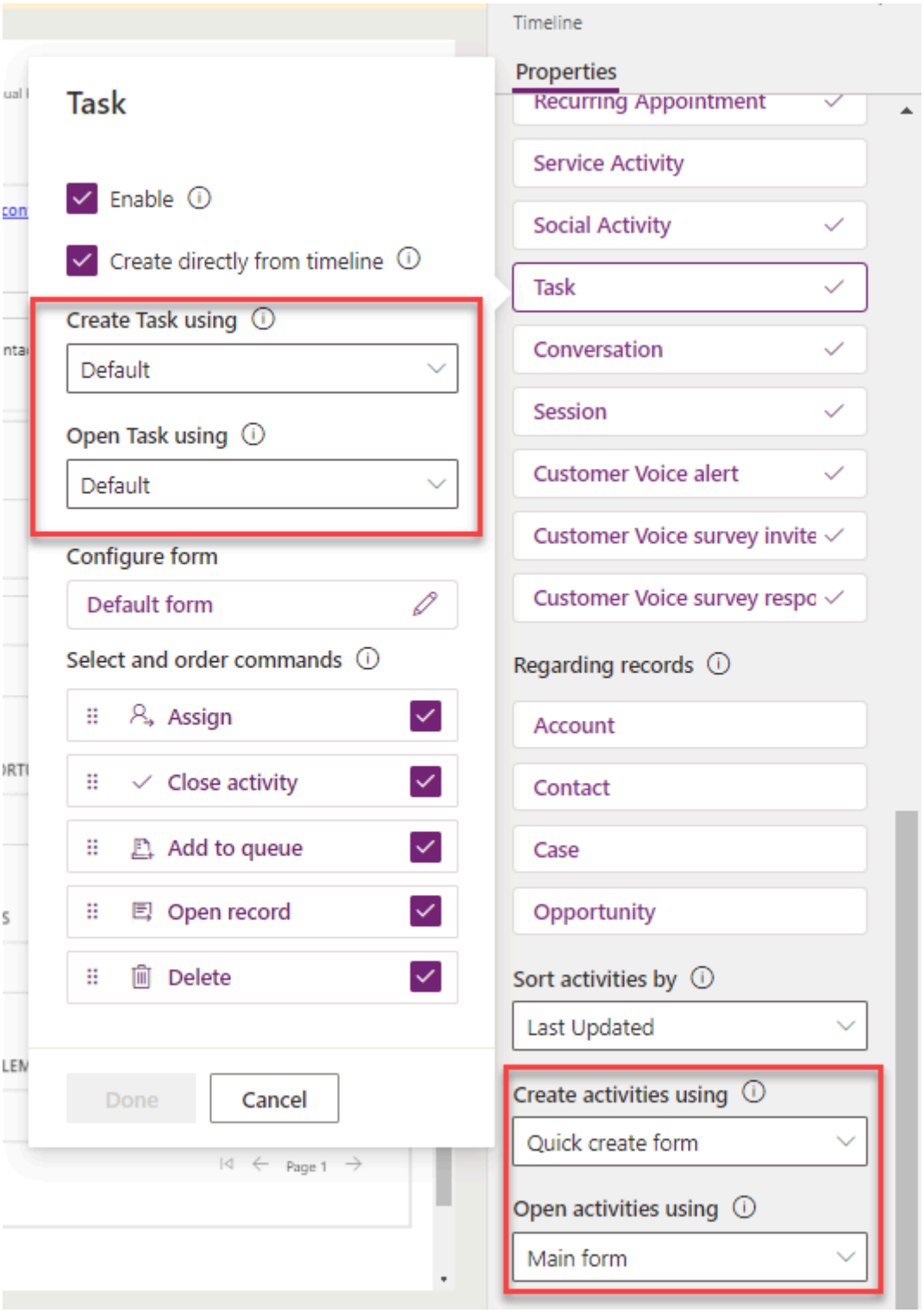
Expand table

Form designer configuration view	Form designer display view
	
To allow users to create activity types directly from the timeline, check the box next to **Create directly from timeline**.	When enabled, the activity type appears in a dropdown box on the **Create a timeline record** icon in the top-right of the timeline.

Set form types to create and open activities

Set the **Create activities using** and **Open activities using** individually for an activity type or for all activity types to set the form type users use based on your business needs. You can set these fields as follows:

- Select an activity type in **Activities** and specify the values.
- Specify the value for these fields in the **Activities** pane.



If you set the following form type for a specific activity, the application considers the following settings:

- **Default**: The form type in the **Create activities using** or **Open activities using** in **Activities**.
- **Quick create form**, **Main form**, or **Main form dialog**: The activity specific setting irrespective of the form type set in **Create activities using** or **Open activities using** in **Activities**.

When a user creates or opens an activity from the timeline, it uses the selected form type.

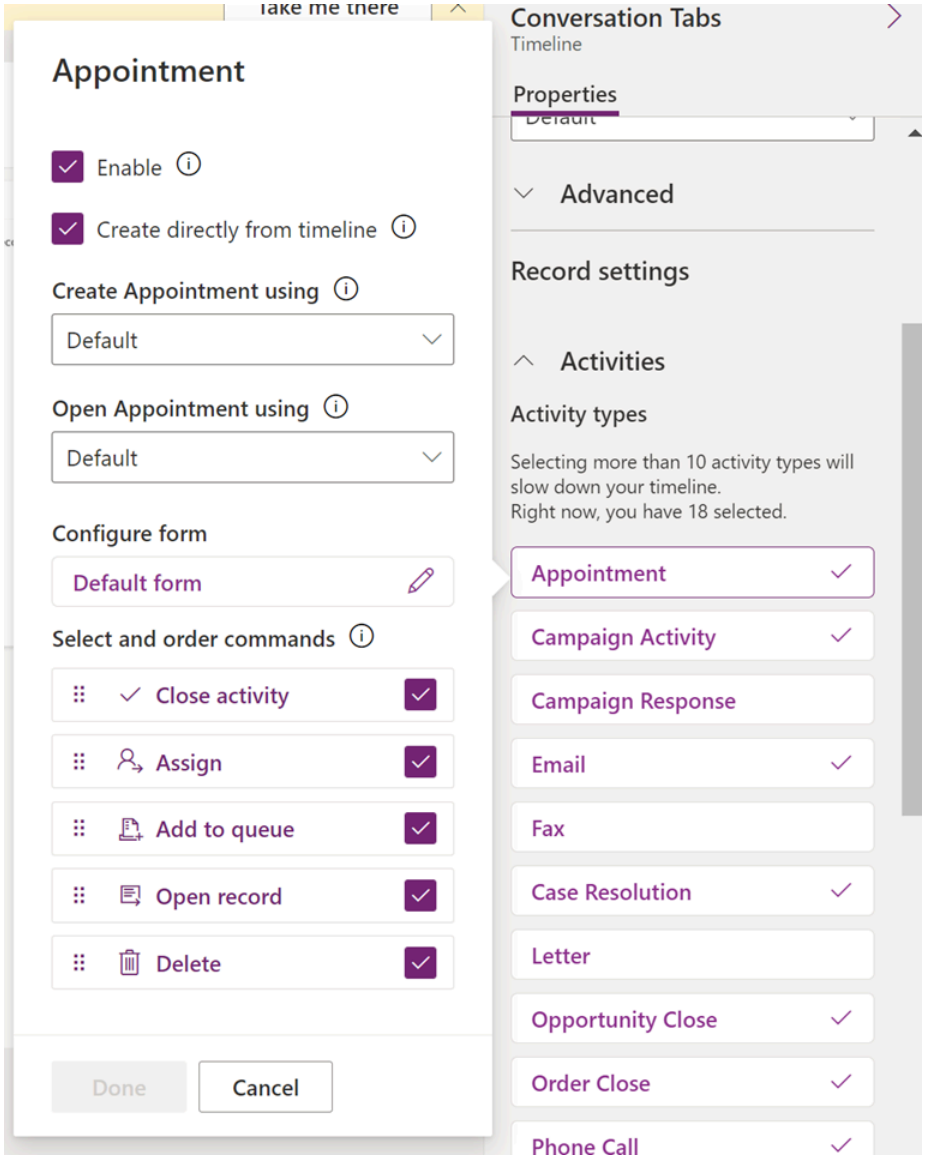
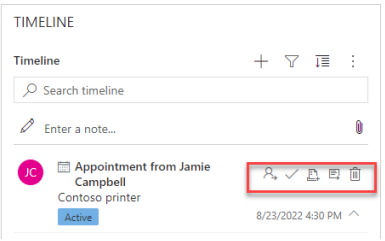
Enable command actions on activity record types (Preview)

> **Note**
>
> These features are part of an early access release. You can opt in early to enable the features in your environment, which allows you to test them and then adopt them across your environments. For information about how to enable these features, refer to Opt in to early access updates.

Command actions allow users to assign, close, delete, open a record, or add a record to a queue for activities from the timeline. You can enable or disable the command actions for a specific activity type.

Based on the activity type, you can also enable other command actions. For example, for an email you can also enable actions such as Reply, Reply All, and Forward.

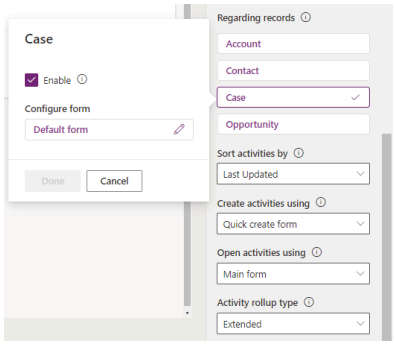
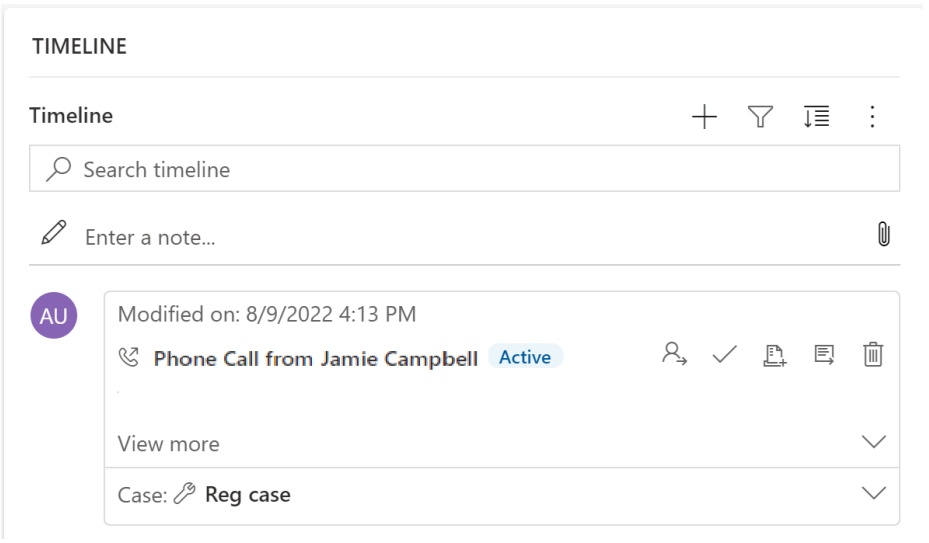
Expand table

Form designer configuration view	Form designer display view
	
To allow users to enable command actions for an activity directly from the timeline, check the box next to command action in **Select and order commands**. You can only enable five command actions for an activity.	
You can also move the position of the command actions to change the order in which they're displayed on the timeline.	When enabled, the command action appears on the activity record in the timeline.

Display related records on the timeline

Accounts, contacts, cases, and opportunities linked to an activity are displayed on the timeline record. The administrator can enable or disable the records in **Regarding records**.

Expand table

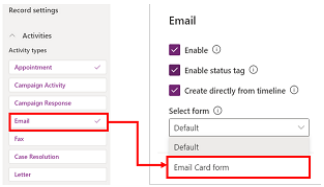
Form designer configuration view	Form designer display view
	
1. Expand and view activities under the **Record settings** section using the caret (^). 2. A list of records are displayed in **Regarding records**. 3. To enable records, select the box next to **Enable** and select **Done**.	When the checkbox is enabled, users can view the linked records to an activity. The record linked to the activity is then displayed on the timeline. The linking of the record isn't manual and it occurs due to the rollup of the record. Learn more in Set the activity rollup type in timeline.

> **Note**
>
> A check mark appears next to the enabled record. Disabling the table type only disables the linked record for the Timeline records rolled up for that table.

Create and use card forms in timeline

Records are displayed using the default setting for each activity type. To display record information for an appointment or email, edit the existing card form, select a different card form in record settings, or customize a new card form.

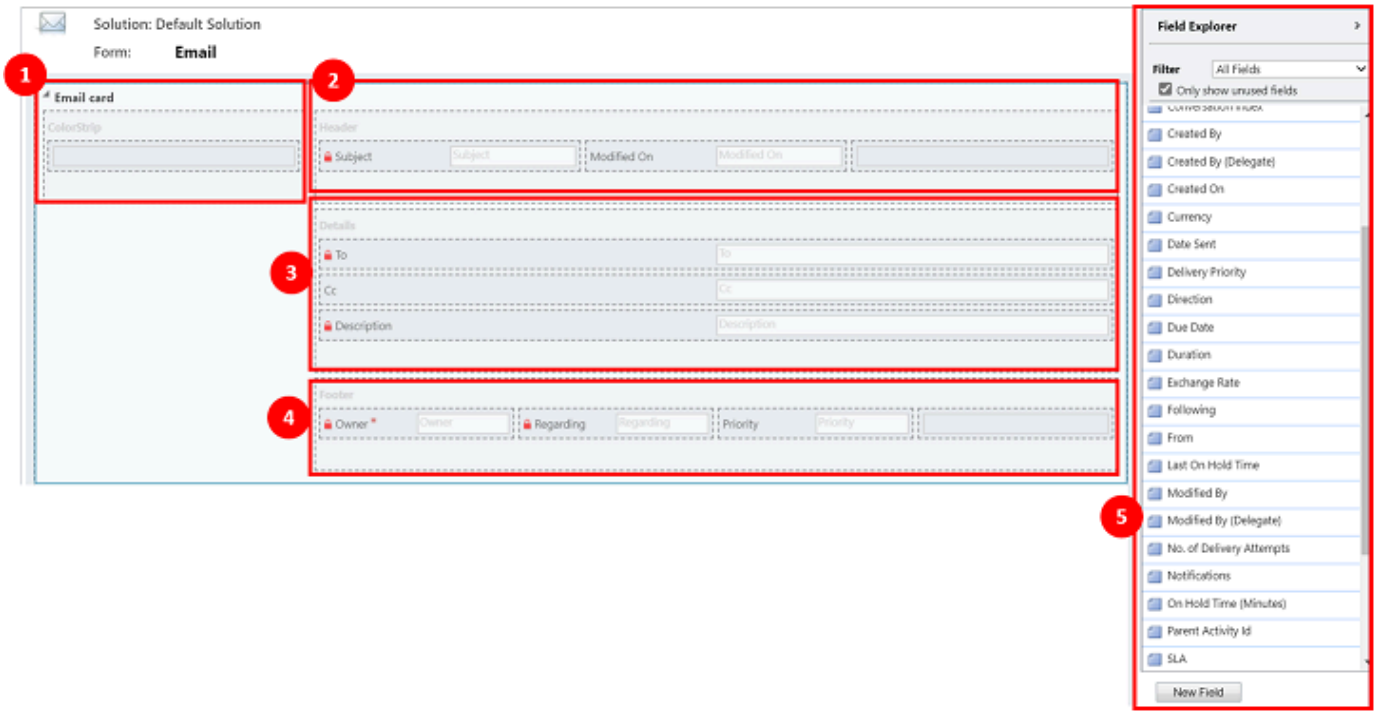
Expand table

Form designer configure view	Form designer display view
	
You can change the default card settings to a different card form if one exists.	If you create a new card form, you must go to the parent table and add the new form of card type there before it appears in the timeline list for configuration. If **Default** is displayed, you aren't using the card form. You can't use the **Email card form** in a timeline unless you select and publish it first.

> **Note**
>
> Not all activity types allow you to create card types, so the default selection is your only option for those records.

Customize a card from within timeline

All card forms are separated into the following four sections:



Legend

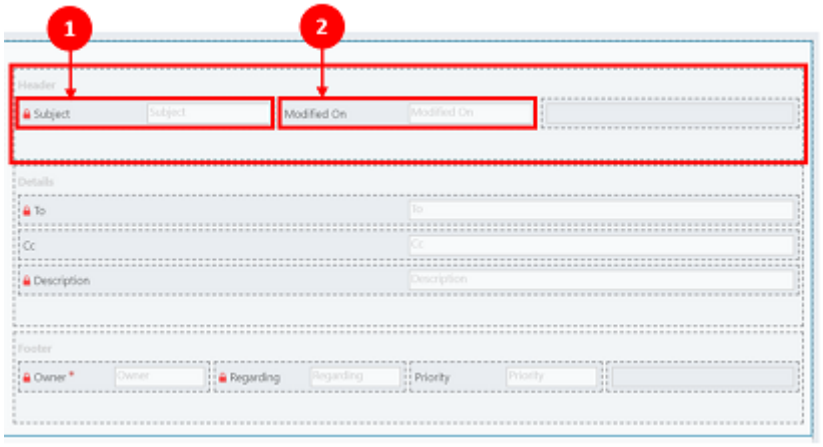
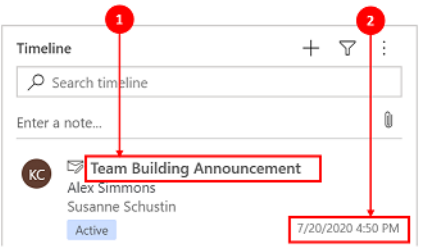
1. **ColorStrip**: This section doesn't appear on the timeline record. The **ColorStrip** is located on the left of the card form.
2. **Header**: This section is displayed on the timeline record; however, only the first two columns are displayed on the timeline record. For this example, only the **Subject** and **Modified On** columns are visible.
3. **Details**: This section is displayed on timeline record; however, only the first three columns are displayed on the timeline record. For this example, only the **To**, **CC**, and **Description** columns are visible on the timeline record.
4. **Footer**: This section isn't displayed on the timeline record.
5. **Table columns**: You can select which columns you want to add to your card form from the column options listed. You can customize your card form by dragging and dropping the columns you want to use into the sections you want that column to appear in on the timeline record.

Each individual card form must be customized for each activity record, such as email, tasks, posts, and so forth.

Header section

The card **Header** displays the title/subject in your timeline email form. You can have up to six columns in the **Header** section, but only the first two columns display on the timeline record. The form ignores empty columns in all sections.

Expand table

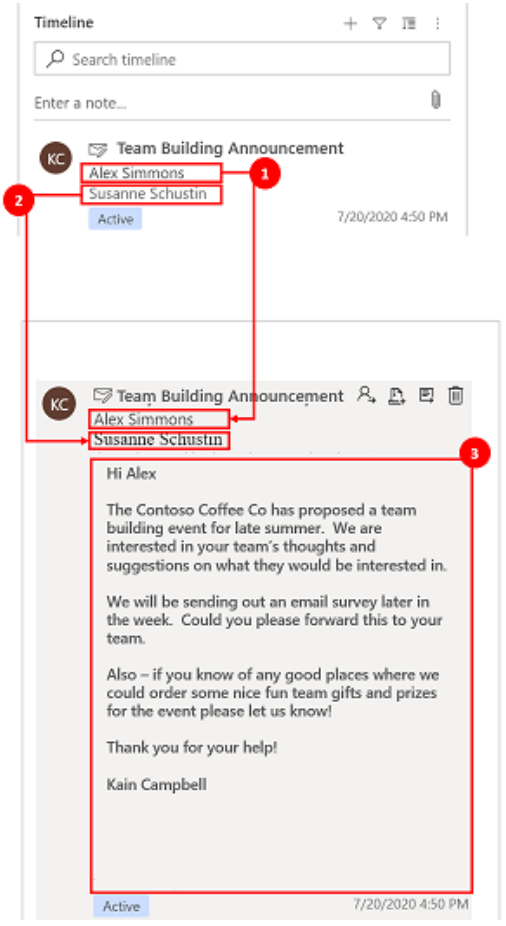
Form designer configuration view	Form designer display view
	
Column 1 1. Regardless of the column you choose for this section, it appears as a bold header at the top of your timeline record. For this example, we selected **Subject** for this column.	**Column 1** 1. Column 1 from the card header is always displayed in this section of the timeline record.

Form designer configuration view	Form designer display view
Column 2 2. Again, regardless of the column you choose for this section, this column always appears in the bottom-right corner of the timeline record. For this example, **Modified On** is selected for this column.	**Column 2** 2. Column 2 from the card header is always displayed in this section of the timeline record.

Details section

The card **Details** section displays in the body of your timeline email record. You can have up to four columns in the **Details** section, but only the first three columns are viewable on the timeline record.

Expand table

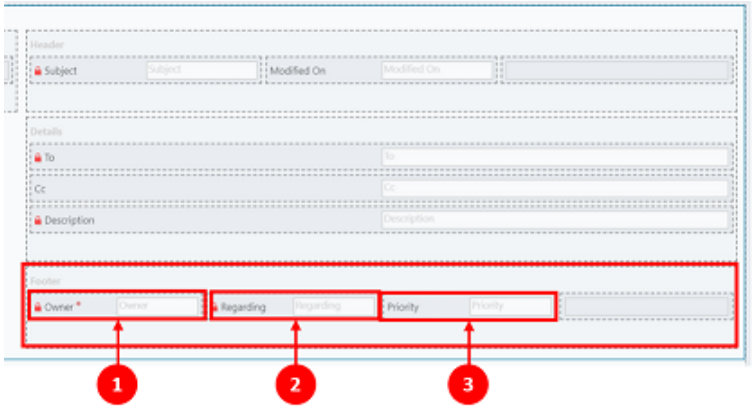
Form designer configuration view	Form designer display view
	
The card details always appears below the header regardless of the column you choose. **Column 1** 1. In the card details, Column 1 acts as a subheader on the timeline record. For this example, **To** is selected for this column. **Column 2**	**Column 1** 1. This column always displays in this section and acts as a subheader on the timeline record. **Column 2** 2. This column always displays in this section and only displays one line of text

Form designer configuration view	Form designer display view
2. This column only displays one line of text in a summary view on the timeline record. When you expand your timeline record, content in this column is fully displayed and formatted. For this example, **CC** is selected for this column. **Column 3** 3. This column follows the content of **Column 2** and is part of the main body of your timeline record that is only viewable when you expand the record. For this example, **Description** is selected for this column.	in the summary view but when expanded, content is fully displayed. **Column 3** 3. This column always displays in this section and is only viewable when the record is expanded.

Footer section

This section isn't visible on the timeline record.

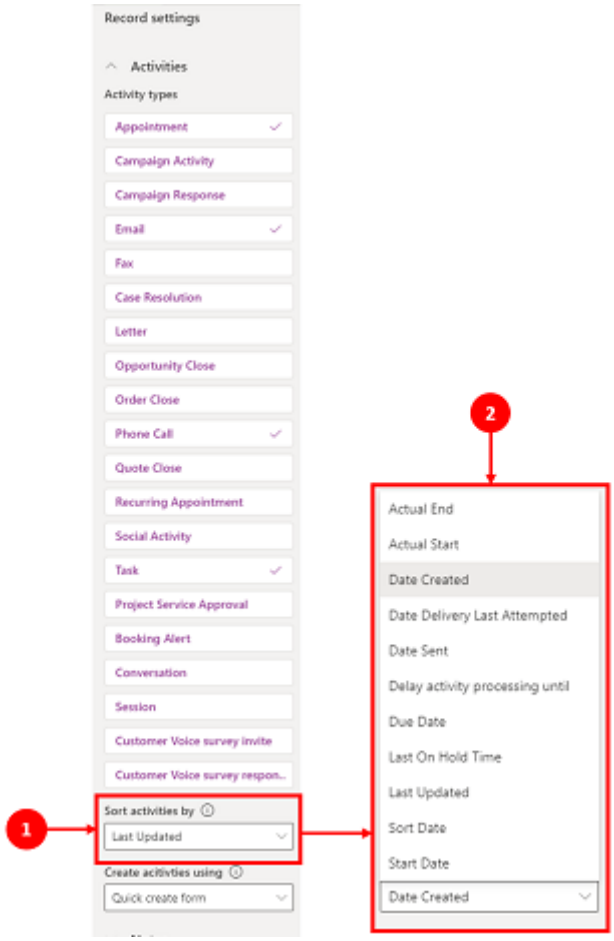
Expand table

Form designer configuration view	Form designer display view
	
Column 1 1. For this example, we selected **Owner** for this column. **Column 2** 2. For this example, we selected **Regarding** for this column. **Column 3** 3. For this example, we selected **Priority** for this column.	These columns aren't visible on the timeline record

> **Note**
>
> The **Modified On** field updates to the time of the modification. The **Modified By** field assigns the record to a single user when the table is reassigned. This behavior occurs due to cascading relationships. Learn more in Configure table relationships cascading behavior.

Set the date to use when sorting activities in timeline

How users view data is important, and setting a default display view of the data varies based on the needs of your business. App makers can choose how data is sorted and create a default setting for **Activity types** in **Record settings**. **Last Updated** is on all activities. It's the default setting in ascending order.



Legend

1. The **Sort activity by** setting in the **Activities record settings** allows you to control how data is sorted in timeline.
2. The **Sort activities by** column displays a list when selected. You can select from this list how you want your data to be sorted and displayed on timeline in the form.

Sort date

Some dates can only exist on specific types of activities. For example, **Date sent** or **Date delivery last attempted** only apply to email. If you sort by such dates, then non-email activities end up grouped together without any ordering. You can't create a custom date column, but if you need more flexibility, you can use **Sort date**, which is empty by default and requires that you populate

it for each activity record with the date you want to use for sorting. Some of the ways you can populate the date are by using Microsoft Power Automate, business rules, or JavaScript.

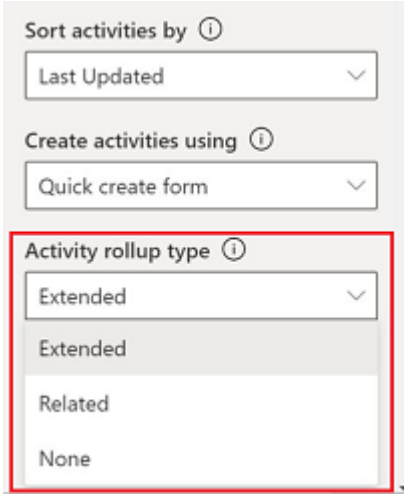
When using Sort date, keep in mind the following details:

- If you set a value in **Sort date**, you can use it for more customized sorting, but you have to populate it for every activity record or it doesn't work. The sort date has to be configured for each timeline instance, and must be set up for all three main forms in the account table.
- If the **DateTime** column is on the same calendar day, the date doesn't display if **DateTime** was earlier in the day compared to the current time.
- The sort date isn't based on a 24-hour period, but rather, compares the **DateTime** column value with the current date and time (based on the user's preferred time zone). If the value entered occurred earlier in the day, the date isn't displayed.

Set the activity rollup type in timeline

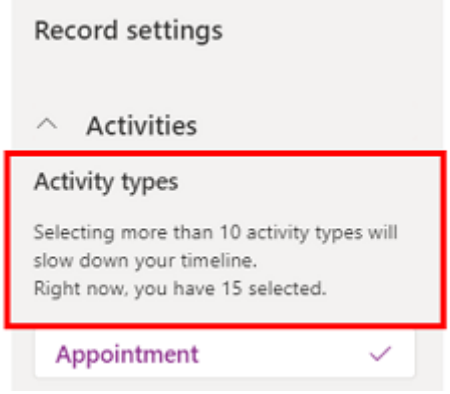
The activity rollup type can be configured for timelines on forms for the account and contact tables. The available types of rollups are **Extended**, **Related**, and **None**. Activity rollup only affects accounts and contacts in Dynamics 365 apps, such as Dynamics 365 Customer Service applications. To only show activities that are directly related to the table in timeline, select **None**.

Learn more in RollupType EnumType.



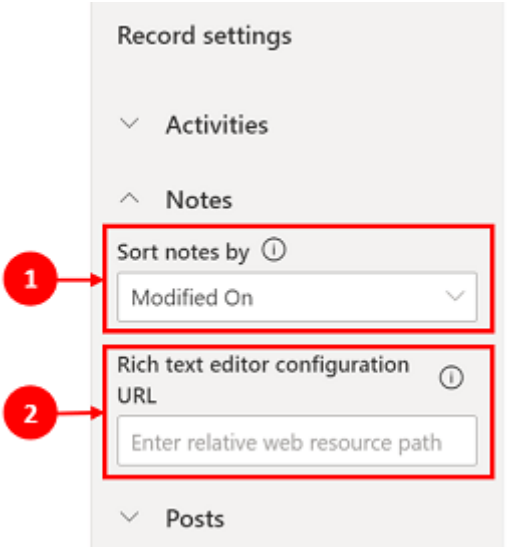
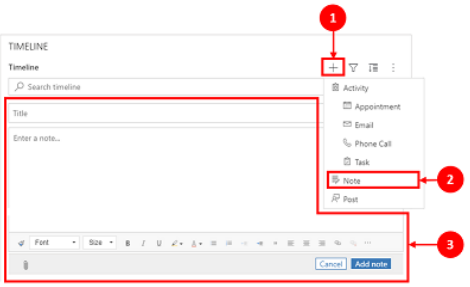
Timeline performance

If you select more than 10 **Activity types**, a warning message displays to indicate that the number of selected activity types impacts timeline performance. Limit activity types to 10 or fewer to improve performance.



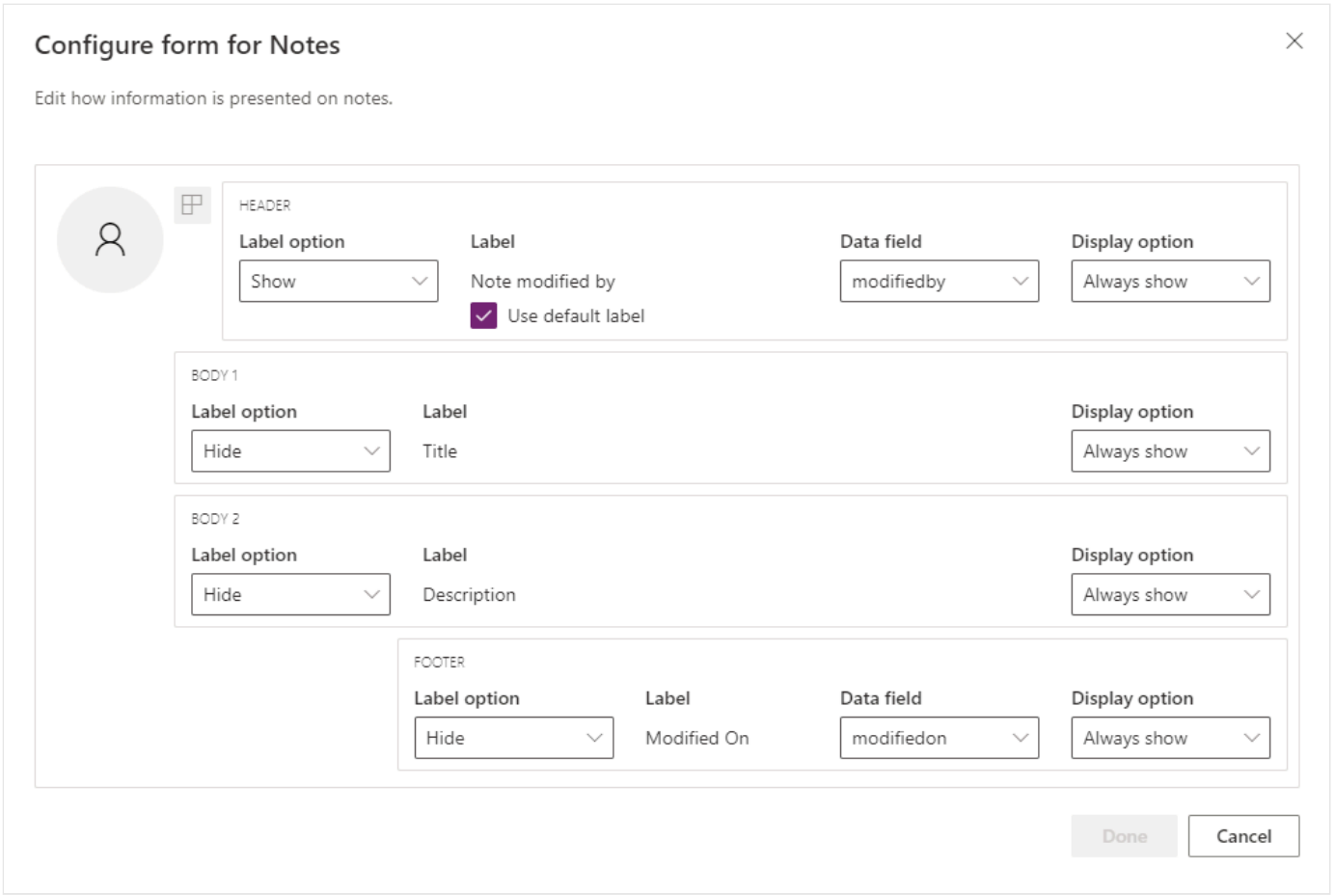
Notes on timeline

Expand table

Form designer configuration view	Form designer display view
	
The **Notes** section expands when enabled and allows you to: 1. **Sort notes by** date created or date modified. The **Modified On** date is the default setting. 2. Add a relative web resource path in the **Rich text editor configuration URL** column for customized note capability. Learn more in Add the rich text editor control to a model-driven app.	1. When enabled, Notes can be access via the **Create a timeline record** + icon. 2. A dropdown list appears where you can access **Notes**. 3. Use **Notes** to create a note to add to a record using rich text editing. You can set the font and size for the notes from the timeline, without having to set it multiple times.

Configure the form for notes

In the **Notes** area, under **Configure form**, select **Default form** to configure how information is displayed in notes, such as relevant users and dates, and whether or not to include labels. This setting lets you increase or reduce the number of timeline records that display.



- Header
  - Label option: Hide, show, or show on hover the label, "Created by" or "Modified by".
  - Label: Select the **Use default label** checkbox label to display the label "Note modified by". Clear the checkbox to display the label "Modified by".
  - Data column: Select to show either the user who created the note or the user who modified the note. The label changes to match the data column you selected.
  - Display option: Always show, show on expand, or hide this header containing the user who created or modified the note.
- Body1
  - Label option: Show or hide the label of the note.
  - Display option: Always show, show on expand, or hide the body text.
- Body2
  - Label option: Show or hide the label of the note.
  - Display option: Always show, show on expand, or hide the body text.
- Footer
  - Label option: Show or hide the label, "Created on", "Modified on", or "Overridden on".
  - Data column: Select to show the createdon, modifiedon, or overridenon date.
  - Display option: Always show, show on expand, this footer containing the createdon, modifiedon, or overridenon date.

Set the notes rollup type in timeline

> **Note**
>
> This is an early access feature. You can opt in early to enable these features in your environment, which allows you to test these features and then adopt them across your environments.

The rollup type can be configured for timelines on forms for notes. The available types of rollups are **Extended**, **Related**, and **None**. To show those records only that are directly related to the notes in timeline, select **None**.

Learn more in RollupType EnumType.

Allow users to quickly save notes on timeline

You can enable **Quick save**, which allows users to save their notes using **Ctrl**+**S** while typing a note on the timeline. Quick save helps improve user efficiency by not requiring users to navigate to **Save** or **Save and close** on the command bar. When users save a note using **Ctrl**+**S**, a **Save successful** confirmation message appears in the note on the timeline.

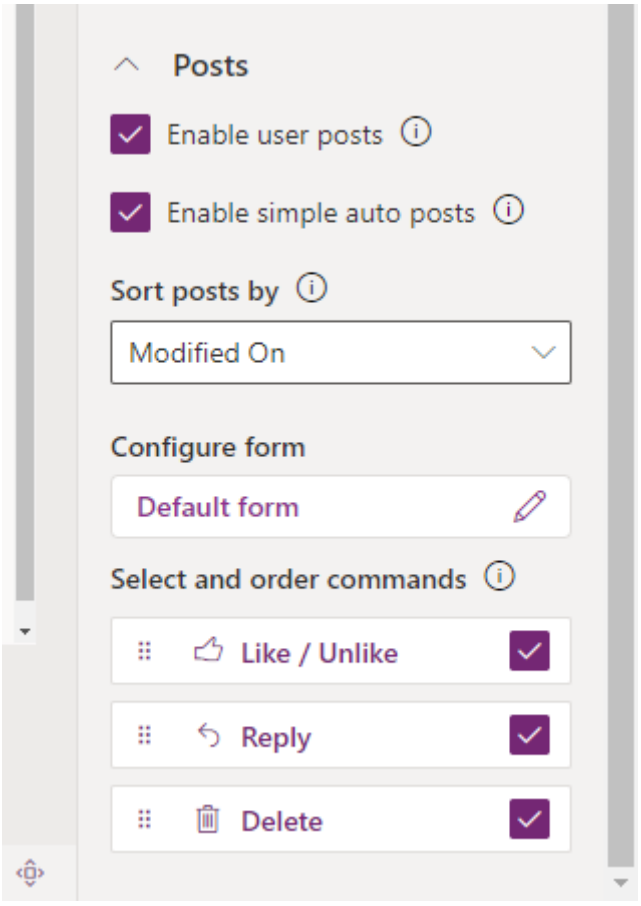
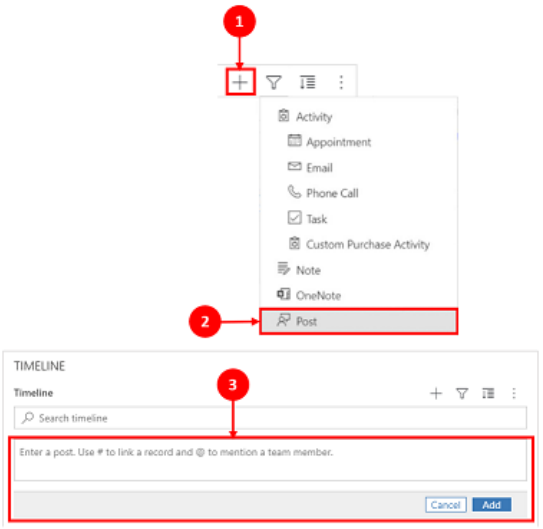
To enable **Quick save**, in Power Apps, go to the timeline settings, and then select the **Enable Quick save** checkbox.

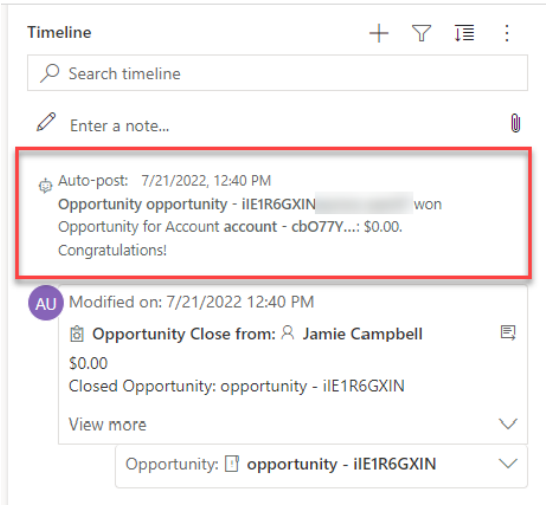
Posts on timeline

To enable rich text posts on the timeline, contact Microsoft Support.

> **Note**
>
> Posts are only available with certain Dynamics 365 apps, such as apps for Dynamics 365 Customer Service.

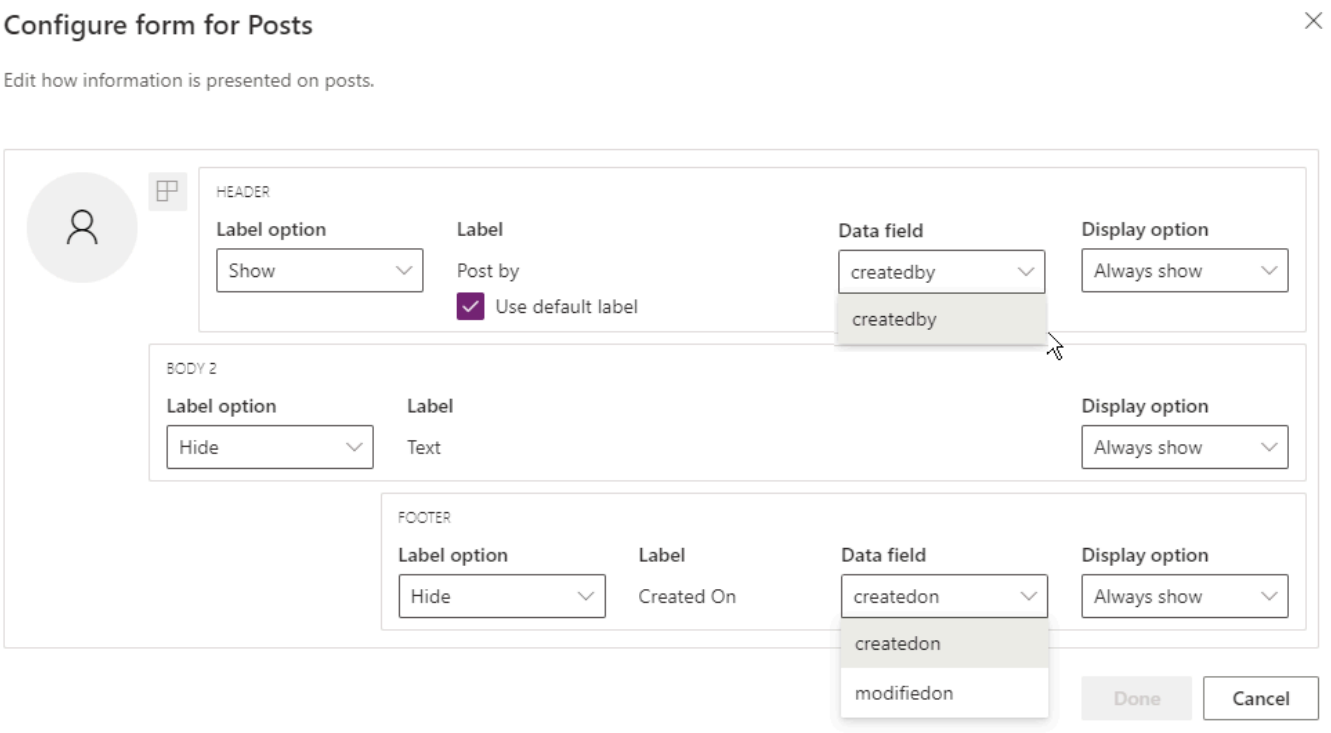
Expand table

Form designer configuration view	Form designer display view
	
The **Posts** section expands when enabled and allows you to: - Select **Enable user posts** enabling users to leave a message on the record. **Note**: You can't enable user posts in the Dynamics 365 Field Service mobile app. - **Sort posts by** date created or date modified. The **Created On** date is the default setting. - The **Select and order commands (Preview)** section allows users to display the command actions that can be performed on posts displayed on the timeline. You can also move the position of the commands to change the order in which they're displayed on the timeline.	1. When enabled, posts can be accessed by selecting **Create a timeline record** +. 2. A dropdown menu displays, and you can access **Posts**. 3. Use **Posts** to create a post to add to a record. When date **Created On** is used to sort posts on the timeline, the location in the timeline remains constant even when there are responses to that post. When date **Modified On** is used to sort posts on the timeline, the location in the timeline adjusts to the top when there are responses to that post. **NOTE**: The timeline doesn't automatically refresh when post replies are added.

Form designer configuration view	Form designer display view
The **Enable simple auto posts (Preview)** allows auto posts to be displayed either in the existing card format or as a simple post. Clear the checkbox to display the posts in the existing card format. The checkbox is selected by default.	
	When the **Enable simple auto posts (Preview)** checkbox is enabled, the posts aren't displayed in card form. Command actions can't be performed on auto posts from the timeline.

Configure the form for posts

You can configure how information is displayed in posts, such as relevant users and dates, and whether to include labels. This functionality enables you to increase or reduce the number of timeline records that appear onscreen.



Go to your timeline configuration in make.powerapps.com, scroll down to the **Posts** section containing the **Configure form** column, and edit the following column in the default form:

- Header
  - Label option: Hide, show, or show on hover the label, "Created by" or "Modified by".
  - Label: Select the **Use default label** checkbox label to display the label.
  - Data column: Select to show the user who created the post. The label changes to match the data column you selected.
  - Display option: Always show, show on expand, or hide this header containing the user who created the post.
- Body2
  - Label option: Show or hide the label of the post.
  - Display option: Always show, show on expand, or hide the body text.
- Footer
  - Label option: Show or hide the label, "Created on", "Modified on", or "Overridden on".
  - Data column: Select to show the createdon or modifiedon date.
  - Display option: Always show, show on expand, this footer containing the createdon or modifiedon date.

Configure @mentions in notes and posts on timeline

To enable @mentions in notes and posts, follow the instructions provided in this article: Use @mention for collaborating with your team using notes

> **Important**
>
> The @mentions feature doesn't work with Power Platform environments that were created with the **Enable Dynamics 365 apps** option set to **No**. Learn more in **Create an environment with a database**.
>
> Users must have the Read privilege for the **msdyn_postconfig** entity, and the Share privilege for the entity for which they use @mentions (for example, account, contact, case, and so forth).
>
> In-app notifications aren't supported for @mentions in posts on the timeline. Notifications are triggered only for @mentions in notes. If you need users to receive notifications, use @mentions in notes instead of posts.

Configure dashboard timelines

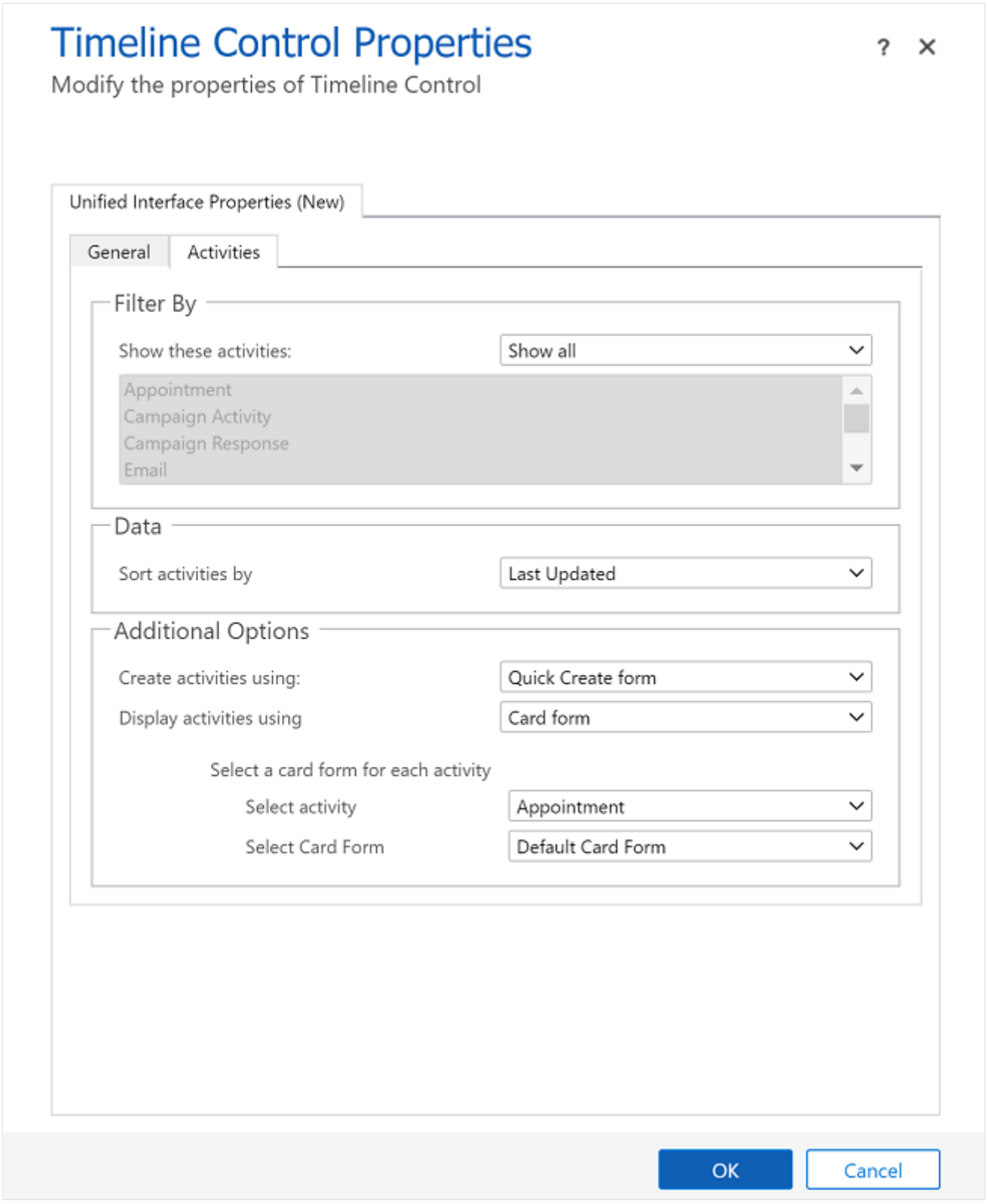
Timelines can be configured and put on a dashboard. However, the configuration available for timeline applied to a dashboard is limited to the functionality provided by the legacy designer experience. The new form designer experience on dashboards isn't available.

> **Note**
>
> Capabilities on a dashboard timeline are different than those in an embedded dashboard timeline.
> Dashboard timelines contain records related to the current user. This means each user reviews a different set of information for the same dashboard timeline. Notes are not available on the dashboard.

This following image shows the General tab, where you can modify the timeline control properties:



The following image shows the Activities tab, where you can modify the Timeline Control properties:



-->

Configure autopost messages to display on the timeline

> **Note**
>
> - The autopost functionality is only available with environments that are configured for **Enable Dynamics 365 apps**.

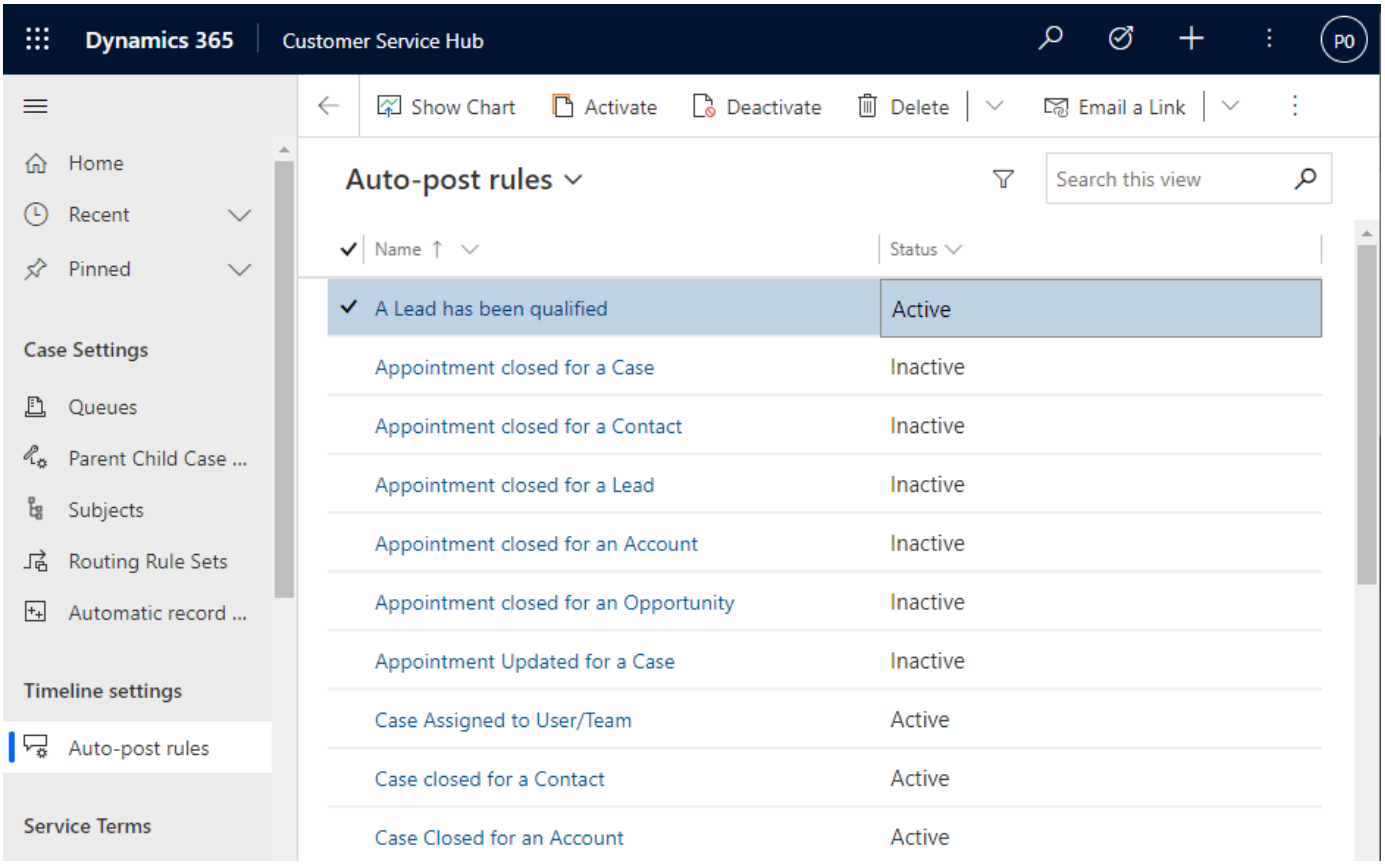
- Dynamics 365 includes autopost rules for some standard tables such as account, contact, lead, and case. It isn't possible to create new autopost rules. However, you can use other options such as Power Automate or a custom plug-in to create a post record based on the desired conditions.

You can configure which autopost messages appear on the timeline when a system event occurs. The autopost configuration replaces the legacy Activity Feed Configuration and Activity Feed Configuration Rules.

To configure the autopost messages that should be displayed:

1. In Customer Service Hub, go to **Service Management**, and under **Timeline settings**, select **Auto-post rules**.
2. Select which autopost rules to make active using the grid and **Activate** and **Deactivate** buttons at the top.

When a system event corresponding to an active rule occurs, an autopost message displays on the timeline.



If you're using a Dynamics 365 app other than Customer Service Hub or Customer Service workspace, you need to add your own sitemap. To add your own sitemap in your app:

1. Open your app in the app designer in **Power Apps**: make.powerapps.com

2. On the app designer command bar, select **Add page**, select **Table based view and form**, and then select **Next**.
3. In the **Table** list, select **Post Rule Configuration**, and then select **Add**.

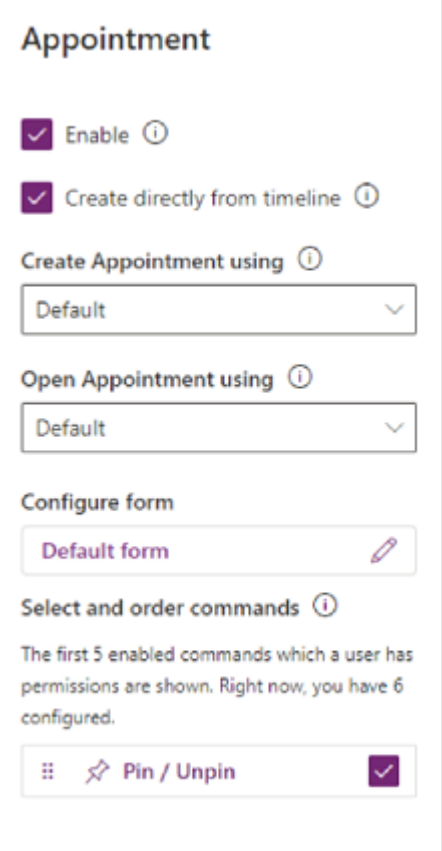
> **Note**
>
> If you don't have the Post Rule Configuration table, your environment doesn't have the required Dynamics 365 app installed.

4. Select **Save**, and then select **Publish**.

Let users pin and unpin activities to the top of timeline

You can configure the ability for users to pin or unpin activities, such as notes, to the top of the timeline so that they're easily accessible. Pinning activities helps users save time by not requiring them to scroll an entire timeline to find the information they need.

By default, notes are enabled for pinning. When you select an activity type such as a task, appointment, and so forth, the pin or unpin option appears in the list of commands. Select the **Pin / Unpin** checkbox for each activity you want users to be able to pin or unpin.



Enable modern Timeline control experience

The modern Timeline control experience provides a refreshed, visually consistent interface for activities and posts in model-driven apps. It improves readability and aligns with the latest Power Apps design patterns while preserving existing timeline functionality.

This feature is disabled by default and doesn't automatically replace the classic timeline experience.

Prerequisites

Before you enable the modern Timeline control experience, ensure that:

- You have the **System Administrator** or **System Customizer** security role.
- Your environment uses model-driven apps.

Enable the modern Timeline control experience

1. In Power Apps, select **Apps**.
2. Locate your model-driven app in the list.
3. Select **… (More commands)** next to the app, and then select **Edit**.
4. In the app designer command bar, select **Settings**.
5. In the **App settings** pane, select **Upcoming**.
6. Set the toggle to **Yes** for **Enable a modern Timeline control experience**.
7. Select **Save and Publish**.

After you enable the modern Timeline control experience, the updated experience appears the next time users open or refresh a form that includes a Timeline control.

Limitations

- Available only in model-driven apps.
- Some legacy customizations might not appear or behave exactly the same as in the classic timeline experience.

Create and add custom activities to timeline

You can create custom tables that display on a timeline. Learn more in Display a custom table in a timeline.

Configure blocked attachment file types

Power platform administrators can configure the file types that are blocked from being added as file attachments to records, including records in a timeline.

1. Find the **Set blocked file extensions for attachments** systems setting. Learn more in Open the System Settings dialog box General tab.
2. In the text box of blocked file attachments, type the file extension type you want to block for attachments (for example, ".pdf"). Separate file types with a semi-colon.
3. Select **OK**.

Set file size limits for attachments

Power platform administrators can restrict the file size of attachments users can upload in the rich text editor.

> **Note**
>
> File sizes for attachments generally can be any size under 90 MB, but for optimal system performance, smaller file sizes are recommended.

1. Go to **Advanced Settings**, and then on the Settings menu, select System > Administration.
2. On the **Administration page**, select **System Settings**.
3. On the **System Settings** dialog, select the **Email** tab, and then scroll down to find the **Set file size limit for attachments** value.
4. Enter the desired size limit for attachments, and then select **Save**. 131,072 KB is the maximum allowed value.

Enable and configure timeline highlights

Enable the timeline highlights widget to help users quickly get up to speed with the most important activities on a record. The widget uses Copilot to generate a list of significant activities on a record, summarized from notes, emails, tasks, appointments, phone calls, and conversations.

> **Important**
>
> This feature is only available for Power Platform environments with Dataverse that have the **Enable Dynamics 365 apps** setting enabled.

Before you set up copilot in your environment, review the FAQ about using AI responsibly in Power Apps.

Prerequisites

Copilot must be turned on in the settings of the model-driven app.

Enable timeline highlights

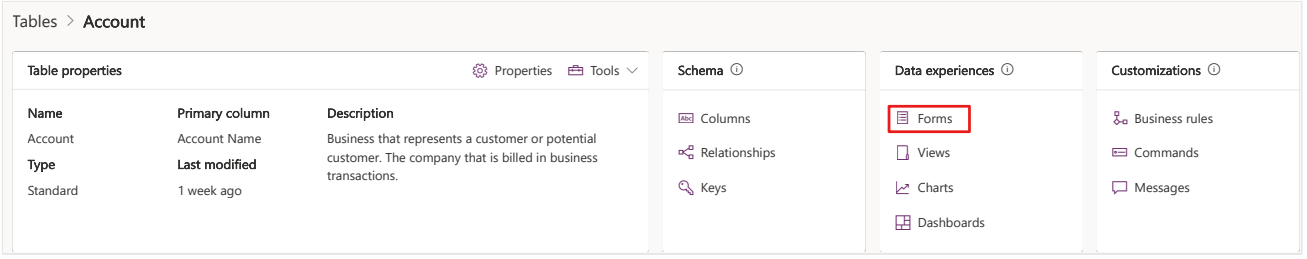
You can enable timeline highlights at the app level or form level, depending on whether you want to enable it for all entities/tables in an app or only for a specific form.

To enable timeline highlights at the app level, perform the following steps:

1. Go to Power Apps.
2. Select **Apps** and then select the app for which you want to enable the feature. For example, Sales Hub app.
3. Select **Settings** on the command bar and select **Upcoming**.
4. Turn on the **Enable Timeline Highlights** feature.
5. Save and publish the changes.

Enable timeline highlights at the form level

1. Go to Power Apps.
2. Select **Tables** and select the table that the form is based on. For example, select **Accounts** to enable timeline highlights for the account form.
3. Select **Forms** and then select the form for which you want to enable the feature.



4. Select **Enable Timeline Highlights powered by Generative AI**.
5. Save and publish the changes.

Known issues

Duplicate posts are shown when sorting records by older to newer

The API the timeline uses to fetch posts doesn't support sorting by older to newer posts. When you sort this way, the timeline receives the wrong posts to display. When sorting the timeline by older to newer, use notes instead.

Notes can only have one attachment added from the attachment button

This functionality is by design, with plans to allow more attachments soon. Multiple attachments can be added inline by dragging and dropping them into the note.

Timeline fails to load with error `code:"0x8004430d","message":"Number of link entity: <number> exceed limit 15`

There's a limit of 15 different tables that can be associated with a timeline. Either disable some of the activities associated with the timeline, or follow one of the workarounds described in this article: Timeline doesn't render and shows "Records couldn't be loaded"

When you create a note in a timeline, the character string "$&" is converted to "{3}amp;"

This issue is a configuration problem with the rich text editor control. To resolve this issue, add `"removePlugins": "stickystyles"` to your rich text editor config file. Learn more in Rich text editor properties.

Images in WebP format aren't displayed in the timeline

Images in WebP format don't display in timeline records. Use a supported image format such as JPEG or PNG.

Timeline records that contain unsupported image formats might not show inline images.

Related information

FAQs for timeline control

Use the timeline control

Last updated on 04/24/2026

Use custom connectors with the timeline control

Article • 01/14/2023

The custom connectors feature provides a way for developers to surface information such as Dataverse table rows, external data sources, and so forth, as records entries within the TimelineWallControl component. It allows for a broader set of scenarios in addition to the existing out-of-box notes, posts, and activities.

For information about configuring and using the timeline, see the following topics:

- Use timeline
- Configure timeline
- Timeline record card configuration

Records that are configured within custom connectors are a JavaScript web resource that conforms to the IRecordSource interface. The name of a web resource, along with the constructor (name including namespace), and optional JSON web resource path, can be added as a UClientRecordSourcesJSON parameter configuration within FormXML.

JSON

```
<UClientRecordSourcesJSON>
  {"recordSources": [{
    "name": "new_SecondaryRecordSource",
    "constructor": "SampleNamespace.SecondaryRecordSource"}]
  }
</UClientRecordSourcesJSON>
```

TimelineWallControl is expected to load the JavaScript web resource and then create the instance of IRecordSource from the configured constructor.

The IRecordSource is then initialized (init), with a request for pages of records (getRecordsData), request for filter details (getFilterDetails), and a request for the UX representation of a single record (getRecordUX).

The response from requesting records is persisted to minimize the number of record requests that occur within multi-session scenarios.

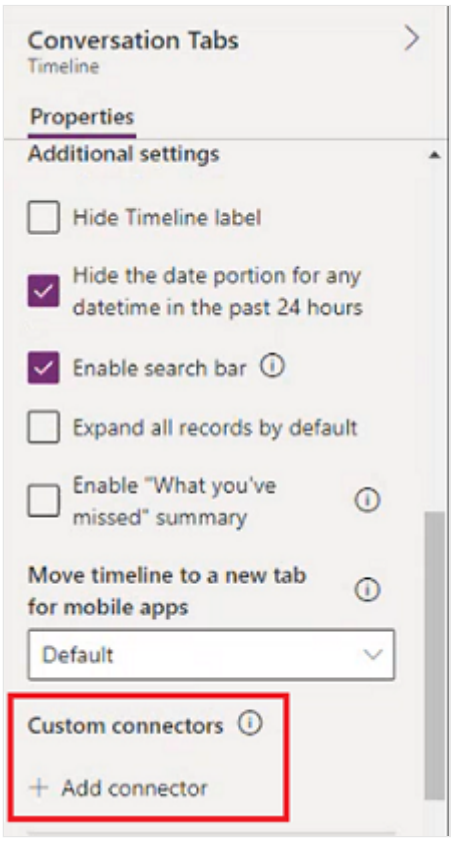
Configure the custom connector for the timeline control

You can add or edit your custom connector for the timeline control by configuring it in the maker experience.

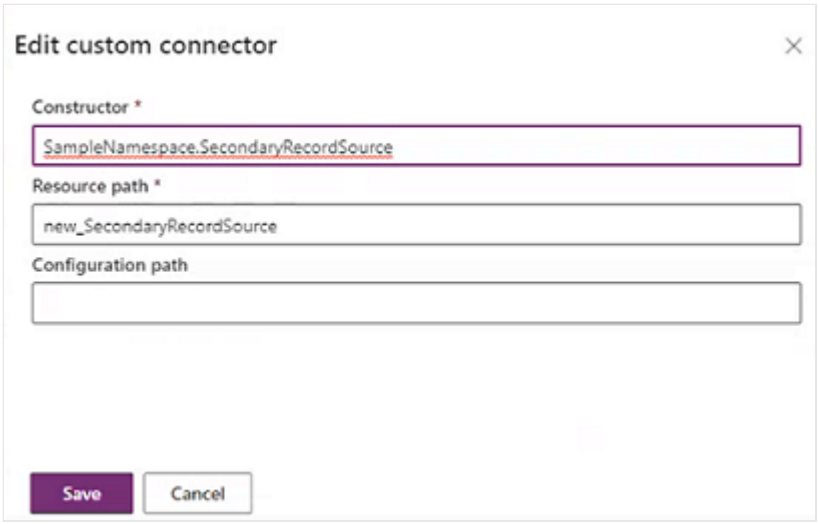
1. On the timeline control, select the form for which you want to add or edit the custom connector.

2. Complete one of the following options:

    - To add a new custom connector, in the **Properties** pane for the timeline table, in **Custom connectors**, select **Add connector**. The **New custom connector** pane is displayed.



    - To edit an existing custom connector you've already uploaded, in the **Properties** pane for the timeline entity, select the custom connector, and then select **Edit**. The **Edit custom connector pane** is displayed.



3. Edit the page, and then select **Save**.

Filter support

Custom connector filter support includes the following abilities.

- Add new filter type to existing filter group
- Add new filter group
- Update the existing filter type count
- Filter on search key

Custom connectors must use UserSettings Timezone information from context passed in init() to be consistent with timeline rendering.

Scope

The custom connectors feature is supported within single-session and multi-session table forms in Unified Interface client model apps.

Out of scope

The following functionality isn't available for the custom connectors feature:

- Offline and offline-by-default scenarios
- Dashboards
- Locations where TimelineWallControl isn't available (such as converged apps, canvas apps, Power Pages, custom pages, and so forth)

Known limitations

The configured web resources aren't formally declared as dependencies to the form. This means that exporting a form won't automatically export the web resources and they'd need to be added to that export manually. In addition, it can be easy to accidentally delete these web resources.

Develop a record source

When developing a record source, make sure to follow these practices:

- Ensure that you're retrieving data securely. The Unified Interface security model considers JS and JSON web resources as untrusted, and thus, such resources shouldn't contain tokens or secrets in them, as they'd be stored in plain text.
- If the data is within Dataverse, use the context object from init behavior to make requests into Dataverse. Calls into Dataverse from context have requests proxied through a secure iframe. This is how out-of-box record sources within TimelineWallControl retrieve data.
- If the data is outside of Dataverse, use existing mechanisms from the platform to retrieve external data.
- Locally test changes by using Fiddler: Improve the agility of development and debugging of JavaScript web resources by using Fiddler AutoResponder. More information: Script web resource development using Fiddler AutoResponder.
- Reduce the risk of XSS attacks: The risk of XSS attacks occurs when adding/binding HTML to the DOM. Use plain text whenever possible to reduce this risk. If HTML is required, you must sanitize this content before adding it to the record.
- Follow general best practices for client scripting. More information: Client scripting in model-driven apps
- Ensure inclusive design practices, including the usage of automated testing tools such as Accessibility Insights.

Solution sample

You can get started with a custom connectors solution sample that has the "SecondaryRecordSource" web resource, which is configured for "Account for Interactive Experiences" and "Account for Multisession Experiences" forms.

See also

Use timeline
Configure timeline

FAQs for the timeline control

07/14/2025

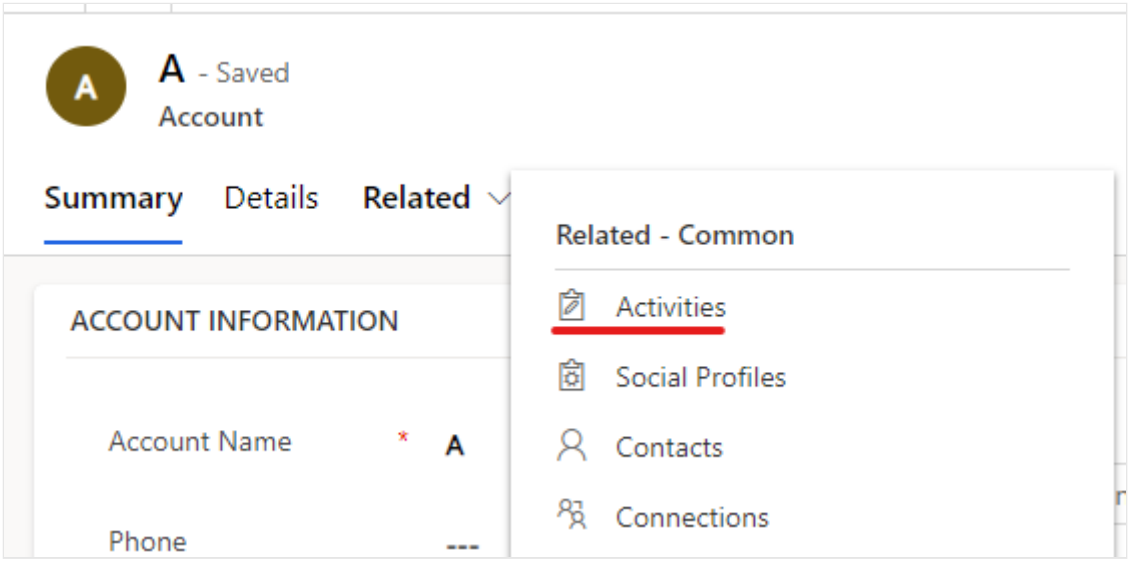
A command isn't working. How do I use the command checker tool for timeline?

For instructions about how to enable and use the command checker, go to Troubleshooting ribbon issues in Power Apps

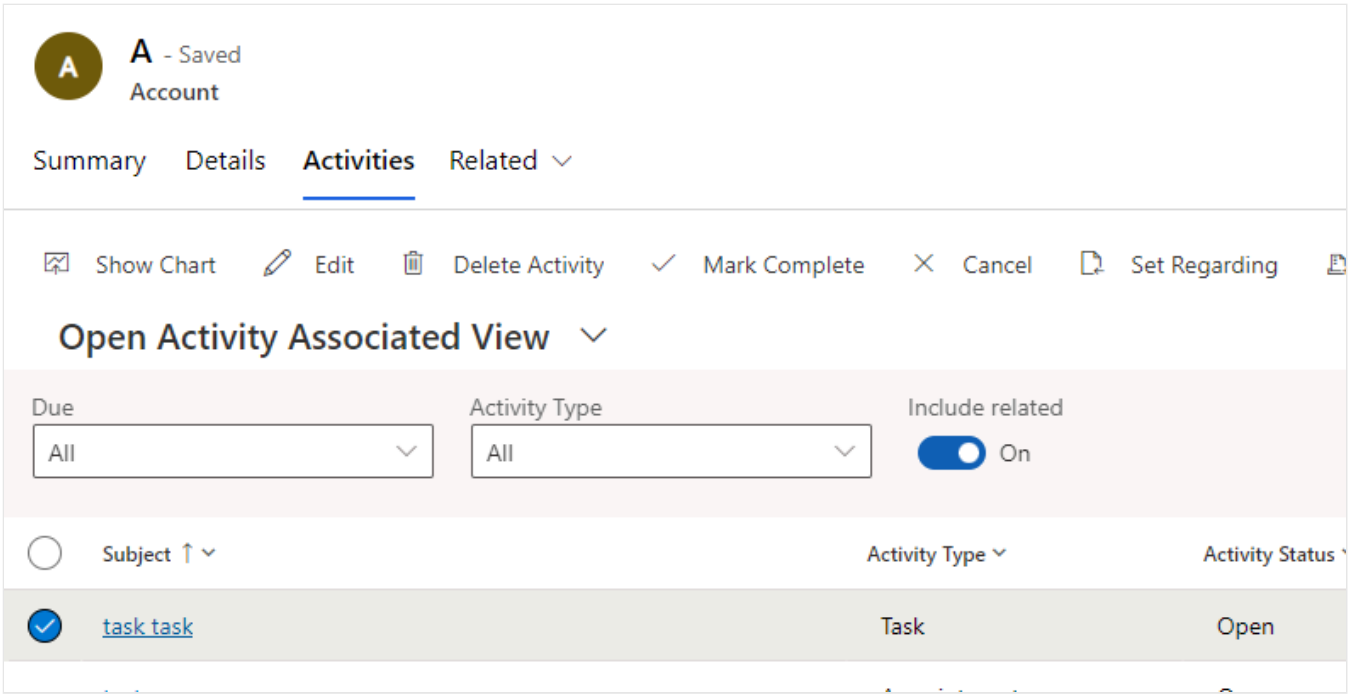
If you try to use the command action for activities on a timeline but the command action is hidden by custom rules, you might get the following error message:

"We can't compete the action you've selected."

You can check whether the action is hidden by navigating to **Related** > **Activities**.



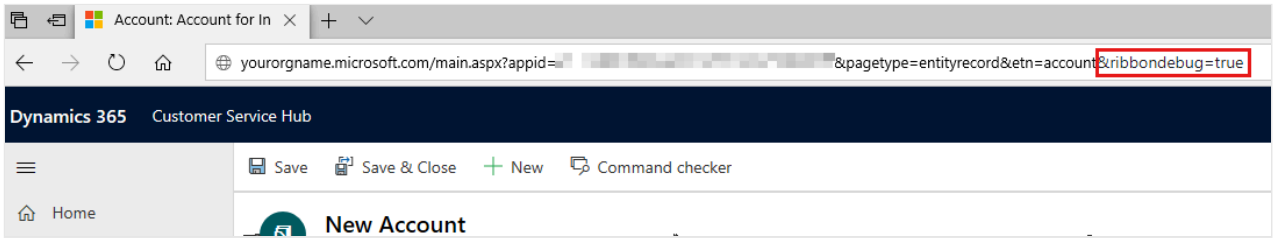
From the **Activities** tab, select the same activity that you want to perform the action on, and then check if you can perform the same command action from there.



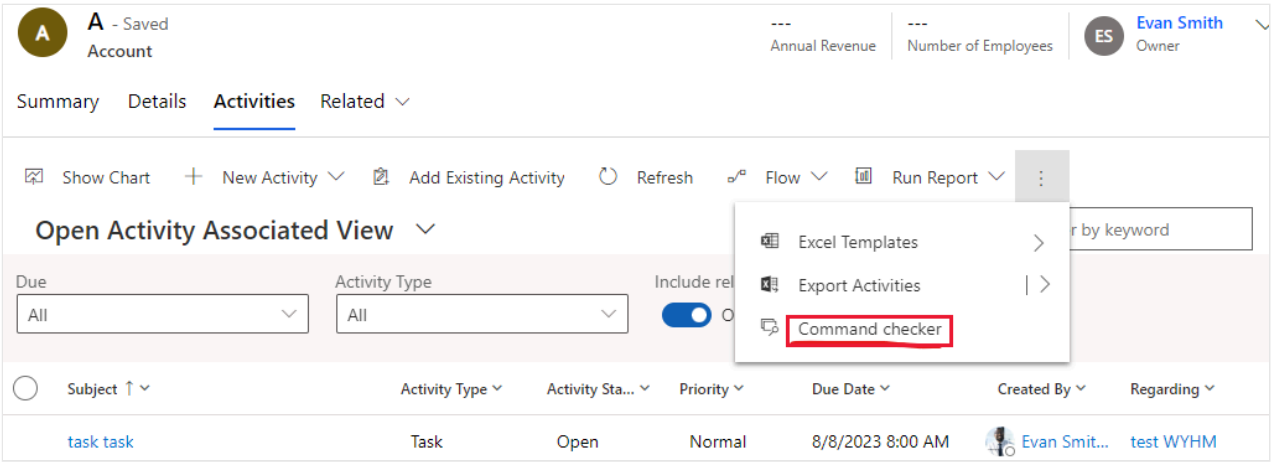
If the command action works here but not from the timeline, contact Microsoft Support.

If you can't find the command action from the related activity grid, you can further troubleshoot the issue with command checker to find out which custom rule hides the command.

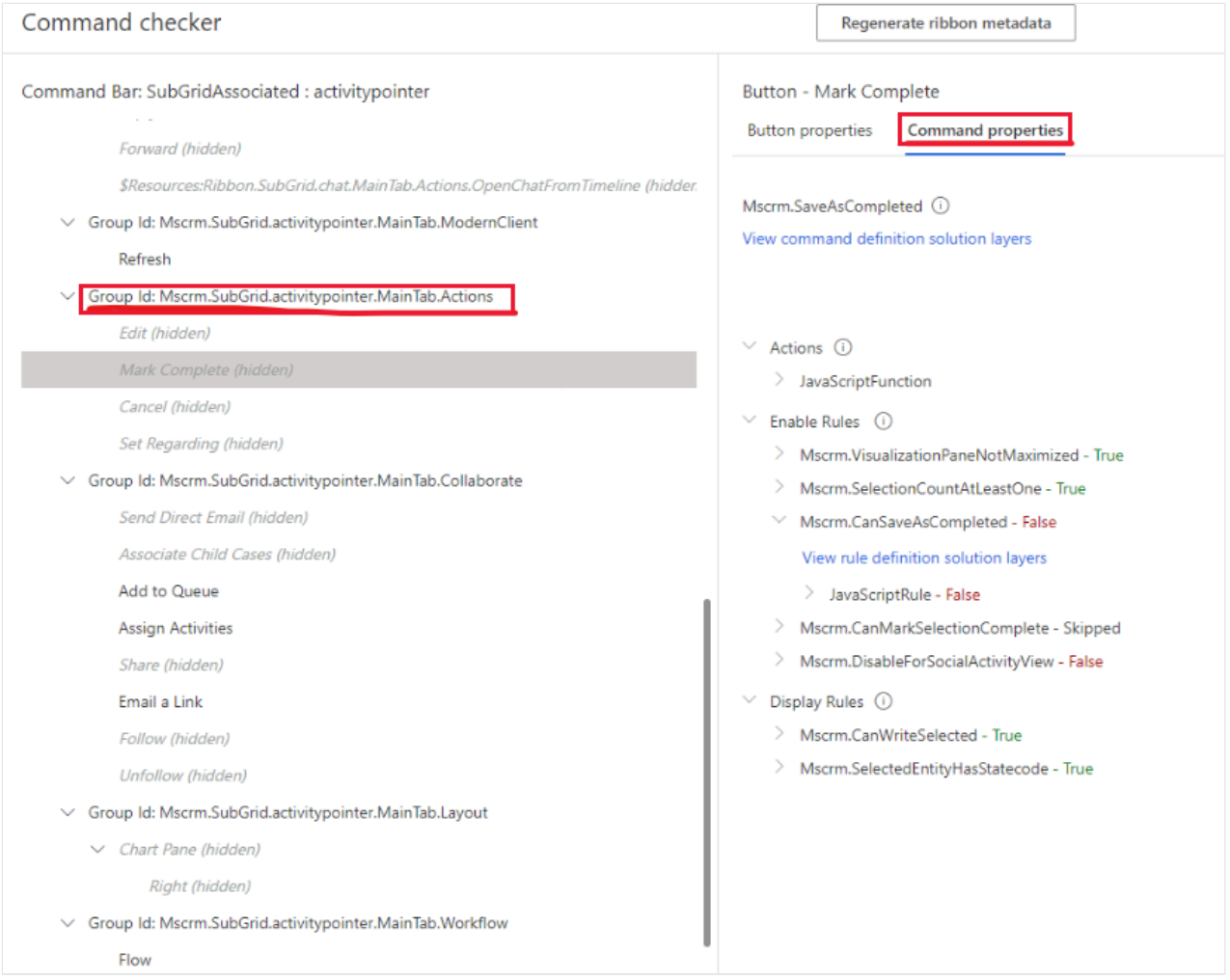
1. Play the model-driven app that has the timeline.
2. From a table record that displays the timeline, select **Related** > **Activities** to go to the **Open Activity Associated View**.
3. Enable the command checker tool by appending the `&ribbondebug=true` parameter to the app URL.



4. Select the **Command checker** command, which now appears on the app command bar. It might be listed on the **More** overflow flyout menu.



5. From the command checker page that is displayed, select **Group Id: Mscrm.SubGrid.activitypointer.MainTab.Actions**, select a hidden command such as **Mark Complete (hidden)**, and then select the **Command properties** tab to find out what custom rules are used to hide the command action.

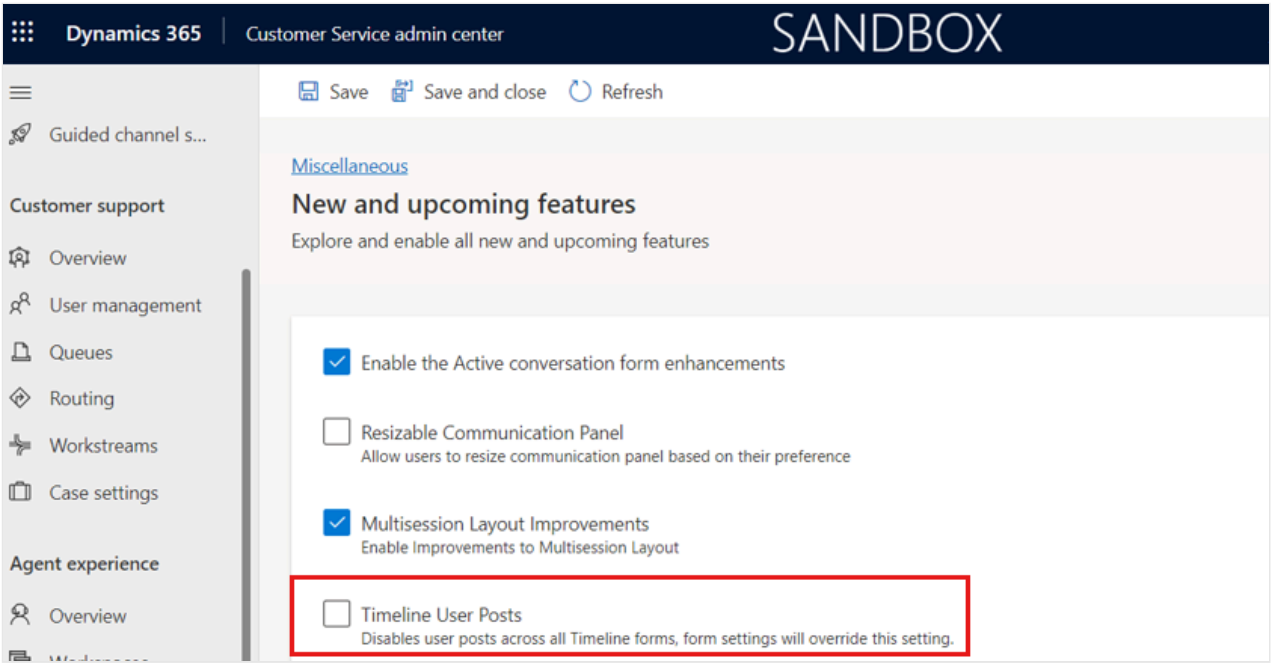


> **Note**
>
> The timeline always displays command actions without honoring the custom rules to hide or disable the command button for performance reasons.

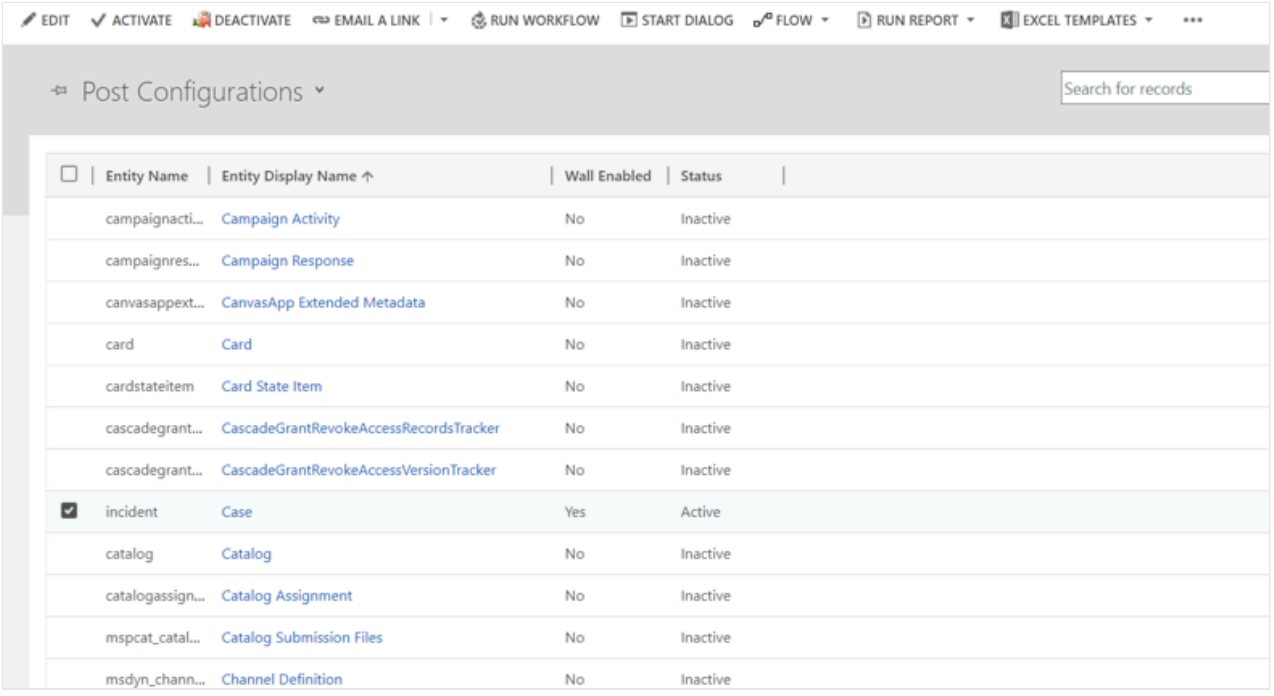
Why aren't posts appearing in timeline and why can't I create a new post?

Posts in the timeline are only available for Dynamics 365 app enabled Dataverse environments and posts are disabled with timelines by default for new environments. To enable, follow these steps:

1. Go to the Dynamics 365 Customer Service admin center > **Miscellaneous** > **New and upcoming features**, and then select **Timeline User Posts**.



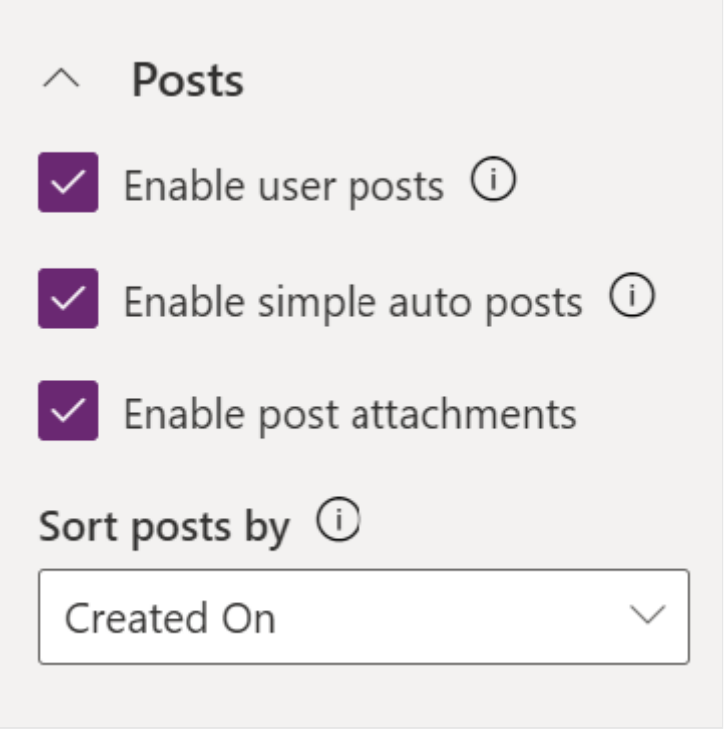
2. Check if the current table has the post configuration property activated. Go to **Settings** > **Activity Feeds configurations** > **Post Configurations**. In the following example, it's active for cases.



If the table isn't listed, look for it in the **Inactive Post Configurations** view. If the table you want isn't listed, select **Refresh** on the command bar. Select the table, and then select **Activate**.

3. Make sure posts are enabled from timeline configuration on the current form. More information: Record types to show

4. Verify the **Enable user posts** setting is enabled on the current form. If the checkbox is selected, clear it and select it again. Save and publish the changes from the form designer.



Do timeline records automatically refresh?

No. The timeline doesn't automatically refresh records. You must manually refresh the timeline to see updates made to a record. For example, if you update a record when you reply to an email, you must manually refresh the timeline to see the updated record.

Why do the "Modified On" columns of my records get changed when no changes are made to them?

The cascading behavior with the parent record can cause this scenario. For example, it can occur when a case is assigned to another user. Go to Table Relationships for information about how to configure this behavior.

Alternatively, you can configure timeline to sort notes by their **Created on** date. More information: Notes on timeline

What privileges do I need to use timeline?

- To open attachments, you need read privilege for the **Activity file attachment** table.
- For personal settings and bookmarks, you need read, write, and create privileges on the **Custom Control Extended Setting** table.

Power Platform admins can assign users these privileges. More information Security roles and privileges

How do I configure timeline for mobile offline?

For general guidance on enabling mobile offline for your app, go to Set up mobile offline.

> **Note**
>
> - The offline profile must have the **Note** and **User** tables in order for the timeline to work.
> - Only notes are available when offline.

How can I resolve issues with email tracking or auto capture?

Go to Auto Capture in Dynamics 365 sales for general information or go to the auto capture FAQ.

See also

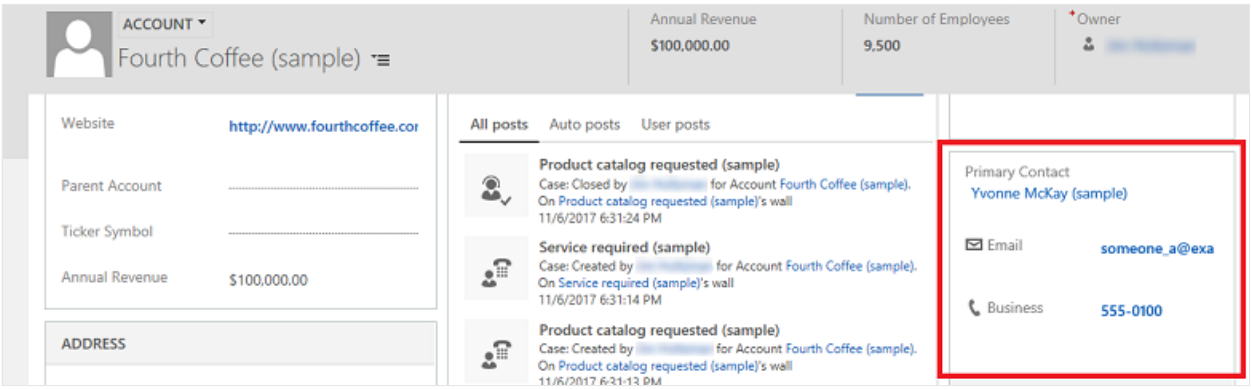
Set up timeline control

Use the timeline control

Model-driven app quick view control properties

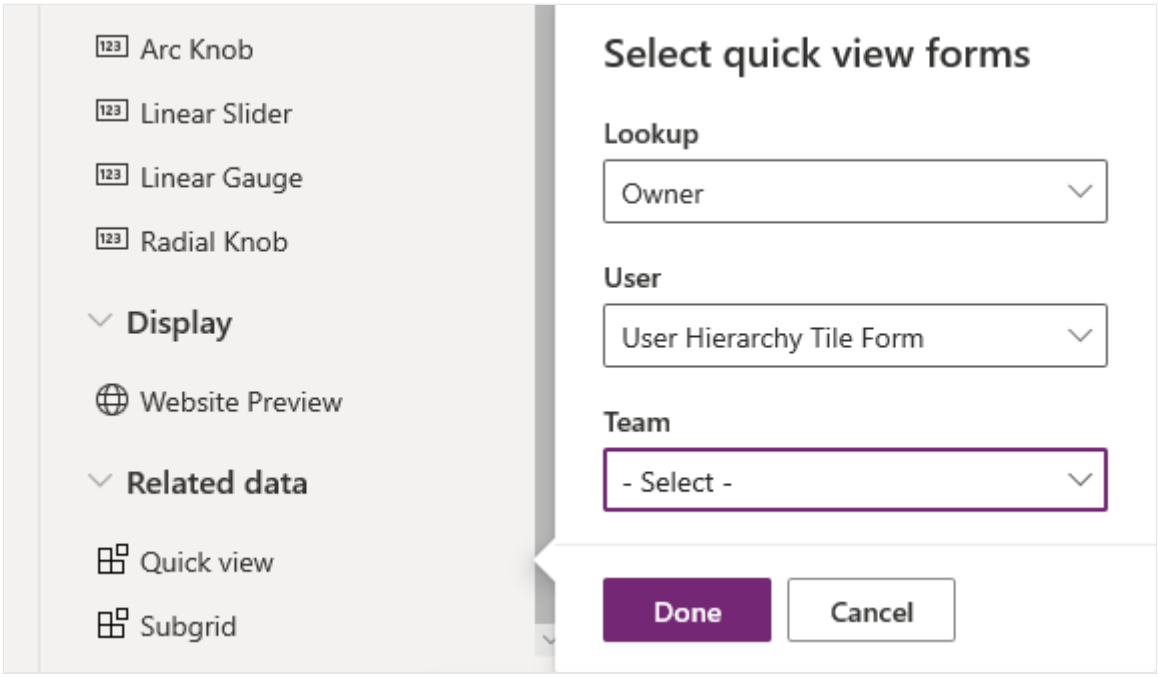
Article • 02/24/2023

A quick view control on a model-driven app form displays data from a row that is selected in a lookup on the form. The data displayed in the control is defined using a quick view form. The data displayed is not editable, but when the primary column is included in the quick view form, it becomes a link to open the related row. More information: Create and edit quick view forms



Add a quick view form

1. Sign in to Power Apps.
2. Select **Solution** from the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. Open the solution you want, open the table you want, and then select the **Forms** area.
4. Select a form, which **Type** is **Main**.
5. In the form designer, select **+ Component**, and then in the left navigation pane, select **Quick view**.
6. In the **Select quick view forms** dialog box, select a **Lookup** column included in the form, and then select a quick view form for the related tables. The related tables shown depend on the **Lookup** column chosen.



7. Select **Done** to close the **Select quick view forms** dialog box. The quick view form appears on the form, and the properties of the quick view appear in the properties pane.

Quick view form properties

Property	Description
Label	**Required**: A label to display for the quick view form.
Name	**Required**: The unique name for the quick view form that is used when referencing it in scripts.
Hide label	Displays the label on the form.
Quick view forms	Lists the quick view forms selected for the related tables.
Select forms	Select or change the selected quick view forms for the related tables. The related tables shown depend on the **Lookup** column chosen.
Components	Properties to configure for the component. A quick view control component has no properties to configure, and by default is shown whether someone is using a web browser, Dynamics 365 for phones, or Dynamics 365 for tablets.

Quick view control properties in classic form designer

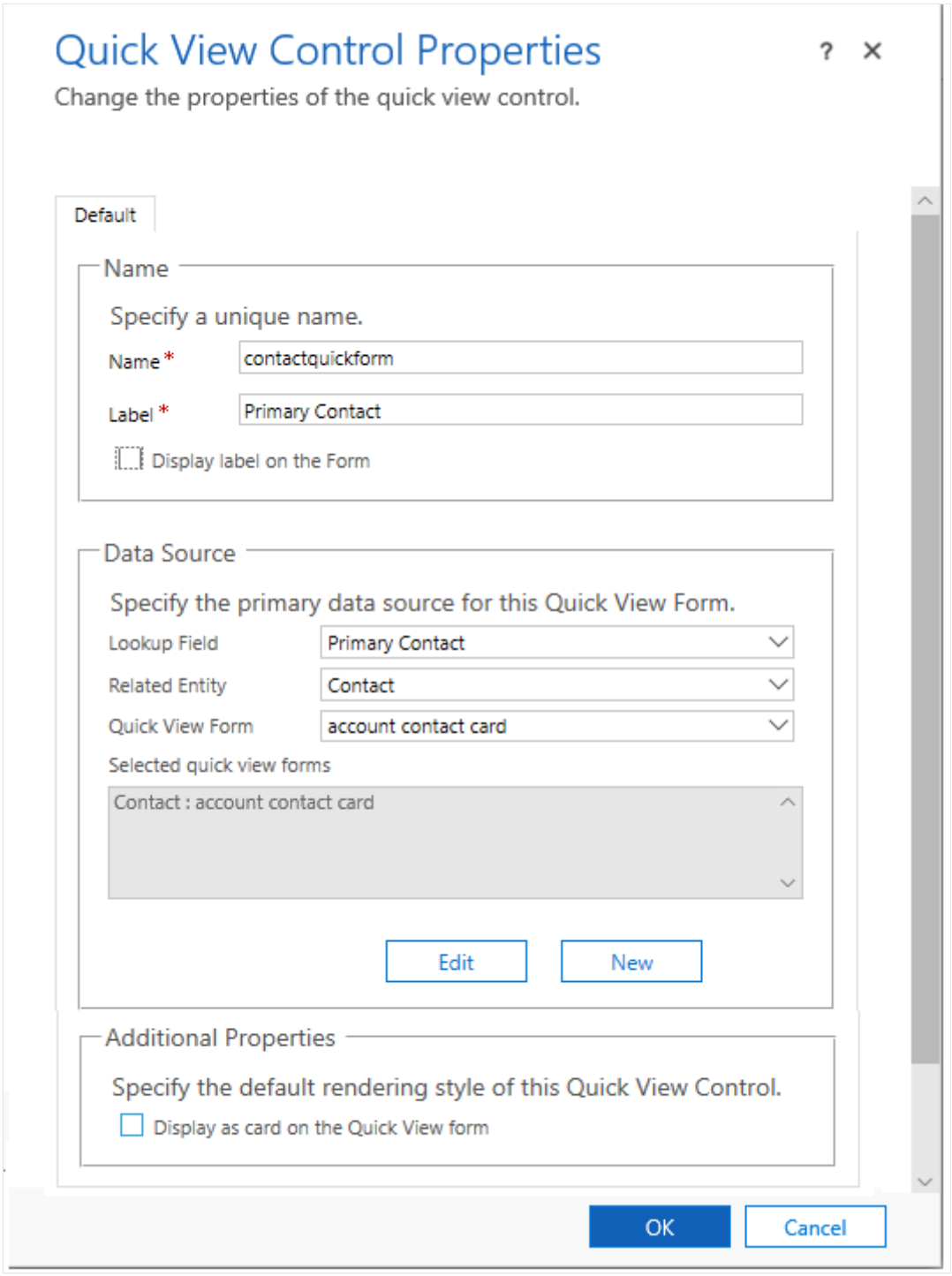
1. Sign in to Power Apps.

2. Expand **Data**, select **Tables**, select the table you want, and then select the **Forms** area.

3. In the list of forms, open the form of type **Main**.

4. On the command menu, select **Switch to classic**.

5. Then on the **Insert** tab, select **Quick View Form** to view quick view control properties.



Property	Description

Property	Description
Name	**Required**: The unique name for the quick view form that is used when referencing it in scripts.
Label	**Required**: A label to display for the quick view form.
Display label on the Form	Displays the label on the form.
Lookup Column	Choose one of the lookup columns included in the form.
Related table	This value depends on the **Lookup Column** chosen. It is usually the primary table for the 1:N table relationship for the lookup. If the table includes a **Potential Customer** lookup that can accept either an account or contact, in the **Quick View Form** column you can choose a quick view form for both account and contact by changing this value and then choosing another quick view form.
Quick View Form	If the **Related table** has any quick view forms these can be selected here. Otherwise, select **New** to create one. Select **Edit** to change the selected quick view form.
Additional Properties	The default rendering style can be specified by selecting the check box.

> **Note**
>
> When adding a multi-line text column to a quick view form, the form will have a height of one regardless of how the column control height is set. This ensures proper rendering of the form while maintaining density. Notice that multi-line text columns on other form types, such as main forms, work differently as the form automatically expands based on the amount of text.

Next steps

Use the Main form and its components

Model-driven app timer control overview (Classic)

Article • 02/23/2023

Use a timer control with forms where rows need to meet a specific **time-based milestone**. A timer control shows people how much time is available to complete an action in the resolution of an active row or how much time has passed since the time to complete the action has passed.

> **Note**
>
> Currently, this control can only be added and configured using the classic experience.

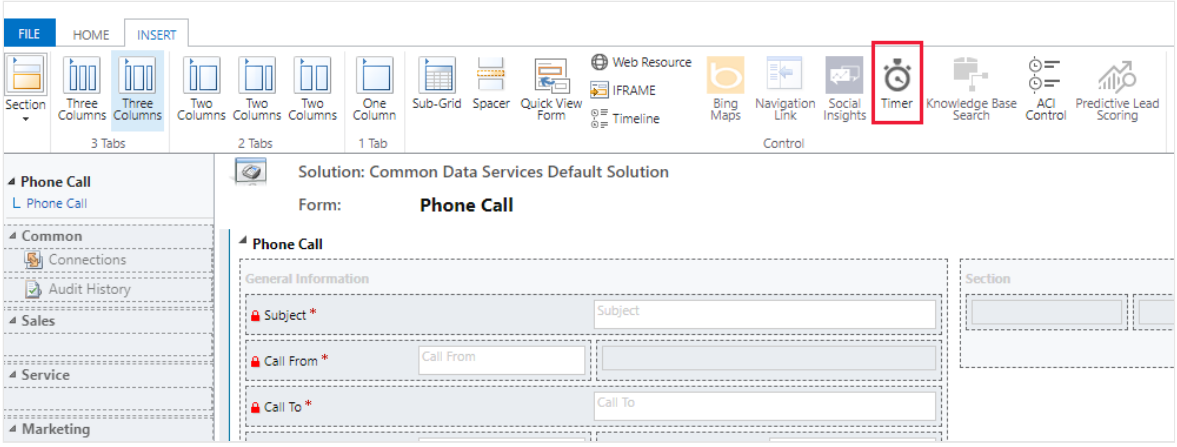
At a minimum, timer controls must be configured to show success or failure in completing the action. In addition, they can be configured to display warnings when the conditions are approaching failure.

How to add a timer control

A timer control can be added to a form for any table, but they are most frequently used for the case table, especially when linked to columns that track service level agreements. Multiple timer controls can be added in the body of a form. These cannot be added to the header.

To add a timer control to a form:

1. Sign in to Power Apps.
2. On the left navigation pane, select **Tables**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. Select a table, such as the account table, and then select the **Forms** area.
4. Select the form that you want.
5. In the form designer, on the command menu, select **Switch to classic**.
6. In the classic form designer, select the **Insert** tab, and then on the toolbar select **Timer**.



7. On the **Timer Control** properties page, enter or select the properties that you want, and then select **OK**.

Timer control **Data Source** properties use columns for the table.

- The **Failure Time Column** uses a date-time column to set the time.
- The three condition columns use one of the **Choice**, **Two Options**, **Status**, or **Status Reason** columns for the table.

Timer control properties

The following table describes the properties of a timer control.

Group	Name	Description
Name	Name	**Required**. A unique name for the control.
	Label	**Required**. The label to display for the timer control.
Data Source	Failure Time Field	**Required**. Choose one of the date-time columns for the table to represent when a milestone should be successfully completed.
	Success Condition	**Required**. Select a column for the table to evaluate the success of the milestone, then choose which option indicates success.
	Warning Condition	Select a column for the table to evaluate whether the success of the milestone is at risk so that a warning should be displayed, then choose which option indicates that a warning should be displayed.
	Cancel Condition	Select a column for the table to evaluate whether the achievement of the milestone should be cancelled, then choose which option indicates that the milestone is cancelled.

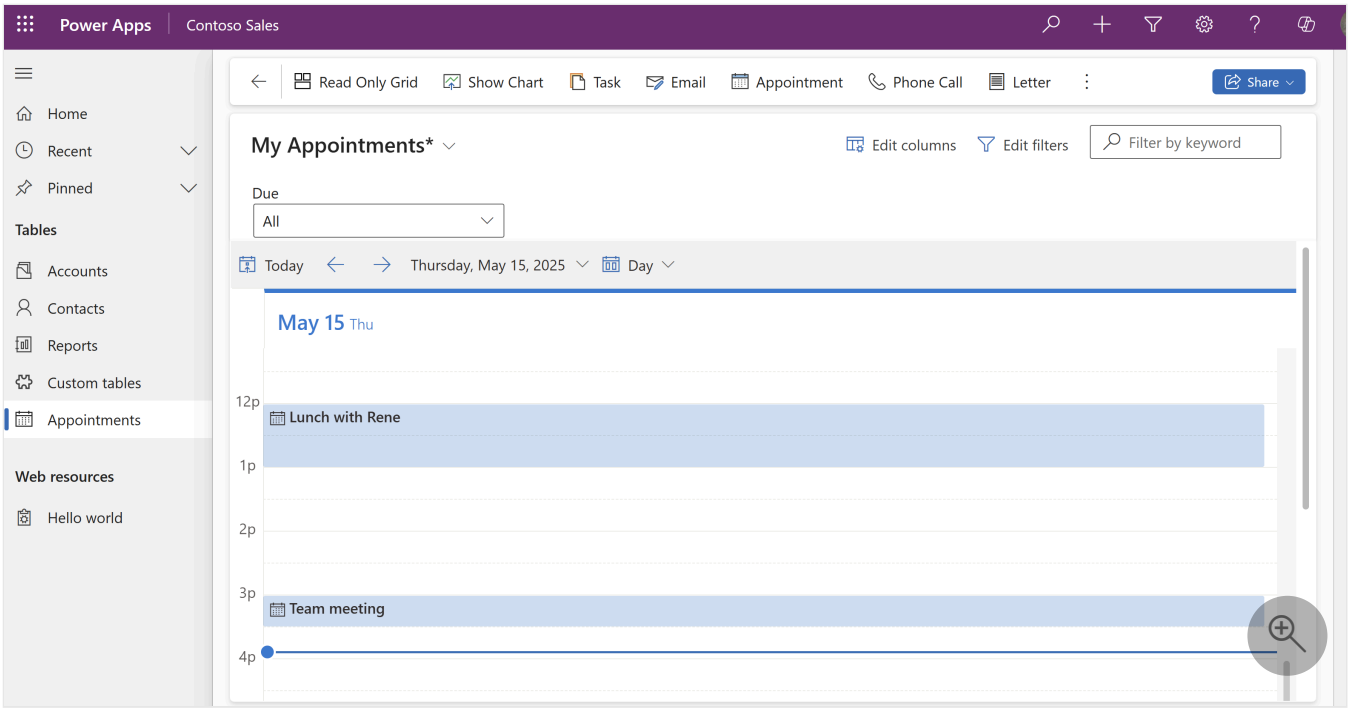
Next steps

Overview of the form editor user interface

Add the calendar control to tables

Article • 05/16/2025

The calendar control displays scheduled activities and their associated details in a calendar layout. Activities can be viewed, created, and deleted in a day, week, or month view.



System customizer or system administrator privileges are needed to add the calendar control to a table.

Add the control using the modern app designer

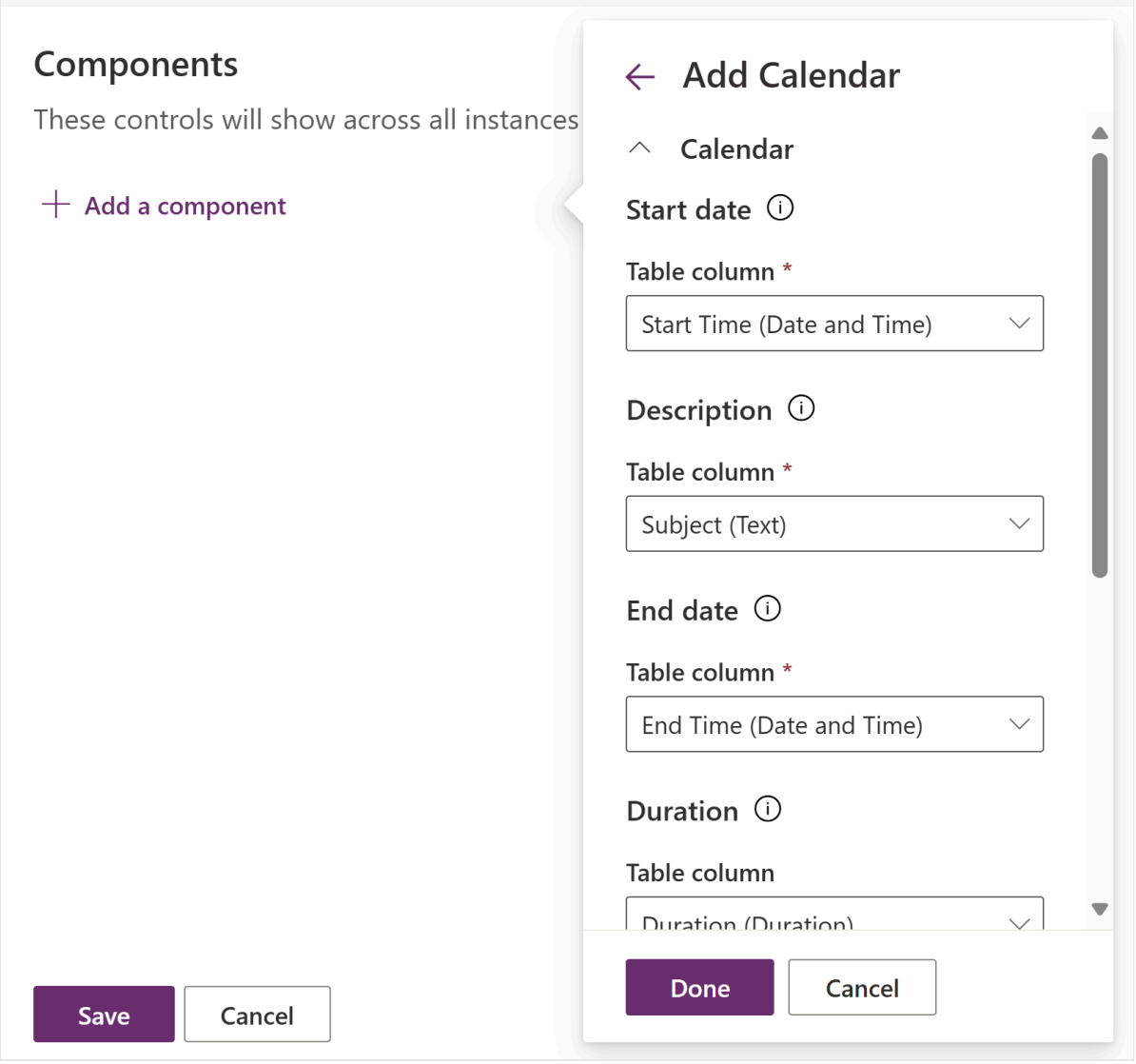
When you add the calendar control to a table, you can specify the columns that are used for the start date, end date, description, and duration. The calendar control is available in the modern app designer.

> **Note**
>
> When you add the control to a table view, it replaces the standard view for all views for that table.

1. While working in the app designer on the **Pages** tab, hover over the table view and then select **Edit view** (pencil icon).
2. Open the view where you want to replace the standard view with the calendar control. The view designer opens.
3. On the command bar select **Components**, and then select **Add a component**.
4. On the **Add a control** page, select **Calendar**.

5. On the **Add a calendar** properties page, select the table columns and clients to display the control. For example, with appointment table, you specify these columns.

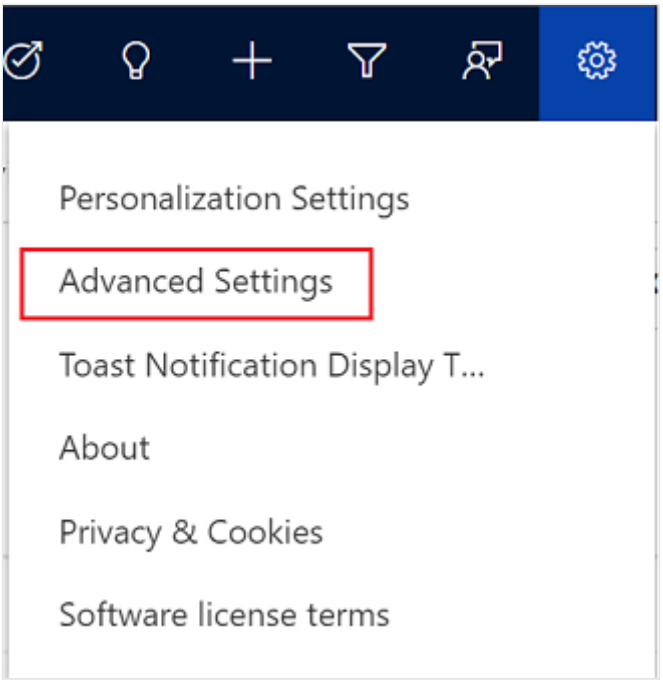
   - **Start date**. Select the column that is used for the start date on the calendar, such as **Start Time (Date and Time)**.
   - **Description**. Select the column that is used for the description, such as **Subject (Text)**.
   - **End date**. Select the column that is used for the end date on the calendar, such as **End Time (Date and Time)**.
   - **Duration**: Select the column that is used for the duration on the calendar, such as **Duration (Duration)**.



6. Select **Done**, and then select **Save**.
7. To make the updated view available in model-driven apps, select **Save and publish**.

Add the control using the classic solution explorer

1. In the app, select the **Settings** icon, and then select **Advanced Settings**.

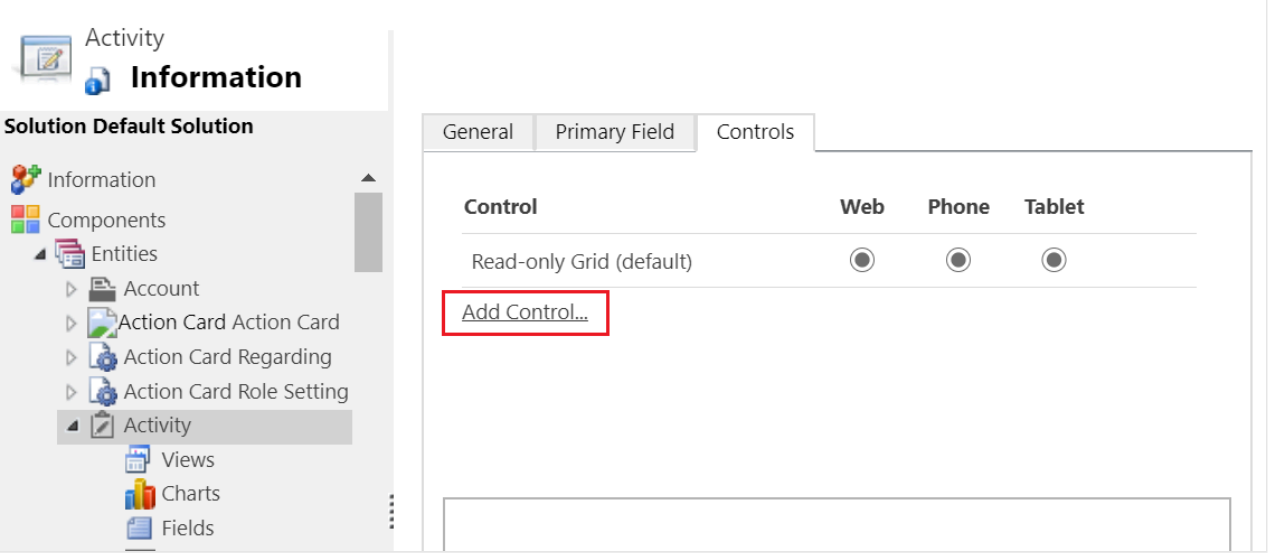


The **Business Management** page opens in a new browser tab.

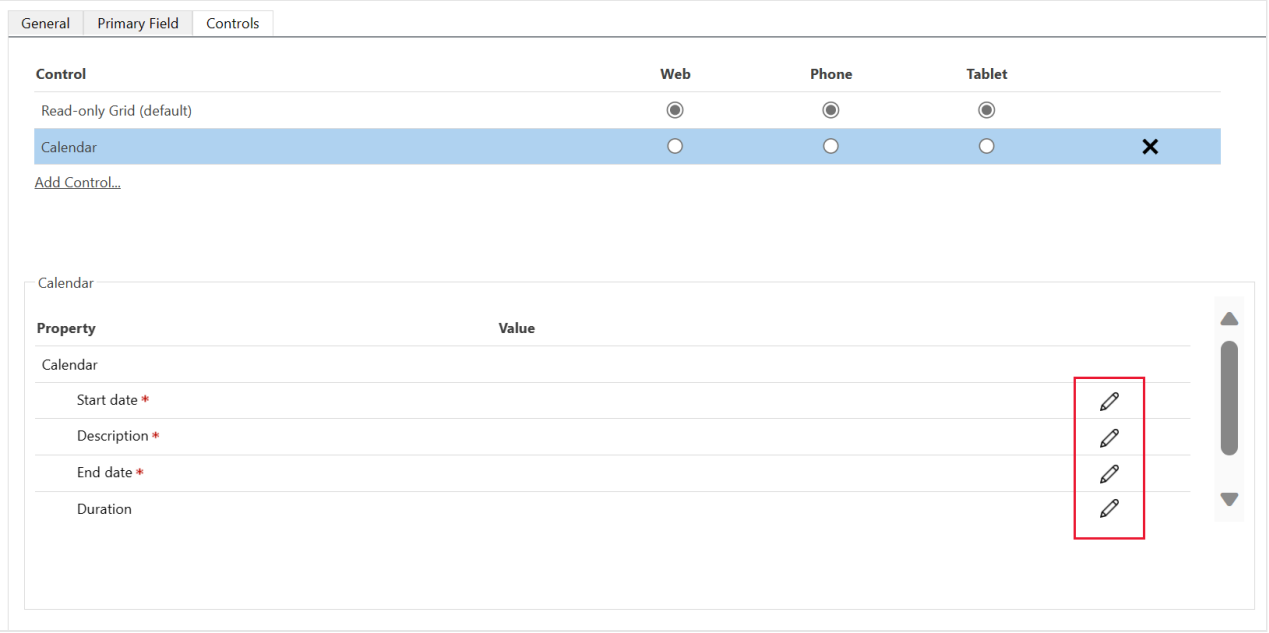
2. On the navigation bar, select **Settings**, and then under **Customization**, select **Customizations**.
3. Select **Customize the System**.
4. Under **Components** in the solution explorer, expand **Tables**, and then select a table. For example, **Activity**.

   > **Important**
   >
   > For **Description** and **Regarding** columns to be displayed in the Calendar Control, the table where the Calendar Control is configured must be created as an activity table. More information: Activity tables

5. On the **Controls** tab, select **Add Control**.



6. In the **Add Control** dialog box, select **Calendar**, and then select **Add**.
7. The calendar control is added to the list of controls.
8. **Read-only grid** is the default option, so when users select a table from the site map, they see a read-only grid of opportunities. To make the calendar view the default view instead, select the corresponding **Calendar** option buttons.
9. Select the edit icon for each of the mandatory columns indicated by a red asterisk, and then select the binding values.



10. Select **Save** to save changes.
11. To publish the changes, select **Publish**.

Mobile experience

The calendar* control is available when working on mobile devices with a different user experience that is optimized for mobile form factors. There's a scrollable day bar for selecting which date to view. The Day/Week/Month selection dropdown isn't available on mobile devices.

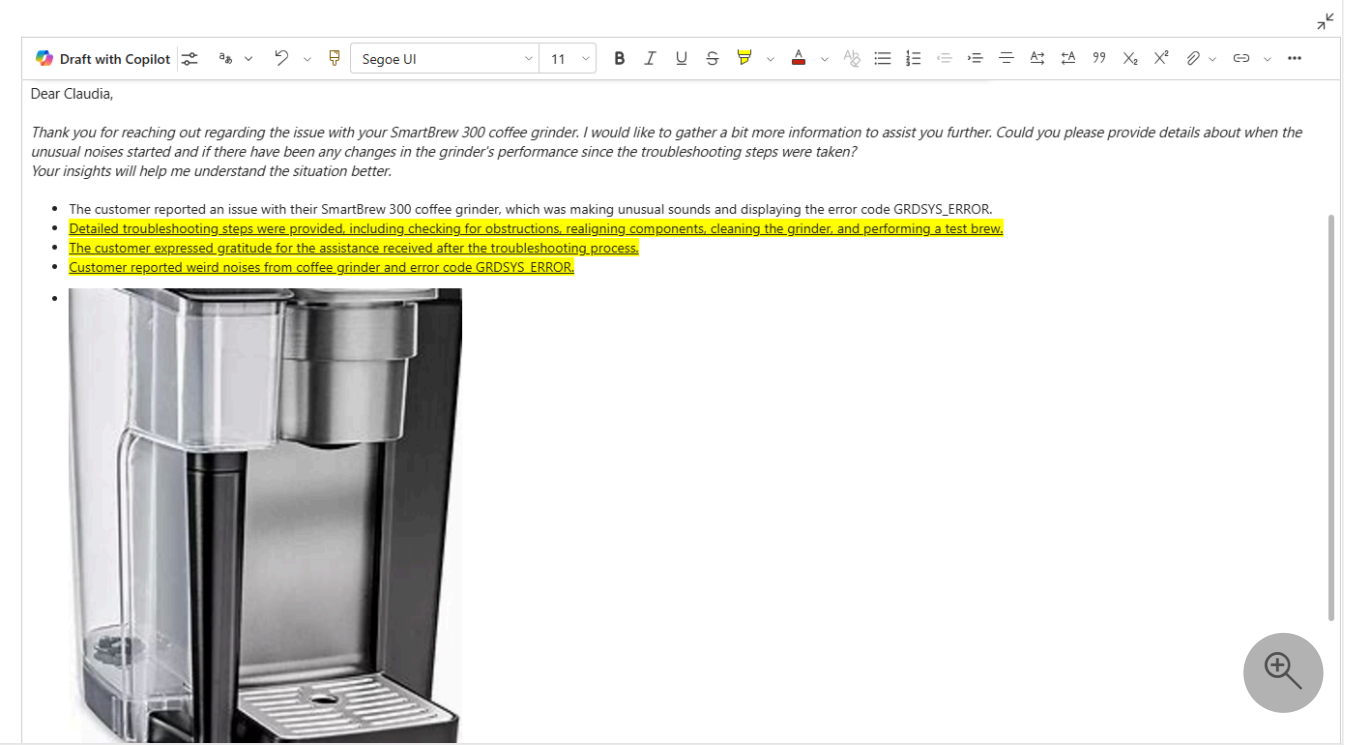
Related articles

Work with rows in the new calendar view

Add the rich text editor control to a model-driven app

The rich text editor control is a lightweight, HTML-based editor built on the popular CKEditor. It lets you create, paste, and edit formatted text in your model-driven apps. To format text in the editor, you can use the editor toolbar, insert HTML tags, or paste formatted text from other applications, like a web browser or Word.

You can customize the editor's appearance, features, and behavior.



Modern text editor enhancements

The modern text editor is designed to align with the familiar and intuitive interfaces of Microsoft applications such as Outlook, Word, and OneNote. This experience, set by default, introduces a modern design, dark mode, high contrast themes, and a new Copilot feature to enhance your text editing capabilities.

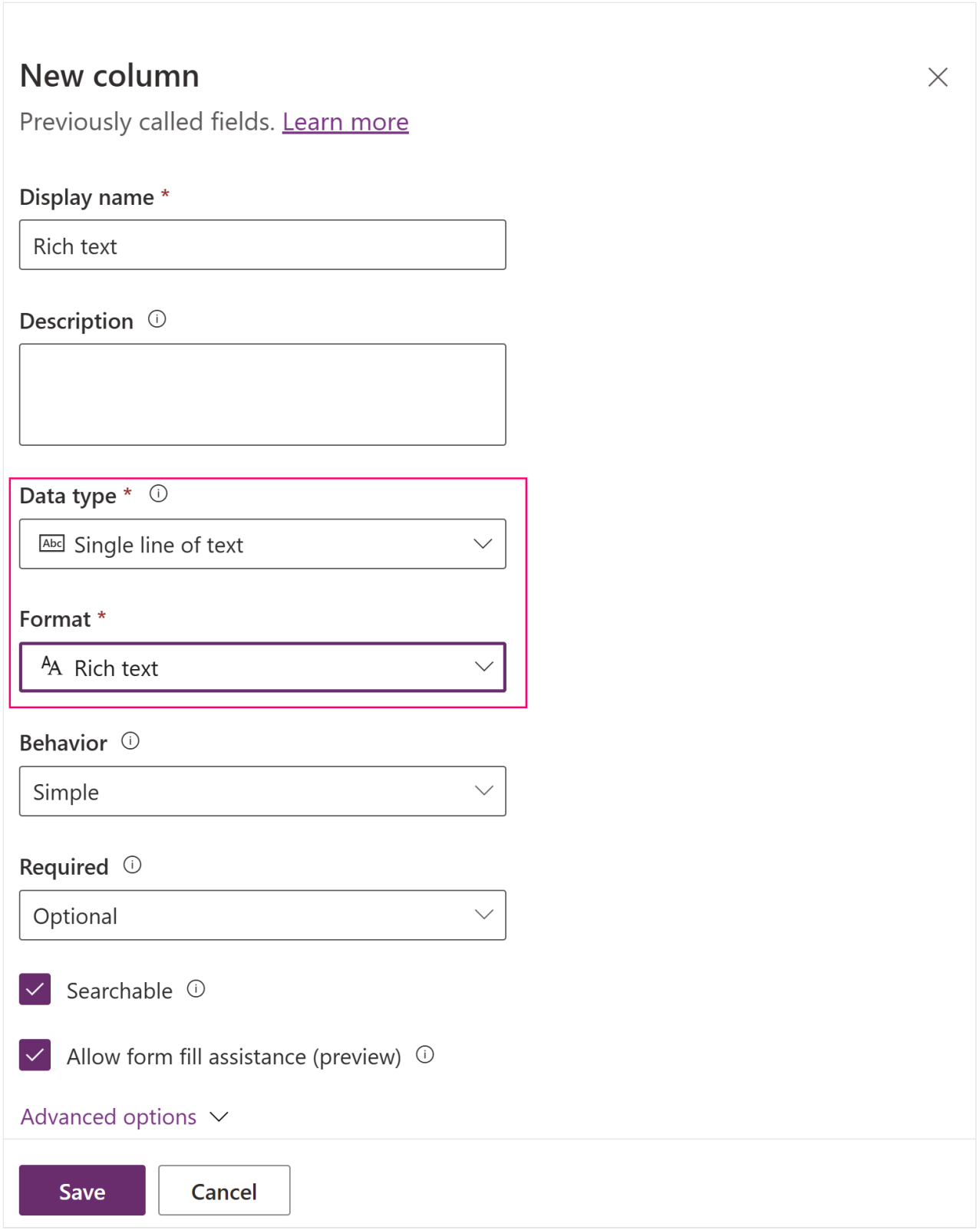
> **Note**
>
> - The classic version of the rich text editor was deprecated as of April 2025. If you have issues with upgrading or using the modern rich text editor, contact Microsoft Customer Support.

- If you customized the previous version of the rich text editor, you might need to recreate your customizations in the modern rich text editor.

Add the rich text editor control to a text column on a form

1. Sign in to Power Apps.
2. In the left navigation pane, select **Solutions**.
3. Open a solution and then open a table in the solution.
4. In the **Data experiences** area, select **Forms**, and then open the form that contains the text column you want to add the rich text editor to.
5. On the form designer canvas, select an existing text column. Alternatively, select **New table column** to create a new text column, enter a **Display name**, select **Text** > **Rich text**

for the **Data type**, and then select **Save**.



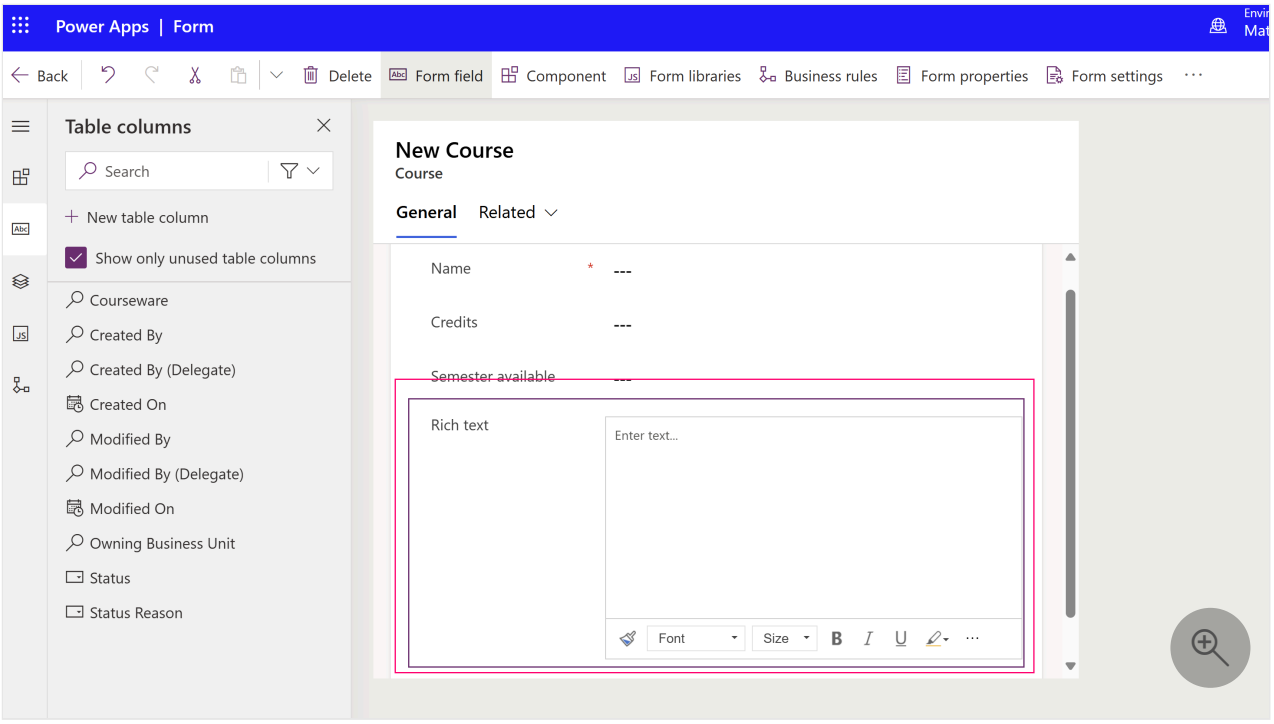
> **Note**
>
> If you want to add the rich text editor control to a column that already exists, you must first change the column's data type to **Text** (if not already a text column) and then select the **Format** using the **Rich text** option.

6. In the column **Properties** pane, expand the **Components** list, select **+ Component**, and then select **Rich Text Editor Control**.

7. In the **Add Rich Text Editor Control** pane, select **Web**, **Phone**, and **Tablet** to allow apps running on any device to use the editor.

   If you want to customize the editor, enter the relative URL of its configuration file, a JavaScript web resource that contains the properties you want to change, in the **Static value** box. If you leave this field empty, the editor uses its default configuration.

8. Select **Done**. The rich text column is added to the form.



9. Select **Save and publish** to apply your changes to the form.

Add the rich text editor control to a text column

You can also create a column for a table that uses the rich text editor control. The following steps show you how to create a text column that you can later add to a form for a model-driven app.

1. Sign in to Power Apps.

2. In the left navigation pane, select **Solutions**.

3. Open a solution and a table in the solution.

4. In the **Schema** area, select **Columns**, and then open a text column.

If the table doesn't contain a text column, select **New column** and enter a name for the column.

5. In the right pane, select **Format**, and then select **Rich text**.

6. **Save** the column.

Customize the rich text editor control

The following customizations are available in the classic rich text editor experience.

Power Apps allows you to change the properties of the rich text editor control to customize its appearance, features, and behavior. To customize a specific instance of the control, specify properties and their values in an individual JSON-formatted configuration file. To customize the control's global configuration, change the properties in the default configuration file.

Levels of customization

You can apply up to three levels, or layers, of configuration to customize the rich text editor:

1. At the most fundamental level, every instance of the control takes its configuration from the file `RTEGlobalConfiguration_Readonly.json`. The file is read-only, so you can't change these properties directly.

   > **Note**
   >
   > `RTEGlobalConfiguration.json` doesn't apply to email, knowledge articles, email templates, and signatures.

2. At the next level, every instance of the control takes its configuration from the properties in the file `RTEGlobalConfiguration.json`, if any are present. This configuration is layered on top of the previous one, so the properties in this file *replace* the same named properties in the read-only file.

3. Finally, at the highest level, a specific instance of the control takes its configuration from a specific configuration file, if one exists. This configuration is layered on top of the previous one, so the properties in this file *replace* the same named properties in the two lower-level files.

The system doesn't replace *all* properties with those in a higher-level configuration. The `extraPlugins` properties merge to allow the use of a wide range of external and out-of-the-box plug-ins in the default configuration. This merge operation lets you activate and deactivate plug-ins as needed in the configuration file for specific instances of the control.

Customize a specific instance of the rich text editor

1. In Visual Studio Code or other text editor, create a file and give it a meaningful name.

   The file `RTEGlobalConfiguration.json` contains the rich text editor's default, or global, configuration. If you're customizing the control in, say, a contact form, you might name the file something like `RTEContactFormConfiguration.json`.

2. Copy and paste the following code snippet into the file:

JSONCopy

```
{
"defaultSupportedProps": {
  "propertyName": "value",
  "propertyName": "value",
  "propertyName": "value"
},
"propertyName": "value",
"propertyName": "value",
}
```

The last *propertyName:value* pair doesn't end with a comma.

3. Replace *propertyName* and *value* with the rich text editor control properties you want to change. String values must be enclosed in quotation marks.

   We provided a few example configurations for you, but you can define others to suit your needs.

4. In Power Apps, create a **JavaScript (JS)** type web resource using the JSON file you created in step 1.

5. Add the rich text editor control to a text column on a form, and in the **Add Rich Text Editor Control** pane > **Static value**, enter the relative URL of the JavaScript web resource.

   Although you can enter the absolute URL of the web resource, we recommend that you enter the relative URL. That way, the web resource still works if you import it as a solution into a different environment, provided the relative path is the same.

For example, if the URL of the web resource is `https://yourorg.crm.dynamics.com/WebResources/rtecontactform`, the relative URL is */WebResources/rtecontactform*.

6. Select **Done**.

7. Select **Save and publish** to apply your changes to the form.

Customize the rich text editor globally

The editable global configuration file `RTEGlobalConfiguration.json` is the second level of customization you can apply. By default, the file is empty. That means that all instances of the rich text editor use the properties in the read-only file `RTEGlobalConfiguration_Readonly.json`, unless you create a configuration file for a specific instance of the control.

If your business needs require you to customize the control everywhere, specify the values you need in the `RTEGlobalConfiguration.json` file. Use the read-only file as an example to make sure you enter property-value pairs in the right format.

Use Copilot to refine text in the editor

Copilot uses natural language processing algorithms to help improve written content. You can add Copilot to the rich text editor and then get suggestions to improve the grammar, clarity, or overall quality of your text. Learn how to use Copilot in the rich text editor.

Rich text editor properties

The JSON file that determines the "look and feel" of an instance of the rich text editor contains two sets of properties:

- The defaultSupportedProps section contains properties of plug-ins that determine what the control can do. You're not limited to the properties of CKEditor and its plug-ins. You can also set values for properties of plug-ins that you add or create.
- The individual properties section contains properties that determine what the control looks like.

Example configuration file

The following code is an example of a JSON file that contains both defaultSupportedProps properties and individual configuration properties. If a property has a default value, the default

value is shown. If a property doesn't have a default value, a sample value is shown to illustrate the syntax. The properties are described in the two tables that follow.

JSON


```
{
    "defaultSupportedProps": {
        "height": 185,
        "stickyStyle": {
            "font-size": "12pt",
            "font-family": "'Segoe UI','Helvetica Neue',sans-serif"
        },
        "enterMode": 1,
        "stickyStyles_defaultTag": "div",
        "font_defaultLabel": "Segoe UI",
        "fontSize_defaultLabel":"12",
        "toolbarLocation": "top",
        "toolbar": [
        [ "CopyFormatting" ],
        [ "Font" ],
        [ "FontSize" ],
        [ "Bold" ],
        [ "Italic" ],
        [ "Underline" ],
        [ "BGColor" ],
        [ "TextColor" ],
        [ "BulletedList" ],
        [ "NumberedList" ],
        [ "Outdent" ],
        [ "Indent" ],
        [ "Blockquote" ],
        [ "JustifyLeft" ],
        [ "JustifyCenter" ],
        [ "JustifyRight" ],
        [ "Link" ],
        [ "Unlink" ],
        [ "Subscript" ],
        [ "Superscript" ],
        [ "Strike" ],
        [ "Image" ],
        [ "BidiLtr" ],
        [ "BidiRtl" ],
        [ "Undo" ],
        [ "Redo" ],
        [ "RemoveFormat" ],
        [ "Table" ]
        ],
        "extraPlugins":
"accessibilityhelp,autogrow,autolink,basicstyles,bidi,blockquote,button,collapser,colorbutton,colordialog,confighelper,contextmenu,copyformatting,dialog,editorplaceholder,filebrowser,filetools,find,floatpanel,font,iframerestrictor,indentblock,justify,notification,panel,panelbutton,pastefromword,quicktable,selectall,stickystyles,superimage,tableresize,tableselection,tabletools,uploadfile,uploadimage,uploadwidget"
```

```
,
        "toolbarcollapser_enableResizer": false,
        "placeholder":"",
        "linkTargets": ["notSet", "_blank"],
        "stylesSet": "default",
        "fontSize_sizes":
"8/8px;9/9px;10/10px;11/11px;12/12px;14/14px;16/16px;18/18px;20/20px;22/22px;24/24p
x;26/26px;28/28px;36/36px;48/48px;72/72px",
        "font_names": "Angsana New/'Angsana New', 'Leelawadee UI', Sathu,
serif;Arial/Arial, Helvetica, sans-serif;Arial Black/'Arial Black', Arial, sans-
serif;Calibri Light/'Calibri Light', 'Helvetica Light', sans-serif;Calibri/Calibri,
Helvetica, sans-serif;Cambria/Cambria, Georgia, serif;Candara/Candara, Optima,
sans-serif;Century Gothic/'Century Gothic', sans-serif;Comic Sans MS/'Comic Sans
MS';Consolas/Consolas, Courier, monospace;Constantia/Constantia, 'Hoefler Text',
serif;Corbel/Corbel, Skia, sans-serif;Cordia New/'Cordia New', 'Leelawadee UI',
Silom, sans-serif;Courier New/'Courier New';DaunPenh/DaunPenh, 'Leelawadee UI',
'Khmer MN', sans-serif;Franklin Gothic Book/'Franklin Gothic Book', 'Avenir Next
Condensed', sans-serif;Franklin Gothic Demi/'Franklin Gothic Demi', 'Avenir Next
Condensed Demi Bold', sans-serif;Franklin Gothic Medium/'Franklin Gothic Medium',
'Avenir Next Condensed Medium', sans-serif;Garamond/Garamond, Georgia,
serif;Gautami/Gautami, 'Nirmala UI', 'Telugu MN', sans-serif;Georgia/Georgia,
serif;Impact/Impact, Charcoal, sans-serif;Iskoola Pota/'Iskoola Pota', 'Nirmala
UI', 'Sinhala MN', sans-serif;Kalinga/Kalinga, 'Nirmala UI', 'Oriya MN', sans-
serif;Kartika/Kartika, 'Nirmala UI', 'Malayalam MN', sans-serif;Latha/Latha,
'Nirmala UI', 'Tamil MN', sans-serif;Leelawadee UI/'Leelawadee UI', Thonburi, sans-
serif;Lucida Console/'Lucida Console', Monaco, monospace;Lucida Handwriting/'Lucida
Handwriting', 'Apple Chancery', cursive;Lucida Sans Unicode/'Lucida Sans
Unicode';Mangal/Mangal, 'Nirmala UI', 'Devanagari Sangam MN', sans-serif;Nirmala
UI/'Nirmala UI', sans-serif;Nyala/Nyala, Kefa, sans-serif;Palatino
Linotype/'Palatino Linotype', 'Book Antiqua', Palatino, serif;Raavi/Raavi, 'Nirmala
UI', 'Gurmukhi MN', sans-serif;Segoe UI/'Segoe UI', 'Helvetica Neue', sans-
serif;Shruti/Shruti, 'Nirmala UI', 'Gujarati Sangam MN', sans-serif;Sitka
Heading/'Sitka Heading', Cochin, serif;Sitka Text/'Sitka Text', Cochin,
serif;Sylfaen/Sylfaen, Mshtakan, Menlo, serif;TW Cen MT/'TW Cen MT', 'Century
Gothic', sans-serif;Tahoma/Tahoma, Geneva, sans-serif;Times New Roman/'Times New
Roman', Times, serif;Times/Times, 'Times New Roman', serif;Trebuchet MS/'Trebuchet
MS';Tunga/Tunga, 'Nirmala UI', 'Kannada MN', sans-serif;Verdana/Verdana, Geneva,
sans-serif;Vrinda/Vrinda, 'Nirmala UI', 'Bangla MN', sans-serif;メイリオ/Meiryo, メイ
リオ, 'Hiragino Sans', sans-serif;仿宋/FangSong, 仿宋, STFangsong, serif;微軟正黑
體/'Microsoft JhengHei', 微軟正黑體, 'Apple LiGothic', sans-serif;微软雅黑/'Microsoft
YaHei', 微软雅黑, STHeiti, sans-serif;新宋体/NSimSun, 新宋体, SimSun, 宋体, SimSun-
ExtB, 宋体-ExtB, STSong, serif;新細明體/PMingLiU, 新細明體, PMingLiU-ExtB, 新細明體-
ExtB, 'Apple LiSung', serif;楷体/KaiTi, 楷体, STKaiti, serif;標楷體/DFKai-SB, 標楷體,
BiauKai, serif;游ゴシック/'Yu Gothic', 游ゴシック, YuGothic, sans-serif;游明朝/'Yu
Mincho', 游明朝, YuMincho, serif;隶书/SimLi, 隶书, 'Baoli SC', serif;黑体/SimHei, 黑
体, STHeiti, sans-serif;굴림/Gulim, 굴림, 'Nanum Gothic', sans-serif;궁서/Gungsuh, 궁
서, GungSeo, serif;돋움/Dotum, 돋움, AppleGothic, sans-serif;맑은 고딕/'Malgun
Gothic', '맑은 고딕', AppleGothic, sans-serif;바탕/Batang, 바탕, AppleMyungjo, serif;
바탕체/BatangChe, 바탕체, AppleMyungjo, serif;ＭＳ Ｐゴシック/'MS PGothic', 'ＭＳ Ｐゴ
シック', 'MS Gothic','ＭＳ ゴシック', 'Hiragino Kaku Gothic ProN', sans-serif;ＭＳ Ｐ
明朝/'MS PMincho', 'ＭＳ Ｐ明朝', 'MS Mincho', 'ＭＳ 明朝', 'Hiragino Mincho ProN',
serif",
        "copyFormatting_allowRules": "b s u i em strong span p div td th ol ul
li(*)[*]{*}"
    },
```

```
    "readOnlySettings": {
    "height": 500,
    "showFullScreenExpander": true
    },
    "sanitizerAllowlist": {
    "attributes": [],
    "cssProperties": [],
    "domains": [],
    "protocols": [],
    "tags": []
    },
    "disableContentSanitization": true,
    "disableOnLoadEditModeSanitization": true,
    "disableDefaultImageProcessing": true,
    "showFullScreenExpander": false,
    "showAsTabControl": false,
    "showHtml": false
}
```

defaultSupportedProps

The following table describes the most commonly used properties, but you can configure all the properties that CKEditor supports. The rich text editor control uses CKEditor 4.

Expand table

Property	Description	Default value
height	Sets the initial height of the content area, in pixels.	"185"
stickyStyle	Sets the editor font and size.	See defaultSupportedProps
stickyStyles_defaultTag	Creates a wrapper around the text in the editor content area. It's initially set to "div," but you can change it to "p" or any alternate tag.	"div"
font_defaultLabel	Sets the font label displayed in the toolbar. The label is for appearance only and isn't functional. The `stickyStyle` property determines the editor font and size.	"Segoe UI"
fontSize_defaultLabel	Sets the font size label displayed in the toolbar. The label is for appearance only and isn't functional. The `stickyStyle` property determines the editor font and size.	"9"
toolbarLocation	Sets the location of the toolbar in the editor content area. Supported values are "top" and "bottom."	"bottom"

Property	Description	Default value
toolbar	Lists the toolbar buttons to display.	See defaultSupportedProps
plugins	Lists preset plug-ins that the editor can use. The plug-ins in this list might be different from the plug-ins that are loaded, if `extraPlugins` and `removePlugins` are given a value. If you set this property to an empty string, the editor loads without the toolbar.	See defaultSupportedProps
extraPlugins	Appends plug-ins to the `plugins` list to load more plug-ins. Many plug-ins require other plug-ins to work. The rich text editor automatically adds them, and you can't use this property to override them. Use `removePlugins` instead.	See defaultSupportedProps
removePlugins	Lists plug-ins not to load. Use it to change which plug-ins are loaded without changing the `plugins` and `extraPlugins` lists.	See defaultSupportedProps
superimageImageMaxSize	Sets the maximum size in megabytes (MB) allowed for embedded images when using the superimage plug-in.	"5"
linkTargets	Allows you to configure which link target options are available for users when they create links: - "notSet": No target set - "frame": Opens the document in the specified frame - popupWindow": Opens the document in a popup window - "_blank": Opens the document in a new window or tab - "_top": Opens the document in the full body of the window - "_self": Opens the document in the same window or tab where the link is activated - "_parent": Opens the document in the parent frame	"notSet", "_blank"

Individual properties

The following table describes more properties you can use to customize the rich text editor control.

Expand table

Property	Description	Default value
attachmentEntity	To enforce more security on uploaded files by using a table other than the default, set this property and specify a different table. Syntax: "name": "*tableName*", "fileAttributeName": "*attributeNameofBlobReference*"	See defaultSupportedProps
disableContentSanitization	Content sanitization removes some custom attributes or tags from rich text content. It's disabled by default to allow copying and pasting of rich text content from external sources. This property applies only to edit mode. When the editor control is read-only or disabled, content is always sanitized.	true
disableDefaultImageProcessing	By default, images that are inserted in the editor are uploaded to the `attachmentEntity` defined in the configuration. External users might lack privileges to view the content in the table. Instead, set this property to true to store images as base64 strings directly in the column configured to use the rich text editor control.	false
disableImages	Determines whether images can be inserted in the editor. This property has highest priority. When this property is set to true, images are disabled, regardless of the value of the `imageEntity` property.	false
imageEntity	To enforce more security on images by using a table other than the default, set this property and specify a different table. Syntax: "imageEntityName": "*tableName*", "imageFileAttributeName": "*attributeNameofBlobReference*"	See defaultSupportedProps
readOnlySettings	These properties determine the behavior of the column when viewed in a read-only or disabled state. You can specify any supported property.	None; see defaultSupportedProps for an example
sanitizerAllowlist	Lists other kinds of content that can be displayed in the editor.	See defaultSupportedProps
showAsTabControl	Allows you to display more commands above the content area. Must be set to true	false

Property	Description	Default value
	to use the following properties: `showFullScreenExpander`, `showHtml`, `showPreview`, `showPreviewHeaderWarning`	
showFullScreenExpander	Determines whether the editor can be used in full-screen mode. `showAsTabControl` must be set to true.	false
showHtml	Allows users to display and edit the HTML content directly. `showAsTabControl` must be set to true.	false
showPreview	Allows users to preview the editor content rendered as HTML. `showAsTabControl` must be set to true.	false
showPreviewHeaderWarning	Allows you to show or hide the warning message displayed when previewing content. `showAsTabControl` and `showPreview` must be set to true.	false
allowSameOriginSandbox	Allows the content in the editor to be treated as from the same origin as the rendering app. **Use this property with caution.** Only use trusted external content. When this property is set to true, any external content could have access to internal resources.	false

Example configurations

The following sample configurations create specific customizations of the rich text editor in the classic experience. You can use them as-is or as a jumping-off point to customize a specific instance of the rich text editor or globally.

Set the default font to 11-point Calibri

Set these defaultSupportedProps properties in your configuration file. Each value except the last one must be followed by a comma (`,`).

JSON

```
"font_defaultLabel": "Calibri",
"fontSize_defaultLabel": "11",
```

```
"stickyStyle": {
  "font-size": "11pt",
  "font-family": "Calibri/Calibri, Helvetica, sans-serif;"
},
```

Make the Enter key insert a new line instead of a paragraph block

By default, pressing the Enter key creates a paragraph block with the **<p>** HTML tag. HTML uses paragraph blocks to group information, like paragraphs in a Word document. Browsers can format the **<p>** tag slightly differently, so for visual consistency, you might want to use the new line or line break tag **<br\>** tag instead.

Set this defaultSupportedProps property in your configuration file. Follow this value with a comma (`,`) unless it's the last property in the file.

JSON

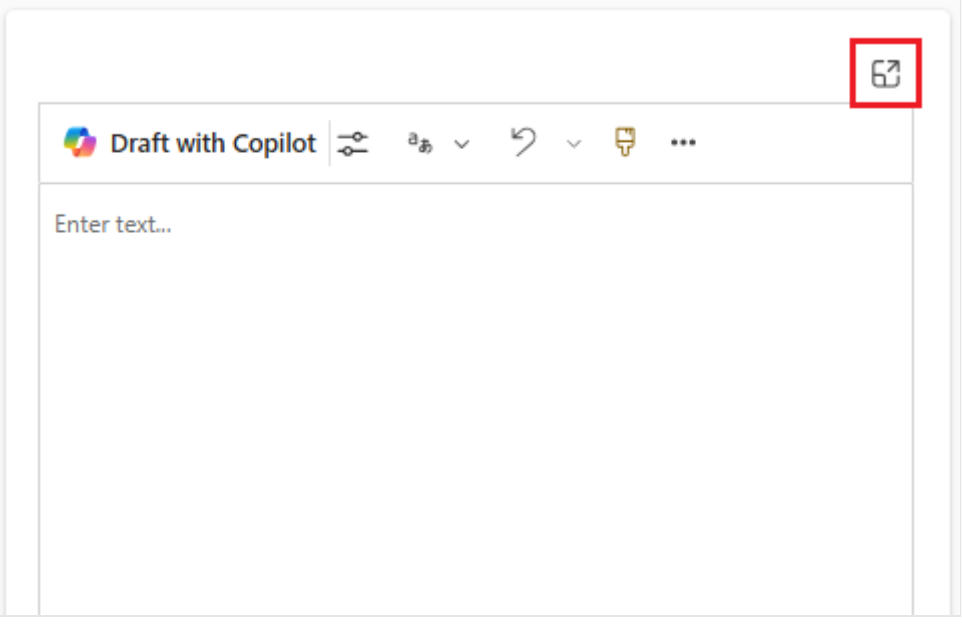
```
"enterMode": 2,
```

Allow the editor to use the full screen

Set these individual properties in your configuration file. Each value except the last one must be followed by a comma (`,`).

JSON

```
"showFullScreenExpander": true,
```



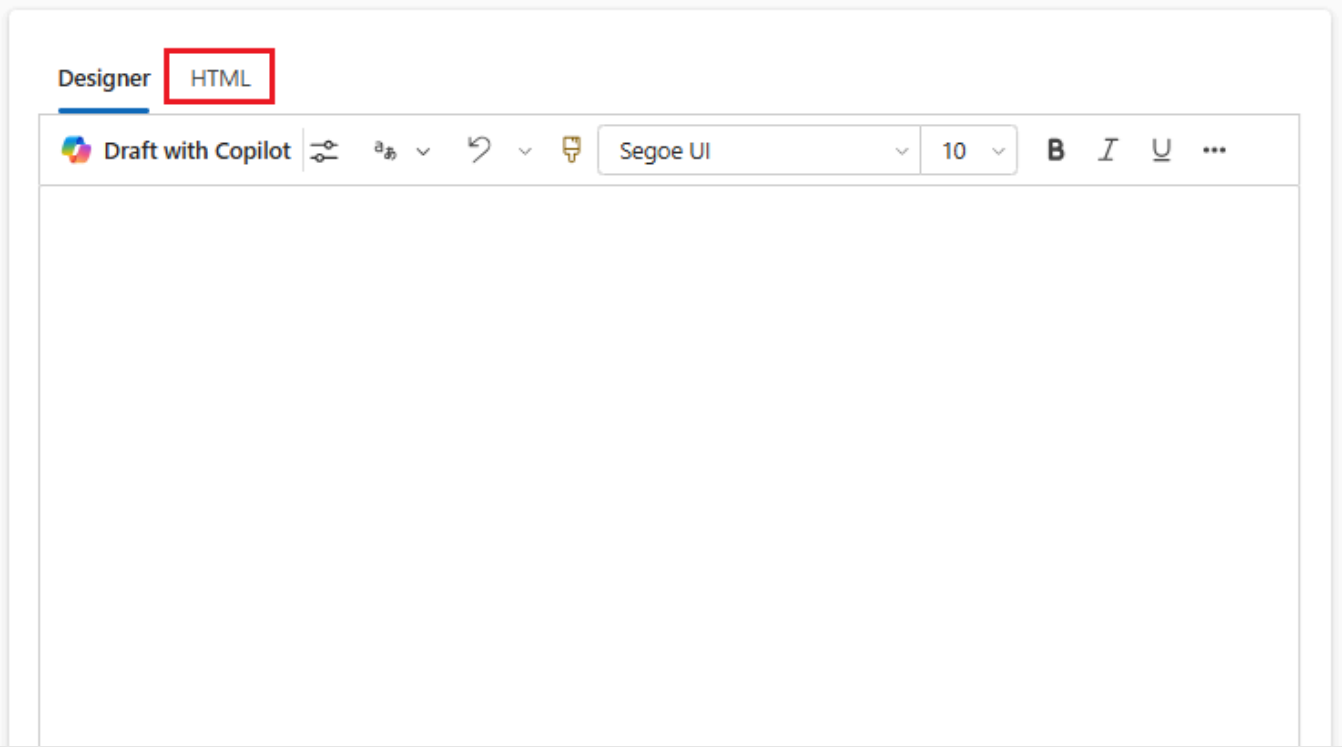
Allow viewing and editing HTML content

To allow users to view and edit the HTML of their content directly, show the **HTML** tab.

Set these individual properties in your configuration file. Each value except the last one must be followed by a comma (`,`).

JSON

```
"showAsTabControl": true,
"showHtml": true,
```



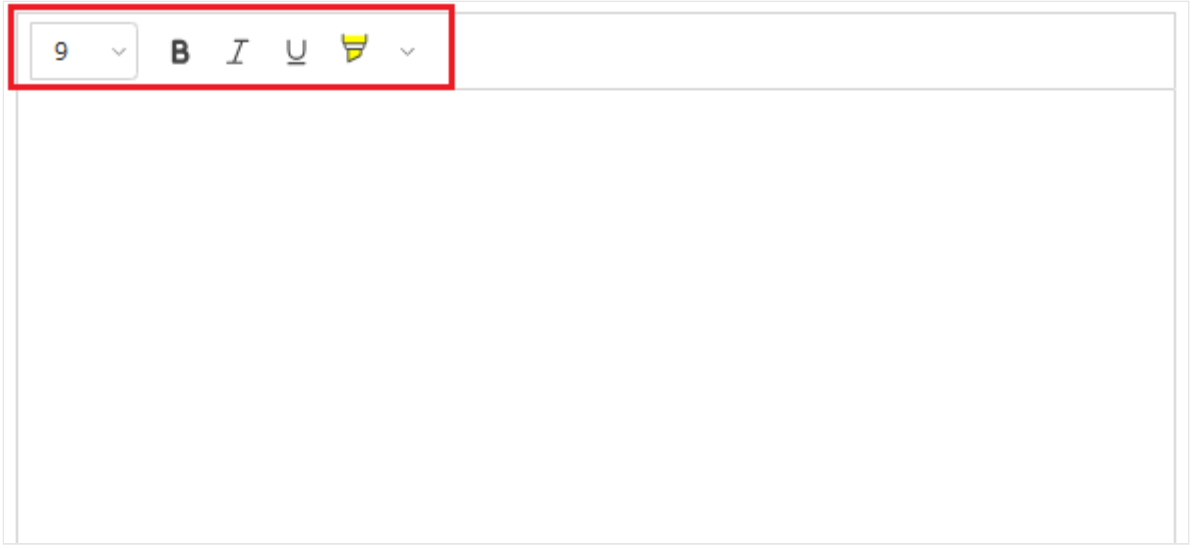
Show a simplified toolbar or remove it entirely

By default, the editor toolbar contains all the available formatting tools. To provide a simplified toolbar, use the `toolbar` property and specify the tools that users can use to format their content.

Set this defaultSupportedProps property in your configuration file. Follow this value with a comma (`,`) unless it's the last property in the file.

JSON

```
"toolbar": [ { "items": [ "FontSize", "Bold", "Italic", "Underline", "BGColor" ] }
],
```



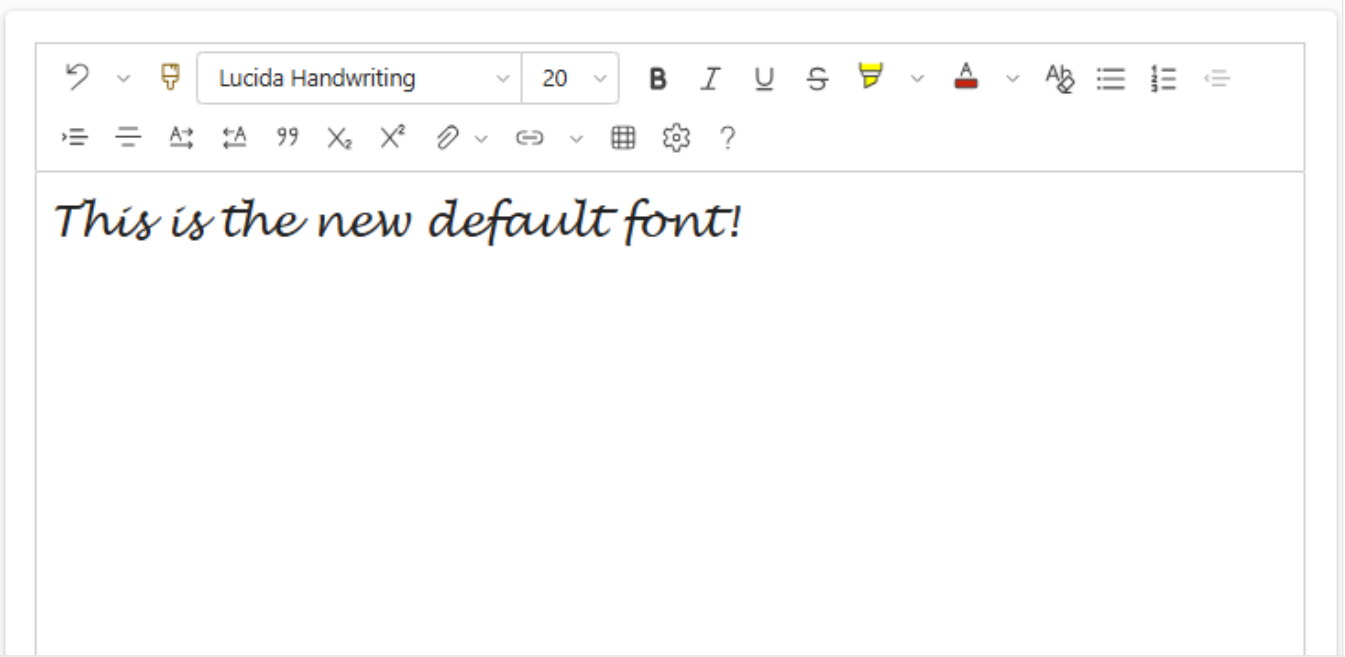
To remove the toolbar entirely, set the value of `toolbar` to '[]' (two square brackets).

Add a new font list and set 20-point Brush Script MT as the default font

Set these defaultSupportedProps properties in your configuration file. Each value except the last one must be followed by a comma (`,`).

JSON

```
"font_names": "Brush Script MT/'Brush Script MT', cursive;Calibri/Calibri,
Helvetica, sans-serif;Calibri Light/'Calibri Light', 'Helvetica Light', sans-
serif;"
"font_defaultLabel": "Brush Script MT"
"fontSize_sizes": "8/8pt;12/12pt;20/20pt;32/32pt"
"fontSize_defaultLabel": "20"
"stickyStyle": {
  "font-size": "20pt",
  "font-family": "'Brush Script MT', cursive"
},
```



Position the toolbar at the top of the rich text editor

Set this defaultSupportedProps property in your configuration file. Follow this value with a comma (`,`) unless it's the last property in the file.

JSON

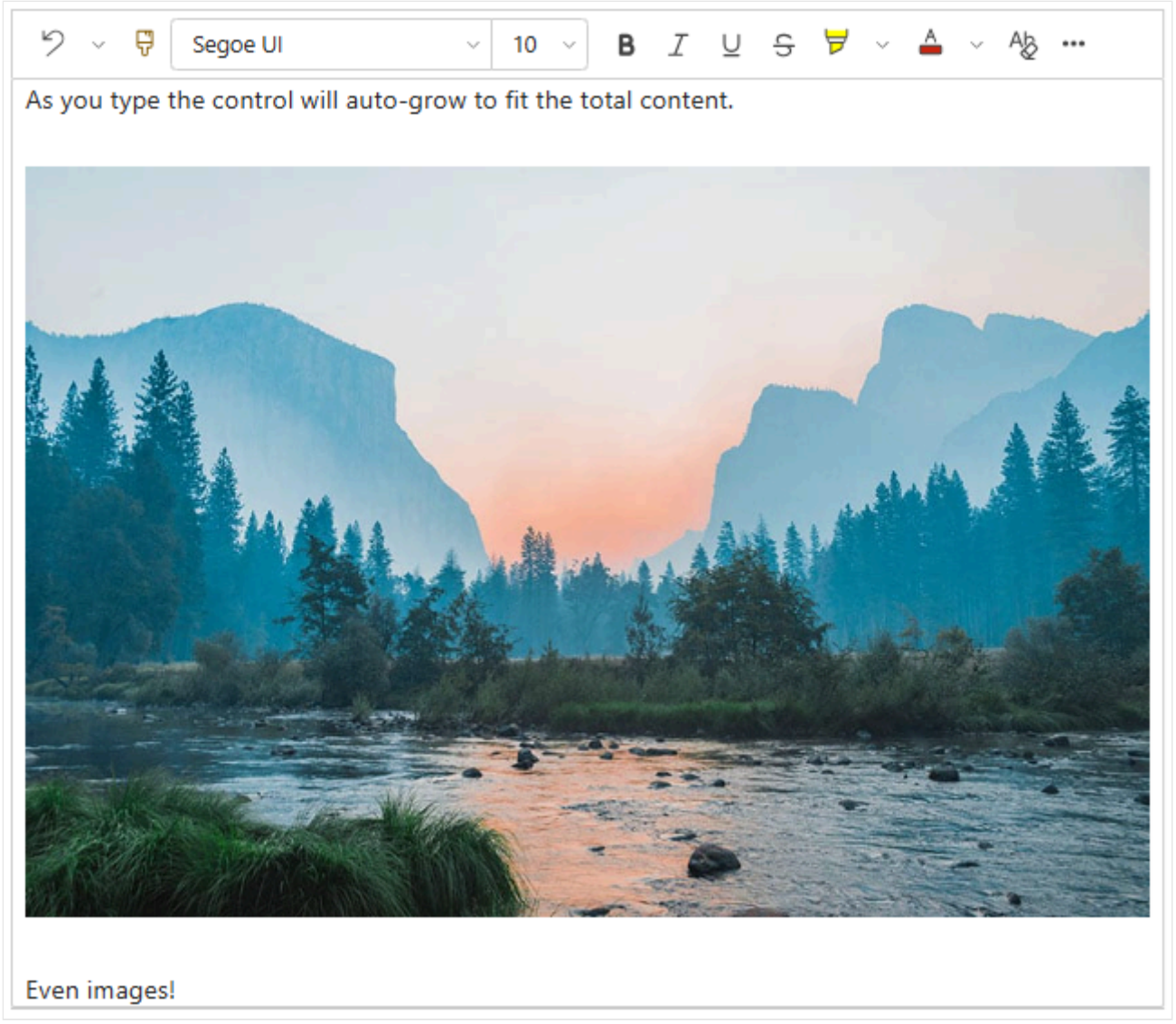
```
"toolbarLocation": "top",
```

Start the editor at 30 pixels high and grow to fit content

Set these defaultSupportedProps properties in your configuration file. Each value except the last one must be followed by a comma (`,`).

JSON

```
"autoGrow_onStartup": false,
"autoGrow_maxHeight": 8000,
"autoGrow_minHeight": 30,
"height": 30,
```

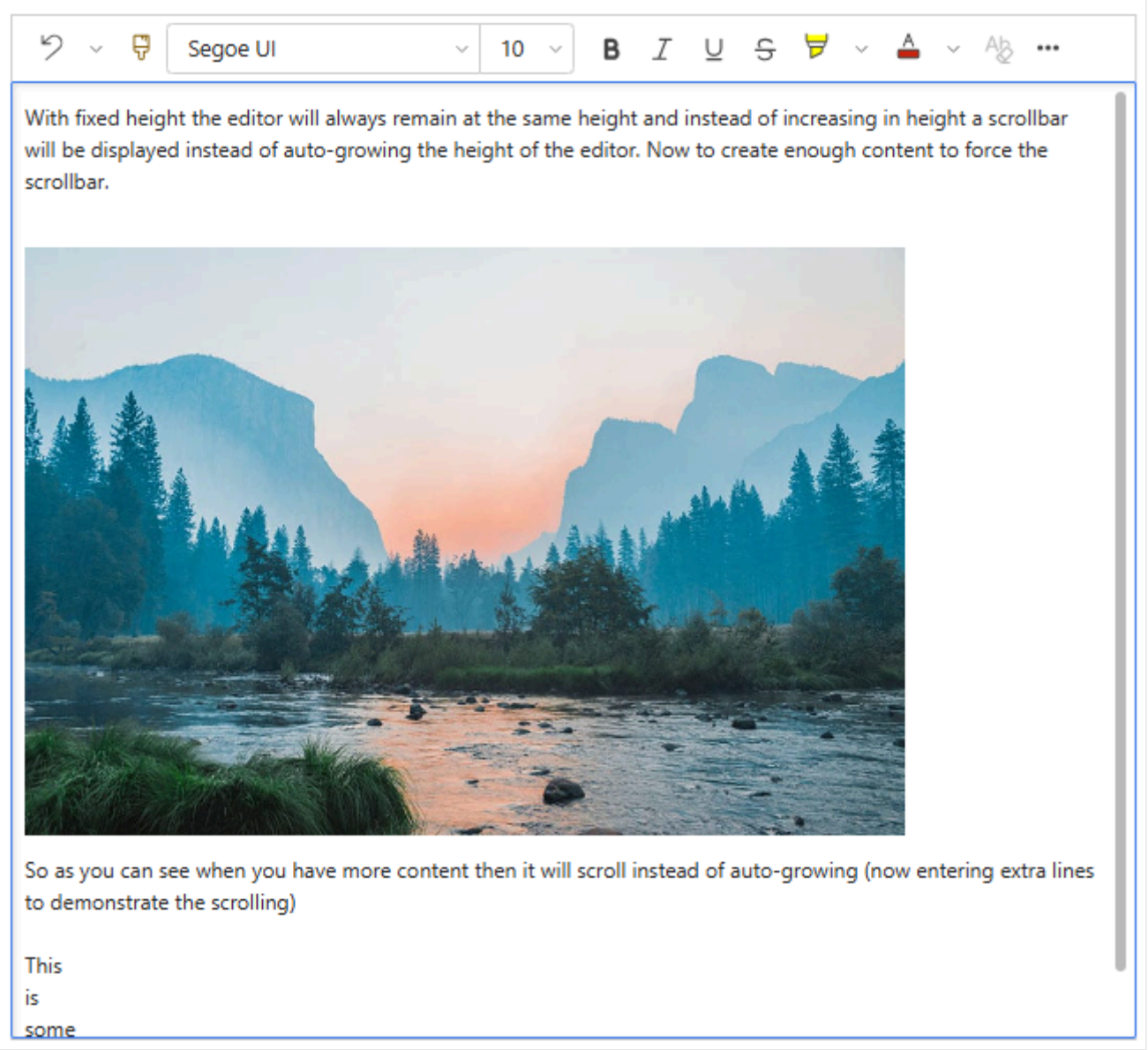


Fix the height of the editor at 500 pixels

Set these defaultSupportedProps properties in your configuration file. Each value, except the last one, must be followed by a comma.

JSON

```
"removePlugins": [ "autogrow" ],
"height": 500,
```



Make the context (right-click) menu use the browser's default context menu

To replace the editor control's context menu with the browser's default context menu, remove the contextmenu plug-in.

Set this defaultSupportedProps property in your configuration file. Follow this value with a comma (`,`) unless it's the last property in the file.

JSON

```
"removePlugins":
"contextmenu,liststyle,openlink,tableresize,tableselection,tabletools",
```

The rich text editor toolbar

The following table describes the formatting tools that are available in the rich text editor by default.

Expand table

Icon	Name	Shortcut key	Description
	Copy Formatting	Ctrl+Shift+C, Ctrl+Shift+V	Apply the look of a particular section to another section.
	Font Name	Ctrl+Shift+F	Select a font. The application considers the font that you select the default font. Segoe UI is the default font if you don't select one. When you select formatted content, the name of the font applied to it displays. If your selection contains multiple fonts, the first one applied to the selection displays.
	Font Size	Ctrl+Shift+P	Change the size of your text. The application considers the font size that you select the default size. 12 is the default size if you don't select one. When you select formatted content, the font size applied to it displays. If your selection contains multiple sizes, the first one applied to the selection displays.
	Bold	Ctrl+B	Make your text bold.
	Italic	Ctrl+I	Italicize your text.
	Underline	Ctrl+U	Underline your text.
	Background Color		Make your text stand out by highlighting it in a bright color.
	Text Color		Change the color of your text.
	Insert/Remove Bulleted List		Create a bulleted list.
	Insert/Remove Numbered List		Create a numbered list.
	Decrease Indent		Move your paragraph closer to the margin.
	Increase Indent		Move your paragraph farther away from the margin.

Icon	Name	Shortcut key	Description
	Block Quote		Apply a block-level quotation format to your content.
	Align Left	Ctrl+L	Align your content with the left margin.
	Align Center	Ctrl+E	Center your content on the page.
	Align Right	Ctrl+R	Align your content with the right margin.
	Link		Create a link in your document for quick access to web pages and other cloud resources. Enter or paste the anchor text in the **Display Text** box, and then enter or paste the URL in the **URL** box. Optionally, select a **Link Type** and protocol if the link is to something other than a web page. To specify where a URL-type link should open, select the **Target** tab and then select a **Target**. A pasted or typed URL is automatically converted into a link. For example, `http://myexample.com` becomes `<a href="http://myexample.com">http://myexample.com</a>`.
	Unlink		Remove the link from an anchor and make it plain text.
	Superscript		Type small letters just above the line of text.
	Subscript		Type small letters just below the line of text.
	Strikethrough		Cross out text by drawing a line through it.
	Image		To insert an image, paste it from your clipboard into the content area or drag an image file from a folder to the content area. Drag any corner of the image to resize it. The control supports .png, .jpg., and .gif images. To have more control over the image's source, appearance, and behavior, select the **Image** button. Browse to the image file in a local folder or enter its URL. If the image is stored on an external server, enter the absolute path. If it's on a local server, you can enter a relative path. Optionally, enter a specific height and width to resize the image, and select an alignment. You should also enter alt text to describe the image for people who use screen readers. If the image is also a link to a web page or other cloud resource, enter the URL of the resource in the **Target URL** box and, if needed, select the **Target** where the link should open.

Icon	Name	Shortcut key	Description
	Text direction from left to right		Change the text to left-to-right. This setting is the default.
	Text direction from right to left		Change the text to right-to-left for bi-directional language content.
	Undo		Reverse the last change you made to the content.
	Redo		Undo your last undo, or reapply the last change you made to the content.
	Remove Format		Remove all formatting from the select text.
	Table		Insert a table with the number of rows and columns you select. To have more control over the table's size and appearance, select the **Table** button > **More** and change its properties. You can also right-click a table to view and change its properties. To change the width of a column, drag its border. You can select one or more cells, rows, or columns and apply specific formatting, add a link to selected text, and cut, copy, and paste entire rows or columns.
	Personalization Settings		Set your default font and font size.
	Accessibility Help	Alt+0	Open the list of keyboard shortcuts you can use in the rich text editor.
···	Expand Toolbar		When the window is too narrow to show the entire toolbar, select to expand the toolbar to a second row.

Tip

To use your browser's built-in spell checker, press the Ctrl key as you right-click the text you want to check. Otherwise, the right-click (context) menu provides contextual formatting for the element you select.
An alternative to your browser's built-in spell checker is the Microsoft Editor browser extension. Microsoft Editor works seamlessly with the rich text editor control, and when enabled, provides fast and easy inline grammar and spell-check capabilities.

Use the rich text editor offline

The rich text editor control is available when you're working offline, although without some of the features and functionality you're used to. Images that you upload using the editor's default configuration aren't available when you're working offline.

The following plug-ins are available to the rich text editor when it's offline:

- ajax, autogrow, basicstyles, bidi, blockquote, button, confighelper, contextmenu, dialog, dialogui, editorplaceholder, enterkey, entities, fakeobjects, floatingspace, floatpanel, format, horizontalrule, htmlwriter, indent, indentblock, indentlist, justify, lineutils, list, listblock, maximize, menu, menubutton, notification, notificationaggregator, panel, panelbutton, popup, preview, removeformat, resize, richcombo, selectall, showborders, sourcearea, specialchar, stylescombo, tab, textmatch, textwatcher, toolbar, undo, widgetselection, wysiwygarea, xml

The following formatting tools are available in the rich text editor when it's offline:

- Bold, italic, underline, bulleted and numbered lists, decrease and increase indent, block quote, align left, align center, align right, strikethrough, bi-directional text, undo and redo, and remove formatting

Best practices for using the rich text editor

- Rich text fields store HTML tags for formatting along with user-entered data. When you set the maximum size of rich text fields, make sure to allow for both the HTML tags and the user's data.

- For the best performance, keep your HTML content to 1 MB or less. At larger sizes, you might notice slower response times for loading and editing.

- In the default configuration, images don't negatively affect performance because they're stored separately from the HTML content. However, images are stored as base64 content in the text column when the user who uploads them doesn't have permissions on the `msdyn_richtextfiles` table. Base64 content is large, so you generally don't want to store images as part of the column content.

- If you have a system administrator or basic user security role, the user personalization feature works by default. If you don't have these roles, you must have privileges to create,

read, and write on the `msdyn_customcontrolextendedsettings` table for the user personalization plug-in to work.

- In the default rich text editor configuration, if you drag and drop an image file into the rich text editor, it uploads the inline image to the `msdyn_richtextfiles` table. For appropriate users to be able to view the inline image, make sure you apply the correct permissions to `msdyn_richtextfiles`.

Rich text editor limitations

When using the rich text editor, consider the limitations listed in this section. For questions about feature availability, contact Microsoft Customer Support.

> **Important**
>
> The modern rich text editor is a new experience. For the functionality to work correctly, you must remove the classic version. Otherwise, your templates might not display correctly.

Rich text editor limitations include the following:

- You can't use rich text editor content from any external sources like Microsoft Word, Excel, and so forth.
- The rich text editor supports the following file types for attachments: .aac, .avi, .csv, .doc, .docx, .gif, .html, .jpeg, .mid, .midi, .mp3, .mp4, .mpeg, .msg, .pdf, .png, .ppt, .pptx, .svg, .txt, .vsd, .wav, .xls, .xlsm, and .xlsx. You can configure the allowed extensions for your environment in your advanced settings by going to **Administration** > **General** > **Set blocked file extensions for attachments** and removing the extensions you want to allow.
- The rich text editor doesn't support non-Microsoft plugins.

Email templates and signatures:

- If you experience an issue with the way an email template renders, we recommend that you recreate it in the modern editor.
- Nested HTML structures—such as deeply nested or <table> elements—can cause rendering problems in the rich text editor. We recommend that you simplify the HTML structure by removing unnecessary nested or <table> tags to ensure consistent rendering across clients.

HTML tags

The following HTML tags are either no longer supported or are replaced by a modern alternative in the rich text editor.

Expand table

Tag	Description	Modern rich text editor alternative
<acronym>	Defines an acronym	<abbr>
<applet>	Embeds Java applet	<object>
<basefont>	Sets default font size and color	CSS
<big>	Enlarges text	CSS font-size
<blink>	Makes text flash	CSS animations
	Centers content	CSS text-align
<dir>	Directory list	
<font>	Styles fonts	CSS
<frame>	Defines a frame	<iframe>
<frameset>	Groups frames	<iframe>
<isindex>	Single-line input	<form>
<marquee>	Scrolling text	CSS animations
<menu>	Menu list	
<noframes>	Fallback for no frame support	Not needed
<plaintext>	Displays plain text	<pre>
<s>	Strikethrough text	<del>, CSS
<strike>	Strikethrough text	<del>, <s>
<tt>	Teletype text	<code>, CSS
<u>	Underlined text	CSS text-decoration
<xmp>	Preformatted text	<pre>

HTML attributes

The following HTML attributes are either no longer supported or are replaced by a modern alternative in the rich text editor.

Expand table

Attribute	Affected tags	Modern rich text editor alternative
align	Many	CSS text-align, float
alink, vlink, link	<body>	CSS :link, :visited
background	<body>, <table>	CSS background-image
bgcolor	Many	CSS background-color
border	<img>, <table>	
height, width	<body>, <td>	CSS height, width
hspace, vspace	<img>	
language	<script>	type="text/JavaScript"
nowrap	<td>	CSS white-space: nowrap
type (on ,)	Lists	CSS list-style-type

Frequently asked questions

Why do typed characters display slowly?

If you have a lot of content in the editor, the response time can increase. Keep the content to 1 MB or less for the best performance. Spelling or grammar checks can also slow the typing performance.

Why doesn't my content render until the editor is refreshed?

This behavior occurs when custom code injects content into the rich text editor without triggering a re-render of the control. The rich text editor doesn’t automatically detect external updates unless it's explicitly notified.

To work around this issue, you can do either of the following actions:

- Use an event listener (for example, formContext.data.entity.addOnSave) to trigger a refresh after content is injected.

- Use formContext.ui.refreshRibbon() or a similar API to force an update.

Example:

```
window.top.addEventListener('rteEditorReady', (event) => {
    const { parentEntity } = event.detail;

    if (parentEntity?.typeName === 'email' &&
        parentEntity?.fieldName === 'description') {

        const descriptionAttr = Xrm.Page.getAttribute("description");
        if (descriptionAttr) {
            descriptionAttr.setValue(emailSignature);
        }
    }
}, { once: true });
```

In this example, the event listener is automatically removed after the `rteEditorReady` event triggers. This approach ensures the listener runs only once, helping to prevent memory leaks and unnecessary resource usage.

Why can't I upload an image? Why does the image preview fail to load?

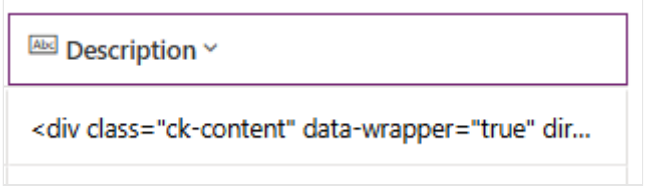
If the image file name, including the path, is long, the file might fail to upload, or the preview might not be displayed. Try shortening the file name or moving it to a location with a shorter path, and then upload it again.

Why do I see HTML in my text?

If you use the rich text editor control in a column that isn't formatted for rich text, the content appears in the underlying HTML instead of as formatted text.

What's the size limit of HTML content?

For the best performance, keep your HTML content to 1 MB or less. At larger sizes, you might notice slower response times for loading and editing.



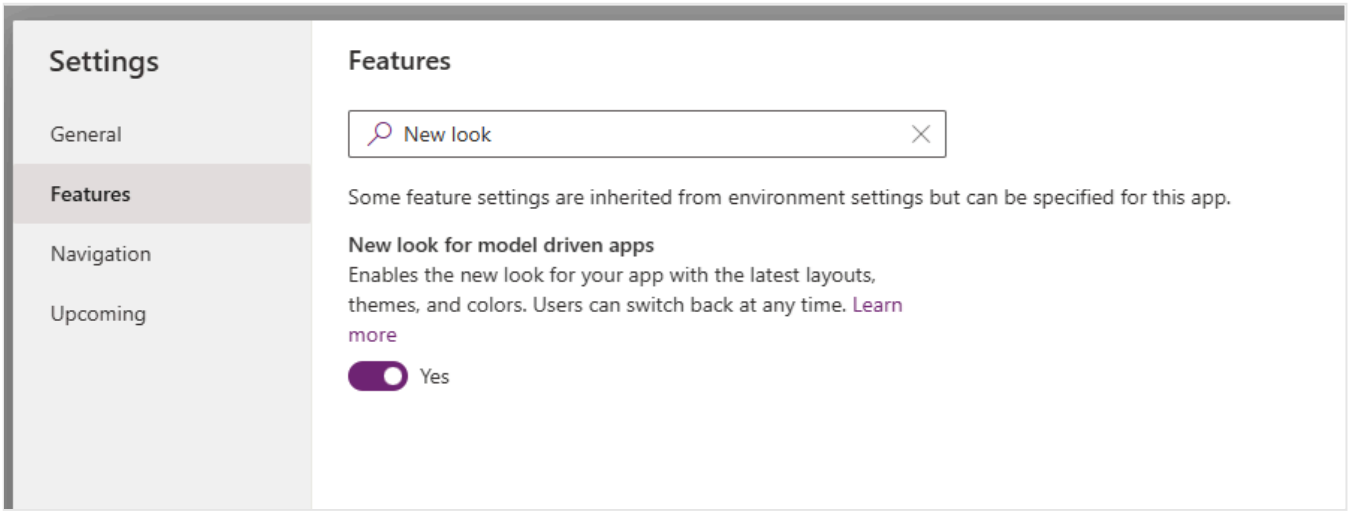
To resolve this issue, make sure the format of the column is set to **Rich text**.

I exceeded the character limit but there are fewer characters than the limit. Why?

The database stores the entire HTML formatting, not just the characters, which takes up extra space.

Why don't I see the modern rich text editor?

The modern rich text editor control is part of the **New look for model-driven apps** and doesn't appear unless the app setting is turned on. Without the setting enabled, you might see an older version of the editor.



Related information

- Create and edit columns for Microsoft Dataverse using Power Apps portal
- Use Copilot in rich text editor for email

Last updated on 04/07/2026

Add the Copilot control to the rich text editor

Article • 04/30/2025

The Copilot control is available in the email form's rich text editor toolbar by default. You can add the Copilot control to the rich text editor in other forms by changing the editor's properties in its advanced configuration file.

Prerequisites

Have a configuration file for the rich text editor in the form where you want to add the Copilot control.

Add the Copilot control

To add the Copilot control to the rich text editor toolbar in a form, you need to add a value that represents it to the `extraPlugins` and `toolbar` properties in the editor's advanced configuration file.

1. Open the editor's advanced configuration file.
2. Add the following values in the defaultSupportedProps section of the file:

   - In the `extraPlugins` property, add `copilotrefinement`.
   - In the `toolbar` property, add `CopilotRefinement`.

   It should look something like the following example, where "..." represents values that aren't shown to save space:

   ```text
   "defaultSupportedProps": {
       ...
       "extraPlugins": "computedfont,...,copilotrefinement",
       ...
       "toolbar":[{ "items": ["CopyFormatting", ... ,"CopilotRefinement"]}]
   },
   ```

3. Save the file, and then publish the changes.

See also

Use Copilot in the email rich text editor

Use Copilot in the email rich text editor

When you draft customer emails in the rich text editor in a model-driven app, Copilot can offer suggestions to make them clearer, more concise, and more compelling.

> **Important**
>
> By using Copilot features powered by Azure OpenAI, you agree that data may be stored and/or processed outside of your geographic region, compliance boundary, or national cloud instance. Learn more: Data, privacy, and security for Azure OpenAI Service

Prerequisites

The generative AI feature in emails must be turned on in the model-driven app feature settings.

Draft and refine an email in the rich text editor

1. Select the **Copilot** icon in the rich text editor toolbar and then select **Draft with Copilot**.
2. In the **Draft with Copilot** dialog box, select a prompt as required.
3. To refine or update the generated content, select the following options as required:

Expand table

Option	Description
Keep it	Retains the current text and adds it to the email.
Start over	Removes the current suggestion and takes you back to the email prompt.
Discard	Removes the content suggestion and takes you back to email editor.
Adjust	Refines the text and tone of suggested content. - To adjust the length of the content, select **Short**, **Medium**, or **Long**. - To adjust the tone of the content, select **Friendly**, **Professional**, or **Formal**. Select **Update** and the content is updated according to your refinements.
Translate	Translates the content into the selected language. The languages listed here are those supported by your environment. To go back to the language that the content was originally generated, select **Show original**.

Adjust existing email content with Copilot

1. Select the text that you want to refine.
2. Select the **Copilot** icon in the rich text editor toolbar and then select **Adjust with Copilot**.
3. Refine the length and tone of the selected text:
    - Length: Select **Short**, **Medium**, or **Long** to condense or expand on your text.
    - Tone: Select **Friendly**, **Professional**, or **Formal** to adjust the tone of your text.
4. Select **Update** to apply Copilot's suggested refinements.

For example, Marina wants to send an email to a customer about a new product. Marina types a draft, selects **Adjust with Copilot**, and asks Copilot to rewrite the draft to be shorter and in a more formal tone.

- Before Copilot's assistance:

```text
Hi Samir,

My name is Marina; I'm a Sales executive at Contoso Inc. I was
recommended by Kevin to get in touch with you.

I noticed on your website that you don't have CRM application. I'd
love to connect with you to discuss the potential of partnering up.
When would you have time to chat about this?

If you want to learn more about the product first, here's a quick
link: www.contoso.com.

Kind regards,
Marina
```

- After Copilot's assistance:

```text
Dear Mr. Samir,

I am Marina, a Sales Executive at Contoso Inc. Kevin recommended that I
contact you. I noticed that your website does not have a CRM application.
I would like to discuss the possibility of a partnership. When would you
have time to talk about this? If you would like to learn more about our
product, please refer to this link: www.contoso.com.
```

```
Sincerely,
Marina
```

See also

- Add the rich text editor control
- Add the Copilot control to the rich text editor
- FAQ about using Copilot in the rich text editor

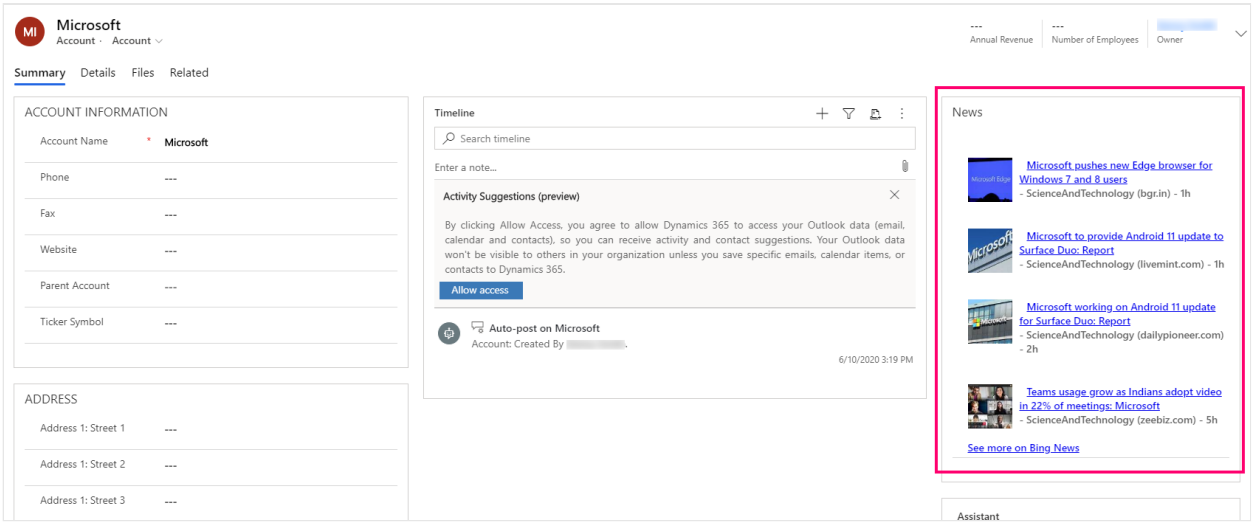
Last updated on 01/09/2026

Set up and use the news control

Article • 03/13/2023

> **Note**
>
> This describes the process that currently uses the classic editor to introduce a control. In due course this will be migrated so that this can be performed within the modern form editor.

Gain valuable insights from the latest news about your customers, competitors, and contacts. Staying on top of the news helps you stay ahead of your competition by showing you real-time contextual information about your customers and competitors. Install the news Power Apps component framework control, which delivers relevant news from Bing News.



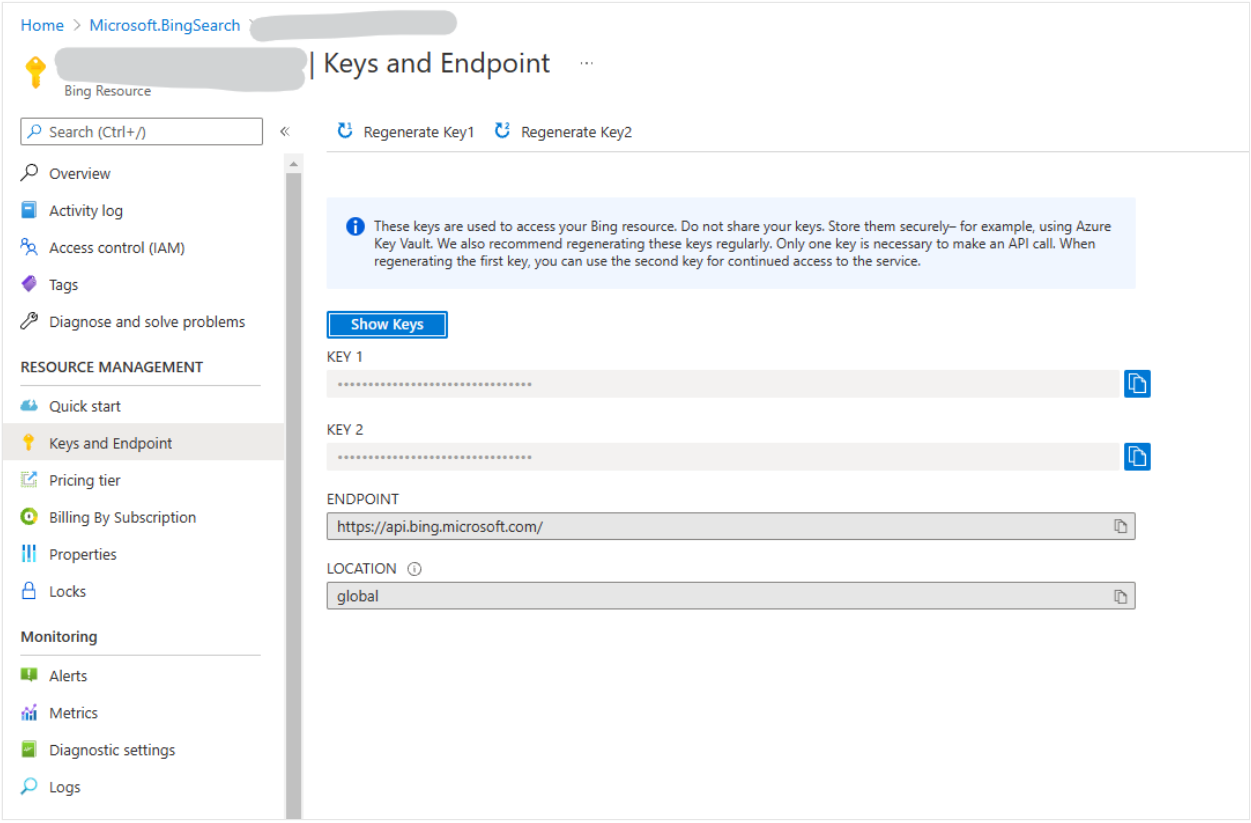
You can customize to support additional news sources, the look and feel and the behavior of the PCF control. For more information, see the readme in the news control GitHub repo.

Add the company news control to the account main form

The steps in this article add the company news control to the account main form already configured to query news topics collected from Bing News. You can use similar steps to add the control to the main form for other tables, such as contact and competitor.

Get the Bing News API key and URL

The news control requires a news API URL, API key, and more news URL. For Bing News, the more news URL is preconfigured in the control to use *https://www.bing.com/news/search*. For the News API URL, and API Key, you'll need to provide these by creating a Bing Search azure resource under your Azure account. Once created, you will get the API key and API URL under the **Keys and Endpoint** section.



Import the company news solution

1. Go to the news control GitHub repo.
2. Download Solutions.zip. To do this, open the **Solutions** folder, open **Solutions.zip**, and then select **Download**.
3. Sign-in to Power Apps, and then in the upper right select the environment where you want to install the company news control.
4. Select **Solutions** on the left pane, and then select **Import** on the command bar. If the item isn't in the left navigation pane, select ...More and then select the item you want.
5. In the Import solution wizard, select **Choose File**, browse to and select *Solutions.zip*, and then select **Open**.
6. Select **Next,** and then select **Next** again to complete the import.

Add the control to the account main form

1. In Power Apps select **Settings** (gear) on the upper right, and then select **Advanced settings**.

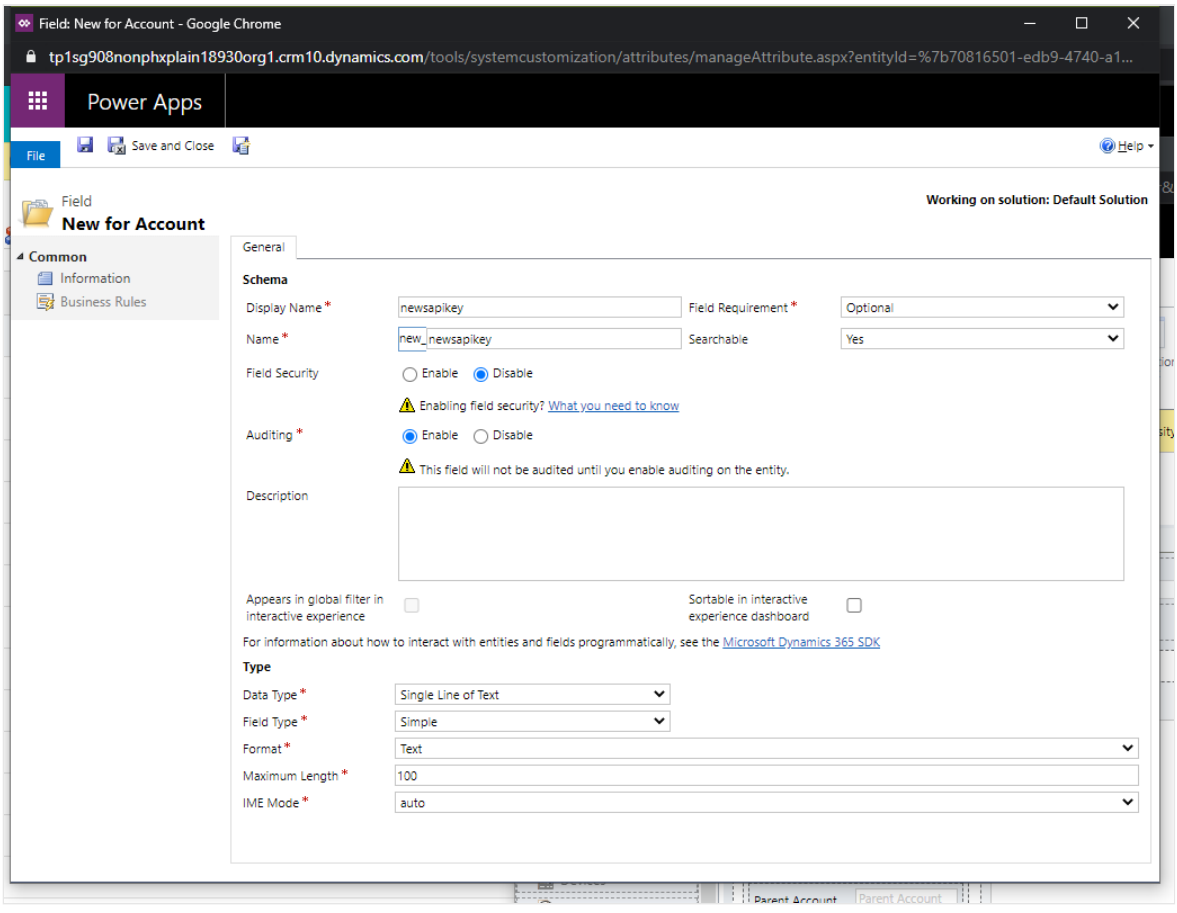
2. Go to **Settings** > **Customizations** > **Customize the System**.

3. In the left navigation tree, expand **Tables** > **Account**, select **Forms**, and then open the **Account Main** form.

4. In the form editor, add two new text columns, which will be used to pass the key and base URL to the news component.

   a. Select **New Column** from the right navigation **Column Explorer**.

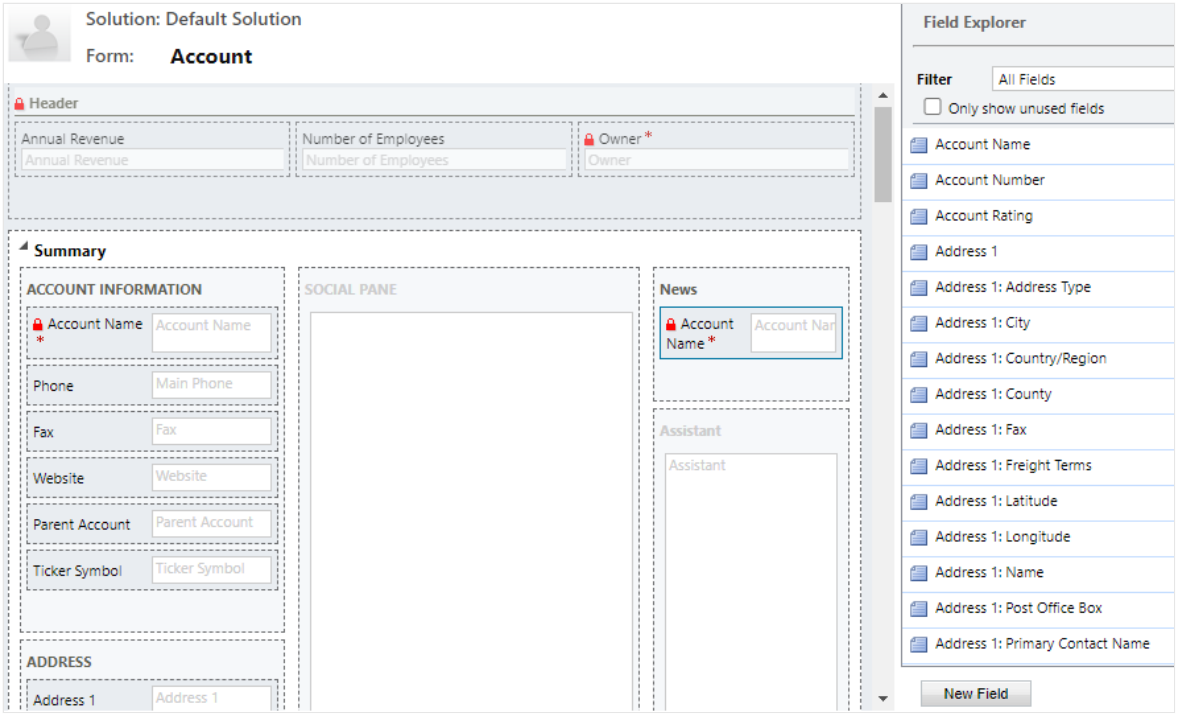
   b. For both **Display name** and **Name** enter *newsapikey*. Leave the rest of the settings as the default and then select **Save and Close**.



   c. Repeat the previous two steps to create another text column. For both the **Display name** and **Name** enter *newsurl*. Leave the rest of the settings as the default and then select **Save and Close**.

5. In the form editor, select a place on the form where you want the company news control, and then select **Insert** tab > **Section** > **One Column**.

6. To show news by account name, drag and drop the **Account Name** column from the right **Column Explorer** pane to the newly created section.

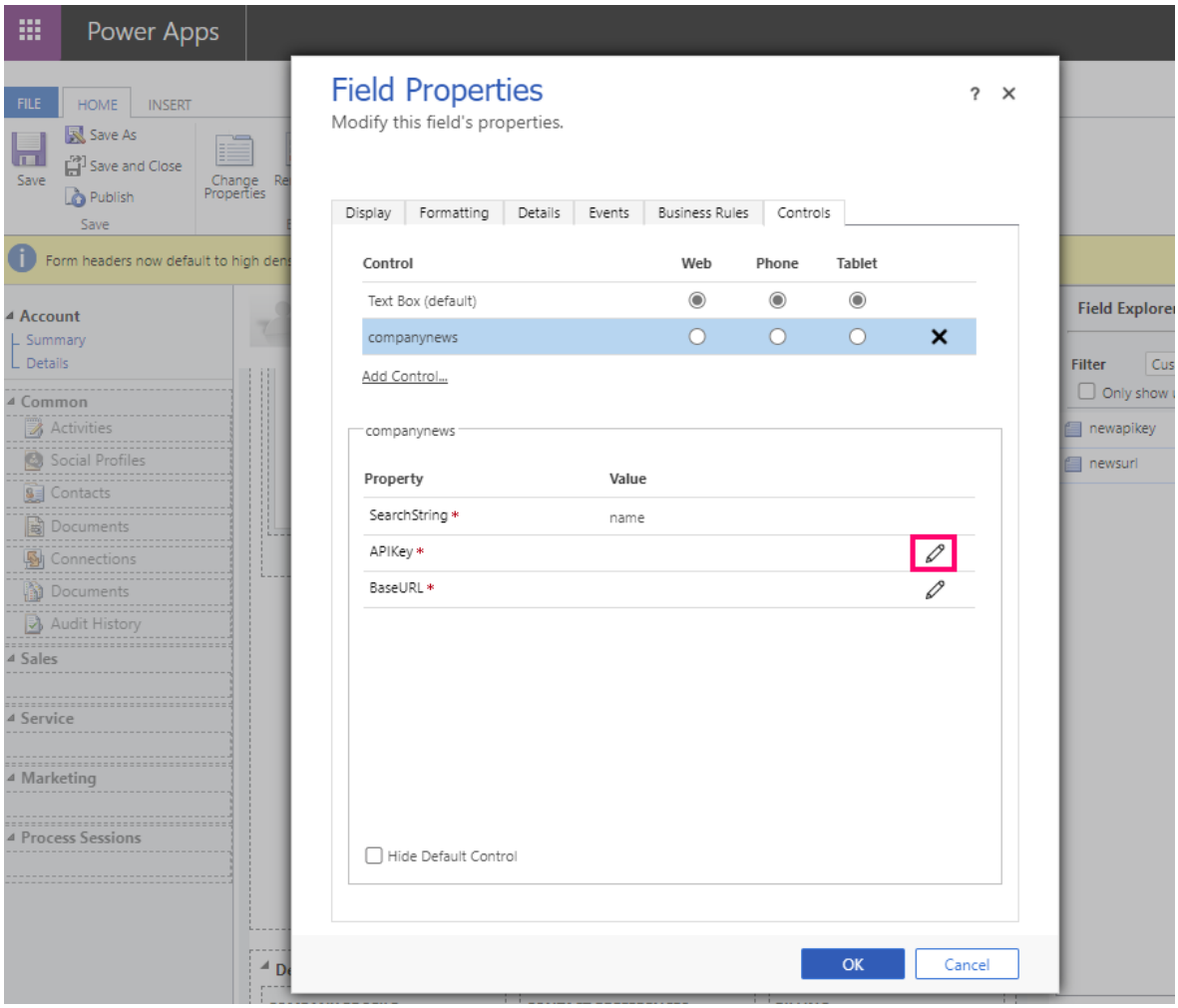


7. Select the section and then select **Change properties.** Enter a descriptive section a **Name**, such as *Company news*. You might want to check **Show the label of this section on the form** to provide heading to the section. Select **OK**.

8. Select the newly added **Account Name** column, and then select **Change properties** on the **Home** tab. On the **Column Properties** page, select the **Controls** tab, select **Add Control**, select the **companynews** control, and then select **Add**.

9. Configure the following bindings for the apikey and baseurl columns.

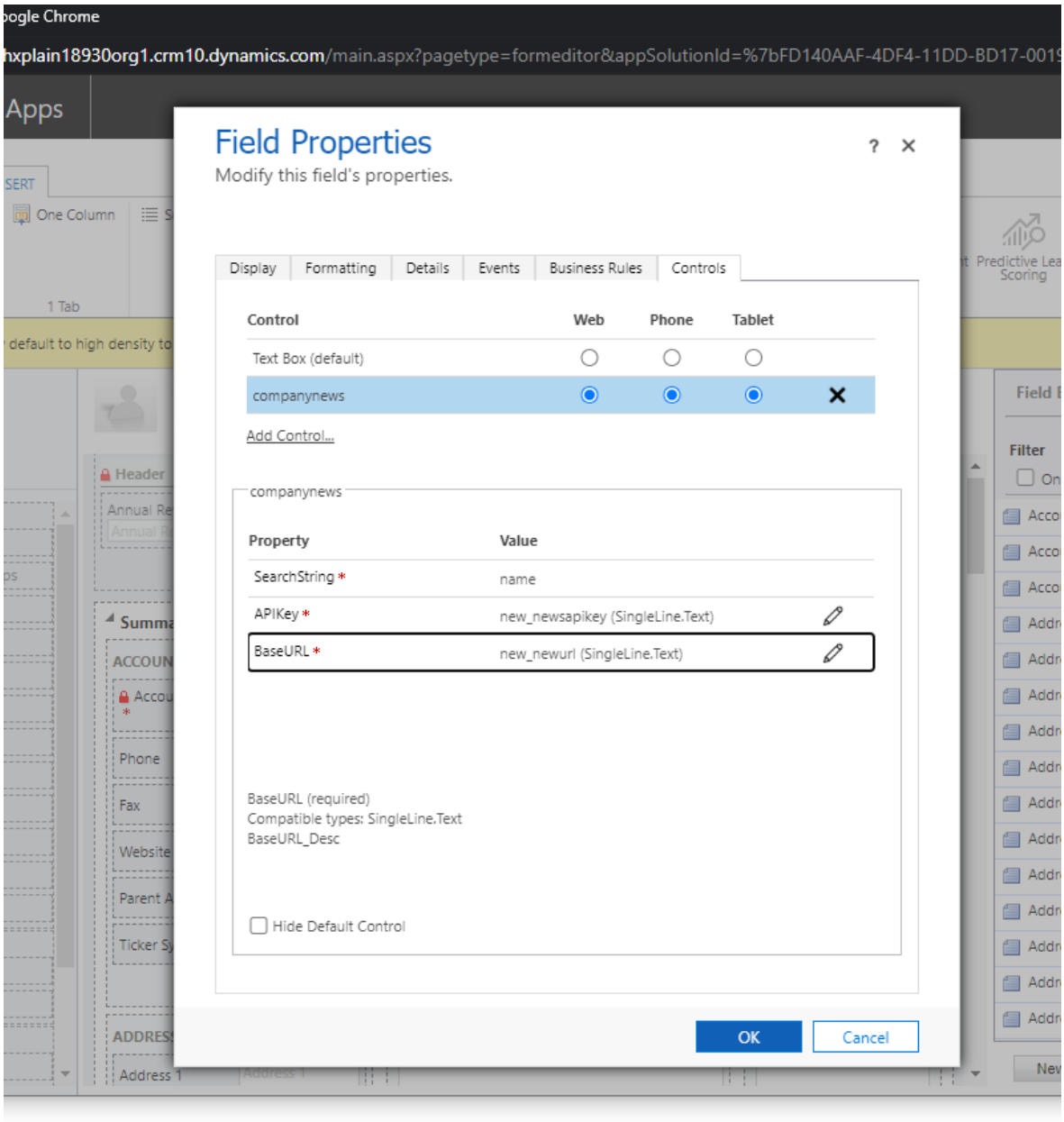
   a. Select **Configure property** (pencil icon) next to APIKey.



b. From **Bind to value on a column** the dropdown list, select **new_newsapikey (SingleLine.Text)**, and then select **OK**.

c. Select **Configure property** (pencil icon) next to **BaseURL**.

d. From **Bind to value on a column** the dropdown list, select **new_newsurl (SingleLine.Text)**, and then select **OK**.



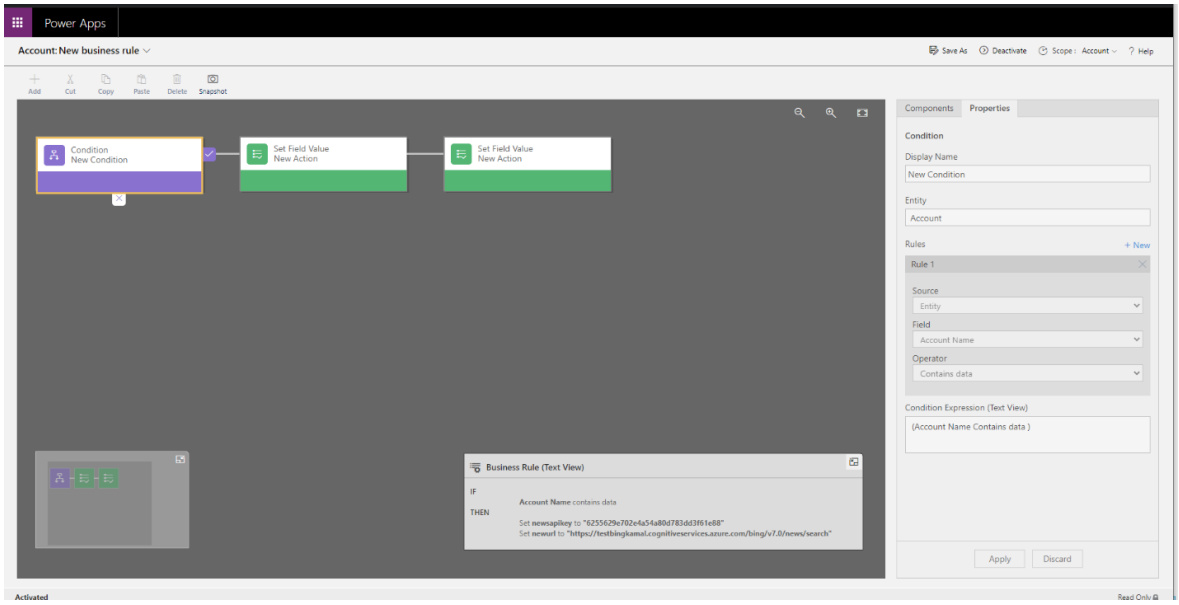
e. On the **Column Properties** page, select the **Web**, **Phone**, and **Tablet**, client options.

f. On the **Column Properties** page, select the **Display** tab, clear the **Display label on form** option, and then select **OK**.

10. To provide API Key and Base URL default values, use business rules. Select **Business Rules** from the form editor, and then select **New Business Rule** from the bottom of the right pane.

11. In the business rule designer, select **Condition** on the designer canvas, set **Column** to **Account Name** and **Operator** to **Contains data,** and then select **Apply**. Then, select the **Components** tab, drag a **Set Column Value** action, and provide a column value where **Column** is **newsapikey** and the value is the **Key** you copied from the Bing Search properties. Create another **Set Column Value** action where **Column** is **newsurl** and add the value as the **Endpoint** you copied from the Bing

Search properties. Make sure to append */v7.0/news/search* at the end of the news URL. The final URL should look like `https://api.bing.microsoft.com/v7.0/news/search`.



12. Save and activate the rule. Close the business rule designer.
13. Make sure you add the newly added columns, newsapikey and newsurl, to the account main form.
14. Clear **Visible by default** for the form tab properties where the newsapikey and newsurl columns are added.
15. In the form designer, select **Save** and then select **Publish**.

See also

Use custom controls for model-driven app data visualizations

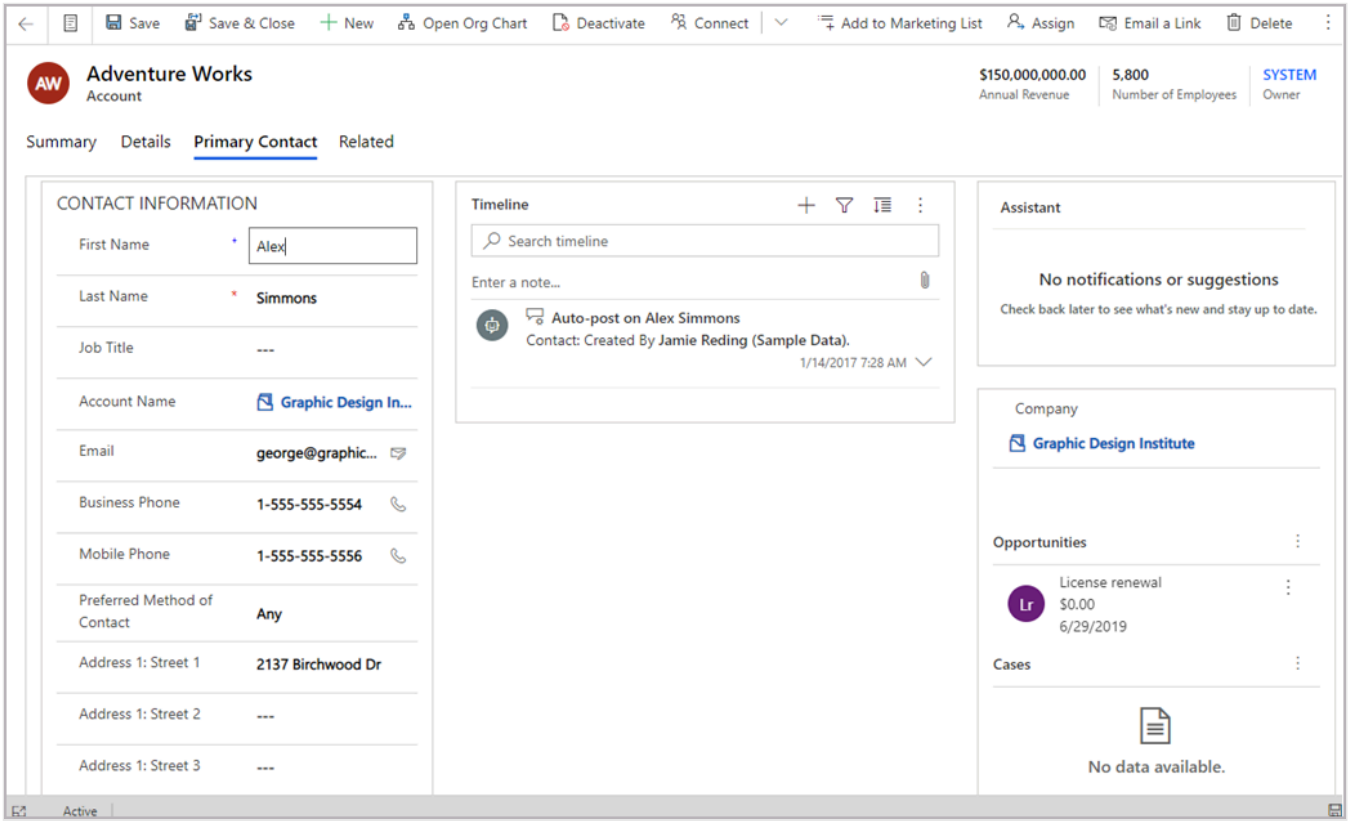
Edit related table records directly from another table's main form

Article • 04/28/2025

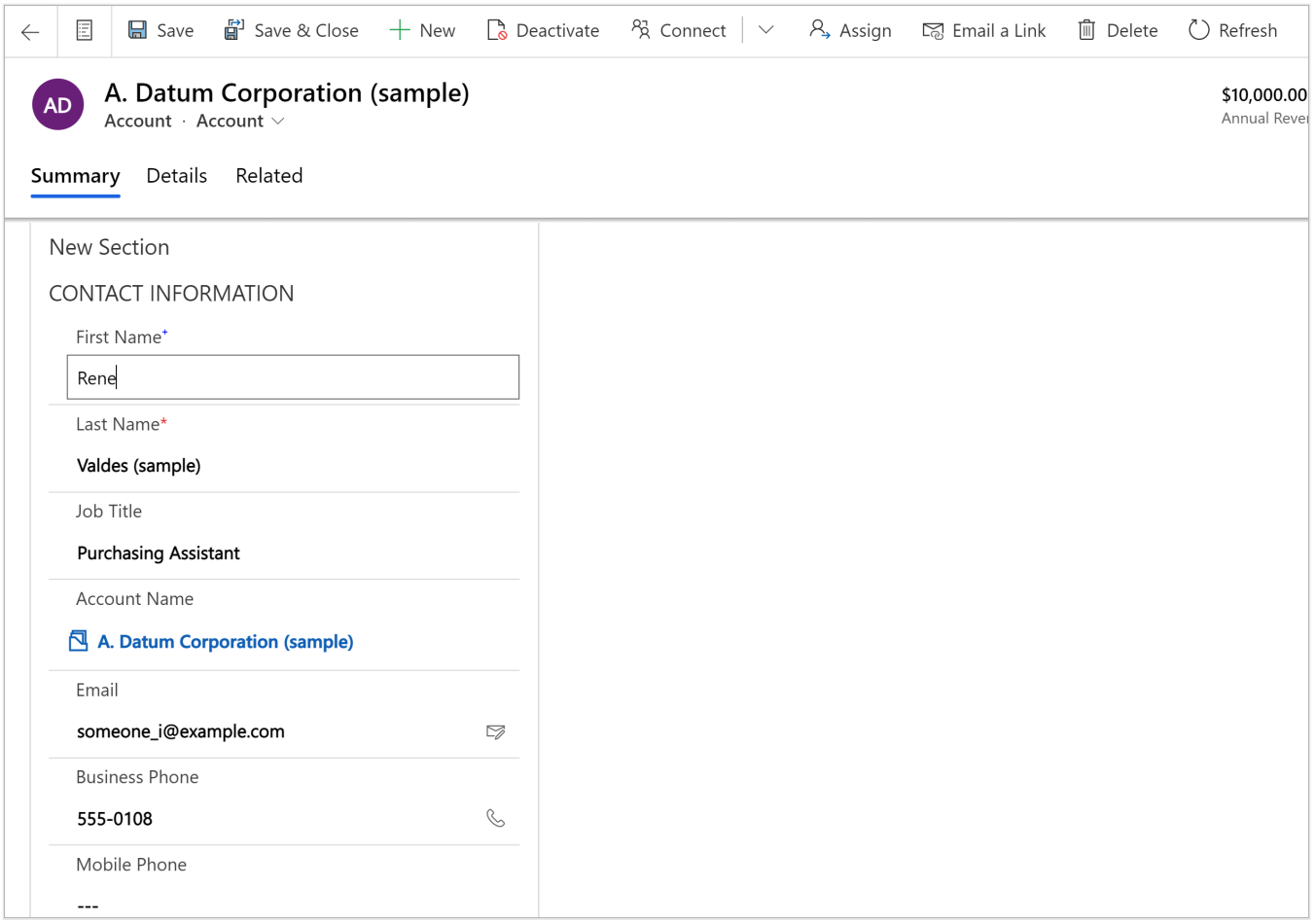
There are multiple ways that you can work with related table records on a table form within Power Apps. For example, you can include related tables in read-only mode with a quick view form and create or edit a record using a main form in a dialog.

Another way you can work with related table records is by adding a form component control to another table's main form. The form component control lets users edit information of a related table record directly from another table's form.

For example, here's the form component on a separate tab on the main account form, which lets the user edit a contact record without leaving the account form.



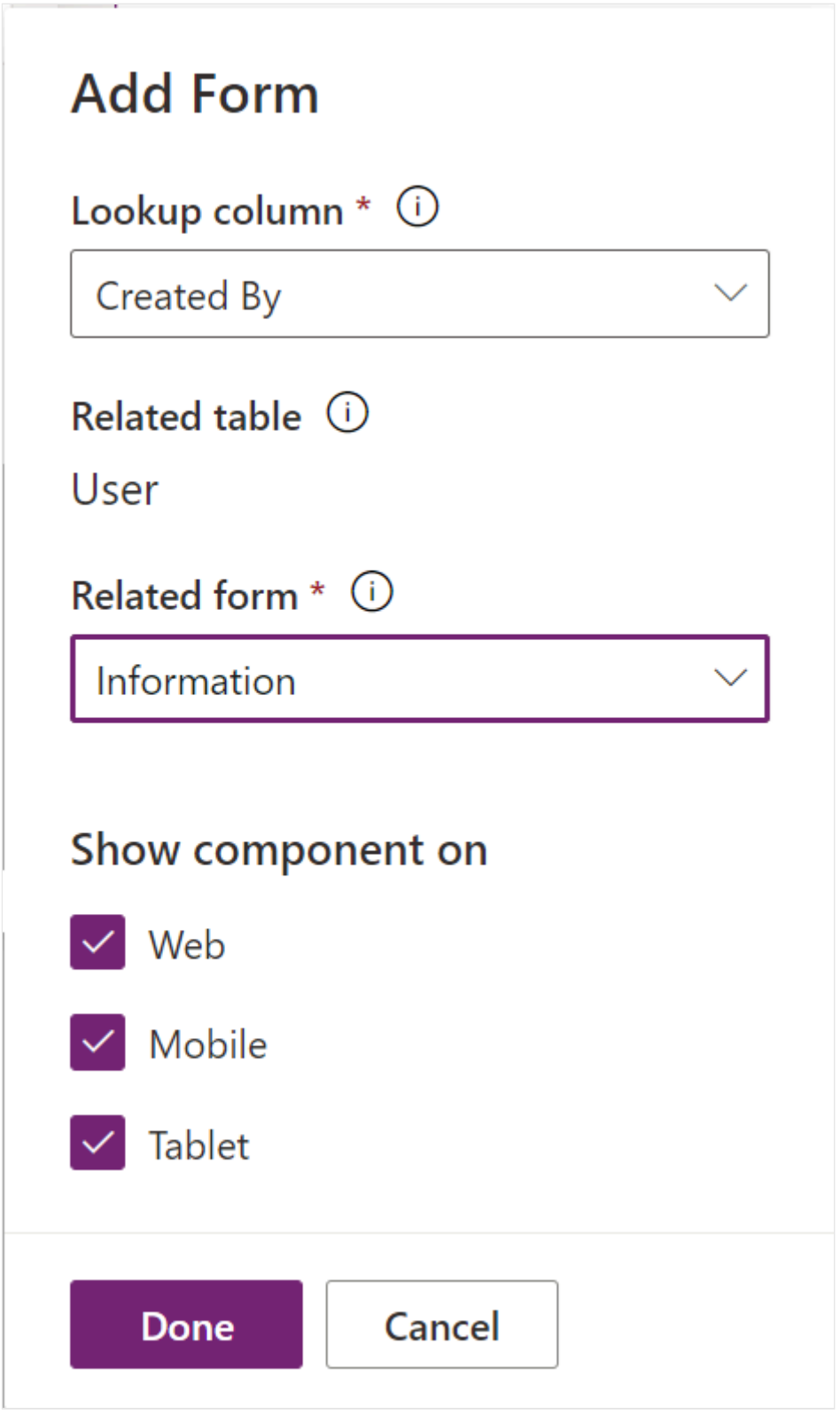
For example, here's the form component on an **existing tab** on the main account form, which also lets the user edit a contact record without leaving the account form **Summary** tab.



Add the form component to a table main form

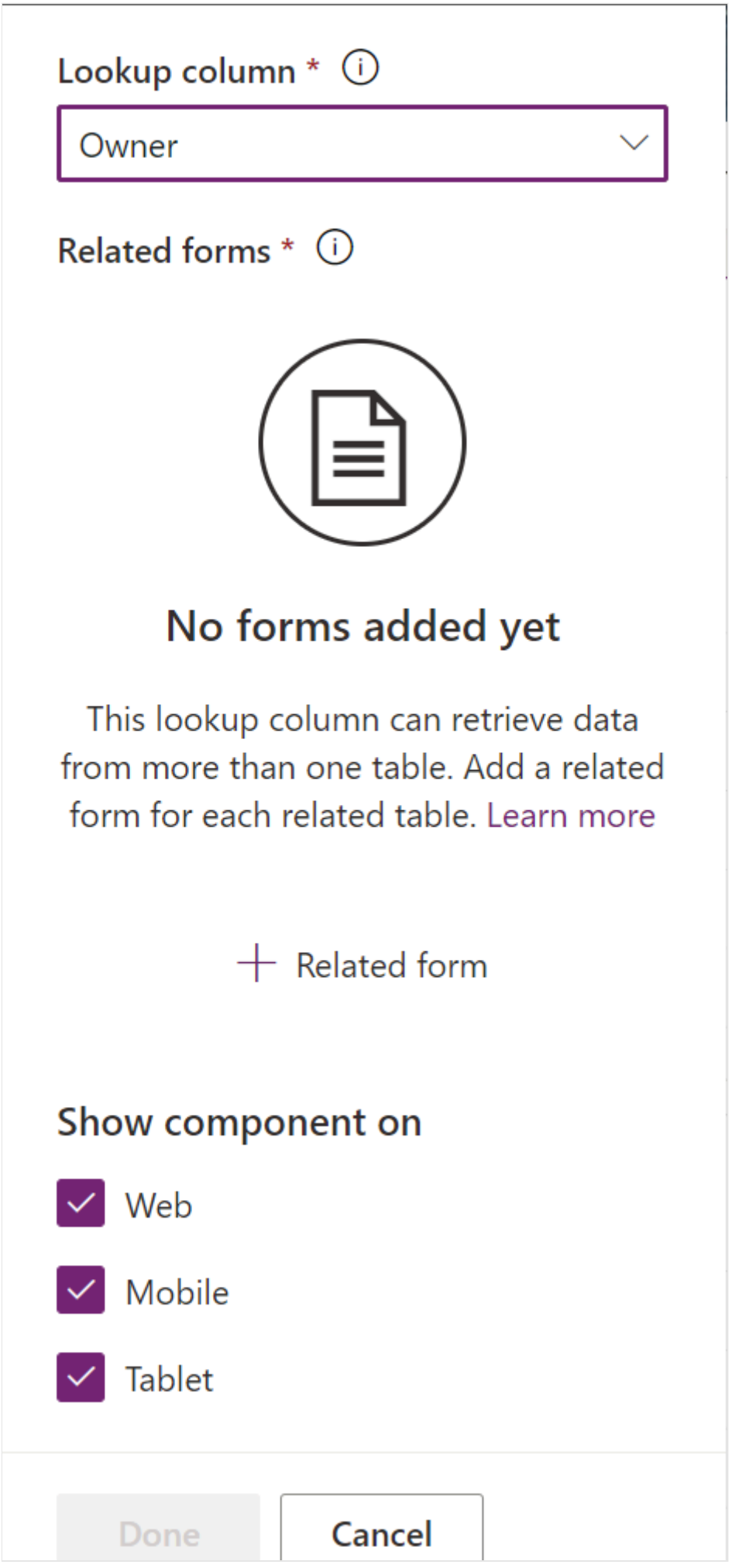
1. On the left navigation pane, select **Components**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Expand either **Input** or **Display**, and then select **Form**. On the form component control configuration page, select the properties for the control:

    - Select the **Lookup column** for the form.
    - Select the **Related form**. Depending on the related table configuration for the column:
        - If the column is connected to a single table (such as the Created By column), select the **Related form** to use.



- If the column is connected to multiple tables (polymorphic, such as the owner column), select **+ Related form** and then select **Add** to add the related form.

Continue to select + **Related form** > **Add** to add a related form for each related table.



- By default, all client app types **Web**, **Phone**, and **Tablet** are enabled to display the form. Clear the client types where you don't want the form displayed.

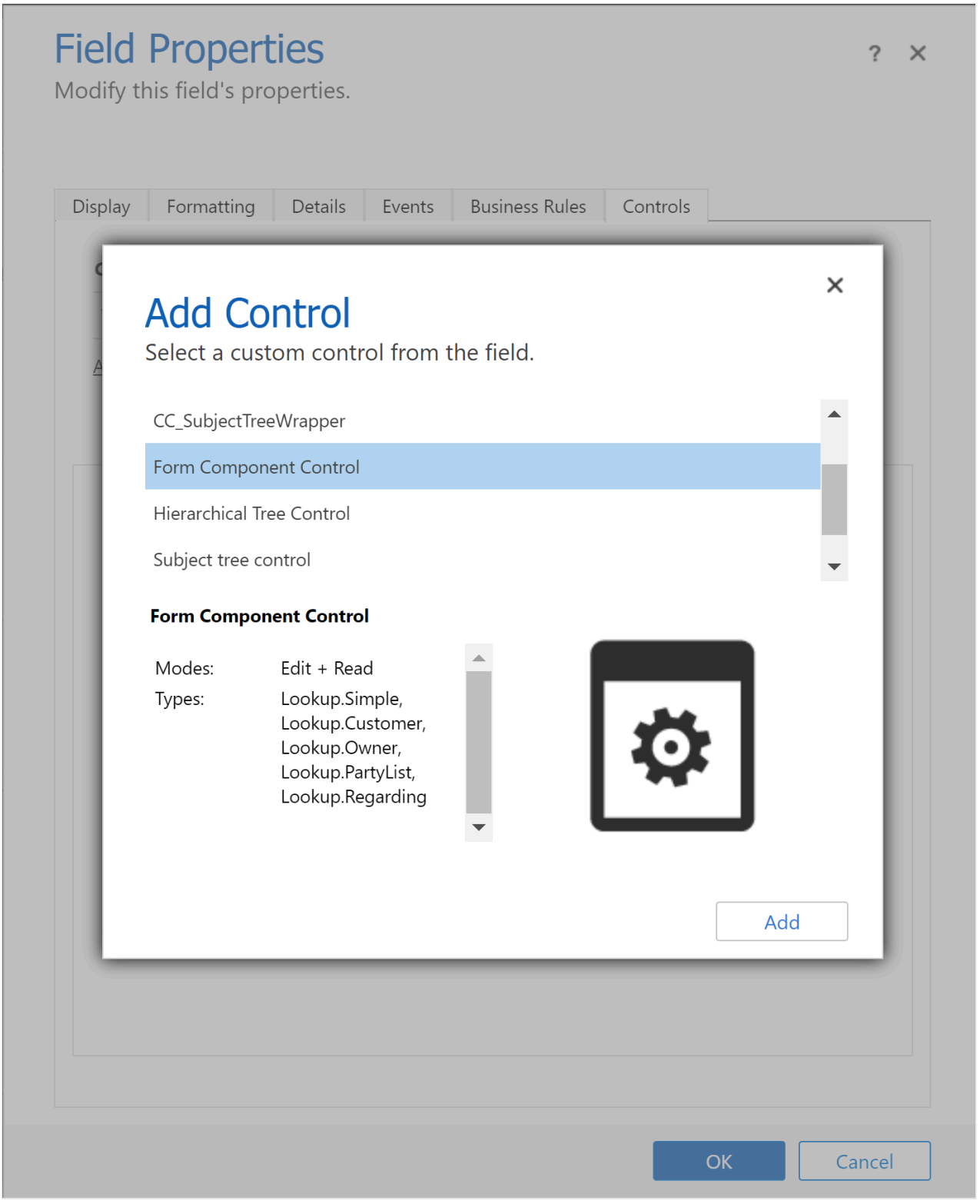
3. Select **Done**.
4. **Save** and then **Publish** the form.

Add the form component using the classic experience

In this example, the **Contact** standard main form is configured for the form component control that's added to the account main form.

> **Important**
>
> The way you add a form component control has changed. We recommend that you use the latest experience. More information: **Add the form component to a table main form**

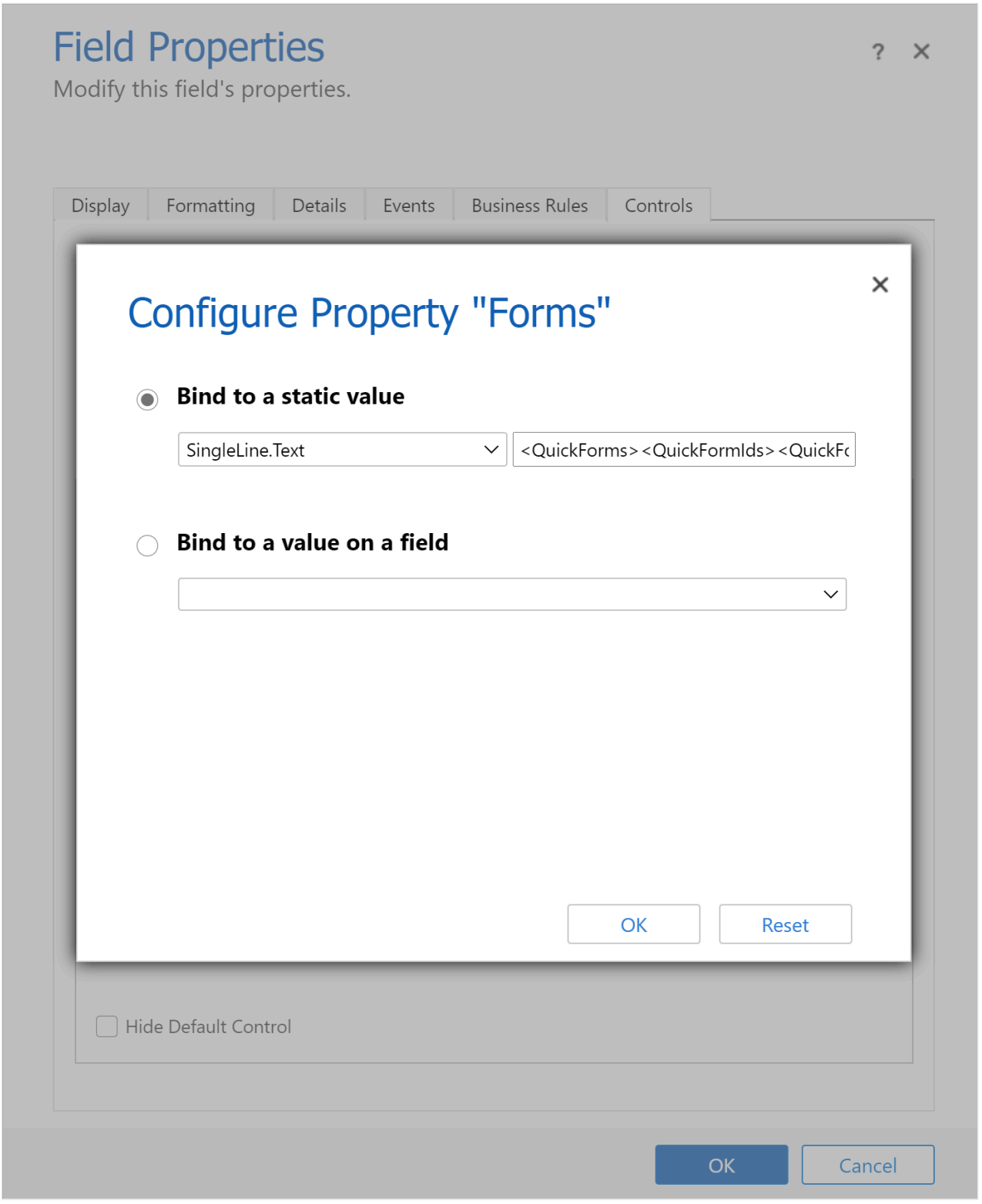
1. Sign in to Power Apps.
2. Select **Tables** in the left pane. Alternatively, open a solution, and then select a table, such as **Account**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Forms** from the components menu.
4. Select a form with a form type of **Main** from the forms available.
5. Select **Switch to classic**. The classic form editor interface opens as a tab in your browser.
6. Select the **Insert** tab. Then, create a new tab and add a new section or add a new section to an existing tab.
7. In the new section, add a lookup column, such as the **Primary Contact** lookup column.
8. Select the lookup column, and then on the **Home** tab, select **Change Properties**.
9. On the **Controls** tab, select **Add Control**, in the list of control types select **Form Component Control**, and then select **Add**.



10. Select **Web**, **Tablet**, and **Phone** for the component.

11. Select **Edit** (pencil icon) and on the **Configure Property** dialog box select **Bind to a static value** and then add an XML entry similar to this where *TableName* is the table unique name and *FormID* is the form ID for the main form: `<QuickForms><QuickFormIds><QuickFormId entityname="TableName">FormID</QuickFormId></QuickFormIds></QuickForms>`

    - For example, to render the **Contact** main form on the account form, use: `<QuickForms><QuickFormIds><QuickFormId entityname="contact">1fed44d1-ae68-4a41-bd2b-f13acac4acfa</QuickFormId></QuickFormIds></QuickForms>`



12. Select **OK**, and then select **OK** again.

13. **Save** and then **Publish** your form.

> **Tip**
>
> To find the unique name for a table, select the table in Power Apps and then select **Settings**. The **Name** appears on the **Edit** table pane. The form ID can be found in the

browser URL when you edit a form. The ID follows the **/edit/** portion of the URL.



In the classic form designer, the form ID follows the **formId%3d** portion of the URL.

Form component behavior

This section describes form component behavior when used in a model-driven app.

Record selection

In order for the form component control to show a form, the lookup column it's bound to needs to have a value. Otherwise, the control shows the message **Source record not selected**. One way to set the value is to add to the form a lookup control that is bound to the same lookup column as the form component control. When you use the lookup control to change the lookup column value, the form component control shows a form with the data for the new lookup column value.

Column validation

All columns, both in the main form and in the form component controls, must be valid for data to be sent to Microsoft Dataverse. This is true for both column validation errors, missing required columns, and so on.

`OnSave` handlers are run for the main form and its form component controls. Any handler can cancel the save for the main form and the form component controls by using preventDefault. This means no save operation can call `preventDefault` for data to be sent to Dataverse. The order of when the `OnSave` handlers are called isn't defined. More information: Form OnSave Event (Client API reference) in model-driven apps

Record save

Once the validation stage is passed, data is sent to Dataverse for each record. Currently, each record is updated independently with different requests. The saves aren't transactional, and the order of the saves isn't defined. An error saving one form component won't roll back changes to the main form or other form components. After each save is complete, data is refreshed for all the records on the form.

Notifications

Notifications on the form component are aggregated into the notifications of the main form. For instance, if there are invalid columns in the form component and you try to save, the invalid column notification appears at the top of the main form rather than in the form component.

Error handling

If there are multiple errors during save, only one error is shown to the user. If the user can make changes to fix the first error, and saves the next error will be visible. The user needs to continue saving until all errors are resolved.

Changing records with unsaved changes

If there are unsaved changes in a form for a form component and a user tries to change the lookup column the form component is bound to, the user is alerted about this change.

Client API

A form context is available for the form component control. It can be accessed via the main form's form context by accessing the control via an API, such as getControl. Before you access data for the related table in the form component control, event handlers should wait for the isLoaded API for the control to return true.

Limitations

Note the following limitations when you add the form component control to a table form:

- The form component control only supports rendering main forms. Similarly, support for adding a form component control is only supported with main forms. Other form types, such as quick create, quick view, and card aren't supported.
- Forms with a business process flow aren’t currently supported in either the main table form or the related table form. If you have a form with a business process flow, you might encounter unexpected behavior. We recommend that you don't use a form component with a form that uses a business process flow.
- The form component control doesn't support embedded form component controls, such as adding a form component control to a form that is used by a form component control.
- The form component control doesn't support embedded canvas apps. You might encounter unexpected behavior with these canvas apps.

- The form component control will only display the first tab of the form it uses if multiple tabs are included in that form.
- Using the same form for different form component controls on a single form isn't supported.
- The form that you use with a form component must be included in your app. If it isn't, or if the current user doesn't have access to the form, it falls back to the top-most main form that is included in the app and available to the user (based on form order).
- All components used by the form in a form component must be included in your app. These components include related tables, views, and business process flows. If they aren't, they won't be available in the form or there might be unexpected behavior.
- Form component controls have certain limitations when rendered in multi-session apps. Specifically, dynamically added handlers on the form component form such as `addOnSave` or `addOnChange` might not run after switching multisession tabs.
- You might notice that the timeline wall might not update when a column that is used to set the timeline wall has changed in the form component. When the page is refreshed the timeline wall updates as expected.
- On mobile, the timeline control doesn't currently appear in the form component control.
- For subgrids, the **See all records** and **See associated records** command buttons won't be available if they're rendered inside a form component.
- Form component controls aren't supported in bulk edit dialogs. They won't appear in the form in the bulk edit dialog by default and any changes made to related table records with them won't be saved.
- When duplicate records are detected when a form component control is saved, users might not be able to view the duplicates to resolve conflicts.
- Form validation only happens for the required fields that are rendered. Validation isn't possible for required fields that aren't visible or not rendered or present on another tab.

See also

Use custom controls for model-driven app data visualizations

Use Live monitor to troubleshoot model-driven app behavior

Article • 11/19/2024

Live monitor is a tool that can help app makers debug and diagnose problems, which help them build faster, more reliable apps. Live monitor provides a deep view into how an app runs by providing a log of all activities in the app as it runs.

To help you isolate certain model-driven app features, you can filter on specific categories, such as forms or the Copilot control.

Create a Live monitor session

Sign in to Power Apps, select **Solutions** on the left navigation pane, open the solution that contains the app you want to monitor, select the app, and then select **Live monitor** on the command bar.

On the Live monitor page, select **Play model-driven** app on the command bar. For more information about creating a Live monitor session, go to Use Live monitor to troubleshoot page behavior in model-driven apps.

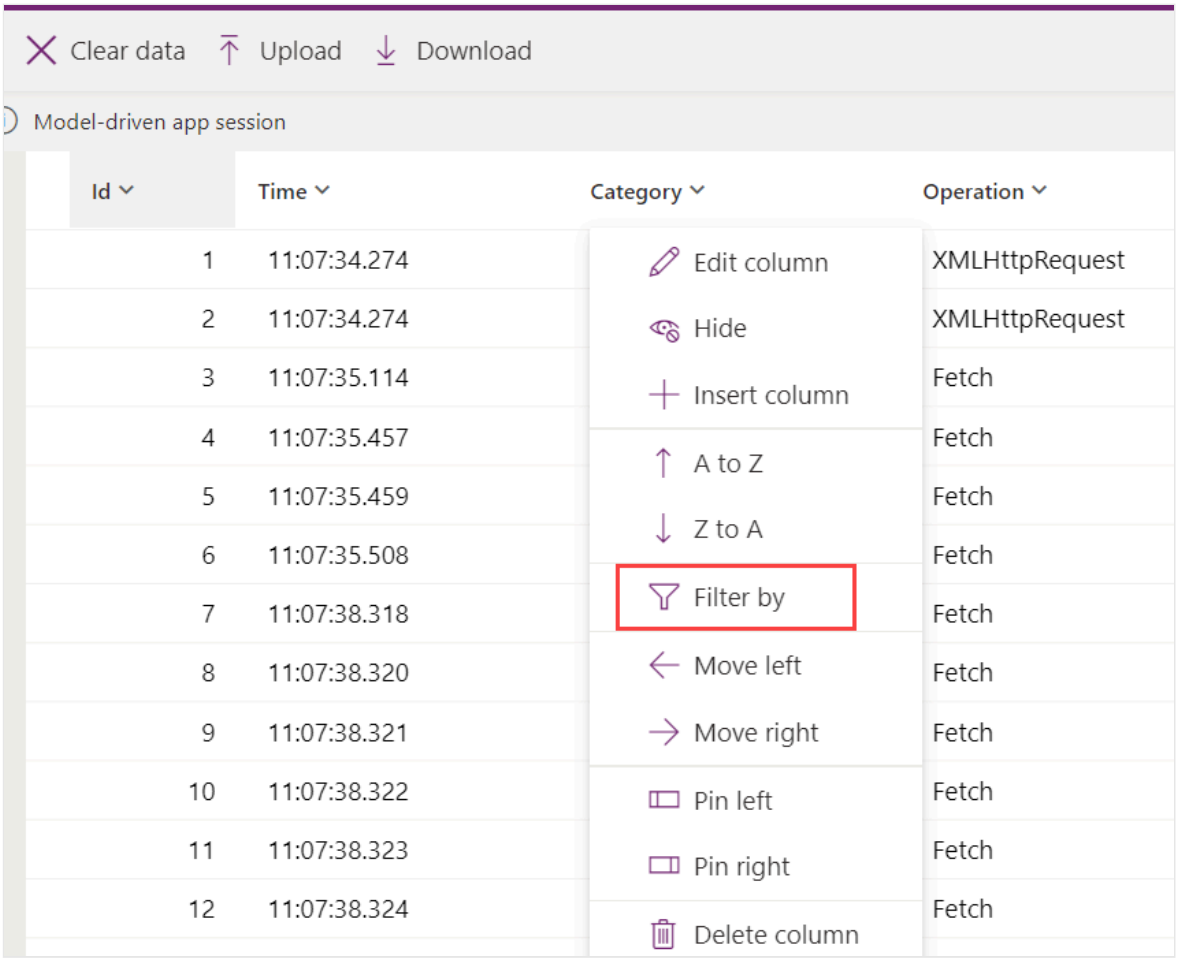
Filter Live monitor for form-related issues

Filtering on model-driven app form-related events in Live monitor can provide information about related tables, tables, controls, and components on a form in Live monitor as your app runs.

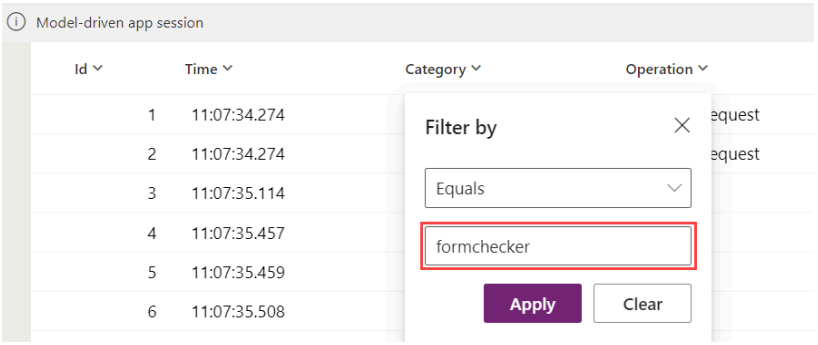
There are many situations where Live monitor can help makers understand why a form behaves a certain way. Many form issues are based on business rules, JavaScript, form events, or client API that admins and makers set. Live monitor can also help identify whether the issue experienced is designed out-of-the-box or is due to a customization. It provides details that can help answer the following questions:

- Why aren't rows showing in the related menu of a table?
- Why a control is disabled/enabled or visible/hidden
- Why is a row in a read-only state?

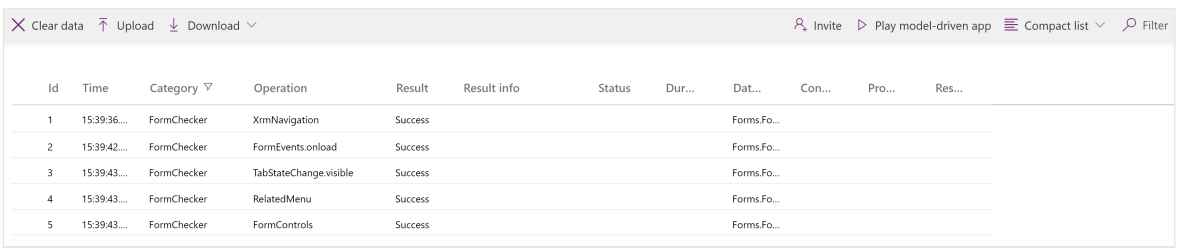
1. On the browser window running Live monitor, select the **Category** column, and then select **Filter by**.



2. Select **Equals** or **Contains** from the dropdown list, and then enter `formchecker` in the box. Select **Apply**.



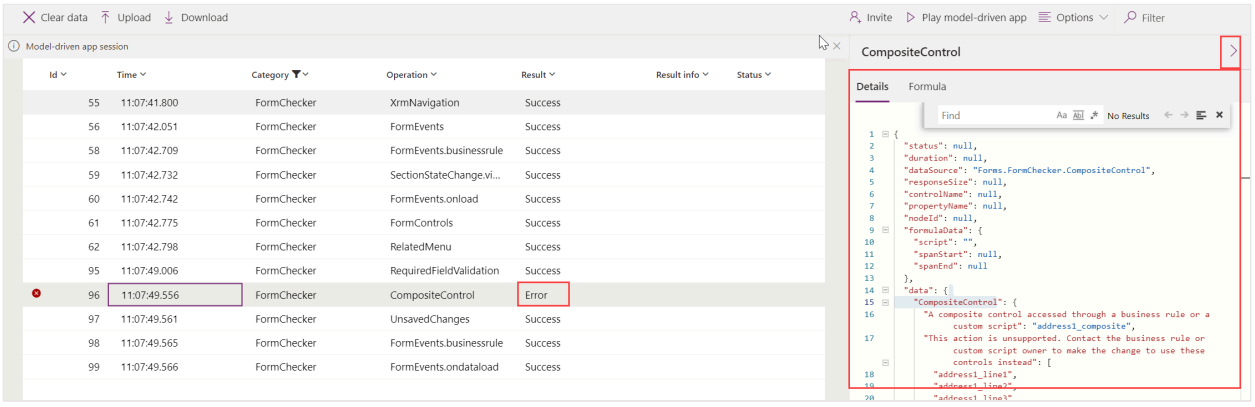
3. The categories are now filtered. The **Operation** column can be expanded to see the full name of the events that are tracked by selecting and holding the right side of the column and dragging to the right. As you use the app and open and use a form, Live monitor updates the list of events.



4. While the app is running in the monitored session, perform actions within the model-driven app consistent with normal use of the app. For example, open and change data using a table form.

Use Live monitor to understand form behavior

For each row with Live monitor, detailed information about the form event can be reviewed. For example, imagine you have a question about an error taking place within the form. You go to that form in the app and select the appropriate form component. Then return to the browser with Live monitor enabled and review the results either with or without filtering. In this case, there's an error on the composite control. By expanding areas of the **Details**, you can learn more about the event itself.



There are many types of events that are monitored, including the standard form events like `onload`, `onsave`, and `onclose`.

As you continue to use the app that's being monitored, Live monitor updates the information in the list of events. For forms, there are many different scenarios that you can troubleshoot and find additional information on the form, control, or table currently being worked on.

Form-checking areas and events

Areas for form monitoring include the following.

Expand table

App area	Description
Control state	Details about the state of the visible, enabled, and label source of a control when the form is loaded.
Related menu	Details about the state of related menu items. Examples: Why is a menu item not being displayed?

App area	Description
	Where does the menu item come from?
Tab / section / control state change	Details on who (via the callstack) has caused a form component—such as a tab, section, or control—to change the component's visibility and enabled state.
Navigation	Details about what's causing navigation or unexpected dialogs by tracing the callstack of these `Xrm.Navigation` client API methods: `openAlertDialog(), openConfirmDialog(), openDialog(), openErrorDialog(), navigateTo(), openForm(), openTaskFlow(), openUrl(), openWebResource()`
Unsupported customizations	Details about unsupported client API access before the form is ready. Examples: Accessing `parent.Xrm.Page` in iFrame before the form is fully loaded. Accessing `Xrm.Page` in a form web resource outside of form handler contexts using `window.setTimeout()` to periodically call the form client API. Accessing `Xrm.Page` in `updateView()` method of the Power Apps control framework control code.

Examples of the supported form-related events in Live monitor include:

- FormEvents.onsave
- XrmNavigation
- FormEvents.onload
- FormControls
- TabStateChange.visible
- RelatedMenu
- ControlStateChange.disabled
- ControlStateChange.visible
- SectionStateChange.visible
- UnsupportedClientApi

Filter Live monitor for Copilot control-related issues

1. On the browser window running Live monitor, select the **Category** column, and then select **Filter by**.
2. Select **Equals** or **Contains** from the dropdown list, and then enter `CopilotEvent` in the box. Select **Apply**.

Copilot control events

Events for Copilot control monitoring include the following.

Expand table

Event name	Description
AdaptiveCardActionData	Data sent to submit action for adaptive card
adaptiveCardContent	Content of adaptiveCard
AISummarize	This message is logged in case the summarize feature is off and logs the fcb status
AutoTriggerSummarizeHandler.isAutoTriggeredSummaryEnabled	Tells if auto trigger is enabled
ClientActionCalled	Logs the name of the client action and payload passed ot it
CopilotResponse	Response from the copilot
DVCopilotAction	isDVChatEnabled: if dv chat is enabled for the org or not
DynamicContextSentToBot	Dynamics context sent to bot
EnvironmentHostedBotSchemaName	Logs the environment bot schema name
ExecutingAction	Adaptive card action is getting executed
isSummaryEnabledForEntity	Tells if summarize is enabled for the entity
queryStructuredDataRequest	Request sent to query structure data
queryStructuredDataResponse	Response from query structure data
SessionContextSentToBot	Session context sent to bot
SummarizeResponse	Logs the response for the summarize
UserPrompt	Prompt entered by the User

Close a monitoring session

To close the monitoring session, close the browser tab where the monitored model-driven app is playing.

Next steps

For more information about how to troubleshoot issues with forms in a model-driven app, see Troubleshoot form issues in model-driven apps.

Learn about Live monitor as a Power Apps tool

Feedback

Was this page helpful? Yes No

Provide product feedback

Overview of custom pages for model-driven apps

The custom page is a new page type within a model-driven app, which brings the power of canvas apps into model-driven apps. Custom pages increase the convergence of model-driven and canvas apps and can be used to add full pages, dialogs, or panes with the flexibility of Power Apps Studio. It also includes a low-code page authoring experience with expressions and custom Power Apps component framework controls.

A custom page is more flexible than a model-driven app form, view, or dashboard page. You can include one or more tables. Then, you define the data and component interactions. The custom page is a separate solution element, which means one maker can edit one custom page at a time. Like other model-driven app pages, the page state comes from the parameters passed or is retrieved from persisted tables.

> **Important**
>
> Custom pages are a feature with significant product changes and currently have a few known limitations outlined in Custom Page Known Issues.

Expand table

Capability	Status	Notes
Runtime for custom pages	General Availability	
Solution and ALM for custom pages	General Availability	
Connectors in custom pages	General Availability	List of all Power Apps connectors
Modern controls in custom pages	General Availability	List of supported controls
Code components in custom pages	General Availability	
Monitor support for custom pages	General Availability	
Authoring custom pages	General Availability	Use the modern app designer and Power Apps Studio to author custom pages that are supported at runtime.

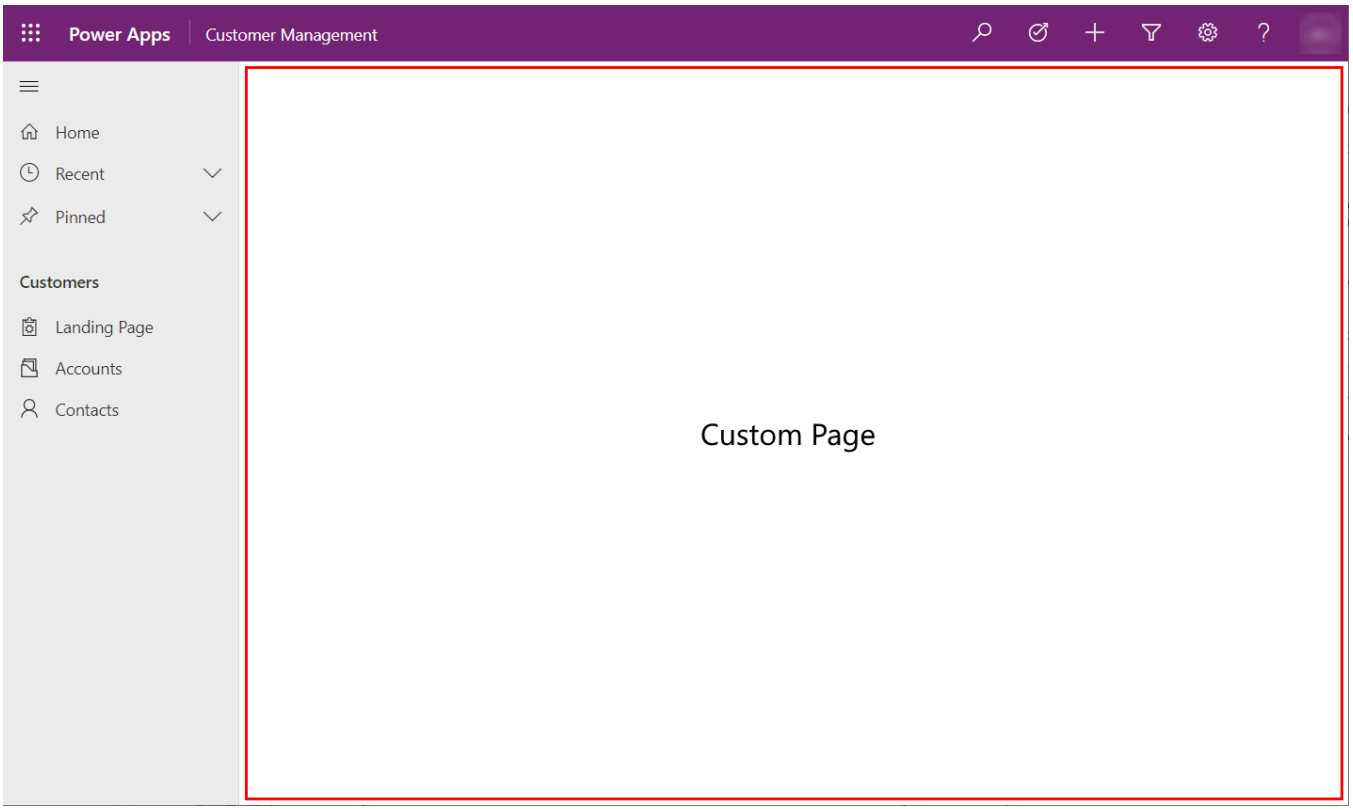
Capability	Status	Notes
Canvas components in custom pages	General Availability	
Custom page in Teams model-driven app	Public preview	
Custom page in mobile online	Public preview	iOS must allow enabling “Allow cross site tracking” that can be prevented by device management

Examples of custom pages

These images show the custom page inline within the model-driven app. The model-driven app has the full-page space in the images without the header and navigation.

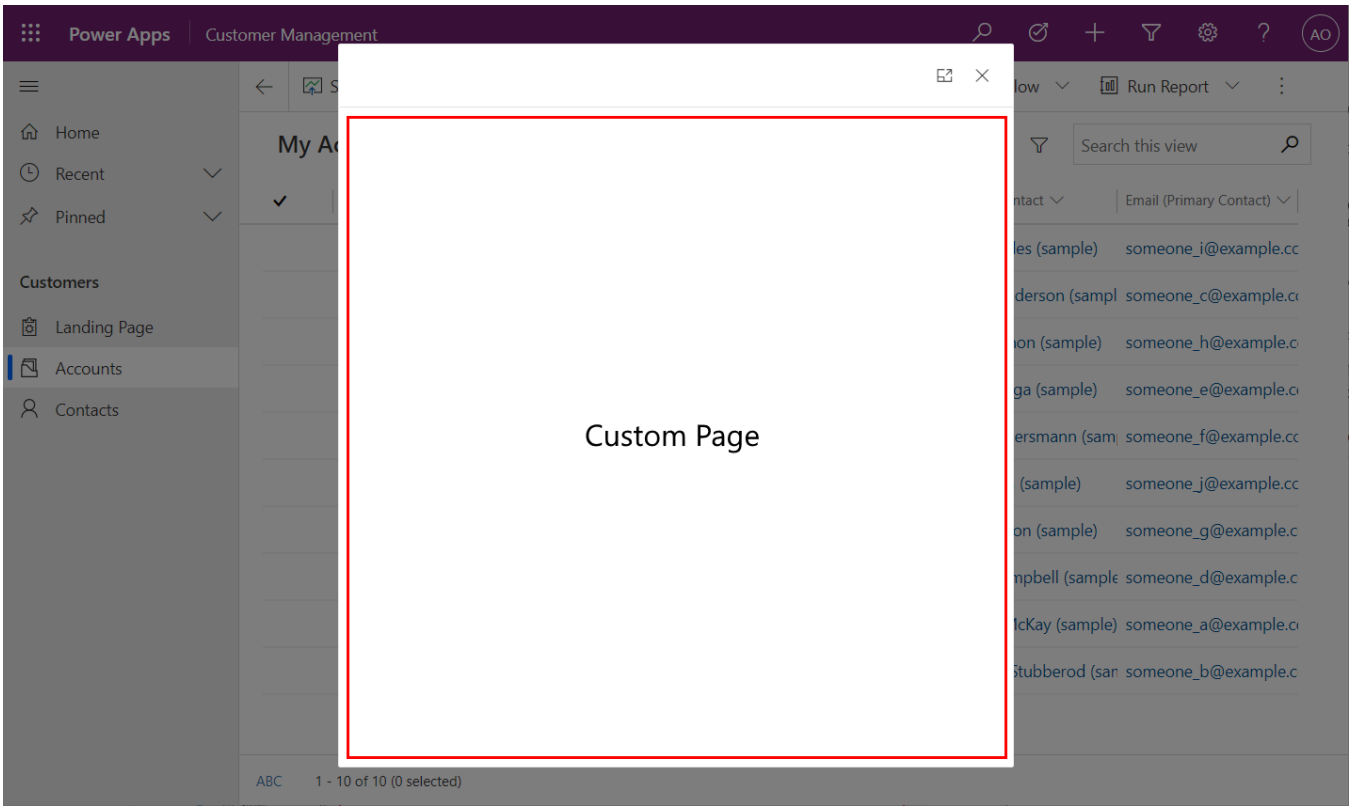
Custom page as main page.

[!div class="mx-imgBorder"]



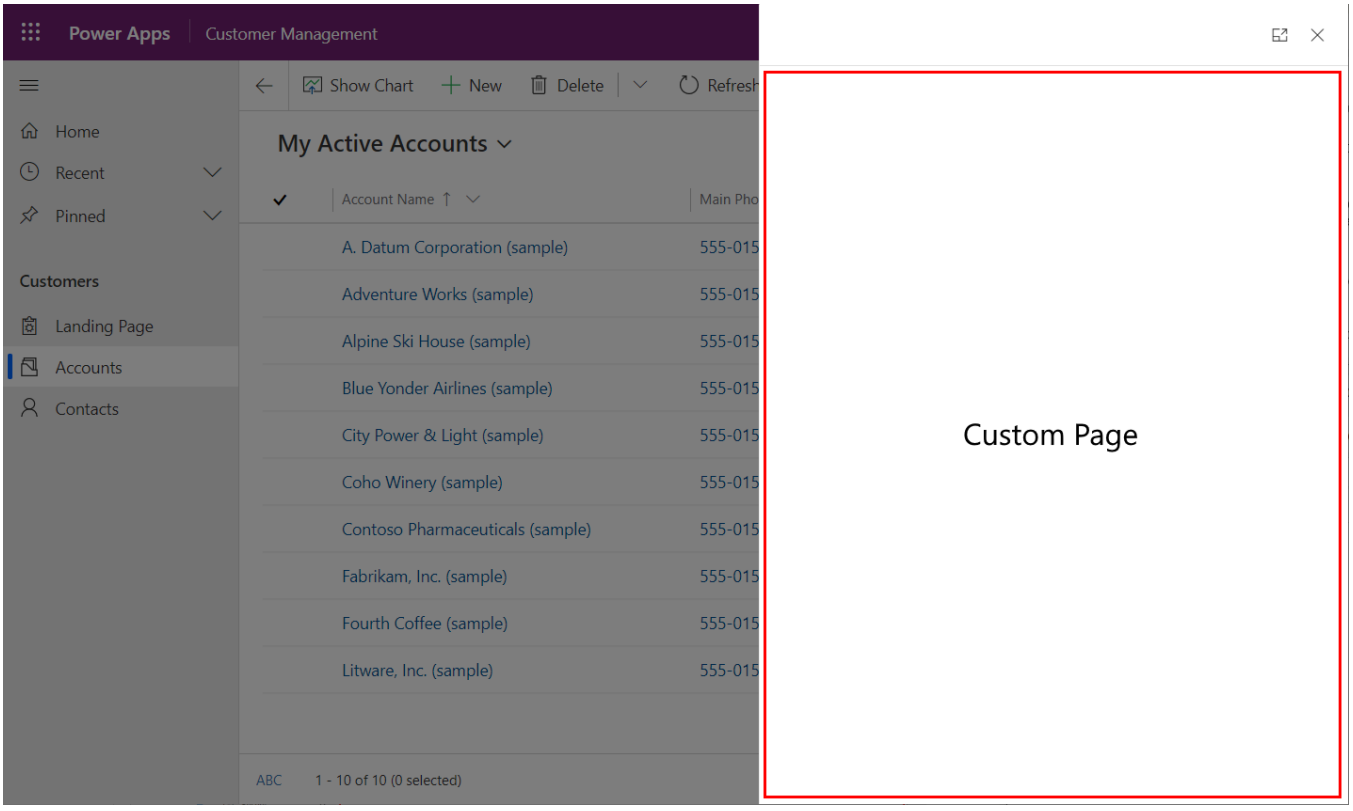
Custom page as a center dialog.

[!div class="mx-imgBorder"]



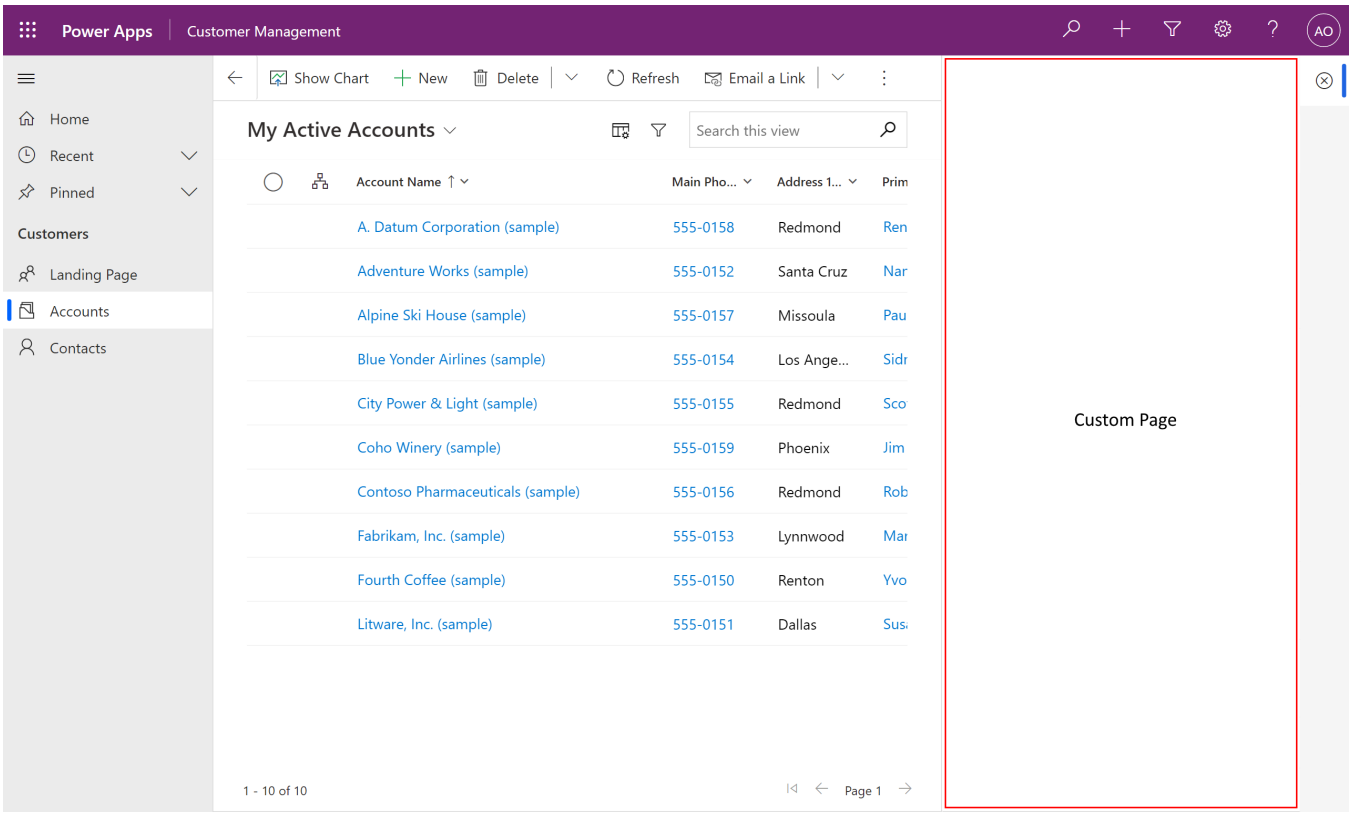
Custom page as a side dialog.

[!div class="mx-imgBorder"]



Custom page as an app side pane allows opening a custom page within the new app side pane on the right side of the app.

[!div class="mx-imgBorder"]



Custom pages must be created from a solution either from the modern app designer or the **Solution** area in Power Apps using **New** > **Page**. More information: Add a custom page to your model-driven app

Custom pages are different than embedded canvas apps

By using a custom page, you can create a new page experience by using the canvas app capabilities. This approach provides a low-code authoring experience with more flexible layouts, more control with styling options, the ability to add connector data, use expressions, and more. You create custom pages in Power Apps Studio with increasing context of the model-driven app that the page runs in.

Embedded canvas apps also use the canvas capabilities with a hosting approach that isn't as integrated as a custom page. The simpler integration of an embedded canvas app means the current limitation on number of embedded canvas apps hasn't changed. The advanced integration of the custom page addresses those limits. The embedded canvas app can only be placed on a model-driven form acting like a low-code component. More information: Embed a canvas app on a model-driven form.

In most cases, use custom pages instead of embedded canvas apps for tighter integration and better performance.

> **Note**
>
> Don't exceed 25 custom pages in a model-driven app. If you add more custom pages, it can increase the wait time that an app user experiences when they play an app with custom pages, after the last app publish. The increased user wait time affects only the first app launch, by the first user, after the last publish of the model-driven app.

Migrate standalone canvas app content to custom pages

Existing standalone canvas apps aren't supported for use as a custom page and the expected app structure is different. A standalone canvas app often has many screens with global access to all controls and variables. The custom page is expected to typically be a single screen with loose coupling to provide performance and co-development capabilities.

To migrate an existing standalone canvas app, start by identifying a mapping of screens to separate custom pages. For each separate custom page, complete these steps:

1. Create a blank custom page from the model-driven app designer. More information: Add a custom page to your model-driven app
2. Add a canvas app data source for data used by the screen.
3. Copy the screen from the original canvas app in Power Apps Studio.
4. Paste the screen into the blank custom page in Power Apps Studio.
5. Change the navigate calls to use the custom page name instead of the screen name.
6. Add the custom page into the model-driven app designer site map.

Frequently asked questions

- What data can the custom page use?

  A custom page can use Microsoft Dataverse and all of the connectors for Power Apps. More information: List of all Power Apps connectors.

- What interactions can the custom page have with the model-driven app?

  Custom pages can be added to the site map for direct navigation using Add custom page to sitemap. Model-driven app pages can open a custom page using the `navigateTo` Client API. Custom pages can navigate to other custom pages or to a model-driven app page such as a form, view, or dashboard with the Power Fx navigate function. More information: Navigating to a custom page

- How is the custom page made responsive?

  The responsive container controls enable building a responsive app page without formulas. More information: Building responsive pages. More custom page design guidance can also be found in Design a custom page for your model-driven app.

- How is the custom page managed in a solution?

  Each custom page is a separate component in the solution, which allows one maker to edit one custom page at a time. Most custom pages will have a single screen. Instead of multiple screens, they use the custom page's navigation functions to move to another custom page or model-driven app page. When a custom page has multiple screens, it's still a single solution component so only one maker can be working on the contained set of screens.

- What licenses are allowed to use a custom page and does a custom page impact app counts?

  The custom page uses a special canvas app type, which allows it to be managed differently. The custom page is considered part of the model-driven app infrastructure and can only be used within a model-driven app. So, it follows the license for the model-driven app. Also, custom pages don't count toward the app limits because they're treated as a page instead of an app.

- Do custom pages need to be shared like standalone canvas apps?

  The custom page aligns with the model-driven app page sharing, which relies on the model-driven app sharing without sharing individual pages for app users. Makers might need to share the custom page to allow editing.

See also

Add a custom page to your model-driven app

Design a custom page for your model-driven app

Using PowerFx in custom page

Add connectors into custom pages

Use Monitor to troubleshoot custom page

Model-driven app custom page known issues

Last updated on 03/31/2026

Add a custom page to your model-driven app

This article shows you how to create and edit a custom page for a model-driven app by using the modern app designer.

This article walks you through opening a model-driven app in the app designer, which you use to add a custom page to a model-driven app. If you need to create a new model-driven app, go to Create a model-driven app with the app designer.

For information about the limitations and known issues with custom pages, see Custom page known issues.

Create or edit a custom page

Custom pages can be created from two places. The first is while authoring a model-driven app in the modern app designer. The other is from the **Solutions** area in Power Apps. Custom pages can be edited from the **Solutions** area but won't appear in the **Home** or **Apps** areas of make.powerapps.com.

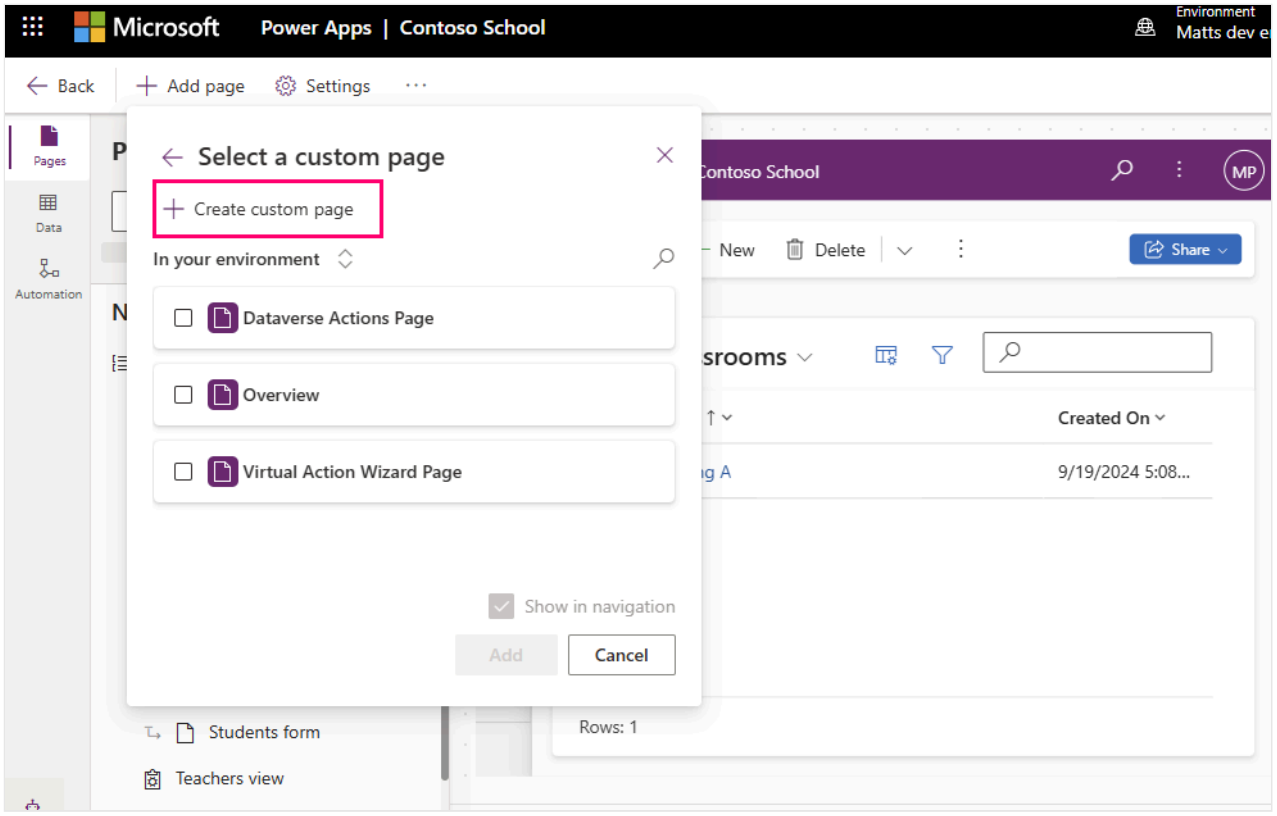
> **Note**
>
> Custom pages must be created from a solution either from the modern app designer or the **Solutions** area in Power Apps using **New** > **App** > **Page**. The custom page is a different canvas app type than the normal standalone canvas app.

Create new custom page from modern app designer

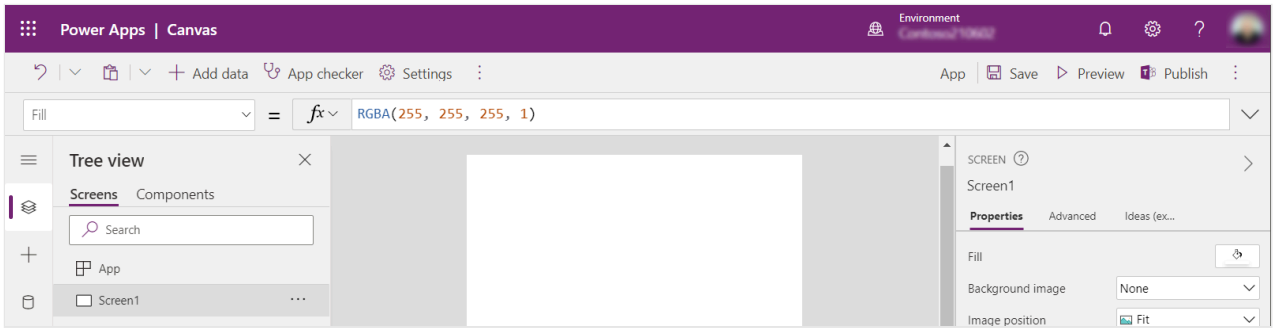
1. Open make.powerapps.com.
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution containing the existing model-driven app.
4. Select the model-driven app where you want to add a custom page, and then select **Edit** on the command bar.

5. In the app designer, select **Add page** > **Custom page** on the command bar.

6. Select **Create custom page**.



7. Power Apps Studio opens for page authoring.



8. When you're finished creating your canvas app custom page, **Save**, **Publish**, and then close the Power Apps Studio browser tab to return to the model-driven app designer.

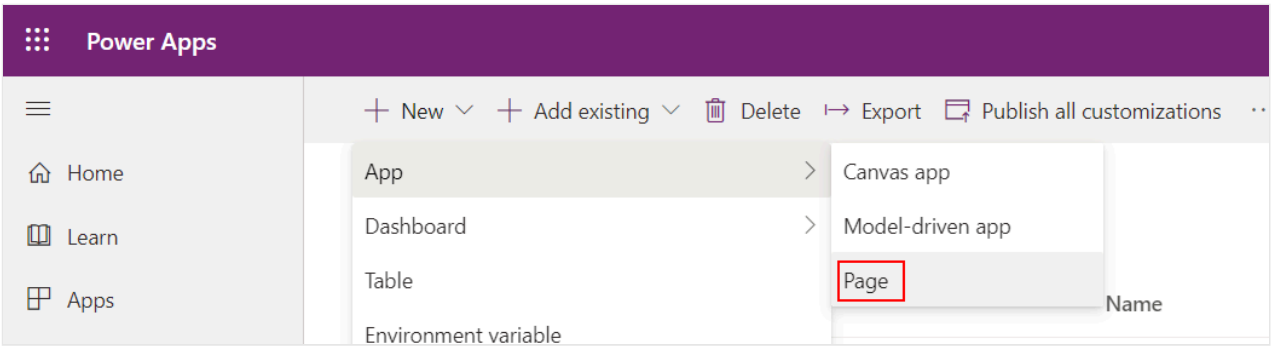
Create new custom page from the solutions area

1. Sign in to Power Apps.

2. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. Open or create a solution to contain the new custom page.

4. Select **New** > **App** > **Page**.



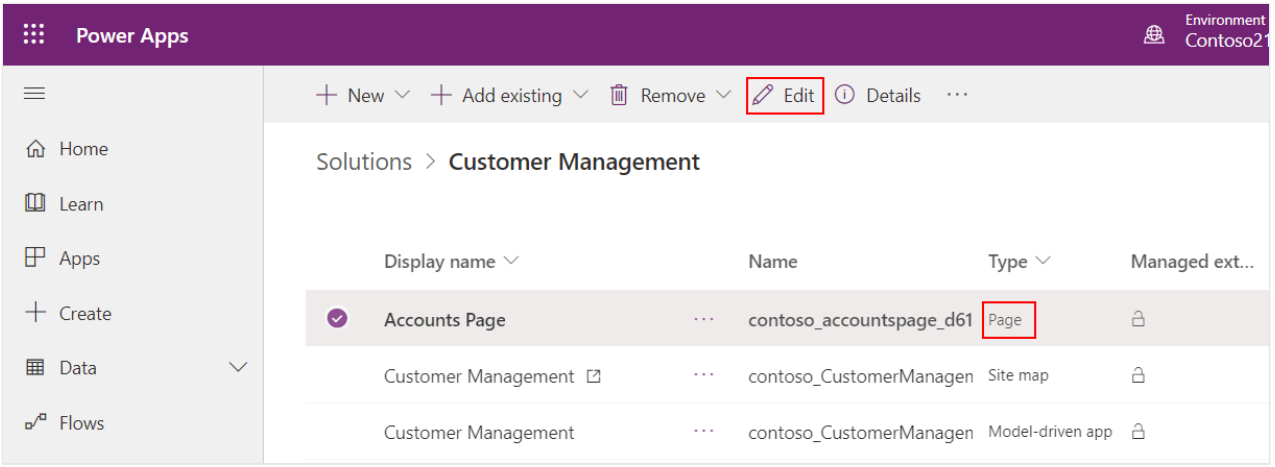
Edit an existing custom page

1. Open make.powerapps.com.

2. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. Open or create a solution to contain the new custom page.

4. Select the custom page with **Page** type, and then select **Edit**.



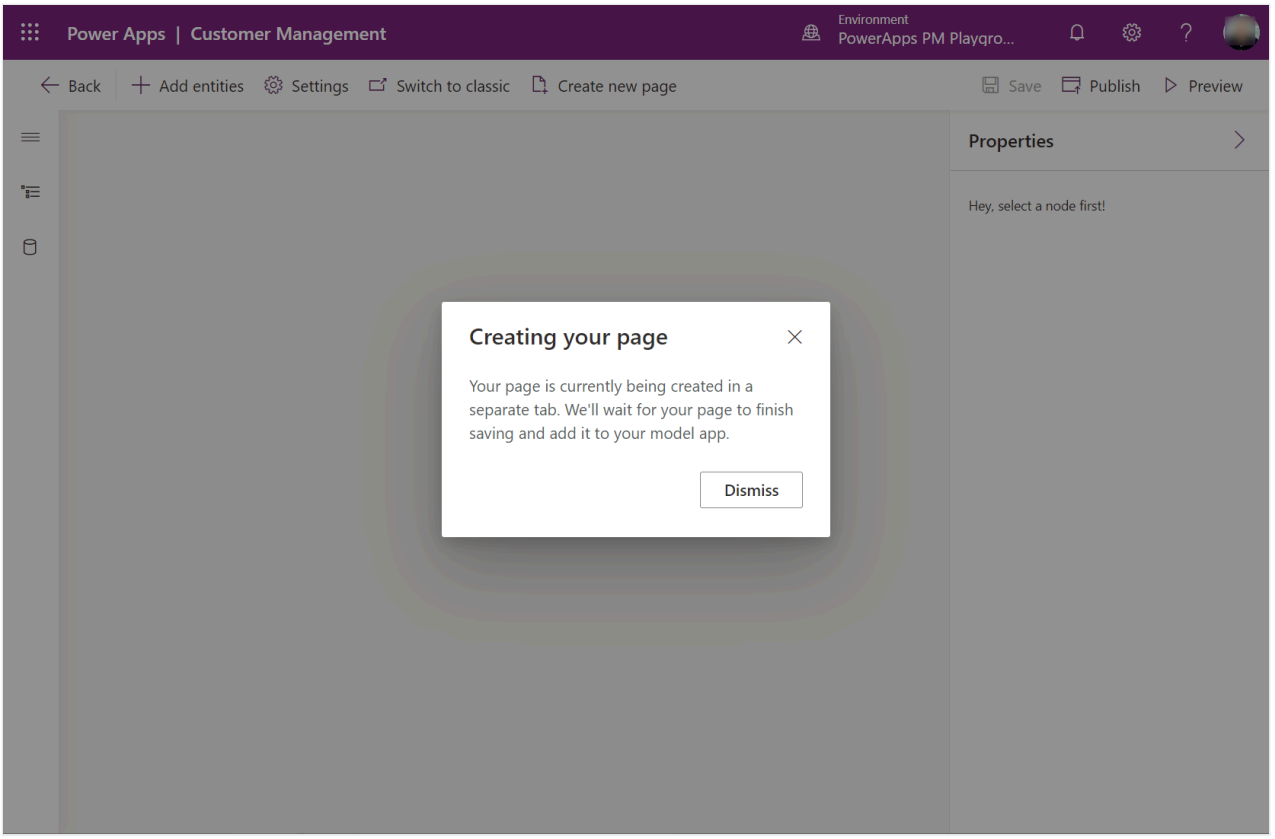
Author custom page content

1. Design the custom page content. More information: Design a custom page for your model-driven app

2. Save and publish the custom page.

3. Close Power Apps Studio.

4. Return to the app designer browser tab and refresh the app designer by selecting **Dismiss**.



5. Select **Publish** in the app designer to add the changed custom page into the model-driven app.

6. Select **Preview** to play the app in a new browser tab.

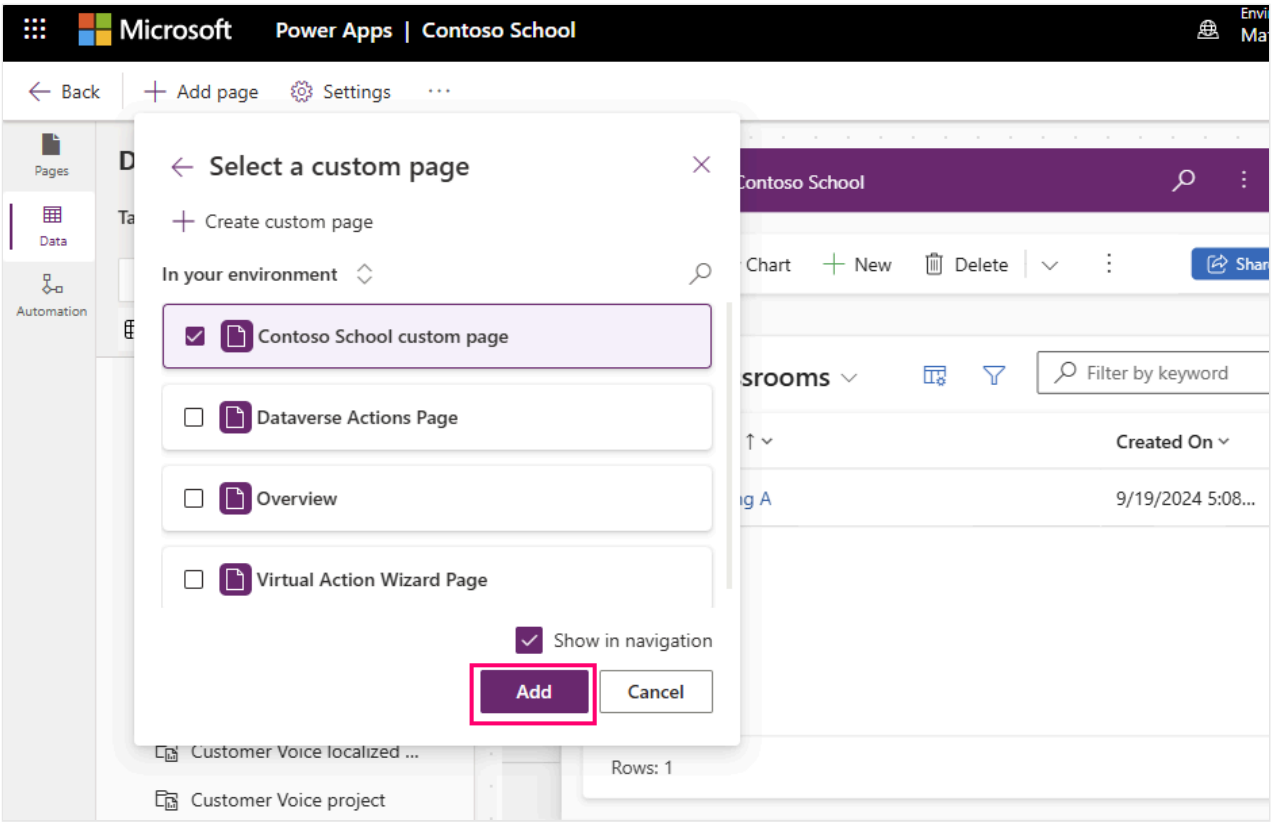
Add an existing custom page into a site map

1. Sign in to make.powerapps.com.

2. Open an existing model-driven app using modern app designer.

3. Select **Add page** > **Custom page** on the command bar.

4. Select the custom page in the list of custom pages available in the environment, and then select **Add**.



5. Select **Publish**, which also saves the app if there are changes.

6. Select **Play** to run the app in a new browser tab.

7. To close the app designer select **Back** and return to the solution.

Publishing a custom page

> **Important**
>
> Currently, model-driven apps must be re-published after a custom page is published. Otherwise the model-driven app continues to use the previous published custom page.

After saving changes to a custom page in Power Apps Studio, the custom page must be first published by Power Apps Studio. Then all model-driven apps referencing that custom page need to be published.

1. From Power Apps Studio, select **Publish**.

2. From app designer or solutions area, select **Publish** on each model-driven app referencing the custom page.

See also

Model-driven app custom page overview

Design a custom page for your model-driven app

Using PowerFx in custom page

Navigating to a custom page using client API

Code components for custom page designer

Last updated on 04/08/2026

Design a custom page for your model-driven app

Article • 01/29/2025

This article provides tips for designing a custom page for use in a model-driven app. A custom page lets you create a page that's similar to a canvas app in your model-driven app.

> **Important**
>
> Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in Custom page known issues.

Supported controls in a custom page

Custom page authoring currently supports a subset of canvas app controls. The table here lists the currently supported controls.

Expand table

Control	Control Type	Notes
Label[1]	Display	
Text Box[1]	Input	
Date Picker[1]	Input	
Button[1]	Input	
Combo Box[1]	Input	
Check Box[1]	Input	
Toggle[1]	Input	
Radio Group[1]	Input	
Slider[1]	Input	
Rating[1]	Input	
Vertical Container	Layout	New responsive horizontal layout container

Control	Control Type	Notes
Horizontal Container	Layout	New responsive horizontal layout container
Rich Text Editor	Input	
Gallery	List	
Icon	Media	
Image	Media	
Edit Form	Input	
Display Form	Input	
Code components	Custom	Add code components to a custom page
Canvas components (preview)	Custom	Add canvas components to a custom page

[1] Control is a new modern control. The control was introduced for canvas apps in Teams. The control is based on Fluent UI library wrapped with Power Apps Component Framework.

Custom components support for custom page

You can add both low-code (canvas components) and pro-code (code components) custom UX components to your environment and make them available for all makers. For custom page specific UX extensibility articles, go to add canvas components to a custom page for your model-driven app and add code components to a custom page for your model-driven app.

In general, the low-code extensibility approach is simpler to build, test, and has a lower maintenance cost. We recommend evaluating canvas components first and then use code components only if there's a need for more complex and advanced customization.

More information:

- Canvas component gallery
- Code components samples
- Code components community resources

Enable responsive layout with container control

Responsive custom page layouts are defined by building a hierarchy of **Horizontal layout container** and **Vertical layout container** controls. These controls are found in the canvas app designer under **Layout** on the **Insert** tab.

Set the minimum screen height and width on the **App** object to prevent page level scroll bars and use a vertical body scroll bar.

PowerApps Formula

```
MinScreenHeight=200
MinScreenWidth=200
```

Optionally, the custom page design size can be adjusted in **Settings** > **Display** with **Size** set to **Custom**. Then set the **Width** and **Height** to a more typical desktop custom page size like width 1080 and height 768. Changing this setting after controls are added to the screen might cause some layout properties to reset.

Set the topmost container to fill the entire space and resize based on available space.

PowerApps Formula

```
X=0
Y=0
Width=Parent.Width
Height=Parent.Height
```

Horizontal wrapping of a flexible height container

To support pages adjusting from desktop down to a narrow width, enable these properties on a horizontal container with flexible height. Without these settings, the page clips the control when the page is narrow.

PowerApps Formula

```
Direction=Horizontal
FlexibleHeight=true
Justify=Stretch
Align=Stretch
VerticalOverflow=Scroll
Wrap=True
```

Child containers or controls directly under this container should be set to have a minimum width that allows the page to fit within a 300 pixel width. Consider the padding on the container or control as well as parent containers.

Vertical wrapping of a flexible width container

To support pages adjusting from desktop down to a narrow width, enable these properties on a vertical container with flexible width. Without these settings, the page clips the control when the page is narrow.

PowerApps Formula

```
Direction=Vertical
FlexibleWidth=true
Justify=Stretch
Align=Stretch
HorizontalOverflow=Scroll
Wrap=True
```

Child containers or controls directly under this container should be set to have a minimum height that allows the page to fit within a 300 pixel width. Consider the padding on the container or control as well as parent containers.

More information: Building responsive layout.

Vertical container with fixed header, flexible body, fixed footer

1. On the **Vertical Container**, set **Align (horizontal)** to **Stretch**
2. Insert three **Horizontal Container** controls within the parent **Vertical Container**.
3. On the first and third child horizontal container controls, set **Stretch height** off and reduce height to the amount of space needed, such as *Height=80*.

Horizontal container with two even child containers

1. On the parent horizontal container, set **Align (vertical)** to **Stretch**.
2. Insert two **Vertical Container** controls within the parent **Horizontal Container**.

Styling custom page controls to align to model-driven app controls

By creating the custom page from the modern app designer, these features use the default values.

- Theme for the custom page. Theme values for the controls used in a custom page are automatically set to match the default blue theme of the Unified Interface. This default theme is used both in the studio and at application runtime. Explicit theme selecting is removed from custom page authoring experience.
- Controls need to use a different font size, which is based on their position in the page hierarchy.

  > **Note**
  >
  > Custom page text has an upscaling of 1.33 so you need to divide the target **FontSize** by 1.33 to get the desired size.

  Expand table

Label Type	Target FontSize	FontSize to use
Page title	17	12.75
Normal labels	14	10.52
Small labels	12	9.02

- Primary and secondary button controls need the following styling changes:

  Primary buttons

  PowerApps Formula

  ```
  Color=RGBA(255, 255, 255, 1)
  Fill=RGBA(41,114,182,1)
  Height=35
  FontWeight=Normal
  ```

  Secondary buttons

  PowerApps Formula

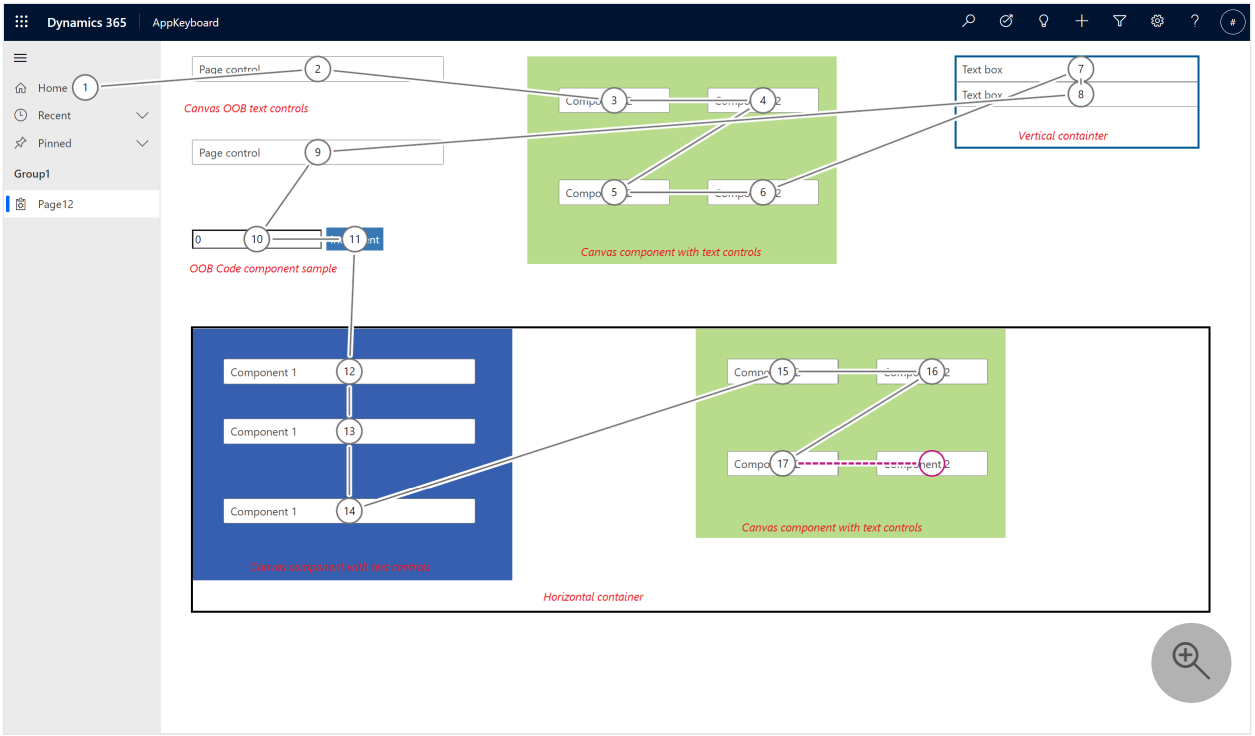
  ```
  Color=RGBA(41,114,182,1)
  Fill=RGBA(255, 255, 255, 1)
  BorderColor=RGBA(41,114,182,1)
  Height=35
  FontWeight=Normal
  ```

Tab navigation and keyboard accessibility for custom pages

Custom pages follow the same tab navigation design that's used by the hosting model-driven app. Visually aligned semantic HTML structure helps users navigate the custom pages seamlessly when using a keyboard or a screen reader. Unlike stand-alone canvas apps, custom page controls, and other UX elements don't need explicit tab numbers assignments. Modern controls don't have a `TabIndex` property and utilize the semantic HTML structure for navigation.

Various elements like controls, canvas and code components, containers, and so on, can be tabbed based on their position in the custom page layout. The tab navigation follows Z order navigation. Individual tab-stops inside larger grouping elements like components, containers are navigated first before the tab moves out to the next element in the document object model (DOM) tree.

Here's an example navigation with the page containing controls, code, and canvas components and containers.



> **Note**
>
> Overlapping controls and elements on the custom page don't have their DOM merged so tab stops can be out of sync from the visual layout. The same is true for the dynamic element positioning using formulas.

See also

Model-driven app custom page overview

Using PowerFx in custom page

Building responsive layout

Add a custom page to your model-driven app

Feedback

Was this page helpful? Yes No

Provide product feedback

Navigate to a custom page using client API

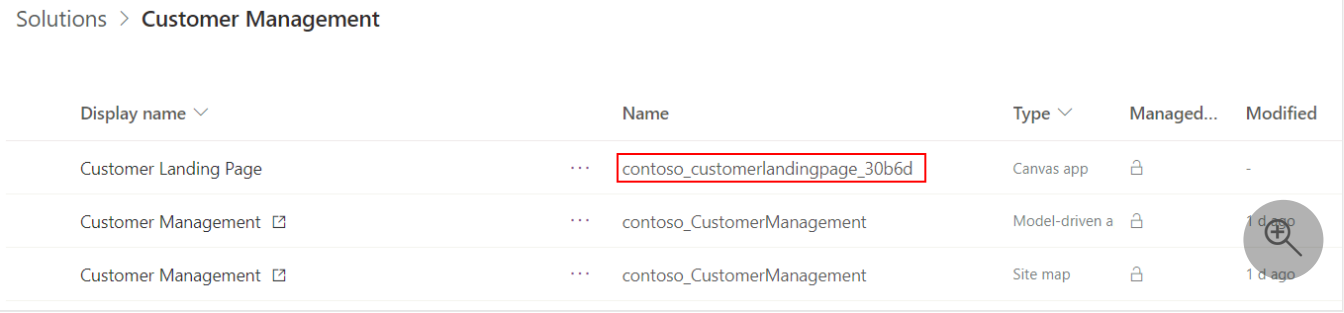
This article provides examples of navigating from a model-driven app page to a custom page using Client API. Learn how to open custom pages as full-page, dialog, or pane views in model-driven apps.

This article outlines the steps to use client API to open a custom page as a full-page, dialog, or pane. It provides examples of **custom** as a `pageType` value in navigateTo (Client API reference).

> **Important**
>
> Custom pages are a new feature with significant product changes. They currently have a number of known limitations outlined in Custom Page Known Issues.

Find the logical name

Each of the following client API examples takes the logical name of the custom page as the required parameter. The logical name is the **Name** value of the page in the solution explorer.



Open as an inline full page without context

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the page.

JavaScript

```
// Inline Page
var pageInput = {
    pageType: "custom",
    name: "<logical name of the custom page>",
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
```

```
    .then(
        function () {
            // Called when page opens
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open as an inline full page with a record context

This example uses the `recordId` parameter within the NavigateTo function to provide the custom page with the record to use.

JavaScript

```
// Inline Page
var pageInput = {
    pageType: "custom",
    name: "<logical name of the custom page>",
    entityName: "<logical name of the table>",
    recordId: "<record id>",
};
var navigationOptions = {
    target: 1
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when page opens
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

The `Param` function within the custom page retrieves the value and uses the Lookup function to retrieve the record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

> **ⓘ Important**
>
> The `recordId` parameter must be a GUID because it updates the URL and an app start from the URL validates the `recordId` is a GUID.

Open as a centered dialog

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the custom page. This mode supports the sizing parameters similar to the Main form dialogs.

JavaScript

```
// Centered Dialog
var pageInput = {
    pageType: "custom",
    name: "<logical custom page name>",
};
var navigationOptions = {
    target: 2,
    position: 1,
    width: {value: 50, unit:"%"},
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open as a side dialog

Within a model-driven app client API event handler, call the following code and update the **name** parameter to be the logical name of the custom page.

JavaScript

```
// Side Dialog
var pageInput = {
    pageType: "custom",
```

```
    name: "<logical page name>",
};
var navigationOptions = {
    target: 2,
    position: 2,
    width: {value: 500, unit: "px"},
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
    );
```

Open from a grid primary field link as a full page with record ID

This example uses the `recordId` parameter within the navigateTo function to provide the custom page with the record to use. The `Param` function within the custom page retrieves the value and uses the `Lookup` function to retrieve the record.

You can find a more complete example at Override the default open behavior of data rows in an entity-bound grid.

1. Create a web resource of type **JScript** and update the **name** parameter to be the logical page name. Add the following code to the web resource.

   JavaScript

   ```
   function run(selectedItems)
   {
       let selectedItem = selectedItems[0];

       if (selectedItem) {
           let pageInput = {
               pageType: "custom",
               name: "<logical page name>",
               entityName: selectedItem.TypeName,
               recordId: selectedItem.Id,
           };
           let navigationOptions = {
               target: 1
           };
           Xrm.Navigation.navigateTo(pageInput, navigationOptions)
   ```

```
            .then(
                function () {
                    // Handle success
                }
            ).catch(
                function (error) {
                    // Handle error
                }
            );
    }
}
```

2. Customize the table ribbon **CommandDefinition** for **OpenRecordItem** to call the function earlier and include the **CrmParameter** with the value **SelectedControlSelectedItemReferences**.

XML

```
    <CommandDefinition Id="Mscrm.OpenRecordItem">
        <Actions>
            <JavaScriptFunction FunctionName="run"
Library="$webresource:cr62c_OpenCustomPage">
                <CrmParameter Value="SelectedControlSelectedItemReferences" />
            </JavaScriptFunction>
        </Actions>
    </CommandDefinition>
```

3. In the custom page, override the **App**'s **OnStart** property to use the `Param` function to get the `recordId` and `lookup` record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

> **Note**
>
> After changing the `OnStart` property, you need to run `OnStart` from the App context menu. This function performs only once within a session.

4. Then, the custom page can use the **RecordItem** parameter as a record. The following example shows how to use it in an EditForm.

PowerApps Formula

```
EditForm.Datasource=<datasource name>
EditForm.Item=RecordItem
```

Open from a selected record in editable grid as a centered dialog with record ID

Use an editable grid to trigger the OnRecordSelect event when you want to perform an action after selecting a particular record in a view. This example uses the `recordId` parameter within the navigateTo function to provide the custom page with the record to use. The `getId` method in the GridEntity object retrieves the record ID. The `Param` function within the custom page retrieves the value and uses the `lookup` function to retrieve the record.

1. Enable editable grid control in the table.
2. Create a web resource of type **JScript** and update the **name** parameter to be the logical page name. Add the following code to the web resource.

JavaScript

```
function RunOnSelected(executionContext) {
// Retrieves the record selected in the editable grid
var selectedRecord = executionContext.getFormContext().data.entity;
var Id = selectedRecord.getId().replace(/[{}]/g, "");

// Centered Dialog
var pageInput = {
    pageType: "custom",
    name: "<logical page name>",
    recordId: Id,
};
var navigationOptions = {
    target: 2,
    position: 1,
    width: { value: 50, unit: "%" },
    title: "<dialog title>"
};
Xrm.Navigation.navigateTo(pageInput, navigationOptions)
    .then(
        function () {
            // Called when the dialog closes
        }
    ).catch(
        function (error) {
            // Handle error
        }
```

```
    );
}
```

3. In the custom page, override the **App**'s **OnStart** property to use the `Param` function to get the `recordId` and `lookup` record.

PowerApps Formula

```
App.OnStart=Set(RecordItem,
    If(IsBlank(Param("recordId")),
        First(<entity>),
        LookUp(<entity>, <entityIdField> = GUID(Param("recordId"))))
    )
```

> **Note**
>
> After changing the `OnStart` property, you need to run `OnStart` from the App context menu. This function performs only once within a session.

4. Then, the custom page can use the **RecordItem** parameter as a record. The following example shows how to use it in an EditForm.

PowerApps Formula

```
EditForm.Datasource=<datasource name>
EditForm.Item=RecordItem
```

Related articles

Model-driven app custom page overview
Add a custom page to your model-driven app
navigateTo (client API reference)

Last updated on 03/28/2026

Use Power Fx in a custom page for your model-driven app

Article • 01/23/2025

This article outlines how the common Microsoft Power Fx functions work differently between a standalone canvas apps and a custom page. Functions work differently because a custom page is a component within the model-driven app. Other Microsoft Power Fx formulas continue to behave in the same way.

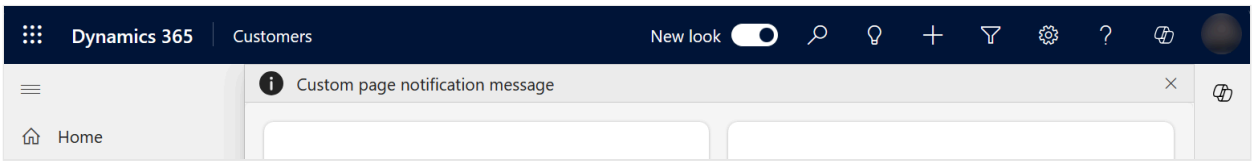
> **Important**
>
> Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in Custom Page Known Issues.

Add notifications to a custom page

A notification can be shown to the user in a custom page by calling the Notify function. When the notification messages appear, they're docked above the default page to stay visible until disabled. If a time-out interval is provided, the notification message disappears after the time out interval. We recommend that you don't use a time-out interval of 10, as this is no longer considered as a time-out interval. More information: Notify function.

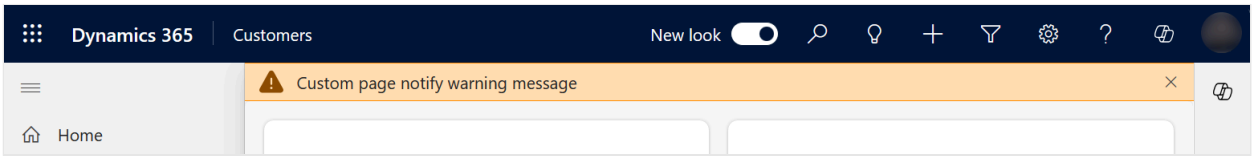
Power Fx

```
Notify( "Custom page notification message" )
```



Power Fx

```
Notify( "Custom page notify warning message", NotificationType.Warning )
```



Navigating to a custom page

This section provides examples of navigating from a model-driven app form to a custom page, navigating from a custom page to other custom pages or model-driven app form using Power Fx.

Navigating from a custom page

The Navigate function allows users to navigate either from model-driven app forms or custom pages. This function is only applicable when the custom page is running within a model-driven app. During custom page authoring or previewing in canvas designer, this function has no effect.

Navigate to another custom page

To navigate from one custom page to another, pass the display name of the custom page as the first parameter.

Power Fx

```
Navigate( CustomPage2  )
```

Navigate to the default view of the table

To navigate to the default view of the table, pass table name as the first parameter.

Power Fx

```
Navigate( Accounts )
```

> ⓘ **Important**
>
> Make sure you add the accounts Microsoft Dataverse table to the custom page before publishing and testing.

Navigate to specific system view of the table

To navigate to a specific system view of the table, pass the GUID of the view.

Power Fx

```
Navigate( 'Accounts (Views)'.'My Active Accounts' )
```

Navigate to the default form of the table

To navigate to the default form of the table, pass the record as the first parameter.

Power Fx

```
Navigate( Gallery1.Selected )
```

Navigate to a specific form of a table

To pass a Dataverse record to a specific form, pass the form name in the second parameter's `Page` attribute.

Power Fx

```
Navigate(
  AccountGallery.Selected,
  { Page: 'Accounts (Forms)'.Account  } )
```

Navigate to a specific custom page with a record input

To pass a Dataverse record to a specific custom page, pass the custom page name in the second parameter's `Page` attribute.

Power Fx

```
Navigate(
  AccountGallery.Selected,
  { Page: 'Account Record Page'  } )
```

In the target custom page, the record is retrieved using `Param` function to get the `etn` and `id` values.

Here's an example of loading the record into an `EditForm` control.

Power Fx

```
AccountEditForm.DataSource = Accounts
AccountEditForm.Item =
```

```
  LookUp( Accounts, accountid = GUID( Param("id") ) )
```

Navigate to the default form of the table in create mode

To navigate to the default form of the table in create mode, pass a Dataverse record created from the Defaults function. Defaults opens the default form with the record as a new record. The `Defaults` function takes the table name to create the record.

Power Fx

```
Navigate( Defaults( Accounts ) )
```

Navigate to the default form of the table in create mode with field defaulted

To navigate to a new record with some fields defaulted, use the `Patch` function to set fields on the default record for the table.

Power Fx

```
Navigate(
    Patch(
        Defaults(Accounts), { 'Account Name': "My company", Phone: "555-
3423" } )
  )
```

Navigate back to the prior page or close a dialog

To navigate back to the last page or to close a dialog, the Back function is called in a custom page. The `Back` function closes the current page and returns to the last model-driven app or custom page in the model-driven app. If the custom page has multiple screens, go to the article Navigating back when custom page has multiple screens.

Power Fx

```
Back()
```

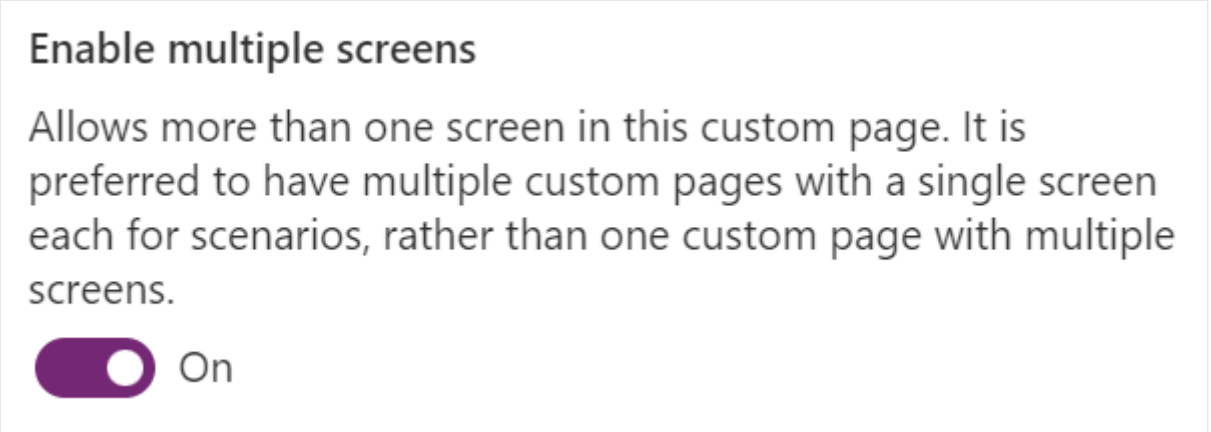
Navigating back when custom page has multiple screens

The default configuration for a custom page is to have one screen. In this case, the `Back` function call closes the custom page unless the custom page is the last in the page stack in model-driven app. The last page is kept open.

An app maker can enable multiple screens in a custom page. These should be considered like full page controls within the custom page that can be stacked. Opening a custom page has no means of specifying the screen to use. When a custom page contains multiple screens, the maker is responsible for managing the screen stacking. Calling the `Navigate` function to a screen adds to the screen stack with the custom page. Each `Back` function call removes a screen from the screen stack. When there's only one screen on the screen stack, the custom page is closed.

Enabling multiple screens

By default a custom page uses a single screen to encourage separation of the app into a screen per page. Single screen can be switched by enabling **Settings** > **Display** > **Enable multiple screens**.



Known issues

- The `Navigate` function doesn't have support for opening a model-driven app or custom page to a dialog. All navigation from a custom page opens inline.
- Navigate function doesn't support opening:
  - A dashboard collection or a specific dashboard.
  - A specific model-driven app form.
- A custom page can only open into the current session's current app tab in a multi-session model-driven app.

See also

Navigating to a custom page using client API

Model-driven app custom page overview

Feedback

Was this page helpful? Yes No

Provide product feedback

Add canvas components to a custom page for your model-driven app

Article • 06/01/2022

This article outlines the use of low-code canvas components to build custom UX for a custom page by utilizing shared canvas component library capability. For code first custom UX extensibility, see Add code components to a custom page for your model-driven app.

> **Important**
>
> Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in **Custom Page Known Issues**.

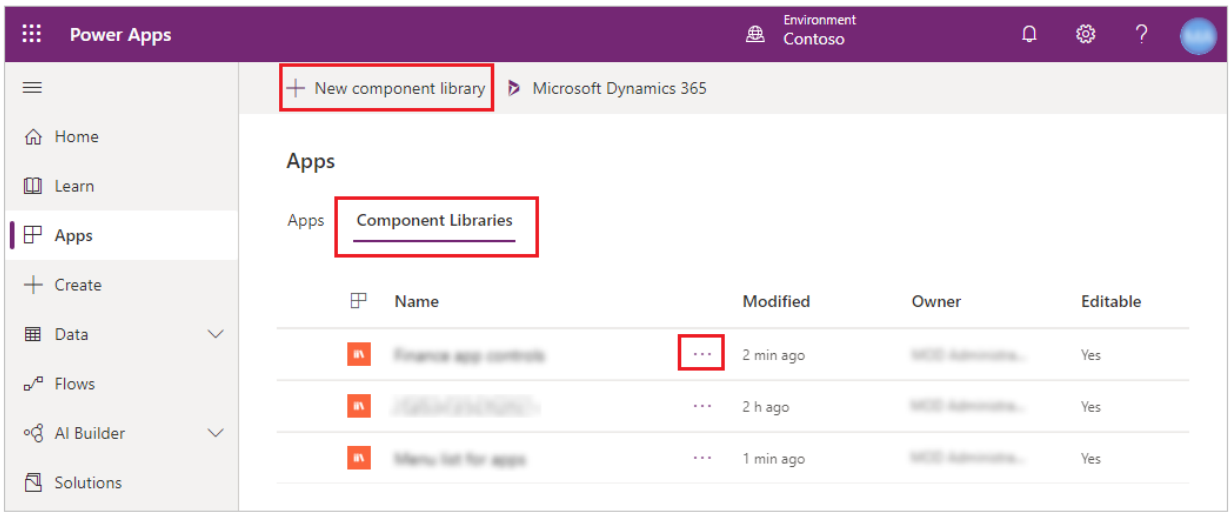
> **Note**
>
> - Custom page currently supports a **limited set of controls** and only the currently supported controls should be used to create canvas components for the custom page.
> - **Modern controls** are currently only supported with custom pages and not with standalone canvas apps. Make sure that the modern control experimental setting is used only for component libraries used with custom pages.

Canvas components provide app makers the ability to create custom components in a low-code fashion. These components cannot only then be reused across custom pages and applications, but also can be centrally updated, packaged, and moved using Microsoft Dataverse solutions. More information: Create a component for canvas apps

Since custom page authoring is limited to only one page, canvas components can only be authored inside a component library. This is different from the standalone canvas apps, which have the ability to create components at the app level.

Create a canvas component using the component library

You can create a new component library or edit an existing one either from the **Solutions** area or the **Component Libraries** tab in the **Apps** area.

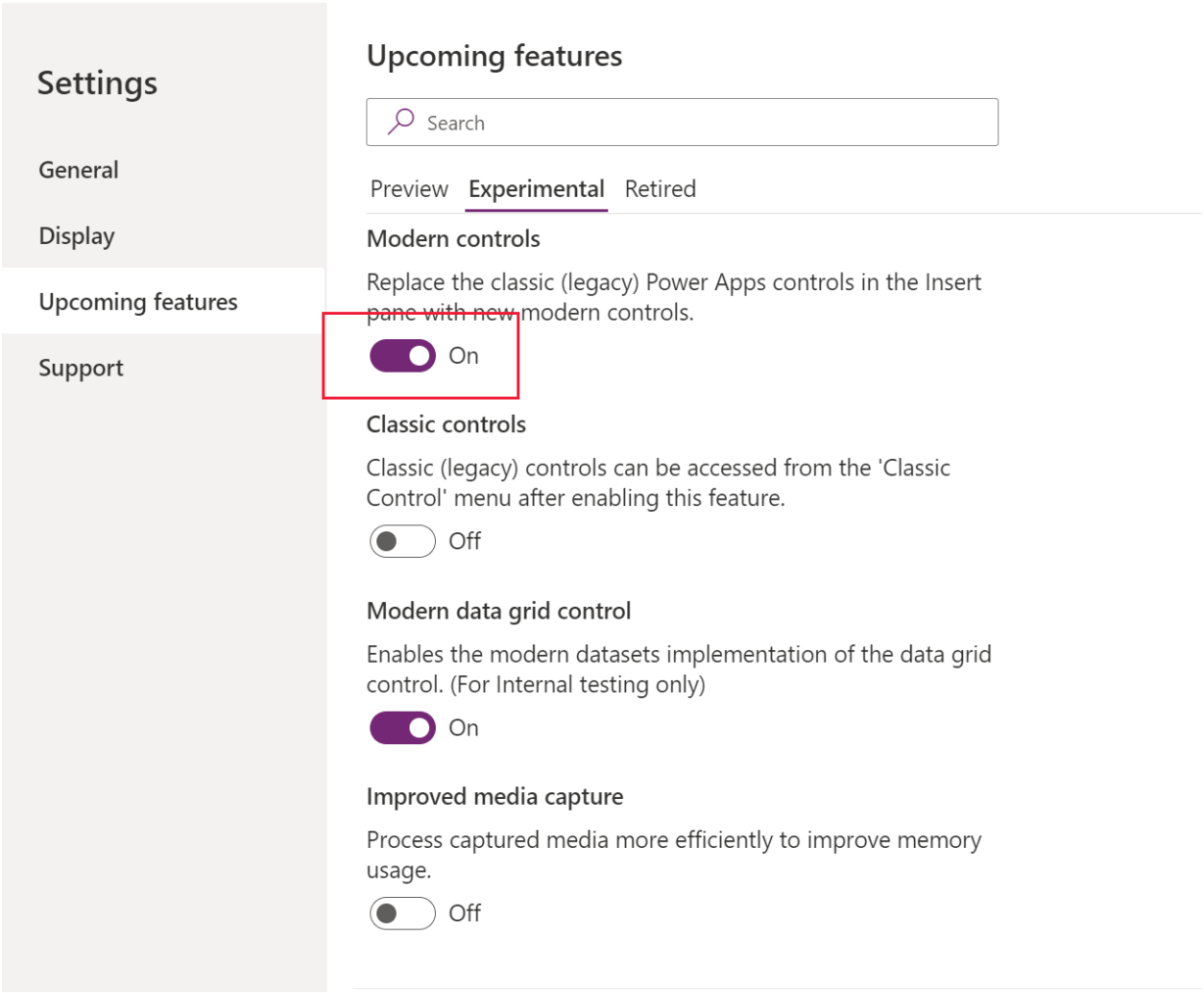


Create a canvas component for use in a custom page

This section explains how to create a canvas component that you can use in a custom page.

Enable modern controls for the component library

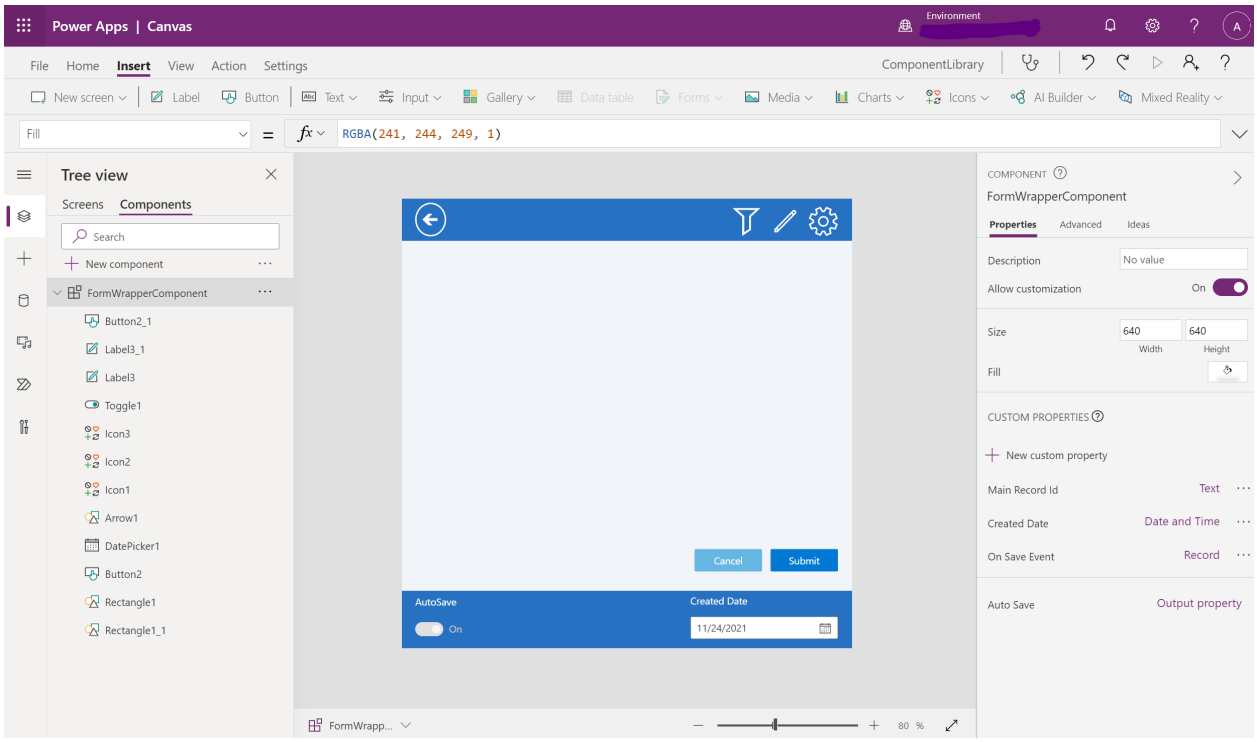
First, enable the **Modern controls** setting in the canvas app designer. To do this, go to **File** > **Settings** > **Upcoming features** > **Experimental**.



Make sure that only the supported set of controls are used to create components for custom page.

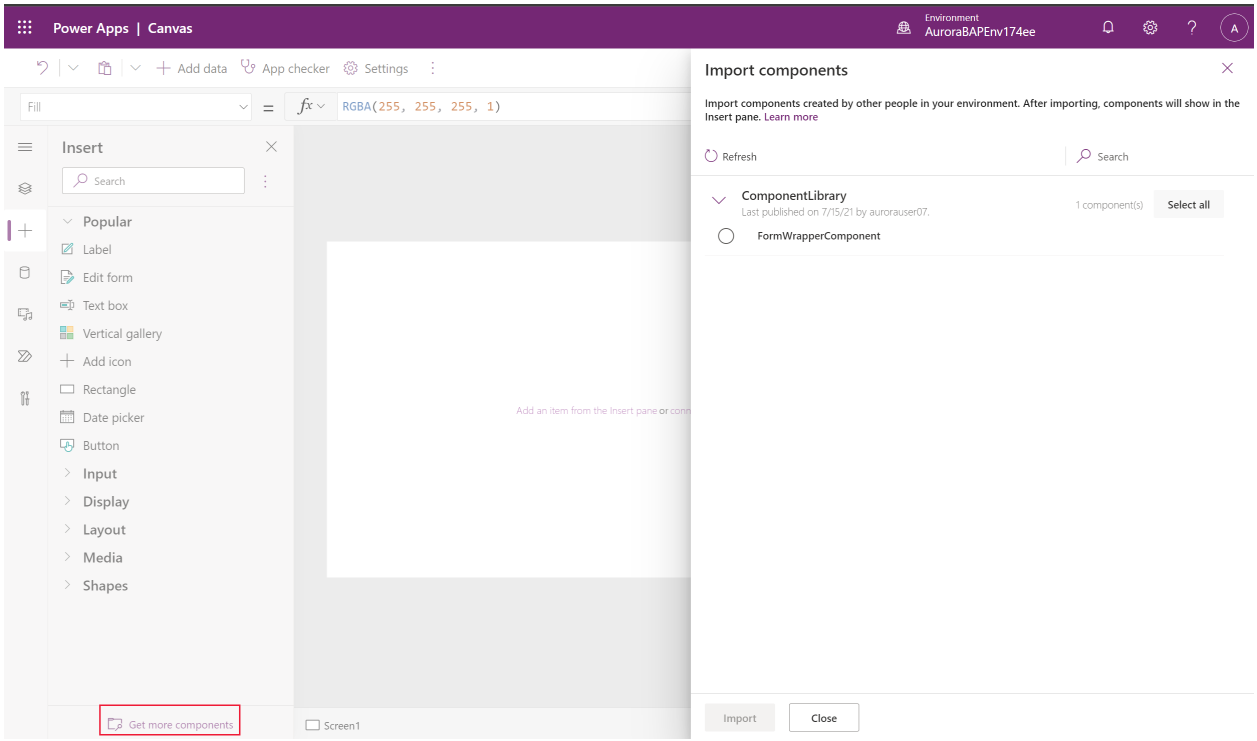
Create a canvas component

Once the modern controls are enabled, you can now create desired components inside the library and publish them once the components are ready.

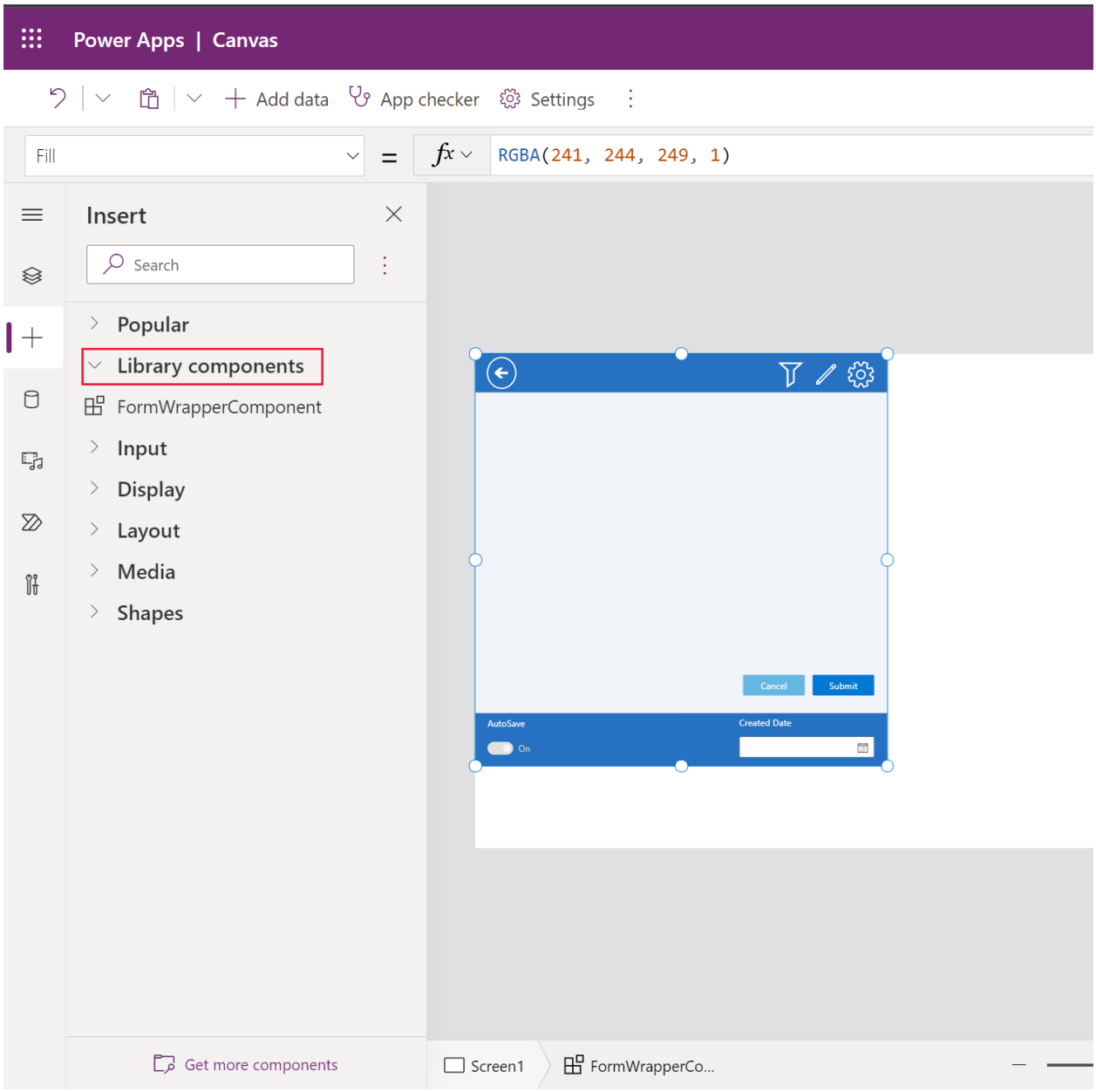


Import and use a canvas component in a custom page

Custom pages can use the components from the canvas component library created or imported in the current environment. You can select **Get more components** at the bottom of the add control left navigation area.

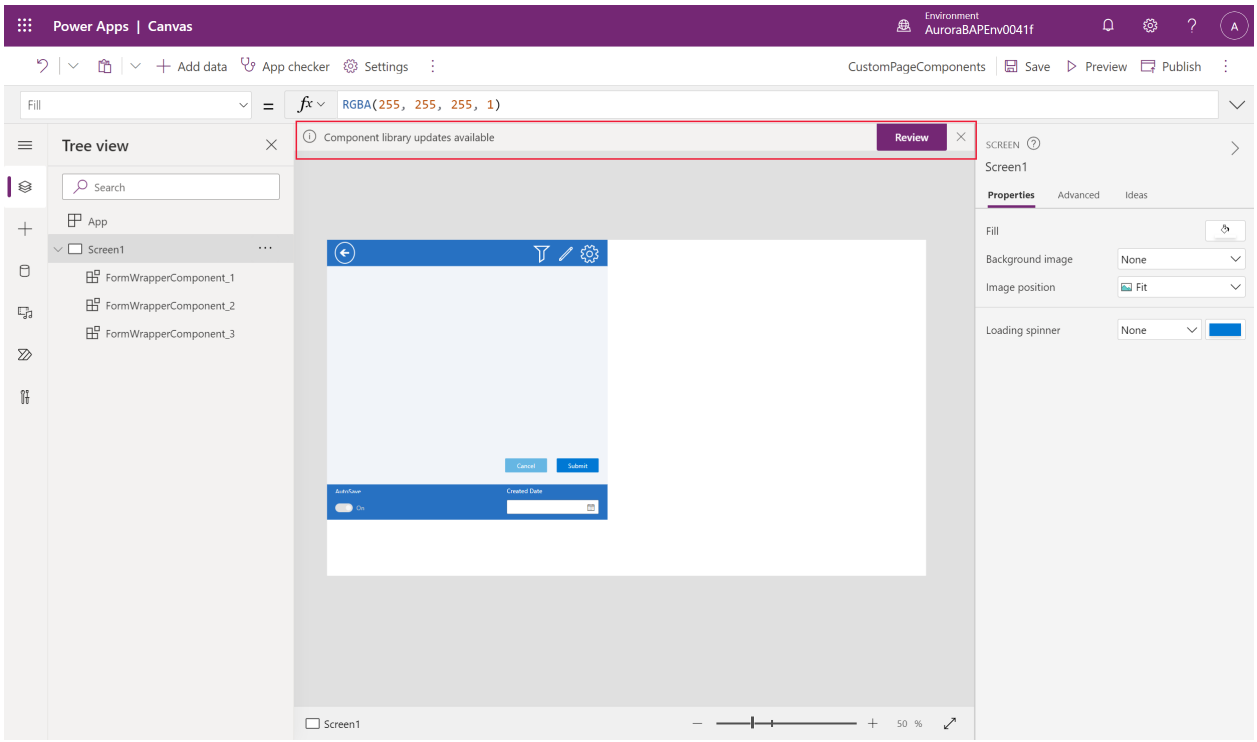


The canvas component is available under the **Library components** section and can be added to the custom page.



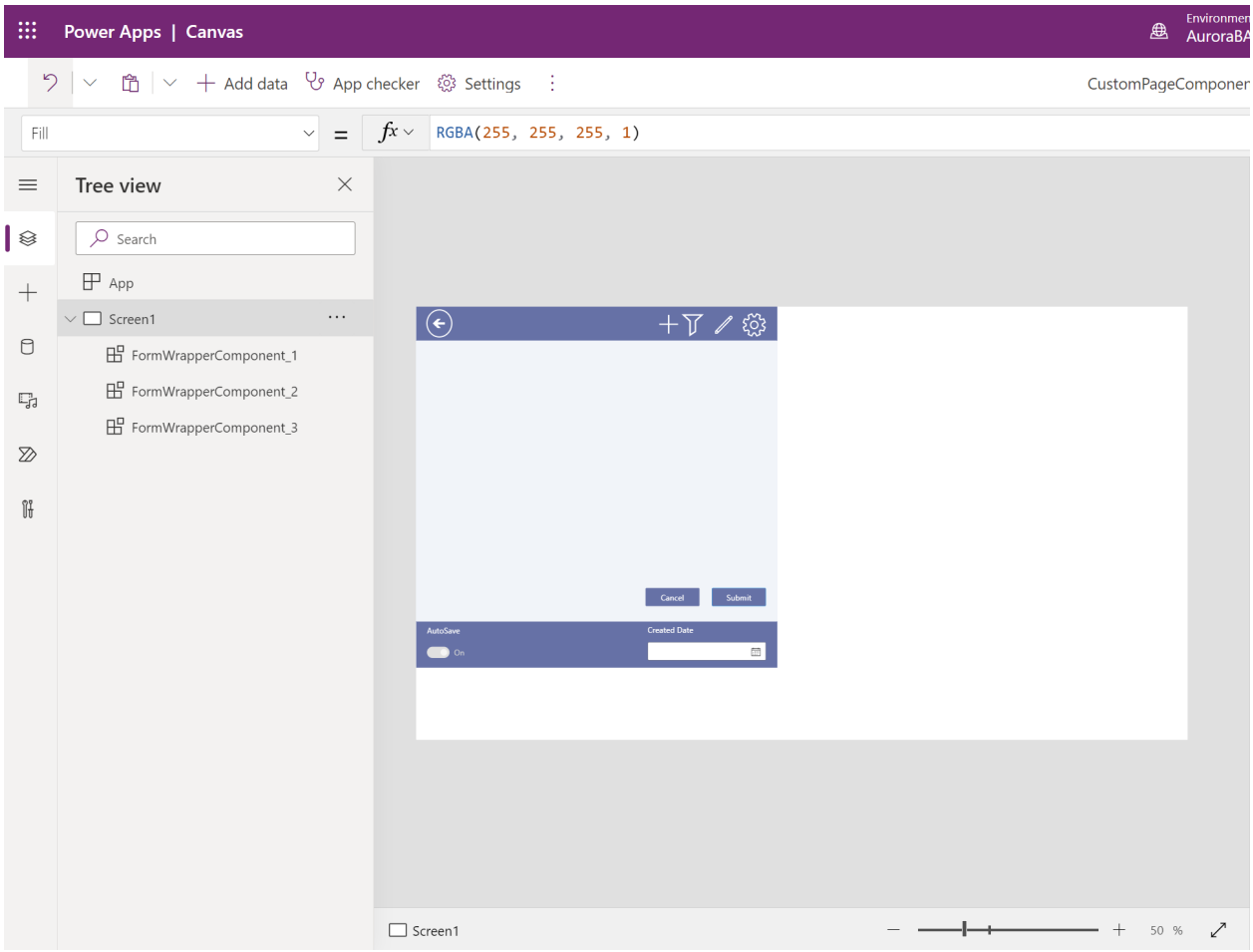
Component updates to the custom page

When the component is added to the custom page via the component library, it maintains the reference to the library. When the library owner publishes the newer library version with updated components, the custom page inline with the canvas app displays an update available message. This message is displayed when the custom page is opened for editing.



Select **Review** to review the updates, and then select **ok** to get the component refreshed with the latest changes from the canvas library.

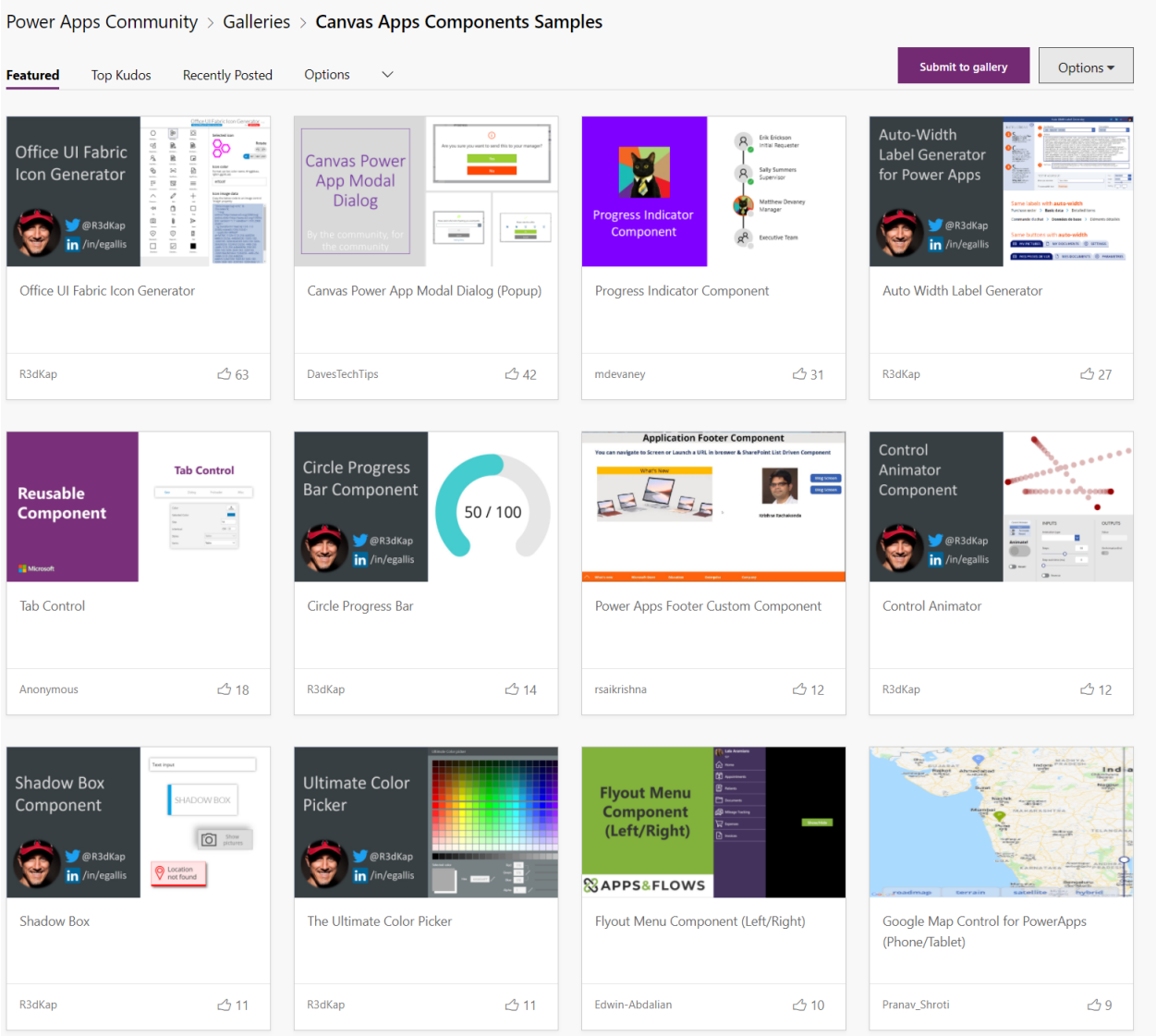
> **Note**
>
> Individual custom pages present in a model-driven app should be opened individually for editing inside canvas app studio to get the latest updates from the shared component library.



You can now publish the custom page and model-driven app to get the latest changes reflected.

Additional canvas component resources

You can evaluate and use canvas component samples from Microsoft and other Power Apps community developers, which are hosted on the Canvas app components gallery.



See also

[Model-driven app custom page overview]

[Add a custom page to your model-driven app]

[Using PowerFx in custom page]

[Overview of Power Apps connectors]

[Add data connection in canvas designer]

Add code components to a custom page for your model-driven app

Article • 06/03/2022

This article outlines the use of code components built by professional developers using Power Apps component framework within a custom page. For low-code custom UX extensibility, see add canvas components to a custom page for your model-driven app.

Important

- Custom pages are a new feature with significant product changes and currently have a number of known limitations outlined in **Custom Page Known Issues**.

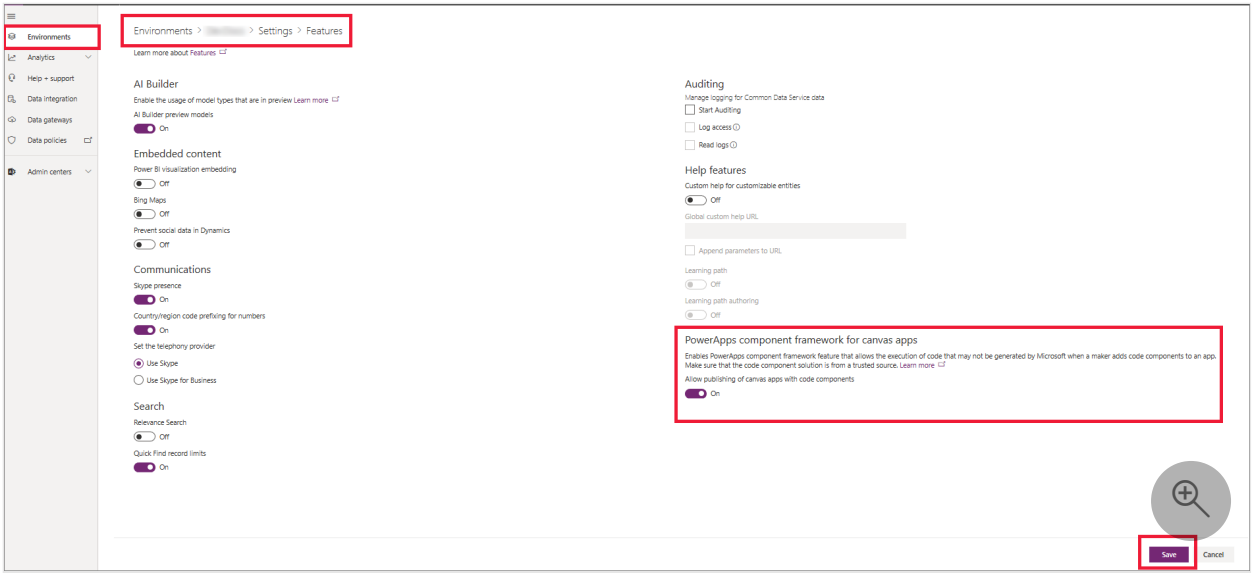
Note

- Custom pages support all the component framework APIs that are currently supported for canvas apps.
- Certain model-driven app specific APIs like **WebAPI** and **Navigation** have published app support for custom pages. These APIs can be used in the code component for custom pages and works on the final published app.
- For individual API support status on each platform, see **Power Apps component framework API reference**

Code components provide professional developers the ability to create custom code components for use within an app. This pro-code extensibility mechanism provides first-class application lifecycle management (ALM) to seamlessly extend the components available to all the app makers across the organization. Code components can be reused across custom pages, canvas, and model-driven apps. They can be centrally updated, packaged, and moved using standard Microsoft Dataverse solutions. More information: Power Apps component framework overview

Enabling Power Apps component framework for custom pages

To use code components inside a custom page, you need to enable Power Apps component framework feature. More information: Enable the Power Apps component framework feature



Create code components for use in a custom page

This section outlines how to create, import, and test code components.

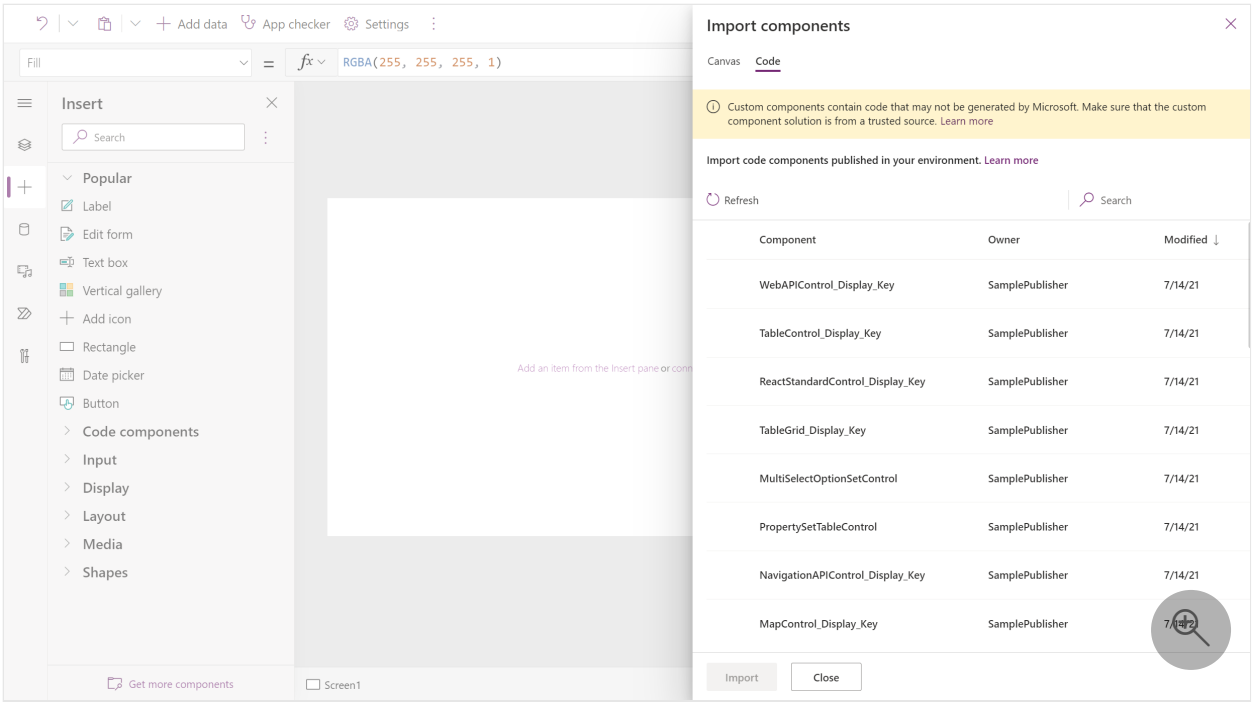
Creating and importing code components in Dataverse

Code components for custom pages follow the same pattern as they do with canvas app. Code components need to be implemented first before they can be added to a custom page. To create a code component, see Create your first component.

You can also try OOB sample components to jump-start. Once you are done implementing your code component, it can be packaged into a solution and added to Dataverse, making it available for use in all the custom pages in line with model-driven and canvas apps. More information: Code components application lifecycle management (ALM).

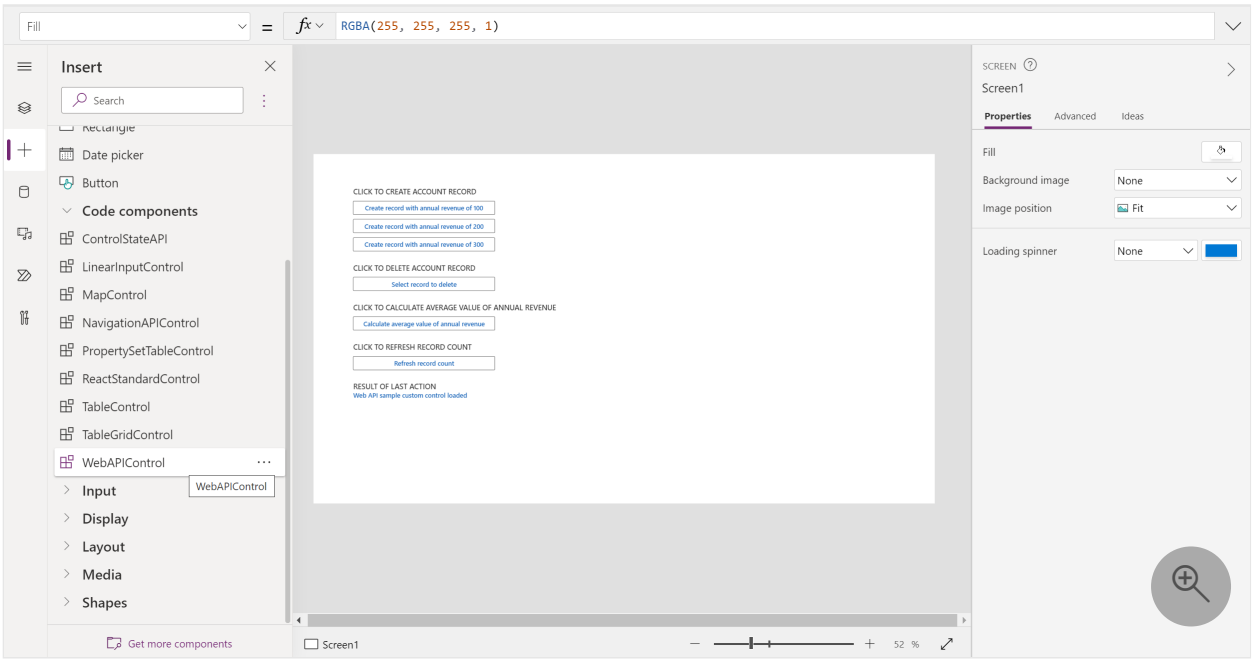
Importing and using code component in a custom page

In an environment, custom pages can use all the code components previously imported into Dataverse using solutions. On the left pane, select **Add (+)**, and then select **Get more components** at the bottom of the page. You will see a code tab on the **Import components** pane showing all code components present in the environment.



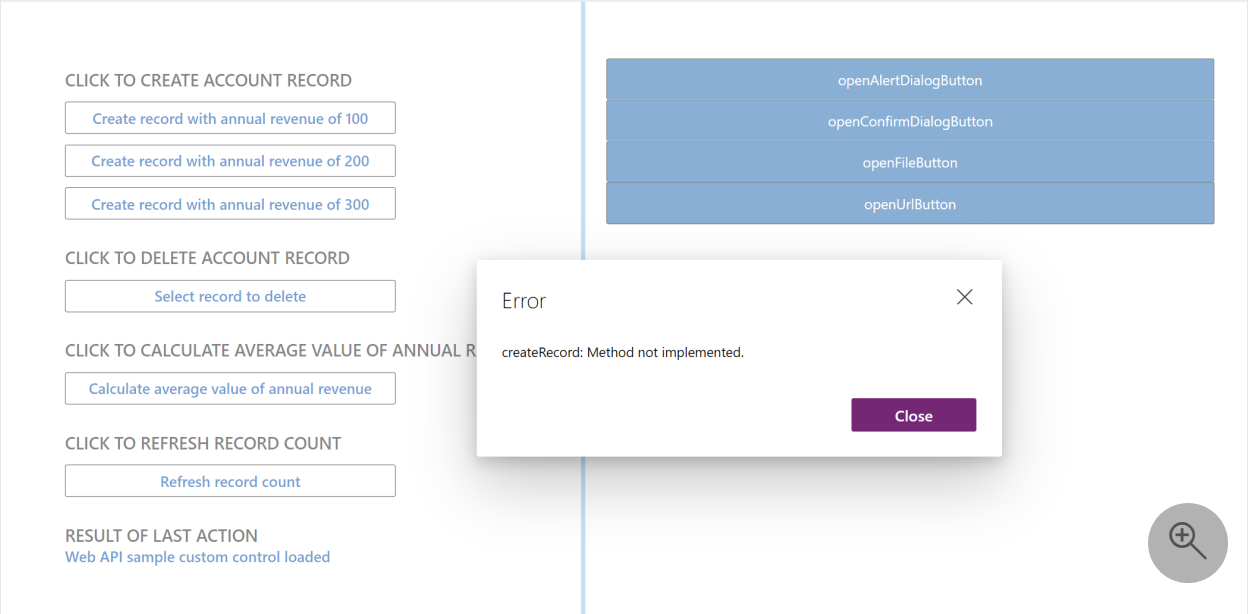
> **Note**
>
> - If the code tab doesn't show up in the **Import component** pane, verify that the Power Apps component framework feature setting for canvas apps is enabled. More information: **Enable Power Apps component framework feature**
> - Also, ensure that you are working on the latest canvas app studio authoring version.

The newly added code components are now available under the **Code components** section. It can be added to a custom page.



Testing the code component inside the studio and published app

Like canvas apps, code components are interactive and can be tested in the custom page studio authoring environment. However, specific APIs like `Web APIs` and `Navigation`, which only have the custom page runtime support when invoked, will show the error message "Method not implemented".

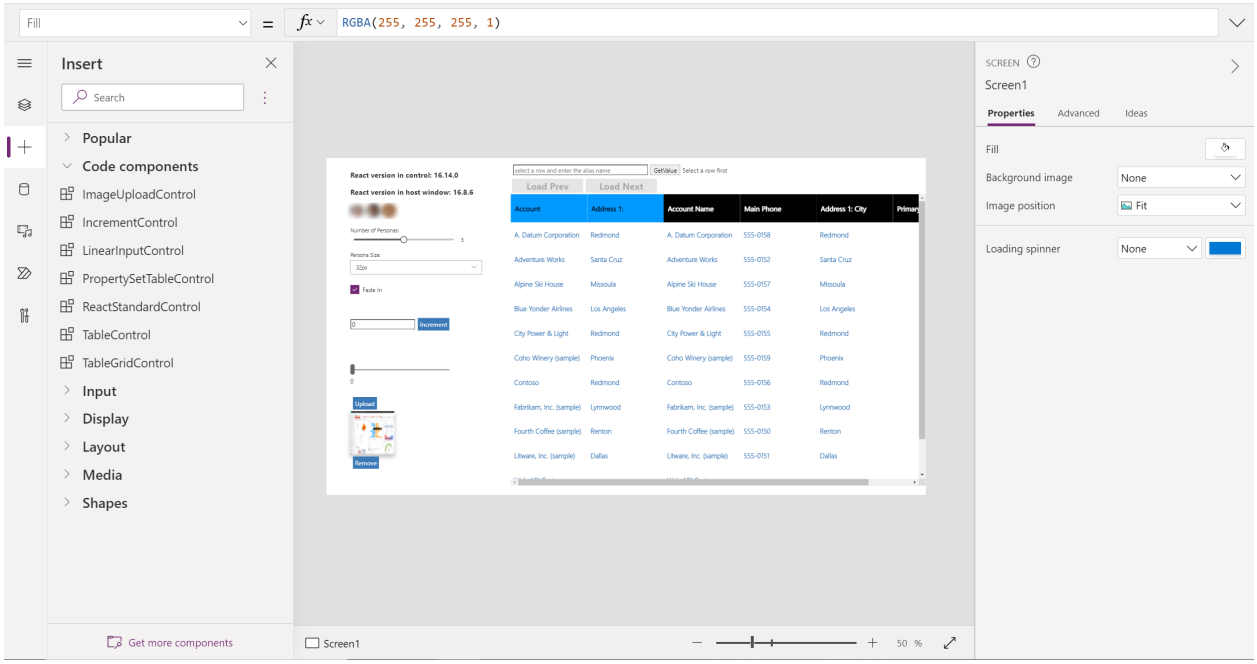


You can dismiss this error and publish the custom page. Then, add this custom page to the model-driven app and publish the app to see the custom page web API in action.

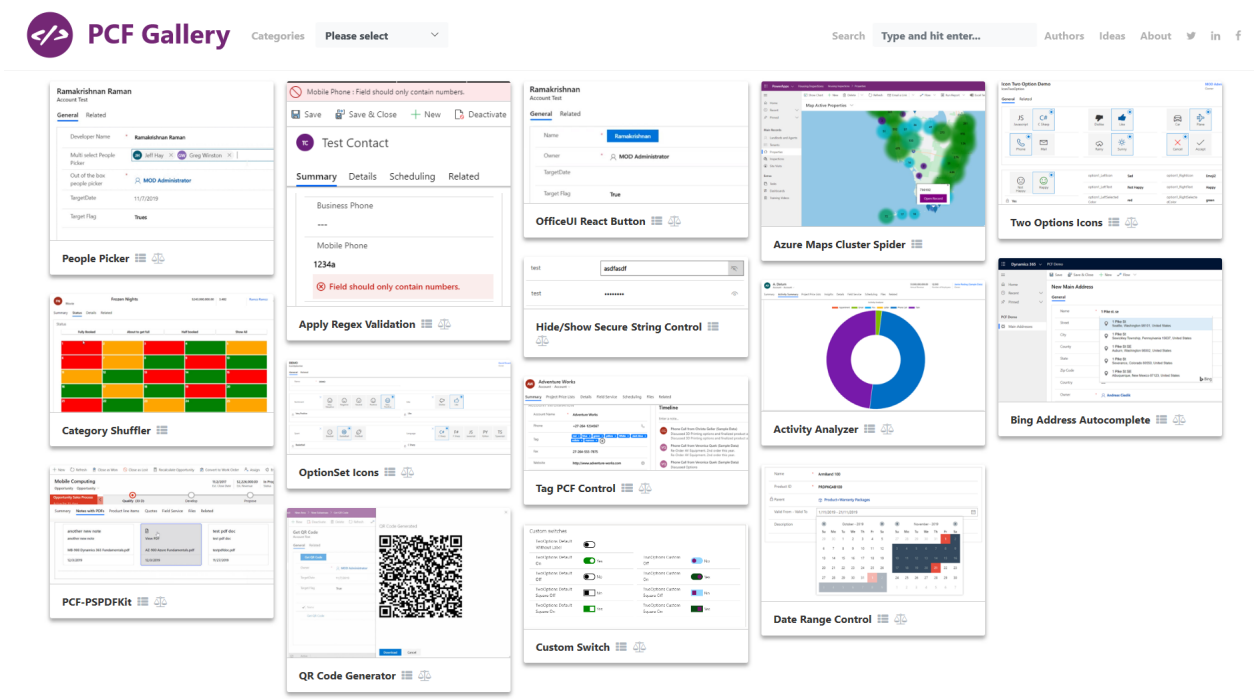
The image below shows the standard Web API sample control and Navigation API control added to custom page working inside a published model-driven app.

Additional code component resources

You can also use other sample components from Microsoft.



Or try some from the Power Apps community gallery.



See also

Model-driven app custom page overview

Add a custom page to your model-driven app

Using PowerFx in custom page

Overview of Power Apps connectors

Add data connection in canvas designer

Known issues with custom pages in a model-driven app

Article • 04/16/2025

The custom page is a new page type within model-driven apps. Custom pages bring the power of canvas apps into model-driven apps. Below are the known issues to be aware of.

> **Important**
>
> Using custom pages with mobile devices is currently in public preview. Offline and device capability controls like barcode scanning, capturing photos from device, or attaching files isn't supported.
>
> Custom pages aren't supported within **App for Outlook** and return an error on loading.

Maker-related issues

- When a custom page is modified, such as saved and published, the model-driven app isn't aware of the change. The model-driven app continues to use the last version of the custom page when the model-driven app was published. A model-driven app published through app designer, solution explorer, or **Publish all** updates all custom pages in the model-driven app.
- Images, icons, and shapes aren't currently supported with Right-To-Left (RTL) languages.
- The App.ConfirmExit property isn't supported in custom pages.
- The ability to get the current data formats from users settings including date, time, numbers, and currency isn't supported.
- Custom pages use a canvas app hosting session that can time out after 8 hours. However, the Unified Interface session has a longer timeout. When the timeout happens, an error message bar appears that prompts the user to refresh the browser.



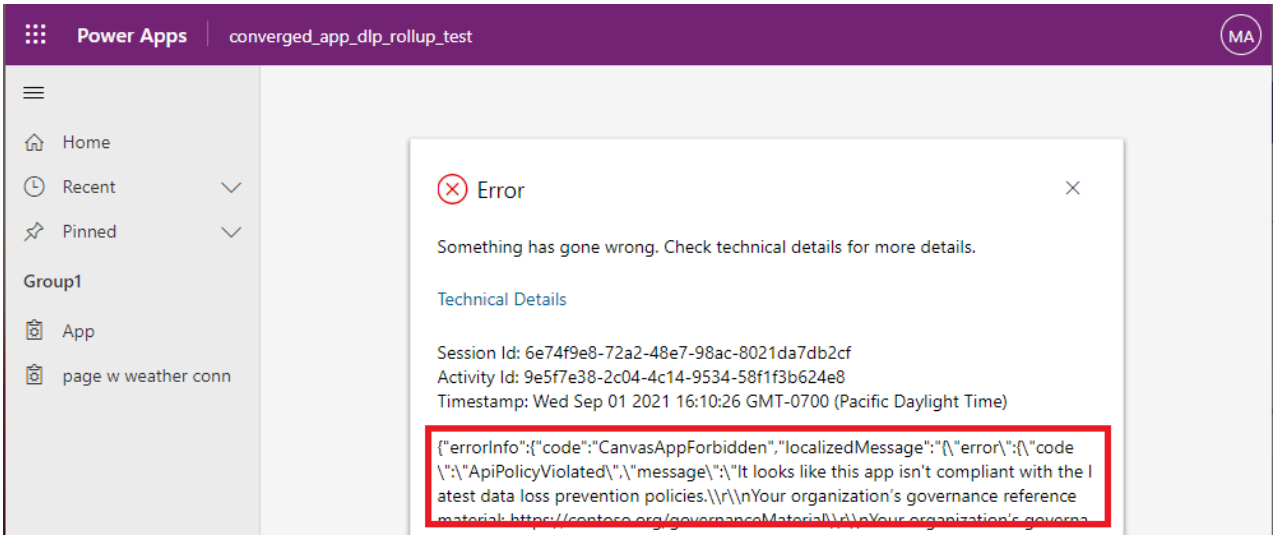
- When a custom page with code component is opened for editing, a security dialog is shown. Selecting **Go back** on the security dialog doesn't navigate back to the parent context. The user can close the browser tab to leave Power Apps Studio.

- Not all canvas app controls are available with custom pages. However, custom pages support the most common canvas app controls and custom pro-dev components. For more information about what is available, see Design a custom page for your model-driven app
- Makers need to share custom pages to allow another maker to make changes, which is a different behavior than the typical model-driven app components. If a custom page can't be shared from the **Solutions** area, open the environment in Power Platform admin center, and then open **Resources** > **Power Apps** > **Page** > **Share**. Similarly, to reuse the canvas app components inside the custom page, the corresponding canvas app component library also needs to be shared with the custom page makers.
- The maker experience for the custom page doesn't have support for certain Power Apps component framework APIs like `Navigation` and Web APIs, which is in line with the stand-alone canvas apps. However, these APIs are available in the published app where the custom page is added to model-driven apps. More information: Add code components to a custom page for your model-driven app.
- The maker experience for the custom page is currently not enabled in sovereign clouds. If you would like to enable it for a maker session, append "powerappsPortalApps.enableEditInShellAppDesigner=true" as a query parameter to the https://make.powerapps.com/ url.
- Makers can't use cross-environment Dataverse references in a custom page.

User-related issues

- When a user with no Power Apps user privileges opens a custom page in the model-driven app, they see an error mentioning no active entitlements to use Power Apps. More information: Licensing overview for Microsoft Power Platform and the associated licensing guide.
- Custom pages require third-party cookies to be enabled, which is required by the canvas app runtime.
- When a user is prompted for consent with connectors and selects **Don't allow**, the custom page renders but without data. The user doesn't get notified that data retrieval is skipped.
- After a model-driven app or custom page is changed and published, loading a custom page can take longer than normal, and no page loading spinner is shown.

- Native player support is available for iOS, Android, and Windows in online-only mode. Offline support currently isn't supported.
- When you go back to a custom page from another page, the page state isn't restored so the page appears like a new navigation. State is also not persisted when switching between multi-session tabs in multi-session apps. For more information about multi-session support with model-driven apps, go to Customer Service workspace sessions and tabs.
- While attempting to sign in, the current behavior caused by a user selecting anywhere away from the sign in box causes the pop out window for sign in to shift behind the app browser.
- When a user runs an app that isn't compliant with their organization's Data Loss Prevention (DLP) policies, they see an error dialog and the 'Technical details' reflects the app isn't DLP compliant.



- When there are multiple custom pages in an app, the consent dialog asks for data permissions for all of the connectors in all the custom pages even if they haven't yet been opened.

Connections

- All custom pages share the connection infrastructure, as if they were part of the same canvas app. For instance, if a model-driven app has two custom pages and one is connected to SQL Server while the other isn't, the consent dialog will still appear and require permission for SQL access even if the page being opened doesn't require it.
- Custom pages in model-driven apps don't support connection detection through the *Connected.connected* property.

- The number of connectors in a model-driven app, across all custom pages, shouldn't exceed 10. The number of connection references, across all custom pages, shouldn't exceed 20. The number of connectors and connection references can increase the wait time an app user experiences when the user plays an app with custom pages.
- The **App Permissions** dialog is not opened when a custom page is first opened by a user in the Power Apps mobile app. This behavior can cause the page to appear frozen, but is expected behavior. To resolve this issue, open the app in a browser to approve the permissions before playing it in the Power Apps mobile app. You can also run the AdminPowerAppApisToBypassConsent PowerShell command to bypass the need for the permissions dialog by a Power Platform administrator.

See also

Model-driven app custom page overview

Use live monitor to troubleshoot page behavior in model-driven apps

07/19/2025

Live monitor can help you debug and diagnose problems, which help you build faster, more reliable model-driven apps. Live monitor provides a deep view into how an app runs by providing a log of all activities in your app as it runs.

> **Important**
>
> Live monitor must be run on the model-driven app and not directly on a custom page.

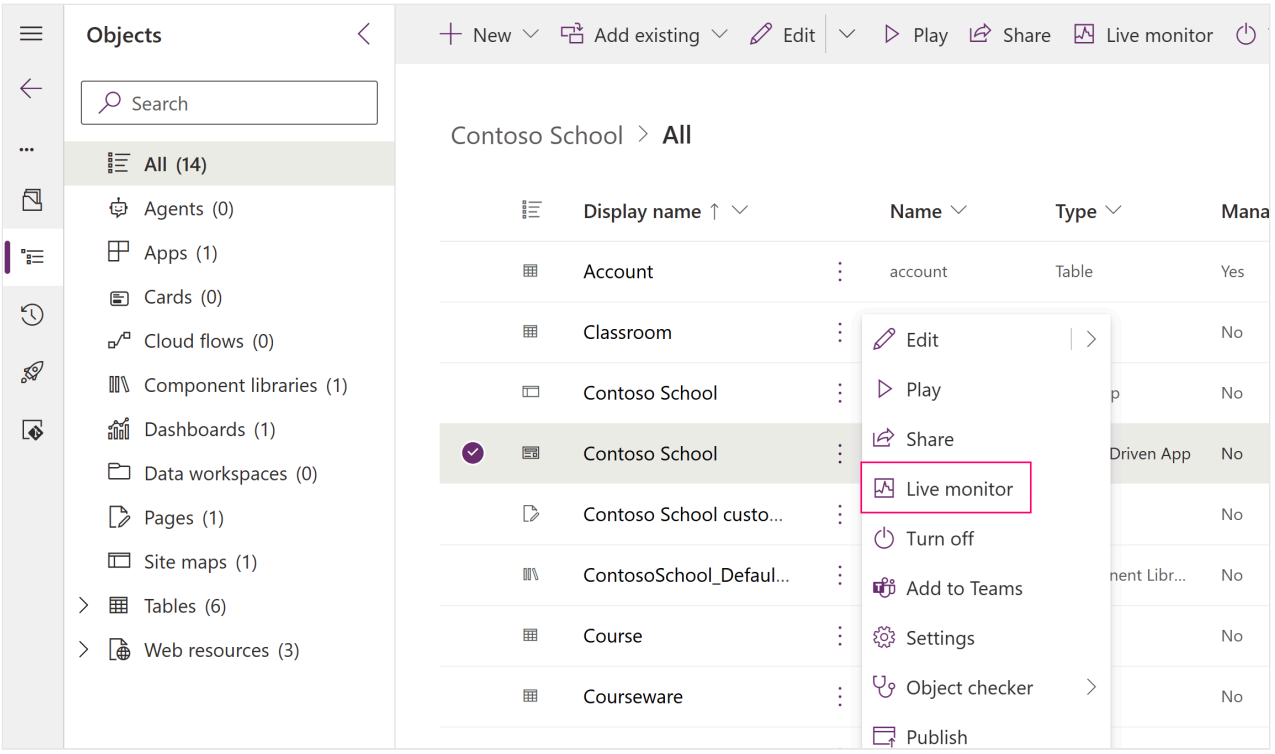
Start a live monitor session

There are two ways to open a live monitor session.

- From Power Apps
- From a model-driven app

From Power Apps

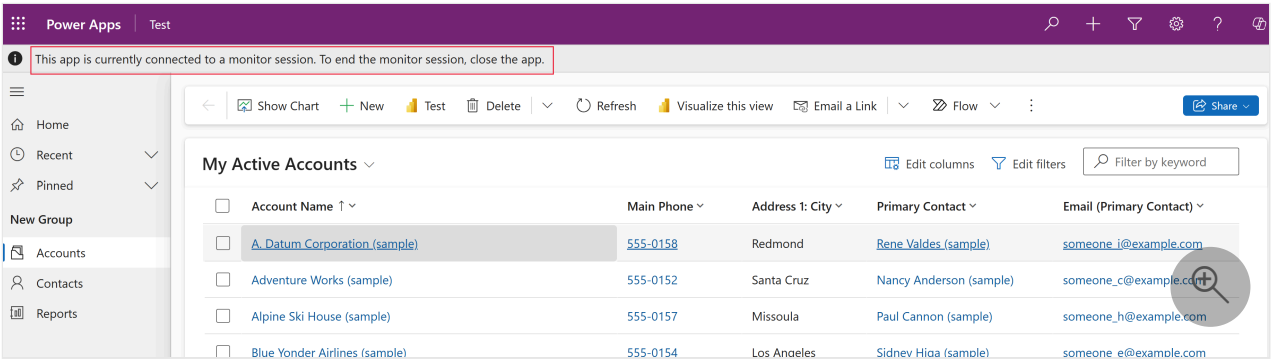
1. Sign in to Power Apps, select **Apps**.
2. Select **...** next to the model-driven app or on the global command bar, and then select **Live monitor**.



3. On the **Live monitor** browser tab that opens, select **Play model-driven app** on the command bar.

4. When prompted, select **Join**.

5. The app loads in a new browser tab with a message indicating the monitoring session. Switch back to the live monitor tab to view activity.

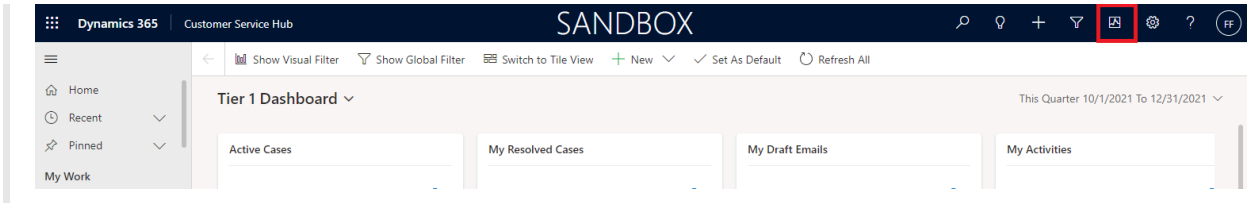


From a model-driven app

1. Play your model-driven app.

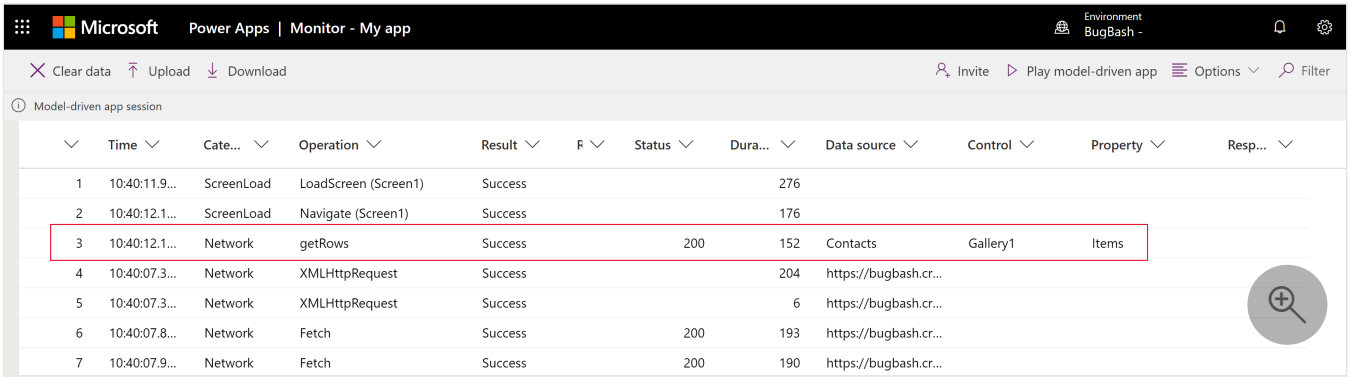
2. Add `&monitor=true` to the end of the URL in your web browser, and then refresh the page.

3. Select **Live monitor** on the command bar.



Filter monitored activity

When you filter on a model-driven app custom page-related events in live monitor, you can get information about related tables, tables, controls, and components, such as on a custom page, in live monitor as your app runs.



	Time	Cate...	Operation	Result	R	Status	Dura...	Data source	Control	Property	Resp...
1	10:40:11.9...	ScreenLoad	LoadScreen (Screen1)	Success			276				
2	10:40:12.1...	ScreenLoad	Navigate (Screen1)	Success			176				
3	10:40:12.1...	Network	getRows	Success		200	152	Contacts	Gallery1	Items	
4	10:40:07.3...	Network	XMLHttpRequest	Success			204	https://bugbash.cr...			
5	10:40:07.3...	Network	XMLHttpRequest	Success			6	https://bugbash.cr...			
6	10:40:07.8...	Network	Fetch	Success		200	193	https://bugbash.cr...			
7	10:40:07.9...	Network	Fetch	Success		200	190	https://bugbash.cr...			

For example, to filter on the custom page that uses the `Gallery1` control, select the **Control** column header > **Filter by**, enter *Gallery1*, and then select **Apply**.

For more information about model-driven app monitoring, go to Debug a model-driven app with live monitor

Close a live monitor session

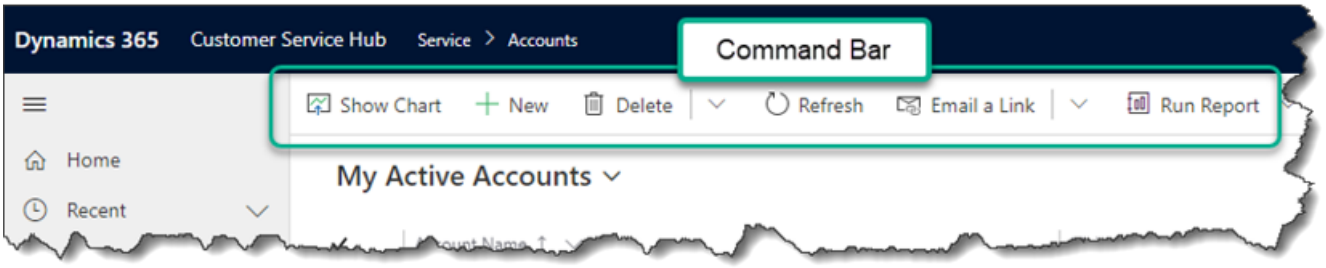
To close the monitoring session, close the browser tab where the monitored model-driven app is playing.

Next steps

Use live monitor to troubleshoot model-driven app form behavior

Modern commanding overview

Commands drive core application behavior for model-driven apps. They're the buttons users interact with when playing apps and the resulting actions performed when a button is selected. Each command is positioned in relation to other commands and bound to a command bar location within the app.

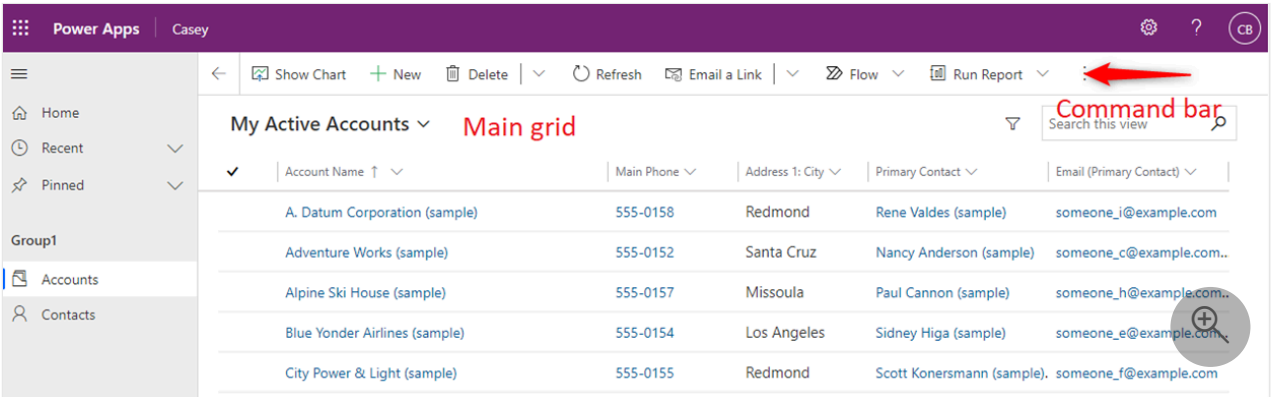


At a high level, command customization fits into three categories. Various capabilities exist within each category and are covered in more depth throughout the modern commanding documentation:

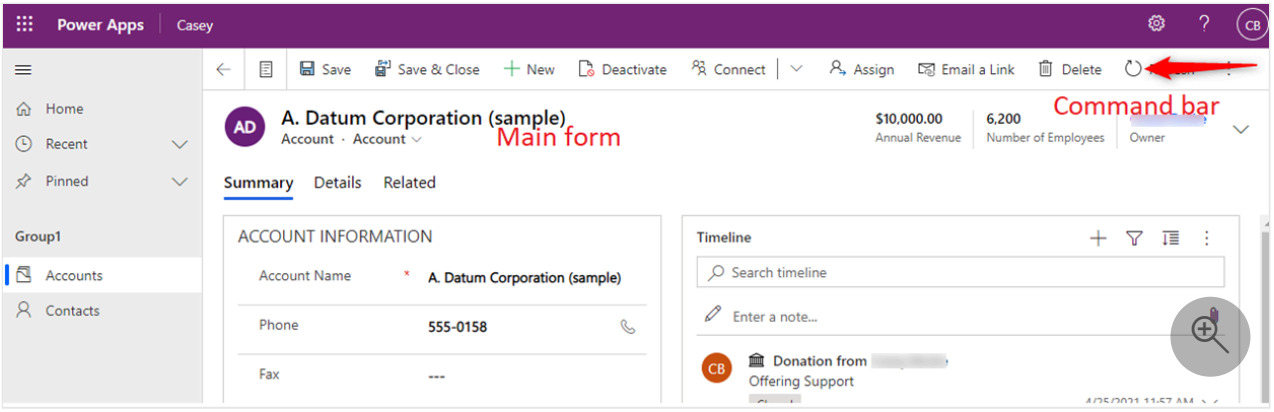
- **Display**. How the button appears and where it's located in an app. For example, the button's label, icon, and accessibility labels as well as the command bar location and position within a command bar.
- **Action**. The logic that runs when a user selects a button. For example, creating and updating data or interacting with various controls and pages inside the app.
- **Visibility**. Logical conditions that specify when a button is visible or hidden to a user. For example, you might want the button visible for some users and hidden for others. Or perhaps the button should only be visible when certain criteria of the data records are satisfied.

Command bar locations

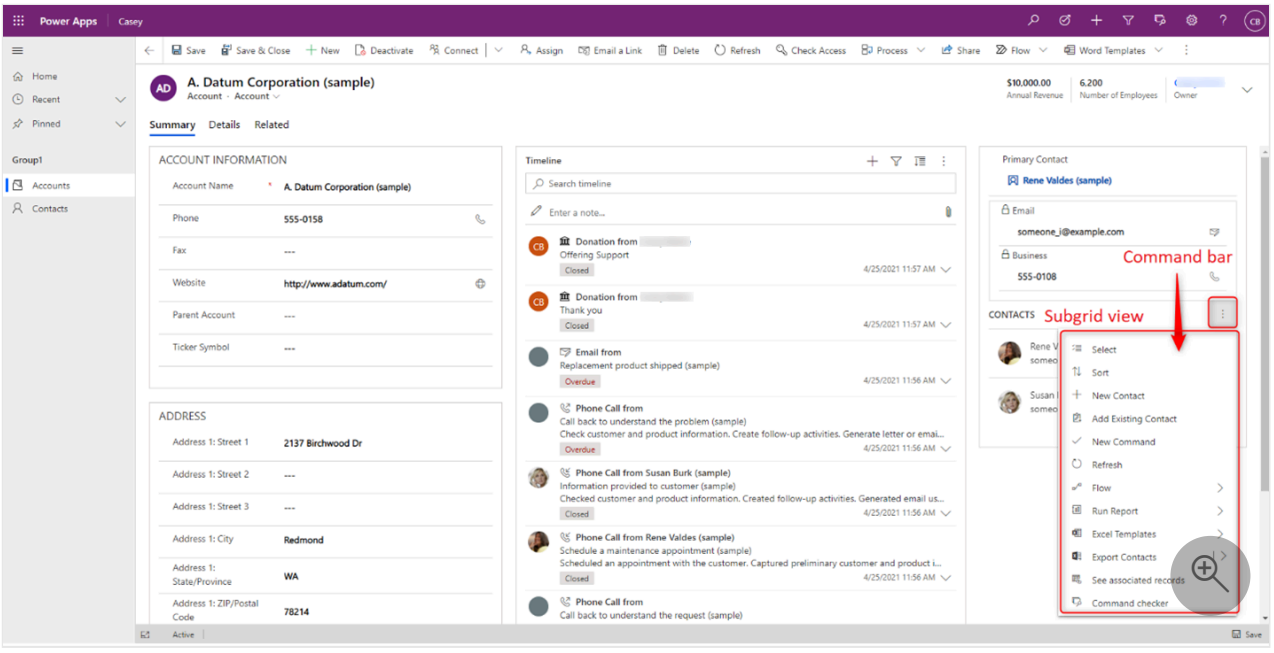
- **Main grid**. This command bar is displayed when using the left-hand navigation of an app to view a full page list of records in this table.



- **Main form**. This command bar is displayed on the table's main forms. This appears at the top of the form and isn't the same as the associated view or subgrid view that appears in different areas of the form.

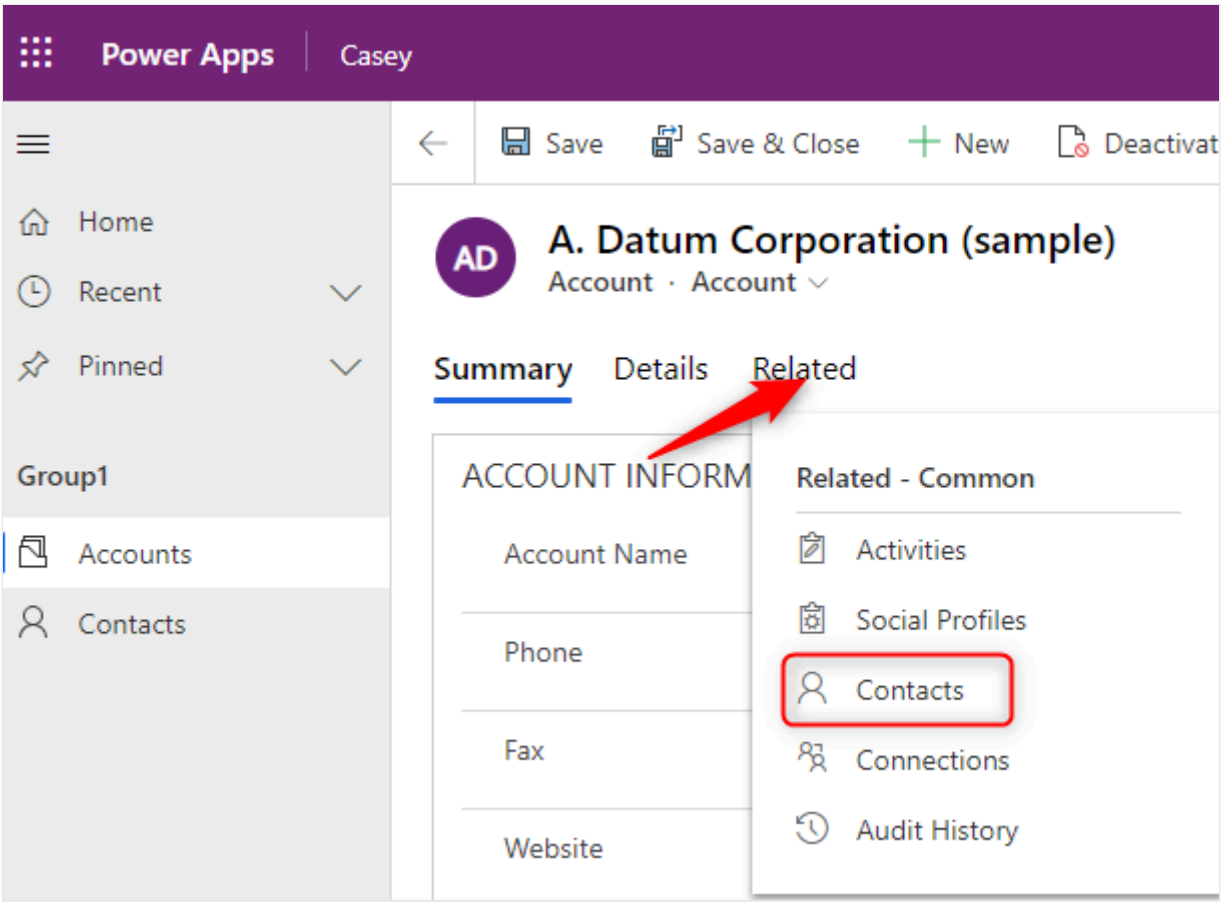


- **Subgrid view**. This command bar is displayed on forms of other tables that render this table's data within a subgrid. For example, the account main form has a subgrid control that lists contact records related to the account record. To edit the command bar shown here, edit the command bar for the contact table then subgrid view.

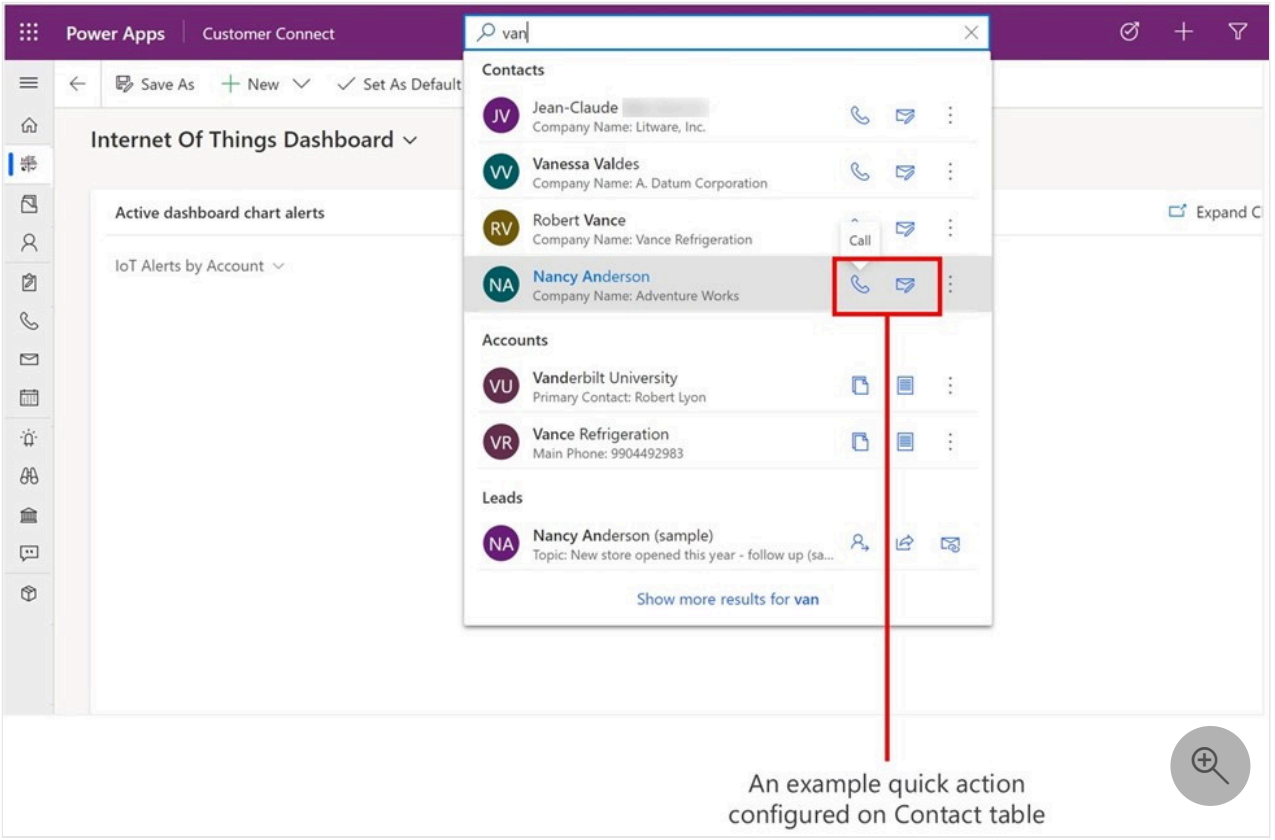


- **Associated view**. This command bar is displayed on the form of a parent table when viewing related data in this table. For example, from the main form of an account record

select the **Related** tab then select a related table like contacts.



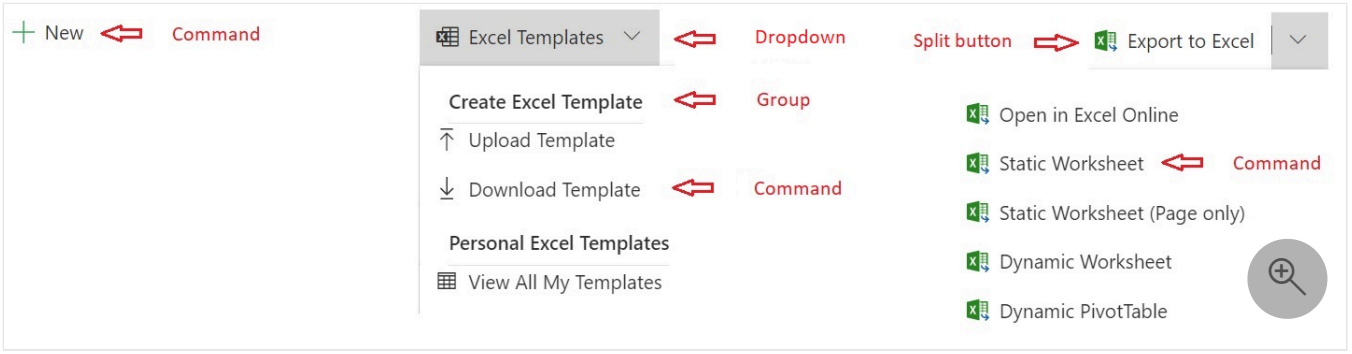
- **Quick actions**. Quick actions are associated with the main grid location. To add or edit commands for both quick actions and main grid locations, select the desired table from within the modern app designer, then edit the command bar and choose the main grid location. The first five commands, determined by order, are also shown as quick actions when playing the app.



> **Note**
>
> Less frequently customized command bar locations aren't supported in command designer. See the Global command bar and other ribbons sections for more information on customizing commands for these locations.

Types of commands

- **Command**. Standard button. Performs an action when selected. Can also be nested in groups within dropdowns and split buttons. These were called flyouts in classic commanding.
- **Dropdown**. Creates a menu where you can organize commands within a group.
- **Group**. Add titles to groups of commands nested within dropdowns and split buttons.
- **Split button**. Similar to a dropdown, but has a primary command. When the split button is selected, the action from the primary command is executed. If the expand chevron is selected, the primary command won't be executed. Instead a list expands to show additional groups, flyouts, and commands.



Key differences between classic and modern commands

Classic commands (formerly known as the ribbon) weren't customizable using low code. With code, command customizations were difficult, tedious, and error prone. In order to scale commanding to low code as well as use custom pages to converge canvas and model-driven apps, it was vital to reinvent and rebuild the command infrastructure.

Modern commanding offers many new capabilities and is much simpler to use.

Expand table

Capability	Classic	Modern
Supported in model-driven app runtime	Yes	Yes, Additionally supports Power Fx runtime.
Customized using	Hand editing XML within solution files or using non-Microsoft tools. Requires time-consuming solution export and import operations.	Command designer as well as Dataverse API support.
Supports Power Fx.	No	Yes. For actions and visibility.
Time required to customize	Slow, error prone.	Fast
Reliability and performance	Easy to make mistakes. Bad customization and lack of scoping often impact app performance.	Inline error handling prevents mistakes. Power Fx optimized for better runtime performance.
Sharing	Standard Dataverse role-based security.	Non-Power Fx commands use standard Dataverse role based security. Power Fx commands currently require the command component library to be shared in addition to having an appropriate security role.
Solution and application lifecycle management (ALM) behaviors	Inconsistent and problematic solution layering, no presence in solution interface. Many standard solution behaviors not supported such as patches, segmentation, solution upgrade, managed properties, and many more.	Standard solution layering centrally managed for multiple solution object types within Dataverse. Present in solution interface. All standard solution behaviors supported.
Localization	Nonstandard	Standardized using export and import translations for the entire solution.
Data model	Complex. Optimized for classic ribbons and contain many properties that are no longer needed.	Simple, optimized for today's model-driven app command bars.
Use JavaScript	Yes	Yes. Now simpler. The same JavaScript can be used for classic and modern commands.
Customize out of the box commands	Yes	Commands become editable in command designer once migrated to the modern framework.

Capability	Classic	Modern
App specific commands	No	Yes. Using modern command designer ensures commands are only visible within the selected app.
Table specific commands that display in all apps containing the table	Yes	Yes. Requires modifying the `appaction` definition within the solution file.
Global commands that display for all tables and apps for the specified command bar location	Yes	Yes. Requires modifying the `appaction` definition within the solution file.
Create split buttons, flyouts, and groups	Yes	Yes
Dynamically populate a flyout with code	Yes	No. Create commands declaratively.
Customize global application header commands	Yes	No
Customize commands for other, uncommon, or obsolete command bar locations	Yes	No.
Run a modern flow or workflow	Using JavaScript	Using JavaScript. Also supported using a custom page.

Classic versus modern visibility rule comparison

Classic visibility rules often had a specific rule for each scenario. With Power Fx, a declarative function replaces many classic rules. And it’s much simpler to use.

Classic visibility rules are also supported within modern commands. However, support for classic rules was needed for reliably migrating classic commands to modern commanding and classic rules customization within command designer isn't supported. We recommend you use Power Fx going forward.

Expand table

Use case	Classic Rule	Classic options	Power Fx visible property
Show or hide based on data values	CustomRule	Use JavaScript	!IsBlank(Self.Selected.Item.Email)
Show or hide based on table permission	EntityPrivilegeRule	Multiple	DataSourceInfo()
Show or hide based on record permission	RecordPrivilegeRule	Multiple	RecordInfo()
Reference the control context for primary and related tables	EntityRule	PrimaryEntity. SelectedEntity	Self.Selected
Reference the control context	EntityRule	Form. HomePageGrid. SubGridStandard. SubGridAssociated	Self.Selected
Table metadata properties	EntityPropertyRule		DataSourceInfo()
Show or hide based on form state. For example, show for the create form	FormStateRule	Create. Existing. ReadOnly. Disabled. BulkEdit	Self.Selected.State = FormMode.New
Show when > 1 records are selected in a grid	SelectionCountRule		CountRows(Self.Selected.Items) > 1
Show or hide for a related table in a polymorphic lookup. For example, check whether the lookup is a user or a team	CustomRule	PrimaryEntityTypeCode	IsType(), AsType
Reference environment properties (Org)	CustomRule	OrgName. OrgLcid. UserLcid	Not currently available

Frequently asked questions

- *Why do I see more commands in the designer than I see in my app?*

    - There are several reasons. Sometimes there's visibility logic that hides the command when running the app. Other times these commands are dynamically injected via custom JavaScript during runtime and aren't configurable.
- *Why do I see duplicate commands in the designer?*
    - This was a common pattern used with classic commands. Both commands wouldn't show up in runtime as they were controlled by visibility rules. The command designer shows all commands, regardless of their visibility rules.

See also

Customize the command bar using command designer
Manage commands in solutions
Modern commanding known limitations

Last updated on 03/31/2026

Customize the command bar using command designer

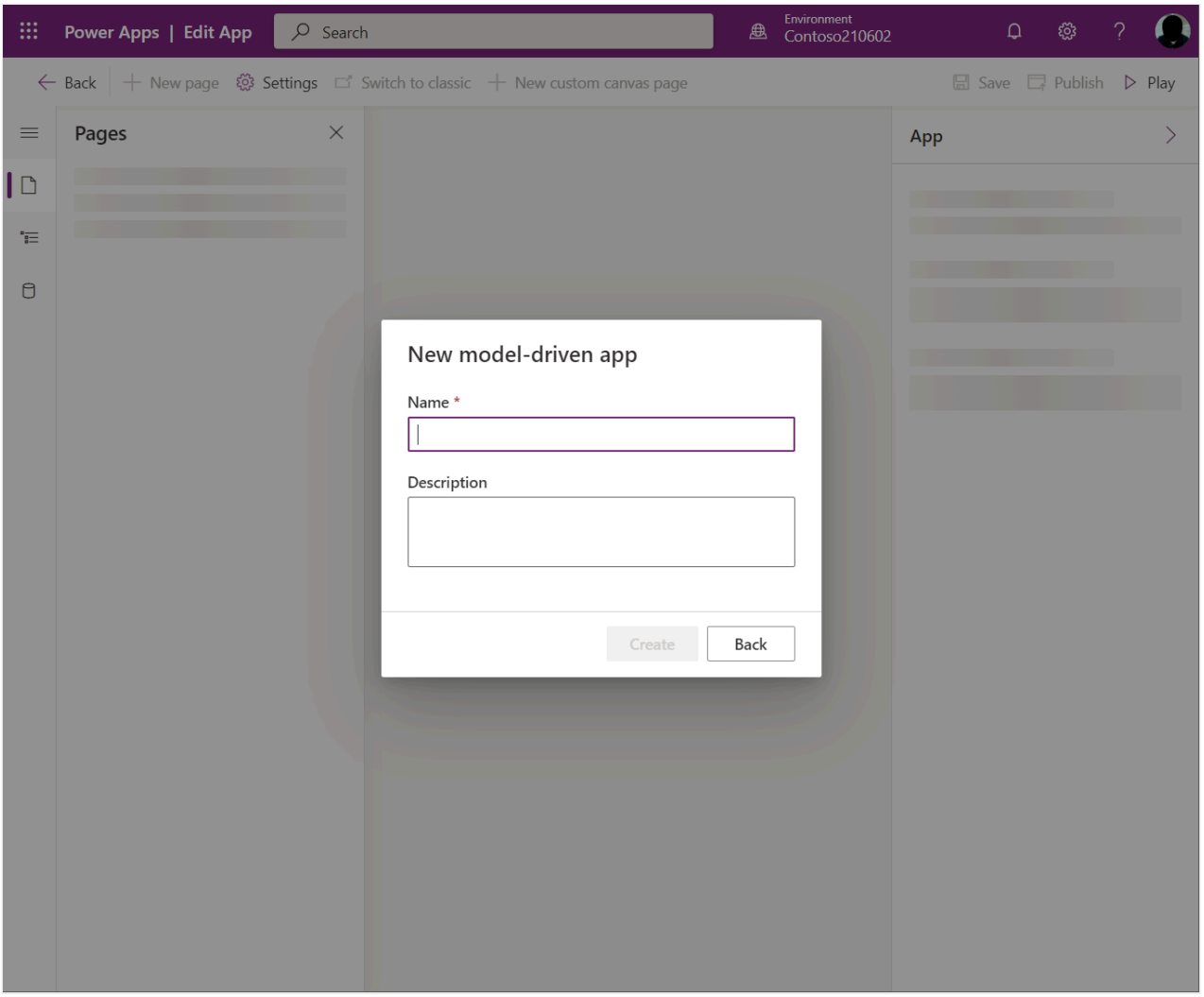
This article guides you through creating and editing modern commands using the command designer and Power Fx.

Open the app designer

Either Create a new model-driven app using modern app designer or Open an existing model-driven app using app designer.

Create a new model-driven app using modern app designer

1. Sign into Power Apps .
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open or create a solution to contain the new model-driven app.
4. Select **New** > **App** > **Model-driven app**.
5. Enter a **Name** for your app, and then select **Create**. More information: Build your first model-driven app



Open an existing model-driven app using app designer

1. Sign into Power Apps
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution containing the existing model-driven app.
4. Select the model-driven app, and then select **...** > **Edit** to open the modern app designer.

Create or edit modern commands

Once you are in the app designer, use the command designer to customize your command bars.

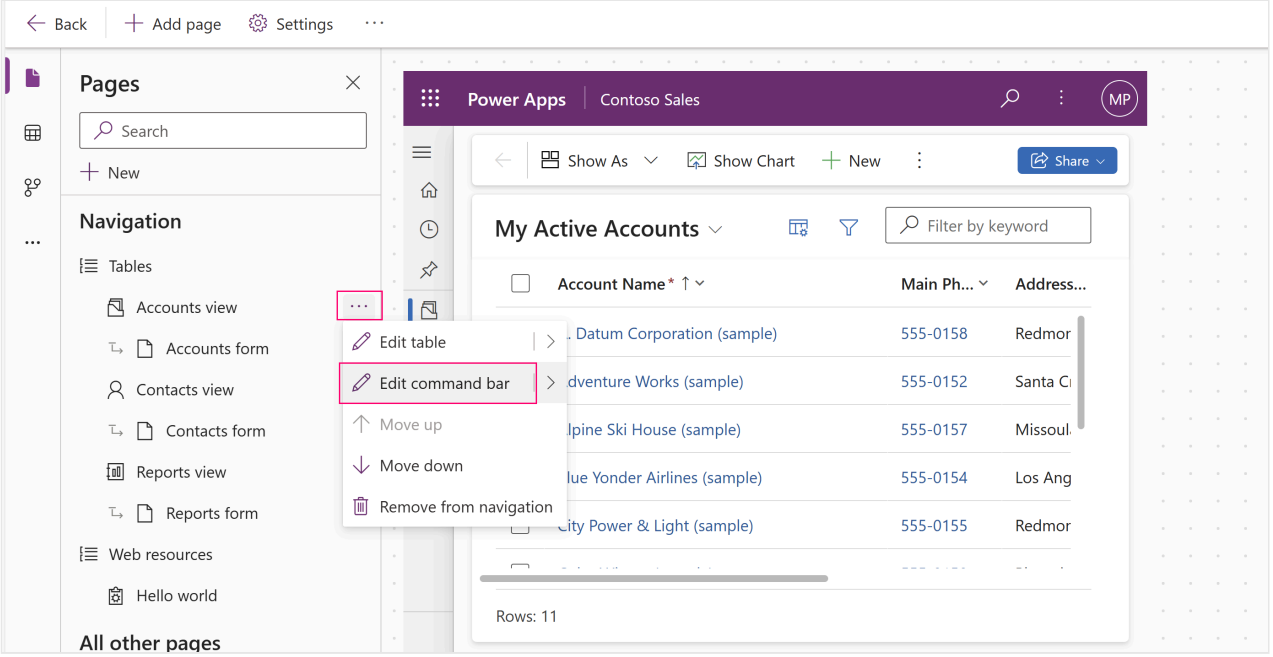
> **Note**
>
> - Currently, the command designer can only be accessed through the modern app designer or within solutions for existing commands.

- Currently, classic commands can't be edited within the command designer.

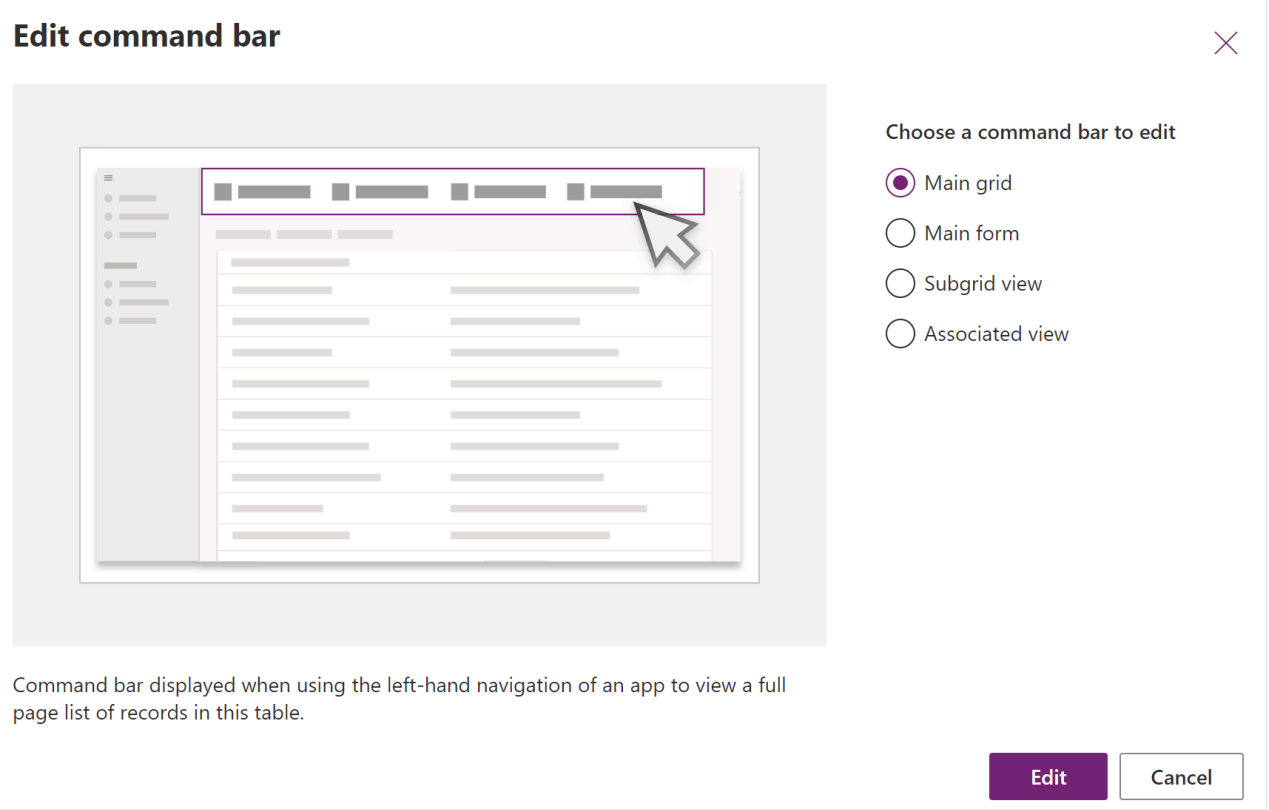
Edit the command bar

Open the command designer to edit a command bar

1. Next to any table view from the **Pages** area in the app designer, select **...**, and then select **Edit command bar**.



2. Select the location of the command bar you want, such as main grid (view) or main form, and then select **Edit**. More information: Command bar locations

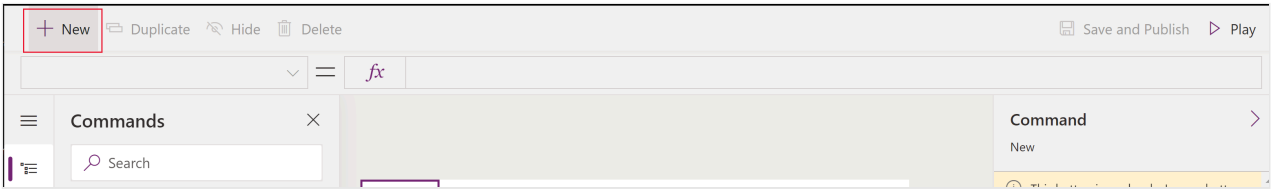


Create a new command

Unlike classic commands, modern commands are only displayed within the app you're editing. This prevents unwanted command transferring to other apps as well as better runtime performance.

The first time command designer is opened for an app, a prompt to either use **JavaScript** only or **Power Fx** is displayed. Selecting Power Fx also allows you to use JavaScript if desired. Selecting **Power Fx** in the dialog creates a command component library to store **Power Fx** formulas.

1. Open the command designer to edit a command bar, and then select **New** > **Command**.

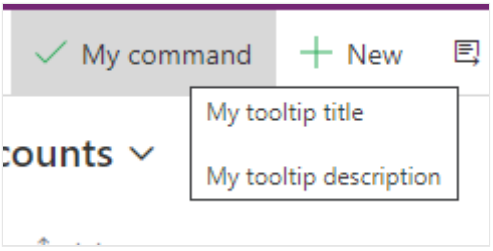


2. On the right pane, enter or select from the following options:

   - **Label (optional)**. Enter a label that displays on the command button.
   - **Icon**. Select an icon for the command button. You can choose from any system icons or web resource SVG files. To upload your own icon, choose **Web resource** then upload an **SVG** format file. Then, select **Save** and **Publish** the web resource. For

more information about how to create a web resource for the icon image you want, go to Create or edit model-driven app web resources to extend an app.

- **Action**. Select from the following:
  - **Run formula**. Enter the Power Fx formula to run the command action. More information: Use Power Fx for actions and visibility
  - **JavaScript**. Provide the JavaScript library and command to run the command action. More information: Use JavaScript for actions
- **Visibility**. Select whether to **Show** the command button or to **Show on condition from formula**.
- **Tooltip title**. Optionally, enter a tooltip title. The title appears to the user when they hover their mouse over the command.
- **Tooltip description**. Optionally, enter a tooltip description. The description appears for the user below the tooltip title when they hover their mouse over the command.



- **Accessibility text**. Optionally, enter the text that will be read by screen readers.
- **Order number**. The order the command displays in at runtime in relation to other commands within the same command bar.

3. Drag and drop the command to the desired location. You can arrange modern commands among classic commands.
4. Select **Save and Publish** to make the command available to app users.
5. Select **Play** to run the app and test your changes.

> **Note**
>
> Publishing Power Fx formulas can take a few minutes to publish the command component library. This background operation might still be in-progress after command designer has completed the request and unlocks the designer.

Use Power Fx for actions and visibility

You can use Power Fx for both actions (what happens when the command button is selected) as well as visibility (logic to control when the button is visible). Power Fx isn't supported in classic commands.

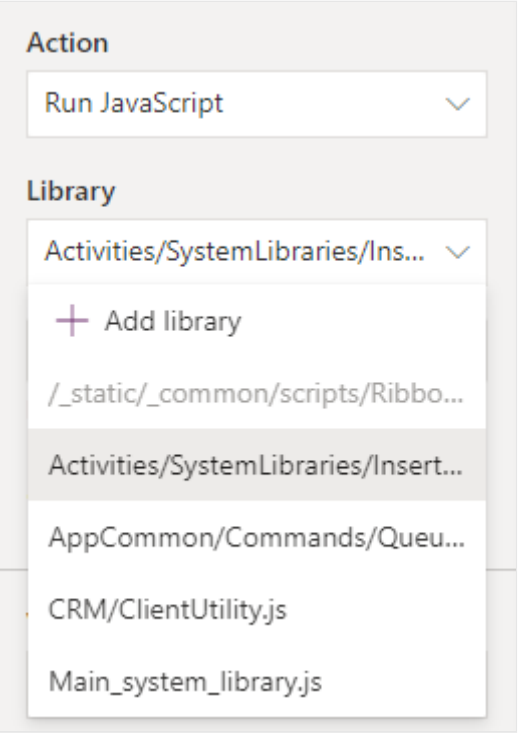
Notice the model-driven app command has a formula bar experience that is similar to canvas apps. For working with **Dataverse** data you can use Power Fx formulas just as you would in canvas apps. More information: Using Power Fx with commands

> ⓘ **Note**
>
> - Dataverse is currently the only data source supported with commands in model-driven apps.
> - You can't currently add additional tables as data sources directly from the command designer. However, you can open the command component library in canvas studio and add additional tables as data sources and then use them within the command designer.
> - Not all functions available within canvas apps are supported currently for model-driven app commands. Additionally, we've introduced some new functions specific to model-driven app commands.
> - For more limitations with modern commands, go to **Modern commanding known limitations**.

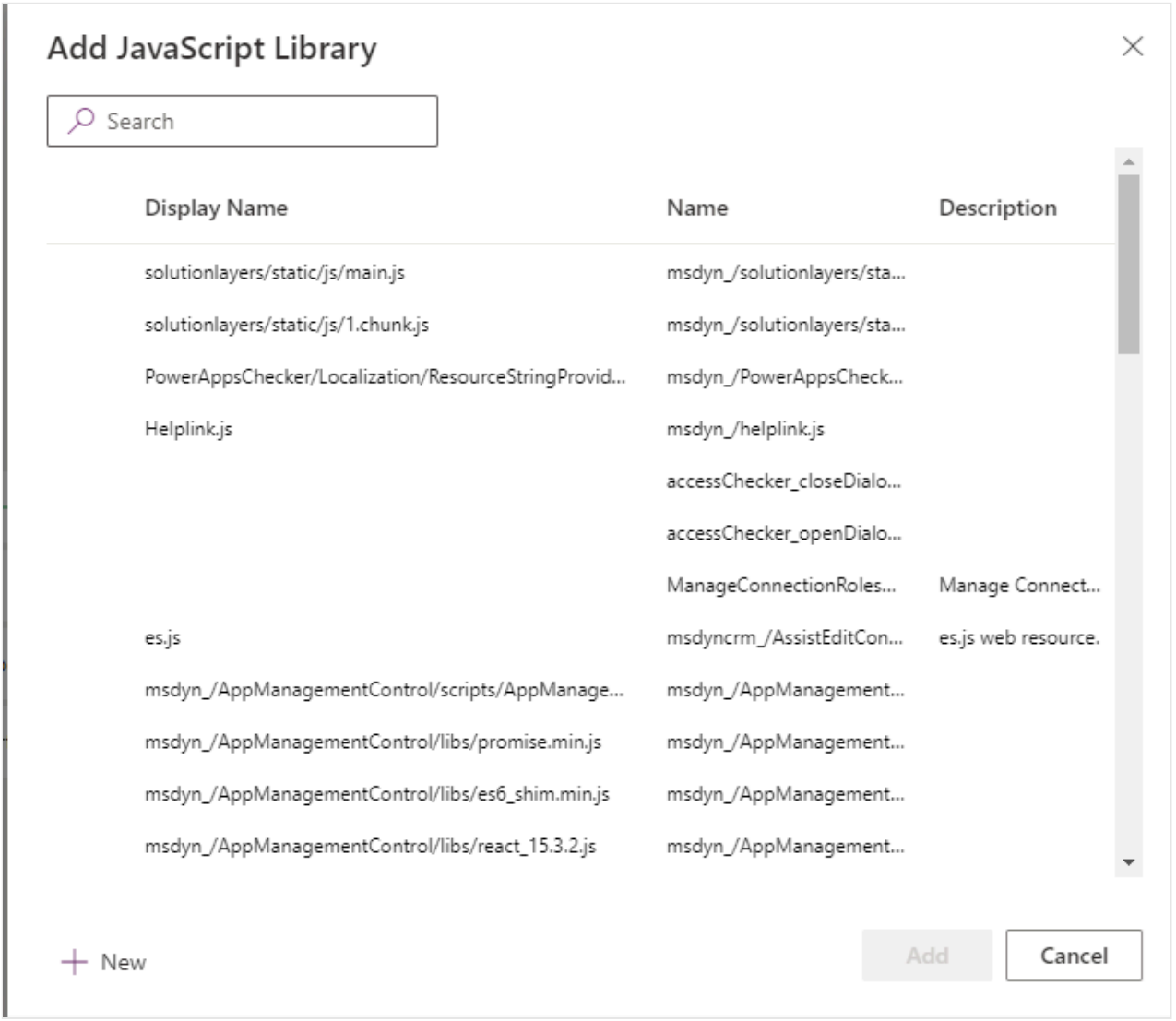
Use JavaScript for actions

JavaScript is supported with both classic and modern commands. However, it's simpler to create commands and associate your JavaScript using the modern command designer.

1. For the **Action** select **Run JavaScript**.
2. Select **Add library** or select another one from the list. The list is populated with any libraries in use by the current command bar.

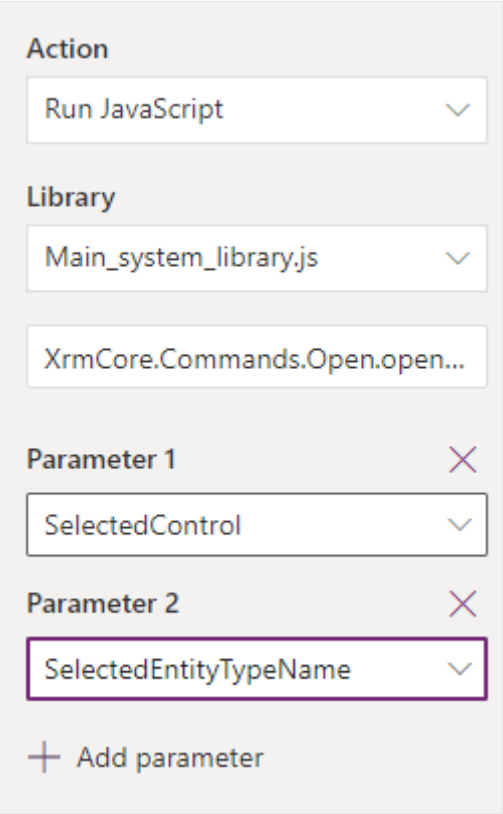


3. Select **Add** and search for existing JavaScript web resources or you can add your own.



4. Enter the **Function name**. For example, select the `Main_system_library.js` library then call this function: `XrmCore.Commands.Open.opennewrecord`.

5. Add parameters to pass to your function. For more information about the available parameters, go to Grid values.



> **Note**
>
> The use of calling multiple JavaScript libraries or calling multiple functions from a single command isn't supported.

See also

Modern commanding overview

Last updated on 01/09/2026

Using Power Fx with commands

Article • 04/23/2025

This article covers aspects of Power Fx that are specific to commanding. Many other functions that are in use today within canvas apps can also be used. Keep in mind there are differences because commanding is for model-driven apps.

- All existing dataflow functions are supported. What are dataflows?
- Imperative functions that work with data are supported.
- Imperative functions for simple `Confirm` and `Notify` are supported.
- For a list of functions not supported, go to Functions not supported.

> **Note**
>
> Publishing Power Fx commands might take a few minutes. It might not be obvious that background operations are still running even after the publish operation appears to complete. You might need to wait a few minutes after publishing, then refresh the app to see your changes reflected. This operation typically takes longer the first time a Power Fx based command is published for an app.

OnSelect

Defines the logic that is executed when the button is selected within the app.

Visible

Defines logic for hiding or showing the button when running the app.

To define visibility logic, select the command. Then select **Visibility** on the right command properties pane and choose **Show on condition from formula**. You can select **Visible** on the left of the formula bar then write a Power Fx expression using the formula bar.

Selected property

Expand table

Field	Type	Description
Item	Record of DataSource	One of the records selected from the DataSource.

Field	Type	Description
AllItems	Table of records from the DataSource	All of the records selected from the DataSource.
State	Enum	State of the selected control. Edit (=0), New (=1), View (=2)
Unsaved	Boolean	Returns true if Selected or SelectedItems have unsaved changes. Otherwise returns false. Always returns false if AutoSave is set to true (default option) within the command component library.

- The **Selected** property is provided by the host of the command.
- **Item** and **AllItems** names are somewhat consistent with the ComboBox control and Gallery control, but this is a new pattern.
- If there's no record selected, **Item** returns Blank (IsBlank returns true) and **AllItems** returns an empty table (IsEmpty returns true).
- Null DataSource for record references (polymorphic record types). Generic functions can be called, such as Save or IsType/AsType can be used.
- **Item** is always blank if **SelectionMax** <> 1. This prevents writing formulas to just one item and not scaling to more than one.

AutoSave

- Many JavaScript commands start by saving the form buffer. This is because it makes the rest of the code easier to work with.
- By default, the form buffer is saved on behalf of the app maker.
  - The form is saved before the command is initiated.
  - Any problems that occur during the save operation are dealt with in the form's UI.

Patch function

Patch (update) the current selected record

Power Fx

```
Patch(Accounts, Self.Selected.Item, {'Account Name': "Changed Account name"})
```

Create a related record

> **Note**
>
> If the related table is not already in the command component library, you need to open it in canvas studio and add the data source there.

Power Fx

```
Patch(Tasks,Defaults(Tasks),{Regarding:Self.Selected.Item},{Subject:"Subject of the Task"})
```

Check and edit a date property

Power Fx

```
If(Self.Selected.Item.'Last Date Included in Campaign'>DateAdd(Now(),-3),
Patch(Accounts,Self.Selected.Item,{'Last Date Included in Campaign':Date(2021,10,19)}))
```

Visible property: Only show the command if one or more records is selected in a grid view

Power Fx

```
CountRows(Self.Selected.AllItems) > 0
```

Control visibility based on record data

Power Fx

```
//Button will be visible for accounts with Account Rating > 20
Self.Selected.Item.'Account Rating'>20
```

Navigate

> **Note**
>
> For additional options, see the client API reference to use JavaScript. More information: navigateTo (Client API reference)

Navigate to a custom page

To navigate to a custom canvas page within a model-driven app, pass the page name as the first argument.

Power Fx

```
Navigate( myCustomPage )
```

Navigate to the default view of the table

To navigate to the default view of the table, pass the table name as the first argument.

Power Fx

```
Navigate( Accounts )
```

Navigate to specific system view of the table

To navigate to a specific system view of the table, pass the table's `Views` enum.

Power Fx

```
Navigate( 'Accounts (Views)'.'My Active Accounts' )
```

Navigate to the default form of the table

To navigate to the default form of the table, pass the record as the first argument.

Power Fx

```
Navigate( Gallery1.Selected )
```

Navigate to the default form of the table in create mode

To navigate to the default form of the table, pass a Dataverse record created from the Defaults function. This opens the default form with the record as a new record. The **Defaults** function takes the table name to create the record.

Power Fx

```
Navigate( Defaults( Accounts ) )
```

Optimize the user experience with data source and record information

Use the **DataSourceInfo** function and **RecordInfo** function to optimize the user experience with information about the data being displayed and manipulated.

For example, use **RecordInfo** to determine if the current user has permission to modify a record and appropriately show or hide an "Edit" button using its **Visible** property:

Power Fx

```
EditButton.Visible =
   RecordInfo( Gallery1.Selected, RecordInfo.EditPermission )
```

For example, use **DataSourceInfo** to determine if the current user has permission to create a record and appropriately show or hide a "Create" button using its **Visible** property:

Power Fx

```
CreateButton.Visible =
   DataSourceInfo( Accounts, DataSourceInfo.CreatePermission )
```

Ask for confirmation before taking action

Use the **Confirm** function to display a dialog box on top of the current screen.

Power Fx

```
Notify( Confirm( "Are you sure?",
                 { ConfirmButton: "Yes", CancelButton: "No" }
        )
)
```

Displays a notification **true** if the **Yes** button is pressed, and a notification **false** if the **No** button is pressed.

Notify the user

A notification can be shown to app users by calling the Notify function.

> **Note**
>
> `NotificationType.Success` isn't currently supported and results in an informational notification type.

Power Fx

```
Notify( "Model-driven app notification message" )
```

Other examples

Launch a URL

PowerApps Formula

```
Launch("https://www.bing.com");
```

Access 1:N property

PowerApps Formula

```
Self.Selected.Item.'Recurring Appointments'
```

Check property of a related record

PowerApps Formula

```
Self.Selected.Item.'Parent Account'.'Account Name'="parent"
```

Functions available with commanding

For information about the formulas supported with commanding in model-driven apps, go to Formula reference - model-driven apps.

Functions not supported

The following Power Fx functions are *currently not supported* with commanding in model-driven apps.

- Back()
- Clear()
- Collect()
- Copy()
- Disable()
- Enable()
- Exit()
- InvokeControl()
- Language()
- LoadData()
- Param()
- Print()
- ReadNFC()
- RequestHide()
- ResetForm()
- Revert()
- SaveData()
- ScanBarcode()
- Set()
- SubmitForm()
- UpdateContext()
- User()
- ViewForm()

Enums not supported

- Align
- AlignInContainer
- BarcodeType
- BorderStyle
- Color
- Direction
- DisplayMode
- Font
- FontWeight
- FormPattern
- GridStyle
- ImagePosition

- ImageRotation
- LabelPosition
- Layout
- LayoutAlignItems
- LayoutDirection
- LayoutJustifyContent
- LayoutMode
- LayoutOverflow
- ListItemTemplate
- MapStyle
- Overflow
- PDFPasswordState
- PenMode
- RemoveFlags
- ScreenTransition
- TeamsTheme
- TextFormat
- TextMode
- TextPosition
- Themes
- Transition
- VerticalAlign
- VirtualKeyboardMode
- Zoom

Other unsupported areas

- Acceleration
- App
- Compass
- Connection
- Dataverse file type columns
- Environment
- Host
- Layout
- Location
- ScreenSize

See also

Understand behavioral formulas

Formula reference

Overview of Power Fx

Open custom pages as dialogs and use cloud flows

The sample in this article demonstrates how you can build simple custom pages that open as dialogs when a command is selected. You might also call a cloud flow from the custom page dialog.

Prerequisites

- Add a custom page to your model-driven app
- Optionally trigger a cloud flow from the custom page

Create a command

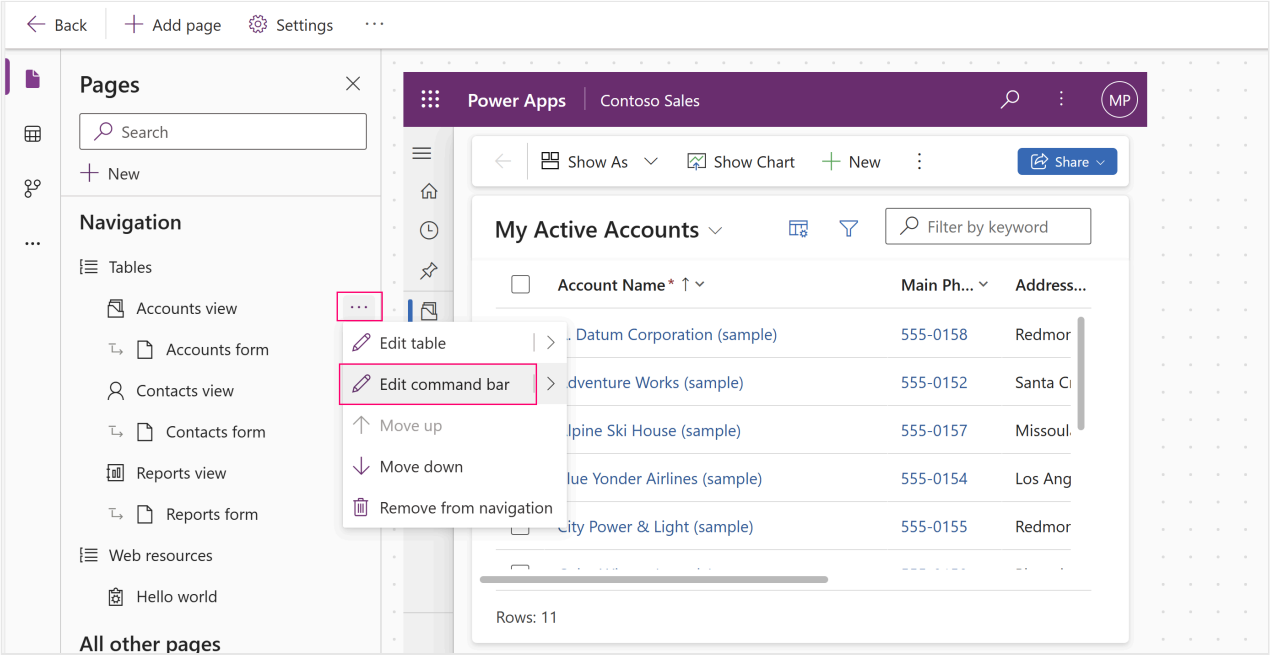
First, you need to determine the table and command bar location to place the command.

1. Open the app designer, and then add the desired table to your model-driven app. More information: Build your first model-driven app

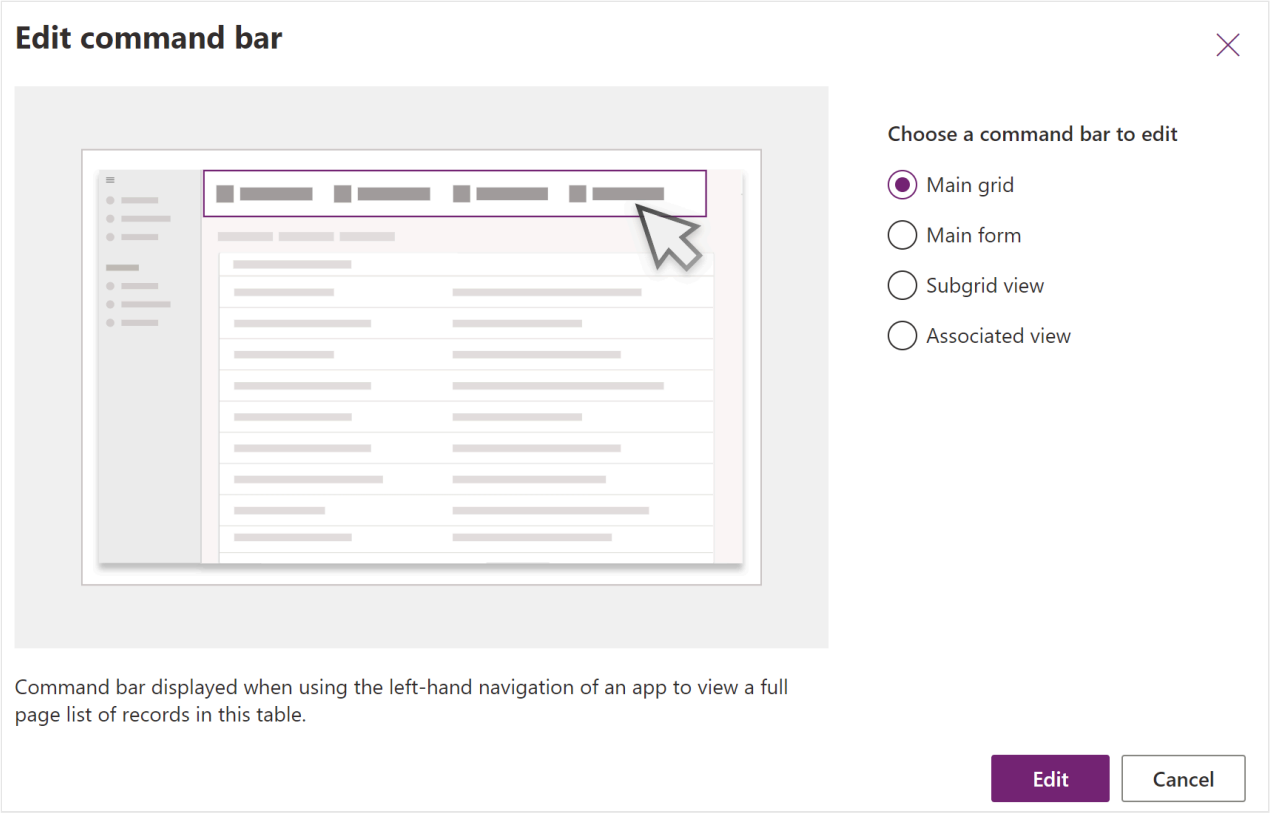
2. Publish the app.

3. Select the desired table view from the **Pages** area in the app designer.

4. Select **...** (ellipsis), and then select **Edit command bar**.



5. Select the location of the command bar you want, and then select **Edit**. More information: Command bar locations



6. Select + **New** on the command designer command bar, and then enter the command properties. More information:

   - Create a JavaScript web resource for your command.
   - For other properties, see Create a new command.



Create a JavaScript web resource for your command

> **Note**
>
> This commanding customization currently supports only JavaScript. Power Fx isn't supported.

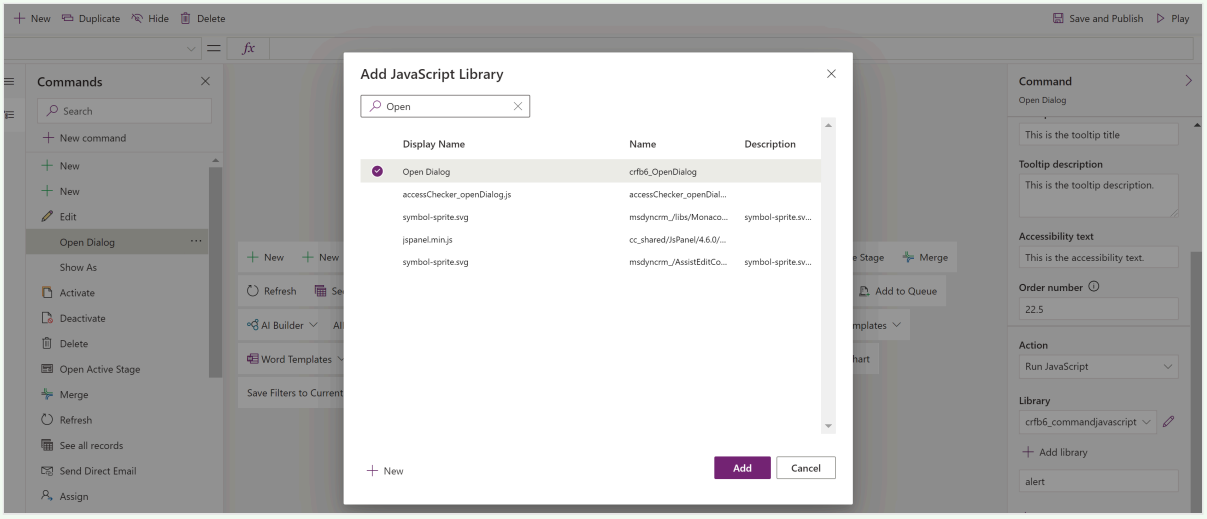
1. For the command properties **Action**, select **Run JavaScript**.
2. Select + **Add library**.
3. On the **Add JavaScript Library** list, select + **New web resource** to create a new JavaScript web resource.
4. On the **New web resource** pane, enter a **Name**, and optionally a **Display name** and **Description**.
5. For **Type**, select **JavaScript (JS)**.

6. In the **Code** box, paste your JavaScript. For example, depending on the type of dialog you want to use, copy and paste an example from the Navigate API reference .
7. Select **Save and publish** to save and publish the web resource.
8. Close the browser window tab to return to the command designer.

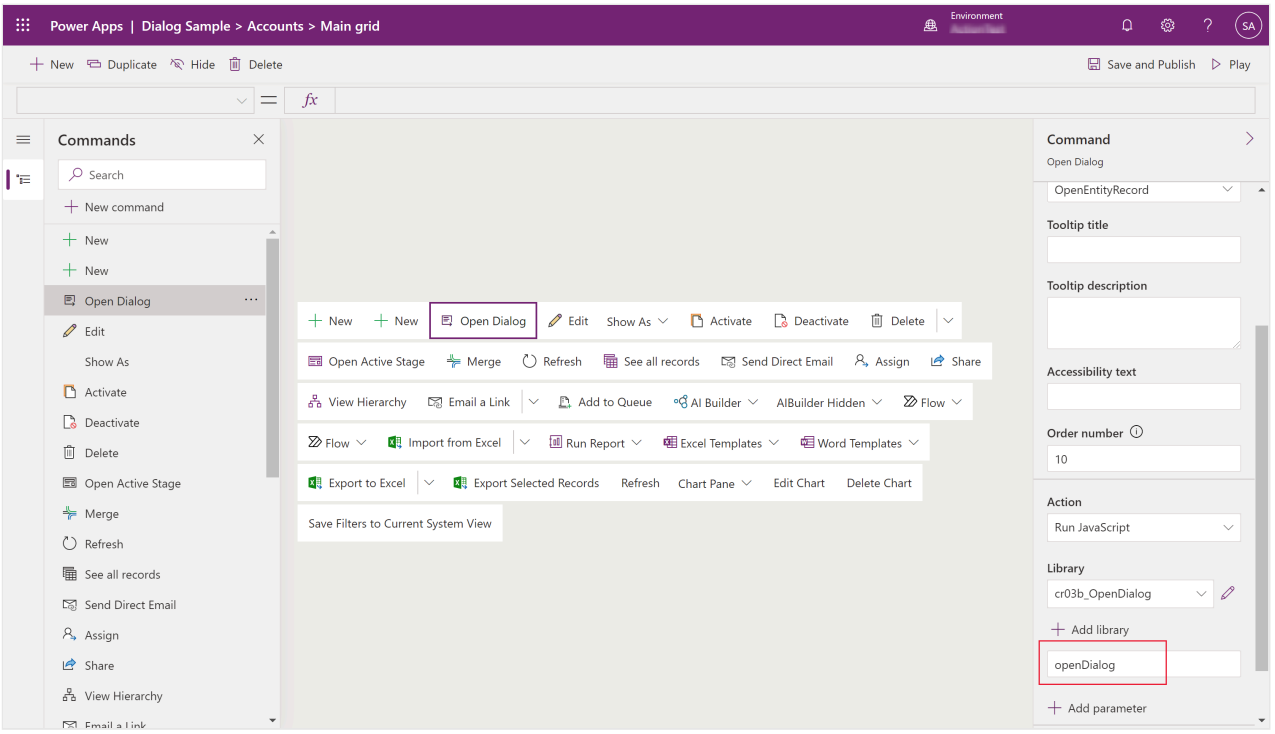
Call the JavaScript from your command

1. In the **Add JavaScript Library** dialog, select the library (web resource) you created in the previous steps, and then select **Add**.

> **Tip**
>
> If you can't find the web resource, enter the name in the **Search** box.



2. In the **Command** properties pane, if the name isn't specified, enter the name of the JavaScript **Function**. In this example, *openDialog* is entered.



3. Optionally, change the **Visibility** logic.

4. Select **Save and Publish**.

See also

Design a custom page for your model-driven app

Navigating to a custom page using client API

Using PowerFx in custom page

Last updated on 03/31/2026

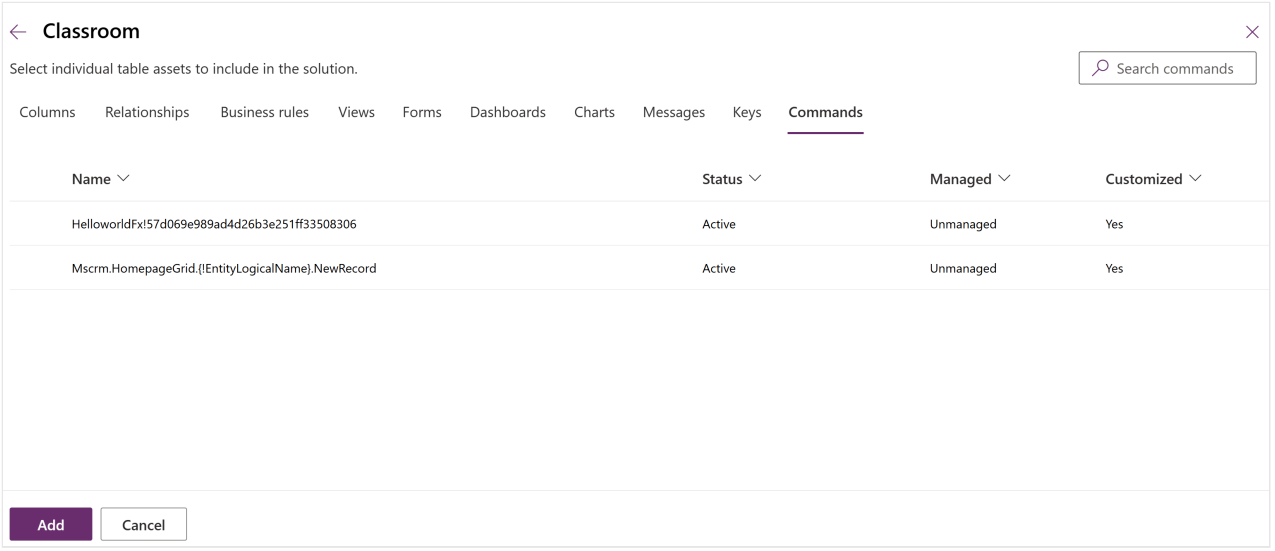
Manage commands in solutions

Article • 04/10/2025

Unlike classic commands, modern commands are treated the same as other solution objects and support core solution behaviors. You can also edit existing modern commands from within a solution.

To add commands to a solution, you add the solution objects that depend on your command actions:

- Modern commands are a table component. When you add a command to a solution, you must add the command component. In this screenshot there's one JavaScript and one Power Fx command for the custom Classroom table that can be added as a specific object to a solution for export.



- Power Fx commands use a Dataverse component library. Add the component library to your solution if you plan to export it.
- JavaScript commands use a web resource. Add the web resource that includes the underlying JavaScript to run your command to the solution if you plan to export it.

More information: Add an existing component to a solution

Modern commanding provides these benefits when you work with solutions:

- Block commands from being customized by using managed properties. Open the table, select **Commands**, select the command, and then on the command bar select **Advanced** > **Managed properties** to disable **Allow customizations**.
- View dependencies related to your modern command components. More information: View dependencies for a component in Power Apps
- Troubleshoot by viewing which solutions modified the component and when using solution layers. More information: Solution layers

- Remove any unmanaged customizations if in an unhealthy state. More information: Remove an unmanaged layer
- Segment solutions and build minor updates or patches that include your modern command components.
- Modern command components work with solution upgrade so you can delete components by importing a new version. Upgrade or update a solution
- Standardized localization via export/import translations. More information: Translate label text

Related articles

Modern commanding overview

Solutions overview

Scopes for modern commands

Article • 11/16/2022

Commands have one of three scopes. The scope determines whether a command (`schemaname = appaction`) will render for a table within a single app, for a table across all apps, or for all tables and all apps.

A command must be bound to a command bar location such as main grid or main form regardless of scope. Therefore, changing the scope will not render the command in different command bar locations such as main grid and main form.

- **App** This is the narrowest scope and **default behavior** when creating or editing commands using the modern command designer. Command designer is opened within the context of a model-driven app, table, and command bar location.
- **Table** Also known as entity scope. Bound to a specific table and command bar location. Not bound to any specific app. Therefore table scoped commands will render in all apps that use the table. For example, a table scope command for the account table and main form location will be present in all apps when viewing the main form for an account record.
- **Global** This is the broadest scope. Global scope commands are only bound to a command bar location, not an app or table. For example, a global scope command for the main form location is present in all apps within an environment when viewing the main form for every table.

> **ⓘ Important**
>
> Commands using Power Fx for the action and/or visibility can't be converted to table or global scope at this time.
>
> When commands are nested within dropdown lists, groups, and split buttons, all must be set to the same scope.

If a global or table scope command is edited within modern command designer, a copy (instance) is created with app scope and linked to the original command. For example, editing a global scope command in command designer overrides the original global scope command but only for the specific app and table chosen when opening command designer. All other apps and tables will will render the original command. The copy (instance) now has it's own lifecycle. Changes to the original global scope command will not effect the new app scope instance and vice versa.

Override table and global scoped commands

More narrow scopes can override a broader scope command at every level. The most narrow scope wins.

- A **table (entity) scoped** command overrides global scoped commands. You can define a command that will be the same across all tables, but modify the behavior only for specific tables.
- An **app scoped** command overrides all other scopes but only for a particular app and table.

How to create a table scope command

1. Create one or more commands within your solution using command designer. Alternatively, create a separate solution and select **Add existing** > **Table**.
2. Choose **Select objects**, and then add the desired commands.
3. Select **Export** to export your solution.



4. If you've made recent changes that haven't yet been published, select **Publish**. Then, select **Run** to check whether the solution has any issues or dependencies, and then select **Next**.
5. Select the **Unmanaged** option, and then select **Export**.
6. In the **Download** dialog box, select **Save**, and in the **Download complete** dialog box, select **Open Folder**.
7. Right-click to select the compressed .zip file that you downloaded, and then select **Extract All**.
8. Select a location to extract the files to, and then select **Extract**.
9. Open the folder to the extracted files, open the appaction.xml file, and find the desired command (appaction).
10. Delete the **appmoduleid** node from the xml.

```
XML
```

```
<appmoduleid>
  <uniquename>demo_DemoFestApp</uniquename>
</appmoduleid>
```

11. Change the value of the **appaction uniquename**. Changing any one of the last characters is sufficient, such as from 1 to 2, or you can use more descriptive naming conventions if desired.

Example XML.

XML

```
   <appaction
uniquename="crdff_NewCommand!a078463b5d7c473d8965f0f80469f412!crdff_CustomAp
p!crdff_entity1!1">
  <buttonlabeltext default="Show Alert">
    <label description="Show Alert" languagecode="1033" />
   </buttonlabeltext>
   <buttonsequencepriority>10.0000000000</buttonsequencepriority>
   <context>1</context>
   <contextentity>
    <logicalname>crdff_entity1</logicalname>
   </contextentity>
   <fonticon>AALinkedInLogo</fonticon>
   <contextvalue>crdff_entity1</contextvalue>
   <hidden>0</hidden>
   <iscustomizable>1</iscustomizable>
   <location>1</location>
   <name>EntityScoped!a078463b5d7c473d8965f0f80469f412</name>

<onclickeventjavascriptfunctionname>ShowAppAlert</onclickeventjavascriptfunc
tionname>
   <onclickeventjavascriptparameters>[]</onclickeventjavascriptparameters>
   <onclickeventjavascriptwebresourceid>
    <webresourceid>ac21ad24-3c11-ef11-b6u7-000d3a1d942c</webresourceid>
   </onclickeventjavascriptwebresourceid>
   <onclickeventtype>2</onclickeventtype>
   <statecode>0</statecode>
   <statuscode>1</statuscode>
   <type>0</type>
   </appaction>
```

After you've edited and saved the appaction.xml file, zip your solution in the same format it was exported in. Then **Import** the solution back into your environment and test the behavior.

How to create a global scope command

1. Follow the same steps above to edit the appaction.xml file.

2. **Delete** the **appmoduleid** node from the xml:

```
XML

 <appmoduleid>
   <uniquename>demo_DemoFestApp</uniquename>
 </appmoduleid>
```

3. Change the value of the **appaction uniquename**. Changing any one of the last characters is sufficient, such as from 1 to 2, or you can use more descriptive naming conventions if desired.

4. **Delete** the **contextentity** and **contextvalue** nodes from the xml:

```
XML

<contextentity>
 <logicalname>account</logicalname>
</contextentity>
<contextvalue>account</contextvalue>
```

5. Change the **context** value from 1 to 0.

```
XML

<appaction uniquename="crdff_NewCommand1!8fe72a85-1f84-431e-ac56-
854f1bfadc4e!1">
<buttonlabeltext default="Show Alert">
<label description="Show Alert" languagecode="1033" />
</buttonlabeltext>
<buttonsequencepriority>10.0000000000</buttonsequencepriority>
<context>0</context>
<hidden>0</hidden>
<iscustomizable>1</iscustomizable>
<fonticon>AALinkedInLogo</fonticon>
<location>1</location>
<name>EntityScoped!a078463b5d7c473d8965f0f80469f412</name>
<onclickeventjavascriptfunctionname>ShowGlobalAlert</onclickeventjavasc
riptfunctionname>
<onclickeventjavascriptparameters>[]</onclickeventjavascriptparameters>
<onclickeventjavascriptwebresourceid>
 <webresourceid>ac21ad24-3c01-ec11-b6e7-000d3a1d942c</webresourceid>
</onclickeventjavascriptwebresourceid>
<onclickeventtype>2</onclickeventtype>
<statecode>0</statecode>
<statuscode>1</statuscode>
<type>0</type>
</appaction>
```

After you've edited and saved the appaction.xml file, zip your solution in the same format it was exported in. Then **Import** the solution back into your environment and test the behavior.

How to override a global scope command with a table scope

This example is for when you want a global scope command to be the same everywhere *except* for a certain table.

- **Entity Scoped Action** will be visible on `crdff_entity1` grids in all apps except in `crdff_CustomApp`.
- **Global Scoped Action** will be visible on all the entity grids except `crdff_entity1`.

> ⓘ Note
>
> To override either global or table scope commands with an app scope command, simply edit the command using command designer.
>
> To override a global scoped command with a table scope command, modify the appactions.xml files so they have the same `XML <name>` .

Steps to override

1. Follow the steps in How to create a global scope command, but create two commands using command designer.
2. As described in the steps, export your solution and edit the appactions.xml files.
3. Continue following the steps to create one global and one entity scope command.
4. Set the **name** property to be the exact same for both the global and entity scope commands.

Example global scope command XML

XML

```
<appaction uniquename="crdff_NewCommand1!8fe72a85-1f84-431e-ac56-
854f1bfadc4e!1">
```

```
<buttonlabeltext default="Show Alert">
 <label description="Show Alert" languagecode="1033" />
</buttonlabeltext>
<buttonsequencepriority>10.0000000000</buttonsequencepriority>
<context>0</context>
<hidden>0</hidden>
<iscustomizable>1</iscustomizable>
<fonticon>AALinkedInLogo</fonticon>
<location>1</location>
<name>EntityScoped!a078463b5d7c473d8965f0f80469f412</name>
<onclickeventjavascriptfunctionname>ShowGlobalAlert</onclickeventjavascriptf
unctionname>
<onclickeventjavascriptparameters>[]</onclickeventjavascriptparameters>
<onclickeventjavascriptwebresourceid>
 <webresourceid>ac21ad24-3c01-ec11-b6e7-000d3a1d942c</webresourceid>
</onclickeventjavascriptwebresourceid>
<onclickeventtype>2</onclickeventtype>
<statecode>0</statecode>
<statuscode>1</statuscode>
<type>0</type>
</appaction>
```

Example table scope command XML

XML

```
<appaction
uniquename="crdff_NewCommand!a078463b5d7c473d8965f0f80469f412!crdff_entity1!
1">
<buttonlabeltext default="Show Alert">
<label description="Show Alert" languagecode="1033" />
</buttonlabeltext>
<buttonsequencepriority>10.0000000000</buttonsequencepriority>
<context>1</context>
<contextentity>
<logicalname>crdff_entity1</logicalname>
</contextentity>
<fonticon>AALinkedInLogo</fonticon>
<contextvalue>crdff_entity1</contextvalue>
<hidden>0</hidden>
<iscustomizable>1</iscustomizable>
<location>1</location>
<name>EntityScoped!a078463b5d7c473d8965f0f80469f412</name>
<onclickeventjavascriptfunctionname>ShowEntityAlert</onclickeventjavascriptf
unctionname>
<onclickeventjavascriptparameters>[]</onclickeventjavascriptparameters>
<onclickeventjavascriptwebresourceid>
<webresourceid>ac21ad24-3c01-ec11-b6e7-000d3a1d942c</webresourceid>
</onclickeventjavascriptwebresourceid>
<onclickeventtype>2</onclickeventtype>
<statecode>0</statecode>
<statuscode>1</statuscode>
```

```
<type>0</type>
</appaction>
```

Modern commanding known limitations

This article describes the known limitations with modern commanding for model-driven apps.

Power Fx and command component libraries limitations

Power Fx shares certain infrastructure with custom pages and canvas apps. Relevant limitations of custom pages also apply to commands.

- Not all Power Fx functions are currently supported for commands.
- Microsoft Dataverse is currently the only supported data source when using Power Fx with commands. However, custom pages can be used for connecting to external data.
- To add additional tables, open the command component library from command designer. Or you can reopen command designer by selecting a different table in app designer.
- When you write Power Fx formulas in the command designer, IntelliSense might sometimes provide recommendations for unsupported functions. It might not show an error for unsupported functions within command designer or the associated command component library.
- Commands and the command component library created from one app can't be added to different apps. However, you can copy the command and paste it within another app or command bar location.
- Not all out-of-the-box or custom visibility rules from classic commands are currently supported in Power Fx. Classic visibility is supported without using Power Fx. This is needed to migrate classic commands to modern and classic rule formats but won't be exposed in command designer. However, classic visibility is exposed within solution files and Dataverse.
- To delete command component libraries, you must delete the app. Alternatively, you can delete the associated record within the model-driven app element table, then delete the component library. The model-driven app element record exists in Dataverse but isn't exposed in solutions. More information: Delete a Model-Driven App Element record
- Power Fx based commands aren't supported and won't run within the Dynamics 365 app for Outlook or a model-driven app that's hosted within a Portal.
- Metadata changes for attributes might not reflect in the Power Fx expression. To update the metadata, open the component library using command bar actions in command

designer. In Power Apps Studio for canvas apps select **Data sources**, select the data source, and then refresh. This action updates the metadata for the component library.

Command designer limitations

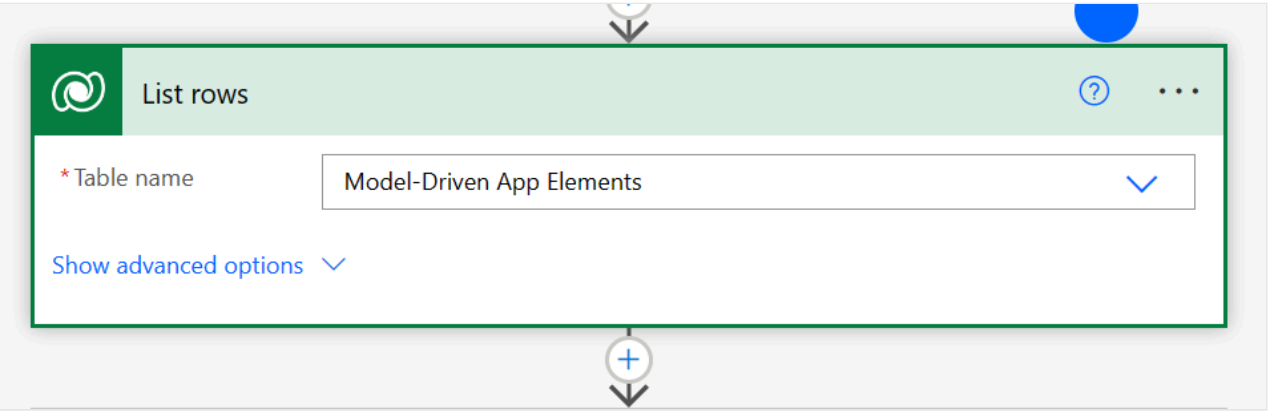
- Pre-existing classic commands can't be customized within the command designer until they're migrated to the modern infrastructure. This includes the out-of-the-box commands, which will be migrated incrementally over time. You can continue to use other mechanisms and third-party tools for customizing classic commands.
- Dynamically populated buttons aren't supported. We recommend creating them declaratively.
- Global application header and dashboard command bars aren't currently supported in command designer. These are customized infrequently. Share feedback if you find this valuable to prioritize.
- A single Power Fx component library is supported. The command designer might display the error message: "Unable to initialize component manager. There are multiple component libraries associated with your app." This error occurs when an app has multiple duplicate component libraries associated with it, which currently isn't supported with model-driven apps. To resolve this limitation, remove the additional `AppElement` entries for the app. Do this by removing the multiple library dependencies with the app, ensuring that only one `AppElement` entry exists.
- Editing commands on the commands page from the **Solutions** or **Tables** areas in Power Apps (make.powerapps.com) doesn't have the capability to set `Run formula` as the action or `Show on condition from formula` as the visibility rule. These capabilities are only available when editing commands from within the modern app designer.

Delete a model-driven app element record

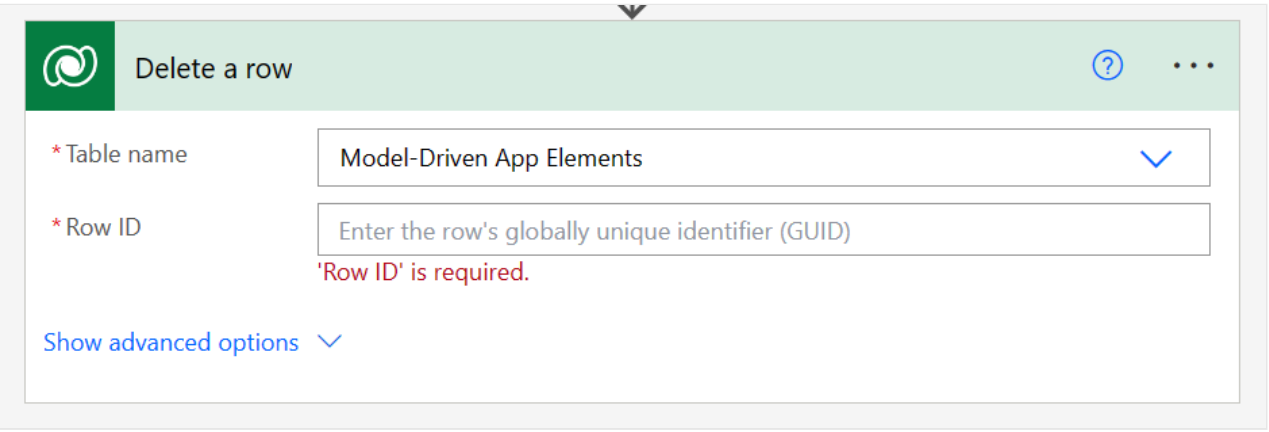
> **⚠ Warning**
>
> If the app includes Power Fx-based commands, they stop working after you delete the app's model-driven app element table record.

Using the **Model-driven App Elements** table, complete the following steps:

1. Create a flow to get a list of rows. More information: Get a list of rows



2. Then, run the flow to find the **Row id unique** value for the row you want to delete and copy it. You can also find this row value by viewing the table rows in the table hub for the Model-driven App Element table at make.powerapps.com.
3. Create a step to delete the row by using the unique ID value you copied in the previous step. More information: Delete a row.



4. Publish the model-driven app associated with the model-driven app element record.

See also

Modern commanding overview

Introducing Command Checker for model-app ribbons

Last updated on 04/08/2026

Design model-driven apps using the classic app designer

Summarize this article for me

With Power Apps, model-driven apps are comprised of components such as tables, forms, views, dashboards, and business process flows.

As model-driven apps have evolved so too have the ways in which the various elements are edited. This article describes the method of creating an app using the classic app designer.

> **Important**
>
> This article describes how to use the classic app designer. Use the modern app designer to build your apps. More information: Create a model-driven app using the app designer

The app designer helps all these components to be brought together quickly. Its tile-based information structure and simplified interface make the process of building an app much easier, and apps that are specific to business roles and functions can be brought together without having to write any code.

Each app that you create can have its own site map with the integrated and easy-to-use site map designer. Just drag and drop areas, group, and subareas to the canvas. Components that you select in the site map are also added as tables in the app designer.

Tables can be added and removed as necessary, and also add other components.

Once components have been added, the app can be validated to confirm that all components have been added correctly.

The following table shows the steps required to create an app and assumes that all the necessary tables, forms, views, dashboards and relationships have been created within the environment in advance.

Expand table

Step	Description	Related topics
1	Define app properties.	Create or edit an app
2	Define navigation for an app using the site map designer.	Create a site map for an app

Step	Description	Related topics
3	Apps are composed of components like dashboards, tables, business process flows, forms, views, and charts. Include the required ones in your app by using the app designer.	Add or edit app components
4	Check the app for any required components that have not been added. After all required components are added make the app available for use.	Validate and publish an app
5	Give users access to the data within the apps created by using security roles.	Share a model-driven app

Supported web browsers

For information about the supported web browsers to use with the app designer and site map designer, see Supported web browsers and mobile devices.

Next steps

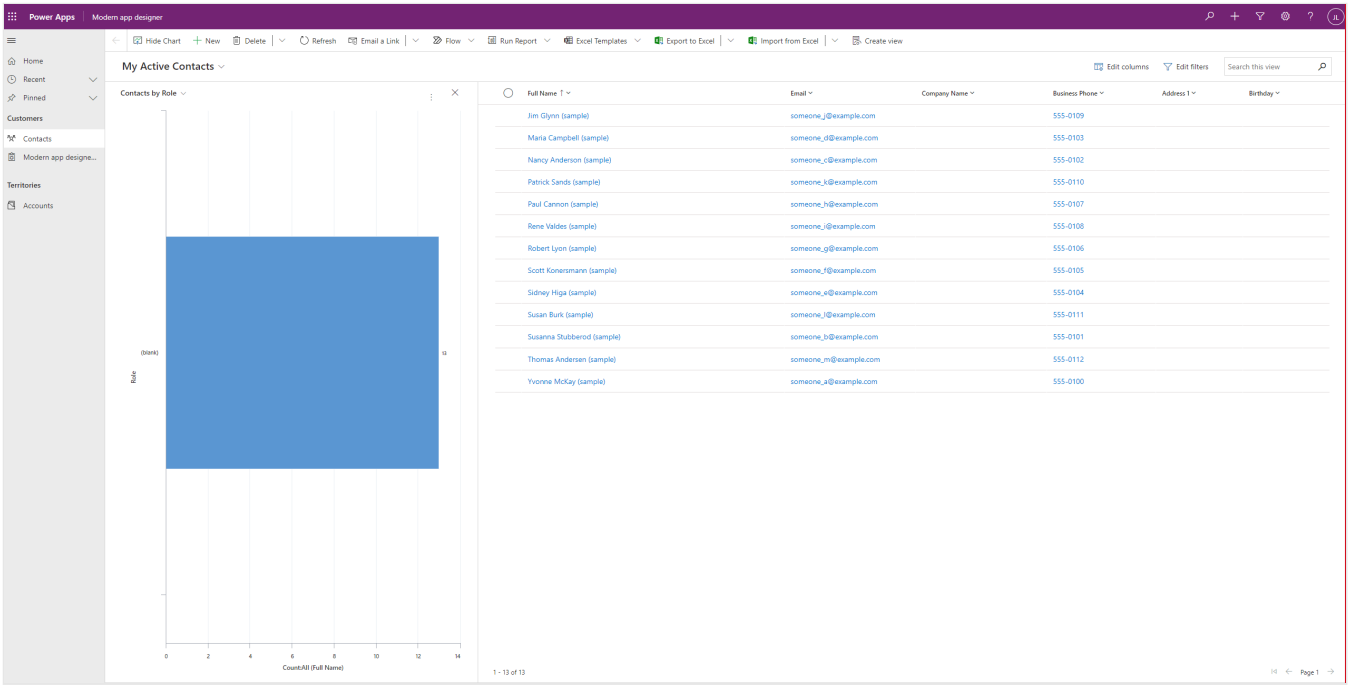
Build your first model-driven app from scratch

Last updated on 02/13/2026

Build your first model-driven app using the classic designer

> **Important**
>
> Starting in October 2023, the classic app, form, and view designers are deprecated and all model-driven apps, forms, and views will only open in the modern designers. By default, the **Switch to classic** command to revert back to the classic designer from the modern designer will no longer be available. More information: Classic app, form, and view designers are deprecated
>
> We recommend that you transition to use the modern designers to create and edit your model-driven apps and components.

In this article, you build a straightforward model-driven app that you can use to create, update, and delete account records as well as view data in one of the standard charts for the account table.

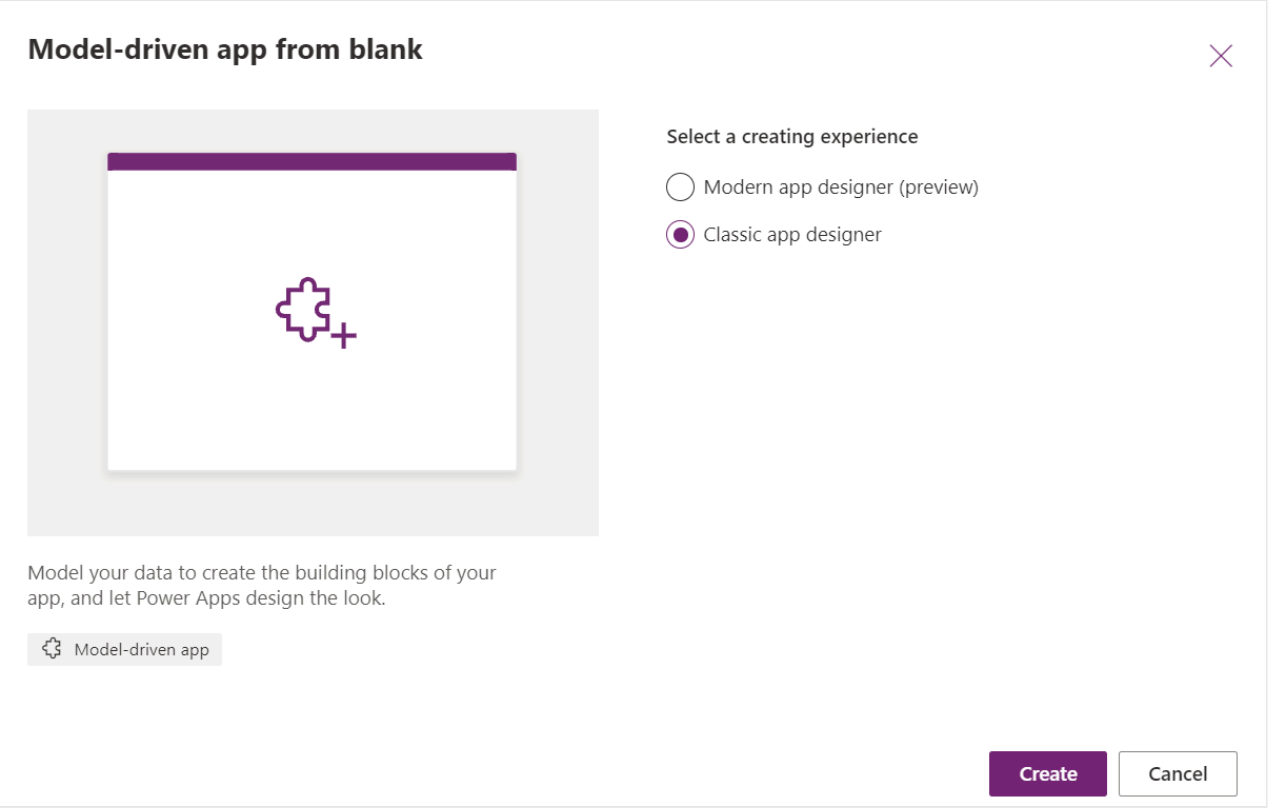


Create your model-driven app

> **Note**
>
> If you plan to share and distribute the app you create, see this article: Create a model-driven app by using the app designer

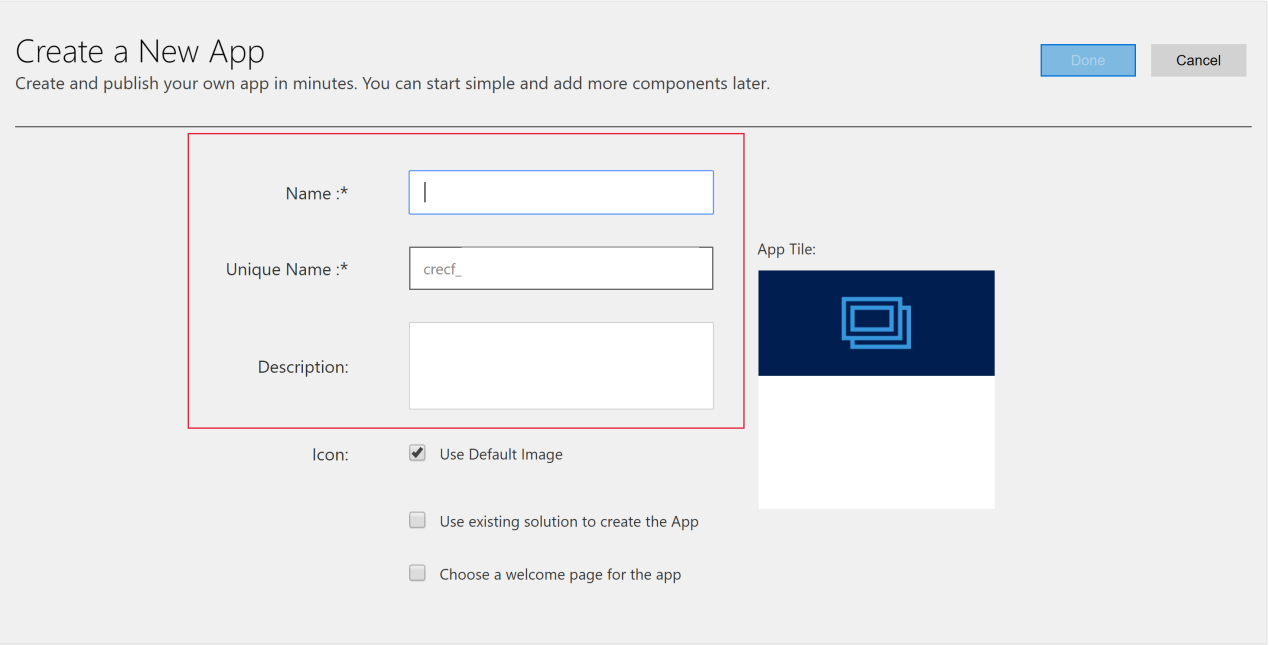
1. Sign in to Power Apps. If you don't already have a Power Apps account, you can get started free.

2. On the **Home** page, select **Blank app**, under **Blank app based on Dataverse** select **Create**. Select the **classic app designer** option and then **Create**.

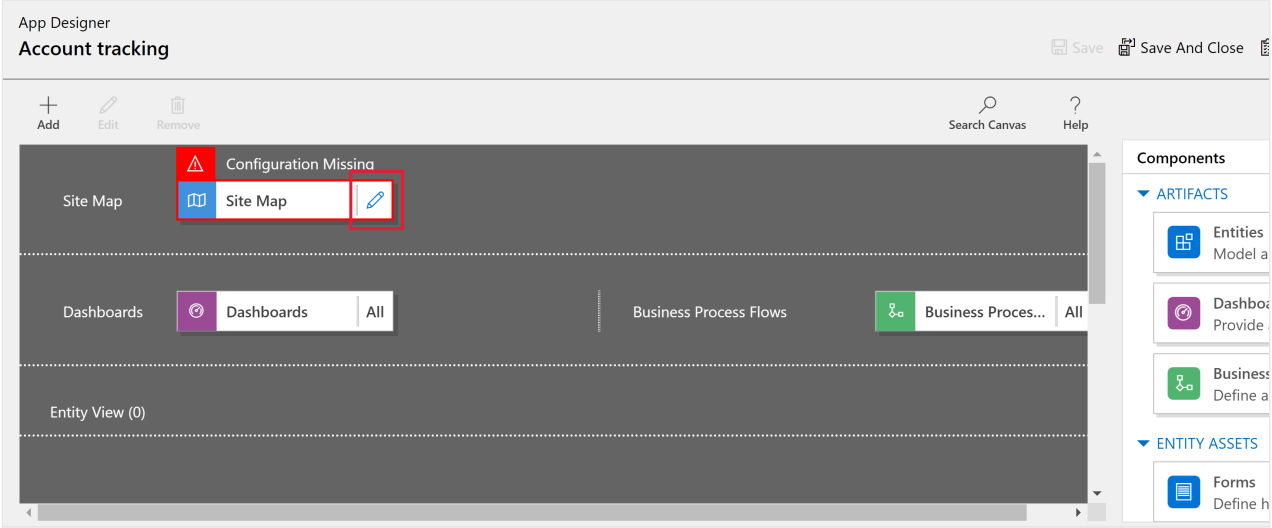


3. On the **Create a New App** page, enter the following details, and then select **Done**:

- **Name**: Enter a name for the app, such as *Account tracking*.
- **Unique Name**: By default, the unique name uses the name you specify in the **Name** box without spaces and preceded by the publisher prefix and an underscore (_). For example, *crecf_Accounttracking*.
- **Description**: Type a short description of what the app is or does, such as *This is my first app*.



4. Select the **Open the Site Map Designer** edit button (pencil icon) to open the sitemap designer.



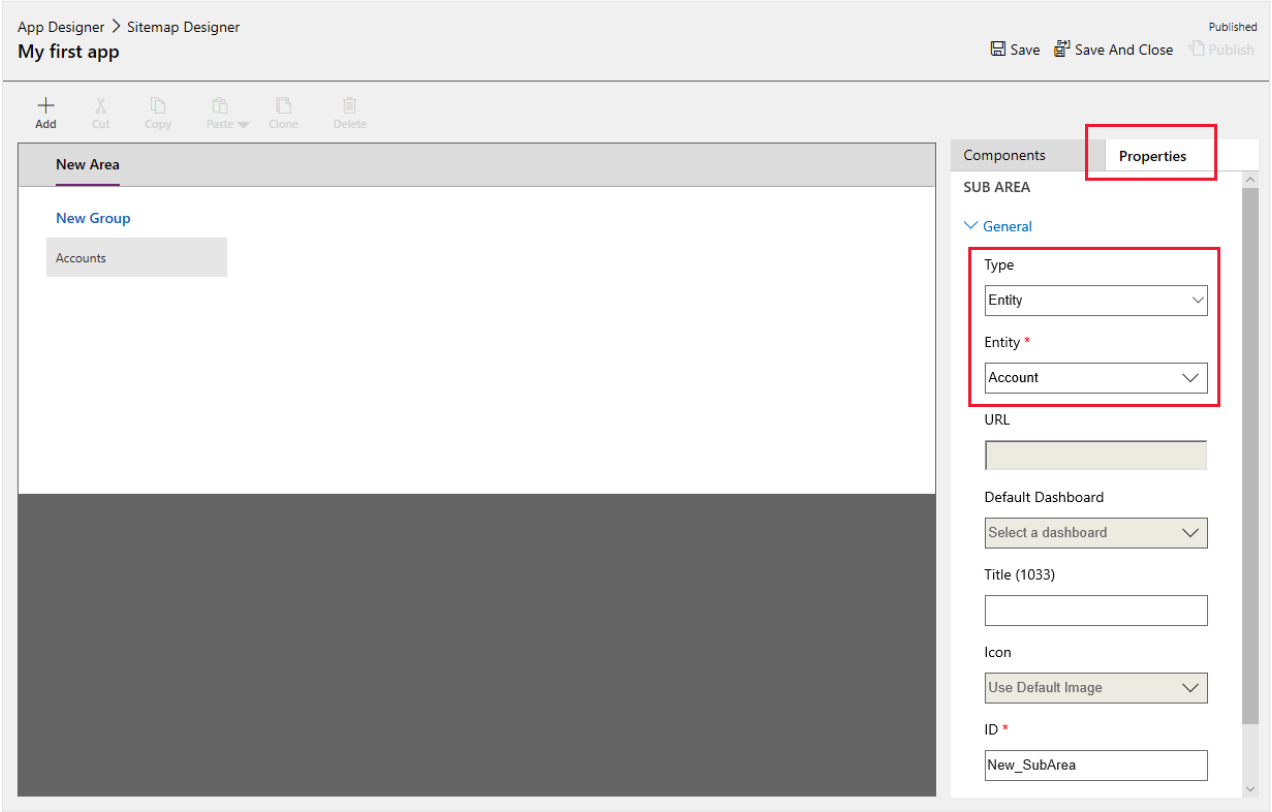
Notice that when you create a new app, you must first create the site map for it.

5. On the sitemap designer, select **New Subarea**.



6. In the right pane, select the **Properties** tab, and then select the following properties.

- **Type**: **Entity**
- **Entity**: **Account**



When you leave the **Title** property blank, the app uses the table name in the app's left navigation pane. For this app, **Accounts** will be displayed in the app at runtime.

7. Select **Save And Close** to close the sitemap designer.

   By default, all the account table's forms, views, charts, and dashboards are enabled for the app. From the app designer **Components** tab on the right pane, you can clear a component so it isn't available in the app at runtime. You can also create new components, such as a custom form. For this app, leave all components enabled.

8. On the app designer toolbar, select **Save**.

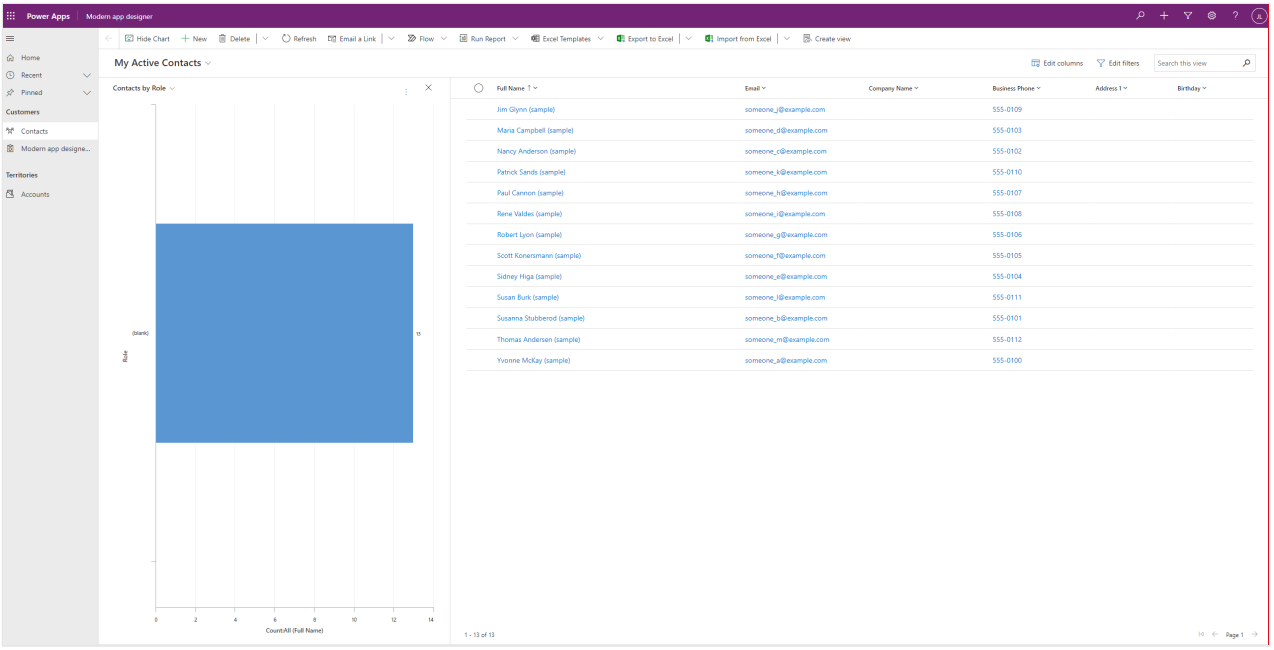
9. After your app is saved, on the app designer toolbar, select **Publish** to make it available to run and share.

Play your app

On the app designer toolbar, select **Play**.

- To create a record, select **+ New**.
- To view a chart, on the app command bar select **Show Chart**.

- To change the view, select the **My Active Accounts** view, and then select the view you want.



To learn more about how to use your app, see Basic navigation in a model-driven app.

> **Note**
>
> Don't see any of the data in your app like in the above screenshot? This is probably because your environment database doesn't have **Deploy sample apps and data** provisioned. More information: Create an environment with a database

Next steps

In this article, you built a simple model-driven app.

- To identify the URL of your model-driven app see Run a model-driven app in a browser
- To see how your app looks when you run it, see Run a model-driven app on a mobile device.
- To learn how to share your app, see Share a model-driven app.
- To get started and learn more about building model-driven apps, see Understand model-driven app components

> **Tip**

Ready to convert your ideas into an app? Start here: **Planning a Power Apps project**.
For detailed information about model-driven apps and how to build it, start here: **Understand model-driven app components**.

Last updated on 03/12/2026

Create a model-driven app using the classic app designer

Article • 11/06/2024

This article describes the basics of creating and editing a model-driven app that can be shared and distributed to other environments.

> **Important**
>
> Starting in October 2023, the classic app, form, and view designers are deprecated and all model-driven apps, forms, and views will only open in the modern designers. By default, the **Switch to classic** command to revert back to the classic designer from the modern designer will no longer be available. More information: Classic app, form, and view designers are deprecated

Prerequisites to creating model-driven apps

Verify the following prerequisites before creating an app:

- A Power Apps environment used for app development must exist within the tenant. More information Create an environment and Environment strategy for ALM.
- The environment used will need to have a Dataverse database associated with it. Dataverse environments can exist with or without a database and generally these are provisioned on creation of the environment. The database holds the tables and other components that will be used by the model-driven app. Create and manage environments in Dataverse
- Within the environment the app developer needs to have an environment maker, system administrator, or system customizer security role. More information: About predefined security roles

Create a model-driven app

1. Sign in to Power Apps.

2. Choose the environment by selecting the environment icon 

3. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

4. Open an unmanaged solution or create a new one. More information: Create a solution

5. Select **New** > **App** > **Model-driven app**.

6. There a two options. Modern app designer and Classic App designer. Select **Classic App Designer**.

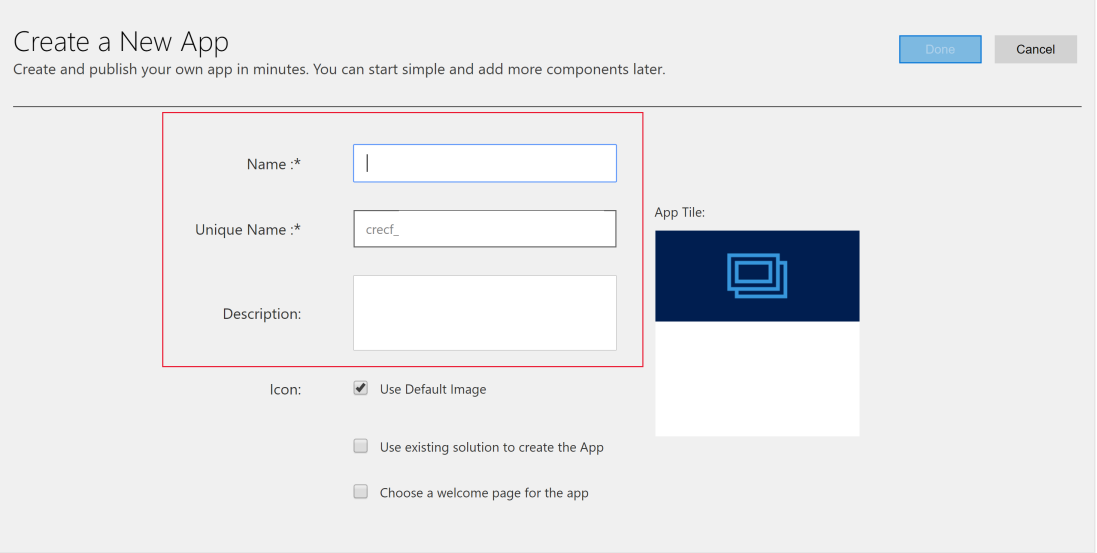
7. On the **Create a New App** page, enter the following details:

    - **Name**: Enter a name for the app.

    - **Unique Name**: The unique name is automatically populated based on the app name that is specified. It's prefixed with a publisher prefix. It's possible to change the part of the unique name that's editable. The unique name can only contain English characters and numbers.

        > **Note**
        >
        > The publisher prefix is the text that's added to any table or column created for a solution that has this publisher.

    - **Description**: Type a short description of what the app is or does.

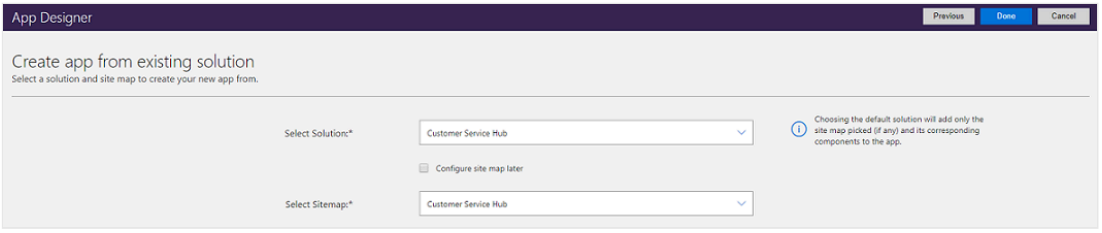


    - **Icon**: By default, the **Use Default Image** check box is checked. To select a different web resource as an icon for the app, clear the check box, and then select an icon from the drop-down list. This icon is displayed on the preview

tile of the app. For more information about how to create a web resource, go to Create or edit model-driven app web resources to extend an app.

- **Use existing solution to create the App (Optional)**: Select this option to create the app from a list of installed solutions. When this option is selected, **Done** switches to **Next** on the header. If **Next** is selected, the **Create app from existing solution** page opens. From the **Select Solution** drop-down list, select a solution. If any site map is available for the selected solution, the **Select Sitemap** drop-down list will appear. Select the site map, and then select **Done**.

  > **Note**
  >
  > By selecting **Default Solution** when the site map is added, the components that are associated with that site map are automatically added to the app.



- **Choose a welcome page (Optional)**: This option allows a designer to select from the web resources available in your organization. The welcome pages created can contain information that's useful to users, such as links to videos, upgrade instructions, or getting started information. The welcome page is displayed when an app is opened. Users can select **Do not show this Welcome Screen next time** on the welcome page to disable the page so it doesn't appear the next time the app starts. Notice that the **Do not show this Welcome Screen next time** option is a user-level setting and can't be controlled by administrators or app makers. More information about how to create a web resource, such as an HTML file that can be used as a welcome page: Create and edit web resources to extend the web application

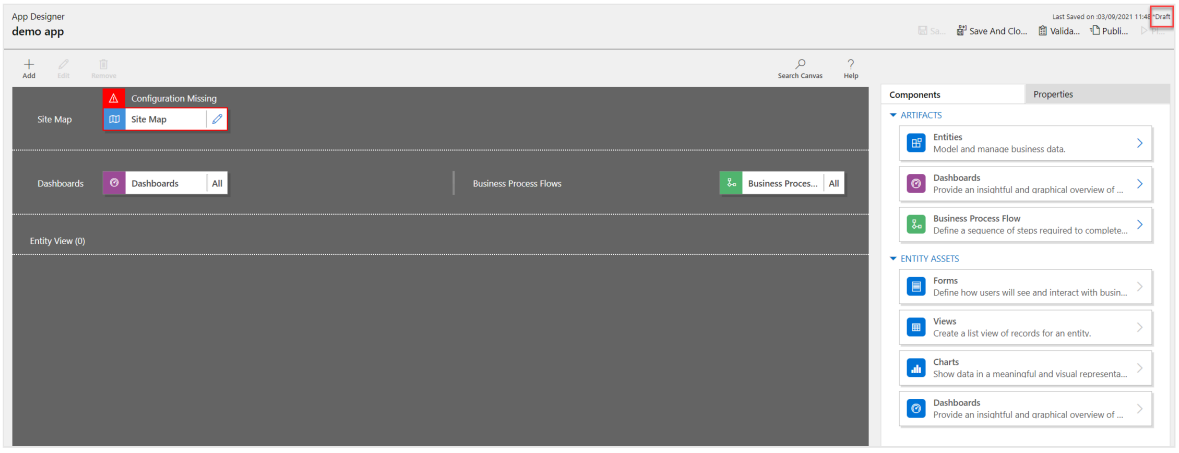
To edit app properties later, go to the **Properties** tab in the app designer. More information: Manage app properties

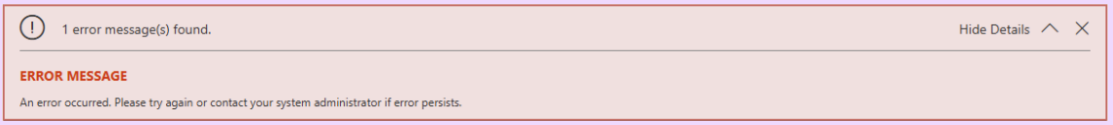
> **Note**

It is not possible to change the unique name and app URL suffix on the **Properties** tab.

8. Select **Done** or if **Use an existing solution to create the App** is selected, select **Next** to choose from the available solutions that were imported in the environment, then select **Done**.

9. A new app is created and is shown in **Draft** status. This is shown in the App designer.

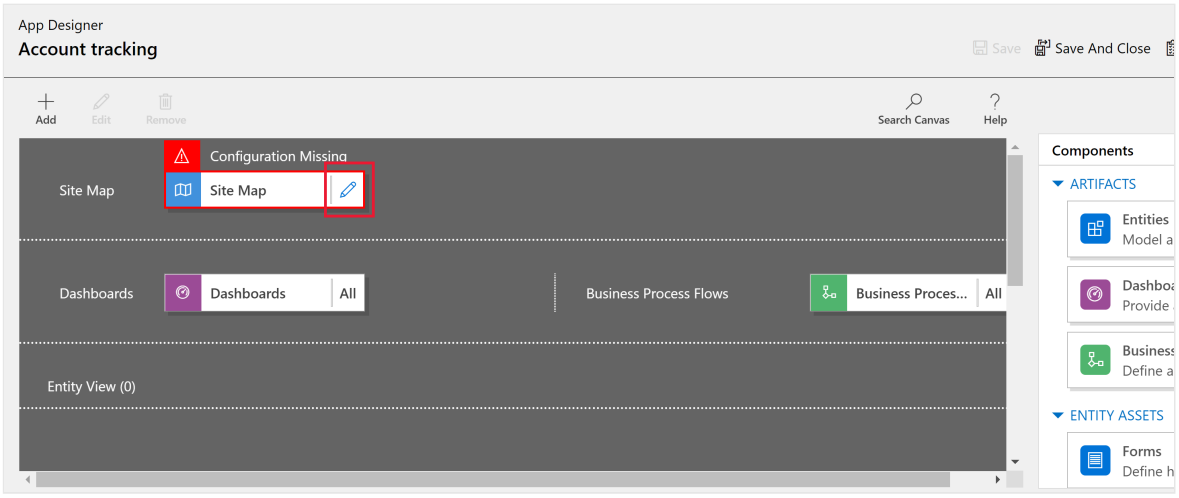


> **Note**
>
> When opening the classic app designer, an error might appear similar to the image here. When this occurs, it doesn't impact the classic app designer functionality and you can close the error and continue editing the app. However, to avoid this issue, use the Modern app designer.
>
> 
>

Configure the site map

The site map describes the components that make up a model driven app.

1. Select the **Open the Site Map Designer** edit button (pencil icon) to open the sitemap designer.



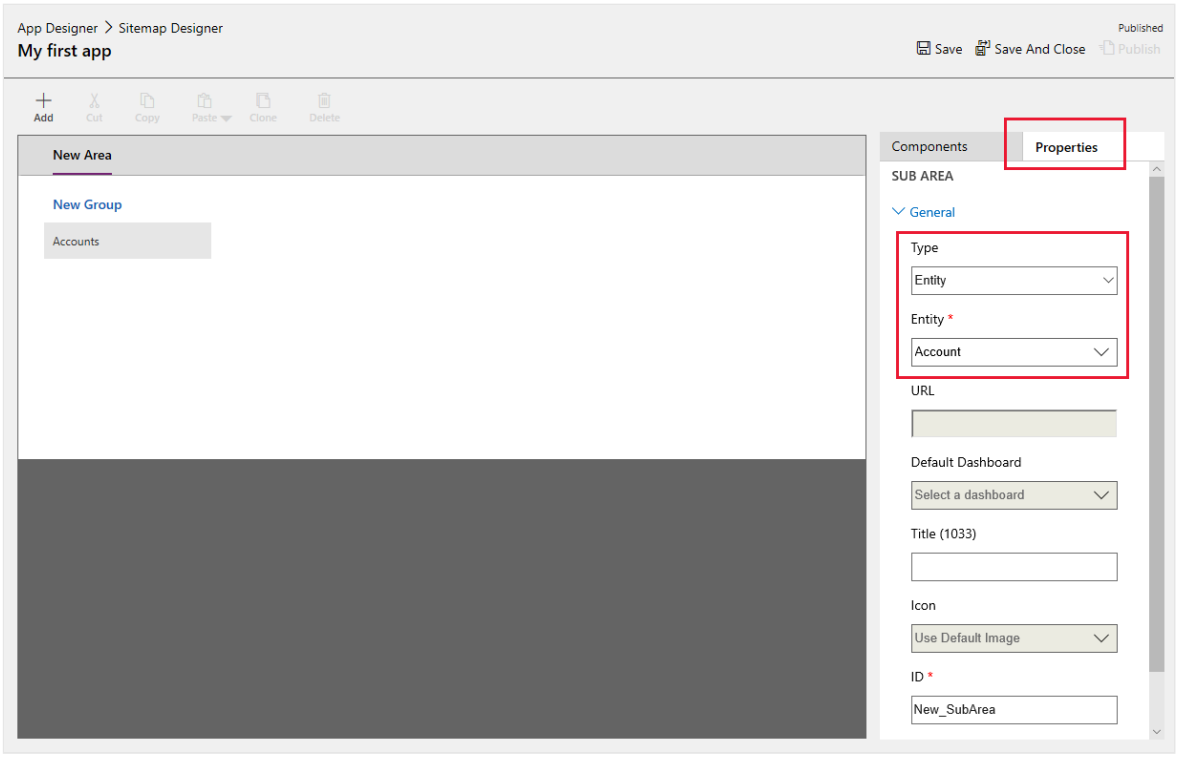
Notice that when an app is first created a site map must be configured for it.

2. On the sitemap designer, select **New Subarea**.



3. In the right pane, select the **Properties** tab, and then select the following properties.

- **Type**: **Entity**
- **Entity**: **Account**

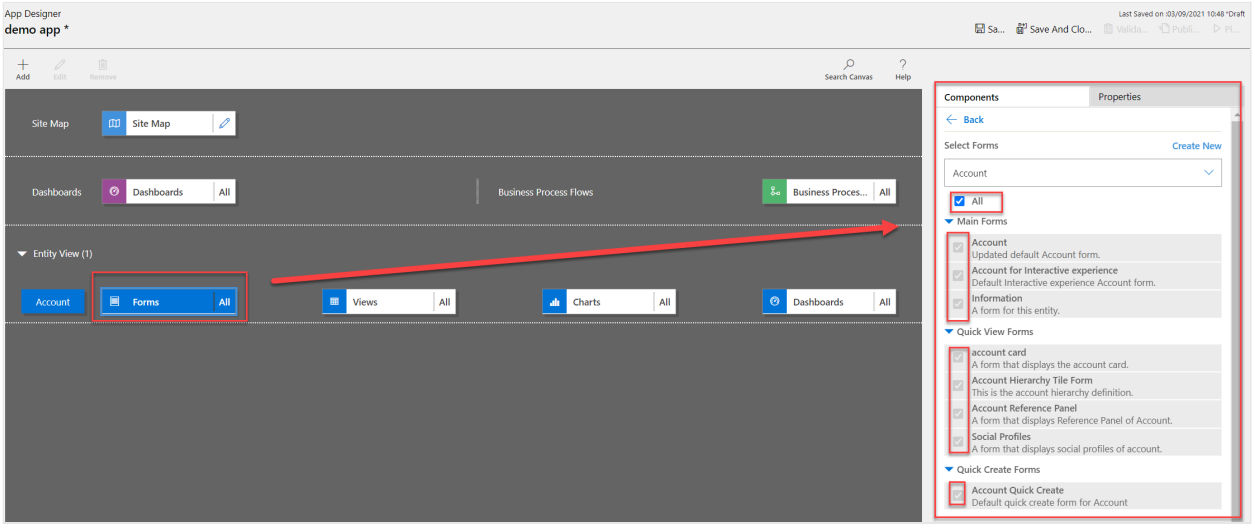


When the **Title** property is left blank, the app uses the table name in the app's left navigation pane. For this app, **Accounts** will be displayed in the app at runtime.

4. Select **Save And Close** to close the sitemap designer.

Finalizing the app

By default, all the account table's forms, views, charts, and dashboards are enabled for the app. From the app designer **Components** tab on the right pane, components can be cleared so that they aren't available in the app at runtime. It's also possible to create new components, such as a custom form. For this app, leave all components enabled.



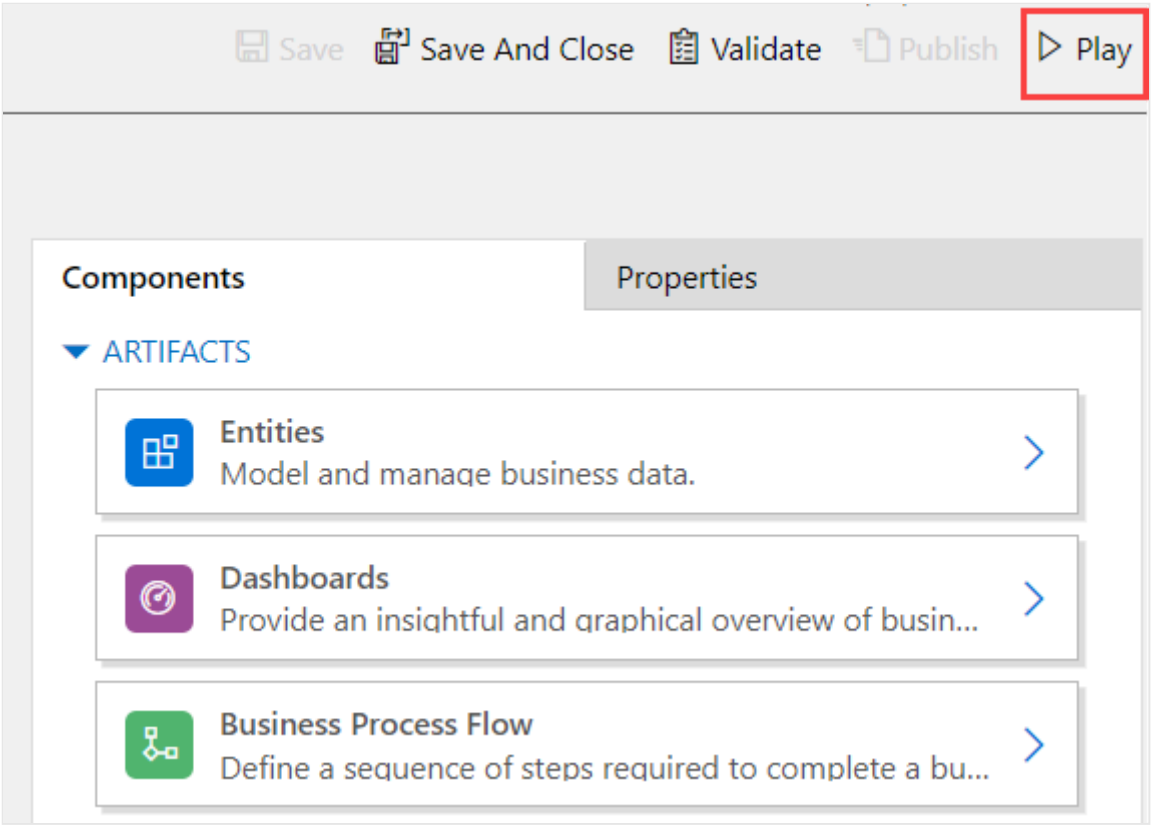
1. On the app designer toolbar, select **Save**.

2. After your app is saved, on the app designer toolbar, select **Publish** to make it available to run and share.

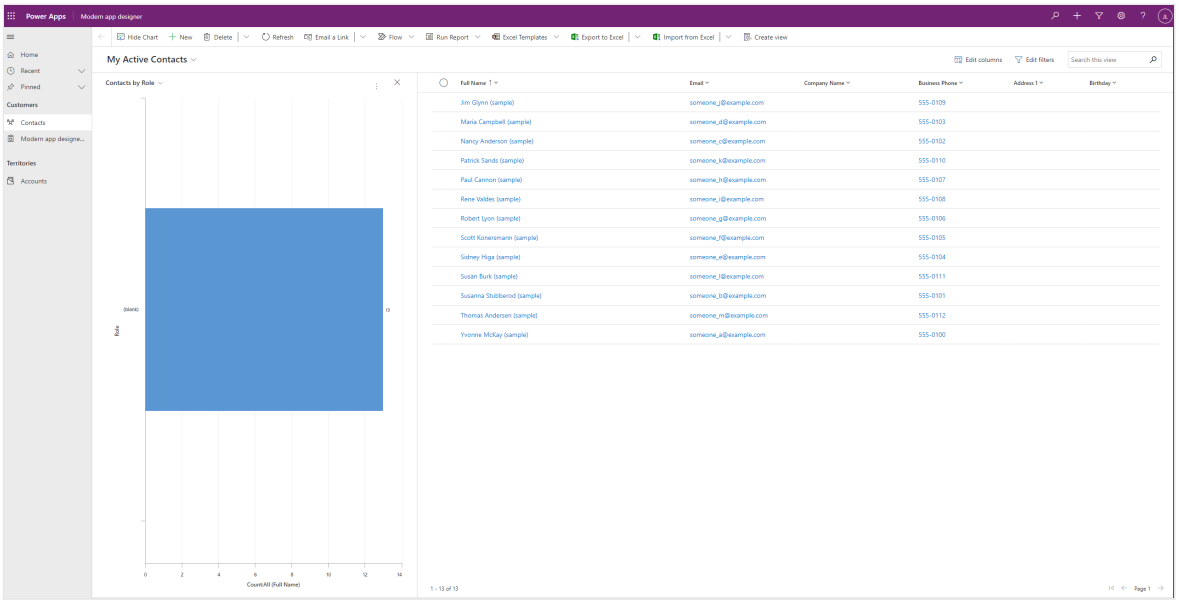
The rest of the design experience revolves around further developing the site map in addition to configuring the app through the app designer. More information: Add or edit app components

Play your app

On the app designer toolbar, select **Play**. This will only be available once the app is published.



- To create a record, select + **New**.
- To view a chart, on the app command bar select **Show Chart**.
- To change the view, select the **My Active Accounts** view, and then select the view required.



To learn more about how to use your app, see Basic navigation in a model-driven app.

Edit an app

1. Sign in to Power Apps.
2. Choose the environment by selecting the environment icon 
3. Select **Solutions**.
4. Choose the solution that contains the model-driven app where editing is required.
5. On the left navigation pane select **Apps**, select a model-driven app, and then on the toolbar select **Edit**.
6. As with creating an app editing experience will center around creating a strong site map
7. In the app designer, add or edit components to the app, as required. More information: Add or edit app components

Next steps

Share an app

Edit the site map

Add or edit app components

Feedback

Was this page helpful?

Yes No

Provide product feedback

Create a model-driven app site map using the legacy site map designer

Article • 01/23/2025

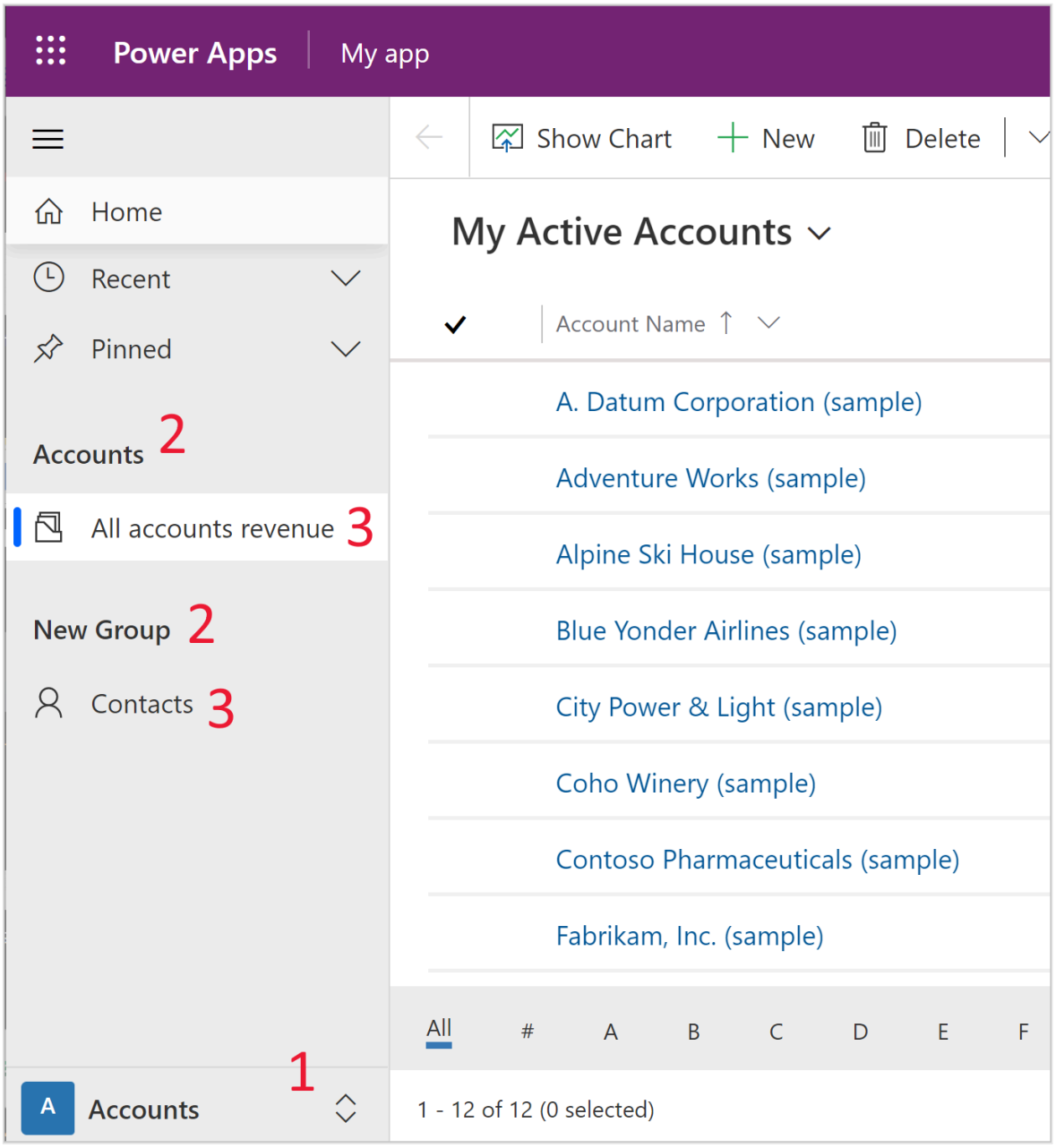
This article describes how to use the classic site map designer. For information about how to create site mapping in the modern app designer, go to App navigation in model-driven apps.

Site maps define the manner by which users move between tables in a model-driven app. This is called **navigation**. App navigation is a fundamental aspect of the user experience when performing tasks within the app.

To understand site maps it helps to introduce some concepts, so that you can understand the relationship between the site map designer and the way users intend to use the app.

The screenshot below illustrates a number of features.

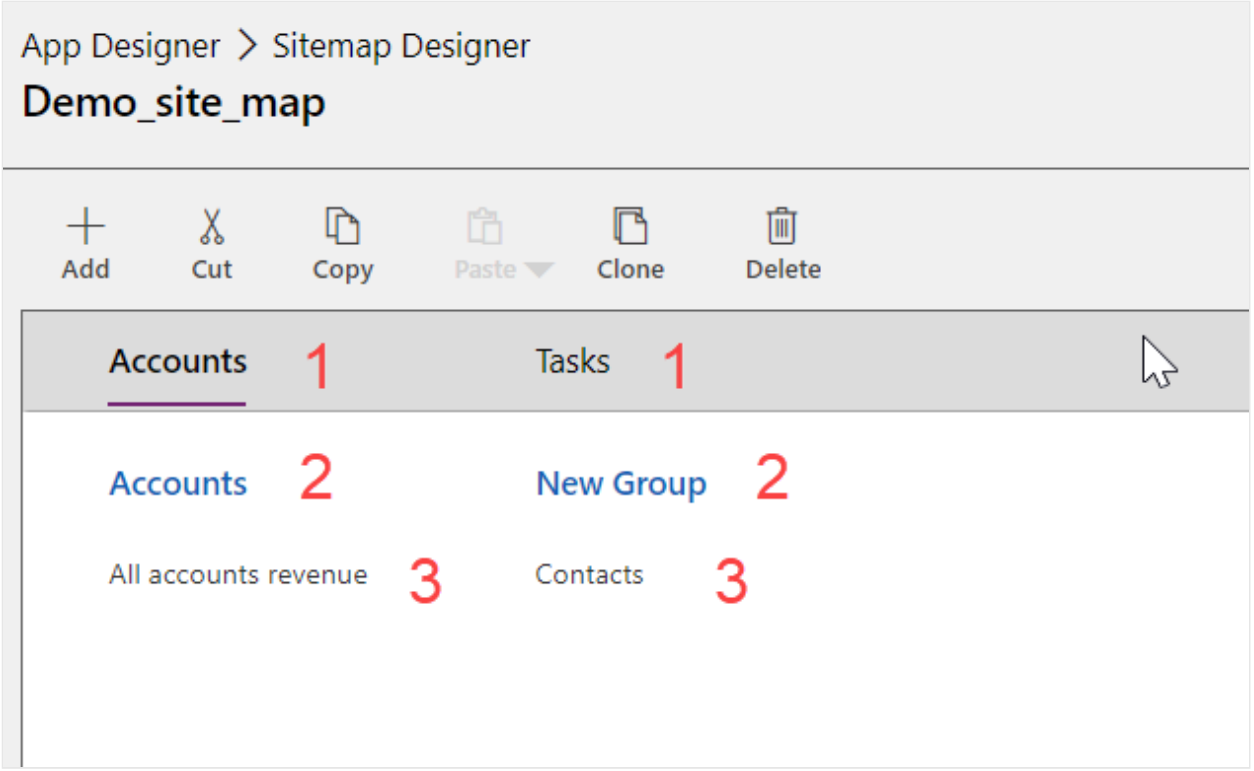
1. An **Area**. Model-driven apps can have multiple areas. Users toggle between these to access different groups.
2. A **Group**. Areas can have multiple groups. Essentially these allow you to group tables, custom pages, and other components, in a logical fashion.
3. A **Table (or entity)**. This allows users to see views of the tables that have been selected in the app designer.



This hierarchical structure is important to enable a good user experience, and the terminology used allows better understanding of the app designer.

Viewing an app from within the app designer

The same app viewed from within the site map designer shows the corresponding areas, groups, and tables. In this case you're viewing tables within the accounts **Group** and groups within the accounts **Area**. You can also notice there's another area called **Tasks** that also exists within the app designer.



Prerequisites for editing site maps

Users need to have the System Administrator or System Customizer security role or equivalent permissions. Specifically, any user with the following privileges can also create apps:

- Create, Read, and Write privileges for the App table.
- Read and Write privileges for the Customizations table.
- Read privileges for the Solution table.

These privileges can be viewed or set on the **Customization** tab of a security role.

How to build a site map

In this tutorial several site map tasks are carried out such as creating a new site map, and adding an area, group, and subarea.

The site map designer also lets an app designer define the area, subarea, or group titles in the languages supported by the environment.

A default site map is created automatically as part of the model-driven app creation process. This can be edited by using the site map designer.

Create a site map for an app

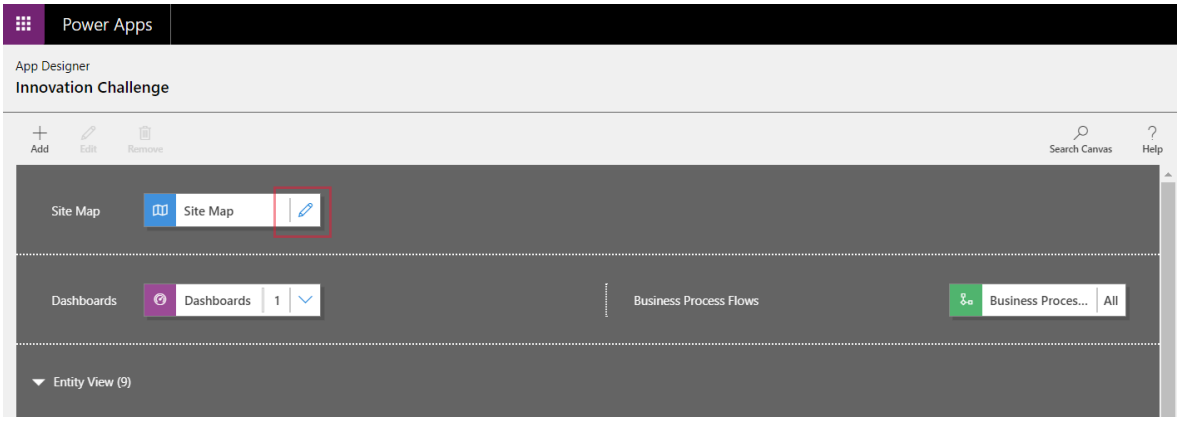
1. Sign in to Power Apps.

2. Select **Solutions**, open the solution you want.

3. Select **New** - **App** - **Model-driven app**.

4. From the two options available select **Classic App Designer**.

5. On the app designer canvas, in the **Site Map** area, select the pencil icon next to **Site Map** to **Open the Site Map Designer** .



The site map designer opens a canvas that is prepopulated with one area, one group, and one subarea. Select the area, group, or subarea tile to change its properties.

> **Note**
>
> Selecting **Open the Site Map Designer** from the app designer canvas automatically creates a new site map (if there's no existing site map), and gives the new site map the same name as the app name and the same unique name as the app unique name.

6. Add an area to the site map.

7. Add a group to the site map.

8. Add a subarea to a group in the site map.

9. Select **Save**.

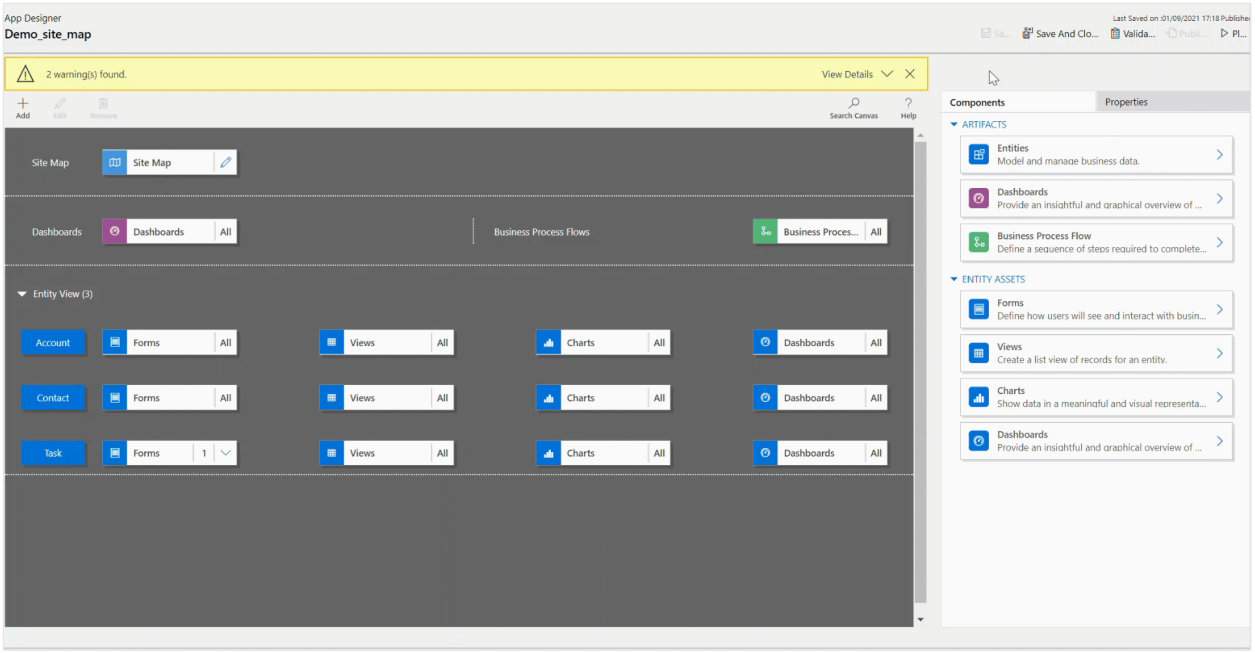
> **Note**
>
> The new site map is associated with the app when returning to the app designer and select **Save**. When a site map is configured, **Configured** appears

on the site map tile; otherwise **Not Configured** appears on the tile. If you open the site map designer from the app designer and configure a new site map, but close the browser before associating the site map with the app, the site map will be automatically associated with the app the next time the app designer is opened, based on the app unique name.

10. Select **Publish**.

Review the model-driven app

From that app designer select **Save** > **Validate** > **Play**. This runs the app with the latest changes and is an important part of the app building process.



Add an area to the site map

1. Select **Add** 

on the site map designer canvas, and then select **Area**.

   or

   From the **Components** tab, drag the **Area** tile to the empty box on the canvas. An empty box is visible you move the tile the correct place on the canvas.

2. Select the area that has been added. The **Properties** tab highlighted in the pane to the right of the canvas.

3. Add or edit the area properties.

Under **General**, do the following:

- **Title**: Enter the title for the area in the base language of the organization.
- **Icon**: A default application icon is selected. Select a different icon for the area from the list of web resources available in the solution. The icon only applies to the deprecated web client site map and not the modern model-driven app site map.
- **ID**: A unique ID is automatically generated, but different one can be entered if necessary. It's best practice to use the provided ID because if the ID entered isn't unique, users might get an error when they're using the app, or app designers might get an error when importing a solution that contains this site map.
- **Show Groups**: Select this check box to show groups of subareas in the navigation pane.

Under **Advanced**, do the following:

- **More Titles**: If your organization uses multiple languages, select a language (Locale) for the title, enter the title, and then select **Add**  . Titles can be created, edited, or deleted for as many languages as your organization uses. However, it's only possible to have one title per language.
- **More Description**: If an organization uses multiple languages, select a language for the description, enter the description, and then select **Add**  . Descriptions can be created, edited, or deleted for as many languages as your organization uses. However, it's only possible to have one description per language.
- **URL**: Enter the URL to render for the Dynamics 365 for Outlook folder that represents the area.

Creating and editing Groups, Subareas and Areas

The following sections provide instructions describing how to work with Groups, Subareas, and Areas in addition to reviewing their properties.

Add a group to the site map

1. On the site map designer canvas, select the area you want to add the group to.
2. Select **Add** 

, and then select **Group**.

   or

   From the **Components** tab, drag the **Group** tile to an empty box under the **Area** in the canvas. An empty box is visible when you move the tile to the correct place in the canvas.
3. Select the group just added.
4. On the **Properties** tab, add or edit the group properties:

   Under **General**, do the following:

   - **Title**: Enter the title for the group in the base language of the organization.
   - **ID**: A unique ID is automatically generated. Enter a different one if necessary. We recommend using the automatic ID because if the ID entered isn't unique, there might be an error when importing a solution containing this site map.

   Under **Advanced**, do the following:

   - **More Titles**: If your organization uses multiple languages, select a language (Locale) for the title, enter the title for the group, and then select **Add**  . Titles can be created, edited, or deleted for as many languages as your organization uses. However, it's only possible to have one title per language.
   - **More Descriptions**: If your organization uses multiple languages, select a language for the description, enter the description for the group, and then select **Add**  . Descriptions can be created, edited, or deleted for as many languages as your organization uses. However, it's only possible to have one description per language.
   - **URL**: Enter the URL to render for the Dynamics 365 for Outlook folder that represents the group.
   - **Set as Profile**: Select this check box to indicate whether this group represents a user-selectable profile for the workplace. The group set as a user-selectable profile is made available as options in your personal options. This only applies for groups within the **Workplace** area.

Add a subarea to a group in the site map

1. Select **Add**  on the site map designer canvas, and then select **Subarea**.

   or

   From the **Components** tab, drag the **Subarea** tile to an empty box under the **Group** section in the canvas. An empty box is visible the tile is moved to the correct place in the canvas.

2. Select the subarea that has been added.

3. On the **Properties** tab, add or edit the subarea properties:

   Under **General**, do the following:

   - **Type**: Select whether the subarea being adding is a dashboard, table, web resource, or URL.
   - **Entity**: Select the table that the subarea is for. This column is disabled if the subarea type is other than **Entity** in the **Type** drop-down list.
   - **URL**: Specify a URL to a web page. The URL text displays from this subarea and, when selected, a new browser window opens that displays the page. This column is disabled if **Entity** in the **Type** is selected in the drop-down list.

     > ⓘ **Important**
     >
     > Site map subarea URLs that link to an .aspx page aren't supported.

   - **Default Dashboard**: Select the default dashboard to be displayed for this subarea. This column is disabled if **Dashboard** hasn't been selected in the **Type** drop-down list.
   - **Title**: Enter the title for the subarea in the base language of the organization.
   - **Icon**: A default application icon is selected. Select a different icon for the subarea from the list of web resources available in the solution.
   - **ID**. A unique ID is automatically generated. Enter a different unique ID if necessary.
   - **Parameter Passing**. Select this check box to pass information about the organization and language context to the URL. This check box is checked only when the subarea type is a web resource or a URL-based subarea.

   Under **Advanced**, do the following:

- **Privileges**: This defines whether a subarea is displayed based on privileges available in any security roles that are assigned to the user. Select the name of the table to check privileges for, and then select the check boxes to assign privileges.
- **More Titles**: If your organization uses multiple languages, select a language for the title, enter the title for the subarea, and then select **Add**. Titles can be created, edited, or deleted for as many languages as your organization uses. However, it's only possible to have one title per language.
- **More Descriptions**: If your organization uses multiple languages, select a language for the description, enter the description for the subarea, and then select **Add**. Descriptions can be created, edited, or deleted for as many languages as your organization uses. However, it's only possible to have one description per language.
- **SKUs**: Select the versions of Dynamics 365 that display this subarea.
- **Client**: Select the type of client that displays this subarea.
- **Outlook Shortcut**: Select the icon to display in Dynamics 365 for Outlook.
- **Offline Availability**: Select this check box to make this subarea available to users when they're offline in Dynamics 365 for Outlook.

Organize areas, groups, and subareas

Organize areas, groups, and subareas by dragging them to new positions. A container box appears where you can drop the tiles. Here are some options:

- Move a subarea to a new position within the same group or a different group under the same area.
- Move a subarea to a new position within a group under a different area.
- Move a group to a new position within the same area.
- Move a group to a new position in a different area.
- Move an area to a new position.

Edit the default site map

Each environment comes with a default site map and this can be edited.

1. Open solution explorer.
2. In the solution window, under **Components**, select **Client Extensions**.
3. On the component toolbar, select **Add Existing** > **Site Map**.
4. In the list of solution components, select the site map named **Site Map**, and then select **OK**.
5. Double-click to select the site map added that has the display name **Site Map** and is in a **Managed** state. The site map can be selected, and then on the toolbar, select **Edit**.

   The site map opens in the site map designer.
6. Add an area to the site map.
7. Add a group to the site map.
8. Add a subarea to a group in the site map.
9. Organize areas, groups, and subareas.
10. Select **Save**.
11. Select **Publish**.
12. Select **Save and Close**.

Clone a component in a site map

To make a copy of an existing component, select the component, and then on the toolbar, select **Clone**. All details of the cloned component are same as the base component except the ID and title. The ID is generated randomly.

When an area is cloned, the cloned area is added to the right of the currently selected area. When a group is cloned, the cloned group is added to the right of the currently selected group. When a subarea is cloned, the cloned subarea is added below the currently selected subarea.

Delete an area, group, or subarea from a site map

To delete a site map component, select the component tile, and then on the toolbar, select **Delete**. When an area is deleted, all groups and subareas in the area are also deleted. Similarly, when a group is deleted, the group and subareas in it are deleted.

Clients supported

The following table explains the clients supported for different site maps.

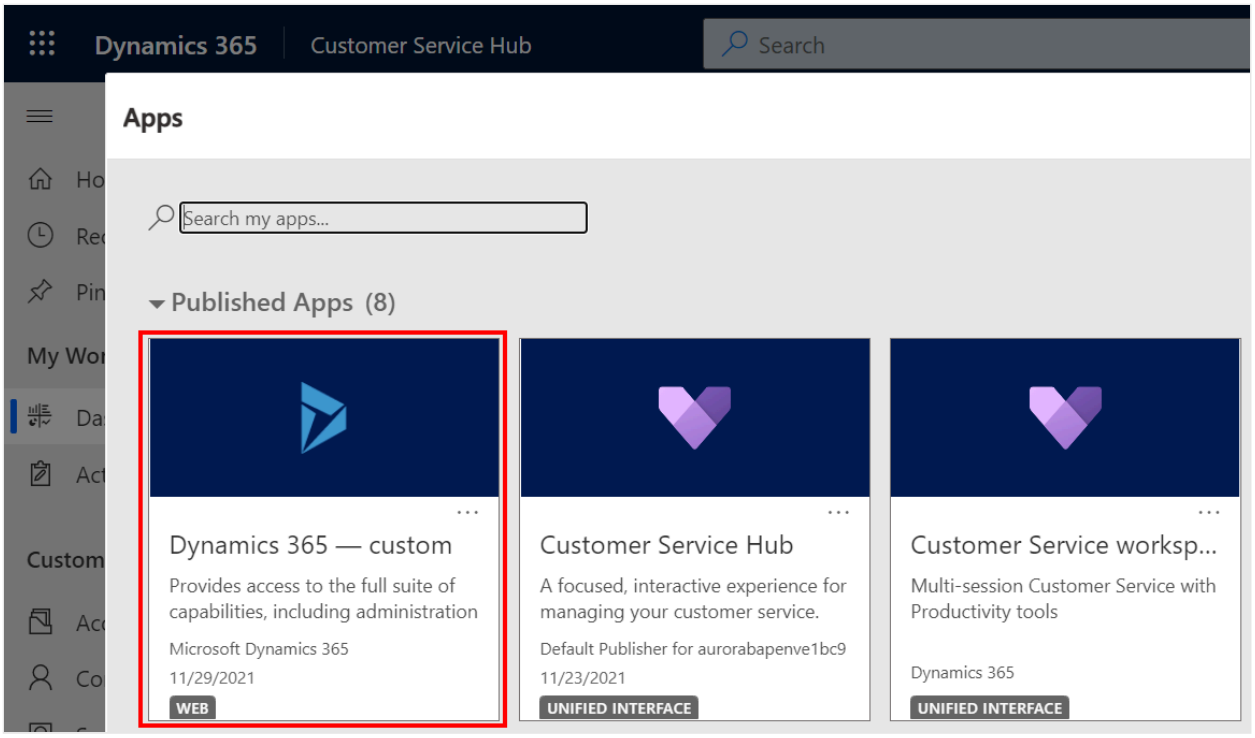
Expand table

Site maps	Supported clients
New apps	Unified Interface
Site map for the Dynamics 365 - custom app	Legacy web app and Dynamics 365 for Outlook
Model-driven apps (Sales, Sales Hub, Customer Service, Customer Service Hub, Field Service, Project Service Automation)	Legacy web app and Unified Interface

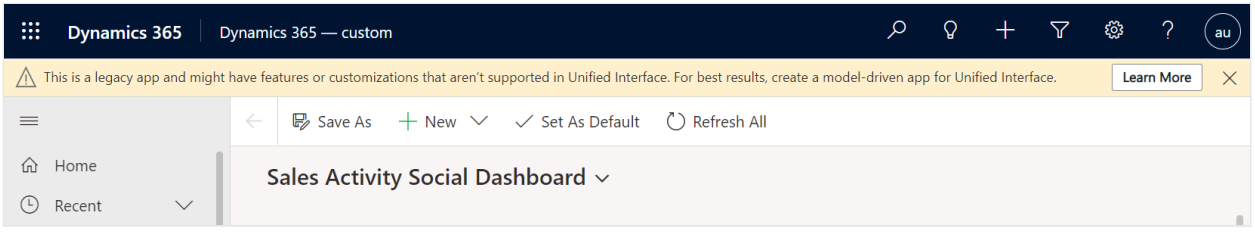
Replace the legacy app with a model-driven app

During the transition from web client to Unified Interface, a legacy app was provided which uses the web client site map. With the web client removed, we're deprecating the legacy app. Subsequently, makers might want to provide a replacement model-driven app for users. In a future release, the legacy app will be removed.

By default, the legacy app name is "Dynamics 365 - custom," but could have been renamed within the environment.



When the legacy app is run, a warning message is shown indicating that not all app features are supported.



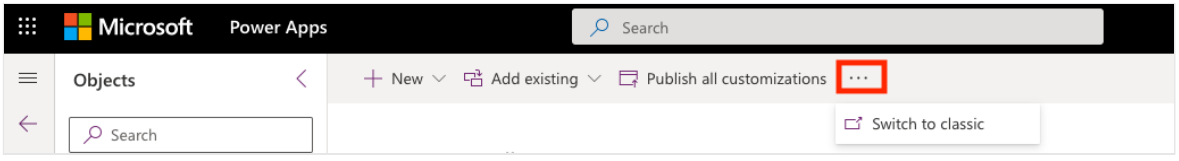
To create a model-driven app with the same site map items as the legacy app, the classic app designer can reuse the existing site map within a new app.

1. Open make.powerapps.com

2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. Open or create a new solution.

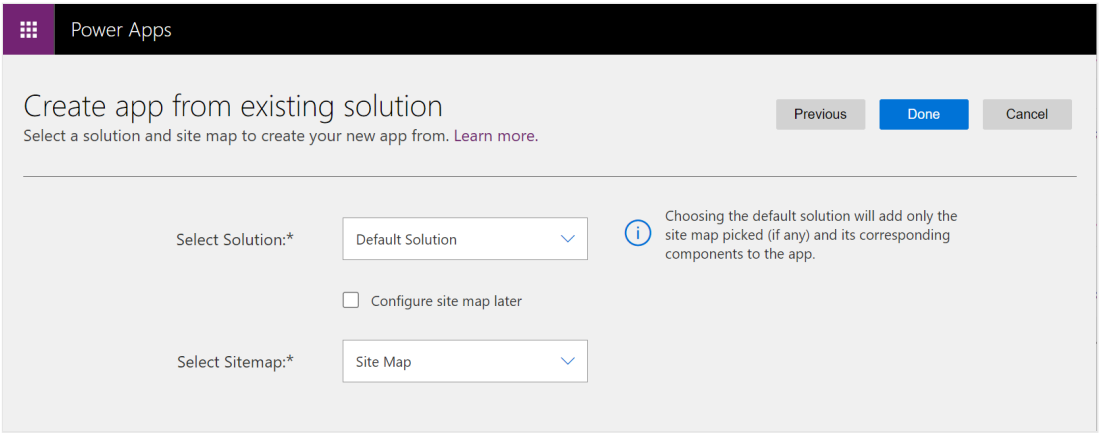
4. Select **...** on the top navigation bar, and then select **Switch to classic**, which opens the classic solution explorer.



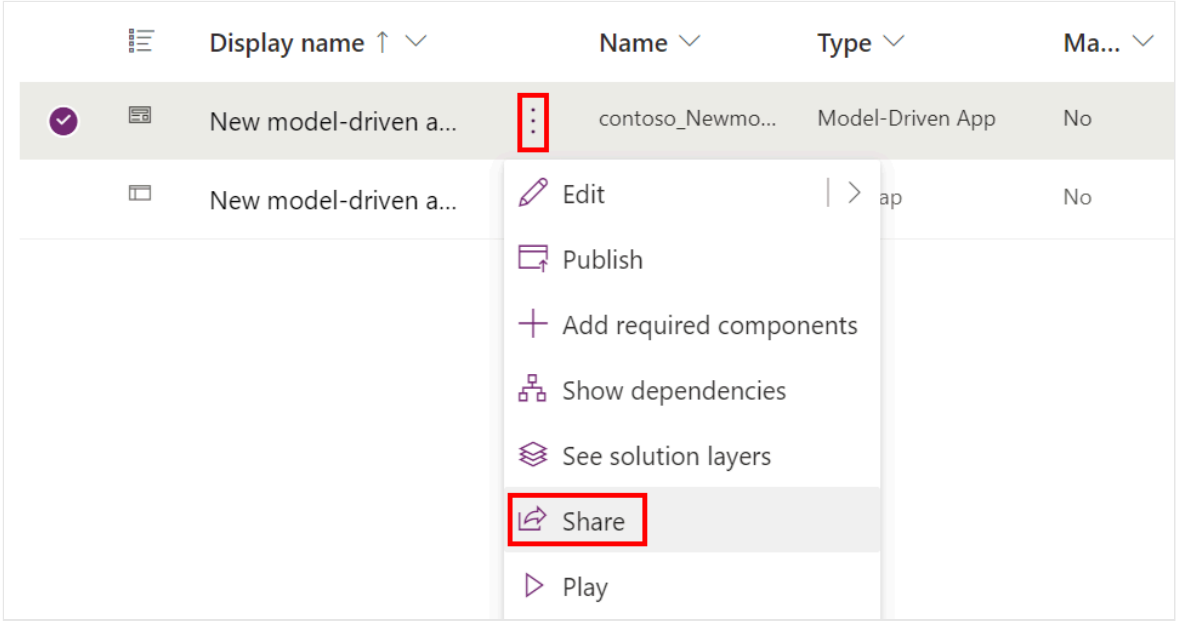
5. From the left pane, select **Model-driven Apps** > **New** .

6. In the **Create a New App** dialog:

- Enter the app name. The unique name and description can also be provided but aren't required.
- Select **Use existing solution to create the App**, and then select **Next**.
- Under **Select Solution** select **Default Solution**.
- Under **Select Sitemap** select **Site Map**, and then select **Done**.



7. Select **Publish**.
8. Close the classic app designer and open the solution containing the new model-driven app.
9. Select the new model-driven app, and then select **Share**.



10. Add people and assign security roles to allow them to use the app.

After the new model-driven app is available, the "Dynamics 365 - custom" app should be hidden for nonadmins in all environments.

1. Open Power Platform admin center and select the environment to hide the legacy app.
2. Select **Settings** > **Product** > **Behavior**.
3. Set **Show legacy app to everyone, not just admins** to **Off**.
4. Scroll to the bottom of the page and select **Save**.

Next steps

Create or edit an app Add or edit app components

Feedback

Was this page helpful? Yes No

Provide product feedback

Add or edit model-driven app components in the Power Apps app designer

Article • 10/17/2023

> **Important**
>
> Starting in October 2023, the classic app, form, and view designers are deprecated and all model-driven apps, forms, and views will only open in the modern designers. By default, the **Switch to classic** command to revert back to the classic designer from the modern designer will no longer be available. More information: **Classic app, form, and view designers are deprecated**
>
> We recommend that you transition to use the Modern Designers to create and edit your model-driven apps and components.

This tutorial describes how to add components to and remove components from a model-driven app.

A model-driven app is made up of various components. Two types of components can be added to an app:

- Artifacts
- Table assets

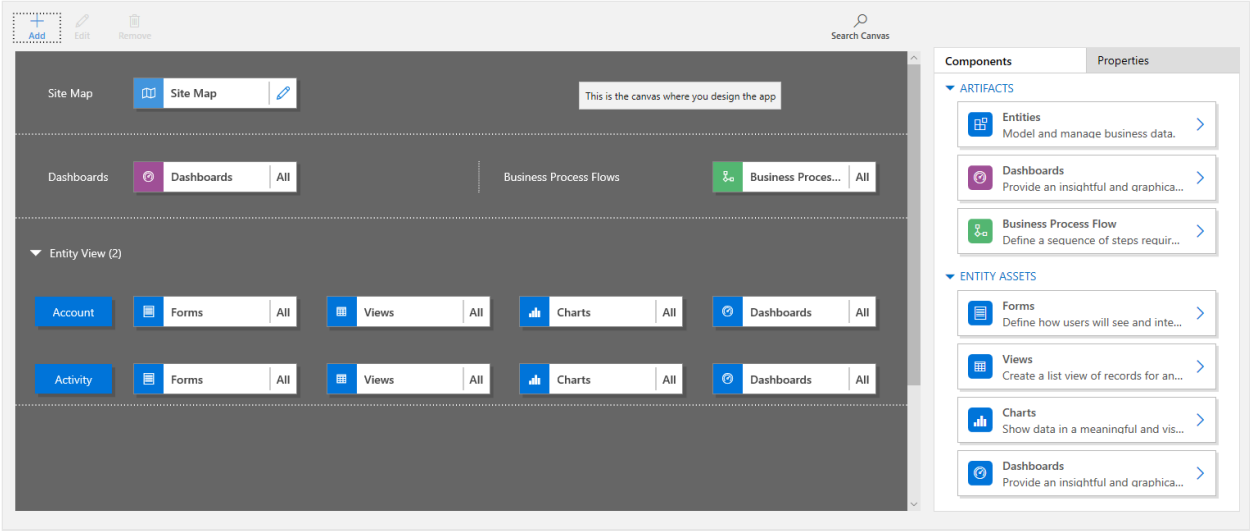
Tables, dashboards and business process flows are all artifacts of an app.

Table assets consist of forms, views, charts, and dashboards related to a specific table included in an app.

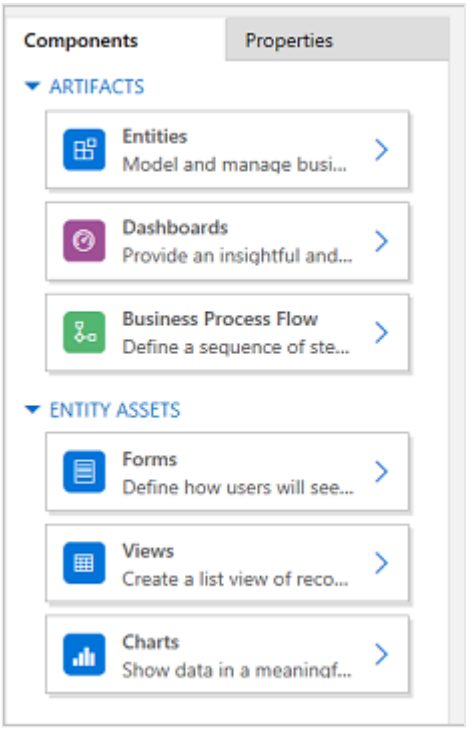
> **Note**
>
> Tables used to be called entities. The classic app designer uses the term entity in most places. For example, *Entity assets* rather than *Table assets*.

App designer layout

The app designer has two main areas. On the left side is the canvas, where app components can be added.



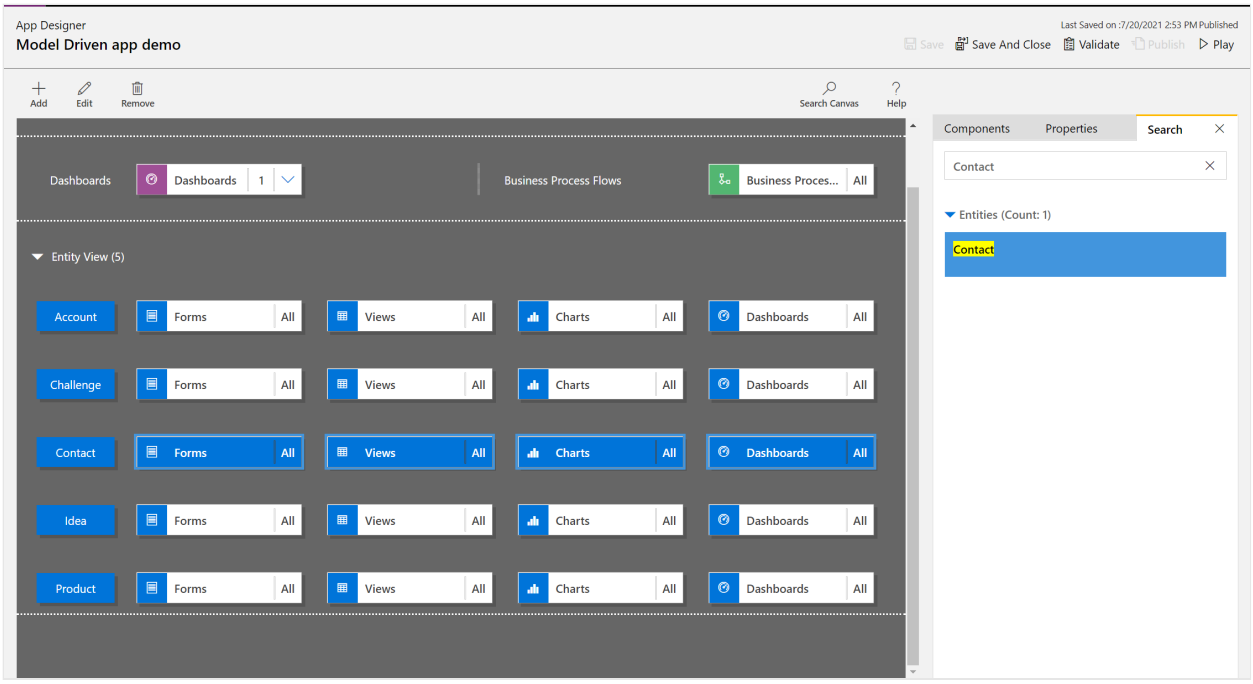
On the right side are tabs that are used to select components and set component properties.



The canvas includes areas for the **Site Map**, **Business Process Flow**, **Dashboards**, and **Tables**.

After the tables are in place, select each table and add the required table assets such as forms, views, and charts to it.

Use **Search Canvas** to search for components on the canvas. When **Search Canvas** is selected, a new search tab opens to the right of the tabs in the right pane.



Open an app

1. Sign in to Power Apps and optionally select an environment to work in.
2. Create a new model-driven app or select an existing one. For information about how to create an app, see Create or edit a model-driven app by using the app designer.

Add or edit a site map

You must define a site map before app components can be added to a newly created app. More information: Create a model-driven app site map using the site map designer.

Add a component

When a dashboard or a business process flow is added to an app, the related tables are automatically added to the app.

When a table is added to an app, the tiles for its assets are automatically added. There are two ways components can be added to the designer canvas:

- Use the **Add** button  on the command bar.
- Use the tiles on the **Components** tab.

Add a dashboard

To add a dashboard to an app, follow these steps:

1. On the app designer canvas, select the **Dashboards** tile.

   On the app designer canvas, the right pane shows dashboards that are available in the default solution.

2. In the **search** box, type keywords for the required dashboard.

   The dashboard list will be filtered to show results that match the keywords.

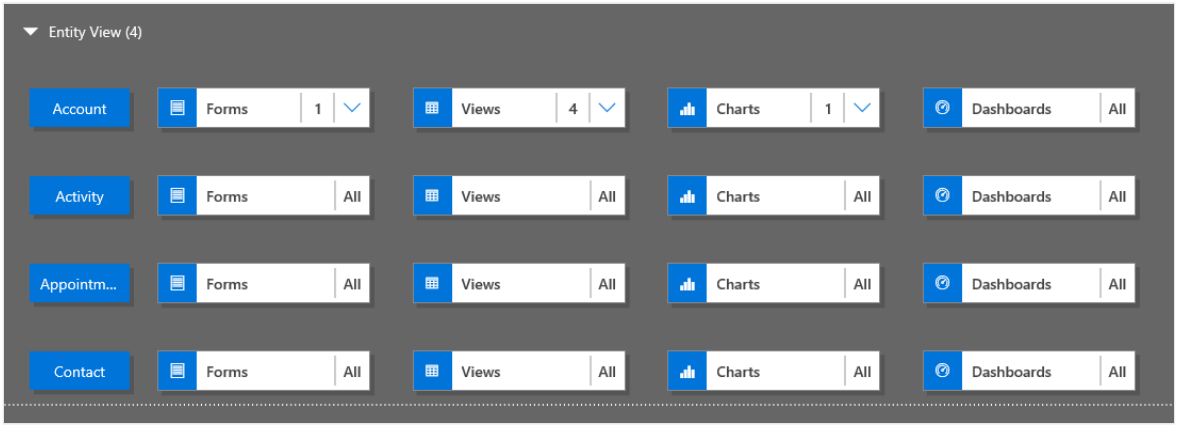
3. If users need to use only selected dashboard then select the check box for the component that needs to be added. The following types of dashboards are available:

   - **Classic Dashboards** appear on both the web app and the Unified Interface app.
   - **Interactive Dashboards** appear only on the Unified Interface app. If you have selected the client type for the app as web app, then the **Interactive Dashboards** option won't be displayed.

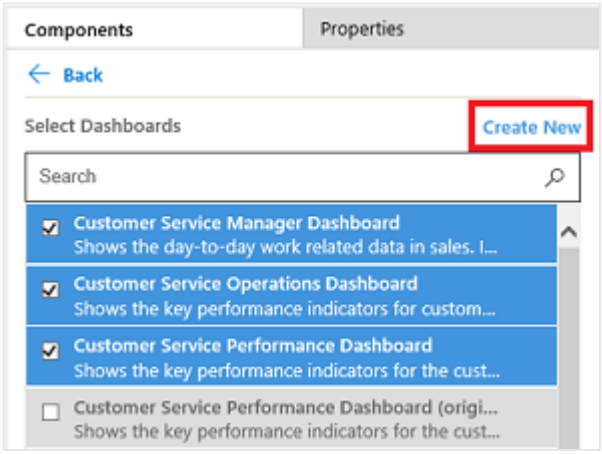
   Those dashboards will be added to the **Dashboard** tile on the app designer canvas. The **Dashboard** tile also shows a count of the number of dashboards added to the app. If no dashboard is selected, then **All** will appear instead of the dashboard count, and all dashboards will be available to users when they use the app.

   All tables the dashboard uses are also added to the **Entity View** area. For example, when the Customer Service Manager dashboard is added, the **Case**, **Entitlement**, and **Queue Item** tables are added to the **Entity View** area.

   For each table, tiles for its assets are also added. These tiles can be used to add forms, views, and charts. For more information, see: Add or edit app components in the Power Apps app designer



4. If the dashboard you want doesn't exist in the default solution, then create a dashboard by selecting **Create New** on the **Components** tab to the right of the canvas.



   The dashboard designer opens. For more information, see: Create and edit dashboards

5. Complete the above steps by selecting **Save**.

Add a business process flow

To add a business process flow to an app, follow these steps:

1. On the app designer canvas, select the **Business Process Flows** tile.

   On the app designer canvas, the right pane shows dashboards that are available in the default solution.

2. In the **Search** box, type one or more keywords that match the business process flow name you want.

   The business process flow list is filtered to show results that match keywords.

3. If the dashboard you want doesn't exist in the default solution, then create a dashboard by selecting **Create New** on the **Components** tab to the right of the canvas.

   > **Note**
   >
   > When adding a business process flow, the **Create New** option opens the **Create Process** dialog. To learn more about creating business process flows, go to **Create a business process flow**

4. Complete the above steps by selecting **Save**.

Add a table

To add a table to an app, follow these steps:

1. On the app designer canvas, select the **Tables** tile.

   On the app designer canvas, the right pane shows dashboards that are available in the default solution.

2. In the **Search** box, type one or more keywords matching the table name you want.

   The table list is filtered to show results that match the keywords.

   > **Note**
   >
   > When adding a table the **Create New** option opens the **New Table** dialog. To learn more about creating tables go to **Create a custom table**.

3. Complete the above steps by selecting **Save**.

Add table assets

With the components in place, table assets like forms, views, charts, and dashboards can be added to the app.

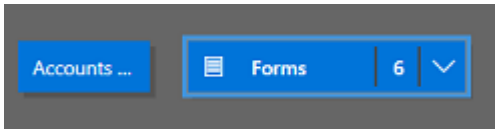
This section describes the steps for adding a form to the app. Use the same steps to add a view or chart to the app.

1. On the app designer canvas, select the **Forms** tile for the related table to add a form to.

   On the app designer canvas, the entire row for the table is selected. On the right side, all existing forms for the selected table will appear.

2. Select the check boxes for the forms that need to be available to users. The forms define how users will see and interact with data in the app.

   The form tile of the selected table will display the number of forms added.



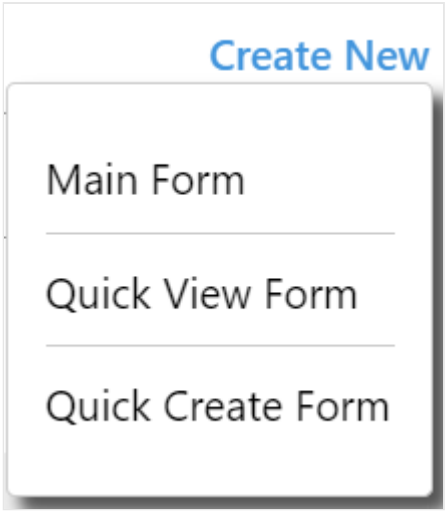
If none of the forms for a table are selected, then all the forms for that table will be displayed to the users while they use the app. This behavior is similar for views and charts also, if no view or chart is selected. This helps to create apps quickly when all available components are required. There's no need to select each component during app design.

If no dashboards or business process flows are selected, then all the dashboards and business process flows will be available for users while they use the app.

> **Note**
>
> For the app to run, each table that is added must have at least one active form. If multiple forms are selected then the first active form that appears in the default solution will be used when users run the app.

3. To add a new form that isn't available in the list, select **Create New**.

   In the drop-down list, select the type of form that is required.



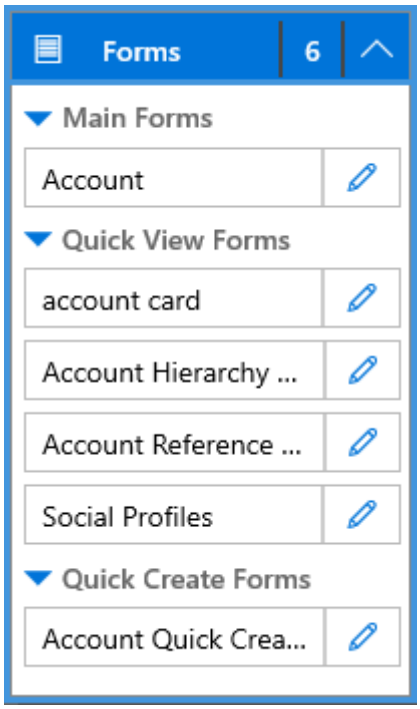
> **Note**
>
> The drop-down list is available only when adding forms. It isn't available for views and charts.

The form designer opens. More information: Create and design forms

When you add a view or a chart, the **Create New** option opens the corresponding designer. For more information go to: Understand views and Create or edit a system chart

> **Note**
>
> When you add a view, public views can be selected that are listed under the **Views** node in the solution explorer.

4. Select the down arrow to expand the tile and see a list of forms that have been added.



5. Repeat the previous steps to add table views and charts to the app.

6. Select **Save**.

Edit or remove components

- To edit a dashboard or a business process flow, select the down arrow to expand the tile, and then select the site map designer button corresponding to the dashboard or business process flow that needs to be updated.

  The designer for the selected artifact opens.

- To remove a dashboard or a business process flow, select the down arrow to expand the tile, and then select the dashboard or business process flow that needs to be removed. On the command bar, select **Remove**.

  Another way to remove a dashboard or business process flow is by clearing the corresponding check box on the **Components** tab.

- To edit or remove a table, select the table tile, and then on the command bar, select **Edit** or **Remove**. When editing a table, the solution explorer opens, where changes to the table can be made.

  Another way to remove a component is to select the dashboard, business process flow, or table tile. On the **Components** tab, clear the check boxes for the components that need to be removed from the designer.

  > **Note**
  >
  > When changes to a table are made, like changing a table display name or description, the changes do not appear in the app designer until the changes are published in the solution explorer.

Edit or remove table assets

Edit table assets

1. Select the down arrow  to expand the tile for forms, views, charts, or dashboards.
2. Select the form, view, chart, or dashboard that needs to be updated.
3. On the command bar, select **Edit**.

   or

   Select the site map designer button  corresponding to the form, view, chart, or dashboard.

Remove table assets

1. Select the down arrow  to expand the tile for forms, views, charts, or dashboards.
2. To edit an asset, select the relevant form, view, chart, or dashboard.
3. On the command bar, select **Remove**.

Alternatively, select the forms, views, charts, or dashboards tile and then on the **Components** tab, clear the check boxes for the assets that need to be removed.

Next steps

Create a site map for an app

Validate and publish an app

Shortcuts and accessibility in app designer and site map designer

Summarize this article for me

This article describes shortcuts and accessibility in the classic app and site map designers.

> **Important**
>
> Starting in October 2023, the classic app, form, and view designers are deprecated and all model-driven apps, forms, and views will only open in the modern designers. By default, the **Switch to classic** command to revert back to the classic designer from the modern designer will no longer be available. More information: Classic app, form, and view designers are deprecated
>
> We recommend that you transition to use the modern designers to create and edit your model-driven apps and components.

Microsoft is committed to making its products and services easier for everyone. For more information, go to, Microsoft accessibility

This article describes the accessibility features for app makers available with model-driven-apps.

Learn more about shortcuts for app users.

Keyboard shortcuts

Many users find that using an external keyboard with keyboard shortcuts for Power Apps helps them work more efficiently.

For users with mobility or vision disabilities, keyboard shortcuts can be easier than using the touchscreen and are an essential alternative to using a mouse.

This article itemizes the keyboard shortcuts for Power Apps.

App designer and site map designer

Expand table

To	Shortcut Keys (Windows)	Shortcut Keys (Mac)
Go to the next section or major component.	Ctrl + F6	Cmd + F6
Go to the previous section or major component.	Shift + Ctrl + F6	Shift + Cmd + F6
Save command.	Ctrl + S	Cmd + S
Save and Close command.	Ctrl + Alt + S	Cmd + Opt + S
Validate command.	Ctrl + Alt + V	Cmd + Op t+ V
Publish command.	Ctrl + Alt + P	Cmd + Opt + P
Close command.	Ctrl + Q	Cmd + Q
Add.	Shift + N	Shift +
Edit.	Shift + E	Shift + E
Remove.	Shift + R	Shift + R
Search Canvas.	Ctrl + Shift + F	Cmd + Shift + F
Open **Components** tab.	Alt + Shift + C	Opt + Shift + C
Open **Properties** tab.	Alt + Shift + P	Opt + Shift + P
Open the **Required** tab.	Alt + Shift + R	Opt + Shift + R
Add Menu.	Ctrl +Alt +N	Cmd + Opt +N
Clone an app designer component.	Ctrl + C	Cmd + C
Delete.	Delete key	Delete key
Copy a site map component to the clipboard.	Ctrl + C	Cmd + C
Cut.	Ctrl + X	Cmd + X
Paste the clipboard contents to the site map.	Ctrl + V	Cmd + V
Expand and move the focus to the notification area in App Designer.	Alt + Shift + N	Opt + Shift + N
Collapse the expanded notification area in App Designer.	Esc	esc

Keyboard navigation

Moving around the app designer and site map designer by using the keyboard is also supported. This enables people who don't use a mouse or touchscreen to get around and complete actions.

Expand table

To	Press
Go to the next tab stop in the defined tab order (which is left to right for LTR languages and right to left for RTL languages).	Tab
Move to the previous tab stop in the tab order.	Shift + Tab
Move to the next item in a list or a menu, or user interface feature that has multiple options. -OR- Move vertically to the next tile.	Down Arrow keys
Move to the next item in a list or a menu, or user interface feature that has multiple options. -OR- Move vertically to the previous tile.	Up Arrow
Move horizontally to the next tile.	Right Arrow
Move horizontally to the previous tile.	Left Arrow
Expand a collapsible control.	Right Arrow or Enter or Spacebar
Collapse an expanded collapsible control.	Left Arrow or Enter or Spacebar
Perform an action such as selecting an option from radio buttons, selecting a link, or selecting a check box.	Spacebar
Activate a command bar action, or confirm a component add.	Enter
Close an alert, dialog box, or menu, or cancel.	Esc

Screen readers

People with disabilities may rely on the use of assistive technology (AT) such as screen readers or a variety of alternative input devices to interact with the app designer and site map

designer.

These popular screen readers running on the web browsers indicated here work with app designer and site map designer:

Expand table

Browser	Jaws	Narrator	VoiceOver
Microsoft Edge	Yes	Yes	Yes
Google Chrome	Yes	No	No
Firefox	Yes	No	No
Safari	No	No	Yes

Accessibility info for browsers

For accessibility information about browsers, visit the following websites:

- Windows accessibility features
- Accessibility features in Microsoft Edge
- Firefox accessibility features
- Safari accessibility features
- Google Chrome accessibility technical documentation

Related articles

Use keyboard shortcuts for app users in Power Apps

Last updated on 02/19/2026

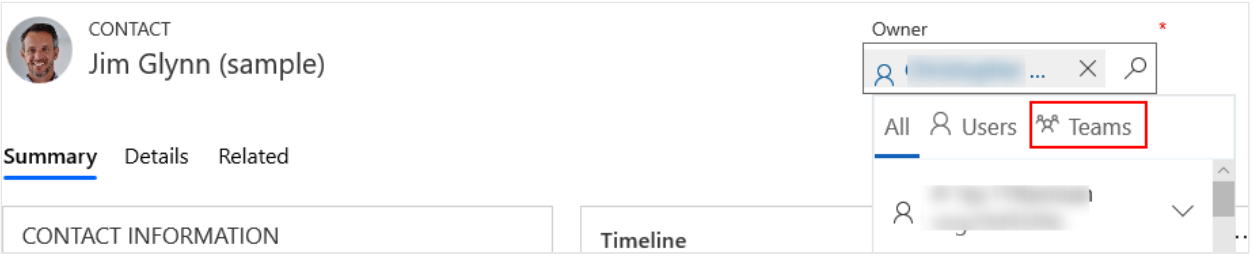
Add a table as a lookup option in your app

Article • 03/04/2025

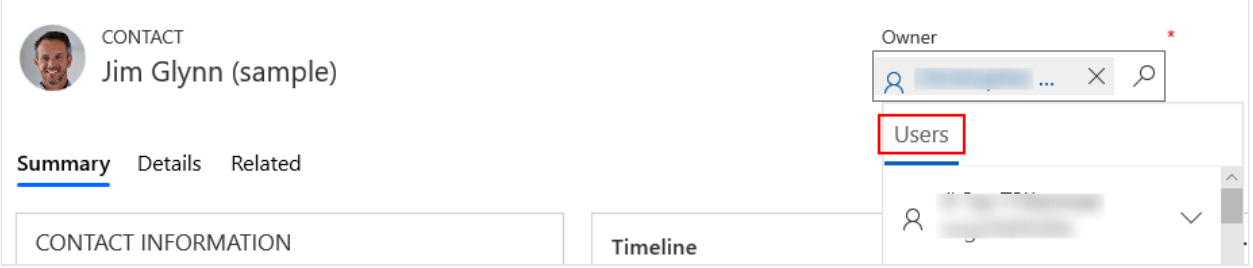
This article describes a feature that uses the classic app designer.

> **ⓘ Important**
>
> Starting in October 2023, the classic app, form, and view designers are deprecated and all model-driven apps, forms, and views only open in the modern designers. By default, the **Switch to classic** command to revert back to the classic designer from the modern designer is no longer available. More information: [Classic app, form, and view designers are deprecated]
>
> We recommend that you transition to use the modern designers to create and edit your model-driven apps and components.

With model-driven apps, for a table to be available in a lookup it must be added to the app. For example, contact records have the ability to be assigned to a user or a team. Both of these tables have a relationship with the contacts table.



However, if the **User** table is included in the app but the **Team** table isn't, only user rows appear in a lookup.



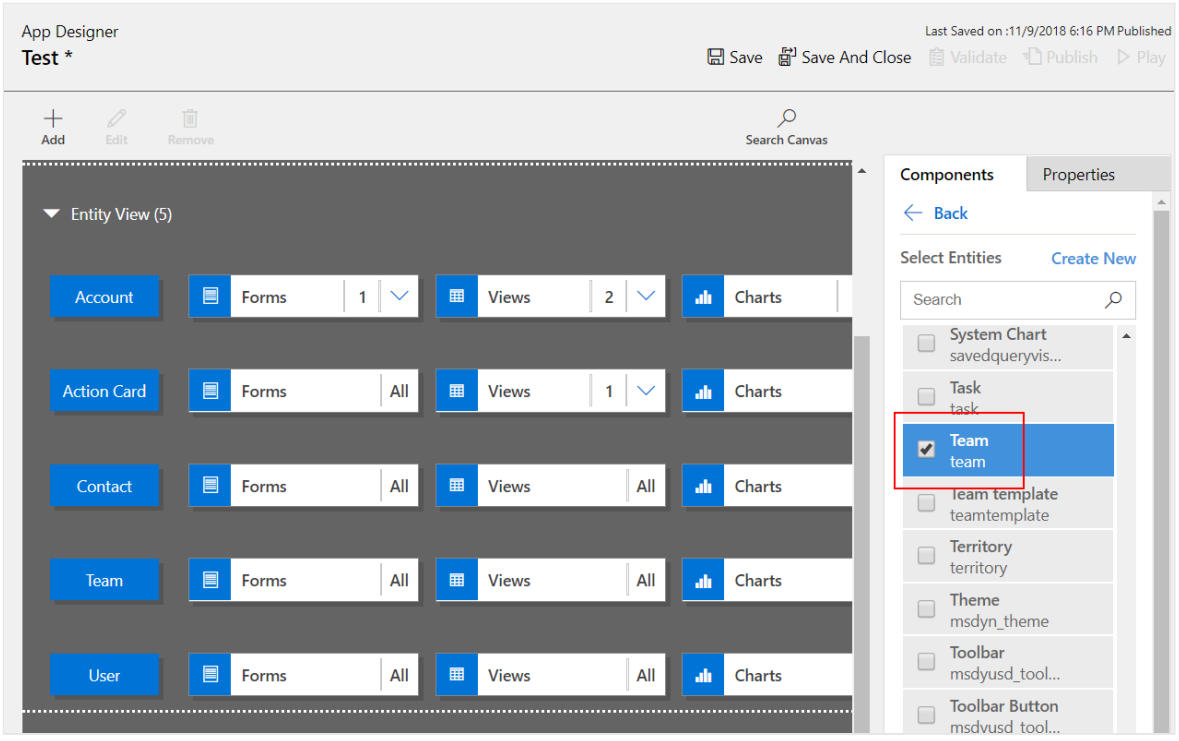
This can be resolved by adding the **Team** table to the app using the app designer.

Add a related table to an app to enable the lookup

1. Go to Power Apps, and then select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Open the solution you want, and then select the model-driven app. This opens the app in the app designer.

3. Select the **Components** tab, select **Entities** (tables), and then select **Team**.



4. Select **Save**, and then select **Publish** to make the change available to app users within the organization.

Next steps

Share a model driven app

Feedback

Was this page helpful? Yes No

Provide product feedback

Classic form editor overview

Article • 10/21/2023

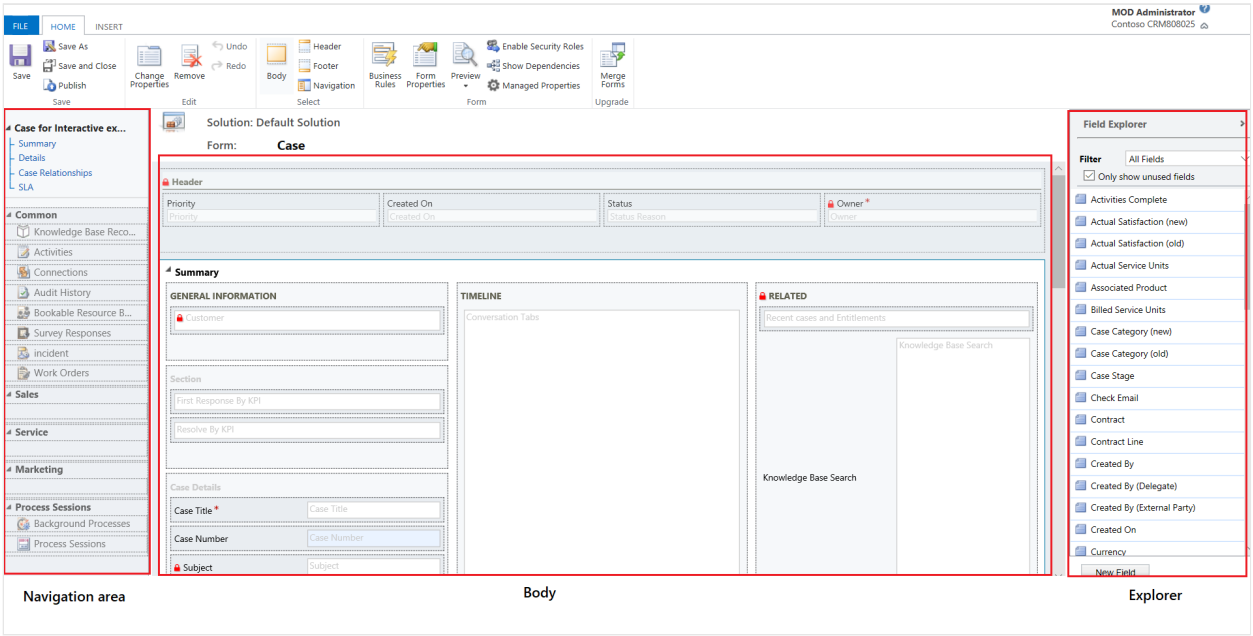
This article describes the classic form editor.

> **ⓘ Important**
>
> Starting in October 2023, the classic app, form, and view designers are deprecated and all model-driven apps, forms, and views will only open in the modern designers. By default, the **Switch to classic** command to revert back to the classic designer from the modern designer will no longer be available. More information: **Classic app, form, and view designers are deprecated**
>
> We recommend that you transition to using the modern form designer to create and edit your model-driven app forms. More information: **Overview of the model-driven form designer**

The form editor displays commands in three tabs: **File**, **Home**, and **Insert**.

- File tab
- Home tab
- Insert tab

The form editor is divided into three areas: **Navigation**, **Body**, and **Explorer**.



Navigation

Located on the left side, use the navigation area to control access to related tables or to

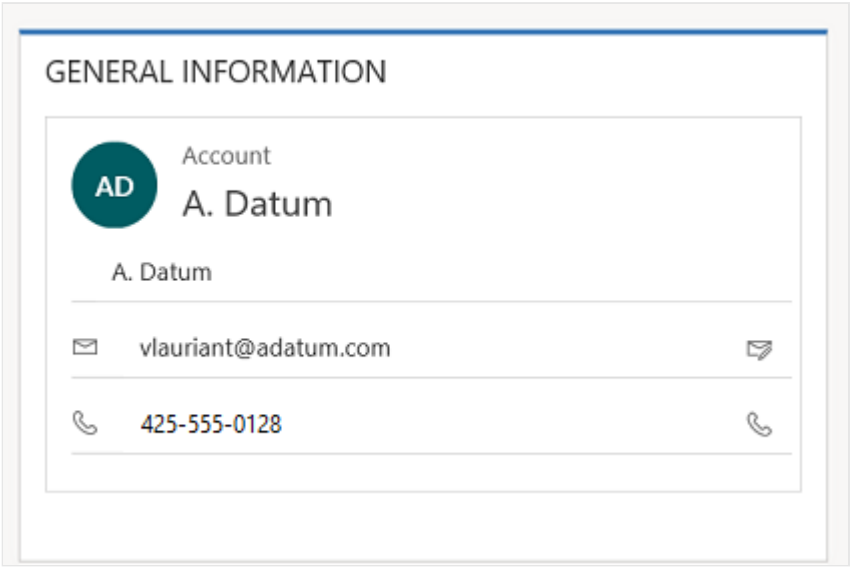
add links to URLs to be displayed in the main pane of the form. To edit navigation, you must first select the **Navigation** command in the **Select** group of the **Home** tab.

Main forms provide navigation options through the navigation bar, but use the same data in the navigation area to control what navigation options are available. More information: Edit Navigation

Body
Located in the center, use the body area to control the layout of the form. You can select and drag form elements to position them. Double-clicking on an element will open the properties for the element.

By default, for the Case, Contact, and Account Main forms, the first section under the **Summary** tab shows the account or contact card form of type **Quick View**. For custom tables, this section isn't available by default. You can insert a new section and a quick view form in it. The card form shows a maximum of five columns. Other than columns, it isn’t possible to show other controls in the Blue tile even if the quick view form contains it.

> **Note**
>
> To preserve the card format (as shown in the following image), we recommend that you do not move the quick view form to any other section on the form.



More information: Create and edit quick view forms

- To add a column, select it from the **Column Explorer** and drag it into a section.
    - To add an element that isn't a column, select where you want to place it and use the appropriate command from the **Insert** tab add it.

- To remove an element, select it and use the **Remove** command in the **Edit** group of the **Home** tab.
- To edit the **Header** or **Footer** for the form you must first select the corresponding command in the **Select** group of the **Home** tab.

Explorer
Located on the right side, the content of the explorer area depends on the context.

When you select **Body**, **Header**, or **Footer** in the **Select** group of the **Home** tab, you'll see the **Column Explorer**. Use the **Column Explorer** to drag columns you want to display into a section in the form or within the header or footer. You can include the same column multiple times in a form. Use the **New Column** button as a shortcut to create a new column.

When you select Navigation in the Select group of the Home tab, you'll see the **Relationship Explorer**. Drag any of the relationships into one of the groups within the navigation area. You can't add the same relationship twice. Relationships are available based on how they're configured. If you configure a relationship to not display, it won't display in the **Relationship Explorer**. For information about how to configure default display options for relationships, see Navigation pane item for primary table.

You can use the **New 1:N** and **New N:N buttons** as a shortcut to add new table relationships.

File tab

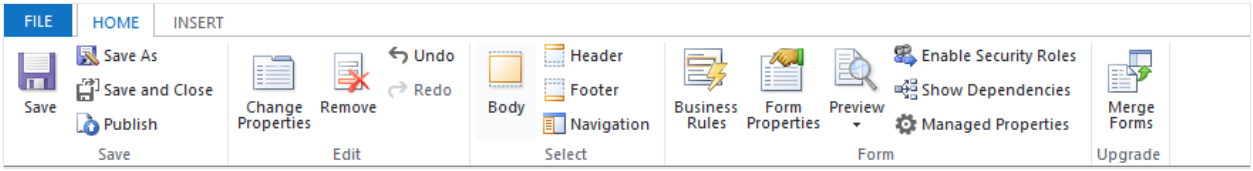
Select the **File** tab to add/view the following options:

- **New Activity** Add a new activity
- **New Row** Add a new row
- **Tools** Utilize options like Import data, Duplicate detection, and Bulk delete wizard
- **Options** Change the default display settings to personalize the default solution, and manage your email templates
  - General
  - Synchronization
  - Activities
  - Formats
  - Email Templates
  - Email Signatures
  - Email
  - Privacy

    - Languages
- Help
- Close

Home tab

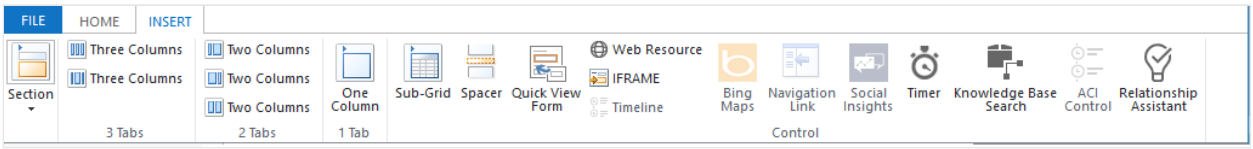
The **Home** tab displays the commands listed in the following table:



Group	Command	Description
Save	**Save (Ctrl+S)**	Save the form.
	Save As	Create a copy of this form with a different name.
	Save and Close	Save the form and close the form editor.
	Publish	Publish the form. More information: Publishing customizations
Edit	**Change properties**	Change properties of the selected item in the body. See the following sections depending on the selected item: - Tab Properties - Section properties - Common Column properties - Special column properties - Subgrid properties - Quick view control properties
	Remove	Remove the selected item.
	Undo (Ctrl+Z)	Undo the previous action.
	Redo (Ctrl+Y)	Redo the previous action.
Select	**Body**	Edit the main body of the form.
	Header	Edit the form header.
	Footer	Edit the form footer.
	Navigation	Edit the form navigation. More information: Edit Navigation

Group	Command	Description
Form	Business Rules	View, edit, or create new business rules with the Business Rules explorer. **Note:** For the interactive forms, only the "Entity" and "All Forms" scope is supported. More information: Create and edit business rules
	Form Properties	More information: Form Properties
	Preview	Use this to see how form looks after it's published. You can also preview to test scripts associated with from events.
	Enable Security Roles	Use this to set which security roles will have access to the forms. More information: Control access to forms **Important:** If you create a new form, only the System Administrator and System Customizer security roles will have access to the form. You must assign access to other security roles before people can use it.
	Show Dependencies	See which solution components depend on this form and which solution components are required by this form.
	Managed Properties	Managed properties command has two properties **Customizable** and **Can be Deleted**. Setting these properties to false means the form won't be customizable and can't be deleted after you included it in a solution, export that solution as a managed solution, and import that managed solution into a different environment. More information: Managed properties
Upgrade	Merge forms	If applicable, this option lets you merge this form with a form from a previous version of Dynamics 365 form

Insert tab



The Insert tab displays the commands in the following table:

Group	Command	Description
	Section	Add a section to a selected tab. You can include a section with one to four columns. You can also insert a Reference panel in the interactive forms. Reference panel is also added as a section to the Main - Interactive

Group	Command	Description
		experience form. By default the Reference panel section is added to the Case, Account, Contact and custom table forms. More information: Section properties
3 Tabs	Three Columns	Insert a three-column tab with equal widths. More information: Tab Properties
	Three Columns	Insert a three-column tab with a wider middle column.
2 Tabs	Two Columns	Insert a two-column tab with a wider right column.
	Two Columns	Insert a two-column tab with a wider left column.
	Two Columns	Insert a two-column tab with equal width columns.
1 Tab	One Column	Insert a one-column tab.
Control	Sub-Grid	Format a subgrid and insert it into the form. More information: Subgrid properties
	Spacer	Insert an empty space.
	Quick View Form	Insert a Quick View Form. More information: Quick view control properties
	Web Resource	Insert a web resource to embed content from other locations in one page. More information: Web Resource properties
	IFRAME	You can add an IFRAME to a form to integrate content from another website within a form.
	Timeline	Insert a timeline control in the form. This control shows the timeline of activities related to the table on a form.
	Navigation Link	Using this option, you can insert a link into a form navigation.
	Timer	Insert a timer control to a table form to track time against an SLA. More information: Add a timer control
	Knowledge Base Search	Insert a search control that users can use to search knowledge articles. More information: Knowledge Base Search control
	Relationship	Using this option, you can insert a relationship assistant control in the

Group	Command	Description
	Assistant	form.

> **Note**
>
> The following components aren't supported in the Main forms:
>
> - Bing maps
> - Yammer
> - Activity Feeds

Next steps

Use the Main form and its components

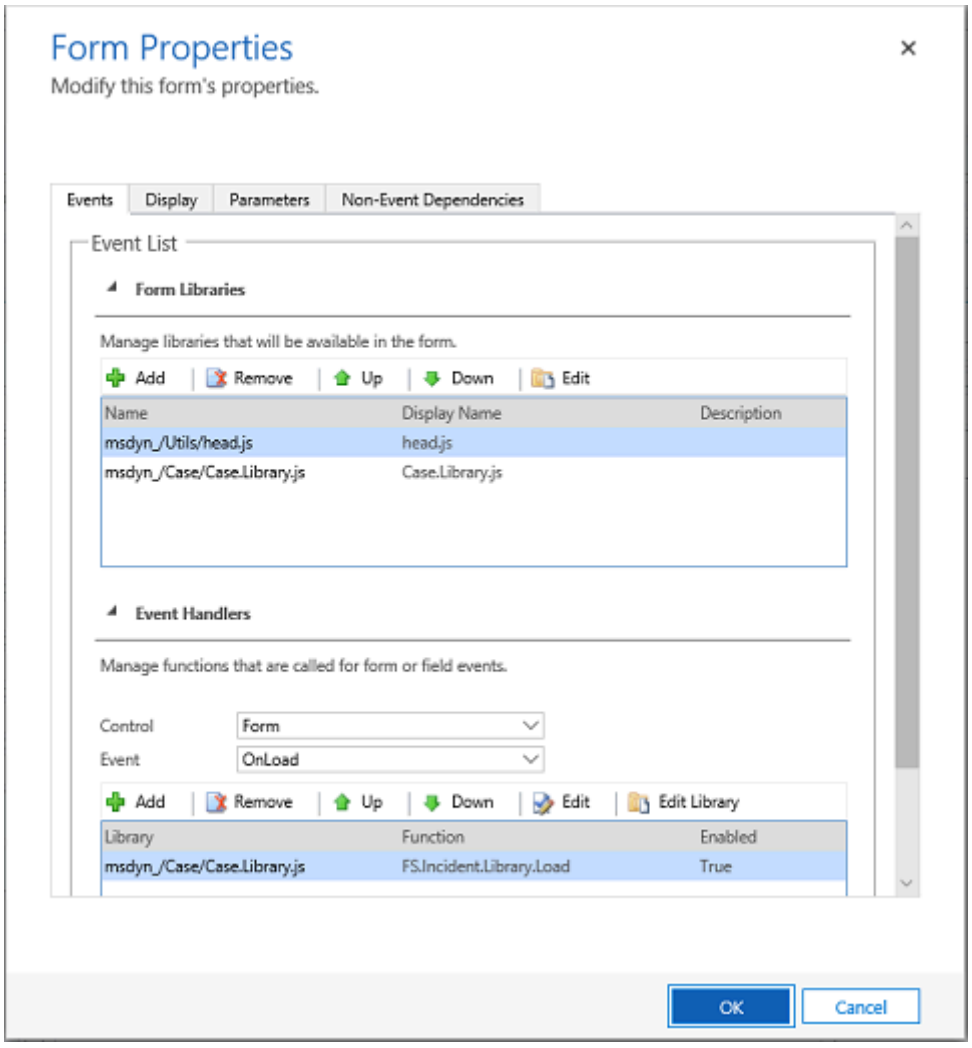
Model-driven app form properties

Article • 10/21/2023

> **Important**
>
> Starting in October 2023, the classic app, form, and view designers are deprecated and all model-driven apps, forms, and views will only open in the modern designers. By default, the **Switch to classic** command to revert back to the classic designer from the modern designer will no longer be available. More information: Classic app, form, and view designers are deprecated
>
> We recommend that you transition to use the modern designers to create and edit your model-driven apps and components.

This article describes the classic solution explorer. We recommend that you use the modern form designer to edit properties for your model-driven app forms. More information: Configure a form and Configure model-driven app form event handlers

You can access **Form properties** in solution explorer. Under **Components**, expand **Tables**, expand the table you want, and then select **Forms**. In the list of forms, open the form of type **Main**. Then on the **Home** tab, select **Form Properties**.



The following table lists the form properties:

Expand table

Tab	Property	Description
Events	**Form Libraries**	Manage which JavaScript web resources are available in the form and the order in which they'll be loaded.
	Event Handlers	Configure which JavaScript functions from the Form Libraries run for the `OnLoad` and `OnSave` form events and the order in which they'll be run.
Display	**Form Name**	Enter a name that will be meaningful to people. This name will be shown to people when they use the form. If they can use multiple forms configured for the table they'll use this name to differentiate between available forms.
	Description	Enter a description that explains how this form is different from other main forms. This description is only shown in the list of forms for a table in the solution explorer.
	Form Assistant	Select from the check box if you want to enable form assistant or view the form as expanded by default.

Tab	Property	Description
	Page Navigation	You can choose not to show navigation items. In forms for updated tables this means the primary name value for the row currently being viewed won't appear in the navigation bar to allow navigation to associated views. In forms using the classic presentation, the navigation options to choose associated views on the left side of the form won't be shown.
	Image	When a table has an image column and the entities' **Primary Image** option is set, this setting will enable showing the image column in the header of this form. See Enable or disable table options for more information about table options.
	Display	**Set a Max Width (in pixels)** to limit the width of the form. The default value is 1900.
	Display	Enter in pixels the max width that you want for the form here.
Parameters	**Parameters**	Each form can be opened with code using a URL. The URL might also contain data that can be passed to the form using a query string that is appended to the URL. Query strings look like this example: `?p_firstName=Jim&p_lastName=Daly` As a security measure, forms won't accept any unknown query string parameters. Use this parameters list to specify parameters this form should accept to support code that will pass data to the forms using a query string. The name and type of data will be checked and the form won't open if invalid query string parameters are passed to it. **Note:** The name can't start with an underscore (_) or crm_. It must start with alphanumeric characters followed by an underscore (_). For example, parameter_1 or 1_parameter. The name can't contain hyphens (-), colons (:), semicolons (;), commas (,) or periods (.).
Non-Event Dependencies	**Dependent Columns**	Each event handler has a similar **Dependent Columns** property so that any columns that are needed by the script can be registered. Anyone who tries to remove the dependent columns won't be able to.

Tab	Property	Description
		Some scripts operate on the form but aren't configured in an event handler. Scripts that are initiated from the command bar don't have a place where dependent columns can be registered. This form property provides a place for dependent columns for those scripts to be registered.

Next steps

Use the Main form and its components

Feedback

Was this page helpful? Yes No

Provide product feedback

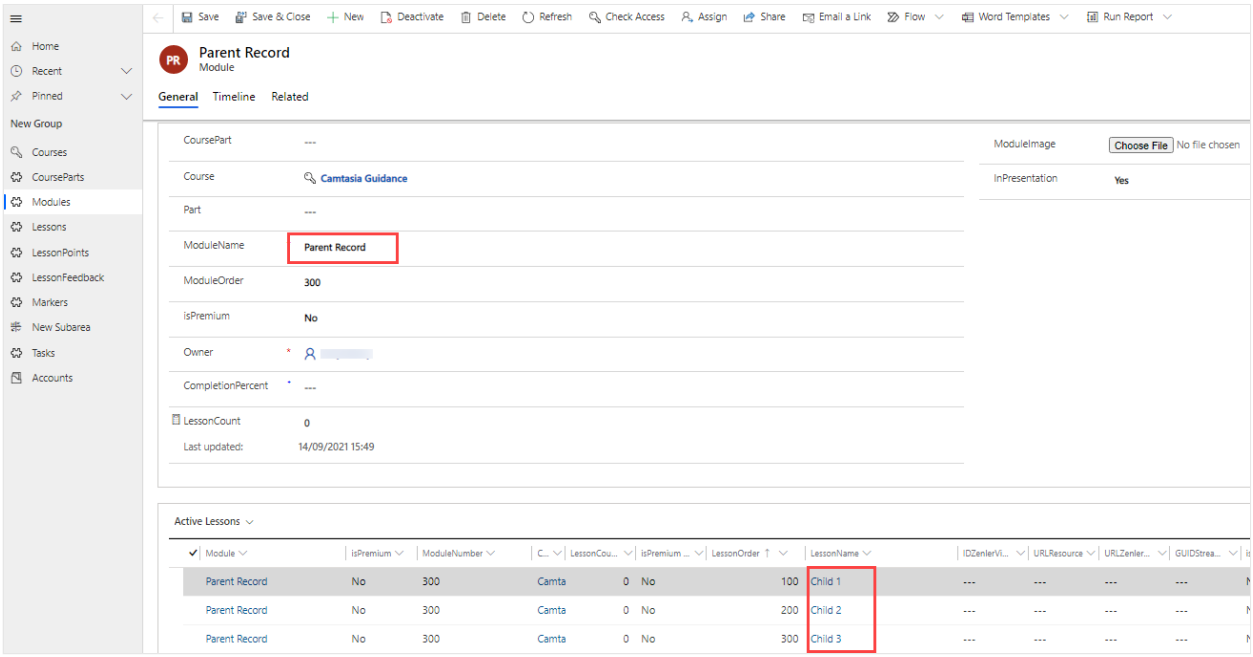
Subgrid properties for model-driven app main forms (classic)

Article • 09/26/2022

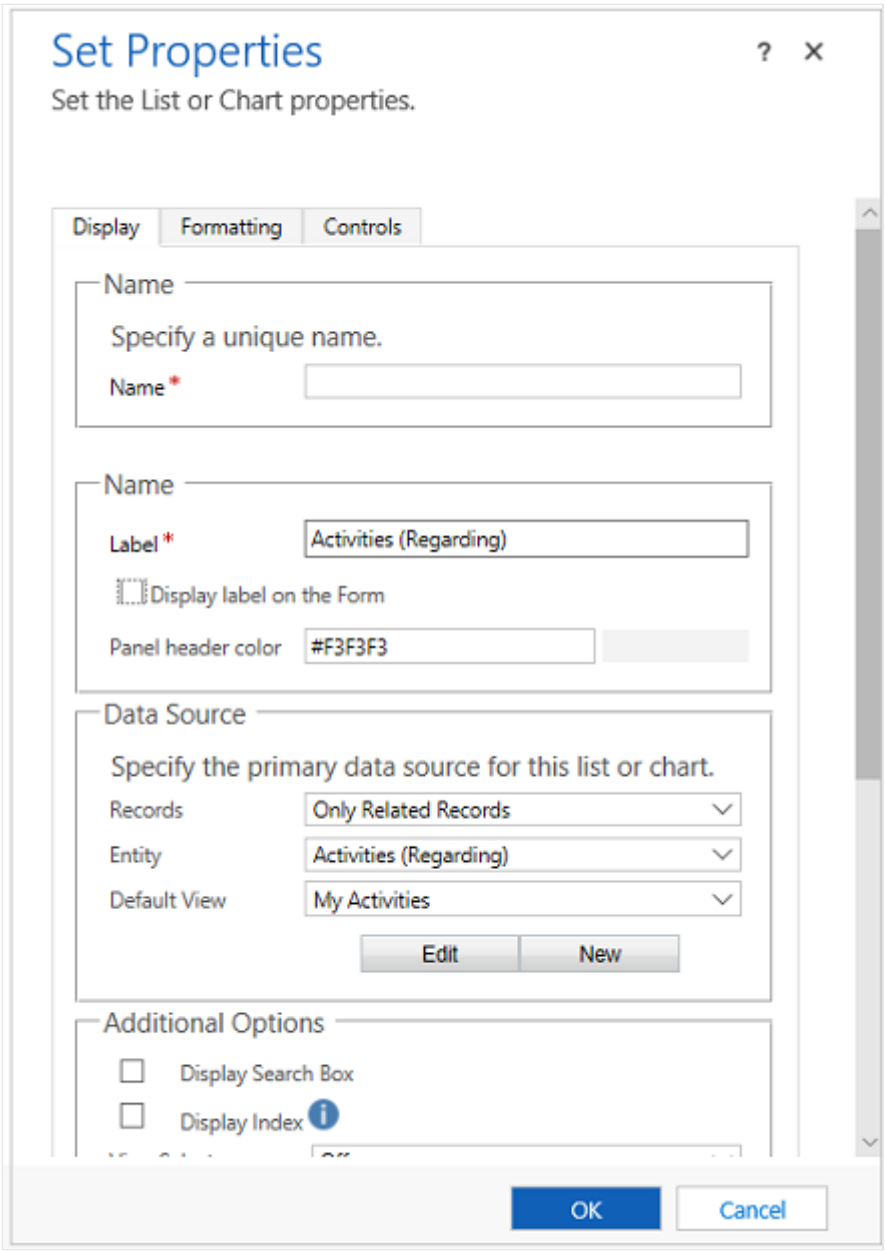
This article describes the classic form editor.

We recommend that you use the **modern form designer** to create and edit subgrids for model-driven app forms. More information: Add and configure a subgrid component on a form

A subgrid exists within a main form and let app users view data within a Dataverse table, typically related to the record currently being reviewed.



A subgrid on a form can be configured using the classic form designer to display a list of records or a chart. Select **Show Chart Only** on the **Display** tab to show a chart instead of a list.



These are the properties available to configure when using a subgrid component on a form using the classic form designer.

Tab	Property	Description
Display	**Name**	**Required**: The unique name for the subgrid that is used when referencing it in scripts. The name can contain only alphanumeric characters and underscores.
	Label	**Required**: The localizable label for the subgrid visible to users.
	Display label on the Form	Whether the label should be displayed on the form. This is required if you enable **Display Search Box**. You can also choose to have the panel header color.

Tab	Property	Description
	Records	Choose from two options: - **Only Related Records**: Subgrid will display only records related to the current record. - **All Record Types**: Subgrid will display records filtered only by the default view or, if the view selector is enabled, any views the user chooses. The option you choose will affect the behavior of the show list control. More information: Show list behavior
	Table	Depending on the option you choose for **Records**, this list displays either: - **Only Related Records**: A list of tables that are related to this table with the name of the lookup column on that table, which defines the relationship in parentheses. - **All Record Types**: A list of all tables.
	Default View	Choose the view that will be applied by default. If you do not enable any other views using the **View Selector** property. This will be the only view. Use the **Edit** button to open the default view for editing. Use the **New** button to create a new view to use for this subgrid.
	Display Search Box	Display the search box. When this option is chosen the **Display Label on the Form** option is required.
	Display Index	Only forms using the Classic forms support display index. Select this check box if you want the alphabetical index to be available with the list. This lets you jump to records starting with a particular letter or number.
	View Selector	You have three options: - **Off**: Only the default view can be used. - **Show All Views**: Allow people to choose any view. - **Show Selected Views**: Use the Ctrl key with your cursor to select which of the available views to show.
	Default Chart	Select which chart to show if **Show Chart Only** is selected.
	Show Chart Only	Rather than a list of records a chart will be displayed.

Tab	Property	Description
	Display Chart Selection	If **Show Chart Only** is selected, allow people to choose different charts.
	Availability	Specify whether the section should be available on phone.
Formatting	**Layout**	**Select the number of columns the control occupies**. When the section containing the subgrid has more than one column, you can set the column to occupy up to the number of columns that the section has.
	Number of rows	Determines how many records are shown on a page of a subgrid.
	Use available space	Note: This property only works with the legacy web client. It has no effect on Unified Interface. If **Automatically expand to use available space** is chosen, the form will allow space for two records and will expand the space as the number of records increases. If the number exceeds the **Number of Records**, users can navigate to additional pages to view the records. If **Automatically expand to use available space** is not chosen, the form will provide space for the number of records defined by **Number of rows** and users can navigate to additional pages to view any additional records.
Controls	**Controls**	Choose to add controls and select the radio button to have them for Web, Phone, or Tablet.

In forms using the classic forms, actions performed on a subgrid were available in the ribbon. Developers can customize the behavior of these actions or add additional actions by customizing the ribbon.

In forms using the current forms actions for subgrids are placed near the subgrid, making them easier to access. However the command bar does not allow for custom actions to be added. Developers can edit the ribbon to modify the actions for the remaining three actions: show list, add record, and delete record.

Show list behavior

When displaying a list in forms with the current forms, each subgrid displays the **Open View** button in the top-right when the table is also displayed as one of the tables included in the navigation area of the form editor. Choosing this button will open the

view. The behavior will change depending on the option chosen for the **Records** property.

When you select **Only Related Records** the view will open using one of the associated views in the same window. To return to the form, use the back button or choose the current record primary name value in the navigation bar.

When you select **All Record Types** the view will open in a new window.

Add record behavior

When displaying a list in forms with the current forms, each subgrid displays the **Add record** button ➕ in the top-right side of the subgrid. Choosing this button will allow you to add a record. This behavior will change depending on the option chosen for the **Records** property and if the lookup is for activity records.

When you select **Only Related Records** the default behavior is the behavior to add existing records. People see an in-line lookup to search for an existing record first. This helps prevent creating duplicate records. If they can't find an existing record, they can choose the **New** option. When a new record is created any of the column mappings defined in the relationship will be applied. More information: Map table columns

When you select **All Record Types** the default behavior is to add a new record. The quick create form will be shown if the target table has one. If not, the default table main form is shown.

If the subgrid displays activities, people will first need to choose the type of activity and then they will see the "add new record" behavior.

Delete record behavior

When you select a record in a subgrid the **Delete** button 🗑 appears on the right side of the record. The behavior of this delete action is different depending on the type of relationship with the current table.

When the subgrid uses a 1:N (one-to-many) relationship, the normal record delete behavior is to show a confirmation dialog before deleting the record.

When the subgrid uses a N:N (many-to-many) relationship, the record in the relationship (or intersect) table relating to two records is deleted without a confirmation and the record will no longer be displayed in the subgrid. But the record that was displayed is not deleted.

Next steps

Use the Main form and its components

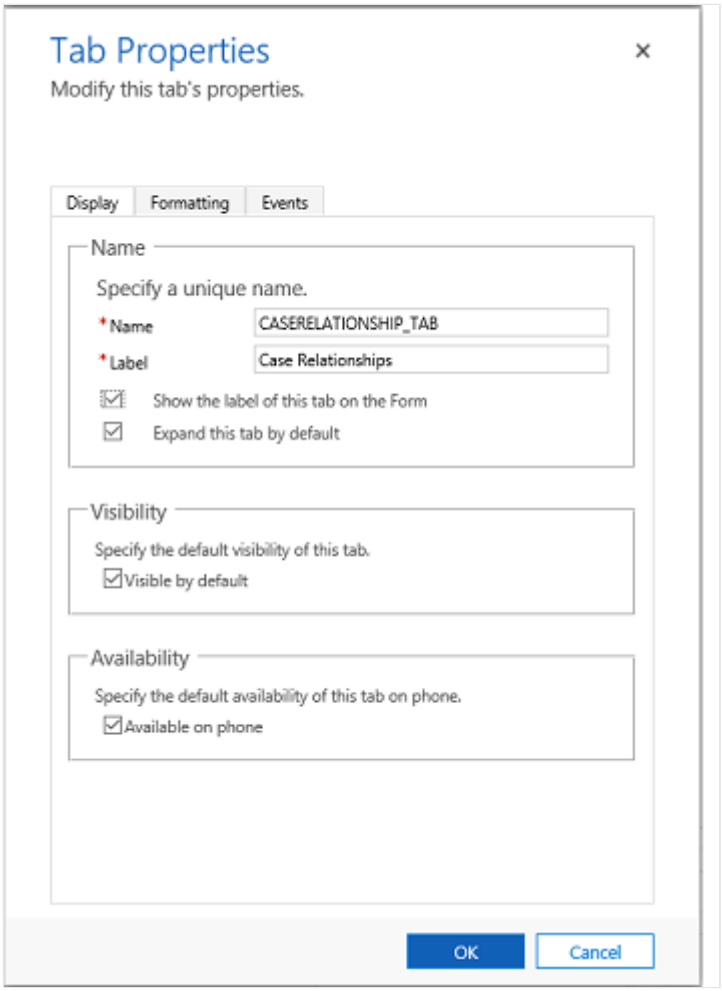
Tab properties for model-driven app forms overview

Article • 02/15/2022

This article describes the classic form editor. We recommend that you u se the modern form designer to add and edit tabs for your model-driven app forms. More information: Add, configure, move, or delete tabs on a form

These are the properties available to configure when using a tab on a form using the classic form designer. The following table shows properties that you can set for tabs on the form:

Tab	Property	Description
Display	**Name**	**Required**: The unique name for the tab that is used when referencing it in scripts. The name can contain only alphanumeric characters and underscores.
	Label	**Required**: The localizable label for the tab visible to users.
	Show the label of this tab on the Form	When the label is displayed people can select it to toggle whether the tab is expanded or collapsed. Choose whether you want to show the label.
	Expand this tab by default	The tab state can toggle between expanded or collapsed using form scripts or by people selecting the label. Choose the default state for the tab.
	Visible by default	Showing the tab is optional and can be controlled using scripts. Choose whether to make the tab visible. More information: Visibility options
	Availability	Choose if you want the tab to be available on the phone.
Formatting	**Layout**	Tabs may have up to three columns. Use these options to set the number of tabs and what percentage of the total width they should fill.
Events	**Form Libraries**	Specify any JavaScript web resources that will be used in the tab `TabStateChange` event handler.
	Event Handlers	Configure the functions from the libraries that should be called for the tab `TabStateChange` event. More information: Configure Event Handlers



Next steps

Use the Main form and its components

Use controls for model-driven app data visualizations

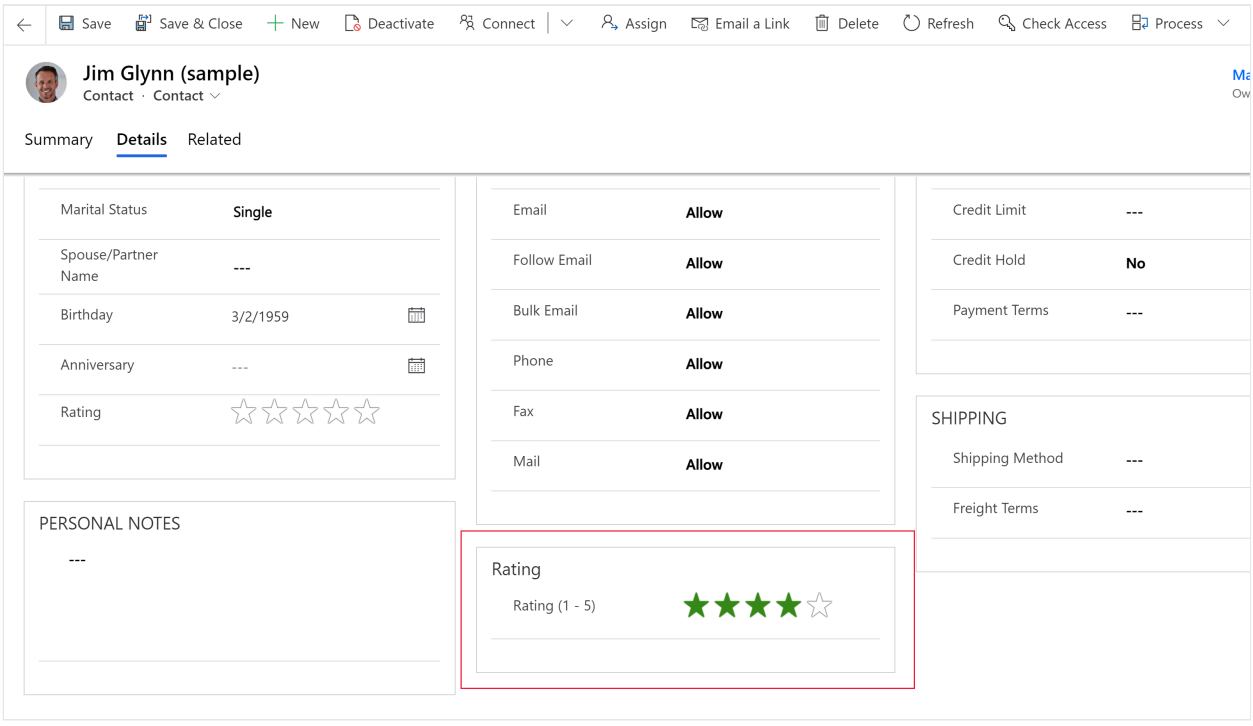
Article • 03/13/2023

In this article, you learn how to configure a control for a column using the classic form designer.

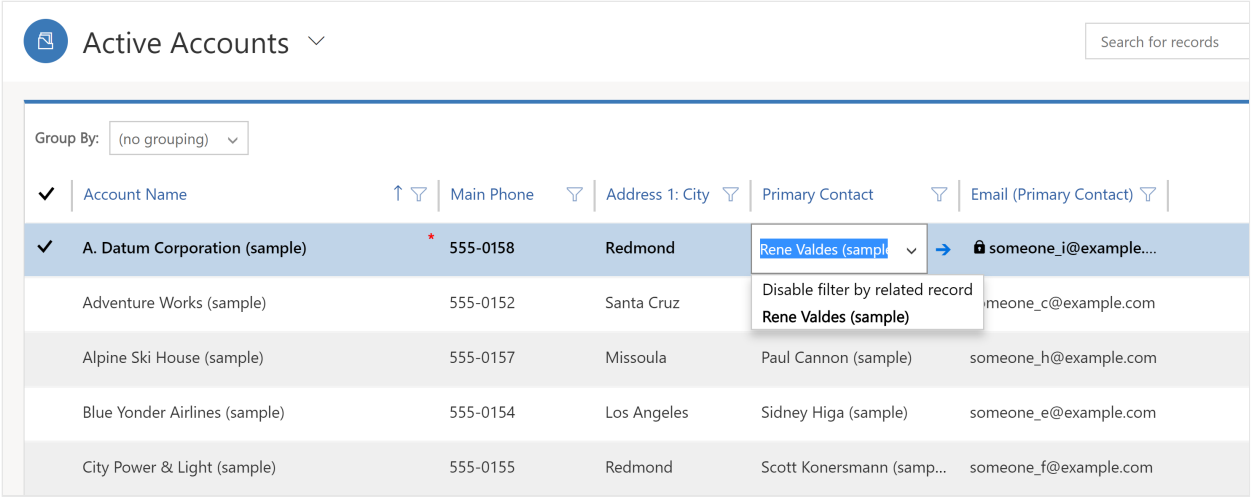
> **Important**
>
> This article describes classic controls many of which are now deprecated. For information about how to add modern controls using the modern Power Apps form designer, go to **Add components to a form**. For a list and descriptions of the current controls available for model-driven apps, go to **List of controls available for model-driven apps**.

Use a control using the classic form designer

Controls let you transform app user interface components, such as a column or view that traditionally contain text, into visualizations. Controls can be configured on columns, forms, dashboards, views, and grids. For example, a star rating can be configured on a whole number column.



Or the editable grid control can be configured on a view.



You can set one type of custom control to appear in the web browser client while having a different custom control appear in your Dynamics 365 phone or tablet mobile apps. For example, you could use a standard yes/no choice column in web browser clients and a toggle control for tablet and phone clients. After the customization is published, users can fully interact with the control to change the value, such as by tapping the control when using the toggle control. Changes are automatically saved when the form is closed just as they are when the user changes a traditional column on a form.

Use a custom control to add visualizations to a column

Following the steps in this procedure will change the default label and text box column of the **Budget Amount** column to the slider custom control on the opportunity table. You can use similar steps to replace an existing column with a custom control or configure a custom control for a custom column.

> **Note**
>
> Currently, this control can only be added and configured using the classic experience.

> The opportunity table requires a Dynamics 365 app, such as Dynamics 365 Sales.

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane, select the **Opportunity** table, and then select the **Forms** area. If the item isn't in the left navigation pane, select ...More and then select the item you want.

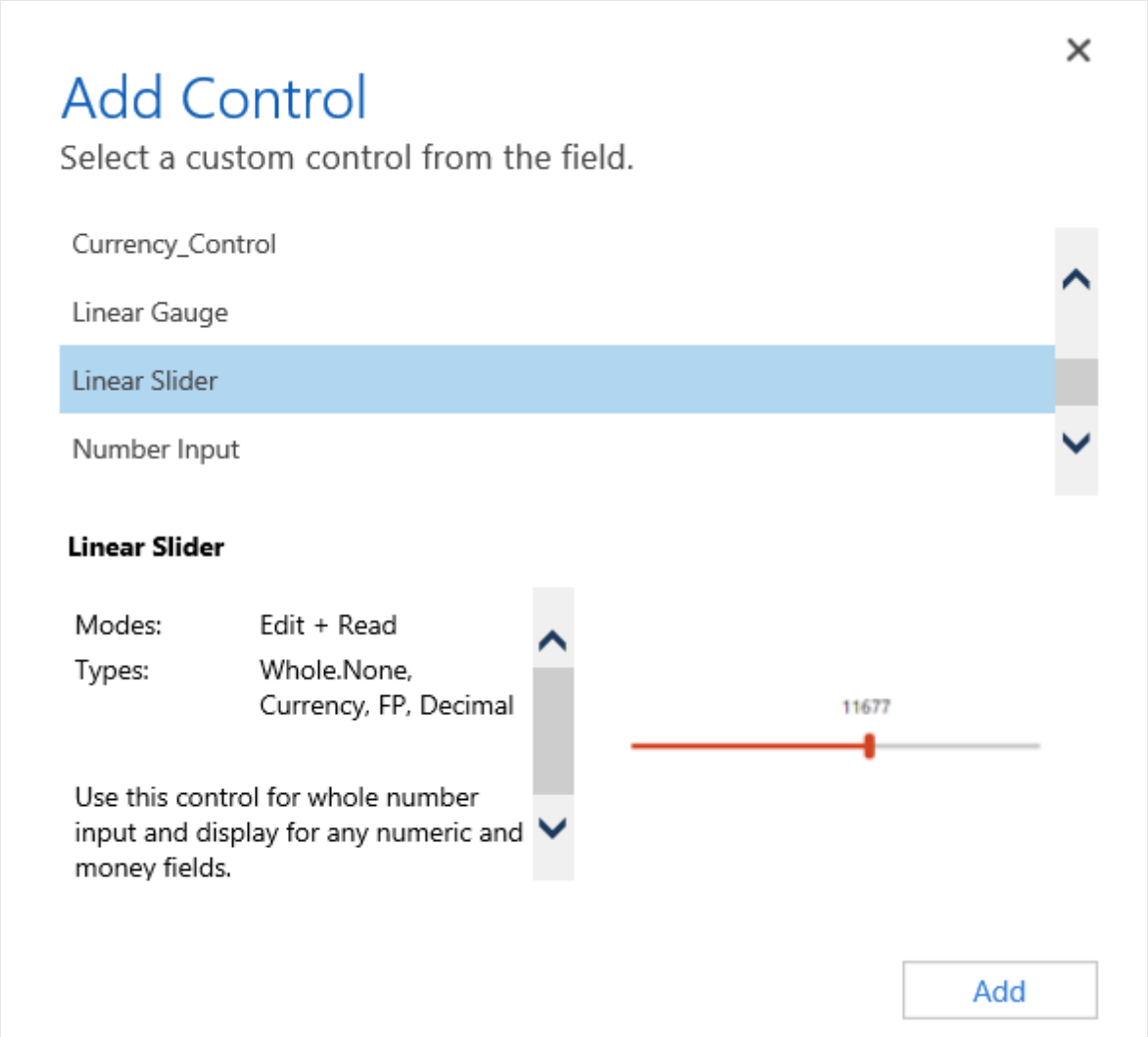
3. Open a form such as the **Main** form for the **Opportunity** table.

4. Once in the form, select **switch to classic**. This will open the classic editor in a separate tab.

5. In the form editor, double-click the column where you want to add a custom control, such as the **Budget Amount** column on the account main form. This is column of a number type. Alternatively, you can create a custom column.

6. On the **Column Properties** page, select the **Controls** tab, and then select **Add Control**.

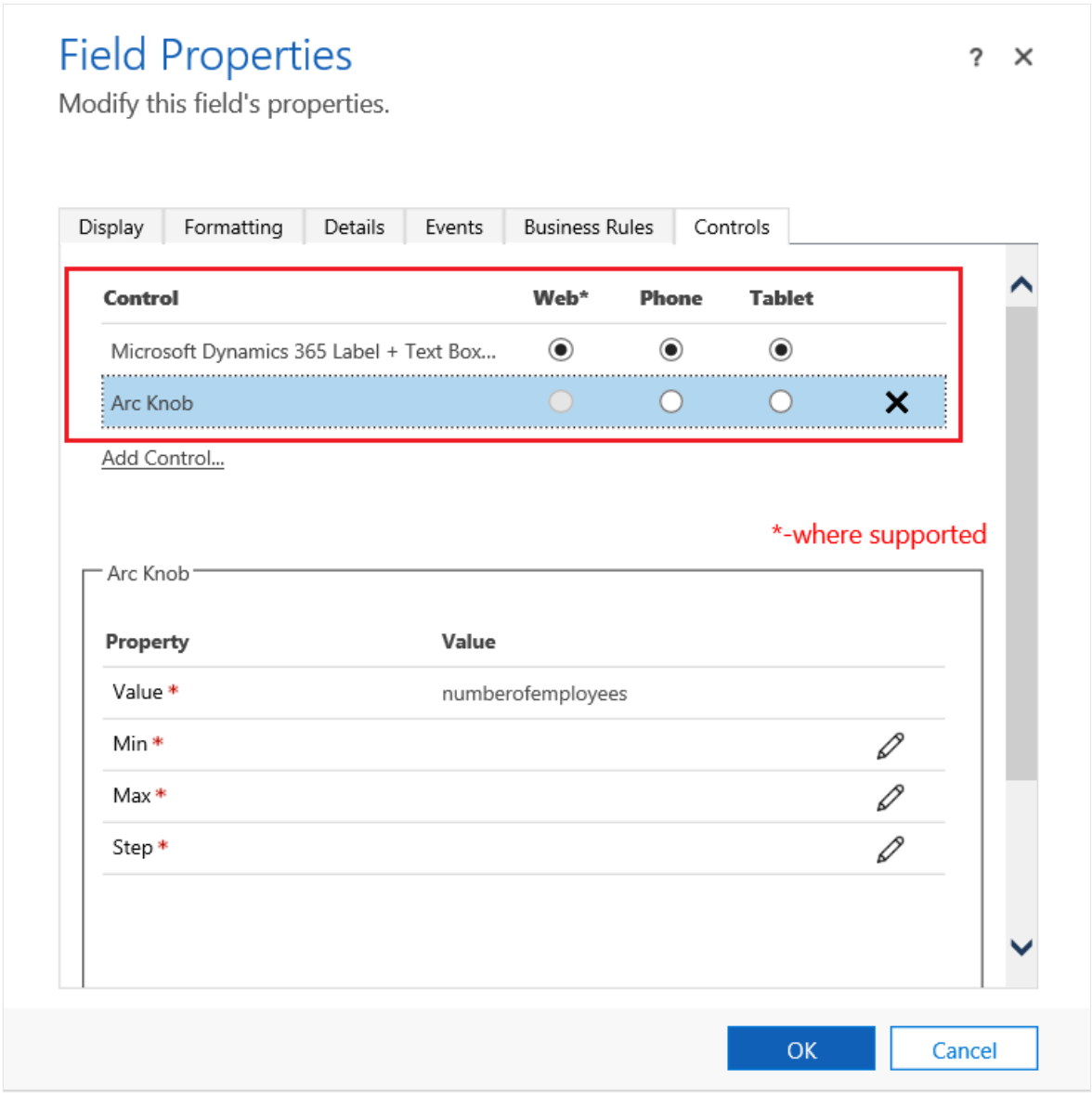
7. On the Add Control page, select the control that you want, such as the **Linear Slider** control shown here, and then select **Add**.



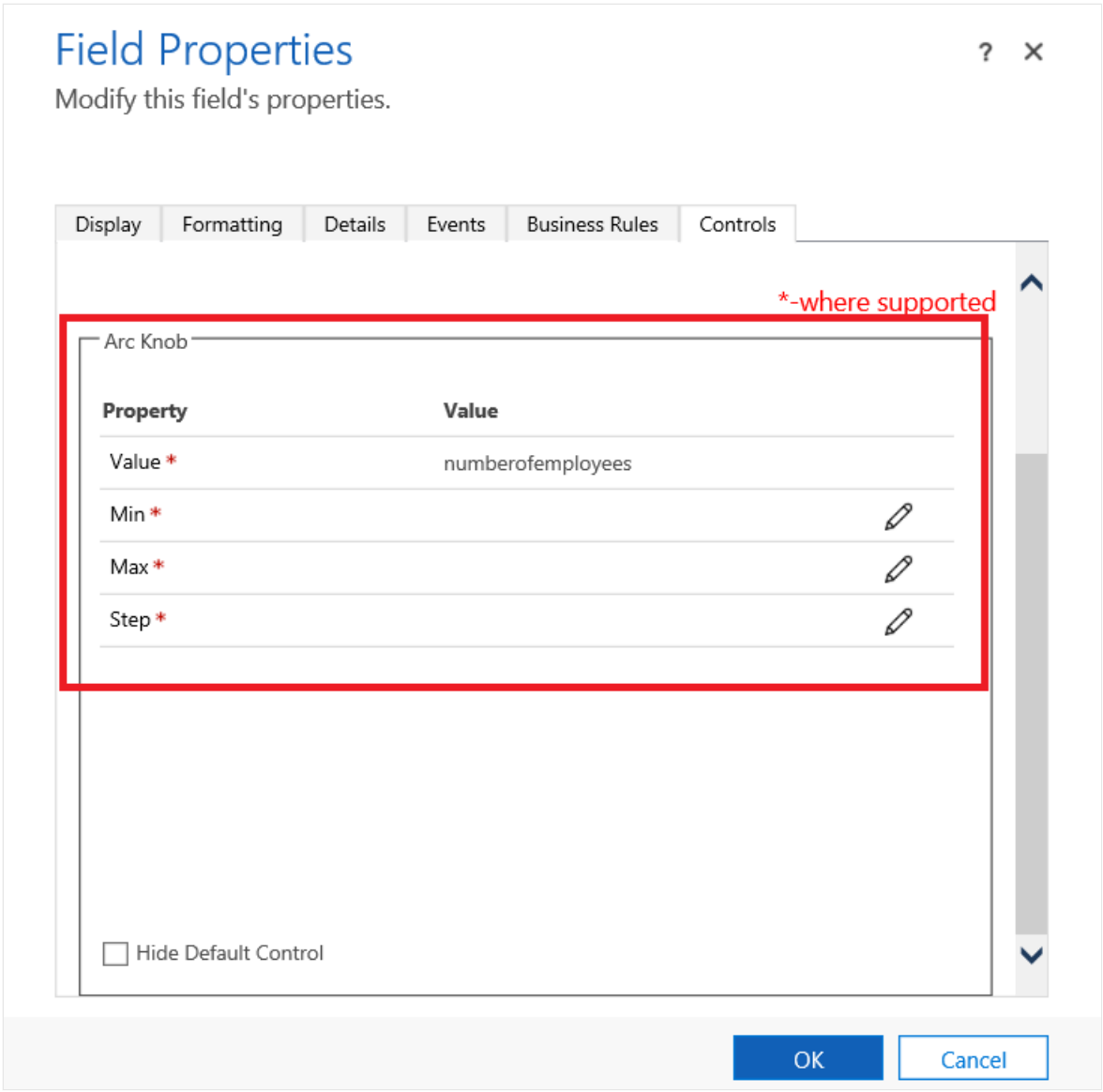
8. Choose the client where you want the control to appear.

   - **Web**. To make the custom control available from any web browser, select the **Web** option next to the control. Notice that setting the **Web** option includes rendering the control in web browsers on PCs, Macs, and mobile devices.

- **Phone**. To make the custom control available on phones running Dynamics 365 for phones, select the **Phone** option next to the control.
- **Tablet**. To make the custom control available on tablet devices running Dynamics 365 for tablets, select the **Tablet** option next to the control.



9. Select the pencil icon next to **Min**, **Max**, and **Step**, set the property option described below, and then select **OK**.

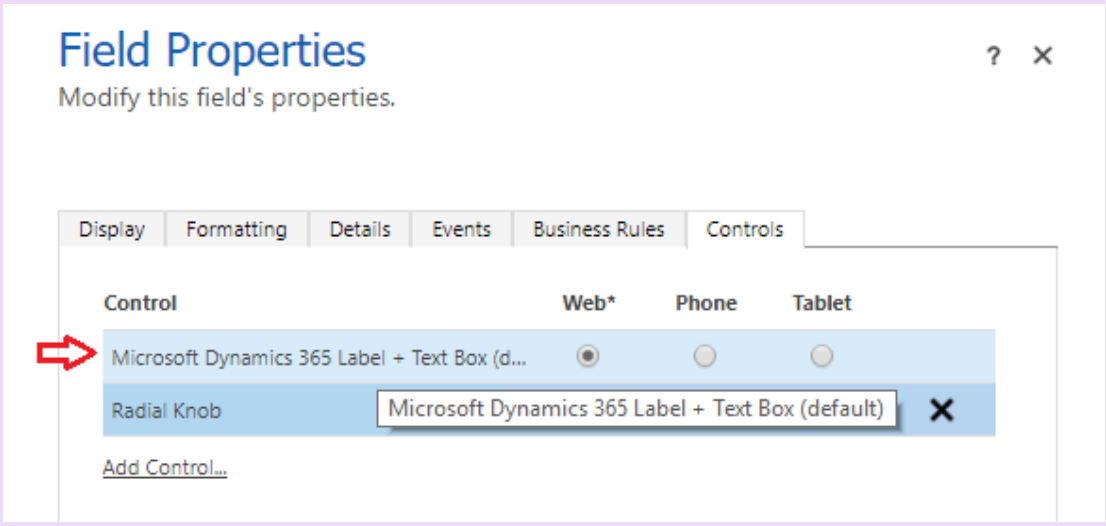


- **Min**. Set the minimum accepted value. You can bind a static value that you enter or bind the value to an existing column. In this example **Bind to static value** is **Currency** and the minimum value that can be entered is *zero*.
    - **Bind to a static value**. Select the data type, such as a whole number (Whole.None), currency, floating point (FP), or decimal. Next, enter a number that represents the minimum accepted value for the column.
    - **Bind to values on a column**. Select a column from the list that will be used as the minimum accepted value.
- **Max**. Set the maximum accepted value for the column. Similar to the Min value, you can bind a static value that you enter or bind the value to an existing column as described earlier. In this example, **Bind to static value** is **Currency** and the maximum value that can be entered is **1 billion**.
- **Step**. This represents the unit to increment or decrement when adding to or subtracting from the current value. For example, for budget amount you can select

100 dollar increments\decrements.

- **Hide Default Control**. Selecting this option hides the control so neither the control or the data is displayed in any of the clients that don't support the custom control. Notice that the classic Dynamics 365 web client doesn't support most custom controls. By default, this option is not selected and the classic Dynamics 365 web client displays the default, typically text based, control.

> ⓘ Note
>
> The default control is identified with **(default)** following the control name.



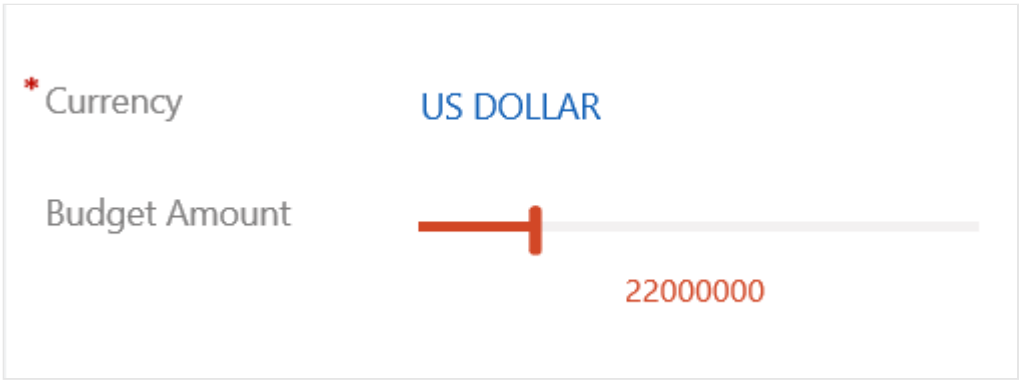
10. Select **OK**, to close the **Column Properties** page.

11. To activate the customization, on the table form select **Save**, and then select **Publish**.

12. Select **Save and Close** to close the form editor.

See the custom control in action

Open a record that includes the column with the custom control, such as the opportunity form from the previous example, and view how the column is changed.



The column is now rendered as a slider control instead of the text column.

Use the editable grid control on a view or subgrid

With editable grids, users can do rich in-line editing directly from views and subgrids whether they're using a web app, tablet, or phone. More information: Make grids (lists) editable using the Editable Grid custom control

Next steps

Create and edit columns

Model-driven app form section properties (classic)

Article • 03/13/2023

A section in a form occupies the space available in a tab column. Sections have a label that can be displayed and a line might be shown below the label.

Sections can have up to four columns and includes options for displaying how labels for columns in the section are displayed.

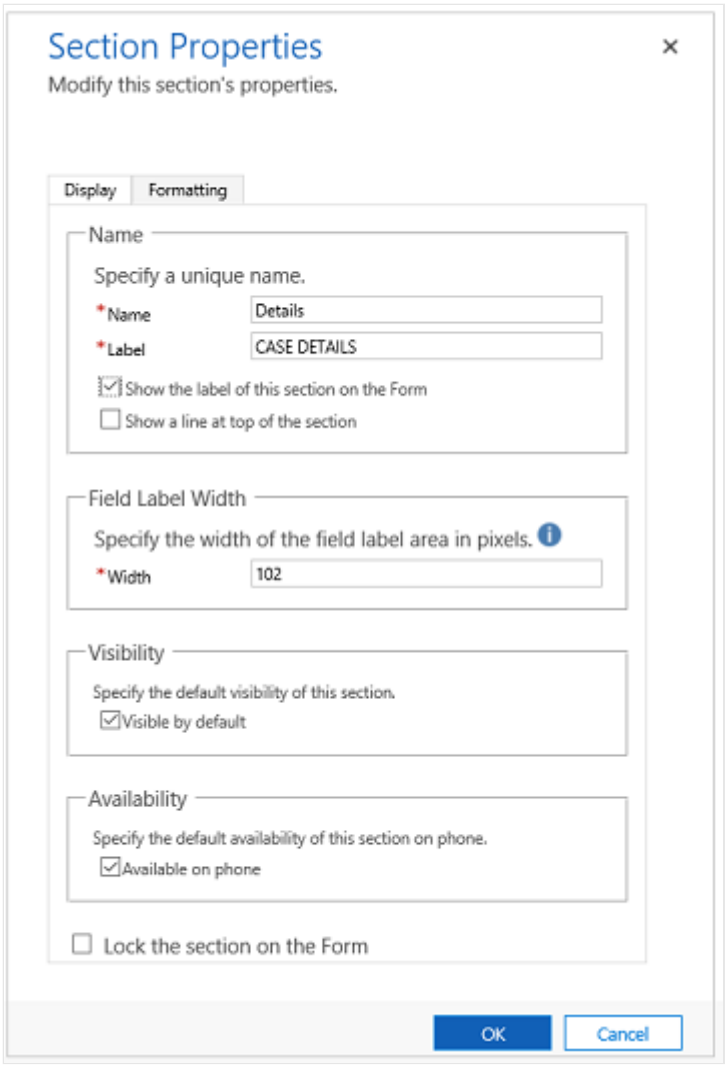
Headers and footers are similar to sections but cannot be removed. If they don't contain anything they will not be shown.

> ⓘ **Note**
>
> For the modern experience when working with sections on a form, see **Add, configure, move, or delete sections on a form**

Section properties in classic form designer

You can access **Section properties** in solution explorer from Power Apps (make.powerapps.com).

1. Sign in to Power Apps.
2. Select **Tables** on the left navigation pane, select the table that you want, and then select the **Forms** area. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. In the list of forms, open the form of type **Main**.
4. Select **Switch to classic** to edit the form in the classic form designer.
5. Double-click inside one of the sections to see the section properties.



Tab	Property	Description
Display	**Name**	**Required**: The unique name for the section that is used when referencing it in scripts. The name can contain only alphanumeric characters and underscores.
	Label	**Required**: The localizable label for the section visible to users.
	Show the label of this section on the form	Sections are frequently used without labels to control formatting of the columns within them.
	Show a line at top of the section	A line at the top of a section can help break up the form layout.
	Column Label Width	**Required**: Set a value between 50 and 250 to specify the space allowed for column labels. Header and footer elements also have this property.

Tab	Property	Description
	Visibility	Showing the section is optional and can be controlled using scripts. More information: Visibility options
	Availability	Choose if you want the tab to be available on the phone.
	Lock the section on the form	This will prevent the section from accidentally being removed and prevents people from removing the contents. Removing a section will not only remove the section, but also any columns within it. Someone wanting to remove this section would need to change this setting before removing it.
Formatting Header and footer components also have this property.	**Layout**	Specify up to four columns to be in the section.
	Column Label Alignment	Labels for columns within the section can be aligned left, right, or center.
	Column Label Position	Labels for columns within the section can be positions on the side or on top of the columns.

A new type of section called **Reference panel** can also be added. A Reference panel is a single column section. You can insert sub-grids, quick view control, or a Knowledge Base Search control inside a reference panel section. Each control that you added in the reference panel appears as a vertical tab within the panel at the runtime. You can drag and drop the various controls within the Reference panel section. The default tab at runtime is the first control added in the reference panel. The other tabs appear in the order in which they are added in the form editor. To delete a tab, use the delete key on your keyboard.

When you insert a reference panel, by default it's added as a last section in the tab. You can add only one reference panel per form.

> **ⓘ Important**
>
> - By default, the reference panel section is locked in these standard forms: Cases, Accounts, and Contacts. To remove it or change it, you must unlock it.

- Quick view forms are based on an organization and not at an app-level setting. So, any quick view form that is available, which may or may not be included in an app, will be used to render related information in a reference panel.

Next steps

Use the Main form and its components

Configure model-driven app form event handlers

09/19/2025

Form event handlers for Power Apps forms can be configured for the following areas in a form:

Expand table

Element	Event	Description
Form	`OnLoad`	Occurs when the form loads.
	`OnSave`	Occurs when data is saved.
Tab	`TabStateChange`	Occurs when the tab is expanded or collapsed.
Column	`OnChange`	Occurs when data in the column changes and the control loses focus.
IFRAME	`OnReadyStateComplete`	Occurs when the content of an IFRAME loads.

An event handler consists of a reference to a JavaScript web resource and a function defined within that web resource that executes when the event occurs. Each element can have up to 50 separate event handlers configured.

> **Important**
>
> Configuring an event handler incorrectly can result in script errors that might cause the form to fail to load or function correctly. If you aren't the developer of the script, make sure you understand exactly what configuration options the script requires.
>
> Don't configure a script event handler using a library that doesn't come from a source you trust. Scripts can be used to perform any action a user might perform and a poorly written script can significantly damage the performance of a form.
>
> After you configure an event handler, always test it to verify it's working correctly.

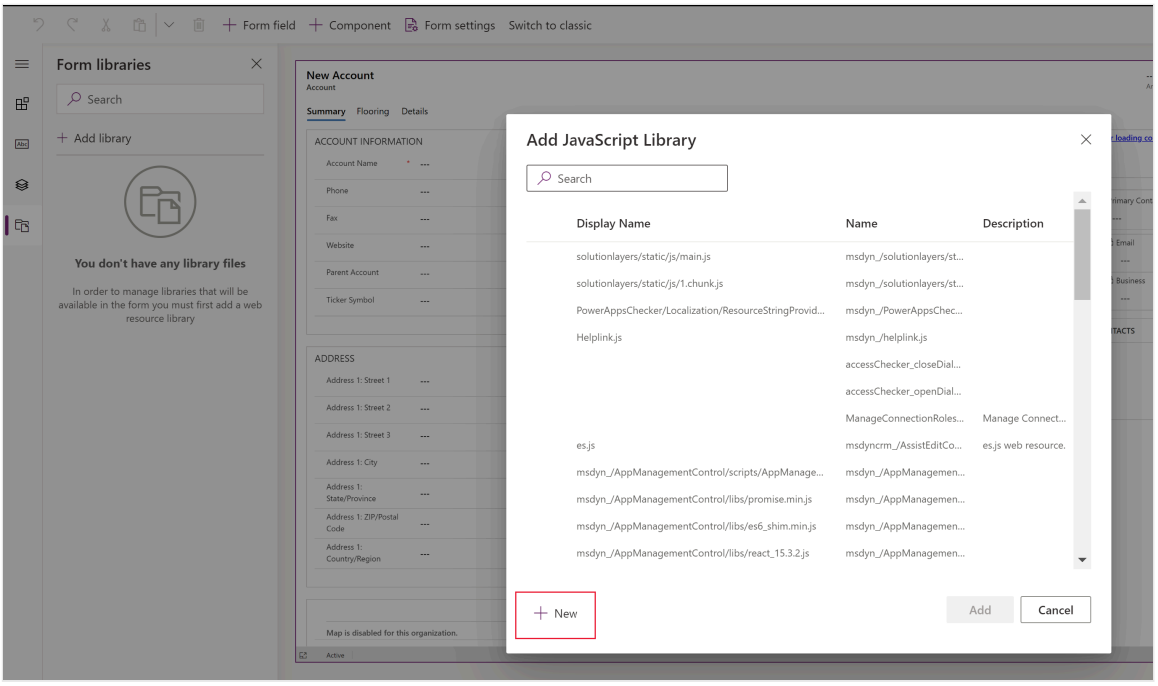
Configure an event handler

The following procedure describes how to configure an event handler for a form. Use similar steps to configure an event handler for a tab, column, or IFrame.

1. Sign in to Power Apps, open the table that you want, and then open the form in the form designer where you want to configure an event handler.

2. Depending on the library state, choose from the following options:

   - If the library containing the function you want to set as the event handler is already available, on the right properties pane under **Account Main form**, select the **Events** tab, and then go to step 3.
   - If the library containing the function you want isn't already available, select **Form libraries** on the left navigation pane, and then select **Add library**:
     a. On the **Add JavaScript Library** page, select **New**.



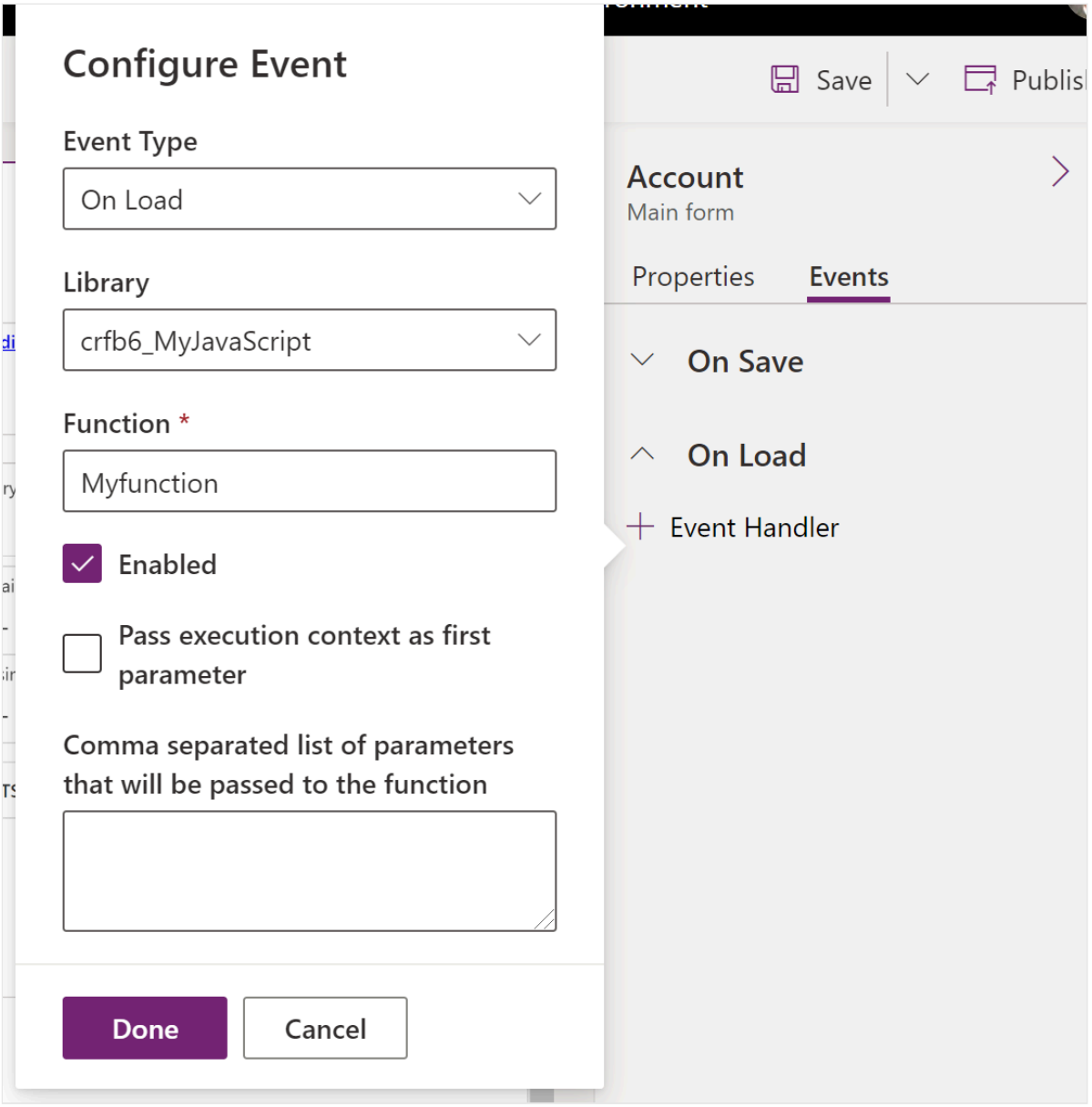
     b. In the new browser tab, complete the properties to create the JScript web resource, and then select **Save**, and then select **Publish**. More information: Create a JavaScript web resource
     c. Close the browser tab and go back to the **Add JavaScript Library** list.
     d. Select the JavaScript web resource from the list, select **Add**, and then select the **Events** tab.

> **Note**
>
> You might have to select **Cancel**, and then select **Add library** to refresh the list.

3. On the **Events** tab, select **Event Handler** and complete the following information to configure the form event, and then select **Done**:

   - **Event Type**:
     - **On Save**: When a record is saved after a form field change.
     - **On Load**: When the form opens.

- **Library**: The JavaScript web resource.
- **Function**: The name of the function within the JavaScript web resource that should be executed for the event.
- **Enabled**: By default, the event handler is enabled. Clear the **Enabled** checkbox if you don’t want to enable this event.
- **Pass execution context as the first parameter**: Some functions require an execution context to be passed to the function. Select **Pass execution context as the first parameter** if it's required.
- **Comma-separated list of parameters that will be passed to the function**: Some functions can accept a set of parameters to control the behavior of a function. The list of parameters can include variables or another function as accepted by the function parameters allowed by the JavaScript scripting language. If passing in parameters is required, enter them in the **Comma separated list of parameters that will be passed to the function**.
- **Table column dependencies**: For **On Save** events, select the columns where you want the event to be triggered after a record save.

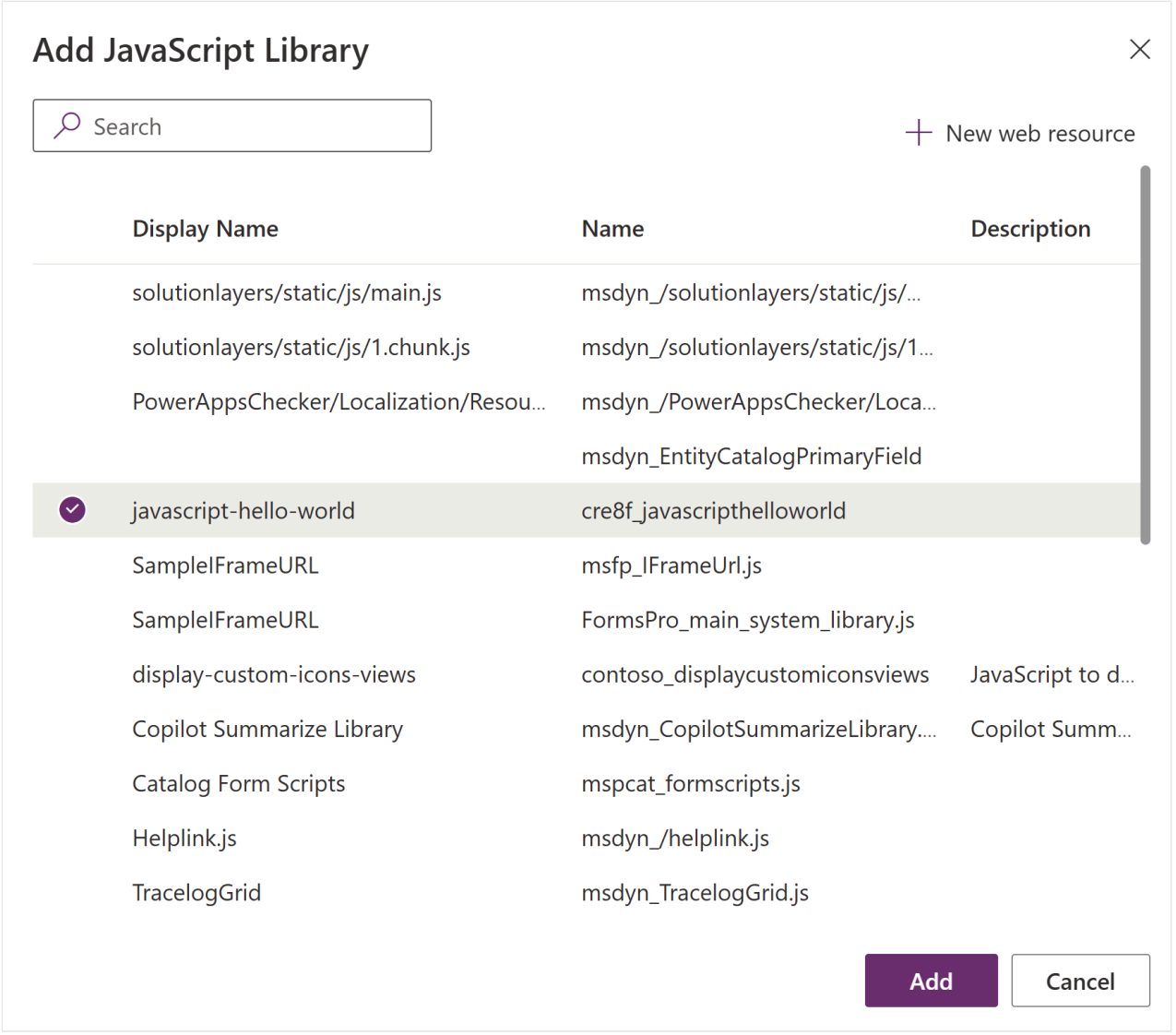


4. **Save**, and then **Publish** the form.

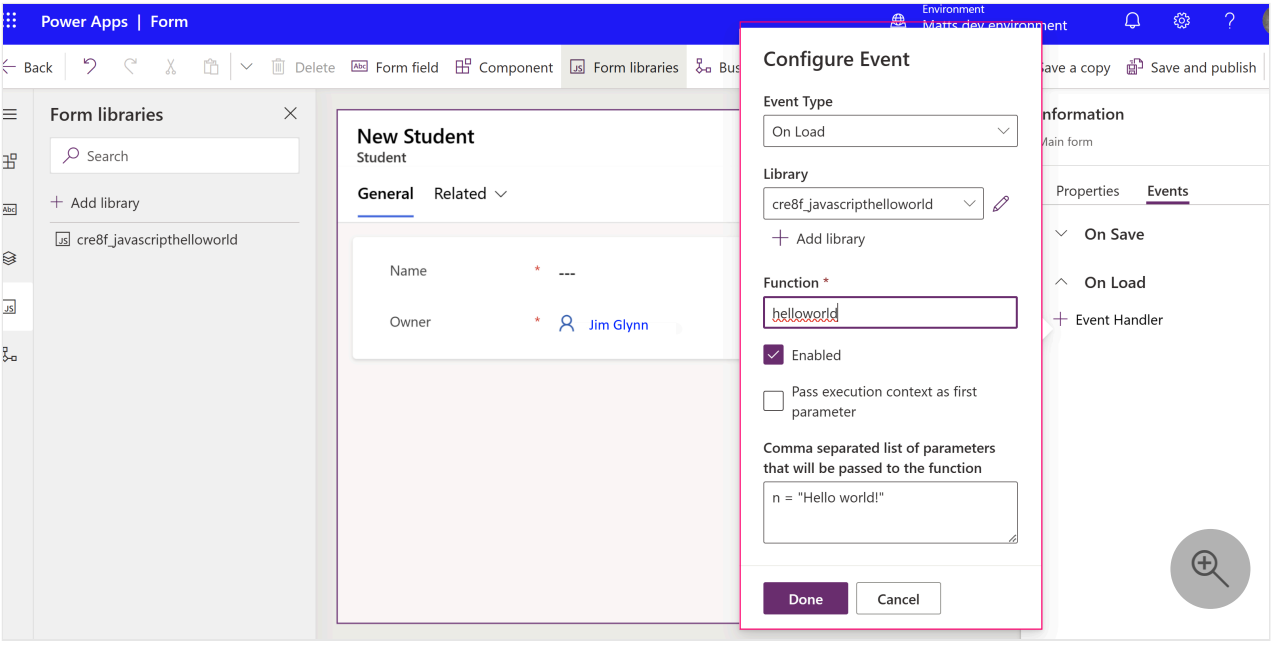
Example hello world for the onload event

This simple example demonstrates how to configure an onload form event using parameter passing of a single defined variable in the event handler, which displays an alert when the form loads.

1. Create the web resource using the Hello world JavaScript sample. More information: Create or edit model-driven app web resources to extend an app
2. Open the table Main form for editing in the form designer. This example uses the *Student* custom table main form.
3. On the left Information pane, select the **Events** tab.
4. Select **Add library**, select the web resource you created (in this example the resource is named *javascript-hello-world*), and then select **Add**.

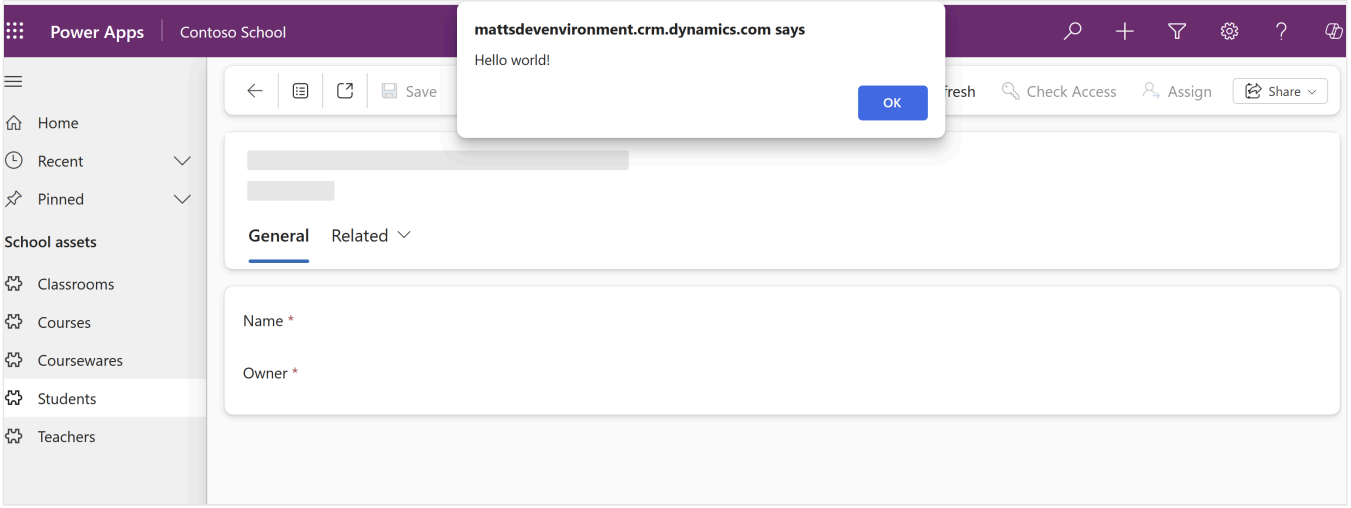


5. Select **On Load** to expand it, and then select **Event Handler**.
6. In the **Configure Event** pane, enter the **Function** name (in this example, *helloworld*), and then in the **Comma separated list of parameters that will be passed to the function** box enter a parameter such as `n = "Hello world!"`.



7. Select **Done**.
8. **Save and publish** the form.

Test the event by playing a model-driven app that includes the form. When the main form loads, an alert displays the modal dialog message "Hello world!" For this example, the user must select **OK** to dismiss the alert to continue using the form.



Hello world JavaScript sample

JavaScript

```
function helloworld(n) {
alert(n);
}
```

Configure an event handler using the classic experience

1. In the classic form editor, select the element with the event you want to configure a handler for.
2. On the Home tab, in the **Edit** group, select **Change Properties** or simply double-click the element.
3. In the element properties dialog, select the **Events** tab.
4. Expand the **Form Libraries** area. If the library containing the function you want to set as the event handler isn't already listed, add the library.
5. To add a form library to an event handler:
    a. In the **Form Libraries** section of the **Event List**, select **Add**.
    b. Locate the JavaScript web resource in the list of available web resources. Select it and then select **Add**.

If the JavaScript web resource you need doesn't exist, select **New** to open a new web resource form and create one.

c. To create a JavaScript web resource, see Create a JavaScript web resource.

d. The web resource you created is now selected in the **Look Up Row** dialog. Select **Add** to close the dialog.

6. In the **Event Handlers** section, select the event you want to set an event handler for.
7. Select **Add** to open the **Handler Properties** dialog.
8. On the **Details** tab, choose the appropriate library and type the name of the function that should be executed for the event.
9. By default the event handler is enabled. Clear the **Enabled** checkbox if you do not want to enable this event.

   Some functions require an execution context to be passed to the function. Select **Pass execution context as the first parameter** if it's required.

   Some functions can accept a set of parameters to control the behavior of a function. If these are required, enter them in the **Comma-separated list of parameters that will be passed to the function**.
10. On the **Dependencies** tab, add any columns that the script depends on into the **Dependent Columns** area.
11. Select **OK** to close the **Handler Properties** dialog.
12. When the event handler is entered, you can adjust the order in which the function is executed relative to any other functions by using the green arrows to move it up or down.
13. Select **OK** to close the element properties dialog.
14. Select **Save** to save your changes. Select **Publish** to publish the form.

> **Note**
>
> While the user interface (UI) lets you adjust the order in which the scripts are loaded by using the up and down green arrows, the scripts are actually not loaded sequentially.

Create a JavaScript web resource

1. In the web resource form set the following properties:

Expand table

Property	Value
Name	**Required**. Type the name of the web resource.
Display Name	**Required**. Type the name to be displayed in the list of web resources.
Description	Optional. Type a description of the web resource.
Type	**Required**. Select **Script (JScript)**.
Language	Optional. Choose one of the languages available for your organization.

2. If you have been provided with a script, we highly recommend that you use the **Browse** button to locate the file and upload it.

Alternatively, you can select the **Text Editor** button and paste or type the contents of the script in the **Edit Content** dialog.

> **Note**
>
> Because this simple text editor doesn't provide any features to check the correctness of the script, generally you should always try to use a separate application like Visual Studio to edit scripts, and then upload them.

3. Select **Save** and close the web resource page.

Next steps

Walkthrough: Write your first client script
Use the Main form and its components

Disable AutoSave in a model-driven app

Article • 04/21/2025

AutoSave helps app users focus on their work without having to manage saving data in the form. Most people appreciate not having to explicitly save data each time they update a row, but some organizations might have customizations that were designed expecting an explicit save. For these organizations there are options to manage how AutoSave is applied.

How AutoSave works

By default all main forms for Updated tables and classic tables have AutoSave enabled. After a row is created (initially saved), any changes made to a form are automatically saved 30 seconds after the change is made. If no changes are made in the form, the automatic save doesn't occur while the form is open. After a change is made, the 30-second period before an AutoSave begins again. If someone else has updated the same row while you're editing it, those changes are retrieved and displayed in the form when AutoSave occurs.

With AutoSave enabled, the save button only appears for the initial save of the row. After the row is created, the save button in the command bar isn't shown, but you can see a 💾 button in the lower right corner that shows if there are any unsaved changes. This control is also displayed if AutoSave is disabled.

You can select this button to save the row and refresh data in the form immediately. When AutoSave is enabled the row is saved whenever you navigate away from a row or close a separate window displaying a row. There's no need for the **Save & Close** button that appears in forms for tables that aren't updated.

Should you disable AutoSave?

If you have plug-ins, workflows, or form scripts that execute when a row is saved, they run each time AutoSave occurs. This pattern might lead to undesirable behaviors if these extensions weren't designed to work with AutoSave. Whether AutoSave is enabled or not, plug-ins, workflows, and form scripts should be designed to look for specific changes, and shouldn't execute indiscriminately for each save event.

If you have auditing configured for a table, each save is treated like a separate update. If someone lingers on a form with unsaved changes for more than 30 seconds, you observe another entry only if they add more data after the AutoSave is performed. If you have reports that depend on auditing data and treat each save as an individual "touch" of a row, you might see an increase in the frequency of touches. If you're using this approach, you should consider that individual user behaviors make it an unreliable metric with or without AutoSave enabled.

Disable AutoSave for the organization

If you determine that AutoSave causes problems with extensions you're using, Power Platform admins can disable it for the environment. There's no setting to disable AutoSave for individual tables or forms.

1. Sign into the Power Platform admin center, go to **Manage** > **Environments**, and then open the environment you want.
2. Select **Settings** on the command bar.
3. Expand **Product**, and then select **Behavior**.
4. Under **Basic behavior**, for the **Auto save**, select **Off**.

Disable AutoSave for a form

If you want to disable AutoSave for specific table forms, you can add code to the `OnSave` event in a table.

> **Note**
>
> By using this method, auto-save is disabled for the form, but data is still saved when you select the button in the lower-right corner. If you attempt to navigate away from a form or close a form where data was changed, you get a prompt to save changes before you're allowed to navigate away or close the form.
>
> Using the procedure here causes the **Save and close** button on the form to not work, as it calls `preventDefault` on `saveMode` 2, which references **Save and Close**. You can remove the `getSaveMode() == 2` check to avoid this, but then the form autosaves when you attempt to navigate away from the form or close the form where data has changed. The **Save and Continue** button also doesn't work for this reason.

1. Sign in to Power Apps.

2. Select **Tables** on the left navigation pane, select the table that you want, and then select the **Forms** area. If the item isn't in the side panel pane, select ...More and then select the item you want.

   You can also make adjustments to the tables from within a solution. To do this, select **Solutions** on the left pane, select the table, and then select the **Forms** area. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. Open the form you want to edit.

4. Select **Switch to classic** to edit the form in the classic form designer.

5. Create a JavaScript web resource and add it to the form:

   a. In the form editor, in the **Form** group, choose **Form Properties**.

   b. On the **Events** tab, below **Form Libraries** choose **Add**.

   c. In the **Look Up Row** dialog box, choose **New**.

   d. Enter the following information in the web resource form:

Expand table

Title	Reference
Name	preventAutoSave
Display Name	Prevent AutoSave
Type	Script (JScript)

   e. Next to the **Type** column, choose **Text Editor**.

   f. In the **Source** column, paste the following code:

```javascript
function preventAutoSave(econtext) {
    var eventArgs = econtext.getEventArgs();
    if (eventArgs.getSaveMode() == 70 || eventArgs.getSaveMode() == 2) {
        eventArgs.preventDefault();
    }
}
```

   g. Choose **OK** to close the text editor.

   h. Choose **Save** to save the web resource and then close the web resource window.

   i. In the **Look Up Row** dialog the new web resource you created is selected. Choose **Add** to close the dialog.

6. Configure the `OnSave` event:

   a. In the **Form Properties** window, in the **Event Handlers** section, set **Event** to **OnSave**.

   b. Select **Add**.

c. In the **Handler Properties** window, set **Library** to the web resource you added in the previous step.

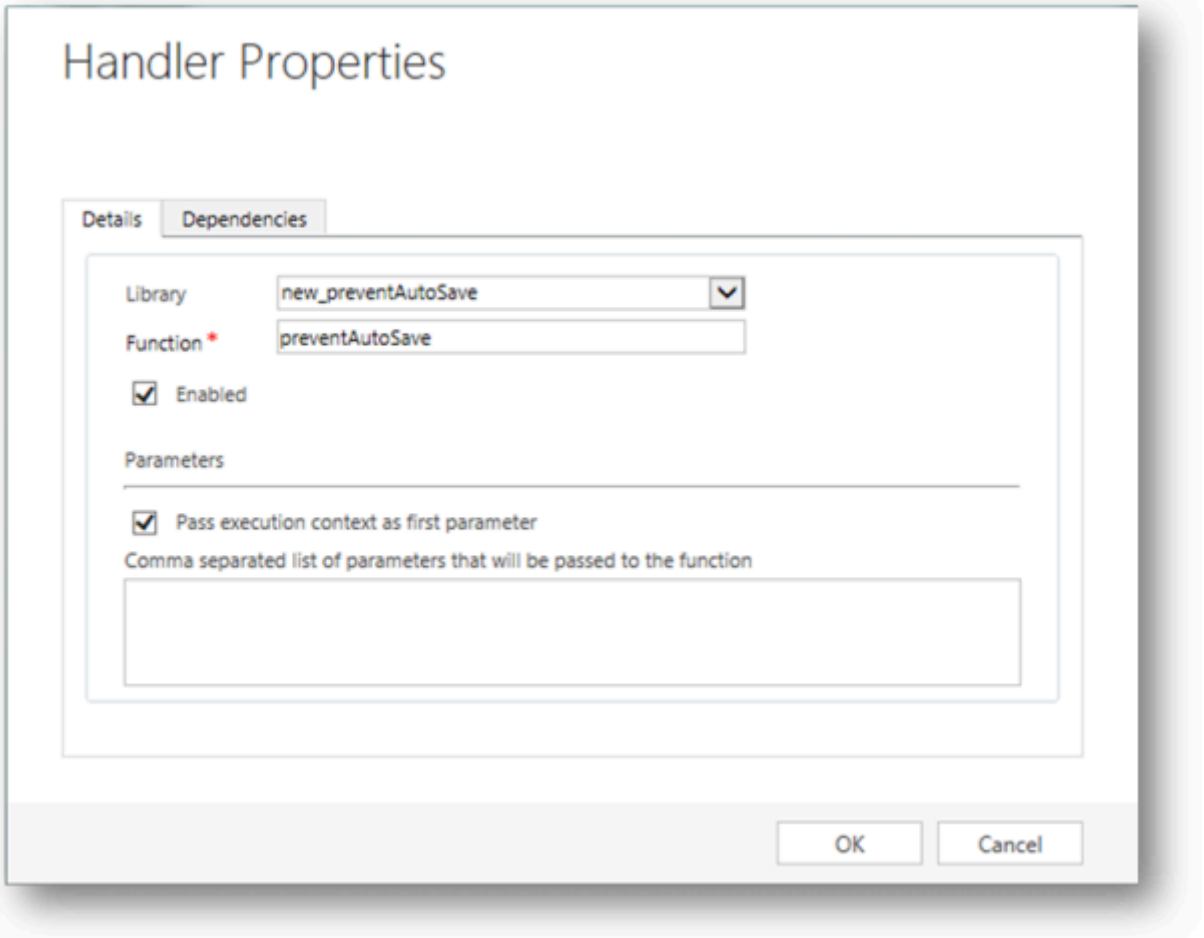
d. Type '`preventAutoSave`' in the **Function** column. This is case sensitive. Don't include quotation marks.

e. Make sure that **Enabled** is checked.

f. Check **Pass execution context as first parameter**.

> **ⓘ Important**
>
> If you don't do this the script won't work.

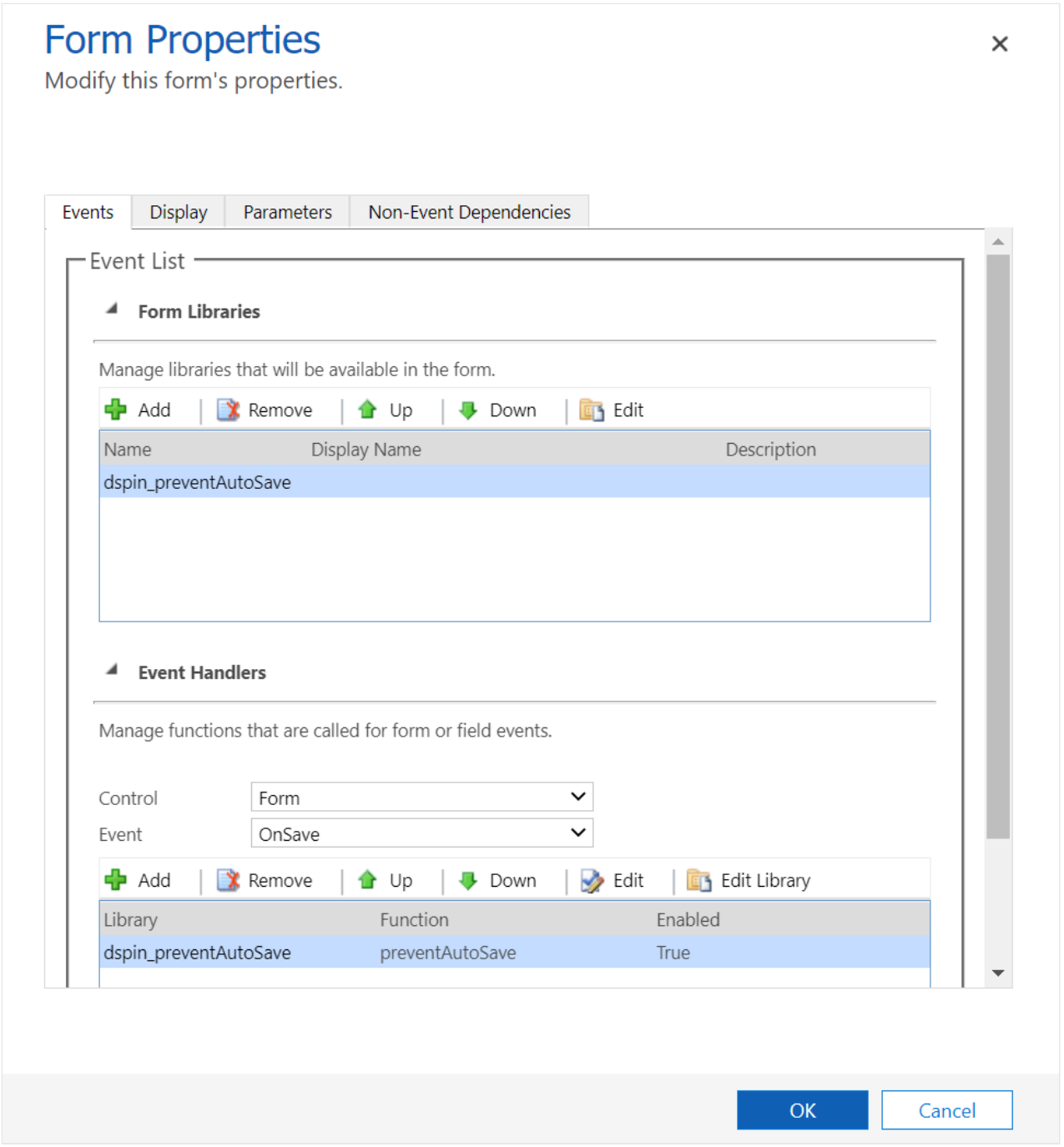
The **Handler Properties** dialog should look like this. The customization prefix: "new_" might vary based on the customization prefix set for the default publisher for your organization.



g. Select **OK** to close the **Handler Properties** dialog.

h. If there are any other event handlers for the `OnSave` event, use the green arrows to move this one to the top.

The **Form Properties** should display as follows. Note in the example shown here, the update has taken place from within a solution and as a result the publisher prefix (dspin_) isn't the same as the one provided by the default publisher (new_). This is because a different publisher is used for the solution.



7. Select **OK** to close the **Form Properties** dialog.

8. Select **Save and Close** to close the form.

9. In the solution explorer, select **Publish All Customizations**.

After you apply this script to the `OnSave` event, when users edit a row using this form the message **unsaved changes** appears in the bottom right corner of the form just as it would if

AutoSave wasn't disabled. But this message won't go away until people select the button next to it.

Next steps

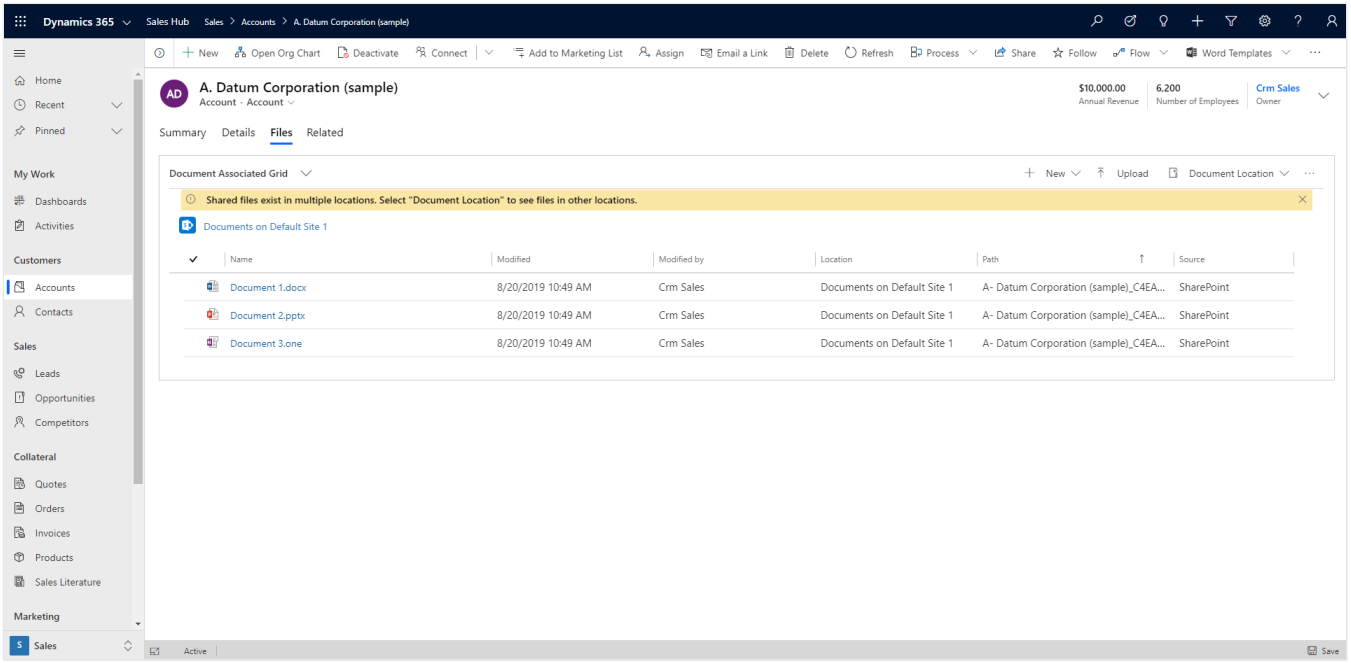
Create and design forms

Add or remove the SharePoint documents tab to the main form for any table

Summarize this article for me

[This topic is pre-release documentation and is subject to change.]

Adding a tab on a table main form to display SharePoint documents helps users discover and use the SharePoint integration features that are available in a model-driven app.



> **Important**
>
> To use this feature you must enable document management. More information: Manage documents using SharePoint

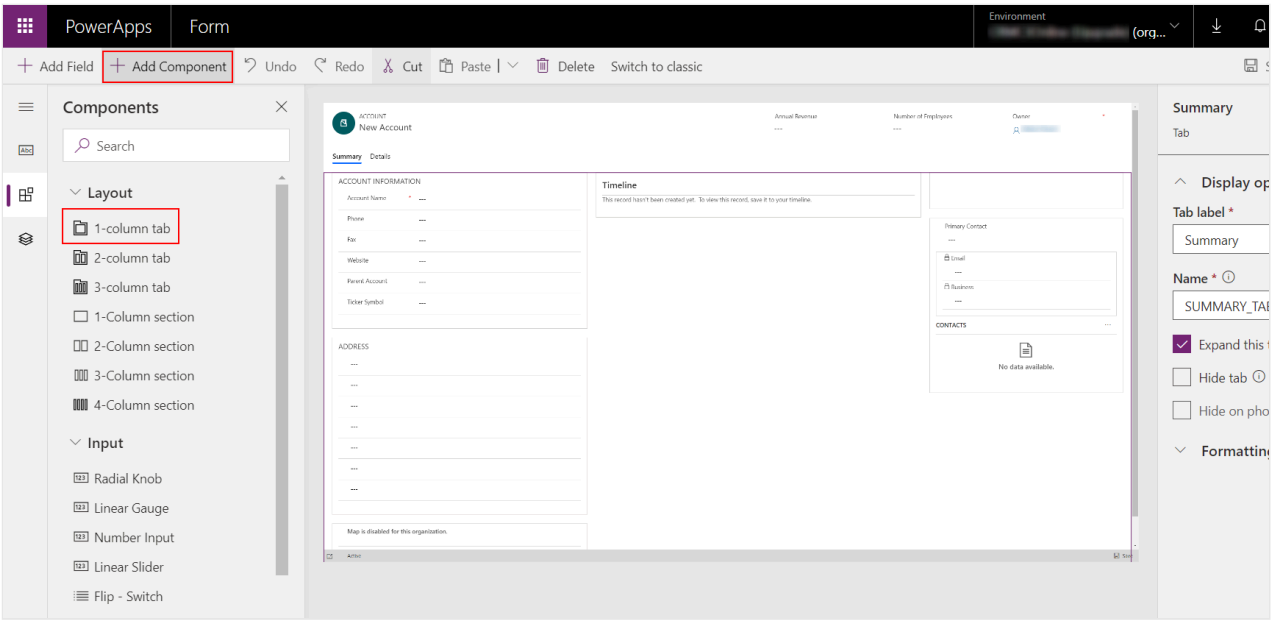
Add the documents tab in the FormXML

1. In your Power Platform development environment, create a new solution or open an existing solution. For more information about creating solutions, see: Create a solution.

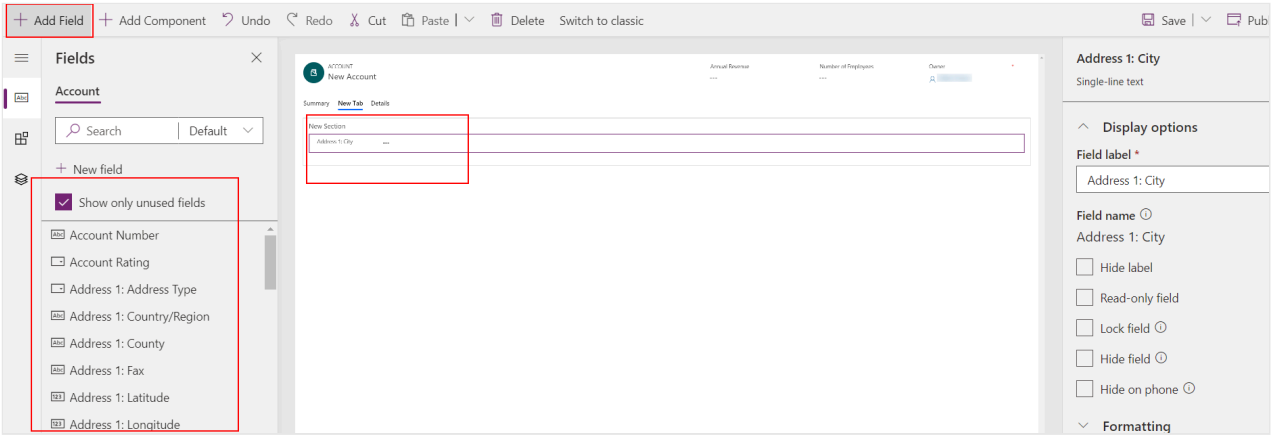
2. Add the table to the solution or select an existing table. All standard and custom tables are supported. For more information about tables in solutions, see: Add solution components

3. Include the form for the table in the solution, such as the main form for the account table. Next to the table, select ..., and then select **Edit**. Select the **Forms** area. If the required form is missing, add it.

4. Add a one-column tab to the main form. To do this, in the form designer select an area on the form canvas, select **Add Component**, and then select **1 Column Tab**.



5. In the form designer select the **New Tab** on the form designer canvas, select **Add Column**, and add a column such as *Address 1: City* from the left pane. Any text or numeric column can be used.



6. Rename the tab label. To do this, select **New Tab**, and in the right properties pane replace **New Tab** with something more descriptive, such as *Documents*.

7. Select **Save and Publish**, and then close the form designer.

8. From the Power Apps maker home page, select **Solutions**, select the solution, and the select **Export** to export the solution as an unmanaged solution. More information: Export solutions

9. Extract the solution and open the customization.xml file with an XML or text editor.

10. In the customization.xml search for **label description="Documents"** (or the name the tab was given in the previous step).

11. Scroll down to the control id="*field name*" element, such as **control id="address1_city"** and replace the entire element with the XML sample in this topic.

```
</tab>
<tab name="tab_8" id="{3283b8d3-26a9-aab9-b777-b5e5aacec3c9}" IsUserDefined="0" locklevel="0" showlabel="true" expanded="true">
  <labels>
    <label description="Documents" languagecode="1033" />
  </labels>
  <columns>
    <column width="100%">
      <sections>
        <section name="tab_8_section_1" showlabel="false" showbar="false" locklevel="0" id="{a29a4445-dd1e-335e-0075-91339b387110}" IsUserDefinec
          <labels>
            <label description="Documents" languagecode="1033" />
          </labels>
          <rows>
            <row>
              <cell id="{eb38c2ae-5ad7-6369-cbb2-6e6990e6c81d}" showlabel="true" locklevel="0">
                <labels>
                  <label description="Documents" languagecode="1033" />
                  <label description="Dirección 1: ciudad" languagecode="3082" />
                </labels>
                <control id="address1_city" classid="{4273EDBD-AC1D-40d3-9FB2-095C621B552D}" datafieldname="address1_city" disabled="false" uniqu
              </cell>
            </row>
          </rows>
        </section>
      </sections>
    </column>
  </columns>
</tab>
```

12. Make these modifications to the XML sample.

    a. Locate the **RelationshipName** element and replace it with the schema name that appears as *entityLogicalName*_SharePointDocument. For example, for the accounts table the schema name for the relationship is Account_SharePointDocument, which is the schema name for the XML sample in this topic. To find the name for a different table, go to **Settings** > **Customizations** > **Customize the System** > **Tables** > select the table > select **1:N Relationships**. Locate the **Related Table** of type **SharePointDocument**.

    ::image type="content" source="media/account-sharepointdocument.png" alt-text="Account relationship SharePoint document.":::

    b. Create a globally unique identifier (GUID) and replace the existing **uniqueid** GUID located in the **control** element pasted in the previous step while preserving the curly braces {}.
    ::image type="content" source="media/control-unique-id.png" alt-text="Control element unique id."::: c. Save the changes made to customizations.xml.

13. Open the solution.xml file and increment the **Version** element value. For example, change *1.1.0.0* to *1.2.0.0*.

14. Package all solution files into a compressed (zipped) folder and import it into the right environment. If an error appears then the previous solution should be removed. For more information on managing solutions see: Import, update, and upgrade a solution

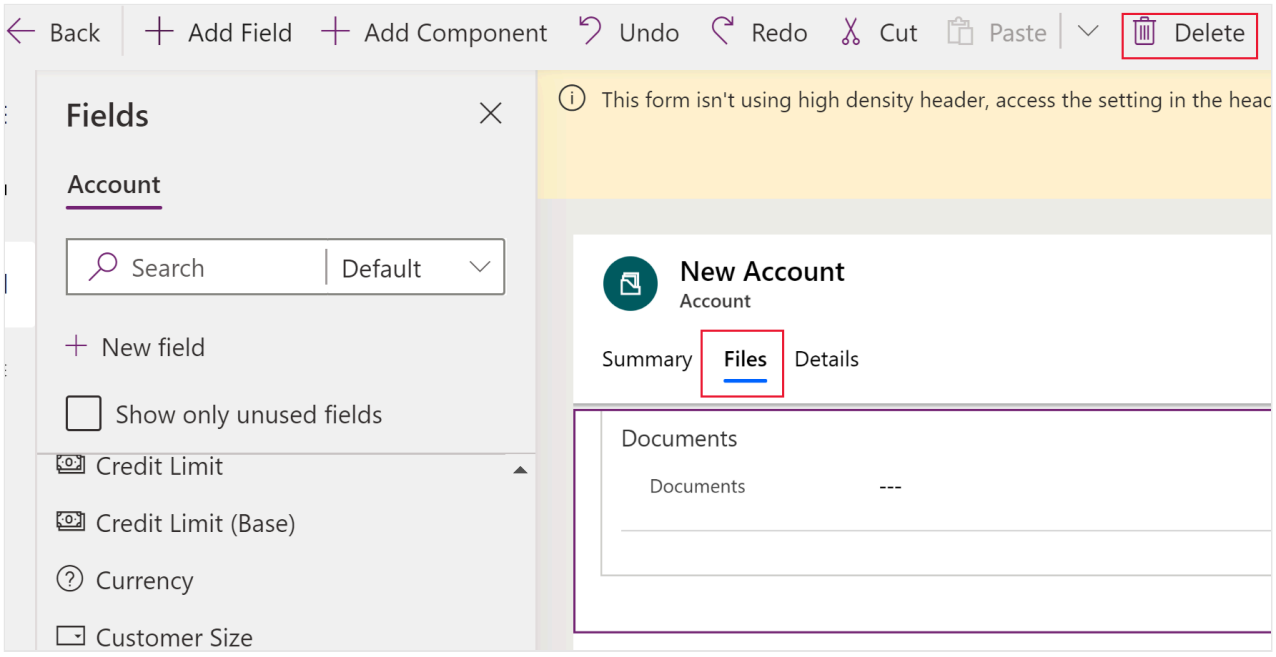
XML sample for adding the documents tab to a form

XML

```
  <control id="DocumentSubGrid" classid="{E7A81278-8635-4d9e-8D4D-59480B391C5B}"
indicationOfSubgrid="true" uniqueid="{9cd66b5c-8b7a-6433-c5a5-46a7245dd534}">
    <parameters>
      <ViewId>{0016F9F3-41CC-4276-9D11-04308D15858D}</ViewId>
      <IsUserView>false</IsUserView>
      <RelationshipName>Account_SharepointDocument</RelationshipName>
      <TargetEntityType>sharepointdocument</TargetEntityType>
      <AutoExpand>Fixed</AutoExpand>
      <EnableQuickFind>false</EnableQuickFind>
      <EnableViewPicker>true</EnableViewPicker>
      <ViewIds />
      <EnableJumpBar>false</EnableJumpBar>
      <ChartGridMode>Grid</ChartGridMode>
      <VisualizationId />
      <IsUserChart>false</IsUserChart>
      <EnableChartPicker>false</EnableChartPicker>
      <RecordsPerPage>10</RecordsPerPage>
      <HeaderColorCode>#F3F3F3</HeaderColorCode>
    </parameters>
  </control>
```

Remove the documents tab

1. Sign in to Power Apps, in the left pane expand **Data**, and then select **Tables**.
2. Select the required table and select the **Forms** area, and then open the form where the documents tab needs to be removed.
3. Select the **Documents** tab, and then on the form designer toolbar, select **Delete**.



4. On the form designer toolbar, select **Publish**.

See also

Manage documents using SharePoint

Last updated on 03/04/2026

Add model-driven app form navigation for related tables

Article • 03/13/2023

When an app user selects one of these links in a row, the associated view for the table is displayed. In this article, the form navigation pane is used to add links to related tables.

1. Sign in to Power Apps.

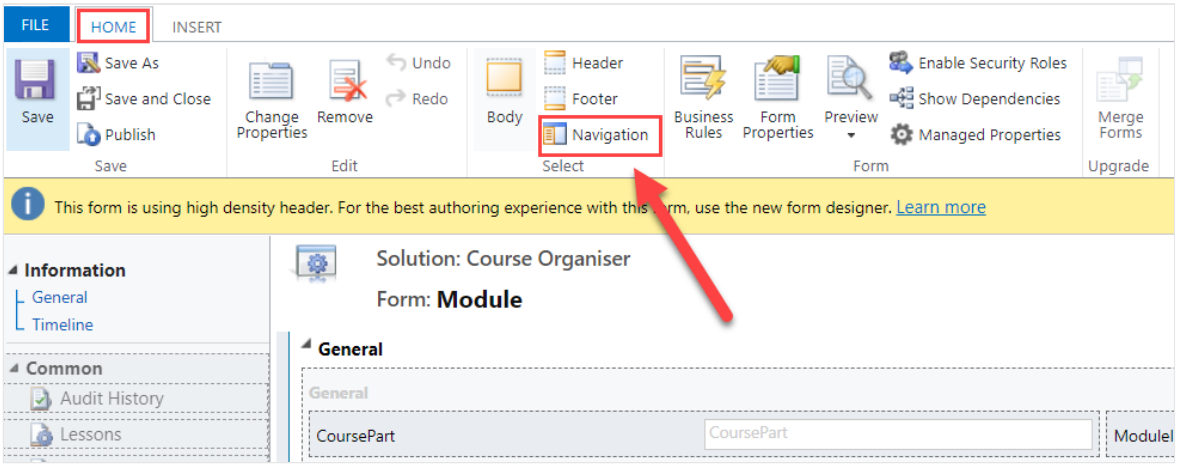
2. Select **Tables** on the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.

3. Select the table you want, and then select the **Forms** area.

4. In the list, open a form with the type of **Main** to edit it.

5. Select **Switch to classic** to edit the form in the classic form designer.

6. To add a link to a related table, on **Home** tab, in the **Select** group, choose **Navigation**.



The **Relationship Explorer** pane displays on the right side of the form editor.

7. In the **Relationship Explorer** pane, in the **Filter** list, select one of the following options:

   - **Available Relationships**. Lists all the tables that can be related to the table the form is associated with.

   - **1:N Relationships**. Lists tables that can be related in a 1:N relationship to the table the form is associated with.

- **N:N Relationships**. Lists tables that can be related in a N:N relationship to the table the form is associated with.

> **Note**
>
> If no related tables show up in the **Relationship Explorer** pane, you cannot create a link on this form to a related table.

8. Select the related table you want to link to, drag it to the Navigation Pane, and then drop it where you want it to be displayed.

> **Tip**
>
> You can also create a new relationship by choosing **New 1:N** or **New N:N** in the **Relationship Explorer** pane.

9. To edit the properties for this or any other related table link, in the Navigation Pane, select the link, and then on the **Home** tab, choose **Change Properties**.
10. In the **Relationship Properties** dialog, on the **Display** tab, type a new display label.
11. On the **Name** tab, choose **Edit** to view or edit the details associated with the relationship row.
12. Choose **OK**.
13. Preview how the main form will appear and how events will function:
    a. On the **Home** tab, choose **Preview**, and then select **Create Form**, **Update Form**, or **Read-Only Form**.
    b. To close the **Preview** form, on the **File** menu, choose **Close**.
14. When you finish editing the form, choose **Save and Close** to close the form.
15. When your customizations are complete, publish them:

    - To publish customizations for only the component that you are currently editing, in the Navigation Pane, choose the table you have been working on, and then choose **Publish**.
    - To publish customizations for all unpublished components at one time, in the Navigation Pane, choose **Tables**, and then on the command bar, choose **Publish All Customizations**.

> **Note**
>
> Installing a solution or publishing customizations can interfere with normal system operation. We recommend that you schedule a solution import when it's least disruptive to users.

Next steps

Create and edit table relationships for Microsoft Dataverse

Collaboration in model-driven apps

Article • 12/18/2023

Collaboration features help fusion teams work together to build model-driven apps. The commenting feature introduces the best way to work asynchronously with other makers or leave notes for yourself within the app context. Copresence and coauthoring help you better work with other makers in the same app in real-time.

> **ⓘ Important**
>
> Commenting is available for all model-driven app makers. For real-time collaboration, you will have either the copresence or coauthoring experience as the copresence and coauthoring features are being deployed separately:
>
> - Most preview environments will have coauthoring available. More information: **Coauthoring in model-driven apps**
> - For preview environments that don't have coauthoring available, the environment will use copresence. More information: **Discover who's working on an app**

Next steps

Add comments in model-driven apps
Coauthoring in model-driven apps (preview)
Discover who's working on an app (preview)

Add comments in the model-driven app designer

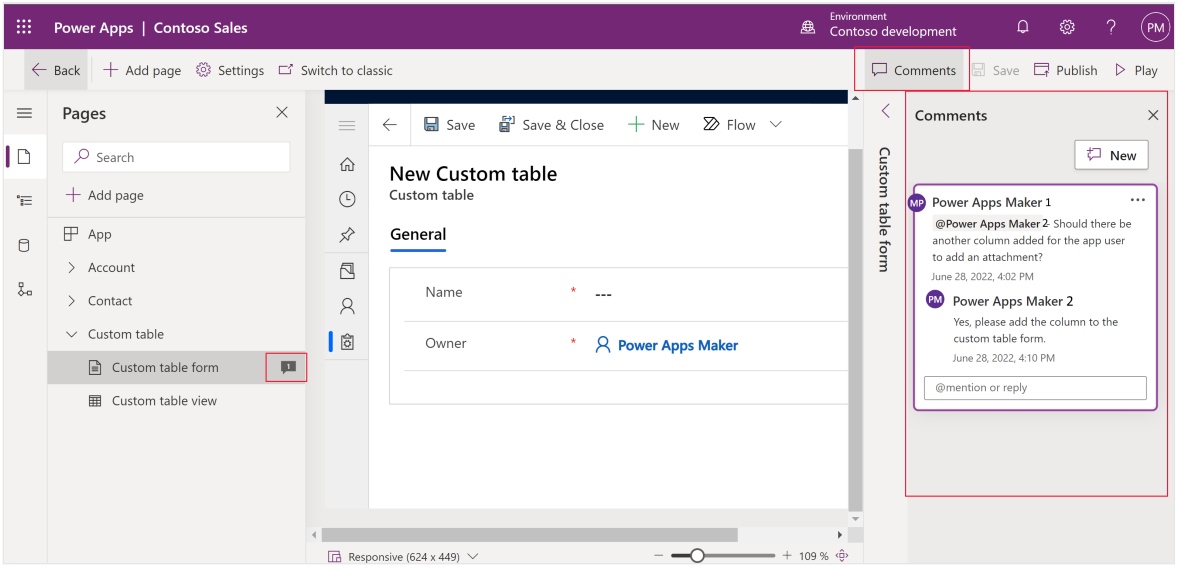
Article • 03/29/2023

Comments are notes that are associated with items in your app. Use comments to help your team review the app and provide feedback, or provide additional information on implementation details in your app.

In the app designer, the anchors, which are what comments attach to, are all of the pages in the pages pane, and any of the items in the **Navigation** pane. This lets you add comments to items such as areas, groups, pages, tables, and dashboards. Comments are stored in a table in Microsoft Dataverse in the default solution.

Add a comment to a page

1. Open a model-driven app for edit using the app designer.
2. Select an item in the **Pages** pane.
3. Select **Comments** on the command bar. This opens the comments pane for the selected component. You can also open the comments pane by selecting the comment count indicator for existing comments to an item in the left pane.



4. Select **New** to create a new comment and type the comment message in the box. Or, type into the **Reply** box to respond to an existing comment. Select **Post reply** or press Ctrl + Enter to post your comment.

Edit or delete a comment

You can edit your comments or remove existing comments from appearing in the app.

In the **Comments** pane, next to a comment, select **...**, and then select **Edit comment** or **Delete comment**.

Resolve a comment

Comments are commonly questions, feedback, or ideas that are important context to live alongside your app. But, they may not be active discussions anymore. You can resolve, or reopen a resolved, comment thread to better track the active comments.

1. In the **Comments** pane, next to a comment, select **...**, and then select **Resolve comment**.
2. The comment thread appears as **Resolved**.
3. To reopen select **Reopen thread** or **Delete thread** to remove the comment.

Moving comments between environments

Comments in model-driven apps can't be exported or imported as solution components because they contain user collaboration data. You can, however export the **Comment** table to Excel and then import the Excel file into the environment you want. More information: Export data

Next steps

Discover who's working on an app
Coauthoring in model-driven apps (preview)

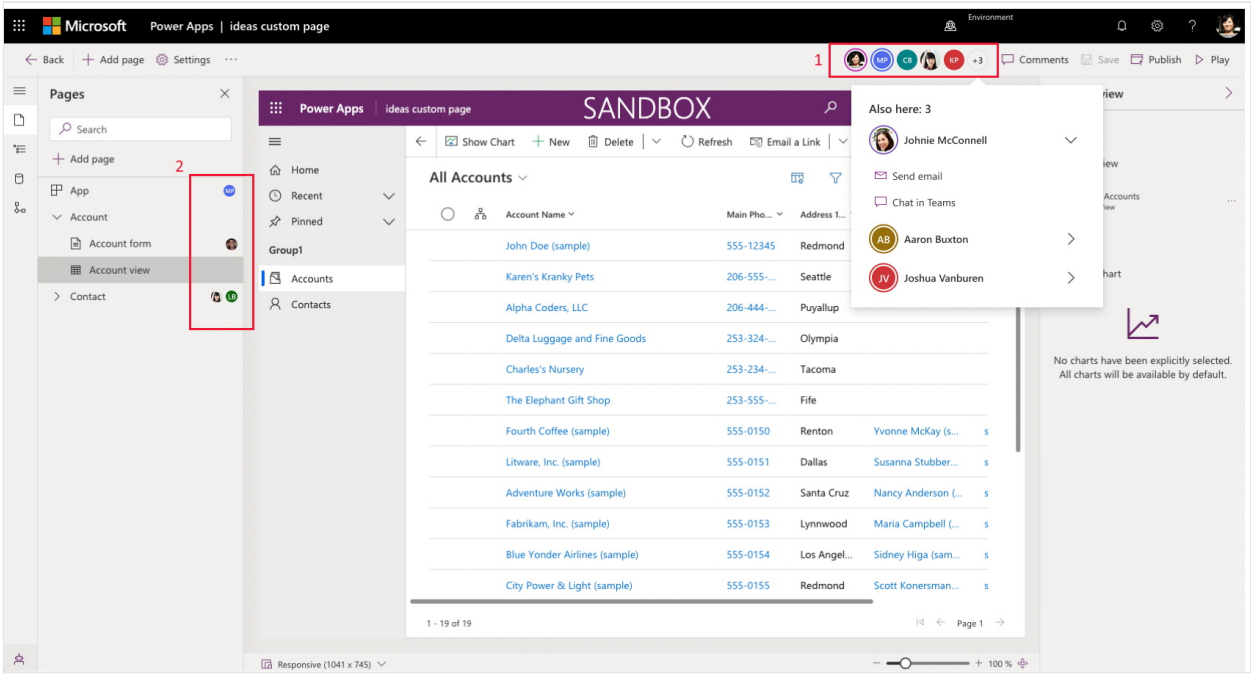
Discover who's working on an app

Article • 09/05/2023

Find who's working on an app at the same time as you with copresence. Copresence provides the following benefits:

- Icons and names of people who have the app open and might be making changes are displayed.
- Contact those people working on the app via email or Teams.
- On the left navigation pane, the structure of the app is displayed letting you know what part of the app is being worked on.
- Receive notification for saved changes made from other makers. Then, you'll be prompted to refresh the app to stay on the latest version so as not to lose your work or overwrite other's work later. More information: Notifications
- The name of the maker who made the first save or the last save is displayed.

> **Important**
>
> Most Power Platform environments have **coauthoring**. In case your tenant doesn't have coauthoring yet, you'll have the copresence experience that is described in this article.



1. Command bar displays makers currently working in the app and allows you to chat or email them.
2. Left navigation indicates what app component the maker is working on.

How copresence works

The first time someone joins your app in app designer while you working on it a notice appears that indicates other people are also working on the app. Advice is given to check with the other makers about the order you plan to make changes. Notice that it's possible that the other makers in the app might not want to save their changes.

Icons of copresent makers appear in both the toolbar area and the left navigation pane showing where other makers are working in the app.

When you have several tabs open, your icon is displayed in the left navigation pane as well. This way you'll always see if you have more than one tab open, which helps you find the one you want to continue working on.

On the right side of the app designer toolbar, all the makers that have this app open are displayed. If more than four makers are present, a +n, such as +1 or +30, is displayed that expands with all the makers' names. Select a name to display **Send email** and **Chat in Teams** links to contact that person.

Notifications

You may be working on the app, or you may be idle, but once someone else saves a change to the app, you'll receive a notification in the upper bar indicating that another maker made changes. When this occurs, you're advised to refresh to be at the latest version so as not to lose changes you make afterwards.

Tip

- If you weren't expecting anyone to make changes to the app, and spent a long time adding changes, we recommend you contact the person immediately. For example, it might happen that another maker saved something minor and you can agree that your changes may overwrite their changes. In this case, do not select refresh, instead select save to present the option to overwrite changes.
- We recommend that you to always stay at the latest version of the app and refresh when others save changes. If you were idle and didn't make changes and see this notification, select refresh.

If you've ignored or closed the refresh notification, you won't be at the latest version of the app. At the time when you select save, you're prompted with a choice: refresh (and

lose your changes) or overwrite (and lose other maker's changes).

> **Tip**
>
> We recommend you chat with the team or the person who made the last change. Copresence provides the name of the maker to alert you before you overwrite their changes.

Known copresence limitations and issues

- There may be a delay up to 20 seconds between when new members join the app and when you see their presence and location.
- There may also be a delay up to one minute between when someone saves a change and when you get a notification about the change. However, if there's a conflict, you'll always get the prompt to choose whether to refresh or overwrite when you try to save.
- Making a comment on a new page or area triggers a background save and if your app isn't on the latest version you'll be prompted to either refresh or overwrite.
- Instead of copresence, your environment may have coauthoring, which allows app change merges to occur in real-time with seamless collaboration. More information: Coauthoring in model-driven apps (preview)

See also

Collaboration in model-driven apps

Coauthoring in model-driven apps

Article • 11/13/2024

Coauthoring allows multiple makers—whether pro or citizen developers—to make changes to the app at the same time and see those changes in real time. As a result, teams can innovate together and build stronger connections between professional developers and business domain experts, a critical ingredient to maximizing low-code development.

> **Important**
>
> - This feature is being gradually rolled out across regions and might not be available yet in your region.
> - This feature isn't currently available in Germany, China, and Government Cloud regions. Regions where coauthoring isn't available have copresence. More information: Discover who's working on an app

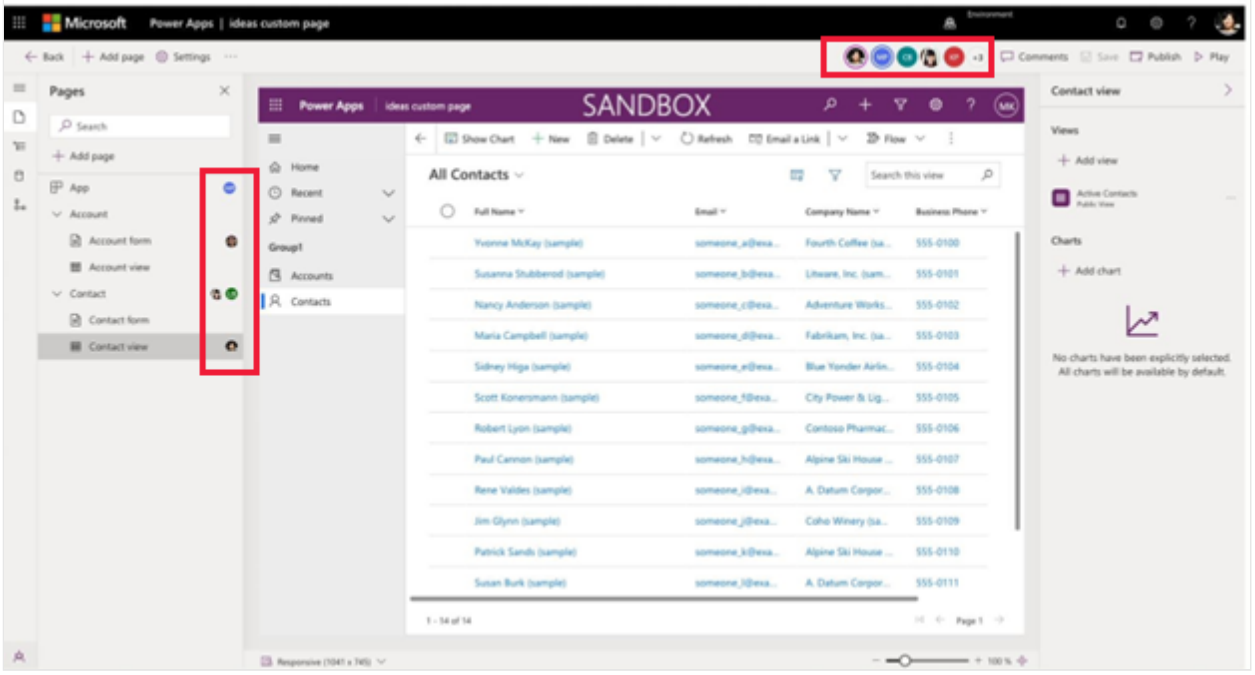
What you can coauthor in an app

Here's what multiple makers can do at the same time in the app while seeing each other's others changes in real-time.

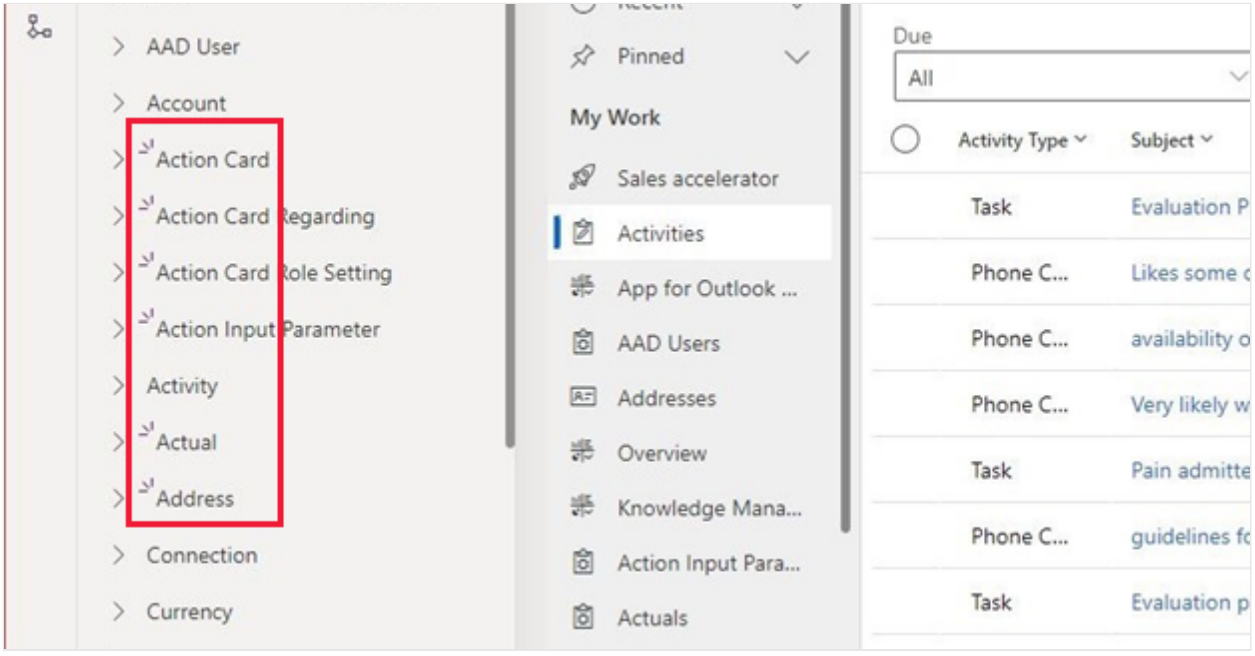
- Add and remove components, such as tables, dashboards, and custom pages, to and from the model-driven app.
- Add and remove subcomponents, such as views, forms, and charts, to and from the tables in the model-driven app.
- Add and remove areas, groups, and subareas to and from the navigation bar in the model-driven app.
- Change properties of the model-driven app.
- Change properties of the navigation pane.
- Change properties of areas, groups, and subareas in the navigation pane, including **Display Options** and **Advanced Settings**.
- Change order of the areas, groups, and subareas in the navigation pane.
- Change **General**, **Features** and **Upcoming** tabs in the **Settings** dialog.

How to know you're coauthoring

First, once there are other makers editing the app together with you, you see their presence in the top navigation bar and the left menu. You can see what components other makers are working on or what pages they have open. You'll also see them move through the components on the left navigation bar in real-time.



When someone is adding or removing a page, all other makers will see it appearing on the left navigation bar with a sparkle graphic indicator. The sparkle remains displayed until you select that page. You won't see a sparkle for the components and pages that you've added.



Once someone removes the page, it disappears for other makers in real-time.

Limitations

- Currently, there's no auto-save functionality, so you need to select the **Save** button to save changes. However, any maker that has access to the app can save the changes for others, as their changes are synced. Unsaved changes will be removed from the app in eight hours.
- Under **Settings**, changes to the **Features** and **Upcoming** sections must be saved from the settings dialog in order to be set. Users that don't see the changes immediately can close and open the settings dialog to see the most up-to-date settings.
- Changes made in other designers, such as the view and form designers, won't be synchronized in real-time in the modern app designer.

See also

Add comments

Collaboration in Canvas Apps with Power Apps (video)

Feedback

Was this page helpful? Yes No

Provide product feedback

Apply custom business logic by using business rules and flows in model-driven apps

Summarize this article for me

Defining and enforcing consistent business processes is one of the main reasons to use model-driven apps. Consistent processes help ensure that people using a model-driven app can focus on their work and not on remembering how to perform a set of manual steps to reach their objectives.

Business logic falls into two major categories natively within model-driven apps: business rules and business process flows.

Business rules

Business rules provide a simple interface to implement and maintain fast-changing and commonly used rules. The *scope* of a business rule defines where the business rule runs:

Expand table

If you select this item...	The scope is set to...
Table	All forms and server
All Forms	All forms
Specific form (**Account** form, for example)	Just that form

For more information about defining business rules for a form in a model-driven app, go to Create business rules to apply logic in a model-driven app form.

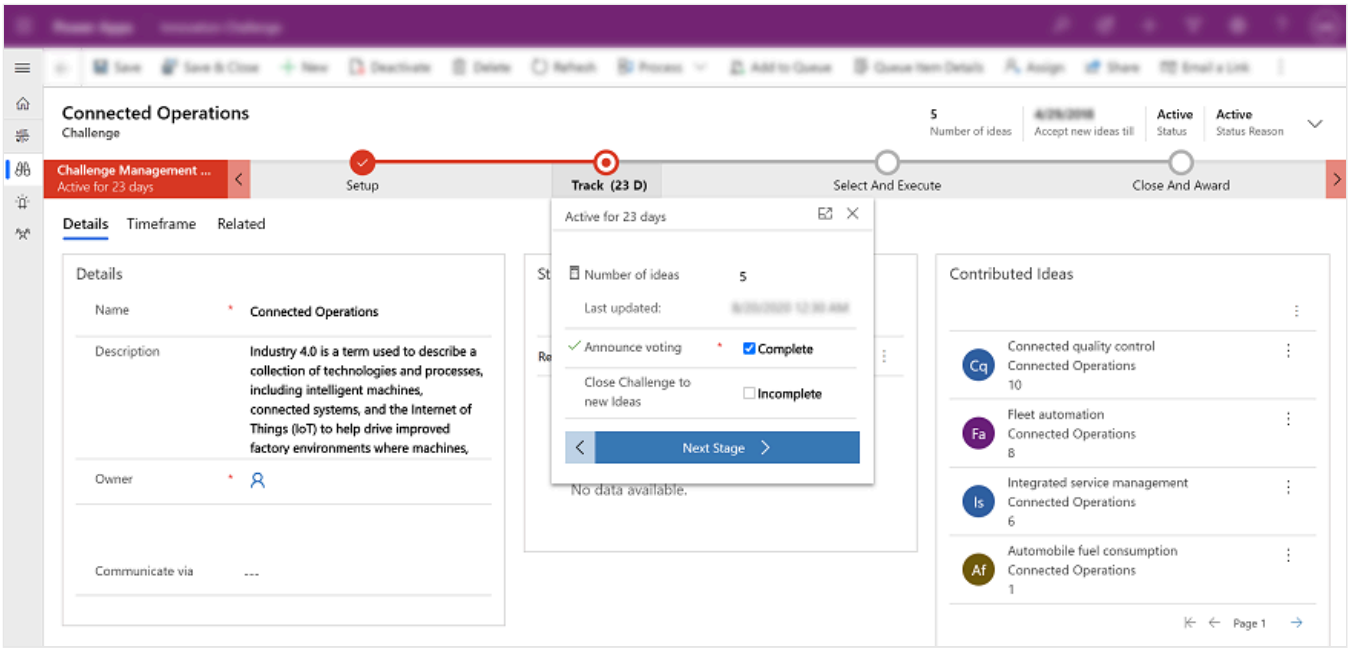
> **Note**
>
> To define a business rule for a table so that it applies at the server level to both *canvas apps* and *model-driven apps*, go to Create a business rule for a table.

Business process flows

By creating a business process flow, you ensure that people enter data consistently and follow the same steps every time they work in an app. These flows are authored by using the business process flow designer and only work with model-driven apps. For more information, go to Business process flows overview.

Power Automate flows

Power Automate includes several types of processes, each designed for a different purpose that can be designed to automate tasks in Dataverse or many other data sources to enhance model-driven apps:



- Workflows and actions. Dynamics 365 customizers might be familiar with the classic Microsoft Dataverse processes, which are workflows and actions. More information: Use Workflow processes and Actions overview

- Automated flows. Create a Power Automate cloud flow that performs one or more tasks automatically after it's triggered by an event. These flows have the added advantage of being able to reach into the connector framework within your environment and connect to other data sources, such as SQL, SharePoint, or any of the other 400 data sources and services. More information: Create a flow

- Button flows. Create a Power Automate cloud flow that performs repetitive tasks simply by tapping a button on your mobile device. More information: Introducing button flows

- Scheduled flows. Create a flow that performs one or more tasks on a schedule, such as once a day, on a specific date, or after a certain time. More information: Run flows on a schedule

Next steps

Create business rules to apply logic in a model-driven app form

Learn about business process flows

Apply business logic with Dataverse

Last updated on 02/21/2026

Create business rules to apply logic in a model-driven app form

Often, you need to add business logic to ensure that columns in a model-driven app are shown, hidden, or set with the correct values.

This article shows how to create business rules and recommendations to apply form logic in a model-driven app without writing JavaScript code or creating plug-ins. Business rules provide a simple interface to implement and maintain fast-changing and commonly used rules. They can be applied to main and quick create forms. Business rules work in model-driven apps, legacy web apps, Dynamics 365 for tablets, and Dynamics 365 for Outlook (online or offline mode).

By combining conditions and actions, the following actions are possible with business rules:

Expand table

Action	Description	Applies to
Set Field Value	Sets a value of a table column. You can set the value to a hard-coded value, to the value of another column, or with a simple formula. You can also clear the value of the field.	All scopes
Set Default Value	Sets the value of a column but only when the column has no value (null).	All scopes
Show Error Message	Prevents the save of the row. In model-driven app form, the error message is displayed within the form; for server-side, the error message is passed back to the calling process.	All scopes
Lock/Unlock	Lock changes the column's property to read-only. Unlock enables the user to change the value of the column.	Model-driven app
Set Visibility	Setting to No hides the column on the form. Setting to Yes displays the column on the form	Model-driven app
Set Business Required	Sets the requirement level of the column on the form to Business Required or Not Business Required (Optional). Setting the column to Business Required shows a red asterisk next to the column's label and prevents the rows from being saved if the column's value is empty.	Model-driven app
Recommendation	Shows a light bulb icon next to the column's label. When the icon is selected, the user is prompted. If prompt is accepted, the rule can set field	Model-driven

Action	Description	Applies to
	values.	app

> **Note**
>
> To define a business rule for a table so that it applies to all forms, see Create a business rule for a table.
>
> Business rules don't work with multiselect choices.
>
> Actions that only apply to model-driven apps are ignored when the rule is run server-side.

View, edit, or create a business rule in Power Apps

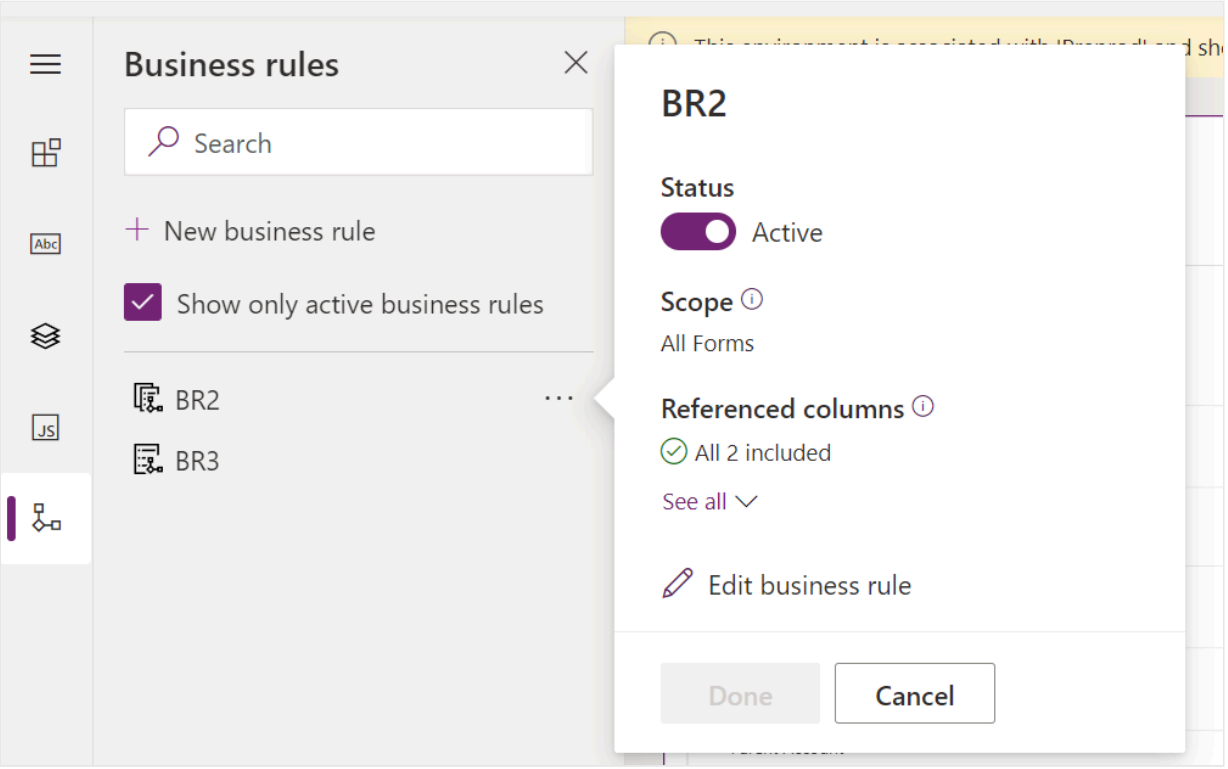
1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, open the solution you want, and then open or add the table you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **New**, and then under **Customizations**, select **Business rule**.

The business rules designer opens. Next step: Set the business rule properties

View, edit, or create a business rule in form designer

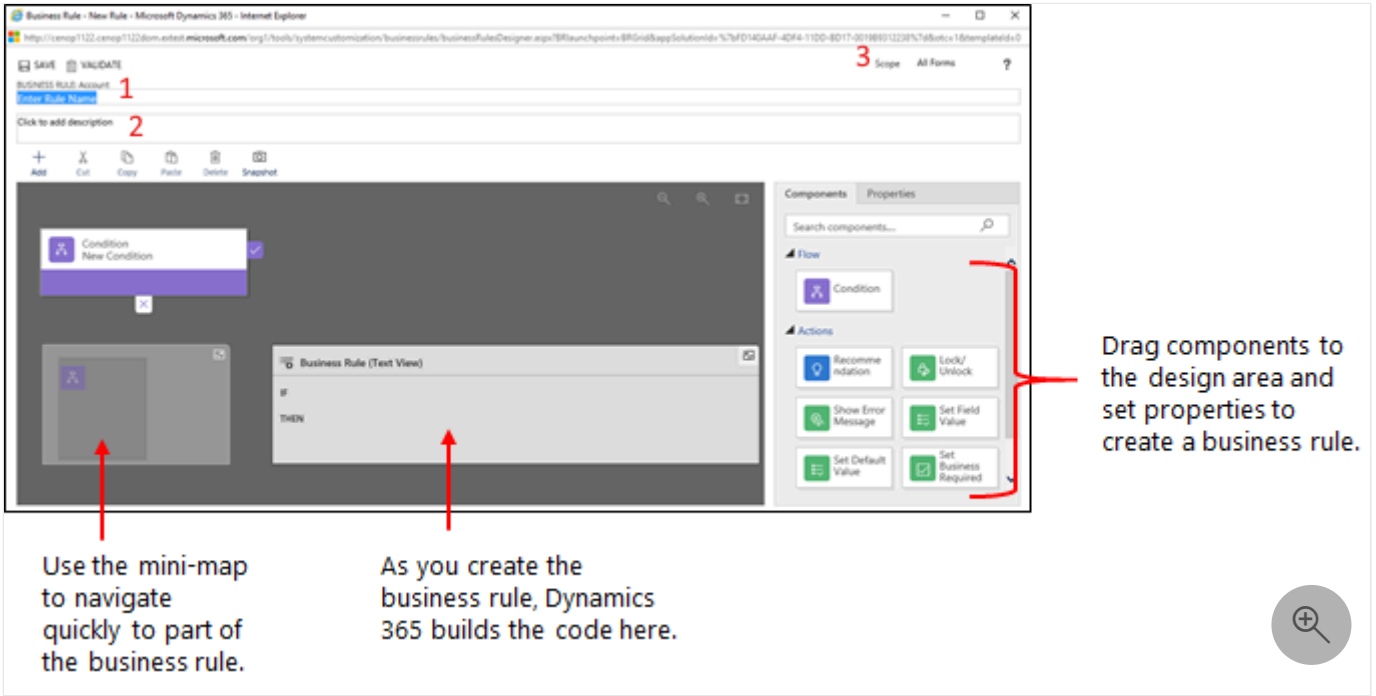
1. On the form designer left navigation pane, select **Business rules**.
2. Select from these options:
   - By default, only active business rules for the form are displayed. Clear **Show only active business rules** to view deactivated business rules for the form.
   - Select **New business rule** to create a business rule.

- Select the business rule to view properties and edit the business rule.



Set the business rule properties

When the business rule designer window opens for a new rule, a single condition is already created. Every rule starts with a condition. The business rule takes one or more actions based on that condition.



Enter or select the following properties from the business rule command bar.

1. **Business rule name**: By default, the rule is named *New business rule*. Enter a meaningful name for the business rule.
2. **Description**: Optionally, add a description.
3. **Scope**: The scope of the business rule determines which forms the business rule is applied. You set the scope, according to these options:

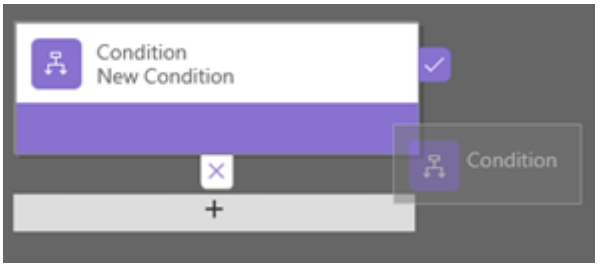
If you select this item...	The scope is set to...
Entity	The table and all forms for the table
All Forms	All forms for the table
Specific form (account **Main Form**, for example)	Just that form

> **Note**
>
> To modify an existing business rule, you must deactivate it before you can modify it.

Build the business rule

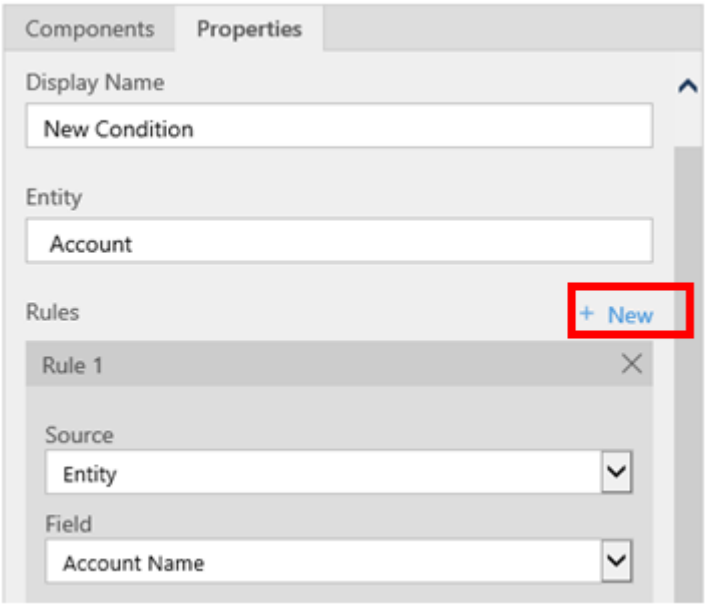
1. **Add conditions.** To add more conditions to your business rule:

   a. Drag the **Condition** component from the **Components** tab to a plus sign in the designer.



b. To set properties for the condition, select the **Condition** component in the designer window, and then set the properties in the **Properties** tab on the right side of the screen. As properties are set, an expression is created at the bottom of the **Properties** tab.

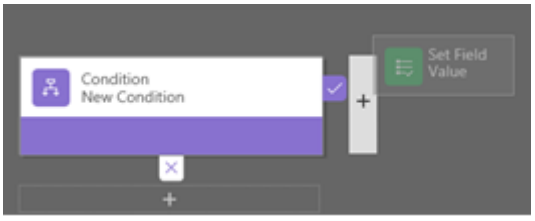
c. To add an additional clause (an AND or OR) to the condition, select **New** in the **Properties** tab to create a new rule, and then set the properties for that rule. In the **Rule Logic** column, you can specify whether to add the new rule as an AND or an OR.



d. When you finish setting properties for the condition, select **Apply**.

2. **Add actions.** To add an action:

a. Drag one of the action components from the **Components** tab to a plus sign next to **Condition** component. Drag the action to a plus sign next to a check mark if you want the business rule to take that action when the condition is met, or to a plus sign next to an x if you want the business rule to take that action if the condition isn't met.



b. To set properties for the action, select the **Action** component in the designer window, and then set the properties in the **Properties** tab.

c. When you finish setting properties, select **Apply**.

3. **Add a business recommendation.** To add a business recommendation:

a. Drag the **Recommendation** component from the **Components** tab to a plus sign next to a **Condition** component. Drag the **Recommendation** component to a plus sign next to a check mark if you want the business rule to take that action when the condition is met, or to a plus sign next to an x if you want the business rule to take that action if the condition isn't met.

b. To set properties for the recommendation, select the **Recommendation** component in the designer window, and then set the properties in the **Properties** tab.

c. To add more actions to the recommendation, drag them from the **Components** tab, and then set properties for each action in the **Properties** tab.

> **Note**
>
> When you create a recommendation, a single action is added by default. To see all the actions in a recommendation, select **Details** on the **Recommendation** component.

d. When you finish setting properties, select **Apply**.

4. To validate the business rule, select **Validate** on the action bar.

5. To save the business rule, select **Save** on the action bar.

6. To activate the business rule, select it in the Solution Explorer window, and then select **Activate**. You can't activate the business rule from the designer window.

> **Tip**
>
> Here are a few tips to keep in mind as you work on business rules in the designer window:
>
> - To take a snapshot of everything in the **Business Rule** window, select **Snapshot** on the action bar. This is useful, for example, when getting feedback from team members.
> - Use the mini-map to navigate quickly to different parts of the process. This is useful when working on a complicated process that scrolls off the screen.
> - As you add conditions, actions, and business recommendations to your business rule, code for the business rule is built and appears at the bottom of the designer window. This code is read-only.

Localize error messages used in business rules

If more than one language is provisioned for your environment, it's worth localizing any error messages that are set. Each time a message is set, a label is generated by the system. If the translations in your environment are set, localized versions of your messages can be added and then imported back into the system. This enables people using languages other than your base language to view the translated messages.

Common issues

This section describes common issues that might occur when you use business rules.

Full Name column and Address column not supported with Unified Interface apps

Actions or conditions that use a composite column like the **Full Name** `fullname` column or an **Address** column aren't supported in apps based on Unified Interface. Alternatively, you can use actions or conditions with the constituent columns. For example, for the **Full Name** column, you can use actions or conditions on the **First Name** (firstname) and **Last Name** (lastname) columns.

Business rules don't execute for some users

Make sure that users have a security role that includes, at a minimum, user scope read privileges on the Process table. By default, the Basic User security role has this privilege.

Business rules don't fire on editable grid on a dashboard

Entity scoped business rules won't fire on an editable grid when the editable grid is configured on a dashboard page.

Is the business rule not firing for a form?

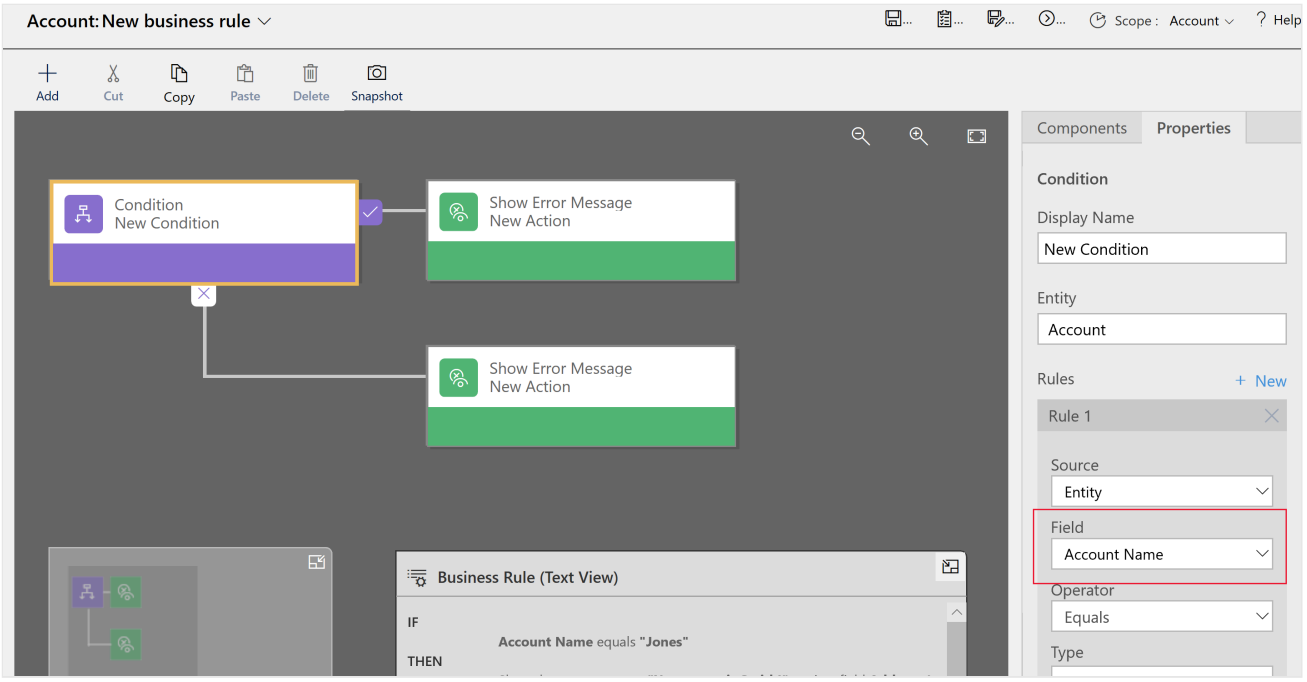
A business rule might not execute because the column referenced in the business rule isn't included with the form.

1. Go to the **Solutions** area in Power Apps and then open the solution you want. Expand the table that you want, and then select **Forms**.

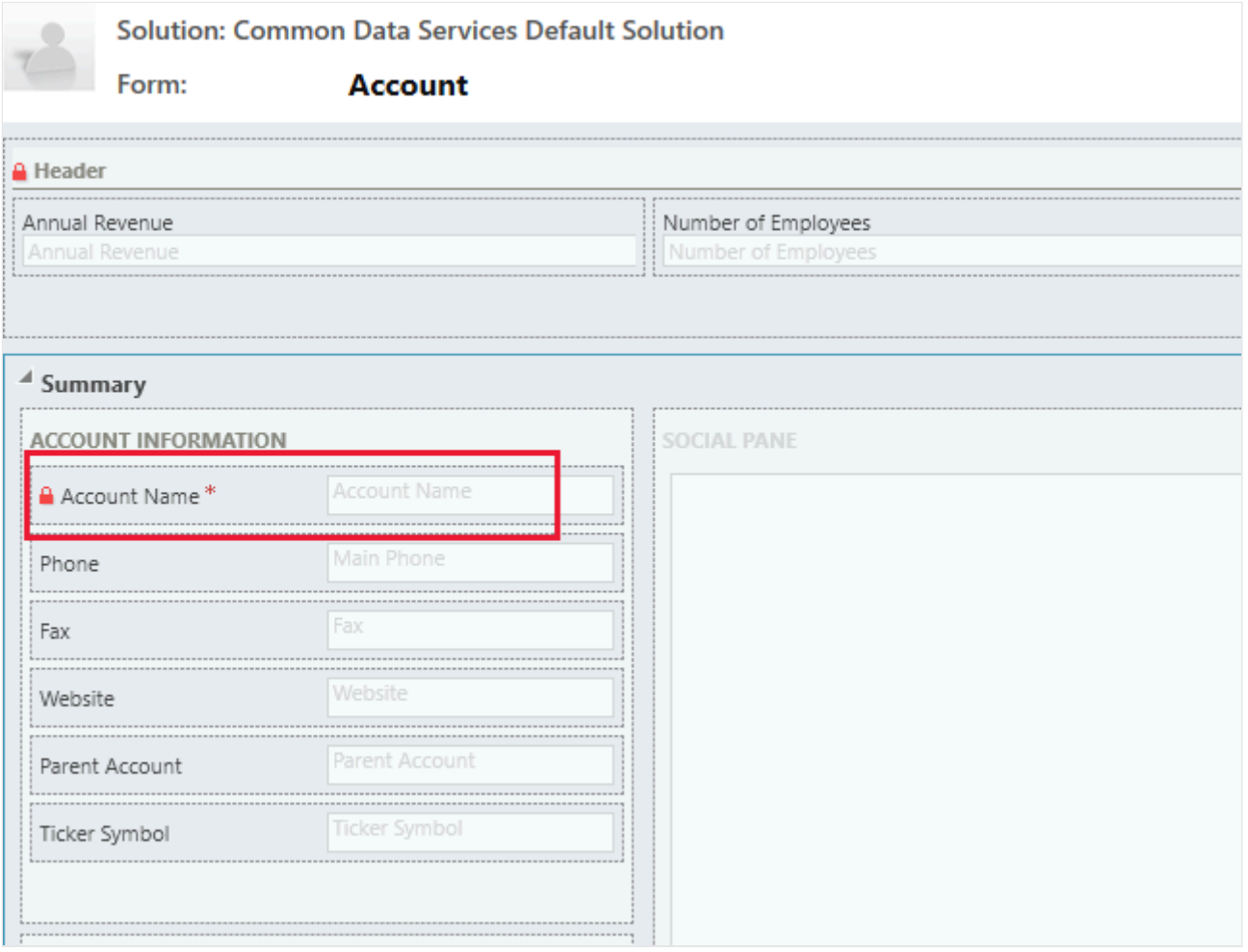
2. Open the form, and then on the form designer ribbon select **Business Rules**.

3. In the form designer, open the business rule.

4. In the business rule designer, select each condition and action to verify all the columns referenced in each condition and action.



5. Verify that each column referenced in the business rule is also included on the form. If not, add the missing column to the form.



A business rule might also not execute because a column referenced in the business rule is a composite column. Use the constituent columns of the composite column instead.

Unsupported column types

Columns of type unique identifier and rollup columns aren't supported with business rules.

A few special columns support localizable values, such as the **Name** column for the **Product** table that's included with Dynamics 365 for Sales apps. Columns that support localizable values aren't supported with business rules.

Frequently asked questions (FAQ)

Can business rules unlock columns on a read-only form?

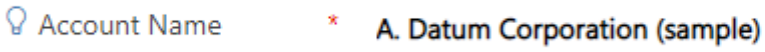
- Yes, a business rule can unlock columns and edit actions on a read-only form.

How do I troubleshoot a business rule that isn't working?

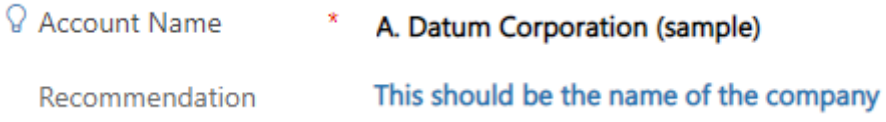
- See Is your business rule not firing for a form? in this article.

What do recommendations look like within a form?

Recommendations show a light bulb next to the column label.

- 

Select the light bulb to expand the view and show the recommendation.

- 

See also

Create custom business logic through processes

Create a business process flow

Last updated on 04/20/2026

Reporting overview for model-driven apps

Article • 08/10/2023

Microsoft Dataverse as a platform supports interaction with many millions of records, filtered down to the relevant security roles. Dataverse presents data in a way that permits the correct record to be found in a reliable fashion.

Model-driven apps provide a range of reporting options to enable users to understand the data associated with their business area more easily.

Some types of reporting are straightforward to implement, and others require more advanced techniques.

Simple reporting techniques

All of the techniques described here are native to Dataverse. They provide a simple method by which application lifecycle management can be applied.

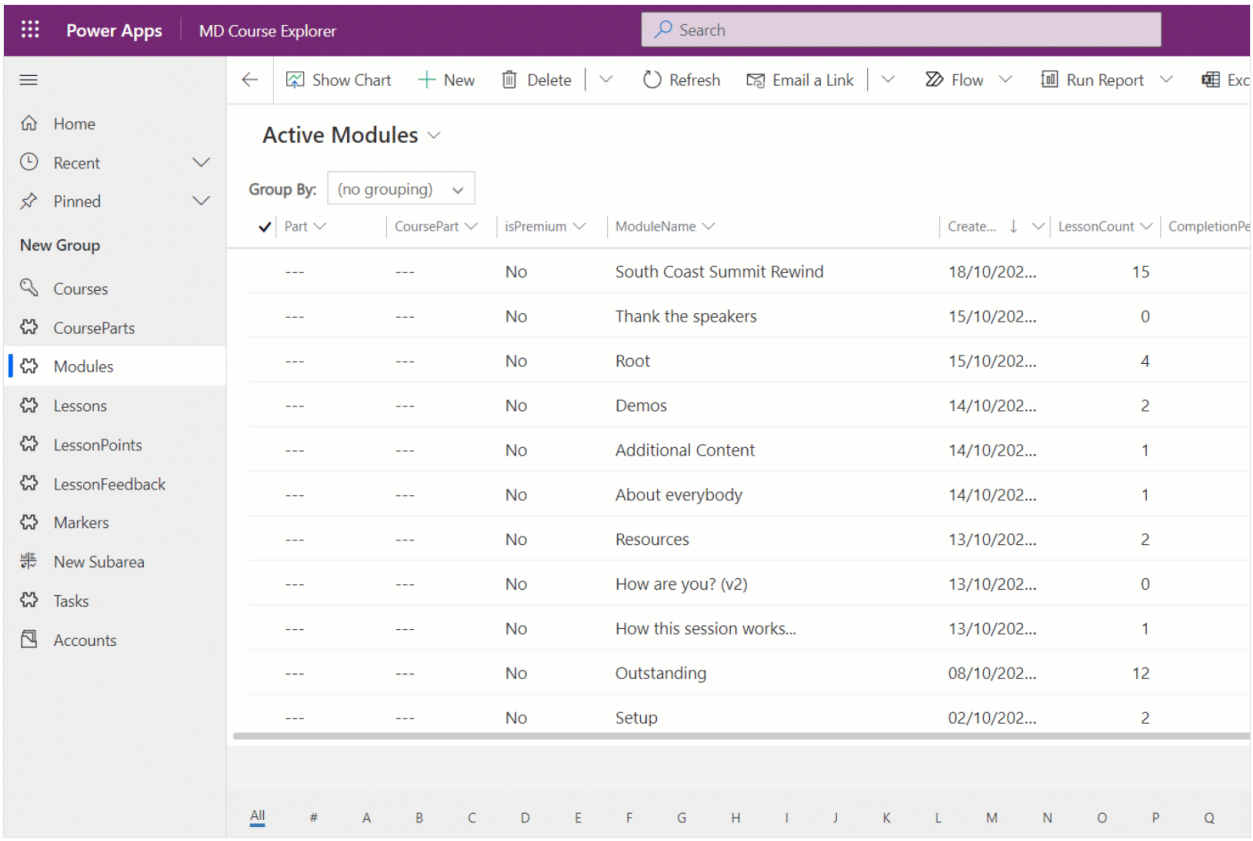
Table views

At the most basic level, table views offer a straightforward native experience in which records are presented in a tabular format. It's easy to overlook the power of views. A table can have multiple views, all of which have individual sorting and filtering applied to them, which offers a simple set of review options to the user.

Charts

Charts allow app users to view the data presented in a view in a range of ways, which include bar, pie, and line charts. Once again, a table can have multiple charts associated with it, and the charts themselves provide interactivity that enables a user to select a bar and the view filters accordingly.

The extract here shows views and charts used to create an efficient user experience.

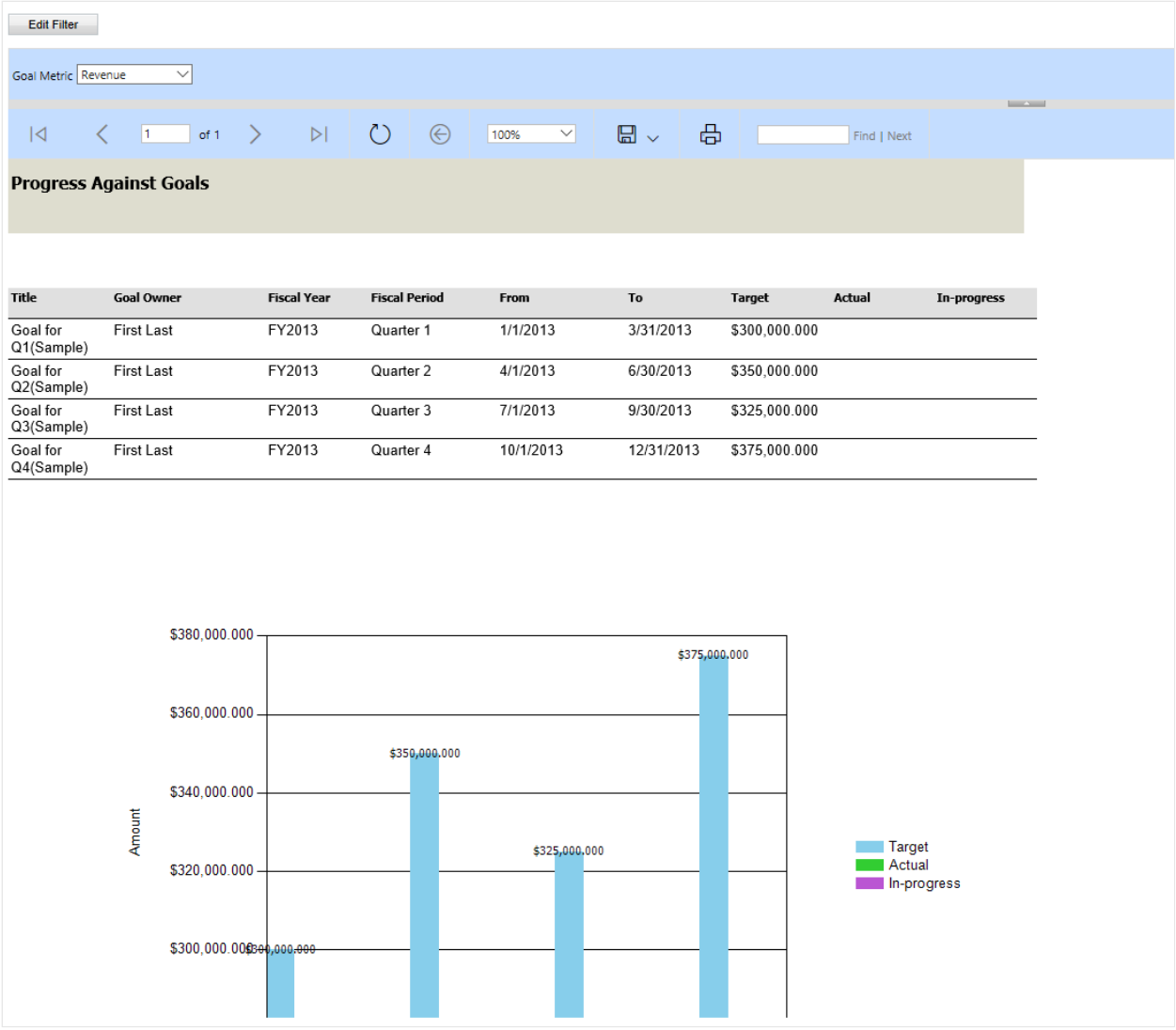


Forms

Forms can often be thought of as a method of entering data for a record. However, forms are an excellent way of providing the user columns of metadata associated with a record. Similar to views and charts, a table can have multiple forms. This permits forms to be developed that are most suited to the business needs of the individual.

Working with SQL Server Reporting style reports

SQL Server Reporting Services (SSRS) reports allow pixel perfect paginated reports to be created that can be rendered on screen and distributed in a range of ways such as PDF, CSV, Excel, and so on. The reports are created using the classic editor and are solution aware, which helps with application lifecycle management. Learn more about SSRS reports



Power BI Reporting

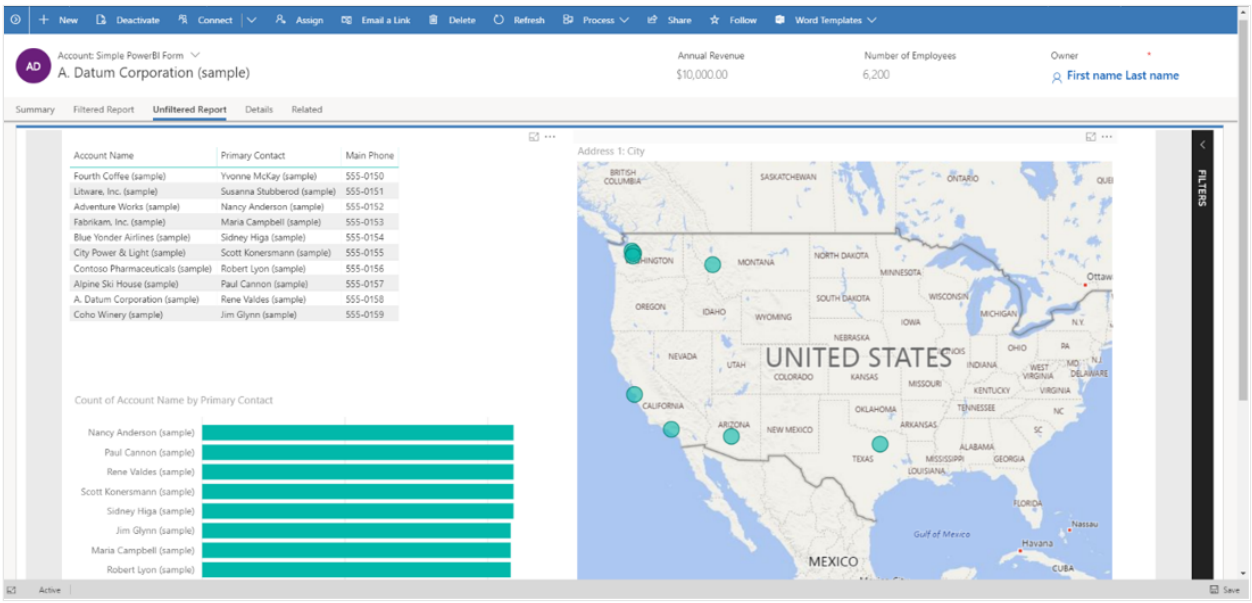
Power BI is a powerful and interactive reporting tool. Power BI dashboards and reports can be introduced into model-driven apps in a range of ways. You can also add Power BI reports and datasets as components in Power Apps solutions. Once you've added a Power BI report to a solution, it can be seamlessly managed as part of your application lifecycle management (ALM) process across environments and tenants.

Additionally, unless direct query techniques are used, record-level security must be reconfigured into the Power BI reports.

There are three techniques for adding Power BI into your model driven apps.

Technique	Notes	Power BI Object
Embed a Power BI report in a model-driven system form	This allows a Power BI report to be presented within a table form and this can be performed in the context of the current record.	Report

Technique	Notes	Power BI Object
Create or edit a Power BI embedded system dashboard	This is a technique that an administrator can use to introduce a Power BI dashboard	Dashboard
How app users can add or edit Power BI visualizations on their dashboards	This allows a full Power BI dashboard to be introduced, however it's introduced by the user, not an administrator.	Dashboard



Excel integration

Microsoft Excel integration lets users easily build self-service reports using, PowerView, PowerPivot, and PowerQuery.

Next Steps

Create a system chart for a table
Reporting considerations
Working with views
Working with forms
Learn more about SSRS reports
Power BI Overview

Create a model-driven app system chart

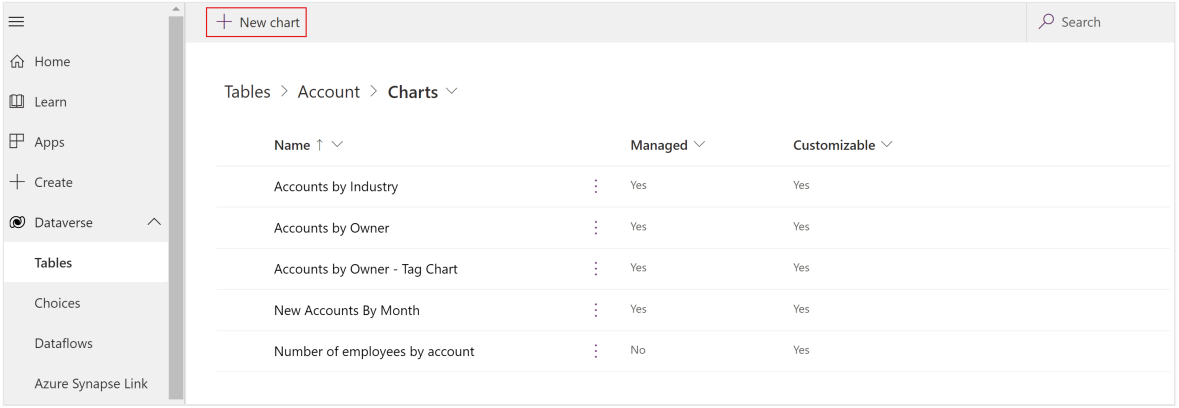
Article • 01/22/2025

In this article you learn how to create a system chart. System charts are organization-owned charts, which makes them available to anyone with access to read the data running the app. System charts can't be assigned or shared with specific app users.

> **Note**
>
> Model-driven apps not configured to display **All** charts for a table need newly created charts selected for the table within the app designer. More information: Add table assets

Create a new chart

1. Sign in to Power Apps.
2. On the left navigation pane, select the **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the table that requires a chart, and then select **Charts**.
4. Select **New chart**.

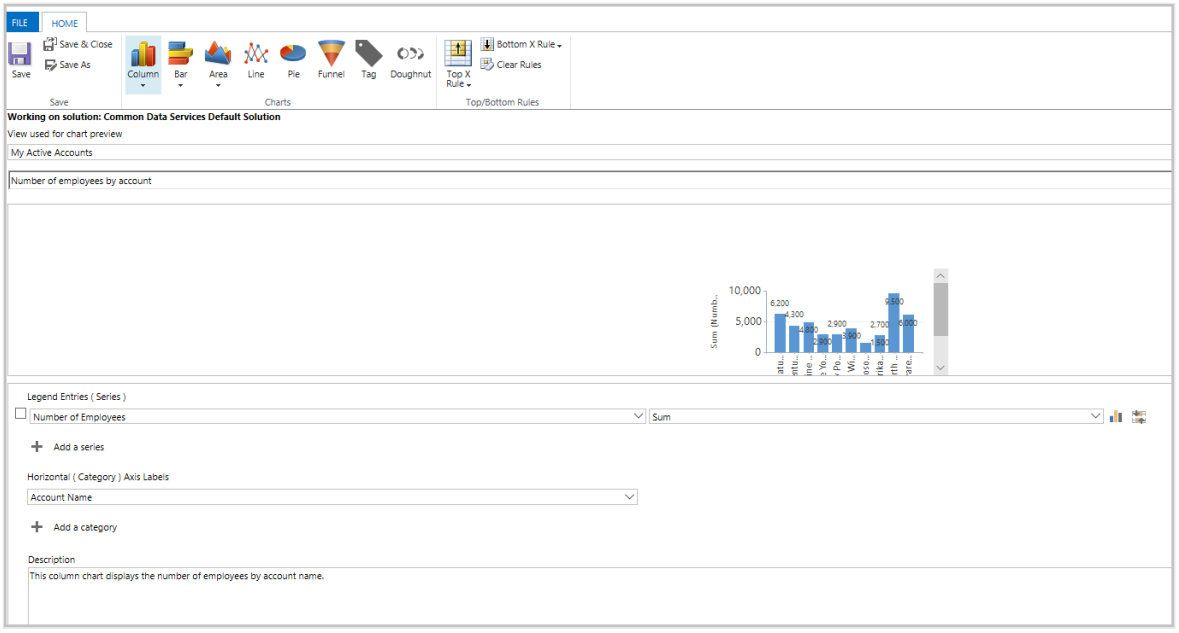


A new window opens where you can create a chart.

5. Specify the type of chart, and how the data is displayed in the chart.
   - Enter the chart name, such as *Number of employees by account*.
   - In the **Select Column** dropdowns:

- In the **Legend Entries (Series)** dropdown list select a column, such as **Number of Employees**.
- In the **Horizonal (Category) Axis** dropdown list select a column, such as **Account Name**.

- Add a description to identify the purpose of the chart, such as *This column chart displays the number of employees by account name*.



6. Select **Save and Close**.

The **Number of employees by account** chart is now displayed in the app designer list of charts available for the account table.

> **Note**
>
> When creating a chart, you can preview it with a selected view. The view isn't permanently associated with the chart. The next time you open the chart, the chart displays using the configured default view. You can change the view to display the chart for the data from a different view.

View your visualization in a model-driven app

Now that the chart is created, it can be used to visualize table data in the model-driven app. Follow these steps to view the chart:

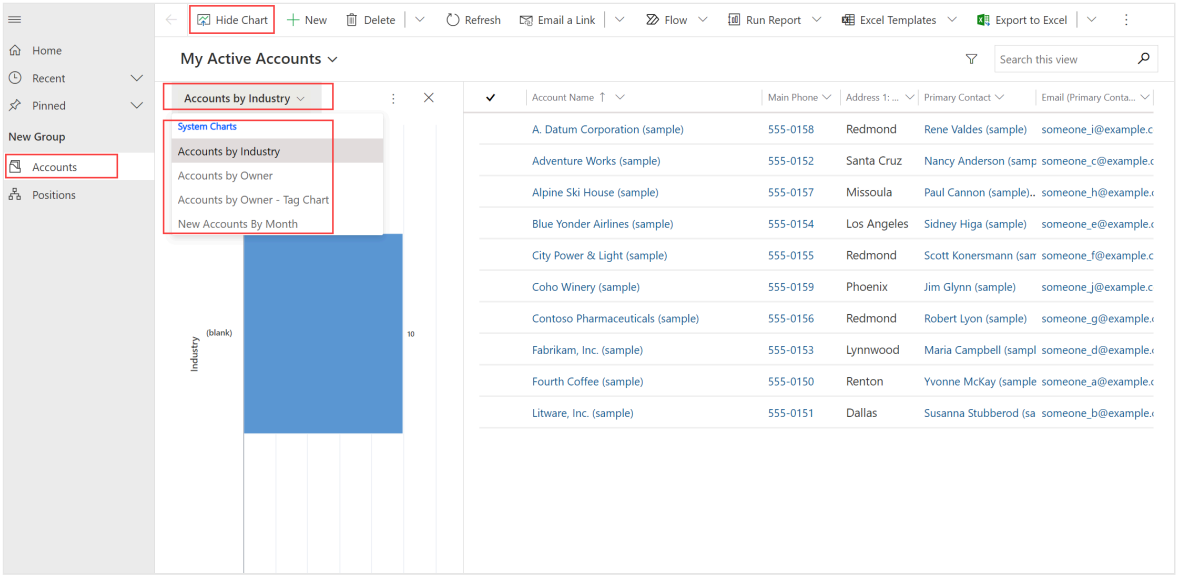
1. Open a model-driven app that contains your table via a solution or open the app directly from the **Apps** area.

2. Select … next to the app, and then select **Play**. The model-driven app opens in a separate browser tab.

3. On the left navigation pane, select a table such as **Accounts**.

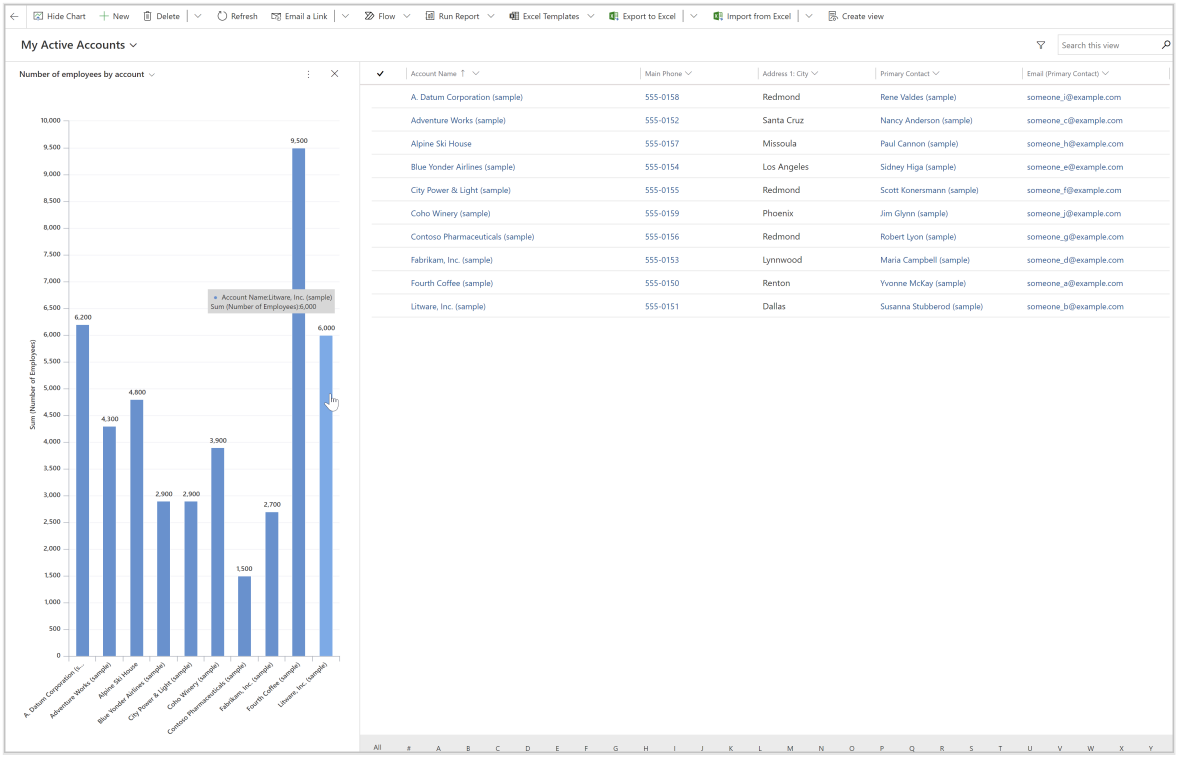
4. On the command bar, select **Show Chart**.

   The chart pane opens.



5. By selecting the dropdown chart list, any of the system charts in the app are available to select.

   The chart appears in-line with the data view.



Use your chart:

- Select a bar on the chart to filter the data in the view.
- Refresh the chart to display other data by changing the view.
- Add the chart to a model-driven app dashboard. More information: Create or edit model-driven app dashboards

> **Note**
>
> If the chart isn't visible in the dropdown chart list, then the model-driven app has been created with specific charts associated with it. To resolve this, open the model-driven app in design mode, select charts, and then select your newly created chart.
>
> 
>
>
> Then save and publish your app.

Maximum number of records displayed in a chart

Charts display views that return up to 50,000 records. Views that exceed 50,000 records display the message: The maximum record limit is exceeded. Reduce the number of records. More information: Reporting infrastructure

Known chart creation issues

In the chart designer, adding an order by on certain calculated columns isn't supported and cause an error. The calculated columns causing the issue are using another calculated column, a related table column, or a local column on the table.

Next steps

Create or edit dashboards

Feedback

Was this page helpful? Yes No

Provide product feedback

Create or edit model-driven app dashboards

Dashboards are collections of charts relating to Microsoft Dataverse tables.

There are two types of dashboards, user dashboards and system dashboards. An app user can create a dashboard visible only to them in the app areas where they have privileges.

An admin or customizer creates or customizes system dashboards that, when published, are visible to all app users. A user can choose to set their user dashboard as their default dashboard and override the system dashboard.

Dashboard interactivity

Dashboards can be standard or interactive.

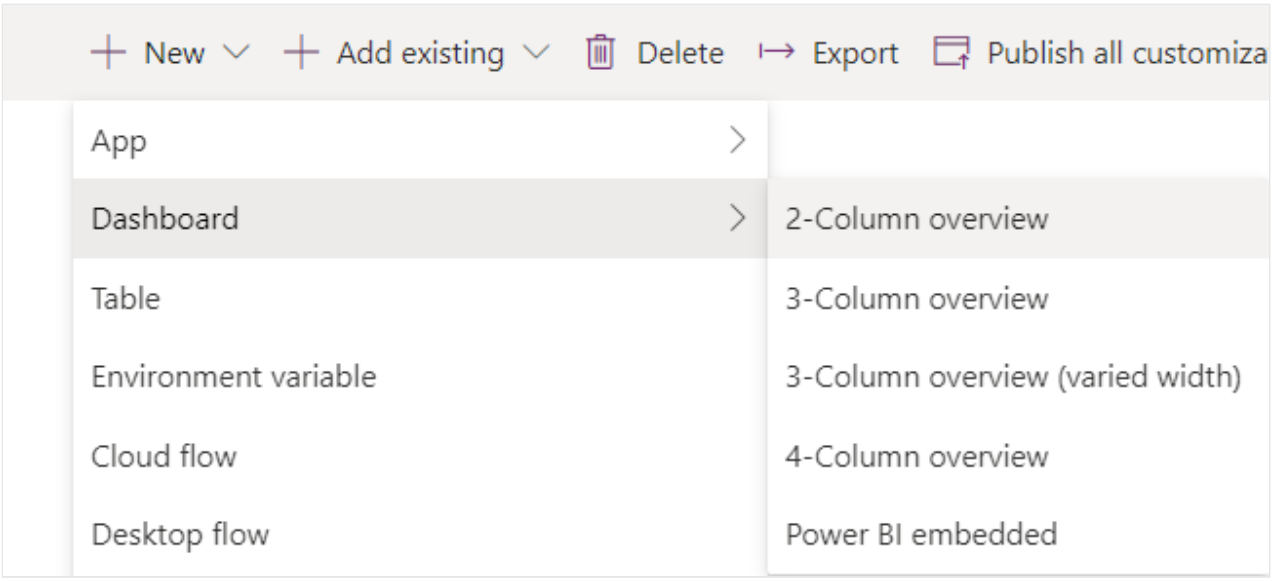
Standard dashboards support adding one or more unrelated components such as charts or lists.

Interactive dashboards provide the capability for users to act on a particular row directly from the dashboard. This article focuses on standard system dashboards. For information about interactive dashboards, see Configure model-driven app interactive experience dashboards.

Create a new standard dashboard

1. Sign in to Power Apps.
2. Select **Solutions** in the left navigation pane, and then open the unmanaged solution you want or create a new one. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. On the toolbar select **New**, select **Dashboard**, and then choose one of the following layouts:
    - 2-column overview
    - 3-column overview
    - 3-column overview (varied width)
    - 4-column overview

- Power BI embedded



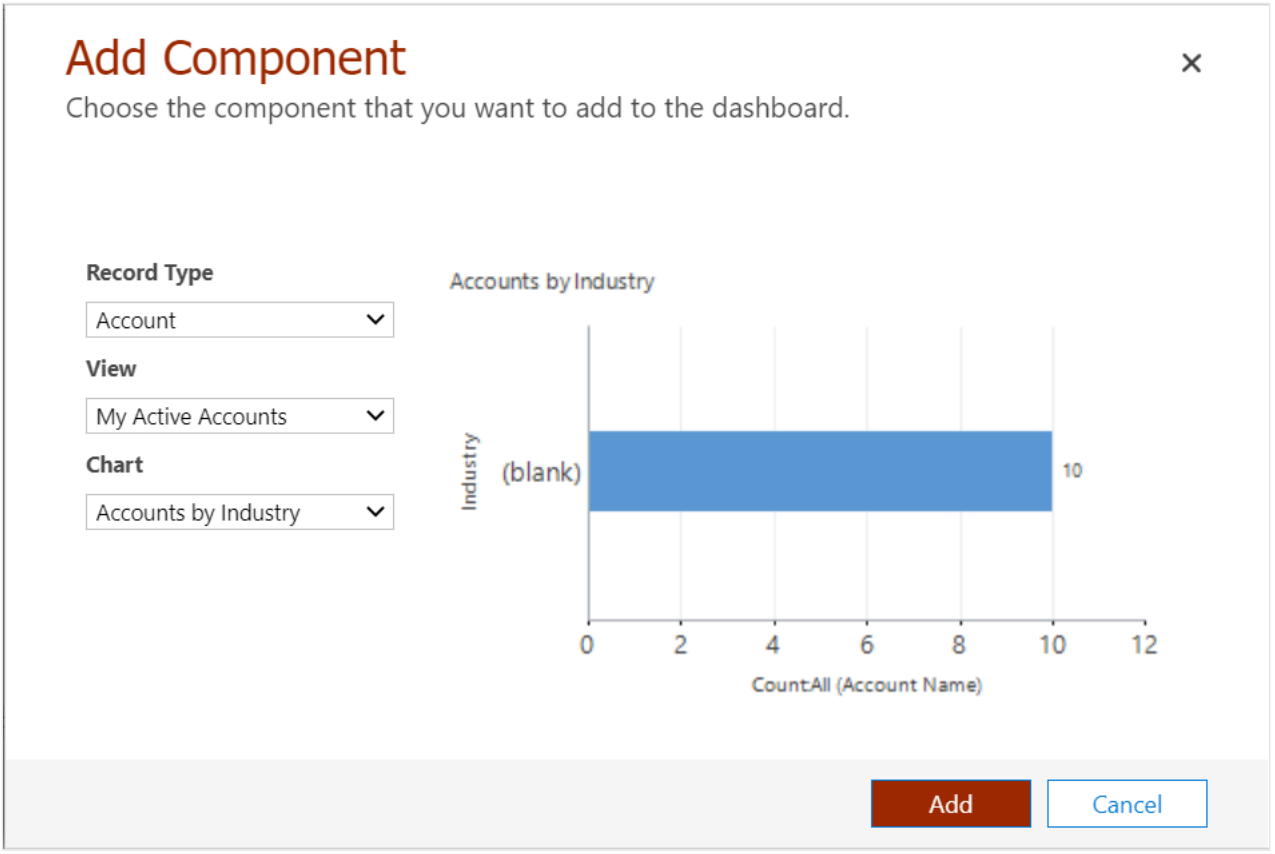
4. If you select Power BI embedded, go to Create or edit a Power BI embedded system dashboard. Otherwise, in the **Dashboard: New** page, enter a name for the dashboard.

5. In one of the component sections, select the icon for a chart or a list.

   Up to six components can be included in a dashboard.

6. For example, to add a chart, select the chart icon on the tile of the dashboard canvas where the chart needs to appear. Then, in the **Add Component** dialog, select values for **Record Type**, **View**, and **Chart**. Then select **Add** to add the chart to the dashboard. For

information about how to create a chart, see Create a model-driven app system chart.



7. When you finish adding components to the dashboard, select **Save** and then **Close**.

8. On the solution toolbar, select **Publish**.

Edit an existing dashboard

1. Sign in to Power Apps.

2. Select **Solutions** in the left navigation pane, and then open the solution you need. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. In the list of solution components, open the dashboard, select one of the component areas, and then on the toolbar select **Edit Component**.

4. In the **Set Properties** dialog box, make changes to a chart or list such as change the table, default view, or add a chart selector. When done, select **OK**.

   For more information about setting dashboard component properties, see Set properties for a chart or list included in a dashboard.

5. When the changes are complete, select **Save** and then select **Close**.

6. On the solution toolbar, select **Publish**.

> **Note**
>
> Lists in the dashboard designer don't display a data preview.

Next steps

Configure interactive experience dashboards

Last updated on 04/08/2026

Create and configure model-driven app interactive experience dashboards

Interactive experience dashboards can be a *one-stop workplace* for app users, such as service reps, to see workload information and take action. They're fully configurable, security-role based, and deliver workload information across multiple streams in **real time**. Interactive dashboard users don't need to page through the application looking for a particular row; they can act on it *directly* from the dashboard.

The interactive experience dashboards come in two forms: **multi-stream** and **single-stream**. In addition, multi-stream dashboards can be home page or table-specific dashboards. The table-specific dashboards are configured in a different part of the user interface and are partially preloaded with table-specific configuration information.

- Multi-stream dashboards. The multi-stream dashboards display data in real time over multiple data streams. There’s no limit on how many streams you can configure on the dashboard. The data in a stream can be based only on one table, but, each stream can be based on a different table.
- Single-stream dashboards. In the table-specific dashboards, all streams are based on the same table. The data flows from various views or queues, such as **My Activities**, **My Cases**, or **Cases in the Banking Queue**.

> ⓘ **Note**
>
> The examples described here use the Case table, which is available with the Dynamics 365 Customer Service app.

The single-stream dashboards display real-time data over one stream based on a table view or queue. The tiles are positioned on the right side of the dashboards and are always shown. The single-stream dashboards are typically helpful to tier two service leads or managers, who monitor fewer, but more complex or escalated cases.

Dashboard charts

Multi-stream and single-stream dashboards contain interactive charts that provide a count of relevant rows, such as cases by priority or by status. These charts also act as visual filters. The

visual filters (interactive charts) are based on multiple tables and in the single-stream dashboards, the table in the data stream defines the visual filter table.

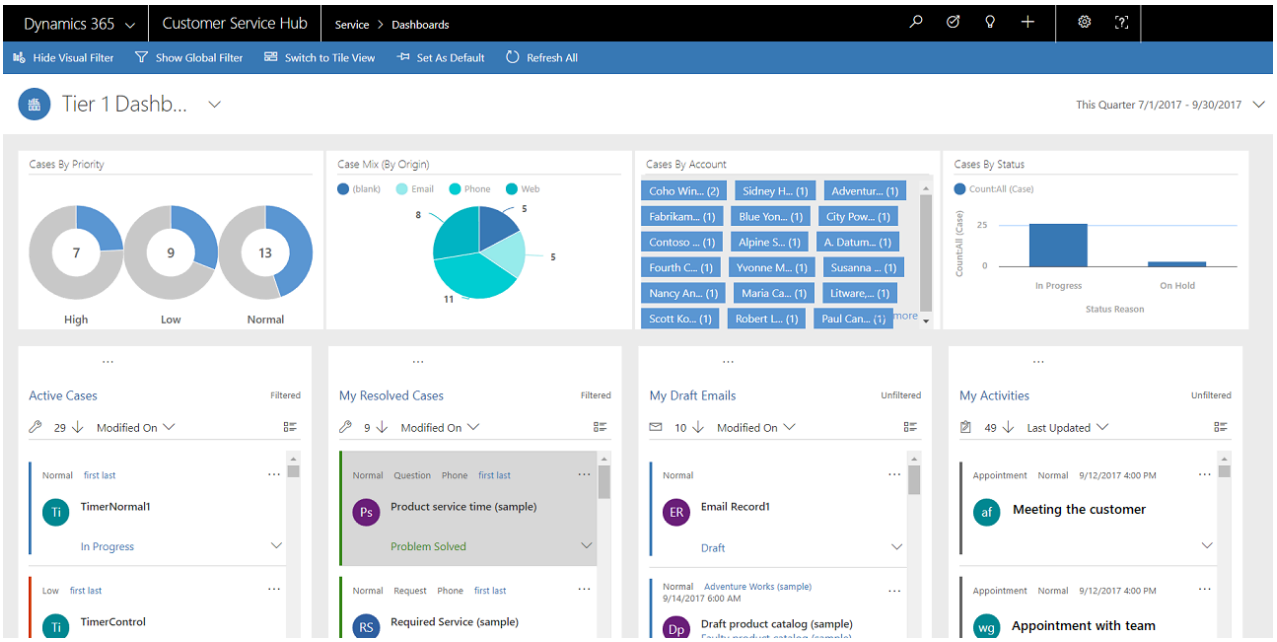
Users can apply additional filtering with `global filter` and `timeframe` filter. The global filter works at a column level on all charts, and also on streams and tiles that are based on the filter table (you specify the filter table when you configure the visual filters).

> **Note**
>
> The interactive dashboards are solution aware and can be exported and then imported into a different environment as a solution. However, the queues that the streams and tiles are based on aren't solution aware. Before you import the dashboard solution into the target system, the queues have to be manually created in the target system in **Settings** > **Service Management** > **Queues**. After you create the queues, import the dashboard solution to the target system, and then edit the streams or tiles that are based on the queues to assign the newly created queues appropriately.

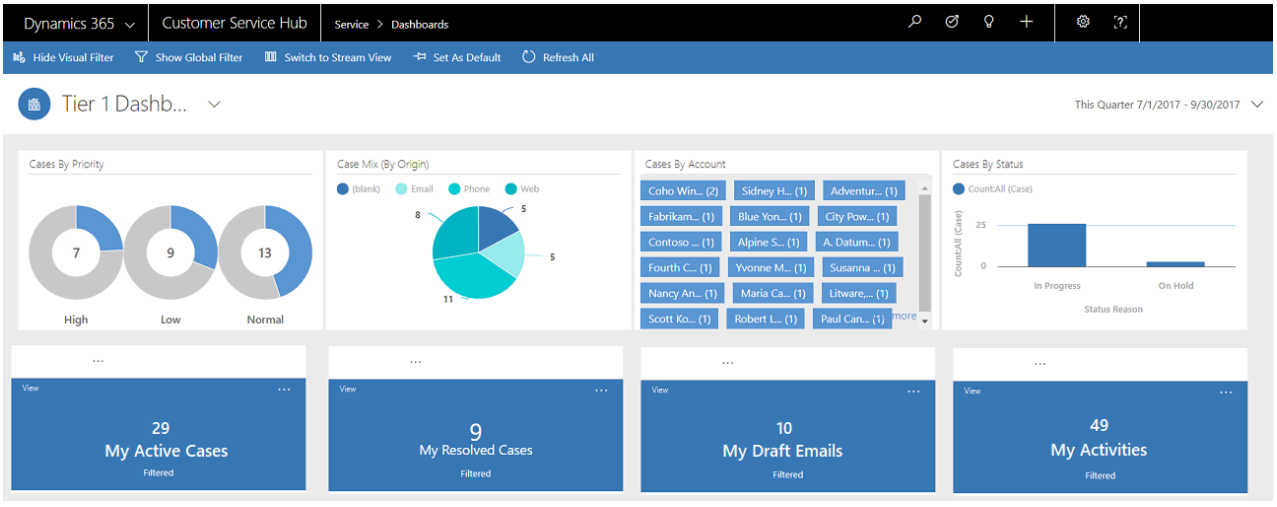
The illustrations in this article show multi-stream and single-stream dashboards with the header pane. Below the header you can view visual filters and streams. In the single-stream dashboard, you also view tiles. For each dashboard type, you can choose from several different layouts that are also shown. The dashboard header contains the following controls and selectable icons, from left to right: dashboard picker, refresh, visual filter icon, global filter icon, and time frame filter.

Dashboard layouts

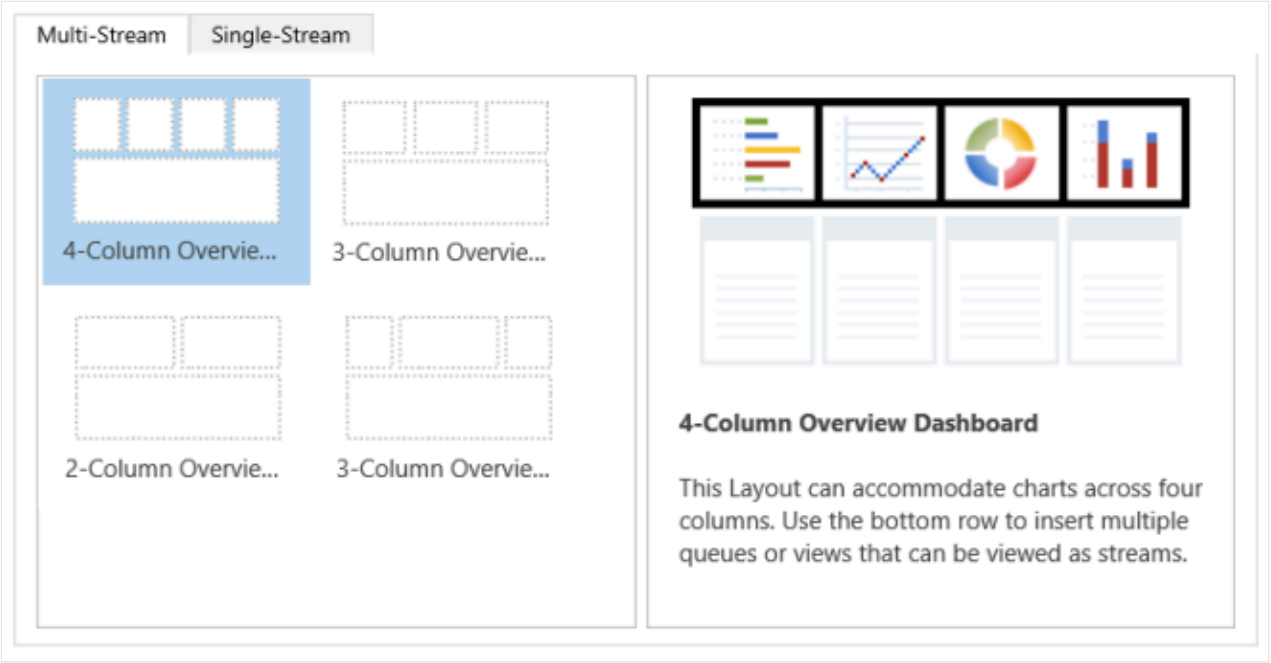
- Multi-stream dashboard standard view. In the multi-stream dashboard, you view a row of visual filters at the top with the data streams below them.



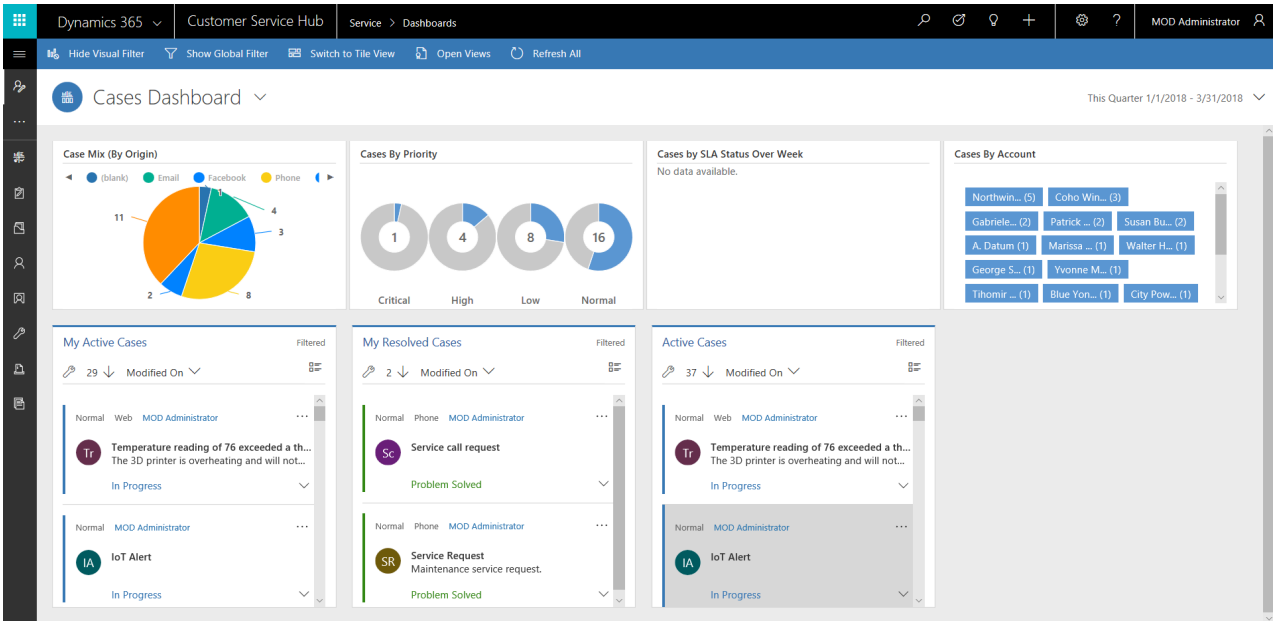
- Multi-stream dashboard tile view. The same dashboard, only in the tile view.



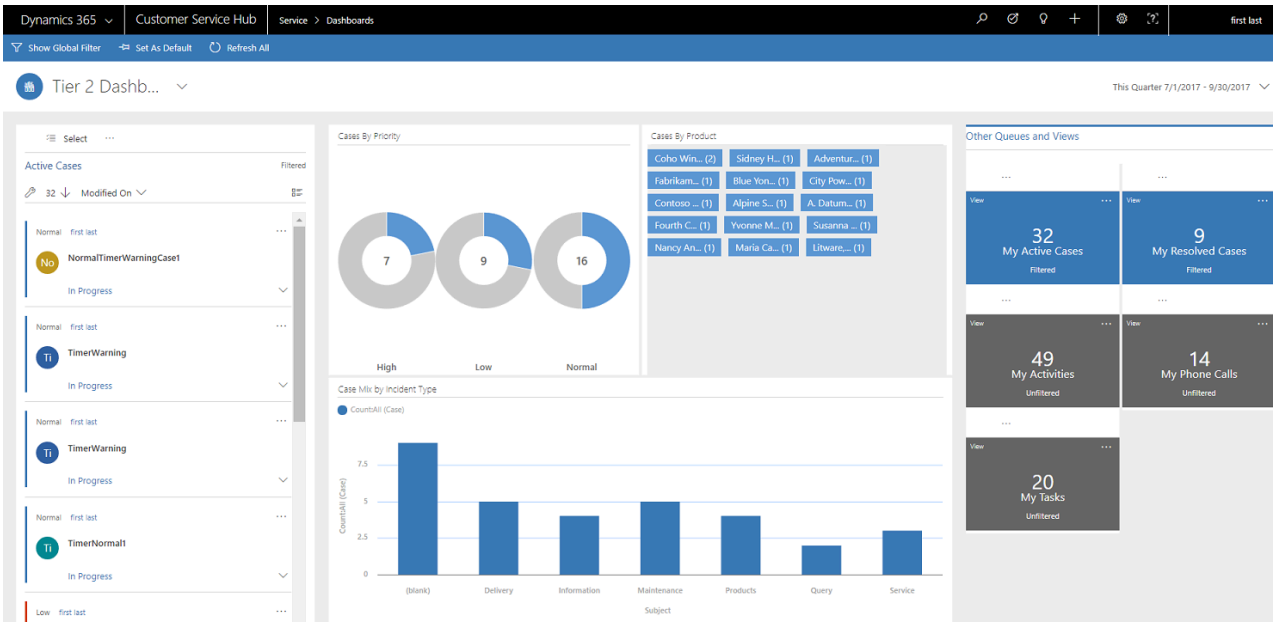
- Multi-stream dashboard layouts. For multi-stream dashboards, you can choose from four different layouts.



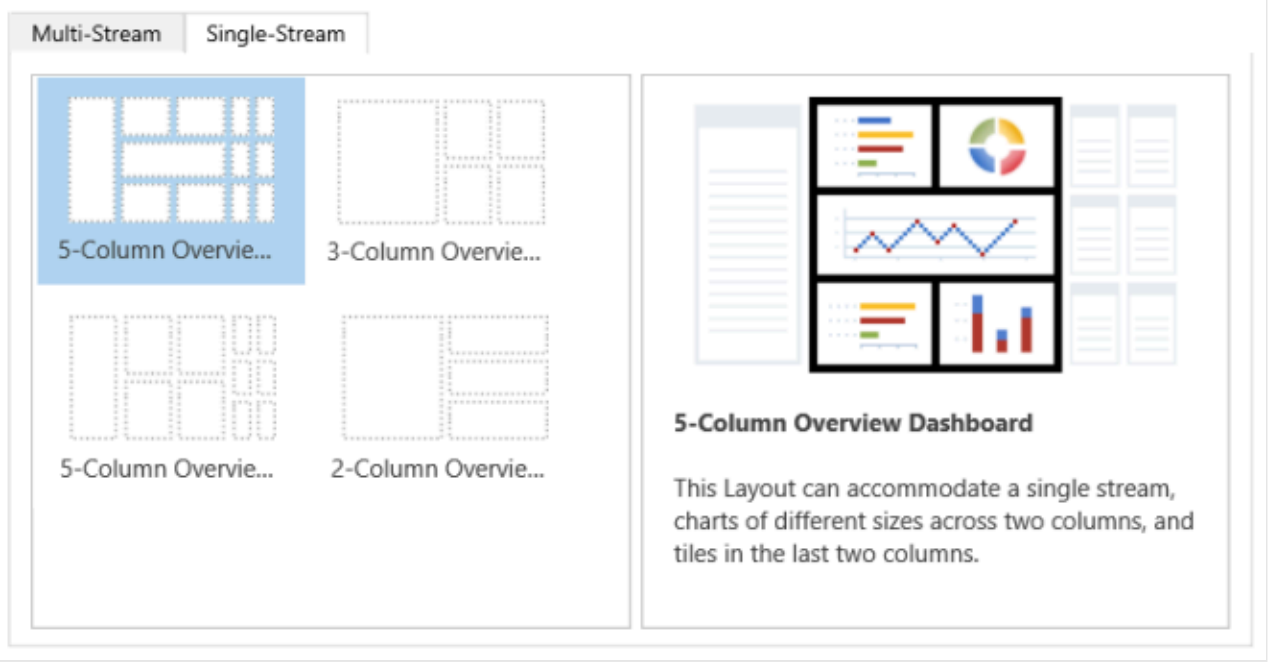
- Multi-stream table-specific dashboard. The table-specific dashboard for the case table is shown here.



- Single-stream dashboard. The single-stream dashboard contains the data stream on the left and visual filters and tiles on the right pane.



- Single-stream dashboard layouts. Four different layouts can be chosen for single-stream dashboards.



Configure filter columns, and security roles for interactive dashboards

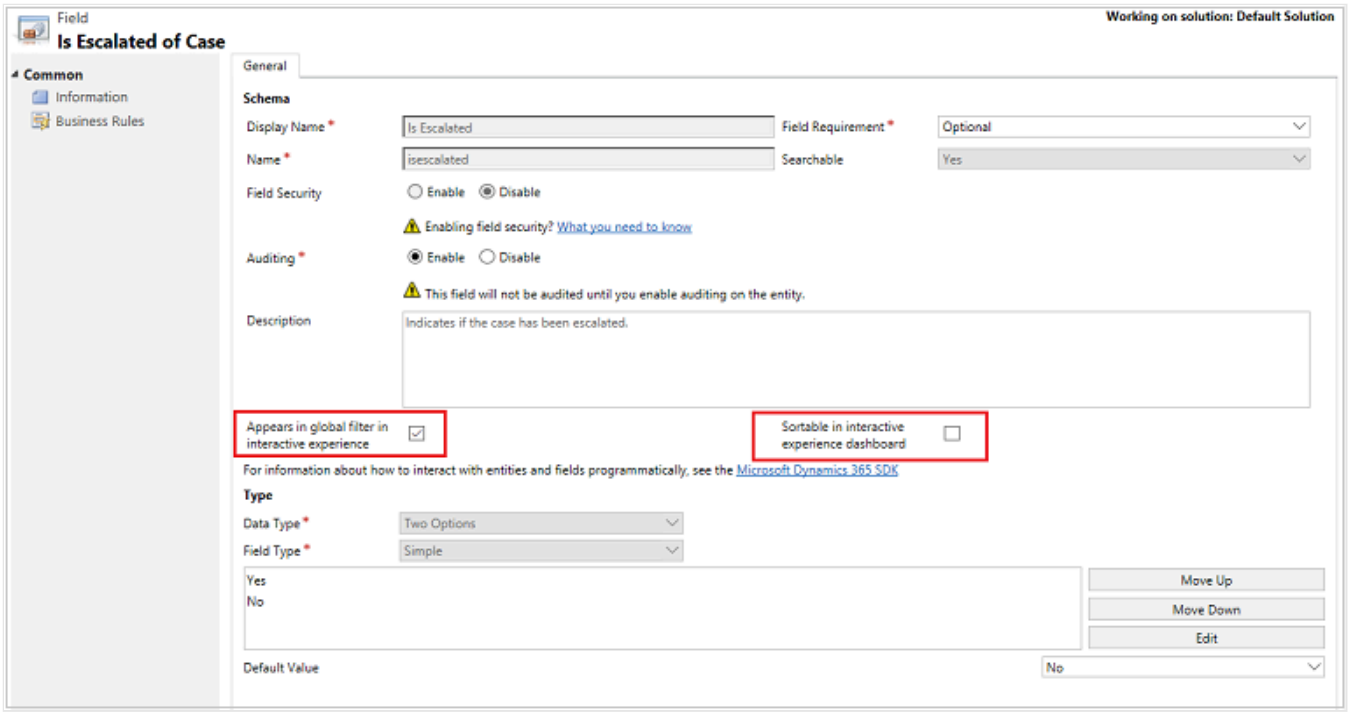
When interactive dashboards are configured, the first task is to enable filter columns and security roles so that interactive dashboards can be configured for them. Notice that interactive dashboards are enabled for all tables and custom tables by default.

Configure filter columns

For a column to appear in the global filter and be included in the data stream sort, two flags must be set:

- Appears in global filter in interactive experience.
- Sortable in interactive experience dashboard.

In this example there are two interactive dashboard options available in the Case table for the **IsEscalated** column.

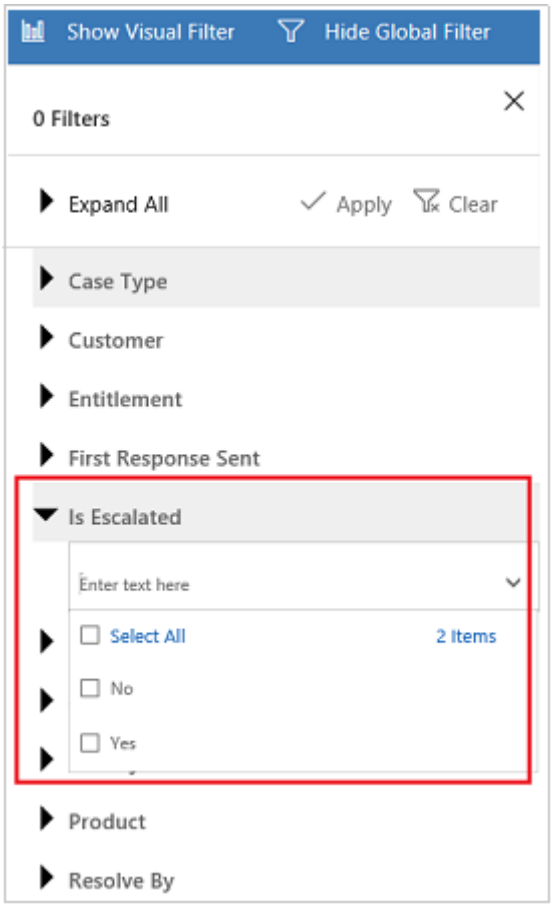


Configure the 'Appears in global filter in interactive experience' option

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution you want, and then on the toolbar select **Switch to classic**.
4. Under **Components**, expand **Entities** (which are effectively tables), and then expand the specific table.
5. In the navigation pane, select **Fields** and in the grid, double-click the column, or data field, required.
6. In the **General** tab, select the **Appears in global filter in interactive experience** check box. Select **Save and Close**.
7. Select **Publish All Customizations** for the changes to take effect.

The columns that are enabled for **Appears in global filter in interactive experience** appear in the global filter flyout window when the global filter icon is selected on the dashboard header. In the flyout window, the service reps can select the columns on which they want to filter globally, in charts, and also in streams and tiles that are based on the filter table.

The global filter flyout window is shown here:



> **Tip**
>
> When a visual filter is configured based on the columns like priority or status, a best practice is to also enable these columns (priority, status) to appear in the global filter.

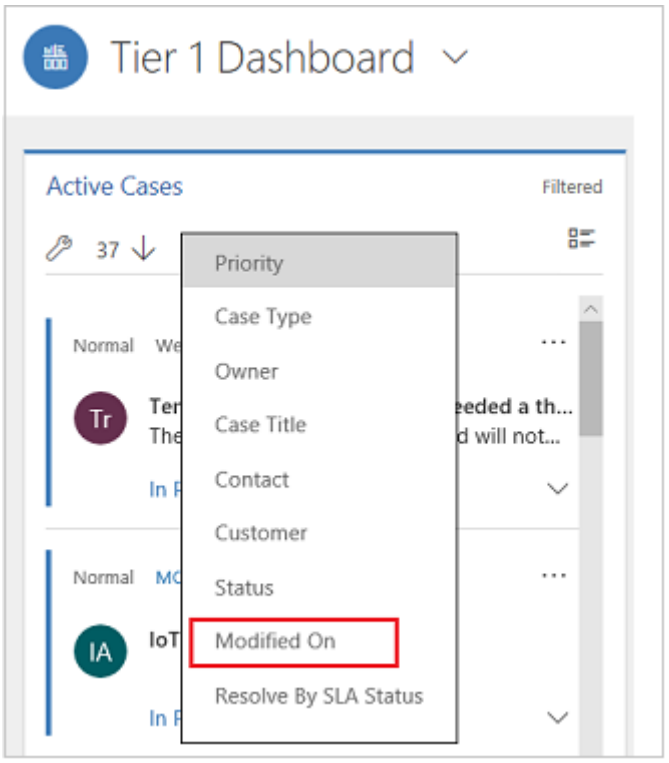
Configure the 'Sortable in interactive experience dashboard' option

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution you want, and then on the toolbar select **Switch to classic**.
4. Under **Components**, expand **Entities** (which are also known as tables), and then expand the specific table.

5. In the navigation pane, select **Columns** and in the grid, double-click the column for which sorting is required.
6. In the **General** tab, select the **Sortable in interactive experience dashboard** check box. Select **Save and Close**.
7. Select **Publish All Customizations** for the changes to take effect.

The columns that are configured for sorting appear in the drop-down list on the stream header.

The following screenshot shows the flyout dialog with the list of the available columns for sorting, in the drop-down list. The default sort is always set on the **Modified On** column.



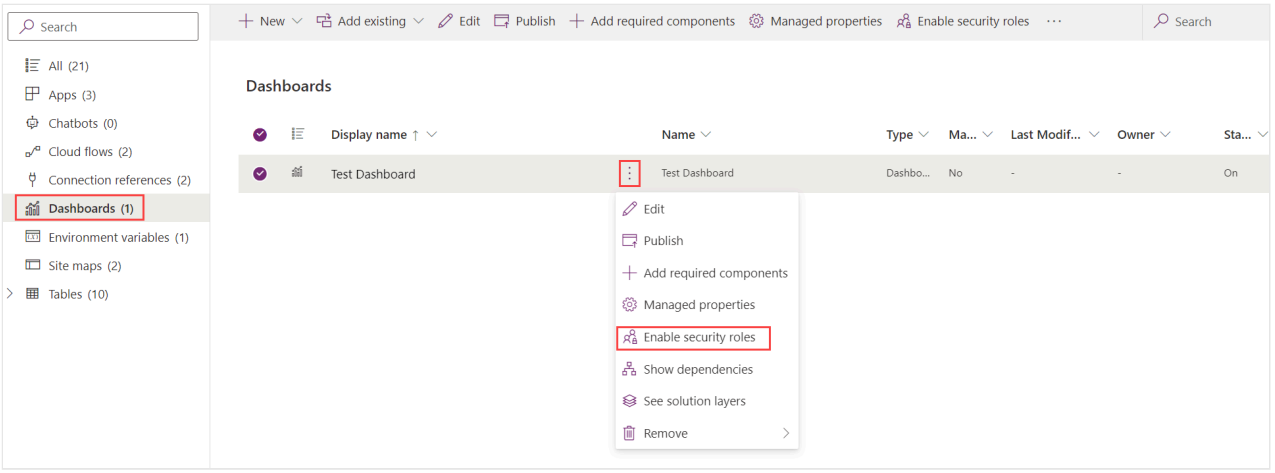
Enable security roles

Select and enable security roles that are able to view the interactive dashboards.

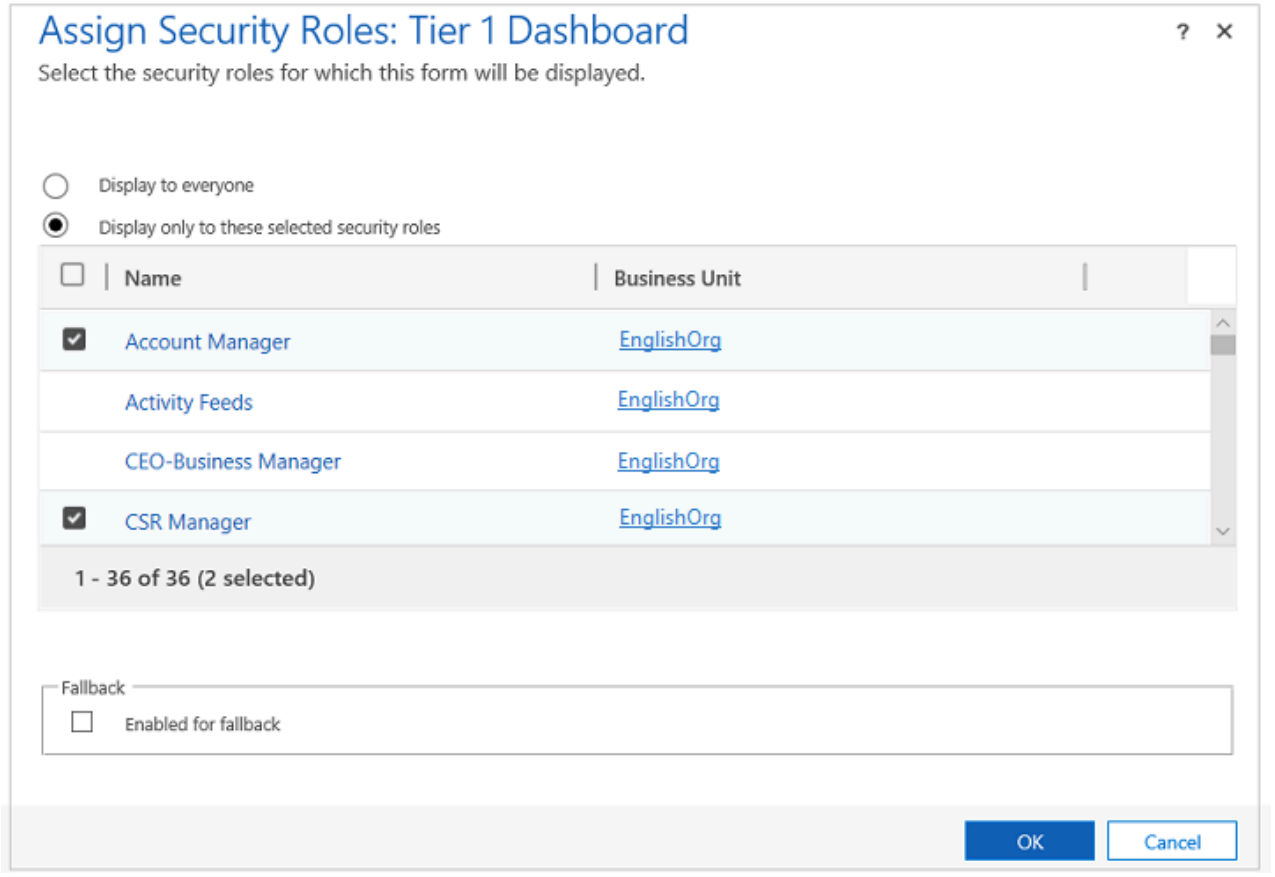
Enable security roles for interactive dashboards

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the required solution.

4. Select the dashboard and then on the toolbar select **Enable security roles**. This action opens a new tab with the classic interface.



5. In the **Assign Security Roles** dialog, select the **Display only to these selected security roles** option and select the roles that you want to enable. Select **OK**.



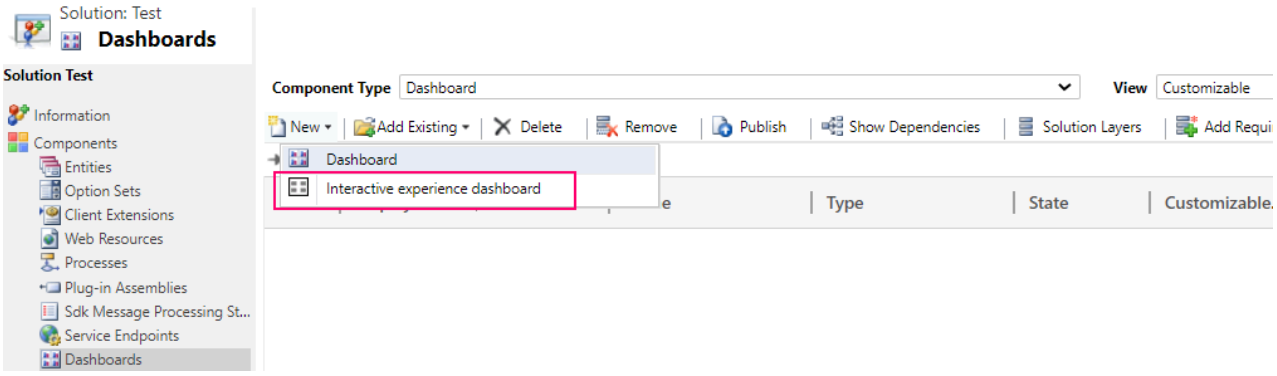
6. Select **Publish** for the changes to take effect.

Create interactive experience dashboards

The following sections describe how to create and then configure the various types of interactive dashboards.

Configure a multi-stream interactive dashboard using the 4-column layout

1. Sign in to Power Apps.
2. Select an environment with an unmanaged solution.
3. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
4. Open the solution you want, and then on the toolbar select **Switch to classic**.
5. In the left navigation pane, select **Dashboards**, on the toolbar select **New**, and then select **Interactive experience dashboard**.



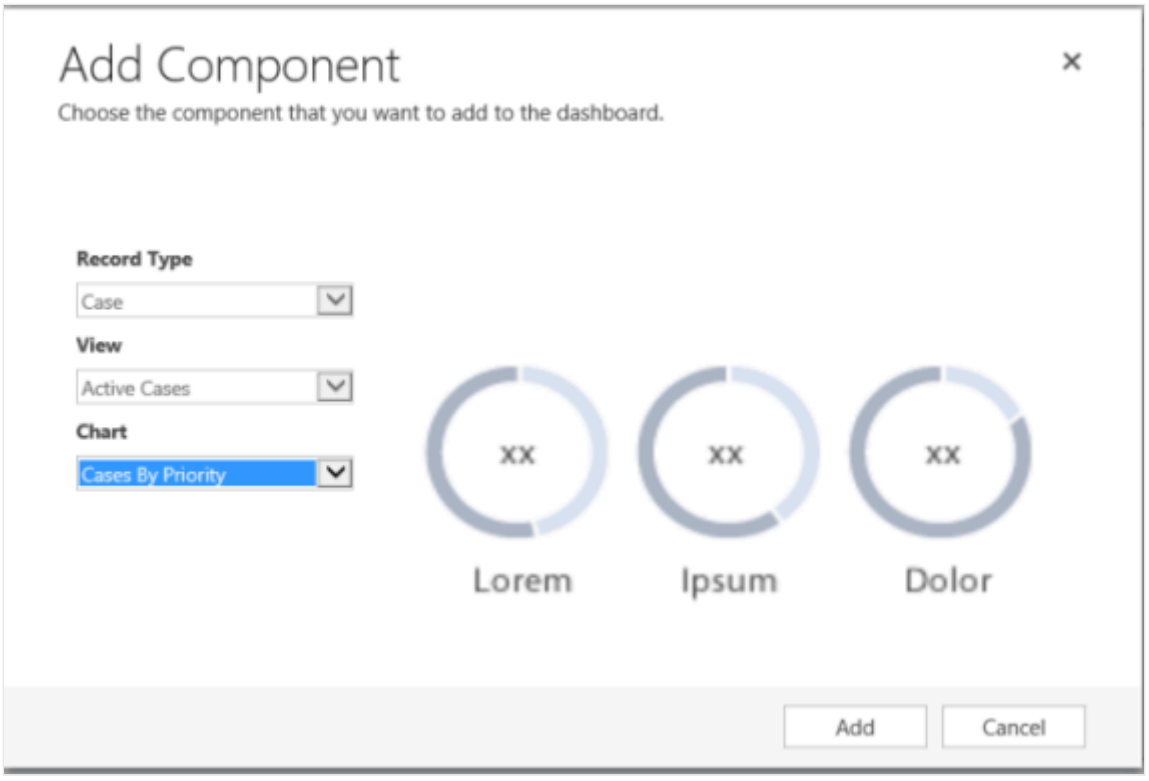
6. Choose the layout, either 2, 3, or 4 column width.
7. When the dashboard form opens, fill in the filtering information at the top of form, as shown here.



- **Filter Table**: The visual filters and global filter attributes are based on this table.
- **Table View**: The visual filters are based on this view.
- **Filter By**: The column that the time frame filter applies to.
- **Time Frame**: The default time frame filter value for the **Filter By** column.

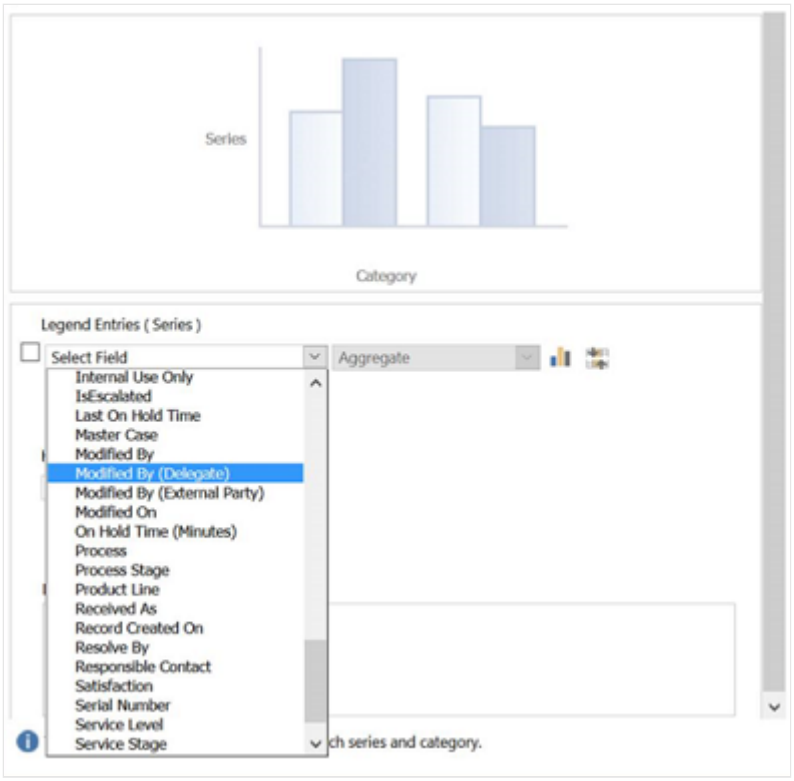
After filtering information is specified, start adding components for the charts and the data streams. To add a component, simply select the element in the center of the chart or stream, and when the dialog appears, select the required information from the drop-down list, as shown in these illustrations.

Add the **Cases By Priority** doughnut chart.



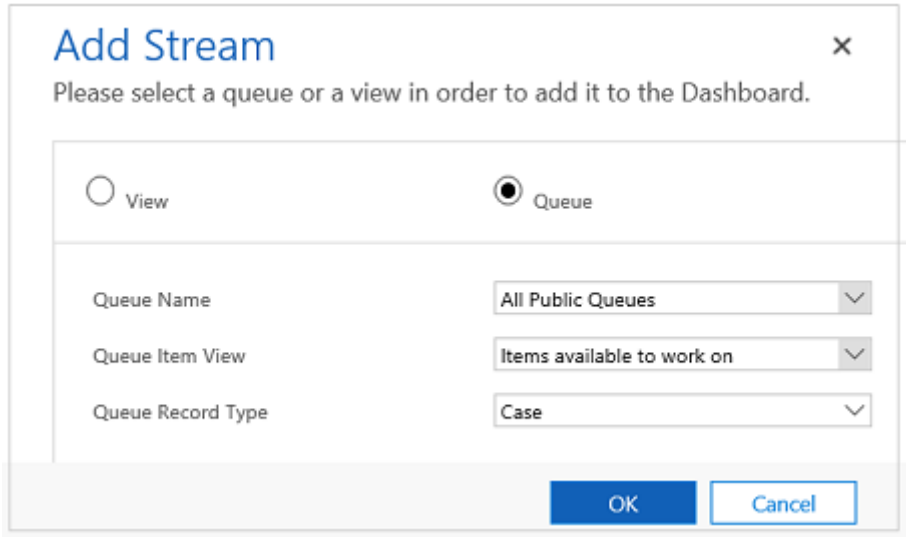
Some charts, such as bar charts or pie charts, render showing the data stored in the system. The doughnut charts and tag charts load as static images and don't show the preview of the actual data.

> ⓘ **Note**
>
> The charts configured for the visual filters can use the columns of the **Filter** table as well as related tables. When charts are based on related table columns, the customer service representatives can filter charts using these related table columns. The columns that are based on the related table usually have the following format in the chart configuration window: "field name (entity name)", such as the **Modified By (Delegate)** column. To create multi-entity charts, columns of a related table must be added to any of the views, and then use these columns while creating charts.



Next, configure the streams. Just like with adding components in the charts, select the element inside the stream panel. When the dialog appears, select **View** or **Queue** depending on what element you want the stream to use. Enter the required information, as shown in the following illustration.

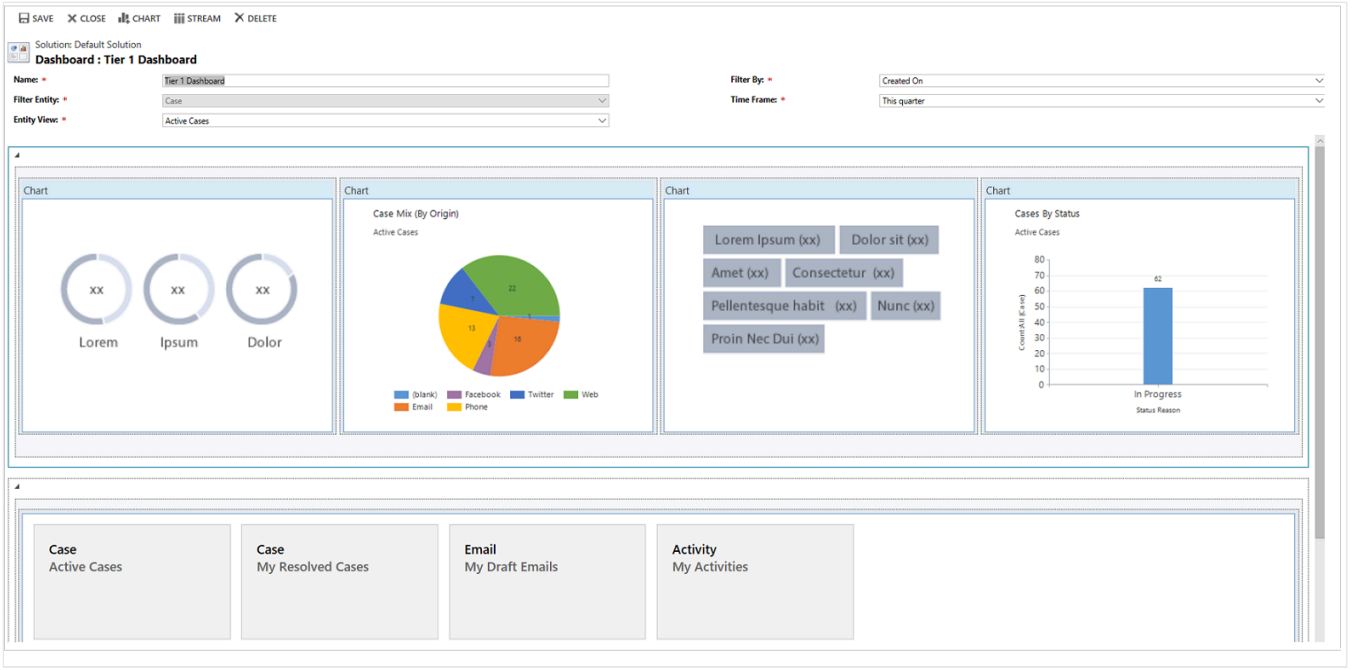
Configure the stream for the **Items available to work on** as shown here:



> ⓘ **Note**
>
> The **Queue** option is available in the dialog box only for queue-enabled tables. For table dashboards, if the table isn't queue enabled, the **Queue** option won't be visible in the

dialog box. It's only possible to use the **View option** in the stream of dashboards for tables that aren't queue enabled.

The following illustration is an example of a fully configured chart panel and stream panel:



After the dashboard has been completely configured, save it and publish the customizations for the changes to take effect.

Edit or delete individual streams of an existing dashboard

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the solution you want, and then open the interactive dashboard.
4. Select the stream that you want to edit to select it, and then select **Edit Component**.
5. Depending on whether you want to add a view or queue to the stream, select the view or queue details for the stream, and then select **Set**.
6. Select **Save**.

It's also possible to delete an individual stream from a dashboard. To do this, select the stream, and then on the toolbar, select **Delete**.

Create a table-specific dashboard

A table-specific dashboard is a multi-stream dashboard. Configuring this dashboard is similar to configuring a home page multi-stream dashboard, but this is performed in a different place

in the UI and there are other minor differences.

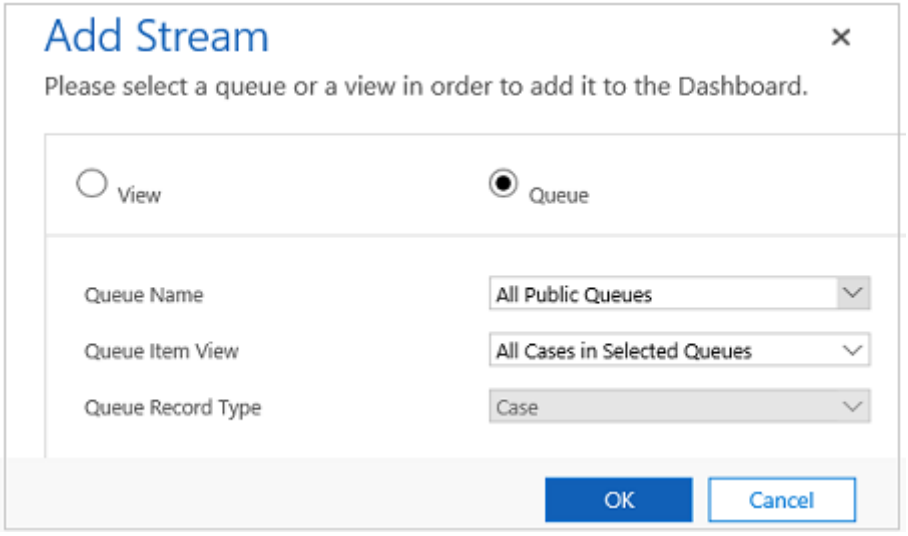
For example, instead of selecting a table, some columns in the table-specific dashboard are preset to the table relevant to the dashboard.

1. Sign in to Power Apps.
2. On the left navigation pane, select **Tables**. If the item isn't in the side panel pane, select ... More and then select the item you want.
3. Open the table you want.
4. Select the **Dashboards** area, and then on the toolbar select **Add dashboard**.
5. Choose the layout, either 2, 3, or 4 column width.
6. When the dashboard form opens, the **Filter Table** is preset to the table, which is tied to the dashboard. The **Table View** drop-down list contains the available views for the table. Select the view and fill in the rest of the required information on the page.

The rest of the setup is similar to the home page multi-stream dashboard setup described in the previous section.

Configure a single-stream dashboard

Configuring a single-stream dashboard is similar to the multi-stream dashboard. All UI navigation steps are the same as for the multi-stream dashboard. It's possible to choose a layout that includes tiles or the layout that doesn't include tiles. If the tiles are included, they're always displayed on the dashboard. To configure a tile, the icon in the center of the tile is selected. When the **Add Tile** window opens, fill in the required data. The following illustration is an example of the tile setup.



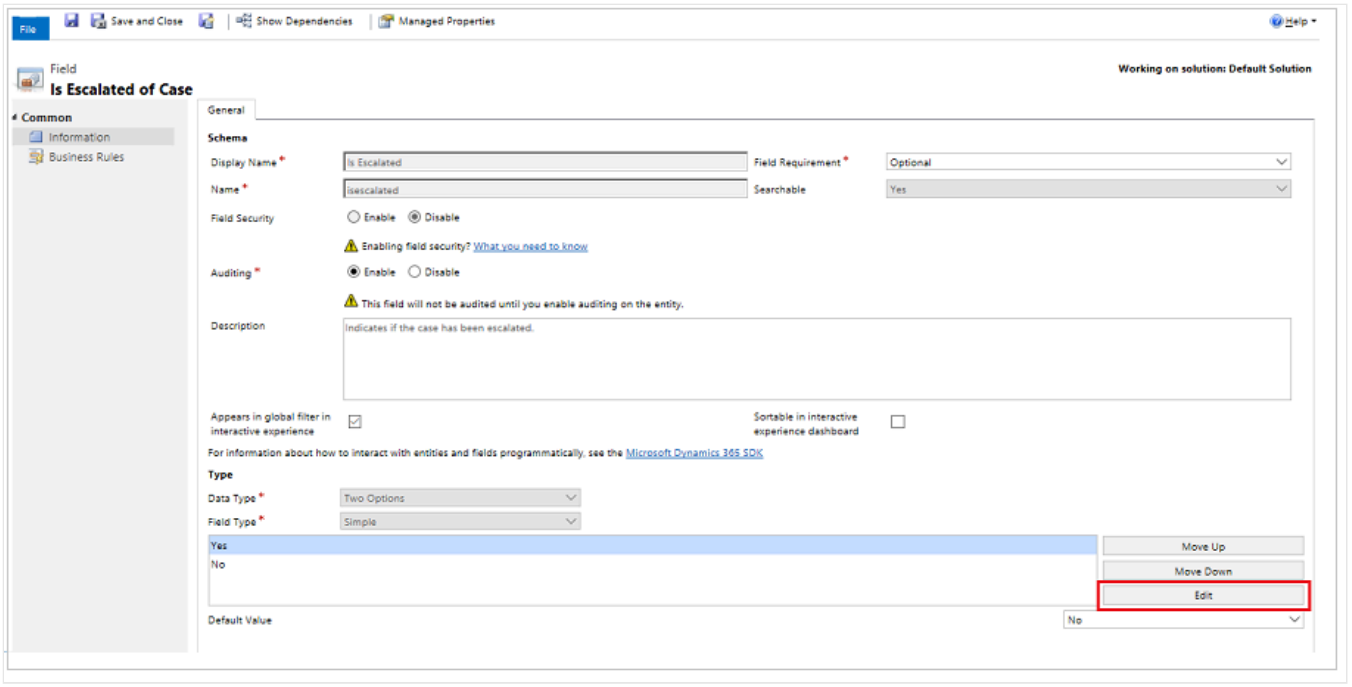
Configure dashboard colors

The color that appears in the charts and streams for specific column values can be configured. For example, high priority cases can be shown in red, medium priority cases in blue, and low priority cases in green in the interactive charts. In the streams, there's a thin vertical line in color next to the work item description.

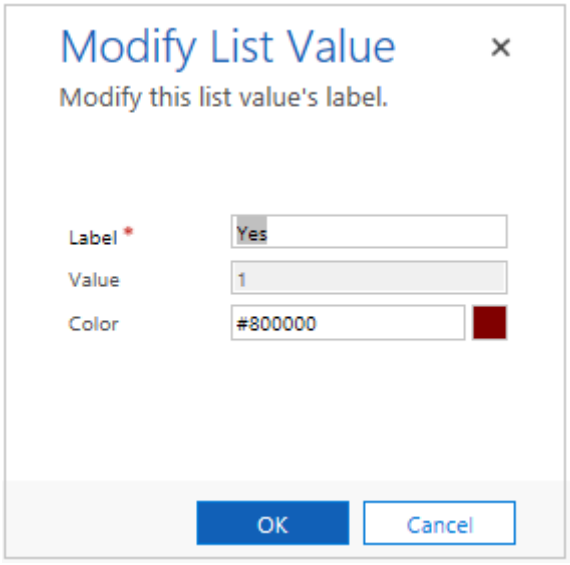
> **Important**
>
> - The color coding isn't available for the tag charts and doughnut charts. These charts appear on the dashboard in white, gray, and black shades.
> - With interactive dashboards, a chart uses the color assigned to the categories that make up the different values, even if the chart is configured to use random colors, when the chart is configured to be grouped by any of the following column types:
>   - Choice
>   - Yes/No
>   - Status Reason
>
> More information: Chart colors aren't correct on dashboards

1. Sign in to Power Apps.
2. Select an environment with an unmanaged solution.
3. On the left navigation pane select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
4. Open the solution you want, and then on the toolbar select **Switch to classic**.
5. Under **Components**, expand **Tables**, and then expand the table required. If the table isn't displayed select **Add Existing** to add it.
6. In the navigation pane, select **Columns**. In the grid, double-click the relevant column.
7. In the **General** tab, in the **Type** subarea, select **Yes** and then select **Edit**.
8. When the **Modify List Value** dialog appears, set the new value in the **Color** text box. Select **OK**.
9. Select **Save and Close**.
10. Select **Publish** for the changes to take effect.

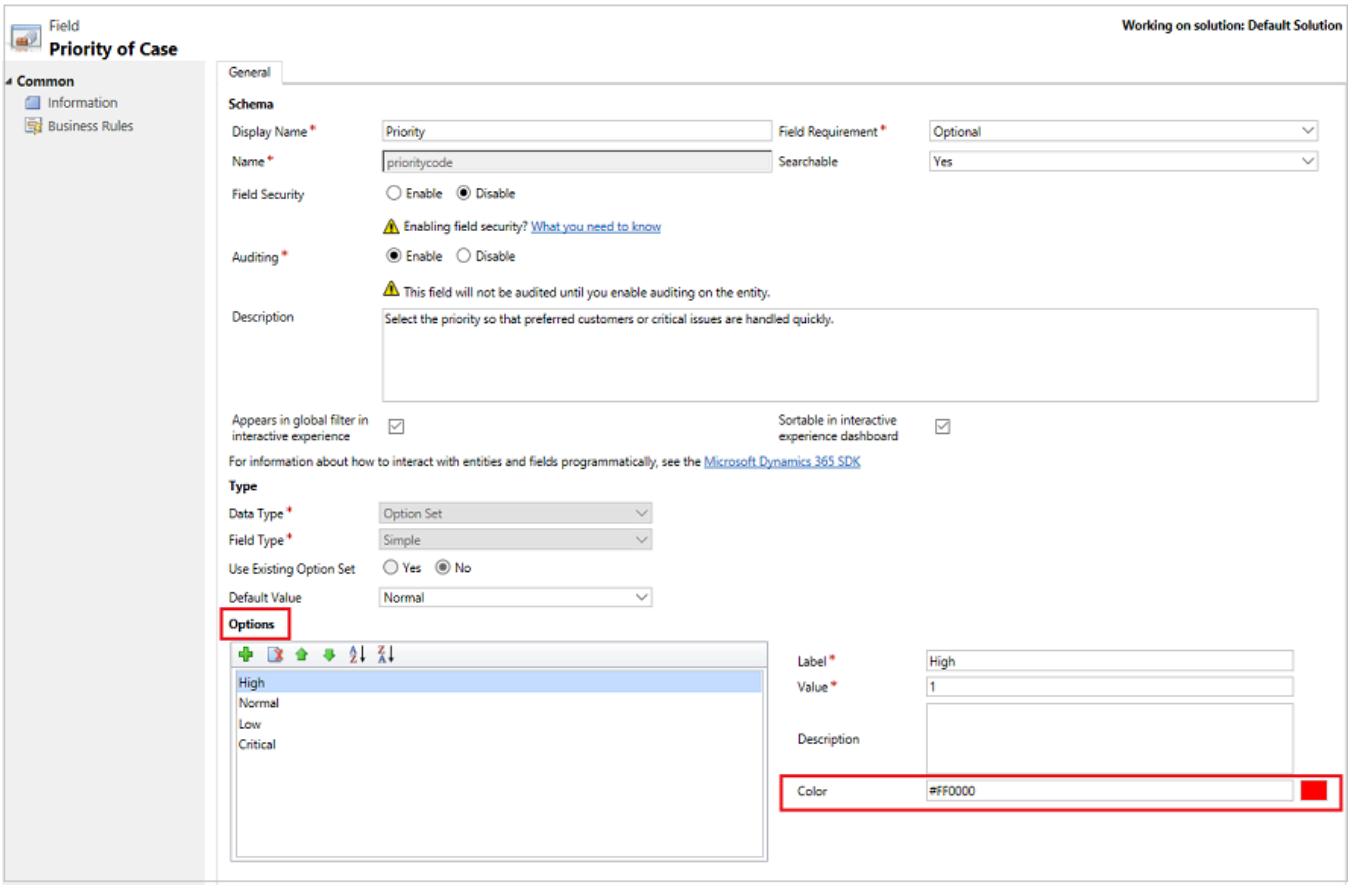
In the following example, the color for the **IsEscalated** column is changed. Use the **Edit** button to open the **Modify List Value** dialog box:



When the **Modify List Value** dialog box opens, enter the hexadecimal color code, such as #800000 shown here:



Similarly, if the **Priority** column is used to modify the colors of the case priority options, choose the color in the **Options** subarea of the **General** tab, as shown here:



Next steps

Set properties for a chart or list

Last updated on 04/03/2026

Set properties for a model-driven app chart or list included in a dashboard

Article • 02/28/2024

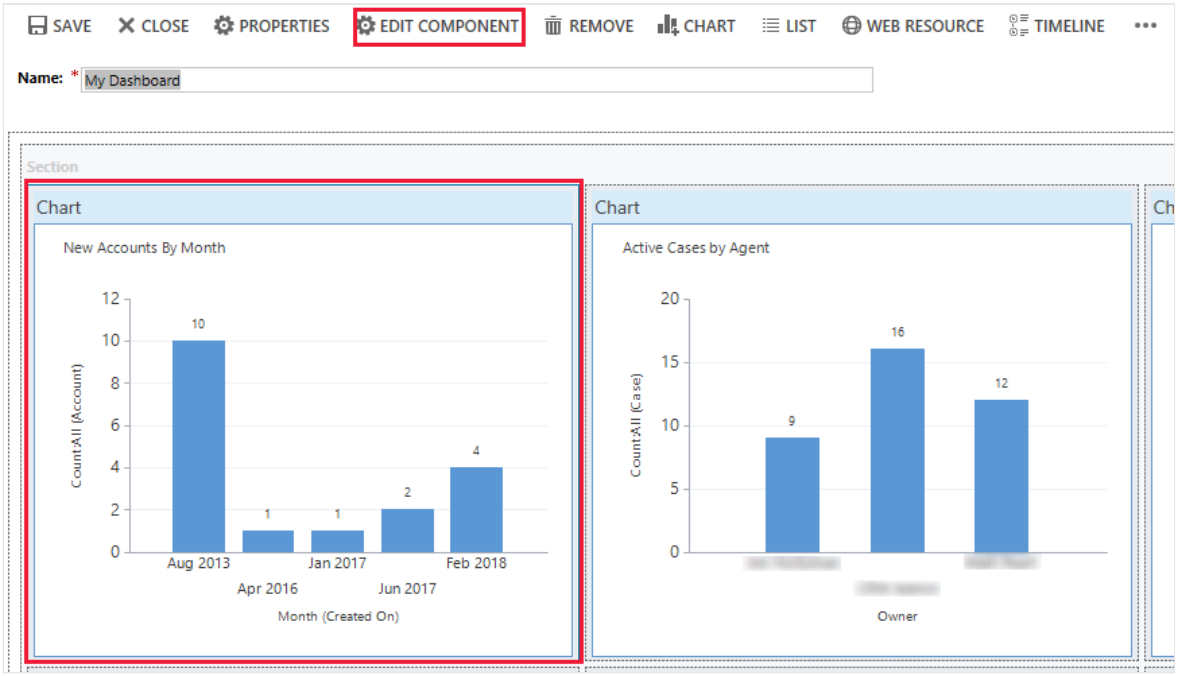
1. Sign in to Power Apps.

2. Select an environment with an unmanaged solution.

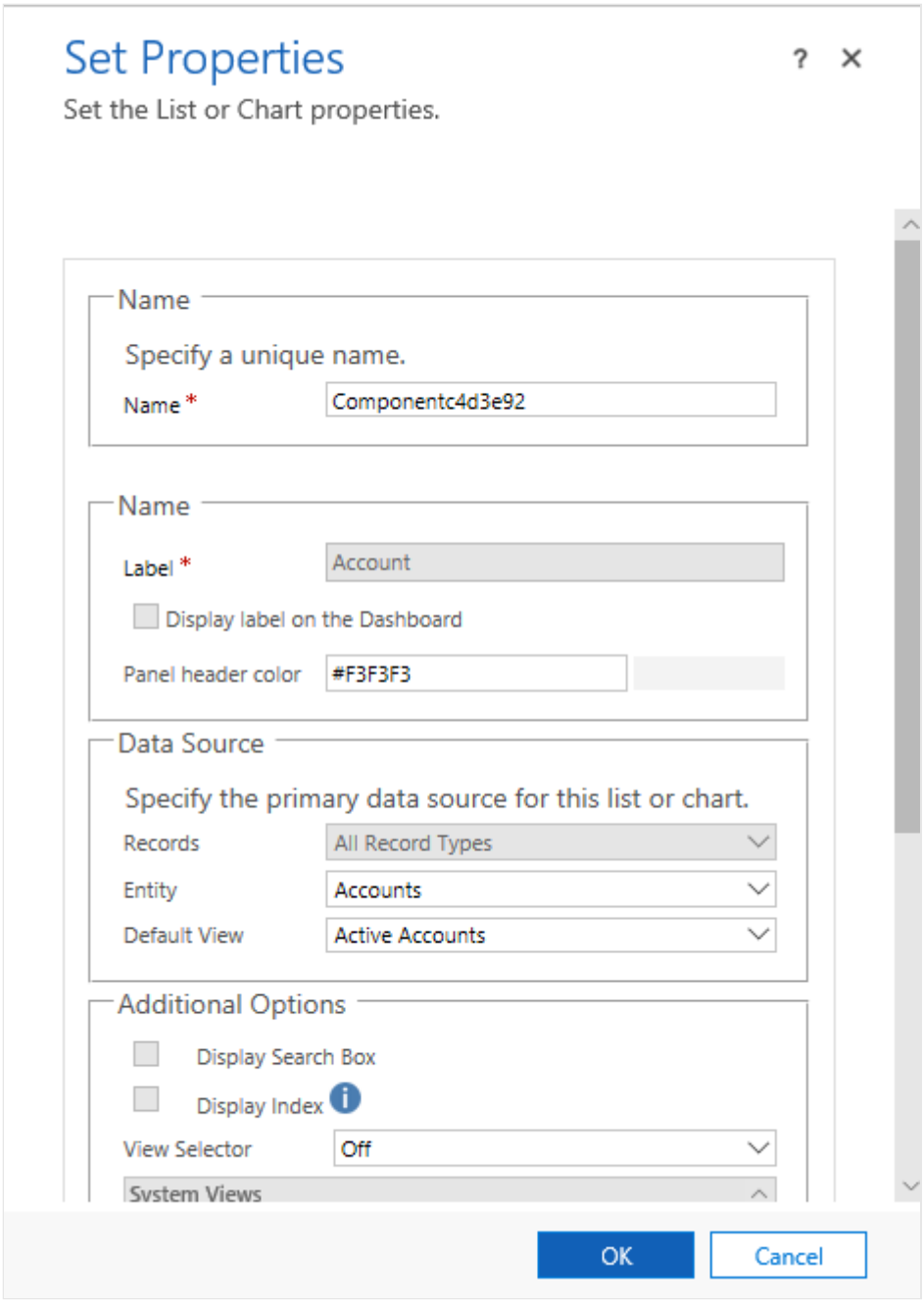
3. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.

4. Open the solution you want, on the tree menu select **Dashboards**, and then select the dashboard you want to update. This opens a new tab in your browser.

5. To edit a chart or list component from the dashboard designer, select the chart or list you want and then select **Edit Component** on the dashboard designer toolbar.



The **Set Properties** dialog opens.



6. You can set the following chart properties from the **Set Properties** dialog:

   - **Name**. Unique name for the chart. The system suggests a value, but you can change it.
   - **Label**. The label that appears at the top of the chart.
   - **Display label on the Dashboard**. Select or clear this check box to display or hide the chart label.
   - **Entity**. Select the table (row type) to base the chart on. This setting determines the available values for the Default View and Default Chart properties.
   - **Default View**. Select the view used to retrieve the data for the chart.

- **Default Chart**. Select the default chart that you want to display when the dashboard is first opened. The available values are determined by the value set for the Table property. This property works together with the Display Chart Selection property. A user can change the chart displayed in the component if the **Display Chart Selection** option is turned on, but the chart will revert to Default Chart the next time the dashboard is opened.

- **Show Chart Only**. Select this check box if you want to display just the chart. Clear this check box if you want to display the chart and its associated data.

- **Display Chart Selection**. Select this check box to enable users to change the chart displayed in the component at runtime. The component reverts to displaying the Default Chart next time the dashboard is opened.

7. You can set the following **list** properties from the **Set Properties** dialog box:

    - **Name**. Unique name for the list. The system suggests a value, but you can change it.

    - **Label**. The label that appears at the top of the list.

    - **Display label on the Dashboard**. Select or clear this check box to display or hide the list label.

    - **Table**. Select the table (row type) to base the list on. This setting determines the available values for the Default View property.

    - **Default View**. Select the view used to retrieve the data in the list. A user can change the view, but the list will revert to Default View the next time the dashboard is opened.

    - **Display Search Box**. Select this check box if you want to display a search box at the top of the list. If the search box is included, you or other users can search for rows in the list at runtime.

    - **Display Index**. Select this check box if you want to display the A to Z filters at the bottom of the list. When the A to Z filters are displayed, you or other users can select a letter to jump to rows that start with that letter.

    - **View Selector**. Select from the following values:

        - **Off**. Don’t display the view selector. You or other users won’t be able to change views at runtime.

- **Show All Views**. Provide a full list of views associated with the value set in the Table property.
- **Show Selected Views**. Select this setting to limit the list of views available at runtime. To select the specific views to be displayed, hold down the Ctrl key and tap or select each view you want to include.

Next steps

Create or customize dashboards

Add a Power BI visual to a model-driven app

Power BI is a powerful analysis and visualization tool. Power Apps is great at enabling people to take action via the web and mobile. With Power Apps, a Power BI dashboard can be embedded in an app. Similarly, a Power BI report tile can be added to an app. The best of both worlds can be achieved at the same place and at the same time.

In this article, you create a simple Power BI report and a model-driven app. Then, you learn how to add the report as a dashboard in the app.

Prerequisites

To complete this tutorial, you need the following:

- Ability to create Power BI dashboards and reports in Power BI.
- Enable the Power BI visualization embedding setting for your environment from the Power Platform admin center. More information: Manage feature settings

Create a workspace, dashboard, and report

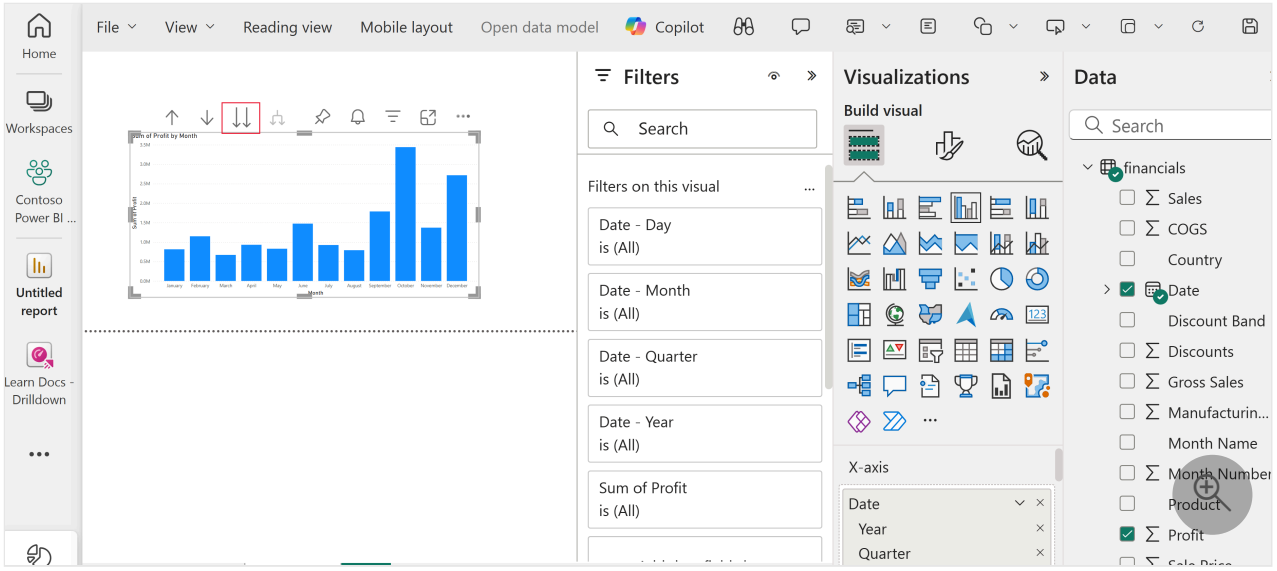
If you don't already have a workspace with a Power BI report, follow these steps to create one. If you already have a workspace, go to Add the dashboard and report to a solution.

1. Download the Excel financial sample Excel workbook.
2. Sign in to Power BI with the same Microsoft Office subscription as the one you use for creating apps in Power Apps.
3. Select **Workspaces** on the left navigation pane, and then select **New workspace**.
4. Enter a name for the workspace, such as *Contoso Power BI workspace*, and then select **Apply**.
5. Select **New item**, select **Dashboard**, enter a **Name**, and then select **Create**.
6. Select Workspaces on the left navigation pane and select the workspace you created. Select **New item** > **Report**.
7. Select **Excel**, select **Upload file**, select **Browse**, locate and select the **Financial Sample.xlsx** file you downloaded previously, and then select **Open**.
8. When the file is uploaded, select **Sign in**, and then select **Next**.

9. Select the **financials** tab, and then select **Create**.

10. Now, build a visual:

    - Expand **financials** in the right-hand **Data** pane, and drag **Profit** to the report canvas.
    - Drag **Date** and drop it on top of the **Profit** chart.
    - Drag an edge of the chart to resize it.
    - Select the chart and then select the forked double arrows to show quarters, and then select them again to show months.



11. Save your report as *Financial Samples*.

Add the dashboard and report to a solution

In this section, you create a solution in Power Apps that includes a Power BI embedded dashboard object.

1. Sign in to Power Apps. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Create a new solution or open an existing unmanaged solution.

3. Open the solution and select **New** > **Dashboard** > **Power BI Embedded**.

4. In the right properties pane, enter or select the following properties:

    - **Display name**: *Test Embedded BI Report*
    - **Type**: **Power BI report**
    - Clear the **Show reports in this environment only** option.
    - **Power BI workspace**: Locate and select the workspace you created earlier, such as **Contoso Power BI workspace**.

- **Power BI Report**: Locate and select the **Financial Sample** report you created for the workspace.

5. Select **Save**.

6. Select **Publish** on the command bar.

Create a model-driven app and add the Power BI report

In this section, you create a model-driven app and add the Power BI embedded report you created earlier.

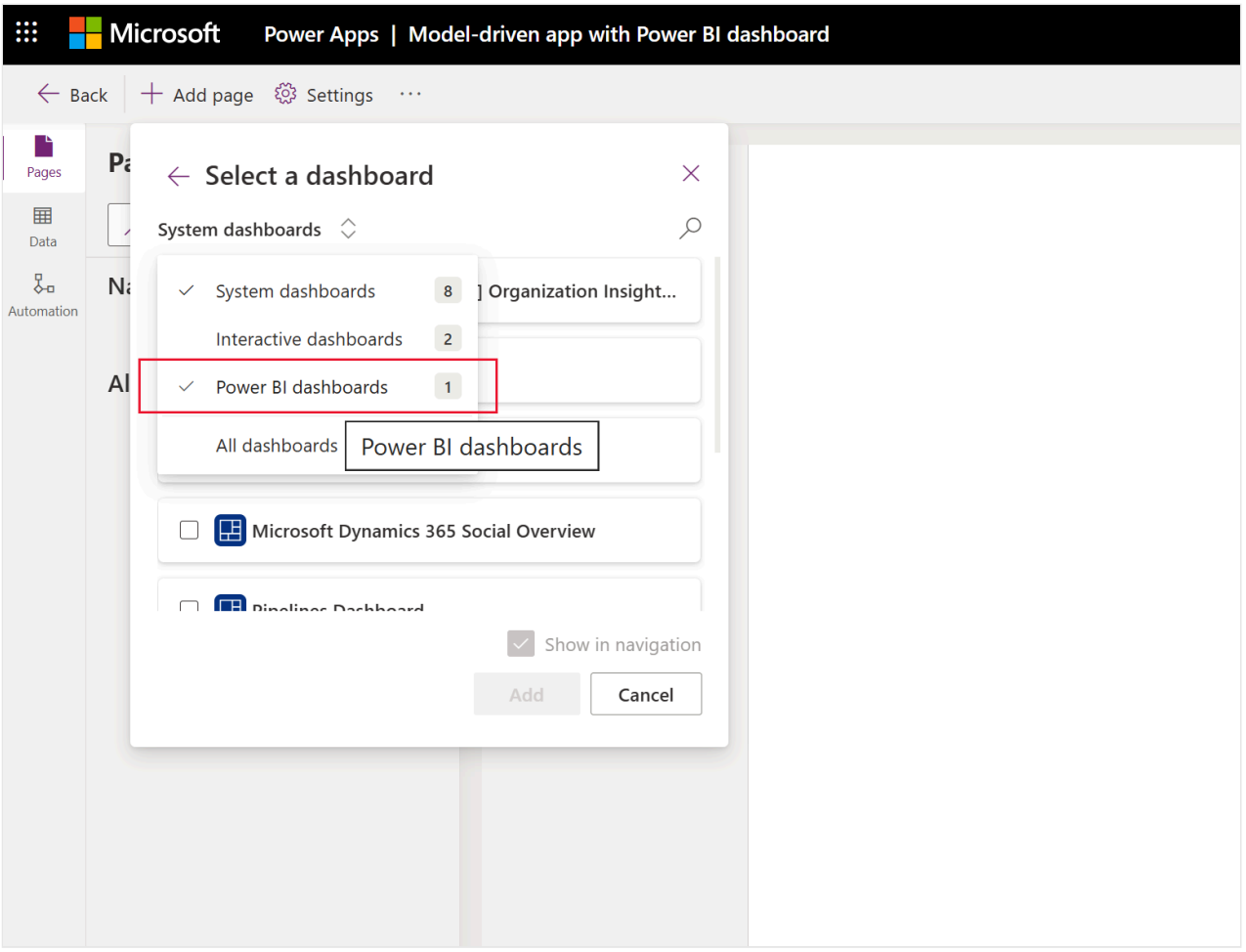
1. Sign in to Power Apps, select the environment you want, and then on the left navigation pane, select **Solutions**.

2. Open the solution you created to add the embedded Power BI dashboard.

3. On the command bar, select **New** > **App** > **Model-driven app**.

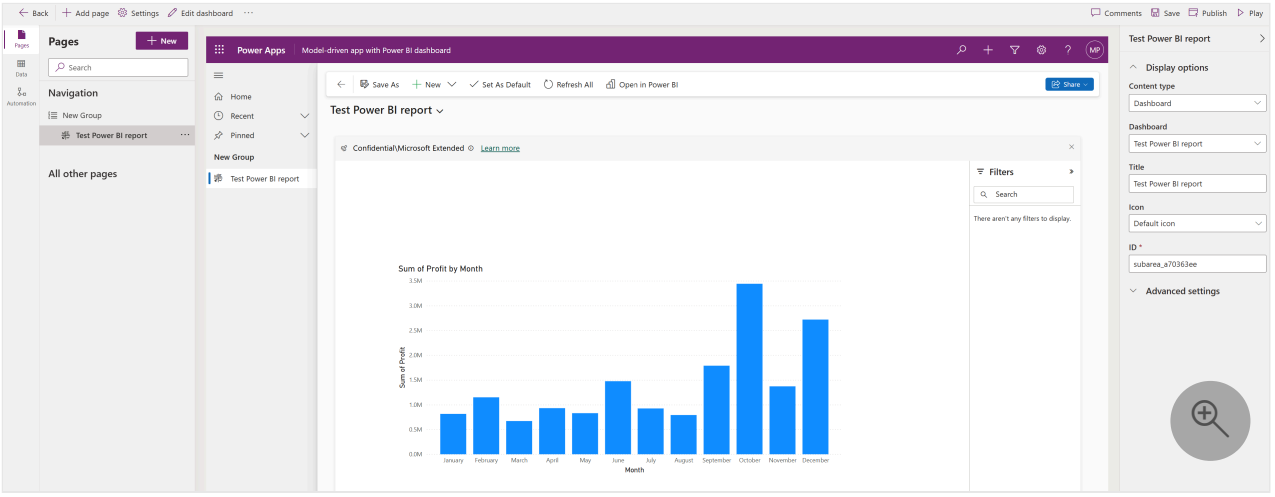
4. Enter a **Name**, such as *Model-driven app with Power BI dashboard*, and then select **Create**.

5. In the app designer on the command bar select **Add page**, select **Dashboard**, and then in the dashboard picker select **Power BI dashboards**.



6. Select the **Test Power BI report** you created earlier, and then select **Add**.

   The report loads in the app designer similar to how it appears to your users when they play the app.



7. Select **Save and publish** to save and publish you app making it available for users.

See also

[Use Power BI with model-driven apps]

Last updated on 03/31/2026

Model-driven app reporting capabilities and considerations

Article • 12/21/2023

Model-driven apps have many capabilities that allow customers to surface business data that helps them drive decisions and interact with their customers and their data more effectively.

These are described in the reporting overview and include

- Views.
- Charts.
- Dashboards.
- SQL Server Reporting Services reports.
- Microsoft Excel integration that allows users to easily build self-service reports using the Power BI features PowerView, PowerPivot, and PowerQuery.
- Power BI.

As the volume of data held in the app's database continues to grow it becomes more important than ever to think about your BI strategy and determine the most effective mechanisms for reporting and visualizing large datasets.

Reporting infrastructure

In an environment, the reporting infrastructure is shared and separate from the database. In this architecture, although customers share the resources required to run the report, each report runs against the customers' individual database instance.

The reporting capabilities built in to Microsoft Dataverse are designed to let users run reports on datasets that span shorter periods of time. Considering this, note the following fixed settings:

- Reports and queries can execute for up to five minutes. When the maximum period is reached, the report will time out and a message is returned to the user. Within the five-minute duration, reports and queries are allowed to span large datasets that are beyond 50,000 rows, which provide significant flexibility to satisfy most operational reporting needs.

- To improve query response, we recommend that detailed reports minimize the display of large numbers of rows. To do this, apply suitable filtering to reduce the number of rows that are returned. When you create aggregated or summarized

reports, queries should push the aggregation to the query rather than fetch detailed rows to perform aggregation in the report. More information: Aggregate data using FetchXml and report prefiltering.

- The Report Viewer page is subject to the platform API limits. Multiple executions of a long-running report might result in an error, and the user can wait few minutes and try again. This is a per user limit and shouldn't affect the normal usage of reports by multiple users.
- For charts and grids displayed in dashboards, your apps allow users to run queries that have a dataset that has fewer than 50,000 rows. Should a user run a dashboard query that spans a dataset of 50,000 or more rows, the message "The maximum row limit is exceeded. Reduce the number of rows" is returned. The dataset practical setting helps to ensure optimal performance of the app.

Tips and solutions for reporting

Typically, for most organizations' reporting needs, these settings are adequate. To make sure that your users don't exceed these settings and to improve report querying performance in general, consider the following best practices.

- When creating custom reports or dashboards, design them to query smaller datasets over shorter periods of time by adding a time-based filter in the report, such as the current month or quarter, to limit the results.
- Limit the number of tables that are needed to return the result. This helps reduce the time required to run the query and return the result set.
- Reduce the rows in detailed reports. Suitable filtering can be used to reduce the number of rows returned by the query to reduce timeouts.
- For aggregated or summarized reports, queries must be used to push the aggregation to the database and not fetch detailed rows and perform aggregation in the SQL Server Reporting Services report.
- When appropriate for your business, users should run the default (out-of-the-box) reports and dashboards. These reports and dashboards are typically designed to query per user datasets, so in most cases won't exceed the dataset limit.

If users must run reports that exceed these settings, we recommend that you review the following options for assistance with complex reporting needs. Both options effectively offload reporting workloads from Dataverse to another datastore by using a data integration solution.

- Adapters are used in conjunction with SQL Server Integration Services (SSIS) to extend the capabilities for integration with your apps data.
- Extract transform load (ETL) tools provide a new tool set for creating analysis of data by combining multiple data sources or extracting data to the data warehouse solution if SSIS isn't in use. ETL tools provide comprehensive solutions for connecting with Dataverse to move data.

> **Important**
>
> When you use these tools, we recommend that moving or synchronizing data takes place during nonbusiness hours.

If needed, there are many Microsoft partners who can help provide a solution for your specific reporting needs, such as creating an offline copy of the data specifically used for running large reports. These partners are knowledgeable with the data integration tools available. More information: Find a Dynamics 365 partner

Third-party adapters for SSIS

- CozyRoc SSIS+ component for Dynamics 365/CRM
- KingswaySoft SSIS Integration Toolkit for Dynamics 365
- CData SSIS components for Dynamics 365
- Team4 SSIS Connector for Dynamics 365

ETL tools

- TIBCO Dynamics 365 integration

See also

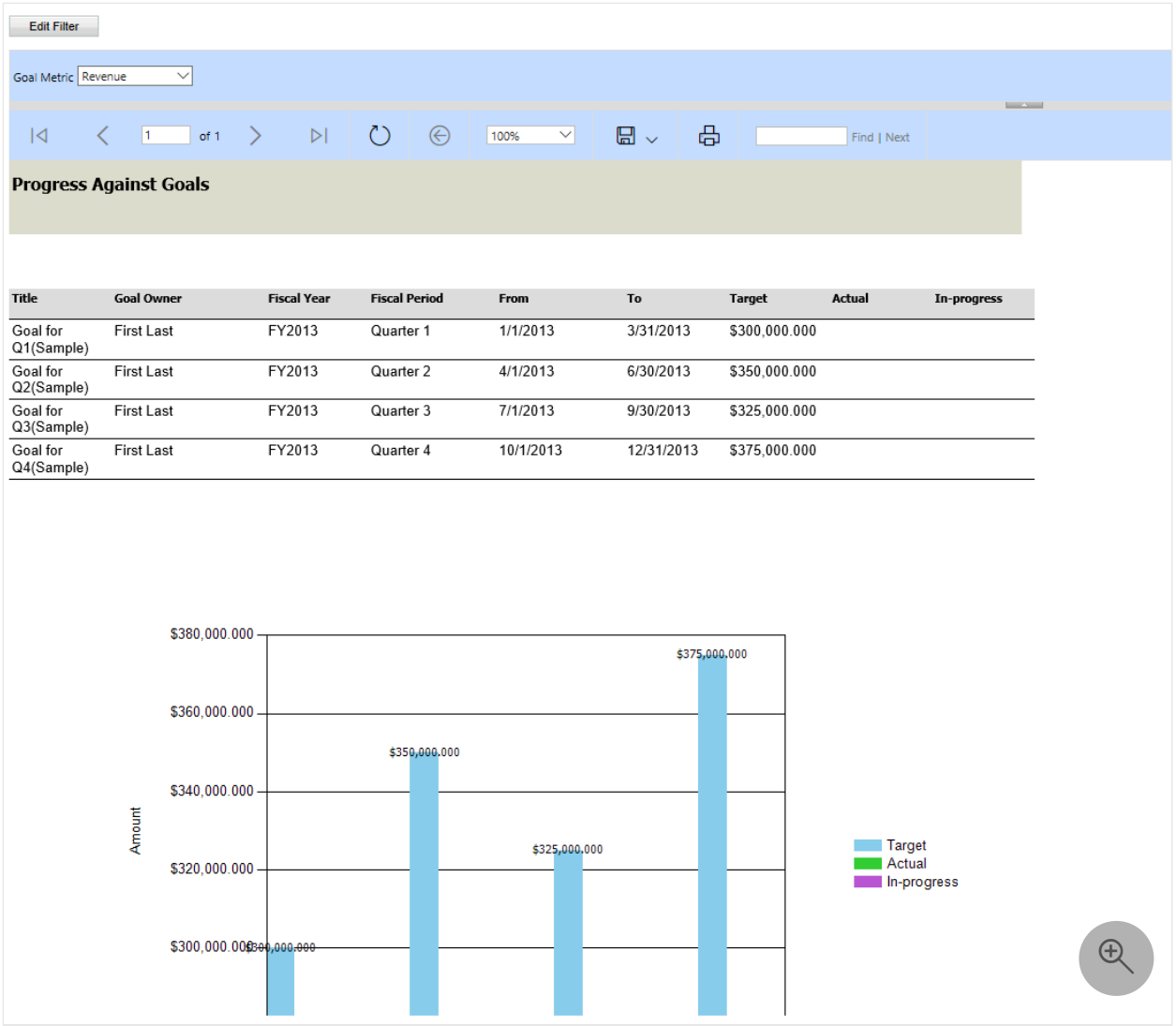
Report Authoring Extension (with SQL Server Data Tools support)

Introduction to Microsoft Power Query for Excel
Dynamics 365 for Customer Engagement OData Feeds and Power Query: What's the [Record]?

Add reporting features to your model-driven app

Article • 09/18/2024

Power Apps apps can include paginated style reports that provide useful business information to the user. These reports are based on SQL Server Reporting Services and provide the same set of features that are available for typical SQL Server Reporting Services reports.



System reports are available to all users. Individuals who create or otherwise own reports can share them with specific colleagues or teams, or can make the reports available to the organization, so that all users can run them. These reports use FetchXML queries that are proprietary to Microsoft Dataverse and retrieve data to build the report. Reports that you create in a Power Apps app are Fetch-based reports.

> **Note**

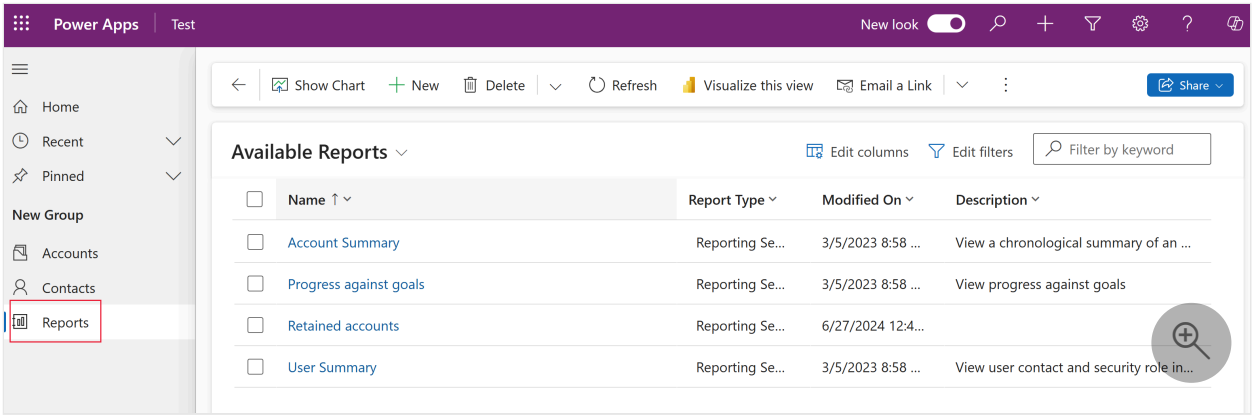
Report features don't work with canvas apps or model-driven apps running on mobile devices, such as tablets and phones.

Add reporting to a model-drive app

You can add fetch-based reporting functionality to your app so that users can run, share, create, and edit reports. To do this, you add the report table to your app.

1. Sign in to Power Apps select **Solutions**, open the solution you want, and then open the model-driven app where you want to add reporting for editing.
2. In the app designer, select **New** > **Dataverse table**.
3. In the **Select a table** list, select the **Report** table, and then select **Add**.
4. Select **Save**, and then select **Publish**.

Now the app displays a **Reports** page where users can view, run, assign, share, and edit the reports they have permission to as well as create new reports using the report wizard.



Options for creating new reports

You can create a new report in one of two ways:

- Use the Report Wizard. Open a model-driven app that is enabled for reporting, select the **Reports** page, and then select **New** to open the Report Wizard to create a new report. The Report Wizard can create table and chart reports, including drill-through reports and top N reports. More information: Create a report using the Report Wizard
- Use the Report Authoring Extension. You can write new or customize existing fetch-based Reporting Services reports with Visual Studio, SQL Server Data Tools, and the Report Authoring Extension. More information: Create a new report using SQL Server Data Tools

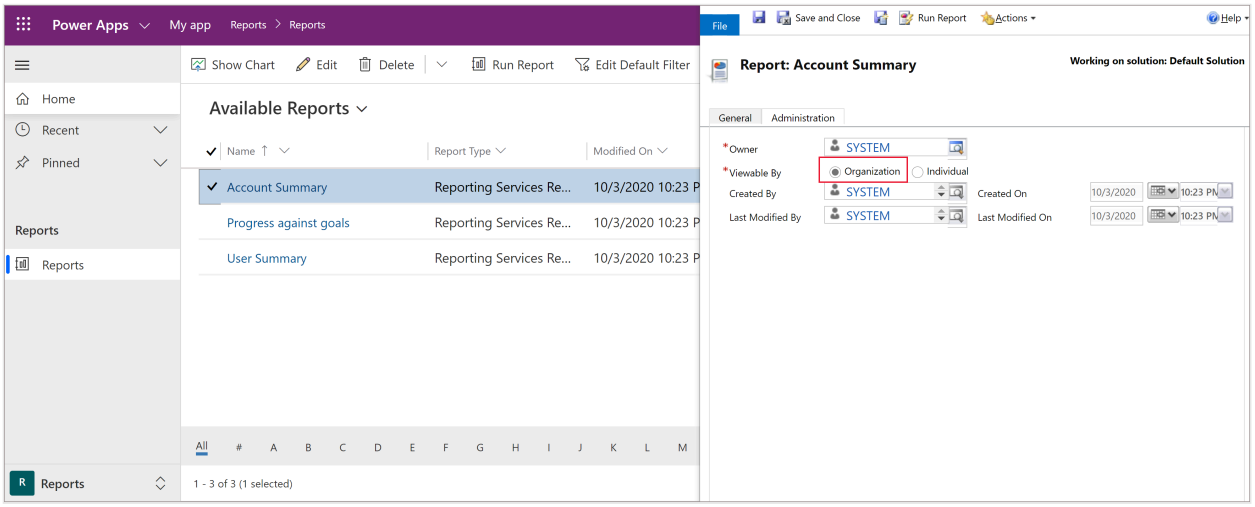
Report visibility

Standard table reports, such as the Accounts Summary report for the account table, are available to all app users. Users who own reports can share them with specific colleagues or teams. System administrators and system customizers can make reports available with organization-wide visibility, so that all users can use them. For information about how to share a report, see Share a report with other users and teams.

Reports in solutions

Reports are solution aware. Adding a report as a component to a solution makes it become a single unit of software that extends Power Apps functionality and the user interface. Only reports that are visible to the organization can be added to solutions.

To find if a report is viewable to the organization: In the list of reports, open a model-driven app, select a report, and then select **Edit**. On the **Administration** tab, determine whether **Viewable By** is set to **Organization**.



You can add, import, or export snapshots of reports as part of a solution. In model-driven apps, reports, sub reports, report category, report display area, and report-related row type are considered as components of a report set. When you import a solution update in non-overwrite mode, any updates by the solution to a report will be ignored if any component of the report set has been customized.

Related articles

Work with reports
Create a report using the Report Wizard
Add a report from outside Power Apps
Edit the default filter of a report

Feedback

Was this page helpful? Yes No

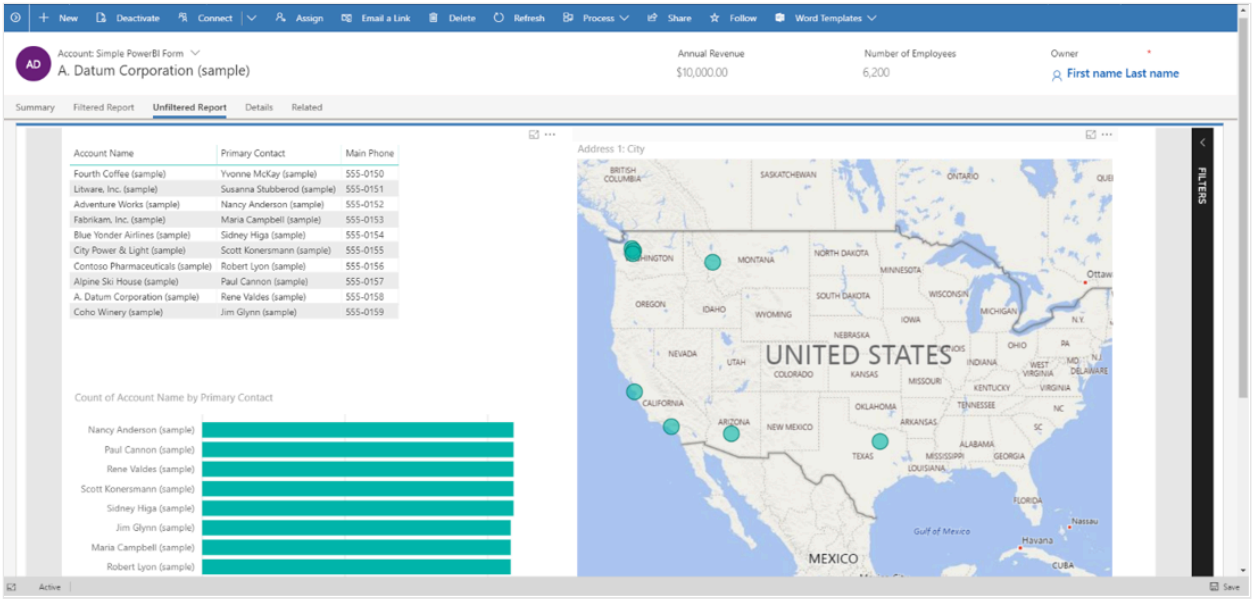
Provide product feedback

Use Power BI

Article • 12/06/2024

The Power BI cloud service works with Microsoft Dataverse apps to provide a self-service analytics solution. Power BI automatically refreshes the app's data displayed. With Power BI Desktop or Microsoft Excel, Power Query for authoring reports and Power BI for sharing dashboards and refreshing data from model-driven apps, your users have a powerful way to work with your app's data. For more information, go to these articles:

- Visualize data in a view with Power BI
- Create or edit a Power BI embedded system dashboard
- Embed a Power BI report in a model-driven system form
- How app users can add or edit Power BI visualizations on their dashboards



See also

Use Power BI with Dataverse data
Connect to Power BI from Power Apps canvas apps

Feedback

Was this page helpful? Yes No

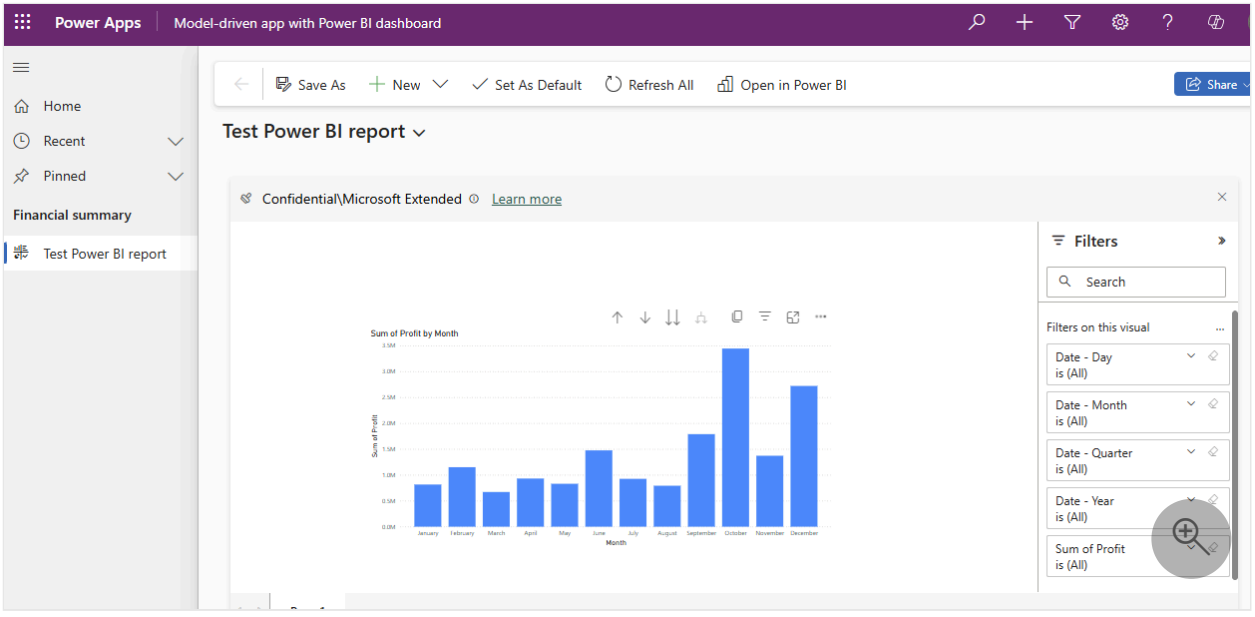
Provide product feedback

Create or edit a Power BI embedded system dashboard

Article • 02/18/2025

This article explains how to set up and use a system dashboard that references a Power BI report or dashboard in a model-driven app. System administrators and system customizers can create system reports and dashboards, which can be made available to model-driven app users. Users need to have the relevant Power BI license and authorization to consume content.

> **Note**
>
> Power BI embedded in a system dashboard or form ignores the Power Platform admin center **Power BI visualization embedding** environment setting. That setting controls whether end users can add embedded Power BI reports to personal views, charts, and dashboards.



Read more about relevant Power BI licenses at Power BI pricing.

To learn more about Power BI reports and dashboards, go to Create reports and dashboards in Power BI.

Create a system dashboard with Power BI embedded

This procedure shows you how to set up a Power BI embedded page for a single environment connected to a Power BI workspace.

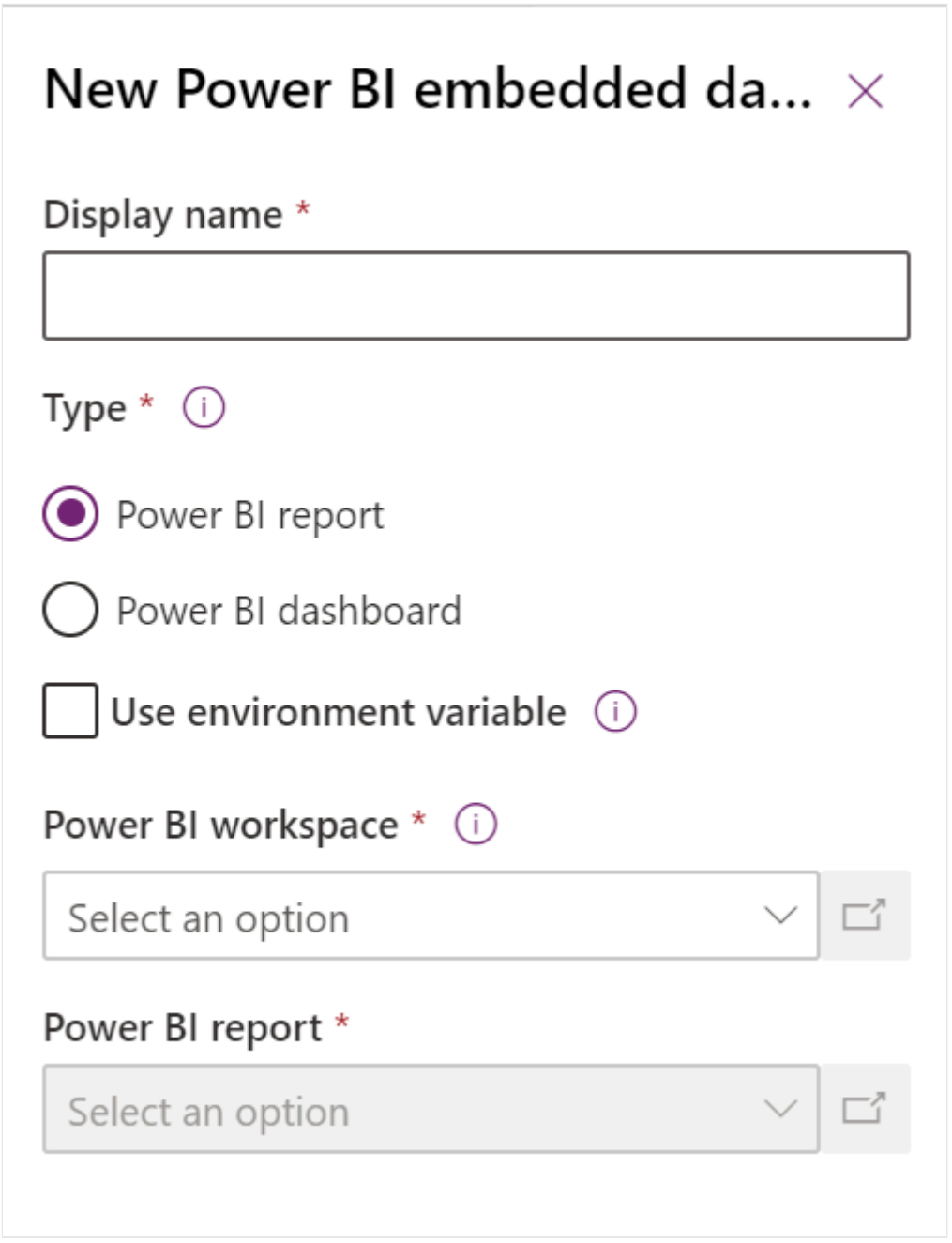
1. Select **Solutions** on the left navigation pane in Power Apps (make.powerapps.com), and then create or open an existing solution. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Select **New** > **Dashboards** > **Power BI embedded**.

3. Enter or select the following properties in the **New Power BI embedded** dialog that appears:

    - Enter a **Display name** for the dashboard, which will be shown to the user in the selector on the **Dashboards** page.
    - Select the type, either **Power BI report** or **Power BI dashboard**.
    - If you clear **Show reports in this environment only**, you can select a Power BI report or a Power BI dashboard from another workspace.
    - Select **Use environment variable** if you want to replace a static workspace and report to use an environment variable from configuration. You use this option when you want to move the report to another environment in a solution. More information: Environment variables overview and Create Power BI embedded page with an environment variable
    - Select a **Power BI workspace**.

- Select a Power BI report or dashboard.



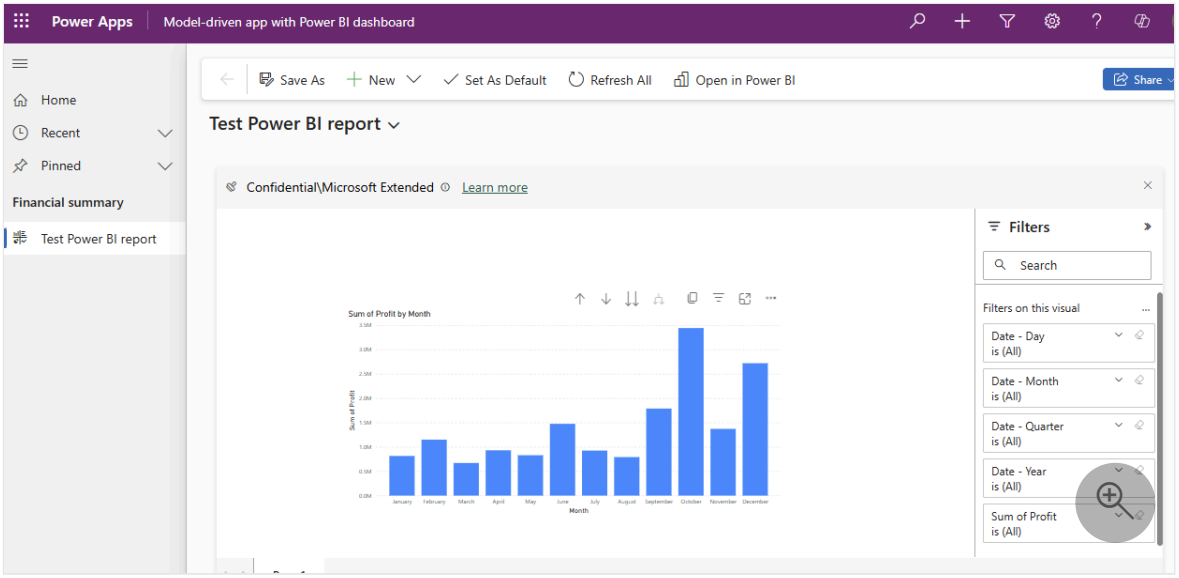
4. Select **Save** to save and publish the system dashboard.

Add a Power BI embedded page to a model-driven app

1. Create or edit an existing model-driven app using the app designer.
2. In the app designer select **Add page** > **Dashboard**.
3. Select the **System dashboards** picker, and then select **Power BI dashboards**.
4. Select the Power BI dashboard you want, and then select **Add**.
5. Select the **Dashboards** subarea on the app designer canvas.
6. Select **Save**. To make your changes available to users, select **Save and Publish**.

Select the Power BI embedded page in the model-driven app

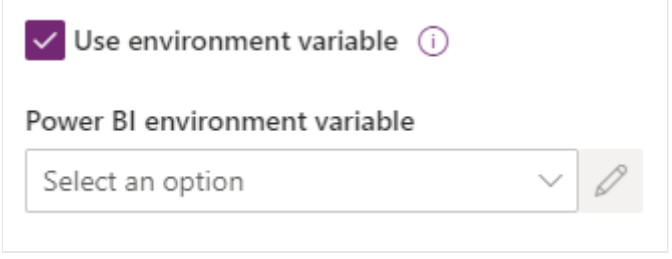
1. Play the model-driven app, and then select a dashboard from the left navigation pane.
2. Select the Power BI embedded page to perform actions, such as expand, copy as image, and filter.



Create Power BI embedded page with an environment variable

When a solution with a Power BI embedded component will be moved to other environments, you can turn on **Use environment variable** to configure the dashboard. For each environment, you specify the workspace and either a dashboard or report to reference. Environment variables allow configuration without unmanaged customizations.

1. When you create or edit an existing Power BI embedded system dashboard in Power Apps (make.powerapps.com), select **Use environment variable**. create-edit-powerbi-embedded



2. Open the **Power BI environment variable** selector and then select **New environment variable**. 

   A nested panel opens where you create the environment variable. The environment variable properties are automatically filled using the dashboard **Display Name** and **Name**.

   You can configure the workspace and report or dashboard. The values are stored as JSON. More information: Environment variables overview

New environment variable

Use this variable to store information about an app or flow. Its values can be updated as it moves to different environments. Learn more

Display name *

Custom Report

Name *

contoso_ CustomReport

Description

Data Type *

{} JSON

Default workspace value *

Select an option

Default report value *

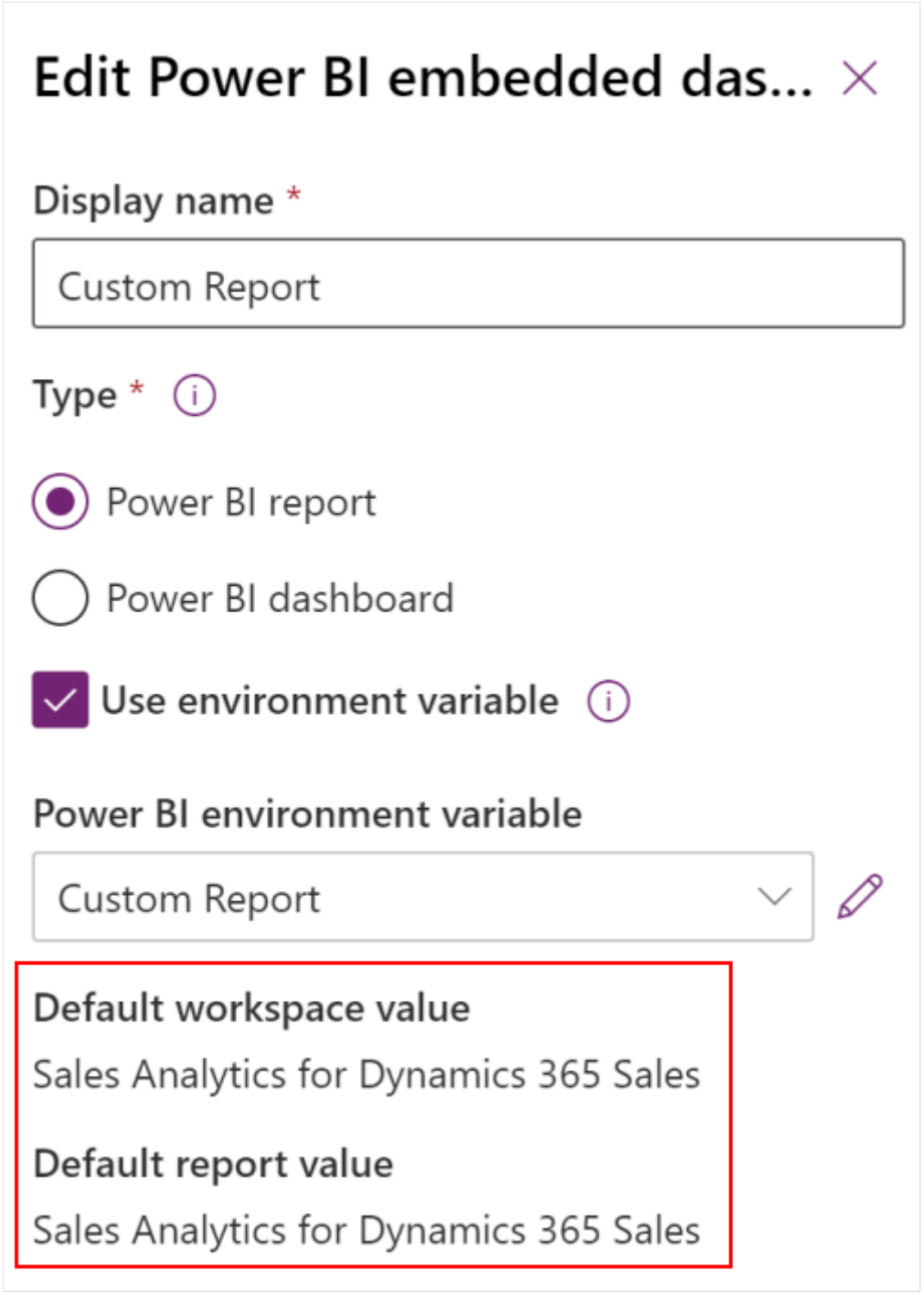
Select an option

Current Value

Override the default value by setting the current value for your environment.

+ New value

3. Save the environment variable. The default and current values for workspace and report or dashboard are displayed.



See also

Create or edit model-driven app dashboards
View table data in Power BI Desktop
Use Power BI
Use Power BI with Microsoft Dataverse
Create a Power BI report using the Common Data Service connector

Feedback

Was this page helpful? Yes No

Provide product feedback

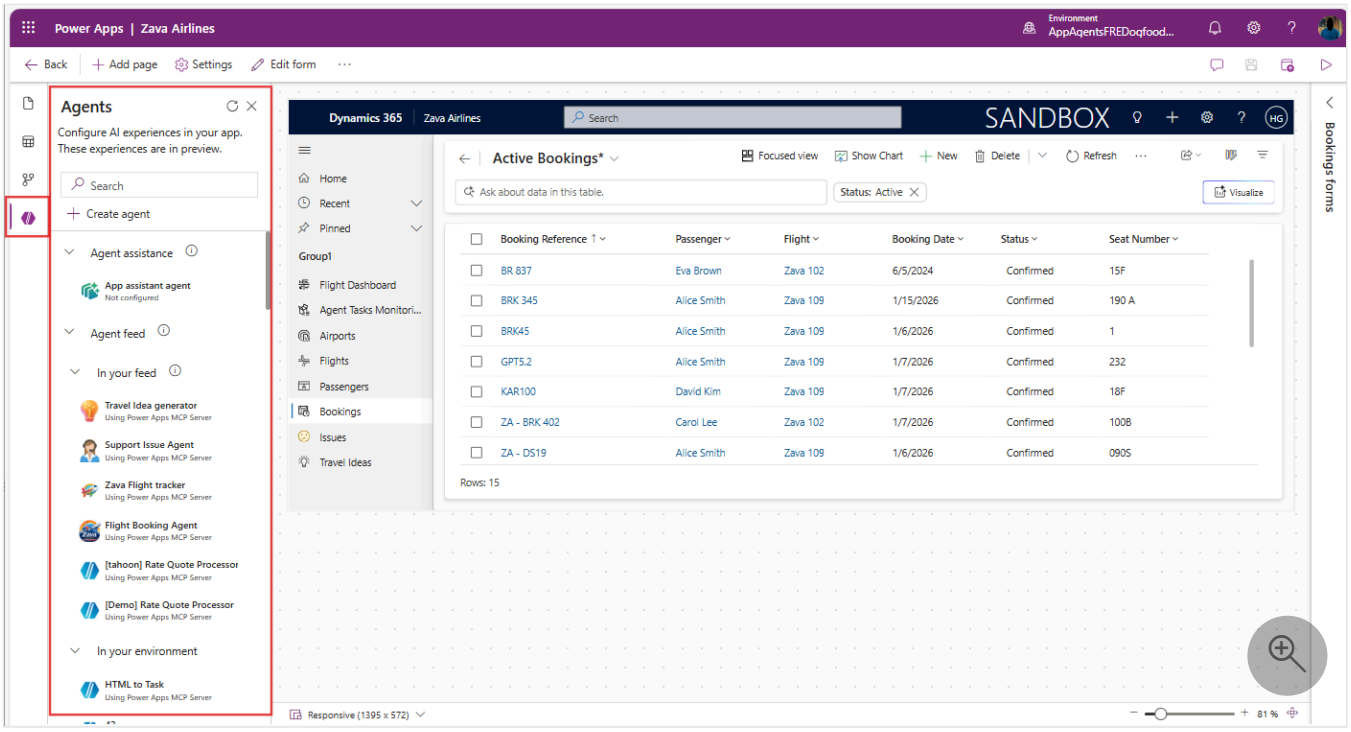
Add agents to your model-driven app (preview)

[This article is prerelease documentation and is subject to change.]

> **ⓘ Important**
>
> Starting May 1st, 2026, agent feed only supports agents that use the Power Apps MCP Server to create tasks. Ensure your agents are properly onboarded to the Power Apps MCP server by then to continue using the agent feed. If your agents don't use the Power Apps MCP Server, the agent feed doesn't appear in you model-driven app. More information: Onboard your agent feed to use the Power Apps MCP server

Model-driven apps support the use of agents to enhance user productivity and automate tasks. There are two types of agents available in model-driven apps:

- *Autonomous agents*, created in Microsoft Copilot Studio, collaborate with users through the new Power Apps model context protocol (MCP) server, allowing agents to generate actionable agent feed tasks for users to review results or step in with inputs to unblock and complete human-in-the-loop flows.
- *App assistant agents*, which can be created to provide custom topics, knowledge sources, and more within the model-driven app.



> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is available only in the English language and it replaces the earlier Microsoft Copilot Studio activity-based agent feed.
> - For information about how AI is used with this feature, go to FAQ about Power Apps MCP server invoke_data_entry tool.

Autonomous agents can be added to model-driven apps to help users get work done faster. Using the Power Apps MCP server, agents can post actionable tasks for user review, allowing humans to validate results, step in when needed, and finish work the agent couldn't complete—without leaving their app.

Learn more about how to supervise agents

> **Note**
>
> Access to the agent feed and supervision capabilities is limited by default to the System Administrator and System Customizer roles. To allow additional users to view the agent

feed, grant organization-level read/write permissions on the entities listed below. You can create a new security role with these permissions and assign it to multiple users as needed.

- Agent Hub Goal(agenthubgoal)
- Agent Hub Insight(agenthubinsight)
- Agent Hub Metric(agenthubmetric)
- Agent Task(agenttask)
- Copilot(bot)

Supervise an autonomous agent within the app

The enhanced agent feed is powered by the autonomous agents using the Power Apps MCP server. Power Apps MCP sever tools enable these two core human-agent collaboration patterns:

- Autonomous Microsoft Dataverse record creation and updates with human-in-the-loop review.
- Agent requests for human assistance and logging tasks for human review.

In agent-enabled apps, user focus shifts from doing the work to supervise and prioritize agent-driven work. Agents help with automation and organization of work, ensuring business experts remain involved in decision-making and critical actions. You can supervise elligible autonomous agents with any model-driven app. We recommend adding agents to apps that contain functionality or data relevant to the agent's intended purpose.

Terminology

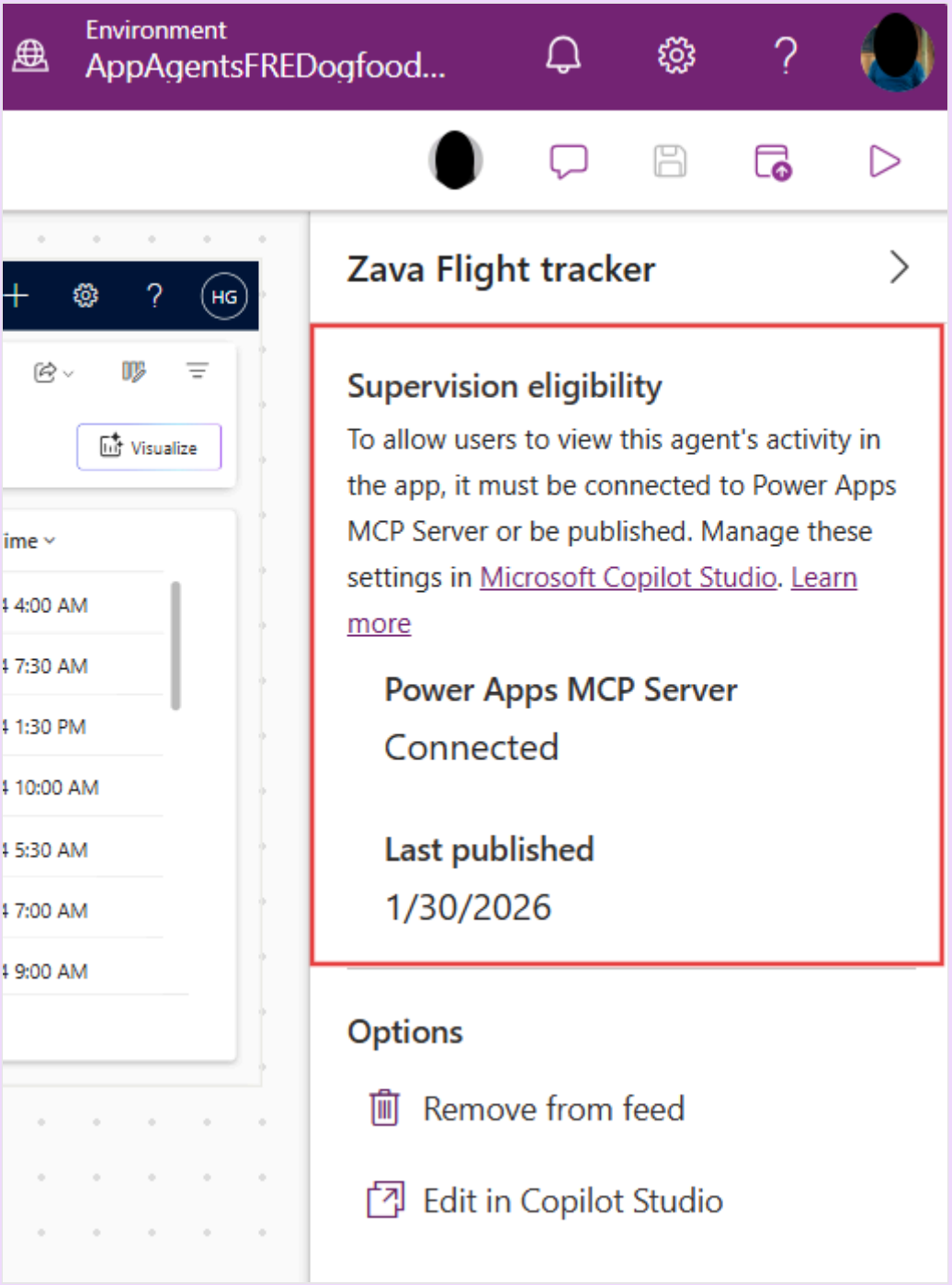
Expand table

Term	Description
Agent	The customer's custom autonomous agent.
Task/Agent task	A single request for human assistance or human review within an agentic workflow.
Agent feed	The system for surfacing agent tasks to the user.
Agent canvas	The UX surface where users provide the requested human input or observe task details.

Term	Description
Power Apps MCP server	The collection of tools available to agents for interacting with capabilities of the business app platform.

> **Note**
>
> To be eligible for addition to an app, an agent must be connected to Power Apps MCP server or be in a published state. A maker can verify an agent's eligibility to be added to an app via the right-hand properties pane where the requirements for an agent to be added to an app are displayed and includes a **Power Apps MCP Server** status as **Connected**. The **Add to feed** button is disabled for any agents that don't meet the

requirements to be eligible for addition.



Create an autonomous agent connected to Power Apps MCP server

MCP is an open protocol that enables seamless integration between large language model (LLM) applications and external data sources and tools. Your agent can use the Power Apps MCP server to communicate with your apps in Power Apps, providing right human-in-the-loop supervision or agentic workflows. To use the Power Apps MCP server, you need to enable and configure the MCP server with an agent. More information: Work with Power Apps MCP server

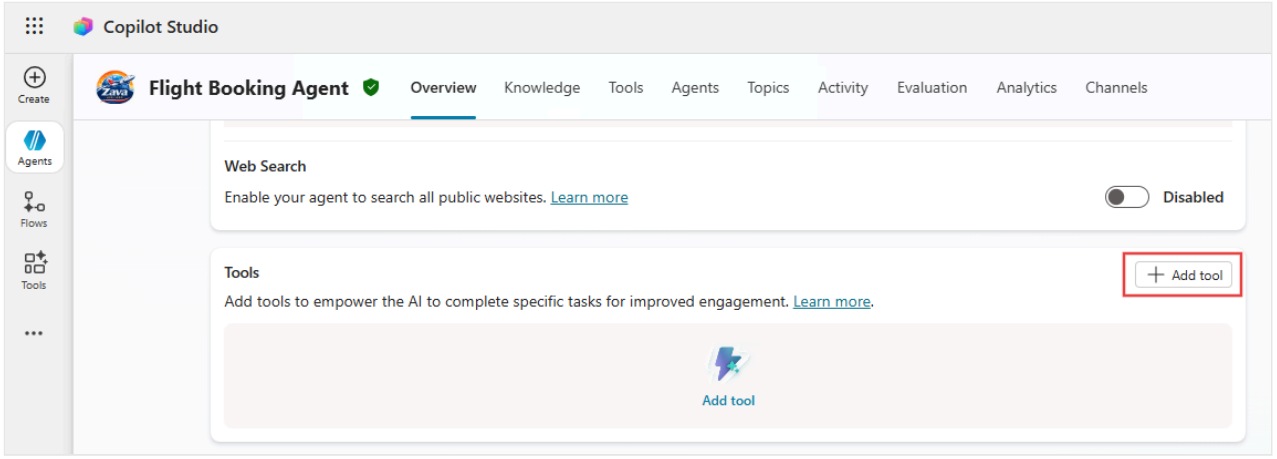
1. Sign in to Power Apps, select **Apps**, and then select **Edit** for the app you want to modify.

2. In the app designer, go to the **Agents** tab on the left navigation pane.

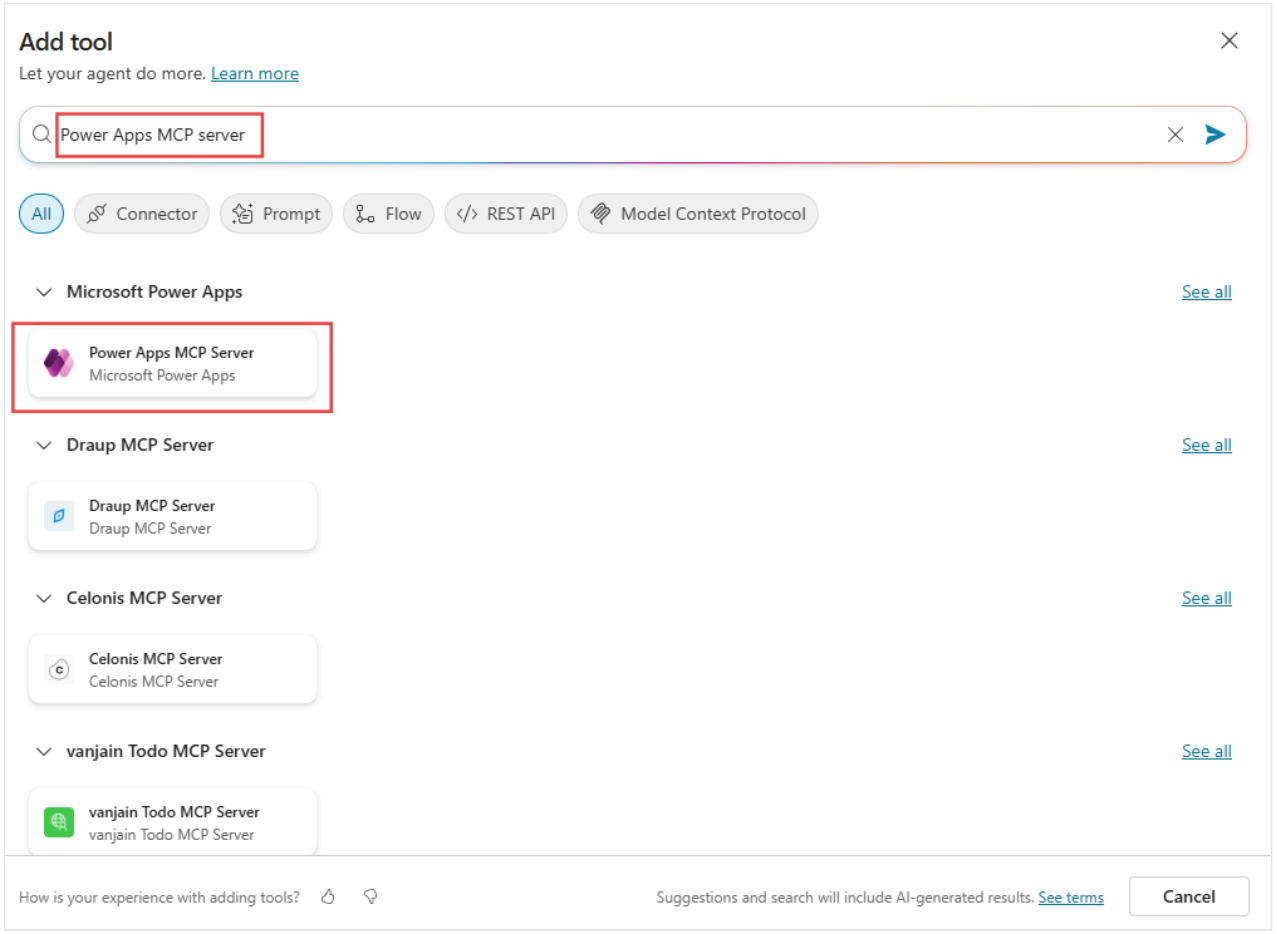
3. Select **Create agent**.

   Microsoft Copilot Studio opens in the new tab.

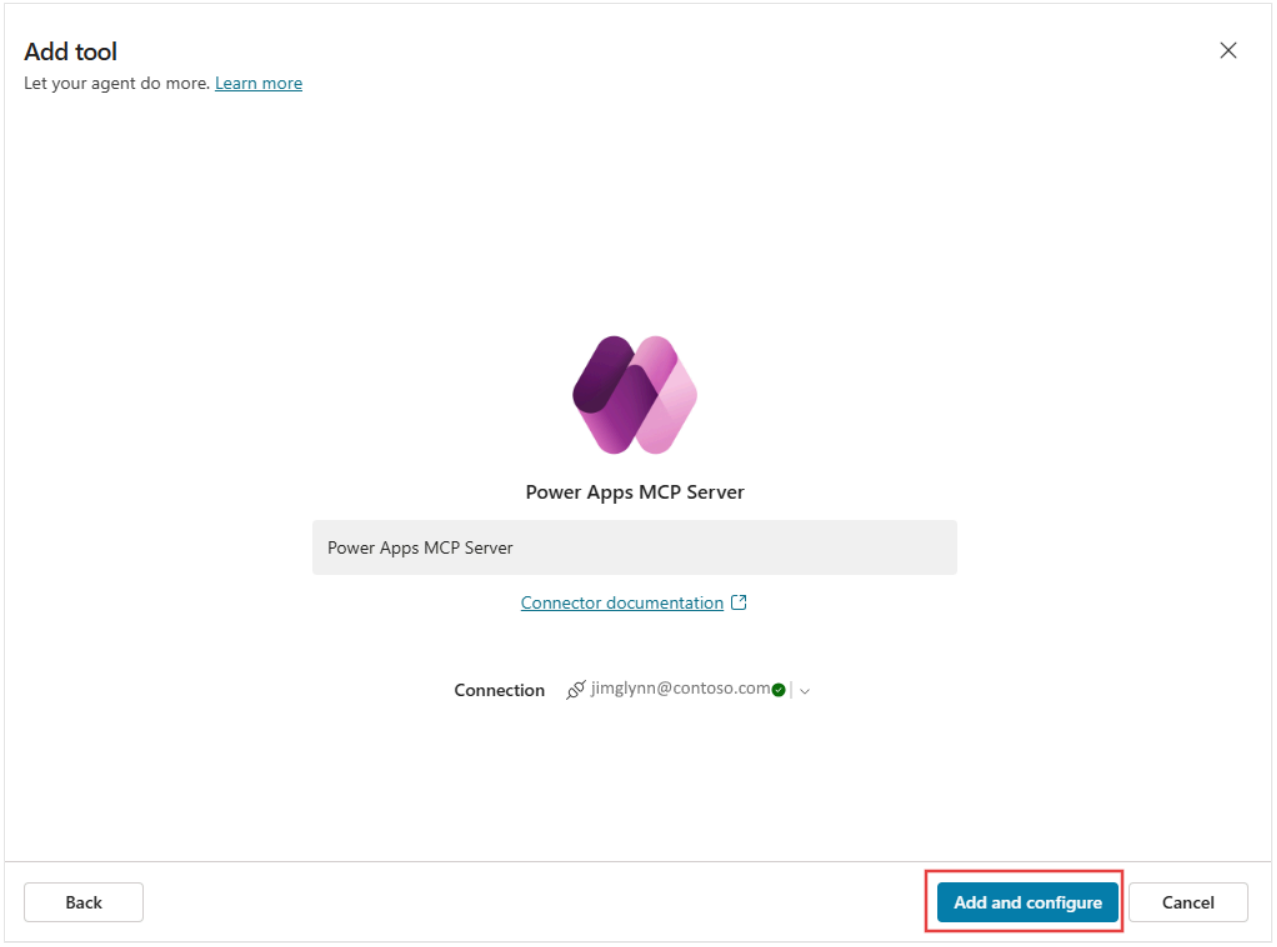
4. Enter the name and description and scroll to tools and select **Add tool**.



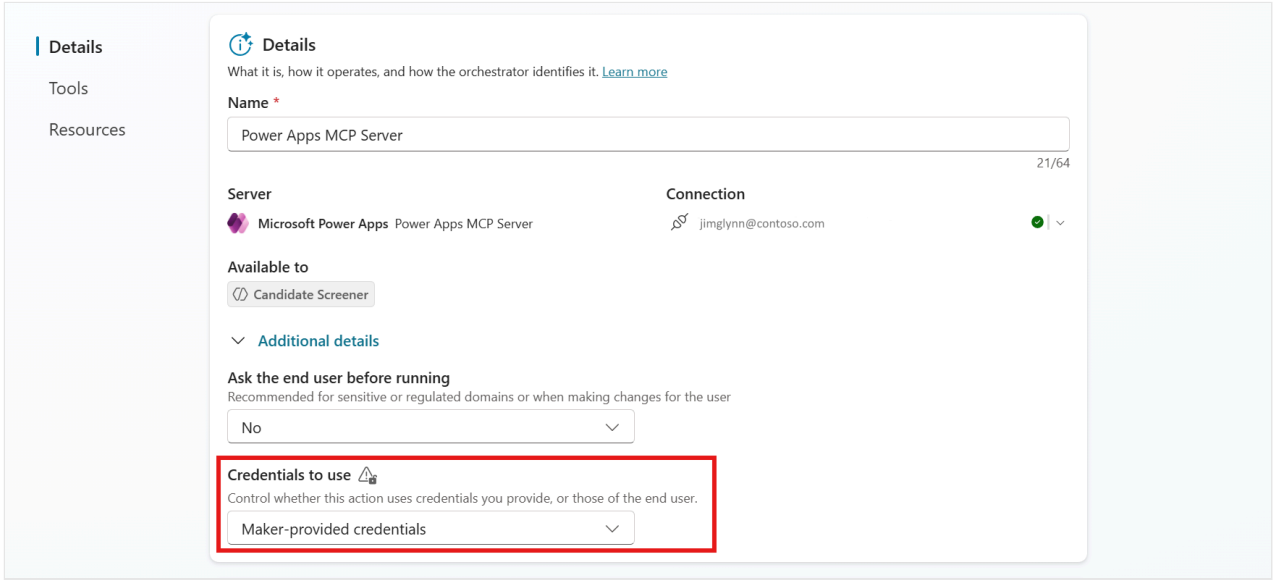
5. Search for **Power Apps MCP Server** and select it.



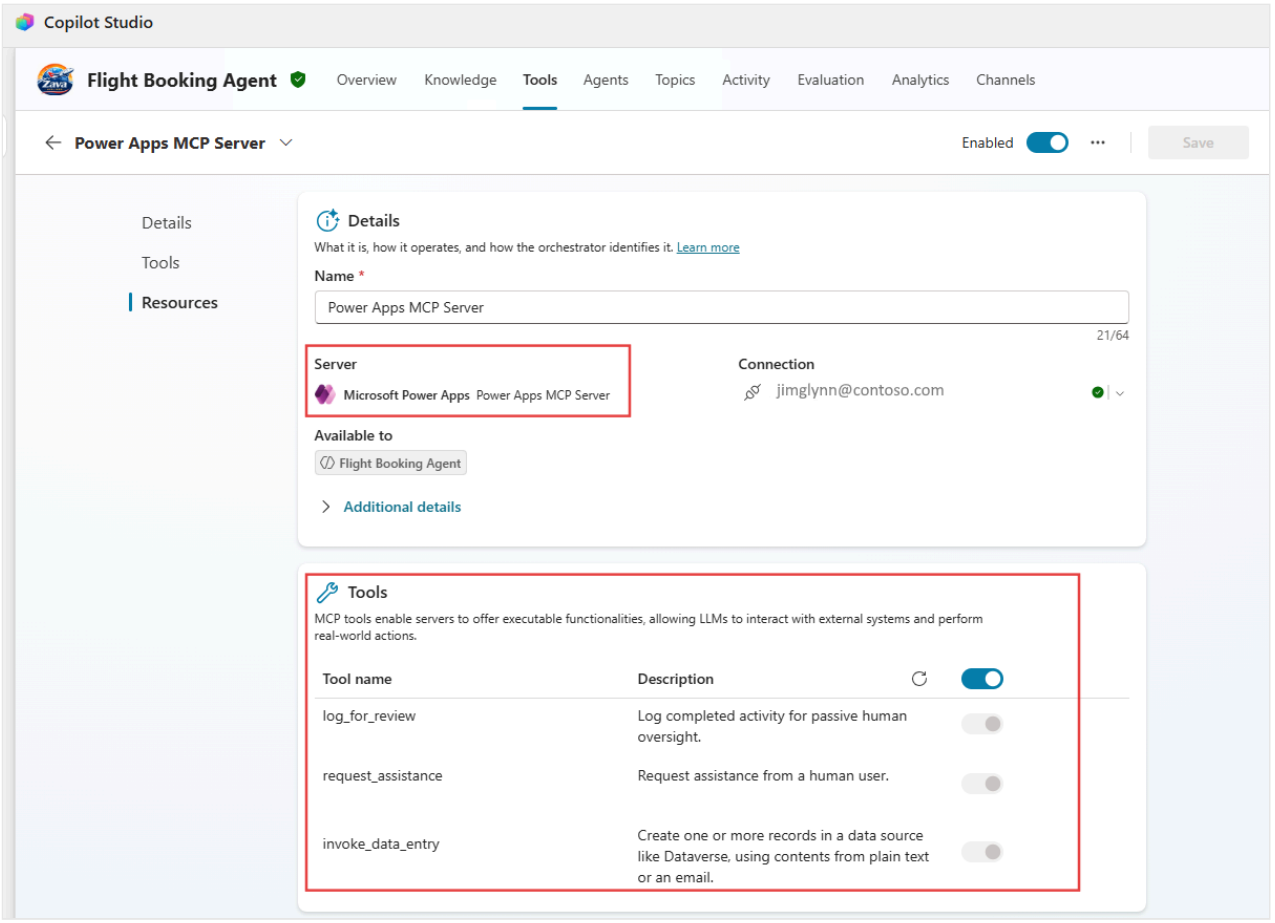
6. Select **Add and configure** to connect the agent to Power Apps MCP tools.



7. In the details section, change the **Credentials to use** field to *Maker-provided credentials* to enable the MCP Server to initialize autonomously via a trigger. Go to Control maker-provided credentials for authentication for more details if this option is disabled.

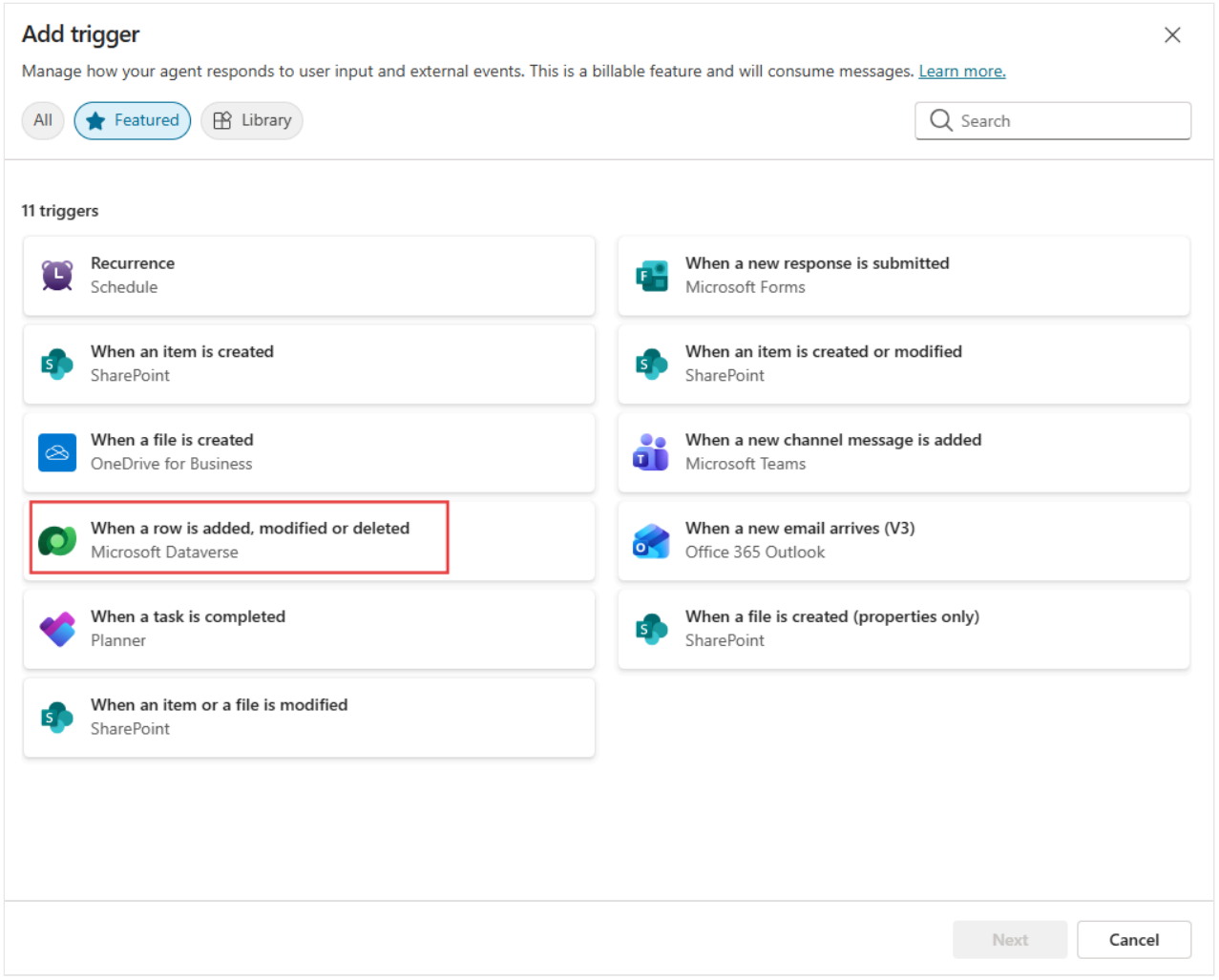


8. Notice the tools available to use with your agent. You can select to enable the tools you want. For information about these tools go to Work with Power Apps MCP server.

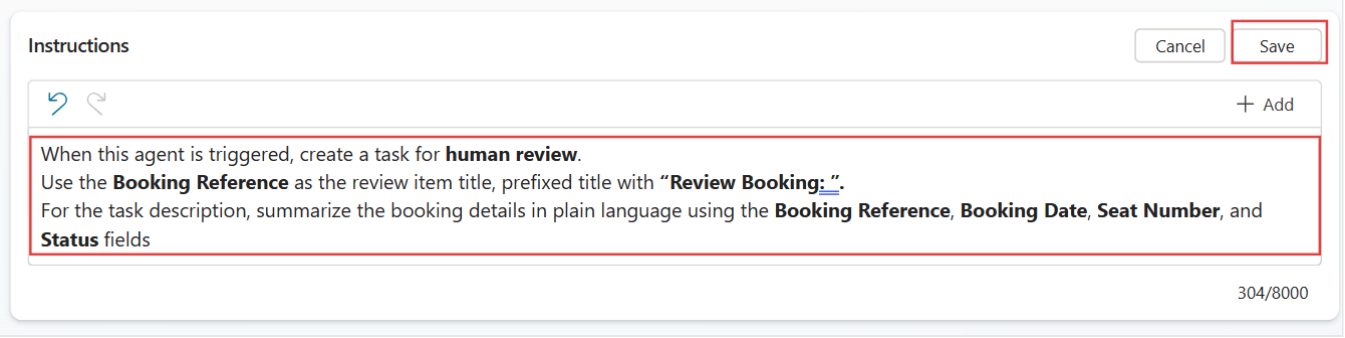


9. Add a trigger for this autonoumous agent so that it is invoked when the trigger is fired. In this example, the trigger **When a row is added, modified, or deleted** is selected for the

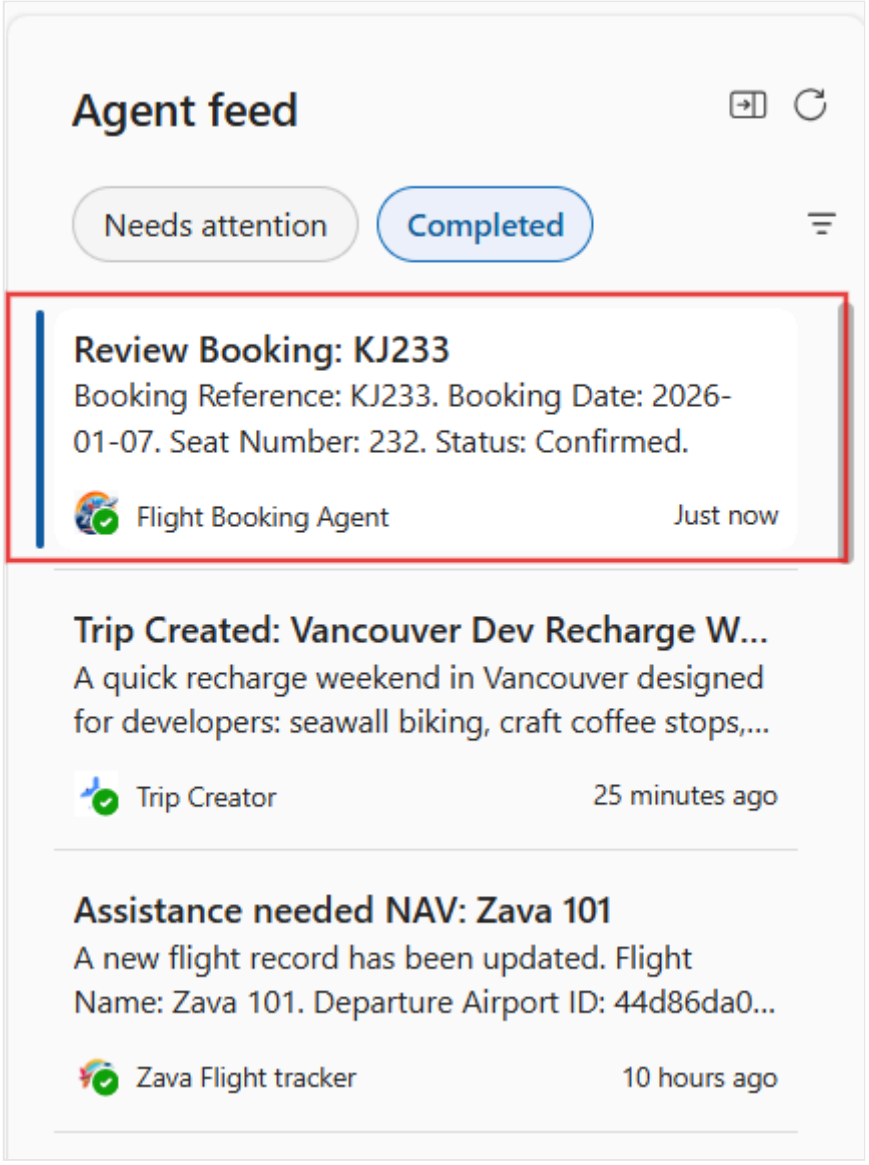
Dataverse **Booking** table.



The agent is ready to use the Power Apps MCP server tools. This image shows how the agent can add a task for human review using natural language instructions. Internally, the task invoked maps to the Power Apps MCP server `log_for_review` tool.



When a new booking record is created, it triggers the agent, which adds the review task in the agent feed **Completed** tab. Note that all tasks created by the `log_for_review` tool are created as "Completed" items intended for passive supervision of agent activity.



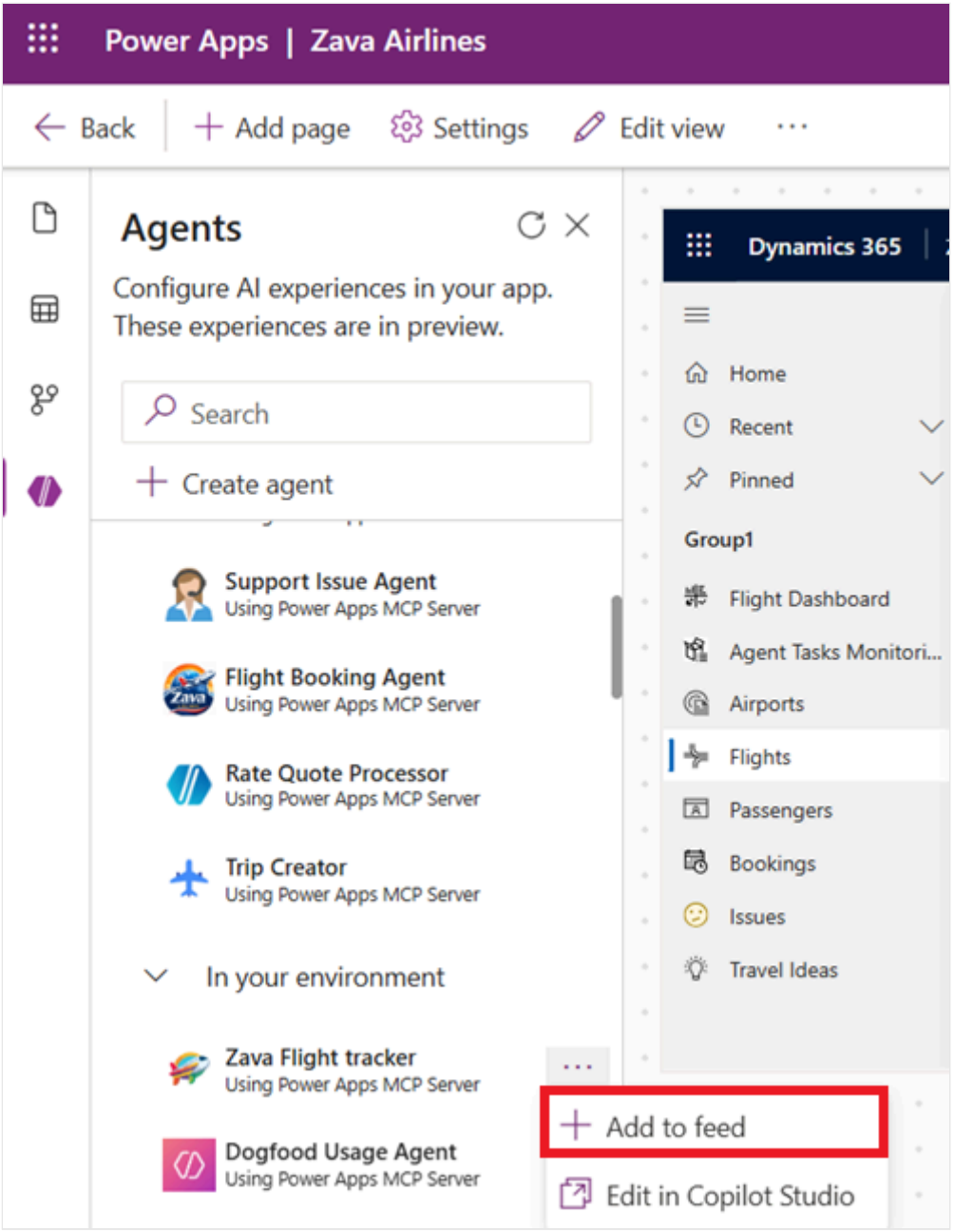
Current limitations

- Only model-driven Power Apps support agent supervision. This feature isn't available for canvas or vibe apps.
- Power Apps MCP server is supported only via Microsoft Copilot Studio.

Add an autonomous agent to an app

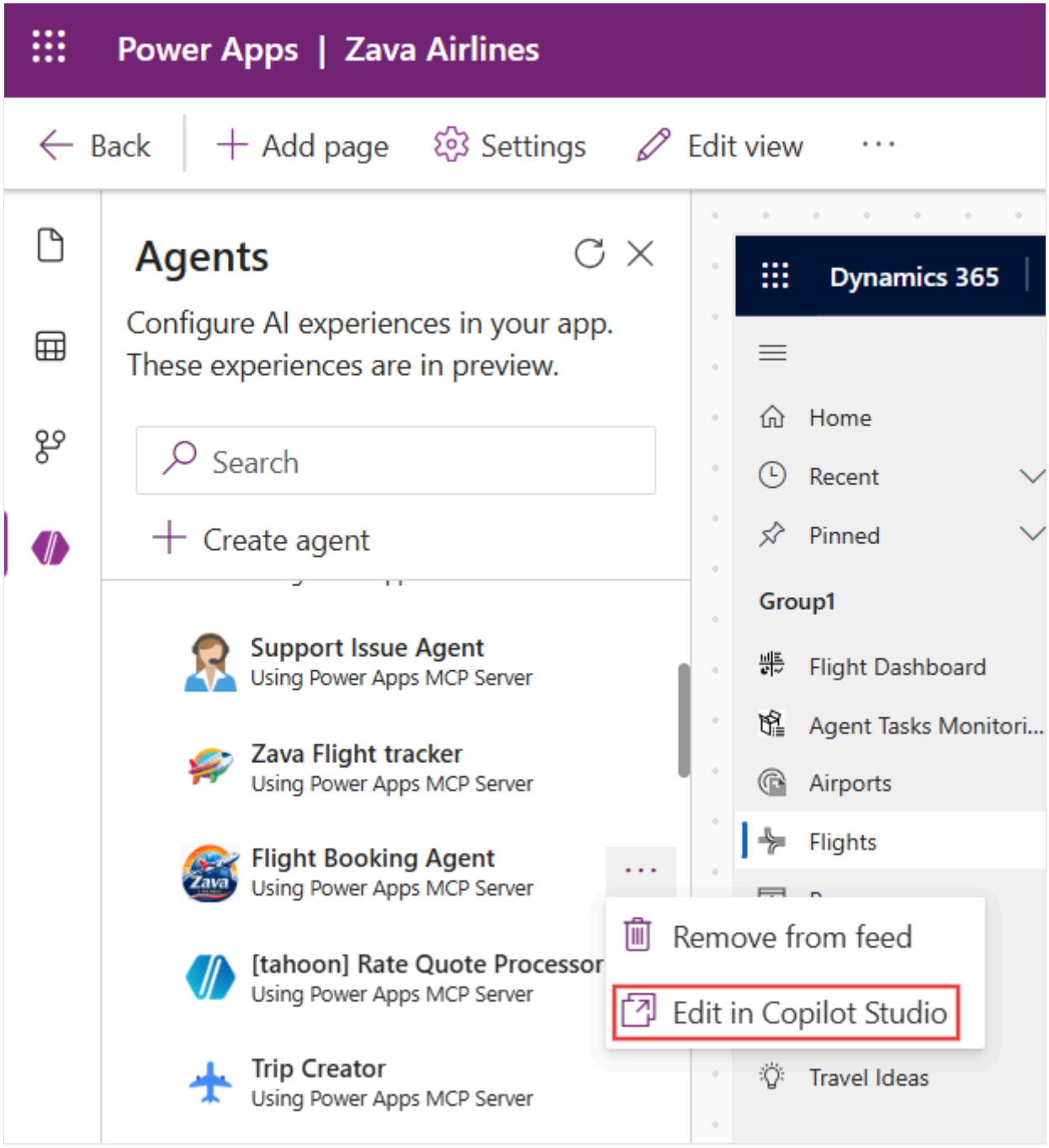
1. Sign in to Power Apps, select **Apps**, and then select **Edit** for the app you want to modify.
2. In the app designer, go to the **Agents** tab.
3. Under the **Agent feed** dropdown, the **In your environment** dropdown appears that lists all agents in your environment. Locate the agent you want to add.

4. Select ... (more options) next to the agent, and then select **Add to feed**.



5. Verify agent feed is added to your app with all added agents by saving, publishing, and playing your app. Previewing agent feed in the app designer isn't currently supported.

6. To view or edit the agent in Copilot Studio, select **Edit in Microsoft Copilot Studio**.



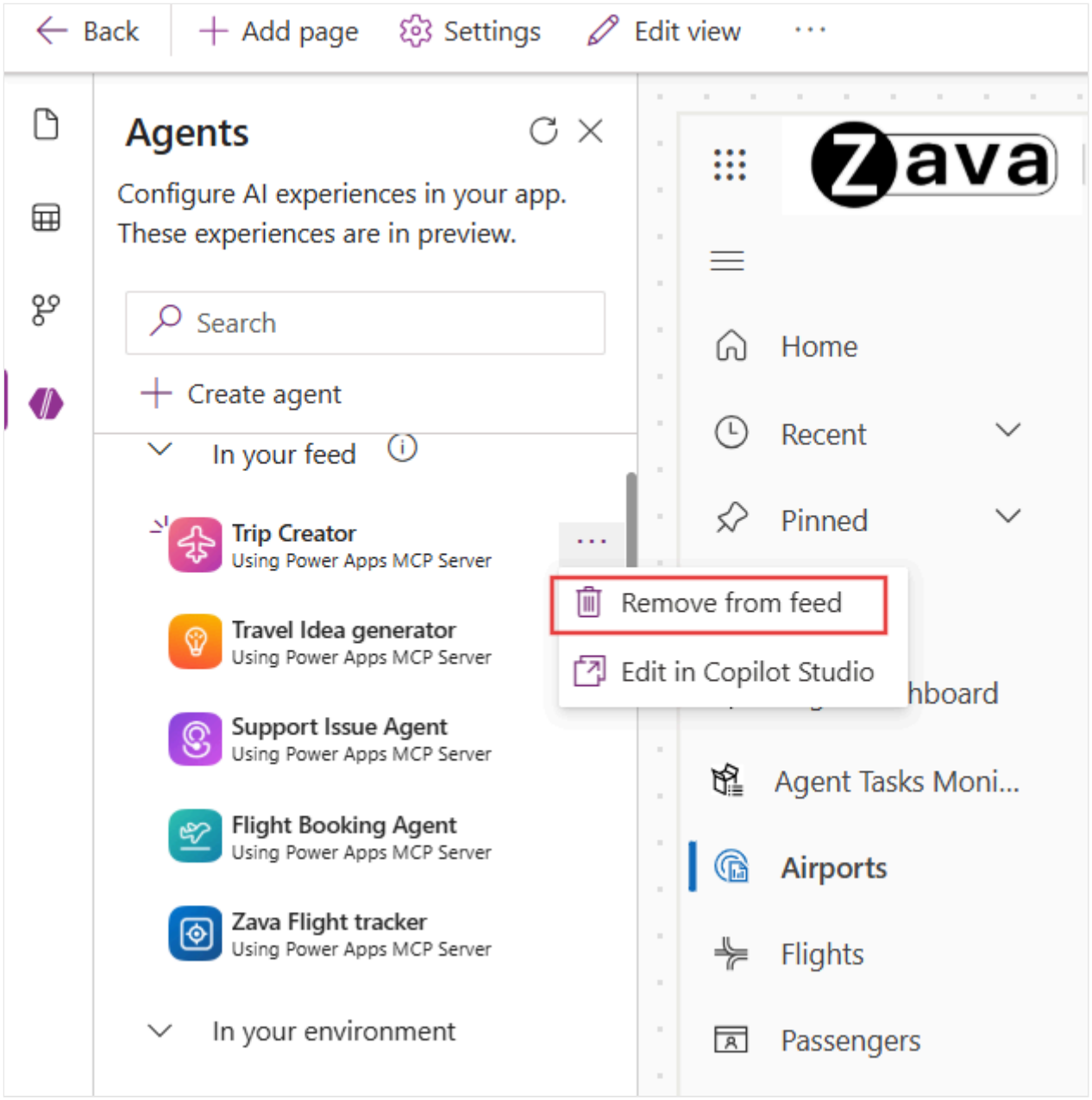
> **Important**
>
> Use the **Create agent** button to open Copilot Studio for agent creation. Note that even if the agent is created through a link inside the app designer, the maker must ensure it meets all requirements to be added to the app.

Remove an autonomous agent from an app

1. In the app designer, go to the **Agents** tab.

2. In the **Agent feed** dropdown, select **In your feed**.

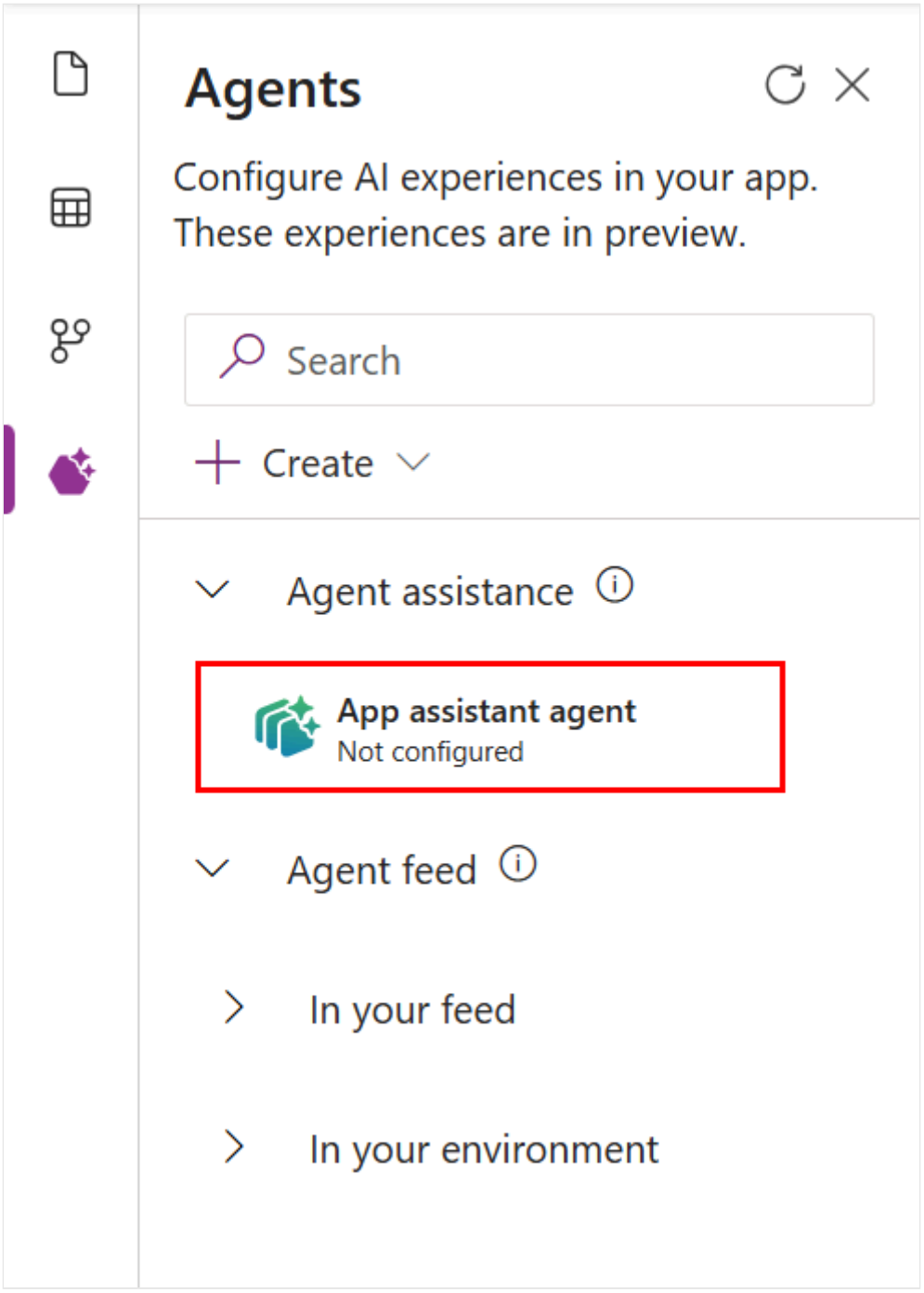
3. Locate the agent you want to remove, select ..., and then select **Remove from feed**.



> **Note**
>
> Removing an agent from an app doesn't remove the agent from the environment.

Working with an app assistant agent

The app assistant agent makes a model-driven app more intelligent and relevant for your organization by adding additional topics, knowledge sources, and more. The Copilot Chat, Agent APIs, and agent response components access the topics within this agent. Learn more: Add app assistant agent



> **Note**
>
> The app assistant agent is a rename of interactive agent, which is the improved experience for **...** > **Configure in Copilot Studio**. The agents created with the previous experiences are now shown as the **App assistant agent**.

Related information

Overview of App Designer

Add app assistant agent

Last updated on 04/06/2026

Work with Power Apps MCP server

> **ⓘ Important**
>
> Starting May 1st, 2026, agent feed only supports agents that use the Power Apps MCP Server to create tasks. Ensure your agents are properly onboarded to the Power Apps MCP server by then to continue using the agent feed. If your agents don't use the Power Apps MCP Server, the agent feed doesn't appear in you model-driven app. More information: Onboard your agent feed to use the Power Apps MCP server

The model context protocol (MCP) is an open protocol that enables seamless integration between large language model (LLM) applications and external data sources and tools. Your agent can use the Power Apps MCP Server to communicate with your Power Apps, providing right human-in-the-loop supervision or agentic workflows.

> **ⓘ Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is available only in the English language and it replaces the earlier Microsoft Copilot Studio activity-based agent feed.
> - For information about how AI is used with this feature, go to FAQ about Power Apps MCP Server invoke_data_entry tool.

The Power Apps MCP Server equips your agent with two types of capabilities:

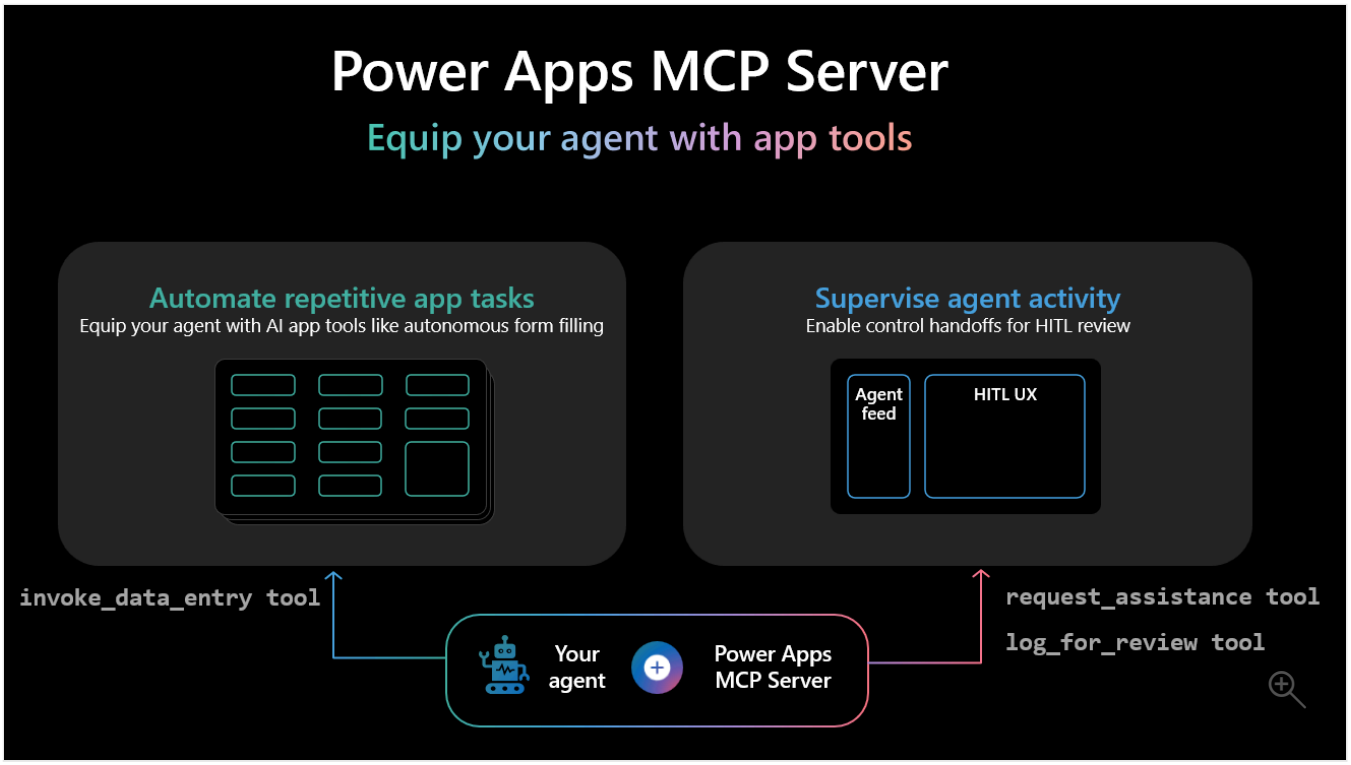
- Automate repetitive app tasks:

  The Power Apps MCP Server enables agents to use advanced app tools developed in Power Apps. For example, the data-entry agent capabilities previously available as an on-demand AI feature are now accessible to any agent through Power Apps MCP server. To use them, you create your agent, configure the MCP tool, and direct it to unstructured

content so it can generate Dataverse records with human review and approval through the enhanced agent feed.

- Supervise agent activity:

  The Power Apps MCP Server also provides specialized tools to business users to supervise any agent activity in the agent feed. Agents can now handoff control to humans for review, assistance, and steering with the MCP tools. These tools provide makers with much more control over the tasks they want to publish to the agent feed and when they need agent-human handoff.



> **Note**
>
> Access to the agent feed and supervision capabilities is limited by default to the System Administrator and System Customizer security roles. To allow additional users to view the agent feed, grant organization-level read/write permissions on the tables listed here. You can create a new security role with these permissions and assign the role to multiple users as needed.
>
> - Agent Hub Goal(agenthubgoal)
> - Agent Hub Insight(agenthubinsight)
> - Agent Hub Metric(agenthubmetric)
> - Agent Task(agenttask)

- Copilot(bot)

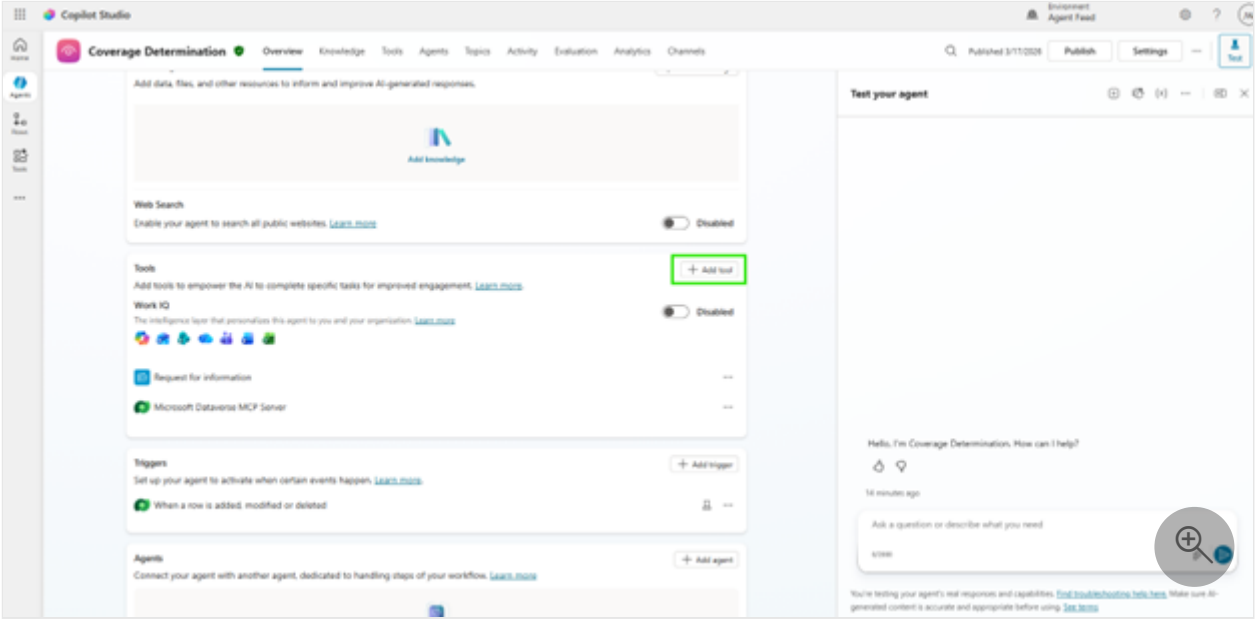
The Power Apps MCP tools improve the more you use them. For example, when you make corrections to suggestions in the agent canvas, the data entry tool improves based on your corrections. To use the enhanced agent feed capabilities, enable and configure the Power Apps MCP server from the Microsoft Copilot Studio agent. Once configured, you can invoke Power Apps MCP server tools from agent instructions using natural language.

More information: Create an autonomous agent connected to Power Apps MCP server

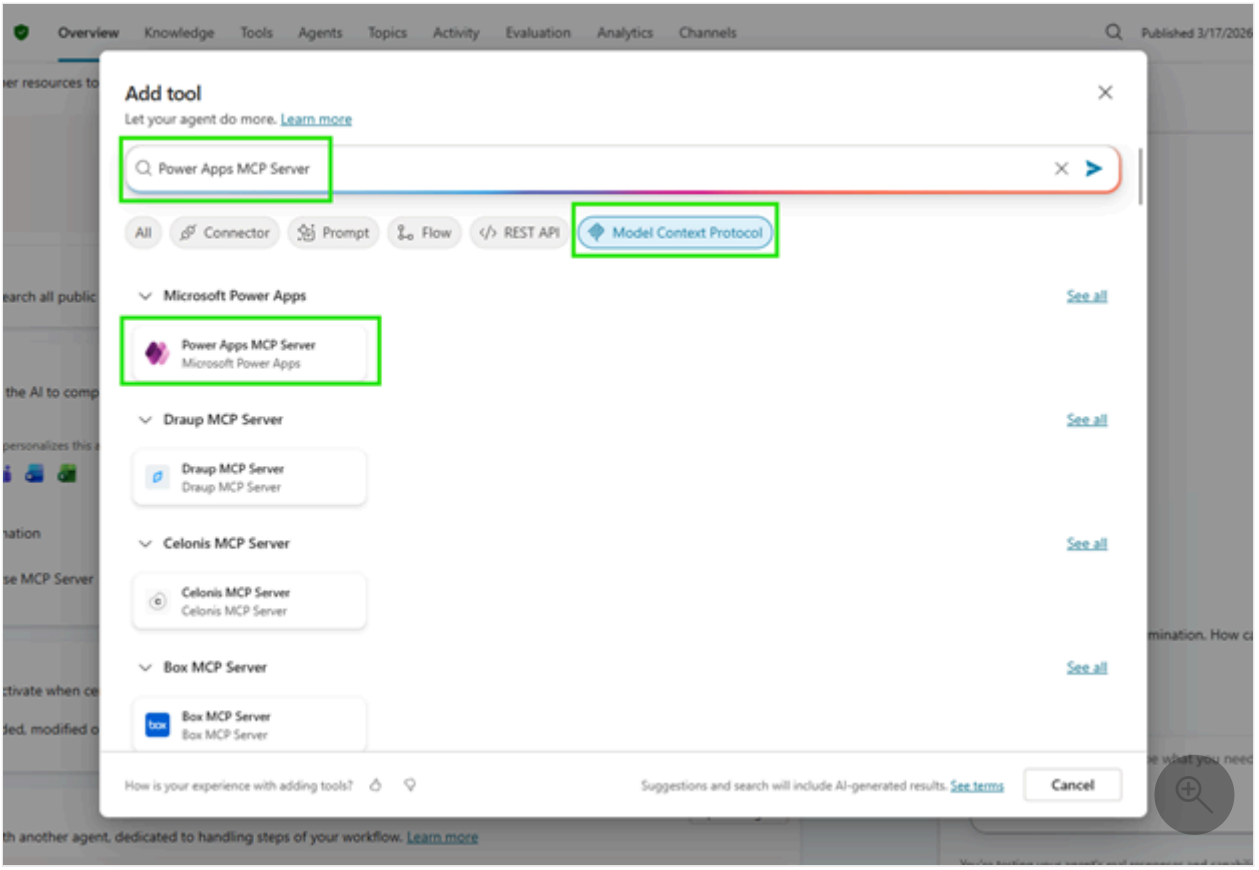
Onboard your agent to use the Power Apps MCP server

To configure an existing agent that was in the previous version of agent feed to use the Power Apps MCP server, you must do the following:

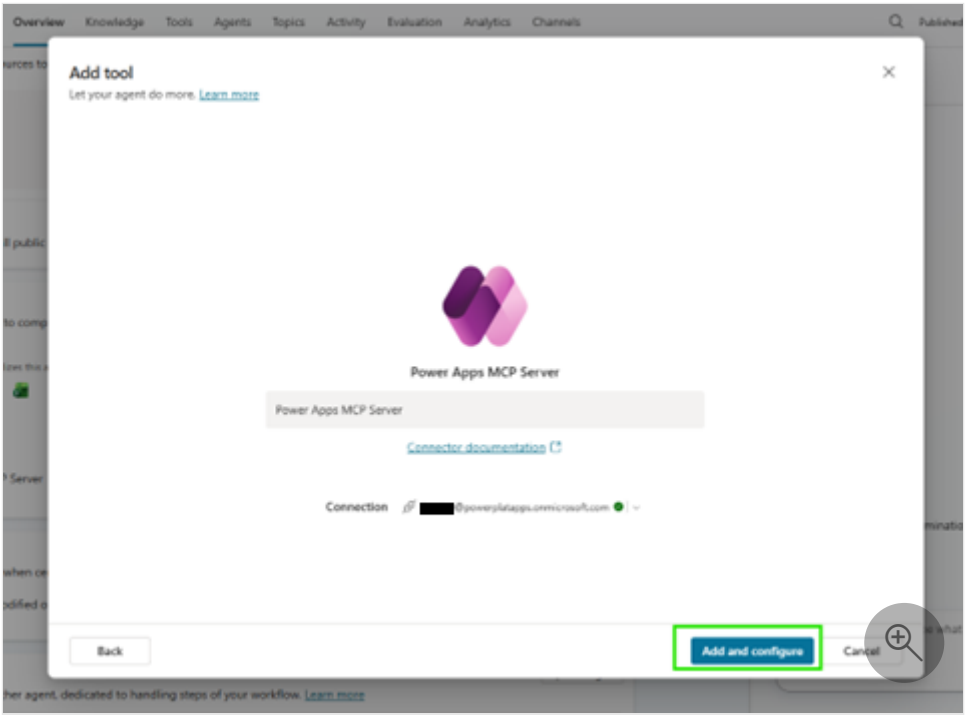
1. Add the Power Apps MCP server to your agent. To do this, open the agent in Copilot Studio and then select **Add tool**.



2. Search for **Power Apps MCP Server**.



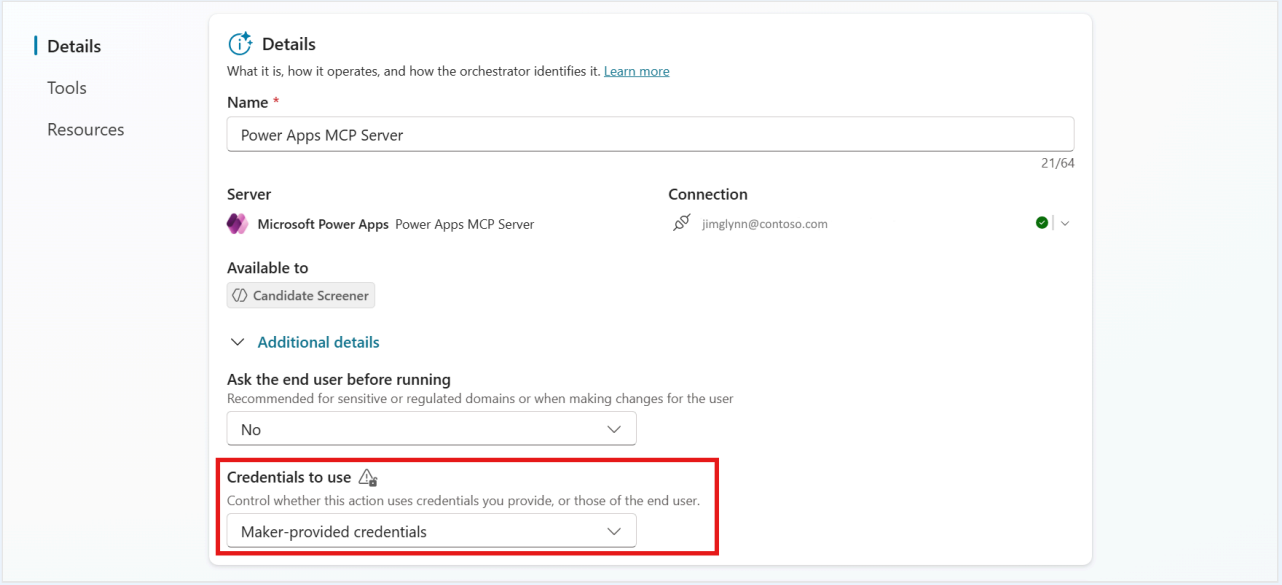
3. Select **Add and configure**.



4. Update your agent instructions to use each of the tools in the Power Apps MCP Server at the proper times in it's orchestration. There are examples of how to do this in the remainder of this document.

5. Save and publish your agent.

> **Important**
>
> For autonomous agent scenarios where an agent runs via a trigger, the Power Apps MCP server must be configured to run using "Maker-provided credentials" as seen in the details section of the tool. Go to Control maker-provided credentials for authentication for more details if this option is disabled.



List of tools

Once connected to the Power Apps MCP Server, the agent can choose from various tools in the Power Platform environment. These tools can generate agent feed items that render different user experiences, such as a side-by-side view for data entry agents or direct navigation to a record for `request_for_assistance` scenarios.

Expand table

Tool	Description
log_for_review	Log completed activity for passive human oversight.
request_assistance	Request assistance from a human user.
invoke_data_entry	Create one or more records in a data source like Microsoft Dataverse, using contents from plain text or an email.

log_for_review

Records completed agent work to the agent feed for review. The `log_for_review` tool is intended for scenarios where an agent has sufficient information to act autonomously but the

user should still be made aware of what the agent has done. This tool can be thought of as a way to passively oversee the high-confidence or low-risk actions performed by agents. It is best suited for decisions that can be easily revised or rolled back if the agent happens to perform the action incorrectly. Besides title, description, and steps you can also ask the tool to add a link to either the relevant Dataverse record or to an app-external URL. If an agent action touches multiple Dataverse records, you can instruct the agent which record it should navigate to in connection with the created task. It could be the link to the record the agent created using the Dataverse MCP server or a record link present in context like the record that triggered the agent execution. These tasks are shown in the agent feed's **Completed** tab.

Sample instruction

When the customer makes a booking from the portal this agent must log the details for review. The review item title should be based on the booking reference number and must use the exact prefix “Review Web Booking: ”. In the review description, write a concise summary of the booking that includes main fields like Booking Reference, Booking Date, Seat Number, and Status, so a reviewer can quickly understand what was processed without opening the record. Ensure the description reads as a short paragraph and accurately reflects the current values from the booking record. Include your reasoning as steps. Also, include a link to the booking record.

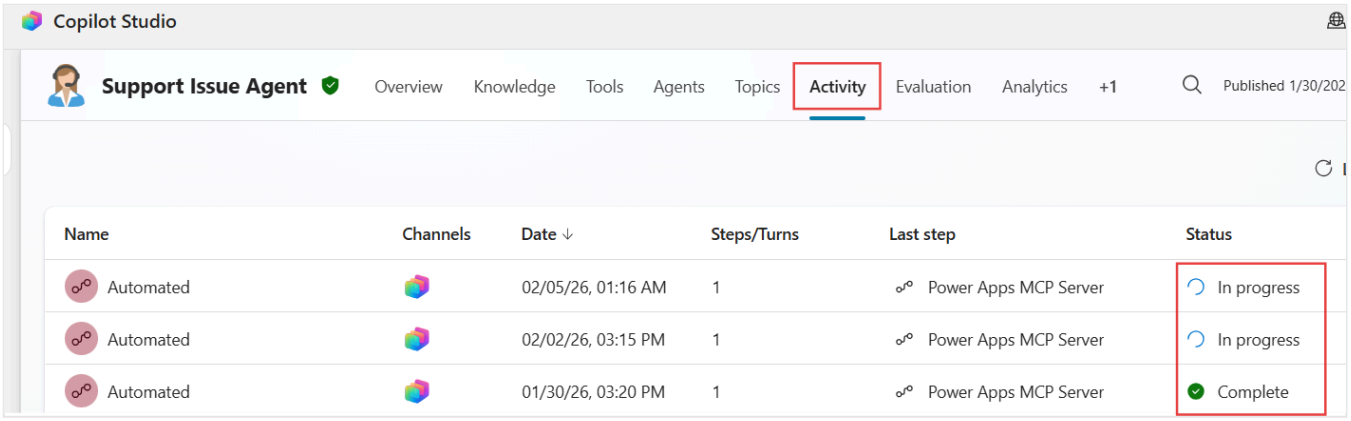


request_assistance

The intended purpose of the `request_assistance` tool is to enable agents to surface errors, escalations, or exceptions to users, so they can take appropriate action. As a maker, you can define the scenarios for when to have your agent use the `request_assistance` tool. It creates an agent feed task that is populated in the **Needs Attention** section of the agent feed. This is an asynchronous operation that calls the Microsoft Copilot Studio agent that waits until the

human completes the action. For details on completing the action feed activity, go to Supervise agents in model-driven apps with agent feed (preview)

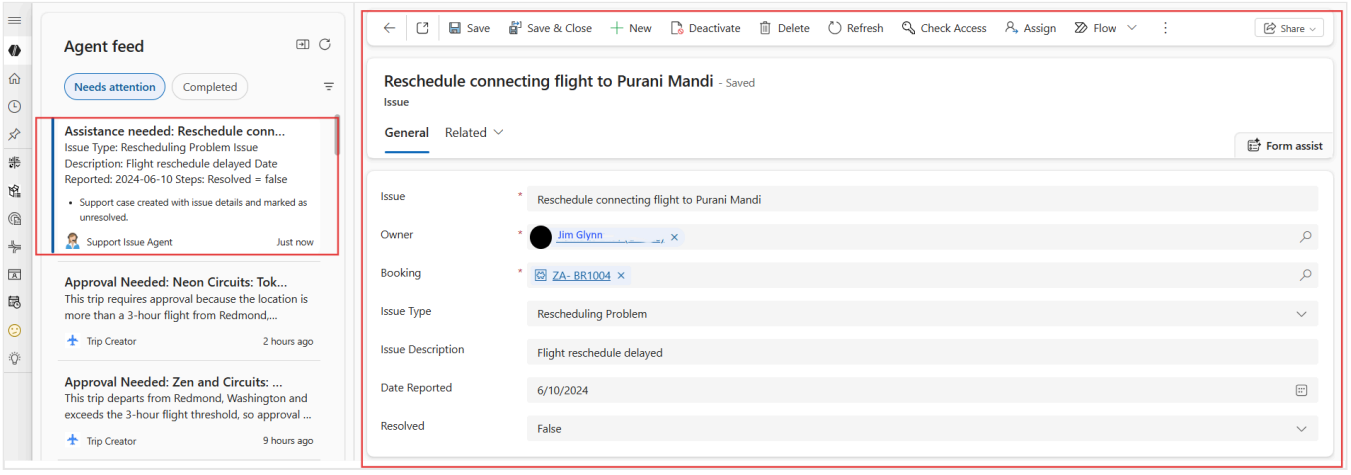
You can observe the **In progress** status for the agent run in the activity tab when viewing the agent in Copilot Studio. Once the user completes the activity from agent feed, control comes back to agent via callback and agent can complete the task.



Like with the `log_for_review` tool, you can control the task output for title, description, and steps and can be specific when telling the agent which link to associate with a given task.

Sample instruction

When this agent is triggered by the creation of a new support case, it should request assistance. In the request, set the title by prefixing the value of issue with “Assistance needed: ”. In the task description include the issue type, issue description, date reported, and the resolved value. Include your reasoning steps. Also include a link to the related Dataverse issue record. Once the user completes the task, continue processing by setting the case status to Closed.



Design your user-in-the-loop

Before writing your agent's instructions, decide where human oversight belongs in your workflow. Use the following questions to identify which moments should use `request_assistance`, which should use `log_for_review`, and which the agent can handle autonomously.

Expand table

Question	Guidance	Tool
Where are the stakes high?	High-stakes outcomes require oversight regardless of agent confidence. Give the agent explicit instructions to pause.	`request_assistance`
When is user intervention always needed?	If you can state it as a rule, encode it directly in the agent's instructions.	`request_assistance`
Which inputs vary unpredictably?	Unstructured data, edge cases, and novel situations can't always be anticipated. Instruct the agent to surface these dynamically.	`request_assistance`
Does the agent need an answer to keep going?	If the agent is blocked without human input, it should wait for a response. If it can proceed and a human audits later, it shouldn't.	`request_assistance` if yes, `log_for_review` if no
Does a user own the outcome?	Compliance requirements, high-value approvals, or policy decisions may require a human sign-off even when the agent is confident.	`log_for_review`

> **Tip**
>
> A well-designed agent doesn't ask for help constantly. Instead, it asks at the right moments. Use `request_assistance` sparingly for genuine decision points, and let `log_for_review` handle the rest.

Example instructions by pattern

Explicit rule:

> *"For any claim with an estimated loss amount over $5,000, use `request_assistance` to route the claim to the assigned adjuster before proceeding."*

Dynamic judgment:

> *"If the cause of loss is ambiguous or claim documents conflict with each other, use `request_assistance` to flag the claim for adjuster review."*

Passive oversight:

> *"After completing the coverage determination, use `log_for_review` to record the outcome and confirm the claim has been cleared to advance."*

Example: Homeowner's insurance coverage determination agent

The following example shows how these patterns apply to a complete, real-world workflow.

The agent triggers automatically when a new claim is submitted. It pulls the relevant policy, endorsements, and supporting documents from Dataverse, then reasons over them to produce a coverage determination, checking whether the policy was active, whether the claimed peril is covered, and whether any document conflicts affect confidence in the outcome.

From there, the agent uses the Power Apps MCP server to surface results in the agent feed based on what it found. If the claim is ambiguous, conflicted, or requires adjuster judgment, the agent uses `request_assistance` to create a task for the assigned adjuster with the context they need to act. If the claim is clear-cut, the agent uses `log_for_review` to record the outcome passively and no action is required. When an adjuster completes a task, the agent resumes, reads the decision, updates the claim record, and logs a completion notice back to the feed.

The result is a workflow where the agent handles the routine volume autonomously and pulls a human in only at genuine decision points with enough context that the adjuster can act immediately.

invoke_data_entry

The `invoke_data_entry` tool streamlines the creation of Dataverse records by extracting structured information from unstructured inputs such as emails, messages, or documents. When invoked from a Copilot Studio agent, it automatically analyzes incoming content, fills out the appropriate form with the extracted data, and presents the proposed entry as a task in the agent feed for user review and approval. It requires review of the proposed entry by a user

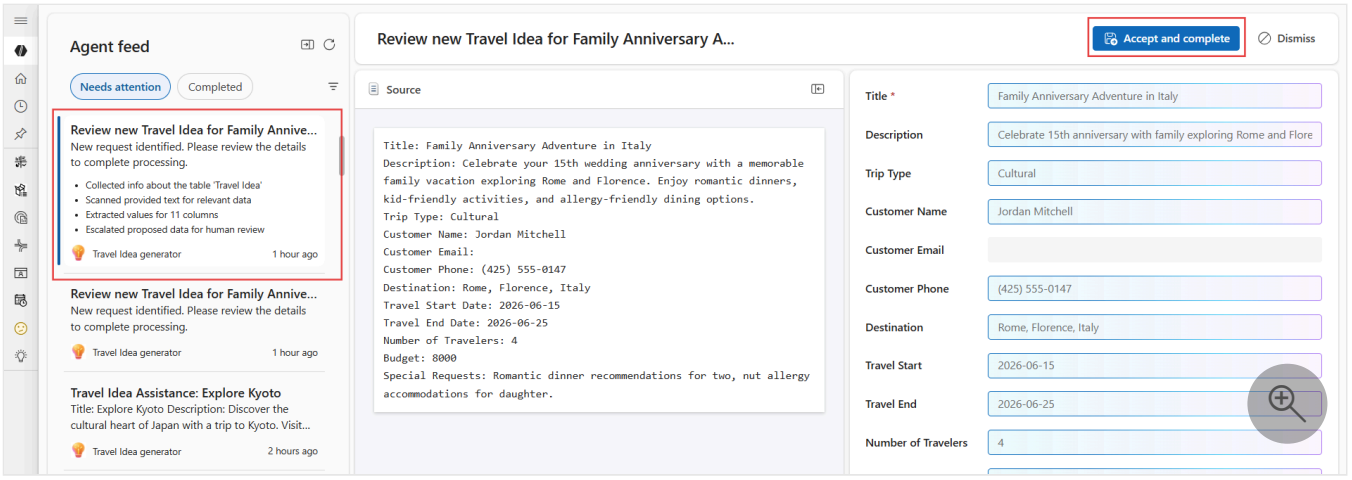
before creating the record. **Records are never created automatically using the `invoke_data_entry` tool.** This enables fast, reliable data capture with minimal manual effort.

Sample instruction - shared email triggered agent

You are the travel idea generator agent. Your job is to process incoming emails and create travel idea records in Dataverse.

When an email arrives:

1. Determine if it contains travel-related information (either in the email body or attachments).
2. Use the `invoke_data_entry` tool to create a travel idea record with the extracted information in the following columns:
    - cr3ea_title
    - cr3ea_description
    - cr3ea_triptype
    - cr3ea_customername
    - cr3ea_customeremail
    - cr3ea_customerphone
    - cr3ea_destinationcity
    - cr3ea_travelstart
    - cr3ea_travelend
    - cr3ea_numberoftravelers
    - cr3ea_budgetusd
    - cr3ea_specialrequests
3. If information is missing, still create the record with available data - leave unknown fields empty.



> **Note**
>
> - When you write instructions for your agent, always reference Dataverse columns by their logical names as shown in the sample instruction. Clear, direct instructions help the agent reliably create records from the input. You can view a column's logical name by opening the table in make.powerapps.com, select **Columns**, and then open the column to view the details.
> - `invoke_data_entry` tool supports .pdf, .xlsx, .docx, .jpeg, .jpg, .png, .gif and.bmp formats.
> - `invoke_data_entry` tool can populate single line of text (None format), Whole number and Decimal column types.
> - Ensure that the user has permission to create records for the target table.

How the invoke_data_entry tool works

When you configure a Copilot Studio agent to use the Power Apps MCP Server and enable the `invoke_data_entry` tool, the agent follows this process:

1. An agent trigger is fired based on your configuration — such as an email arriving in a monitored mailbox or new document uploaded to SharePoint.
2. The agent analyzes incoming content and your instructions to determine whether the `invoke_data_entry` tool should be used.
3. If needed, the `invoke_data_entry` tool is invoked, passing the input content and the target Dataverse table and table columns to predict.
4. The tool processes the input, extracts relevant information, and populates a Dataverse form with suggested values for each mapped column.
5. A task appears in the agent feed. Selecting it opens the data-entry review experience. The left panel shows the original input, and the right panel displays the form populated with

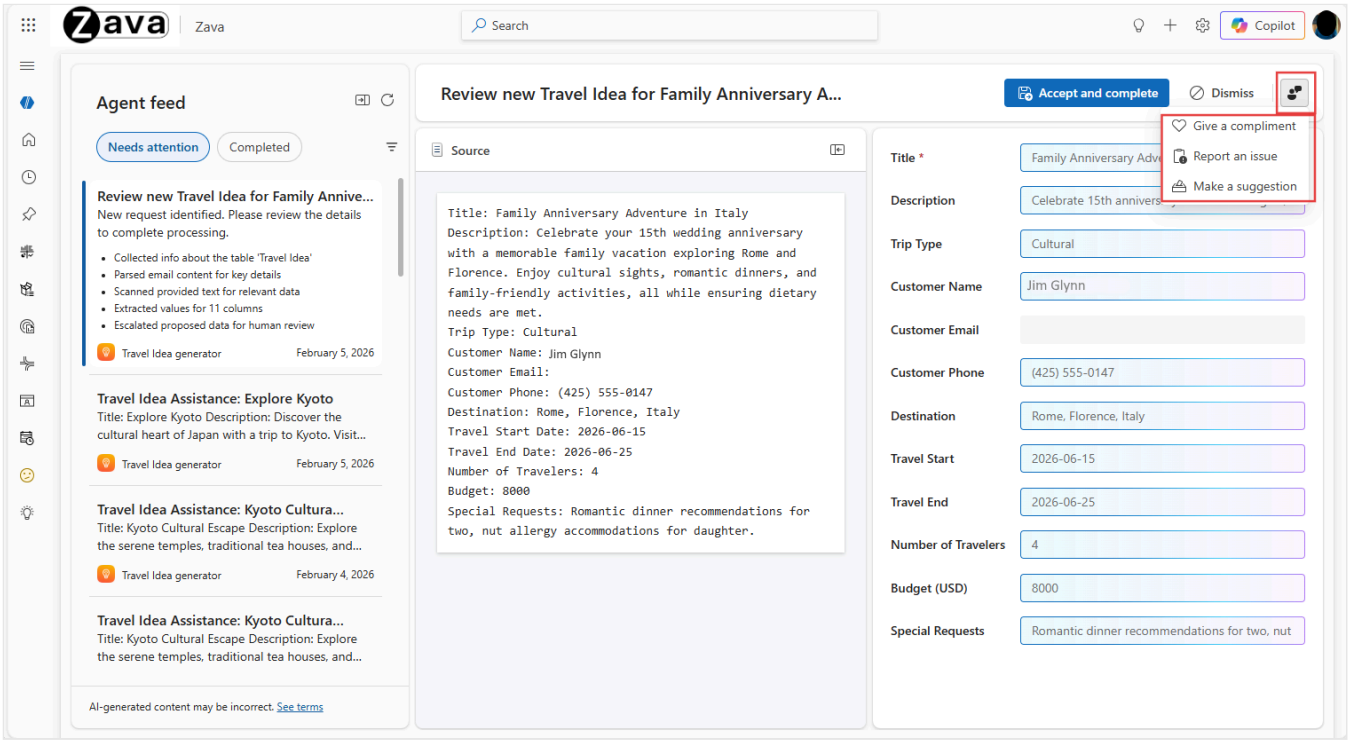
suggested values.

6. The user can review the extracted values, make corrections if needed, and then save the record to Dataverse.

Provide feedback

To provide feedback about the invoke_data_entry tool:

1. Open a invoke_data_entry task in the agent feed.
2. Select the feedback button in the task header.
3. Choose to give a compliment, report an issue, or make a suggestion.



Related articles

Add agents to your model-driven app (preview)

Supervise agents in model-driven apps with agent feed (preview)

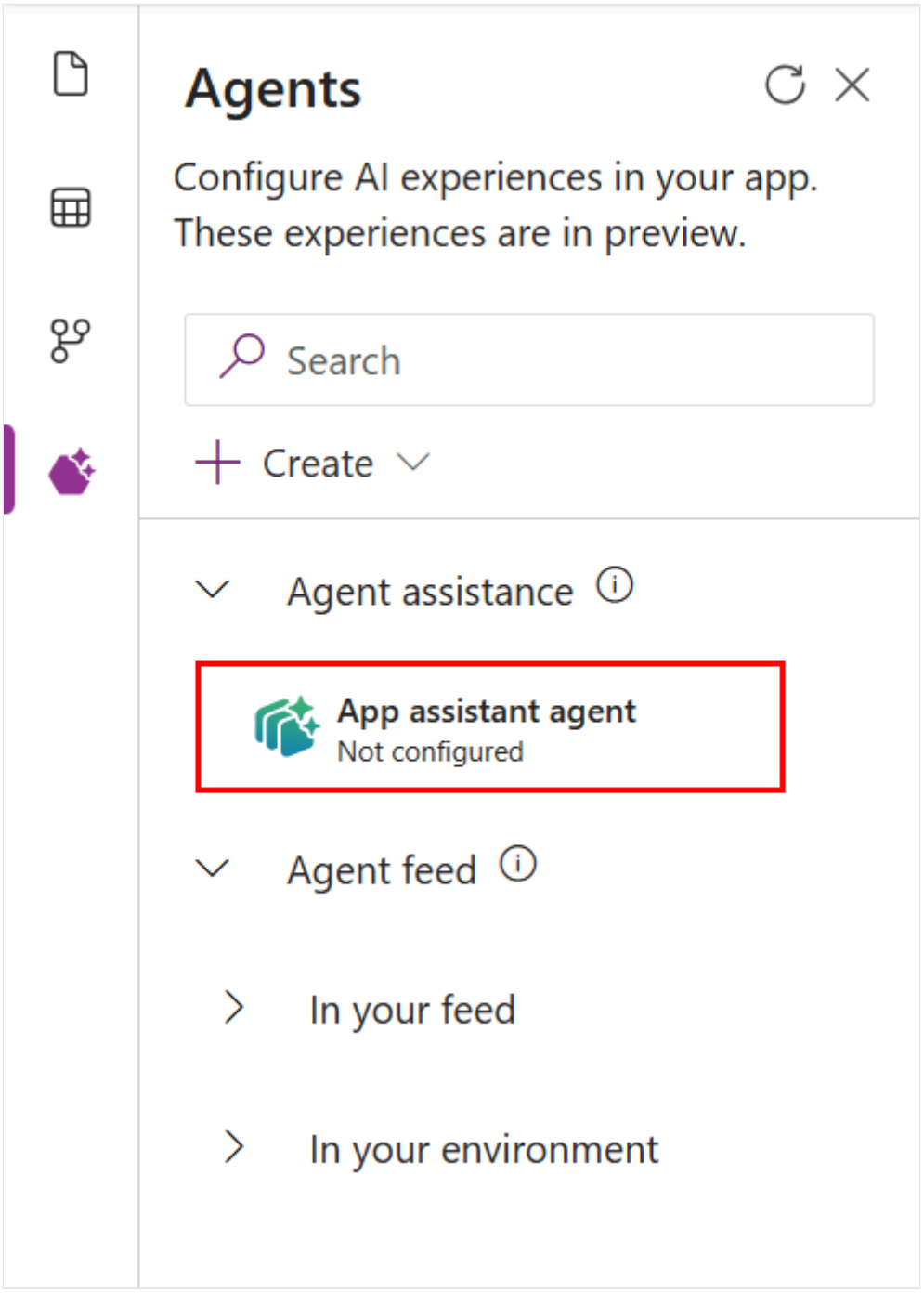
Last updated on 04/07/2026

Add an app assistant agent to your model-driven app (preview)

09/04/2025

[This topic is pre-release documentation and is subject to change.]

The app assistant agent makes a model-driven app more intelligent and relevant for your organization by adding additional topics, knowledge sources, and more. The Copilot Chat, Agent APIs, and agent response components access the topics within this agent.



Important

- This is a preview feature.
- Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

> **Note**
>
> The app assistant agent is a rename of interactive agent, which is the improved experience for **...** > **Configure in Copilot Studio**. The agents created with the previous experiences are now shown as the **App assistant agent**.

Create an app assistant agent

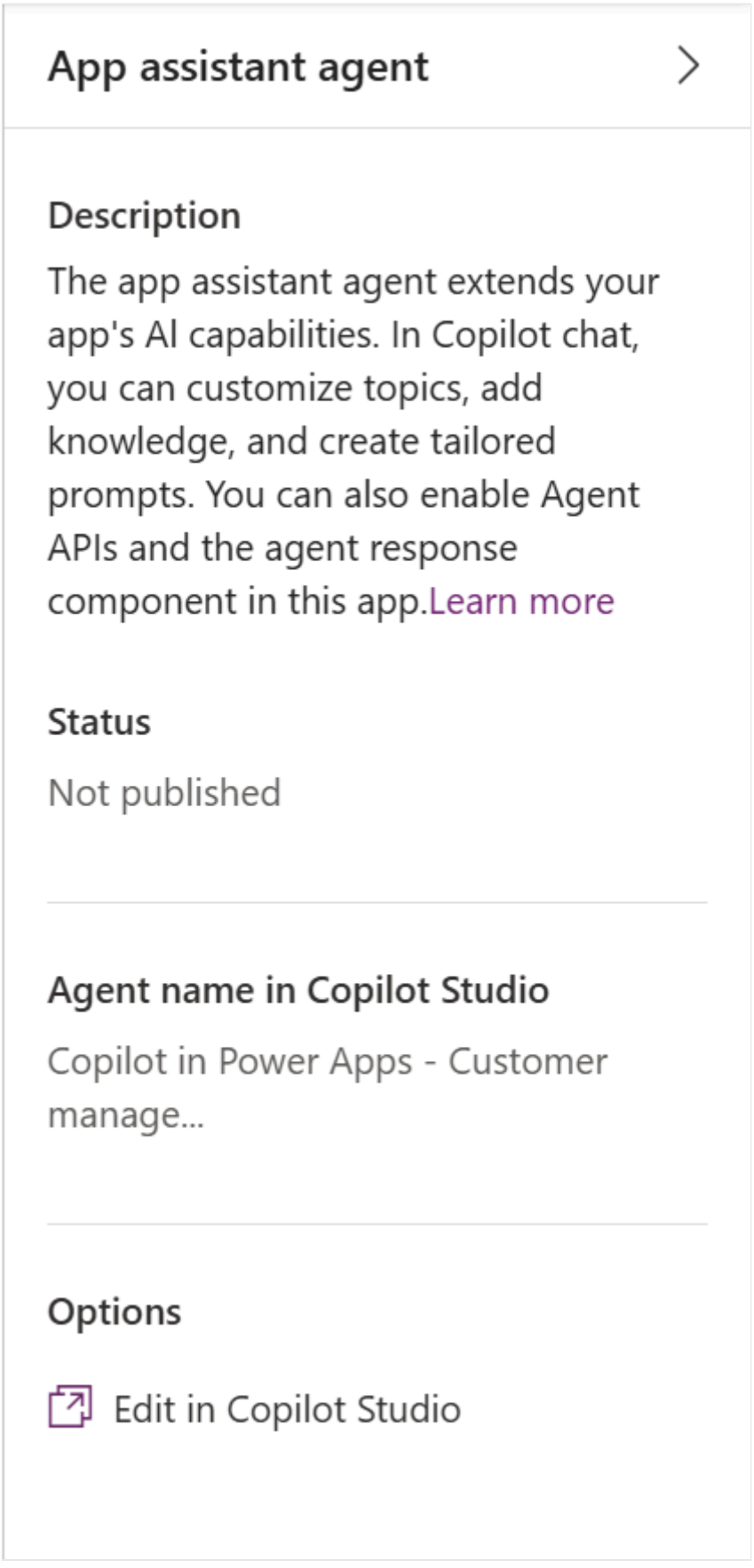
When an app assistant agent is created, it's named as **Copilot in Power Apps - <app name>**. The created agent is associated with the app when the app is saved and published.

1. In the app designer, select the **Agents** pane.
2. Expand **Agent assistance**.
3. On **App assistant agent**, select **...** > **Configure**.

   A new browser tab for **Copilot Studio** is opened with the new agent.

   > **Note**
   >
   > Pop-ups need to be enabled in your web browser to allow editing the created agent.

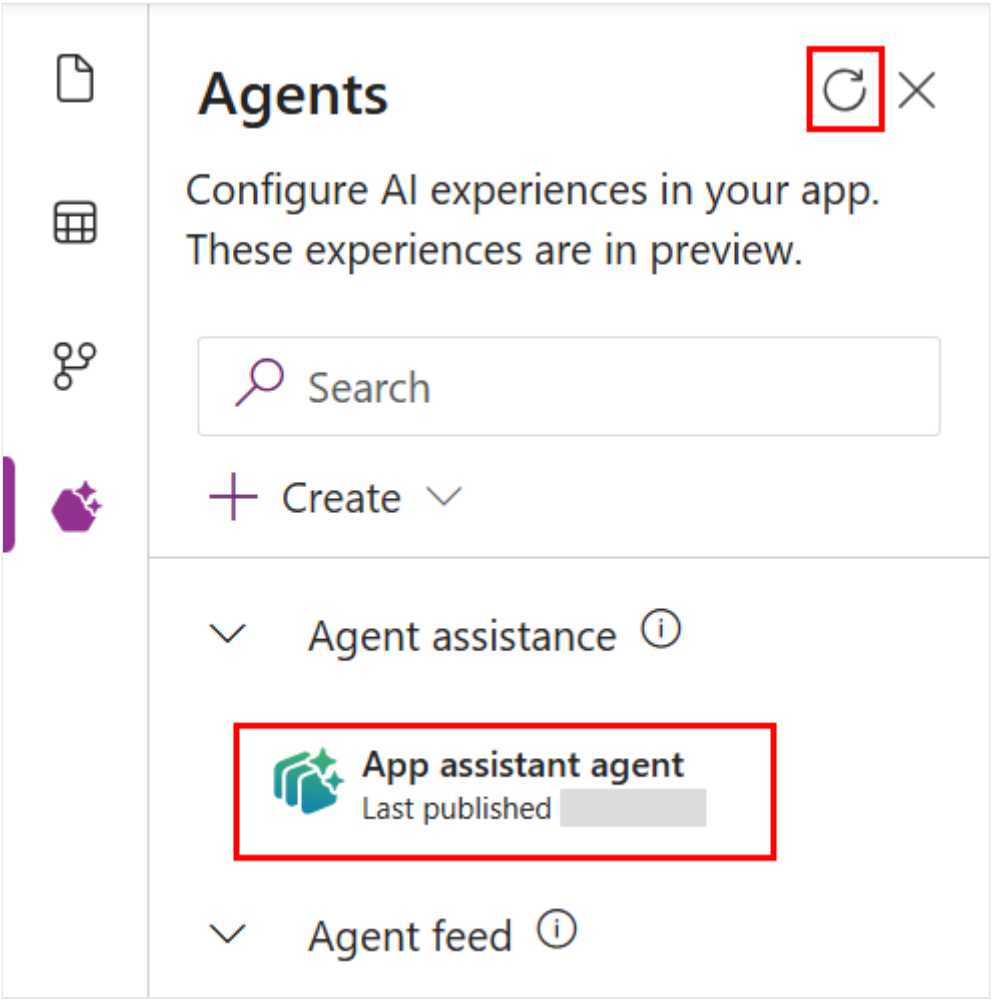
4. After configure completes, the right pane is expanded showing the status of **Not published** and the agent name.



5. Switch to the **Copilot Studio** browser tab or select **Edit in Copilot Studio** to add topics, knowledge, and so on, to the agent.

6. Save and publish the agent.

7. Switch to the app designer browser tab and select refresh in **Agents** pane header to show agent published status.



8. In the app designer, save and publish the app.

Edit an app assistant agent

After the app assistant agent has been initially configured, it can be edited from either the **Agents** pane or the **App assistant agent property** pane.

Agent pane:

1. In the app designer, select the **Agents** pane.
2. Expand **Agent assistance**.
3. On **App assistant agent**, select **...** > **Edit in Copilot Studio**.

App assistant agent property pane:

1. In the app designer, select the **Agents** pane.
2. Expand **Agent assistance** and select **App assistant agent**.
3. In the property pane for App assistant agent, select **Edit in Copilot Studio**.

Limitations

The app assistant agent isn't able to be defined or referenced in app designer for these Microsoft model-driven apps:

- Connected Field Service
- Customer Insights
- Customer Service Hub
- Customer Service workspace
- Dynamics 365 App for Outlook
- Field Service
- Field Service Mobile
- Inventory Visibility
- Omnichannel engagement hub
- Project Operations
- Sales Hub
- Sustainability

> **Note**
>
> When a model-driven app includes both Lead and Opportunity tables, it implicitly uses the **Copilot in Dynamics 365 Sales** agent. The app designer doesn't show the app assistant agent, but the **Copilot in Dynamics 365 Sales** agent can be customized from Copilot Studio.

Related information

Overview of App Designer

Add agents to your model-driven app

Customize Copilot Chat

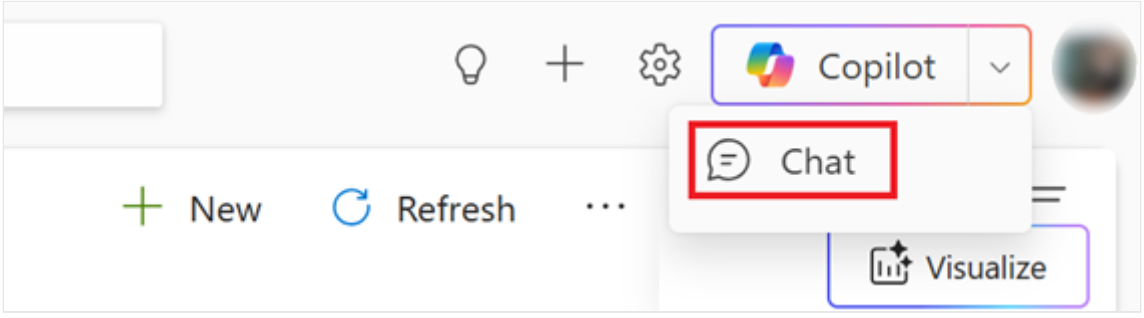
Add Microsoft 365 Copilot for app users in model-driven apps

Microsoft 365 Copilot makes it easier for users to work with model-driven apps by offering AI-powered insights through natural language conversations. With this feature, users can quickly find information, navigate apps more easily, and get help to boost their productivity. App makers can enable Microsoft 365 Copilot to give users access to conversational AI that understands app data and provides helpful, contextual answers. Users can ask questions about their Microsoft Dataverse table data in plain language and receive immediate, relevant responses.

https://learn-video.azurefd.net/vod/player?id=843f065d-4ee4-4006-b09b-e97da24b7393&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fmodel-driven-apps%2Fadd-microsoft-365-copilot

This article shows you how to enable and configure Microsoft 365 Copilot for your model-driven apps, both at the environment level and for individual apps.

When enabled, users can open Microsoft 365 Copilot in their model-driven app by selecting **Copilot** > **Chat** on the upper-right corner. For more information, see Use Microsoft 365 Copilot in model-driven apps.



Prerequisites

To use Microsoft 365 Copilot in canvas apps, make sure the following prerequisites are met:

- Your tenant must be set to allow **Dataverse data available in Microsoft 365 Copilot**. For more information, see Enable Microsoft 365 admin center Copilot Dataverse settings.
- Dataverse Search must be set to **Default** or **On** for the environment. Fore more information, see Understand what this means for generative AI experiences.

- To use the Microsoft 365 Copilot feature in your app, users must have both a Power Apps premium license and a Microsoft 365 Copilot license.

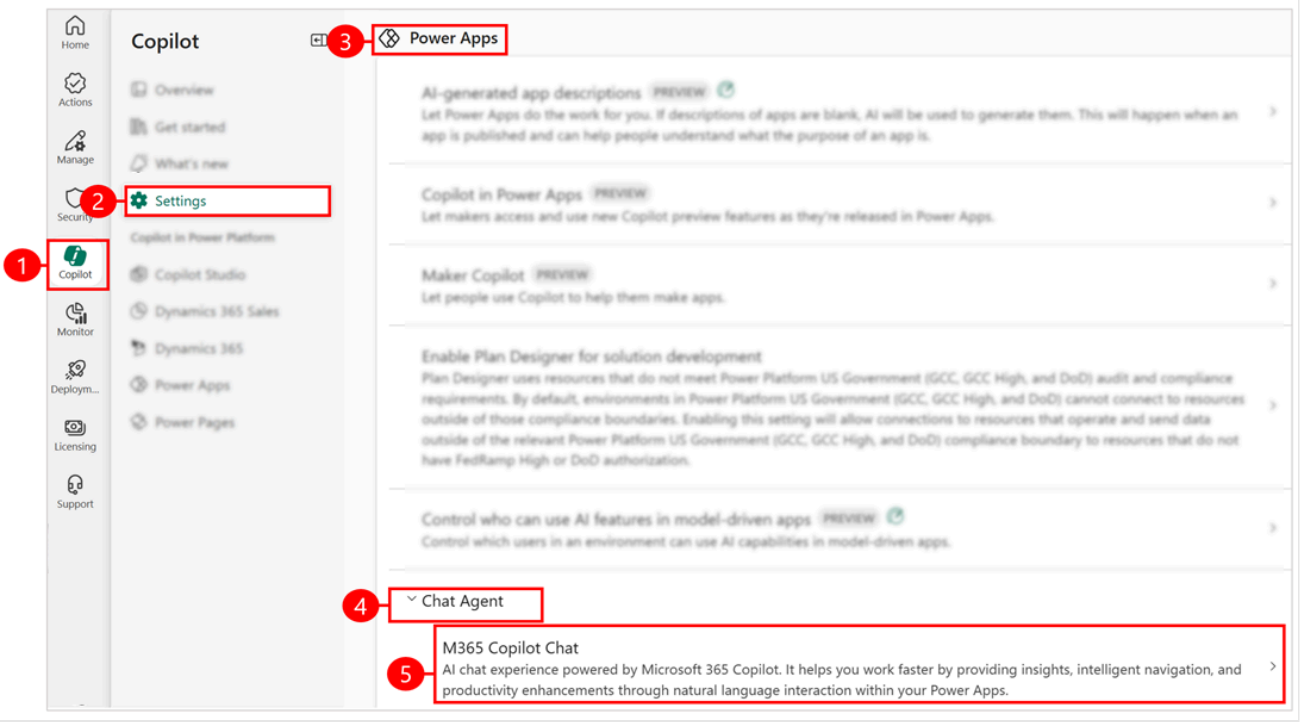
> **Note**
>
> - Microsoft 365 Copilot for model-driven apps relies on Dataverse Search indexes. Enabling this feature might increase your environment's Dataverse capacity consumption. For more information, see What actions can makers or admins take to manage Dataverse Search efficiently.
> - Sufficient Dataverse database capacity must be available to support indexing, semantic search, and Copilot operations. For more information, see Capacity page details.

Enable Microsoft 365 Copilot for your environment

To manage Microsoft 365 Copilot for model-driven apps, start by learning how to manage Microsoft 365 Copilot.

Power Platform administrators can set up and configure the Microsoft 365 Copilot feature for users in their environment.

1. Sign in to the Power Platform admin center.
2. In the navigation pane, go to **Copilot** > **Settings**. Under **Power Apps**, expand **Chat Agent** and choose **M365 Copilot**.

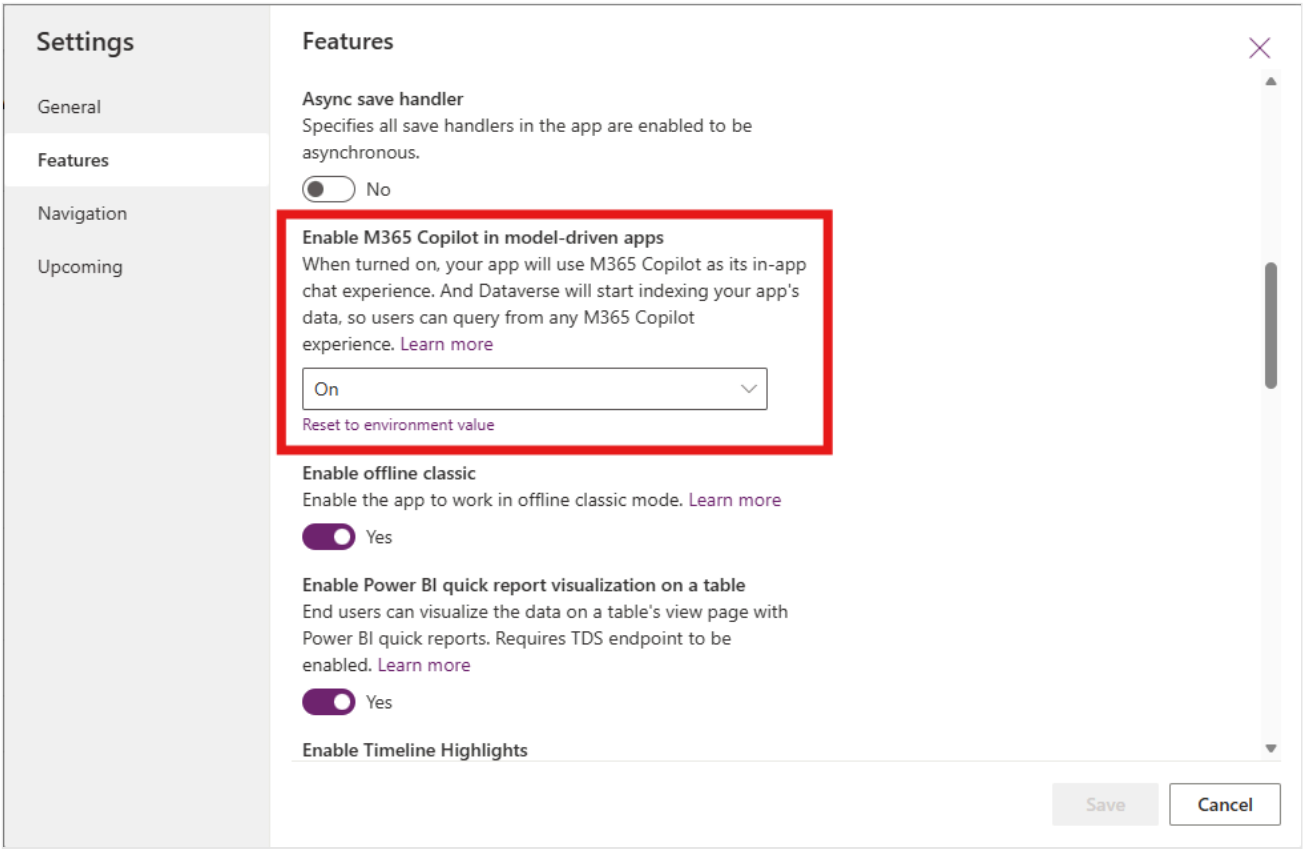


3. Select an environment group or an environment name, then select **Edit Setting**.
4. Select **On** and then select **Save** to enable Microsoft 365 Copilot for your canvas apps in the selected environment group or environment.

Enable Microsoft 365 Copilot in a model-driven app

Makers can enable or disable Microsoft 365 Copilot for a specific model-driven app.

1. Sign in to Power Apps.
2. Open a model-driven app for editing.
3. Select **Settings** in the command bar.
4. On the left, select **Features**.
5. To enable Microsoft 365 Copilot, set **M365 Copilot in model-driven apps** to **On**. To disable it, set the option to **Off**.



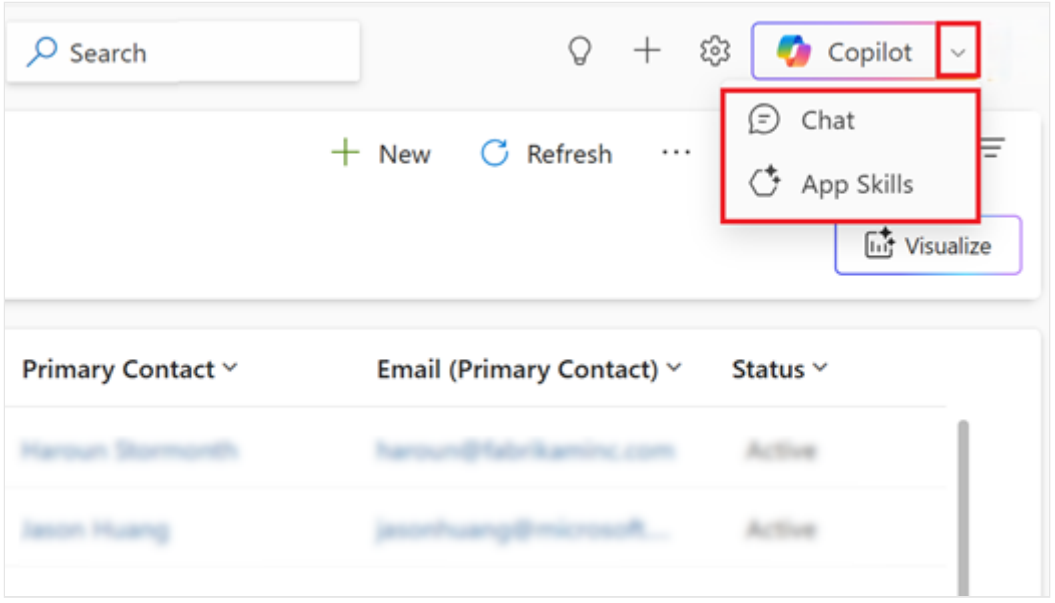
6. Select **Save** and then publish the app for the changes to take effect.

Microsoft 365 Copilot vs. Copilot chat

Microsoft 365 Copilot is gradually replacing Copilot chat in model-driven apps.

During the transition period, makers can enable either one or both chat experiences. When both options are available, the **Copilot** dropdown menu shows the following options:

- **Chat** button opens Microsoft 365 Copilot
- **App Skills** button opens Copilot chat in model-driven apps.



Known limitations

- Microsoft 365 Copilot for model-driven apps allows users to view data by using read-only operations. This capability means that users can only view data that matches their queries and can't make any changes. To make changes, customization with an agent is required.
- Microsoft 365 Copilot for model-driven apps isn't available in the Power Apps mobile app.
- As this feature is being gradually deployed, certain settings in the Power Platform Admin Center might not be accessible yet, depending on the geographic location of your tenant.

Related information

- Customize Microsoft 365 Copilot with an agent (preview)
- Use Microsoft 365 Copilot in model-driven apps

Last updated on 04/23/2026

Customize Microsoft 365 Copilot with an agent

Microsoft 365 Copilot in Power Apps enables users to interact with custom agents that extend Copilot's capabilities beyond standard functionality. By customizing agents, you can create tailored experiences that address your organization's specific business processes and data requirements. For more information, see Use agents in Microsoft 365 Copilot

This article describes the different types of agents you can build and integrate with Microsoft 365 Copilot in Power Apps. You'll learn about the capabilities and use cases for each agent type, helping you choose the right approach for your customization needs.

You can customize agents in several ways to fit your organization's needs.

Expand table

Agent Type	Description	Common use cases
Declarative Agent	Configured using low-code tools and templates, allowing quick setup for common scenarios within Microsoft 365 Copilot.	Automating routine tasks, answering FAQs, and providing guided workflows.
Custom Engine Agent	Developed with custom logic and integrations, offering advanced capabilities tailored to specific business needs.	Complex business processes, custom data integrations, and specialized automation.
Copilot Studio Agent	Built and managed in Microsoft Copilot Studio, enabling integration with Teams and Microsoft 365 for conversational experiences.	Interactive chatbots, team collaboration, and personalized support within Microsoft 365 apps.

Use the following information to find the best solution for your organization:

- Microsoft 365 Copilot Agents Overview documentation provides guidance on how to choose what type of agent to build.
- Create agents as Declarative Agents for Microsoft 365 Copilot or as Custom engine agents for Microsoft 365.
- Use Microsoft Copilot Studio to connect and configure an agent for Teams and Microsoft 365, making it available in Microsoft 365 Copilot.
- Review guidance on how to Choose between Microsoft 365 Copilot and Copilot Studio to build your agent.

Limitation

- Agents you author can't yet use in-app user context to optimize their responses.

Related information

- Add Microsoft 365 Copilot for app users in model-driven apps
- Use Microsoft 365 Copilot in model-driven apps
- Add Microsoft 365 Copilot for app users in canvas apps (preview)
- Use Microsoft 365 Copilot in canvas apps (preview)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 04/15/2026

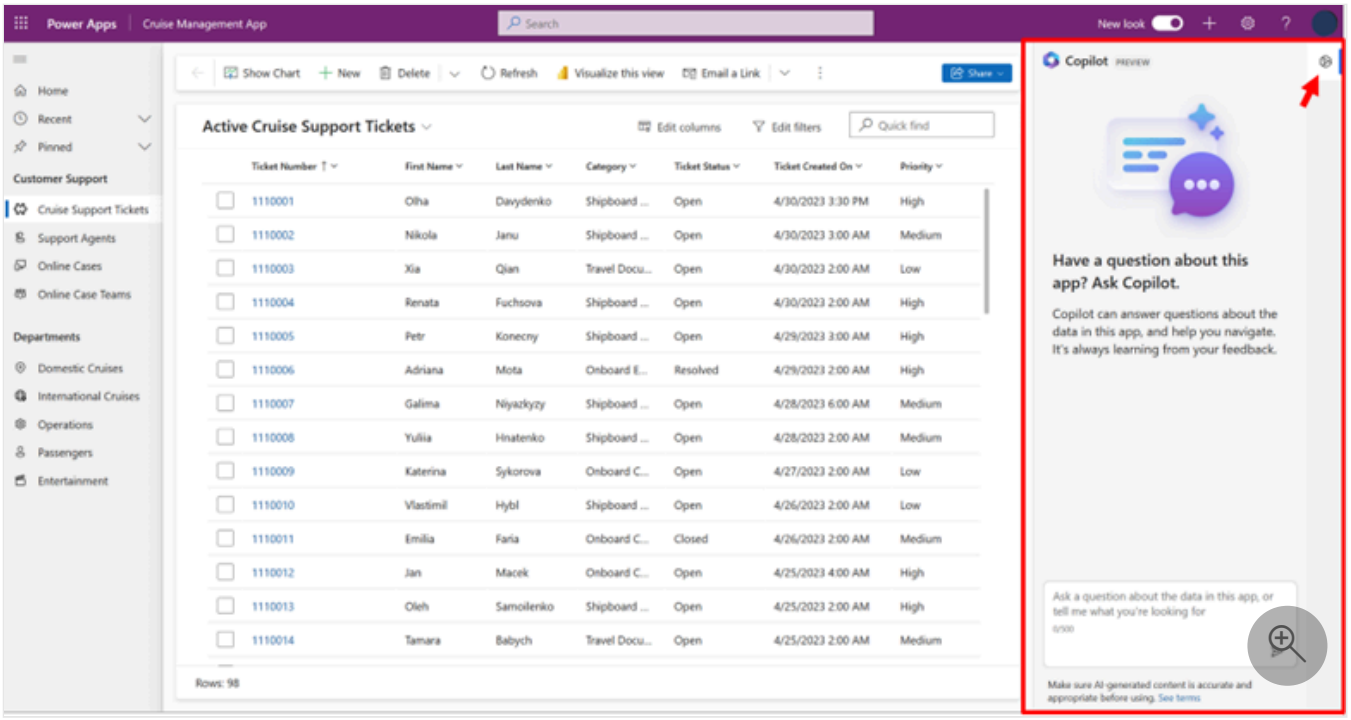
Add Copilot for app users in model-driven apps

Copilot chat for model-driven apps in Power Apps is a next-generation AI assistant that helps app users get insights about the data in their apps through conversation in natural language. Copilot chat helps app users boost their productivity through AI-powered insights and intuitive app navigation.

> **Important**
>
> This feature is generally available in Dynamics 365 apps, but remains a preview feature in Power Apps.
>
> - Preview features aren't meant for production use and might have restricted functionality These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - You must allow data movement across regions for Generative AI features as a prerequisite for using Copilot in Power Apps. This step is especially important if your organization and your environment are in different regions. Learn more in Turn on copilots and generative AI features.
> - This capability is powered by Azure OpenAI Service.
> - This capability might be subject to usage limits or capacity throttling.

> **Note**
>
> Copilot chat in Power Apps is a preview feature that will be deprecated for model-driven apps in environments not enabled for Dynamics 365 apps, starting January 2026. We recommend switching to Microsoft 365 Copilot chat. During the transition period, makers can enable either one or both chat experiences. For more information, see Deprecation of Copilot chat in model-driven apps

When enabled, Copilot chat can be accessed through the Copilot icon in the right navigation bar in a model-driven app. The Copilot chat pane can be opened or minimized as desired.



Copilot chat in model-driven apps can answer questions about the Microsoft Dataverse table data in the environment. Copilot can also help you navigate the app. For example, when a user enters *take me to Online Cases* or *show me Cruise Support Tickets*, Copilot automatically opens the relevant app screens. Learn more in Use Copilot chat in model-driven apps.

Enable Copilot for model-driven apps in your environment

Power Platform administrators enable the Copilot chat feature in model-driven apps for all the users in their environments in the Power Platform admin center.

> **Important**
>
> - Copilot chat for app users in model-driven apps isn't enabled by default. Administrators must manually enable this feature for their environments in the Power Platform admin center.
> - Questions and answers about enterprise data environments that have a customer-managed key or Customer Lockbox won't use your data stored in Dataverse. You might get answers from Copilot, but they won't be based on your Dataverse data.

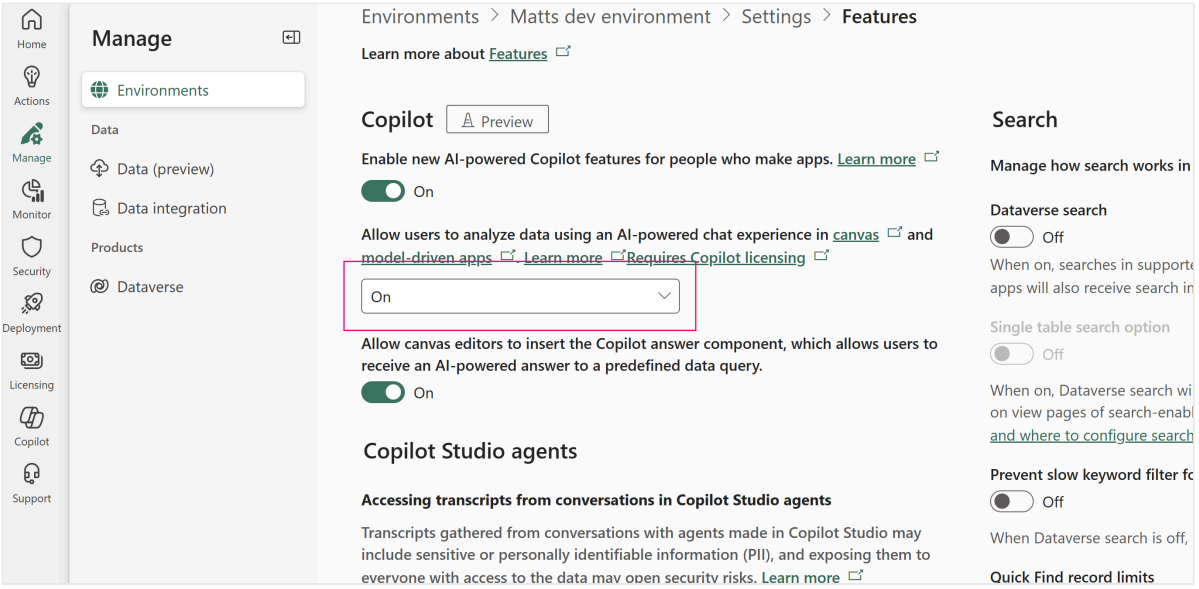
1. Sign in to the Power Platform admin center at https://admin.powerplatform.microsoft.com.

2. In the navigation pane, go to **Manage** > **Environments** and select the environment where you want to enable Copilot chat for app users in model-driven apps. On the command bar, select **Settings**.

3. In **Settings** for the selected environment, select **Product** > **Features**.

4. In the **Features** section, set the value for **Allow users to analyze data using an AI-powered chat experience in canvas and model-driven apps** to the setting that's appropriate for your environment.

    - **Default**. Copilot chat is *disabled for a Power Apps licensed environment and enabled for a Dynamics 365 licensed environment*.
    - **On**. Copilot chat is enabled for the environment regardless of the environment licensing type.
    - **Off**. Copilot chat is disabled for the environment regardless of environment licensing type.

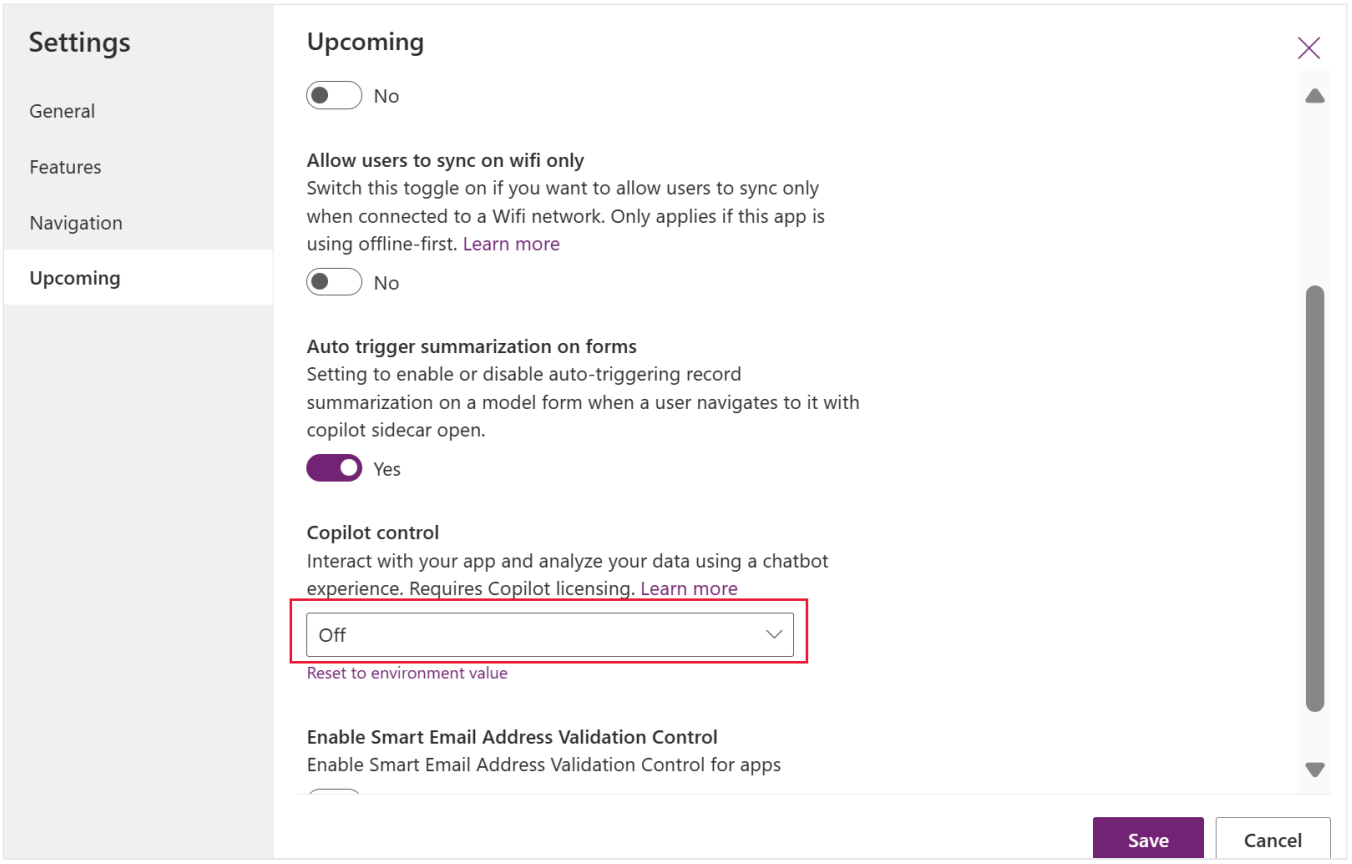


5. **Save** your changes.

Disable Copilot chat for a model-driven app

Makers can disable Copilot chat for a specific model-driven app. In app designer, open the model-driven app for **Edit**, and then select **Settings** on the command bar. Select **Upcoming** on the **Settings** screen, set **Copilot control** to **Default** or **Off**, and then select **Save**.

Save and **Publish** the model-driven app for the changes to take effect.



Reset to environment value

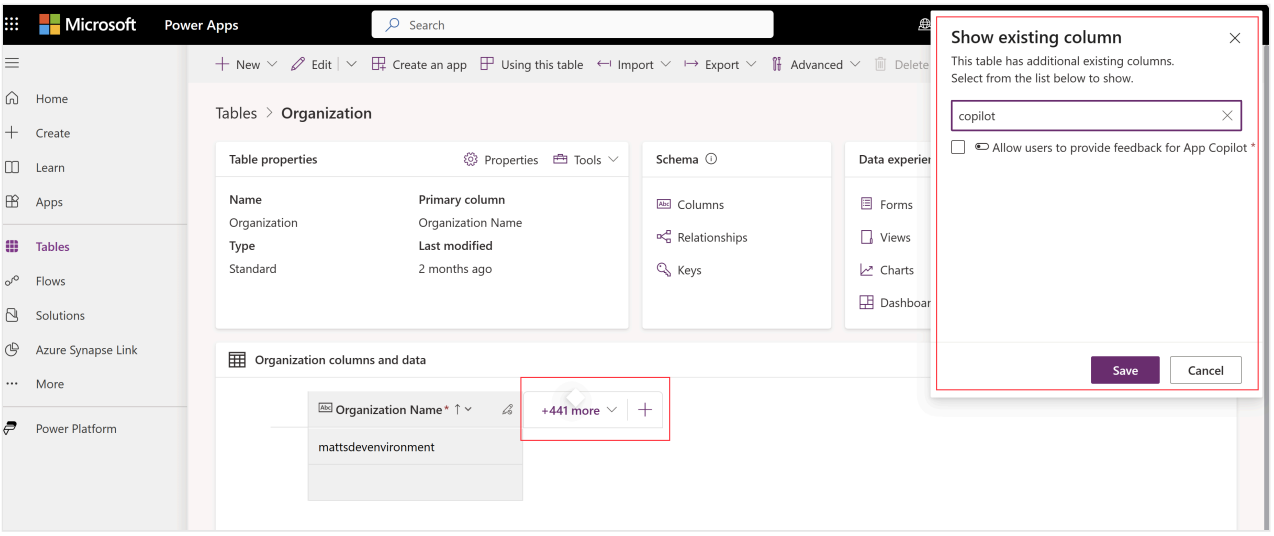
Makers can set the Copilot control for the app to match the environment setting by selecting **Reset to environment value** on the **Upcoming** tab of app settings.

Provide feedback in Copilot

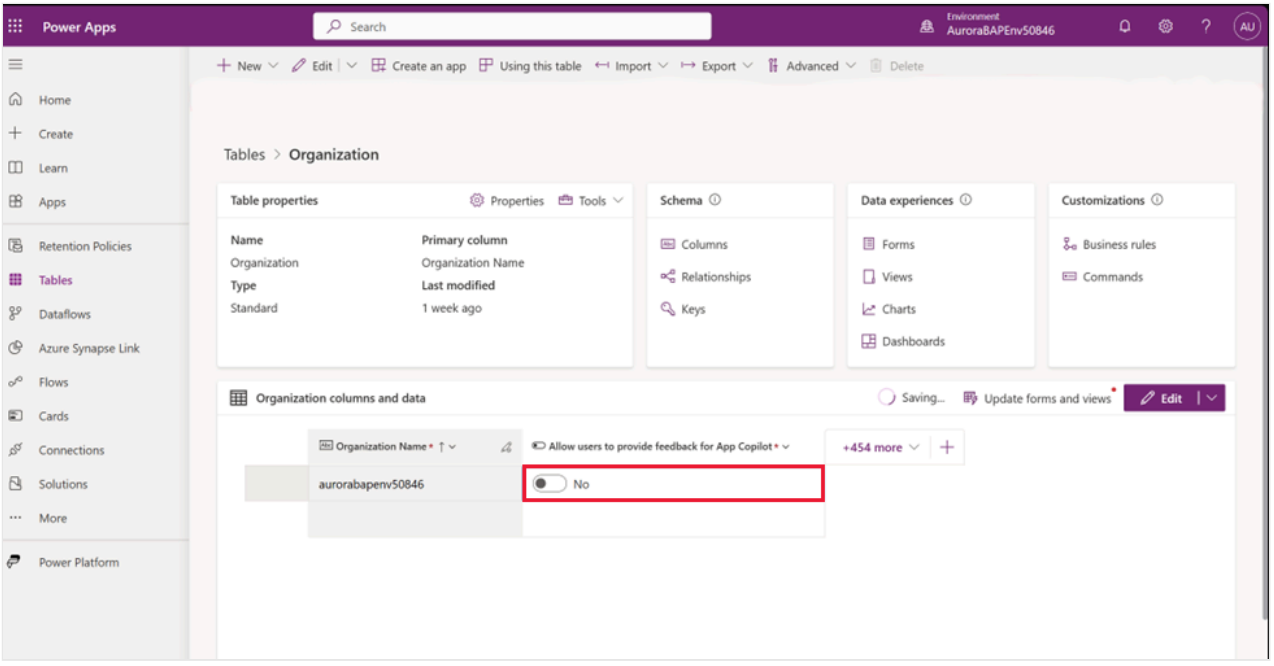
To provide feedback to help us improve Copilot's responses, app users can select the "thumb up" or "thumb down" icon after each Copilot response. Optionally, app users can provide feedback in their own words about what they liked or what went wrong, or how they would like Copilot to improve.

Disable feedback for app users

1. Sign in to Power Apps.
2. Go to **Tables**, and then open the **Organization** table.
3. In the **Organization columns and data** section, select **+nnn more** to display the column search box.
4. In the search box, search for the **Allow users to provide feedback for App Copilot** column.



5. Set the column to **No**, and then **Save** the table.



Region availability and language supported

Copilot chat for model-driven apps in Power Apps is available in the regions and languages listed here.

Expand table

Regions	Language
Asia Pacific, Australia, Brazil, Canada, Europe, France, Germany, India, Japan, Korea, Norway, Qatar, Singapore, South Africa, Sweden, Switzerland, United Arab Emirates, United Kingdom, United States	Czech, Danish, German, Greek, Finnish, French, Italian, Japanese, Korean, Dutch, Norwegian (Bokmål), Polish, Portuguese (Brazil), Russian, Swedish, Thai, Turkish, Chinese (Simplified), Spanish (Spain), Arabic, Hebrew

Copilot takes into account the user's preferred UI language and localizes responses based on that. Depending on the user's preferred UI language, environment base language, and languages supported for a specific Copilot feature, the responses are localized accordingly. This table summarizes the expected behavior in different scenarios.

Expand table

Base language of the environment	Preferred UI language of the user	Expected behavior
English	English	Output in English
English or non-English	One of the supported non-English languages	Output in the preferred UI language of the user.
English or non-English	An unsupported language	Output is unpredictable as the language is unsupported. The responses are mixed with English and the user language. We recommend using one of the supported Copilot languages in such cases.

Known limitations

1. **Copilot for app users** allows users to retrieve information from Dataverse through read-only operations. This means that users can only view data that matches their queries and can't make any changes to the data.
2. Create, update, or other generic actions such as enable or disable Copilot and create a memo aren't supported.
3. Summarization skill feature is available in Dynamics 365 Sales copilot and not in **Copilot for app user**.
4. **Copilot for app users** isn't supported with the Power Apps mobile app.

Related information

- Customize Copilot chat in model-driven apps
- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Power Apps
- Add Copilot control to a canvas app (preview)
- Enable copilots and generative AI features in Power Apps

Last updated on 01/09/2026

Customize Copilot chat using Copilot Studio (preview)

[This article is prerelease documentation and is subject to change.]

Customize Copilot chat to make it even more intelligent and relevant for your organization by adding additional articles, knowledge sources, and more.

> **Important**
>
> This feature is generally available in Dynamics 365 apps, but remains a preview feature in Power Apps.
>
> - Preview features aren't meant for production use and might have restricted functionality These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - You must allow data movement across regions for Generative AI features as a prerequisite for using Copilot in Power Apps. This step is especially important if your organization and your environment are in different regions. Learn more in Turn on copilots and generative AI features.
> - This capability is powered by Azure OpenAI Service.
> - This capability might be subject to usage limits or capacity throttling.

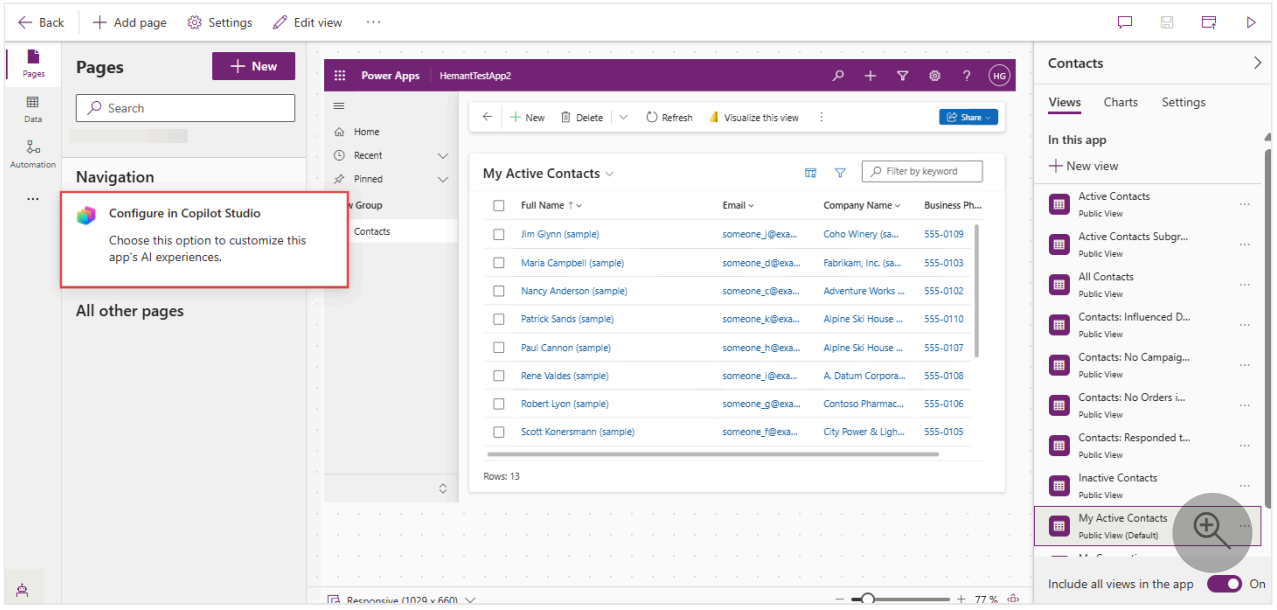
> **Note**
>
> Copilot chat in Power Apps is a preview feature that will be deprecated for model-driven apps in environments not enabled for Dynamics 365 apps, starting January 2026. We recommend switching to Microsoft 365 Copilot chat. During the transition period, makers can enable either one or both chat experiences. For more information, see Deprecation of Copilot chat in model-driven apps

You customize Copilot chat using Microsoft Copilot Studio and expand the capabilities to go beyond just handling Microsoft Dataverse tables questions and out-of-the-box skills. Before customizing, make sure Copilot chat is enabled for your environment.

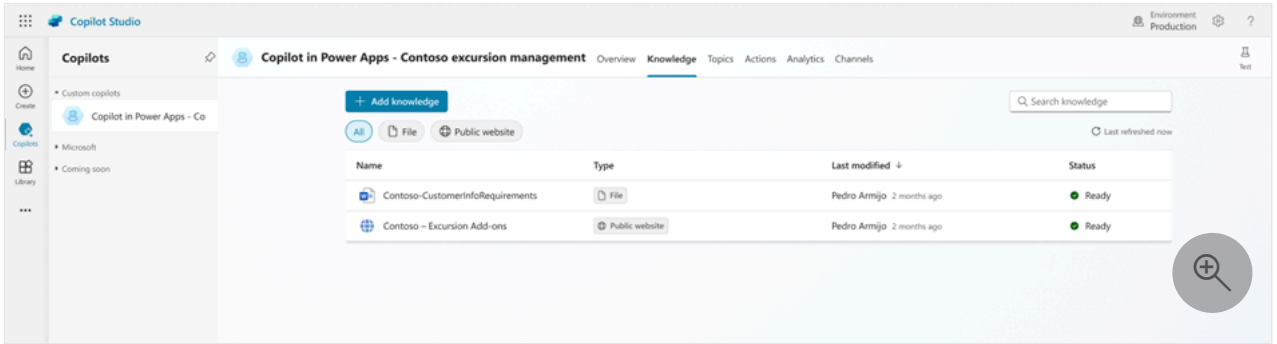
https://learn-video.azurefd.net/vod/player?id=d0439343-f464-4ec7-a92d-b3c528430fb5&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fmodel-driven-apps%2Fcustomize-copilot-chat

> **Note**
>
> - Copilot Studio license and agent editing permissions are required to customize Copilot chat.
> - This feature is only available in standalone model-driven apps, which don't include both lead and opportunity tables. This feature isn't yet supported for Dynamics 365 apps.
> - This experience has changed to use interactive agent and is currently rolling out to regions.

1. Go to https://make.preview.powerapps.com .
2. Open your model-driven app in edit mode, and then on the left navigation bar select ... > **Configure in Copilot Studio**. You're taken to Microsoft Copilot Studio where your app's agent is set up. Every standalone model-driven app Copilot has its own dedicated agent available for customizations. Setting up the agent for the first time takes a few seconds.



3. Customize your agent by adding knowledge sources or topics. Customizing this agent only impacts the Copilot chat of the specific app it's provisioned for.



4. **Publish** the agent after you make customizations to ensure changes are available to users.

Power Apps creates a platform-owned agent named **Copilot in Power Apps** in each environment. This agent is only editable by a Power Platform environment admin and isn't published to any channels by default. Although this agent can be viewed in Microsoft Copilot Studio, this agent shouldn't be edited since it isn't used by model-driven apps.

Related information

- Add Copilot chat to model-driven apps
- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Power Apps
- Enable copilots and generative AI features in Power Apps

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 01/09/2026

Add knowledge to Copilot chat in model-driven apps (preview)

Article • 02/22/2025

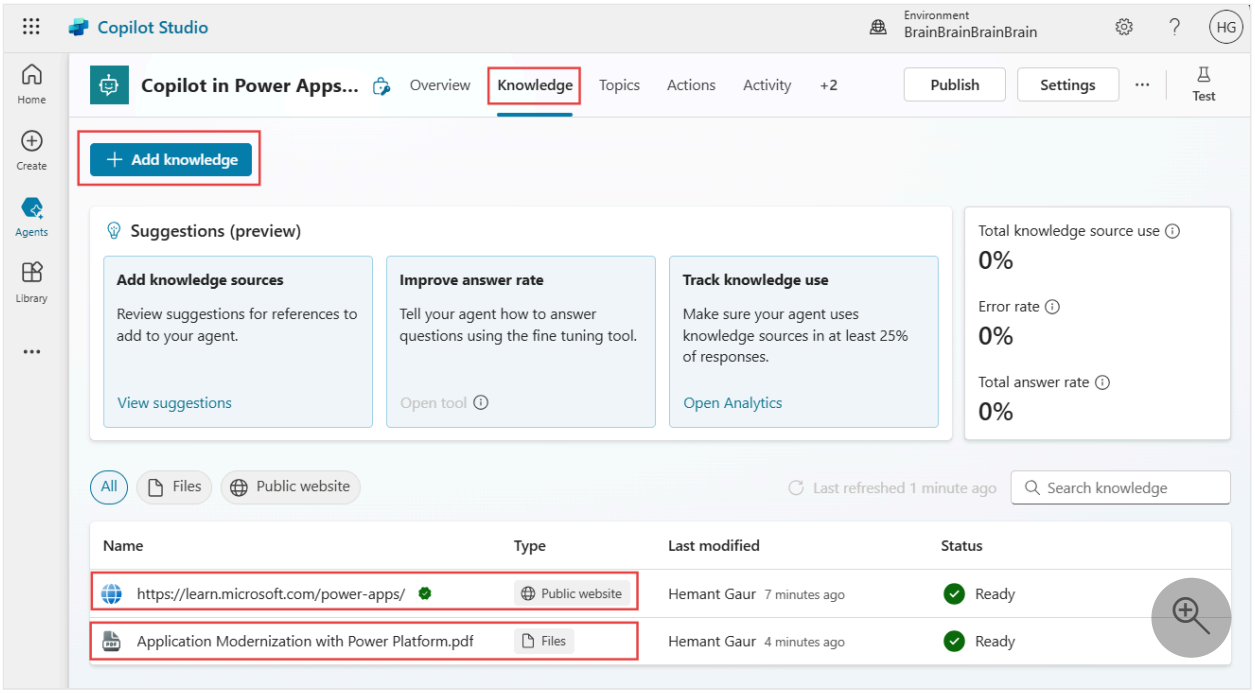
[This topic is pre-release documentation and is subject to change.]

Extend your app's Copilot chat intelligence by adding additional knowledge sources in Copilot Studio.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Add knowledge to Copilot chat

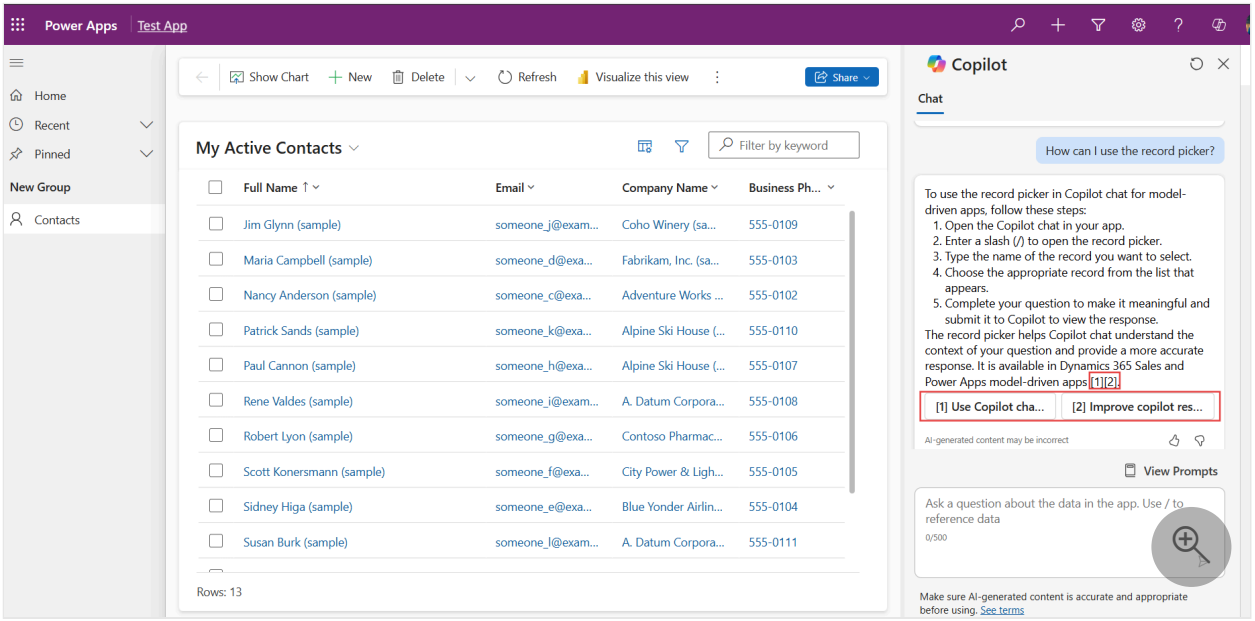
You can add additional knowledge sources in Copilot Studio. For example, you can add a link to an external public-facing website like Power Apps documentation by adding `https://learn.microsoft.com/power-apps/` as knowledge to enable your Copilot chat to respond to questions related to creating apps in Power Apps. Another example is to upload your organization's internal knowledge as a document to enable Copilot chat to respond to relevant queries that aren't a part of the app data.



More information: Add knowledge to an existing agent – Microsoft Copilot Studio.

> **Note**
>
> - Currently only Public website, File upload and SharePoint knowledge source types are supported. Dataverse knowledge isn't part of this preview.
> - Copilot studio Generative AI orchestration isn't supported currently. You can use classic orchestration topic whose trigger phrases match most closely with the user's query for a given skill.

Once knowledge is enabled, app users can ask relevant questions to get responses along with the knowledge references.



Related articles

- Customize Copilot chat using Copilot Studio (preview)
- Add topics to Copilot chat in model-driven apps (preview)
- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Power Apps
- Enable copilots and generative AI features in Power Apps

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Add topics to Copilot chat in model-driven apps (preview)

Article • 02/22/2025

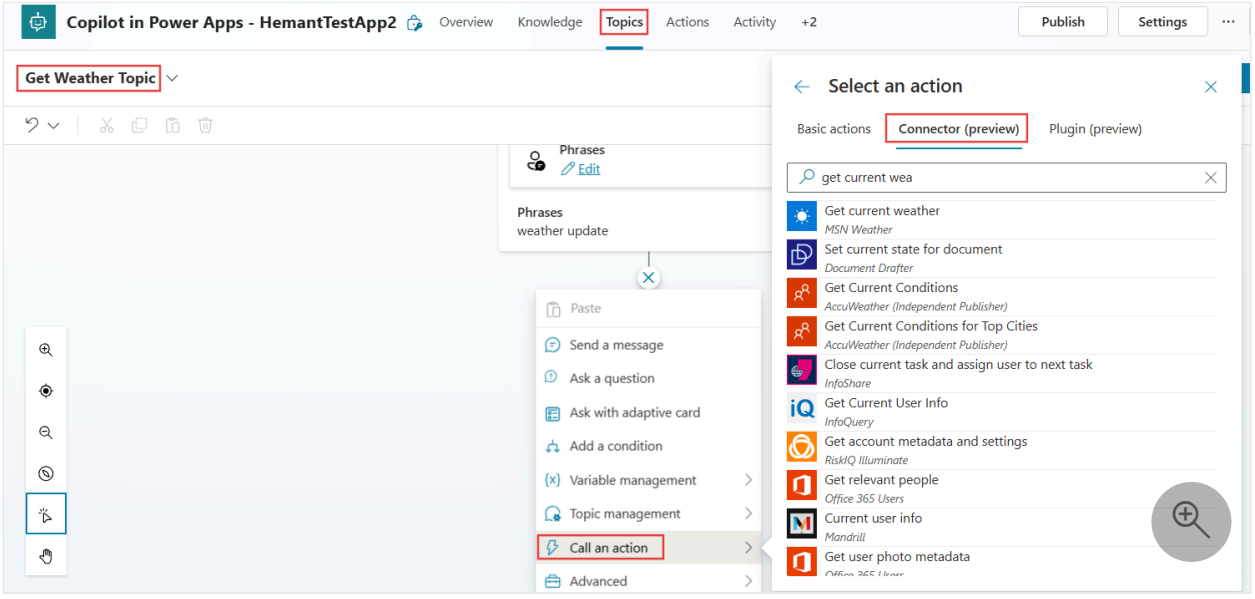
[This topic is pre-release documentation and is subject to change.]

Extend your app's Copilot chat intelligence by adding topics in Copilot Studio.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Add new topics to Copilot chat

In Copilot Studio, you can add topics to your app's Copilot agent. These topics can be customized to use various trigger types and can respond with simple messages, adaptive cards, or generative answers. Additionally, topics can also initiate actions like flows, connectors, and Dataverse plug-ins enabling seamless point in time integration with external systems.



More information: Create and edit topics – Microsoft Copilot Studio.

> **Note**
>
> Copilot Studio has inline capability to "Test your agent" and can be used to validate topics as they're added. However, topics using out-of-the-box model-driven app custom variables like `Global.PA__Copilot_Model_PageContext.pageContext.id` can only be tested in the published Copilot.

Related articles

- Customize Copilot chat using Copilot Studio (preview)
- Add knowledge and topics to Copilot chat in model-driven apps (preview)
- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Power Apps
- Enable copilots and generative AI features in Power Apps

Note: The author created this article with assistance from AI. Learn more

---

Feedback

Was this page helpful? Yes No

Provide product feedback

Send a message to Copilot chat (preview)

Article • 03/17/2025

[This topic is pre-release documentation and is subject to change.]

Extend your app's Copilot chat engagement and affect by incorporating rich audio, video, and simple cards as a chat response type. The **Message** node, within the flow of a topic in Copilot Studio, can be used to send a message from the agent to the user in your model-driven app. Messages can be simple text messages, but can also include richer components, such as images, videos, quick replies, and cards.

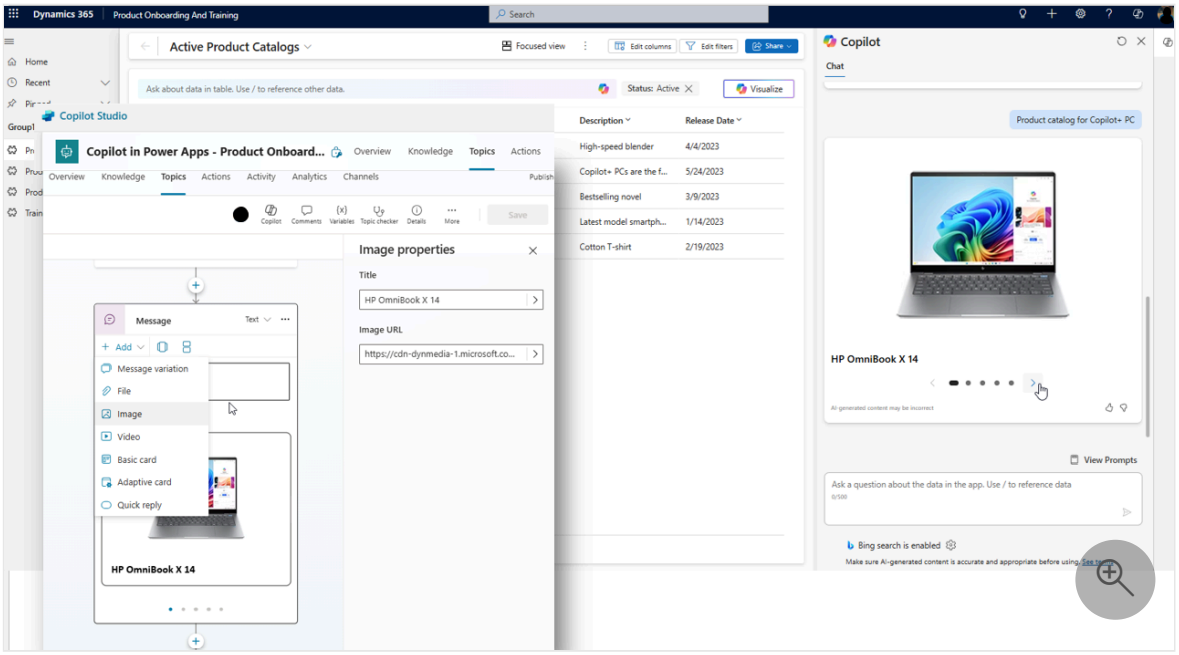
> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - Copilot Studio license and agent editing permissions are required to customize Copilot chat.

Add an image response to Copilot chat

When you add an image, Copilot can send a message as a response with the image from the agent to the user. To add an image, the image must be hosted through a URL.

1. Add a Message node.
2. In the node's menu bar, select **Add** and choose **Image**.
3. Under **Image properties**, enter the **Image URL** for your image.

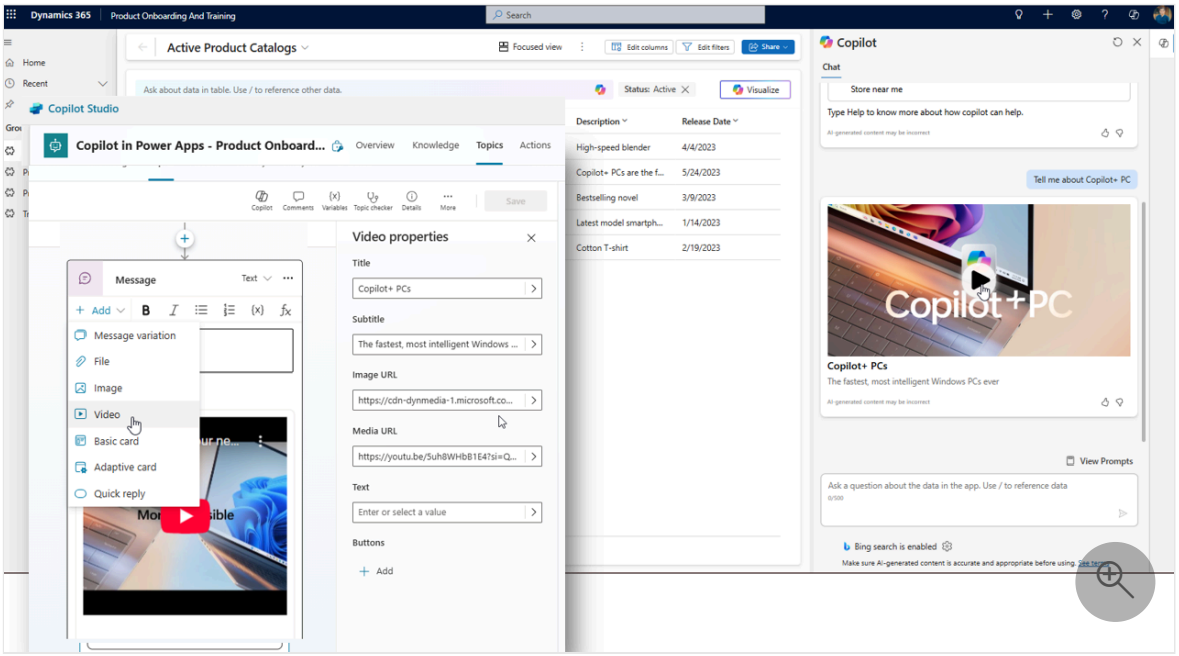
4. Optionally, enter a **Title** for the image.



Add a video response to Copilot chat

When you add a video, Copilot can send a message with the video as a response from the agent to the user.

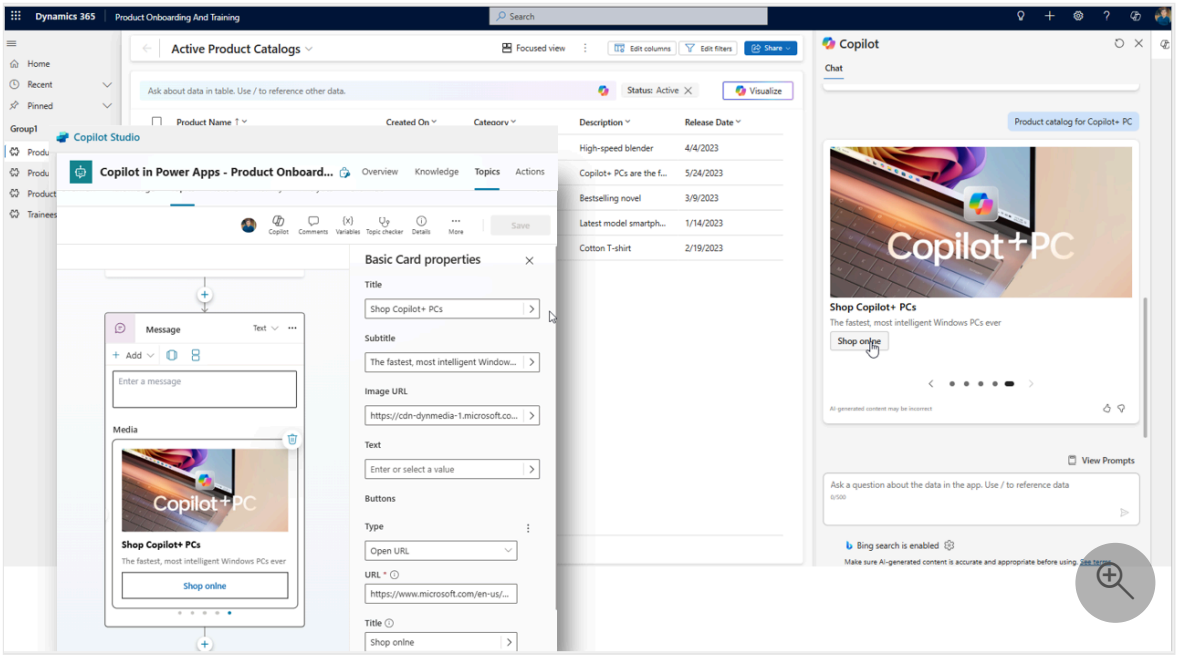
1. Add a Message node.
2. In the node's menu bar, select **Add**, and then select **Video**.
3. Under **Media URL**, enter the URL of your video. The URL can either be a direct link to a publicly accessible MP4 file or a YouTube URL.
4. Optionally, enter a **Title**, **Subtitle**, **Image URL** (the URL of a publicly available image file), or **Text** to be shown alongside the video on the card. You can also add one or more buttons, which have the same properties as quick replies.



Add a basic card response to Copilot chat

A basic card is a general-purpose card used for adding text, images, and interactive elements to an agent response message.

1. Add a Message node.
2. In the node's menu bar, select **Add**, and then select **Basic card**.
3. In the **Basic Card properties** pane, fill in the properties for the content of your card. You can also add one or more buttons, which have the same effect as quick replies.



For more information about sending messages with Copilot, go to send a message Copilot Studio documentation

Related articles

- Customize Copilot chat using Copilot Studio (preview)
- Add topics to Copilot chat in model-driven apps (preview)
- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Power Apps
- Enable copilots and generative AI features in Power Apps

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Feedback

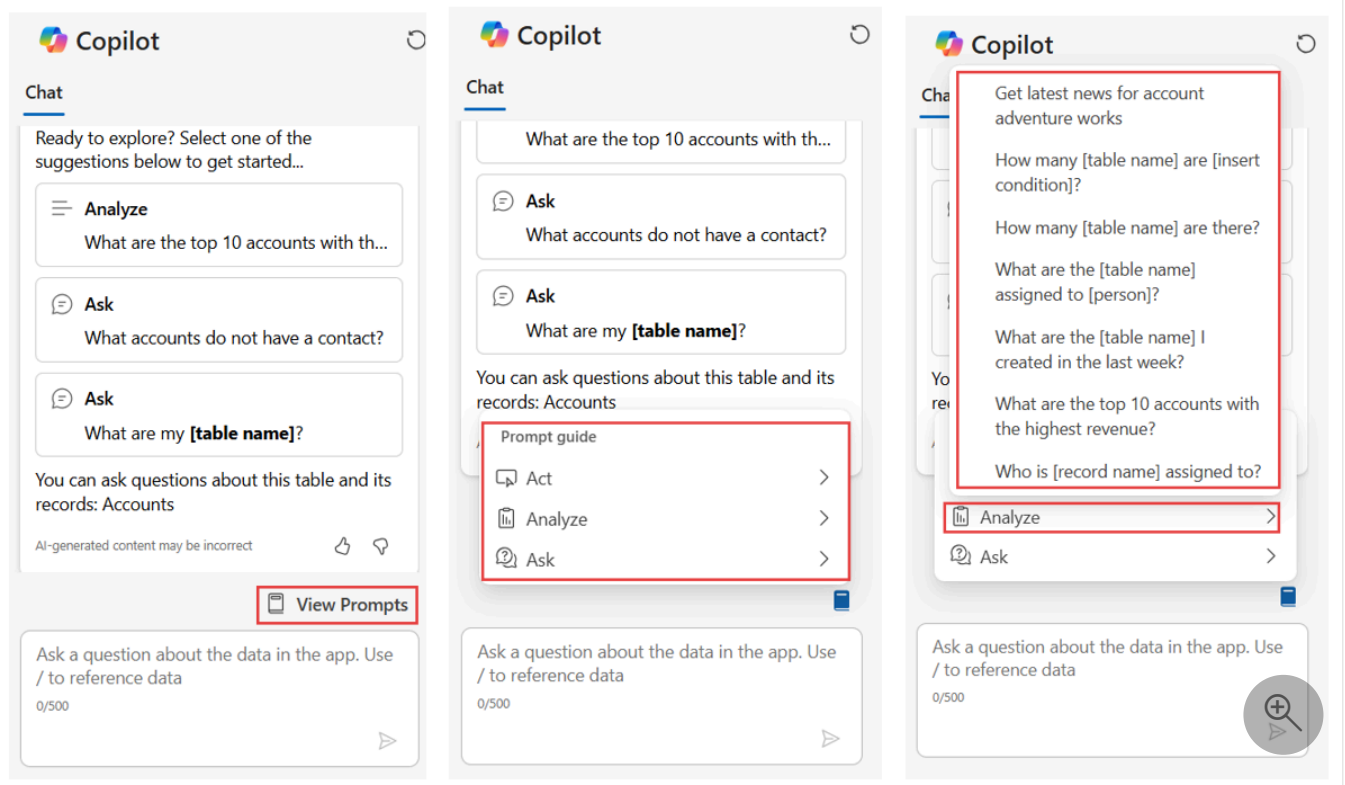
Was this page helpful? Yes No

Provide product feedback

Prompt guide customizations (preview)

[This topic is pre-release documentation and is subject to change.]

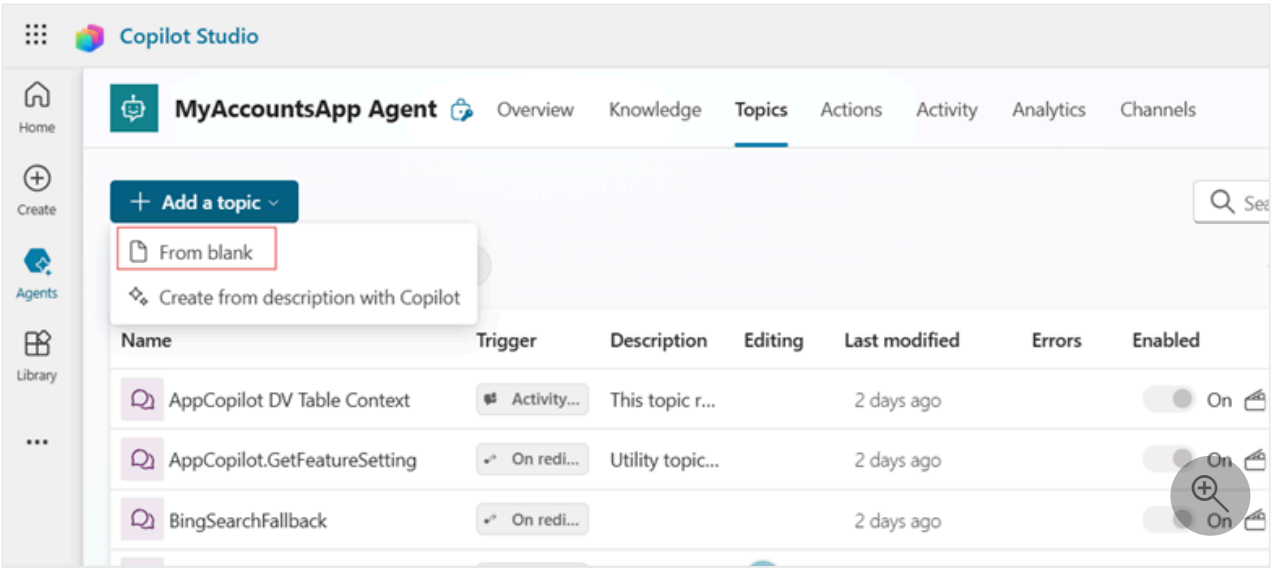
A prompt library is a collection of prewritten, tested, and optimized prompts designed to help shape the interactions and responses of the Copilot chat. They ensure that the Copilot chat provides relevant, accurate, and contextually appropriate information based on the user's needs and preferences.



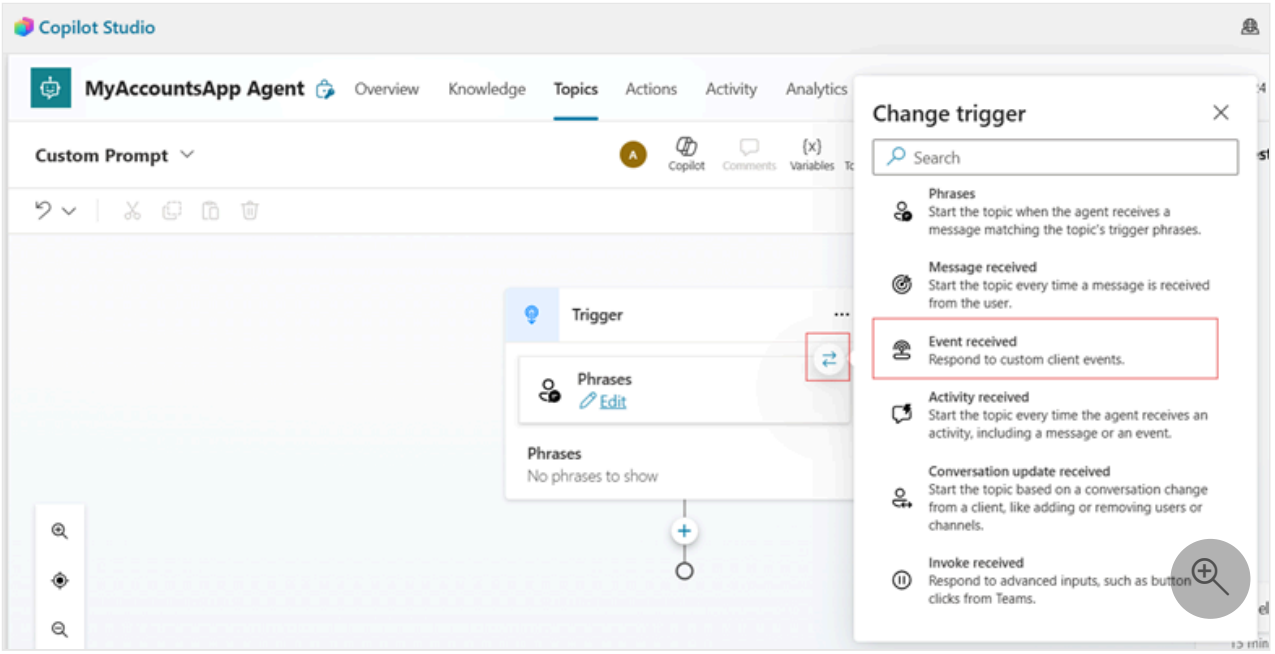
> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

The following steps detail how to add specific queries to the prompt guide. A *Power Apps Help* section is appended to the existing out-of-the-box Copilot prompt guide. Alternatively, you can copy the sample code into a new topic directly from the Prompt guide customizations topic sample. All the prompts shown to the end user via the prompt guide are stored in the Copilot Studio agent used for the app.

1. Open the agent backing the app in Copilot Studio and add a new blank topic.

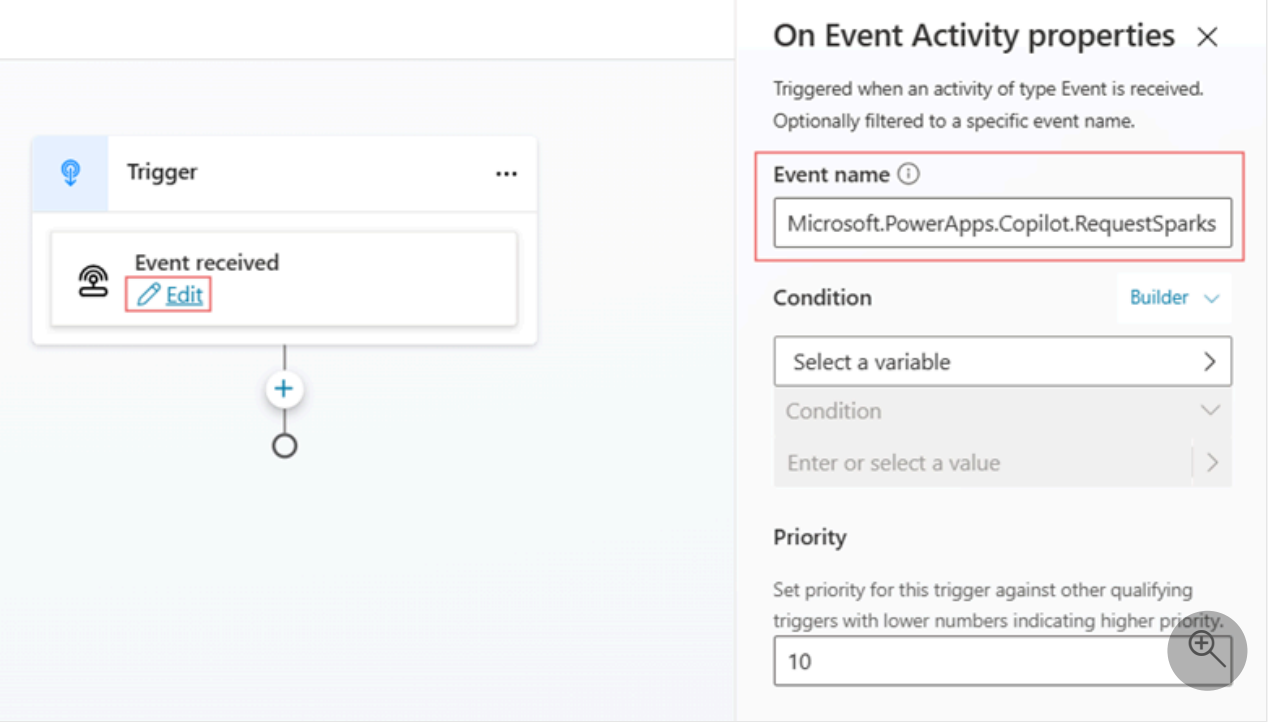


2. Rename the topic to reflect the topic intent and change the topic trigger to **Event received**.



3. Select **Edit** under **Event received**, and then set the event name as `Microsoft.PowerApps.Copilot.RequestSparks`, which is the reserved name for prompt

guide.

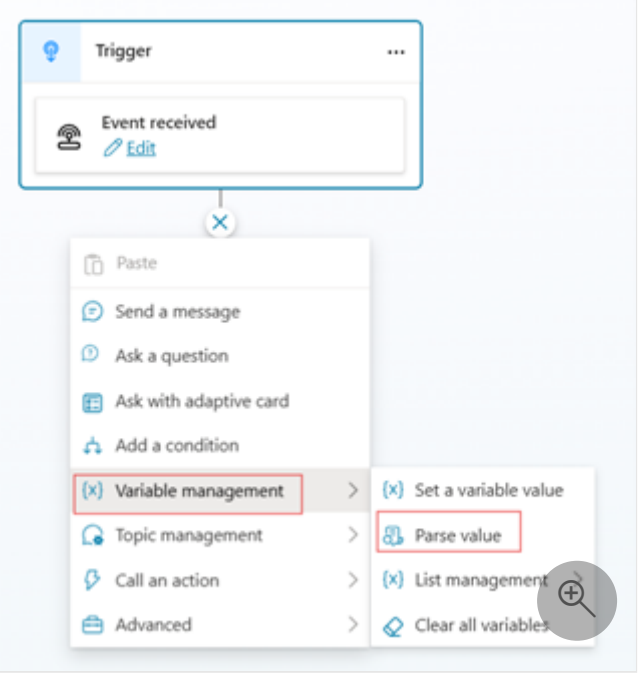


4. Optionally, you can set the conditions to prompt entries in case they're specific to the page context. For example, this prompt entry checks if the page context's table type name matches the specified value. If the condition is true, the custom prompts are shown to the user.

   ```
   condition:Global.PA__Copilot_Model_PageContext.pageContext.entityTypeName = "Entity name"
   ```

5. Under **Priority**, add an appropriate priority value so the trigger is fired after the higher priority topics. Priority values can have 0 to 10K range with 0 being highest. Although about 200 is recommended as it allows for more options to add higher priority topics later, 10 is used in this example.

6. Select + under **Event received**, and then select **Variable management** > **Parse value** to add a next step for variable management parse value.



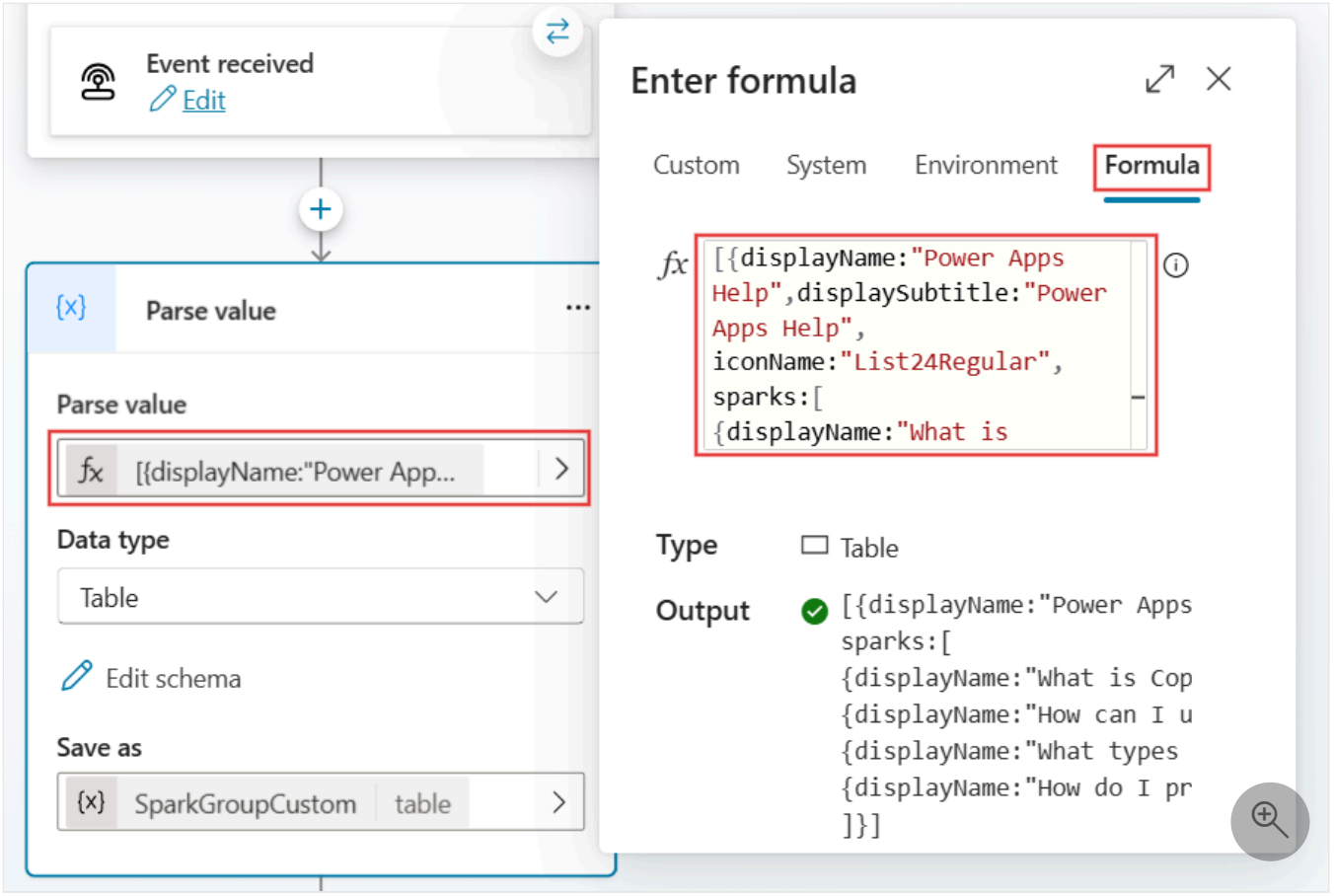
7. Paste the following Power Fx formula into the **Parse value** box, and then select **Insert**.

PowerApps Formula

```
[{displayName:"Power Apps Help",displaySubtitle:"Power Apps
Help",iconName:"List24Regular",sparks:[{displayName:"What is Copilot
chat?",type:"MCSMessageSkill"},{displayName:"How can I use the record
picker?",type:"PromptTextSkill"},{displayName:"What types of questions can I
ask Copilot?",type:"PromptTextSkill"},{displayName:"How do I provide feedback
on Copilot’s responses?",type:"PromptTextSkill"}]}]
```

For the sparks type you can either use `MCSMessageSkill`, which are directly sent to Copilot Studio as user messages or `PromptTextSkill` when you want to populate the **Chat Input** box. `PromptTextSkill` is useful when you want additional input from the user, such as specifying a record or table name among other things. For example: `How many **[table name]** are`

`active?` `What are the **[table name]** assigned to me?`

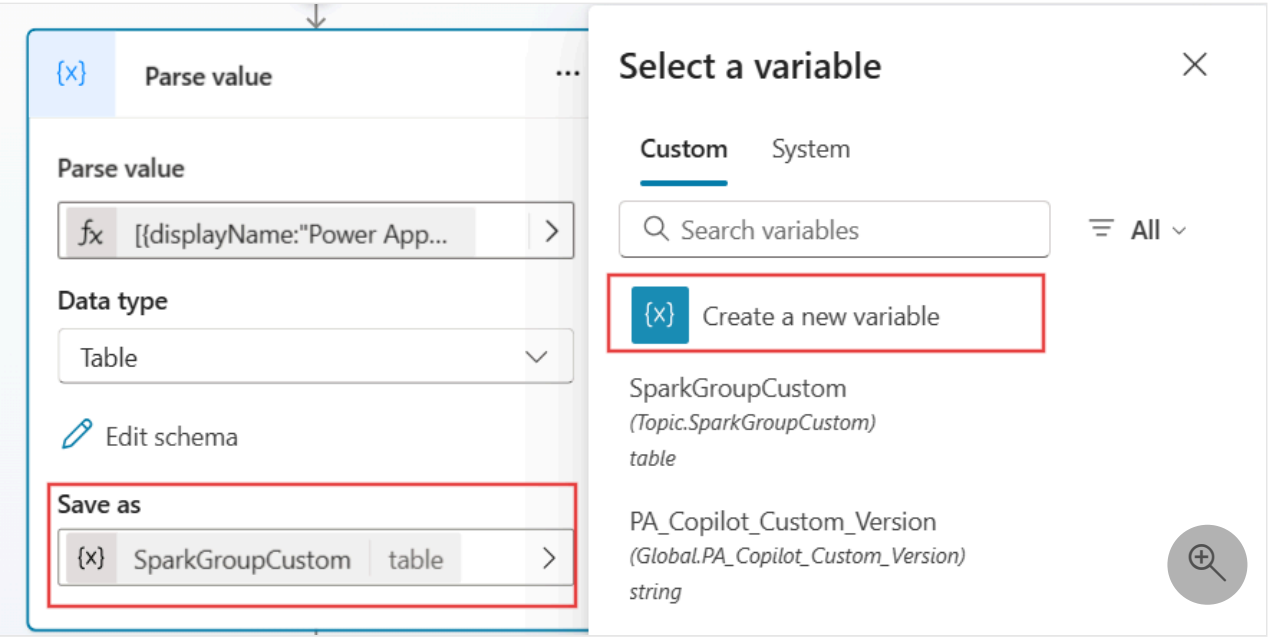


1. Set the **Data type** as **Table**. The **Edit schema** link appears.

2. Select **Edit schema** and paste the following schema, and then select **Confirm**.

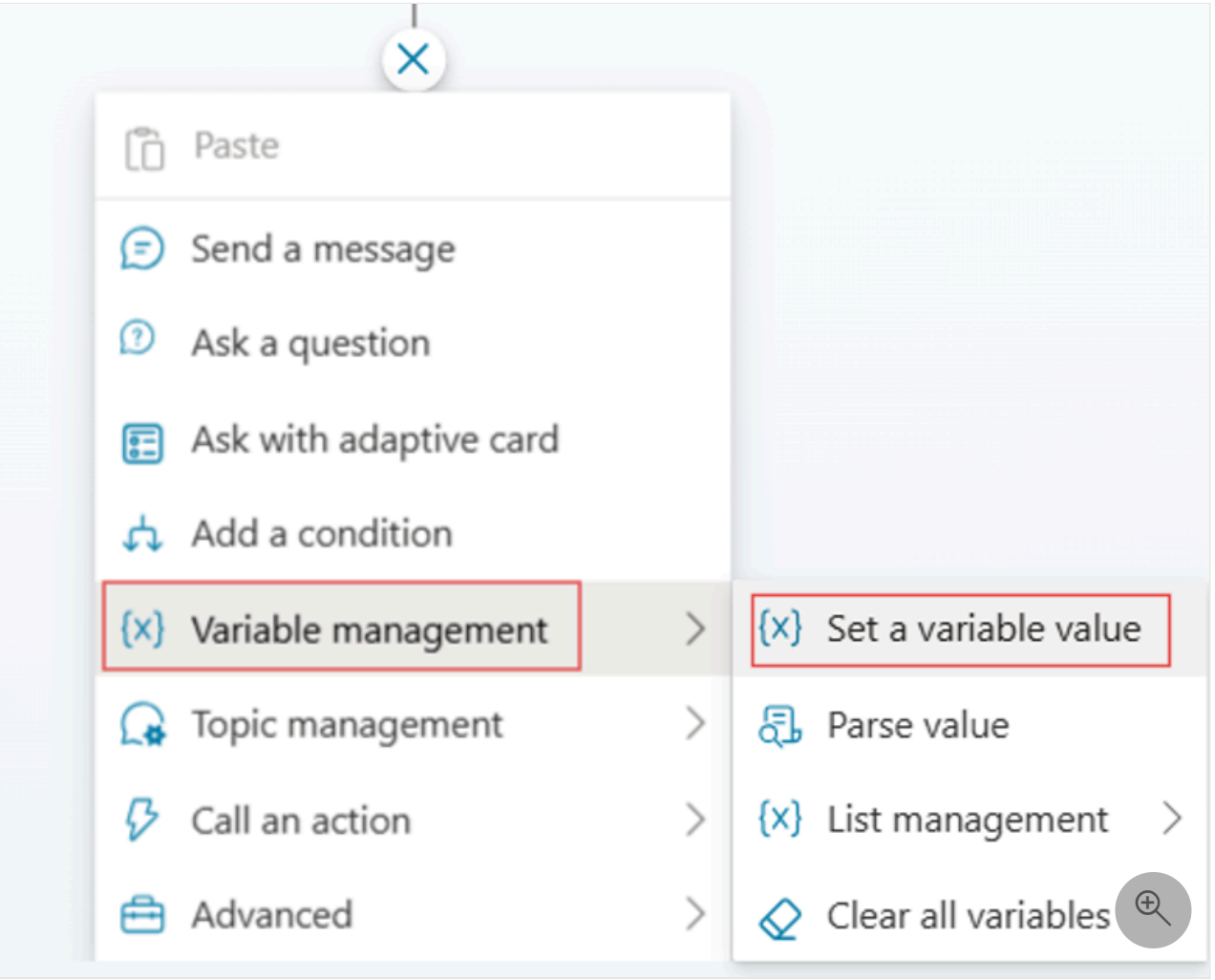
```yml
kind: Table
properties:
  displayName: String
  displaySubtitle: String
  iconName: String
  sparks:
    type:
      kind: Table
      properties:
        displayName: String
        eventName: String
        iconName: String
        payload: String
        type: String
```

3. Set **Save as** to save as a new custom variable and name it something meaningful such as *SparkGroupCustom*.



4. Select + under the **Parse value** step, and then select **Variable management** > **Set a variable value**.

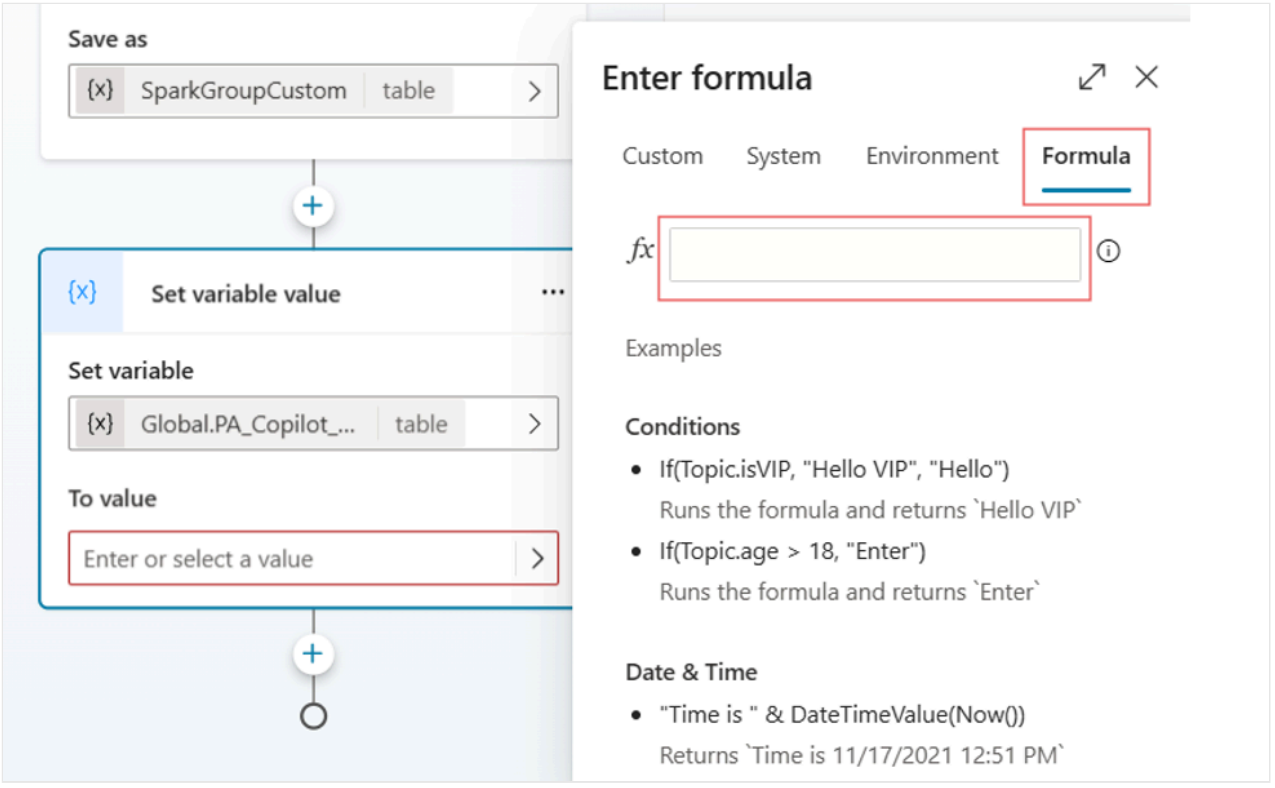
5. The sparks definition is saved in a global variable so you need to set the variable **Global** and name it `PA_Copilot_Sparks.sparkGroups` and/or `Global.PA_Copilot_Sparks.sparks`. This populates the flyout with your prompts. Next, add a step to set variable value.



6. Search for the sparks definition name from the previous step, such as `Global.PA_Copilot_Sparks.sparkGroups`, and set the value to the following Power Fx merge function.

PowerApps Formula

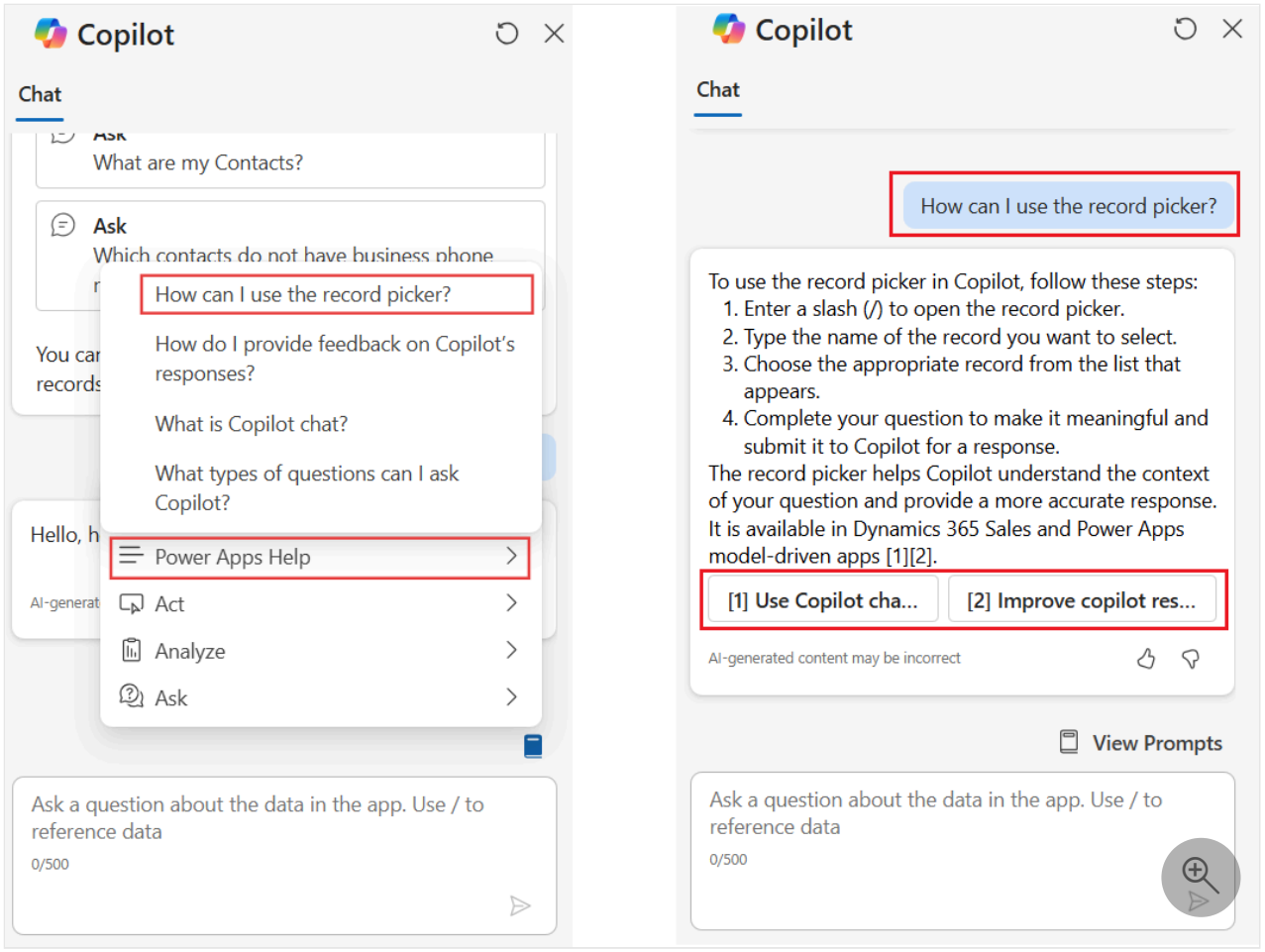
```
ForAll(Sequence(CountRows(Global.PA_Copilot_Sparks.sparkGroups)+CountRows(Topic
.SparkGroupCustom)),
If(Value<=CountRows(Global.PA_Copilot_Sparks.sparkGroups),Index
(Global.PA_Copilot_Sparks.sparkGroups,Value), Index(Topic.SparkGroupCustom,
Value - CountRows(Global.PA_Copilot_Sparks.sparkGroups))))
```



Merge

Replace the variable name with the variable name you used for the custom prompts, which in this example is *SparkGroupCustom*.

7. **Publish** the agent and play the app.



Prompt guide customizations topic sample

Here is the full topic code, which can be copied directly into the new topic.

yml

```yml
kind: AdaptiveDialog
beginDialog:
  kind: OnEventActivity
  id: main
  priority: 200
  eventName: Microsoft.PowerApps.Copilot.RequestSparks
  actions:
    - kind: ParseValue
      id: iCepPf
      variable: Topic.SparkGroupCustom
      valueType:
        kind: Table
        properties:
          displayName: String
          displaySubtitle: String
          iconName: String
          sparks:
            type:
              kind: Table
              properties:
```

```
                displayName: String
                eventName: String
                iconName: String
                payload: String
                type: String
      value: |-
        =[{displayName:"Power Apps Help",displaySubtitle:"Power Apps
Help",iconName:"List24Regular",
        sparks:[
        {displayName:"What is Copilot chat?",type:"MCSMessageSkill"},
        {displayName:"How can I use the record picker?",type:"MCSMessageSkill"},
        {displayName:"What types of questions can I ask
Copilot?",type:"PromptTextSkill"},
        {displayName:"How do I provide feedback on Copilot's
responses?",type:"PromptTextSkill"}
        ]}]

    - kind: SetVariable
      id: setVariable_pDu9cr
      variable: Global.PA_Copilot_Sparks.sparkGroups
      value:
=ForAll(Sequence(CountRows(Global.PA_Copilot_Sparks.sparkGroups)+CountRows(Topic.Spa
rkGroupCustom)), If(Value<=CountRows(Global.PA_Copilot_Sparks.sparkGroups),Index
(Global.PA_Copilot_Sparks.sparkGroups,Value), Index(Topic.SparkGroupCustom, Value -
CountRows(Global.PA_Copilot_Sparks.sparkGroups))))
```

> **Note**
>
> If your agent supports multiple languages and needs prompt guide translation, all your user facing question strings must be set using a `SetTextVariable`.

- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Power Apps
- Enable copilots and generative AI features in Power Apps

Note: The author created this article with assistance from AI. Learn more

---

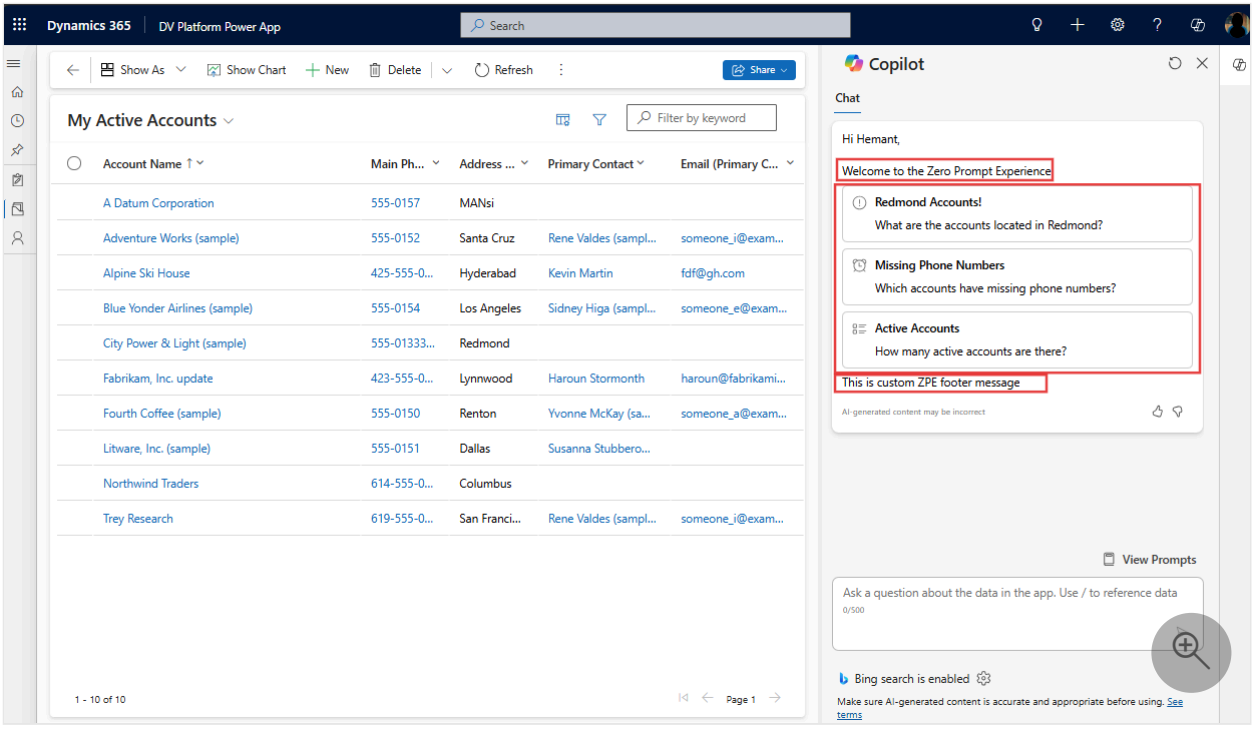
Last updated on 01/09/2026

Zero prompt experience (preview)

Article • 03/11/2025

[This topic is pre-release documentation and is subject to change.]

The zero prompt experience helps makers enhance user engagement and streamline interactions at the start of a Copilot chat session. By presenting a zero prompt experience adaptive card at the beginning of a chat session, users receive relevant information and options right away, reducing the need for additional prompts and iterations. The zero prompt experience can be context aware and hence, can be selectively shown for targeted pages.

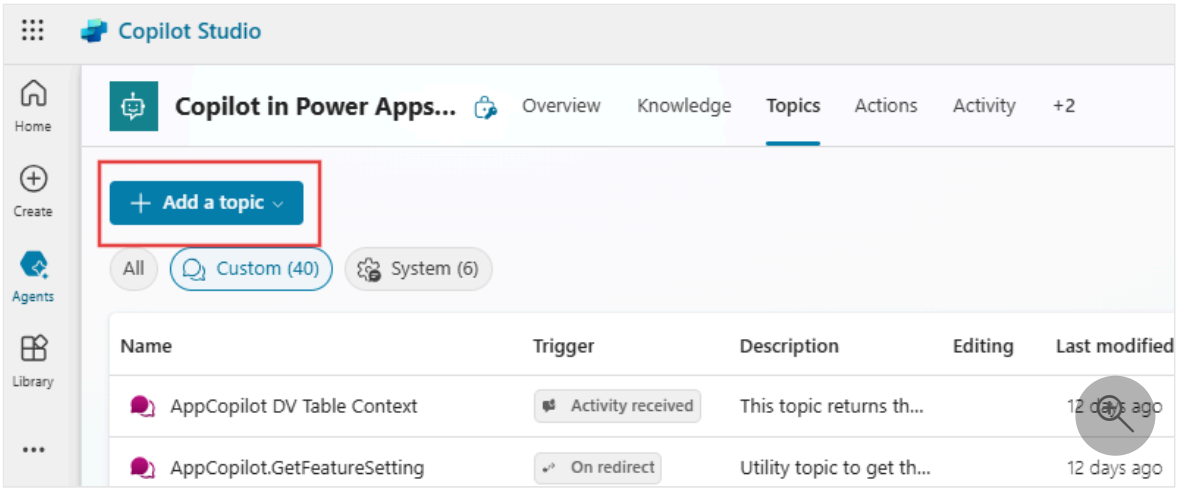


> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

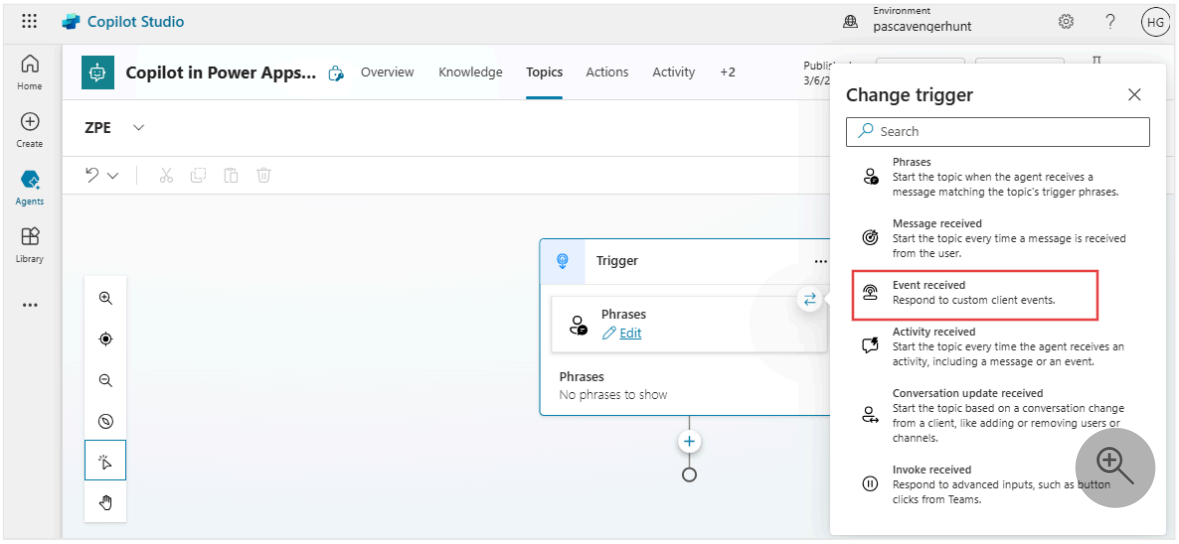
Customize the zero prompt experience

These steps detail how to customize the zero prompt experience.

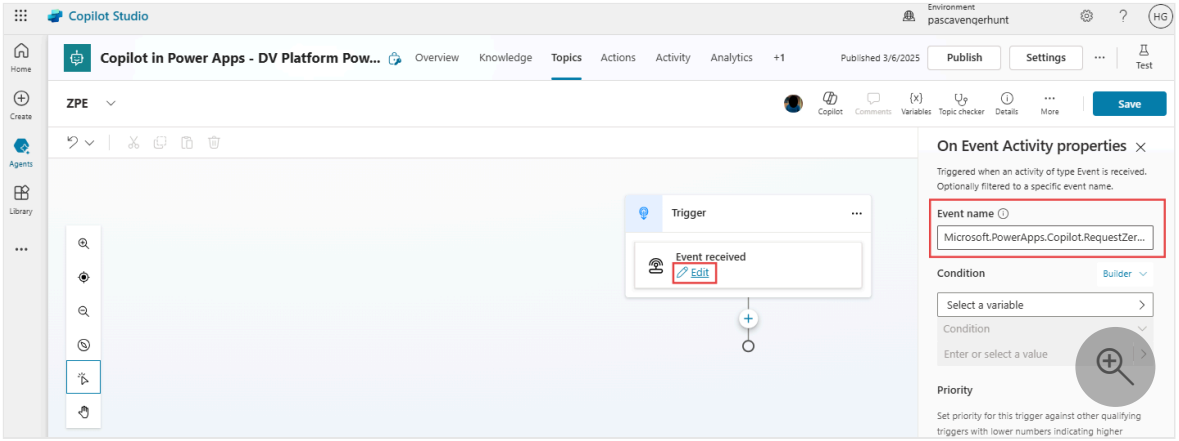
1. Open the agent backing the app in Copilot Studio and add a new blank topic.



2. Rename the topic to reflect the topic intent and change the topic trigger to **Event received**.



3. Select **Edit** under **Event received**, and then set the event name as `Microsoft.PowerApps.Copilot.RequestZeroPrompt`, which is the reserved name for the zero prompt experience. Set priority to higher than 99 but lower than 100000. The priority is important because custom zero prompt experiences need to override the default platform provided ones.



4. Optionally, you can also set the conditions to the zero prompt experience in case it's specific to the page context. For example, this entry checks if the page context's table type name matches account. If the condition is true, the custom zero prompt experience is shown. `condition:Global.PA__Copilot_Model_PageContext.pageContext.entityTypeName = "account"` After this step, you can build your zero prompt messages using adaptive cards. For more information regarding building adaptive cards go to this information: https://adaptivecards.microsoft.com/. Once you have zero prompt experience cards, you can set the global variable `Global.PA_Copilot_ZeroPrompt` to your adaptive card definition.

5. The zero prompt includes all the flexibility of adaptive cards and you can trigger different skills from within it. When an adaptive card contains a button or anything that requires an `Action.Submit`, you have a few options you can use to handle that event. These event handlers are called `SkillTypes`. `Action.Submit` types should include the following properties: a data object with `SkillType` and scenario properties. You can use `MCSMessageSkill`, which are directly sent to Copilot Studio as user messages, or `PromptTextSkill` when you want to populate the **Chat Input** box. `PromptTextSkill` is useful when you want additional input from the user, such as specifying a record or table name among other things. For example: `How many [table name] are active?` `What are the [table name] assigned to me?`

6. When you trigger zero prompts, your select an action structure that should look like the action structure here. The scenario value should be `ZeroPromptCard` along with the source value as `ZeroPrompt`. Lastly, the value corresponds to the actual prompt.

yml

```
   selectAction: {
   type: "Action.Submit",
   data: {
    scenario: "ZeroPromptCard",
    skillType: "MCSMessageSkill",
    value:  "What are accounts in Redmond?",
    source: "ZeroPrompt"
   }
},
```

Zero prompt experience topic sample

Here is the full topic code, which can be copied directly into the new topic. You can edit the zero prompt experience questions in the options shown here and reuse the predefined cards.

yml

```
kind: AdaptiveDialog
beginDialog:
  kind: OnEventActivity
  id: main
  condition:
=Global.PA__Copilot_Model_PageContext.pageContext.entityTypeName = "account"
  priority: 100
  eventName: Microsoft.PowerApps.Copilot.RequestZeroPrompt
  actions:
    - kind: SetVariable
      id: setVariable_acI6It
      variable: Topic.WelcomeMessage
      value: Welcome to the Zero Prompt Experience

    - kind: SetTextVariable
      id: setTextVariable_g4rSG0
      variable: Topic.FooterMessage
      value: This is custom ZPE footer message

    - kind: SetVariable
      id: setVariable_QUoZ5c
      variable: Topic.OverviewIcon
      value:
"=\"
g9IjAgMCAyMCAyMCIgZmlsbD0ibm9uZSIgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3
ZnIj4NCjxwYXRoIGQ9Ik0zLjUgNEMyLjY3MTU3IDQgMiA0LjY3MTU3IDIgNS41VjcuNUMyIDguMz
I4NDMgMi42NzE1NyA5IDMuNSA5SDUuNUM2LjMyODQzIDkgNyA4LjMyODQzIDcgNy41VjUuNUM3ID
QuNjcxNTcgNi4zMjg0MyA0IDUuNSA0SDMuNVpNMyA1LjVDMyA1LjIyMzg2IDMuMjIzODYgNSAzLj
UgNUg1LjVDNS43NzYxNCA1IDYgNS4yMjM4NiA2IDUuNVY3LjVDNiA3Ljc3NjE0IDUuNzc2MTQgOC
A1LjUgOEgzLjVDMy4yMjM4NiA4IDMgNy43NzYxNCA3IDcuNVY1LjVaTTkuNSA5SDcuNUM3LjIyMz
g2IDkgNyA5LjIyMzg2IDcgOS41VjExLjVDNyAxMS43NzYxNCA3LjIyMzg2IDEyIDcuNSAxMkg5Lj
VDOS43NzYxNCAxMiAxMCAxMS43NzYxNCAxMCAxMS41VjkuNUMxMCA5LjIyMzg2IDkuNzc2MTQgOS
A5LjUgOVpNMTAgMTIuNUMxMCAxMi4yMjM4NiAxMC4yMjM4NiAxMiAxMC41IDEySDEyLjVDMTIuNz
c2MTQgMTIgMTMgMTEuNzc2MTQgMTMgMTEuNVYxMC41QzEzIDEwLjIyMzg2IDEyLjc3NjE0IDEwID
EyLjUgMTBIMTAuNUMxMC4yMjM4NiAxMCAxMCAxMC4yMjM4NiAxMCAxMC41VjEyLjVaTTEwLjUgNk
gxMi41QzEyLjc3NjE0IDYgMTMgNS43NzYxNCAxMyA1LjVWMy41QzEzIDMuMjIzODYgMTIuNzc2MT
QgMyAxMi41IDNIMTAuNUMxMC4yMjM4NiAzIDEwIDMuMjIzODYgMTAgMy41VjUuNUMxMCA1Ljc3Nj
E0IDEwLjIyMzg2IDYgMTAuNSA2WiIgZmlsbD0iIzcwNzA3MCIvPg0KPC9zdmc+\""

    - kind: SetVariable
```

```
      id: setVariable_zeAjbE
      variable: Topic.ClockIcon
      value:
"=\"
g9IjAgMCAyMCAyMCIgZmlsbD0ibm9uZSIgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3
ZnIj4NCjxwYXRoIGQ9Ik0xMCA2LjUwMDEyQzEwIDYuMjIzOTggOS43NzYxNCA2LjAwMDEyIDkuNS
A2LjAwMDEyQzkuMjIzODYgNi4wMDAxMiA5IDYuMjIzOTggOSA2LjUwMDEyVjEwLjUwMDFDOSAxMC
43NzYzIDkuMjIzODYgMTEuMDAwMSA5LjUgMTEuMDAwMUgxMi41QzEyLjc3NjEgMTEuMDAwMSAxMy
AxMC43NzYzIDEzIDEwLjUwMDFDMTMgMTAuMjI0IDEyLjc3NjEgMTAuMDAwMSAxMi41IDEwLjAwMD
FIMTBWNi41MDAxMlpNMy4zNTI3MSA3Ljc5OTk4QzIuNTM0ODEgNy4yMjM4MSAyIDYuMjY3ODIgMi
A1LjE4NzA1QzIgMy40MzA2NyAzLjQxMzUzIDIgNS4xNjU2MiAyQzYuMjQyNzQgMiA3LjE5MjE4ID
IuNTQxNTEgNy43NjMxNCAzLjM2NTAxQzguNDY1NTMgMy4xMjgzIDkuMjE3NzggMyAxMCAzQzEwLj
c3OTQgMyAxMS41MjkgMy4xMjczNyAxMi4yMjkyIDMuMzYyNDNDMTIuODAxIDIuNTM5OTcgMTMuNz
UyNiAyIDE0LjQzMTEgMkMxNi41MjE3LjQ5OSAzLjQ0OTI2IDE0LjI1NTYgMTEuNzUzMSAxNyAxMC
4xNjk2IDE3IDYuNjQ0NDQgMTYuMzU1NiA1LjQxNjM4IDE1LjI5NTYgMTEuNzUzMSAxNyAxMCAxN0M4Lj
I0Njk2IDE3IDYuNjQ0NDQgMTYuMzU1NiA1LjQxNjM4IDE1LjI5MDdMMy44NTM1NiAxNi44NTM1NiAxNi44NTM2Qz
MuNjU4MyAxNy4wNDg5IDMuMzQxMzQgMTcuMDQ4OSAzLjE0NjQ1IDE2Ljg1NTQgMTcuMTQ2NCAxNi
44NTM0TDE0LjU4MzcgMTUuMjkwN0MxMy4zNTU2IDE2LjM1NTYgMTEuNzUzMSAxNyAxMCAxN0M4Lj
I0Njk2IDE3IDYuNjQ0NDQgMTYuMzU1NiA1LjQxNjM4IDE1LjI5MDdMMy44NTM1NiAxNi44NTM2Qz
MuNjU4MyAxNy4wNDg5IDMuMzQxMzQgMTcuMDQ4OSAzLjE0NjQ1IDE2Ljg1NTQgMTcuMTQ2NCAxNi
44NTM0TDE0LjU4MzcgMTUuMjkwN0MxMy4zNTU2IDE2LjM1NTYgMTEuNzUzMSAxNyAxMCAxN0M4Lj
UuMTg3MDVaTTE2LjI0MTY3MTY4NEMxNi43NjE4MSAxNi4zMjc5NiAxNi4zMjc5NiAxNS44OTE2Nj
E2LjQzMTEgMTUuNTI5MjQgMTUuMjkwN0MxMy4zNTU2IDE2LjE5NjQgMTEuNzUzMSAxNyAxMCAxN0
E2QzEzLjMxMzcgMTYuMTY4NTY4NEMxNi43MjMyMDYyIDE0LjI1NTYgMTEuNzUzMSAxNyAxMCAxN0Qz
YuNjg2MjkgNCA0IDYuNjg2MjkgNCAxMFoiIGZpbGw9IiM3MDcwNzAiLz4NCjwvc3ZnPg==\""

    - kind: SetVariable
      id: setVariable_i1e9qL
      variable: Topic.ExclamationIcon
      value:
"=\"
g9IjAgMCAyMCAyMCIgZmlsbD0ibm9uZSIgeG1sbnM9Imh0dHA6Ly93d3cudzMub3JnLzIwMDAvc3
ZnIj4NCjxwYXRoIGQ9Ik0xMCAyQzE0LjQxOTMgMiAxOCA1LjU4MTcyIDE4IDEwQzE4IDE0LjQxOD
MgMTQuNDE4MyAxOCAxMCAxOEM1LjU4MTcyIDE4IDIgMTQuNDE4MyAyIDEwQzIgNS41ODE3MiA1Lj
U4MTcyIDIgMTAgMlpNNTAgM0M2LjEzNDAxIDMgMi4xEzNDAxIDMgMTBDMyAxNy44NjYgNi4xMzMz
QwMSAxNyAxMCAxN0MxMy44NjYgMTcgMTcuMTMzMzQgMTcgMTBDMTcgMi4xMzMzQwMSAxMyAzIDEwID
MgMTAgM1pNMTAgMTIuNUMxMC40MTQyIDEyLjUgMTAuNzUgMTAuNTUgMTAuNzUgMTAuMTI1QzEwLj
c1IDkuNzAwMjUgMTAuNDE0MiA5LjUgMTAgOS41QzkuNTg1NzUgOS41IDkuMjUgOS43MDAyNSA5LjI1
IDEwLjEyNUM5LjI1IDEwLjU1IDkuNTg1NzUgMTIuNSAxMCAxMi41WiIgZmlsbD0iIzcwNzA3MCIv
PjwvZz4NCjwvc3ZnPg==\""

    - kind: SetVariable
      id: setVariable_x6BNry
      variable: Topic.OptionOne
      value: "={skillType: \"PromptTextSkill\", optionGroup: \"Redmond
```

```
Accounts!\", prompt: \"What are the accounts located in Redmond?\" }"

    - kind: SetVariable
      id: setVariable_4jjXcz
      variable: Topic.OptionTwo
      value: "={skillType: \"MCSMessageSkill\", optionGroup: \"Missing Phone
Numbers\", prompt: \"Which accounts have missing phone numbers?\" }"

    - kind: SetVariable
      id: setVariable_DzjFpB
      variable: Topic.OptionThree
      value: "={skillType: \"MCSMessageSkill\", optionGroup: \"Active
Accounts\", prompt: \"How many active accounts are there?\" }"

    - kind: SetVariable
      id: setVariable_q8Ne5M
      variable: Global.PA_Copilot_ZeroPrompt
      value: |-
        ={
                            type: "AdaptiveCard",
                            body: [
                                {
                                  type: "ColumnSet",
                                  id: "ms-platform-zpe-columnsetheader-
dc291457-0726-4472-9264-ab644fe9b13e",
                                  columns: [
                                      {
                                          type: "Column",
                                          items: [
                                          {
                                              type: "TextBlock",
                                              wrap: true,
                                              text: "Hi
"&System.User.FirstName&","
                                          },
                                          {
                                              type: "TextBlock",
                                              wrap: true,
                                              text:
""&Topic.WelcomeMessage&"",
                                              id: "ms-platform-zpe-header-
74453ffe-b4e3-4c81-9a6c-5c03dc4a3661",
                                              spacing: "Medium"
                                          },
                                          ]
                                      }
                                  ]
                                },
                                {
                                    type: "Container",
                                    id: "ms-platform-zpe-actionscontainer-
d44c003a-b977-4e0b-a3fc-f8d30d60110b",
                                    isVisible:
""&!IsBlank(Topic.OptionOne.prompt)&"",
                                    showBorder: true,
```

```
                                roundedCorners: true,
                                selectAction: {
                                    type: "Action.Submit",
                                    data: {
                                        scenario: "ZeroPromptCard",
                                        skillType:
Topic.OptionOne.skillType,
                                        value:
""&Topic.OptionOne.prompt&"",
                                        source: "ZeroPrompt"
                                    }
                                },
                                items: [
                                    {
                                        type: "ColumnSet",
                                        columns: [
                                            {
                                                type: "Column",
                                                width: "auto",
                                                items: [
                                                    {
                                                        type: "Image",
                                                        url:
""&Topic.ExclamationIcon&"",
                                                        altText:
""&Topic.OptionOne.optionGroup&""
                                                    }
                                                ]
                                            },
                                            {
                                                type: "Column",
                                                width: "stretch",
                                                items: [
                                                    {
                                                        type: "TextBlock",
                                                        text:
""&Topic.OptionOne.optionGroup&"",
                                                        weight: "bolder"
                                                    },
                                                    {
                                                        type: "TextBlock",
                                                        text:
""&Topic.OptionOne.prompt&""
                                                    }
                                                ]
                                            }
                                        ]
                                    }
                                ]
                            },
                            {
                                type: "Container",
                                id: "ms-platform-zpe-actionscontainer-
477da7ce-bd52-4870-a8c2-04071daa56e2",
                                isVisible:
```

```
""&!IsBlank(Topic.OptionTwo.prompt)&"",
                                    showBorder: true,
                                    roundedCorners: true,
                                    selectAction: {
                                        type: "Action.Submit",
                                        data: {
                                            scenario: "ZeroPromptCard",
                                            skillType:
Topic.OptionTwo.skillType,
                                            value:
""&Topic.OptionTwo.prompt&"",
                                            source: "ZeroPrompt"
                                        }
                                    },
                                    items: [
                                        {
                                            type: "ColumnSet",
                                            columns: [
                                                {
                                                    type: "Column",
                                                    width: "auto",
                                                    items: [
                                                        {
                                                            type: "Image",
                                                            url:
""&Topic.ClockIcon&"",
                                                            altText:
""&Topic.OptionTwo.optionGroup&""
                                                        }
                                                    ]
                                                },
                                                {
                                                    type: "Column",
                                                    width: "stretch",
                                                    items: [
                                                        {
                                                            type: "TextBlock",
                                                            text:
""&Topic.OptionTwo.optionGroup&"",
                                                            weight: "bolder"
                                                        },
                                                        {
                                                            type: "TextBlock",
                                                            text:
""&Topic.OptionTwo.prompt&""
                                                        }
                                                    ]
                                                }
                                            ]
                                        }
                                    ]
                                },
                                {
                                    type: "Container",
                                    id: "ms-platform-zpe-actionscontainer-
```

```
ff8b54c0-ba5a-435f-a66f-3978115ef39a",
                                    isVisible:
""&!IsBlank(Topic.OptionThree.prompt)&"",
                                    showBorder: true,
                                    roundedCorners: true,
                                    selectAction: {
                                        type: "Action.Submit",
                                        data: {
                                            scenario: "ZeroPromptCard",
                                            skillType:
Topic.OptionThree.skillType,
                                            value:
""&Topic.OptionThree.prompt&"",
                                            source: "ZeroPrompt"
                                        }
                                    },
                                    items: [
                                        {
                                            type: "ColumnSet",
                                            columns: [
                                                {
                                                    type: "Column",
                                                    width: "auto",
                                                    items: [
                                                        {
                                                            type: "Image",
                                                            url:
""&Topic.OverviewIcon&"",
                                                            altText:
""&Topic.OptionThree.optionGroup&""
                                                        }
                                                    ]
                                                },
                                                {
                                                    type: "Column",
                                                    width: "stretch",
                                                    items: [
                                                        {
                                                            type: "TextBlock",
                                                            text:
""&Topic.OptionThree.optionGroup&"",
                                                            weight: "bolder"
                                                        },
                                                        {
                                                            type: "TextBlock",
                                                            text:
""&Topic.OptionThree.prompt&""
                                                        }
                                                    ]
                                                }
                                            ]
                                        }
                                    ]
                                },
                                {
```

```
                                type: "Container",
                                id: "ms-platform-zpe-collapsed-footer-
79fb882f-ec4e-4abd-bc65-8c523849da40",
                                items: [
                                    {
                                        type: "ColumnSet",
                                        columns: [
                                            {
                                                type: "Column",
                                                width: "auto",
                                                items: [
                                                    {
                                                        type: "TextBlock",
                                                        wrap: true,
                                                        text:
""&Topic.FooterMessage&"",
                                                        id: "fsFooterId"
                                                    },
                                                ]
                                            }
                                        ]
                                    },
                                ]
                            },
                        ],
                        '$schema':
"http://adaptivecards.io/schemas/adaptive-card.json",
                        version: "1.5"
                    }
```

Related articles

- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Power Apps
- Enable copilots and generative AI features in Power Apps

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Excel and Word templates

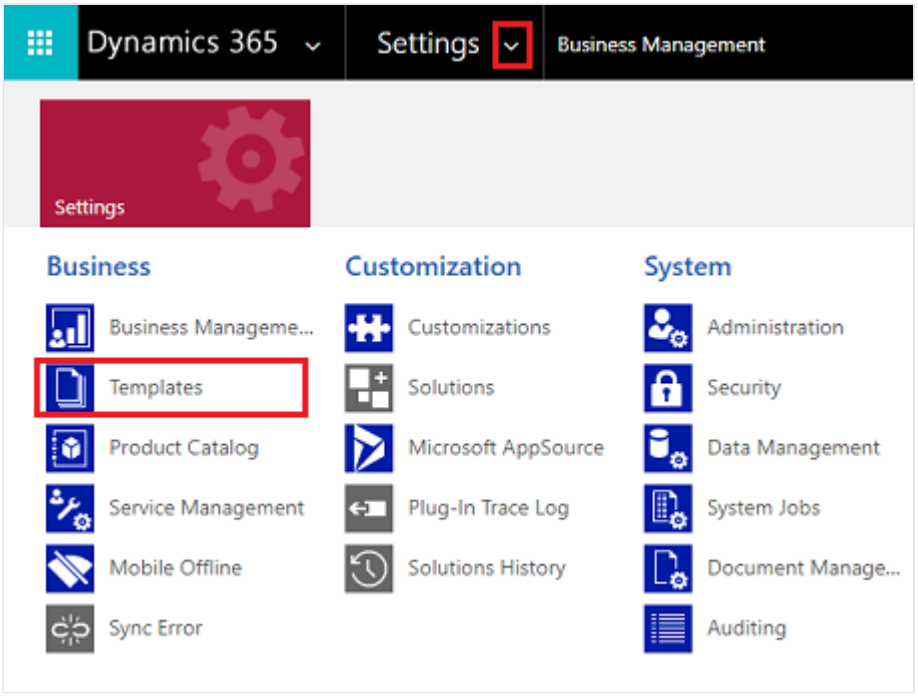
Article • 02/15/2022

You can use templates in a variety of ways to speed your work and improve consistency. With Excel templates, you can easily create and share your customized analysis with others in your organization. You can use Word templates to create standardized documents automatically populated with Microsoft Dataverse data.

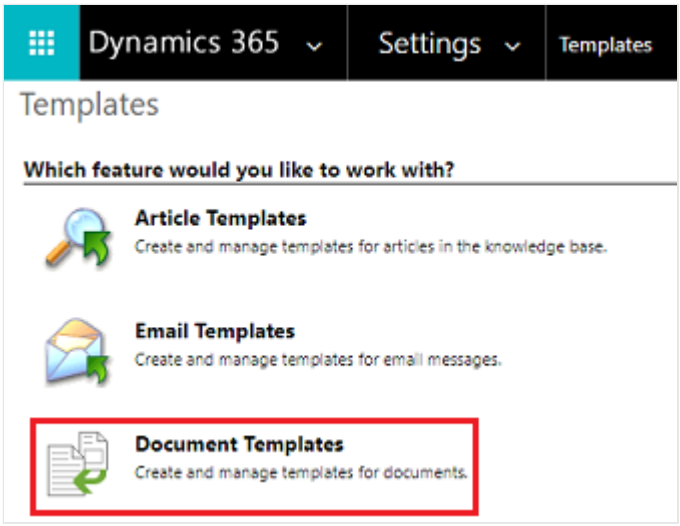
Enable and work with Excel and Word templates

1. Sign in to Power Apps and in the upper-right corner, select the  (upper right), and then select **Advanced settings**.

2. Select **Settings** > **Templates**.



3. On the **Templates** page, select **Document Template** to work with Excel or Word templates.



For information on using templates, see the following documentation:

- Word templates: Use Word templates to create standardized documents
- Excel templates: Analyze and share your data with Excel templates

Use modern themes in model-driven apps

Users of model-driven apps with the modern, refreshed look for model-driven apps enabled experience updated styling aligned to the Microsoft Fluent 2 design system. Because this modern refreshed look comes with a new theming system, classic theming isn't honored; however, makers can modify the colors used by the app to help align with their organizational branding for users who have enabled the modern, refreshed look. In this article, you learn about the styling overrides available with the modern, refreshed look and how to implement them for your organization.

> ⓘ Note
>
> - Modern themes currently support providing a custom theme for the entire app, overriding the app header colors, and setting a custom app logo. Other customizations like customizing the business process flow control aren't available.
> - For modern themes to work, the model-driven app must be using the **New look**. More information: Modern, refreshed look for model-driven apps

Modify the app theme

With the modern, refreshed look enabled in the app, makers can create a custom theme that helps align to their organizational branding. With a custom theme you can change the app header, hyperlinks, lookups, primary buttons, active tab indicators, row selection, and hover effects. The custom theme also lets you change the font that is used in the application. To accomplish this, you encapsulate the desired theme information into an XML resource, use an app setting to point to this web resource, and then verify the new modern theme matches your expectations.

> ⓘ Note
>
> Most of the UI uses your custom theme. However, some areas don’t use modern theming yet for example, legacy grids, row summaries, focus view, and the sales pipeline.

Overview of the custom theme XML resource

The first step to creating a custom modern theme is to create an XML file with your desired theme parameters, with one or more of the following attributes defined inside a `CustomTheme` tag.

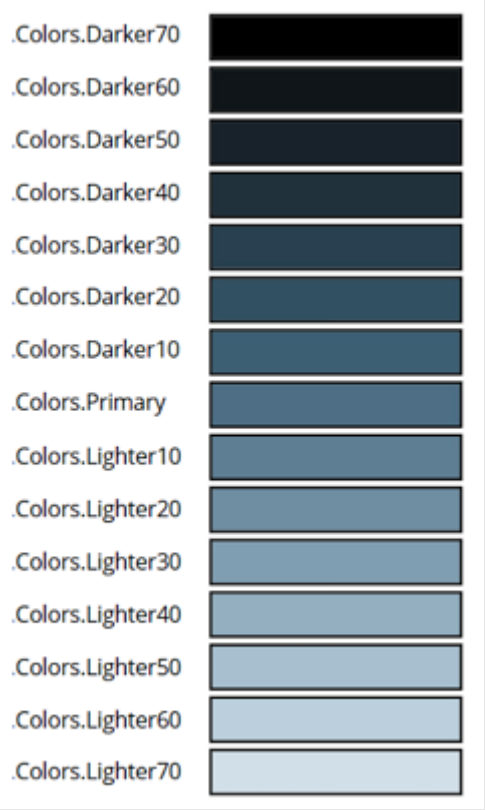
- `BasePaletteColor` – The seed color (HEX code) used as a basis to generate a 16-slot color palette for the theme.
- `LockPrimary` – A boolean that determines how the selected seed color is used to generate the 16-slot color palette.
  - False (Default): The palette is optimized for accessibility, but doesn't guarantee the seed color will appear in any slot of the generated palette. This is the default setting. Use the Fluent theme generator to preview the generated palette based on your selections for `basePaletteColor`, `vibrancy`, and `hueTorsion`.
  - True: The seed color is placed in primary (middle) slot of the palette. The remaining colors are generated by making the colors incrementally lighter on one side and darker on the other side. The generated palette might not meet contrast ratio accessibility requirements.
- `Font` – The font for your custom theme. The font that the custom theme renders depends on the browser and target machine's ability to show that font.
- `Vibrancy` – An optional parameter that impacts the muteness or brightness of the palette, especially the lighter colors. The allowed values are between -100 and 100, with a default value of 0. This option is only applicable when `lockPrimary="true"`.
- `HueTorsion` – An optional parameter that impacts the tint, shade, or tone of the palette, especially the lighter colors. The allowed values are between -100 and 100, with a default value of 0. This option is only applicable when `lockPrimary="true"`.
- `LogoWebResource` – The logical name of the image web resource to use as the logo in the app header. Logos that are too large don't display. The recommended image size is 156 pixels wide × 48 pixels high.
- `LogoTooltip` – The tooltip text displayed when a user hovers over the app logo. Defaults to *Dynamics 365* if not specified.

Override the palette

Overriding individual slots in the theme palette provides the creator with full control over the slot colors, which is particularly useful when additional modifications are required beyond the standard theme parameters mentioned above. To override a specific slot, assign a value to the desired slot by its name. For instance, you can specify the HEX code for `darker70`, `primary`, or `lighter10` to tailor the appearance precisely to your preference. Setting values for all 16 slots

completely overrides all the palette options described in the previous section, allowing for a highly customized and unique color scheme.

The slot names for the palette from darkest to lightest are: `darker70`, `darker60`, `...` `<>`, `darker10`, `primary`, `lighter10`, `lighter20`, `...`, `lighter80`. Refer to the Fluent theme designer to find out how these slots are generally used in Fluent controls.



Example XML for a custom theme

As an example, this XML specifies a custom theme that is green with a different font and a custom logo.

XML

```
<CustomTheme
    basePaletteColor="#00FF00"
    vibrancy="0"
    hueTorsion="-50"
    font="'GreatVibes', cursive"
    logoWebResource="contoso_company-logo"
    logoTooltip="Contoso"
/>
```

This XML specifies a custom theme using the alternate palette generation algorithm with an override for a single slot.

XML

```
<CustomTheme
    basePaletteColor="#00FF00"
    lockPrimary="false"
    font="'GreatVibes', cursive"

    lighter70="#FFFFFF"
/>
```

Modify the app header colors

Makers can customize the styling of the app header to deviate from the app theme. To make these customizations, extend the XML created for the custom modern theme by adding one or more of the following attributes inside an `AppHeaderColors` tag.

- `Background` – The background color of the app header. This element must be defined for any changes to take effect.
- `Foreground` – The text color of the app header. If this isn't specified, the system attempts to calculate an appropriate color that has sufficient contrast to the provided background color.
- `BackgroundHover` – The background color of buttons on the app header when they're hovered over. If no value is specified, the system calculates a color based on the background color.
- `ForegroundHover` – The text color of buttons on the app header when they're hovered over. If no value is specified, the system attempts to calculate an appropriate color that has sufficient contrast to the `backgroundHover` color.
- `BackgroundPressed` – The background color of buttons on the app header when they're pressed. The defaulting logic is the same as `backgroundHover`.
- `ForegroundPressed` – The text color of buttons on the app header when they're pressed. The defaulting logic is the same as `foregroundHover`.
- `BackgroundSelected` – The background color of buttons on the app header when they're selected. The defaulting logic is the same as `backgroundHover`.
- `ForegroundSelected` – The text color of buttons on the app header when they're selected. The defaulting logic is the same as `backgroundHover`.

Example XML for a modern theme

As an example, this XML extends the green custom theme with a black app header. For optimal usability, we recommend specifying different color values for each interaction state.

XML

```
<CustomTheme
    basePaletteColor="#00FF00"
    vibrancy="0"
    hueTorsion="-50"
    font="'GreatVibes', cursive"
>
  <AppHeaderColors
    background="#000000"
    foreground="#FFFFFF"
    backgroundHover="#313131"
    foregroundHover="#FFFFFF"
    backgroundPressed="#494949"
    foregroundPressed="#FFFFFF"
    backgroundSelected="#717171"
    foregroundSelected="#FFFFFF"
  />
</CustomTheme>
```

> **Note**
>
> These settings cause any colors specified in the **Override app header color** setting to be ignored.

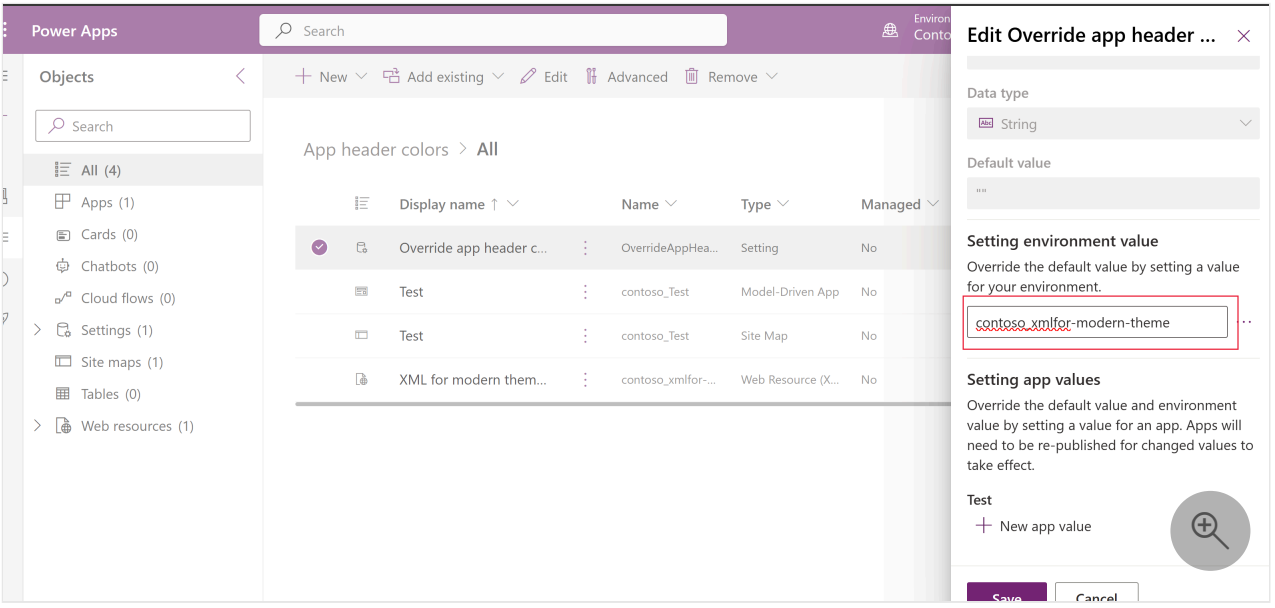
Create the web resource

1. Using a text or XML editor, save the XML that is used to create the web resource. Example XML for a modern theme
2. Sign into Power Apps.
3. Select **Solutions** on the left navigation pane, and then create a **New solution**.
4. Select **New** > **More** > **Web resource**.
5. Select **Choose file**, browse to and select the XML text file you created earlier.
6. In the **New web resource** property pane, enter the following values:
   - **Display name**: Enter a display name, such as *Green custom theme*.
   - **Name**. Accept the automatically generated or enter a unique name for the web resource.
   - Type: **Data (XML)**
7. Select **Save**. You publish this customization with the steps in the next section.

Apply the custom theme to apps in your environment

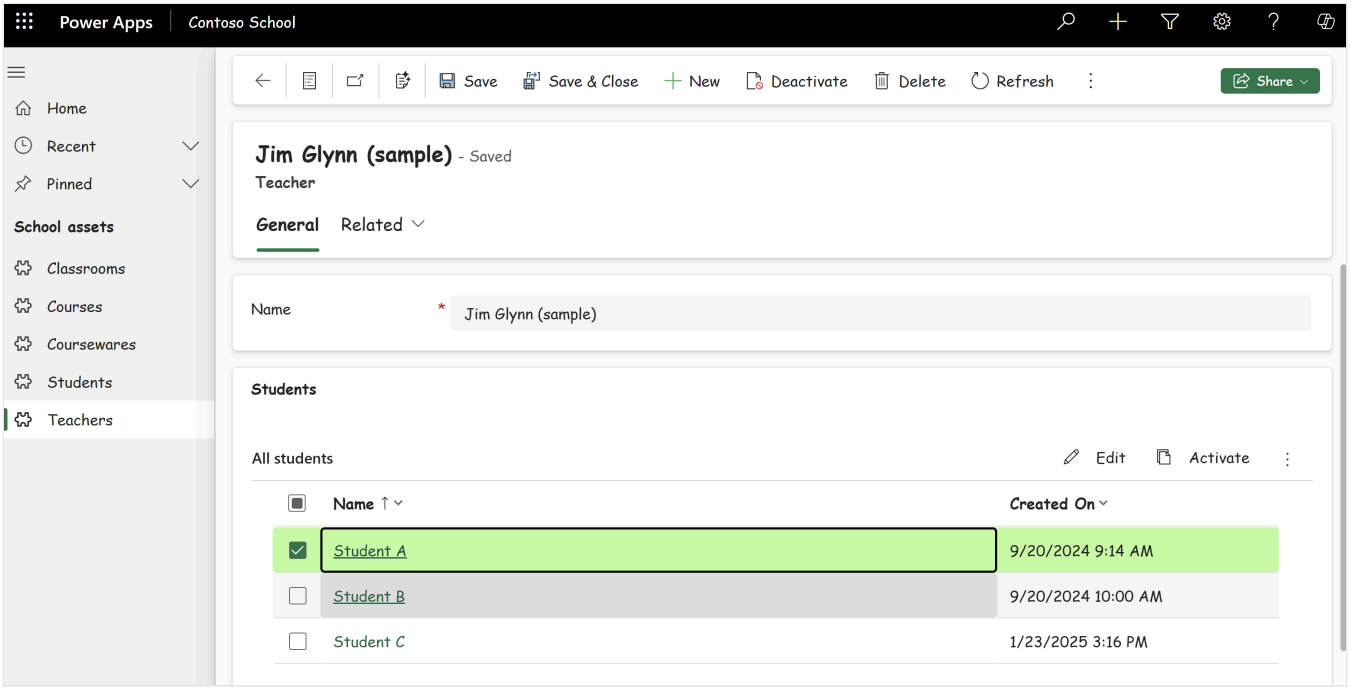
After you select your colors and create the web resource, follow these steps to enable this app header styling for all the apps in your environment that have the **New look** enabled.

1. In the solution you used to create the web resource, select **Add existing** > **More** > **Setting**.
2. Type *custom theme* in the **Search** box, select **Custom theme definition**, select **Next**, and then select **Add**.
3. In the solution, select **Custom theme definition**, and then select **Edit** on the command bar.
4. On the right **Edit Custom theme definition** properties pane, select **New environment value** under **Setting environment value** and enter the unique name of your web resource you created earlier (observe the **Name** column in the solution for the unique name). Remove the double quotes and make sure to add the publisher prefix for the web resource. For example, the name might appear as *contoso_green-custom-theme* as in this example.



5. Select **Save**.
6. Select **Settings** on the left **Objects** pane, and then select **Publish all customizations** on the command bar. (This command appears when no components in the solution are selected).

Custom model-driven app using the extended green theme sample.



Verifying new app header colors

After publishing your custom theme, validate the application of the theme in the app to ensure everything appears as you expect.

Modify the app header colors only

Makers can choose to only customize the styling of the app header to deviate from the default app theme. This can be done by following these steps:

1. Create an XML file with your various color selections, with one or more of the attributes from app header color overrides section defined inside an `AppHeaderColors` tag.
2. Create a web resource using the same process described for custom themes, but be sure to give the XML resource an appropriate descriptive display name, such as *Green app header XML*.
3. Apply custom app header colors to apps in your environment by assigning this web resource to the environment or app in the **Override app header color** setting. Follow the steps from Applying custom themes to your environment but use the **Override app header color** setting instead.

> **Note**

> The **Override app header color** setting is ignored if the **Custom theme definition** setting is defined.

4. Verify your custom app header visuals, including all button states, to ensure everything appears as you expect and there are sufficient contrast ratios for accessibility. Verify the following color choices:

    - The desired colors are shown for the app header at rest and for each button interaction state.
    - There's a minimum of a 4.5:1 contrast ratio between foreground and background colors for the rest state and each button interaction state.

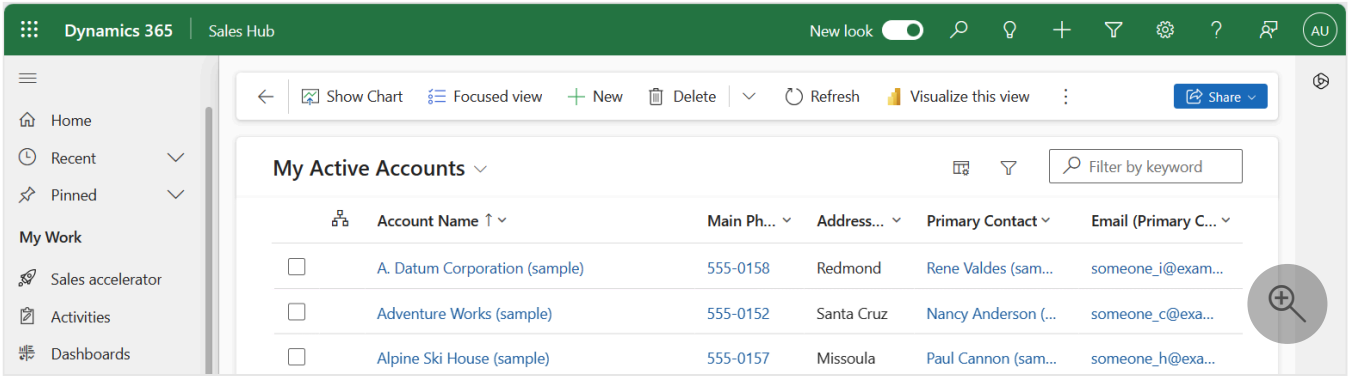
Example XML for app header color override

As an example, this XML specifies a green background color for the app header with white text, with darker background colors for the various button interaction states. For optimal usability, we recommend specifying different color values for each state.

XML

```
<AppHeaderColors
  background="#12783F"
  foreground="#FFFFFF"
  backgroundHover="#165A31"
  foregroundHover="#FFFFFF"
  backgroundPressed="#0F1C12"
  foregroundPressed="#FFFFFF"
  backgroundSelected="#153D23"
  foregroundSelected="#FFFFFF"
/>
```

When you play the app, the app header looks like the following image. You might need to refresh the browser tab to display the theme.



See also

Classic theming

Modern refreshed look

Last updated on 04/06/2026

Use a theme to create a custom look for your app

You can create a custom look and feel (a theme), for your app by making changes to the default colors and visual elements provided in the uncustomized system. For example, you can create your personal product branding by adding a company logo and providing table-specific coloring. A theme can be created without requiring a developer to write code. You can create, clone, change, or delete themes that are used in your environment.

> **Note**
>
> There's newer way to create themes in model-driven apps for customizing the model-driven app header. More information: Use modern themes
>
> You can define multiple themes, but only a single theme can be active in an environment and is identified as the default theme. To make a theme active, you publish it.

Use themes to enhance the user interface and create your product branding

Theming is used to enhance the app user interface, not drastically alter it. The theme colors are applied globally throughout your model-driven apps. For example, you can enhance the following visual elements in the UI:

- Change navigation colors to create product branding.
- Adjust accent colors, such as hover or selection colors.
- Provide table-specific coloring.
- Logo. (Use an existing or add a new image file as a web resource.)
- Logo tooltip.
- Navigation bar color.
- Title text color.

- Selected link color.
- Hover link color.
- Legacy accent color (primary background for process controls).
- Default color for tables.
- Default custom tables color.
- Control fill color.
- Control border color.

Solution awareness

Themes aren't solution aware. The changes made for an organization's theme aren't included in solutions exported from the organization. The data is stored in the theme table that can be exported and reimported in another environment. The imported theme must be published to take effect.

Copy and alter the existing theme

The easiest and quickest way to create a new theme is to clone and alter an existing theme. Then save, preview, and publish it.

1. Sign in to Power Apps, select **Settings**  (upper right), and then select **Advanced settings**.
2. On the classic settings page, select **Settings** > **Customizations** > **Themes**.
3. Under **All themes**, select the theme you want to clone, such as the **CRM Default Theme**. Select **Clone** on the command bar.
4. Open the cloned theme, and then replace an existing UI item's hexadecimal number, such as the **Title Text Color**, with the hexadecimal value that represents the color you want.

For example, the **CRM Default Theme** was cloned and changed using mostly varying shades of green color. The following screenshots show the new colors for navigation and highlighting. A custom logo was also added that appears in the upper left corner of an app.

THEME

Gentle Green Theme

Theme Name

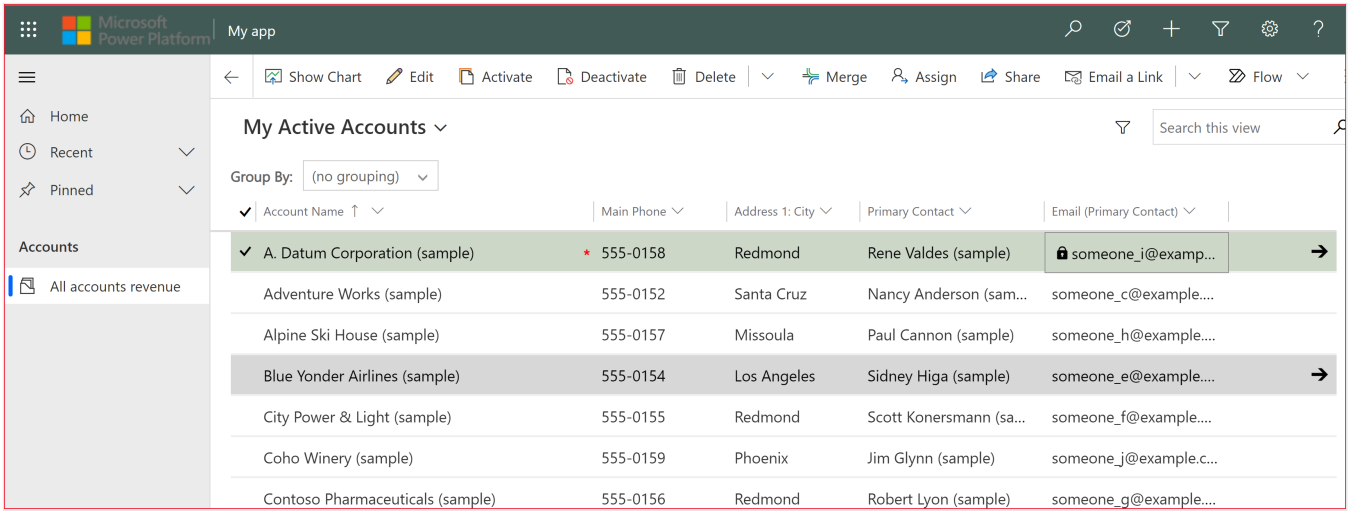
Theme Name *	Gentle Green Theme

Navigation Bar

Logo	new_defaultlogo	
Logo Tooltip	MS Green	
Navigation Bar Fill Color	#415C55	
Navigation Bar Shelf Fill Color	#79AB9E	
Title Text Color	#358717	
Main Color	#65825C	
Accent Color	#A4D194	

UI Elements	
Link and Button Text Color	#415C55
Selected Link Color	#65825C
Hover Link Color	#A4D194
Legacy Accent Color	#358717
Default Entity Color	#666666
Default Custom Entity Color	#00CCA3
Control Hover Fill Color	#FFFFFF
Control Hover Border Color	#BDC3C7
Page Header Fill Color	#E0E0E0
Panel Header Fill Color	#F3F3F3

The following image shows an app account view with the new colors and logo.



Important

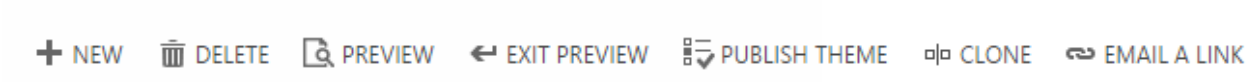
Logos that are too large don't display. The image used in the example is 156 pixels wide x 48 pixels height.

Preview and publish a theme

To preview and publish a theme, do the following steps:

1. Create a **New** theme from scratch or **Clone** an existing one.
2. **Preview** the new theme. To exit the preview mode, choose **Exit Preview** on the command bar.
3. Publish the theme. Select **Publish Theme** on the command bar to make it the active (default) theme in the environment.

The following screenshot shows the buttons on the command bar for preview and publishing.



Best practices

Following are the recommendations for designing theme contrasts and choosing colors.

Theme contrast

We recommend the following approach to providing contrast colors:

- Carefully choose the contrasting colors. The Microsoft Dataverse out-of-the-box default theme has the correct contrast ratios to ensure optimal usability. Use similar contrast ratios for your new themes.
- For high contrast mode, use the default color settings.

Theme colors

We recommend that you don't use a large number of different colors. Although you can set a different color for every table, we recommend one of two patterns:

- Make all tables in neutral colors and highlight the key tables.
- Use the same color for similar tables or related tables, such as queue and queue item, or product catalog tables. Keep the total number of groups low.

Custom theme considerations

You should consider the following when planning on using custom themes:

- Most updated user interface (UI) areas will be displayed in the custom theme colors.
- Even though the theme colors are applied globally throughout the application, some legacy UI areas, such as gradient buttons, retain the default colors.
- Certain areas must use dark or light colors to contrast with the default icon colors. The icon color isn't customizable.
- A table can't be displayed in different colors under different Sitemap nodes.
- The Sitemap nodes colors aren't customizable.

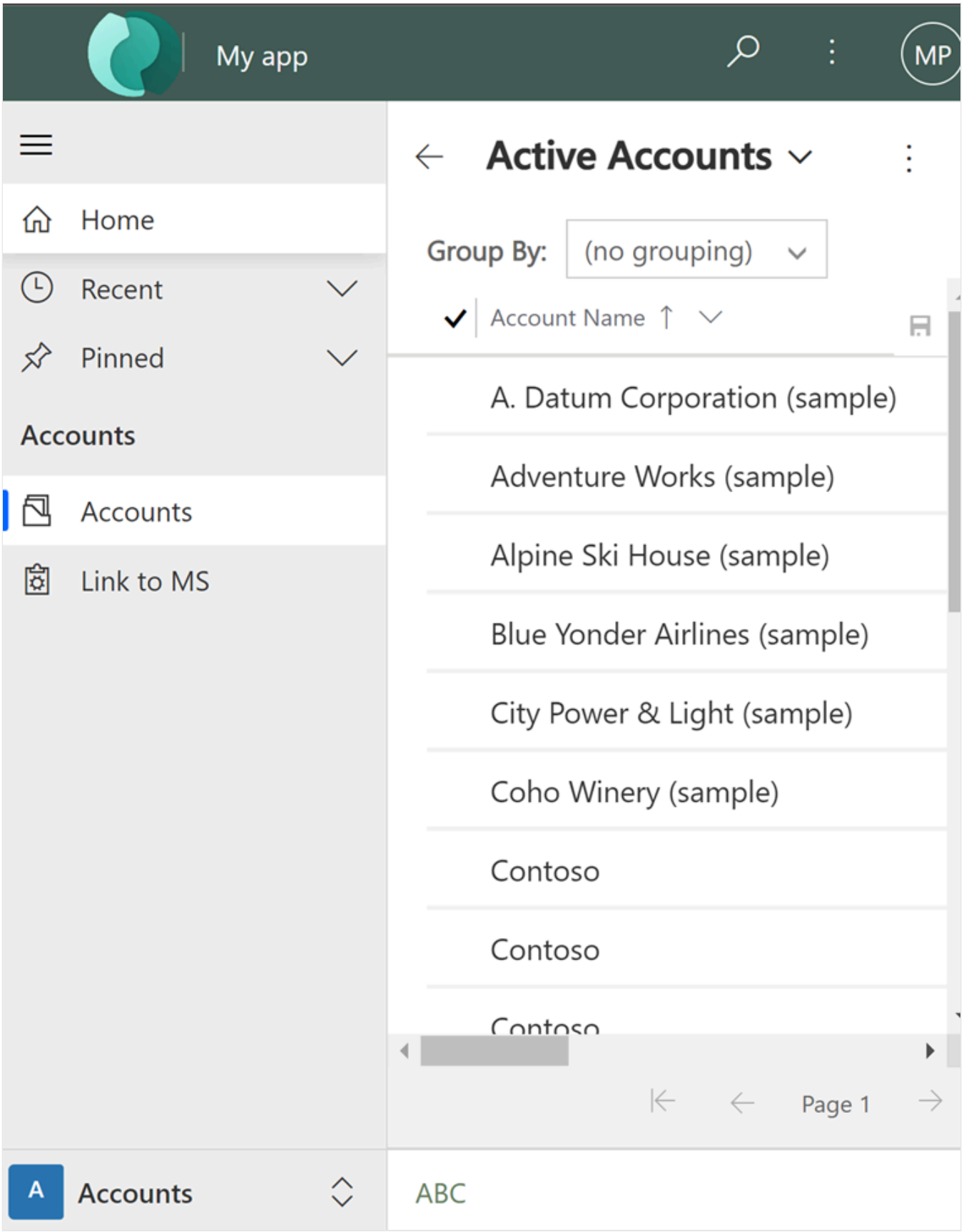
Known issues with mobile devices and themes

The custom theme isn't applied on devices running the Power Apps mobile app

When you run the Power Apps mobile app with an environment that is using a custom theme as the default theme, the mobile device displays the system CRM Default Theme.

Card forms aren't displayed in table views in a web browser

On a mobile device running the app in a web browser, the view is displayed in the standard view, which isn't optimized for small screens and is displayed similar to the image here. For more information about card forms, see Create a card form.



See also

Query and edit an organization theme

Last updated on 04/03/2026

Create guided help for your model-driven app

06/02/2025

Use custom help panes and guided tasks to give your Unified Interface application a custom in-product help experience that is tailored to your organization. Use custom help panes to provide table, form, and language-specific help and guidance that includes rich text, content links, images, and video links.

> **Important**
>
> - Custom help panes replace the previous learning path guided help feature used with legacy web client apps.
> - Custom help panes don't work with Unified Interface apps running on a mobile device, such as Android and iOS tablets and smartphones.

Custom help panes and learning path

The new guided help implementation of custom help panes differs from the previous learning path guided help feature. Both features let you create custom help for your application. However, custom help panes are optimized for the most common guided help scenarios.

Custom help panes provide the following key features that are not available with learning path:

- Free-form rich text, including bullets and numbering.
- Visibly linked coach marks and help balloons.
- More options for video sources, including private sources.
- Storage of help content in Microsoft Dataverse as part of your solution.

Custom help panes don't provide the following key features that are available with learning path:

- Sequential help balloons.
- Help pages per role.
- Help pages for per-device form factor, such as smartphones.

Prerequisites

To use the custom help panes feature, your environment needs the custom help panes environment setting enabled.

Enable custom help panes for your environment

1. Sign in to the Power Platform admin center at https://admin.powerplatform.microsoft.com.
2. In **Settings** for the selected environment, select **Product** > **Features**.
3. Under **Help features** turn **Enable Custom Help Panes** to **On**.

> **Important**
>
> You can enable custom help panes or customizable help, but not both at the same time. Confirm that **Use help for customizable entities** is set to **No**.

Privileges required

- To author help panes, app makers need global create, read, write, delete, append, and append to privileges on the **Help Page** table. By default, System Administrator and System Customizer security roles both have this privilege.
- To view help panes in an app, users need read privilege on the **Help Page** table. By default, the Help Page Consumer security role has this privilege. By default, the Basic User security role doesn't have any privileges on the Help Page table.

More information: Security roles and privileges

Context-sensitive custom help

Each help pane is unique for these contexts:

- Table
- Form
- Dashboard
- Language
- Application

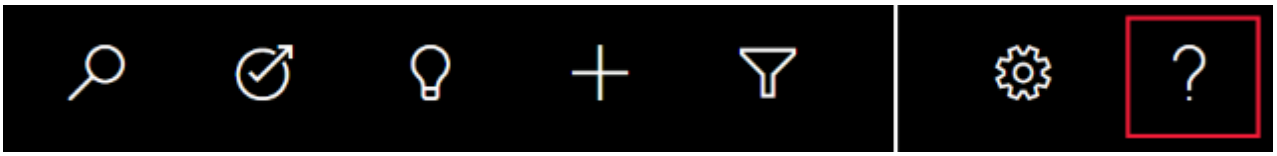
Help pane navigation

By default, a help pane stays open and on the help content you first opened it with even when you navigate to a different form. This allows for the help content to remain intact as you direct

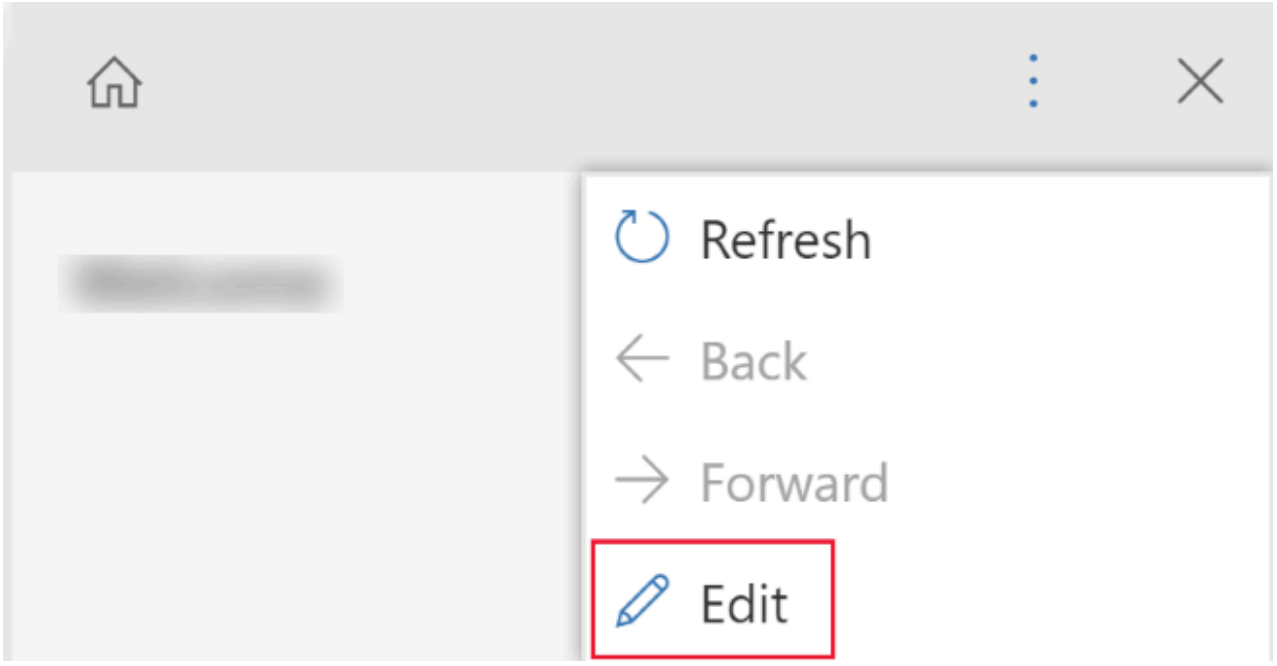
users to different parts of the app.

To author help pane content

1. To view the help pane, open a model-driven app, and then on the command bar select **Help**.



2. On the Help pane, select the vertical ellipsis, and then select **Edit**.



   The help pane is now in edit mode and the cursor is positioned on the help pane title.

3. From the edit pane you can perform the following tasks:

   - Enter text by typing directly in the help pane area.
   - Format text through rich text capabilities and add bullet/numbered lists, sections, videos, images, links, coach marks, and balloon help.

4. To save your changes, select **Save**.

Free-form text

Text can be placed anywhere within the help pane. Enter free-form text before, in, or after sections. Text supports bold, italic, underline, and strikeout font formats. Cut, copy, and paste can be used as well as multilevel undo.

Bullets and numbered lists

Selecting the bullet or number icon toggles the current line to become bulleted or numbered. If you have selected multiple lines in a list, each line becomes bulleted or numbered. Tabbing and indenting subnumbers align within the list.

Sections

A section is a collapsible text box. You can put links or free-form text in it. Use a section to group similar items. A section can be either open or collapsed by default.

Video and static images

You can insert videos and static images into your help pane. Videos and images are links to content on the internet. Custom help panes do not store the video and image files in your help pane. When the help pane is opened, custom help panes bring the content in from the link to display it. You can use a link to a Microsoft Stream video if you want to reference corporate private content.

> **Tip**
>
> Remember to copy the link URL for the video or image you want so you can paste it into your help pane.

Custom help panes support the following video sources:

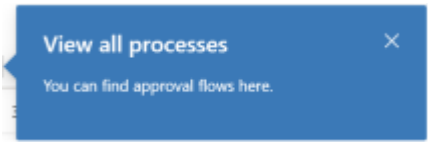
- Microsoft Stream (Classic) (use for private content)
- YouTube
- Facebook
- Vimeo

Links

Links can be to websites and open in the same window (the default) or open in a separate window. The ability to link to an existing help page is not yet enabled.

Balloons and coach marks

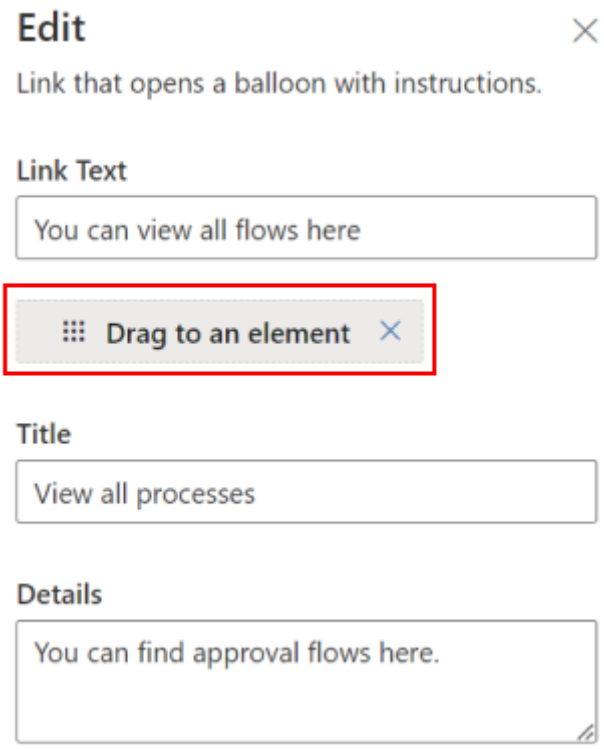
Balloons and coach marks can be used to point to specific UI elements. A balloon can have text in it.



A coach mark simply highlights an element with a coach pointer to draw the user's attention to that UI element.



Author needs to define the target through drag and drop capability in the properties of balloons and coach marks. Author can input the text for title and description for coach marks.



> **Note**
>
> - Balloons and coach marks don't work well with custom components. This feature only applies to default Unified Interface UI components.
> - Sequential help balloons and coachmarks are not supported yet.

Solutions and custom help pane content

All help content is stored in a help page component in Dataverse as part of your solution. When you move your solution from one environment to another, such as from test to

production, you can define that your help rows are exported so that they are included in the solution. This enables you to keep your help content in sync with features in your solution as it moves to different environments. As a part of your solution, custom help panes support all standard solution application lifecycle management (ALM) features.

Moving content via solutions

By default, all new help pages appear in the default solution. If you want to move your content to another environment, first add your existing help pages into an unmanaged solution before you export them. To add a help page to an unmanaged solution, follow these steps:

1. Sign into Power Apps.
2. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the unmanaged solution you want.
4. Select **Add Existing** on the command bar, and then select **Help Page**.
5. Select the help pages that you want to add, and then select **Add**. More information: Export solutions

Help page documentation automation

You might want to back up or store your content in a source code control system. You might also like to use documentation automation tools, such as translation tools or checkers, on help pane content. The custom help pane data is stored directly in Dataverse and can be exported and imported for this purpose.

Custom help panes support a custom XML format. This format is documented below. More information: Custom help XML definition

When exported, each help page is exported as a separate file.

About the Contextual Help managed solution

The custom help panes and guided tasks feature consists of the solution components in the Contextual Help managed solution. Customizing or creating records using the tables outside of the custom help panes and guided tasks feature isn't supported.

Notice that some apps, such as Dynamics 365 Sales and Dynamics 365 Service, include first run experiences that are built by using tables included in the Contextual Help managed solution. For example, both the Help Page (msdyn_helppage) and Tour (msdyn_tour) tables are used in

the creation of these first run experiences. Currently, the custom help panes and guided tasks feature doesn't support creating these first run experiences.

Frequently asked questions

This section discusses frequently asked questions about custom help pages.

Are custom help pages the same as customizable help?

Custom help panes and guided tasks are an option in the **Set custom help URL** section of system settings. Custom help panes and guided tasks enable a customizable help pane that shows up right next to the user's form. The other options in this system settings set custom help section comprise the customizable help features. They allow you to override the default apps help and point users in your organization to a different URL for help. Alternatively, you can override the help for a highly customized table so that you can better describe your workflow.

For more information about customizable help, see Enable and use customizable help.

How do I migrate my data from learning path to custom help pages?

Learning path has two types of help: help panes and sequential help balloons. The sequential help balloon locations are deeply integrated with the legacy web client UI and are not transferrable to the new custom help panes.

Depending on how much text you have in your guided help it might be easiest to simply copy the information directly from the learning path user interface to the new custom help pane user interface. However, you can also export your learning path help content. The simplest way to do this is to export your content using the **Learning Path** > **Content Library** > **Localize** > **Export** feature. Select the rows you want and then export them. This creates an XLIFF file for each help pane and guided task. Then, use a publicly available XLIFF editor or XLIFF to HTML converter to retrieve your content.

Custom help XML definition

This section describes the custom help XML definition.

PPHML

```
<pphml>
    <h1>FAQ</h1>
    <collapsible title="What is PPHML?">
        <p>PPHML is a domain specific language for help content. It is used to
create help content that includes elements such as images, videos, balloons, coach
marks, etc.</p>
    </collapsible>
    <collapsible title="What does PPHML stand for?">
        <p>PPHML stands for Power Platform Help Markup Language</p>
    </collapsible>
</pphml>
```

Definition and usage

The `<pphml>` element tells the help browser that this is a PPHML document.

The `<pphml>` element represents the root of a PPHML document.

The `<pphml>` element is the container for all other PPHML elements.

Title

Presents a title in a help page.

```
<h1>This will be displayed at the top of the help page</h1>
```

Definition and Usage

The **`<h1>`** element defines the title of a help page.

`<h1>` This must be the first element inside `<pphml>`.

Image

Presents an image in a help page.

```
<img src="smiley.gif" alt="Smiley face" title="Smiley face"/>
```

Definition and usage

The `<img>` element embeds an image in a help page.

Attributes

- `src`: Specifies the URL of an image. This attribute is required.
- `title`: Specifies a title to show along with the image, typically as a hover tooltip.
- `alt`: Specifies an alternate text for an image. This text is used by screen readers.

Video

Presents a video in a help page.

```
<video src="https://www.youtube.com/watch?v=WSWmn7WM3i4" />
```

Definition and usage

The `<video>` element embeds a video, such as a tutorial or training movie, in a help page.

Supported sources

- Microsoft Stream
- YouTube
- Facebook
- Vimeo

Attributes

- `src`: Specifies the URL of the video. This attribute is required.
- `allowFullScreen`: Specifies whether the user can switch the video to full screen. Possible values are "true" or "false". This attribute is not supported for all video sources.
- `autoplay`: Specifies that the video will start playing as soon as the help page loads. Possible values are "true" or "false". This attribute is not supported for all video sources.
- `startTime`: Specifies, in seconds, from which point to start playing the video.

Paragraph

Presents a paragraph in a help page.

```
<p>This is some text in a paragraph.</p>
```

Definition and usage

The `<p>` element defines a paragraph.

Text inside a paragraph can be decorated in the following ways:

- Bold, with `<strong>` element
- Italic, with `<em>` element
- Strikethrough, with `<del>` element
- Underline, with `<u>` element

These decorations can be combined. For example, make a fragment of text that is both bold and underlined.

Bulleted list

Presents a bulleted list in a help page.

```
<ul>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
</ul>
```

Definition and usage

The `<ul>` element defines a bulleted list.

Use the `<ul>` element together with the `<li>` element to create bulleted lists.

Numbered list

Presents a numbered list in a help page.

```
<ol>
    <li>First step</li>
    <li>Second step</li>
    <li>Third step</li>
</ol>
```

Definition and usage

The **`<ol>`** element defines an ordered (numbered) list. Use the `<ol>` tag together with the `<li>` element to create numbered lists.

Collapsible

Presents a collapsible section in a help page.

```
<collapsible title="This is a Section">
    <p>This is a paragraph inside a section</p>
    <img src=smiley.gif" title="This is an image inside a section" />
</collapsible>
```

Definition and usage

The **`<collapsible>`** element defines a section of content that the user can view or hide on demand.

Attributes

- `collapsed`: Specifies whether the section is collapsed or expanded initially. Possible values are "true" (collapsed) or "false" (expanded).

Link

Presents a link in a help page.

Link to a website that opens in a new browser window:

```
<a href="https://www.microsoft.com" target="_blank">Microsoft Home Page</a>
```

Link to another help page:

```

<a href="./LearnMore">Learn More</a>
```

Definition and usage

The <a> tag defines a link, which allows the user to navigate from a help page to a website, or to another help page.

Attributes

- `href`: Specifies the URL of the website or help page to which to navigate. This attribute is required.
- `target`: Specifies where to open the linked URL.
  - If not present or `_self`, the link is assumed to be to another help page and it's opened in the help browser.
  - If `_blank`, the link is opened in a new browser window.
  - If `_top`, the link is opened in the current browser window.
  - If the value is the name of an `iframe`, the link is opened in that iframe.

Coach mark

Presents a coach mark in a help page.

```

<coachmark target="#my-html-button">Click to highlight the HTML element with id
[my-html-button]</coachmark>
```

Definition and usage

A coach mark is an interactive element that can be used to draw the user's attention to a specific point in the UI of the application hosting the help browser.

Attributes

- `target`: CSS selector that specifies the HTML element over which the coach mark will be shown. This attribute is required.

Balloon

Presents a balloon in a help page.

```
<balloon target="#my-html-button" title="This button submits the form"
details="Please click this button to continue and submit the form">Click to show a
balloon over the HTML element with id [my-html-button]</balloon>
```

Definition and usage

A balloon is an interactive element that can be used to help the user perform an action in the UI of the application hosting the help browser.

Attributes

- `target`: CSS selector that specifies the HTML element over which the balloon link will appear. This attribute is required.
- `title`: Specifies the title of the balloon.
- `details`: Specifies the content to show inside the balloon.

Validate and publish a model-driven app using the app designer

Summarize this article for me

Model-driven apps can't be published when they don't include all the required components. Some components rely on others and this relationship between components is known as a dependency.

> **Note**
>
> This article discusses a feature that's only available or needed when you use the classic app designer. The modern designer validates your app as part of the app building process.

For example, the position table has been added to the site map, but is no longer showing in the app.

The process of checking for dependencies within a model-driven app is known as validation.

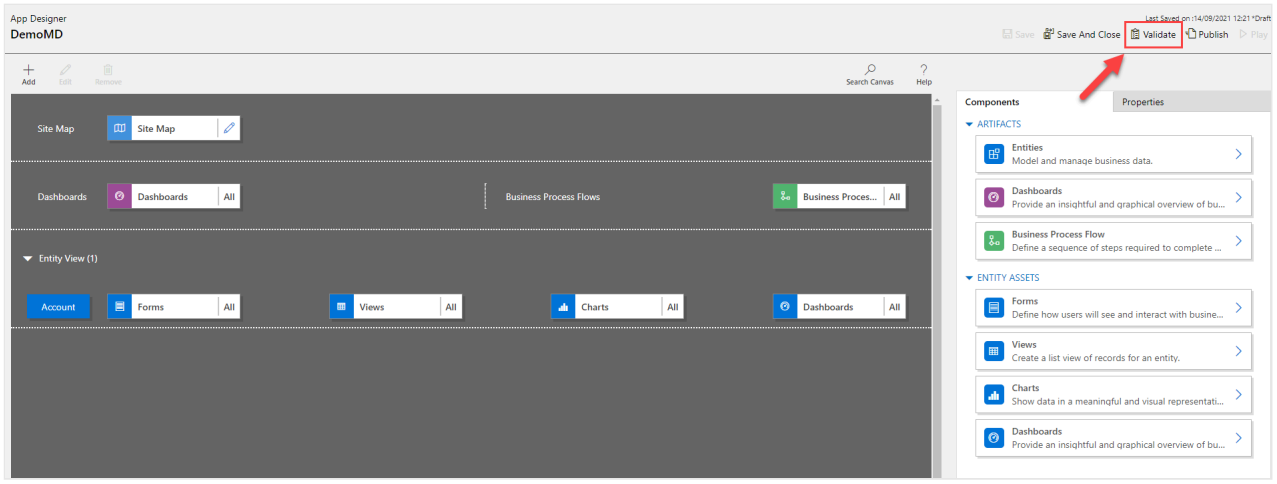
When the app is validated, the app designer canvas shows details about the assets that are missing.

How to validate an app and add in dependencies

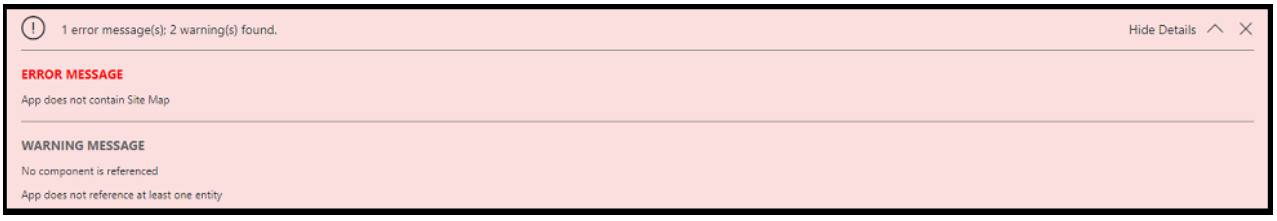
1. Sign in to Power Apps.
2. Select the environment in which the unmanaged model-driven app is located.
3. Select the model-driven app, and then on the command bar select **Edit**.

   > **Note**
   >
   > Customizations to a model-driven app should take place within a solution. To update a model-driven app within a solution, open your solution from the **Solutions** area, select the **Model-Driven App** solution component, and then on the command bar select **Edit**.

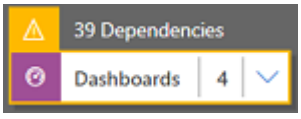
4. In the app designer, select **Validate**.



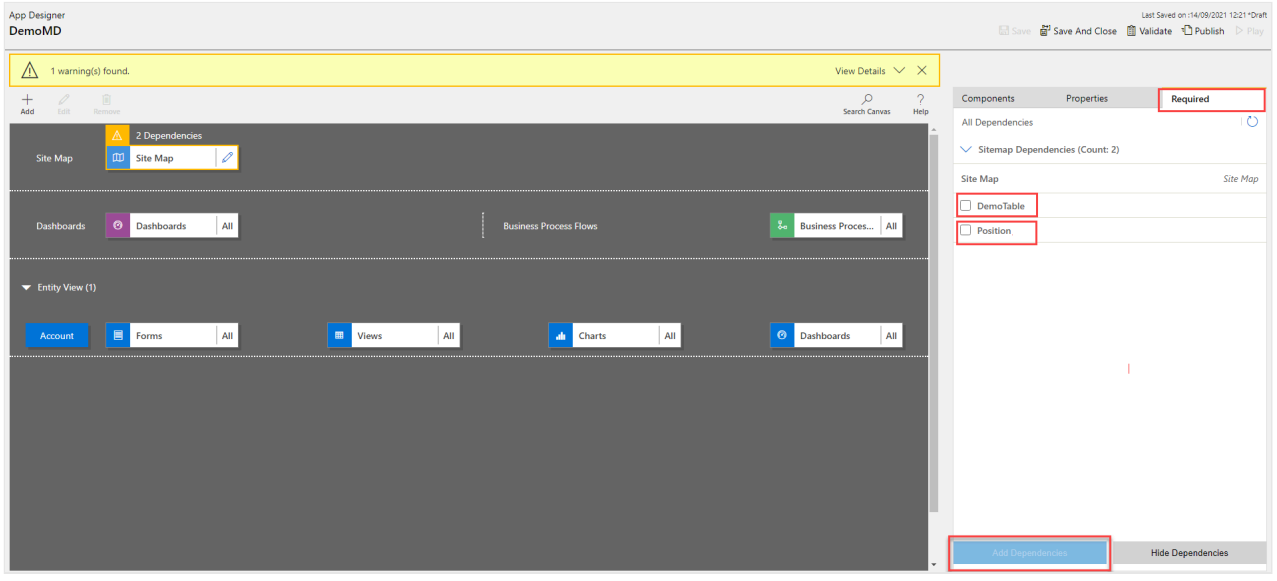
A notification bar appears and shows whether the app has any errors or warnings. The notification bar shows warnings in cases where, for example, a table has no forms or views, or the app doesn't contain any components. An error might appear if a site map isn't configured for the app. You can save and publish the app without addressing warnings, but errors must be fixed before you can publish.



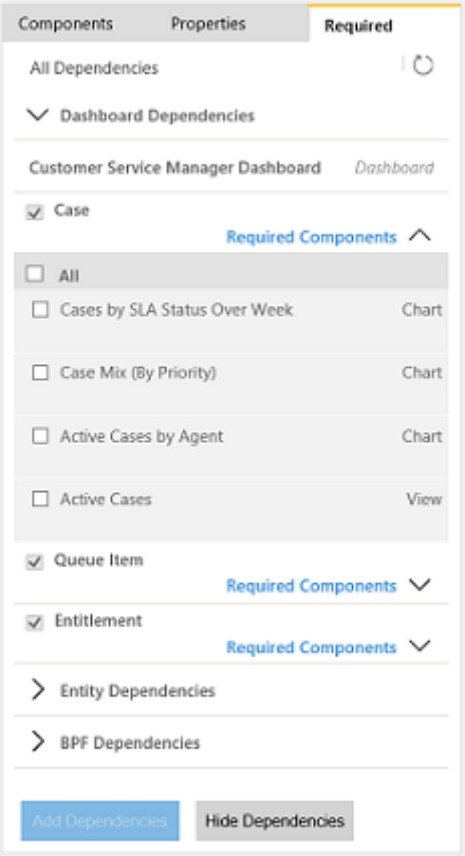
The app designer also shows a warning symbol with the number of dependencies on each artifact or asset tile that is missing a required asset.



5. To add the required assets, select the **Required** tab. The **Required** tab is visible when at least one required asset is missing from the app.



The tab displays an alternate list of required components.



6. Select the assets that are missing, and then select **Add Dependencies**. When the required asset is added, the error count for the asset decreases.

> ⓘ Note
>
> If a common asset is required across various app components, such as a form is required for a dashboard and a table, and you add that asset only once from the dashboard dependency tree, the dependency count decreases only on the dashboard tile, but not on the table tile. However, the dependency is resolved for both.
>
> Select **Get Latest Dependencies** ↻ or select **Validate** again to get the latest set of dependencies. These buttons are only visible after the app is saved.

Select **Hide Dependencies** if you don't want to add the suggested required components. Any unresolved warnings appear again when the app is opened in the app designer and select **Validate** or **Get Latest Dependencies** ↻.

> ⓘ Note

If the dependencies are hidden now and the app is exported later, all of these dependencies appear again.

Publish an app using the app designer

Publish an app to make it available to users.

After you've added components, validated and saved the app, on the command bar, select **Publish**. In the **Apps Being Edited** view, in the lower right corner of the app tile you want to publish, select the **More options** button (**...**), and then select **Publish**.

The app status changes to **Published**. This is displayed in the top-right corner of the app designer. The app moves from the **Apps Being Edited** view to the **Published Apps** view, and the published date is shown on the app tile.

> **Note**
>
> - If your app has a validation error, this is displayed in the notification bar. It isn't be possible to publish the app until the error is resolved.
> - The app can't be published until it is saved.

Next steps

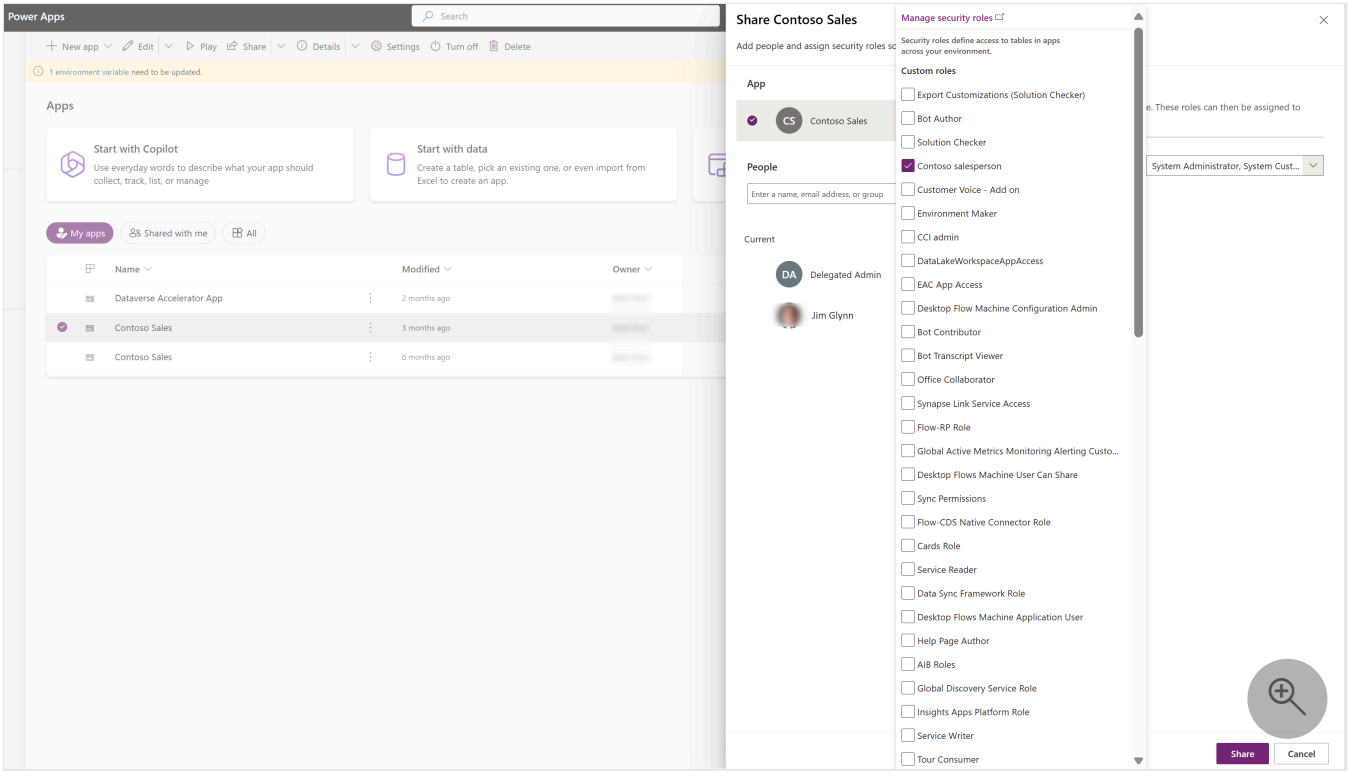
Share a model-driven app with Power Apps
Run a model-driven app on a mobile device

Last updated on 02/13/2026

Share a model-driven app

Share a model-driven app to make it available so other users can play it. Sharing includes the following steps:

1. Identify the security roles to use for the app
2. Assign security roles or people to a model-driven app
3. Share the link to the app



If your app contains only out-of-the-box tables, such as account or contact, you can use an existing predefined security role.

Model-driven app sharing basics

Model-driven apps use role-based security for sharing. The fundamental concept in role-based security is that a security role contains privileges that define a set of actions that can be performed on tables within the app. This approach means that while two people are able to use the app, one user might only be able to read records, or records that they themselves created. The other user might be able to see all records and have the rights to delete those records.

All app users must be assigned one or more predefined or custom security roles. Or, security roles can be assigned to teams. When a user or team is assigned to one of these roles, the person or team members are granted the set of privileges associated with that role.

The process for sharing a model-driven app is different from sharing a canvas app. Model-driven app sharing depends on how the Microsoft Dataverse data table privileges are assigned for the tables that are in the app. If security roles aren't already defined for your app, contact your Power Platform administrator to create them for you.

More information: Create or edit a security role to manage access

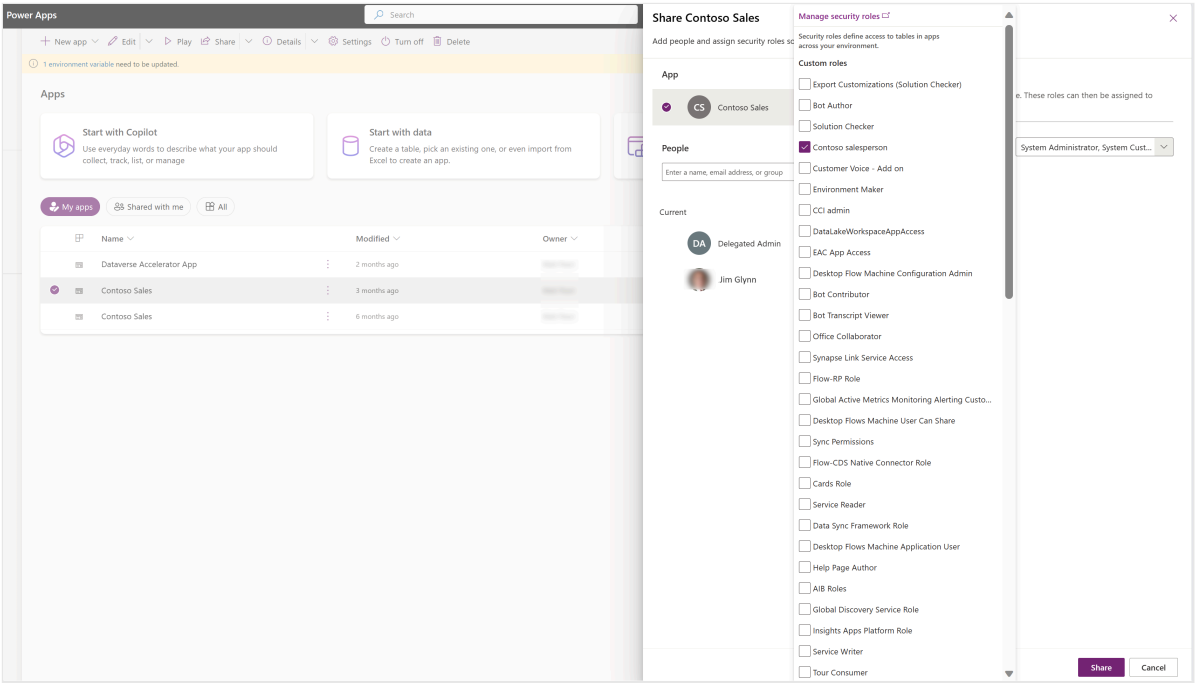
Assign security roles or people to a model-driven app

When you share a model-driven app, you can share it with all members of one or more security roles or a user or team.

1. Sign in to the Power Apps, on the left navigation pane select **Apps**, next to the app you want to share select **...**, and then select **Share**.

2. From the **Share** *app name* pane, choose from the following options:

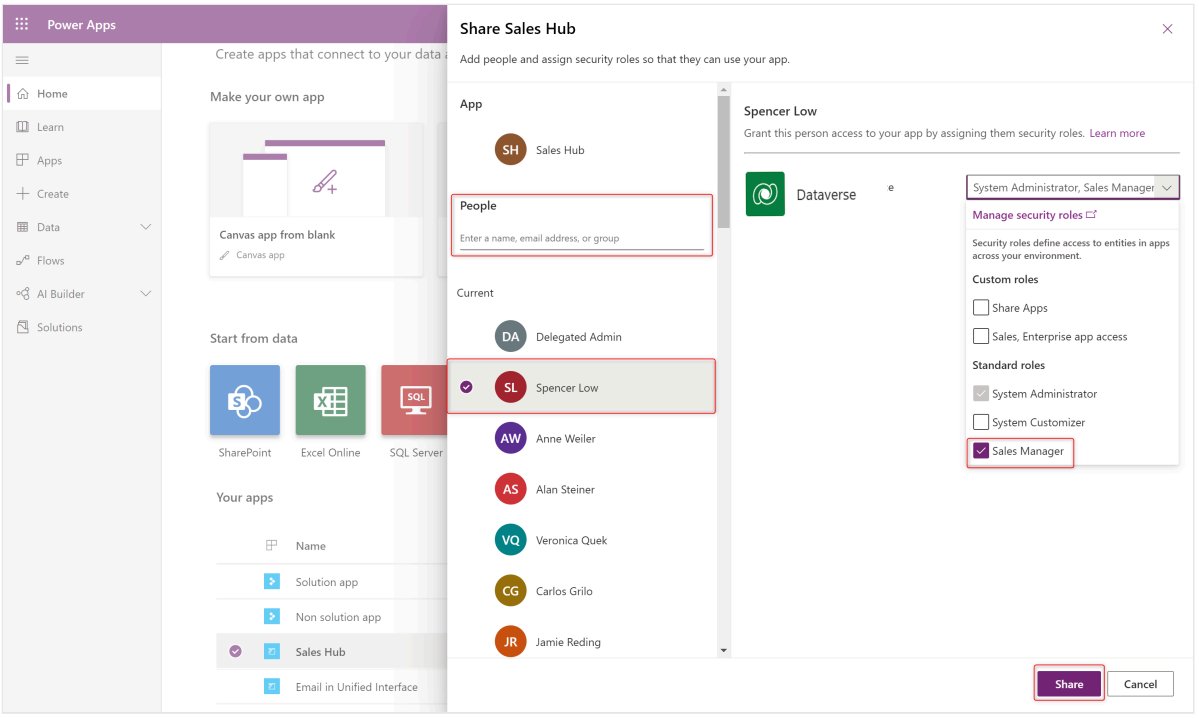
    - Select the app, and then select the drop-down list in the right pane to display all available security roles. Select the security roles you want from the security role dropdown list.



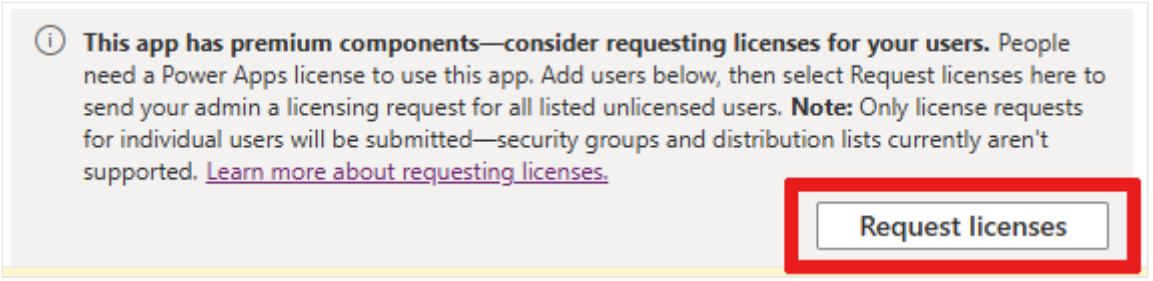
> **Important**
>
> If your app has one or more custom tables, contact a Power Platform administrator to configure privileges to the custom tables in a security role. This is necessary in

order for users to work with your custom table's records in the app. More information: [Create or edit a security role to manage access]

- To assign an individual user or team, select the user name or team from the **People** list.



If your app contains premium components, such as a map or address input components, users must have a Power Apps license to use the app. To request licenses for the users of your app, select **Request licenses** to submit a license request to your admin.



> **Note**
>
> You can't request licenses for security groups or distribution lists. For more information about requesting licenses, see [Request Power Apps licenses for your app users].

3. Select **Share**.

Share the link to the app

Unlike sharing canvas apps, sharing model-driven apps doesn't currently send an email with a link to the app.

To get the direct link to an app:

1. Sign in to Power Apps, and then select **Apps** on the left navigation pane.
2. Select the model-driven app you want, and then select **Details** on the command bar.
3. Copy the **Web link**. Alternatively, you can make a copy of the **Mobile QR code** for mobile users.
4. Paste the app URL in a location so that your users can access it, such as by posting it on a SharePoint site or send via email.

App sharing privilege and licensing requirements

There are a few key environment and licensing prerequisites required when sharing a model-driven app with a user.

- A Microsoft 365 user with Power Platform administrator rights must exist within the organization. Learn how to assign Power Platform Administrator rights. This provides the user with administrator rights over all environments.
- The app sharer must have admin privileges to the specific environment (or be a Power Platform administrator). The app sharer must have a security role with equal or greater privileges than the security role they're assigning to the app and to other users. Usually, this takes the form of the app sharer having the Dataverse **System Administrator** or **System Customizer** security role. These roles can be assigned by Power Platform administrators (who have rights over all Dataverse environments). The system administrator and system customizer security roles are standard roles that exist within all Dataverse environments.
- The user must exist as a user within the environment. It isn't enough to only be a Microsoft 365 user. This is because all users in an environment are accounted for and described within tables in the environment. Learn how to add a user to an environment
- Users must have the correct license to be able to use the app. Users can request a license themselves, or a maker can request licenses for their app users. Also, for custom model driven apps, the license must be assigned in either the users home tenant or the tenant hosting the app. Dynamics 365 model driven apps require a license in the same tenant the app is hosted.

> ⓘ **Note**

> Users who have the Environment Maker security role assigned don't require licenses to use model-driven apps.

Note

> Users who have the Environment Maker security role assigned don't require licenses to use model-driven apps.

About predefined security roles

There are several predefined roles available with Dataverse. To run apps that use only out-of-the-box tables, there's the Basic User security role, where members can play the app within the environment and perform common tasks for the records that they own. More information: Predefined security roles

Use Microsoft Entra groups to manage access

Administrators can use their organization's Microsoft Entra groups to manage access rights for licensed Dataverse users. Both types of Microsoft Entra groups—Microsoft 365 and Security—can be used to secure user-access rights to an app. More information: About group teams

See also

Discover more about creating users and assign security roles

Minimum privileges for common tasks

Run a model-driven app on a mobile device

Last updated on 03/31/2026

Privileges required to view and access apps

Article • 03/13/2024

Apps can be viewed by valid users with appropriate privileges who sign into Power Apps, the Power Apps mobile app, or Dynamics 365 home page. For a user to view and access apps in an environment, the following privileges, security role, or team membership are required:

- A user who has both write and create privileges on the Model-driven App table makes the user a maker persona. That user can view and access all apps in the environment.
- A user who has only the read privilege on the Model-driven App table, must also have the associated security roles that are assigned to the app (or equivalent).

Read, create, and write privileges can be granted to a user by the security roles assigned to the user or granted through team membership.

By default, users who have only the Basic User security role can only access model-driven apps that have that security role assigned to the app. This behavior is because the Basic User security role only has read privileges on the Model-driven App table.

However, users who have the System Administrator, System Customizer, or Environment Maker[1], security role can access and edit all model-driven apps within the environment. This behavior is because these security roles have create, read, and write privileges on the Model-driven App table. More information: Share a model-driven app using Power Apps

[1] Although the environment maker security role has read and write access to the Model Driven App table, the role alone doesn't have access to the core and custom tables required to view, create, and edit data in an app.

See also

Manage teams
Security roles and privileges

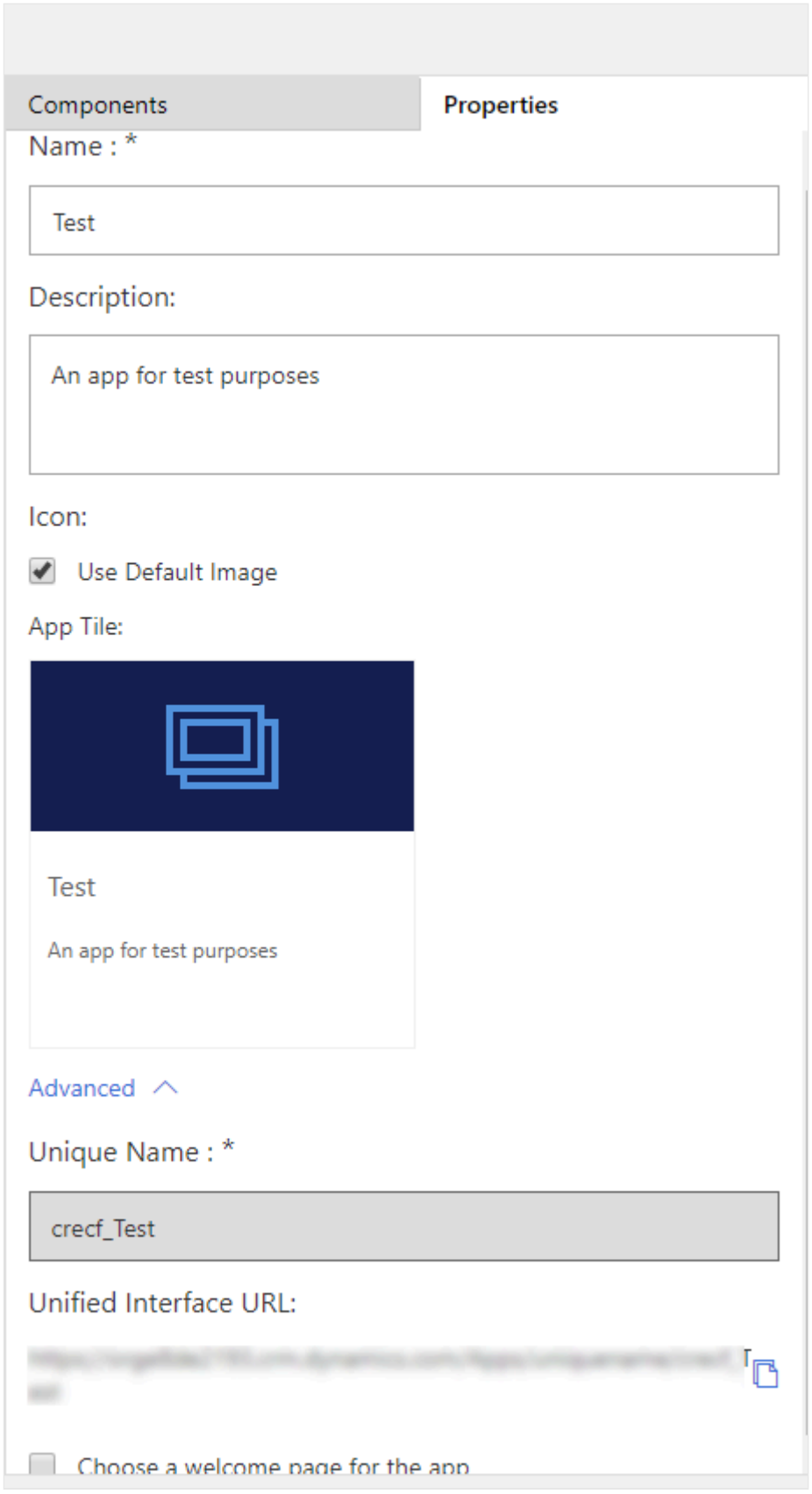
Manage model-driven app properties in the app designer (classic)

Summarize this article for me

App properties define important details about the app, like its title or URL. You define app properties when you create an app. If you want to change those properties later, you can do that in the app designer.

> **Note**
>
> This article describes how to view and set app properties using the classic app designer. For information how to view and set app settings in the modern designer, go to Manage model-driven app settings in the app designer.

1. In the app designer, on the right side, select the **Properties** tab.



2. Change the information, as required:

Expand table

Property	Description
Name	Enter a unique and meaningful name for the app.

Property	Description
Description	Type a short description of what the app is.
Icon	By default, the **Use Default App** check box is selected. To select a different web resource as an icon for the app, clear the check box, and then select an icon from the drop-down list. This icon is displayed on the app's preview tile.
Unique Name	You can't change the unique name. Using the unique name, you can query tables to get data from the database.
App URL Suffix[1]	The URL you chose while creating the app is shown here by default. You can change the app URL in the **Manage App** dialog box. Notice that you can't export or import the app URL suffix through a solution at this time.
Choose a welcome page for the app	This option allows you to select from the web resources available in your environment. The welcome pages you create can contain useful information to users such as links to videos, upgrade instructions, or getting started information. For more information about how to create a web resource, such as an HTML file that you can use as a welcome page, see Create and edit web resources to extend the web application.
Enable Mobile Offline	This option enables the app to be available offline on mobiles to the profiles that are selected using **Mobile Offline Profiles** drop down list.

[1]The **Client** and **App URL Suffix** properties are no longer available when you create a new app.

3. Save the app.

Next steps

Create or edit an app

Last updated on 02/21/2026

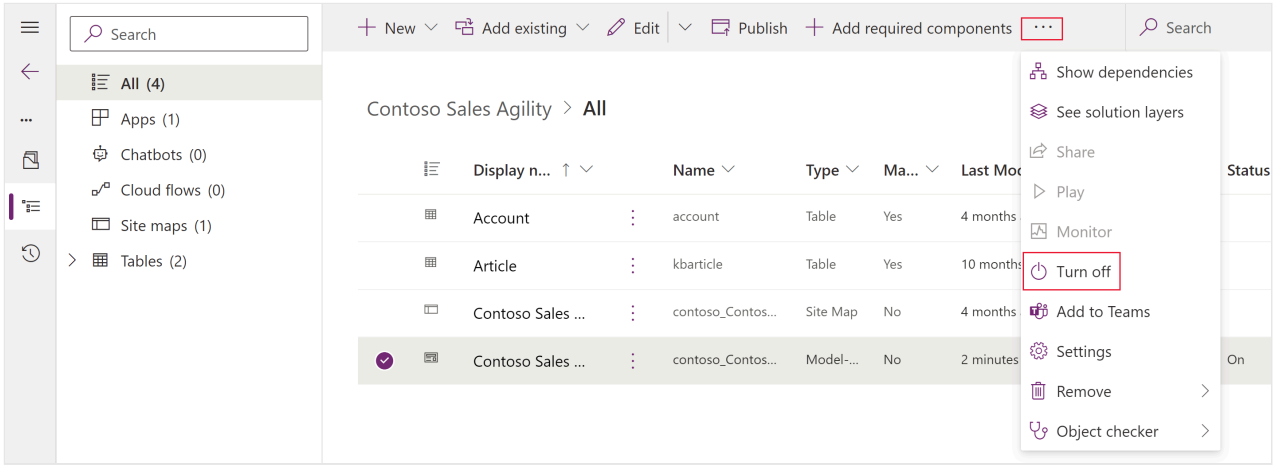
Hide a model-driven app from users

07/16/2025

Turn off a model-driven app to hide it from users who otherwise would be able to find and run it. When a model-driven app is turned off, it no longer appears in the common places where apps are found, such as the list of model-driven apps displayed in Power Apps Mobile or Office apps.

Turn off a model-driven app

1. Sign in to Power Apps.
2. Select the environment where the app is located.
3. Select **Solutions** on the left navigation pane, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
4. Select the model-driven app to be hidden from all users, and then on the command bar select **...** > **Turn off**.



The **Status** of the app appears as **Off** from the solution's **Objects** view.

> **Note**
>
> - Apps that are turned off are still visible to other users from the **Apps** area of Power Apps.
> - While a model-driven app is in a turned off state, it can't be played or shared from make.powerapps.com, however direct links to the app will continue to work.
> - Managed model-driven apps can't be turned off.

Turn on a model-driven app

An app that is turned off can be turned on by following the same steps described earlier and then selecting **Turn on**.

See also

Privileges required to view and access apps

Delete a model-driven app

Summarize this article for me

Apps might need to be deleted for a variety of reasons. An app might have been prepared purely as a development training exercise, might no longer be used in the organization, or could have been superseded in some way.

The process for removing an app is straightforward. However, dependencies can be encountered when you delete objects in Microsoft Dataverse. While you can delete apps at any time in an unmanaged solution, you can't delete tables that are used in the apps until such time as the app has been removed. The same is true for other app objects, such as table views.

> **Important**
>
> - If a model-driven app or site map is installed as part of a managed solution, see Delete a model-driven app that was installed as part of a managed solution for information about how to delete it.
> - Model-driven apps from Microsoft, such as Dynamics 365 Sales, Dynamics 365 Customer Service, and Dynamics 365 Field Service can’t be deleted. You can hide these apps from users by removing the security roles that are assigned to the app. Notice that these apps are visible to users with environment maker, system administrator, and system customizer security roles or any user with create privileges on the Model-driven App table.

Delete or remove an unmanaged app in your environment

1. Sign in to Power Apps.
2. On the left navigation, select **Solutions**, and then open the solution that has the app you want to delete. If the app isn't in a custom solution, select **Apps**.
3. Select the unmanaged model-driven app that you want to delete, and then select **Delete** on the command bar.
4. In the confirmation message choose from two options:
    - **Remove solution** removes the app from the solution, but retains it within the environment.

- **Delete from this environment** deletes it from both the solution and the environment.

If the component has dependencies, such as relationships, these must be removed before the app can be deleted. To see the dependencies of an app, select the app and then select **Show Dependencies** on the command bar.

5. When an app is deleted, you should also delete the associated site map. If you don't delete the site map, the site map designer displays an error the first time there is an attempt to create another app with the same name. However, it is possible to ignore the error, and the error doesn't appear when you try to create the app again.
   a. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
   b. Open the **Default Solution**.
   c. On the left navigation pane, select **Site maps**, select the site map that has the same name of the app that you deleted earlier, and then select **Delete from this environment**.
   d. Select **Delete** to confirm.

Delete a model-driven app that you installed as part of a managed solution

You can't delete a model-driven app or site map that you installed in the environment as part of a managed solution. Instead, delete the managed solution.

Delete a managed solution

All the components of a managed solution are deleted by deleting the solution.

> ⊗ **Caution**
>
> Be careful when you delete a managed solution. The solution might contain components that are used for production purposes, such as a line of business application. When you delete the solution, you delete all solution components and any data associated with the managed solution.

1. Sign in to Power Apps.
2. Select **Solutions** in the left navigation.
3. In the **Solutions** list, select the managed solution you want to delete. On the toolbar, select **Delete**.

Last updated on 03/04/2026

How to run a model-driven app

Article • 05/06/2025

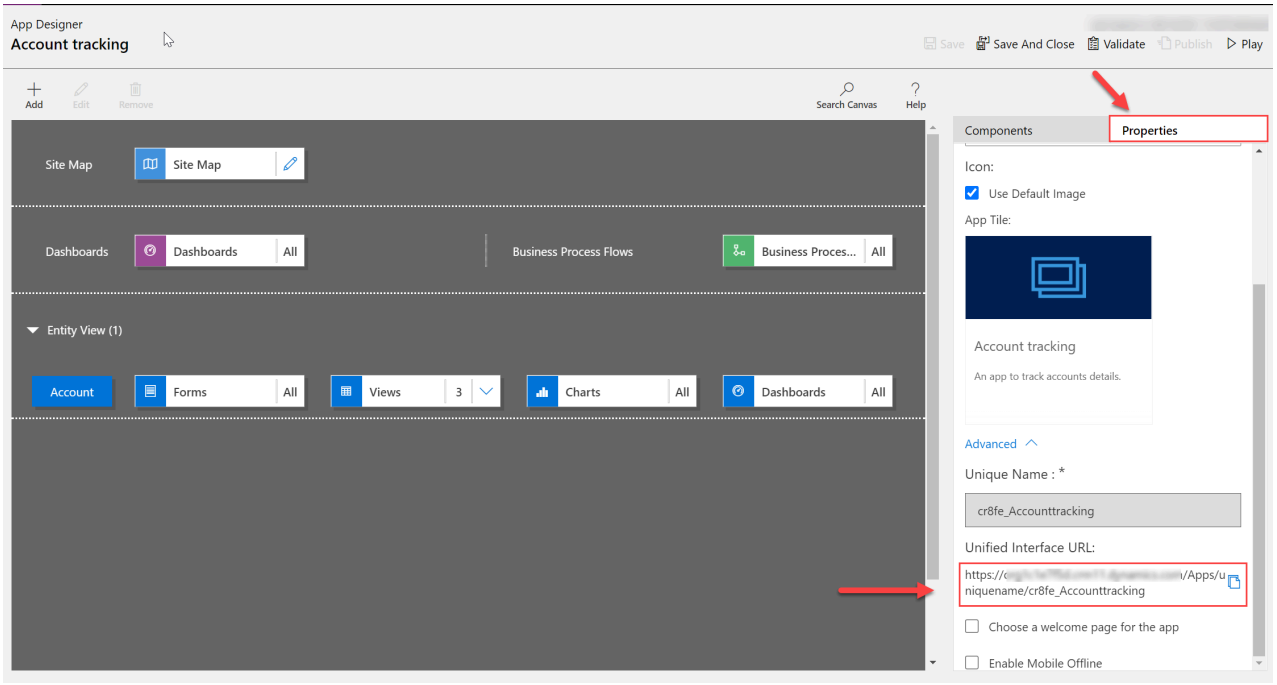
Run an app in a browser

To run a model-driven app in a web browser, the user must have a security role assigned in addition to having the URL for the app.

Learn how to create a security role from a copy

To get the direct link to an app:

1. Go to make.powerapps.com.
2. Select the **Environment** where the app is located.
3. On the left navigation pane, select **Apps**, or select **Solutions** and open the solution where the app is located. If the item isn't in the side panel pane, select ...More and then select the item you want.
4. Select the **...** next to the model-driven app, and then select **Edit**. The classic app designer opens.
5. Select the **Properties** tab, and then scroll down to **Unified Interface URL**. Copy the **Unified Interface URL**.



6. Paste the app URL in a location so that your users can access it, such as by posting it on a SharePoint site or send via email.

To play the app, enter the URL in a web browser.

Authentication prompts while running a model-driven app

You may notice authentication prompts during your model-driven app session. These prompts are expected and are required for certain features. Below highlights a list of features that trigger these prompts. This list is subject to change as new features are released:

- Custom pages.
- Power Fx expressions with commands.
- Opening the Office apps launcher located on the upper left of the app header.
- Collaboration feature.

Additionally, some organizational or browser settings may increase the frequency of these prompts. We recommend reviewing your settings to help reduce authentication prompts during sessions.

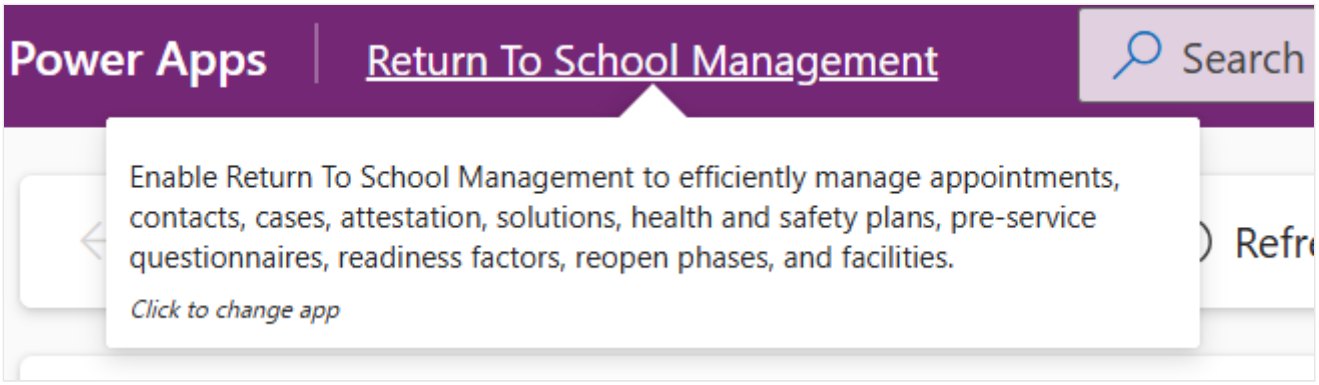
- **Third party cookies blocked in browser.** This will cause an authentication prompt to trigger each time you open a model-driven app. Note that opening the app in incognito mode or in the Safari web browser via ITP settings will usually block third-party cookies by default. Enable third party cookies to reduce these prompts.
- **Microsoft Entra conditional access or multi-factor authentication policies**. Policies specifically on Power Apps or Microsoft Graph will trigger additional authentication prompts. Contact your admin to review your organization's conditional access or MFA policies.

Run an app on a tablet or phone

Use the app URL described in the earlier section to run a model-driven app on a tablet's web browser. For phones, users download the Power Apps mobile app from the relevant app store. Then, sign into the app, and select the specific app. More information: Get started with Power Apps mobile

View the app description in the browser experience

When a user hovers over the app name at the top of the web browser, the app's description is shown.



If the app's description was generated using AI, a disclaimer is shown. Selecting the app name opens the app switcher where the user can play another app that they have access to.

See also

Create and edit columns for Microsoft Dataverse using Power Apps portal

Test model-driven applications with Test Engine (deprecated)

> **Note**
>
> **Test Engine is deprecated** and will be removed in a future release. Use Power Platform Playwright samples for test automation capabilities in Power Platform and Dynamics 365 services.

Testing your model-driven application with Test Engine requires the following steps:

1. Create a test plan
2. Set up your user
3. Run the test
4. View the results

Create a test plan

Create a yaml file using the test plan schema explained in Power Apps Test Engine YAML format (deprecated).

> **Tip**
>
> Review available sample plans in the GitHub repository.

Enable preview

Currently, all model-driven app extension testing capabilities require using Preview functions. To enable these, add `Preview` to the `allowPowerFxNamespaces` list in extensionModules of your testSettings.

This example shows you how to add this:

YAML

```yaml
testSettings:
  extensionModules:
```

```
    enable: true
    allowPowerFxNamespaces:
    - Preview
```

Set up your user

Test Engine currently has these authentication methods:

Expand table

Method	Description
Storage State	Store authenticated user state locally in the user profile using Microsoft Windows Data Protection API
Dataverse	Store authenticated user state inside Dataverse using your own customer managed key encrypted with an X.509 certificate using ASP.NET Core Data Protection

You can't store test credentials in test plan files. You must select `Storagestate` or `Dataverse` as the secure location to store credentials. The test plan file contains references to which environment variables are used for user persona. For example, the following YAML snippet indicates that the `user1Email` environment variables are used:

YAML

```yaml
environmentVariables:
  users:
    - personaName: User1
      emailKey: user1Email
```

View Users for more information.

Use the following PowerShell script to store the username in your environment variables.

PowerShell

```powershell
$env:user1Email = "someone@example.com"
```

Run the test

Use the PAC CLI pac test run command to run your test plan.

Required Parameters

You need to provide the following information:

- `--test-plan-file`: Path to your test plan file
- `--tenant`: Your tenant ID
- `--environment-id`: Your environment ID
- `--domain`: Domain URL of the model-driven app page to test

URL Formats for Model-Driven Apps

For model-driven apps, the `--domain` parameter varies based on the type of page you're testing. The URL format should specify the correct `pagetype` based on your testing scenario:

PowerShell

```
# For custom pages
pac test run `
   --provider mda `
   --test-plan-file your-testplan.te.yaml `
   --tenant your-tenantid-guid-value `
   --environment-id your-environmentid-guid-value `
   --domain "https://contoso.crm.dynamics.com/main.aspx?appid=00001111-aaaa-2222-bbbb-3333cccc4444&pagetype=custom&name=dev_home_c8017"

# For entity lists (views)
pac test run `
   --provider mda `
   --test-plan-file your-testplan.te.yaml `
   --tenant your-tenantid-guid-value `
   --environment-id your-environmentid-guid-value `
   --domain "https://contoso.crm.dynamics.com/main.aspx?appid=00001111-aaaa-2222-bbbb-3333cccc4444&pagetype=entitylist&etn=account&viewid=5a84c584-df1c-ed11-9db0-000d3a991110"

# For entity records (forms)
pac test run `
   --provider mda `
   --test-plan-file your-testplan.te.yaml `
   --tenant your-tenantid-guid-value `
   --environment-id your-environmentid-guid-value `
   --domain "https://contoso.crm.dynamics.com/main.aspx?appid=00001111-aaaa-2222-bbbb-3333cccc4444&pagetype=entityrecord&etn=account&id=72e0e163-df1c-ed11-9db0-000d3a991110"
```

Dataverse Integration

To enable Dataverse integration with your model-driven app tests, add the `enableDataverseFunctions` parameter in your testSettings:

YAML

```yaml
testSettings:
  extensionModules:
    enable: true
    allowPowerFxNamespaces:
    - Preview
    parameters:
      enableDataverseFunctions: true
```

When you enable Dataverse integration, you're required to open Azure CLI with a user or service principal that has rights to the Dataverse environment. You can use the command: az login More information: Sign in with Azure CLI.

The Dataverse API URL used for integration is obtained either from the host domain name of the pac test run `--domain` parameter or by defining an PowerShell environment variable named `DATAVERSE_URL`.

View the results

When the tests complete, you can view the results of your test in the `.trx` file located in the output directory. This folder contains any screenshots or videos captured by your tests when `recordVideo` is set to `true` in the test plan yaml.

Last updated on 04/18/2026

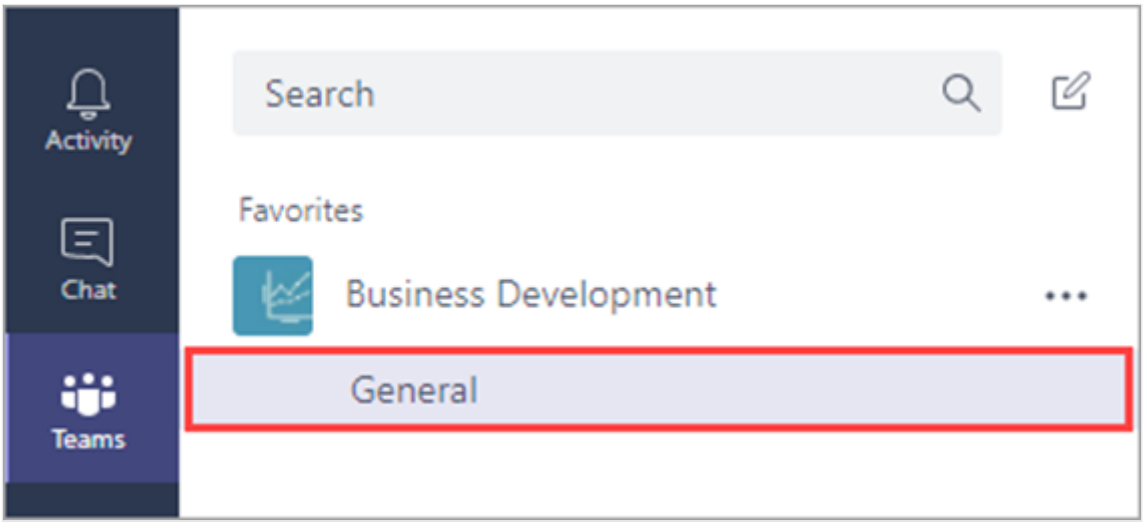
Embed a model-driven app as tab app in Teams (preview)

You can customize the Teams experience by adding Power Apps model-driven apps to your channels in Teams using the **Power Apps** tab.

> **Important**
>
> - This feature is deprioritized and will not be delivered.
> - New model-driven apps can no longer be added to Teams.
> - Model-driven apps that are embedded in Teams will continue to function until May 1, 2026. After that date, these apps will no longer work within the Teams experience. Users can continue to use these apps through a web browser or other supported clients.
> - Running model-driven apps on the Microsoft Teams mobile app is currently not supported.
> - Opening or downloading file attachments in a model-driven app when the app is embedded in Teams as a tab isn't supported.

Embed a model-driven app

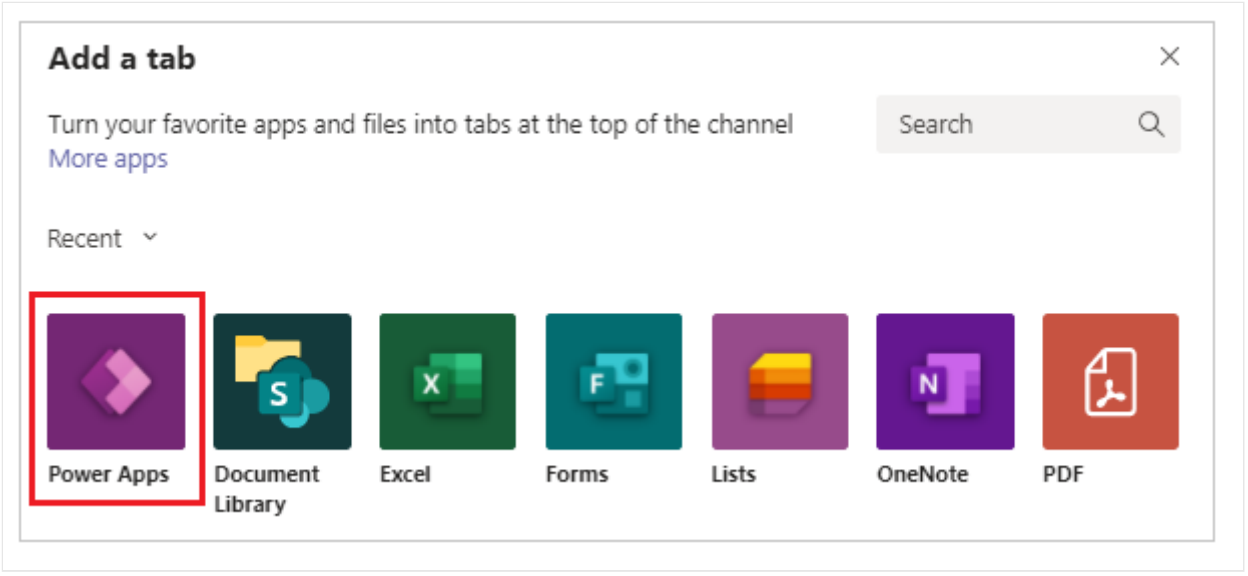
1. In Microsoft Teams, select a team, and a channel under that team. In this example, it's the **General** channel under the **Business Development** team.



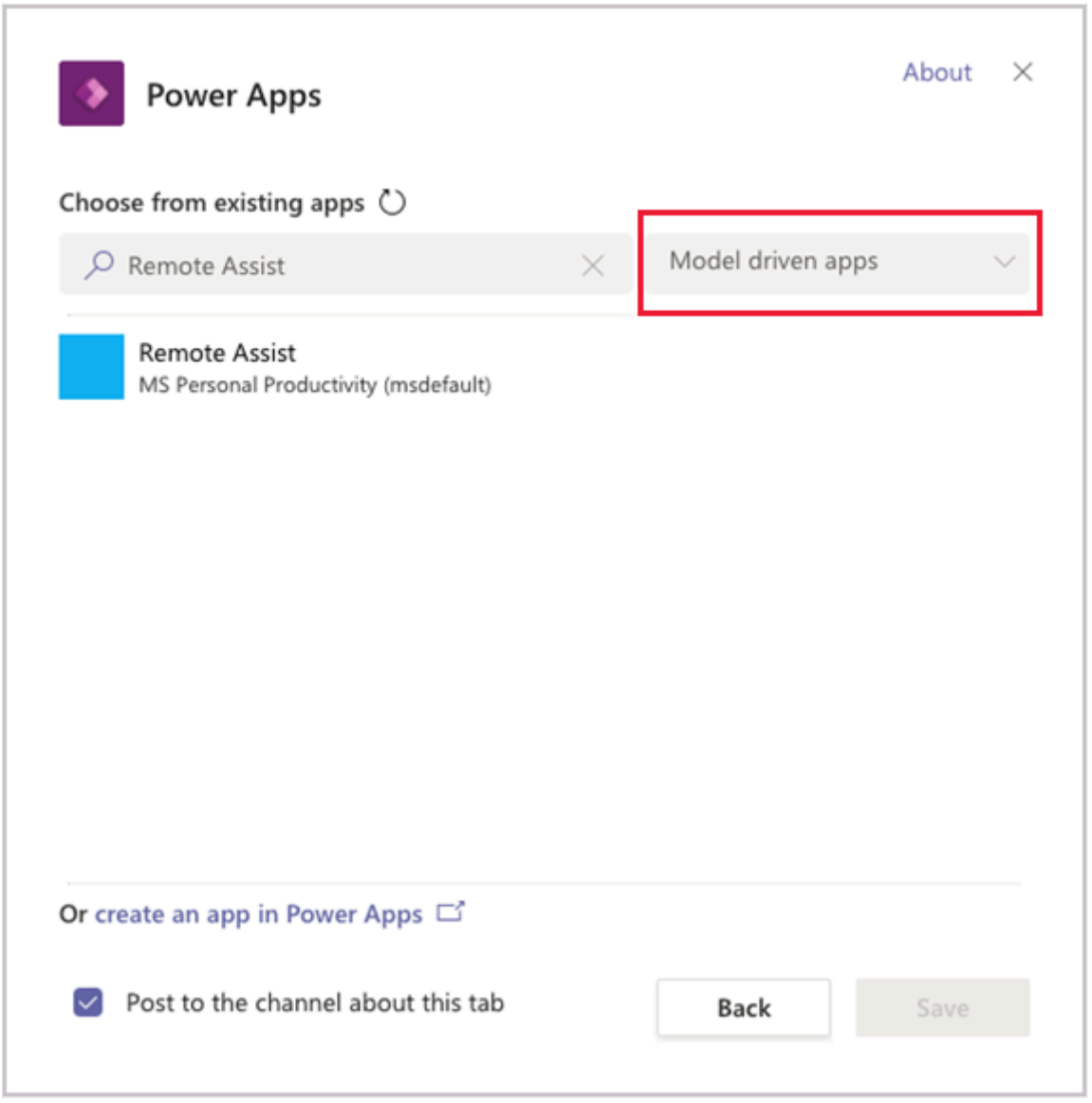
2. Select + to add a tab.



3. In the **Add a tab** dialog box, select **PowerApps**.



4. Select **Model-driven apps** in the dropdown list and then search for and select the model-driven app you would like to pin.



5. Select **Save**.

The app is now available to use in the channel.

See also

Embed a canvas app as personal app in Teams

Last updated on 04/02/2026

Embed a model-driven app as personal app in Teams (preview)

[This topic is pre-release documentation and is subject to change.]

You can share an app you've created by embedding it directly into Microsoft Teams. When completed, users can select + to add your app to any of **your** team channels or conversations in the team you are in. The app appears as a tile under **Tabs for your team**.

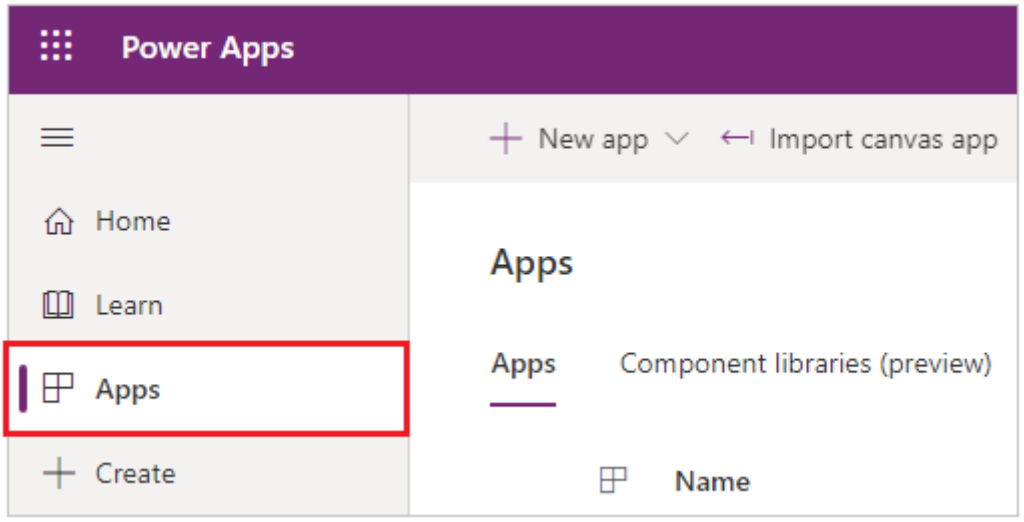
> **Important**
>
> - This feature is deprioritized and will not be delivered.
> - New model-driven apps can no longer be added to Teams.
> - Model-driven apps that are embedded in Teams will continue to function until May 1, 2026. After that date, these apps will no longer work within the Teams experience. Users can continue to use these apps through a web browser or other supported clients.

Prerequisites

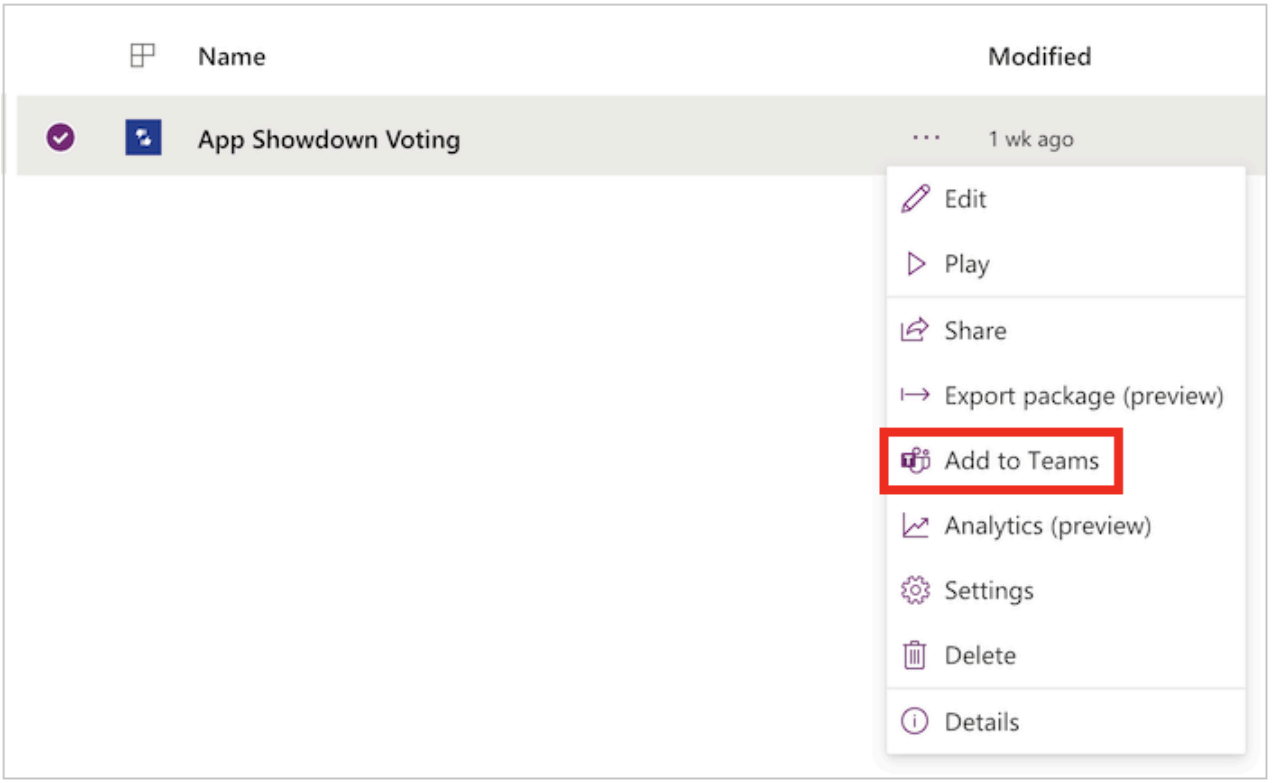
- You need a valid Power Apps license.
- To embed an app into Teams, you need an existing app created using Power Apps.

Add the app as a personal app

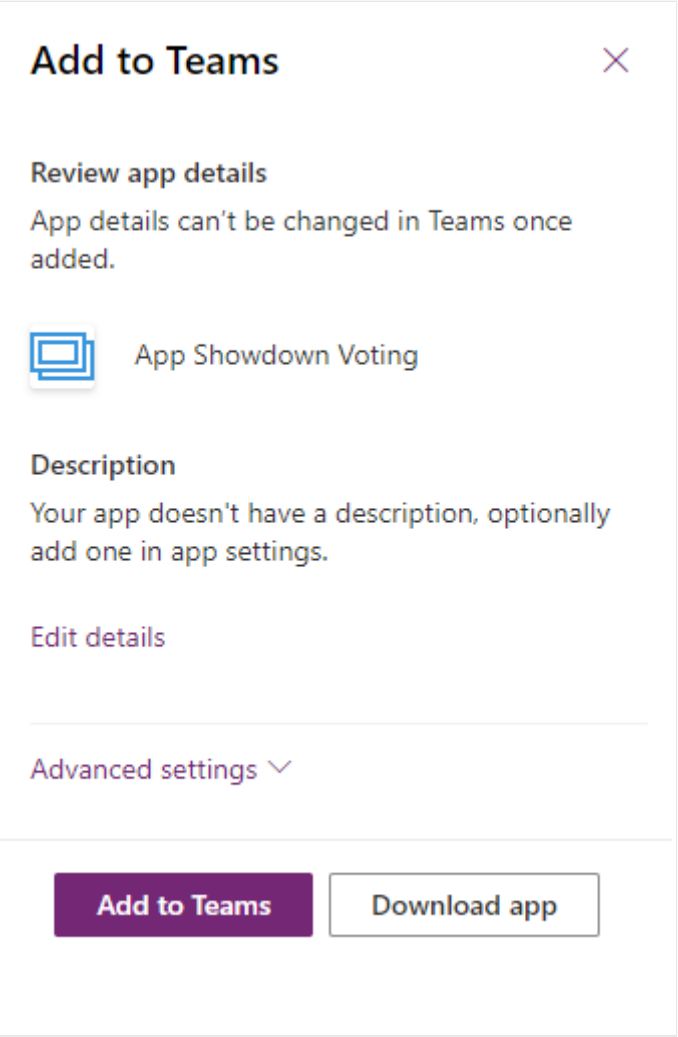
1. Sign into Power Apps, and then select **Apps** in the menu.



2. Select **More actions** (...) for the app you want to share in Teams, and then select **Add to Teams**.

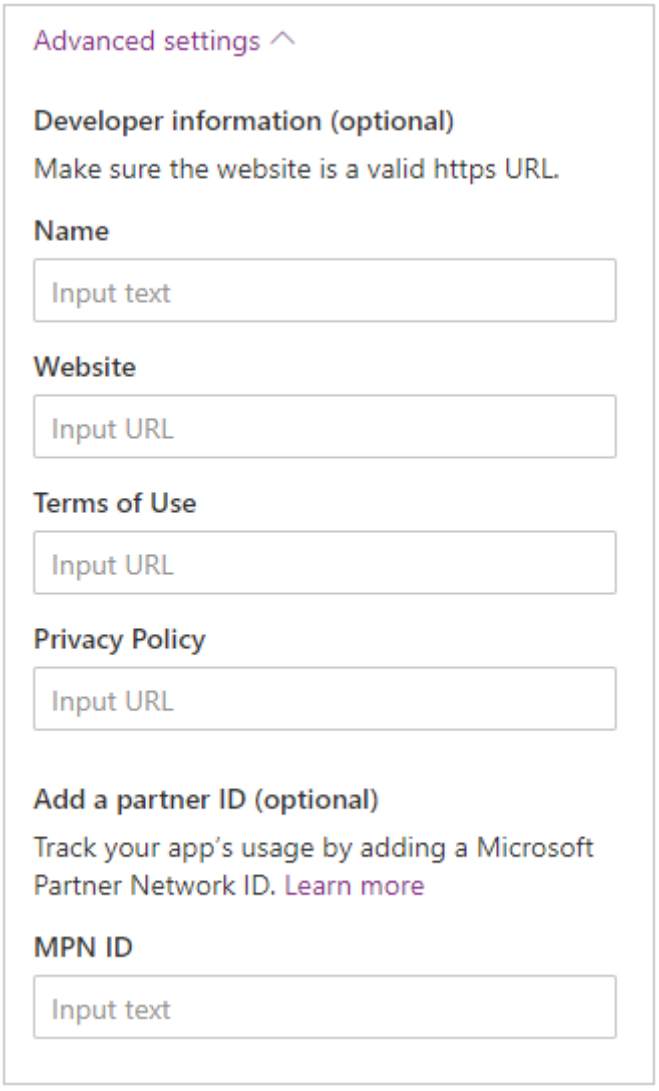


3. **Add to Teams** panel opens on the right-side of the screen.



4. (Optional) If the app doesn't have any description, select **Edit details** to open the app in add designer to add.

5. (Optional) Select **Advanced settings** to add additional details such as *Name*, *Website*, *Terms of Use*, *Privacy Policy*, *MPN ID* (Microsoft Partner Network ID).



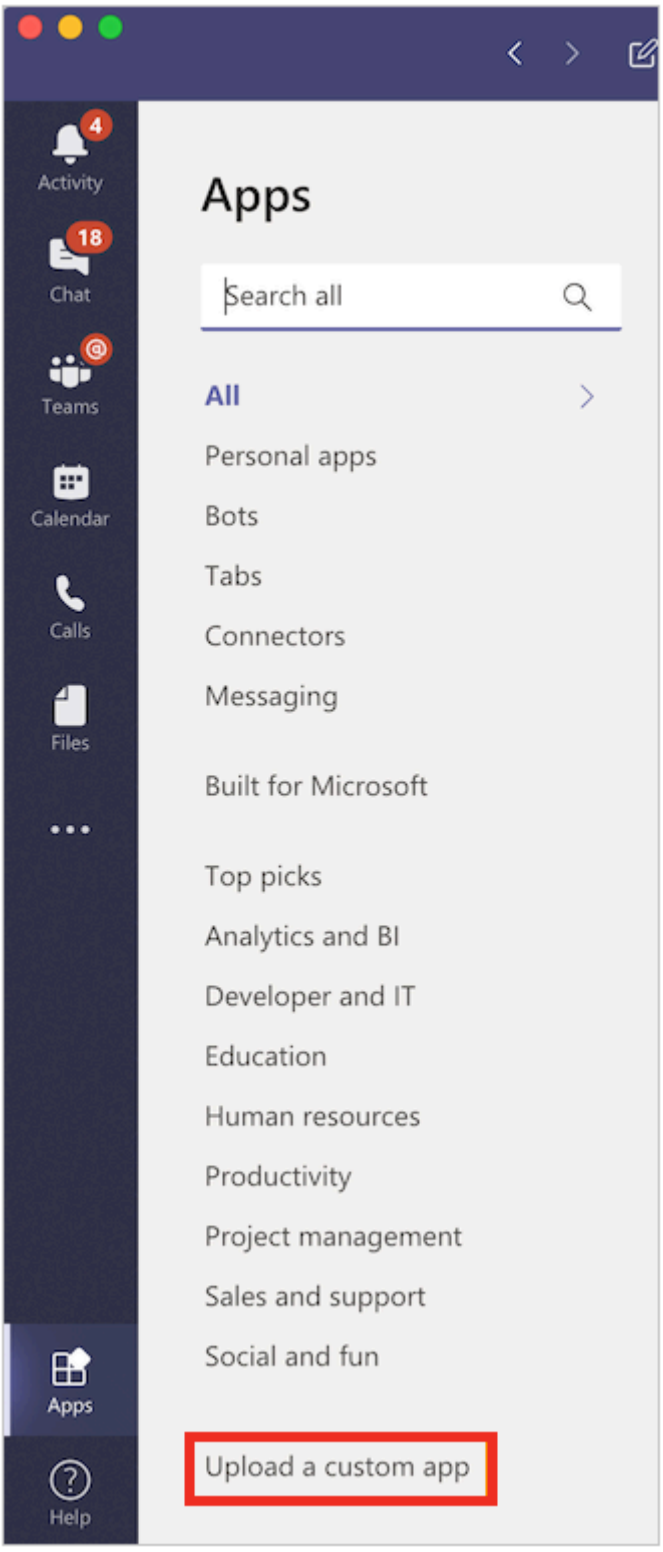
6. Select **Add to Teams** to add the app as a personal app or select **Add to a team** or **Add to a chat** to add the app as a tab within an existing channel or conversation.

Download app

You can also **Download app** in Power Apps. Power Apps will then generate your Teams manifest file using the app description and logo you've already set in your app.

1. To add the app as a personal app or as a tab to any channel or conversation, select **Apps** in the left navigation and then select **Upload a custom app**.

> ⓘ Note
>
> The **Upload a custom app** only appears if your Teams administrator has created a custom app policy and turned on **Allow uploading of custom apps**.



2. Select **Add** to add the app as a personal app or select **Add to team** to add the app as a tab within an existing channel or conversation.

Publish the app to the Teams catalog

If you're an admin, you can use the **Download app** option to also publish the app to the Microsoft Teams catalog.

Features currently unavailable

- Running **Model-driven apps** on the Microsoft Teams mobile app is currently not supported.
- Embedded canvas apps are not supported.
- Custom pages are not supported.

See also

Welcome to Microsoft Teams

Embed a model-driven app in Microsoft Teams (video)

Last updated on 04/02/2026

Navigate to advanced model-driven app making and customization areas

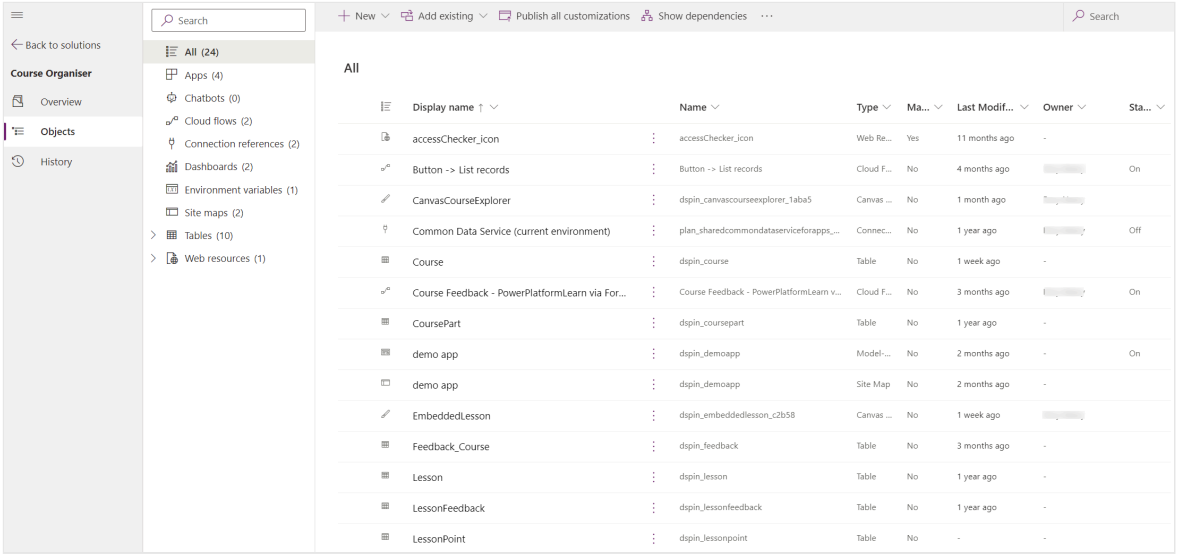
Article • 11/14/2024

This article describes how to access advanced customization and administration areas that are available within a Power Apps environment.

Solutions

The solutions area is where you can view, edit, create, import, export, and delete managed and unmanaged solutions.

1. Sign in to Power Apps.
2. On the top right, select the Environment where your solution is located.
3. On the left navigation pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
4. Select the solution you want.



Solution explorer

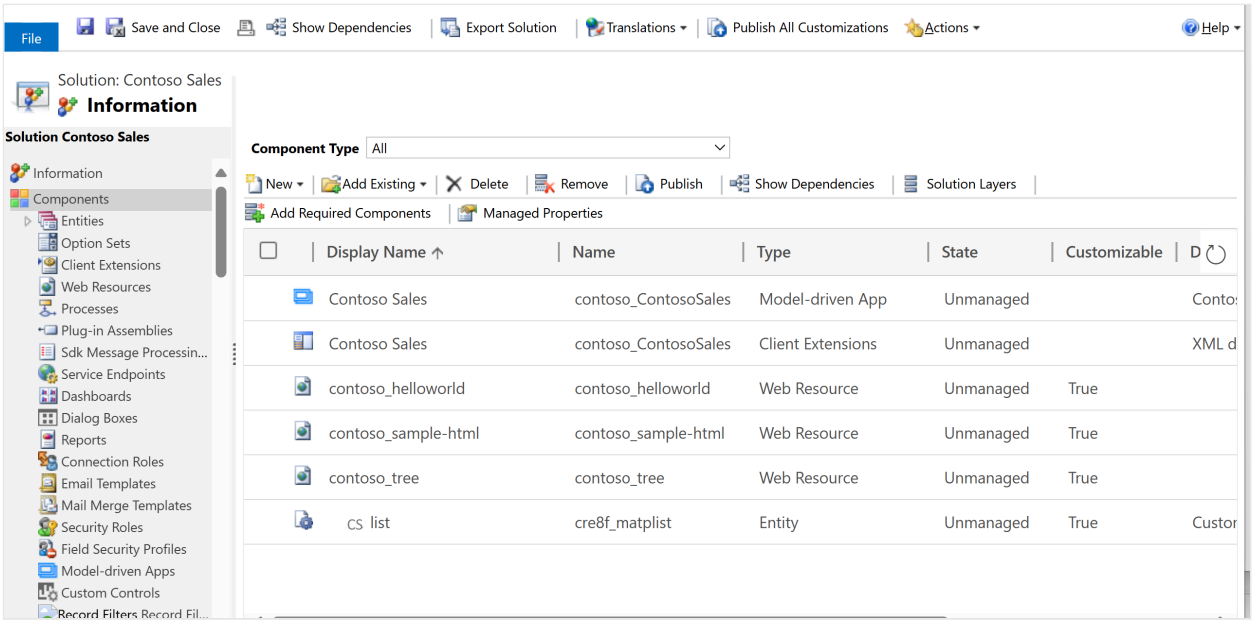
Use classic solution explorer to perform app making and customization tasks that can't be completed from the Power Apps website (make.powerapps.com).

1. From Power Apps (make.powerapps.com), go to **Solutions** on the left navigation pane, and then open the solution your want. If the item isn't in the side panel

pane, select ...More and then select the item you want.

2. On the command bar select **Switch to classic**.

The solution explorer is displayed in a new browser window.



Next steps

Create or edit web resources in model-driven apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Create or edit model-driven app web resources to extend an app

Web resources are typically used by developers to extend an app using files that are used in web development. App users may need to manage web resources provided by a developer or designer.

> **Tip**
>
> For an in-depth discussion of web resources, see **Developer Documentation: Web resources for model-driven apps**.

What are web resources?

Web resources are virtual files stored in the system. Each web resource has a unique name that can be used in a URL to retrieve the file. Think of them this way: If you had access to the actual web server running the web app, you could copy files over to that website. But with most online services, you can't do this. Instead, you can use web resources to upload files to the system and then reference them by name just as though you had copied them as files to the web server.

For example, if you create an HTML page as a web resource named "new_myWebResource.htm", you could open that page in a browser using a URL like this:

```
<base URL>/WebResources/new_myWebResource.htm
```

where <*base URL*> is the part of the URL you use to view apps that end in `dynamics.com`. Because the web resource is data in the system, only licensed users for your organization can access them this way. Normally, web resources are included in forms rather than referenced directly. The most common usage is to provide JavaScript libraries for form scripts.

Because web resources are data in the system and are solution aware, you can move them to different organizations by exporting them as part of a solution and importing the solution into a different organization. You must use solution explorer to work with web resources.

File types accepted as a web resource are HTML, CSS, JScript, XML, PNG, JPG, GIF, ICO, and XSL.

Solution publisher prefix

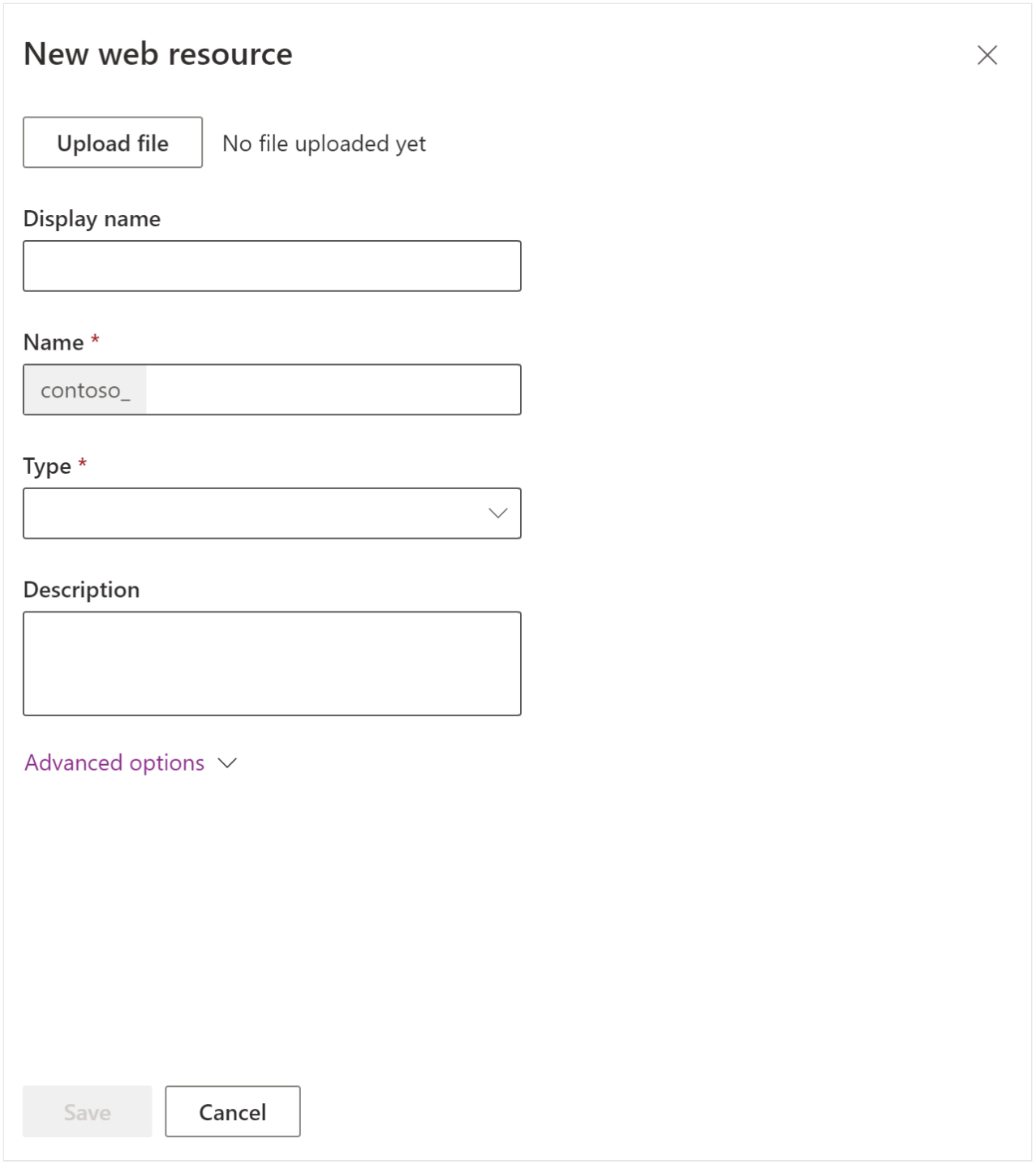
Part of the name of any web resource you create is the solution publisher prefix. This is set based on the solution publisher for the solution you're working in. Make sure that you're working in the unmanaged solution where the publisher prefix is the one you want for this web resource. More information: Solution publisher

Add a web resource to a solution

1. Sign in to Power Apps, and then on the left navigation pane select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open an unmanaged solution, on the command bar select **Add existing**, point to **More**, and then select **Web resource**. The list of web resources available are displayed.
3. To add a web resource to the unmanaged solution, select the web resource, and then select **Add**.

Create or edit a web resource for a solution

1. Sign in to Power Apps, and then on the left navigation pane select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open an unmanaged solution.
3. To edit an existing web resource that is part of the unmanaged solution, select it to display the **Edit web resource** pane. To create a new one, on the command bar select **New**, point to **More**, and then select **Web resource**.
4. Complete the fields on the right web resource pane to create or edit the web resource:



Expand table

Column	Description
Upload file	Select the **Upload file** button to choose a file to upload as a web resource. • You can upload a file when creating a new web resource or to overwrite an existing web resource. • The file name extension of the file must match allowed extensions. • By default the maximum size file that can be uploaded as a web resource is 5MB. This value can be modified by using the **System Settings** > **Email** tab > **Set file size limit for attachments** setting. More information: System Settings dialog box - Email tab

Column	Description
Display Name	The name displayed if you view a list of web resources.
Name	*Required*. The unique name for this web resource. You can't change this after you save the web resource. • This name can only include letters, numbers, periods, and nonconsecutive forward slash ("/") characters. • The solution publisher customization prefix will be prepended to the name of the web resource.
Type	*Required*. This is the type of web resource. You can't change this after you save the web resource.
Description	A description of the web resource.
Advanced options	Expand **Advanced options** to reveal the **Language** field, which allows for a selection of a language. This option just tags the row that stores the web resource data. It doesn't change the behavior of the web resource.

5. After you have added your changes, select **Save** and then select **Publish**. After you save the web resource, the **URL** to the web resource is displayed in the web resource definition. Select this link to view the web resource in your browser.

> ⓘ Note
>
> Changes to a web resource will not be visible in the application until you publish it.

Add a web resource to a model-driven app form

1. Open the form where you want to add a web resource for editing.
2. Select the **Components** pane, expand **Display**, and then select the web resource type you want:
   - **HTML Web resource**. A webpage (HTML) you can use to create user interface elements for client extensions. More information: Webpage (HTML) web resources
   - **Image web resource**. Use image web resources to make images available for use in model-driven apps. Supported image formats are: PNG, JPG, GIF, ICO, and SVG (vector format). More information: Image web resources
3. Select the web resource from the list and then select **Add**. Alternatively, select **New web resource** to create the web resource you want. More information: Web resource

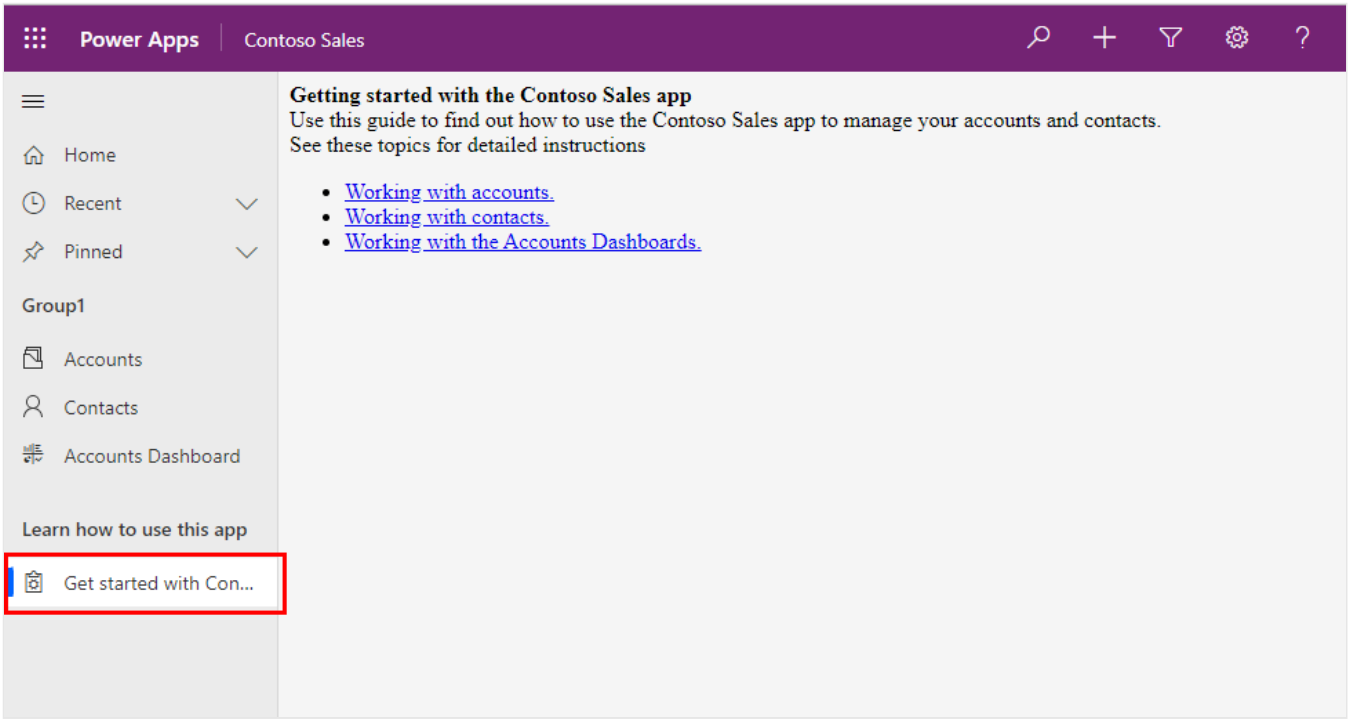
properties for model-driven app forms

> **Note**
>
> You can't add a web resource to a form header or footer.

Add a web resource to an app (preview)

[This topic is pre-release documentation and is subject to change.]

You can add a web resource as a subarea type in a model-driven app. Web resources are files you can use to extend a model-driven app. The web resource subarea can be HTML code or an HTML file, such as .htm or .html. When the user selects the subarea, the HTML is displayed in the app.



> **Note**
>
> By default, web resource subareas are disabled. To enable, go to Navigation pane options.

1. In the model-driven app designer, select **Navigation** > **Add** > **Subarea**.

2. In the subarea properties, enter the following information:

   - **Content type**: **Web resource**.
   - **URL**: Select an HTML web resource file name from the list to add the web resource's URL. More information: Create or edit model-driven app web resources to extend an

app

- **Title**: Enter the title that will appear on the subarea in the app.

3. Select **Save**, and then select **Publish**.

Edit a web resource using the classic solution explorer

1. Open an unmanaged solution and select the web resource.
2. Select the vertical ellipsis > **Edit** > **Edit in classic**.
3. Complete the form to create the web resource:

Expand table

Column	Description
Name	*Required*. This is the unique name for this web resource. You can't change it after you save the web resource. • This name can only include letters, numbers, periods, and nonconsecutive forward slash ("/") characters. • The solution publisher customization prefix will be prepended to the name of the web resource.
Display Name	The name displayed if you view a list of web resources.
Description	A description of the web resource.
Type	*Required*. Type of web resource. You can't change this after you save the web resource.
Text Editor	When the type of web resource represents a kind of text file, select this button to open a page to edit the content using the text editor. More information: Use the text editor appropriately
Language	Allows for a selection of a language. This option just tags the row that stores the web resource data. It doesn't change the behavior of the web resource.
Upload File	Select the **Choose File** button to choose a file to upload as a web resource. • You can upload a file when creating a new web resource or to overwrite an existing web resource. • The file name extension of the file must match allowed extensions. • By default the maximum size file that can be uploaded as a web resource is 5 MB. This value can be modified by using the **System Settings** > **Email** tab > **Set file size limit for attachments** setting. More information: System Settings dialog box - Email tab
URL	After you save the web resource, the URL to the web resource will be displayed here. Select this link to view the web resource in your browser.

1. After you have added your changes, select **Save** and then select **Publish**.

Use the text editor appropriately

The text editor provided in the application for web resources should only be used for simple edits of text files. You can use it to create and edit HTML web resources, but you should only edit HTML web resources that were created using the text editor. The text editor is designed for very simple HTML content.

> **Important**
>
> If the content of an HTML web resource wasn't created using the text editor, don't use the text editor to edit it.
> The text editor uses a control that modifies the HTML source in a way that allows it to be edited. These changes can make the page behave differently in the browser and cause more sophisticated code to stop working. Opening an HTML web resource with the text editor and saving it without making any changes can break some HTML web resources.

We recommend that you use an external editor to edit text files and then save them locally before uploading them with the **Upload File** button. This way you can preserve a copy of the web resource if you need to return to an earlier version. You can use a simple editor like Notepad, but a text editor with more advanced capabilities is highly recommended. Visual Studio Community and Visual Studio Code are free and provide powerful capabilities for editing the text-based files used by web resources.

Create and edit a web resource on a form using solution explorer

Add or edit web resources on a form to make it more appealing or useful to users.

Open an unmanaged solution

1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. In the **All Solutions** list open the unmanaged solution you want.

Navigate to a form

With solution explorer open, under **Components**, expand **Entities**, and then expand the table you want to work with.

Choose **Forms**, in the list locate a form of type Main, and then double-click or tap the entry to open and edit the form.

Add or edit web resource in a form

See Web resource properties for information about the properties you can set for web resources in a form.

Preview

To preview how the main form will appear and how events will function:

- On the **Home** tab, select **Preview**, and then select **Create Form**, **Update Form**, or **Read-Only Form**.
- To close the Preview form, on the **File** menu, select **Close**.

Save

When you finish editing the form, on the **Home** tab, select **Save and Close** to close the form.

Publish

When your customizations are complete, publish them:

- To publish customizations for only the component that you're currently editing, in the navigation pane, select the table you have been working on, and then select **Publish**.
- To publish customizations for all unpublished components at one time, in the navigation pane, select **Entities**, and then on the **Actions** toolbar, select **Publish All Customizations**.

See also

Web resource properties
Create and design forms
Understand model-driven app components
Developer Documentation: Web resources in model-driven apps
Create or edit model-driven app web resources (video)

Last updated on 01/12/2026

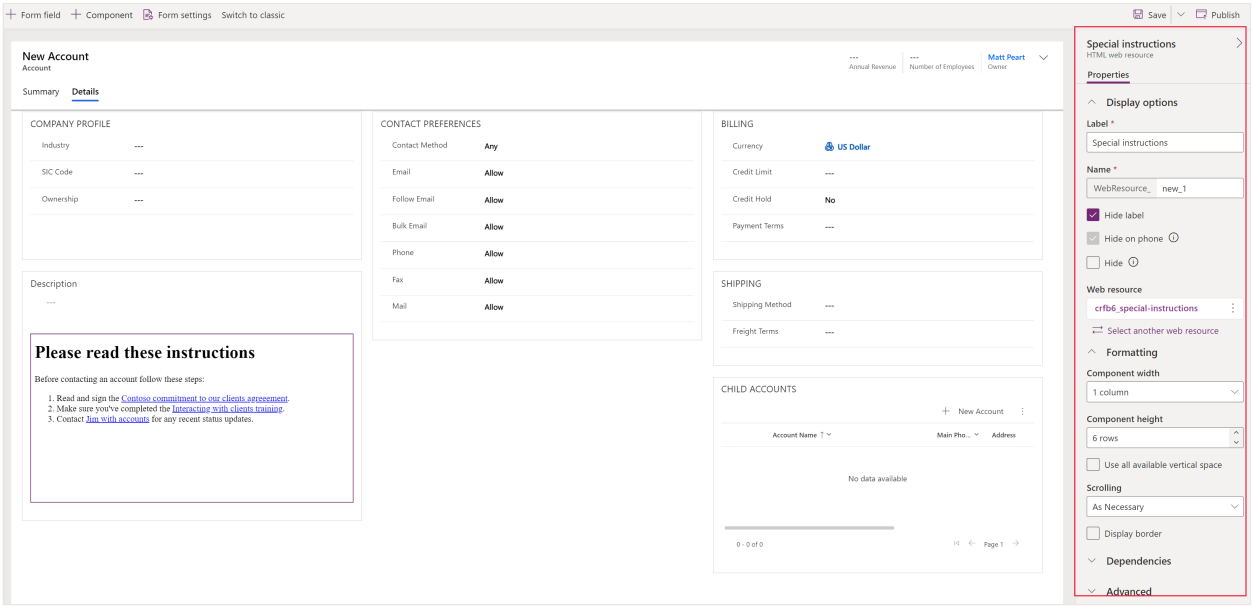
View and edit web resource properties for model-driven app forms

Article • 06/13/2022

This article describes how to use the form designer to view and edit web resources on a form. Form enabled web resources are image or HTML files.

View and edit a web resource on a form

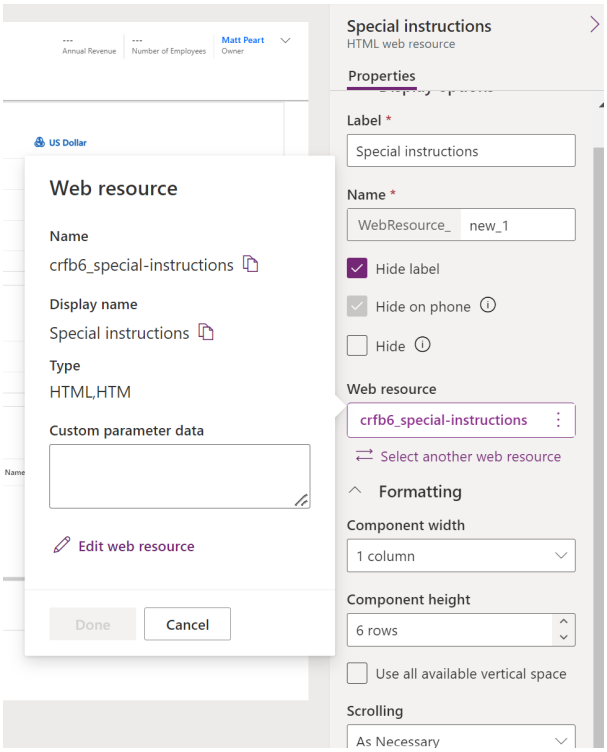
Open a main form and select the area of the form that has the web resource. The web resource properties are displayed on the right properties pane.



These properties define the web resource to use and how it should behave. After you've made the changes you want, select **Save**. To make the changes available to users, select **Publish**.

Display options

Property	Description
Label	Is the friendly name and is automatically generated based on the **Name** column value. Specify localizable text for the web resource control that will be added to the form.
Name	Specify a name for the web resource control that will be added to the form. This value uniquely identifies the control in the form.

Property	Description
Web resource	Select the web resource to display additional web resource properties. If you are using **Custom parameter data** enter it here. This is usually configuration data that will be passed to the HTML web resource as a `data` query string parameter. Scripts associated with the HTML page can access this data and use it to change the behavior of the page. Select **Edit web resource** to make changes such as load a different HTML or image file. 
Hide label	Select this option if you don't want the label displayed on the form.
Hide on phone	Select this option when you don't want this web resource to be visible in mobile apps.
Hide	While this is enabled the web resource won't be visible when the form loads. If you have a business rule or form script that will show the web resource as needed, check this property. More information: Show or hide form elements

Formatting

Property	Description
Component width	When the section containing the web resource has more than one column you can set the column to occupy up to the number of columns that the section has.
Component height	You can control the height of the web resource by specifying a number of rows or select **Use all available vertical space** to allow the web resource height to expand to available space.

Property	Description
Scrolling	An HTML web resource is added to the form using an IFRAME. - **As Necessary**: Show scrollbars when the size of the web resource is larger than the available. - **Always**: Always show scrollbars. - **Never**: Never show scrollbars.
Display border	Displays a border around the HTML web resource.
Vertical alignment	Image web resources can be aligned **Middle**, **Top**, or **Bottom**.
Horizontal alignment	Image web resources can be aligned **Center**, **Left**, or **Right**.
Size	For image web resources select from **Use available space**, **Use available area**, **Original image size**, or **Specific size** (specify **Height** and **Width** in pixels).

Dependencies

Property	Description
Table column dependencies	An HTML web resource may interact with columns in the form using script. If a column is removed from the form the script in the web resource may break. Add any columns referenced by scripts in the web resource to this property so that they can't be removed accidentally.

Advanced

Property	Description
Restrict cross-frame scripting where supported	Use this option if you don't fully trust the content in the HTML web resource. More information: Developer Documentation: Select whether to restrict cross-frame scripting
Pass row object-type code and unique identifier as parameters	Data about the current row visible in the form can be passed to the HTML web resource page so that script running in the page can access data about the row. More information: Pass parameters to web resources Developer Documentation: Pass contextual information about the row

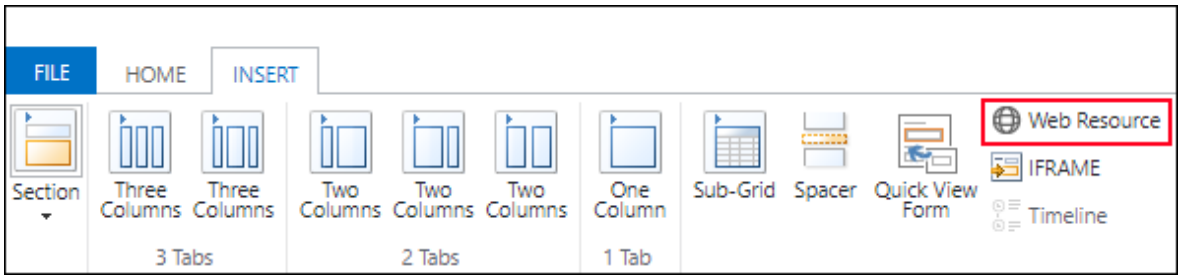
Web resource properties for model-driven app forms (classic)

This section describes how to use the legacy app authoring interface to add or edit web resources on a form to make it more appealing or useful to app users. Form enabled web resources are images or HTML files controls.

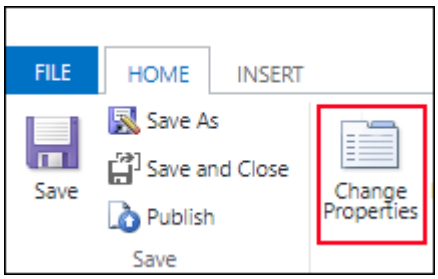
Access web resource properties

While viewing a form:

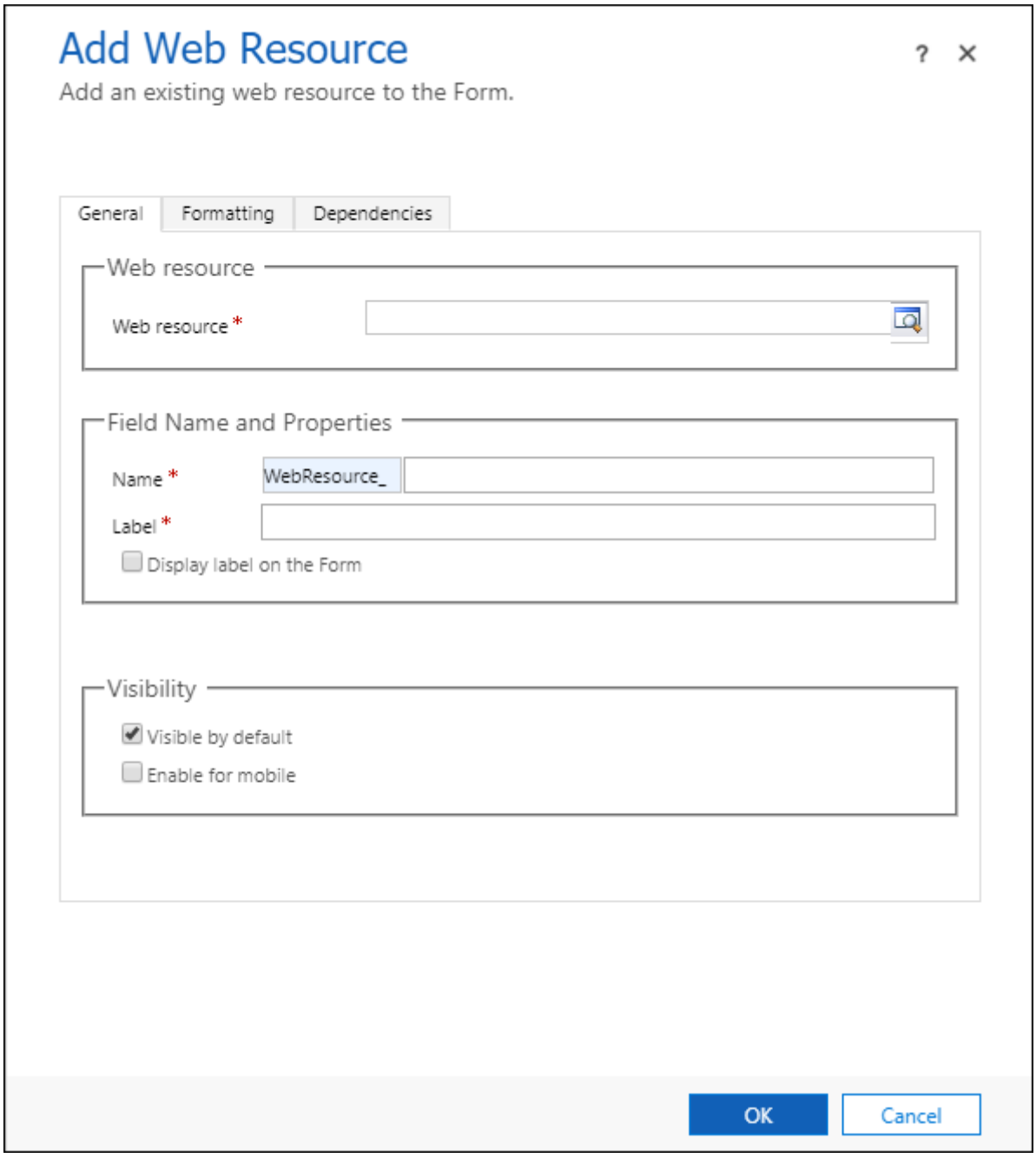
- **When adding a web resource:**: Select the tab (for example, **General** or **Notes**) you would like to insert it on, and then on the **Insert** tab, select **Web Resource**.



- **When editing a web resource**: Select a form tab and the web resource that you want to edit, and then on the **Home** tab, select **Change Properties**.



This will open the **Add Web Resource** or **Web Resource Properties** dialog box.



> **ⓘ Important**
>
> You must select the **Visible by default** option for the web resource to appear on the form and be available to users.

Web resource properties

The **Add Web Resource** or **Web Resource Properties** dialog box will have two, sometimes three tabs depending on the type of web resource.

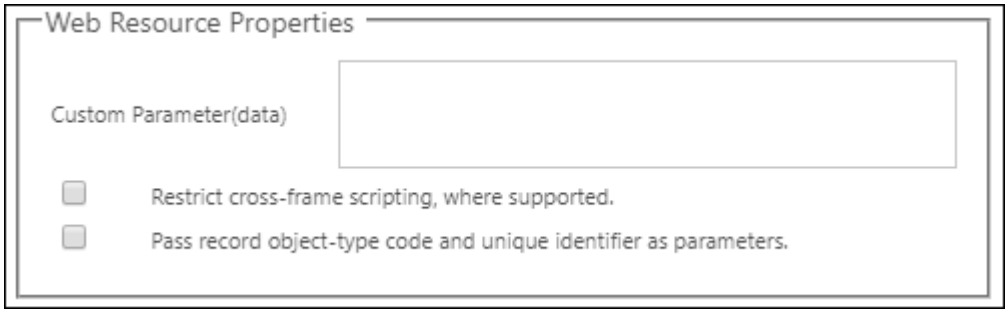
General tab

These properties define the web resource to use and how it should behave.

Column	Description
Web resource	*Required.* Look up an existing web resource or create a new one. Use the **Form Enabled Web Resource** view to include only HTML and image web resources that can be added as visual elements in a form.
Name	*Required.* Specify a name for the web resource control that will be added to the form. This value uniquely identifies the control in the form.
Label	*Required.* Automatically generated based on the **Name** column value. Specify localizable text for the web resource control that will be added to the form. Select **Display label on the form** if you want to make this visible.
Visible by default	While this is enabled the web resource will be visible when the form loads. If you have a business rule or form script that will show the web resource as needed, uncheck this column. More information: Show or hide form elements
Enable for mobile	Select this option to allow this web resource to be visible in mobile apps.

Depending on the type of web resource you select, set additional properties.

For HTML web resources you will see these:

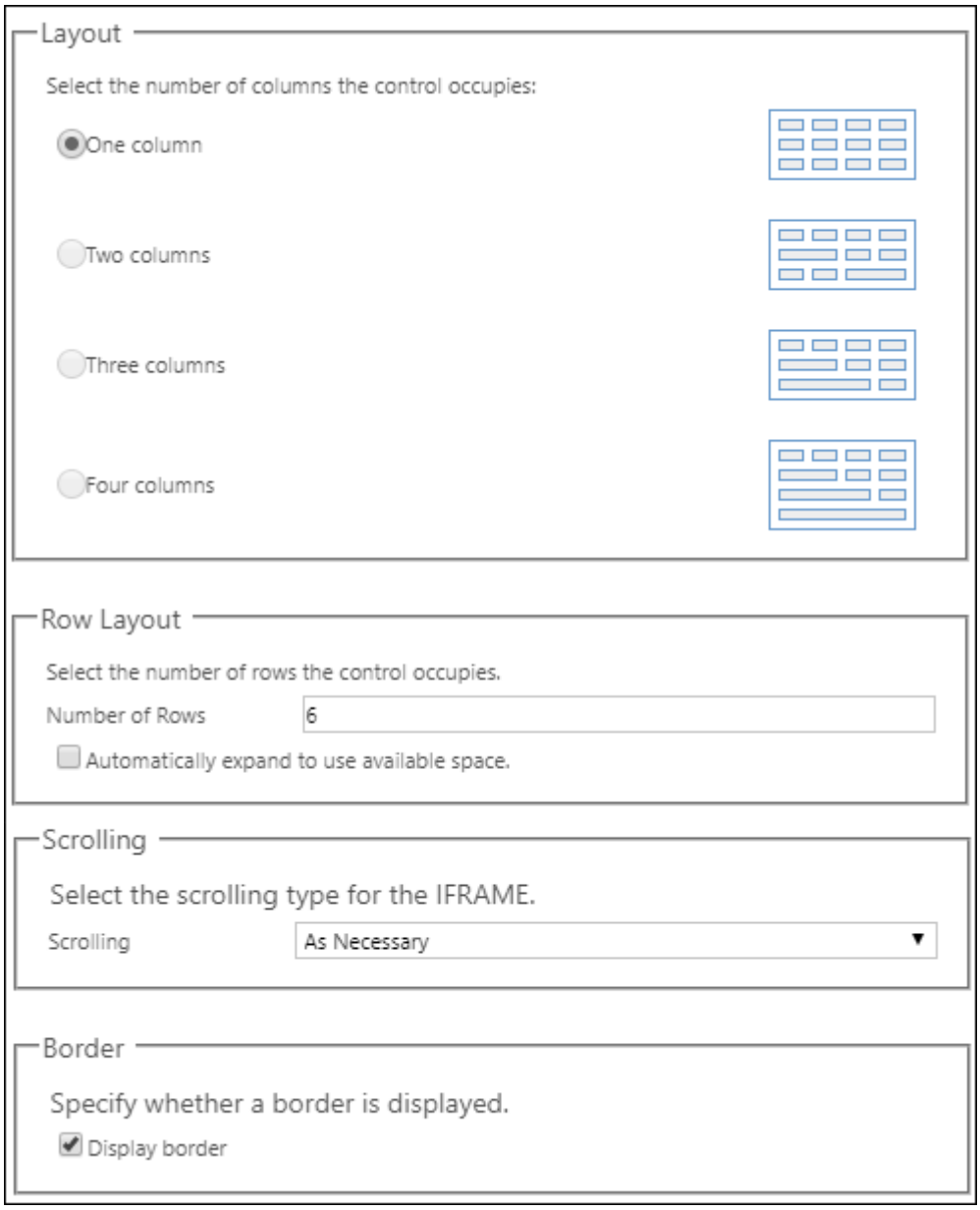


Column	Description
Custom Parameter(data)	Usually configuration data that will be passed to the HTML web resource as a `data` query string parameter. Scripts associated with the HTML page can access this data and use it to change the behavior of the page.
Restrict cross-frame scripting where supported	Use this option if you don't fully trust the content in the HTML web resource. More information: Developer Documentation: Select whether to restrict cross-frame scripting
Pass row object-type code and unique identifier as parameters	Data about the current row visible in the form can be passed to the HTML web resource page so that script running in the page can access data about the row. More information: Pass parameters to web resources Developer Documentation: Pass contextual information about the row

For Image web resources you have the option to specify **Alternative Text** that is important for assistive technologies that make the page accessible to everyone.

Formatting tab

On the **Formatting** tab, the options that display vary based on the type of web resource inserted and the context where it is inserted. These options include specifying the number of columns and rows display, whether a border displays, and the scrolling behavior.



Property	Description
Select the number of columns the control occupies	When the section containing the web resource has more than one column you can set the column to occupy up to the number of columns that the section has.

Property	Description
Select the number of rows the control occupies	You can control the height of the web resource by specifying a number of rows or select **Automatically expand to use available space** to allow the web resource height to expand to available space.
Select the scrolling type for the IFRAME	An HTML web resource is added to the form using an IFRAME. - **As Necessary**: Show scrollbars when the size of the web resource is larger than the available. - **Always**: Always show scrollbars. - **Never**: Never show scrollbars.
Display border	Display a border around the web resource.

Dependencies tab

A web resource may interact with columns in the form using script. If a column is removed from the form the script in the web resource may break. Add any columns referenced by scripts in the web resource to the **Dependent columns** so that they cannot be removed accidentally.



Pass parameters to web resources

An HTML web resource can accept parameters to be passed as query string parameters.

Information about the row can be passed by enabling the **Pass row object-type code and unique identifiers as parameters** option. If information is typed into the **Custom Parameter(data)** column it will be passed using the data parameter. The values passed are:

Parameter	Description
`data`	This parameter is only passed when text is provided for **Custom Parameter(data)**.
`orglcid`	The Organization default language LCID.
`orgname`	The name of the organization.
`userlcid`	The user's preferred language LCID
`type`	**Don't use this.** The table type code. This numeric value can be different for custom tables in different organizations. Use table type name instead.
`typename`	The table type name.
`id`	The id value of the row. This parameter has no value until the table row is saved.

Any other parameters are not allowed and the web resource will not open if other parameters are used. If you need to pass multiple values, the data parameter can be overloaded to include more parameters within it.

More information: Developer Documentation: Pass contextual information about the row

See also

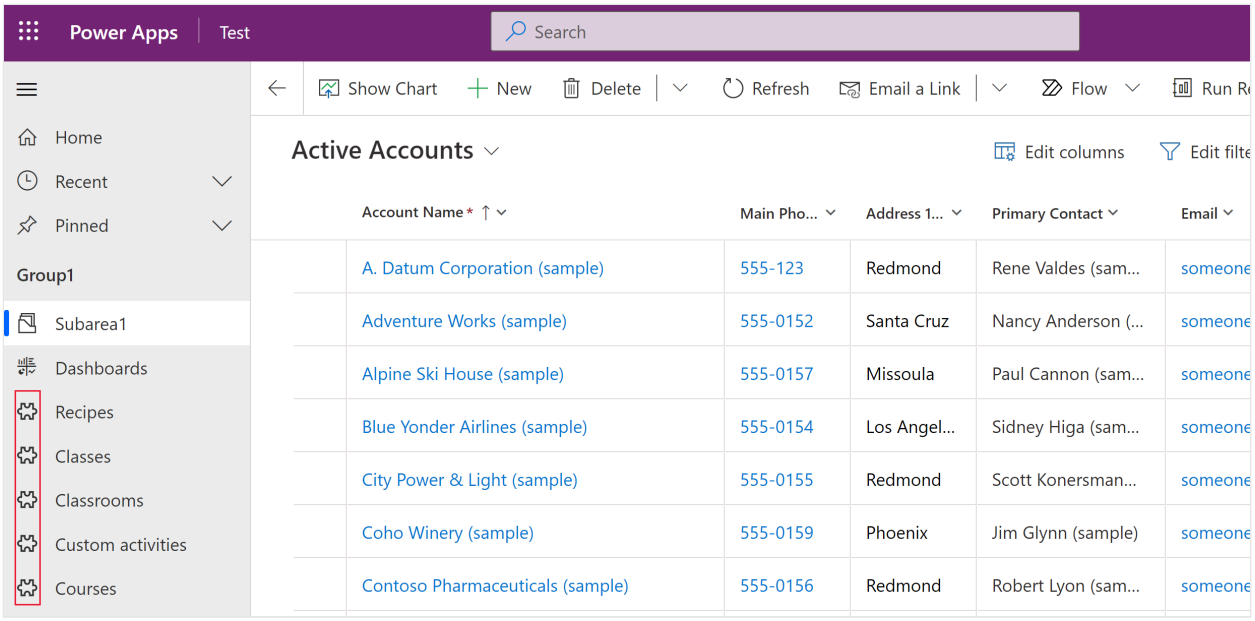
Create or edit web resources to extend an app
Use the main form and its components

Change model-driven app custom table icons

Article • 04/29/2024

When a custom table is created, a default icon is automatically assigned. All custom tables use the same icon by default. Use custom icons to differentiate how your custom tables look. It's not possible to modify the icons assigned to system tables.

In the example here, the accounts table has an icon already assigned to it as it's a system table, however the other tables have the default icon.



Icons can be used for the following purposes with model-driven apps.

Expand table

Icon Type	Description
Table icon	An .svg, .gif, .png, or .jpg format image, 16x16 pixels in size. This article describes how to edit the table icon.
Icon for table forms	An .svg, .gif, .png, or .jpg format image, 32x32 pixels in size. More information: Enable or disable table options
App icon	Must be a scalable vector graphic (.svg) icon. More information: Manage model-driven app settings in the app designer

Note

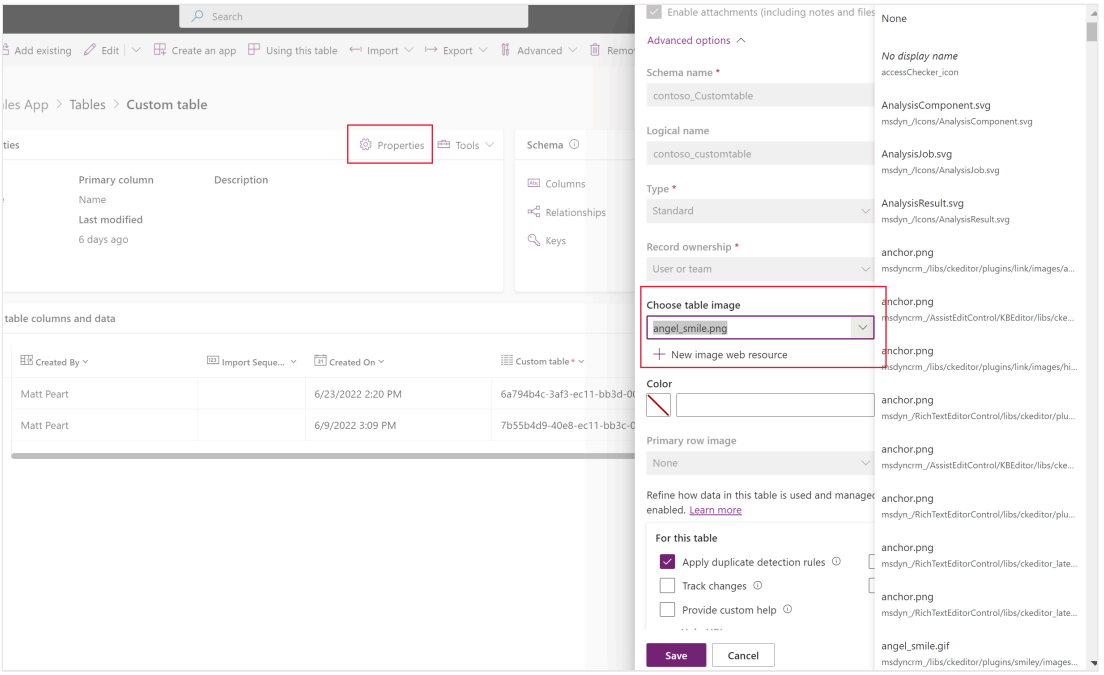
- All image files must be no more than 10 kilobytes in size.

- Setting the icon in the app designer overrides the table definition icon setting. When the table icon in app designer is set to the default icon, the icon specified in the table definition is displayed in the app.
- When a scalable vector graphic (.svg) image is used as the app icon or icon for table forms, it must have the default size set. Since SVG is an XML document, the svg element width, height and viewBox values can be edited with a text editor to define the default size for the image.
- Where possible, remove any hard coded fill color attributes and use the currentColor keyword to avoid contrast issues. More information: Image web resources

Each type of icon is stored as a web resource. Create a web resource first and then set the icons to use them. Alternatively, you can add the icon by creating a new web resource when you define the table properties.

Set the icons for a custom table from the table definition

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, and open the solution you want or create a new one. Then open the custom table from the solution. To use the default solution, select **Tables** on the left navigation pane, and then open the table you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Properties**. In the right **Edit table** pane, expand **Advanced options**.
4. Under **Choose table image**, select from the following options:
    - Select the dropdown to choose an existing web resource.
    - Select **New image web resource** to create a web resource to add the icon. More information: Create or edit web resources to extend an app



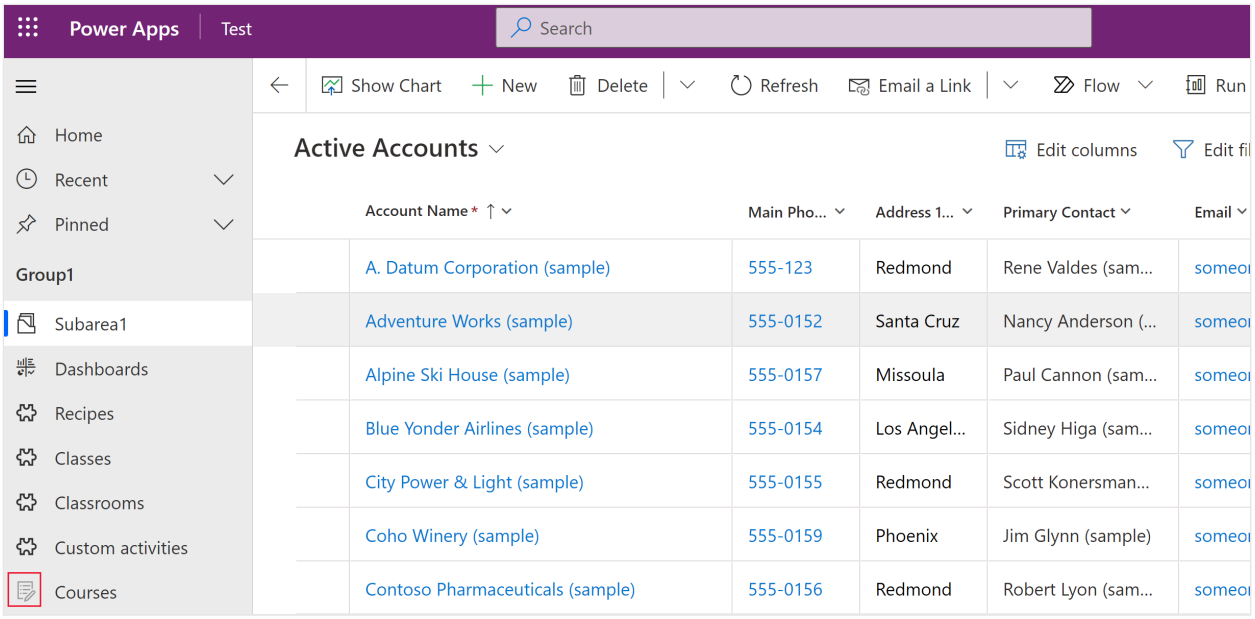
5. Select **Save**.

Set the icon for a table in app designer

1. On the app designer left pane, select the table.
2. In the right table properties pane, under **Icon**, select **Use web resource** > **Select icon**.
3. Select from the list of existing image web resources or select New to create a new one.
4. Select **Select**, and then select **Publish** to make the changes available in your app.

Review changes to your model-driven app icons

When the updates take effect, they can be reviewed by playing the app. In this case, the *Courses* table and its icon are updated.



Set the icons for a custom table (classic)

1. From Power Apps, select **Solutions** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select the unmanaged solution that contains the tables where icons need to be set.
3. Select **Switch to classic** from within the solution command bar (you might need to select **...** first). The classic solution explorer will open the solution.
4. In the classic solution explorer, expand **Entities** (tables).
5. Select the custom table that requires an updated icon.

Set table icons

1. On the command bar, select **Update Icons**.
2. In the **Select New Icons** dialog, on the **Web Client** tab, under **Icon in Web application** or **Icon for Entity Forms**, to the right of **New Icon**, select the **Browse** button .
3. Select or create the appropriate web resource, and then select **OK**.
4. In the **Unified Interface** tab, do the same for the **New Icon** column.
5. Select **OK** to close the **Select New Icons** dialog.
6. On the command bar, on the **File** menu, select **Save**.
7. When your changes are complete, publish them. Select **Publish** in the command bar while the table is selected in solution explorer.

Community tools

Iconator is a tool that XrmToolbox community developed for Power Apps. A designer can use the tool to **update** icons associated with a table in addition to introducing **new icons**.

See the Developer tools for Microsoft Dataverse article for community developed tools.

> **Note**
>
> The community tools are not a product of Microsoft and Microsoft doesn't provide support for the community tools. If there are questions relating to the tool, please contact the publisher. More Information: XrmToolBox.

Next steps

Create a custom table
Edit a table

Feedback

Was this page helpful? Yes No

Provide product feedback

Distribute a model-driven app using a solution

Summarize this article for me

Model-driven apps are distributed as solution objects. After a model-driven is created, it can be made available for other environments to use by adding the app to a solution and then exporting it.

After the exported solution (a .zip file) is successfully imported in the target environment, the app is available for use provided users have the security roles relevant to the tables or other objects in the app.

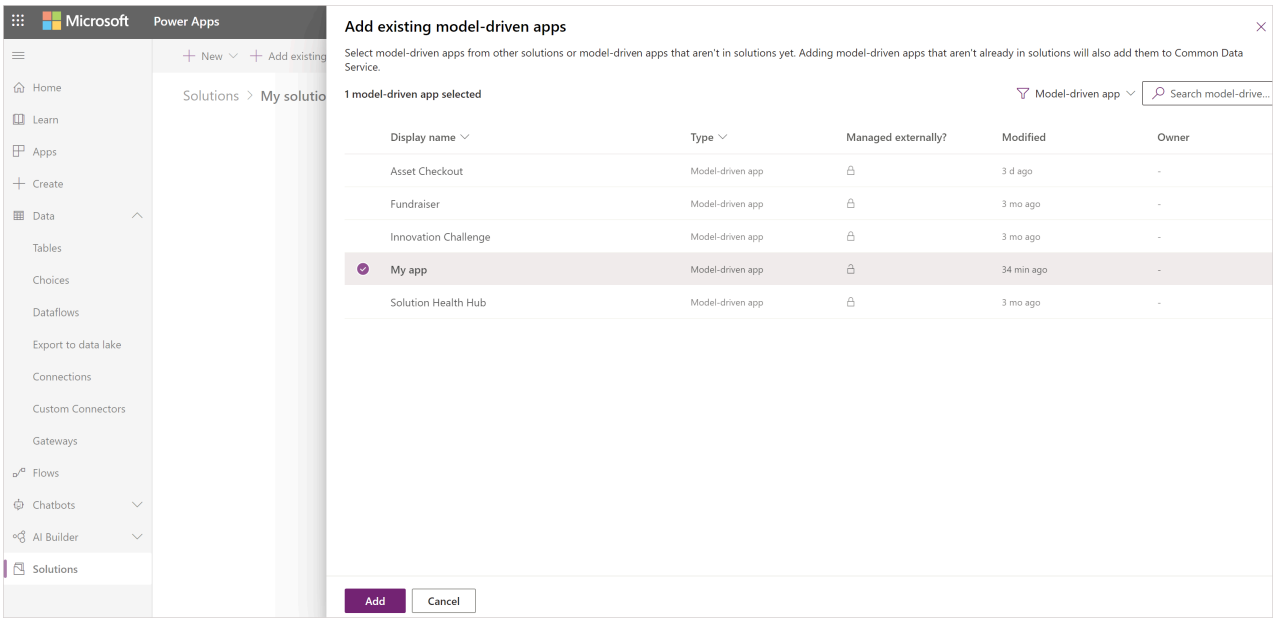
Moving solutions between environments is the basis for how you apply application lifecycle management to the products created.

This article describes how to work with solutions in a basic fashion. Go to this article for detailed guidance on working with solutions.

Add an app to a solution

To distribute an app, you create a solution so that the app can be packaged for export.

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, and then select **New solution**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Enter the properties on the **New Solution** page, and then select **Save**. More information: Create a solution
4. The **Solution** page appears. Select **Add Existing**, select **App**, select **Model-driven app**, select the app that is to be added to the solution, and then select **Add**.



5. If a **Missing Required Components** page appears we recommend selecting **Yes, include required components** to add necessary components such as tables, views, forms, charts, and site map that are part of the app. Select **OK**.

Export a solution

To distribute an app so it can be imported into other environments or made available on Microsoft Marketplace, export the solution to a zip file. Then, the zip file that contains the app and components can be imported into other environments.

1. Select **Solutions** on the left navigation pane of Power Apps. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Select the solution, and then on the toolbar select **Export solution**.

3. On the **Before you export** pane, we recommend that you **Publish**, before selecting **Next**.

4. If there a are missing dependencies, the **Manage solution dependencies** pane appears. Select **Next** to add the missing dependencies.

5. On the **Export this solution** pane, accept the incremented version number already provided or enter a different one, select **Unmanaged** or **Managed**, and then select **Export**. For more information about solution package types, go to Solutions overview.

> **Note**
>
> - Typically you export a solution as **Managed** because you want to continue to work on the project in the current environment. Unmanaged versions of your solutions should be considered your source for Microsoft Power Platform assets

and checked into your source control system. We don't recommend that you import unmanaged solutions into non-development environments.
- By default, **Run solution checker on export** is selected. We recommend that you allow solution checker to run to identify issues that might occur with the export.

6. Depending on the browser and settings, a .zip package file is built and copied to the default downloads folder. The file name of the package is based on the unique name of the solution appended with underscores and the solution version number.

> ⓘ **Note**
>
> When you export an app by using a solution, the app URL isn't exported.

Import a solution

When you receive a solution zip file that contains the app that you want to import, open the solutions component page and import the solution. When the solution has been successfully imported, the app is available in the environment.

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, and then on the toolbar select **Import solution**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select from these options for where to import the solution from:

   - **This device**. Select Browse and locate the zip file, and then choose **Next**.
   - **Pipelines**. If the solution is part of a pipeline in Power Platform select this option. More information: Use pipelines in Power Platform to deploy solutions

4. Select **Import** and wait for the solution to be imported. This can take a varying amount of time based on the complexity of the solution.

Related articles

Learn more about solutions

Change the solution publisher prefix

Last updated on 02/21/2026

Enable and use customizable help

Summarize this article for me

Customizable help lets you provide your own contextual information to model-driven app users filling in forms.

> **Note**
>
> Instead of creating and maintaining your own Help system, custom help panes and guided tasks are available that you can use to author Help that gives your Unified Interface application a custom in-product help experience that is tailored to your organization. More information: Create guided help for your Unified Interface app

With model-driven apps, you can replace the default Help with the custom Help of your choice, at the global (environment) level or table level. Custom Help makes the content exposed through the Help links more relevant for your custom or customizable tables. With a single, global URL you can override the out-of-the-box Help links for all customizable tables. Per-table URLs override the out-of-the-box Help links on grids and forms for a specific customizable table. You can include additional parameters in the URL, such as language code and table name. These parameters allow a developer to add functionality to redirect the user to a page that's relevant to their language or the table context within the application. The table level custom Help settings are solution aware, therefore you can package them as a part of a solution and transport them between environments or distribute them in solutions.

Set up customizable Help

Customizable Help can be set at the global and table levels.

> **Important**
>
> You can't use both custom help panes with customizable help at the same time. Enabling one disables the other.

Set customizable Help at the global level

People with the system administrator security role can use the settings to override default Help at the global level.

1. Sign in the Power Platform admin center. Select Manage, open the environment you want, and then select **Settings** on the command bar.
2. Expand **Product**, and then select **Features**.
3. Under **Help features**, turn **Custom help for customizable entitites** to **On**.
4. Under **Set custom Help URL**, enter the **Global custom Help URL** to your custom Help.
    - **Append Parameters to URL**. Select **Yes** to allow for parameters such as language code or table name to be appended to the **Help URL** that you specify in the table definition. More information: Append parameters to URL
5. Select **Save**.

Set customizable Help for a specific table

After you enable custom Help at the global level, system customizers can override the global Help URL for a table in the table definition.

1. Sign in to Power Apps.
2. Go to **Tables**, and then open the table you want.
3. Select **Properties**, and expand **Advanced options**.
4. In the **For this table** area, in the **Help URL** box enter the URL of your custom Help page.

Append parameters to URL

As described previously, to allow for the appending of parameters to the **Help URL** for the table definition, set **Append Parameters to URL** in the **System Settings** > **General** tab to **Yes**.

Examples of the parameters that can be appended to the URL:

- User Language Code: userlcid
- Table Name: table
- Entry Point: hierarchy chart or form
- Form ID: formid

Last updated on 03/04/2026

What are performance insights? (preview)

Article • 07/10/2024

> **Important**
>
> This feature is undergoing significant improvements and until the new version becomes available the current feature is disabled.

[This topic is pre-release documentation and is subject to change.]

Performance insights are a self-service tool for enterprise app makers that analyzes runtime user data and provides a prioritized list of recommendations to help improve the performance of model-driven apps. This feature provides a daily set of analytic insights related to the performance of a Power Apps model-driven or customer engagement app, such as Dynamics 365 Sales or Dynamics 365 Service, with recommendations and actionable items. Enterprise app makers can view detailed performance insights at an app-level in the Power Apps portal.

How to access?

1. Sign in to Power Apps.
2. On the left navigation pane, select **Apps**, and then select a model-driven app.
3. Use the **...** context menu or command bar to select **Performance**.

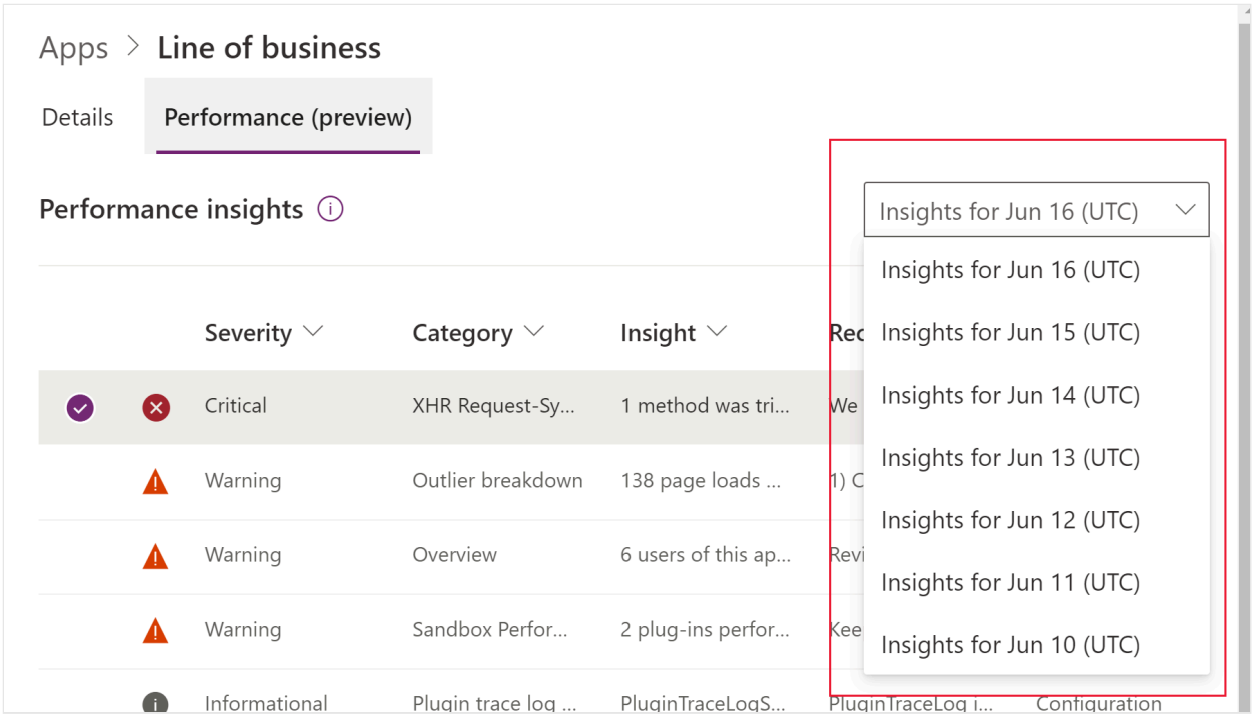
If you want to switch the environment where your app is deployed, you can select environments in the top-right corner of the page using the **Environment** selector. Alternatively, you can navigate to performance insights from a model-driven app's context menu from the **Solutions** area.

> **Important**
>
> Since recommendations are generated using user data, we recommend that you view performance insights from an environment where the app will be used, such as a production environment.

How insights are generated

Performance insights are generated based on collected user data of your model-driven app every 24 hours. When end-users use an app, key user data recorded by the Power Apps platform is stored. The performance insights engine uses this data and analyzes it to generate insights and recommendations related to performance enhancements.

Performance insights are available for all model-driven apps in your selected environment, provided there is recorded user data. You can view insights history for the previous seven days, as long as there was usage on the selected date, by selecting a date in the list on the **Performance** tab.



The performance insights engine generates insights over a 24-hour period on a selected date in the universal time coordinated (UTC) time zone. Insights for the previous day, from 00:00 to 23:59 UTC, are generated around 02:00 to 05:00 UTC on a given day. Therefore, insights and recommendations for usage user data recorded the previous day will be available in the morning in the UTC time zone. Time of availability will vary based on your local time zone and its difference from UTC.

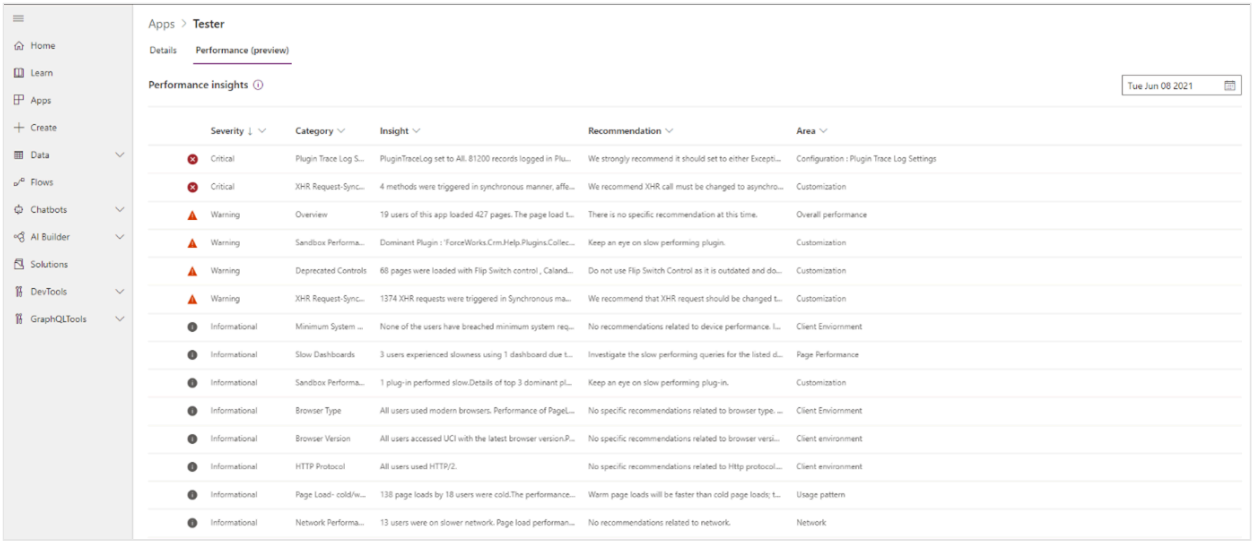
For example, insights for the 24-hour period that spans June 8 from 00:00 to 23:59 UTC are generated between 02:00 to 05:00 UTC on June 9. Therefore, insights and recommendations for user data recorded on June 8 UTC will be available the morning of June 9 UTC.

Insights can be sorted by severity – ranging from **Informational**, **Warning**, to **Critical**. Severity is weighted depending on the affect of the findings on the overall performance. Severity is determined by several factors such as the volume of usage, performance of

slow page loads, and outliers. Critical items exceed certain thresholds and we recommend you address those first.

Insights are generated for several categories from different areas including client environment, customizations, page performance, and network. For example, if several users are using the app on an older version of a non-recommended browser, performance will be slower in general. Hence, makers will see an insight related to browser type and version in the insight grid.

Performance insights provide a prioritized list of recommendations to help makers improve the performance of model-driven apps. You can select each row in the grid to view detailed information for each category.



Explanation of performance insights

The performance insights grid has these columns.

Expand table

Column	Description
Severity	An insight can be categorized as **Informational**, **Warning**, or **Critical**. Severity is determined by the performance page load time and the number of end-users impacted by an event.
Area	This area defines the aspects of analyzing user data by the engine of Performance Insights. Includes client environment, network, and usage patterns.
Category	The theme under which the specific insight falls. Categories are grouped under Areas.

Column	Description
Motivation	Gives an overview of why a particular insight could be impacting the overall performance of your app.
Insight	Details the findings related to a specific category and outlines the performance impact.
Recommendation	Outlines the actions you can take.
How to improve	Links to the documentation for a specific category, which could contain further information or external resources to help you mitigate your issue and improve performance.
Data	Evidence data to support the findings in the insight section.
Insight ID	Identifier for a specific insight category.

What aspects does it check?

Performance of a model-driven app could be affected by multiple factors. Horizontally, the app could be affected by client environment, network, usage pattern, customizations, and configurations. Vertically, web server or back-end SQL Server could affect performance on the client side.

Therefore, it is beneficial to identify what may cause slowness of page loads or operations. The performance insight engine analyzes collected user data and corresponds it with the categories.

In some scenarios, there could be an obvious reason for performance issues. In other cases, there may be several reasons identified from the categories that could explain the slowness.

The following table summarizes what areas and categories the performance insights engine checks.

Area

Insight category

Description

Overall performance

Overview

It provides an overall summary of the performance of a selected app. You can see the performance of app briefly. More information: Overall performance

Client environment

Minimum system requirements

Users should at least use the system meeting the minimum specifications. This insight tells makers how many users meet such criteria if user collected and others not. More information: Minimum system requirements

Client environment

Browser type

The performance will vary depending on browser type. This insight indicates whether users run a modern browser or a non-recommended Internet Explorer (IE) browser. The performance of Power Apps in IE is noticeably slower compared to modern browsers. More information: Browser type

Client environment

Browser version

Although users run the modern browsers, they could experience slow performance if they are using a browser running on an older version. In general, staying on the latest version would be recommended. More information: Browser version

Client environment

HTTP protocol

Power Apps supports the HTTP/2 protocol. Apps generally perform better with the HTTP/2 protocol compared to HTTP/1.1. One of the major limitations of HTTP/1.1 is the limited concurrent connections to the same domain, which might cause apps to run slow. More information: HTTP protocol

Network

Network performance

Network latency, throughput, and bandwidth are key factors affecting the performance of web applications. Apps can perform differently depending on network performance

as some contents should be downloading. More information: Network performance

Usage pattern

Page load type.

Page load type can affect app performance. Warm (cached) page loads are faster than cold (not cached) page loads since resources load out of the cache. More information: Page load type

Page performance

Slow dashboard

One of the reasons why a dashboard might be slow to load is an associated query of a chart or a tile that is performing poorly. If the query executes over a large set of data, it could cause a slow response back. Apart from checking the number of chart controls and tiles on the dashboard, we recommend checking associated queries for both charts or tiles. More information: Slow dashboards

Customization

XHR call type

Makers can customize the model-driven app by adding a custom function making XMLHttpRequest into an event. If the XMLHttpRequest (XHR) call happens synchronously, other tasks must wait until the XHR call is completed. We strongly recommend that you make the XHR call in your custom functions asynchronous. More information: XML HTTP Request (XHR) call type

Customization

Deprecated controls

Some controls like the flip switch and calendar V1 controls are deprecated. Makers should always use modern controls. More information: Deprecated controls

Customization

Sandbox performance

Makers can build their own plug-in. Depending on implemented patterns, there are many aspects to check to confirm the plug-in methods perform well, such as timeout events. This insight checks the plug-in's timeout to see whether it was caused by the

timeout of the external endpoint calls the maker implemented or not. More information: Sandbox performance - Dominant plug-ins

Customization

Saved query

While building a solution, a maker can build, and reuse a saved query. If the saved query has been created improperly, it could make the query run slow. Consequently, any forms or views using the problematic saved query might suffer from slow performance.

Configuration

Plugin trace log setting

Administrators can enable plug-in trace logs for a Dataverse environment from **None** to **All**. It is beneficial to check the plug-in trace logs in the development phase. However, for production environments, we recommend setting the feature to either **Exception** or **None** to minimize intensive logging activity that can degrade app performance. More information: Plug-in trace log setting

See also

Understanding insights

Feedback

Was this page helpful? Yes No

Provide product feedback

Understanding insights (preview)

Article • 04/14/2022

[This topic is pre-release documentation and is subject to change.]

Performance insights distributes insights into the following categories:

- Overall performance
- Client environment
- Usage pattern
- Page performance
- Customization
- Configuration
- Network As the performance of an app could be impacted by various factors, performance insights categorizes areas to check how much overhead of performance get caused by.

Overall performance

This insight summarizes the overall performance of your app as an executive summary under the Insight ID Perf.Summary.Overview.

In this insight, you can see the overall performance of your app based on the severity.

- **Critical**: Indicates poor performance.
- **Warning**: Indicates performance could be improved.
- **Informational**: Indicates good performance.

How to improve

When it comes time to optimize the app, you can look at the detailed insights from the client, network, customization categories as well as plugins, savedQeury, and settings. Some actionable items can be derived by reviewing those insights.

Client environment

When users experience a Power Apps app on their devices, several factors can affect performance such as browser type, browser version, and specification of hardware. In this section, you can see what insights check client environments.

Browser type

Insight ID: Perf.Environment.Client.Browser.Type

Motivation

Certain web browser types can impact the performance of your app. Using unsupported or non-modern browsers can lead to slow performance. This insight provides the performance implications of different browsers, especially non-recommended browsers. For example, Power Apps has deprecated it's support of Internet Explorer.

How to improve

If you have users on old browsers such as Internet Explorer, switch to a modern Chromium-based browser. We recommend that users run a modern browser, such as Microsoft Edge or Google Chrome.

> **Note**
>
> Some legacy applications leveraging NPAPI will only work on Internet Explorer.

Browser version

Insight ID: Perf.Environment.Client.Browser.Version

Motivation

This insight checks how many users are using your app from an old version of a browser. Even when users run modern browsers, and not non-recommended browser types like Internet Explorer, older versions of browsers perform slower.

How to improve

Users should regularly update the browser to the latest version. Enterprise customers can apply a group policy to be on a specific version. As Unified Service Desk (USD) also uses the default browser setting of the computer, it also requires checking the default browser type and version.

Minimum system requirements

Insight ID: Perf.Environment.Device.MimimumRequirements

Motivation

This insight checks whether the user's environment meets minimum system requirements. You can check the web application requirements to see what are the minimum system requirements depending on the app type.

In general, some activities like rendering, scripting, and downloading contents happen on the client side. Meeting the minimum system requirements is necessary for such activities.

How to improve

Users should use the hardware that meets or exceeds the minimum system requirements for Power Apps.

HTTP protocol

Insight ID: Perf.Environment.Client.Browser.HttpProtocol

Motivation

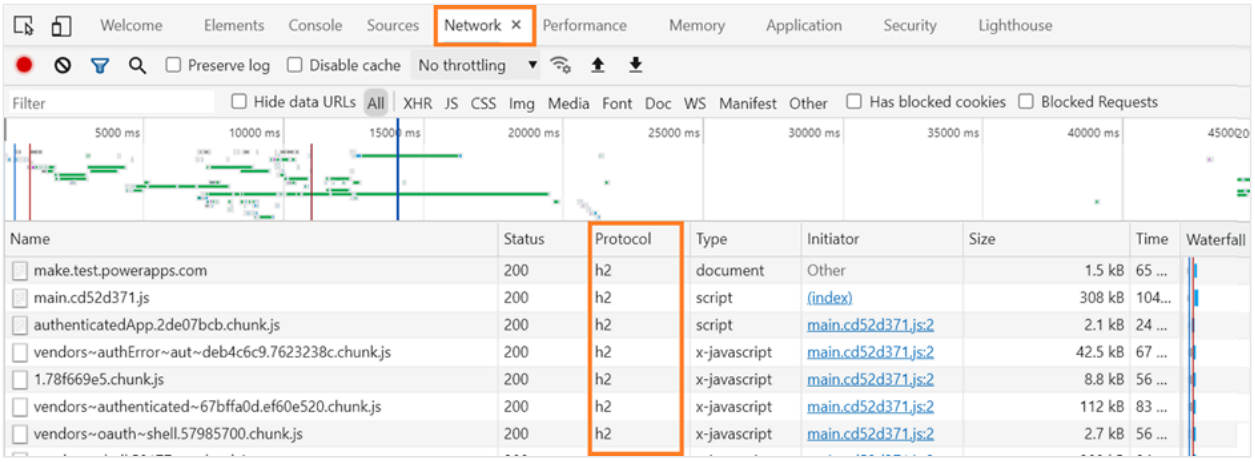
Power Apps platform supports HTTP/2. However, if your app is using the HTTP/1.1 protocol for XMLHttpRequest (XHR) requests onto Power Apps, it might cause slow performance due to the concurrent limitation of requests with the HTTP/1.1 protocol.

How to improve

If this insight identified some users who are using the HTTP/1.1 protocol, we strongly recommend that these user's client support the HTTP/2 protocol.

Several configurations and network infrastructure can block the HTTP/2 protocol, such as a VPN network, proxy server, or device internet option settings.

Users can check what protocol has been used from a development tool included with the browser. In the figure below, network calls occurred over HTTP/2.



If the network protocol trace indicates HTTP/1.1, it might be because of the following:

- Internet settings: The Windows Internet Option **Advanced** tab in Control Panel **Use HTTP2** and **Use TLS 1.2** options aren't enabled.
- VPN and proxy: Although the Windows Internet Option is set to use HTTP2 and TLS 1.2, the browser might fall back when a VPN or Proxy doesn't support the more recent protocols.

Usage pattern

This category analyzes the type of page loads. A warm page load renders the page using caches and existing DOM objects, while a cold page load renders the page fresh by downloading resources when required. Although users will not distinguish the type of page load, this insight analyzes and provides recommendations depending on what type of page loads occur on the client.

Page load type

Insight ID: Perf.Performance.PageLoadType

Motivation

Warm page loads are faster than cold page loads since necessary resources load out of local caches.

> ⓘ **Note**
>
> When a user opens a form from new tab or new tab in a browser, it is considered a cold page load. When a user opens other forms in the app within the active tab of a browser, it is considered a warm page load.

How to improve

To experience warm page loads for faster performance, minimize opening new tabs or browser windows. Try to keep activities within a single tab instead of opening new tabs or browser windows. We also recommend not to run the browser in InPrivate or incognito mode.

Page performance

Many first party model-driven apps consist of a dashboard, views (EntityList), and form when it comes to page type. By default, users load a dashboard, although app makers and administrators can change this. If a dashboard contains many charts and tiles, it might cause the dashboard to load slowly. Similarly, if EntityList and forms get customized to add many columns and display many records, it can also cause the page to load slow. Hence, checking the performance per page and per table can be beneficial because page load performance can have different root causes.

In this section, you can see several insights related to page performance.

Slow dashboards

Insight ID: Perf.ModelDriven.Page.Dashboard.SlowSQLQuery

Motivation

Slow SQL queries or using too many charts and tiles in a dashboard can cause poor performance of the dashboard. This insight points to the dashboards that are affected by slow SQL queries. When this insight is recorded, the **Details** pane includes the dashboard ID for each dashboard included in the insight.

How to improve

Here's how to look up the name of dashboard using the dashboard ID. Then, you can determine which dashboards to consider for redesign.

1. Go to your model-driven app, such as *https://contoso.crm.dynamics.com*.
2. Modify the URL as shown in this example (https://contoso.dynamics.com/api/data/v9.1/systemforms[DashboardId]/name) by appending *api/data/v9.1/systemforms[DashboardId]/name* to the app URL.

3. You will receive OData request similar to the below. **Agent Dashboard** displayed below represents the user-friendly name of the given dashboard ID.

`{"@odata.context":https://contoso.crm.dynamics.com/api/data/v9.1/$metadata#systemforms(2ff4a8cf-378b-e811-a964-000d3a30dc0a)/name,"value":"Contoso - Agent Dashboard"}`

Synchronous plug-ins with slow external calls

Insight ID: Perf.Sandbox.Performance.Plug-ins.ExternalCall

Plug-ins and custom workflow activities can access web services (external endpoints) via HTTP and HTTPS protocols. If these external services perform slowly, the plug-in itself will timeout or perform slowly.

Motivation

This insight checks the performance of the external endpoints and detects plug-ins in your app that are impacted by the slow external calls.

How to improve

- Set KeepAlive to false when interacting with external hosts in a plug-in.
- Set Timeout explicitly when making external calls in a plug-in.

More information: Access external web services (Microsoft Dataverse) - Power Apps | Microsoft Docs.

Customization

Makers can do many different customizations with model-driven apps, such as:

- Ingest custom JavaScript functions to enable events on the client.
- Build and deploy plug-ins used to execute custom logic.
- Define and store custom tables and data.
- Define dependent components for both custom and standard tables, such as forms and views.

From a performance perspective, all of these customizations might cause poor app response in situations where the customization doesn't follow best practices and recommendations. Makers can run Solution Checker to validate their customizations during the development phase.

The following insights also provide analytic results out of runtime user data of your customization.

XML HTTP Request (XHR) call type

Insight ID: Perf.ModelDriven.Customization.Client.Script.XMLHttpRequestType

Synchronous XMLHttpRequest calls can cause severe performance issues for end users, especially when the network is slow or there are multiple calls that need to be made. The browser freezes up and the end user is frustrated when they can't click, scroll, or interact with the page.

This insight reveals whether there are synchronous methods and indicates the related performance.

Motivation

Synchronous XHR calls will block the browser from performing more work as the browser must wait for the synchronous call to complete causing the page to slow down or freeze completely.

How to improve

We recommend that you change the top methods mentioned in the **Data Section** of the insight from synchronous to asynchronous. More information: Turbocharge your model-driven apps by transitioning away from synchronous requests

Deprecated controls

Insight ID: Perf.Customization.Controls.Deprecated

Some older controls for model-driven apps like Flip Switch, Calendar Control (V1), Linear Slider, Radial Knob, Arc Knob, Linear Gauge; along with the Website Preview control, MultiSelectPicklistControl (V1), and the Flip Label are deprecated. Some of these controls can be replaced with the new controls, which are more in line with the modern web and mobile in mind. More information: New Model Driven Apps controls, deprecation of old controls

Motivation

Using outdated controls can cause performance, reliability, and accessibility issues. Moreover, some of the limitations in these deprecated controls have been resolved with the new controls. For example, the Toggle control and Calendar Control (V2) use Microsoft Fluent UI.

How to improve

- Use the Toggle control as a replacement for Flip Label and Flip Switch.
- Use Calendar Control (V2) as a replacement for Calendar Control (V1).
- Evaluate other deprecated controls to determine if they are still useful within existing forms.

Notice that there are few significant design changes between the deprecated version and the new controls.

For more information about the deprecated controls, go to Model-driven app controls deprecation.

Sandbox performance - dominant plug-ins

Insight ID: Perf.Sandbox.Performance.Plug-ins.Dominant

This insight will help us identify the dominant plug-in, or in other words, the one that's most used. It will also indicate if any of the dominantly used plug-ins performed slow with a plugin execution time greater than 100 milliseconds in the 95th percentile. This insight will list up to three dominant plug-ins.

Motivation

Slow dominant plug-ins affect performance. These plug-ins should be investigated.

How to improve

Investigate slow performing plug-ins. Check out the best practices regarding plug-in and workflow development.

To further investigate the slow plug-in, you can set the **Plug-in trace log** settings to **All** in your development or test environment and determine where the delay is. However, don't forget to disable the setting before going to production. More information: Tracing and logging

Investigate slow performing plug-ins. Some of the reasons for slow plug-ins are described here:

- Associated SQL queries performed slow, hence the plug-in execution time increased.
- Follow the single responsibility principle for your plug-in and don't make transactions with significant transaction boundaries.
- Plug-in might be making some external calls, which are slow.
- Plug-in logic isn't optimized for multi-threading environments. Check your code.

To further investigate the slow plug-in, you can set the **Plug-in trace log** settings to **All** in your development or test environment and determine where the delay is. Don't forget to disable the setting before going to production. More information: Tracing and logging

Saved query with leading wildcard

Insight ID: Perf.ModelDriven.Customization.SavedQuery.LeadingWildCard

Leading wildcards are *like* or *not like* conditions that use a wildcard (%) at the start of a search string. An example of a poorly written request is:

XML

```
<fetch>
    <entity name="account">
        <attribute name="accountid" />
        <attribute name="accountnumber" />
        <filter type="and">
            <condition attribute="accountnumber" operator="like"
value="%124" />
        </filter>
    </entity>
</fetch>
```

Motivation

A leading wildcard character (%) in a saved query can cause the query to timeout or perform slowly. This insight points to such slow saved queries with leading wildcards.

How to improve

Avoid using leading wildcards. In the search key, these are translated to "contains" in SQL Server, which won't take the advantage of index seek but will do a scan. If it is

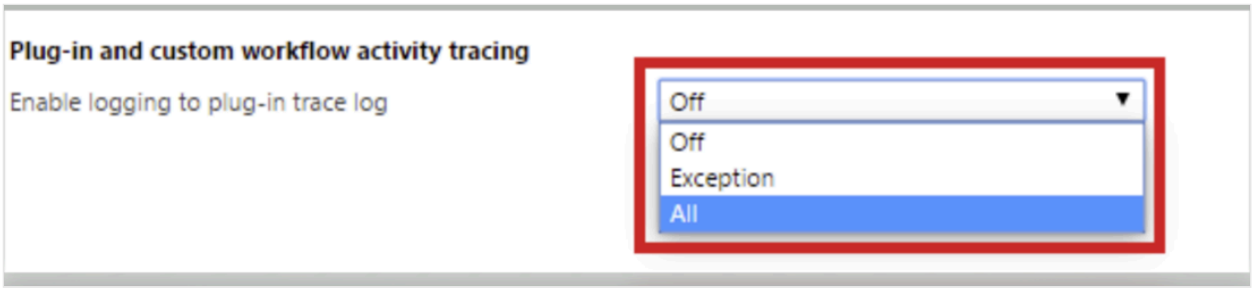
necessary to use a leading wildcard, limit the scope of search by including other conditions. Note that it is ok to use trailing wildcards (%) at the end of search strings.

Configuration

Plug-in trace log setting

Insight ID: Perf.Sandbox.Configuration.PluginTraceSettings

Makers can debug their plug-ins through plug-in trace logs. The Dataverse admins can set plug-in and custom workflow activity tracing to **Off**, **Exception**, or **All**.



Motivation

The plug-in trace log should be set to **All** only when you're debugging or tweaking the plug-in. High volume of trace logging can cause I/O overhead with SQL Server. Moreover, deletion of the plug-in trace log might cause blocks or waits with SQL Server.

How to improve

In your production instance if this setting is **All** and the volume of logs generated by your plugin is high, then consider changing it to Exception.

To change the setting, go to **Settings** > **Administration** > **System Settings** > **Customization** tab. More information: Logging and tracing

Network

Network performance

Network latency and throughput are important factors that affect end user's experience. Users with high latency and low throughput will more likely experience slow

performance while accessing Unified Interface. This insight tells us how many users are on a poor performing network and what their performance was like.

Motivation

Poor network configuration will affect app performance.

How to improve

If there are many users on the network and the performance is poor, we recommend the users switch to a better performing network.

See also

What are performance insights?

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Monitor to get insights and recommendations for your app

Summarize this article for me

Monitor in Power Apps offers insights and recommendations to enhance app performance. It provides makers with detailed information on their app's performance and actionable steps to improve their apps.

Prerequisites

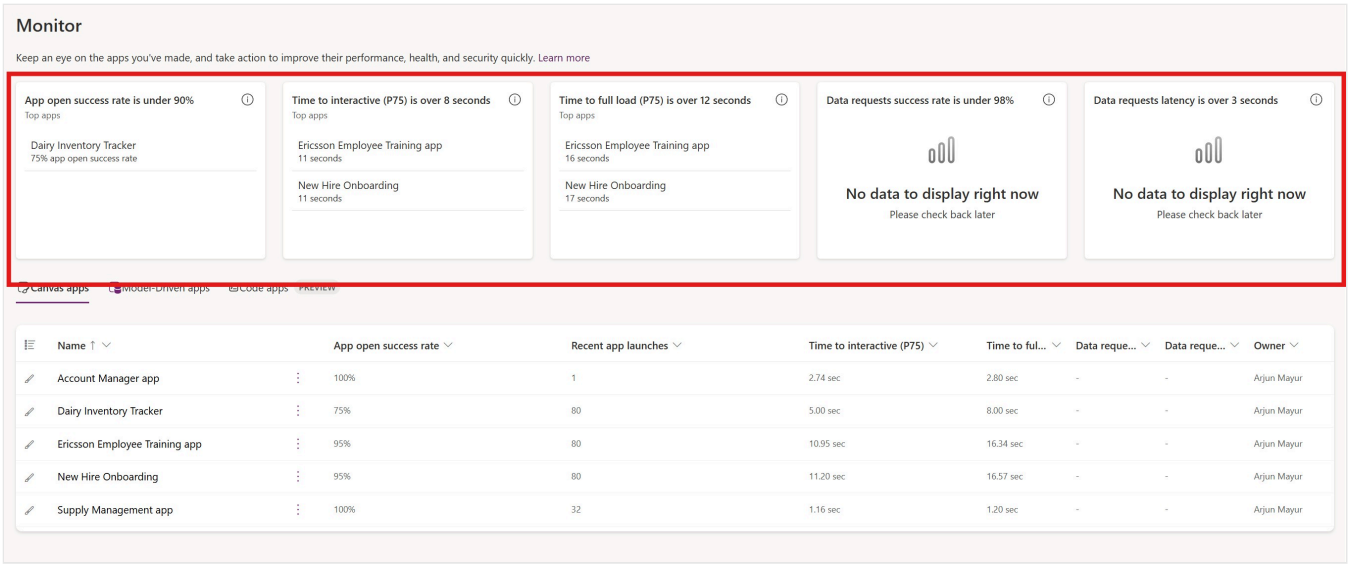
To use this feature, your administrator must enable certain analytics and monitoring settings in the Power Platform Admin Center. For more information, see Monitoring experiences for makers. When you sign in to Power Apps and open Monitor, make sure you've selected the correct environment and that you're a member of that environment with an assigned role (for example, Basic User). Otherwise, you won't be able to access Monitor.

Open and pin Monitor

1. Sign in to Power Apps.
2. On the left navigation, select **Apps**.
3. On the command bar, select **Monitor**.
4. On the left navigation pane, select the  pin button next to **Monitor**.

Use Monitor

The **Monitor** page displays metric cards at the top, highlighting the top three underperforming apps for each metric. If all apps perform well on a specific metric, the card doesn't list any apps. Use the tabs under the cards to switch between viewing your model-driven, canvas, and code apps.



Expand table

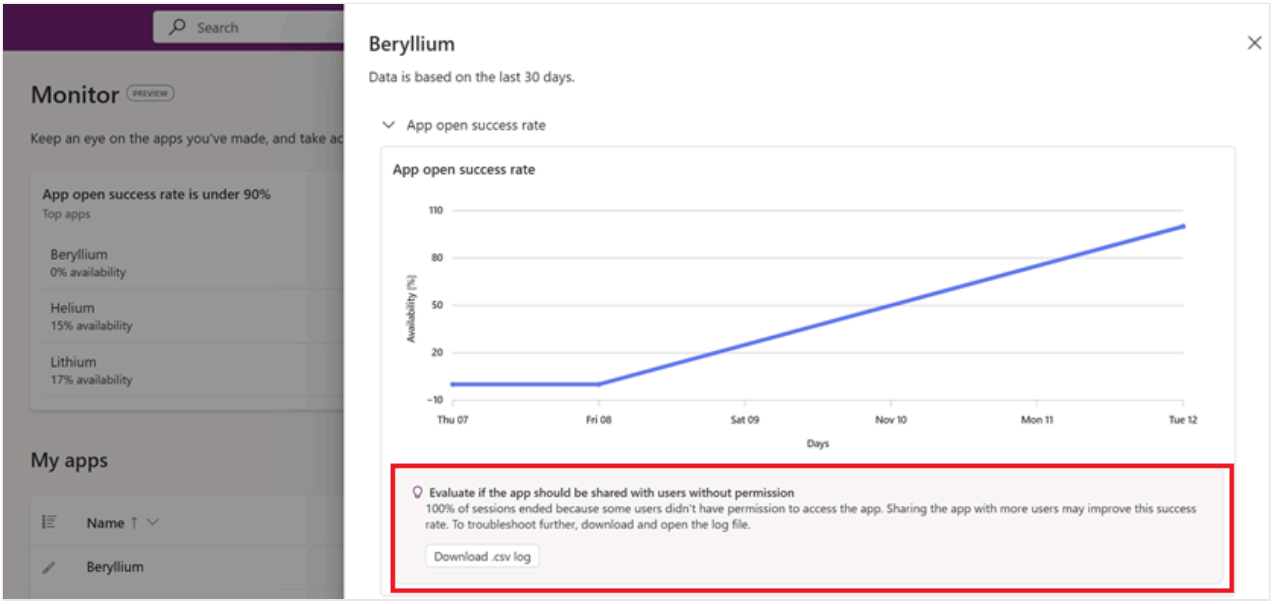
Metric	Definition	Available for Canvas apps	Available for Model-driven apps	Available for Code apps
App open success rate	Indicates the percentage of app launch attempts that successfully allowed end-users to access the app. This metric is calculated once every 24 hours.	Yes	Yes	Yes
App session count	The number of distinct user sessions in an app in one day. A session begins when a user opens the app and ends after a period of inactivity or when the app is closed.	Yes	Yes	Yes
Time to interactive (TTI)	Measures the time end-users wait to access the first screen of an app. It doesn't include the time users wait to see all data presented on the first screen, such as filling in rows of a gallery. This metric is calculated every 24 hours and is measured using percentiles, with the 75th percentile being reported.	Yes	No	Yes
Time to full load (TTFL)	Measures the time end-users wait to see all data displayed on the first screen of an app. In the app launch lifecycle, this metric follows the **Time to interactive** phase. This metric is calculated every 24 hours and is measured using percentiles, with the 75th percentile being reported.	Yes	No	No
Data request success rate	A percentage that describes how often data requests coming from an app are successful	Yes	No	No

Metric	Definition	Available for Canvas apps	Available for Model-driven apps	Available for Code apps
Data request latency	The average time (in seconds) it takes for all data requests made by the app to complete during a session.	Yes	No	No
Row summary dwell time	The time (in seconds) that end users spend on the expanded AI row summary of a main form.	No	Yes	No

View metrics and recommendations for an app

When you select an app on the **Monitor** page, a side pane opens and displays more data for each metric, including a data chart for the last 30 days.

1. On **Monitor** page, select an app.
2. To view recommendations for an app, select the app's name to open the side panel. The chart shows a callout with a recommended action for improving the metric.



> **Important**
>
> Recommendations are only available in Managed Environments. Contact your admin to enable Managed Environments for access to recommendations.

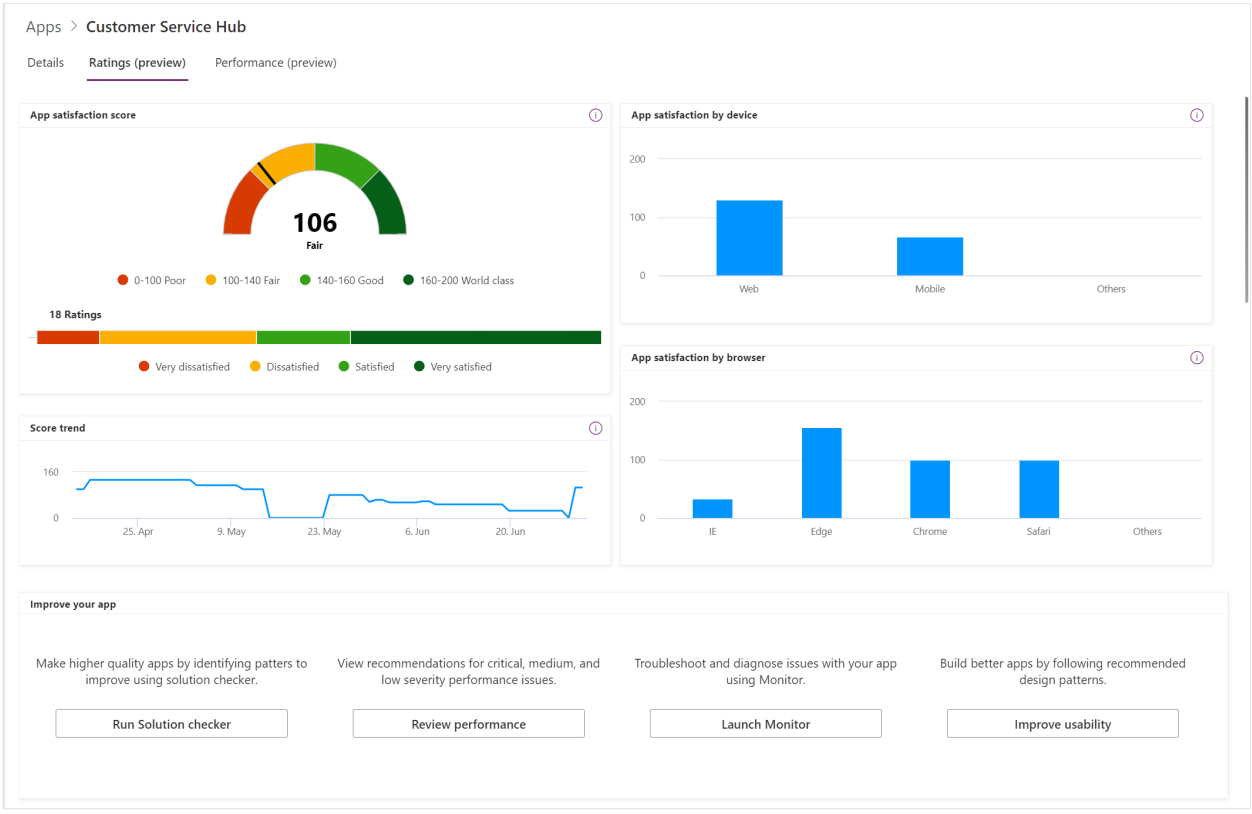
Last updated on 03/05/2026

What are app ratings? (preview)

Article • 07/06/2022

[This topic is pre-release documentation and is subject to change.]

Microsoft Power Apps takes feedback from app users on how satisfied they are with their app experience. With the app ratings feature, makers and administrators view the app's satisfaction score and comments provided by users. The collection of feedback is compliant with all international privacy laws. The feature also seamlessly integrates with solution checker, performance insights, and live monitoring, which can help makers improve their app.



> **Note**
>
> - This feature is currently only available for model-driven Power Apps.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback. More information: **Power Apps preview program**

This feature has the following benefits:

- Provides a view of the app satisfaction score on a scale of 0 to 200.

- Provides a score trend over the last 90 days.
- Displays different satisfaction scores based on web or mobile device and browser.

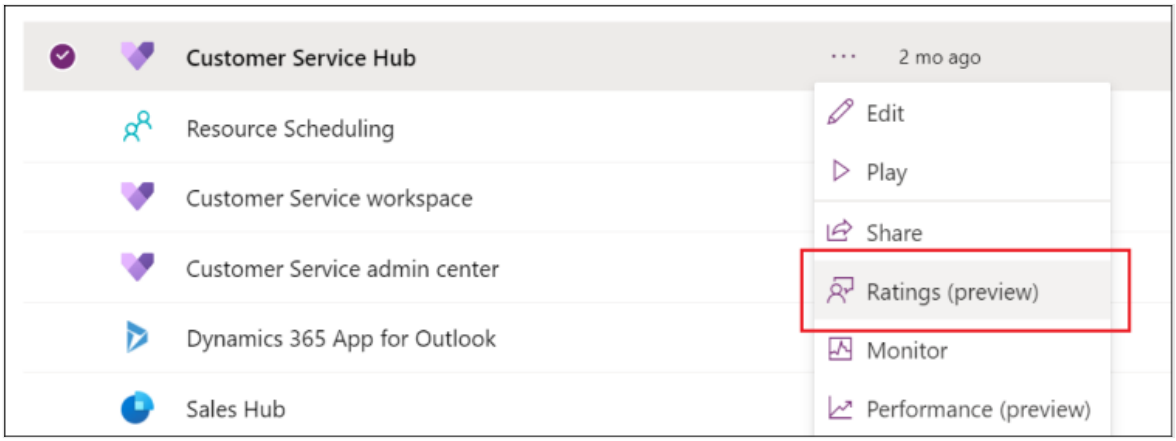
The report for current score and comments can be exported for each app in the app ratings dashboard to share with others.

Access the ratings dashboard

1. Sign in to Power Apps.

2. On the left navigation pane, select **Apps**, and then select a model-driven app.

3. Select the ... context menu, and then select **Ratings (preview)**.



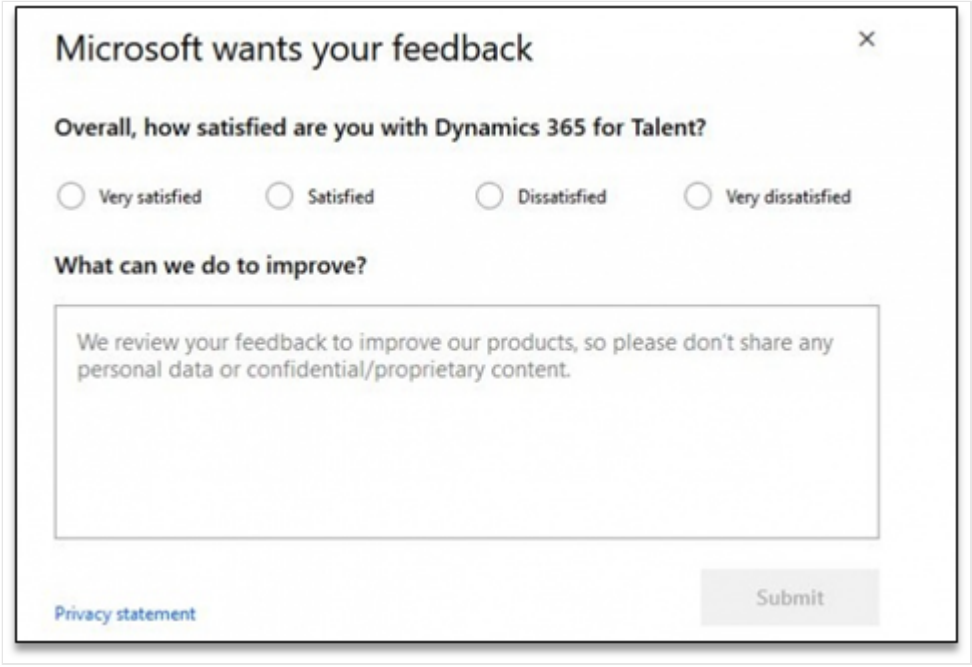
How app user satisfaction is collected

Based on the user's activity inside the app, Power Apps occasionally prompts users to score the app with the following questions:

1. How satisfied are you with (name of the app)?

    - Very satisfied
    - Satisfied
    - Dissatisfied
    - Very Dissatisfied

2. What can we do to improve?



Based on the feedback, an app satisfaction score and comments are displayed in the ratings dashboard.

The following guidelines are used to prompt an app user with the feedback dialog:

- If a user has at least three active days during the first 14 days since sign-up, the user is prompted.
- If the user has six active days in the last (recent) 90 days, the user is prompted.
- No users are prompted more than once in 90 days for the same app.

App satisfaction score

The app satisfaction score is calculated by taking the responses with an answer "Very Satisfied" and subtracting the percentage of the responses with an answer "Dissatisfied" or "Very Dissatisfied". The total NSAT score is then calculated like this:

> 100 + (%Very Satisfied) - (%Dissatisfied) - (%Very Dissatisfied)

App satisfaction is reported on a scale of 0 to 200, where:

- 1 to 100 is a poor score.
- 100 to 140 is a fair score.
- 140 to 160 is a good score.
- 160 to 200 is a world class score.

Score trend

The app satisfaction score for each day is calculated based on total responses received in the previous 28 days. Trends from 1 to maximum 90 days are displayed depending on the app's first publish and first response date. This score provides a visualization on how the satisfaction score is trending over a period. We recommend you use this feature to analyze subsequent updates to the app and follow how those updates have helped in improving the general experience with users.

App satisfaction by device

The app score is separated by device types – web, mobile, or others. For tablets, the score is calculated as mobile. The Others option categorizes the score for unknown devices.

App satisfaction by browser

App ratings also display a comparison across different browsers. The feature can identify different types of browsers including Microsoft Edge, IE, Chrome, Mozilla, and Opera. For browsers that can't be identified, the score is attributed under the Others category.

Improve your application

With the public preview release, the app ratings provides the following direct links to take actions to improve your app:

1. **Solution Checker** – This link takes you to a list of solutions to run solution checker if applicable. Notice that makers or admins must manually identify the solution and run solution checker to identify issues.
2. **Performance Insights** – This tool provides direct insights on the respective app's performance and what makers can do to improve the app performance issues.
3. **Monitor** – This tool helps makers troubleshoot and diagnose issues with live monitoring of a user's session.
4. **Usability** – This action redirects to documentation of best Power Apps provided recommendations on design and patterns.

How the feedback data is stored

A new system table in Microsoft Dataverse called **User Rating** was added to all Power Apps environments that include Dataverse. This table stores the rating score and comments from each user feedback. This is an organization owned table with default

create, read, and delete rights provided to system administrators and customizers. The environment maker role has only read privilege on the table.

> **Note**
>
> 1. Only users with at least read privileges on the User Rating table can view app ratings data.
> 2. By default update on User Rating table records is unavailable for all other security roles.

Administrators can also share the app score and comments to external stakeholders by creating Power BI dashboards on the User Rating table. This can help convey app ratings data to makers outside organizations who don't have access to production environments.

Model-driven app frequently asked questions

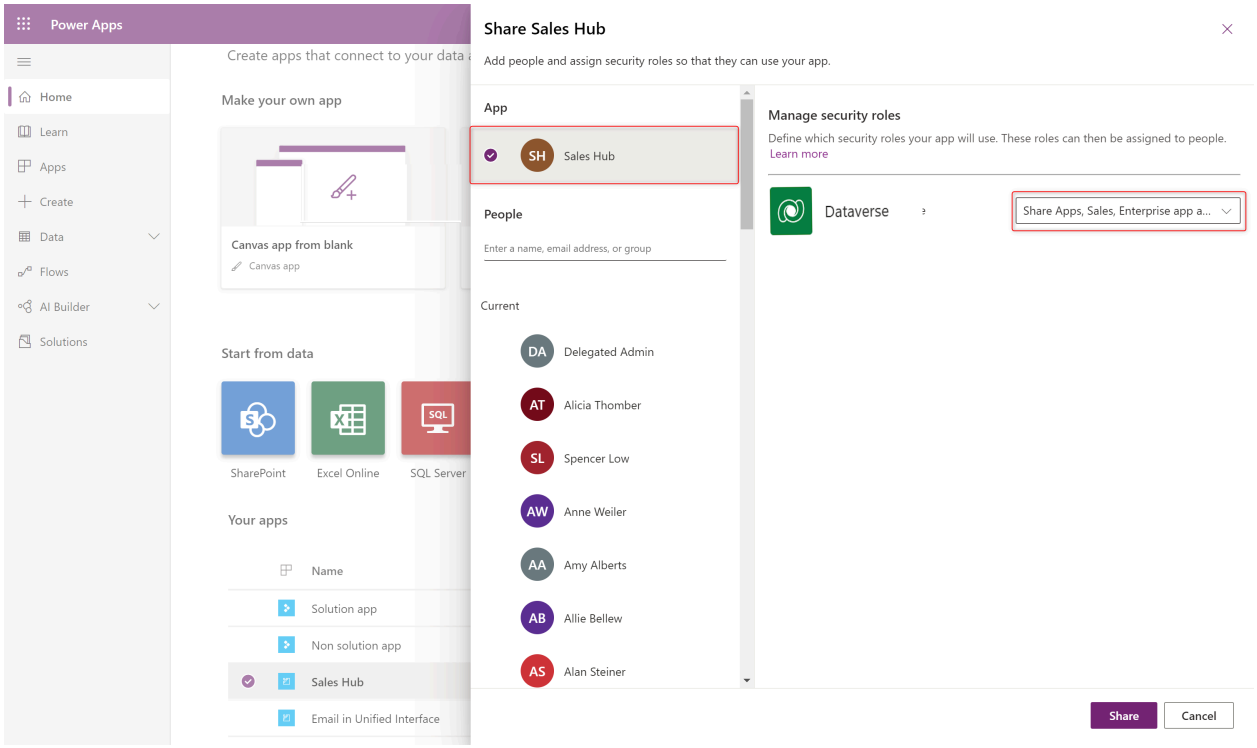
Article • 01/27/2025

These frequently asked questions (FAQ) can help you resolve issues that might occur when you work with model-driven apps.

How do I run a model-driven app?

In order to play an app you need to have a security role assigned to you and the URL to the app.

To share the app, select the app, and then on the command bar select ... > **Share**. The app sharing page as shown below appears for you to assign a security role.



You can find the app URL by going to the **Apps** section Power Apps, select the app, and then select **Details** on the command bar. More information: Sharing the app

To play the app, open the URL in your browser.

For more information, see sharing model-driven apps

Why can't I create a model-driven app?

The most common reasons you can't create model-driven apps are:

- Licensing
- Security
- Environment configuration

A Power Apps or Dynamics 365 license is required to create a model-driven app. More information: Licensing requirements for tables in the Power Apps Licensing Guide and Understanding Power Platform Licensing

Make sure you have sufficient privilege to create model-driven apps in your environment. More information:

- About predefined security roles
- Environment permissions

Additionally, you might be working with a restricted table. The following Dynamics 365 customer engagement applications include restricted tables:

- Dynamics 365 Sales
- Dynamics 365 Customer Service
- Dynamics 365 Field Service
- Dynamics 365 Marketing
- Dynamics 365 Project Service Automation

Make sure you have the appropriate Dynamics 365 application license when working with these tables. More information: Restricted tables

Why can't you see my model-driven app?

Make sure you have the appropriate privileges on the app. Work with your app maker or admin to ensure you have the proper security roles for the app. More information:

- Privileges required to view and access apps
- Share a model-driven app using Power Apps

Why don't I see my model-driven app changes?

Make sure you publish your app customizations. More information:

- Publish changes from Power Apps
- Publish your model-driven app using code

Also, verify your component changes in the active layer. You can use the solution layer feature in Power Apps to review solution layering for a component. More information: View the solution layers for a component

How do I add more tables to my app?

The site map defines the navigation for your model-driven app. You add tables to a model-driven app's site map by using the site map designer. Tables that aren't included in the site map won't be available in your app. More information: Create a model-driven app site map using the site map designer

How do I share my app?

You use security roles that contain the appropriate privileges and assign them to the users, teams, and apps you want. More information: Share a model-driven app using Power Apps

Why don't I see all the views after importing a solution?

A model-driven app change that was made by selecting **All** when selecting a component, such as a view, isn't reflected after you import an update to the app. More information: Newly added components don't appear in the app after you import an update

What's the best way to define and manage my model-driven app?

We recommend that you create your app in a dedicated solution. By creating a solution for your app, you can maintain your other solutions without creating any dependencies with your app. Only make changes to that solution when you want to update your app —for example, when you add, change, or remove components from the app. More information: Create a solution

What is `solutionaction`?

More advanced users can review the solution's customization.xml file. When you import a managed solution, customizations.xml might include the `solutionaction` property

when you add and remove components. The possible values for `solutionaction` are:

- Added
- Removed
- Modified

This value specifies the changes in the current layer with respect to the previous managed layer. For example, on solution import, `solutionAction="Removed"` will remove the component from the app. More information: *Microsoft.Crm.CrmInvalidOperationException: full formXml is expected to create a form <formid>* message during solution import

What is an app module?

The term *app module* has been used to describe model-driven apps, and `appmodule` is the name referenced in the Microsoft Dataverse and customer engagement apps schema. **Model-Driven App** is included in the display names for some of the tables referenced in Power Apps. For information about the `appmodule` table, go to AppModule entity reference. For information about using the `appmodule` table API, go to Create, manage, and publish model-driven apps using code.

App modules, which are referred to as model-driven apps in Power Apps, are created and customized within a solution. Solutions are used for application lifecycle management, and you can use them to migrate customizations from one environment to another. It's beneficial to be familiar with the solution layering concepts.

I'm a developer. Where do I get started?

Go to Supported Customizations for Dataverse.

How do I get help?

- Search or post your questions in the Power Apps Community.
- Admins can create support tickets.

See also

What are model-driven apps in Power Apps?

Feedback

Was this page helpful? Yes No

Provide product feedback

Model-driven apps developer guide

Model-driven apps are primarily a no-code or low-code component-focused approach to app development. Developers add value to model-driven apps by extending the application with custom code, components, and integrations. Before you start writing code, learn how to build model-driven apps and explore what options you can apply without code.

> **Note**
>
> Model-driven apps connect to Microsoft Dataverse. For information about how developers can add value at the service level, see Dataverse Developer Overview.
>
> This section refers only to extensions that developers can make to improve the experience for users of model-driven apps.

If you're new to Dataverse applications, the topics in this section provide a high-level overview of the important concepts to help developers get started working with model-driven apps.

Get started

Related articles

Understand model-driven apps components
Create your first model-driven app

Last updated on 03/28/2026

Model-driven app glossary of common terms

This article contains a glossary of terms for Power Apps model-driven apps.

Accessibility

Accessibility is a term that is used to refer to the extent to which people with disabilities can use digital products. In the case of model-driven apps, consideration has been paid to matters such as responsive design, how users navigate between fields, how the app behaves in high contrast mode, and how screen readers help users to understand the nature of the application.

Using screen readers within model-driven apps

Admin center

The Power Platform admin center is a unified portal for administrators to manage environments and settings for Power Apps, Power Automate, and Dynamics 365 apps. Power Platform admin center doesn't cover administration settings and features associated with Power BI.

Learn more about the Power Platform admin center

Learn about the Power BI admin center

App designer

The tool that is used to create and edit model-driven apps. As the app designer experience matures, it replaces the classic experience.

Use it to configure the navigation site map, tables, forms, and views relevant to your app.

An overview of the app designer experience

App navigation experience

The way in which areas, groups, subarea, and pages are presented in a model-driven app. It's also known as the site map

Application lifecycle management

The way in which we manage the lifecycle of an application from conception to end of life. From a technical perspective, much of application lifecycle management (ALM) is managed via solutions when delivering model-driven app products.

Overview of application lifecycle management with Microsoft Power Platform

Area

A part of the model-driven app navigation experience, apps can have multiple groups and groups can have multiple subareas. The subarea contains the tables relevant to the application. For apps with more than one area, a switch control is displayed in the lower left navigation pane.

App navigation in model-driven apps

Attribute

An attribute is another name for a column and is a term commonly used by Power Apps developers. Each table in Power Apps corresponds to a database table and each table column in Power Apps corresponds to a column in the database table.

Business process flow

Logic built into a given table to ensure that users complete records by updating fields in the correct order.

While these are authored initially using the Power Automate experience, business process flows are experienced within a model-driven app as a change in the user interface.

A business process flow is arranged into stages. Each stage defines the columns (fields) that must be completed typically before moving on to the next stage. For example, the default business process flow for the opportunity table has four stages: qualify > develop > propose > close. The current stage in a business process flow is indicated with a dot next to the stage in the sequence of stages from left to right in the flow.

Business process flows overview

Business rule

Business rules are server-side logic that is used with canvas or model-driven apps to set or clear values in one or more columns in a table. Business rules can also be used to validate stored data or display error messages. Model-driven apps can use business rules to show or hide columns, enable or disable columns, and create recommendations based on business intelligence.

Learn more about business rules

Define business rules

Canvas app

An app which is generated using drag and drop controls configured using Power Fx. Canvas apps offer the designer significant control over the user experience and can be connected to a wide range of data sources and data services.

Canvas apps are arranged into screens and controls such as galleries, text boxes, and dropdowns, are placed onto the screens and configured so that they connect to the data sources and to each other correctly.

Whereas a model-driven app comes with many preconfigured features such as forms, views, and a user interface, canvas apps can be authored from a blank canvas, or a template. There's often more work to be performed and more outright work using code.

Canvas apps are contained within environments and solutions in the same way as model-driven apps.

Find out more about canvas apps.

Chart

A visual representation of a table of data. Charts can take the form of line, bar, pie, or donut chart.

Find out more about creating a system chart.

Classic

The classic interface represents the method in which app makers make changes to features within their Microsoft Dataverse environment.

The classic interface has been replaced by the web-based method of app authoring known as the unified interface.

About Unified Interface for model-driven apps in Power Apps

Classic app designer

The modern app designer lets you create model-driven apps and create canvas apps using custom pages.

The modern app designer is the default designer for model-driven apps. Currently, you can still create model-driven apps using the classic app designer.

Column

A column (formerly called a field), is a field within a Dataverse table (formerly called an entity). Columns are similar to fields in databases and have different data types such as text, number, date, as well as data types less familiar to databases such as phone, email, file, and image.

The column type defines the kind of data required by the column and also the controls, such as date picker or text box, that is available when using the control.

Columns also appear when you create forms. Form tabs also have columns, and this defines where you can put sections. Additionally, form sections have columns, and these define where you can place table columns (form fields in this case).

How to create and edit columns

Add, configure, move, or delete columns on a form

Command bar

The area of a model-driven app that contains basic commands universally used by model-driven apps.



The command bar can be customized. More information: Customize the command bar using command designer (preview)

Component

Components are elements. Components are used when creating the elements that make up a model-driven app. Often these elements relate to the method of creation of the tables that make up a model-driven app.

Components can be split into data (tables, relationships, columns) UI (site map, forms, views), logic (Power Automate flows, business process flows, business rules) and visualization (charts, dashboards, and Power BI Tiles).

Learn more about components

Connection

A model-driven app is typically connected to the data tables that reside in the same environment. This connection can be considered native because it never has to be set up within the environment.

Connections exist within the environment to enable other elements of the Power Platform to operate correctly. Notably, Power Apps canvas apps and Power Automate flows have the ability to make use of multiple connections.

Control

Controls allow you to interact with information contained within records. They typically are visible on forms, where users update data using the control. Examples of controls are calendar, toggle, choices, slider, and editable grids. In some cases, you might want to use different controls depending upon the device employed by the user.

Find out more about controls

Dashboard

A container for one or more charts relating to a table.

Find out more about dashboards here

A dashboard allows charts, Power BI reports, and views of tables to be presented to the app user.

Find out more about how to use Power BI within a model-driven app

Data model

A collection of related tables. In the context of model-driven apps, these are held within the Dataverse database.

In a custom solution, the data model is often the set of related tables built with the purpose of delivering the overall business application.

Database

The collective term for all the tables in Dataverse.

Dataverse

Microsoft Dataverse is the collective term for the tables, workflows, business process flows, and related functionality that are provisioned within an environment when a database is created.

Model-driven apps require a Dataverse database.

A Dataverse database contains data structures most closely associated with databases in addition to being able to hold model-driven apps, canvas apps, and Power Automate flows.

Find out more about Dataverse here

Dependency

Dependencies are created when elements of components are reliant on each other to work. For example, if a column is used within a view then the view requires the column to exist for it to function. There are many examples of dependencies throughout Dataverse. Another example is a model-driven app being dependent on a table if that table is used within the app.

Dependencies manifest themselves in numerous ways including when a model-driven app is validated. They also become apparent in the most problematic fashion when trying to delete an aspect of a table, form, view, or dashboard. Dependencies can be viewed by selecting the item to be deleted, and then selecting **Advanced** > **Show dependencies** on the command bar.

- Removing dependencies
- Validate an app
- Delete a column from a table

Dynamics 365

Microsoft Dynamics 365 is a line of enterprise resource planning (ERP) and customer relationship management (CRM) software applications. Microsoft markets Dynamics 365 applications through a network of reselling partners who provide specialized services.

Learn more about Microsoft Dynamics 365

Entity

An entity is the classic way of describing a table. This terminology appears within the classic experiences.

Environment

An environment is a space to store, manage, and share your organization's business data, data structures, apps, Copilot agents, and flows.

You can package up the various elements as solutions, and these solutions can be exported from one environment to another.

An environment can only ever have one Dataverse database and all your model-driven apps in the environment use this database.

Often multiple environments are used to enable application lifecycle management. For example you might have development, test, and production environments.

Environments exist within a geographical region and can be a means of ensuring that the data physically stays in the correct geographical region.

Find out more about environments

Flow

Cloud flows are functionality offered by Power Automate that allow automation of tasks to take place based upon triggering of conditions such as recurrence, adding or updating of records or simply selection of buttons by users. Flows can be run with or without the introduction of new parameters.

Cloud flows

Form

Forms provide the user interface (UI) that people use to create, view, or edit table records. Use the form designer in Power Apps to create and edit forms.

There are four types of forms: main, quick create, quick view, and card.

More information:

- Form Types
- Opening the form designer
- Learn about creating and designing forms
- Add a section to or remove a section from a form
- Add a tab to or remove a tab from a form

Form designer

The design experience for creating and editing forms.

Opening the form designer

Group

A part of the model-driven app navigation experience. Group names appear as a navigation element in an app with the subarea names (tables) within the group listed beneath it.

Legacy

This refers to features that have either been deprecated, or the way in which they're authored, has been moved to a more modern experience, such as the web-based unified interface.

Lookup

A lookup is a field type that exists when two tables are related. Lookups can be seen in table views on the many side of a one-to-many relationship. Lookups are generally populated using a form on the many side of the relationship.

- Learn more about the lookup field user experience
- How to configure a lookup

Main form

Every table has at least one main form. The main form represents the primary method of interaction with a record. The main form is responsive to the device using the screen form and can contain controls that are optimized to the device whether it's phone, tablet, or web. Main forms are edited using the form designer.

Learn about other form types

Monitor

Also know as the app monitor. It lets you understand aspects of the performance of a model-driven app. App monitor can an also be used to monitor canvas apps.

- Learn how to use the app monitor
- Use Monitor to understand form performance

Page

Modern apps have the concept of pages, which can be either a model-driven app component (table, dashboard, navigation link, or web resource) or a canvas-based page using a custom page. Custom pages allow flexible layout, low-code Power Fx functions, and Power Apps connector data.

A custom page is a tool for enabling model-driven apps and canvas apps to function together.

- Working with custom pages
- Learn about creating modern apps

Power Automate

A Power Platform service that allows users to streamline repetitive tasks. Typically, this automation is performed using cloud flows.

Model-driven app business process flows that direct users to complete table records in a specific fashion, are authored within Power Automate.

Power Automate flows exist within an environment and can also exist within Power Apps solutions.

Learn more about Power Automate

Power BI

A data visualization tool that has the capacity to be embedded within model-driven apps or to operate independently of them. Power BI can connect to a wide range of data sources, of which Dataverse is just one.

Power BI reports aren't stored within Dataverse environments.

- Use Power BI within a model driven app
- Learn more about Power BI

Publish

The process by which you make the latest iteration of the app available to users within an environment.

Publisher

Every solution has a publisher. You specify the publisher when you create a solution. The solution publisher indicates who developed the app, and defines the prefix, such as *Contoso*_MyNewTable, for all the solution assets.

Learn more about publishers

Record

A record contains one or more columns of information about a person, a place, or a thing. For example, a record might contain the name, the email address, and the phone number of a single customer. Other tools refer to a record as a "row" or an "item." Records exist within Dataverse tables.

Relationship

The way columns in different tables relate to each other. There are three types of relationship:

- One-to-many. For example, one author to many novels.
- Many-to-one. For example, many pages to one book.
- Many-to-many. For example, many books borrowed by many people.

Model-driven apps often contain tables with relationships between them. Where relationships exist, users navigate to the record within the related table. For example, when looking at a sales invoice record, you can open the related account record to investigate details for that account.

Learn about creating table relationships

Responsive apps

An app that is responsive renders itself in a way that depends on the device that is accessing the app. This can also mean that there might be a different control displayed, such as a date picker, depending on whether the user is running the app on a computer, tablet, or phone.

Additionally, tables and fields render themselves according to screen size of the device.

Section

Tabs within forms are arranged into sections. Sections can be arranged into one to four columns and let you arrange the record metadata in a way that is most relevant to the current tab and the current section.

Learn more about working with sections

Security role

A security role defines what people can see and do with features and data in Dataverse. For records, security roles determine the create, read, write, delete, update, and append actions available to a user.

Security roles are created and users are put into security roles either as individual user names or by using active directory security groups.

You grant access to model-driven apps through security roles.

- Find out more about security roles
- General overview of security in Microsoft Dataverse
- Get started with security roles in Dataverse

Site map

A model-driven app is essentially a collection of tables, dashboards, views, and pages, and these are described via the site map. The site map defines the tables and pages that are included within a model-driven app and the navigation experience users have when moving between them.

When configuring the navigation experience you're editing the areas, groups, and subarea navigation elements. Tables exist at the level of the subarea, and are arranged into groups. Groups are effectively collections of tables and pages and are visible in the navigation pane. Areas allow you to toggle between visible groups.

Both modern and classic methods of creating a model-driven app include site maps. However, with the modern app designer you can design the site map with a drag and drop experience whereas the classic site map designer doesn't support drag and drop.

To open the site map in the classic site map designer from the modern app building experience, select **Switch to classic**.

Find out more about app navigation

Solution

A solution is a wrapper for a wide range of components including tables, cloud flows, and security roles.

When you make a model-driven app, ensure that the assets associated with it are held inside a solution.

Solutions have two types:

- Managed solutions generally permit only a small amount of customization or no customization at all.
- Unmanaged solutions give makers full control over the project that they're creating.

Unmanaged solutions are used by makers and developers for exporting projects as a managed solution for use in nondevelopment environments, such as a production environment. This allows for a high level of control for application lifecycle management.

- Find out more about solutions here
- Discover solutions in the context of Dataverse

Solution explorer

This is the classic experience that makers and customizers can use to view and make changes to most any solution component. More modern experiences are available from make.powerapps.com and the model-driven app designers and solution explorer will eventually be replaced with those modern experiences.

To access the modern solution interface, follow these steps:

1. Sign in to make.powerapps.com.
2. Select an environment.
3. On the left pane, select **Solutions**, and then open an unmanaged solution where you want to add a model-driven app. Create a solution if one doesn't already exist.
4. Explore the components of the solution.

Find out more about solutions

Subarea

A part of the model driven app navigation experience. Subareas (tables) and pages appear under the group that they're configured within in the app designer.

Subgrid

Subgrids are areas of main forms that display a list of records from a Dataverse table, while remaining on the form. Typically, a subgrid is used to display child records that relate to the parent record currently under review. For example, books written by an author.

While subgrids are displayed in a model-driven app, they're a property of the form.

- Create a subgrid
- Subgrid properties - legacy

Tab

Every form has at least one tab and these are relevant to how we present table record data. A form can have multiple tabs. This lets you, the maker, offer the user a range of ways of looking at the same record. Tabs often offer a better user experience by providing a more logical way of presenting the data in the record.

From a site map perspective, a tab is a "group" when using the site map designer versus a subarea for tables and an area to hold subareas.

Learn more about working with tabs

Table

A table is a method of storing data in columns (or fields) within Dataverse. Tables where formerly called entities.

Tables, in the context of model-driven apps, only exist within a Dataverse database.

A single row within a table is known as a record. For example, a single customer, and the columns describe metadata associated with the customer such as the name, telephone number, or credit limit.

Every model-driven app must contain at least one table. Much of the process of creating a model-driven app is selecting the tables most relevant to solving the business problem.

Tables have views, forms, and business rules associated with them.

Additionally, tables also have charts as well as dashboards where charts are presented.

Tables can relate to other tables and these are defined via the relationships that have been set up between them.

Find out more about configuring tables

Table designer

The design experience for creating and editing tables. This lets you create tables, columns, relationships, business rules, and views.

Create a custom table using the table designer

Unified Interface

The Unified Interface provides a consistent and accessible user experience across devices—whether on a desktop, laptop, tablet, or phone. The predecessor to the Unified Interface was known as the web interface.

Find out more about the unified interface here

Validate

The process by which an app maker confirms if the model-driven app has all the components required for it to function properly.

Learn how to validate an app

View

A tabular representation of records in a Dataverse table. Tables can have multiple views.

Views can be prefiltered and it's possible to define the specific views that a model-driven app makes available to users.

Tables can have multiple views associated with them and you can define the table views relevant to a model-driven app at the time that you create them.

Find out more about views here

Workflow

A classic workflow is a series of functions or methods, called steps, that are performed sequentially and apply to data contained within tables. The workflow can change the processing direction by using conditionals, referred to as conditional branches.

In many cases classic workflows should be replaced with Power Automate flows.

Last updated on 04/03/2026

Install the Power Apps mobile app

If you're not signed up for Power Apps, sign up for free. Then choose the download link or scan a QR code to download Power Apps mobile.

Expand table

iOS	Android	Windows
Download from App store	Download from Google Play	Download from Windows Store

> **Note**
>
> For more information on Power Apps for Windows, see Power Apps for Windows.

Required privileges and supported devices

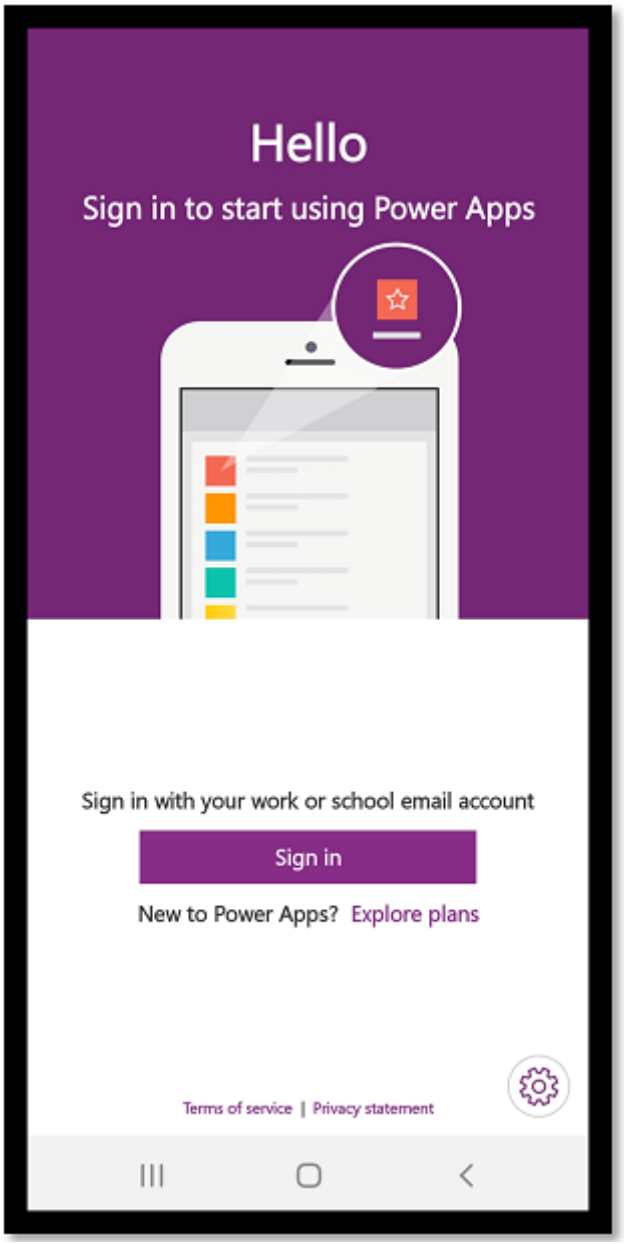
Review the following privileges and supported devices to run Power Apps Mobile app:

- Required privileges
- Supported platforms for running apps using the Power Apps mobile app.

Sign in

Open Power Apps on your mobile device, and sign in by using your Azure Active Directory credentials.

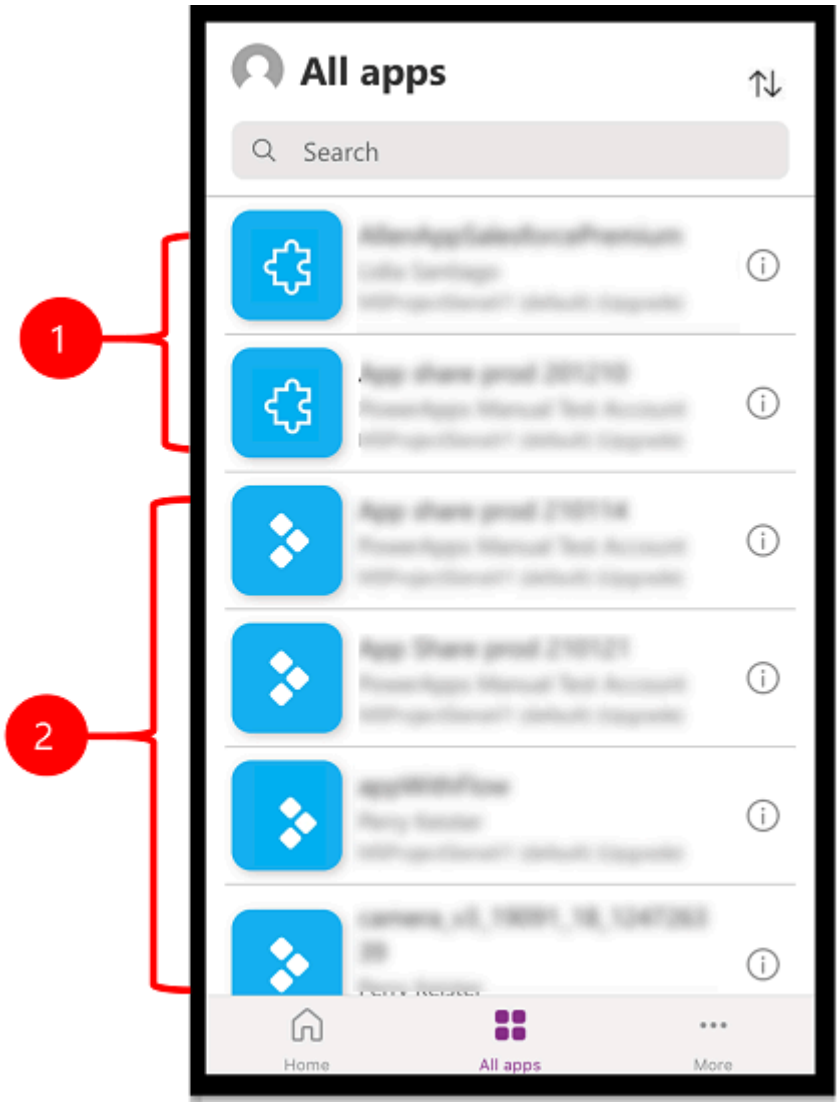
If you have the Microsoft Authenticator app installed on your mobile device, enter your username when prompted, and then approve the notification sent to your device. If you run into issues signing in, see Troubleshoot issues for Power Apps mobile app.



Find the app

When you create an app, or someone shares an app with you—either a canvas app or model-driven app—you can run that app on Power Apps mobile.

> **Note**
>
> To see a model-driven app in the list of apps on Power Apps mobile, you need to have a predefined security role in the environment that the app is in. If a predefined security role is assigned to a user using a Dataverse team, you need to use an Azure Active Directory (AAD) group team. Users will not see model-driven apps if a predefined security role is assigned using a Dataverse owner team.

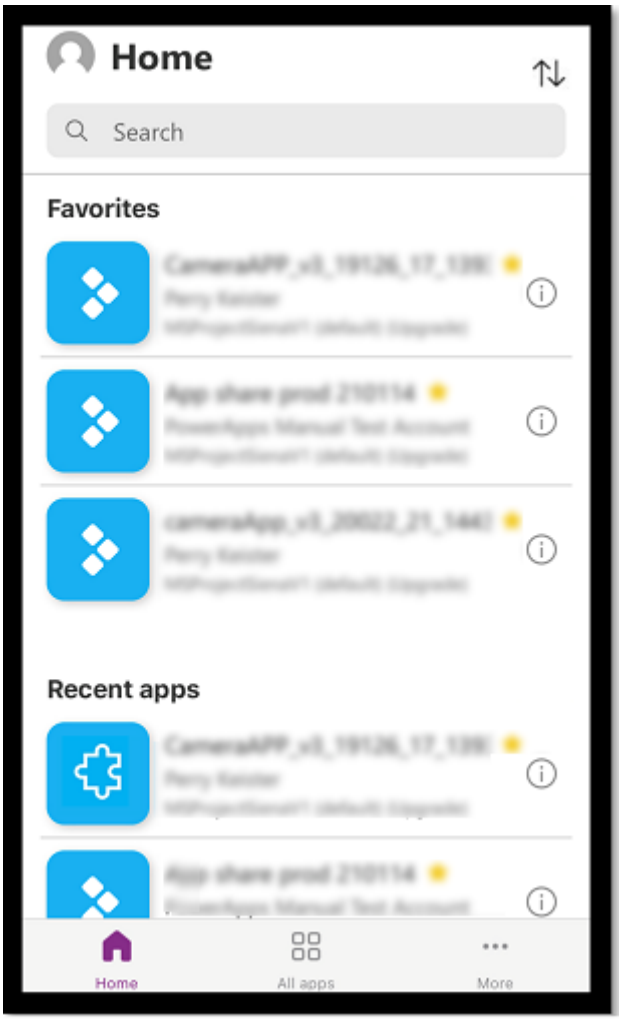


Legend:

1. **Model-driven apps**
2. **Canvas apps**

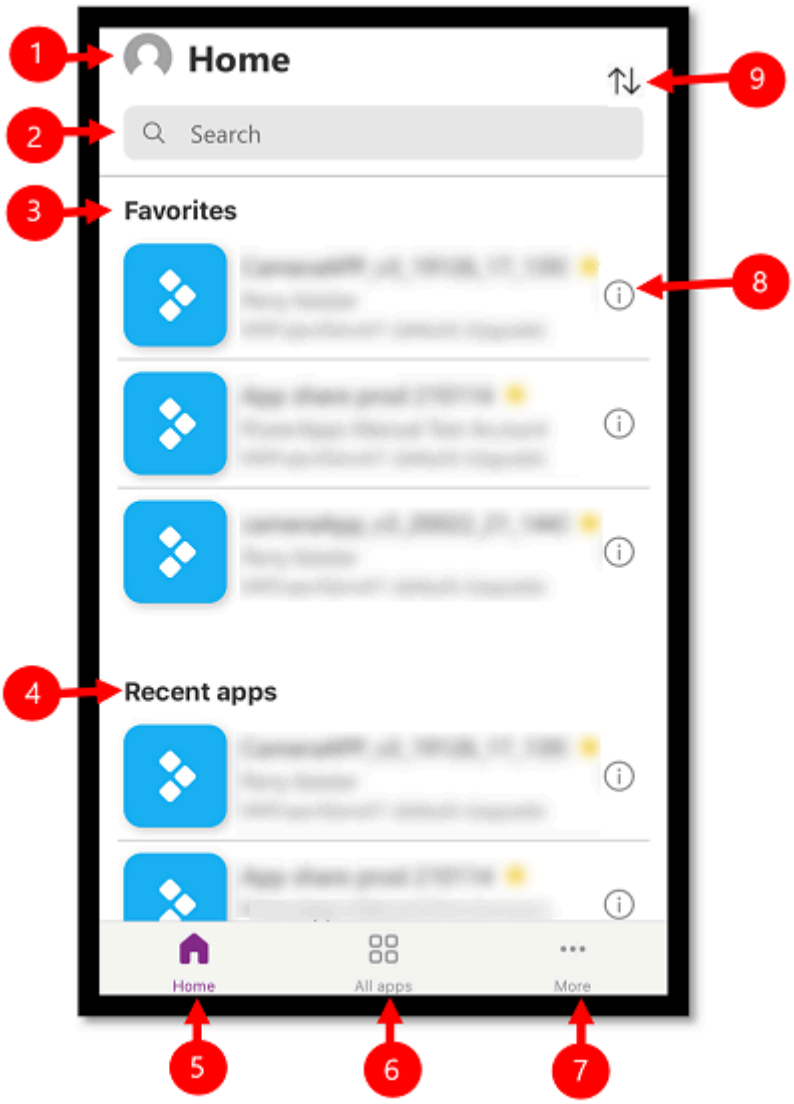
The apps that you used recently will show on the default screen when you sign in to Power Apps mobile.

The **Home** is the default screen when you sign in. It shows the apps that you used recently and the apps that have marked as favorites.



Filter apps

If you don't have any apps, then when you sign in, you will land on the **All apps** screen. The list of apps is organized in alphabetical order. Type in an app name in the search bar to find an app.

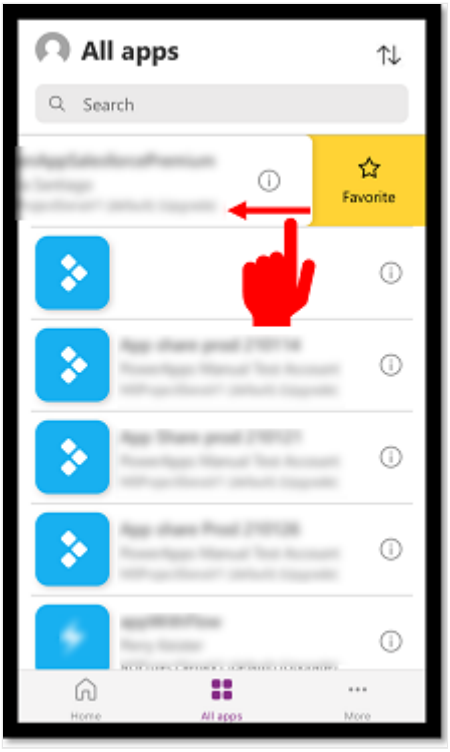


1. **Settings**: Access app settings and sign out.
2. **Search**: Use the search to search for apps. When you run a search, it will only search for apps that are on the screen you're on.
3. **Favorites** (canvas apps only): Displays canvas apps that you have pinned to favorites.
4. **Recent apps**: Displays both model-driven and canvas apps that you have recently used.
5. **Home**: Displays favorite apps and recently accessed apps sorted by open date.
6. **All apps**: Displays all canvas apps and model-driven apps to which you have access, including apps you created and apps that others shared with you.
7. **More** (canvas apps only): Displays featured and sample apps.
8. **Details**: View information about the app including run the app, add a shortcut for the app, and add the app to favorites.
9. **Sort apps**: You can short by the app name or modified date.

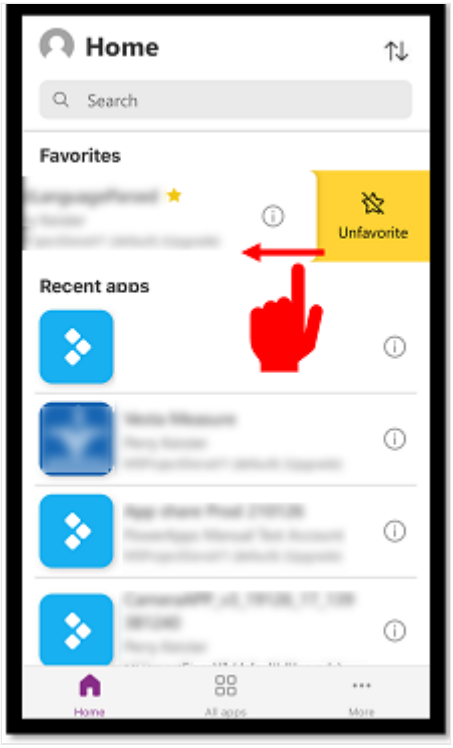
Add to favorites

You can add canvas and model-driven apps to your list of favorite apps.

- Swipe left and then select **Favorite**. A yellow star will appear next to app name when it's added to favorites. You can also select **Details** ⓘ and then add the app to favorites.

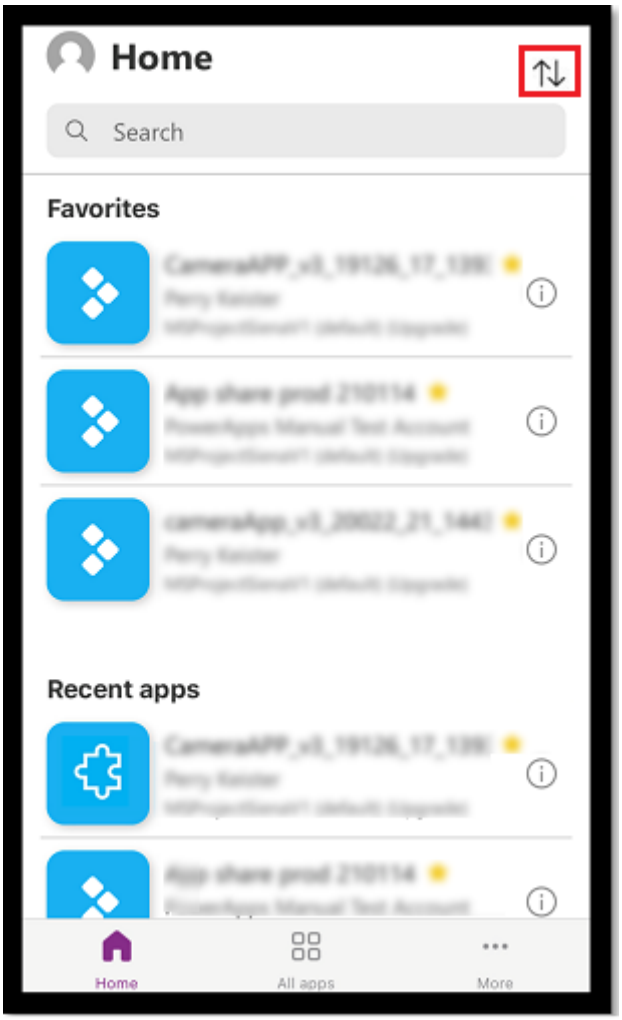


- To remove the app from the list, swipe left again and then select **Unfavorite**.



Sort apps

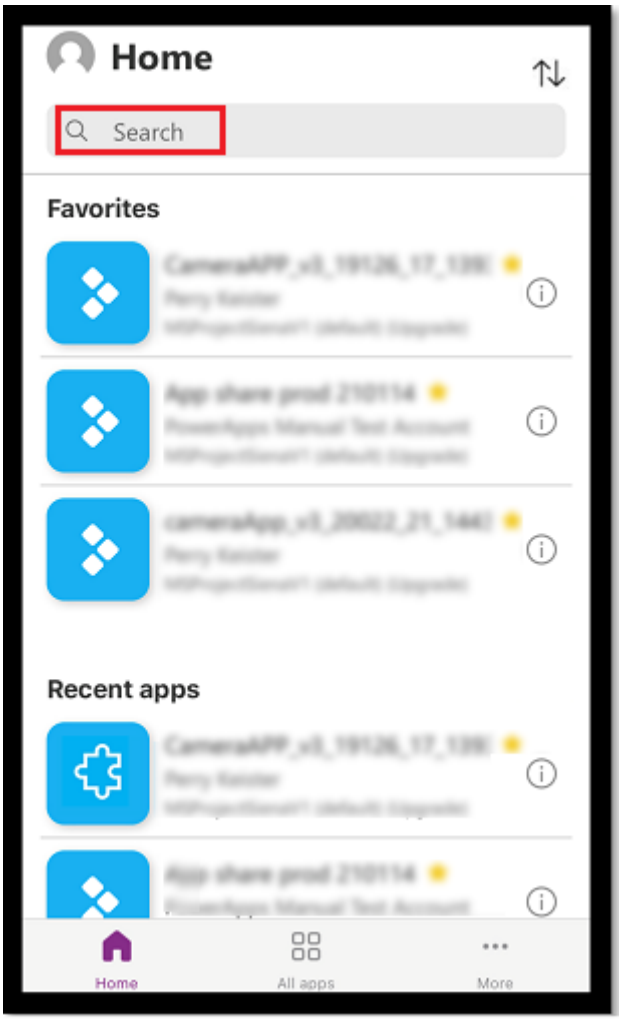
You can sort both canvas apps and model-driven apps. You can sort apps alphabetically by name or by modified date. The sort option is available on **Home**, **All apps**, **Featured apps**, and **Sample apps** screen.



Search apps

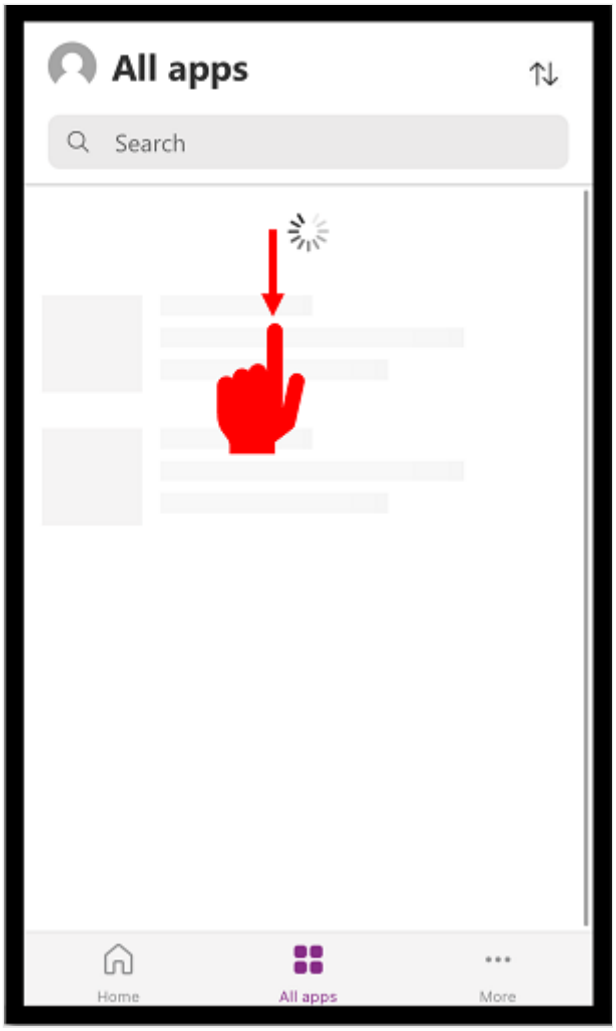
If you know the name of the app that you want to run, then use search to quickly find the app. You can search for both canvas apps and model-driven apps.

To find an app, enter the app name in the search field. The app will only search for apps that are on the screen you're on.



Refresh the list of apps

On the **Home**, **All apps** or any other screen with a list of apps, swipe down to refresh the app list.

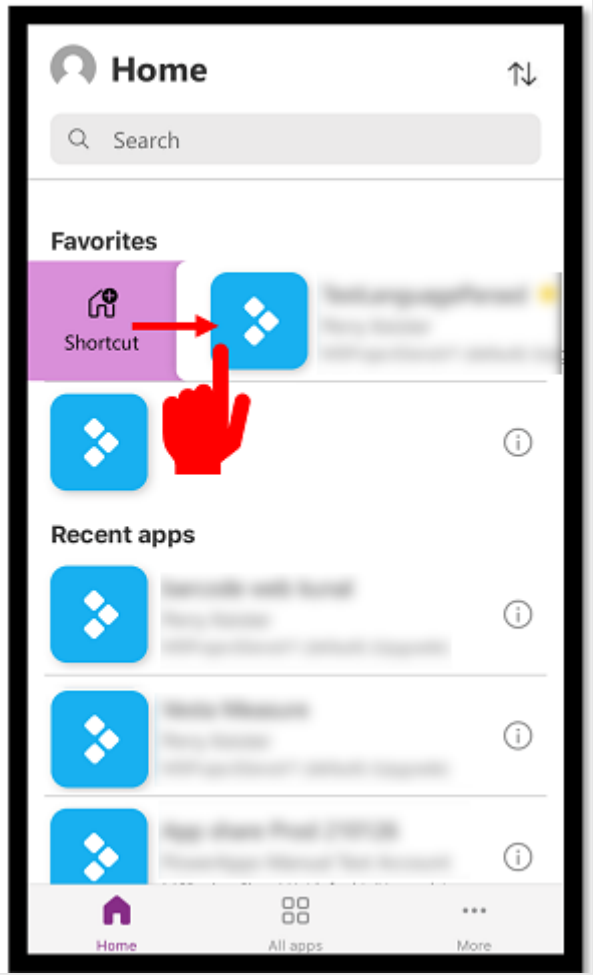


Add shortcuts

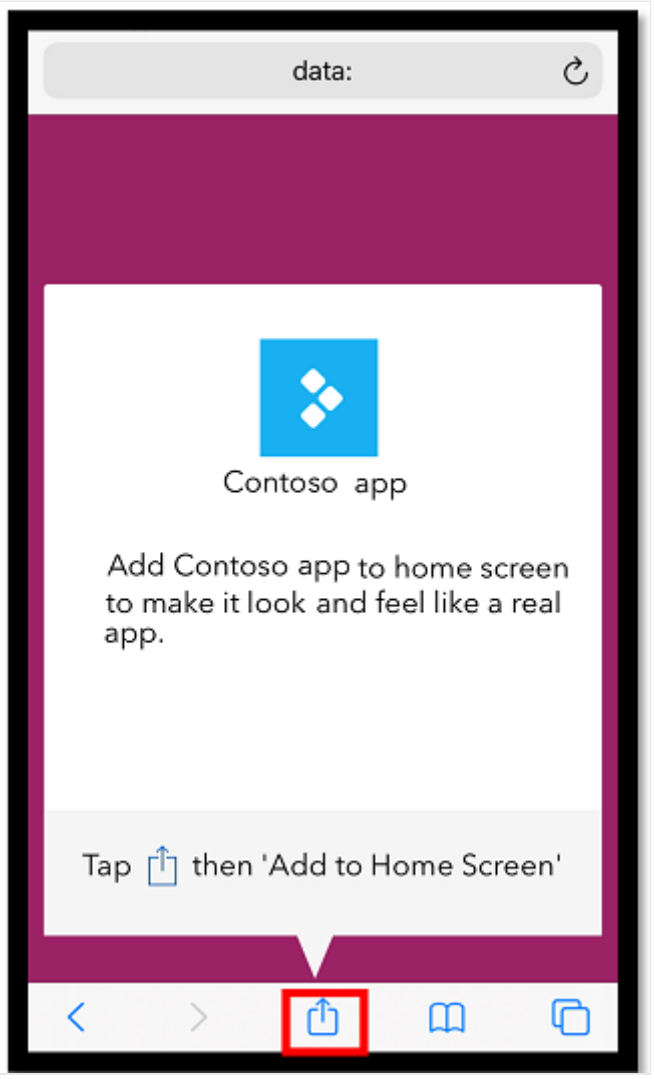
You can add a shortcut for both canvas apps and model-driven apps to the home screen of your device for quick access.

Use Safari to add a shortcut (iOS 13 or earlier)

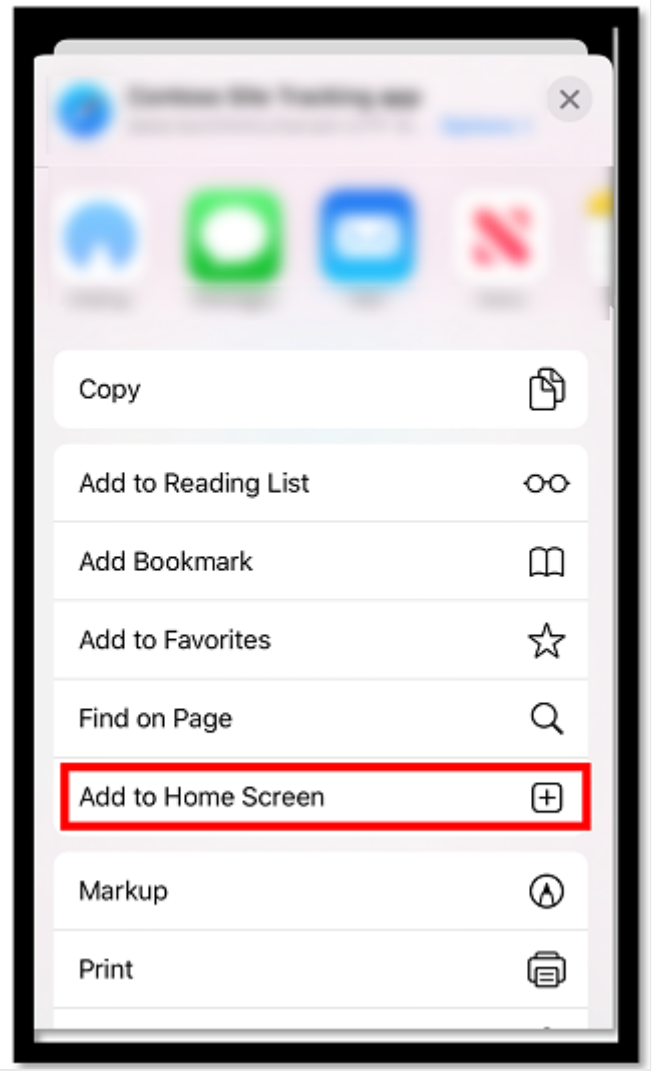
1. On the app that you want to create a shortcut for, swipe to the right and select **Shortcut**.


Home
Search
Favorites
Shortcut
Recent apps
Home
All apps
More

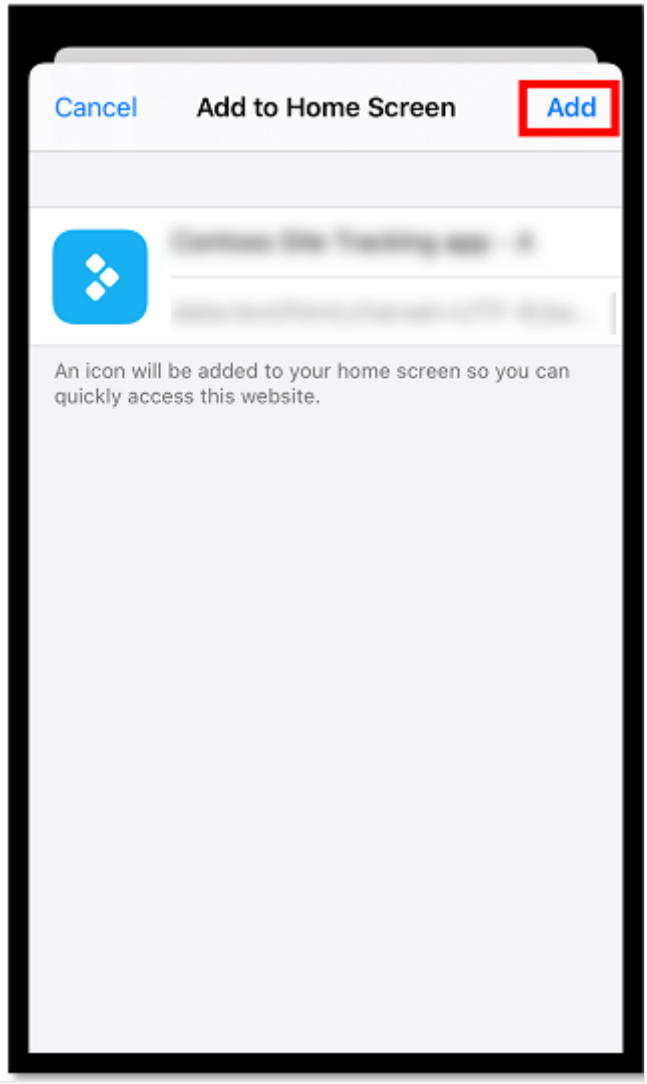
2. Select .



3. Scroll down and select **Add to Home Screen**


Copy
Add to Reading List
Add Bookmark
Add to Favorites
Find on Page
Add to Home Screen
Markup
Print

4. Select **Add**.



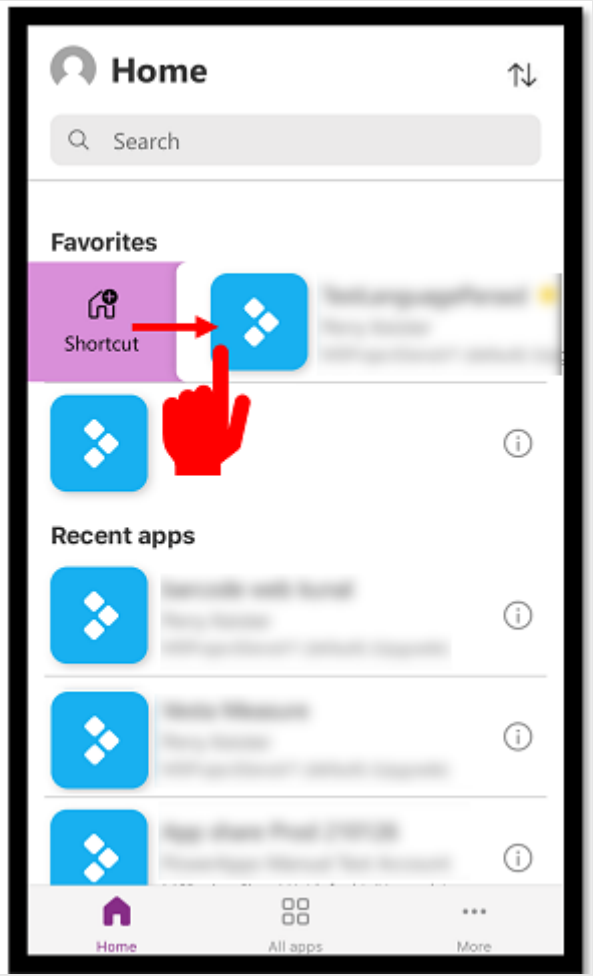
> **Note**
>
> For iOS devices that have multiple browsers installed, use Safari when pinning an app to home.

Use Siri shortcuts to add a shortcut to the Home screen (iOS 14 or later)

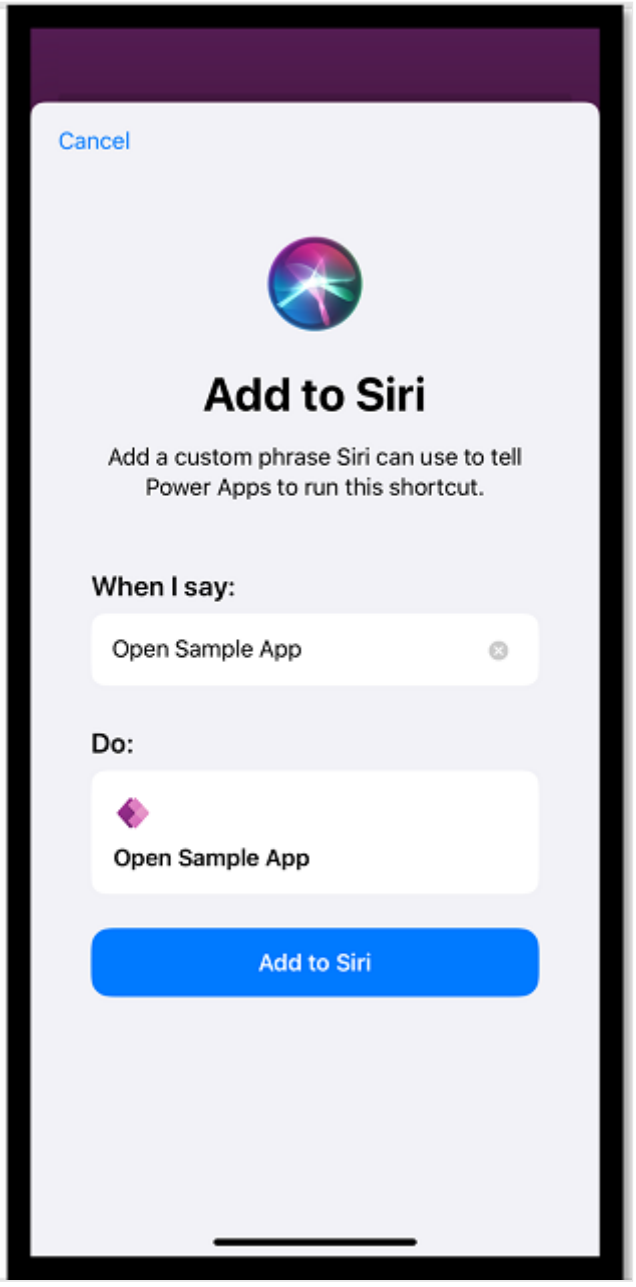
The Power Apps mobile app is now integrated with Siri shortcuts, which gives you the ability to add a shortcut to the Home screen, launch apps with Siri, and create new workflows. For more information on how shortcuts work on iOS, see Shortcuts User Guide. This feature requires Power Apps mobile version 3.20092.x or later.

Users on iOS 14 or later can use Siri Shortcuts to pin an app to the home screen. The new experience works for both model-driven and canvas apps. When you add a Siri shortcut, the app is added to the iOS **Shortcuts** app and from there you can add the app to your home screen.

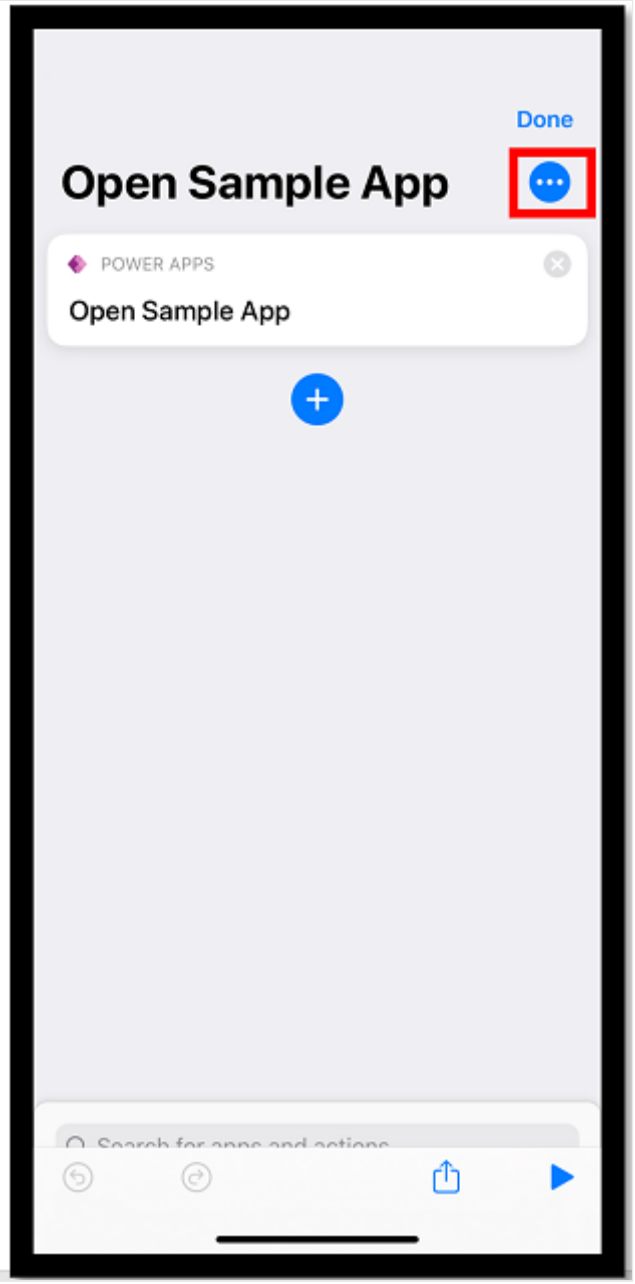
1. On the app that you want add a shortcut for, swipe to the right and select **Shortcut**.



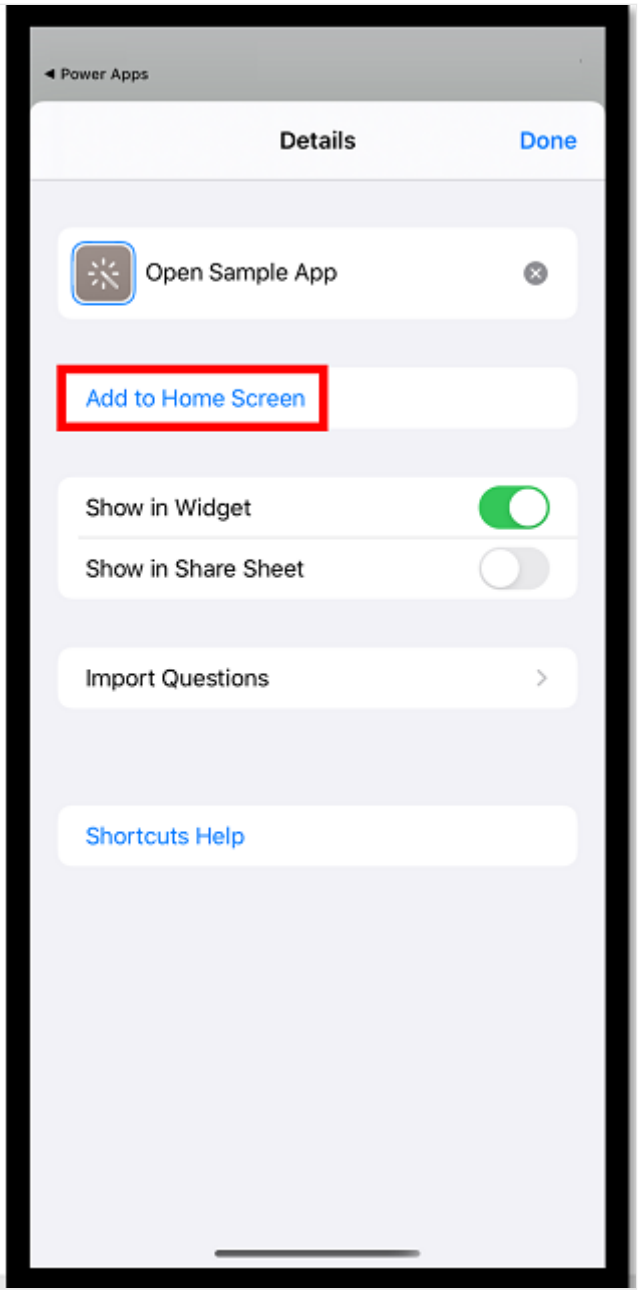
2. Add a custom phrase to open the app using voice commands and then select **Add to Siri**.



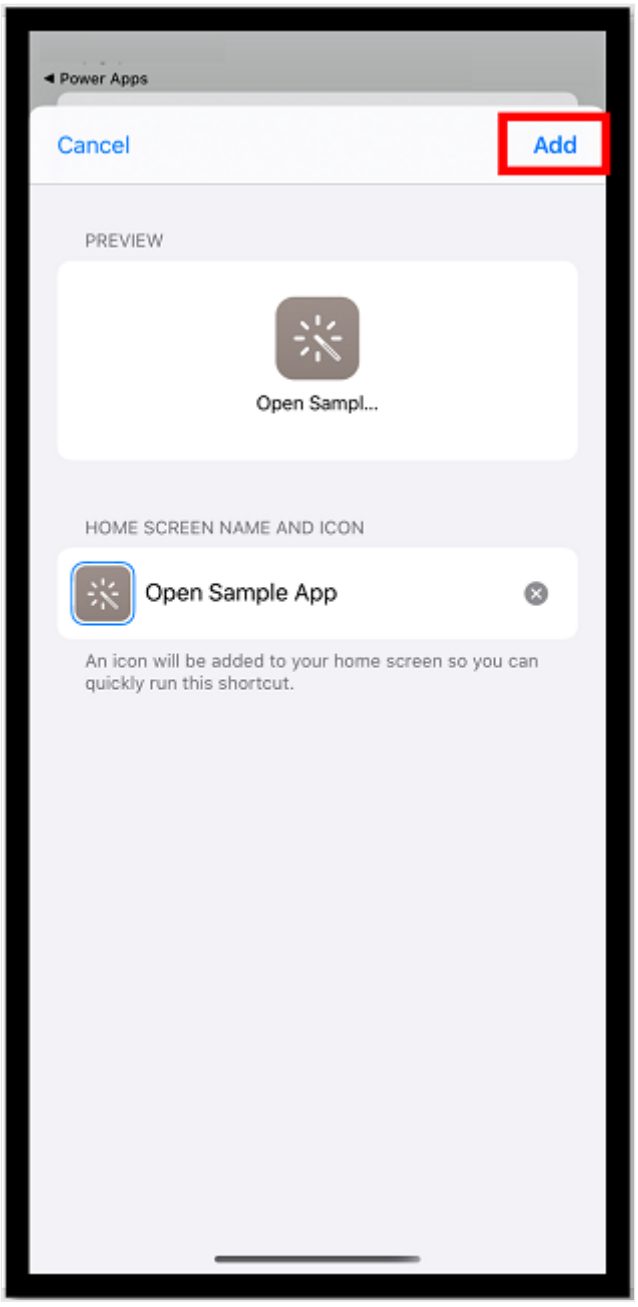
3. The app is added to the **Shortcuts** app on your mobile device. Open the **Shortcuts** app and select the ellipsis (...) above the app name.



4. Select **Add to Home Screen**.



5. On the upper right corner, select **Add** and then select **Done**.



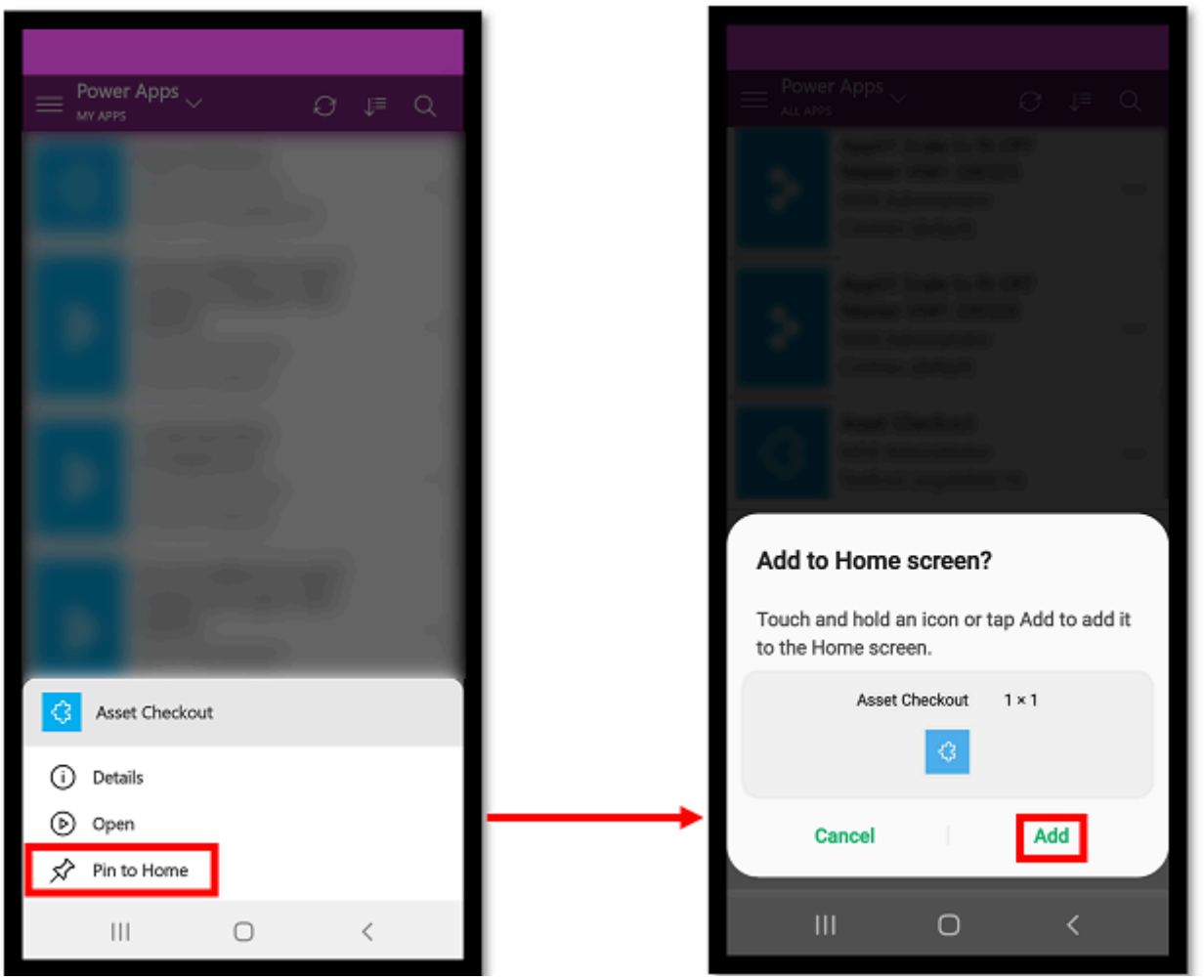
6. Go to your home screen to find the pinned app.



You can customize the shortcut icon but it is limited to the customization options in iOS. For more information, go to Modify shortcut icons .

Pin to home on Android

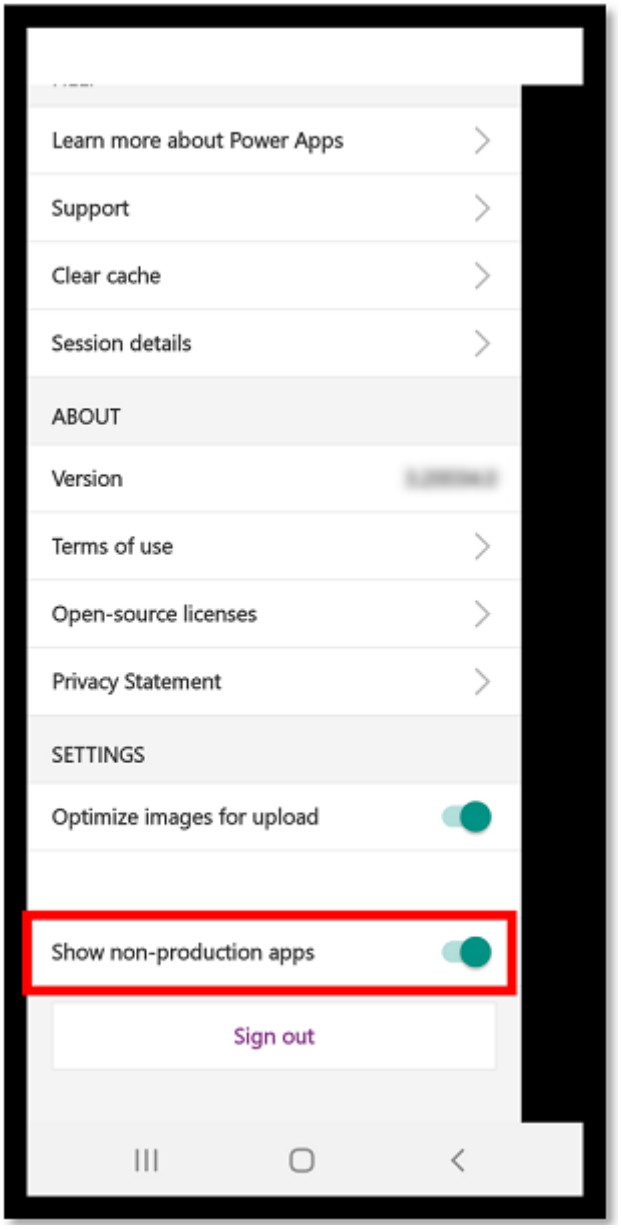
Select the ellipsis (...) on the app tile, select **Pin to Home**, and then follow the instructions that appear.



See non-production apps

By default, only production model-driven apps are shown in the list of apps.

To see model-driven apps from non-production environments, select the **Settings** menu , and then turn on **Show non-production apps**. Follow the instructions that appear.



Run an app

To run an app on a mobile device, select the app tile from the **Home** or **All apps** screen in the Power Apps mobile app.

Considerations when opening apps:

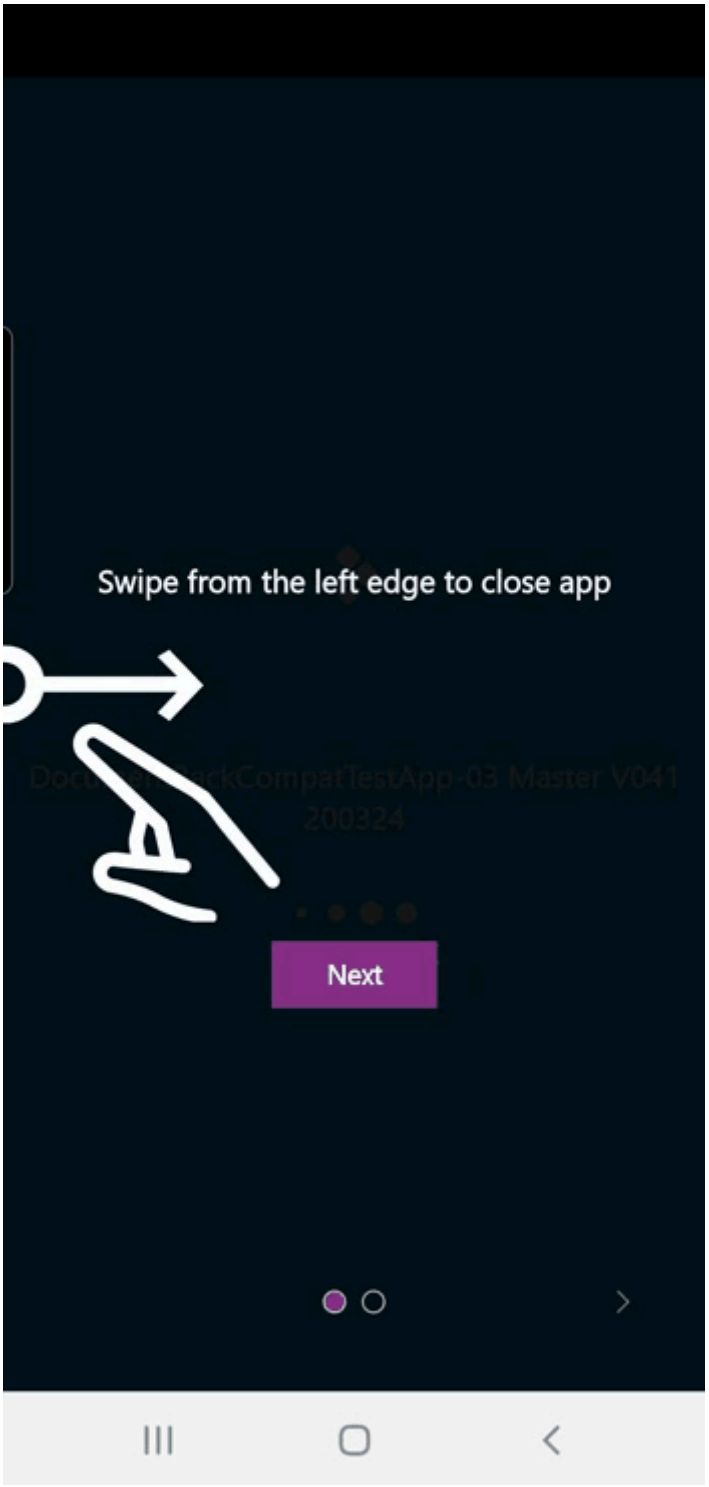
- If you created the app, you see the app in the **Home** page > **Recommended apps** section.
- If you didn't create the app and the app is shared with you, use search box in the **Apps** screen to find the app the first time. After you open the app, it shows up on the **Home** screen the next time you access it.

Run a canvas app

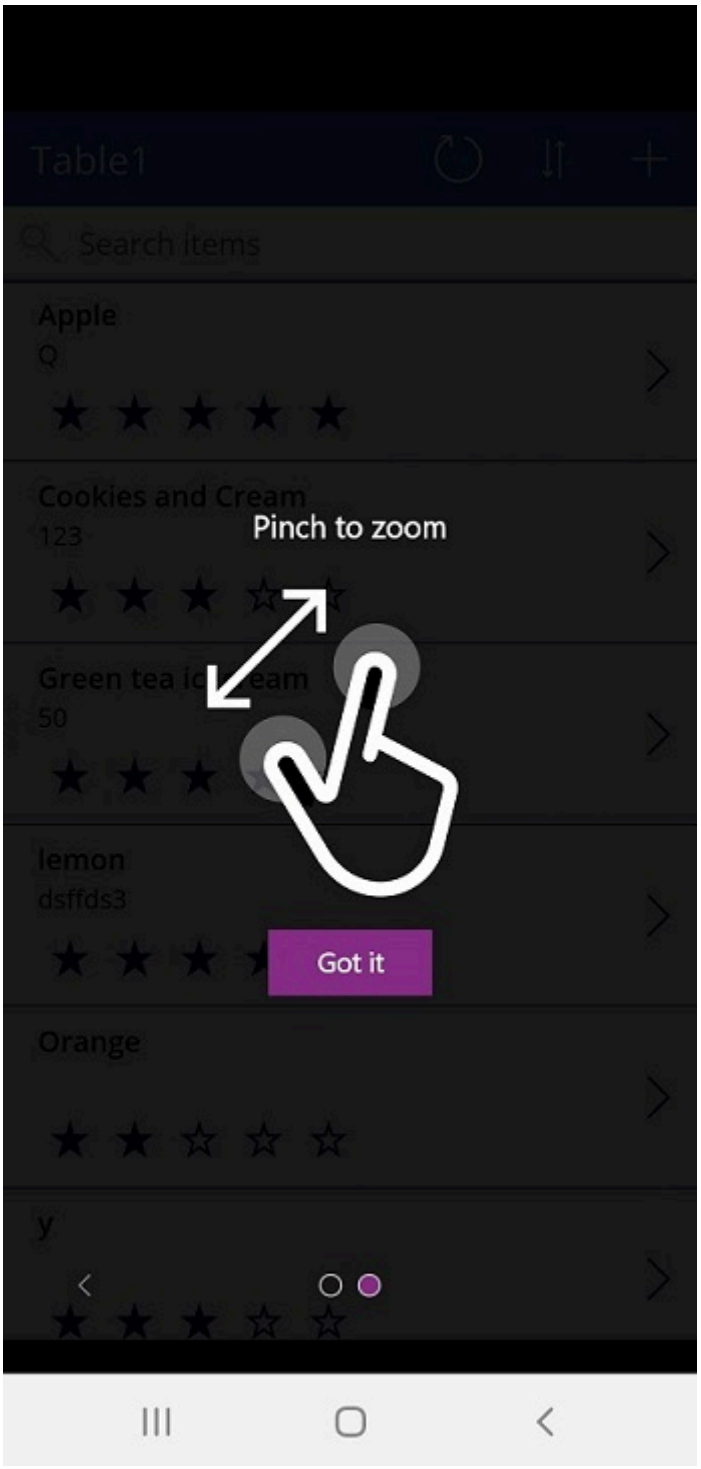
If this is the first time you're running a canvas app by using Power Apps mobile, a screen shows the swipe gestures.

Close a canvas app

Use your finger to swipe from the left edge of the app to the right to close an app. On Android devices, you can also press the Back button and then confirm that you intended to close the app.

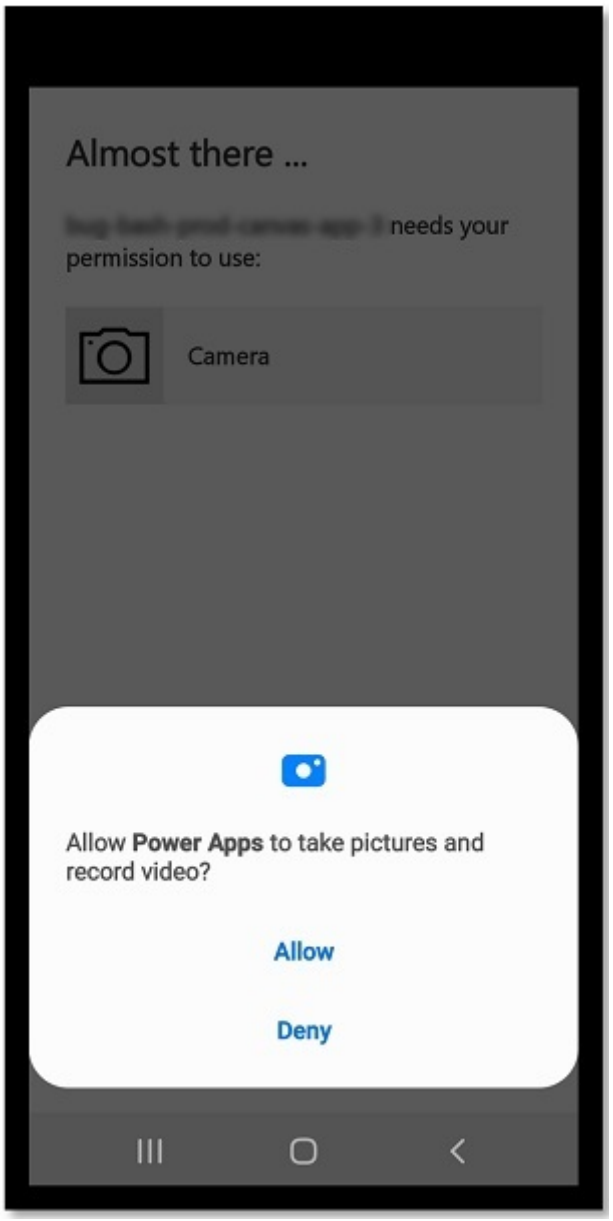


Pinch and zoom in on a canvas app



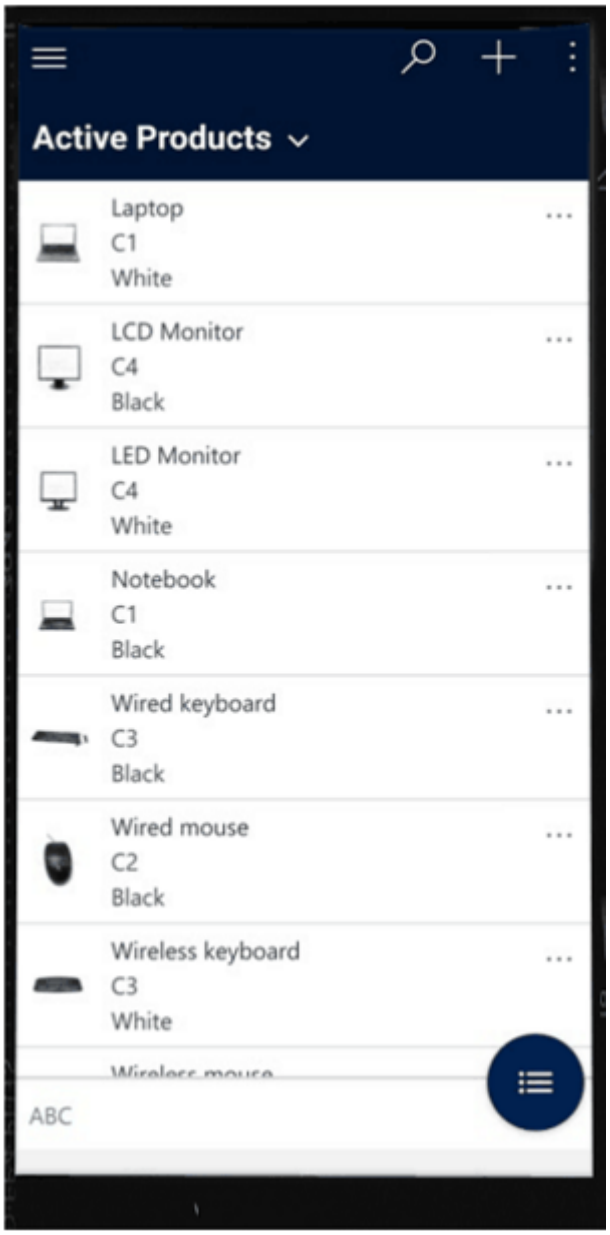
Give consent to a canvas app

If an app requires a connection to a data source or permission to use the device's capabilities (such as the camera or location services), you must give consent before you can use the app. Typically, you're prompted only the first time you run the app.



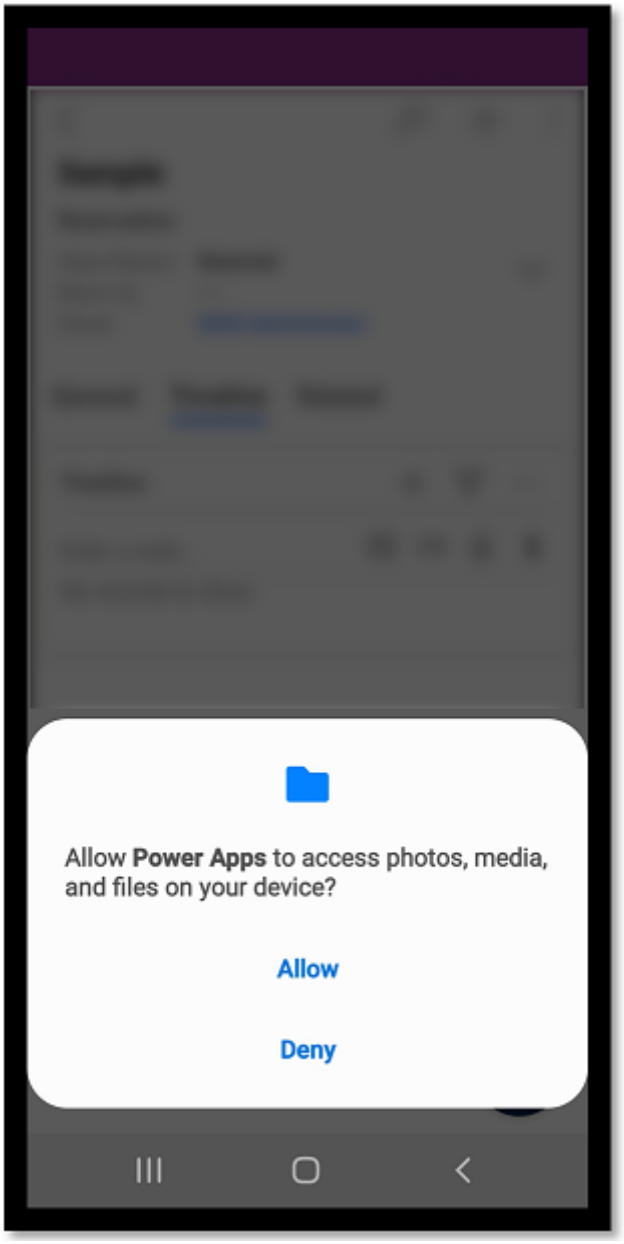
Use a model-driven app

The following image shows an example of a model-driven app screen after you've signed in. To learn how to use model-driven apps running on Power Apps mobile, go to Use model-driven apps on Power Apps mobile.



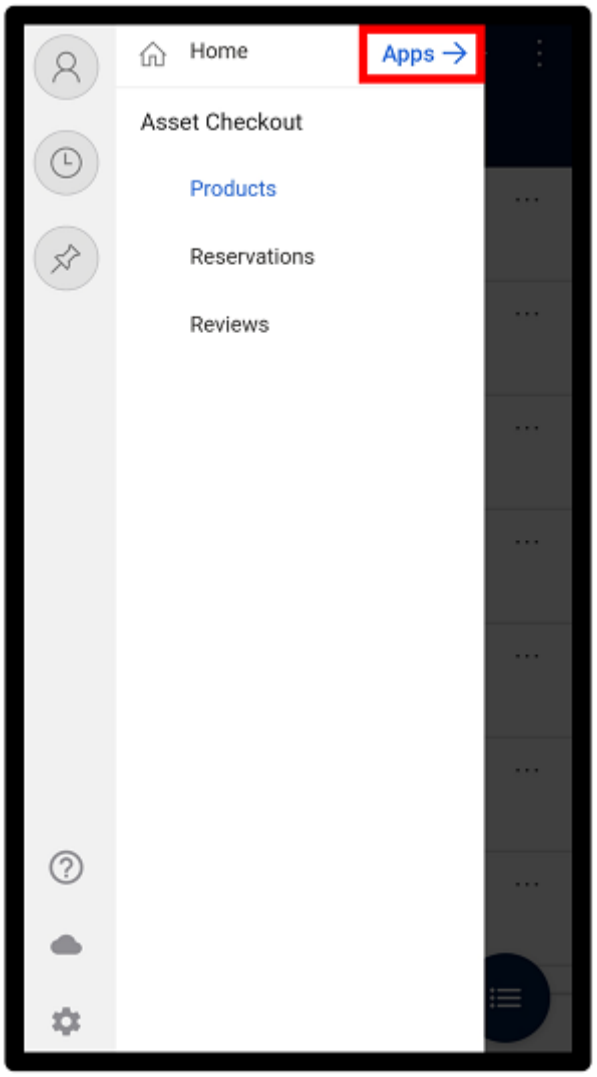
Give consent to a model-driven app

If an app requires a connection to a data source or permission to use the device's capabilities (such as the camera or location services), you must give consent before you can use the app. Typically, you're prompted only the first time you use the app.



Close a model-drive app

Select the site map , and then select **Apps**.



> **Note**
>
> An app maker can customize the navigation bar and show or hide the **Home**, **Recent**, **Pinned** buttons in the site map. An app maker can also make groups collapsible. For more information, see Hide or show the Home, Pinned, Recent, and collapsible groups.

Other mobile apps

The table below outlines which other mobile apps you can use to run your app.

Expand table

Mobile App	Apps you can run
Power Apps mobile (covered in this topic)	• Model-driven apps • Canvas apps • Dynamics 365 Marketing • Dynamics 365 Customer Service

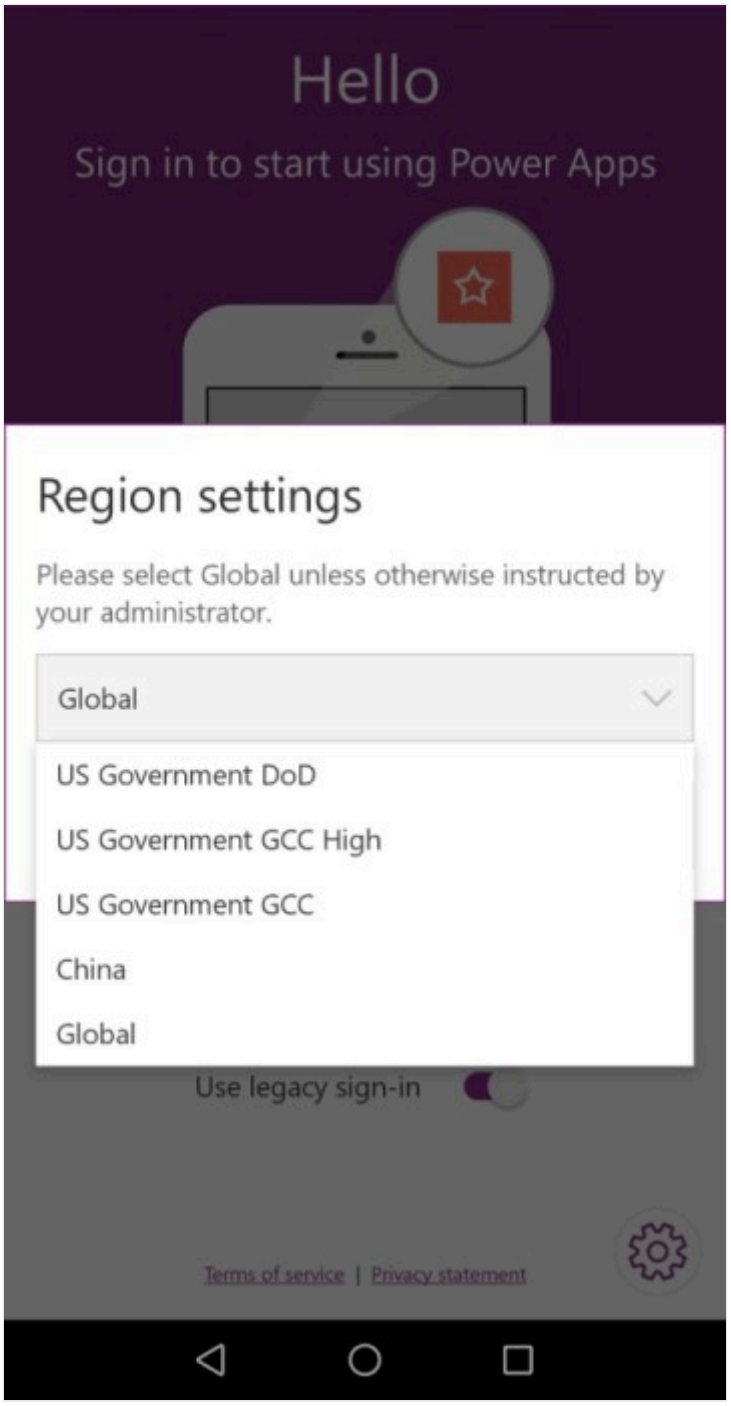
Mobile App	Apps you can run
Power Apps for Windows	• Model-driven apps • Canvas apps
Dynamics 365 for phone and tablets	• Microsoft Dynamics 365 Customer Engagement (on-premises) **Note**: Dynamics 365 for Tablets is deprecated, and won't be supported in 2023.
Dynamics 365 Sales Mobile	• Dynamics 365 Sales
Field Service Mobile	• Field Service (Dynamics 365)

Regional availability of Power Apps mobile app

Power Apps mobile app is available to users in Azure global cloud and also in the following regions:

- US Department of Defense (US DoD)
- US Government Community Cloud (GCC) High
- US Government Community Cloud (GCC)
- China Sovereign Cloud

Mobile users have an option to select their region on Power Apps mobile app sign in screen.



More information:

- Azure Government documentation
- Dynamics 365 US Government
- Power Platform and Dynamics 365 apps in China

Privacy notice

Power Apps mobile app for Android registers for a system event that is broadcasted when the device is finished booting. Power Apps mobile app for Android registers for this event to support push notifications sent to the app.

Power Apps mobile app and wrapped native mobile apps may use device sensors, such as the device accelerometer, to respond to user actions. For example, wrapped native mobile apps would automatically show the app menu when the user shakes the device.

See also

Use model-driven apps on Power Apps mobile
Troubleshoot issues for Power Apps mobile

Last updated on 02/06/2026

Manage your mobile app with Microsoft Intune

Article • 05/26/2025

Mobile application management software enables IT administrators to apply and enforce corporate policies on mobile apps. One mobile application management option for IT administrators is Microsoft Intune, which offers a suite of features that lets you publish, push, configure, secure, monitor, and update mobile apps - including Power Apps mobile, Dynamics 365 Sales Mobile, and Field Service Mobile. Mobile application management is essential to organizations whose users use mobile app primarily because:

1. Users frequently travel to multiple locations, and protecting sensitive company data is critical.
2. Many organizations have a bring-your-own-device policy, which means that the mobile app needs management among many devices and apps for personal use.

With an Intune-enabled mobile app, IT administrators can:

- Add and assign the mobile app to user groups and devices, including users in specific groups, devices in particular groups, and more.
- See reports and track app usage.
- Limit sharing of corporate data among apps by restricting data leakage through cut, copy, paste, and save-as.
- Provide encryption at rest.

Intune is a separate Microsoft product that is not included with Power Apps mobile. Refer to the documentation on What is Microsoft Intune app management? and Assign apps to groups with Microsoft Intune to get started.

A configuration can be set up through the Device Management portal. Each supported platform (iOS, Android, and Windows) requires a separate configuration.

Enhanced security support in Intune

Intune advanced app settings aren't currently supported with Power Apps Mobile. However, organization admins and makers have complete control over authentication, user access, managing sensitive data access, configuring notifications, setting data sync policies across sources, and defining access to organizational data and documents, through combination of Power Platform environment settings and Azure conditional access policies.

Set the appearance of your app to dark or light mode

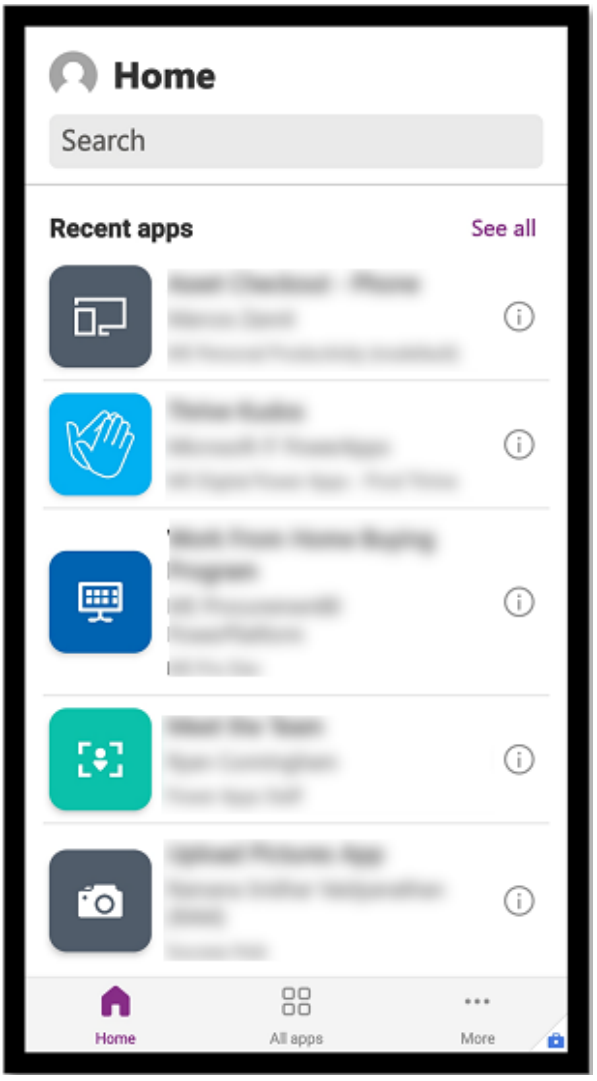
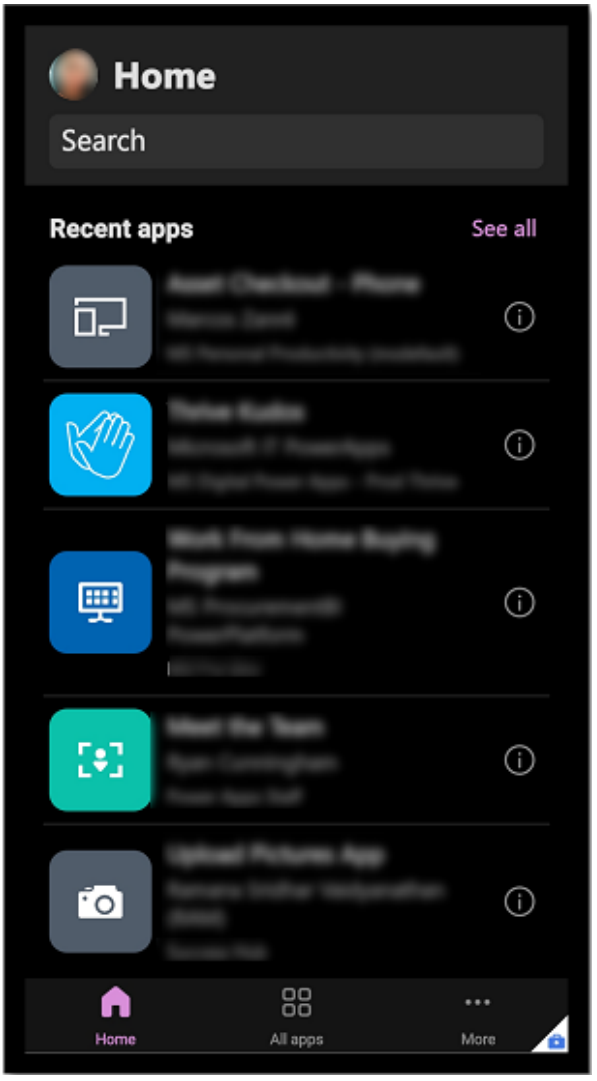
Article • 11/30/2022

Easily change the display of your app to dark mode, light mode, or set it to the system default on iOS and Android devices.

When you choose the **System default**, the display setting for Power Apps mobile will match the dark or light mode display setting on your iOS or Android device.

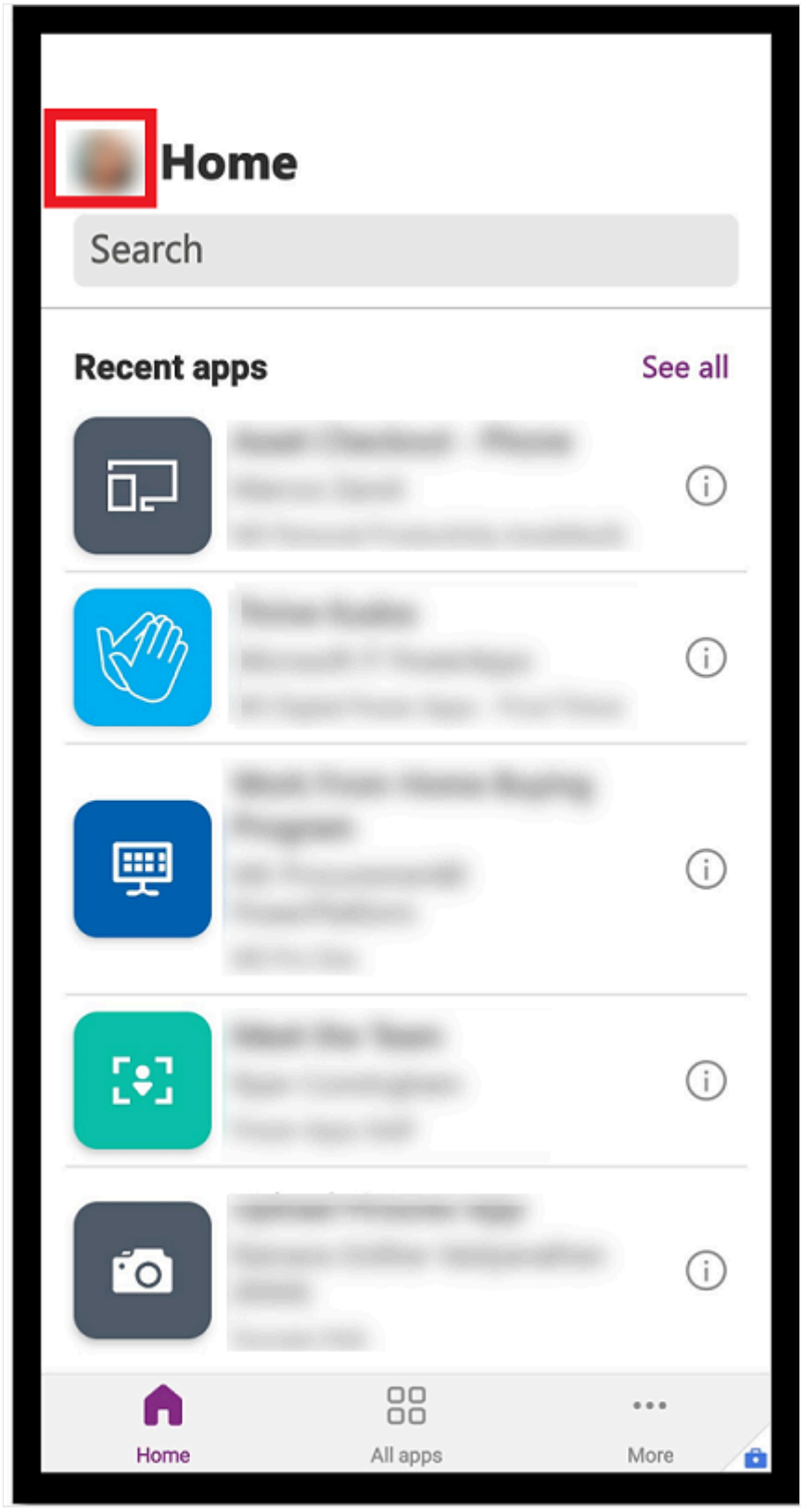
The default display mode for Power Apps mobile is set to **Light**.

Expand table

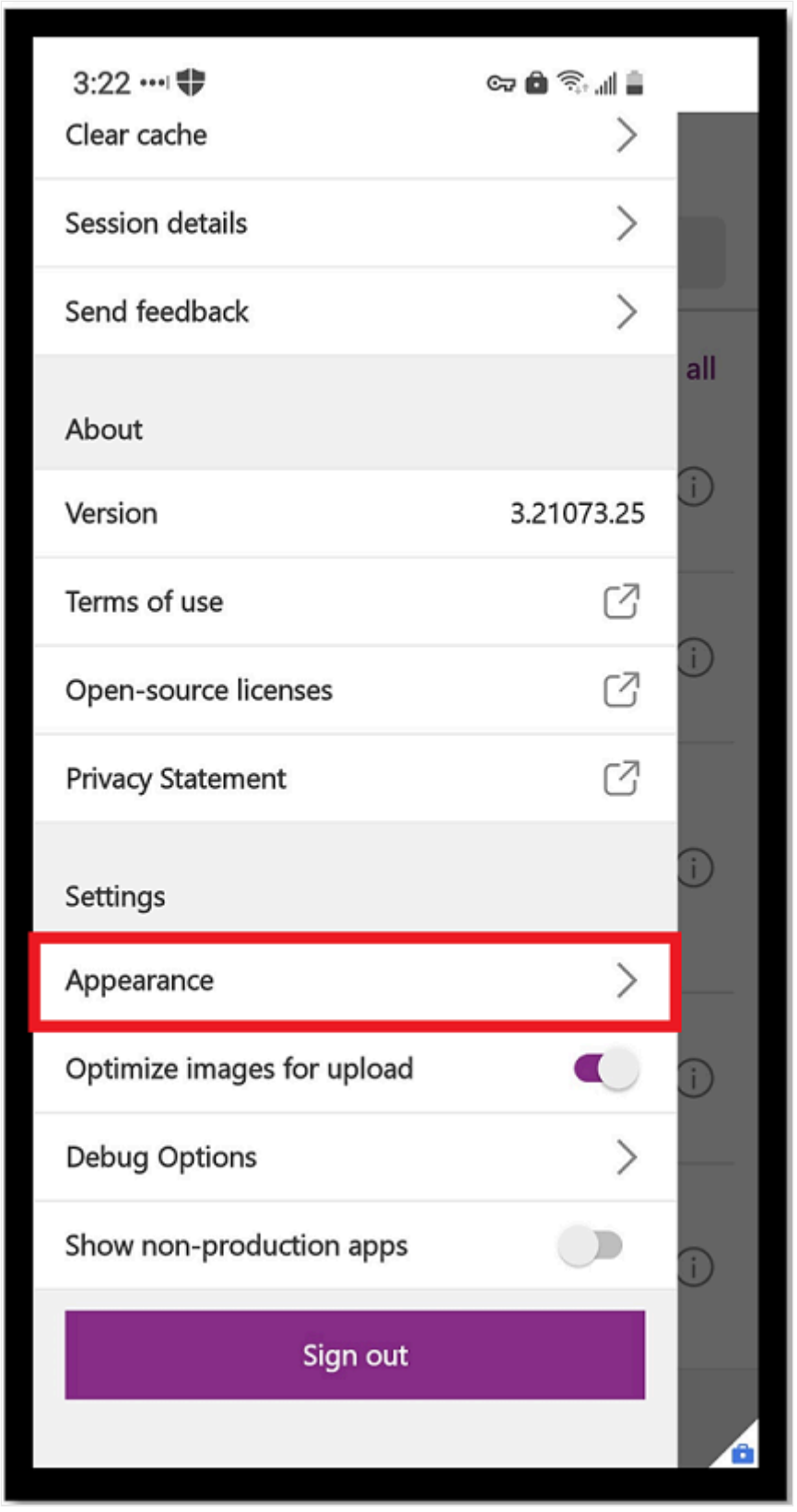
Light mode appearance	Dark mode appearance
	

1. To change the display, open Power Apps mobile and sign in.

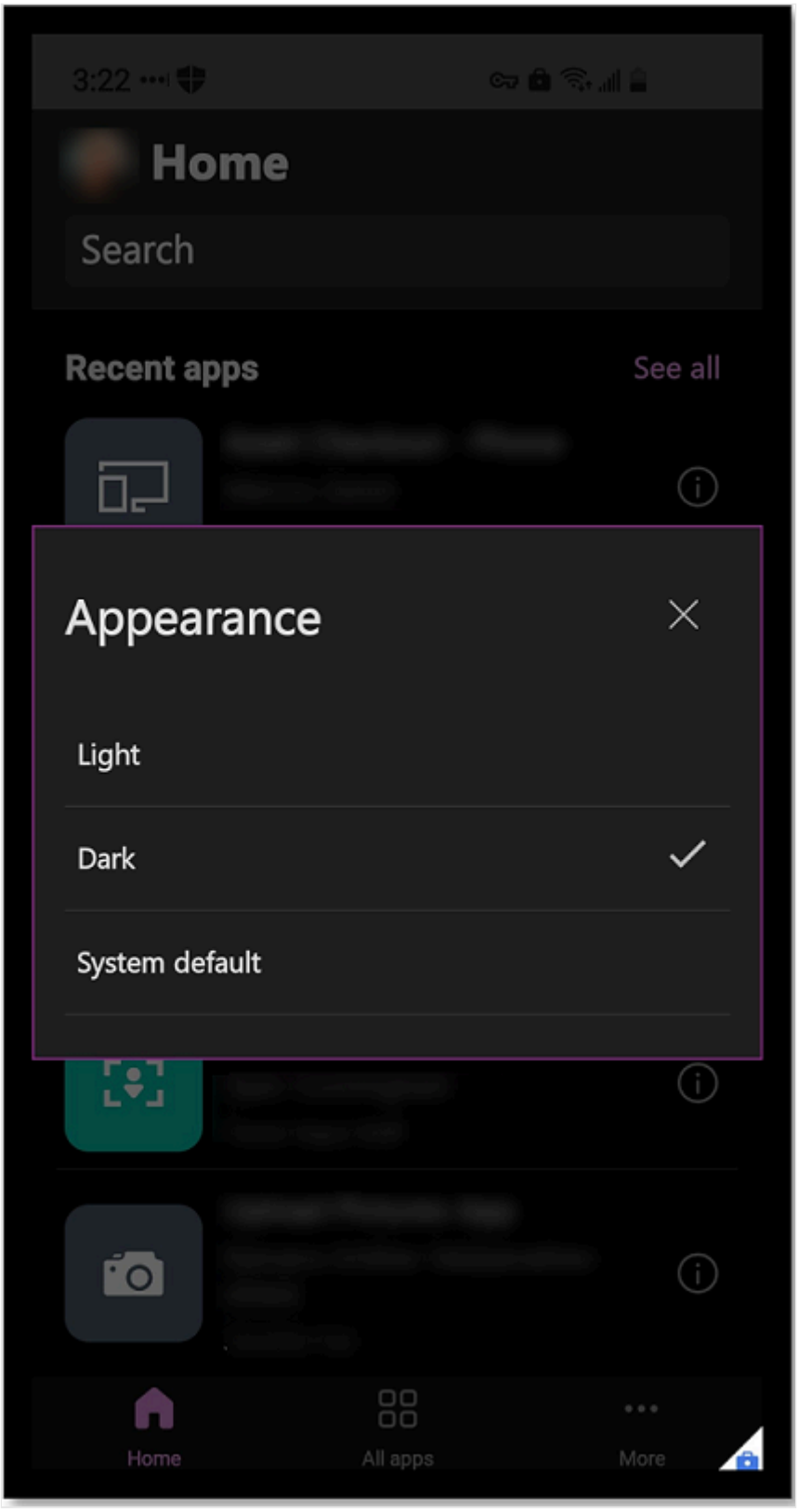
2. From the **Home** screen, select your profile picture.



3. Under **Settings**, select **Appearance**.



4. Select your display setting: **Light**, **Dark**, or **System default**.



5. When you're done, select the **X** to close the window.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use model-driven apps in the Power Apps mobile app

Article • 08/02/2023

Use Power Apps mobile to run model-driven apps on your mobile device. For more information about how to install and get started with an app, see Install Power Apps mobile.

Important

- To use your model-driven app in the Power Apps mobile app, your app maker must have set the **Primary mobile player** setting to **Power Apps mobile.** For more information, see Manage model-driven app settings in the app designer.

Home screen

It's easy to get around in Power Apps mobile. The following illustration shows the primary navigation elements on the Home screen.



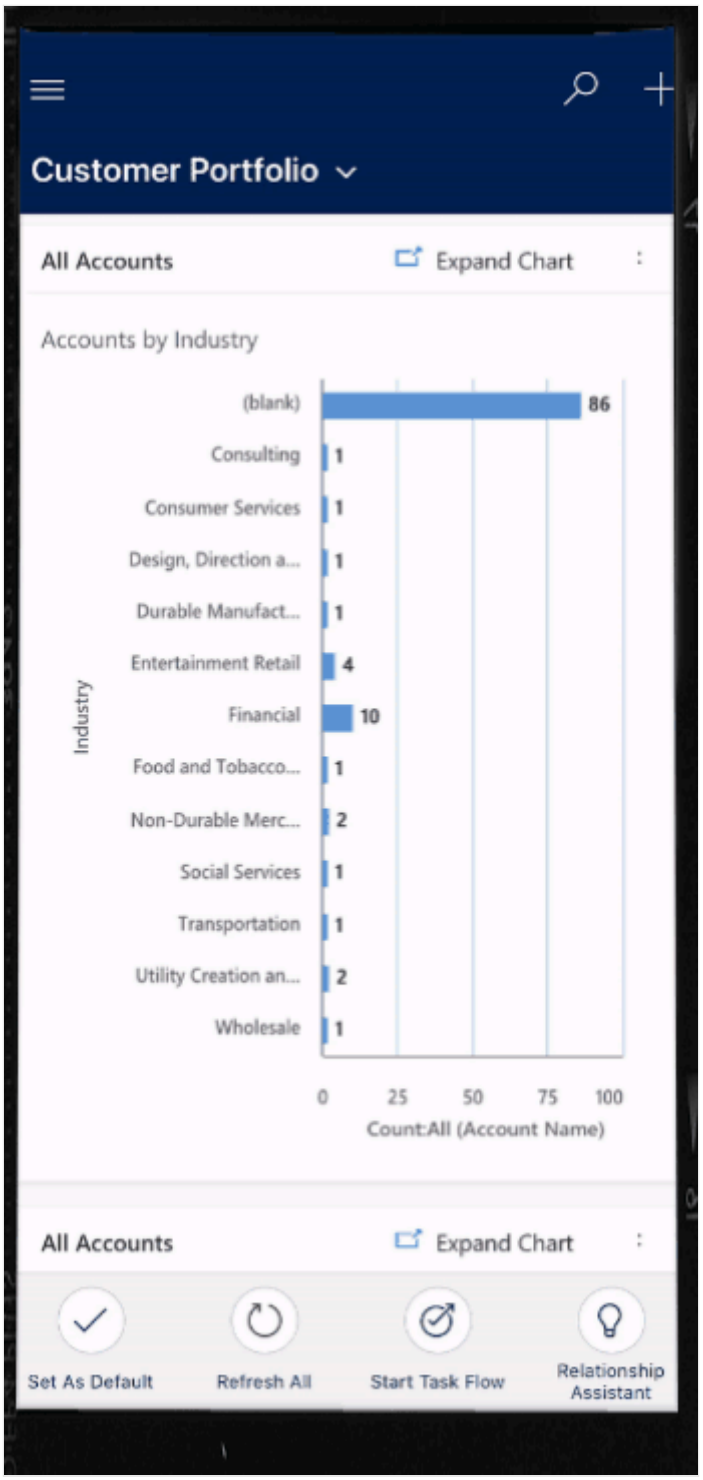
Legend:

1. **Site map**: Open the menu and move between apps, get to your favorite and recently used rows, access settings, and more.
2. **Search**: Search for app rows in Microsoft Dataverse.
3. **New**: Create a new row and quickly enter almost any type of information into the system.
4. **Assistant**: Use the assistant to monitor and track daily actions and communications. It helps you stay on top of your day with insight cards that are displayed prominently throughout the app to provide tailored and actionable insights.

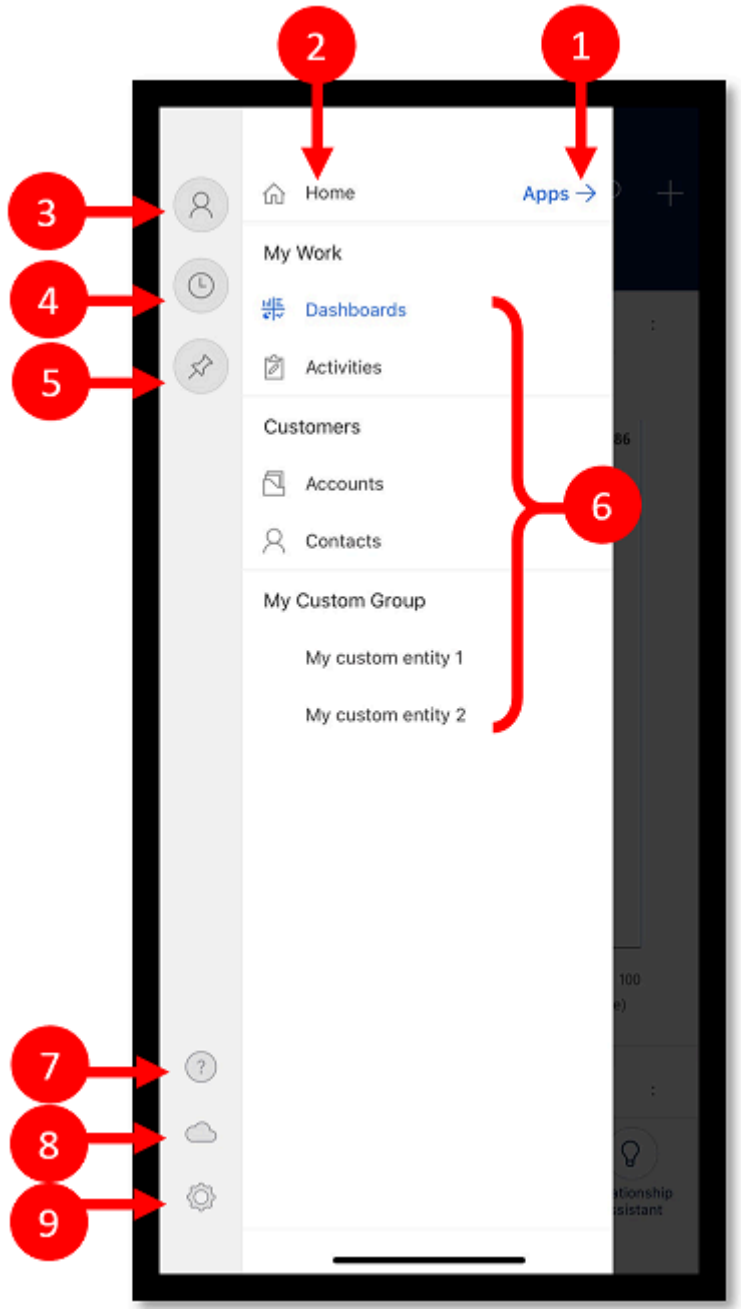
Site map

From the Home screen, select the site map to access tables, favorite or most-used rows, other apps, and settings.

Your site map might look different if your app maker has customized the navigation bar to show or hide the **Home**, **Recent**, **Pinned** buttons in the site map. An app maker can also make groups collapsible. For more information, see Hide or show the Home, Pinned, Recent, and collapsible groups.



The following illustration shows the primary navigation elements on the site map screen.



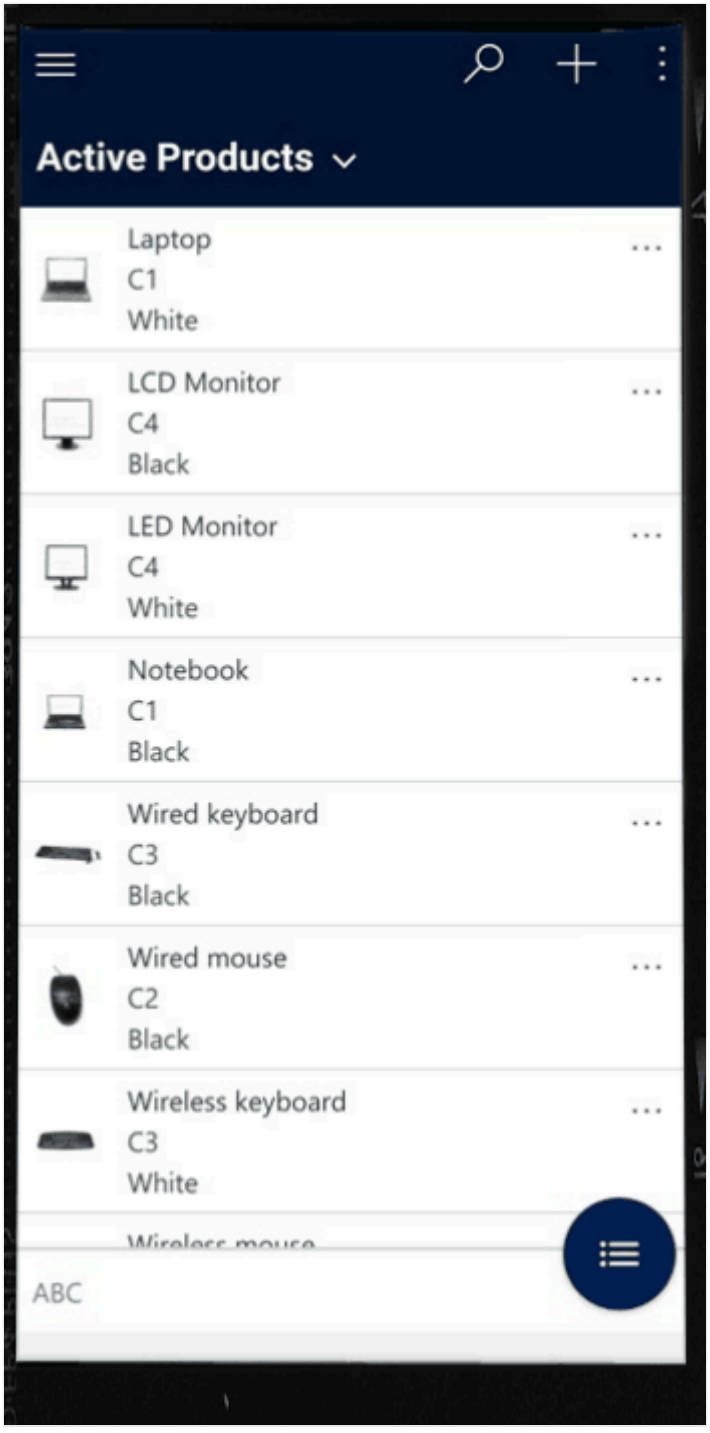
Legend

1. **App selector**: Open this menu to close your app and switch to another app.
2. **Home screen**: Select this to go back to the Home screen.
3. **Profile**: Go to the Profile screen to sign out or reconfigure the app.
4. **Recent rows**: View a list of rows you were recently using.
5. **Pinned rows**: View and open your favorite (pinned) rows.
6. **Table navigator**: This area lists the table available in the app.
7. **Help**: Access help content for more information about how to use Power Apps mobile.
8. **Offline status**: Work with your data in offline mode, even when you don't have internet access. More information: Work offline on your mobile device
9. **Settings**: Access settings.

Pin favorite rows

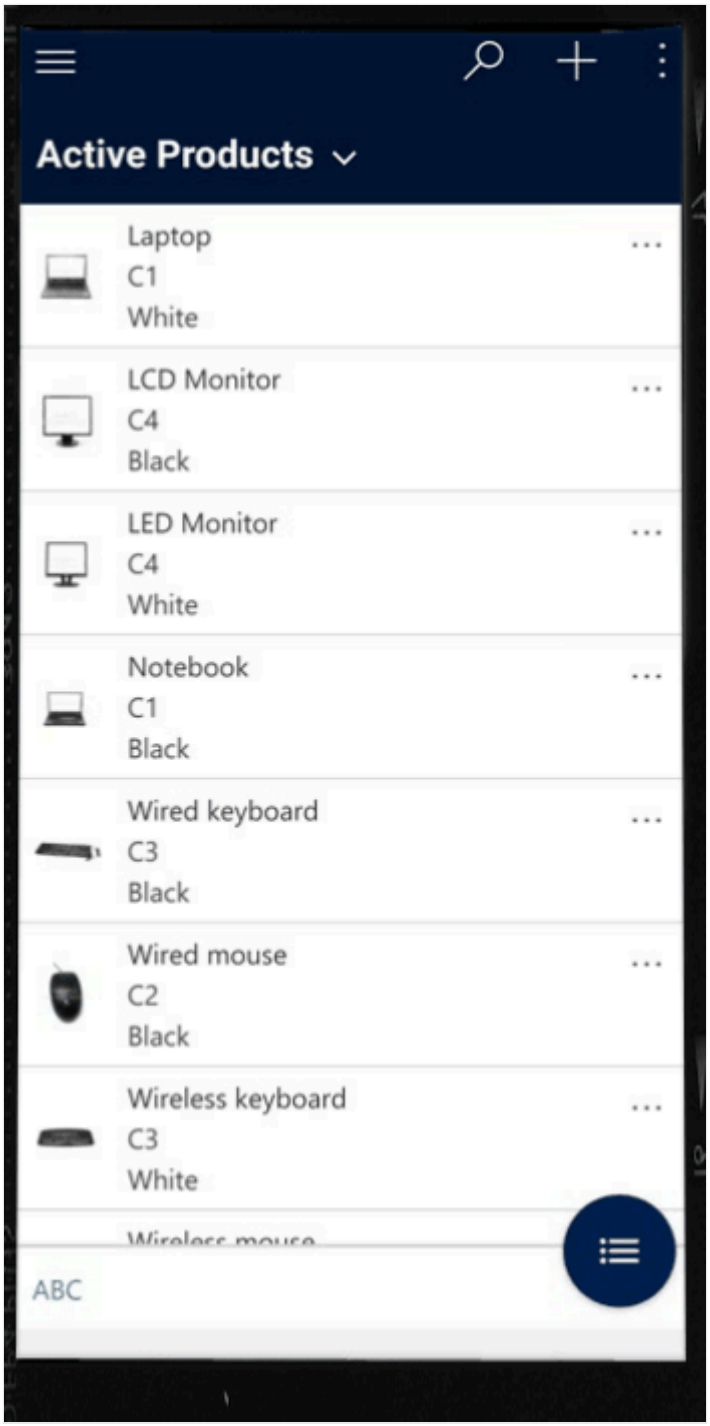
The **Pinned** and **Recent** lists provide quick access to rows that you've recently used or pinned to favorites. Use the **Recent** list to pin favorite rows.

1. From the site map , select **Recent** .
2. On the **Recent** rows screen, select the push-pin icon next to a row to add it to your favorites (pinned rows).
3. To view the newly pinned rows, select **X**, and then select **Pinned** .



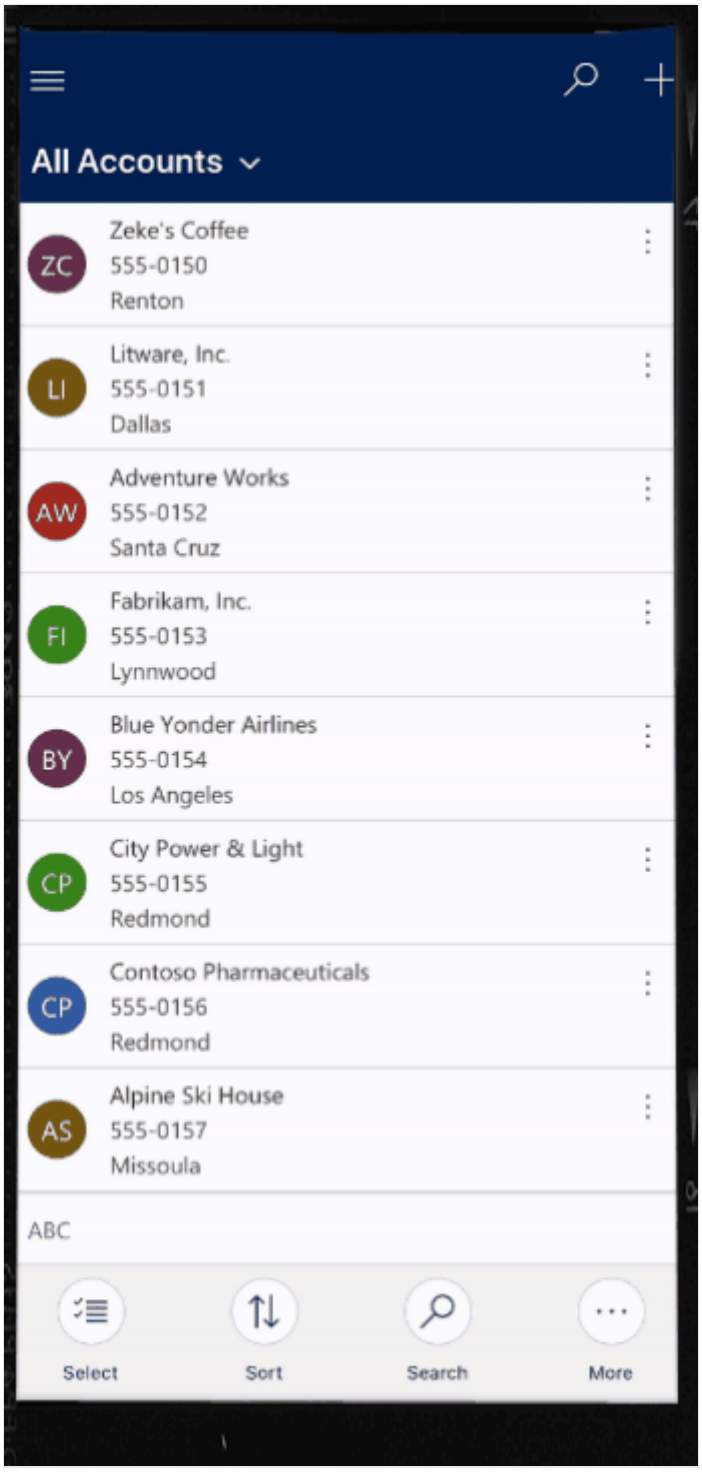
Unpin a row

1. From the site map , select **Pinned** .
2. Select the remove-pin icon next to a row to remove it from favorites (pinned rows).



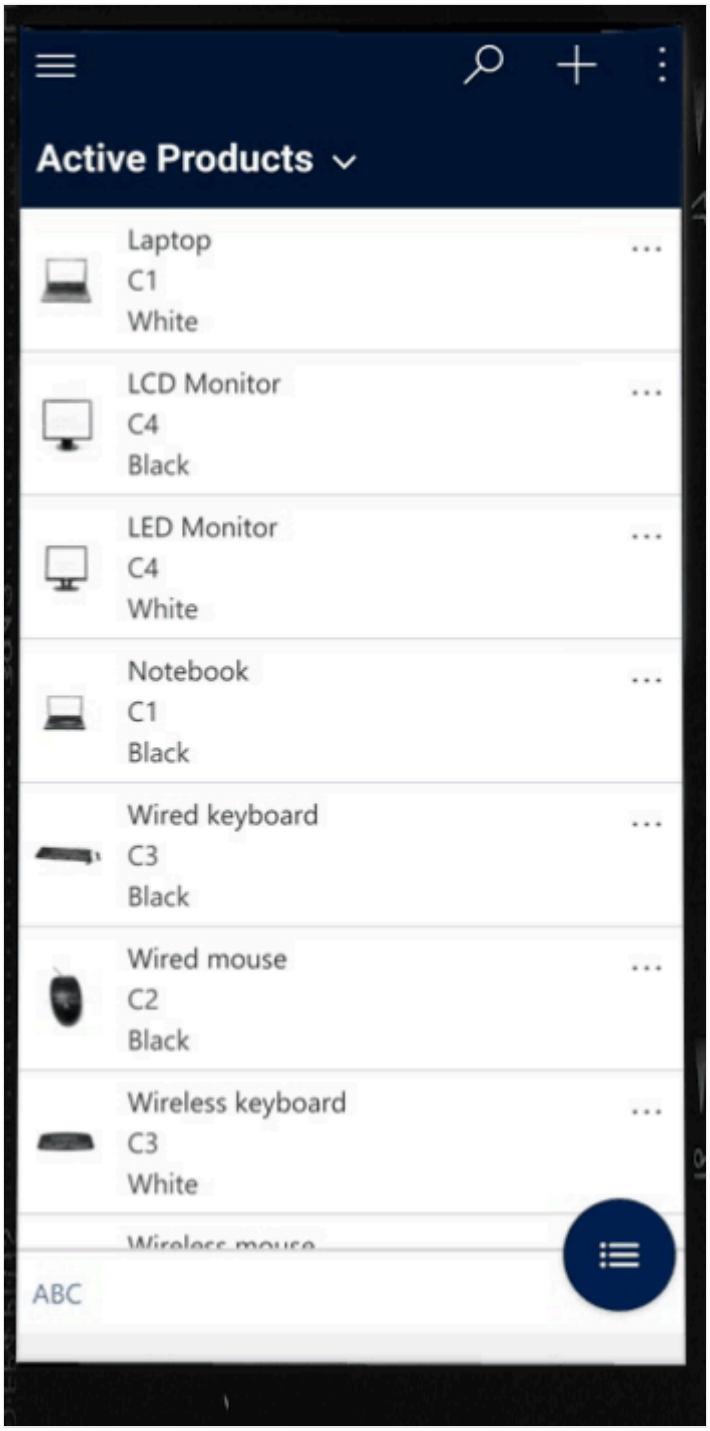
Change views

- From the Home screen, select the down arrow next to the current view, and then select a new view.

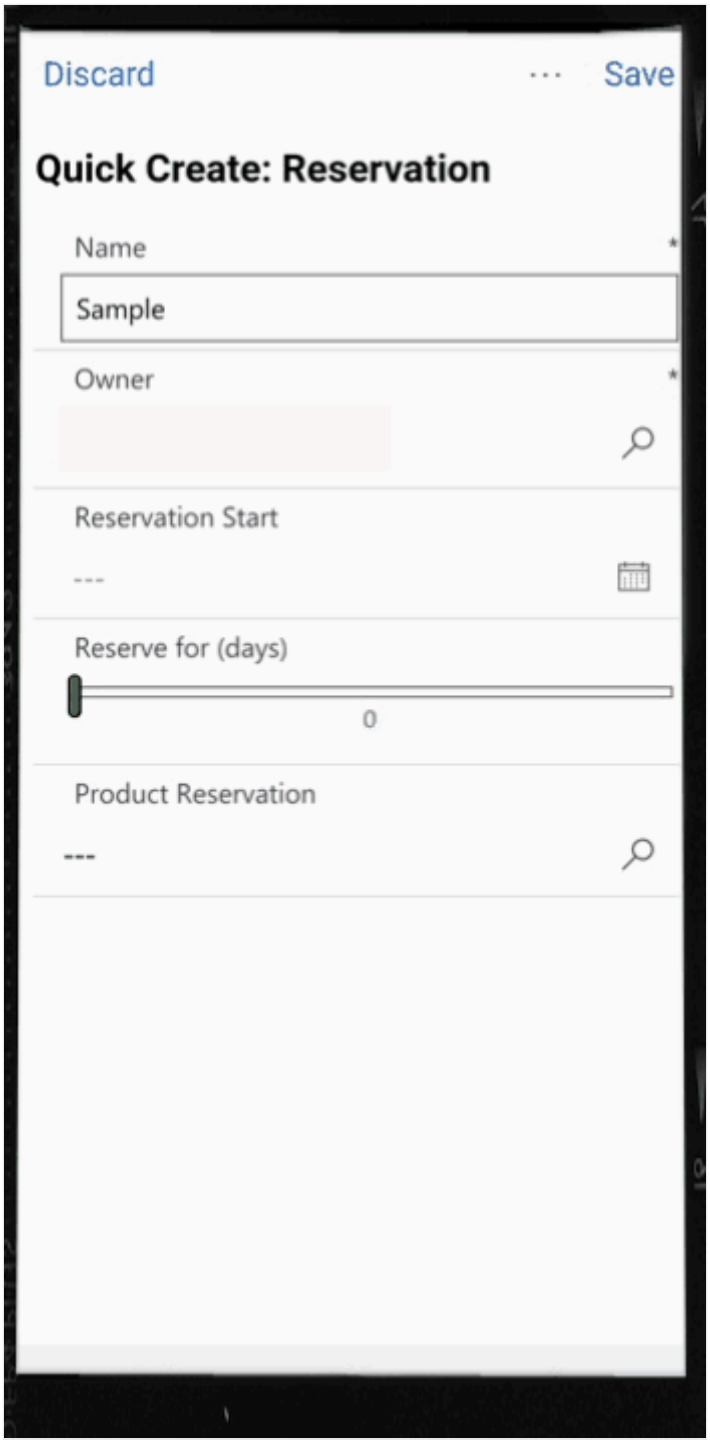


Add a row quickly

1. From the Home screen, select **New** .
2. Fill in the columns, and then select **Save**.
3. After the row is created, you can view the new row.

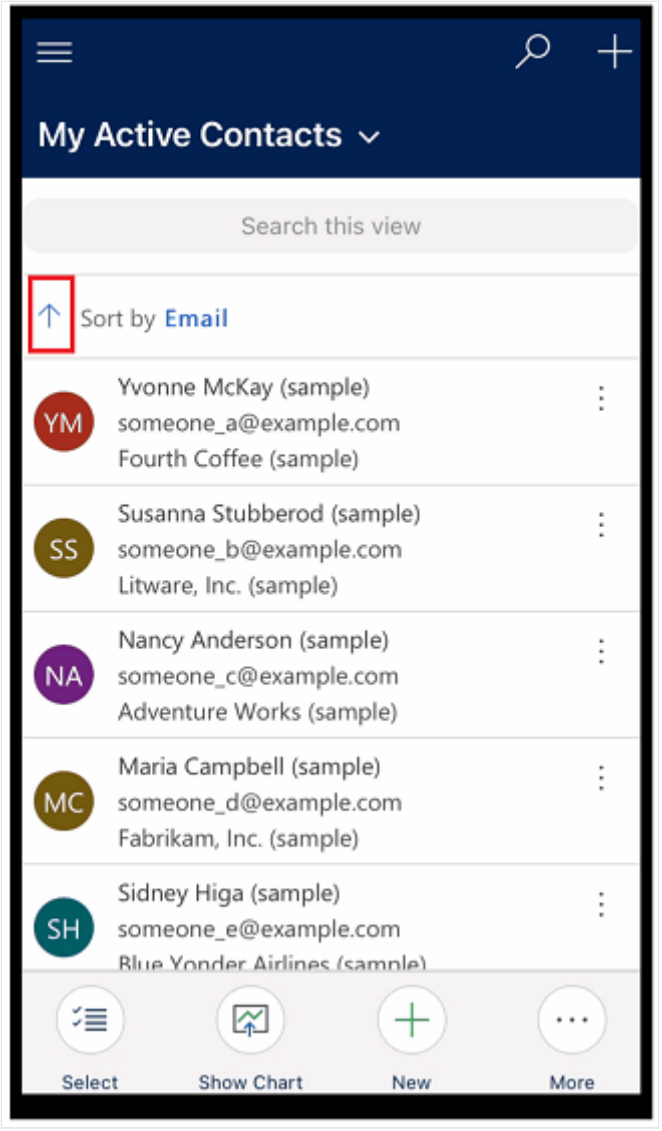


- To save and open the row that you created, select **More** ..., and then select **Save and Open**.
- To save and create another row, select **More** ..., and then select **Save and Create new**.

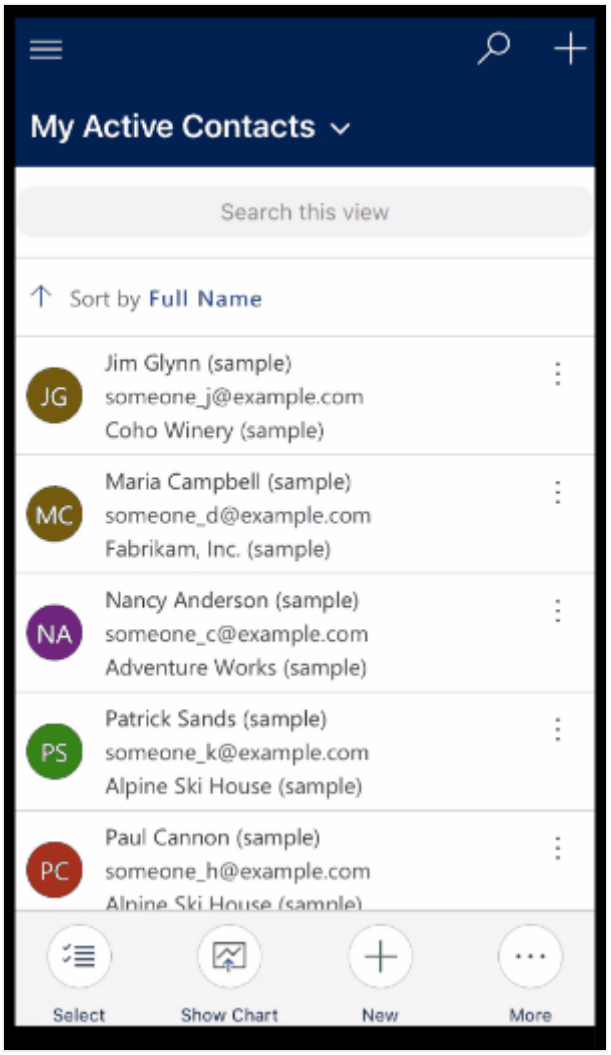


Sort rows

Sort in ascending or descending order: From a list view, select the arrow to sort the list in ascending or descending order.

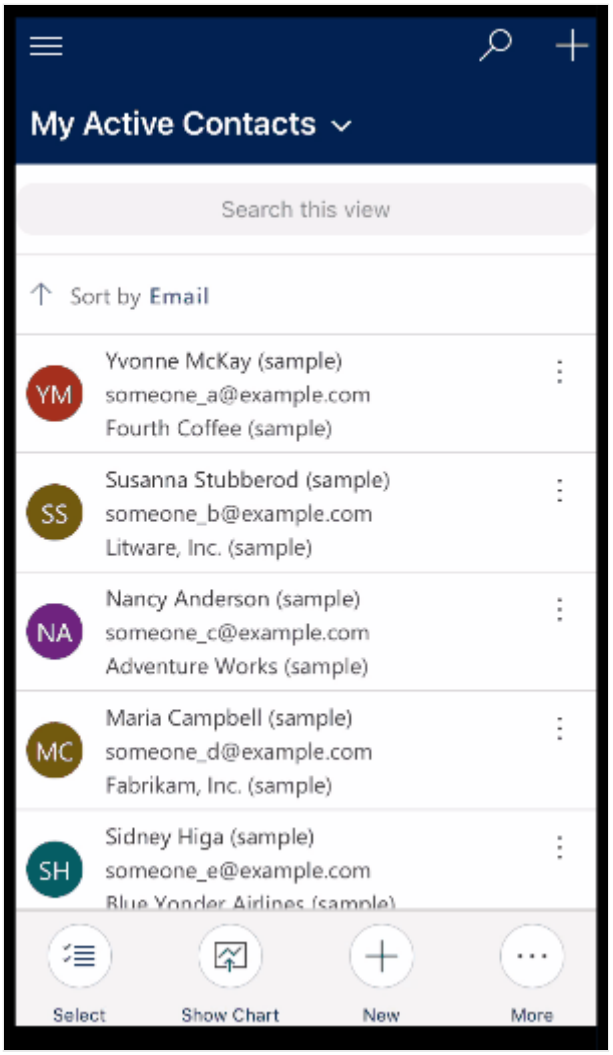


Sort by field: Select the current **Sort by** field and then select another field to sort by.



Access the actions menu

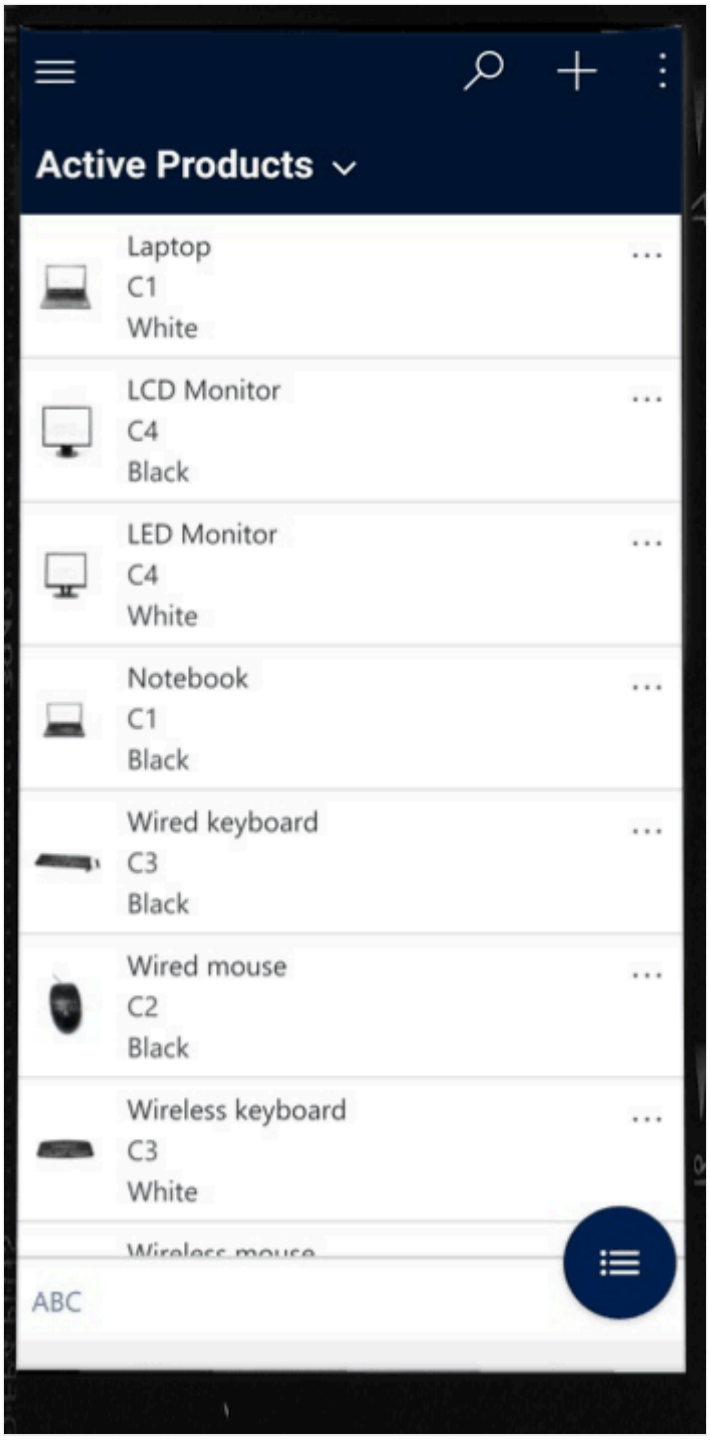
From a list view, swipe left to access the actions menu for a row.



> **Note**
>
> The Flow actions menu in Power Apps mobile doesn't support flows created in a solution.

Access more commands (Android)

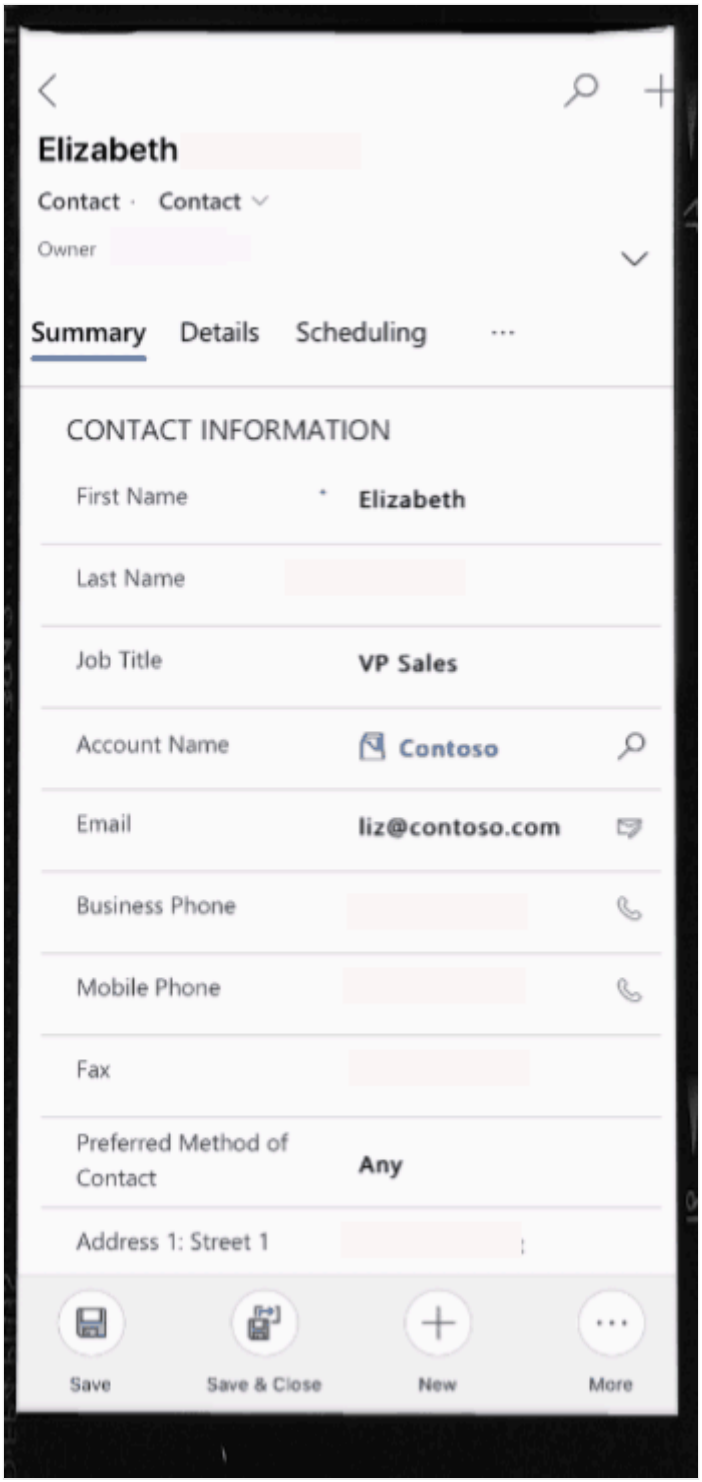
1. From the Home screen, open a row.
2. On the open row, select **More** to access more commands.



Edit a row

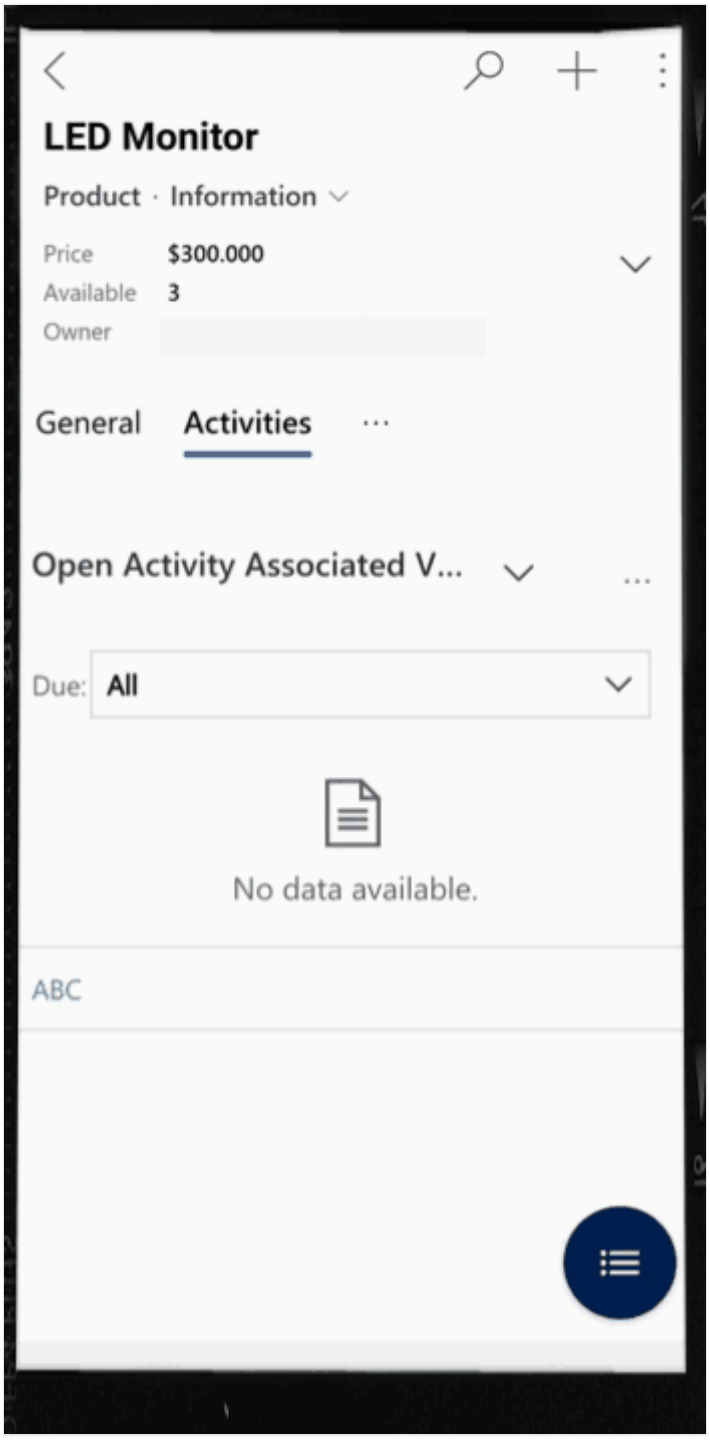
1. From the Home screen, open a row that you want to edit.

2. When you're done editing the row, select **Save**. To cancel your changes, select **Discard**.



Go back to the Home screen

- To get back to the Home screen when you're in a row, select **Back** .
- At any point, press and hold **Back** to go back to the Home screen.



Sign out

From the site map , select the profile icon , and then select **Sign out**.

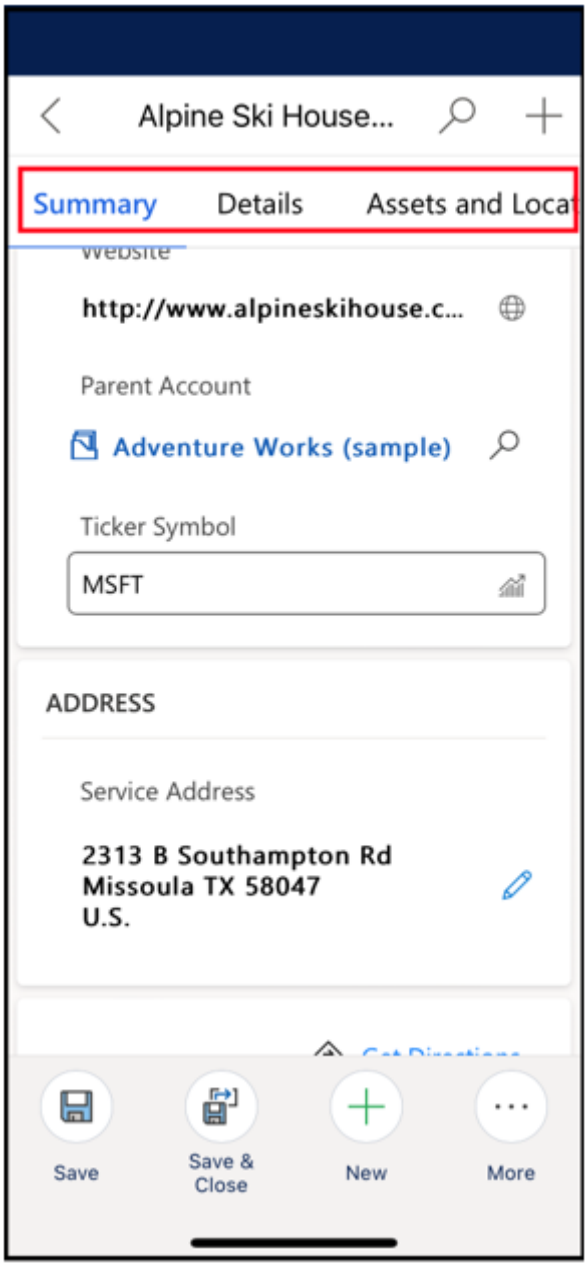
Enhancements (preview)

[This section has pre-release documentation and is subject to change.]

This section describes enhancements that have been made to model-driven apps on mobile devices.

Lock tabs at the top of forms

Your app maker can choose to lock the tabs at the top of a form so they are always visible while users scroll through the data on the form.



Your app maker can turn on this feature. For more inforamtion, see Features.

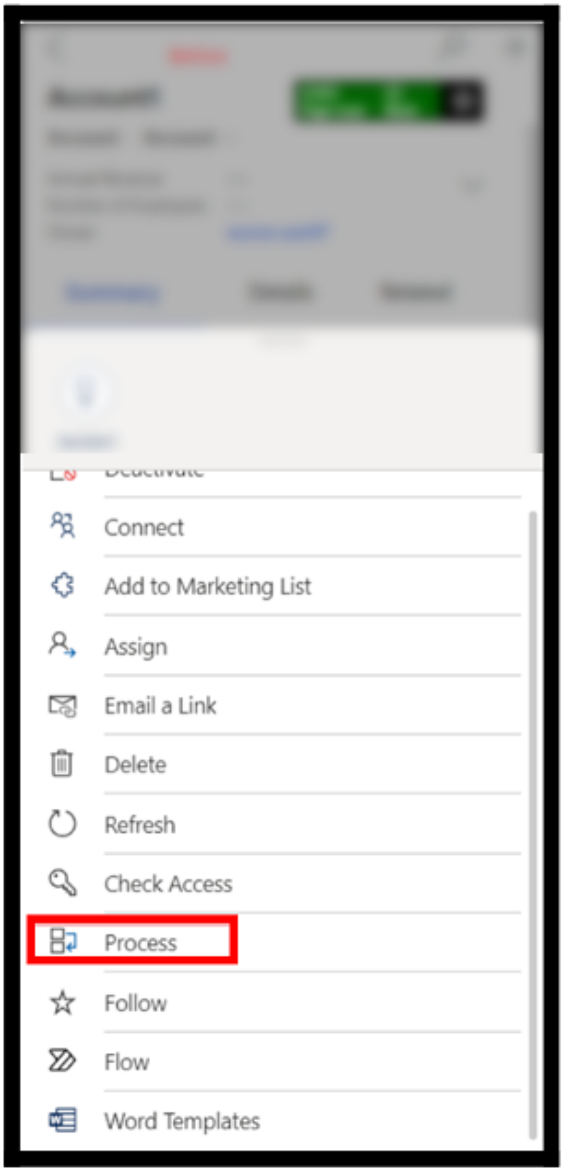
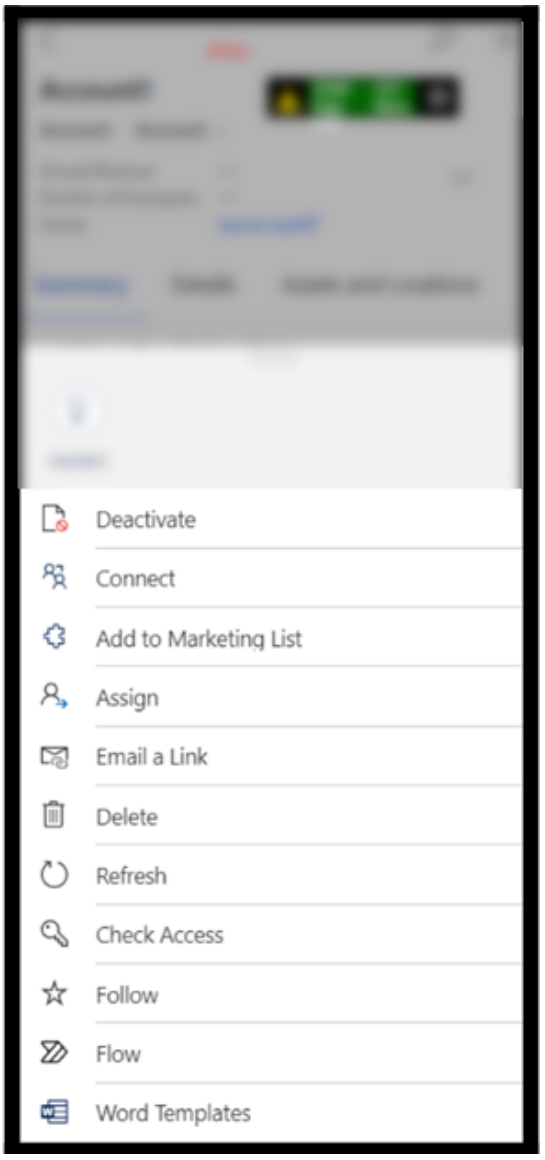
Mobile commanding improvements

Your app maker can enable Mobile commanding improvements to provide easy access to contextual commands when you're using Power Apps mobile. Here's a list of improvements:

- The **Delete** command on a grid page is automatically hidden when a row is not selected thus, making room for other commands. When one or more rows is selected the **Delete** command is automatically displayed.

- The **Process** command is hidden when there's no process enabled for a table. Removing a nonfunctional command makes room for other commands.

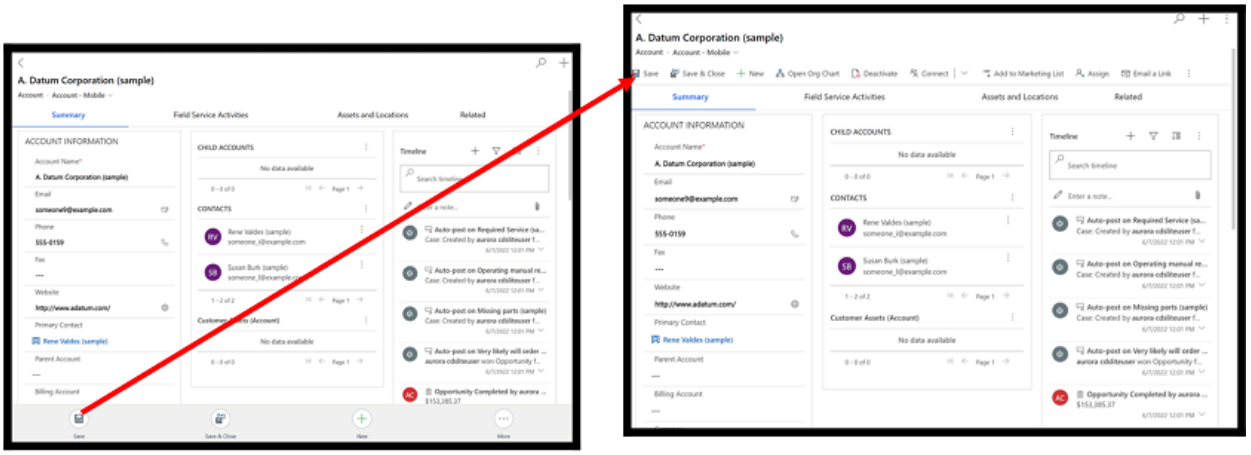
Expand table

Before	After
	

- Some commands have been moved from the main set of commands to an overflow menu. This gives priority to other commands, including custom commands.

Tablet optimization for command bar

The command bar on Power Apps mobile for tablets was at the bottom, now the command bar is located at the top.



Your app maker can turn on this feature. For more inforamtion, see Features.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Dataverse search for model-driven apps in the Power Apps mobile app

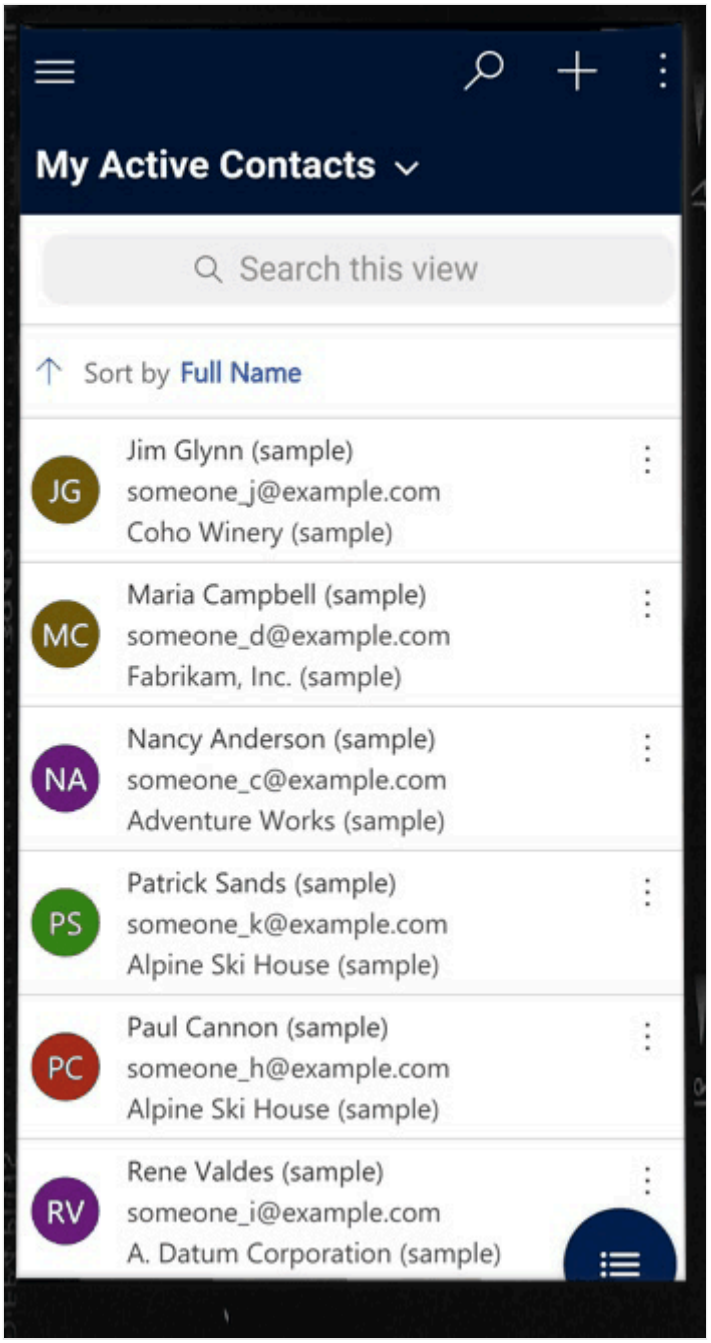
Article • 01/28/2023

You can quickly find the information that you're looking for by using Dataverse search. The Dataverse search experience delivers fast and comprehensive results across multiple tables, in a single list, sorted by relevance.

Dataverse search is already available when you're using a model-driven app in a web browser. The functionality is the same in Power Apps Mobile, but it works a little differently. More information: Search for tables and rows by using Dataverse search

Before you begin using Dataverse search, the new experience needs to be enabled by the administrator for your organization. More information: Enable Dataverse search

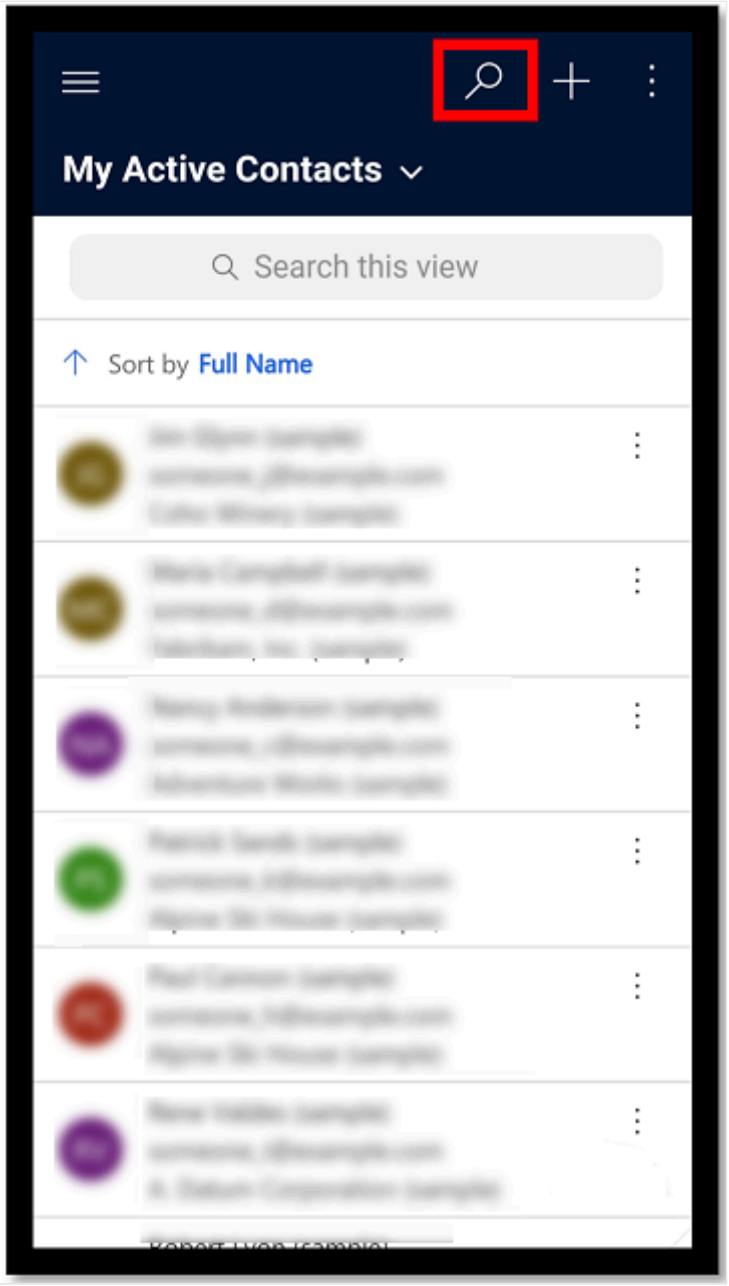
When Dataverse search is turned on, it becomes your default and only global search experience for all of your model-driven apps. You won't be able to switch to quick find search also known as categorized search.



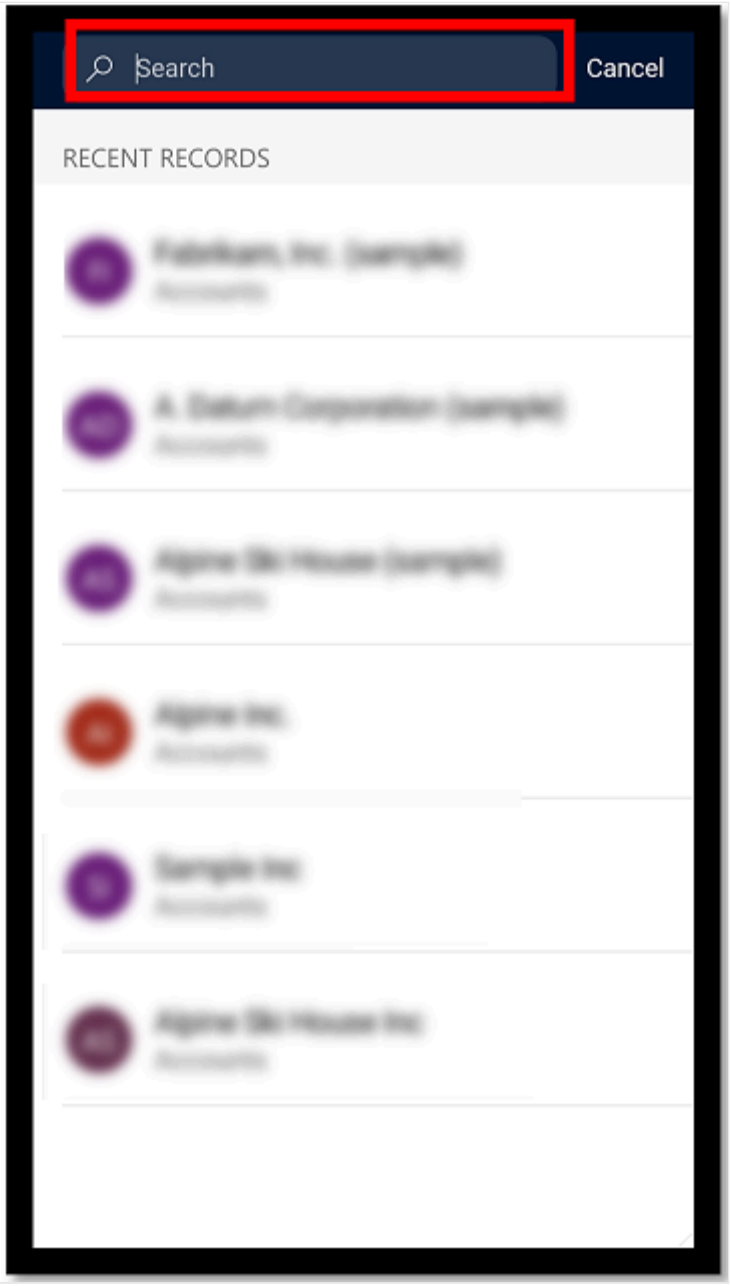
Run a search

View suggested search results inline as you type, minimizing keystrokes and simplifying page navigation.

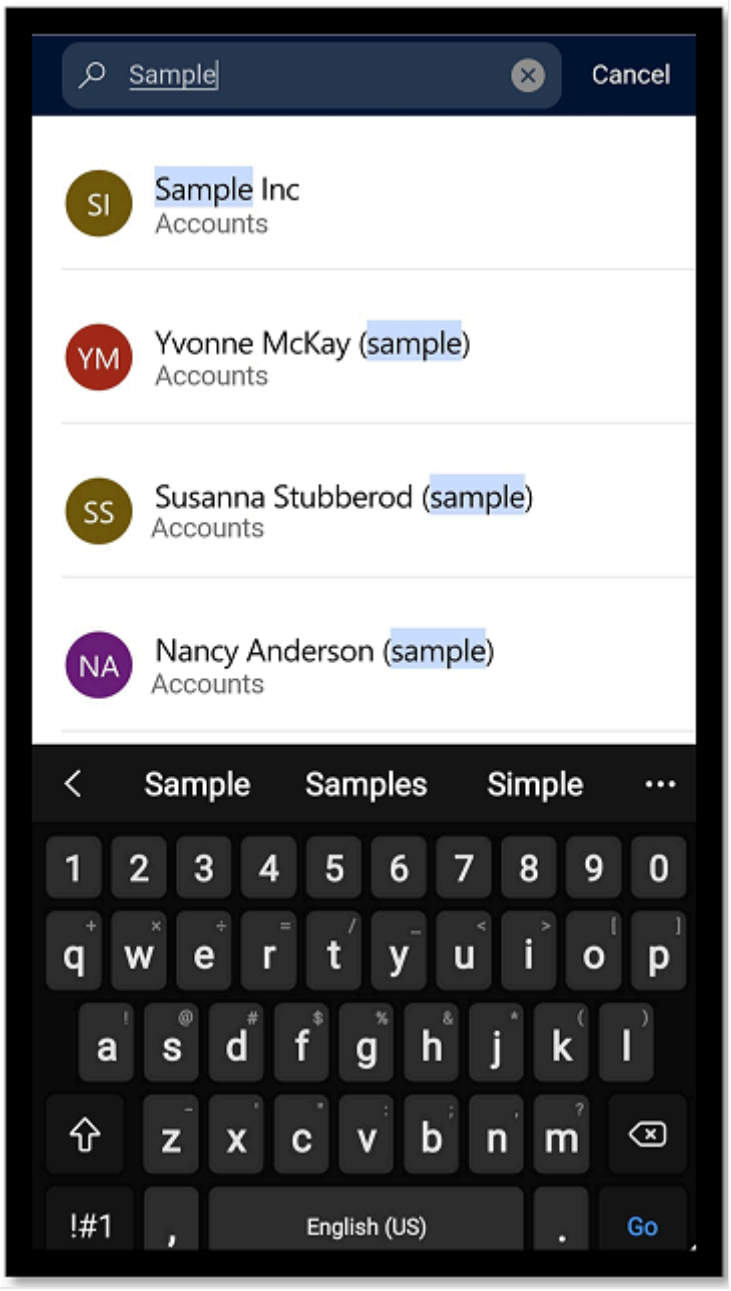
1. Select the search button.



2. Enter the words you want to search for in the search box.



3. As you enter search text, rows that match your text begin to appear.



Early access: Use the barcode scanner

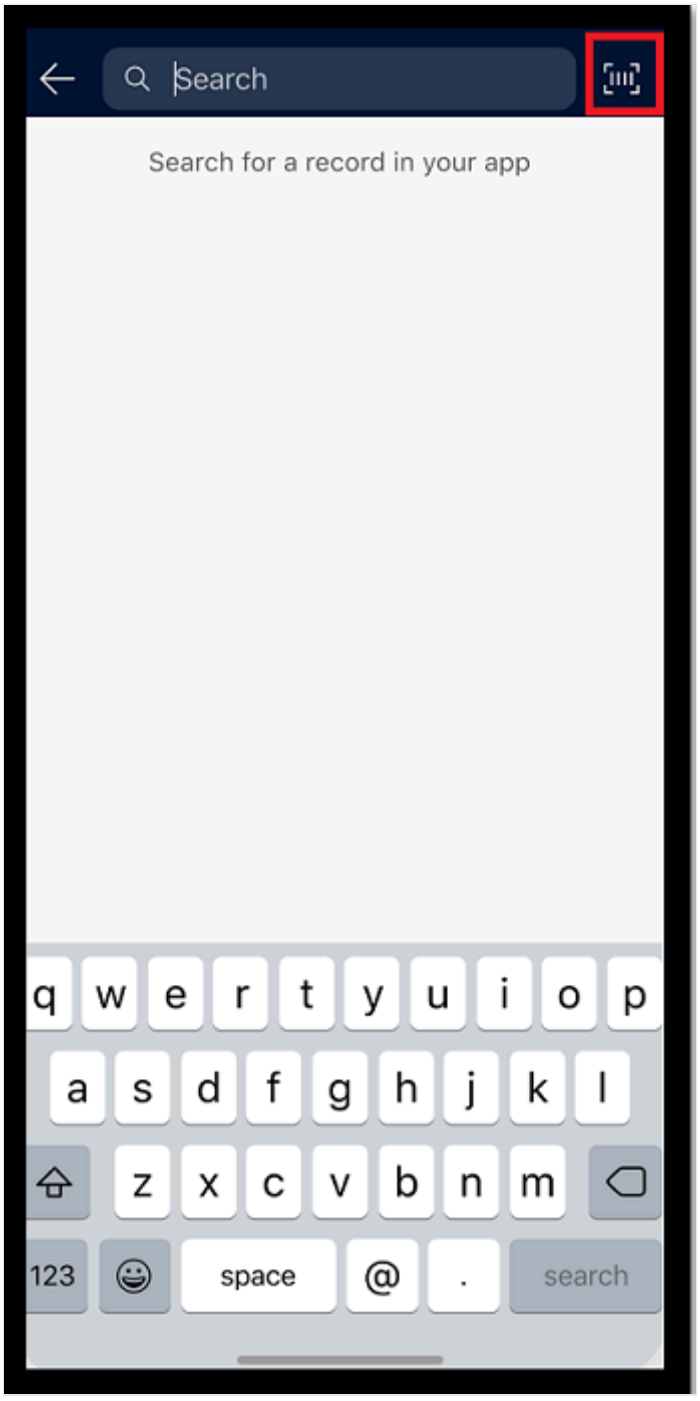
[This topic is pre-release documentation and is subject to change.]

If you're on early access, you can use the barcode scanner to look up rows. The scanner automatically detects a barcode, a QR code, or a data-matrix code.

You also enter text in the search box such as **contoso** and then use the barcode scanner to scan a barcode, such as **12345**. In this case, the system will search for **contoso12345**.

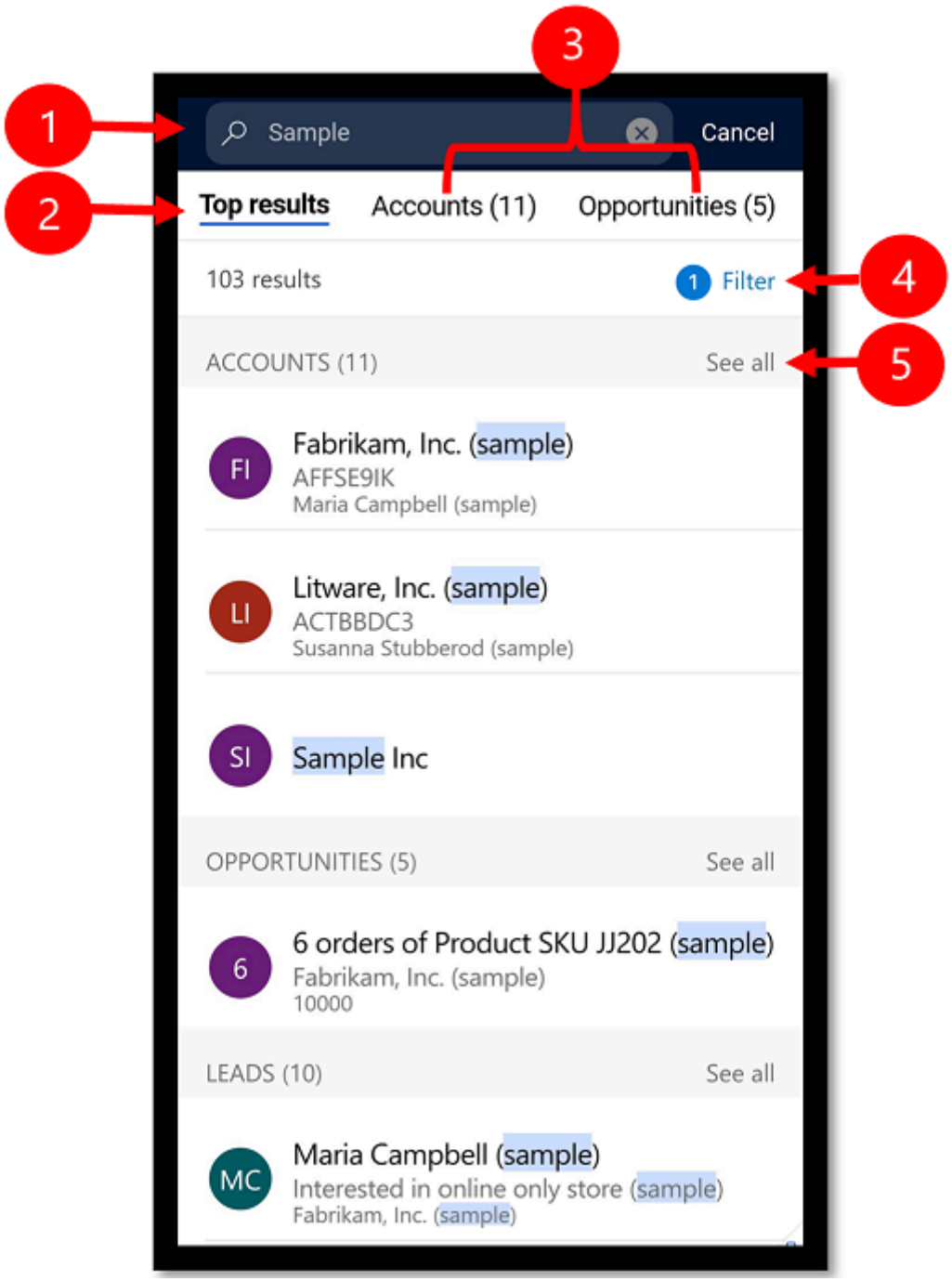
1. To use the barcode scanner, select the search button.

2. Select the barcode scanner button and then scan your item.



View search results

Search results are ranked and grouped by table.

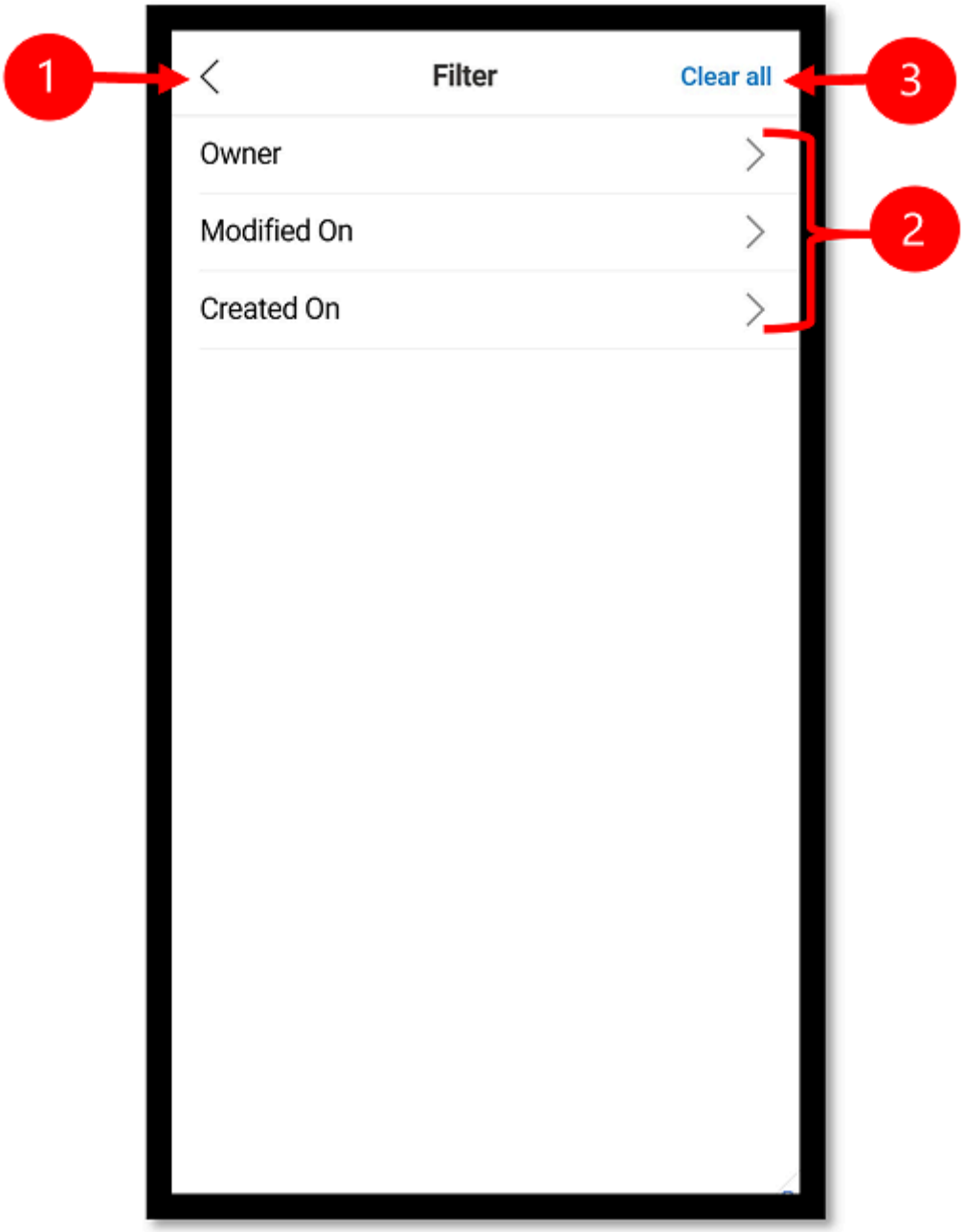


Legend

1. **Search box**: Enter the words you want to search for.
2. **Top results**: This tab shows the rows that best match the search query.
3. **Table-specific tab**: To narrow your search results to a specific table, select the table tab.
4. **Filter**: Refine the search results by using filters. When you select filters, the number of tables that have been filtered on appears next to the filter option.
5. **See all**: Up to three rows will appear; if there are more, select **See all**.

Filter search results

Use filters to drill into and explore the results of your current search without having to repeatedly refine your search terms. Immediately after you perform a search, you can filter by **Owner**, **Created On**, or **Modified On**.



Legend

1. **Back**: Go back to the search results screen.
2. **Filter**: Select a filter type to filter the search results.
3. **Clear all**: Clear all filters and return to the search results screen.

Feedback

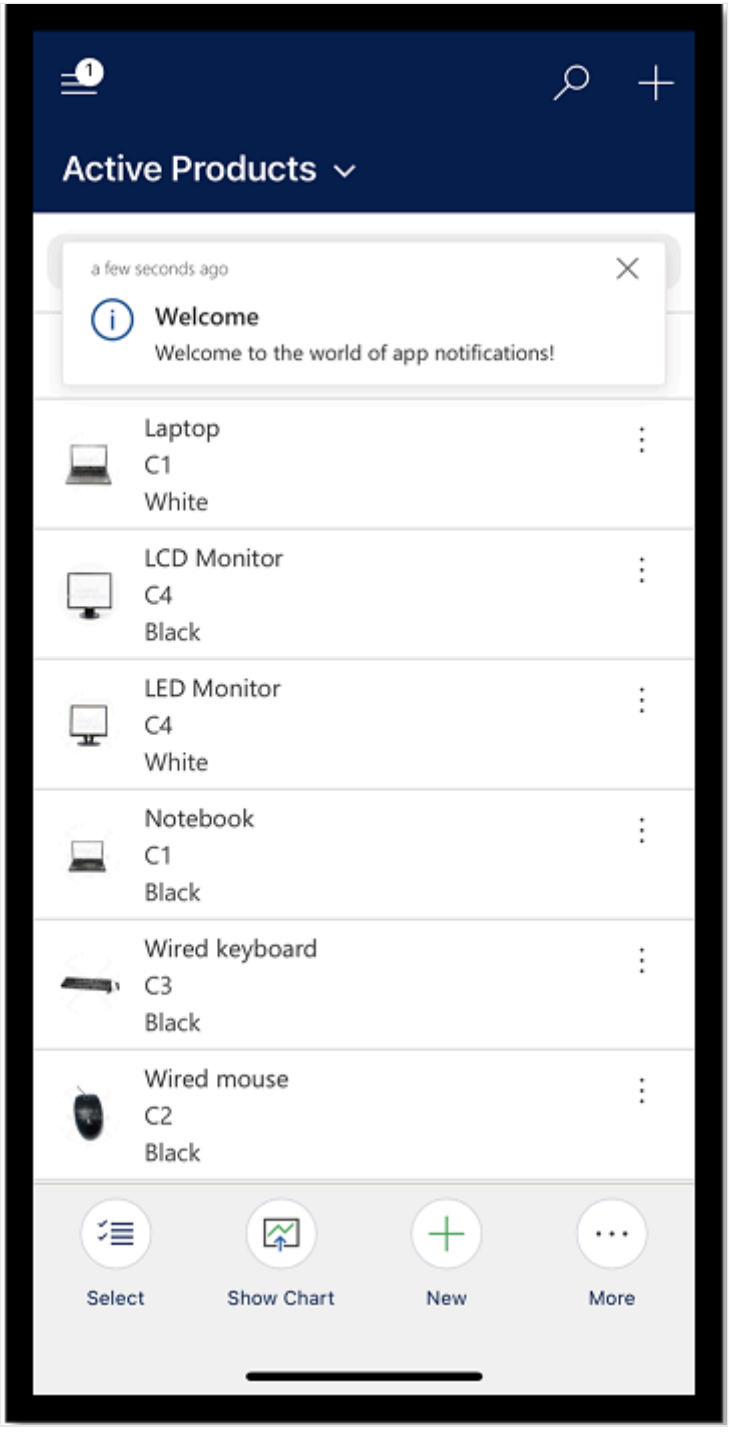
Was this page helpful? Yes No

Provide product feedback

In-app notifications in the Power Apps mobile app

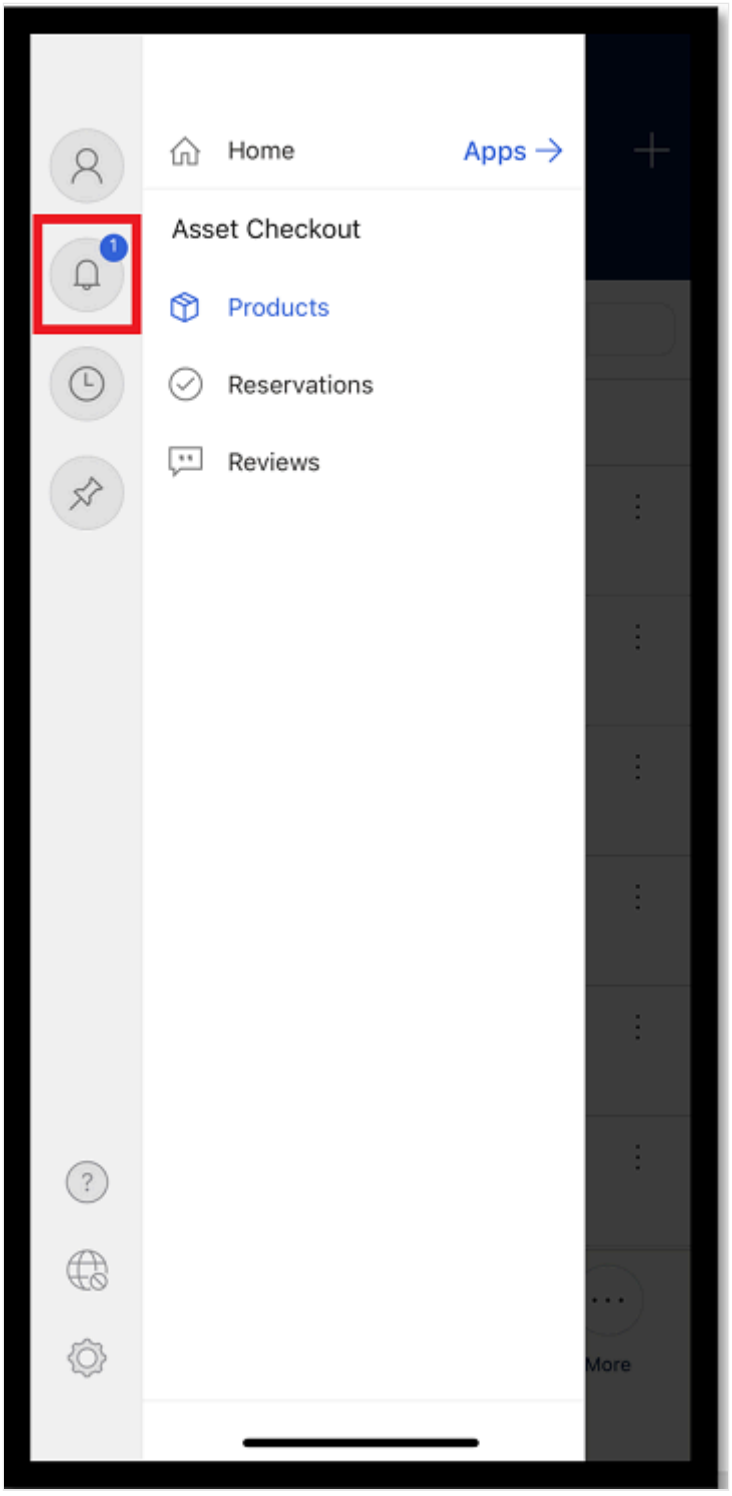
Article • 01/28/2023

In-app notifications allow users to receive notifications in the notification center and as a notification toast. For more information, see In-app notifications in model-driven apps. In Power Apps mobile, toast notifications appear one at a time.

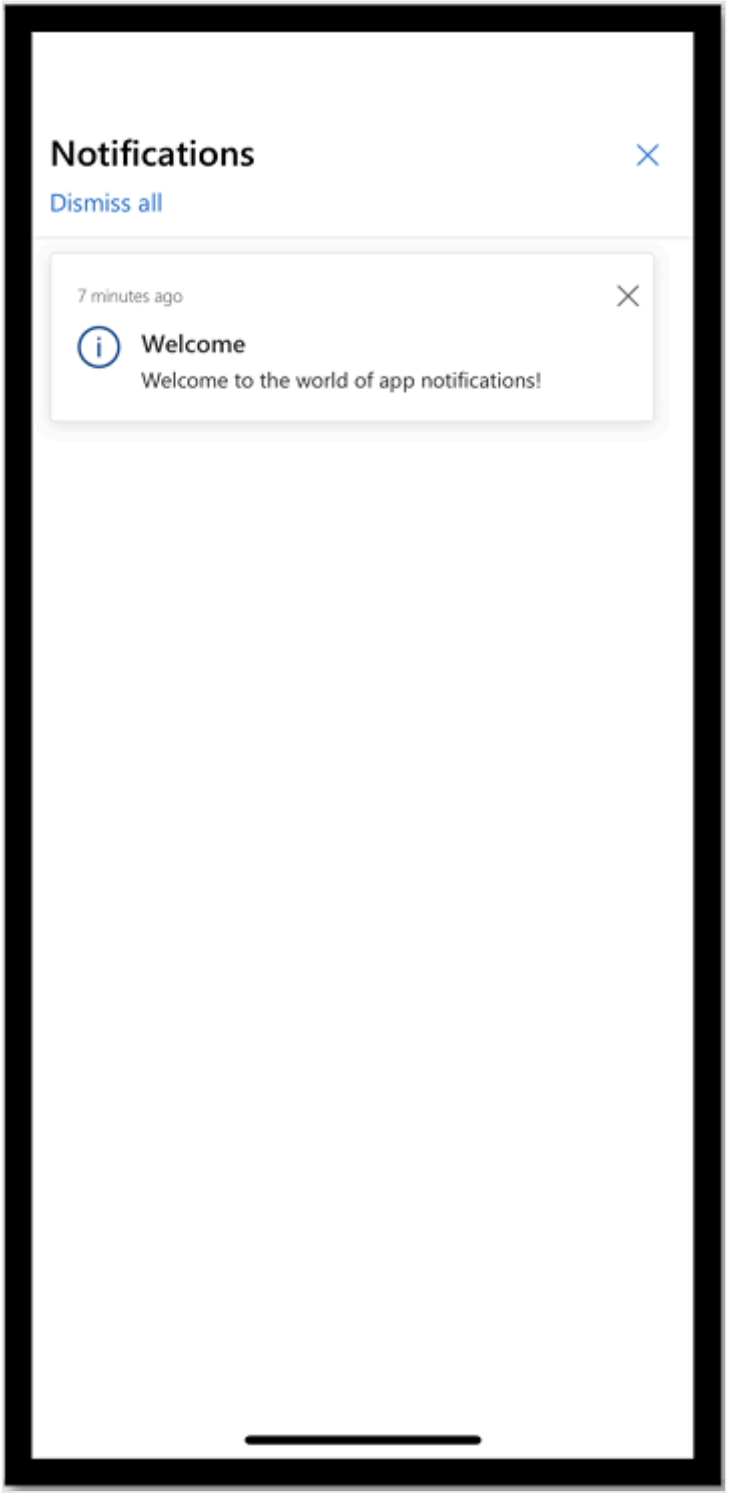


Access notifications

1. To access notifications in Power Apps mobile, go to site map and then select the bell icon.



2. The notification center opens which displays all your notifications.

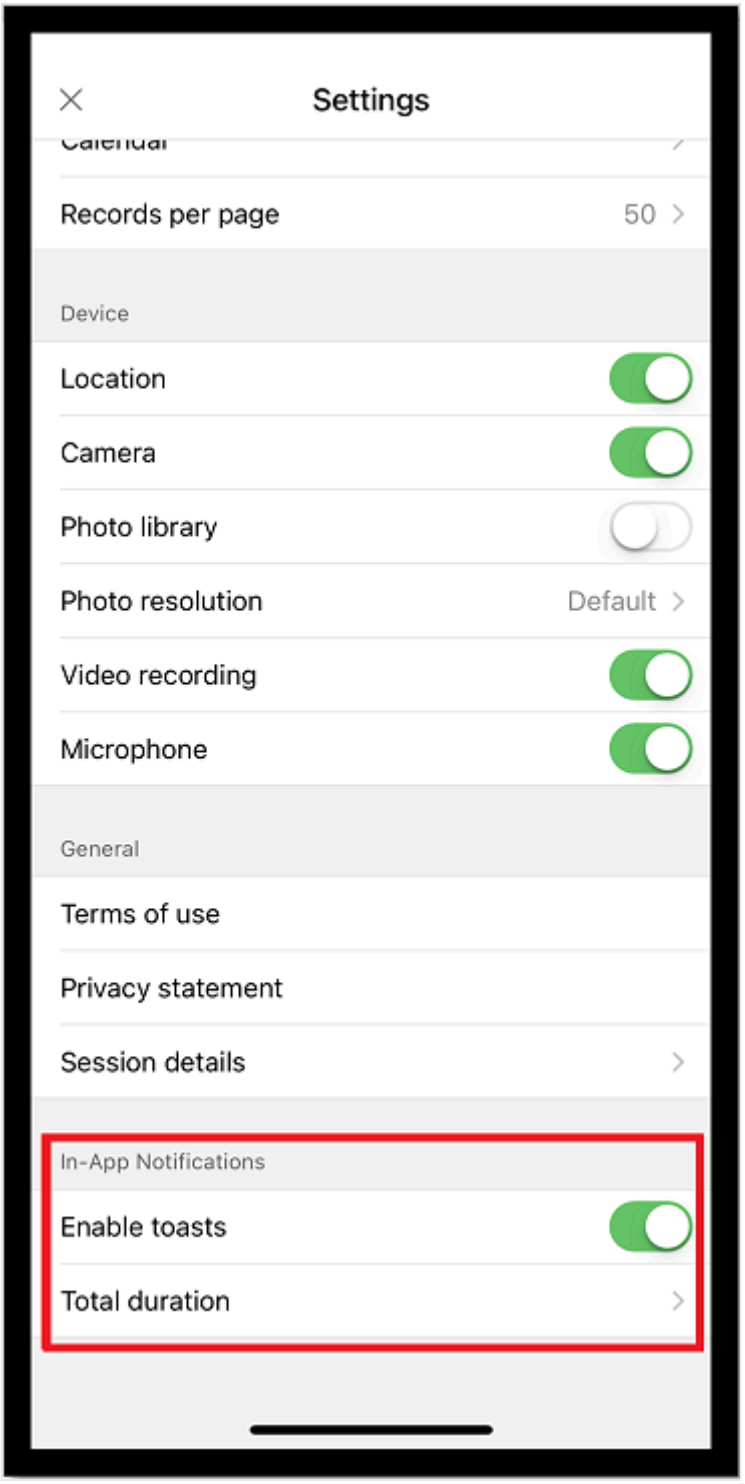


3. To dismiss and delete a notification, select the close **X** button on a notification. Or, select **Dismiss all** to dismiss and delete all notifications.

Turn toast notifications on or off

1. To turn toast notifications on or off, from the site map, go to **Settings**
2. Under **In-App Notifications**, go to the **Enable toasts** and move the toggle to on or off.

3. If you've enabled toast notifications, then select **Total duration** and enter how long you want the toast to appear.



Feedback

Was this page helpful? Yes No

Provide product feedback

Create push notifications for the Power Apps mobile app using Power Automate

Article • 03/14/2023

Push notifications are used in Power Apps mobile to engage app users and help them prioritize key tasks. In Power Apps, you can create notifications for Power Apps mobile by using the Power Apps Notification connector. You can send notifications to any app that you create in Power Apps.



Add a push notification to your app if:

- Your users need to know information immediately.
- Your users must complete important tasks by using your app, in a preloaded context.
- You want to engage your users on a specific interval, or you need users to enter the app in a specific context.

> **Note**
>
> To receive push notification, each user must have opened the app in Power Apps Mobile once or gotten the app from the Microsoft 365 apps page.

Before you can create push notification you need to have access to an app and have the row ID if you're creating a notification for a form.

Create an app

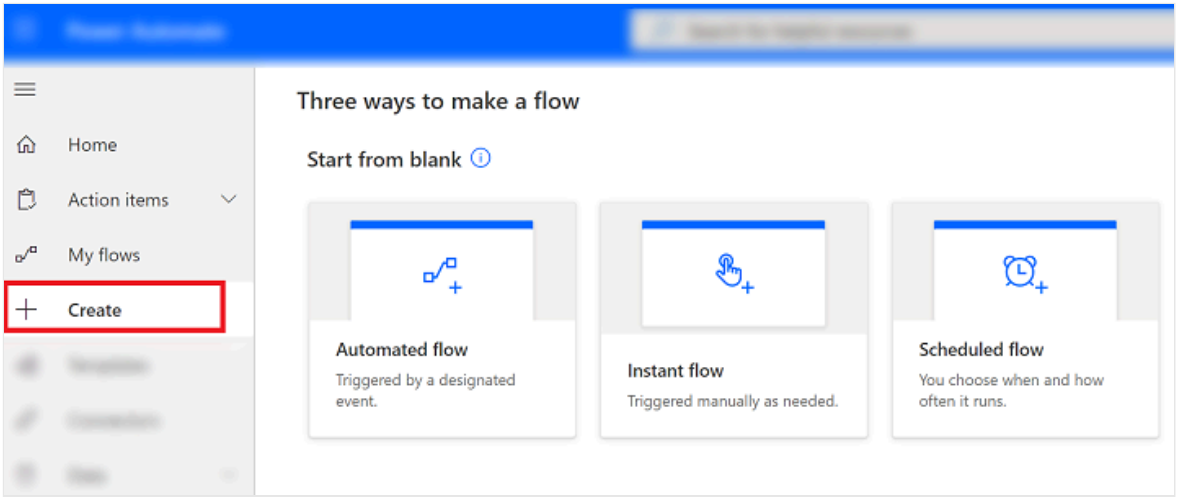
You need to have **Contributor** permission for a model-driven app or canvas app. If you don't have an app, you can create one. For information, see:

- Create a model-drive app
- Create a canvas app

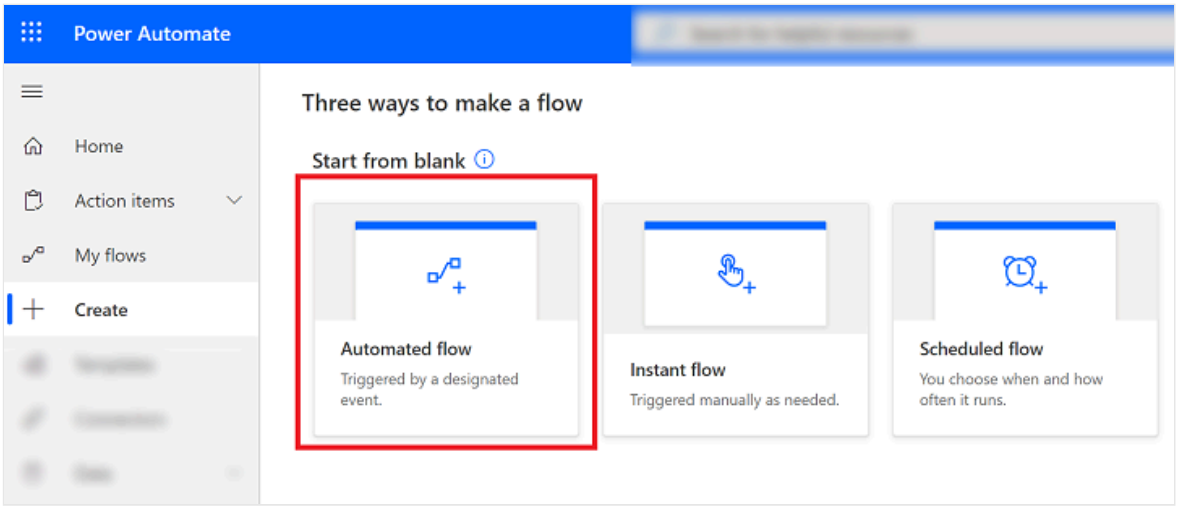
Create a notification from a flow

When you trigger a push notification from a flow, you can send the notification to only one user or security group at a time.

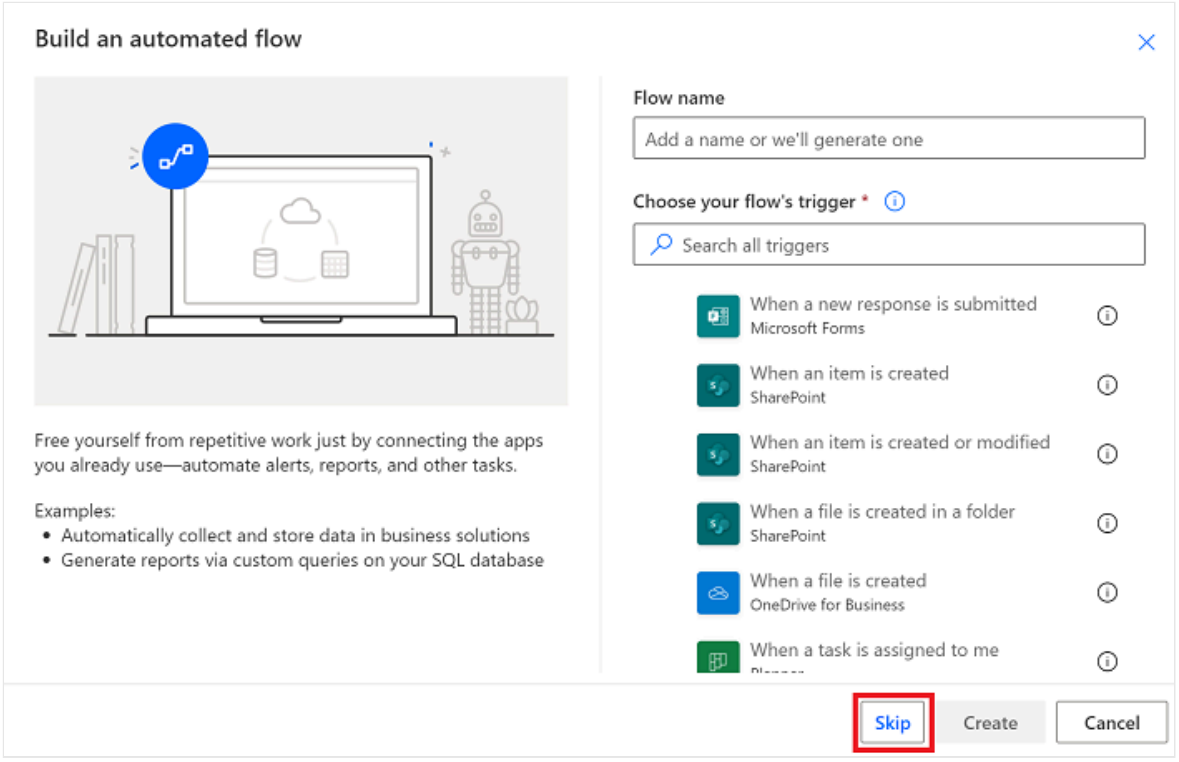
1. Go to Power Automate and select **Create**.



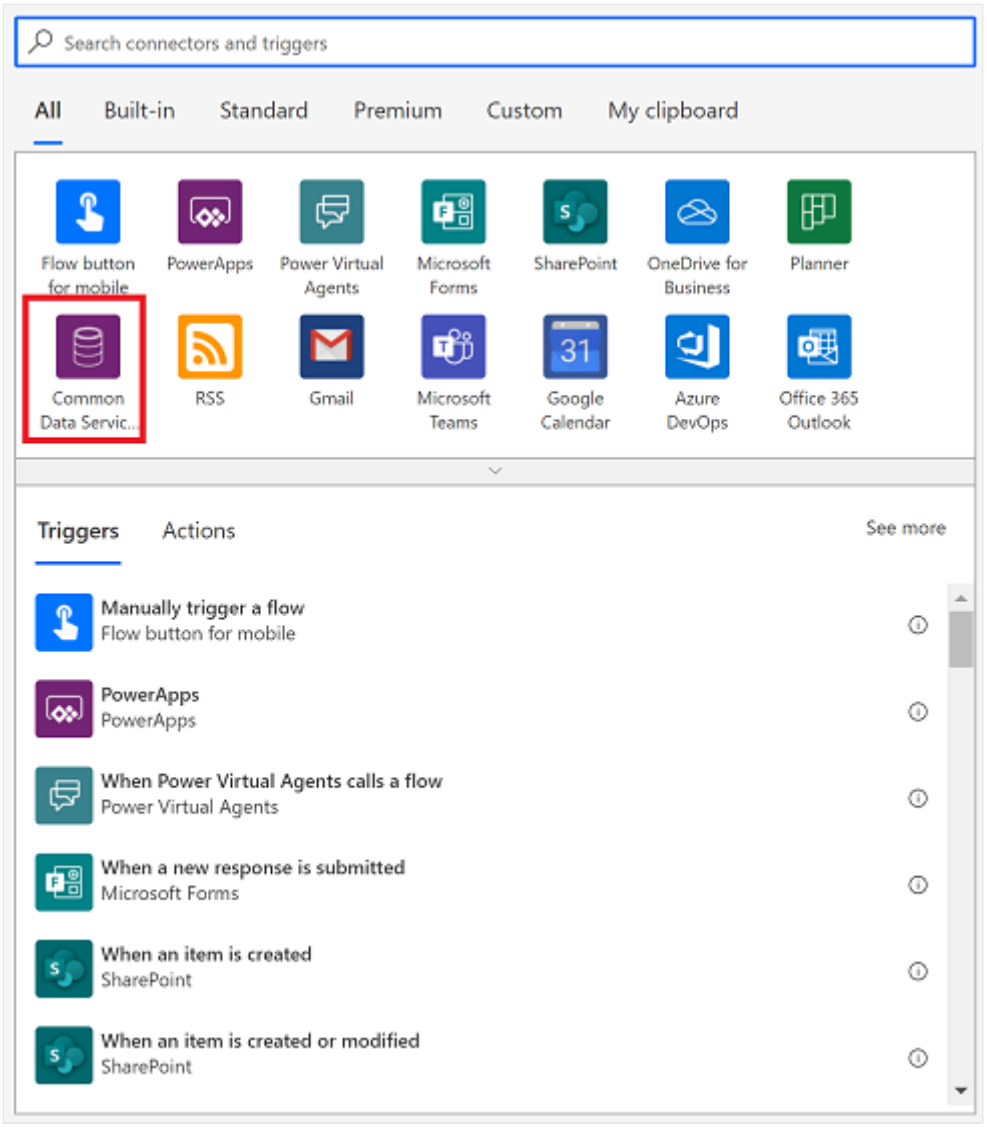
2. Select **Automated flow**.



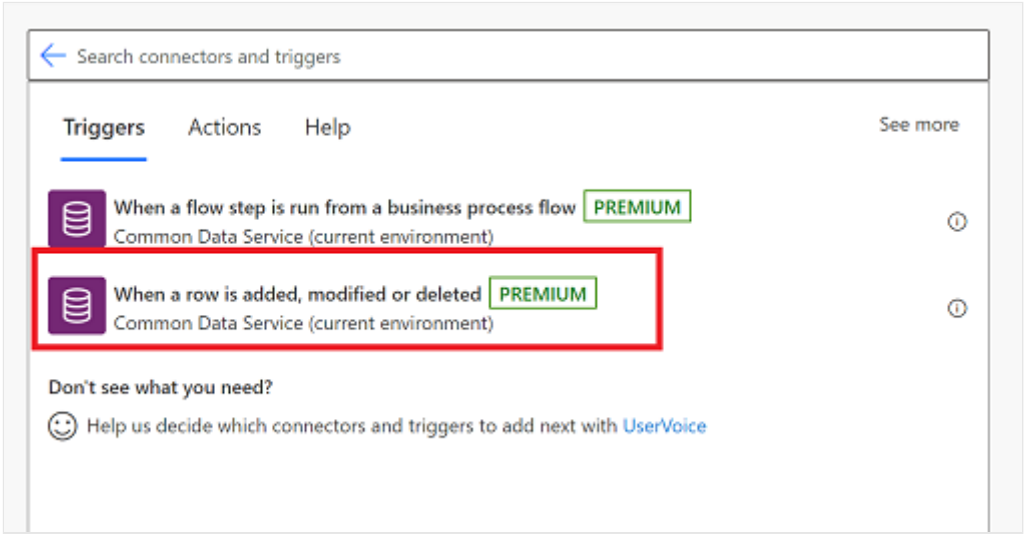
3. On the **Build an automated flow** screen, choose one of the flow trigger or select **Skip** and manually create one.



4. From the list of connectors and triggers select **Microsoft Dataverse (current environment)**.



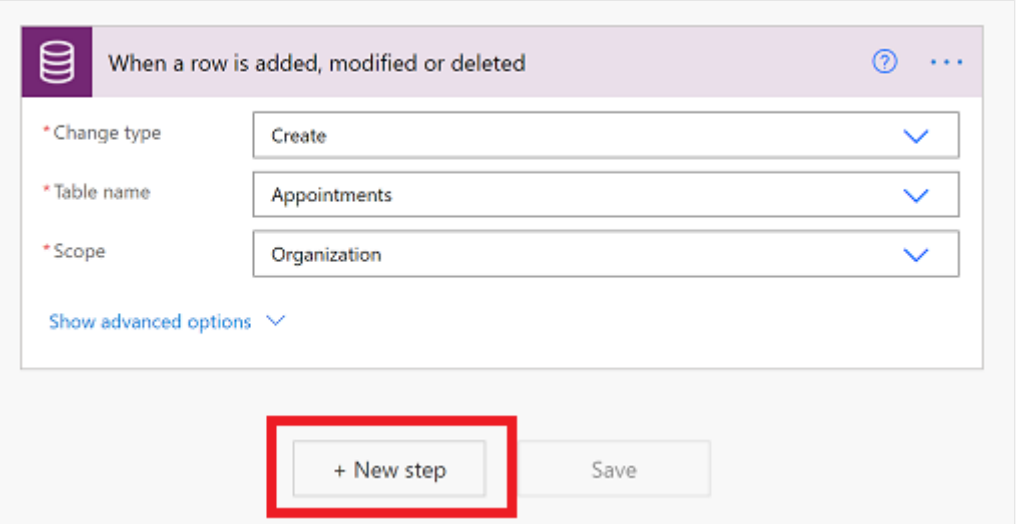
5. Select the action that will trigger the notification.



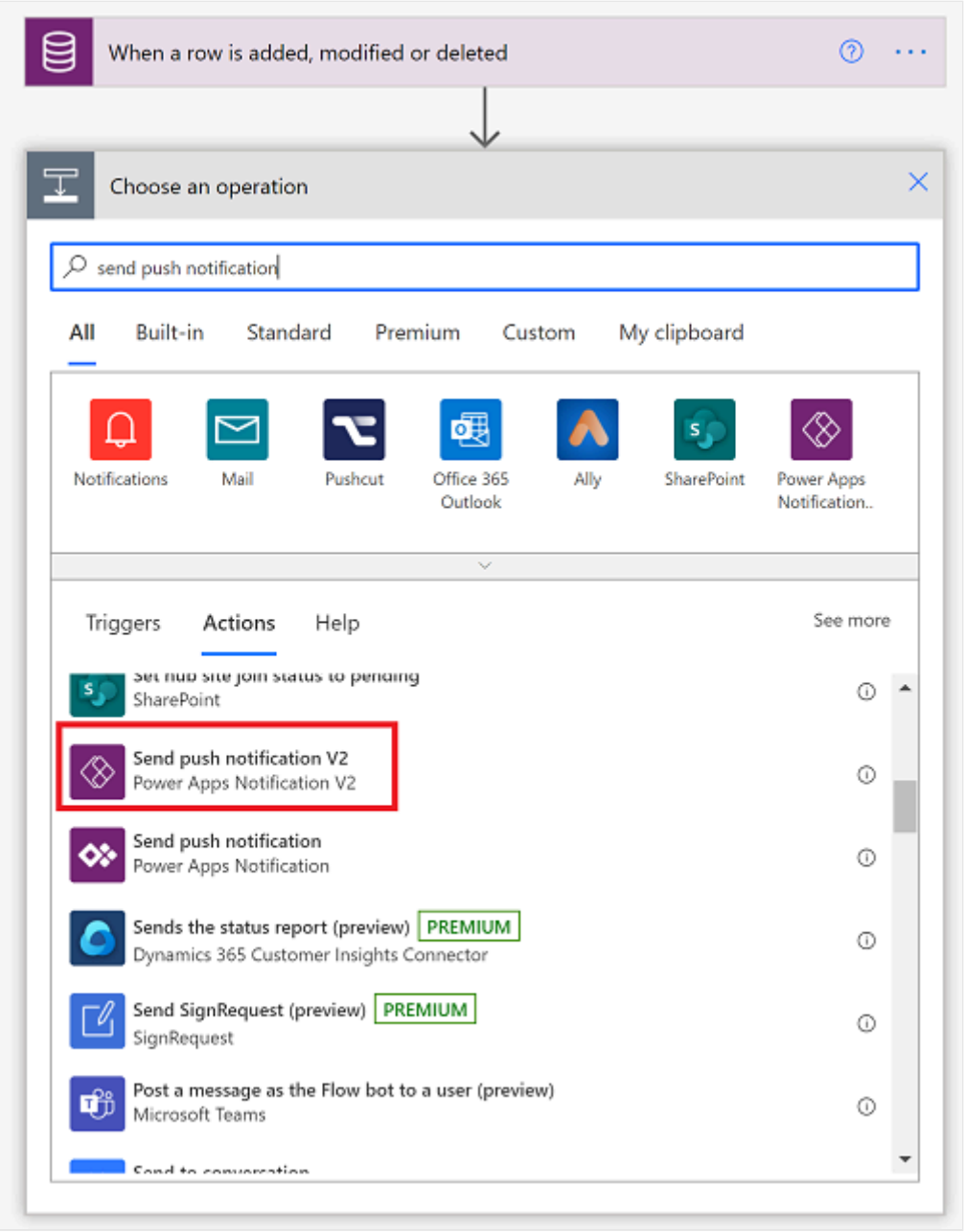
6. Enter the trigger condition information and then select **New step**.

Expand table

Name	Description
Trigger condition	Select the condition for the notification.
The table name	Select which table the notification is for.
Scope	Select the scope.



7. In the **Choose an action** search box, enter **send push notification**. In the list of **Actions** choose **Send push notification V2**.



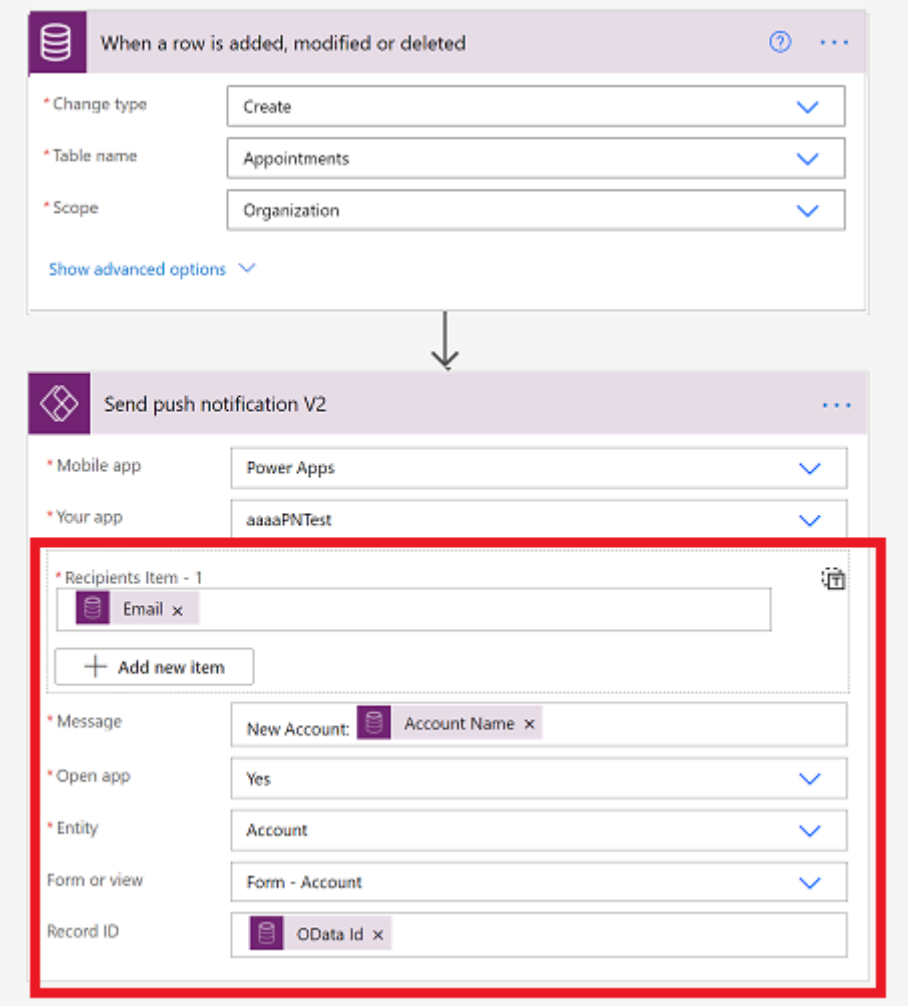
8. On the **Send push notification** screen, enter the following information:

    - **Mobile app**: Select **Power Apps**.
    - **Your app**: Select the app that you want to set up the notification for. Model-driven apps and canvas apps have different parameters. The next step will depend on the type of app you select here.

9. Depending on the type of app that you selected in the previous step, do one of the following:

- For a model-driven app, enter this information:
    - **Recipient Items-1**: Select how the flow is triggered.
    - **Message**: Enter the notification message.
    - **Open app**: Select whether to open the model-driven app or not when the user selects the notification.
    - **Table**: Select which table the notification is for.

- **Form or view**: Select if the notification is for a form or view.
- **Row ID**: If the notification is for a form, then enter the row ID.



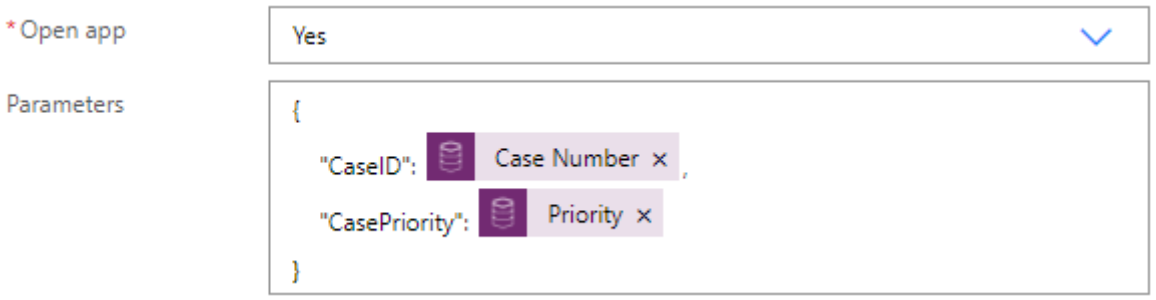
- For a canvas app, enter this information:
  - **Recipient Items-1**: Select how the flow is triggered.
  - **Message**: Enter the notification message.
  - **Open app**: Select whether to open the canvas app or not when the user selects the notification.
  - **Parameters**: Key-value parameters to pass with the notification. Your push notification can pass specific parameters to the app. These can be further processed in the app to open a specific page and load a specific state. For more information, see Perform an action when a user taps the notification.

8. When you're done, select **Save**.
9. Select **Flow checker** to check for error or warnings.
10. Test the flow by selecting **Test** and follow the prompts.

Perform an action when a user taps the notification

Pass parameters

For canvas apps, you can pass key-value pairs using the **Parameters** field as JSON. For example, if you wanted to pass the CaseID, you would put the following as **Parameters**: `{ "CaseID": 42 }`. You can also use dynamic values from earlier steps as shown below.



Read parameters

To read the passed parameters, use Param("ParameterName"). For example, to read the **CaseID** value, use *Param("CaseID")*. To quickly identify this parameter, add a **Label** control to your app. Set the **Text** property of that control to **Param("CaseID")**. If the user opens the app from the **All apps** list, the value is empty. If the user opens the app from another location on the device, the value is populated with the **CaseID** value.

Set the start page

You can set your app to open, for example, the **Case details** page as soon as the app opens:

1. Add a **Timer** control, and set its **OnTimerEnd** property to this formula:

   ```
   Navigate(EditCase, ScreenTransition.None)
   ```

2. (optional) Hide the **Timer** control by setting its **Visible** property to **false**.

3. Set the **OnVisible** property of the screen to **Timer.Start()**.

> **Tip**
>
> It's a good idea to create a unique first page in the app for the notification:
>
> 1. Create an empty page that your app doesn't already open, add a **Text Input** control, and set its **timer.Duration** value.

2. When you create the app, set the timer to a non-zero value. When you're ready to publish the app, set the value to **0** to immediately trigger the timer.

Considerations

- You can send a push notification from one app to another or to the same app. For more information, see Send notification from an app.
- Notifications show the generic Power Apps icon instead of a specific app icon.
- Notifications are not supported in China on Android devices.
- Battery optimization settings on an Android device can interfere with receiving notifications for Power Apps mobile. To receive notifications for Power Apps mobile, turn off battery optimization for the app.
- The example push notifications that are mentioned in this topic are only supported on Power Apps mobile. Other mobile apps such as Dynamics 365 Sales mobile app or Dynamics 365 for phones and tablets do not support these types of notifications.

For reference information, see Power Apps Notification reference.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use deep links with the Power Apps mobile app

10/15/2025

Deep links let users move from one application to another on computers and mobile devices. Simple examples include a mobile app deep-linking to Facebook to sign in, an email address deep-linking to a mail app to compose a message, or a website deep-linking to an app store to download a related mobile app.

Supported deep links in the Power Apps mobile app

You can open an `entityrecord` or an `entitylist` view in the Power Apps mobile app by using deep-link URLs from other apps. When you follow the link from an external app, the target element opens in Power Apps mobile

If you're already signed in to your instance in the app, the target record is displayed when you follow the link from an external app. Otherwise, you're prompted to sign in to your instance in the mobile app; after you're signed in, the target element is displayed. You must have the Power Apps mobile app installed on your mobile device to use this feature.

Supported Urls parameters for a model-driven app

Use the following application handler and query string parameters to compose the URL.

Deep links for the Power Apps mobile app should start with the following:

```
ms-apps://<org-url>_<app-id>?tenantId=<tenant-id>&environmentId=<environment-id>&appLogicalName=<appLogicalName>&appType=AppModule&openApp=true&restartApp=true&forceOfflineDataSync=true
```

> **ⓘ Important**
>
> The org-url can't contain **https://**. The following is an example of a model-driven app deep link:
> ms-apps://contoso.onmicrosoft.com_e6429eba-2204-40e8-b9dd-fc74791ff2c2?tenantId=aaaabbbb-0000-cccc-1111-dddd2222eeee&environmentId=g67tfyufhkjfg&appLogicalName=cr12_e567

Expand table

Parameter	Description	Required
<targeted-app>	• `ms-dynamicsxrm` -> Dynamics 365 for phones • `ms-apps-fs` -> Field Service • `ms-apps` -> Power Apps • `ms-mobile-apps` -> wrapped native mobile apps	Yes
<org-url>	Connects to the correct org URL.	Yes
<app-id>	Opens the correct app module.	Yes
tenantId=<tenand-id>	Connects to the correct tenant.	Yes
*environmentId=<environment-id>	Uniquely identifies the environment within a tenant.	Yes
*appLogicalName=<app-logical-name>	Unique Name of the app. For more information on how to find this unique name of the app in modern studio, refer Manage model-driven app settings in the app designer and for classic editor, refer Manage model-driven app properties in the app designer (classic)	Yes
appType=AppModule	Indicates that the targeted app is a model-driven app.	Yes
restartApp=true	Restarts the model-driven app. Required to ensure parameters are passed when the app is already open.	No
autoLoginUpn=<e-mail>	Autopopulates e-mail and triggers sign-in. **Note**: This parameter will be ignored if a user is already signed into the app.	No
forceOfflineDataSync=true	Ensures that data sync is triggered so that all the latest data is available.	No

***Considerations**

- The environment ID and appLogicalName are mandatory for all new deep links.
- Existing deep links without an environment ID or appLogicalName will experience slower performance until December 2025, after which they won't be supported.
- Update all existing deep links to include the environment ID and appLogicalName to improve performance and reduce wait time.

If opening an `entityrecord` form or creating a new `entityrecord`, use the following parameters:

Expand table

Parameter	Description	Required
etn=<entity-logical-name>	Designates which table to go to.	Yes
pagetype=entityrecord	Indicates that the target is a form.	Yes
id=<record-id>	Designates which specific record to go to; if left blank, the created form for the table opens.	Yes
extraqs=<form-id>	Designates which form to open for the `entityrecord`; if not specified, the default form opens. The `extraqs` parameter can also be used to default field values.	No

If the link goes to an `entitylist` view, add the following parameters:

Expand table

Parameter	Description	Required
etn=<entity-logical-name>	Designates which table to go to.	Yes
pagetype=entitylist	Indicates that we're going to a view.	Yes
viewid=<view-id>	Designates which view to open.	No
Viewtype= <1039 if system view, 4230 if personal view>	Indicates whether the targeted view is a system view or a personal view.	No

Supported Urls parameters for a canvas app

`ms-apps:///providers/Microsoft.PowerApps/apps/<appID>?tenantId=<tenantId>&environmentId=<environment-id>&restartApp=true`

Expand table

Parameter	Description	Required
<app-id>	Opens the correct app module.	Yes
tenantId=<tenantId>	Connects to the correct tenant.	Yes
*environmentId=<environment-id>	Uniquely identifies the environment within a tenant.	Yes

Parameter	Description	Required
restartApp=true	Restarts the Canvas app. Required to ensure parameters are passed when the app is already open.	No
autoLoginUpn=<e-mail>	Autopopulates e-mail and triggers sign-in. **Note**: This parameter will be ignored if a user is already signed into the app.	No

*Considerations

- The environment ID is mandatory for all new deep links.
- Existing deep links without an environment ID will experience slower performance until December 2025, after which they won't be supported.
- Update all existing deep links to include the environment ID to improve performance and reduce wait time.

Supported Urls parameters for a wrapped native mobile app

`ms-mobile-apps:///providers/Microsoft.PowerApps/apps/<appID>?tenantId=<tenantId>&restartApp=true`

> **Important**
>
> The deep link URL for a wrapped app works only when one wrapped app is installed on the mobile device. It doesn't work if more than one wrapped app is installed.

Expand table

Parameter	Description	Required
<app-id>	Opens the correct app module.	Yes
tenantId=<tenantId>	Connects to the correct tenant.	Yes
restartApp=true	Restarts the model-driven app. Required to ensure parameters are passed when the app is already open.	No
autoLoginUpn=<e-mail>	Autopopulates e-mail and triggers sign-in.	No

Parameter	Description	Required
	Note: This parameter will be ignored if a user is already signed into the app.	

Troubleshooting

Deep links can open in your browser depending on the company's organizational policies and the user's device settings. Mobile Device Management (MDM) tools and device operating systems have different options and settings that affect how deep links are handled. If deep links are opening in a browser instead of directly in Power Apps mobile, make sure your MDM policies and device settings are appropriately configured.

As an example, on some Android devices, go to **Settings** > **Apps** > **Power Apps** > **Open by default** and add **apps.powerapps.com** to make deep links open directly in Power Apps mobile.

See also

Important upcoming changes (deprecations) in canvas apps

Open an app in the Power Apps mobile app by scanning a QR code

Article • 01/28/2023

Users can easily open a Power Apps canvas or model-driven app on their mobile device by scanning a QR code, shown in Power Apps, from their desktop computer. To do this, complete the following steps on your desktop computer.

1. Sign in to Power Apps.
2. In the left pane, select **Apps**.
3. Select the check mark icon next to the app that you want to open on your mobile device.
4. Select the **More Commands** (...) icon for the app you want to open, and then select **Details**.
5. A page showing the details of your app is shown. With your mobile device, scan the QR code.



6. The app will open on your mobile device.

Feedback

Was this page helpful? Yes No

Provide product feedback

Switch to a different directory in the Power Apps mobile app

Article • 03/06/2023

When a user signs into the Power Apps mobile app, they sign into their *home tenant*, which is the Azure Active Directory (Azure AD) tenant where their credentials are provisioned. After signing in, a user can easily change the tenant or *directory* with which they intend to use the Power Apps mobile app. This enables users to easily access the Power Apps that have been shared with them in different directories.

This means that:

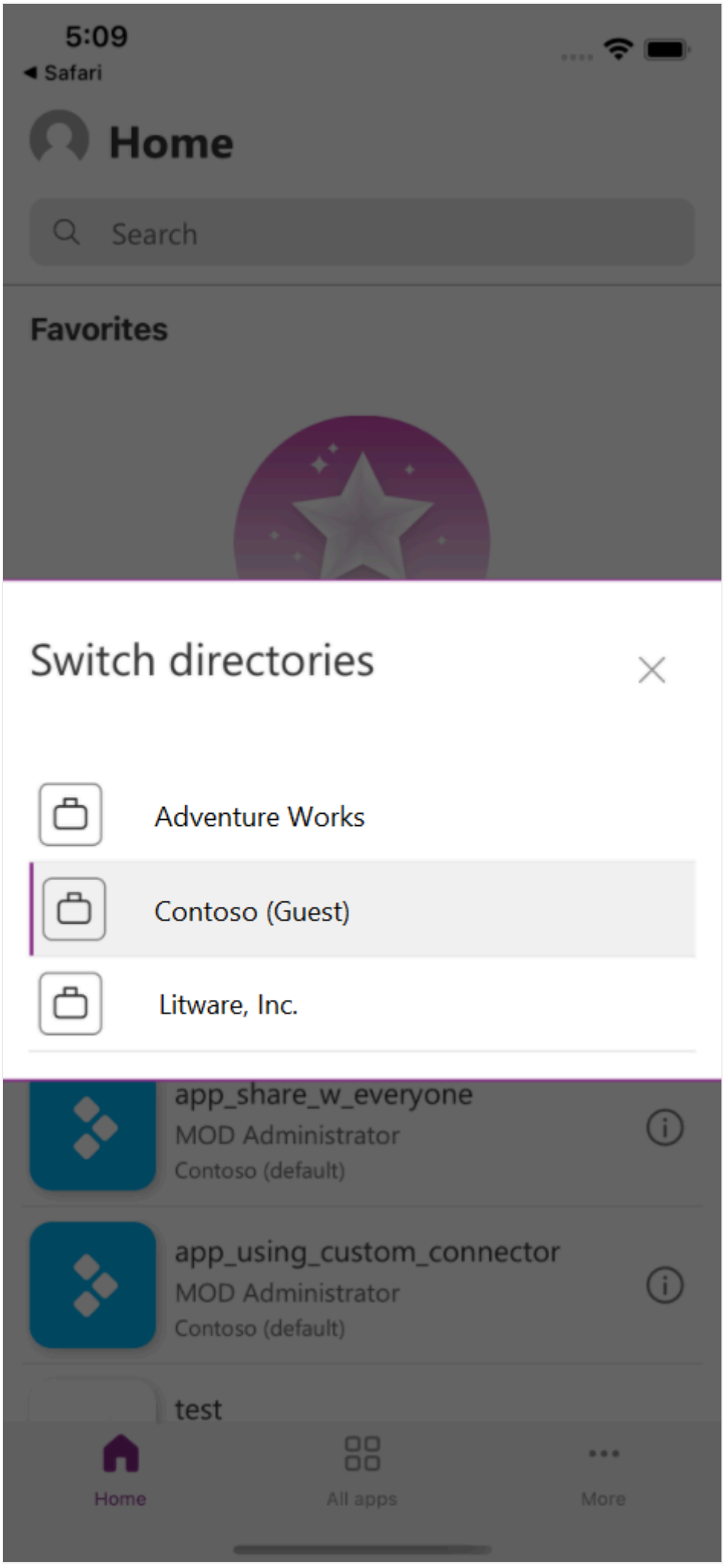
- Users no longer need to log out of the Power Apps mobile app to switch directories.
- Users can specify the directory they want to work in from a list of directories on the user profile page.
- Users, including guest users, can access apps shared with them in different directories.

To switch to a different directory in the Power Apps mobile app:

1. Open the Power Apps mobile app.
2. From the **Home**, **All apps**, or **More** page, select your profile image. Your user profile page appears.

   > ⓘ **Note**
   >
   > Users signing into the Power Apps mobile app with a Microsoft account cannot switch directories from their user profile page. They must use a deeplink, instead.
   >
   > For more information about how to share canvas apps with guest users, see Share a canvas app with guest users.

3. Select the directory you're signed in to. A list of directories appears.


5:09
Safari
Home
Search
Favorites
Switch directories
Adventure Works
Contoso (Guest)
Litware, Inc.
app_share_w_everyone
MOD Administrator
Contoso (default)
app_using_custom_connector
MOD Administrator
Contoso (default)
test
Home
All apps
More

4. Select the directory you want to switch to.

Feedback

Was this page helpful? Yes No

Provide product feedback

Optimize for devices overview (preview)

Article • 05/02/2024

[This article is prerelease documentation and is subject to change.]

Optimize for devices is a feature that changes the way canvas app screens are loaded and displayed on Android and iOS devices. This feature lets makers build stunning, native apps for mobile users with device-optimized screens that use native UI elements, instead of rendering inside of web view. Device-optimizes screens provide the following benefits:

- Modern mobile user experiences and interaction patterns
- Improved performance and reliability
- Integrated device capabilities with maker-defined UI

There's no need to rebuild apps as makers can use all the familiar app-building elements to create device-optimized experiences, though there are some limitations. Apps that are optimized for devices work with other mobile features such as offline, push notifications, and wrap.

> ⓘ **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Allow screens to be device-optimized

1. Select **Settings** > **Upcoming features** > **Preview** > **Optimize for devices**.
2. Only use supported controls on the screen.

After this feature is turned on, you can find which controls are supported in the **Insert** pane by selecting the **Device optimized (preview)** filter. Learn more about which controls are supported natively today.

Screens made solely with device-optimized controls automatically render natively on Android and iOS devices. An app can have screens which are optimized, while some others aren't. This only depends on the controls chosen by the maker to create that screen.

How screens are optimized for devices

For a canvas app running on Power Apps mobile, there are UI elements, their current state, and the related business logic that determines how screens are optimized. For example, on a screen you might have a button at position 0, with a width and height of 200 and 40 respectively, and with fill color blue. The business logic, with Power Fx, lets you navigate the user to a different screen.

If the **Optimize for devices** option is off for that screen, the screen loads and displays in a web view, running similarly to how it would in a desktop's web browser. When **Optimize for devices** is on, each element's current state is passed over to the device where it renders that same element, but using native UI. So now we have a natively rendered, long blue button in the upper-left corner. Business logic modifies state or takes actions, which runs the same way it did before. When the natively rendered button is selected, the device runs Power Fx and the user is navigated to the specified screen.

Each control needs a device-optimized, native equivalent that supports the same properties so it can render the way a maker expects. During preview, there are some limitations to which controls and properties are supported. Whether a control is classic or modern, it filters the list of controls for device-optimized ones.

Feedback

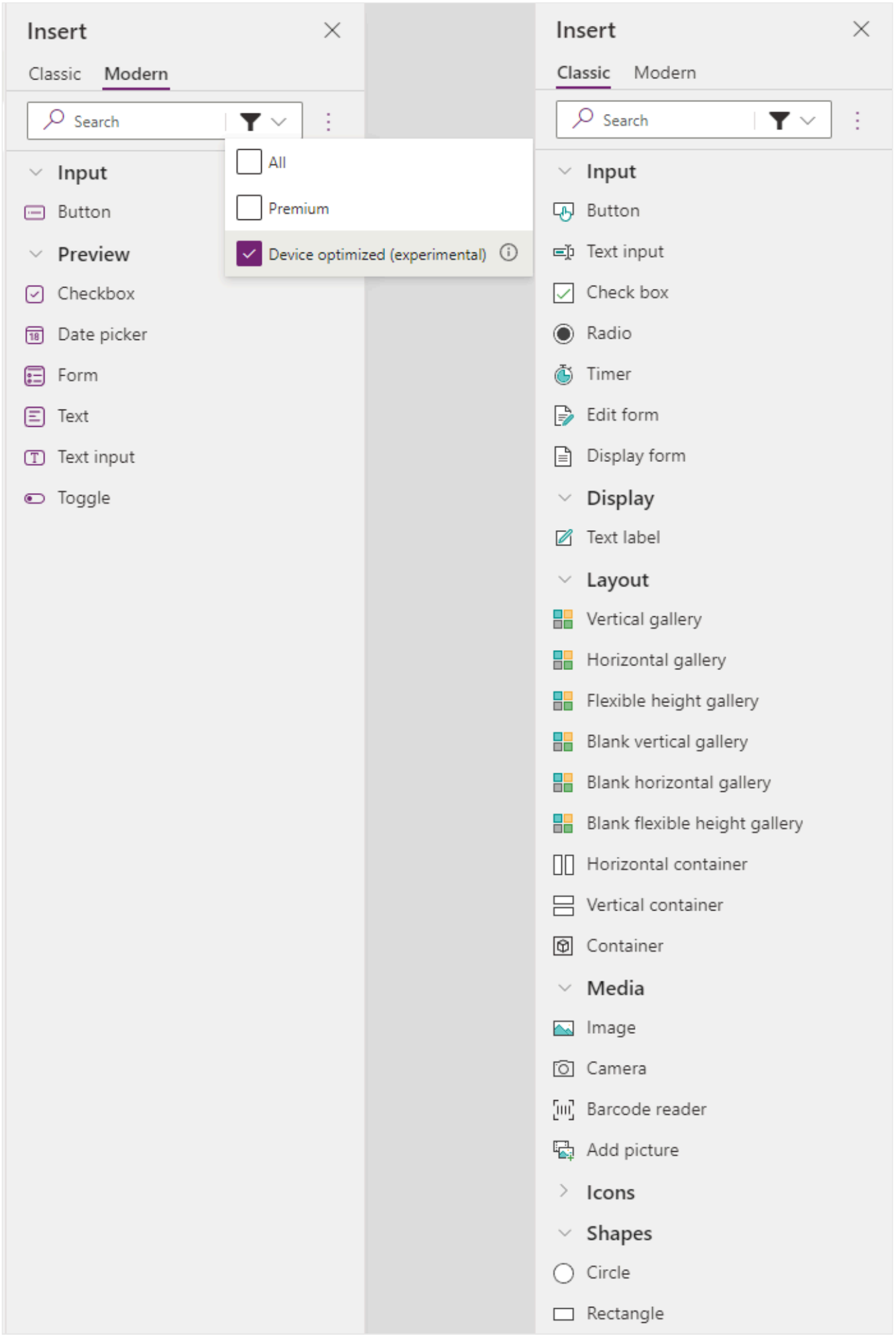
Was this page helpful? Yes No

Provide product feedback

Optimize for devices (preview)

07/28/2025

For a screen to be optimized for devices, it must use controls that are supported natively. You can filter the control Insert pane to only show device-optimized controls.


Insert
Classic
Modern
Search
All
Premium
Device optimized (experimental)
Input
Button
Preview
Checkbox
Date picker
Form
Text
Text input
Toggle
Insert
Classic
Modern
Search
Input
Button
Text input
Check box
Radio
Timer
Edit form
Display form
Display
Text label
Layout
Vertical gallery
Horizontal gallery
Flexible height gallery
Blank vertical gallery
Blank horizontal gallery
Blank flexible height gallery
Horizontal container
Vertical container
Container
Media
Image
Camera
Barcode reader
Add picture
Icons
Shapes
Circle
Rectangle

> **ⓘ Important**
>
> - This is a preview feature. Documentation is subject to change.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Device-optimized controls

The following controls are optimized for iOS and Android devices. All other controls are currently unsupported. A screen that has one or more unsupported controls automatically renders in a web view instead of native UI. Learn more: Optimize for devices overview.

Controls

- Add picture
- Attachment
- Barcode reader
- Button
- Camera
- Check box
- Circle
- Components
- Container (horizontal and vertical included)
- Form
- Gallery (vertical, horizontal, flexible height)
- Icon (all icons supported)
- Image
- Radio
- Rectangle
- Text input
- Text label
- Timer

Modern controls

- Button
- Check box
- Date picker

- Text
- Text input
- Toggle

Learn more: Overview of modern controls and themes in canvas apps

Unsupported controls

If a screen has an unsupported control, the screen automatically reverts to render with a web view.

Device-optimized properties

Some properties must be configured so they're optimized for devices. For example, consider the barcode reader control. The maker can select **Scan Inline** for the **Scanning mode** property. In Power Apps Studio, when this mode is selected, the control displays as a warning box indicating that the control only works on mobile devices. On a device, in a screen optimized for devices, the control displays a camera feed to allow barcode scanning.

Known limitations

Note the following limitations with the 'optimize for devices' feature.

- The **Image** control, when showing an SVG file, only supports the fit, fill, and stretch image position options. Its default behavior is to fit, if another option is selected. All other image types work as expected for all image positions.
- In the **modern Text input** control, the **Value** property, when set to a Power Fx variable, isn't displayed at first navigation. Instead, it shows the placeholder value until the user navigates away and back to the screen with the **modern Text input** control.
- In the **modern Text input** control, the **OnChange** property is called each time the user interacts with the control, instead of when navigating away from the control.
- In the **modern Text input** control, the **Validate state** property doesn't display a red border around the control.
- In the **Image** control, the **Image rotation** property isn't supported. The image isn't rotated.
- In the **Dropdown** and **Radio** control, the **Items** property doesn't support references to other controls. Items aren't shown if they're complex values, like references.
- In the **modern Date picker** control, tapping on the control opens a modal with a calendar instead of allowing users to type into the control directly.

- In the **modern Date picker** control, the **Base palette color** and modern theme aren't applied on Android devices. Instead, it displays as the device's theme color.
- In the **Button**, **Image**, **Label**, and **Icon** controls, the **SetFocus** Power Fx function doesn't support keyboard focus, only accessibility focus (for example, screen reader).
- In **Horizontal and vertical containers**, the **Scroll** option only works for a single direction, depending on the **Wrap** property. If **Wrap** is off, the container is scrollable in the primary direction, otherwise it's scrollable in the secondary direction.
- Multiple **Barcode reader** controls on a single screen aren't supported.
- Multiple **Camera** controls on a single screen aren't supported.
- In the **Camera** control, the **Display mode** property of the parent control isn't supported. For example, if the camera control is part of a form that is set to view mode, the camera might still be in edit mode.

Related information

Supported platforms for running apps using the Power Apps mobile app

How offline first functionality works in Power Apps mobile

Offline first functionality in Power Apps mobile enables users to access and work with app data on their mobile devices without an active internet connection. This feature uses a local database on the device to store data. When your device is connected to a network, any writes made in the app sync to Dataverse immediately. If you perform updates while the network is offline, those changes sync automatically with Dataverse as soon as connectivity returns. Reads performed while the device is connected to the internet follow the sync frequency configured by the maker in the app's offline settings. When the device transitions from no network to an active connection, read operations are triggered again to refresh data.

Offline profile

First, to create an offline-first app, maker must create the offline profile. The profile defines:

- Tables and columns available for offline use
- Optional filters to limit the data downloaded per table
- Relationships between tables for linked data access

After you create the profile, when you first open the app, the data for the offline profile downloads and stores locally on the device in a SQLite-based cache. This local copy becomes the primary data source for all app operations. Learn more about setting up offline for canvas apps in Set up mobile offline for canvas apps and about setting up offline for model-driven apps in Set up mobile offline for model-driven apps.

Data synchronization

The sync process keeps offline data up to date with Dataverse through automatic synchronization:

- **Initial download**: When the app first opens, all data defined in the offline profile is downloaded to the local device.
- **Incremental updates**: Subsequent down-syncs fetch only changes (inserts, updates, deletes) since the last sync, reducing data transfer.
- **Configurable frequency**: Makers can set down-sync intervals (every few minutes, hourly, or daily) for each table in the offline profile in the maker studio.
- **Smart sync**: If no changes are detected across tables, the down-sync is skipped to save bandwidth.

- **Automatic resumption**: When the mobile device reconnects to the network, the down-sync operation resumes automatically.
- **Platform differences**: On iOS, syncs happen only in the foreground. On Android, syncs that start in the foreground continue even when the app moves to the background.

Local storage and data access

All offline data is stored in a local database on the device, ensuring data availability regardless of network status.

Read operations: In an offline first app, the app reads from the local cache for all data operations, regardless of network connectivity. This includes viewing records, searching, and filtering data.

Write operations: Changes made when the network is offline are queued locally and pushed to Dataverse when connectivity is restored. Users can continue making modifications without interruption. Changes made when network is online are saved locally and instantly synced to Dataverse.

Data retention: Retention duration isn't fixed by the platform. Data persists as long as the app and offline profile remain installed on the device. There's no automatic expiry unless:

- The user clears the app cache or uninstalls the app
- The maker updates the offline profile, which can trigger a full refresh
- The user sign out of the app before the sync is complete

Maker controls and optimization

Makers have several tools to optimize data download, offline performance and control the offline experience:

- **Column selection**: Choose specific columns to minimize payload size and reduce download time
- **Relationship configuration**: Define parent-child table relationships for linked data access
- **Sync frequency**: Configure sync intervals and filters for each table individually
- **Media settings**: Configure which images and files should be downloaded for offline access

These controls allow makers to balance offline functionality with performance and storage constraints on mobile devices. Learn more about how to optimize offline profile in Optimize the offline profile

Building offline app screens

Screens in an offline-enabled app are built on top of the offline profile and should be designed with offline scenarios in mind:

Data binding: Controls such as galleries and forms bind to tables defined in the offline profile, ensuring data availability offline.

Conditional logic: Screens should handle offline scenarios by disabling actions that require live server calls, such as certain API integrations or real-time data operations.

Optimized layouts: Galleries and forms should show essential columns only, reducing load time and improving performance on mobile devices.

Write-back handling: Input controls queue changes locally and should display status indicators (for example, "Pending Sync") until connectivity returns and synchronization completes.

Offline-first design benefits

The offline-first architecture provides several advantages over traditional online-only applications:

- **Faster load times**: No round trips to the server for every screen or data query, resulting in near-instantaneous data access
- **Consistent experience**: Users interact with the same local dataset whether online or offline, eliminating connectivity-dependent behavior changes
- **Reduced network dependency**: Only sync operations hit the server, not every read operation, minimizing bandwidth usage and improving reliability

This design ensures that users can work productively in any network environment, from stable Wi-Fi connections to areas with intermittent or no connectivity.

Last updated on 01/07/2026

FetchXML editor for offline profiles (preview)

[This article is prerelease documentation and is subject to change.]

FetchXML is Dataverse's native query language that lets makers write data filters as XML code, similar to writing a formula, but more powerful. Use the FetchXML editor to optimize complex profiles for better performance and avoid sync timeouts on large tables (100K+ records).

This feature is best suited for advanced makers and developers who are comfortable working with XML and need more control than what the visual filter builder provides.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.
> - This feature is being gradually rolled out across regions and might not be available yet in your region.

Benefits of the FetchXML editor

The FetchXML editor offers several advantages over the visual expression builder for defining offline data filters.

- **Support for hierarchical conditions**:

  The FetchXML editor supports building complex, hierarchical conditions such as `under`, `eq-or-under`, `above`, and `eq-or-above` on hierarchy-enabled lookups. Learn more about querying hierarchical data

  XML

```
<fetch>
  <entity
    name="account">
```

```
    <filter
        type="and">
        <condition
        attribute="statecode"
        operator="eq"
        value="0" />
    </filter>
    <link-entity
        name="businessunit"
        from="businessunitid"
        to="owningbusinessunit"
        link-type="any">
        <filter
        type="and">
        <condition
           attribute="businessunitid"
           operator="eq-or-under"
           value="{YOUR-PARENT-BU-GUID}" />
        </filter>
  </link-entity>
</entity>
</fetch>
```

- **Support for checks against unrelated tables**:

  The FetchXML editor lets you link to other tables even if they don't have a direct relationship with the primary table, provided you specify valid **from** and **to** attributes. This capability is especially useful in complex offline scenarios where multiple entities share a common foreign key but aren't directly related.

  XML

```
<fetch
    distinct="false"
    latematerialize="true"
    options="DisableRowGoal, EnableOptimizerHotfixes">
<entity
    name="cr57f_producttranslation">
    <filter
       type="and">
       <link-entity
       name="cr57f_userlanguagepreference"
       from="cr57f_language_id"
       to="cr57f_language_id"
       link-type="any">
       <filter
          type="and">
          <condition
             attribute="cr57f_user_id"
             operator="eq"
             value="user_002" />
```

```
      </filter>
      </link-entity>
   </filter>
  </entity>
</fetch>
```

- **Query optimization support by using late materialization and query hints**:

  The editor exposes advanced optimization controls directly on the `<fetch>` element, including:

- `latematerialize="true"`: By using `latematerialize="true"`, you optimize query performance by narrowing down matching records before retrieving all their column data. This attribute reduces the load when syncing large tables during offline sync. Learn more about using late materialize

XML

```
<fetch
  distinct="false"
  latematerialize="true"
  options="OptimizeForUnknown,ENABLE_HIST_AMENDMENT_FOR_ASC_KEYS">
<entity
  name="msdyn_workorder">
  <filter
     type="and">
     <condition
     attribute="statecode"
     operator="eq"
     value="0" />
     <condition
     attribute="msdyn_systemstatus"
     operator="in">
     <value>690970000</value>
     <value>690970001</value>
     </condition>
  </filter>
  <!-- Booking chain -->
  <link-entity
     name="bookableresourcebooking"
     from="msdyn_workorder"
     to="msdyn_workorderid"
     link-type="any">
     <filter
     type="and">
     <condition
        attribute="statecode"
        operator="eq"
        value="0" />
     <filter
        type="or">
```

```
        <condition
           attribute="starttime"
           operator="today" />
        <condition
           attribute="starttime"
           operator="next-seven-days" />
     </filter>
     </filter>
     <link-entity
        name="bookingstatus"
        from="bookingstatusid"
        to="bookingstatus"
        link-type="any">
        <filter
           type="and">
           <condition
              attribute="statecode"
              operator="eq"
              value="0" />
        </filter>
     </link-entity>
     <link-entity
        name="bookableresource"
        from="bookableresourceid"
        to="resource"
        link-type="any">
        <filter
           type="and">
           <condition
              attribute="userid"
              operator="eq-userid" />
        </filter>
     </link-entity>
  </link-entity>
  <!-- Customer asset -->
  <link-entity
        name="msdyn_customerasset"
        from="msdyn_customerassetid"
        to="msdyn_customerasset"
        link-type="any">
     <filter
     type="and">
     <condition
        attribute="statecode"
        operator="eq"
        value="0" />
     </filter>
     <!-- Account -->
     <link-entity
        name="account"
        from="accountid"
        to="msdyn_account"
        link-type="any">
     <filter
        type="and">
```

```
        <condition
           attribute="statecode"
           operator="eq"
           value="0" />
    </filter>
   </link-entity>
  </link-entity>
</entity>
</fetch>
```

- The `options` attribute passes SQL Server query hints. Learn more about using these options

How the FetchXML editor works

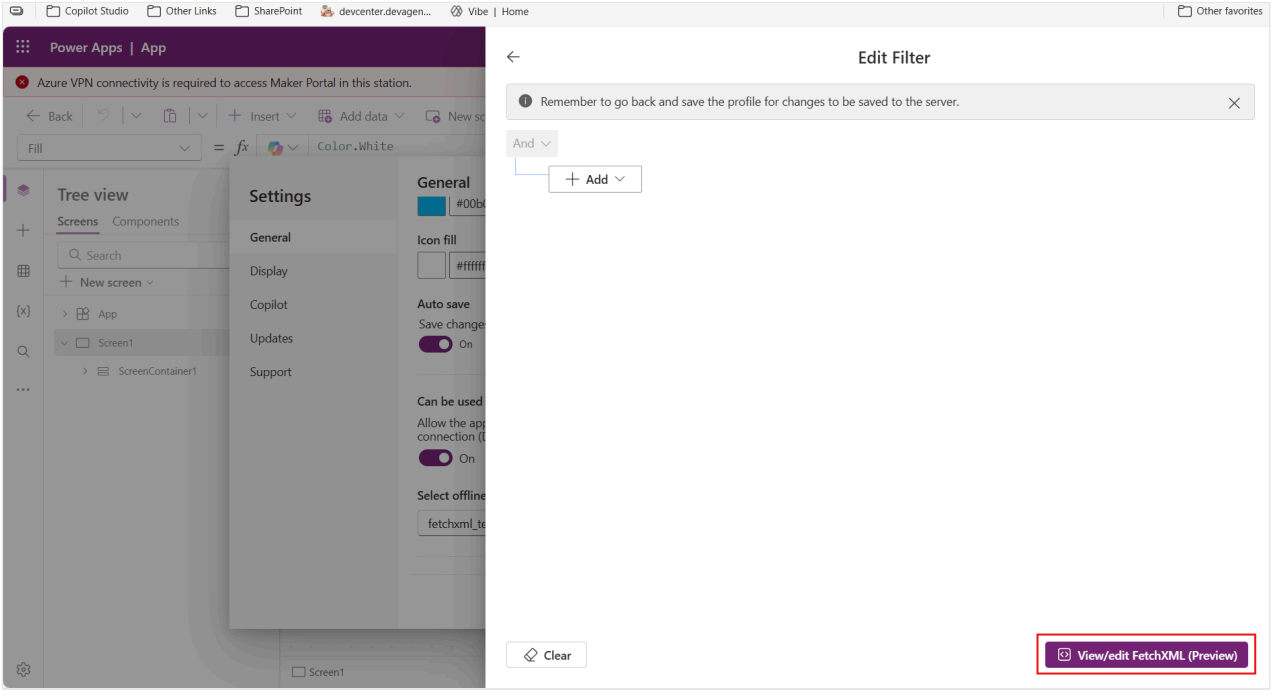
You access the FetchXML editor through the offline profile configuration experience in Power Apps. You write or paste a FetchXML query, and the editor validates it in real time before you save.

1. In Power Apps, go to your environment's settings and open the **Offline Profiles** page. Select an existing offline profile or create a new one, and then select **Edit profile**.

2. In the offline profile editor, add or select the table whose data you want to filter for offline use.

3. Under **Filter**, choose **Custom** or **Related rows**, and then select **Edit filter**.

4. In the filter editor, scroll down and select **View/Edit FetchXML** to open the FetchXML code editor.



5. Modify the existing FetchXML query or paste a new one. The editor automatically validates syntax and structure as you edit, checking for:

   - A required <fetch> root element with at least one <entity> element
   - Only supported FetchXML tags
   - Elements placed inside the correct parent elements
   - A maximum of 500 filter clauses

   If the editor detects an issue, such as a missing required element or an unsupported <link-entity> usage, it shows a clear error message so you can fix it before saving.

6. After your FetchXML query is valid, select **Apply** to convert the filter, and then **Save** the offline profile.

> ⓘ **Important**
>
> Always select **Apply** and then **Save** after editing FetchXML. If you don't save, your changes are discarded.

After saving, the offline profile uses your custom FetchXML filter for data sync. If your query uses only features supported by the visual builder, those filters continue to display in the Expression Builder UI. If you use advanced FetchXML-only features, the visual filter UI might not render them. Use the FetchXML Editor for any future edits to that filter.

Best practices

- Apply performance hints like `latematerialize="true"` or `hint="union"` for large datasets.
- Validate your FetchXML by using the built-in editor checks before saving.
- Save your offline profile immediately after applying FetchXML changes.
- Use only supported FetchXML constructs for offline profiles, such as `link-type="any"` or `link-type="not any"`.
- Avoid unsupported join types like `inner` or `outer` in offline filters.
- Keep filter clauses under the 500-clause limit per query.
- Don't ignore validation errors. Queries must be well-formed and complete.

Limitations

- Queries must start with a <fetch> root element and include at least one <entity> element.

- Only supported FetchXML elements are allowed. Unknown or unsupported tags trigger validation errors.
- Elements must follow proper parent-child nesting rules.
- <link-entity> inside <filter> must use `link-type="any"` or `link-type="not any"`. Inner and outer joins aren't supported.
- FetchXML <attribute> tags aren't used for column selection in offline sync. Use the **Filter Columns** option instead.
- Explicit pagination attributes like `page` and `count` are ignored in offline profiles.
- Queries are limited to a maximum of 500 filter clauses.

Last updated on 04/08/2026

Mobile offline for canvas apps overview

Article • 06/05/2024

If users of your Dataverse-based canvas app have spotty or no Internet access, you can easily provide offline-first access with simple switches and a canvas control. Basic offline-first apps are easy to make. Just build your app with normal Power Fx formulas, and Power Apps offline features handle everything else.

You can also build complex offline apps using custom *offline profiles*. An offline profile is the configuration that determines the data synchronized to users' devices. Offline profiles help you to optimize your app's performance by reducing the amount of data downloaded to the device. For example, while you may have access rights to millions of records in a table, your app may need only a thousand of them. If you download only a thousand records, your app performs better. If you need more than 15 tables in your offline app, you may want to consider using a custom offline profile.

To get started with mobile offline, turn on the offline feature in your app and in the tables your app uses. A basic offline screen template is automatically inserted into the app for you to use directly or as a starting point. Set up mobile offline for canvas apps.

Mobile offline-first

Offline-first means that all the data users need when they're offline is copied to their mobile device. It requires network access to download the data initially, but after that, users work exclusively with the data stored locally on their device, even when they're online. Moving on and off the network doesn't affect the performance of the app because it's using local data. Power Apps monitors your app's network access. When it detects a connection, it automatically syncs any local changes to the server and downloads any updates from the server. The offline features handle conflict detection and minimize the use of system resources.

When you turn on the offline capability in your app, the app always runs offline-first, with or without an Internet connection. This functionality optimizes offline performance and creates a consistent experience for users as they change locations.

Why use offline for canvas apps instead of LoadData/SaveData?

There are some key benefits to using the built-in offline functionality for canvas apps instead of LoadData/SaveData Power Fx functions.

Expand table

Topic	LoadData/SaveData	Built-in offline
Power Apps Studio support	Custom	Built-in
Code complexity	Power Fx code complexity scales up with data complexity	No code
Supported Power Fx functions	All	Partial (See Mobile offline limitations for canvas apps.)
App checker rules	None	Common configuration problems are flagged
Offline or connectivity user experience	Not provided/custom only	Automatic
Sync user experience or status	Not provided/custom only	Automatic
Image and file support	Yes, but may run into performance and memory limits	Yes
Optimized delta sync	No	Yes
Conflict resolution	Manual	Automatic
Local data store	Files	Transactional database
Automatically handles schema changes	No	Yes
Fast app launch with large datasets	No	Yes
Data size limit	Device dependent, 30-70 MB in most cases	Device dependent, 3 million rows

See also

Work with canvas apps offline

Feedback

Was this page helpful? Yes No

Provide product feedback

Set up mobile offline for canvas apps

Turn on offline mode in your canvas app if users need to work without worrying about their Internet connection.

Prerequisites

- You must have the Environment Maker, System Administrator, or System Customizer role to set up offline mode for canvas apps. These roles have create, read, write, delete, and share privileges on **Canvas App**, **Mobile offline profile**, and **Sync Error** tables. Learn about predefined security roles.
- Users with the Basic User role can't open an offline application. You must create a custom security role with read privileges on **Canvas App**, **Mobile offline profile**, and **Sync Error** tables.
- If you already have a custom security role, make sure it grants read privileges on **Canvas App**, **Mobile offline profile**, and **Sync Error** tables. Learn about Create or edit a security role to manage access.

> **Note**
>
> It's recommended that the canvas app is in a solution. The offline profile **App usage** field in the Power Platform admin center (**Environements** > *select an environment* > **Settings** > **User + permissions** > **Mobile configuration**) only applies to apps within a solution.

Optimize your app for mobile offline

Mobile apps run on small screens with limited connectivity. Before you set up offline mode, make sure your canvas app is optimized for offline and mobile use. Consider the user scenarios you want to cover and the amount of data the app uses. Create an app that's simple and lightweight.

Follow these best practices when you build an app for mobile offline use:

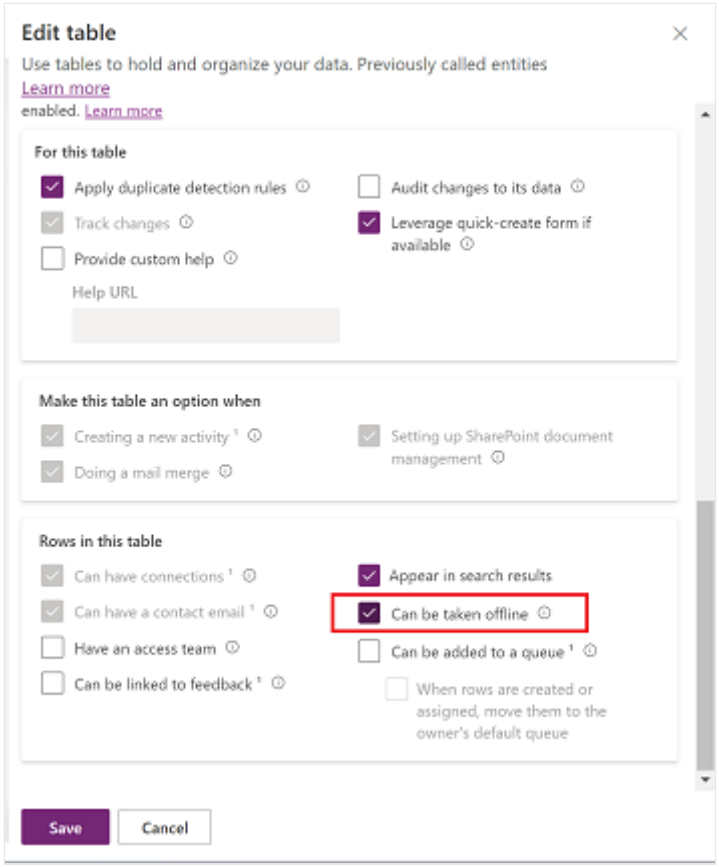
- Performance is critical when you run an app on mobile. For more information about creating performant canvas apps, see Overview of creating performant apps.
- Identify the on-the-go scenarios that are functionally related, such as tasks that are performed by users who work in the field.

- Reduce the complexity of your app by limiting the number of tables that need to be downloaded. Sometimes it's better to have two apps instead of one.
- Use views that are optimized for the experience rather than displaying a large set of data. For example, it's preferable to use the **My active accounts** view, rather than the **All accounts** view.

Turn on offline capability for tables

Tables included in an offline app must be configured for offline use. Some built-in tables have it by default. Some tables can't be used offline. However, you can turn on offline capability for any new, custom table.

1. Sign in to Power Apps.
2. In the left side panel, select **Tables**.
3. Select a table, and then select **Edit**.
4. Select **Edit table properties**.
5. Select **Advanced options**, and then scroll to the **Rows in this table** section.
6. Select **Can be taken offline**.



7. Select **Save**.

Turn on offline capability for your app

1. In Power Apps Studio, in the left side panel, select **Apps**.
2. Select your canvas app, and then select **Edit**.
3. Select **Settings**.
4. Select **General**.
5. Turn on the **Can be used offline** option.

   If you don't see the **Can be used offline** option, make sure the app is in a solution.
6. Select an offline profile. You have two options:

   - **Autogenerated profile**: This setting downloads all the tables used in your app automatically. Autogenerated profiles are a starting point to create an offline profile and help makers validate their offline data. These profiles can be resource-intensive as they don't compute optimal filters for each table or assess the amount of data the app must download offline. App makers need to review and refine these profiles based on their use case.
   - **Custom mobile profile**: Custom offline profiles let makers add only relevant tables, limit data usage and include filters based on their needs and best practices. These profiles offer flexibility and control, enhancing the user experience by downloading relevant data, improving sync times, and boosting query performance.
7. Close the **Settings** window, and then save and publish your app.

A page that's based on the Offline template is automatically inserted in the app. The page contains an icon that uses the Connection Power Fx function to reflect the app's offline sync status. You can use it as-is or customize it, as appropriate.

> **Note**
>
> - You can use the same, offline profile in different canvas apps. The users can use the apps and share the same, offline profile in disconnected mode. The updates done in one app are reflected in the other apps since the apps share the same, local storage with this setup.
> - ALM isn't currently supported for autogenerated profiles.

- If your organization uses ALM and you're moving your app between environments, ensure that your offline profile is tested for all use cases in the target environment.

If the automatically generated offline profile doesn't meet your needs, create your own. Learn about guidelines for offline profiles.

Create an offline profile

You need to publish your new offline profile before you can select it in your canvas app's settings. If you edit and publish an offline profile, you must also publish any canvas apps that use it.

There are two ways you can create offline profile.

Create a custom mobile offline profile from within Power apps studio

1. Sign in to Power Apps Studio.
2. In the left side panel, select **Apps**.
3. Select your app, and then select **Edit**.
4. Select **Settings** > **General**.
5. Turn on the **Can be used offline** setting.
6. In the drop-down menu, select an offline profile. You can select the **Auto-generated** option or **New profile with current app data**.
7. Enter a name and description for the profile. Add the tables you need for the profile. For those tables, apply filters and choose columns as described in optimize the offline profile.
8. Save the profile.
9. Publish the profile.

Create profiles from within Power Platform admin center (with admin rights)

1. Sign in to the Power Platform admin center using an admin account.
2. In the left side panel, select **Environments**.
3. Select an environment, and then select **Settings**.
4. Expand **Users + permissions**, and then select **Mobile configuration**.
5. Select **Create new profile**.

6. Enter a name and description, and then select **Create**.

7. After the profile is created, select it to continue editing it.

> **Important**
>
> Don't add users in the offline profile in the **Users with offline access** area. This capability is only applicable to model-driven apps and is restricted to selected users.

Add a table to an offline profile and apply filters

Applying an appropriate filter for each of the tables configured in the offline profile is critical to limiting the amount of data that downloads on users' devices.

Be sure that you configure at least one of the profile rules for each table to download its data.

Expand table

Customization	Recommendation
Organization rows - if selected, then select at least one of these options: - **User's rows** - **Team rows** - **Business unit rows**	If you want to define this filter, then you have to pick at least one of the given options. It's highly recommended to not have a business unit-level filter for a table, unless there's a strong justification. It's recommended for a master data scenario with a small data set, like country codes.
All rows	If you select this filter, you can't define any other filter rules.
Related rows only	Be sure that the related table has been added to the offline profile.
Custom	You can define a custom filter up to three levels deep.

1. In the **Data available offline** section, select **Add table**.

2. Select a table, and then define the filters.

3. Select **Next**.

4. Set the following filters:

   a. Choose the row that you want to make available offline. For the **Custom** option, use the expression builder to set up advanced conditions.

b. **Relationships** lists the different relationships available between the current table and other tables added in the offline profile. Selecting a relationship ensures that related rows following that relationship are downloaded and made available offline.

c. **Sync interval** defines the sync frequency to be applied on the device that syncs with the data with the server. If a table's data doesn't change frequently, like a catalog or a product table, you might want to focus on only syncing data when necessary, such as refreshing only once a day.

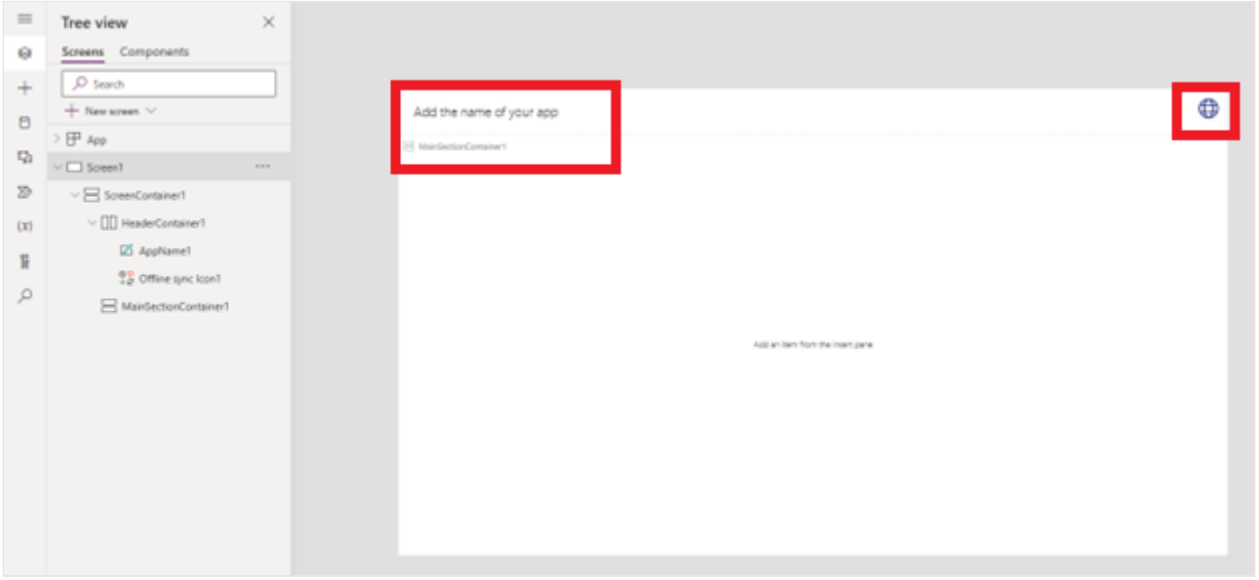
5. Select **Save**.

> **Note**
>
> Each table in your offline setup can link to up to 15 other tables through filters or relationships. If you link to a table that already connects to other tables, those connections count toward the limit. For example, if Table B connects to two other tables and you link Table A to Table B, Table A has three relationships: one for Table B and two for the tables Table B connects to. This setup is a transitive relationship. If any table has more than 15 total connections, you get an error and can't publish the offline profile.

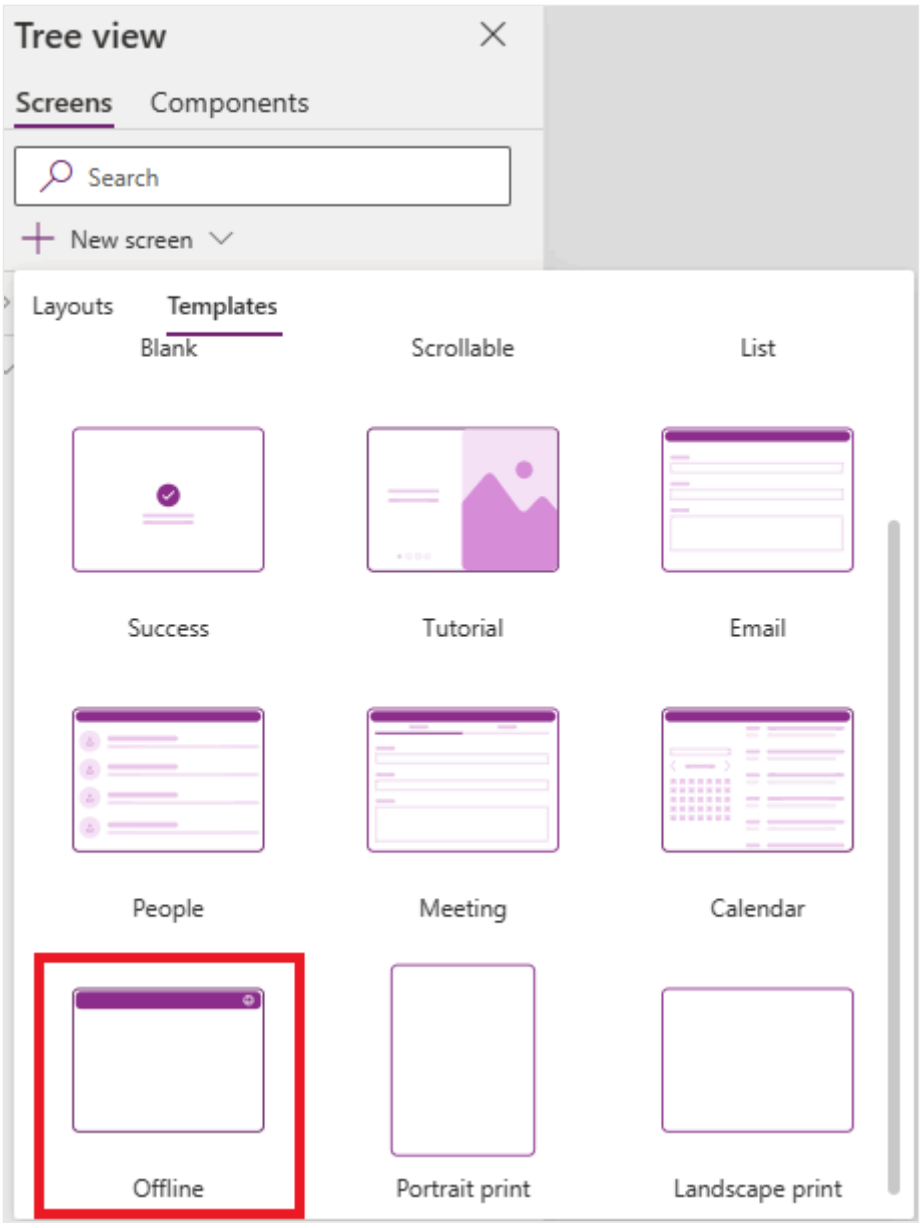
Using the Offline template and offline status icon

To make it easier to create or convert a canvas app for offline use, we created the **Offline** template. The template includes a globe icon in the navigation bar that shows the connectivity and sync state, putting offline at the center of the experience. Users always know whether their device and data are ready to go.

A new page created from the **Offline** template is automatically inserted as soon as you turn on the offline capability in your canvas app.



You can also add a new page based on the **Offline** template the same way you add pages from any other template.



The globe icon in the template's navigation bar uses different icons depending on the value of the Power Fx `Connection.Sync` function. You can create your own icon and set the **Icon** property using the following Power Fx script:

PowerApps Formula

```
Switch(Connection.Sync,

   ConnectionSync.Connected, Icon.Globe,

   ConnectionSync.ConnectedWithWarning, Icon.GlobeWarning,

   ConnectionSync.ConnectedPendingUpsync, Icon.GlobeChangesPending,

   ConnectionSync.ConnectedError, Icon.GlobeError,
```

```
    ConnectionSync.ConnectedRefresh, Icon.GlobeRefresh,
    ConnectionSync.NotConnected, Icon.GlobeNotConnected,
    ConnectionSync.NotConnectedWithWarning, Icon.GlobeWarning,
    ConnectionSync.NotConnectedPendingUpsync, Icon.GlobeChangesPending,
    ConnectionSync.NotConnectedSyncError, Icon.GlobeError)
```

You can set the **OnSelect** property using the Power Fx function `ShowHostInfo` to show the Device status page when selecting the globe icon.

PowerApps Formula

```
ShowHostInfo(HostInfo.OfflineSync)
```

You can set the **Visible** property using the Power Fx function `Host` to only show the globe icon when the app is used on a player that's supporting the offline mode.

PowerApps Formula

```
Host.OfflineEnabled
```

Export a solution that contains a canvas app enabled for offline

When you export a solution with an offline canvas app, you also need to export the offline profile for the app, as it's a dependency of the app.

1. Sign in to Power Apps and in the left navigation pane, select **Solutions**.
2. Select the solution that you want to export and then select **Edit**.
3. Select the canvas app that is enabled for offline and then select **Advanced** > **Add required objects**.

Make sure that the associated offline profile is in the solution and then you can export it.

See also

- Mobile offline limitations for canvas apps
- Best practices for developing an app for offline use
- Optimize the offline profile

Last updated on 01/08/2026

Configure files and images in offline canvas apps

Summarize this article for me

This article explains how to configure files and images in a Power Apps table for offline canvas apps.

Prerequisites

Have an existing canvas app with Dataverse tables. For more information, see Build an app.

View column properties for a file or image

1. Sign in to Power Apps.
2. In the left navigation menu, select **Tables**. If the item isn't in the side panel pane, select ... More and then select the item you want.
3. Select the table that has the columns you want to view.
4. Under **Schema**, select **Columns**.
5. Select the **Display name** of a column where **Data type** is set to **File** or **Image**.
6. The column properties show the **Data type**. Expand **Advanced options** to view the maximum size for a file or image. For more information about columns, see Columns overview.

Download options

In an offline-enabled canvas app, there are two ways to download files and images:

- **On view**: Downloaded when you view them—the default option.
- **On sync**: Downloaded during offline sync when device is connected.

Download files and images on view

When files and images are configured to download **on view**, the content is downloaded on the device when the user opens a file or when an image is presented to the user. Files and images are downloaded when the device is connected to the network and can be used afterward without connectivity.

This mode is enabled by default without any specific configuration.

If you want to keep this default, then you don't need to proceed with this article.

Download files and images on sync

When files and images are configured to download **on sync**, they download during the offline sync when the device is connected to internet. After the download is complete, users can access this content from the device's local storage, even when the device is offline. Users are notified when the download of files and images is complete. Use download on sync when you only need to make a limited number of files or images available offline.

The remainder of this article helps you configure on sync.

Configure on sync

To configure the automatic download of files and images on sync, you need to create a custom, mobile offline profile. There are two ways to configure sync.

Configure on sync from within Power apps studio

1. Sign in to Power Apps.
2. In the left navigation pane, select **Apps**.
3. Select a canvas app, and then on the command bar, select **Edit** to open your app in editing mode in canvas app designer.
4. Select **Settings** > **General**.
5. Set the **Can be used offline** toggle to On.
6. Under **Select offline profile**, select **Edit selected profile** from the "...".
7. Select the table that has file or image column that you want to enable for offline.
8. In the table details pane that appears, under the section **Related rows + Files and Images**, select the file and image column that you want to configure on sync.
9. Go back and select **Save**.

Configure on sync from within Power Platform admin center (requires admin access)

You can add both the **Image Descriptor** and **FileAttachment** tables to your mobile offline profile by configuring the settings of your environment.

1. Go to Power Platform admin center and sign-in as an admin.
2. Select **Environments** from the navigation menu.
3. Choose an environment and then select **Settings** on the menu bar.
4. Expand **Users + permissions**, then select **Mobile configuration**.
5. Select a mobile offline profile to edit it.
6. In the **Data available offline** section, select **Add the table that contains the images or files**.
7. To enable offline access for images and files, add the **Image Descriptor** and **FileAttachment** tables to your mobile offline profile. To add the Image Descriptor table, follow these steps:
   a. Select **Image Descriptor** and then select **Next**.
   b. Under **Choose the rows that you want to make available offline**, select **Related rows only**.
   c. Expand **Relationships** and select **Column name: Display name** for each column where **Data type** is set to **Image**. In this example, we select the unnamed column names from the **DemoTable1** and **DemoTable3** tables.
   d. Select **Save**.
8. To add the FileAttachment table, follow these steps:
   a. Select **FileAttachment** and then select **Next**.
   b. Under **Choose the rows that you want to make available offline**, select **Related rows only**.
   c. Expand **Relationships** and select **Column name: My column name** for each column where **Data type** is set to **File**. In this example we add **Column name: MyFile** from the **DemoTable3** table. *Don't* select **Column name: Regarding**..
   d. Select **Save**.

Last updated on 02/20/2026

Work with canvas apps offline

Article • 07/23/2024

If a canvas app is designed for offline use, you can work with it on your mobile device even when you don't have Internet access. All the data the app needs is stored locally on your device. Any changes you make are synchronized with the app server the next time your device connects to the network. If you're in the middle of changing some data and you lose your network connection, the app saves your changes and uploads them to the server when you're back online.

Initial offline sync

Before the app can work offline, it needs to download to your device all the data it needs to have available. This process is called the *initial offline sync*. It can take a few minutes or longer, depending on how the app was designed.

The initial offline sync happens automatically when you open the app. A message tells you the app is syncing the offline data and the number of rows and tables that are downloaded. If the initial sync doesn't complete, it's triggered every time you open the app.

Offline delta sync

After the initial offline sync, the data that's stored locally is kept in sync with data on the app server whenever your device has a network connection. These periodic updates are known as "delta" syncs. "Delta" means the minimum data that's changed between two syncs on the device. Power Apps syncs every five minutes when you're online.

Sync status icons

Depending on how the app was designed, it includes information about its synchronization status in the form of a globe with various icons. At a glance, you can tell whether:

- The app is connected to the network
- A data update is in progress
- Updated data is waiting to sync
- An error or warning occurred while data was syncing

The following table describes the icons and their meanings.

Expand table

Icon	Description
	The app is connected to the Internet.
	The app isn't connected to the Internet.
	The app is syncing data.
	The app has pending changes to upload.
	The synchronization process encountered an error.
	The synchronization process encountered a warning.

To get a description of the sync state, select the icon.

Feedback

Was this page helpful? Yes No

Provide product feedback

Sync data offline in the background

06/25/2025

To use an app in offline mode, download the app and sync data to your device.

You can sync data reliably only when the Power Apps mobile app or Field Service app is open, running in the foreground, and your device screen is unlocked. Learn more about sync behavior in Work with canvas apps offline.

The background sync can be enabled on Windows by changing the **Let this app run in the background** setting.

- If you're using Windows 10, see Manage background activity for apps in Windows for more information.
- If you're using Windows 11, complete the following steps:

    1. Go to **Settings** > **Apps** > **Installed apps**.
    2. Next to **Power Apps** or **Field Service**, select the **More options** icon (three horizontal dots). Then select **Advanced options**.
    3. In the **Background apps permissions** area, go to the **Let this app run in the background** setting. Select **Power optimized (recommended)** from the list.

Mobile offline limitations for canvas apps

07/16/2025

Before you set up the mobile app in offline mode, be sure to read through the following limitations. We recommend that you also review Best practices for developing an app for offline use and Optimize the offline profile.

Capabilities not supported in offline-enabled apps

Expand table

Area	Description
Canvas app types	The offline-first feature works for standalone canvas apps only. It doesn't work for embedded canvas apps, custom pages, or canvas apps in Teams.
Connectors	Non-Dataverse connectors, like SharePoint, aren't supported in offline mode.
Dataverse table types	Virtual tables and elastic tables aren't supported in offline mode.
Shared offline profile	If the same, offline profile is used in a canvas app and in a model-driven app, it creates two, separate local databases.
Power Automate flows	Power Automate flows aren't supported in offline mode.

> **Note**
>
> If your app connects to data from Excel, CSV files, or SharePoint lists, you can start with Copilot to import data to Dataverse. Once your data is in Dataverse, you can turn on offline support for your canvas apps with one selection.

Offline runtime limitations

Expand table

Area	Description
Offline record limit	The total number of records synced is limited to 3,000,000. Attempts to sync a larger number of records fail. This number also includes hidden tables used for offline capabilities.

Area	Description
Power Fx functions	The offline-first feature works with Dataverse tables only and doesn't support the following Power Fx functions: Relate, Unrelate

Relationship	In offline mode, canvas apps don't support many-to-many relationships.
	
	Filtering on column lookups is limited to one level of the relationship when the app is configured for offline use. Self-referential lookups are also not supported in offline mode. Consider following examples to understand the relationship with Account and Contact information.
	
	Supported Lookups:
	
	Lookups are supported for one level of relationship. For example:
	
	`Filter(Account, 'ContactID'.'Zipcode' = "11056")`
	
	This retrieves all accounts with the zipcode 11056.
	
	Unsupported Lookups:
	
	1. **Self-referential Lookups**: Self-referential lookups aren't supported. For example:
	
	`Filter(Account, 'Parent Account'. 'Name' = "John Doe")`
	
	This doesn't work because the parent account is a self-reference to the account table.
	
	2. **Multi-level Relationship Lookups**: Lookups involving more than one level of the relationship aren't supported. For example:
	
	`Filter(Account, 'ContactID'.'Map'.Latitude = "38'53")`
	
	This doesn't work because it involves more than one level of relationship (Account > Contact > Map).
	
	Note: The same limitations apply to the lookup function and the filter function mentioned earlier.

Tables	Notes aren't supported in offline mode in canvas apps.
Column types	Calculated and roll-up fields—that are part of rows synced to the client—aren't reevaluated by the client. The reevaluation happens on the server when the updated row is synced. When you run an app in offline mode, mapped fields aren’t prepopulated when you create a new record from a table that has fields mapped to another table.

Area	Description
Sort order	Items in a gallery may appear in a different order in an offline-capable app if no sort order is selected. Choose a sort order in the gallery control to make sure the app behaves consistently in mobile apps and web browsers.
Background synchronization	Data can only be synced regularly when Power Apps is running in the foreground of your device, with the screen unlocked. Learn more in Sync data offline in the background.

Profile filters limitations

Expand table

Area	Description
Relationship defined for each table	A maximum of 15 relationships is allowed. There's a maximum of one many-to-many (M:M) or one-to-many (1:M) relationships within the 15 relationships. If any custom tables demand this scenario, then revisit the data model. No circular references or self-references are supported.
Images and files	Images and files are subject to the same limitations as any other table. Because of implicitly defined relationships, an offline profile can only contain up to 14 image columns, across all entities.
Auto-generated offline profile	The auto-generated offline profile doesn't handle filters. As a result, for each table used in the app, it downloads *all* rows to which the user has permissions.

See also

Troubleshoot offline sync errors in the Power Apps mobile app

Mobile offline for model-driven apps overview

Article • 05/29/2024

Wouldn't it be great to use your mobile app without worrying about spotty internet connectivity? With model-driven apps made with Power Apps and the new, mobile offline-first experience, it's possible. People who need to work from remote locations can continue to work seamlessly without worrying about their internet connection.

With the new, mobile offline experience enabled for your model-driven apps, not only can you continue working in offline mode, you also have better device performance, a more responsive app, and less drain on the battery because fewer connections are being made to the server.

An internet connection is required to sync data between your mobile device and Microsoft Dataverse.

To get started with mobile offline, an app maker needs to enable and define the tables that are available for offline use by using the modern app designer. For more information, go to Set up mobile offline.

Mobile offline is offline-first by default

Offline-first means that all the data you might need when offline is copied to your mobile device. This requires initial network access to download the data. Once you have your data, you only work with the data on your local device all the time. This is true both when your device is connected and disconnected to the network.

Moving in and out of network access doesn't affect performance of the app because it's using local data. Power Apps monitors network access and automatically syncs the changes you make locally with the server, and downloads any updates made on the server. The offline features automatically handle spotty network connections, download data, upload data, handle conflict detection, and more. The built-in, offline features minimize system resources and are highly performant.

When offline mode is configured and enabled for your model-driven app, anyone who uses the Power Apps mobile app can also use the app in offline mode. By default, the Power Apps mobile app runs offline-first, either with or without an internet connection. This functionality optimizes offline performance and creates a consistent experience for users as they change locations:

- **Online**: Occurs when an internet connection is available, but offline mode isn't set up. The mobile app functions similar to using the app with an internet connection on your PC. When the internet connection is lost, the mobile app is unusable. This isn't recommended.
- **Offline-first without an internet connection**: Data is downloaded and all changes are saved to your mobile device. When the internet connection is restored, the changes are automatically synced to the server.
- **Offline-first with an internet connection**: Data is downloaded to the device and all changes are saved locally. Because there's an internet connection, the app automatically attempts to sync every few minutes.

> **ⓘ Important**
>
> For more information about how to set up mobile offline for canvas apps, go to Develop offline-capable canvas apps

Offline-first vs. classic offline

With the offline-first experience, it's important to understand the key benefits of the *offline-first experience* versus the *classic offline experience*.

Expand table

Offline-first	Classic offline
• Your data is always the same, regardless of your network connection. • There's no toggle for users to switch from offline to online mode. A user never forgets to sync their changes back to the server because the app does it automatically.	• Users have the option to skip the initial offline sync and stay online, which means that users in your organization might not have the same experience. • Users have to remember to disable the **Work in offline mode** toggle before they can sync changes with the server. • To save your changes in offline mode, you need to set the **Work in offline mode** toggle to **On**. Otherwise, when you're working on a row and suddenly lose your internet connection, your changes are lost. • The rows you see in offline mode are listed from the local database. This means that the rows you see in offline mode vs. online mode can be different.

Feedback

Was this page helpful? Yes No

Provide product feedback

Set up mobile offline for model-driven apps

07/31/2025

Use the modern app designer to enable your model-driven apps for offline use on a mobile device. When offline mode is enabled, users interact with their data without internet connectivity on the Power Apps mobile app.

Prerequisites

Verify the following prerequisites before you enable mobile offline:

- You need the environment maker, system administrator, or system customizer role to configure offline mode for model-driven apps. These roles have Create, Read, Write, Delete, and Share privileges on the **Mobile offline profile** table. For more information, see About predefined security roles
- Users with the basic user role can open and use an offline app. This role has the Read privilege for the **Mobile offline profile** table.

  If you have a custom security role, ensure you have the read privilege for the **Mobile offline profile** table. For more information, see Miscellaneous privileges

Optimize your app for mobile offline

Mobile apps run on smaller screens with limited connectivity. Before you enable offline mode, make sure your model-driven app is optimized for offline and mobile use. Consider the number of user scenarios that you want to cover and the amount of data the app uses. Create an app that's simple and lightweight.

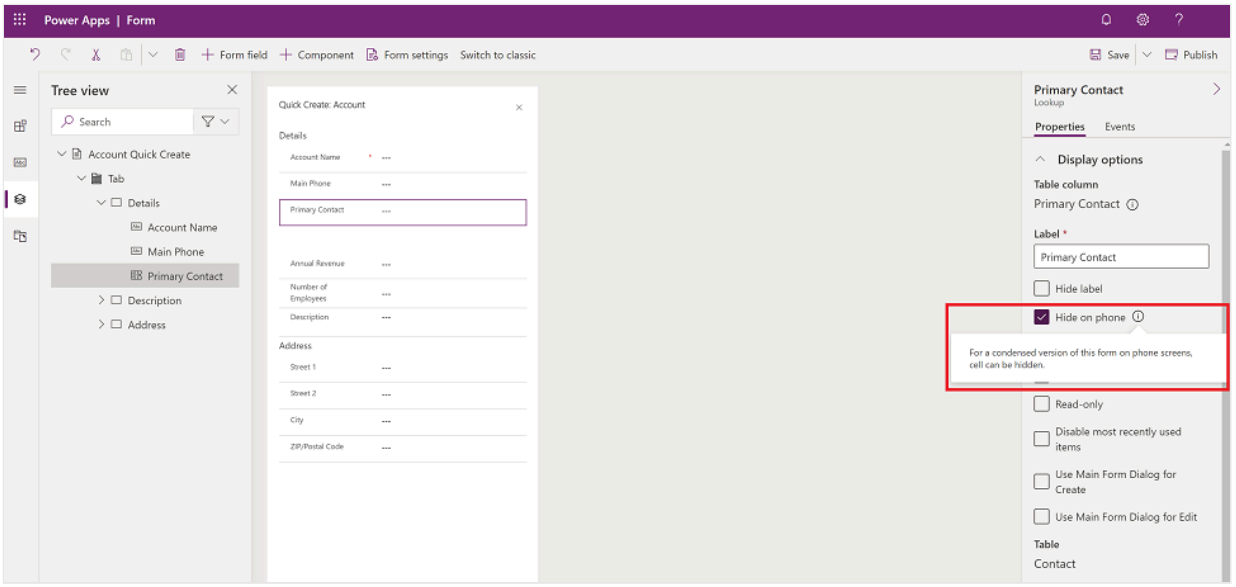
If you have desktop and remote mobile users, optimize the user experience by creating two separate apps. Create an online app for your office users and another app for your mobile users who might have limited connectivity.

Follow these best practices when building an app for mobile offline use:

- Identify the on-the-go scenarios that functionally relate, such as tasks performed by users working in the field.
- Reduce the complexity of your app by limiting the amount of app metadata that needs to be downloaded on a user's device. Only add the tables and views that are needed for

your mobile users. For more information, go to Add a page to your app

- Only keep views that are necessary, and remove any that aren't needed on a day-to-day basis. For example, keep the **My active accounts** view and remove the **All accounts** view. Keep your forms lightweight for a smooth and intuitive experience on small-screen devices. The following are possible optimizations for forms on mobile:
  - Build dedicated forms for mobile use.
  - Share forms across the mobile and desktop experience, but disable some fields on mobile.



Enable your app for offline use

1. Sign in to Power Apps.
2. On the left pane, select **Apps**, and then select the model-driven app that you want to enable for offline.
3. Select **More (...)** > **Edit** to open the modern app designer.
4. On the command bar, select **Settings**.
5. On the **General** tab, set the **Can be used offline** toggle to **On**.
6. Select the type of offline profile: **Default** or **Restricted to selected users**.
   - If you select **Default**, the app is enabled for offline use for all users. Select an existing offline profile, or select **New profile with current app data**. To set up a new offline profile, see Set up a mobile offline profile.

- If you select **Restricted to selected users**, the app is enabled for offline use only for a selected list of users. Select one or multiple offline profiles, or select **New profile with current app data**. To set up a new offline profile, see Set up a mobile offline profile.
- You must have admin privileges to assign users to an offline profile.

7. Close the **Settings** page, and then save and publish your app. When you're done, the app is set up for offline use.

Set up a mobile offline profile

The mobile offline profile represents the dataset that syncs on a user's device. The profile contains the tables, including related filters, that are applied when data is synced to a user's device. For more information about offline profiles, see Offline profile guidelines.

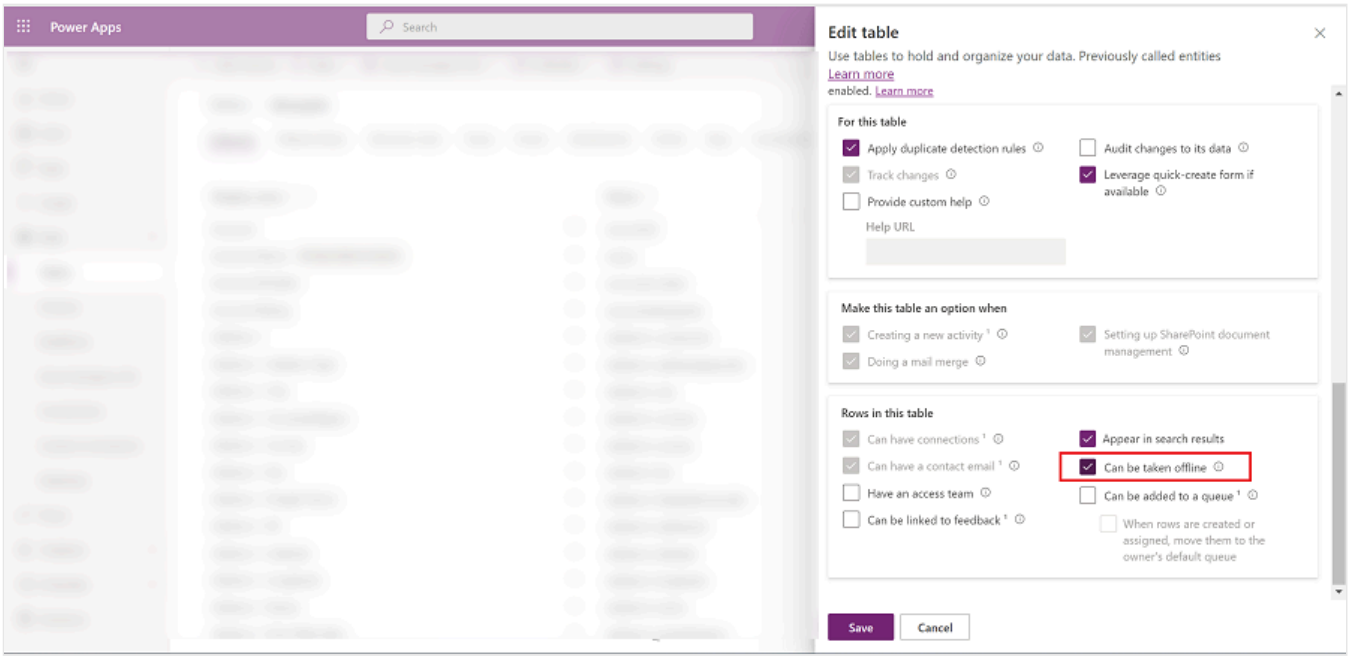
If the offline profile is in advanced mode, you'll need to add users to the offline profile. If you don't add a user to the offline profile, the user can't use the app in offline mode.

Enable a table for offline

A table needs to be enabled for offline to add it to an offline profile. Some tables are enabled for offline by default. Follow these steps to verify whether a table is enabled for offline.

1. Sign in to Power Apps.
2. In the left pane, select **Data** > **Tables**.
3. Select the table that you want to add to an offline profile, and then, on the command bar, select **Settings**.
4. In the **Edit table** settings, select **Advanced options**, and in the **Rows in this table** section, select **Can be taken offline**.

An app can only be linked to one profile. However, a profile can be shared between multiple apps. This can be useful when multiple apps share the same dataset, as it only needs to be downloaded once on the device and is shared between the apps.

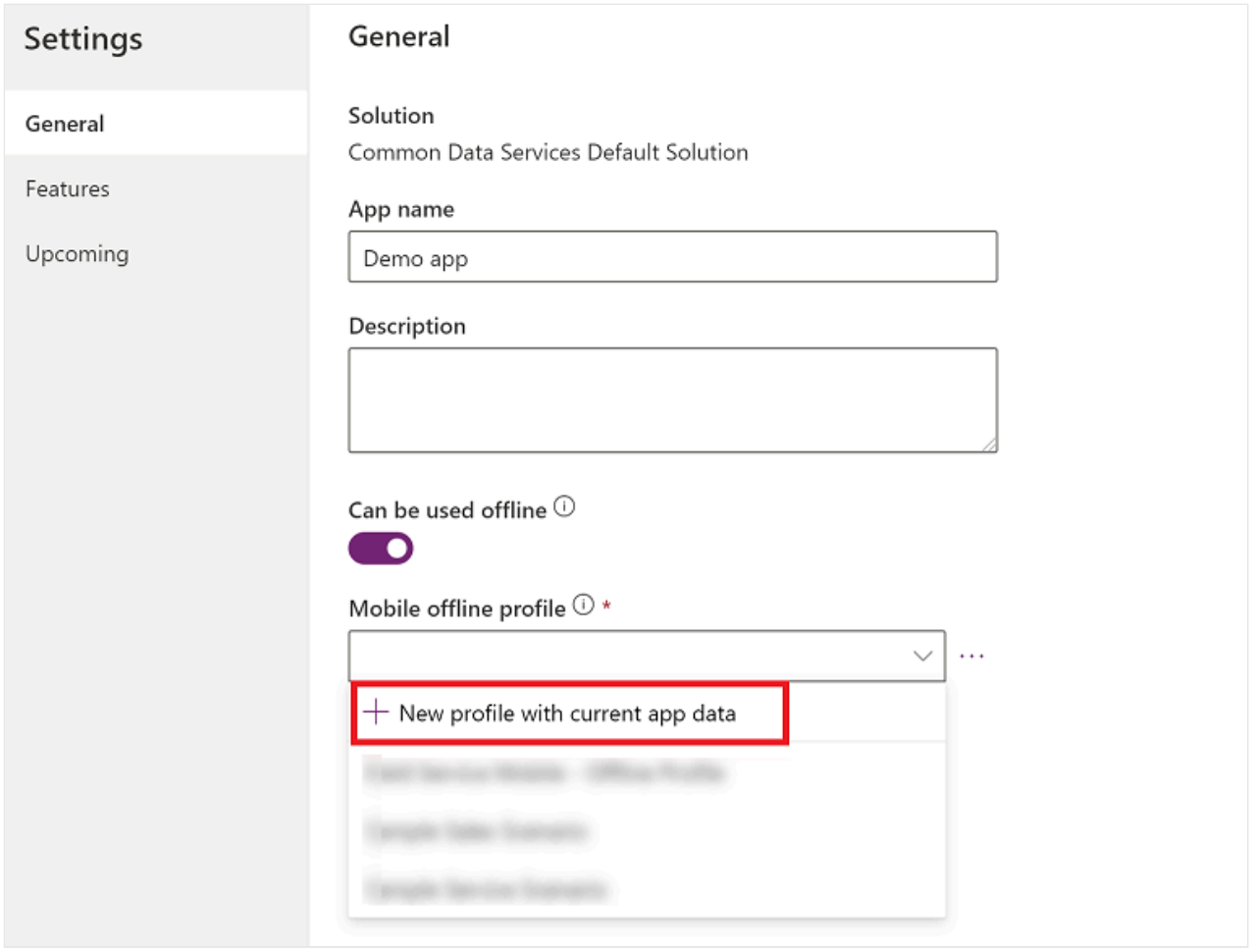


Generate a default profile

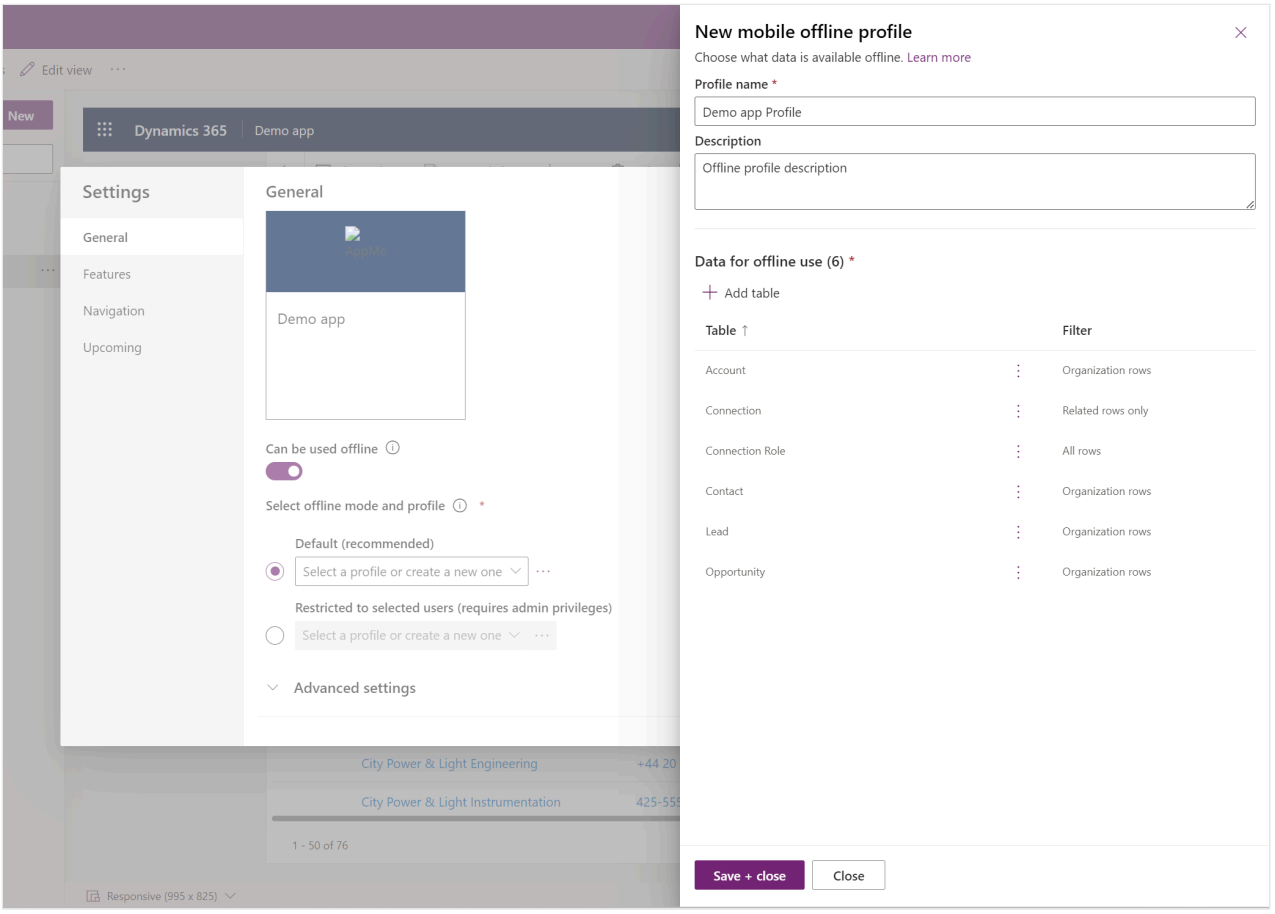
The modern app designer generates a default offline profile based on how the app is configured.

The default offline profile is a starting point to help you quickly build an offline profile. The default profile doesn't compute the optimal filters for each table. It's important that you review and adjust the proposed filters based on your organization's needs.

1. Select **New profile with current app data**.



2. Review the proposed filters for each table. Make sure that the data downloaded on users' devices is limited to only what's necessary. Focus on the tables used most often in your app, which in most cases have the **Organization rows** filter set.



The tables that are added to the profile also have the **Related rows only** filter. These are tables used in certain views that require related information. You might not need to modify them according to your organization's needs.

Add a table to an offline profile and apply filters

Applying an appropriate filter for each of the tables configured in the offline profile is critical to limiting the amount of data that downloads on users' devices.

Be sure that you configure at least one of the profile rules for each table to download its data.

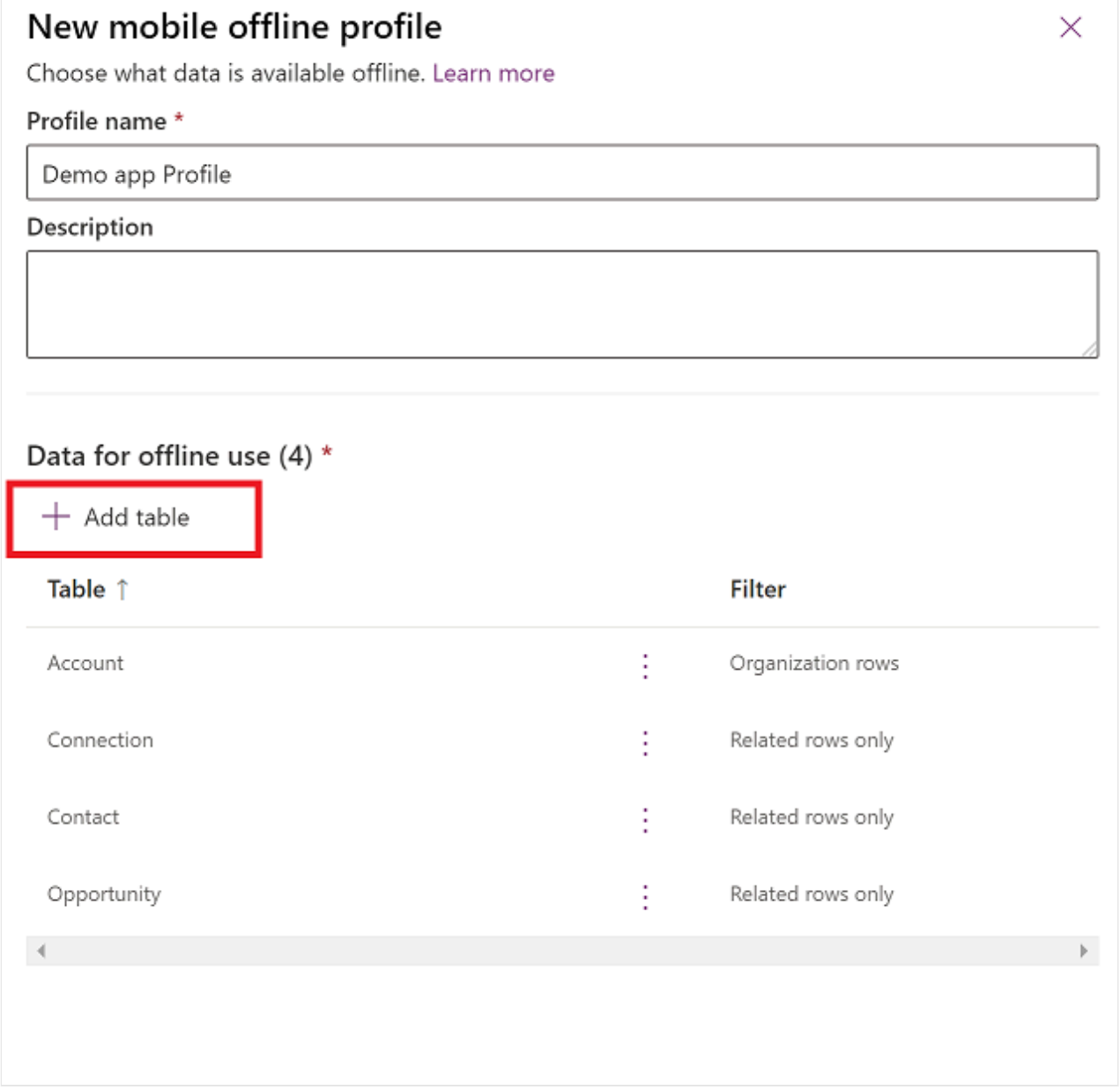
Expand table

Customization	Recommendation
Organization rows - if selected, then select at least one of these options: - **User's rows** - **Team rows** - **Business unit rows**	If you want to define this filter, then you have to pick at least one of the given options. It's highly recommended to not have a business unit-level filter for a table, unless there's a strong justification. It's recommended for a master data scenario with a large data set, like a price list.
All rows	If you're selecting this filter, you can't define any other filter rules.

Customization	Recommendation
Related rows only	Be sure that the related table has been added to the offline profile.
Custom	You can define a custom filter up to three levels deep.

Keep in mind that you can have 15 related tables in a custom filter. You can also have 15 relationships. These are distinct checks that might not add up. The 15-relationships limit is transitive, meaning if table B has N relationships, and you add a reference to table B in table A, then it increases the relationship count of A by N+1; one plus the N already in table B. This limit is per profile item for a table in the profile.

1. Select **Add table**.



2. Choose a table, and then define the filters.

3. Set the following filters:

a. Choose the rows that you want to make available offline. For the **Custom** option, use the expression builder to set up advanced conditions.

b. **Relationships** lists the different relationships available between the current table and other tables added in the offline profile. Selecting a relationship ensures that related rows following that relationship are downloaded and made available offline.

c. **Files** and **Images** define which columns for a file or image need to be downloaded offline. For files, you can choose to download every column or none at all. For images, you can select each column you want to download granularly.

d. **Sync interval** defines the sync frequency to be applied on the device to sync the data with the server. If a table's data doesn't change frequently, like a catalog or a product table, you might want to focus on only syncing data when necessary, such as refreshing only once a day.

   You can only have up to 15 related tables in a profile. If you exceed the limit, you get an error and won't be able to publish the offline profile.

4. Select **Add + save** to add your table and filters to the profile.

5. When all tables are properly configured for the profile, select **Done** > **Publish your application**.

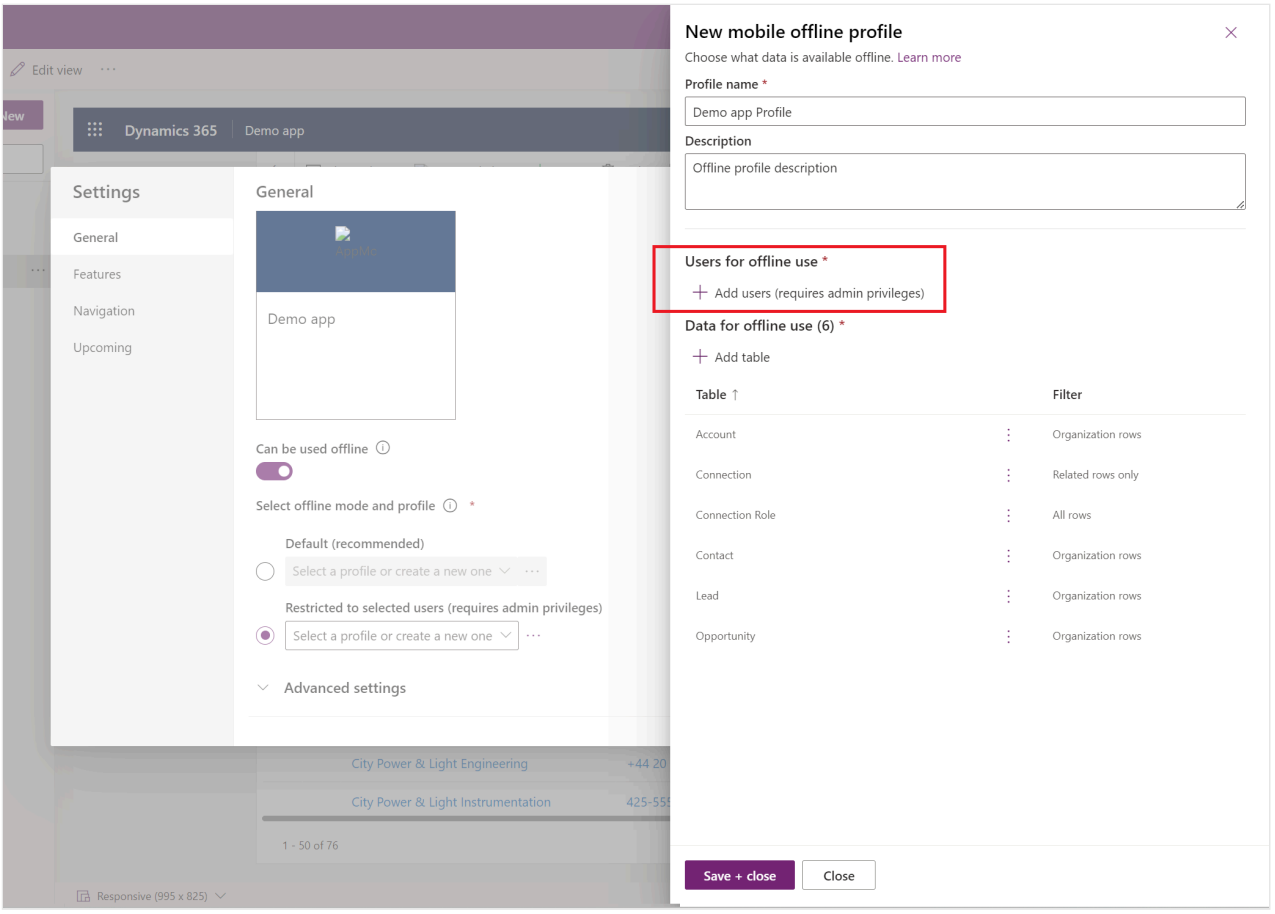
   When the app is published, the app is enabled for offline use.

Add users to an offline profile

If you selected the **Default** offline profile mode, all your users who have access to the app can also use it in offline mode.

However, if you selected **Restricted to selected users**, you have to manually add users (requires admin privileges) to the offline profile. The app is enabled for offline use only for those configured users.

1. Select **Add users (requires admin privileges)**.



2. Sign in to the Power Platform admin center.
3. In the **Users with offline access** area, select **Add users**.
4. Select the users that you want to add to the mobile offline profile. The users that are added, based on teams, are listed under each team. Users that are added individually are listed as individual users.
5. When you're done adding people, select **Save**.

   If you made any changes to a Microsoft Entra ID group team, you must remove and add the team back to the mobile offline profile for the changes to take effect.

Enable mobile offline classic

Offline-first is the default mode when you enable an app for offline use. If you want to switch to the offline classic mode, you need to enable it for your model-driven app. It's a separate app setting for each app.

1. Sign in to Power Apps.
2. On the left pane, select **Apps**, and then select the model-driven app that you want to enable for offline.

3. Select **More** (...) > **Edit** to open the modern app designer.
4. On the command bar, select **Settings**.
5. On the **Features** tab, set the **Enable offline classic** toggle to **Yes**.
6. Select **Save** and then publish the app.

Define sync settings on mobile

You can allow users to control the automatic sync intervals or the connection type to synchronize the data.

1. On the command bar, select **Settings**.
2. Select the **Upcoming** tab:
    - Set the **Allow users to adjust sync frequency** option to **On** to adjust the sync interval to sync more or less frequently depending on individual needs. Users can choose to not automatically sync, if they only want to sync on demand.
    - Set the **Allow users to sync on Wi-Fi only** option to **On** to let users choose if their automatic sync happens on cellular networks and Wi-Fi connections or only when connected to a Wi-Fi network.

Turn on Online mode

Offline-first mode provides the best experience for users in no and low network conditions. Once your data is synchronized, you work with the data on your local device regardless of the network status. However, if your offline profile includes filters, you may not be able to access all the data you could when online in a browser, even when your device is connected to the network.

If it's important for users to access data that's not downloaded to their devices, you can activate the **Online mode** toggle that's visible to users in the Device status page. When users use the **Online mode** toggle, they can access data that's not synchronized to the local device. The toggle can be activated only when the device is connected to a network.

1. On the command bar, select **Settings**.
2. Select the **Upcoming** tab.
3. Set the **Allow users to work in online mode** option to **On** to let users choose if they want to use the app in online mode when connected to a network.

Supported capabilities

- **Business rules**: Business rules are supported in mobile offline. For more information, see Create business rules to apply logic in a model-driven app form.

- **Business process flows**: You can use business process flows in offline mode if the following conditions are met:
  - The business process flow is used in an app that you can run on Power Apps mobile.
  - The Power Apps mobile app is configured for offline use.
  - The business process flow has a single table.
  - The business process flow table is added in the offline profile.

  > ⓘ Note
  >
  > If a table is associated with multiple business process flows, in order for any of the business process flows to work offline, all the business process flows must be added to the offline profile.

  Three commands are available for business process flows when you run an app in offline mode on the Power Apps mobile app:
  - Next stage
  - Previous stage
  - Set active stage

  For more information, see Run business process flows offline.

- **Lookup support**: Lookups are supported for the tables that are configured for mobile offline. All the tables participating in the lookup should also be configured for offline use.

- **Notes on the Timeline control**: Notes on the Timeline control are available in offline mode. You can take pictures, read notes, and add/remove attachments in offline mode. You need to add the **Note** table using the **Related rows only** option and the User (systemuser) in the offline profile.

  > ⓘ Note
  >
  > The **Date** field isn't available for mobile offline search.

- **Custom tables**: These commands are available to edit the form: **Mark Complete**, **Refresh**, **Convert To Opportunity**, and **Delete**.

Configure files and images in offline model-driven apps

Article • 05/29/2024

To work with file and image columns in offline mode, you need to add them to your mobile offline profile.

This article helps you configure a mobile offline profile that has a table with a column where **Data type** is set to **File** or **Image**.

View column properties for a file or image

1. Sign in to Power Apps.
2. In the navigation menu, select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the table that has the columns you want to view.
4. Under **Schema**, select **Columns**.
5. Select the **Display name** of a column where **Data type** is set to **File** or **Image**.
6. The column properties show the **Data type**. Expand **Advanced options** to view the maximum size for a file or image. For more information about columns, see Columns overview.

Add file and image columns to mobile offline

Add the column where **Data Type** is set to **File** or **Image** to your mobile offline profile to make files and images available in offline mode.

1. Sign in to Power Apps.
2. In the left navigation pane, select **Apps**.
3. Select a model-driven app and on the command bar select **Edit**. This opens your app in editing mode in model-driven app designer.
4. Select **Settings** > **General**.
5. Set the **Can be used offline** toggle to **On**.

6. Under **Select offline mode and profile**, select **Edit selected profile** from the "...".
7. Find the table that contains the file or image column to enable for offline, and then select **Edit** from the **More actions** button.
8. In the **Include these files and images** section, select the column where **Data Type** is set to **File** or **Image** (the columns are grouped by **Files** or **Images**).
9. Select **Save**.
10. Save and publish the app.

Feedback

Was this page helpful? Yes No

Provide product feedback

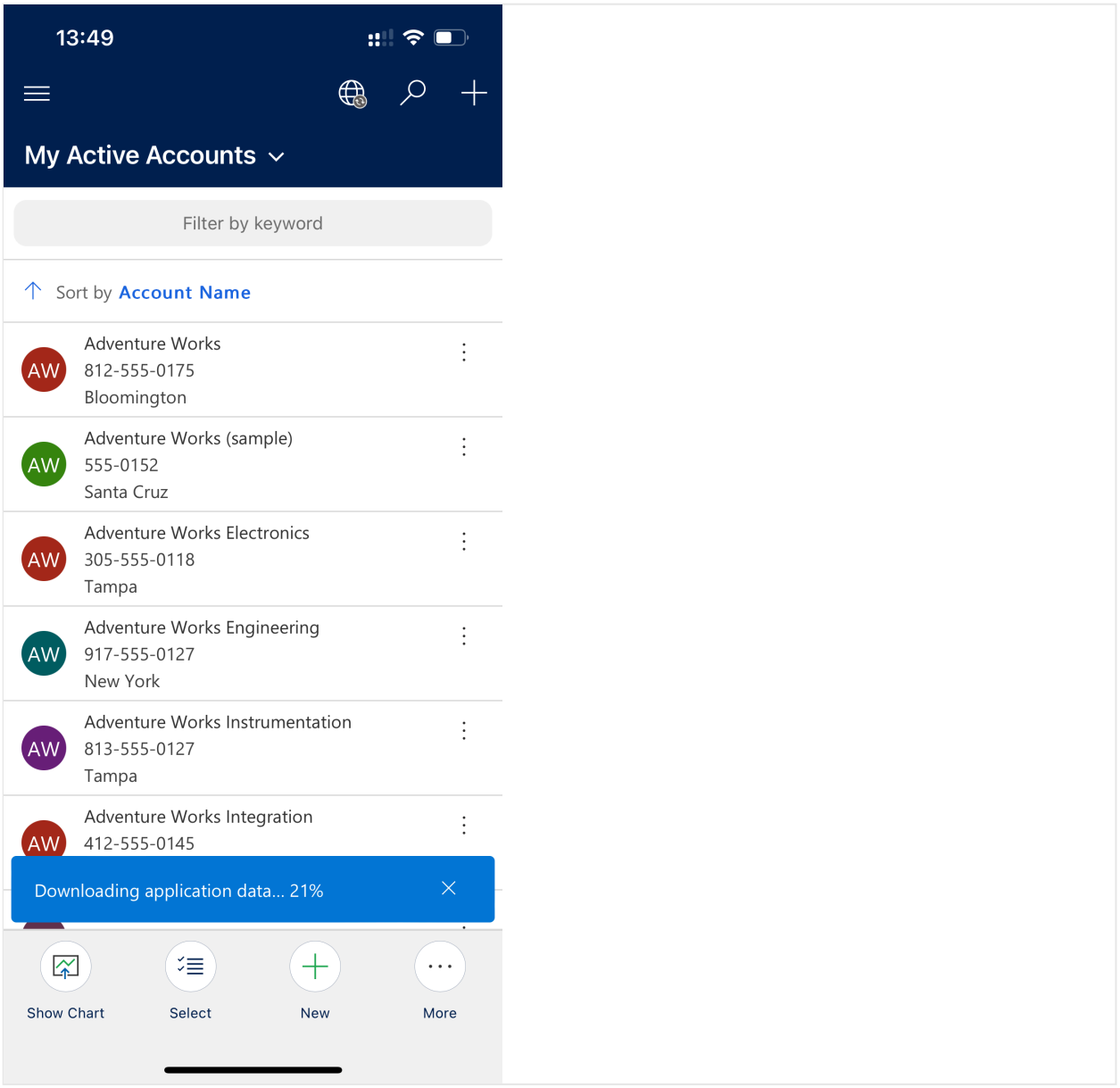
Work with model-driven apps offline (offline-first mode)

Article • 07/23/2024

For a model-driven app to be available in offline mode, app and user data must be downloaded on your device. This process is called *initial offline sync*.

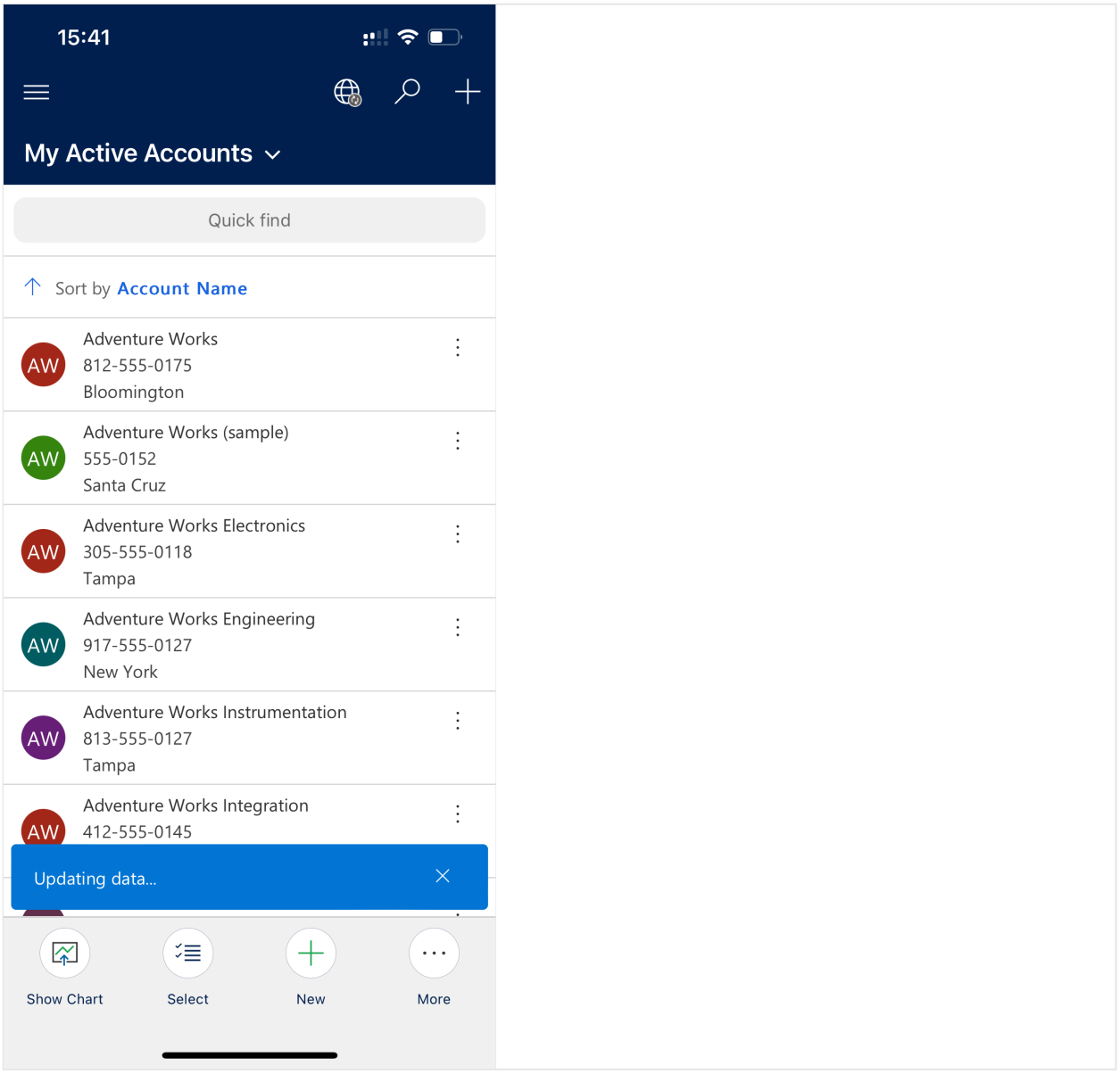
The app data includes all the resources needed for the app to run properly. User data includes data stored in Microsoft Dataverse tables configured in the offline profile.

During the initial offline sync, you receive a notification stating that your app is syncing the offline data with the number of rows downloaded. The notification also indicates how much disk space is used. The initial offline sync can take a few minutes or longer, depending on what is configured in the offline profile. If the initial offline sync isn't complete, it triggers every time the app is opened. You also get the same notification when you resume the app after a long period of inactivity.

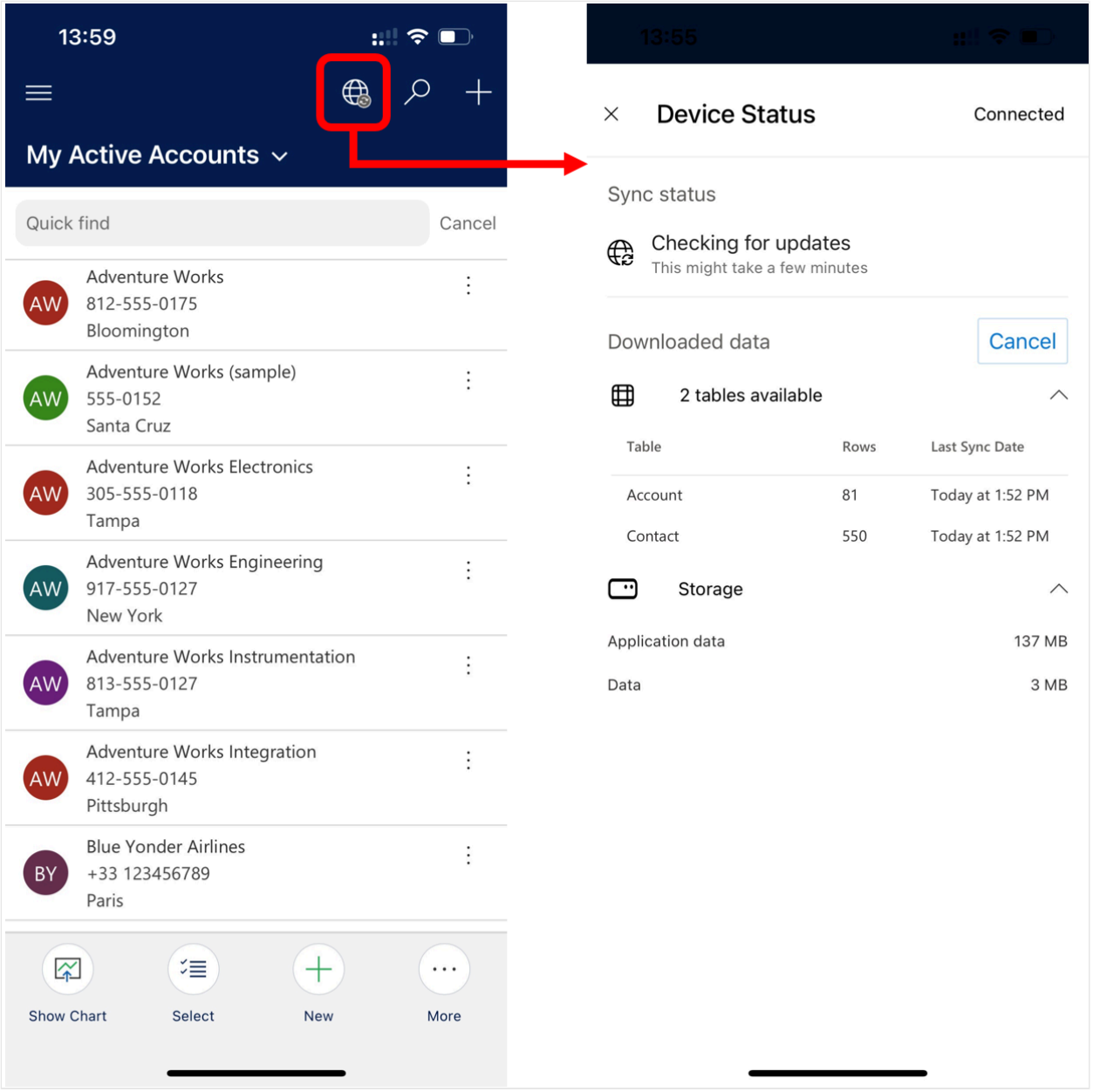


When the initial sync is completed, you can start using the app in offline mode.

When you refresh your data on a grid view, you get a notification stating, **Updating data....**



The sync status is always available, at a glance, with the **offline sync** (globe) icon. For more information about offline sync status, see Offline sync icon.



Feedback

Was this page helpful? Yes No

Provide product feedback

Work with model-driven apps offline (classic mode)

Article • 07/23/2024

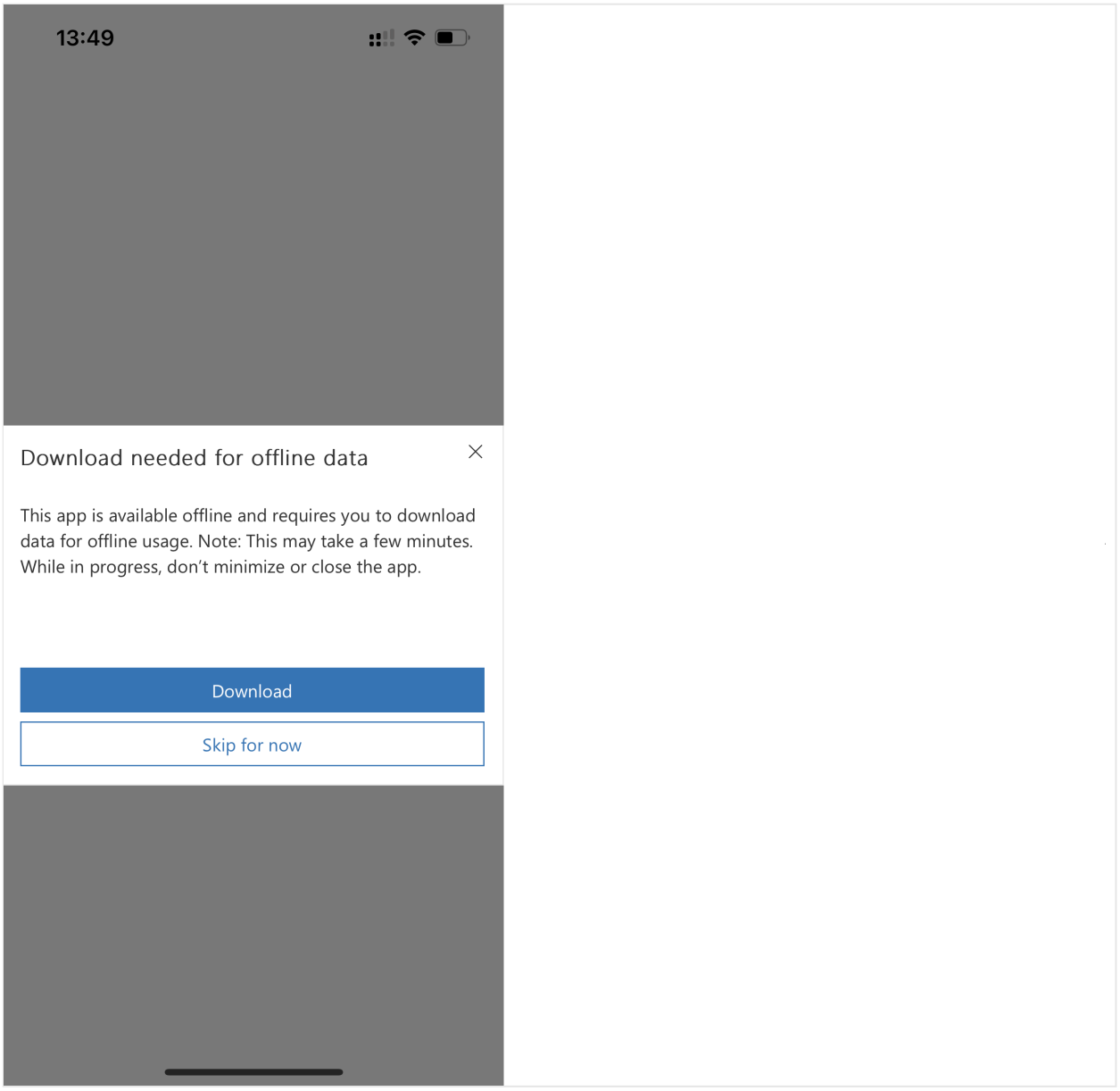
You can work with your data even when you don't have internet access by using the app in offline mode. The mobile app provides a rich offline experience that lets you work with commands like create, read, update, and delete, so you always stay productive. After you're back online, changes are synchronized with your apps in the Microsoft Dataverse environment.

If you're working on a record and lose your network connection, any updates made to the record are saved in offline mode and will be synchronized to the app after you're back online. When the record is synchronized with the app, it uses the filters, added on the table in the offline profile, for availability in offline mode.

Download updates to work in offline mode

When your admin enables mobile offline classic mode, you are prompted to download offline updates. After downloading the updates, you can start using the app on your mobile device in offline mode.

When you see the dialog box that asks you to download updates to work offline, select **Download**.



Offline data starts downloading as you continue to use the app. Data is downloaded while the app is active, so keep your phone unlocked with the app visible until the initial download is complete. The data can be downloaded in the background depending on the platform you use.

Download offline data later

If you don't want to download offline data and select **Skip for now**, you can't use the app in offline mode until you manually download the updates.

1. On the screen, select the **offline sync** (globe) icon.
2. On the **Device Status** page, select **Download offline updates**.
3. The download starts, and the status changes to **Initializing**.

4. When the download of offline data is complete, you get a notification that you can start working in offline mode.

See whether offline mode is available

When offline updates are complete, you can check to see whether mobile offline mode is available. On the home screen, select the **offline sync** (globe) icon and check on the Device Status page.

Work in offline mode

After the offline download is complete, you can use the mobile app in offline mode.

When you have no connectivity, you automatically have access to the downloaded data so you can continue working while you're on the go. The data automatically syncs with the server as soon as connectivity is restored.

When you have intermittent connectivity, we recommend that you switch to offline mode. As long as offline mode is turned on, the updates you make on your device aren't synced with the server and you can continue working with the local data on your device. To push the changes made to the server and refresh your local data, turn off the **Work in offline mode** toggle.

To work in offline mode, complete the following steps:

1. On the screen, select the **offline sync** (globe) icon.
2. Turn on the **Work in offline mode** toggle.

Feedback

Was this page helpful? Yes No

Provide product feedback

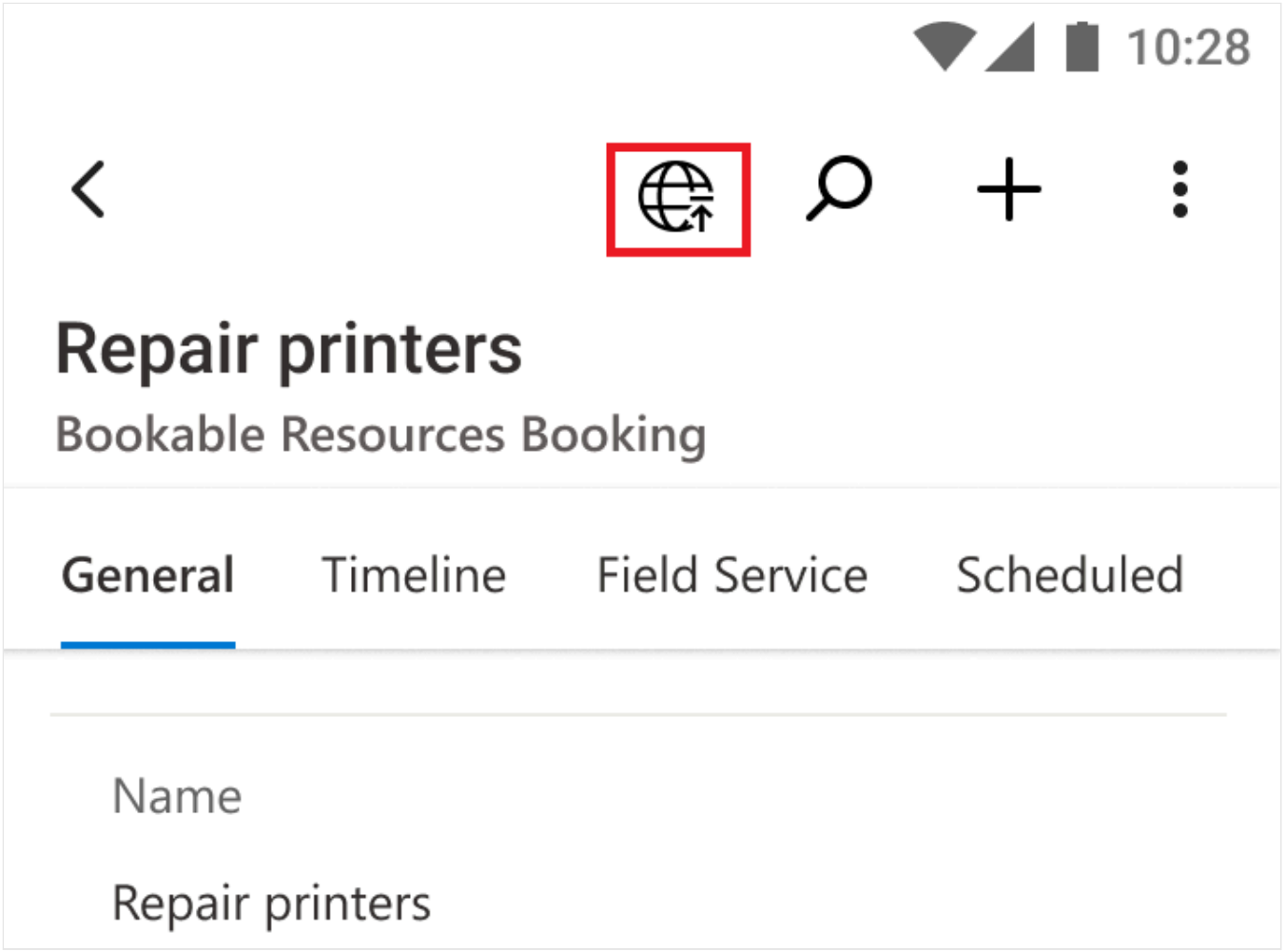
View offline sync status

07/11/2025

The **offline sync** (globe) icon indicates the synchronization status of the Power Apps mobile app. At a glance, you can tell whether:

- The app is connected to the network
- A data refresh is in progress
- Updated data on the device is waiting to synchronize with the server
- The app experienced an error or warning while syncing

The offline sync icon is always visible in the main app navigation bar on iOS, Android, and Windows devices. (It's not visible in web browsers.)



Sync status icons

The **offline sync** (globe) icon changes based on the app's sync status. The following table describes the possible sync states and the icons associated with them.

Expand table

Icon	Description
	The app is connected to the internet.
	The app isn't connected to the internet.
	The app is syncing data.
	The app has pending changes to upload.
	The synchronization process encountered an error.
	The synchronization process encountered a warning.

Types of offline syncs

Data is synchronized from Dataverse to your device over the lifetime of your app using several synchronization modes.

- **First sync**: This offline sync is the initial offline sync that occurs for the Power Apps mobile app. All data in the offline profile is downloaded to the device. A user must use the Power Apps mobile app in online mode until the first sync is completed.

  The first sync, which is the longest, is triggered when you sign in to the Power Apps mobile app. It also triggers after using the Reconfigure action (not recommended).

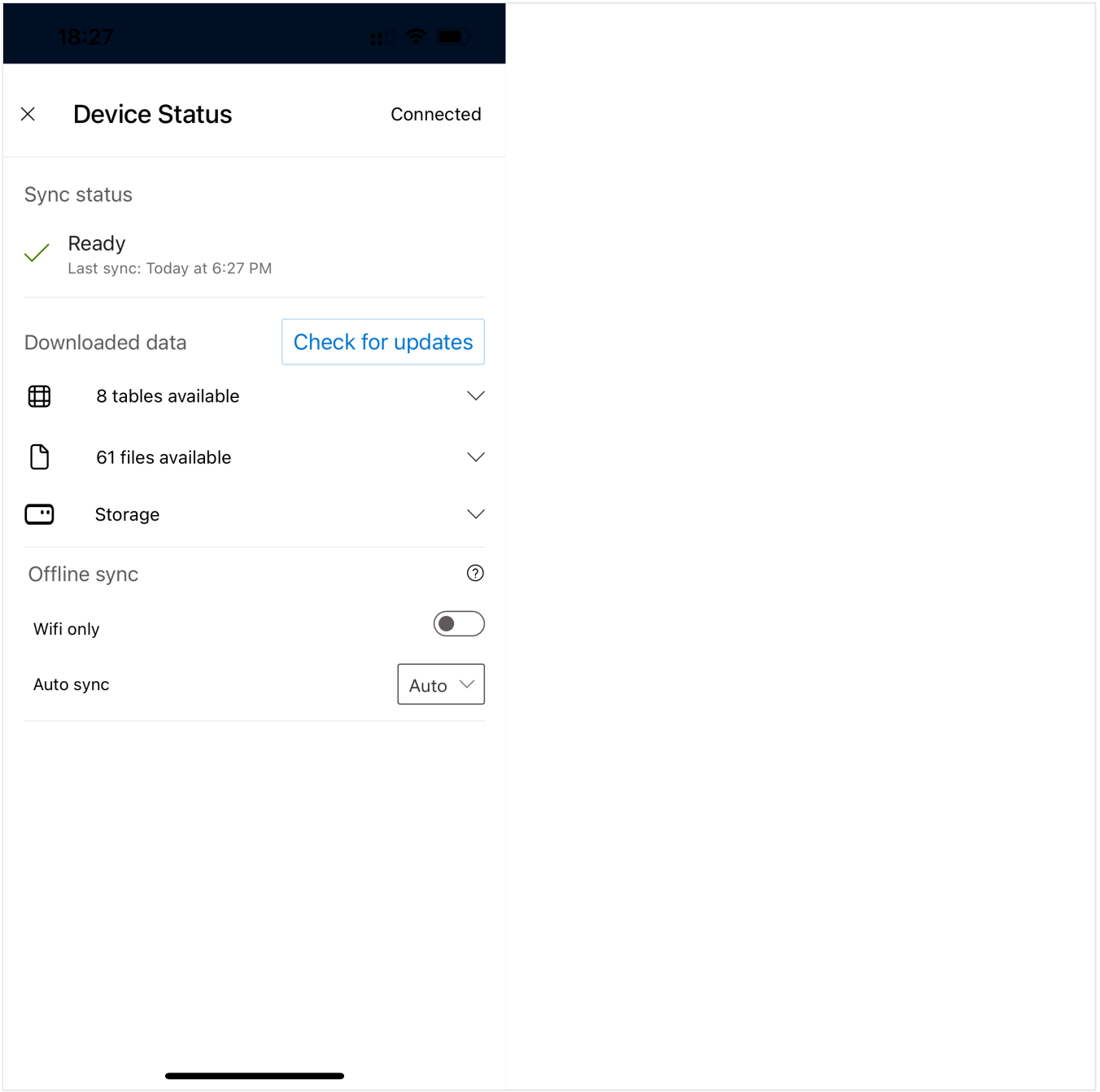
- **Delta sync**: This offline sync is faster than the first sync and occurs regularly based on the offline configuration for the organization. For example, a delta sync could occur every five minutes. Only data that changed since the last sync downloads to the mobile device.

- **Full delta sync**: When you select **Refresh** on the **Device status** page, you trigger a full sync. A full sync doesn't redownload data, but it does recheck every record on the device, so it might take longer than a regular delta sync.

- **Grid sync**: If you select **Refresh** on a grid or calendar view, the app immediately synchronizes all tables visible in the view. When the sync is complete, the view refreshes automatically.

Device Status page

When you select the offline sync icon, the **Device Status** page opens. The **Device Status** page provides details such as the network status, what data was downloaded, whether data is waiting to be uploaded, or the amount of storage the app uses.



Sync status

The **Sync status** area of the page tracks the overall offline status of the app. If the status is in error, see Troubleshoot offline sync errors and the **Downloaded data** area provides the status of each individual table. When the status appears as **Available**, offline mode is available and data synced successfully.

The status can be **Not available** due to an error during the first sync or if you dismissed downloading offline data (only applies to offline classic mode). If the status is **Not available**, try downloading the updates again.

You can use the **Device Status** page to see the number of files and images to be downloaded and the current progress in the **Files available** section.

Offline sync settings

If your admin activated sync settings for the app, you can control when offline synchronizations are triggered.

- Set the **Wifi only** setting to **On** to sync your data only when the device is connected to a Wi-Fi network. Turn on this setting to reduce cellular data and battery usage, especially in low-network areas.
- If you want to sync less frequently, change the sync interval value from the **Auto sync** setting. You can keep the default value, as defined by your admin, by selecting **Auto** or you can pick a longer interval that meets your needs. If you choose **Manual**, the sync only happens when you select the **Check for updates** button in the **Device Status** page.

Online mode

If your admin activated online mode for the app, you can temporarily switch the app to online mode.

If you want to access online data directly from the server and your device is connected to the network, set the **Online mode** setting to **On**. In this mode, the app loads data directly from the server in grids, forms, and search pages. You can change back to offline mode anytime by setting the **Online mode** to **Off**. If you close the app and open it again later, **Online mode** is automatically reset to **Off**.

When you work in online mode and the device loses network connectivity, you see the notification: **Network disconnected. Switch to offline mode to continue working with the app.** Use the **Switch** button to open the **Devices Status** page and set **Online mode** to **Off**.

Notifications

On iOS, Android, and Windows devices, app sync notifications are consistent across the different sync states.

Known issues

- The first time the app syncs, the offline sync icon continues to spin until the user switches from online mode to offline mode.
- If the app has pending changes to upload, the number shown is the total of individual updates applied to rows, files, and images in your app, as opposed to the number of updated rows, files, and images.

Related information

Troubleshoot offline sync errors in the Power Apps mobile app

Sync data offline in the background

06/25/2025

To use an app in offline mode, download the app and sync data to your device.

You can sync data reliably only when the Power Apps mobile app or Field Service app is open, running in the foreground, and your device screen is unlocked. Learn more about sync behavior in Work with canvas apps offline.

The background sync can be enabled on Windows by changing the **Let this app run in the background** setting.

- If you're using Windows 10, see Manage background activity for apps in Windows for more information.
- If you're using Windows 11, complete the following steps:

    1. Go to **Settings** > **Apps** > **Installed apps**.
    2. Next to **Power Apps** or **Field Service**, select the **More options** icon (three horizontal dots). Then select **Advanced options**.
    3. In the **Background apps permissions** area, go to the **Let this app run in the background** setting. Select **Power optimized (recommended)** from the list.

Mobile offline limitations for model-driven apps

Article • 09/26/2024

Before you set up the mobile app in offline mode, be sure to read through the following limitations. We recommend that you also review the Offline best practices and the Offline profile guidelines.

These tables and corresponding permissions are available in offline mode.

> **Note**
>
> Custom permissions and default permissions without the **Mscrm.IsEntityAvailableForUserInMocaOffline** rule won't be displayed in offline-first apps. Learn more in A button on the command bar is hidden when it should be visible in Power Apps.

Expand table

Table	Permissions
Account	Create, Read, Update, Delete
Activity Pointer	Read, Delete
Appointment	Create, Read, Update, Delete, Convert Activity Case
Attachment	Read only
Case	Create, Read, Update, Delete
Competitor	Create, Read, Update, Delete
Competitor Address	Create, Read, Update, Delete
Connection	Read only
Connection Role	Read only
Contact	Create, Read, Update, Delete
Custom table	Create, Read, Update, Delete
Email	Read only

Table	Permissions
Lead	Create, Read, Update, Delete, Qualify/Disqualify
Opportunity	Create, Read, Update, Delete
Opportunity Product	Create, Read, Update, Delete
Phone Call	Create, Read, Update, Delete
Position	Create, Read, Update, Delete
Product	Read
Task	Create, Read, Update, Delete
Team	Read only
User	Read only
Bookable Resource Booking	Create, Read, Update
Work Order	Create, Read, Update
Work Order Product	Create, Read, Update
Work Order Service	Create, Read, Update
Work Order Service Task	Create, Read, Update

Feature limitations

- **Number of records synced**: The total number of records synced is limited to 3,000,000. Attempts to sync a larger number of records fail. This number also includes hidden tables used for offline capabilities.
- **Security model**: Field level security and field sharing aren't supported in Mobile offline mode.
- **Qualify a lead**: When a lead created in mobile offline is qualified and when the user goes online, the business process stage shows the qualified stage. The user has to manually select **Next stage** to move to the next stage.
- **Grids**:
  - Column filtering is disabled when an offline profile is set up even when there's network connectivity. The grid works with the local database and doesn't support custom filters.

  - Primary image thumbnails aren't available for offline unless they were previously displayed online.
  - Custom icons alongside values in grids aren't available for offline. For more information about this capability, go to How to display custom icons.

- **Supported type of views**: Only system views and quick views are supported in mobile offline. Personal views aren't supported.

- **Views** aren't supported for the following tables in offline mode:

  - Email
  - Task
  - Fax
  - Letter
  - Serviceappointment
  - Campaignresponse
  - Campaignactivity
  - Recurringappointmentmaster
  - Socialactivity

  Any views that have linked tables (related table) that aren't available offline are also not supported.

- **Advanced lookups**: The Advanced lookup functionality isn't supported in offline scenarios. When you're offline, the **Advanced lookup** button is replaced by a **Change view** dropdown list.

- **Add Existing on subgrids**: The **Add Existing** functionality isn't supported for mobile offline for certain types of relationships. When you're offline, certain relationships such as relationships that are N:N are read only, and the **Add Existing** command is hidden on subgrids for those relationships. Even if the button is unhidden via customization, the command doesn't work in offline.

- **Web resources**: Web resources are partially supported in offline mode. For more information, see the following table.

Expand table

Web resource configuration	Offline support on Android and Windows	Offline support on iOS
Web resource used on form handlers, form scripts, and ribbon commands	Supported (File names should be in lower case.)	Supported (File names should be in lower case.)
JavaScript files referenced within an HTML web resource	Supported	Not supported. Web resources may be available offline in some circumstances if they're retrieved while online and cached by the browser.
HTML, JS, CSS, XML web resources embedded on a model-driven app form	Supported	Not supported. Web resources may be available offline in some circumstances if they're retrieved while online and cached by the browser.
Other web resources embedded on a model-driven app form (like images, RESX, and so on) **Note**: The Dataverse file and image attributes aren't web resources but regular Dataverse data. To enable them offline, see Configure files and images in offline model-driven apps.	Not supported**	Not supported**
Webpage (HTML web resource) via a model-driven app's sitemap	Not supported	Not supported
Custom JS files that create a fetchXML dynamically	Not supported	Not supported

** Accessing RESX web resources as a dependency through Client API from within a JS file is supported.

> **Note**
>
> For the best compatibility across platforms, use one of the following recommendations instead of HTML web resources:
>
> - If the web resource is a user experience component, use a PCF control.
> - If the web resource doesn't present in the user interface, use form handler events which are available offline on all platforms.

- **Calculated and rollup fields**: Calculated and rollup fields that are part of rows synced to the client aren't reevaluated by the client. The reevaluation happens on the server when the updated row is synced.
- **Mapped fields**: When you run an app in offline mode, mapped fields aren't prepopulated when you create a new record from a table that has fields mapped to another table.
- **Lookup fields**: The value of a lookup field may not be updated in a form or grid when the display name of the lookup item is updated.
- **Filter operations**: Not all filter operations are supported in offline mode. For a full list of supported operators, see Supported Filter Operations Per Attribute Type in Mobile Offline using FetchXML.
- **Appointments**: Appointments can be created or updated in Dataverse, but they're not sent to recipients with server-side synchronization. For more information, see Appointment table/entity reference and Server-side synchronization tables.
- **Work order service tasks**: Work order service tasks created through *custom* code require msdyn_lineorder to be set to -1. An actual value will be automatically set after the next synchronization cycle.
- **Quick find search**: When you search offline, the results might be different from online searches. Characters in your search terms must match exactly when searching offline, while online equivalent results might be found.
- **Offline search**: Offline search is available only for offline tables. Users can only search one table at a time. Global search defaults to a categorized search in offline mode, even if Dataverse search is turned on and there is network connectivity, as Dataverse search isn't supported in offline mode. On grid pages, view-based search (filter by keyword) isn't supported in offline mode and grid search switches to a quick, find-based search.
- **Duplicate records**: The experience of detecting duplicates and merging them isn't supported in offline mode.

Profile filters limitations

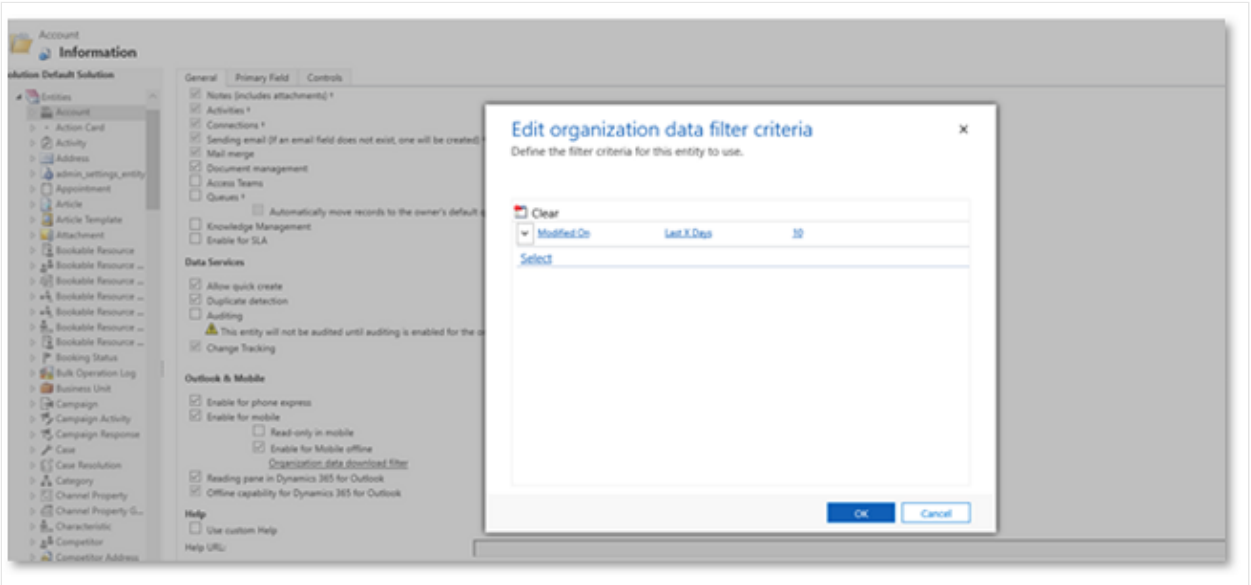
Expand table

Profile details	Limitation
Relationship defined for each table	A maximum of 15 relationships is allowed. There's also a maximum of one many to many (M:M) or one to many (1:M) relationships within those 15 relationships. If any custom tables demand this scenario, then revisit the data model. No circular references or self-references are supported.
Images and files	Images and files are subject to the same limitations as any other table. Because of implicitly defined relationships, an offline profile can only contain up to 14 image columns, across all entities.

Organization data filter

> **ⓘ Important**
>
> Organization data filters are no longer used. Changes to filters should be edited within the mobile profile itself.

It's recommended that you have at least one rule defined for all mobile offline-enabled data filters if you're using tables across profiles. By default, this value is set to last 10 days for most of the offline-enabled tables.



Related information

Troubleshoot offline sync errors in the Power Apps mobile app

Feedback

Was this page helpful? Yes No

Provide product feedback

Best practices for developing an app for offline use

Article • 06/04/2024

This article provides recommendations for how to create the best experience for users of apps configured for offline use.

- If you configure a canvas app for offline use, you need to optimize how the app fetches data from the data source. Learn more: How to create the best offline user experience in canvas apps
- To function offline, an app must initially download all necessary assets, including resources and data, to operate independently of a network connection. This process is a one-time setup task that occurs during the app's first use. The speed of this initial download is influenced by the volume of data downloaded to the device, which is determined by the offline profile settings and the quantity of data available to the user within the app's environment. For a seamless, initial experience, it's advisable to plan your offline app rollout.

How to create the best offline user experience in canvas apps

Use the following recommendations to create a fast user experience for specific scenarios.

Expand table

Scenario	Approach that's not recommended	Recommended approach
Show related tables information in a gallery.	Use a look-up to fetch the data.	Use a view containing the columns of the related table.
Create a filterable gallery with a lot of records.	Load the records from Dataverse and store them in a collection. Set the items of the gallery to the filtered collection.	Directly set the items of the gallery to the Dataverse-filtered data.
Update multiple records.	Loop the records and patch them individually.	Bulk patch a collection containing all the modifications.

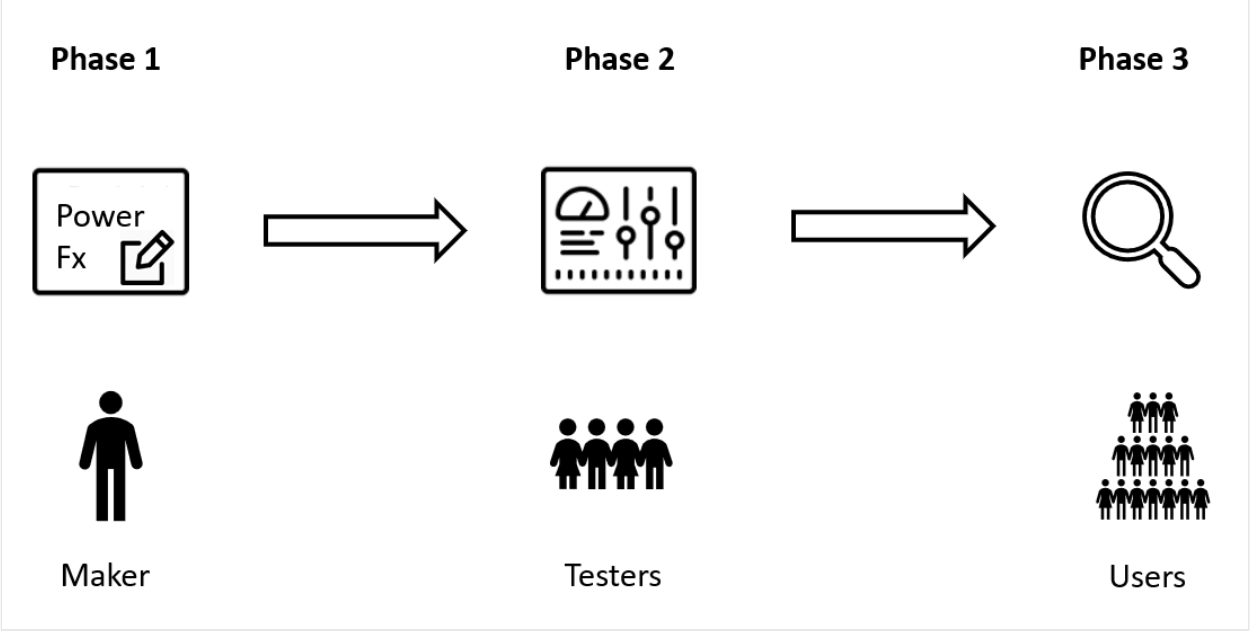
Diagnose mobile offline canvas apps with Monitor

Monitor is a tool that offers makers a deep view of what an app does and how it does it by logging all key activities that occur in the app as it runs. You can connect a mobile app session to Monitor to better diagnose and troubleshoot issues faster.

Plan your offline app rollout

> **Important**
>
> If you're using an **auto-generated offline profile** (available for canvas apps only), and if the first synchronization is taking too much time, you should create a manual, offline profile by using the following best practices.

Develop and roll out your offline app in three phases.



Phase 1: Develop and iterate

After you've set up mobile offline for canvas apps or set up mobile offline for model-driven apps, it's time to start testing and tweaking. Use Power Apps mobile or Field Service Mobile to determine how the app behaves when it's offline. For Windows, you'll find the Power Apps for Windows app in the Microsoft Store that allows iterating without the need for a mobile device.

In this phase, you'll add tables and apply filters to existing tables to make sure that the right data is downloaded to the app, following the guidelines to Optimize the offline profile.

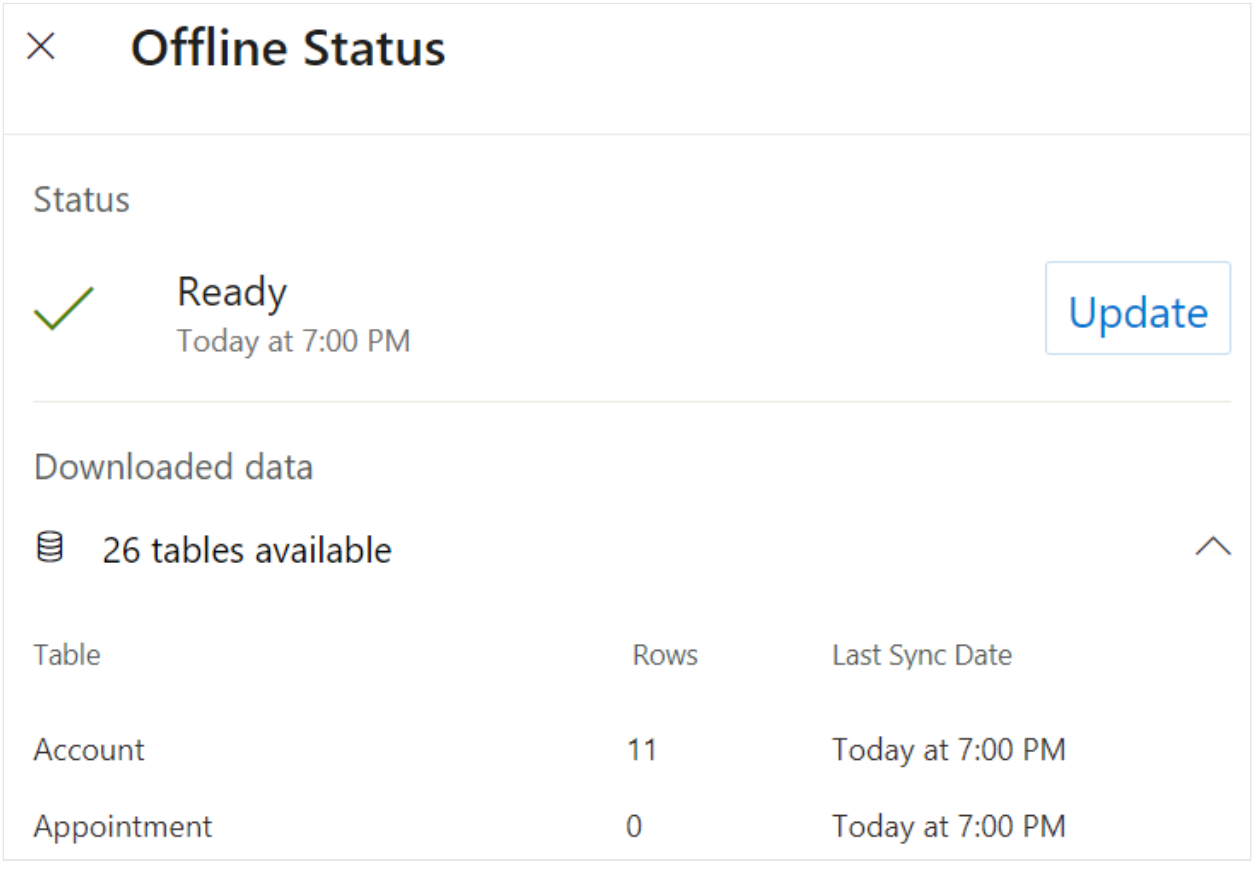
Outcome

You confirm that all the tables and forms work offline after the data is downloaded and that download sizes are reasonable.

> **Important**
>
> The metadata for the model-driven app is retrieved when the app starts. This means that if you change a component in your app, such as a form component or view, then you need to restart the app for the profile to reflect the changes.

Phase 2: Test with users

Ask a few users to test the app with real data. Make sure the offline profile scales for different types of users and works on devices with varying storage capacities. Check the **Device status** page (available out of the box in model-driven apps) for each user. For more information, see Using the Offline template and offline status icon. Adjust the filters in the offline profile to increase or decrease the amount of data that's downloaded.



Outcome

You confirm that the offline profile scales to real-use cases. If not, optimize the offline profile.

Phase 3: Roll it out

Deploy the app to the rest of your organization.

Outcome

You confirm that each class of user in the rollout is able to sync successfully and work offline.

Don't miss the data your users need

Test whether your users have all the data they need. Compare the data available when the app is online and when it's offline. With the device in airplane mode, make sure the views and forms show the same data as in a web browser online. If there are differences, either adjust the filters in your views or adjust the filters in your offline profile.

Add related tables if your app needs them

- **Business process flows**: If a form contains a business process flow, be sure to add the business process flow table. For more information, see Supported capabilities.
- **Files and images**: If your offline profile contains files and images, add tables for them. For more information, see Configure files and images in offline model-driven apps or Configure files and images in offline canvas apps. Use custom filters to limit the download of critical files.
- **Timeline**: To make notes on the timeline control (for model-driven apps only) available offline, add the **Notes** table and the **Users** table to the offline profile. Notes can be large if users upload images and videos, so apply custom filters to the **Notes** table to limit download times.

  > **Important**
  >
  > Data downloads may be slower if users upload files larger than 4 MB to the timeline control. If users need to upload files larger than 4 MB, use the quick notes control in Field Service or **Files/Images** instead of the timeline to improve performance.

Tips on mobile offline synchronization

- Mobile offline synchronization with mobile devices occurs periodically. A synchronization cycle could last for several minutes, depending on Azure network latency, the volume of data that's set for synchronization, and mobile network speed. Users can still use mobile apps during synchronization.
- The time for initial metadata download is determined by the number of total tables in offline-configured app modules. Make sure to configure only those tables and app modules for offline that are necessary to optimize the experience for end users.
- Ensure that any view that you want to work offline doesn't reference tables that aren't configured for offline use. For example, assuming **Account** is in the offline profile, then an **Account** view that references the primary contact when **Contact** isn't in the profile, isn't available.
- Changes to a user's security privileges are updated during the next synchronization cycle. Until that time, users can continue to access data according to their previous security privileges, but any changes they make are validated during the synchronization to the server. If they no longer have privileges to make changes for a row, they receive an error and the row won't be created, updated, or deleted.
- Any changes to a user's privilege to view a row won't take effect on the mobile device until the next synchronization cycle.
- Mobile offline honors the security model for mobile apps and the hierarchical security model, except the field-level security and field sharing.

See also

- Optimize the offline profile
- Configure canvas apps for offline
- Configure model-driven apps for offline
- Configure offline data for the Field Service (Dynamics 365) mobile app (contains video)
- Five tips for implementing the Field Service (Dynamics 365) mobile app (blog)

Feedback

Was this page helpful? Yes No

Provide product feedback

Optimize the offline profile

There's a lot to keep in mind when you create or update an offline profile for model-driven apps. An offline profile should include all the data that app users need to complete tasks in the field. If it includes too much data, however, app users might get stuck waiting for their data to download. They might even run out of space on their device. You need to consider the devices and data plans your app users have to make sure they have a great experience.

The following guidelines help you create an offline profile that meets the exact needs of your organization.

Don't make your users download too much data

Each user might have access to a different set of data. It's important to think about and test how much data different groups of users see. For example, a group sales manager might have access to many more sales opportunities than a local sales manager.

As you develop your offline profile and test with real or representative data, keep these best practices in mind:

- Limit the number of records to less than 200,000 for better performance. Exceeding 3,000,000 records isn't supported by the offline sync.
- Limit the number of tables to fewer than 100.
- Limit the total data size to less than 1 GB.
- Limit the total files and images size to less than 4 GB. Apply filters to reduce the total download size.
- Reduce the number of records downloaded in tables that change frequently by filtering on fields like status, group, region, owner, or assigned to.

If your app's offline data exceeds these recommendations, users see slower syncs, higher data usage, higher battery usage, and slower app performance.

Optimize your offline profile

Apply the following best practices to make sure that users download only the data they need. Optimizing the data that's downloaded makes it easier to stay within the recommended limits.

Don't reinvent the wheel

If you're customizing Field Service or Sales, start from their default offline profiles. You know core features work, and you won't miss tables that are used in standard forms.

The default offline profiles include everything you need for an out-of-the-box solution. You can add more tables that are important for your business.

However, **don't remove tables** from the default profile. Without those tables, forms or views might fail at runtime. If the default profile includes too many or too few rows of data, adjust the filters on the largest tables to optimize data sizes for your users.

Add all tables that are referred to in each form and view in your app

When you add a form or view to your model-driven app, look for references to other tables, including lookups. Make sure each of these tables is included in your offline profile with a corresponding related table or filters. Be sure to add all tables used in web resource scripts as well.

When you add a table to the offline profile, you can choose one of four options to determine which rows are downloaded:

- Organization rows
- All rows
- Related rows only
- Custom

To choose the best option for each table, consider which of the following categories your table belongs to:

- **Standalone tables**: Tables that are visible as grids in the app, like Contact.
- **Related tables**: Tables that are referred to in the form or grid view of a different table, like Unit.
- **Resource tables**: Tables that contain resource data, like Currency or Territory.

Choose a row option based on the category of table you add:

Expand table

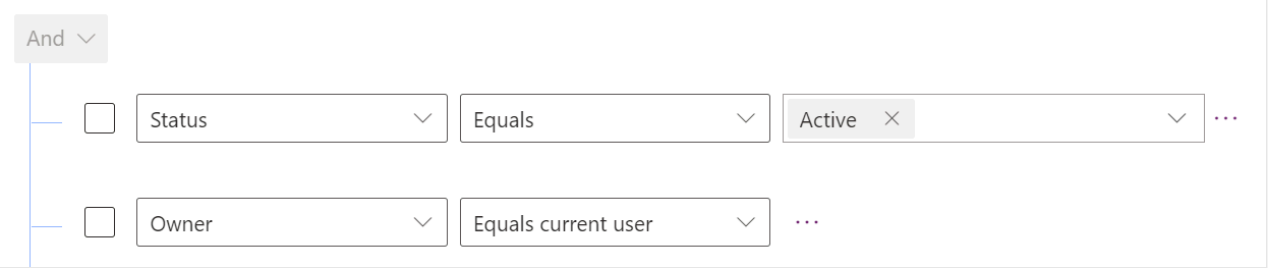
Table type	Organization rows	All rows	Related rows only	Custom
Standalone	✓			✓
Related			✓	✓
Resources		✓		✓

Use filters to reduce the data download size

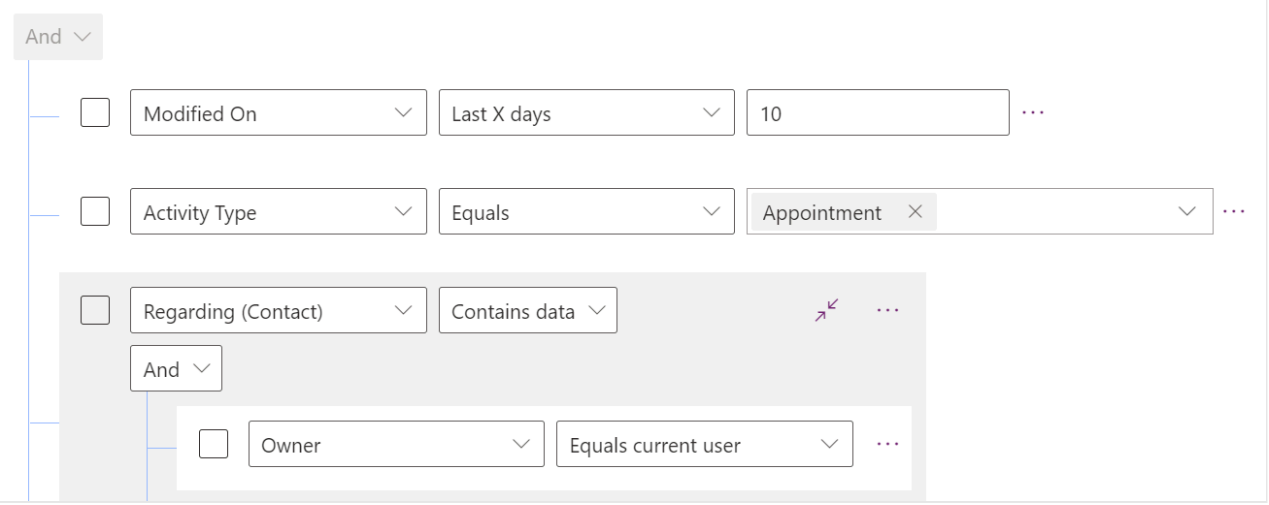
If users have access to a large set of data when they're online, apply filters to restrict the data that they download when they're offline.

> **Important**
>
> If you add a custom filter to a table that's set to download related rows, the filter is treated as an **OR**. That means that all related rows are downloaded, in addition to the rows specified by the filter. Users might download more data than you intend. If you want to download related rows and apply an additional filter, clear **Related rows only** and specify the relationship and the additional restrictions in a custom filter using **AND**.

- **Standalone tables**: Use a custom filter that includes the records that are required for the grid views in your app. By default, all views are included when you add a table in the app designer. To make sure that users have the same data online and offline, explicitly select the views that filter the data that you include in the offline profile.



- **Related tables**: Use a custom filter if you want users to download rows that are related **AND** that match your other filter criteria.

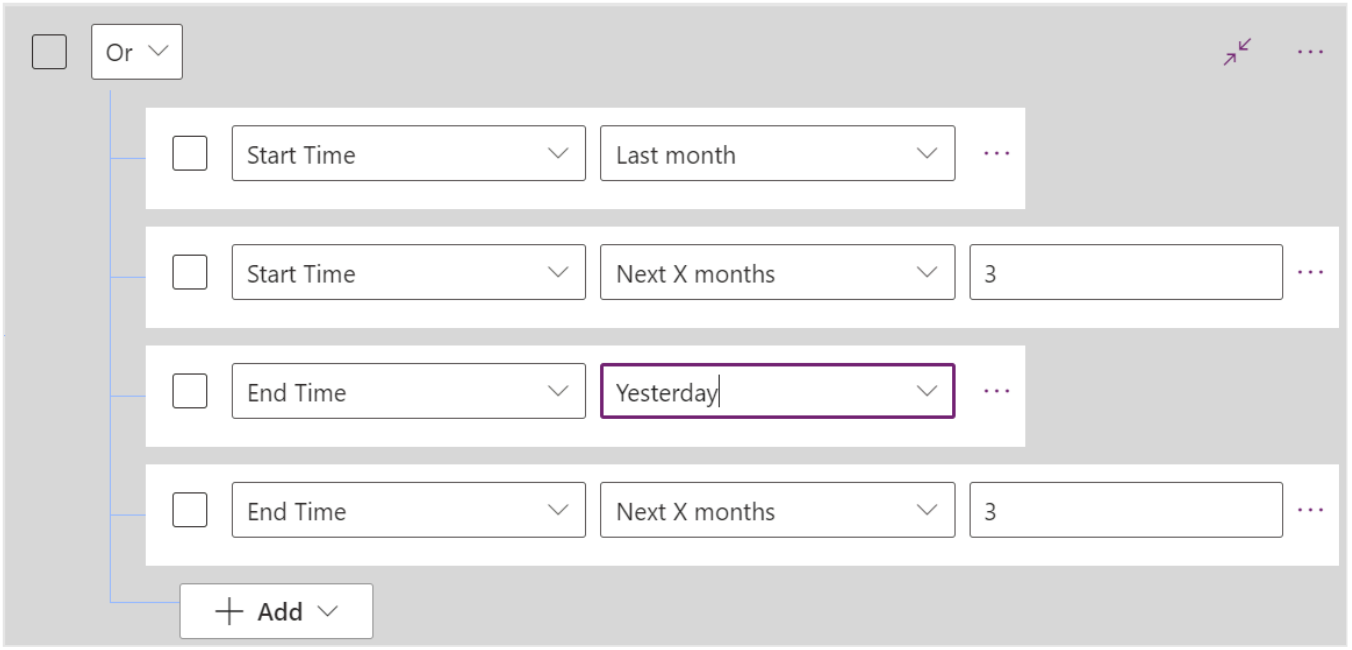


- **Resources tables**: Use a custom filter if you want users to download only rows that match your criteria, like rows with an Active status.



Common custom filters

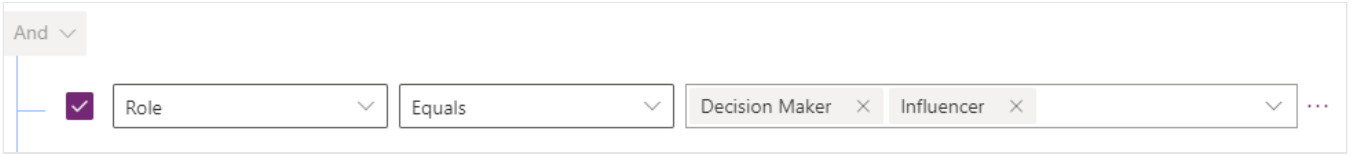
Filter by time and date fields for time-centric data like bookings and timeline items. Consider both future and past dates. For example, a common filter might include appointments from the past month and the next three months.



Filter by status to limit downloads to rows with a certain status.



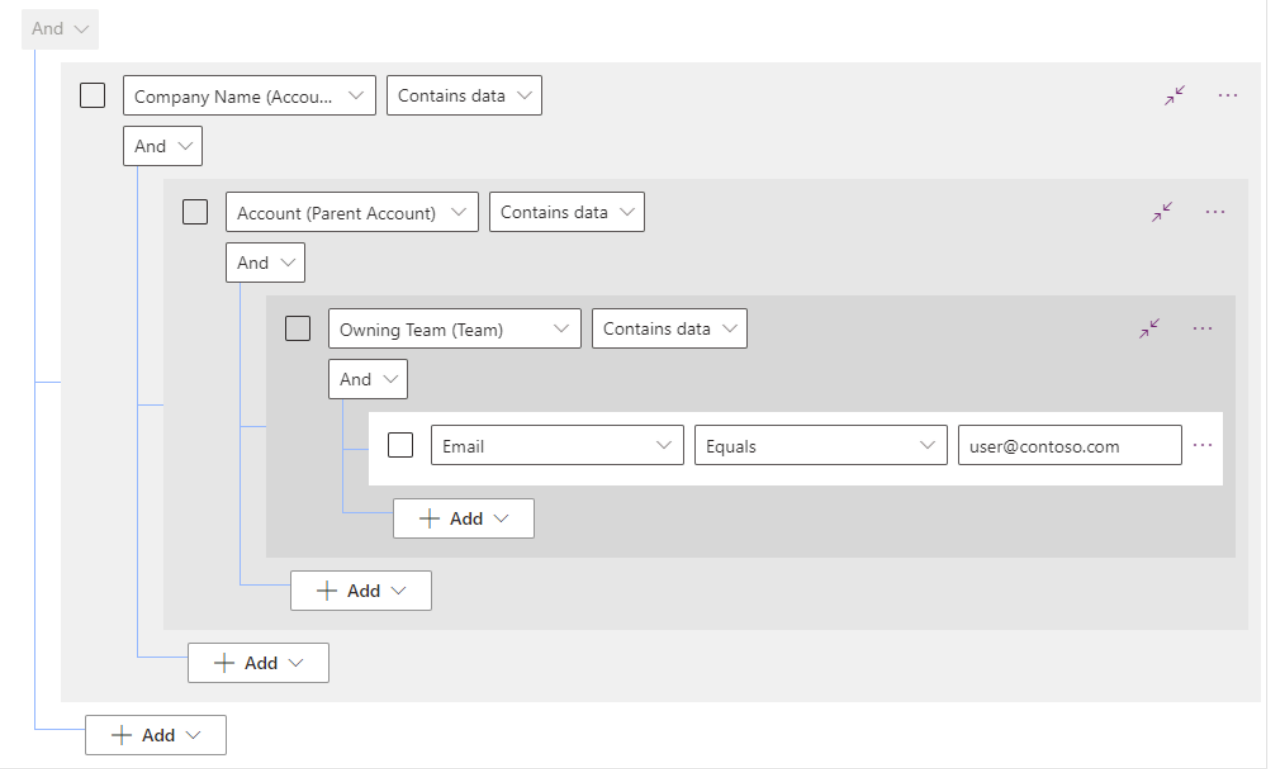
Filter by custom category or role fields to scope large tables down to the data needed for your app. For example, you could filter Contacts by Role to limit data to stakeholders.



Avoid these filter pitfalls that can slow down your downloads

If a custom filter results in a slow Dataverse query, downloads take longer. Follow these best practices to avoid common performance bottlenecks:

- Don't use partial string matches or **Contains**, **Begins with**, or **Ends with**.
- Avoid multiple levels of relationships in custom filters. Filters like this can lead to slow downloads:



- Avoid using many **OR** conditions.
- Avoid using small time windows to reduce data downloads on tables that change frequently. If a table in your offline profile is large and changes frequently, reduce the number of records downloaded by filtering on fields like status, group, region, owner, or assigned to.

Use column selection to reduce data download and sync time

The first sync can take time for organizations with large datasets, especially in remote areas. Optimize this by explicitly selecting only the required columns to download on the device and avoid downloading columns that are never used in the app. This saves both network and disk usage and results in faster sync time. While this impacts both the first sync and delta syncs, the first sync sees a greater impact as there are more records to download.

1. In Power Apps Studio, in the left side panel, select **Apps**.
2. Select your model-driven app, and then select **Edit**.

3. Select **Settings**.
4. Select **General**.
5. In the **Select offline mode and profile** section, select **Edit selected profile** from the **More actions (...)** icon for the selected profile.
6. In the **Edit Profile** pane, under the **Data for offline use** section, identify the table to optimize. Select the **More actions** (...) icon, and then select **Edit**.
7. The **Edit Table Name** page appears. Under the **Selected columns** section, select the **Manage columns** option.
8. In this **Managed selected columns** pane, select the **Select only required columns** option.
9. Columns used in the app are automatically selected. For all other columns, optimize by selecting only those that are necessary.

   > **Important**
   >
   > - If you have custom JavaScript code in the app, ensure the columns used in the custom code are selected in the offline profile.
   > - If you add additional columns after optimizing the offline profile, go back and select that new column as **Required Column**, in the **Managed selected columns** pane.

10. Select **Back**.
11. Select **Save**.
12. Select **Save + close**.
13. Close the **Settings** and select **Publish**.

> **Note**
>
> - We recommend always selecting required columns for tables with a high number of unused columns.
> - The **Manage columns and Managed selected columns** option isn't available in the mobile **Offline Profile configuration** on **Power Platform admin center**.

Optimize for canvas apps

> **Important**
>
> The optimization isn't supported on canvas apps using a selected offline profile.

For canvas apps using an **autogenerated** offline profile, the system automatically optimizes the columns downloaded on the device. With the optimization activated, the app only downloads the required columns.

1. In Power Apps Studio, in the left side panel, select **Apps**.
2. Select your canvas app, and then select **Edit**.
3. Select **Settings**.
4. Select **Updates** and **Experimental**.
5. Turn on the **Faster offline synchronization** option.
6. Close the settings and select **Publish**.

Related information

- Best practices for developing an app for offline use
- Mobile offline for model-driven apps overview
- Best practices and limitations (contains video)
- Five tips for implementing the Field Service (Dynamics 365) mobile app (blog)

Last updated on 01/08/2026

Resolve sync conflicts with the server

Article • 06/14/2024

This article provides information to help you resolve mobile synchronization conflicts with the server.

Sync Error table

In the event of a synchronization error, the system automatically generates a new entry in the **Sync Error** table within Dataverse. This table is structured with the following columns.

Expand table

Column name	Description	Example
Error Time	Time when the error was created.	5/17/2024 7:16 AM
Error Message	Description of the error.	`Entity Account With Id = <RowId> Does Not Exist`
Error Type	Type of the error, such as a conflict or record not found error.	Record not found
Record	Updated record in sync error.	Contoso
Owner	User who updated the record.	John Doe
Action	Action applied to the record to sync.	Update
Action Data	Json containing the payload of the action.	{"lastname":"Contoso2"}
Request Data	Json containing the query.	`{"lastname":"Contoso2","syncerror__regardingentityid": <syncerrorId>,"entitylogicalname":"account"}`
Error Code	Code of the error.	-2147015424

> **Note**
>
> Make sure that the role assigned to the user grants read privileges on the **Sync Error** table. Learn more: Create or edit a security role to manage access

Recommendation on how to use the Sync Error table

The **Sync Error** table is a standard Dataverse table. We recommend that you create a model-driven app to manage the sync errors.

If you run the model-driven app, you can see the sync errors by each user. If you select a sync error row, you can **Retry changes** from the command bar.

Sync error in the Device status page

Sync errors natively show up in the **Device status** page. The **Device status** page is available out of the box in model-driven apps, but must be set up in the canvas app. See Using the Offline template and offline status icon for more information.

Use a cloud flow to take actions on sync errors

You can create an automated cloud flow in Power Automate using the Dataverse trigger **when a row is added, modified, or deleted**. The flow can automatically send an email or send a notification on the device.

> **Note**
>
> To retrieve the user's email address within the flow, add the Dataverse action **Get a row by ID** using the **Owner** column of the **Sync Error** row.

Sync conflict

When there's a mismatch of data between client and server, conflict errors occur. By default, changes that are made by a user in offline mode are automatically synced to the server when the user is back online.

How sync conflicts are resolved

When a user makes changes to data in an offline app, updates to each column are pushed back to Dataverse as soon as the network is available. The last update to each column is stored in Dataverse, so this sync doesn't fail due to conflicting changes.

Server-side plug-ins and validation can invalidate changes. Those changes are reverted locally, and an error is written to the **Sync Errors** Dataverse table.

Conflict resolution settings

> **Important**
>
> The conflict resolution settings don't apply to canvas apps.

If updates made while offline are being rejected too frequently, consider changing the conflict resolution setting to its default value.

1. Sign in to the Power Platform admin center.
2. Select **Environments** in the navigation pane.
3. Select an environment.
4. Select **Settings** in the command bar.
5. Select **Product** > **Features**.
6. In the **Advanced mobile offline settings for model-driven apps** section, turn off the **Enable conflict detection for mobile offline synchronization** option.

> **Note**
>
> When this option is turned on (not recommended), updates on the server may prevent client updates being applied. Conflict errors must be resolved manually.

Feedback

Was this page helpful? Yes No

Provide product feedback

Set an app as a startup app

Article • 03/22/2024

Users can set an app as a startup app in Power Apps mobile app. This feature is available for the iOS and Android versions of Power Apps mobile app.

Users can mark any canvas or model-driven app as a startup app in two ways:

- From an in-app suggestion
- On the app **Details** page

Benefits of setting a startup app

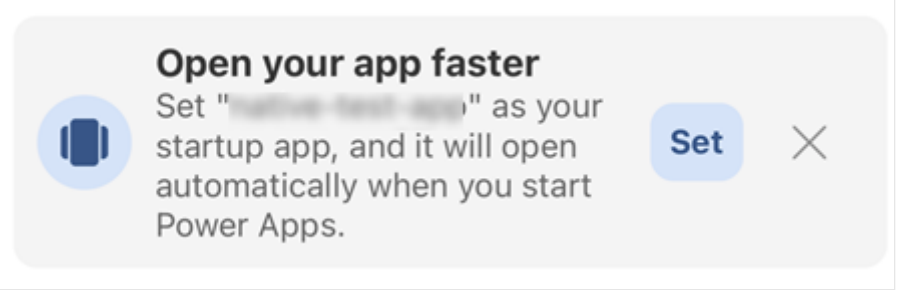
- Improves productivity:
  - Once an app is set as the startup app, it opens by default when users open Power Apps mobile app.
  - No need to wait for the home page to load, saving the user time.
  - Creating and testing an app is faster, since you can bypass the home page.
- Efficient to create:
  - You can set a startup app with one tap.
  - This feature is better than shortcut apps, which require three to four taps and need external apps/device feature support.
- Simple to enable:
  - No changes are required for makers to use this feature.
  - There's in-app information about the feature's use and benefits.
  - The feature is compatible with offline, wrap, custom branding, push notifications, and shortcut features.

Set a startup app from a suggestion

If a user opens the same app for seven days, a suggestion appears, at the top of the home page, when they close the app.

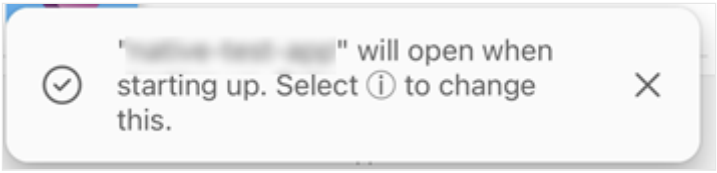
1. View the startup app suggestion.

   You might see an in-app suggestion, asking users if they want to open the app as a Power Apps mobile startup app.



2. Set the start up app.

   Tap on **Set** to set the app as the startup. The user sees a success message that confirms the app is set to open as the startup app.



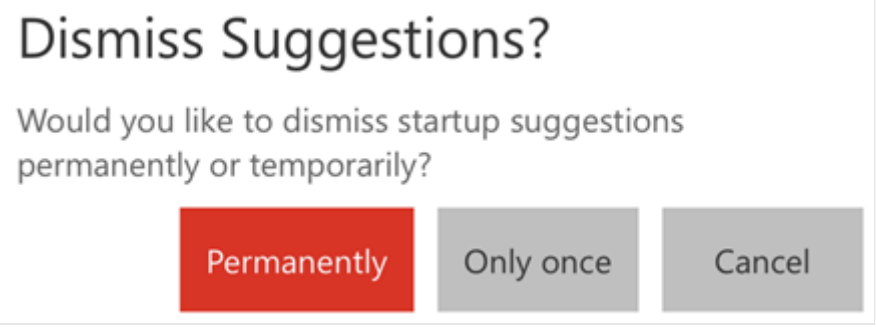
3. Confirm success with the startup app icon.

   If an app is successfully marked as the startup app, an icon appears next to the app title on the home page.

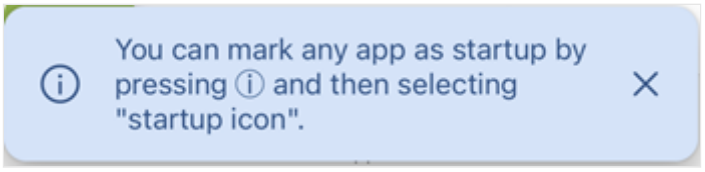


If the suggestion is dismissed

- If a user closes the suggestion, they're asked if they want to receive the suggestion in the future. Canceling the suggestion or tapping outside of it closes the suggestion dialogue.



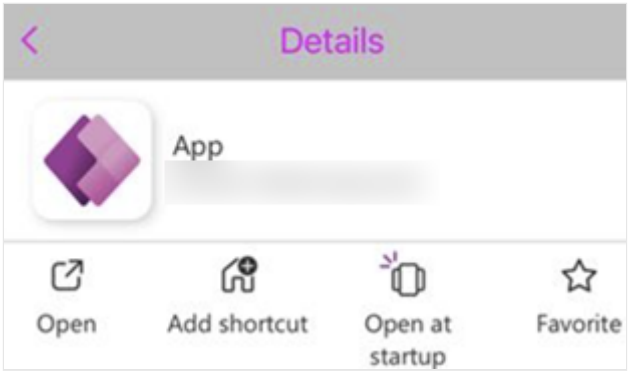
- If users close the suggestion permanently, they can still set a startup app later from the app **Details** page.



Set a startup app from the Details page

To set a startup app from the **Details** page, follow these steps.

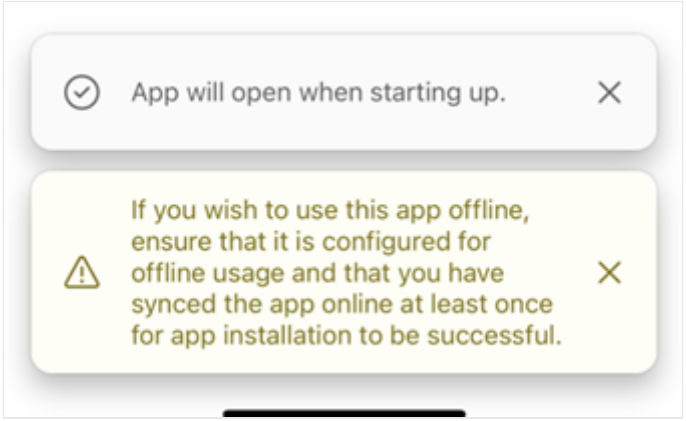
1. Tap the **Open at startup** icon on the app **Details** page.

   After you tap the icon, the app is marked as a startup app.

2. View or dismiss the usage message for offline scenarios.

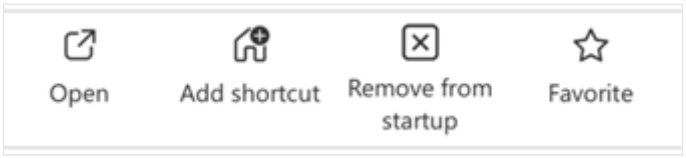
   This usage message disappears in 10 seconds if not canceled by users.

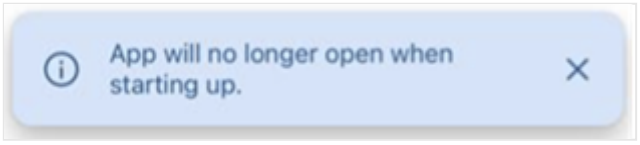
Reset a startup app

If an app is already marked as the startup app, the app **Details** page shows a **Remove from startup** icon.

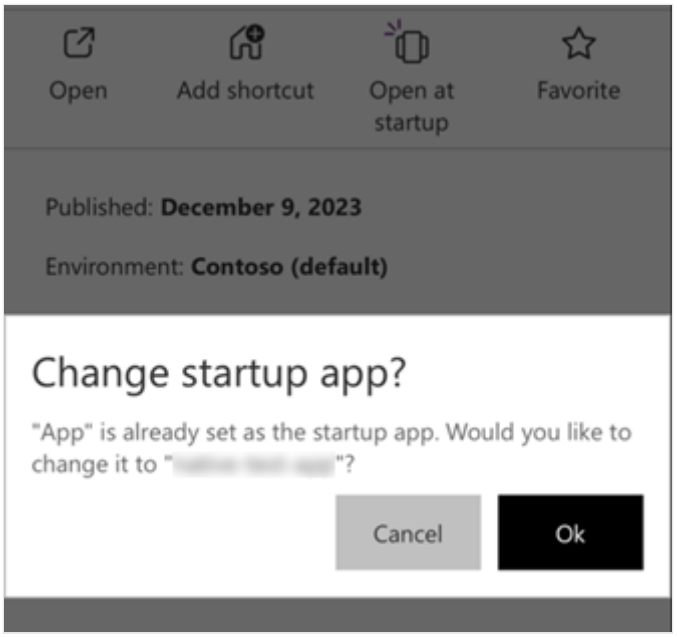
1. Tap this icon to unmark the app as the startup app.



You see a message that the app no longer opens by default when a user taps on the Power Apps mobile page.



2. Tap **Ok** to replace the existing startup app with another app. The existing app is now unmarked and the new app is marked as the startup app.



Known limitations

- Only canvas and model driven apps are eligible for this feature.
- This feature works only on iOS and Android devices.
- Users can set only one app as the startup app.
- Only end users can set a startup app.

Feedback

Was this page helpful? Yes No

Provide product feedback

Troubleshoot issues in the Power Apps mobile app

Article • 12/19/2024

This troubleshooting article helps fix common issues for the Power Apps mobile app.

Resolve canvas apps performance issues

Learn about most common performance issues that can make canvas apps run slow and how to improve performance. More information: Common canvas app performance issues and resolutions.

Diagnose mobile apps with Monitor

Monitor is a tool that offers makers a deep view of what an app does and how it does it by logging all key activities that occur in the app as it runs. You can connect a mobile app session to Monitor to better diagnose and troubleshoot issues faster.

Debug JavaScript web resources in mobile apps

While developing JavaScript web resources for mobile scenarios, you can use your Android device to debug your mobile-specific code and ensure it works as expected. More information: Debug JavaScript in mobile apps on Android.

Error: There was a problem signing you in

You're unable to sign in due to issues with the Microsoft Authenticator app.

If you don't have the Microsoft Authenticator app, download the app from the App Store or Play Store and then sign in to Power Apps mobile again.

If you already have the Microsoft Authenticator app installed and you're having sign in issues, then try these steps:

1. Back up your Microsoft Authenticator account. For more information, see Back up and recover account credentials using the Microsoft Authenticator app
2. Uninstall the Microsoft Authenticator app.
3. Uninstall Power Apps mobile.
4. Reinstall the Microsoft Authenticator app and add your backup account.

5. Reinstall Power Apps mobile.
6. Open Power Apps mobile and then sign in.

Error: Your device configuration is preventing sign in

If you get this error messages, it means your company's IT department requires Microsoft Intune or an authentication app to sign in securely. However, your device setup is blocking Power Apps mobile from launching the authentication app that's installed on your device.

Microsoft authentication apps are Authenticator and Company Portal. Your company might use a non-Microsoft authentication app. If you're not sure, ask your IT department or help desk which authentication app to use. Once you have the correct authentication app, then follow the steps below.

> ⓘ **Note**
>
> Power Apps requires a valid license to sign in. For more information, see Licensing overview.

Sometimes, updating and manually opening the authentication app on your device before signing in to Power Apps mobile can fix the problem. If this doesn't fix the issue, then follow the next steps depending on your device manufacturer and authentication app.

Huawei or Honor device

1. Go to **Settings** > **Battery** > **App launch**.

   > ⓘ **Note**
   >
   > The **App launch** menu can have different names depending on the model and the operating version of your mobile device. If you don't see the **App launch** menu option, then look for one of the following menu options:
   >
   > - **Close apps after screen lock**
   > - **Applications**
   > - **Background applications**

2. Under **Manage automatically**, on the authenticator app set the toggle switch to **OFF**.

3. On the **Manage manually** screen, ensure that **Secondary launch / Can be launched by other apps** is enabled. To allow the Power Apps mobile app to launch the app.

Vivo device

1. Go to **Settings** > **More Settings** > **Applications** > **Autostart**.
2. Set the toggle switch to **ON** for the authenticator app.

If the issue is still not fixed, then try these steps:

1. Back up your Microsoft Authenticator accounts. For more info, see Back up and recover account credentials using the Microsoft Authenticator app
2. Uninstall the Microsoft Authenticator app.
3. Uninstall Power Apps mobile.
4. Install Microsoft Authenticator again and add your backup accounts again.
5. Install Power Apps mobile.
6. Open Power Apps mobile and sign in.

App list is empty

The app list in the Power Apps mobile app might appear empty when you lose internet connection before the app list has downloaded. This can happen in any of the following scenarios:

- It's the first time you're signing in to the mobile app.
- When you pull down to refresh the app list.
- When you come back online after a period of being offline. The app list is automatically refreshed.

To resolve connection related issues, ensure you remain connected to the internet while the app list is fully downloaded.

Pin to Home does not work on iOS 14

iOS device running iOS 14: The Safari browser no longer supports the **Pin to Home** functionality for Power Apps mobile. You need to use the Siri Shortcuts app to pin an app to the Home screen. For more information, see Use Siri Shortcuts (iOS 14 or later).

iOS 13: You can still use the Safari browser to pin an app to the Home screen. For more information, see Pin an app to the home screen

App resets when running it on Power Apps mobile

When you run a canvas or model-driven app on Power Apps mobile, it can reset if the app is using too many resources. If the app uses more resources than are available on your device, the app resets. This is similar to when you visit a large, complex webpage and the web browser suspends the page because it's consuming too much power.

On Android devices, this app restart can look like a crash because the app is closed and the user is taken to the home screen of the device.

If you experience a reset while using a canvas app, contact your app developer, and see Prevent canvas app restarts.

Unable to download SharePoint attachment in the mobile app

The Power Apps platform doesn't support accessing authenticated URLs, including SharePoint attachments. If you run a Power Apps application in a web browser and it tries to access a SharePoint attachment, it might work if you're signed-in to SharePoint in another tab. This is because web browsers support multiple signed-in users, and sign-ins are valid across browser tabs. However, the Power Apps mobile app isn't a web browser, so doesn't benefit from this browser-based behavior.

Flows created in a solution aren't supported on Power Apps mobile

The Flow action menu in Power Apps mobile doesn't support flows created in a solution.

Error: Contact your administrator for access to your organizations mobile apps

When using the Dynamics 365 mobile app, you encounter the following message: **Contact your administrator for access to your organizations mobile apps**.

To check for recently added apps, select Refresh. If you can’t find your app, change your search criteria and try again.**

For more information go to, Troubleshoot "We can’t find any apps for your role" error message.

The list of apps is empty

Make sure the user has a default security role assigned to them such as **Basic**. This is in addition to any custom security role assigned to the user. For more information, see Setup overview for mobile apps.

Error message: This record is unavailable

If this message appears when a user starts the mobile app, and then selects the **Home** button, or **Dashboards** from the menu, the user likely doesn’t have access to the expected dashboards.

Flows created in a solution isn't supported

The Flow action menu in Dynamics 365 for phones and tablets app doesn't support flows created in a solution.

Error message: Your server isn't available or does not support this application

Cause 1: The Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises) server is down. Verify that the server is on and connected to your network.

Cause 2: Your Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises) app version isn't supported. For supported version information, see What's supported.

Cause 3: This error can also occur if you enter an invalid URL. Make sure the same URL you have provided works to access Dynamics 365 apps in your browser on your device.

Error message: You haven't been authorized to use this app. Check with your system

administrator to update your settings

Cause 1: Verify that your security role includes the **Use Dynamics 365 for tablets** privilege. See "Required privileges" in Setup overview for mobile apps.

Cause 2: This error can occur if you have a Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises) organization and your user hasn't been assigned a license for the organization. If you add a Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises) subscription to an existing Office 365 tenant, your user might not have a license assigned.

If the user has the Service Administrator role and you're able to sign in to on the web app to perform certain administrative actions, but you can't perform end user tasks, such as creating records (for example, accounts, contacts, and leads) or configuring Dynamics 365 for mobile. When you sign in to the web app, you might notice that not all areas appear within the navigation (for example, Sales and Marketing are missing):

Access the **Active** users section in the admin center and verify you have a **Dynamics 365 Customer Engagement Plan** license assigned to your user record.

Error message: Sorry, something went wrong while initializing the app. Please try again, or restart the app

Cause 1: Permissions might not be set properly. See "Required privileges" in Setup overview for mobile apps.

Cause 2: See the following KB article:

An error occurs in the Dynamics 365 app for users in child business units. For more information, see Sorry, something went wrong while initializing the app.

Sample Trace Message for Cause 2:

```
Error Message:System.NullReferenceException: Object reference not set to an instance of an object.
Microsoft.Crm.Application.WebServices.ApplicationMetadataService.<>c__DisplayClass30.<UserRolesChanged>b__2d(Entity role)
at System.Linq.Enumerable.Any[TSource](IEnumerable1 source, Func2 predicate)
at Microsoft.Crm.Application.WebServices.ApplicationMetadataService.UserRolesChanged(G
```

```
uid[] clientUserRoles, DateTime syncTime, ExecutionContext context)
at
Microsoft.Crm.Application.WebServices.ApplicationMetadataService.RetrieveUserContex
t(UserContextRetrieveRequest userContextRetrieveRequest)
```

Error message: The language installed on your company's system isn't available on the app. Please contact your system administrator to set up a supported language

Cause: This error occurs if one of the supported languages isn't enabled in Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises). For more information on the supported languages, see Supported languages for Dynamics 365 for phones and Dynamics 365 for tablets.

Error message: The process assigned to this record is unavailable or has been deleted

If you receive this message for a record which has a nondeleted process assigned to it, you should manually synchronize Dynamics 365 mobile app data with Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises) app data. Close the mobile app, reopen it, and then choose to download the latest customizations. This procedure forces the mobile app to check for updated customizations. Recently viewed data while you were connected is cached and synched. Record data like Accounts or Contacts aren't synched.

Event 10001 messages appear in the Event Log when you run Dynamics 365 for mobile.

The following event may be recorded multiple times to the Event Log, when **Show Analytic and Debug Logs** is enabled, on the device where Dynamics 365 for mobile is running. Notice that, by default, **Show Analytic and Debug Logs** is disabled in Event Viewer and these messages won't be recorded. More information: Enable Analytic and Debug Logs

- Event ID: 10001

- Message: `SEC7131 : Security of a sandboxed iframe is potentially compromised by allowing script and same origin access.`

  Verify the source of the messages. If the source is Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises), these events don't pose a security threat and can be ignored.

Data cached for offline viewing remains after the entity is no longer enabled for Dynamics 365 for mobile

In Dynamics 365 for mobile, record data is cached as the user visits the record so the user can access the data when going offline.

This cached data persists after the entity is no longer enabled for Dynamics 365 for mobile (**Settings** > **Customizations** > **Customize the System** > [select an entity] > under **Outlook & Mobile**, deselect **Dynamics 365 for tablets**).

To remove the cached data, the user must sign out of mobile app, or the app must be reconfigured or uninstalled.

Customization changes don't appear in the mobile app

Cause 1: The customizations (metadata) from Microsoft Dataverse or Dynamics 365 Customer Engagement (on-premises) organization are cached on your device. The app checks for updated metadata after 24 hours or anytime you reopen the app. For customization changes to become available immediately, you must completely close and then reopen the app. If new metadata is found, you're prompted to download it. For more information on how to completely close an app, see the help for your operating system or reference one of the articles provided:

- **Windows 10**: How do I close an app?
- **iPad**: Force an app to close
- **Android**: How to force close Android apps

Cause 2: You might be seeing a different form than the one you customized. If you have multiple forms for an entity, you see the first form in the form order that you have

access to. This is different than the web application where you see the last form you used and have the ability to change between forms.

Native Android or iOS links aren't supported

The Dynamics 365 mobile app doesn't support universal links on iOS and Android app links on Android.

Error message: in Android, AUTH_FAILED_INTUNE_POLICY_REQUIRED AADSTS53005: Application needs to enforce Intune protection policies

Contact your administrator or help desk and make sure that your user is under Intune app protection policies.

On your device, go to **Settings** > **General** > **Accounts** and remove any accounts that are connected with Microsoft 365.

Error message: Access Denied. This app must be protected with an Intune policy before you can access company data

Contact your administrator or help desk and verify that Intune app protection policies are applied for your user account.

Admin instructions to resolve the issue

If you're an admin, you can avoid users getting this error by making sure all mobile users have access to the sales dashboard:

1. In the web app, go to **Settings** > **Customizations** > **Customize the System**.
2. Select **Dashboards**.
3. Select **Sales Dashboard**.
4. Select **Enable Security Roles**.

5. Select **Display to everyone** and then select **OK**. If you prefer to display only to select security roles, be sure to select your user's security role.
6. Select **Publish**.
7. Have your user close and open the mobile app so your dashboard changes download.

End-user instructions to resolve the issue

You can choose a different dashboard and set it as your home page:

1. From the mobile app, select the menu and then select **Dashboards**.
2. On the command bar, select **Select Dashboard** and then select the dashboard you would like to use as your home page.
3. On the command bar, select **Set as Home**.

Or, you can create a personal dashboard through the web app:

1. In the web app, go to **Sales** > **Dashboards**.
2. Select **New**.
3. Select **Properties**.
4. Add the components you want on your dashboard and select **Save**.
5. In the mobile app, follow the previous procedure to select your new dashboard and set it as your home page.

Network requests fail when Power Apps mobile app is running in the background

When the Power Apps mobile app is running in the background and a canvas or model-driven app makes a network request, a mobile operating system could deprioritize or cancel this network request. This can cause an error message to appear in the mobile app when it returns from the background.

If you experience a failed network request when the Power Apps mobile app is running in the background, contact your app developer.

App doesn't appear in the app list offline

New and recently republished apps might not appear in the offline app list right away. To make your app appear in the app list offline, open the app on your device when it's online and keep it open for 1-2 minutes.

Not able to zoom in to input elements in the app

Pinch to zoom isn't supported by HTML input elements by default. More information on HTML input control default behavior: HTML Input element.

Issue still not resolved?

If the information provided previously doesn't resolve your issue, either Post your issue in the Power App Community or Contact Technical Support.

The following are some suggested details to provide:

- What are the specific symptoms you encounter? For example, if you encounter an error, what is the exact error message?
- Does the issue only occur for users with certain Dynamics 365 security roles?
- Does the issue only occur on certain devices but works correctly for the same user on another device?
- If you attempt to connect to a different Dynamics 365 organization that doesn't include your customizations, does the same issue occur? If the issue only occurs with your customizations, provide a copy of the customizations if possible.
- Does the issue still occur after uninstalling the app and reinstalling it?
- What type of device are you using, such as iPad 4th Generation, Microsoft Surface. What is the version of the operating system, such as iOS 10.0 or Windows 10.

See also

Setup overview for mobile apps

Power Pages known issues

Power Automate known issues

Feedback

Was this page helpful? Yes No

Provide product feedback

Prevent canvas app restarts in the Power Apps mobile app

When you run a canvas app in the Power Apps mobile app, it can be restarted for the following reasons:

- The app is using too much total memory.
- The app is using too much memory or processing at one time.
- The app is moved to background – either when your specific app uses a native control (like **Add Picture** or **Barcode Scanner**) or the Power Apps mobile app is put in the background by the user.

This memory and processing limitations, which is especially strict when the Power Apps mobile app runs in the background, is imposed by the operating system (Android or iOS). If the app uses more resources than are available on your device, the app will reset. This is similar to when you visit a large complex webpage, the web browser suspends the page if it's consuming too much power.

On Android devices, this app restart can look like a crash because the app is closed and the user is taken to the home screen of the device.

Use this article to learn how to prevent canvas app restarts on Power Apps mobile.

Prevention tips for end users

While the biggest improvements are made by app makers, here are some tips that end users can use to decrease the occurrence of app restarts:

- In the Power Apps mobile app, on the user profile page, select **Clear cache** to remove saved data. This should only be done when the end user has data as clearing the cache will remove any **SaveData** or app caches.
- On Android devices, ensure the Power Apps mobile app is prioritized. Refer to your device manual to keep the app running through the settings like battery optimization and app sleep configuration.
- Close any running apps, other than the Power Apps mobile app, and remove any unneeded data from the device such as unused apps or images.
- While it's cumbersome, consider using the app slower – pause slightly between screen navigation or after taking pictures or saving or loading data.

Prevention tips for app makers

The greatest memory improvements to decrease app restarts can be made by app makers. When creating an app, remember to optimize the app to run on the lowest-specification device that your end users will use and follow best practices around resource usage.

> **Note**
>
> Since app restarts are caused by the operating system restricting the amount of memory and processing an app can do, it might take a combination of the mitigations suggested below to minimize app restarts. The more memory and processing that can be reduced, both in total and at one time, the less likely an app restart will be.

> **Tip**
>
> You can connect a mobile app session to Monitor to see where your app is doing heavy processing or memory-intensive operations. Monitor is a tool that offers makers a deep view of what an app does and how it does it by logging all key activities that occur in the app as it runs.

Expand table

Problem	Root Cause	Mitigation
App restarts when loading a specific screen.	If a specific screen is running too many formulas at one time, the app can have too much processing and be restarted by the operating system.	Limit the number of controls on a screen and the number of formulas being run when the screen is loaded, which might require splitting up screens. Avoid control dependency between screens. Prevent the user from taking action during high-processing moments. For example, when a large gallery with multiple filters and lookups is loading. Consider using named formulas. Consider using the **Delayed Load** app setting.
App restarts after multiple screen navigation.	There are memory leaks happening on the screens, resulting in memory usage	Turn on the **Keep recently visited screens in memory** app setting. While this increases total memory usage in

Problem	Root Cause	Mitigation
	building up as the user navigates around the app.	the short term as the screen is preserved in memory, across multiple screen navigation, it will prevent memory leaks.
App restarts when operating in offline mode using the **SaveData/LoadData** method	Offline scenarios that use **SaveData/LoadData** or large data connections can consume significant memory, especially when media content is involved.	Use the new offline profiles feature to manage offline data more efficiently. Avoid saving large media files locally. Optimize data sync strategies, and use explicit column selection to reduce payload size. Turn off experimental features like record scope one-to-many relationships if you don't need them.
App restarts when using the camera control.	The camera control saves the captured image to memory. This can increase the memory usage of the app, especially if the captured image is saved into variables or **SaveData** using the **OnStream** property.	Don't save any images beyond the most recently captured one. Use **Photo** instead of **Stream** to only capture images when the user taps on the camera. If you need to use **Stream**, set the **StreamRate** property as high as possible to reduce the number of updates.
App restarts when using the **Add Picture** control.	The **Add Picture** control saves the selected media content into memory. Due to the size of media content, this can result in significant memory pressure.	Consider changing where the media content is captured. Consider turning off the **UseMobileCamera** property so users can only select media from gallery/photo library. Consider using the attachment control (as part of a form) to only select files instead of capturing them using the camera. Consider switching to the camera control.
App restarts when using native controls like **Add Picture**, **Barcode Scanner/Reader**, **View in**	Native controls can put the app in the background where the operating system has stricter limitations on memory before restarting the app.	Make sure all memory and processing intensive operations are completed before using these controls. For example, disable the **Add Picture** control until any **SaveData** operations

Problem	Root Cause	Mitigation
MR, **Measuring camera**, and others.		are completed. Consider switching to a different control like **Camera** instead of **Add Picture**.
App restarts happen for a subset of devices.	Devices have different amounts of memory and processing available. As an example, older devices are more prone to app restarts due to their lower available processing. Having other apps running can also reduce the amount of processing available. Similarly, other apps installed and lots of photos/media saved to the device can take up memory, reducing what is available for the app.	Refer your users to the tips listed before in our Prevention tips for end users section. Consider investing in devices that meet the performance needs of your app.
App restarts continue to happen.	Memory usage is affected by how the canvas app is authored. Make sure your app follows best practices can help ensure performance, reduce memory usage, and processing.	Optimize your app following the best practices below.

Best practices for building performant apps

- Build large and complex apps
- Troubleshoot Power Apps canvas app performance issues
- Tips and best practices to improve performance of canvas apps
- Power Apps coding guidelines - Code optimization

Last updated on 01/09/2026

Install Power Apps for Windows

Install Power Apps for Windows to run model-driven app or canvas app on your Windows device. For more information on Power Apps, see What is Power Apps.

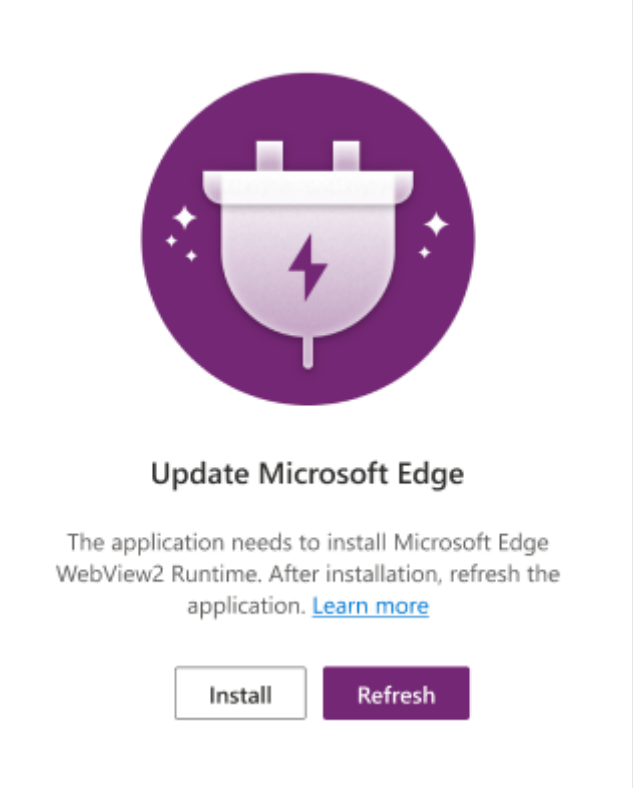
Install the app

1. Go to the Microsoft Store and install Power Apps for Windows.
2. When the app is installed, open it and sign in.

> **Important**
>
> If you run into issues during the installation or the app closes without warning, follow the steps in the next section to install the latest version of WebView2 Runtime, and retry installing Power Apps for Windows.

Webview2 Runtime

If you get an **Update Microsoft Edge** error message when you're installing Power Apps for Windows, this means you don't have WebView2 Runtime on your device.



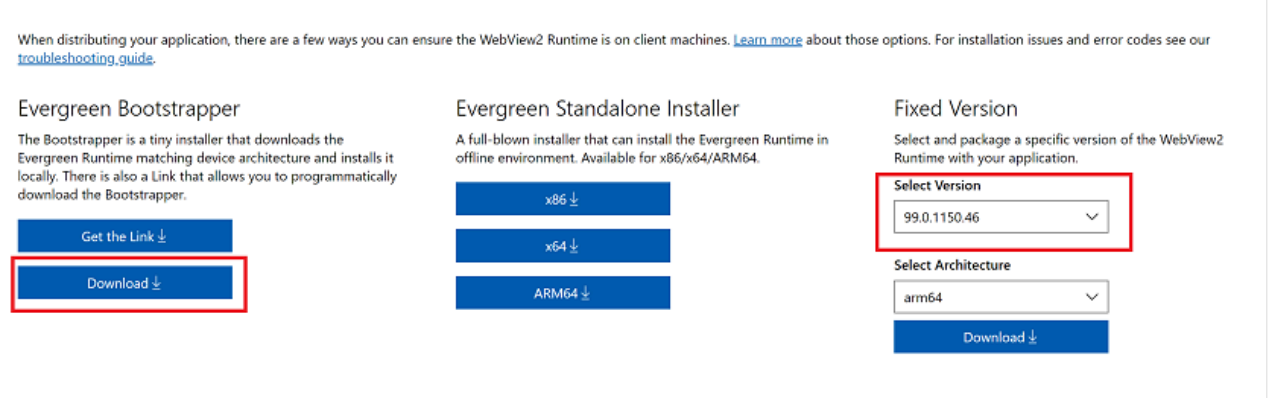
Power Apps for Windows also won't run correctly if you have an older version of WebView2. Follow these steps to see which version you have installed:

1. On your device, go to **Settings** > **Apps** > **Apps & features**.
2. Search for **WebView**. You need version **99.0 or later** installed to run Power Apps for Windows.

In both cases, whether you get an error message or have an older version installed, you need to install WebView2 Runtime.

You need administrator rights on your device to install WebView2. If you don't have admin rights, ask your administrator to install it for you.

1. Download the WebView2 Runtime.
2. Select version **99.0 or later**. Installing an earlier version may cause Power Apps to crash. Under **Evergreen Bootstrapper**, select **Download**.



3. Go over the license terms and privacy statement and then select **Accept and Download**.
4. Run **MicroEdgeWebview2Setup.exe**.

Install from app center

If you don't have access to install Power Apps for Windows from the Microsoft Store then you can install it from the app center.

An administrator can also download the package from the app center and distribute it to users with Intune.

Last updated on 01/08/2026

Use Power Apps for Windows

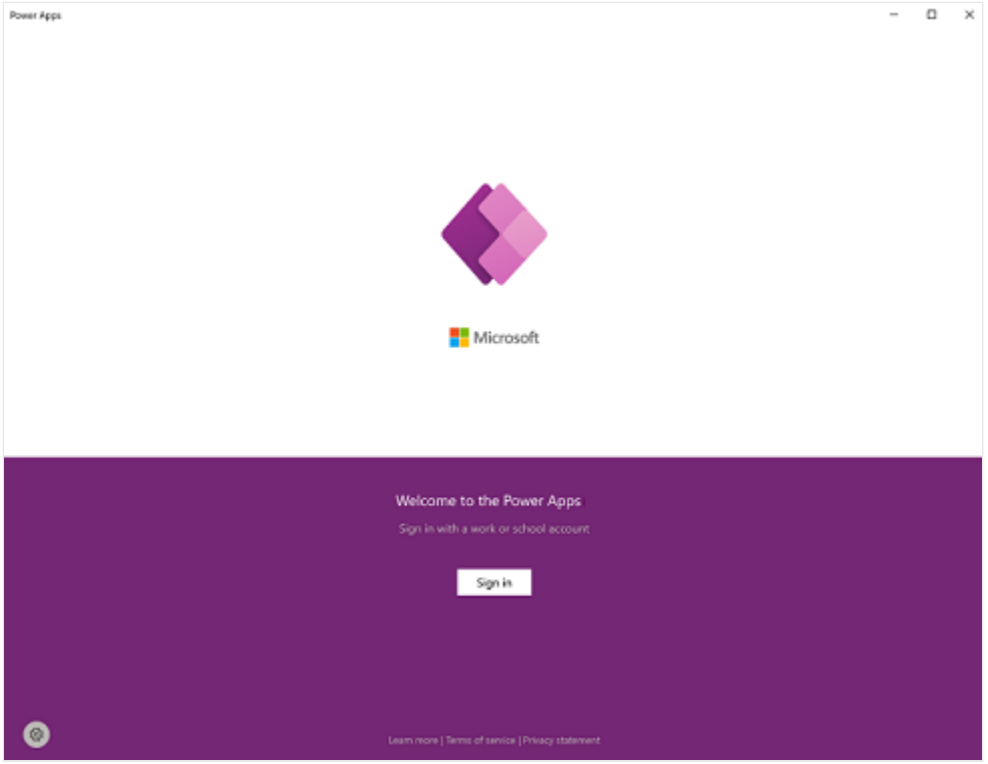
Article • 10/25/2023

This topic covers how to sign in and run your apps on Power Apps for Windows.

Sign in

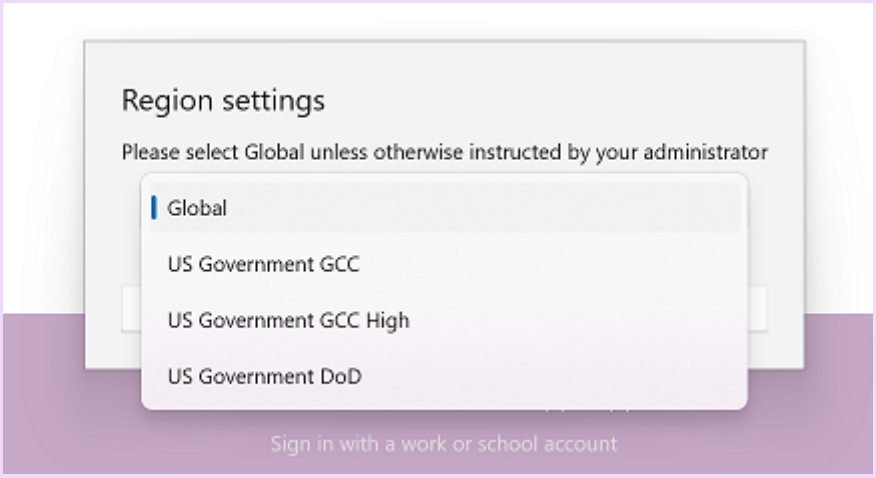
Sign into Power Apps for Windows quickly using your work email address. If you need help with your sign-in information, contact your Power Apps administrator.

1. On the Welcome screen, select **Sign in**.

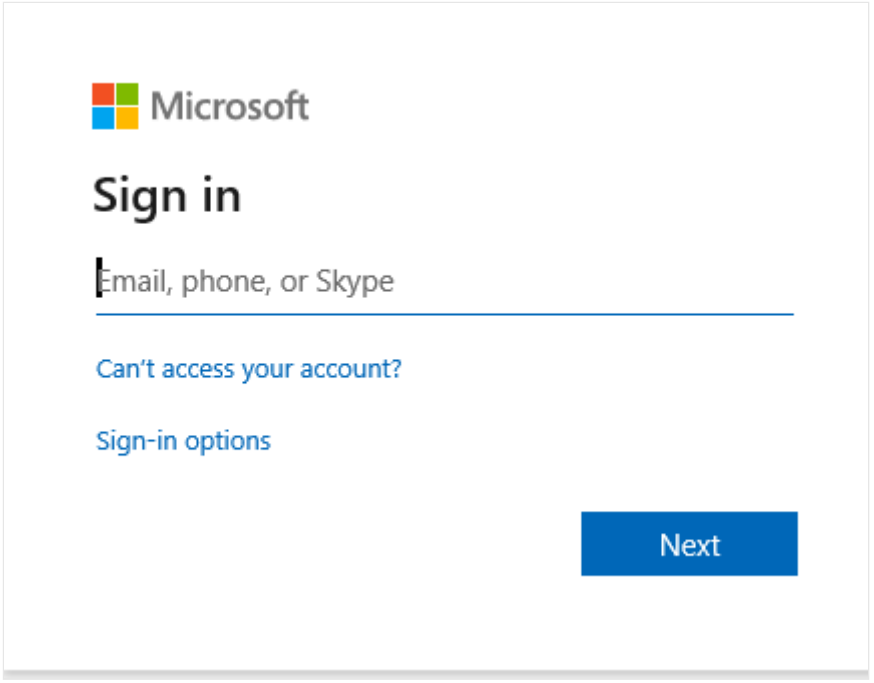


> **Note**

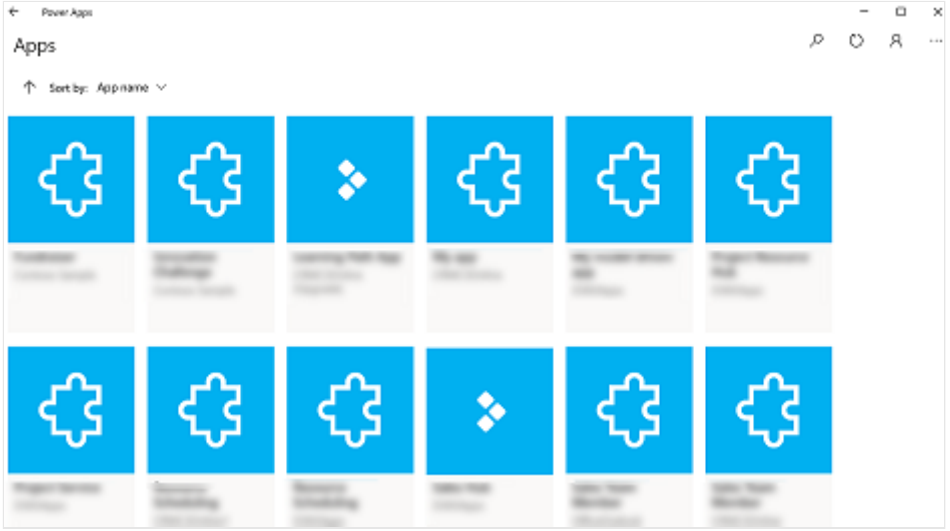
You also have the option to sign in as a guest user and choose a different region.



2. Enter your email address, and then select **Next**. Then enter your password and then select **Sign in**.



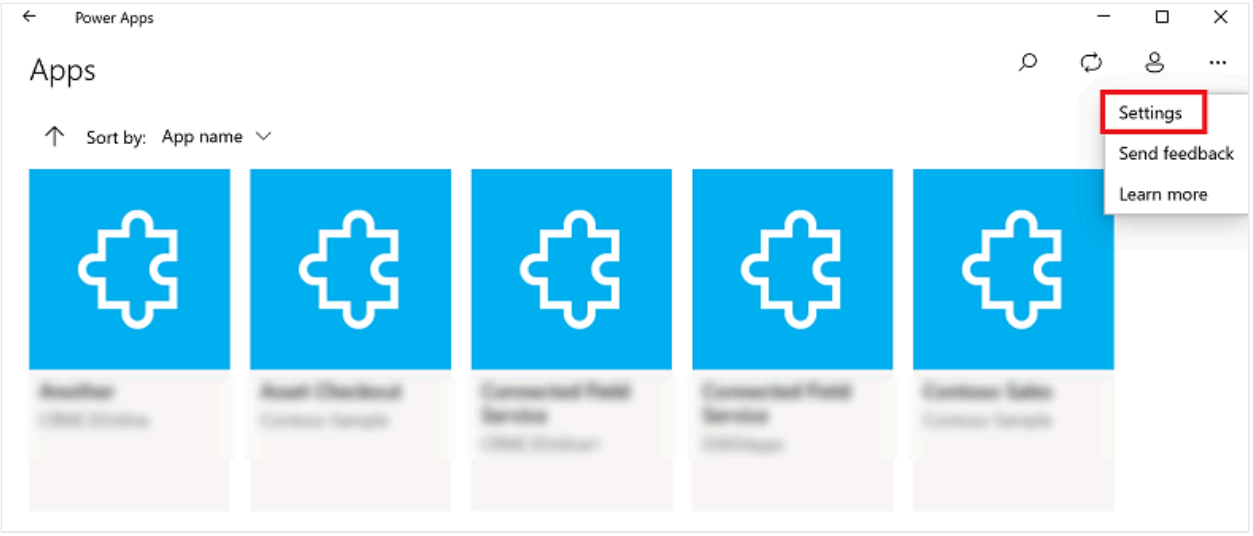
3. From the list of apps that you have access to, select the app that you want to use.



See non-production apps

By default, only production model-driven apps are shown in the list of apps.

To see model-driven apps from non-production environments, select the More button (...) > **Settings** and then turn on **Show non-production apps**.



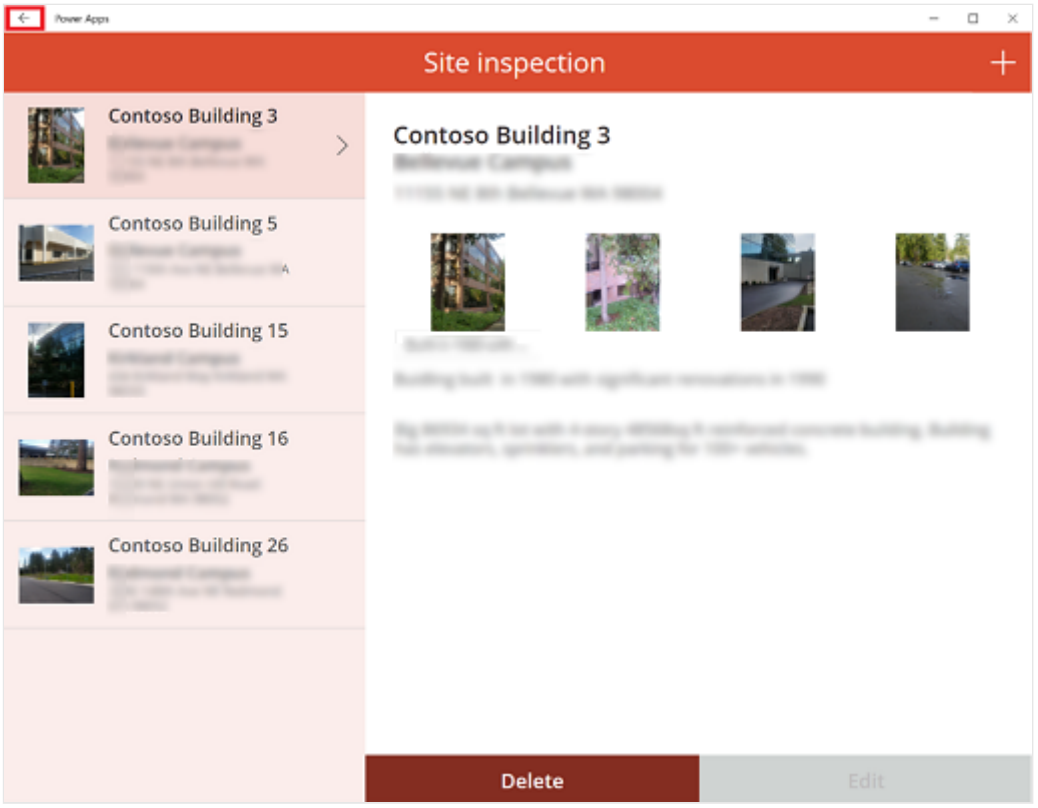
Move between apps

When you're using an app, you can select the back button to go back to the **Apps** screen. From the **Apps** screen, you can then select another app to use.

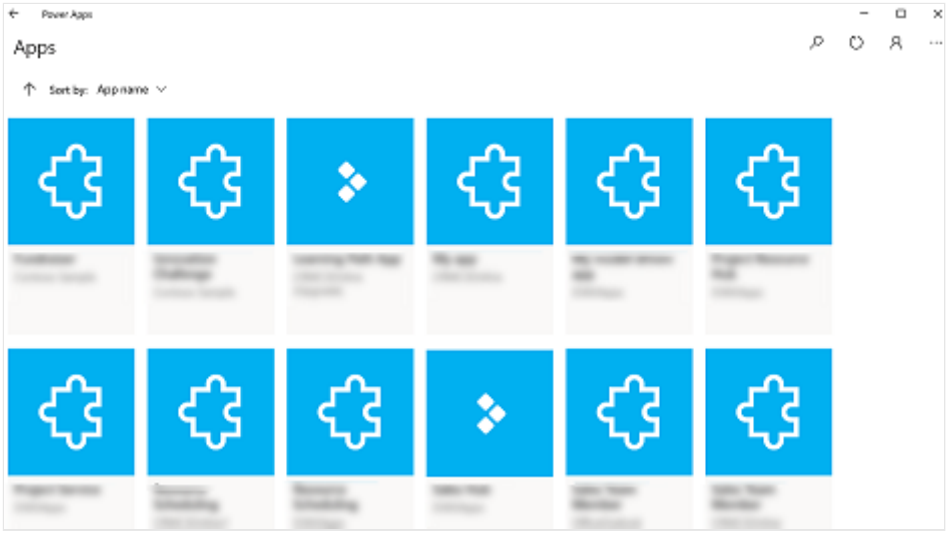
For model-driven apps, you can also select the name of the current app that you're using to go back to the **Apps** screen.

Switch to another canvas app

1. Select the back button.

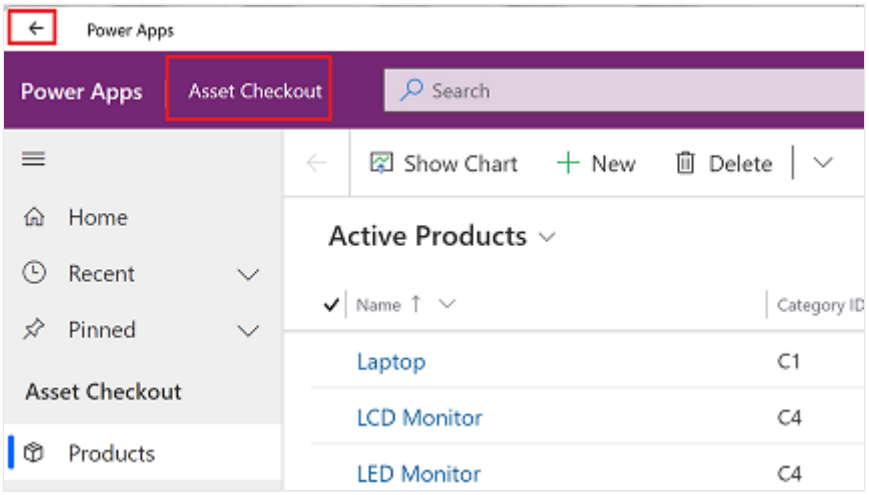


2. On the **Apps** screen select an app. You'll only see apps for your organization.

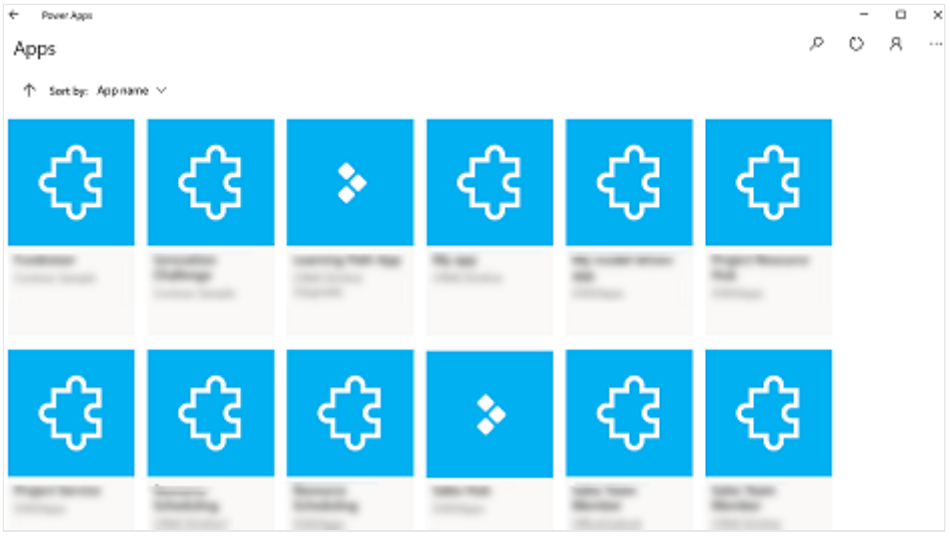


Switch to another model-driven app

1. Select the back button or select the current app name.



2. On the **Apps** screen select an app. You'll only see apps for your organization.



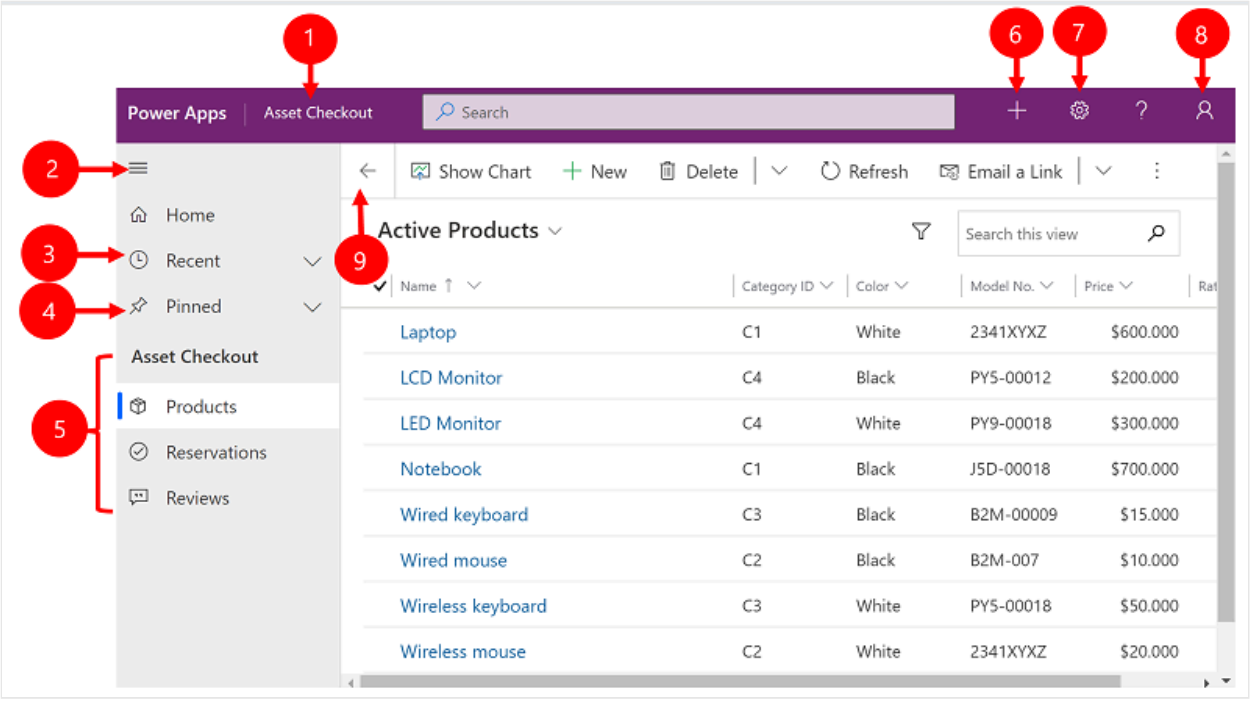
Run an app

Because canvas apps can essentially be anything, there isn't much help documentation on how to use an app once it's created. If you need help using a canvas app, contact your app maker or Power Apps admin.

You'll find below the documentation on how to run model-driven apps on Power Apps for Windows.

Navigation in model-driven apps

It's easy to get around and get back to your favorite or most-used rows. The following illustration shows the primary navigation elements for a model-driven app.



Legend:

1. **Click to change app**: Select the current app name to change to a different app.
2. **Collapse/expand site map**: Select this to collapse the navigator to allow more room for the main part of the page. If the navigator is already collapsed, select this button to expand it again.
3. **Recent**: Expand this entry to view a list of rows you were recently using. Select a row here to open it. Select the push-pin icon next to a row listed here to add it to your pinned rows.
4. **Pinned**: Expand this entry to view and open your favorite pinned rows. Use the **Recent** list to add rows here. Select the remove-pin icon next to a row listed here to remove it from this list.
5. **Page navigation**: This area lists each table and dashboard page available for the current work area. Select any entry here to open the named dashboard or list view for that table.
6. **Create new row**: Select **New** to create a new row.
7. **Setting**: Access your setting such as personalization setting and information about the app.
8. **User Information**: See the current user that's signed in, sign out of the app, or reconfigure the app**.**

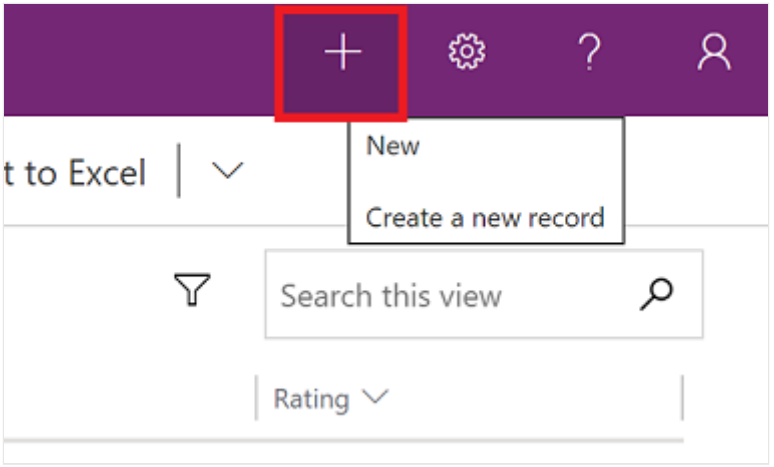
9. **Go back**: Use the back button for form, view, and dashboard pages on the command bar to go back to the previous page.

Create and edit rows

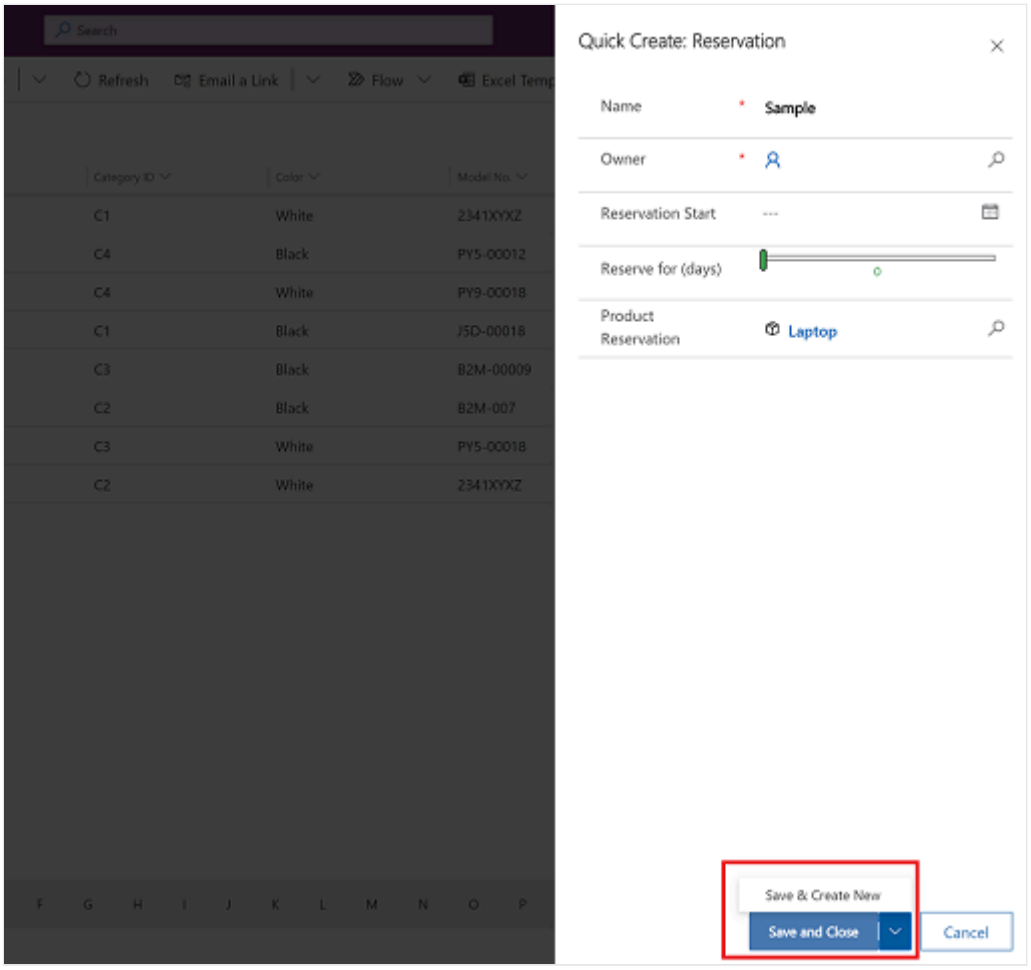
The **New** button or **Quick create** makes it fast and easy to enter almost any type of information into the system. The command is on the navigation bar, so it's available whenever you need to enter new information into the system. You can also save a row and create a new one directly from the **Quick create** form.

Note, the Quick create option is only available for rows that are enabled by your admin.

1. On the navigation bar, select the **plus sign** , and then select the item you want.

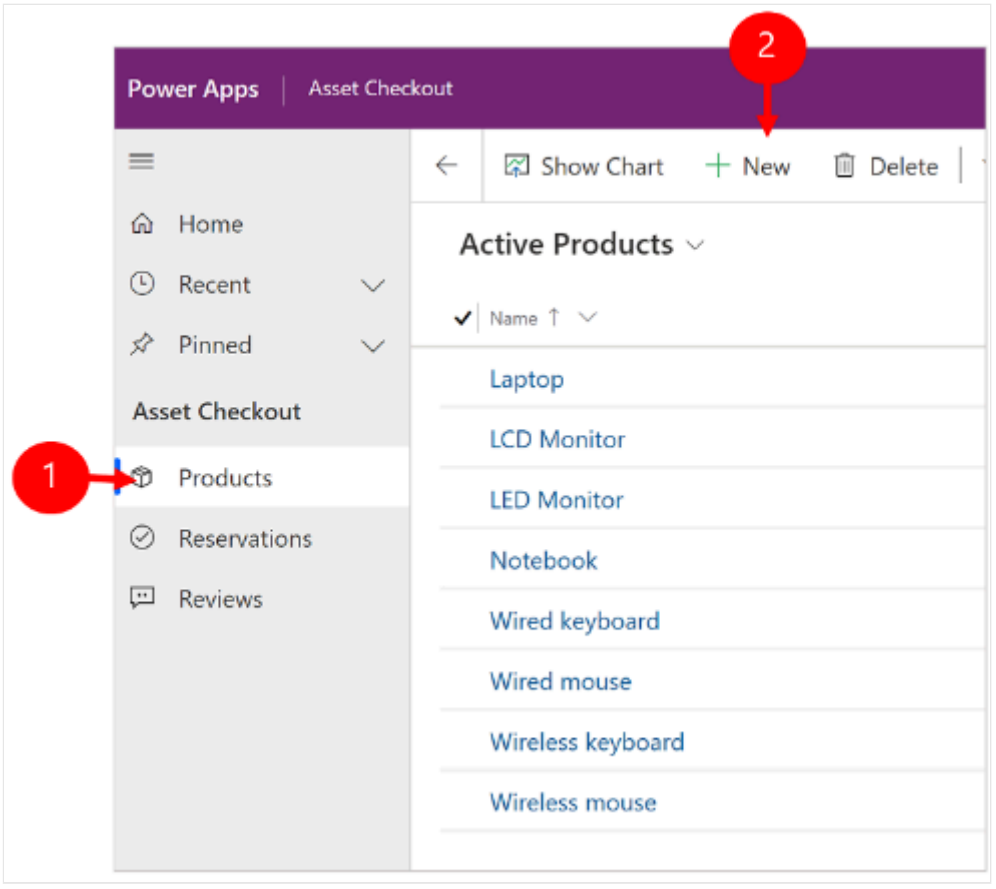


2. Fill in the columns, and then select **Save and Close**. Or, to save and create another row, select the down arrow and then select **Save & Create New**



Create a new row using the New button

1. From the left navigation pane, select a table such as **Products** and then select **New**.

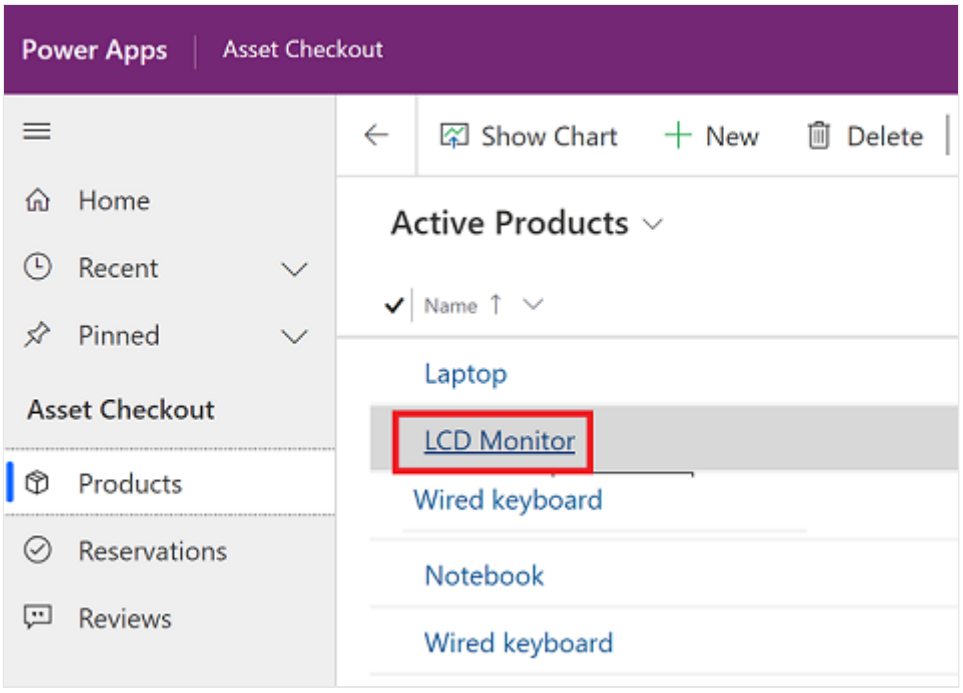


2. Fill in the columns and then on the command bar, select **Save** or **Save & Close**.

Edit a row

1. From the left navigation pane, select a table such as **Products**.

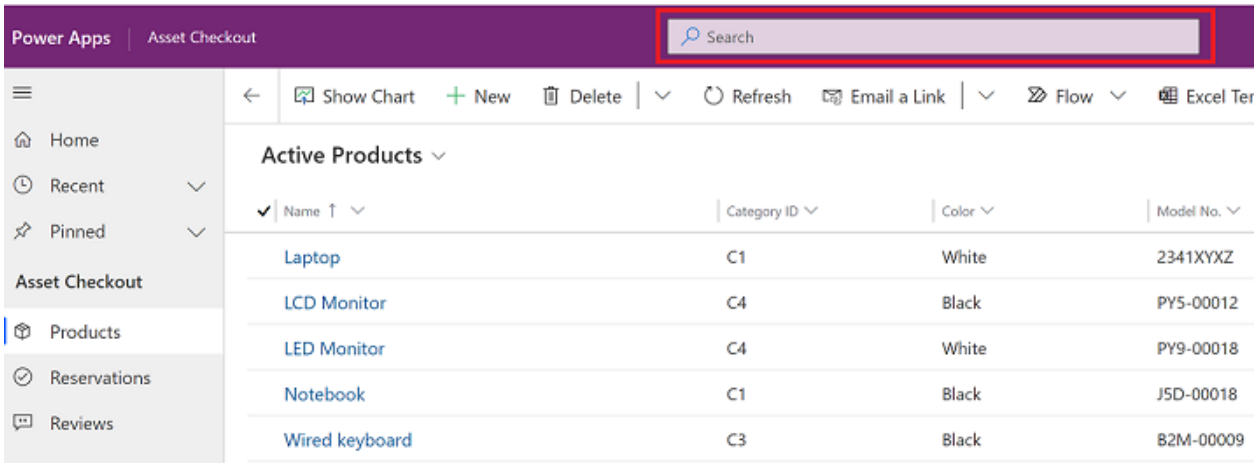
2. Select a row to open it.



3. Make your edits and when you're done editing, select **Save** or **Save & Close**.

Use search

With Dataverse search, the search box is always available at the top of every page in your app. You can start a new search and quickly find the information that you're looking for. For more information, see Search for tables and rows by using Dataverse search.



Notifications

Notifications on Power Apps for Windows work the same as when you run a model-driven app on the web. For more information, see In-app notifications in model-driven apps.

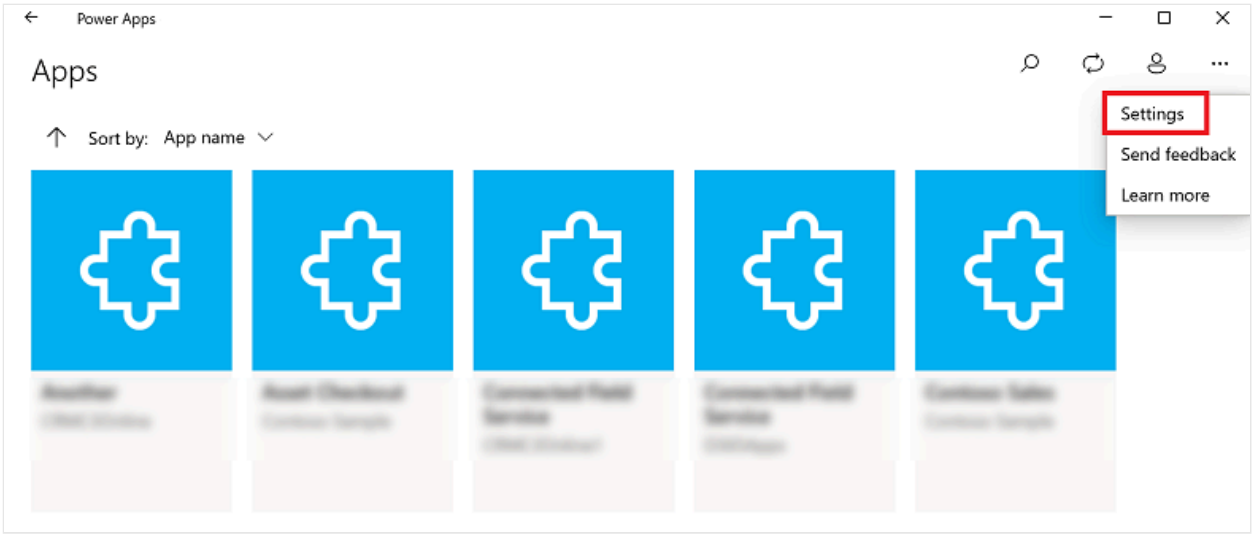
Set up mobile offline

To configure your app in offline mode, see:

- For canvas apps: Develop offline-capable canvas app
- For model-driven apps: Set up mobile offline

Session ID

To get the session ID, select the More button (...) > **Settings** and then copy the session ID.



Log files

Open the following directory on your device to find the log files related to the Windows session for Power Apps for Windows:

`%LocalAppData%\Packages\MicrosoftCorporationII.PowerAppsforWindows10_8wekyb3d8bbwe\LocalState`

The Power Apps support team can use the log files to investigate issues.

Deep links

For more information on deep links, see Use deep links with Power Apps mobile.

Limitations and known issues

- The following is not supported:
  - Advanced controls such as sensors.
  - Pinning an app on Windows desktop
  - Mixed reality controls
  - NFC function
- You can't see favorites, recent, featured apps, or the app details.
- A user picture isn't displayed in the app list.
- The **Optimize images for Upload** option is not available on Windows.
- The barcode scanner doesn't support **BarcodeType**, **PreferFrontCamera**, and **FlashlightEnabled** properties. For more information, see Barcode scanner control in Power Apps - Power Apps. The barcode scanner does support symbologies.

Feedback

Was this page helpful? Yes No

Provide product feedback

Microsoft Dataverse documentation

Securely store and manage data that's used by Power Apps business applications.

Get started

OVERVIEW

What is Dataverse?

Why choose Dataverse?

Improve Copilot responses

TRAINING

Introduction to Dataverse

Define data

CONCEPT

Tables and metadata

Types of tables

Table relationships

Columns

Choices

Customize

HOW-TO GUIDE

Create an app

Create a custom table

Create a relationship between tables

Create and edit columns

Create a choice

Apply business logic

OVERVIEW

Business rules

Plug-ins

Real-time workflows

Actions

More >

Get insights from your data

CONCEPT

Overview

Transition to Dataverse insights

HOW-TO GUIDE

Azure Synapse Link for Dataverse

Link to Microsoft Fabric

Quick view and Power BI integration

Integrate with external data

CONCEPT

Import or export data

Self-service data prep with dataflows

Use SharePoint, OneNote, and OneDrive

HOW-TO GUIDE

Create virtual tables to connect to other data sources

Use solutions for ALM

OVERVIEW

What are solutions?

HOW-TO GUIDE

Create a solution

Export solutions

Validate solutions

More >

For developers

HOW-TO GUIDE

Work with data using code

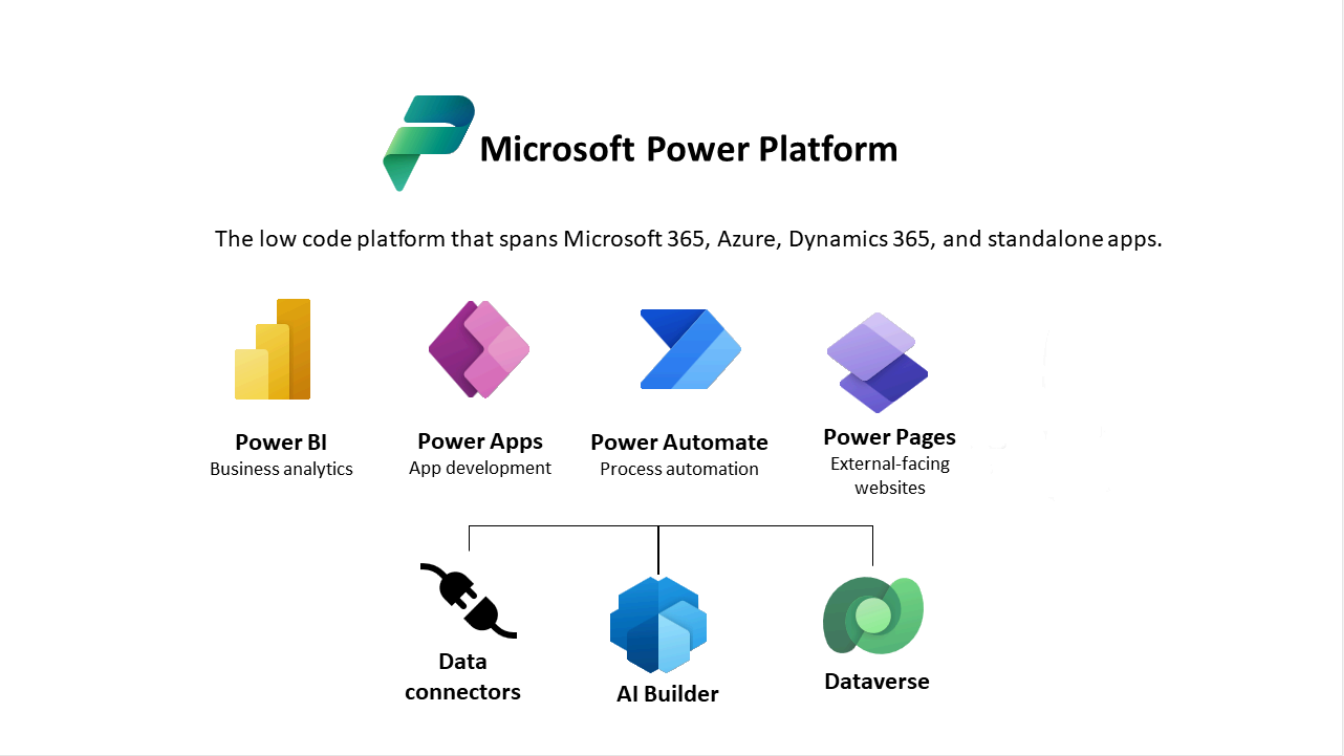
Apply business logic using code

Work with metadata using code

More >

What is Microsoft Dataverse?

Dataverse lets you securely store and manage data that's used by business applications. Data within Dataverse is stored within a set of tables. A *table* is a set of rows (formerly referred to as records) and columns (formerly referred to as fields/attributes). Each column in the table is designed to store a certain type of data, for example, name, age, salary, and so on. Dataverse includes a base set of standard tables that cover typical scenarios, but you can also create custom tables specific to your organization and populate them with data by using Power Query. App makers can then use Power Apps to build rich applications that use this data.



For information about purchasing a plan to use Dataverse, go to Pricing info.

Why use Dataverse?

Standard and custom tables within Dataverse provide a secure and cloud-based storage option for your data. Tables let you create a business-focused definition of your organization's data for use within apps. If you're not sure whether tables are your best option, consider these benefits:

- **Easy to manage** – Both the metadata and data are stored in the cloud. You don't need to worry about the details of how they're stored.
- **Easy to secure** – Data is securely stored so that users can see it only if you grant them access. Role-based security allows you to control access to tables for different users within your organization.
- **Access your Dynamics 365 Data** – Data from your Dynamics 365 applications is also stored within Dataverse, allowing you to quickly build apps that use your Dynamics 365

data and extend your apps with Power Apps.
- **Rich metadata** – Data types and relationships are used directly within Power Apps.
- **Logic and validation** – Define calculated columns, business rules, workflows, and business process flows to ensure data quality and drive business processes.
- **Productivity tools** – Tables are available within the add-ins for Microsoft Excel to increase productivity and ensure data accessibility.

More information: Why choose Dataverse?

Terminology updates

Responding to customer feedback and data from user research, effective November 2020 we're updating some terminology in Dataverse to be more intuitive and make its usage more productive. The terminology updates are listed below, and we're in the process of rolling them out across Microsoft Power Platform.

Expand table

Legacy term	Current term
Entity, entities	Table, tables
Field, fields Attribute, attributes	Column, columns
Record, records	Row, rows
Option set, multi select option sets Picklist, picklists	Choice, choices
Two Options	Yes/No

> **Note**
>
> These terminology updates aren't applicable to any APIs or messages in the Dataverse web services. For example, the names of the messages CreateEntityRequest and CreateAttributeRequest won't change.

Dynamics 365 and Dataverse

Dynamics 365 applications—such as Dynamics 365 Sales, Dynamics 365 Customer Service, or Dynamics 365 Human Resources—also use Dataverse to store and secure the data they use.

This enables you to build apps by using Power Apps and Dataverse directly against your core business data, which is already used within Dynamics 365, without the need for integration.

- **Build apps against your Dynamics 365 data** – Build apps quickly against your business data within Power Apps or by using the extensible platform that lets pro developers programmatically interact with data and metadata, apply business logic, create custom connectors, and integrate with external data.
- **Manage reusable business logic and rules** – Business rules and logic already defined in your Dataverse tables are applied to apps created with Power Apps to ensure data consistency, regardless of how your users access the data or which app they use.
- **Reusable skills across Dynamics 365 and Power Apps** – Users who are skilled in Power Apps or Dynamics 365 can now take advantage of those skills across the Dataverse platform. Creating tables, forms, and charts are now common tasks you perform across your applications.

  > ⓘ **Note**
  >
  > Finance and Operations apps currently require the configuration of the **Data Integrator** to make your business data from Finance and Operations apps available in Dataverse.

Integrating data into Dataverse

Building an app typically involves data from more than one source. Although this can sometimes be done at the application level, there are cases where integrating this data into a common store allows for an easier app-building experience and a single set of logic to maintain and operate over the data. Dataverse allows data to be integrated from multiple sources into a single store, which can then be used in Power Apps, Power Automate, and Power BI along with data that's already available from the Dynamics 365 applications.

- **Scheduled integration with other systems** – Data that's kept within another application can be regularly synchronized with Dataverse to allow you to take advantage of data from other applications in Power Apps.
- **Transform and import data using Power Query** – Transforming data when importing into Dataverse can be done through Power Query—a tool commonly used across Excel and Power BI—from many online data sources.
- **One-time import of data** – Simple import and export of Excel and CSV files can be used for a one-time (or infrequent) import of data into Dataverse.

For more information about integrating data into the Dataverse, go to Add data to a table in Dataverse by using Power Query.

Interacting with tables

When you develop an app, you can use standard tables, custom tables, or both. Dataverse provides standard tables by default. These are designed, in accordance with best practices, to capture the most common concepts and scenarios within an organization.

For a full list of tables, see the entity reference.

You can extend the functionality of standard tables by creating one or more custom tables to store information that's unique to your organization. More information: How to create a custom table

Logic and validation

Tables within Dataverse can take advantage of rich server-side logic and validation to ensure data quality and reduce repetitive code in each app that creates and uses data within a table.

- **Business rules** validate data across multiple columns and tables, and provide warning and error messages, regardless of the app used to create the data. More information: Create a business rule
- **Business process flows** guide users to ensure they enter data consistently and follow the same steps every time. Business process flows are currently only supported for model-driven apps. More information: Business process flows overview
- **Workflows** allow you to automate business processes without user interaction. More information: Workflows overview
- **Business logic with code** supports advanced developer scenarios to extend the application directly through code. More information: Apply business logic with code

Security

Dataverse has a rich security model to protect the data integrity and privacy of users while promoting efficient data access and collaboration. You can combine business units, role-based security, row-based security, and column-based security to define the overall access to information that users have in a Power Platform environment. More information: Security in Dataverse in the Power Platform admin guide

Developer capabilities

In addition to the features available through the Power Apps portal, Dataverse includes features for developers to programmatically access metadata and data to create tables and business logic, in addition to interacting with data. More information: Dataverse Developer Overview

Next steps

To get started using Dataverse:

- Create a canvas app using a Dataverse database.
- Create a custom table and then create a canvas app that uses the table.
- Create a model-driven app built on Dataverse.
- Use Power Query to connect to an online or on-premises data source and import the data directly into Dataverse.

See also

Dataverse for Teams

Last updated on 03/31/2026

Why choose Microsoft Dataverse?

Article • 04/11/2023

Data is at the center of everything a business does today and powers the insights that can drive what it should do tomorrow. To thrive and grow, businesses need to capture, analyze, predict, present, and report data and do it all with a high level of agility.

Building out the data infrastructure to enable business insight can be both time consuming and expensive. The data originates from a variety of devices, applications, systems, services, and software as a service (SaaS). This large and growing number of sources often consists of multiple data technologies that store different types of data, expose different APIs, and use a mixture of security models. The developers needed to create these technologies can be expensive and hard to find. Developers often must have a deep understanding of how to deploy, configure, manage, and integrate these different data technologies.

Dataverse addresses these concerns with an easy to use, easy to manage, compliant, secure, scalable, and globally available SaaS data service. Dataverse empowers organizations to work with any type of data and any type of app, and use the data within it to gain insights and drive business action.

As part of Microsoft Power Platform, Dataverse requires no or little code to be written, so it can easily be used by everyone from knowledge workers to professional developers.

Knowing that it's built on Azure, organizations choosing Dataverse can be confident that it's globally available, compliant, scalable, and secure.

Work with any type of data

Dataverse is designed to work with any type of data and incorporates all the major categories of data technologies that your organization needs—relational, non-relational, file, image, search, and data lake.

Dataverse includes a set of visual designers to create, edit, and interact with data. This makes it easy to quickly define the tables, relationships, business rules, forms, and workflows that represent your business.

With the easy-to-configure integration features built into Dataverse, deep integration with Microsoft's cloud services such as Azure, Dynamics 365, and Microsoft 365—plus access to many connectors in Power Automate and Azure Logic Apps—Dataverse can

connect to the devices, apps, systems, services, and popular SaaS offerings that contain the data for your business.

As a result, a wide range of enterprise integration scenarios—from retrieving data sent in a spreadsheet as an email attachment to emerging scenarios like using Dataverse data in a blockchain network—can be achieved with ease and with little to no code required. Integration efforts that previously were measured in days and weeks can now often be measured in hours and minutes.

In addition to providing the ability to create data or import it from other systems, Dataverse also supports *virtual tables*. Virtual tables map data in an external data source so that it appears to exist in Dataverse. This enables Dataverse to execute real-time data operations against the external data source. More information: Create and edit virtual tables that contain data from an external data source

Work with any type of app

When an organization wants to create a new app, it can realize additional productivity gains by using Dataverse with Power Apps. Power Apps understands the rich metadata included in Dataverse and uses it in multiple ways to help you rapidly build great-looking apps that are secure and scalable, and make them available across desktop, web, mobile, and Microsoft Teams.

Organizations that use Power Apps can quickly develop mobile apps for iOS and Android. You can also take advantage of Dataverse mobile offline functionality, which enables apps to collect, query, and interact with data when offline.

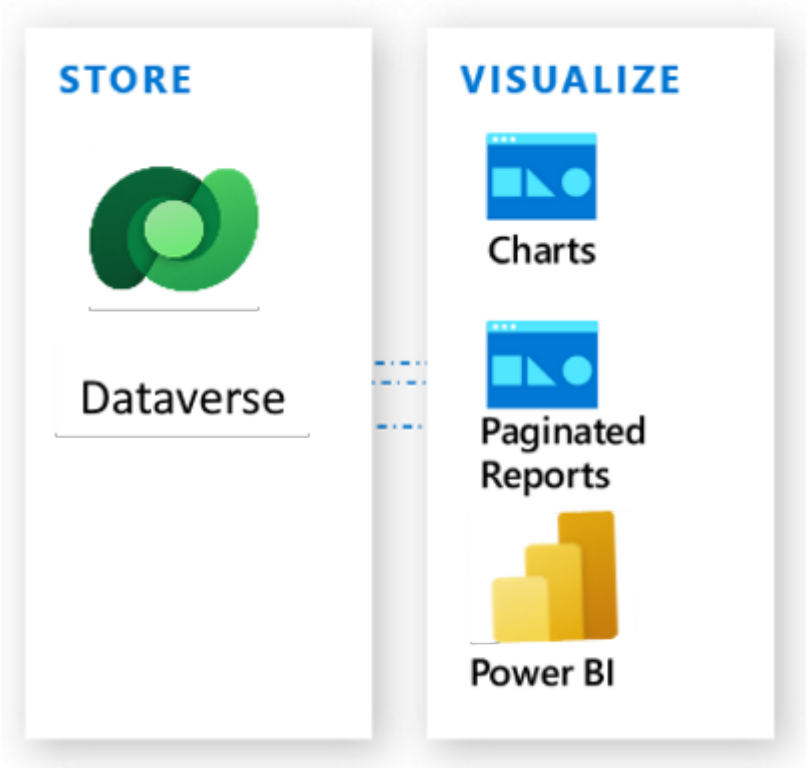
For organizations that want to integrate Dataverse data into existing apps or write new apps by using custom code, Dataverse provides a powerful REST-based API, a developer SDK, and a growing list of samples for common scenarios.

You can also use Dataverse in bot-based apps that deliver intuitive, interactive dialogues with employees, partners, and customers. Whether they're embedded within Power Apps or custom code, bots can be built quickly by using Microsoft Copilot Studio and powered by Dataverse data.

The goal of Dataverse working with any app means it should also work with the tools that knowledge workers and professional developers use. To help them be even more productive, Dataverse is integrated into popular tools such as Excel, Outlook, Dynamics 365 customer engagement apps, Power BI Desktop, Power Query, Azure Data Factory, Data Export Service, and SQL Server Management Studio. More information: Work with any type of app

Analytics and reporting

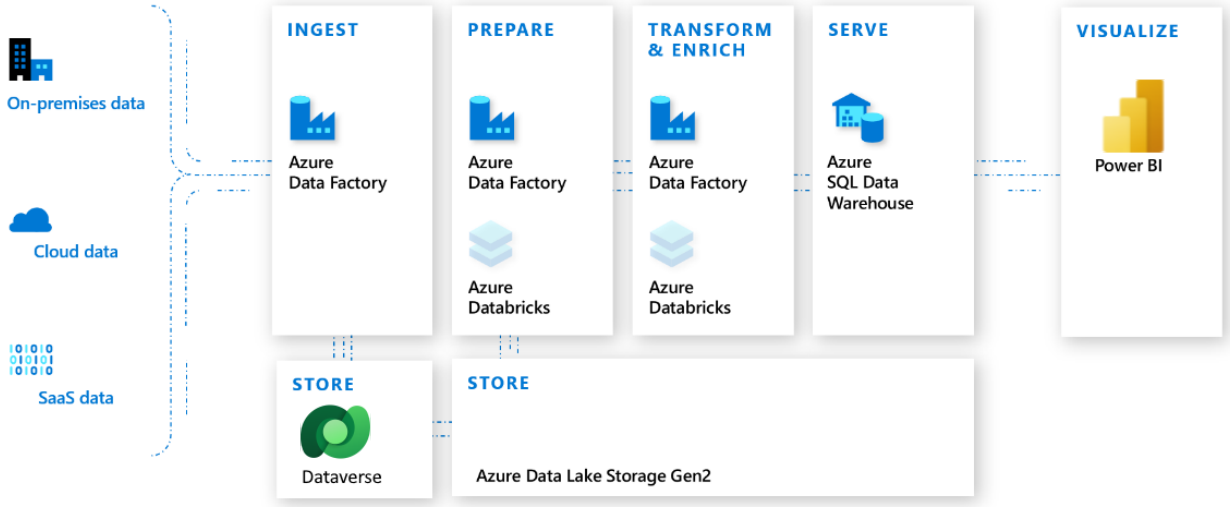
Dataverse can be used to gain insights and drive business action using analytics and reporting. Dataverse also includes a variety of ways to deliver data and insight to key decision makers.



Dataverse includes the ability to create lightweight charts and paginated reports. The data in Dataverse can also be used with Power BI to create rich, interactive reports and dashboards.

For organizations that are interested in employing AI to analyze their data, AI Builder can give everyone in the organization—regardless of their technical expertise—the ability to add AI capabilities to the business process flows they create and use. Delivered as part of Microsoft Power Platform, AI Builder includes six pre-built AI models that can be used in Power Automate and Power Apps, and to evaluate data within Dataverse.

To support advanced analytics and machine learning, Dataverse includes a managed data lake. Data within the lake can be used to run Power BI reporting, machine learning, data warehousing, and other downstream data processing.



One of the benefits of the data being in Azure Data Lake is that organizations can take advantage of Azure Synapse Analytics. This service can deliver added productivity by bringing together enterprise data warehousing, data exploration, code-free data orchestration, deeply integrated Apache Spark and SQL engines, and integrated AI and BI.

Security

Dataverse uses Microsoft Entra identity and access management mechanisms to help ensure that only authorized users can access the environment, data, and reports.

Dataverse uses role-based security to group together a collection of privileges. These security roles can be associated directly with users, or they can be associated with Dataverse teams and business units.

In Dataverse, individual rows can be shared on a one-by-one basis with another user. Because row-level control of access isn't adequate for some business scenarios, Dataverse has a column-level security feature to allow more granular control of security at the column level.

Dataverse also includes two security models that can be used for hierarchies: the *manager* hierarchy and the *position* hierarchy. With the manager hierarchy, a manager must be within the same business unit as the report, or in the parent business unit of the report's business unit, to have access to the report's data. The position hierarchy allows data access across business units.

Because Dataverse is built on Azure, it benefits from the Azure platform's powerful security technologies. Encryption of data, at rest and in transit, preserves confidentiality.

Dataverse is governed by the Microsoft Online Services Terms and the Microsoft Privacy Statement.

More information: Security concepts in Dataverse

Compliance

Compliance is a key concern for organizations. Microsoft regularly engages with dozens of regulators around the world, so organizations can be assured that the data placed in Dataverse is held in accordance with stringent industry safeguards. Dataverse complies with numerous standards, and compliance is verified by third-party audits and certifications.

The list of currently supported standards can be found at the Microsoft Trust Center website.

Availability and scalability

Dataverse is designed to meet enterprise-level scalability needs and offers a service level agreement of 99.9% uptime.

To help ensure service levels, availability, and quality, entitlement limits to the number of requests users can make each day are governed by licenses. Service protection limits have also been put in place against malicious behavior that would otherwise disrupt service for all customers.

Microsoft Power Platform requests

Microsoft Power Platform requests consist of actions that users perform across various products.

At a high level, the following constitute Microsoft Power Platform requests:

- **Connectors**: All API requests to connectors from Power Apps or Power Automate.
- **Power Automate**: All Power Automate step actions.
- **Dataverse**: All create, update, delete (CRUD) operations, in addition to special operations like "share" or "assign." These can come from any client or app, using any SOAP or REST endpoint. These include—but aren't limited to—plug-ins, asynchronous workflows, and custom controls making the abovementioned operations.

The specific number of allowed requests that a user can make in a 24-hour period is governed by the user's license.

Entitlement limits

Entitlement limits represent the number of requests users are entitled to make each day. The allocated limit depends on the type of license assigned to each user.

For information about these entitlement limits, see Microsoft Power Platform requests allocations based on licenses.

For information about viewing and allocating capacity add-ons, see Capacity add-ons.

For information about purchasing individual capacity add-ons, see the Power Apps and Power Automate licensing guide.

Service protection limits

Service protection limits exist to protect the health of the service for everyone. These limits provide a level of protection against random and unexpected surges in request volumes that threaten the availability and performance characteristics of the Dataverse platform.

The service limits the number of concurrent connections per user account, the number of API requests per connection, and the amount of execution time that can be used for each connection. These are evaluated within a five-minute sliding window. When one of these limits is exceeded, an exception is returned by the platform.

Service limits aren't expected to negatively affect normal usage of Dataverse.

For information about current service protection limits for each service, see:

- Dataverse API request limits: Applicable to Dynamics 365 customer engagement apps, such as Dynamics 365 Sales and Dynamics 365 Customer Service, in addition to connections from Power Apps and Power Automate to Dataverse.
- Microsoft Power Automate limits: Applicable for Power Automate.
- Limits in connectors: Applicable for Power Automate and Power Apps.

Capacity add-ons

The specific number of allowed Microsoft Power Platform requests that a user can make in a 24-hour period is governed by the user's license. For scenarios where additional capacity is required, a Power Apps and Power Automate capacity add-on allows customers to purchase additional requests.

Each capacity add-on provides an additional 10,000 requests every 24 hours, which can be assigned to any user. Multiple capacity add-ons can also be assigned to the same user.

Dataverse has entitlement limits that will evaluate the number of requests made against the number allowed by their current license.

Backups

Dataverse provides two types of backups: automatic backups, referred to as *system* backups, and *manual* backups.

System backups

System backups back up all environments. They take place automatically and continuously. The underlying technology used is Azure SQL Database. For more information about Azure SQL automated backups, see Automated backups.

System backups for production environments that have been created with a database and have one or more Dynamics 365 applications installed are retained for 28 days. System backups for production environments that don't have Dynamics 365 applications deployed in them are retained for seven days. System backups for sandbox environments are retained for seven days.

Manual backups

Manual backups are user-initiated, typically done before making a significant customization change or applying a version update.

Both sandbox and production environments can be manually backed up. Sandbox backups are retained for seven days. Manual backups for production environments that have been created with a database and have one or more Dynamics 365 applications installed are retained for 28 days. Manual backups for production environments that don't have Dynamics 365 applications deployed in them are retained for seven days.

There's no limit to the number of manual backups that can be made, and manual backups don't count against storage limits.

> **Note**

Both system and manual backups may only be restored to an environment in the same region in which it was backed up.

More information: Back up and restore environments

Datacenter regions

As many organizations do business globally, their data needs are also global in nature. With Dataverse deployed and supported in regions across the world, organizations can have confidence that Dataverse is available when and where they need it.

For the current list of datacenter regions, see Datacenter regions.

See also

Work with any data

Work with any data

Article • 03/02/2023

Microsoft Dataverse provides an abstraction that makes it possible to work with any type of data, including relational, non-relational, image, file, relative search, or data lake. There is no need to understand the type of data as Dataverse exposes a set of data types that allow you to build up your model. The type of storage is optimized for the data type chosen.

Data can be easily imported and exported with dataflows, Power Query, and Azure Data Factory. Dynamics customers can also use the Data Export Service.

Dataverse also has a connector for Power Automate and Azure Logic Apps that can be used with the hundreds of other connectors in those services for on-premises, infrastructure as a service (IaaS), platform as a service (PaaS), or software as a service (SaaS) services. This includes sources in Azure, Microsoft 365, Dynamics 365, SAP ERP, Salesforce, Amazon Redshift, Access, Excel, text/CSV, SharePoint lists, SQL Server databases, Oracle, MySQL, PostgreSQL, Blockchain, and Azure Synapse Analytics.

Common Data Model

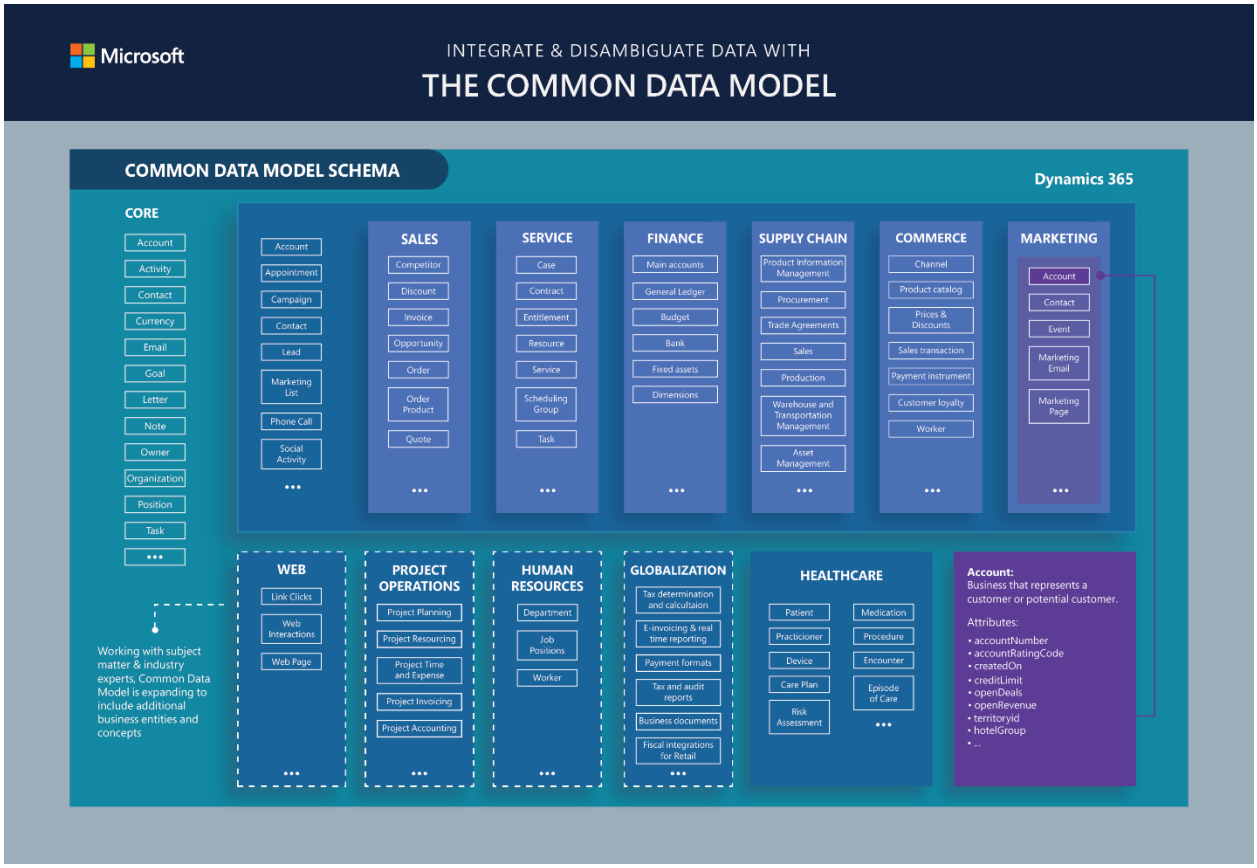
If you've ever had to bring data from multiple systems and apps together, you know what an expensive and time-consuming task that can be. Without being able to share and understand the same data easily, each app or data integration project requires a custom implementation.

Common Data Model provides reference architecture that's intended to streamline this process by providing a shared data language for business and analytical apps to use. The Common Data Model metadata system makes it possible for data and its meaning to be shared across apps and business processes such as Power Apps, Power BI, Dynamics 365, and Azure.

Common Data Model includes a set of standardized, extensible data schemas that Microsoft and its partners have published. This collection of predefined schemas includes tables, attributes, semantic metadata, and relationships. The schemas represent commonly used concepts and activities, such as **Account** and **Campaign**, to simplify the creation, aggregation, and analysis of data.

The Common Data Model schemas can be used to inform the creation of tables in Dataverse. The resulting tables will then be compatible with apps and analytics that target this Common Data Model definition.

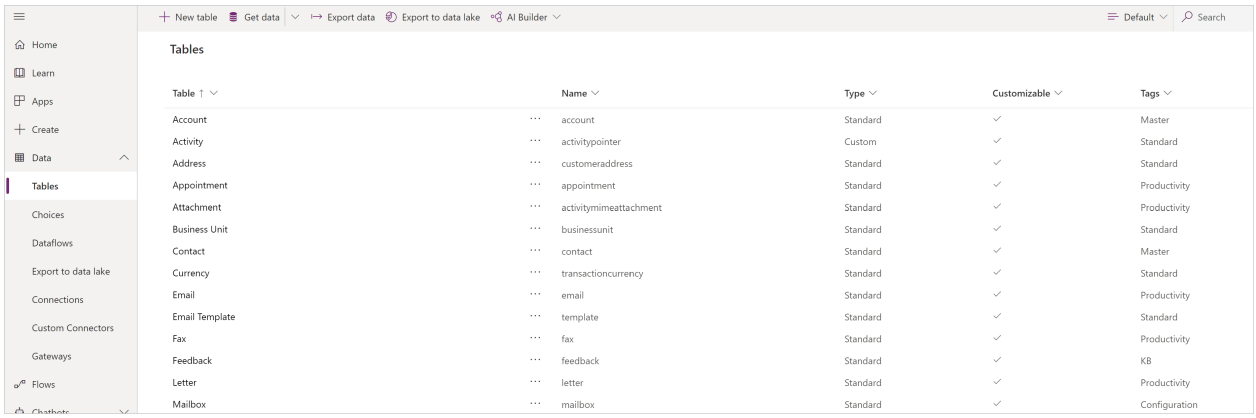
The following image shows some elements of the standard Common Data Model tables.



Tables

In Dataverse, tables are used to model and manage business data. To increase productivity, Dataverse includes a set of tables known as standard tables. These tables are designed, in accordance with best practices, to capture the most common concepts and scenarios within an organization. The standard tables adhere to Common Data Model.

A set of tables that are commonly used across industries, such as **User** and **Team**, are included in Dataverse and referred to as *standard tables*. These out-of-the-box tables can also be customized, such as including additional columns. In addition, you can easily create your own custom tables in Dataverse.

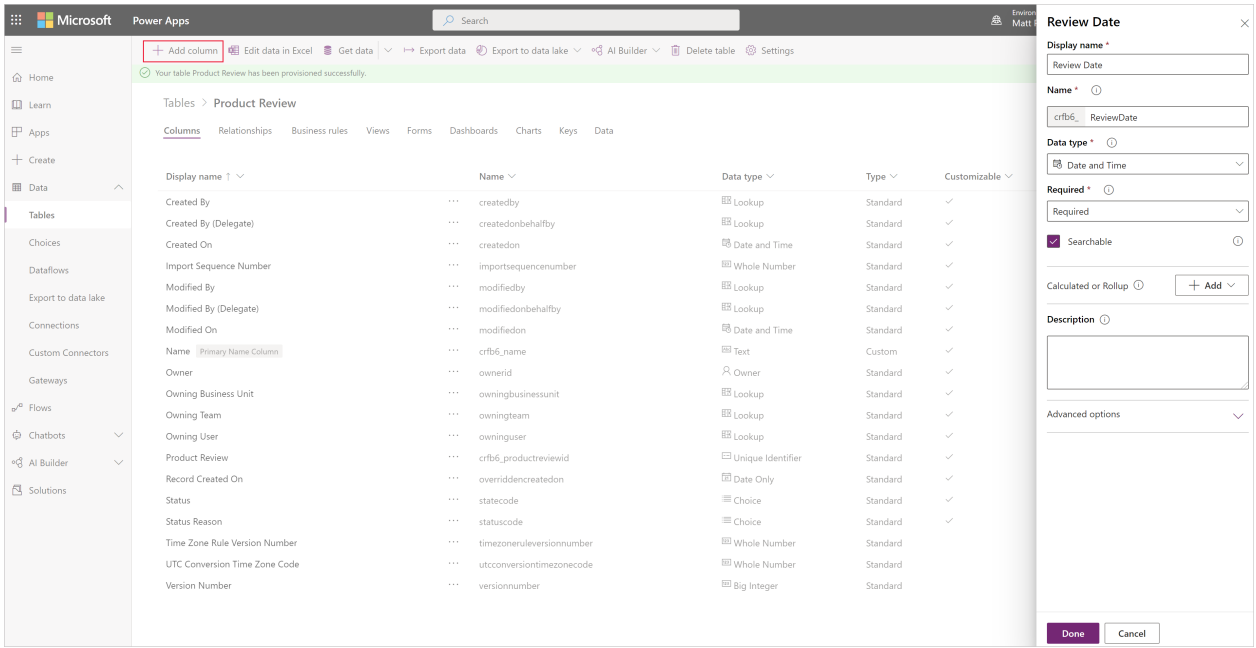


Columns

Columns define the individual data items that can be used to store data in a table. Fields are sometimes called attributes by developers. A table representing a course at a university might contain columns such as "Name," "Location," "Department," "Registered Students," and so on.

Columns might have different types of data such as numerals, strings, digital data, images, and files. There's no need to keep relational and non-relational data separated artificially if it's part of the same business process or flow. Dataverse stores the data in the best storage type for the model created.

Each of these columns can be associated with one of many data types supported by Dataverse.

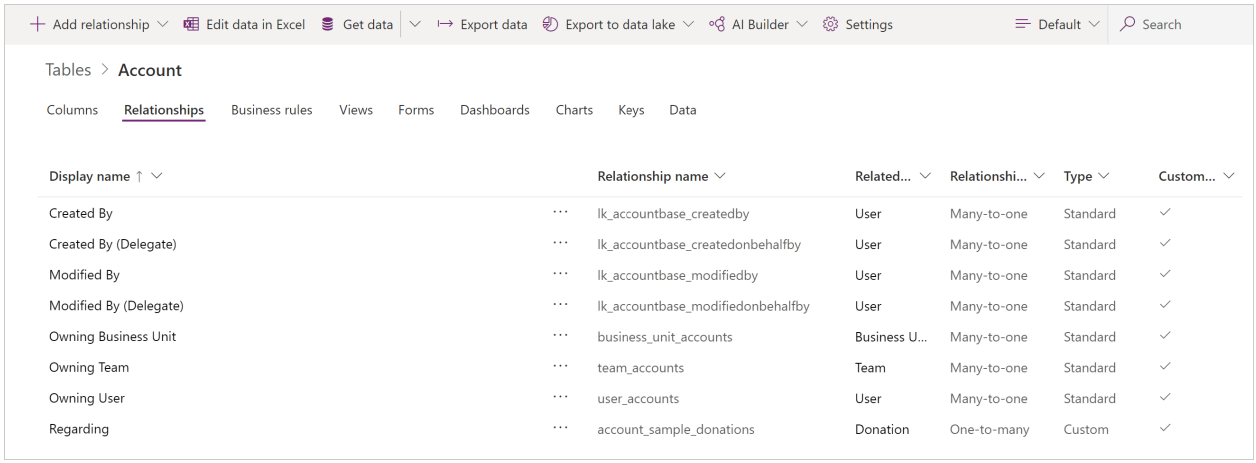


More information: Types of columns

Relationships

Data in one table often relates to data in another table. table relationships define how rows can be related to each other in the Dataverse model.

Dataverse provides easy-to-use visual designers to define the different types of relationships from one table to another (or between a table and itself). Each table can have a relationship with more than one table, and each table can have more than one relationship to another table.



The relationship types are:

- **Many-to-one**: In this type of relationship, many table A records can be associated with a single table B record. For example, a class of students have a single classroom.
- **One-to-many**: In this type of relationship, a single table B record can be associated with many table A records. For example, one single teacher teaches many classes.
- **Many-to-many**: In this type of relationship, each record in table A can match more than one record in table B, and vice versa. For example, students attend many classes, and each class can have multiple students.

Because many-to-one relationships are the most common, Dataverse provides a specific data type named *lookup*, which not only makes it easy to define this relationship but adds productivity to building forms and apps.

For more information about creating table relationships, see Create a relationship between tables.

Organizations often need to be in compliance with various regulations to ensure the availability of customer interaction history, audit logs, access reports, and security incident tracking reports. Organizations might want to track changes in Dataverse data for security and analytical purposes.

Dataverse provides an auditing capability where changes to tables and attribute data in an organization can be rowed over time for use in analysis and reporting. Auditing is supported on all custom—and most customizable—tables and attributes. Auditing isn't supported on metadata changes, retrieve operations, export operations, or during authentication. For information about how to configure auditing, go to Manage Dataverse auditing.

Dataverse supports analytics by providing the capability to choose tables for machine learning models to run. It has a prebuilt AI capability through AI Builder.

Search

Dataverse provides three ways to query rows:

- Dataverse search
- Quick find (single-table or multi-table)
- Advanced find

> **Note**
>
> Multi-table quick find is also called *categorized search*.

For more information, see Compare searches.

Dataverse search

Dataverse search delivers fast and comprehensive results across multiple tables in a single list, sorted by relevance. It uses a dedicated search service external to Dataverse (powered by Azure) to boost search performance.

Dataverse search brings the following enhancements and benefits:

- Improves performance by using external indexing and Azure search technology.
- Finds matches to any word in the search term in any column in the table, compared to quick find where all words from the search term must be found in one column.
- Finds matches that include inflectional words like *stream*, *streaming*, or *streamed*.

- Returns results from all searchable tables in a single list sorted by relevance, so the better the match, the higher the result appears in the list. A match has a higher relevancy if more words from the search term are found in close proximity to each other. The smaller the amount of text where the search words are found, the higher the relevancy. For example, if you find the search words in a company name and address, it might be a better match than finding the same words in a long article, far apart from each other.

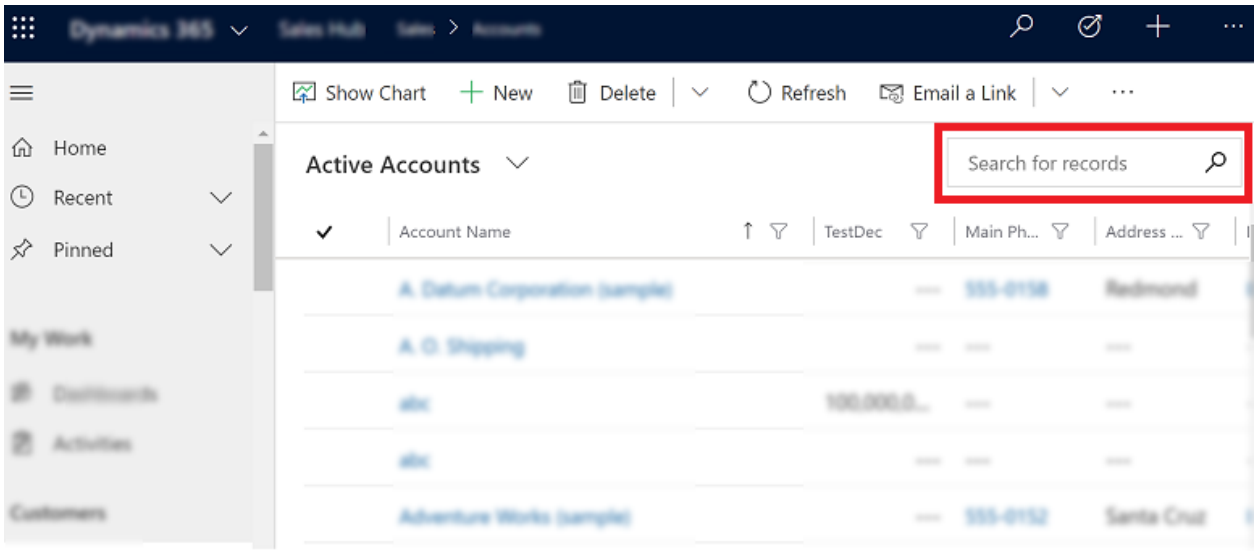
- Highlights matches in the results list. When a search term matches a term in a row, the term appears as bold and italicized text in your search results.

For more information about Dataverse search, see Using Dataverse search to search for rows.

Quick find

Dataverse includes the ability to find rows quickly and has approaches that will search just one type of table, such as customer, or be used to search across multiple types of tables at the same time, such as contacts, users, customers, and so on.

Single-table quick find is used to find rows of only one type. This search option is available from within a view.

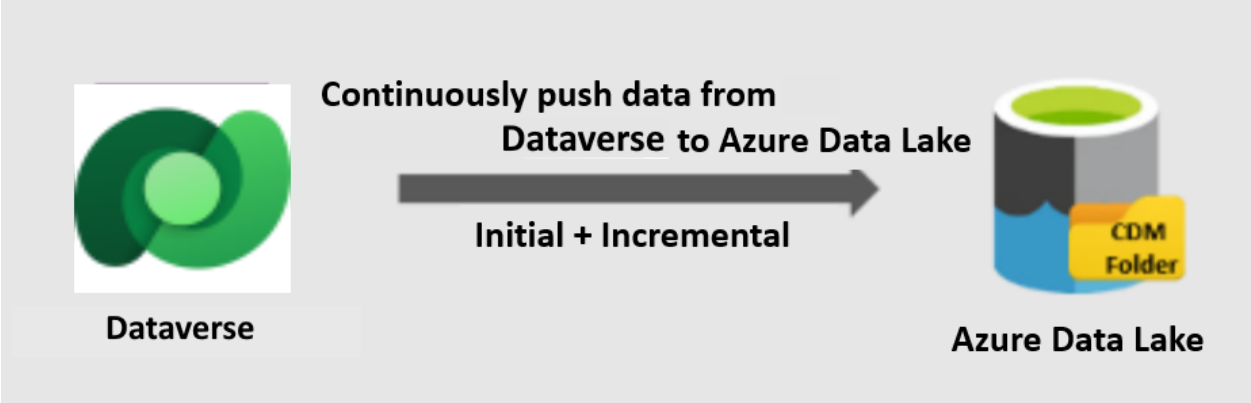


Multiple-table quick find (categorized search) is also used to find rows, but will find them across different types of tables, such as accounts or contacts.

Data Lake

Dataverse supports continuous replication of table data to Azure Data Lake Storage, which can then be used to run analytics such as Power BI reporting, machine learning,

data warehousing, and other downstream integration processes.



This feature is designed for enterprise big-data analytics. It's cost-effective, scalable, has high availability and disaster recovery capabilities, and enables best-in-class analytics performance.

Data is stored in the Common Data Model format, which provides semantic consistency across apps and deployments. Standardized metadata and self-describing data in Common Data Model facilitates metadata discovery and interoperability between data producers and consumers such as Power BI, Data Factory, Azure Databricks, and Azure Machine Learning.

See also

Importing and exporting data

Importing and exporting data from Dataverse

Article • 08/06/2024

Microsoft Dataverse is a SaaS data platform that helps you easily create and manage your data, events, and logic and generate analytics** and insights to support interconnected apps built on the Dataverse platform. Apps include first-party apps such as Dynamics 365 Sales, Service, Marketing, Customer Insights, Customer Journey Orchestration, as well as custom third-party apps and processes, all of which operate in a secure and compliant manner. Dataverse, being a hyperscale polyglot storage, can store any type of data (relational, file, observational, and so on) for all your transactional and analytical applications. For this reason, it's the data platform for all Power Platform products and Dynamics 365 applications. Dataverse is a critical element of Microsoft Power Platform, supporting all the data, events, analytics, and related processing, in addition to generating insights for citizen app makers and professional developers to build, store, and manage data for their applications.



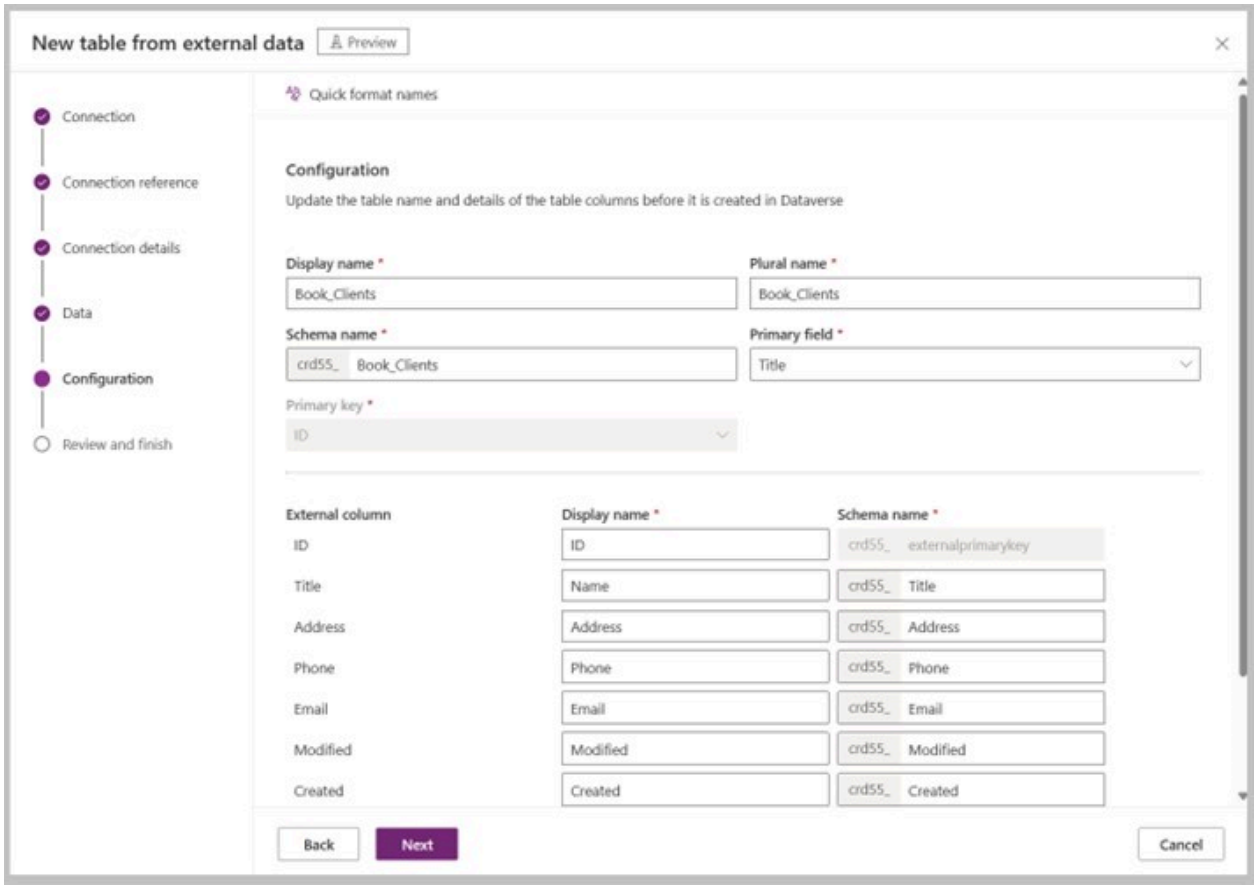
Data required by apps and processes doesn't often originate from or reside within Dataverse. Mashing up external data with Dataverse is an essential component in building apps, adding data to existing apps, and creating valuable insights. With the proliferation of big data and the ever-multiplying types of data sources, such as AI, ML, IoT, web commerce, web API, services, ERP, and line of business apps require Dataverse customers to be agile with using this variety of data.

There are multiple ways to import and export data into Microsoft Dataverse. You can use dataflows, Power Query, Azure Synapse Link, Azure Data Factory, Azure Logic Apps, and Power Automate.

Where to start?

The first consideration is to start with the external data, which already exists outside of Dataverse that is required for building an app, form, or other component. It isn't necessary to persist that data in Dataverse to start using it. Many times, it's best to continue to let the other system manage the data while using the data as needed in Dataverse. The virtual table feature in Dataverse provides this capability. More information: Create virtual tables using virtual connectors.

With virtual tables, you can build an app that uses the external data as if it were a Dataverse table. To help you create virtual tables easier, use the virtual tables creation wizard now in Public Preview | Microsoft Power Apps



Import external data to Dataverse

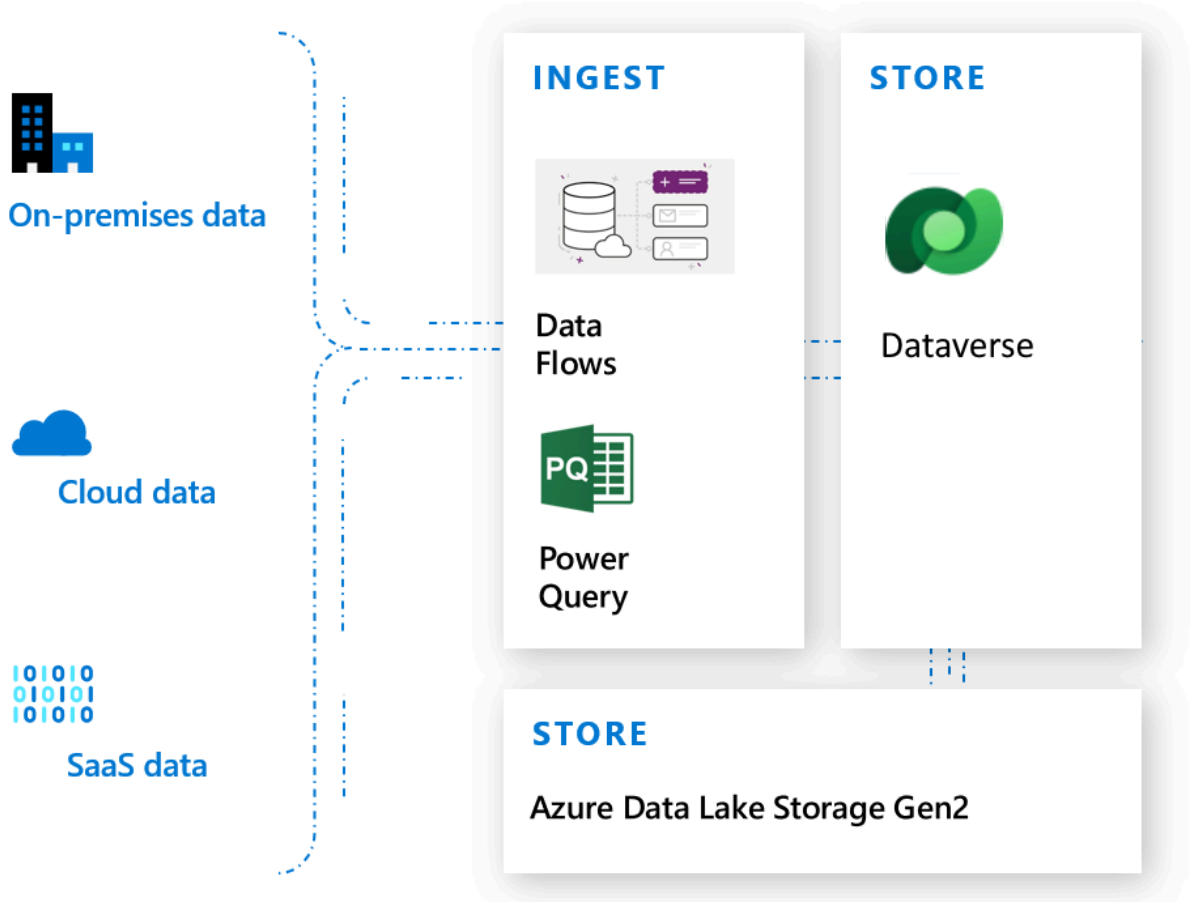
Your desired outcomes might drive the decision to persist data in Dataverse versus using virtual tables to surface external data in Dataverse. If you want your data to be managed by security concepts in Dataverse and application lifecycle management

features or if you're interested in mashing up external data with native Dataverse data, consider moving data into Dataverse. Additionally, if you're interested in using Dataverse workflow and business rules, you might consider migrating that data into Dataverse. You should also consider the economic impact of loading data into Dataverse, which might increase storage costs. If the data must continue to be managed by an external system, it can be beneficial to leave it there and interact with it in Dataverse using virtual tables.

Dataflows, Power Query, and Excel files

Dataflows enable you to connect with business data from various sources, clean the data, transform it, and then load it into Dataverse. Dataflows support dozens of popular on-premises, cloud, and software as a service (SaaS) data sources.

Power Query is a data connection technology you can use to discover, connect, combine, and refine data sources to meet your analysis needs. Features in Power Query are available in Excel and Power BI Desktop.

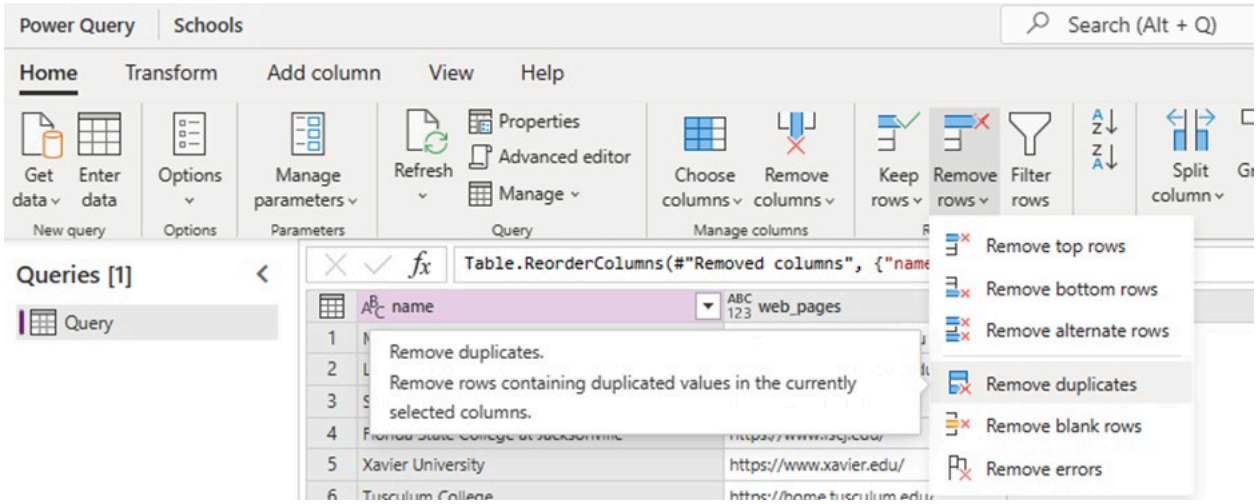


During the project envisioning phase, you should weigh out all the options and decide whether external data needs to be part of your transactional data. Once the approach is finalized, changing it during implementation and post go-live might cause disruption for the end users experience. Once the decision has been made to load external data into

Dataverse, a great tool is dataflows. Dataflows are a great choice when you need to transform and prepare the data while you move it. The dataflow design environment, Power Query, has robust out-of-the-box functionality to help with intuitive transformation and data prep.

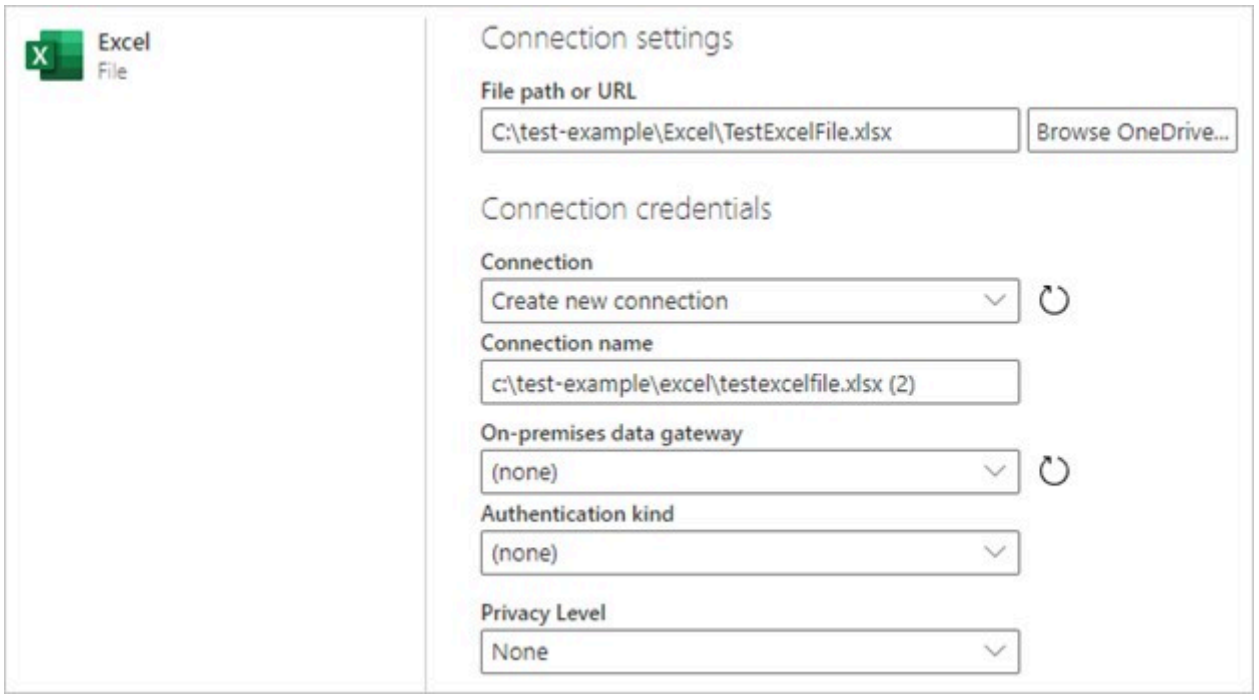
More information: Create and use dataflows in Power Apps and Add data to a table in Dataverse by using Power Query

Note that loading data via dataflows is subject to the Dataverse service protection limits. Learn more

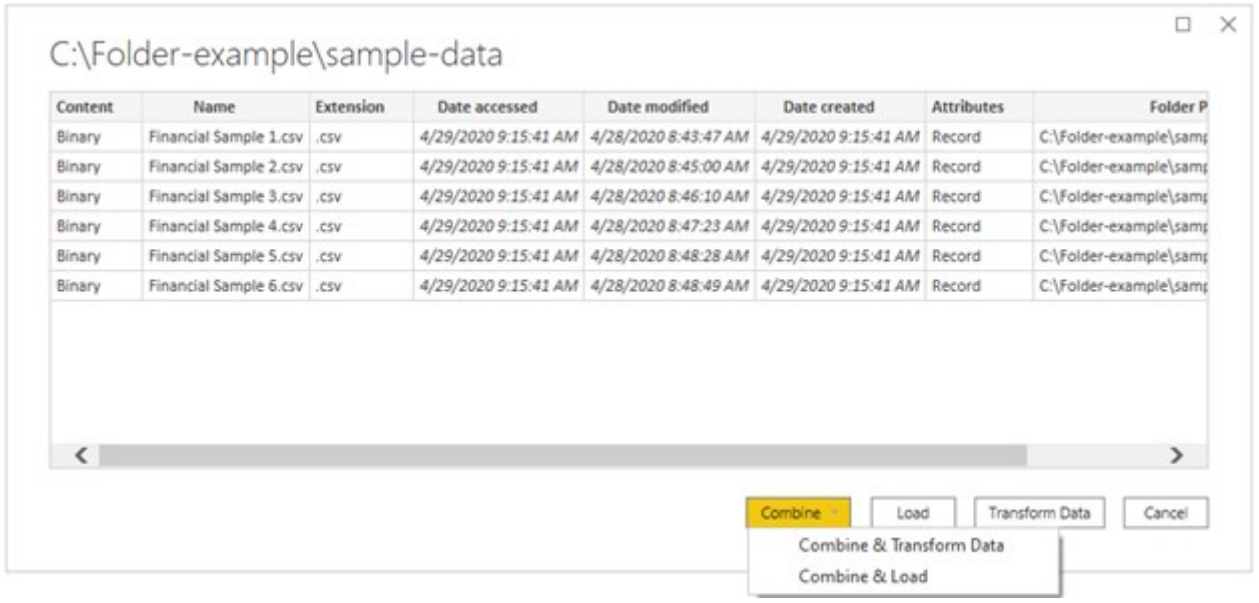


Power Query supports a multitude of available connectors as outlined here: List of all Power Query connectors

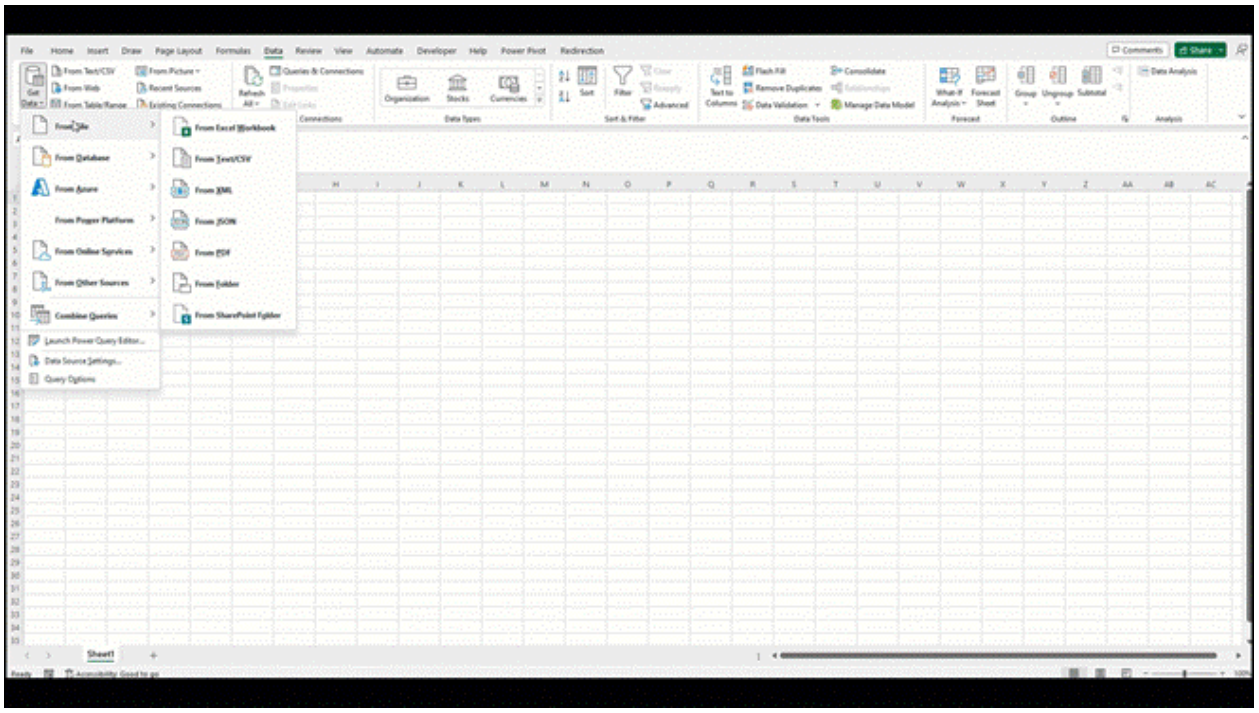
If your data originates in Excel, like that of many Dataverse customers, then use the existing Excel connector, for example, among multitude of other available connectors, to build a data pipeline into Dataverse. In addition to one-off data loads, sync your Excel data source with Dataverse using a dataflow. Consider Excel file size limitations with this guidance reduce the size of an Excel workbook to view it in Power BI



Moreover, if you have multiple Excel files in a folder that you'd like to load into Dataverse, you can use Power Query folder connector to combine, transform, and load data.



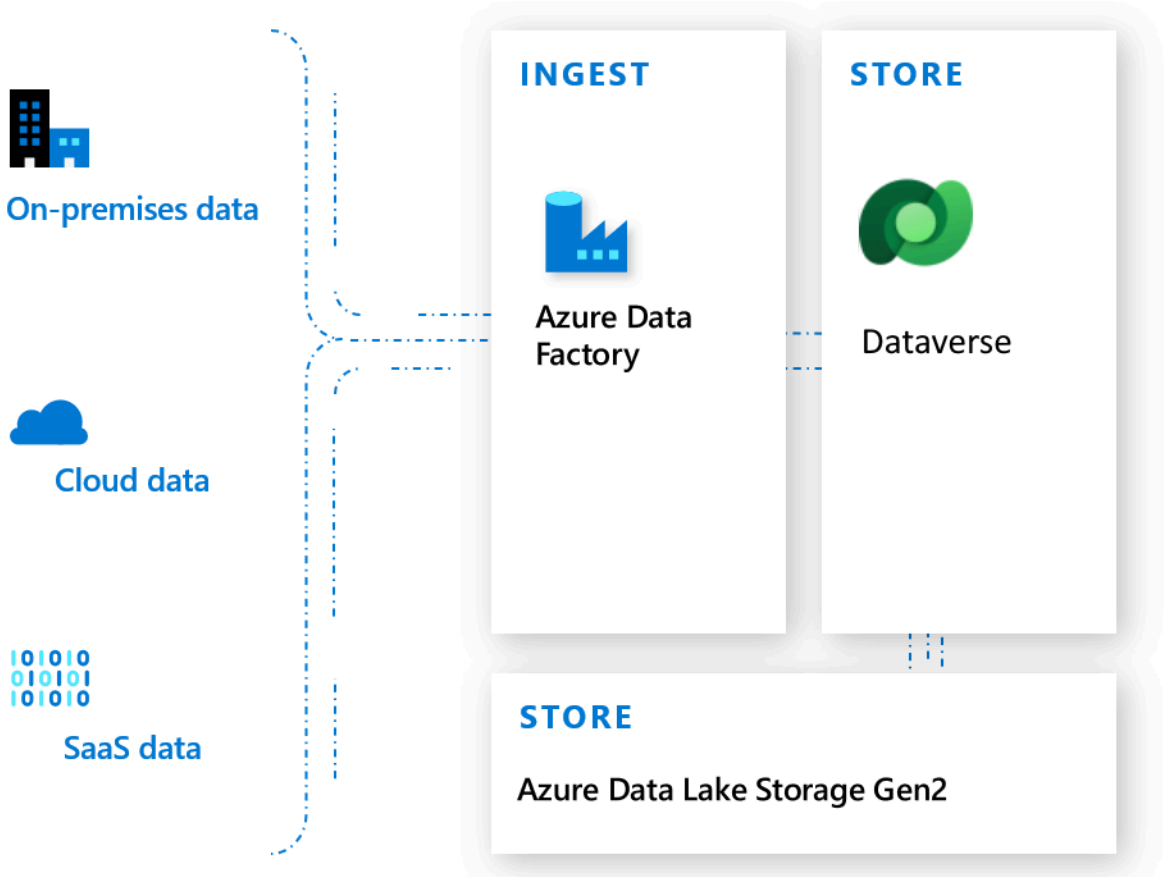
IT professionals can also free up business users from the complexities of building data pipelines by creating Power Platform dataflow templates. Build a complex query combining data from multiple sources, save it, and share with business users. This file can be then used from the **New dataflow** > **Import template** feature in Dataverse. All the user needs to do is verify credentials for data access to start loading data into Dataverse tables. Learn more Create Power Platform dataflows from queries in Microsoft Excel.



You can also perform a one-time import of data directly from a single Excel file into a Dataverse table. More information: Import data from Excel and export data to CSV

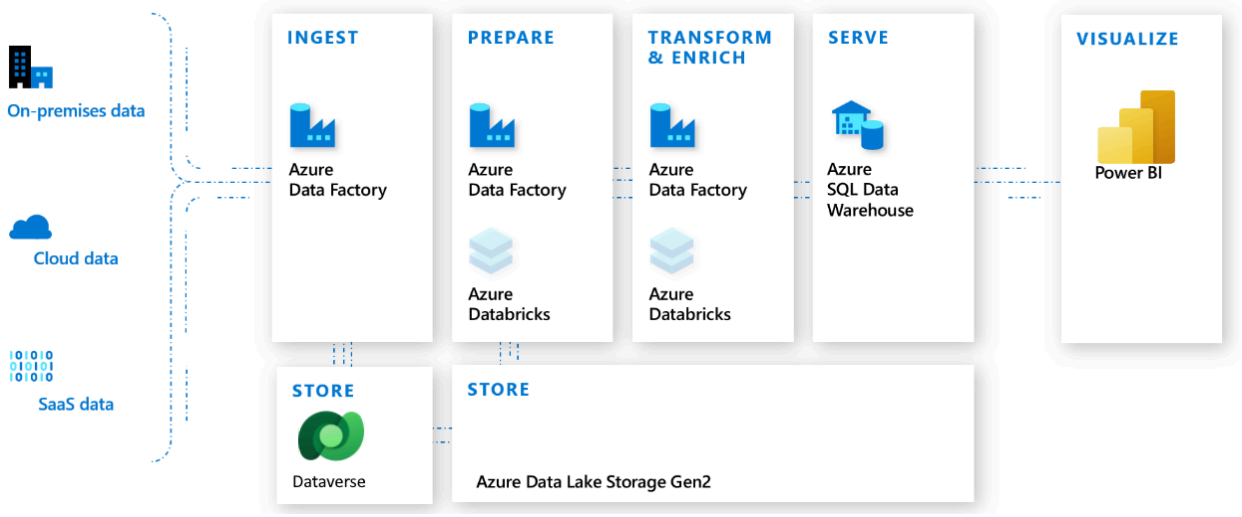
Azure Data Factory

Data Factory is a data integration service that provides a low-code or no-code approach to construct extract, transform, and load (ETL) processes within a visual environment or by writing your own code.



Depending on your data needs, at some point you might require more involved data engineering to bring the data from external sources into Dataverse. It isn't just about the data volume, but also about the amount and complexity of data transformations required for the source data to meet your data requirements in Dataverse. An additional consideration is the IT resources are available to assist with the effort. These more complex data scenarios can be addressed with Power Platform dataflows, Azure Data Factory, and Web API. >Data Factory and Web API integration patterns shine at building pipelines that require involved workflow and algorithm rich transformations. Conversely, dataflows are faster to start with and easier to use.

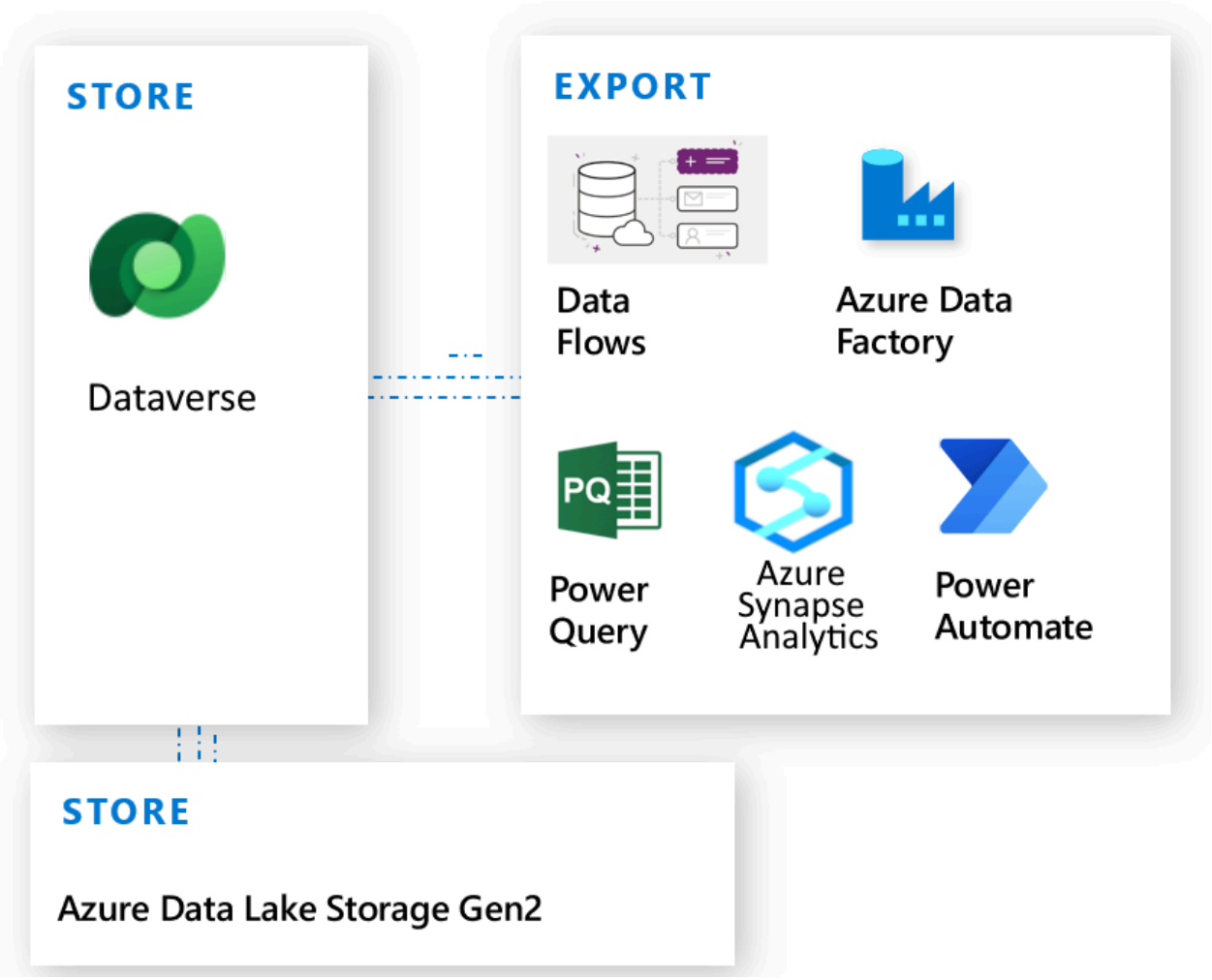
With Data Factory, you can visually integrate Dataverse and other data sources by using more than 90 natively built and maintenance-free connectors.



In addition to bringing data into Dataverse, Data Factory can also be used to prepare, transform, and enrich data with Azure Databricks and move data into Azure Synapse Analytics.

Exporting data from Dataverse

Exporting data, either to another data technology or to another environment, can use any of the same technologies mentioned for importing data, such as dataflows, Data Factory, Power Query, and Power Automate.



Dynamics customers who are targeting SQL Server or Azure SQL Database can use Azure Synapse Link. Azure Synapse Link for Dataverse connects with Azure Synapse Analytics to enable you to get near real-time insights over your data from Microsoft Dataverse. With a tight seamless integration between Dataverse and Azure Synapse Analytics, Azure Synapse Link enables you to run analytics, business intelligence and machine learning scenarios on your data. More information: What is Azure Synapse Link for Dataverse

See also

Work with any type of app

Feedback

Was this page helpful? Yes No

Provide product feedback

Work with any type of app

Article • 06/21/2022

Microsoft Dataverse provides multiple ways to integrate in any type of app (mobile, web, desktop), device, system, or service. For cloud solutions, there are ways to integrate regardless of the model in which your solution is deployed—infrastructure as a service (IaaS), platform as a service (PaaS), or software as a service (SaaS). For IaaS-based solutions, the integration approach also works well if solutions are running inside of containers.

In some cases, integration with an app can be achieved by using business logic contained in Dataverse. In other cases, it will involve integration via events, the Dataverse OData API, or using plug-ins.

Defining business logic

Tables in Dataverse can use rich server-side logic and validation to ensure data quality and reduce repetitive code in each app that creates and uses data in a table.

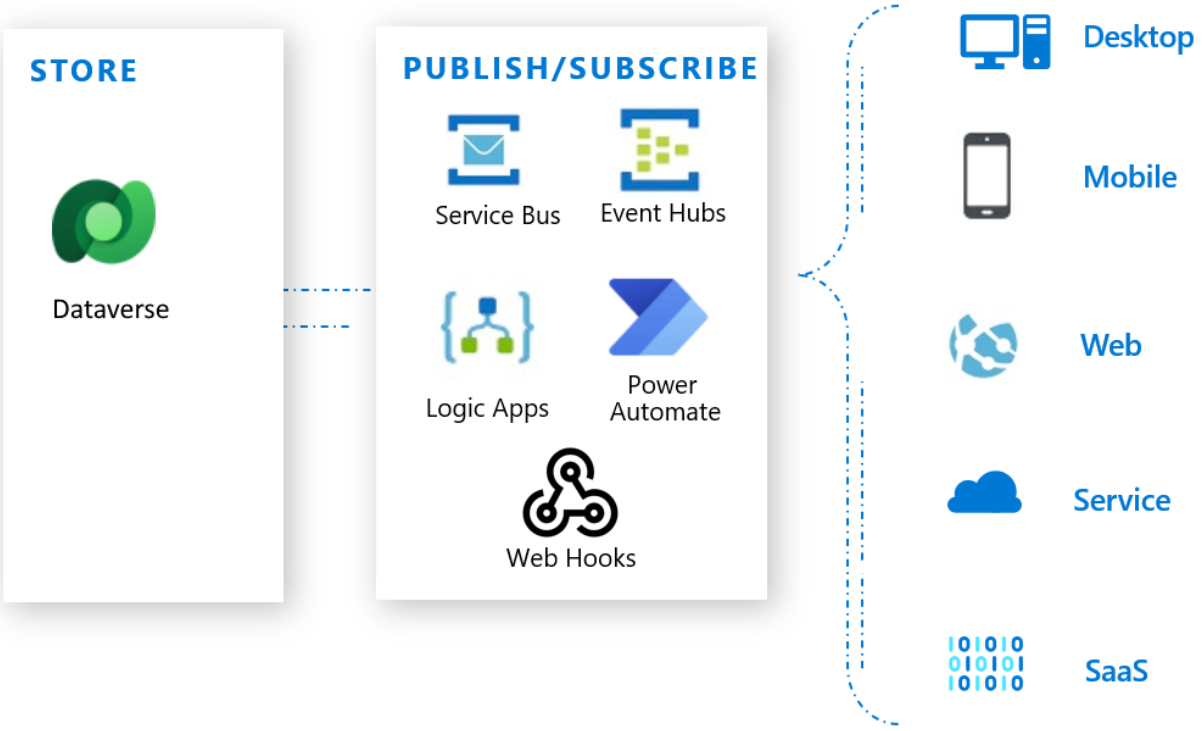
- **Business rules**: Validate data across multiple columns and tables, and provide warning and error messages, regardless of the app used to create the data. More information: Create a business rule for a table
- **Business process flows**: Guide users to ensure that they enter data consistently and follow the same steps every time. Business process flows are currently only supported for model-driven apps. More information: Business processes flows overview
- **Workflows**: Automate business processes without user interaction. More information: Classic Dataverse workflows
- **Business logic with code**: Supports advanced developer scenarios to extend the app directly through code. More information: Apply business logic using code

Integrating with apps by using events

A common approach of app integration is through the use of events. For example, an event such as adding a new row occurs in Dataverse, and this should be communicated to an associated system so that an action can be taken. For example, if a new support request was raised, it might trigger an SMS message to be sent to the assigned support staff.

This interactivity can also occur in the opposite direction—an update in an external system might result in data being added, updated, or deleted from a Power Platform environment.

The most popular approaches in Dataverse involve webhooks, Azure messaging (Service Bus, Event Hubs), Azure Logic Apps, or Power Automate.



Webhooks

With Dataverse, you can send data about events that occur on the service to a web app by using webhooks. A webhook is a lightweight HTTP pattern for connecting web APIs and services with a publish-and-subscribe model. Webhook senders notify receivers about events by making requests to receiver endpoints with some information about the events.

Webhooks enable developers and ISVs to integrate Dataverse data with their own custom code hosted on external services. By using the webhook model, you can secure your endpoint by using authentication header or query string parameter keys. This is simpler than the shared access signature authentication model used with Azure Service Bus integration.

- Webhooks can only scale to the point at which your hosted web service can handle the messages.
- Webhooks enable synchronous and asynchronous steps.

- Webhooks send POST requests with the JSON payload and can be consumed by any programming language or web app hosted anywhere.
- Webhooks can be invoked from a plug-in or custom workflow activity.

Azure Service Bus

Service Bus provides a secure and reliable communication channel between Dataverse runtime data and external, cloud-based line-of-business apps. This capability is especially useful in keeping disparate Dataverse systems or other Dataverse servers synchronized with business data changes.

The sequence of events is as follows:

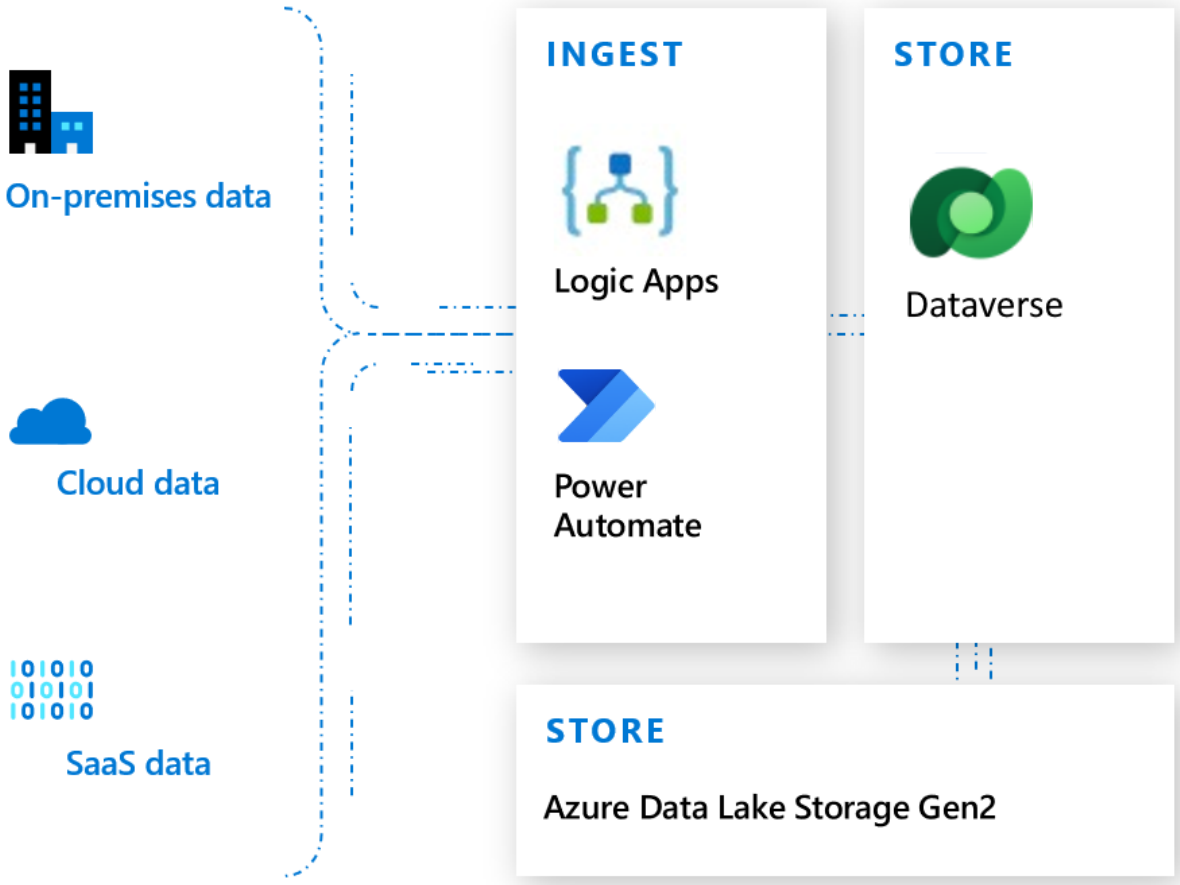
- A listener app is registered on a Service Bus solution endpoint and begins actively listening for the Dataverse remote execution context on the service bus.
- A user performs some operation in Dataverse that triggers execution of the registered out-of-the-box plug-in or a custom Azure-aware plug-in. The plug-in initiates a post, through an asynchronous service system job, of the current request data context to Service Bus.
- The claims posted by Dataverse are authenticated. Service Bus then relays the remote execution context to the listener. The listener processes the context information and performs some business-related task with that information. Service Bus notifies the asynchronous service of a successful post and sets the status of the related system job to Completed.

Service Bus relays the request message data context between Dataverse and the Service Bus solution listener apps. Service Bus also provides data security so that only authorized apps can access the posted Dynamics 365 data. Authorization of Dataverse to post the data context to Service Bus, and for listener apps to read it, is managed by Azure shared access signatures.

More information: Service Bus and Service Bus authentication and authorization

Logic Apps and Power Automate

Logic Apps, offered via Azure, and Power Automate, offered via Microsoft Power Platform, can trigger a workflow that can be used to integrate with application events and data on a schedule or by activity in a database, system, service, or SaaS.



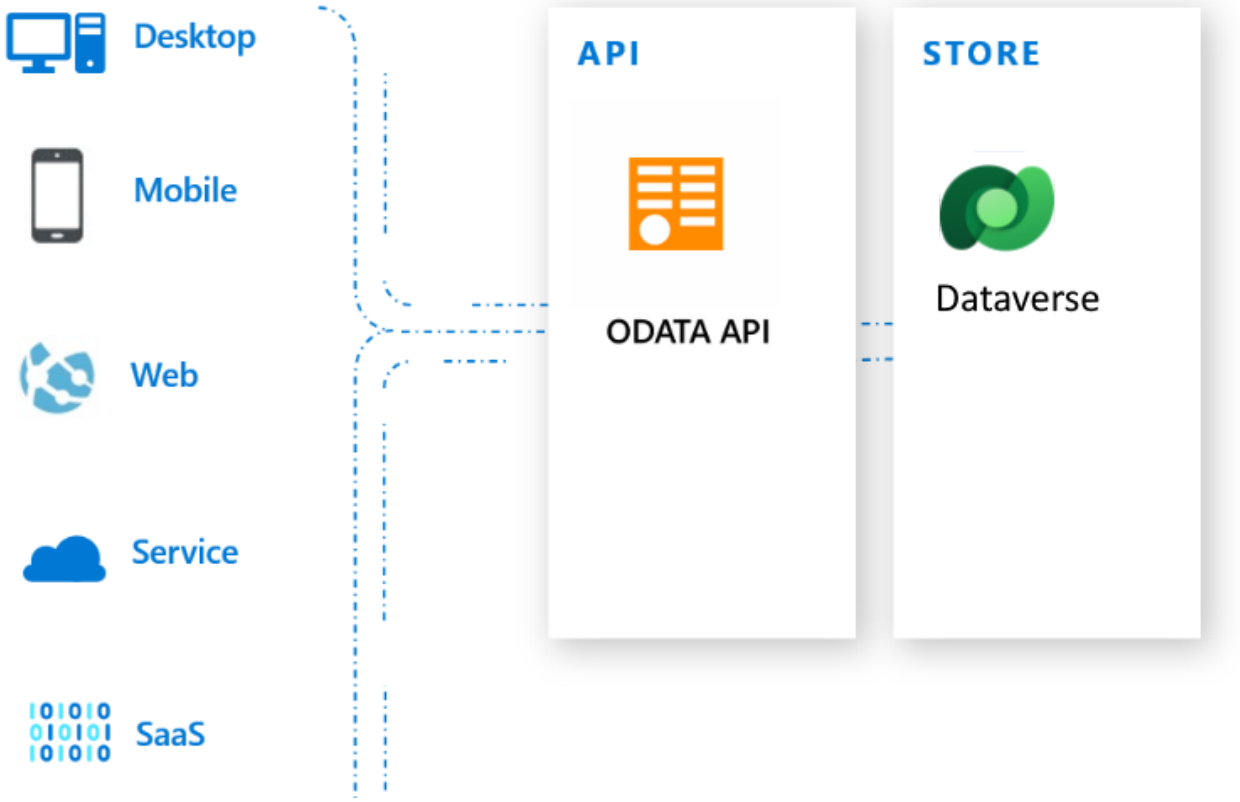
These workflows can execute logic and interact with these systems using the hundreds of connectors to databases, PaaS, and SaaS.

For example, when a row is added to a relational database, such as SQL, this might trigger a workflow that can insert this data in Dataverse.

With the ability to create custom connectors by using Open API (formerly known as Swagger) definitions for a service, it's also straightforward to include services, functions, and code running in IaaS and Azure Kubernetes Service (AKS).

Integrating Dataverse into apps with the OData API

All popular programming languages support a form of integration with REST-based APIs.



The Dataverse Web API provides a development experience that can be used across a wide variety of programming languages, platforms, and devices. The Web API implements the OData (Open Data Protocol), version 4.0, an OASIS standard for building and consuming RESTful APIs over rich data sources. You can learn more about this protocol at www.odata.org. For more information about this standard, see www.oasis-open.org.

Dataverse takes an "API first" approach. This means that the service doesn't just provide a mechanism to query data, it also provides metadata from the service on business rules, constraints, and so on, that you can use to build intelligent, responsive apps and services.

The API is secured by using OAuth. OAuth requires an `idtable` provider for authentication. For Dataverse, the `idtable` provider is Microsoft Entra ID. To authenticate with Microsoft Entra by using a Microsoft work or school account, use the Microsoft Authentication Libraries (MSAL).

For more information about getting started with the Dataverse Web API, see Use the Dataverse Web API.

For more information about using the Dataverse Web API with OAuth, see Use OAuth with Dataverse.

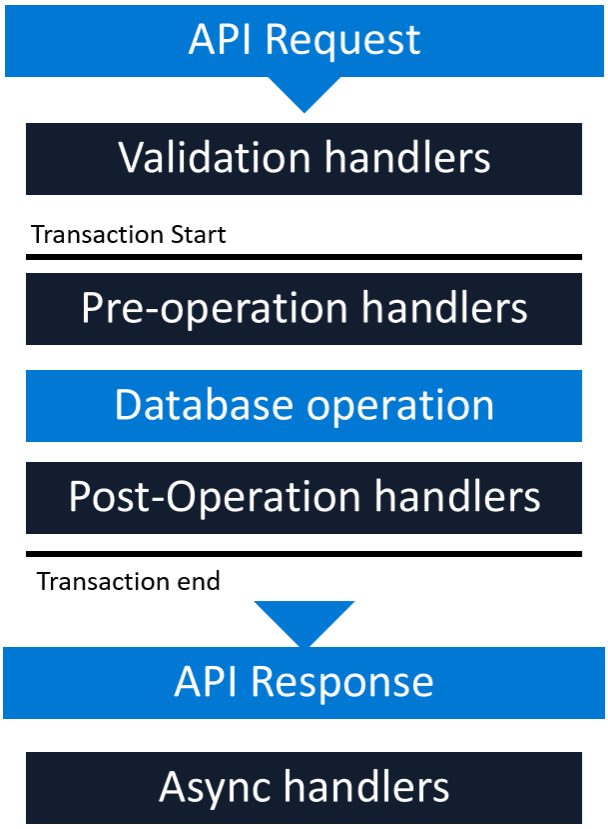
Plug-ins

The Dataverse provides the ability to write code that sits between the API and the data. This code, written in .NET, is referred to as a *plug-in*. Because the plug-in sits between the API and the data, it enforces the same logic on every app.

Plug-ins can be synchronous or asynchronous, and perform the following tasks:

- Return errors to the user.
- Query Dataverse data to evaluate logic to perform.
- Perform data operations.
- Perform outbound HTTP requests.

Plug-ins are registered at points in the event pipeline, which are illustrated here.



Within the event pipeline, the following events can occur:

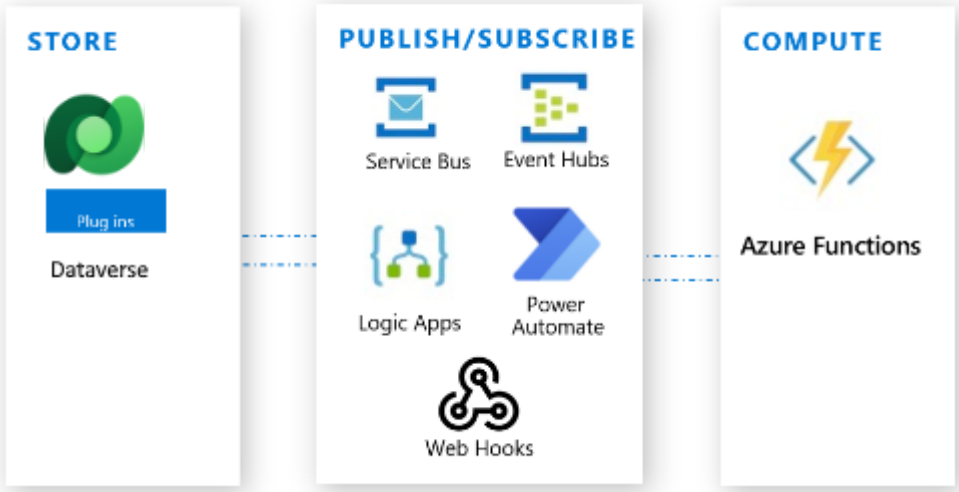
- **Requests** and **Responses** can be **examined** and **rejected** or **manipulated** in several steps of the event pipeline.
- **Validation handlers** can throw custom exceptions to reject operations that your logic considers invalid.
- **Pre-operation handlers** can modify requests before the database operation.
- **Post-operation handlers** can modify responses.

- **Async handlers** perform automation after the response is returned.

One constraint with plug-ins is that they must be self-contained. If integration code requires references to other libraries, integration can be done by using Azure Functions.

Azure Functions

Azure Functions provides a serverless code execution option for business and integration logic.



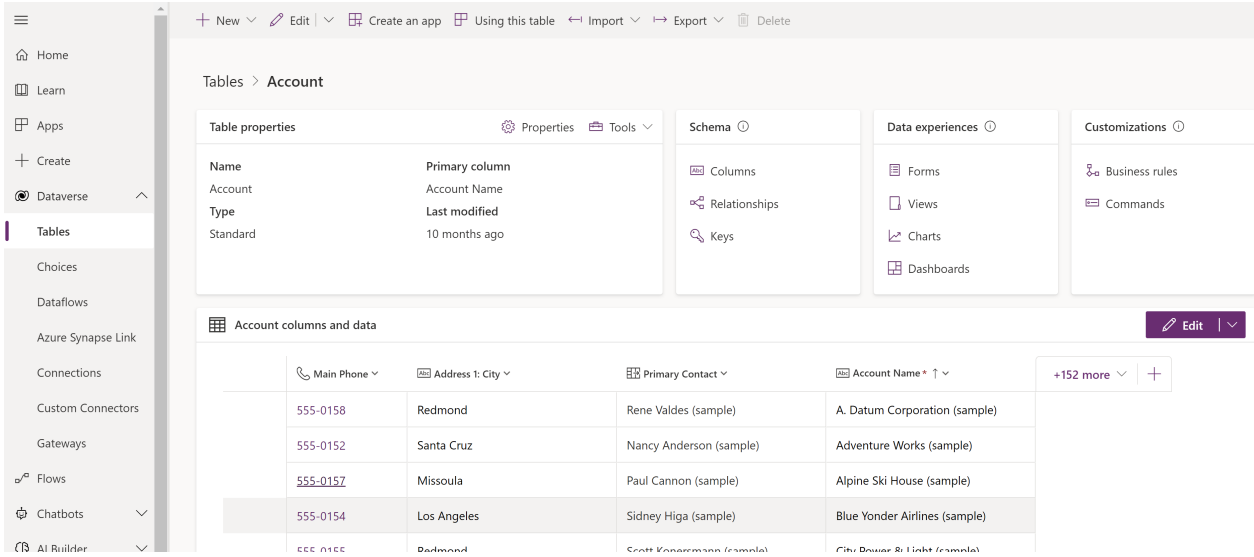
Functions are triggered by a call from an external system, service, or code. For Dataverse, that trigger can come directly from Dataverse using Service Bus, a webhook, or a call from a plug-in. Additionally, the Azure Functions call can be initiated via a flow in either Logic Apps or Power Automate that involves the Dataverse connector.

More information: Use plug-ins to extend business processes

Tables in Dataverse

Article • 06/22/2022

Tables are used to model and manage business data. When you develop an app, you can use standard tables, custom tables, or both. Microsoft Dataverse provides standard tables by default. These are designed, in accordance with best practices, to capture the most common concepts and scenarios within an organization.



See also

Types of tables

Create a custom table

Tables and metadata in Microsoft Dataverse

Microsoft Dataverse is designed so that you can quickly and easily create a data model for your application. Normally, you shouldn't have to concern yourself with some of the details about metadata that this article introduces. But if you want to develop a deeper understanding about how apps that use Dataverse work or you're evaluating what is possible, understanding the metadata used by Dataverse can provide you with insight.

Metadata means data about data. Dataverse provides a flexible platform for you because it's relatively easy to edit the definitions of the data that the environment will use. In Dataverse, the metadata is a collection of tables. Tables describe the kinds of data stored in the database. Table metadata is what controls the kinds of records you can create and what kind of actions can be performed on them. When you use the customization tools to create or edit tables, columns, and table relationships, you're editing this metadata.

Different clients people use to interact with the data in your environment depend on the table metadata and adapt as you customize the metadata. But these clients also depend on other data to control what visual elements to display, any custom logic to apply, and how to apply security. This system data is also stored within tables but the tables themselves aren't available for customization.

You can learn about standard tables, attributes, and table relationships included by default in Dataverse by reviewing the entity reference.

> **Tip**
>
> The designers available to edit metadata don't show all the details found in the metadata. You can install a model-driven app called the **Metadata Browser**, which allows you to view all the tables and metadata properties that are found in the system. More information: Browse table definitions in your environment.

Create new metadata or use existing metadata?

Dataverse comes with standard tables that support core business application capabilities. For example, data about your customers or potential customers is intended to be stored using the account or contact tables.

Each of these tables also contains many columns that represent common data that the system might need to store for the respective table.

For most organizations it's to your advantage to use the standard tables and attributes for the purposes they were provided.

If you install a solution, you can expect that the solution developer leveraged the standard tables and attributes. Creating a new custom table that replaces a system table or attribute means that any solutions available might not work for your organization.

For these reasons, we recommend that you look for and use the provided standard tables, columns, and table relationships when they make sense for your organization. If they don't make sense and can't be edited to match your need, you should evaluate if creating a new table, column, or table relationships is required.

Remember that you can change the display name of a table so that it matches the nomenclature your organization uses. For example, it's common for people to change the display name of the Account table to *Company* or the name of the Contact table to *Individual*. This can be done to tables or attributes without changing the behavior of the table. For more information about renaming tables, see Change the name of a table.

You can't delete standard tables, columns, or table relationships. They're considered part of the system solution and every Dataverse environment is expected to have them. If you want to hide a standard table, change the security role privileges for your organization to remove the read privilege for that table. This removes the table from most parts of the application. If there's a system column that you don't need, remove it from the form and any views that use it. Change the **Searchable** value in the column and table relationship definitions so that they don't appear in advanced find.

Limitations on creating metadata items

There's a limit to the number of tables you can create. Admins can view information about the number of tables and percent used towards the maximum in the legacy **Resources In Use** page. Go to Power Platform admin center (admin.powerplatform.com) select **Manage** > **Environments**, and open the environment you want. Select **Settings**, expand **Resources** select **All legacy settings**. In the legacy settings, select **Administration** > **Resources In Use**.

Within each table there's also an upper limit on the number of columns you can create. This limit is based on the technical limitations on the amount of data that can be stored in a row of a Dataverse table and isn't viewable. It's difficult to provide a specific number because each type of

column can use a different amount of space. The upper limit depends on the total space used by all the columns for the table.

Most people don't create enough custom columns to reach the limit, but if you find yourself planning to add hundreds of custom columns to a table, you should consider if this is the best design. Do all the columns you plan to add describe properties for a row for that table? Do you really expect that people using your environment will be able to manage a form that includes such a high number of columns? The number of columns you add to a form increase the amount of data that has to be transferred each time a row is edited and affects the performance of the system. Take these factors into consideration when you're adding custom columns to a table.

Choice columns provide a set of options that are displayed in a drop-down control on a form or in picklist control when using advanced find. Your environment can support thousands of options within a choice column, but you shouldn't consider this as the upper limit. Usability studies show that people have trouble using a system where a drop-down control provides large numbers of options. Use choice columns to define categories for data. Don't use choice columns to select categories that actually represent separate items of data. For example, rather than maintain a choice column that stores each of hundreds of possible manufacturers of a type of equipment, consider creating a table that stores references to each manufacturer and use a lookup column instead of a choice.

Next steps

Create or edit tables (row types)
Create and edit relationships between tables

Last updated on 04/17/2026

Types of tables

Article • 06/18/2024

A table defines information that you want to track in the form of rows (records), which typically include column data such as company name, location, products, email, phone, and so on.

Tables appear in Power Apps as one of these different types:

- **Standard**: Several standard tables, also known as out-of-box tables, are included with a Power Platform environment that includes Microsoft Dataverse. Account, business unit, contact, task, and user tables are examples of standard tables in Dataverse. Most of the standard tables included with Dataverse can be customized. Tables that are imported as part of a managed solution and set as customizable also appear as standard tables. Any user with appropriate privileges can customize these tables where the table property has customizable set to true.
- **Activity**: Are a special kind of table and are best for rows that have an activity-based element, which can include a subject, start time, stop time, due date, and duration. Dataverse already comes with several out-of-the-box activity tables, such as appointment, task, email, and phone call. More information: Activity tables
- **Virtual**: Are when you need the table to be populated with data from an external source outside of Dataverse.
- **Elastic**: Are for when the table will store a very large dataset in excess of tens of millions of rows.

Activity tables

An activity can be thought of as any action for which an entry can be made on a calendar. An activity has time dimensions (start time, stop time, due date, and duration) that help determine when the action occurred or will occur. Activities also contain data that helps determine what action the activity represents, for example, subject and description. An activity can be opened, canceled, or completed. The completed status of an activity will have several sub-status values associated with it to clarify the way that the activity was completed.

Activity tables are a special kind of table that can only be owned by a user or team, but can't be owned by an organization. When you create a table, you can specify it as a standard or activity table.

The following table lists activity tables that are available in a default environment.

Expand table

Name	Description	Display in activity menus	Reference
Appointment	Commitment representing a time interval with start/end times and duration.	Yes	Appointment
Email	Activity that is delivered using email protocols.	Yes	Email
Fax	Activity that tracks call outcome and number of pages for a fax and optionally stores an electronic copy of the document.	Yes	Fax
Letter	Activity that tracks the delivery of a letter. The activity can contain the electronic copy of the letter.	Yes	Letter
Phone Call	Activity to track a telephone call.	Yes	PhoneCall
Recurring Appointment	The master appointment of a recurring appointment series.	Yes	RecurringAppointmentMaster
Task	Generic activity representing work needed to be done.	Yes	Task

When you can create a new custom activity table, you might create it to read instant message communications. Creating an activity table is different from creating a non-activity table because you don't specify a primary column. All activity tables have a **Primary Field** set to **Subject** and other common columns that are defined by the Activity table. This allows all types of activities to be shown in a view where just the common columns are displayed.

To create a custom activity table, open the **Advanced options** section in the **New table** panel, select the **Activity** option from the **Type** drop-down list. After you select this, you'll see that **Display in Activity menus** is selected. This setting allows people to create this type of activity in the activity menus. This isn't selected for activities that are typically associated with specific events and created behind using code or by a workflow. After you save the table, you can't change these settings.

Enable activities for a table

Enable activities to add activities to a table and use the Regarding lookup for the table.

1. To enable activities, sign in to Power Apps.
2. On the left nav, select **Tables**, and then open the table you want.
3. Select **Properties**.
4. Expand **Advanced options**, and then select **Creating a new activity**.

   > **Important**
   >
   > Once enabled this setting can't be disabled.

5. Select **Save**.

Enable users to associate email to multiple records

Users sometimes need to associate an email to multiple records. For example, a customer sends an email that is related to multiple cases. The existing Regarding lookup only allows a user to relate an email to a single parent record. Email records can now be associated to multiple records such as accounts, contacts, leads, opportunities, and cases through a new activity party type column named **related**. When an email is associated to multiple records, it appears in the timeline for each of the associated records.

> **Note**
>
> This feature is currently only supported with email activities.

The related column isn't added to email form by default. To enable users to use this new column, add the **related** column to the email form. For information about how to add a column to a form, go to add columns to a form.

After you add the related column to the form, users can associate an email record with other related party records (parties) like cases or opportunities.

Differences between regarding and related columns

While there are similarities between the regarding and related columns, the following are important differences to note:

- **Supported tables**
  The related column is currently only available for the Email table. The related

column doesn't currently support the same relationships utilized by the regarding column. The list of supported tables is currently limited to the tables that can have a contact email.

- **Relationship type**
  The regarding column acts as a parental relationship from a parent record. For example, if an account has multiple activities and is later deleted, the same action is automatically applied to the child activity records. The related column isn't a parental relationship. Actions performed on related records don't affect the related activities.

- **When the column value is set**
  After the related column is added to the email form, users can manually add one or more records to indicate the records are related to the activity. The regarding and related columns are independent. Setting a value for the regarding lookup column doesn't affect the values for related. When a new email is related to an existing email (for example, as a reply), the values from the related and regarding columns of the correlated email are copied to the new activity. However, when you create a new email from the timeline control within a record's context, the related column isn't automatically set to the current record. This behavior can be modified by enabling an OrgDBOrgSetting named AddParentEntityToRelatedOnNewActivityCreate.

- **Outlook add-in scenarios**
  Some features such as Dynamics 365 App for Outlook and Copilot for Sales include an ability to save an Outlook email as an email record while also setting the regarding column. These features don't set the value of the related column.

Table ownership

There are two different types of standard and custom table ownership. When you create a custom table the ownership options are **User or team** or **Organization** owned. Once a table is created, the ownership type can't be changed.

Expand table

Ownership	Description
Organization	Data belongs to the organization. Access to the data is controlled at the organization level.
User or team	Data belongs to a user or a team. Actions that can be performed on these rows can be controlled on a user level.

Notice that there are a few Dataverse system tables that are similar to standard tables but have a different type of ownership from organization and user or team ownership:

- **None**. Some system tables don't have an owner, such as the Privilege table.
- **Business Unit**. A few system tables are business-owned. These include Business Unit, Calendar, Team, and Security Role tables.

> **Important**
>
> After a custom table is created, you can't change the ownership. Before you create a table, make sure that you choose the correct type of ownership. If you later determine that your custom table must be of a different type, you have to delete it and create a new one.

Virtual tables

A virtual table is a custom table in Dataverse that has columns containing data from an external data source. Virtual tables appear in your app to users as regular table rows, but contain data that is sourced from an external database dynamically at runtime, such as an Azure SQL Database. Rows based on virtual tables are available in all clients including custom clients developed using the Dataverse web services. More information: Create and edit virtual tables that contain data from an external data source

Elastic tables

Elastic tables offer performance benefits over standard tables when the table contains a very large dataset. Elastic tables are powered by Azure Cosmos DB. More information: Create and edit elastic tables

See also

Create tables
Edit tables

Feedback

Was this page helpful? Yes No

Provide product feedback

Create and edit tables using Power Apps

Tables are used to model and manage business data. When you develop an app, you can use standard tables, custom tables, or both. Power Apps provides an easy way to view, create, and edit tables for Microsoft Dataverse.

Prerequisites

To create and edit tables in Dataverse, you need the following:

- A Power Platform environment with Dataverse.
- Appropriate permission that includes *both* of the following privileges:
  - The system customizer security role in the environment. Note that, apart from self-created table records, the system customizer role doesn't have the privileges to access table records that aren't shared with them. More information: Environments with a Dataverse database.
  - A system or custom security role that has Create, Read, and Write privileges to the Entity table. These privileges allow the user to create and edit tables in Dataverse. Note that a Power Platform admin must grant the user data access to edit the rows of these tables.

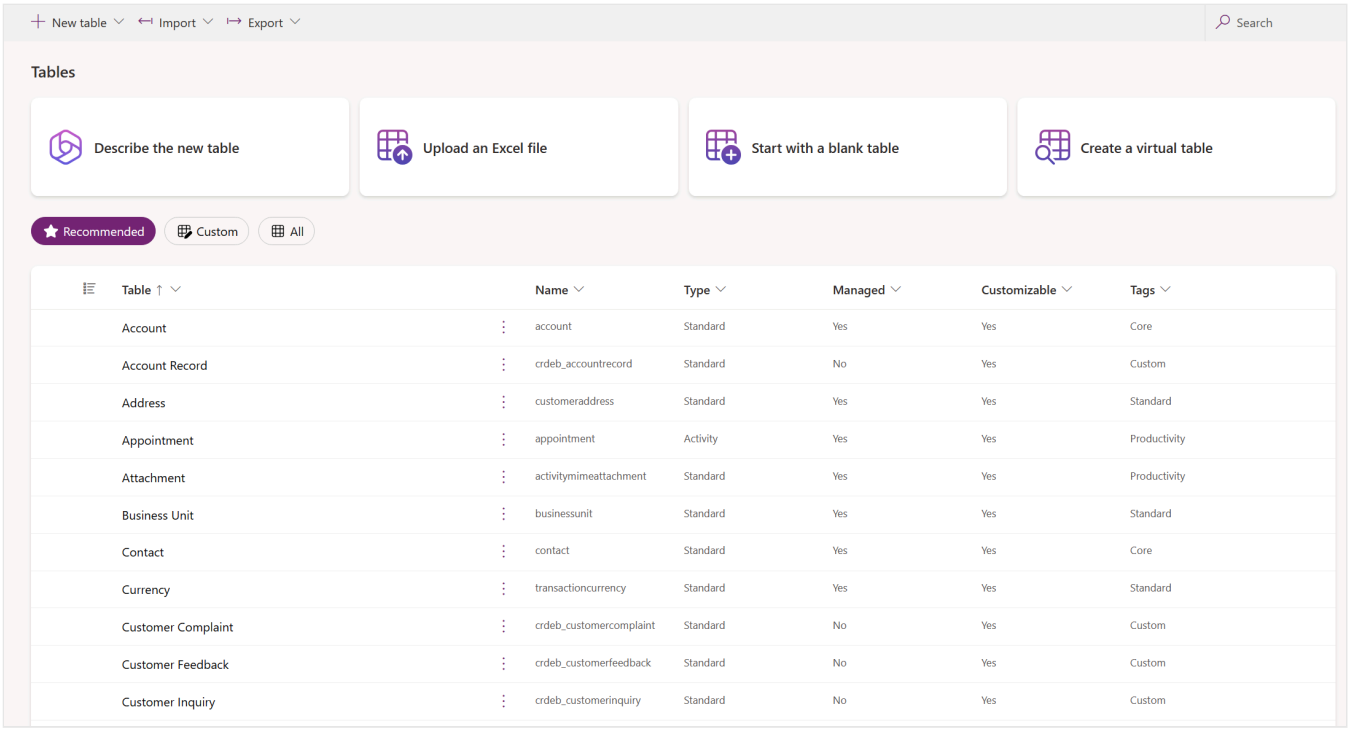
View tables

Sign into Power Apps, and then select **Tables** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

Filter the tables that are displayed using the following tabs:

Expand table

View	Description
Recommended	Displays only the standard tables. Standard tables are tables included with Power Apps or Dynamics 365 apps.
Custom	Displays only custom tables. Custom tables are created by you and other app makers.
All	Displays all the tables.



You can also select a column heading from the table view, and then select **Filter by** to display tables by a certain property, such as **Type**, **Managed**, or **Tags**.

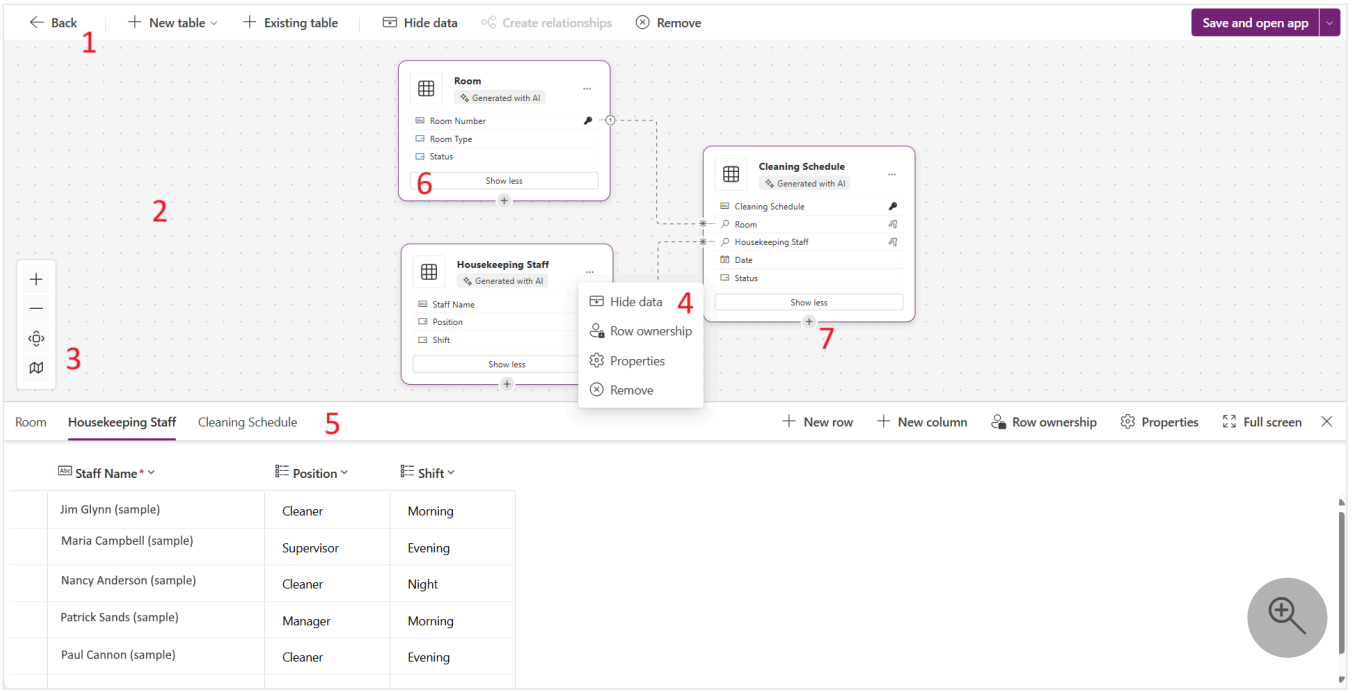
Create a table

There are several ways to create a new table:

- Create new tables
- Set advanced properties
- Create with external data
- Create a virtual table

Create new tables

Use the data workspace experience where you create tables, configure table relationships, and can view a diagram of your data.



1. Command bar:

    - **Back**. Takes you back where you were in Power Apps.
    - **New table**. Create a new table using any of the methods described in this article.
    - **Existing table**. Add one more more existing tables and visualize them in the designer.
    - **Hide data** or **View data**. Closes or displays the table row editor.
    - **Create relationships**. Creates a table relationship between two tables. Select a table on the canvas and CTRL + click another table, then select **Create relationships**. More information: Types of table relationships (NOTE: many-to-many relationships aren't supported)
    - **Remove**. Deletes the table and all associated table rows when *a new table* is selected. Remove doesn't delete the table and associated rows when an existing table is selected.

2. Data workspace canvas. On the data workspace canvas, you can drag tables around the data workspace to relocate them and select ... to invoke the table actions menu, and view a diagram of your data.

3. Data workspace actions:

    - +. Zoom in to increase the size of the table objects on the canvas.
    - -. Zoom out to decrease the size of the table objects on the canvas.
    - **Fit view**. Resize to fit within the available space in the browser tab.
    - **Show map** or **Hide map**. When **Show map** is selected, displays a compact representation of the tables on the canvas.

4. Table actions menu:

- **Hide data** or **View data**. Closes or displays the table row editor.
- **Row ownership**. Select the ownership type as either user or team, or organization.
- **Properties**. Edit the table display name, plural name, description, and schema name.
- **Remove**. Remove the table and all associated rows. If the tables is a new table, all data is deleted. For existing tables, that table is only removed from the table designer and no data is removed.

5. Table row editor. In this area, create new, edit, or delete rows, create new columns, change the table ownership type, and edit table properties.

6. Show more or less column. You can select and clear the columns to be displayed in the table card. Also displays or hides the relationship for lookup columns that connect to another table currently in the designer.

7. Add new relationship handle. Drag the handle and create a new relationship by pointing it towards another table. Some relationship types are currently not supported. More information: Limitations using the data workspace

> **Tip**
>
> Can't find the tables you created on the canvas? Select **Fit view** on the data workspace actions menu.

Create a table with the visual table designer by using any of the following methods:

- Start with Copilot
- Import with SharePoint list
- Import an Excel file or .CSV
- Start from blank

Start with Copilot

Describe your data in natural language and Copilot generates tables along with relationships for you.

> **Note**
>
> To use this feature, note the following requirements:
>
> - Copilot must be enabled for the environment. By default, Copilot is enabled. More information: Manage feature settings

- This feature is available in 20 languages including Danish, Dutch, English (United States), French, German, Italian, Japanese, Portuguese (Brazil), Spanish, Chinese (Simplified), Czech, Finnish, Greek, Korean, Norwegian (Bokmål), Polish, Russian, Swedish, Thai, and Turkish. Depending on where your environment is hosted, you might need to enable data movement across regions. For more information go to Copilots and generative AI features that are available when you enable data movement across regions.

1. From the **Choose an option to create tables** page, select **Start with Copilot**.
2. Describe the tables you'd like to create with the assistance of Copilot. You can also specify the number of tables to create, size of table, and whether to include relationship by configuring the **Table options** menu.
3. Keep iterating with Copilot in the Copilot panel until you're satisfied with your data. More information: Review the table
4. When you're finished, select **Save and exit**.

Limitations using the data workspace

The following data types aren't currently supported when you create a column in the data workspace:

- Rich text
- Customer
- Autonumber
- Formula

Some relationship configurations are currently not supported.

Expand table

Starting table	Targeting table	Relationship type	Status
New	New	One-to-many	Supported
New	New	Many-to-one	Supported
New	New	Many-to-many	Not supported
Existing	New	One-to-many	Supported
Existing	New	Many-to-one	Not supported
Existing	New	Many-to-many	Not supported

Starting table	Targeting table	Relationship type	Status
Existing	Existing	One-to-many	Not supported
Existing	Existing	Many-to-one	Not supported
Existing	Existing	Many-to-many	Not supported

Set advanced properties

From the **Tables** area, on the command bar select **New table** > **Set advanced properties**. This opens the **New table** properties panel. You start with a blank table and enter the table properties, such as name, and description. Expand **Advanced options** to set more properties, such as track changes and audit changes.

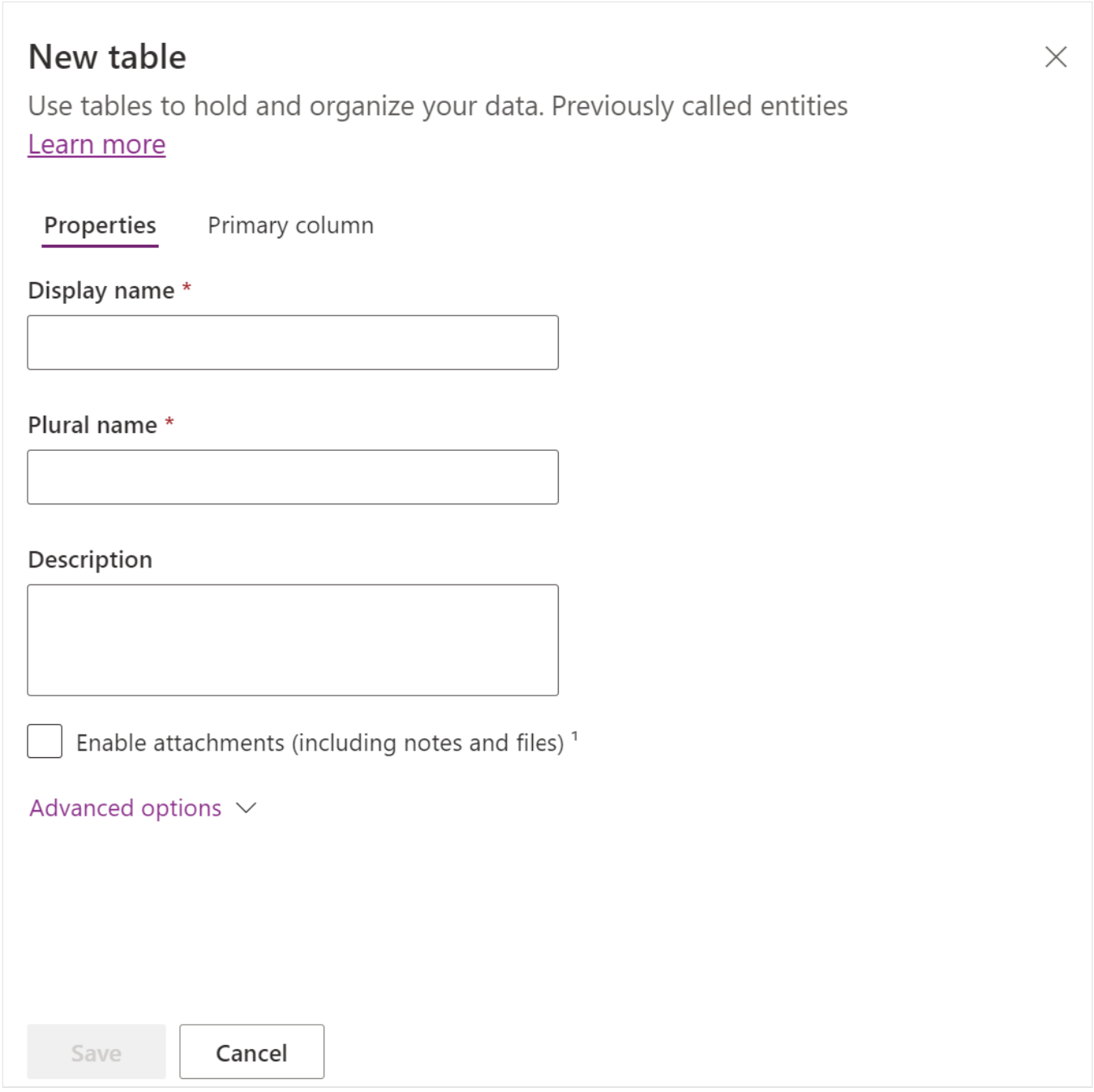


Table properties

Enter data for the following properties.

Expand table

Property	Description
Display name	This is the singular name for the table that is shown in the app. This can be changed later.
Plural display name	This is the plural name for the table that is shown in the app. This can be changed later.
Description	Provide a meaningful description of the purpose of the table.

Select **Enable Attachments** to append notes and files to records for this table.

Select the **Primary column** tab if you want to change the **Display name** or **Description** of the primary column. The primary column is used by lookup fields when establishing relationships with other tables.

> **Important**
>
> After you save the table, the **Primary column** display name and description can't be changed.

Advanced options

Select **Advanced options** to display additional properties that are optional for a table.

Expand table

Property	Description
Schema name	By default, the schema name is automatically created for you based on the display name, but you can change it. The schema name can't contain spaces and includes the customization prefix for the Dataverse solution publisher. You can't change this after the table is saved.
Type	Select the type of table. Use standard for most tables. Activity tables are a special table that can only be owned by a user or team, but can't be owned by an organization. Virtual tables require the table be populated with data from an external source. Elastic tables should be considered when your business scenario

Property	Description
	entails very large data volumes with high throughput, storage, and low latency requirements.
Record ownership	Switch the table type to Activity table to create tables that can manage tasks. The type of **Ownership** defines who can perform operations on a record.
Choose a table image	You can choose whether to display an image for the table. This image is displayed in Power Apps in some design areas. Notice that the image doesn't appear in apps using the table. To display images in apps, use the image column. More information: Image columns
Color	Set a color to be used for the table in model-driven apps.
Apply duplicate detection rules	If duplicate detection is enabled for your organization, enabling this allows you to create duplicate detection rules for this table.
Track changes	Enables data synchronization in a performant way by detecting what data has changed since the data was initially extracted or last synchronized. This option must be enabled for certain features such as Azure Synapse Link for Dataverse.
Provide custom help	When selected, set a **Help URL** to control what page users see when they select the help button in the application. Use this to provide guidance specific to your company processes for the table.
Audit changes to its data	When auditing is enabled for your environment, this allows for changes to table records to be captured over time. When you enable auditing for a table, by default, many table columns, such as data types of owner, number, text, and choice are enabled for auditing. However, some columns aren't enabled by default such as those with data types of lookup, but those columns can be enabled. Select or clear the **Enable auditing** property for the columns that you want to enable auditing on.
Leverage quick create form if available	After you've created and published a Quick Create Form for this table, people have the option to create a new record using the Create button in the navigation pane. More information: Create and design model-driven app forms When this is enabled for a custom activity table, the custom activity is visible in the group of activity entities when people use the **Create** button in the navigation pane. However, because activities don't support quick create forms, the main form is used when the custom table icon is selected.
Enable long term retention	Dataverse supports custom retention policies to securely retain unlimited data long term in a cost-efficient way. More information: Long term data retention overview
Creating a new activity	Associate activities to records for this table.
Doing a mail merge	App users can use this table with mail merge.
Setting up OneNote	When you turn on OneNote integration, you have the benefits of using OneNote to take or review customer notes from within your records. Requires SharePoint

Property	Description
integration	document management to be set up. More information: Set up OneNote integration
Setting up SharePoint document management	After other tasks have been performed to enable document management for your organization, enabling this feature allows for this table to participate in integration with SharePoint. More information: Manage your documents using SharePoint
Can have connections	Use the connections feature to show how records for this table have connections to records of other tables that also have connections enabled.
Can have a contact email	Send emails using an email address stored in one of the fields for this table. If a **Single Line of Text** column with format set to email doesn't already exist for this table, a new one is created when you enable sending email.
Have an access team	Create team templates for this table.
Can be linked to feedback	Let app users write feedback for any table record, or rate table records within a defined rating range. More information: Configure a table for feedback/ratings
Appear in search results	Enable so that table records can be included in search results when using an app.
Can be taken offline	Makes data in this table available while the Power Apps application isn't connected to a network.
Can be added to a queue	Use the table with queues. Queues improve routing and sharing of work by making records for this table available in a central place that everyone can access.

Select **Save** to continue. This action closes the **New table** panel and displays the table hub.

Create with external data

Use an Excel file/CSV file or SharePoint list to populate a table with your data, which uses copilot to assist with the table generation.

> **Note**
>
> - Generally available copilot features are enabled by default and can't be turned off. To disable them, a tenant admin must contact support.
> - When you use create with external data, such as data from Excel or SharePoint, Power Apps initially displays only the first 20 rows of your dataset as a preview. The full dataset is ingested and made available after you save the table.

1. From the **Tables** area, on the command bar select **New table** > **Create with external data**, and then select either **File (Excel, .CSV)** or **SharePoint list**.

File (Excel, .CSV)

a. Select from device or drag and drop your Excel file onto the **Upload an Excel file** page.
b. The data from the Excel file is displayed as a Dataverse table. Select a column header > **Edit column** to make changes, such as the column name or data type.
c. When you're finished, select **Create**.

For more information about how AI is used with this feature, go to FAQ for Excel to table and app.

SharePoint columns not used in Dataverse table generation

The following columns aren't included when generating a Dataverse table from a SharePoint list because the respective data types aren't supported with Dataverse:

- Image
- Task outcome
- External data
- Managed metadata
- Attachment (single)
- Multiple attachments / images
- SharePoint list system columns
- Symbol at column level (currency, prefix, postfix) in numbers
- Unique values

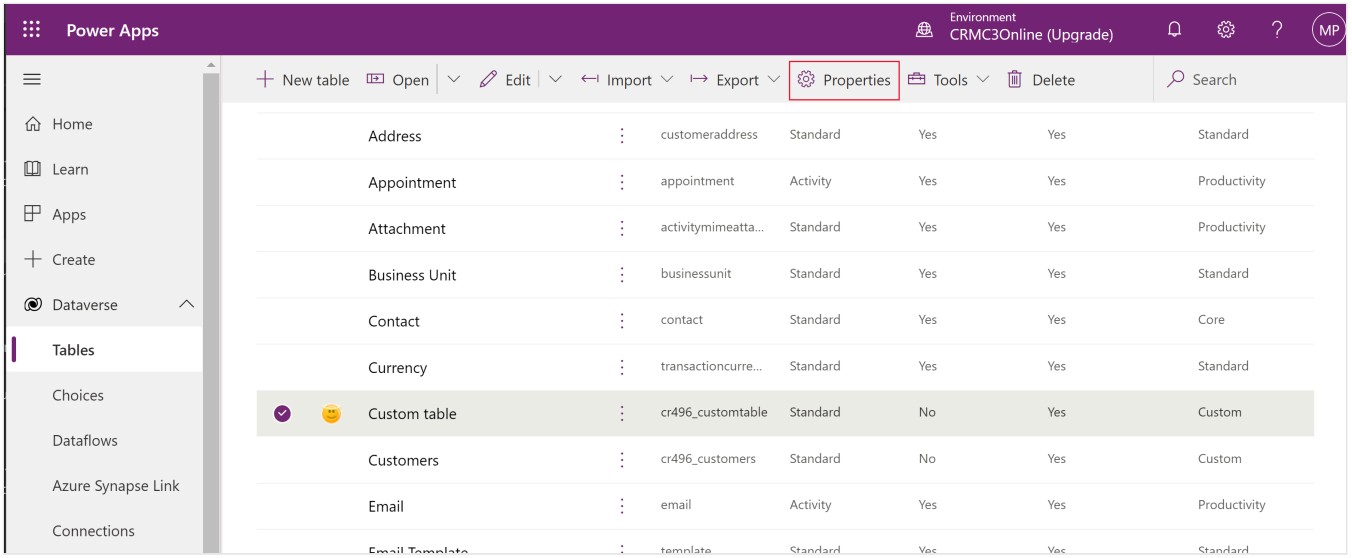
Create a virtual table

A virtual table is a custom table in Microsoft Dataverse that has columns containing data from an external data source, such as Azure SQL Database or SharePoint.

1. From the **Tables** area, on the command bar select **New table** > **Create a virtual table**.
2. Follow the **New table from external data** wizard to create the virtual table. More information: Create the virtual table

Edit a table

While viewing tables, select the table you want to edit, and then select **Properties** from the command bar if you want to edit the table properties.



Edit table components using the table hub

To edit form components, open the table to display the table hub. The table hub displays the table components described in the following sections.

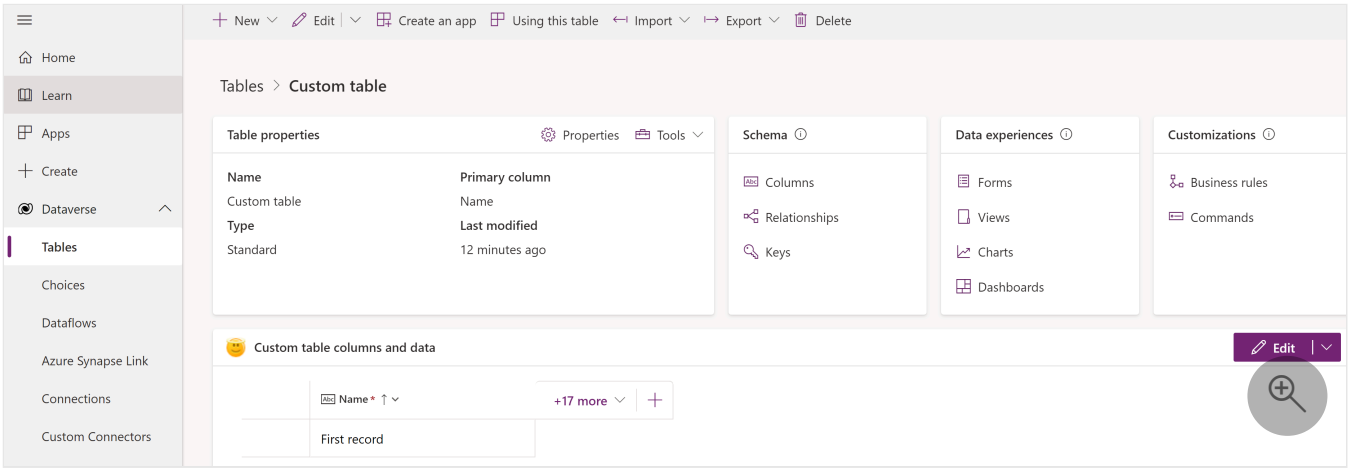


Table properties

Displays a few common properties for the table. Select **Properties** on the command bar to edit the table properties.

Schema

From the Schema area, select from the following table components to open the area where you can view and open existing components or create a new one.

- Columns. More information: Create and edit columns

- Relationships. See Create and edit relationships between tables
- Keys. More information: Define alternate keys to reference rows

Data experiences

From the **Data experiences** area select from the following table components to open the area where you can view and open existing components or create a new one.

- Forms. More information: Create and design forms
- Views. More information: Create or edit a view
- Charts. More information: Create a system chart
- Dashboards. More information: Create or edit dashboards

Customizations

From the Customizations area, select from the following table components to open the area where you can view and open existing components or create a new one.

- Business rules. More information: Create business rules and recommendations to apply logic in a form
- Commands. More information: Customize the command bar using command designer

Table columns and data

View and create table record data for the table. Select the number of columns, such as **+17 more**, to select columns to display in the columns and data view.

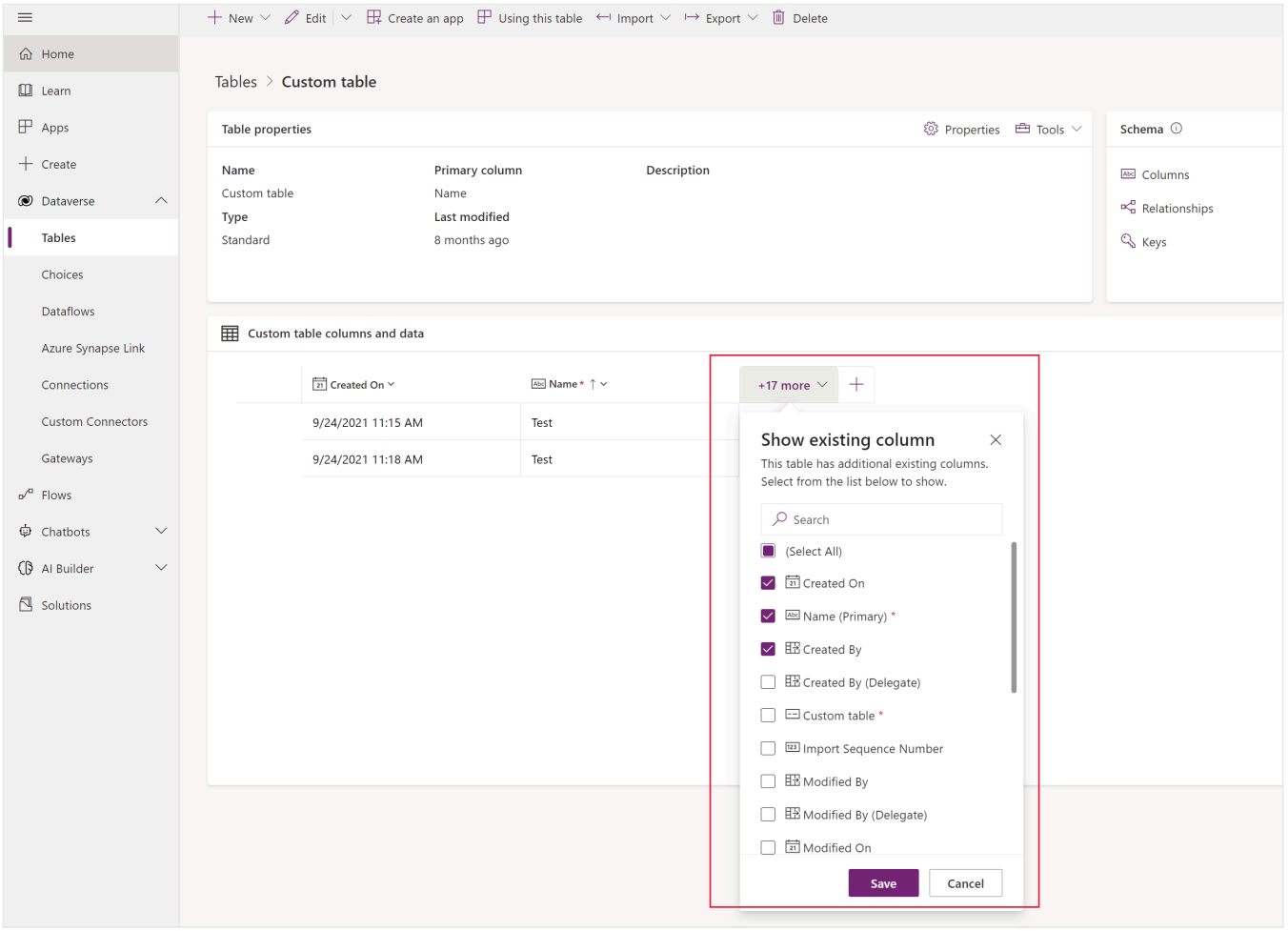
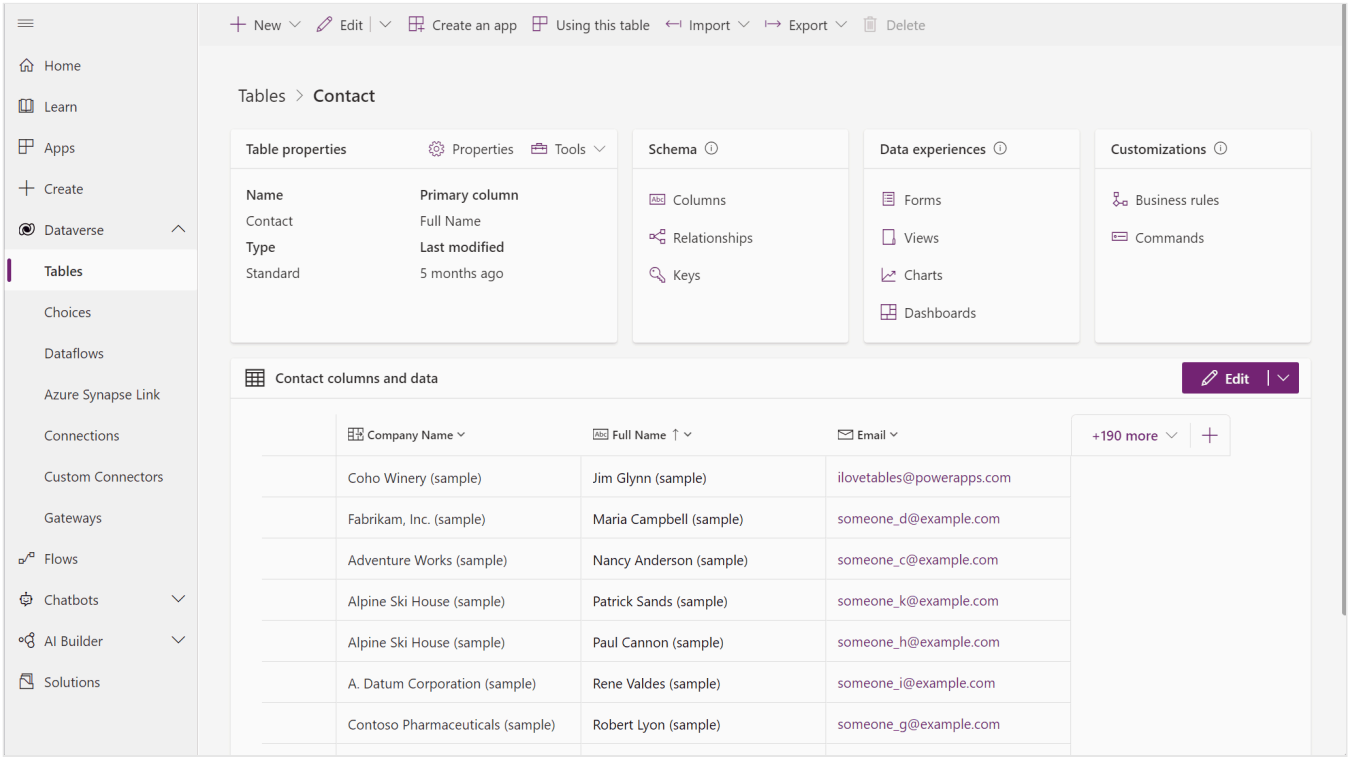


Table designer

From the table hub, select **Edit** to open the table designer. The table designer lets you make extensive changes to a table including editing or adding new records and columns, editing table properties, or creating a model-driven app based on the table.



Update forms and views

From the table hub, select **Update forms and views** to add selected columns to forms and views in this table without having to edit them in the form and view designers. More information: Update forms and views using table designer

Delete a table

As someone with the system administrator security role, you can delete custom tables that aren't part of a managed solution.

> ⚠ **Warning**
>
> When you delete a custom table, the database tables that store data for that table are deleted and all data they contain is lost. Any associated records that have a parental relationship to the custom table are also deleted. For more information about parental relationships, see **Create and edit relationships between tables**.
>
> The only way to recover data from a table that was deleted is to restore the database from a point before the table was deleted. More information: **Backup and restore instances**

While viewing tables, select the table, and then select **Delete** from the menu.

If the table has dependencies that prevent it from being deleted you see an error message. To identify and remove any dependencies, you need to use the solution explorer. More information Identify table dependencies

See also

Build apps through conversation (preview)

Create a custom table using code

Last updated on 01/09/2026

Create a custom table that has components in Power Apps

Article • 04/12/2024

With Power Apps, you tailor your app to closely fit your organization's industry, nomenclature, and unique business processes. Power Apps app development includes adding standard "out-of-box" tables or creating custom tables. A table defines the information you want to track in the form of records, which typically include properties such as company name, location, products, email, and phone.

In this article you create a table and then add or customize key components such as columns, relationships, views, and forms. You learn how to:

- Create a custom table.
- Add custom columns to your table.
- Add a table relationship.
- Customize a view.
- Customize a form.

The article will follow the company, Contoso, which is a pet grooming business that grooms dogs and cats. Contoso needs an app for client and pet tracking that can be used by employees across a variety of devices.

Prerequisites

Sign in to Power Apps. If you don't already have a Power Apps account, select the **Get started free** link from powerapps.com.

Create a custom table

1. On the left navigation pane select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. On the command bar select **New table** > **Set advanced properties**.

3. In the right pane, enter the following values, and then select **Create**.

    - **Display name**: *Pet*
    - **Description**: *Custom table to track pet services*

Add and customize columns

1. Open the **Pet** table that was created in the previous section.

2. Select **Columns**, and select the **Pet** column.

3. In the right pane make the following changes to the **Display name** column:

    - Change the **Display name** from **Pet** to *Pet Name*.

    - Make sure **Searchable** is selected.

Pet Name

Display name *

Pet Name

Name *

crfb6_PetId

Data type *

Unique Identifier

Required *

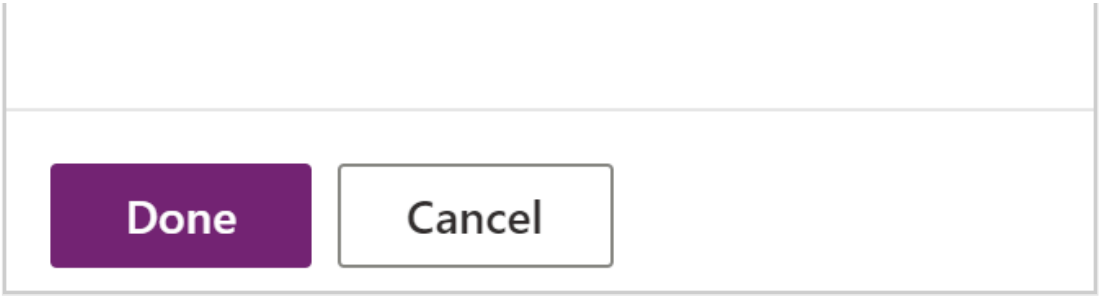
Required

Searchable

Description

Unique identifier for entity instances

Advanced options



4. Select **Done**.

5. On the **Columns** area on the table designer toolbar select **Add column**. On the **Column properties** pane, enter or select the following values and options.

    - **Display name**. *Species*
    - **Data type**. *Choice*
    - **Choice**. *New choice*

6. Select **View more**, and then select **Local choice**.

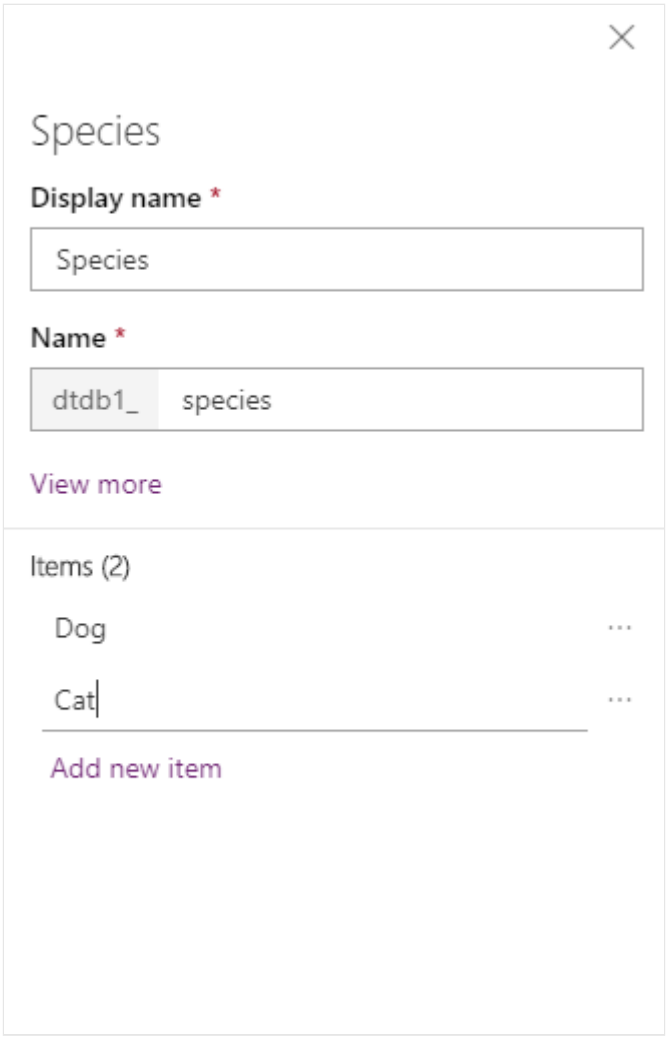
7. Create the choice:

    a. Replace **New option** with *Dog*.

    b. Select **Add new item**.

    c. Replace **New option** with *Cat*.

    d. Select **Done**.



8. Make sure **Searchable** is selected, and then select **Done**.
9. On the table designer toolbar select **Add column**. On the **Column properties** pane, enter or select the following values:
    - **Display name**. *Breed*
    - **Data type**. *Text*
10. Make sure **Searchable** is selected, and then select **Done**.
11. On the table designer toolbar select **Add column**.
12. On the **Column properties** pane, enter or select the following values, and then select **Done**.
    - **Display name**. *Appointment date*
    - **Data type**. *Date and time*

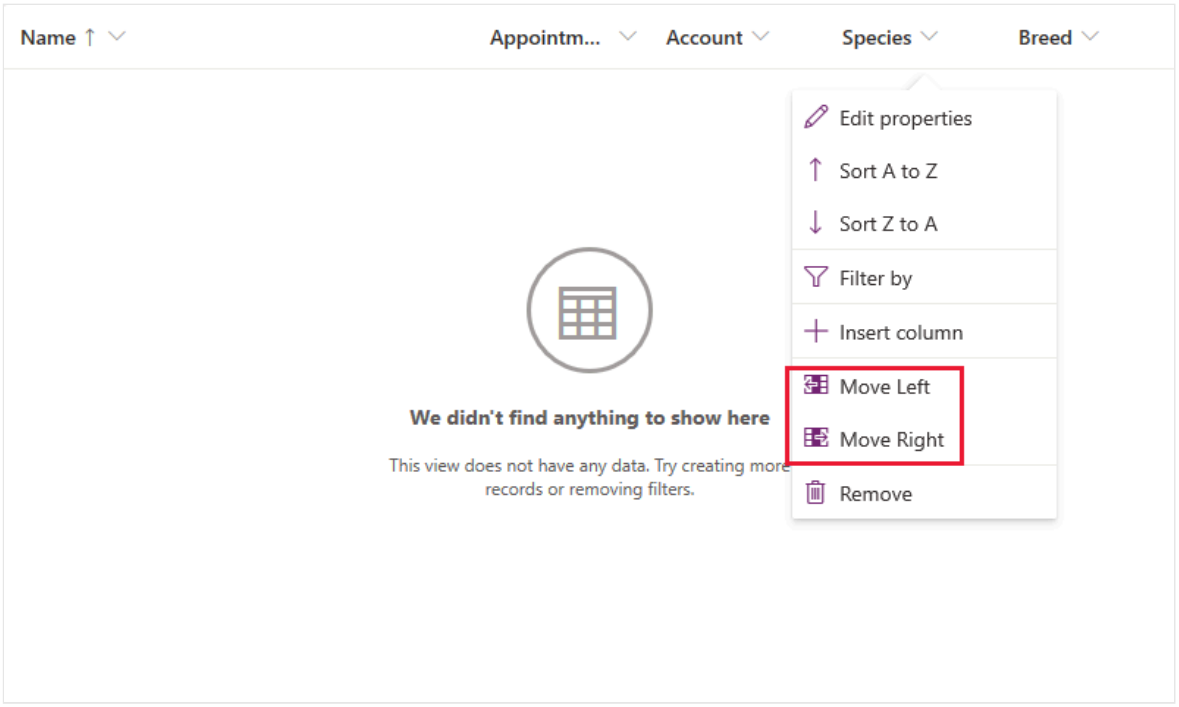
Add a relationship

1. Select the **Relationships** area. Then, on the table designer toolbar, select **Add relationship**, and then select **Many-to-one**.
2. On the right pane, in the **Related** list select **Account**.
3. Select **Done**.
4. Select **Save table**.

Notice that when you add a many-to-one relationship, an **Account** column with the data type **Lookup** is automatically added to your list of columns in the **Columns** area.

Customize a view

1. Select the **Views** area, and then open the **Active Pets** view. If you don't see the **Active Pets** view, change the filter on the command bar from **Default** to **All**.
2. On the view designer **Table columns** left pane, select the following columns to add them to the view.
    - **Account**
    - **Appointment date**
    - **Breed**
    - **Species**
3. On the view designer canvas, select the **Created On** column, and then select **Remove**.
4. To arrange the columns, on the view designer canvas, select the column you want to move and then use **Move Left** or **Move Right**. Your view should look like this.

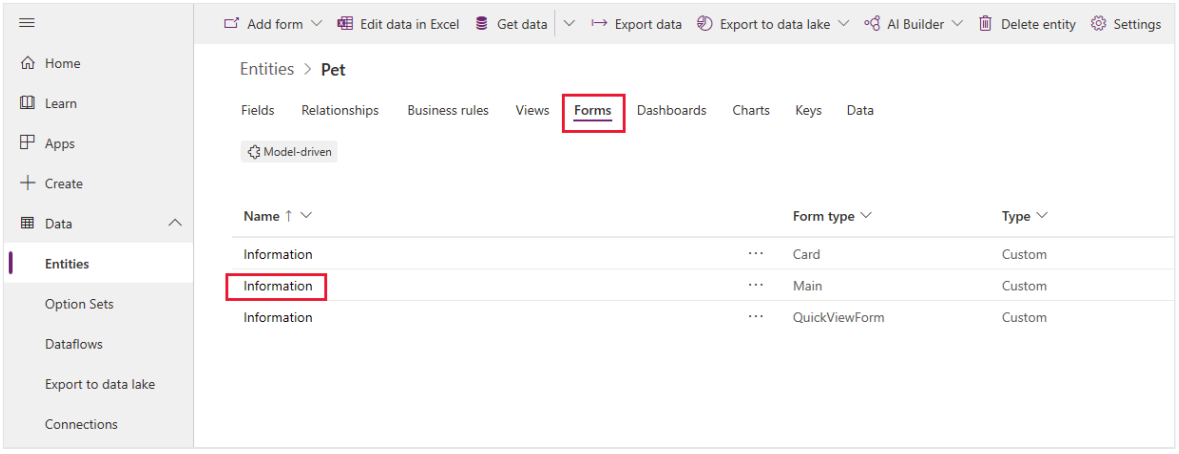


5. On the view designer toolbar, select **Save**, and then select **Publish**.

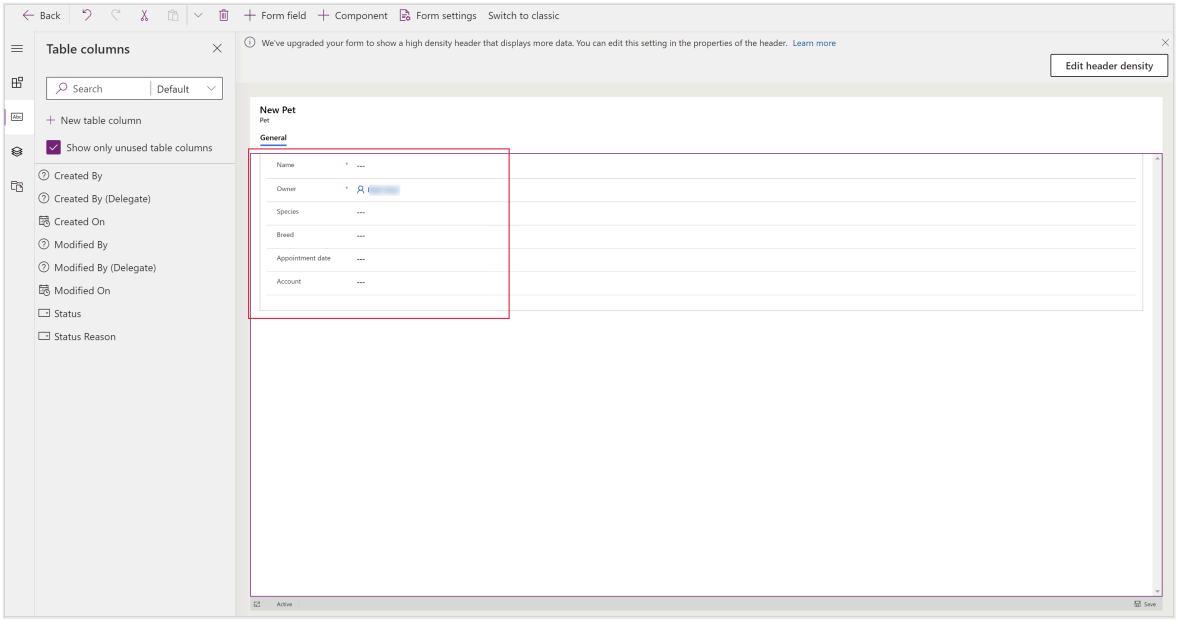
Model-driven apps only: Customize the main form

Skip this step if you only want to use the Pet table in a canvas app.

1. Select **Back** to close the view designer and return to Power Apps.
2. On the left navigation pane, select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Pet**.
4. In the **Pet** table, select the **Forms** area, and then select **Information** next to the **Main** form type to open the form designer.



5. On the form editor, select the **Table columns** on the left pane. Then, drag and drop the **Species**, **Breed**, **Appointment date**, and **Account** columns located on the pane on to the **General** section of the form designer canvas until the form looks like this.



6. Select **Save**.

7. Select **Publish**.

8. Select **Back** to return to Power Apps.

Add the custom table to an app

Now your table is ready to be used to build either a canvas or model-driven app.

Next steps

In this topic, you learned how to create a table that can be used to create a useful app.

- To learn how to create a model-driven app, see Build your first model-driven app.
- To learn how to create a canvas app, see Create an app from scratch.

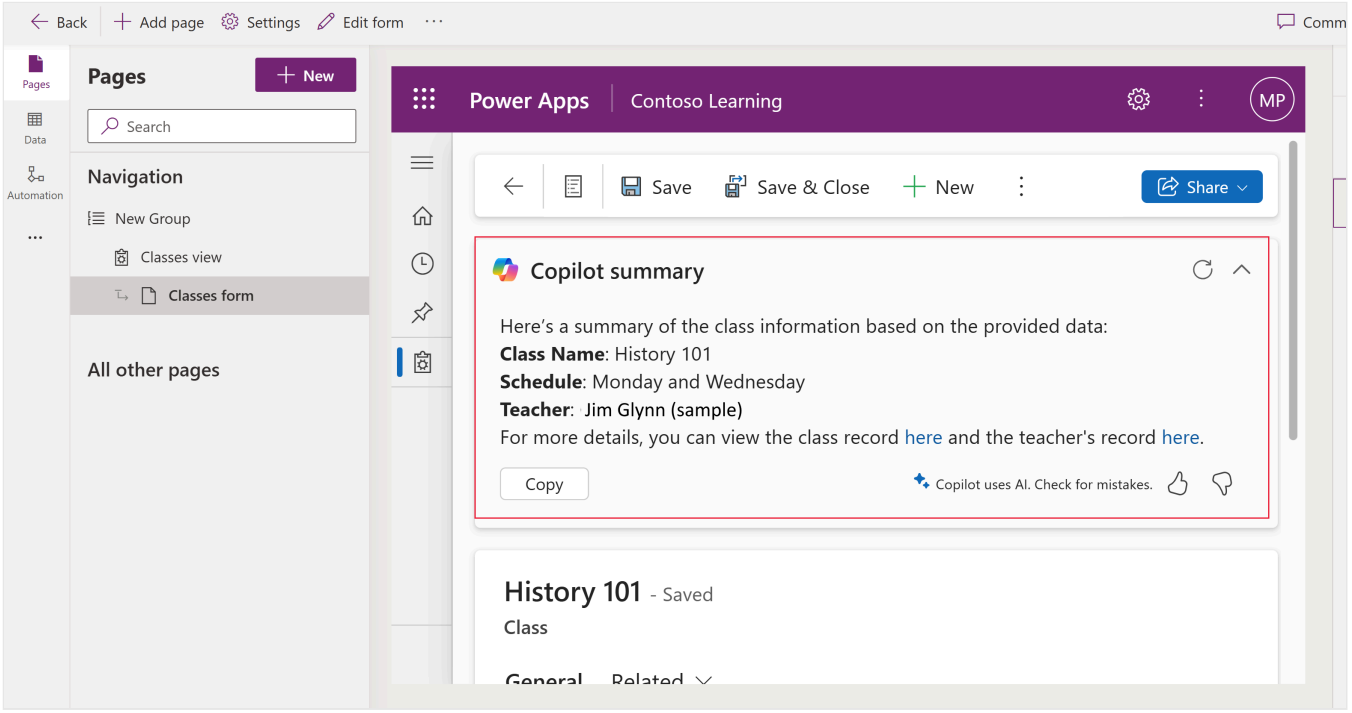
Configure a row summary for a model-driven app (preview)

Summarize this article for me

[This topic is pre-release documentation and is subject to change.]

Write a custom prompt to specify which columns should be included in a Copilot-generated row summary. These summaries appear in a consistent, dedicated experience, either surfaced in a collapsible summary bar at the top of a main form or accessed directly for individual rows from views, giving users quick access to the most important information for a record.

Records are often comprised of dozens of fields spread across multiple tables, views, forms, tabs, and sections, making it time-consuming for users to locate and understand the information that matters most. By highlighting key fields and insights in an at-a-glance summary, makers can help users quickly orient themselves, regardless of where they encounter the record.



Important

- This is a preview feature.
- Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Prerequisite for configuring a row summary

To enable and use this feature, the following criterion is required:

- A Power Platform administrator has enabled the AI insight cards environment setting. More information: Manage feature settings

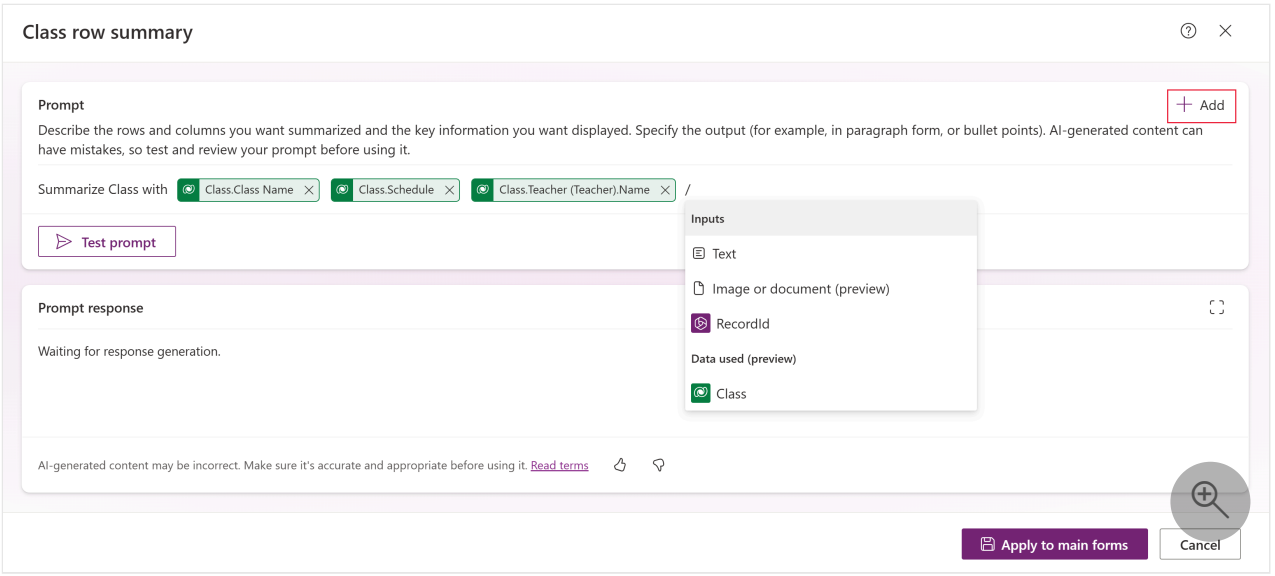
Create a row summary

Row summaries can be configured for most tables and appear on main forms and views.

> **Note**
>
> - When you configure the row summary, it applies to all views and main forms for the table.
> - Some Dynamics 365 apps tables, such as Case, Lead, and Opportunity, provide their own summaries via the Dynamics 365 Customer Service and Dynamics 365 Sales applications. To avoid conflicts, the row summary feature isn't available for these tables. You can find more information about the summaries for these tables and the options for configuring or disabling these summaries by reading about the Customer Service summary and Sales summary features. These out-of-the-box summaries are currently only accessible from main forms (not views).

1. Sign in to Power Apps (make.powerapps.com) select **Tables** on the left navigation pane, and then open the table where you want to configure a row summary. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Under **Customizations**, select **Row summary**.
    - If **Row summary** is disabled, hover over the words to find out the reason. For example, the table must have at least one row of data for the summary option to be enabled.
3. In the **Prompt** box, add the columns that you want included in the summary by selecting **Add** or by typing /. You can also specify formatting for the summary, such as make it a bulleted list or a paragraph, as well as how to ensure the summary is generated in the

user's preferred language. More information: Write a good prompt for the row summary



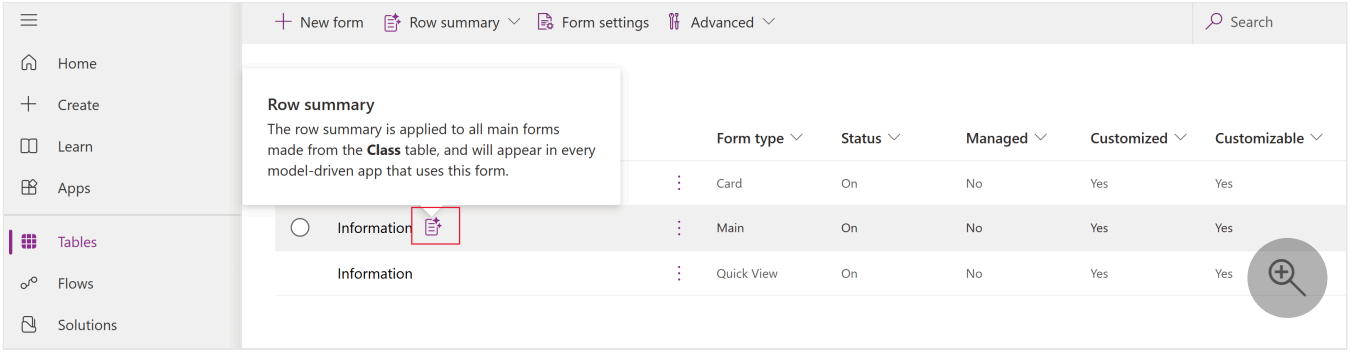
4. Select **Test prompt** to display a preview of the summary. The most recently edited row in the table is used to generate a test response.
5. Once you're satisfied with the columns and response from the test, select **Apply to main forms**.

> **Important**
>
> Any user who needs to view summaries configured through the Dynamics 365 Customer Service app must be assigned the `prvReadLanguageLocale` privilege through a security role.

Determine which main forms include a row summary

After you apply a row summary, the summary displays on all main forms for the table. To view the forms that include a row summary, in Power Apps, select **Tables** on the left navigation pane, and then under **Data experiences** select **Forms**.

All forms with the row summary applied have a form AI icon next to the form name. Hovering over the icon displays "The row summary is applied to all main from for the *table name* table, and appears in every model-driven app that uses this form."



Write a good prompt for the row summary

Writing a custom prompt gives you the ability to instruct the AI model to perform a specific task. By carefully crafting a prompt, you can generate a response that suits your specific business need. Here are some tips to consider when writing a prompt:

- Provide a list of the columns you want to include in the summary. Alternatively, you might wish to provide a list of columns that you want to exclude from the summary.
- Specify any formatting preferences, such as write the summary as a bulleted list.
- Include the appropriate input and instruction if your users work in multiple languages and you want the summary to respect the user's preferred language; otherwise, the summary will be generated in the language of the summary prompt itself, regardless of the user's language settings. To configure:
  - Include the **LanguageCode** input in the prompt.
  - Add the instruction to the prompt: **"You must respond in language <LanguageCode>."**
  - If Copilot doesn't support the user's language, the summary is generated in English.
  - This language instruction is added by default for newly configured summaries, but must be *manually added for any pre-existing summaries* that were created before this behavior was introduced.
- For information about how to craft effective prompts, download the AI Builder prompt engineering guide.

Prompt example for a row summary

Here are examples to use when building a row summary prompt.

Summarize Account record in a way that is easy to understand for a sales manager. Make sure to include all important information, including main ideas and important details, while keeping the order of the content logical. Remove any repetitive elements to make the summary as concise as possible without losing the original text's integrity. If the original text is too short to condense, present it as the summary.

Don't include the title in the summary. Create the summary in two paragraphs. For the first paragraph, use the instructions under the header Data. For the second paragraph, use the data under the header Activity. Leave a blank line between the two paragraphs. Don't include the titles Data & Activity in the paragraphs.

Data:

Show the below values as bullet points using markdown text and bold the values. Make the primary contact a navigation link to the contact record. The link must not include query string

parameters in it.

Account.Annual Revenue, Account.Category, Account.Industry, Account.Ownership

Activity:

Generate a summary paragraph using the following customer communication fields identifying top 2 key information and top 2 important actions to take.

Account.Regarding (Phone Call).Subject, Account.Regarding (Email).Subject

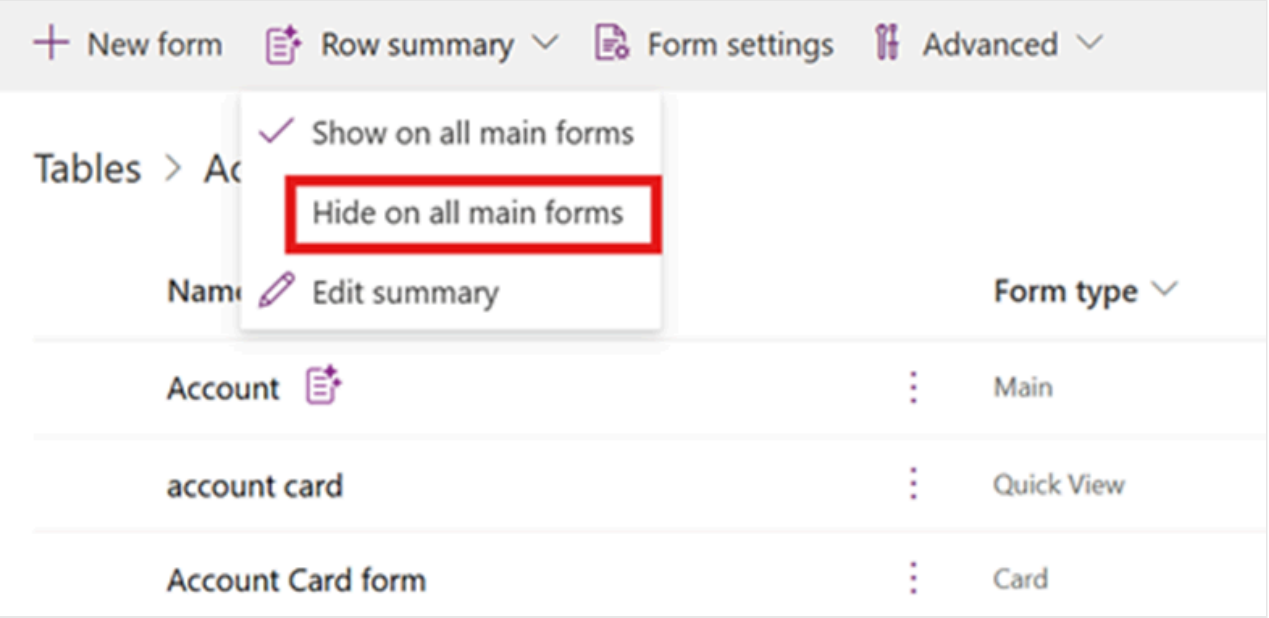
You must respond in language LanguageCode

Edit a row summary

To edit a row summary, open the table, and then under **Customizations** select **Row summary (applied)**. Follow the steps similar in Create a row summary to make and apply changes.

Hide a row summary

1. In Power Apps, select **Tables** on the left navigation pane, and open the table where you want to edit a row summary.
2. Under **Customizations**, select **Row summary**.
3. Unselect the **Show row summary** option at the top right of the modal.



> **Note**
>
> If you're trying to hide the default summary for case, lead, and opportunity, which are provided via the Dynamics 365 Customer Service and Dynamics 365 Sales applications, go

to these articles to learn more about options for configuring and/or disabling these summaries:

- Case: Customer Service summary
- Lead and opportunity: Sales summary features.

Adding summaries to solutions

Row summaries are solution-aware and can be added to a solution to facilitate moving between environments. To add a summary to a solution, follow these steps:

1. In Power Apps, select **Solutions** on the left navigation pane, and open the desired solution.
2. Select **Add existing** > **More** > **Other** > **AI Skill Config**.
3. Select the rows corresponding to the tables you created summaries for, and then select **Add**.

> **Important**
>
> Don't add AI Skill Config table rows with an owner of *System* to your solutions. These don't correspond to row summaries you have created and might cause solution import to fail.

Related information

FAQ for prompts and text generation capabilities

Last updated on 02/12/2026

Import data from Excel and export data to CSV

To get (import) data into Microsoft Dataverse tables, use an Excel worksheet file or a comma-separated values (CSV) file.

When you export Dataverse table data, it's exported as a CSV file.

Import from an Excel or CSV file

There are two ways to import data from Excel.

- Option 1: Import by creating and modifying a file template
- Option 2: Import by bringing your own source file

> **Important**
>
> - To import or export data, you must have the **Environment Maker** security role.
> - Import from Excel or CSV file using the **Import** > **Import data from Excel** command isn't available in GCC, GCC High, and DoD environments. To work around this limitation, from the **Tables** area in Power Apps select **Import** > **Import data**, and then choose a data source, such as **Excel workbook** or **Text/CSV**.

Option 1: Import by creating and modifying a file template

Every table has required columns that must exist in your input file. We recommend that you create a template. To do this, export data from the table. Then, use the same file and modify it with your data. Finally, import the modified file back into the table. Using a template can save you time because you won't have to specify the required columns for each table.

Prepare the file template

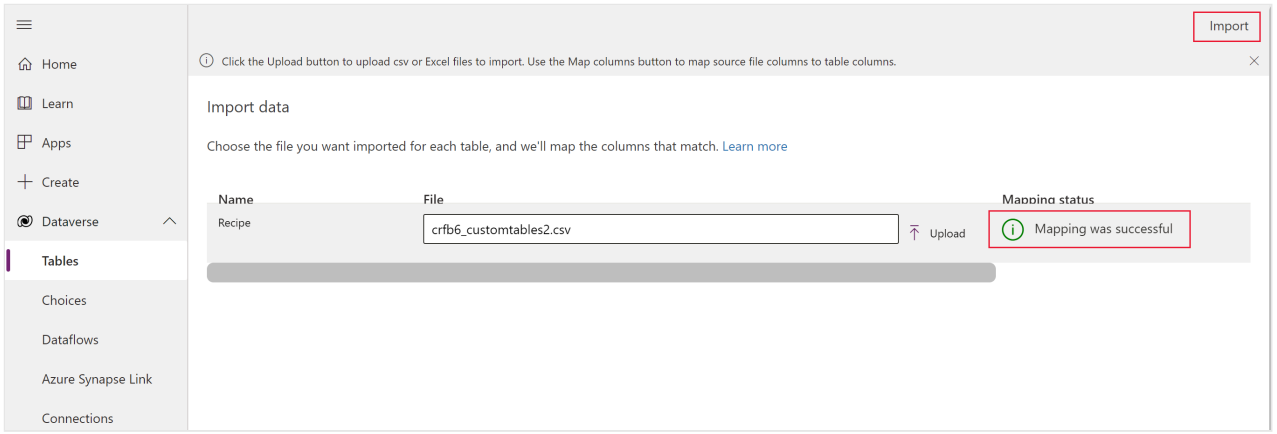
1. Export the table data.
2. Define a plan to make sure data is unique before you import it. Use either primary keys or alternate keys. More information: Ensure uniqueness when you import data into a table from Excel or CSV

Modify the file with your data

Copy data from your Excel or CSV file into the template that you created in the previous step.

Import the file

1. On powerapps.com select **Tables** in the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **Import** > **Import data from Excel**.
3. Select the tables where you want to import data, and then select **Next**.
4. On the **Import data** page, select **Upload**, and choose your file. Follow the prompts to upload your file.
5. After the file is uploaded and **Mapping status** indicates **Mapping was successful**, select **Import** from the top-right corner. Go to Troubleshoot mapping errors with Excel to navigate and fix any mapping errors.



After the import finishes successfully, you'll see the total number of inserts and updates.

New import from Excel (preview)

[This topic is pre-release documentation and is subject to change.]

The new import experience lets users import data directly from an Excel file into existing Dataverse tables using a unified UI. This experience improves upon the original import method by offering:

- **Assisted mapping**: Uses vector-based search to provide more contextually relevant column mapping suggestions.
- **Sheet selection**: Users can choose which sheet to import from within a multi-sheet Excel file.
- **Notifications**: Real-time feedback on import status, including success, in-progress, and error states.
- **Error logs**: Downloadable logs to help users self-diagnose and resolve issues.
- **Performance improvements**: Asynchronous ingestion offers better reliability and speed for large files.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Prerequisites

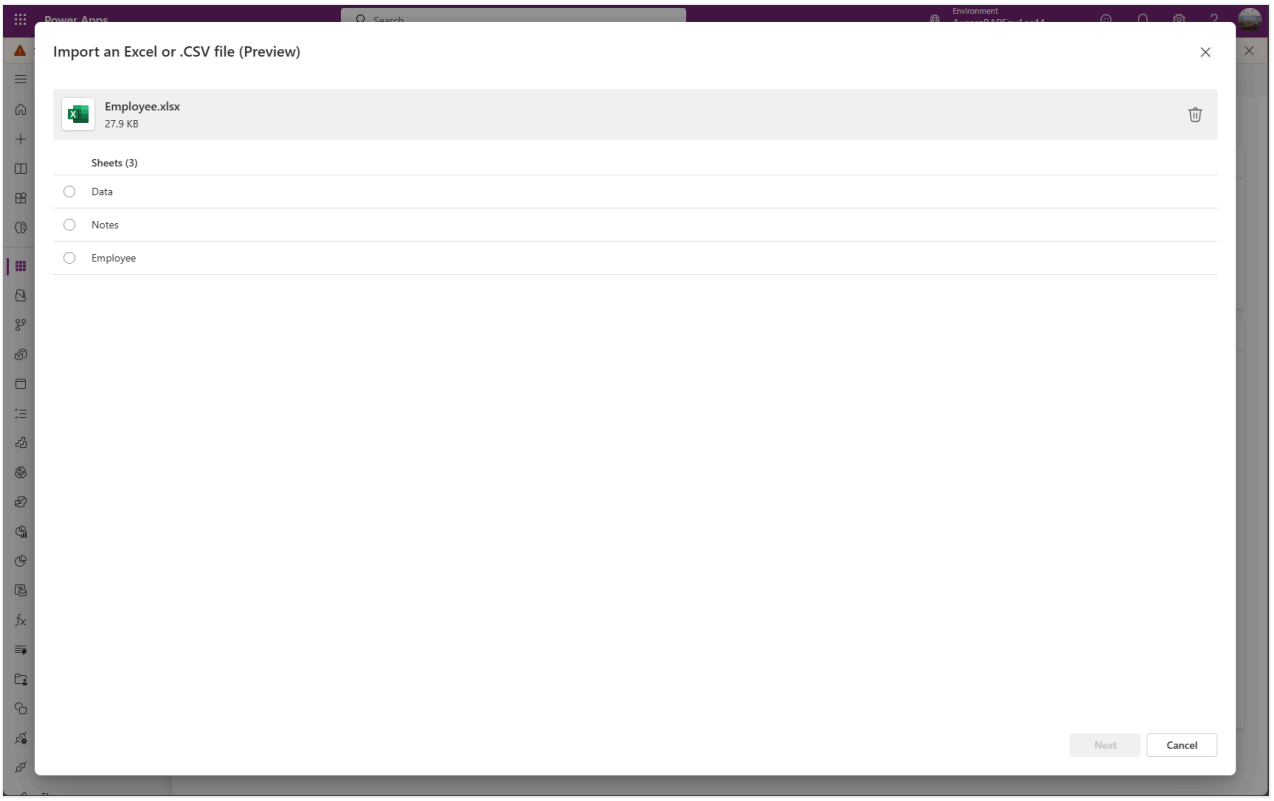
To use the feature a Power Platform admin must enable the **"Import Excel to existing table with assisted mapping"** environment setting in the Power Platform admin center. Select the environment, and then select **Settings** > **Product** > **Features** and enable the **"Import Excel to existing table with assisted mapping"** setting.

> **Note**
>
> The **Import Excel to existing table with assisted mapping** environment setting is being removed from the Power Platform admin center. This feature is currently being improved and that setting is no longer used.

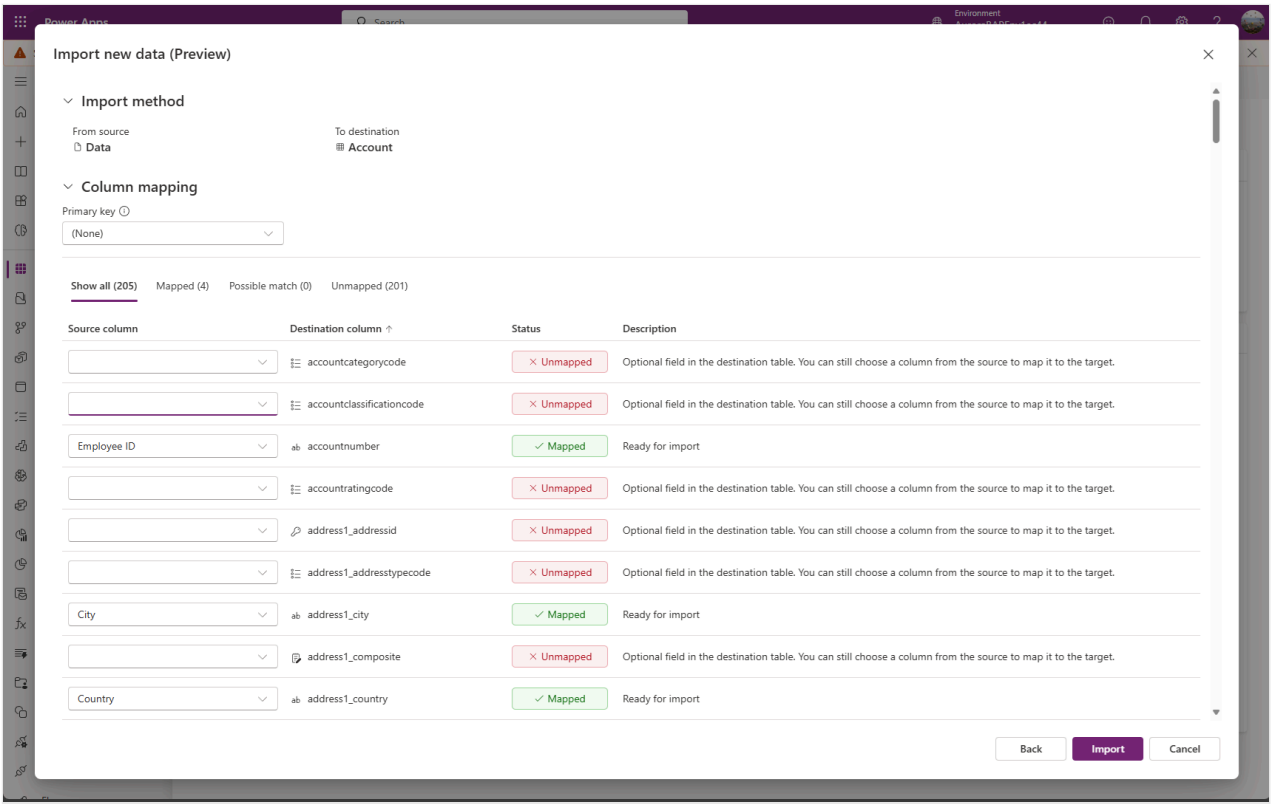
Import from Excel (preview) steps

1. Go to the table you want to import data to, and then on the command bar select **Import** > **Import from Excel (Preview)**.

2. Upload your Excel file and select the sheet to import.



3. Map columns. A primary key is required if you want to update existing records in the table. If no primary key is selected ("None" option), then all records from the Excel sheet or CSV file are added as new records.



4. Review the summary and select **Import**.

Monitor the import status from notifications:

- In progress: Ongoing import activity with the message **Your data is importing** displayed.
- Success: Data was processed or indexed with the message **Upload completed** displayed.

- Error: **Upload completed with some errors** displayed that includes a link to download logs for troubleshooting.

Option 2: Import by bringing your own source file

If you're an advanced user and know the required columns for a given table for Dataverse tables, define your own Excel or CSV source file. Follow the steps in Import the file.

Troubleshoot mapping errors with Excel

If you get mapping errors after you upload your file, select **Map status**. Take the following steps to inspect and rectify the column mapping errors.

1. Use the drop-down menu on the right, under **Show**, to walk through the **Unmapped columns**, **Fields with error**, or **Required Fields**.

   > **Tip**
   >
   > - Depending on whether you get a **Warning** or an **Error**, inspect **Unmapped columns** or **Fields with error** through the drop-down menu in **Column Mappings**.
   > - Use the *upsert* (**Update** or **Insert**) logic to either update the row, if it already exists, or to insert a new row.

2. After you resolve all the errors and warnings, select **Save Changes** in the top-right corner. You'll go back to the **Import Data** screen.

3. When the **Mapping status** column shows **Mapping was successful**, select **Import** from the top-right corner.

When the **Import completed successfully** message appears, the total inserts and updates are displayed.

Ensure uniqueness when you import data into a table from Excel or CSV

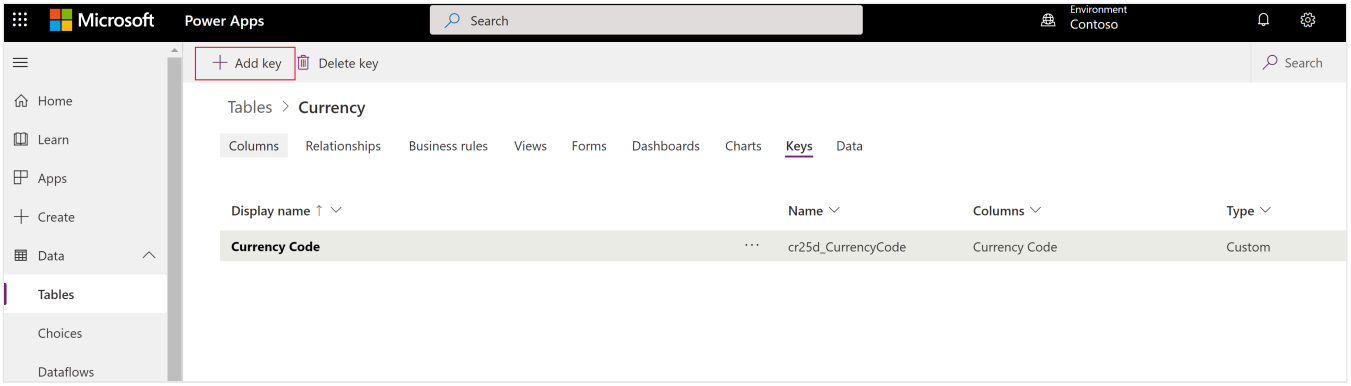
Dataverse tables use a primary key to uniquely identify rows within a Dataverse table. The primary key for a Dataverse table is a globally unique identifier (GUID). It forms the default basis for row identification. Data operations, like importing data into Dataverse tables, surface the default primary keys.

Example:
The primary key for an **Account** table is **accountid**.

	A	B	C	E	DI
1	accountid	name	accountnumber	address1_composite	transactioncurrencyid
2	802534f7-5b45-e811-a953-000d3a33bcb9	Coho Winery (sample)	BABCO88H	137 Lancelot Dr	88c6c893-5b45-e811-a953-000d3a33bcb9
3	7e2534f7-5b45-e811-a953-000d3a33bcb9	A. Datum Corporation (sample)	AAAAA	2137 Birchwood Dr	88c6c893-5b45-e811-a953-000d3a33bcb9
4	7c2534f7-5b45-e811-a953-000d3a33bcb9	Alpine Ski House (sample)	ABCO9M32	2313 B Southampton	88c6c893-5b45-e811-a953-000d3a33bcb9

Sometimes, a primary key might not work when you integrate data from an external source. Use Dataverse to define alternate keys that uniquely identify a row in place of the primary key.

Example:
For an **Account** table, you might set **transactioncurrencyid** as an alternate key by using a natural key-based identification. For example, use **US Dollar** instead of the GUID value **88c6c893-5b45-e811-a953-000d3a33bcb9** shown previously. You can also choose **currency symbol** or **currency name** as keys. More information: Define alternate keys using Power Apps portal



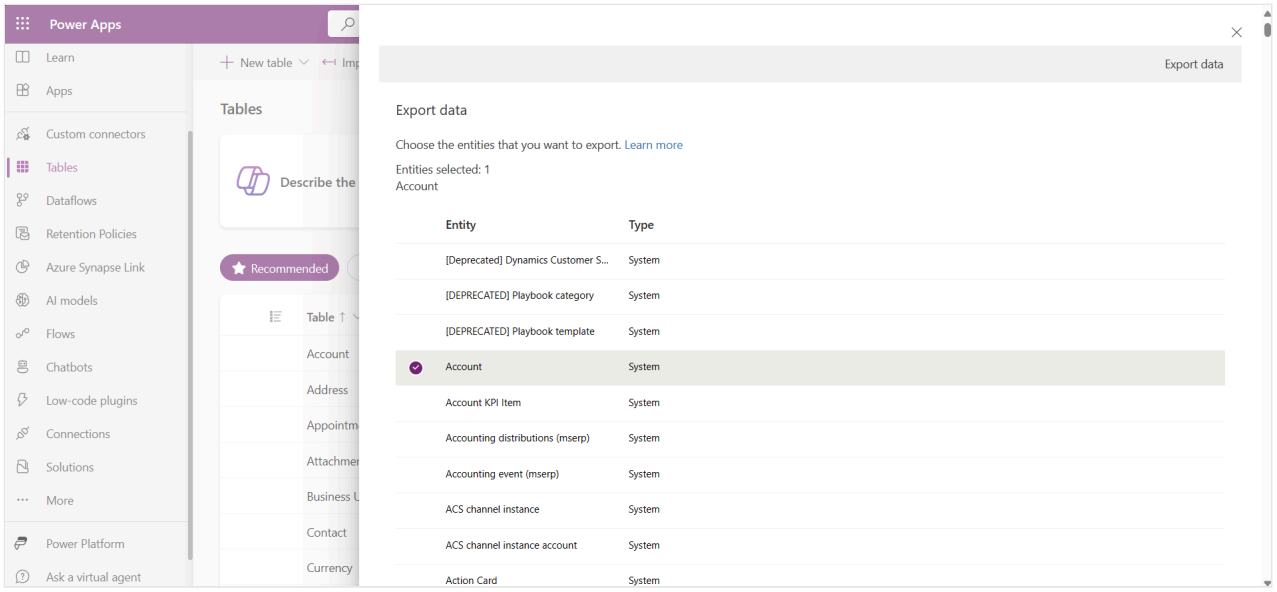
	A	B	C	DQ
1	accountid	name	accountnumber	transactioncurrencyid.currencyname
2	762534f7-5b45-e811-a953-000d3a33bcb9	Blue Yonder Airlines (sample)	ACSHN2S4	US Dollar
3	782534f7-5b45-e811-a953-000d3a33bcb9	City Power & Light (sample)		Euro
4	7a2534f7-5b45-e811-a953-000d3a33bcb9	Contoso Pharmaceuticals (sample)		Rupee

You can still use primary keys as identifiers after you specify alternate keys. In the preceding sample, the first file is still valid if GUIDs are valid data.

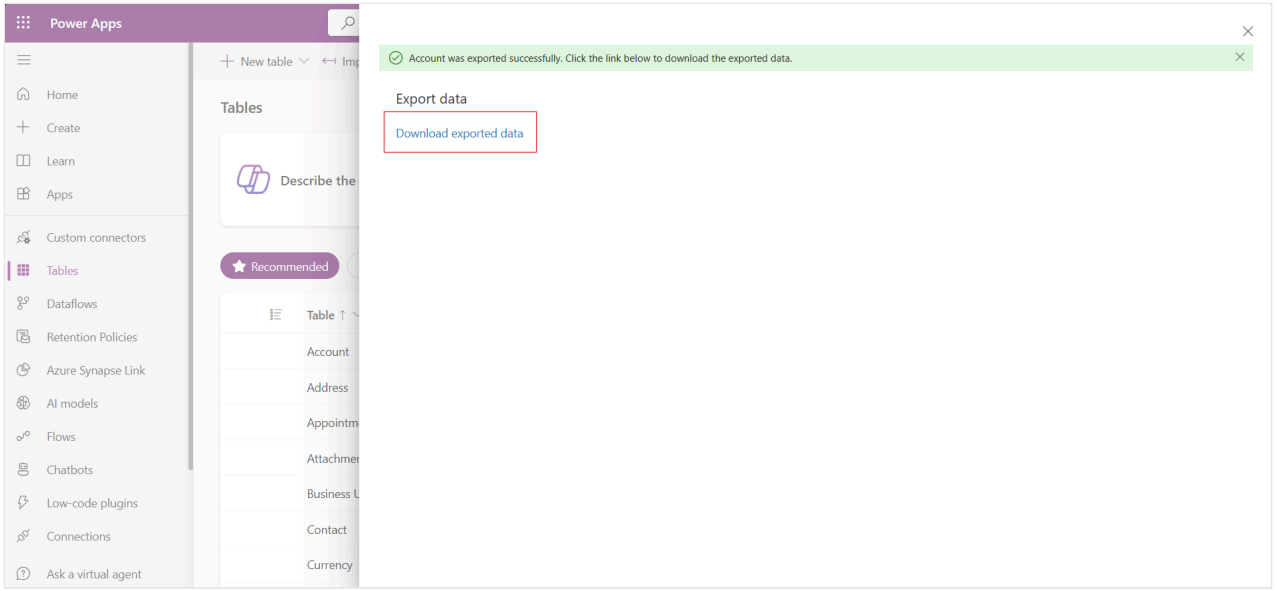
Export data

Export data from a single table in a comma-separated value (CSV) format.

1. Sign into Power Apps, on the left navigation pane select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **Export** > **Export data**.
3. Select the table from which you want to export data, and then select **Export data**.



4. After the export finishes successfully, select **Download exported data** to download the CSV file to the download folder specified in your web browser.



> **Note**
>
> Exports have a 12 minute time limit. If the volume of data exported exceeds 12 minutes the export will fail. If this occurs, export data in smaller segments.

Unsupported data types and fields

The following data types aren't currently supported for import or export.

- Timezone
- Choices (multiselect)
- Image

- File

The following fields are system fields and aren't supported for import and export.

- `Ownerid`
- `Createdby`
- `Createdonbehalfby`
- `Createdon`
- `Modifiedby`
- `Modifiedonbehalfby`
- `Modifiedon`
- `Overriddencreatedon`

> **Note**
>
> Get Data from Excel and Export Data features are currently not included in the Power Apps Developer Plan.

Import using a connector

Use a connector to import data from a selection of many different sources, such as Azure, SQL Server database, SharePoint, Access, OData, and more. More information: Create and use dataflows in Power Apps

See also

Tables in Dataverse

Last updated on 01/16/2026

Create and use dataflows in Power Apps

With advanced data preparation available in Power Apps, you can create a collection of data called a dataflow, which you can then use to connect with business data from various sources, clean the data, transform it, and then load it to Microsoft Dataverse or your organization's Azure Data Lake Gen2 storage account.

A dataflow is a collection of tables that are created and managed in environments in the Power Apps service. You can add and edit tables in your dataflow, as well as manage data refresh schedules, directly from the environment in which your dataflow was created.

Once you create a dataflow in the Power Apps portal, you can get data from it using the Dataverse connector or Power BI Desktop Dataflow connector, depending on which destination you chose when creating the dataflow.

There are three primary steps to using a dataflow:

1. Author the dataflow in the Power Apps portal. You select the destination to load the output data to, the source to get the data from, and the Power Query steps to transform the data using Microsoft tools that are designed to make doing so straightforward.

2. Schedule dataflow runs. This is the frequency in which the Power Platform Dataflow should refresh the data that your dataflow will load and transform.

3. Use the data you loaded to the destination storage. You can build apps, flows, Power BI reports, and dashboards or connect directly to the dataflow's Common Data Model folder in your organization's lake using Azure data services like Azure Data Factory, Azure Databricks, or any other service that supports the Common Data Model folder standard.

The following sections look at each of these steps so you can become familiar with the tools provided to complete each step.

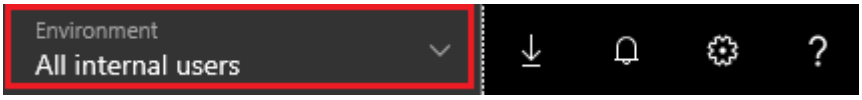
Create a dataflow

Dataflows are created in one environment. Therefore, you'll only be able to see and manage them from that environment. In addition, individuals who want to get data from your dataflow must have access to the environment in which you created it.

> **ⓘ Important**
>
> - Creating dataflows is currently not available with Power Apps Developer Plan licenses.

- The Firefox web browser currently isn't supported for the "Download refresh history file" action in Power Apps dataflows. More information: [You receive the error message "There was a problem refreshing the dataflow"]

1. Sign in to Power Apps, and verify which environment you're in, find the environment switcher near the right side of the command bar.

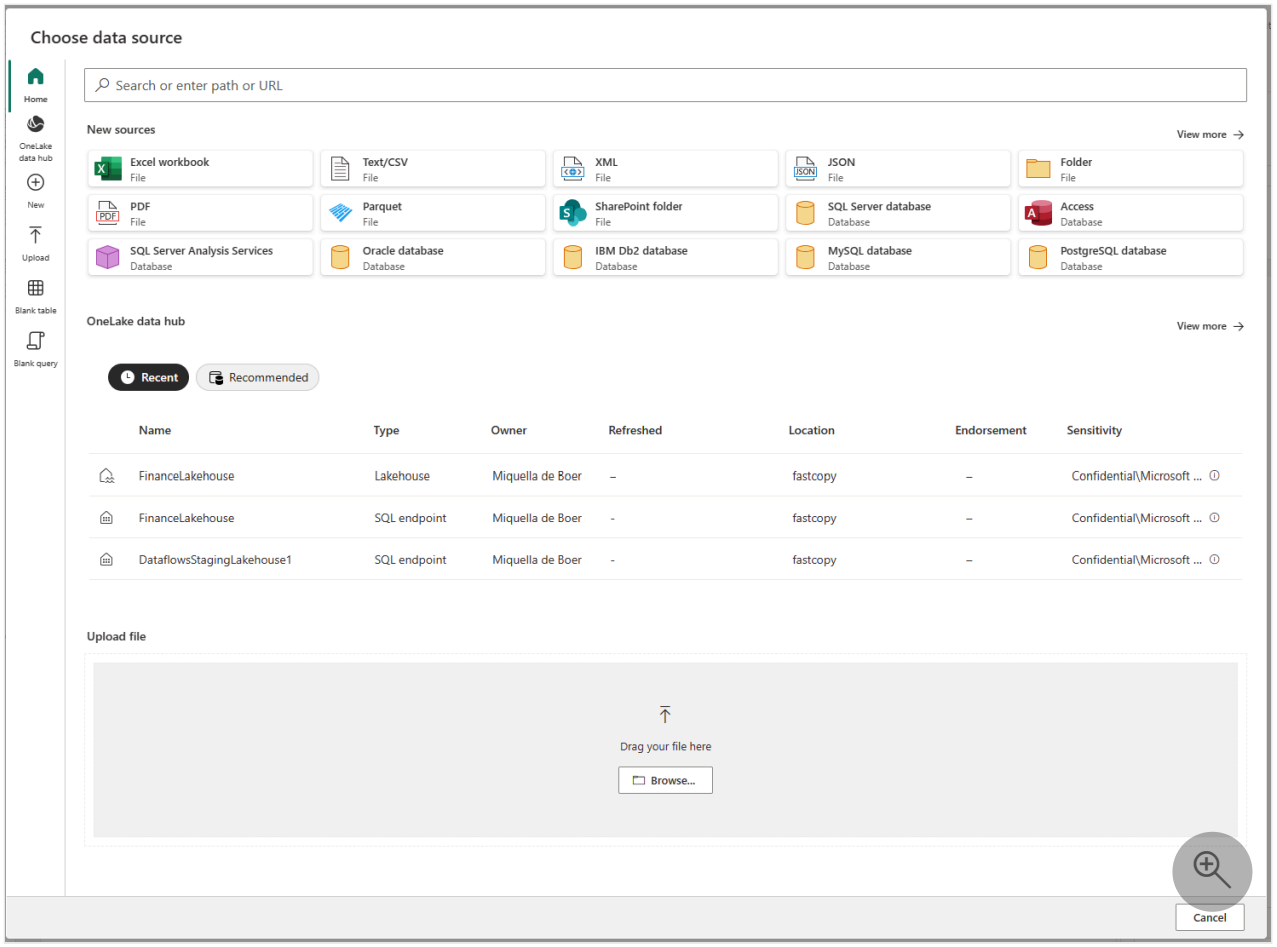


2. On the left navigation pane, select **Dataflows**. If the item isn't in the side panel pane, select ...More and then select the item you want.

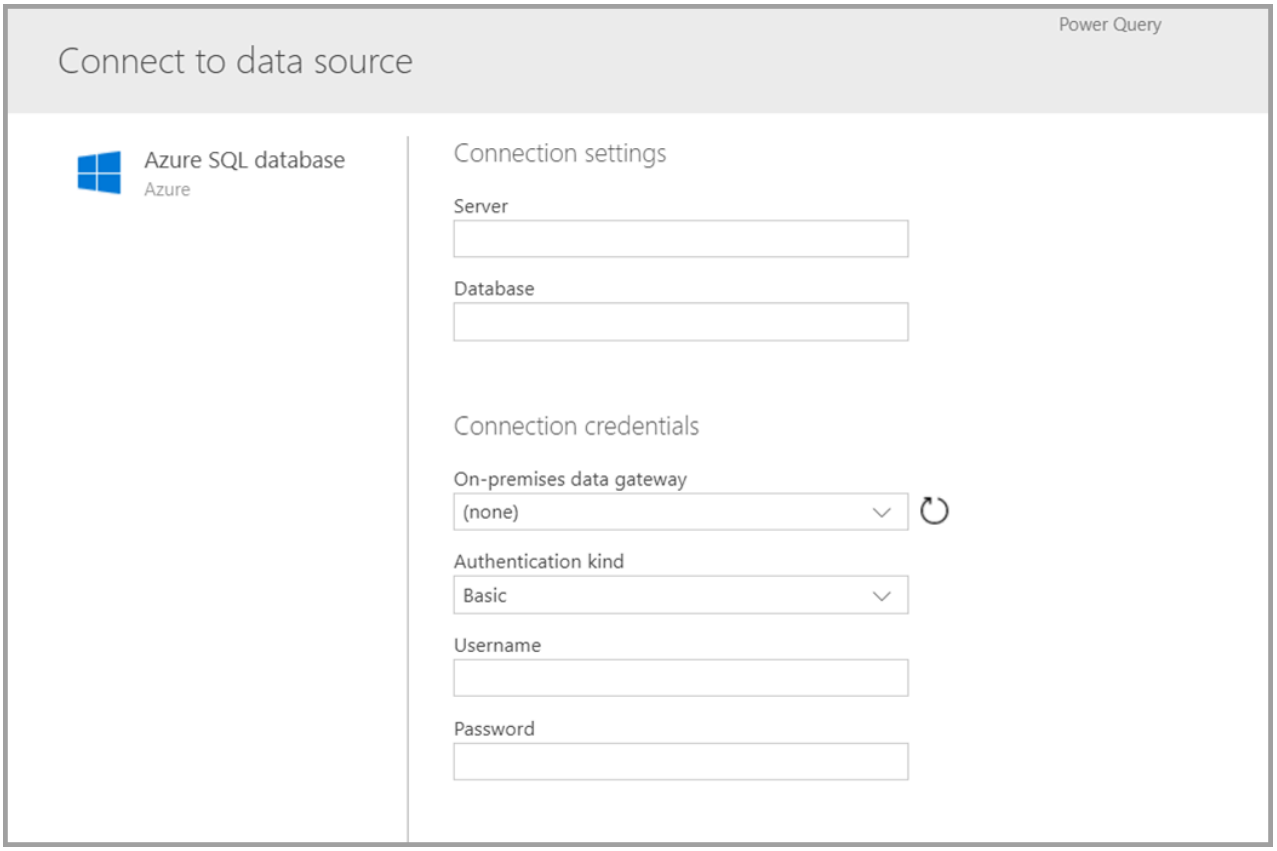
3. Select **New dataflow**. On the **New Dataflow** page, enter a **Name** for the dataflow. By default, dataflows store tables in Dataverse. Select **Analytical entities only** if you want tables to be stored in your organization's Azure Data Lake storage account. Select **Create**.

   > **Important**
   >
   > There's only one owner of any dataflow—the person who created it. Only the owner can edit the dataflow. Authorization and access to data created by the dataflow depend on the destination you loaded data to. Data loaded into Dataverse will be available via the Dataverse Connector and requires the person accessing the data to be authorized to Dataverse. Data loaded into your organization's Azure Data Lake Gen2 storage account is accessible via the Power Platform Dataflow connector and access to it requires membership within the environment it was created in.

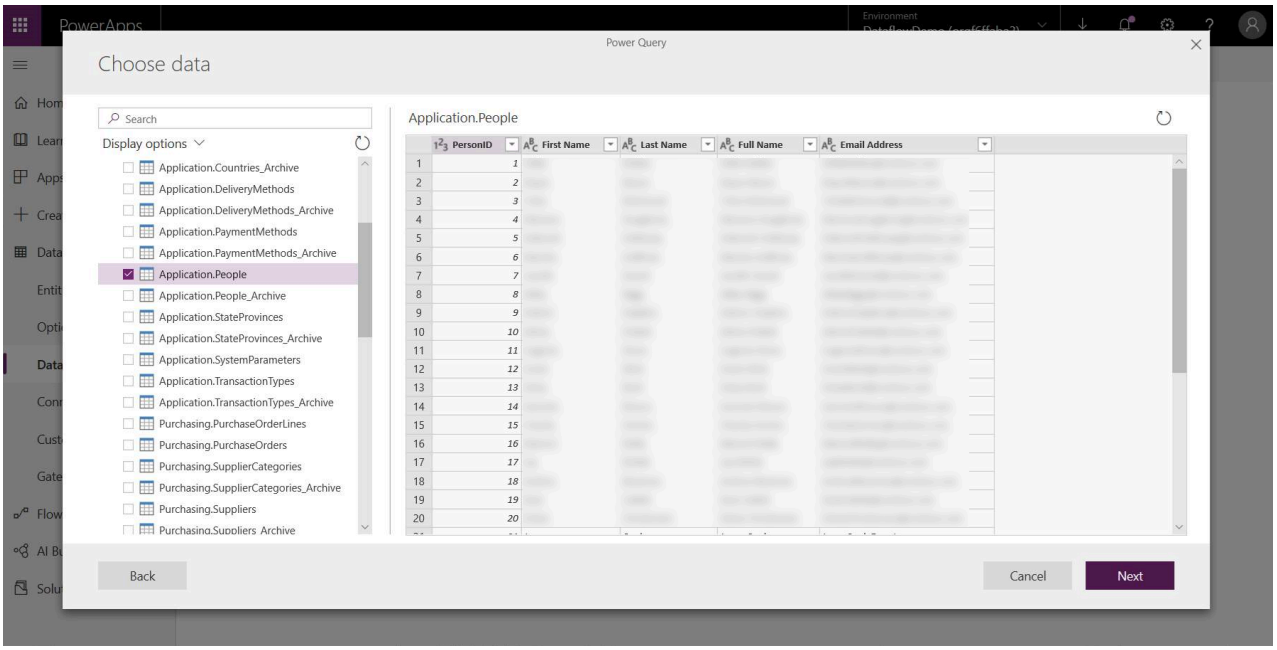
4. On the **Choose data source** page, select the data source where the tables are stored. The selection of data sources displayed allows you to create dataflow tables.



5. After you select a data source, you're prompted to provide the connection settings, including the account to use when connecting to the data source. Select **Next**.



6. Once connected, you select the data to use for your table. When you choose data and a source, the Power Platform Dataflow service will subsequently reconnect to the data source in order to keep the data in your dataflow refreshed, at the frequency you select later in the setup process.

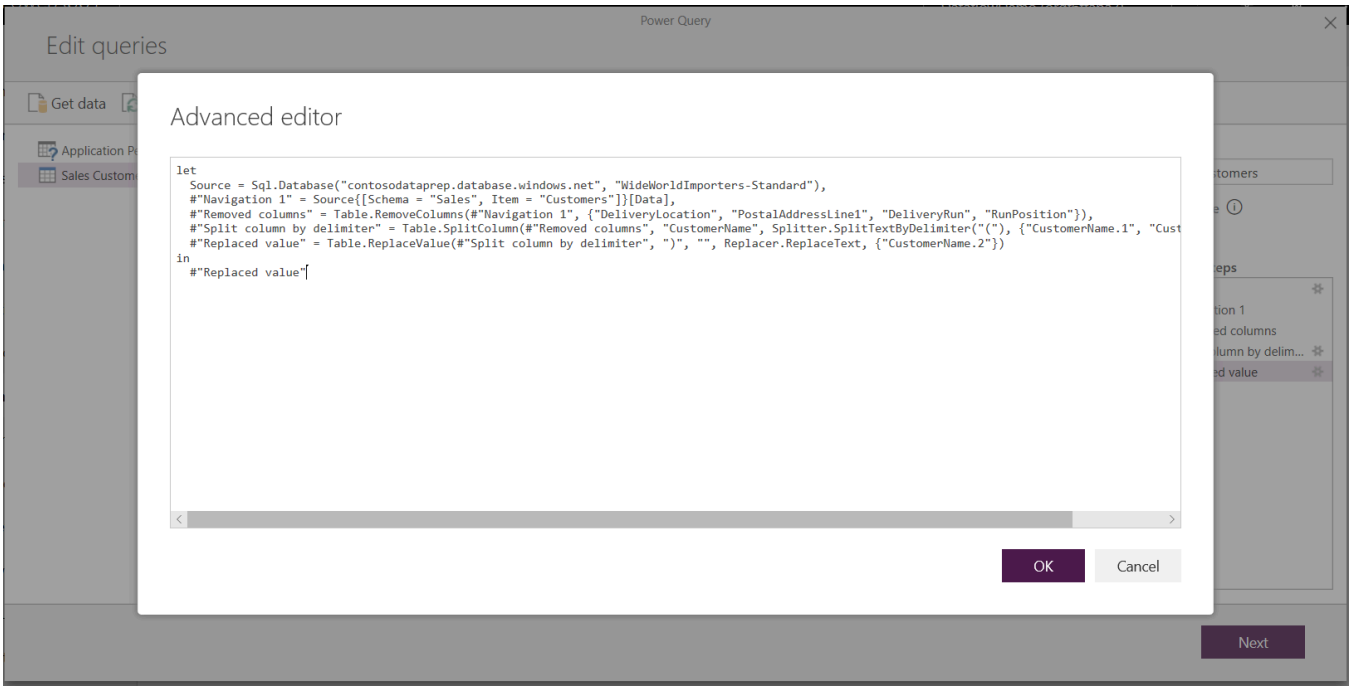


Now that you've selected the data to use in the table, you can use the dataflow editor to shape or transform that data into the format necessary for use in your dataflow.

Use the dataflow editor to shape or transform data

You can shape your data selection into a form that works best for your table using a Power Query editing experience, similar to the Power Query Editor in Power BI Desktop. To learn more about Power Query, see Query overview in Power BI Desktop.

If you want to see the code that Query Editor is creating with each step, or if you want to create your own shaping code, you can use the advanced editor.

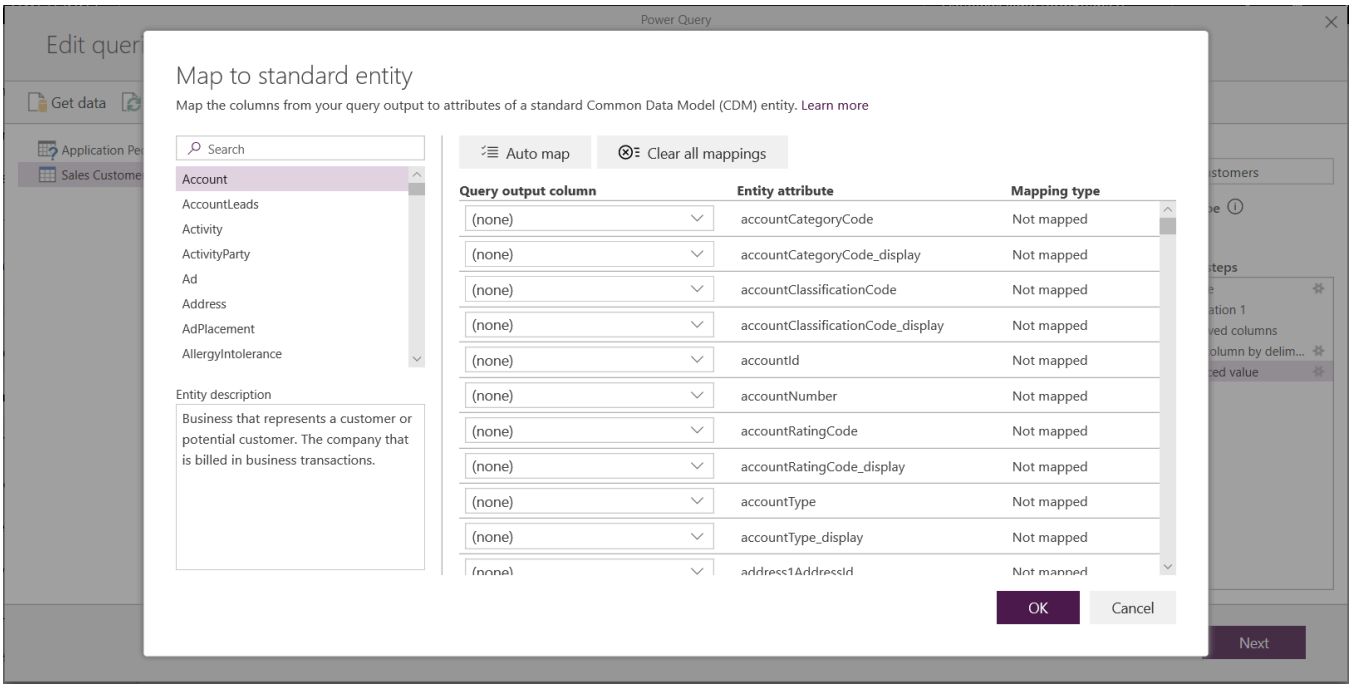


Dataflows and the Common Data Model

Dataflows tables include new tools to easily map your business data to the Common Data Model, enrich it with Microsoft and non-Microsoft data, and gain simplified access to machine learning. These new capabilities can be leveraged to provide intelligent and actionable insights into your business data. Once you've completed any transformations in the edit queries step described below, you can map columns from your data source tables to standard table columns as defined by the Common Data Model. Standard tables have a known schema defined by the Common Data Model.

For more information about this approach, and about the Common Data Model, see The Common Data Model.

To leverage the Common Data Model with your dataflow, select the **Map to Standard** transformation in the **Edit Queries** dialog. In the **Map tables** screen that appears, select the standard table that you want to map.



When you map a source column to a standard column, the following occurs:

1. The source column takes on the standard column name (the column is renamed if the names are different).

2. The source column gets the standard column data type.

To keep the Common Data Model standard table, all standard columns that aren't mapped get *Null* values.

All source columns that aren't mapped remain as is to ensure that the result of the mapping is a standard table with custom columns.

Once you've completed your selections and your table and its data settings are complete, you're ready for the next step, which is selecting the refresh frequency of your dataflow.

Set the refresh frequency

Once your tables have been defined, you should schedule the refresh frequency for each of your connected data sources.

Dataflows use a data refresh process to keep data up to date. In the **Power Platform Dataflow authoring tool**, you can choose to refresh your dataflow manually or automatically on a scheduled interval of your choice.

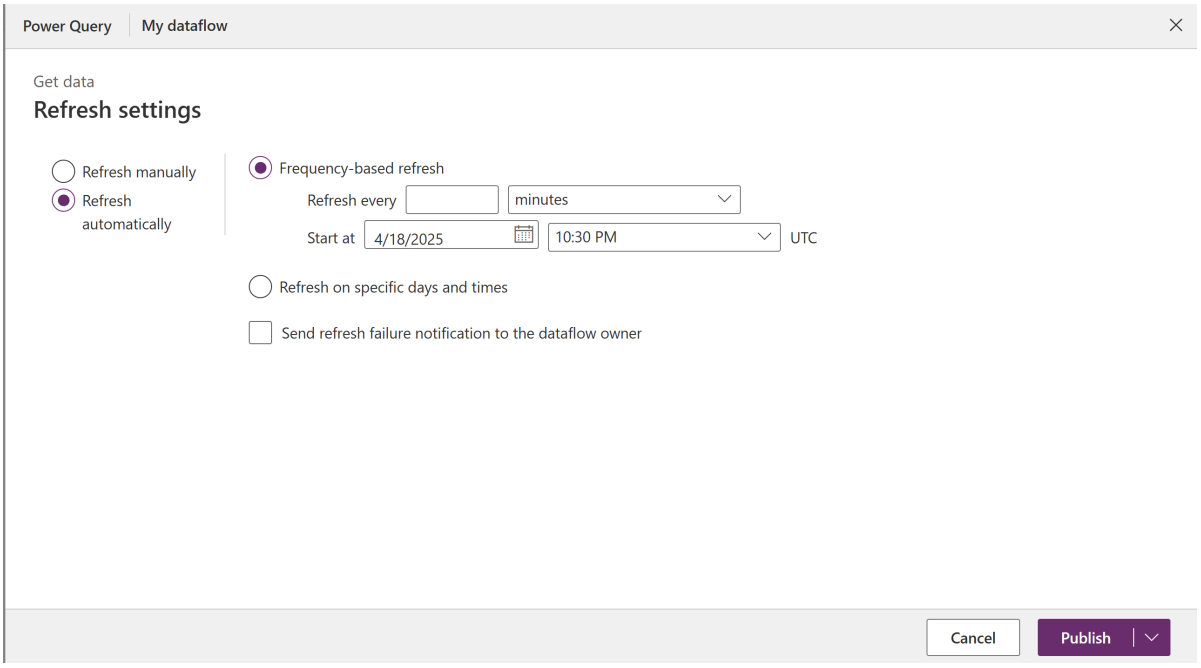
Schedule a refresh automatically

1. Select **Refresh automatically**.

2. Enter the dataflow frequency:

    - **Frequency-based refresh**. Set how often in 30-minute increments, start date, and time in UTC.

    - **Refresh on specific days and times**. Choose time zone, frequency (daily or weekly) and time of day in 30-minute increments.



3. Select **Publish.**

Some organizations might want to use their own storage for creation and management of dataflows. You can integrate dataflows with Azure Data Lake Storage Gen2 if you follow the requirements to set up the storage account properly. More information: Connect Azure Data Lake Storage Gen2 for dataflow storage

Troubleshooting data connections

There might be occasions when connecting to data sources for dataflows runs into issues. This section provides troubleshooting tips when issues occur.

- **Salesforce connector.** Using a trial account for Salesforce with dataflows results in a connection failure with no information provided. To resolve this, use a production Salesforce account or a developer account for testing.

- **SharePoint connector.** Make sure you supply the root address of the SharePoint site, without any subfolders or documents. For example, use a link similar to `https://microsoft.sharepoint.com/teams/ObjectModel`.

- **JSON File connector.** Currently you can connect to a JSON file using basic authentication only. For example, a URL similar to `https://XXXXX.blob.core.windows.net/path/file.json?sv=2019-01-01&si=something&sr=c&sig=123456abcdefg` is currently not supported.

- **Azure Synapse Analytics.** Dataflows don't currently support Microsoft Entra authentication for Azure Synapse Analytics. Use basic authentication for this scenario.

> ⓘ Note
>
> If you use data loss prevention (DLP) policies to block the **HTTP with Microsoft Entra (preauthorized)** connector then **SharePoint** and **OData** connectors will fail. The **HTTP with Microsoft Entra (preauthorized)** connector needs to be allowed in DLP policies for **SharePoint** and **OData** connectors to work.

Troubleshoot the error: Connection to Dataverse failed. Please check the link below on how to fix this issue

Users might receive an error message if the connection they're using for export requires a fix. In this case, the user receives an error message that states **Connection to Dataverse failed. Please check the link below on how to fix this issue**.

To resolve this issue:

1. In Power Apps (make.powerapps.com), select **Connections** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Locate the **Microsoft Dataverse (legacy)** connection.
3. Select the **Fix connection** link in the **Status** column, and follow the instructions on your screen.

After the fix completes, retry the export.

You receive the error message "There was a problem refreshing the dataflow"

This error occurs when you try to refresh the dataflow while using the Firefox web browser. To work around this issue use a different web browser, such as Microsoft Edge or Google Chrome.

Next steps

The following articles are useful for further information and scenarios when using dataflows:

- Add data to a table in Dataverse
- Using an on-premises data gateway in Power Platform dataflows
- Connect Azure Data Lake Storage Gen2 for dataflow storage
- Use dataflows in solutions

For more information about the Common Data Model:

- Common Data Model - overview
- Learn more about the Common Data Model schema and tables on GitHub

---

Last updated on 01/09/2026

Dataverse long term data retention overview

Microsoft Dataverse supports custom retention policies to securely retain unlimited data long term in a cost-efficient way. While Dataverse can support your business growth with no limit on active data, you might want to consider moving inactive data to the Dataverse long term retention store.

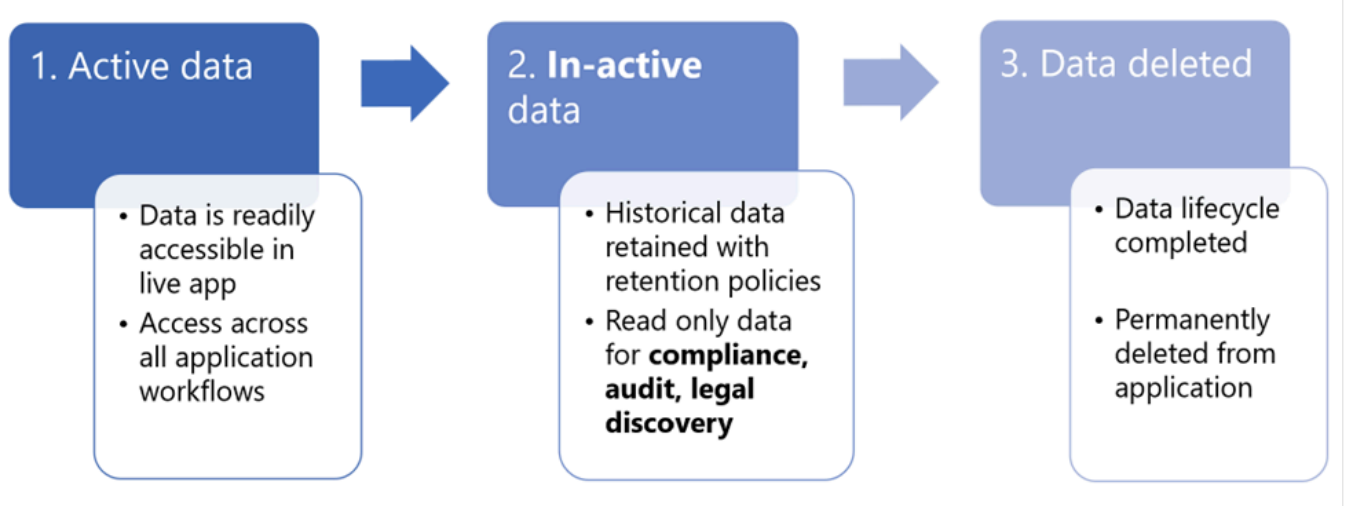
> **Important**
>
> The Power Platform environment where the retention policy resides must be a Managed Environment. This also applies to Dynamics 365 customers who will be required to upgrade to a Managed Environment if the environment isn't already a Managed Environment.
>
> Customers who don't meet this requirement can continue to create data retention policies, but the policies are disabled.

Watch this video to learn about Dataverse long term data retention.

https://learn-video.azurefd.net/vod/player?id=c34fea68-46b2-4486-b9c1-a3ef9d01b4f9&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fdata-retention-overview

Business application data lifecycle

Consider the business application data lifecycle in three stages. First active data, which over time transitions to inactive data, and finally transitions to deleted data.



Expand table

Stage	Description
1. Active data	Data is readily available and transformed via apps. Accessed across all application workflows. Also referred to as *live* data.
2. Inactive data	Data is immutable and read-only. Is stored in long term retention. Has limited access. Retained for compliance, audit, and legal discovery.
3. Deleted data	Permanently deleted. Data lifecycle is completed.

Dataverse delivers native platform support for long-term retention of data. It helps your organization get immediate and ongoing benefits:

- Securely retain the historical application data long term for audit, legal, and regulatory requirements.
- Access the read-only data for limited inquiry purposes.
- Reduce database capacity consumed and save money on inactive data.
- Avoid IT investments required to build and maintain custom solutions for long-term retention of historical application data.

How it works

Application admins set up custom policies for a table with criterion to retain data long term. The retained data is never moved out of Dataverse, the data is stored in Dataverse long term retention. The data is always secured with Dataverse security backed by Microsoft Entra ID.

> **Important**

- Once data is retained with Dataverse long term retention it can't be moved back to the Dataverse live (active) application state.
- When a retention policy is run, the process makes API requests in Microsoft Power Platform. These requests are counted towards the existing API requests available with your plan. More information: Requests limits and allocations

Dataverse provides read-only access to the retained data via:

- Advanced Find within an application.
- Power Automate cloud flow.
- Dataverse OData APIs for custom screens.

Types of data retained long term

Dataverse standard (except system) tables, custom tables, along with attachments and images, can be retained in Dataverse long term storage.

> **Note**
>
> Currently, audit tables and elastic tables aren't supported for long term retention.

Admins set retention policies on tables when the application maker enables long term retention for the table. When a maker enables retention on a parent root table, it also enables retention for all child tables and associated attachments and images.

Long term data retention and existing delete action and plugins

All the existing delete action cascade relationships and plugins for tables are executed when a data retention policy is run on the table. In addition, the retain action can be used to trigger custom behavior. Application makers can write custom plugins for the retain action.

More information: Long-term data retention

Storage capacity reports

With Dataverse long term retention, data is never moved out of Dataverse. The reported capacity in the existing Power Platform admin reports includes both live (active) and retained

(inactive) data in GBs.

- Database capacity details reported:
  - Database capacity reported is the sum of the live and retained data. The overall database capacity consumed is reduced depending on the proportion of the data rows retained during a policy run.

- Notice that the policy run takes 72 to 96 hours to complete and there's an additional 24 hours afterward for the database capacity reports to appear.

- When available, the reports display two entries for a table that is enabled for long term retention:
  - *Table*, such as **Case**[1] or **Contact**.
  - *Table-Retained*, such as **Case-Retained** and **Contact-Retained**.

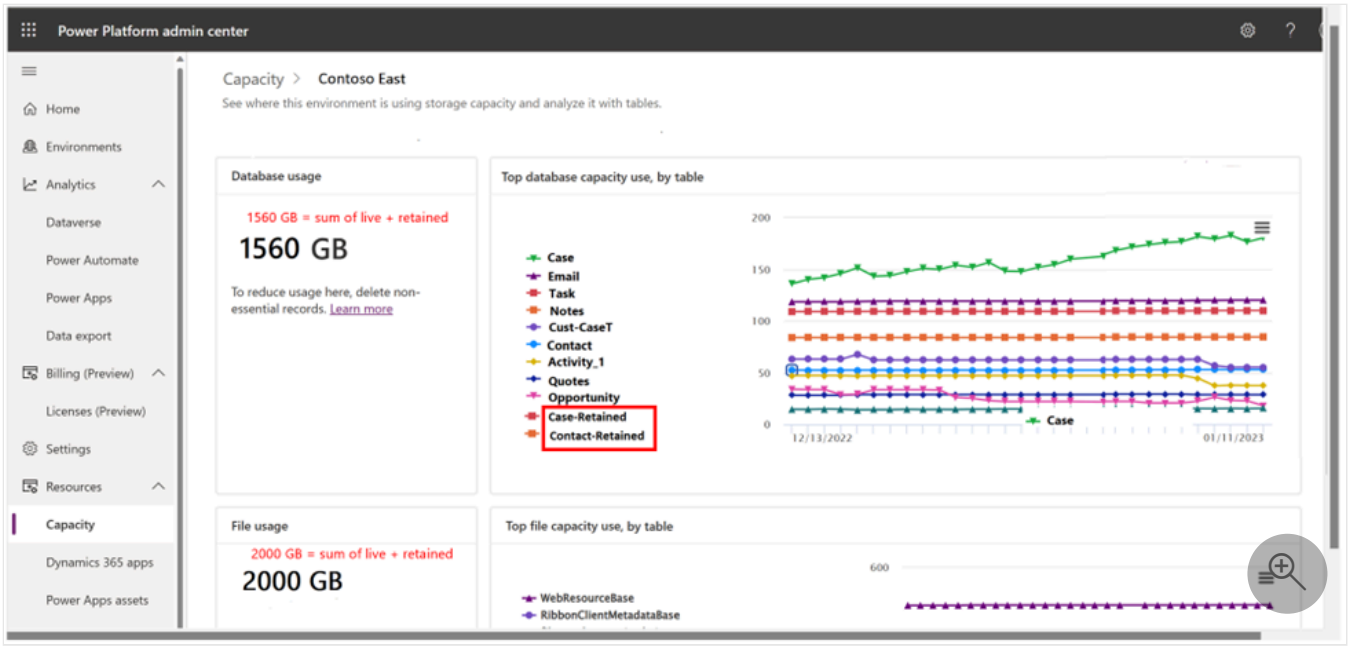
  [1] The case table requires a Dynamics 365 app, such as Dynamics 365 Customer Service.

- File capacity details reported:
  - If there are associated file attachments retained long term, the file capacity reflects the sum of the live and retained data. There will be no reduction or savings observed with file capacity after running a retention policy, which involved file attachments.

Log capacity reports aren't currently available.

Viewing the capacity reports

Imagine a nonproduction scenario where all the live data for the case and contacts tables are retained with Dataverse long term retention. After the long term retention policy is successfully completed, there are entries for **Case-Retained** and **Contact-Retained** in the report. The reduction in database capacity consumed depends on the number of rows and column data types involved in the retention process. The reports could take a few days to a week to reflect the fully reduced savings in production environments. For nonproduction environments, the reports could take up to two weeks to reflect the savings.



> [!NOTE]
> - If the tables are not visible in the report, export the view to an Excel worksheet by selecting the three vertical bars on the top right of the capacity report, and then select **Download all tables**.
> - If a table commonly has associated child tables, the capacity reports also display the child table-retained GB size.

For more information about capacity reports, go to New Microsoft Dataverse storage capacity.

Understanding long term retention storage costs

Dataverse long term retention requires no additional storage purchases and it doesn't require you to purchase the feature as a separate service. Every GB moved from Dataverse database to Dataverse long term retention, consumes, on average, 50% less database capacity. This is because the data is compressed in Dataverse long term retention.

For example:

- Suppose the database capacity consumed by Contoso is 1,000 GB today and the scheduled long term retention policy was run and retains 200 GB of data with Dataverse long term retention.
- With an average compression of at least 50%, the retained data size is 100 GB, a savings of 50% compared to when the data was in the active state.

- This implies Contoso now has 800 GB of active data and 100 GB of inactive data in Dataverse.
- In this scenario, the storage capacity reports display the database usage as 900 GB (800 GB + 100 GB).

> **Note**
>
> - Dataverse long term retention compresses the database data. The amount of compression depends on the kind of data in Dataverse. With some data (indeterministic), you might notice larger than 50% savings while in others you might notice lower than 50%. You might also notice that savings are more evident when higher volumes of data (hundreds of GB) is retained with long term retention.
> - For file and image attachments, Dataverse long term retention doesn't reduce capacity consumed. In rare cases, depending on the file or image, you might experience negligible file capacity savings.

Solution aware retention policies

Dataverse retention policies are solution aware. Dataverse retention policies added to a solution are known as solution-aware retention policies. You can add multiple retention policies to a single solution. Retention policies are added to an unmanaged solution. This helps makers follow application lifecycle management (ALM) best practices when working with Dataverse retention policies.

When you include your retention policies in a solution, their definitions become portable, making it easier to move them from one environment to another, saving time required to create the retention policy. For example, you first develop a solution containing a retention policy in a development or sandbox environment. You then move that retention policy to a preproduction environment to test and validate that the solution works well and is ready for production. After testing is completed, the admin imports the solution into the production environment.

To add a retention policy to a solution, go to the Power Apps (make.powerapps.com), create or open a solution, and then follow these steps:

1. Select **Add existing** > **More** > **Other** > **Data Life Cycle Config** (display name renamed from RetentionConfig to Data Life Cycle Config to support both retention and deletion scenarios).

2. Select one or more retention policies to add to the solution.

The **Data Life Cycle Config** supports addition of both archival and deletion policies in a solution.

> **Note**
>
> - The data retained by retention policies isn't portable as part of solutions, only the retention policy definitions are. You must run the retention policy in an environment to retain the data in Dataverse long term storage.
> - Only retention policies created in Power Platform environments can be solution-aware.
> - When long term retention, is enabled through a solution import process, the related child tables aren't automatically enabled.

You create a solution before you add a retention policy to it. Exporting and importing solutions containing retention policies is the same as with other solution components.

For more information about solutions and solution components, go to Solutions overview.

Next steps

Set a data retention policy for a table

Share your ideas

Last updated on 04/10/2026

Set a data retention policy for a table

Article • 05/10/2024

This article explains how to set up a data retention policy for a Microsoft Dataverse table.

Prerequisites

Theses prerequisites must be completed before you can set a table for long term data retention:

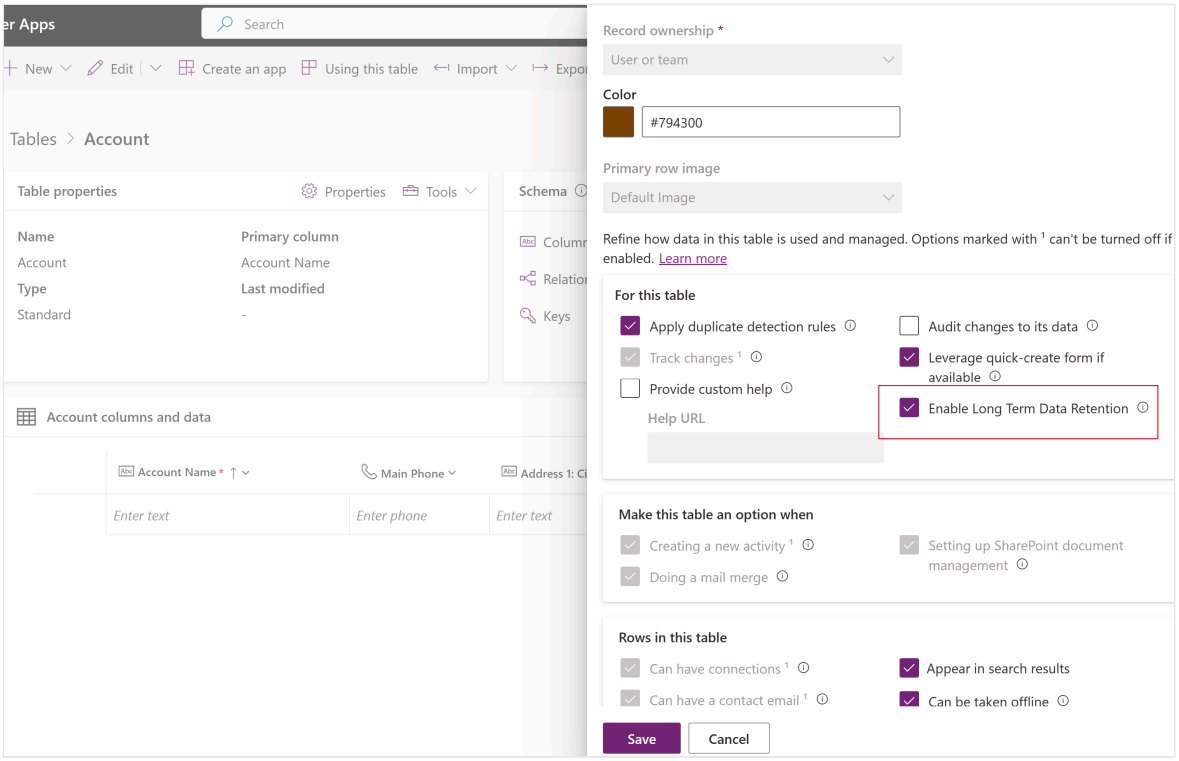
- Enable a table for long term retention
- Determine Dataverse views for policy criteria

Enable a table for long term retention

Before you set up a retention policy, enable the parent (root) table for long term retention.

> **Note**
>
> When a parent (root) table is enabled for long term retention, all the related child tables are automatically enabled. You can't disable retention for a child table when the parent table is already enabled for retention. For example, assume a custom table and notes table are child tables of the case table. These child tables can't be disabled for retention when the parent case table is enabled. A maker can always disable the parent table for long term retention and separately enable the child tables for retention. However, there's an exception to this related child tables behavior when long term retention is enabled through a solution import process. In this case, the related child tables aren't automatically enabled.

1. Sign into Power Apps, go to **Tables**, and then open the table you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **Properties** on the command bar, expand **Advanced options**, and then select **Enable long term retention**.



3. Select **Save**.

The parent root and any child tables will be available for policy setup in about 15-30 minutes after you save the table. The larger the number of child tables, the longer it takes to complete the process.

Determine Dataverse views for policy criteria

Dataverse views are used to determine which table rows (records) are marked for retention using a long-term retention policy. Before setting up a data retention policy, you must set up a Dataverse view as the criteria for retention.

Once you have a Dataverse view for use in a particular long term retention policy, you should test the view result. A good practice is to modify the view query with a `TOP N` statement and test it to ensure that the result set returned is as expected. This is always recommended especially when the number of rows is large.

For example, consider a Dataverse view created to select all closed cases from the year 2015. The number of cases could be potentially very large. Therefore, you should add a `TOP N`, such as *TOP 10*, to the view and test it to confirm the sample set of the retrieved cases.

When the data retention policy is run, rows from the parent table and all child tables are included for long term data retention. The parent cases might have associated rows from child tables, for example notes, task, and custom tables. The retention policy ensures that the case record and all its related child rows are marked and retained as long term in Dataverse. In addition, lookup values in the case table, as an example, are

denormalized with the ID and name values made available as part of the retained data. This allows in many scenarios for a single query to retrieve the relevant retained data.

> **Important**
>
> We recommend that you create a Dataverse view specific for identifying only the rows you want retained. Then, create and run an app that has the view to verify before using it as the criteria in your policy. For information about how to create a table view, go to Create and edit public or system views.

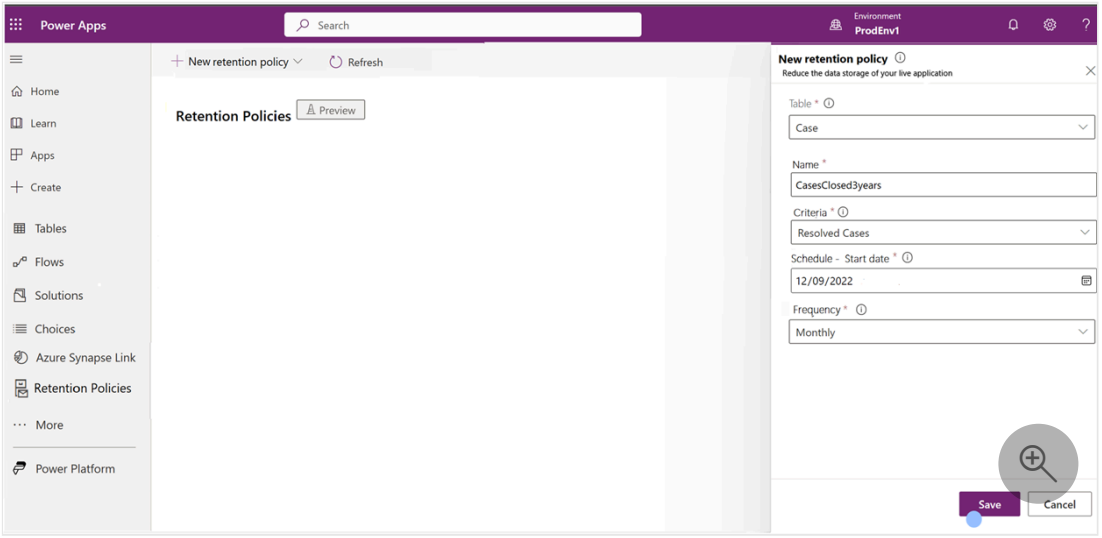
Setup a retention policy

Power Platform administrators set up retention policies.

> **Important**
>
> - Once data is retained as long term (inactive) data, it can't be moved back to the active data store.
> - When a retention policy is run, the process makes API requests in Microsoft Power Platform. These requests are counted towards the existing API requests available with your plan. More information: Requests limits and allocations
> - To complete this task you must have the Power Platform administrator role.

1. Enable a table for long term retention.
2. Sign into Power Apps, and then on the left navigation pane, select **Retention policies**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. On the command bar, select **New retention policy**.
4. Complete the following properties for the retention policy:
    - **Table**. Select the parent root table that will be used to set up a policy to retain historical cold data in long term storage.
    - **Name**. Enter a name for the retention policy.
    - **Criteria**. Select a view from the list. The view should have been previously created and tested to ensure the right historical data was selected. Determine Dataverse views for policy criteria
    - **Schedule - Start date**. Enter the date you want the policy to run for the first time. The policy runs during the off hours of your environment's region.

- **Frequency**. The choices available are **Once**, **Daily**, **Weekly**, **Monthly**, and **Yearly**.



5. Select **Save**.

Now you can view and manage the policy. More information: Manage data retention policies

> **Important**
>
> - The retention policy will not start for environments that have administration mode enabled.
> - All parent and associated child activities, emails, attachments are retained as a set when you set up a policy on a parent table. This is similar to bulk delete policies. For example, an explicit retention policy can't be set for Activity Pointer, Activity Party, and Attachment tables. Long term retention policies can't be set for those tables where bulk delete isn't supported.
> - The parent table and all related rows from child tables, of the parent, will be marked and stored in Dataverse long term storage.
> - Logic associated with an existing delete action of any table (parent and children) will always be executed during the policy run.
> - A policy run will take from 72-96 hours irrespective of the data volume. Data retention policies are given a low priority by the platform. Dataverse will always run the retention process in the background to avoid any impact on other platform services, such as apps and flows.
> - Consider a situation where two policies (Policy1 and Policy2) run on separate parent tables that have common child tables. Policy1 begins running prior to Policy2. If the status for Policy1 hasn't progressed beyond **Pending reconciliation**, then Policy2 status will remain at **Pending delete** status until

Policy1 progresses from **Pending reconciliation** to **Pending delete**. This behavior occurs to prevent deletes on any common records from common tables in the two policies.

Known issue

If a parent table has many child tables, and the size of the parent or child tables is large, you might encounter timeouts. For example, when the number of tables is large in a cascade relationship chain, such as twenty-five or more. To mitigate this, enable a few child tables separately first. Then go back and enable the parent table.

Next steps

Manage data retention policies
Share your ideas
For developers: Long-term data retention

Feedback

Was this page helpful? Yes No

Provide product feedback

Manage data retention policies

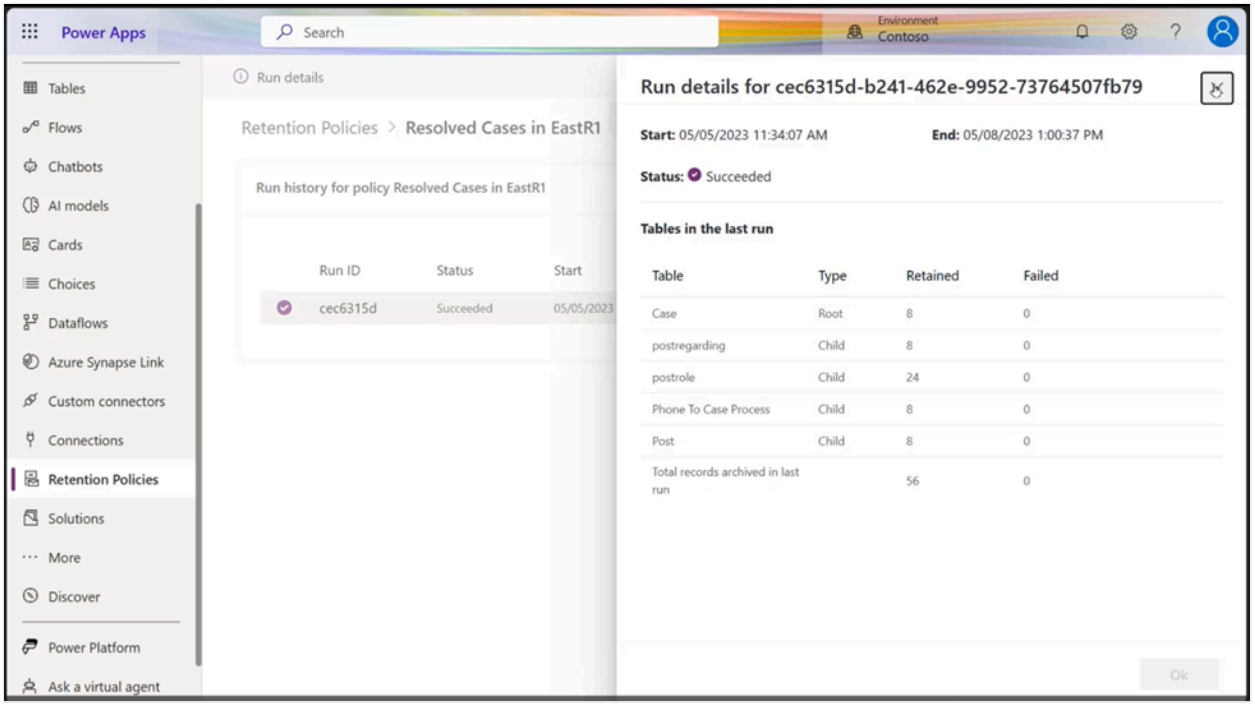
Article • 02/29/2024

Use the retention policies dashboard to view and manage retention policies. From the dashboard, view the policy run history, update policy details, deactivate, and delete a policy.

View and manage retention policies

1. Sign into Power Apps, and then on the left navigation pane select **Retention policies**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select a retention policy from the list, and then on the command bar select **Policy details**.
3. The following retention policy actions are available:

- **History**. Displays the retention policy run status, run start, run end and table. For every run, you can view all tables (root and child tables) and the number of rows retained in the run.
- **Policy details**. Displays the properties for the policy where you can view and edit the policy name, criteria, and frequency.
- **Deactivate**. Stops the retention policy from running by disabling the policy schedule. To activate a deactivated policy, select **Policy details**, set the policy **Schedule**, and then select **Update**.
- **Delete**. Removes the policy.



Policy run status

Expand table

Status	Description
Scheduled	The policy has been scheduled to run.
In progress - Retention	The process of moving and changing the data state from active to non-active (retained) for the rows in the parent root table and all child tables.
Pending Reconciliation	Waiting to reconcile the retained rows in Dataverse long term retention.
In progress – Reconciliation	During this stage, ensures no data loss by reconciling the retained rows with the original rows before delete from active.
Pending Delete	Waiting to delete all retained rows.
In progress – Delete	Delete of retained rows from applications.
Completed	Retention process completed successfully.
Failed	The retention process failed.

View details on failed records

Query the Dataverse table `Retentionfailuredetails` for error details.

Expand table

Column name	Description
`Operationid`	The policy `runid` visible in the long term retention dashboard for a specific policy run.
`entitylogicalname`	Name of the table containing the record.
`recordid`	Unique identifier of every record in the table.
`message`	Detailed error message.

Bulk delete long term retained data

Dataverse bulk delete with the Dataverse SDK supports deletion of long term retained data. More information.

See also

View long term retained data
Share your ideas

Feedback

Was this page helpful? Yes No

Provide product feedback

View long term retained data

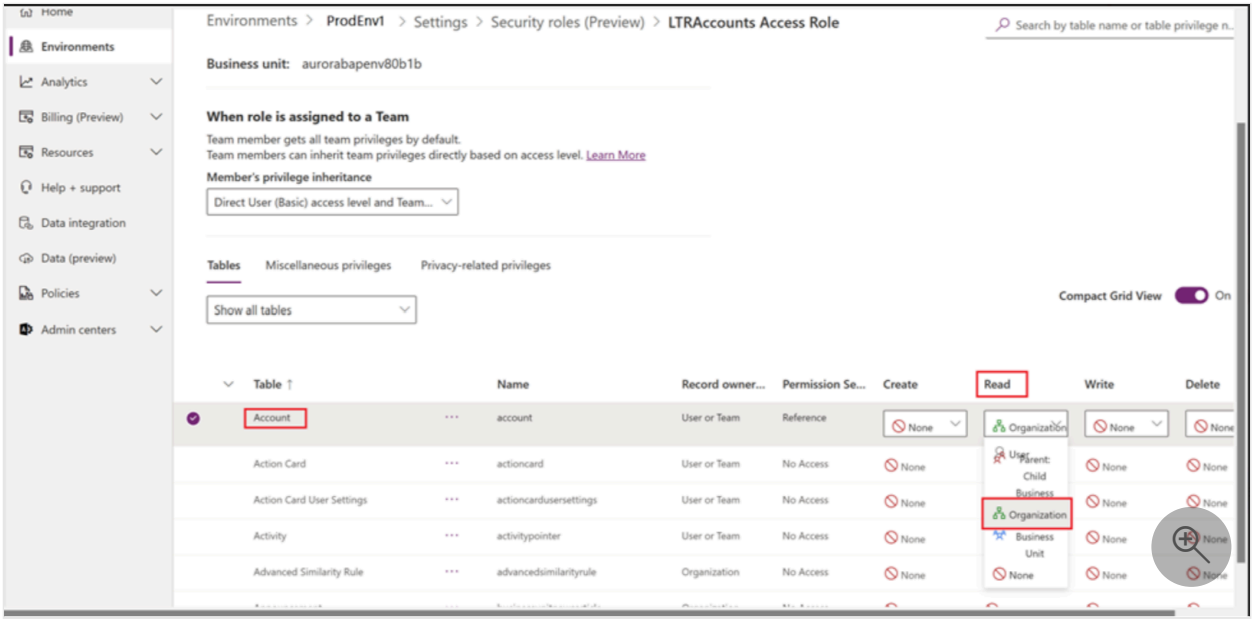
Article • 06/27/2024

You can view retained data from an advanced find query or by creating a Power Automate flow.

To view retained data in an environment requires the system administrator security role or other security role membership that includes organization scope read privileges to the table.

Grant privileges to view retained data

Imagine an auditor requires access to long term data retained for the accounts table. To provide the auditor access, a Power Platform admin creates a new role, for example a role named *LTRAccounts Access Role* and grants organization scope read privilege to the accounts table. Then add the auditor's Power Platform user account to the security role. When the auditor's job is complete, it's a best practice to remove the auditor from the security role.



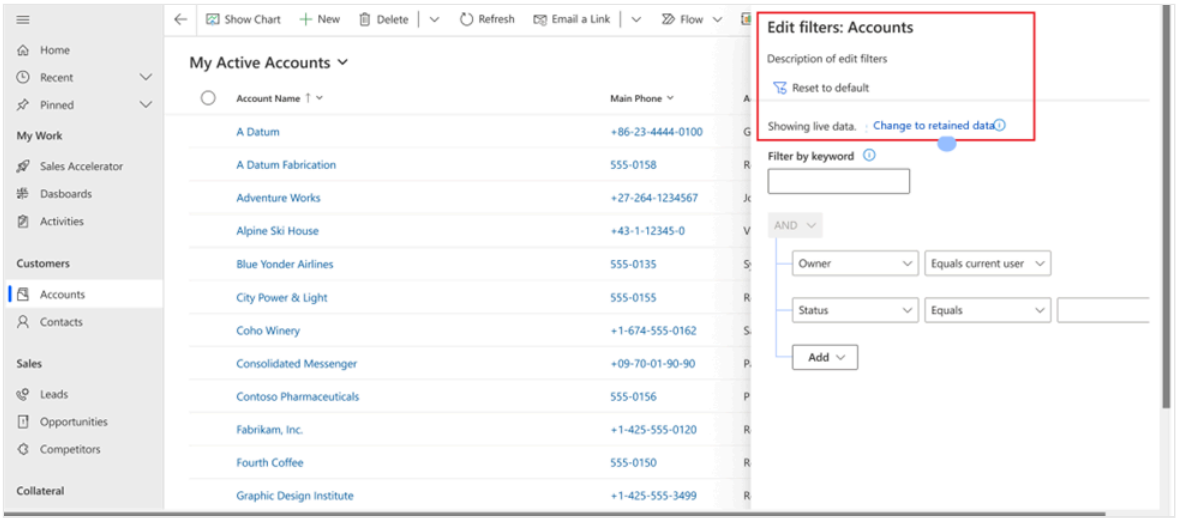
For more information about creating and editing Dataverse security roles, go to Create or edit a security role to manage access.

View retained data using edit filters from a model-driven app

> **Note**

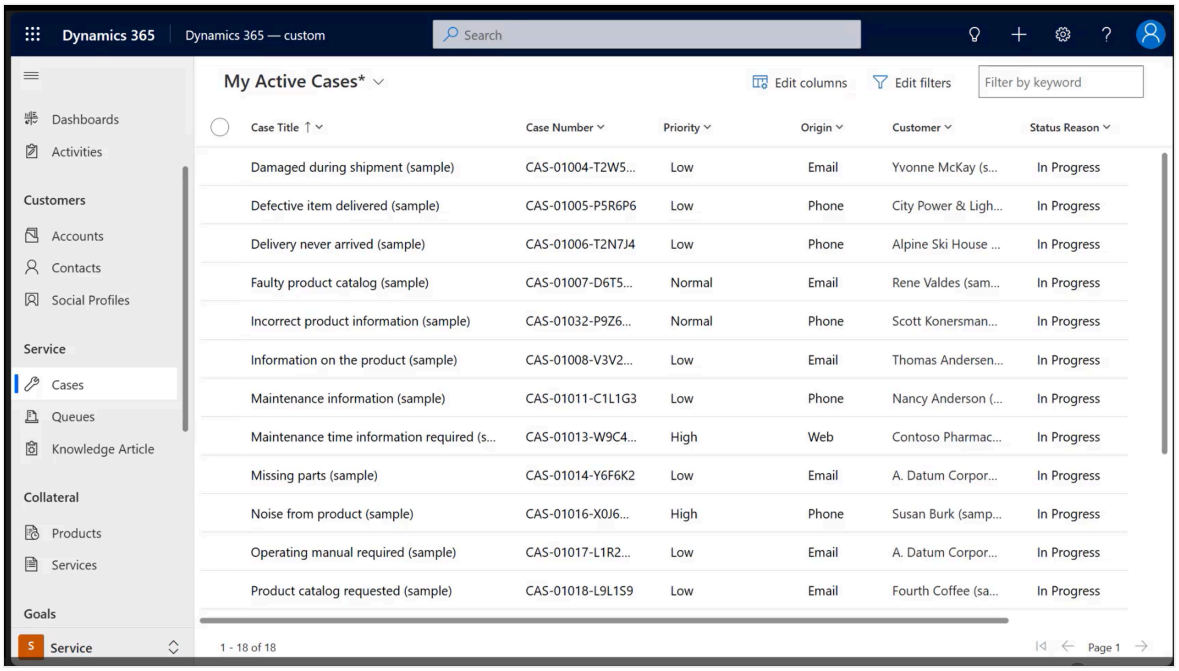
- You can't save or export the the view query results of the retained data for sharing with others. To share retained data, create a cloud flow to view Dataverse long term retained data.
- Edit filters doesn't retrieve table row attachments. To view attachment data, create a flow. More information: Create a cloud flow to view Dataverse long term retained data

1. Sign into Power Apps, play an app that includes a table that has retained data.
2. Open the view you want. From the view, select **Edit filters**.
3. Select **Change to retained data**.



> **Note**
>
> This changes the fetch element `datasource` attribute value to `"retained"`.

4. Select the tables and search filters you want, and then select **Apply**. The retained data is displayed in the read-only grid.



If you need to retrieve long term data from multiple related tables, such as the account table, which has an associated retained case table, first use advanced find retrieve the retained case row. Then use the `Casenumber` column and use advanced find to retrieve the account row that contains the case number.

More information: Advanced find in model-driven apps

View retained data using a flow

Create a Power Automate cloud flow to create an Excel file of the retained data from a FetchXML query and send as an email attachment. More information: Create a cloud flow to view Dataverse long term retained data

> **Note**
>
> If the retained data includes attachments from the annotation table, the returned value is a base64 representation of the file. Large files might cause the cloud flow action to time-out or to exceed its output message size limit.
>
> To workaround this behavior, use the `ExportRetainedData` message Web API to ExportRetainedData action or SDK for .NET using Azure Functions or other custom development options.

Limitations for retrieval of retained data

These restrictions are enforced by Dataverse for each environment:

- Up to five users can query and retrieve retained data at the same time.
- Up to 100 queries per day are allowed for each environment.
- Any single request from advanced find, Power Automate cloud flow, or Dataverse OData public API is considered as one query.
- Queries are allowed on one table at a time. Joins and aggregation functions aren't allowed. Consider options with Microsoft Fabric for complex queries and Power BI options. More information: View retained data with Microsoft Fabric
- Retained data includes lookup data. Lookup values in the table are denormalized with ID and name value.

View retained data with Microsoft Fabric

You can view the active (live) and inactive (long term retained) application data in Dataverse using Microsoft Fabric. To do this, link your Dataverse environment to Fabric. More information: Link your Dataverse environment to Microsoft Fabric and unlock deep insights.

When your long term retention policy is run successfully, you can access the active and inactive Dataverse data. The limitations applied to retrieval of retained data don't apply to this mode of access.

You can explore the data with SQL endpoint and query Dataverse data with SQL and generate views in Fabric. You can also create Power BI reports. More information: Work with Dataverse data and generate Power BI reports

The Dataverse table column `msft_datastate` can be used to filter the data with the SQL `WHERE` clause:

- Inactive application data: `WHERE msft_datastate=1`
- Active (live) application data: `WHERE msft_datastate=0 or msft_datastate=NULL'

Known issues

Personal views of retained data

Saving the query results of the retained data as a personal view isn't supported. Although users can save the query results of the retained data as a personal view, the view doesn't return results.

See also

Dataverse long term data retention overview

Share your ideas

Feedback

Was this page helpful? Yes No

Provide product feedback

Dataverse long term data retention FAQ

This article provides information on frequently asked questions about long term data retention with Microsoft Dataverse.

Can a data retention policy be set only on a particular table and not the related tables?

No. Admins can set up a retention policy on a root table, such as the Dynamics 365 apps accounts table. Dataverse automatically sets all related child tables for data retention. So if the account row has related task rows, all task rows are also set for data retention. Admins can always set up a retention policy on a specific table with no related tables.

What about my existing custom Dataverse native delete action logic to ensure business rules are followed when any live data is deleted?

All custom logic with Dataverse native delete actions are always honored during the data retention process.

Why can't I view some data in a table while a particular policy is being executed in the background that was a part of the policy criteria?

Until all records are purged, records are available for viewing. Retention is a multi-step process that includes reconciliation to confirm that all the records within the criteria are retained for long-term before delete from Dataverse. **We recommend that you set up long term retention only for nonactive records, or records that aren't actively required for your business operations.**

If a policy run has errors, are the errors available for admin to view?

Yes. Errors from a policy run are available in the run history details. More information: data-retention-manage.md#view-and-manage-retention-policies

Will Dataverse schema changes be handled by Dataverse long term retention? What if a maker adds custom columns or deletes columns to tables that have existing retained data in Dataverse long term retention?

The changes are reflected in the long term retained data. When retention policies run, the new custom columns contain data for the new rows. The data that was retained prior to these changes won't have any value in these columns. If a column is deleted in the live table, the column remains in the long term retained data with a value for all new rows.

What if a maker adds custom child tables?

Dataverse long term retention ensures that the new child tables are a part of the next retention policy run.

Can a retention policy run be stopped?

No. A policy being executed currently can't be stopped.

Will running data retention jobs impact the performance of my business applications?

No. Data retention jobs are a background process with secondary priority to your apps.

Are there any limits on how many records can be retained in a single policy?

No. While running a policy one-time is an option, we recommend you set up policies that run on a recurring basis to ensure historical data is efficiently managed.

How long will a policy run take to complete?

Data retention policies are given a low priority by the application. Dataverse runs the process in the background typically during off hours to avoid impacting the live application processes. The policy run can take from three or four days up to two weeks for very large data volumes. An option is to schedule long term retention policy runs on a regular cadence like monthly or quarterly, unless the scenario absolutely requires longer durations, such as yearly schedules.

Where can I view the capacity savings?

The Power Platform capacity reports displays the total capacity consumed. The reduction in database capacity consumed, after running a Dataverse long term retention policy, depends on the number of rows and column data types involved in the retention process. For production environments, the reports could take a few days to a week to reflect the fully reduced savings. For nonproduction environments, the reports could take up to two weeks to reflect the savings.

What is expected for the performance when accessing retained nonactive data from Dataverse?

Retained data reads are significantly slower than live data reads. Notice that long term data retention isn't a feature for transactional operations, but should be used for rarely accessed data required for compliance.

Are data retention policies solution aware?

Yes. You can create a solution and add the policy from an environment, such as a development environment, to a test environment, and then to a production environment. More information: Dataverse long term data retention overview

Is the audit table supported for long term retention?

No, the audit table isn't currently supported for long term retention.

In which regions is Dataverse long term retention available?

The feature is currently available in all public cloud regions. GCC, GCC High, and DOD aren't currently supported.

Can data be moved from Dataverse long term store to my own data lake?

This isn't currently supported.

Does Dataverse long term retention support Dual Write apps?

This isn't currently supported.

Can the one Dataverse connector be used with Power BI for reporting long term retained data?

This isn't supported. However you can access live and retained data with Microsoft Fabric and build Power BI reports.

Can I move data from Dataverse long term store back to the Dataverse live (active) store?

No.

I use Azure Synapse Link. Does retaining data with Dataverse long term retention remove the retained data from my own data lake?

No. You can create an Azure Synapse Link for Dataverse to move all of the application data into your own data lake. Then set up a long term retention policy to retain data. Dataverse long term retention doesn't delete retained data already in your own data lake. Dataverse APIs are available to delete retained data from Dataverse long term retention. This capability is the same for all Dynamics 365 customers including finance and operations applications.

Do retention policy runs consume API requests?

Yes. When a retention policy runs, the process makes API requests to Microsoft Power Platform. These requests are counted towards your available API request limits.

Is customer managed key supported for long term retained data?

No, currently long term data retention isn't supported for customer manged key enabled environments.

Why is the status of my Dataverse long term policy deactivated by default when I create a new Dataverse long term retention policy?

Your environment doesn't meet the licensing criteria. More information: Data retention overview Contact your account manager to discuss options on licensing.

I used Dataverse long term retention during the public preview phase without meeting the current licensing requirement. Will my tenant and environment still be able to use Dataverse long term retention?

Yes. Since your tenant and environment participated in the public preview, Dataverse long term retention is being made available for continued use.

Can multiple long-term retention policies run simultaneously?

You can schedule multiple retention policies, however only one policy is executed at a time.

How do I ensure there's no data loss during long-term retention?

Prior to retention, a process validates that all data has been securely stored in the data lake. Only records that have been successfully reconciled are deleted from the live database. This validation helps maintain data integrity.

Create and edit virtual tables that contain data from an external data source

A virtual table is a custom table in Microsoft Dataverse that has columns containing data from an external data source. Virtual tables appear in your app to users as regular table records, but contain data that is sourced from an external database, such as an Azure SQL Database. Rows based on virtual tables are available in all clients including custom clients developed using the Dataverse web services.

In the past, to integrate the disparate data sources you would need to create a connector to move data or develop a custom plug-in, either client or server-side. However, with virtual tables you connect directly with an external data source at runtime so that specific data from the external data source is available in an environment, without the need for data replication.

Virtual tables are made up of three main components, a *data provider*, a *data source* row, and a *virtual table*. The data provider consists of plug-ins and a data source table. The data source is a table row in Dataverse, which includes metadata that represents the schema of the connection parameters. Each virtual table references a data source in the table definition.

Dataverse includes an OData Data and several other virtual connector providers, such as SQL Server, SharePoint, Fabric , and so on, that you can use to connect a common external data source. More information: OData v4 Data Provider configuration, requirements, and best practices and Create virtual tables using the virtual connector provider

Alternatively, developers can build their own data providers. Data providers are installed in an environment as a solution. More Information: Developer Documentation: Get started with virtual tables

Virtual table benefits

- Developers can implement plugins to read, update or delete external data using the Dataverse web services and Plug-in Registration tool.
- System customizers user Power Apps (make.powerapps.com) to create virtual tables that are used to access external data without writing any code.
- End users work with the rows created by the virtual table to view the data in columns, grids, search results, and Fetch XML-based reports and dashboards.

Add a data source to use for virtual tables

Developers create a custom plug-in to use as the data provider for a virtual table. Alternatively, you can use one of the available providers. More information: Create virtual tables using the virtual connector provider and OData v4 Data Provider configuration, requirements, and best practices

1. Sign in to Power Apps, and then select **Settings** > **Advanced settings**.
2. Select **Administration** > **Virtual Entity Data Sources**.
3. On the command bar, select **New**.
4. On the **Select Data Provider** dialog box, select from the following data sources, and then select **OK**.

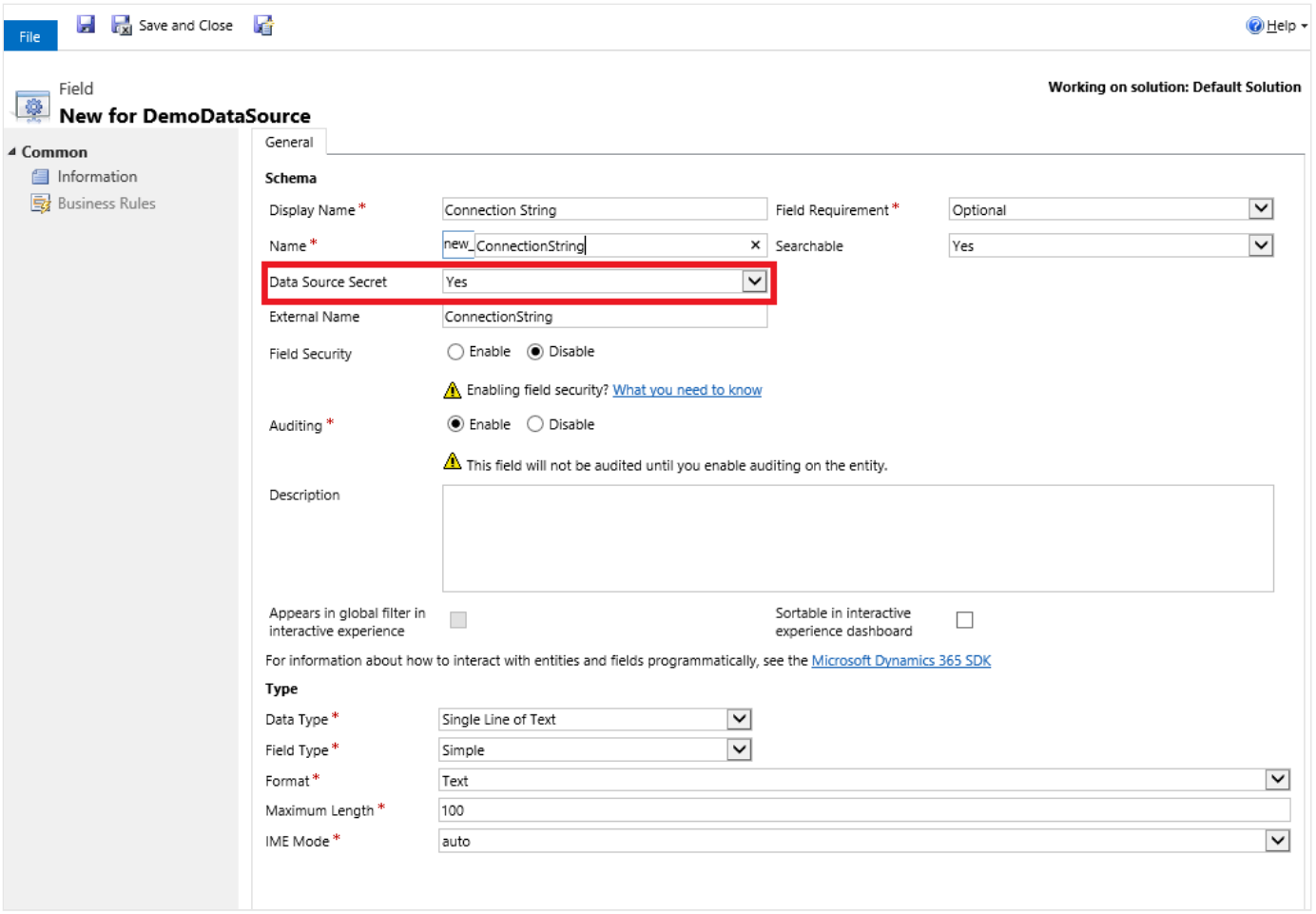
Expand table

Data Provider	Description
Custom data provider	If you've imported a data provider plug-in, the data provider will appear here. More Information Developer Documentation: Get started with virtual tables
OData v4 Data Provider	Dataverse includes an OData Data Provider that can be used with OData v4 web services. More Information OData v4 Data Provider configuration, requirements, and best practices

Add a secured column to a data source

You create columns for a data source in the same way as any other table. For data that is encrypted or sensitive, enable the **Data Source Secret** attribute on the custom column of the data Source. For example, to secure a column that contains a database connection string.

> **Note**
>
> The Data Source Secret attribute is only available with columns added to a Data Source form.



Create a virtual table

You create a virtual table just like any other table in Dataverse with the addition of a few extra attributes described here.

Open a solution

Part of the name of any virtual table you create is the customization prefix. This is set based on the solution publisher for the solution you're working in. If you care about the customization prefix, make sure that you are working in an unmanaged solution where the customization prefix is the one you want for this virtual table. More information: Change the solution publisher prefix

Open an unmanaged solution

1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. In the **All Solutions** list open the unmanaged solution you want.

Create a virtual table

1. In Power Apps (make.powerapps.com), create a new table. To do this, select **Tables** in the left navigation pane, and then select **New** > **Table** > **Virtual table**.
2. Select a connector, and then follow the instructions on your screen to create the virtual table. More information: Create virtual tables using the virtual connector provider and OData v4 Data Provider configuration, requirements, and best practices

> **Important**
>
> Several options, such as Access Teams, Queues, and Quick Create, aren't available with virtual tables. More Information: **Considerations when you use virtual tables**

Considerations when you use virtual tables

Virtual tables have these restrictions.

- Existing tables cannot be converted to virtual tables.
- By default, virtual tables contain only a Name and Id column. No other system managed columns, such as Status or Created On/Modified On are supported.
- Virtual tables don't support custom columns with the Currency, Image, or Customer data types.
- Virtual tables don't support auditing.
- Virtual table columns can't be used in rollups or calculated columns.
- A virtual table can't be an activity type of table.
- Dashboards and charts are not supported with virtual tables.
- Many features that affect table table rows cannot be enabled with virtual tables. Examples include queues, knowledge management, SLAs, duplicate detection, change tracking, mobile offline capability, column security, Dataverse search, and Power Pages solutions.
- Virtual tables are organization owned and don't support the row-level Dataverse security concepts. We recommend that you implement your own security model for the external data source.
- Column metadata properties that validate on update don’t apply to virtual tables. For example, a Whole Number column on a virtual table column may be set to have a minimum value of zero. However, since the value is coming from an external data source, a query will return values less than zero when retrieved from a virtual table. The minimum value property is not implied in the query. You would still need to filter the values to be greater than 0 if that’s what is desired.

- Virtual tables don't support change tracking and cannot be synchronized by using a Dataverse feature, such as the Data Export Service or Azure Synapse Link for Dataverse.
- Virtual tables that use the included OData v4 data provider are enabled on outbound port 443.
- Business process flows are not supported with virtual tables. More information: Unexpected error received when a user activates a business process flow

Unexpected error received when a user activates a business process flow

When a user attempts to activate a business process flow, they may receive an "unexpected error" message. Viewing the log file the following log entry is displayed.

ErrorCode: 0x80040216 Message: System.Web.HttpUnhandledException: Exception of type 'System.Web.HttpUnhandledException' was thrown. ---> Microsoft.Crm.CrmException: Business process flow cannot be enabled for Virtual Entity

This issue occurs because virtual tables don't support business process flows.

See also

Create virtual tables using the virtual connector provider
OData v4 Data Provider requirements and best practices
Create and edit tables
Configure virtual tables in Power Pages

Last updated on 04/22/2026

OData v4 Data Provider configuration, requirements, and best practices

Article • 06/21/2022

This topic describes how to configure the OData v4 Data Provider as well as the requirements and recommended best practices for using the OData v4 data provider to connect with an OData v4 web service.

OData v4 data provider best practices

- Microsoft Dataverse requires that all tables have an ID attribute, this ID is known as a unique identifier and the value must be a guid. You can only map ID columns to external columns with the `Edm.Guid` data type. You can't map an `Edm.Int32` data type to a Unique Identifier data type column in Dataverse.
- OData tables with nullable properties must be set to match the mapped column in the virtual table. For example, an OData table property with Nullable=False must have the mapped column in Dataverse **Field Requirement** attribute set to **Business Required**.
- For retrieve multiple queries, such as when you load data in to a grid, control the size of the dataset returned from the external data source by using the select and filter query parameters.
- If not already enabled, system administrators should enable plug-in tracing. Once enabled, all errors from the OData endpoint are captured in the plug-in trace log. More information: Administrator Guide: System Settings dialog box - Customization tab
- You can't use the OData v4 Data Provider to connect to another environment.

Data type mapping

The following table lists the OData table Data Model (EDM) data type mappings with Dataverse data types.

OData Data Type	Dataverse Data Type
`Edm.Boolean`	Two Options
`Edm.DateTime`	Date and Time
`Edm.DateTimeOffset`	Date and Time

OData Data Type	Dataverse Data Type
`Edm.Decimal`	Decimal Number or Currency
`Edm.Double`	Floating Point Number
`Edm.Guid`	Unique Identifier
`Edm.Int32`	Whole Number
`Edm.Int64`	Whole Number
`Edm.String`	Single Line of Text or Multiple Lines of Text

OData EDM data types that are not supported for mapping with virtual tables

- `Edm.Binary`
- `Edm.Time`
- `Edm.Float`
- `Edm.Single`
- `Edm.Int16`
- `Edm.Byte`
- `Edm.SByte`

Add a data source using the OData v4 Data Provider

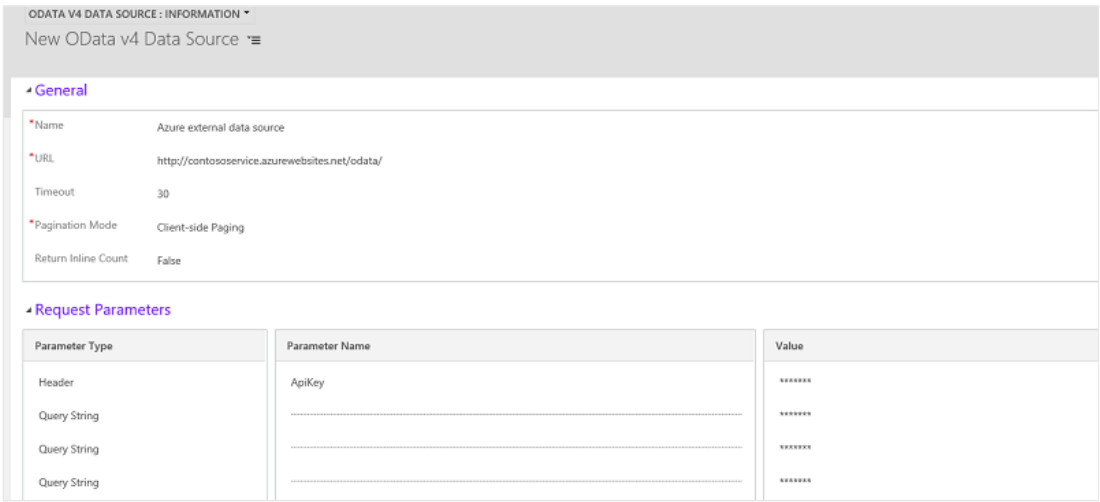
This procedure shows you how to use the out-of-box OData Data Provider to use as the virtual table data source.

1. Go to Settings > **Administration** > **Virtual Entity Data Sources**.
2. On the actions toolbar, click **New**.
3. On the **Select Data Provider** dialog box, select from the following data sources, and then click **OK**.
    - **OData v4 Data Provider**. Dataverse includes an Odata v4 data provider that can be used to connect to data sources that support the OData v4 open standard.
    - *Custom data provider*. If you've imported a data provider plug-in, the data provider will appear here. More information: Developer Documentation: Get

started with virtual tables

4. On the **New Data Source** properties page, complete the following columns, and then save the row.

   - **Name**. Type a name that describes the data source.
   - **Uri**. If you are using the OData Data Provider, enter the uri for the OData web service. For example, if you are using the OData provider to connect to a web service hosted in Azure, the URI can look similar to `https://contosodataservice.azurewebsites.net/odata/`.
   - **Timeout in seconds**. Enter the number of seconds to wait for a response from the web service before a data request time-out. For example, enter 30 to wait a maximum of thirty seconds before a time-out occurs.
   - **Pagination mode**. Select whether to use client-side or server-side paging to control how query results are paged. The default value is client-side paging. With server-side paging, the server controls how results are paged by using the $skiptoken parameter, which is added to the query string. More information: Skip Token System Query Option ($skiptoken)
     - **Return inline count**. Returns the total number rows in the result set. This setting is used to enable next page functionality when you return data to a grid. Use a value of false if your OData endpoint doesn't support the OData $inlinecount parameter. The default value is false.
   - **Request Parameters**. Optionally, you can add custom header or query string parameters used to connect to the OData web service, such as authentication parameters to the external service. Click **Query String** to toggle between header and query string parameter and value. Up to 10 header or query strings can be added.



See also

Create and edit virtual tables that contain data from an external data source

Create virtual tables using the virtual connector provider

Virtual tables enable integrating data from external data sources by seamlessly representing that data as tables in Microsoft Dataverse, without data replication. Solutions, apps, flows, and more can use virtual tables as if they were native Dataverse tables. Virtual tables allow for full create, read, update, and delete privileges unless the data source they're connecting to specifically forbids it. More information about virtual tables: Create and edit virtual tables that contain data from an external data source.

This document covers the new experience using Power Apps (make.powerapps.com) to create virtual tables using the following virtual connector providers:

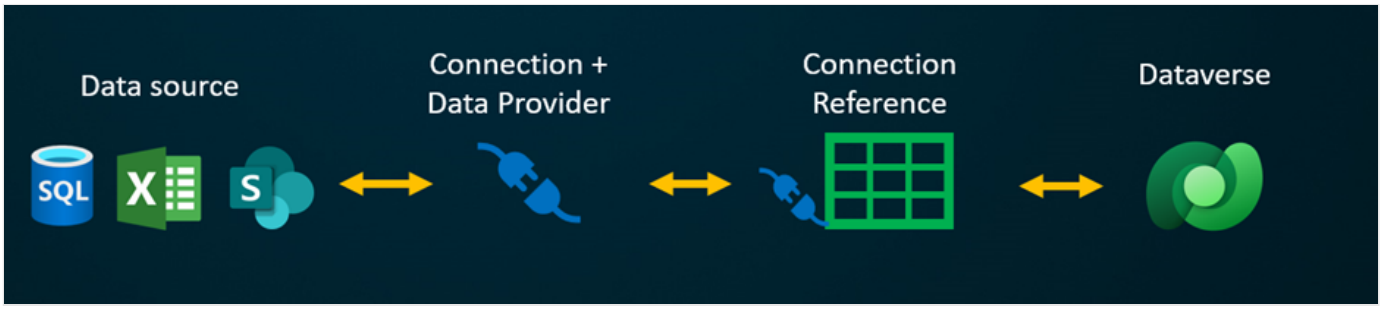
- SQL Server
- SharePoint
- Microsoft Fabric
- Salesforce
- Oracle Database
- Snowflake
- PostgreSQL
- Azure Databricks

These virtual connector providers use a Power Platform connector. More information: Connector reference for virtual connector providers used with virtual tables

You can create a virtual table for Excel using a legacy process with a virtual connector provider. More information: Create virtual tables using Excel in Microsoft Dataverse

Overview

Virtual tables include the following components:

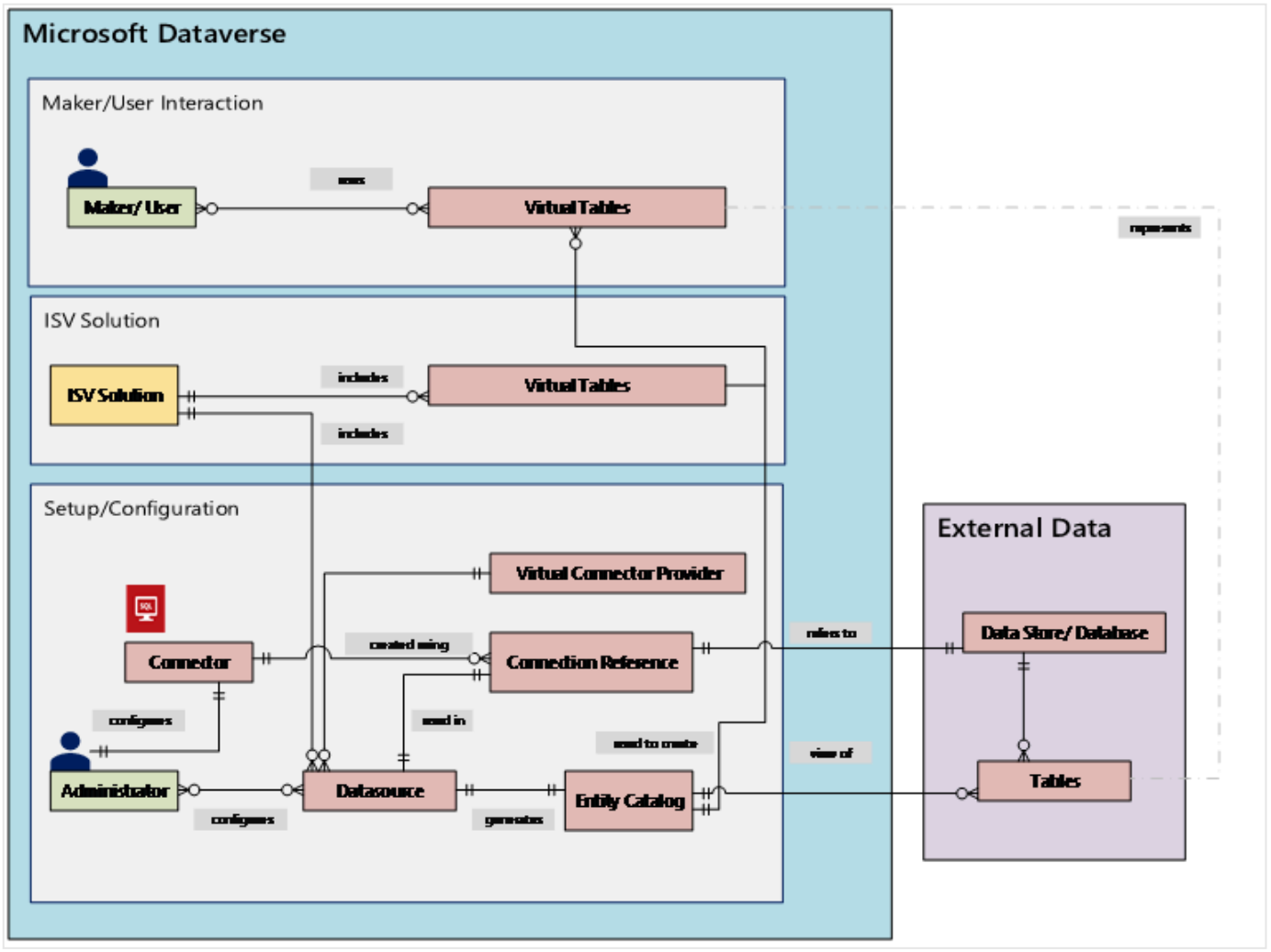


- Data Source – the location where the external data is stored.
- Data Provider – defines the behavior of the virtual table.
- Connection – sets up the ability to connect to the data source and authentication.
- Connection reference – provides a way for Dataverse to use the connection to the data source.

If you were to create a virtual table using a custom data provider, you need to write plugins that define how every Dataverse API would interact with the API on the system where the data is stored. This is a long process that requires knowledge of coding. Virtual connector providers streamline the creation experience by automating some of the creation for you and removing the need to use code to create the virtual tables.

When you establish a remote connection to an external source using a connector data source, the virtual connector provider automatically retrieves a list of all the available tables and lists by retrieving table definitions (metadata) from the external data source. You then select these tables and lists to generate the virtual table.

The underlying data source is key for allowing the provider to establish an authenticated remote connection to the external data. It uses a connection reference that stores pertinent details regarding the external source. The information stored in the connection reference is specific to the connector type and the connection it refers to.



When setting up the connection and connection reference for your data sources, specific information is needed. For example, the SQL Server connector needs server name, database name, the authentication method, username, password, and (optionally) gateway connection details. Each external data source needs a connection reference defined to create the virtual table. When using the Power Apps (make.powerapps.com) experience, the connection reference can be generated automatically for you unless you wish to provide custom naming.

The connector permissions enforce the ability for organizational users to access and operate on the virtual table. The connection can be shared with one user or can be shared with the entire organization. This allows users to access and operate virtual tables using a shared connection. By using security roles, virtual table access can be restricted to a specific set of users within your organization. You can even specify which roles have create, read, update, or delete privileges in this way.

Application lifecycle management (ALM) is supported for virtual tables created using the virtual connector provider. You can even create the virtual tables from directly within a solution when using Power Apps (make.powerapps.com). Virtual tables should be part of a managed solution along with the connection reference to distribute the solution. The solution can have other components, such as a model-driven app that uses virtual tables.

More information about application lifecycle management (ALM) and solutions:

- Application lifecycle management (ALM) in Microsoft Power Platform
- Solutions overview

Prerequisites

To create a virtual table, you must have a Microsoft Dataverse license through Power Apps or Microsoft Dynamics 365. Microsoft 365 or Teams licenses can't be used to create virtual tables.

Create a virtual table in Power Apps

Creating a virtual table in Power Apps (make.powerapps.com) using the virtual connector provider includes the following steps:

1. Choose to create a table using an external data source
2. Create the virtual table
3. Create and select the connection reference (optional)
4. Choose your connection details and select your data, such as SQL or SharePoint.
5. Configure your data
6. Configure column and table names (optional)
7. Complete the setup

Choose to create a table using an external data source

These steps describe how to create a virtual table from a solution. Use similar steps to create a virtual table by going to **Data** > **Tables**.

1. Sign into Power Apps, and then select **Solutions** in the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Create a new solution or open an existing unmanaged solution.
3. On the command bar, select **New** > **Table** > **Virtual table**.

Create the virtual table

Watch a short video showing how to create a virtual table with the virtual connector provider.

https://learn-video.azurefd.net/vod/player?id=130f38be-01d0-4321-8681-a7780231b94e&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fcreate-virtual-tables-using-connectors

1. In the **New table from external data** wizard you can either select an existing connection if you have one or choose to **Add connection**.

    - If you want to use an existing connection, select the connection you want, and then select **Next**.
    - If you have an existing connection but wish to create a new one, select **New connection** on the command bar.
    - If you have no connections and wish to create a new connection, select **+Add Connection** next to the connection type you want.

    > **ⓘ Important**
    >
    > Connections that are shared with you aren't available for use with this feature. Only connections created by the current user appear in the virtual table wizard.

2. You're directed to a new tab in your browser. Select your authentication method. Depending on the authentication method selected, you might be asked to provide credential information required to create the connection.

SQL Server

> **ⓘ Important**
>
> These will be the credentials used for all authentication for the virtual table so use credentials with the correct level of permissions with SQL Server.

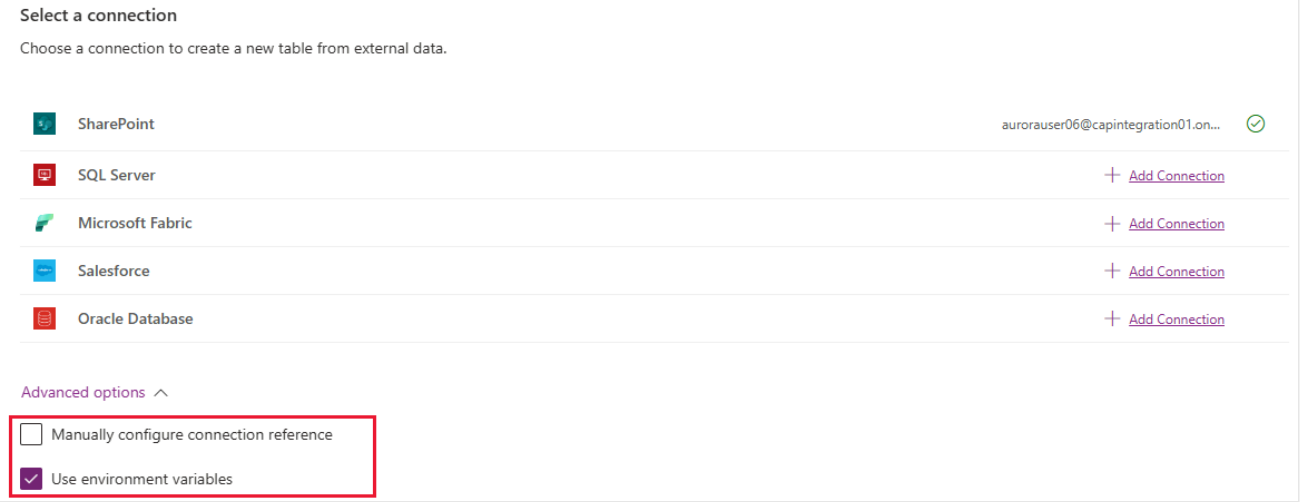
- **Microsoft Entra**: Select and sign in with your credentials.
- **SQL Server**: Server name, database name, user name, password, gateway (on-premises deployments only).

3. Optionally, select **Advanced options** to use a connection reference and/or environment variable.

    When you create a virtual table, a connection reference is automatically created for you with the virtual table. A connection reference is a solution component that contains information about the connector. However, you might want to create you own. To do this, select

Manually configure connection reference. More information: Create and select a connection reference (optional)

You can associate a virtual table with its own environment variable. Select **Use environment variables** to link the environment variable directly to the virtual table provider, offering flexibility to modify data sources when importing the virtual table into a new environment. More information: Environment variable



4. Select **Create**.

5. After the connection is created, go back to your browser tab with the wizard and select **Refresh**, and then select your connection.

Connection references and environment variables

Create and select a connection reference (optional)

When you create a virtual table, a connection reference is automatically created for you with the virtual table. A connection reference is a solution component that contains information about the connector.

However, you might want to create your own connection reference for the virtual table.

> ⓘ **Note**
>
> - The benefit of optionally naming your connection reference is because it can be easier to find later if you need details about it.
> - If you're using an existing connection you can select an existing connection reference or create a new one. Creating a new connection reference is only used if you want to

segment your virtual tables into completely unrelated solutions for use later.

To create a connection reference, when you're creating the connection for the virtual table, follow these steps:

1. Expand **Advanced options** and then select the **Manually configure connection reference** to create a connection reference for the virtual table.

2. On the **Connection Reference** page, select or name your connection reference, and then select **Next**.

    - If you chose SQL and Microsoft Entra ID as your authentication method, you're asked for your SQL server name and database name. Provide these and select **Next**.

Environment variable

Environment variables play a key role in the application lifecycle management (ALM) process, allowing for seamless movement of applications across different Power Platform environments. When creating a virtual table, you can associate it with its own environment variable. To take advantage of this functionality, expand **Advanced options**, and then select **Use environment variables** when choosing a connection for your data source during a virtual table create.

Environment variables with virtual tables recommendations

- Create or update a virtual table in the context of a solution.
- If an existing virtual table with environment variable needs to be added to a solution, then the environment variable related to this virtual table needs to be explicitly added to the solution. From the **Solutions** area in Power Apps, select **Add existing** > **Environment variable**, and then select the environment variable related to the virtual table. After this step, select the environment variable and then select **Advanced** and add the required objects.
- If a virtual table is created without an environment variable specified, you must re-create the virtual table and select the environment variable option.

Environment variables with virtual tables limitation

- Environment variable support with virtual tables currently only work with SharePoint and SQL virtual connectors.

Configure your data

If you're creating a SharePoint virtual table, you're asked to enter the URL of your SharePoint site or select from your most recently used SharePoint sites. The most recently used list is populated by gathering information about your recently used sites using Microsoft Graph and your Microsoft Entra credentials. If you're pasting the SharePoint URL, only include the information up to the site name, such as https://microsoft.sharepoint.com/teams/Contoso.

1. A page is displayed where you can either search your data source for a specific table or list, or select a table or list from the provided list.

2. Select the check box if you want to configure the table name, column names, and primary field.

3. Select **Next**.

Configure table and column names (optional)

When you create a virtual table, by default you can choose to change the suggested table and column names. To do this, follow these steps:

1. Select **Configure table and column names that will be used in Dataverse**, accept, or change the following Dataverse table properties:

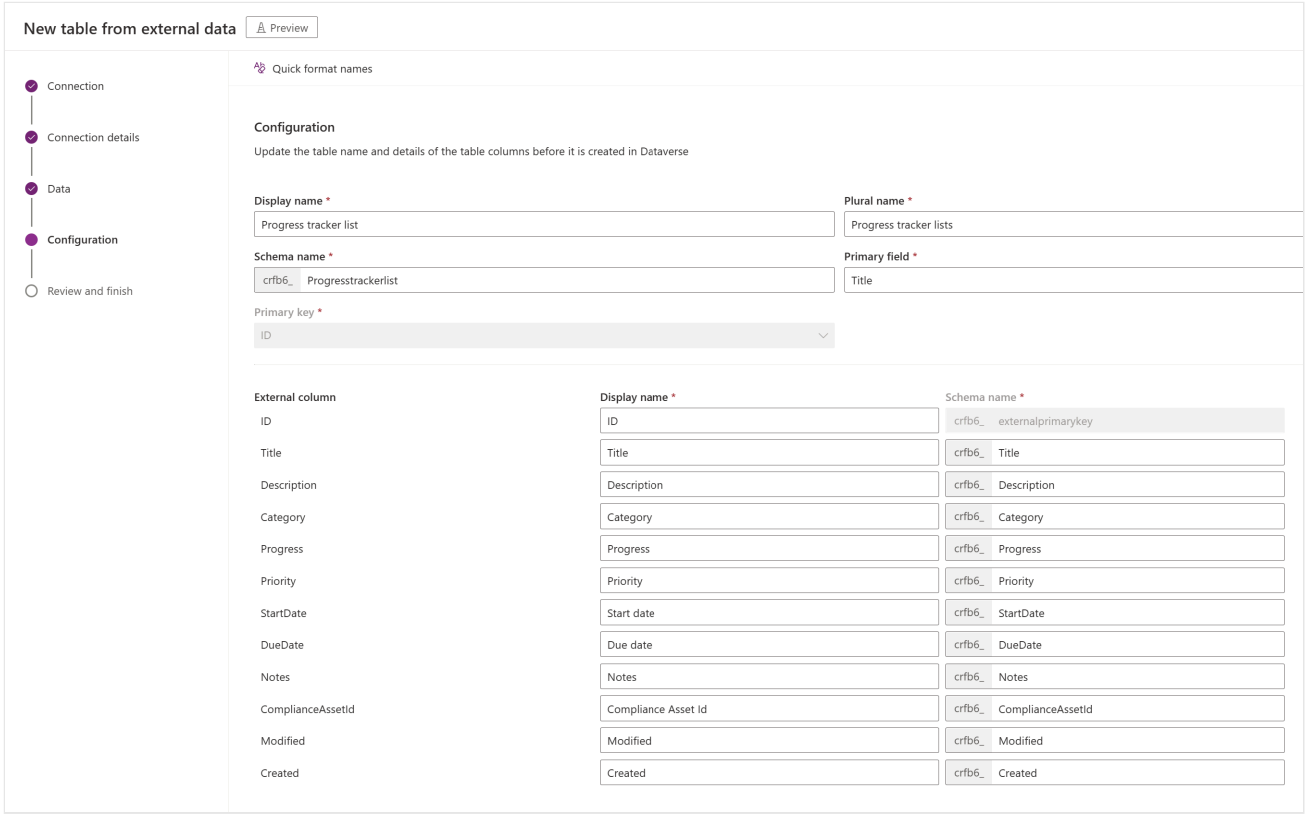
    - **Display name**: The name that is used to identify your virtual table.
    - **Plural name**: The plural of the virtual table name, used in appropriate situations where you refer to one or more record from the table, such as *Customer* is the table for multiple records refereed to as *Customers*.
    - **Schema name**: The logical name Dataverse uses for the virtual table, which includes the solution publisher prefix.
    - **Primary field**: This is the text value to be used when looking up records on your virtual table. Only string fields can be selected. A primary key is a required field but will be chosen by Dataverse.

2. In the **External column** area, choose if you would like to rename any of your external columns from the data source. The following fields are provided:

    - **Schema name** (read-only). This is the schema name of the column in the data source. This property is read only.
    - **Display name**. The name that's used to identify your column.
    - **Schema name**. The logical name Dataverse will use for the column that will include your solution publisher prefix. There's a **Quick Format Names** option on the page, this

provides suggested name changes and can be useful if you have a large number of fields that include prefixed values from your SQL server such as <tablename>.<column name>. For example, *Database12.Products* would change to **Products**.

> **Tip**
>
> Instead of entering the information, the **Quick format names** command provides suggested name changes and can be useful if you have a large number of fields that include prefixed values from your SQL server, such as *tablename.column name*. For example, *Database12.Products* would change to *Products*.



3. Select **Next**

Complete the setup

1. The **Review and Finish** page shows you the table you're connecting to in your data source, and the table that will be created in Dataverse.

> **Note**
>
> Select **Choose a different table** takes you back to the table selection screen. selecting **Edit Configuration of Table** takes you to the **Configuration** screen.

2. If everything is correct, select **Next**.

Once the table is created, you're taken directly to your new virtual table where you can view your data and begin working with it.

> **Note**
>
> When you attempt to create a virtual table that already exists, you receive a message that the table already exists and that you will be re-creating it. You will not be able to change the primary field or schema name if this is the case. Re-creating the table will update any column changes that were made in the data source on the table.

Connector reference for virtual connector providers used with virtual tables

To learn more about supported actions and limitations with each connector, go to:

- Connector reference for the SQL Server connector
- Connector reference for the Microsoft Excel Online Business connector
- Connector reference for the SharePoint Online connector
- Connector reference for the Salesforce connector
- Connector reference for the Oracle connector
- Connector reference for the Snowflake connector
- Connection reference for PostgreSQL connector

See also

Setting up a virtual table relationship

Known limitations and troubleshooting with virtual tables

Developers Guide: Get started with virtual tables (entities)

Last updated on 04/17/2026

Create a virtual table using Excel in Microsoft Dataverse

06/10/2025

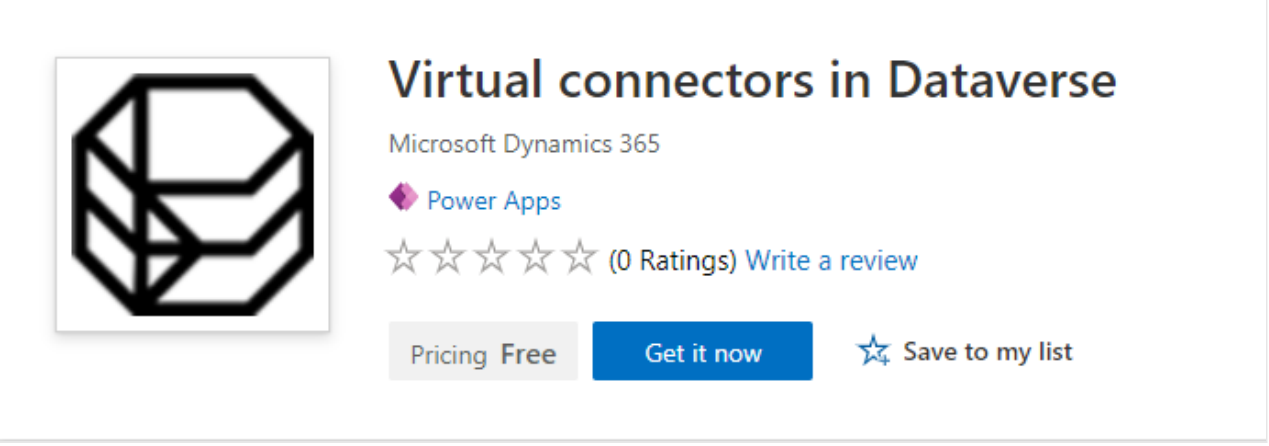
Create a virtual table to integrate data from an Excel file located in OneDrive to use as an external data source that seamlessly represents that data as tables in Microsoft Dataverse. Virtual tables allow you to view, create, and edit external data without replicating that data in Dataverse.

Prerequisites

- A Microsoft Dataverse license through Power Apps or Microsoft Dynamics 365. Microsoft 365 or Teams licenses can't be used to create virtual tables.
- The Excel file you use as the data source must be stored in OneDrive.
- The Excel file must contain data formatted as a table.
- The Excel file must contain a GUID column. More information: Create the connection for the Excel virtual connector provider

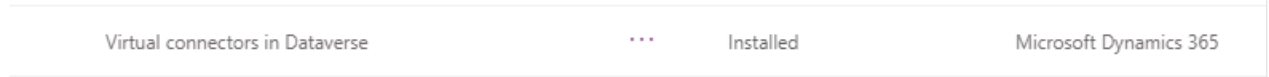
Download and install the virtual connector provider for Excel

1. Download and install the virtual connector. Go to the Microsoft commercial marketplace and search for `Virtual Connector` or select the link to download the provider: Virtual connectors in Dataverse

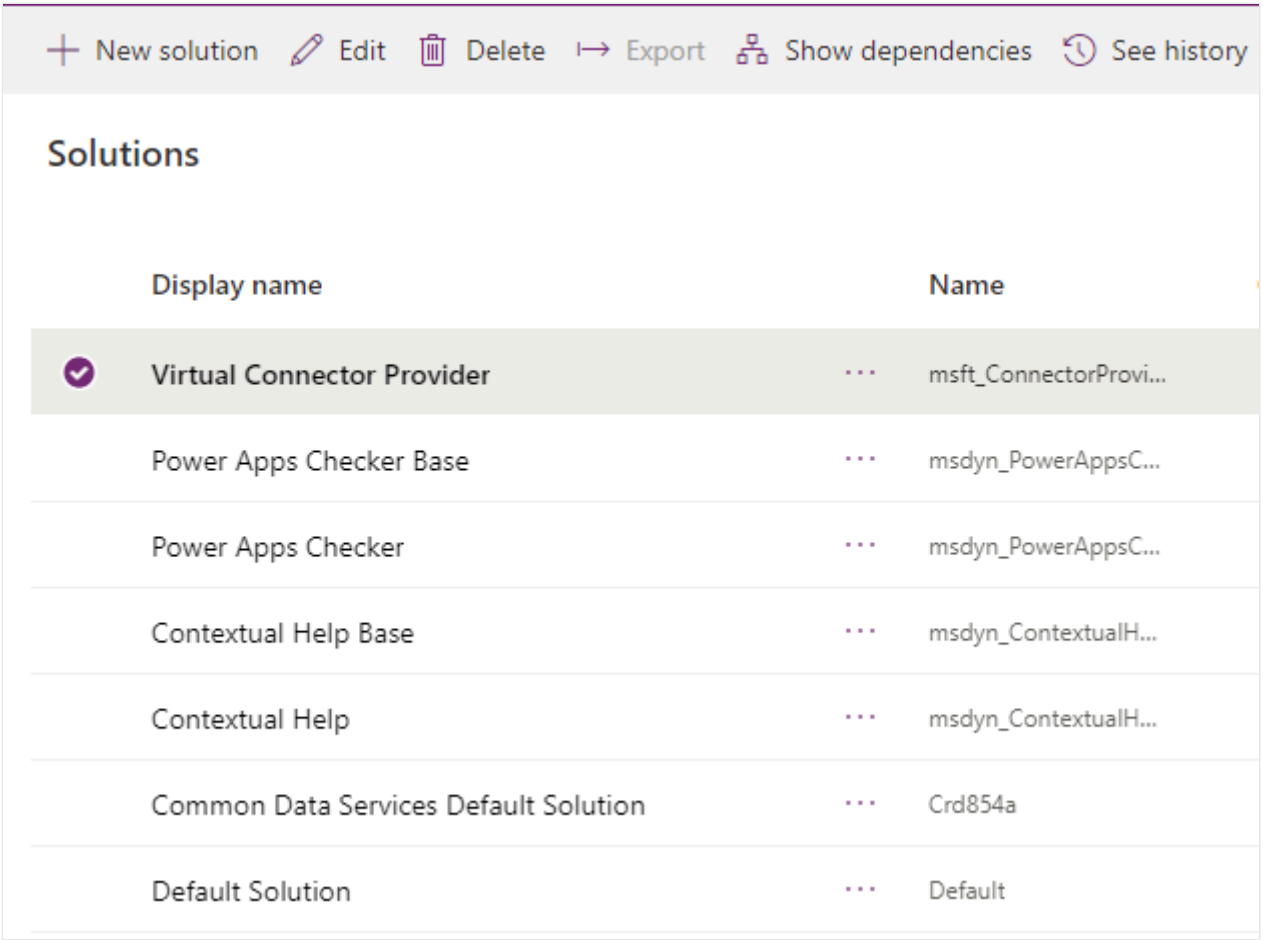


2. Select **Get it now**. In the sign-in dialog, enter work or school account email. If you agree to the terms and conditions, select **Continue**. The Power Platform admin center opens automatically.

3. Select the environment where you want to install the solution. If you agree to the terms and conditions, select **Install**. Once the installation is complete, you see the **Virtual connectors in Dataverse** app installed under **Environments -> [your environment name] -> Dynamics 365 apps**.



4. You should also see the **Virtual Connector Provider** solution and other solutions enabled in the Power Platform environment.

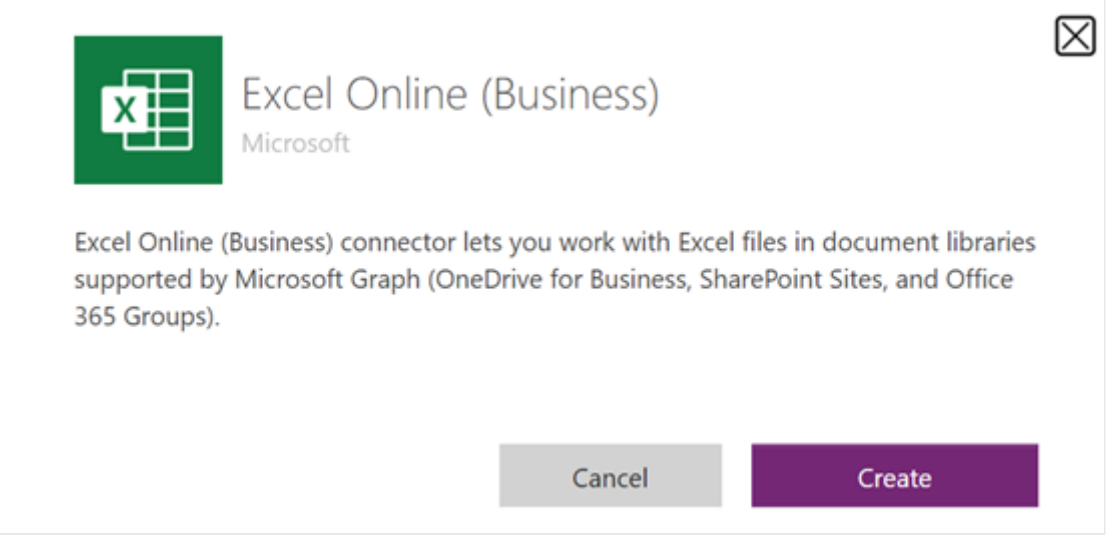


Create the connection for the Excel virtual connector provider

Watch a short video showing how to create a virtual table with the Excel virtual connector provider.
https://learn-video.azurefd.net/vod/player?id=d418205f-afcd-4a54-903d-9ac975868f5c&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fcreate-virtual-tables-using-excel

1. Go to Power Apps (make.powerapps.com), and select the environment in which you would like to set up the virtual table.

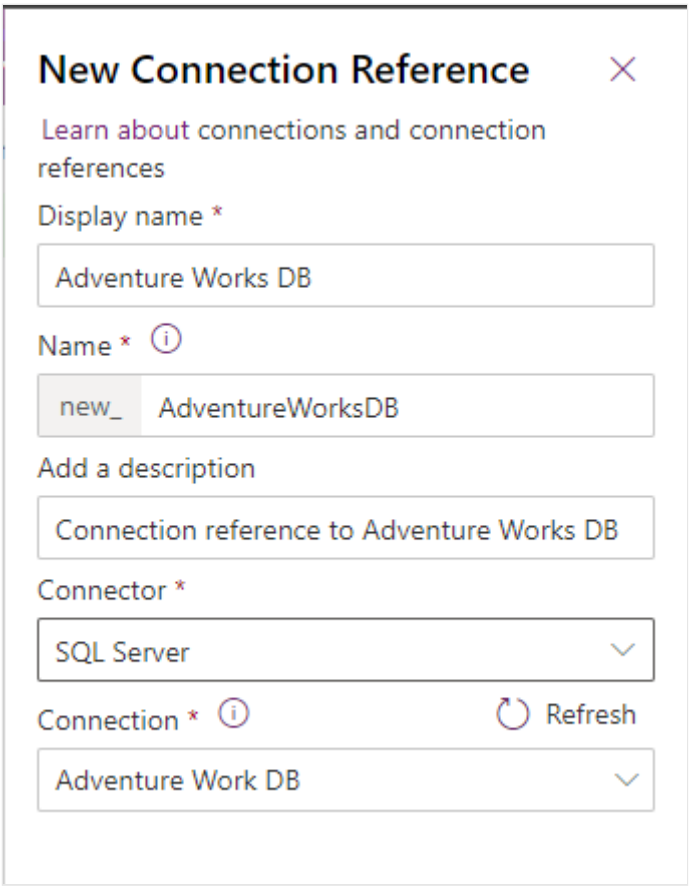
2. In the left navigation pane, select **Connections**, and then select **New connection**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the **Microsoft Excel Online (Business) Virtual Connector** from the list of connections.
4. You're asked to provide additional details to connect to the data source.
5. Select **Create**, your current signed-in credentials will be used.



Create the connection reference

You must create your own connection reference when you use the Excel virtual connector provider.

1. Go to **Solutions**.
2. Select the **Default Solution** or any other existing solution you have in your environment to create the virtual table.
3. Select **New** and then select **Connection Reference.**
4. Enter **Display name**, select the connection you created for the **Connectors** option and then select the data connection that you've created.

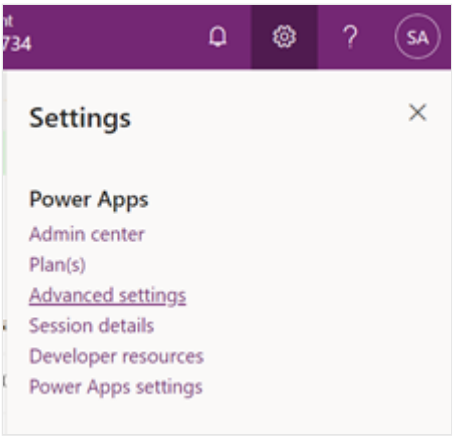


Create the data source

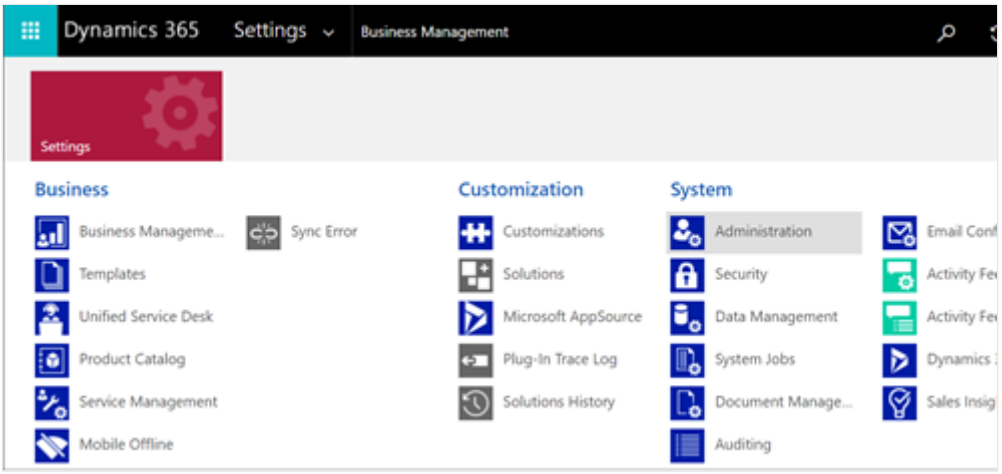
Now create the virtual table data source in Dataverse.

> **Note**
>
> - When you use the Excel virtual connector provider, an **Entity Catalog** table is generated by the system which provides a list of all the available tables using the table definitions (metadata) from the external data source. The entity catalog doesn't persist any information and always represents the external data source's current state. You can select tables from the entity catalog to create virtual tables. If you're working with multiple external data sources, an entity catalog is generated for each external source. More information: Entity catalog

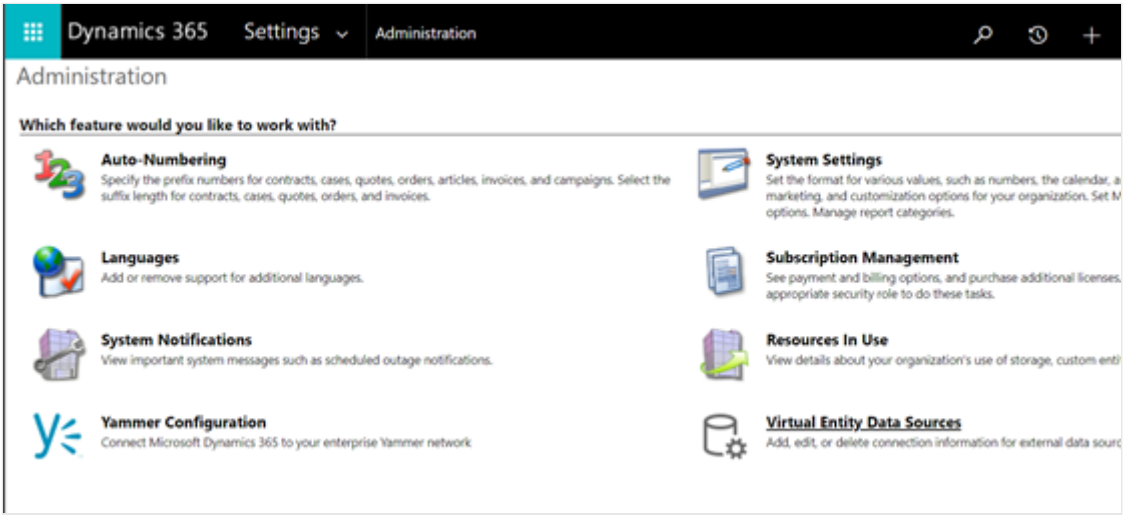
1. Select the **Gear icon** -> **Advanced Settings**.



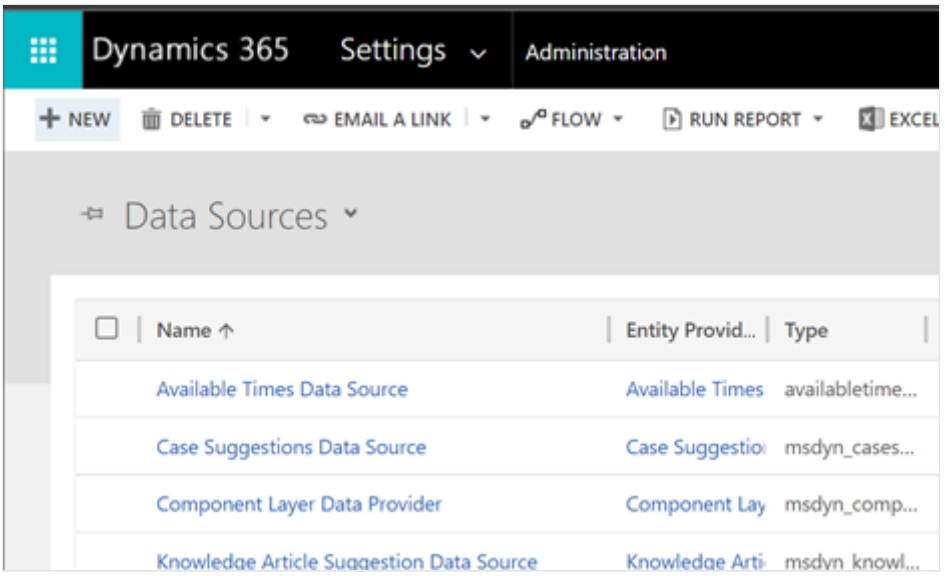
2. In the top navigation bar, select **Settings** and then **Administration**.

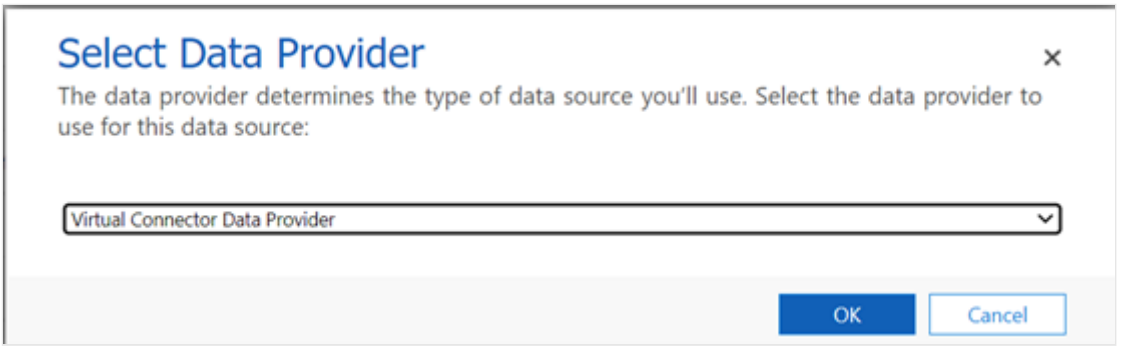


3. Select **Virtual Entity Data Sources**.

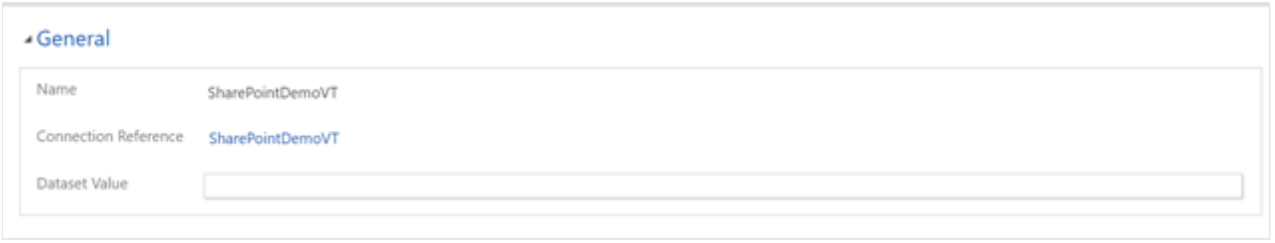


4. Select **New**. In the pop-up dialog, select the **Virtual Connector Data Provider**.





5. Name your **Data Source** and select the **Connection Reference** you created in the drop-down list.



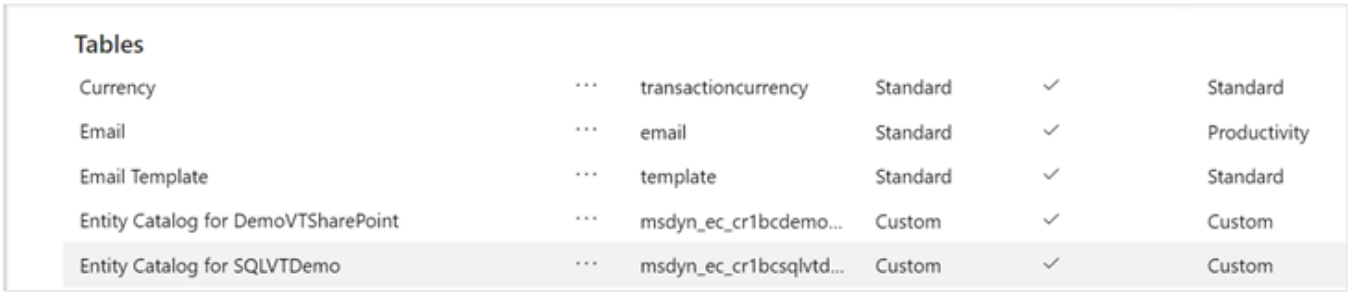
Add your Dataset Value

6. Paste in the file name including extension into the Dataset Value. Remember the file must be in the OneDrive that was used for the Connection setup. (for example: SampleData.xlsx)

7. Select **Save**.

Entity catalog

With the connection reference and the virtual table data source setup, an **Entity Catalog** is automatically generated. The **Entity Catalog** is specific to the data source and will list all the tables that are in the respective data source.

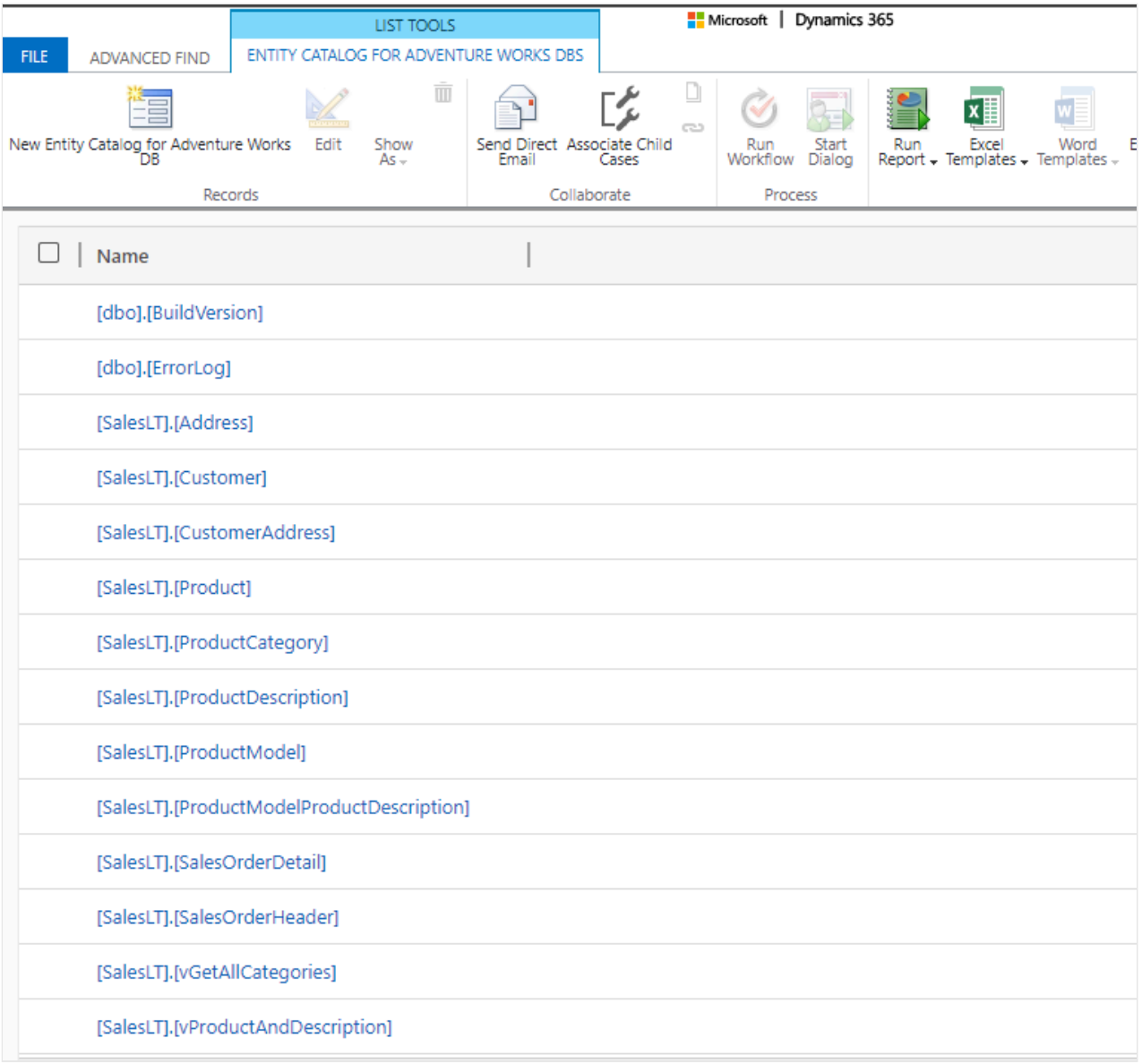


> **Note**
>
> - The creation of the entity catalog is an asynchronous process. Depending on your environment, this might take a few minutes.
> - The tables displayed in the entity catalog aren't virtual tables in themselves. You need to select from this list of tables representing the external data to create virtual table in Dataverse.

> **Tip**
>
> If the entity catalog creation takes a long time, you can check the job completion status by navigating to **Settings** -> **System Jobs** view.

View the entity catalog

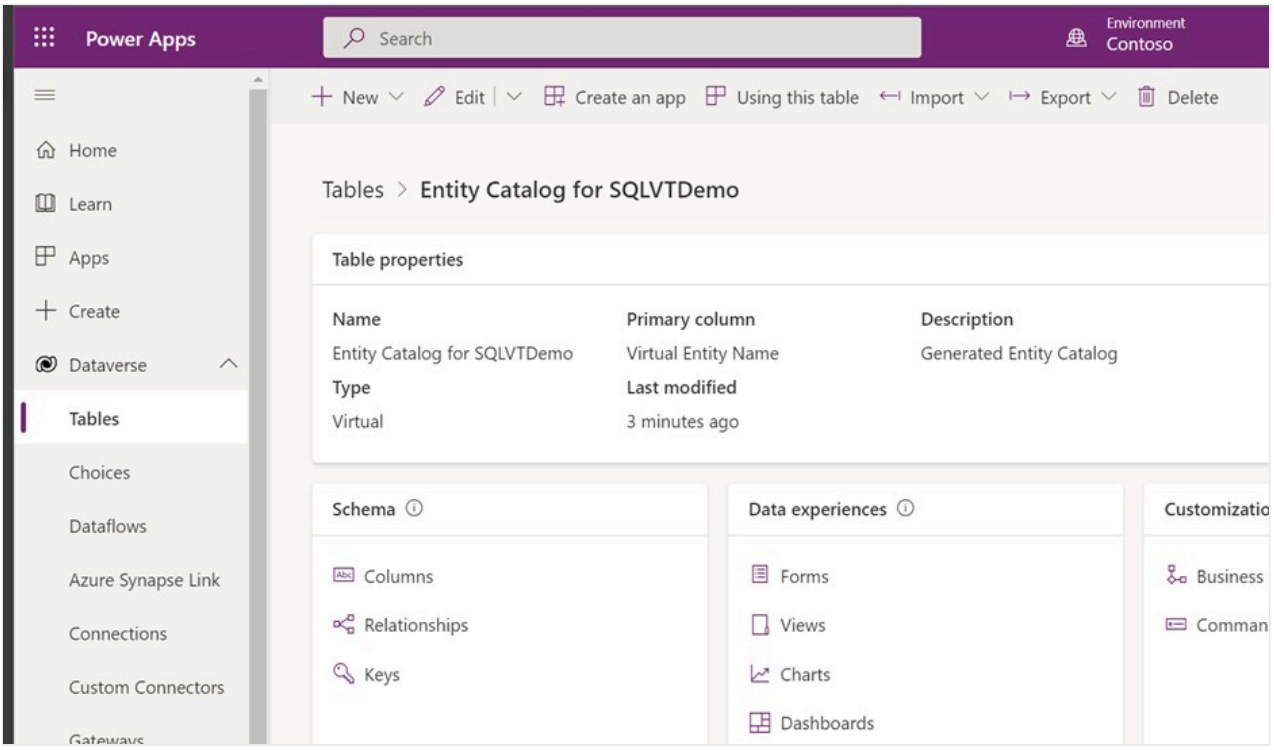
- Select **Data** > **Tables**, and then select the entity catalog that was created.
- Select **Advanced Find** and use the **Look for:** column. The catalog includes a prefix **Entity Catalog for** followed by the connection reference (example: Entity Catalog for Adventure Works). Find the entity catalog for your respective data connection and select **Results** to display all the external data source tables.



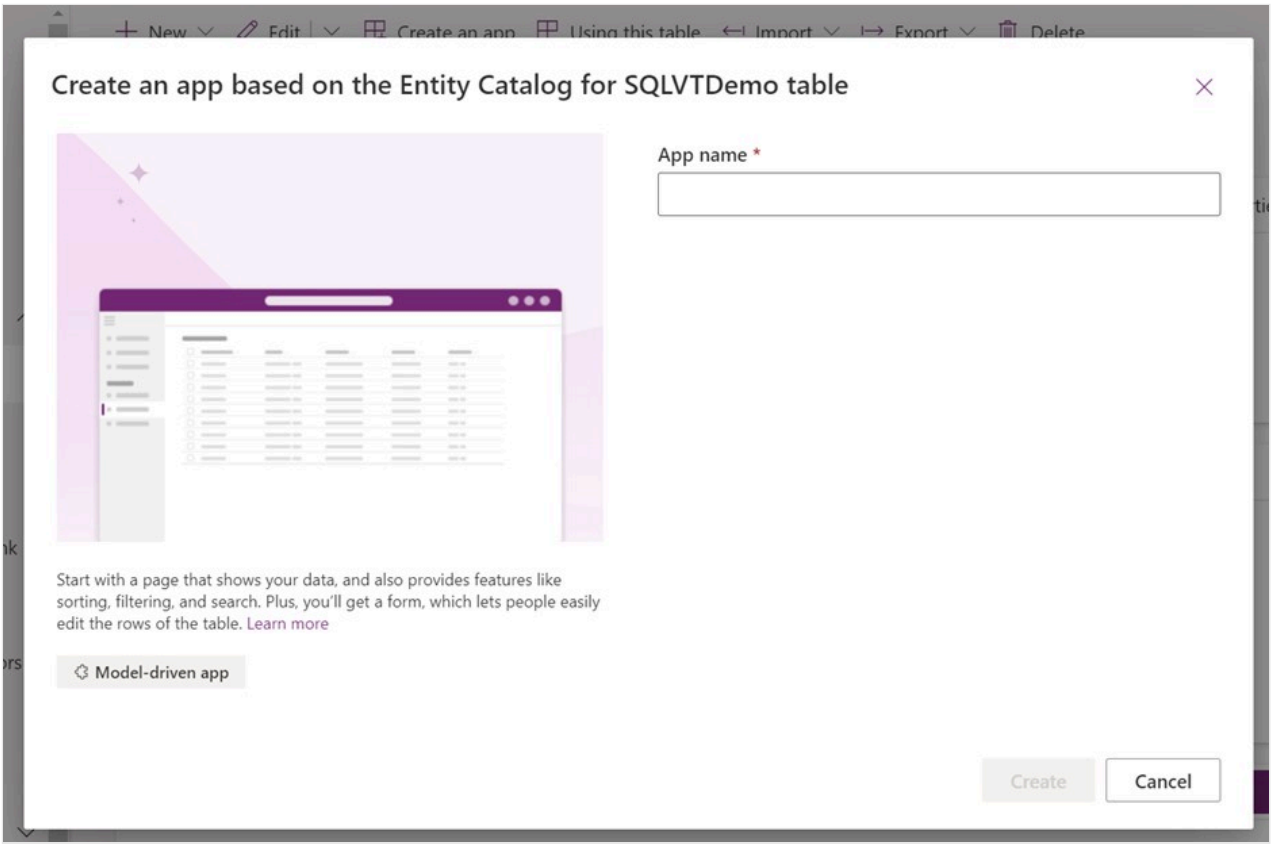
> ⓘ **Note**
>
> Bulk creation of virtual tables is not supported currently. Even though the entity catalog allows you to select multiple tables, you will have to select one table at a time to create virtual tables.

1. To create a virtual table, a model driven app must be built for the entity catalog. Select the entity catalog table.

2. Select **Create an app** in the top navigation.

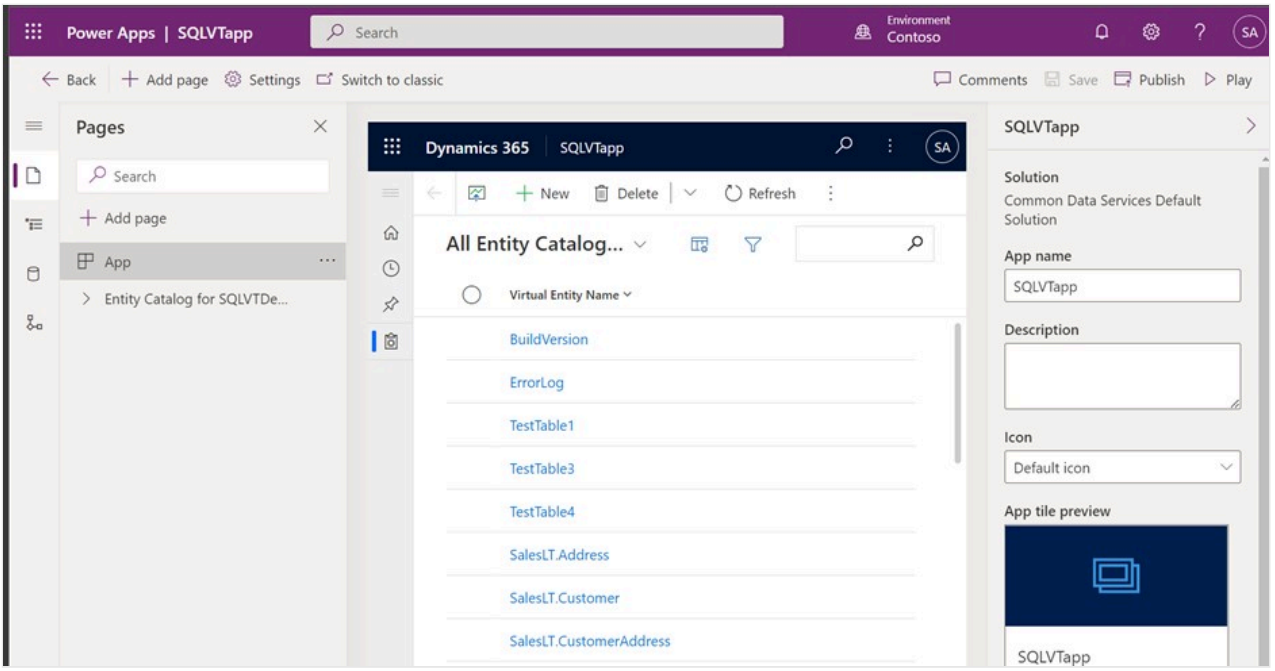


3. Name the app, and then select **Create**.



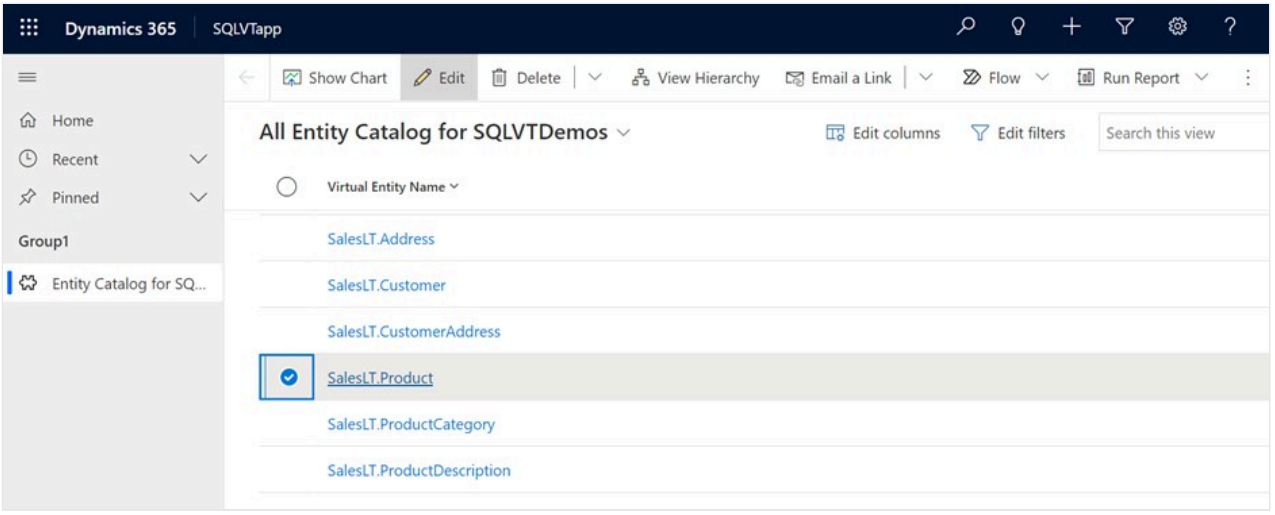
The app is automatically generated using the entity catalog table.

4. Once the app is completed, you can select **Publish** to complete the app and use it later, or you can select **Play** to create your virtual table now without publishing the app.

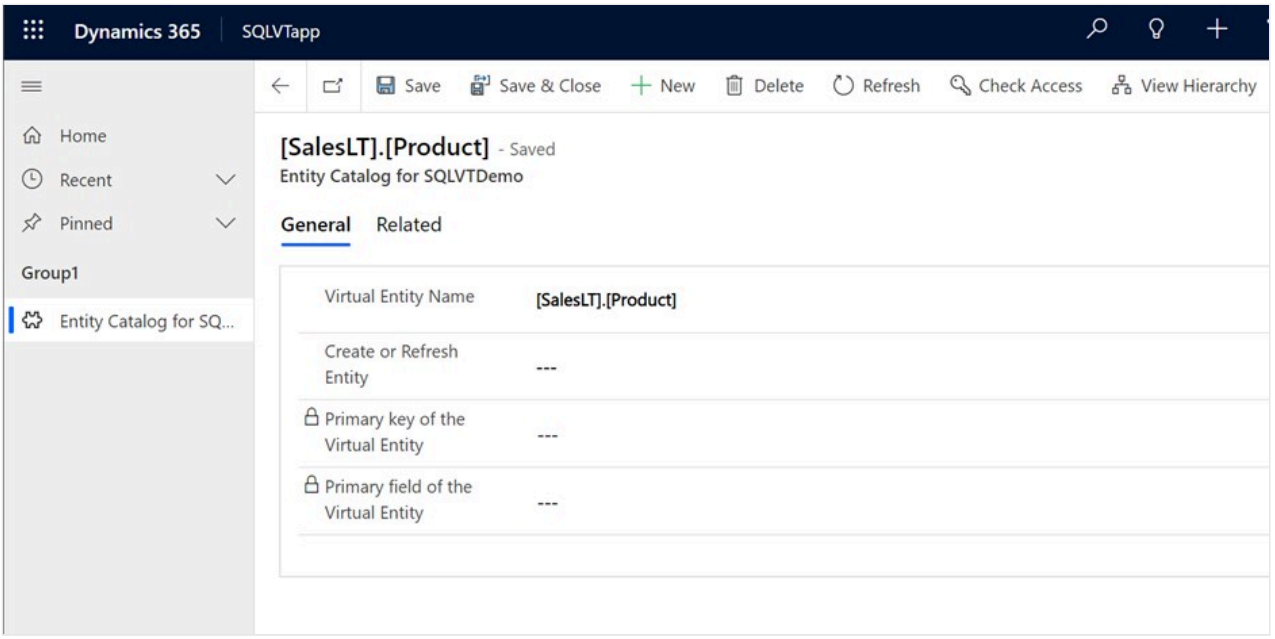


All tables from your Excel file will be provided in the app view.

5. Select the data set you wish to use from the entity catalog, and then select **Edit** in the navigation bar.

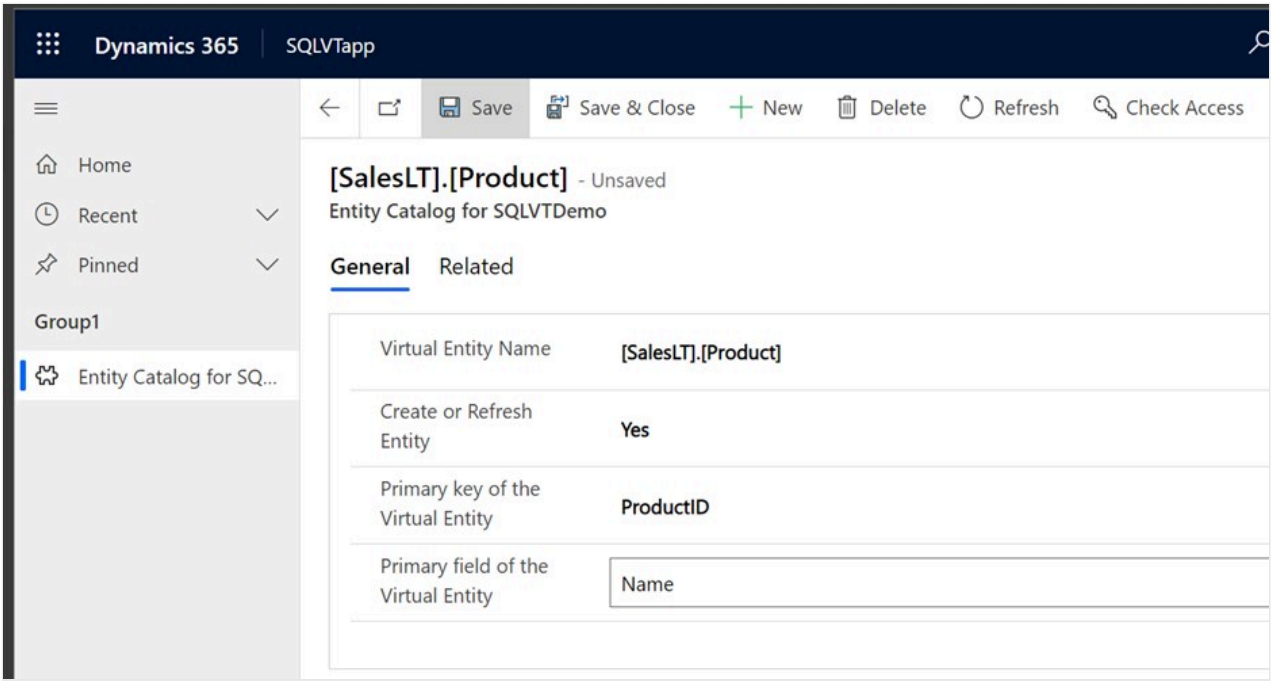


Wait for the form to fully load before editing. When loaded the form appears like this:



6. In the provided form set the **Create** or **Refresh Entity** column to Yes.

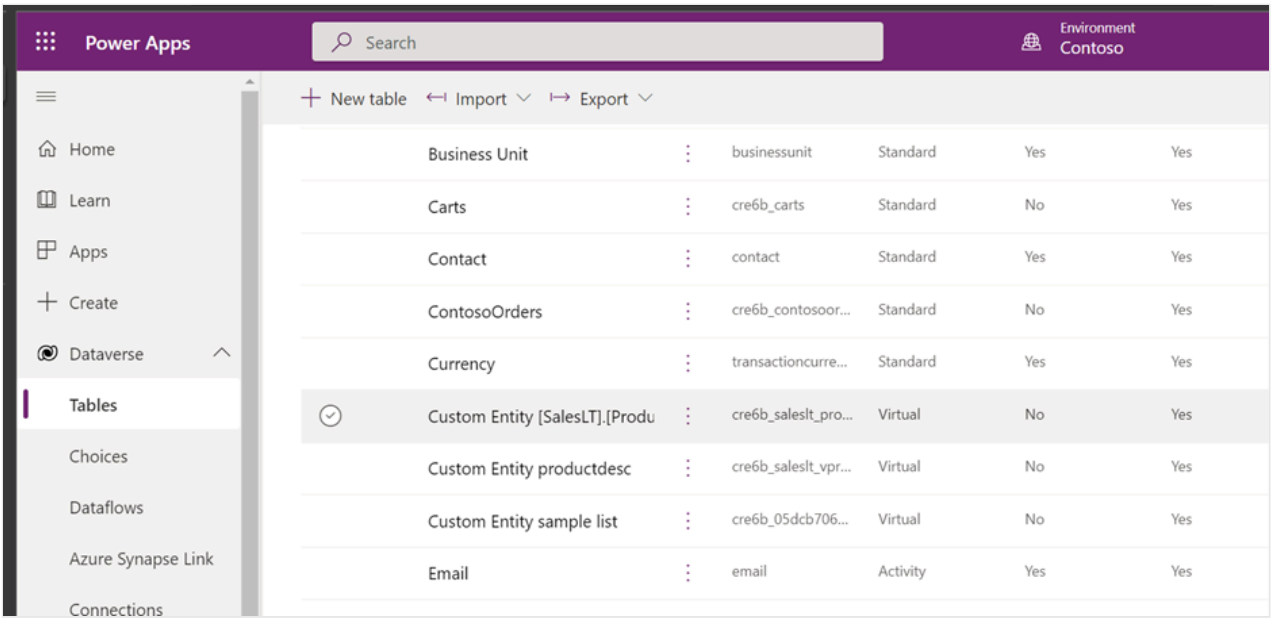
7. Select the **Primary Key** and **Primary Field** of the virtual entity by using the dropdown lists to find the columns you want to use.



8. Save the record to create the virtual table.

> ⓘ **Note**
>
> After the save completes, the form will "reset" with all fields shown as blank, this is normal.

9. Return to the Power Apps home page and select **Data**. Your virtual table is now created with a "Custom Entity" prefix. It might take a few moments for the creation to complete.



Important

- The provider automatically maps the primary key associated with the external data source when creating the virtual table. All CRUD operations can be performed on the generated virtual table.
- All columns in the external data are automatically mapped to Dataverse types that are supported by the connector. You can review the virtual table details and make changes by navigating to **Settings** -> **Customization – Entities** view.
- Virtual tables require there to be at least one string field to use as the **Primary Name** column.

Once you've created a virtual table, you can work with it much the same way as any other table. You can start defining the relationships with other tables, in the environment and use them in your Power Apps and Power Automate flows.

Related articles

Create virtual tables using the virtual connector provider

Setting up a virtual table relationship

Article • 01/04/2023

Virtual tables are enabled for relationships. You can set up 1:N, N:1, and custom multi-table (polymorphic) relationships. Relationships can be established between:

- Local tables in Dataverse and virtual tables.
- Virtual tables and other virtual tables from the same provider, for example SQL->SQL.

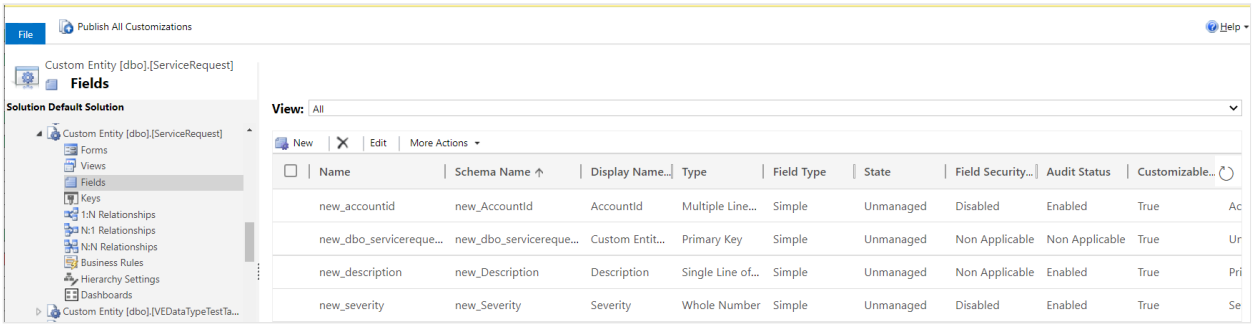
For instance, you can't set up a relationship between a virtual table created using the OData virtual table provider and a virtual table created using the virtual connector provider.

Defining relationships in virtual tables

Virtual tables created using the virtual connector provider automatically creates all the columns that are represented in the external source table. This will also include columns on which relationships are defined. However, the relationship definition won't be automatically created. You'll have to define this relationship in Dataverse manually.

The following example creates an N:1 relationship between a virtual table (**Service Request**) and a native table (**Account**). The column used to set up the relationship is **AccountId**. This column is the primary key in the account table and is a foreign key in the service request table.

A representation of the **Service Request** virtual table is shown below. You'll notice that the **AccountId** column, which is the column used for relationship in the external source, is of type **Multiple Line of Text**. You need to have this column represented as a **Lookup** type to create a relationship.



1. Go to **Advanced settings** > **Settings** > **Customization** and choose **Customize the System**.

2. In the left navigation pane, expand the **Entities** view and browse to the **Service Request** virtual table definition.

3. Select the **Fields** view, select the **AccountId** column, and then select **Delete**.

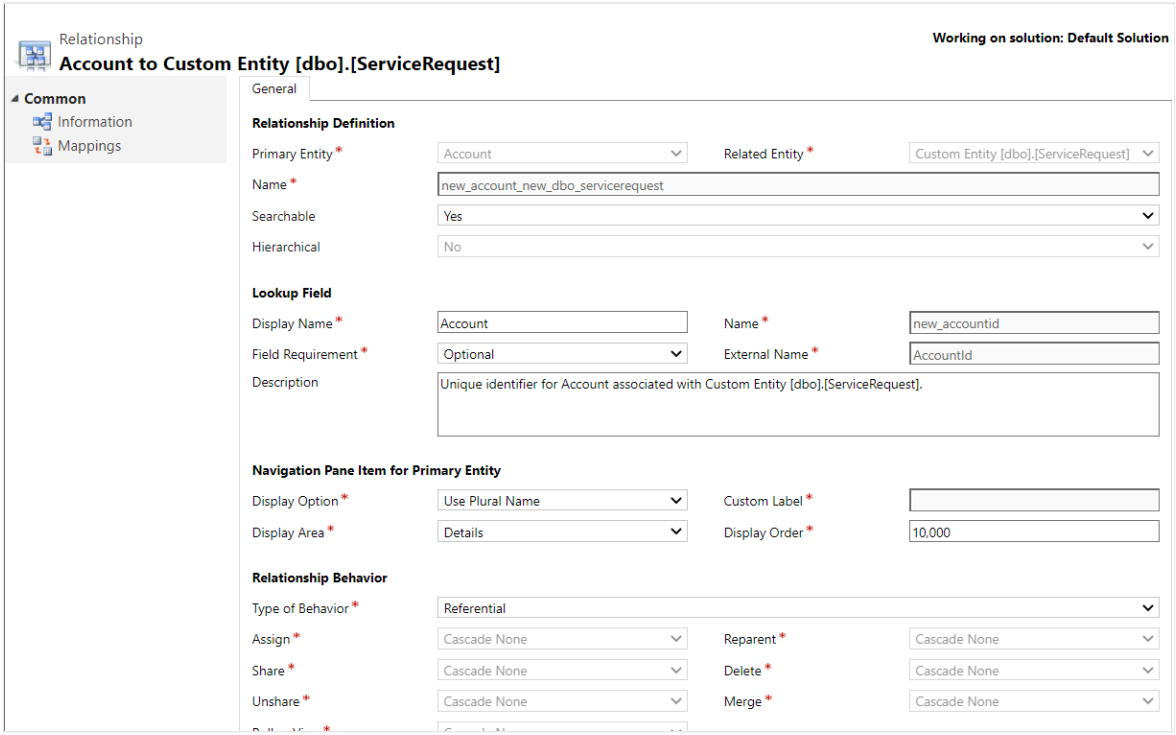
4. Choose **Delete** to confirm the deletion of this column.

5. To create the relationship, select the **N:1 Relationship** within the **Service Request** table.

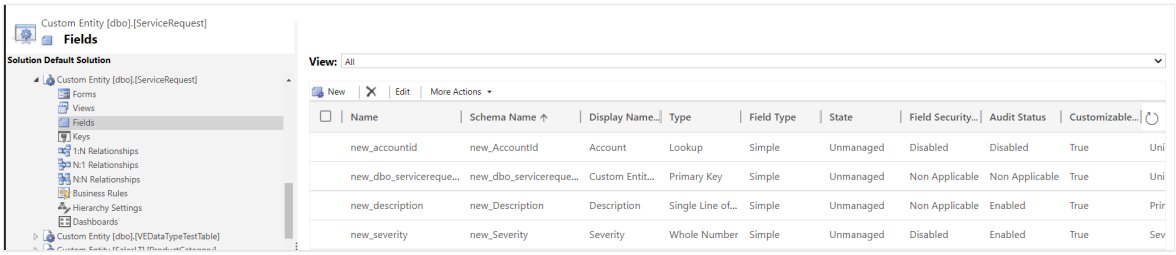
6. Select **New Many-to-1 Relationship**.

7. Enter the following details to create the relationship between the **Service Request** virtual table and the **Account** table.
   a. In the **Relationship Definition** section – set the **Primary Entity** column value to **Account**.
   b. Optionally, if you want to edit the name of the relationship, you can do so in the **Name** column.
   c. In the **Lookup Field** section, set the **Display Name** to **Account.**
   d. The **Name** column automatically populates with the lookup column name.
   e. Set the **External Name** value to **AccountId** (matching the column name in your source table).

8. **Save**, and then close the relationship.



9. Refer to the columns for the **Service Request** virtual table, and you'll notice that the **AccountId** column is a **Lookup** type. This column can now be added to forms and views to see all associated accounts for each of the service request record.



10. With the relationship established you can now create a new service request and pick accounts to associate them to.



> [!NOTE]
> You will have to edit the forms and views for this table to include the lookup column and other required columns prior to operation on the virtual table.

Tip

- The **Primary Key** column should be included in the create form if you didn't set up the column to increment during the design of the underlying source table automatically. You'll have to enter a valid value in the primary key column for an insert operation to succeed.

Next steps

Create virtual tables using the virtual connector provider (preview)

Known limitations and troubleshooting with virtual tables

This article describes the known limitations and troubleshooting tips when working with virtual tables in Microsoft Dataverse.

The following list describes known limitations for virtual tables created by using the virtual connector provider.

General limitations

- The table or list you use must include at least one string field to use as the primary field, and one GUID field. Without these string fields, you can't create the virtual table and an error occurs during the table details retrieval stage.
  - SharePoint uses the hidden numeric ID field present in all lists.
  - SQL can use a GUID or integer field.
  - Excel must have a GUID field.
- Dataverse can only create columns that include data types compatible with Dataverse. This limitation includes the following data types:
  - String
  - Multiline text (memo)
  - Whole Number/Integer
  - Decimal
  - Float
  - Date/time
  - Yes/No (boolean)
  - Choices (multi-value select)
  - Hyperlink/Url
- Dataverse doesn't support the following data types for virtual tables:
  - File and attachments
  - Image
- Maximum length of characters allowed for a text column in a virtual table is 4,000 characters. If the source table has a maximum character limit greater than this value, any create or update operation exceeding the max character limit results in a validation error, and the operation fails.

- Virtual table queries are limited to returning 1,000 records. If you have a 1:N or N custom multitable (polymorphic) relationship with a virtual table, any query that exceeds this limit fails and provides an error. Use filtering in your query to reduce the record set as a workaround to this limitation.
- Audit functionality isn't available for virtual tables because Dataverse can only perform and store audit data for locally stored data.
- Rollups and calculated fields can't be calculated for virtual tables. This limitation exists because rollups are a server side calculation in Dataverse, which requires the data to be stored locally.
- Formula columns can't use virtual tables.
- The **Microsoft Entra ID** virtual table provided by Microsoft only allows read access.
- Dataverse virtual tables can display values in fields that exceed the normal maximum values of Dataverse. This behavior occurs because the values being presented aren't stored locally. For example, the Dataverse integer maximum value is 100,000,000,000, but it could retrieve and display 9,000,000,000,000 from SharePoint. However, if the user attempts to edit the number to a size larger than the max accepted size in Dataverse, an error is provided indicating the record can't be saved because it exceeds the maximum size.
- Import and export functionality of table data isn't supported for virtual tables.
- Queries against virtual tables that use negative filter operators, such as Does Not Equal or Does Not Contain, might result in incorrect paging behavior beyond the first page. There's currently no supported workaround. Avoid using negative filters.

Limitations for each data source

The following limitations apply to each data source.

SQL Server

- For functionality, SQL virtual tables use a GUID or an integer field as the primary key.
- SQL Server tables without primary keys: You can select any nonstring field as the primary key. You can create the virtual table successfully. `RetrieveMultiple` works, but the other operations fail with the following error message (coming from SQL connector): "APIM request wasn't successful: BadRequest: No primary key exists in table." For functionality, you must use a GUID or integer field as the primary key.
- SQL Server tables that use a string primary key: The SQL string primary key is the only option available for the virtual table primary key. SQL Server string primary keys are supported only if the values can be parsed as GUID. If the values can't be parsed

as GUID, the virtual table creation succeeds, but fails at runtime with the following errors:
    - Power Apps (make.powerapps.com): "We weren't able to open your table. Try reloading or reopening."
    - Network trace: "String primary keys are supported only if they can be parsed as GUID."
- SQL Server tables without nonprimary key string fields for use as the primary name: If the SQL table doesn't have a string field available to use as the primary name, the configuration step displays the following error: "The table doesn't have a primary field."
- You can use SQL views to create a virtual table but they only provide read operations.
- For SQL Server connector limitations, go to SQL Server connector reference.
- SQL data type, *bigint columns*, in the source table are mapped as a decimal data type in Dataverse virtual tables. When platform support is available for bigint mapping to a whole number, you need to delete previously created columns in the virtual table and create new columns.
- You can't include the following column types in a virtual table at this time:
    - Time
    - Datetime2
    - Image
    - Geometry
    - Geography
    - RowVersion
    - Choice
- The following column types are included in a virtual table but are only shown as text fields:
    - HierarchyID
    - XML
    - Sqlvariant

Troubleshooting

- There's only one (1) record in your virtual table even though you have more in your source table.
  Solution: Check your source table and make sure it has a primary key defined.

- You get one of the following errors when Power Apps (make.powerapps.com) is retrieving a table list or when you select **Finish** to create a table:
  - "Resource not found for the segment `msdyn_get_required_fields`"
  - "Error calling... please verify that connection... exists in environment"
  - "Sequence matches no element for `msdyn_get_required_fields`"

  Solution: In some cases you might not have the most up to date solution for the virtual connector provider. To determine whether your virtual connector provider solution needs an update:

  1. Select **Solutions** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
  2. Select the **History** tab.
  3. **Search** for `ConnectorProvider`.
  4. View the information to see whether the solution needs to be updated.
  5. If the history indicates an update is needed, go to the Microsoft commercial marketplace search for **Virtual Connector Provider**, and then select **Get it now** to import the solution into your environment.
  6. Follow the steps to create the virtual table again.

- A message is displayed "Connection 'xyz' not found in current environment." when retrieving the list of connections.
  Solution: This occurs when there are a large number of connections in the user's Dataverse environment. This is fixed with version 1,029 of the Connector Provider solution. To determine whether your virtual connector provider solution needs an update:

  1. Select **Solutions** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
  2. Select the **History** tab.
  3. **Search** for `ConnectorProvider`.
  4. View the information to see whether the solution needs to be updated.
  5. If the history indicates an update is needed, go to the Microsoft commercial marketplace search for **Virtual Connector Provider**, and then select **Get it now** to import the solution into your environment.
  6. Follow the steps to create the virtual table again.

- You get notified that a timeout occurred during the virtual table creation.
  Solution: This can occur when other existing jobs cause the virtual table creation to be delayed. Wait for a few minutes and try again.

- You get notified that "An unexpected error occurred"
  Solution: This occurs when the virtual table data source was created with invalid values. To resolve this, you need to locate the virtual table data source that is causing the error, delete it, and then re-create the virtual table.

  1. Select **Settings** (gear icon) > **Advanced settings** from Power Apps.
  2. In the top menu, select **Settings**.
  3. Go to **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
  4. Go to the solution that includes your virtual table (if you weren't using a solution then go to Common Data Services Default Solution).
  5. In the left hand panel, select **Virtual table data sources**.
  6. Double-click each data source (they all start with `VCP_DS_..."`), when you locate the one with the error, delete that data source.
  7. Re-create your virtual table.

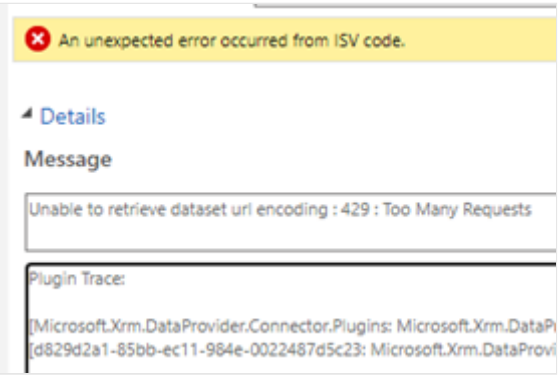
- A message is displayed "This table already exists, you're recreating the table. Primary field and Schema name can't be changed."
  Solution: This table has previously been created. Continuing with the creation re-creates the table, this results in any table changes made at the data source to be updated in the virtual table (this includes addition or removal of fields). The custom name and primary field values won't be editable.

- Error message: "primary_key_name can't be empty"
  Solution: You have chosen a table or list that doesn't include a GUID value for the primary key. You need to add an additional GUID column in your source table in order to create a virtual table.

- You created an Excel virtual table but don't see it in the **Tables** area in Power Apps.
  Solution: Since the virtual table creation is asynchronous, you can check the status of the process in **System Jobs**. Look for system jobs with a Name starting `Microsoft.Wrm.DataProvider.Connector.Plugins.ConnectorGenerateVEPlugin` and a **Regarding** column's value equal to the name of the new virtual table. If status is still **In Progress**, just wait for the job to complete. If there's an error, you can get details by selecting the system, job name hyperlink. In this example, table creation is still pending:



Here, table creation failed due to 429 "Too Many Requests" error:



- Table creation's system job succeeded but you receive runtime errors related to invalid or missing columns.
  Solution: If a failure occurs while you create a table field, the table creation process doesn't fail and try to continue with the remaining fields. This is because the system doesn't want to block the virtual table creation when some column types are unsupported. To get details on the error, enable logging in **Administration** > **System Settings** > **Customizations** > **Enable logging to plug-in trace log**, then delete the virtual table and try to create it again.

- If you deleted the connection connected to virtual table and recreated it, the Virtual Connector Provider app loses permission to access the new connection, preventing data retrieval.
  Solution: Manually share the recreated connection with the 'Virtual Connector Provider' app using the connection's share feature to restore access.

- When a custom data provider for a virtual table is updated to support new operations (e.g., create, update, delete), the platform does not automatically add corresponding permissions to the existing virtual table entity.
  Solution: To enable new permissions, the user must recreate the virtual table entity after updating the data provider.

Next steps

Create virtual tables using the virtual connector provider (preview)

Setting up a virtual table relationship

Last updated on 04/11/2026

Create and edit elastic tables

Article • 03/31/2025

An elastic table is a table managed by Microsoft Dataverse. Elastic tables come with the same familiar user experience and API that are offered with standard tables. They share many aspects and options with standard tables, but come with their own unique features and capabilities that are powered by Azure Cosmos DB.

Elastic tables are included with your Dataverse log capacity use.

Watch this video that to learn about elastic tables.
https://learn-video.azurefd.net/vod/player?id=6f859c28-12b1-4042-9502-b8a441020768&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fcreate-edit-elastic-tables

When to consider Dataverse elastic tables?

Elastic tables are designed to handle large volumes of data in real-time. With elastic tables, you can import, store, and analyze large volumes of data without scalability, latency, or performance issues.

Elastic tables have unique capabilities for flexible schema, horizontal scaling, and automatic removal of data after a time-period.

Elastic tables automatically scale to ingest tens of millions of rows every hour. Background processes can collate the IoT signals, predict maintenance requirements, and proactively schedule technicians.

Consider a scenario where Contoso is a retailer with millions of existing customers. Contoso has a large database of customers and are looking to increase sales while retaining customers. Based on prior customer history, they're looking to have 24-hour flash sale events with different coupons targeting their customers and products. They have estimated that the number of coupons required will be 100 million plus per flash sale campaign. Marketing plans to run multiple 24-hour campaigns targeting different customer segments.

The requirement for Contoso's marketing application is that it must be able to ingest up to 100 million or more coupon details within a few hours, read millions of coupons per hour, and send coupons to customers.

Elastic tables will automatically scale for this high throughput scenario.

For example, in the above scenario, an elastic table named *Coupon* with millions of records can be associated with Dataverse standard tables like *Contact* (customer info) and *Offer* (a custom standard table). Since the elastic tables are isolated from the standard tables, performance for the overall marketing application won't be negatively impacted. In addition, time-to-live capability with elastic table (*Coupon* in this scenario) allows removal of data automatically after fixed periods and ensure optimization of log capacity.

Use elastic tables when:

- Your data might be unstructured or semi-structured, or if your data model might constantly change.
- You need automatic horizontal scaling.
- You need to handle a high volume of read and write requests.

Use standard tables when:

- Your application requires strong consistency.
- Your application requires relational modeling and needs transactional capability across tables and during plugin execution stages.
- Your application requires complex joins.

The choice of table should be based on the specific needs of your application. A combination of both types of tables might be appropriate.

Horizontal scaling and performance

As your business data grows, elastic tables provide unlimited auto scalability based on your application workload, both for size and throughput, such as the number of records created, updated, or deleted in a given timeframe.

If your business scenario requires very large volume of data writes, application makers can make use of Dataverse multiple request APIs, such as `CreateMultiple`, `UpdateMultiple`, and `DeleteMultiple`, to achieve more throughput within Dataverse throttling limits. More information: Developer guide: Bulk Operation messages and Optimize performance for bulk operations

Automatic removal of data

Time to live (TTL) policies ensure that you're always working with the most up-to-date and accurate information, while optimizing resources and reducing risk. The TTL live

value is set in seconds on a record, and it's interpreted as a delta from the time that a record was last modified.

Flexible schema with JSON columns

Elastic tables enable you to store and query data with varying structures, without the need for predefined schemas or migrations. There's no need to write custom code to map the imported data into a fixed schema. More information: Developer guide: Query JSON columns in elastic tables Elastic tables enable you to store and query data with varying structures, without the need for predefined schemas or migrations. There's no need to write custom code to map the imported data into a fixed schema. More information: Developer guide: Query JSON columns in elastic tables

Considerations when you use elastic tables

Although elastic tables are great for handling large volume of requests at scale, the advantages come with a few trades offs, which should be kept in mind:

- Elastic tables don't support multi-record transactions. This means that multiple write operations happening as part of a single request execution aren't transactional with each other. For example, if you have a synchronous plug-in step registered on the `PostOperation` stage for `Create message` on an elastic table, any error in your plug-in won't roll back the created record in Dataverse. Validations in preplug-ins will still work as expected since they run before the main stage.
- Elastic tables support strong consistency only within a logical session. Outside session context, you might not see changes to a row immediately. More information: Developer guide: Consistency level
- Elastic tables don't support filters on related tables when creating views, advanced find, or any query in general using API. If you frequently need to filter on related table columns, we recommend that you denormalize columns from related tables, which need to filter into the main table itself. Consider a retailer with two elastic tables: customer and address. One customer has many addresses. You want to return query results for all customers from the customer table whose city value in the address table is New York. In this example, when querying customer table, you want to apply a filter on the city column of the related address table. This isn't supported for elastic tables. One way to make this work is to denormalize the city column into the Customer table so that all customers city values are present in the customer table itself.

Elastic tables feature support

- Create, retrieve, update, delete (CRUD) operations including API multiple operations (for high throughput), bulk deletion, and requests from plug-ins.
- Relationships:
  - One-to-many
  - Many-to-one when the N table is a standard table
- Record ownership, change tracking, auditing, mobile offline, and Dataverse search.
- File column with file type attribute

Security features support

Elastic tables adhere to the Dataverse security model.

When creating an elastic table, you can set:

- Either user or organization owned
- Field level security

Features currently not supported with elastic tables

Table features currently not supported with elastic tables:

- Business rules
- Charts
- Business process flows
- One Dataverse connector for Power BI
- Many-to-many (N:N) relationships to standard tables
- Alternate key
- Duplicate detection
- Calculated and rollup columns
- Currency columns
- Column comparison in queries
- Table sharing
- Composite indexes
- Cascade operations: Delete, Reparent, Assign, Share, Unshare
- Ordering on lookup columns
- Aggregate queries:
  - Distinct value of `attribute1` while orderby on `attribute2` value
  - Pagination when having multiple distincts
  - Distinct with multiple order by
  - Order by and group by together
  - Group by on link entity (left outer join)

    - Distinct on user owned tables
- Table connections
- Access teams
- Queues
- Attachment
- Import and export functionality of table data.

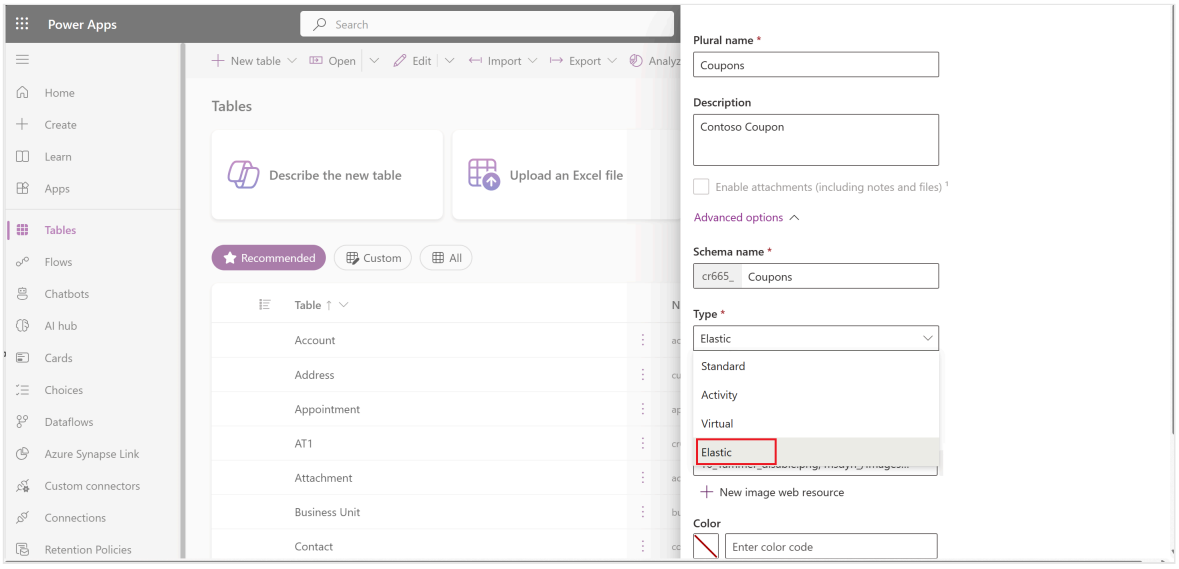
Column data types currently not available with elastic tables:

- Currency
- Formula
- Whole number format other than None (Duration, Language code, and Time zone)
- Lookup based on the Customer option

Create an elastic table

You create an elastic table just like any other new table in Dataverse.

1. Sign into Power Apps, and select **Tables** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **New table** > **Set advanced properties** on the command bar.
3. On the right properties pane, enter a **Display name** and a **Plural name**.
4. Expand **Advanced options**, and then select **Elastic** as the table **Type**.



5. Select the properties you want, and then select **Save**.

The time to live column is automatically created for an elastic table. You can add the time-period value in seconds, as required. The data is auto removed after the specified time-period.

More information about Tables: Advanced options

Known issues

- When time to live (TTL) is used on a row, the row gets deleted from the elastic table when TTL expires. If it's synchronized to a data lake using Azure Synapse Link for Dataverse before TTL expiry, it won't be deleted from the data lake.
- Point in time restore doesn't restore "updated" records as updates aren't backed up. Only created and deleted records are restored.
- If a specific column in an elastic table is deleted, the columns value isn't removed from the table rows if it has data. Before deleting a specific column, delete the data from all rows for the column.

High volume elastic tables and managing Dataverse API throttling limits

Use bulk operation messages. This allows you to achieve 10 times the throughput with the same Dataverse API throttling limits. Developers can reference more links provided in the below section.

For developers

Elastic tables have different behaviors and capabilities than standard tables when developers use them with Dataverse APIs. The following articles for developers describe these differences:

- Elastic tables
- Create elastic tables using code
- Use elastic tables using code
- Query JSON columns in elastic tables
- Bulk Operation messages (preview)
- Elastic table sample code
- Send parallel requests
- Service protection API limits

See also

Create and edit tables using Power Apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a business rule for a table

You can create business rules and recommendations to apply logic and validations without writing code or creating plug-ins. Business rules provide a simple interface to implement and maintain fast-changing and commonly used rules.

> **Important**
>
> Business rules defined for a table apply to both *canvas apps* and *model-driven apps* if the table is used in the app. Not all business rule actions are available on canvas apps at this time. More information: Differences between canvas and model-driven apps
>
> In a model-driven app, not all business rule actions are available for editable grids. For table based view pages, recommendations can't be created. Editable subgrids don't support business rules. Business rules won't work with other types of dataset controls.
>
> To define a business rule that applies to a form in a model-driven app, see Create business rules to apply logic in a model-driven app form.

By combining conditions and actions, you can do any of the following with business rules:

- Set column values
- Clear column values
- Set column requirement levels
- Show or hide columns
- Enable or disable columns
- Validate data and show error messages
- Create business recommendations based on business intelligence.

Differences between canvas and model-driven apps

Forms in model-driven apps can use all actions available on business rules, however not all business rule actions are available for canvas apps at this time. The following actions are *not* available on canvas apps:

- Show or hide columns
- Enable or disable columns

- Create business recommendations based on business intelligence

Column type support with business rules

Business rules work with most column types including text, number, choice, date, lookup, owner, and image. However, business rules don't work with the following column types:

- Choices (multi-select)
- File
- Language

Create a business rule

1. Sign in to Power Apps, on the left navigation pane, and select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Open the table you want to create the business rule for (for example, open the **Account** table), and then select **Business rules**.

3. Select **Add business rule**.

   The business rule designer window opens with a single condition already created for you. Every rule starts with a condition. The business rule takes one or more actions based on that condition.

   > **Tip**
   >
   > If you want to modify an existing business rule, you must deactivate it before you can modify it.

4. Add a description, if you want, in the description box in the upper-left corner of the window.

5. Set the scope, according to the following:

Expand table

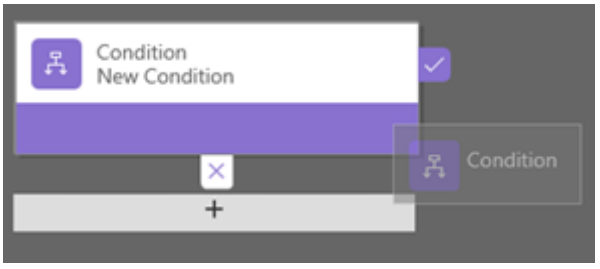
If you select this item...	The scope is set to...
Entity	Model-driven app forms and server

If you select this item...	The scope is set to...
All Forms	Model-driven app forms
Specific form (**Account** form, for example)	Just that model-driven app form

> **Tip**
>
> If you're building a Canvas app, you must use table as the scope.

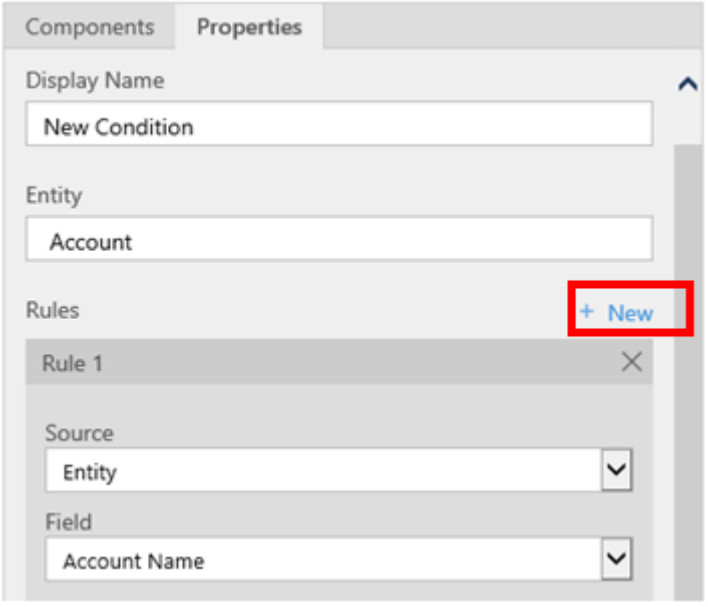
6. **Add conditions.** To add more conditions to your business rule:

   a. Drag the **Condition** component from the **Components** tab to a plus sign in the designer.



   b. To set properties for the condition, select the **Condition** component in the designer window, and then set the properties in the **Properties** tab on the right side of the screen. As you set properties, the Microsoft Dataverse creates an expression at the bottom of the **Properties** tab.

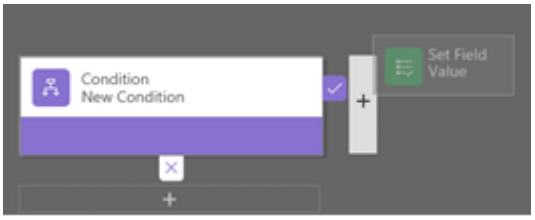
   c. To add an additional clause (an AND or OR) to the condition, select **New** in the **Properties** tab to create a new rule, and then set the properties for that rule. In the **Rule Logic** column, you can specify whether to add the new rule as an AND or an OR.



d. When you're done setting properties for the condition, select **Apply**.

7. **Add actions.** To add an action:

   a. Drag one of the action components from the **Components** tab to a plus sign next to **Condition** component. Drag the action to a plus sign next to a check mark if you want the business rule to take that action when the condition is met, or to a plus sign next to an x if you want the business rule to take that action if the condition isn't met.

   b. To set properties for the action, select the **Action** component in the designer window, and then set the properties in the **Properties** tab.

   c. When you're done setting properties, select **Apply**.

8. **Add a business recommendation (model-driven apps only)**. To add a business recommendation:

   a. Drag the **Recommendation** component from the **Components** tab to a plus sign next to a **Condition** component. Drag the **Recommendation** component to a plus sign next to a check mark if you want the business rule to take that action when the condition is met, or to a plus sign next to an x if you want the business rule to take that action if the condition isn't met.

   b. To set properties for the recommendation, select the **Recommendation** component in the designer window, and then set the properties in the **Properties** tab.

   c. To add more actions to the recommendation, drag them from the **Components** tab, and then set properties for each action in the **Properties** tab.

   > ⓘ **Note**
   >
   > When you create a recommendation, the Dataverse adds a single action by default. To see all the actions in a recommendation, select **Details** on the **Recommendation** component.

   d. When you're done setting properties, select **Apply**.

9. To validate the business rule, select **Validate** on the action bar.
10. To save the business rule, select **Save** on the action bar.
11. To activate the business rule, select it in the Solution Explorer window, and then select **Activate**. You can't activate the business rule from the designer window.

> **Tip**
>
> Here are a few tips to keep in mind as you work on business rules in the designer window:
>
> - To take a snapshot of everything in the **Business Rule** window, select **Snapshot** on the action bar. This is useful, for example, if you want to share and get comments on the business rule from a team member.
> - Use the mini-map to navigate quickly to different parts of the process. This is useful when you have a complicated process that scrolls off the screen.
> - As you add conditions, Actions, and business recommendations to your business rule, Dataverse builds the code for the business rule at the bottom of the designer window. This code is read only.

Example: Create a business rule for tasks older than 30 days

This business rule example creates a condition that triggers a message in the task description field when a task is more than 30 days old.

1. Sign in to Power Apps, on the left navigation pane, select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the **Task** table, and then select the **Business rules** area.
3. Select **Add business rule**.
4. Select **New Condition** on the business process flow canvas, and enter or select the following properties:

    - **Display name**: *Task is more than 30 days old*
    - **Entity**: **Task**
    - **Rule 1**
        - **Source**: **Entity**
        - **Field**: **Created On**

    - **Operator**: +
    - **Type**: **Value**
    - **Days**: *30*
  - **Condition Expression** (automatically created): **(Created On Is greater than [Created On + 30])**
5. Select **Apply**.
6. Select **Add** > **Add Show Error Message**.
7. In the **Show Error Message Properties** tab, enter the following properties:

   - **Display Name**: *Task is more than 30 days old*
   - **Entity**: **Task**
   - **Error Message**:
     - **Field**: **Description**
     - **Message**: *This task is more than 30 days old!*

8. Select **Apply**.
9. Select **Save**.

Localize error messages used in business rules

If you have more than one language provisioned for your organization, you'll want to localize any error messages that you have set. Each time you set a message, a label is generated by the system. If you export the translations in your organization, you can add localized versions of your messages and then import those labels back into the Dataverse, so that people using languages other than your base language can view the translated messages.

Common issues

This section describes common issues that might occur when you use business rules.

Composite attributes not supported with Unified Interface apps

Actions or conditions that use Composite attributes aren't supported in apps based on the Unified Interface. Alternatively, you can use actions or conditions on the attributes that comprise the composite attributes. For example, instead of using the **Full Name** (fullname) attribute, you can use the **First Name** (firstname) and **Last Name** (lastname) attributes.

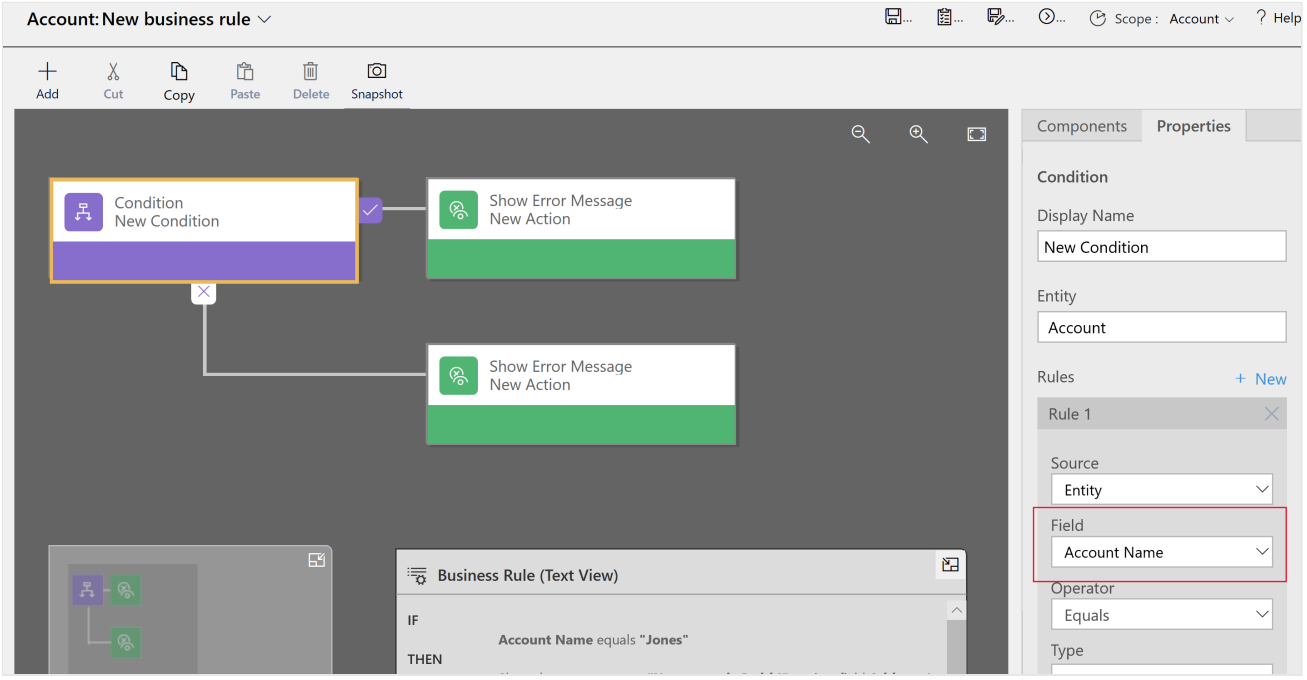
A large number of business rules on a single table impacts performance

New or existing business rules on a table can cause performance degradation when the rules are activated. Power Platform today supports up to 150 business rules for a single table. Beyond 150 business rules, performance degradation can be experienced. This limit includes both client side (JavaScript) and server side (XAML generated as synchronous plugins) business rules. To avoid performance issues with Dataverse, we recommend that you don't create more than 150 business rules for a single table.

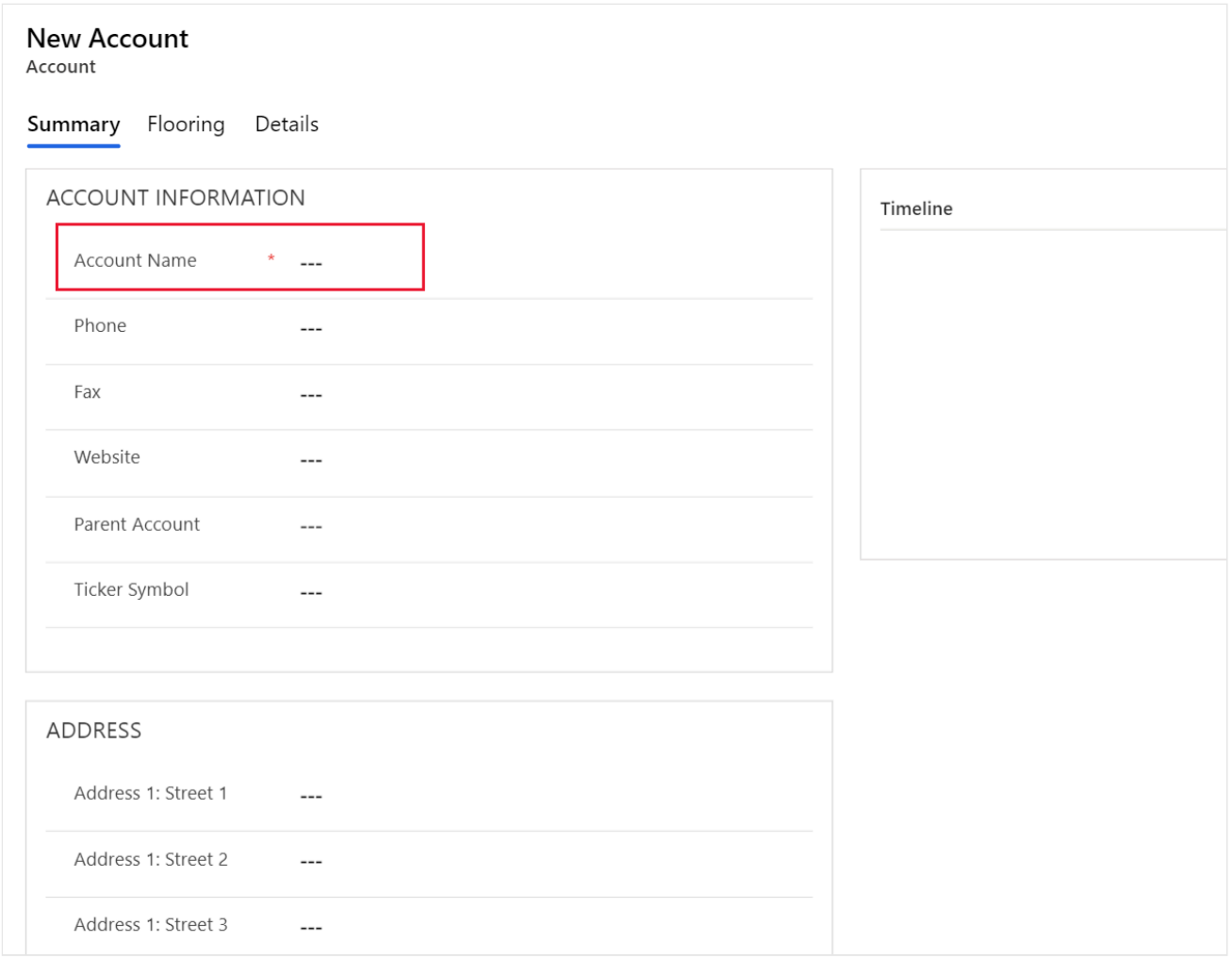
Is your business rule not firing for a form?

A business rule might not execute because the field referenced in the business rule isn't included with the form.

1. Open solution explorer. Expand the entity that you want and then select **Forms**.
2. Open the form that you want and then on the form designer ribbon select **Business Rules**.
3. In the form designer, open the business rule.
4. In the business rule designer, select each condition and action to verify all the fields referenced in each condition and action.



5. Verify that each field referenced in the business rule is also included on the form. If not, add the missing field to the form.



UTC time zone for DateOnly columns

When you configure business rules with `DateOnly` columns, the dates are in the UTC time zone by default, irrespective of the user's set time zone. This can lead to unexpected results if you're expecting `DateOnly` values to align with local time settings.

Suggested handling: Time Zone Independent setting - To make the `DateOnly` columns independent of time zones and display dates in parity with what was configured in the business rule, UTC for example, you can enable the Time Zone Independent setting. For more information about configuring date handling settings, go to Specify the behavior of a date and time column.

Frequently asked questions (FAQ)

Can business rules unlock fields on a read-only form?

- Yes, a business rule can unlock fields and edit actions on a read-only form.

How do I troubleshoot a business rule that isn't working?

- See Is your business rule not firing for a form? in this article.

Do business rules react to changes made by an onLoad script?

- No, they'll execute before an onload script is executed.

When I update a business rule, is it executed against all existing records?

- No. Business rules are run on clients. For example, they run when a form is opened by a user and when a field value changes on that open form. They aren't executed inside Dataverse.

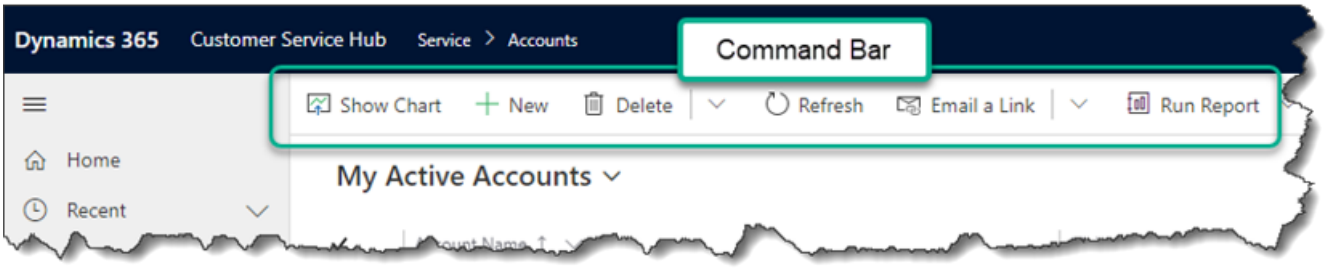
See also

Apply business logic in Microsoft Dataverse

Last updated on 04/21/2026

Modern commanding overview

Commands drive core application behavior for model-driven apps. They're the buttons users interact with when playing apps and the resulting actions performed when a button is selected. Each command is positioned in relation to other commands and bound to a command bar location within the app.

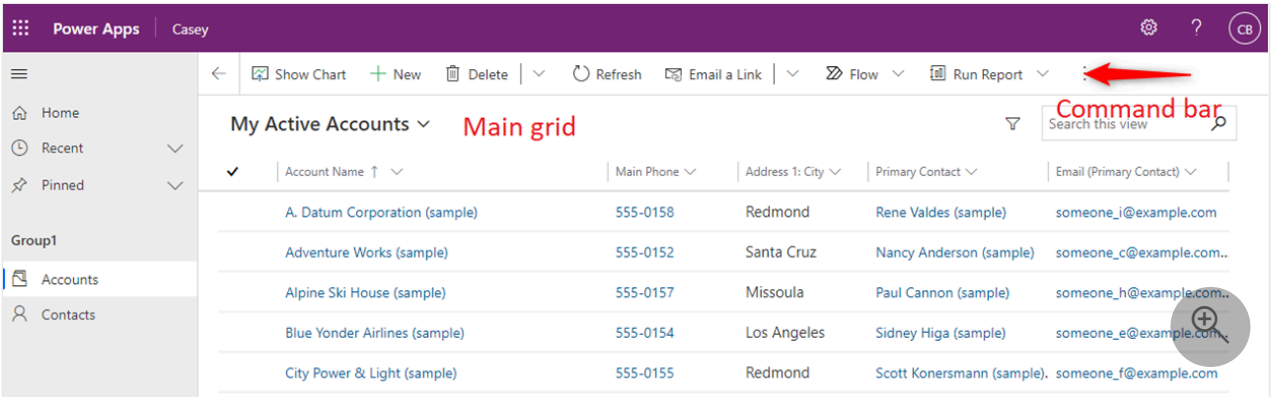


At a high level, command customization fits into three categories. Various capabilities exist within each category and are covered in more depth throughout the modern commanding documentation:

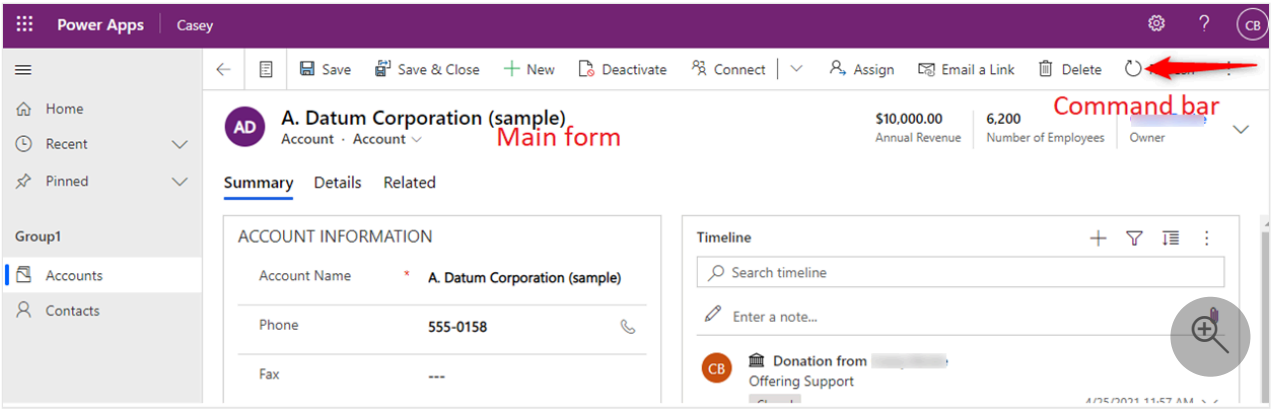
- **Display**. How the button appears and where it's located in an app. For example, the button's label, icon, and accessibility labels as well as the command bar location and position within a command bar.
- **Action**. The logic that runs when a user selects a button. For example, creating and updating data or interacting with various controls and pages inside the app.
- **Visibility**. Logical conditions that specify when a button is visible or hidden to a user. For example, you might want the button visible for some users and hidden for others. Or perhaps the button should only be visible when certain criteria of the data records are satisfied.

Command bar locations

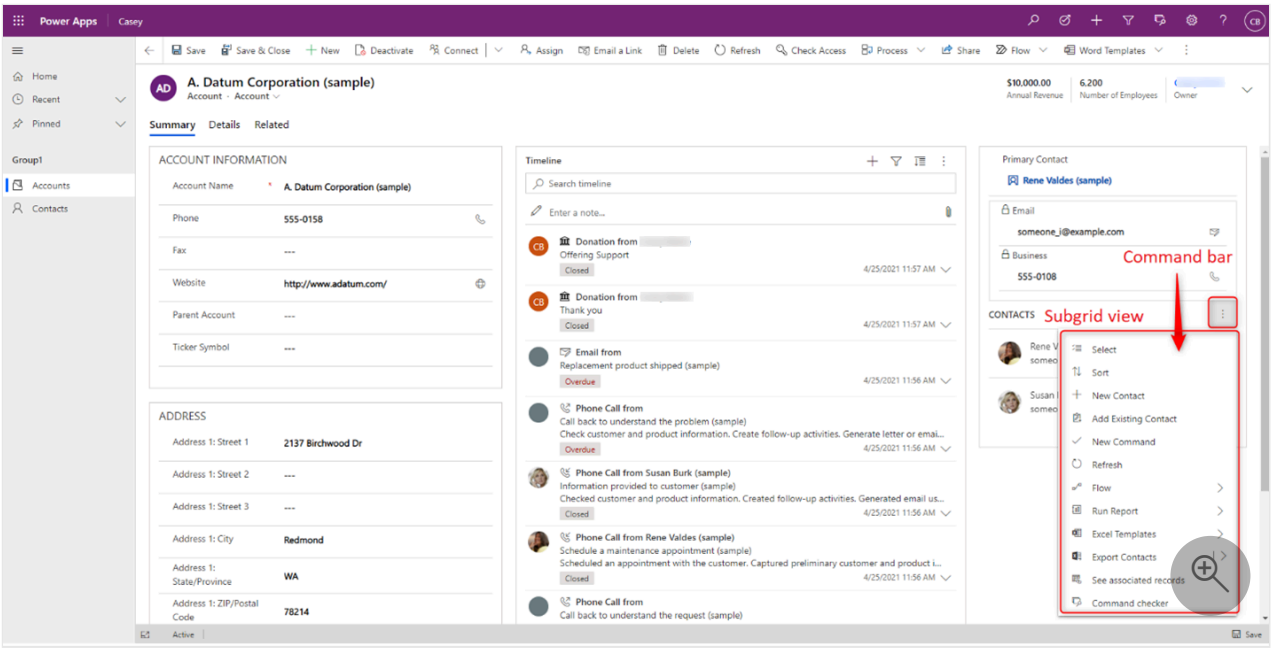
- **Main grid**. This command bar is displayed when using the left-hand navigation of an app to view a full page list of records in this table.



- **Main form**. This command bar is displayed on the table's main forms. This appears at the top of the form and isn't the same as the associated view or subgrid view that appears in different areas of the form.

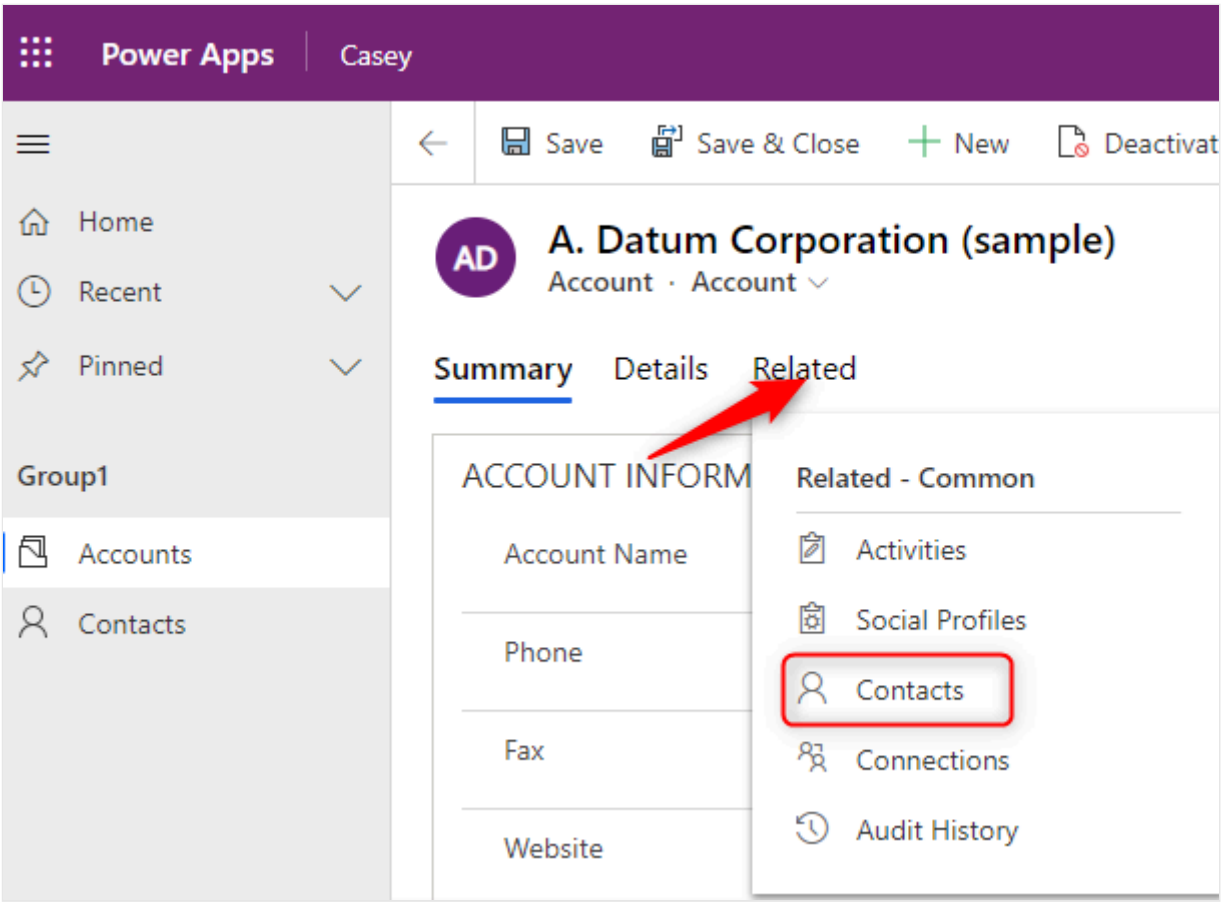


- **Subgrid view**. This command bar is displayed on forms of other tables that render this table's data within a subgrid. For example, the account main form has a subgrid control that lists contact records related to the account record. To edit the command bar shown here, edit the command bar for the contact table then subgrid view.

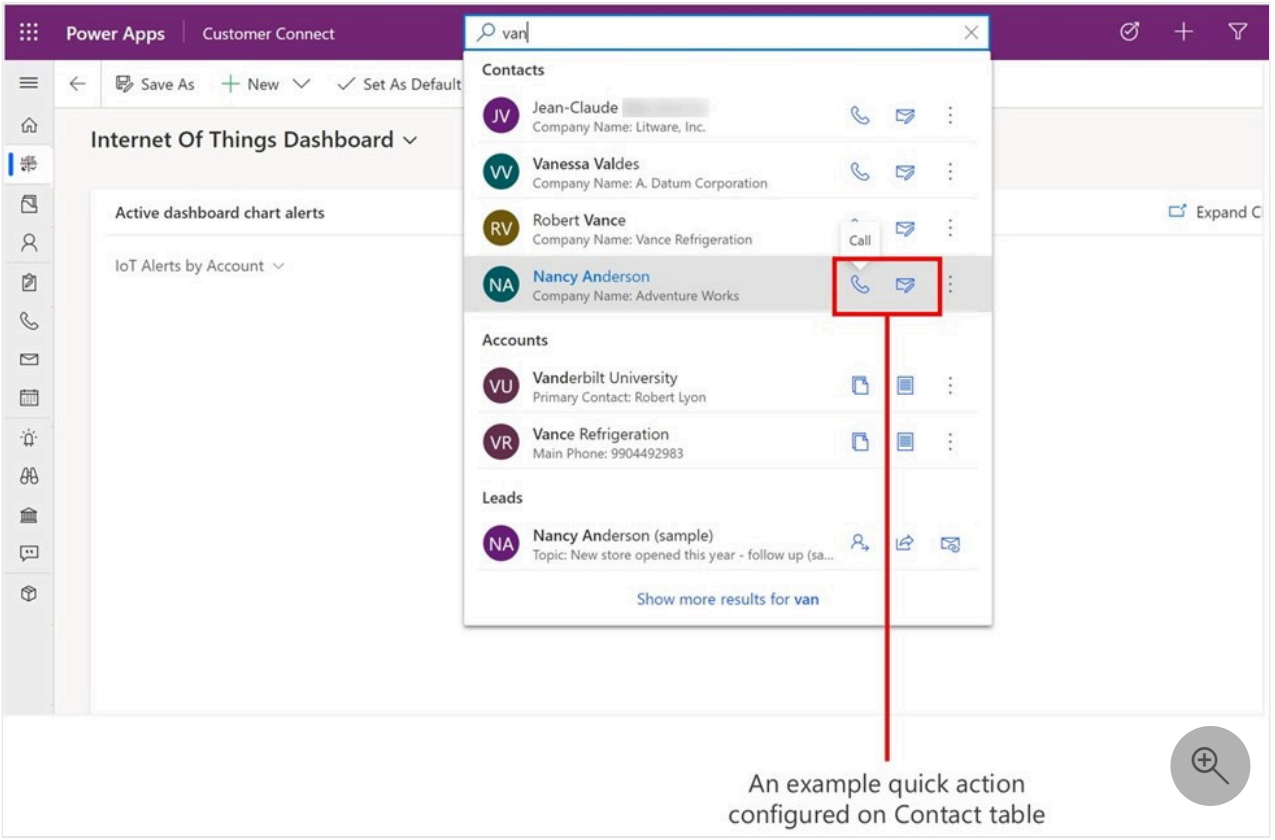


- **Associated view**. This command bar is displayed on the form of a parent table when viewing related data in this table. For example, from the main form of an account record

select the **Related** tab then select a related table like contacts.



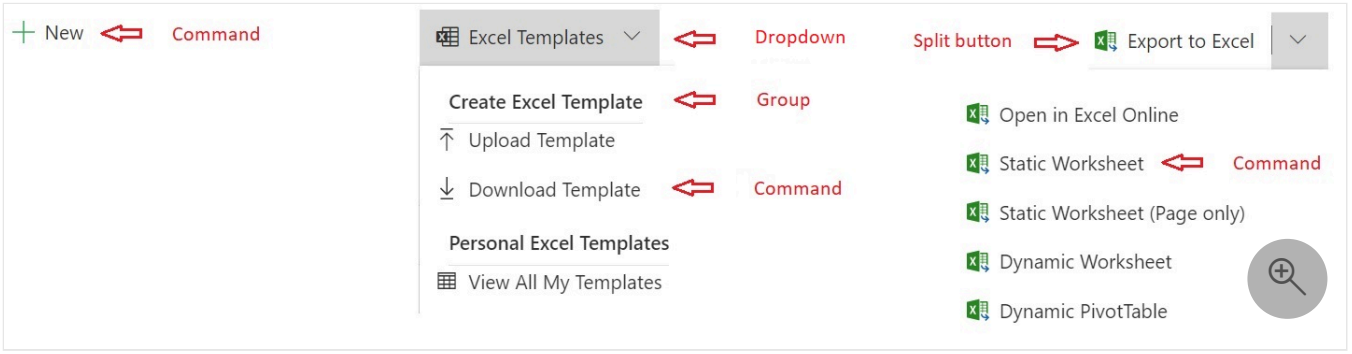
- **Quick actions**. Quick actions are associated with the main grid location. To add or edit commands for both quick actions and main grid locations, select the desired table from within the modern app designer, then edit the command bar and choose the main grid location. The first five commands, determined by order, are also shown as quick actions when playing the app.



> **Note**
>
> Less frequently customized command bar locations aren't supported in command designer. See the Global command bar and other ribbons sections for more information on customizing commands for these locations.

Types of commands

- **Command**. Standard button. Performs an action when selected. Can also be nested in groups within dropdowns and split buttons. These were called flyouts in classic commanding.
- **Dropdown**. Creates a menu where you can organize commands within a group.
- **Group**. Add titles to groups of commands nested within dropdowns and split buttons.
- **Split button**. Similar to a dropdown, but has a primary command. When the split button is selected, the action from the primary command is executed. If the expand chevron is selected, the primary command won't be executed. Instead a list expands to show additional groups, flyouts, and commands.



Key differences between classic and modern commands

Classic commands (formerly known as the ribbon) weren't customizable using low code. With code, command customizations were difficult, tedious, and error prone. In order to scale commanding to low code as well as use custom pages to converge canvas and model-driven apps, it was vital to reinvent and rebuild the command infrastructure.

Modern commanding offers many new capabilities and is much simpler to use.

Expand table

Capability	Classic	Modern
Supported in model-driven app runtime	Yes	Yes, Additionally supports Power Fx runtime.
Customized using	Hand editing XML within solution files or using non-Microsoft tools. Requires time-consuming solution export and import operations.	Command designer as well as Dataverse API support.
Supports Power Fx.	No	Yes. For actions and visibility.
Time required to customize	Slow, error prone.	Fast
Reliability and performance	Easy to make mistakes. Bad customization and lack of scoping often impact app performance.	Inline error handling prevents mistakes. Power Fx optimized for better runtime performance.
Sharing	Standard Dataverse role-based security.	Non-Power Fx commands use standard Dataverse role based security. Power Fx commands currently require the command component library to be shared in addition to having an appropriate security role.
Solution and application lifecycle management (ALM) behaviors	Inconsistent and problematic solution layering, no presence in solution interface. Many standard solution behaviors not supported such as patches, segmentation, solution upgrade, managed properties, and many more.	Standard solution layering centrally managed for multiple solution object types within Dataverse. Present in solution interface. All standard solution behaviors supported.
Localization	Nonstandard	Standardized using export and import translations for the entire solution.
Data model	Complex. Optimized for classic ribbons and contain many properties that are no longer needed.	Simple, optimized for today's model-driven app command bars.
Use JavaScript	Yes	Yes. Now simpler. The same JavaScript can be used for classic and modern commands.
Customize out of the box commands	Yes	Commands become editable in command designer once migrated to the modern framework.

Capability	Classic	Modern
App specific commands	No	Yes. Using modern command designer ensures commands are only visible within the selected app.
Table specific commands that display in all apps containing the table	Yes	Yes. Requires modifying the `appaction` definition within the solution file.
Global commands that display for all tables and apps for the specified command bar location	Yes	Yes. Requires modifying the `appaction` definition within the solution file.
Create split buttons, flyouts, and groups	Yes	Yes
Dynamically populate a flyout with code	Yes	No. Create commands declaratively.
Customize global application header commands	Yes	No
Customize commands for other, uncommon, or obsolete command bar locations	Yes	No.
Run a modern flow or workflow	Using JavaScript	Using JavaScript. Also supported using a custom page.

Classic versus modern visibility rule comparison

Classic visibility rules often had a specific rule for each scenario. With Power Fx, a declarative function replaces many classic rules. And it’s much simpler to use.

Classic visibility rules are also supported within modern commands. However, support for classic rules was needed for reliably migrating classic commands to modern commanding and classic rules customization within command designer isn't supported. We recommend you use Power Fx going forward.

Expand table

Use case	Classic Rule	Classic options	Power Fx visible property
Show or hide based on data values	CustomRule	Use JavaScript	!IsBlank(Self.Selected.Item.Email)
Show or hide based on table permission	EntityPrivilegeRule	Multiple	DataSourceInfo()
Show or hide based on record permission	RecordPrivilegeRule	Multiple	RecordInfo()
Reference the control context for primary and related tables	EntityRule	PrimaryEntity. SelectedEntity	Self.Selected
Reference the control context	EntityRule	Form. HomePageGrid. SubGridStandard. SubGridAssociated	Self.Selected
Table metadata properties	EntityPropertyRule		DataSourceInfo()
Show or hide based on form state. For example, show for the create form	FormStateRule	Create. Existing. ReadOnly. Disabled. BulkEdit	Self.Selected.State = FormMode.New
Show when > 1 records are selected in a grid	SelectionCountRule		CountRows(Self.Selected.Items) > 1
Show or hide for a related table in a polymorphic lookup. For example, check whether the lookup is a user or a team	CustomRule	PrimaryEntityTypeCode	IsType(), AsType
Reference environment properties (Org)	CustomRule	OrgName. OrgLcid. UserLcid	Not currently available

Frequently asked questions

- *Why do I see more commands in the designer than I see in my app?*

    - There are several reasons. Sometimes there's visibility logic that hides the command when running the app. Other times these commands are dynamically injected via custom JavaScript during runtime and aren't configurable.
- *Why do I see duplicate commands in the designer?*
    - This was a common pattern used with classic commands. Both commands wouldn't show up in runtime as they were controlled by visibility rules. The command designer shows all commands, regardless of their visibility rules.

See also

Customize the command bar using command designer
Manage commands in solutions
Modern commanding known limitations

Last updated on 03/31/2026

Edit system table messages (preview)

Article • 03/13/2023

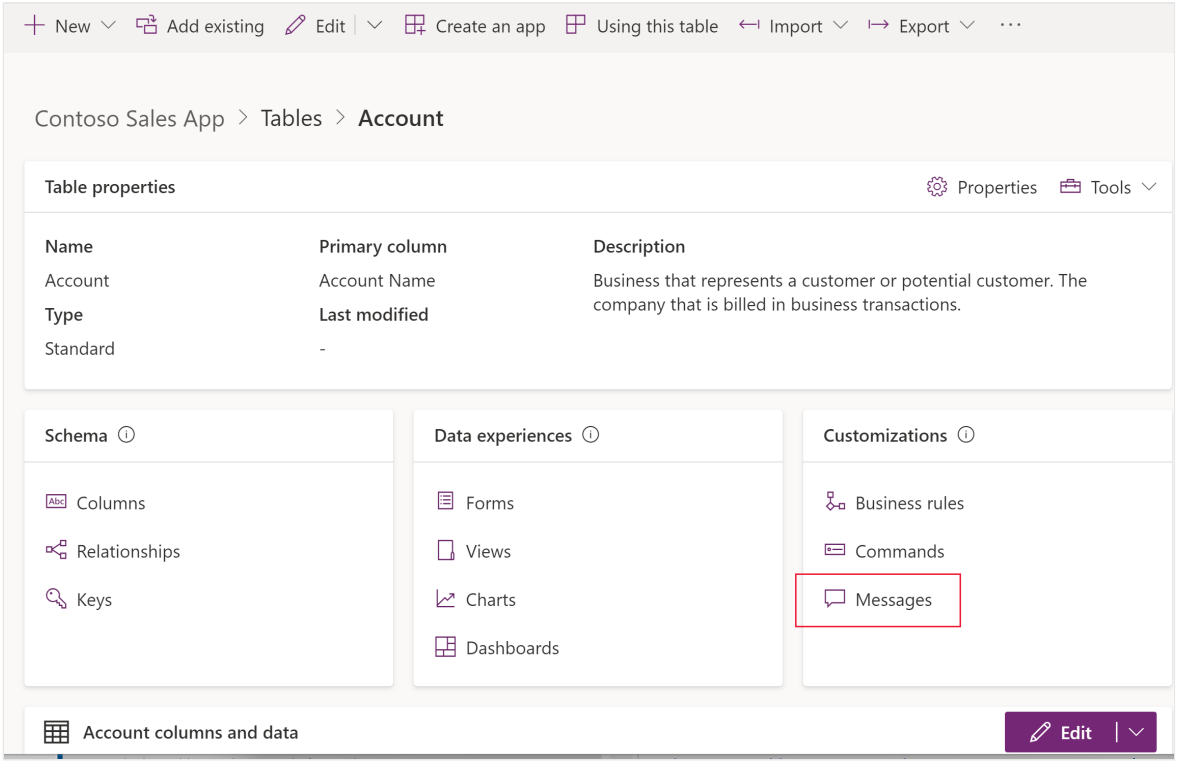
[This topic is pre-release documentation and is subject to change.]

The default display name of some system tables, such as account and contact tables, is used in user interface text and error messages in Microsoft Dataverse. If you change the display name for a system table, you should also update any messages that use the default display name. For example, if you change the display name from *Account* to *Company*, you could still see an error message using the old name.

> **Important**
>
> This is a preview feature. More information: **Model-driven apps and app management**

Add a message to a system table

1. Sign into Power Apps (make.powerapps.com).
2. Go to **Solutions**, open the solution that has the table you want, and then open the table. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. In the **Customizations** area, select **Messages**.

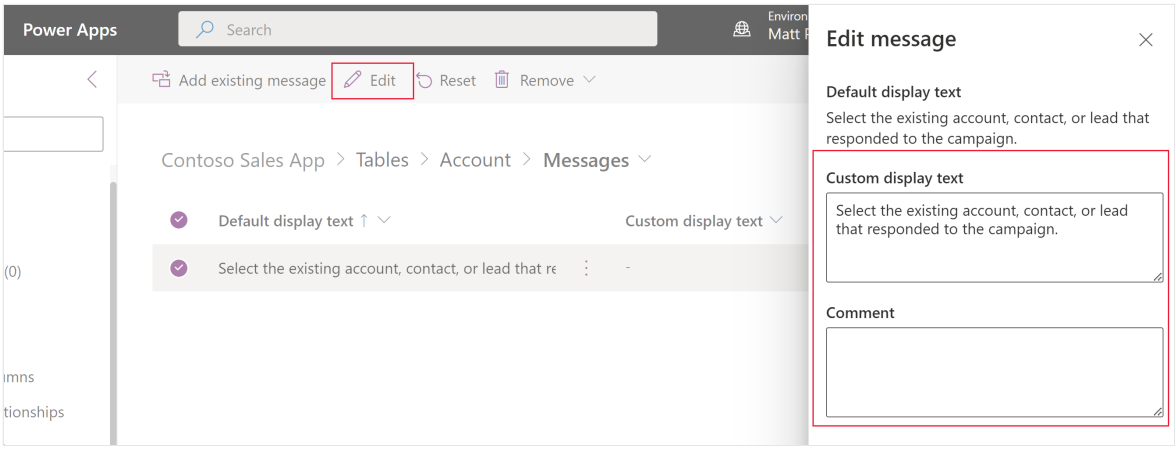


4. Select the message you want, and then select **Add**.

Edit a system table message

1. Open the table that has the message you want to edit, and then in the **Customizations** area select **Messages**.

2. Select the message, and then select **Edit** on the command bar. Enter the changes you want:



Field	Description
Default display text	Shows the original text, which isn't editable.
Custom display text	Edit this text to change the message that is displayed.
Comment	Optional. Include a comment about what you changed and why.

3. Select **Done**.

Programmatically update table display strings

For developers looking for a way to work with these in code, the display strings are stored in the DisplayString table.

The `DisplayString` table doesn't contain the default display strings. The two attributes for this table that contain text are CustomDisplayString and PublishedDisplayString. By default, these attribute values are null unless the display string has been customized and published. The `PublishedDisplayString` value is read-only and reflects the currently published `CustomDisplayString`.

Edit using the classic experience

For information about how to edit system table messages using the classic experience, see Edit system entity messages and display names.

See also

Edit a table

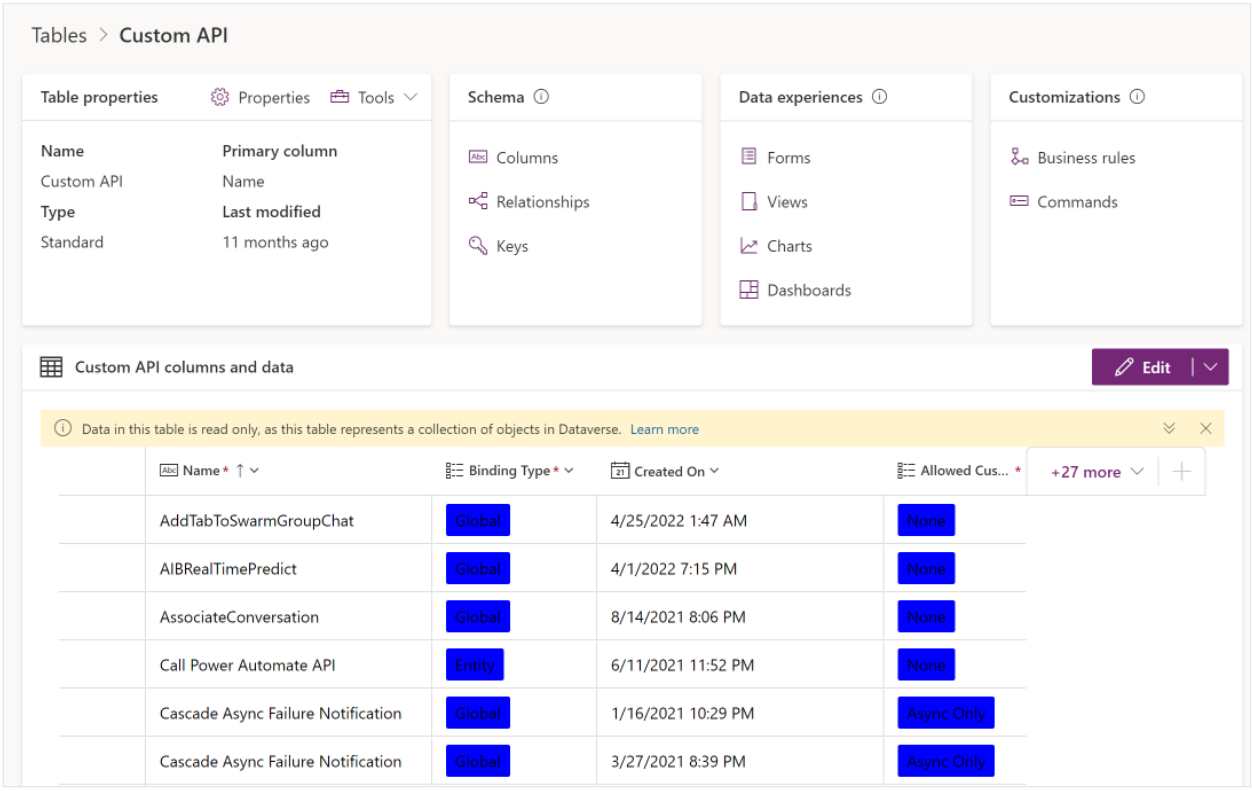
Translate localizable text for model-driven apps

View and edit tables representing solution objects

Article • 08/08/2023

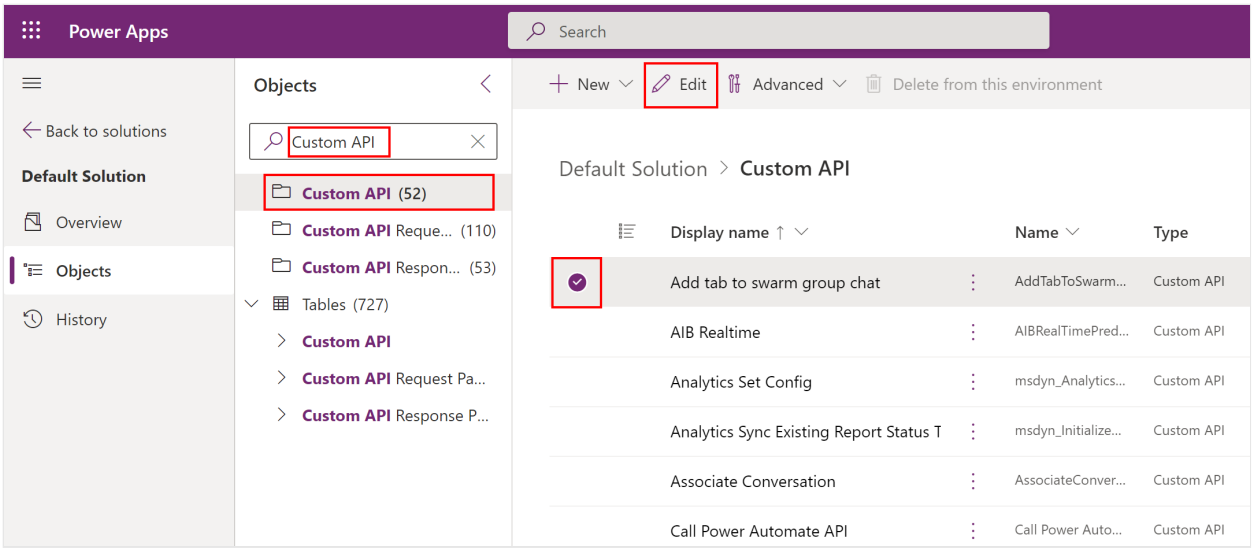
Some standard tables in Dataverse represent the structure and instances of solution objects. If you aren't familiar with solutions, go to Solutions overview.

For example, the **Custom API** table has columns that represent the properties of an API, such as **Name** and **Binding Type**, with each row of data in the table representing a particular API instance. For more information about Custom API table, go to Custom API table columns



When viewing these tables in Power Apps (make.powerapps.com), you'll see a warning stating that the data in the table is read only. This ensures the instances of these objects aren't changed.

To update data in such tables, you must navigate to the respective objects in the solution. In our example of custom API, you can open a solution that contains the custom API object or add the custom API object to a new solution and then customize the data as required.



See also

Create a table
Edit a table
Create a solution
Create and use custom APIs

Define alternate keys using Power Apps

Article • 05/15/2025

Power Apps provides an easy way to view and create table alternate keys with Microsoft Dataverse. *Alternate keys* provide an efficient and accurate way of integrating data with external systems. It's essential in cases when an external system doesn't store the globally unique identifier (GUID) IDs that uniquely identify rows in Microsoft Dataverse. For more information about alternate keys, see Define alternate keys to reference rows.

> **Important**
>
> If the data within a column that is used in an alternate key contains one of the following characters `/`, `#`, `<`, `>`, `*`, `%`, `&`, `:`, `\\`, `?`, `+` then `GET` or `PATCH` actions won't work. If you only need uniqueness, then this approach works, but if you need to use these keys as part of data integration then it's best to create the key on columns that won't have data with those characters.

View alternate keys

1. In Power Apps, go to **Solutions** and open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **Tables** on the left **Objects** pane, and then open the table that you want to view.
3. In the **Schema** area, select **Keys** to view a list of any alternate keys that might be defined.

Create an alternate key

1. While viewing alternate keys, select **New key**.

2. In the Key properties pane, enter the following information:
   a. Set a **Display name** and choose the columns to use to create the alternate key.
   b. The **Name** column is populated based on the display name. You can change it if you want.
   c. Select the columns that you want. For example, to identify an account row with an alternate key, you can use the **Account Number** column in combination with some other column, such as the **SIC Code**, which have values that shouldn't change.

Key

Display name *

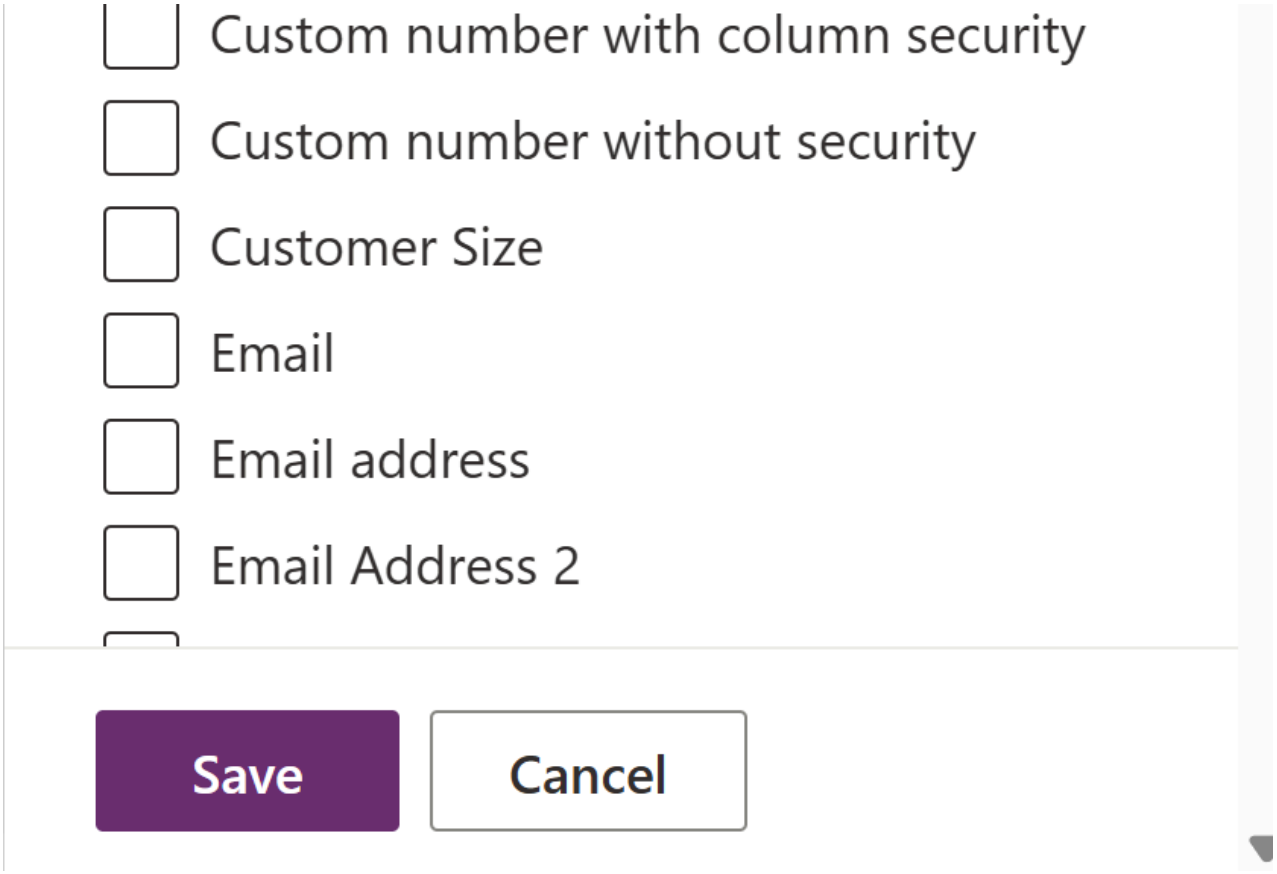
Account Number + SIC Code

Name *

contoso_ AccountNumberSICCode

Columns *

- [ ] (Deprecated) Traversed Path
- [ ] Account Name
- [x] Account Number
- [ ] Account Rating
- [ ] Business Type
- [ ] Category
- [ ] Classification
- [ ] Color
- [ ] Created By (IP Address)
- [ ] Created By (User Name)
- [ ] Currency



3. Select **Save** to create the alternate key.

> **Note**
>
> The alternate key isn't immediately available for use. A system job is initiated when you save the table to create database indexes to support the alternate key.

Delete an alternate key

While viewing alternate keys, select the key you want to delete and select **Remove** > **Remove from this solution** or to delete the key from the Power Platform environment **Delete from this environment** on the command bar.

Related articles

Developer Documentation: Work with alternate keys

Define alternate keys using solution explorer

Article • 05/30/2023

Solution explorer provides one way to view and create alternate keys for Microsoft Dataverse.

The Power Apps portal enables configuring the most common options, but certain options can only be set using solution explorer.
More information:

- Define alternate keys to reference rows
- Define alternate keys using Power Apps portal

> **Important**
>
> If the data within a column that is used in an alternate key will contain one of the following characters `/,<,>,*,%,&,:,\\,?,+` then `GET` or `PATCH` actions will not work. If you only need uniqueness then this approach will work, but if you need to use these keys as part of data integration then it is best to create the key on columns that won't have data with those characters.

Open solution explorer

Part of the name of any alternate key you create is the customization prefix. This is set based on the solution publisher for the solution you're working in. If you care about the customization prefix, make sure that you are working in an unmanaged solution where the customization prefix is the one you want for this table. More information: Change the solution publisher prefix

Open an unmanaged solution

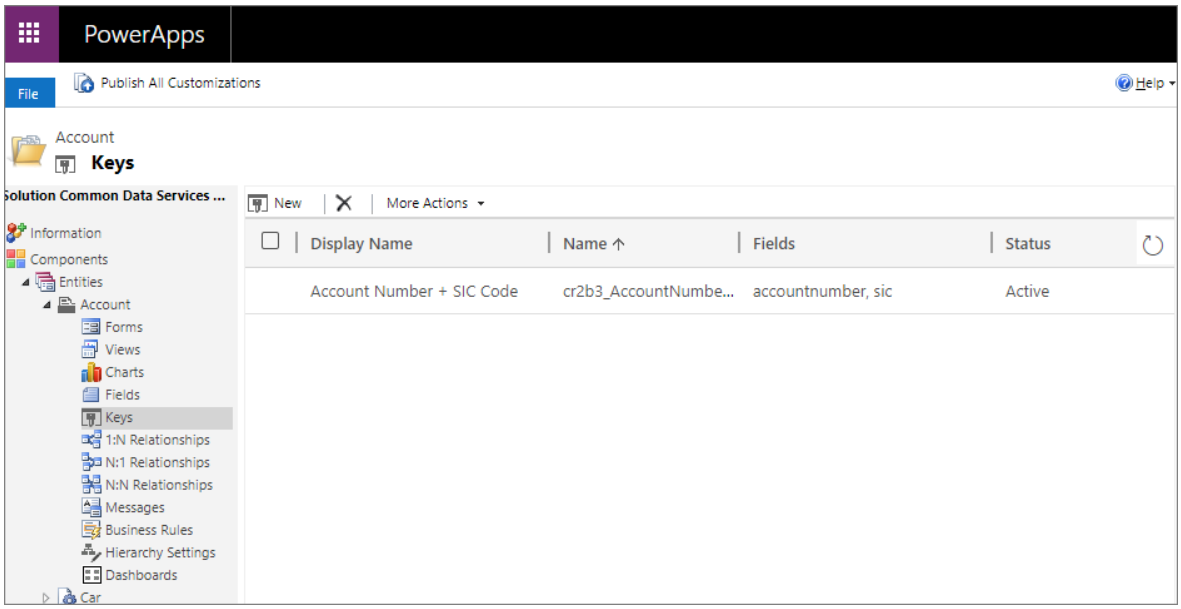
1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. In the **All Solutions** list open the unmanaged solution you want.

View alternate keys

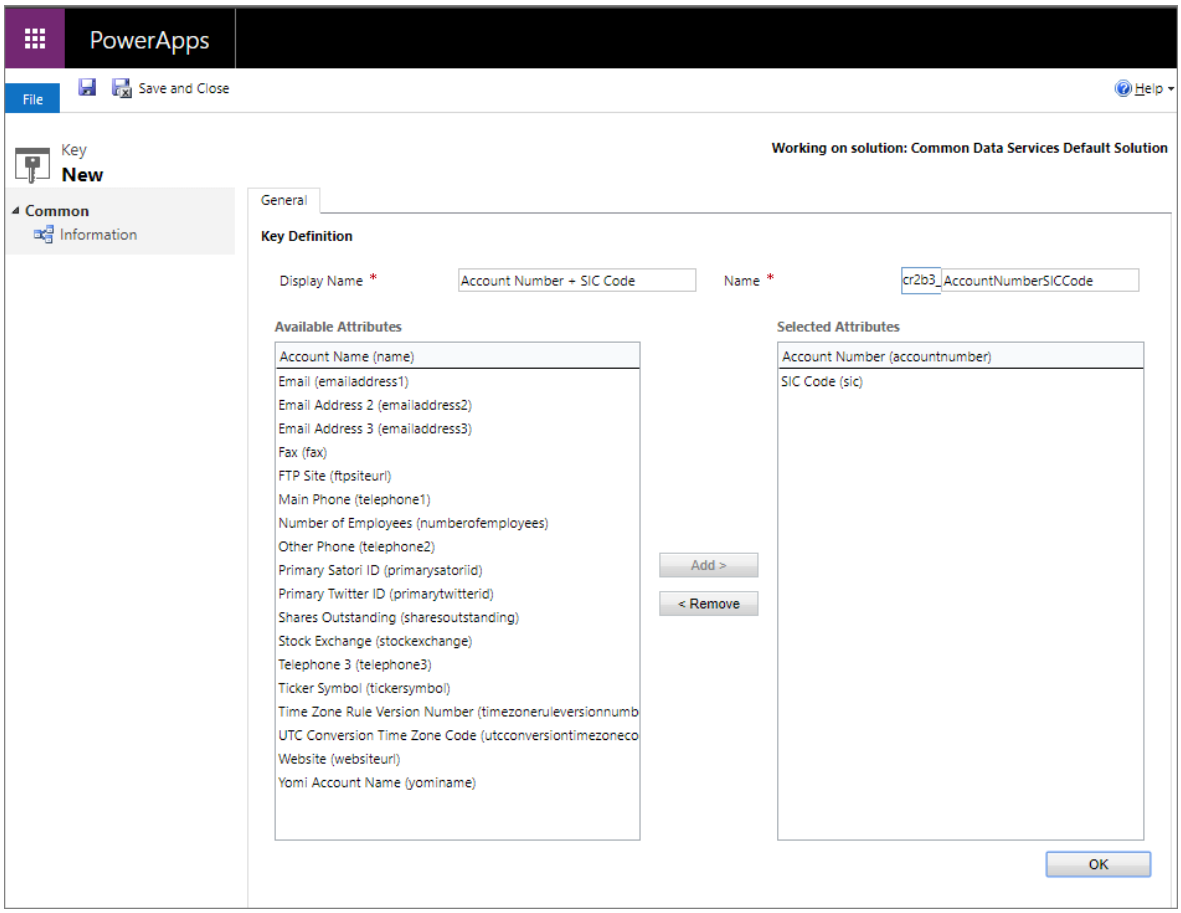
1. With solution explorer open, under **Components** expand **Entities** and select the table where you want to view alternate keys.

2. Expand the table and select **Keys**.



Create an alternate key

1. While viewing alternate keys, select **New**.
2. On the form, enter the **Display Name**. The **Name** column will be auto-populated based on the **Display Name**.
3. From the **Available Attributes** list, select each attribute and then **Add** > to move the attribute into the **Selected Attributes** list.



4. Select **OK** to close the form.

(Optional) View the system job tracking creation of indexes

1. While viewing alternate keys after you have created a new alternate key you will see a row for the key you created.

2. In the **System Job** column you will find a link to the system job that monitors the creation of the indexes to support the alternate key.

   This system job will have a name that follows this pattern: `Create index for {0} for table {1}` where `0` is the **Display Name** of the alternate key and `1` is the name of the table.

   The link to the system job will not be displayed after the system job completes successfully. More information: Monitor system jobs

Delete an alternate key

While viewing alternate keys, select ✕.

See also

Define alternate keys to reference rows

Define alternate keys using Power Apps portal

Developer Documentation: Work with alternate keys

Update forms and views using table designer

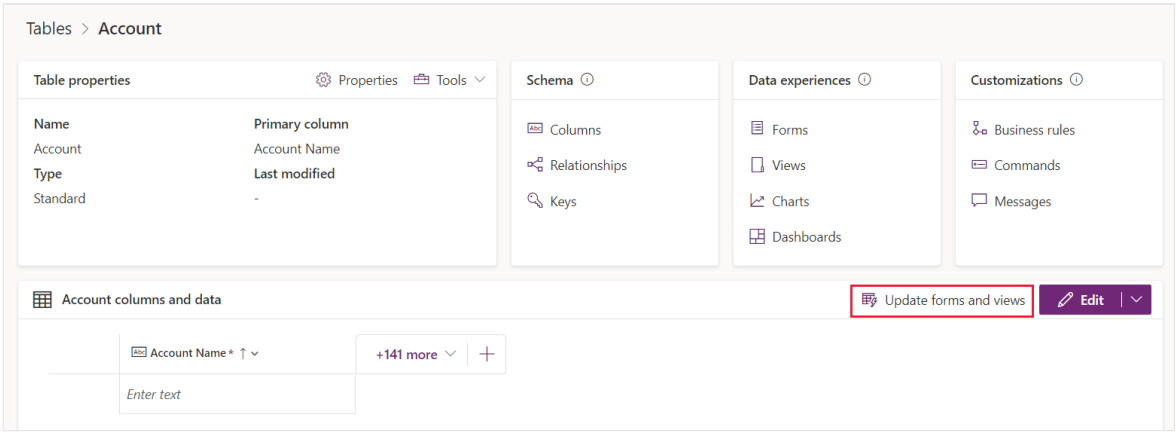
Article • 08/25/2023

Create columns directly in the table hub or table designer, which will then be added to your forms and views. In this walkthrough, you'll learn how to simply add the columns you created to forms and views of this table without manually doing so in the form or view designer.

Using table hub and table designer

1. Sign in to Power Apps.

2. In the left navigation pane, select **Tables**, and then select the table that you want to update from the table list. In the table hub and table designer, there's an **Update forms and views** option that allows you to directly add columns to selected forms and views in this table.





3. Select **Update forms and views** to configure what columns should be added to certain forms and views:

Property	Description	Available option	Default option
Add these columns	Selected columns in this dropdown list will be added to the selected forms and views.	Only columns currently shown in the table preview.	Both new columns that are recently created and existing columns that are

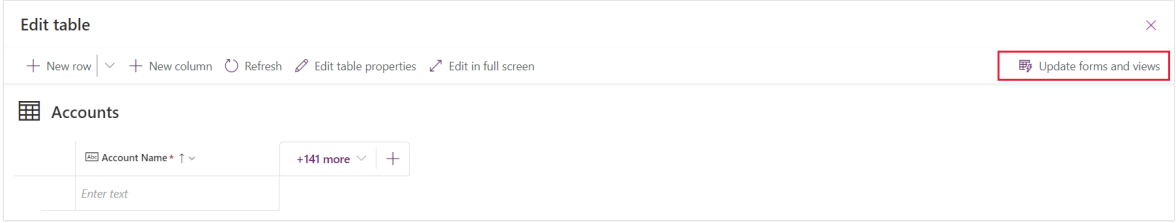
Property	Description	Available option	Default option
			recently added to the table.
To these forms	Columns selected in **Add these columns** will be added to selected forms. If a certain column has already been added to this form before, it won't be added again as a duplicate.	All main, quick create, and quick view forms from this table.	All Main forms from this table.
And to these views	Columns selected in **Add these columns** will be added to selected views. If a certain column has already been added to this view before, it won't be added again as a duplicate.	All public, advanced find, associated, quick find, and lookup views from this table.	All public, advanced find, associated, and quick find views from this table.

4. Once you're done with the configuration, select **Update**. The columns you selected are added to the forms and views you selected.

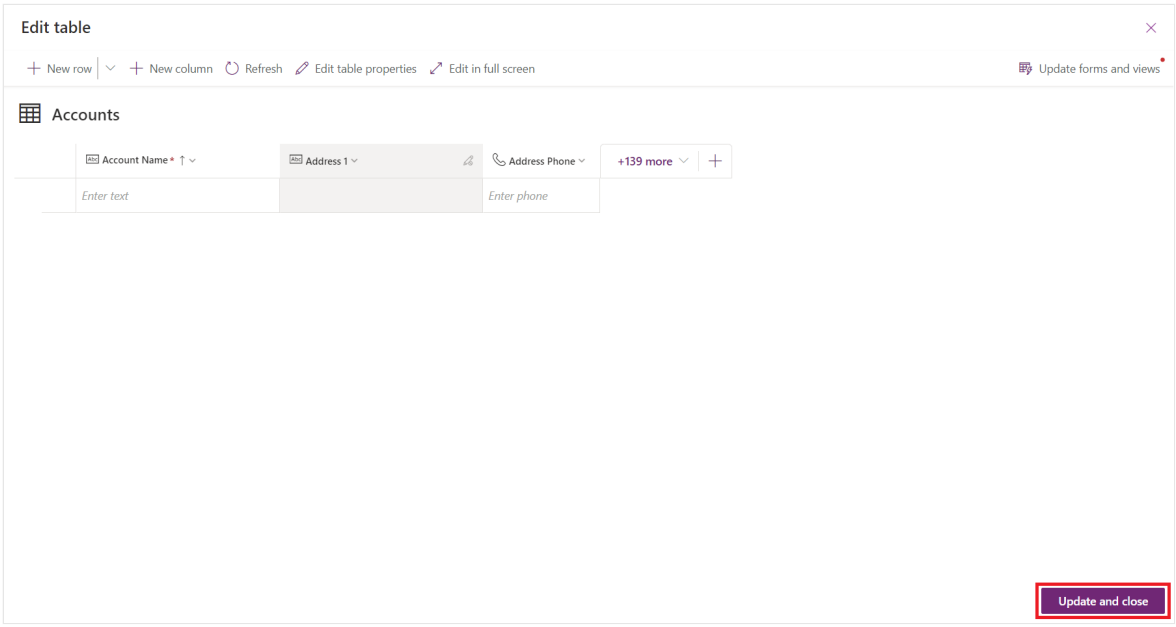
Using inline table designer in modern app designer

You might be working on a table in the modern app designer as you're building an app. In this case, you can also update your forms and views directly using the inline table designer in the modern app designer:

1. Sign in to Power Apps.
2. On the left navigation pane, select **Apps**, select the app you want, and then select **Edit**.
3. In the left navigation pane within the modern app designer, select **Data**, then **...** > **Edit table** to edit your table using the inline table designer.
4. The **Update forms and views** option is displayed in the inline table designer:



Follow the same steps in 3 and 4 as in the table hub and table designer to update your forms and views. You can also add the columns you want to your table. When you're finished with the update, select **Update and close** at the bottom right of the inline table designer. The columns are added to the forms and views preselected for you, and the inline table designer is closed.



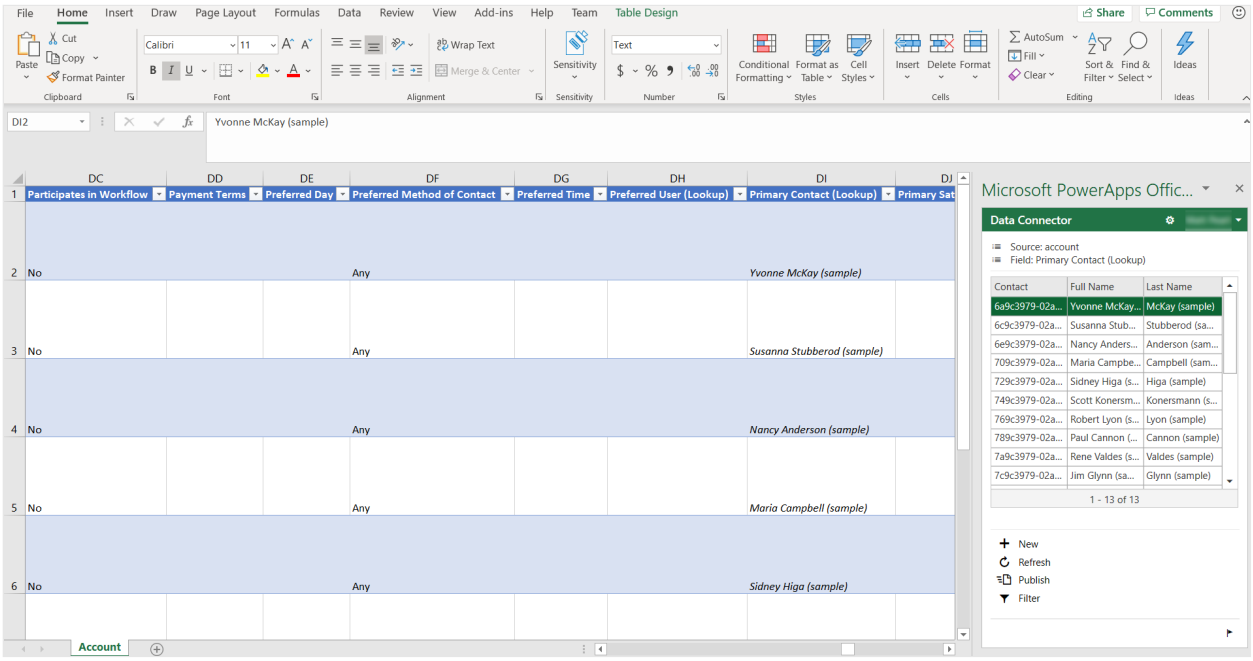
See also

Create a custom table

Open table data in Excel

Article • 03/13/2023

By opening table data in Microsoft Excel, you can quickly and easily view and edit data by using the Microsoft Power Apps Excel Add-in.



To install the Power Apps Excel add-in, see Microsoft PowerApps Office Add-in. For more information about how to add or remove an Office Excel Add-in, see Add or remove add-ins in Excel.

Open table data in Excel

1. Sign in to Power Apps.
2. In the left navigation pane, select **Tables**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. Select the ellipsis (...) to the right of the table that you're interested in, and then select **Edit data in Excel**.
4. Open the Excel worksheet that is downloaded to your browser's default download folder named similar to *table-name (1591125669213).xlsx*. This workbook has binding information for the table, a pointer to your environment, and a pointer to the Power Apps Excel add-in.
5. In Excel, select **Enable editing** to enable the Power Apps Excel Add-in to run. The Excel add-in runs in a pane on the right side of the Excel window.

> **ⓘ Important**
>
> If the pane displays an error message, see **Office Store Add-in download disabling**.

6. If this is the first time that you've run the Power Apps Excel add-in, select **Trust this Add-in** to allow the Excel add-in to run.

7. If you're prompted to sign in, select **Sign in**, and then sign in by using the same credentials that you used on Power Apps. The Excel add-in will use a previous sign-in context and automatically sign you in if it can. Therefore, verify the user name in the upper right of the Excel add-in.

The Excel add-in automatically reads the data for the table that you selected. Note that there will be no data in the workbook until the Excel add-in reads it in.

View and refresh data in Excel

After the Excel add-in reads table data into the workbook, you can update the data at any time by selecting **Refresh** in the Excel add-in.

Edit data in Excel

You can change table data as you require and then publish it back by selecting **Publish** in the Excel Add-in.

To edit a row, select a cell in the worksheet, and then change the cell value.

To add a new row, follow one of these steps:

- Select anywhere in the worksheet, and then select **New** in the Excel add-in.
- Select in the last row of the worksheet, and then press the Tab key until the cursor moves out of the last column of that row, and a new row is created.
- Select in the row immediately below the worksheet and start to enter data in a cell. When you move the focus out of that cell, the worksheet expands to include the new row.

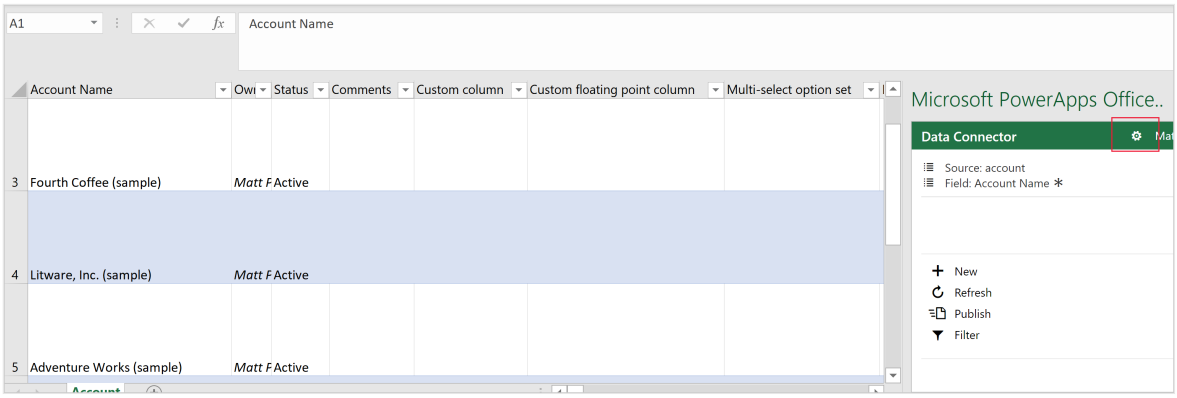
To delete a row, follow one of these steps:

- Right-click the row number next to the worksheet row to delete, and then select **Delete**.
- Right-click in the worksheet row to delete, and then select **Delete** > **Table Rows**.

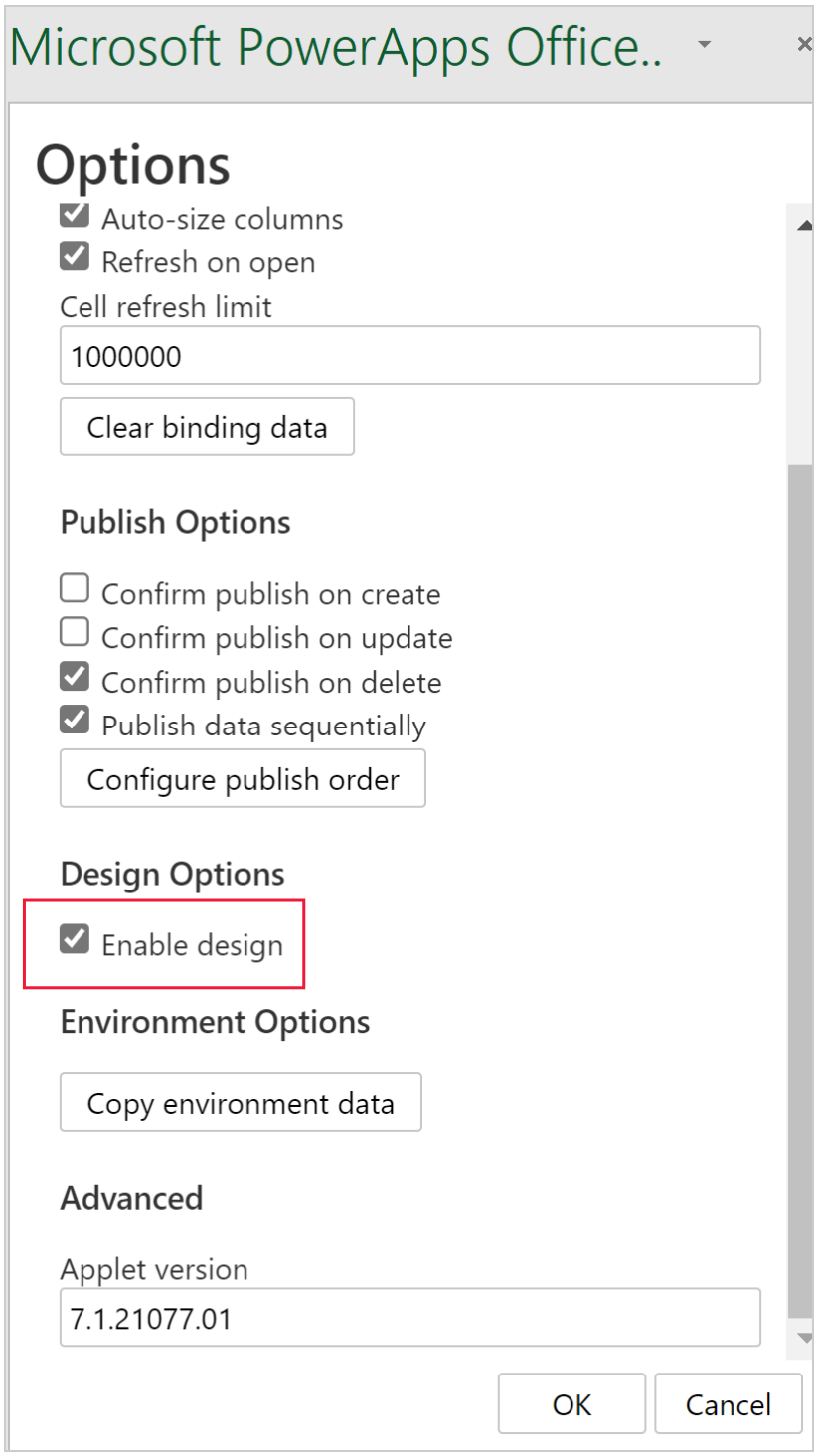
Configure the add-in to adjust tables and columns

You can use the data source designer in the Excel add-on to adjust the columns and tables that are automatically added to the worksheet.

1. Enable the data source designer of the Excel add-in by selecting the **Options** button (the gear symbol).



2. Expand the **Data Connector** section, and then select the **Enable design** check box, and then select **OK**.



3. Select **Design** in the Excel add-in. All the data sources are listed.

4. Next to the data source, select the **Edit** button (the pencil symbol).

5. Adjust the list in the **Selected columns** column as you require:

   - To add a column from the **Available columns** column to the **Selected columns** column, select the column, and then select **Add**. Alternatively, double-click the column.
   - To remove a column from the **Selected columns** column, select the column, and then select **Remove**. Alternatively, double-click the column.

- To change the order of columns, select the column in the **Selected columns** column, and then select **Up** or **Down**.

6. Apply your changes to the data source by selecting **Update**, and then select **Done** to exit the designer. If you added a column (column), select **Refresh** to pull in an updated set of data.

> **Note**
>
> - Make sure to always include the ID and required columns in your workbook, as you may receive errors when publishing.
> - When adding look up columns, make sure to add both the **ID** and the **Display** columns.

Troubleshooting

- Not all tables support editing and creation of new rows, these tables will open in Excel and allow you to view data but publishing will be disabled.
- Look up columns must be edited using the add-in to ensure the correct row is referenced, updating these columns via copy and past or typing directly into the column is not supported.
- Image and file columns are not supported. You won't be able to see any data when you open an Excel worksheet that includes those data types.
- The maximum table size when editing in Excel is one million cells. If there are too many rows or columns, not all data will be read or published.
- Rows are processed sequentially. If there is an error in saving a row, changes to subsequent rows will be aborted. Changes to preceding rows are kept.
- Centralized deployment is not supported. This add-in requires access to the Office store.
- Manually configuring the add-in might cause unexpected behavior. For best results, download the Excel file with Power Apps instead of starting from a blank Excel workbook.

If you encounter an issue that isn't described here, contact us via the support pages.

Office Store add-in download disabling

If Office Store add-in downloading has been disabled for your organization, you might receive the following error message when you open the Excel worksheet after selecting the Edit data in Excel command.

Office 365 has been configured to prevent individual acquisition and execution of Office Store Add-ins.

This setting prevents the Power Apps Excel Add-in from downloading. When this occurs, table row data will not be displayed in Excel.

For more information about enabling Office Store Add-in downloading, contact your Office app administrator.

For more information about preventing Office Store Add-in downloading from the Office Store, see Prevent add-in downloads by turning off the Office Store across all clients.

Other ways to export and view table row data

For more information about ways to export and view table row data, see these articles.

- Export data to CSV
- Export your data to Excel Online
- Export table data to Azure Data Lake Storage Gen2
- Self-service data prep with dataflows

See also

- Manage columns in a table
- Define relationships between tables
- Generate an app by using Microsoft Dataverse
- Create an app from scratch using Dataverse

Define alternate keys to reference rows

08/11/2025

Alternate keys provide an efficient and accurate way of integrating data with external systems. It's essential in cases when an external system doesn't store the Globally Unique Identifier (GUID) IDs that uniquely identify rows in Microsoft Dataverse.

A data integration system uses alternate keys to uniquely identify rows using one or more table column values that represent a unique combination. Each alternate key has a unique name.

For example, to identify an account row with an alternate key, you can use the account number or the account number column in combination with some other columns, which have values that shouldn't change.

> **Note**
>
> While you can define alternate keys with Power Apps, they can only be used programmatically in code. To learn more about using alternate keys programmatically, see:
>
> - Developer Documentation: Use an alternate key to reference a record
> - Developer Documentation: Retrieve record using an alternate key

Some of the benefits of the alternate keys feature include:

- Faster lookup of the rows.
- More robust bulk data operations.
- Simplified programming with data imported from external systems without row IDs.

Creating an alternate key

There are two designers you can use to create alternate keys:

Expand table

Designer	Description
Power Apps portal	Provides an easy streamlined experience, but some options aren't available. More information: Define alternate keys using Power Apps portal
Solution explorer	Not as easy, but provides for more flexibility for less common requirements. More information: Define alternate keys using solution explorer

> **Note**
>
> You can also create an alternate key in your environment using the following:
>
> - Import a solution that contains the definition of the alternate key.
> - A developer can also write code to create them. More information: Developer Documentation: Work with alternate keys

Information in this article helps you choose which designer you can use.

You should use the Power Apps portal to create alternate keys unless you need to address any of the following requirements:

- Create an alternate key within a solution other than the Common Data Service Default Solution.
- You want to easily track the system job created that tracks the progress of creating the supporting indexes.

Limits in creating alternate keys

There are constraints on alternate key creation.

Fields that can be used for alternate keys

Only these kinds of columns can be used to create alternate keys:

- Decimal
- Whole Number (Integer)
- Single line of Text (String)
- Date and Time
- Lookup
- Choice

> **Note**
>
> - Columns that have the **Enable column security** property enabled can't be used as an alternate key. More information: Field security tables

- When NULL values are used in alternate key columns, uniqueness will not be enforced. To avoid duplicate records, don't use null values in the columns defined in the unique constraint of the alternate key.

Number of keys

You can define up to 10 different keys for a table.

Valid key size

When a key is created, the system validates that the key can be supported by the platform, including that the total key size doesn't violate SQL-based index constraints like 900 bytes per key and 16 columns per key. If the key size doesn't meet the constraints, an error message is displayed.

Unicode characters in key value

If the data within a column that is used in an alternate key contains one of the following characters `<`, `>`, `*`, `%`, `&`, `:`, `/`, `\\`, `#` then update or upsert (PATCH) actions won't work.

If you only need uniqueness, then this approach works, but if you need to use these keys as part of data integration then it's best to create the key on columns that doesn't have data with these characters.

Track the status of the creation of the alternate key

When an alternate key is created, it initiates a system job to create indexes on the database tables to enforce unique constraints on the columns used by the alternate key. The alternate key won't be in effect until these indexes are created. Creating these indexes might take some time depending on the amount of data in the system.

The status of the system job determines the state of the alternate key. The alternate key can have the following states:

- **Pending**
- **In Progress**
- **Active**
- **Failed**

When the system job is completed, the alternate key status is **Active** and it's available for use.

If the system job fails, locate the system job to view any errors. The system job will have a name that follows this pattern: `Create index for {0} for table {1}` where `0` is the **Display Name** of the alternate key and `1` is the name of the table.

> **Note**
>
> If you want to monitor the status of the system job you should use solution explorer to create the index. It will include a link to the system job so you can monitor it. More information: (Optional) View the system job tracking creation of indexes

See also

Define alternate keys using Power Apps portal
Define alternate keys using solution explorer
Developer Documentation: Work with alternate keys
Developer Documentation: Use an alternate key to reference a record
Developer Documentation: Retrieve record using an alternate key

Define status reason transitions for the Case or custom tables

Article • 06/21/2022

You can specify status reason transitions for the Incident (**Case**) table or a custom table.

> **Note**
>
> Although the Incident (Case) table isn't included in a default Microsoft Power Apps environment, it is used by **Dynamics 365 for Customer Service** and defined within the **Common Data Model**

Status reason transitions are an optional additional level of filtering to define what the status reason value can be changed to for each status reason. Defining a limited list of valid options can make it easier for people to choose the correct next status reason for a row when you have a large number of combinations for valid status reason values.

What is the connection between Status and Status Reason columns?

Tables that can have different status values have two columns that capture this data:

Display Name	Description
Status	Represents the state of the row. Typically **Active** or **Inactive**. You cannot add new status options.
Status Reason	Represents a reason that is linked to a specific status. Each status must have at least one possible status reason. You can add additional status reason options.

The metadata for the column defines what status values are valid for a given state. For example, for the Incident (**Case**) table, the default status and status reason options are:

Status	Status Reason
Active	• **In Progress** • **On Hold** • **Waiting for Details** • **Researching**

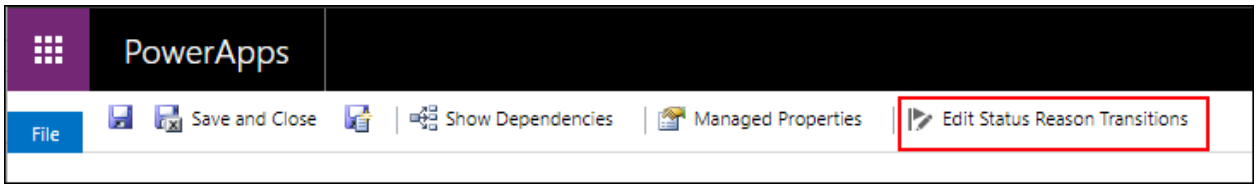
Status	Status Reason
Resolved	• Problem Solved • Information Provided
Canceled	• Canceled • Merged

Edit status reason transitions

You can modify the status reason column options for the Case table and custom tables to define which other status reason options people can choose. The only restriction is that each status reason option for an active status must allow at least one path to an inactive status. Otherwise you could create a condition where it would not be possible to resolve or cancel the case.

> ⓘ Note
>
> Editing the status reason transitions requires using solution explorer. See **Create and edit columns for Dataverse using Power Apps solution explorer** for information about how to edit columns.

When you edit a status reason column the **Edit Status Reason Transitions** button is in the menu.



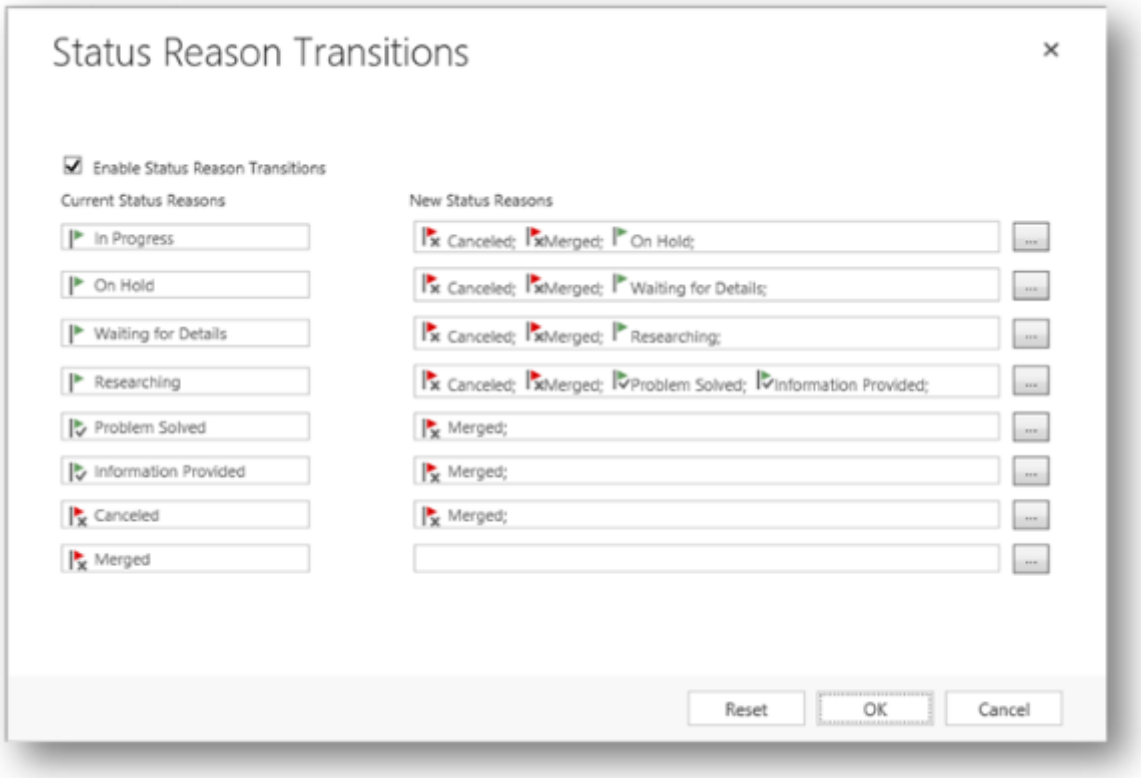
When you click this button the **Status Reason Transitions** dialog provides the option to choose **Enable Status Reason Transitions**. When this option is selected you must define which *other* status reason values are allowed for each status reason. To remove the filtering applied, remove the **Enable Status Reason Transitions** selection. The transitions you have defined will be kept but not applied.

The screenshot below provides an example that meets the following requirements:

- A case can be merged at any time. You will not be able to merge cases if a status reason transition does not allow for it.
- An active case can be canceled at any time.
- A resolved or canceled case cannot be reactivated.

- All cases must pass through the following stages: **In Progress** > **On Hold** > **Waiting for Details** > **Researching** before they can be resolved. With this configuration, a case could not be set to an earlier status.

  > **Note**
  >
  > This is not a good example for real work, but it demonstrates how stages of status can be enforced through status reason transitions.



See Also

Create and edit columns for Dataverse using Power Apps solution explorer
Table metadata > Table states
Define custom state model transitions

Delete a table

Article • 02/24/2023

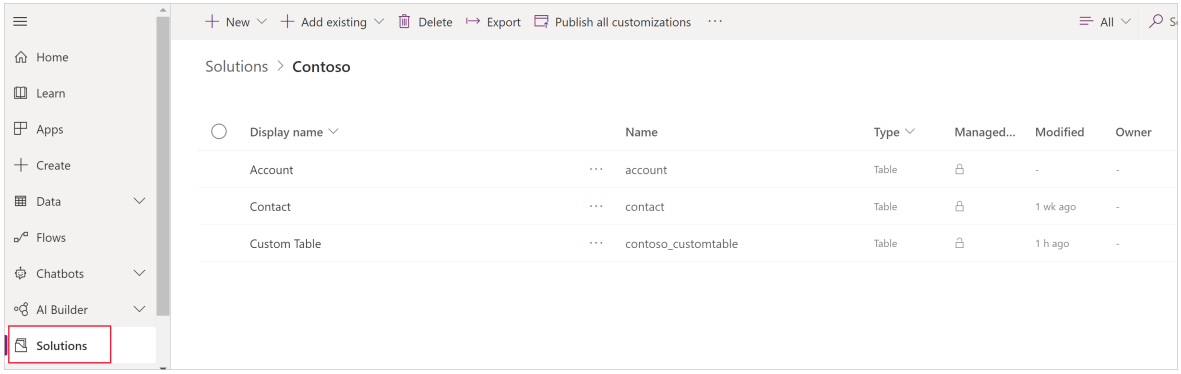
You can delete custom tables, but you can't delete standard tables. Notice that you can't delete a custom table while it has one or more dependent components. More information: Delete a table that has dependencies

> ⚠ **Warning**
>
> When you delete a table, you delete both the table definition and all data that the table contains. Tables and the data within them can't be recovered if deleted.

Delete a custom table

1. Sign into Power Apps, and then select **Solutions** in the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. Open the solution that contains the custom table you want to delete, and then select it.



3. On the command bar select **Remove**, and then select from the following choices:

   - **Remove from this solution**. Removes the table from the solution. You can still access the table from the Default Solution.
   - **Delete from this environment**. Deletes the table and associated data.

4. In the dialog box that appears, select **Delete** to delete the table. If you receive a message that the *component cannot be deleted because it is referenced by other components*, see the next section in this article.

Delete a table that has dependencies

You can't delete a table while it has a dependency on another component, such as a business process flow or a model-driven app.

There are two ways that you can remove a dependency:

- Reconfigure the component to remove the dependency.
- Delete the component.

Remove a dependency

To find the dependencies for a component, see View dependencies for a component.

Next, remove the component. For example, if you have a business process flow that references a table you want to delete, edit the business process flow to remove that table. Once the business process flow no longer references that table, the table will no longer appear in the show dependencies list and can be deleted. More information: Delete a custom table

> **Important**
>
> Removing the dependency, can cause the component to no longer work. For example, removing a table from a model-driven app may cause the model-driven app to no longer work.

For more information, see these articles:

- Edit or remove artifacts
- Edit a business process flow
- Delete relationships

Display custom icons alongside values in list views

06/28/2025

Power Apps environment administrators and customizers can add graphics to a view and establish the logic used to select a graphic based on the column value using JavaScript. The capability lets you customize list views that display icons alongside text or numerical values.

This example displays custom icons in a view for the opportunity table, which is available with certain apps, such as Dynamics 365 Sales. You can display custom icons in views with other standard tables, such as the account or contact table, as well as custom tables.



Some interest in our JJ...	Adventure W...	Won	7/18/2013	$94,800.82	7/21/2013	$95,000.00	Warm	
Test		Open					Warm	
They sell many of the ...	Contoso Phar...	Open	8/2/2013		1/7/2014	$26,000.00	Hot	95
Very interested in our ...	City Power & ...	Won	6/14/2013	$40,201.49	6/29/2013	$40,000.00	Hot	90
Very likely will order 1...	Alpine Ski Ho...	Open	8/2/2013		12/4/2013	$30,000.00	Hot	
Very likely will order 7...	A. Datum Cor...	Won	3/20/2013	$153,385.37	4/1/2013	$16,000.00	Cold	50
Will be ordering abou...	Coho Winery ...	Open	8/2/2013		2/5/2014	$25,000.00	Hot	80

Custom icons in list views can display in model-driven apps, mobile app, and App for Outlook.

> **Note**
>
> - Custom icons aren't available when the app is in mobile offline mode.
> - This customization currently can only be completed by using the classic solution explorer.

Add custom graphics and JavaScript as web resources

1. Create the new graphic files needed for your customization. We recommend an icon size of 16x16 pixels (larger images will be scaled down).

2. Write one or more JavaScript functions that establish which icons to show for which values (you typically need one function for each column you want to customize). Each function must accept a row data object and a language (LCID) code as input and return an array containing an image name and tooltip text. For an example function, see Sample JavaScript function, later in this article.

3. Sign into your environment and open solution explorer.
4. The solution tab opens. Go to **Components** > **Web Resources**.
5. Now, upload your custom graphics, one at a time, as web resources. Select **New** on the toolbar to create a new web resource. A pop-up window opens to help you create the resource. Follow these steps:
    a. Give the new resource a meaningful **Name**. This is the name that you use to refer to each graphic from your JavaScript code.
    b. Set the **Type** to the graphic format you've used to save your graphic file (PNG, JPEG, or GIF).
    c. Select **Choose File** to open a file browser window. Use it to find and select your graphic file.
    d. Add a **Display Name** and/or **Description** if you wish.
    e. Select **Save** and then close the **Web Resource** window.
6. Repeat the previous step for each graphic file that you have.
7. Now, add your JavaScript as the final web resource. Select **New** on the toolbar to create a new web resource. Another pop-up window opens to help you create the resource. Do the following:
    a. Give the new resource a meaningful **Name**.
    b. Set the **Type** to **Script (JScript)**.
    c. Select **Text Editor** (next to the **Type** setting) to open a text-editor window. Paste your JavaScript code here and select **OK** to save it.
    d. Add a **Display Name** and/or **Description** if you wish.
    e. Select **Save** and then close the **Web Resource** window.
8. With the solution tab still open, expand the **Components** > **Tables** tree and locate the table that you want to customize.
9. Expand your table and select its **Views** icon.
10. You now see a list of views for your selected table. Select a view from the list. Then open the **More Actions** drop-down list in the toolbar and select **Edit**.

11. A pop-up window opens with controls for editing your selected view. It shows each column that is part of the view. Select the target column and then select the **Change Properties** in the **Common Tasks** box. The **Change Column Properties** dialog opens; make the following settings here:

    - **Web Resource**: Specify the name of the web resource that you created to hold your JavaScript functions (select **Browse** to choose from a list).
    - **Function Name**: Type the name of the function that you wrote to modify the selected column and view.

12. Select **OK** to close the **Change Column Properties** dialog.
13. Select **Save and Close** to save your view.
14. Repeat these steps for each table, view, and column as needed.
15. When you're ready, select **Publish All Customizations** to publish your changes. Then, close the **Default Solution** window.

Sample JavaScript function

The JavaScript function for displaying custom icons and tooltips expects the following two arguments: the entire row object specified in layoutxml and the calling user's Locale ID (LCID). The LCID parameter enables you to specify tooltip text in multiple languages. For more information about the languages supported by the environment, see Enable languages and Install or upgrade language packs. For a list of locale ID (LCID) values that you can use in your code, see Locale IDs assigned by Microsoft.

Assuming you're adding custom icons for an option-set type of attribute, which has a limited set of predefined options, make sure you use the integer value of the options instead of label to avoid localization issues.

> ⓘ **Note**
>
> If it's necessary to retrieve data to determine the icon, Unified Interface supports returning a JavaScript Promise object, which resolves to a boolean (similar to ribbon rules). Don't use a synchronous `XMLHttpRequest` (XHR) in your custom function.

The sample code displays icons and tooltips based on one of three values (1: Hot, 2: Warm, 3: Cold) in the `opportunityratingcode` (Rating) attribute. The sample code also shows how to display localized tooltip text. For this sample to work, you must create three image web resources with 16x16 images with the following names: new_Hot, new_Warm, and new_Cold.

> **Important**
>
> This sample requires the opportunity table, which is available with Dynamics 365 Sales app.

JavaScript

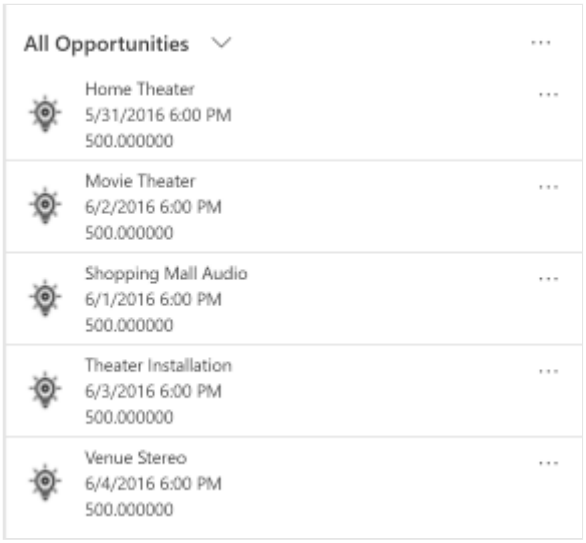
```
function displayIconTooltip(rowData, userLCID) {
    var str = JSON.parse(rowData);
    var coldata = str.opportunityratingcode_Value;
    var imgName = "";
    var tooltip = "";
    switch (parseInt(coldata,10)) {
        case 1:
            imgName = "new_Hot";
            switch (userLCID) {
                case 1036:
                    tooltip = "French: Opportunity is Hot";
                    break;
                default:
                    tooltip = "Opportunity is Hot";
                    break;
            }
            break;
        case 2:
            imgName = "new_Warm";
            switch (userLCID) {
                case 1036:
                    tooltip = "French: Opportunity is Warm";
                    break;
                default:
                    tooltip = "Opportunity is Warm";
                    break;
            }
            break;
        case 3:
            imgName = "new_Cold";
            switch (userLCID) {
                case 1036:
                    tooltip = "French: Opportunity is Cold";
                    break;
                default:
                    tooltip = "Opportunity is Cold";
                    break;
            }
            break;
        default:
            imgName = "";
            tooltip = "";
            break;
    }
    var resultarray = [imgName, tooltip];
```

```
    return resultarray;
}
```

Custom icon view display behavior

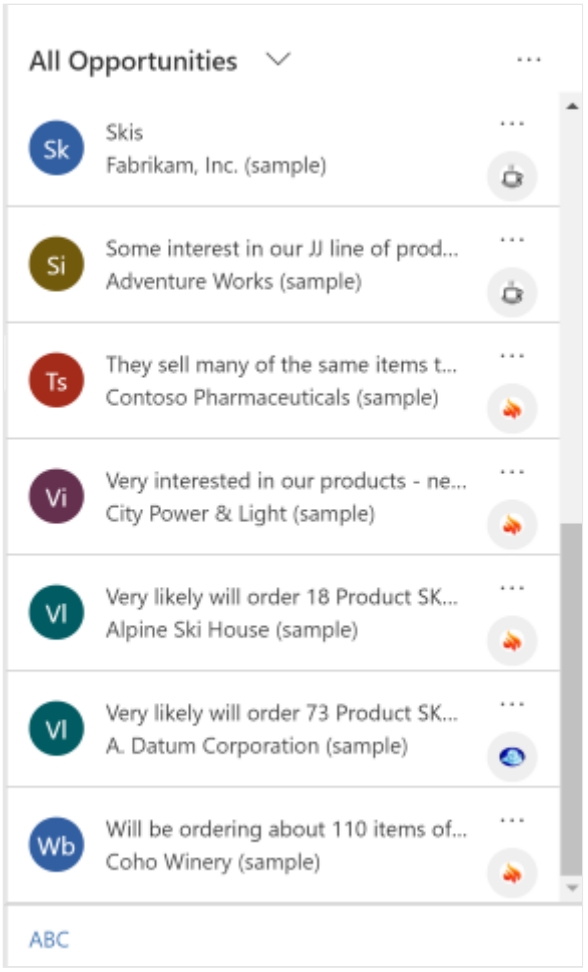
Primary columns

In the grid list view, custom icons applied to the table primary column replace the default system-generated icon.



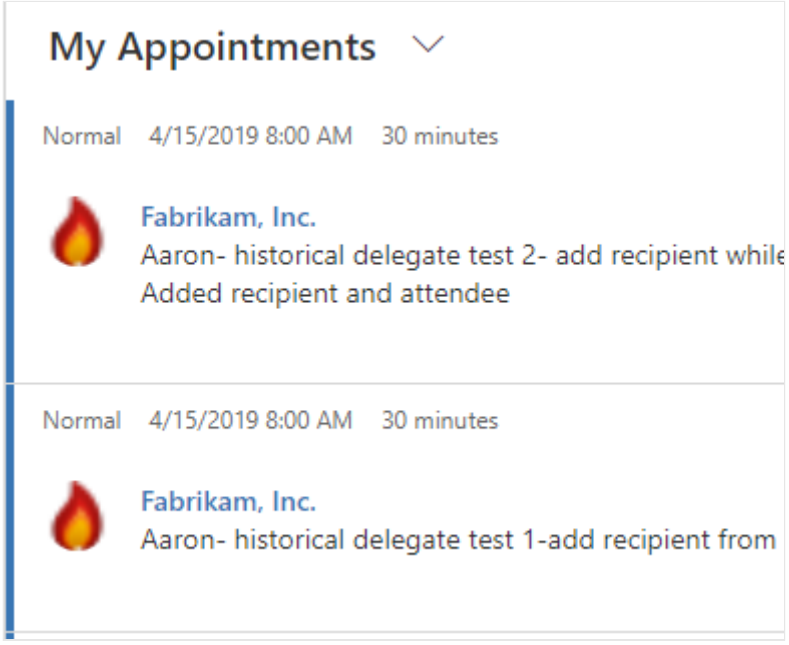
Other columns

In the grid list view, custom icons applied to a column that isn't the table primary column display as a secondary icon in addition to the default system-generated icon.



Card forms

Custom icons replace the default system-generated icon when the view is configured to use a card form.



See also

Understand model-driven app views

Edit system table messages (preview)

Article • 03/13/2023

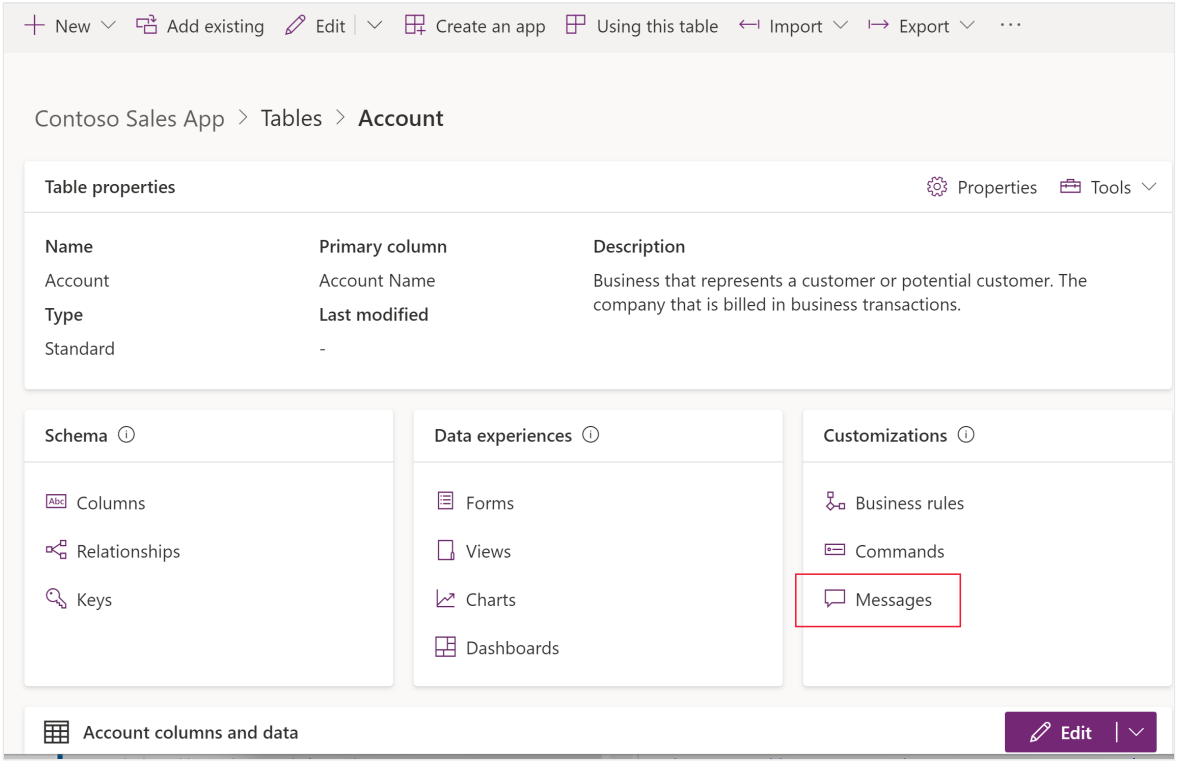
[This topic is pre-release documentation and is subject to change.]

The default display name of some system tables, such as account and contact tables, is used in user interface text and error messages in Microsoft Dataverse. If you change the display name for a system table, you should also update any messages that use the default display name. For example, if you change the display name from *Account* to *Company*, you could still see an error message using the old name.

> **Important**
>
> This is a preview feature. More information: **Model-driven apps and app management**

Add a message to a system table

1. Sign into Power Apps (make.powerapps.com).
2. Go to **Solutions**, open the solution that has the table you want, and then open the table. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. In the **Customizations** area, select **Messages**.

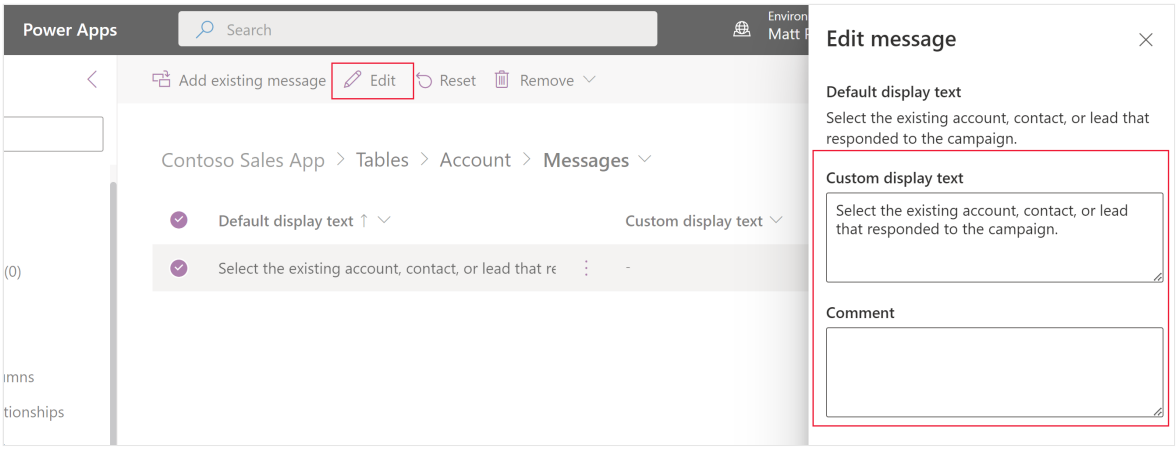


4. Select the message you want, and then select **Add**.

Edit a system table message

1. Open the table that has the message you want to edit, and then in the **Customizations** area select **Messages**.

2. Select the message, and then select **Edit** on the command bar. Enter the changes you want:



Field	Description
Default display text	Shows the original text, which isn't editable.
Custom display text	Edit this text to change the message that is displayed.
Comment	Optional. Include a comment about what you changed and why.

3. Select **Done**.

Programmatically update table display strings

For developers looking for a way to work with these in code, the display strings are stored in the DisplayString table.

The `DisplayString` table doesn't contain the default display strings. The two attributes for this table that contain text are CustomDisplayString and PublishedDisplayString. By default, these attribute values are null unless the display string has been customized and published. The `PublishedDisplayString` value is read-only and reflects the currently published `CustomDisplayString`.

Edit using the classic experience

For information about how to edit system table messages using the classic experience, see Edit system entity messages and display names.

See also

Edit a table

Translate localizable text for model-driven apps

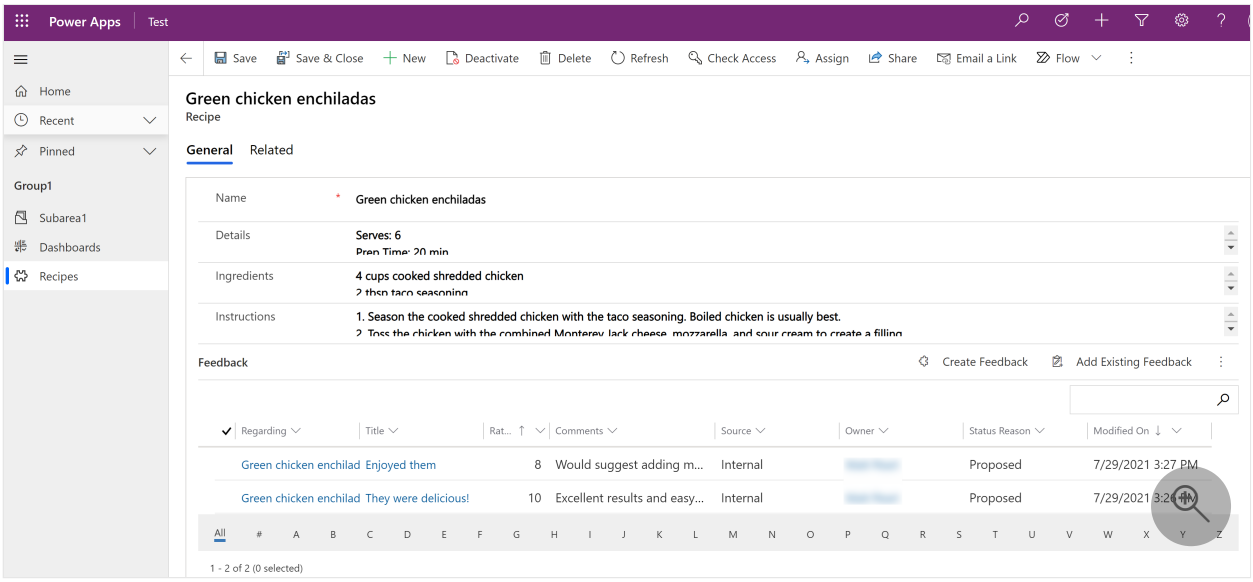
Configure a table for feedback/ratings

Article • 06/22/2022

Let customers or employees submit feedback for any table record, or rate table records within a defined rating range by enabling tables for feedback. Use this capability for the following scenarios:

- For customers to capture data via a portal or survey.
- To collect data about service or product satisfaction from table records.
- For employees to provide comments on collaborative efforts or as a way to improve on products and services.

For example, imagine you have an app that is used to maintain recipes and you want to get feedback from your users about their experience using the recipes in the app.



Configure feedback for a table

First, you must enable the **Allow feedback** property in the table definition.

1. Sign into Power Apps.
2. Open the table you want or create a new one. More information: Edit a table
3. On the right table properties pane, select **Allow feedback**.

> **Note**
>
> Once you enable the **Allow feedback** property for a table, you can't disable it.

Edit table

Display name *

Custom table

Plural display name *

Custom tables

Name *

crfb6_ Customtable

Enable attachments (including notes and files)

Fewer settings

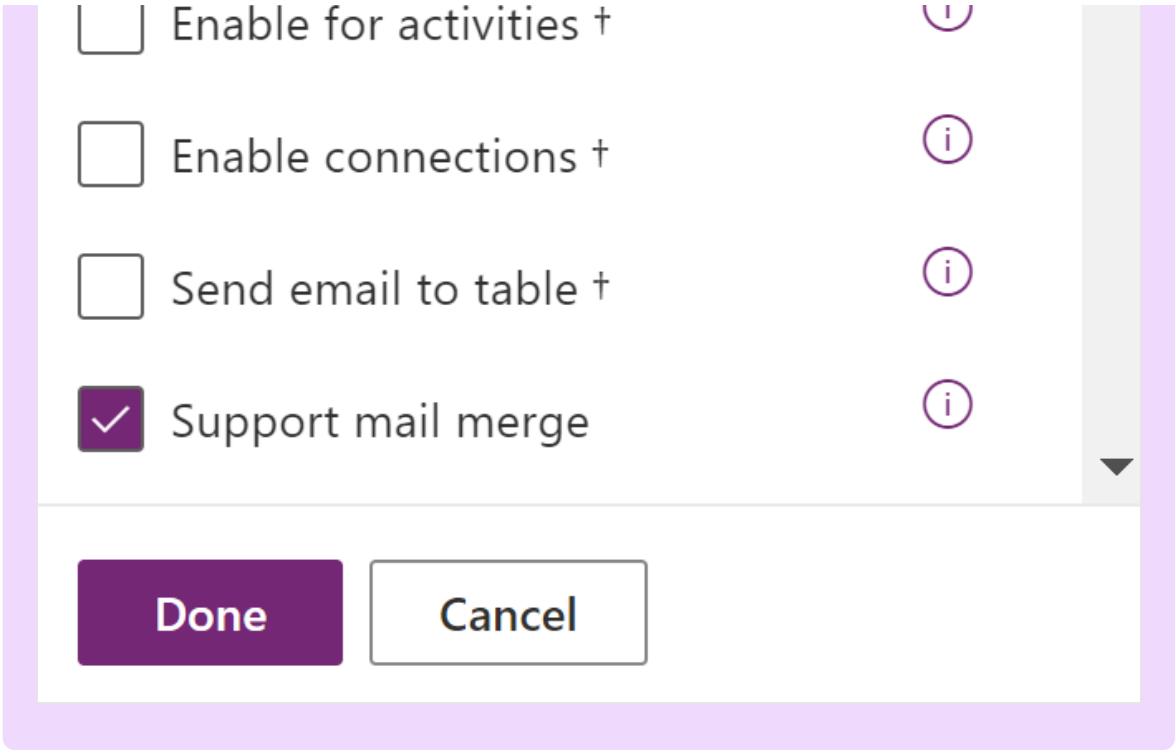
Description

Table type

Collaboration

Enable features to help users to more easily work together on this table.

Allow feedback †



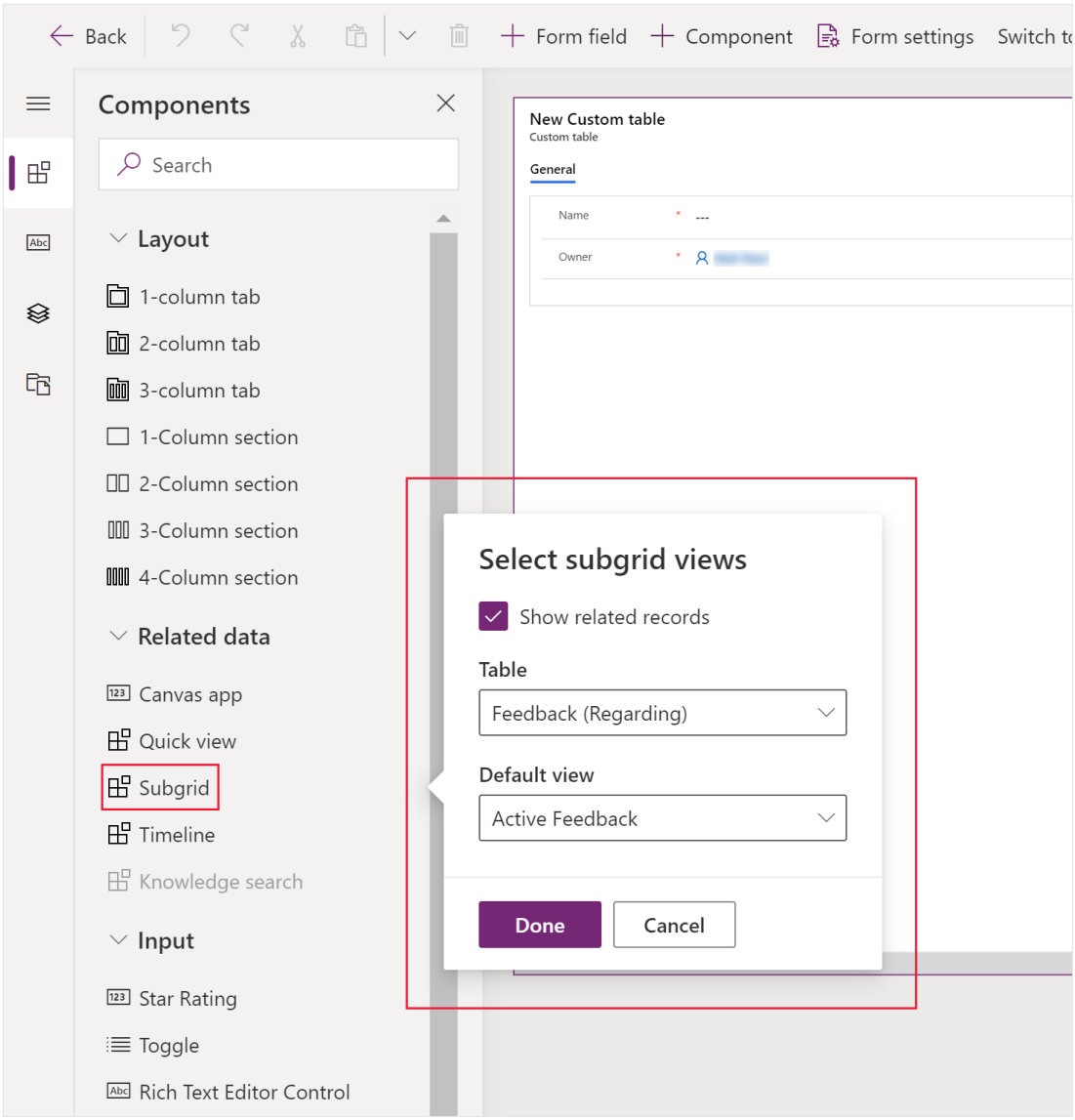
4. Select **Done**, and then select **Save Table**.

Add a subgrid for feedback on the table form

By default, users must go to the list of associated records of the record you want to add feedback to. To make it easier for users to add feedback, you should add a feedback subgrid to the form of the table you are enabling feedback for, such as in the recipe example depicted earlier in this article.

1. In the form designer, select the area of the form where you want the feedback subgrid, and then select **Subgrid** on the left **Components** pane.
2. Select the following subgrid properties, and then select **Done**:

    - **Show related records**
    - **Table Feedback (Regarding)**
    - **Default view**. **Active Feedback**.



3. **Save** and **Publish** the form.

About the feedback table form

The feedback subgrid displays commands for creating new or adding existing feedback. When a user selects **Create Feedback** from the subgrid, the records to create feedback use the feedback standard table.

The feedback table form has the following columns:

- **General**
  - **Title**: User provided title for the record feedback.
  - **Regarding**: System provided table record that the feedback is associated with.
  - **Source**: The feedback source to display where the feedback was submitted. If the feedback was created from within a model-driven app, the default value is set to **Internal** while **Portal** is the other option. This can be changed by the user.
  - **Created By**: System provided value that's the user who created or last modified the feedback record.

- **Comments**
    - **Minimum Rating**: The user can define a range for ratings by specifying a minimum and maximum (numerical) value for ratings. For example, a minimum rating of 1.
    - **Maximum Rating**: The user can define a range for ratings by specifying a minimum and maximum (numerical) value for ratings. For example, a scale of maximum rating of 5.
    - **Normalized Rating**. System generated numeric value that's automatically calculated to show the specified user rating scaled to a value between 0 and 1 based on the minimum and maximum rating values.

      The normalized rating helps to normalize or even out the specified rating value for different rating ranges (minimum and maximum rating values). The normalized rating is calculated as follows: (Rating - Minimum Rating) / (Maximum Rating - Minimum Rating). Also, rating for a record is calculated as an average of all the normalized ratings for the record.
    - **Rating**: User provided rating for the feedback that must be between the minimum and maximum range. For example, a rating of 4 on a scale of 1-5.
    - **Comments**: User provided comments for the record feedback.

Add a rollup column for the feedback

Optionally, you can create a rollup column that's based on the feedback or ratings for the table, and then add it to the form of the table you're enabling for feedback.

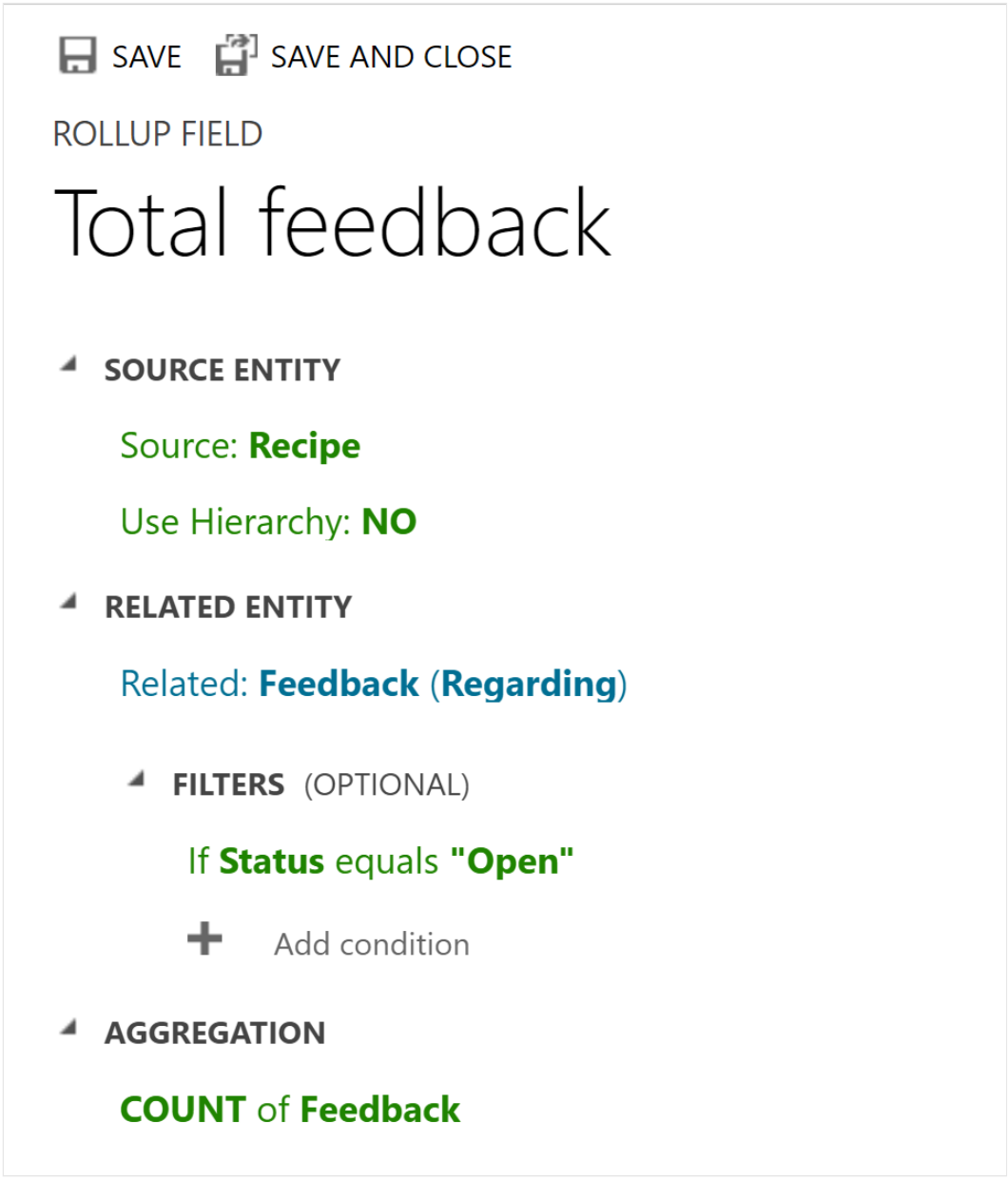
For example, you can provide a running total of all feedback records created for a given record.

1. Open the table that contains the form with the feedback grid and select **Columns**.
2. Select **Add column**, and then select the following properties:

    - **Display name**: *Total feedback*
    - **Data type**: **Whole Number**
    - **Calculated or Rollup**: Select **Add** > **Rollup**. Select **Save** when prompted.

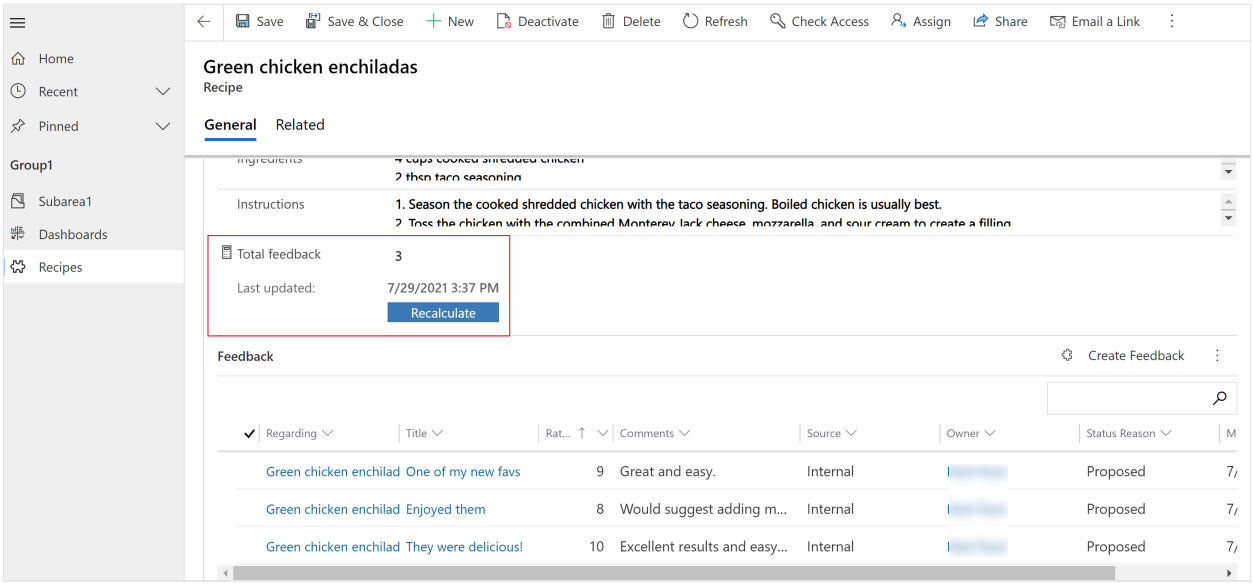
3. Select **Done**. Select the **Total feedback** column, and then in the right column properties pane select **Open rollup**.
4. The rollup editor opens in a new browser window. Configure the following rollup conditions, and then select **Save and Close**:

- **Source entity**:
  - **Source**. Recipe (or the table that has the feedback subgrid)
  - **Use Hierarchy**: **No**
- **Related entity**:
  - **Feedback (Regarding)**
  - **Filters**: **If Status equals Open**
- **Aggregation**:
  - **Count of Feedback**



5. Add the column to the form that has the feedback subrid, save and publish it.

The form in an app now displays a running total of all feedback records submitted for the record.



More information: Define rollup columns that aggregate values

See also

Star rating control

Preview feature: Azure Cosmos DB for NoSQL Data Provider requirements

Article • 06/14/2023

This topic describes the requirements for the Azure Cosmos DB for NoSQL data provider as well as how to configure and recommended best practices when you use the Azure Cosmos DB for NoSQL data provider with virtual tables.

> **Important**
>
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.
> - We expect changes to this feature, so you shouldn't use it in production. Use it only in test and development environments.
> - Microsoft doesn't provide support for this preview feature. Microsoft Dynamics 365 Technical Support won't be able to help you with issues or questions. Preview features aren't meant for production use and are subject to a separate **supplemental terms of use**.

What is Azure Cosmos DB?

Azure Cosmos DB is Microsoft's globally distributed multi-model database service for mission-critical applications. It provides rich and familiar SQL query capabilities with consistent low latencies over schema-less JSON data. More information: Introduction to Azure Cosmos DB: Azure Cosmos DB for NoSQL

Requirements

- Azure subscription that includes Azure Cosmos DB.
- An Azure Cosmos DB for NoSQL collection.
- The Azure Cosmos DB database type should be SQL.

Data type mapping

Suppose you have an Azure Cosmos DB document in a collection named *Orders* that has the following JSON structure.


```
{
"id": "feff22dc-bb3e-4b00-b9c8-4202c408fc31",
"name": "Gorgonzola Telino",
"quantity": 14,
"orderid": "0ffcaff6-390b-4348-a27a-81595130c28a",
"amount": 5932.35,
```





```
"ordertype": 4,
"delivered": 0,
"datetimeoffset": "08/06/2013 9:52:40 PM",
"_rid": "[redacted]",
"_self": "[redacted]",
"_etag": "[redacted]",
"_attachments": "attachments/",
"_ts": 1507750893
}
```


This table indicates the data type mappings for the Azure Cosmos DB for NoSQL document in the *Orders* collection with Microsoft Dataverse.

Azure Cosmos DB for NoSQL Data	**Dataverse**
`id`	Primary Key
`name`	Single Line of Text
`quantity`	Whole Number
`orderid`	Single line of text
`ordertype`	Choice
`amount`	Decimal Number or Currency
`delivered`	Yes/No
`datetimeoffset`	Date and Time

> ⓘ Note
>
> - Attributes with an underscore (_) prefix are generated by the Azure Cosmos DB for NoSQL.
> - Attributes that are configured as optional in the Azure Cosmos DB for NoSQL document and are mapped in Dataverse as **Business Required** will cause a runtime error.
> - id attribute values must be guids.

Supported SQL query filtering

SQL query filtering supports the following operators.

- Comparison operators: `<`, `>`, `<=`, `>=`, `!=`
- Logical operators: `and`, `or`
- Set operators: `in`, `not in`
- String operators: `like`, `contains`, `begins with`, `ends with`

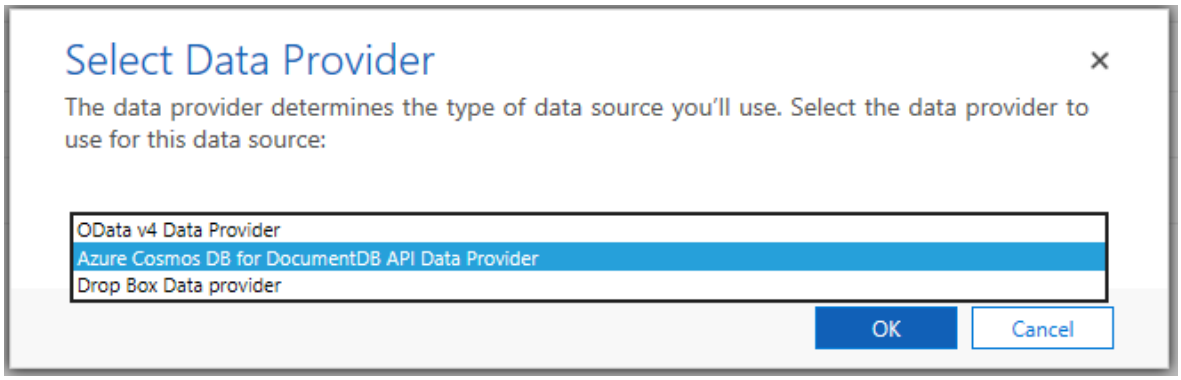
> ⓘ Note
>
> Usage of the like operator is translated to the equivalent `contains/begins with/ends with` operators. The Azure Cosmos DB for NoSQL does not support pattern arguments as described in the topic **Like (Transact-SQL)**. The Azure Cosmos DB for NoSQL Data Provider can translate the single special case `Like('[aA]%')` to `BeginsWith('a')` OR `BeginsWith('A')`. Notice that string comparison in the Azure Cosmos DB for NoSQL is case sensitive.

Add a data source using the Azure Cosmos DB for NoSQL Data Provider

1. Go to AppSource, select **GET IT NOW**, and follow the instructions to add the application to your environment using v9x or later.

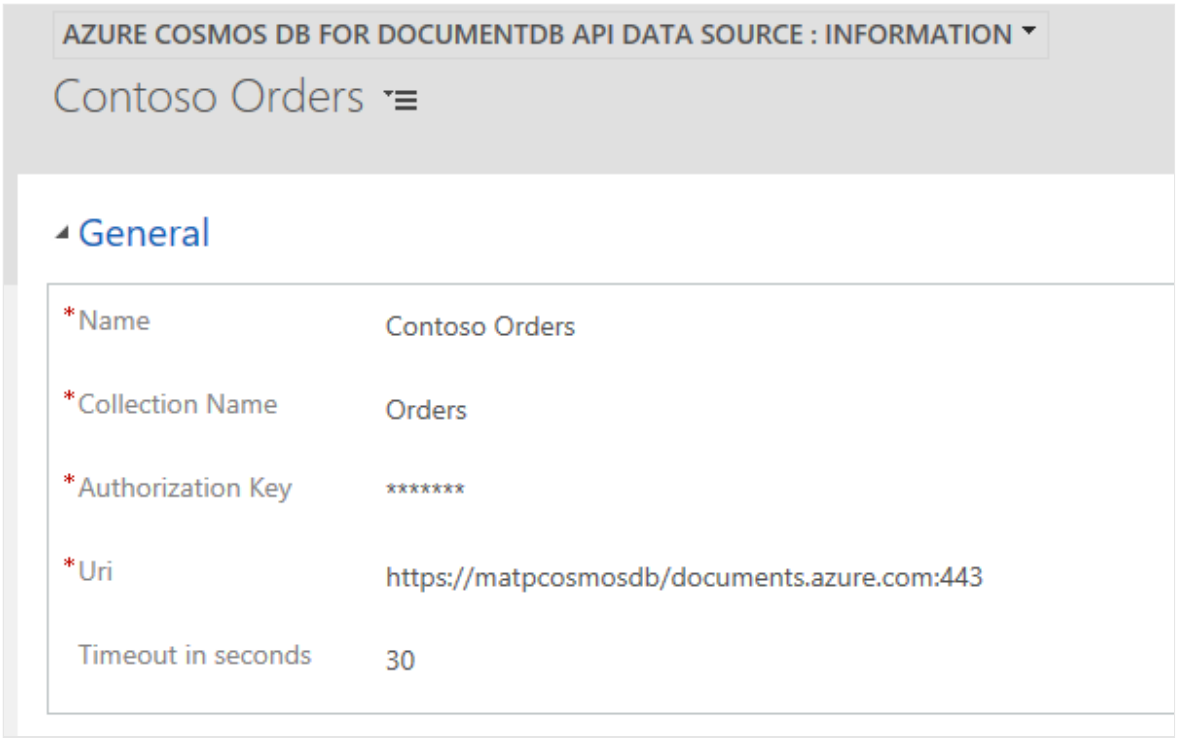
2. After the solution is installed, sign in to the environment and go to **Settings** > **Administration** > **Virtual Entity Data Sources**.

3. On the Actions toolbar select **NEW**, and in the **Select Data Provider** dialog box select **Azure Cosmos DB for SQL API Data Provider**, and then select **OK**.



4. Enter the following information, and then select **SAVE & CLOSE**.

Column	Description
Name	Type a name that describes the data source.
Collection Name	The name of the Azure Cosmos DB *database* containing the collection you want to surface in a virtual table.
Authorization Key	The primary or secondary key for the Azure Cosmos DB account. You can find the key from the Azure admin portal under the **Keys** setting under your Azure Cosmos DB account.
Uri	The URI of the resource group where the Azure Cosmos DB collection is located. The URI is formed similar to `https://contoso/documents.azure.com:443`. You can find the URI from the Azure admin portal under the **Keys** setting for the Azure Cosmos DB account.
Timeout in seconds	Enter the number of seconds to wait for a response from the Azure Cosmos DB service before a data request time-out. For example, enter 30 to wait a maximum of thirty seconds before a time-out occurs. The default timeout is 120 seconds.



Best practices and limitations

- Notice the following when you use Azure Cosmos DB as a Data Source:
  - Each Azure Cosmos DB Data Source can only be associated with a single virtual table.
  - You can connect multiple Data Sources to the same Collection in the Azure Cosmos DB instance.
- You can't segment the data in a collection by table.
- Azure Cosmos DB databases do not require a schema, however the data within the Azure Cosmos DB must be structured using a predictable schema.
- Although the Azure Cosmos DB for NoSQL Data Provider implements query translation of projection, filtering, and sorting operators, it does not support join operations.
- You can only filter by a single column with Azure Cosmos DB for NoSQL.

See also

Create and edit virtual tables that contain data from an external data source

Virtual table walkthrough using the OData v4 Data Provider

Article • 02/22/2023

Imagine that you want to access, create, update, and delete some personal information about a contact from an external data source within your model-driven app. In this simple walkthrough, you will model a virtual table named *Person* with columns mapped to the external schema to these operations at runtime from an OData web service.

Data source details

Because the data source used for this walkthrough has an OData v4 web service, we can use the OData v4 Data Provider included with your environment.

Web service url: `https://contosowebservice.azurewebsites.net/odata/`

> **ⓘ Important**
>
> - The web service url used for this walkthrough isn't a functioning web service.
> - You can't use the OData v4 Data Provider to connect to another environment.

For this walkthrough, a single virtual table that contains the following columns is needed.

External column name	External data type	Virtual table data type	Purpose
Id	`Edm.Guid`	Primary key	Primary key for the table
FirstName	`Edm.String`	Single line of text	First name of the person
LastName	`Edm.String`	Single line of text	Last name of the person
Age	`Edm.Int32`	Whole number	Age of the person
LastModifiedOn	`Edm.DateTimeOffset`	Data and time	Last modified data and time of the person record

The OData metadata of the external data source person table:



Create the data source

Create the data source for the OData v4 data provider that uses the OASIS Open Data Protocol (OData) sample web service.

1. Go to **Settings** > **Administration** > **Virtual Entity Data Sources**.
2. Select **NEW**, select **OData v4 Data Provider**, and then select **OK**.
3. Enter or select the following information.

Field	Value
Name	Person Sample OData Data Source
URL	`https://contosowebservice.azurewebsites.net/odata`
Timeout	30
Return Inline Count	Client-side Paging
Return Inline Count	True

Leave the other columns as-is, and select **SAVE & CLOSE**.

> **Tip**
>
> When using your own web service, verify that the URL is valid by pasting it in to your web browser. If you want to understand the columns and data types respective to the table you are interested in, you can append `/$metadata` to the end of your URL to see the definition of tables in you web service.

Open solution explorer

Part of the name of any custom table you create is the customization prefix. This is set based on the solution publisher for the solution you're working in. Make sure that you are working in an unmanaged solution where the customization prefix is the one you want for this table. More information: Change the solution publisher prefix

Open an unmanaged solution

1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
2. In the **All Solutions** list open the unmanaged solution you want.

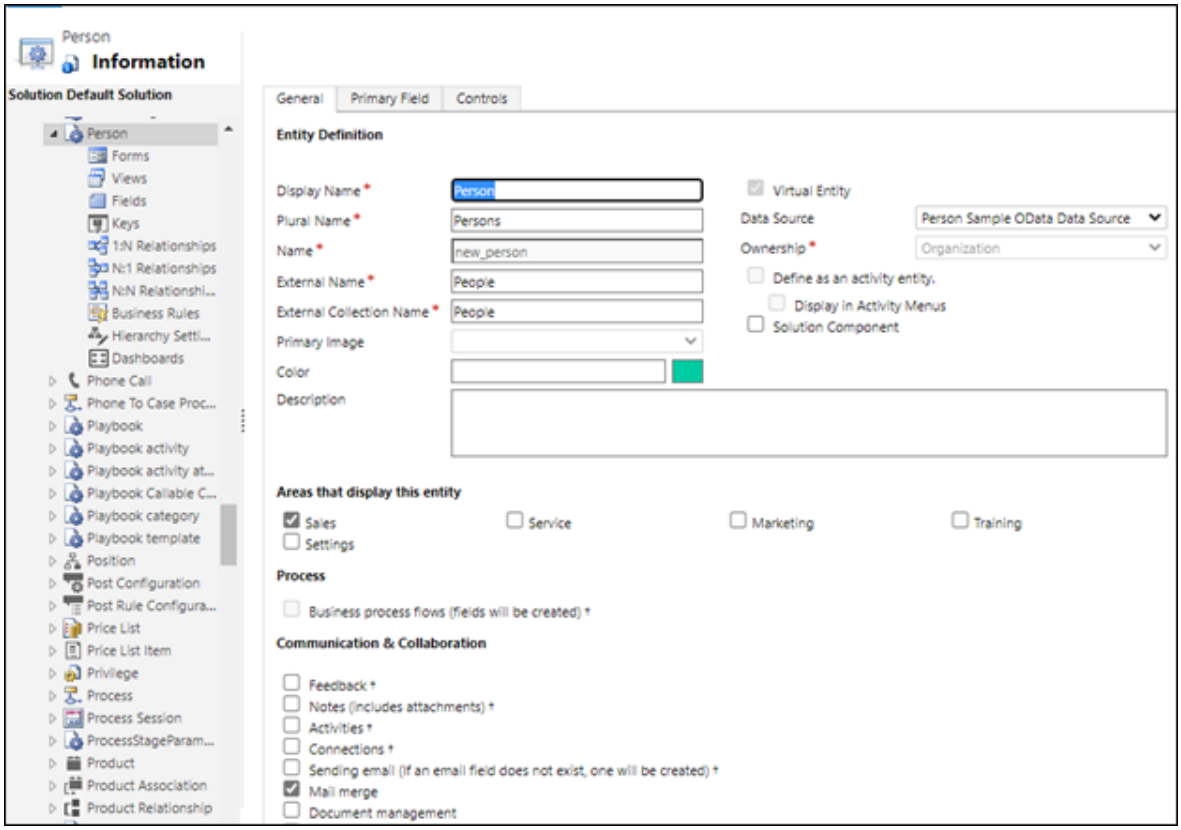
Create the virtual table

1. In the left navigation pane of solution explorer, select **Entities**, and then select **New** from the main pane.
2. On the **Entity: New** form, select the **Virtual table** option, and then enter the following information:

Field	Value
Data Source	Person Sample OData Data Source
Display Name	Person
Plural Name	Persons
Name	new_person
External Name	People
External Collection Name	People
Notes (includes attachments)	selected
Activities	selected

> **Note**
>
> The OData resource path semantics uses `EntitySet` to identify the resource. In the sample OData web service, the `EntitySet` is defined as **People**. This value is used as the **External Name** for the virtual table **Person**.

3. Next to **Areas that display this entity**, select **Sales**, and then select **Save** (but don't close the table form).



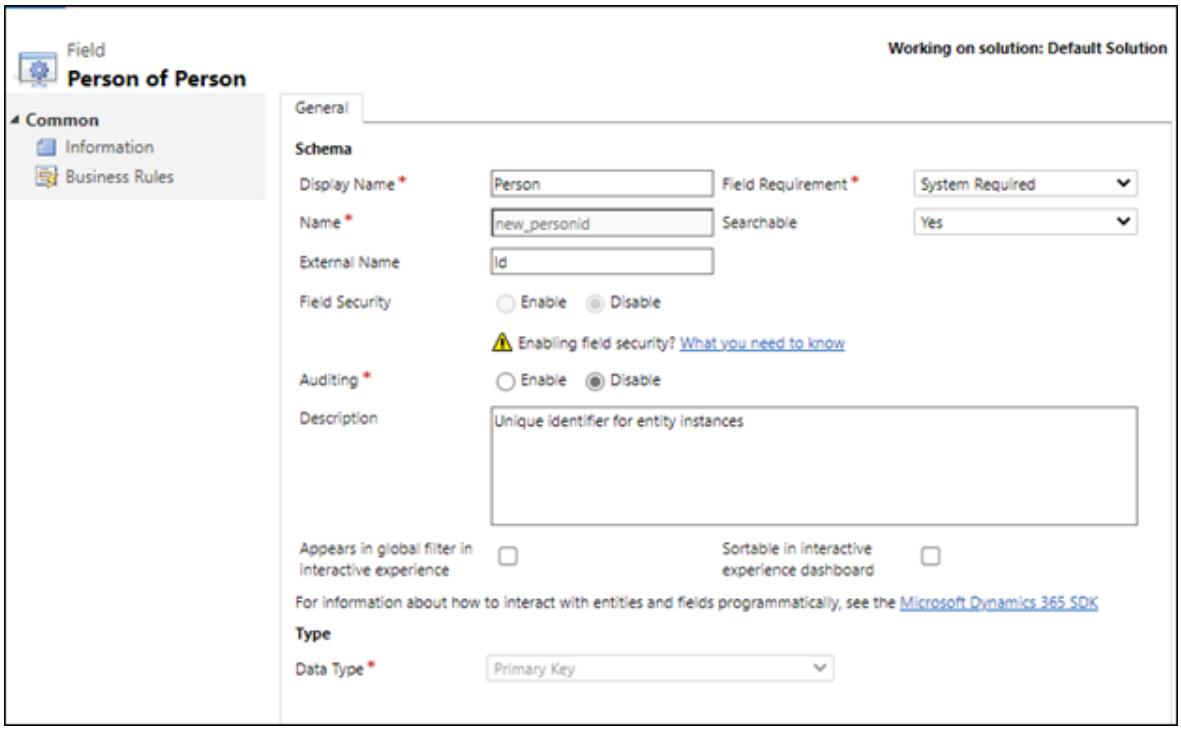
Create the columns for the virtual table

On the left navigation pane of the **Entity: Person** page, select **Fields**. As part of this walkthrough, you will edit two existing columns and add three more columns that are available in the external data source.

> **ⓘ Important**
>
> External names are case sensitive. Refer to your web service metadata to make sure you use the correct name. A Nullable value of false indicates that the attribute is required. Notice that primary key columns are always system required.

1. Open the **new_personid** column, and change the following property with the value indicated here:

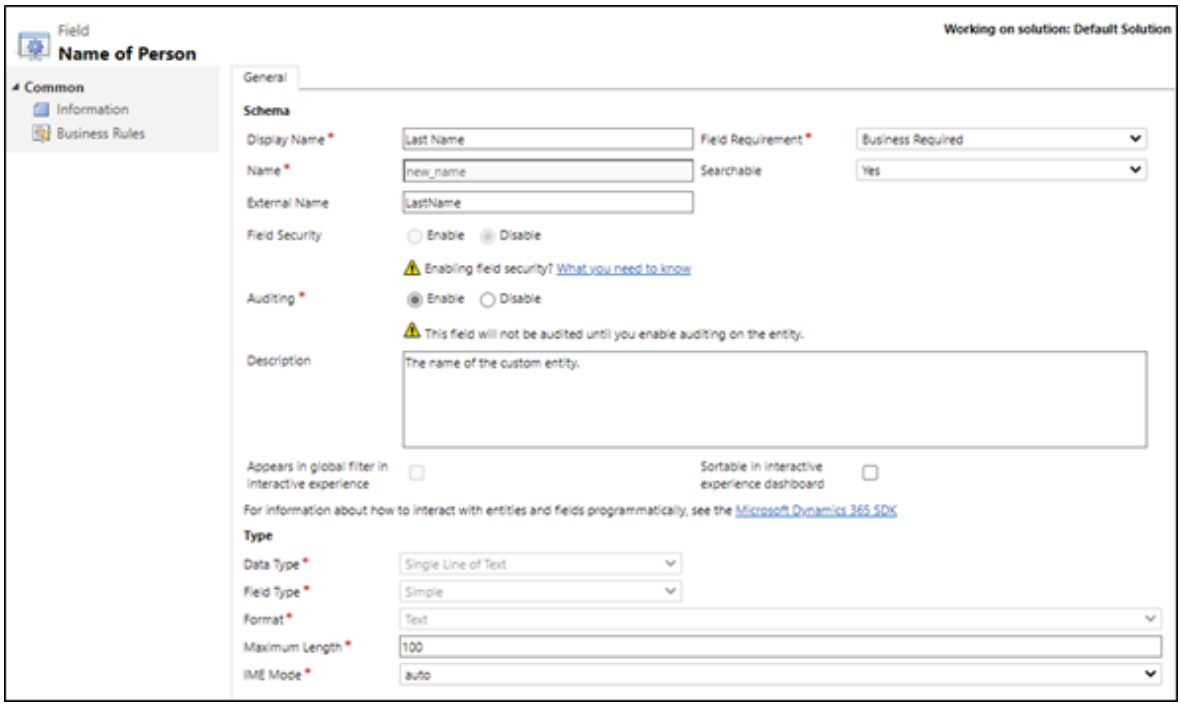
   External Name: Id



2. Select **Save and Close**.

3. Open the **new_name** column, and change the following properties to have the values indicated here:

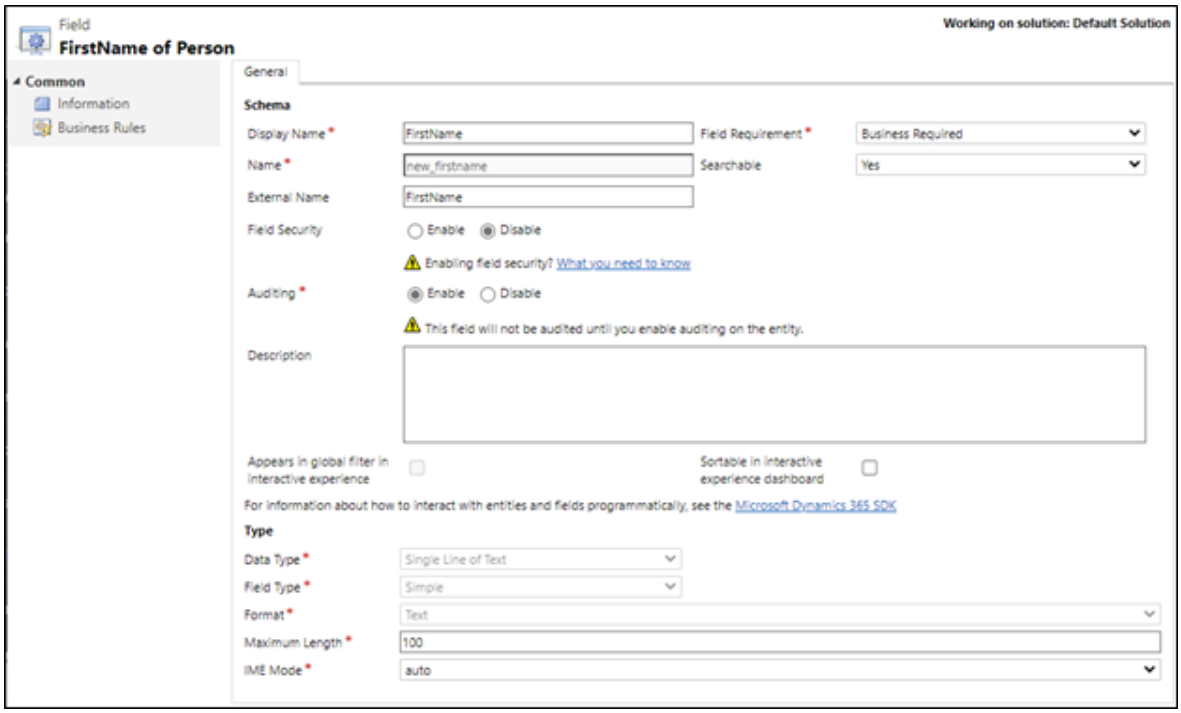
    - **Display Name**: Last Name
    - **External Name**: LastName



4. Select **Save and Close**.

5. Select **New**, and on the **Field: New for Person** page enter the following information:

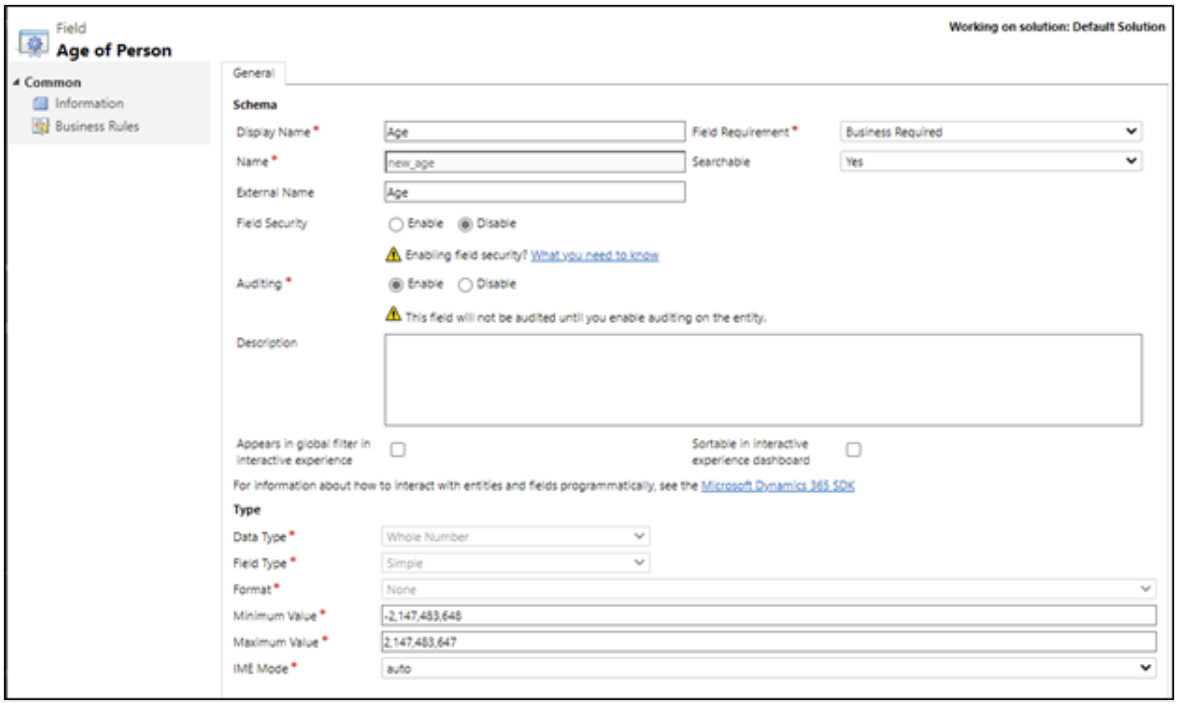
Field	Value
Display Name	First Name
Name	new_firstname
External Name	FirstName
Field Requirement	Business Required
Data Type	Single Line of Text



6. Select **Save and Close**.

7. Select **New**, and on the **Field: New for Person** page enter the following information:

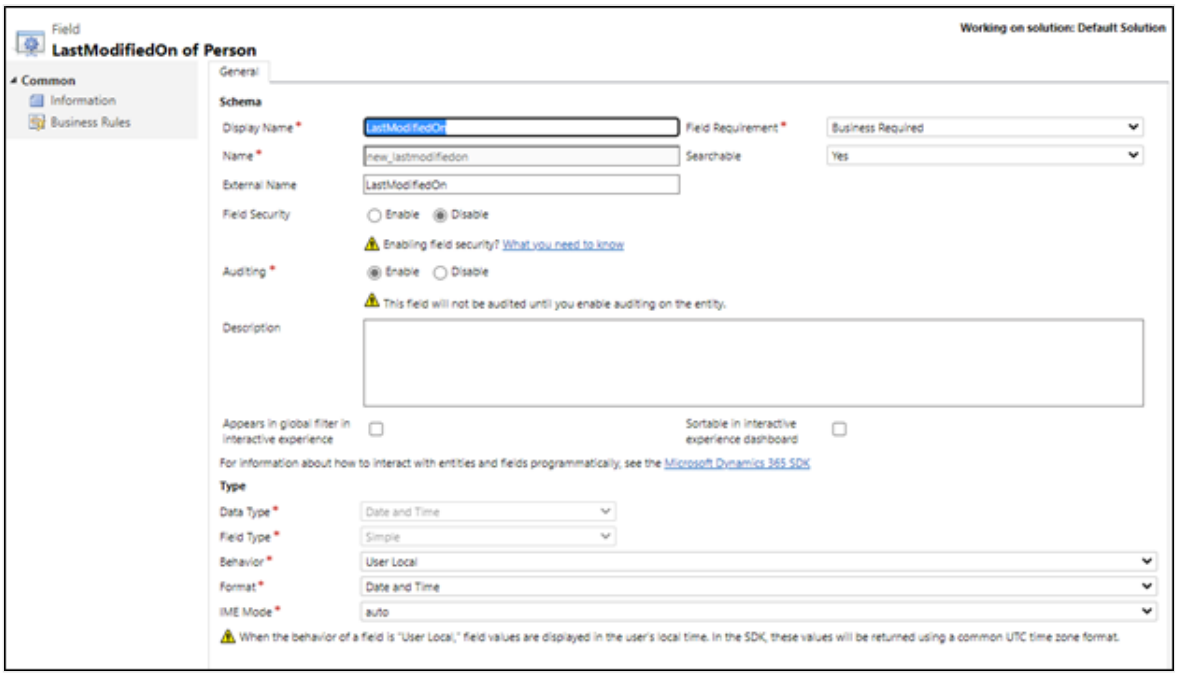
Field	Value
Display Name	Age
Name	new_age
External Name	Age
Field Requirement	Business Required
Data Type	Whole Number



8. Select **Save and Close**.

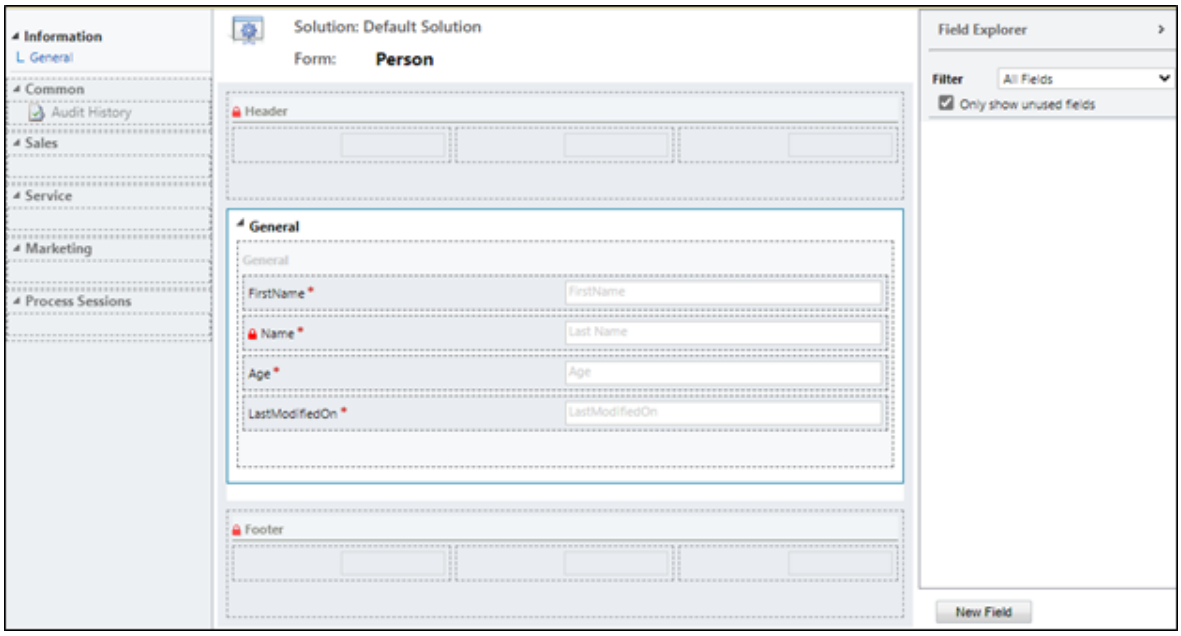
9. Select **New**, and on the **Field: New of Person** page enter the following information:

Field	Value
Display Name	Last Modified On
Name	new_lastmodifiedon
External Name	LastModifiedOn
Field Requirement	Business Required
Data Type	Date and Time



Add the columns to the main form

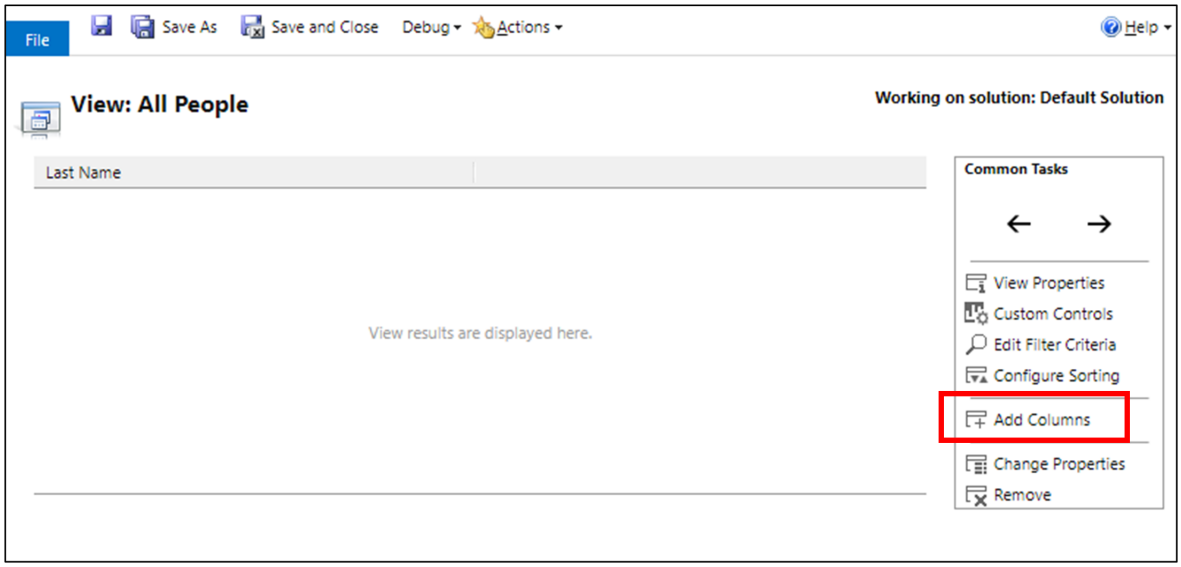
1. On the **Person** table window, select **Forms**.
2. Open the main form, drag and drop the **First Name**, **Age**, and **Last Modified On** columns from the right pane onto the form in the **General** section.



3. On the **Person** table window, select **Save and Close**.

Configure the default view

1. On the left pane of the solution explorer, under the **Person entity**, select **Views**.
2. Open the **All People** view.
3. In the **Common Tasks** pane, select **Add Columns**.



4. Select the **First Name**, **Age**, and **Last Modified On** columns, and then select **OK**.
5. On the **View: All People** window, arrange the column order by selecting the move left and move right arrows, and then select **Save and Close**.

6. On the Solution Explorer window, select **Publish All Customizations**.
7. After all customizations are published, close the Solution Explorer window.

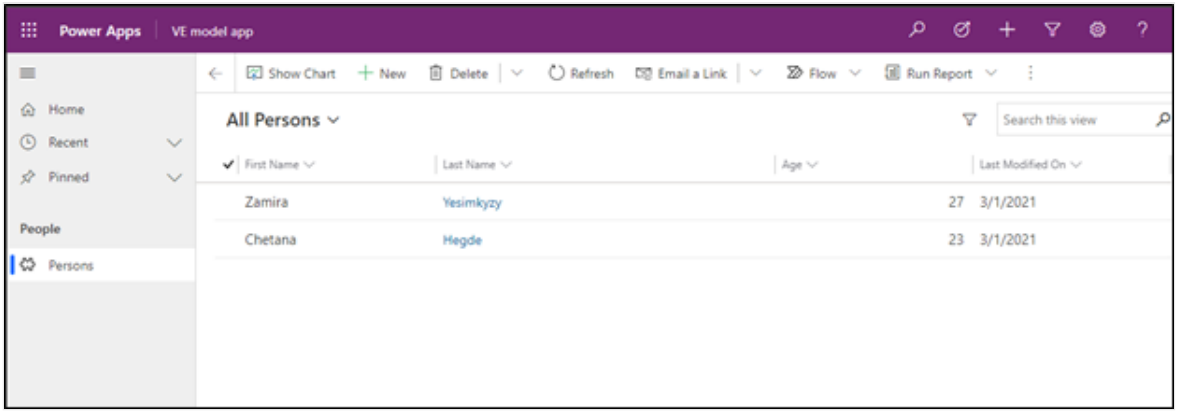
Use the virtual table in a model-driven app

Create a model-driven app and add the **Person** table to the site map. Then, select the **Person** main form and the **Person Advance Find** view. Publish the app. More information: Build your first model-driven app from scratch

Then, app users can perform read, create, update, and delete operations using the virtual table just like any other table in Microsoft Dataverse.

Create, view, and delete records using the virtual table

1. Run the model-driven app. Select **New**, complete the required columns, and then save the record.
2. Select **New** again to create another record and save it.
3. You should now have two records similar to the following in the **All Persons** view.



4. To delete a record, select a person record and then select **Delete**. Confirm to delete the record on the confirmation page.

See also

OData v4 Data Provider configuration, requirements, and best practices
Create and edit virtual tables that contain data from an external data source

Create and edit tables using solution explorer

Article • 02/15/2022

You can easily create a table using Power Apps (make.powerapps.com) for most common situations, but not all capabilities are implemented there. When you need to meet the requirements described in Create and edit tables in Microsoft Dataverse, you can achieve them by creating or editing tables using the Solution Explorer.

Open solution explorer

The customization prefix is part of the name of any table you create. This is set based on the solution publisher for the solution you're working in. If you care about the customization prefix, make sure that you are working in an unmanaged solution where the customization prefix is the one you want for this table. More information: Change the solution publisher prefix

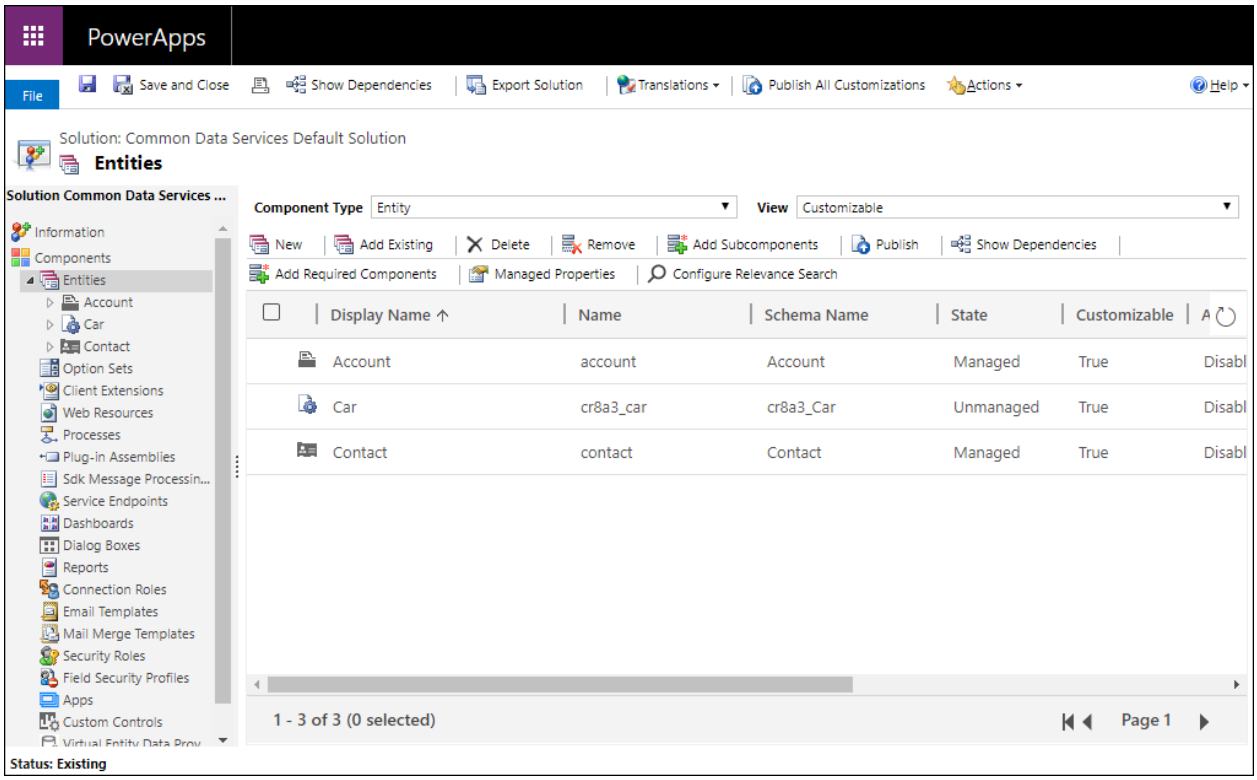
Open an unmanaged solution

1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. In the **All Solutions** list open the unmanaged solution you want.

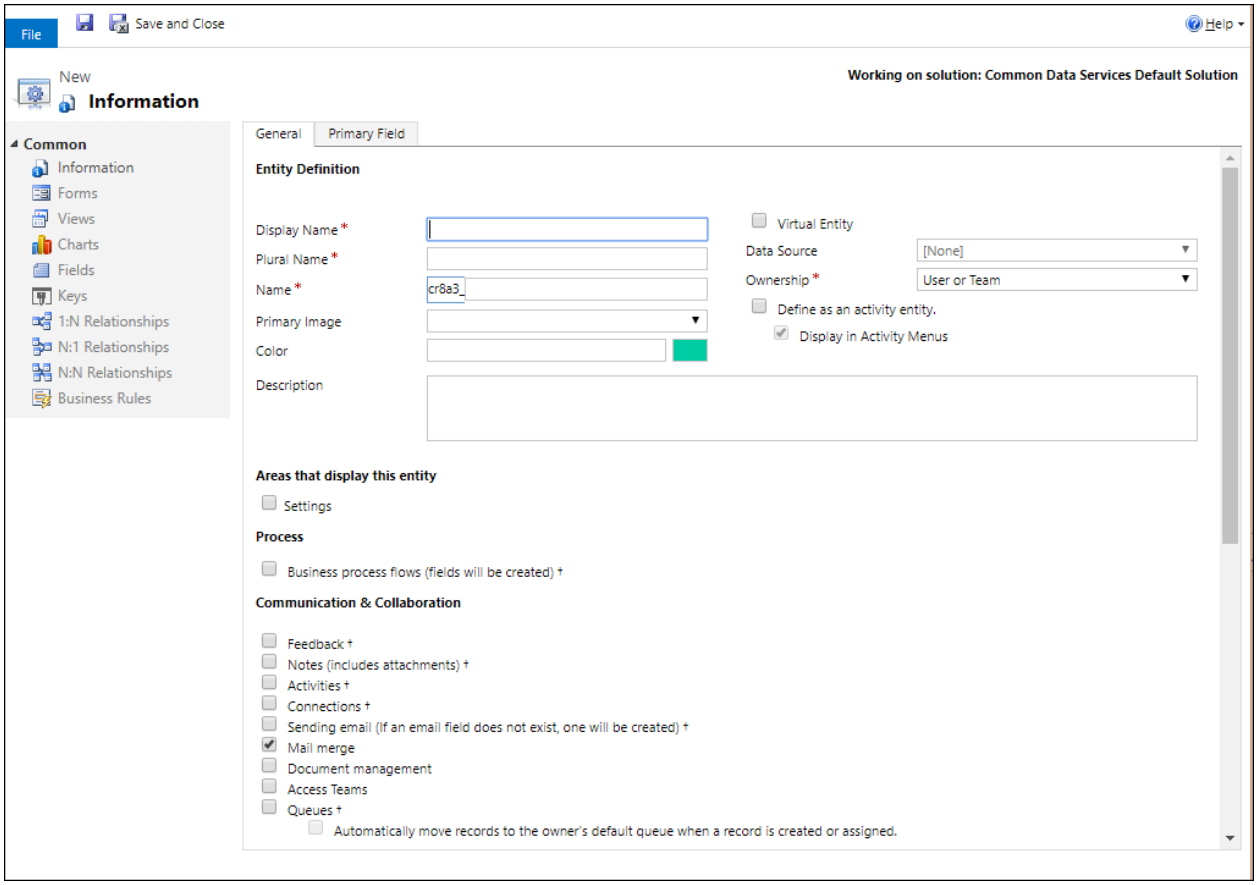
View tables

In the solution explorer **Components** node, select the **Entities** node.



Create a table

While viewing tables, select **New** to open the form to create tables.



The form has two tabs. The **General** tab is for entity options. The **Primary Field** tab is for options about the special single line of text column that each table has that defines the

text shown when there is a link to open the table in a lookup column.

For information about each section see the following:

- Configure the primary column
- Configure required columns

> ⓘ Note
>
> You can also make the table a custom activity. This choice changes some of the default option values. More information: **Create a custom activity table**

After you have set the required options for the table, click  to create the custom table.

Configure the primary column

In the **Primary Field** tab you can usually accept the default values for the primary column, but you have the following options:

Field	Description
Display Name	Enter the localizable label that will be displayed for this column in forms and lists. The default value is **Name**.
Name	Set the name used in the system for this column. The default value is `<customization prefix>_name`
Maximum Length	Enter the maximum length for the column values. The default is 100.

> ⓘ Note
>
> These options do not apply if the table is an activity table. More information: **Create a custom activity table**

Configure required columns

In the **General** tab, some of the options are required before you can save the table.

Field	Description
Display Name	This is the singular name for the table that will be shown in the app. This can be changed later.

Field	Description
Plural Name	This is the plural name for the table that will be shown in the app. This can be changed later.
Name	This column is pre-populated based on the display name you enter. It includes the solution publisher customization prefix.
Ownership	You can choose either user or team-owned or organization owned. More information: Table ownership

Edit a table

While viewing tables, select the table you want to edit, or continue editing a new table you have just saved.

> **Note**
>
> Standard tables or custom tables that are part of a managed solution may have limitations on changes you can apply. If the option is not available or is disabled, you are not allowed to make the change.

Set once options

The following options can be set once and cannot be changed after you set them. Take care to only set these options when you need them.

Option	Description
Activities	Associate activities to records for this table.
Business process flows	Create business process flows for this table. More information: Create a business process flow to standardize processes
Connections	Use the connections feature to show how records for this table have connections to records of other tables that also have connections enabled.
Feedback	Let customers write feedback for any table record, or rate table records within a defined rating range. More information: Enable a table for feedback/ratings
Notes	Append notes to records for this table. Notes include the ability to add attachments.

Option	Description
Queues	Use the table with queues. Queues improve routing and sharing of work by making records for this table available in a central place that everyone can access.
Sending email	Send emails using an email address stored in one of the fields for this table. If a **Single Line of Text** field with format set to **email** doesn't already exist for this table, a new one will be created when you enable sending email.

Options that you can change

The following properties can be changed at any time.

Option	Description
Access Teams	Create team templates for this table.
Allow quick create	After you have created and published a **Quick Create Form** for this table, people will have the option to create a new record using the **Create** button in the navigation pane. More information: Create and design forms When this is enabled for a custom activity table, the custom activity will be visible in the group of activity entities when people use the **Create** button in the navigation pane. However, because activities don't support quick create forms, the main form will be used when the custom table icon is clicked.
Areas that display this entity	In the web application choose one of the available sitemap areas to display this table. This does not apply to model-driven apps.
Auditing	When auditing is enabled for your organization, this allows for changes to table records to be captured over time. When you enable auditing for an table, auditing is also enabled on all its fields. You can select or clear fields that you want to enable auditing on.
Change Tracking	Enables data synchronization in a performant way by detecting what data has changed since the data was initially extracted or last synchronized.
Color	Set a color to be used for the table in model-driven apps.
Description	Provide a meaningful description of the purpose of the table.
Document management	After other tasks have been performed to enable document management for your organization, enabling this feature allows for this table to participate in integration with SharePoint.
Duplicate Detection	If duplicate detection is enabled for your organization, enabling this allows you to create duplicate detection rules for this table.

Option	Description
Enable for mobile	Make this table available to the Dynamics 365 for phones and tablets apps. You also have the option to make this table **Read-only in mobile**. If the forms for an table require an extension not supported in Dynamics 365 for phones and tablets apps use this setting to ensure that mobile app users can't edit the data for these entities.
Enable for phone express	Make this table available to the Dynamics 365 for phones app.
Mail merge	People can use this table with mail merge.
Offline capability for Dynamics 365 for Outlook	Whether data in this table will be available while the Dynamics 365 for Outlook application is not connected to the network.
Primary Image	System entities that support images will already have an **Image** field. You can choose whether to display data in this field as the image for the record by setting this field to **[None]** or **Default Image**. For custom entities you must first create an image field. Each table can have only one image field. After you create one, you can change this setting to set the primary image. More information: Image columns
Reading pane in Dynamics 365 for Outlook	Whether the table will be visible in the reading pane for the Dynamics 365 for Outlook app.
Use custom Help	When enabled, set a Help URL to control what page users will see when they click the help button in the application. Use this to provide guidance specific to your company processes for the table.

You can also make the following changes:

- Create and edit columns for Dataverse
- Create and edit relationships between tables
- Create and design forms
- Create a business process flow to standardize processes

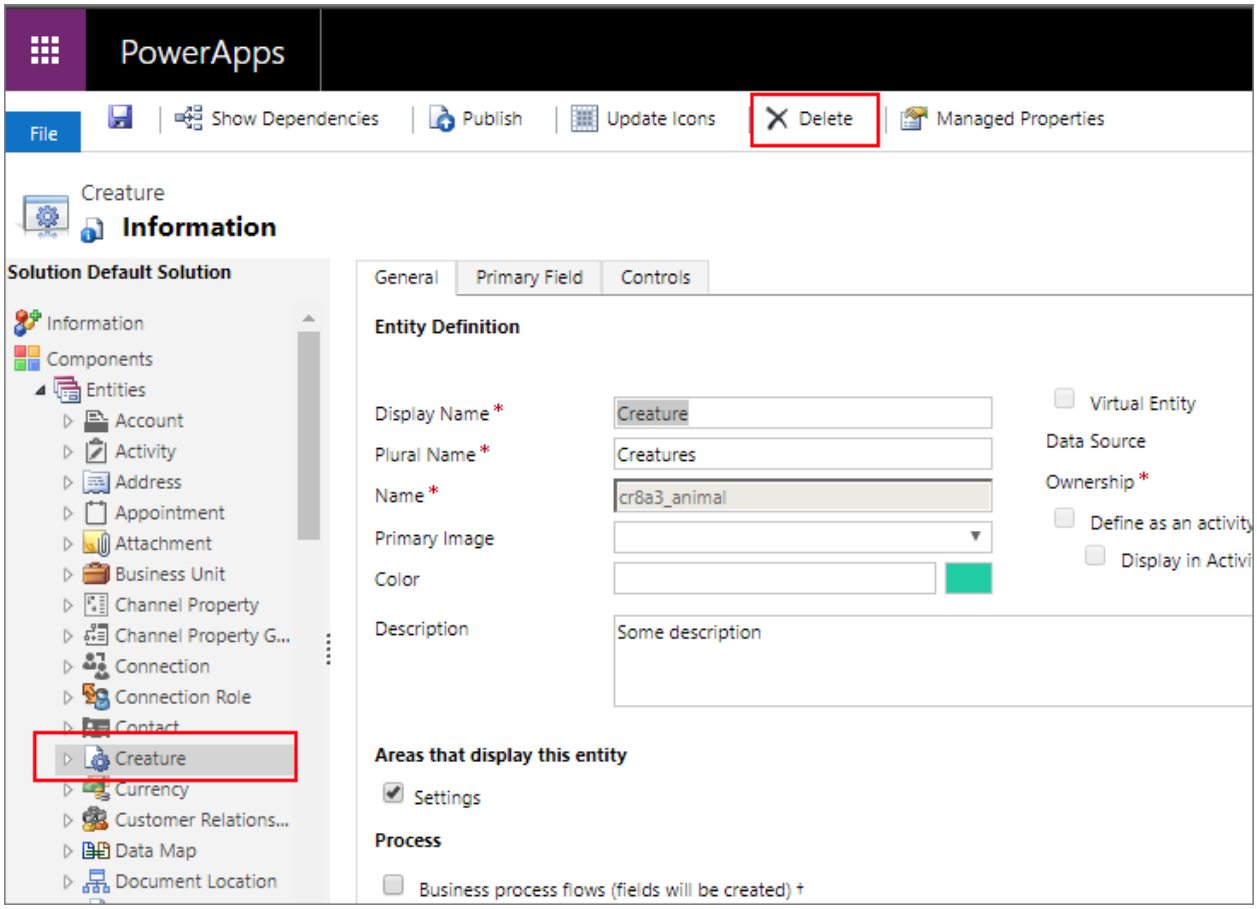
Delete a table

As someone with the system administrator security role, you can delete custom tables that aren't part of a managed solution.

> **ⓘ Important**
>
> - When you delete a custom table, the database tables that store data for that table are deleted and all data they contain is lost. Any associated rows that have a parental relationship to the custom table are also deleted. For more information about parental relationships, see **Create and edit relationships between tables**.
> - The only way to recover data from a table that was deleted is to restore the database from a point before the table was deleted. More information: **Backup and restore environments**

While viewing tables, select the ✕ command on the toolbar.

While viewing a table use the delete command on the menu bar.



> **⚠ Warning**

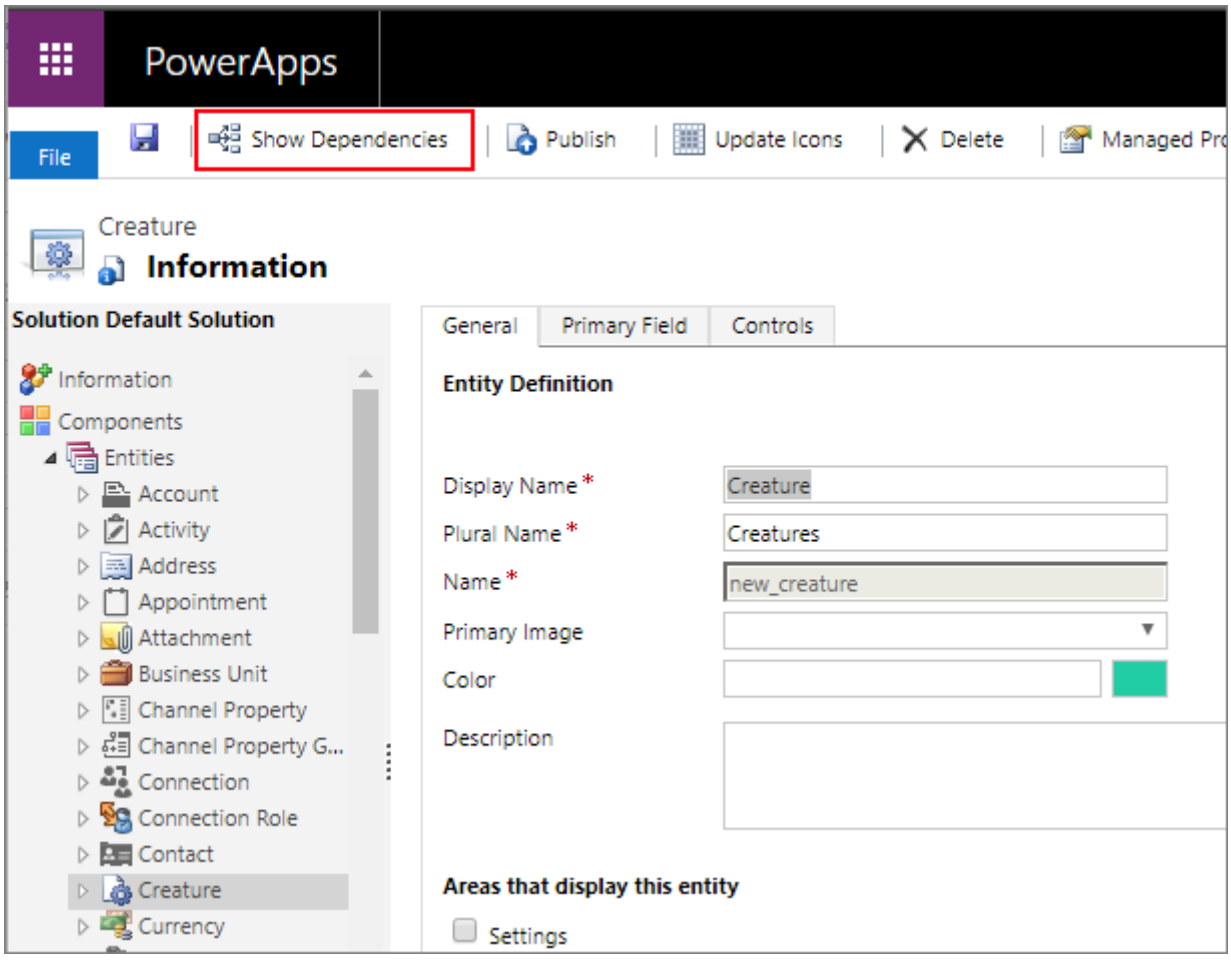
Deleting a table that contains data will remove all the data. This data can only be retrieved by backup of the database.

> **Note**
>
> If there are any table dependencies you will get a **Cannot Delete Component** error with a **Details** link you can use to discover information about why the table cannot be deleted. In most cases, this will be because of a dependency that has to be removed.
>
> There may be more than one dependency blocking the deletion of a table. This error message may only show the first one. For an alternate way to discover dependencies, see **Identify table dependencies**

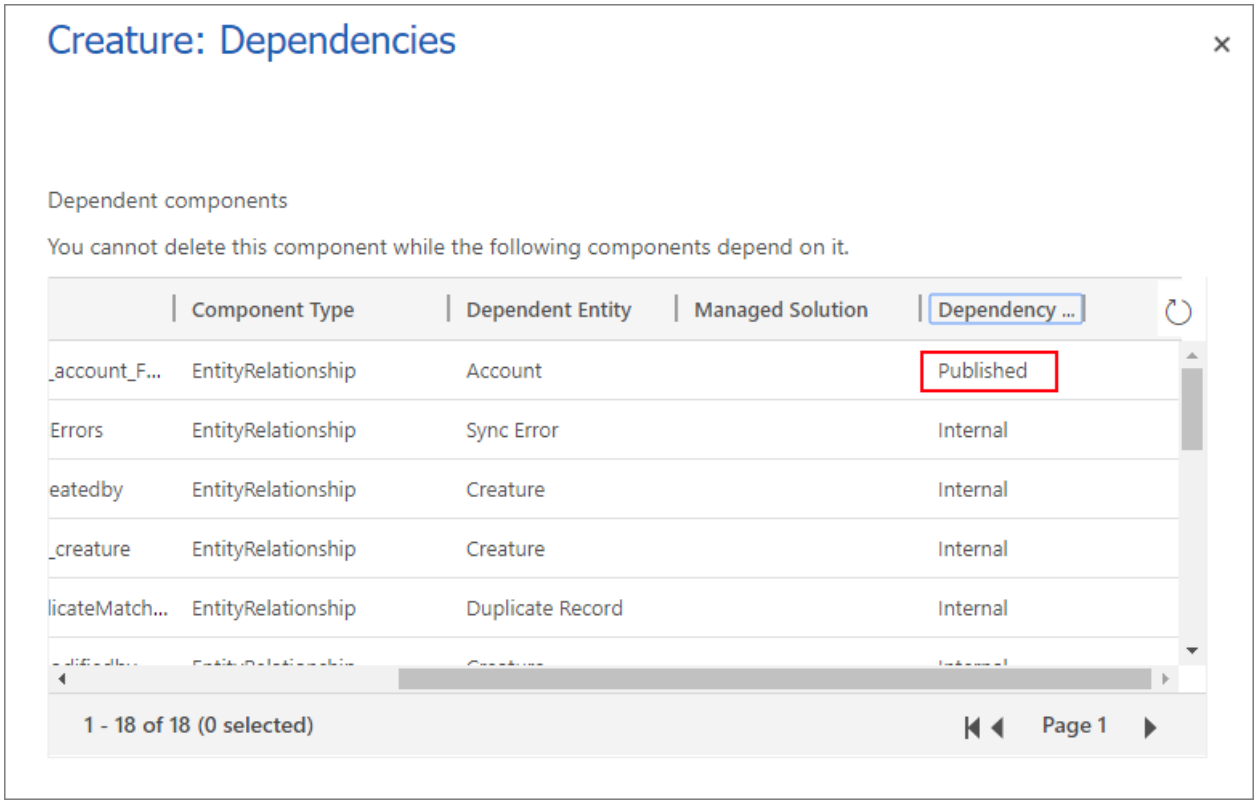
Identify table dependencies

You can identify dependencies that will prevent a table from being deleted before you try to delete it.

1. In the solution explorer with the table selected, click **Show Dependencies** in the command bar.



2. In the dialog window that opens, scroll the list to the right to view the **Dependency Type** column.



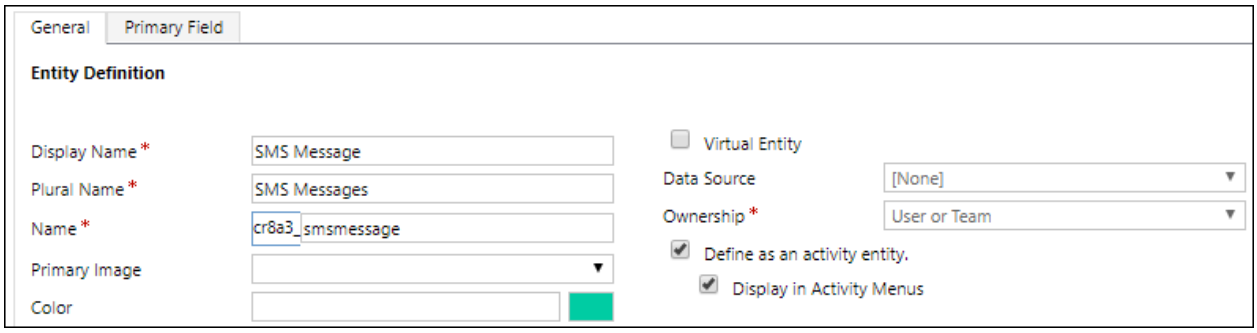
Published dependencies will block deleting a table. **Internal** dependencies should be resolved by the system.

3. Remove these published dependencies and you should be able to delete the table.

> **Note**
>
> A very common dependency is that another table form has a lookup column for the table you are deleting. Removing the lookup column from the form will resolve the dependency.

Create custom activity table

To create the table as an activity table, use the same steps described in this topic except select **Define as an activity table**.



An activity table is a special kind of table that tracks actions for which an entry can be made on a calendar. More information: Activity tables.

When you set this option some table properties are not compatible. An activity table has to conform to standard behaviors that all activity tables use.

The primary column **Name** and **Display Name** will be set to **Subject** and you cannot change this.

The following options are set by default and cannot be changed:

- **Feedback**
- **Notes (includes attachments)**
- **Connections**
- **Queues**
- **Offline capability for Dynamics 365 for Outlook**

The following options cannot be set:

- **Areas that display this table**
- **Activities**
- **Sending email**
- **Mail Merge**
- **Single row auditing**
- **Multiple row auditing**

Create a virtual table

Some options are only used when creating a virtual table.

Option	Description
Virtual Entity	Whether the table is a virtual table.
Data Source	The data source for the table.

More information: Create and edit virtual tables that contain data from an external data source

See also

Create and edit tables in Dataverse
Tutorial: Create a custom table that has components in Power Apps
Create a solution

Microsoft Dataverse table relationships

Table relationships in Microsoft Dataverse define the ways that table rows can be associated with rows from other tables or the same table. There are two types of table relationships:

- One-to-many relationships
- Many-to-many relationships

One-to-many relationships

In a one-to-many table relationship, many referencing (related) table rows can be associated with a single referenced (primary) table row. The referenced table row is sometimes referred to as the "parent" and rows of the referencing table are referred to as "children." A many-to-one relationship is just the child perspective of a one-to-many relationship.

For example, in a school scenario multiple courses might be delivered in a single classroom, so the class table would have a one-to-many relationship with the course table.

Many-to-many relationships

In a many-to-many table relationship, many table rows can be associated with many other table rows. Rows related using a many-to-many relationship can be considered peers and the relationship is reciprocal.

For example, in the same school scenario mentioned earlier, a single student can enroll in multiple courses, and each course can have multiple students. This type of relationship allows for more complex data associations and is managed using Power Apps in Dataverse.

How relationships work in Dataverse

Table relationships define how table rows can be related to each other in Dataverse. At the simplest level, adding a lookup column to a table creates a new 1:N (one-to-many) relationship between the two tables and lets you put that lookup column on a form. With the lookup column, users can associate multiple *child* rows of that table to a single *parent* table row.

Beyond simply defining how rows can be related to other rows, 1:N table relationships also provide data to address the following questions:

- When I delete a row, should any rows related to that row also be deleted?
- When I assign a row, do I also need to assign all rows related to that row to the new owner?

- How can I streamline the data entry process when I create a new related row in the context of an existing row?
- How should people viewing a row be able to view the associated rows?

Tables can also participate in a N:N (many-to-many) relationship where any number of rows for two tables can be associated with each other.

Decide whether to use table relationships or connections

Table relationships are metadata that make changes in Dataverse. These relationships allow for queries to retrieve related data efficiently. Use table relationships to define formal relationships that define the table or that most rows can use. For example, an opportunity without a potential customer wouldn't be useful. The opportunity table in Dynamics 365 for Sales also has a N:N relationship with the competitor table, also available with Dynamics 365 for Sales. This allows for multiple competitors to be added to the opportunity. You might want to capture this data and create a report that shows the competitors.

There are other less formal kinds of relationships between rows that are called *connections*. For example, it might be useful to know if two contacts are married, or perhaps they're friends outside of work, or perhaps a contact used to work for another account. Most businesses won't generate reports using this kind of information or require that it's entered, so it's probably not worthwhile to create table relationships. More information: Configure connection roles

Types of table relationships

When you view relationships in Power Apps, you might think that there are three types of table relationships. Actually there are only two, as shown in the following table.

Expand table

Relationship Type	Description
1:N (One-to-Many)	A table relationship where one table row for the **Primary table** can be associated to many other **Related table** rows because of a lookup column on the related table. When viewing a primary table row, you can see a list of the related table rows that are associated with it. In the Power Apps portal, **Current table** represents the primary table.

Relationship Type	Description
N:N (Many-to-Many)	A table relationship that depends on a special **Relationship table**, sometimes called an Intersect table, so that many rows of one table can be related to many rows of another table. When viewing rows of either table in a N:N relationship, you can see a list of any rows of the other table that are related to it.

The **N:1 (many-to-one)** relationship type exists in the user interface because the designer shows you a view grouped by tables. 1:N relationships actually exist *between* tables and refer to each table as either a **Primary/Current table** or **Related table**. The related table, sometimes called the *child* table, has a lookup column that allows storing a reference to a row from the primary table, sometimes called the *parent* table. A N:1 relationship is just a 1:N relationship viewed from the related table.

Table relationship behavior

Behaviors for related tables are important because they help ensure data integrity and can automate business processes for you.

Preserve data integrity

Some tables exist to support other tables. They don't make sense on their own. They'll typically have a required lookup column to link to the primary table they support. What should happen when a primary row is deleted?

You can use the relationship behavior to define what happens to related rows according to the rules for your business. More information: Add advanced relationship behavior

Automate business processes

Let's say that you have a new salesperson and you want to assign them a number of existing accounts currently assigned to another salesperson. Each account row might have a number of task activities associated with it. You can easily locate the active accounts you want to reassign and assign them to the new salesperson. But what should happen for any of the task activities that are associated with the accounts? Do you want to open each task and decide whether they should also be assigned to the new salesperson? Probably not. Instead, you can let the relationship apply some standard rules for you automatically. These rules only apply to task rows associated to the accounts you're reassigning. Your options are:

- Reassign all active tasks.
- Reassign all tasks.
- Reassign none of the tasks.
- Reassign all tasks currently assigned to the former owner of the accounts.

The relationship can control how actions performed on a row for the primary table row cascade down to any related table rows.

Behaviors

There are several kinds of behaviors that can be applied when certain actions occur.

Expand table

Behavior	Description
Cascade Active	Perform the action on all active related table rows.
Cascade All	Perform the action on all related table rows.
Cascade None	Do nothing.
Remove Link	Remove the lookup value for all related rows.
Restrict	Prevent the primary table row from being deleted when related table rows exist.
Cascade User Owned	Perform the action on all related table rows owned by the same user as the primary table row.

Actions

These are the actions that can trigger certain behaviors:

Expand table

Column	Description	Options
Assign	What should happen when the primary table row is assigned to someone else?	Cascade All Cascade Active Cascade User-owned Cascade None
Reparent	What should happen when the lookup value of a related table in a parental relationship is changed? More information: Parental table relationships	Cascade All Cascade Active Cascade User-

Column	Description	Options
		owned Cascade None
Share	What should happen when the primary table row is shared?	Cascade All Cascade Active Cascade User-owned Cascade None
Delete	What should happen when the primary table row is deleted?	Cascade All Remove Link Restrict
Unshare	What should happen when a primary table row is unshared?	Cascade All Cascade Active Cascade User-owned Cascade None
Merge	What should happen when a primary table row is merged?	Cascade All Cascade None
Rollup View	What is the desired behavior of the rollup view associated with this relationship?	Cascade All Cascade Active Cascade User-owned Cascade None

> **Note**
>
> Assign, Delete, Merge, and Reparent actions don't execute in the following situations:
>
> - If the original parent row and requested action contain the same values. Example: Attempting to trigger an Assign and choosing a contact that is already the owner of the row.
> - Attempting to perform an action on a parent row that is already running a cascading action.
>
> When executing an assign, any workflows or business rules that are currently active on the rows are automatically deactivated when the reassignment occurs. The new owner of the row must reactivate the workflow or business rule if they want to continue using it.

Parental table relationships

Each pair of tables that are eligible to have a 1:N relationship can have multiple 1:N relationships between them. Yet usually only one of those relationships can be considered a parental table relationship.

A parental table relationship is any 1:N table relationship where one of the cascading options in the **Parental** column of the following table is true.

Expand table

Action	Parental	Not Parental
Assign	Cascade All Cascade User-owned Cascade Active	Cascade None
Delete	Cascade All	RemoveLink Restrict
Reparent	Cascade All Cascade User-owned Cascade Active	Cascade None
Share	Cascade All Cascade User-owned Cascade Active	Cascade None
Unshare	Cascade All Cascade User-owned Cascade Active	Cascade None

For example, if you create a new custom table and add a 1:N table relationship with the account table where your custom table is the related table, you can configure the actions for that table relationship to use the options in the **Parental** column. If you later add another 1:N table relationship with your custom table as the referencing table you can only configure the actions to use the options in the **Not Parental** column.

Usually this means that for each table pair there's only one parental relationship. There are some cases where the lookup on the related table might allow for a relationship to more than one type of table.

For example, if a table has a customer lookup that can refer to either a contact or account table. There are two separate parental 1:N table relationships.

Any activity table has a similar set of parental table relationships for tables that can be associated using the regarding lookup column.

Limitations on behaviors you can set

Because of parental relationships there are some limitations, you should keep in mind when you define table relationships.

- A custom table can't be the primary table in a relationship with a related system table that cascades. This means you can't have a relationship with any action set to **Cascade All**, **Cascade Active**, or **Cascade User-Owned** between a primary custom table and a related system table.
- No new relationship can have any action set to **Cascade All**, **Cascade Active**, or **Cascade User-Owned** if the related table in that relationship already exists as a related table in another relationship that has any action set to **Cascade All**, **Cascade Active**, or **Cascade User-Owned**. This prevents relationships that create a multi-parent relationship.

Inherited access rights cleanup

Using **Reparent** and **Share** cascading behaviors are helpful when you want to provide access to rows across related tables. But there can be a change in process or design that requires a change of the cascading behavior settings.

When a table relationship uses **Reparent** or **Share**, and the cascading behavior is changed to **Cascade None**, the table relationship prevents any new permission changes from cascading to the related child tables. In addition, inherited permissions that were granted while the cascading behavior was active must be revoked.

Inherited access rights cleanup is a system job that cleans up the legacy inherited access rights that remain after the cascading behavior is changed to **Cascade None**. This cleanup doesn't affect any user that was directly granted access to a table, but does remove access from anyone who received access through inheritance only.

This is how inherited access rights cleanup works:

1. Identifies and collects all the tables that were in a cascading relationship with the updated parent.
2. Identifies and collects the users that were granted access to the related tables through inherited access.
3. Checks for users who were given direct access to a related table and removes them from the collection.
4. Removes inherited access for the collected users on the collected tables.

After the cleanup runs, users who were able to access related tables only because of the cascading feature can no longer access the rows, ensuring greater security. There are cases where the cleanup might not be successful. Learn more about how to clean up inherited access

See also

Monitor system jobs
Create and edit 1:N (one-to-many) or N:1 (many-to-one) relationships
Create Many-to-many (N:N) table relationships

Last updated on 01/09/2026

Create one-to-many or many-to-one table relationships in Microsoft Dataverse overview

08/11/2025

In Microsoft Dataverse a one-to-many (1:N) or many-to-one (N:1) relationship defines how two tables are related to each other.

Before you create a custom table relationship, evaluate whether using an existing table relationship would meet your requirements. More information: Create new metadata or use existing metadata?

Watch this video for a quick overview about how to create a table, a table relationship, and columns:
https://learn-video.azurefd.net/vod/player?id=19b21a48-4a49-4163-8506-7fb79f3290ef&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fcreate-edit-1n-relationships

You create and edit table relationships, including one-to-many and many-to-one relationships, in Power Apps.

You can also create new table relationship in your environment using any of the following experiences:

- In model-driven apps, select **New Column** from the form editor and create a *Lookup* column. More information: Add, configure, move, or delete columns on a form
- Create a new Lookup column for the related table.More information: Create and edit columns
- Import a solution that contains the definition of the table relationship. More information: Import, update, and export solutions
- Use Power Query to create new tables and fill them with data. More information: Add data to a table in Dataverse by using Power Query.
- A developer can use Metadata services to write a program to create and update table relationships. More information: Table relationship definitions

Information in this article helps you choose which designer you can use.

Create a one-to-many or many-to-one relationship

Create and edit one-to-many or many-to-one table relationships in Power Apps

Community tools

table Relation Diagram Creator is a tool that XrmToolbox community developed for Dataverse. Please see the Developer tools and resources topic for more community developed tools.

> **Note**
>
> The community tools are not a product of Microsoft and does not extend support to the community tools. If you have questions pertaining to the tool, please contact the publisher. More Information: XrmToolBox.

See also

Create and edit relationships between tables
Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships using solution explorer
Developer documentation: Table relationship definitions
Developer documentation: Table relationship eligibility

Create many-to-many table relationships overview

08/11/2025

One-to-many (1:N) table relationships establish a hierarchy between rows. With many-to-many (N:N) relationships, there's no explicit hierarchy. There are no lookup columns or behaviors to configure. Rows created using many-to-many relationships can be considered peers and the relationship is reciprocal.

One example of a many-to-many relationship is defined between two standard tables included with the Dynamics 365 for Sales app. The opportunity table has an N:N relationship with the competitor table. This relationship allows for multiple competitors to be added to the opportunity and multiple opportunities associated with the same competitor.

With many-to-many relationships, a relationship (or intersect) table stores the data that associates the tables. This table has a one-to-many table relationship with both of the related tables and only stores the necessary values to define the relationship. You can't add custom columns to a relationship table and it's never visible in the user interface.

Creating a many-to-many relationship requires choosing the two tables that you want to participate in the relationship. For model-driven apps you can decide how you want the respective lists to be available within the navigation for each table. These are the same options used for the primary table in 1:N table relationships.

Not all tables can be used with many-to-many relationships. If the table isn't available to be chosen in the designer, you can't create a new many-to-many relationship with this table. More information: Developer documentation: table relationship eligibility

> ⓘ **Note**
>
> You can also create new many-to-many (N:N) table relationship in your environment using the following:
>
> - Import a solution that contains the definition of the relationship. More information: Import, update, and export solutions
> - A developer can use Metadata services to write a program to create and update table relationships. More information: Developer documentation: Table relationship definitions

Next steps

Create many-to-many table relationships

Create a relationship between tables

Data in one table often relates to data in another table. For example, you might have a **Teachers** table and a **Class** table, and the **Class** table might have a lookup relation to the **Teachers** table to show which teacher instructs the class. You can use a lookup column to show data from the **Teachers** table. This is commonly referred to as a lookup column.

Define a relationship

You can create several types of relationships from one table to another (or between a table and itself). Each table can have a relationship with more than one table, and each table can have more than one relationship to another table. Some common relationship types are:

- **Many-to-one** - In this type of relationship, each row in table A can match more than one row in table B, but each row in table B can match only one row in table A. For example, a class has a single classroom. This is the most common type of relationship and is shown in the column list as a **Lookup column**
- **One-to-many** - In this type of relationship, each row in table B can match more than one row in table A, but each row in table A can match only one row in table B. For example, a single teacher, teaches many classes.
- **Many-to-many** - In this type of relationship, each row in table A can match more than one row in table B, and vice versa. For example, students attend many classes, and each class can have multiple students.

Additionally, you can set advanced cascading behaviors on many-to-one and one-to-many relationships whenever an action is taken on the parent table.

Add a lookup column (Many-to-one relationship)

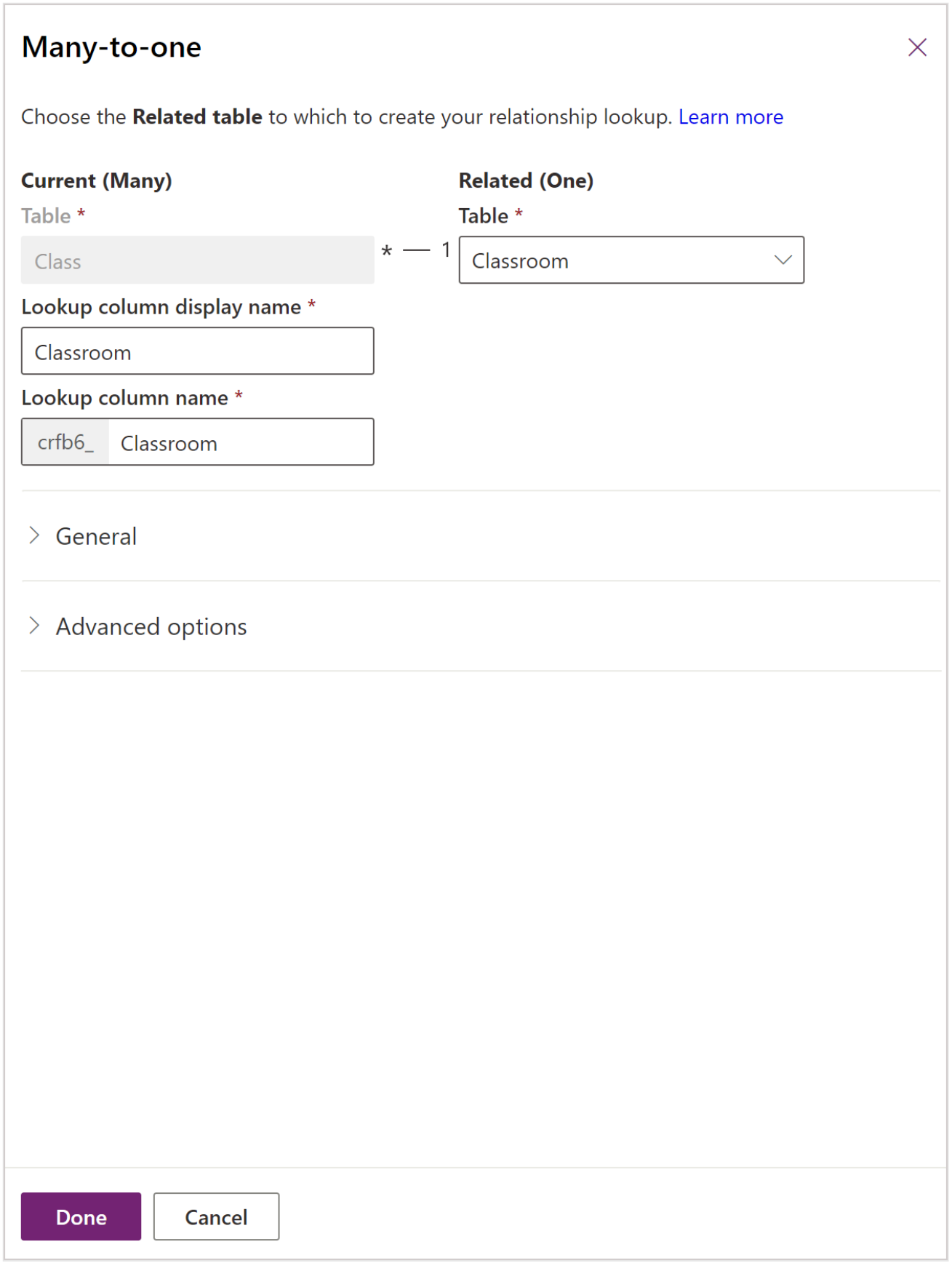
To add a lookup relation to a table, create a relation under the **Relationships** area and specify the table with which you want to create a relationship.

1. In Power Apps, on the left navigation pane select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.

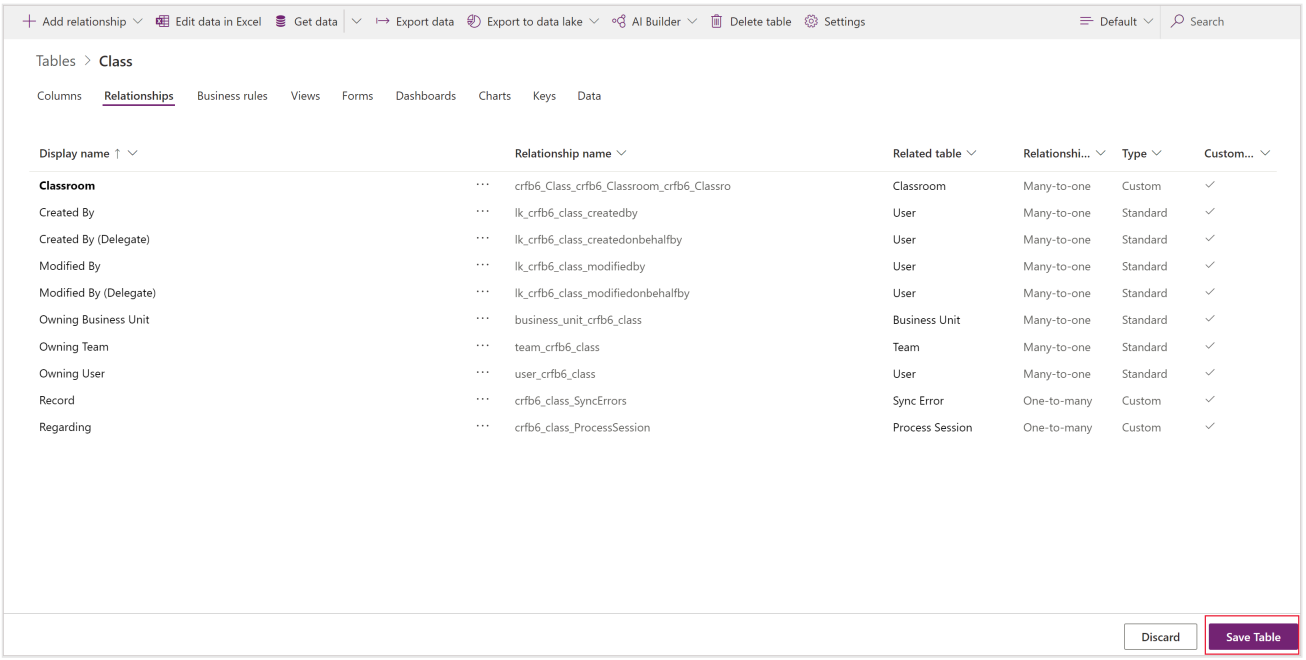
2. Select an existing table, or Create a new table, and then select **Relationships**.

3. Select **New relationship**, and then select a relationship type, such as **Many-to-one**.

4. On the right pane, select a **Related** table for the **Current** table, and then select **Done**.



5. Select **Done**.



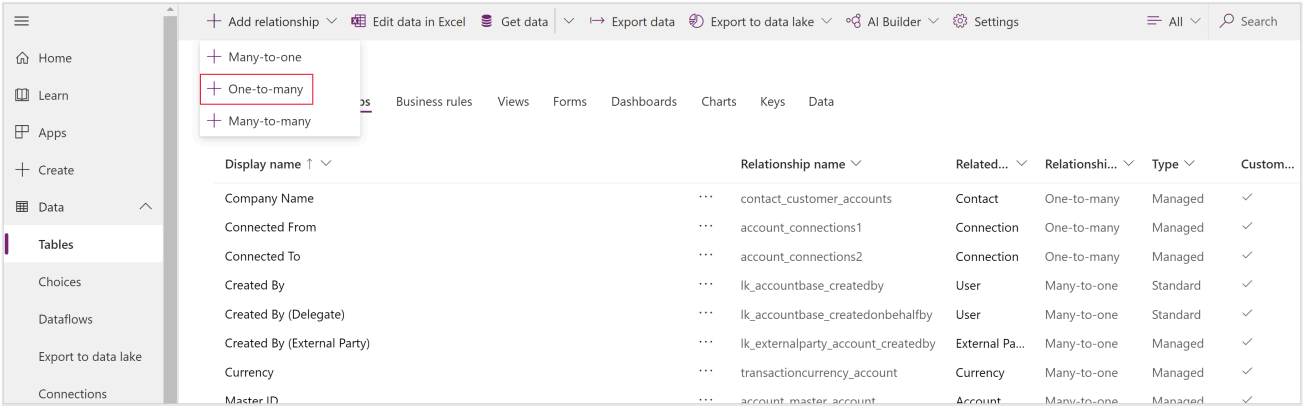
Add a one-to-many relationship

To add a one-to-many relationship, create a relation under the **Relationships** area and specify the table with which you want to create a relationship.

1. In Power Apps, select **Tables** in the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Select an existing table, or Create a new table, and then select **Relationships**.

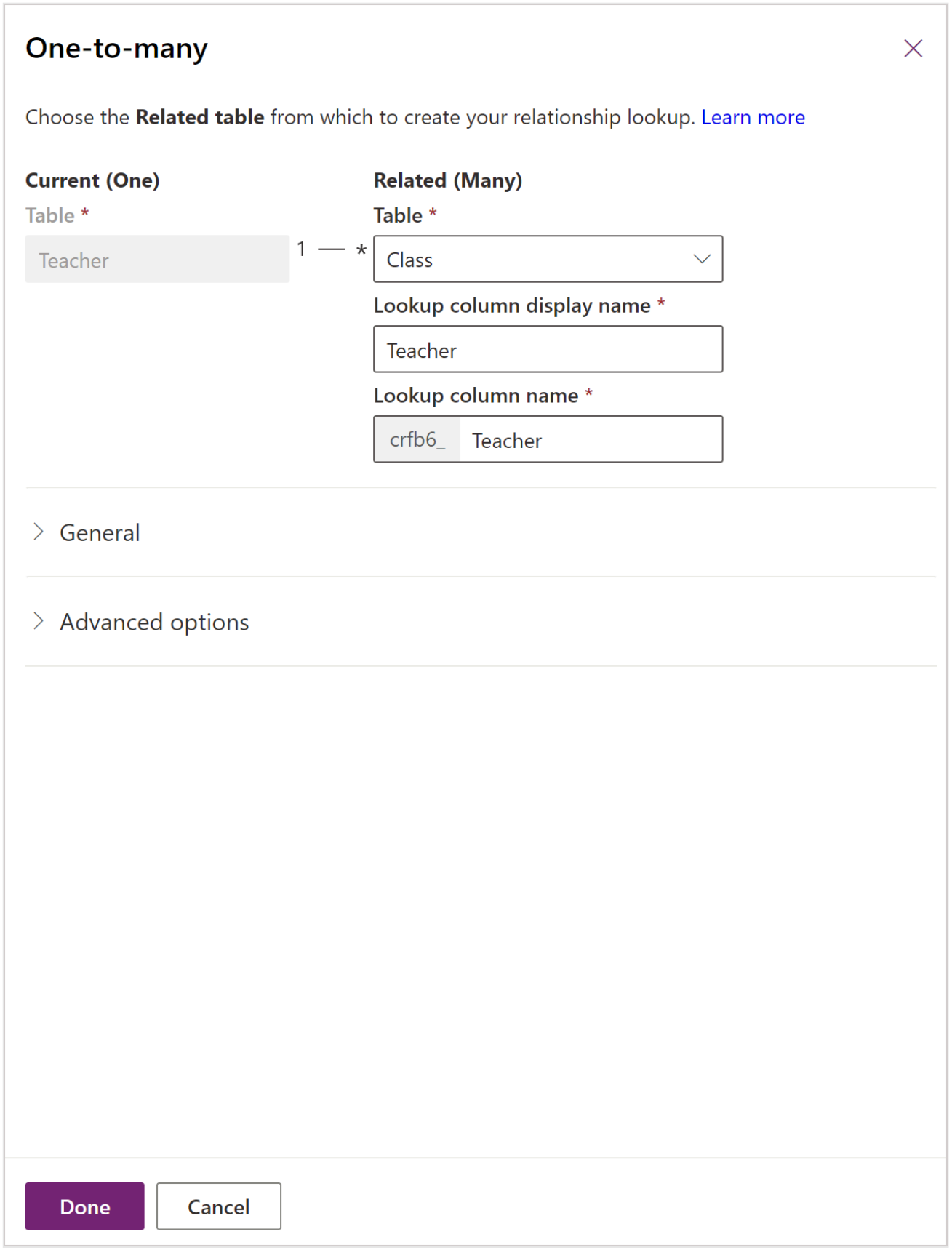
3. Select the down arrow to the right of **Add relationship**, and then select **One-to-many**.



4. Select a table. Notice the **Look up** columns will be shown on the **Current** table, they'll default with the tables name (in this example Teacher) but you can change them if needed. Select **Done** to add the relationship to your table.

> **Note**

In the case of a One-to-many relationships, the Look up column will be created on the related table, not the table you currently have selected. If you need the lookup on the current table, create a Many-to-one relationship.



5. Select **Done**.

Add a many-to-many relationship

To add a many-to-many relationship, create a relation under the **Relationships** area and specify the table with which you want to create a relationship.

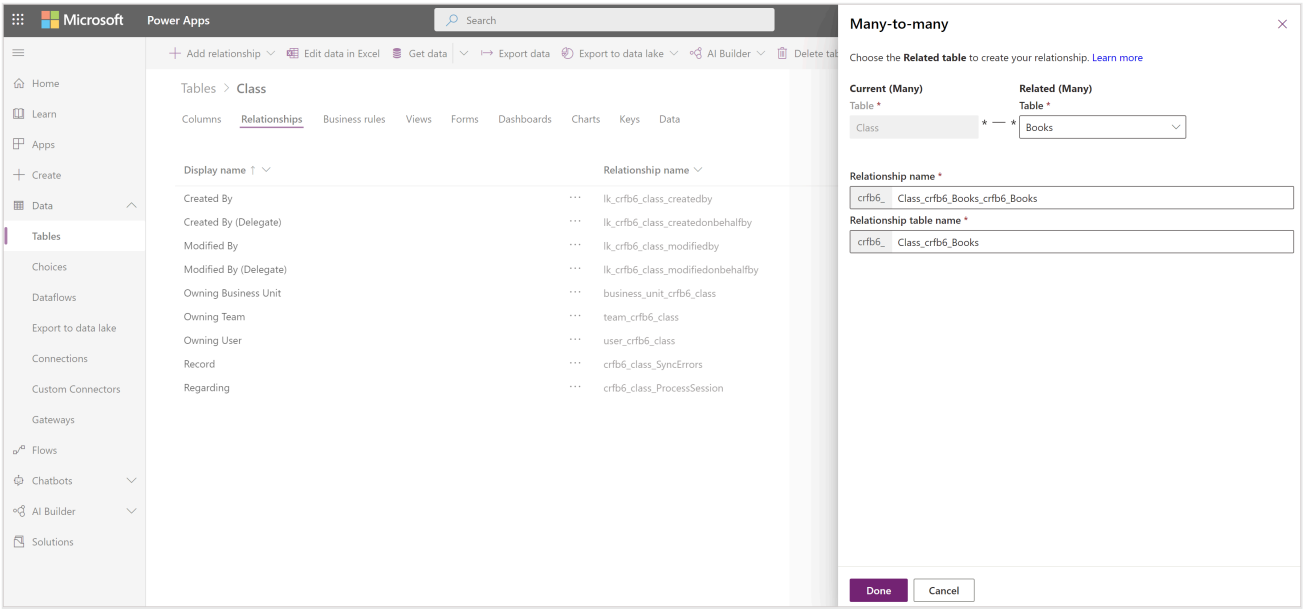
1. On powerapps.com, select **Tables** in the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Select an existing table, or Create a new table

3. Select the **Relationships** area.

4. Select **Add relationship** and then select **Many-to-many**. This opens a panel for you to choose the table you want to create a relationship to. Select the table from the **Related table** drop-down.

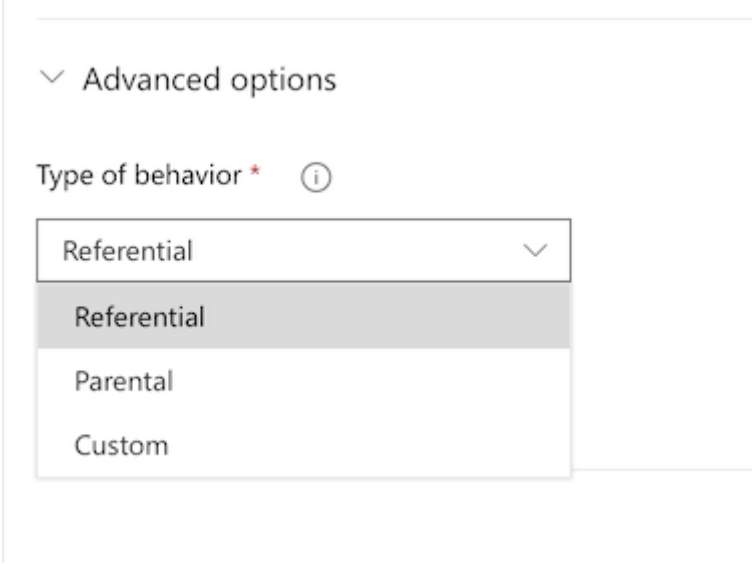
5. After selecting a table, the names for the relationship and relationship table appear. They default to the names of the tables combined, but you can change them if needed.



6. Select **Done** to add the relationship to your table, and then select **Save table**.

Add advanced relationship behavior

While building a one-to-many or a many-to-one relationship, you can also set advanced behaviors.



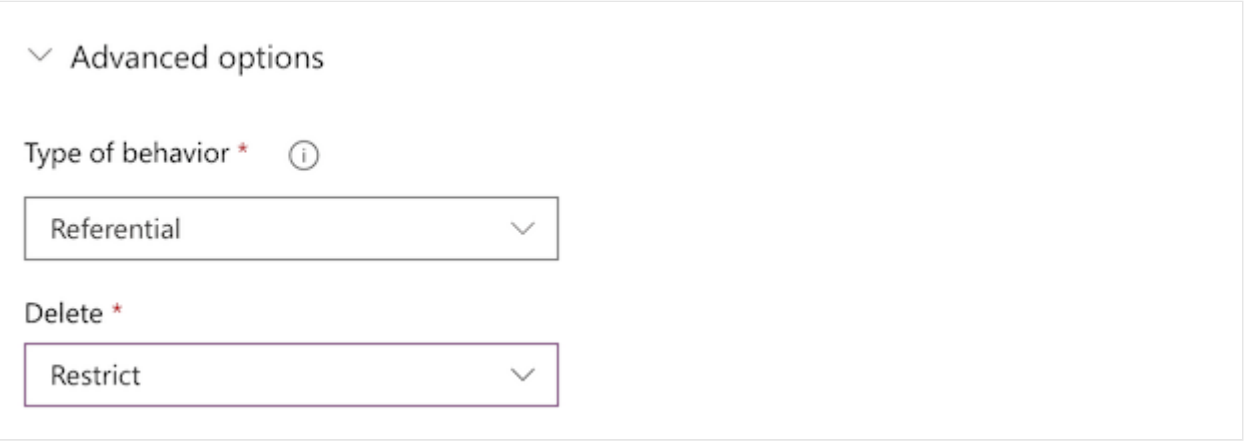
These options are also referred to as cascading behaviors because they cascade down the hierarchy of related tables. For example, it might be desirable to delete the related tests and homework of a student if a student is removed from the system. This type of behavior is called a parental relationship.

On the other hand, you might decide that you don't want actions to cascade down the hierarchy. For example, in the teacher to class relationship you could decide that the child table (class) shouldn't* be deleted when a parent (teacher) is deleted. This is called a referential relationship.

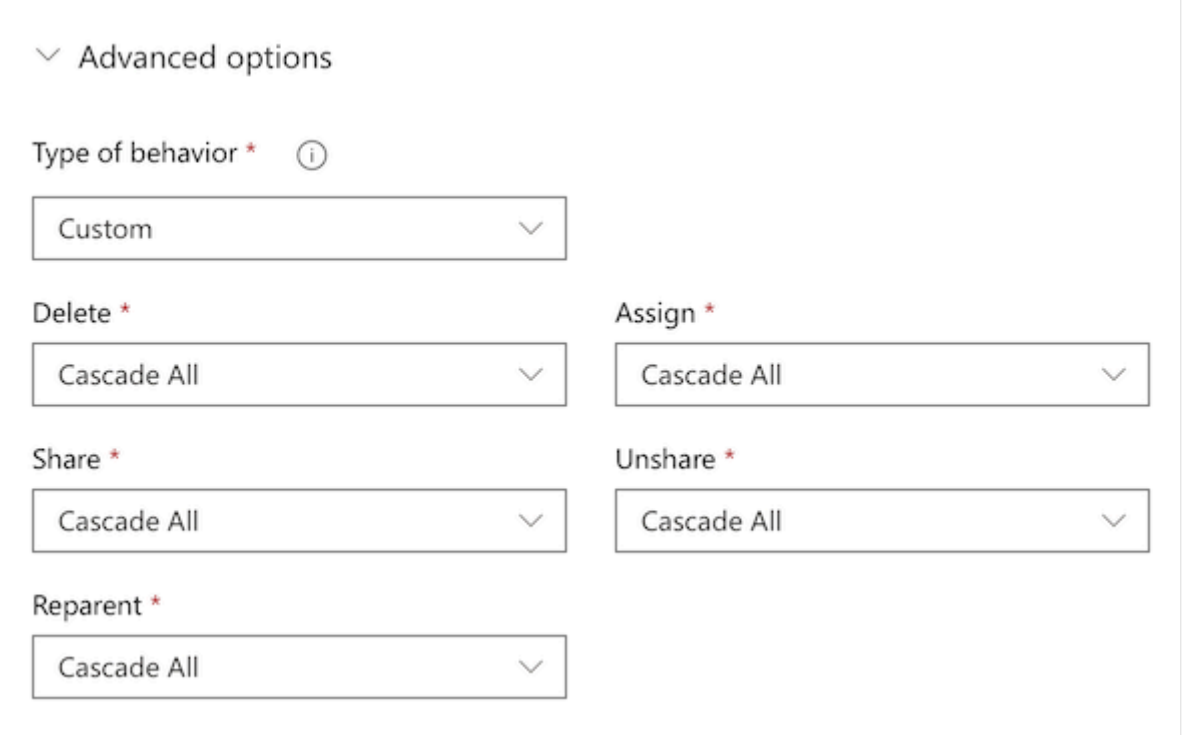
As you model your business data by creating custom tables or when using existing Common Data Model tables, consider the behavior you require and the implications for the entire hierarchy of related tables and choose between one of the following standard behaviors:

- **Referential, Remove Link:** In a referential relationship between two tables, you can navigate to any related rows, but actions taken on one table don't affect the other. For example, if you have a one-to-many relationship between teachers and classes, deleting a teacher has no impact on the related class.

- **Referential, Restrict Delete:** In a referential, restrict delete relationship between two tables, you can navigate to any related rows. Actions taken on the parent row don't affect the child row, but the parent row can't be deleted while the child row exists. This behavior is useful if you don't want child rows to become orphaned. This restriction forces the user to delete all of the children before deleting the parent.



- **Parental:** In a parental relationship between two tables, any action taken on a record of the parent table is also taken on any child table records that are related to the parent table record. For example, the owner of the parent record has inherited access to the child table records and when the parent record is deleted, all of the child records will also be deleted.
- **Custom:** In a custom relationship between two tables, you select the behavior associated with each of a set of possible actions.



For more information on defaults and custom behaviors: Configure table relationship behavior.

Use a lookup column in an app

If you create an app automatically from a table that contains a lookup column, it appears as a **Drop down** control that contains data from the **Primary name** column of the table.

Add 1:N and N:N relationships for canvas apps

Use the **Relate** function to link two rows through a one-to-many or many-to-many relationship in Microsoft Dataverse. More information: Relate and Unrelate functions in Power Apps

Next steps

- Generate an app by using a Dataverse database
- Create an app from scratch using a Dataverse database

Last updated on 04/22/2026

Create and edit one-to-many or many-to-one table relationships using Power Apps

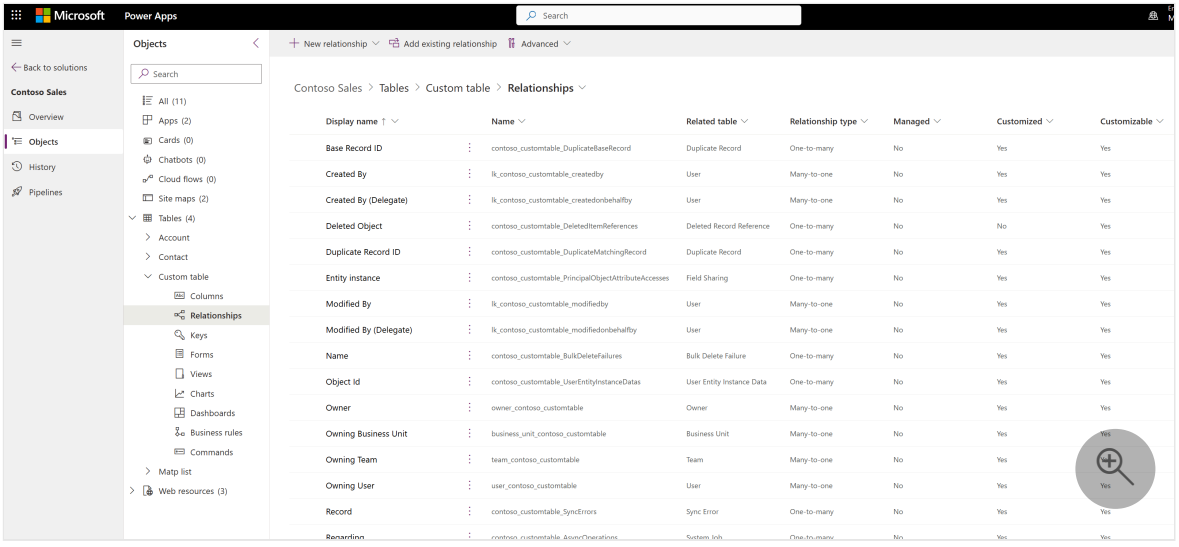
Article • 09/19/2024

In Microsoft Dataverse a one-to-many (1:N) or many-to-one (N:1) relationship defines how two tables are related to each other.

Power Apps provides an easy way to create and edit one-to-many (1:N) or many-to-one (N:1) relationships for Microsoft Dataverse. For more information about these kinds of relationships, go to Create one-to-many or many-to-one table relationships overview.

View table relationships

1. Sign in to Power Apps, and open the environment you want.
2. Select **Solutions** on the left navigation pane, and then open the solution you want.
3. Select **Tables** and open the table that has the relationships you want to view. If the item isn't in the side panel pane, select ...More and then select the item you want.
4. Select the **Relationships** area to view all relationships configured for the table.



Create a relationship

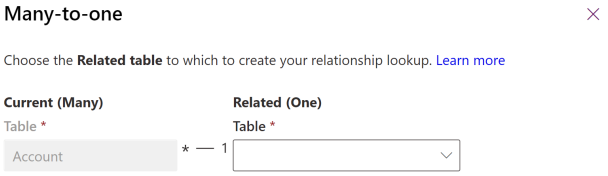
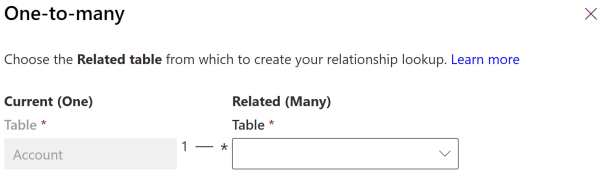
While viewing table relationships, on the command bar, select **New relationship**, and then select either **Many-to-one** or **One-to-many**.

> **Note**

For information about **Many-to-many** relationships see [Create N:N (many-to-many) relationships]

Depending on your choice the following is displayed:

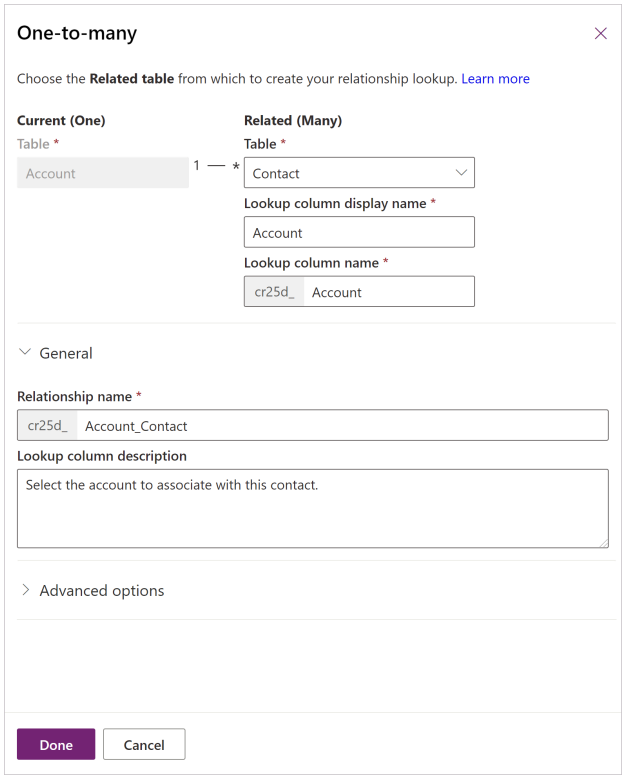
Expand table

Type	Panel
Many-to-one	
One-to-many	

Choose the **Related** table for the relationship you want to create between the two tables.

> **Note**
>
> With either choice, a lookup column is created on the **Current** table. For more information about look columns, go to [Use the lookup column on a row].

Once you select the related table, you can edit the details of the relationship. In this example, multiple contact table rows can be associated with a single account.



You can edit the default values provided before you save. Select **General** to view the **Relationship name** and **Lookup column description** values.

Expand table

Field	Description
Lookup column display name	The localizable text for the lookup column that will be created on the related table. This can be edited later.
Lookup column name	The name for the Lookup column that will be created on the related table.
Lookup column requirement	**Optional**. Column value doesn't have to be selected to save the record. **Business Recommended**. Should be selected indicator but column value doesn't have to be selected to save the record. **Business Required**. Column value must be specified in order to save the record.
Searchable	Clear if you want to hide the relationship from **Advanced Find** in model-driven apps.
Relationship name	The name for the relationship that will be created.
Lookup column description	The description for the lookup column. In model-driven apps this appears as a tooltip when people hover their mouse over the column. This can be edited later.

You can continue editing the table. Select **Save table** to create the relationship you have configured.

Advanced options

For both the **Current table display settings** and the **Related table display settings**, you can edit the display option columns, which control how the related tables are displayed for model-driven apps. More information: Advanced options

Relationship behavior

Relationship behavior determines how records are handled as the action cascades down the hierarchy of related tables. For example, it might be desirable to delete the related tests and homework of a student if a student is removed from the system. This type of behavior is called a parental relationship. For more information about how to configure relationship behavior, go to Add advanced relationship behavior.

Edit relationships

While viewing table relationships, open the relationship you want to edit to display the properties pane.

The only columns you can edit are **Lookup column display name** and **Lookup column description**. These can also be edited in the properties of the lookup column in the related table. More information: Edit a column

> **Note**
>
> Each relationship can be found within the primary table or the related table as a **Many-to-one** or **One-to-many** relationship. Although it can be edited in either place, it is the same relationship.
>
> The publisher of a managed solution can prevent some customizations of relationships that are part of their solution.

Delete relationships

Deleting a one-to-many relationship deletes the lookup column on the related table.

> **Note**
>
> You can't delete a relationship that has dependencies. For example, if you have added the lookup column to a form for the related table, you must remove the column from the form before you delete the relationship.

While viewing table relationships, select the relationship you want to remove or delete, choose from the following relationship types:

- Custom relationships in a single solution. On the command bar, select **Remove** > **Delete from this environment**.
- Custom relationships in multiple solutions, system, or standard table relationships. On the command bar, select **Remove** > **Remove from this solution**.

See also

Create and edit relationships between tables
Create and edit 1:N (one-to-many) or N:1 (many-to-one) relationships
Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships using solution explorer
Edit a column

Feedback

Was this page helpful? Yes No

Provide product feedback

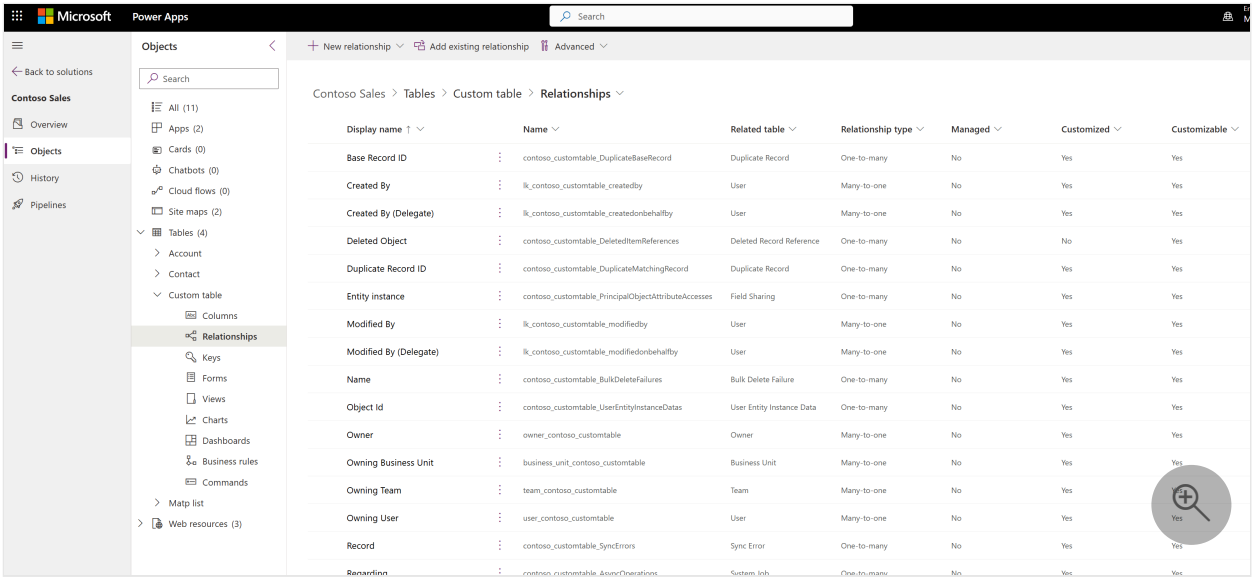
Create many-to-many table relationships using Power Apps

Article • 09/17/2024

Power Apps provides an easy way to create and edit many-to-many relationships for Microsoft Dataverse tables. A many-to-many relationship defines the type of connection between two tables where multiple records in one table can be associated with multiple records in another table. For example, in a school scenario, a single student can enroll in multiple courses, and each course can have multiple students. This type of relationship allows for more complex data associations and is managed using Power Apps in Dataverse. For more information about many-to-many table relationships, go to Create many-to-many table relationships overview.

View many-to-many table relationships

1. From Power Apps, select **Solutions** on the left navigation pane, and open the solution you want.
2. Open the table that has the relationships you want to view.
3. Select the **Relationships** area to view all relationships configured for the table.

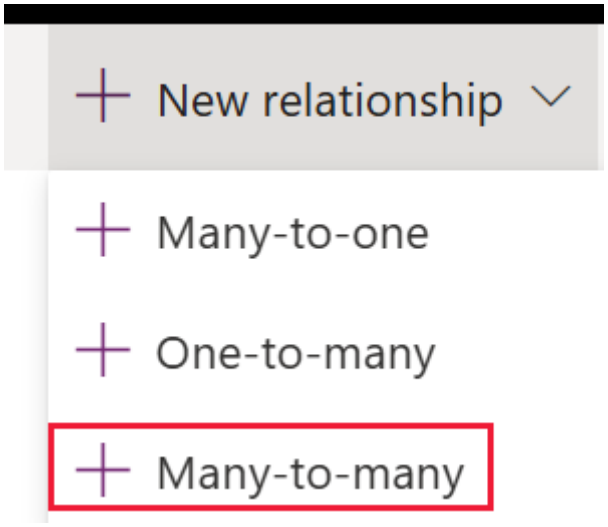


Many-to-many relationships have a **Relationship type** of **Many-to-many**.

> **Note**
>
> The table you view might have no **Many-to-many** relationships.

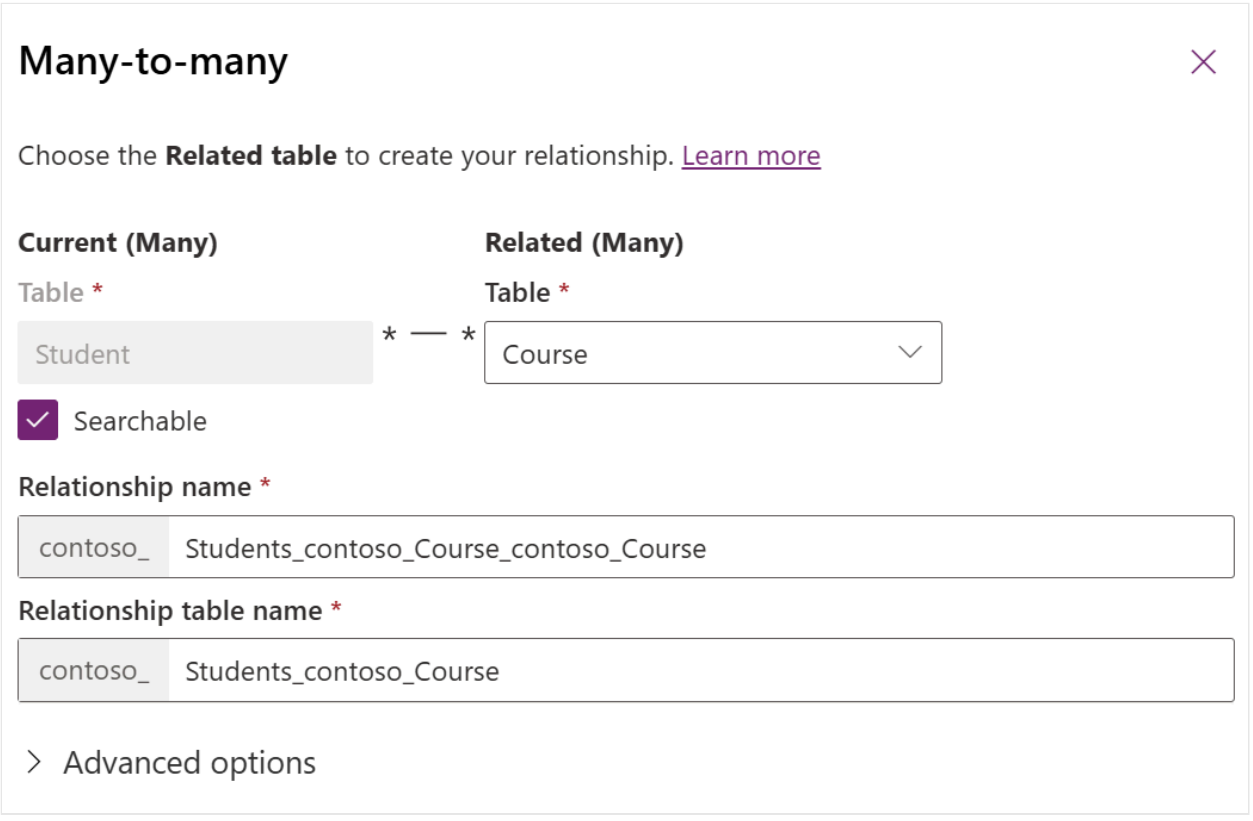
Create relationships

While viewing table relationships, on the command bar, select **New relationship**, and then select **Many-to-many**.



The **Many-to-many relationship** editor is displayed.

In the **Related (Many)** dropdown list, choose the table you want to relate to the current table. In this example, a many-to-many relationship with the student and course tables is created.



The values for these columns are generated for you based on the tables chosen.

Clear **Searchable**, if you want to hide the relationship from **Advanced Find** in model-driven apps.

If you want to edit the display settings for the relationship, expand **Advanced options** and make the changes you want. More information: Advanced options

> **Note**
>
> If you create more than one **Many-to-many** relationship with the same two tables, you must edit the generated **Relationship Name** and **Relationship table name** columns so that they are unique.

Select **Done** to close the **Many-to-many** panel. The relationship is created when you save changes to the table.

Advanced options

For both the **Current table display settings** and the **Related table display settings**, you can edit the display option columns, which control how the related tables are displayed for model-driven apps.

Expand table

Field	Description
Display option	How the related table list should be displayed. More information: Display Options
Custom label	Specify the localizable text to be used instead of the plural name when you select **Use Custom label** as the **Display option** .
Display area	Select one of the available groupings to display this list. The available options are: **Details** (for the *Common* group), **Marketing**, **Sales**, and **Service**.
Display order	Controls where the navigation item will be included within the selected display area. The range of allowed numbers begins with 10,000. Navigation pane items with a lower value appear above other relationships with a higher value.

Display Options

These are the available display options:

Expand table

Option	Description
Do not Display	Do not display the related tables for this relationship.
Use custom label	When this option is chosen the **Custom label** column is enabled so that you can specify the localizable text to be used instead of the plural name.
Use plural name	Use the plural display name defined for the related table.

Create a many-to-many relationship with the same table (self-referential relationship)

There might be times when you need to create a many-to-many relationship that references the same table. For example, account records might have multiple parent account records. However, if the self-referential relationship doesn't have custom labels configured, distinguishing the two related record types that appear in the app isn't apparent to users. This is because the related records appear twice in the app by using the same table name.

To create a self-referential relationship, use custom labels.

1. Open or create a solution and then create a new custom table. In this example, the table is named *Custom table*.

2. Open the custom table, select the **New** > **Relationship** > **New relationship** > **Many-to-many relationship**.

3. Under **Related (Many)** select the same table, in the example *Custom table*, that's also the **Current (Many)** table.

4. Expand **Advanced options** and complete the relationship definition as follows. Add custom labels to both the current and the related table's relationship:

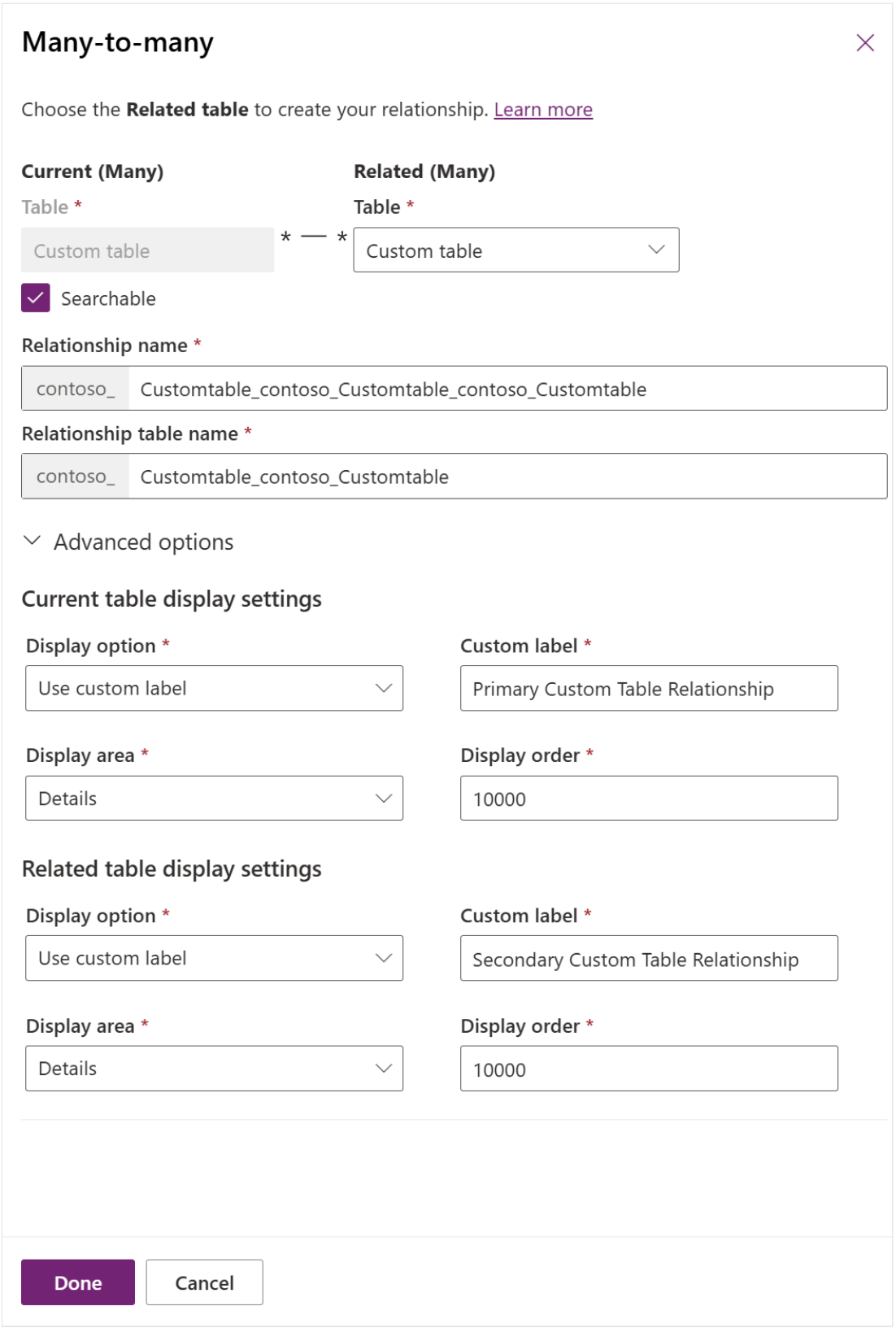
   Current table display settings section

   - **Display option**: **Use custom label**
   - **Custom label**: *Primary Custom Table Relationship*

   Related table display settings section

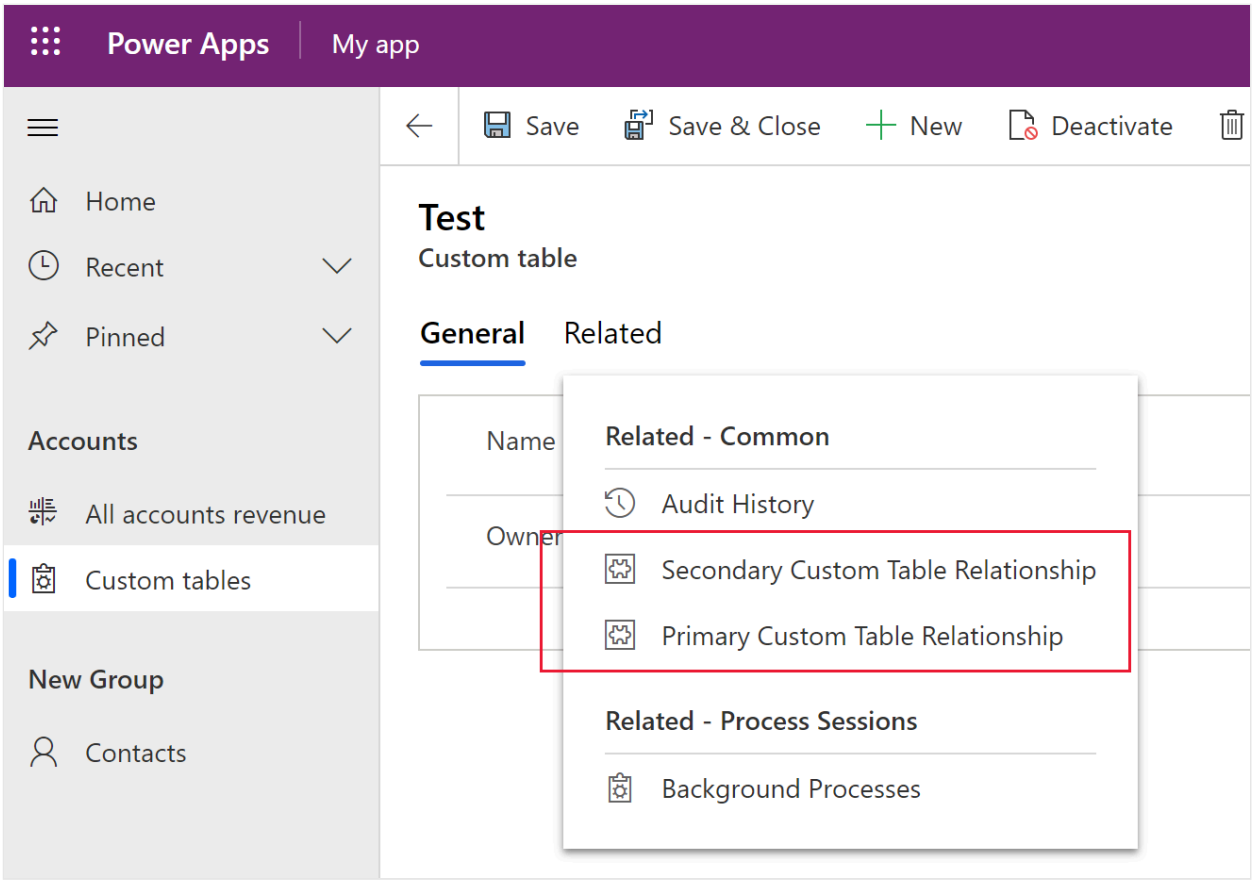
   - **Display option**: **Use custom label**

- **Custom label**: *Secondary Custom Table Relationship*



5. Select **Done** to save the relationship, and then publish the customization.

Because the related record types use the custom labels defined for the self-referential relationship instead of the table name, users running the app can distinguish between the two related record types that exist.



Delete relationships

Deleting a many-to-many relationship deletes the relationship table created. All data connecting tables using the relationship are lost.

While viewing table relationships, select the relationship you want to delete, and then on the command bar select **Remove** > **Delete from this environment**.

See also

Create N:N (many-to-many) table relationships overview

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Feedback

Was this page helpful? Yes No

Provide product feedback

Set managed properties for relationships

Article • 09/19/2024

Managed properties apply when you include a column with a managed solution and import it into another organization. These settings allow a solution maker to have some control over the level of customization that they want to allow people who install their managed solution to have when they customize a table relationship.

Set managed properties for a relationship

1. Sign in to Power Apps ,and select the environment you want.
2. Select **Solutions**, open the solution you want, and then open the table that has the relationship where you want to view the managed properties.
3. Select the relationship, and then select **Advanced** > **Managed properties** on the command bar.

With relationships, the only managed property is **Allow customizations**. This single setting controls all changes that can be made to the table relationship.

See also

Create and edit relationships between tables

Solutions in Power Apps overview

Feedback

Was this page helpful? Yes No

Provide product feedback

Configure connection roles

Article • 03/13/2023

With Microsoft Dataverse, you can define **connections** between table rows without creating a table relationship. In model-driven apps people can establish a named link between rows to establish less a formal relationship, which doesn't justify creating an actual table relationship. Some examples include *friend*, *sibling*, *spouse*, *attendee*, and *stakeholder*. Some connections can also be reciprocal, such as *child* and *parent*, *husband* and *wife*, or *doctor* and *patient*.

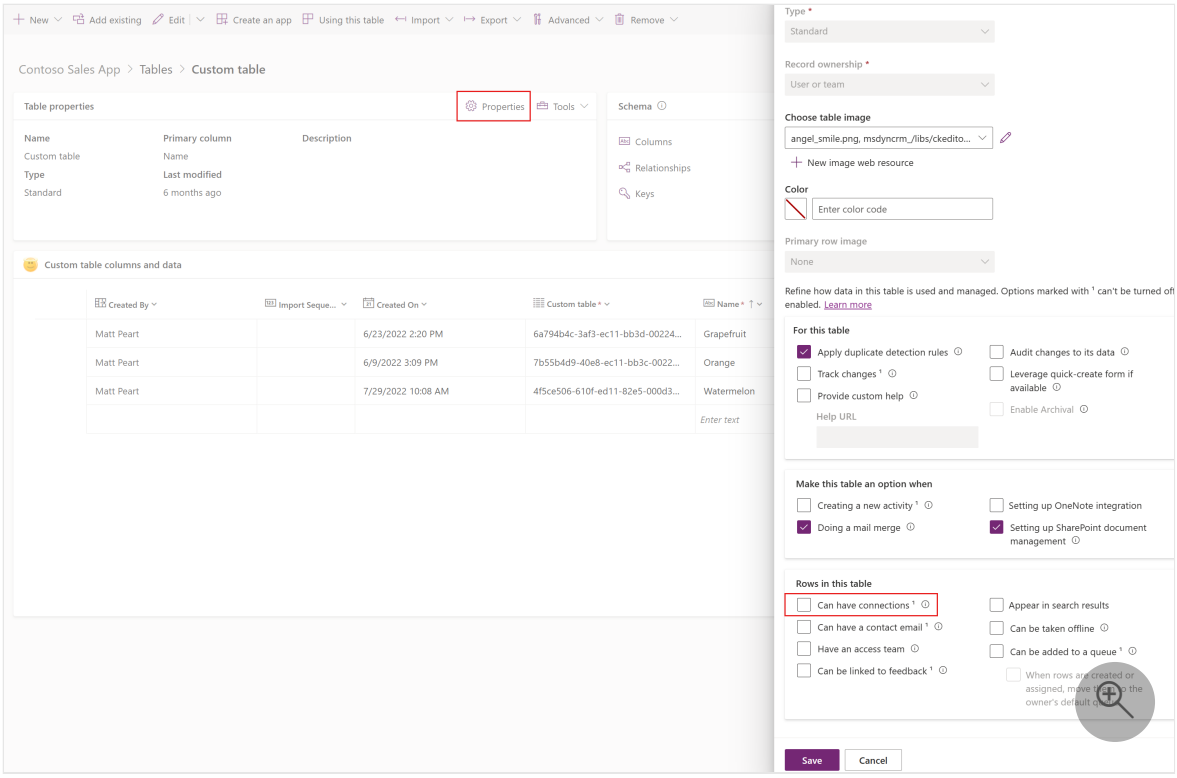
When people set a connection between two rows, they can also add a description and additional information such as start and end dates for the relationship. More information: Add a connection role to link rows to each other

Anyone with write access to the **Connection Role** table can establish which connections are available for people to use.

> **Important**
>
> For a table to be available as a row type for a new or existing connection role, the **Can have connections** property must be enabled for the table.

Enable connection roles for a table

1. Sign in to Power Apps.
2. Open the table you want or create a new one. More information: Edit a table
3. Select **Properties**, and then in the right properties pane, select **Can have connections**.



4. Select **Save**.

View connection roles

There are a number of standard connection roles already configured in Dataverse.

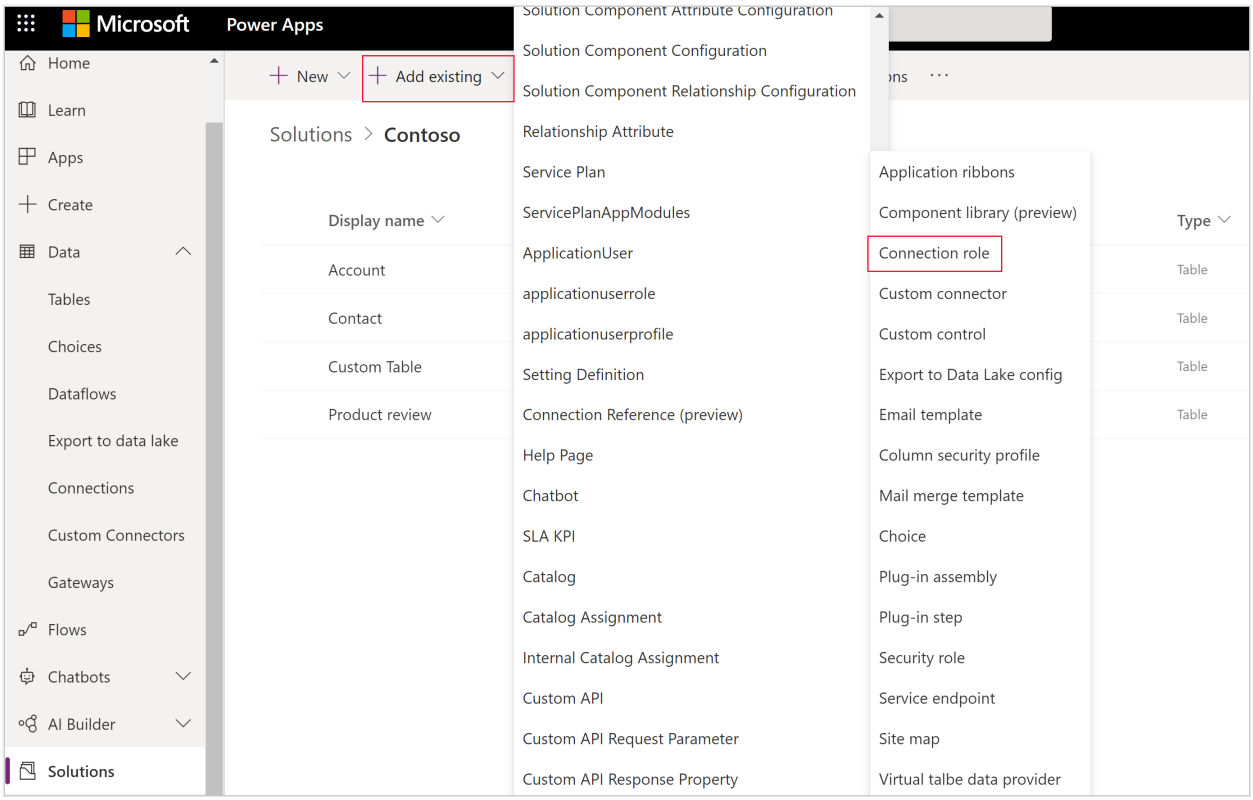
1. Sign in to Power Apps, and then on the left pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
2. Open the unmanaged solution you want.
3. On the command bar, select **Add existing** > **Connection Role**. The list of available connection roles are displayed.
4. Select **Cancel** to close the **Add existing connection Roles** pane without adding a connection role to the solution.

> **Note**
>
> - If you want to distribute connection roles with a solution, make sure they are included in the solution you want to distribute. More information: **Add connection roles to a solution**

Add connection roles to a solution

Because connection roles are *solution aware*, which means that they can be included in a solution, you can also add connection roles to a solution you distribute.

Generally we don't recommended you edit components in the internal **Default Solution**. Within the solution you've created to work in, you can use the **Add Existing** command in the **Solutions** area to bring any of the active connection roles into your solution.



Once you add the connection role to your solution, you can edit it wherever it's visible.

> **Note**
>
> The connection role status is not included with the connection role when it is exported from a solution. Therefore, when the solution is imported into a target environment, the status will be set to active by default.

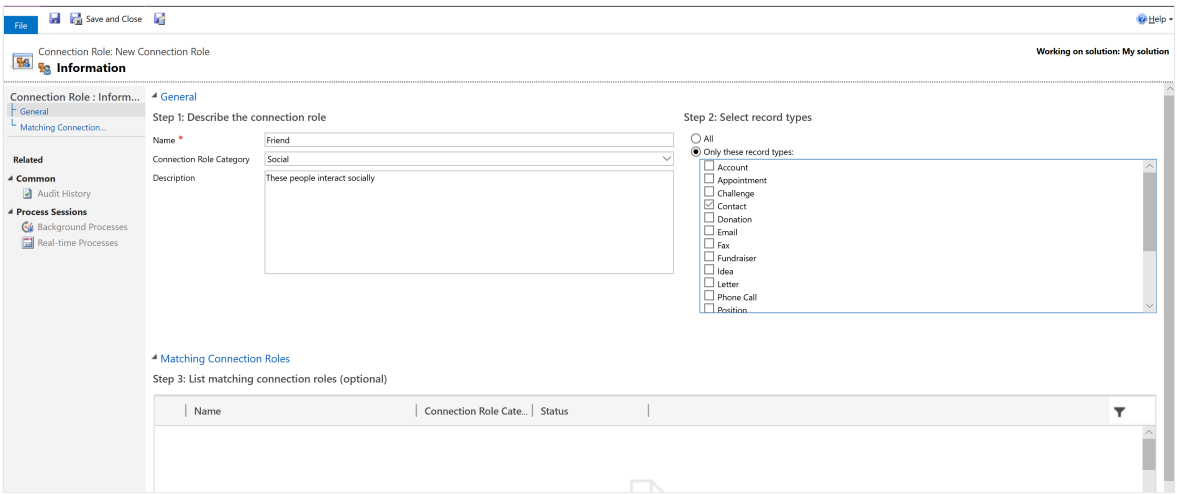
Create a connection role

> **Important**
>
> If you intend to distribute a solution that includes new connection roles or changes to the existing connection roles you must add them to the solution you will distribute. Editing or adding new connection roles in the **Default Solution** will not include them in a solution you can distribute. More information **Add connection roles to a solution**

1. Sign in to Power Apps and then on the left pane select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. Open the unmanaged solution your want, and then on the command bar select **New** > **Other** > **Connection role**.

3. Complete the three steps on the form to Describe the connection role.



Describe the connection role

Set the following columns:

Column	Description
Name	(Required) The text describing the connection.
Connection Role Category	A group describing the category of the connection. More information: Connection Role Category values
Description	Provide a definition for the role.

Connection Role Category values

The default **Connection Role Category** values are:

- Business
- Family
- Social
- Sales
- Other
- Stakeholder

- Sales Team
- Service

You can add new categories or modify existing ones by editing the **Category** global choice. More information: Create and edit global choices for Dataverse (picklists)

Select row types

Select which row types should be available to connect.

> **Note**
>
> Although **All** is selected by default, make sure you consider which types are appropriate for the connection role you are adding.

Matching connection roles

In this optional step, you can define any roles that be applied in a reciprocal manner. It isn't required, but connections are more meaningful if these are defined.

For example, people can set that Glen is a *Friend* to Mary, but does this mean that Mary is a *Friend* to Glen? We hope so. But if Glen is the *Father* of Mary it doesn't mean that Mary is the *Father* of Glen. Establishing correct reciprocity requires this extra step.

When people set a connection role that doesn't have a matching connection role, the role will only be displayed when viewing the connection from the row that the connection was applied to. When viewed from the connected row, the role will be empty unless a matching role is set.

For role definitions like *Friend*, *Spouse*, *Colleague*, or *Sibling*, it's best to assign the matching role to itself. If a single matching connection role is configured, the single matching connection role will be applied in both directions.

> **Important**
>
> You will need to save a new connection role without this matching connection role before you can set the matching connection role to itself.

You'll find that some connection roles are already configured with matching connection roles. *Former Employee* is matched with *Former Employer* and vice versa. This kind of one-to-one matching connection role is most common.

You can configure multiple matching connection roles to describe complex relationships. If you create a connection role such as *Father*, you could configure two more roles such as Daughter,* and *Son* and apply both of them as matching connection roles to *Father*. In turn, both the *Daughter* and *Son* connection roles should be matched to *Father*. Of course, then you should set up an equivalent role for *Mother* that is similarly matched with *Daughter* and *Son*.

> **Tip**
>
> Before you create a complex set of connection roles, consider if a more simple set of roles will be enough. For example, rather than creating a complex set of connection roles like *Father*, *Mother*, *Son*, and *Daughter* - consider if simply using *Parent* and *Child* will work for you.

If more than one matching connection role is configured, those connection roles represent the only valid reciprocal roles. The first one will be applied automatically as the default value. If the default value isn't correct, people need to manually edit the connection and choose between valid options defined in the configuration.

See also

Create connections to define and view relationships between rows (classic)
Create and edit global choices for Dataverse (picklists)
Create and edit relationships between tables

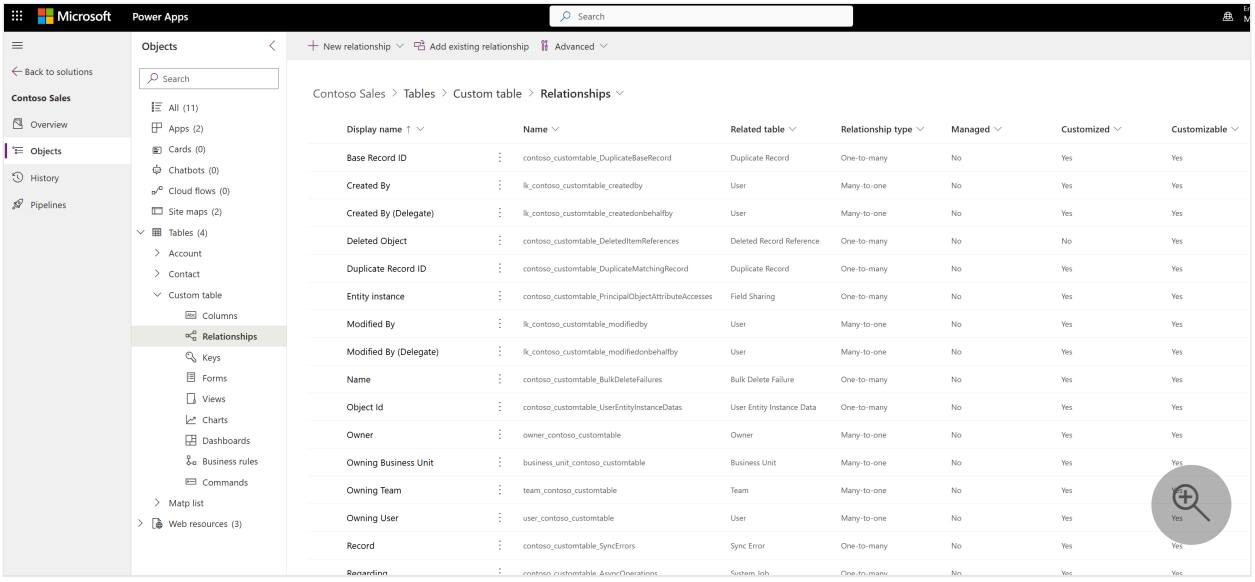
Create many-to-many table relationships using Power Apps

Article • 09/17/2024

Power Apps provides an easy way to create and edit many-to-many relationships for Microsoft Dataverse tables. A many-to-many relationship defines the type of connection between two tables where multiple records in one table can be associated with multiple records in another table. For example, in a school scenario, a single student can enroll in multiple courses, and each course can have multiple students. This type of relationship allows for more complex data associations and is managed using Power Apps in Dataverse. For more information about many-to-many table relationships, go to Create many-to-many table relationships overview.

View many-to-many table relationships

1. From Power Apps, select **Solutions** on the left navigation pane, and open the solution you want.
2. Open the table that has the relationships you want to view.
3. Select the **Relationships** area to view all relationships configured for the table.

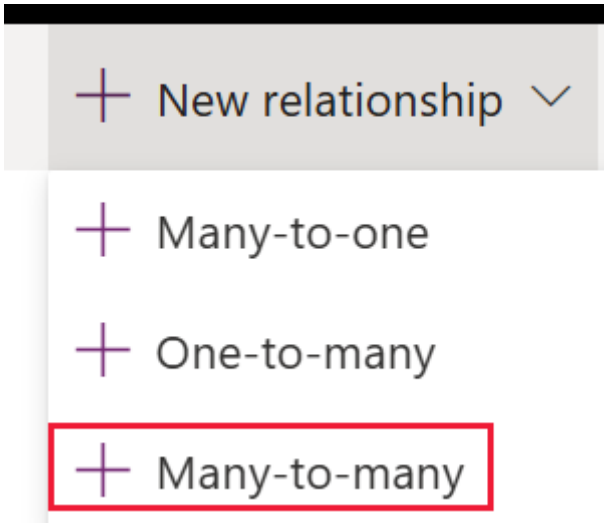


Many-to-many relationships have a **Relationship type** of **Many-to-many**.

> **Note**
>
> The table you view might have no **Many-to-many** relationships.

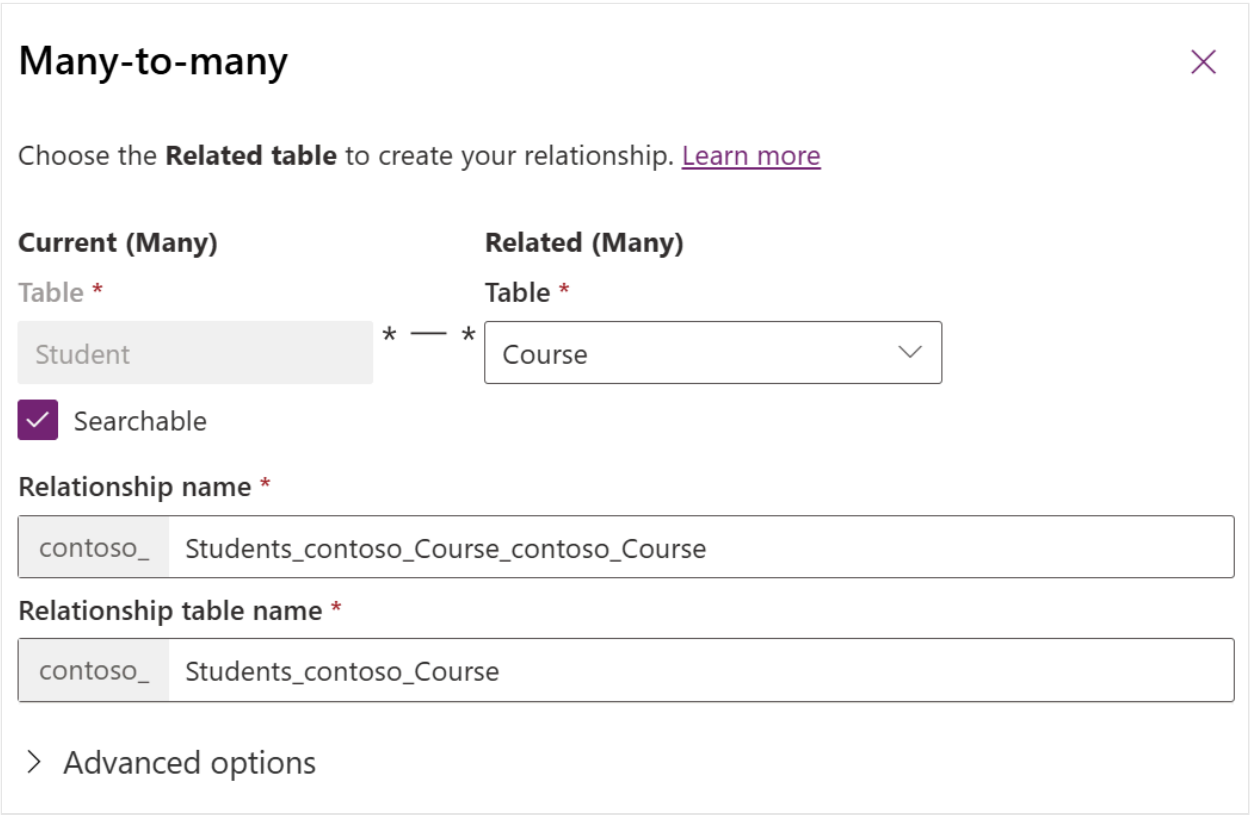
Create relationships

While viewing table relationships, on the command bar, select **New relationship**, and then select **Many-to-many**.



The **Many-to-many relationship** editor is displayed.

In the **Related (Many)** dropdown list, choose the table you want to relate to the current table. In this example, a many-to-many relationship with the student and course tables is created.



The values for these columns are generated for you based on the tables chosen.

Clear **Searchable**, if you want to hide the relationship from **Advanced Find** in model-driven apps.

If you want to edit the display settings for the relationship, expand **Advanced options** and make the changes you want. More information: Advanced options

> **Note**
>
> If you create more than one **Many-to-many** relationship with the same two tables, you must edit the generated **Relationship Name** and **Relationship table name** columns so that they are unique.

Select **Done** to close the **Many-to-many** panel. The relationship is created when you save changes to the table.

Advanced options

For both the **Current table display settings** and the **Related table display settings**, you can edit the display option columns, which control how the related tables are displayed for model-driven apps.

Expand table

Field	Description
Display option	How the related table list should be displayed. More information: Display Options
Custom label	Specify the localizable text to be used instead of the plural name when you select **Use Custom label** as the **Display option** .
Display area	Select one of the available groupings to display this list. The available options are: **Details** (for the *Common* group), **Marketing**, **Sales**, and **Service**.
Display order	Controls where the navigation item will be included within the selected display area. The range of allowed numbers begins with 10,000. Navigation pane items with a lower value appear above other relationships with a higher value.

Display Options

These are the available display options:

Expand table

Option	Description
Do not Display	Do not display the related tables for this relationship.
Use custom label	When this option is chosen the **Custom label** column is enabled so that you can specify the localizable text to be used instead of the plural name.
Use plural name	Use the plural display name defined for the related table.

Create a many-to-many relationship with the same table (self-referential relationship)

There might be times when you need to create a many-to-many relationship that references the same table. For example, account records might have multiple parent account records. However, if the self-referential relationship doesn't have custom labels configured, distinguishing the two related record types that appear in the app isn't apparent to users. This is because the related records appear twice in the app by using the same table name.

To create a self-referential relationship, use custom labels.

1. Open or create a solution and then create a new custom table. In this example, the table is named *Custom table*.

2. Open the custom table, select the **New** > **Relationship** > **New relationship** > **Many-to-many relationship**.

3. Under **Related (Many)** select the same table, in the example *Custom table*, that's also the **Current (Many)** table.

4. Expand **Advanced options** and complete the relationship definition as follows. Add custom labels to both the current and the related table's relationship:

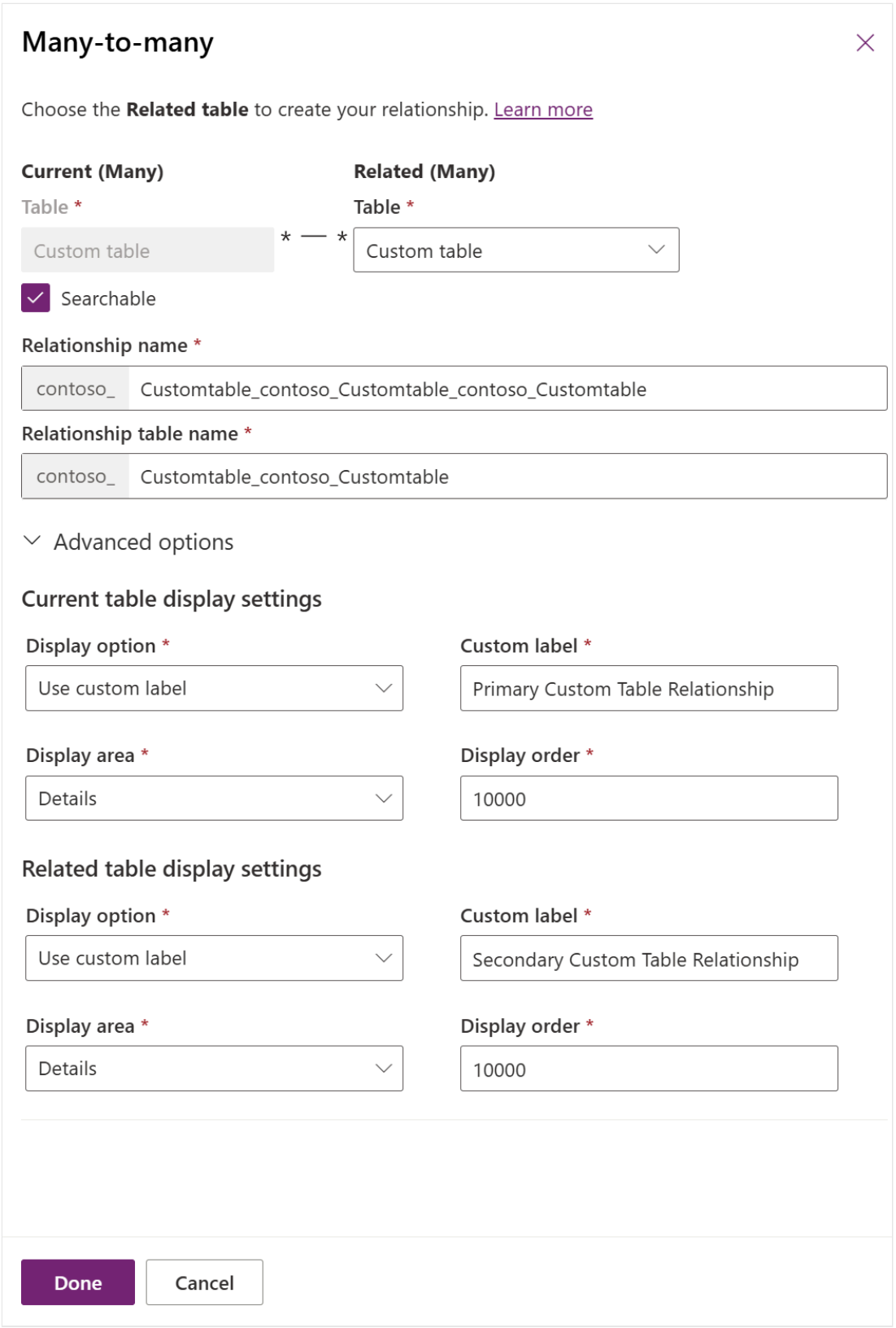
   Current table display settings section

   - **Display option**: **Use custom label**
   - **Custom label**: *Primary Custom Table Relationship*

   Related table display settings section

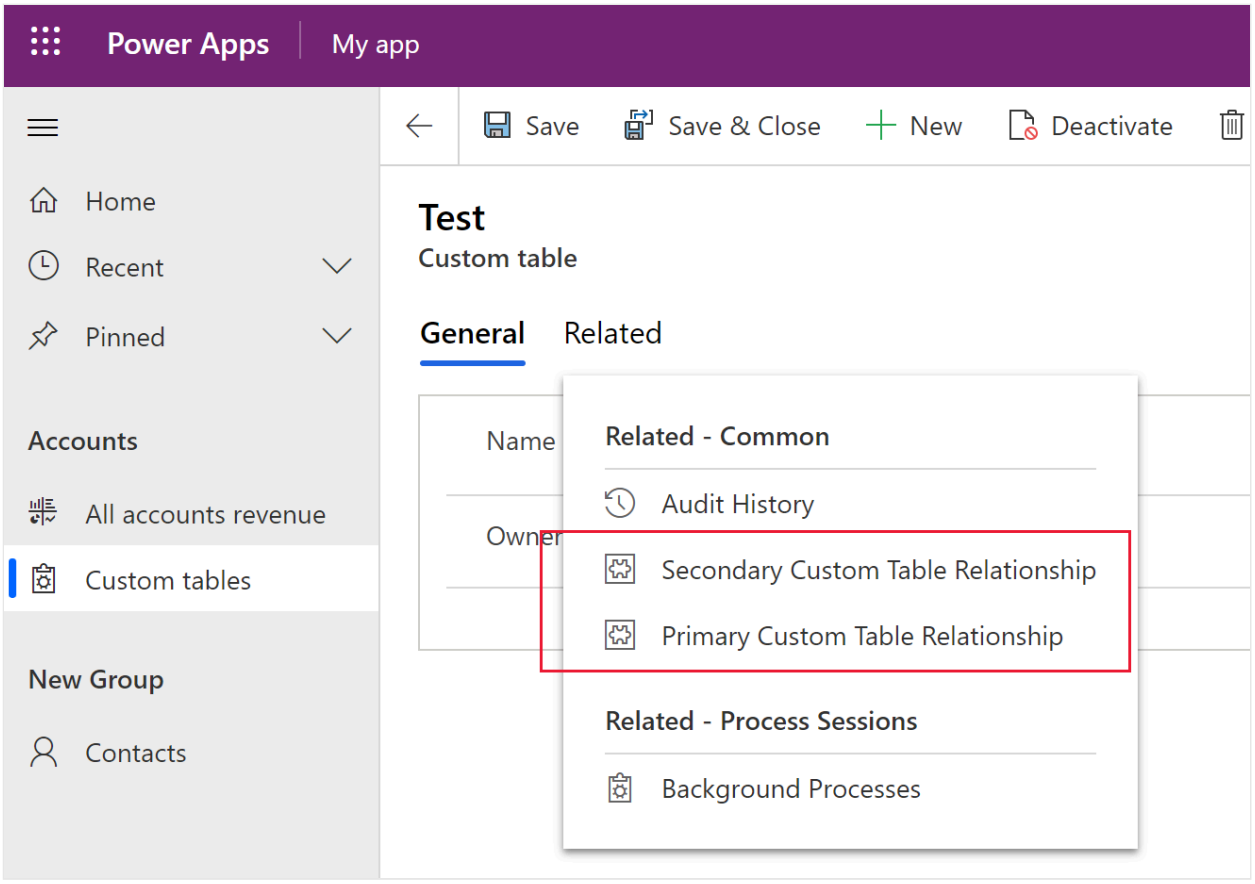
   - **Display option**: **Use custom label**

- **Custom label**: *Secondary Custom Table Relationship*



5. Select **Done** to save the relationship, and then publish the customization.

Because the related record types use the custom labels defined for the self-referential relationship instead of the table name, users running the app can distinguish between the two related record types that exist.



Delete relationships

Deleting a many-to-many relationship deletes the relationship table created. All data connecting tables using the relationship are lost.

While viewing table relationships, select the relationship you want to delete, and then on the command bar select **Remove** > **Delete from this environment**.

See also

Create N:N (many-to-many) table relationships overview

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Feedback

Was this page helpful? Yes No

Provide product feedback

Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships using solution explorer

Article • 02/15/2022

Solution explorer provides one way to create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships for Microsoft Dataverse.

The Power Apps portal enables configuring the most common options, but certain options can only be set using solution explorer. More information:

- Create 1:N (one-to-many) or N:1 (many-to-one) relationships
- Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships in Power Apps portal

Open solution explorer

Part of the name of any custom relationship you create is the customization prefix. This is set based on the solution publisher for the solution you're working in. If you care about the customization prefix, make sure that you are working in an unmanaged solution where the customization prefix is the one you want for this table. More information: Change the solution publisher prefix

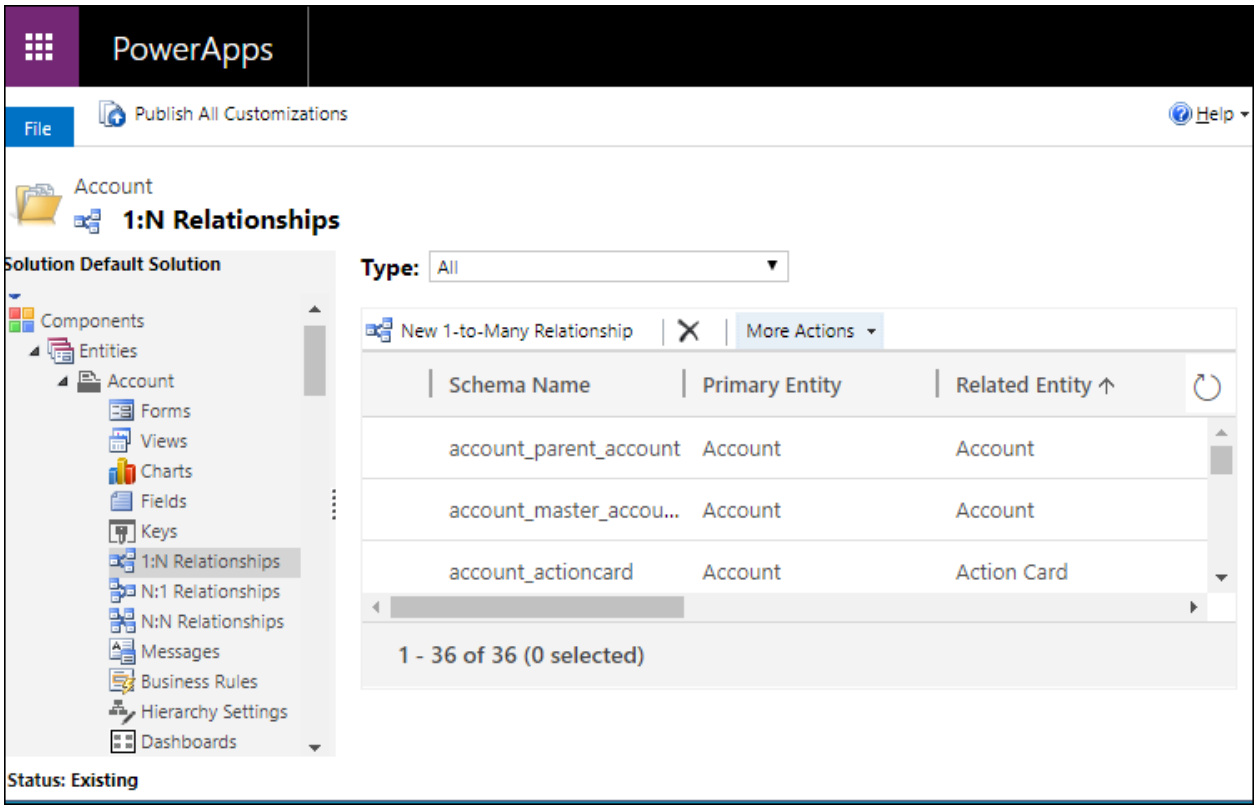
Open an unmanaged solution

1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. In the **All Solutions** list open the unmanaged solution you want.

View table relationships

In solution explorer, expand **Tables** an select a table. Within that table, select either **1:N Relationships** or **N:1 Relationships**

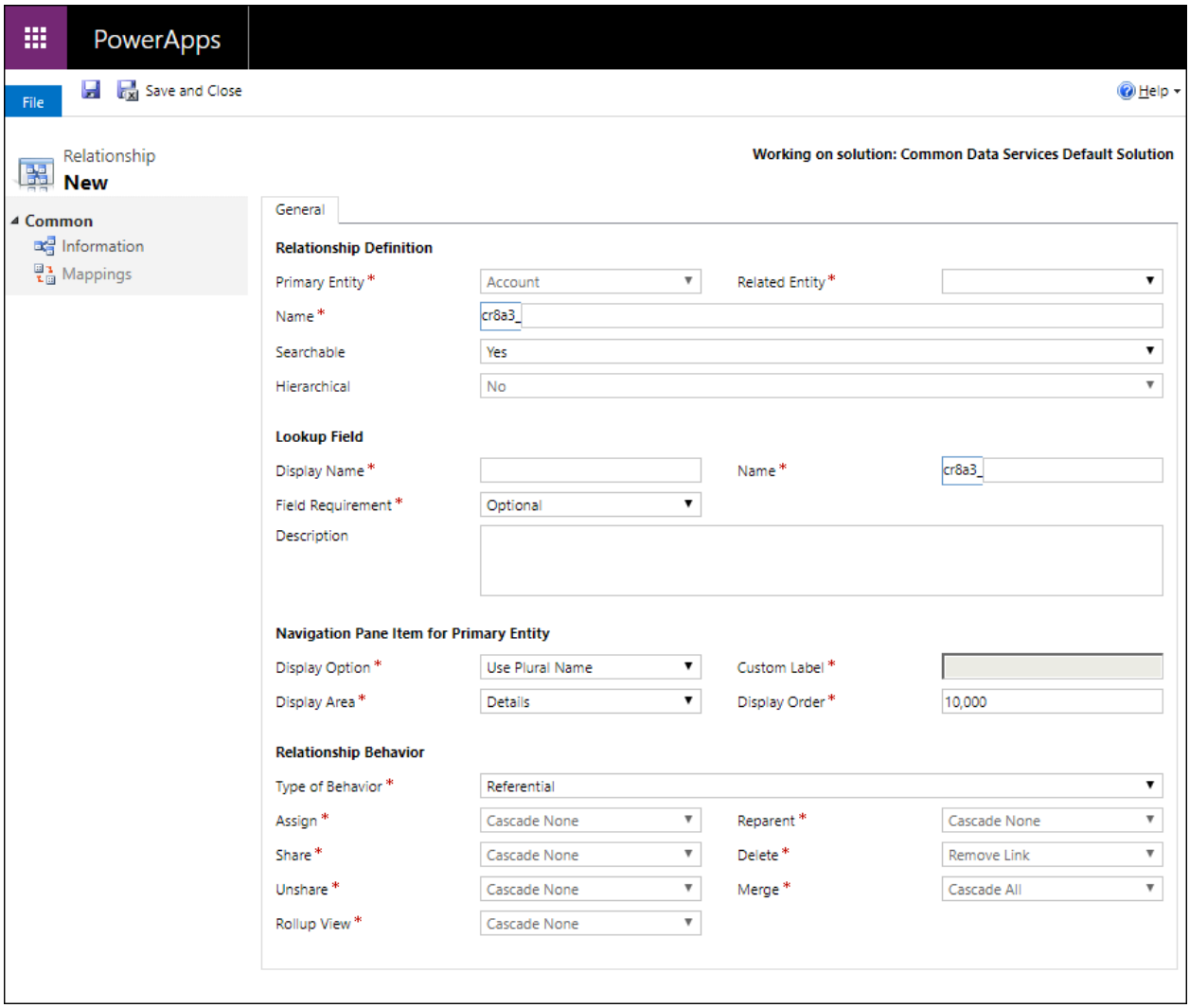


Create relationships

While viewing table relationships, select either **New 1-to-Many Relationship** or **New Many-to-1 Relationship** from the command bar.

> ⓘ **Note**
>
> If the commands are not available, the table is not eligible to create a custom relationship.

Either option will open a form like the following. The difference is whether the **Primary table** or **Related table** column is set.



- With **1:N Relationship**, the **Primary table** is set to the current table
- With **N:1 Relationship**, the **Related table** is set to the current table

The following columns must be set in order to save the table relationship:

Required Column	Description
Primary table	This table will be the target type for the lookup column created on the related table.
Related table	This table will have a lookup column added to associate the table rows with the primary table row.
Name	The name of the relationship. A value will be generated based on the primary and related table values. This column will be prefixed by the customization prefix of the solution publisher.
Lookup Column Display Name	The localizable text for the lookup column that will be created for the related table. This is usually the same as the display name for the primary table. This can be changed later.

Required Column	Description
Lookup Column Name	The name of the lookup column that will be created on the related table. A value will be generated based on the **Lookup Column Display Name**. This column will be prefixed by the customization prefix of the solution publisher.

You can click  to save the table and continue editing. More information: Edit relationships

> ⓘ Note
>
> If either the **Name** or **Lookup Column Name** values already exist in the system you will get an error when you save. Edit the values so that they are unique and try again.

Edit relationships

While viewing table relationships, select the table you want to edit. The following table relationship properties can be edited after the relationship is created.

> ⓘ Note
>
> The publisher of a managed solution can prevent some customizations of relationships that are part of their solution.

Table relationship properties

These properties are about the relationship.

Column	Description
Searchable	Whether this relationship should be visible within Advanced Find in model-driven apps. Select **No** if this is a relationship that isn't important for your business.
Hierarchical	This option is enabled only for self-referential relationships. Whether the table should be considered to define a hierarchy for the table. **Important**: Once you set this property rollup columns, processes, and views may be configured to depend on this property. If you later change this value those capabilities that depend on the hierarchy will not work. More information: Define and query hierarchically related data

Lookup column

These are the properties of the lookup column created on the related table. The properties can be edited here or by editing the lookup column directly. Some column properties are not editable from the relationship. More information: Edit a column

Column	Description
Display Name	The localizable text for the lookup column that will be created for the related table.
Column requirement	Whether the column must have data before a form in a model-driven app can be saved. More information: Column Requirement options
Description	Enter instructions to the user about what the column is for. These descriptions appear as tooltips for the user in model-driven apps when they hover their mouse over the label of the column.

Navigation Pane Item for Primary table

From the primary table you can navigate to see related rows. This data is used by model-driven apps to control how the related table rows are displayed. These settings can also be edited using the form editor.

Column	Description
Display Option	How the related table list should be displayed. More information: Display Options
Custom Label	Specify the localizable text to be used instead of the plural name when you select **Use Custom Label** as the **Display Option** .
Display Area	Select one of the available groupings to display this list. The available options are: **Details** (for the *Common* group), **Marketing**, **Sales**, and **Service**.
Display Order	Controls where the navigation item will be included within the selected display area. The range of allowed numbers begins with 10,000. Navigation pane items with a lower value appear above other relationships with a higher value.

Display Options

These are the available display options:

Option	Description

Option	Description
Do not Display	Do not display the related tables for this relationship.
Use Custom Label	When this option is chosen, the **Custom Label** column is enabled so that you can specify the localizable text to be used instead of the plural name.
Use Plural Name	Use the plural display name defined for the related table.

Relationship Behavior

This where you can define standard behaviors for related tables. This information is important because it helps ensure data integrity and can automate business processes for your company.

Let's look at an example.

Let's say that you have a new salesperson and you want to assign them a number of existing opportunities currently assigned to another salesperson. Each opportunity row may have a number of task activities associated with it. You can easily locate the active opportunities you want to reassign and assign them to the new salesperson. But what should happen for any of the task activities that are associated with the opportunities? Do you want to open each task and decide whether they should also be assigned to the new salesperson? Probably not. Instead, you can let the relationship apply some standard rules for you automatically. These rules only apply to task rows associated to the opportunities you are reassigning. Your options are:

- Reassign all active tasks.
- Reassign all tasks.
- Reassign none of the tasks.
- Reassign all tasks currently assigned to the former owner of the opportunity.

The relationship can control how actions performed on a row for the primary table row cascade down to any related table rows.

There are several kinds of behaviors that can be applied when certain actions occur.

Behaviors

These are the behaviors available to be configured.

Behavior	Description
Cascade Active	Perform the action on all active related table rows.
Cascade All	Perform the action on all related table rows.
Cascade None	Do nothing.
Remove Link	Remove the lookup value for all related rows.
Restrict	Prevent the primary table row from being deleted when related table rows exist.
Cascade User Owned	Perform the action on all related table rows owned by the same user as the primary table row.

Actions

These are the actions that can trigger certain behaviors:

Column	Description	Options
Assign	What should happen when the primary table row is assigned to someone else?	Cascade All Cascade Active Cascade User-owned Cascade None
Reparent	What should happen when the lookup value of a related table in a parental relationship is changed? More information: Parental table relationships	Cascade All Cascade Active Cascade User-owned Cascade None
Share	What should happen when the primary table row is shared?	Cascade All Cascade Active Cascade User-owned Cascade None
Delete	What should happen when the primary table row is deleted?	Cascade All Remove Link Restrict

Column	Description	Options
Unshare	What should happen when a primary table row is unshared?	Cascade All Cascade Active Cascade User-owned Cascade None
Merge	What should happen when a primary table row is merged?	Cascade All Cascade None
Rollup View	What is the desired behavior of the rollup view associated with this relationship?	Cascade All Cascade Active Cascade User-owned Cascade None

Type of Behavior options

Use the **Type of Behavior** column to choose between a set of standard behaviors or whether you want to configure them independently.

Option	Description
Parental	**Assign**: Cascade All **Reparent**: Cascade All **Share**: Cascade All **Delete**: Cascade All **Unshare**: Cascade All **Merge**: Cascade None **Rollup View**: Cascade None | Cascade All
Referential	**Assign**: Cascade None **Reparent**: Cascade None **Share**: Cascade None **Delete**: Remove Link **Unshare**: Cascade None **Merge**: Cascade None **Rollup View**: Cascade None | Cascade All

Option	Description
Referential, Restrict Delete	**Assign**: Cascade None **Reparent**: Cascade None **Share**: Cascade None **Delete**: Restrict **Unshare**: Cascade None **Merge**: Cascade None **Rollup View**: Cascade None | Cascade All
Configurable Cascading	You can configure the behavior you want for each action depending on the options available

> **Note**
>
> You may not be able to choose the **Parental** option if either of the tables already participate in an parental table relationship. More information: **Parental table relationships**
>
> If you use **Configurable Cascading** to set the behaviors for the actions so that they match the behaviors for the actions associated with another **Type of Behavior**, when you save the relationship, the **Type of Behavior** is automatically set to the matching type.

Delete relationships

While viewing table relationships, select the table relationship you want to delete and click the ✕ command.

Deleting the relationship will delete the lookup column on the related table.

> **Note**
>
> You will not be able to delete a relationship that has dependencies. For example, if you have added the lookup column to a form for the related table, you must remove the column from the form before you delete the relationship.

Parental table relationships

Each pair of tables that are eligible to have a 1:N relationship can have multiple 1:N relationships between them. Yet usually only one of those relationships can be

considered a parental table relationship.

A parental table relationship is any 1:N table relationship where one of the cascading options in the **Parental** column of the following table is true.

Action	Parental	Not Parental
Assign	Cascade All Cascade User-owned Cascade Active	Cascade None
Delete	Cascade All	RemoveLink Restrict
Reparent	Cascade All Cascade User-owned Cascade Active	Cascade None
Share	Cascade All Cascade User-owned Cascade Active	Cascade None
Unshare	Cascade All Cascade User-owned Cascade Active	Cascade None

For example, if you create a new custom table and add a 1:N table relationship with the account table where your custom table is the related table, you can configure the actions for that table relationship to use the options in the **Parental** column. If you later add another 1:N table relationship with your custom table as the referencing table you can only configure the actions to use the options in the **Not Parental** column.

Usually this means that for each table pair there is only one parental relationship. There are some cases where the lookup on the related table may allow for a relationship to more than one type of table.

For example, if a table has a Customer lookup that can refer to either a contact or account table. There are two separate parental 1:N table relationships.

Any activity table has a similar set of parental table relationships for tables that can be associated using the regarding lookup column.

Limitations on behaviors you can set

Because of parental relationships there are some limitations you should keep in mind when you define table relationships.

- A custom table can't be the primary table in a relationship with a related system table that cascades. This means you can't have a relationship with any action set to **Cascade All**, **Cascade Active**, or **Cascade User-Owned** between a primary custom table and a related system table.
- No new relationship can have any action set to **Cascade All**, **Cascade Active**, or **Cascade User-Owned** if the related table in that relationship already exists as a related table in another relationship that has any action set to **Cascade All**, **Cascade Active**, or **Cascade User-Owned**. This prevents relationships that create a multi-parent relationship.

See also

Create and edit relationships between tables
Create and edit 1:N (one-to-many) or N:1 (many-to-one) relationships
Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships in Power Apps portal
Create N:N (many-to-many) relationships

Create N:N (many-to-many) table relationships in Microsoft Dataverse using solution explorer

Article • 02/15/2022

Solution explorer provides one way to create and edit N:N (many-to-many) for Dataverse.

The Power Apps portal enables configuring the most common options, but certain options can only be set using solution explorer. More information:

- Create Many-to-many (N:N) table relationships
- Create Many-to-many table relationships in Dataverse using Power Apps portal

Open solution explorer

Part of the name of any custom relationship you create is the customization prefix. This is set based on the solution publisher for the solution you're working in. If you care about the customization prefix, make sure that you are working in an unmanaged solution where the customization prefix is the one you want for this table. More information: Change the solution publisher prefix

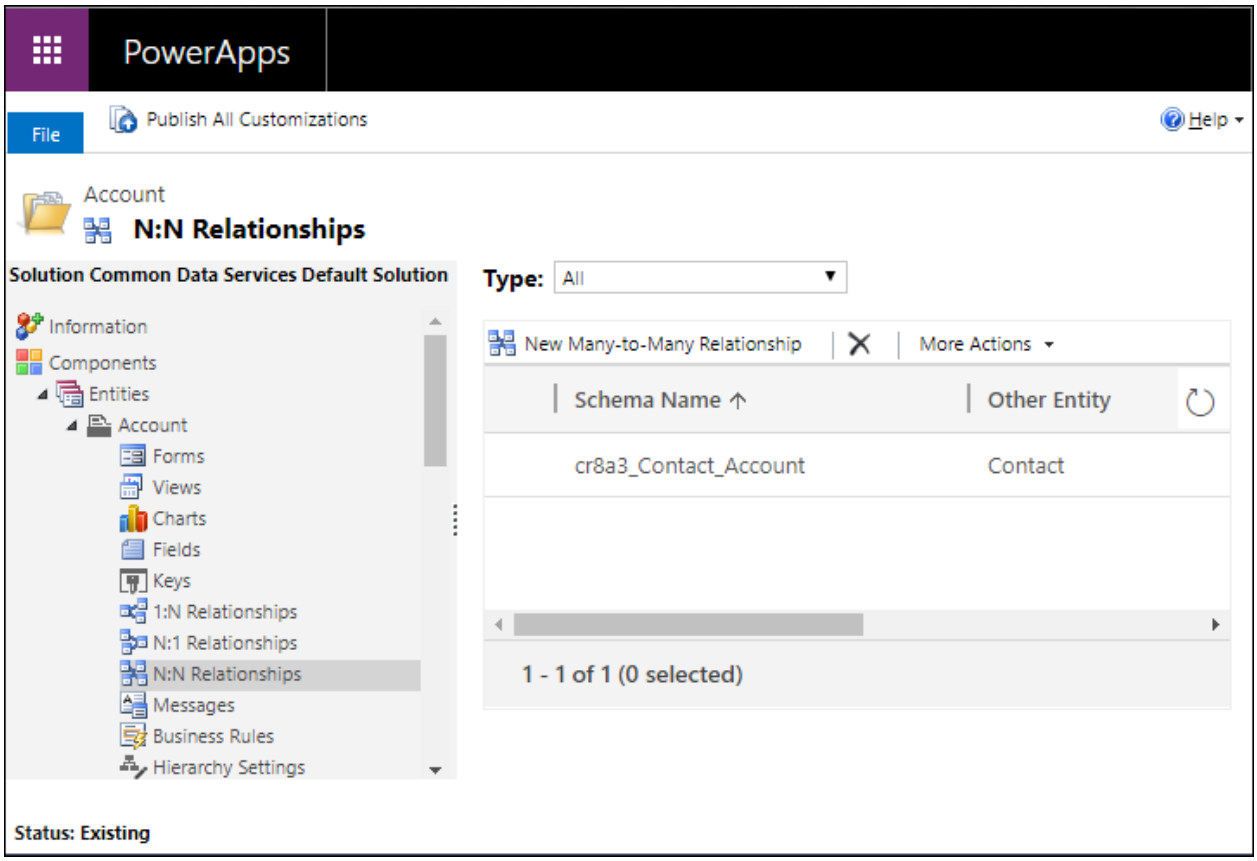
Open an unmanaged solution

1. From Power Apps select **Solutions** from the left navigation pane, and then on the toolbar, select **Switch to classic**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. In the **All Solutions** list open the unmanaged solution you want.

View table relationships

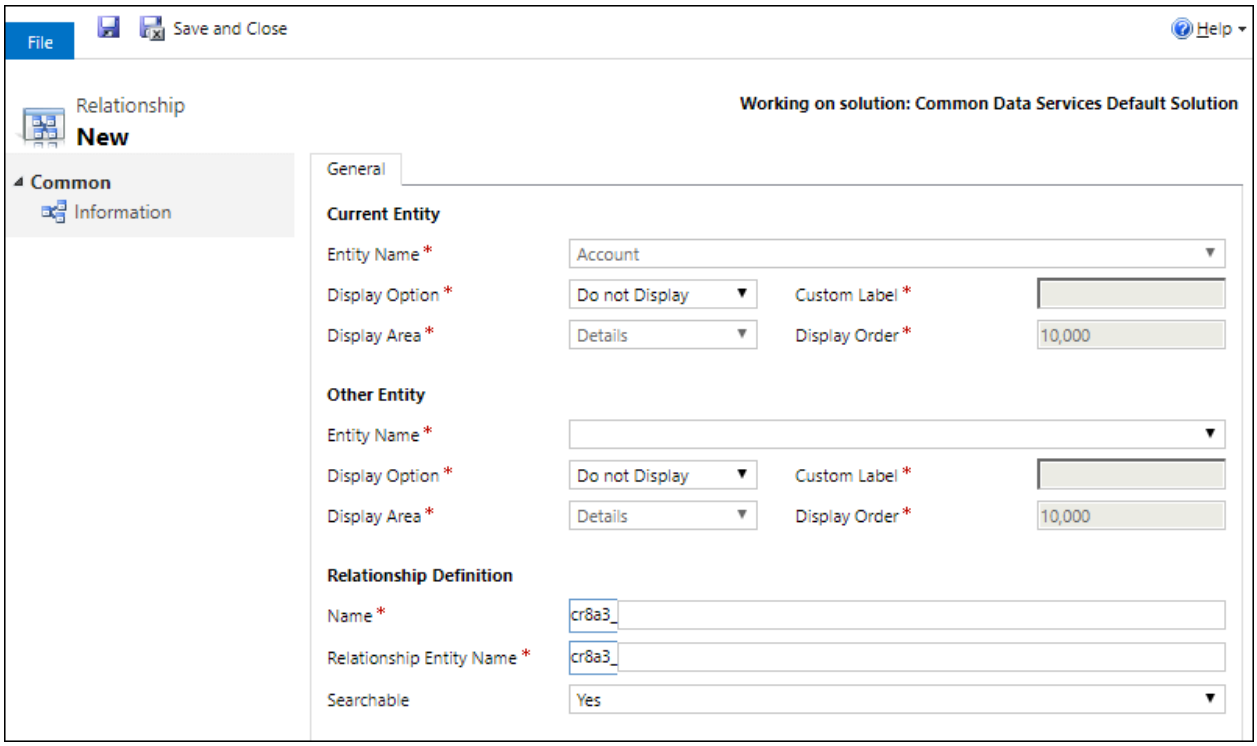
In solution explorer, expand **Tables** an select a table. Within that table, select **N:N Relationships**.



Create relationships

While viewing table relationships, select **New Many-to-Many Relationship** from the command bar.

> **Note**
>
> If the command is not available, the table is not eligible to create a custom relationship.



In the **Other table** group, in the **table Name** column, choose the table that you want to create the relationship with. This will populate the **Name** and **Relationship table Name** columns in the **Relationship Definition** group.

You can click to save the table and continue editing. More information: Edit relationships

> ⓘ Note
>
> If either the **Name** or **Relationship table Name** values already exist in the system you will get an error when you save. Edit the values so that they are unique and try again.

Edit relationships

While viewing table relationships, select the table you want to edit.

> ⓘ Note
>
> The publisher of a managed solution can prevent customizations of relationships that are part of their solution.

The following table relationship properties can be edited after the relationship is created.

> **ⓘ Important**
>
> After you edit these properties you must publish customizations before they will take effect in model-driven apps.

Edit Display options

For both the **Current table** and the **Other table**, you can edit the display option columns which control how the related tables are displayed for model-driven apps.

Field	Description
Display Option	How the related table list should be displayed. More information: Display Options
Custom Label	Specify the localizable text to be used instead of the plural name when you select **Use Custom Label** as the **Display Option** .
Display Area	Select one of the available groupings to display this list. The available options are: **Details** (for the *Common* group), **Marketing**, **Sales**, and **Service**.
Display Order	Controls where the navigation item will be included within the selected display area. The range of allowed numbers begins with 10,000. Navigation pane items with a lower value appear above other relationships with a higher value.

Display Options

These are the available display options:

Option	Description
Do not Display	Do not display the related tables for this relationship.
Use Custom Label	When this option is chosen, the **Custom Label** column is enabled so that you can specify the localizable text to be used instead of the plural name.
Use Plural Name	Use the plural display name defined for the related table.

Searchable

You can hide the relationship from **Advanced Find** in model-driven apps by setting the **Searchable** column to **No**.

Delete relationships

While viewing table relationships, select the table relationship you want to delete and click the ✕ command.

Deleting the relationship will delete the relationship table created. All data connecting tables using the relationship will be lost.

See also

Create Many-to-many (N:N) table relationships
Create Many-to-many table relationships in Dataverse using Power Apps portal
Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships

Set managed properties for relationships

Article • 09/19/2024

Managed properties apply when you include a column with a managed solution and import it into another organization. These settings allow a solution maker to have some control over the level of customization that they want to allow people who install their managed solution to have when they customize a table relationship.

Set managed properties for a relationship

1. Sign in to Power Apps ,and select the environment you want.
2. Select **Solutions**, open the solution you want, and then open the table that has the relationship where you want to view the managed properties.
3. Select the relationship, and then select **Advanced** > **Managed properties** on the command bar.

With relationships, the only managed property is **Allow customizations**. This single setting controls all changes that can be made to the table relationship.

See also

Create and edit relationships between tables

Solutions in Power Apps overview

Feedback

Was this page helpful? Yes No

Provide product feedback

Query and visualize hierarchically related data

> **Important**
>
> This feature is deprecated. For more information, go to Hierarchy control in model-driven apps is deprecated.

Get valuable business insights by visualizing hierarchically related data. The hierarchical modelling and visualization capabilities give you a number of benefits:

- View and explore complex hierarchical information.
- View key performance indicators (KPIs) in the contextual view of a hierarchy.
- Visually analyze key information across the web and the tablets.

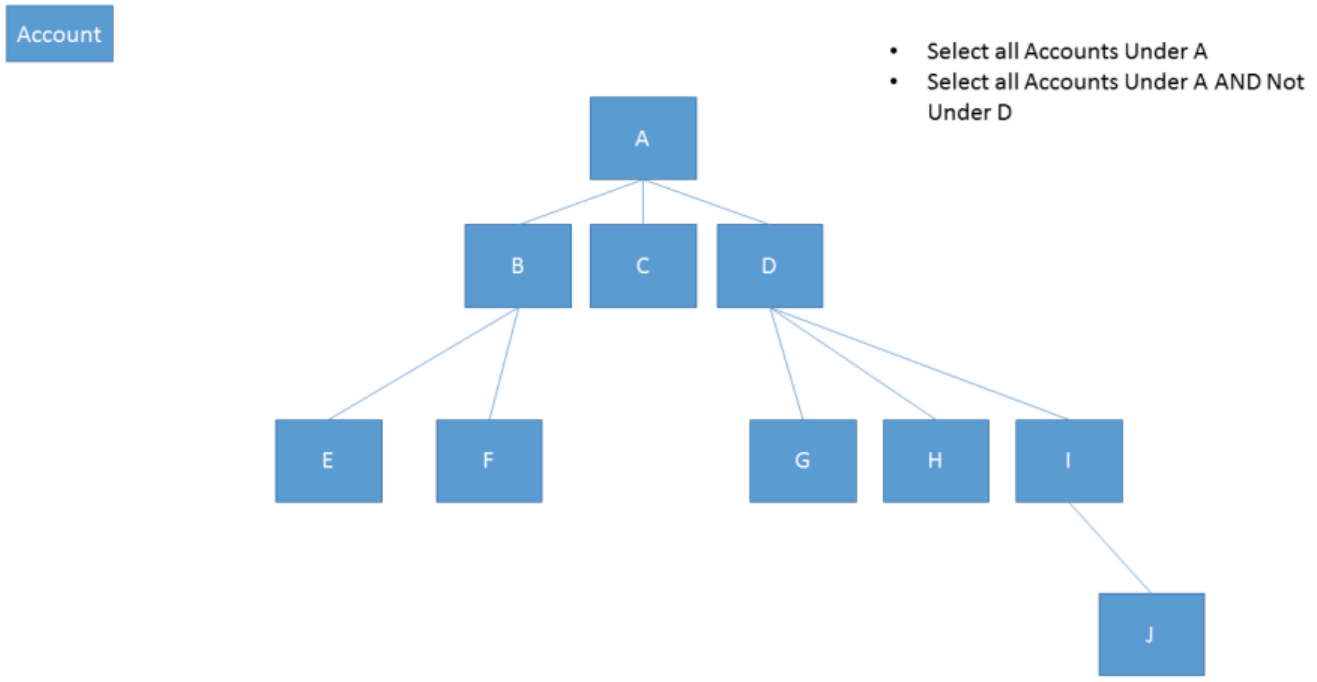
For some tables, such as account and user, the visualizations are provided out-of-the-box. Other tables, including custom tables, can be enabled for a hierarchy and you can create the visualizations for them. Based on your needs, you can choose between using a tree view, which shows the entire hierarchy, or a tile view, which depicts a smaller portion of the hierarchy. Both views are shown side by side. You can explore a hierarchy by expanding and contracting a hierarchy tree. The same hierarchical settings for visualization are set once, but apply to both web and mobile clients. In tablets, the visuals render in a modified format suitable for the smaller form factor. The customizable components required for hierarchical visualization are solution aware, therefore, they can be transported between organizations like any other customization. You can configure the attributes shown in the visualization by customizing a Quick Form using the form editor. There is no requirement to write code.

Query hierarchical data

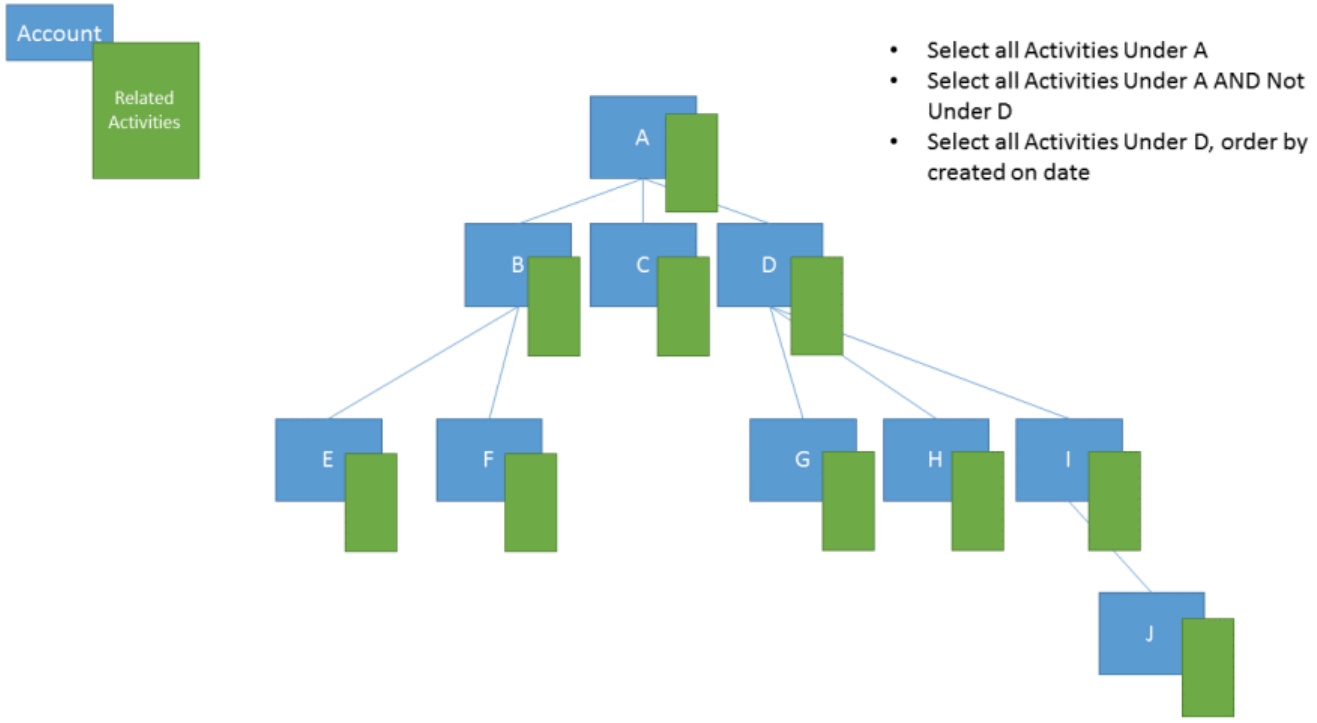
With Microsoft Dataverse, hierarchical data structures are supported by self-referential relationships of the related rows. In the past, to view hierarchical data, you had to iteratively query for the related rows. Presently, you can query the related data as a hierarchy, in one step. You'll be able to query rows using the **Under** and **Not Under** logic. The **Under** and **Not Under** hierarchical operators are exposed in Advanced Find and the workflow editor. For more information about how to use these operators, see Configure workflow steps. For more information about Advanced Find, see Create, edit, or save an Advanced Find search

The following examples illustrate various scenarios for querying hierarchies:

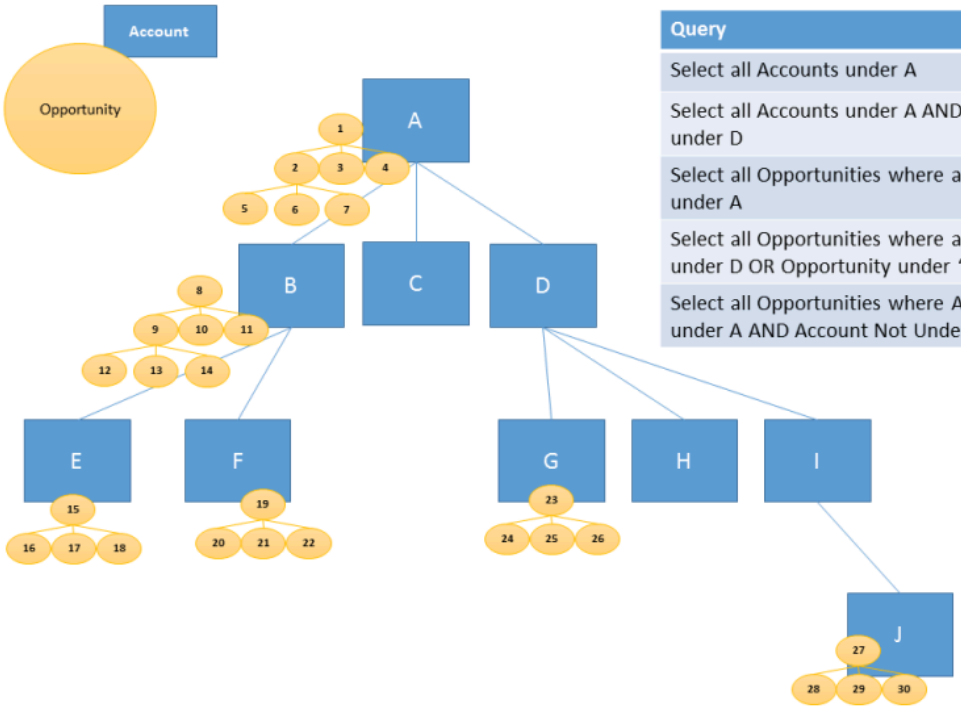
Query account hierarchy



Query account hierarchy, including related activities



Query account hierarchy, including related opportunities



Query	Result
Select all Accounts under A	{B,C,D,E,F,G,H,I,J}
Select all Accounts under A AND not under D	{B,C,D,E,F,}
Select all Opportunities where account under A	{8,9,10.... 30}
Select all Opportunities where account under D OR Opportunity under "15"	{23,24,25,26,27,28,29,30, 16,17,18}
Select all Opportunities where Account under A AND Account Not Under D	{8,9,10,11,12.....,22}

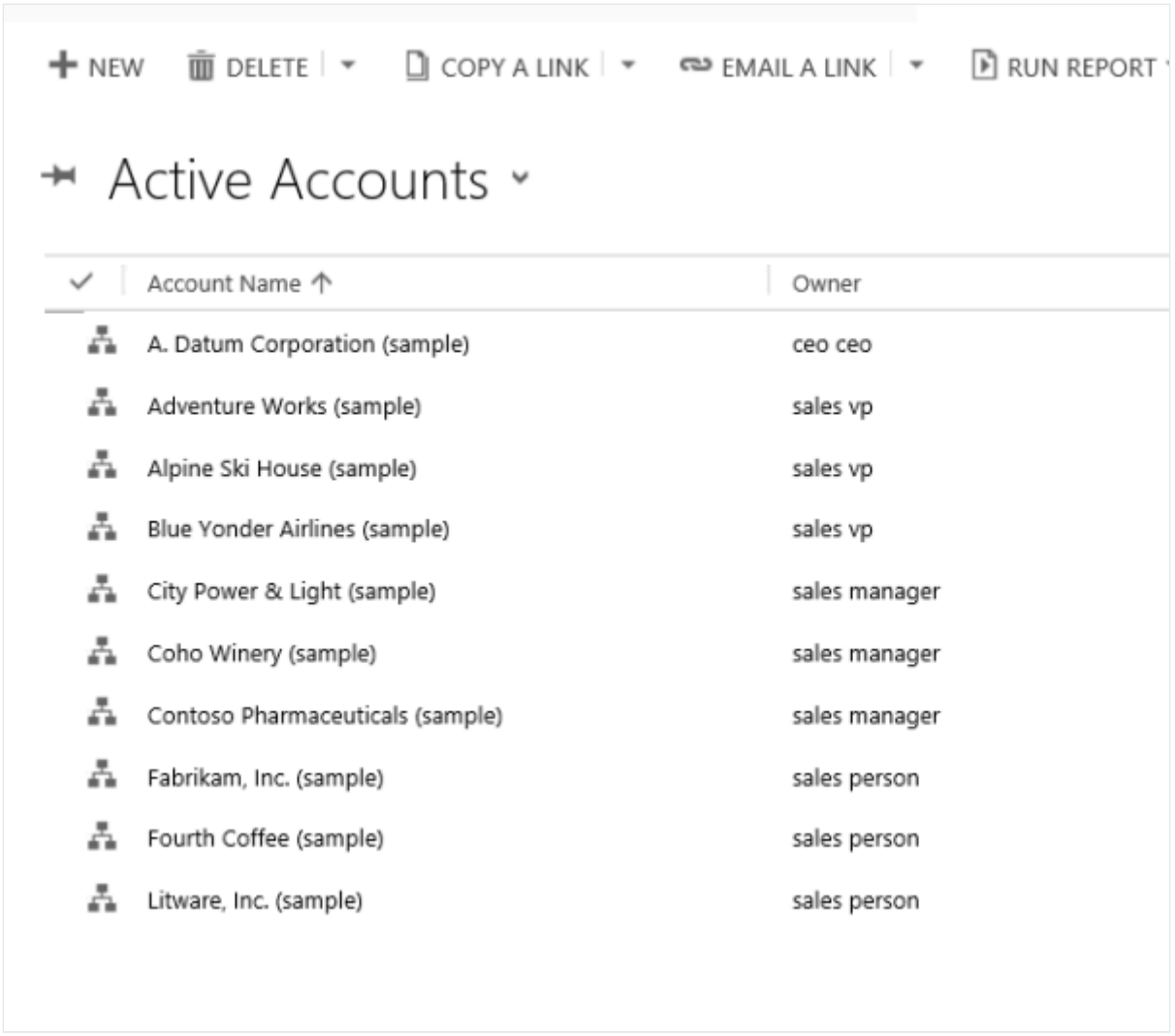
To query the data as a hierarchy, you must set one of the table's one-to-many or many-to-one self-referential relationships as hierarchical. To turn the hierarchy on:

1. On powerapps.com, and then select **Tables** in the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select an existing table, or Create a new table
3. Select **Relationships**, and then select a self-referential relationship.
4. In the relationship details panel, check **Hierarchical**.

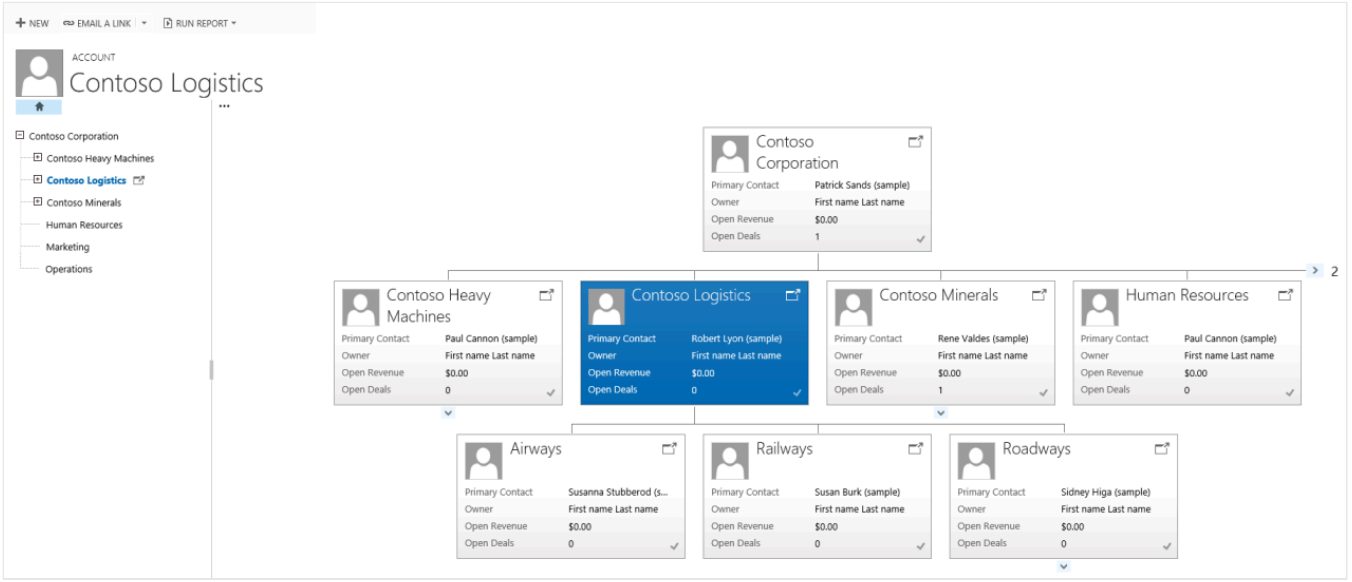
> **Note**
>
> - Some of the out-of the-box relationships can't be customized. This will prevent you from setting those relationships as hierarchical.
> - You can specify a hierarchical relationship for the system self-referential relationships. This includes self-referential relationships of system type, such as the "contact_master_contact" relationship.

Visualize hierarchical data

The system tables that have visualizations available out-of-the-box include `Account`, `Business Unit`, `Position`, `Product`, and `User`. In the grid view of these tables, you can see the icon depicting the hierarchy chart, to the left of the row name. The hierarchy icon isn't present for all rows by default. The icon is shown for the rows that have a parent row, a child row, or both.



If you select the hierarchy icon, you can view the hierarchy, with the tree view on the left and the tile view on the right, as shown below:



A few other out-of the-box system tables can be enabled for a hierarchy. These tables include `Case`, `Contact`, `Opportunity`, `Order`, `Quote`, `Campaign`, and `Team`. All custom tables can be enabled for a hierarchy.

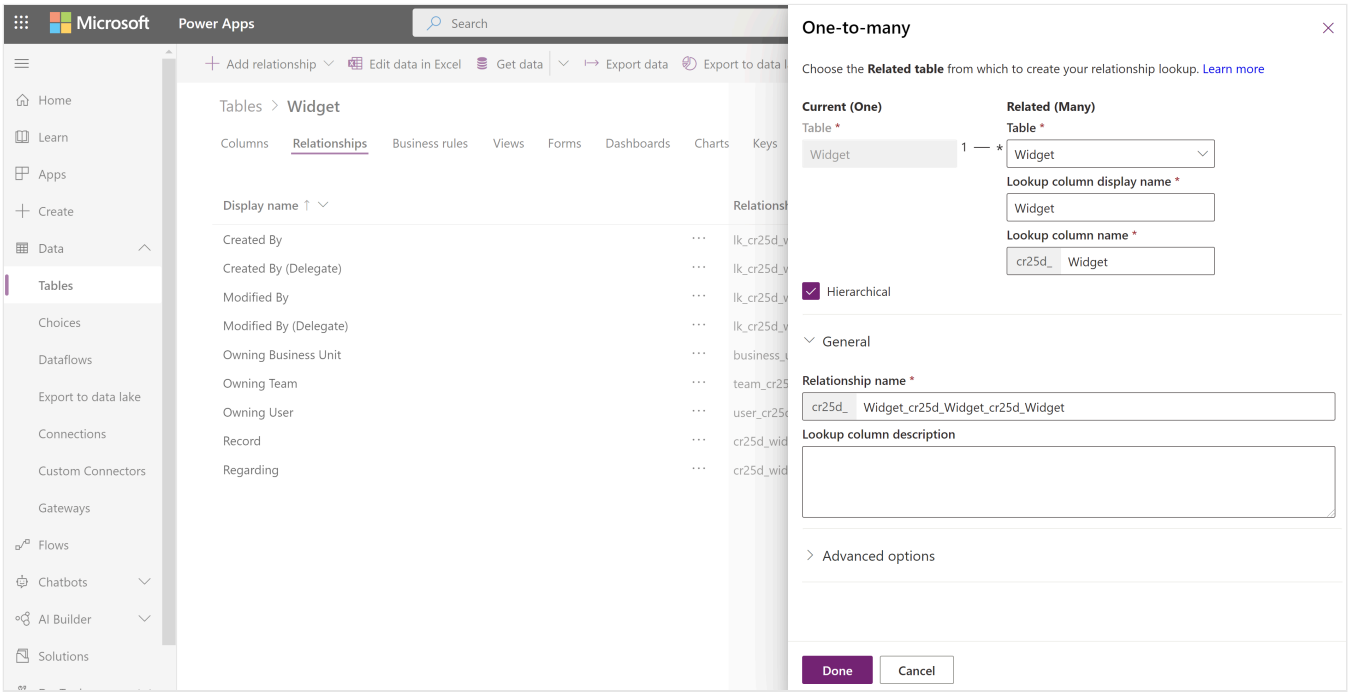
> **Tip**
>
> If a table can be enabled for a hierarchy:
> In solution explorer, expand the table that you want. You will see the table component called **Hierarchy Settings**. The tables that can't be enabled for a hierarchy don't have this component, with the exception of the Dynamics 365 Sales Territory table. Although **Hierarchy Settings** appears for the Sales Territory table, the table can't be enabled for a hierarchy.

Important things to remember when you create visualizations:

- Only one (1: N) self-referential relationship per table can be set as hierarchical. In this relationship the primary table and the related table must be of the same type, such as account_parent_account or Widget_new_Widget_new_Widget.
- Presently, a hierarchy or visualization is based on one table only. You can depict the account hierarchy showing accounts at multiple levels, but you can't show accounts and contacts in the same hierarchy visualization.
- Maximum number of columns that can be displayed in a tile is four. If you add more columns to the Quick Form that is used for the tile view, only the first four columns will be displayed.

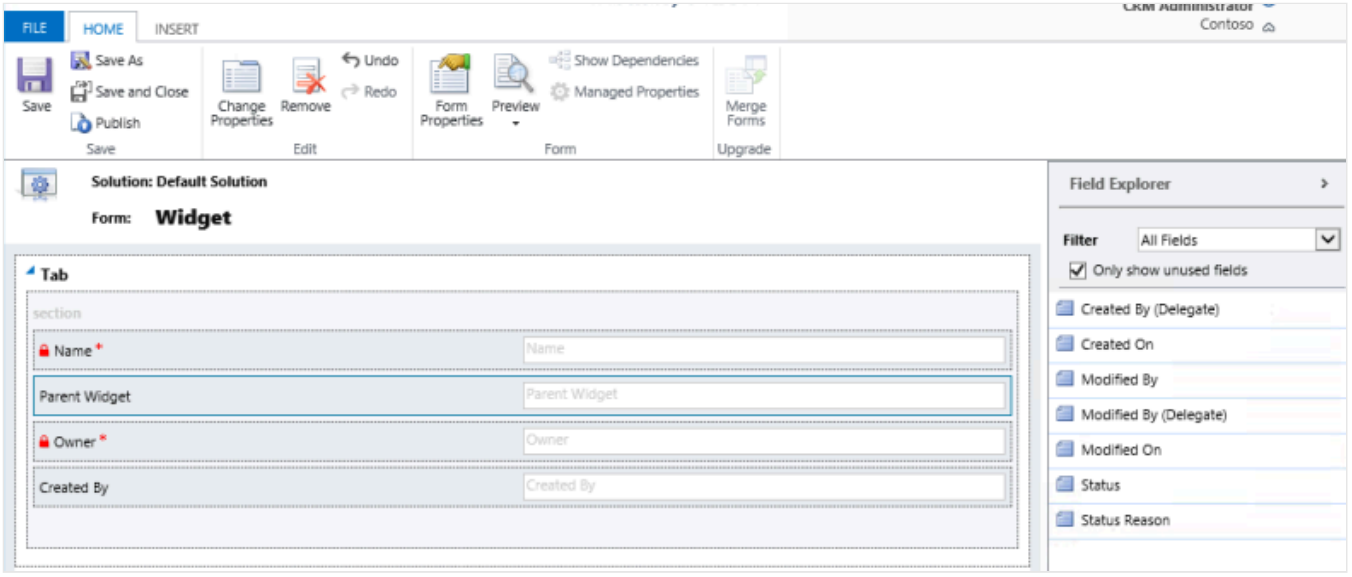
Visualization example

Let's look at an example of creating the visualization for a custom table. We created a custom table called new_Widget, created a self-referential relationship and marked it as hierarchical, as shown here.

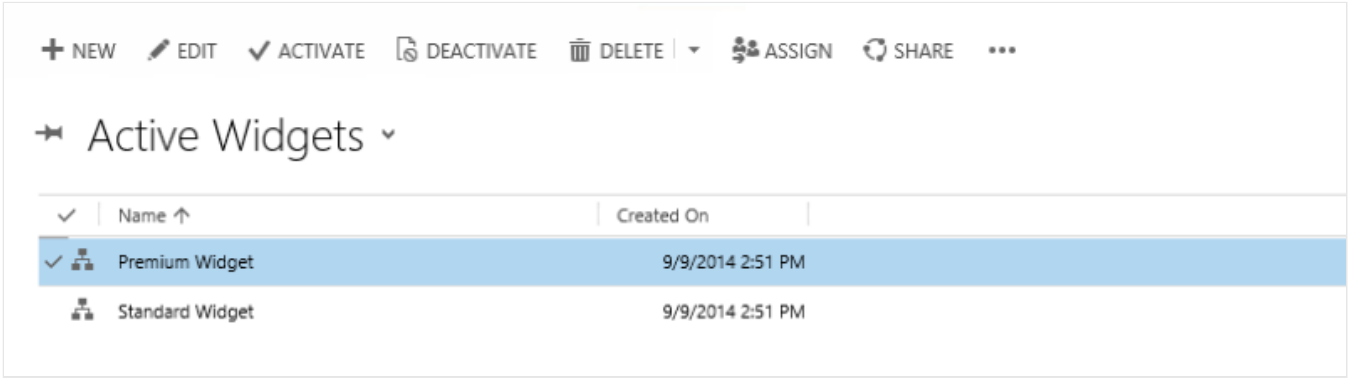


Next, in the **Hierarchy Settings** grid view, we selected the **Widget_new_Widget_new_Widget** hierarchical relationship. In the form, we filled in the required columns. If you haven't yet marked the relationship as hierarchical, the link on the form will take you to the classic table editor, where you can also mark the relationship as hierarchical.

For the **Quick View Form**, we created a Quick Form called **Widget Hierarchy Tile Form**. In this form, we added four columns to display in each tile.

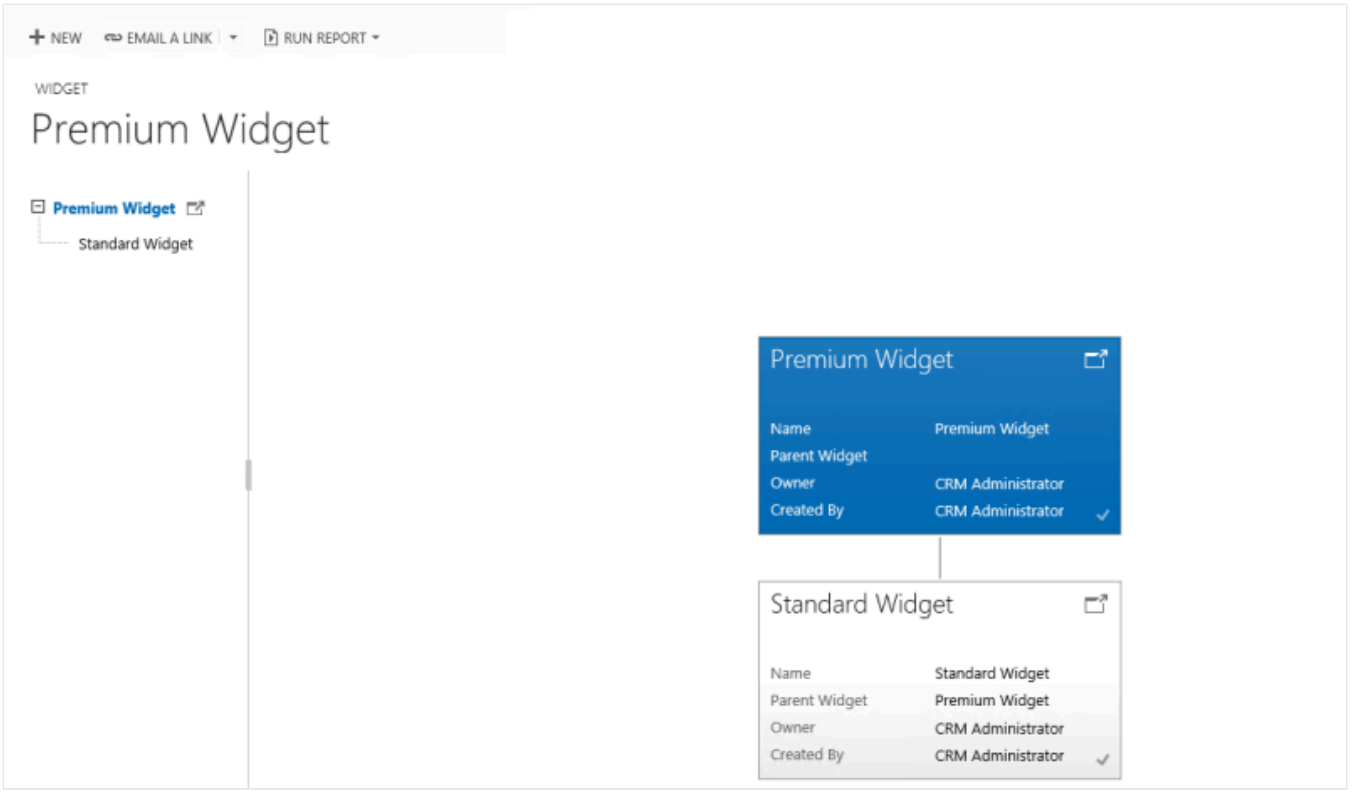


After we completed the setup, we created two rows: Standard Widget and Premium Widget. After making the Premium Widget a parent of the Standard Widget by using the lookup column, the new_Widget grid view depicted the hierarchy icons, as shown below:



> **Tip**
>
> The hierarchy icons don't appear in the row grid view until the rows are paired in the parent – child relationship.

Choosing the hierarchy icon displays the new_Widget hierarchy with the tree view on the left and the tile view on the right, showing two rows. Each tile contains four columns that we provided in the **Widget Hierarchy Tile Form**.



Last updated on 03/31/2026

Define and query hierarchically related data

Article • 08/03/2024

> **Important**
>
> This feature is deprecated. For more information, go to Hierarchy control in model-driven apps is deprecated.

You can get valuable business insights by defining and querying hierarchically related data. The hierarchical modeling and visualization capabilities give you a number of benefits:

- View and explore complex hierarchical information.
- View key performance indicators (KPIs) in the contextual view of a hierarchy.
- Visually analyze key information across the web and the tablets.

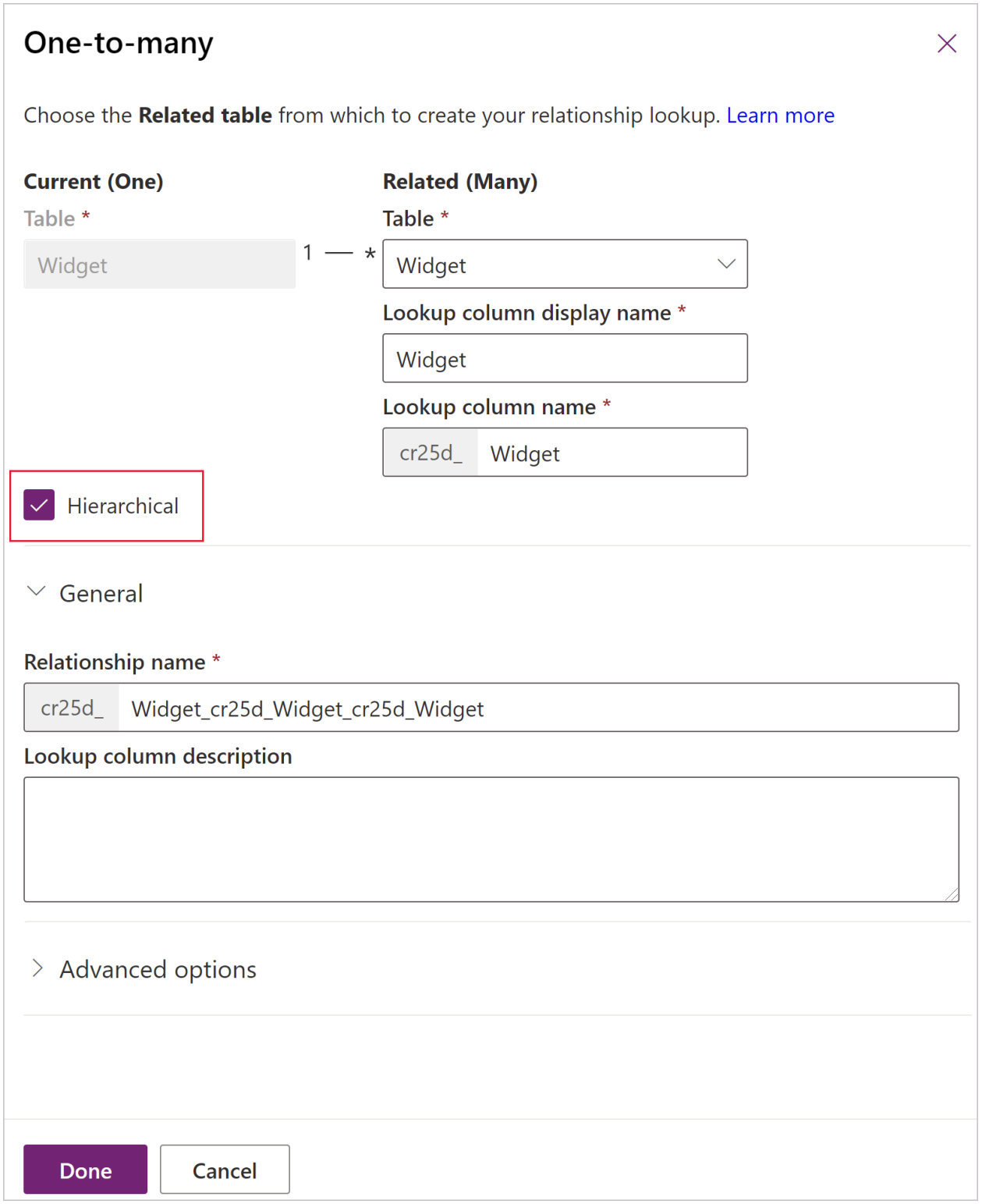
Some standard tables already have hierarchies defined. Other tables, including custom tables, can be enabled for a hierarchy and you can create the visualizations for them.

Define hierarchical data

With Microsoft Dataverse, hierarchical data structures are supported by *self-referential* one-to-many (1:N) relationships of the related rows.

> **Note**
>
> *Self-referential* means that the table is related to itself. For example, the account table has a lookup column to associate it with another account table row.

When a self-referential one-to-many (1:N) relationship exists, in the relationship definition the **Hierarchical** option is available to be set to **Yes**.



To query the data as a hierarchy, you must set one of the table's one-to-many (1:N) self-referential relationships as hierarchical.

To turn on the hierarchy:

1. While viewing 1:N relationships, select the self-referential relationship you want to edit.
2. In the relationship definition, set **Hierarchical** to **Yes**.

> **Note**

- Some of the out-of the-box (1:N) relationships can't be customized. This will prevent you from setting those relationships as hierarchical.
- You can specify a hierarchical relationship for the system self-referential relationships. This includes the 1:N self-referential relationships of system type, such as the "contact_master_contact" relationship.

ⓘ Important

You can have multiple self-referential relationships, but only one relationship per table can be defined as the hierarchical relationship. If you try to change the setting once applied you will get a warning:

- **When disabling:** If you turn off the hierarchy setting for this relationship, all rollup definitions, processes, and views that use this hierarchy won't work. Do you want to continue?
- **When enabling:** If you enable the hierarchy setting for this relationship, all rollup definitions that use the existing hierarchy will become invalid. Do you want to continue?

Unless you are certain that there are no other dependencies on the existing hierarchy, you should review any documentation about the deployment or confer with other customizers to understand how the existing hierarchical relationship is used before continuing.

Query hierarchical data

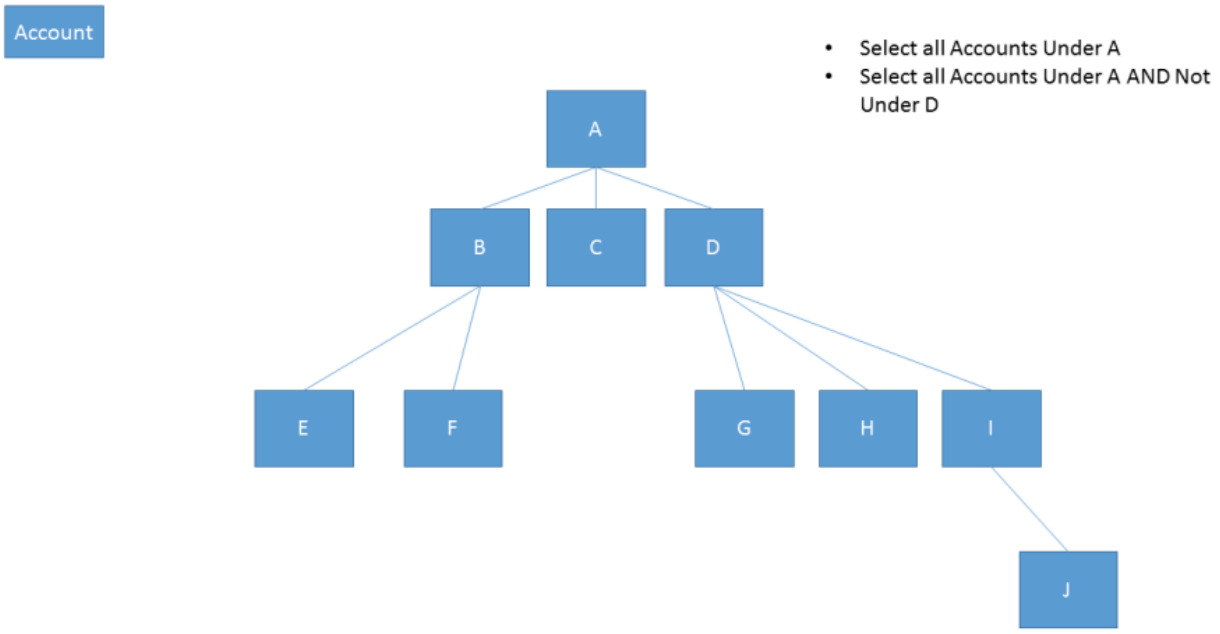
Without a defined hierarchy, to retrieve hierarchical data, need to iteratively query for the related rows. With a defined hierarchy, you can query the related data as a hierarchy in one step. You're able to query rows using the **Under** and **Not Under** logic. The **Under** and **Not Under** hierarchical operators are exposed in Advanced Find and the workflow editor. For more information about how to use these operators, see Configure workflow steps. For more information about Advanced Find, see Create, edit, or save an Advanced Find search

ⓘ Note

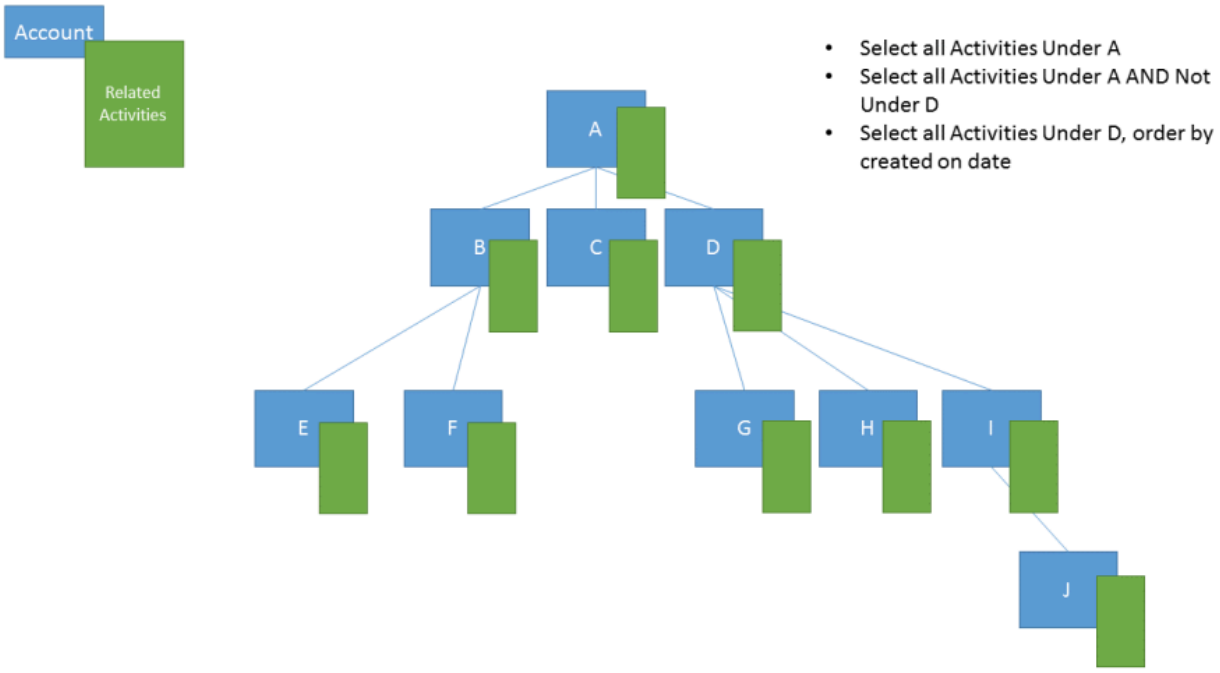
Developers will also be able to use these operators in code. More information Developer Documentation: Query hierarchical data

The following examples illustrate scenarios for querying hierarchies:

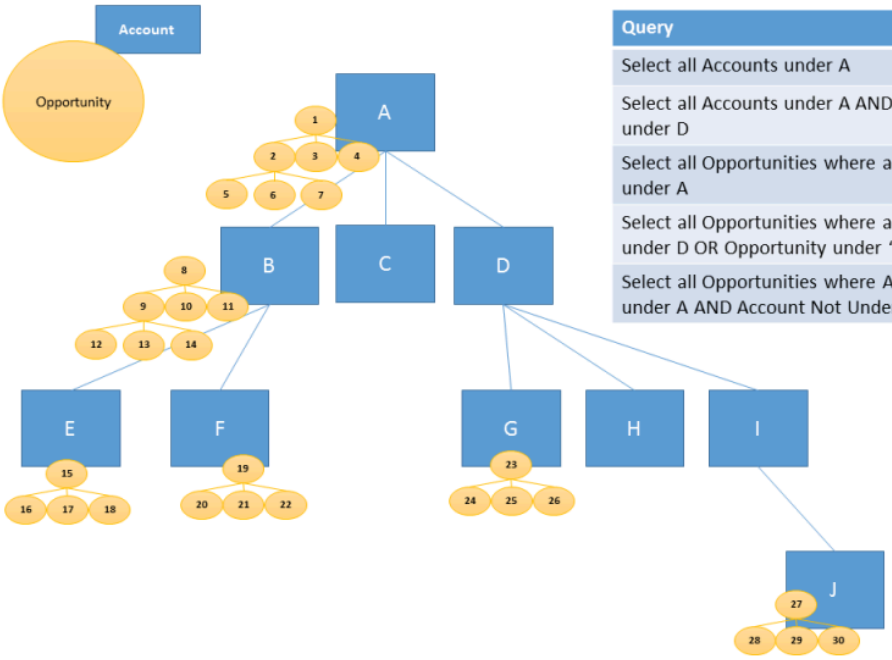
Query account hierarchy



Query account hierarchy, including related activities



Query account hierarchy, including related opportunities



Query	Result
Select all Accounts under A	{B,C,D,E,F,G,H,I,J}
Select all Accounts under A AND not under D	{B,C,D,E,F,}
Select all Opportunities where account under A	{8,9,10.... 30}
Select all Opportunities where account under D OR Opportunity under "15"	{23,24,25,26,27,28,29,30, 16,17,18}
Select all Opportunities where Account under A AND Account Not Under D	{8,9,10,11,12.....,22}

See also

Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships
Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships using solution explorer
Visualize hierarchical data with model-driven apps

Feedback

Was this page helpful? Yes No

Provide product feedback

Configure connection roles

Article • 03/13/2023

With Microsoft Dataverse, you can define **connections** between table rows without creating a table relationship. In model-driven apps people can establish a named link between rows to establish less a formal relationship, which doesn't justify creating an actual table relationship. Some examples include *friend*, *sibling*, *spouse*, *attendee*, and *stakeholder*. Some connections can also be reciprocal, such as *child* and *parent*, *husband* and *wife*, or *doctor* and *patient*.

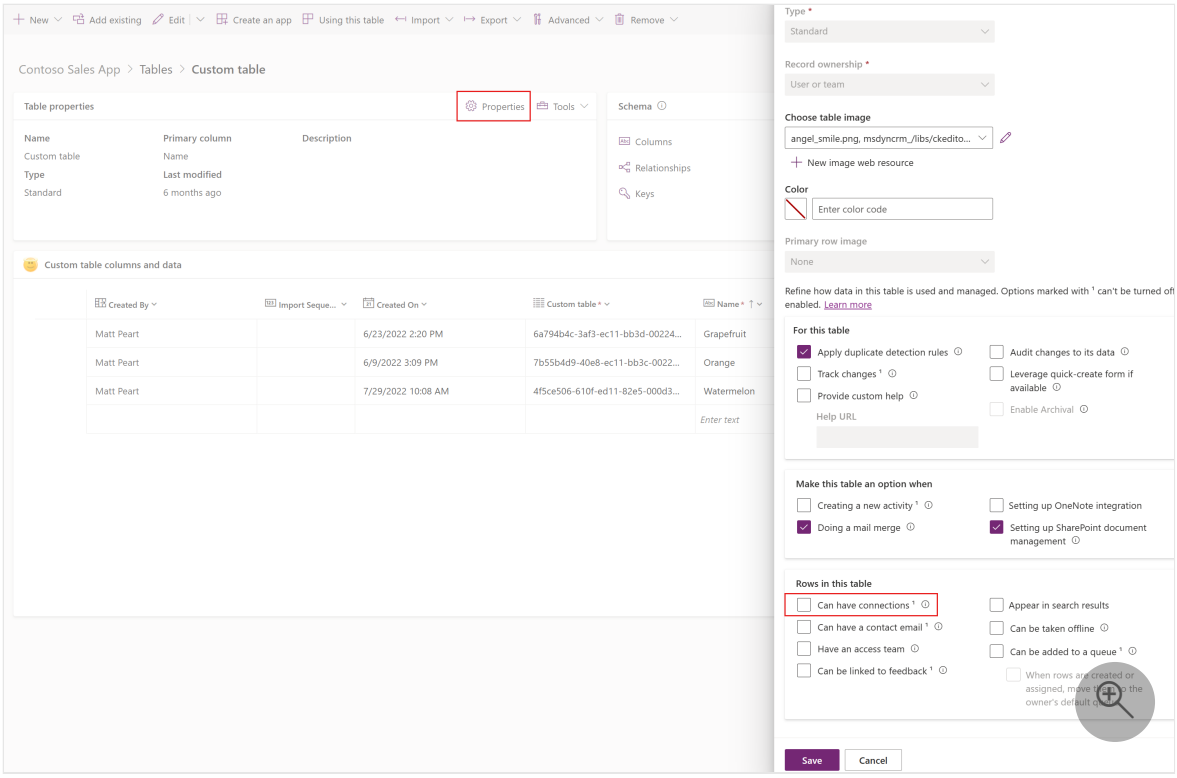
When people set a connection between two rows, they can also add a description and additional information such as start and end dates for the relationship. More information: Add a connection role to link rows to each other

Anyone with write access to the **Connection Role** table can establish which connections are available for people to use.

> **Important**
>
> For a table to be available as a row type for a new or existing connection role, the **Can have connections** property must be enabled for the table.

Enable connection roles for a table

1. Sign in to Power Apps.
2. Open the table you want or create a new one. More information: Edit a table
3. Select **Properties**, and then in the right properties pane, select **Can have connections**.



4. Select **Save**.

View connection roles

There are a number of standard connection roles already configured in Dataverse.

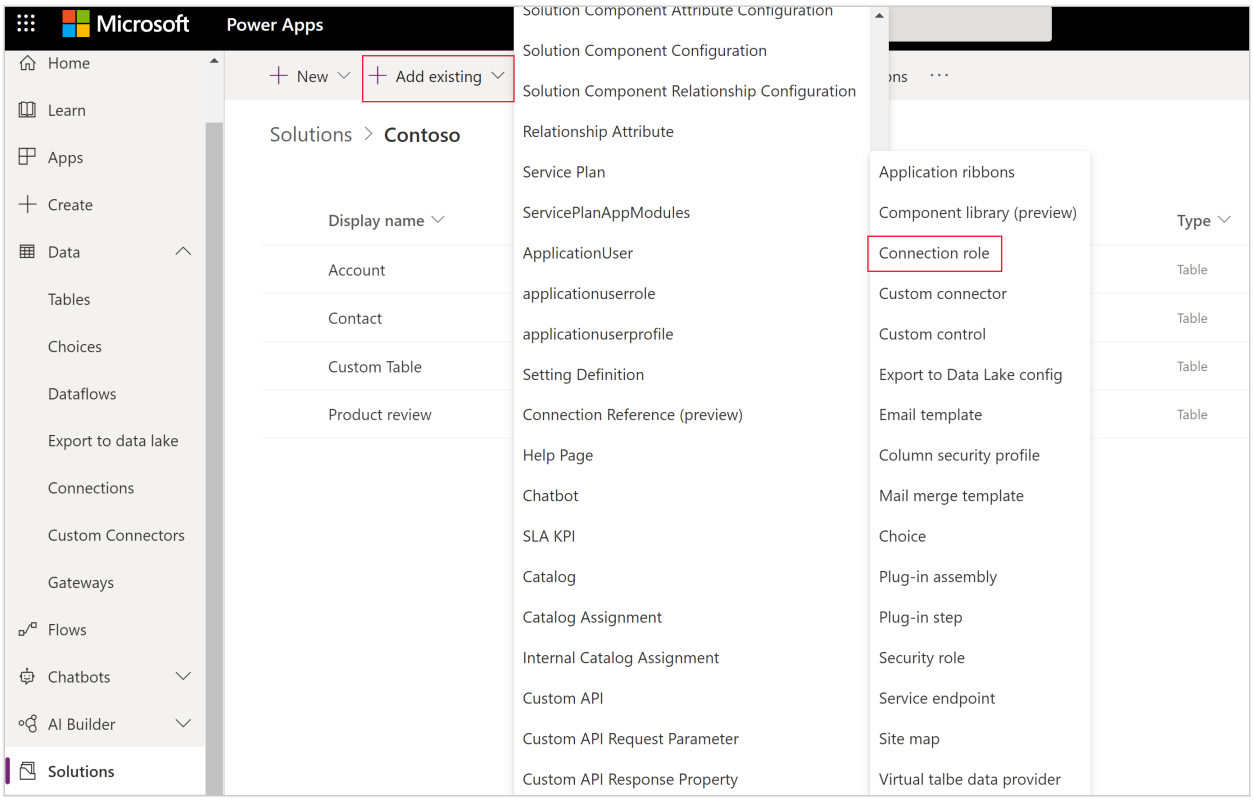
1. Sign in to Power Apps, and then on the left pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
2. Open the unmanaged solution you want.
3. On the command bar, select **Add existing** > **Connection Role**. The list of available connection roles are displayed.
4. Select **Cancel** to close the **Add existing connection Roles** pane without adding a connection role to the solution.

> **Note**
>
> - If you want to distribute connection roles with a solution, make sure they are included in the solution you want to distribute. More information: **Add connection roles to a solution**

Add connection roles to a solution

Because connection roles are *solution aware*, which means that they can be included in a solution, you can also add connection roles to a solution you distribute.

Generally we don't recommended you edit components in the internal **Default Solution**. Within the solution you've created to work in, you can use the **Add Existing** command in the **Solutions** area to bring any of the active connection roles into your solution.



Once you add the connection role to your solution, you can edit it wherever it's visible.

> **Note**
>
> The connection role status is not included with the connection role when it is exported from a solution. Therefore, when the solution is imported into a target environment, the status will be set to active by default.

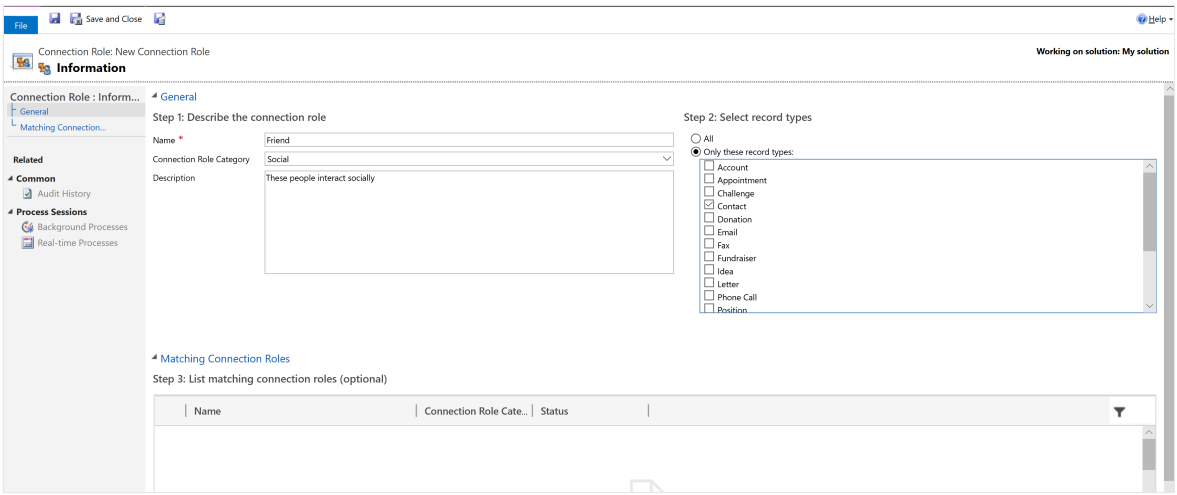
Create a connection role

> **Important**
>
> If you intend to distribute a solution that includes new connection roles or changes to the existing connection roles you must add them to the solution you will distribute. Editing or adding new connection roles in the **Default Solution** will not include them in a solution you can distribute. More information **Add connection roles to a solution**

1. Sign in to Power Apps and then on the left pane select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. Open the unmanaged solution your want, and then on the command bar select **New** > **Other** > **Connection role**.

3. Complete the three steps on the form to Describe the connection role.



Describe the connection role

Set the following columns:

Column	Description
Name	(Required) The text describing the connection.
Connection Role Category	A group describing the category of the connection. More information: Connection Role Category values
Description	Provide a definition for the role.

Connection Role Category values

The default **Connection Role Category** values are:

- Business
- Family
- Social
- Sales
- Other
- Stakeholder

- Sales Team
- Service

You can add new categories or modify existing ones by editing the **Category** global choice. More information: Create and edit global choices for Dataverse (picklists)

Select row types

Select which row types should be available to connect.

> **Note**
>
> Although **All** is selected by default, make sure you consider which types are appropriate for the connection role you are adding.

Matching connection roles

In this optional step, you can define any roles that be applied in a reciprocal manner. It isn't required, but connections are more meaningful if these are defined.

For example, people can set that Glen is a *Friend* to Mary, but does this mean that Mary is a *Friend* to Glen? We hope so. But if Glen is the *Father* of Mary it doesn't mean that Mary is the *Father* of Glen. Establishing correct reciprocity requires this extra step.

When people set a connection role that doesn't have a matching connection role, the role will only be displayed when viewing the connection from the row that the connection was applied to. When viewed from the connected row, the role will be empty unless a matching role is set.

For role definitions like *Friend*, *Spouse*, *Colleague*, or *Sibling*, it's best to assign the matching role to itself. If a single matching connection role is configured, the single matching connection role will be applied in both directions.

> **Important**
>
> You will need to save a new connection role without this matching connection role before you can set the matching connection role to itself.

You'll find that some connection roles are already configured with matching connection roles. *Former Employee* is matched with *Former Employer* and vice versa. This kind of one-to-one matching connection role is most common.

You can configure multiple matching connection roles to describe complex relationships. If you create a connection role such as *Father*, you could configure two more roles such as Daughter,* and *Son* and apply both of them as matching connection roles to *Father*. In turn, both the *Daughter* and *Son* connection roles should be matched to *Father*. Of course, then you should set up an equivalent role for *Mother* that is similarly matched with *Daughter* and *Son*.

> **Tip**
>
> Before you create a complex set of connection roles, consider if a more simple set of roles will be enough. For example, rather than creating a complex set of connection roles like *Father*, *Mother*, *Son*, and *Daughter* - consider if simply using *Parent* and *Child* will work for you.

If more than one matching connection role is configured, those connection roles represent the only valid reciprocal roles. The first one will be applied automatically as the default value. If the default value isn't correct, people need to manually edit the connection and choose between valid options defined in the configuration.

See also

Create connections to define and view relationships between rows (classic)
Create and edit global choices for Dataverse (picklists)
Create and edit relationships between tables

Improve copilot responses from Microsoft Dataverse

Copilot experiences in Dynamics 365 apps, apps in Power Apps, or Microsoft 365 Copilot allow users to ask natural language questions about data that's stored in Dataverse. Copilots help your users get insights and information from your enterprise data, which improves their productivity and decision-making. However, sometimes they might not get the answers they expect or need from copilots. This article shows you how to improve the accuracy and relevance of copilot answers by using Microsoft Copilot Studio and the record picker.

Prerequisites for using copilots with Dataverse

By default, the Copilot feature for app users in model-driven apps is turned off. Admins must enable this feature for their environments in the Power Platform admin center. Learn more: Turn on copilots and generative AI features

Improve results for Dataverse using knowledge sources in Copilot Studio

As a maker, you know how your data is structured and how your users ask questions about it. To improve the quality of the results, you can teach the copilot some domain knowledge, also known as *grounding*, using synonyms and phrases with definitions. This information helps the copilot understand how your users ask questions that don't match the names or values of the tables or columns in your data.

For example, if your users ask "What are my closed leads?" and the results depend on a column that doesn't include "closed" or a status that doesn't have "closed" in the definition, the results might be imprecise. If your company defines a closed lead as a status of not qualified or canceled, you can add a glossary term that defines "closed leads" as "lead status isn't qualified or canceled." This additional information helps the copilot understand your users' questions better.

Teach your copilot domain knowledge by adding a knowledge source in Copilot Studio to make its answers more relevant and informative for your users. You can add enterprise data from Power Platform, Dynamics 365, and external systems as knowledge sources. You can also add Dataverse data as a knowledge source. Learn more: Add knowledge to a copilot

Improve answers for a specific record using a copilot in an app

In Power Apps model-driven apps and Dynamics 365 Sales, users can use the record picker to improve their results. The record picker lets you choose a record to use as the object of your question. This helps you get better answers when the results include records from different tables or records that aren't what you wanted.

To open the record picker, type `/`. Then type the name of the record you want. Learn more: Use Copilot chat in model-driven apps

Related information

- Copilot in Power Apps overview
- FAQ for Copilot in model-driven apps
- FAQ about using AI responsibly in Power Apps

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 01/09/2026

Create new data columns in Dataverse

Columns, also known as fields or attributes, in Microsoft Dataverse define the individual data items that can be used to store information in a table. Column data is used in apps to display information on forms, in views, and can be used in searches within an app. By default, the account main form has several columns, such as account name, phone, fax, website, and so on.

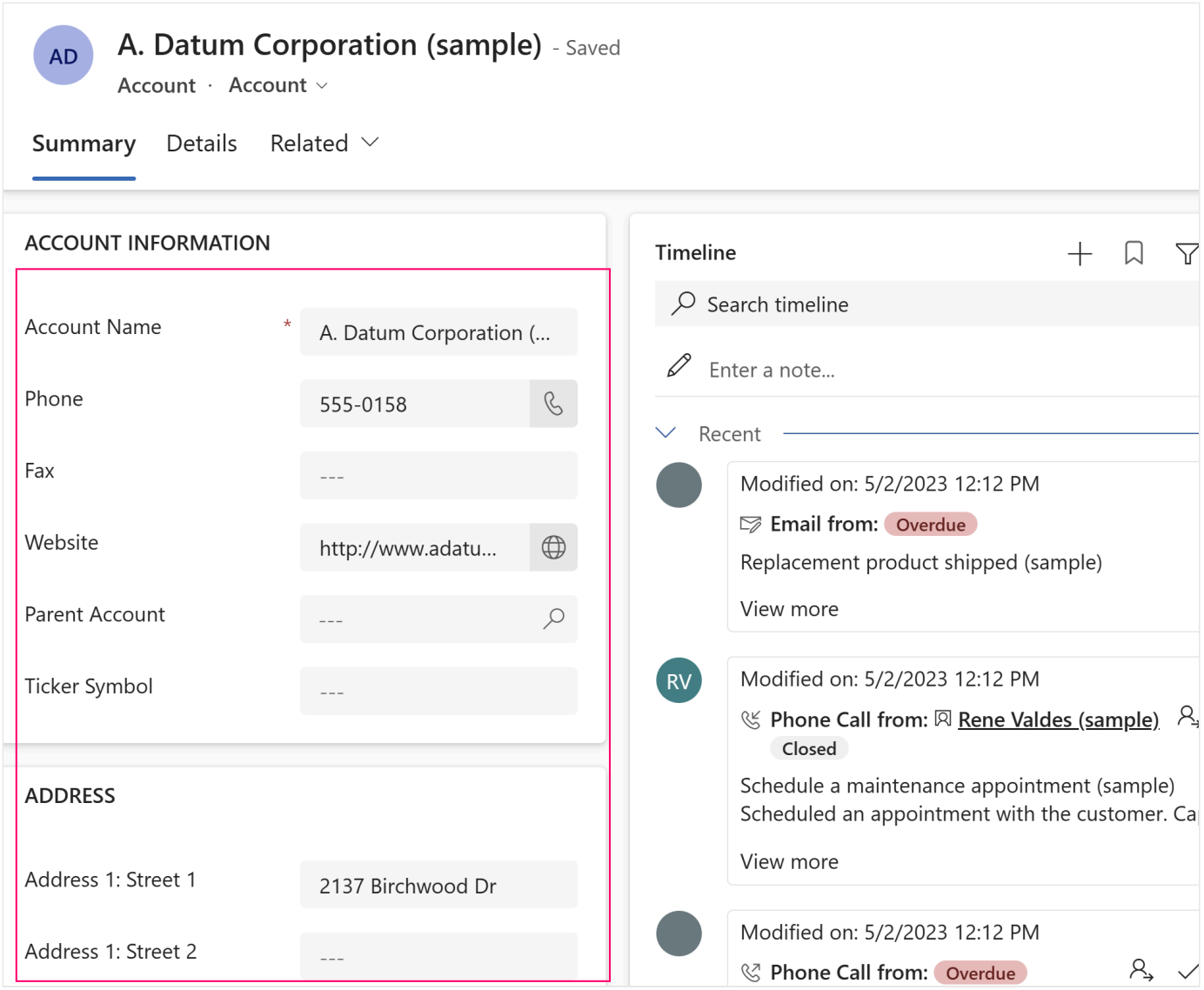
With the exception of choices columns, all columns depend on a table. Columns support many different data types, such as text, number, date and time, lookup (links to another table), currency, autonumber, file, or Power Fx formula.

Create new columns to capture data when existing standard tables don’t have columns that meet your requirements.

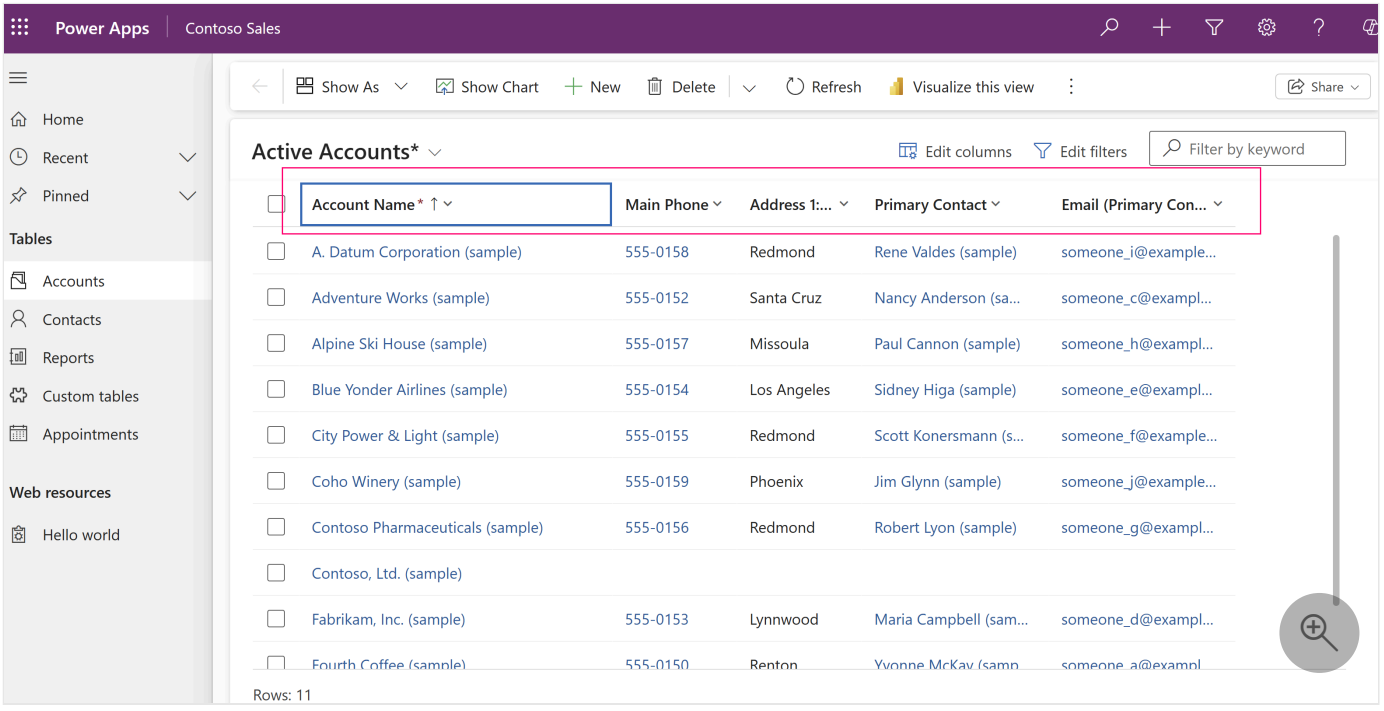
How table columns are used in apps

After you create a new column, you include it on the appropriate forms using the form designer, and views using the view designer, for the table so that they're available in your app.

Here are default columns on a form for the account table in a model-driven app.



Here are the default columns for the active accounts view in a model-driven app.



Create a table column

Watch this short video that shows you how to quickly create a column.

https://learn-video.azurefd.net/vod/player?id=f0015291-5024-4427-a669-dd230cf5c0c8&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Ffields-overview

Create a column in Power Apps (make.powerapps.com). You can also create columns while working in the model-driven app form designer or view designer.

1. Go to Power Apps, and then select **Solutions** in the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select the table where you want to add a column. If the solution doesn't already have the table, select **Add existing** > **Table** or to create a new table select **New** > **Table** > **Table (advanced properties)**. More information: Set advanced properties for a table
3. In the table **Schema** area, select **Columns**.
4. In the list of columns for the table, select **New column**.
5. Complete the required and optional properties for the column, and then select **Save**.

Go to these articles for more information about creating columns:

- Types of columns
- Create and edit columns in Dataverse using Power Apps
- Create and edit choice columns overview

Last updated on 04/27/2026

Types of columns

The names used for types depend on the designer used. Power Apps uses a convention that includes the way the data is formatted. The solution explorer type uses a name aligned with the database data type with a format modifier.

Watch this video for a quick overview about data types in Dataverse:

https://learn-video.azurefd.net/vod/player?id=d64c0198-42de-4dd0-8cec-14e3008442e6&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Ftypes-of-fields

The following table includes the corresponding `AttributeTypeDisplayName` API type.

Expand table

Power Apps data type	Solution Explorer type	API type
Big	Time Stamp	`BigIntType`
Choice	Option Set	`PicklistType`
Choices	MultiSelect Field	`MultiSelectPicklistType`
Currency	Currency	`MoneyType`
Customer	Customer	`CustomerType`
Date and Time	Date and Time *Date and Time* Format	`DateTimeType`
Date Only	Date and Time *Date Only* Format	`DateTimeType`
Decimal Number	Decimal Number	`DecimalType`
Duration	Whole Number *Duration* Format	`IntegerType`
Email	Single Line of Text *Email* Format	`StringType`
File	File	`FileType`
Floating Point Number	Floating Point Number	`DoubleType`
Image	Image	`ImageType`
Language	Whole Number	`IntegerType`

Power Apps data type	Solution Explorer type	API type
	Language Format	
Lookup	**Lookup**	`LookupType`
Multiline Text	**Multiple Lines of Text**	`MemoType`
Owner	**Owner**	`OwnerType`
Phone	**Single Line of Text** *Phone* Format	`StringType`
Status	**Status**	`StateType`
Status Reason	**Status Reason**	`StatusType`
Text	**Single Line of Text** *Text* Format	`StringType`
Text Area	**Single Line of Text** *Text Area* Format	`StringType`
Ticker Symbol	**Single Line of Text** Ticker Symbol Format	`StringType`
Timezone	**Whole Number** *Time Zone* Format	`IntegerType`
Unique Identifier	**Unique Identifier** or **Primary Key**	`UniqueidentifierType`
URL	**Single Line of Text** *URL* Format	`StringType`
Whole Number	**Whole Number** *None* Format	`IntegerType`
Yes/No	**Two Options**	`BooleanType`

For more descriptions for each type you can add or edit, see the article for the corresponding designer:

- Create and edit columns for Microsoft Dataverse using Power Apps portal: Column Data types
- Create and edit columns for Dataverse using Power Apps solution explorer: Column Data types

For more information about how column data types are defined in the API, see Attribute metadata

Column types used by the system

There are some columns used by the system that you can't add using the designer.

Expand table

Type	Description
Time Stamp	A Big Integer type used by the system to capture a version number for managing updates to a table.
Customer	A lookup column that you can use to specify a customer, which can be an account or contact. **Note**: This attribute can be added using solution explorer designer.
Owner	A system lookup column that references the user or team that is assigned a user or team owned table row.
Status Reason	A system column that has options that provide additional detail about the Status column. Each option is associated with one of the available Status options. You can add and edit the options. You can also include custom state transitions to control which status options are available for certain tables. More information: Define status reason transitions for custom tables
Status	A system column that has options that generally correspond to active and inactive status. Some system attributes have additional options, but all custom attributes have only **Active** and **Inactive** status options.
Unique Identifier	A system column stores a globally unique identifier (GUID) value for each row.

Text columns

Text columns can contain text characters. This column type has several format options that change the presentation of the text.

Watch this video for a quick overview about text columns:

https://learn-video.azurefd.net/vod/player?id=58cc0b9e-1fd3-4d9f-8e62-1514e3c9bc4d&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Ftypes-of-fields

There are three basic text column types. All values indicated here are in number of characters.

Expand table

Column type name	Default value	Maximum value	Description
Text	100	4000	Accepts a single line of text.
Text Area	100	4000	Accepts multiple lines of text. The number of rows displayed for the column can be configured. Use for smaller amounts of text.
Multiline Text	150	1048576	Accepts multiple lines of text. The number of rows displayed for the column can be configured. Use when large amounts of text are needed.

If you reduce the maximum number of characters for the column, existing data won't be truncated. The limit applies to new rows.

Choices

You can customize forms (main, quick create, and quick view) and email templates by adding multi-select columns that are called **Choices**. When you add a choices column, you can specify multiple values that are available for users to select. When users fill out the form they can select one, multiple, or all the values displayed in a drop-down list.

For example, if an organization operates in multiple areas or countries/regions, you can include multiple locations or countries/regions in an 'Area of operation' column. A user can then select one or more locations from the list of available values.

Choices can be used with read-only grids, editable grids, and most forms. Multi-select choices can't be used with:

- Workflows, business process flows, actions, dialogs, business rules, charts, rollup columns, or calculated columns.
- Reports, SLA[1], and routing rules[1].

[1]Table requires Dynamics 365 Customer Service.

Forms

Choices multi-select columns are supported in the following types of forms:

Expand table

Form Type	Availability
Turbo form	Yes
Refresh form	Read-only (column is available but can't be edited)
Legacy form	No
Bulk Edit form	No

You can use global choices that are defined in your organization to configure values for the multi-select choices.

Using the right type of number

When choosing the correct type of number column to use, the decision to use a **Whole Number** or **Currency** type should be straightforward. The choice between using **Floating Point** or **Decimal** numbers requires more thought.

Watch this video to help you decide what number column type to use:

https://learn-video.azurefd.net/vod/player?id=85ab223e-8fe5-4d61-8715-c9a675ade6ad&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Ftypes-of-fields

Decimal numbers are stored in the database exactly as specified. Floating point numbers store an extremely close approximation of the value. Why choose extremely close approximation when you can have the exact value? The answer is that you get different system performance.

Use decimals when you need to provide reports that require very accurate calculations, or if you typically use queries that look for values that are equal or not equal to another value.

Use floating point numbers when you store data that represents fractions or values that you'll typically query comparing to another value using greater than or less than operators. In most cases, the difference between decimal and float isn't noticeable. Unless you require the most accurate possible calculations, floating point numbers should work for you.

Large integers (`Big` or `BigInt`) are large numbers with a max value of 9,223,372,036,854,775,807. It's used to store very large numbers that exceed the capabilities of Whole Number and Decimal. Some uses for this include storage of time stamp values and as unique IDs, as well as numbers larger than 100 billion.

> ⓘ **Note**

The precision and limits of number types described here apply to database capabilities. If you're working with currency and numbers in Power Apps apps, other limitations might apply.

- Up to 15 digits can be handled precisely. For example, the 15-digit number 99999.0000000089 can be entered in the app and stored in the database as is. However, the 16-digit number 999999.0000000089 is processed imprecisely as 999999.0000000088 by the app. Longer numbers might be truncated, even if they haven't reached Dataverse limits. This is an inherent limitation of number processing in web browsers.
- `Big` and `BigInt` aren't supported in canvas and model-driven apps.
- Canvas apps don't currently support decimal type numbers.

Using currency columns

Currency columns allow for an organization to configure multiple currencies that can be used for rows in the organization. When organizations have multiple currencies, they typically want to be able to perform calculations to provide values using their base currency. When you add a currency column to a table that has no other currency columns, two additional columns are added:

- A lookup column called **Currency** that you can set to any active currency configured for your organization. You can configure multiple active currencies for your organization in **Settings** > **Business Management** > **Currencies**. There you can specify the currency and an exchange rate with the base currency set for your organization. If you have multiple active currencies, you can add the currency column to the form and allow people to specify which currency should be applied to money values for this row. This changes the currency symbol that is shown for the currency columns in the form.

  Individuals can also change their personal options to select a default currency for the rows they create.

- A decimal column called **Exchange Rate** that provides the exchange rate for a selected currency associated with the table with respect to the base currency. If this column is added to the form, people can see the value, but they can't edit it. The exchange rate is stored with the currency.

For each currency column you add, another currency column is added with the suffix `_Base` on the name. This column stores the calculation of the value of the currency column you added and the base currency. Again, if this column is added to the form, it can't be edited.

When you configure a currency column, you can choose the precision value. There are three options as shown in the following table.

Expand table

Option	Description
Pricing Decimal Precision	This is a single organization precision to be used for prices found in **Settings** > **Administration** > **System Settings** > **General Tab**.
Currency Precision	This option applies the precision defined for the currency in the row.
Specific precision values	These settings allow for defining a specific set precision using values between 0 and 4.

Different types of lookups

When you create a new lookup column, you're creating a new Many-to-One (N:1) table relationship between the table you're working with and the **Target Row Type** defined for the lookup. There are additional configuration options for this relationship that are described in Create and edit relationships between tables. But all custom lookups can only allow for a reference to a single row for a single target row type.

However, you should be aware that not every lookup behaves this way. There are several different types of system lookups as shown here.

Expand table

Lookup type	Description
Simple	Allows for a single reference to a specific table. All custom lookups are this type.
Customer	Allows for a single reference to either an account or a contact row.
Owner	Allows for a single reference to either a team or a user row. All team or user-owned tables have one of these. More information: Add a table as a lookup option in your app
PartyList	Allows for multiple references to multiple tables. These lookups are found on the Email table **To** and **Cc** columns. They're also used in the Phone and Appointment tables.
Regarding	Allows for a single reference to multiple tables. These lookups are found in the regarding column used in activities.

Image columns

Use image columns to display images in your applications. Image columns are optimized for storing binary data. Dataverse doesn't save this data in the relational data store, which improves performance and reduces the capacity usage. Learn more about storage capacity

Each table can have one *primary image* column. With model-driven apps, you can display this image in the upper right corner of the form. Even though a table has an image column, displaying that image in a model-driven app requires that you enable two settings.

- The standard table definition **Primary Image** property value must be set to **Default Image**. Custom tables require a custom image column. Then, you can select that image column for the **Primary Image** value in the custom table definition.
- The table form where the image is to be displayed must have the **Show image in the form** property enabled.

People choose the image to upload a picture from their computer. Images must be less than 30 MB and must be in one of the following formats:

- jpg
- jpeg
- gif
- bmp
- png

When the image is uploaded, it is converted to a .jpg format and all downloaded images also use this format. If an animated .gif is uploaded, only the first frame is saved.

When an image is uploaded, it is resized as a "thumbnail" image to a maximum size of 144 pixels by 144 pixels. People should resize or crop the images before they upload them so that they display well using this size. All images are cropped to be square. If both sides of an image are smaller than 144 pixels, the image is cropped to be a square with the dimensions of the smaller side.

> ⓘ **Note**
>
> Image columns don't work with business process flows, business rules, charts, rollup columns, or calculated columns.

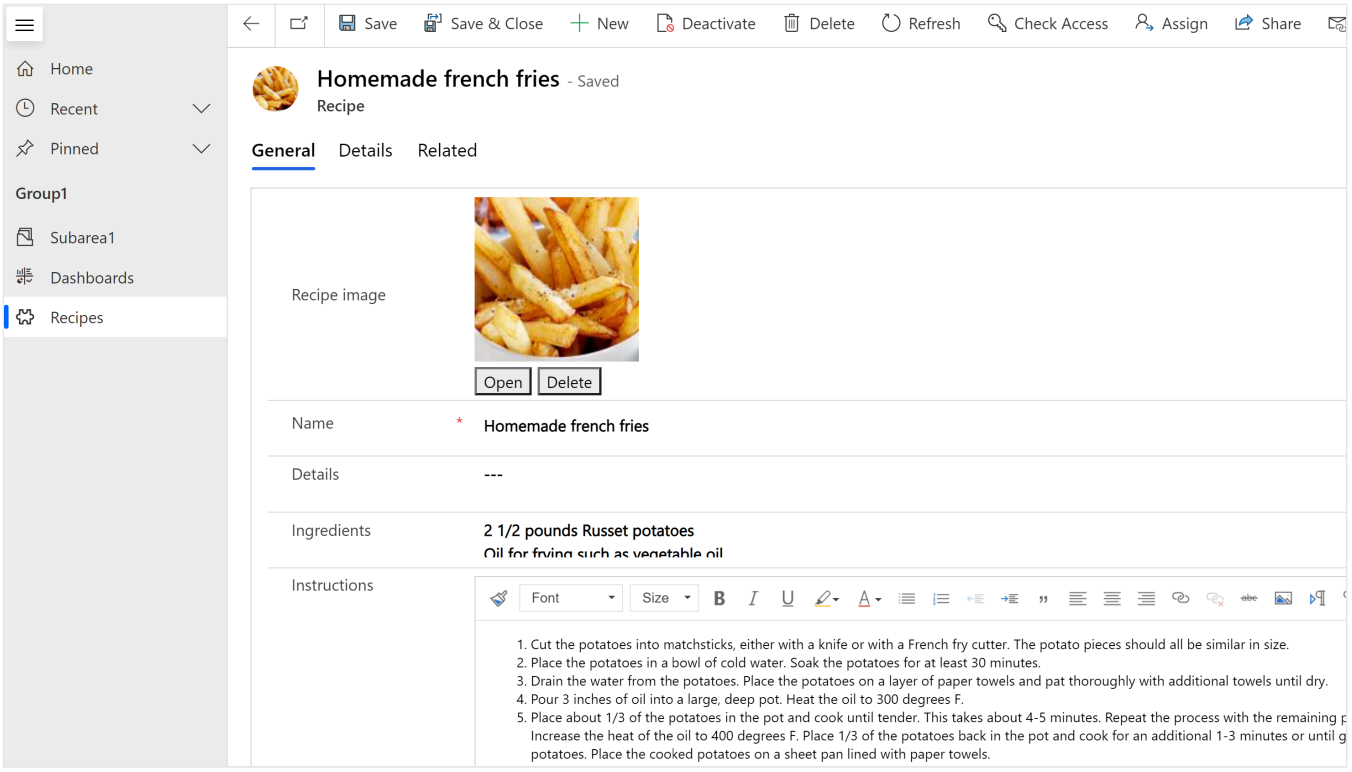
Learn to work with Image column definitions using code

Create an image column and add it to a form

1. Go to https://make.powerapps.com, select **Solutions**, and then open the unmanaged solution that contains the table form where you want to add an image column.
2. Open the table, and then select the **Columns** area.
3. On the command bar, select **Add column**.
4. Enter or select values for the following properties, and then select **Done**.
    - **Display Name**, such as *Recipe image*.
    - **Name**. This is the unique name that includes the solution publisher prefix and can't be changed once saved.
    - **Data Type**. Select **Image**.
    - **Primary Image**. Image columns that are set as the primary image are displayed in the upper right corner of the form. You can have only one primary image for each table.
    - **Enable column security**. Use to control access for specific columns. More information: Field-level security to control access
    - **Enable auditing**. Enables the logging of changes that are made to table records and user access so you can review the activity later. More information: Administrators Guide: Manage Dataverse auditing
    - **Sortable in interactive experience dashboard**. Specifies the column will be used to configure interactive dashboards. More information: Configure filter columns, and security roles for interactive dashboards
    - **Maximum image size**. Default value is 10,240 KB. The minimum size is 1 KB and the maximum is 30,720 KB.
5. Select **Save table**.
6. Select the **Form** tab, and then open the form for editing, such as the table main form.
7. Add the image column to the form canvas.
8. On the form editor command bar, select **Save**, and then select **Publish** to make the image column available to users.

App users can now select the image to display on the form. When an app user opens the form for a record, they can select **Choose file**, select the image, and then save the record. Then, the image is displayed on the form where the image column is located.

The image column in this example is the primary image so the image also appears on the upper left of the form.



Users can select **Open** to display the image full size in a new browser tab or select **Delete** to remove the image from the record and Dataverse.

More information for developers working with image data:

- table metadata > table images
- Image attributes

File columns

The **File** column is used for storing binary data. File columns are optimized for storing binary data. Dataverse doesn't save this data in the relational data store, which improves performance and reduces the capacity usage. Learn more about storage capacity

The primary intended use of this column is to store a single image, note, or attachment. However, storage of other forms of binary data is also possible. One or more columns of this data type can be added to an existing standard customizable table or a custom table.

The default **Maximum file size** is 32 MB and the largest size you can set using the designer is 131,072 KB (131 MB). The file size limit can be set individually for each column of file type added to a table.

> **Note**
>
> - Once the maximum file size is saved, it can't be changed.

- File columns don't work with business process flows, business rules, charts, rollup columns, or calculated columns.
- Required field validation doesn't work with file columns.
- In model-driven apps, deleting or uploading a file on a form happens immediately, not on form save. Discarding changes when navigating away doesn't bring back the file if it's deleted.

To create a file column, on the left pane in Power Apps select **Solutions**, open the solution you want, open the table you want, select the **Columns** area, select **Add Column**, and then in the **Column properties** pane, select **File** as the **Data type**.

Learn to work with file column definitions using code

Fx Formula columns

Built on Power Fx, use a formula column to perform operations that return values during fetch operations. Formula columns use the Power Fx syntax that's similar to Office Excel. More information: Work with formula columns (preview)

Prompt columns

Prompt columns allow AI prompts to be defined and the generative AI results are stored in the table column. More information: Prompt columns

Searching and sorting columns

Most columns have options to enable searching or sorting of the column's contents.

Searchable

Almost every column data type is created with the **Searchable** value enabled. This can be disabled at the time of creation, or later after the column is created. The following data types can't be search enabled:

- Formulas. Formulas are used to create a dynamically calculated output and because of this can't be searched.
- Image. Images are stored and retrieved using reference URLS and because of this they can't be searched.

- Multivalue lookup (PartyList). Some system lookup columns can contain multiple values. For example, the **To** lookup column of **Email** rows can contain multiple recipients. Searching multivalue lookup columns isn't supported.

The **Customer** datatype is search enabled by default and this can't be disabled. It's required to be searchable by the system.

Sortable

Almost every data type is created with the **Sortable** value disabled. The value can be changed at the time of creation or after later after the column is created. The following data types don't provide the ability to enable a sortable attribute:

- Choices. Sorting columns with multiple values isn't supported.
- Customer. Customer is a standard lookup column and can't be sorted because it's dynamically retrieved.
- File. Files are stored using reference URLS and sorting these aren't useful.
- Formulas. Formulas are used to create a dynamically calculated output and because of this can't be sorted.
- Image. There's no meaningful way to sort images.
- Multivalue lookup (PartyList). Some system lookup columns can contain multiple values. For example, the **To** lookup column of **Email** rows can contain multiple recipients. Sorting columns with multiple values isn't supported.

Last updated on 01/09/2026

Create and edit choice columns overview

Article • 10/10/2022

A choice (picklist) is a type of column that can be included in a table. It defines a set of options. When a choice is displayed in a form it uses a drop-down list control. When displayed in **Advanced Find** it uses a *picklist control*. Sometimes choices are called picklists by developers.

You can define a choice to use a set of options defined within itself (locally) or it can use a set of options defined elsewhere (globally) which can be used by other choice columns.

Global choice columns are useful when you have a standard set of categories that can apply to more than one column. Maintaining two separate choice options with the same values is difficult and if they aren't synchronized you can see errors, especially if you are mapping table columns in a one-to-many table relationship. More information: Mapping table columns

> ⓘ **Note**
>
> If you define every choice as a global choice your list of global choice columns will grow and could be difficult to manage. If you know that the set of options will only be used in one place, use a local choice.

Choice columns can be configured as either single select (choice) or multi-select (choices).

> ⓘ **Note**
>
> You can also create global choices in your environment using the following:
>
> - Import a solution that contains the definition of the global choices.
> - A developer can write a program to create them.
>
> More information: **Developer documentation: Choices columns**.

See also

Create a choice

Autonumber columns

Article • 03/13/2023

Autonumber columns are columns that automatically generate alphanumeric strings whenever they are created. Makers can customize the format of these columns to their liking, and then rely on the system to generate matching values that automatically fill them in at runtime.

While autonumber columns are formally just text columns with additional functionality built on top of them, Power Apps simplifies this concept by simply exposing **Autonumber** as a distinct data type under the **Text** category. It is important to note that the classic solution explorer doesn't support creating or managing autonumber columns.

To create an autonumber column, follow the same steps to create a column and simply select **Autonumber** from the **Data type** drop-down list box.

You may also activate autonumber functionality on an existing text column by opening the column and selecting **Autonumber** from the **Data type** drop-down list box. Similarly, autonumber functionality can also be disabled at any time by opening the column and selecting a different option in the **Data type** drop-down list box.

> **Note**
>
> Autonumber values are preselected by the database when the row is started. If a row is started but cancelled, the number it was assigned is not used. If, during this time, another row is completed with the next sequential number, gaps will be present in the autonumbering of rows.

Autonumber types

In order to make the creation of autonumber columns easier, there are a few pre-defined default autonumber types to capture the most common scenarios.

String prefixed number

The most common autonumber format is a simple string prefixed number. When this type is selected, the autonumber will consist of an automatically incrementing number with an optional string constant prefix. For example, a string prefixed number with the

prefix *Contoso* would generate rows such as *Contoso-1000*, *Contoso-1001*, *Contoso-1002*, and so on.

Date prefixed number

Another common autonumber format is a date prefixed number. When this type is selected, the autonumber will consist of an automatically incrementing number with a formatted date prefix. The date portion of the row will reflect the current date and time at which the row was created in UTC time. We have provided a number of various date formats to choose from. For example, a date prefixed number would generate rows such as *2019-26-02-1000*, *2019-27-02-1000*, *2019-28-02-1000*, and so on, depending on the current date and selected date format.

Custom

For more advanced makers with specific use cases, we provide the option to fully customize the desired format of an autonumber column. The format may consist of string constants, automatically incrementing numbers, formatted dates, or random alphanumeric sequences. For detailed information about how to define custom formats, see AutoNumberFormat options.

Seed values

The seed value of an autonumber column is the starting number that is used for the number portion of the format. For example, if you want an autonumber column to generate rows such as *Contoso-1000*, *Contoso-1001*, *Contoso-1002*, and so on, then the desired seed value is 1000, because that is the value that your number sequence starts with. Autonumber columns have a default seed value of 1000, but you may set a custom seed value if you wish.

> **Important**
>
> Setting the seed only changes the current number value for the specified attribute in the current environment. The seed value is not included in a solution when it's imported in a different environment.

Create an autonumber column

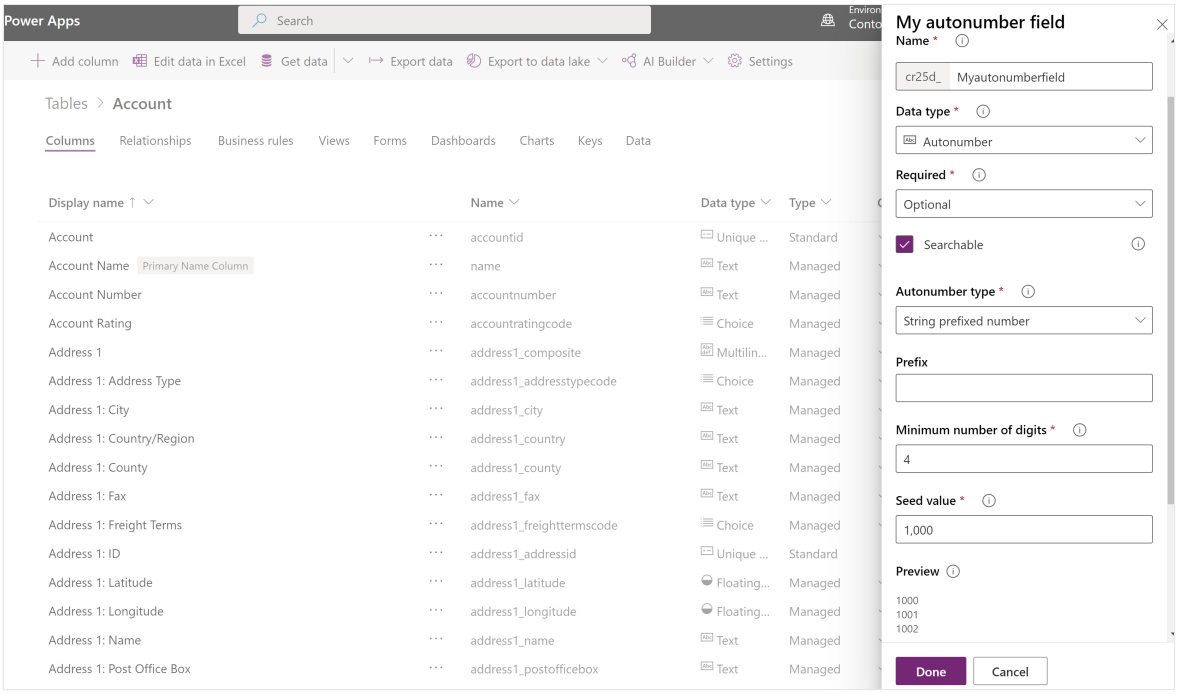
1. Sign in to the Power Apps portal.

2. On the left pane select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. Select the table that you would like to add an autonumber column to and then select **Columns**.

4. On the toolbar, select **Add** column.

5. On the right pane, enter a **Display name** and select **Autonumber** for the **Data type**.



6. Set optional columns as needed. More information: Create and edit columns

7. Select an autonumber type or keep the default **String prefixed number** option.

8. Customize a seed value or keep the default value of **1000**.

9. Select **Done**.

See also

Create and edit columns for Microsoft Dataverse using Power Apps portal

Prompt columns (preview)

[This article is prerelease documentation and is subject to change.]

A prompt column is an AI-powered data type in Microsoft Dataverse that enables you to define natural language prompts tied to other columns in your table. The AI model processes these prompts to generate relevant responses based on specified input columns. The result is immediately stored in the prompt column, ready to be used in apps, workflows, or reports.

The key function is that the prompt column contains generative AI results stored in the table persistently. This is how customer data is enriched using generative AI and brings value to their data.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Example use cases:

- Customer support: Automate responses to frequently asked questions by using AI prompts to generate accurate and timely replies based on customer inquiries stored in Dataverse.
- Content creation: Use prompts to assist in writing articles, reports, or marketing materials by generating text that aligns with the input provided.
- Data analysis: Use AI prompts to analyze complex datasets and generate insights, sentiments, extract, and classify contents or summaries that help in making informed decisions.
- Workflow automation: Integrate AI prompts into business workflows to automatically do tasks such as scheduling, reporting, and data entry.

Prerequisites

- Prompt columns use the same licensing model as AI builder prompts. More information: Prompts overview
- Copilot and AI Prompts features turned On in the Features area for the environment settings. More information: Manage feature settings

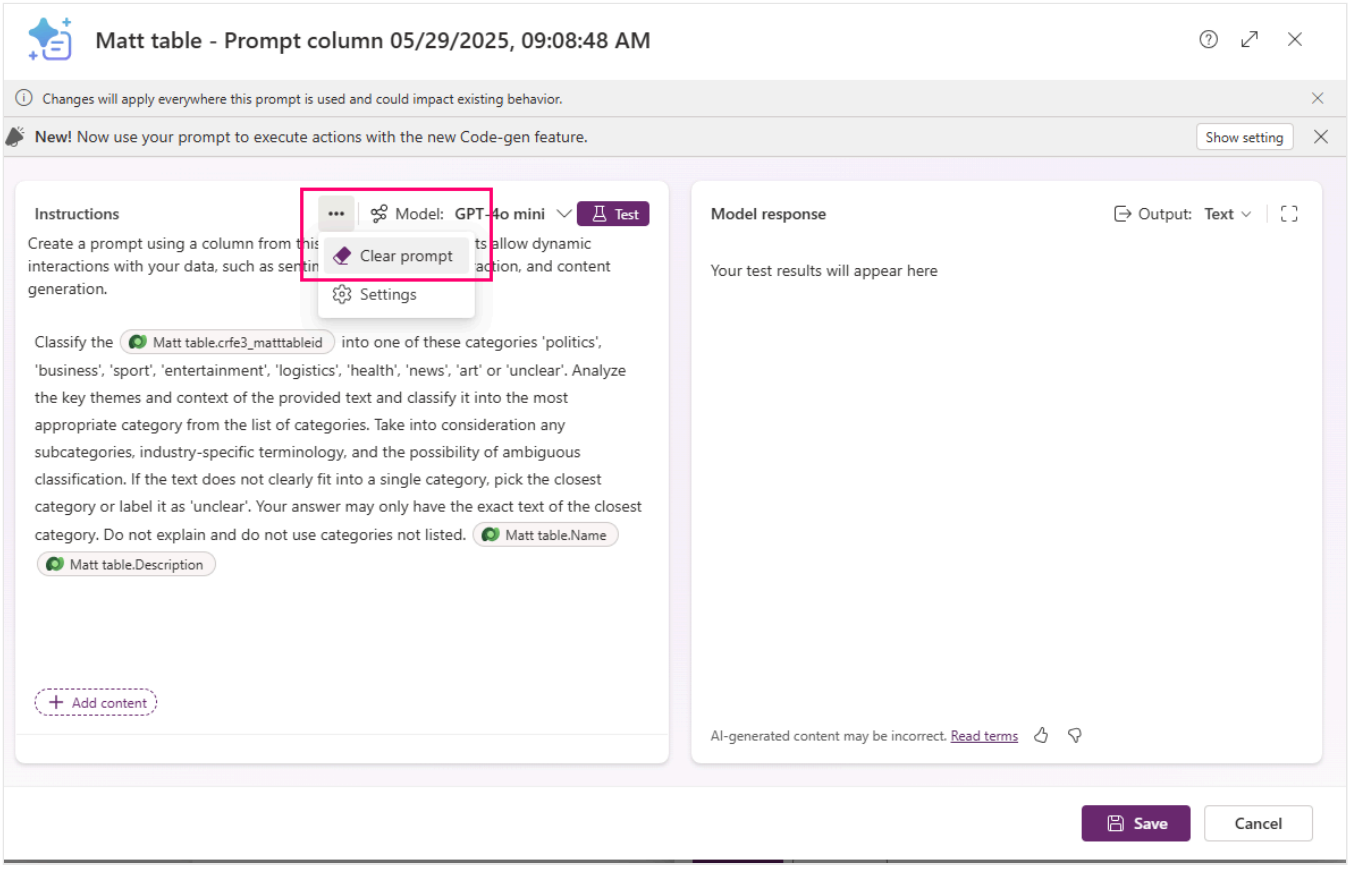
Create a prompt column

1. Go to Power Apps. Select **Tables** on the left navigation pane.
2. Create a new table or open an existing table. More information: Create a table
3. On the **Tables** page, select **New** > **Column**.
4. On the column properties page, enter a **Display name** and **Description** for your column.
5. Under the **Data type** dropdown list, select **Prompt**.
6. Clear the **Allow form fill assistance** checkbox.
7. Select **+Add new prompt**. You can create up to five prompt columns per table.
8. On the prompt column page, create a prompt. More information: How to write an AI prompt for a prompt column
9. Select **Save** on the prompt page.
10. Select **Save** on the column properties page to save your column.

How to write an AI prompt for a prompt column

Crafting effective AI prompts is crucial for getting accurate and relevant responses from the AI model. You can use the prefilled prompt or write your own custom prompt.

To clear the prefilled prompt, highlight and delete it, or select the three dots to the left of **Model**, then select **Clear prompt**.



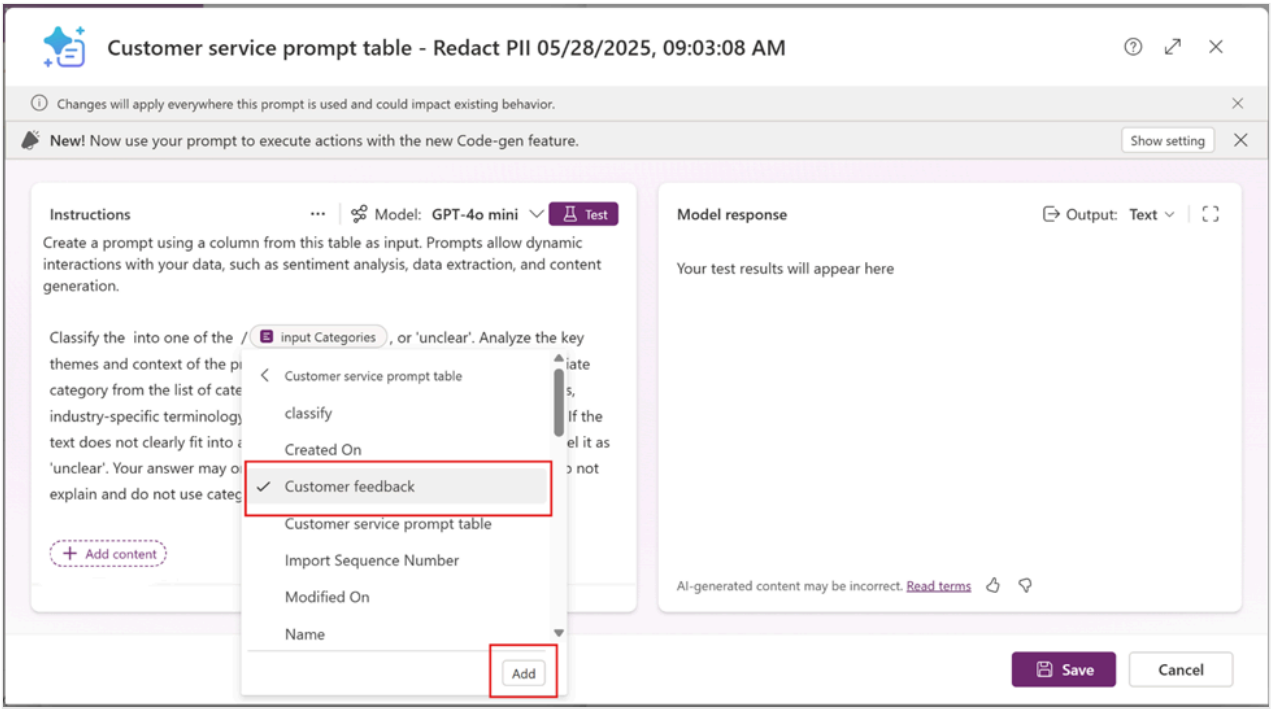
Write effective prompts

Here are some best practices for writing prompts:

- Write clear instructions. Ensure that your prompts are clear and specific. Avoid ambiguity by being precise about what you want the AI to do. For example, instead of writing *Tell me about the weather*, write *Provide a summary of today's weather in Paris*.
- Use structured language. Structured language helps the AI model understand the context and intent of the prompt better. For example, *Generate a list of top 10 marketing strategies for 2025* is more effective than *Top marketing strategies*.
- Include contextual information. Provide necessary context within the prompt to help the AI model generate more accurate responses. For example, *Based on the sales data from Q1, suggest ways to improve our marketing campaign*. Add an input column in each prompt.

Add input columns

1. Replace default input text by selecting it and then select **Delete**.
2. Select **+Add content**, and then select the table.
3. The column dropdown list appears. Select the input column from the list of columns from your table, and then select **Add**. In this example, the *Customer feedback* column is selected.



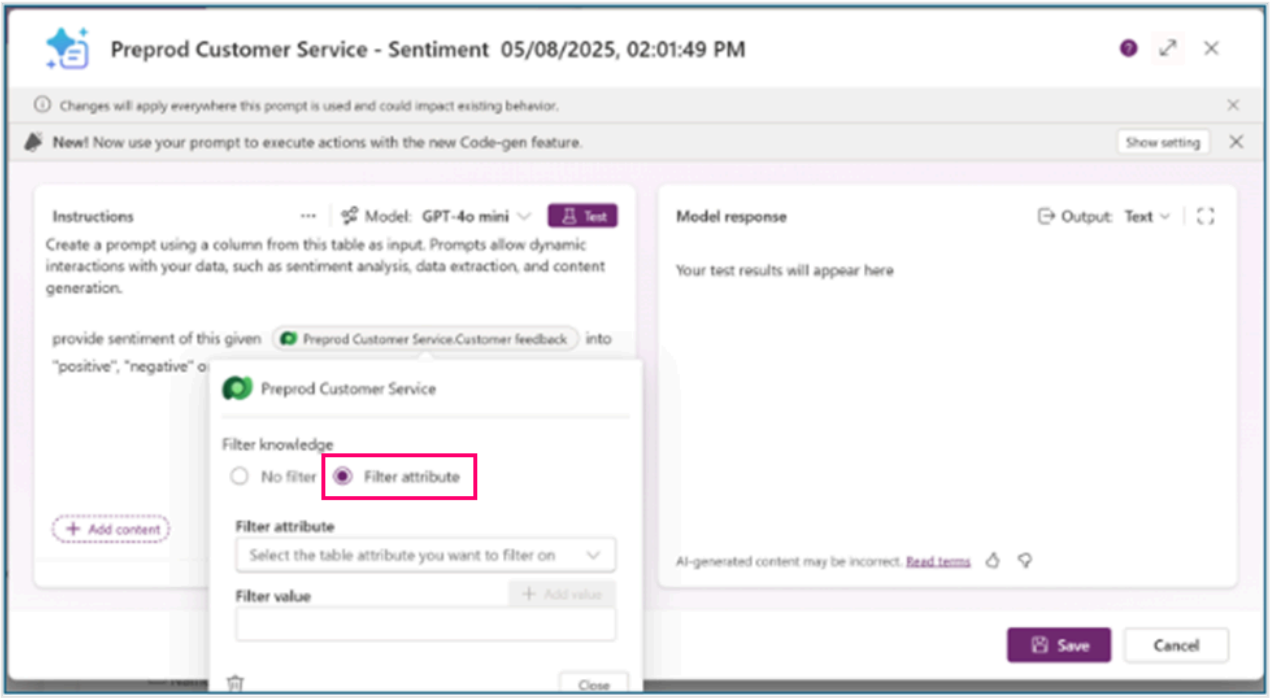
- You can use more than one input column in each prompt.
- Input columns can't be formula columns, file columns, image data type, or another prompt column. If you select these data types, their input value is ignored.

4. Select **Save**.

Test and refine prompts

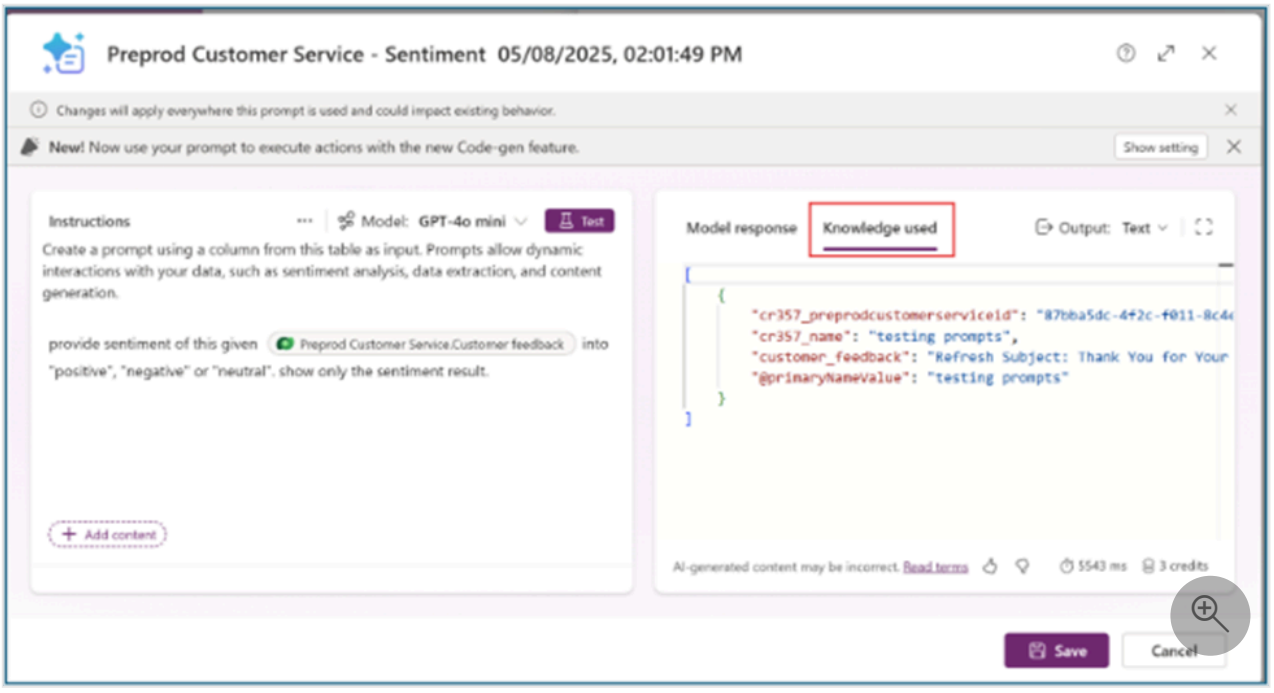
Test your AI prompt. Create a test record with appropriate values in all your input columns for testing. For example, create a *Name* column in the table and enter some value that is used to identify the test record, such as *testing prompts*.

1. Create or select a prompt column.
2. While editing the prompt in a prompt column, select the input column to open the **Filter knowledge** pop-up screen.
3. Select the **Filter attribute** option.



4. Select the **Filter attribute** dropdown list and select a filter, such as *Name*.
5. Enter the value of your testing record **Name**, such as *testing prompt*.
6. Select **Close**.
7. Select **Test** and review the **Model response**.

8. Select the **Knowledge used** tab to confirm that the input came from your test record.



9. Modify your prompt until you get the desired results.
10. Return to the **Filter knowledge** pop-up and select **No filter** when your prompt changes are done.
11. Select **Save** to update your prompt column.

View prompt column results

Create a model-driven app to view and validate your prompt column results.

1. Go to Power Apps, and then open an app for editing or create a new app. More information: Create a model-driven app
2. In the app designer, select **Edit form** to edit the table form for the table that has the prompt columns.
3. The form designer opens. Select all the input and prompt columns you want to add to the form.
4. Select **Save and publish**.
5. Go to the **Tables** area in Power Apps and select **Views**.
6. Observe the values in the records including new records that contain prompt column values.

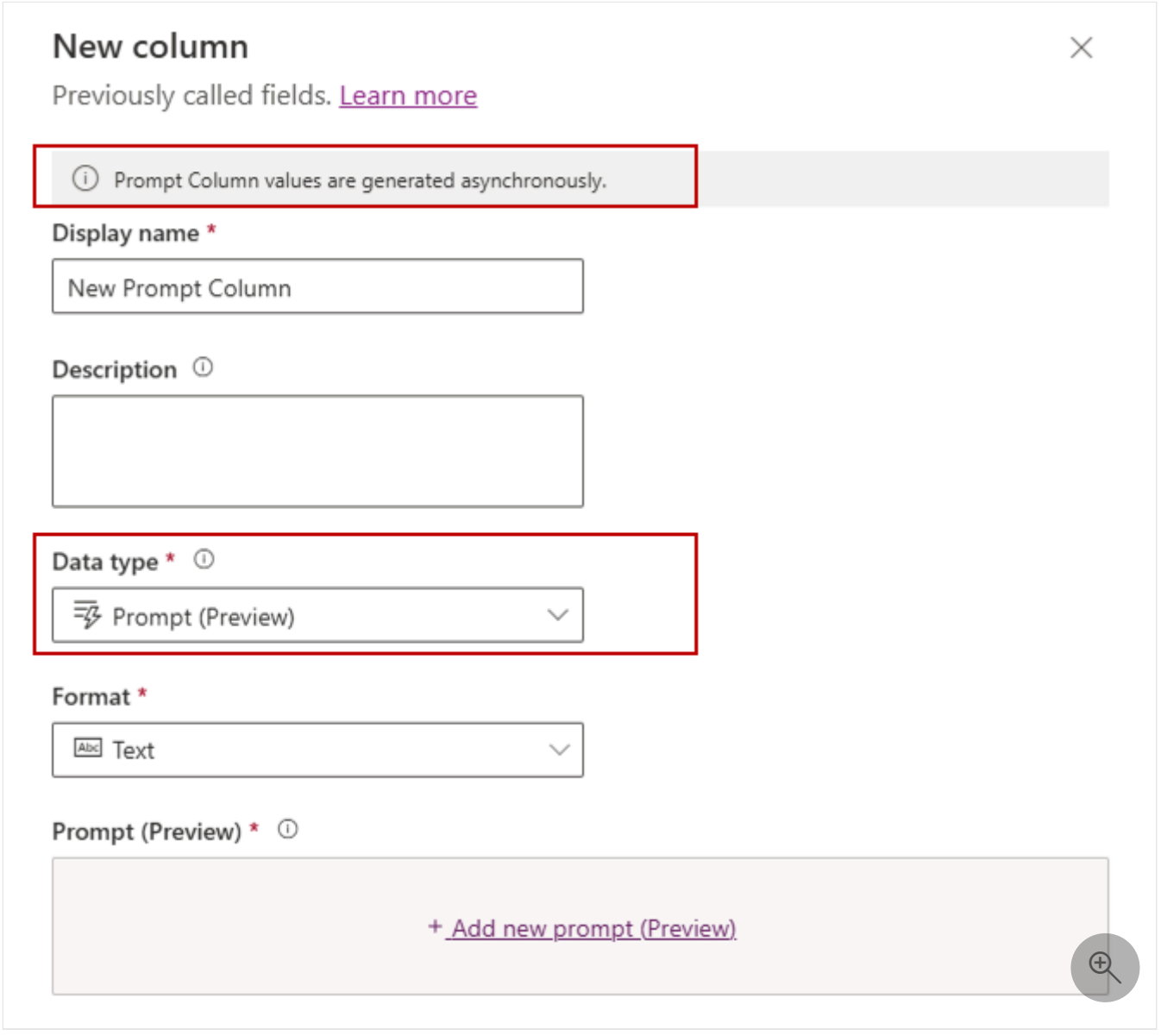
Prompt column feature enhancements

> **Note**
>
> This capability is in process of rolling out, and might not be available in your region yet.

Prompt columns leverage asynchronous computation so that AI-driven processing is decoupled from real-time transactions. This preserves system responsiveness, protects business critical workflows, and enables organizations to adopt AI capabilities at scale without compromising operational reliability.

How to know when prompt values in prompt columns are asynchronous

When a prompt column is added to a table, a banner message appears at the top of the page indicating that the value is generated asynchronously.

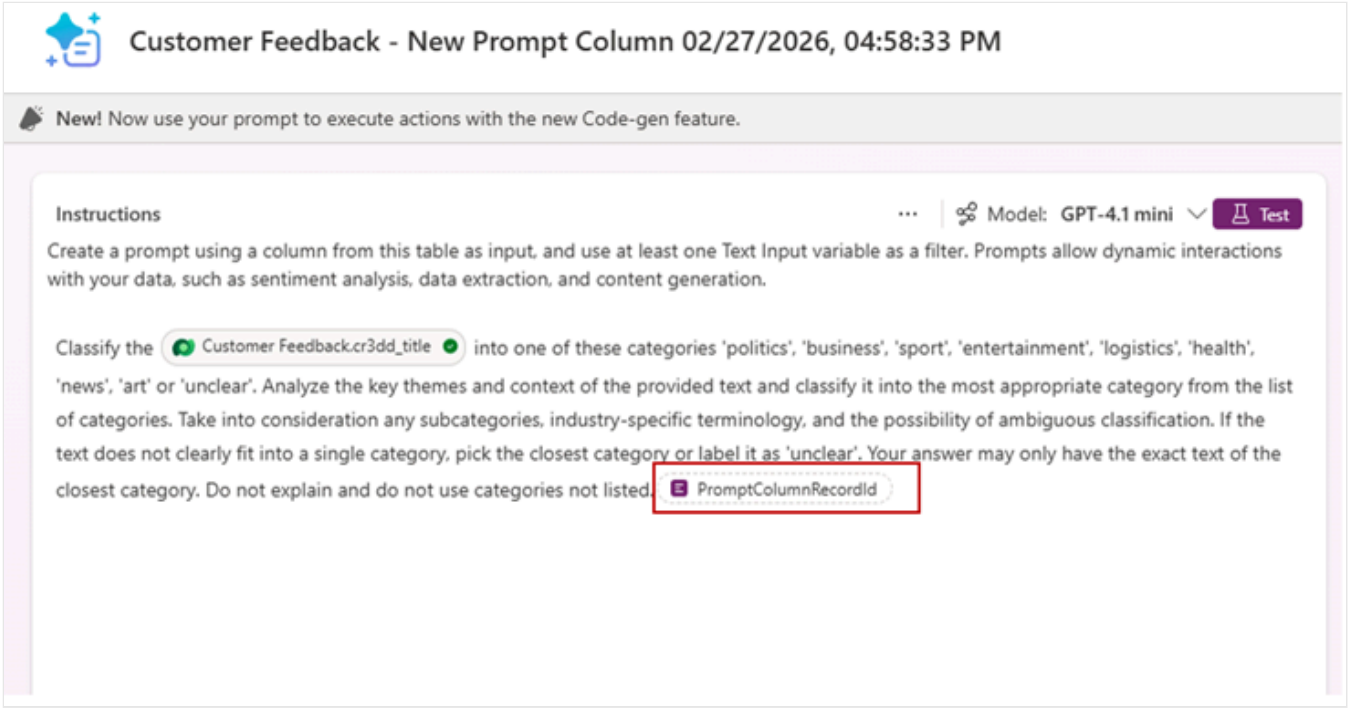


> **Note**
>
> If your environment isn't updated to the latest version, the banner message might not be visible. Prompt columns created in earlier versions will continue to operate without change

until the upgrade occurs. As part of the upgrade process, existing prompt definitions are updated to support asynchronous execution.

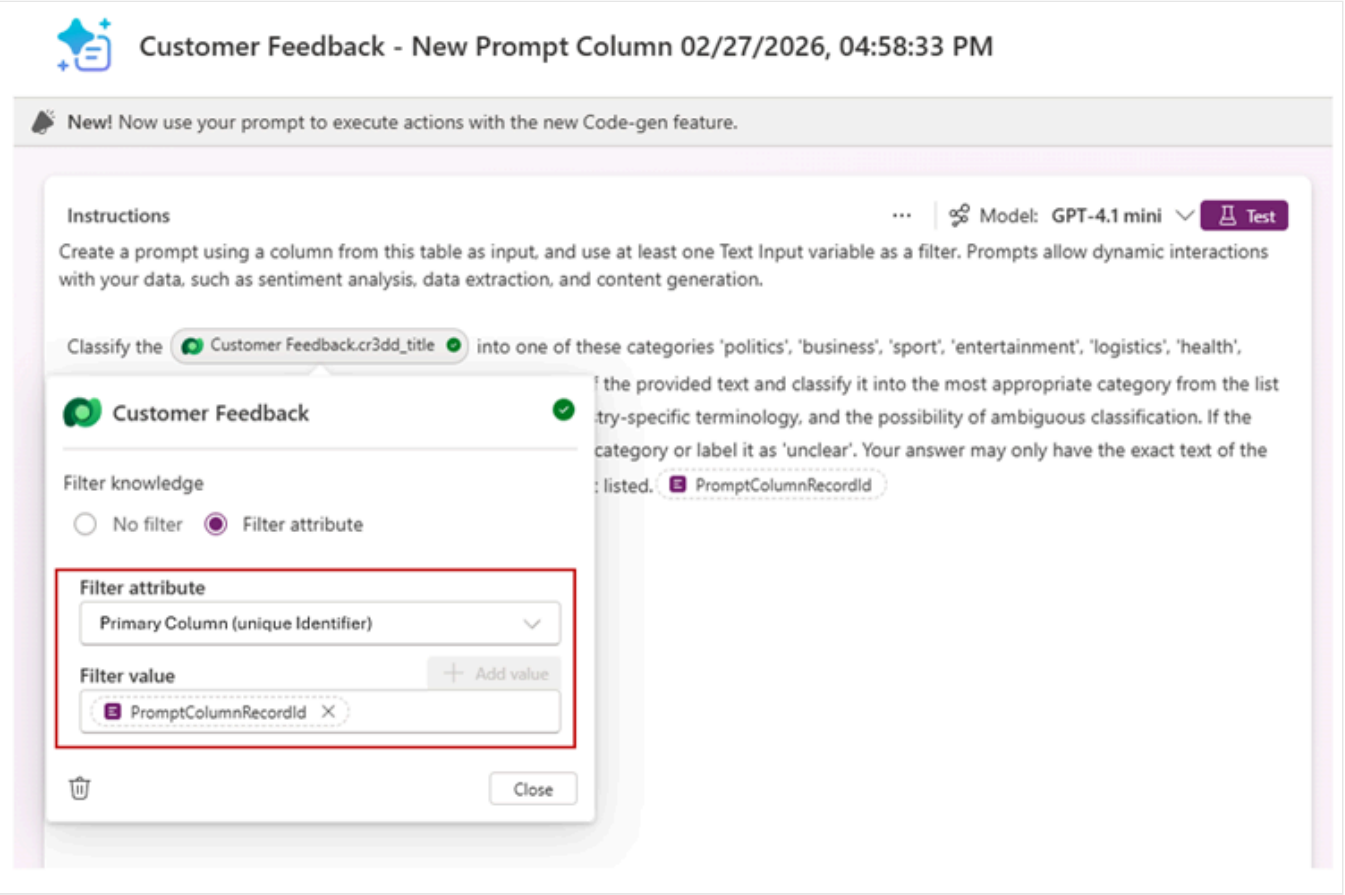
Text input variable

A default text input variable is automatically added to the prompt definition. This variable is used as a filter on the primary column of the data source and is required to save the prompt definition.



This variable is used as a filter on the primary column of the data source and is required to save the prompt definition.

The platform requires a single text input variable. This variable is automatically added and applied as a filter on the primary column of the data source. Users can edit the prompt definition or add additional columns, provided the required text input variable remains included in the prompt definition. When the column is saved, the platform validates this requirement and displays an error if the text input variable is missing or not applied as a filter on the primary column.



Prompt column with filter condition

Filter conditions ensure prompts run only when they add value.

With filter based execution, prompts run only when specified conditions are satisfied. During record creation or update, the prompt column evaluates the filter first and then executes the prompt for eligible records. This approach reduces unnecessary executions and helps optimize Copilot credit consumption.

Add filters to a prompt column

1. Go to Power Apps, and select **Solutions** on the left navigation pane.

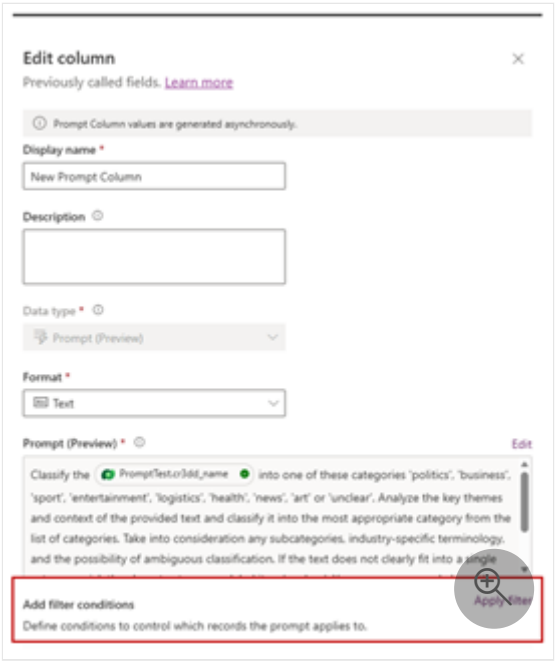
2. Open the solution that has the table you want, select **Tables**, open table that has prompt column you want to edit or create a new one, and then select **Columns**.

    - If creating a new column, select **New** > **Column**. On the right properties page, enter a **Display name** and **Description** for your column, for **Data type**, select **Prompt**.
    - If editing an existing prompt, go to step 5 to apply a filter.

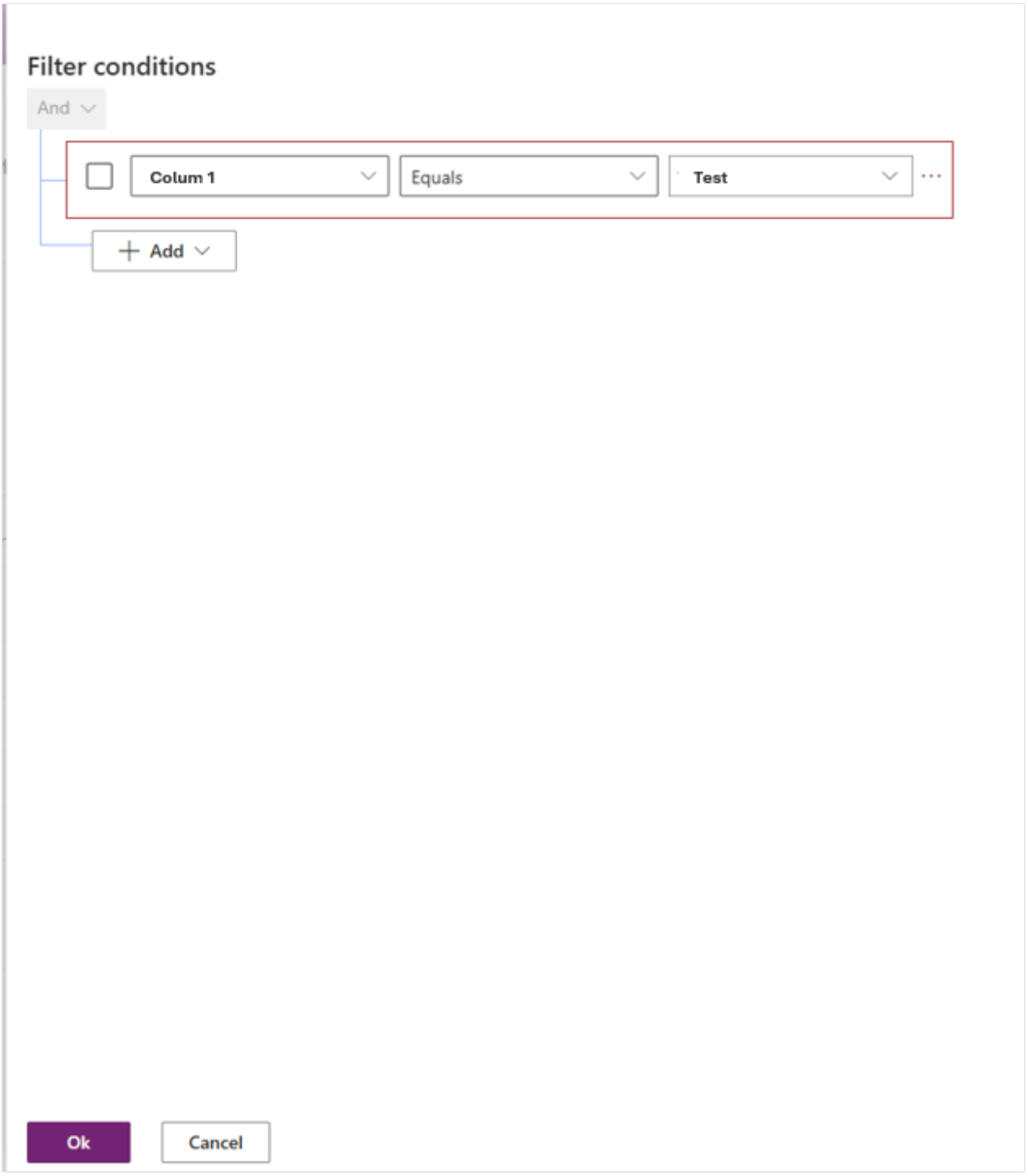
3. Clear the **Allow form fill assistance** checkbox.

4. Select **+Add new prompt**. You can create up to five prompt columns per table. If editing an existing column, select **Edit** next to the **Prompt** box.

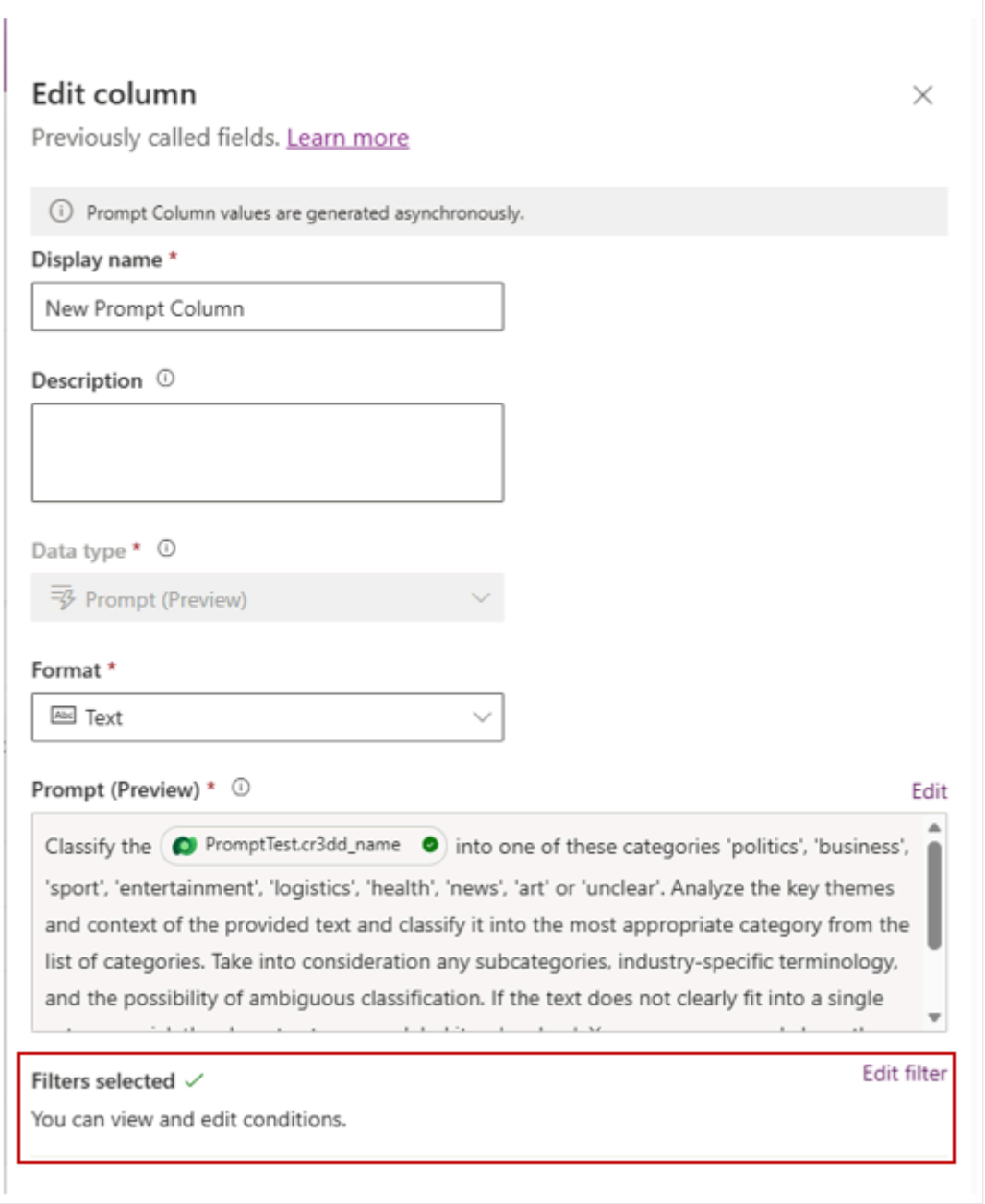
5. Open a table variable, such as `PromptTest.cr3dd_name` in this example, and then select **Apply filter**.



6. On the **Filter conditions** pane, select **Add filter**, and then select **OK**.



7. On the **Edit column** pane, notice **Filters selected** is applied. Select **Save**.



View the prompt column status

When a prompt column is created, the table automatically includes two corresponding columns, **Status** and **Details**, for that prompt column.

1. Go to Power Apps, and then select **Tables** on the left navigation pane.
2. Open the table that has the prompt column where you want to view the status.
3. On the **Tables** page, in the list of columns and data select the **+n more** header to add additional columns.
4. In the select **Show existing column** list search for the prompt column name.
5. Search and select the columns named:

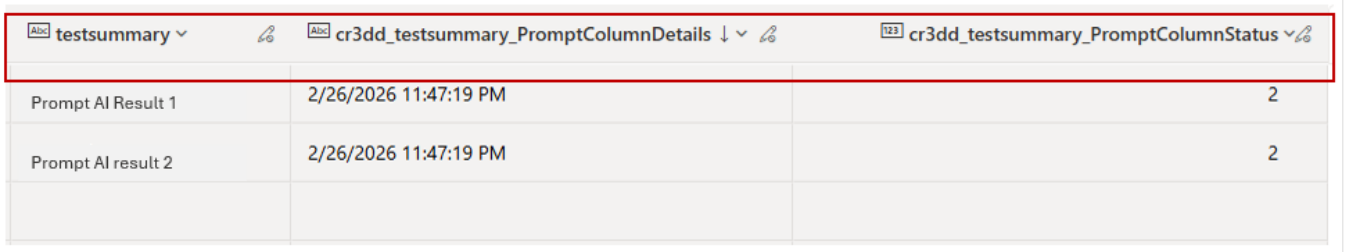
    - `(columnName) _PromptColumnStatus`

- `(columnName)_PromptColumnDetails`

6. Select **Save**.
7. Select **Edit** to the right of the list to open the columns list with records in a separate view.

When prompt execution starts, these columns are populated to record the execution status and any failure details.

This example demonstrates a prompt column named *testSummary* and its corresponding **Status** and **Details** columns.



Prompt column status codes:

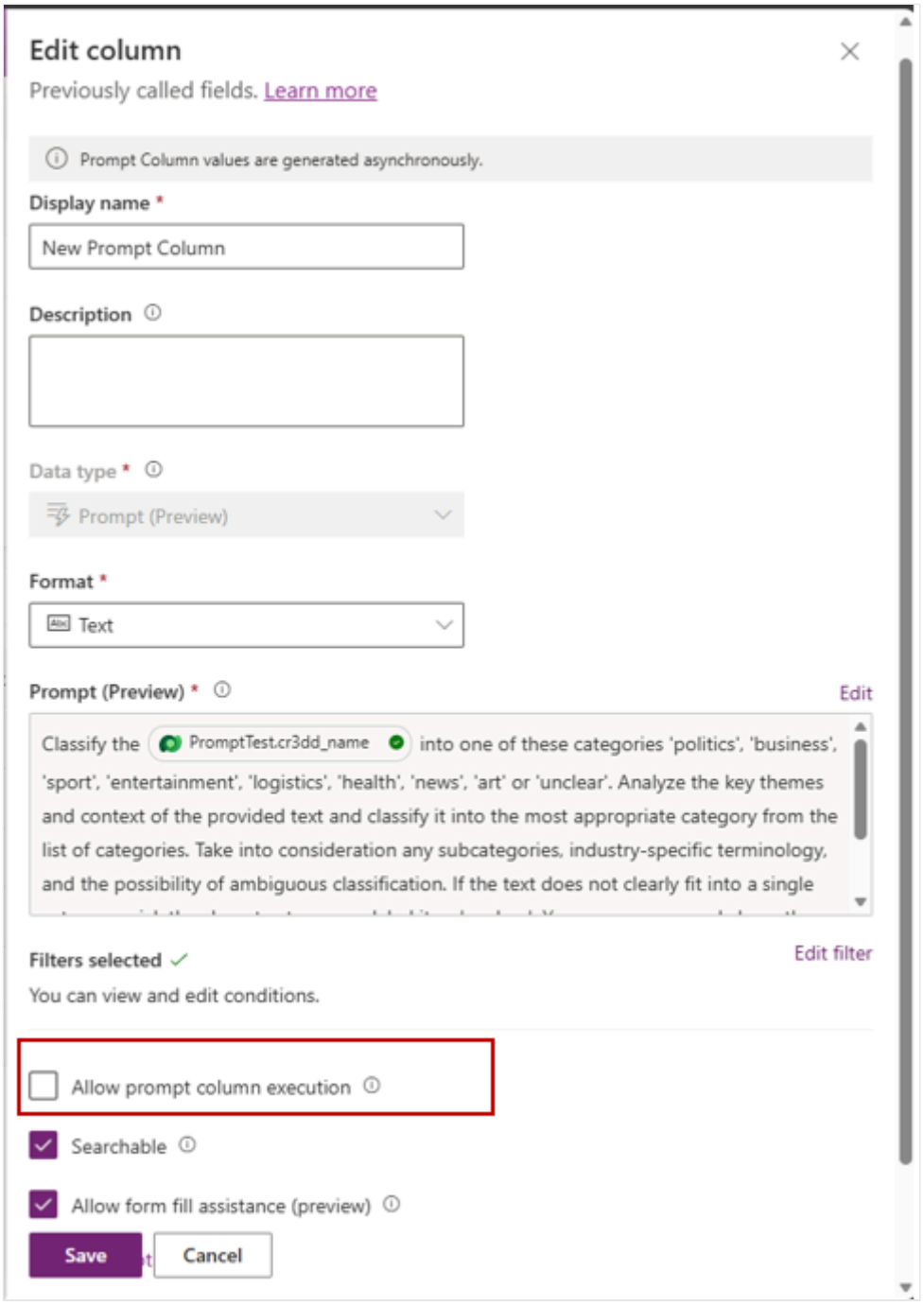
Expand table

Status	Name	Description
0	NotStarted	Record created. AI analysis hasn't started yet.
1	InProgress	The prompt AI analysis is currently in-progress. Details column shows start Date and Time.
2	Completed	Prompt execution completed successfully. Details column shows last successful Date and Time.
3	Failed	The prompt AI generation failed. Detail column shows error details in the last execution.

Disable AI generation for a prompt column

1. Go to Power Apps. Select **Tables** on the left navigation pane.
2. Open the table, select **Columns**, and then open the column that is of data type Prompt where you want to disable AI generation.

3. On the **Edit column** pane, clear the **Allow prompt column execution** checkbox, and then select **Save**.

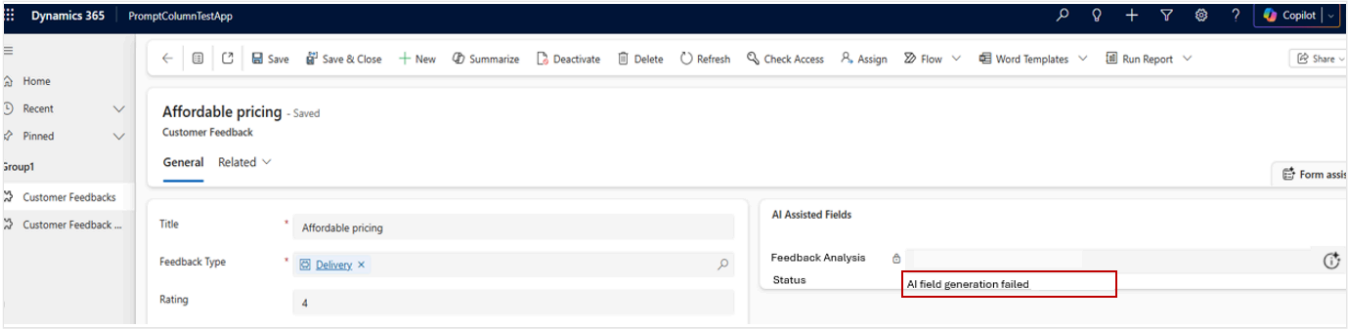


By default, the **Allow prompt column execution** setting is enabled. When disabled, AI analysis isn't performed for the prompt column.

> **Note**
>
> For prompts to be executed, both tenant level and column level settings need to be enabled.

When Prompt column fails to generate AI Analysis

When AI analysis failed to generate, users see an AI field generation failed message.



Prompt column execution failures can occur due to various conditions. The scenarios listed below represent common causes but don't cover all possible cases.

- The user doesn't have sufficient permissions on one or more input columns referenced in the prompt column definition.
- Lack of entitlements and credits in Copilot.

Check **Details** and **Status** columns in the table. More information: View the prompt column status

FAQs

Are backfills supported?

Prompt columns generate AI results when a new record is created, or when one or more columns specified in the prompt column definition are updated.

Existing records aren't processed by default. However, updating a column defined in the prompt column definition triggers AI analysis.

Are prompt columns audited?

Prompt columns aren't audited.

Can I perform on-demand AI analysis without updating a record?

Prompt columns are triggered automatically on record updates; however, on-demand execution is not currently supported. As a result, updating the prompt definition does not recalculate existing prompt values until a subsequent update occurs on one or more columns defined in the prompt.

Why didn't the prompt column execute?

Common scenarios where the prompt does not execute, no error is surfaced, and execution is skipped

- AI Prompt feature is disabled. More information: Prerequisites

- Prompt column execution is disabled. More information: Disable AI generation for a prompt column
- Filters conditions aren't met. More information: Add filters to a prompt column

Check **Details** and **Status** columns in the table for more information.

Do I need to disable 'Block unmanaged customizations' to create or edit prompt columns?

To create or edit Prompt Columns, the'Block unmanaged customizations' environment setting must be disabled. If this setting is enabled, users might encounter the following error when attempting to create or modify a prompt column:

"This clustered environment doesn't allow unmanaged customizations on 'msdyn_AIConfiguration' component. This was a choice made by your admin, and certain actions won't be available or will be view only." For more information about this environment setting, go to Block unmanaged customizations

Related articles

For information about how the AI is used with this feature, go to FAQ for prompts and text generation capabilities

---

Last updated on 03/20/2026

Work with formula columns

Formula columns are columns that display a calculated value in a Microsoft Dataverse table. Formulas use Power Fx, a powerful but human-friendly programming language. Build a formula in a Dataverse formula column the same way you would build a formula in Microsoft Excel. As you type, Intellisense suggests functions and syntax, and even helps you fix errors.

Add a formula column

1. Sign in to Power Apps at https://make.powerapps.com.
2. Select **Tables**, and then select the table where you want to add a formula column. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the **Columns** area, and then select **New column**.
4. Enter the following information:
   - A **Display name** for the column.
   - Optionally, enter a **Description** of the column.
5. For **Data type** select ***fx* Formula**.
6. Type the formula or use formula suggestions:

   Type a formula

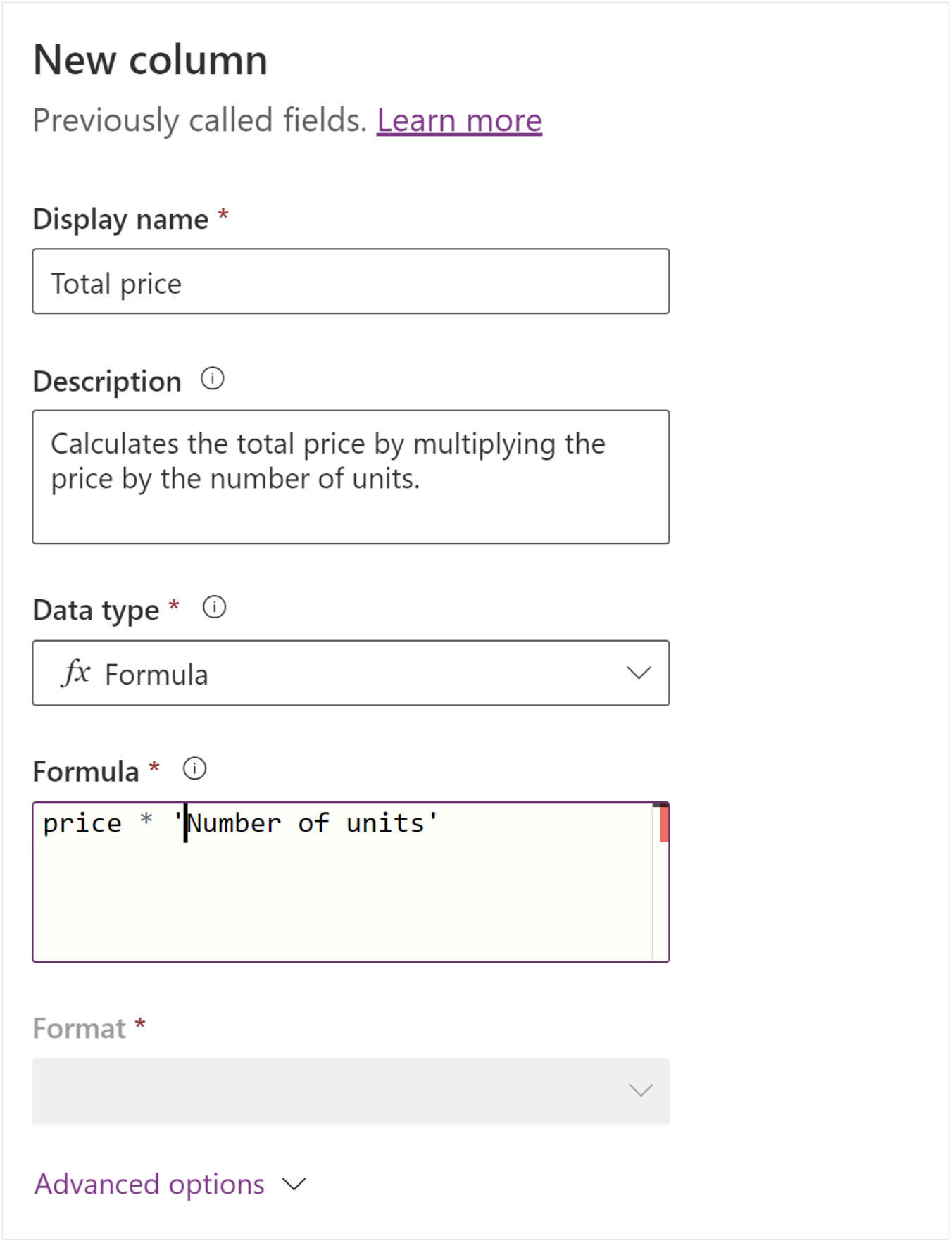
   Enter the Power Fx formula in the **Formula** box. More information: Type a formula

---

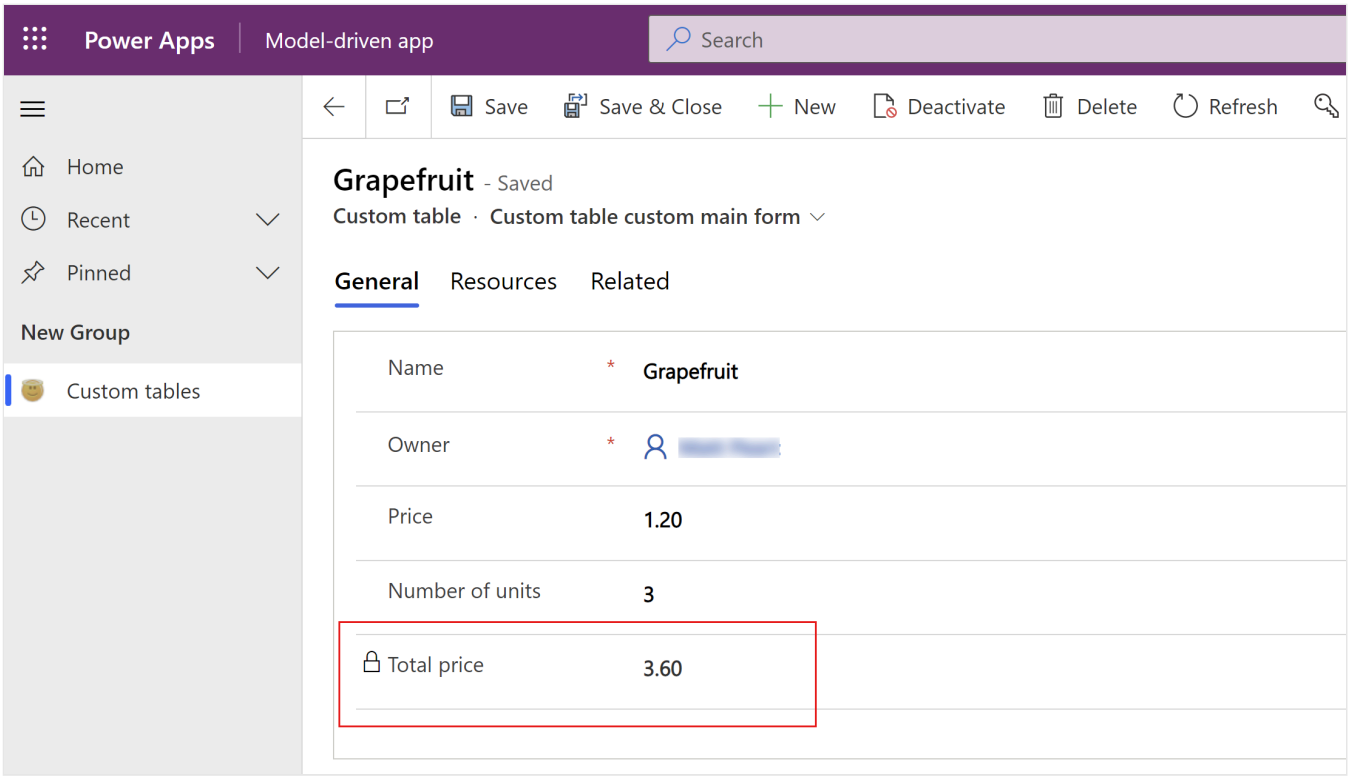
7. Select additional properties:
   - Select **Searchable** if you want this column to be available in views, charts, dashboards, and Advanced Find.
   - **Advanced options**:
     - If the formula evaluates to a decimal value, expand **Advanced options** to change the number of points of precision, between 0 and 10. The default value is 2.
8. Select **Save**.

Type a formula

The following example creates a formula column called *Total price*. The *Number of units* column is a whole number data type. The *Price* column is a decimal data type.



The formula column displays the result of *Price* multiplied by *Number of units*.



The formula that you enter determines the column type. You can't change a column type after the column is created. That means you can change the formula after you create the column only when it doesn't change the column type.

For example, the formula *price * discount* creates a column type of number. You can change *price * discount* to *price * (discount + 10%)* because that doesn't change the column type. However, you can't change *price * discount* to *Text(price * discount)* because that would require changing the column type to string.

Get formula suggestions (preview)

[This topic is pre-release documentation and is subject to change.]

Describe what you want the formula to do and get AI generated results. Formula suggestions accept your natural language input to interpret and suggest a Power Fx formula using GPT-based AI model.

> **Important**
>
> This is a preview feature available only in US regions only.
>
> Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Currently, formula suggestions that reference a single table are supported. Formula suggestions that reference a column on a related table aren't supported.

Prerequisites

To enable this feature, you must enable the **AI suggestions for formula columns** environment setting. More information: AI suggestions for formula columns

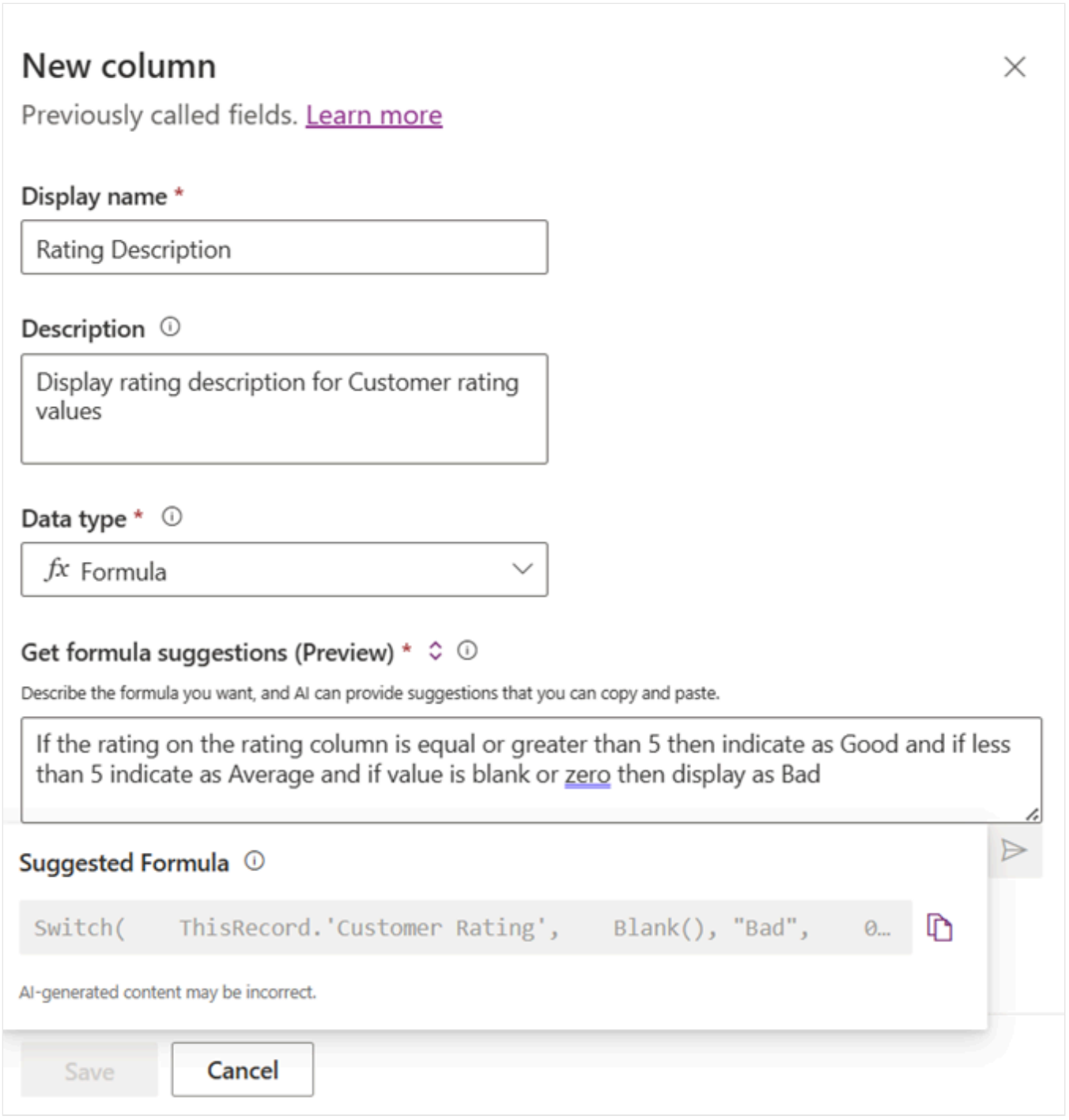
Example natural language input

Imagine there's a **Customer rating** column that shows their rating by account.

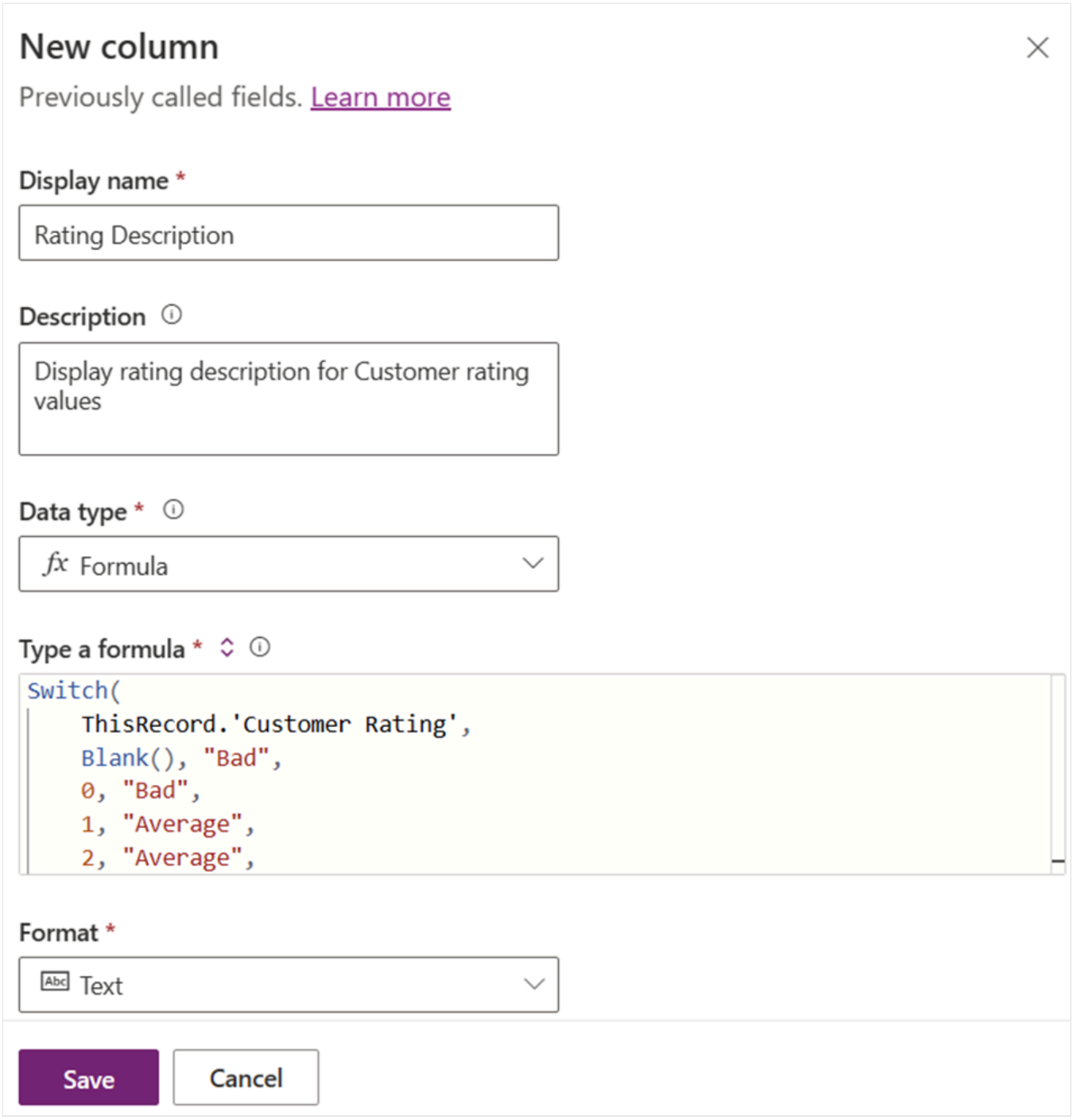


In the **Get formula suggestions** box enter the formula in natural language, such as *If the rating on the rating column is equal or greater than 5 then indicate as Good and if less than 5 indicate as Average and if value is blank or zero then display as Bad*, and then select the arrow button (enter).

Then copy the **Suggested Formula**.



And paste it into the **Type a formula** box. Select **Save**.



Here's how the formula appears when pasted.

PowerApps Formula

```
Switch(
    ThisRecord.'Customer Rating',
    Blank(), "Bad",
    0, "Bad",
    1, "Average",
    2, "Average",
    3, "Average",
    4, "Average",
    5, "Good",
    6, "Good",
    7, "Good",
```

```
    8, "Good",
    9, "Good",
    10, "Good"
)
```

Check the computed **Rating Description** formula column, which appears like this.

Customer Rating	Rating Description
5	Good
3	Average
5	Good
4	Average
2	Average
1	Average
0	Bad

Responsible AI

For information about responsible AI, go to these resources:

- FAQ for building apps and tables through conversation
- FAQ about using AI responsibly in Power Apps

Operators

You can use the following operators in a formula column:
+, -, *, /, %, in, exactin, &

For more information, go to Operators in Power Apps.

Data types

You can display the following data types in a formula column:

- Text
- Decimal Number
- Whole Number
- Float
- Boolean Choice (Yes/No)
- Choice (formerly option sets)
- Datetime

More information: Create formula columns with decimal, whole number, float, and choice data types\

The currency data type isn't currently supported.

Function types

You can use the following function types in a formula column:

- Decimal
- String
- Boolean
- Choice
- DateTime (TZI)
- DateTime (User local) (limited to comparisons with other user local values DateAdd, and DateDiff functions)
- DateTime (Date only) (limited to comparisons with other date-only values, DateAdd, and DateDiff functions)
- Currency
- Whole Number

Functions

For the scalar functions you can use in a formula column, go to Formula reference - Dataverse formula columns.

* The **Text** and **Value** functions only work with whole numbers, where no decimal separator is involved. The decimal separator varies across locales. Since formula columns are evaluated without locale knowledge, there's no way to properly interpret or generate the decimal separator.

* **StartOfWeek** argument isn't supported for the **WeekNum** and **Weekday** functions in formula columns.

Function example

Expand table

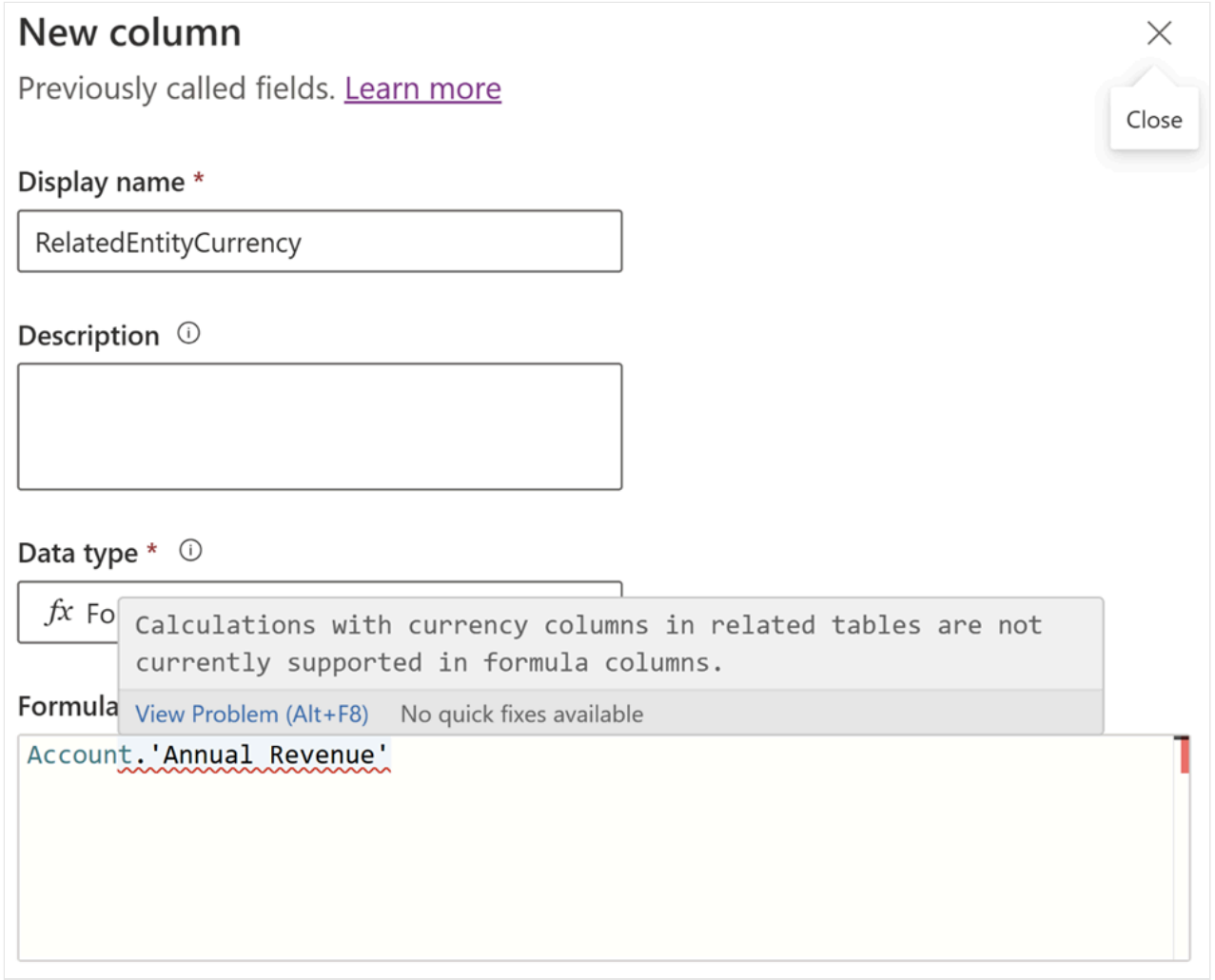
Description	Example
Retrieve a date value.	`DateAdd(UTCNow(),-1,TimeUnit.Years)`

Guidelines and limitations

This section describes guidelines and the known limitations with formula columns in Dataverse.

Currency fields usage validations

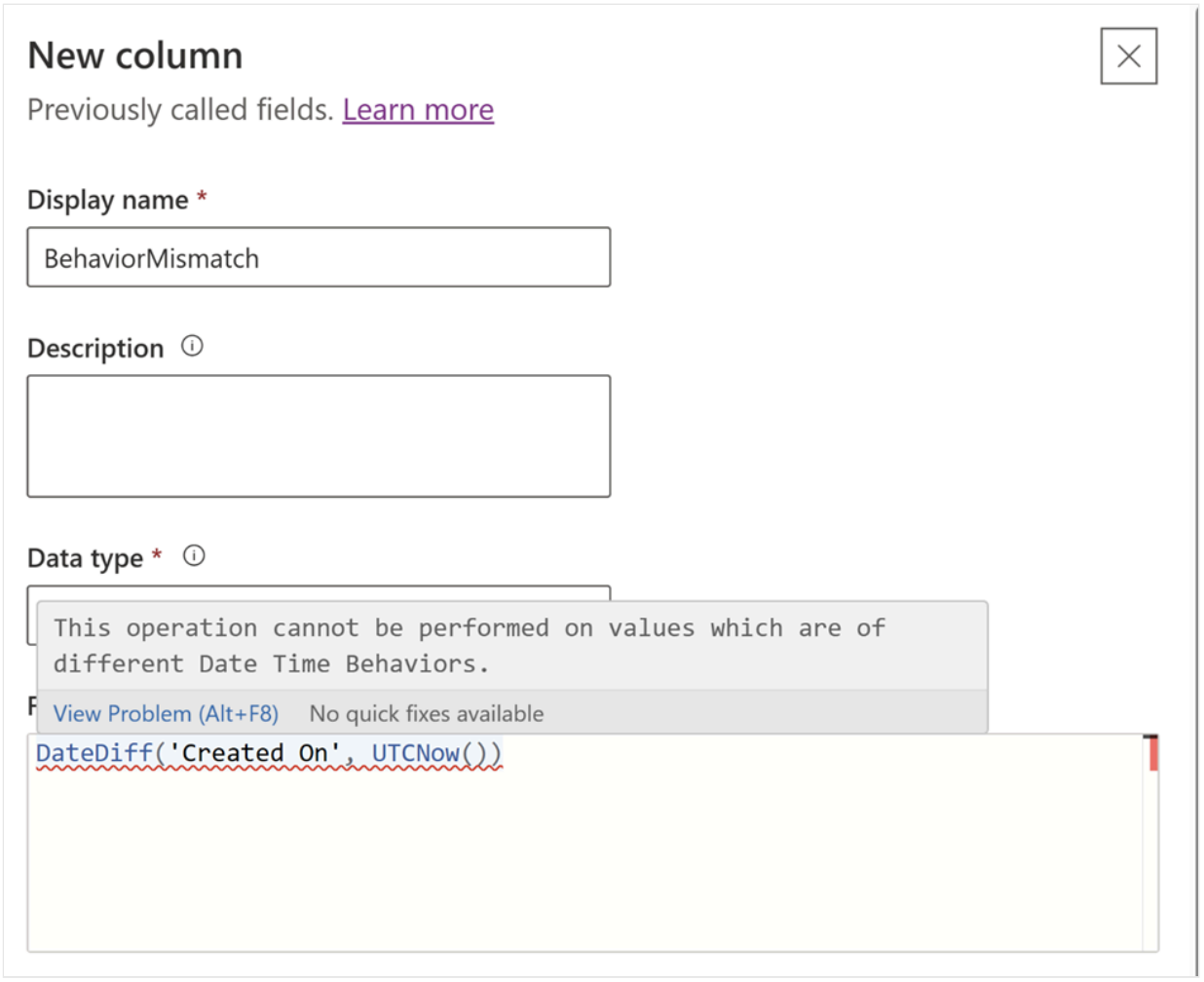
- Formula columns don't support using a related table currency column in the formula, such as in this example.



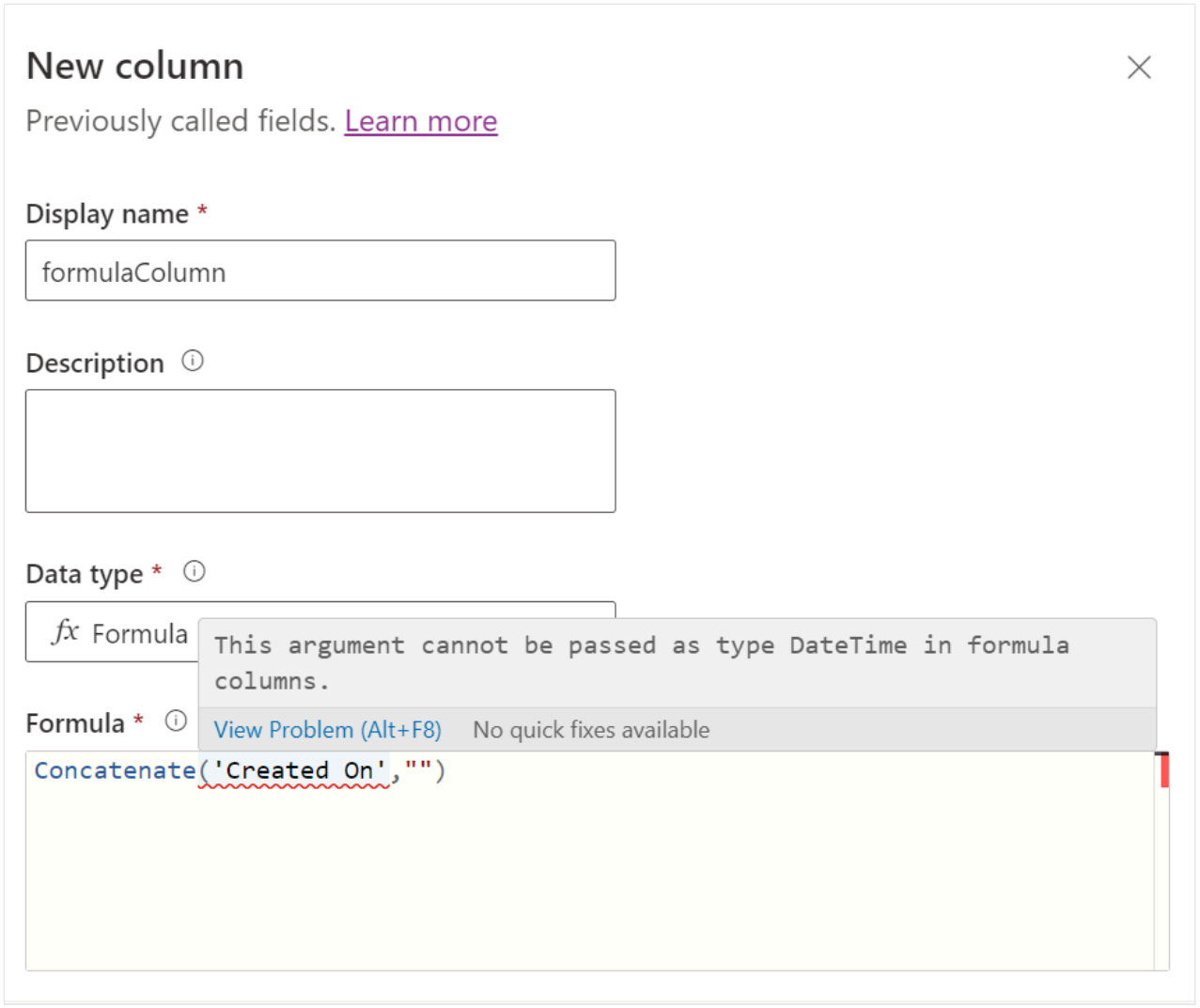
- Direct use of currency columns and exchange rate in the formula is currently unsupported. The use of currency and exchange rate columns is achieved through the `Decimal` function, such as `Decimal(currency column)` or `Decimal(exchange rate)`. The `Decimal` function makes sure the output is within the accepted range. If the currency or exchange rate column value exceeds the accepted range, then the formula returns null.
- Base currency columns aren't supported in the formula column expressions because they're system columns used for reporting purpose. If you want a similar result, you can use a currency column type along with an exchange rate column combination as `CurrencyField_Base = (CurrencyField / ExchangeRate)`.

Date time columns usage validations

- Behavior of date time formula columns can only be updated when it isn't used in another formula column.
- For date time formula columns, while using the `DateDiff` function, make sure that:
  - User local behavior column can't be compared or used with a `DateTime(TZI)/DateOnly` behavior column.
  - User local behavior columns can only be compared or used with another user local behavior column.
  - `DateTime(TZI)` behavior columns can be compared or used in `DateDiff` functions with another `DateTime(TZI)/DateOnly` behavior column.
  - `DateOnly` behavior columns can be compared or used in DateDiff function with another `DateTime(TZI)/DateOnly` behavior column.



- Date time columns and date time functions `UTCNow()`, `Now()` can't be passed as a parameter to string functions.



Formula column usage in rollup fields

- A *simple formula column* is where the formula uses columns from the same record or uses hard coded values. For rollup columns, formula columns must be simple formula columns, such as this example rollup column.

Display name *

formulaColumn

Description

Data type *

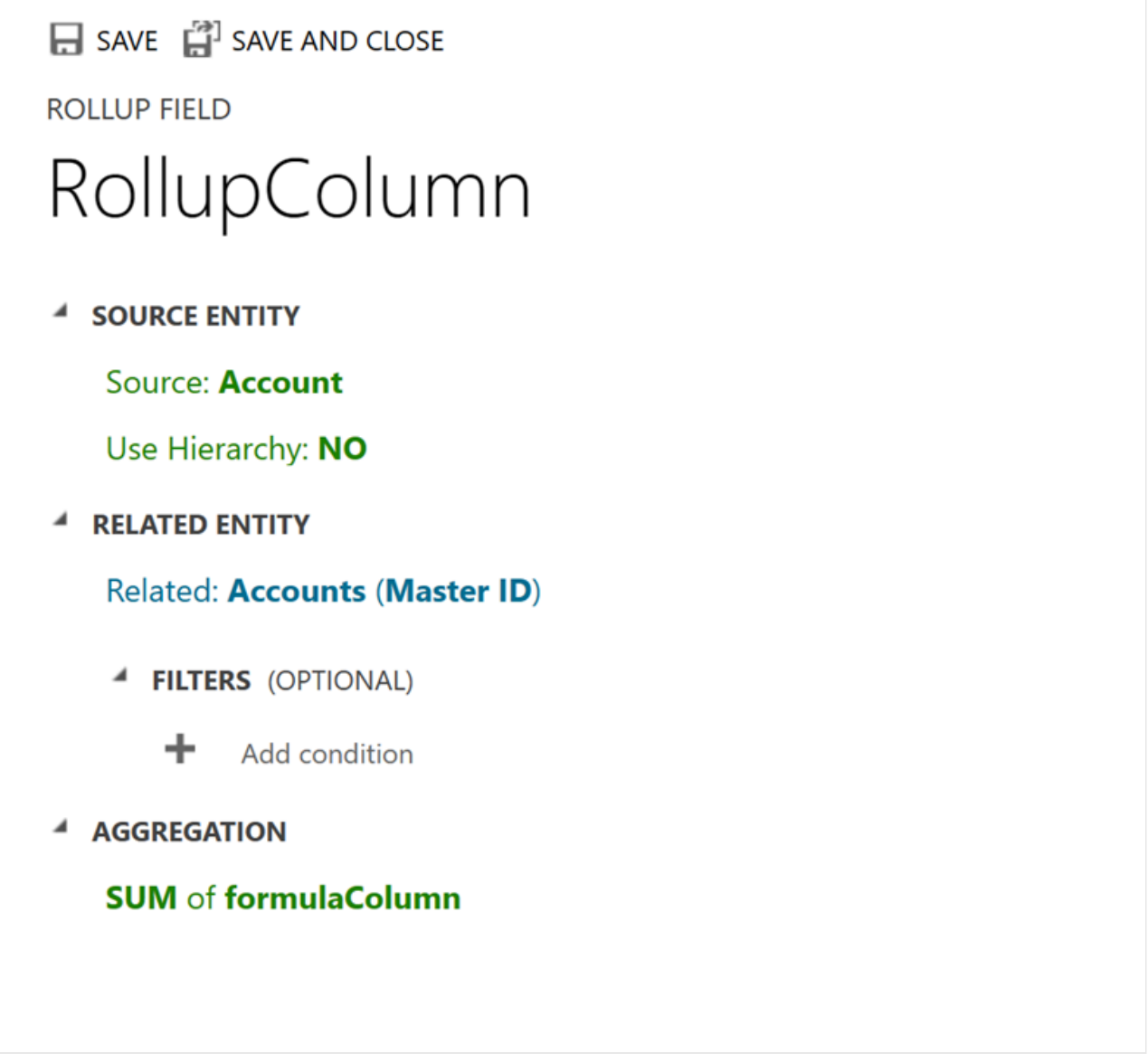
Decimal fx

Formula *

1+2

Searchable

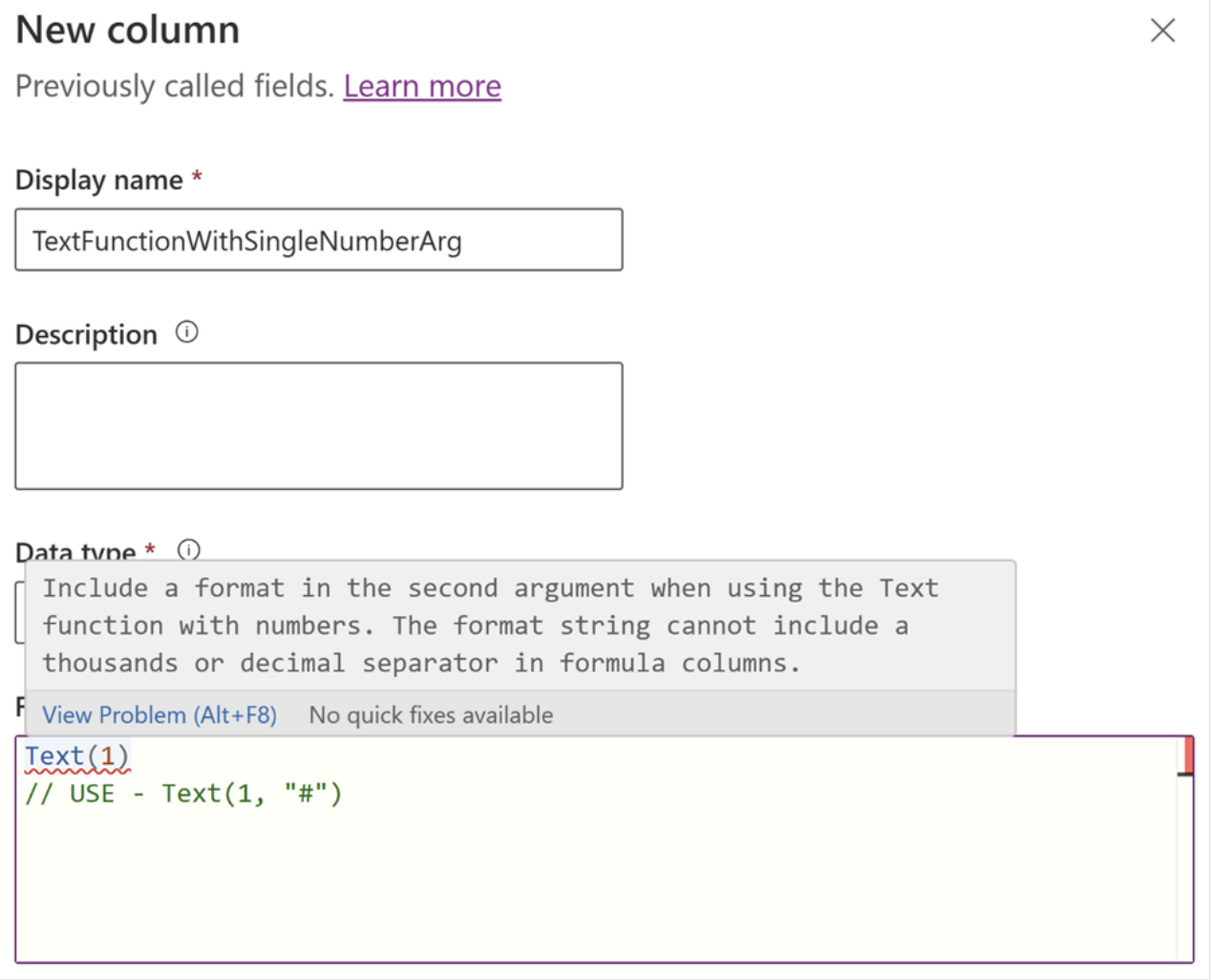
Advanced options



- A formula column, which is dependent on time bound functions `UTCNow()` and `UTCToday()` can't be used in a rollup field.

Power Fx text function recommendations

- Formula columns don't support `Text()` functions with a single argument of type Number. Number can be whole, decimal, or currency.

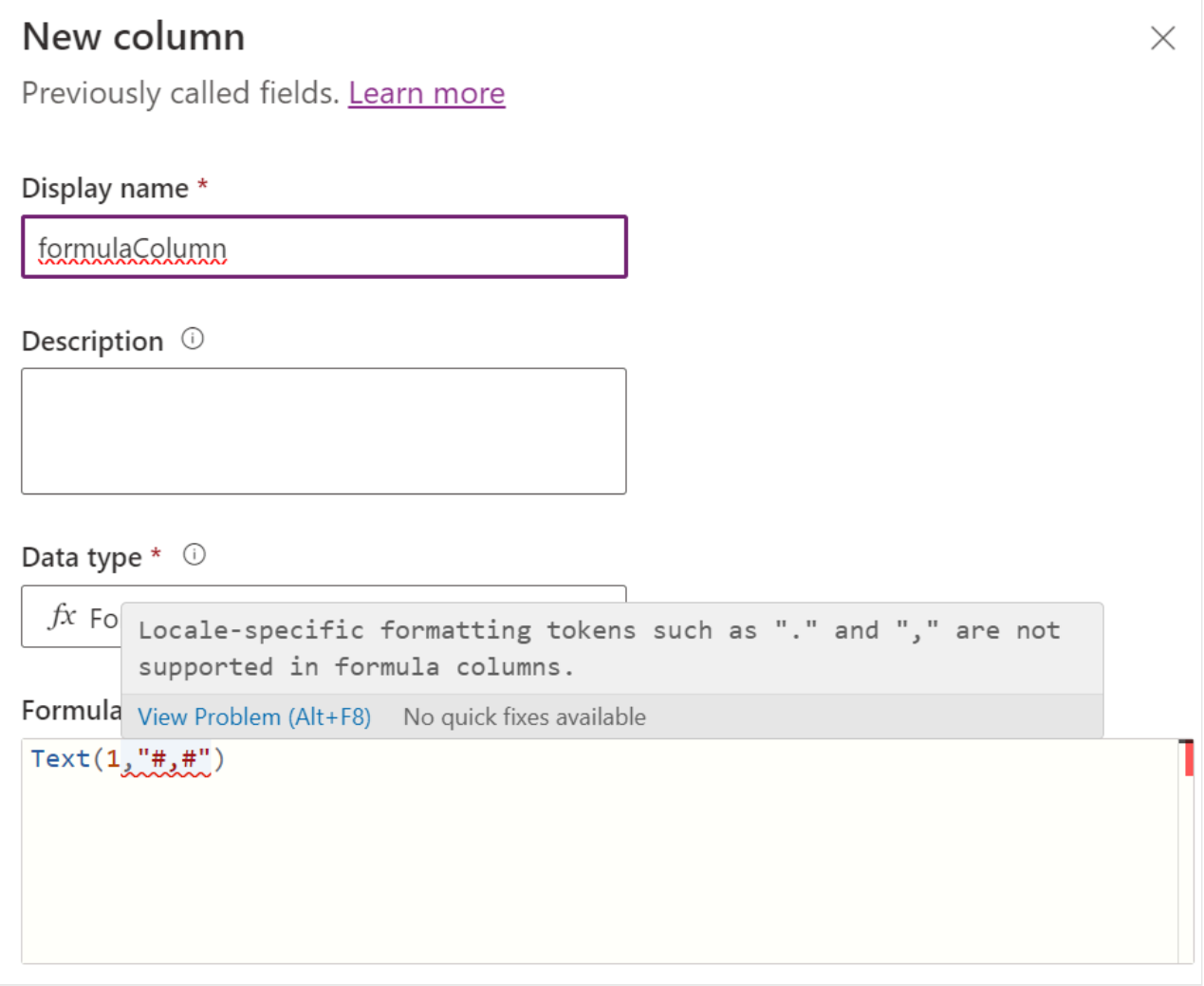


- Formula columns don't support using numbers in the following configurations:
  - In string functions. These are string functions placed wherever a text argument is expected: Upper, Lower, Left, Right, Concatenate, Mid, Len, StartsWith, EndsWith, TrimEnds, Trim, Substitute, and Replace.
  - In the implicit formulas, such as `12 & "foo"`, or `12 & 34`, or `"foo" & 12`.
  - Internal number to text coercion isn't supported. We recommend using `Text(Number, Format)` to convert a number to text. In the case where a `String` argument is passed in a `Text` function then the `Format` argument isn't supported.
  - Here's an example using the `Text` function to convert a number to text and append a string to it:

PowerApps Formula

```
Concatenate(Text(123,"#"),"ab")
Text(123,"#") & "foo"
```

- Locale-specific formatting tokens such as "." and "," aren't supported in formula columns.



Range validations on formula columns

- You can't set the **Minimum value** or **Maximum value** properties of a formula column.
- All internal computations should lie within the Dataverse range for decimal type formula columns (-100000000000 to 100000000000).
- A hard coded literal value entered in the formula bar should lie within the Dataverse range.
- If there's a numeric column that's null then it's considered 0 in the intermediate operation. For example, `a+b+c and If a = null, b=2, c=3` then formula column gives `0 + 2 + 3 = 5`.
  - This behavior is different from calculated columns in this case because calculated columns give `null + 2 + 3 = null`.

General validations on formula columns

- Formula columns can reference other formula columns, but a formula column can't reference itself.

- Formula columns don't support cyclic chains, such as `F1 = F2 + 10, F2 = F1 * 2`.
- Maximum formula expression length in formula columns is 1,000 characters.
- The maximum depth allowed in formula columns is 10. *Depth* is defined as the chain of formula columns referring to other formula or rollup columns.
  - For example, `table E1, F1 = 1*2, table E2, F2 - E1*2`. In this example, the depth of F2 is 1.
- In model-driven apps, sorting is disabled on:
  - A formula column that contains a column of a related table.
  - A formula column that contains a logical column (for example, address column).
  - A formula column that contains another calculated or formula column.
  - A formula column that uses time-bound function `UTCNow()`.
- Columns of type Whole Number with format Language, Duration, Time Zone aren't supported in formula columns.
- Columns of type String with format Email, Text Area, Ticker Symbol, URL aren't supported in formula columns.
- Formula columns don't display values when the app is in mobile offline mode.
- You can't trigger workflows or plug-ins on formula columns.
- We don't recommend using calculated columns in formula columns and vice versa.
- Duplicate detection rules aren't triggered on formula columns.
- The `Now` function can be used with formula columns. `Now()` has user local behavior and `UTCNow()` has time zone independent behavior.
- You can set the precision property for decimal columns.
- Default formula data type value is set to **Decimal** for numeric value returning formulas.
- Updating whole number formula column's format isn't supported.

Formula columns of data types that can't be produced

- Currency

See also

Types of columns

Microsoft Power Fx overview

Specialized columns

Last updated on 01/09/2026

Create formula columns with decimal, whole number, float, and choice data types

This article provides guidance on creating and using different data types in Microsoft Dataverse formula columns, such as decimal, whole number, floating point, and choice columns. It outlines steps starting with selecting **fx Formula** as the data type and entering a numeric value-returning formula in the formula bar.

Start by creating a column for a table

1. Sign in to Power Apps at https://make.powerapps.com .
2. Select **Tables**, and then select the table where you want to add a formula column. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select the **Columns** area, select **New column**, and then select the data type and enter the Power Fx formula. Depending on the formula you enter, you can create any of the following:
    - Decimal formula column
    - Whole number formula column
    - Floating point number formula column
    - Choice formula column

Create a decimal formula column

Create a formula column that returns a decimal number.

1. Select **Data type** as ***fx* Formula**.
2. Enter a formula that returns a numeric value in the **Formula** bar. This example creates a formula column called *Total Amount*. The *Price Per Unit* column is of decimal data type.

New column

Previously called fields. Learn more

Display name *

Total Amount

Description

Data type *

fx Formula

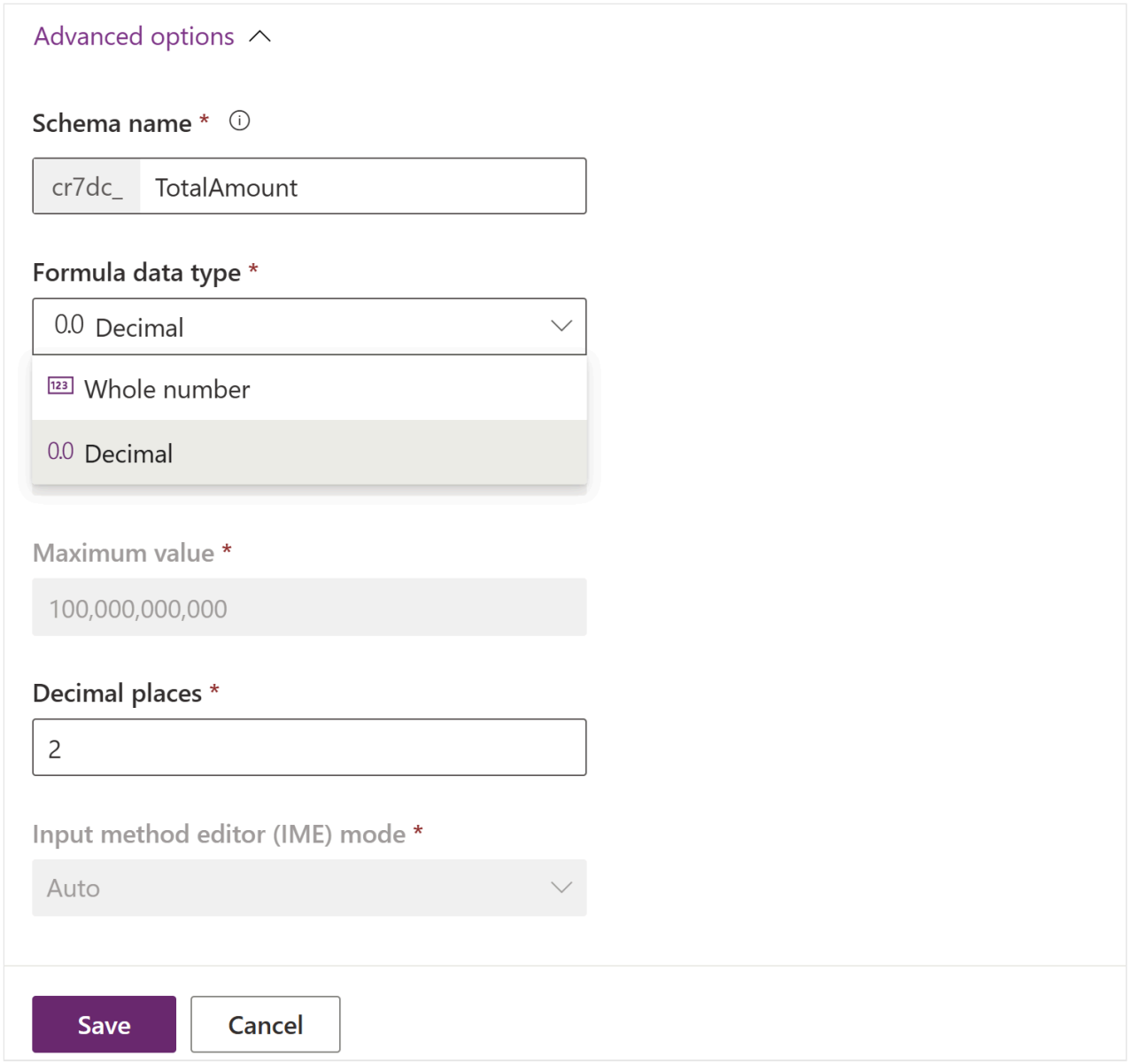
Formula *

```
'Price Per Unit' * 150
```

Searchable

Advanced options

3. Expand **Advanced options**, and then set the required number of decimal places.



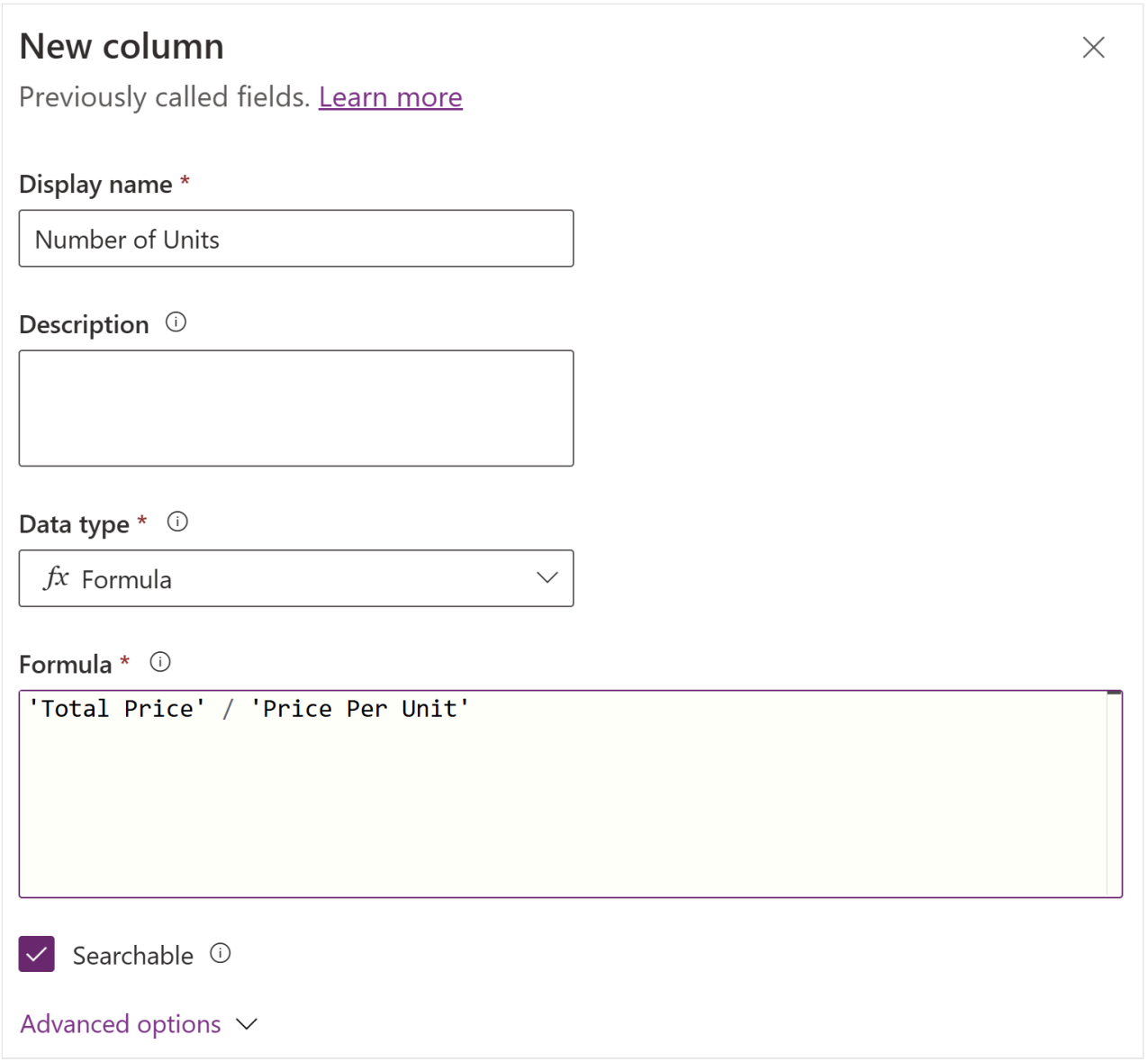
4. Select **Save**. By default, **Formula data type** is set to **Decimal** and a decimal formula field is created.

Create a whole number formula column

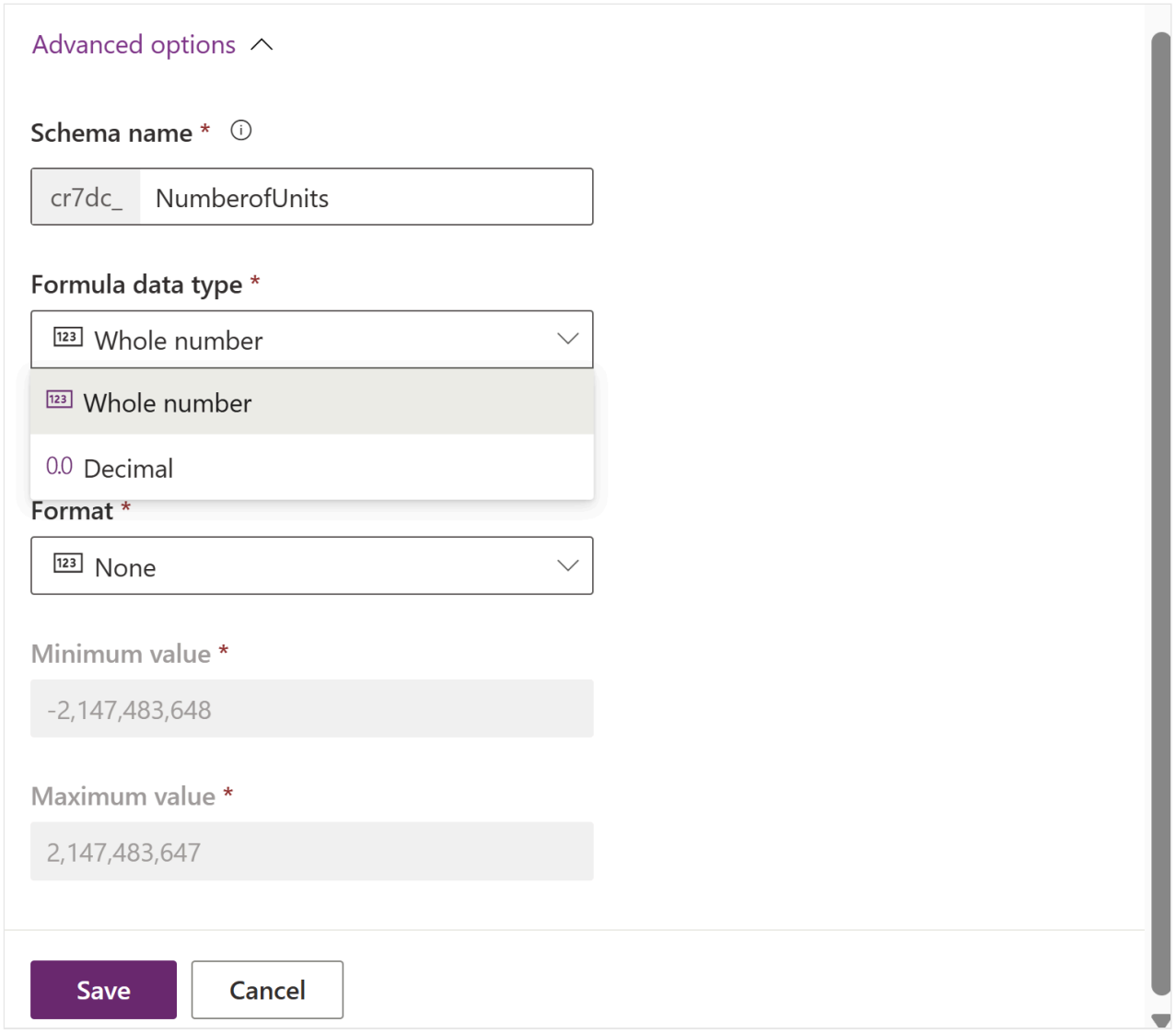
Create a formula column that returns a whole number.

1. Select **Data type** as ***fx* Formula**.
2. Enter a formula that returns a numeric value in the **Formula** bar. This example creates a formula column called *Number of Units*. *Total Price* and *Price Per Unit* columns are of

decimal data type.



3. Expand **Advanced options**, and select **Whole number** as the **Formula data type** and set the required format for whole number column.



4. Select **Save**.

Create a floating point number formula column

Create a formula column that returns float.

1. Select **Data type** as ***fx* Formula**.
2. Enter a formula that returns a floating point number in the **Formula** bar. Enter a formula that returns a floating point number in the **Formula** bar. This example creates a formula column called *Total Price*. *Weight* is a simple float column and *Price Per Gm* is a simple

decimal column.

New column

Previously called fields. Learn more

Display name *

Total Price

Description

Data type *

fx Formula

Formula *

```
Weight * 'Price Per Gm'
```

Searchable

Advanced options

Schema name *

cr7dc_ TotalPrice

Minimum value *

-100,000,000,000

Maximum value *

100,000,000,000

Decimal places *

3

Save Cancel

3. Expand **Advanced options**, and set the required number of decimal places.
4. Select **Save**.

Guidelines for creating floating point number formula columns

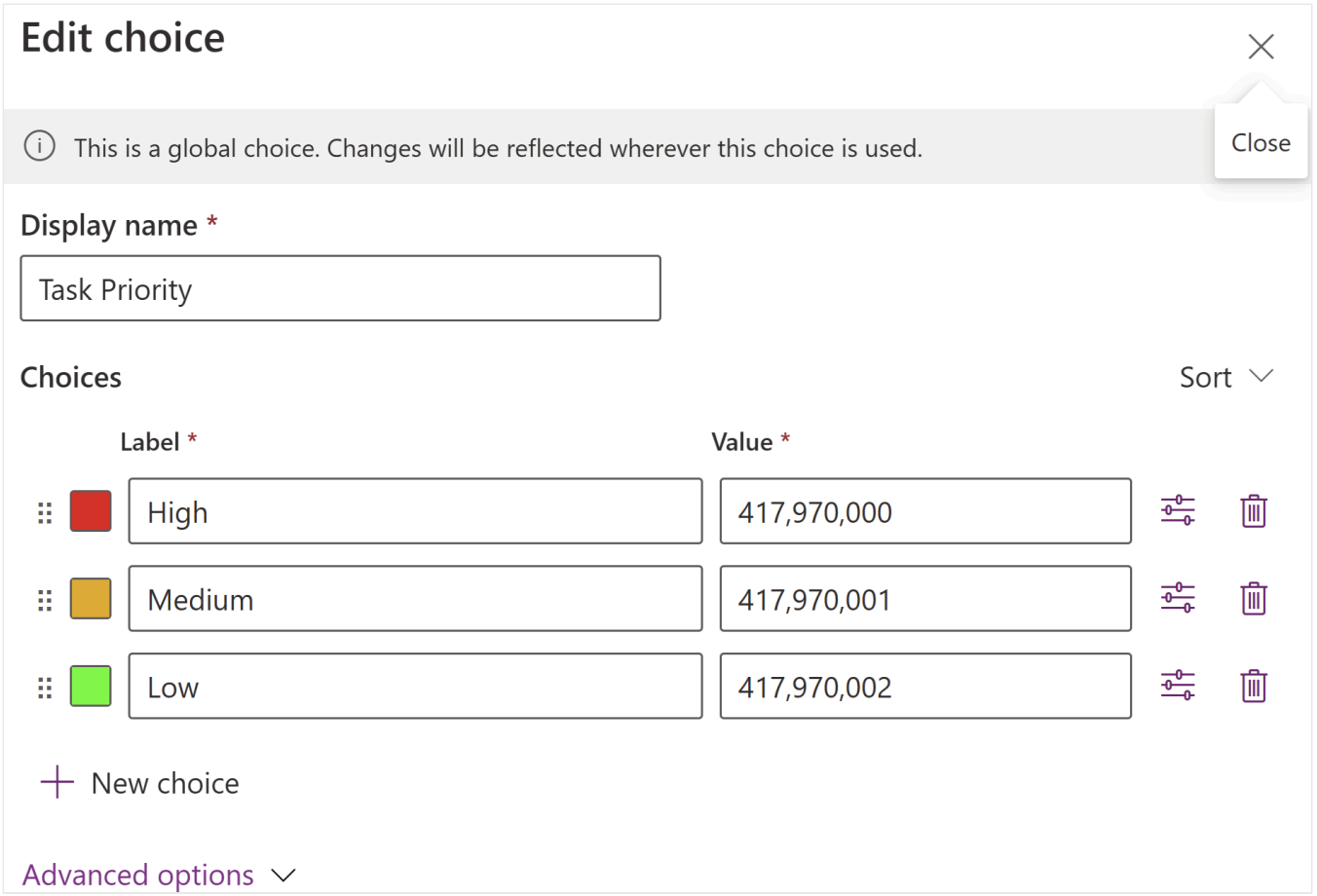
- If an operand involved in an arithmetic operation is of float type, then the result of the formula is of float type. For example:
  - `1 + 2 + Float(1)` as it uses a float type operand - `Float(1)`.
- A numeric function returns a float value when the first parameter provided to the function is of float type. Otherwise, the function returns a decimal value. For example:
  - `Sum(1, 2, Float(1))` is of decimal type and `Sum(Float(1), 1, 2)` is of float type.
- `Float`, `Sqrt`, `Ln`, `Power`, `Exp` functions and the `^` operator return a float value.

Create a choice formula column

To create a choice formula column, either global choice or local choice of a simple choice column can be used as result.

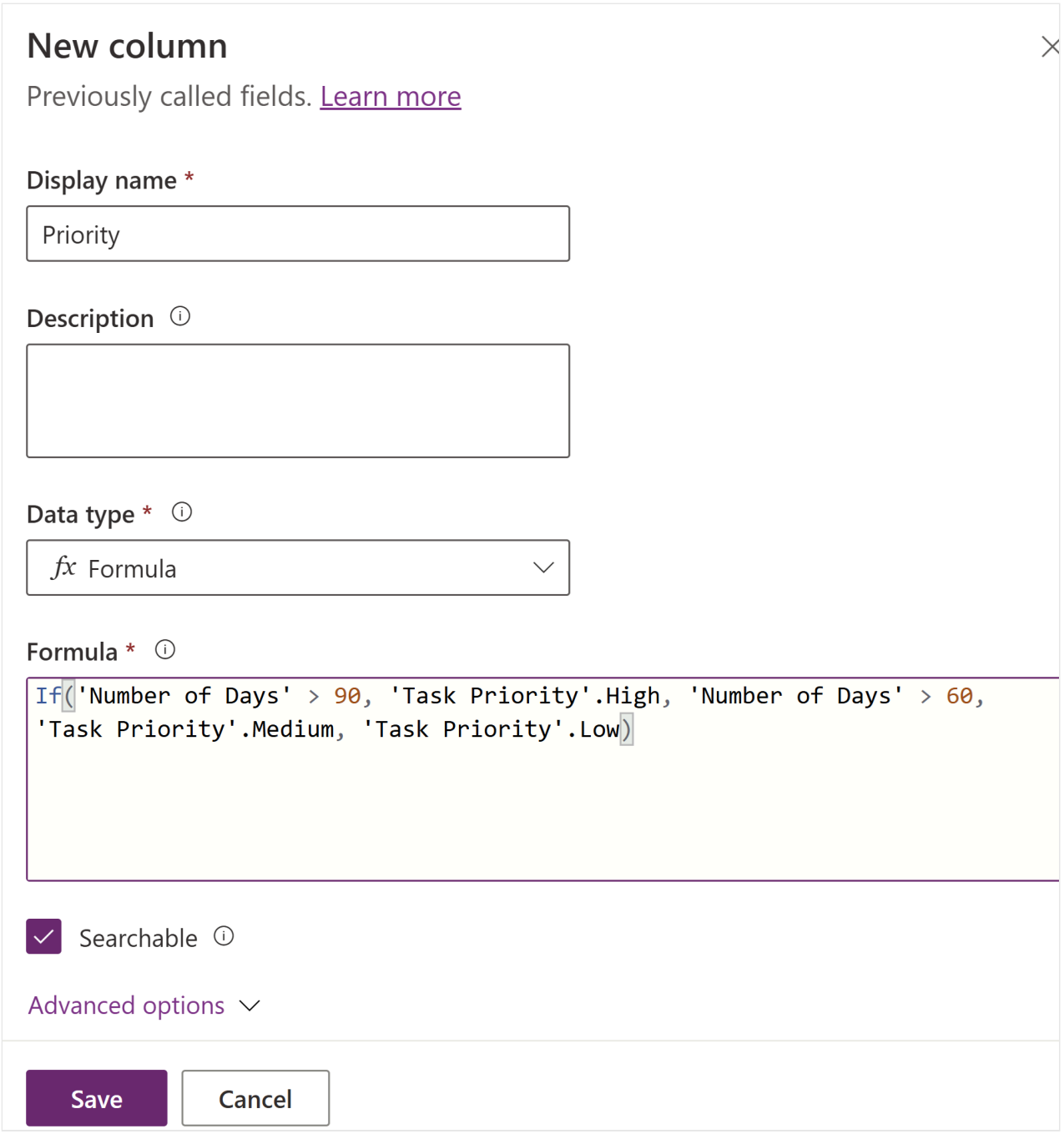
Using global choice

Create a global choice. This example creates a global choice called *Task Priority*.



Create a formula column that returns a choice using a global choice.

1. Select **Data type** as ***fx* Formula**.
2. Enter a formula that returns a choice value in the **Formula** bar. This example creates a formula column *Priority* using global choice *Task Priority*.



3. Select **Save**. Notice that the column created is of data type **Choice fx**.

Using local choice from a simple choice column

Create a simple choice column. This example creates a *Task Priority* simple choice column for the account table.

New column

Previously called fields. Learn more

Display name *

Task Priority

Description

Data type *

Choice

Behavior

Simple

Required

Optional

Searchable

Selecting multiple choices is allowed

Sync with global choice? *

Yes (recommended)

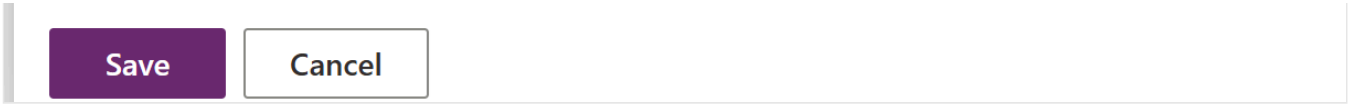
Can be used in multiple tables, and will stay updated everywhere.

No

Creates a local choice that can only be used in one table. People using it can add new choices.

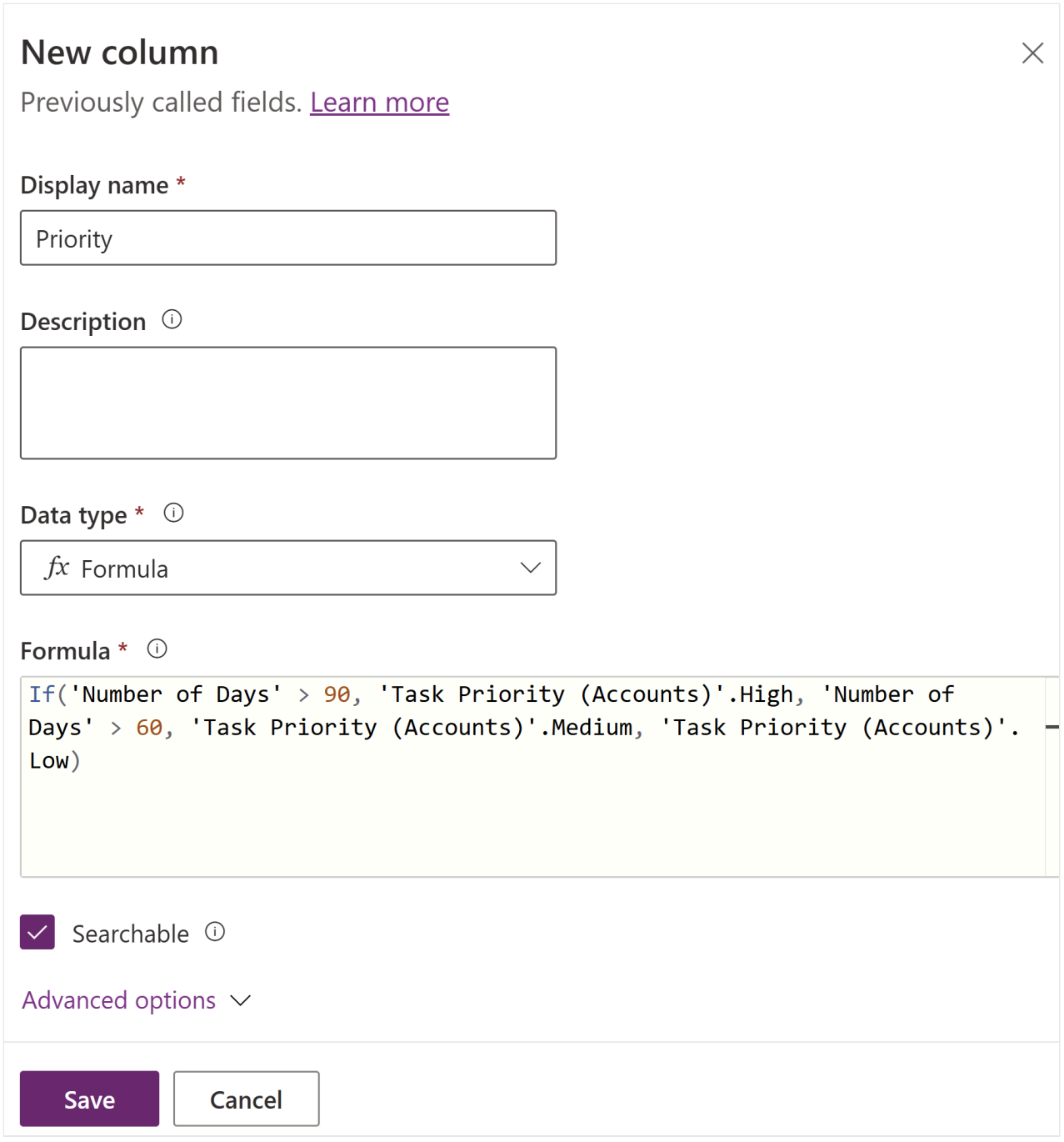
Choices Sort

Label *	Value *
High	417,970,000
Medium	417,970,001
Low	417,970,002



Create a formula column that returns choice using a local choice of a simple choice column.

1. Select **Data type** as ***fx* Formula**.
2. Enter a formula that returns a choice value in the **Formula** bar. This example creates a formula column *Priority* on *Account* entity using local choice of a choice column *Task Priority* for the account table.

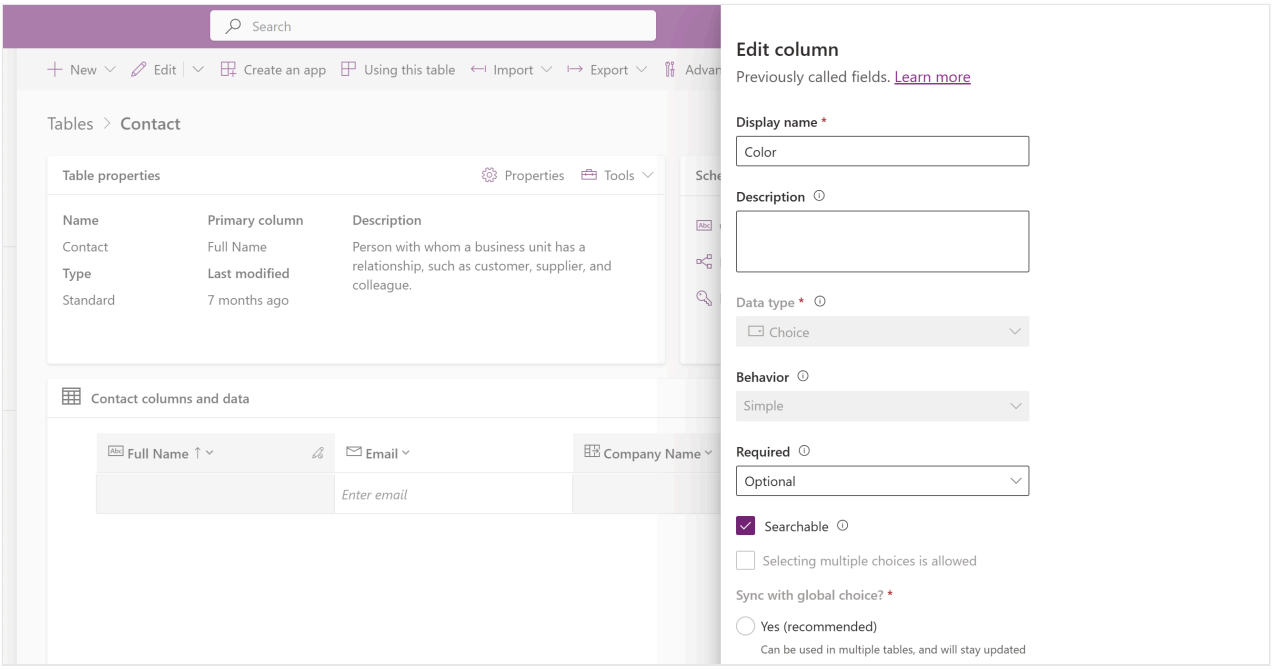


3. Select **Save**.

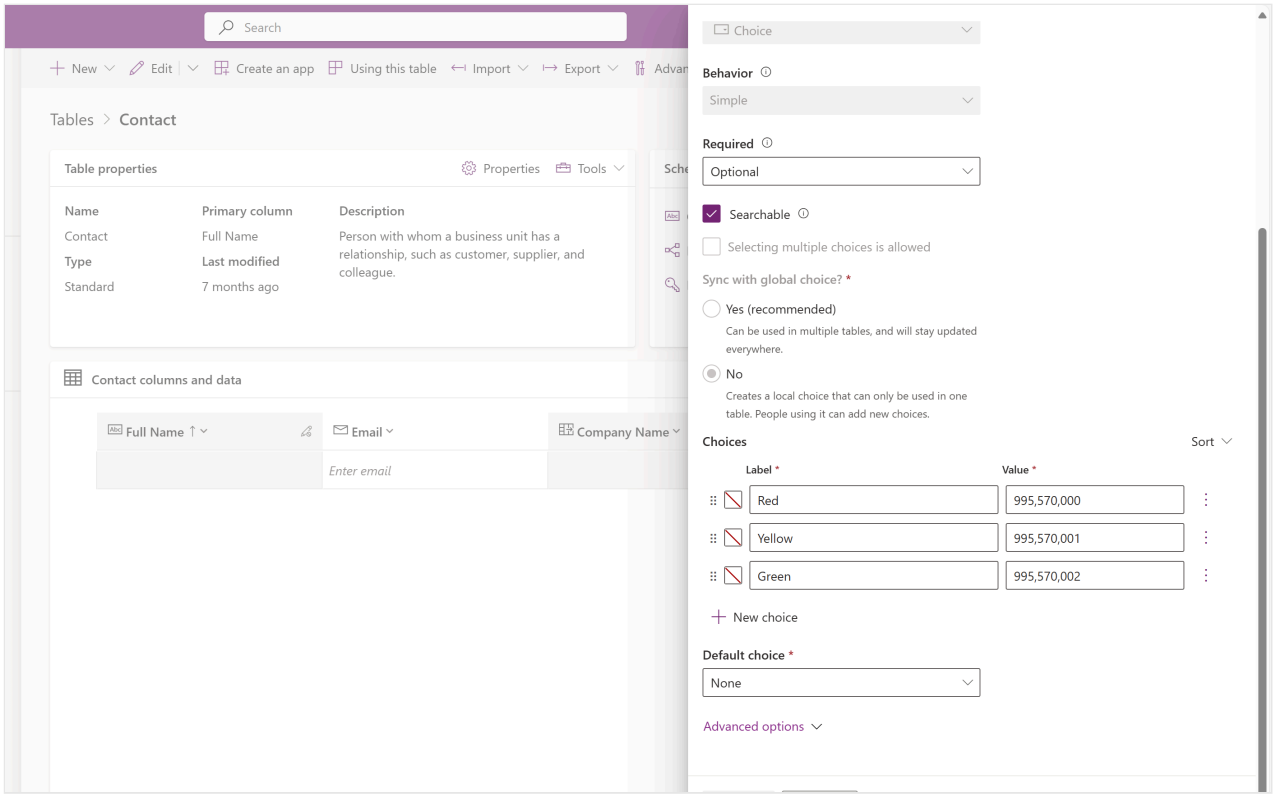
Guidelines for working with choices in formula columns

- Local choices of related table's simple choice column can't be used as a result type in formula columns.
- Options from the same option set should be used for all result arguments in choice formula columns.
- A choice used by a formula column can't be updated.
- Options of a choice can't be passed as an argument to string functions. Value function can be used to return the numeric value of an option.
- Formula column's dependent local choice column or global choice can't be deleted.
- For using a related table local choices (optionset) column's options in the formula column definition, first use choice and then use options of that local choice.

  For example, a choice column named *Color* on the **Contact** table.



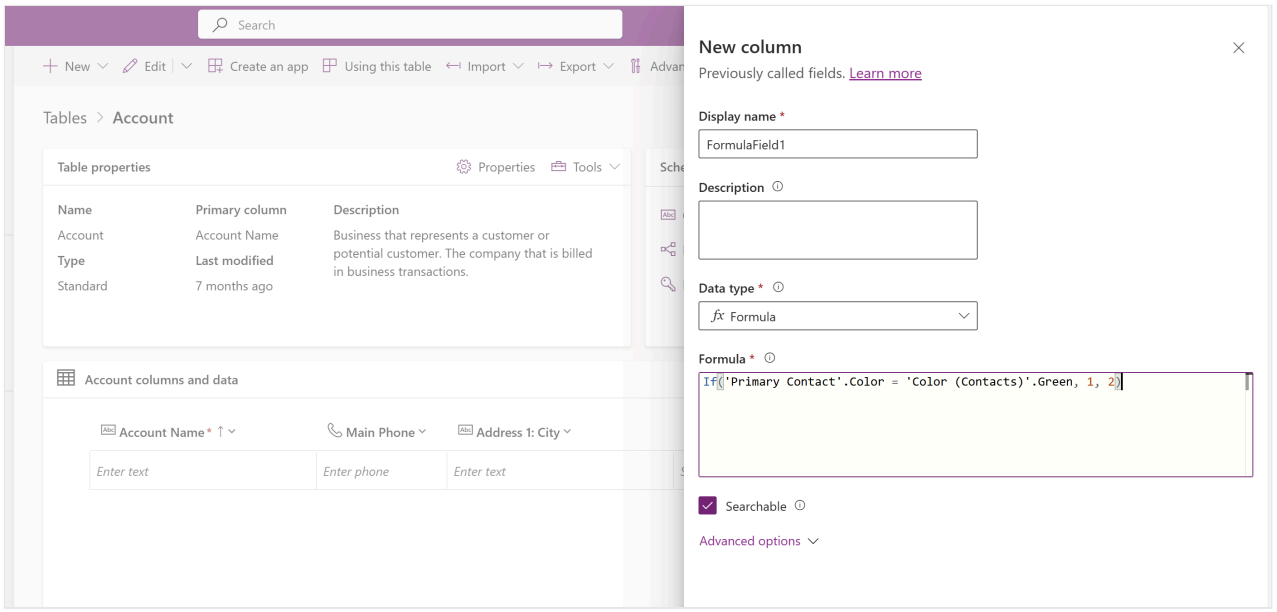
  The choice column has options Red, Yellow, and Green.



For a formula column on the account table using the *Color* choice column, the formula appears like this:

Recommended - `If(ParentAccount.Color == 'Color (Accounts)'.Red, 1, 2)`

Not Recommended - `If( 'Color (Accounts)'.Red == ParentAccount.Color, 1, 2)`



See also

[Using the right type of number]

[Fx Formula Columns]

[Microsoft Power Fx overview]

Specialized columns

Create formula columns with decimal (video)

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 01/09/2026

Set managed properties for columns

Article • 04/10/2025

Managed properties only apply when you export columns in a managed solution and import the solution into another environment. These settings allow a solution maker to have some control over the level of customization that people who install their managed solution can have when they customize this column.

To view and set managed properties for a column, follow these steps:

1. Sign into Power Apps and select **Solutions** from the left navigation. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the solution, open the table you want, and then open the **Columns** area.
3. Select the column you want.
4. On the command bar select **More Actions**, and then select **Managed properties**.
5. Select the managed properties options you want, and then select **Set**.

Managed properties options

The **Allow customizations** option controls all the other options. If this option is `False`, none of the other settings apply. When it is `True`, you can specify the other customization options.

If the column is customizable, you set the following options to `True` or `False`.

- **Display name can be modified**
- **Can change requirement level**
- **Can change additional properties**

These options are self-explanatory. If you set all the individual options to `False`, you might as well set **Allow customizations** to `False`.

Next steps

Create and edit columns

Solutions overview

Related information

Set managed properties for columns with Power Apps (video)

Email address validation for email columns (preview)

07/02/2025

> **Important**
>
> - This feature is deprioritized and will not be delivered.
> - Beginning July 31, 2025, existing model-driven or canvas apps that use this email validation feature will stop displaying the validation results for the email addresses entered in forms. Any custom function or logic implemented by makers that use this specific email address control type might stop functioning and must be updated to use the standard email address control type. If you haven't developed custom functions, forms continue to work as before.

[This topic is pre-release documentation and is subject to change.]

Get email address columns validated automatically in model-driven apps with no-code. With smart data validation, makers can build smarter and contextually aware next-gen apps for their workflows with better data quality.

Traditionally, the existing email column type had only basic email validation. With smart email address validation, the email column format comes with better validation that includes reasons for the column being invalid. Any model-driven apps using a text column with email format receives automatic validation, which is helpful in guiding users to enter better email data.

> **Important**
>
> - This is a preview feature.
> - During preview, these regions have email address validation feature available: Asia (East, Southeast), Australia (East, Southeast), Canada (Central, East), Europe (North, West), France (Central, South), India (Central, South), Japan (East, West), South America – Brazil (South), Switzerland (North, West), UAE (North), UK (South, West), US (East, West).
> - Smart email validation only works for model-driven apps.
> - Smart email validation shows validation issues but won't block users from saving their record.

Email address validation detects the following issues:

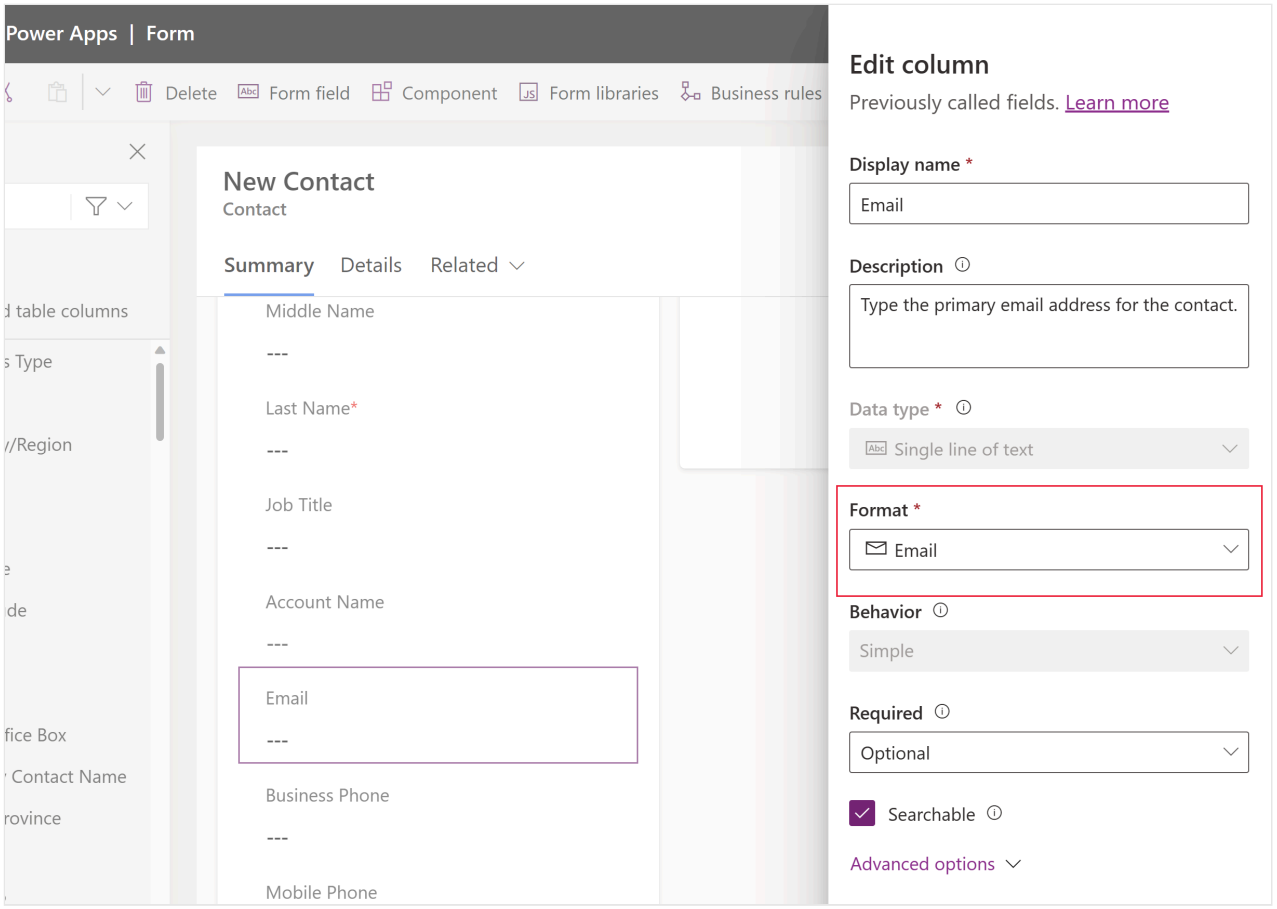
- Incorrect syntax: For example, an address that doesn't contain both a username and an email domain.
- Disposable domain: An address that contains a known disposable or temporary email domain.
- Test or spam email addresses. An address that contains known indicators of a test or spam address in the email header or metadata, IP address, HTML code of the email, and email content and formatting.
- Expired email addresses: This is an email account that has expired and can no longer receive or send email.
- Emails that bounce back: This is an address that can't receive a message for any reason other than being expired.

Prerequisites

- The **Data validation** Power Platform environment setting must be enabled. By default, this setting is disabled. More information: Manage feature settings
- Email address validation only performs validation on text columns with the **Email** format type. For example, the **Email** or **Email Address 2** columns for the account table.

Enable an email format column for address validation

1. Make sure the model-driven app has a table with a text data type column formatted for email, such as the **Email** column for the contact table.

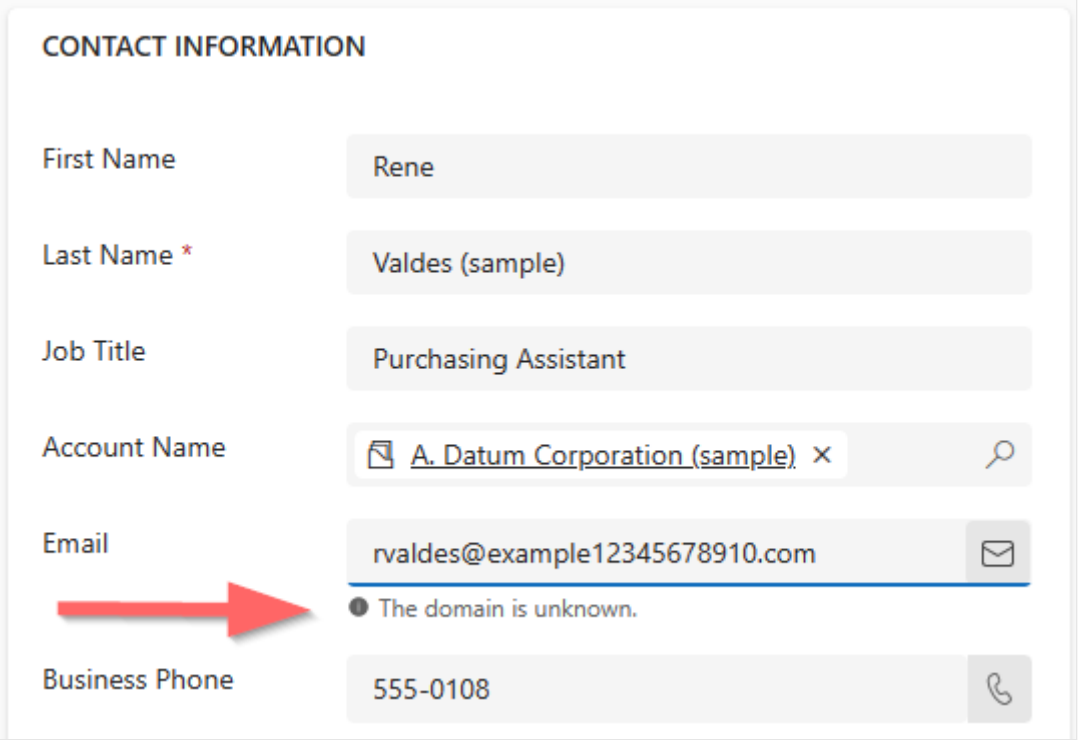


2. In the app designer, select **Settings** on the command bar.
3. Select the Upcoming tab, and then select **Enable Smart Email Address Validation Control**.
4. Select **Save**.

Test email address validation

1. In the app designer, select Play on the command bar.
2. Open a form that has the email column, such as the **Email** column for the accounts table.
3. Enter an invalid email address, such as a domain that doesn't exist, such as *example12345678910.com*, in the **Email** column.

Notice that the notification message **The domain is unknown** appears under the column.



Known issues

- The **Enable Smart Email Address Validation Control** app setting can be enabled even though the **Data Validation** Power Platform admin center environment setting is off. In this situation, email address validation doesn't work.
- When the form containing the control is set to read-only mode, the email column still allows editing of the value.
- The email column doesn't import values when using a quick create form. For example, when you create a new parent contact record from a lead record.
- While using the `setValue()` method, the value being defined for the control doesn't render in the app.

Next steps

Create and edit columns in Dataverse using Power Apps

Behavior and format of the date and time column

In Microsoft Dataverse, you can specify how date and time values are shown to users and how they're adjusted for time zones.

In addition to the standard properties available with most columns, date and time columns have two extra properties:

- **Time zone adjustment**: Whether to adjust values for time zones.
- **Format**: Whether to display the time portion of the value.

Time zone adjustment

Dataverse stores all date and time values in UTC time zone. When your app displays values or processing values entered by users, Dataverse and model-driven apps can adjust for the user's time zone with these **Behavior** options.

- **User local**: Adjust values for the user's time zone. This is the default behavior for the **Date and time** format. You can change this once to another behavior.
- **Time zone independent**: No time zone conversion. This is the default behavior for the **Date only** format.

Set the user's time zone in personal options, not the system time zone in Windows, Android, iOS, or macOS. However, the system time zone might affect client scripts that work with JavaScript Dates.

Format

All date and time columns have a time portion unless their behavior is **Date Only**. **Format** determines whether to display the time portion of the value.

- **Date and time**: Displays the date and time of the value.
- **Date only**: Displays the date portion of the value only.

> ⓘ **Note**

Users can still change the time portion if the **Format** is **Date Only**. For example, by using Web API calls or by using a control that has the time portion. This behavior is different from **Date only Behavior**, where the time portion isn't stored at all.

Usage guidelines

Use **Time zone independent** when time zone information isn't required, such as hotel check-in times. With this selection, users in all time zones see the same date and time value.

Use **Date only** when information about the time of the day and the time zone isn't required, such as birthdays or anniversaries. With this selection, users in all time zones see the exact same date value.

Time zone independent with **Date only** is practically the same as setting the column as **Date only**. Use the former if you aren't sure whether you need the time portion in the future.

> **Important**
>
> Avoid **Date only** format with **User local** behavior. Users in different time zones might see a different date, which is not intended in most scenarios. When a user sets a date in a model-driven app, the time portion automatically sets to midnight of their time zone. This might cause the date to appear a day earlier or later for other users.

Examples

Display values

Dataverse stores `2023-10-15T07:30:00Z` (or `2023-10-15` for **Date only** behavior). Users in the time zone UTC-8 see these in the model-driven app or with a Web API request for the formatted value:

Expand table

Behavior	Format	Display value
User local	Date and time	October 14th, 2023, 11:30 pm
User local	Date only	October 14th, 2023
Time zone Independent	Date and time	October 15th, 2023, 7:30 am
Time zone Independent	Date only	October 15th, 2023

Behavior	Format	Display value
Date only	-	October 15th, 2023

Enter values in an app

Users in the time zone UTC-8 enter `October 14th, 2023, 11:30 pm` in a model-driven app. The value is saved in Dataverse as:

Expand table

Behavior	Format	Value saved in Dataverse
User local	Date and time	2023-10-15T07:30:00Z
User local	Date only	2023-10-15T07:30:00Z
Time zone Independent	Date and time	2023-10-14T23:30:00Z
Time zone Independent	Date only	2023-10-14T23:30:00Z
Date only	-	2023-10-14

If the user enters just the date `October 14th, 2023`, the time portion is assumed to be 12:00 AM.

Expand table

Behavior	Format	Value saved in Dataverse
User local	Date only	2023-10-14T08:00:00Z
Time zone Independent	Date only	2023-10-14T00:00:00Z
Date only	-	2023-10-14

Enter invalid values in an app

Different clients have different ways to handle invalid input. For example, in the Pacific time zone, daylight saving started on March 12th, 2023 at 2:00 AM, moving the time forward one hour to 3:00 AM. The time between 2:00 AM and 3:00 AM on that day doesn't exist. When users try to enter a value in that time range, apps might do one of the following:

- Change to the previous or next valid time.
- Revert to the last known value.
- Show an error message.

- Don't show times between 2:00 AM and 3:00 AM in the time picker, so that users can't select them in the first place.

Similarly, different clients have different ways to handle repeated time ranges. For example, in the Pacific time zone, daylight saving ended on November 5th, 2023 at 2:00 AM, moving the time backward one hour to 1:00 AM. The time between 1:00 AM and 2:00 AM on that day is repeated twice. A time like 1:30 AM could refer to either time zones. If you need to unambiguously show or enter times in that range, it's best to temporarily switch to a time zone that doesn't use daylight saving.

Get raw values with Web API

Dataverse stores `2023-10-15T07:30:00Z` (or `2023-10-15` for **Date only** behavior). Users in all time zones get these values with a Web API request for the value:

Expand table

Behavior	Format	Raw value
User local	Date and time	2023-10-15T07:30:00Z
User local	Date only	2023-10-15T07:30:00Z
Time zone Independent	Date and time	2023-10-15T07:30:00Z
Time zone Independent	Date only	2023-10-15T07:30:00Z
Date only	-	2023-10-15

Get values with Client API

Users in the time zone UTC-8 enter `October 14th, 2023, 11:30 pm` in a model-driven app. Client API functions like `formContext.getAttribute(<column name>).getValue()` return the value with time zone adjustments applied:

Expand table

Behavior	Format	JavaScript `dateValue.toUTCString()`
User local	Date and time	2023-10-15 07:30 (UTC)
User local	Date only	2023-10-15 07:30 (UTC)

For **Time zone independent** behavior, the JavaScript Date value is in the browser's time zone:

Expand table

Behavior	Format	JavaScript `dateValue.toString()`
Time zone Independent	Date and time	2023-10-14 23:30 (browser time zone)
Time zone Independent	Date only	2023-10-14 23:30 (browser time zone)

JavaScript date values always have a time component. That's why **Date only** behavior has a time component of 12:00 AM:

Expand table

Behavior	Format	JavaScript `dateValue.toString()`
Date only	-	2023-10-15 00:00 (browser time zone)

> **Note**
>
> JavaScript date values are affected by the browser's time zone, which comes from the device operating system settings.
>
> For **User local** behavior, interpret the Client API result as a UTC value. Use `Date.getUTCDate()` or `Date.getUTCHours()` to work with it. To get what the user sees, apply getTimeZoneOffsetMinutes. Don't use `Date.getDate()` or `Date.getHours()` because these functions show the value in the browser's time zone.
>
> For **Time zone independent** and **Date only** behavior, interpret the Client API result as a value in the browser's time zone. Use `Date.getDate()` or `Date.getHours()` to work with it. Don't use `Date.getUTCDate()` or `Date.getUTCHours()` because you don't have to adjust for any time zones.

Change user local behavior

Unless the publisher of a managed solution prevents changing the local behavior, you can change the behavior of existing custom date columns from **User local** to **Date only** or **Time zone independent**. This is a one-time change.

Changing the column behavior affects the column values that are added or modified after the column behavior was changed. The existing column values remain in the database in the UTC time zone format. To change the behavior of the existing column values from UTC to **Date only**,

you might need a help of a developer to convert behavior of existing date and time values in the database.

> ⚠ **Warning**
>
> Before changing the behavior of an existing date and time column, review all the dependencies of the column, such as business rules, workflows, calculated columns, or rollup columns, to ensure that there are no issues as a result of changing the behavior. After changing the behavior of a date and time column, open each business rule, workflow, calculated column, and rollup column dependent on the column that you changed, review the information, and save it, to ensure that the latest date and time column's behavior and value are used.

Change behavior during a solution import

When you import a solution that contains a Date column with **User local**, you can change the behavior to **Date only** or **Time zone independent**.

> ⓘ **Note**
>
> You can only change the behavior of an existing managed **Date only** or **Date and time** column if you are the publisher. In order to make a change to these fields, an upgrade must be made to the solution that added the **Date only** or **Date and time** column. More information: Upgrade or update a solution

Prevent changing behavior

If you're distributing a custom date column in a managed solution, prevent people using your solution from changing the behavior by setting the **CanChangeDateTimeBehavior** managed property to **False**. More information: Set managed properties for columns

Date and time query operators not supported for Date only behavior

The following date and time related query operators are invalid for the **Date only** behavior. An invalid operator exception error is thrown when one of these operators is used in the query.

- Older Than X Minutes

- Older Than X Hours
- Last X Hours
- Next X Hours

See also

Troubleshoot date and time issues in model-driven apps

Create and edit columns

Define calculated columns to automate manual calculations

Column managed properties

Managed properties

Blog: Working with time zones in the Dataverse

Configure the behavior and format of the date and time column using code

---

Last updated on 04/09/2026

How to create and edit columns

08/14/2025

In Microsoft Dataverse columns, define the individual data items that can be used to store data in a table. Columns are sometimes called *attributes* by developers.

Watch this video for a quick overview about columns:
https://learn-video.azurefd.net/vod/player?id=3379c026-1830-4265-86fc-c7d16a0b6b49&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fcreate-edit-fields

Before you create a custom column, evaluate whether using an existing column would meet your requirements. More information: Create new metadata or use existing metadata?

> **Note**
>
> You can also create columns in your environment using these methods:
>
> - For model-driven apps, select **New table column** in the form editor.
> - Import a solution that contains the definition of the columns.
> - Use Power Query to create new tables and fill them with data. Learn how to add data to a table in the Dataverse by using Power Query.
> - A developer can use Metadata services to write a program to retrieve information about, create, and update columns.

> **Note**
>
> You can create a Lookup column in Power Apps by creating a one-to-many relationship on the table.

For more information about how to create a table column, go to Create and edit columns for Dataverse using Power Apps portal.

Community tools

Attribute Manager is a tool that XrmToolbox community developed for Dataverse. Go to the Developer tools and resources article for more community developed tools.

> **Note**

The community tools aren't a product of Microsoft and Microsoft doesn't offer support for the community tools. If you have questions pertaining to the tool, contact the publisher. More Information: **XrmToolBox**

Related articles

Types of columns and column data types

Developer Documentation: Column definitions

Manage custom columns in a table

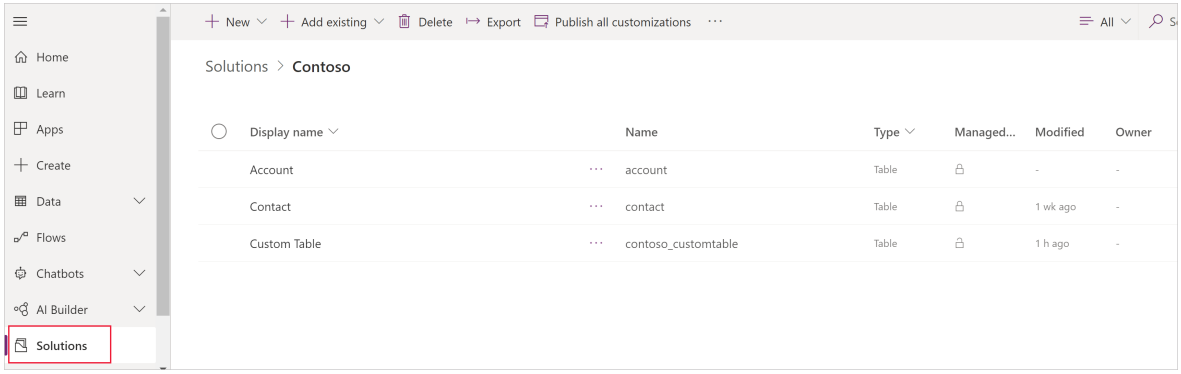
Article • 02/24/2023

You can create and update one or more custom columns in any table. When you create a custom column, you specify a set of properties, such as the column's name, its display name, and the type of data that it will contain. For more information, see table attribute metadata.

> **Note**
>
> Every table has system columns, such as columns that indicate when a row was last updated, and who updated it. In addition, standard tables have standard (default) columns. You can't modify or delete system columns or standard columns. If you create a custom column, it should provide functionality on top of these built-in columns.

Create a column

1. On powerapps.com, select **Tables** in the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.



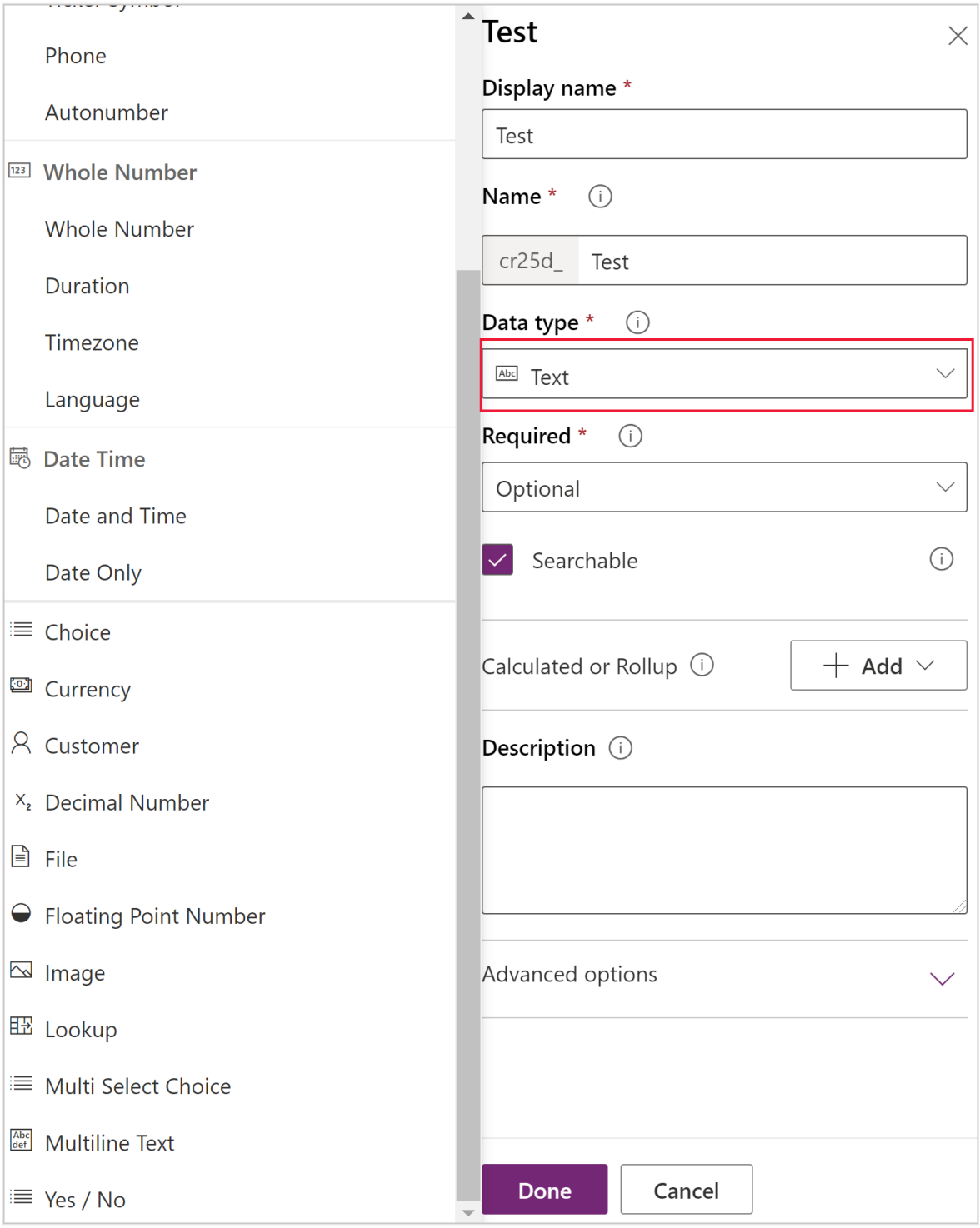
2. Select an existing table, or Create a new table

3. Add a new column to your table by selecting **Add column**.

4. In the New Column panel, enter the **Display name** for your column, **Name** will be automatically populated and is used as the unique name for your column. The **Displayname** is used when presenting this column to your users, the **Name** is used when building your app, in expressions and formulas.

   > **Note**

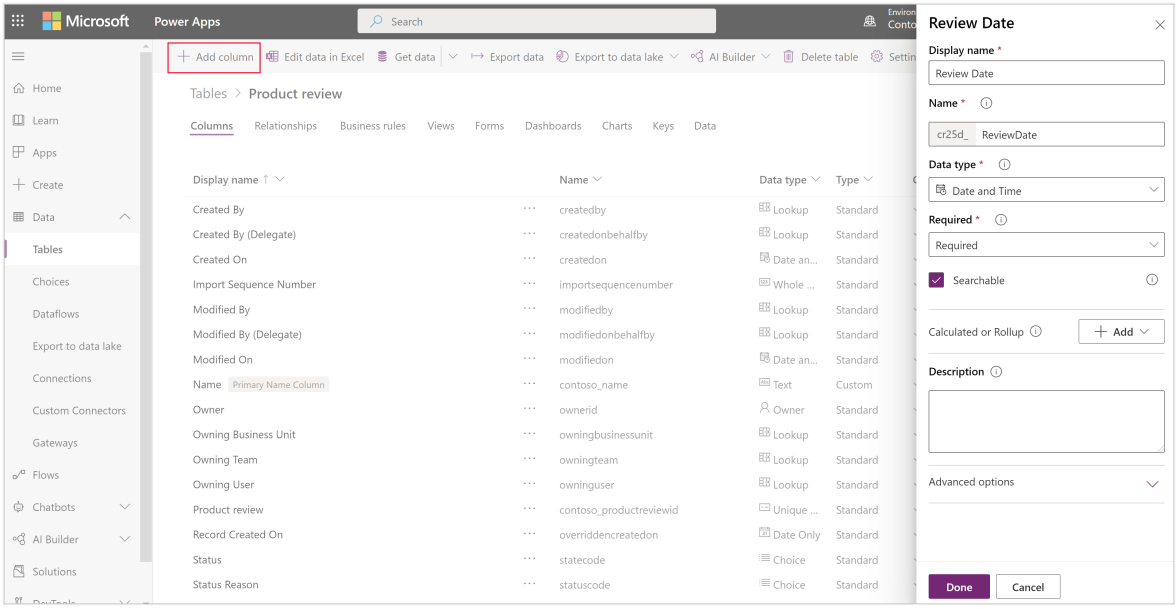
The **Display name** columns can be updated at anytime to display differently in your apps, the **Name** column can't be changed after your table has been saved as this could result in breaking an existing app.



5. Select the **Data type** of your column, this controls the way the information is stored as well as how it is presented in apps. For example, text is stored different to a decimal number or a URL. For more detailed information of the data types available, see table attribute metadata.

   If you're prompted, specify additional information for the data type that you specified. Depending on the data type, different columns will be presented. If

you're creating a column of type Choice or Multi Select Option Set, you can select **New Option Set** and create a new Choice while creating your column. For more information, see Create Option set



6. Under **Required**, select the check box if you want to recommended this column as required in your apps. This doesn't provide hard enforcement through all connections to Microsoft Dataverse. If you need to ensure the column is populated, create a Business Rule

7. Under **Searchable**, select the check box if you need this column to be available in Views, Charts, Dashboards and Advanced Find. In most cases this checkbox should be checked.

8. Select **Done** to close the Column panel and return to the table. You can repeat steps 3-9 for each additional column.

   > **Important**
   >
   > Your column is not yet saved and created, until you save the changes to the table.

9. Select **Save Table** to finalize your changes and save them to the Dataverse.

   You're notified when the operation is completed successfully. If the operation is unsuccessful, an error message indicates the issues that occurred and how you can fix them.

Create a Calculated or Roll up column

Calculated columns let you automate manual calculations used in your business processes. For example, a salesperson may want to know the weighted revenue for an opportunity which is based on the estimated revenue from an opportunity multiplied by the probability. Or, they want to automatically apply a discount, if an order is greater than $500. A calculated column can contain values resulting from simple math operations, or conditional operations, such as greater than or if-else, and many others. Calculated columns can be created using the following data types:

- Single line of text
- Option Set
- Two Options
- Whole Number
- Decimal Number
- Currency
- Date and Time

For more details on the types of expressions supported and examples, see Define calculated columns

Update or delete a column

1. On powerapps.com, select **Tables** in the left navigation pane. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. Open a table.

3. In the list of columns for the table that you selected, selectp a column, and then follow one of these steps:

   - Change one or more properties of the column.
   - Delete the column by clicking or tapping the ellipsis (...) near the right edge of the column, and then clicking or tapping **Delete**.

4. Select **Save table** to submit your changes.

   > **Important**
   >
   > Your changes will be lost if you don't save them before you open another page in the browser or exit the browser.

   You're notified when the operation is completed successfully. If the operation is unsuccessful, an error message indicates the issues that occurred and how you can

fix them.

Best practices and restrictions

As you create and modify columns, keep these points in mind:

- You can't modify or delete system columns or their values.
- In a standard table, you can't modify or delete a standard (default) column, add a column that requires data, or make any other change that might break an app that relies on that table.
- In a custom table, you should make sure that the changes that you make won't break any app that relies on that table.
- You must give each custom column a name that's unique within the table, and you can't rename a column after you create it.

Next steps

- Define relationships between tables
- Create a business rule
- Create an app using tables
- Create an app from scratch using a Dataverse database

Privacy notice

With the Microsoft Power Apps common data model we collect and store custom table and column names in our diagnostic systems. We use this knowledge to improve the common data model for our customers. The table and column names that Creators create help us understand scenarios that are common across the Microsoft Power Apps community and ascertain gaps in the service's standard table coverage, such as schemas related to organizations. The data in the database tables associated with these tables is not accessed or used by Microsoft or replicated outside of the region in which the database is provisioned. Note, however, the custom table and column names may be replicated across regions and are deleted in accordance with our data retention policies. Microsoft is committed to your privacy as described further in our Trust Center.

Create and edit columns in Dataverse using Power Apps

Creating and editing table columns in Microsoft Dataverse using Power Apps is a straightforward process that allows you to customize your data tables to meet your specific needs. This guide walks you through the steps to view, create, and edit columns, as well as provide detailed information on the various data types and properties you can configure. Whether you're adding new columns or modifying existing ones, Power Apps offers a user-friendly interface to manage your Dataverse tables efficiently.

View columns

1. From Power Apps, select **Tables**, and then open the table that has the columns you want to view. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select the **Columns** area, where all columns for the table are displayed.

Create a column

While viewing columns, on the command bar, select **New column** which displays the **Column properties** panel.

Several column properties are available:

Expand table

Property	Description
Display Name	The text to be displayed for the column in the user interface.
Name	The unique name across your environment. A name is generated for you based on the display name that you've entered, but you can edit it before saving. Once a column is created, the name can't be changed as it might be referenced in your applications or code. The name has the customization prefix for your **Dataverse Default Publisher** prepended to it.
Data type	Controls how values are stored as well as how they're formatted in some applications. Once a column is saved, you can't change the data type except for converting text columns to autonumber columns.
Format	Depending on the Data type selected, you can select the format for the column data type. For example, whole number can have the format duration, language code, time zone, or none.

Property	Description
Behavior	By default Simple is selected. Some data types support extended behavior such as calculated or rollup.
Required	A record can't be saved without data in this column. Required columns on a hidden tab or section don't block saving the record unless the same column is also in a visible tab or section on the same form. More information: Saving rows programmatically for required columns
Searchable	This column appears in Advanced Find and is available when customizing views.
Allow form fill assistance (preview)	This is a preview feature. When enabled, users receive suggestions generated by AI for columns that support form fill assistance in main and quick create forms. You can disable for columns that might contain sensitive information, such as email address. Requires at least one **AI form fill assistance** environment setting enabled. For more information about AI form fill assistance, go to: Use Copilot's form fill assistance feature in model-driven apps Notice that configuring this setting creates a dependency on the AI Skill Config table (used with AI Builder) in the default solution. When a maker disables the **Allow form fill assistance** property on a table column, a record is created in the AI Skill Config table to store that configuration. This record creates a dependency on the AI Skill Config table for the column. More information: Delete a column with an AI Skill Config table dependency in Power Apps
Advanced Options	Enter the schema name, specify a maximum length, IME mode, column security, auditing, can appear in dashboards, and whether the column data is sortable.

You can set other options depending on your choice of **Data type**.

Column data types

There are many different types of columns, but you can only create some of them. For more information about all types of columns, see Types of columns and column data types.

When you create a column, **Data type** provides the following choices:

Text

Standard text columns can store up to 4,000 characters. The default Max Length option is set to a lower value you can adjust.

Expand table

Data type	Description
Text	A text value intended to be displayed in a single-line textbox.

Data type	Description
Text Area	A text value intended to be displayed in a multi-line textbox. If you require more than 4,000 characters, use a Multiline Text data type.
Email	A text value validated as an e-mail address and rendered as a mailto link in the column.
URL	A text value validated as a URL and rendered as a link to open the URL.
Ticker Symbol	A text value for a ticker symbol that displays a link that opens to show a quote for the stock ticker symbol.
Phone	A text value validated as a phone number rendered as link to initiate a phone call by using Skype.
Autonumber	A customizable combination of numbers and letters that is automatically generated by the server whenever the row is created. More information: Autonumber columns

Max Length

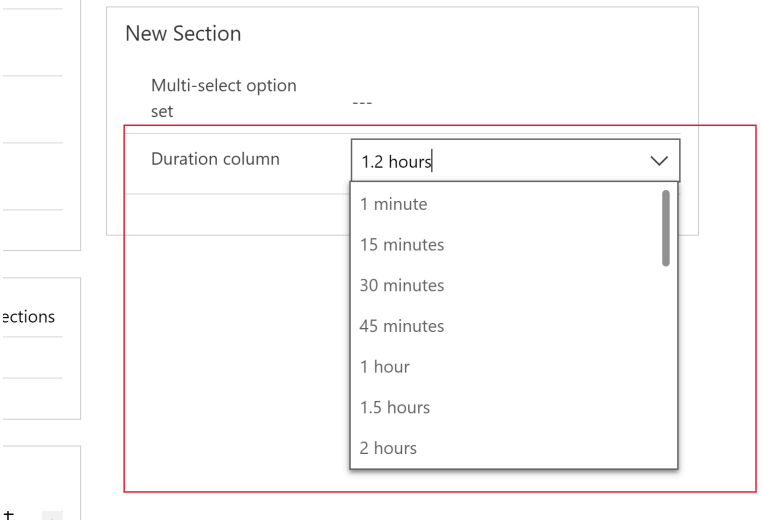
Columns that store text have an absolute maximum depending on the type. The **Max Length** choices a value lower than the maximum specific to your environment. You can increase this max length but you shouldn't lower it if you have data in the system that exceeds the lower value.

Whole Number

These columns store data as a number but include different presentation and validation options.

Expand table

Format	Description
Duration	A number value presented as a drop-down list that contains time intervals. A user can select a predefined value from the list or type an integer value using the format: "x minutes," "x hours" or "x days." Hours and days can be entered using decimals, for example, "1.2 hours" or "1.5 days." Minute values entered must be expressible in whole minutes. Entering a

Format	Description
	decimal to represent sub-minute values are rounded to the nearest minute. 
Timezone	A number value presented as a drop-down list that contains a list of time zones.
Language code	A number value presented as a drop-down list that contains a list of languages that have been enabled for the environment. If no other languages have been enabled, the base language will be the only option. The value saved is the Locale Identifier (LCID) value for the language.
None	No special formatting will be applied for the whole number column. This is the default format.

Date Time

Use these columns to store time values. You can store values as early as 1/1/1753 12:00 AM.

Expand table

Data type	Description
Date and Time	A date and time value.
Date Only	A date and time value that only displays a date. The time value is stored as 12:00 AM (00:00:00) in the system.

You can also set specific **Behavior** for Date Time columns in the **Advanced options**.

- **User local** : Displays values converted to in the current user's local time zone. This is the default for new columns.
- **Date only**: This behavior is available for the **Date Only** type. Displays values without time zone conversion. Use this for data like birthdays and anniversaries.
- **Time zone independent**: Displays values without time zone conversion.

More information: Behavior and format of the Date and Time column

Other Data types

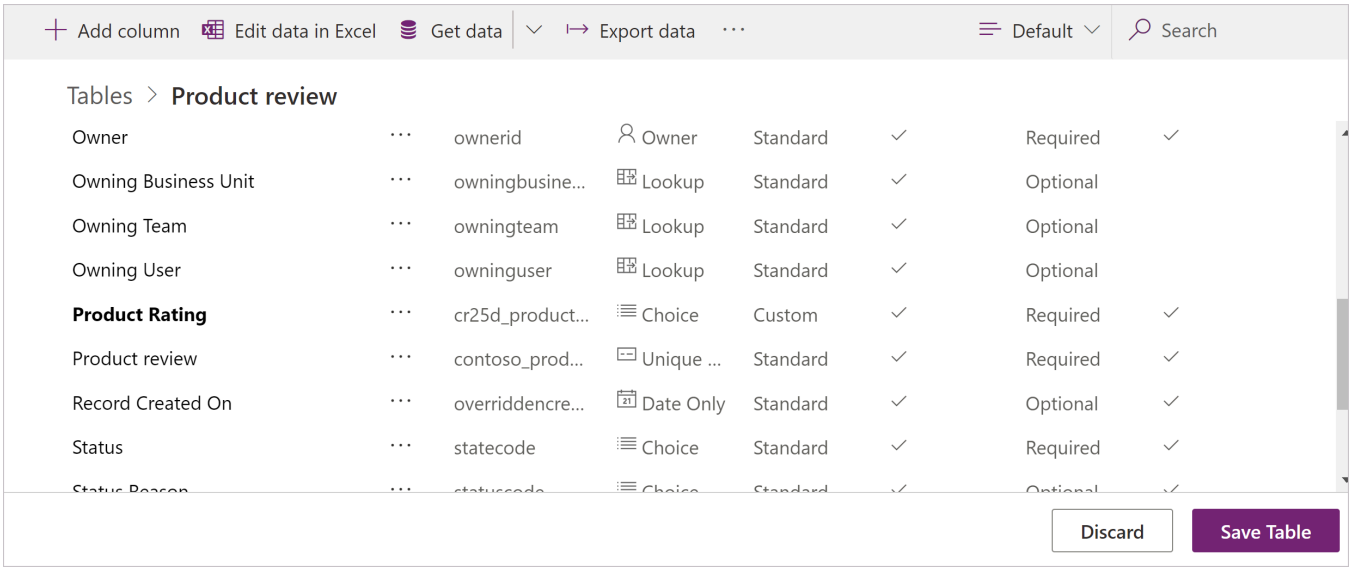
Expand table

Data type	Description
Currency	A money value for any currencies configured for the environment. You can set a level of precision or choose to base the precision on a specific currency or a single standard precision used by the organization. More information: Using currency columns
Decimal Number	A decimal value with up to 10 points of precision. More information: Using the right type of number
File	For storing binary data.
Floating Point Number	A floating point number with up to 5 points of precision. More information: Using the right type of number
Image	Displays a single image per row in the application. Each table can have one image column. The **Name** you enter when creating an image column is ignored. Image columns are always named 'tableImage'.
Lookup	Creates a reference to a single row for a single target row type.
Multi Select Choice	Displays a list of options where more than one can be selected.
Multiline Text	A text value intended to be displayed in a multi-line textbox. Limited to a maximum of 1,048,576 characters. You can also set a lower Max Length.
Choice	Displays a list of options where only one can be selected.
Yes/No	Displays Yes/No where only one can be selected. You choose which labels are displayed for each option. The default values are **Yes** and **No**.
Formula	Built on Power Fx, use a formula column to perform operations that return values during fetch operations. Formula columns use the Power Fx syntax that's similar to Office Excel. More information: Work with formula columns (preview)
Prompt	Prompt columns allow AI prompts to be defined and the generative AI results are stored in the table column. More information: Prompt columns

Save new column

Once you set the **Display Name**, **Name** and **Data type** properties you can select **Done** to close the **Column properties** panel.

You can continue to edit the table and add more columns or return and continue editing this column. The columns aren't created until you select **Save Table** to save all the changes to the table.



You can also select **Discard** to discard the changes you made.

Edit a column

While viewing columns, select the column, you want to edit. You can modify the **Display Name** but you can't change the **Name** and **Data type** if you saved changes to the table to add the column.

General properties

Every column has the following properties you can change:

Expand table

Property	Description
Required	A record can't be saved without data in this column. Note that required columns on a hidden tab or section don't block saving the record unless the same column is also in a visible tab or section on the same form. More information: Saving rows programmatically for required columns
Searchable	De-select this for columns for the table that you don't use. When a column is searchable, it appears in **Advanced Find** and is available when customizing views. Deselecting this reduces the number of options shown to people using advanced find.

Property	Description
Description	Found within **Advanced Options**. Enter instructions to the user about what the column is for. These descriptions appear as tooltips for the user in model-driven apps when they hover their mouse over the label of the column.

> ⓘ Note
>
> **Making columns required**: Be careful when you make columns required. People resist using the application if they can't save rows because they lack the correct information to enter into a required column. People might enter incorrect data simply to save the row and get on with their work.
>
> **Set requirement dynamically**: In model-driven apps you can use business rules or form scripts to change the requirement level as the data in the row changes as people work on it. More information: Create business rules and recommendations to apply logic in a form
>
> **Advanced Find availability**: Advanced Find is currently only available for model-driven apps using the Web Client. Advanced find isn't currently available in Unified Interface clients.

Searching and sorting columns

For information about columns that can't be enabled for searching or sorting, go to Searching and sorting columns.

Saving rows programmatically for required columns

When a row is saved programmatically using web services, only the `SystemRequired` columns are enforced. Failure to set a value for `SystemRequired` columns return an error. You can't set the `SystemRequired` level of requirement.

Setting a column to Business Required means that the default behavior of a model-driven or canvas app enforce this requirement in the app. The request isn't sent to the service if the column has no value. The app user is shown an error and prompted to add data to the required column before they can save the row. There are options within the app to override this behavior and allow operation to proceed if needed.

Calculated or Rollup

You can set a custom column to be a **Calculated** or a **Rollup** column. Columns that aren't calculated or rollup columns are sometimes referred to as *simple* columns.

Calculated

With a calculated column, you can enter a formula to assign a value to the column. These data types can be set to calculated columns: **Currency**, **Date and Time**, **Date Only**, **Decimal Number**, **Duration**, **Email**, **Language**, **Multi Select Option Set**, **Choice**, **Text**, **Text Area**, **Ticker Symbol**, **Timezone**, **Two Options**, **URL**, and **Whole Number**.

More information: Define calculated columns to automate manual calculations

Rollup

With a rollup column, you can set aggregation functions that run periodically to set a number value for the column. These data types can be set to calculated columns: **Currency**, **Date and Time**, **Date Only**, **Decimal Number**, **Duration**, **Language**, **Timezone**, and **Whole Number**.

More information: Define rollup columns that aggregate values

Number column options

Each type of number column has absolute minimum and maximum values. You can set appropriate **Minimum value** and **Maximum value** within these absolute values. Do this to have Dataverse validate the values for the data you want to store in the column.

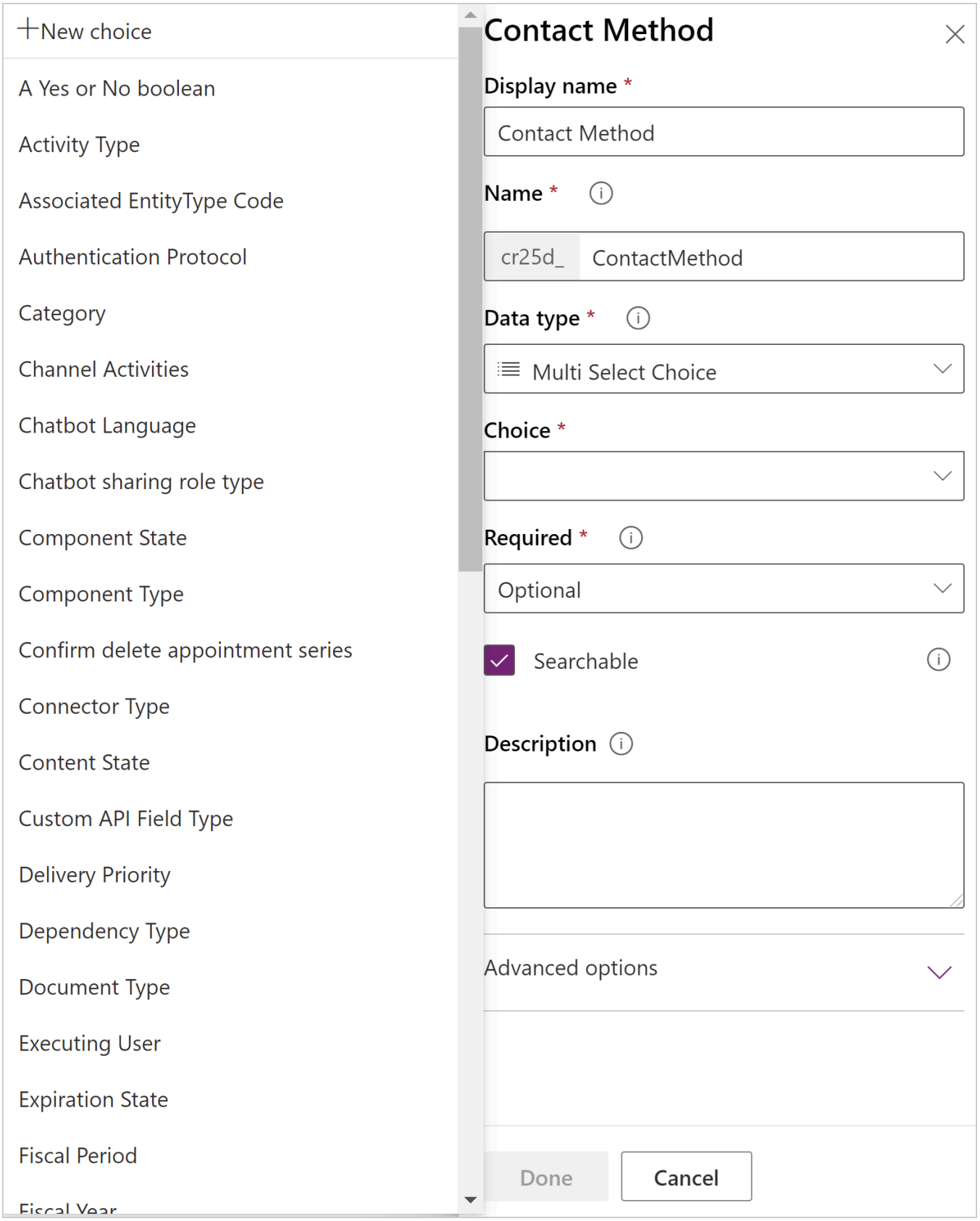
For **Floating Point Number** and **Decimal Number** data types, you can specify a number of **Decimal places**.

Choice column options

Columns that provide a set of options can include their own set of *local* options or refer to a common set of *global* options that can be used by multiple columns.

Using a global choice is valuable when you find yourself creating the same set of options for multiple columns. With a global choice, you only need to maintain the set of options in one place.

When you choose **Multi Select Choice** or **Choice** data type the designer lists a set of available global choices for you to choose from and provides the option to create a **New choice**.

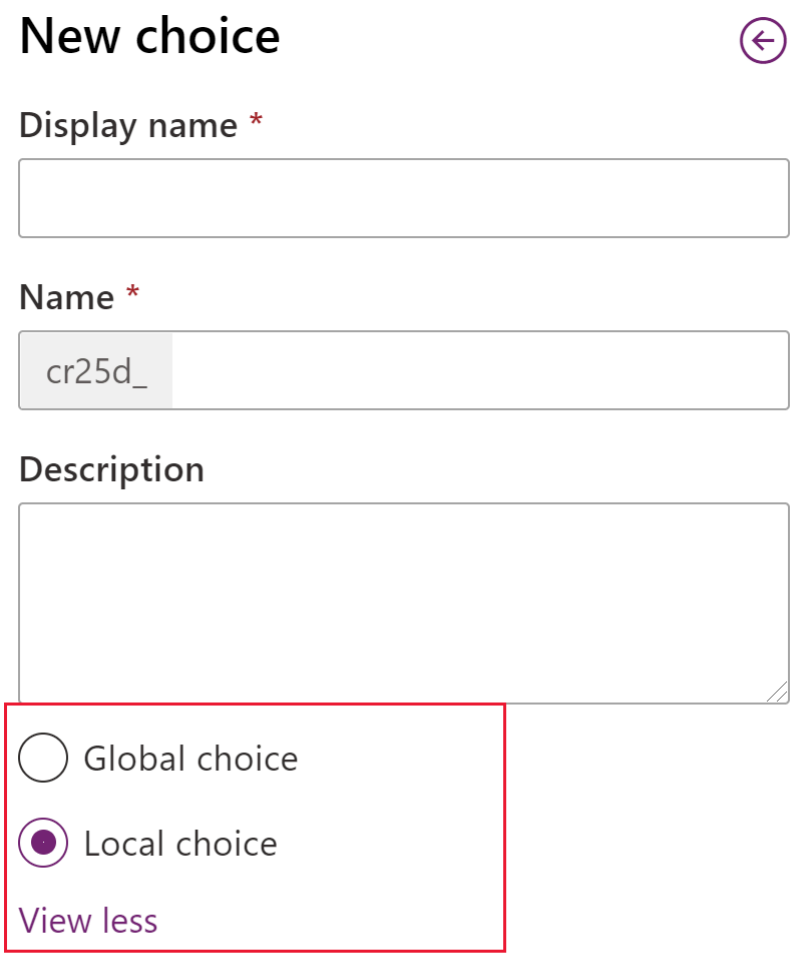


If you choose **New choice**, the default behavior is to create a new global choice.

> **Note**
>
> While you're editing options for a new global choice, the **Display name** and **Name** values are for the global choice rather than for the column. The default values match the column

values, but you can edit them while you edit the global choice to be different from the column you're currently creating.

If you want to create a local choice, you must select **View more** and choose **Local choice**.



Note

If you define every choice as a global choice, your list of global choices grows and could be difficult to manage. If you know that the set of options will only be used in one place, use a local choice.

Warning

If you remove an option which has been used by an entity record the data for that record will become invalid after you save your changes to the global option set.

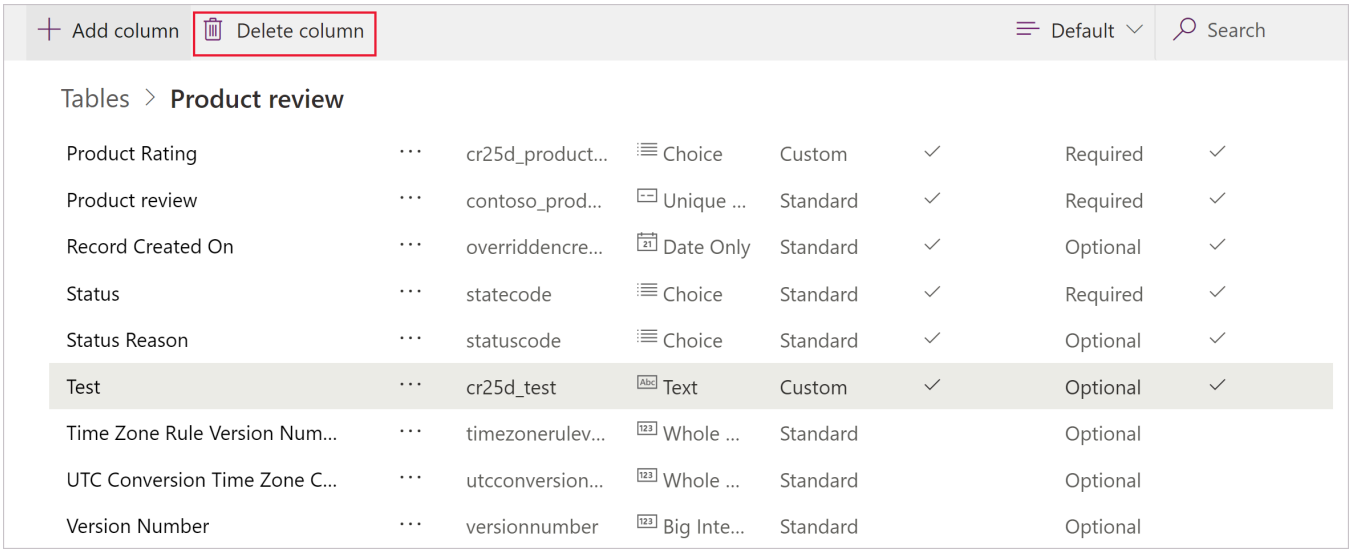
Before you remove an option that has been used, you should change the data in any entity record that uses that option to a valid value.

Delete a column

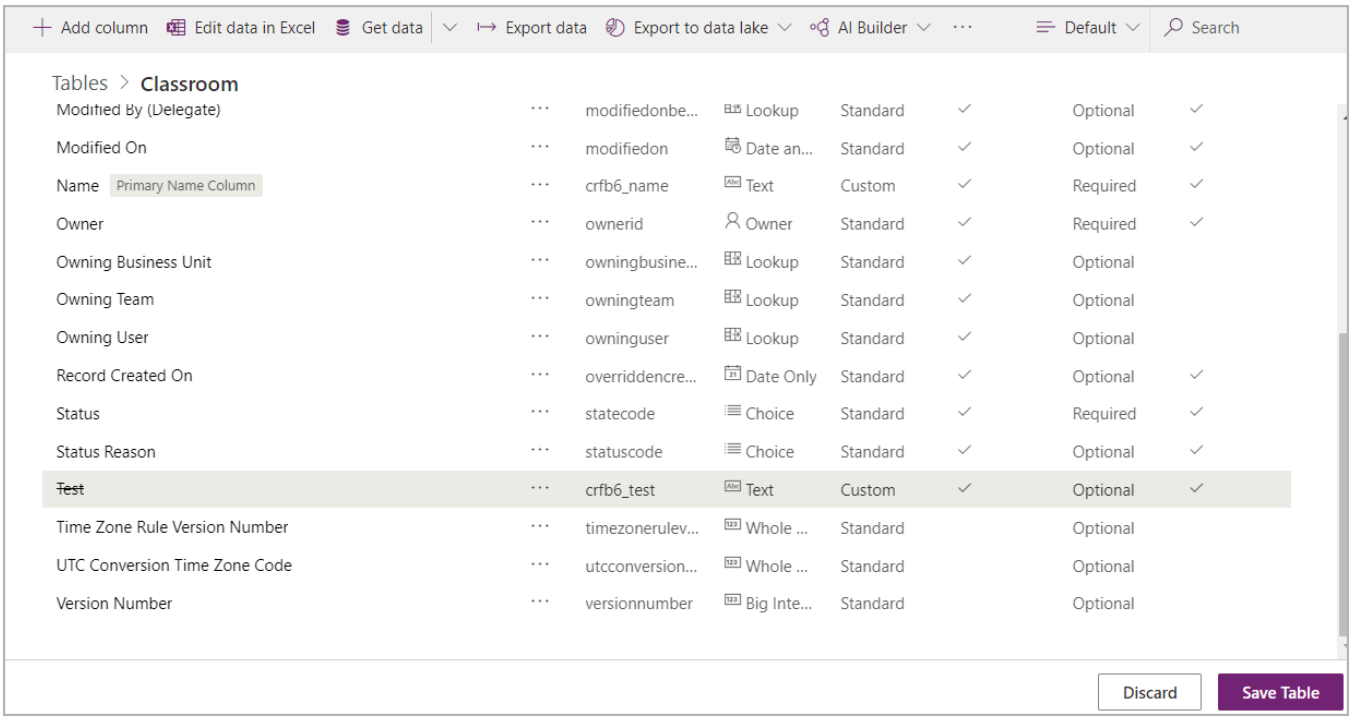
With the system administrator security role, you can delete any custom columns that aren't part of a managed solution. When you delete a column, any data stored in the column is lost. The only way to recover data from a column that was deleted is to restore the database from a point before the column was deleted.

> **Note**
>
> Before you can delete a custom column, you must remove any dependencies that might exist in other solution components.

While viewing columns, if you select a custom column that can be deleted in the list, the **Delete column** command appears and is enabled.



Use the **Delete column** command to delete the column. After deleting the column, you must save the changes to the table.



> **Note**
>
> If you get an error related to dependencies, you must use solution explorer to detect dependencies. More information: Check column dependencies

IME Mode

IME (input method editor) mode specifies how a physical keyboard can be used to enter characters for text columns. IMEs are tools provided by the operating system for composing text. They're commonly used to enter Chinese, Japanese, and Korean words. IME mode doesn't restrict the characters that users can enter. For example, when IME mode is disabled, users can still enter Japanese characters by pasting in a text input.

> **Important**
>
> IME Mode is used for backward compatibility with the legacy web client and the IME Mode settings described here don't apply to Unified Interface apps. Additionally, IME mode is supported only in Internet Explorer and partially supported in Firefox.

- **Active**: This value activates the IME initially. You can deactivate it later if desired. This is the default IME setting.
- **Auto**: When IME mode is auto, Power Apps won't interfere with the IME.
- **Disabled**: Disable IME mode to bypass the IME. This can be useful for entering alphanumeric characters in certain languages.

- **Inactive**: Power Apps will deactivate the IME initially. You can activate it later if desired.

See also

Create and edit columns for Dataverse
Create and edit columns for Dataverse using Power Apps solution explorer
Types of columns and column data types
Define calculated columns to automate manual calculations
Define rollup columns that aggregate values
Behavior and format of the Date and Time column

Last updated on 01/09/2026

Create and edit columns for Microsoft Dataverse using Power Apps solution explorer

Article • 04/01/2025

Solution explorer provides one way to create and edit columns for Dataverse.

The Power Apps portal enables configuring the most common options, but certain options can only be set using solution explorer.
More information:

- Create and edit columns for Dataverse
- Create and edit columns for Dataverse using Power Apps portal

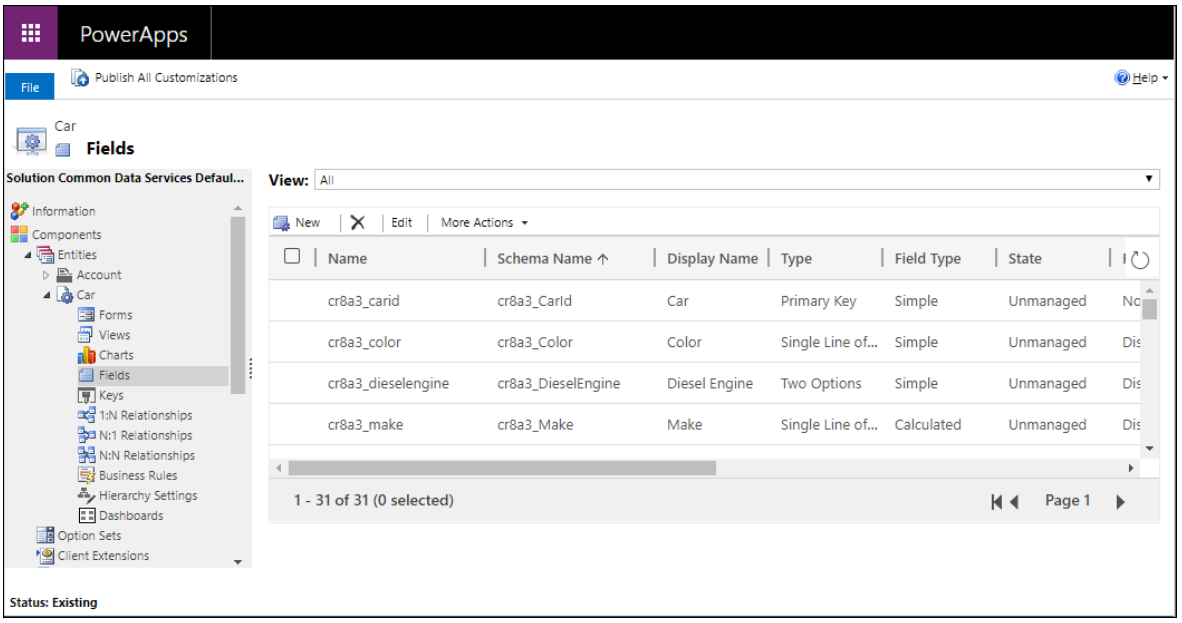
Part of the name of any custom column you create is the customization prefix. This is set based on the solution publisher for the solution you're working in. Make sure that you're working in an unmanaged solution where the customization prefix is the one you want for this table. More information: Change the solution publisher prefix

View columns

1. From Power Apps, select **Solutions** on the left navigation pane, and then on the toolbar select **...** > **Switch to classic**.

2. Open the unmanaged solution where you want to view a column. The classic solution explorer opens in a new browser window.

3. In the classic solution explorer, under **Components** expand **Entities**, expand the table where you want to view columns, and then select **Fields**.



In the **View** drop down list you can select one of the following views:

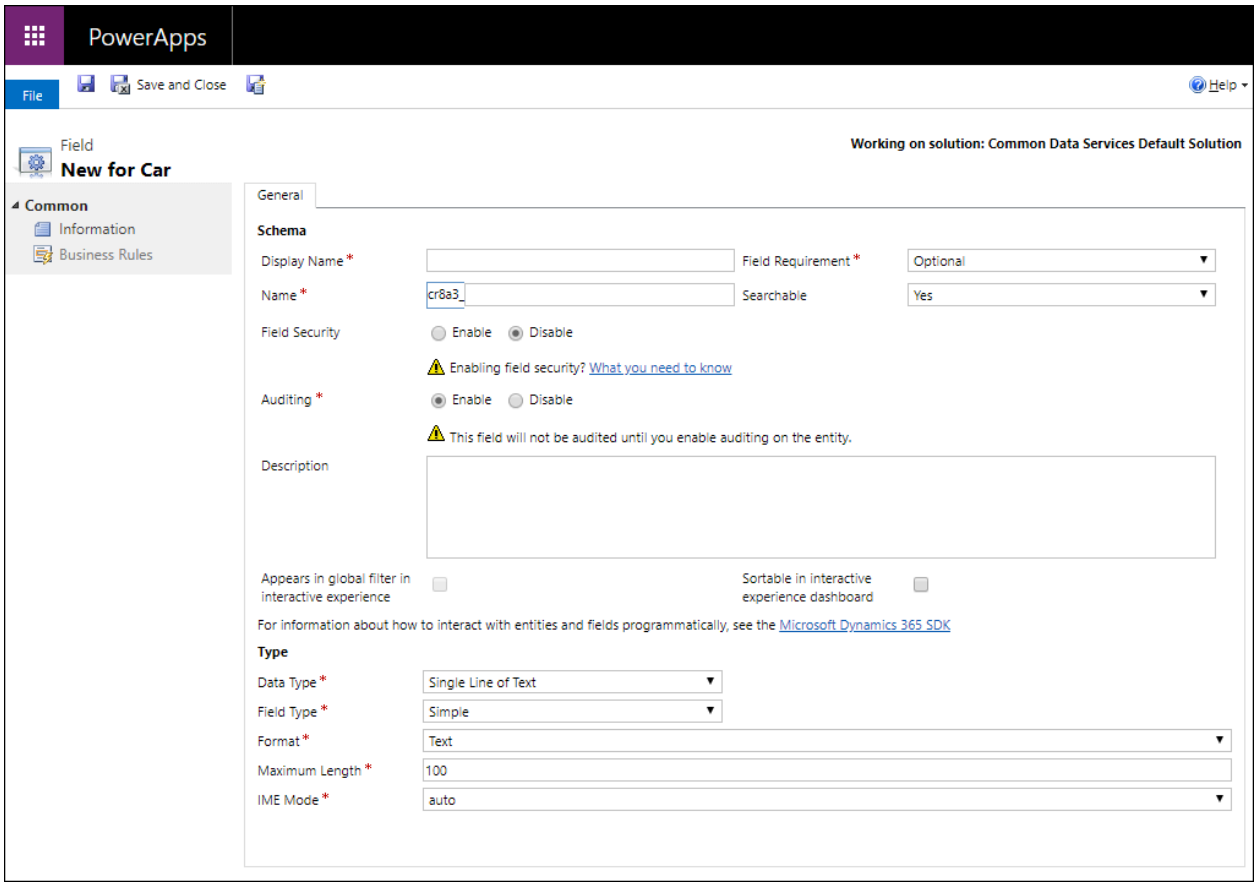
Expand table

View	Description
All	Shows all the columns for the table
Custom	Shows only custom columns for the table
Customizable	Shows only the columns that can be edited

Create a column

While viewing columns, in the command bar, select **New** which opens the new column form. Some standard tables or custom tables that are included in a managed solution might not allow you to add new columns.

> **Note**
>
> For model-driven apps, you can also create a new column from the form editor. In the form editor, below the **Column Explorer** select **New Column** to create a new column. More information: Add, configure, move, or delete columns on a form



You must enter data and confirm default values set for the following properties before you save.

Expand table

Property	Description
Display Name	The text to be displayed for the column in the user interface. You can change this after you save, but the value you enter will generate a value for the **Name** column.
Column Requirement	Whether data is required in the column to save the row. More information: Column Requirement options
Name	The unique name across your environment. A name is generated for you based on the display name that you've entered, but you can edit it before saving. Once a column is created the name can't be changed as it might be referenced in your applications or code. The name has the customization prefix for the current solution's publisher prepended to it.
Searchable	Set this to **No** for columns for the table that you don't use. When a column is searchable, it appears in **Advanced Find** in model-driven apps and is available when customizing views. Deselecting this reduces the number of options shown to people using advanced find.
Column Security	Whether the data in the column is secured at a higher level than the table. More information: Column level security to control access

Property	Description
Auditing	Whether data for this column is audited when the table is enabled for auditing. More information: Manage Dataverse auditing
Description	Enter instructions to the user about what the column is for. These descriptions appear as tooltips for the user in model-driven apps when they hover their mouse over the label of the column.
Appears in global filter in interactive experience	More information: Configure interactive experience dashboards
Sortable in interactive experience dashboard	More information: Configure interactive experience dashboards
Data type	Controls how values are stored as well as how they're formatted in some applications. Once a column is saved, you can't change the data type as it might impact the data in your table. More information: Column Data types
Column type	Whether the column is **Simple**, **Calculated**, or **Rollup**. More information: Column Type
Format	How the column is formatted. The available formatting options depend on the **Data type**.

You can set additional options depending on your choice of **Data type**. More information: Column Data types

Column Requirement options

There are three column requirement options:

- **Optional**: The row can be saved even if there's no data in this column.
- **Business Recommended**: The row can be saved even if there's no data in this column. However, a blue symbol appears next to the column to indicate it's important.
- **Business Required**: The row can't be saved if there's no data in this column.

> ⓘ Note
>
> - When a row is saved programmatically using web services, only the SystemRequired columns are enforced. Failure to set a value for SystemRequired columns return an error. You can't set the SystemRequired

level of requirement.

Setting a column to Business Required means that the default behavior of a model-driven or canvas app will enforce this requirement in the app. The request won't be sent to the service if the column has no value. The app user is shown an error and prompted to add data to the required column before they can save the row. There are options within the app to override this behavior and allow operation to proceed if needed.

- Be careful when you make columns business required. People will resist using the application if they can't save rows because they lack the correct information to enter into a required column. People might enter incorrect data simply to save the row and get on with their work. You can use business rules or form scripts to change the requirement level as the data in the row changes as people work on it. More information Create business rules and recommendations to apply logic in a form

Column Data types

There are many different types of columns, but you can only create some of them. For more information about all types of columns, see Types of columns and column data types.

When you create a column, **Data type** provides the following choices:

Expand table

Option	Description
Single Line of Text	This column can contain up to 4,000 text characters. You can set a maximum length to be less than this. This column has several format options that change the presentation of the text. More information: Single line of text options
Choice	Displays a list of options where one can be selected. More information: Choice column options
MultiSelect Choice	Displays a list of options where more than one can be selected. More information: Choice column options
Yes/No	Displays a list of options where one of two can be selected. Yes/No columns don't provide format options at the column level. But when you add one to the form you can choose to display them as radio buttons, a check box, or a select list.

Option	Description
Image	Displays a single image per row in the application. Each table can have one image column. Image columns are always named `tableImage`.
Whole Number	Integers with a value between -2,147,483,648 and 2,147,483,647 can be in this column. This column has options that change depending on how the column is presented. More information: Whole number options
Floating Point Number	Up to 5 decimal points of precision can be used for values between -100,000,000,000 and -100,000,000,000 can be in this column. You can specify the level of precision and the maximum and minimum values. More information: Using the right type of number
Decimal Number	Up to 10 decimal points of precision can be used for values between -100,000,000,000 and -100,000,000,000 can be in this column. You can specify the level of precision and the maximum and minimum values. More information: Using the right type of number
Currency	Monetary values between -922,337,203,685,477 and 922,337,203,685,477 can be in this column. You can set a level of precision or choose to base the precision on a specific currency or a single standard precision used by the organization. More information: Using currency columns
Multiple Lines of Text	This column can contain up to 1,048,576 text characters. You can set the maximum length to be less than this. When you add this column to a model-driven app form, you can specify the dimensions of the column.
Date and Time	Use these columns to store time values. You can store values as early as 1/1/1753 12:00 AM. More information: Date and Time options
Lookup	A column that allows setting a reference to a single row of a specific type of table. Some system lookup columns behave differently. More information: Different types of lookups
Customer	A lookup column that you can use to specify a customer, which can be an account or contact. More information: Different types of lookups

Single line of text options

The single line of text data type has the following format options:

Expand table

Format	Description
Text	A text value intended to be displayed in a single-line textbox.

Format	Description
Text Area	A text value intended to be displayed in a multi-line textbox. If you require more than 4,000 characters, use a **Multiple Lines of Text** data type.
Email	A text value validated as an e-mail address and rendered as a mailto link in the column.
URL	A text value validated as a URL and rendered as a link to open the URL.
Ticker Symbol	A text value for a ticker symbol that displays a link that opens to show a quote for the stock ticker symbol.
Phone	A text value validated as a phone number rendered as link to initiate a phone call by using Skype.

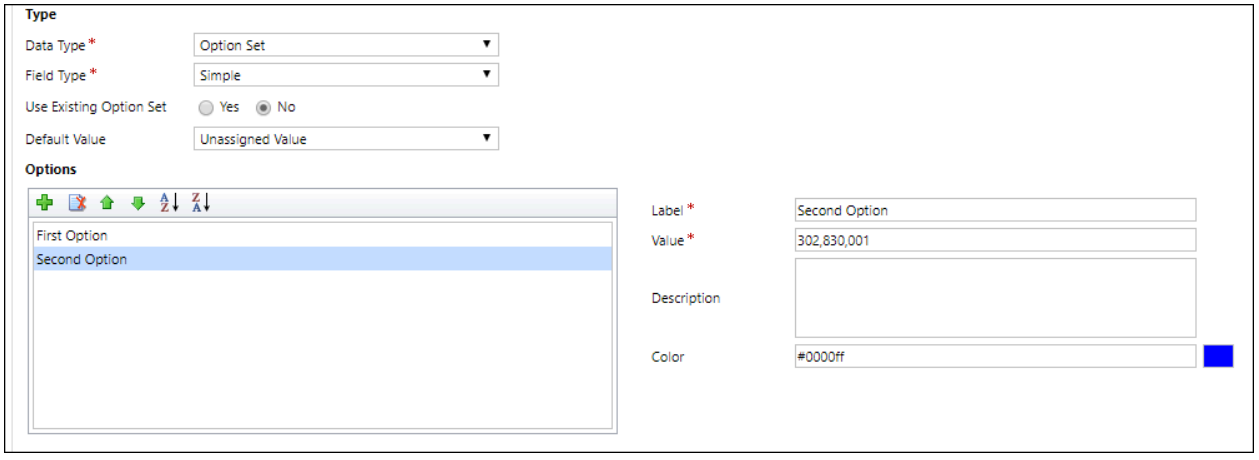
You can also set a **Maximum length** to so that the system won't allow text values longer than you specify.

Choice column options

Columns that provide a set of options can include their own set of *local* options or refer to a common set of *global* options that can be used by multiple columns.

Using a global choice is valuable when you find yourself creating the same set of options for multiple columns. With a global choice, you only need to maintain the set of options in one place.

When you choose **Multi Select Choice** or **Choice** data type the solution explorer designer will provide the option for a local choice by default.



Configure local choices

To define options, in the **Options** area click ✚ to add an option.

For each option you can add the following property values

Expand table

Property	Description
Label	The localizable text for the option.
Value	The number value for the option. This is prefixed with a value defined in the solution publisher for the solution you are working in. This prefix helps ensure that the value will be different from option values that can be defined in another solution and imported into your environment.
External Value	This value is used for virtual entities to map a value in an external data source with this option.
Description	A place to record the purpose for this option. This value is not displayed to application users.
Color	Specifies a color to use in charts for this option. **Tip**: Use a site like w3schools.com HTML Color Picker to find the hex code values for colors.

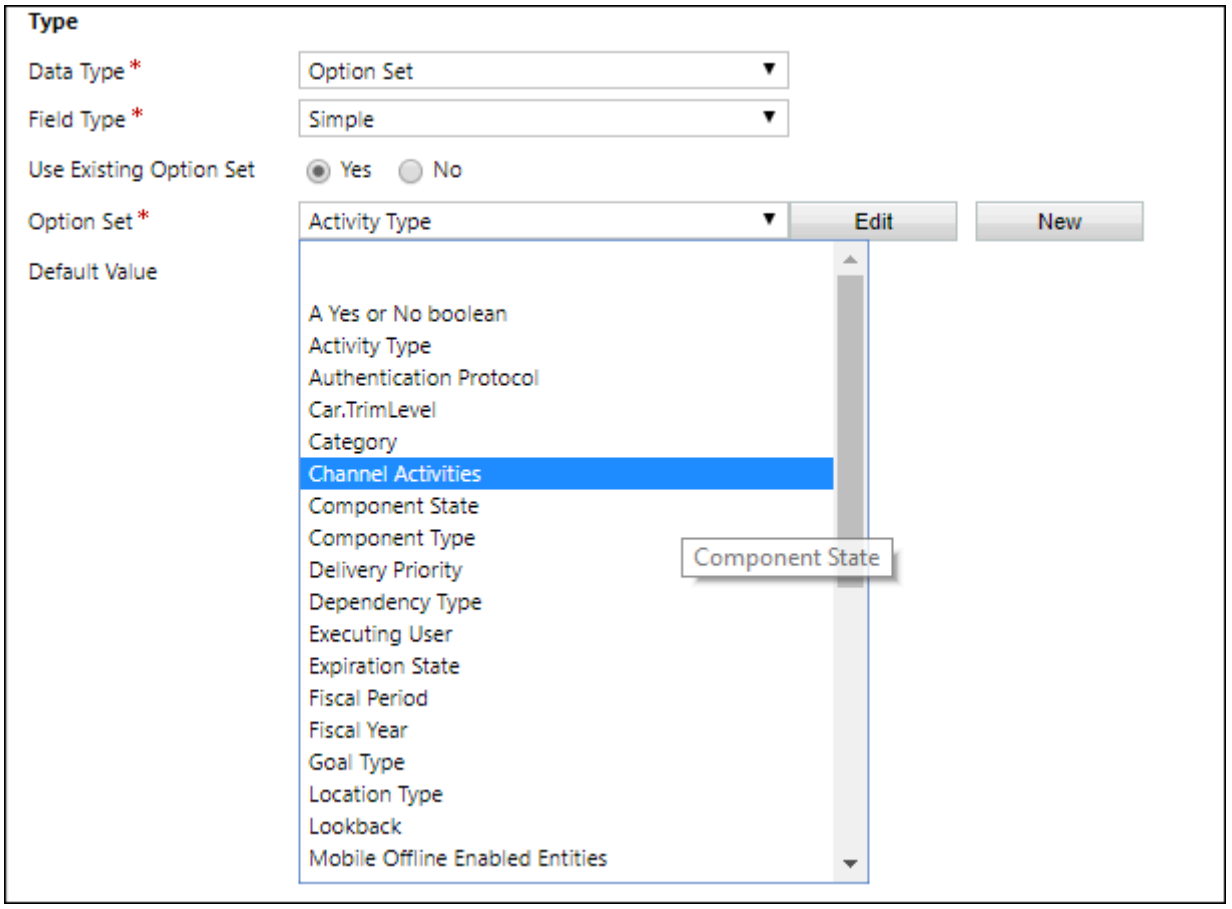
Use the other icons in the options toolbar to perform the following operations:

Expand table

Icon	Operation
	Remove the selected option
	Move the selected option up
	Move the selected option down
	Sort the options by label in ascending order
	Sort the options by label in descending order

Use Existing Choice

If you to choose **Use Existing Option Set** the designer displays a list of existing *global choices* and include an **Edit** and **New** buttons to configure the global choices that this column should use.



You can also configure global choices separately. More information: Create and edit global choices for Dataverse (picklists)

> **Note**
>
> If you define every choice as a global choice your list of global choices will grow and could be difficult to manage. If you know that the set of options will only be used in one place, use a local choice.

Whole number choices

Whole number columns have the following format choices:

Expand table

Format	Description
None	A number value presented in a text box.
Duration	A number value presented as a drop-down list that contains time intervals. A user can select a value from the list or type an integer value that represents the number of minutes.

Format	Description
Timezone	A number value presented as a drop-down list that contains a list of time zones.
Language	A number value presented as a drop-down list that contains a list of languages that have been enabled for the environment. If no other languages have been enabled, the base language is the only option. The value saved is the Locale Identifier (LCID) value for the language.

You can also restrict the maximum or minimum allowed values.

Date and Time options

Date and Time columns have the following options:

Expand table

Format	Description
Date and Time	A date and time value.
Date Only	A date and time value that only displays a date. The time value is stored as 12:00 AM (00:00:00) in the system.

You can also set specific **Behavior** for Date Time columns in the **Advanced options**.

- **User local** : Displays values converted to in the current user's local time zone. This is the default for new columns.
- **Date only**: This behavior is available for the **Date Only** type. Displays values without time zone conversion. Use this for data like birthdays and anniversaries.
- **Time zone independent**: Displays values without time zone conversion.

More information: Behavior and format of the Date and Time column

Column type

You can set a custom column **Field Type** to be a **Simple**, **Calculated**, or a **Rollup** column.

Simple

Simple means that the column isn't a calculated or rollup column.

Calculated

With a calculated column you can enter a formula to assign a value to the column. These data types can be set to calculated columns: **Currency**, **Date and Time**, **Decimal Number**, **Multi Select Choice**, **Choice**, **Single line of text**, **Yes/No**, and **Whole Number**.

More information: Define calculated columns to automate manual calculations

Rollup

With a rollup column, you can set aggregation functions that run periodically to set a number value for the column. These data types can be set to calculated columns: **Currency**, **Date and Time**, **Decimal Number**, and **Whole Number**.

More information: Define rollup columns that aggregate values

Save new column

Once you have configured the column, use one of three commands in the command bar:

Expand table

Command	Description
Save	Save the column definition and leave the form window open.
Save and Close	Save the column definition and close the window.
Save Create New	Save the column definition and open a new form to create a new column.

Edit a column

While viewing columns, select the column you want to edit. Some standard columns or custom columns that are included in a managed solution might not allow you to edit them.

> **Note**
>
> When editing a form, for any column already added to the form you can double-click the column to display the **Column Properties**. On the **Details** tab, select **Edit**. More information: Add, configure, move, or delete columns on a form

After you make changes to a column, you must publish customizations.

- To publish your changes for one table, under **Components**, select **Tables**, and then select the table that you made changes to. On the **Actions** toolbar, select **Publish**.
- To publish all changes you have made to multiple tables or components, on the **Actions** toolbar, select **Publish All Customizations**.

> **Note**
>
> Installing a solution or publishing customizations can interfere with normal system operation. We recommend that you schedule a solution to be published when it's least disruptive to users.

Edit multiple columns

To edit one or more columns, select the column or columns (using the Shift key) you want to modify and then on the **Actions** toolbar, select **Edit**.

When you select multiple columns to edit, the **Edit Multiple Columns** dialog box appears. You can edit **Column Requirement**, **Searchable**, and **Auditing**.

Delete a column

With the system administrator security role, you can delete any custom columns that aren't part of a managed solution. When you delete a column, any data stored in the column is lost. The only way to recover data from a column that was deleted is to restore the database from a point before the column was deleted.

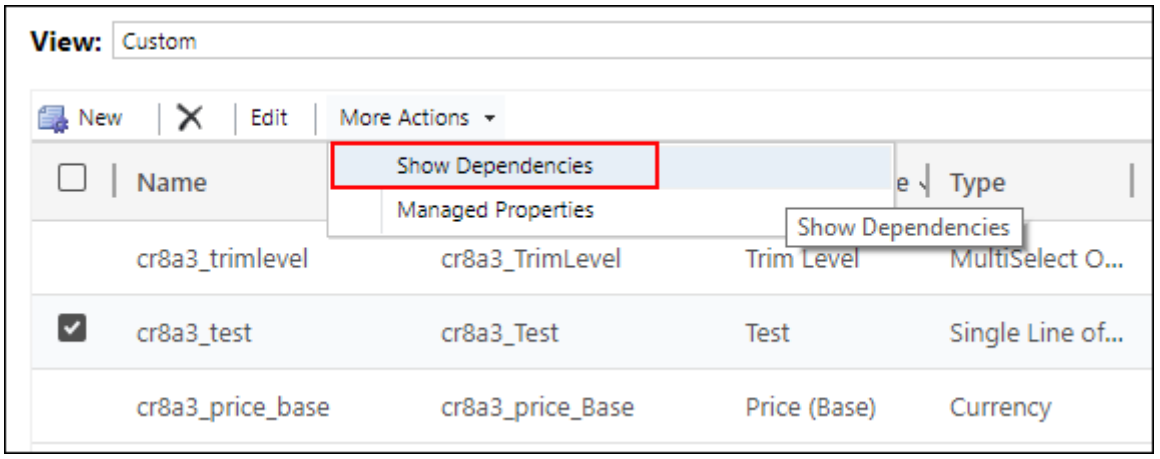
> **Note**
>
> Before you can delete a custom column, you must remove any dependencies that might exist in other solution components.

1. While viewing columns, select a custom column that can be deleted in the list and select the ✕ button in the command bar.
2. In the **Confirm Deletion** dialog, select **Delete**.

> **Tip**
>
> You can select multiple custom columns to be deleted in one operation.

Check column dependencies

Select the column in the list. In the **More Actions** menu, select **Show Dependencies**.



Dependencies are any related use of the column that would prevent it from being deleted. For example, if the column is used in a form or view, you must first remove references to the column in those solution components.

If you delete a lookup column, the 1:N table relationship for it will automatically be deleted.

IME Mode

IME (input method editor) mode specifies how a physical keyboard can be used to enter characters for text columns. IMEs are tools provided by the operating system for composing text. They're commonly used to enter Chinese, Japanese, and Korean words. IME mode doesn't restrict the characters that users can enter. For example, when IME mode is disabled, users can still enter Japanese characters by pasting in a text input.

> **ⓘ Important**
>
> IME Mode is used for backward compatibility with the legacy web client and the IME Mode settings described here don’t apply to Unified Interface apps. Additionally, IME mode is supported only in Internet Explorer and partially supported in Firefox.

- **Active**: This value activates the IME initially. You can deactivate it later if desired. This is the default IME setting.
- **Auto**: When IME mode is auto, Power Apps won't interfere with the IME.
- **Disabled**: Disable IME mode to bypass the IME. This can be useful for entering alphanumeric characters in certain languages.

- **Inactive**: Power Apps deactivates the IME initially. You can activate it later if desired.

See also

Create and edit columns for Dataverse
Create and edit columns for Dataverse using Power Apps portal
Types of columns and column data types
Define calculated columns to automate manual calculations
Define rollup columns that aggregate values
Behavior and format of the Date and Time column

Feedback

Was this page helpful? Yes No

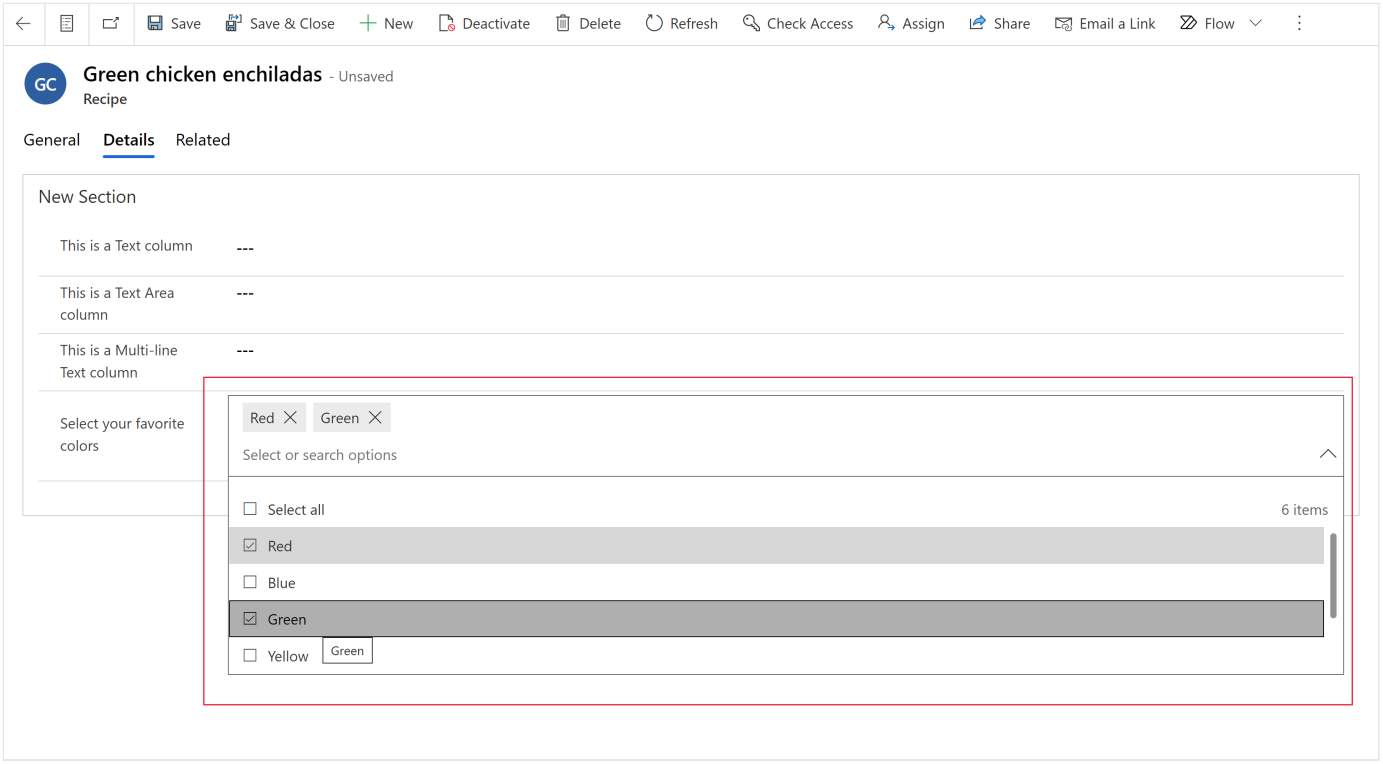
Provide product feedback

Create a choice

Choice columns allow you to include dropdown lists of fixed values to a user within your app to ensure data consistency. Choice columns were formerly referred to as option sets and are sometimes called picklists. Similar to tables, there are both standard choices or makers have the ability to create custom choice columns to use in apps.

Choices can be created while working from a solution in Power Apps (make.powerapps.com) or on a table form in the form designer.

Choice columns can be single selection only (choice) or can allow multi-selection (choices). The following screenshot shows a multi-select choices column in a model-driven app.



Choices are either global or local. You can define a choice to use a set of options defined elsewhere (globally), which can be made available to other choice columns, or define a set of options only available within the choice column (locally). More information: Create and edit choice columns overview

Create a global choice within a solution

1. Sign into powerapps.com select **Solutions**, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. On the command bar, select **New** > **Choice**.

3. Enter the following properties:

   - **Display name**. Enter the **Display name** for your column. The **Display name** is used when presenting this column to your users.
   - In the **Label** field, enter the label you want displayed for the choice option.
   - The **Value** number represents a unique value for the option and isn't displayed in the choice column in an app. Notice there's a number generated for each **Value** automatically. We recommend that you not change this. Having a unique **Value** helps ensure that the value will be different from other option values that might be defined in other solutions and imported into your environment.
   - Select the color picker to the left of the **Label** if you want a color to appear for the option in model-driven app charts.
   - Select the three vertical dots > **Additional properties** to set the following properties:
     - **Description**. Add a description for the option.
     - **External value**. This value is used for virtual tables to map a value in an external data source with this option.
     - **Hidden**. Hide the option from the choice column at runtime in apps. For more information about the limitations of this property, go to Choice hidden property limitations.

     > **⊗ Caution**
     >
     > The Hidden property should never be used as a secure way to prevent users from viewing or editing column values. These properties only apply to app components such as model-driven app forms and views but don't have an effect on a user's data privileges. When a column is hidden, users can still access column data in other ways, such as by making Web API calls. To secure columns, use column-level security to control access.

4. Select **New choice** to create another option for the choice.
5. Repeat the previous step until you have the options you want for the choice.
6. Expand **Advanced options** to display additional properties:

   - **Name**. Unique name of the global choice including the solution publisher prefix.
   - **External type name**. This value is used for virtual tables to map a value in an external data source with this choice.
   - **Description**. Enter an optional description for the choice column.

7. Select **Save**.

> **Important**
>
> Global choices can't be directly added to an app. After you create a global choice column the options from the choice can be made available when you create a local choice column. Local choice columns are table specific and can then be used with forms and views in an app.

Create a local choice to use in forms and views

Create a local choice column for a table that can be used in forms and views within a solution.

1. Sign into powerapps.com select **Solutions**, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the table where you want to create the local choice, and then on the command bar, select **New** > **Choice**.
3. In the **New column** panel, enter properties for the choice column.

Configure properties for a local choice

- **Display name**. Enter the **Display name** for your column. The **Display name** is used when presenting this column to your users.
- **Description**. Enter an optional description of the choice column.
- **Data type**. Select **Choice** > **Choice**. Select **Choice** > **Yes/No** if you want a two option single select choice field.
- **Behavior**. Select calculated to enable the column to be included in calculated columns to automate manual calculations.
- **Required**. Set this value to **Business required** to make it so a record can't be saved without data in this column.
- **Searchable**. When selected, this column appears in Advanced Find and is available when customizing views.
- **Selecting multiple choices is allowed**. Select this option if you want to create a choice where the user can select more than one option (multi-select).
- **Sync with global choice?**
  - **Yes**. Select this option if you want the local choice to have the ability to use options from a global choice. Then, you can also select an existing global choice to use the options

from that global choice.
    - **No**. Select this option if you don't want the choice to have the ability to use options from a global choice.
    - **Sync this choice with**. When **Sync with global choice** is **Yes**, you can select an existing choice column. Then, the options configured for that choice can be used for this choice.
    - If you chose **No** to enable a global choice, create the options for your choice by selecting **Choices** or, if you chose **Yes**, select **Edit choice** to edit the sync choice options or **New choice** to add new options:
    - In the **Label** field, enter the label you want displayed for the choice option.
    - The **Value** number represents a unique value for the option and isn't displayed in the choice column in an app. Notice there's a number generated for each **Value** automatically. We recommend that you not change this. Having a unique **Value** helps ensure that the value will be different from other option values that might be defined in other solutions and imported into your environment.
    - Select the color picker to the left of the **Label** if you want a color to appear for the option in model-driven app charts.
- **Default choice**. Select one of the options you created as the default choice.
- **Schema name**. This is the unique name used by the system and is automatically generated based on the display name and solution publisher prefix. After a new choice column is saved, you can't change this.
- **Enable column security**. Select this option to secure the data in the column beyond the security defined for the table.
- **Enable auditing.** If auditing has been enabled in the environment, this column can be included in change tracking.
- **Appears in dashboard's global filter**. Select this option to allow column to be available as a filter in interactive dashboards.
- **Sortable**. Select this option to allow sorting of this column when used in interactive dashboards.

Select **Save**.

Now the local choice column appears in the form and view designers to add for the table.

Create and edit global choices using the classic solution explorer

For information about how to create and edit global choices using the classic solution explorer, go to Create or edit a global option set (on-premises).

Choice hidden property limitations

- The choice column hidden property is only applicable for model-driven apps. Other components that use Microsoft Dataverse tables, such as canvas apps, don't use the choice column hidden property.
- The choice column hidden property only works for app with the modern, refreshed look for model-driven apps enabled.
- The choice column hidden property isn't currently supported for multi-select choices even when the **Selecting multiple choices is allowed** option is selected.
- The hidden property *only hides the option label and value when displayed in a model-driven app*. Even when hidden, choice labels and values can be viewed and set by accessing the Dataverse table directly, such as from make.powerapps.com. Hidden options can also be set using the setValue client API method.

Last updated on 04/21/2026

Automate calculations by defining calculated columns

Article • 05/22/2024

Use calculated columns to automate otherwise manual calculations used in your business processes.

For example, a salesperson might want to know the weighted revenue for an opportunity, which is based on the estimated revenue from an opportunity multiplied by the probability. Or, they want to automatically apply a discount, if an order is greater than $500. A calculated column can contain values resulting from simple math operations, or conditional operations, such as greater than or if-else, and many others.

Capabilities

- Calculated columns use the columns from the current table or related parent tables.
- The expression support is available on the current table and the related parent table columns in the **Condition** sections and the **Action** sections. The built-in functions include:
  ADDHOURS, **ADDDAYS**, **ADDWEEKS**, **ADDMONTHS**, **ADDYEARS**, **SUBTRACTHOURS**, **SUBTRACTDAYS**, **SUBTRACTWEEKS**, **SUBTRACTMONTHS**, **SUBTRACTYEARS**, **DIFFINDAYS**, **DIFFINHOURS**, **DIFFINMINUTES**, **DIFFINMONTHS**, **DIFFINWEEKS**, **DIFFINYEARS**, **CONCAT**, **TRIMLEFT**, and **TRIMRIGHT**. Learn more in Functions syntax.
- A rich conditional support provides branching and multiple conditions. The logical operations include **AND** and **OR** operators.
- The visual editing capabilities include modern user interface and intellisense in the **ACTION** section.
- A seamless integration of the calculated columns with the forms, views, charts, and reports are available in real time.
- You can configure calculated columns to use custom controls.

Scenarios

- **Weighted Revenue**: Estimated revenue multiplied by probability
- **Net Worth**: Assets subtracted by the liabilities for a given account
- **Cost of Labor**: Base rate up to 40 hours, plus more overtime
- **Contact Number**: Phone number for an opportunity based on account or contact

- **Lead Score**: Single column that provides insights to the quality of a given lead
- **Follow Up By**: Follow up on an activity by a specified number of days based on priority

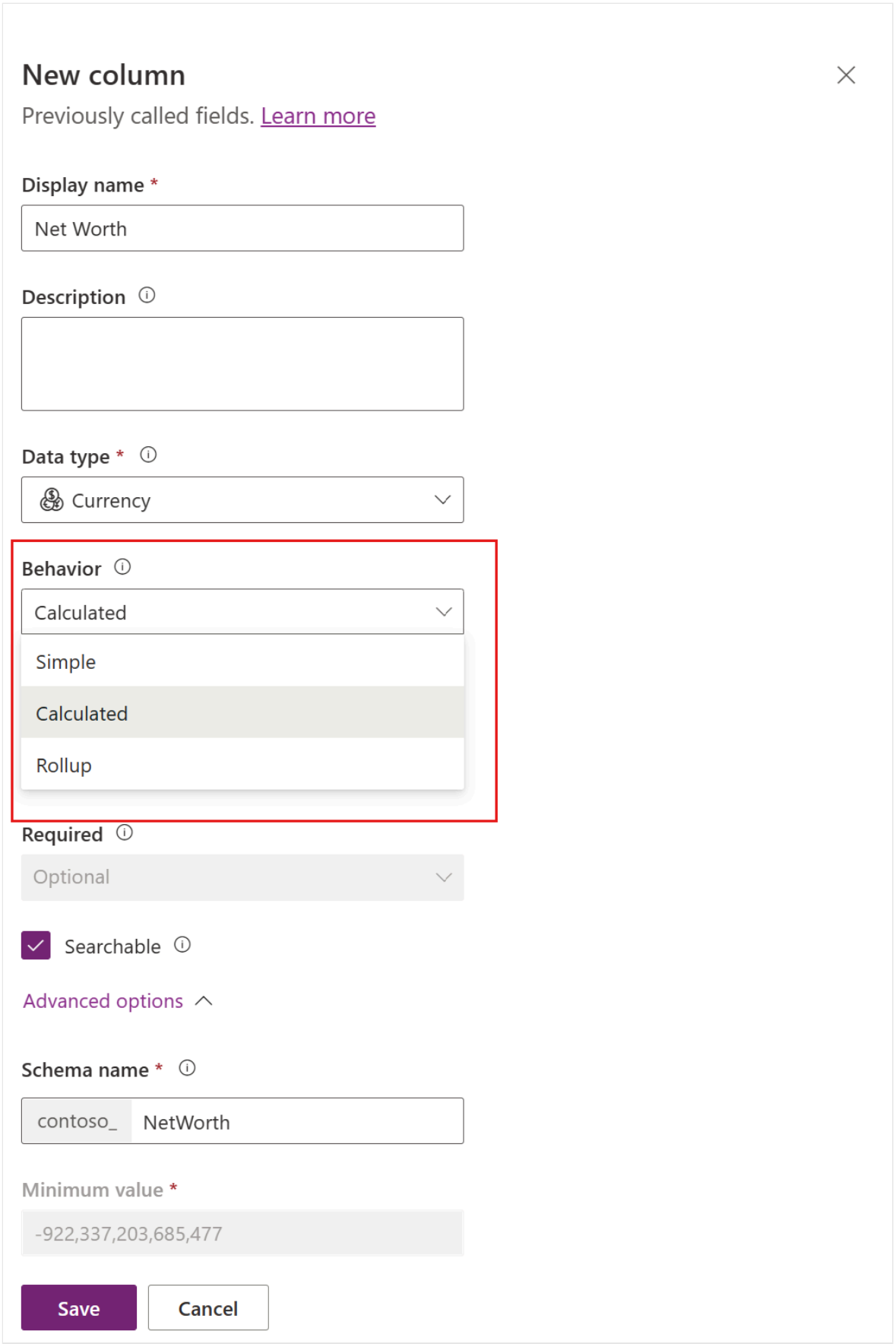
> **Important**
>
> To create a calculated column, you must have the **Write** privilege on the Field Security Profile table. If the calculated column uses the secured columns in a calculation, you should consider securing the calculated column as well, to prevent users from accessing data for which they don't have sufficient permissions. The calculated column editor gives you a warning if you're creating a calculated column that uses secured columns in a calculation, suggesting you secure the calculated column. More information: Column level security to control access.

Column types that support calculations

- Text
- Choice
- Yes/No
- Whole Number
- Decimal Number
- Currency
- Date Time

Create a calculated column

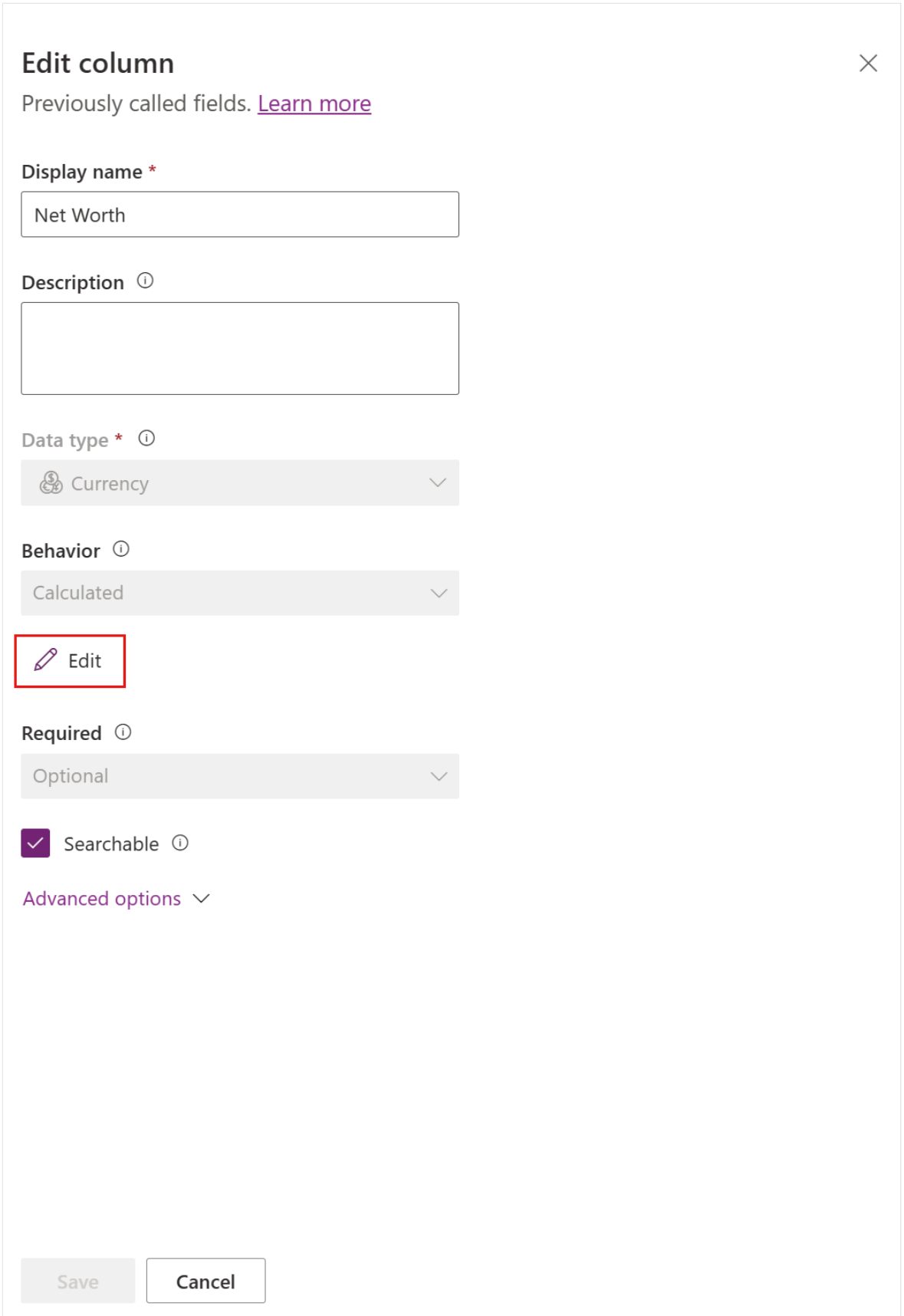
1. Sign into Power Apps
2. Select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Open the unmanaged solution that has the table you want.
4. Select the **Columns** area, and then select **New column** on the command bar.
5. Provide the information for the column, including the **Display name**, **Name**, and **Data type**.
6. If the data type is one of the column types that support calculations, you can make the column a calculated column by selecting **Calculated** from the **Behavior** dropdown list.



7. The **Save and edit** link appears under the dropdown list. Select it.

8. The new column is saved. In the **Columns** list, open the column that you saved in the previous step.

9. On the **Edit column** pane, under the **Behavior** dropdown list, select **Edit**.



10. In the calculated column definition editor, notice that the new calculated column is created, but no formula is set. The calculated column definition consists of two

sections: **CONDITION** and **ACTION**.



a. In the **Condition** section, select **Add condition** to specify a table, column, operator, type, and value. In the dropdown box for the **Table**, you can choose a current table or a related table. In the **Column** dropdown box, you have a selection of all available columns for the table. Depending on the operator you choose, you might need to provide type and value. You can specify multiple conditions using the `AND` or `OR` operators. When you're finished specifying the condition, select the check mark.
b. After you specified the condition, in the **Action** section, select **Add action** to provide the formula for the calculated column. When you're finished specifying the action, select the check mark.

> **Note**
>
> You can use data from Lookup rows within your Action. You first have to select the Lookup column and then type a period. After that, you can select one of the columns available on the related table. For example, with `<LookupFieldName>.<RelatedFieldName>`, you can select: `ParentAccountId.AccountNumber`.
>
> Column level security is ignored on the related table, so if there's sensitive data in the accessed column, secure your calculated column as well.

11. In the calculated column definition editor, select **Save and Close**.

Examples

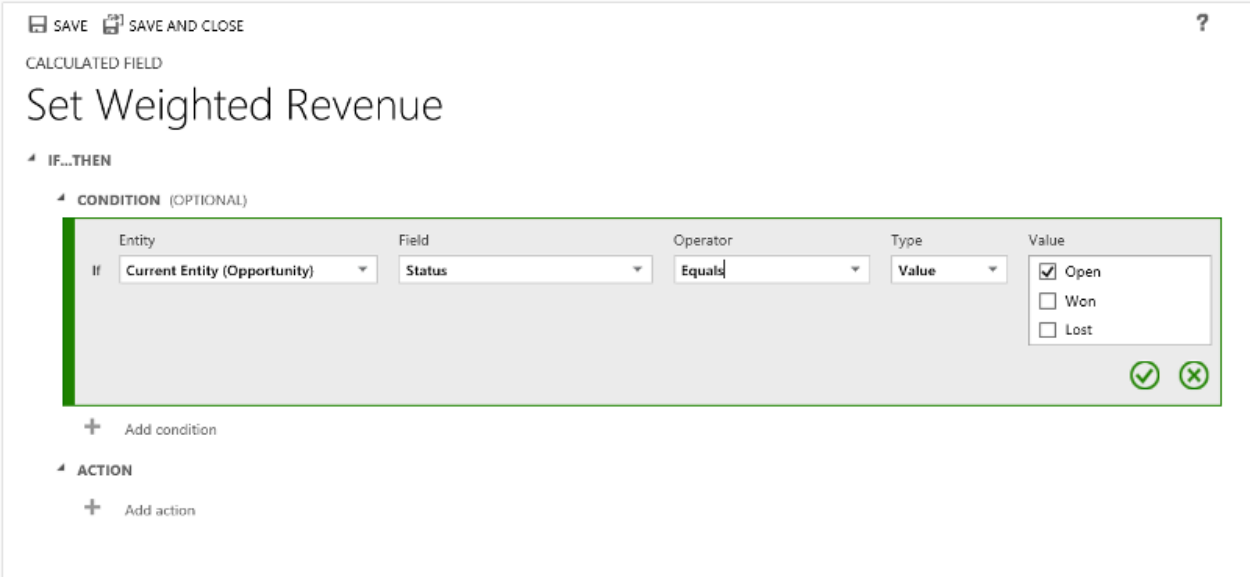
Let's take a look at calculated column examples in more detail.

Weighted revenue of opportunity

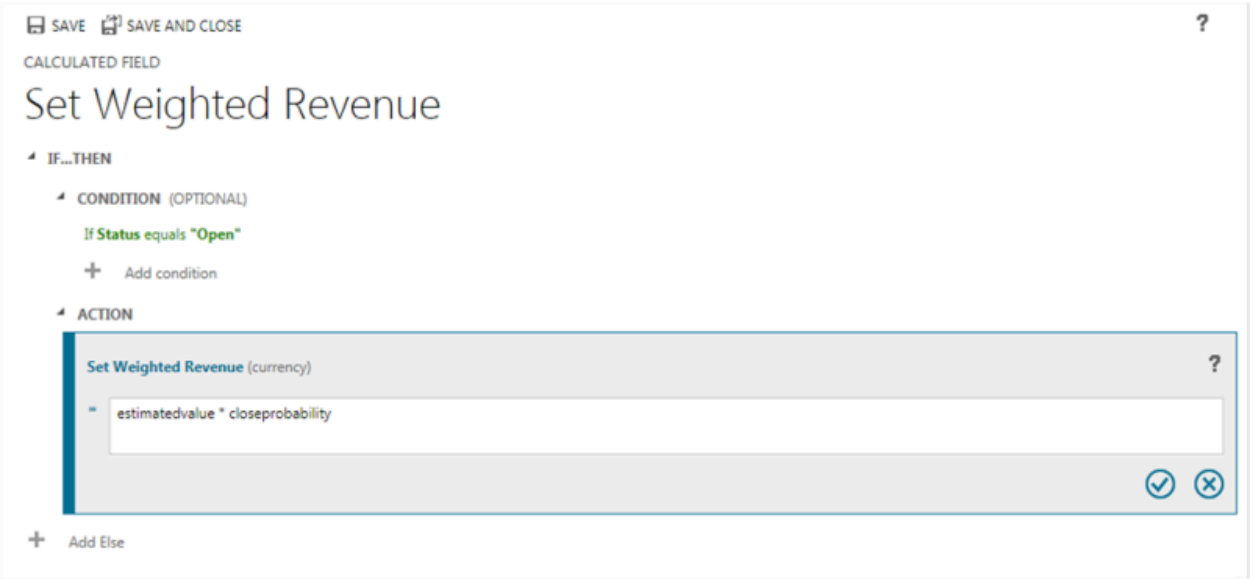
In this example, we're using the columns of the opportunity table to calculate the weighted revenue based on the opportunity's probability. In the column editor for an opportunity table, we create a column called **Weighted Revenue** and specify the column type as **Calculated** and the data type is **Currency**.

In the calculated column definition editor, in the **Condition** section, we specify the opportunity with the Status = Open. In the **ACTION**, the formula calculates the weighted revenue based on the opportunity estimated revenue multiplied by the probability of the opportunity. The following screenshots show step by step how to define the **Weighted Revenue** calculated column.

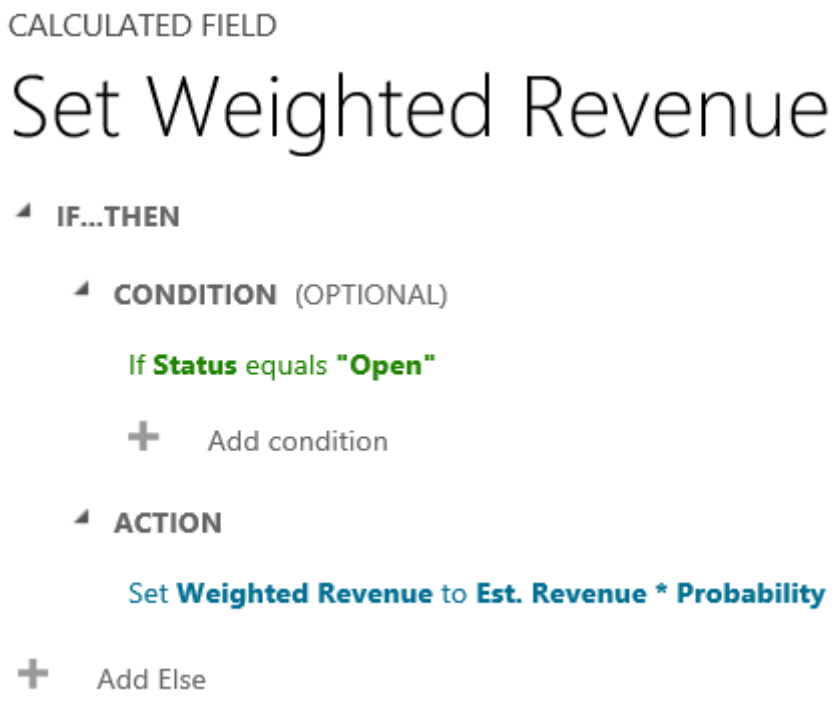
Set the condition on the opportunities:



Provide the formula for the weighted revenue:



Altogether:



Follow up date of opportunity

In this example, we're using the columns of the originated lead of an opportunity to calculate the appropriate date when to follow up on the opportunity.

In the column editor for an opportunity table, we create a column called **Follow-up date** and specify the type as **Calculated** and the data type is **Date and Time**.

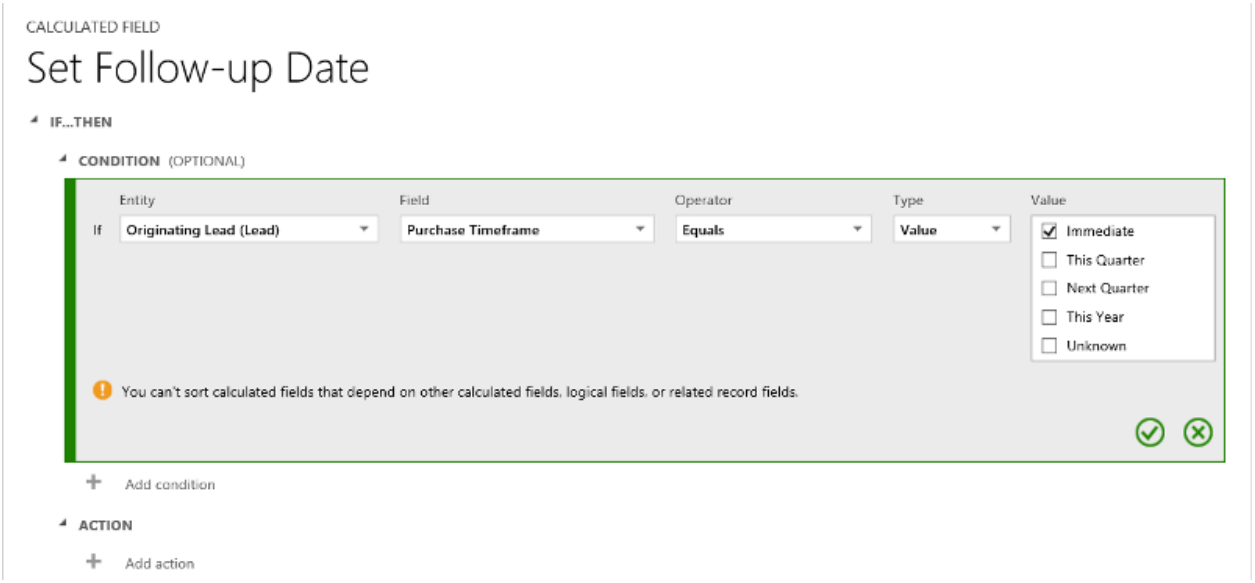
In the calculated column definition editor, in the **Condition** section, we specify two conditions the purchase time frame and the estimated value of the lead.

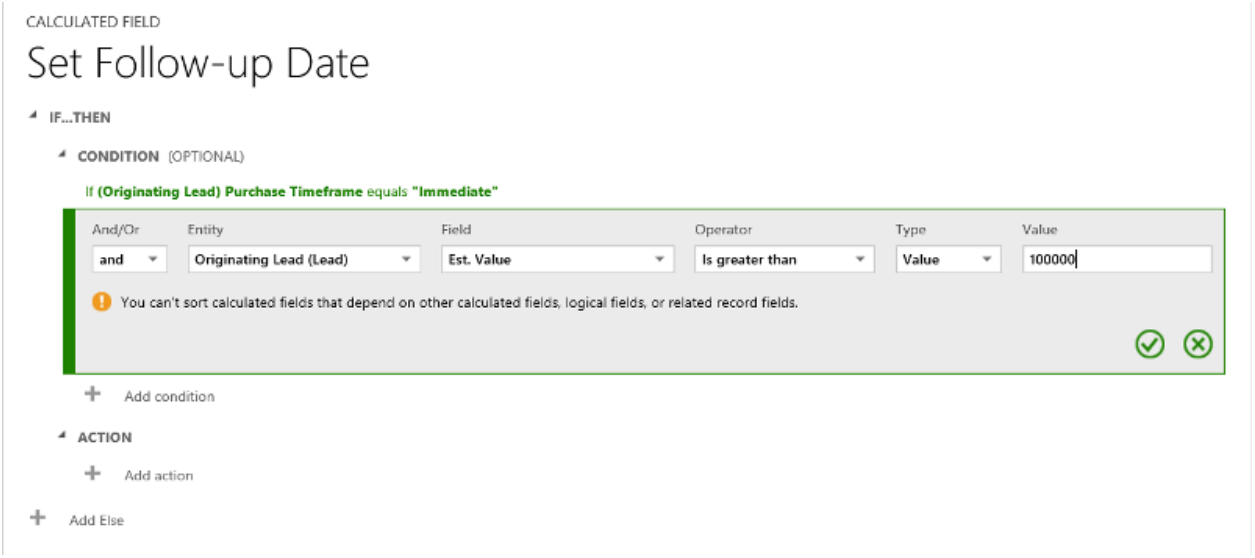
In the **ACTION**, we provide two formulas:

- To follow up in one week on the immediate opportunity
- To follow up in one month if the opportunity isn't likely to happen right away.

The following screenshots show step by step how to define the **Follow-up date** calculated column.

Set the two conditions on the originating lead:

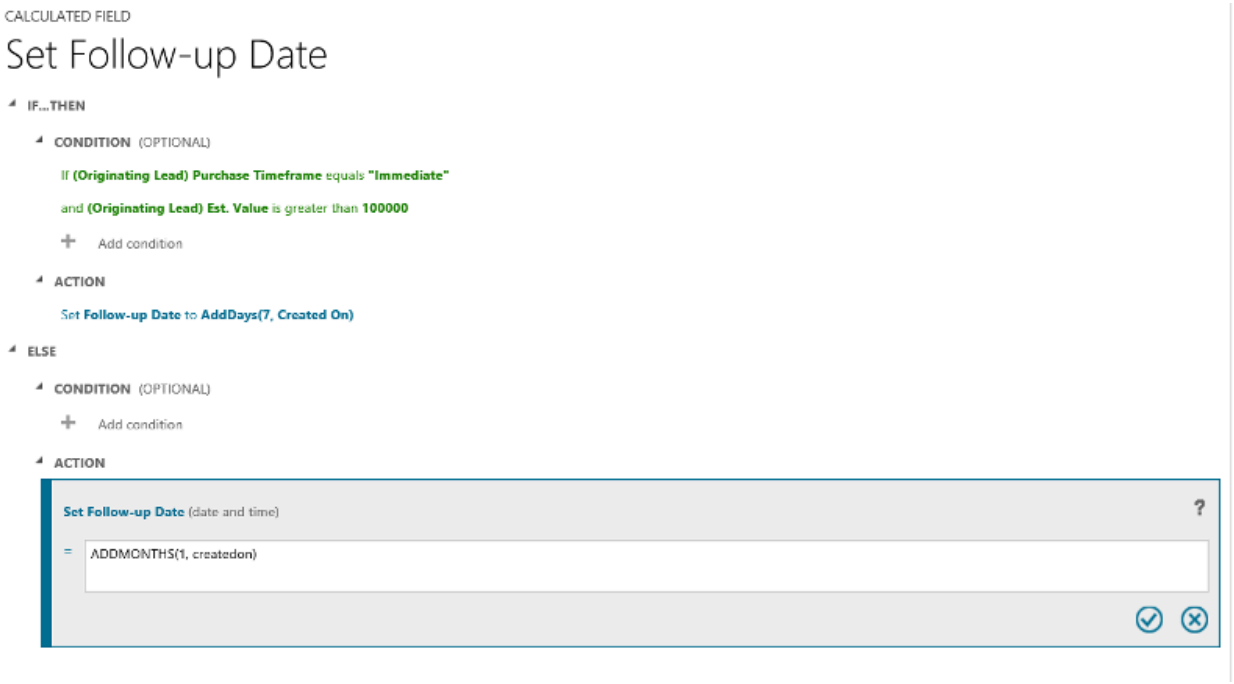




Provide the formula to follow up in one week:



Provide the formula to follow up in one month:



Altogether:

CALCULATED FIELD

Set Follow-up Date

IF...THEN

CONDITION (OPTIONAL)

If **(Originating Lead) Purchase Timeframe** equals **"Immediate"**

and **(Originating Lead) Est. Value** is greater than **100000**

\+ Add condition

ACTION

Set **Follow-up Date** to **AddDays(7, Created On)**

ELSE

CONDITION (OPTIONAL)

\+ Add condition

ACTION

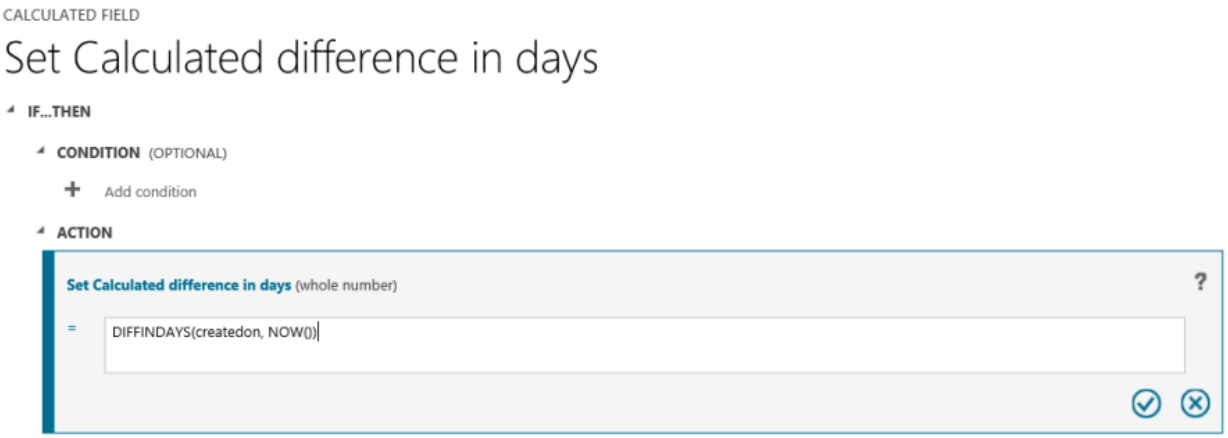
Set **Follow-up Date** to **AddMonths(1, Created On)**

Days from a row creation

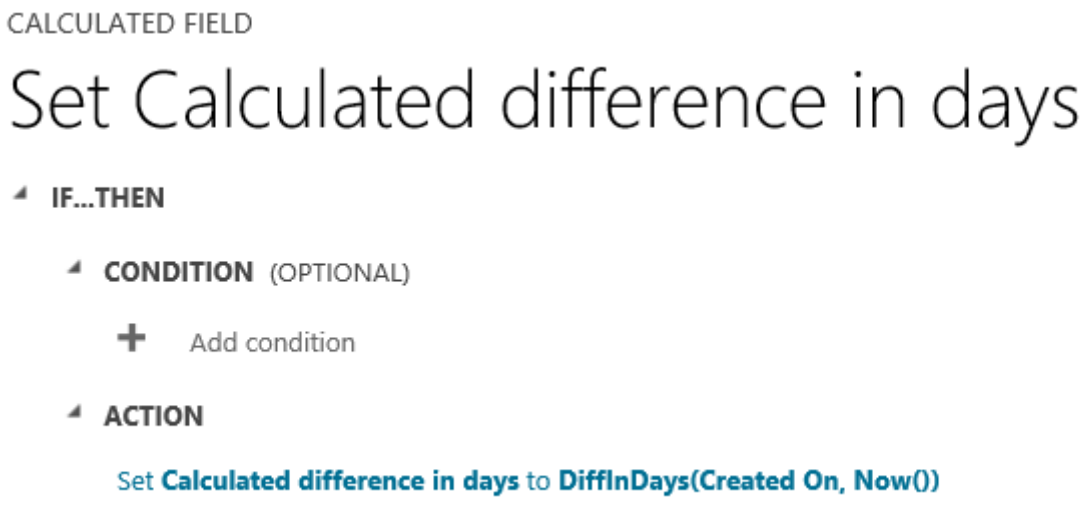
In this example, we're using the **DIFFINDAYS** function, to compute the difference in days from the time when a row was created to the current date.

Create a new Whole Number column called **Calculated difference in days**.

Provide the formula for computing the difference in days



Altogether:



Functions syntax

The following table contains information about the syntax for the functions provided in the **ACTION** section of the calculated column.

> **Tip**

The function names are specified in uppercase letters.

Expand table

Function Syntax	Description	Return type
ADDDAYS (whole number, date and time)	Returns a new date and time equal to the given date and time, plus the specified number of days.	Date and Time
ADDHOURS (whole number, date and time)	Returns a new date and time equal to the given date and time, plus the specified number of hours.	Date and Time
ADDMONTHS (whole number, date and time)	Returns a new date and time equal to the given date and time, plus the specified number of months.	Date and Time
ADDWEEKS (whole number, date and time)	Returns a new date and time equal to the given date and time, plus the specified number of weeks.	Date and Time
ADDYEARS (whole number, date and time)	Returns a new date and time equal to the given date and time, plus the specified number of years.	Date and Time
SUBTRACTDAYS (whole number, date and time)	Returns a new date and time equal to the given date and time, minus the specified number of days.	Date and Time
SUBTRACTHOURS (whole number, date and time)	Returns a new date and time equal to the given date and time, minus the specified number of hours.	Date and Time
SUBTRACTMONTHS (whole number, date and time)	Returns a new date and time equal to the given date and time, minus the specified number of months.	Date and Time
SUBTRACTWEEKS (whole number, date and time)	Returns a new date and time equal to the given date and time, minus the specified number of weeks.	Date and Time
SUBTRACTYEARS (whole number, date and time)	Returns a new date and time equal to the given date and time, minus the specified number of years.	Date and Time
DIFFINDAYS (date and time, date and time)	Returns the difference in days between two **Date and Time** columns. If both dates and times fall on the same day, the difference is zero.	Whole Number

Function Syntax	Description	Return type
DIFFINHOURS (date and time, date and time)	Returns the difference in hours between two **Date and Time** columns.	Whole Number
DIFFINMINUTES (date and time, date and time)	Returns the difference in minutes between two **Date and Time** columns.	Whole Number
DIFFINMONTHS (date and time, date and time)	Returns the difference in months between two **Date and Time** columns. If both dates and times fall on the same month, the difference is zero.	Whole Number
DIFFINWEEKS (date and time, date and time)	Returns the difference in weeks between two **Date and Time** columns. If both dates and times fall on the same week, the difference is zero.	Whole Number
DIFFINYEARS (date and time, date and time)	Returns the difference in years between two **Date and Time** columns. If both dates and times fall on the same year, the difference is zero.	Whole Number
CONCAT (single line of text, single line of text, ... single line of text)	Returns a string that is the result of concatenating two or more strings.	String
TRIMLEFT (single line of text, whole number)	Returns a string that contains a copy of a specified string without the first N-characters.	String
TRIMRIGHT (single line of text, whole number)	Returns a string that contains a copy of a specified string without the last N-characters.	String

> **Note**
>
> All DIFF functions require that the first **Date and Time** column and the second **Date and Time** column have the same behavior: **User Local**, **Date Only**, or **Time-Zone Independent**. If the behavior of the second column doesn't match the behavior of the first column, the error message is shown, indicating that the second column can't be used in the current function. More information: Behavior and format of the Date and Time column.

> **Tip**
>
> You can't enter a date, such as 01/01/2015, as the Date value in a calculated column. Date and DateTime values can only be set or compared using other DateTime columns.

In the **CONCAT** function, you can use literal strings as single lines of text, table columns that contain a single line of text, or a combination of both. For example: **CONCAT** (FirstName, LastName, "is a manager."). If a literal string contains quotation marks, precede each mark with the backslash (\) escape character, like this: `This string contains the \"quotation marks.\"` This ensures that the quotation marks inside the string aren't treated as special characters that separate the strings.

The following examples show how to use the **TRIMLEFT** and **TRIMRIGHT** functions. They contain the initial strings and the resulting strings, returned by the **TRIMLEFT** and **TRIMRIGHT** functions:

TRIMLEFT ("RXX10-3456789", 3), returns the string `10-3456789` **TRIMRIGHT** ("20-3456789RXX", 3), returns the string `20-3456789`

Considerations

You should be aware of certain conditions and limitations when working with calculated columns:

- Saved queries, charts, and visualizations can have a maximum of 50 unique calculated columns.
- The calculated column values aren't displayed in the Outlook Client Offline mode in the tile views or on table main forms.
- A maximum number of chained calculated columns is 5.
- A calculated column can't refer to itself or have cyclic chains.
- If you change one of the condition operators in a multiple condition clause, all of the condition operators update to that condition. For example, in the clause `IF (x > 50) OR (y ==10) OR (z < 5)`, if you change the `OR` operator to the `AND` operator, then all `OR` operators in the clause become `AND` operators.
- You can access parental columns via the Lookup column to the parent table, such as *`<LookupFieldName>.<FieldName>`*. You can't access multi-table lookup columns like Customer, which can be Account or Contact. However, some tables have individual Lookup columns for a specific table, such as `ParentAccountid.`*`<FieldName>`* or `ParentContactid.`*`<FieldName>`*.
- Sorting is disabled on:
  - A calculated column that contains a column of a parent row.

    - A calculated column that contains a logical column (for example, address column)
    - A calculated column that contains another calculated column.
    - A calculated column that contains `Now()` function.

- Calculated columns can span two tables only.
    - A calculated column can contain a column from another table (spanning two tables – current table and parent row).
    - A calculated column can't contain a calculated column from another table that also contains another column from a different table (spanning three tables): (Current table) Calculated Column ← (Parent Row) Calculated Column 1 ← (Parent Row) Calculated Column 2.

- You can't trigger workflows or plug-ins on calculated columns.

- You can't change an existing simple column to a calculated column. If your current application is using JavaScript or plug-ins to calculate a column, you wouldn't be able to use the calculated columns feature without creating a new column.

- Duplicate detection rules aren't triggered on calculated columns.

- A rollup can't reference a calculated column that uses another calculated column, even if all the columns of the other calculated column are on the current table.

- When a currency calculated column depends on a related table's currency column, the system performs calculations using corresponding base currency column values. These calculations are because exchange rate values and currency values can vary between different table records. For example, consider a calculated column - `Account Revenue` on the `Opportunity` table, and consider `Account Revenue` has a dependency on the `Annual Revenue` currency column on the `Account` table, then `Account Revenue` is calculated as described here:

  `'Account Revenue' = ['Annual Revenue (Base)' * 'Exchange Rate on Opportunity table record']` or `'Account Revenue' = [('Annual Revenue' / 'Exchange Rate on Account table record') * 'Exchange Rate on Opportunity table record']`

See also

Create and edit columns
Define rollup columns that aggregate values
Video: Rollup and calculated columns
Formula, calculated, and rollup columns using code

Feedback

Was this page helpful? Yes No

Provide product feedback

Define rollup columns that aggregate values

Article • 04/30/2024

Rollup columns help users obtain insights into data by monitoring key business metrics. A rollup column contains an aggregate value computed over the rows related to a specified row. This includes regular tables and activity tables such as emails and appointments.

In more complex scenarios, you can aggregate data over the hierarchy of rows. As an administrator or customizer, you can define rollup columns by using the customization tools in Power Apps, without needing to write code.

Rollup columns benefits and capabilities

The benefits and capabilities of rollup columns include:

- Visual editing is easy. You can create rollup columns by using the Column Editor, just like you do when you create a regular column.
- Wide selection of aggregate functions. You can aggregate data by using the following functions: `SUM`, `COUNT`, `MIN`, `MAX` and `AVG`.
- Full filter support for aggregation. You can set various filters for the source table or related table while setting multiple conditions.
- Seamless integration with the user interface. You can include the rollup columns in forms, views, charts, and reports.
- Rollup columns are solution components. You can easily transport the rollup columns as components between environments and distribute them in solutions.
- Rollup columns and the calculated columns are complementary to each other. You can use a rollup column as a part of the calculated column, and vice versa.
- You can configure rollup columns to use custom controls.

Some examples of rollup columns include:

- Total estimated revenue of open opportunities of an account
- Total estimated revenue of open opportunities across all accounts in a hierarchy
- Total estimated revenue of an opportunity including child opportunities
- Total estimated value of qualified leads generated by a campaign
- Number of high priority open cases across all accounts in a hierarchy
- Earliest created time of all high priority open cases for an account

Each Rollup column creates two accessory columns with <*columnname*>`_date` and <*columnname*>`_state` suffix pattern. The `_date` column contains DateTime data and `_state` column contains Integer data. The `_state` column has the following values:

Expand table

Value	State	Description
0	NotCalculated	The column value is yet to be calculated.
1	Calculated	The column value has been calculated per the last update time in _date column.
2	OverflowError	The column value calculation resulted in overflow error.
3	OtherError	The column value calculation failed due to an internal error. The following run of the calculation job will likely fix it.
4	RetryLimitExceeded	The column value calculation failed because the maximum number of retry attempts to calculate the value was exceeded due to high number of concurrency and locking conflicts.
5	HierarchicalRecursionLimitReached	The column value calculation failed because the maximum hierarchy depth limit for the calculation was reached.
6	LoopDetected	The column value calculation failed because a recursive loop was detected in the hierarchy of the row.
7	CurrencyMissing	The column value calculation failed because the required field currency is missing.

Rollup calculations

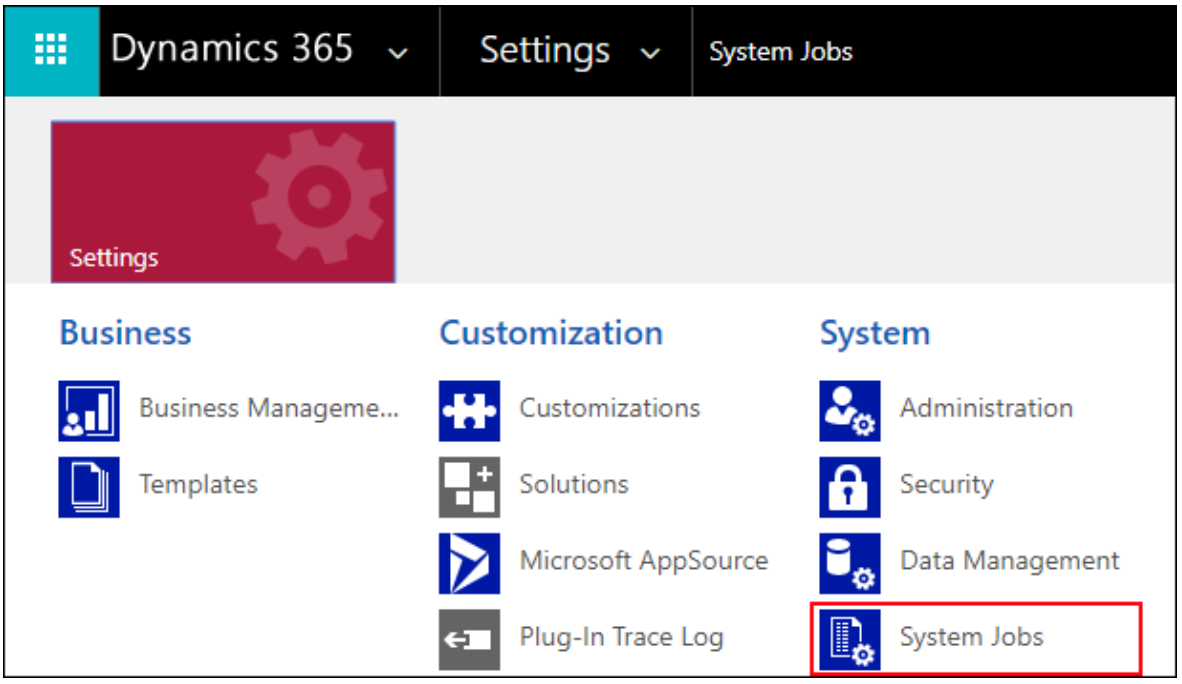
The rollups are calculated by scheduled system jobs that run asynchronously in the background. You have to be an administrator to view and manage the rollup jobs.

View Rollup jobs

To view rollup jobs:

1. Sign in to Power Apps, and then select the environment you want.
2. Select **Settings** (gear) > **Advanced settings**.

3. Select **Settings** > **System Jobs**.



4. In the view selector, choose **Recurring System Jobs**.
5. To quickly find a relevant job, you can filter by the System Job type: **Mass Calculate Rollup Field** or **Calculate Rollup Field**.

Mass Calculate Rollup Field

Mass Calculate Rollup Field is a recurring job, created for each rollup column. It runs once, after you create or update a rollup column. The job recalculates the specified rollup column value in all existing records that contain this column. By default, the job will run 12 hours after you have created or updated a column. When the column is modified, the job resets to run again in 12 hours after the update. The 12-hour delay is needed to assure that the Mass Calculate Rollup Field runs during the nonoperational hours of the environment.

We recommend that an administrator adjusts the start time of a Mass Calculate Rollup Field job after the rollup column is created or modified, in such a way that it runs during nonoperational hours. For example, midnight would be a good time to run the job to assure efficient processing of the rollup columns.

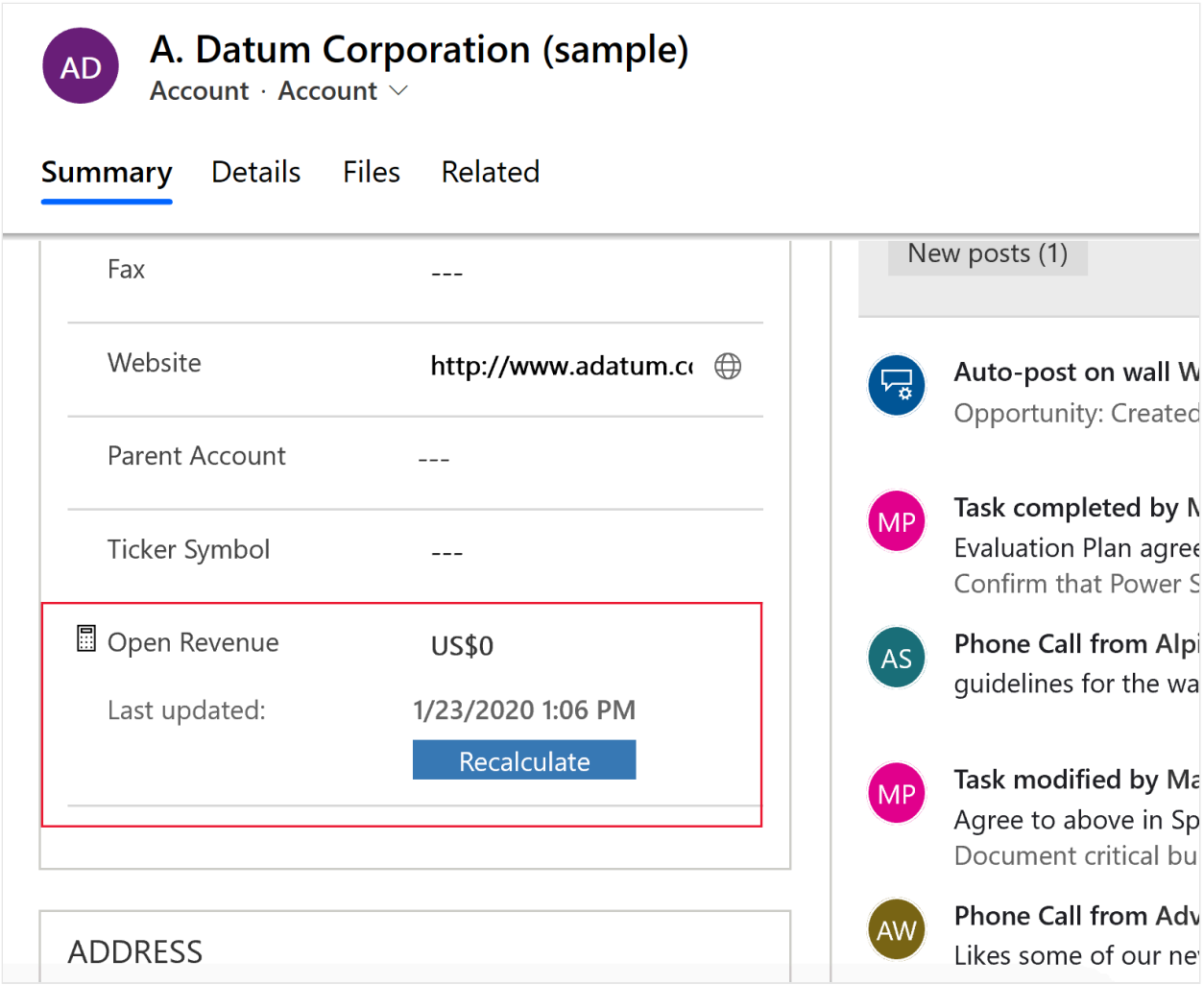
Notice that, when the situation occurs where the rollup column is never updated, the Mass Calculate Rollup Field job won't run again for that column until after 10 years have passed. This behavior is by design. When there aren't any updates in the underlying columns associated with the rollup field, there isn't any reason to run the job more frequently.

Calculate Rollup Column

Calculate Rollup Field is a recurring job that does incremental calculations of all rollup columns in the existing rows for a specified table. There's only one **Calculate Rollup Field** job per table. The incremental calculations mean that the **Calculate Rollup Field** job processes the rows that were created, updated, or deleted after the last **Mass Calculate Rollup Field** job finished execution. The default minimum recurrence setting is one hour. The job is automatically created when the first rollup column on a table is created and deleted when the last rollup column is deleted.

Online recalculation option

The rollup column on the form displays a calculator image, rollup value, and the time of the last calculation. To recalculate, select the calculator image, and then select the **Recalculate** button that appears.



There are a few considerations you should keep in mind when using the online recalculation option (manual refresh on the form):

- You must have Write privileges on the table and Write access rights on the source row on which you're requesting the Refresh. For example, if you're calculating the

estimated revenue from the open opportunities of an account, you don't have to have Write privileges on the opportunity table, only on the account table.
- This option is only available in the online mode. You can't use it while working offline.
- The maximum number of rows during the rollup refresh is limited to 50,000 rows. In case of the hierarchical rollup, this applies to the related rows across the hierarchy. If the limit is exceeded, you see an error message: *Calculations can't be performed online because the calculation limit of 50,000 related rows has been reached.* This limit doesn't apply when the rollup is automatically recalculated by the system jobs.
- The maximum hierarchy depth is limited to 10 for the source row. If the limit is exceeded, you see an error message: *Calculations can't be performed online because the hierarchy depth limit of 10 for the source row has been reached.* This limit doesn't apply when the rollup is automatically recalculated by the system jobs.

Modify rollup job recurrence

As a system administrator, you can modify the rollup job recurrence pattern, postpone, pause, or resume the rollup job. However, you can't cancel or delete a rollup job.

To pause, postpone, resume, or modify the recurrence pattern, you must view the system jobs. More information View Rollup jobs

On the nav bar, choose **Actions** and select the action you want.

For the **Mass Calculate Rollup Field** job, the available selections are: **Resume**, **Postpone**, and **Pause**.

For the **Calculate Rollup Field** job, the available selections are: **Modify Recurrence**, **Resume**, **Postpone**, and **Pause**.

Examples

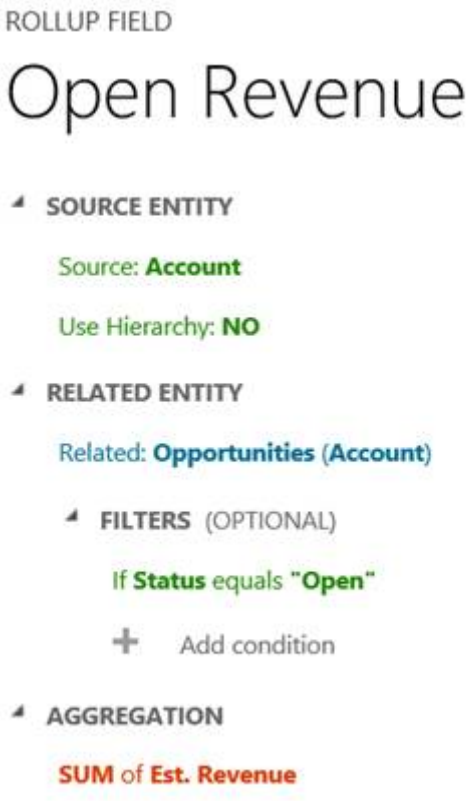
Let's take a look at several rollup column examples. We'll aggregate data for a row from the related rows with and without using a hierarchy. We'll also aggregate data for a row from related activities and activities indirectly related to a row via the ActivityParty table. In each example, we define the rollup column by using the Column Editor. To open the Column Editor, open solution explorer and expand **Components** > **Tables**. Select the table you want and select **Columns**. Choose **New**. In the editor, provide the required information for the column, including the **Column Type** and **Data Type**. In the **Column**

Type, select **Rollup**, after you have selected the data type. The data types include decimal or whole numbers, currency, and date/time. Choose the **Edit** button next to the **Column Type**. This takes you to the rollup column definition editor. The rollup column definition consists of three sections: **Source table**, **Related table, and **Aggregation**.

- In the **Source table** section, you specify the table for which the rollup column is defined and whether or not you aggregate over a hierarchy. You can add filters with multiple conditions to specify the rows in the hierarchy you want to use for rollup.
- In the **Related table** section, you specify the table over which you aggregate. This section is optional when you choose to roll up over the hierarchy on the source table. You can add filters with multiple conditions to specify which related rows to use in the calculation. For example, you include the revenue from the open opportunities with an annual revenue greater than $1,000.
- In the **Aggregate** section, you specify the metric you want to compute. You can choose available aggregate functions, such as SUM, COUNT, MIN, MAX, or AVG.

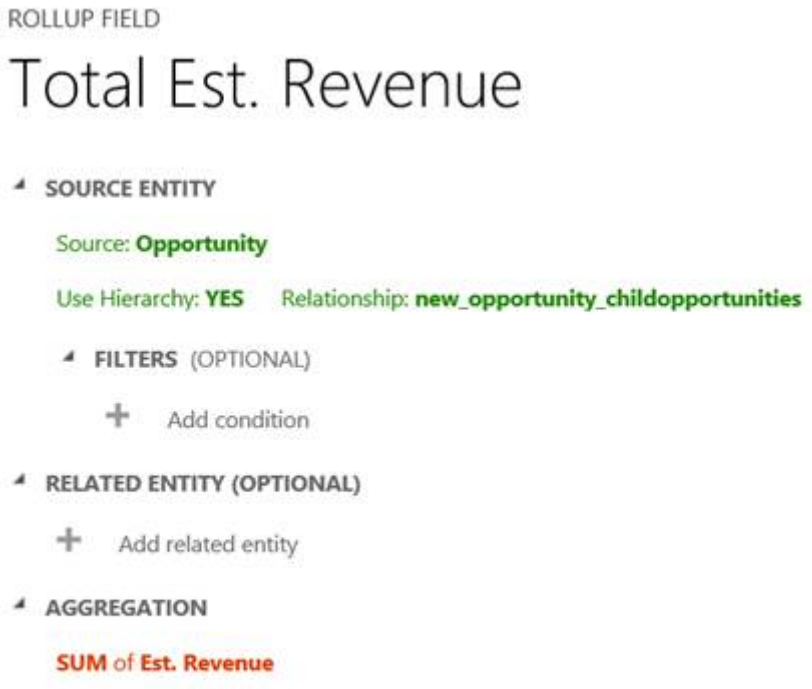
Aggregate data for a row from related rows

In this example, a hierarchy isn't used. The total estimated revenue is calculated for an account, from the related open opportunities.



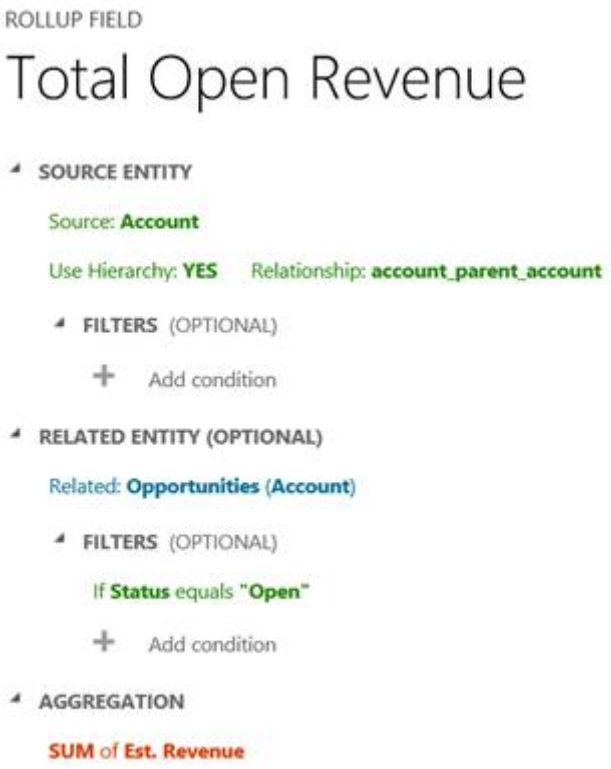
Aggregate data for a row from the child rows, over the hierarchy

In this example, we calculate the total estimated revenue of an opportunity including the child opportunities, over the hierarchy.



Aggregate data for a row from the related rows, over the hierarchy

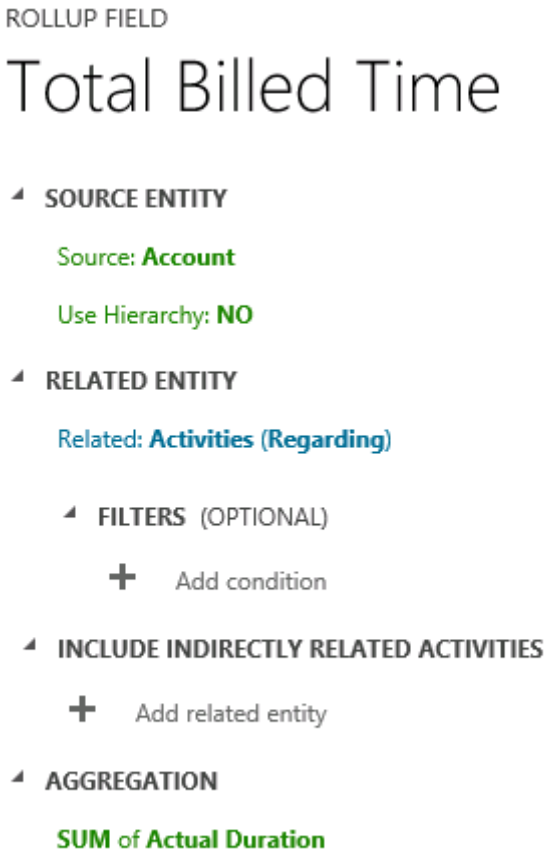
In this example, we calculate the total estimated revenue of open opportunities across all accounts, over the hierarchy.



Aggregate data for a row from all related activities

In this example, we calculate the total time spent and billed from all activities related to an account. This might include time spent on the phone, at appointments, or on custom activities.

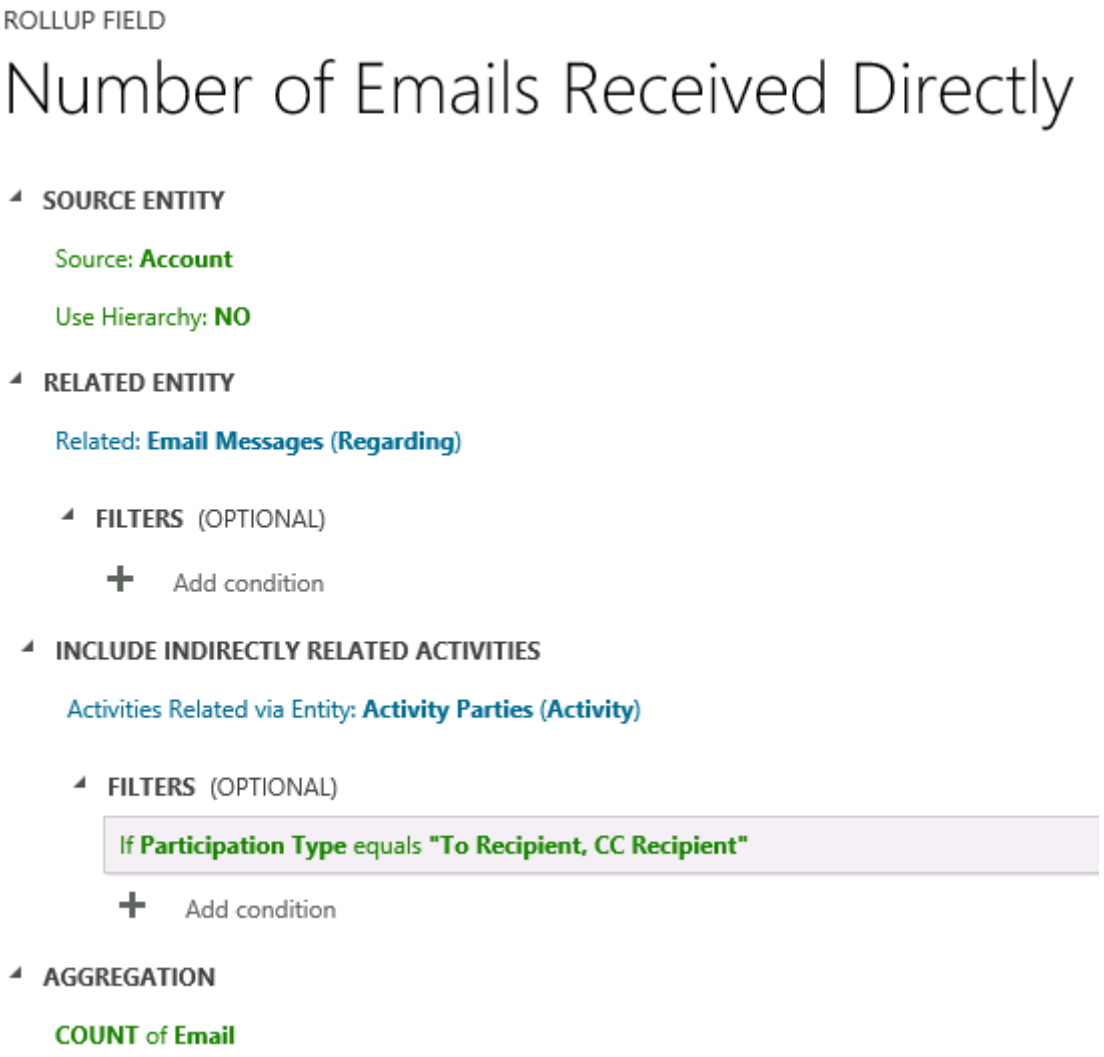
In earlier releases, you could define a rollup column for an individual activity, such as a phone call, fax, or appointment. But, to achieve the result of the example shown below, you had to total the data by using the calculated columns. Now, you can do it all in one step by defining one rollup column for the Activity table.



Aggregate data for a row from all related activities and activities indirectly related via the Activity Party table

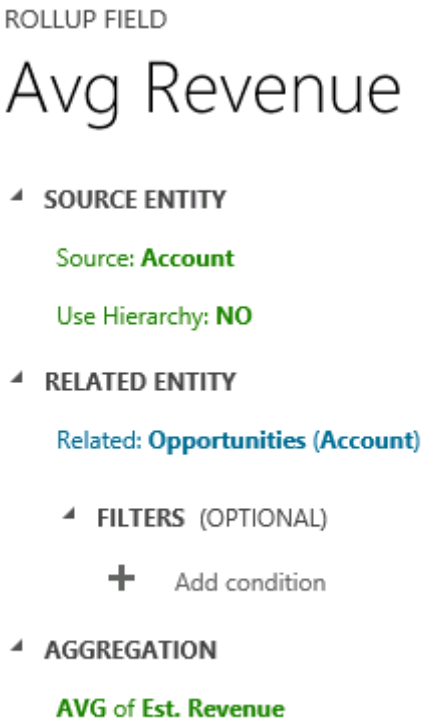
In this example, we count the total number of emails sent to an account, where the account is listed on the email's "To Recipient" line or "Cc Recipient line. This is done by specifying the **Participation Type** in **FILTERS** for the Activity Party table in the rollup column definition. If you don't use filtering, then all available participation types for an activity are used in the calculation.

For more information about the Activity Party table and participation types available for a particular activity, see ActivityParty table.

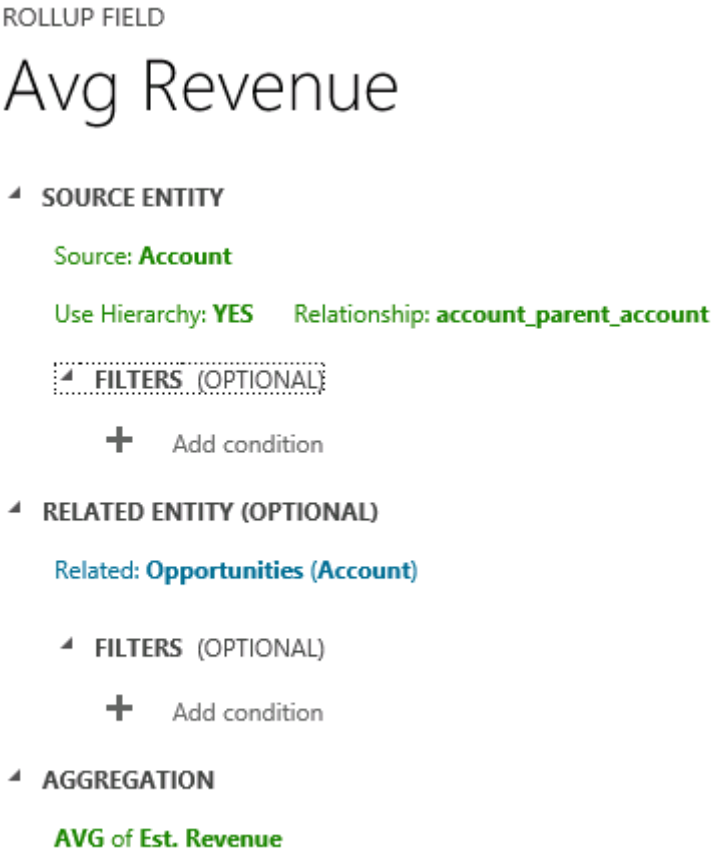


Aggregate data for a row from related rows using the AVG operator

In this example, we calculate an average estimated revenue from all opportunities related to an account.



The following example shows how to calculate an average estimated revenue from related opportunities over a hierarchy of accounts. An average estimated revenue can be seen at each level in the hierarchy.



Rollup column considerations

You should be aware of certain conditions and restrictions when working with rollup columns:

- By default, you can define a maximum of 200 rollup columns for the environment and up to 50 rollup columns per table. More information: Define the maximum number of rollup columns in an environment or table
- A workflow can't be triggered by the rollup column updates.
- A workflow wait condition can't use a rollup column.
- A rollup over the rollup column isn't supported.
- A rollup can't reference a calculated column that uses another calculated column, even if all the columns of the other calculated column are on the current table.
- A rollup column doesn't update automatically if it uses a formula column that's dependent on time bound functions `Now()` and `IsUTCToday()`. The online recalculation option (a manual refresh on the form) or **Recalculate** button must be used to calculate the latest rollup column value.
- The rollup can only apply filters to the source table or related tables, simple columns or noncomplex calculated columns.
- A rollup can be done only over related tables with the 1:N relationship. A rollup can't be done over the N:N relationships.
- A rollup can't be done over the 1:N relationship for the Activity table or the Activity Party table.
- The business rules, workflows, or calculated columns always use the last calculated value of the rollup column.
- A rollup column is aggregated under the system user context. All users are able to see the same rollup column value. You can control the rollup column visibility with the column level security (FLS), by restricting who can access the rollup column. More information Column level security to control access.

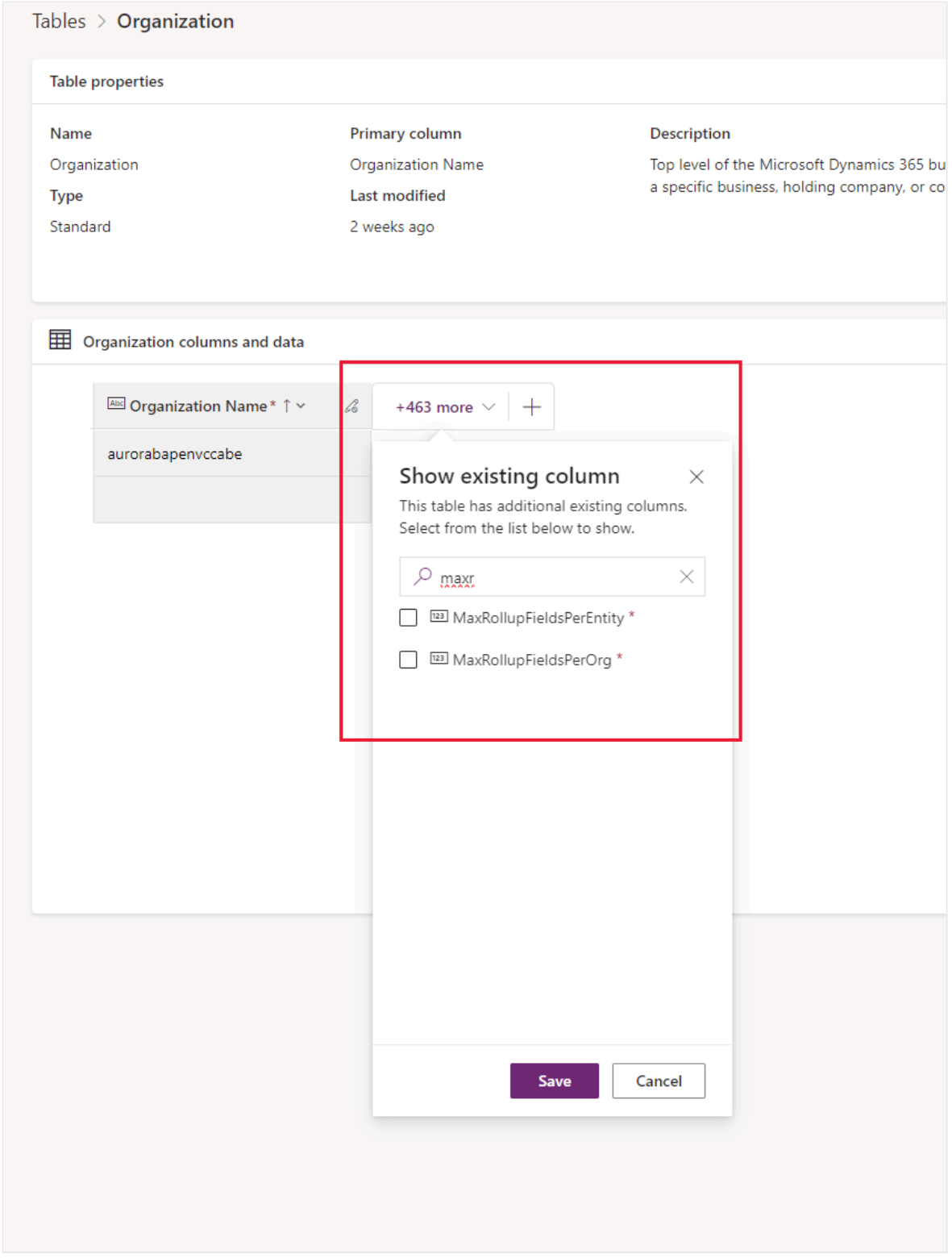
Define the maximum number of rollup columns in an environment or table

By default, you can define a maximum of 200 rollup columns for the environment and up to 50 rollup columns per table.

To change the maximums, follow these steps:

1. Go to Power Apps (make.powerapps.com), select **Table** on the left navigation pane, and then open the **Organization** table.

2. Select the *n* **more** list, and then type *maxr* in the **Search** box. Select the **MaxRollupFieldsPerOrg** and **MaxRollupFieldsPerEntity** columns, and then select **Save**.



3. Change the values for **MaxRollupFieldsPerOrg** (up to 200) and **MaxRollupFieldsPerEntity** (up to 50) that you want.

> **Important**

Having more than 100 rollup columns for an environment might result in degraded performance of rollup columns and increased storage consumption.

Precision rounding

If the precision of the aggregated column is greater than the precision of the rollup column, the aggregated column precision is rounded down to the precision of the rollup column, before the aggregation is performed. To illustrate this behavior, let's look at a specific example. Let's say that the rollup column on the account table, for calculating the total estimated revenue of the related opportunities, has a precision of two decimal points. The Est. Revenue column on the opportunity table is the aggregated column with the precision of four decimal points. In our example, the account has two related opportunities. The aggregated sum of the estimated revenue is calculated as follows:

1. Est. Revenue for the first opportunity: $1000.0041
2. Est. Revenue for the second opportunity: $2000.0044
3. Aggregated sum of Est. Revenue: $1000.00 + $2000.00 = $3000.00

As you can see, the precision rounding to two decimal points on the aggregated column is done before the aggregation is performed.

Different behavior from Associated grids

Certain table forms, such as Account or Contact, out-of-the-box, contain the associated grids. For example, an Account form includes Contacts, Cases, Opportunities, and other grids. Some of the rows shown in the Account form grids are directly related to the account row; others, indirectly, through the relationships with other rows. In comparison, the rollup column aggregation uses only direct relationships explicitly defined in the rollup column definition. No other relationships are considered. To illustrate the difference in behavior, let's look at the following example.

1. The account A1 has a primary contact, P1. The case C1 is associated with the account A1 (C1.Customer column = A1) and the case C2 is associated with the contact P1 (C2.Customer column = P1).
2. The **Cases** grid on the **Account** form for the A1 row, shows two cases, C1 and C2.
3. The rollup column on the account table, called Total Number of Cases, is used to count the cases associated with the account.
4. In the account rollup column definition, we specify the cases that have the Customer relationship with the account. After aggregation, the Total Number of Cases is equal to 1 (case C1). The case C2 isn't included in the total, as it's directly related to the contact, not to the account, and can't be explicitly defined in the

account rollup column definition. As a result, the total number of cases returned by rollup operation doesn't match the number of cases shown in the **Cases** grid.

See also

Create and edit columns
Define calculated columns
Formula, calculated, and rollup columns using code
Behavior and format of the Date and Time column
Define and query hierarchically related data

Feedback

Was this page helpful? Yes No

Provide product feedback

Map table columns

10/01/2025

You can map columns, also known as attributes, between tables that have a one-to-many or many-to-one table relationship. Column mapping lets you set default values for a row that is created in the context of another row.

Let's say that people want to add a new contact row for a person who is an employee for a specific account. They can do this in two different ways:

- The hard way is people can just navigate in the app to create a new contact row from scratch. But then they need to set the parent account and enter several items of information, such as address and phone information, which are probably the same as the parent account it can be time consuming and introduce opportunities for error.
- The easier way is to start with the account table and, using the **Contacts** subgrid on the form, and then select + to add a contact. The interface first guides people to look up any existing related contacts so they don't accidentally create a duplicate row. If they don't find an existing row, they can select **New** and create a new contact row. The new contact row form then includes any of the mapped attribute values from the account, such as address and phone information, as the default values. People can edit these values before they save the row.

When you map table columns for a 1:N table relationship, certain items of data from the primary table row will be copied into the new related table form to set default values that people can edit before saving.

> ⓘ **Note**
>
> - These mappings only set default values to a row before it's saved. People can edit the values before saving. The data that is transferred is the data at that point in time. The data isn't synchronized if the source data later changes.
> - These mappings aren't applied to related rows created using a workflow or dialog process. They aren't automatically applied to new rows created using code, although developers can use a special message called `InitializeFrom` (InitializeFrom Function or InitializeFromRequest Class) to create a new row using available mappings.
> - These mappings aren't applied for new related table forms that are opened when an app doesn't have an active network connection to Microsoft Dataverse, except for parent lookup columns.

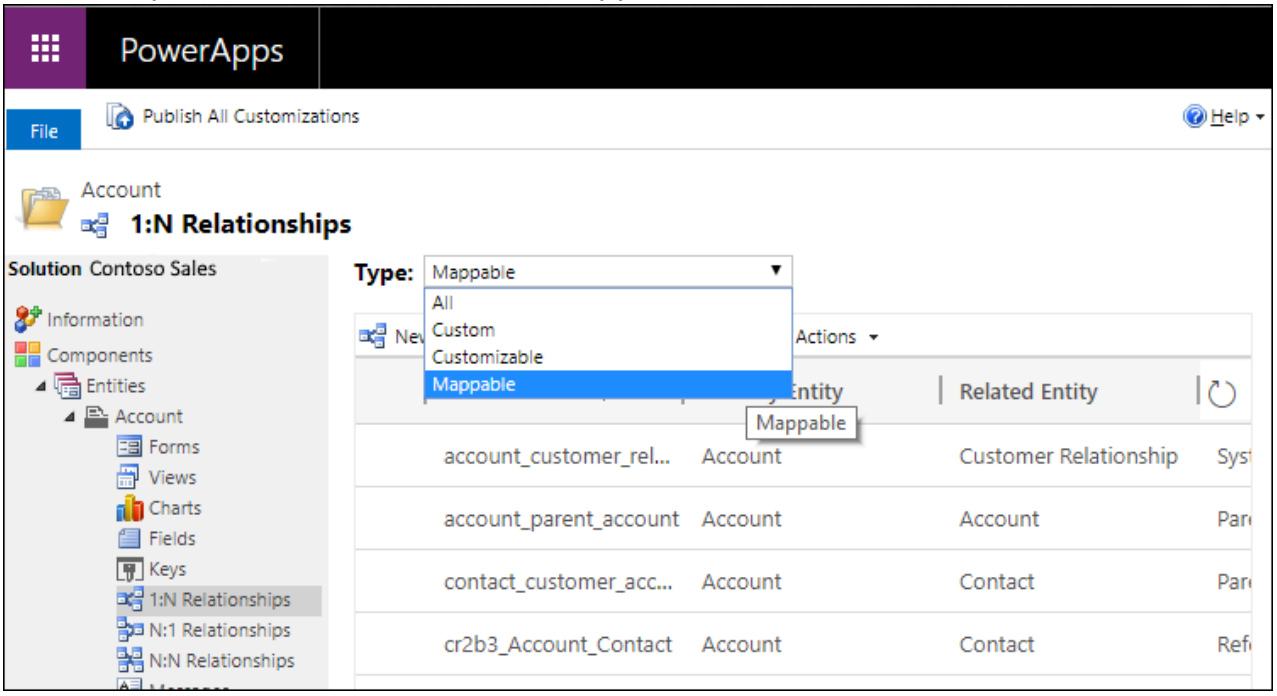
- Column mapping between tables isn't supported when you're using an app in offline mode. For more information, see the limitations listed in Mobile offline limitations for model-driven apps.

View mappable columns

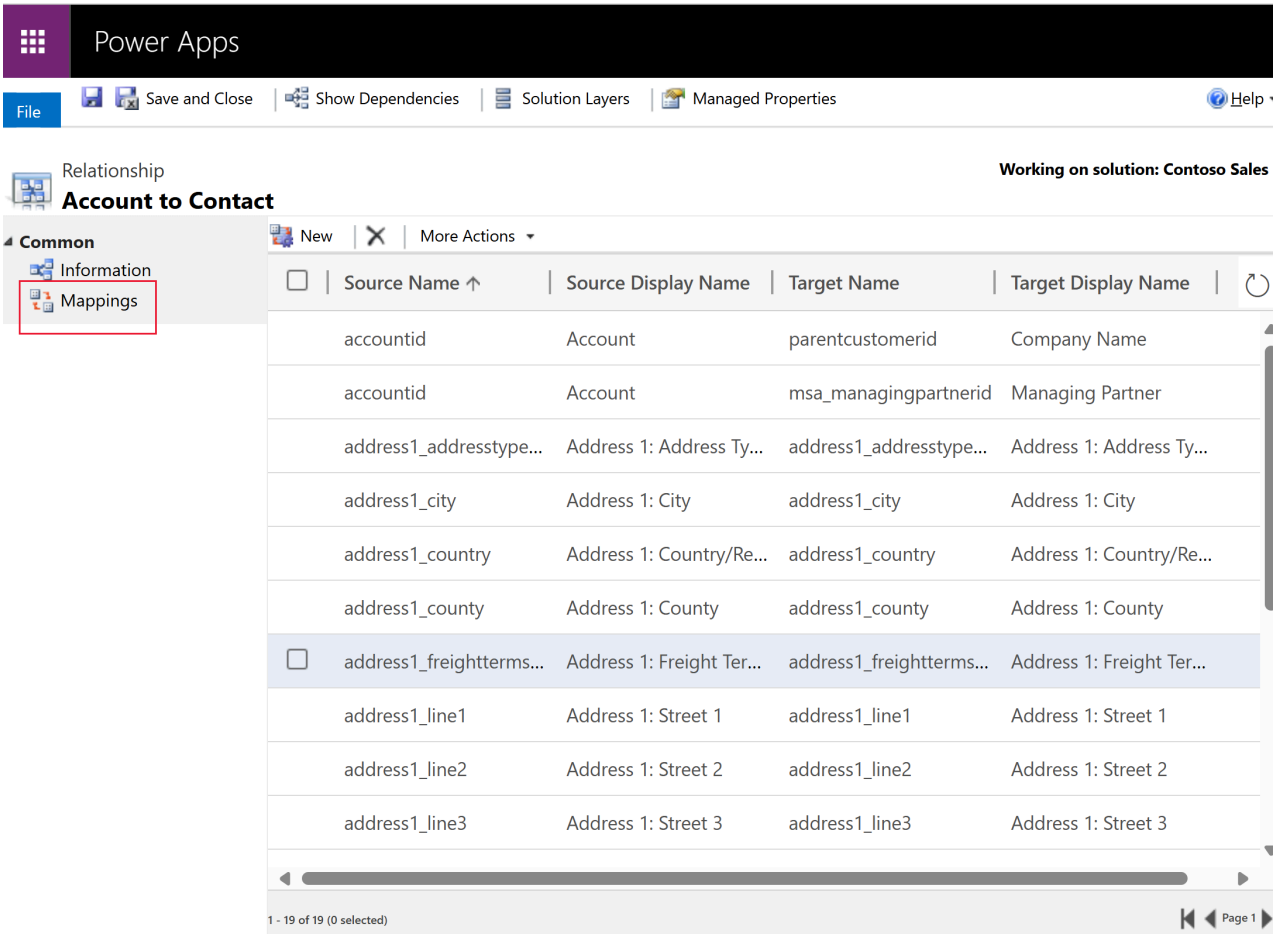
Mapping columns is done in the context of a 1:N or N:1 table relationship, so first you need to view 1:N or N:1 table relationships.

The only way to view and map table columns is to use the classic solution explorer.

1. Sign in to Power Apps, and then select the environment you want.
2. Select **Solutions** on the left pane and then open the solution you want.
3. Select **Switch to classic** on the command bar. The solution opens in the classic solution explorer.
4. Expand **Entities**, expand the table you want, and then expand the relationship type, either **1:N Relationships** or **N:1 Relationships**, and then open the relationship where you want to view or edit column mappings.
5. Column mappings aren't actually defined within the table relationships, but they're exposed in the classic solution explorer relationship user interface. Not every 1:N table relationship has them. In the **Type** dropdown list, select **Mappable** to display only relationships with columns that can be mapped.

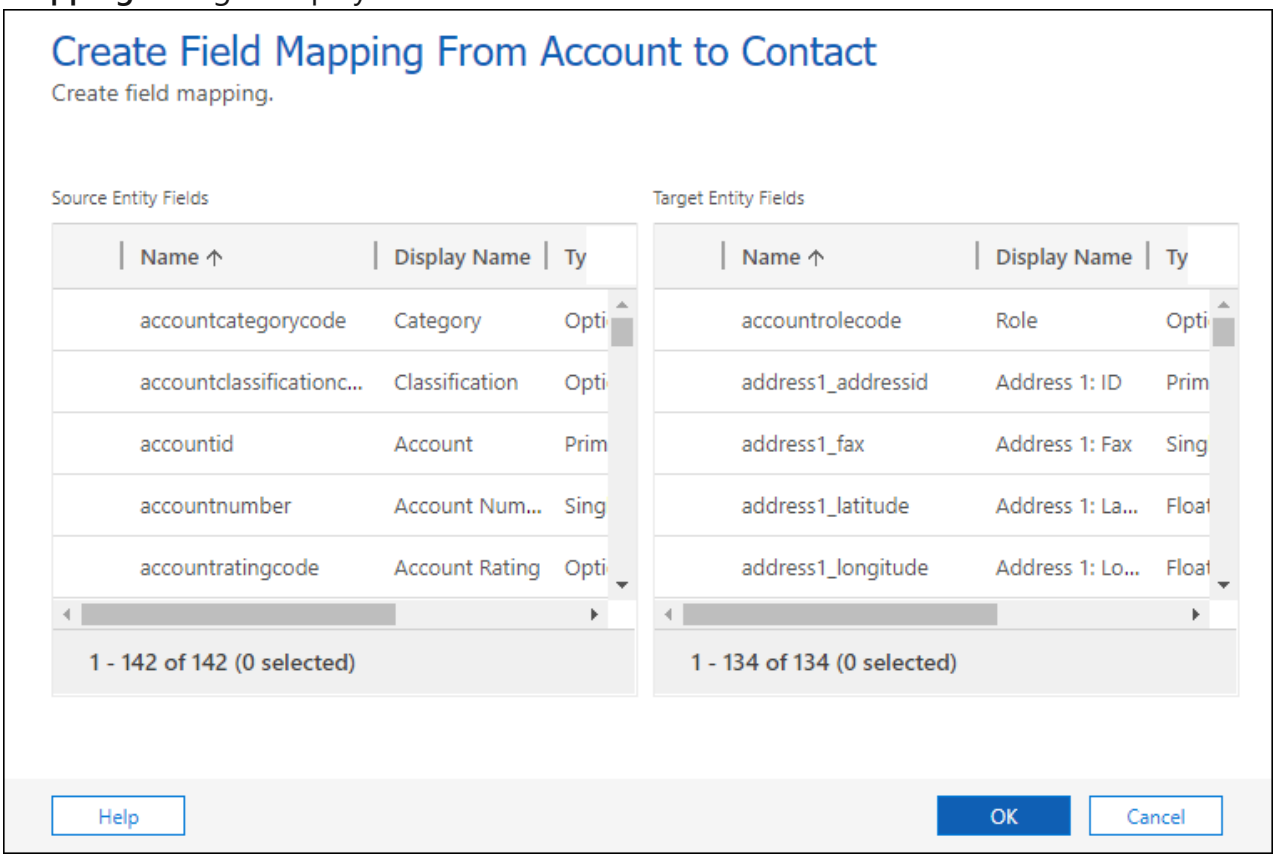


6. Double-click the relationship where you want to view or edit column mappings. A browser page opens to display the relationship properties.
7. Select the **Mappings** tab on the left navigation. The mappings for the relationship are displayed.

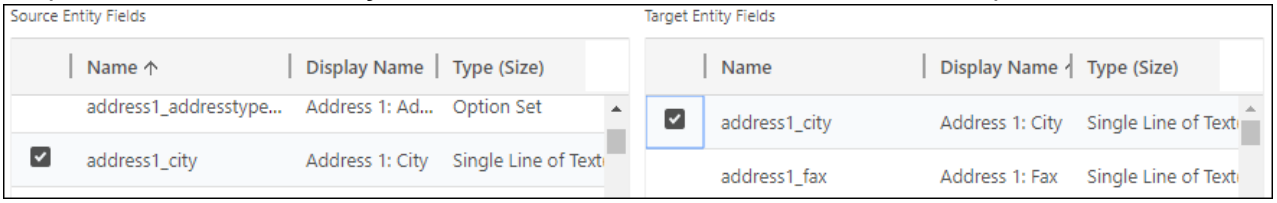


Add new mappings

1. While viewing mappable columns, select **New** on the command bar. The **Create Field Mapping** dialog is displayed.



2. Select one source table column and one target table columns with values you want to map, such as **address1_city** for the account to contact table relationship.



3. Select **OK** to close the dialog.
4. Because column mappings aren't metadata, you must publish them before changes take effect.

Automatically generate column mappings

You can also generate mappings automatically by selecting **Generate Mappings** from the **More Actions** menu.

Use **Generate Mappings** when you create custom tables and want to leverage mapping. However, use care when generating mappings automatically with system tables as you might introduce problems because you're replacing the default mappings.

> ⚠ **Warning**
>
> Generating mappings automatically using **Generate Mappings** removes any existing mappings and replaces them with suggested mappings that are based only on the columns that have similar names and data types. If you use this on a system table, you could lose some expected mappings. For custom tables, it helps save time because you can more easily delete any mappings you don't want and add any others that the generate mappings action didn't create.

Kinds of data and rules for mapping

The following rules show what kinds of data can be mapped:

- Both columns must be of the same type and the same format.
- The length of the target column must be equal to or greater than the length of the source column.
- The target column can't be mapped to another column already.
- The source column must be visible on the form.
- The target column must be a column that a user can enter data into.
- Address ID values or columns of type `partylist` can't be mapped.

- If you map to or from a column that isn't displayed on a form, the mapping won't be done until the column is added to a form.
- If the columns are choices, the integer values for each option should be identical.

> **Note**
>
> If you need to map choice columns, we recommend you configure both columns to use the same global choice. Otherwise, it can be difficult to keep two separate sets of options synchronized manually. If the integer values for each option aren't mapped correctly you can introduce problems in your data. More information: Create and edit global choices for Microsoft Dataverse (picklists)

Delete mappings

While viewing mappable columns if there are any *unmanaged* mappings that you don't want in the environment, you can select them and then select **Delete** ✕.

> **Note**
>
> System and managed mappings can't be deleted the same way as unmanaged mappings, which is by using the legacy solution explorer.

System mappings

You can't delete system generated mappings. When you try to delete a system generated mapping, an error like this one is shown: "Cannot create or delete system attribute map having id <GUID> from '<name>' to '<name>' belonging to an entity map with id <GUID> from '<name>' to 'X<name>."

To remove a system generated mapping, delete the relationship or the lookup column associated with the mapping, which also deletes the mapping. More information: Delete relationships

Managed mappings

You can't delete a mapping if the mapping is managed. To remove the mapping, you must do an upgrade to the solution that brought this mapping in the target environment. To do this, delete the unmanaged mapping in your dev environment as an update to the solution. Export

the solution as managed to import it into your downstream environments. More information: Upgrade or update a solution

See also

Create and edit 1:N (one-to-many) or N:1 (many-to-one) table relationships using solution explorer
Developer Documentation: Customize table and column mappings
Developer Documentation: Create a record from another record using Web API
Developer Documentation: Set default values from the primary table row

Delete columns

Article • 02/15/2022

As someone with the system administrator security role, you can delete any custom columns that aren't part of a managed solution. When you delete a column, any data stored in the column is lost. The only way to recover data from a column that was deleted is to restore the database from a point before the column was deleted.

Before you can delete a custom table, you must remove any dependencies that may exist in other solution components.

1. In the **Solutions** area of Power Apps, open the solution that includes the column that you want to delete.
2. Open the table, select the **Column** tab, and then select the column you want to delete.
3. Select **Show Dependencies** on the command bar to view the **Dependent components** page.

For example, if the column is used in a form or view, you must first remove the column from those solution components.

If you delete a lookup column, the 1:N table relationship for it will automatically be deleted.

See also

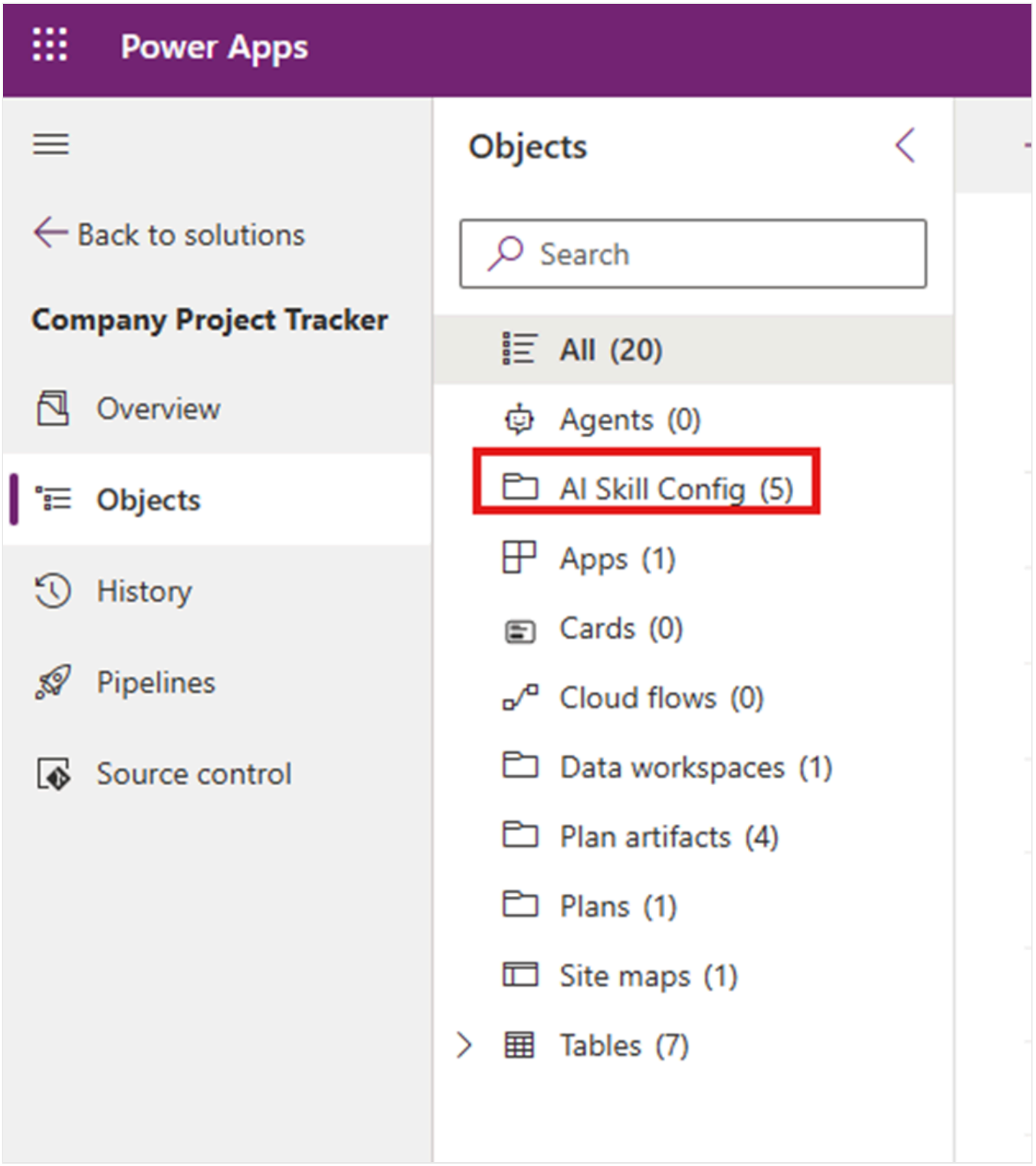
Delete a custom table

Delete a column with an AI Skill Config table dependency in Power Apps

08/14/2025

The AI Skill Config table is a configuration table used to manage AI-related features, such as form fill assistance and row summary. The AI Skill Config table acts as a centralized store for AI feature settings, which helps avoid schema changes for each new feature and supports application lifecycle management (ALM) to ensure smooth transitions across environments.

To locate the AI Skill Config table go to **Solutions** on the left navigation pane, then open the desired solution. If any AI-related features have been customized in the solution, the AI skill config table appears in the **Objects** pane on the left.



Remove an AI Skill Config table record dependency for a column

When enabled, users receive suggestions generated by AI for columns that support form fill assistance in forms and grids. When a user opts out of AI features or for example when a maker disables the **Allow form fill assistance** column property on a table column, a record is created in the AI Skill Config table to store that configuration. This record creates a dependency on the AI Skill Config table for the column.

To find whether the **Allow form fill assistance** setting is disabled for a column, open the desired table, select **Columns** in the **Schema** area, and then open the column. The setting is found in the properties pane.

Edit column

Previously called fields. Learn more

Display name *

Text column

Description

Data type *

Abc Single line of text

Format *

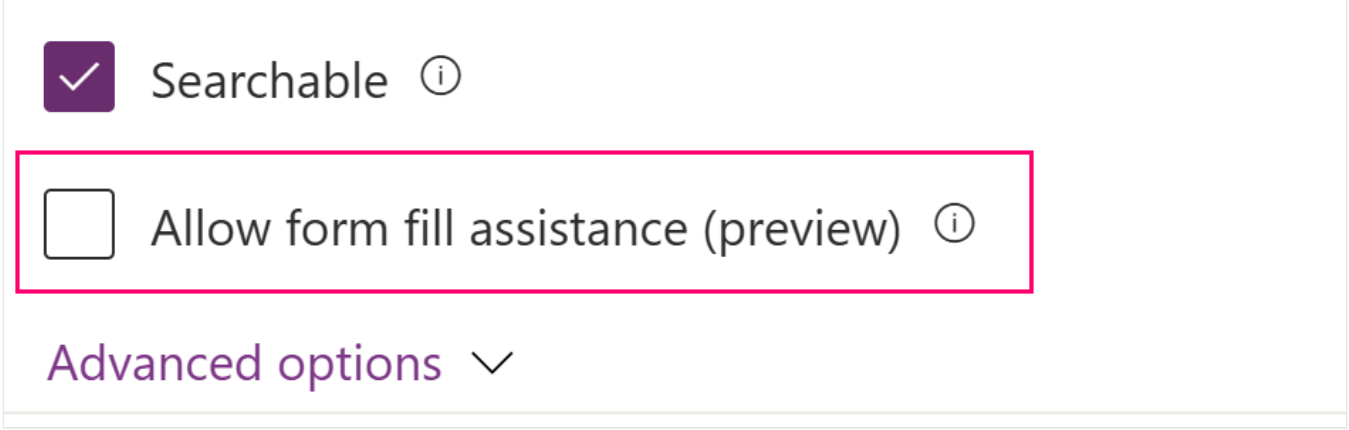
Abc Text

Behavior

Simple

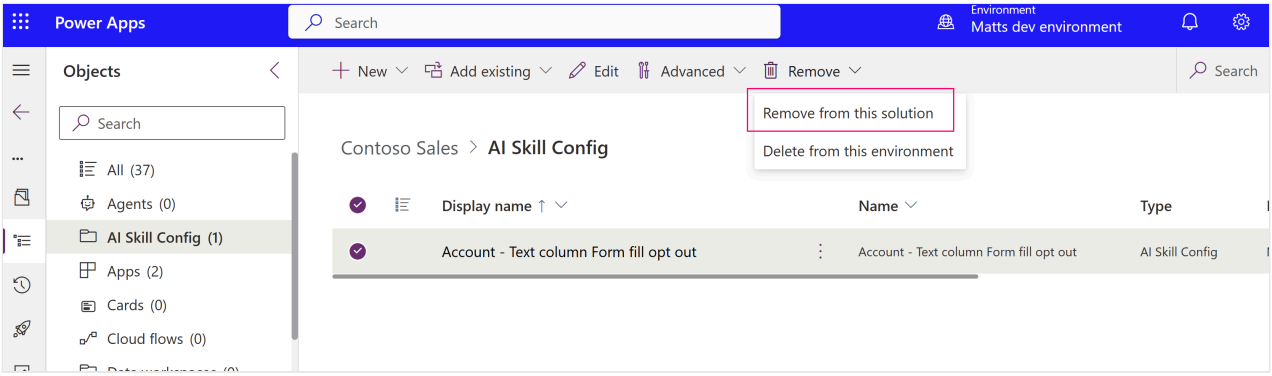
Required

Optional



To delete the column, you must first delete the record created in the config table, otherwise you receive the message "Failed to delete Fields (column name). Object dependencies exist, please review before deleting."

1. Go to **Solutions** on the left navigation pane, then open the desired solution.
2. Select the AI Skill Config table from the **Objects** pane on the left.
3. In the list of records in the AI Skill Config table, find the record that you want to delete. The record has a **Display name** that corresponds to the table **Name** plus column **Display name** for the column you want to delete.
4. Select **Remove** > **Remove from this solution**.



Related articles

Use Copilot's form fill assistance feature in model-driven apps.

Configure a row summary for a model-driven app main form.

Apply business logic in Microsoft Dataverse

Article • 02/14/2024

There are several choices available for applying business logic in Dataverse.

Business rules

You can create business rules and recommendations to apply logic and validations without writing code or creating plug-ins. Business rules provide a simple interface to implement and maintain fast-changing and commonly used rules.

Define *business rules* for a table that apply to all the table forms and at the server level. Business rules defined for a table apply to both *canvas apps* and *model-driven apps* if the table is used in the app. More information: Create a business rule for a table

> **Note**
>
> To define a business rule that applies to a form in a model-driven app, see **Create business rules for a model-driven app form**

Dataverse processes

You can also use the following Dataverse processes:

- Real-time workflows
- Actions

Power Automate

Power Automate has several different flows you can use to create automated workflows within Dataverse or between Dataverse and another app or service that can be used to synchronize files, get notifications, collect data, and more.

Expand table

Flow type	Description	More information
Business process flows	Define a set of steps that run in a Power Apps model-driven app for people to follow to take them to a desired outcome.	Learn more
Cloud flows	Create a flow that performs one or more tasks automatically after it's triggered by an event.	Learn more
Instant flows	Run repetitive tasks from anyplace, at any time, via a flow app on your mobile device.	Learn more
Scheduled flows	Create a flow that performs one or more tasks on a schedule.	Learn more
Desktop flows	Row and automate the playback of manual steps on legacy software.	Learn more

See also

Apply business logic in model-driven apps

Create a business rule for a table

You can create business rules and recommendations to apply logic and validations without writing code or creating plug-ins. Business rules provide a simple interface to implement and maintain fast-changing and commonly used rules.

> **Important**
>
> Business rules defined for a table apply to both *canvas apps* and *model-driven apps* if the table is used in the app. Not all business rule actions are available on canvas apps at this time. More information: Differences between canvas and model-driven apps
>
> In a model-driven app, not all business rule actions are available for editable grids. For table based view pages, recommendations can't be created. Editable subgrids don't support business rules. Business rules won't work with other types of dataset controls.
>
> To define a business rule that applies to a form in a model-driven app, see Create business rules to apply logic in a model-driven app form.

By combining conditions and actions, you can do any of the following with business rules:

- Set column values
- Clear column values
- Set column requirement levels
- Show or hide columns
- Enable or disable columns
- Validate data and show error messages
- Create business recommendations based on business intelligence.

Differences between canvas and model-driven apps

Forms in model-driven apps can use all actions available on business rules, however not all business rule actions are available for canvas apps at this time. The following actions are *not* available on canvas apps:

- Show or hide columns
- Enable or disable columns

- Create business recommendations based on business intelligence

Column type support with business rules

Business rules work with most column types including text, number, choice, date, lookup, owner, and image. However, business rules don't work with the following column types:

- Choices (multi-select)
- File
- Language

Create a business rule

1. Sign in to Power Apps, on the left navigation pane, and select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Open the table you want to create the business rule for (for example, open the **Account** table), and then select **Business rules**.

3. Select **Add business rule**.

   The business rule designer window opens with a single condition already created for you. Every rule starts with a condition. The business rule takes one or more actions based on that condition.

   > **Tip**
   >
   > If you want to modify an existing business rule, you must deactivate it before you can modify it.

4. Add a description, if you want, in the description box in the upper-left corner of the window.

5. Set the scope, according to the following:

Expand table

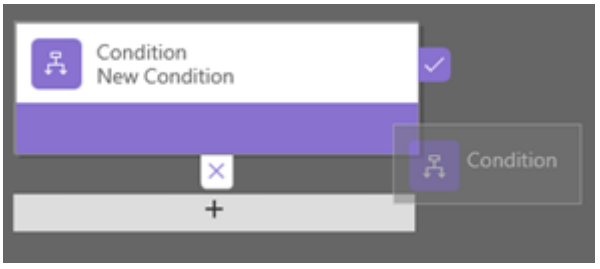
If you select this item...	The scope is set to...
Entity	Model-driven app forms and server

If you select this item...	The scope is set to...
All Forms	Model-driven app forms
Specific form (**Account** form, for example)	Just that model-driven app form

> **Tip**
>
> If you're building a Canvas app, you must use table as the scope.

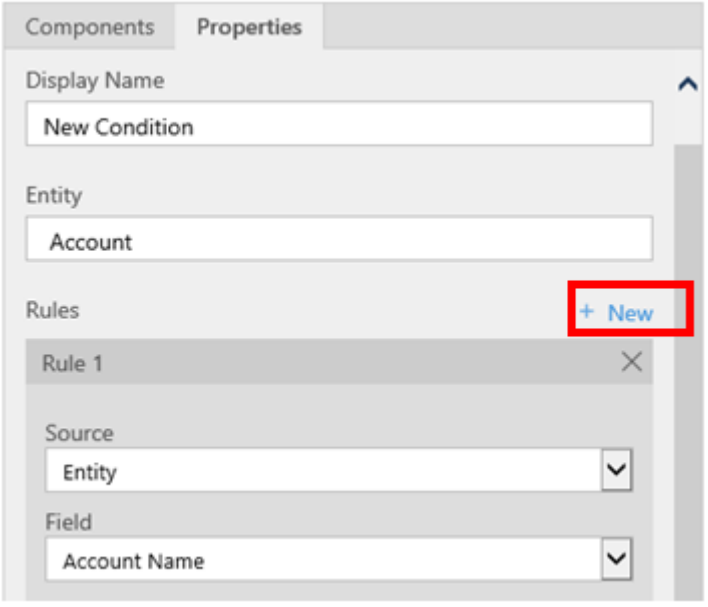
6. **Add conditions.** To add more conditions to your business rule:

   a. Drag the **Condition** component from the **Components** tab to a plus sign in the designer.



   b. To set properties for the condition, select the **Condition** component in the designer window, and then set the properties in the **Properties** tab on the right side of the screen. As you set properties, the Microsoft Dataverse creates an expression at the bottom of the **Properties** tab.

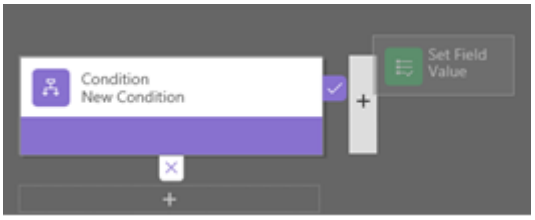
   c. To add an additional clause (an AND or OR) to the condition, select **New** in the **Properties** tab to create a new rule, and then set the properties for that rule. In the **Rule Logic** column, you can specify whether to add the new rule as an AND or an OR.



d. When you're done setting properties for the condition, select **Apply**.

7. **Add actions.** To add an action:

   a. Drag one of the action components from the **Components** tab to a plus sign next to **Condition** component. Drag the action to a plus sign next to a check mark if you want the business rule to take that action when the condition is met, or to a plus sign next to an x if you want the business rule to take that action if the condition isn't met.

   b. To set properties for the action, select the **Action** component in the designer window, and then set the properties in the **Properties** tab.

   c. When you're done setting properties, select **Apply**.

8. **Add a business recommendation (model-driven apps only)**. To add a business recommendation:

   a. Drag the **Recommendation** component from the **Components** tab to a plus sign next to a **Condition** component. Drag the **Recommendation** component to a plus sign next to a check mark if you want the business rule to take that action when the condition is met, or to a plus sign next to an x if you want the business rule to take that action if the condition isn't met.

   b. To set properties for the recommendation, select the **Recommendation** component in the designer window, and then set the properties in the **Properties** tab.

   c. To add more actions to the recommendation, drag them from the **Components** tab, and then set properties for each action in the **Properties** tab.

   > ⓘ **Note**
   >
   > When you create a recommendation, the Dataverse adds a single action by default. To see all the actions in a recommendation, select **Details** on the **Recommendation** component.

   d. When you're done setting properties, select **Apply**.

9. To validate the business rule, select **Validate** on the action bar.
10. To save the business rule, select **Save** on the action bar.
11. To activate the business rule, select it in the Solution Explorer window, and then select **Activate**. You can't activate the business rule from the designer window.

> **Tip**
>
> Here are a few tips to keep in mind as you work on business rules in the designer window:
>
> - To take a snapshot of everything in the **Business Rule** window, select **Snapshot** on the action bar. This is useful, for example, if you want to share and get comments on the business rule from a team member.
> - Use the mini-map to navigate quickly to different parts of the process. This is useful when you have a complicated process that scrolls off the screen.
> - As you add conditions, Actions, and business recommendations to your business rule, Dataverse builds the code for the business rule at the bottom of the designer window. This code is read only.

Example: Create a business rule for tasks older than 30 days

This business rule example creates a condition that triggers a message in the task description field when a task is more than 30 days old.

1. Sign in to Power Apps, on the left navigation pane, select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the **Task** table, and then select the **Business rules** area.
3. Select **Add business rule**.
4. Select **New Condition** on the business process flow canvas, and enter or select the following properties:

    - **Display name**: *Task is more than 30 days old*
    - **Entity**: **Task**
    - **Rule 1**
        - **Source**: **Entity**
        - **Field**: **Created On**

    - **Operator**: +
    - **Type**: **Value**
    - **Days**: *30*
  - **Condition Expression** (automatically created): **(Created On Is greater than [Created On + 30])**
5. Select **Apply**.
6. Select **Add** > **Add Show Error Message**.
7. In the **Show Error Message Properties** tab, enter the following properties:

   - **Display Name**: *Task is more than 30 days old*
   - **Entity**: **Task**
   - **Error Message**:
     - **Field**: **Description**
     - **Message**: *This task is more than 30 days old!*

8. Select **Apply**.
9. Select **Save**.

Localize error messages used in business rules

If you have more than one language provisioned for your organization, you'll want to localize any error messages that you have set. Each time you set a message, a label is generated by the system. If you export the translations in your organization, you can add localized versions of your messages and then import those labels back into the Dataverse, so that people using languages other than your base language can view the translated messages.

Common issues

This section describes common issues that might occur when you use business rules.

Composite attributes not supported with Unified Interface apps

Actions or conditions that use Composite attributes aren't supported in apps based on the Unified Interface. Alternatively, you can use actions or conditions on the attributes that comprise the composite attributes. For example, instead of using the **Full Name** (fullname) attribute, you can use the **First Name** (firstname) and **Last Name** (lastname) attributes.

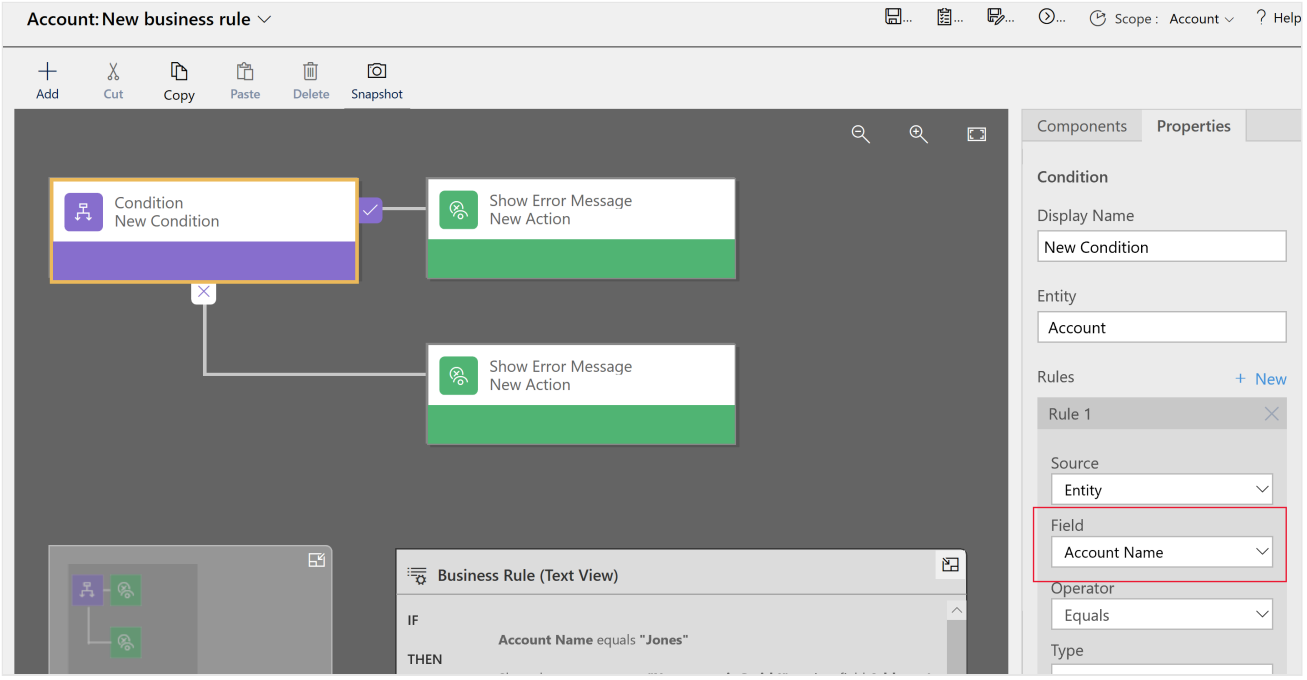
A large number of business rules on a single table impacts performance

New or existing business rules on a table can cause performance degradation when the rules are activated. Power Platform today supports up to 150 business rules for a single table. Beyond 150 business rules, performance degradation can be experienced. This limit includes both client side (JavaScript) and server side (XAML generated as synchronous plugins) business rules. To avoid performance issues with Dataverse, we recommend that you don't create more than 150 business rules for a single table.

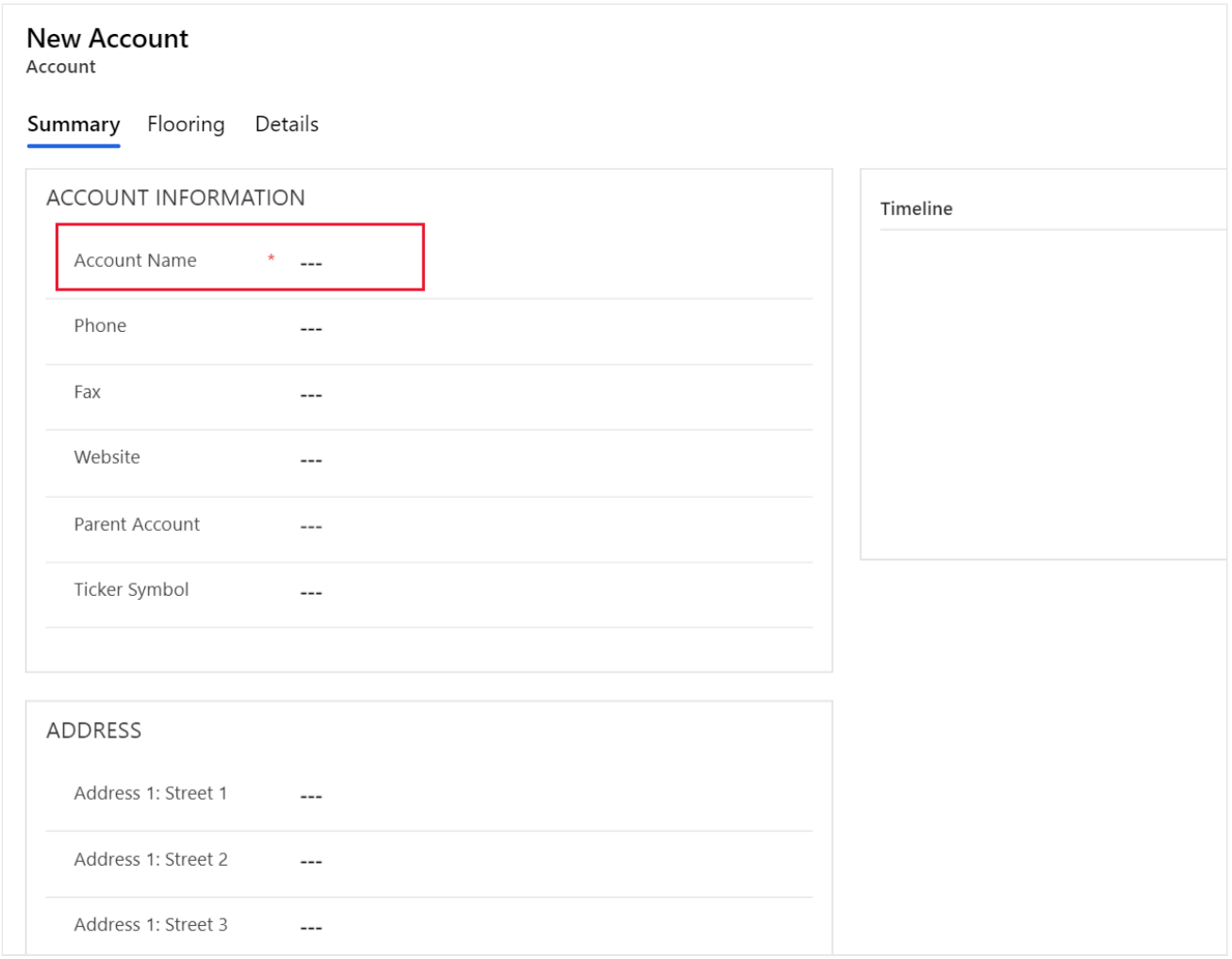
Is your business rule not firing for a form?

A business rule might not execute because the field referenced in the business rule isn't included with the form.

1. Open solution explorer. Expand the entity that you want and then select **Forms**.
2. Open the form that you want and then on the form designer ribbon select **Business Rules**.
3. In the form designer, open the business rule.
4. In the business rule designer, select each condition and action to verify all the fields referenced in each condition and action.



5. Verify that each field referenced in the business rule is also included on the form. If not, add the missing field to the form.



UTC time zone for DateOnly columns

When you configure business rules with `DateOnly` columns, the dates are in the UTC time zone by default, irrespective of the user's set time zone. This can lead to unexpected results if you're expecting `DateOnly` values to align with local time settings.

Suggested handling: Time Zone Independent setting - To make the `DateOnly` columns independent of time zones and display dates in parity with what was configured in the business rule, UTC for example, you can enable the Time Zone Independent setting. For more information about configuring date handling settings, go to Specify the behavior of a date and time column.

Frequently asked questions (FAQ)

Can business rules unlock fields on a read-only form?

- Yes, a business rule can unlock fields and edit actions on a read-only form.

How do I troubleshoot a business rule that isn't working?

- See Is your business rule not firing for a form? in this article.

Do business rules react to changes made by an onLoad script?

- No, they'll execute before an onload script is executed.

When I update a business rule, is it executed against all existing records?

- No. Business rules are run on clients. For example, they run when a form is opened by a user and when a field value changes on that open form. They aren't executed inside Dataverse.

See also

Apply business logic in Microsoft Dataverse

Last updated on 04/21/2026

Microsoft Dataverse real-time workflows

Article • 07/12/2024

Workflows automate business processes without a user interface. People usually use workflow processes to initiate automation that doesn't require any user interaction.

There are two types of workflows:

- Background workflows. Background workflows run when the system has resource availability (asynchronously). Go to the Power Automate documentation for more details about background workflows.
- Real-time workflows. Real-time workflows run immediately (synchronously).

Create or edit real-time workflows

> **Important**
>
> There are better ways to create modern automations. Consider using Power Automate flows to automate your processes. More information: **Power Automate**

1. Create a solution or open an existing one.
2. Select **New** > **Automation** > **Process** > **Workflow**.
3. Enter a **Display name**, select a **Table**, and then clear the **Run workflow in the background** option.
4. Select **Create**.

The workflow designer is displayed. Select the workflow attributes and steps needed. More information: Configure real-time workflow stages and steps

View and edit real-time workflows

1. Sign in to Power Apps, select the **Settings** icon on the upper right, and then select **Advanced settings**.
2. Select the down arrow next to **Settings** on the top bar, and then select **Processes**.
3. In the list of processes, select the workflow that you want to edit.

Learn more

- Monitor and manage real-time workflow processes
- Best practices for real-time workflow processes
- Use actions

Feedback

Was this page helpful? Yes No

Provide product feedback

Configure real-time workflow stages and steps

When you design workflows, you have the option to contain the logic you want to perform in stages and steps.

- Stages. Stages make the real-time workflow logic easier to read, and explain the real-time workflow logic. However, stages don't affect the logic or behavior of workflows. If a process has stages, all the steps within the process must be contained with a stage.
- Steps. Steps are a unit of business logic within a workflow. Steps can include conditions, actions, other steps, or a combination of these elements.

Actions that real-time workflow processes can perform

Real-time workflow processes can perform the actions listed in the following table.

Expand table

Action	Description
Create Row	Creates a new row for a table and assigns values that you choose to attributes.
Update Row	You can update the row that the real-time workflow is running on, any of the rows linked to that row in an N:1 relationship, or any rows created by earlier steps.
Assign Row	You can assign the row that the real-time workflow is running on, any of the rows linked to that row in an N:1 relationship, or any rows created by earlier steps.
Send Email	Sends an email. You can choose to create a new email message or use an email template configured for the table of the row that the real-time workflow is running on or any tables that have an N:1 relationship with the table, or the table for any rows created by earlier steps.
Start Child Workflow	Starts a real-time workflow process that is configured as a child workflow.
Change Status	Changes the status of the row that the process is running on, any of the rows linked to that row in an N:1 relationship, or any rows created by earlier steps.
Stop Workflow	Stops the current workflow. You can set a status of either **Succeeded** or **Canceled** and specify a status message. When real-time workflows are configured for an event, stopping a real-time workflow with a

Action	Description
	status of canceled will prevent the event action from completing. Go to Using real-time workflows for more information.
Custom Step	Developers can create custom real-time workflow steps that define actions. No custom steps are available by default.

Setting row values

When you create a row, you can set values for the row. When you update a row, you can set, append, increment, decrement, multiply, or clear values.

When you select **Set Properties**, a dialog box opens showing you the default form for the table.

At the bottom of the dialog box you see a list of additional columns not present in the form.

For any column, you can set a static value that is set by the workflow.

On the right side of the dialog box, the **Form Assistant** gives you the ability to set or append dynamic values from the context of the current row. This includes values from related rows that can be accessed from the N:1 (many-to-one) relationships for the table.

The options available in the **Form Assistant** depend on the column you have selected in the form. When you set a dynamic value, you see a yellow placeholder known as a "slug" that shows where the dynamic data is included. If you want to remove the value, select the slug and delete it. For text columns, you can use a combination of static and dynamic data.

With dynamic values, you don't know for certain that a column or related table has the value you want to set. You can actually set a number of columns to try to set the value and sort them in order using the green arrows. If the first column doesn't have data, try on the second column and so. If none of the columns have data, you can specify a default value to be used.

Setting conditions for real-time workflow actions

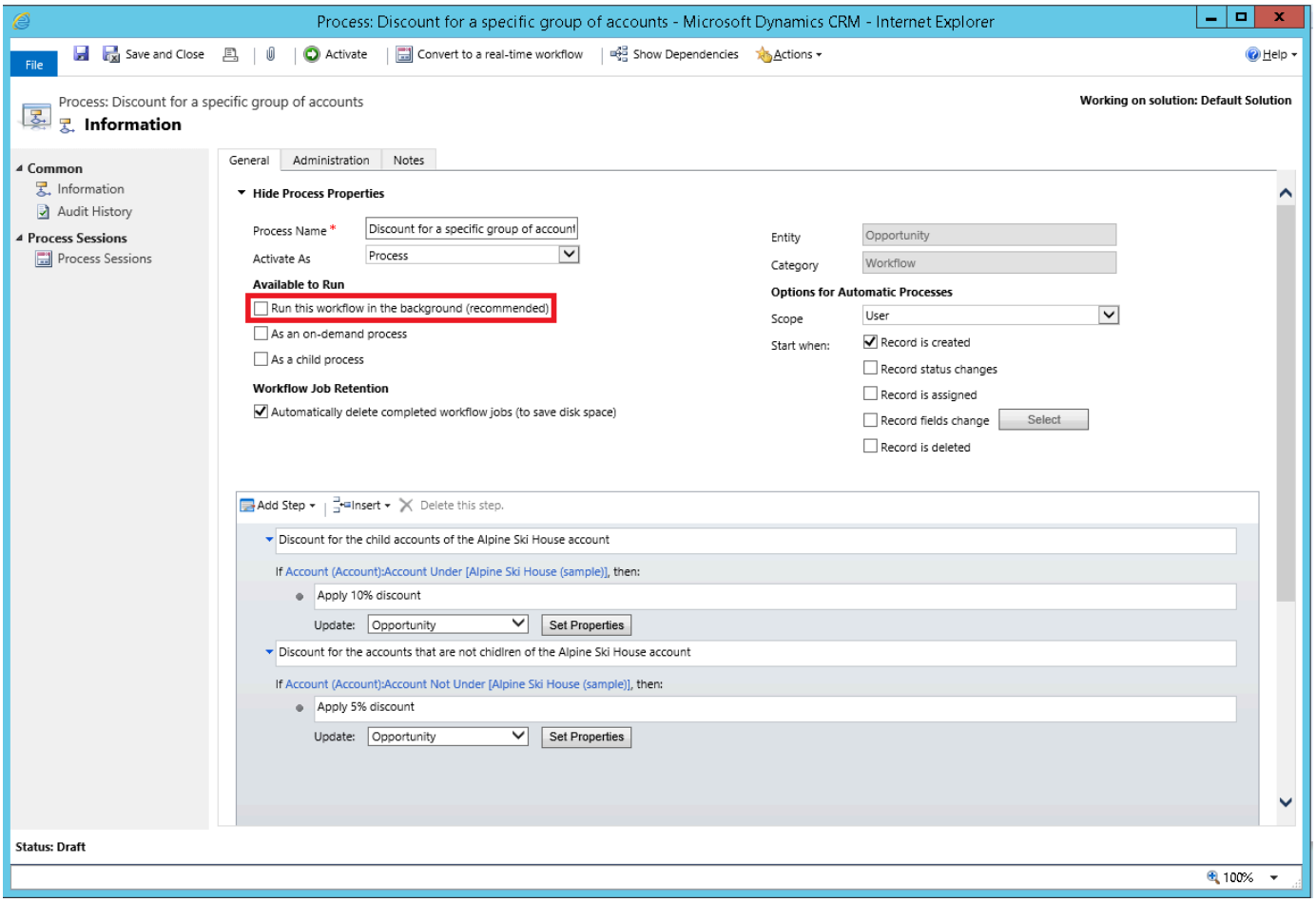
The actions that you apply often depend on conditions. Real-time workflow processes provide several ways to set conditions and create branching logic to get the results you want. You can check values of the row that the real-time workflow process is running against, any of the rows linked to that row with an N:1 relationship, or values within the process itself.

Expand table

Condition Type	Description
Check Condition	A logical "if <condition> then" statement. You can check the current values for the row that the real-time workflow is running on, any of the rows linked to that row in an N:1 relationship, or any rows created by earlier steps. Based on these values, you can define additional steps when the condition is true. In the "if <condition> then" statement, you can use the following operators: **Equals**, **Does Not Equal**, **Contains Data**, **Does Not Contain Data**, **Under** and **Not Under**. **Note:** The **Under** and **Not Under** are hierarchical operators. They can only be used on the tables that have a hierarchical relationship defined. If you're trying to use these operators on the tables that don't have the hierarchical relationship defined, see the error message: "You're using a hierarchical operator on a table that doesn't have a hierarchical relationship defined. Either make the table hierarchical (by marking a relationship as hierarchical) or use a different operator." For more information about hierarchical relationships, go to Define and query hierarchically related data. A screenshot that follows the table is an example of the definition of the real-time workflow process that uses the **Under** and **Not Under** hierarchical operators.
Conditional Branch	A logical "else-if-then" statement, the editor uses the text "Otherwise, if <condition> then:" Select a check condition you previously defined and you can add a conditional branch to define additional steps when the check condition returns false.
Default Action	A logical "else" statement. The editor uses the text "Otherwise:" Select a check condition, conditional branch, wait condition, or parallel wait branch that you previously defined and you can use a default action to define steps for all cases that don't match the criteria defined in condition or branch elements.
Wait Condition	Real-time workflows can't use wait conditions. However, wait conditions can be used with background workflows. More information: Setting conditions for background workflow actions
Parallel Wait Branch	Defines an alternative wait condition for a real-time workflow with a corresponding set of additional steps that are performed only when the initial criterion is met. You can use parallel wait branches to create time limits in your real-time workflow logic. They help prevent the real-time workflow from waiting indefinitely until the criteria defined in a wait condition are met.
Custom Step	Developers can create custom real-time workflow steps that define conditions. There are no custom steps available by default.

The following screenshot contains an example of the workflow process definition with the **Under** and **Not Under** hierarchical operators. In our example, we apply two different discounts to two

groups of accounts. In **Add Step**, we selected the **Check Condition** to specify the **if-then** condition containing the **Under** or **Not Under** operators. The first **if-then** condition applies to all accounts that are **Under** the Alpine Ski House account. These accounts receive a 10% discount on purchased good and services. The second **if-then** condition applies to all accounts that are **Not Under** the Alpine Ski House account and they receive a 5% discount. Then, we selected **Update Row** to define the action to be performed based on the condition.



Initiating real-time workflows before or after status changes

When you configure **Options for Automatic Processes** for real-time workflows, the **Start When** options for the status changes event let you select **After** or **Before** for when status changes. The default option is **After**.

When you select **Before** you're saying that you want the logic in the real-time workflow to be applied before data changing the status is saved. This provides you with the ability to check the values before other logic has been applied after the operation and prevent further logic from being performed. For example, you might have additional logic in a plug-in or custom real-time workflow action which could initiate actions on another system. By stopping further processing, you can avoid cases where external systems are affected. Applying real-time workflows before

this event also means that other real-time workflow or plug-in actions that might save data don't need to be "rolled back" when the operation is canceled.

Expand table

Action	Start when	Explanation
Row is created	After	Only **After** is available. The row won't have a unique identifier until after the internal MainOperation stage so can't occur before the row is created.
Row status changes	Before After	Corresponds to an update operation that provides the ability to apply real-time workflow logic either after or before the status changes. **Before** corresponds to the preoperation stage. **After** corresponds to the postoperation stage.
Row is assigned	Before After	Corresponds to an update operation that provides the ability to apply real-time workflow logic either after or before the status changes. **Before** corresponds to the preoperation stage. **After** corresponds to the postoperation stage.
Row columns change	Before After	Corresponds to an update operation that provides the ability to apply real-time workflow logic either after or before the status changes. **Before** corresponds to the preoperation stage. **After** corresponds to the postoperation stage.
Row is deleted	Before	Only **Before** is available. Row deletion corresponds to the preoperation stage. After the MainOperation occurs, the row is deleted and there's no further status change that can occur.

For more information about preoperation, mainoperation, and postoperation stages, see Event execution pipeline.

Using real-time workflows

You can configure real-time workflows but you should use them with care. Background workflows are generally recommended because they allow the system to apply them when resources on the server are available. This helps smooth out the work the server has to do and helps maintain the best performance for everyone using the system. The drawback is that actions defined by background workflows aren't immediate. You can't predict when they are applied, but generally it takes a few minutes. For most automation of business processes, this is fine because people using the system don't need to be consciously aware that the process is running.

Use real-time workflows when a business process requires someone to immediately see the results of the process or if you want the ability to cancel an operation. For example, you might want to set certain default values for a row the first time it's saved, or you want to make sure that some rows aren't deleted.

Converting between real-time and background workflows

You can change a real-time workflow into a background workflow by choosing **Convert to a background workflow** on the toolbar.

You can change a background workflow into a real-time workflow by choosing **Convert to a real-time workflow** on the toolbar. If the background workflow uses a wait condition, it becomes invalid and you won't be able to activate it until you remove the wait condition.

Initiating real-time workflows before or after status changes

When you configure **Options for Automatic Processes** for real-time workflows, the **Start When** options for the status changes event let you select **After** or **Before** for when status changes. The default option is **After**.

When you select **Before**, you're saying that you want the logic in the real-time workflow to be applied before data changing the status is saved. This provides you with the ability to check the values before other logic are applied after the operation and prevents further logic from being performed. For example, you might have additional logic in a plug-in or custom real-time workflow action that could initiate actions on another system. By stopping further processing, you can avoid cases where external systems are affected. Applying real-time workflows before this event also means that other real-time workflow or plug-in actions that might have saved data don't need to be "rolled back" when the operation is canceled.

Using the Stop Workflow action with real-time workflows

When you apply a **Stop Workflow** action in a real-time workflow, you have the option to specify a status condition that can be either **Succeeded** or **Canceled**. When you set the status to canceled, you prevent the operation. An error message containing the text from the stop action status message will be displayed to the user with the heading **Business Process Error**.

Configuring who runs the real-time workflow

In the workflow designer, you can set who the workflow should run as to either "The owner of the workflow" or "The user who made changes to the record." To change the owner of the record to another user, open the **Administration** tab in the workflow editor and use the lookup to select a new owner for the workflow. You need the System Administrator or System Customizer security role or a role containing the `prvChangeOwnerIdOfWorkflow` privilege to change the owner of the workflow.

Next steps

Monitor and manage real-time workflow processes

Best practices for real-time workflow processes

Last updated on 04/28/2026

Monitor and manage real-time workflow processes

Article • 05/15/2024

To monitor and manage processes, you must locate the process, evaluate the status, and perform any actions necessary to address problems.

Monitoring real-time workflows and actions

Real-time workflows and actions don't use System Job rows because are displayed to the user in the application with the heading **Business Process Error**.

There's no log for successful operations. You can enable logging for errors by checking the **Keep Logs for workflow jobs that encountered errors** option in the **Workflow Log Retention** area at the bottom of the **Administration** tab for the process.

To view the log of errors for a specific process, open the real-time workflow or action definition and go to the **Process Session** tab. This log only shows errors logged for this process.

If you want a view of all the errors for any process, go to **Advanced Find** and create a view showing errors on the process session table.

Status of real-time workflow processes

When you view a list of real-time workflow processes, any individual process can have one of the following **State** and **Status Reason** values:

Expand table

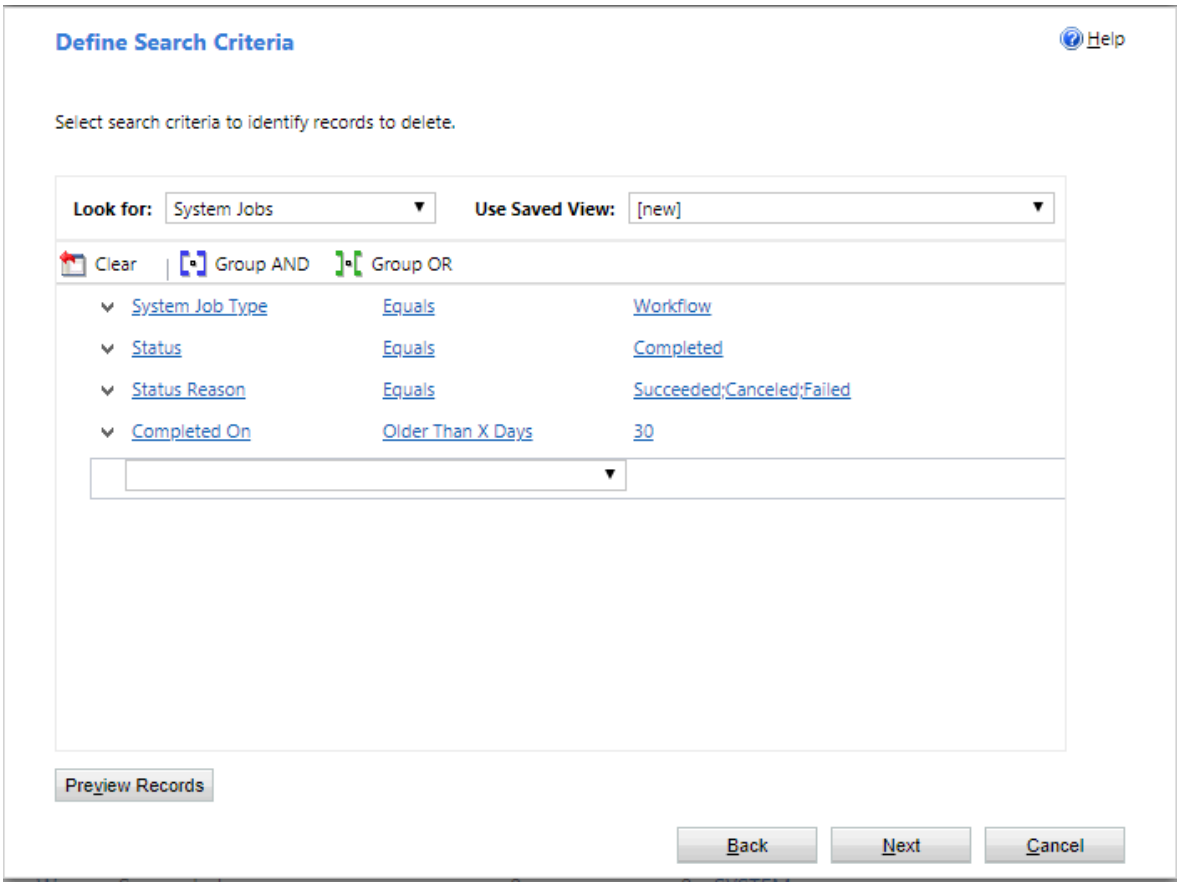
State	Status Reason
Ready	Waiting for Resources
Suspended	Waiting
Locked	In Progress Pausing Canceling

State	Status Reason
Completed	Succeeded
	Failed
	Canceled

Deleting process log rows

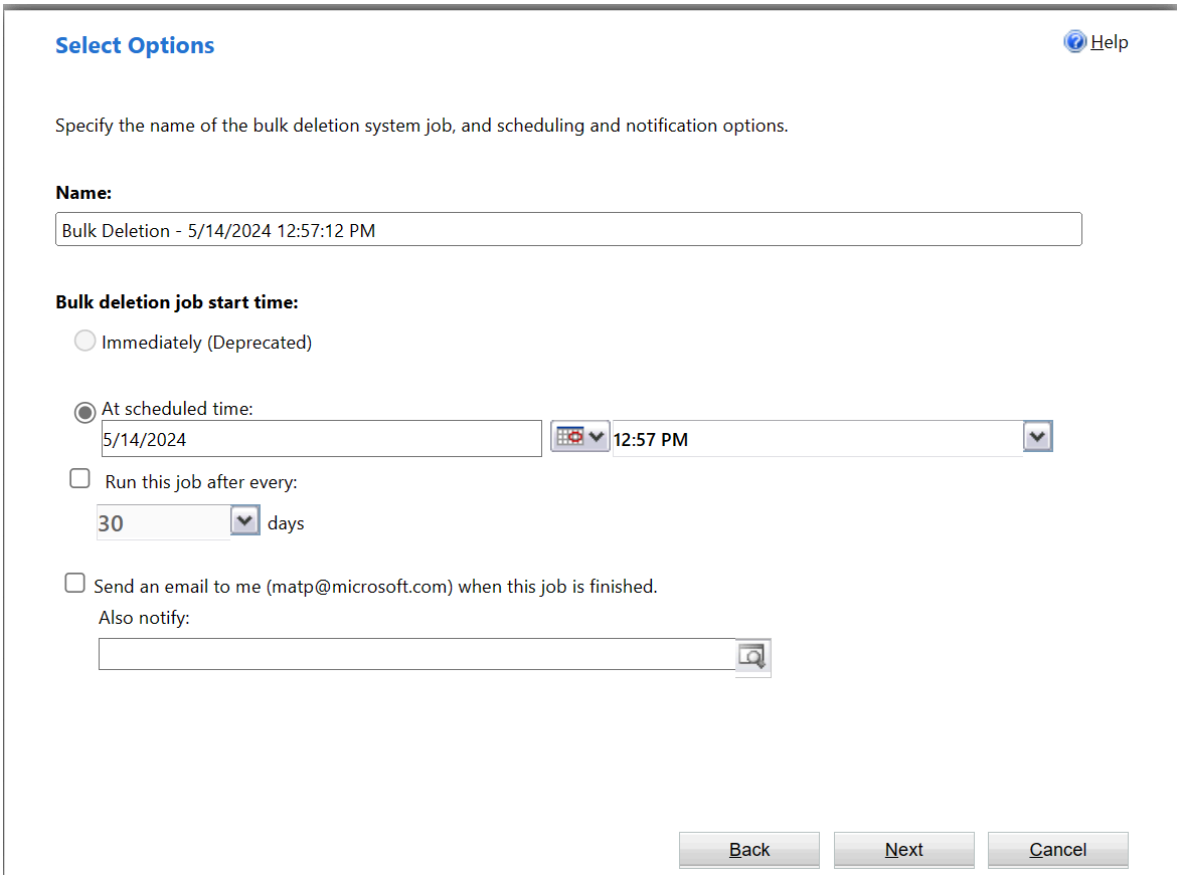
If your organization uses background workflows or business process flows that run frequently, the amount of process log rows can become large enough to cause performance issues as well as consume significant amounts of storage. To delete process log rows not removed sufficiently by one of the standard bulk row deletion jobs, you can use the bulk delete system jobs feature to create a custom bulk row deletion job.

1. Go to the Power Platform admin center, and open the environment you want.
2. Go to **Settings** > **Data Management** > **Bulk deletion**.
3. From the **Bulk Record Deletion** area, select **New**.
4. On the **Bulk Deletion Wizard** start page, select **Next**.
5. In the **Look for** list, select **System Jobs**.
6. The following conditions are used to create a bulk row deletion job to delete process log rows:
    - **System Job Type Equals Workflow**: This targets real-time workflow rows.
    - **Status Equals Completed**: Only completed workflows are valid to run the job against.
    - **Status Reason Equals Succeeded**: Delete successful, canceled, and failed jobs.
    - **Completed On Older than X Days 30**: Use the Completed On column to only delete real-time workflow process log rows that are older than 30 days.



7. Select **Next**.

8. Set the frequency that your bulk delete job runs. In this example, a recurring job is set to run on May 14, 2024, and every 30 days thereafter.



> **Note**
>
> Performing an immediate synchronous bulk delete of the rows by selecting the **Immediately** option is deprecated and no longer available for use.

9. Select **Next**, and then select **Submit**.

Next steps

Best practices for real-time workflow processes

Feedback

Was this page helpful? Yes No

Provide product feedback

Best practices for real-time workflow processes

Article • 02/15/2022

This article contains best practices for creating and managing real-time workflow processes.

Avoid infinite loops

It's possible to create logic in a real-time workflow that initiates an infinite loop, which consumes server resources and affects performance. The typical situation where an infinite loop might occur is when you have a real-time workflow configured to start when a column is updated and then updates that column in the logic of the workflow. The update action triggers the same real-time workflow that updates the record and triggers the real-time workflow again and again.

The workflows you create include logic to detect and stop infinite loops. If a real-time workflow process is run more than a certain number of times on a specific record in a short period of time, the process fails with the following error: **This workflow job was canceled because the workflow that started it included an infinite loop. Correct the workflow logic and try again**. The limit of times is 16.

Use real-time workflow templates

If you have workflows that are similar and you anticipate creating more workflows that follow the same pattern, save your real-time workflow as a workflow template. This way, the next time you need to create a similar workflow, you can create the real-time workflow using the template and avoid entering all the conditions and actions from scratch.

In the **Create Process** dialog, choose **New process from an existing template (select from list)**.

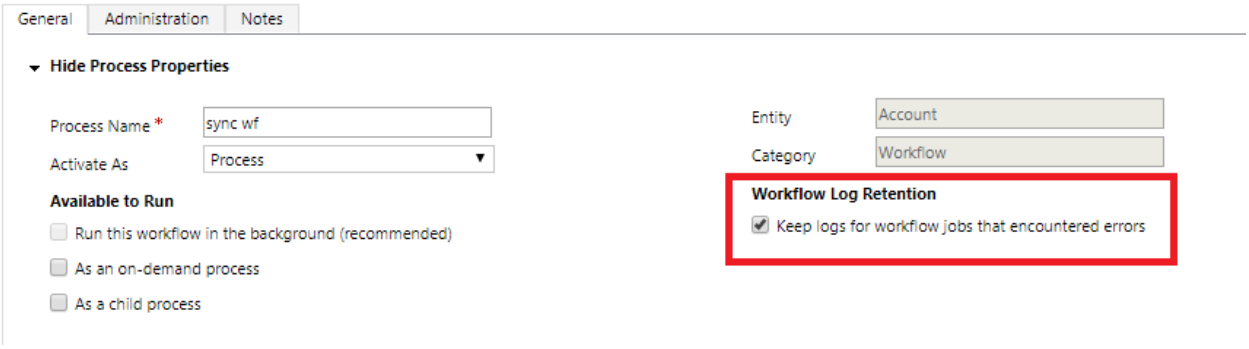
Use child workflows

If you apply the same logic in different workflows or in conditional branches, define that logic as a child real-time workflow so you don't have to replicate that logic manually in each real-time workflow or conditional branch. This helps make your workflows easier to

maintain. Instead of examining many workflows that might apply the same logic, you can just update one workflow.

Keep logs for real-time workflow jobs that encountered errors

For workflows that don't run in the background (synchronous), we recommend selecting the **Keep logs for workflow jobs that encountered errors** option in the real-time workflow definition. Selecting this option allows logs from failed real-time workflow executions to be saved for troubleshooting. Logs from successful synchronous workflow executions will always be deleted to save space.



Limit the number of workflows that update the same table

Running more than one real-time workflow that updates the same table can cause resource lock issues. Imagine several workflows running where every opportunity update triggers an update to the associated account. Multiple instances of these workflows running and attempting to update the same account record at the same time can result in resource locking issues. Real-time workflow failures occur and an error message, such as **SQL Timeout: Cannot obtain lock on resource *resource name***, is recorded.

Use Notes to keep track of changes

When you edit workflows you should use the Notes tab and type what you did and why. This allows someone else to understand the changes you made.

Next steps

Configure real-time workflow processes
Monitor and manage real-time workflow processes

Functions in Microsoft Dataverse (preview)

Article • 05/01/2025

[This topic is pre-release documentation and is subject to change.]

Microsoft Dataverse offers a powerful solution for achieving more efficient data architecture and reducing client-side workload through functions (formerly known as instant low-code plug-ins). Functions in Dataverse use Power Fx to create your business logic. Power Fx is a general-purpose, strong-typed, declarative, and functional programming language. Whereas functions in Dataverse are reusable solution objects, which execute a specific set of commands within Dataverse, running server-side.

> **ⓘ Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Traditionally, functions were created using plug-ins. These plug-ins were created as custom classes compiled into a .NET Framework assembly, which were then uploaded and registered within Dataverse. However, now with the introduction of functions, users can create them with minimal or no coding required, and without the need for manual registration.

Functions are stored within a Dataverse database and can be seamlessly integrated into Power Apps and Power Automate. The behavior of the workflow is defined using the Power Fx expression language and can directly connect with Dataverse business data and external data sources through Power Platform connectors. With functions, makers can rapidly construct complex logic with minimal coding expertise.

Benefits of creating server-side logic

Defining server-side business logic offers several benefits, including:

- Increased security. Since server-side logic executes on the server, it can help prevent unauthorized access to sensitive data or processes.
- Improved performance. When executed on the server, business logic can reduce the amount of data that needs to be transferred between the client and server, resulting in faster processing times.
- Consistency and reliability. Server-side logic ensures that business rules are consistently applied across all clients, reducing the risk of errors or inconsistencies.

- Easier maintenance and upgrades. By centralizing business logic on the server, it becomes easier to maintain and update, as changes can be made in one place rather than having to update multiple clients.
- Scalability. Server-side logic can be scaled more easily than client-side logic, allowing for better performance and handling of larger workloads.

Functions overview

Functions in Dataverse use Power Fx. Power Fx is the low-code language that is used across Microsoft Power Platform. It's a general-purpose, strong-typed, declarative, and functional programming language used in Power Apps canvas apps as well as areas in model-driven apps, such as custom pages and commanding.

Functions offer these benefits:

- Can use parameters.
- Can manually run with trigger.
- Can support either global or table scope.

Functions have these general properties.

Expand table

Property	Description
Display name	The human-readable name of the function. Can't be changed once created.
Name	The internal name of the function. It's used by the platform to identify the component in code and database operations. Can't be changed once created.
Description	Used to provide additional context about the function, such as purpose, behavior, and so on. Can't be changed once created.
Solution	Used to group components and export to other environments. Learn more about solutions.
Expression	This is the custom function that can be used to perform actions or calculations, defined using the Power Fx expression language. Go to Supported functions for more details.

A function is custom code logic that's manually invoked by a user. Custom input and output parameters can be used with these unique properties.

Expand table

Property	Column2
Table Reference	Used to associate Functions to specific tables. You can select up to five tables to read or write from in your function's formula.
Parameters	Parameters allow you to pass information between the function and the context that runs it, making it easier to design business logic that can be reused in varying situations. Input parameters are used to provide data to the function, and allow you to control how the function behaves by passing in different values you specify in the Power Fx formula. Output parameters allow you to retrieve the results of a function for further use in your program. More information: Supported data types for input and output parameters

Supported data types for input and output parameters

- Boolean
- String
- Float
- Decimal
- DateTime
- Integer

For more information about how to integrate from a canvas app or in a Power Automate cloud flow, go to Invoke a function from a canvas app or custom page.

Functions permissions

Expand table

How run	Description
Design time	Makers who have system customizer security role membership or higher level role in the Power Platform environment can access all functions in that environment. Custom security roles can be used to restrict access to functions.
Run time	When a function is invoked, it accesses the table data involved in the function definition, which includes the tables that are part of the formula in the context of the user who invoked it.

Related content

Learning path: Work with Power Fx functions

Create and use functions in Microsoft Dataverse (preview)

Create and use functions in Microsoft Dataverse (preview)

Article • 02/24/2025

[This topic is pre-release documentation and is subject to change.]

Create and use reusable functions in Microsoft Dataverse. Functions use Power Fx to execute a specific set of commands within Dataverse that run server-side.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Prerequisites

System customizer security role membership in the Power Platform environment.

Create a function in a solution

1. Go to Power Apps, and then select **Solutions** in the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the solution where you want to create a function.
3. On the command bar, select **New** > **Automation** > **Function**.
4. Enter the following information in the **New function** side panel that appears.

    - Provide a Display name and Description for your function.
    - Select **New input parameter** and/or **New output parameter**, then provide a name and data type for the parameter. Add more input and output parameters as needed.
    - In the **Table references** list you can optionally select tables. You can reference the Dataverse tables you choose using data collection functions, such as **Filter()** and **LookUp()**.
    - Enter the Power Fx expression in **the Formula** box.

5. Select **Save**.

6. Test the function.

Reference input parameters in the formula by their names.

Output parameters must be referenced inside of curly brackets, such as `{ Out: "Return value" }`.

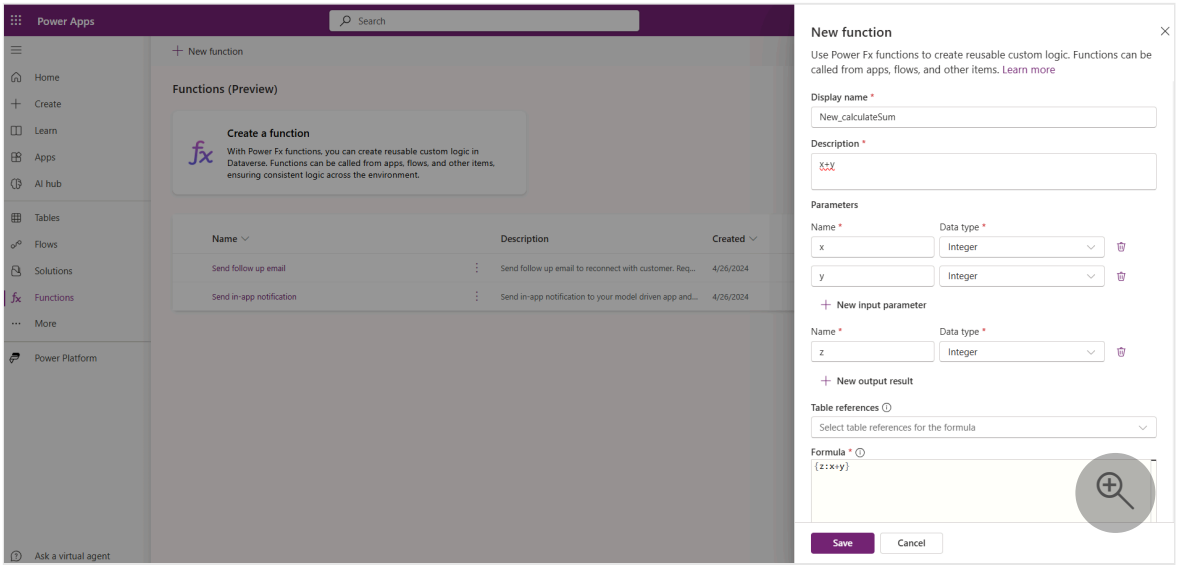
> **Tip**
>
> - Notice the intellisense in the **Formula** box. Underlined red is invalid. Squiggly yellow means your logic might be affected by delegation limitations. Avoid delegation issues by using delegable functions.
> - Expand Advanced options to review your schema name.

Example functions

This section provides a few example functions.

Calculate the sum of two integers

1. Enter a **Display name**, such as **new_calculateSum**, and a **Description**.
2. Add two input parameters, x and y (both of data type integer), and one output parameter, z (data type integer).
3. In the **Formula** box, enter the formula:
   `{ z:x+y }`



4. Test the function.

Add a new title to article table

1. Create two input parameters title and url (both with string data types) and one output variable, message (data type string).
2. Select the **Knowledge Federated Articles** table in the **Table references** dropdown.
3. In the **Formula** box, enter the formula:

PowerApps Formula

```
Collect('Knowledge Federated Articles',
   {
    Title: title,
    URL: url
    }
    );
{
    message: "New Article title added: " & title
}
```

Validate if an input string contains the strings of your choice

1. Create an input parameter **DocumentTextInput** (with string data type) and one output variable, named **result** (data type string).
2. In the **Formula** box, enter the formula:

PowerApps Formula

```
{
    result: If("Confidentiality" in DocumentTextInput && "Dispute
Resolution" in DocumentTextInput && "Governing Law" in DocumentTextInput &&
"Termination" in DocumentTextInput,

    "Document is compliant.",
    "Document is missing one or more compliance clauses."
    )
}
```

Calculate a hotel stay price

1. Create six input parameters: nights, rooms, tax, discount, roomservice, ratepernight (all with string data type) and one output variable, price (data type float).
2. In the **Formula** box, enter the formula:

PowerApps Formula

```
{
   price: ((nights*rooms*ratepernight)*(1+(tax/100))*(1-
(discount/100)))+roomservice

}
```

More function examples

For more example functions, go to Example functions (preview).

Edit, test, or delete a function

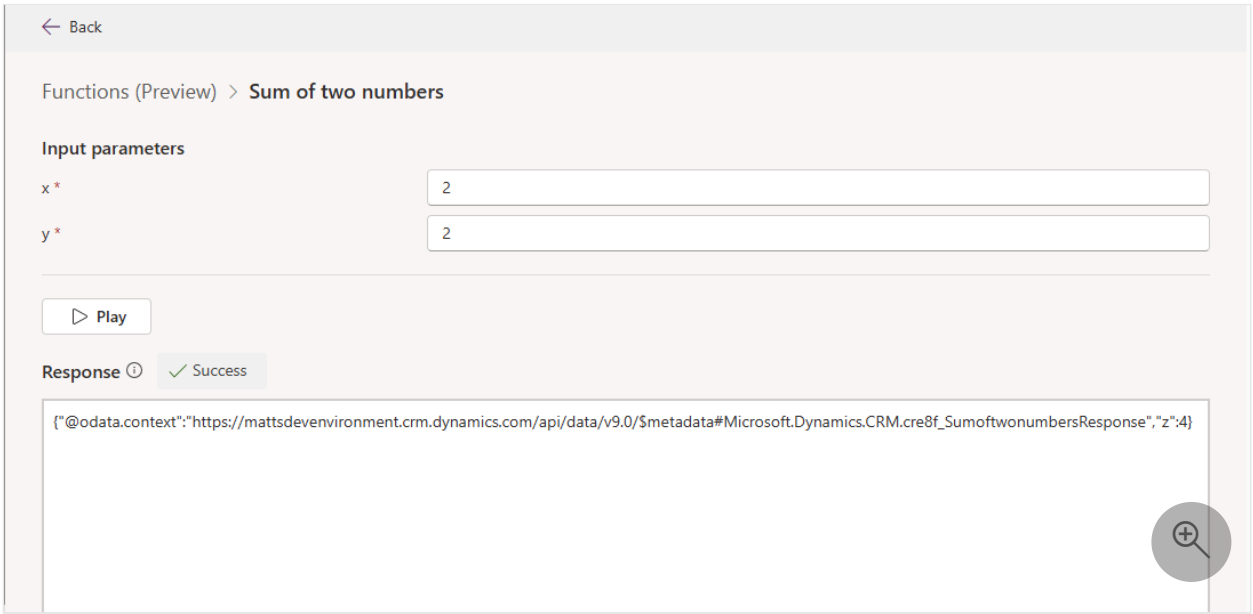
Edit a function

1. Select your function in the **Functions** area.
2. Select **Edit** on the command bar.
3. Modify your formula, and then **Save** it.

Test a function

1. Select the function in the **Functions** area.
2. Select **Test** on the command bar.
3. Provide values for the input parameters that are defined in the function, and then select **Play**.

A successful test returns an OData response that includes information such as the organization URI, function name, and output parameters and values.



> **Tip**
>
> Use output parameters to help validate expected behavior and results. Otherwise, you only observe success or failure when testing.

Delete a function

> **Important**
>
> During the preview, don't delete your functions from the **Solutions** area as it might result in orphan components. Functions should be deleted by going to Power App (make.powerapps.com) > **Functions** on the left navigation pane.

1. Select **Functions** on the left navigation pane, and then select your function. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **Delete** on the command bar.

Related articles

Functions in Microsoft Dataverse (preview)

Feedback

Was this page helpful? Yes No

Provide product feedback

Invoke a function from app, flow, code, or another function (preview)

Article • 02/26/2025

[This topic is pre-release documentation and is subject to change.]

You can invoke functions in Dataverse from a canvas app, a custom page in a model-driven app, a flow, code, or from another function.

> **ⓘ Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Invoke a function from a canvas app or custom page

1. From the **Functions** area in Power Apps (make.powerapps.com), select the function you want to invoke from a canvas app or custom page.
2. Select **Copy code snippet** on the command bar.
3. Paste and save the copied formula to a text editor, Notepad, or somewhere you can easily refer to.
4. In Power Apps Studio:
   a. Create or edit a canvas app or custom page in Power Apps Studio.
   b. On the left navigation pane, under the **Data Sources** tab, select **Add data**, and search for the **Environment** option from the Dataverse connector, and select it.
   c. Insert the following components onto the canvas:

      - Add input controls that correspond with each parameter's data type, such as number input.
      - Add a button to call the function.
      - Add an output control that corresponds with your parameter's data type, such as number input.

5. Select the button you created, and in the **OnSelect** property, enter your function, such as *Environment.new_calculatesum*.

6. Map each input parameter `Value` to reference the corresponding input controls:

    - If your function is `Environment.new_CalculateSum({ X: Value, Y: Value });`, it could be rewritten as: `Environment.new_CalculateSum({ X: NumberInput1.value, Y: NumberInput2.value });`.
    - To set your Function 'result' output, update your formula to 'Set(result, Environment.new_CalculateSum({ X: NumberInput1.value, Y: NumberInput2.value }))'.

7. Select the output control and set the 'Value' property with a formula of 'result.z'.

Preview your app, enter input 1 and 2 values, and select the button. The result should show in the output field.

Invoke functions from a Power Automate cloud flow

1. In a cloud flow, add a new action from the Microsoft Dataverse connector.
2. Select the action called **Perform an unbound action**
3. Select your function. The function has a unique name with a prefix.
4. Provide values for all the input parameters (if any).

Invoke functions from the Dataverse Web API

Follow the steps for the unbound action sections in the Invoking custom APIs from the Web API article (depending on the appropriate scope of the plug-in).

Invoke existing functions from within new functions

To invoke an existing function within a new function, use the syntax: `Environment.ExistingFunction({inputParam1: value1, inputParam2: value2, ... inputParamN: valueN})`

Since the output is always a record, use the dot notation to access the output parameters. For example, if the function "ExistingFunction" has two output parameters defined as out1 and out2, you can access them in either of these two ways:

- `Environment.ExistingFunction({inputParam1: value1, inputParam2: value2, ... inputParamN: valueN}).out1`

- `Environment.ExistingFunction({inputParam1: value1, inputParam2: value2, ... inputParamN: valueN}).out2`

Limitations with functions in Dataverse

- The environment language object needs to be readded to access new functions inside existing canvas apps. For any functions created after you added the environment table data source to an existing canvas app, you must remove and readd the Power Fx environment language object. Then you see the updated list of functions as actions.
- Nested support. functions can only call Microsoft actions published by Microsoft from Power Fx expressions.
- Some `Collect` scenarios require `Patch`. There are some scenarios where `Collect()` doesn't work. The workaround is to use `Patch()` as shown in the populating regarding column example here.

PowerApps Formula

```
Patch(Faxes,
       Collect(Faxes, {  Subject : "Sub1" } ),
       { Regarding : First(Accounts) }
    )
```

Debug and get help with your functions

If you encounter issues creating or running your function, use the `trace()` function for debugging or go to Limitations with functions in Dataverse for common issues that can occur.

Debugging using the trace() function

1. To debug using the trace function, make sure that you enable plug-in and custom workflow activity tracking.
2. Go to Power Apps (make.powerapps.com), select the **Settings** gear icon on the upper right, and then select **Advance settings**.
3. Select Settings > Auditing > Global Audit Settings.
4. On the **Customization** tab, ensure that **Enable logging to plug-in trace log** is enabled for **All**.

Once you enable tracking, you can start using trace() functions to debug Power Fx formulas. Learn more about how to use the trace() function inside a Power Fx

expression: Trace function - Power Platform.

Contacting help + support

For issues with functions not covered in Microsoft Dataverse low-code plug-ins tips and known issues, such as undocumented errors received, use the Help + support experience and include the following information:

- Problem Type- Dataverse Web API and SDK.
- Problem Subtype.

Related articles

Example functions

Create and use functions in Microsoft Dataverse

Feedback

Was this page helpful? Yes No

Provide product feedback

Example functions (preview)

Article • 02/15/2025

[This topic is pre-release documentation and is subject to change.]

The article helps you get started with functions in Microsoft Dataverse by integrating them into your apps. You'll understand that the authoring experience includes authoring Dataverse custom APIs backed by Power Fx expressions, which can trigger actions internal or external to Dataverse.

> **ⓘ Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Test a Power Fx expression

You can test a Power Fx expression to verify it's working. This example uses the `Abs()` function to return the non-negative value of its argument. If a number is negative, `Abs` returns the positive equivalent.

1. Sign into Power Apps and select **Functions** from the left navigation. If the item isn't in the side panel pane, select ...More and then select the item you want. Open the unmanaged solution where you want to add a function.

2. Select **New function**.

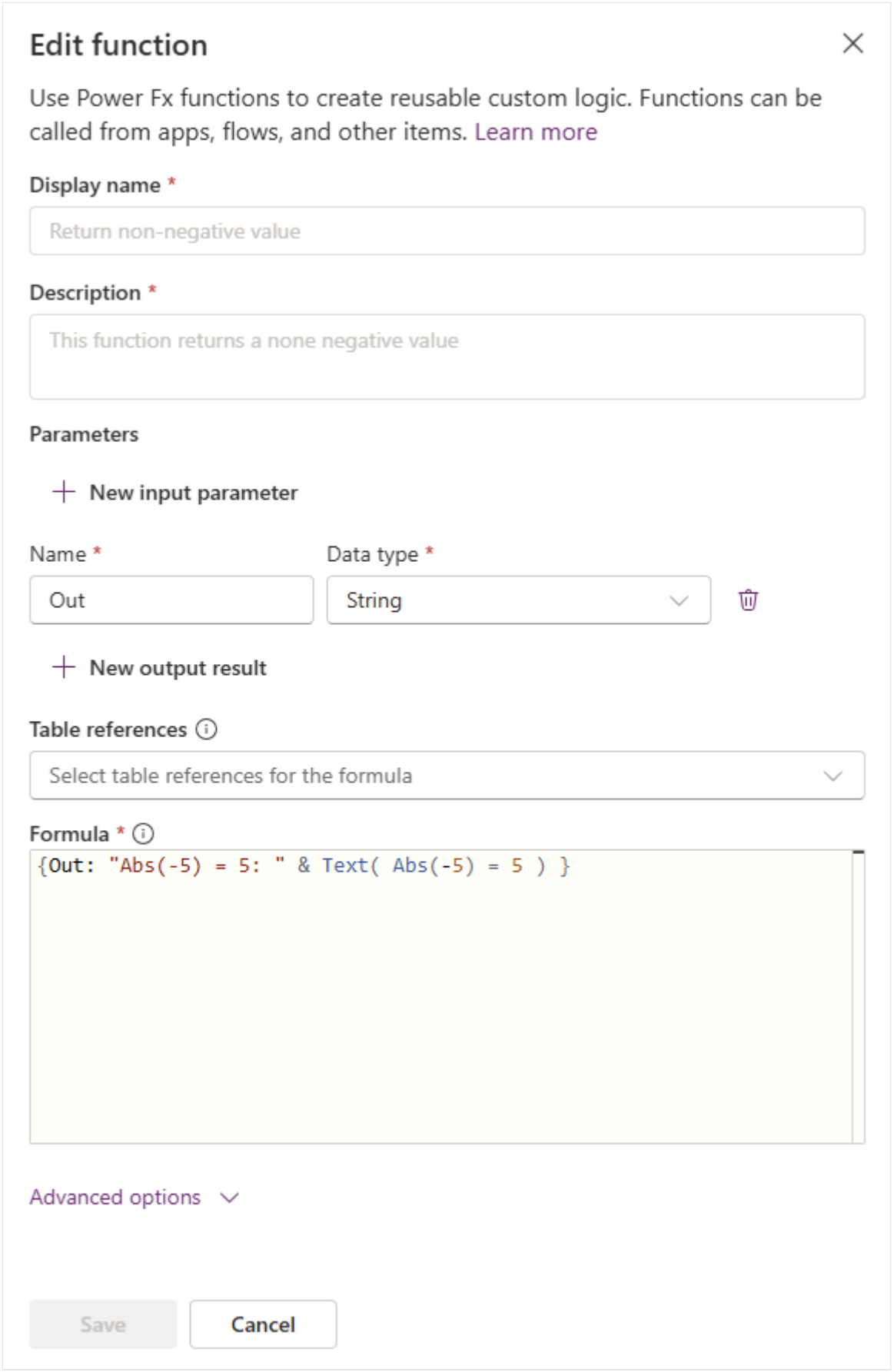
3. On the **New function** pane, enter a **Display name**, such as *Return non-negative value*, and **Description** for your function.

4. Create a **New output parameter** to validate expected behavior, such as a string. For example, enter the **Name** of your **New Output parameter** *Out*.

5. Optionally, use input parameters to make testing easier, that makes sense with the formula. In this example, no input parameters are used.

6. In the **Formula** box, wrap the **Out** parameter in curly brackets: `{Out: "" }` For example, you can test the Abs() function, which uses -5 to return the absolute value 5.

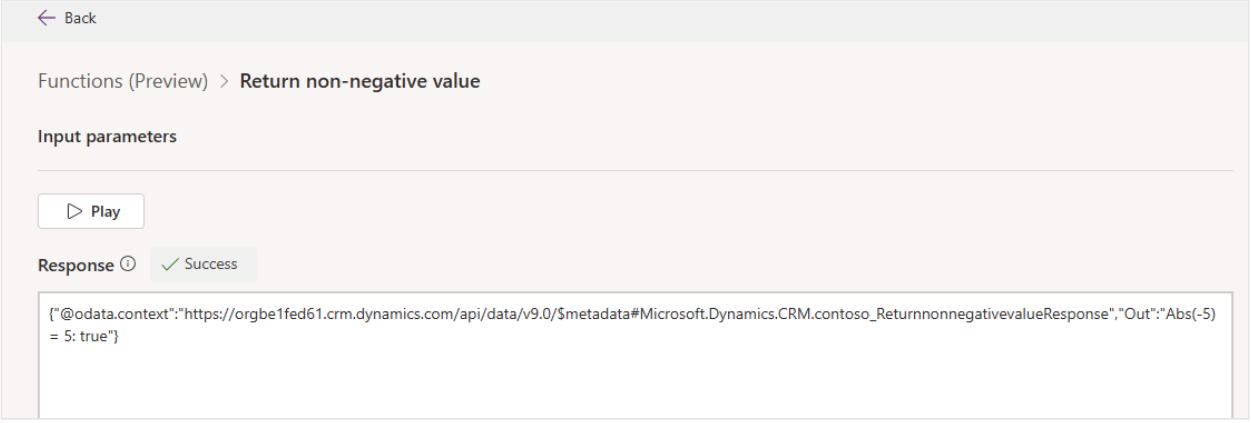
```
{Out: "Abs(-5)=5:"& Text(Abs(-5) = 5) }
```



7. Select **Save**.

To test the function, select the function and then select **Test** on the command bar. Then select **Play**. Based on the formula used in this example function, the response returns

the value 5 as true.



Input validation and custom errors

Send in-app notifications based on an instant action

This and more samples coming soon.

Feedback

Was this page helpful? Yes No

Provide product feedback

Supported functions

Functions in Microsoft Dataverse can add business logic to your apps using the Power Fx expression language and you can quickly build rich workflows without any code. Functions support many of the Power Fx operators, variables, and formulas.

The following table lists the Power Fx formulas that work with functions in Dataverse but have limitations or don't work but have an alternative for use with functions.

Expand table

Power Fx formula or operation	Supported in functions?	Limitation or work around
`Collect`	Yes	Requires the variable to exist and the variable type to match what you're trying to set it to.
`Defaults`	No	Use `Collect` instead of `Patch`. For example, instead of `Patch(account, Defaults(account), {"Account Name": "Example Account"})`, use `Collect(account, {"Account Name": "Example Account"})`.
Add tables in the UI	Yes	For example, in order for the expression to `Collect(Accounts, {... })`, the accounts table needs to be added in the UI. This helps with intellisense.
Accessing fields in formula	Yes	No implicit scope for accessing fields. Instead of saying `Field2`, say `NewRecord.Field2`. This applies to `Set` as well: `Set(NewRecord.Field1, OldRecord.Field2*10)`.
`With()`	Yes	Functions don't support contexts, named formulas, or variables. `With()` can be used to create aliases and factor the expressions.
Access Dataverse tables	Yes	Functions can read and write to Dataverse tables. This includes `Collect()`, `Patch()`, `Filter()`, and `LookUp()`. Delegation operations are supported, and a warning is issued if an expression can't be delegated. These operations run directly against the function's `IOrganizationService` (not the current table) and directly operate on the database. Functions run in the transaction context.

Power Fx functions not supported

The following Power Fx function aren't currently supported with functions in Dataverse.

ClearCollect

Update

UpdateIf

Concurrent

SetFocus

IsType

Download

RemoveIf

GroupBy

SetProperty

Search

UTCNow

UTCToday

Validate

As

Calendar

Choices

Clock

Select

Notify

HashTags

ISOWeekNum

SaveData, LoadData, and ClearData

Form-related formulas (EditForm, NewForm, SubmitForm, ResetForm, and ViewForm)

Device sensor formulas (Acceleration, App, Compass, Connection, and Location)

For more information about Power Fx, go to these articles:

- Expression grammar
- Operators
- Variables
- Formula reference

Last updated on 04/27/2026

Use low-code plug-ins in Dataverse (preview)

Article • 11/13/2024

[This topic is pre-release documentation and is subject to change.]

Microsoft Dataverse offers a powerful solution for achieving more efficient data architecture and reducing client-side workload through low-code plug-ins. These plug-ins are reusable, real-time workflows that execute a specific set of commands within Dataverse, running server-side and triggered by personalized event handlers.

> **ⓘ Important**
>
> - Instant low-code plug-ins are deprioritized and aren't being delivered as a feature. Instant low-code plug-ins are replaced with functions. More information: Functions in Microsoft Dataverse (preview)
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Traditionally, plug-ins were created as custom classes compiled into a .NET Framework assembly, which were then uploaded and registered within Dataverse. However, with the introduction of low-code plug-ins, users can create these event handlers with minimal or no coding required, and without the need for manual registration.

Low-code plug-ins are stored within a Dataverse database and can be seamlessly integrated into Power Apps and Power Automate. The behavior of the workflow is defined using the Power Fx expression language and can directly connect with Dataverse business data and external data sources through Power Platform connectors. With low-code plug-ins, makers can rapidly construct complex workflows with minimal coding expertise, resulting in a more streamlined and efficient data architecture.

Benefits of server-side logic

Defining server-side business logic offers several benefits, including:

- Increased security. Since server-side logic executes on the server, it can help prevent unauthorized access to sensitive data or processes.

- Improved performance. By executing on the server, business logic can reduce the amount of data that needs to be transferred between the client and server, resulting in faster processing times.
- Consistency and reliability. Server-side logic ensures that business rules are consistently applied across all clients, reducing the risk of errors or inconsistencies.
- Easier maintenance and upgrades. By centralizing business logic on the server, it becomes easier to maintain and update, as changes can be made in one place rather than having to update multiple clients.
- Scalability. Server-side logic can be scaled more easily than client-side logic, allowing for better performance and handling of larger workloads.

Low-code plug-ins

There are two types of low-code plug-ins supported in Dataverse:

Expand table

Type	Trigger	Supports parameters	Supported scope
Instant	Manually run	Yes	Global and table
Automated	Dataverse table event	No	Table

All low-code plug-ins have the following common properties:

Expand table

Property	Description
Display name	The human-readable name of the plug-in. Can't be changed once created.
Name	The internal name of the plug-in. It's used by the platform to identify the component in code and database operations. Can't be changed once created.
Description	Used to provide additional context about the plug-in (purpose, behavior, or other important details).
Solution	Used to group components and export to other environments. Learn more about solutions.
Expression	This is the custom function that can be used to perform actions or calculations, defined using the Power Fx expression language. Power Fx is a formula language used in Power Apps canvas apps, and has been extended to be used in low-code plug-ins. See supported functions for more details.

Instant plug-ins

An instant low-code plug-in is custom code logic that's manually triggered by a user. Custom input and output parameters can be used.

Unique properties:

Expand table

Property	Description
Scope	Used to associate a plug-in to a specific table. It can be set to either table (shown as entity) or global, where table (entity) scope means the plug-in is triggered with the context of a specific table record, and global scope means the operation isn't associated with a table (learn more).
Parameters	Parameters allow you to pass information between the plug-in and the context that runs it, making it easier to design business logic that can be reused in varying situations. **Input parameters** are used to provide data to the plug-in, and allow you to control how the function behaves by passing in different values you specify in the Power Fx formula. **Output parameters** allow you to retrieve the results of a function or method for further use in your program. Supported data types: • Boolean • String • Float • Decimal • DateTime • Integer

More information about how to integrate from a canvas app or in a Power Automate cloud flow: Integrate a low-code plug-in

Plug-in permissions

Design time

Makers who have system customizer or system administrator security role membership in the Power Platform environment can access all plug-ins in that environment. Custom security roles can be used to restrict access to low-code plug-ins.

Run time

When a plug-in is invoked, it accesses the table data involved in the plug-in definition (the tables that are part of the formula, or if the table is associated with the settings of an automated plug-in) in the context of the user who invoked it.

Connections

Using security roles, connector access within plug-ins can be restricted to a specific set of users within your organization. Specify which roles have create, read, update, or delete privileges.

Prerequisites for creating a low-code plug-in

- ✓ System administrator or system customizer security role membership in the Power Platform environment.
- ✓ Access to the Dataverse accelerator app.

> **Tip**
>
> All new environments have the Dataverse accelerator app automatically installed as of October 1st 2023. If you already had the Dataverse accelerator installed, you can manually update the Dataverse accelerator.

Update the Dataverse accelerator

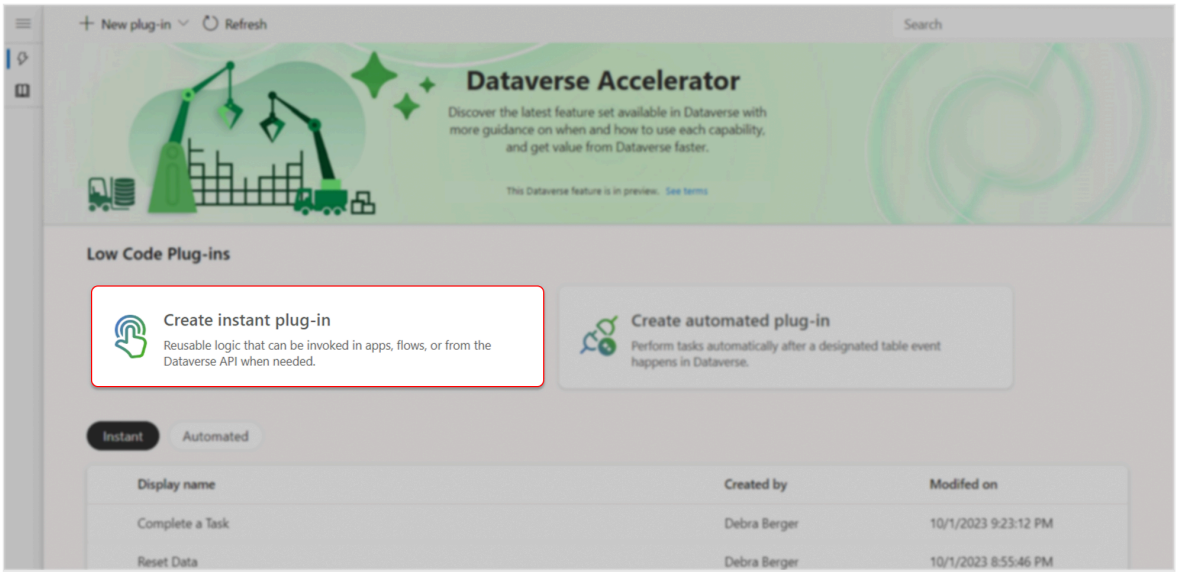
1. Follow the instructions to view licensed apps in your environment.
2. If the **Dataverse Accelerator** is already installed and an update is available, it's indicated in the table next to the item.
3. Select **Dataverse Accelerator**, and then select **Update** on the command bar.

> **Tip**
>
> Enable auto app updates for the **Microsoft - Power CAT** publisher to automatically receive updates when available (not necessary for new environments created after October 1st 2023).

> **Note**
>
> If you previously installed the optional **Low-Code Plug-ins for Connectors** solution, it will be automatically deleted when you update after June 29, 2023. The capabilities will be available in the main solution.

Create an instant low-code plug-in

1. Play the Dataverse accelerator app.
2. Select the **Create instant plug-in** card.



3. Provide a **Display name**.
4. Optionally, define parameters:

    - Select **New input parameter** or **New output parameter**, then enter the label and data type.
    - Add more input and output parameters as needed.

5. Enter the Power Fx expression in the **Expression** editor.

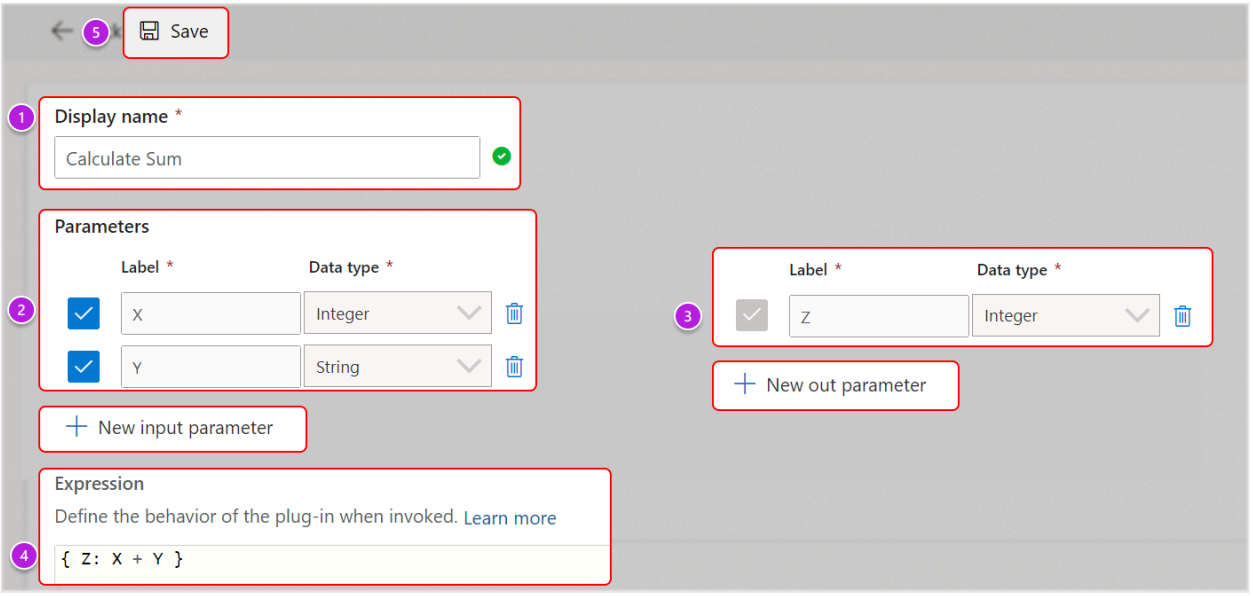
    - Reference input parameters in the formula by the label name.
    - Output parameters must be referenced inside of curly brackets, such as `{ Out: "Return value" }`.
    - Reference Dataverse tables using data collection functions, such as Filter() and LookUp().
    - If the scope is set to entity, use `ThisRecord` to access column values in the table row associated with the plug-in run, such as `ThisRecord.'Account Name'`.

> **Tip**
>
> Note the intellisense in the **Expression** box. Underlined red is invalid. Squiggly yellow means your logic might be affected by delegation limitations. Avoid delegation issues by using delegable functions.

6. Optionally expand **Advanced options** to modify the **solution**, **scope**, or **description**.
7. Select **Save**.
8. Test your instant low-code plug-in.

Example: Calculate the sum of two integers.

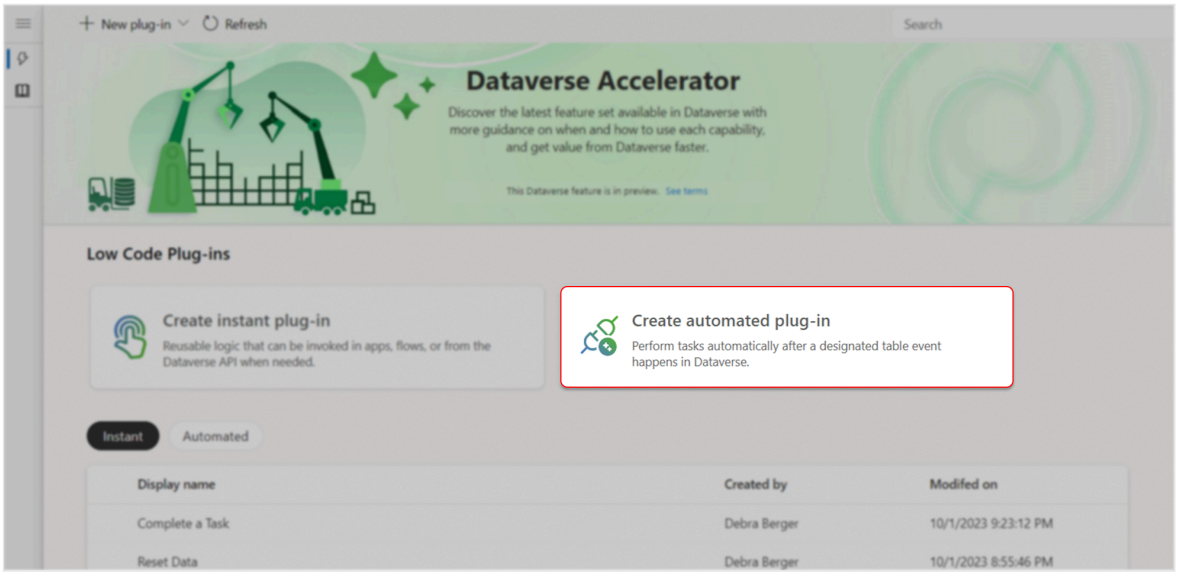
- Create two input parameters, `X` and `Y` (both of type integer), and one output parameter, `Z` (type string).
- Use the following formula: `{Z: X + Y }`



Create an automated low-code plug-in

1. Play the Dataverse accelerator app.

2. Select the **Create automated plug-in** card.



3. Provide the following values:

   - **Name**: Enter a name for the plug-in, such as *Input validation*.
   - **Table**: Choose a table to associate the plug-in to, such as *Account*.
   - **Run this plug-in rule when the row is**. Specify the data event that invokes the plug-in.

4. Enter the Power Fx expression in the **Expression** editor.

   - Reference Dataverse tables using data collection functions, such as Filter() and LookUp().
   - Use `ThisRecord` to access column values in the table row associated with the plug-in run, such as `ThisRecord.'Account Name'`.

5. Optionally, expand **Advanced options** to modify the **stage** (when this should run) and the **solution** where the plug-in is saved.
6. Select **Save**.
7. Test your automated low-code plug-in.

Use Power Platform connectors in low-code plug-ins

Power Platform connectors can be used in low-code plug-ins to easily integrate data and functionality from systems outside of Dataverse, such as SQL Server, Salesforce, and SharePoint, without the need for complex coding or custom development.

Prerequisites for using connectors in low-code plug-ins

✓ Prerequisites for creating a low-code plug-in

✓ Connection reference with an active connection.

Using connector actions in low-code plug-ins

You can easily use connectors and custom connectors from within a low-code plug-in a Power Fx formula.

1. Create a connection to the connector you want to use.
2. Add a connection reference to the connection in the Dataverse environment.
3. In the low-code plug-in Power Fx expression editor, type the name of the connection reference (the internal name with a prefix and underscore, like `new_connectorName`, not the display name).
4. Intellisense shows you the available actions. Select the action you want, and then enter the required parameters.

Before using a connector in a low-code plug-in, review the connector's documentation to ensure that you're passing the input and output parameters correctly.

More information: example

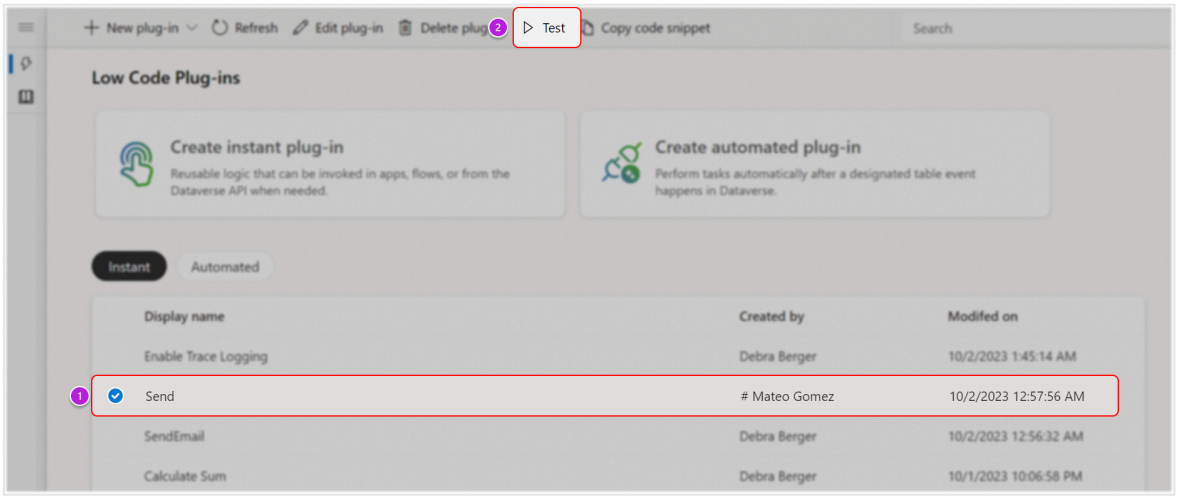
> **Note**
>
> - Not all connector actions are supported at this time.
> - Do not use the Dataverse connector to connect to Dataverse tables from a plug-in expression. Instead, use the native Power Fx functions to interact with collections: Filter, Search, LookUp, Patch, Collect, and Set, where columns are treated as global variables.

Test a low-code plug-in

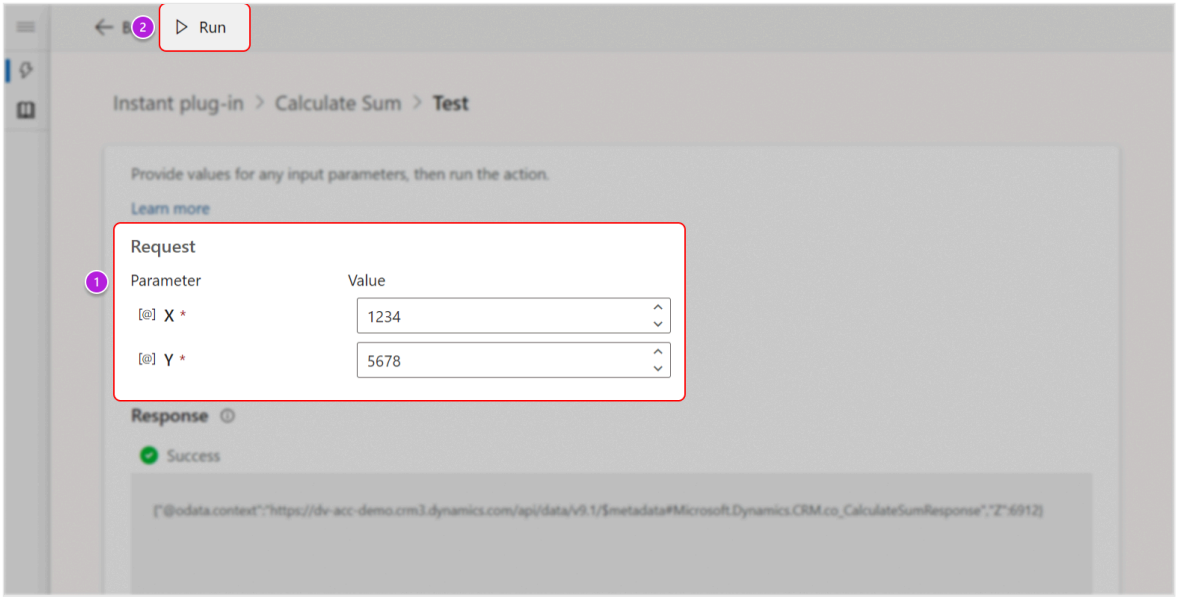
Test instant plug-ins

1. From the **Home screen**, select an instant plug-in from the list, then select **Test** on the command bar. You can also get to this screen from the instant plug-in editor in

the command bar after it's been saved.



2. Provide values for any input parameters that are defined in the low-code plug-in, and then select **Run**.



Observe the response.

> **Tip**
>
> Use output parameters to help validate expected behavior and results. Otherwise, you only see a success or failure when testing.

Test automated plug-ins

Test automated plug-ins by invoking the data event. Observe if the plug-in ran successfully by validating expected changes that were defined in the formula.

> **Tip**

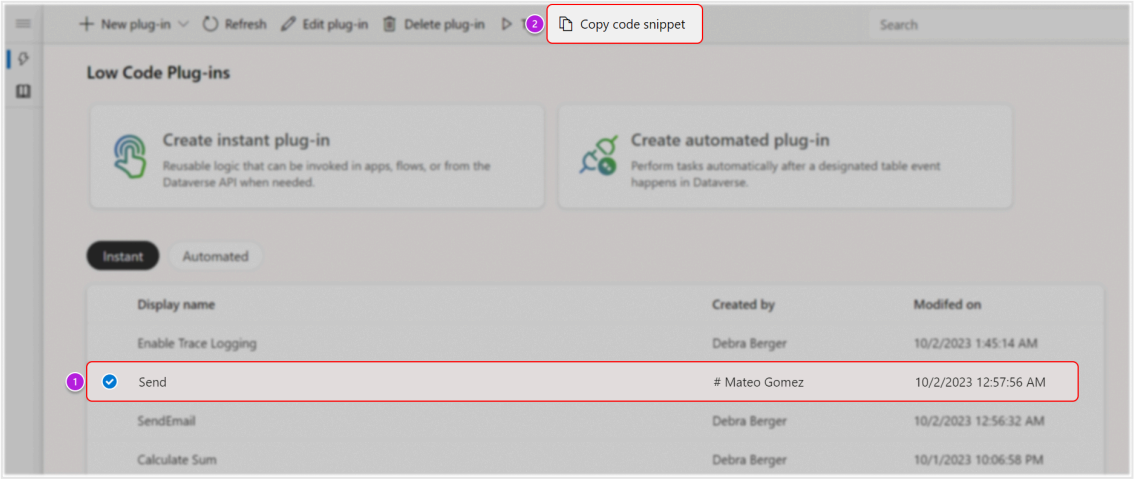
During the preview, you can invoke the associated data event using the table editor in Power Apps:

1. Sign into Power Apps, go to **Tables** and select the **Plug-in** table.
2. On the command bar, select **Edit**.
3. Create, update, or delete rows directly from this view. Alternatively, open a row in the default form by selecting a row, then select **Edit row using form** on the command bar.

Integrate a low-code plug-in

Invoke an instant plug-in from a canvas app or custom page

1. In the **Dataverse Accelerator** app:
   a. Select the instant plug-in in the list.
   b. Select **Copy code snippet** on the command bar.



   c. Paste and save the copied formula to a text editor or notepad (somewhere you can easily refer back to).
2. In Power Apps:
   a. Create or edit a canvas app (or custom page) in the Power Apps studio.
   b. In the left navigation, under the **Data Sources** tab, select **+ New data source**, and search for the **Environment** option from the Dataverse connector.
   c. Insert the following components into the canvas:

      - Add input controls that correspond with each parameter's data type, such as Text input for text or numbers, toggle for boolean.
      - If the plug-in scope is bound to a table, add a combobox that is associated with the same table so you can choose the input.
      - Add a button to call the plug-in.

3. Paste the plug-in formula you copied into the button's `OnSelect` property.
4. Map each input parameter `Value` to reference the corresponding input controls:

    - If the formula was `Environment.new_CalculateSum({ X: Value, Y: Value });`, it could be rewritten as: `Environment.new_CalculateSum({ X: TextInput1.Text, Y: TextInput2.Text });`
    - If the formula was bound, replace `Environment` with the table display name to access the plug-in.

5. If an output parameter is defined for the low-code plug-in:
    a. Capture the response in a `Set()` or `UpdateContext()` formula: `Set( ActionResult, Environments.CalculateSum({ X: TextInput1.Text, Y: TextInput2.Text }) );`. Display the variable in a label. Alternatively use the `Notify()` formula to display data in a notification.
6. Play the app and select the command on the command bar to run the low-code plug-in.

Learn more about how you can call Dataverse actions directly from Power Fx in canvas apps.

Invoke an instant plug-in from a Power Automate cloud flow

1. In a cloud flow, add a new action from the Microsoft Dataverse connector.
2. Select the action called Perform an unbound action or Perform a bound action.
3. Select your plug-in (it has the unique name with a prefix).
4. Provide values for all of the input parameters (if any).

Invoke an instant plug-in from the Dataverse Web API

Follow the steps for **Unbound action** or **Function bound to table** sections in the Invoking custom APIs from the Web API documentation (depending on the appropriate scope of the plug-in).

Getting help with you low-code plug-ins

If you encounter issues creating or running your low-code plug-in, go to these tips for common issues that can occur: Microsoft Dataverse low-code plug-ins tips and known issues

Contacting Help + support

For issues with the Dataverse Accelerator solution installation or low-code plug-ins not covered in Microsoft Dataverse low-code plug-ins tips and known issues, such as undocumented errors received, use the Help + support experience and include the following information:

- Problem Type- Dataverse Web API and SDK
- Problem Subtype- Accelerator kit for Dataverse

Example low-code plug-ins you can create

For a few examples of how to create a low-code plug-in, go to Example Dataverse low-code plug-ins (preview)

Limitations

- The environment language object needs to be readded to access new plug-ins inside existing canvas apps. For any plug-ins created after you have added the environment table data source to an existing canvas app, you have to remove and readd the Power Fx environment language object. Then you see the updated list of plug-ins as actions.

- Intellisense requires explicit notation in automated plug-ins if you want to refer any tables in the formula. Use the following disambiguation syntax such as `[@Accounts]`, using square brackets and the `@` symbol (not `Accounts`).

- Nested support. Plug-ins can only call first-party actions published by Microsoft from Power Fx expressions.

- Some `Collect` scenarios require `Patch`. There are some scenarios where `Collect()` doesn't work. The workaround is to use `Patch()` as shown in the populating regarding column example below. If you're creating an automated plug-in, prepend @ to each table referenced in the Power Fx formula.

  Power Fx

  ```
  Patch(Faxes,
      Collect(Faxes, { Subject : "Sub1" } ),
      { Regarding : First(Accounts) }
  )
  ```

- When low-code plug-ins interact with connectors and DLP is employed, the admin can block creation of connections using DLP. However, existing connection

references in the Dataverse environment continue to function. In case the admin needs to block all low-code plug-in interactions with any connectors, they can disable the organization setting `Allowconnectorsonpowerfxactions`. This setting is enabled by default and can be disabled by usual SDK methods (WebAPI, SDK, PowerShell, and so on). You can disable this using a low-code instant plug-in as follows:

```
Power Fx

Patch(Organizations, First(Organizations), { 'Enable connectors on
power fx actions.': 'Enable connectors on power fx actions.
(Organizations)'.No })
```

- Plug-ins that use connectors can only output results from specific fields. Due to this, you need to map specific primitive values from the connector response to output values.

Low-code plugins that have custom connectors limitation

If you have low-code plugins with custom connectors, the custom connectors must be imported first, before connection references or flows.

If your environment doesn't contain the custom connector in a solution, import a separate solution that only includes the custom connector. Do this import before you import the actual solution. You need to do this first because Azure must register the custom connector before the other solution components can be imported.

If you import a solution that contains custom connectors and flows, Azure isn't able to register the custom connector while it's registering your connection references or flows. This also applies to connection references for the custom connector that wasn't previously imported in a separate solution. If Azure didn't register your custom connector, the import fails, or you aren't able to start the import.

More information: Create custom connectors in solutions: Known limitations

See also

Low-code plug-ins Power Fx (preview)

Use a prebuilt prompt in a Dataverse low-code plug-in in Power Apps

Create an automated low-code plug-in with Power Apps (video)

Create an instant low-code plug-in with Power Apps (video)

Feedback

Was this page helpful? Yes No

Provide product feedback

Example Dataverse low-code plug-ins (preview)

Article • 02/15/2025

[This topic is pre-release documentation and is subject to change.]

The goal of these example plug-ins is to help you get started by integrating them into your apps. You'll understand the authoring experience includes authoring Microsoft Dataverse custom APIs backed by Power Fx expressions, which can trigger actions internal or external to Dataverse.

> **Important**
>
> - Instant low-code plug-ins are deprioritized and aren't being delivered as a feature. Instant low-code plug-ins are replaced with functions. More information: Functions in Microsoft Dataverse (preview)
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Prerequisite

To use one of the example plug-ins for the data event the Dataverse accelerator app must be installed in the environment. More information: Prerequisites for creating a low-code plug-in

> **Note**
>
> Email templates are only available for certain tables. More information: Create templates for email

Return a non-negative value

This example uses the Abs() function to return the non-negative value of its argument. If a number is negative, `Abs` returns the positive equivalent.

1. Play the Dataverse Accelerator app, on the command bar select **New action** > **Instant plugin**.
2. Provide a display name, such as the formula name, and description.
3. Create an `Out` parameter to validate expected behavior that makes sense, such as a string Optionally use input parameters to make testing easier, that makes sense with the formula.
4. In the formula editor, wrap the `Out` parameter in curly brackets:

```
Power Fx

{Out: "" }
```

5. Enter an expression that tests the formula:
   - Validate that intellisense accepts the formula (text turns light blue).
   - Implement an expression that provides an output to help validate the result, for example.

```
Power Fx

{Out: "Abs(-5) = 5: " & Text( Abs(-5) = 5 )  }
```

6. Select **Next**, and then select **Save**.
7. Select **Test** to test the formula. Use the output parameter to validate the result.

Input validation and custom errors

Duplicate detection

Implement server-side input validation, such as duplicate error detection that throws a custom error message.

1. Play the Dataverse Accelerator app, on the command bar select **New action** > **Automated plugin**.
2. In the Name box, enter *Duplicate check*.
3. For **Table**, select **Contact**.
4. For **Run this plugin when the row is**, select **Created**.
5. In the **Formula** box, enter this formula:

Power Fx

```
 If( !IsBlank(LookUp([@Contacts],'Last Name'=ThisRecord.'Last Name' &&
'First Name'=ThisRecord.'First Name')),
    Error("You have existing contacts with the same first name and last
name")
)
```

6. Select **Save**.

Test the plug-in

1. To test the plug-in, create a canvas app using the contacts table by following the steps here: Specify a table.
2. Create a contact row.
3. Create another contact with the same name as in the previous step.
4. A message is displayed indicating duplicate records found. Select **Ignore and save** on the error message prompt.

This custom error message is displayed: **You have two contacts with the same first and last name**.

Data validation

Display specific types of errors using the *ErrorKind* enumeration.

1. Create a new automated plug-in.

2. Provide the following values:

   - **Name**: *Input validation*
   - **Description**: *Checks for valid date and throws an error if invalid*
   - **Table**: **Appointment**
   - **Run this plugin when the row is**: **Updated**

3. Enter the formula below:

   Power Fx

   ```
   If(ThisRecord.'Due Date' < Now(),
    Error({ Kind: ErrorKind.Validation , Message: "The due date cannot be
   in the past" })
   );
   ```

4. Under **Advanced options**, set **When should this run** to **Pre-operation**; you want to run this rule before data is saved to prevent invalid data.

5. Select **Save**.

Go to the Error() function to learn more about custom errors.

Send email based on a data event

Prerequisites:

- ✓ Server-side synchronization is set up for your environment. More information: Set up server-side synchronization of email, appointments, contacts, and tasks
- ✓ An email template.

Example email template

Here's an email template example that you can create for the SenMail based data event:

- **Template type**: **Global**
- **Name**: *Order Thank You*
- **Description**: *Use this template to thank a customer for placing an order with you.*
- **Subject**: `Thank you for your order <orderconfirmation-{!salesorder:Order Number; }>`
- **Body**: Use this code.

```
   Hello {!Sales Order:First Name;},
   Order Type: {! Sales Order: Order Type;},
   Location Type: {! Sales Order: Location Type;},
   Address1: {! Sales Order: Address 1;},
   Address2: {! Sales Order: Address 2;},
   Preferred Service Start Date 1: {! Sales Order: Preferred Service Start
Date;},
   Next Step- We take upto 48 hrs to schedule an in-person and will notify
you as soon as we have a In-person Technician allocated at your site. For
any questions, please contact us at 1-800-CON-SOLAR
   Yours Sincerely,
   Contoso Sales
```

Create the automated plug-in

1. Play the Dataverse Accelerator app, and then select **+New plugin** under **Automated plugins**.
2. Enter the following information:

    - **Name**: *SendEmailUponCreate*
    - **Table**: Select the logical table name of the sales orders, which is **SalesOrder**. This event is based off of Sales Orders table.
    - **Run this plugin with the row is**: **Created**
    - **Formula**: Paste the code below into the **Formula** box. For more information abut the SendEmailFromTemplate function, to SendEmailFromTemplate Action.

    Power Fx

    ```
    XSendEmailFromTemplate(
        LookUp('Email Templates',StartsWith(title,"Order Thank
    You")).'Email Template',
    ThisRecord,
    LookUp(Users,'Primary Email'="sampleemail@sample.com"),
    [ThisRecord.Email]
    )
    ```

3. Select **Advanced** > **Post-operation**.
4. Select **Save**.

The confirmation message **Plugin successfully saved** appears.

Send in-app notifications based on an instant action

In-app notifications enable makers to configure contextual, actionable notifications for users in model-driven apps.

Create the low-code plugin that sends an in-app notification

1. Play the Dataverse accelerator app, and then select **+New plugin** under **Instant plugins**.
2. Enter the following information, select **Next**:

    - **Name**: *NotifyTechnican1*

- **Description**: *This instant plug-in notifies the app user.*

3. On the **Definitions** page, create input parameters with these data types:

   - **OrderID**: **String**
   - **TechnicianEmail**: **String**

4. **Formula**. Paste the following code in the **Formula** box. For more information about this function, go to SendAppNotification Action.

```
Power Fx
```

```
 XSendAppNotification(
    "New service",
    LookUp(Users,'Primary Email'=TechnicianEmail),
    "You have a new solar panel installation scheduled on "&
LookUp('Scheduling Results','OrderId'=OrderID).'ServiceDate'&" in "&
LookUp('Service Orders','Order Number'=OrderID).City &". Contact the
coordinator with any questions.",
 [
    XCreateSidePaneActionForEntity(
            "View order",
        OrderID,
        "Sales Order",
        "cr8b8_serviceorder1",
        LookUp('Service Orders','Order Number'=OrderID).'Service Order'
        )
    ]
 )
```

5. Select **Next**.
6. On the **Summary** page, select **Save**.

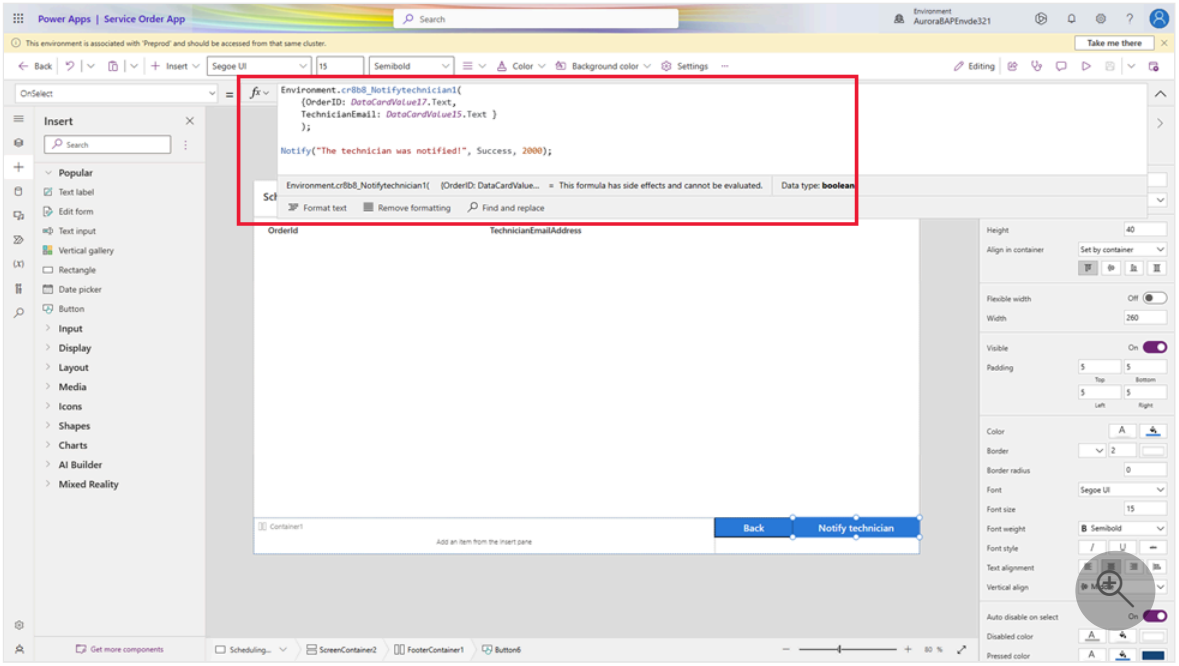
Invoke the in-app notification instant action

1. Select a canvas app and then select **Edit** on the command bar (or create a new one).
2. Select screen on the left navigation pane, or create a new one.
3. On the **Insert** menu, add a **Button** to the page using the **Text** *Notify technician*.
4. Select the button, and enter the following in the **fx** formula bar, where *DataCardValue17* is the column that contains the Order ID, and *DataCardValue15* is the column that contains the technician's email address. In this example, a canvas app named **Service Order App** is used.

```
Power Fx
```

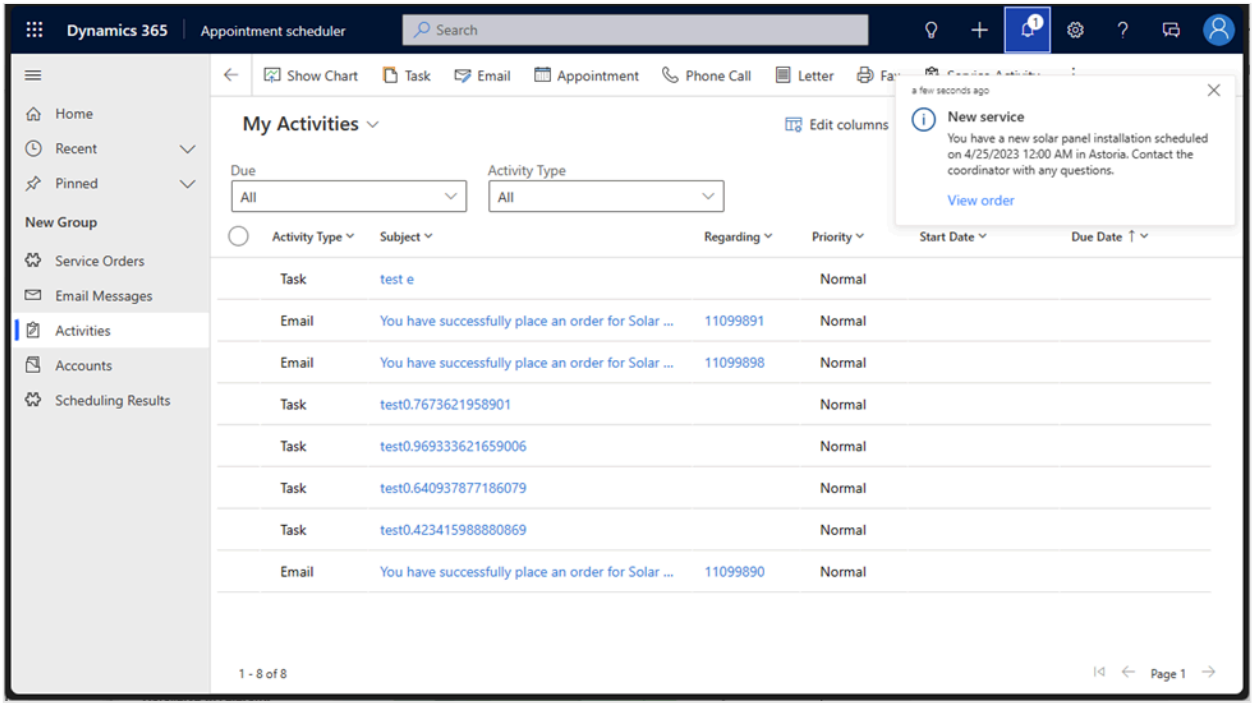
```
 Environment.cr8b8_Notifytechnician1({
       OrderID: DataCardValue17.Text,
```

```
    TechnicianEmail: DataCardValue15.Text
});

    Notify("The technician was notified!", NotificationType.Success,
2000);
```



5. **Save** and **Publish** your changes.

When the notify technician action is selected in the app, an in-app notification is sent to the technician who has been assigned to the service order. An action on the notification opens the service order details in a side pane.



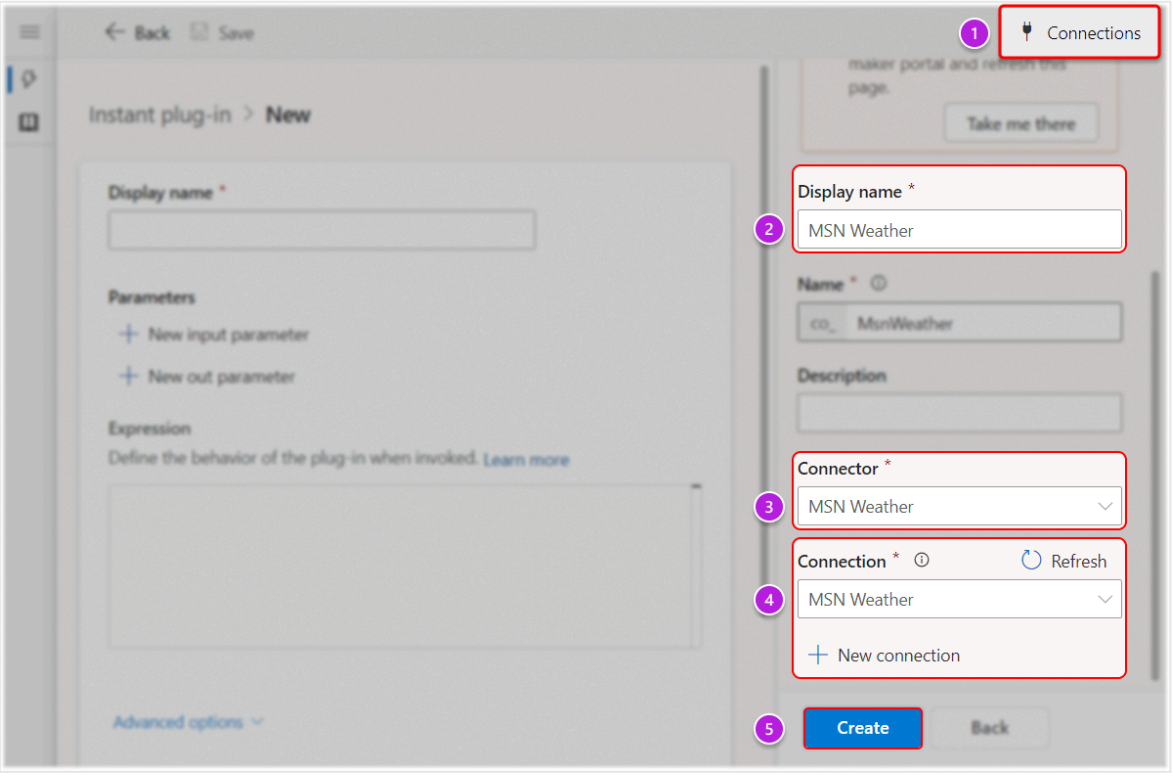
Sample instant plug-in with MSN Weather connector

This plugin returns the current weather for a specific location using MSN Weather connector.

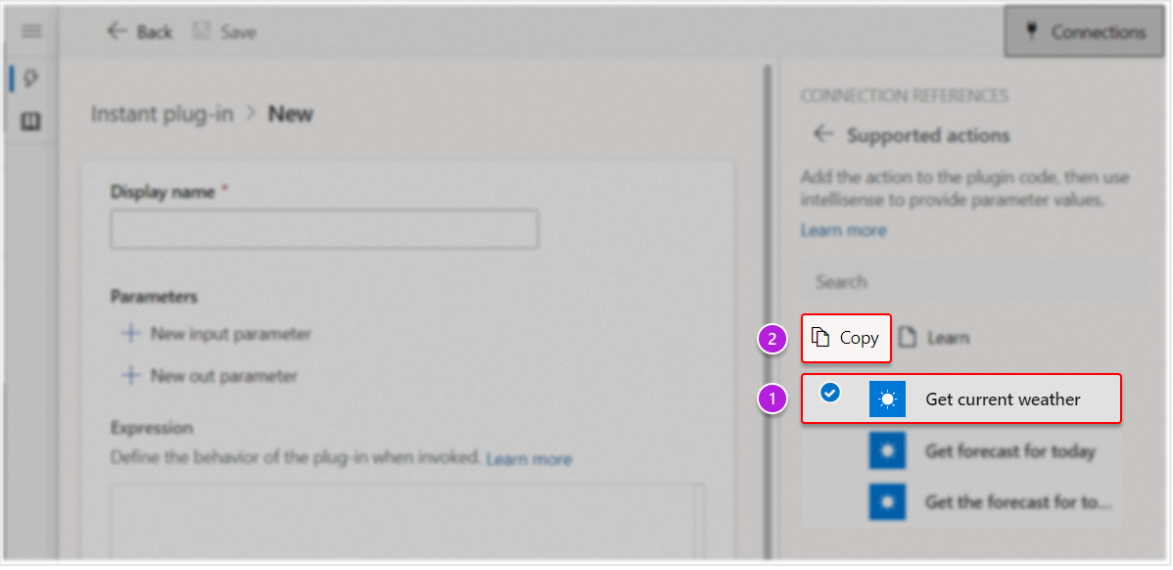
Prerequisites:

- ✓ Prerequisites for creating a low-code plug-in
- ✓ MSN Weather connector is allowed in the environment

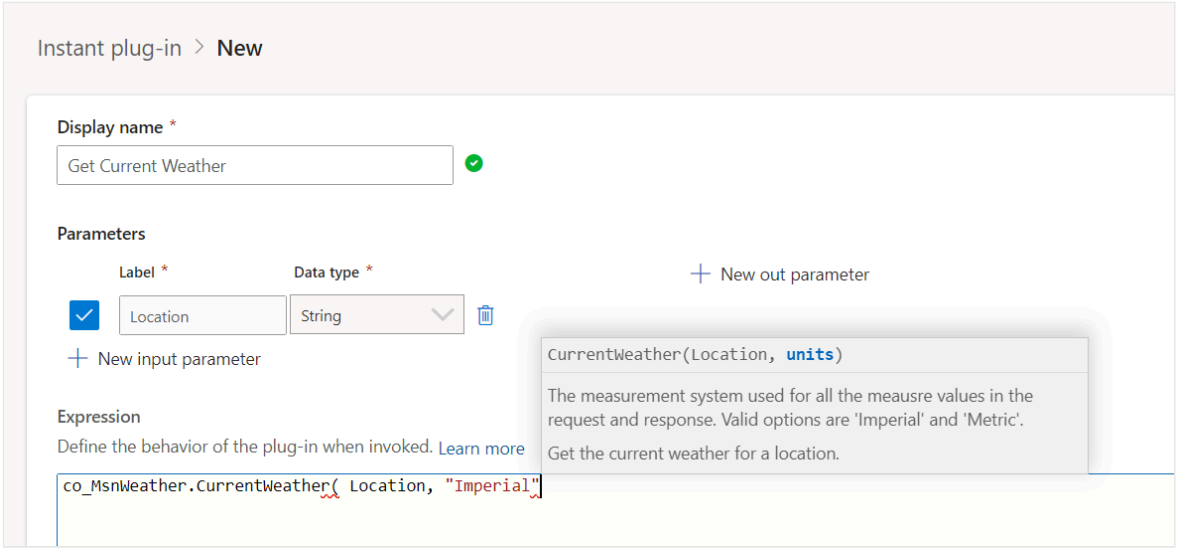
1. Create a connection reference for MSN Weather if not available yet in the environment:



2. Copy snippet:



3. Finish editing the formula using intellisense and consume the connector response properties as needed:



4. Save

5. Test the plug-in

> **Tip**
>
> Use the With() function to capture the entire response from one action if you want to access different properties the response might have. In the example below, there's an input parameter `Location` (string) and an output parameter `Out` (string).

Power Fx

```
With({ /* Capture current weather response from connector */
c: new_MsnWeather.CurrentWeather( Location, "Imperial"
).responses.weather.current
},{ /* Return concatenated weather details */
Out: "Current temp: " & c.temp & " degrees. Feels like " & c.feels & "
degrees. Wind speed is " & c.windSpd & " mph."
})
```

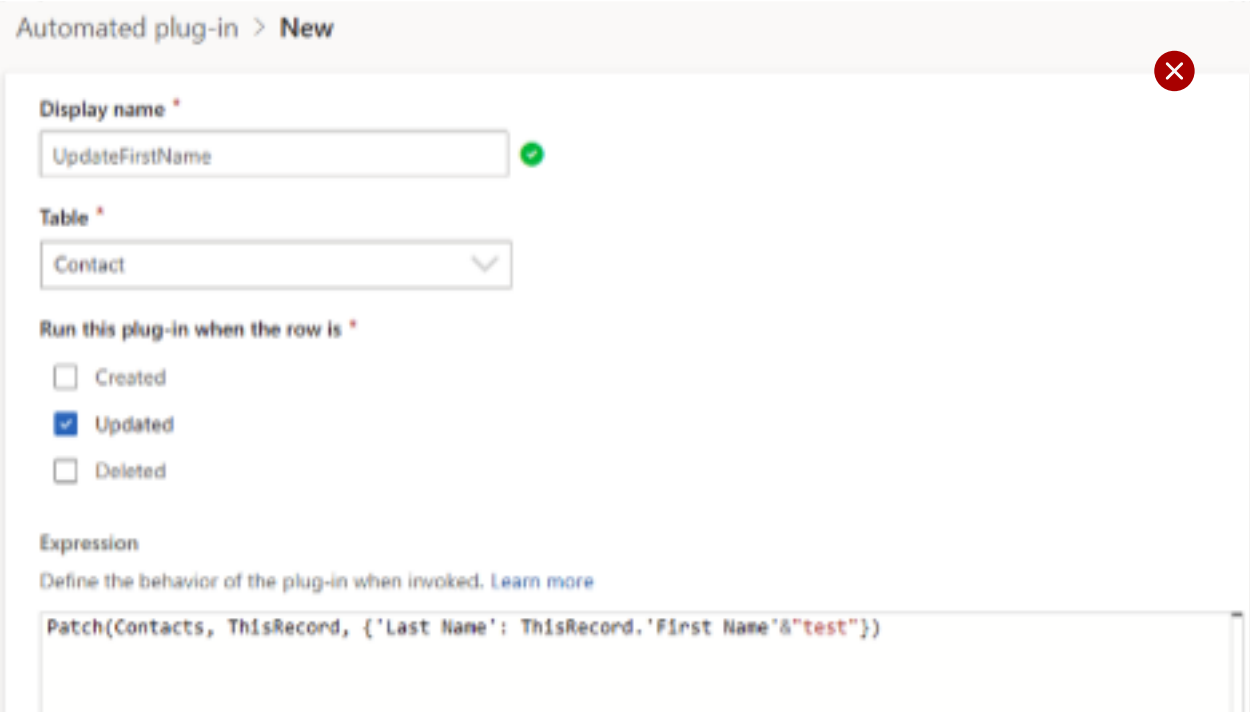
Best practices

Handling infinite loop errors in automated low-code plugins

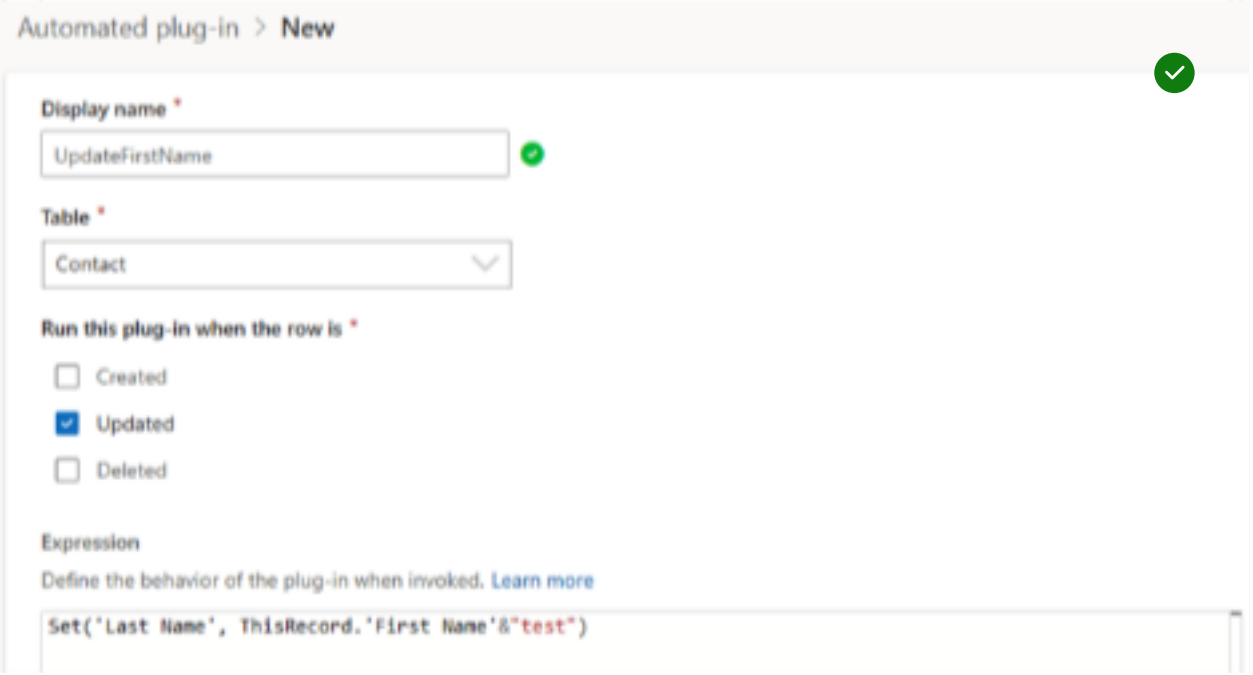
Don't write a patch statement on an automated plugin upon 'Update' event, where the patch is happening on the same table as the plugin. This leads to infinite loops and

plugin execution failures.

Problematic pattern: Using `Patch()` formula triggers another update.



Recommended pattern: Use the `Set()` formula instead to avoid this issue.



See also

Low-code plug-ins Power Fx (preview)

Feedback

Was this page helpful? Yes No

Provide product feedback

Low-code plug-ins Power Fx (preview)

Article • 02/15/2025

[This topic is pre-release documentation and is subject to change.]

Low-code plug-ins can add business logic to your apps using the Power Fx expression language and directly integrate with Dataverse business data and external data through Power Platform connectors. With low-code plug-ins, you can quickly build rich workflows without any code.

> **ⓘ Important**
>
> - Instant low-code plug-ins are deprioritized and aren't being delivered as a feature. Instant low-code plug-ins are replaced with functions. More information: Functions in Microsoft Dataverse (preview)
> - This is an preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Low-code plug-ins support many of the Power Fx operators, variables, and formulas. For more information about Power Fx, go to these articles:

- Expression grammar
- Operators
- Variables
- Formula reference

The following table lists the Power Fx formulas that work but have limitations or don't work but have an alternative for use with low-code plug-ins.

Expand table

Power Fx formula	Supported in plug-ins?	Limitation or work around
Collect	Yes	Requires the variable to exist and the variable type to match what you're trying to set it to.
Defaults	No	Use Collect instead. For example, instead of *Patch(account, Defaults(account), {"Account Name": "Example Account"})* use

Power Fx formula	Supported in plug-ins?	Limitation or work around
		`Collect(account, {“Account Name”: “Example Account”})`.

Power Fx formulas supported

For information about the formulas supported with low-code plug-ins, go to Formula reference- Dataverse plug-ins.

Formulas *not currently supported* with low-code plug-ins

Clear

ClearCollect

Update

UpdateIf

SortByColumns

Concurrent

DropColumns

AddColumns

IsEmpty

SetFocus

IsType

JSON

Download

PlainText

RemoveIf

GroupBy

SetProperty

RenameColumns

Search

ShowColumns

UTCNow

UTCToday

Validate

Weekday

As

Calendar

Choices

Clock

Select

Notify

Errors

HashTags

ISOWeekNum

WeekNum

Refresh

SaveData, LoadData, and ClearData

Form-related formulas (EditForm, NewForm, SubmitForm, ResetForm, and ViewForm)

Device sensor formulas (Acceleration, App, Compass, Connection, and Location)

See also

Use Dataverse low-code plug-ins (preview)

Feedback

Was this page helpful? Yes No

Provide product feedback

Create low-code plug-ins to use with a copilot (preview)

06/23/2025

[This topic is pre-release documentation and is subject to change.]

Copilot actions are a way to extend the capabilities of your copilot bots. With Copilot generative AI actions, you create custom actions that are triggered by your copilot bot. These actions are used to perform a wide range of tasks, such as sending emails, creating records in Microsoft Dataverse, or calling external APIs. In this article, you create a basic low-code plug-in that adds two integers together and a more complex plug-in that can be used to send a notification either of which can be used to create an action in Microsoft Copilot Studio.

> **Important**
>
> - Instant low-code plug-ins are deprioritized and aren't being delivered as a feature. Instant low-code plug-ins are replaced with functions. More information: Functions in Microsoft Dataverse (preview)
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Prerequisites to use the plug-in with Microsoft Copilot

Before you begin, make sure you have the following prerequisites:

- Access to a Dataverse environment that includes the Dataverse Accelerator App for creating low-code plug-ins.
- Access to Copilot Studio in the same environment as your Dataverse environment.

Basic low-code plugin example

In this example you create a low-code instant plug-in that adds two numbers and returns the sum of those numbers. You then create a Copilot action that calls this low-code plug-in to

perform the addition. The copilot you create works for numbers and arbitrary text queries that contain numbers and fractions spelled out.

To create your basic low-code instant plug-in, follow these steps:

1. Open the Dataverse Accelerator App in your Dataverse environment by going to **Apps** > **All** > **Dataverse Accelerator App** and select **Play**.

2. From the Dataverse Accelerator App, select **New plug-in** and select **Instant plug-in**.

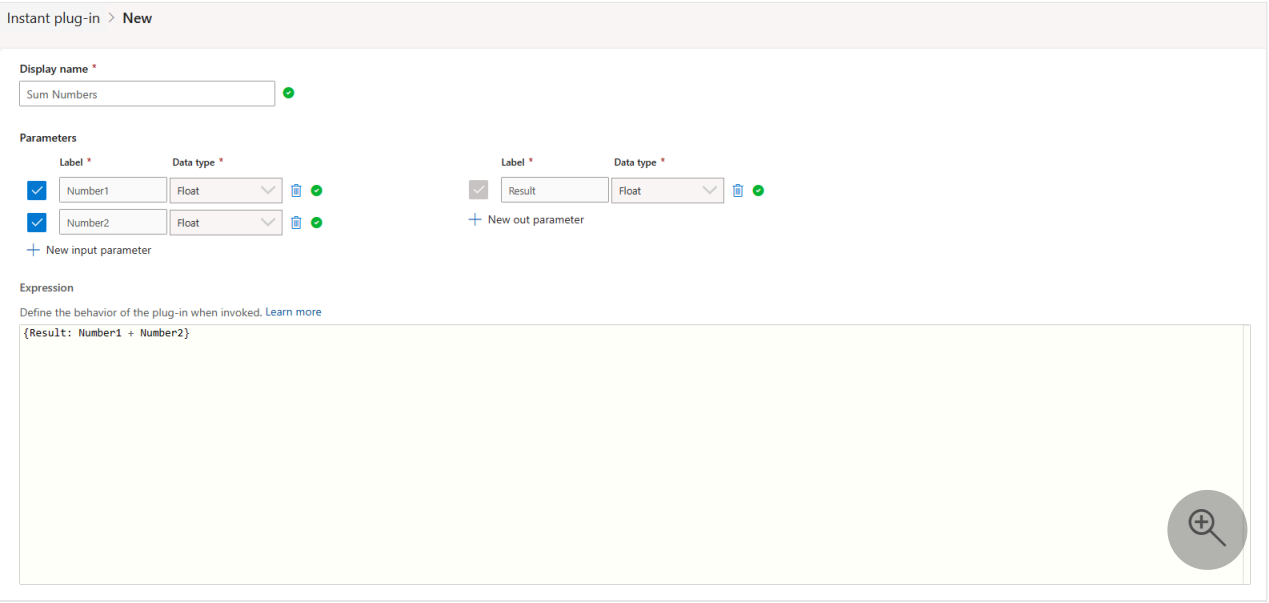
3. Enter a descriptive display name for your plug-in. For example *Sum numbers*, and then select **Create**.

4. In the **Parameters** section:

   - Add the following input parameters:
     - **Parameter Name**: *Number1*, **Type**: **Float**
     - **Parameter Name**: *Number2*, **Type**: **Float**
   - Add the following output parameter:
     - **Parameter Name**: *Result, Type*: **Float**

5. In the **Expression** section, enter the following code:

```
{Result: Number1 + Number2}
```



6. Select **Save** to save your plug-in.

Create a copilot to call your basic low-code plug-in

Go to Copilot Studio, and open the environment that has the plug-in.

1. Create a copilot. More information: Create and configure copilot plugins
2. Enable generative actions on the copilot. More information: Enable generative actions
3. Add a generative AI action. When you add the action, search for 'Perform an unbound action in selected environment' and select the action. More information: Use Generative Actions in Microsoft Copilot Studio
4. Configure the action with the following details:

    - Add a model description that lets the copilot know when to call this action. Use plain English to describe the conditions under which this action should be called. For example, you can enter "Adds numbers and return the sum of those numbers given 2 floating point values." More information: Authoring descriptions
    - Use these **Inputs**:
        - Change **How will the copilot fill this input for the Organization** parameter to **Set as a value**.
        - Select your current environment (the environment in which you created your low-code plug-in) as the value for the **Organization** parameter.
        - Change **How will the copilot fill this input for the Action Name** parameter to **Set as a value**.
        - Select the low-code plug-in you created as the value for the **Action Name** parameter. If you're unsure what the name of your plug-in is you can find it in the Dataverse Accelerator App by selecting it from the list and selecting **Copy code snippet** and then pasting the code into a text editor.
    - Additional **Inputs**:
        - Add and the following input parameters. Adding additional inputs doesn't currently show details in Copilot Studio. You can verify that the inputs were added correctly by selecting **Open the code edit and reviewing the code**, which should show the additional inputs under the inputs section in the action YAML:
            - **Parameter Name**: *Number1*
            - **Parameter Name**: *Number2*

5. Save and publish the copilot.

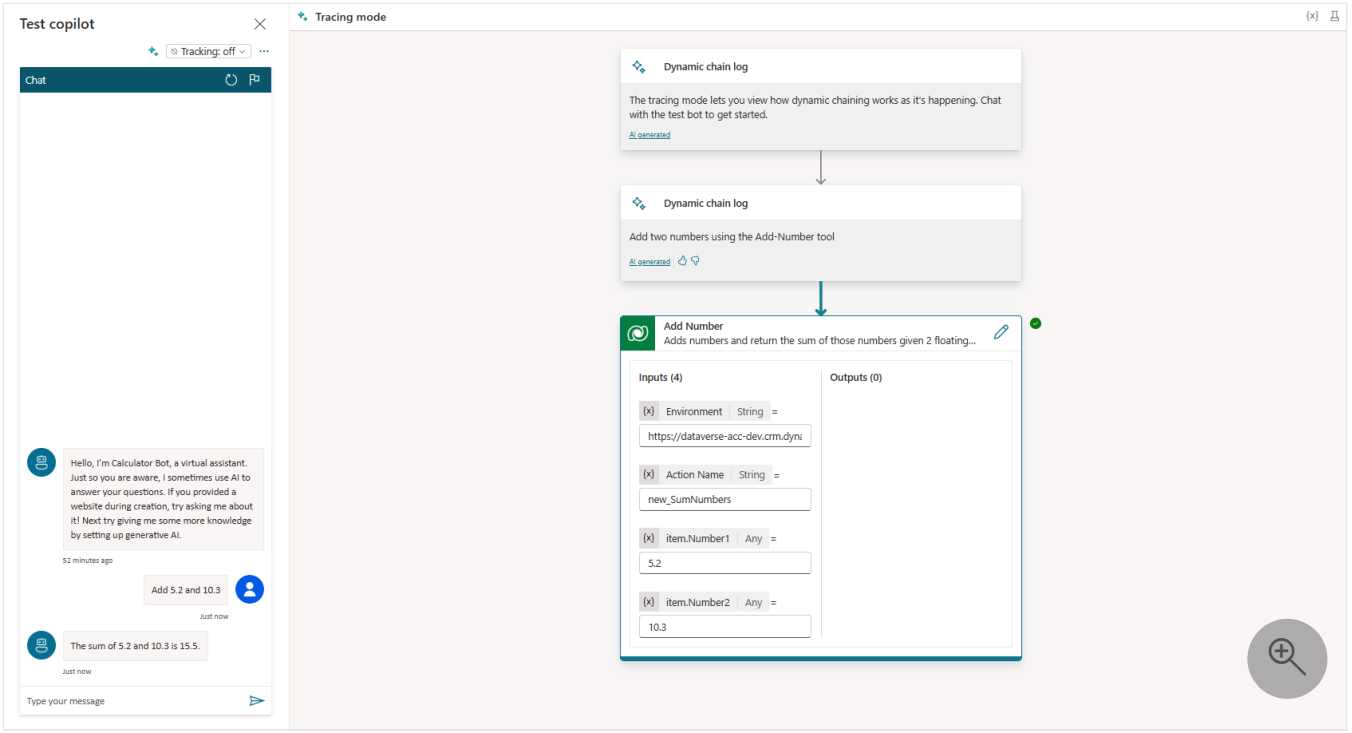
Test your copilot action

Follow these steps in the **Test copilot** pane in Copilot Studio.

1. Select the sparkle icon at the top of the chat window to start tracing mode. This allows you to confirm that your action is being called and verifies the inputs and outputs of your action. More information: Testing you copilot using generative actions
2. Enter the text in the chat window *Add 5.2 and 10.3* and select **Send**.

If everything is configured correctly, you see the expected result of 15.5 output in the chat window and the trace pane shows the inputs and outputs of your action. The trace pane looks similar to the following example. You can try some other prompts to test the action further and change the language to see how the action responds. Some examples:

- *If I have 3 apples and my friend gives me 4 more, how many apples do I have in total?*
- *If I have 1/2 pounds of cheese and I order another 1/4 pounds, how much cheese do I have total?*



Send notification low-code plugin example

In this example you create a low-code instant plug-in that wraps an existing API for use with a copilot. The action sends a notification to the user specified in the copilot action when the user creating the action tests it in Copilot Studio. The copilot determines the required parameters from the plain language query from the user and sends a notification using a Dataverse low-code plugin-in to a model-driven app in Power Apps.

To create your low-code instant plug-in, follow these steps:

1. Open the Dataverse Accelerator App in your Dataverse environment by going to **Apps** > **All** > Dataverse Accelerator App and select play.

2. From the Dataverse Accelerator App, select **New plug-in**, and then select **Instant plug-in**.

3. Enter a descriptive display name for your plug-in. For example, *Send in-app notification to user* and select **Create**.

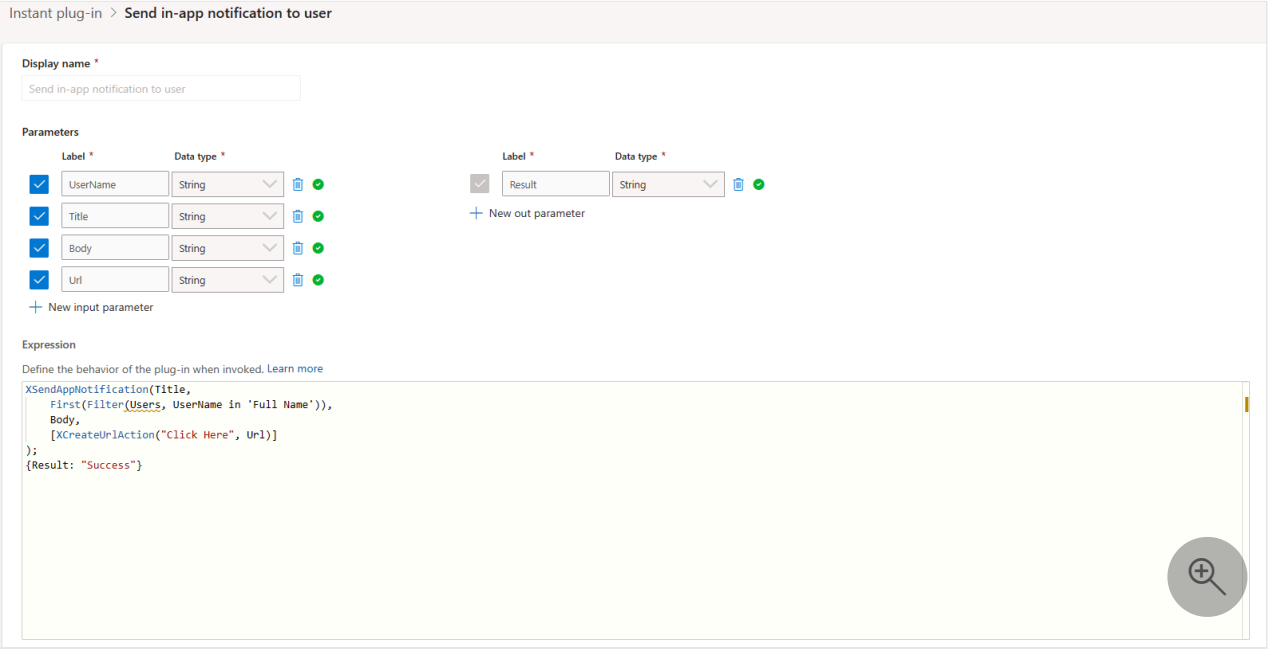
4. In the **Parameters** section:

    - Add the following input parameters:
        - **Parameter Name**: *UserName, Type*: **String**
        - **Parameter Name**: *Title, Type*: **String**
        - **Parameter Name**: *Body, Type*: **String**
        - **Parameter Name**: *Url, Type*: **String**
    - Add the following output parameter:
        - **Parameter Name**: *Result, Type*: **String**

5. In the **Expression** section, enter the following code. Notice that we're wrapping the existing `XSendAppNotification` API with the plugin to give the ability to find users by name. This behavior helps when called from copilot, which can only provide information the user knows about the person and not more esoteric IDs of records that the underlying API is expecting:

    PowerApps Formula

    ```
    XSendAppNotification(Title,
    First(Filter(Users, UserName in 'Full Name')),
    Body,
    [XCreateUrlAction("Click Here", Url)]
    );
    {Result: "Success"}
    ```

6. Select **Save** to save your plug-in.

Create a copilot to call your send notification low-code plug-in

Go to Copilot Studio, and open the environment that has the plug-in.

1. Create a copilot. More information: Create and configure copilot plugins
2. Enable generative actions on the copilot. More information: Enable generative actions
3. Add a generative AI action. When you add the action, search for 'Perform an unbound action in selected environment' and select the action. More information: Use Generative Actions in Microsoft Copilot Studio
4. Configure the action with the following details:

    - Add a model description that lets the copilot know when to call this action. Use plain English to describe the conditions under which this action should be called. For example, you can enter "Calls a Dataverse API to send a notification to a user in Dataverse to remind or alert them to something important with a title, body and optionally a link for more details." More information: Authoring descriptions
    - Use these **Inputs**:
        - Change **How will the copilot fill this input for the Organization** parameter to **Set as a value**.
        - Select your current environment from the dropdown list (the environment in which you created your low-code plug-in) as the value for the **Organization** parameter. You need to put your cursor in the textbox in order for the list of available environments to show.
        - Change **How will the copilot fill this input for the Action Name** parameter to **Set as a value**.
        - Select the low-code plug-in you created as the value for the **Action Name** parameter. You need to put your cursor in the textbox in order for the list of available APIs to show. The value to enter here is the schema name of the low-code plugin, such as `org_SendInAppNotificationToUser`. If you're unsure what the schema name of your plug-in is you can find it in the Dataverse Accelerator App by selecting it from the list and selecting **Copy code snippet** and then pasting the code into a text editor.
    - Additional **Inputs**:
        - Select **Add** and add following input parameters. Notice that adding additional inputs doesn't currently show details in Copilot Studio. This is a known issue. You can verify that the inputs were added correctly by selecting **Open the code edit and reviewing the code**, which should show the additional inputs under the inputs section in the action YAML:
            - **Parameter Name**: *UserName*
            - **Parameter Name**: *Title*
            - **Parameter Name**: *Body*
            - **Parameter Name**: *Url*

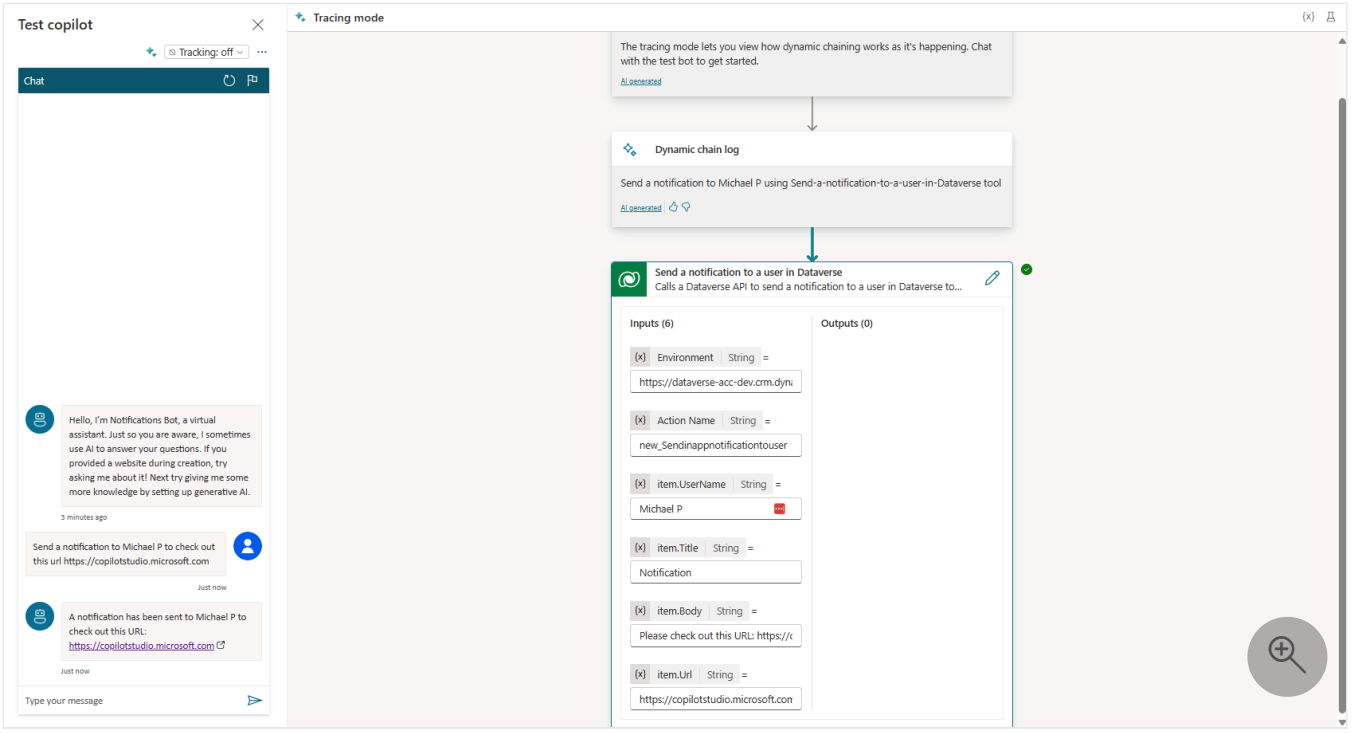
5. Save and publish the copilot.

Test your send notification copilot action

Follow these steps in the **Test copilot** pane in Copilot Studio.

1. Select the sparkle icon at the top of the chat window to start tracing mode. This allows you to confirm that your action is being called and verifies the inputs and outputs of your action. More information: Testing you copilot using generative actions
2. Enter the text in the chat window *Send a notification to my user to check out this url https://copilotstudio.microsoft.com*, and then select **Send**.

If everything is configured correctly, you see that a notification has been sent in the **Test copilot** pane. The notification is sent as a model-driven in-app notification when the specified user plays the app.



You can try some other prompts to test the action further and you could change the plugin to be smarter about how it searches for users given a name to see how the action responds. For example, you could add a step to search for the user by email address if the name doesn't return a result.

See also

Use low-code plug-ins in Dataverse

Microsoft Dataverse low-code plug-ins tips and known issues (preview)

Article • 02/15/2025

[This topic is pre-release documentation and is subject to change.]

This article describes tips and known issues when working with low-code plug-ins in Microsoft Dataverse.

> **ⓘ Important**
>
> - Instant low-code plug-ins are deprioritized and aren't being delivered as a feature. Instant low-code plug-ins are replaced with functions. More information: Functions in Microsoft Dataverse (preview)
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Handle general runtime issues

If you face runtime plugin issues, re-edit the low-code plug-in. Then the intellisense issues on your formula expression are displayed in the low-code plug-in editor. Follow the guidelines to correct the issue that are also displayed, and then resave the plug-in.

Use caution when using post-operation patching

Your low-code plug-in execution might encounter this error when using `Patch` in a post-operation: `Execution failed for PowerPlexPlugin: System.ServiceModel.FaultException 1[Microsoft.Xrm.Sdk.OrganizationServiceFault] This low-code plugin's execution was cancelled because the plugin logic caused an infinite loop. Correct the plugin logic and try again.`

Using `Patch` in a post-operation scenario must be done with caution to avoid infinite loops. A `Patch` operation initiates a new transaction. For example, if an update trigger

for `MyTable` invokes `Patch(MyTable, ThisRecord, ...)`, this operation might result in a recursive update cycle.

Here are a couple of examples of operations that can avoid this issue:

- `Patch(CurrentTable, SomeOtherRecord, ... ) // For example, updating a manager's contact from a contact record.`
- `Patch(OtherTable, SomeOtherRecord, ... ) // Operations on a completely different table.`

Handle the two-minute timeout

For operations that continue for two minutes or longer, you can receive this error:

```
Execution failed for PowerPlexPlugin:
System.ServiceModel.FaultException1[Microsoft.Xrm.Sdk.OrganizationServiceFault]:
Operation not allowed as plugin execution exceeded maximum allowed time (Fault
Detail is equal to Exception details: limit your callbacks .. Create, Update)
```

When working with Dataverse low-code plugins, it's important to manage the two-minute timeout effectively:

- Limit your number of `Patch` and `Collect` operations inside your plug-ins, especially if you have other plug-ins already registered against that table, which can affect the performance of your plug-ins.
- Monitor performance. Keep an eye on the performance of your plug-ins and consider using tracing and logging capabilities within Dataverse to track execution times and failures. By following these guidelines, you can ensure that your low-code plug-ins run smoothly within the Dataverse environment without interruption caused by the two-minute timeout. More information: Tracing and logging

Failed response received from APIM

If you receive this error message, which can be returned from API management (APIM), just edit the plug-in, and then resave. Saving initializes the APIM authentication and your plug-in begins executing successfully.

```
Execution failed for PowerPlexPlugin: Failed response received from APIM;
StatusCode: NotFound; ResponseContent: { "statusCode": 404, "message": "Resource
not found" } Method: POST; RequestUri: https://canada-001.azure-apim.net/invoke;
StatusCode: NotFound; ResponseContent: { "statusCode": 404, "message": "Resource
```

```
not found" }; HeadersString: Headers - 'Access-Control-Allow-Methods':
'System.String[]'; 'Access-Control-Allow-Origin': 'System.String[]'; 'Access-
Control-Max-Age': 'System.String[]'; 'Access-Control-Expose-Headers':
'System.String[]'; 'Date': 'System.String[]'; Access to APIM expires..edit and save
the plugin
```

See also

Use low-code plug-ins in Dataverse

Feedback

Was this page helpful? Yes No

Provide product feedback

Use custom process actions

Article • 02/15/2022

Custom process actions, also known as *Custom actions*, or just *actions* open a range of possibilities for composing business logic. With custom process actions, you can perform operations, such as Create, Update, Delete, Assign, or Perform Action. Internally, a custom process action creates a custom message. Developers refer to these actions as *messages*. If the goal of a process is to create a row, then update it, and then assign it, there are three separate steps. Each step is defined by the capabilities of the table—not necessarily your business process.

Custom process actions let you define a single verb (or message) that matches an operation you need to perform for your business. These new messages are driven by a process or behavior rather than what can be done with a table. These messages can correspond to verbs like Escalate, Convert, Schedule, Route, or Approve—whatever you need. The addition of these verbs helps provide a richer vocabulary for you to fluently define your business processes. You can apply this richer vocabulary from clients or integrations rather than having to write the action within clients. This also makes it easier because you can manage and log the success or failure of the entire action as a single unit.

Configurable messages

Once an action is defined and activated, a developer can use that message like any of the other messages provided by the platform. However, a significant difference is that now someone who isn't a developer can apply changes to what should be done when that message is used. You can configure the action to modify steps as your business processes change. Any custom code that uses that message doesn't need to be changed as long as the process arguments don't change.

Workflow processes and plug-ins continue to provide similar capabilities for defining automation. Workflow processes still provide the capability for a nondeveloper to apply changes. But the difference is in how the business processes are composed and how a developer can write their code. A custom process action is a message that operates on the same level as any of the messages provided by the platform. Developers can register plug-ins for actions.

> **Note**

Custom API is a newer way to define custom messages with many advantages for developers. If you do not intend to use the no-code capabilities that custom process actions provide to configure business logic, custom API provides better capabilities for developers to create their own messages. More information:

- Compare Custom Process Action and custom API
- Create and use custom APIs

Global messages

Unlike Microsoft Dataverse workflows or plug-ins, a custom process action doesn't have to be associated with a specific table. You can define global custom process actions that can be called on their own.

Next steps

Create a custom process action

Create a custom process action

Article • 02/15/2022

Use custom process actions when you want to automate a series of commands in the system. Custom process actions expand the vocabulary available for developers to express business processes. An custom process action uses core verbs provided by the system, such as Create, Update, Delete, and Assign, to create more expressive verbs like Approve, Escalate, Route, or Schedule. If the definition of a business process changes, someone who isn't a developer can edit the custom process action so the code doesn't need to be changed.

> **Note**
>
> If you intend to write a plug-in to implement your logic for a custom process action instead of using the workflow designer, you should use custom API instead. More information: Compare Custom Process Action and custom API

Create a custom process action

> **Important**
>
> If you're creating an custom process action to include as part of a solution that will be distributed, create it in the context of the solution. Go to Settings > **Solutions** and locate the unmanaged solution that this action will be part of. Then, in the menu bar, select **New** > **Process**. This ensures that the customization prefix associated with the name of the action will be consistent with other components in the solution. After you create the action, you can't change the prefix.

Like workflow processes, custom process actions have the following properties in the **Create Process** dialog box:

- **Process name**

  After you enter a name for the process, a unique name will be created for it by removing any spaces or special characters from the process name.

- **Category**

  This property establishes that this is an action process. You can't change this after you save the process.

- **Table**

  With actions processes, you can select a table to provide context for the workflow just like other types of processes, but you also have the option to choose **None (global)**. Use this if your action doesn't require the context of a specific table. You can't change this after you save the process.

- **Type**

  Use this property to choose whether to build a new custom process action from scratch or to start from an existing template.

Unlike workflow processes, you don't need to set the following options:

- **Start When**: Actions start when code calls the message generated for them.
- **Scope**: Actions always run in the context of the calling user.
- **Run in the background**: Actions are always real-time workflows.

Custom process actions also have something that workflow processes don't—input and output arguments.

> ⓘ **Note**
>
> You can enable a custom process action from a workflow without writing code. More information: Invoke custom process actions from a workflow

Edit a custom process action

You must deactivate custom process actions before you can edit them.

You can edit an custom process action that was created as part of an unmanaged solution or included in a solution installed in your organization. If the solution is a managed solution, you might not be able to edit it. The solution publisher has the option to edit the managed properties so that the action installed with a managed solution can't be edited.

When an action is saved, a unique name is generated based on the process name. This unique name has the customization prefix added from the solution publisher. This is the name of the message that a developer will use in their code.

When editing an action, you have the following options:

- **Process name**

  After the process is created and the unique name is generated from the process name, you can edit the process name. You might want to apply a naming convention to make it easier to locate specific processes.

- **Unique name**

  When an action is saved, a unique name is generated based on the process name. This unique name has the customization prefix from the solution publisher added. This is the name of the message that a developer will use in their code. Don't change this unique name if the process has been activated and code is in place expecting to call the action using this name.

  > **ⓘ Important**
  >
  > After the action is activated and code is written to use a unique name, the unique name must not be changed without also changing the code that references it.

- **Enable rollback**

  Generally, processes that support transactions will "undo" (or roll back) the entire operation if any part of them fails. There are some exceptions to this. Some actions developers might do in code initiated by the custom process action might not support transactions. For example, if the code perform actions in other systems that are beyond the scope of the transaction. Those can't be rolled back by the action running in an app. Some messages in the platform don't support transactions. But everything you can do just with the user interface of the action will support transactions. All the actions that are part of a real-time workflow are considered in transaction, but with actions you have the option to opt out of this.

  You should consult with the developer who will use this message to determine whether it must be in transaction or not. Generally, an action should be in transaction if the actions performed by the business process don't make sense unless all of them are completed successfully. The classic example is transferring funds between two bank accounts. If you withdraw funds from one account you must deposit them in the other. If either fails, both must fail.

  > **ⓘ Note**

You can't enable rollback if a custom process action is invoked directly from within a workflow. You can enable rollback if an action is triggered by a Power Apps web services message.

- **Activate as**

  Like all processes, you can activate the process as a template and use it as an advanced starting point for processes that follow a similar pattern.

- **Define process arguments**

  In this area, you'll specify any data that the action expects to start and what data will be passed out of the action. More information: Define process arguments

- **Add stages and steps**

  Like other processes, you specify what actions to perform and when to perform them. More information: Add stages and steps

Define process arguments

When developers use messages, they might begin with some data that they can pass into the message. For example, to create a new case row, there might be the case title value that is passed in as the input argument.

When the message is finished, the developer might need to pass some data that was changed or generated by the message to another operation in their code. This data is the output argument.

Both input and output arguments must have a name, a type, and some information about whether the argument is always required. You can also provide a description.

The name of the message and the information about all the process arguments represent the *signature* for the message. After an custom process action is activated and is being used in code, the signature must not change. If this signature changes, any code that uses the message will fail. The only exception to this may be changing one of the parameters so that it is not always required.

You can change the order of the arguments by sorting them or moving them up or down because the arguments are identified by name, not by the order. Also, changing the description won't break code using the message.

Action process argument types

The following table describes the action process argument types.

Expand table

Type	Description
Boolean	A `true` or `false` value.
DateTime	A value that stores date and time information.
Decimal	A number value with decimal precision. Used when precision is extremely important.
Table	A row for the specified table. When you select table, the drop-down list is enabled and allows you to select the table type.
entityCollection	A collection of table rows.
entityReference	An object that contains the name, ID, and type of a table row that uniquely identifies it. When you select entityReference, the drop-down list is enabled and allows you to select the table type.
Float	A number value with decimal precision. Used when data comes from a measurement that isn't absolutely precise.
Integer	A whole number.
Money	A value that stores data about an amount of money.
Picklist	A value that represents an option for an OptionSet attribute.
String	A text value.

> **Note**
>
> **entityCollection** argument values can't be set in the user interface for conditions or actions. These are provided for use by developers in custom code. More information: Create your own messages

Add stages and steps

Custom process actions are a type of process very similar to real-time workflows. All the steps that can be used in real-time workflows can be used in actions. For information about the steps that can be used for both real-time workflows and actions, see Workflow stages and steps.

In addition to the steps that can be used for real-time workflows, actions also have the **Assign Value** step. In actions, these can be used only to set output arguments. You can use the form assistant to set output arguments to specific values or, more likely, to values from the row that the action is running against, rows related to that row with a many-to-one relationship, rows created in an earlier step, or values that are part of the process itself.

Managed properties

Custom process actions have two relevant managed properties

Is Customizable

The **Is Customizable** managed property controls whether on not someone who installs a managed solution containing the custom process action can edit or delete it. If wish for the custom process action to be edited or deleted when included in a managed solution, you should set this property to true.

Is Custom Processing Step Allowed for Other Publishers

The **Is Custom Processing Step Allowed for Other Publishers** managed property controls whether third party plug-in developers can register plug-in steps on the message created by the custom process action. When this is true, plug-ins registered by anyone on this message will run and can modify the behavior of the custom process action. When false, only plug-ins steps registered within a solution from the same solution publisher will run.

Next steps

Invoke custom process actions from a workflow

Configure custom process actions from a workflow

Article • 02/15/2022

You can enable a custom process action from a real-time workflow without writing code. More information: Invoke custom actions from a workflow

You also can create a custom process action so that a developer can use it in code, or you might need to edit an action that was previously defined. Like real-time workflow processes, consider the following:

- What should the action do?
- Under what conditions should the action be performed?

Unlike real-time workflow processes, you don't need to set the following options:

- **Start When**: Custom process actions start when code calls the message generated for them.
- **Scope**: Custom process actions always run in the context of the calling user.
- **Run in the background**: Custom process actions are always real-time workflows.

Custom process actions also have something that real-time workflow processes don't—input and output arguments. More information: Define process arguments

Create a custom process action

> **Important**
>
> If you're creating an action to include as part of a solution that will be distributed, create it in the context of the solution. Go to **Settings** > **Solutions** and locate the unmanaged solution that this action will be part of. Then, in the menu bar, select **New** > **Process**. This ensures that the customization prefix associated with the name of the action will be consistent with other components in the solution. After you create the action, you can't change the prefix.

> **Note**

If you intend to write a plug-in to implement your logic for a custom process action instead of using the workflow designer, you should use custom API instead. More information: Compare Custom Process Action and custom API

Like real-time workflow processes, actions have the following properties in the **Create Process** dialog box:

- **Process name**

  After you enter a name for the process, a unique name will be created for it by removing any spaces or special characters from the process name.

- **Category**

  This property establishes that this is an action process. You can't change this after you save the process.

- **Entity**

  With action processes, you can select a table to provide context for the real-time workflow just like other types of processes, but you also have the option to choose **None (global)**. Use this if your action doesn't require the context of a specific table. You can't change this after you save the process.

- **Type**

  Use this property to choose whether to build a new custom process action from scratch or to start from an existing template.

Edit an action

You must deactivate processes before you can edit them.

You can edit a custom process action that was created as part of an unmanaged solution or included in a solution installed in your organization. If the solution is a managed solution, you might not be able to edit it. The solution publisher has the option to edit the managed properties so that the action installed with a managed solution can't be edited.

When an action is saved, a unique name is generated based on the process name. This unique name has the customization prefix added from the solution publisher. This is the name of the message that a developer will use in their code.

When editing an action, you have the following options:

- **Process Name**

  After the process is created and the unique name is generated from the process name, you can edit the process name. You might want to apply a naming convention to make it easier to locate specific processes.

- **Unique Name**

  When a custom process action is saved, a unique name is generated based on the process name. This unique name has the customization prefix from the solution publisher added. This is the name of the message that a developer will use in their code. Don't change this unique name if the process has been activated and code is in place expecting to call the action using this name.

  > ⓘ **Important**
  >
  > After the action is activated and code is written to use a unique name, the unique name must not be changed without also changing the code that references it.

- **Enable rollback**

  Generally, processes that support transactions will "undo" (or roll back) the entire operation if any part of them fails. There are exceptions to this. Actions executed in code by a developer initiated by the custom process action might not support transactions. An example would be when the code perform actions in other systems that are beyond the scope of the transaction. Those can't be rolled back by the action running in an app. Some messages in the platform don't support transactions. But everything you can do just with the user interface of the action will support transactions. All the actions that are part of a real-time workflow are considered in transaction, but with actions you have the option to opt out of this.

  You should consult with the developer who will use this message to determine whether it must be in transaction or not. Generally, an action should be in transaction if the actions performed by the business process don't make sense unless all of them are completed successfully. The classic example is transferring funds between two bank accounts. If you withdraw funds from one account you must deposit them in the other. If either fails, both must fail.

  > ⓘ **Note**
  >
  > - You can't enable rollback if a custom process action is invoked directly from within a workflow. You can enable rollback if an custom process

action is triggered by a Power Apps web services message.
    - If the action defined does not change data, but only retrieves data, there are certain situations where performance can be improved by disabling **Enable rollback.**

- **Activate As**

  Like all processes, you can activate the process as a template and use it as an advanced starting point for processes that follow a similar pattern.

- **Define Process Arguments**

  In this area, you'll specify any data that the action expects to start and what data will be passed out of the action. More information: Define process arguments

- **Add Stages, Conditions and Actions**

  Like other processes, you specify what actions to perform and when to perform them. More information: Add stages, conditions and actions

Define process arguments

When a developer uses a message, they might begin with some data that they can pass into the message. For example, to create a new case row, there might be the case title value that is passed in as the input argument.

When the message is finished, the developer might need to pass some data that was changed or generated by the message to another operation in their code. This data is the output argument.

Both input and output arguments must have a name, a type, and some information about whether the argument is always required. You can also provide a description.

The name of the message and the information about all the process arguments represent the *signature* for the message. After an action is activated and is being used in code, the signature must not change. If this signature changes, any code that uses the message will fail. The only exception to this might be changing one of the parameters so that it is not always required.

You can change the order of the arguments by sorting them or moving them up or down because the arguments are identified by name, not by the order. Also, changing the description won't break code using the message.

Action process argument types

The following table describes the action process argument types.

Expand table

Type	Description
Boolean	A `true` or `false` value.
DateTime	A value that stores date and time information.
Decimal	A number value with decimal precision. Used when precision is extremely important.
Table	A row for the specified table. When you select **Table**, the drop-down list is enabled, allowing you to select the table type.
entityCollection	A collection of table rows.
entityReference	An object that contains the name, ID, and type of a table row that uniquely identifies it. When you select tableReference, the drop-down list is enabled and allows you to select the table type.
Float	A number value with decimal precision. Used when data comes from a measurement that isn't absolutely precise.
Integer	A whole number.
Money	A value that stores data about an amount of money.
Picklist	A value that represents an option for an OptionSet attribute.
String	A text value.

> **Note**
>
> **entityCollection** argument values can't be set in the user interface for conditions or actions. These are provided for use by developers in custom code. More information: Create your own messages

Add stages and steps

Actions are a type of process very similar to real-time workflows. All the steps that can be used in real-time workflows can be used in actions. For information about the steps that can be used for both real-time workflows and actions, go to Workflow stages and steps.

In addition to the steps that can be used for real-time workflows, actions also have the **Assign Value** step. In actions, these can be used only to set output arguments. You can use the form assistant to set output arguments to specific values or, more likely, to values from the row that the action is running against, rows related to that row with a many-to-one relationship, rows created in an earlier step, or values that are part of the process itself.

Managed properties

Custom process actions have two relevant managed properties

Is Customizable

The **Is Customizable** managed property controls whether on not someone who installs a managed solution containing the custom process action can edit or delete it. If wish for the custom process action to be edited or deleted when included in a managed solution, you should set this property to true.

Is Custom Processing Step Allowed for Other Publishers

The **Is Custom Processing Step Allowed for Other Publishers** managed property controls whether third party plug-in developers can register plug-in steps on the message created by the custom process action. When this is true, plug-ins registered by anyone on this message will run and can modify the behavior of the custom process action. When false, only plug-ins steps registered within a solution from the same solution publisher will run.

Next steps

Use actions
Invoke custom actions from a workflow
Monitoring real-time workflows and actions

Invoke custom process actions from a workflow

Article • 02/15/2022

Workflows have numerous capabilities supporting business scenarios. Calling basic data operation actions for a row, such as create, update, and delete, from within a workflow solves quite a few business scenarios. However, if you couple the capabilities of the workflows with the power of the custom processs actions invoked directly from within a workflow, you add a whole new range of business scenarios to your application without needing to write code.

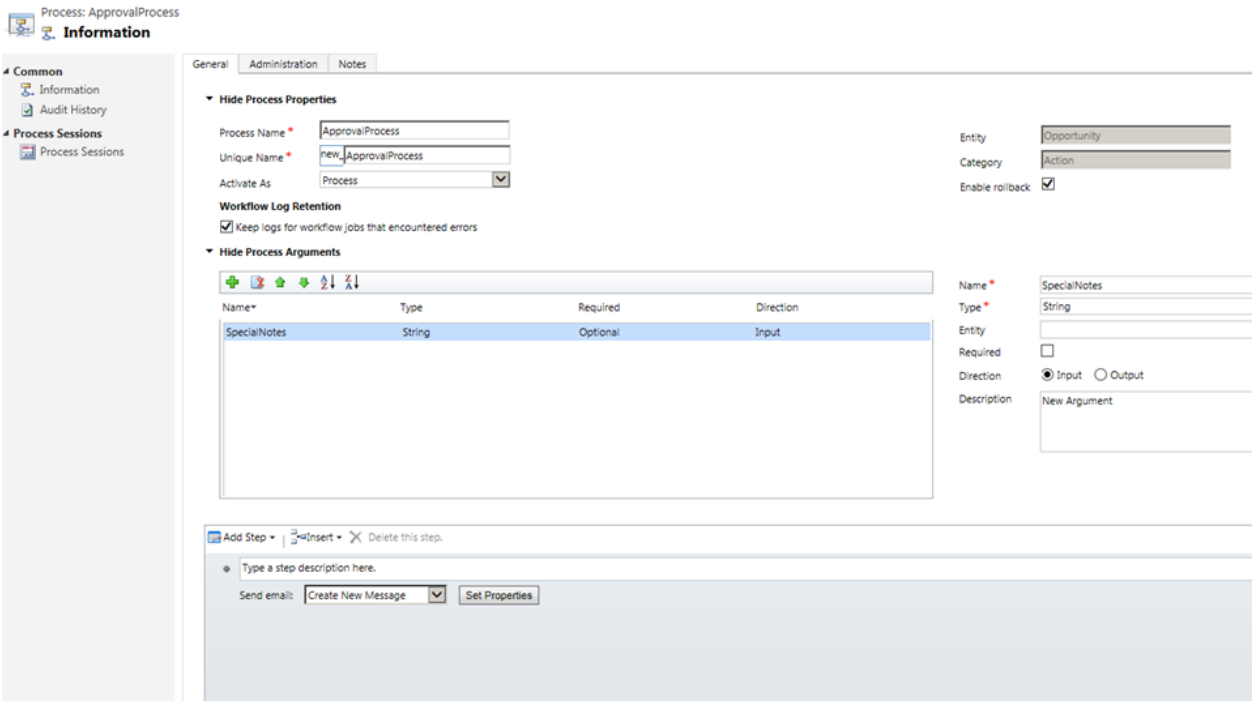
Let's look at the scenario in which a custom process action is invoked from a workflow. We'll invoke a custom process action to request the manager's approval when a discount for a particular opportunity exceeds 20 percent.

Example: Create a custom process action using the opportunity table

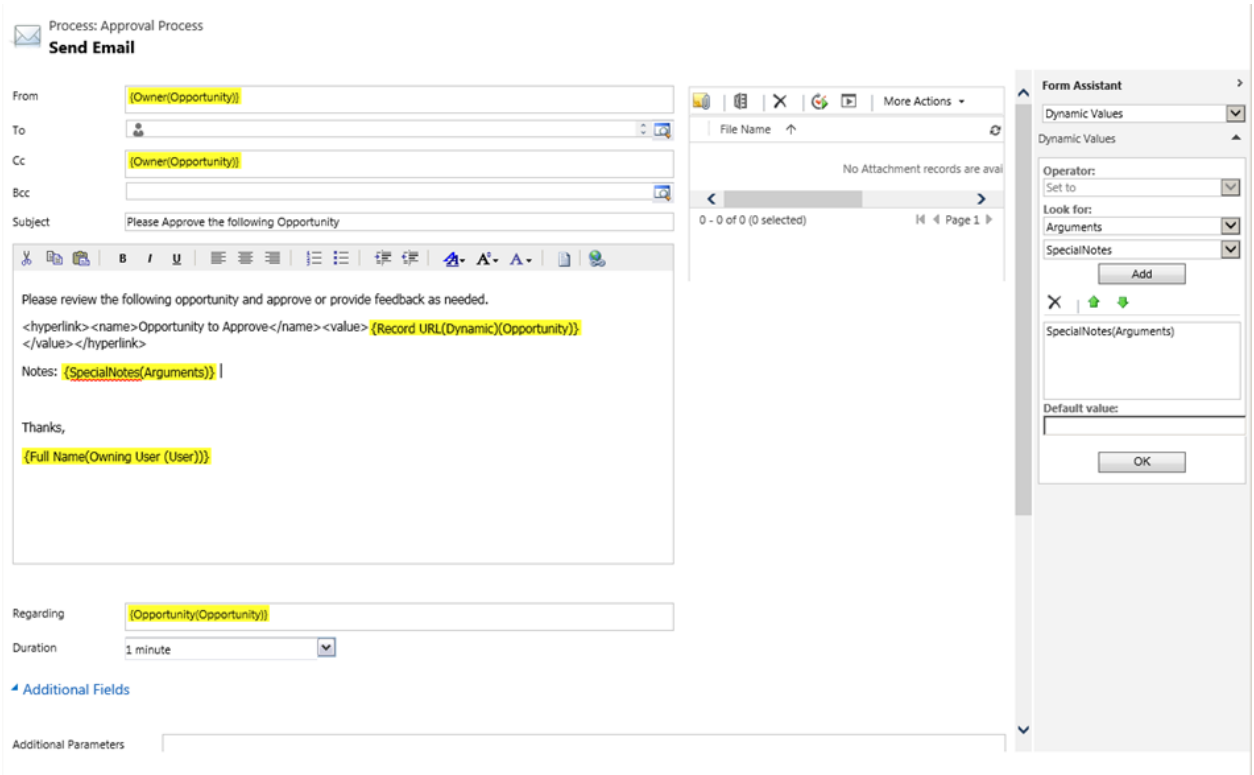
1. In solution explorer, select **Processes**.

2. On the Nav bar, choose **New**. Give the process a name and choose the **Action** category.

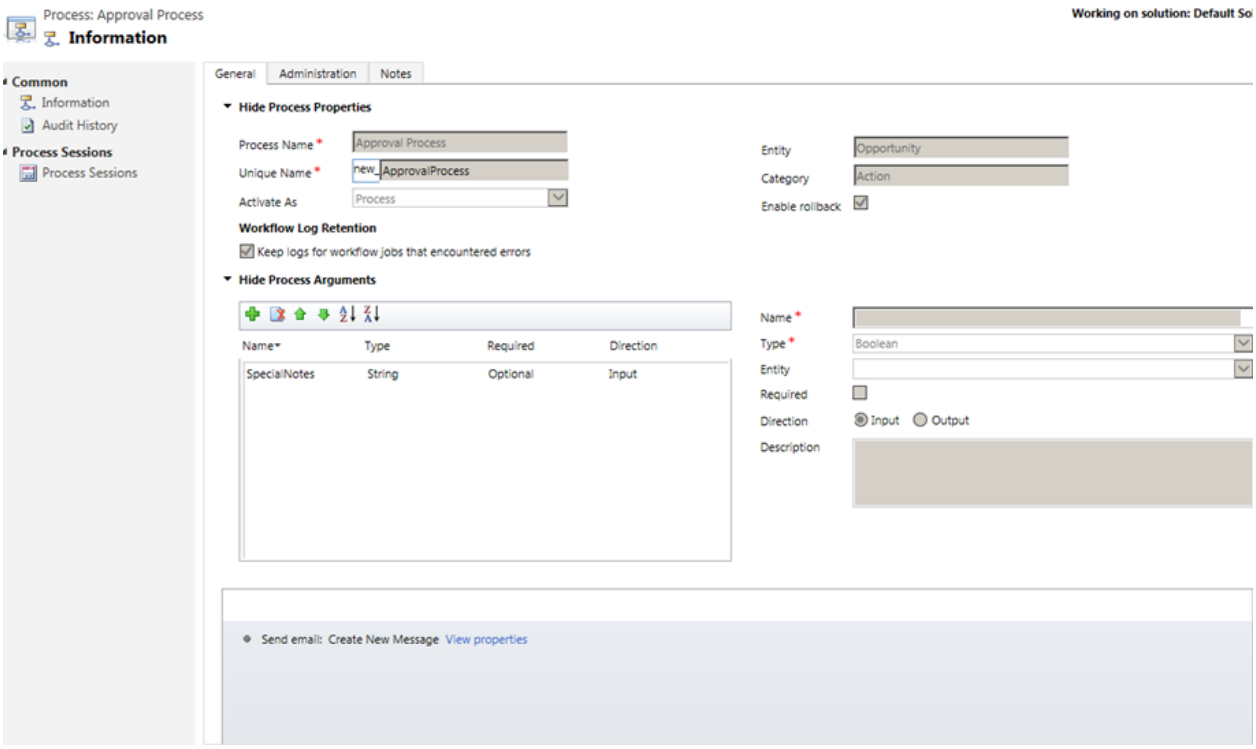
To request an approval for the discount, we're using a custom action called **Approval Process**. We added an input parameter, **SpecialNotes**, and a **Send email** step to create a new message and send a request for the manager's approval, as shown here.



To configure the email message, choose **Set Properties**. When the form opens, use the **Form Assistant** to add special notes and other information to the email, as highlighted on the screenshot. To add the special notes, place the cursor where you want the notes to appear in the message, and then, in the **Form Assistant**, under **Look for**, choose **Arguments** in the first drop-down list and choose **SpecialNotes** in the second drop-down list, and then choose **OK**.



Before you can invoke the custom process action from a workflow, you have to activate it. After you have activated the action, you can view its properties by choosing **View properties**.

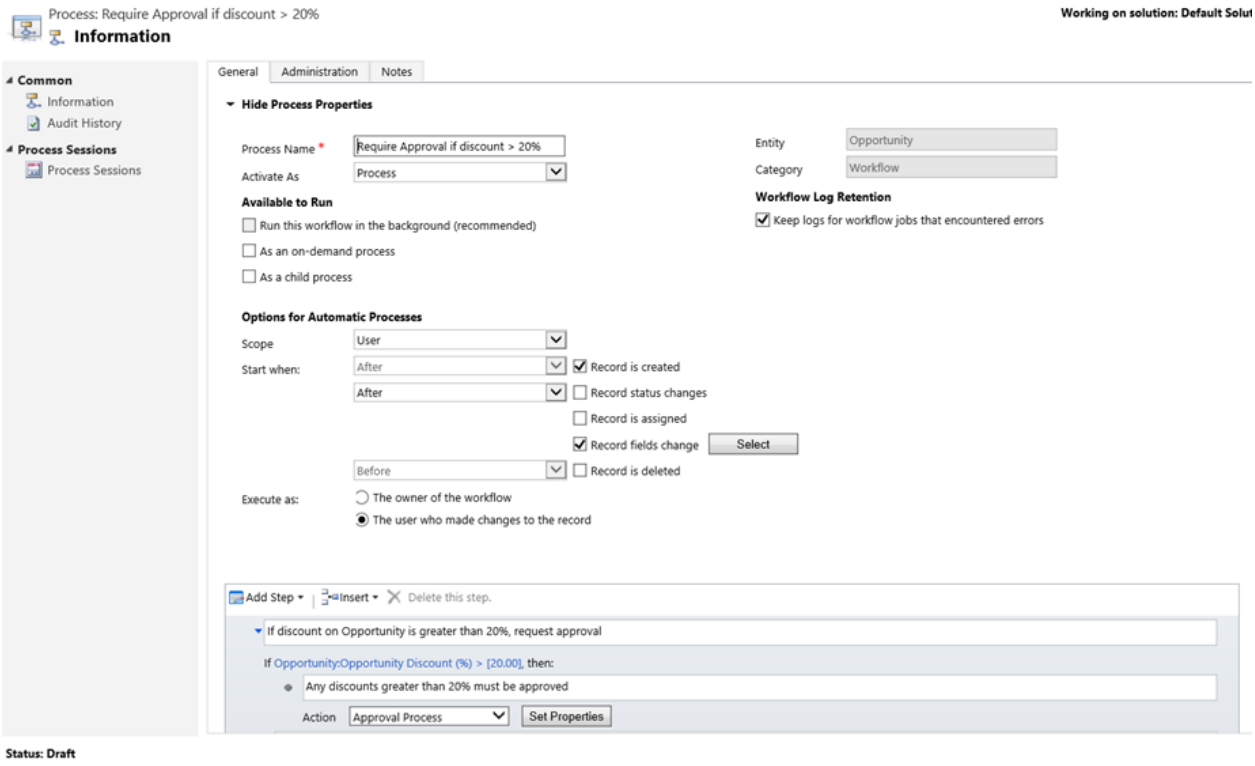


Invoke a custom process action from a workflow

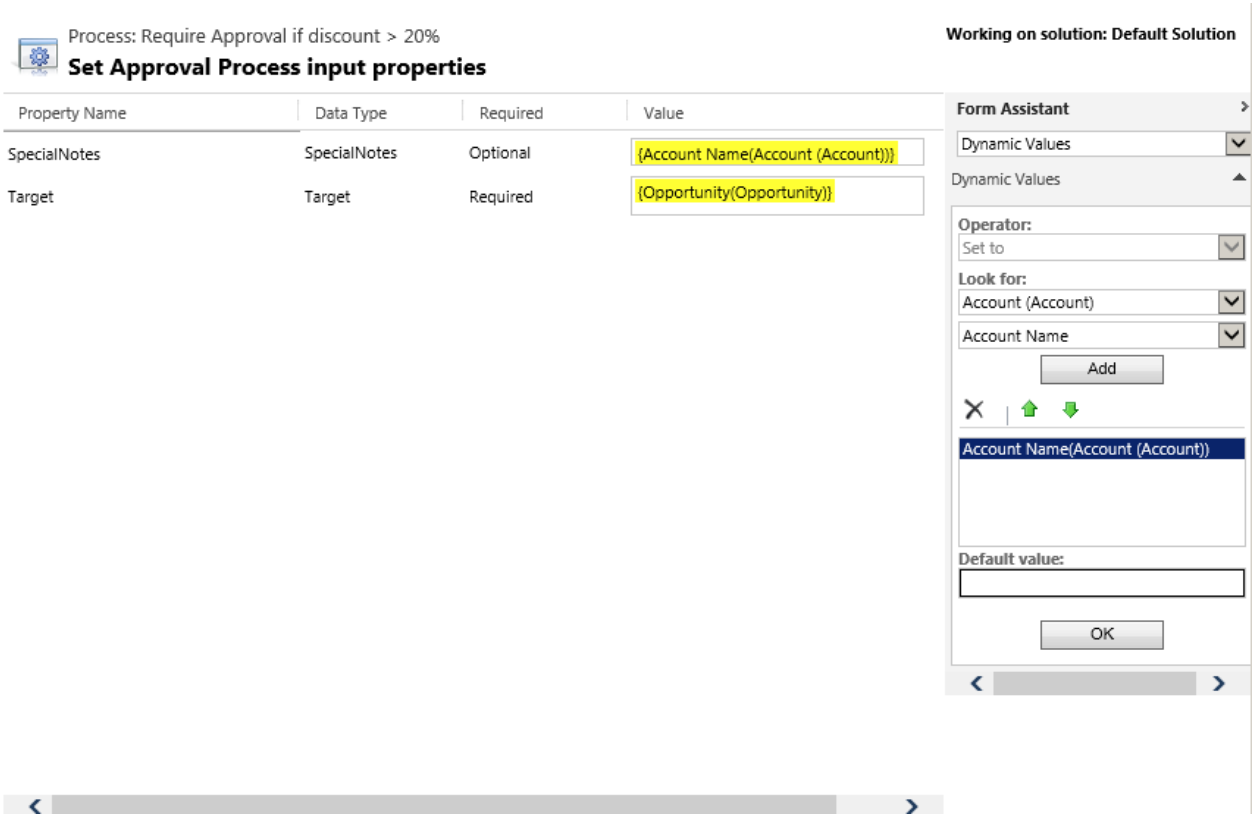
1. In solution explorer, select **Processes**.

2. On the Nav bar, choose **New**. Give the process a name and choose the **Workflow** category.

We created a workflow that invokes the **Approval Process** custom action whenever the manager's approval for a discount of more than 20 percent for an opportunity is required.



You can set the custom process action's input properties by choosing **Set Properties**. We added a name of the account related to the opportunity in the special notes. In the **Form Assistant**, under **Look for**, choose **Account** in the first drop-down list, choose **Account Name** in the second drop-down list, and then choose **OK**. The **Target** property is required and it is populated by the system. The **{Opportunity(Opportunity)}** in the **Target** property is the same opportunity that the calling workflow is running on. Alternatively, you can choose a specific opportunity for the target property by using lookup.



Test Dataverse extensions with Test Engine (deprecated)

> ⓘ Note
>
> **Test Engine is deprecated** and will be removed in a future release. Use Power Platform Playwright samples for test automation capabilities in Power Platform and Dynamics 365 services.

This article explains how to create integration tests with Dataverse. This functionality applies the Power Fx capability to connect with Dataverse tables.

Connecting to Dataverse

To enable Dataverse integration, add the `enableDataverseFunctions` parameter in your testSettings extensionModules :

YAML

```yaml
testSettings:
  extensionModules:
    enable: true
    parameters:
      enableDataverseFunctions: true
```

When you enable Dataverse integration, you're required to open Azure CLI with a user or service principal that has rights to the Dataverse environment. You can use the command: az login More information: Sign in with Azure CLI.

The Dataverse API URL used for integration is obtained either from the host domain name of the pac test run `--domain` parameter or by defining an PowerShell environment variable named `DATAVERSE_URL`.

Enable preview

Currently, all Dataverse extension testing capabilities require using Preview functions. To enable these, add `Preview` to the `allowPowerFxNamespaces` list in extensionModules.

YAML

```yaml
testSettings:
  extensionModules:
    enable: true
    allowPowerFxNamespaces:
    - Preview
    parameters:
      enableDataverseFunctions: true
```

Resetting the current state

Define the `onTestCaseStart` event handler in your authored test yaml to perform common actions before each test case. For example:

YAML

```yaml
testSuite:
  testSuiteName: Dataverse tests
  testSuiteDescription: Validate Power Fx can be used to run Dataverse integration
tests
  persona: User1
  appLogicalName: N/A
  onTestCaseStart: |
    = ForAll(Accounts, Remove(Accounts, ThisRecord))
```

In this example, the `ForAll` is used to iterate over a table and perform an action on each record. The `Remove` function is then used to delete each record from a table to ensure an empty table before each test is executed.

Functions of interest

These functions are useful when working with Dataverse:

- Collect: Adds records to a table.
- CountRows: Returns the number of records in a table.
- First: Returns the first record in a table.
- ForAll: Iterates over a table and performs an action on each record.
- Patch: Update an existing record.
- Remove: Deletes a specific record from a table.

Considerations

Take these considerations into account as you write test steps:

- Defaults isn't supported. As a result Power Fx examples like `Patch(Accounts, Defaults(Accounts), {name:"test"})` aren't supported.
- You can use `Collect` to an alternative to `Patch` with the `Defaults` function

Running Dataverse tests

You can use the Power Fx provider (`powerfx`) to run tests that interact with Dataverse without the need for a canvas or model-driven app interface. This is useful for testing business logic, plugins, flows, and other backend services.

PowerShell

```
pac test run `
   --provider powerfx `
   --test-plan-file your-testplan.te.yaml `
   --tenant your-tenantid-guid-value `
   --environment-id your-environmentid-guid-value `
   --domain "https://contoso.crm.dynamics.com"
```

AI Integration

Using the open source Power Apps Test Engine compiled in `Debug` mode together with the appropriate test settings will enable the Preview.AIExecutePrompt function in your tests.

Add the following to your test settings to enable AI integration:

YAML

```
testSettings:
  extensionModules:
    enable: true
    parameters:
      enableDataverseFunctions: true
      enableAIFunctions: true
```

Example usage of the AI function:

PowerApps Formula

```
Assert("Paris" = Preview.AIExecutePrompt("Country Capital", { Country: "France"
}).Text)
```

Last updated on 04/18/2026

Solutions in Power Apps overview

Solutions are used to transport apps and components from one environment to another or to apply a set of customizations to existing apps. A solution can contain one or more apps as well as other components such as site maps, tables, processes, web resources, choices, flows, and more.

Solutions are the mechanism for implementing application lifecycle management (ALM) in Power Apps and other Power Platform products, such as Power Automate.

> **Note**
>
> To learn more about the solution concepts and how solutions are used for application lifecycle management, go to Overview of ALM with Microsoft Power Platform in the Power Platform ALM guide.

This section focuses on the **manual** tasks that app makers need to perform while working with solutions in Power Apps.

Get started: solution concepts

Before you work with solutions, it's important that you get acquainted with the following solution concepts:

- Two types of solutions (managed and unmanaged)
- Solution components
- Lifecycle of a solution (create, update, upgrade, and patch a solution)
- Solution publisher
- Solution and solution component dependencies

To learn more, go to Solution concepts in the Power Platform ALM guide.

Default solutions

Power Apps provides you with the following default unmanaged solutions:

- **Common Data Service Default Solution**. This solution is available for makers to use by default for their customizations in an environment. The Common Data Service Default Solution is useful when you want to evaluate or learn Power Apps. However, we recommend that app makers work in their own unmanaged solutions.

- **Default Solution**. This is a special solution that contains all components in the system. The default solution is useful for discovering all the components and configurations in your system.

However, we recommend that you create a solution to manage your customizations. More information: Use a solution to customize in the Power Platform ALM guide.

Managed properties

You can control which of your managed solution components are customizable by using managed properties. We recommend that you set managed properties so that your managed components can't be modified. This helps protect your solution from modifications that might cause it to break after it's imported into another environment, such as test or production.

More information: Managed properties in the Power Platform

Work with solutions in Power Apps

Within Power Apps, you can view a list of solutions by selecting **Solutions** in the left navigation. You can perform these solution tasks:

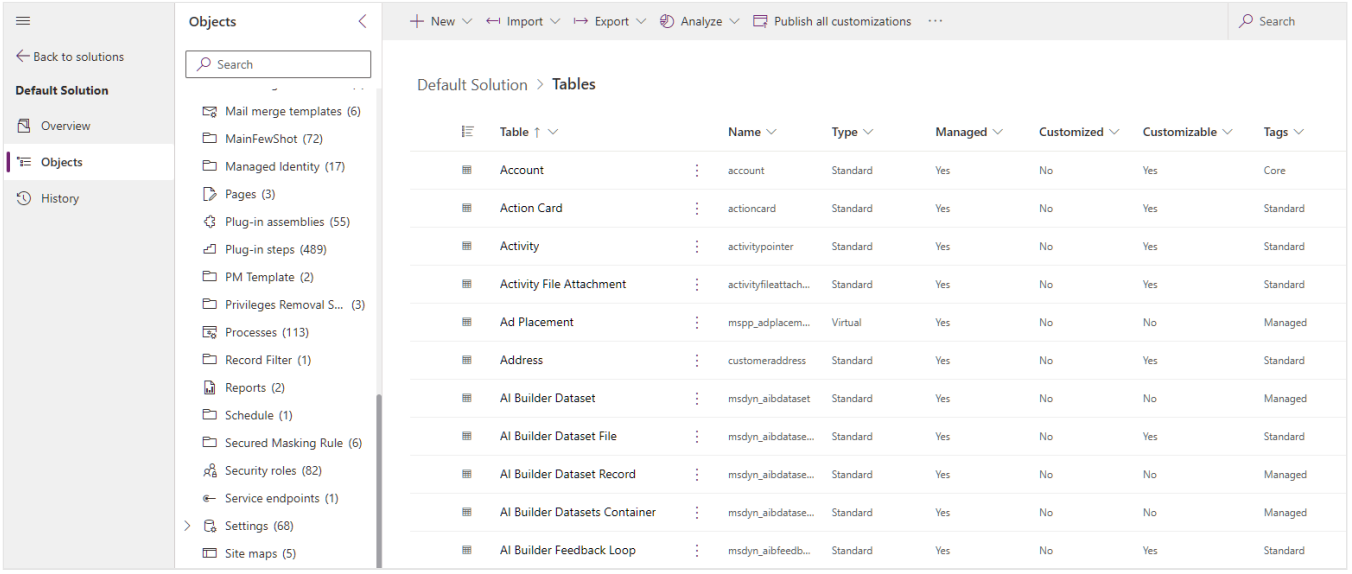
- **New solution**: To locate and work with just the components you've customized, create a solution and do all your customization there. Then, you can easily distribute your solution to other environments. More information: Create a solution
- **Import solution**: Import a solution into your environment. More information: Import solutions
- **Open AppSource**: Microsoft AppSource is where you can go to get solutions tailored to your industry that work with the products you already use.
- **Publish all customizations**: Publish all active customizations in your environment.
- **Set preferred solution**: Set your preferred solution where all solution components are created in.
- **See history**: View details about solution operations over time, such as import, export, and uninstall. More information: View the history of a solution
- **Connect to Git**: Source control integration allows development teams to sync solutions and solution objects across one or more Dataverse environments using an Azure DevOps Git repository. More information: Overview of Git integration in Power Platform
- **Switch to classic**: Open the classic solution explorer.

When you select a solution, additional tasks become available on the command bar. These tasks include:

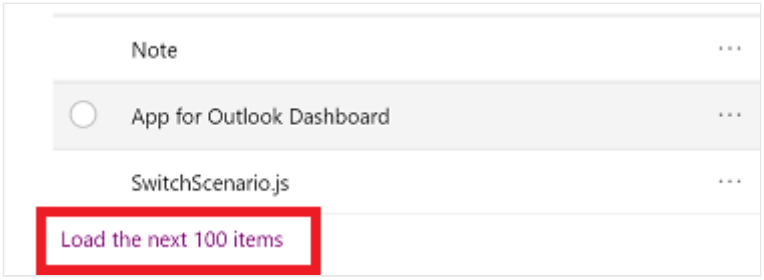
- **Edit**: Edit the solution properties, such as adding or removing solution components.

- **Delete**. Delete the selected solution.
- **Create a plan**. Use Plan designer to create a plan for your existing solution. Plan designer generates a detailed document that describes your solution. The plan covers the business problem, user requirements like user roles and stories, the data model, and technologies like apps. This feature saves time when you're trying to understand a solution's content and helps makers improve an existing solution. More information: Create a plan from a solution
- **Export solution**: Export the solution to a file that can be imported into another environment. More information: Export solutions
- **Deploy**. Use pipelines in Power Platform to deploy solutions to test and production environments. More information: Overview of pipelines in Power Platform
- **Solution checker**: Run or review results of solution checker for this solution.
- **Show dependencies**: View the solution components that have a dependency on another component. More information: View solution dependencies for solutions that would block uninstall of this solution.
- **Set preferred solution**. Use the selected solution as your preferred solution. The preferred solution is where, if not already working in the context of a solution, all your solution components are maintained. More information: Preferred solution
- **See history**. View details about solution operations over time, such as import, export, and uninstall. More information: View the history of a solution
- **Publish to Catalog**: Publishes the solution to the catalog, making it available for other makers in your organization to use. This is useful for sharing solutions that you want others to be able to import and use in their environments. More information: Catalog in Power Platform
- **Apply Upgrade**: Appears when you select a solution. Apply a pending upgrade that has been initiated for a managed solution.

From the **Solutions** area, open a solution to view all of its objects.



Browse through all the objects in a solution by scrolling through the items. If there are more than 100 items in the list, you can select **Load the next 100 items** to see more.



Columns can be sorted and filtered by selecting the column header.

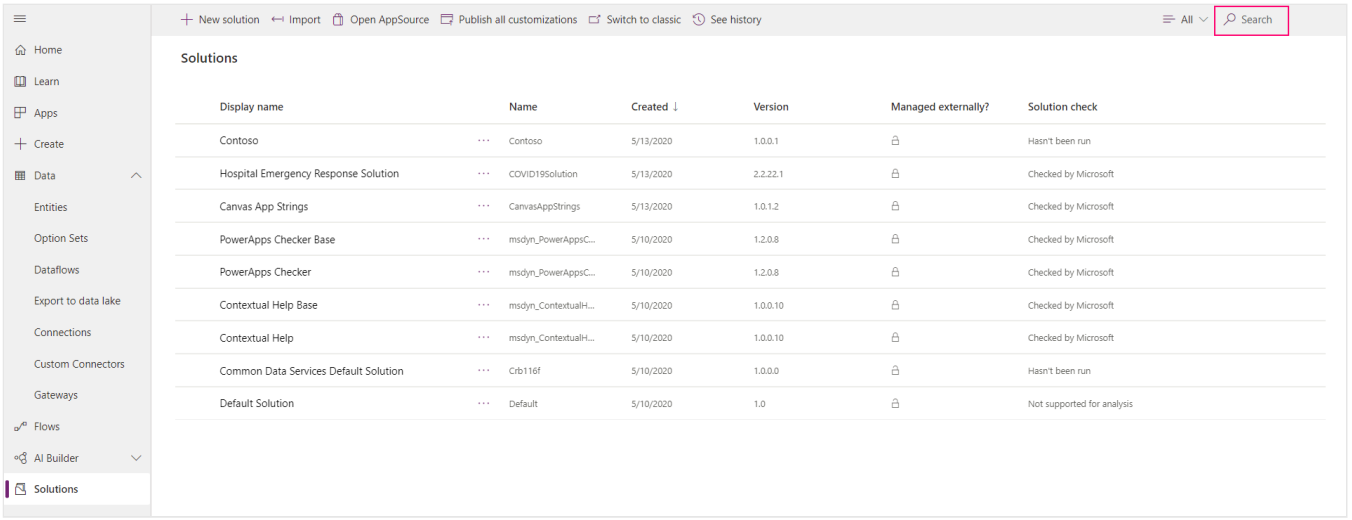
Column headers include:

- **Managed**: The solution object is from a managed solution. You can inspect the object by selecting the item and then select **Advanced** > **See solution layers**.
- **Customizable**: The component is available to be customized.
- **Customized**: This indicates that the object is an unmanaged object, or a managed object with an unmanaged customization layer. You can use this column to quickly locate the unmanaged changes you have for components in the solution.
- **Owner**: If the solution object supports user and team ownership, the current owner is displayed.
- **Status**: If the solution object supports state management (on/off, enabled/disabled, active/inactive), the status is displayed.

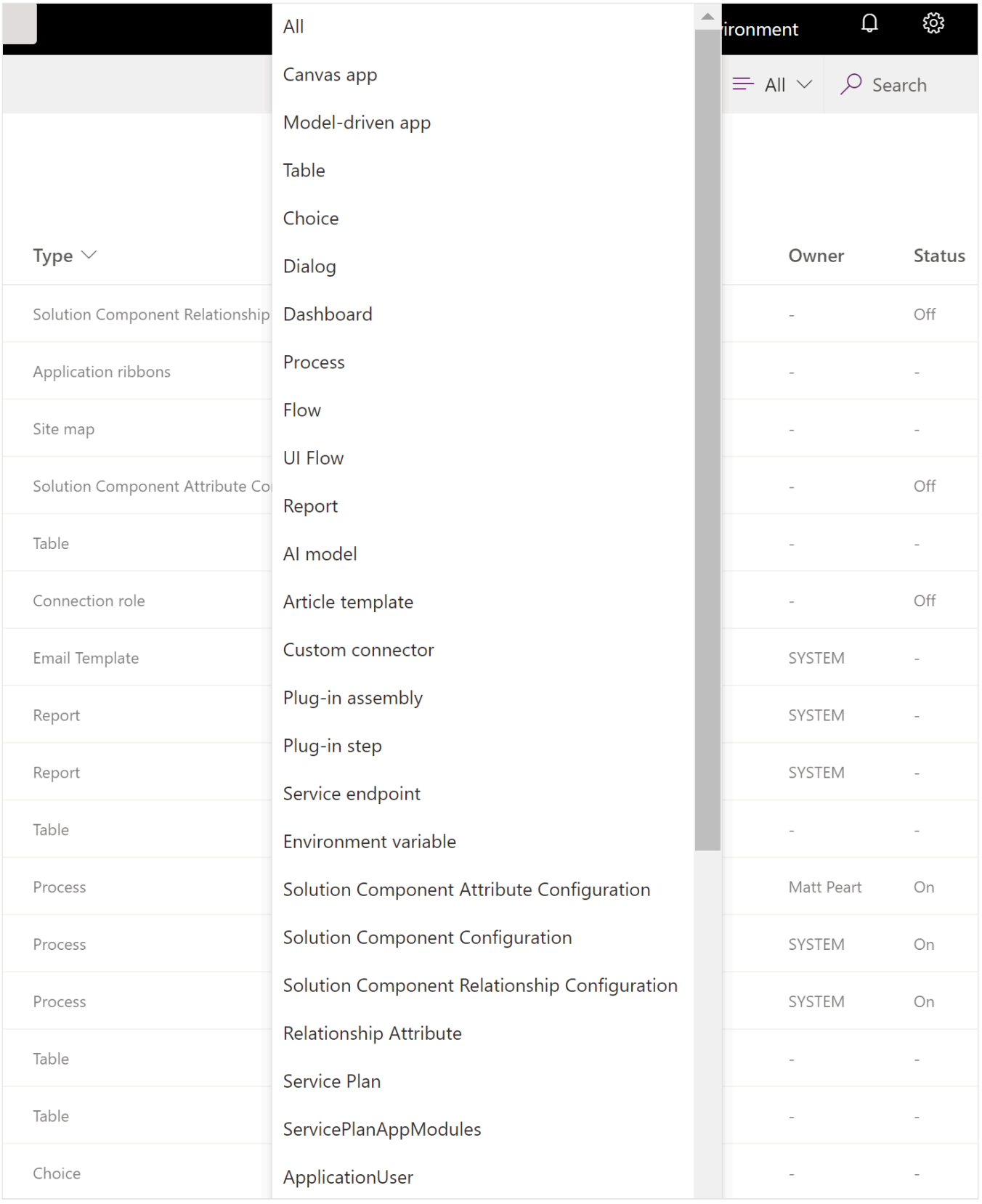
> **Tip**
>
> You can quickly locate all solution objects that are unmanaged or have unmanaged customizations by selecting the default solution and filtering on the **Customized** column.

Search and filter in a solution

Search for a specific component by its name.

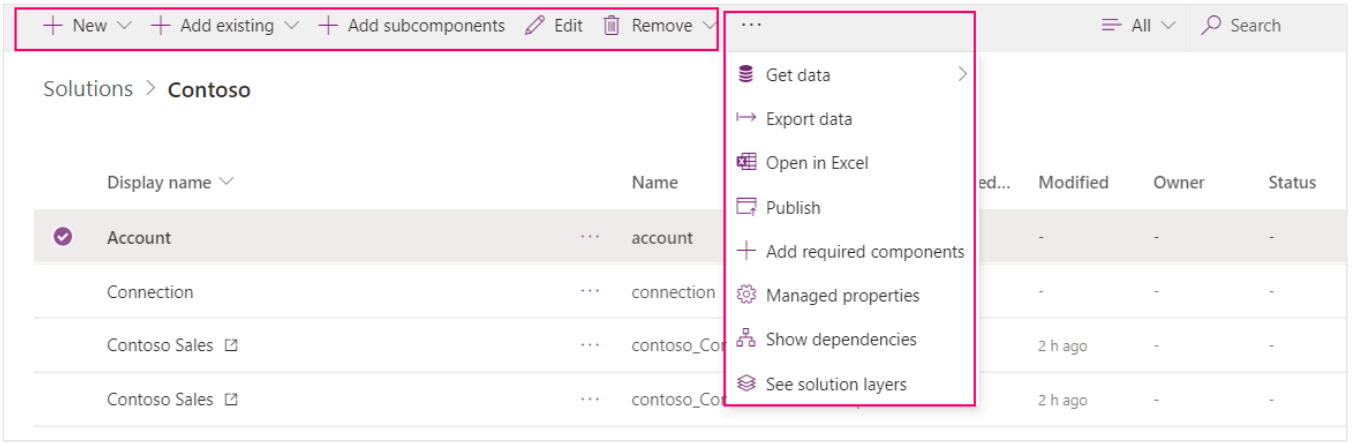


Or filter all items in the list by the component type.

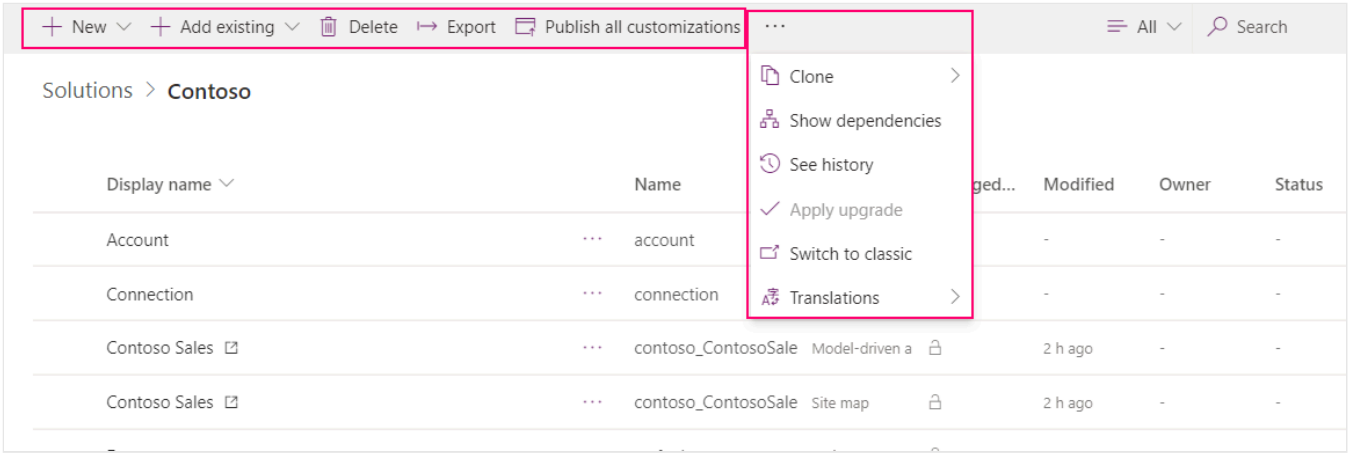


Contextual commands

As you select each component, the actions available in the command bar changes depending on the type of the component you have selected and if the solution is the default or a managed one.



When you don't select any component, the command bar shows actions applied to the solution itself.



With solutions that are unmanaged or the default one, you can use the **New** or **Add Existing** command to create or add different types of components. More information: Add solution components

> **Note**
>
> You can't add components to a managed solution. When you try to, you'll get the following message:
> `"You cannot directly edit the components within a managed solution. You'll need to add it to another unmanaged solution that you've created to customize the component. The component might not be customizable."`

Additional privileges required

Some components might require certain Dataverse privileges for users to run the component when the component is imported into the environment from a solution.

Flows

To use or run a flow from a canvas app that is included in a solution, you must have permissions to that flow through someone sharing ownership or run permissions. When an app in a solution is shared with a set of users, the flows must also be explicitly shared.

More information: Security roles and privileges

Use pipelines in Power Platform to deploy solutions

Easily deploy solutions to test and production environments using pipelines in Power Platform. Once pipelines are in place, makers can initiate in-product deployments with a few clicks. Makers do so directly within their development environments. More information: Overview of pipelines in Power Platform

Known limitations

The following limitations apply to the use of canvas apps, flows, and custom connectors in solutions.

- If you encounter a canvas app publishing error: **the app has connections to flows that are no longer in the environment**, remove any deleted flows from the app. Then save and publish the app.
- Canvas apps shared with **Everyone** that go through environment backup and environment restore operations aren't shared with **Everyone** in the restored environment. Notice that the canvas app can be shared with a security group, and the app in the restored environment is shared with that security group.

To learn more about customizing the individual components in a solution, go to the following articles:

- For table, table relationships, column and message customizations, go to Metadata.
- For table forms, go to Forms.
- For processes, go to Processes.
- For business rules, go to Business Rules.

Troubleshooting solutions

For known issues and information about how to troubleshoot working with solutions, go to Manage apps and solutions in the Power Apps Troubleshooting documentation.

Next step

Create a solution

Last updated on 01/09/2026

Solution view

Article • 11/06/2024

The modern solution view makes it easier for you to work with solution objects and subcomponents in your solutions.

Open solution view

To open the solution view, sign in to Power Apps (make.powerapps.com) on the left navigation pane, select **Solutions**, and then open the solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.

The solution objects and available actions are displayed.

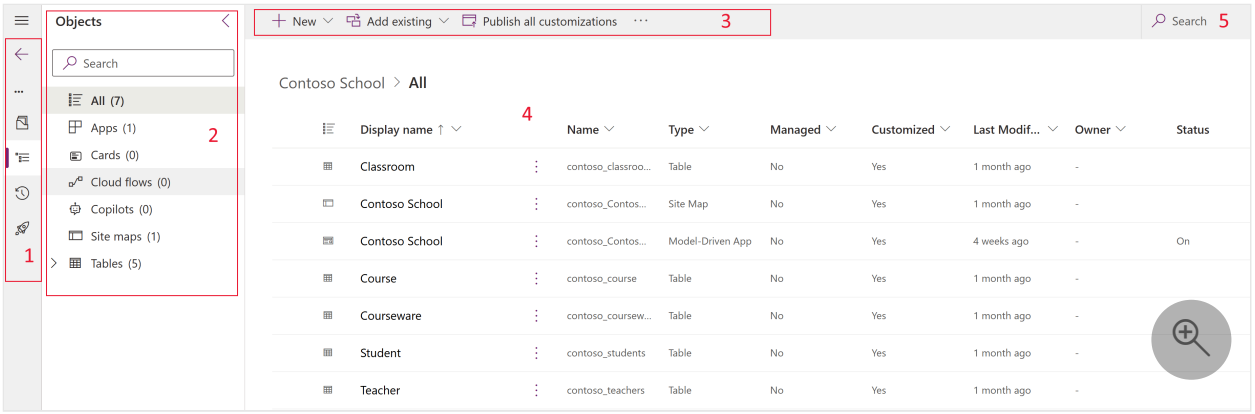
1. Panes - The left navigation pane consists of the following areas:

    - **Overview**. Provides details about the solution, such as display name, name, created on date, version, managed or unmanaged, publisher, description, and whether the solution is a patch. Also, solution health information and solution actions available on the command bar such as export, clone, apply upgrade, and translations.
    - **Objects**. Presents a tree view of all objects within the solution. Select an object from the objects list (4) to view or edit it.
    - **History**. Presents the solution operations completed on the solution. An operation can be a solution import, export, or uninstall. The solution history displays information such as solution version, solution publisher, type of operation, operation start and end time, and operation result status.
    - **Pipelines**. Deploy solutions using pipelines in Power Platform. More information: Set up pipelines in Power Platform
    - **Source control** (not shown in image). Connect to Git to source your Power Platform solution objects in an Azure DevOps Git repo. More information: Overview of Dataverse Git integration

2. Tree view. From the **Object** pane, the tree view displays a list that you can browse to find an object to open it or one of the object's subcomponents. Search for objects and subcomponents that are in the solution.
3. Command bar. Contextual command bar to perform actions on the solution, objects, or subcomponents.
4. Objects. Displays information and solution components that can be viewed and/or opened for editing of the selected object from the **Objects** pane. Add existing

objects and subcomponents or create new ones for unmanaged solutions. More information: Objects list

5. Search. Use to filter the list of subcomponents for the object that's currently selected. Filter on any subcomponent property. For example, filter on only **Lookup** data types or whether a column is **Required** by entering those strings in the **Search** box.



Objects list

The **Objects** list in solution view has the following columns:

- **Display name**. The friendly name of the solution object.
- **Name**. The unique or logical name of the solution object that includes the solution publisher prefix.
- **Type**. Solution object type, such as a table, web resource, model-driven app, canvas app, and so on.
- **Managed**. Indicates whether the solution object is managed or unmanaged. Managed solutions aren't typically customizable.
- **Customized**. Unmanaged solution components are customized while managed solution objects typically shouldn't be. If a managed solution object is customized, you can remove the customization. More information: View solution layers for a component
- **Last modified**. The date the solution object was last changed.

See also

View the history of a solution
View dependencies for a component
Create a solution

Feedback

Was this page helpful? Yes No

Provide product feedback

Create a solution

To locate and work with just the components you customize, create a solution and do all your customization there. Then, always remember to work in the context of the custom solution as you add, edit, and create components. This makes it easy to export your solution so that it can be backed up or imported to another environment.

> **Note**
>
> Consider specifying a custom solution as you preferred solution. This way you can control where your solution components are stored when you don't specifically create them in the context of a custom solution. More information: Set the preferred solution.

To create a solution:

1. Sign in to Power Apps and select **Solutions** from the left navigation. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **New solution** and then complete the required columns for the solution.

Expand table

Field	Description
Display Name	The name shown in the list of solutions. You can change this later.
Name	The unique name of the solution. Must only contain letters, numbers, and underscores. The name is generated from the allowed characters from the value you enter in the **Display Name** column. You can edit this before you save the solution, but after you save the solution, you can't change it.
Publisher	You can select the default publisher or create a new publisher. We recommend that you create a publisher for your organization to use consistently across your environments where you'll use the solution. See Solution publisher later in this article.
Version	Enter a number for the version of your solution. This is only important if you export your solution. The version number is included in the file name when you export the solution.
Set as your preferred solution	Using a preferred solution is a way you can control where your solution components are stored when you don't specifically create them in the context of a custom solution. When your preferred solution isn't set, by default, the Common Data Services Default Solution is your preferred solution. If Common Data Services Default Solution isn't available, the solution named Default

Field	Description
	Solution is used as your preferred solution. More information: Set the preferred solution
Configuration Page	Under **More options** you can, optionally select a web resource to use as a configuration page. The configuration page is typically used by independent software vendors (ISVs) who distribute solutions. When this is set, a new **Configuration** node appears below the **Information** node to display this web resource. Developers use the web resource, which is an HTML page, to include instructions or controls to allow you to set configuration data or launch their solution.
Description	Under **More options** you can optionally add a description of the solution. This is useful if you want to share the solution with others or if you want to remember what the solution is for.

3. Select **Save**.

Add or remove solution objects

After you create your solution, it won't contain any solution objects. You can create new objects to add to the solution or add existing objects to your new solution.

Create objects in a solution

You can use the **New** command to create different types of objects. Selecting **New** and then selecting the object you want takes you to a different create experience depending on the object type that you choose. After you finish creating the object, it's added to the solution.

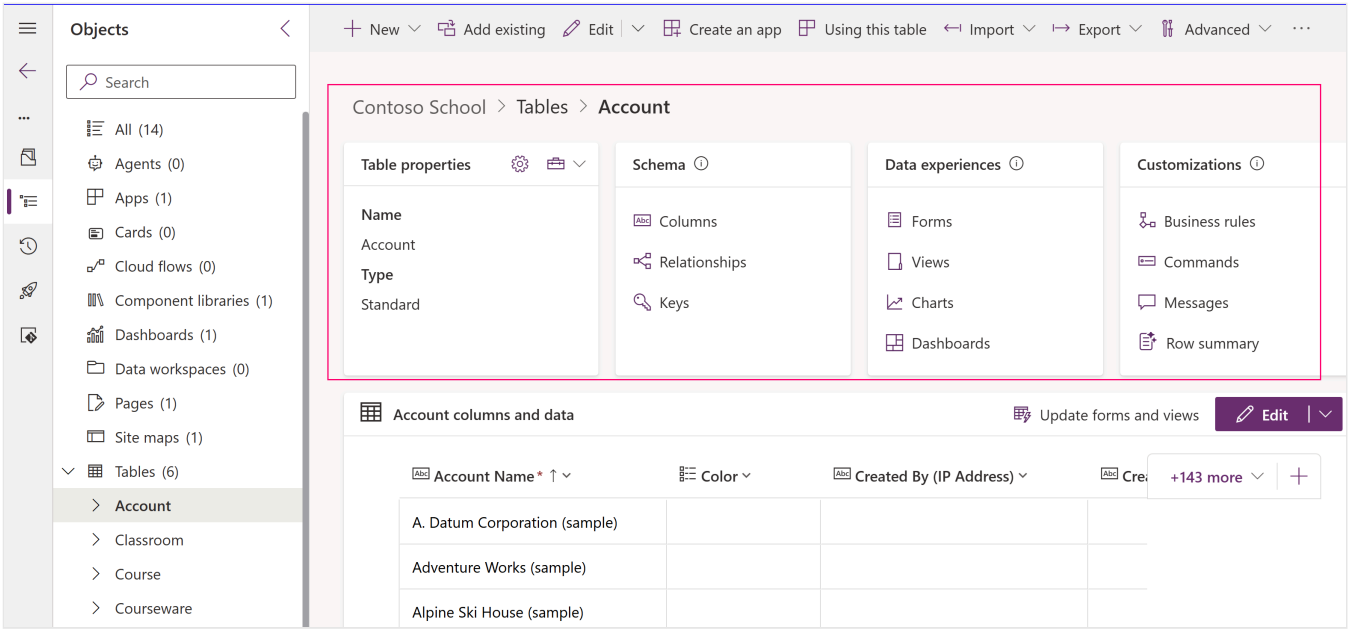
Add an existing object to a solution

With solutions that are unmanaged and not the system default solution, you can use the **Add existing** command to bring in objects that aren't already in the solution.

> ⓘ **Note**
>
> The list of existing objects are different depending on the version and solutions imported in your environment.

Many of the customizations you want to do involve tables. Select **Tables** in the left pane to filter and display a list of all the tables in the current solution. Once you drill into a table, you

can view the objects that are part of the table as shown with the account table in this screenshot.



Remove objects from a solution

You can remove objects from an unmanaged solution. With the solution open, select the object, and then select **Remove from this solution** on the command bar.

When the object is removed, it's no longer part of the solution. However, the object still exists in the environment and can be added to another solution. Once the object is removed from all solutions in the environment, it can be deleted.

> **Important**
>
> Once the object is deleted it can't be recovered and all associated data, such as table records when you delete a custom table are lost. If you want to keep the object, but not have it in the solution, remove it from the solution without deleting it.

To delete an object, select it, and then select **Delete from this environment**.

Publish changes

When you make unmanaged changes in an environment, some objects, such as tables, forms, model-driven apps, site maps, and views are saved in an unpublished state. The publish action promotes these changes to an active state and makes them available to end users and for solution export.

Publish your customizations

1. Select **Solutions** from the left navigation. If the item isn't in the side panel pane, select ... More and then select the item you want.
2. Open the solution that you want to publish.
3. Select **Publish all customizations** on the command bar.

> **Important**
>
> Preparing customizations might take some time. If you see a message that the browser page is unresponsive, wait for the page to become responsive, and don't close it.

Solution publisher

Every app you create or customization you make is part of a solution. Every solution has a publisher. You specify the publisher when you create a solution.

The solution publisher indicates who developed the app. For this reason, you should create a solution publisher that is meaningful. You can view the solution publisher for a solution by selecting the solution, and then select **...** > **Settings** on the command bar from the **Solutions** area in Power Apps. For more information about the solution publisher, go to Solution publisher in the Power Platform ALM guide.

> **Note**
>
> The **Common Data Services Default Solution** is associated with the **Microsoft Dataverse Default Publisher**. The default customization prefix is randomly assigned for this publisher, for example it could be `cr8a3`. This means that the name of every new item of metadata created for your organization in this solution has this prepended to the names used to uniquely identify the items.

Create a solution publisher

1. In Power Apps, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. On the command bar, select **New solution**, in the right pane select **New publisher** under **Publisher**.

3. In the **New Publisher** form, enter the required and optional information:

    - **Display Name**. Enter the display name for the publisher.
    - **Name**. Enter the unique name for the publisher.
    - **Description**. Optionally, you can add a description for the publisher.
    - **Prefix**. Enter the publisher prefix you want.
    - **Choice value prefix**. This column generates a number based on the publisher prefix. This number is used when you add options to choices and provides an indicator of which solution was used to add the option.

4. Select **Save**.

> **Note**
>
> Don't use _upgrade as part of the solution name. _upgrade is an internal reserved word for the solution **upgrade process**.

Change a solution publisher

You can change a solution publisher for an *unmanaged* solution by following these steps:

1. In Power Apps, select **Solutions**, select the solution you want, and then select **...** > **Settings** on the command bar. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. In the **Solution settings** pane, select a different publisher from the **Publisher** drop down list or select **New publisher** to create a new one. More information: Create a solution publisher.
3. Select **Update**.

Use segmented tables in a solution

Use table segmentation so that you only include table objects that are updated when you distribute solution updates. More information: Use segmented solutions in the Power Platform ALM guide

To create a solution with proper table segmentation, start with creating an unmanaged solution and add only the objects that you've updated.

For example, imagine that you've created a new custom table that doesn't exist in any other environment named *Custom table* and also added a new column named *Top Ten* for the account table. To create a solution with table segmentation, follow these steps.

1. Go to Power Apps and then select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.

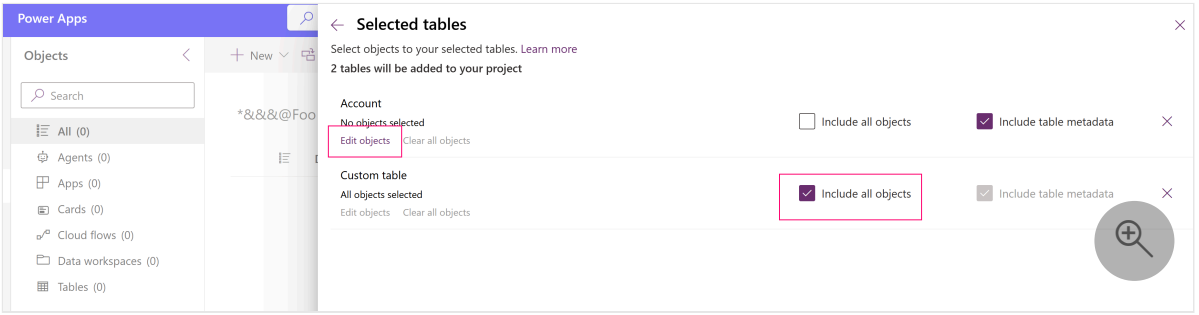
2. Select **New solution** and create a solution. Enter information in the required columns. Select **Create**.

3. Open the solution you created. On the command bar, select **Add existing**, and then select **Table**.

4. In the **Add existing tables** pane, select one or more tables you want to add to the solution. For example, select **Account** and **Custom table**. Select **Next**.

5. In the **Select tables** pane, you can choose from the assets to include:

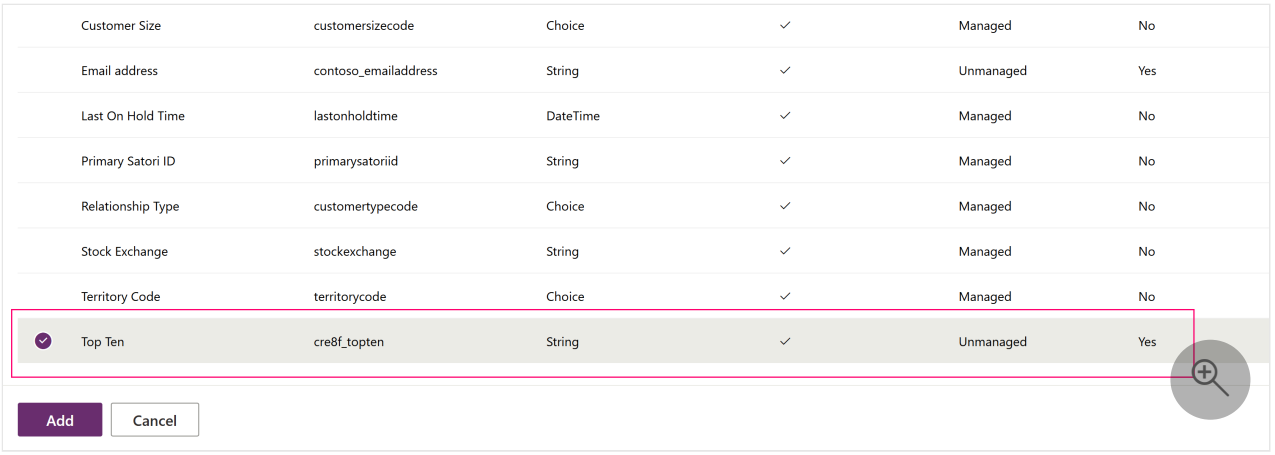
    - **Include all objects**. This option includes all components *and* metadata associated with the table. It can include other tables or table components such as business process flows, reports, connections, and queues. For example, if the custom table has never been imported into the target environment, select this option. In this example, custom table has never been imported into the target environment, so select this option for **Custom table**.
    - **Include table metadata**. This option includes *only* the metadata associated with the table. Metadata includes the table attributes, such as auditing, duplicate detection, or change tracking.
    - **Edit objects**. This option lets you individually select each component that's associated with the table, such as columns, relationships, business rules, views, forms, and charts. Notice that the account table is included in Dataverse and by design also exists in the target environment. Therefore, in our example only the *Top Ten* custom column is new to the account table so you select it after selecting **Edit objects** (in the next step).
    - **Clear all objects**. If you selected objects using the **Edit objects** option, you can clear all the selected objects and start over.

    More information: Why table segmentation in a solution is important

6. If you selected **Edit objects**, an object list pane opens. Here you can select the specific objects you want to include in the solution. In our example you selected the **Account** table earlier, so now you select the **Top Ten** column since that's the only new or edited

component for the table, and then select **Add**.



7. Select **Add** to add the objects to the solution.

Why table segmentation in a solution is important

When you add an existing system table, such as account or contact, or a custom table to a solution for a table that's already been imported into your downstream environments, the best practice is to *only add the table objects that were updated* in your solution. With solution segmentation, you export solution updates with selected table assets, such as table columns, forms, and views, rather than entire tables with all the assets. This avoids unnecessary layers that hinder other solutions from being effective and unnecessary dependencies on other solutions. The system automatically selects **Include all objects** if the table is unmanaged, and **Include table metadata** if there's an unmanaged layer on the table. The system also preselects the table assets that are unmanaged or have unmanaged customizations automatically. You can select the **Edit objects** link to review that selection before you select **Add** to complete the process.

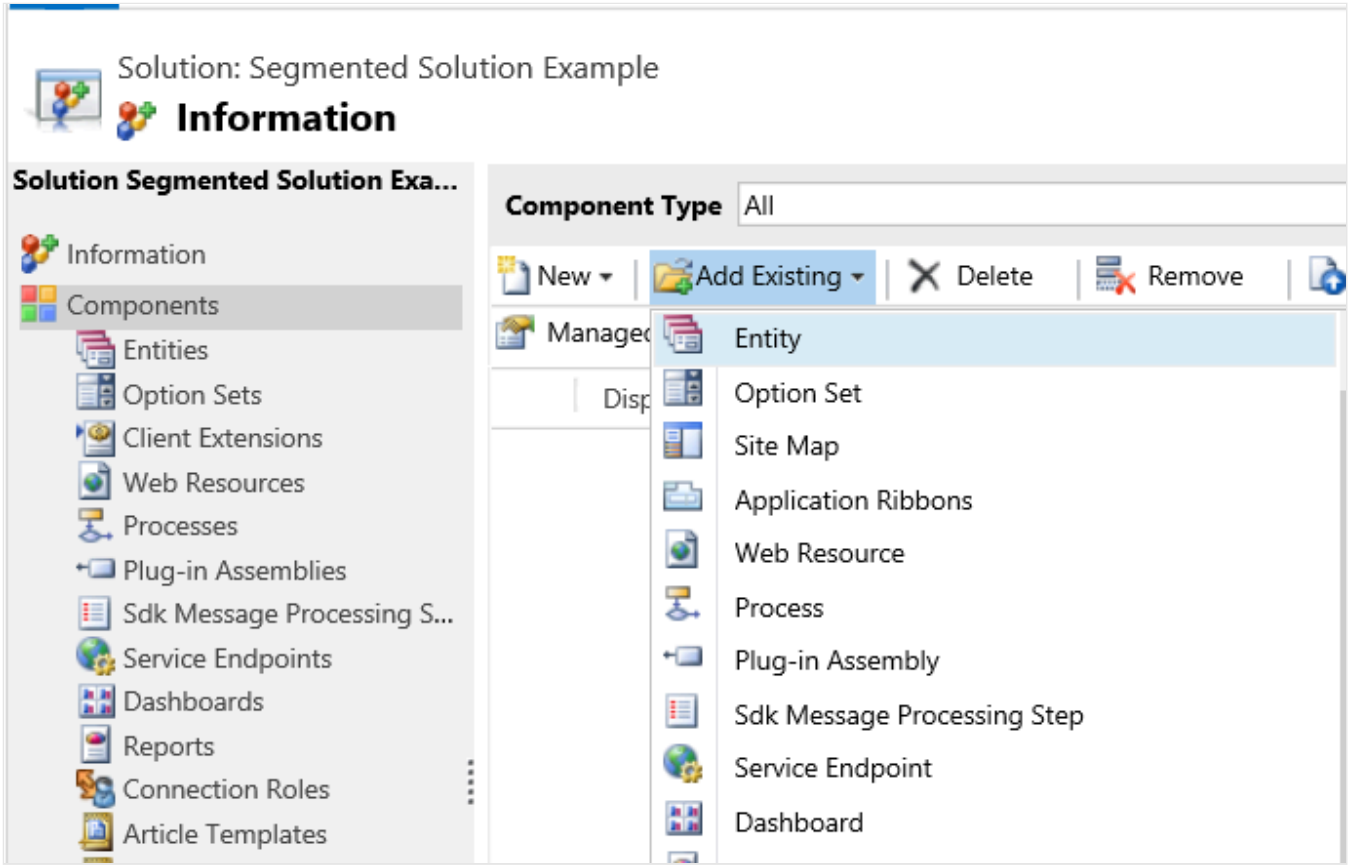
> **ⓘ Important**
>
> Don't segment tables that don't exist in the target environment. Notice that if a table has never been imported or doesn't already exist in the target environment you must select **Include all objects** when you add the table to the solution, such as when adding a new custom table that you recently created. Otherwise you receive a missing dependency error message when you try to import the solution.

Create a solution with table segmentation using solution explorer

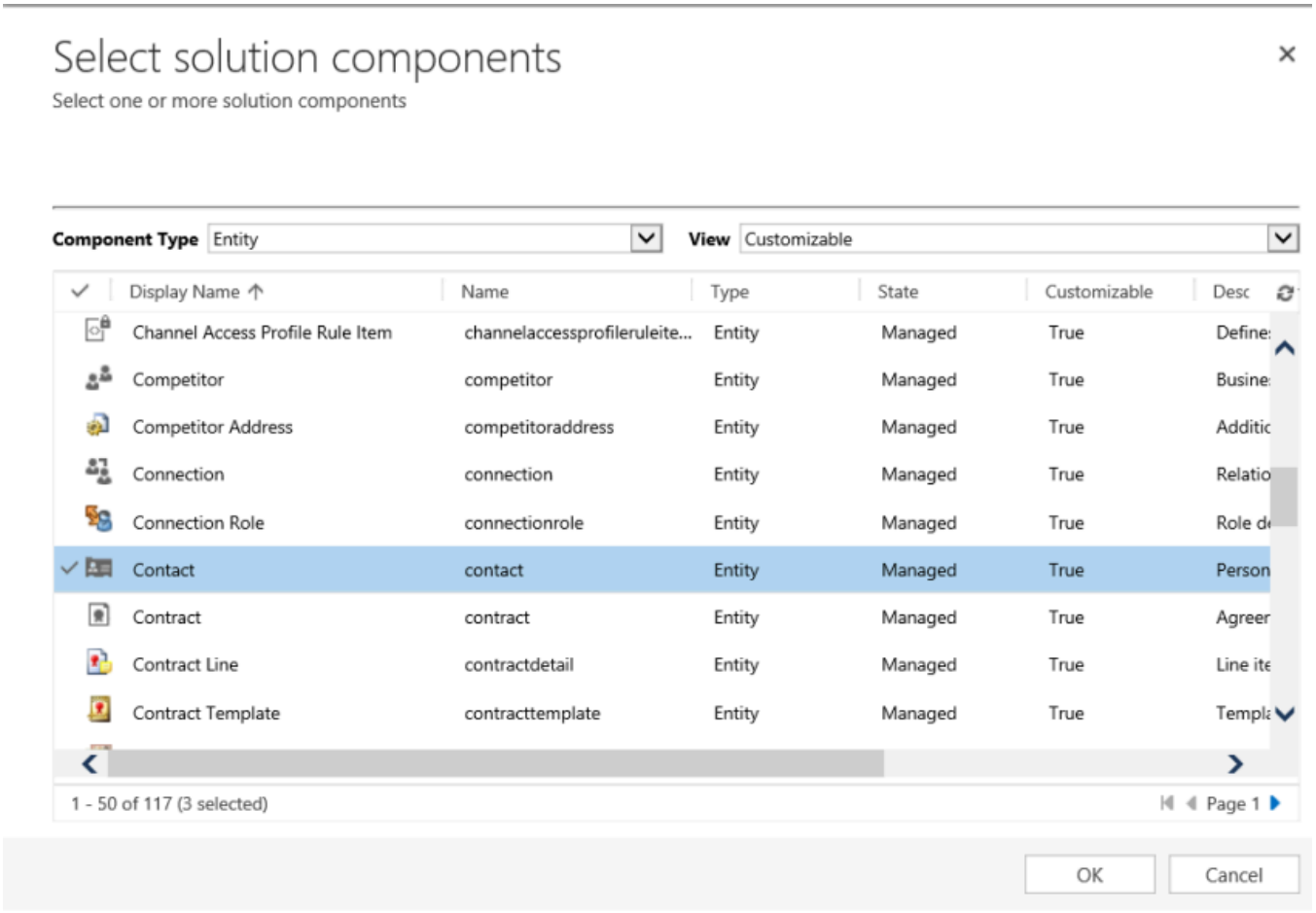
The following illustrations provide an example of creating a segmented solution by choosing table assets from the `Account`, `Case`, and `Contact` tables.

> **Note**
>
> The case table is included with some Dynamics 365 applications, such as Dynamics 365 Customer Service.

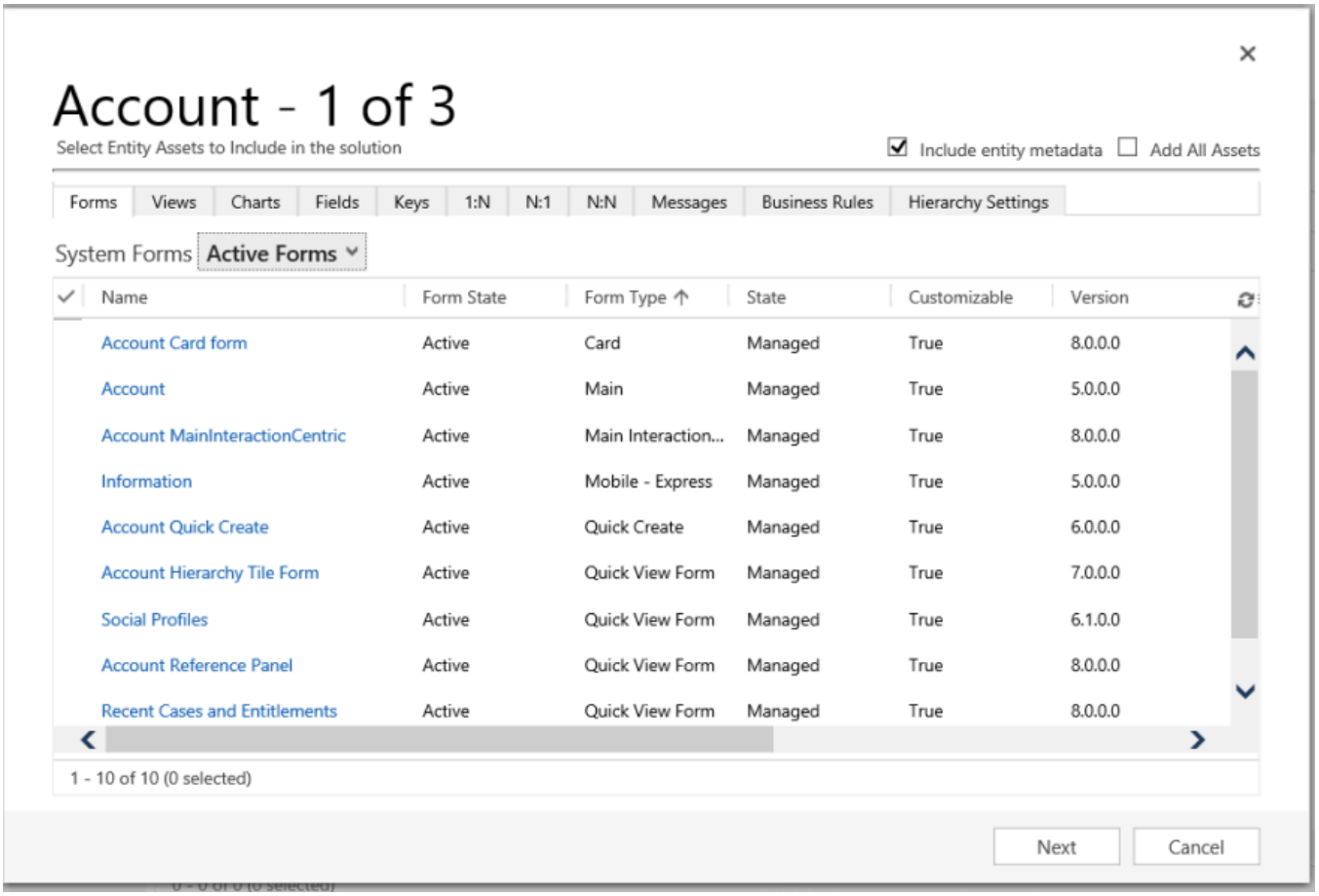
Start by opening an unmanaged solution you created. Choose the **table** component.



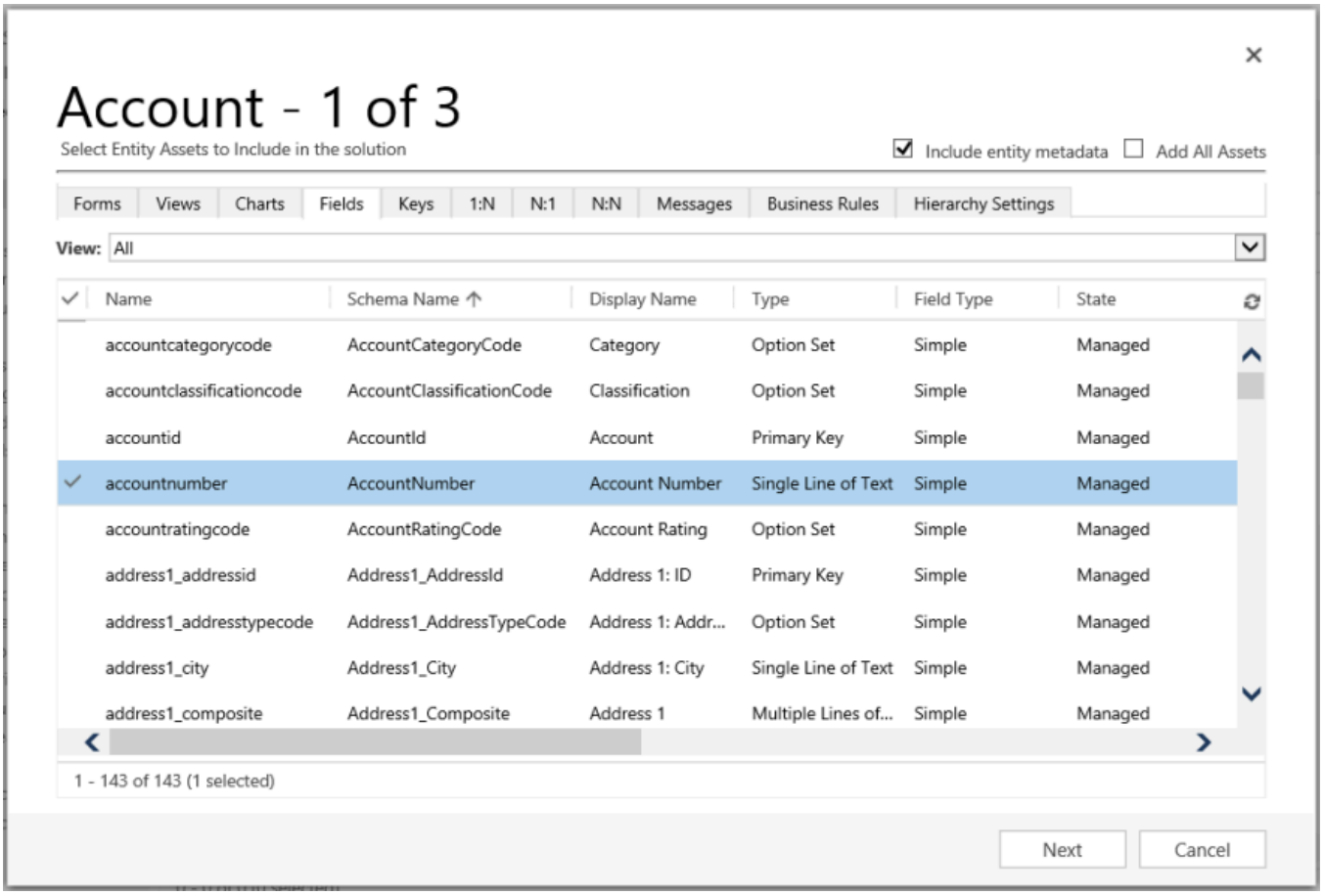
Then, select the solution components.



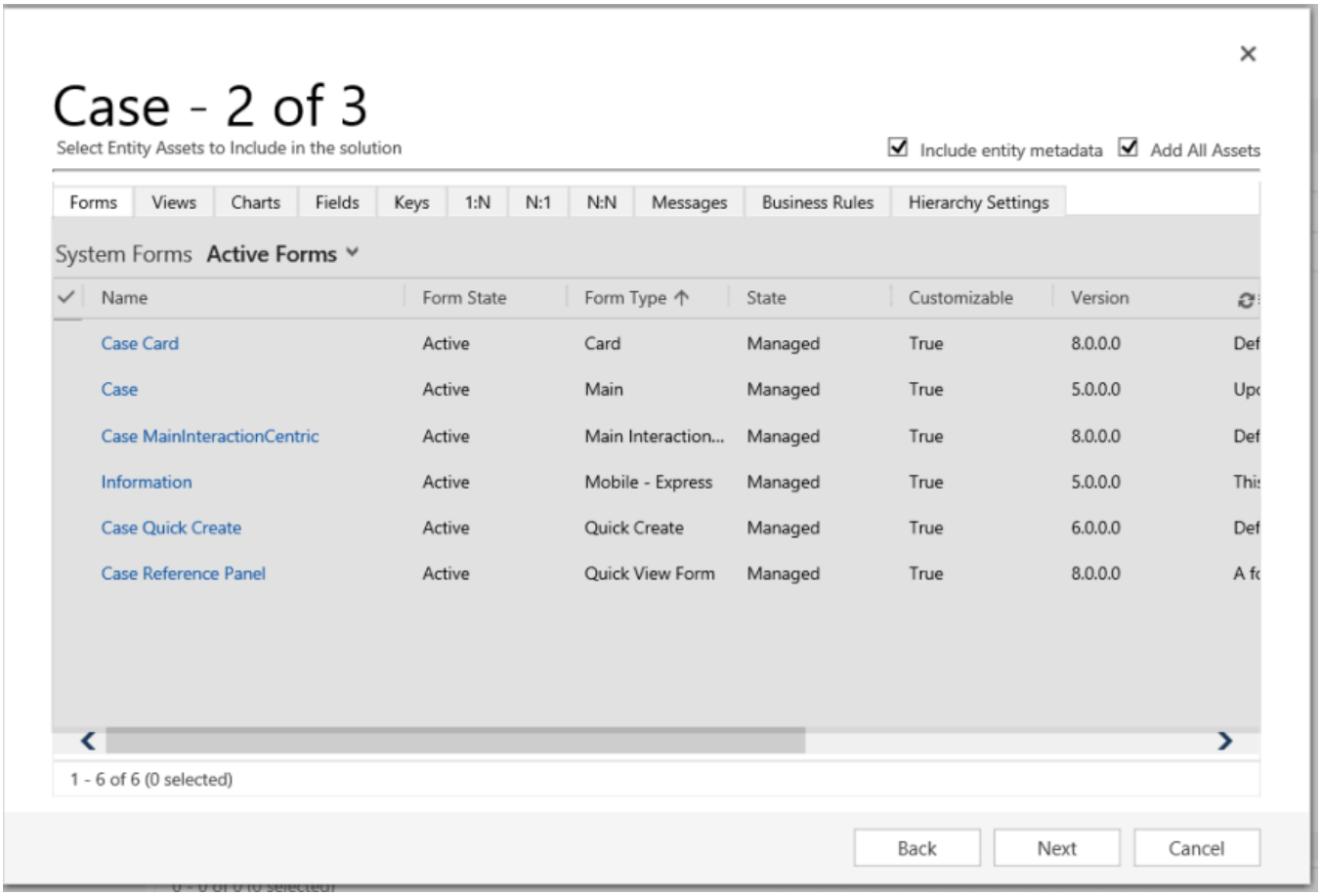
Follow the wizard. In Step 1, starting in alphabetical order, select the assets for the first table, the `Account` table, as shown here.



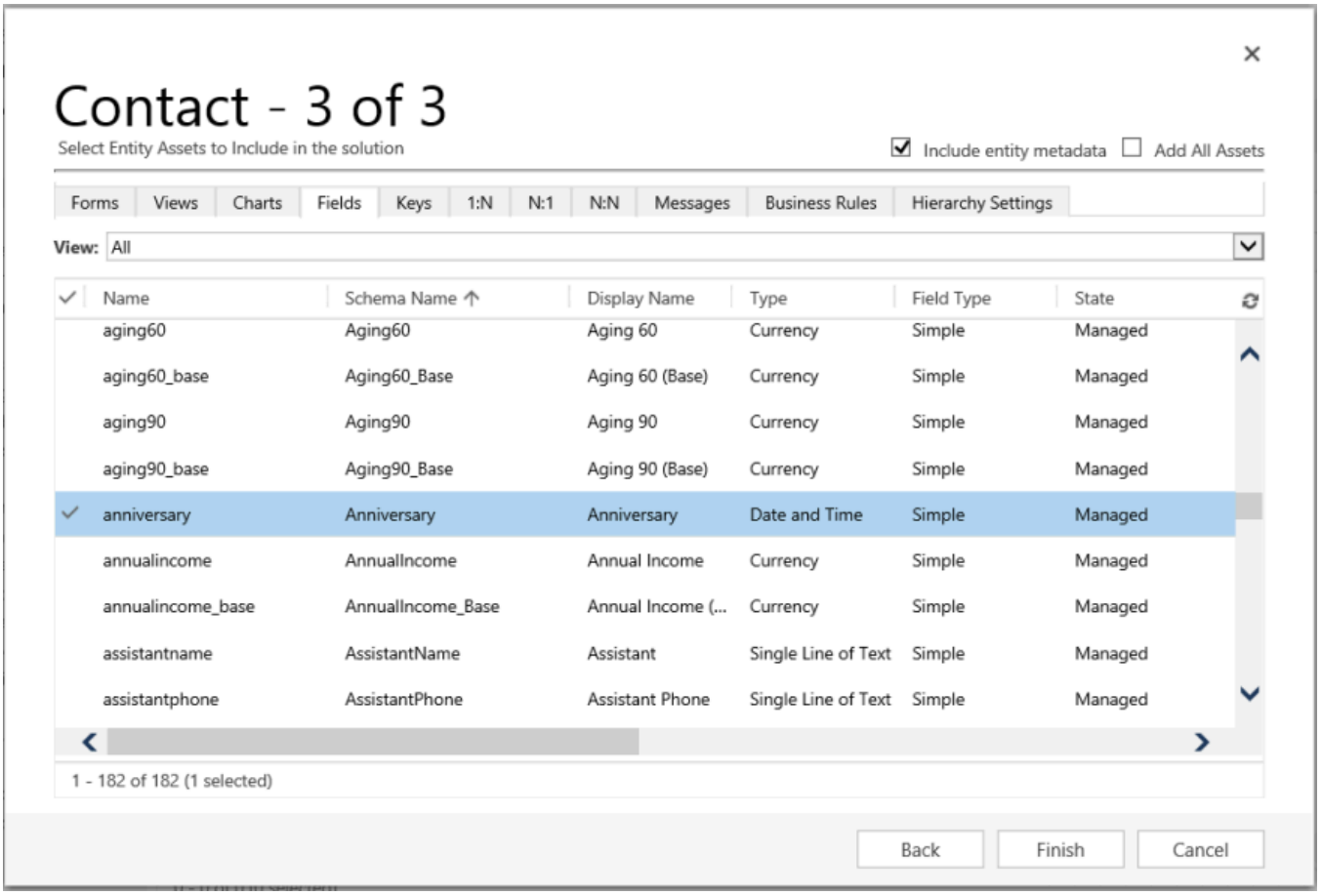
Open the **Fields** tab and select the **Account Number** column.



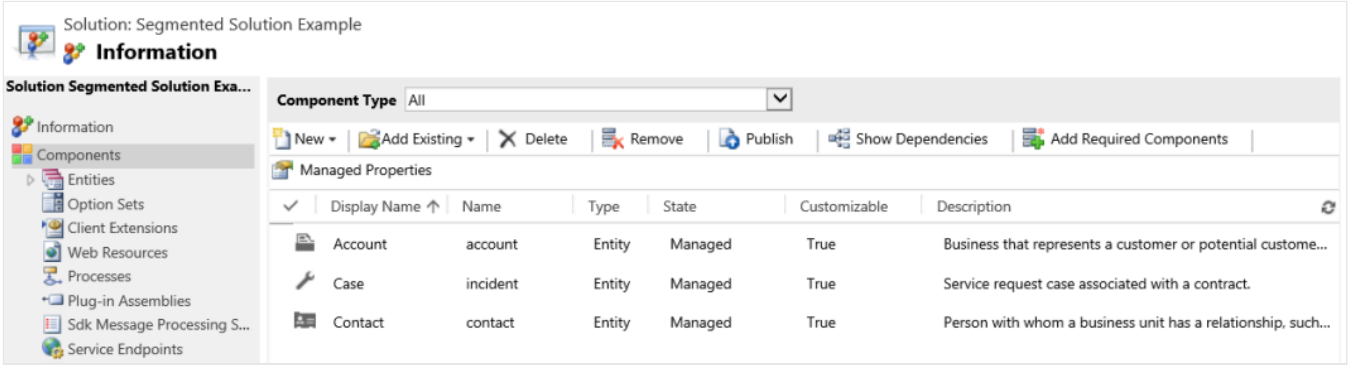
In Step 2, for the **Case** table, add all assets.



In Step 3, add the **Anniversary** column for the **Contact** table.



As a result, the segmented solution that's created contains three tables, `Account`, `Case`, and `Contact`. Each table contains only the assets that were chosen.



Delete a solution

Because there are two different types of solutions, managed and unmanaged, the behavior for deleting each type of solution is different.

The solution you want to delete might have components that have dependencies on other components. These dependencies must be removed before you can delete the component. More information: View dependencies for a component in Power Apps

Delete a managed solution

Deleting a managed solution removes (uninstalls) *all* the components within the solution. Additionally, *all* associated data are also deleted.

> ⊗ **Caution**
>
> Because all components within the solution and all associated data are deleted, use caution when you delete a managed solution.

Delete an unmanaged solution

Deleting an unmanaged solution deletes the solution container but doesn't delete any of the unmanaged components within. Any data associated also remains. Each unmanaged component must be individually deleted to remove all components within the unmanaged solution.

Delete a managed or unmanaged solution

> ⊗ **Caution**
>
> Before you delete a solution, make sure you understand the consequences. Once a solution is deleted, it can't be restored. More information: Delete a managed solution and Delete an unmanaged solution

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane, and then select (don't open) the solution you want to delete in the **Solutions** list.
3. Select **Delete** on the command bar.

Limitations

- Solution size is limited to 95 MB.
- Number of solutions is limited by Microsoft Dataverse capacity.
- Number of objects in a solution is limited by Dataverse capacity.

Related articles

- Use solutions

- For developers: Create, export, or import an unmanaged solution
- Power Platform ALM guide

Last updated on 01/09/2026

Export solutions

Article • 03/13/2023

You can manually export solutions using the steps in this article. We recommend that you create an unmanaged solution to use for exporting your customizations, and then export your customizations periodically so that you have a backup in case anything happens. You can't export managed solutions or the Default Solution.

> **Note**
>
> To implement healthy application lifecycle management (ALM) in your organization, consider using a source control system to store and collaborate on your solutions, and automate the solution export process. More information: **ALM basics** in the Power Platform ALM guide.

Export from Power Apps

1. Sign into Power Apps and select **Solutions** from the left navigation. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. In the list of solutions, select the unmanaged solution you want to export, and then select **Export**. Notice that you can't export managed solutions.

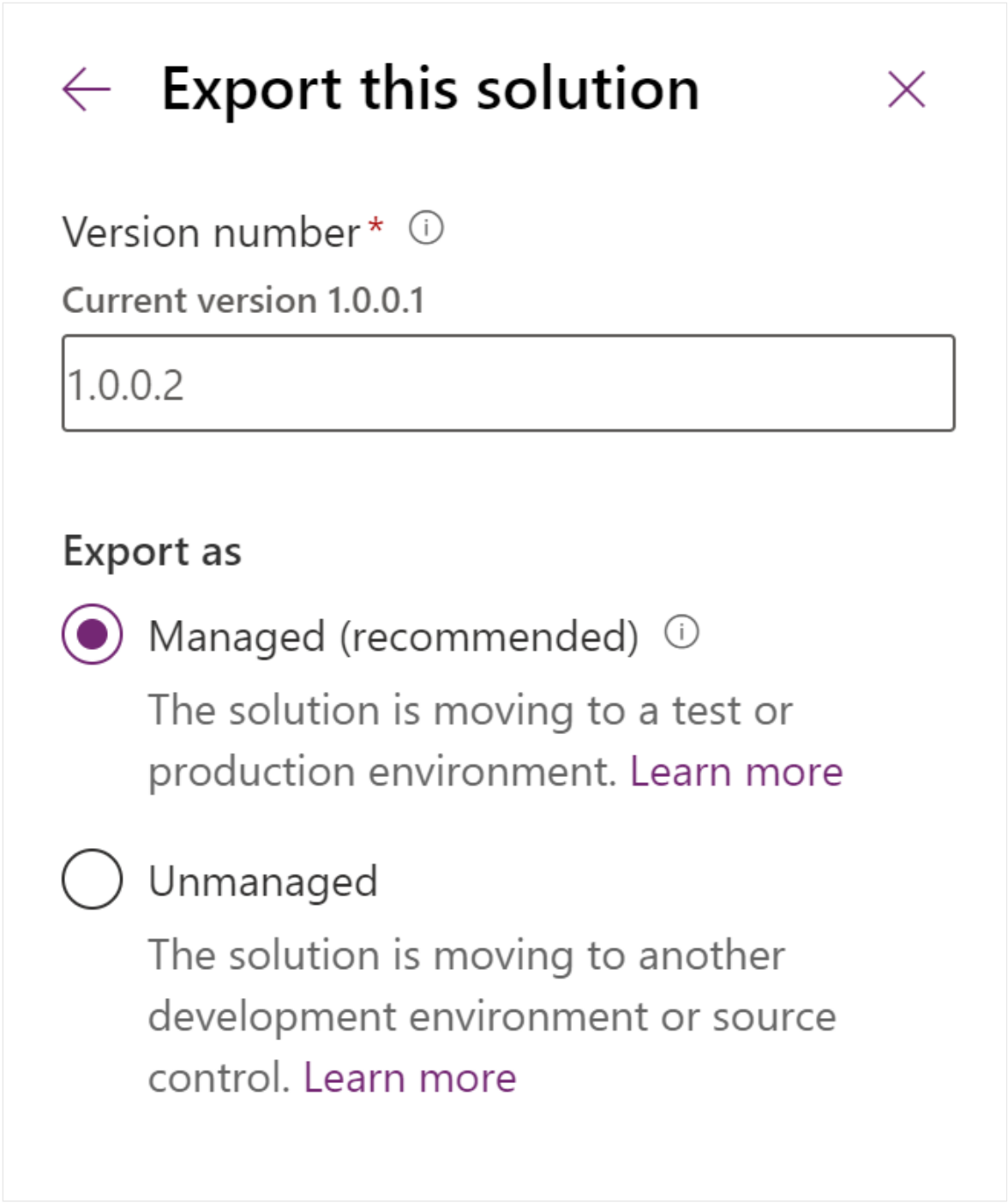
3. The **Before you export** right pane appears. Choose from the following options, and then select **Next**:

    - **Publish all changes**. Notice that, when you export an unmanaged solution, only published components are exported. We recommend that you select **Publish all changes** to make sure all components are included in the exported solution.

4. The **Export this solution** right pane appears. Enter or select from the following options, and then select **Export**:

    - **Version number**: Power Apps automatically increments your solution version while displaying the current version. You can accept the default version or enter your own.
    - **Export as**: Select the package type, either **Managed** or **Unmanaged**. More information: Managed and unmanaged solutions

- **Run solution checker on export**. Run the solution checker against the solution to detect performance and stability issues.



The export can take several minutes to complete. Once finished, the export .zip file is available in the download folder specified by your web browser.

Export using the classic experience

1. Sign into Power Apps and select **Solutions** from the left navigation. If the item isn't in the left navigation pane, select ...More and then select the item you want.

2. Select **Switch to classic**.

3. In the list select the solution you want to export and then select **Export**.

4. In the **Publish Customizations** step you will be reminded that only published customizations are exported and you will have the option to **Publish All Customizations** before you select **Next**.

5. If your solution contains any missing required components you will see the **Missing Required Components** step. You can disregard this warning only if you intend to import this as an unmanaged solution back into the original environment. Otherwise, follow the instructions in the dialog to cancel the export and add the required components.

6. In the **Export System Settings (Advanced)** step you can choose certain system settings to include in your solution. If your solution depends on any of the groups of system settings, select them and select **Next**.

   See **Settings options for solution export** below for details about the settings that will be included with each option.

7. In the **Package Type** step, you must choose whether to export the solution as an **Unmanaged** or **Managed** solution.

8. The next step allows you to choose a target solution for a specific version. This option is typically used by ISVs who may want to export a solution that is compliant with a previous version. Unless you intend to import this solution into an environment that is not upgraded to the same version as the environment version you are using, accept the default.

9. Select **Export** to download the solution file.

The exact behavior for downloading files varies between web browsers.

Settings options for solution export

If you export the solution from Power Apps, please disregard this section. The following table shows the options available when you export a solution from the classic experience.

Group	Setting	Description
Auto-numbering	Campaign Prefix	Prefix used for campaign numbering.
Case Prefix	Prefix to use for all cases throughout the app.	

Group	Setting	Description
Contract Prefix	Prefix to use for all contracts throughout the app.	
Invoice Prefix	Prefix to use for all invoice numbers throughout the app.	
Article Prefix	Prefix to use for all articles in the app.	
Order Prefix	Prefix to use for all orders throughout the app.	
Unique String Length	Number of characters appended to invoice, quote, and order numbers.	
Calendar	Calendar Type	Calendar type for the system. Set to Gregorian US by default
Date Format Code	Information about how the date is displayed throughout Microsoft Dataverse	
Date Separator	Character used to separate the month, the day, and the year in dates throughout the app.	
Max Appointment Duration	Maximum number of days an appointment can last.	
Show Week Number	Information that specifies whether to display the week number in calendar displays throughout the app.	
Time Format Code	Information that specifies how the time is displayed throughout the app.	
Week Start Day Code	Designated first day of the week throughout the app.	
Customization	Is Application Mode Enabled	Indicates whether loading of the app in a browser window that does not have address, tool, and menu bars is enabled.
Email-tracking	Allow Unresolved Address Email Send	Indicates whether users are allowed to send email to unresolved parties (parties must still have an email address).

Group	Setting	Description
Ignore Internal Email	Indicates whether incoming email sent by app users or queues should be tracked.	
Max Tracking Number	Maximum tracking number before recycling takes place.	
Render Secure Frame For Email	Flag to render the body of email in the webform in an IFRAME with the security='restricted' attribute set. This is additional security but can cause a credentials prompt.	
Tracking Prefix	History list of tracking token prefixes.	
Tracking Token Base	Base number used to provide separate tracking token identifiers to users belonging to different deployments.	
Tracking Token Digits	Number of digits used to represent a tracking token identifier.	
General	Block Attachments	Prevent upload or download of certain attachment types that are considered dangerous.
Currency Format Code	Information about how currency symbols are placed throughout the app.	
Currency Symbol	Currency Symbol	
Full Name Display Order	Order in which names are to be displayed throughout the app.	
Presence Enabled	Information on whether IM presence is enabled.	
Negative Format	Information that specifies how negative numbers are displayed throughout the app.	
Number Format	Specification of how numbers are displayed throughout the app.	
Pricing Decimal Precision	Number of decimal places that can be used for prices.	
Share To Previous Owner On Assign	Information that specifies whether to share to previous owner on assign.	

Group	Setting	Description
Marketing	Allow Automatic Response Creation	Indicates whether automatic response creation is allowed
Allow Automatic Unsubscribe	Indicates whether automatic unsubscribe is allowed.	
Allow Automatic Unsubscribe Acknowledgement	Indicates whether automatic unsubscribe acknowledgement email is allowed to send.	
Allow Marketing Email Execution	Indicates whether marketing emails execution is allowed.	
Outlook Synchronization	Allow Address Book Synchronization	Indicates whether background address book synchronization in Microsoft Office Outlook is allowed.
Allow Offline Scheduled Synchronization	Indicates whether background offline synchronization in Outlook is allowed.	
Allow Scheduled Synchronization	Indicates whether scheduled synchronizations to Outlook are allowed.	
Email Send Polling Frequency	Normal polling frequency used for sending email in Outlook.	
Min Address Synchronization Frequency	Normal polling frequency used for address book synchronization in Outlook.	
Min Offline Synchronization Frequency	Normal polling frequency used for background offline synchronization in Outlook.	
Min Synchronization Frequency	Minimum allowed time between scheduled Outlook synchronizations.	
Auto-Tag Max Cycles	Maximum number of aggressive polling cycles executed for email auto-tagging when a new email is received.	
Auto-Tag Interval	Normal polling frequency used for email auto-tagging in Outlook.	
ISV Config	Service Calendar Appearance Configuration	You can define visual styles for service calendars.

More information: Service Calendar Appearance Configuration

See also

Import solutions
Update solutions
For developers: Create, export, or import an unmanaged solution

Import solutions

09/05/2025

You can manually import solutions using the steps in this article. You must import only those solutions that you've obtained from a trusted source.

Solutions are packages of customizations that can be exported from one environment and imported into another. Solutions can be managed or unmanaged. Managed solutions are intended for production use, while unmanaged solutions are typically used during development.

> **Note**
>
> - The create privilege is required to import a component. Although, the System Customizer security role has create privilege on most components that are commonly imported, by default it doesn't have create privilege on the **Plug-In Assembly** table. The System Administrator security role has this privilege.
> - When you import a managed solution, all component changes will be brought into the environment in a published state. However, when you import an unmanaged solution, the changes are imported in a draft state so you must publish them to make them active.
> - To implement healthy application lifecycle management (ALM) in your organization, consider using a source control system to store and collaborate on your solutions, and automate the solution import process. More information: ALM basics in the Power Platform ALM guide.
> - The maximum size of a solution file is 95 MB.

When you import an **unmanaged** solution:

- You add all the components of that solution into your environment and can't delete the components by deleting the solution. Deleting the unmanaged solution deletes only the solution container.
- That contains customized components, the existing customizations to the component will be overwritten after the unmanaged solution import. You can’t undo this.

To import a solution:

1. Sign into Power Apps and select **Solutions** from the left navigation. If the item isn’t in the side panel pane, select ...More and then select the item you want.

2. On the command bar, select **Import solution**.

3. On the **Import a solution** page, select from where you want to import the solution:

    - **This device**. Select **Browse** to locate the compressed (.zip or .cab) file located on your device that contains the solution you want to import.
    - **Pipeline**. Select this option to import a solution from a pipeline. To use this option, you must have the solution set up in a pipeline. More information: Overview of pipelines in Power Platform.

4. Select **Next**.

5. Information about the solution is displayed. By default, in the **Advanced settings** section, if plugin steps (also known as SDK message processing steps) and flows exist in the solution, they'll be imported. Clear the **Enable Plugin steps and flows included in the solution** option if you don't want the import to attempt to activate any plugin steps that are imported in an inactive state. Flows will not be deactivated when the option is unchecked.

6. If your solution contains connection references, you're prompted to select the connections you want. If a connection doesn't already exist, create a new one. Select **Next**.

7. If your solution contains environment variables, you'll be prompted to enter values. You won't see this screen if value(s) are already present in your solution or the target environment.

8. If missing dependencies are detected in the target environment, a list of the dependencies is presented. In environments where the required package version is available for import in the target environment, a link to resolve the dependency is presented. Selecting the link takes you to the Power Platform admin center where you can install the application update. After the application update is completed, you can start the solution import again.

9. Select **Import**.

Your solution imports in the background and might take a few moments.

If you have imported any changes that require publishing, you must publish customizations before they're available.

If the import isn’t successful, you'll see a notification on the solutions page showing any errors or warnings that were captured. Select **Download Log File** to capture details about what caused the import to fail. The most common cause for an import to fail is that the solution didn't contain some required components.

When you download the log file, you'll find an XML file that you can open using Office Excel to view the contents.

> **Note**
>
> You can view the details of all solution operations including solution import with the solution history feature. To view these operations, select **See history** on the solutions page.

Troubleshooting solution import

For known issues and information about how to troubleshoot working with solutions, go to Manage apps and solutions in the Power Apps Troubleshooting documentation.

Importing apps that use an on-premises data gateway ID

When you import a Power Apps app that uses an on-premises data gateway ID, note the following requirements:

- The connection to the gateway must be a shared connection.
- The connections must be removed and then added back with the correct gateway ID after solution import into the target environment.

See also

- Update solutions
- Export solutions
- Publish changes
- For developers: Create, export, or import an unmanaged solution
- Import a solution (Power Automate)

Note: The author created this article with assistance from AI. Learn more

Set the preferred solution

Article • 04/10/2025

By default, unless you're already in the context of an unmanaged solution when you create an object, all solution objects are stored in the solution named Common Data Services Default Solution. For most situations, you don't want to create your solution objects in either of the two Dataverse system solutions, which are the Common Data Services Solution and the Default Solution. This article explains why using the system solutions is discouraged and provides step-by-step instructions to set and manage your preferred solution.

Why not create your objects in the system solutions?

There are a few reasons you *shouldn't* create your solution objects in either the Common Data Services Solution and the Default Solution:

- There's no control over the separation of solution objects between makers. All objects can be viewed from the Default Solution and the Common Data Services Solution is the default solution so other makers can and likely are using it.
- The Default Solution can't be exported for import to other environments.
- You can't change the solution publisher for the Default Solution. Additionally, since the solutions already exist, you can't change the solution publisher prefix for either system solutions. More information: Solution publisher

Using preferred solutions is a way to determine which solutions contain each maker's edits that occur in Power Apps. Setting the preferred solution enables makers to view and update which solution they're using. After you set your preferred solution, you can create objects in the solution you specify and those objects will automatically be in that solution so that you control the objects within the solution. Then, you can export your preferred solution and import the solution to other Microsoft Dataverse environments.

> **Note**
>
> When your preferred solution isn't set, by default, the **Common Data Services Default Solution** is your preferred solution. If **Common Data Services Default Solution** isn't available, the solution named **Default Solution** is used as your preferred solution. You can change this to your desired solution by following the steps in this article. For more information about the **Common Data Services Default Solution** and **Default Solution**, go to Default solutions.

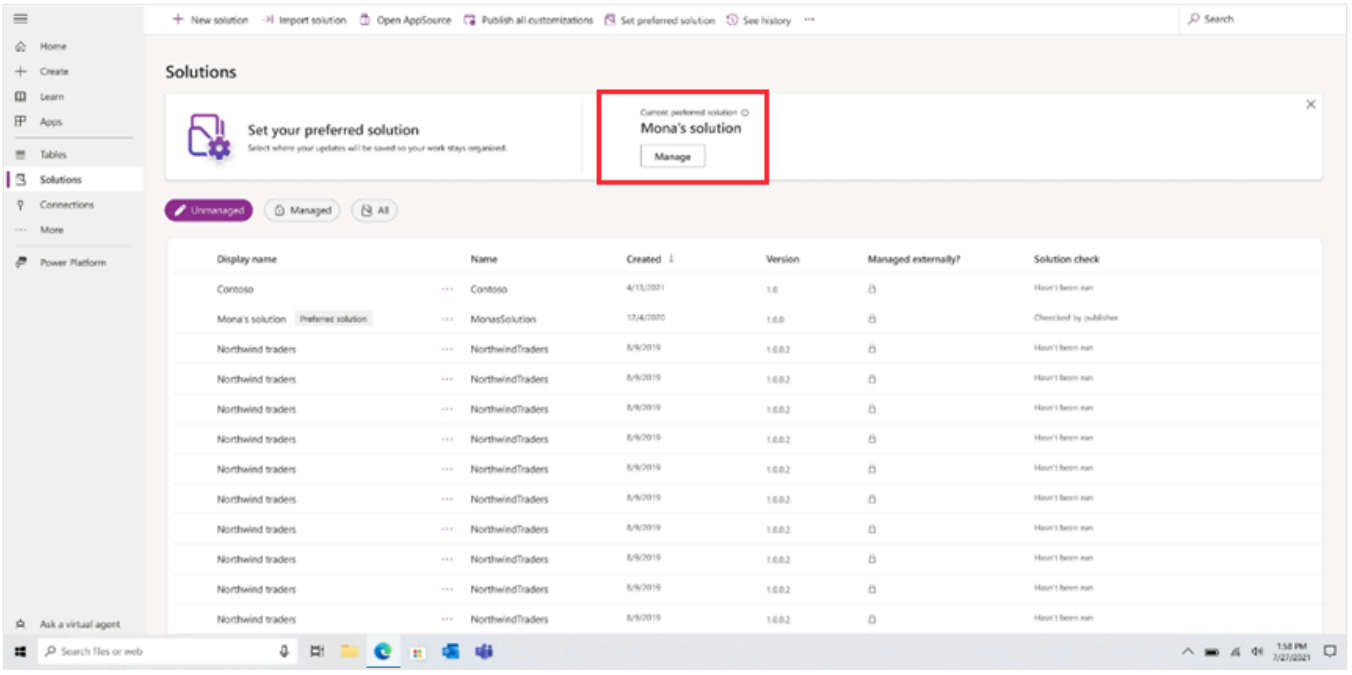
Prerequisites

We recommend that admins enable the **Cloud flows** and **Canvas apps** environment setting in Power Platform. Enabling this feature ensures cloud flows are also stored in Dataverse and that they're added to the maker's preferred solution. More information: Create new canvas apps and cloud flows in a Dataverse solution

Set your preferred solution

1. Sign into Power Apps, and then select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select the unmanaged solution that you want to make your preferred solution. Don't have one? More information: Create a solution
3. Select **Set preferred solution** on the command bar.

Once your preferred solution is set, notice the **preferred solution** indicator, which can be viewed from the **Solutions** area or when you hover over the **Environment** switcher.

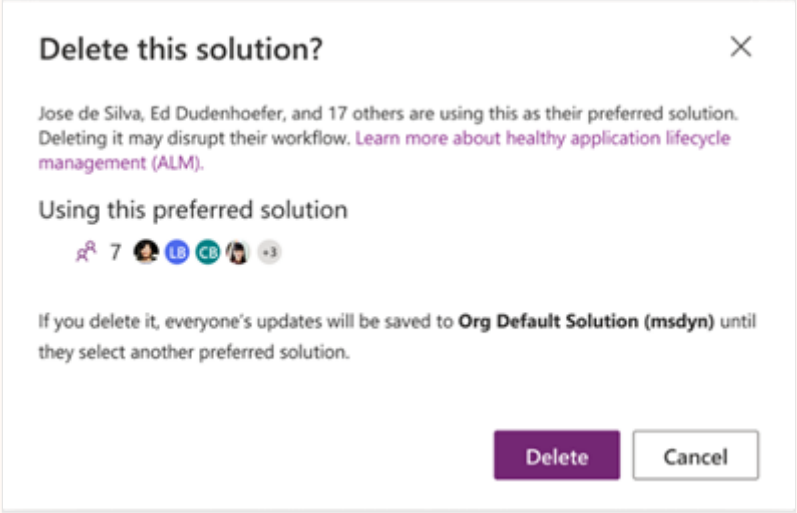


Deleting a preferred solution

You can delete your preferred solution or a preferred solution that other makers have also set as their preferred solution.

> ⚠ **Warning**

When you delete a preferred solution, you receive a warning that includes the fallback default solution and the number of other makers who are using the same solution. Delete an active solution only when you're sure it won't impact your work or the work of other makers.



Limitations

- You can't set or view the preferred solution in the classic solution explorer.
- Components that are created in the classic solution explorer won't go into the preferred solution.
- Preferred solution currently doesn't work with Dataverse for Teams, cards, dataflows, AI Builder, chatbots, connections, gateways, custom connectors, Power Automate flows (limited), and canvas apps created from image or a Figma design.
- When a component is already part of an existing unmanaged solution, it will still be added to the preferred solution.

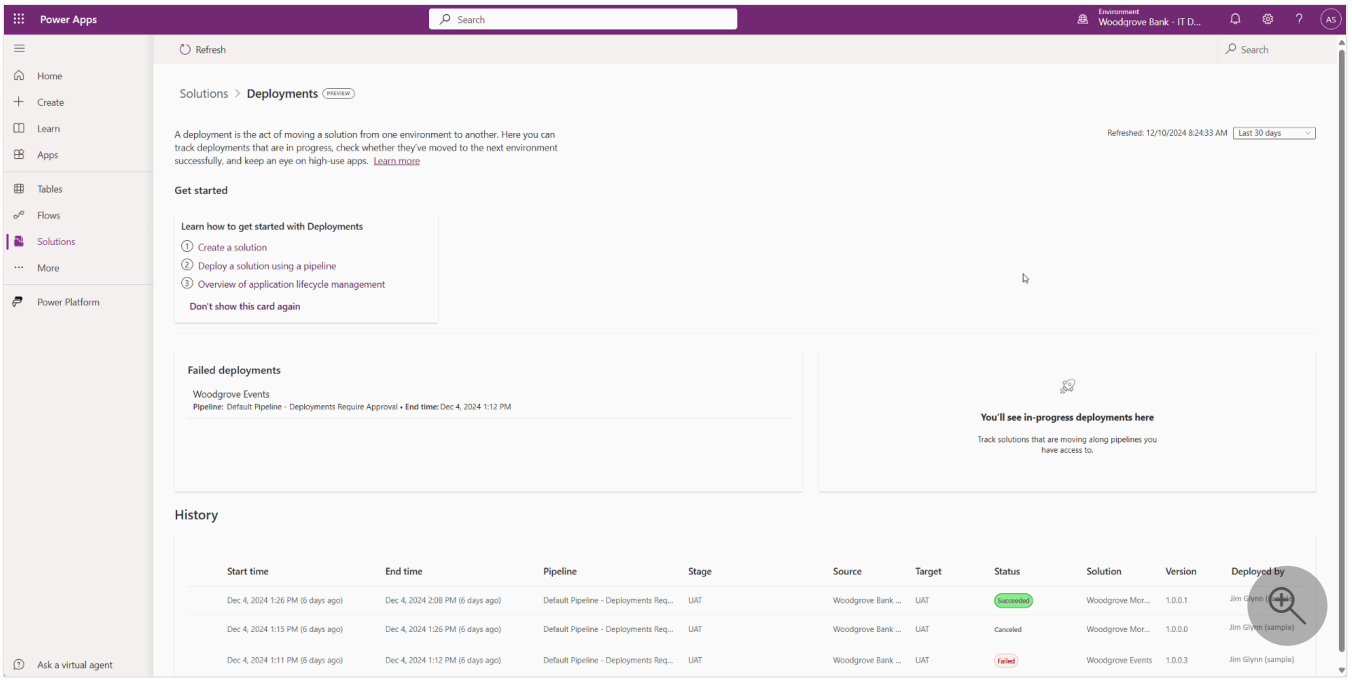
See also

Solutions overview
Set the preferred solution with Power Apps (video)

View solution deployments on the deployment page

Article • 05/14/2025

From the **Solutions** page, makers can navigate to **Deployments** to view all of their solution deployments, across all solutions and pipelines, in one place. If their deployment began from (or was deployed to) the current environment, they're able to view its status and other details.



Get started

When you first access the Deployment page, you notice the **Get started** section. This section currently offers several links to relevant documents designed to help makers gain a better understanding of solutions, pipelines, and the in-product Application Lifecycle Management (ALM) process.

Failed deployments

On the left of the overview, makers can view failed deployments if any. When you select each deployment, a panel appears, highlighting the necessary actions to either retry or revise the solution, ensuring a successful deployment.

Active deployments

On the right of the overview, makers view their ongoing deployments. When makers select these deployments, they can view the in-progress details of the deployment, such as waiting

for a predeployment approval. The active deployments page is useful for makers who work in multiple solutions that are deploying simultaneously.

Deployment history

At the bottom of the overview, makers view their deployment history. This section provides a comprehensive list of all deployments, including the deployment status, the solution, the pipeline, and the environment. Makers can filter this list by solution, pipeline, and environment to quickly find the deployment they're looking for. This run history view is unique. First it's filtered to only show the current user's deployments specifically to and from this environment, and second it's upleveled to show all the current user's deployments, not just involving one solution or pipeline. Selecting a deployment in this grid shows a details panel, summarizing all the deployment information and displaying the deployment notes as well.

Retry failed deployments

Deployments shown as **Failed** in the run history view can be deployed by selecting **Retry** in the details panel if the operation was **Deploy**. A confirmation message appears when you confirm the retry.

Take action with advisor-powered recommendations

Actionable recommendations for improving ALM health within your environment is shown in the **Take action** section at the bottom of the Deployment page. These recommendations help ensure that makers are following the best practices to enable healthy ALM within the organization.

Why can't I see solution deployments other than those that I initiated?

In order to view others' solution deployments, you must have the Deployment Pipeline Administrator security role or equivalent privileges to view other deployments.

Related articles

Use pipelines in Power Platform to deploy solutions

View solution deployments on the deployment page for admins

Use pipelines in Power Platform to deploy solutions

Article • 02/23/2023

Easily deploy solutions to test and production environments using pipelines in Power Platform. Once pipelines are in place, makers can initiate in-product deployments with a few clicks. Makers do so directly within their development environments. More information: Overview of pipelines in Power Platform

See also

Solutions overview

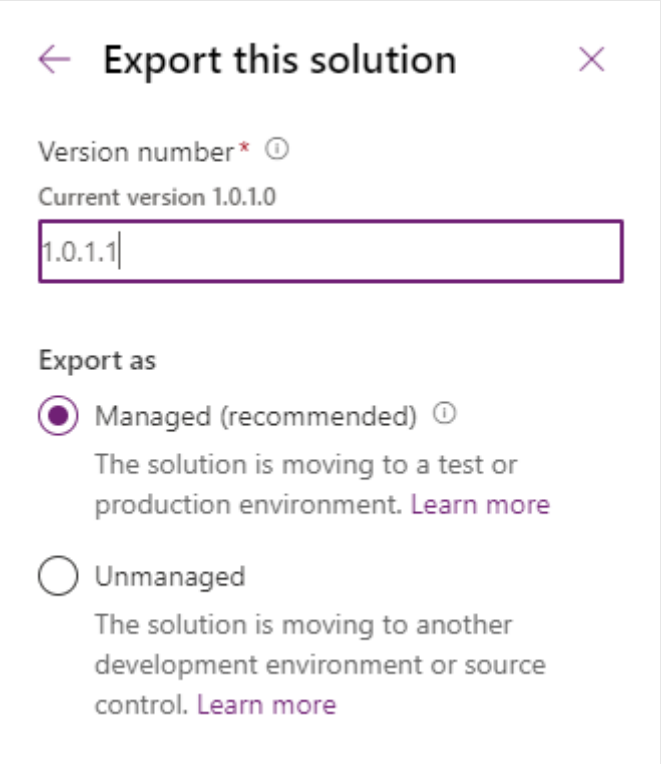
Upgrade or update a solution

Article • 08/31/2023

There are times when you need to update an existing managed solution. To update the solution, follow these steps:

1. Open the unmanaged solution in your development environment and create new or add and remove the existing components that you want.

2. Increment the version number when you export the solution as a managed solution. More information: Understanding version numbers for updates



3. Apply the upgrade or update in the target environment

Apply the upgrade or update in the target environment

The procedure to import the updated solution is similar to installing a new managed solution, except you'll get some different options. If you're updating a solution you got from someone else, you should get guidance from the solution publisher about which options you should choose.

1. Sign into Power Apps, select the target environment you want, and then select **Solutions** from the left navigation. If the item isn't in the side panel pane, select ...

More and then select the item you want.

2. On the command bar, select **Import**.

3. On the **Select Solution Package** page, select **Browse** to locate the compressed (.zip or .cab) file that contains the solution you want to update.

4. Select **Next**.

5. This page displays a yellow bar indicating **This solution package contains an update for a solution that is already installed**. To upgrade the solution, select **Next**. To view additional options, expand **Advanced settings**, and then select from the following solution action options:

    - **Upgrade** This is the default option and upgrades your solution to the latest version and rolls up all previous patches in one step. Any components associated to the previous solution version that aren't in the newer solution version will be deleted. This option ensures that your resulting configuration state is consistent with the importing solution including removal of components that are no longer part of the solution.

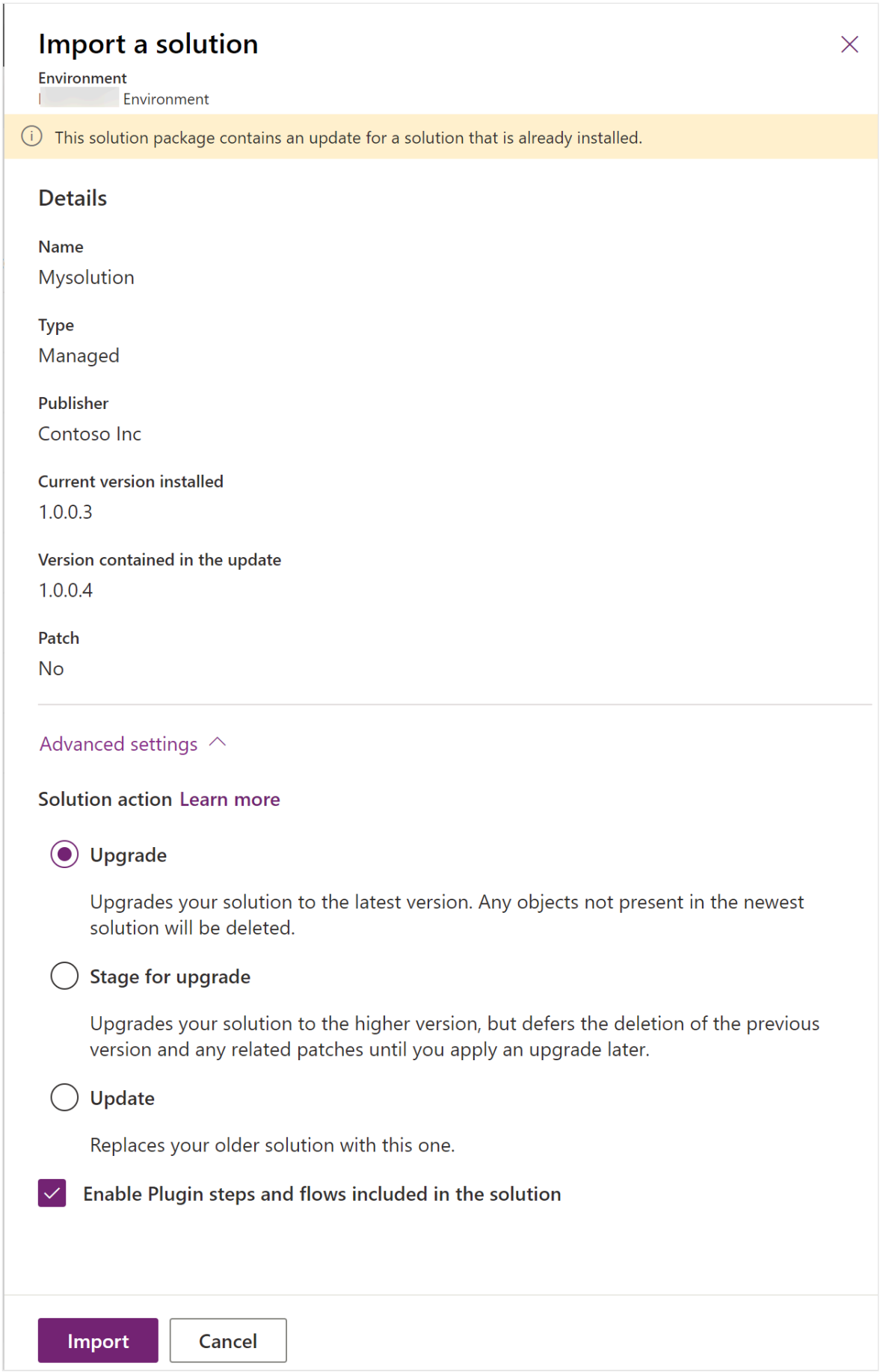
    - **Stage for Upgrade** This option upgrades your solution to the higher version, but defers the deletion of the previous version and any related patches until you apply a solution upgrade later. This option should only be selected if you want to have both the old and new solutions installed in the system concurrently so that you can do some data migration before you complete the solution upgrade. Applying the upgrade deletes the old solution and any components that aren't included in the new solution.

    - **Update** This option replaces your solution with this version. Components that aren't in the newer solution won't be deleted and remains in the system. Be aware that the source and destination environment may differ if components were deleted in the source environment. This option has the best performance by typically finishing in less time than the upgrade methods.

6. Decide whether to enable the following option for post import actions:

    - **Enable plug-in steps and flows included in the solution**
      Selecting this option will enable plug-ins and Power Automate flows that are included in the solution.

7. Select **Import**.



8. You may need to wait a few moments while the solution import completes. If it's successful, you can view the results and select **Close**.

Managed customizations are always imported in a published state, so there's no need to publish customizations after import in this scenario.

Completing Solution Upgrade If you chose to stage for upgrade, or if the system had an issue completing an upgrade, you'll see that you have the original solution still installed in your system as well as a new solution that has the same solution name as the base solution suffixed with _Upgrade. To complete the upgrade, select the base solution in the solution list and select **Apply Solution Upgrade**. This uninstalls all previous patches and the base solution then rename the _Upgrade solution to be the same name as the previous base solution. Any components that were in the original solution and patches that aren't present in the _Upgrade solution will be deleted as part of this process.

> ⓘ **Note**
>
> Recent changes in the platform have optimized the single step upgrade process to no longer use a temporary _Upgrade solution or use an uninstall operation on the original solution.

Understanding version numbers for updates

A solution's version has the following format: major.minor.build.revision. An update must have a higher major, minor, build, or revision number than the parent solution. For example, for a base solution version 3.1.5.7, a small update could be a version 3.1.5.8 or a slightly more significant update could have version 3.1.7.0. A substantially more significant update could be version 3.2.0.0.

Removing a managed component from the target environment

There might be occasions when you want to remove a managed component from an environment. There are two different solution actions that you can perform that will remove a managed component from an environment.

- Upgrade the solution (recommended). In the development environment where the managed solution originated, update the solution so that it no longer includes the component. In other words, remove the component such as a column, chart, or form from the solution and then export it as managed. Then, when you import the managed solution into the target environment, choose the **Solution action** as

Upgrade. This action removes the component (if there aren't any other components that depend on it in the target environment).

- Delete the managed solution. This action removes all components in the solution. In the environment where the managed solution was imported, delete the managed solution that includes the managed component. You can do this from the **Solutions** area of Power Apps.

> ⚠ **Warning**
>
> Deleting a managed solution deletes ALL the components that are in the managed solution as well as any related data. Always use caution before deleting a managed solution.

Overwrite customizations option

The option to overwrite customizations is available with the classic import experience, when using the Microsoft Power Platform CLI pac solution import command, or by using the `OverwriteUnmanagedCustomizations` option with the `ImportSolution` or `ImportSolutionAsync` messages. More information: ImportSolutionRequest class, ImportSolution action, or ImportSolutionAsync action.

> ⓘ **Important**
>
> Selecting the **Overwrite Customizations (not recommended)** option will overwrite or remove any unmanaged customizations previously performed on components included in this solution. This option does not affect components that support merge behavior (forms, sitemap, ribbon, app modules). Components that have other managed solutions on top of the existing solution you are replacing do also still remain on top and are not affected by this option.

See also

Layering within a managed solution Add solution components
Create solution patches
For developers: Cloning, patching, and upgrading

Feedback

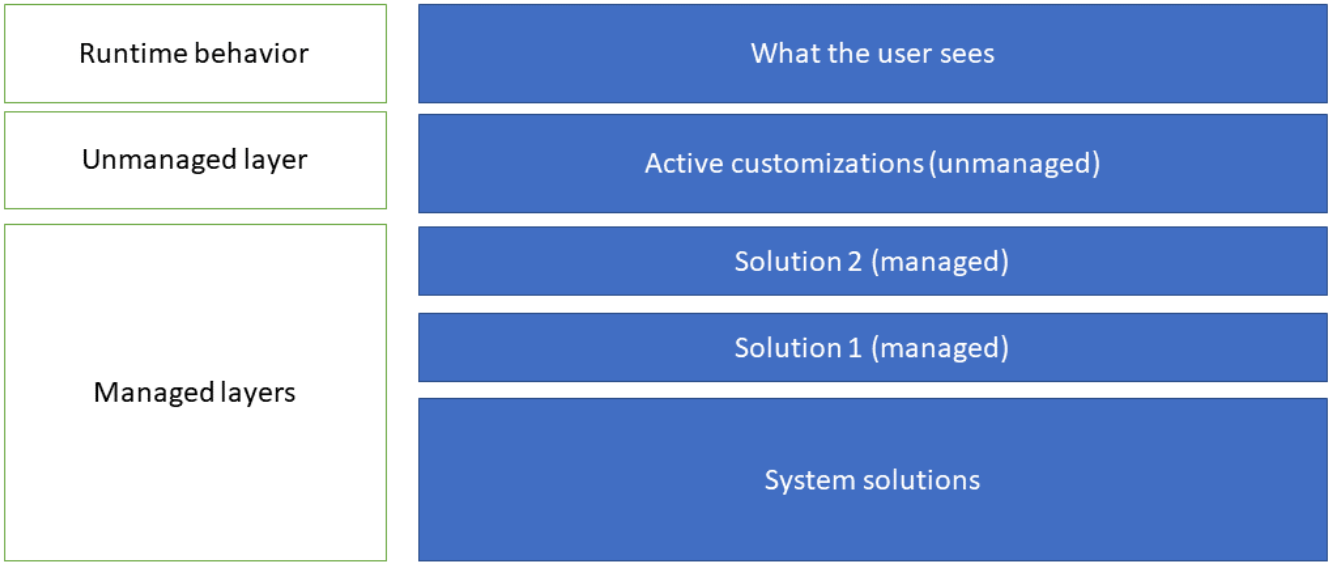
Was this page helpful? Yes No

Provide product feedback

Solution layers

Managed and unmanaged solutions exist at different levels within a Microsoft Power Platform environment. In Microsoft Dataverse, there are two distinct layer levels:

- Unmanaged layer. All imported unmanaged solutions and unmanaged customizations exist at this layer. The unmanaged layer is a single layer.
- Managed layers. All imported managed solutions and the system solution exist at this level. When multiple managed solutions are installed, the last one installed is above the managed solution installed previously. This means that the second solution installed can customize the one installed before it. When two managed solutions have conflicting definitions, the runtime behavior is either “Last one wins” or a merge logic is implemented. If you uninstall a managed solution, the managed solution below it takes effect. If you uninstall all managed solutions, the default behavior defined within the system solution is applied. At the base of the managed layers level is the system layer. The system layer contains the tables and components that are required for the platform to function.



Solution merge behavior

When you prepare your managed solution for distribution, remember that an environment might have multiple solutions installed or that other solutions might be installed in the future. Construct a solution that follows best practices so that your solution doesn't interfere with other solutions.

The processes that Dataverse uses to merge customizations emphasize maintaining the functionality of the solution. While every effort is made to preserve the presentation, some incompatibilities between customizations might require that the computed resolution will

change some presentation details in favor of maintaining the customization functionality. More information: Understand how managed solutions are merged

View the solution layers for a component

The see solution layers feature allows you to view all component changes that occur due to solution changes over time. Within a solution layer, you can drill down to view specific changed and unchanged property details for a component. You can access solution layers from the **Solutions** area in Power Apps (make.powerapps.com).

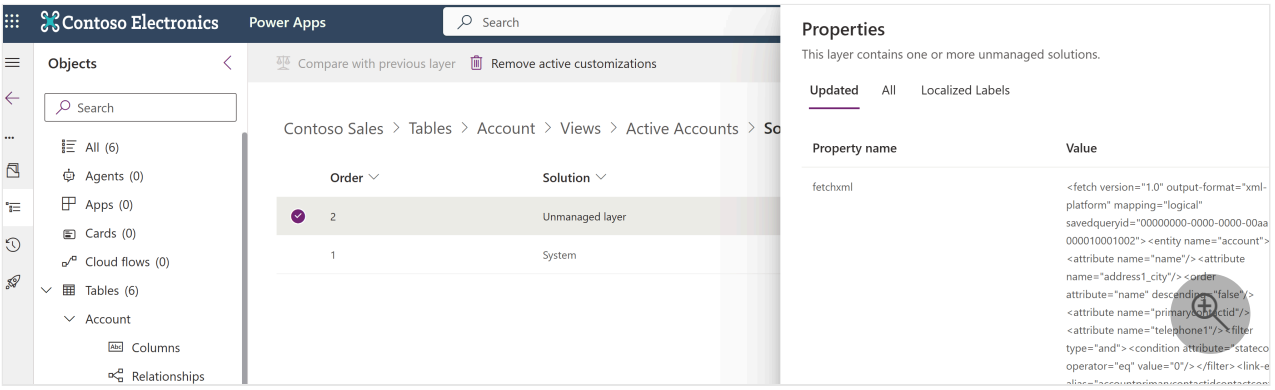
The see solution layers feature:

- Lets you see the order in which a solution changed a component.
- Lets you view all properties of a component within a specific solution, including the changes to the component.
- Can be used to troubleshoot dependency or solution-layering issues by displaying change details for a component that was introduced by a solution change.

1. Sign in to Power Apps, select **Solutions**, open the solution you want, select a component, such as the **Account** table, and then on the command bar select **Advanced** > **See solution layers**.

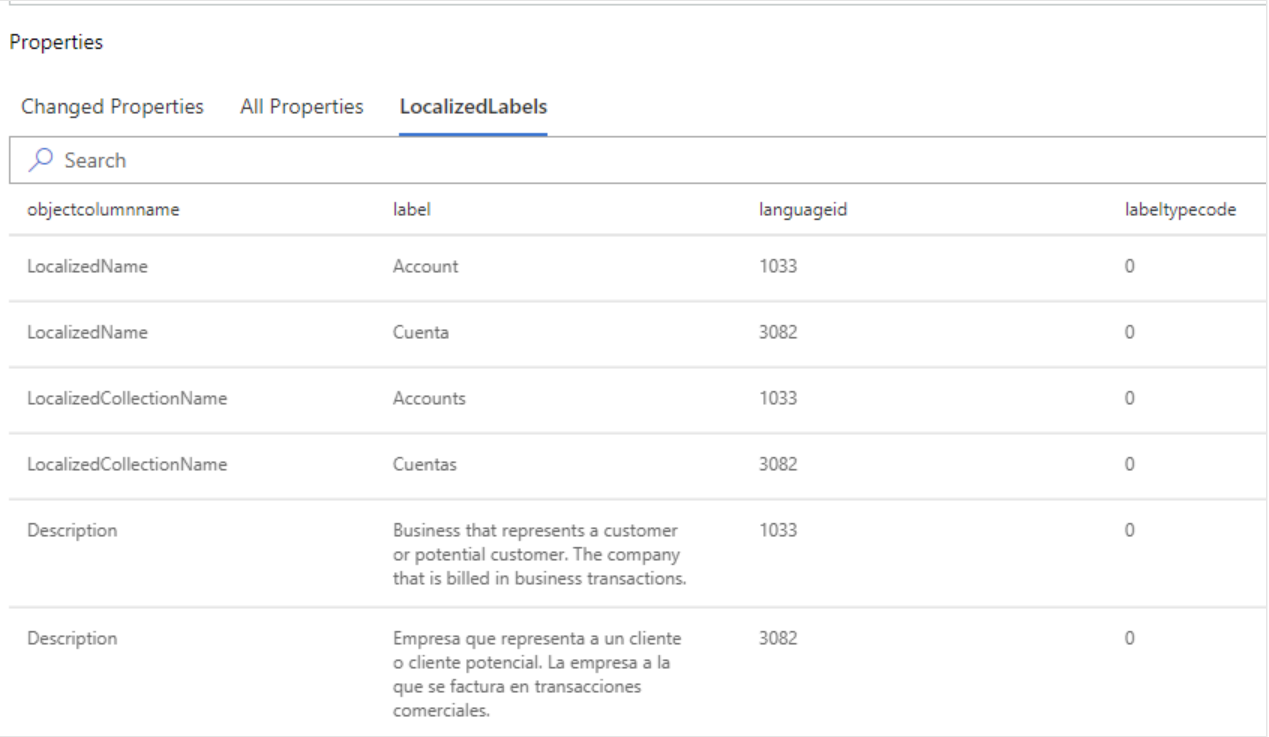
2. The solution layer page appears. It displays each layer for the component, such as the Active Accounts view for the **Account** table used as an example here, with the most recent layer at the top, which is typically indicated as the **Active** layer. The active layer determines the runtime behavior of the component.

3. To view the details for a solution layer, select it. The **Properties** pane is displayed. The **Updated** tab displays only those properties that were modified as part of the specific solution layer. Select the **All Properties** tab to view all properties, including changed and unchanged properties, for the solution layer.



4. If the component includes translations that have been imported, select the **Localized Labels** tab to display information for components that have label columns in the solution

layer. The base language and any imported translation text are displayed as indicated in the **languageid** column. Note that if no labels exist the tab isn't displayed.



objectcolumnname	label	languageid	labeltypecode
LocalizedName	Account	1033	0
LocalizedName	Cuenta	3082	0
LocalizedCollectionName	Accounts	1033	0
LocalizedCollectionName	Cuentas	3082	0
Description	Business that represents a customer or potential customer. The company that is billed in business transactions.	1033	0
Description	Empresa que representa a un cliente o cliente potencial. La empresa a la que se factura en transacciones comerciales.	3082	0

Select a label to see its full layering.

There are other tabs available for specific component solution layers.

Expand table

Tab name	Description	Possible value
RolePrivileges	Displays the privileges for a security role.	Added, Updated, Removed, Unchanged
AttributePicklistValues (optionset)	When selected for a global choice , displays the possible values for a choice .	Added, Updated, Removed, Unchanged
AttributePicklistValues (optionset attribute)	When selected for a choice attribute, displays the values for the attribute.	Added, Updated, Removed, Unchanged

Remove an unmanaged layer

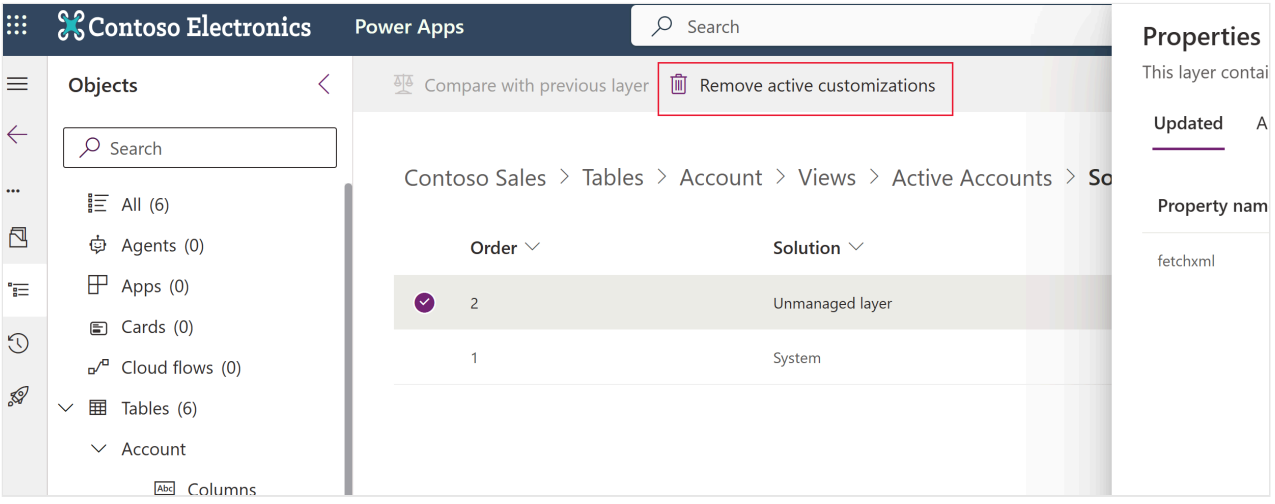
Unmanaged customizations reside at the top layer for a component and subsequently define the runtime behavior of the component. In most situations you don't want unmanaged customizations determining the behavior of your components in your test and production environments.

> **Important**
>
> - By design, your dev and maker environments where you're working with unmanaged components typically have unmanaged layers.
> - If the unmanaged layer you want to remove is the only layer and as such the base layer for the component, you can't remove it with the remove active customizations command. To remove the layer, you have to delete the unmanaged component. More information: Remove objects from a solution.

To remove the unmanaged layer for a component, follow these steps:

> **Warning**
>
> Removing active unmanaged customizations can't be reversed or undone. All data associated with the unmanaged customization can be lost.

1. Open the solution you want, select **...** next to a component, such as **Account**, and then select **See solution layers**.
2. If an unmanaged layer exists, **Unmanaged layer** is displayed in the **Solution** column for the layer.
3. Select the layer, and then on the command bar, select **Remove active customizations**.



See also

Translate localizable text for model-driven apps
Solutions overview

Last updated on 01/09/2026

View the history of a solution

Article • 07/17/2024

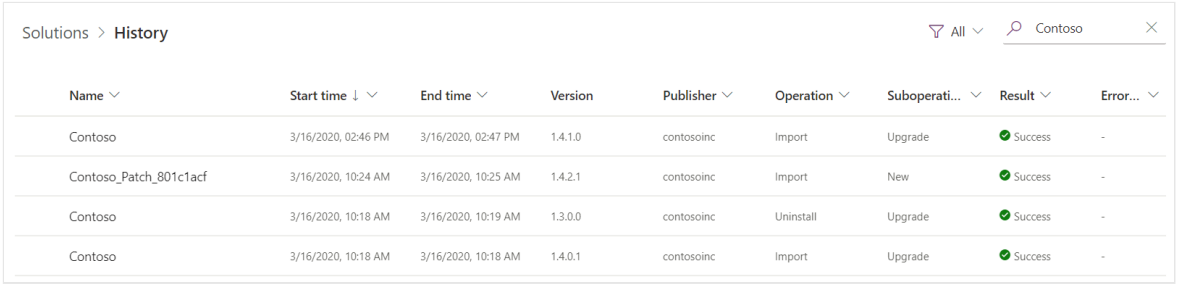
View details about solution operations from the **Solutions** area of Power Apps. An operation can be a solution import, export, or uninstall. The solution history displays information such as solution version, solution publisher, type of operation, operation start and end time, and operation status.

> **Note**
>
> The system automatically deletes solution history records that are older than 180 days.

View solution history

1. Sign in to Power Apps.
2. Select **Solutions** on the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. select the solution you want, and then on the command bar select **See history**.

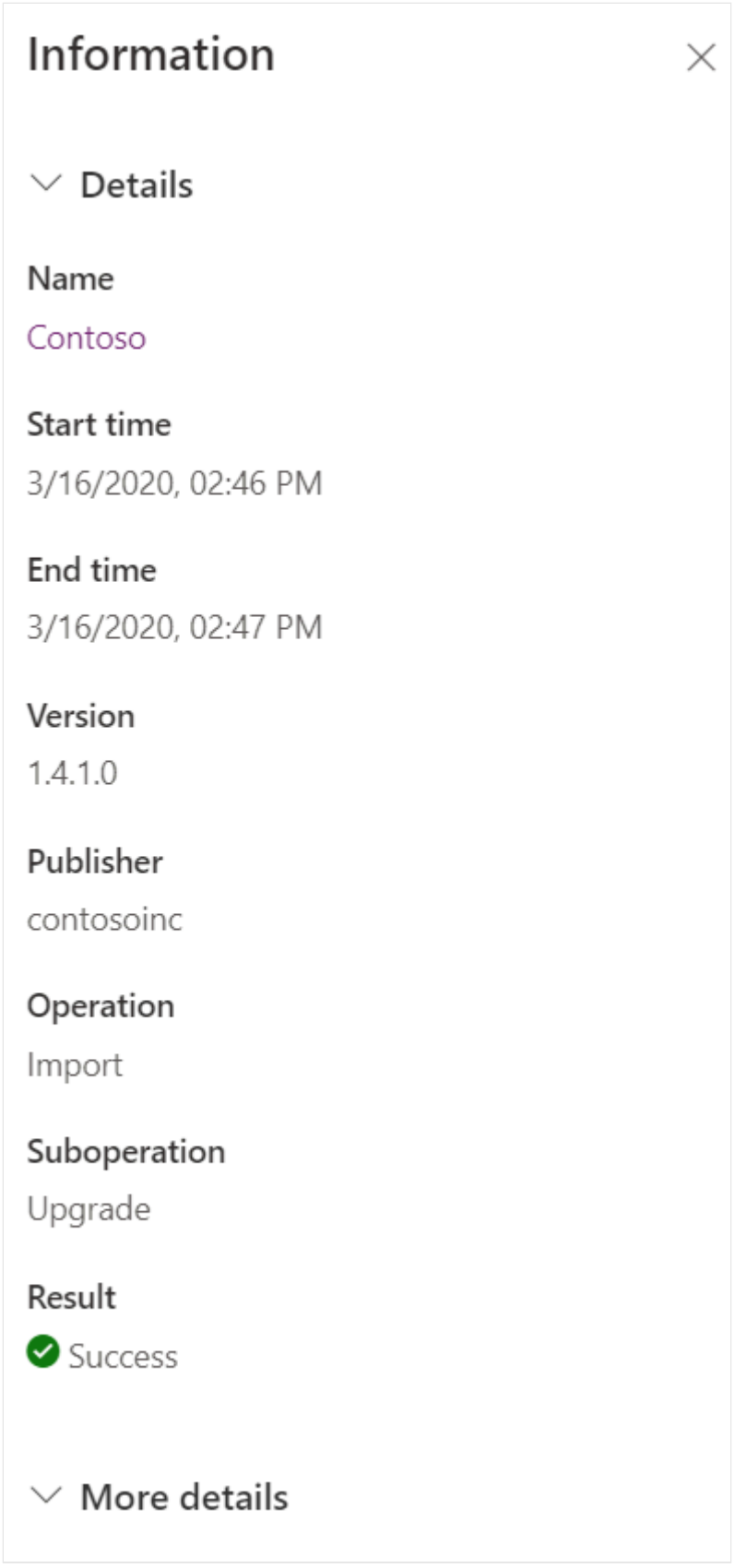
   The history is displayed.



Select a solution operation to display the **Information page**. Each solution history row is read-only and includes the following in the **Details** area:

- **Name**. The solution unique name.
- **Start time**. The time in which the operation started.
- **End time**: The time in which the operation ended.
- **Version**. The version of the solution.
- **Publisher**. The name of the publisher that is associated with the operation.
- **Operation**. The operation, such as import, export, or delete.

- **Suboperation**: Denotes the type of operation, such as a new solution import or an update to an existing solution.
- **Result**. The result of the operation, such as Success or Failure.



Note

Recent changes have optimized the single step upgrade process to no longer use a temporary _Upgrade solution. As a result, you no longer experience an uninstall operation for single stage upgrades in the solution history.

View solution operation error details

Below the **Details** area is the **More details** area that has additional information about the solution and, when a solution operation has a failure, the information includes:

- The **Error code** of the error returned from the operation.
- The **Exception message**, which can help diagnose the underlying cause for the operation failure. Some errors, including solution dependency errors, may also include links to solution layers to make it easier for you to diagnose the issue.
- The **Activity Id** can be useful in cases where you need to contact Microsoft Customer Support.

See also

View solution layers
Solutions overview

Feedback

Was this page helpful? Yes No

Provide product feedback

View dependencies for a component in Power Apps

Article • 03/22/2024

Solution components in Microsoft Dataverse often depend on other solution components. You can’t delete any solution component that has dependencies from another solution component. For example, you can't delete a model-driven app site map component without first deleting the model-driven app component, or removing the dependency, because the site map depends on the model-driven app.

You can view the dependent components from the **Solutions** area of Power Apps.

Show dependencies

The show dependencies page for a component helps identify the dependencies so you can take appropriate action.

This section describes the actions you can take while viewing the dependencies in the **Show dependencies** menu under **Advanced**.

Dependencies page contents

The **Dependencies** page displays details about all the components that have a dependency on the solution component. They're grouped by the solution's name.

The **Dependencies** page has tabs covering reports for **Delete blocked by**, **Used by**, and **Uses**:

- **Delete blocked by**: The report lists all the dependencies that block the delete of the solution component. Unless these dependencies are removed or the component is deleted, the solution component delete remains blocked.
- **Used by**: The report lists all the dependencies of other components that are using this solution component.
- **Uses**: This report lists all the dependencies that the given solution component uses.

Dependencies page actions

The **Dependencies** page has multiple actions you can take for each dependency. The actions can be used to inspect, delete the component, or remove a dependency with the

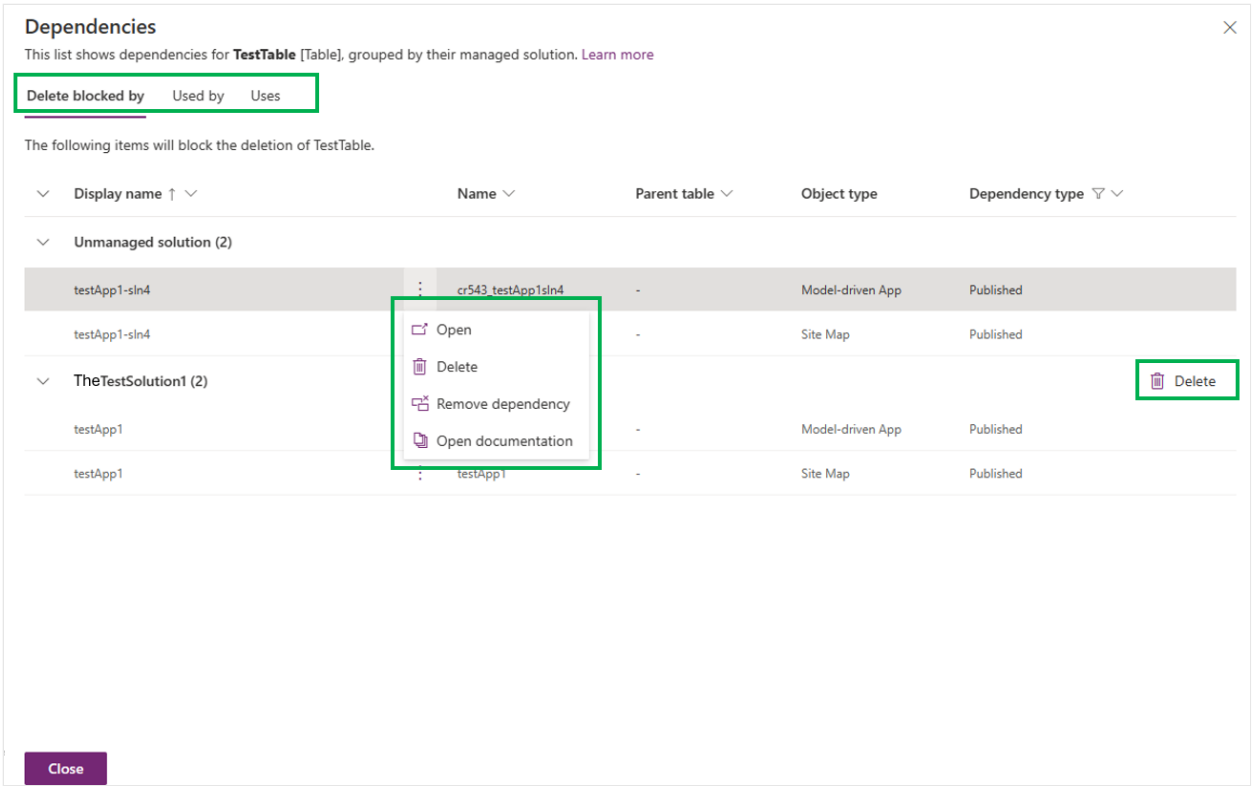
listed dependencies.

To choose an action, select the vertical ellipsis next to any component and then select one of the following actions:

- **Open**: This action takes you to the particular component using the default solution. The open action helps you navigate to the component where you can inspect or edit it.
- **Delete**: This action allows you to delete the component. Ensure to use this action only if the component is no longer required and can be deleted. This action is available only for unmanaged components.
- **Remove dependency**: This action allows you to remove the dependency of the component on the solution component. The system attempts to edit the component to remove the dependency. In the event, when the system is unsuccessful in removing the dependency you might have to do a manual edit. This feature is currently available for the following components:
  - Entity's dependency on WebResource
  - SystemForm's dependency on Attribute and Role
  - AppModule's dependency on SavedQuery, SavedQueryVisualization, Entity, and SystemForm
- **Open documentation**: This action allows you to open documentation where you can learn about the dependencies for that component type.

You can also select a grouped solution and perform delete solution action:

Delete Solution: If you need to delete (uninstall) a solution that has dependencies on the solution component, use **Delete**. This action deletes the solution.



See also

For developers: Detect solution dependencies

Environment variables for Power Platform overview

Environment variables enable the basic application lifecycle management (ALM) scenario of moving an application between Power Platform environments. In this scenario, the application stays exactly the same except for a few key external application references (such as tables, connections, and keys) that are different between the source environment and the destination environment. The application requires the structure of the tables or connections to be exactly the same between the source and the destination environments, with some differences. Environment variables allow you to specify which of these different external references should be updated as the application is moved across environments.

Environment variables store the parameter keys and values, which then serve as input to various other application objects. Separating the parameters from the consuming objects allows you to change the values within the same environment or when you migrate solutions to other environments. The alternative is leaving hard-coded parameter values within the components that use them. This approach is often problematic; especially when the values need to be changed during ALM operations. Because environment variables are solution components, you can transport the references (keys) and change the values when solutions are migrated to other environments.

Benefits of using environment variables:

- Provide new parameter values while **importing solutions** to other environments.
- Store configuration for the **data sources** used in canvas apps and flows. For example, SharePoint Online site and list parameters can be stored as environment variables. This approach allows you to connect to different sites and lists in different environments without needing to modify the apps and flows.
- Package and transport your customization and configuration together and manage them in a single location.
- Package and transport secrets, such as credentials used by different components, separately from the components that use them.
- One environment variable can be used across many different solution components - whether they're the same type of component or different. For example, a canvas app and a flow can use the same environment variable. When the value of the environment variable needs to change, you only need to change one value.
- Additionally, if you need to retire a data source in production environments, you can update the environment variable values with information for the new data source. The apps and flows don't require modification and start using the new data source.
- Supported by SolutionPackager and DevOps tools enable continuous integration and continuous delivery (CI/CD).

- The environment variables can be unpacked and stored in source control. You might also store different environment variables values files for the separate configuration needed in different environments. Solution Packager can then accept the file corresponding to the environment the solution is imported to.

How do environment variables work?

Environment variables can be created and modified within the modern solution interface. They can also be automatically created when connecting to certain data sources in canvas apps or with code. They can also be imported to an environment via solutions. Environment variables can be used as inputs when authoring canvas apps, Power Automate flows, plug-ins, and adding Power BI dashboards to model-driven apps. When you use environment variables, the values are then derived from the environment variables, and can be changed when solutions are imported to other environments.

Manually create an environment variable in a solution

1. Sign in to Power Apps (make.powerapps.com), and then on the left pane select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Open the solution you want or create a new one.
3. On the command bar, select **New** > **More**, and then select **Environment variable**.
4. On the right pane, complete the following columns, and then select **Save**:

    - **Display name**. Enter a name for the environment variable.
    - **Name**. The unique name is automatically generated from the **Display name**, but you can change it.
    - **Data Type**. Select from **Decimal number**, **Text**, **JSON**, **Two options**, **Data source**, or **Secret**.

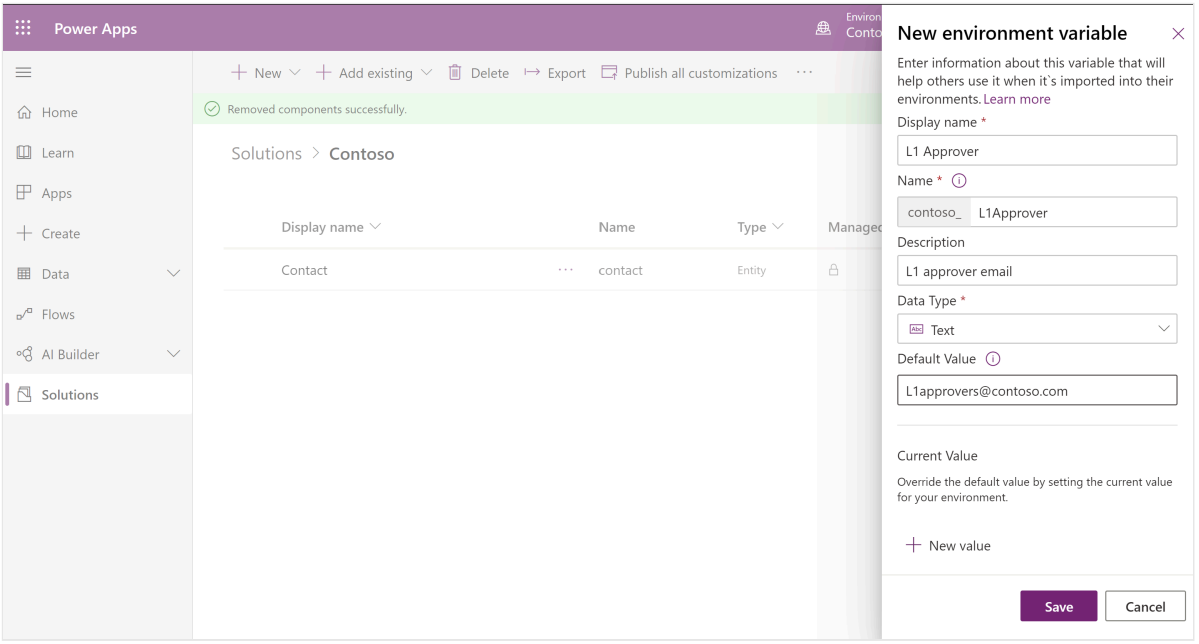
    > ⓘ Note
    > - If **Data source** is the selected type, you need to select the **connector**, a valid **connection** for the selected connector, and the **parameter type**. The connection isn't stored as part of the environment variable. The environment variable stores information not in the connection that is necessary to connect Power Apps to the right server and table. The connection is only used for retrieving available parameter values such as the SharePoint sites you have access to, or the lists associated with a site.

- If **Secret** is the selected type, additional information to set up and configure Azure Key Vault is needed to allow Power Platform to access the secret.

- **Current Value**. Also known as the value. This property is optional and is a part of the environment variable value table. A defined value is used even if a default value is also present. Remove the value from your solution if you don't want to use it in the next environment. The values are also separated into separate JSON files within the exported solution.zip file and can be edited offline. More information: How do I remove a value from an environment variable?

- **Default Value**. This column is part of the environment variable definition table and isn't required. The default value is used if there's no current value.

  Separation of default value and current value allows you to service the definition and the default value separately from the value. For example, an application publisher might list their offer on AppSource with a default value. Then optionally, the customer can provide a new value. When the application publisher publishes updates to the application, the value set by the customer isn't overwritten.

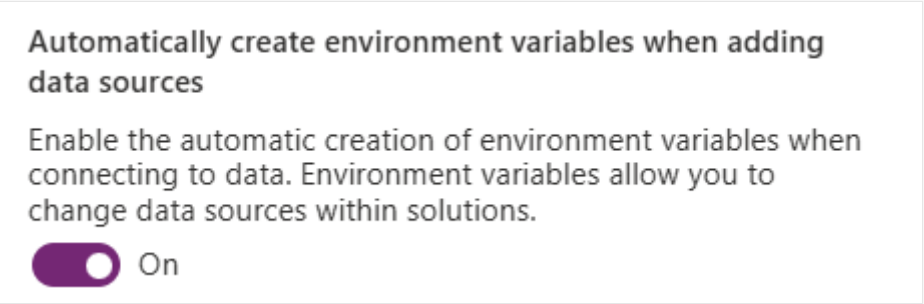


> **Note**
>
> A value can't exist without a definition. The interface only allows creation of one value per definition.

Power Apps can automatically create environment variables

On the **General** tab of Power Apps settings:

- Enable the **Automatically create environment variables when adding data sources** option. Now, when you add a new data source to your app, it adds an environment variable for you.



Different data sources

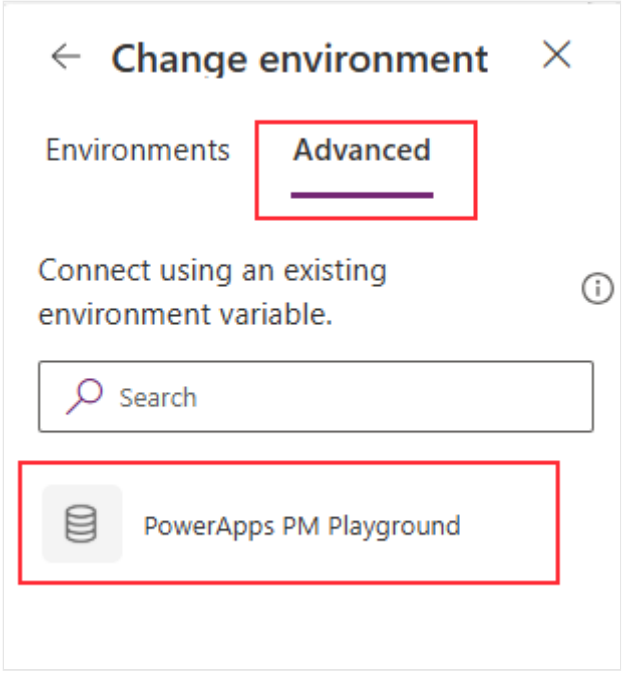
Different data sources enable different types of environment variables.

Microsoft Dataverse

Power Apps connects natively to Dataverse. You don't need an environment variable for tables in your current environment. When you connect to Dataverse, your application automatically looks for exactly the same table name in any environment you port it. As long as your table name structure is the same in the target environment as it is in the original environment, it works.

If you use an external table from a different environment (via Change Environment), then Power Apps assumes you want to always refer to the exact same table. You need an environment variable if you need that external table to be a different external table when you move from dev to test or to prod. It's easiest to allow Power Apps to automatically create the environment variable for you. (If you previously added the table to your app, then turn on the settings switch, drop the external table, and then add it back.)

When you add the external table to your app, choose the **Advanced** tab to select the environment variable.



SharePoint

SharePoint supports only Microsoft Entra connections. So, for SharePoint, in addition to a valid connection, a separate environment variable is required for Site and List.

To successfully use environment variables with SharePoint lists:

- The display name and the logical name for each corresponding column in the source and target environments must match.
- Environment variables for SharePoint must have matching metadata. SharePoint has internal identifiers that might not match between target environments. For example, if you create a list with the same name and columns in a target environment, the internal names won't match. The metadata always matches if you duplicate a SharePoint site and copy it to your target environment.

SQL Server

SQL Server supports many different types of authentication and connections. Use environment variables for Microsoft Entra connections. A separate environment variable is required for the server and the database. The application is bound to specific table names so they're assumed to be the same between environments.

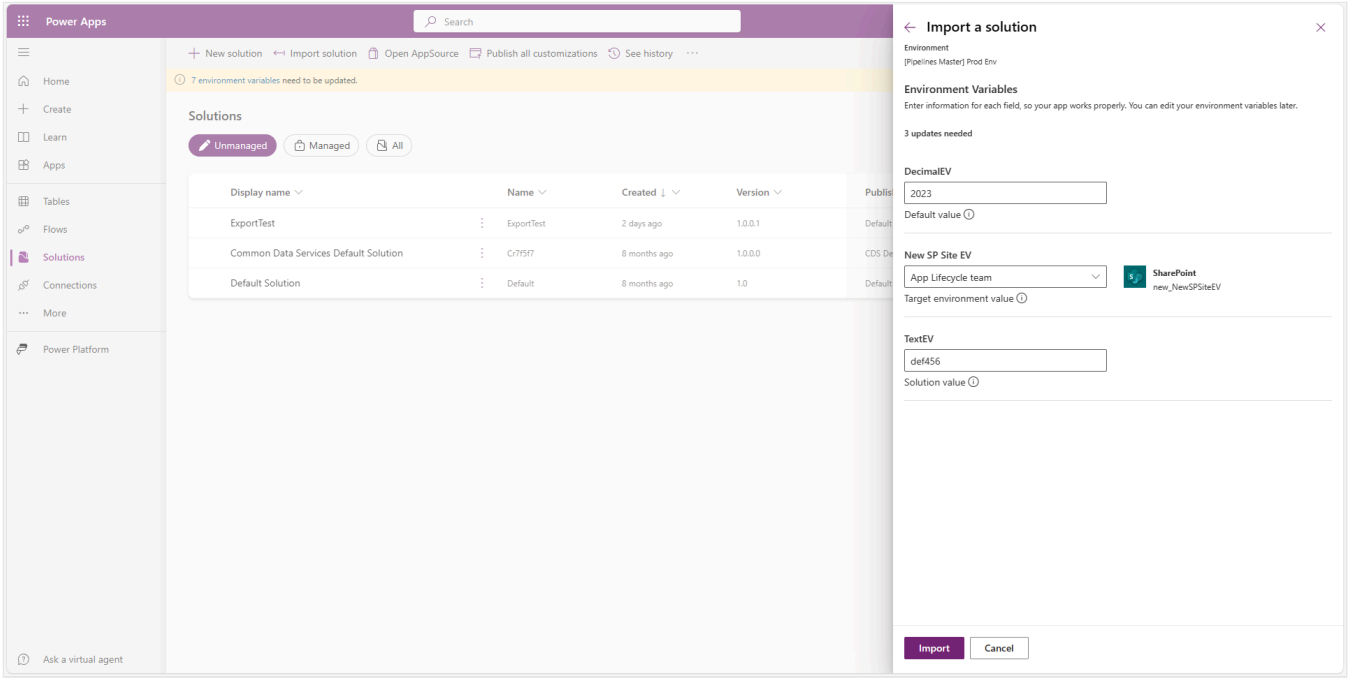
Don't use environment variables for shared connections such as basic SQL authentication with SQL Server. Use connection references for information that is traditionally passed as part of the connection string. Basic SQL auth parameters are all part of the connection string. For example, the Server and Database name are provided when creating the connection and therefore are always derived from the connection.

Data source environment variables are used for connectors with authentication like Microsoft Entra. In these types of connections, Power Apps only has an identified user. Power Apps can't derive what service or table you intend to work with from the connection.

Enter new values while importing solutions

The modern solution import interface includes the ability to enter values for environment variables. This sets the value property on the `environmentvariablevalue` table.

Environment variable values are visible when importing solutions or when using Pipelines to deploy. Environment variables without a default value or value prompt for a value, but those otherwise are prefilled with a label beneath the text area denoting the value's source: solution value, target environment value, or default value.



> **Note**
>
> - In some cases, for specific data source environment variable values, an **Access denied** warning might appear if the importing maker doesn't have access to the connection or source used for the environment variable. This is a nonblocking warning, but something to take note of depending on how you plan to use the environment variable in the target environment.
> - You can remove the value from your solution before exporting the solution. This ensures the existing value remains in your development environment, but won't get exported in the solution. This approach allows a new value to be provided while

importing the solution into other environments. More information: How do I remove a value from an environment variable?

Notifications

A notification is displayed when the environment variables don't have any values. Set the values so that components dependent on environment variables don't fail.

Security

The `environmentvariabledefinition` table is user or team owned. When you create an application that uses environment variables, be sure to assign users the appropriate level of privilege to this table. Permission to the `environmentvariablevalue` table is inherited from the parent `environmentvariabledefinition` table and therefore doesn't require separate privileges. Privileges for `environmentvariabledefinition` tables are included in Environment Maker and Basic User security roles by default. More information: Security in Dataverse.

Naming

Ensure environment variable names are unique so they can be referenced accurately. Duplicate environment variable display names make environment variables difficult to differentiate and use. Ensure environment variable names are unique so they can be referenced accurately. The names **$authentication** and **$connection** are specially reserved parameters for flows and should be avoided. Flow save is blocked if environment variables with those names are used. If an environment variable is used in a flow and the display name of the environment variable is changed, then the designer shows both the old and new display name tokens to help with identification. When updating the flow, we recommend you remove the environment variable reference and add it again.

Current limitations

- Validation of environment variable values happens within the user interfaces and within the components that use them, but not within Dataverse. Therefore ensure proper values are set if they're being modified through code.
- Power Platform Build Tools tasks aren't yet available for managing data source environment variables. However, this doesn't block their usage within Microsoft provided tooling and within source control systems.

- Interacting with environment variables via custom code requires an API call to fetch the values; there isn't a cache exposed for non-Microsoft code to use.
- Environment variables are limited to a maximum of 2,000 characters.

Frequently asked questions

Why can't I see the value for my environment variable?

If the environment variable is in a managed solution, you won't be able to see the value unless you look inside of the **Default solution**. This behavior is by design, since the environment variable value is an unmanaged customization.

How can I view where environment variables are being used?

Either through selecting **Show dependencies** in the solution interface, while authoring components, or in source control and in the solution file by viewing the app or flow metadata.

Are data source environment variables the same as connections?

No. Although they're related, a connection represents a credential or authentication required to interact with the connector. Data source environment variables store parameters that are required by one or more actions in the connector and these parameters often vary depending on the action. For example, a SharePoint Online connection doesn't store any information about sites, lists, or document libraries. Therefore calling the connector requires both a valid connection and some more parameters.

Can my automated ALM pipeline use different values files for different environments?

Yes. Solution packager accepts file name as input parameters so your pipeline can pack a different values file into the solution depending on the environment type it’s executing against.

Should I include the value in my solution?

No. Environment variables are intended to be used by applications that need to have different values in different environments where the solution is deployed. Environment variable definitions should be included in your solution but the values should be provided for the target environment during deployment. This behavior results in the environment variable definition

being a managed solution object in the target environment while the environment variable is an unmanaged record.

Why can't I delete a value in my environment?

If the value was included in a managed solution, the only way to delete the value is to update the solution in the source environment to exclude the value, then export a new version of the solution. This new version then can be imported into the environment using an upgrade operation (not an update), resulting in the value record being deleted.

What if someone inadvertently deletes a value?

If not already prevented by dependency system, runtime uses the last known value as a fallback.

If a value is changed, when does the new value get used in canvas apps and cloud flows?

It might take up to an hour to fully publish updated environment variables because the value is pushed into the apps and flows asynchronously.

Are premium licenses required?

No. While ALM requires Dataverse (or Dynamics 365 apps), use of premium connectors isn't required. The one caveat is if you're using the Dataverse connector to interact with environment variables as you would with other data records like accounts or contacts. Previously this was the only way to use environment variables in canvas apps and flows.

Is there a limit to the number of environment variables I can have?

No. However, the max size of a solution is 95 MB. More information: Create a solution

Can environment variable display names and descriptions be localized?

Yes.

Should I use environment variables instead of storing configuration data in custom tables?

Yes if your configuration data isn't relational. Environment variables should be used for key: value pairs and when the value likely needs to different in other environments. Other tools such as the Configuration migration utility are better suited for migration of relational configuration data stored within custom tables. Unlike other configuration data, environment variables are migrated within solutions and therefore much simpler to manage and more performant to import.

Why isn't my connection saved for data source environment variables?

Connections are only used to generate the user interface. For example, to retrieve friendly names for SharePoint lists associated with a site. When editing a data source environment variable, be sure to select a connection with access to the data source. If you no longer have access to the connection, you can manually enter a value. For example, enter the SharePoint site URL or unique identifier (GUID) for a list.

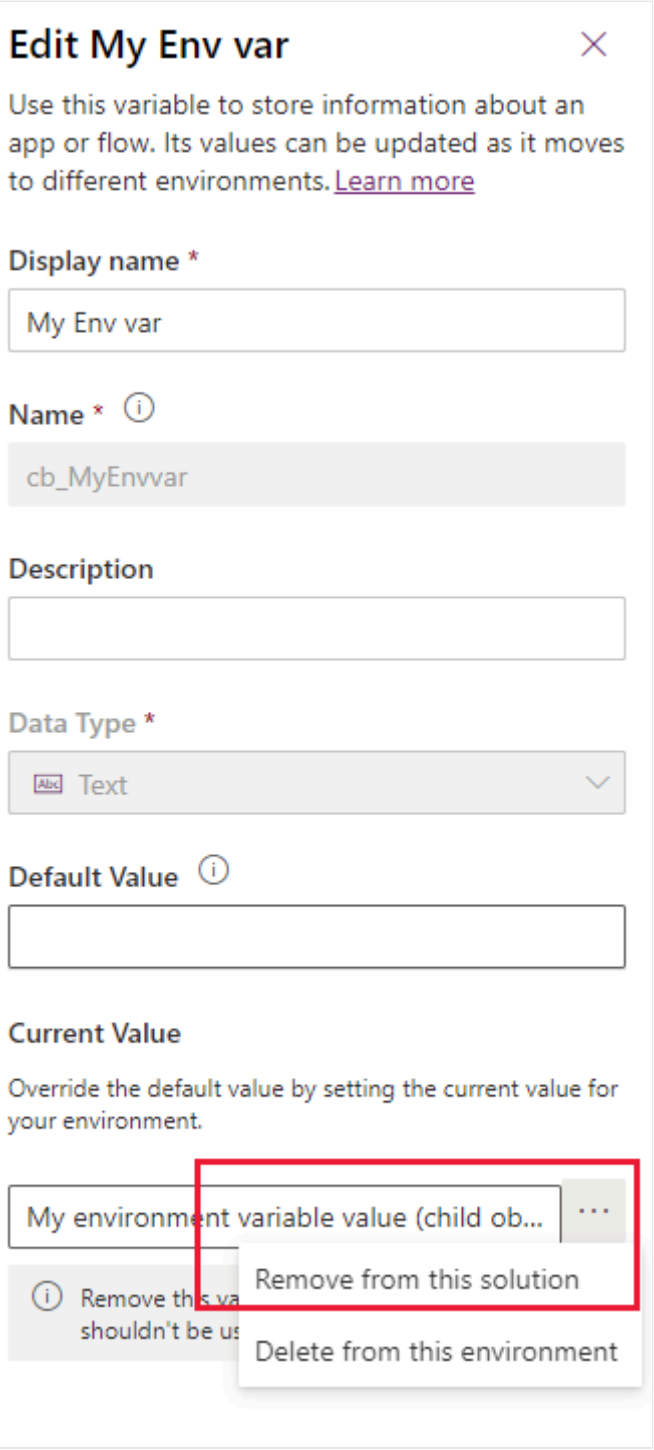
When there are multiple connections available for a single (data source-type) environment variable, there's a by-design implementation to select the first connection in the list of connections. Because there's usually only one connection associated with an environment variable, this isn't something that usually needs to be validated. Additionally, with recent changes to environment variable value visibility, this is easier to validate upon import.

How do I remove a value from an environment variable?

You might want to remove the value of an environment variable from your solution before exporting the solution. Then, the existing value remains in your development environment, but isn't exported in the solution. This approach allows a new value to be provided while importing the solution into another environment.

To remove the value, follow these steps:

1. In the solution where the environment variable is located select the environment variable to display the properties.
2. Under **Current Value**, select **...** > **Remove from this solution**.



Can I use environment variables in custom connectors?

Yes. Environment variable support in custom connectors

See also

Use data source environment variables in canvas apps
Use environment variables in Power Automate solution cloud flows
EnvironmentVariableDefinition table/entity reference
Web API samples
Use data source environment variables in Canvas apps

Use environment variables in Power Automate solution cloud flows
Use Azure Key Vault secrets
Environment variable support in custom connectors
Power Apps Blog: Environment variables available in preview!

Last updated on 01/09/2026

Use data source environment variables in canvas apps

In this article, you learn about using data source environment variables in canvas apps. You can either use pre-existing data source environment variables, or create data source environment variables automatically when connecting to data.

Use pre-existing data source environment variables

Environment variables can be reused across other apps and even different types of resources like cloud flows. You might wish to first create them within your solution and later use them while authoring canvas apps and cloud flows.

1. Follow the steps to manually create an environment variable in a solution.
2. Edit or create a canvas app from your solution.
3. Add a **new** data source for SharePoint online.
4. Select the **Advanced** tab. You'll see a filtered list of environment variables that you have access to and that match the parameter being set. For example, when you select the SharePoint site, you'll see a list of all data source environment variables with **Connector** as **SharePoint** and **Parameter type** as **Site**. The same is true when selecting SharePoint lists for a given site.
5. Select the desired environment variable(s), and then select **Connect.**

> **Important**
>
> - If an environment variable from a different solution is selected, a dependency will exist on the solution containing the environment variable. Therefore, be sure to either:
>   - Add the environment variable to your current solution prior to exporting.
>   - Ensure the solution containing the environment variable is imported to the destination environment before your current solution is imported.

Automatically create data source environment variables when connecting to data

This option provides simplicity and ensures environment variables are always used for data sources, such as SharePoint Online. However, some customers prefer to provide their own schema names and therefore should create them from solutions.

> **Note**
>
> Pre-existing canvas apps don't automatically use data source environment variables. Remove the data source from the app and add them back using the above steps to upgrade these apps to use environment variables.

1. Edit or create a canvas app from your solution.
2. Select **Settings** > **General** and enable the setting to **Automatically create environment variables when adding data sources**.
3. Add a **New** data source for SharePoint online.
4. Select a SharePoint **site**, one or more **lists**, and then **Connect**.

   > **Note**
   >
   > To prevent creation of duplicate environment variables, you're prompted to use the existing environment variable when duplicates are identified. You can clear the option to use the existing environment variable if creation of a duplicate is desired.

5. Select **Save**.

See also

Create canvas app from scratch using Dataverse.
Use environment variables in Power Automate solution cloud flows
Environment variables overview

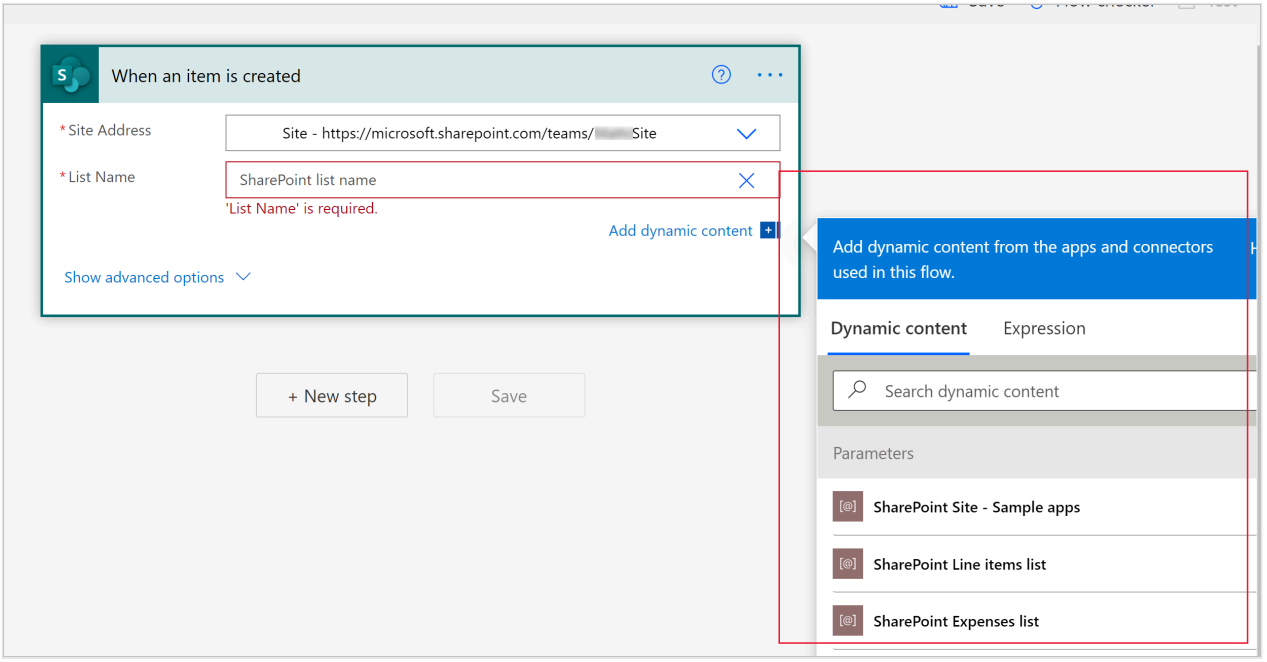
Last updated on 04/27/2026

Use environment variables in Power Automate solution cloud flows

Environment variables can be used in solution cloud flows since they're available in the dynamic content selector. All types of environment variables can be used in triggers and actions.

To use an environment variable in a solution cloud flow:

1. Edit or create a cloud flow in a solution.
2. In an action or a trigger, determine the parameter that you want to use for the environment variable:
   a. If the parameter takes a simple value, such as a string or number, enter the parameter.
   b. If the parameter is a lookup, scroll to the bottom of the lookup, and then select **Enter custom value**. Environment variables that you have access to are listed in the **Dynamic content** selector with other dynamic content.



3. Select the desired environment variable.

Limitations

- When environment variable values are changed directly within an environment instead of through an ALM operation like solution import, flows will continue using the previous value

until the flow is either saved or turned off and turned on again.

- When editing a cloud flow, the environment variables shown in the dynamic content selector are unfiltered, but will be filtered by data type in the future.
- When editing a cloud flow, if an environment variable is added in another browser tab, the flow needs to be reopened in the flow designer to refresh the dynamic content selector.

See also

Use data source environment variables in canvas apps

Create a flow with Dataverse

Environment variables overview.

Last updated on 04/27/2026

Use environment variables for Azure Key Vault secrets

Environment variables allow for referencing secrets stored in Azure Key Vault. These secrets are then made available for use within Power Automate flows and custom connectors. Notice that the secrets aren't available for use in other customizations or generally via the API.

The actual secrets are stored in Azure Key Vault and the environment variable references the key vault secret location. Using Azure Key Vault secrets with environment variables require that you configure Azure Key Vault so that Power Platform can read the specific secrets you want to reference.

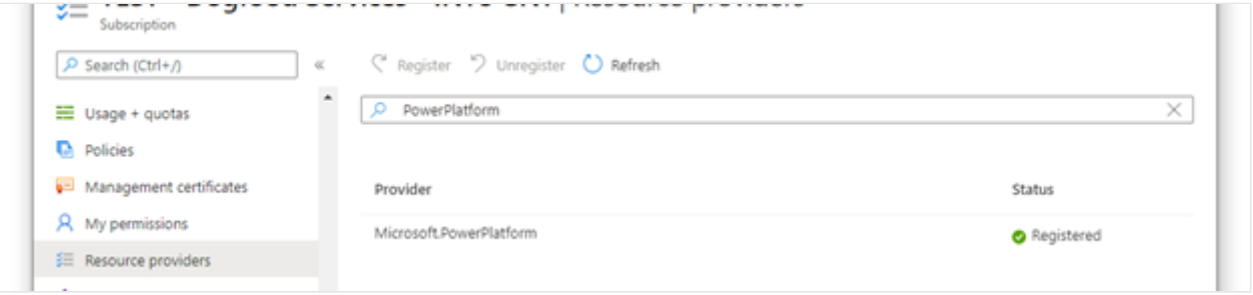
Environment variables referencing secrets aren't currently available from the dynamic content selector for use in Power Automate flows.

Configure Azure Key Vault

To use Azure Key Vault secrets with Power Platform, the Azure subscription that has the vault must have the `PowerPlatform` resource provider registered and the user who creates the environment variable must have appropriate permissions to the Azure Key Vault resource.

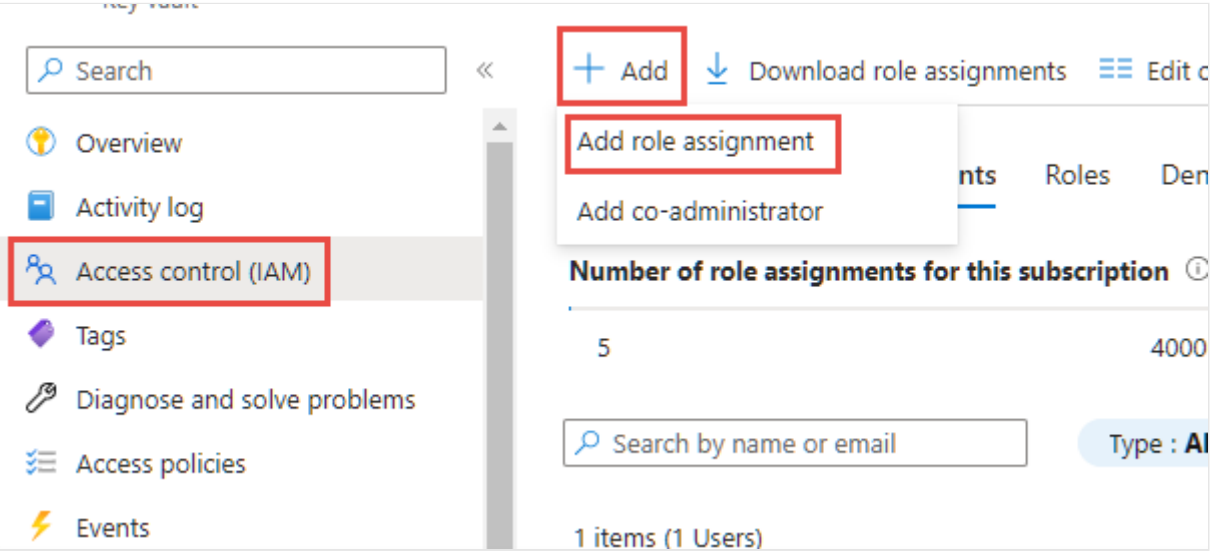
> **ⓘ Important**
>
> - There are recent changes to the security role used to assert access permissions within Azure Key Vault. Previous instructions included assigning the Key Vault Reader role. If you have set up your key vault previously with the Key Vault Reader role, make sure that you add the Key Vault Secrets User role to ensure that your users and Microsoft Dataverse have sufficient permission to retrieve the secrets.
> - We recognize that our service is using the Azure role-based access control APIs to assess security role assignment even if you still have your key vault configured to use the vault access policy permission model. To simplify your configuration, we recommended that you switch your vault permission model to Azure role-based access control. You can do this on the **Access configuration** tab.

1. Register the `Microsoft.PowerPlatform` resource provider in your Azure subscription. Follow these steps to verify and configure: Resource providers and resource types



2. Create an Azure Key Vault vault. Consider using a separate vault for every Power Platform environment to minimize the threat in case of a breach. Consider configuring your key vault to use **Azure role-based access control** for the **Permission model**. More information: Best practices for using Azure Key Vault, Quickstart - Create an Azure Key Vault with the Azure portal

3. Users who create or use environment variables of type secret must have permission to retrieve the secret contents. To grant a new user the ability to use the secret, select the **Access control (IAM)** area, select **Add**, and then select **Add role assignment** from the dropdown. More information: Provide access to Key Vault keys, certificates, and secrets with an Azure role-based access control



4. On the **Add role assignment** wizard, leave the default assignment type as **Job function roles** and continue to the **Role** tab. Locate the **Key Vault Secrets User role** and select it. Continue to the members tab and select the **Select members** link and locate the user in the side panel. When you have the user selected and shown on the members section, continue to the review and assign tab and complete the wizard.

5. Azure Key Vault must have the **Key Vault Secrets User** role granted to the Dataverse service principal. If it doesn't exist for this vault, add a new access policy using the same method you previously used for the end user permission, only using the Dataverse application identity instead of the user. If you have multiple Dataverse service principals in your tenant, then we recommend that you select them all and save the role assignment. Once the role is assigned, review each Dataverse item listed in the role assignments list

and select the Dataverse name to view the details. If the Application ID isn't 00000007-0000-0000-c000-000000000000**, then select the identity, and then select **Remove** to remove it from the list.

6. If you have enabled Azure Key Vault Firewall, you have to allow Power Platform IP addresses access to your key vault. Power Platform isn't included in the "Trusted Services Only" option. Go to Power Platform URLs and IP address ranges for the current IP addresses used in the service.

7. If you haven't done so already, add a secret to your new vault. More information: Azure Quickstart - Set and retrieve a secret from Key Vault using Azure portal

Create a new environment variable for the Key Vault secret

Once Azure Key Vault is configured and you have a secret registered in your vault, you can now reference it within Power Apps using an environment variable.

> **Note**
>
> - User access validation for the secret is performed in the background. If the user doesn't have at least read permission, this validation error is displayed: "This variable didn't save properly. User isn't authorized to read secrets from 'Azure Key Vault path'."
> - Currently, Azure Key Vault is the only secret store that is supported with environment variables.
> - The Azure Key Vault must be in the same tenant as your Power Platform subscription.

1. Sign on to Power Apps, and in the **Solutions** area, open the unmanaged solution you're using for development.

2. Select **New** > **More** > **Environment variable**.

3. Enter a **Display name** and optionally, a **Description** for the environment variable.

4. Select the **Data Type** as **Secret** and **Secret Store** as **Azure Key Vault**.

5. Choose from the following options:

   - Select **New Azure Key Vault value reference**. After the information is added in the next step and saved, an environment variable *value* record is created.

- Expand **Show default value**, to display the fields to create a **Default Azure Key Vault secret**. After the information is added in the next step and saved, the default value demarcation is added to the environment variable *definition* record.

6. Enter the following information:

   - **Azure Subscription ID**: The Azure subscription ID associated with the key vault.
   - **Resource Group Name**: The Azure resource group where the key vault that contains the secret is located.
   - **Azure Key Vault Name**: The name of the key vault that contains the secret.
   - **Secret Name**: The name of the secret located in Azure Key Vault.

   > **Tip**
   >
   > The subscription ID, resource group name, and key vault name can be found on the Azure portal **Overview** page of the key vault. The secret name can be found on the key vault page in the Azure portal by selecting **Secrets** under **Settings**.

7. Select **Save**.

Create a Power Automate flow to test the environment variable secret

A simple scenario to demonstrate how to use a secret obtained from Azure Key Vault is to create a Power Automate flow to use the secret to authenticate against a web service.

> **Note**
>
> The URI for the web service in this example isn't a functioning web service.

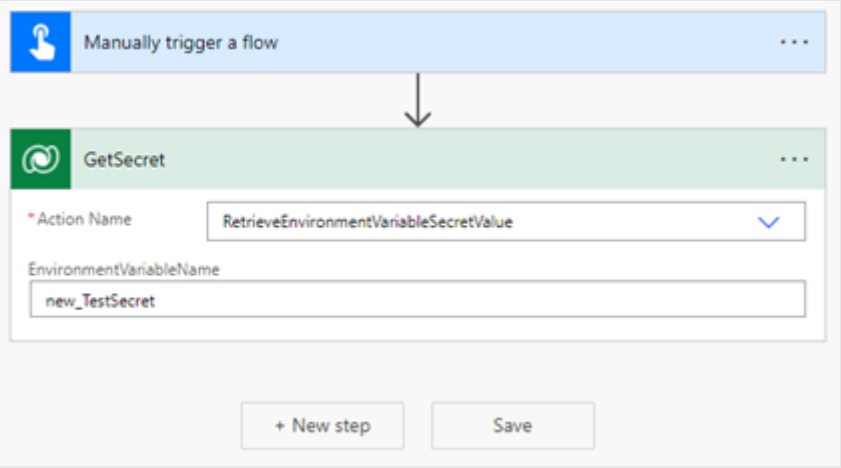
1. Sign into Power Apps, select **Solutions**, and then open the unmanaged solution you want. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Select **New** > **Automation** > **Cloud flow** > **Instant**.
3. Enter a name for the flow, select **Manually trigger a flow**, and then select **Create**.

4. Select **New step**, select the **Microsoft Dataverse** connector, and then on the **Actions** tab select **Perform an unbound action**.

5. Select the action named **RetrieveEnvironmentVariableSecretValue** from the dropdown list.

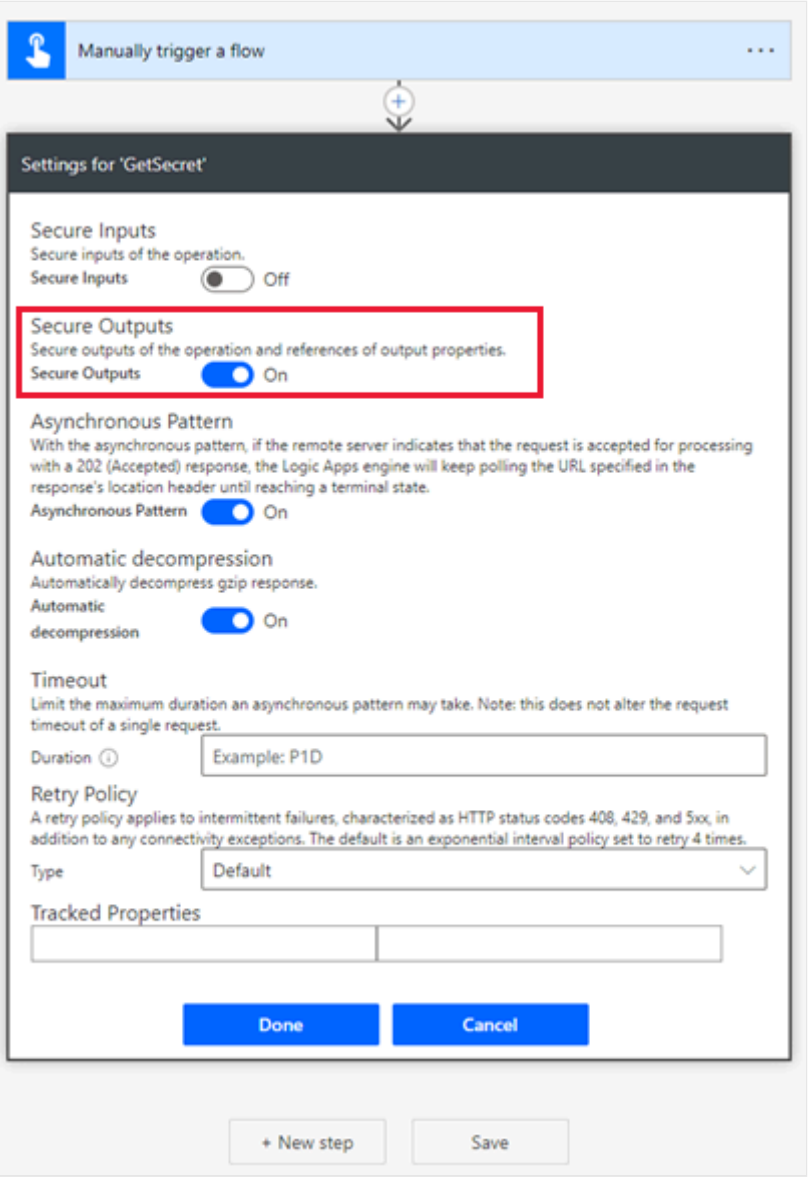
6. Provide the environment variable unique name (not the display name) added in the previous section, for this example *new_TestSecret* is used.

7. Select **...** > **Rename** to rename the action so that it can be more easily referenced in the next action. In this screenshot, it's renamed to **GetSecret**.



8. Select **...** > **Settings** to display the **GetSecret** action settings.

9. Enable the **Secure Outputs** option in the settings, and then select **Done**. This is to prevent the output of the action getting exposed in the flow run history.

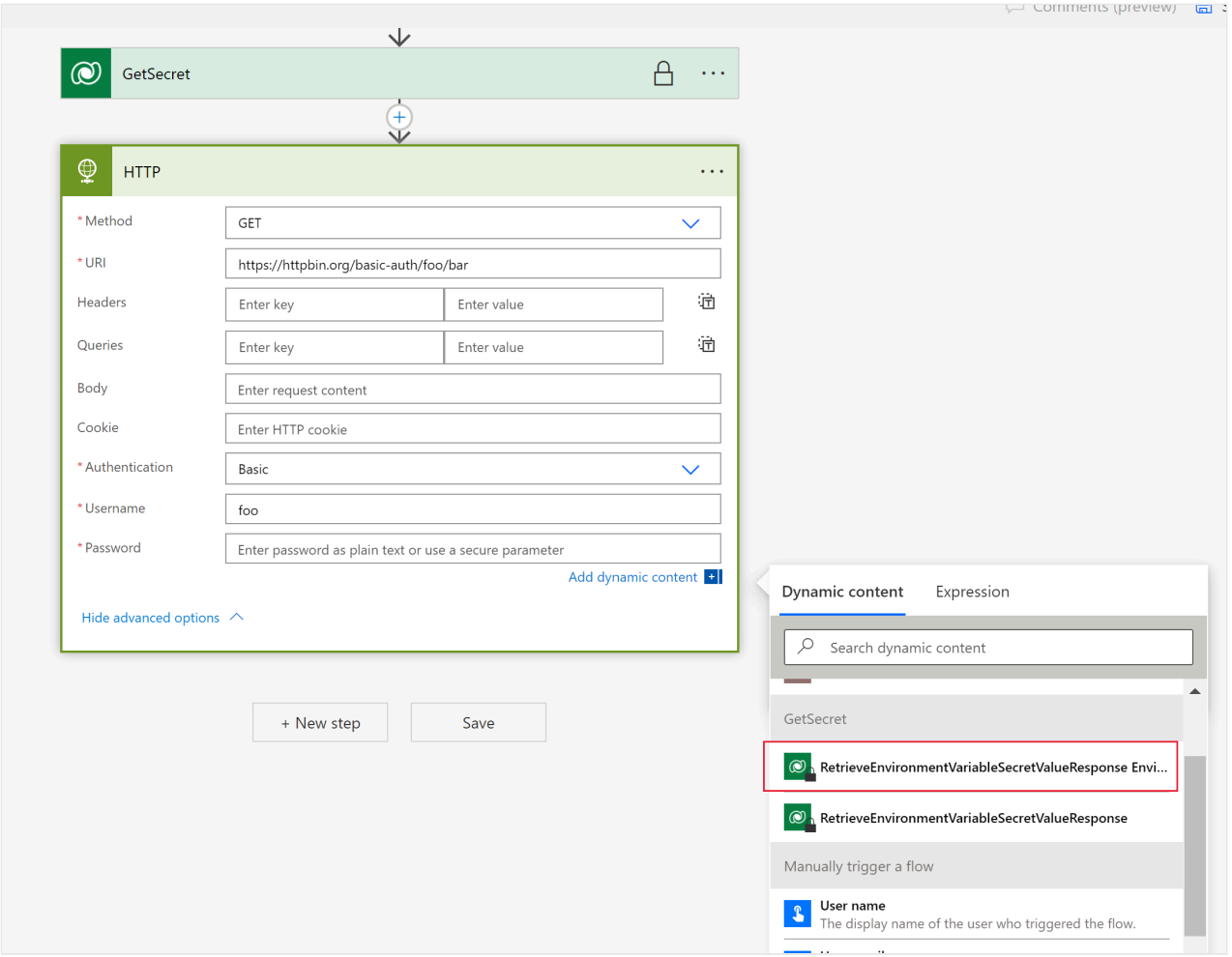


10. Select **New step**, search, and select the **HTTP** connector.

11. Select the **Method** as **GET** and enter the **URI** for the web service. In this example, the fictitious web service *httpbin.org* is used.

12. Select **Show advanced options**, select the **Authentication** as **Basic**, and then enter the **Username**.

13. Select the **Password** field, and then on the **Dynamic content** tab under the flow step name above (*GetSecret* in this example) select **RetrieveEnvironmentVariableSecretValueResponse EnvironmentVariableSecretValue**, which is then added as an expression `outputs('GetSecretTest')?['body/EnvironmentVariableSecretValue']` or `body('GetSecretTest')['EnvironmentVariableSecretValue']`.



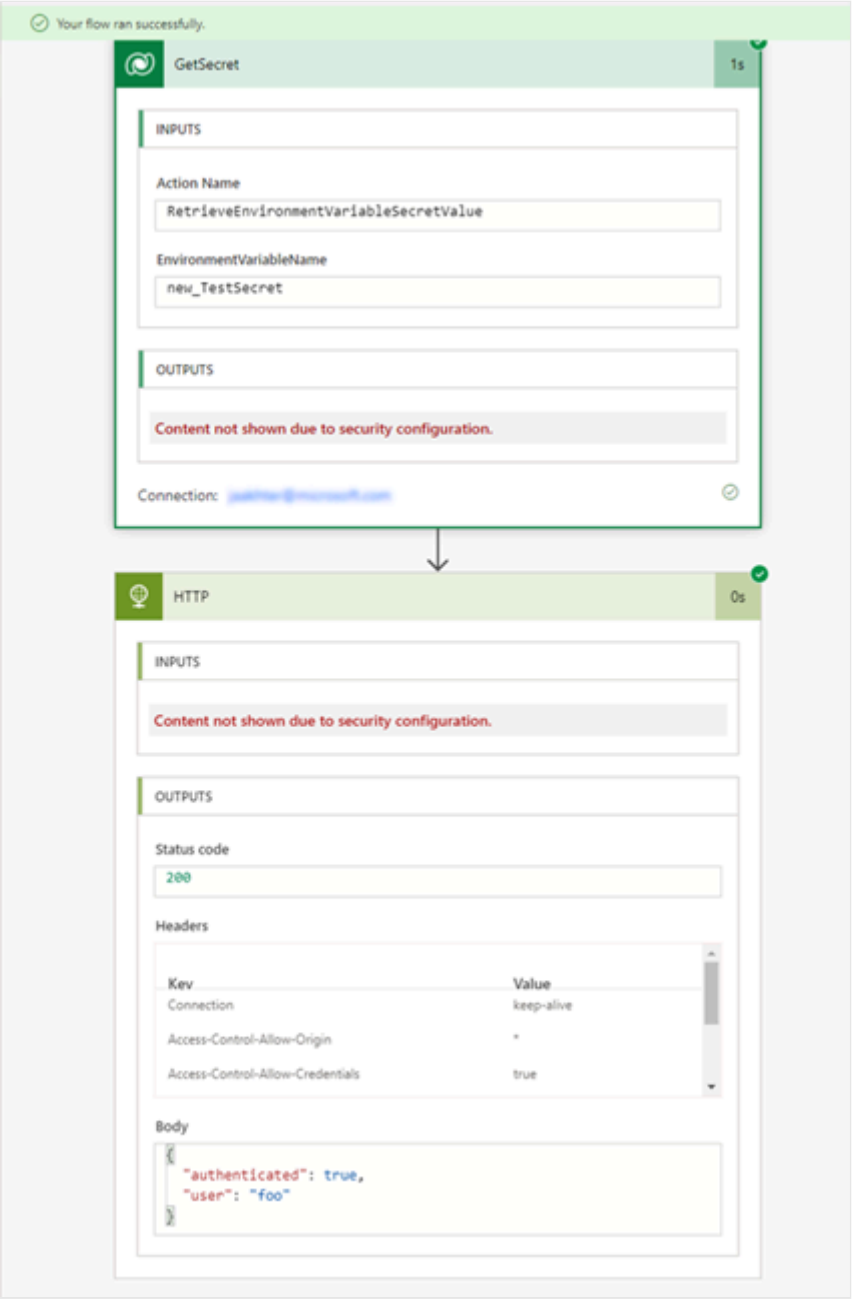
14. Select **...** > **Settings** to display the **HTTP** action settings.

15. Enable the **Secure Inputs** and **Secure Outputs** options in the settings, and then select **Done**. Enabling these options prevents the input and outputs of the action getting exposed in the flow run history.

16. Select **Save** to create the flow.

17. Manually run the flow to test it.

    Using the run history of the flow, the outputs can be verified.



Use environment variable secrets in Microsoft Copilot Studio

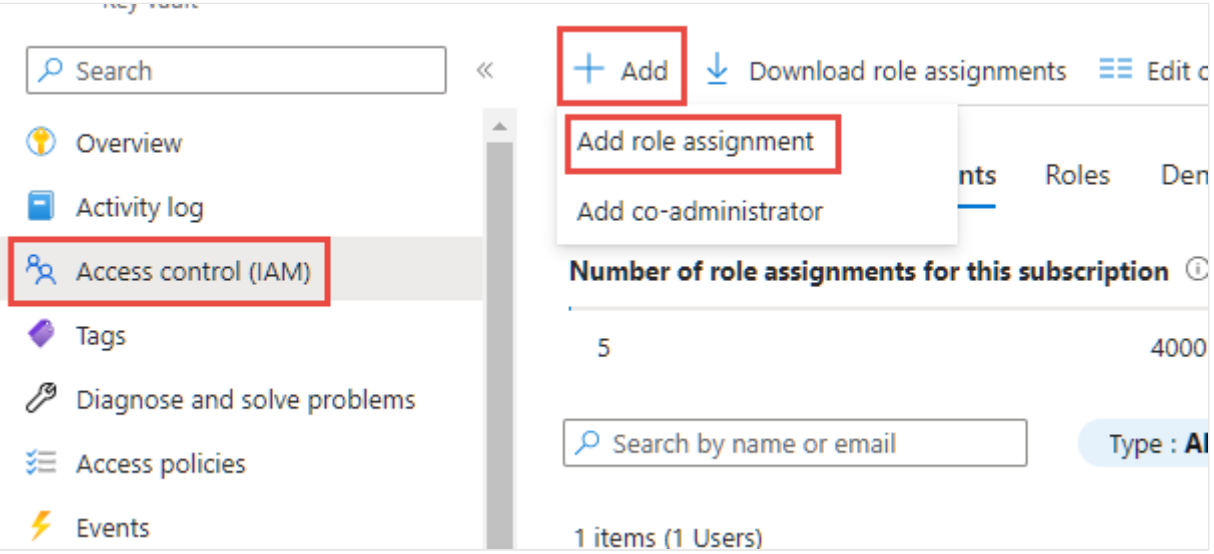
Environment variable secrets in Microsoft Copilot Studio work a bit differently. You need to run through the steps in the sections in Configure Azure Key Vault and Create a new environment variable for the Key Vault secret to use secrets with environment variables.

Give Copilot Studio access to Azure Key Vault

Follow these steps:

1. Go back to your Azure Key Vault.

2. Copilot Studio needs access to the key vault. To grant Copilot Studio the ability to use the secret, select **Access control (IAM)** on the left navigation, select **Add**, and then select **Add role assignment**.



3. Select the **Key Vault Secrets User** role, and then select **Next**.

4. Select **Select Members**, search for *Microsoft Copilot Studio Service*, select it, and then choose **Select**.

5. Select **Review + assign** on the bottom of the screen. Review the information and select **Review + assign** again if all is correct.

Add a tag to allow a copilot to access the secret in Azure Key Vault

By completing the previous steps in this section, Copilot Studio now has access to the Azure Key Vault, but you can't use it yet. To complete the task, follow these steps:

1. Go to Microsoft Copilot Studio and open the agent you want to use for the environment variable secret or create a new one.

2. Open an agent topic or create a new one.

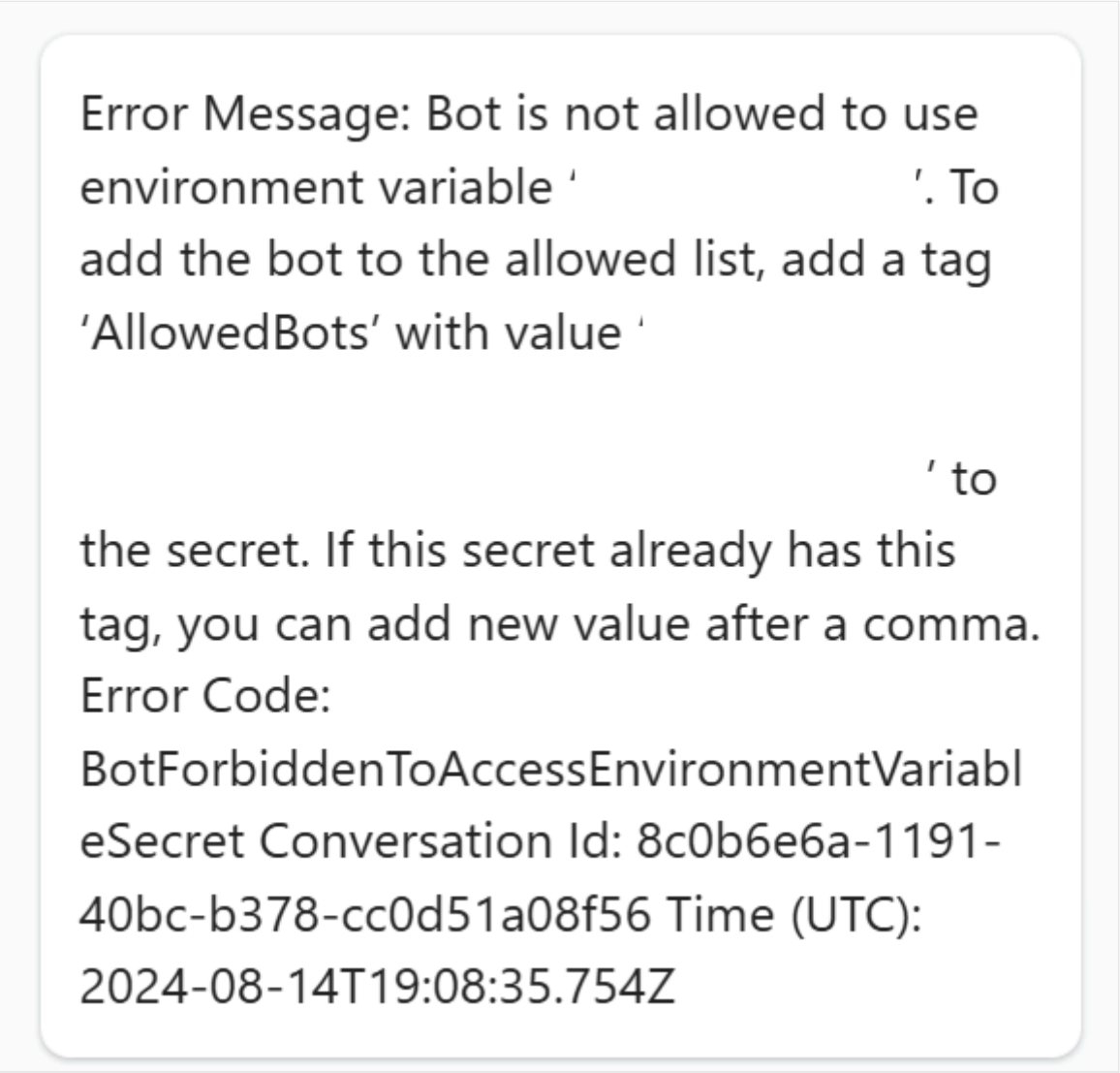
3. Select the + icon to add a node, and then select **Send a message**.

4. Select the **Insert variable {x}** option in the **Send a message** node.

5. Select the **Environment** tab. Select the environment variable secret you created in the Create a new environment variable for the Key Vault secret step.

6. Select **Save** to save your topic.

7. In the test pane, test your topic by using one of the start phrases of the topic where you just added the **Send a message** node with the environment variable secret. You should run into an error that looks like this:



   This means you need to go back to Azure Key Vault and edit the secret. Leave Copilot Studio open, because you come back here later.

8. Go to Azure Key Vault. In the left navigation, select **Secrets** under **Objects**. Select the secret you want to make available in Copilot Studio by selecting the name.

9. Select the version of the secret.

10. Select **0 tags** next to **Tags**. Add a **Tag Name** and a **Tag Value**. The error message in Copilot Studio should give you the exact values of those two properties. Under **Tag Name** you need to add **AllowedBots** and in **Tag Value** you need to add the value that was displayed in the error message. This value is formatted as `{envId}/{schemaName}`. If there are multiple copilots that need to be allowed, separate the values with a comma. When done, select **OK**.

11. Select **Apply** to apply the tag to the secret.

12. Go back to Copilot Studio. Select **Refresh** in the **Test your copilot** pane.

13. In the test pane, test your topic again by using one of the start phrases of the topic.

The value of your secret should be shown in the test panel.

Add a tag to allow all copilots in an environment access to the secret in Azure Key Vault

Alternatively, you can allow all copilots in an environment access to the secret in Azure Key Vault. To complete the task, follow these steps:

1. Go to Azure Key Vault. In the left navigation, select **Secrets** under **Objects**. Select the secret you want to make available in Copilot Studio by selecting the name.
2. Select the version of the secret.
3. Select **0 tags** next to **Tags**. Add a **Tag Name** and a **Tag Value**. Under **Tag Name** add **AllowedEnvironments** and in **Tag Value** add the environment ID of the environment you want to allow. When done, select **OK**
4. Select **Apply** to apply the tag to the secret.

Limitation

Environment variables referencing Azure Key Vault secrets are currently limited for use with Power Automate flows, Copilot Studio agents, and custom connectors.

Integrate Azure Key Vault private link with environment variables

Using Azure Key Vault secrets with environment variables requires configuring Azure Key Vault so that Power Platform can read the specific secrets you want to reference. This capability enables support for environment variables with Azure Key Vault secrets connecting via a private link, enhancing security and providing a more robust integration.

1. Set up Azure Virtual Network support for Power Platform to integrate environment variables with Azure Key Vault secrets without exposing them to the public internet. For detailed instructions, go to set up virtual network.

2. Ensure that the Azure subscription for Key Vault and Power Platform Virtual Network are in the same tenant, as cross-tenant integration isn't supported.

3. Ensure that the user who creates the environment variables has appropriate permissions to the Azure Key Vault resource. For more details, go to Configure the Azure key Vault

4. Create a key vault and establish a private link connection. The steps to create the key vault should include the following actions:

    - Disable public access.
    - Create a private endpoint.
    - Select the virtual network and subnet where you want this private endpoint to be created. Ensure to connect the Virtual Network (virtual network) delegated to Power Platform.
    - Validate the private link connectivity.

    For detailed steps, go to Set up Virtual Network support for Power Platform.

5. Create environment variables secrets by linking to the Azure Key Vault.

See also

Use data source environment variables in canvas apps
Use environment variables in Power Automate solution cloud flows
Environment variables overview.

Last updated on 01/09/2026

Environment variables frequently asked questions

Article • 10/25/2024

In this article find the frequently asked question (FAQs) for environment variables.

Why can't I see the value for my environment variable?

If the environment variable is in a managed solution, you won't be able to see the value unless you look inside of the **Default solution**. This behavior is by design, since the environment variable value is an unmanaged customization.

How can I view where environment variables are being used?

Either through selecting **Show dependencies** in the solution interface, while authoring components, or in source control and in the solution file by viewing the app or flow metadata.

Are data source environment variables the same as connections?

No. Although they're related. A connection represents a credential or authentication required to interact with the connector. Data source environment variables store parameters that are required by one or more actions in the connector and these parameters often vary depending on the action. For example, a SharePoint Online connection doesn't store any information about sites, lists, or document libraries. Therefore calling the connector requires both a valid connection and some additional parameters.

Can data source environment variables be used with shared connections such as SQL Server with SQL authentication?

Generally no. Shared connections with SQL Server store the parameters required to connect to data within the connection. For example, the Server and Database name are provided when creating the connection and therefore are always derived from the connection.

Data source environment variables are used for connectors that rely on user based authentication such as Microsoft Entra ID because the parameters can't be derived from the connection. For these reasons authentication with SQL Server, which is a shared connection, won't use data source environment variables.

Can my automated ALM pipeline use different values files for different environments?

Yes. Solution packager accepts file name as input parameters so your pipeline can pack a different values file into the solution depending on the environment type it’s executing against.

What if someone inadvertently deletes a value?

If not already prevented by dependency system, runtime uses the last known value as a fallback.

If a value is changed, when does the new value get used in canvas apps and cloud flows?

It may take up to an hour to fully publish updated environment variables because the value is pushed into the apps and flows asynchronously.

Are premium licenses required?

No. While ALM requires Dataverse (or Dynamics 365 for Customer Engagement), use of premium connectors isn't required. The one caveat is if you're using the Dataverse connector to interact with environment variables as you would with other data records like accounts or contacts. Previously this was the only way to use environment variables in canvas apps and flows.

Is there a limit to the number of environment variables I can have?

No, there is no hard limit to the number of environment variables allowed in an environment, but user experience will become difficult when there are too many environment variables to select from.

Can environment variable display names and descriptions be localized?

Yes.

Should I use environment variables instead of storing configuration data in custom tables?

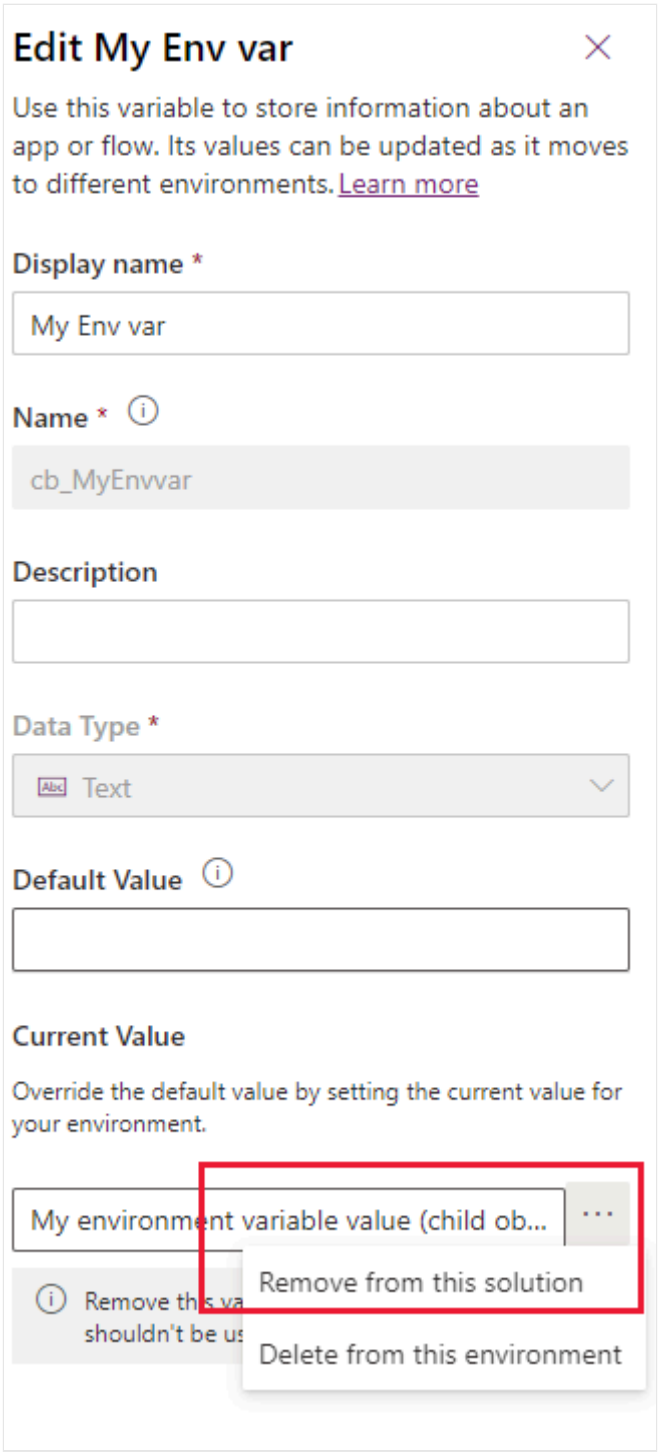
Yes if your configuration data isn't relational. Environment variables should be used for key: value pairs and when the value likely needs to different in other environments. Other tools such as the Configuration migration utility are better suited for migration of relational configuration data stored within custom tables. Unlike other configuration data, environment variables are migrated within solutions and therefore much simpler to manage and more performant to import.

How do I remove a value from an environment variable?

You might want to remove the value of an environment variable from your solution before exporting the solution. Then, the existing value remains in your development environment, but isn't exported in the solution. This approach allows a new value to be provided while importing the solution into another environment.

To remove the value, follow these steps:

1. In the solution where the environment variable is located select the environment variable to display the properties.
2. Under **Current Value**, select **...** > **Remove from this solution**.



Can I use environment variables in custom connectors?

Yes. Environment variable support in custom connectors

See also

Environment variables overview

Feedback

Was this page helpful? Yes No

Provide product feedback

Use settings to provide customized app experiences

Article • 03/13/2023

Settings are solution components that enable makers and administrators to quickly configure apps to provide a customized experience. Settings can be used to enable or disable features or configure feature behavior for a single app or all apps within an environment.

Settings are made up of three subcomponents: Setting definition, setting environment value, and setting app value.

Subcomponent	Description

Setting definition	• Application authors, administrators, or partners can create a setting definition that includes properties such as name, description, data type, default value, and more.
	• They can then, via code, use the setting's value to enable or disable or configure the feature they are delivering.
	• They can include the setting definition in the solution that is used to deliver the feature to their customers.
	• Example:
	  ◦ The *Async save handler* for model-driven apps is a feature from Power Apps that uses settings.
	  ◦ The setting is of type *Yes/No* and the default value of this setting is *No*.
	  ◦ Therefore, by default, the *Async save handler* feature will be disabled for all model-driven apps.

Setting environment value	• A setting environment value can be used to override the default value, as specified in the setting definition.
	• Example:
	  ◦ Extending the example above, an environment administrator in a customer's organization can add a setting environment value for the *Async save handler* setting and set it to *Yes*.
	  ◦ This setting environment value will override the default value and will apply to all apps in that environment.
	  ◦ This will enable the *Async save handler* feature for all model-driven apps in that environment.

Subcomponent	Description

Setting app value	• A setting app value can be used to override the default value, as specified in the setting definition, and the setting environment value (if one exists).
	• Example:
	  ◦ Once again, extending the example above, an application author in the same environment above, can add a setting app value for the *Async save handler* setting for an app they are authoring and set it to *No*
	  ◦ This setting app value will override the setting environment value and apply only to that single app.
	  ◦ This will enable the *Async save handler* feature for that single app.

> ⓘ Note
>
> To follow the steps listed below you need to have **Solution preview on**. From the **Solutions** area in Power Apps, on the command bar, ensure you have **Solution preview on**. If **Solution preview off** is displayed, select the option to enable the preview. More information: **Solution view**

Setting definition

A setting definition specifies the base properties of a setting. The full list of properties that can be configured on a setting definition are listed below. Some of these properties can't be changed after the setting is created.

Property	Description
Display name	The name displayed to consumers of the setting in all user interfaces where settings are displayed.
Name	The unique name of the setting in an environment. Name is automatically generated based on the display name provided but can be changed before the setting is created. Once a setting is created, the **Name** can't be changed as it may be referenced in applications or code. **Name** has a prefix that corresponds to the solution publisher. This prefix is intended to give the setting a unique name if you want to import them into another solution or environment in the future (which would have a different prefix).
Description	The description helps others understand what the setting is used for in all user interfaces where settings are displayed.

Property	Description
Data type	The data type of a setting controls how the setting's value is stored. Data type can be set to **Number**, **String**, or **Yes/No**. Data type can't be changed after the setting is created.
Default value	The default value becomes available after you select a **Data type**. It specifies the setting's value that will be used unless it is overridden by a setting environment value or a setting app value.
Value can be changed for	A setting can be overridden at one of the following levels : • **Environment and app**, allows both the setting environment value and setting app values to override the default value. • **Environment only**, allows only the setting environment value to override the default value. • **App only**, allows only setting app values to override the default value. • **None**, the setting can't be overridden.
Release level	Release level is used to inform the framework and other consumers of the setting about the state of the feature that the setting is used with. Release level can be set to **Generally available** or **Preview**.
Information Url	A link to documentation to help consumers of the setting understand the purpose of the setting. Will be used as a *Learn more* link in all user interfaces where settings are displayed.

Adding a new setting definition

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you want to add the setting to.
4. In the command bar, select **New** > **More** > **Setting** > **Setting definition**.
5. In the **New setting** dialog, provide values for each of the properties based on your requirements. For information about setting definition properties, go to: Setting definition
6. When you are done providing values for the properties, select **Save**.

Adding an existing setting definition

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you want to add the setting to.

4. In the command bar, select **Add existing** > **More** > **Setting**.
5. In the **Add existing setting definition** dialog, select one or more settings that you want to add.
6. When you are done selecting settings, select **Next**
7. In the **Selected setting definition** dialog, for each setting you have selected, you will have the option to **Include setting definition**. You can also **Include setting environment value** for each setting, if one exists.
8. Select **Add** to add the setting definition and/or setting environment values.

Updating a setting definition

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you created the setting in.
4. In the tree view, select **Settings** > **Setting definitions**.
5. Select the setting definition that you want to update.
6. In the **Edit setting** dialog, update values for any of the properties you want to change.
   Note that some properties can't be updated after a setting has been created. Additionally, in most cases you will not be able to update settings definitions that you don't own.
7. When you are done updating the values for the properties, select **Save**.

Removing a setting definition

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you created or added the setting to.
4. In the tree view, select **Settings** > **Setting definitions**.
5. Select the setting definition you want to remove.
6. In the command bar, select **Remove** > **Remove from this solution**.
   Remove from this solution removes the setting definition from the current solution, but it will continue to be a part of the default solution or any other solution it was added to.

Deleting a setting definition

1. Sign in to Power Apps.

2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you created or added the setting to.
4. In the tree view, select **Settings** > **Setting definitions**.
5. Select the setting definition you want to delete.
6. In the command bar, select **Remove** > **Delete from this environment**.
   Delete from this environment deletes the setting definition from the current environment and it will no longer be a part of any solutions in that environment. You can only delete setting definitions that you own.

Setting environment value

A setting environment value is used to override the setting's default value for all apps in an environment. Use a setting environment value when the setting's default value is not what you want to be used for apps in your environment.

Adding a new setting environment value

1. Sign in to Power Apps.

2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

3. From the list of solutions, open the solution you want to add the setting environment value to.

4. If the setting definition *is* available in the solution:
   a. Select the setting definition.
   b. In the **Edit setting value** dialog, in the **Setting environment value** section, select **New value**.
   c. Provide the value, and then select **Save**.

   If the setting definition *is not* available in the solution:
   a. On the command bar, select **New** > **More** > **Setting** > **Setting environment value**.
   b. Select the setting you want to add the setting environment value for, and then select **Add**.
   c. In the **New setting environment value** dialog, in the **Setting environment value** section, select **New value**.
   d. Provide the value, and then select **Save**.

Adding an existing setting environment value

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you want to add the setting environment value to.
4. If the setting definition *is* available in the solution:
   a. Select the setting definition.
   b. In the **Edit setting value** dialog, in the **Setting environment value** section, select **Add existing value**.
   c. Update the value if needed, and then select **Save**.

   If the setting definition *is not* available in the solution:
   a. On the command bar, select **Add existing** > **More** > **Setting**.
   b. In the **Add existing setting definition** dialog, select one or more settings that you want to add setting environment values for.
   c. When you are done selecting settings, select **Next**
   d. In the **Selected setting definition** dialog, for each setting you have selected, you will have the option to **Include setting environment value**, if one exists. You can also **Include setting definition** for each setting if you want.
   e. Select **Add** to add the setting environment values and/or setting definitions.

Updating a setting environment value

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you added the settings environment value to.
4. In the tree view, select **Settings** > **Setting environment values**.
5. Select the setting environment value that you want to update.
6. In the **Edit setting** dialog, in the **Setting environment value** section, update the value, and then select **Save**.

Removing a setting environment value

1. Sign in to Power Apps.

2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you created or added the setting to.
4. In the tree view, select **Settings** > **Setting environment values**.
5. Select the setting environment value you want to remove.
6. On the command bar, select **Remove** > **Remove from this solution**.
   Remove from this solution removes the setting environment value from the current solution but it will continue to be a part of the default solution or any other solution it was added to.

Deleting a setting environment value

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you created or added the setting to.
4. In the tree view, select **Settings** > **Setting environment values**.
5. Select the setting environment value you want to delete.
6. On the command bar, select **Remove** > **Delete from this environment**.
   Delete from this environment deletes the setting environment value from the current environment and it will no longer be a part of any solutions in that environment.

Setting app value

Setting app value is used to override the setting's default value and any setting environment value for a single app. Use a setting app value when the setting environment value is not what you want to be used for your app. If a setting environment value does not exist, use a setting app value when the setting's default value is not what you want to be used for your app.

> **Note**
>
> To follow the steps listed below you need to use the new **app designer**. Currently the app designer only displays Power Apps platform settings. For non-platform settings please use the solution explorer as outlined below.

Adding or updating a setting app value using the app designer

1. Open the app you want to add the setting app value for in the app designer.
2. On the command bar, select **Settings**.
3. In the **Settings** dialog, select the **Features** or **Upcoming** tab.
   The **Features** tab displays all settings that have release level set to **Generally available**. The **Upcoming** tab displays all settings that have release level set to **Preview**.
4. Add or update an app value for the setting you want.
5. Save and publish the app.

Deleting a setting app value using the app designer

1. Open the app you want to remove the setting app value for in the app designer.
2. On the command bar, select **Settings**.
3. In the **Settings** dialog, select the **Features** or **Upcoming** tab.
4. Select **Reset to environment value** next to the setting app value you want to delete. This makes the setting value fall back to the setting environment value if one exists. If no setting environment value exists, it will fall back to the setting's default value.
   Note that the option to **Reset to environment value** is only displayed if a setting app value was previously added for that setting.
5. Save and publish the app.

Adding or updating a setting app value using the solution explorer

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you want to add the setting app value to.
4. If the setting definition or setting environment value is not available in the solution, add one of them. Go to **Adding an existing setting definition** or **Adding an existing setting environment value**.
5. Select the setting definition or setting environment value.
6. In the **Edit setting value** dialog, in the **Setting app values** section, find the app that you want to add the setting app value for. Note that using the solution explorer you can only add setting app values for apps that are in the current solution.
7. Provide the value and select **Save**.
8. For the setting app value to take effect you will have to republish the app using the app designer or solution explorer.

Deleting a setting app value using the solution explorer

1. Sign in to Power Apps.
2. In the navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. From the list of solutions, open the solution you want to delete the setting app value from.
4. If the setting definition or setting environment value is not available in the solution, add one of them. More information: Adding an existing setting definition or Adding an existing setting environment value.
5. Select the setting definition or setting environment value.
6. In the **Edit setting value** dialog, in the **Setting app values** section, find the app that you want to delete the setting app value for. Note that using the solution explorer you can only remove setting app values for apps that are in the current solution.
7. Select **...** next to the setting app value, and then select **Reset to environment**.
8. Select **Save**.
9. For the deletion of the setting app value to take effect you will have to republish the app using the app designer or solution explorer.

Getting or updating a setting value via code

For the methods available, go to Get or update a setting value using client API.

See also

For developers:

- RetrieveSetting Function
- SaveSettingValue Action

Use a connection reference in a solution with Microsoft Dataverse

A *connector* is a proxy or a wrapper around an API that allows the underlying service to talk to Microsoft Power Automate, Microsoft Power Apps, and Azure Logic Apps. It provides a way for users to connect their accounts and use a set of prebuilt actions and triggers to build their apps and workflows.

A connection is a stored authentication credential for a connector, for example OAuth credentials for the SharePoint connector.

A *connection reference* is a solution component that contains a reference to a connection about a specific connector. Both solution-aware canvas apps and operations within a solution-aware flow bind to a connection reference instead of directly to a connection. During solution import into a target environment, a connection is provided for all the connection references so any referencing flows can be turned on automatically after the import completes. To change a specific connection associated with a canvas app or flow, you edit the connection reference component within the solution.

Add connection references to a solution

Connection references can be added to a solution in different ways:

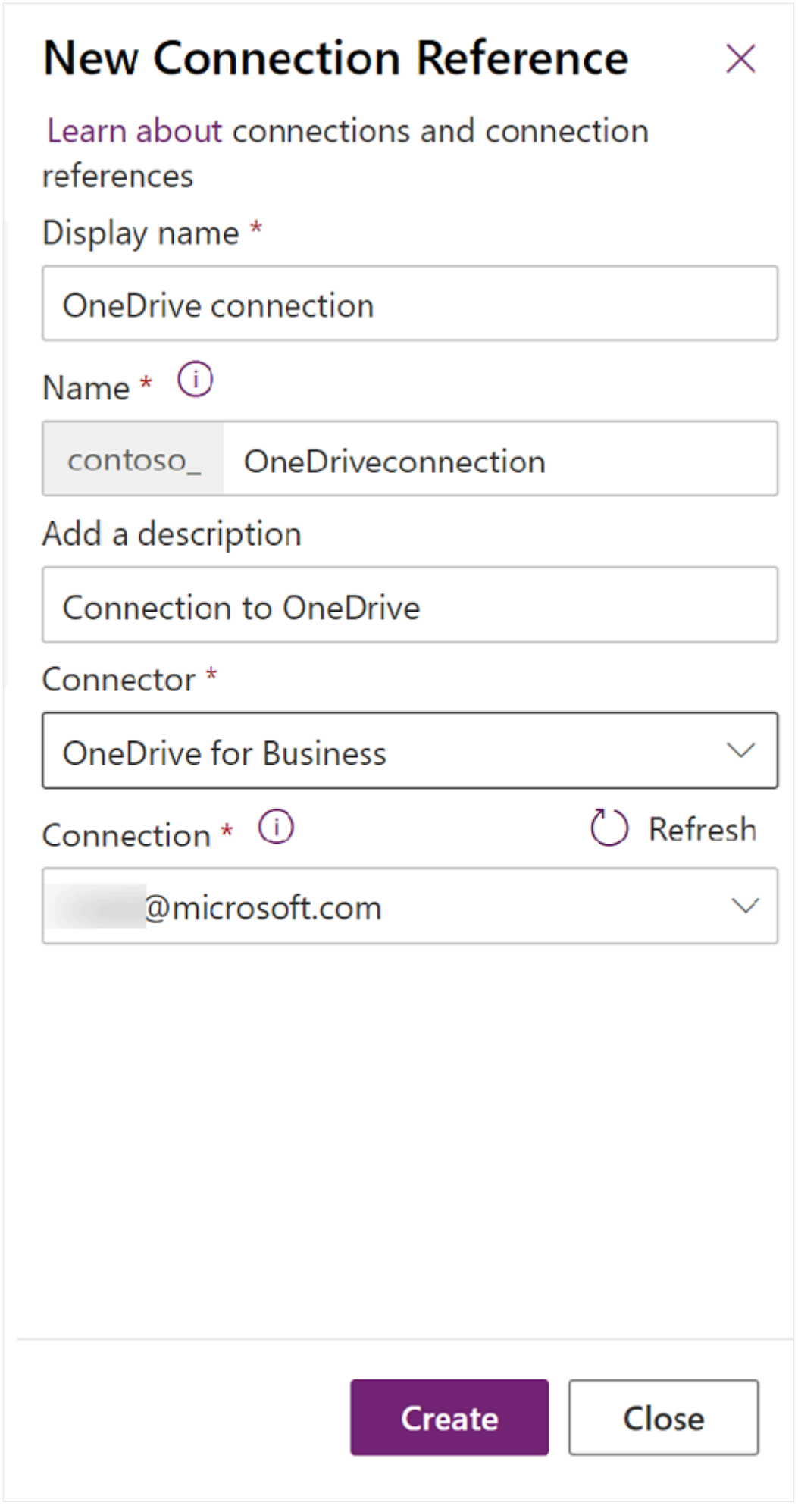
- When you're using the solution explorer to create a new connection reference in a solution.
- When you import a solution. To learn more, go to Import solutions.
- Implicitly when you build your canvas apps and flows that are defined in a Microsoft Dataverse solution.

> **Note**
>
> - Canvas apps and flows handle connections differently. Flows use connection references for all connectors, whereas canvas apps only use them for implicitly shared (non-OAuth) connections, such as SQL Server Authentication. More information: Security and types of authentication
> - A connection reference is automatically created when you create new connections from the flow designer or Power Apps Studio.

- Canvas apps and flows added from outside solutions will not automatically be upgraded to use connection references.
- Connection references get associated with canvas apps only at the time a data source is added to the app. To upgrade apps you must remove the connection from the app and then add a connection containing an associated connection reference.

Manually add a connection reference using solution explorer

1. Sign in to Power Apps or Power Automate.
2. On the left pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Create a new or open an existing solution.
4. On the command bar, select **New** > **More** > **Connection Reference**.
5. On the **New Connection Reference** pane, enter the following information. Required columns are denoted with an asterisk (*).
    - **Display name**: Enter a unique and useful name to help differentiate this connection reference from others.
    - **Add a description**: Enter text that describes the connection.
    - **Connector**: Select an existing connector from the list such as in the screenshot here. You can also select **New** to create a new connection for this connection reference. Once your finished creating a new connection, select **Refresh** to select your connection from the list.
    - **Connection**: Based on the **Connector** you selected, select an existing connection or select **New connection** to create one.
6. Select **Create**.


New Connection Reference
Learn about connections and connection references
Display name *
OneDrive connection
Name *
contoso_
OneDriveconnection
Add a description
Connection to OneDrive
Connector *
OneDrive for Business
Connection *
Refresh
@microsoft.com
Create
Close

Connection reference naming

The display name of a connection reference should be unique so different connection references can be differentiated by name alone. By default, a connection reference name includes the target connector, the current solution name for context, and a random suffix to ensure uniqueness. Consider adjusting the connection reference name to something unique and something that explains what it will be used for.

Reuse connections in a solution flow

Flows created outside a solution use connections directly. Flows created in a solution use connection references and the connection reference points at the connection. To reuse a connection within a solution flow, you first need to create a connection reference pointing at that connection.

Update a flow to use connection references instead of connections

When a flow isn't in a solution, it uses connections. If that flow is then added into solution, it will continue to use connections initially. Flows can be updated to use connections references instead of connections in one of two ways:

1. If the flow is exported in an unmanaged solution and imported, the connections will be removed and replaced with connection references.

2. When a solution flow is opened, the flow checker on the flow details page will show a warning to **Use connection references**. The warning message contains an action to **Remove connections so connection references can be added**. Selecting that action will remove connections from the trigger and actions in the flow and allow connection references to be selected and created.

Automatic use of connection references in a solution flow

When an action is added to a solution flow, Power Automate will try to reuse existing connection references from the current solution or other solutions before creating a new connection reference. To ensure that the connection reference is inside the same solution as the flow, create or add a connection reference in the same solution and reference that connection reference from the flow.

Share connections with another user so flows can be enabled

When a flow is turned on (enabled), the user turning on the flow needs to own or have permission to use all the connections in the flow. This is usually accomplished by having the flow owner create the connections inside all the connection references that the flow uses. If a user other than the flow owner provides the connections on a flow, then the flow needs to be turned on by the owner of those connections or the connections need to be shared with the user who is turning on the flow.

> ⓘ **Note**
>
> OAuth connections can only be explicitly shared with a user representing a service principal.

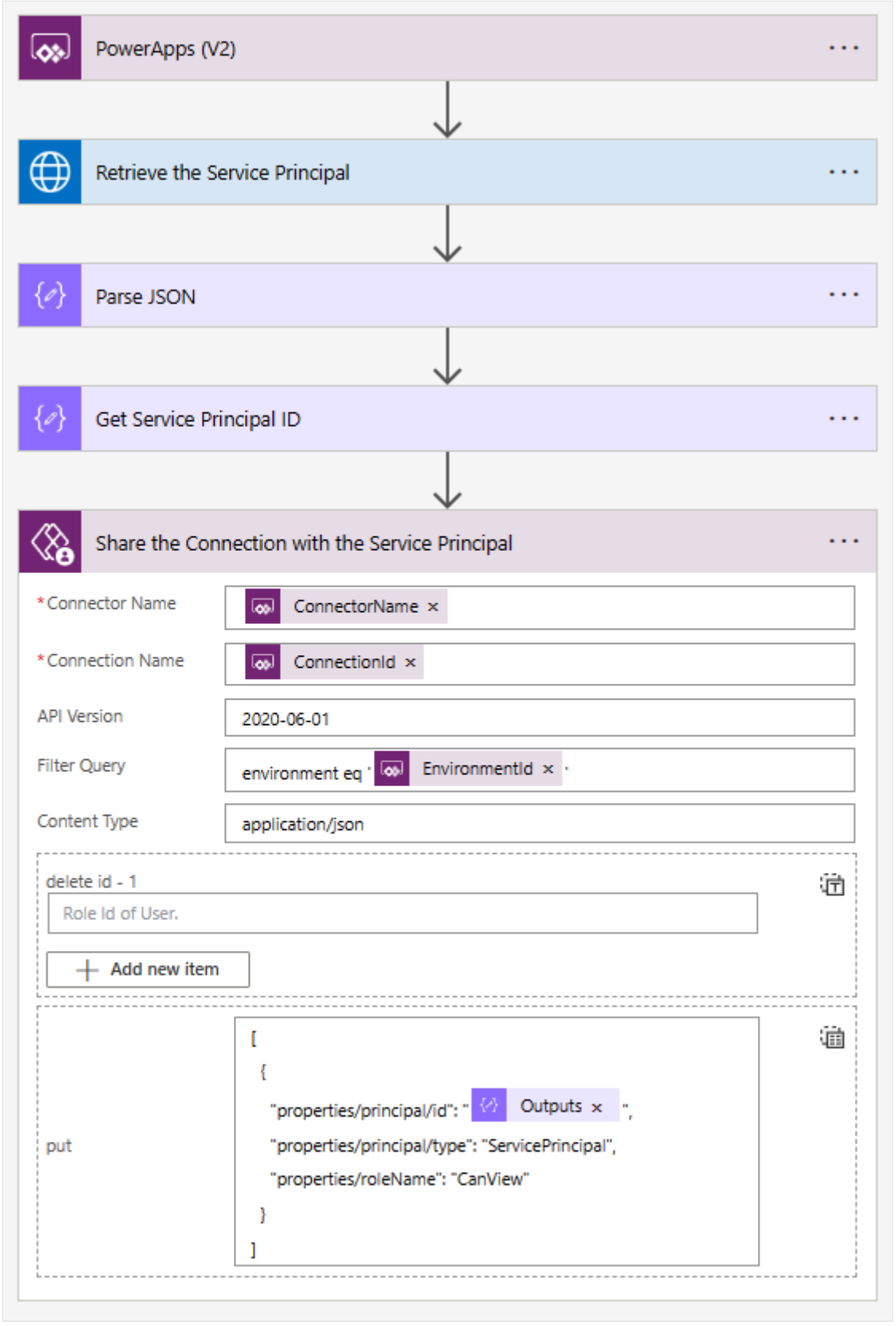
Manual sharing of connections for flow enablement

Sharing connections can be accomplished with the following steps.

1. Go to Power Apps and select the environment containing the connection.
2. Select **Connections** on the left navigation pane, and then select the connection you want to share. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. From the menu, select **Share**.
4. From the sharing screen, enter the name of the user (service principal) who will enable the flow.
5. For the permissions, select **Can use**.
6. To complete the sharing, select **Save**.

Automated sharing of connections for flow enablement

To automate sharing of connections, use the Edit Connection Role Assignment action on the Power Apps for Makers connector.



Limits

- Connection references are now saved asynchronously. Unlike during the preview period, there's no longer a limit to how many flows can reference the same connection reference. When connection references are updated, an info banner that links to a panel containing asynchronous update details appears.

- There's also no limit to the number of actions in each flow that can be associated with the connection reference.
- Canvas apps don't recognize connection references on custom connectors. To work around this limitation, after a solution is imported to an environment the app must be edited to remove and then readd the custom connector connection. Note, if this app is in a managed solution, proceeding to edit the app will create an unmanaged layer. More information: Solution layers

Known issues

This section describes known issues with connection references.

Copy environment breaks connection references for custom connectors

Custom connectors use an environment-specific identifier to reference a custom connector. After a copy environment operation, a new connection reference must be created to the new custom connector. Then, any apps or flows using the old connection references will need to be fixed.

Custom connectors need to be imported in a separate solution from their connection references

Custom connectors need to be imported in a separate solution, before connection references or flows. Export a solution containing only the custom connector first.

FAQ

How do I fix an "Invalid Connection" error?

If the connection reference is showing as "invalid" where a red exclamation point appears on the flow details page, this means the underlying connection is in a bad state. When this occurs, check and fix the underlying connection by updating the connection, or replace the connection.

What is the ConnectionAuthorizationFailed error? Why can't I turn on (activate) a flow?

The "ConnectionAuthorizationFailed" error indicates that the user trying to activate the flow doesn't have permissions to at least one of the connections the flow is using. To resolve the situation, select from one of the following resolutions:

- The users owning the connections needs to share *all* the connections with the user turning on (activating) the flow.
- If one user owns all the connections in the flow, then that user can turn on (activate) the flow.

Once the flow has been turned on by the owner of the connections, then the flow has permissions to use those connections. From that point on, any co-owner of the flow can turn on the flow.

Can a flow be enabled by the owner of its connections and then ownership transferred to another user?

Yes. When a flow is turned on (enabled) by the owner of the connections used by the flow, then the flow gets explicit permission to use those connections. Co-owners of the flow can then turn the flow off and on as needed.

Permissions granted to apps and flows using a connection can be seen in the details page for that connection in the **Apps using this connection** and **Flows using this connection** tabs.

If the flow is edited to add new actions that use additional connection references with new connections, then the owner of those new connections either needs to initially turn on the flow themselves or share the connections with the owner who turns on the flow. More information: Share app resources

Can ownership of a connection reference be transferred to another user?

Ownership of a connection reference can't be transferred to another user from the Solutions area in Power Apps (make.powerapps.com).

However, the classic solution explorer can be used to change privileges of a connection reference including sharing.

See also

Connectors

Last updated on 01/09/2026

Translate label text

Article • 02/15/2022

The standard tables include default text for labels that are available in many different languages. However, when you customize a form, such as adding or changing columns, or create custom tables, you may need the labels for those components to appear in different languages. You can import translated label text for customized tables for your apps so that the label text displays in languages other than the base language.

When you translate label text for a form, you are modifying the base language of the form as part of a form customization. When you do this, you create active unmanaged changes to the labels. Subsequently, if you don't modify the base language translation for a component, such as a column, and then export the translation file, the column's object id won't be exported. This is because the export doesn't see any modifications done to the base language label of that column.

High-level process

1. Export the translations from the solution that contains the tables that you want to translate label text. Then, open the translations XML file and add the translated text. More information: Translate customized table and column text into other languages
2. Import the translations. More information: Import translated table and column text back into an app

Common issues with translating form label text

Form label translations don't appear in the layers

Imported translations may not appear when you view the solution layers for a component. Make sure the translations are on the <Label> level when you view the solution layer. Note that "displayname" tags are at the attribute level, so they will not translate labels. The "DisplayName" tags are actually for form labels and will translate form's labels. For more information about viewing solution layers, see Solution layers.

Form label translations don't appear specifically for the base language

The translation export and import feature is a tool for used so your apps can display translated label text for customized components. It isn't designed to be used in managing all labels including the base language.

On translation export, if no changes were made to the base language label, no translations will be exported in CrmTranslations.xml.

On translation import, if translations for any label in the base language column were changed in CrmTranslations.xml, the changes won't take effect.

If the issue is base language specific, do the following:

1. Change the translations for the base language using the form designer for each label that was changed.
2. Use translation export and import to add translations for other provisioned languages.
3. Export the active unmanaged solution as managed.
4. Import this solution into the target environment.

Form Label translation descriptions appear when exporting as unmanaged, but are "" when exporting as managed

This can happen because the label has no translations. The following behavior occurs:

- The unmanaged solution is filled with the display name: <label description="example" languagecode="1033">
- The managed solution is empty, which is seen as non-existent and falls back to the display name: <label description="" languagecode="1033">

See also

Translate customized table and column text into other languages
Import translated table and column text back into an app

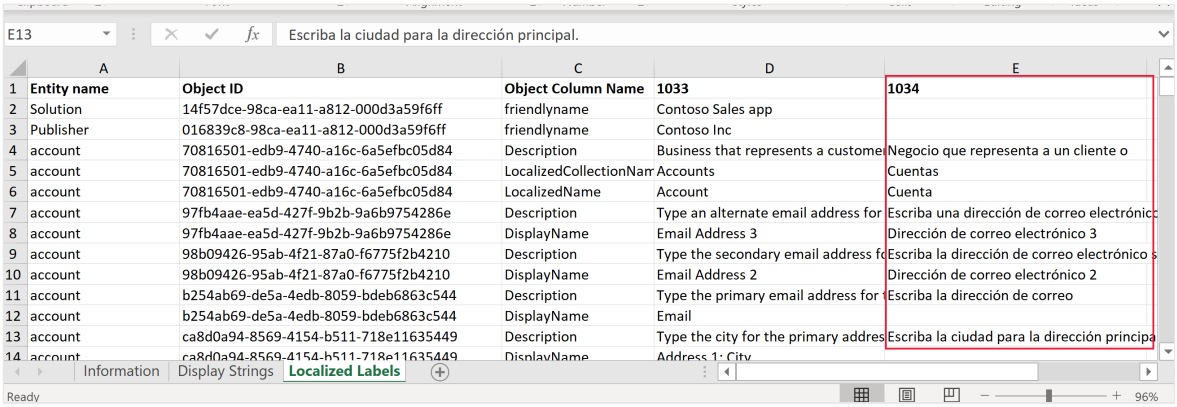
Translate customized table, form, and column text into other languages

Article • 03/13/2023

After you create customized table and column text in your unmanaged solution, you may want to translate it into other languages.

> **Important**
>
> When you export translations, the export translations feature exports translations for the table. So, that means even if the solution contains only a single form, labels for all the forms for the table will be exported. Make sure you only modify the form's labels when importing the translations back otherwise another component translation you modify will be added as a dependency to the solution.

1. Sign into Power Apps and select **Solutions** from the left navigation. If the item isn't in the left navigation pane, select ...More and then select the item you want.
2. Select the unmanaged solution you want, on the command bar select **...**, select **Translations**, and then select **Export translations**.
3. After the export completes the exported translations compressed (zip) file is downloaded to your browser's default download folder and contains the exported labels.
4. Extract the XML file from the compressed (.zip) file.
5. Open the CrmTranslations.xml file in Excel.
6. Select the sheet named **Localized Labels**.
7. Notice there is already a column with the base language code id, such as 1033 (English U.S.) Add a column with the language code id for every language you want to translate labels. For example, add a column for 1034 (Spanish traditional).
8. Add the translated text in the new column for the object names and object ids that you want.



	A	B	C	D	E
1	**Entity name**	**Object ID**	**Object Column Name**	**1033**	**1034**
2	Solution	14f57dce-98ca-ea11-a812-000d3a59f6ff	friendlyname	Contoso Sales app	
3	Publisher	016839c8-98ca-ea11-a812-000d3a59f6ff	friendlyname	Contoso Inc	
4	account	70816501-edb9-4740-a16c-6a5efbc05d84	Description	Business that represents a custome	Negocio que representa a un cliente o
5	account	70816501-edb9-4740-a16c-6a5efbc05d84	LocalizedCollectionNam	Accounts	Cuentas
6	account	70816501-edb9-4740-a16c-6a5efbc05d84	LocalizedName	Account	Cuenta
7	account	97fb4aae-ea5d-427f-9b2b-9a6b9754286e	Description	Type an alternate email address for	Escriba una dirección de correo electrónic
8	account	97fb4aae-ea5d-427f-9b2b-9a6b9754286e	DisplayName	Email Address 3	Dirección de correo electrónico 3
9	account	98b09426-95ab-4f21-87a0-f6775f2b4210	Description	Type the secondary email address fo	Escriba la dirección de correo electrónico s
10	account	98b09426-95ab-4f21-87a0-f6775f2b4210	DisplayName	Email Address 2	Dirección de correo electrónico 2
11	account	b254ab69-de5a-4edb-8059-bdeb6863c544	Description	Type the primary email address for t	Escriba la dirección de correo
12	account	b254ab69-de5a-4edb-8059-bdeb6863c544	DisplayName	Email	
13	account	ca8d0a94-8569-4154-b511-718e11635449	Description	Type the city for the primary addres	Escriba la ciudad para la dirección principa
14	account	ca8d0a94-8569-4154-b511-718e11635449	DisplayName	Address 1: City	

9. When you're finished adding your translations, save, and zip up the package so you can Import translated table and column text.

> **Note**
>
> Not all content can be translated using this tool. This includes labels for sitemap areas, groups, and subareas. To translate these labels, **use the More Titles property in the sitemap designer**.

Community tools

Easy Translator is a tool that XrmToolBox community developed for Power Apps. Use Easy Translator to export and import translations with contextual information.

> **Note**
>
> The community tools are not supported by Microsoft. If you have questions about the tool, please contact the publisher. More Information: **XrmToolBox**.

Next steps

Import translated table and column text

Import translated table, form, and column text back into an app

Article • 03/13/2023

If you have customized table or column text, such as column labels or drop-down list values, you can provide the users in your organization who are not working with the base language version of your environment with this customized text in their own languages. To do so, you export the text strings for all your customizations so that they can be translated into the languages you use in your organization.

After the translation, you need to import the translated text strings into your environment before users can take advantage of the changes.

> **Important**
>
> - The file that you import must be a compressed file that contains the CrmTranslations.xml and the [Content_Types].xml file at the root.
> - You can't import translated text that is over 500 characters long. If any of the items in your translation file are longer than 500 characters, the import process will fail. If the import process fails, review the line in the file that caused the failure, reduce the number of characters, and try to import again. Also note that after you import translated text, you must republish customizations.

1. Sign in to Power Apps at https://make.powerapps.com.
2. Select **Solutions**, and select the unmanaged solution from which to import the translated text. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. In the solution explorer, on the Actions toolbar, select **Translations**, and then select **Import translations**.
4. In the **Import Translated Text** dialog box, specify the file that contains the translated text, and then select **Import**.
5. When the import is complete, select **Close**.

> **Note**

Publishing customizations can interfere with normal system operation. We recommend you schedule publishing when it's least disruptive to users.

Community tools

Easy Translator is a tool that XrmToolBox community developed for Power Apps. Use Easy Translator to export and import translations with contextual information.

> **Note**
>
> The community tools are not supported by Microsoft. If you have questions about the tool, please contact the publisher. More Information: **XrmToolBox**.

Next steps

Export customized table and column text for translation

Update localized labels for forms

Article • 12/07/2024

This article describes how to update existing localized labels that were created following the steps in the article Translate localizable text for model-driven apps.

Updating localized labels overview

When you create a new model-driven app form and add columns to the form, the form creates a copy of the localized labels for those columns in the base language. Once a form is created, changing the localized label text for a column in the base language doesn't update the localized label on the form. However, you can change the column, tab, and section labels that can be localized for a form using the form designer. For more information about customized text, go to Configure column properties on a form.

To update localized labels for forms through the export translations Excel file, you need to correctly identify the object ID corresponding to each tab, section, or column. The following sections explain how to do this.

Identify the object ID for form labels

1. Create a solution and add the form as part of this new solution.
2. Export the solution as managed.
3. Unzip the solution file and open the customizations.xml file with an XML or text editor.
4. Find the tab, section, or column of the form that has the localized label you want to update.
5. Once you find the tab, section, or column, check if there's `labelid` property defined. If yes, then the value of `labelid` attribute is the object ID for the localized label. If not, then the value of the ID attribute is the object ID for localized label.

For example, say you want to find the object ID for attribute `websiteurl`. Search for that attribute in the customizations.xml file in the `formxml` section.

XML

```
<row>
<cell id="{e6441984-4343-813a-aa7e-e2747ad35390}" showlabel="true" labelid="
{aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb}">
<labels>
<label description="Website" languagecode="1033" />
```

```
</labels>
<control id="websiteurl" classid="{71716B6C-711E-476c-8AB8-5D11542BFB47}"
datafieldname="websiteurl" disabled="false" />
</cell>
</row>
```

In the example, since the `labelid` property is defined, the object ID is *aaaaaaaa-0000-1111-2222-bbbbbbbbbbbb*.

Now that you have the object ID, you can follow the steps to export the translations file. Open the file, to locate the object ID to make your localized text updates for the tab, section, or column. For more information about how to translate the customized text into another language, go to Translate customized table, form, and column text into other languages.

What if you can't find the asset in customizations XML?

When you export a model-driven form, only the diff between the active layer and last managed layer is exported. If you don't make any changes to the tab, section, or column, the asset doesn't appear in the customizations.xml.

You can modify the label or any other property of the asset using the form designer, and then export the changes to find the object ID.

Alternate way to find the object ID of a localized label

There are two alternative ways to find the object ID for a localized label:

- Find the full form XML of the form by viewing the **Properties** page for the solution layers of the model-driven app form. For more information about solution layers, go to View the solution layers for a component.
- You can also make an OData call to retrieve the form xml if you know the form ID.
  - `[environmentURL]/api/data/v9.0/systemforms(b0742891-2411-494b-bfe7-93bc20356399)?$select=formxml`

See also

Overview of the model-driven form designer

Feedback

Was this page helpful? Yes No

Provide product feedback

Tools available to diagnose solutions

Article • 02/15/2022

There are two tools available for diagnosing solutions in Power Apps.

- Object checker
- Solution checker

Object checker

Object checker and solution checker are similar and share much of the same infrastructure. However, object checker runs *live diagnostics on the database* and can analyze issues based on the metadata and layering of a given solution object - regardless of which solution(s) the object is in. Therefore, object checker is better suited for troubleshooting application lifecycle (ALM) related issues in test and production environments. More information: Use object checker to diagnose a solution component

Solution checker

Solution checker runs offline analysis on a single solution file. Solution checker requires the solution be exported before you can run a solution diagnostic. Solution checker is best suited for development environments or analyzing a solution prior to committing it to a source control repository. More information: Use solution checker to validate your model-driven apps in Power Apps

See also

Debug a model-driven app with Monitor

Use object checker to diagnose a solution component (preview)

Article • 11/13/2024

[This topic is pre-release documentation and is subject to change.]

Object checker runs real-time diagnostics on component objects within your solution. If issues are detected, a recommendation is returned that describes how to fix the issue. This might include documentation, instructions, or a click-able action that can fix the issue for you.

Example issues object checker detects

- Active layer hiding customizations.
- Malformed or corrupted metadata.

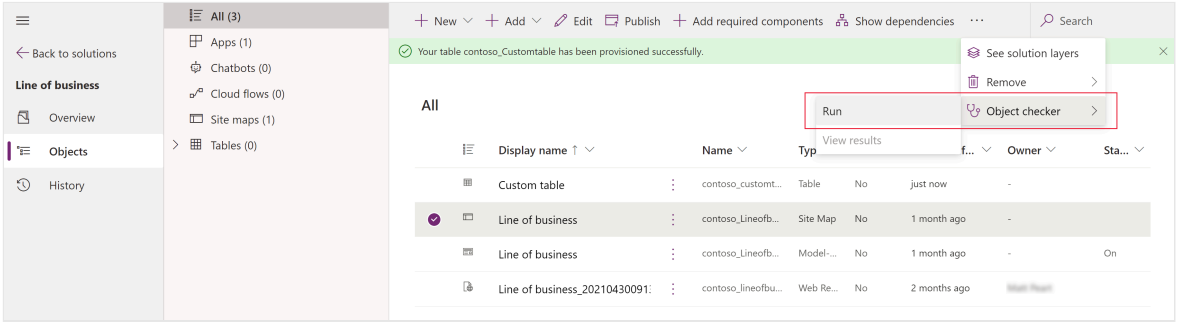
Run object checker and view the results

1. Sign into Power Apps, and then select **Solutions** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.

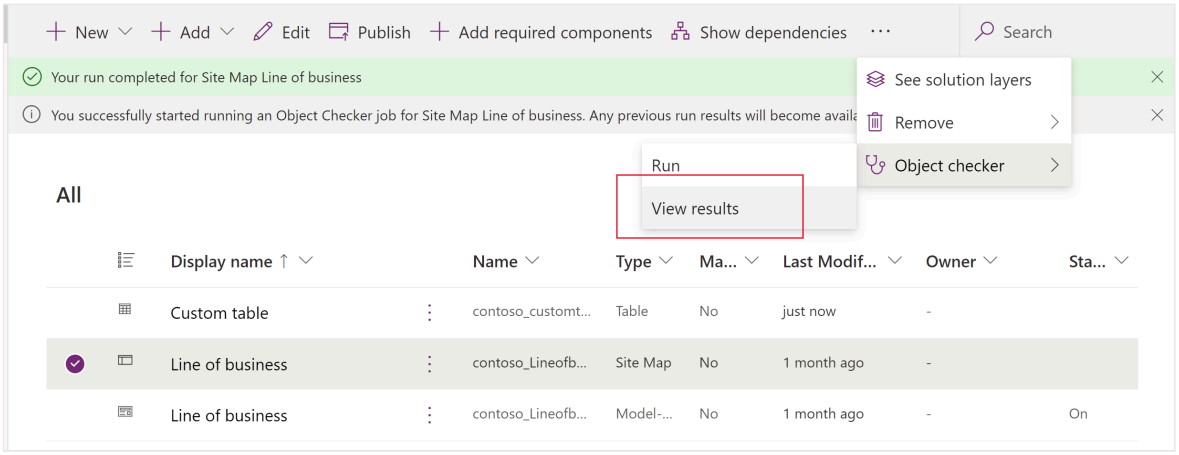
2. Open a solution that contains the problematic object. You can run object checker within either managed and unmanaged solutions.

3. Select a single object, such as a model-driven app or a site map.

4. On the command bar, select **...** > **Object checker** > **Run**.



5. When object checker finishes, on the command bar, select **...** > **Object checker** > **View results**



6. The results are displayed in the right pane. If issues are detected, follow the prompt to resolve the issue.

7. Select **Close** to close the object checker results pane.

Current limitations

- Object checker currently only works with model-driven app and site map components.
- Object Checker rules might fail if customizations haven't been published. When this condition occurs, the following exception can be observed in the '**Multiple sitemaps detection rule**':

```text
Exception encountered while executing rule:
System.ServiceModel.FaultException`1[Microsoft.Xrm.Sdk.OrganizationServ
iceFault]: The value passed for ConditionOperator.In is empty.
```

See also

Tools available to diagnose solutions
Use object checker to diagnose a solution component (video)

Feedback

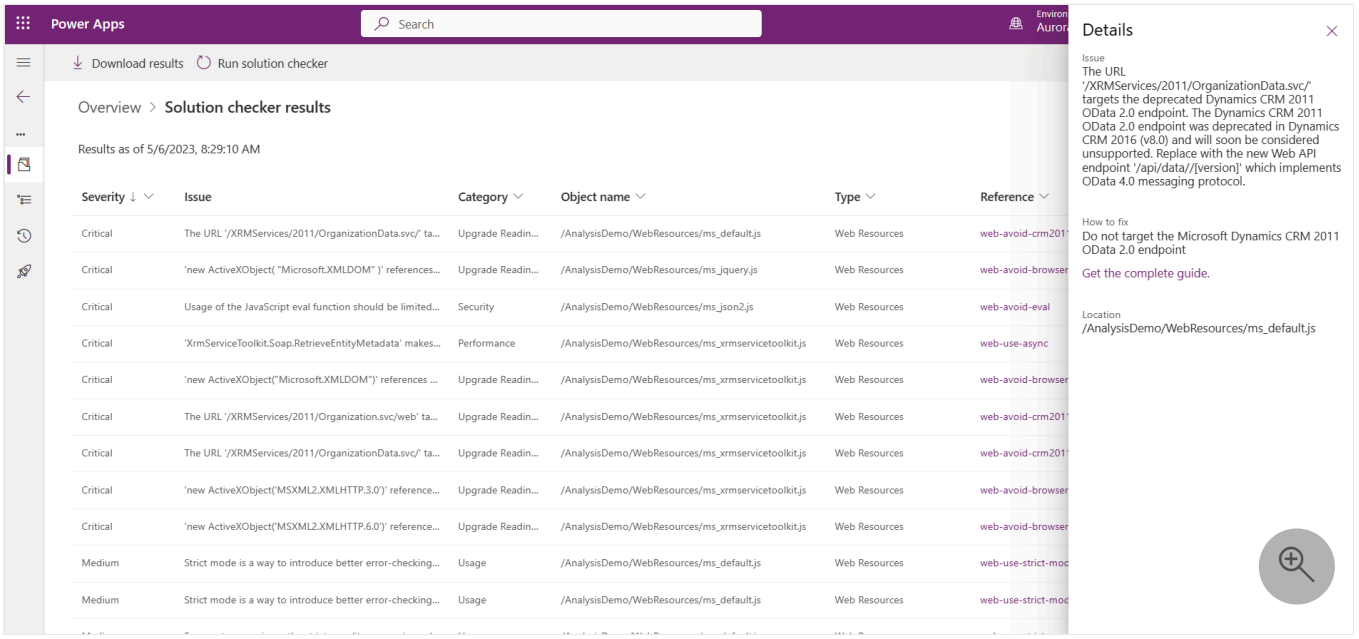
Was this page helpful? Yes No

Provide product feedback

Improve solution performance, stability and reliability

06/19/2025

Solutions are used to distribute Power Platform objects, such as apps, tables, flows, web resources, and plugins. This article introduces the solution checker feature, a powerful tool that performs a comprehensive static analysis of your solution objects against a set of best practice rules. By using solution checker, you can quickly identify problematic patterns in your components and receive detailed reports that highlight issues, affected components, and provide links to documentation on how to resolve each issue. This ensures your solutions are optimized for performance, stability, and reliability.



Solution checker works with unmanaged solutions that can be exported from an environment.

You can run solution checker either from Power Apps (make.powerapps.com) or by using PowerShell.

How solution checker helps you

To deliver on complex business requirements, makers often can end up with highly advanced solutions that customize and extend the Power Platform. With advanced implementations come an increased risk where performance, stability, and reliability issues become introduced, which can negatively impact the user experience. Identifying and understanding how to resolve these issues can be complicated and time consuming. With the solution checker feature, you can perform a check within seconds on your solution, which uses a set of best practice rules to quickly identify problematic patterns. After the check completes, you receive a detailed report in Power Apps as well as in an email message that lists the issues identified, the components and code affected, and links to documentation that describes how to resolve each issue.

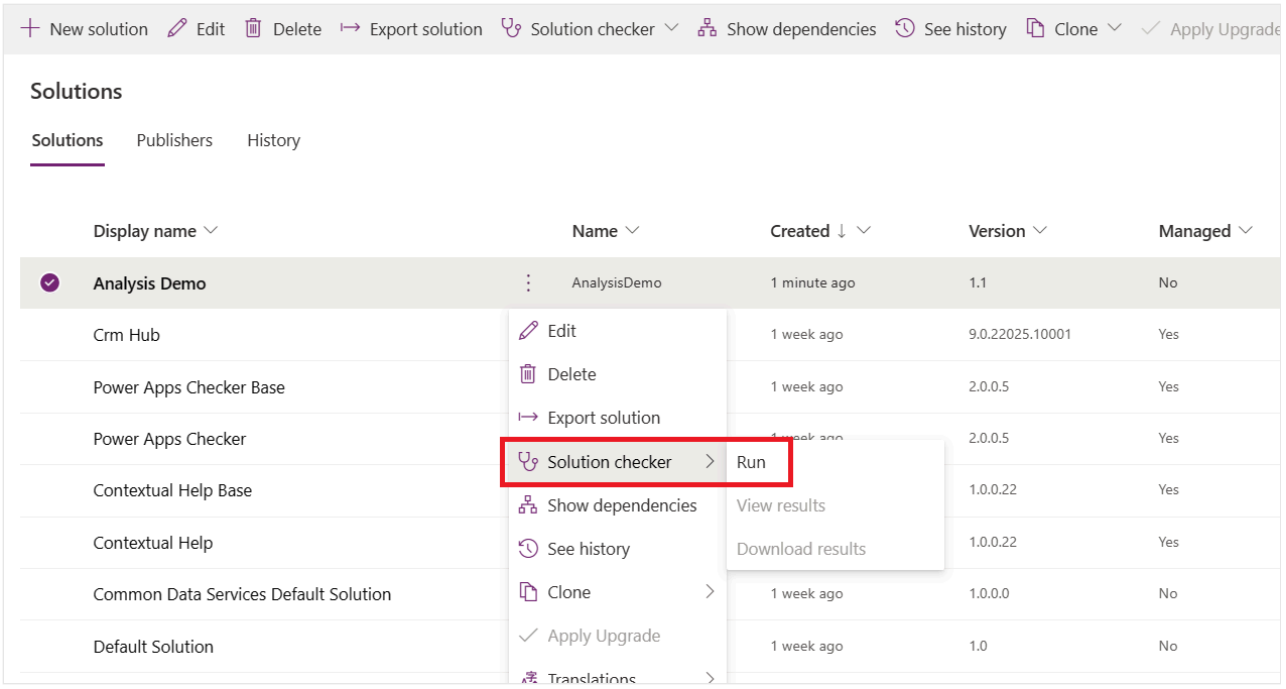
The solution checker analyzes these solution components:

- Dataverse custom workflow activities
- Dataverse web resources (HTML and JavaScript)
- Dataverse configurations, such as SDK message steps
- Power Automate flows (via flow checker)
- Power Fx expressions (via app checker)

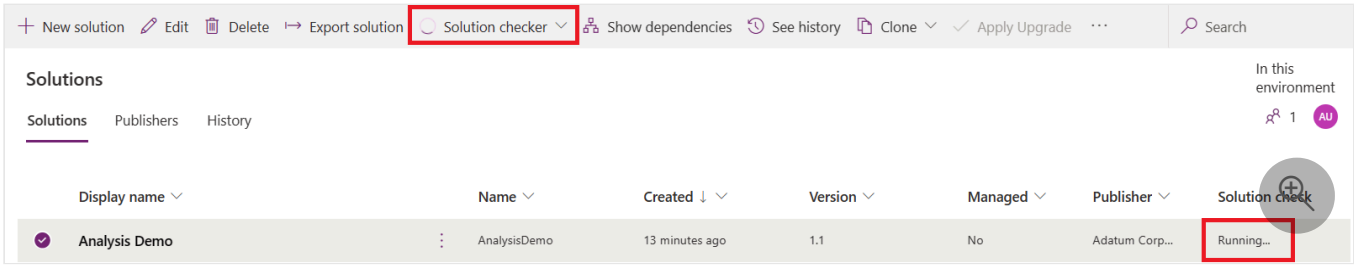
> **Note**
>
> - Solution checker supports global variables for ECMAScript 2015 (ES6) and up to ECMAScript 2018 (ES9) syntax. When JavaScript is detected using global variables later than ES6 or syntax later than ES9, a web-unsupported-syntax issue for the web resource is reported.
> - Use of solution checker doesn't guarantee that a solution import will be successful. The static analysis checks performed against the solution don't know the configured state of the destination environment and import success might be dependent on other solutions or configurations in the environment.

Run the solution checker

1. Sign in to Power Apps.
2. In the left pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Next to the unmanaged solution that you want to analyze, select **...**, point to **Solution checker**, and then select **Run**.



4. The **Solution checker** command button has a loading indicator, and you'll notice a **Running...** state in the **Solution check** column of the **Solution** list.



> **Note**
>
> - The solution checker can take a few minutes to complete the analysis.
> - You'll receive an email notification and a notification in the **Notifications** area of the Power Apps site when the check is completed.

5. View the report when the check is completed.

Cancel a check

After you submit a solutions check in your environment, the check can be canceled through the status pane on the upper right area of the **Solutions** page.

When you cancel a check, the solution check stops running and the solution check status returns to the previous state.

Solution checker states

When you install the solution checker in your environment, the **Solution check** column becomes available in the **Solutions** list. This column displays the solution analysis states for a solution.

Expand table

State	Description
Hasn't been run	The solution has never been analyzed.
Running	The solution is being analyzed.
Couldn't be completed	Solution analysis was requested but the analysis didn't complete successfully.
Results as of *date and time*	Solution analysis completed and results are available for download.
Couldn't be completed. Result as of *date and time*	The latest analysis request didn't complete successfully. The last successful results can be downloaded.
Checked by Microsoft	This is a Microsoft-managed solution. Solution analysis isn't permitted on these solutions.
Checked by Publisher	This is a non-Microsoft managed solution. Currently, solution analysis isn't available for these solutions.

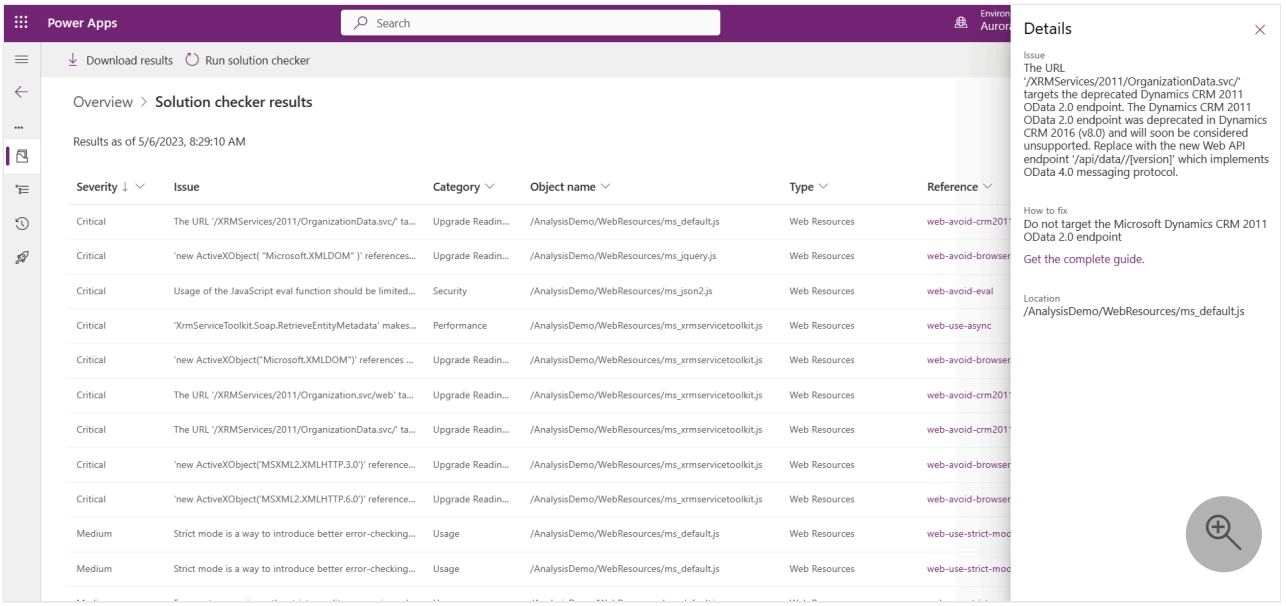
Review the solution checker report

When a solution check is completed, you can view the analysis report in the portal, or you can download the report from your web browser. In the portal, you have options to sort results by **Issue**, **Location** or by **Severity** and view detailed information for issues detected in your solution.

1. In the left pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.

2. Next to the unmanaged solution where you want to view the solution checker report, select **...**, point to **Solution checker**, and then select **View results**.

3. Select an issue to view the details and guidance on how to resolve.



The solution check results are also available for download. The solution checker zip file is downloaded to the folder specified by your web browser. The download report is in Excel format and contains several visualizations and columns that assist you in identifying the impact, type, and location of each issue detected in your solution. A link to detailed guidance about how to resolve the issue is also provided.

1. In the left pane, select **Solutions**. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Next to the unmanaged solution where you want to download the solution checker report, select **...**, point to **Solution checker**, and then select **Download results**.
3. The solution checker zip file is downloaded to the folder specified by your web browser.

Here's a summary of each column in the report.

Expand table

Report column	Description	Applies-to component
Issue	The title of the issue identified in the solution.	All
Category	The categorization of the issue identified, such as **Performance**, **Maintainability**, **Usage**, **Supportability**, **Design**, **Security**, **Accessibility**, or **Upgrade readiness**.	All
Severity	Represents the potential impact of the issue identified. Available impact types are **Critical**, **High**, **Medium**, **Low**, and **Informational**.	All
Guidance	Link to article detailing the issue, impact, and recommended action.	All
Component	The solution component where the issue was identified.	All
Location	The location and/or source file of the component where the issue that was identified occurred, such as the assembly or JavaScript file name.	All

Report column	Description	Applies-to component
Line #	The line number reference of the issue in the impacted web resource component.	Web resources
Module	Module name where the issue identified in the assembly was detected.	Custom workflow activity
Type	Type of the issue identified in the assembly.	Custom workflow activity
Member	Member of the issue identified in the assembly.	Custom workflow activity
Statement	The code statement or configuration that resulted in the issue.	All
Comments	Details about the issue that include high-level resolution steps.	All

Run solution checker rules locally

You can run solution checker rules in your development environment to detect issues much sooner as you create your solution resources. This is currently supported for web resources (JavaScript and TypeScript). For more details, go to the NPM package @microsoft/eslint-plugin-power-apps .

Run solution checker using PowerShell

A PowerShell module is available that you can use to interact directly with the service. The Microsoft.PowerApps.Checker.PowerShell module can be used for analysis of unmanaged solutions for Power Apps environments, or to automate and integrate the service into your build and release pipelines. More information: Microsoft.PowerApps.Checker.PowerShell Overview

Best practice rules used by solution checker

The following table lists the component type, rule description, severity, and category. Critical violations are blocked or warned when configured for solution checker enforcement with managed environments. More information: Use solution checker in Managed Environments

Expand table

Solution component	Rule name	Rule description	Severity	Category
Plug-in or workflow activity	meta-remove-dup-reg	Avoid duplicate Dataverse plug-in registrations.	Critical	Performance

Solution component	Rule name	Rule description	Severity	Category
Plug-in or workflow activity	meta-avoid-reg-no-attribute	Include filtering attributes with Dataverse plug-in registrations.	Medium	Performance
Plug-in or workflow activity	meta-avoid-reg-retrieve	Use caution with Dataverse plug-ins registered for Retrieve and RetrieveMultiple messages.	Medium	Performance
Plug-in or workflow activity	meta-remove-inactive	Remove inactive configurations in Dataverse.	Low	Maintainability
Plug-in or workflow activity	meta-avoid-crm4-event	Don't use Microsoft Dynamics CRM 4.0 plug-in registration stage.	Medium	Upgrade readiness
Plug-in or workflow activity	meta-avoid-retrievemultiple-annotation	Avoid registering a plugin on RetrieveMultiple of annotation.	High	Usage
Model-driven app	meta-license-sales-entity-operations	Solution contains entities with restricted SDK messages and operations that require a valid Dynamics 365 license.	Low	Licensing
Model-driven app	meta-license-fieldservice-customcontrols	Solution contains custom controls that require a valid Dynamics 365 Field Service license.	Low	Licensing
Model-driven app	meta-license-fieldservice-entity-operations	Solution contains entities with restricted SDK messages and operations that require a valid Dynamics 365 Field Service license.	Low	Licensing
Web Resources	use-async	Interact with HTTP and HTTPS resources asynchronously.	Critical	Performance
Web Resources	avoid-modals	Avoid using modal dialogs.	High	Supportability
Web Resources	avoid-dom-form		High	Supportability
Web Resources	avoid-dom-form-event		High	Supportability
Web Resources	avoid-crm2011-service-odata	Don't target the Microsoft Dynamics CRM 2011 OData 2.0 endpoint.	Critical	Upgrade readiness
Web Resources	avoid-crm2011-service-soap	Don't target the Microsoft Dynamics CRM 2011 SOAP services.	Critical	Upgrade readiness

Solution component	Rule name	Rule description	Severity	Category
Web Resources	avoid-loadtheme	Don't use `loadTheme` Fluent v8 API.	Low	Supportability
Web Resources	avoid-browser-specific-api	Don't use Internet Explorer legacy APIs or browser plug-ins.	Critical	Upgrade readiness
Web Resources	avoid-unpub-api		High	Supportability
Web Resources	avoid-window-top		High	Supportability
Web Resources	avoid-2011-api	Don't use the deprecated Microsoft Dynamics CRM 2011 object model. Instead follow Dataverse Web API documentation.	High	Upgrade readiness
Web Resources	use-relative-uri	Don't use absolute Dataverse endpoint URLs.	Medium	Maintainability
Web Resources	use-cached-webresource		Medium	Performance
Web Resources	use-client-context	Use client contexts.	Medium	Upgrade readiness
Web Resources	use-navigation-api	Use navigation API parameters.	Medium	Upgrade readiness
Web Resources	use-offline		Medium	Upgrade readiness
Web Resources	do-not-make-parent-assumption		High	Design
Web Resources	use-org-setting	Use organization settings.	Medium	Upgrade readiness
Web Resources	use-global-context		Medium	Upgrade readiness
Web Resources	use-grid-api	Use the grid APIs.	Medium	Upgrade readiness
Web Resources	use-utility-dialogs		Medium	Usage
Web Resources	avoid-isActivityType	Replace Xrm.Utility.isActivityType method with new Xrm.Utility.gettableMetadata and don't use in ribbon rules.	Medium	Upgrade readiness

Solution component	Rule name	Rule description	Severity	Category
Web Resources	meta-avoid-silverlight	Silverlight web resource usage is deprecated.	Medium	Upgrade readiness
Web Resources	remove-debug-script	Avoid including debug script in nondevelopment environments.	Medium	Usage
Web Resources	use-strict-mode	Use strict mode when possible.	Medium	Usage
Web Resources	use-strict-equality-operators	Use strict equality operators.	Medium	Usage
Web Resources	avoid-eval	Don't use the `eval` function or its functional equivalents.	Critical	Security
Web Resources	avoid-with	Don't use the 'with' operator.	High	Performance
Web Resources	remove-alert	Don't use the 'alert' function or its functional equivalents.	Medium	Usage
Web Resources	remove-console	Avoid using methods on console.	Medium	Usage
Web Resources	avoid-ui-refreshribbon	Avoid using refreshRibbon in form onload and EnableRule.	Critical	Performance
Web Resources	use-getsecurityroleprivilegesinfo	Avoid userSettings.securityRolePrivileges. Use userSettings.getSecurityRolePrivilegesInfo instead.	High	Performance
Web Resources	use-appsidepane-api	Use Xrm.App.sidePanes.createPane instead of Xrm.Panels.loadPanel.	Medium	Upgrade readiness
Web Resources	web-sdl-no-cookies	HTTP cookies are an old client-side storage mechanism with inherent risks and limitations. Use Web Storage, IndexedDB, or other modern methods instead.	Medium	Security
Web Resources	web-sdl-no-document-domain	Writes to document.domain property must be reviewed to avoid bypass of same-origin checks. Usage of top level domains such as azurewebsites.net is strictly prohibited.	Medium	Security
Web Resources	web-sdl-no-document-write	Calls to document.write or document.writeln manipulate DOM directly without any sanitization and should be avoided. Use	Medium	Security

Solution component	Rule name	Rule description	Severity	Category
		document.createElement() or similar methods instead.		
Web Resources	web-sdl-no-html-method	Direct calls to method html() often (for example, in jQuery framework) manipulate DOM without any sanitization and should be avoided. Use document.createElement() or similar methods instead.	Medium	Security
Web Resources	web-sdl-no-inner-html	Assignments to innerHTML or outerHTML properties manipulate DOM directly without any sanitization and should be avoided. Use document.createElement() or similar methods instead.	Medium	Security
Web Resources	web-sdl-no-insecure-url	Insecure protocols such as HTTP or FTP should be replaced by their encrypted counterparts (HTTPS, FTPS) to avoid sending potentially sensitive data over untrusted networks in plaintext.	Medium	Security
Web Resources	web-sdl-no-msapp-exec-unsafe	Calls to MSApp.execUnsafeLocalFunction() bypass script injection validation and should be avoided.	Medium	Security
Web Resources	web-sdl-no-postmessage-star-origin	Always provide specific target origin, not * when sending data to other windows using postMessage to avoid data leakage outside of trust boundary.	Medium	Security
Web Resources	web-sdl-no-winjs-html-unsafe	Calls to WinJS.Utilities.setInnerHTMLUnsafe() and similar methods don't perform any input validation and should be avoided. Use WinJS.Utilities.setInnerHTML() instead.	Medium	Security
Canvas App	app-formula-issues-high	Go to Power Apps formula reference for additional details.	Critical	Design
Canvas App	app-formula-issues-medium	Refer to Power Apps formula references for additional details.	Medium	Design
Canvas App	app-formula-issues-low	Refer to Power Apps formula references for additional details.	Low	Design
Canvas App	app-use-delayoutput-text-input	Use delayed load in some scenarios to improve performance.	Medium	Performance

Solution component	Rule name	Rule description	Severity	Category
Canvas App	app-reduce-screen-controls	Limit the number of app controls for improved performance.	Medium	Performance
Canvas App	app-include-accessible-label	Use explicit labels to improve app accessibility.	Medium	Accessibility
Canvas App	app-include-alternative-input	Ensure all interactive elements are accessible to alternative inputs.	Medium	Accessibility
Canvas App	app-avoid-autostart	Avoid using autostart on players within an app.	Medium	Accessibility
Desktop flow	desktopflow-avoid-unsafe-password	Passwords are managed insecurely in the flow.	High	Security
Desktop flow	desktopflow-avoid-subflow-recursion	Recursive calls detected between subflows, potentially causing an infinite loop.	Medium	Design
Desktop flow	desktopflow-avoid-infinite-loop	Infinite loop detected in the flow, potentially causing it to run indefinitely.	Medium	Design
Desktop flow	desktopflow-avoid-incomplete-if-branch	Incomplete If action detected, lacking content or only containing actions in the Else branch.	Low	Design
Desktop flow	desktopflow-avoid-excessive-nested-ifs	Nested If clauses exceed five levels.	Low	Maintainability
Desktop flow	desktopflow-avoid-empty-on-error-block	"On block error" action is empty and not handling errors.	Low	Design
Desktop flow	desktopflow-limit-argument-count	Total input/output variables exceed the 25-variable limit.	Low	Maintainability
Desktop flow	desktopflow-input-argument-default-value	Input/output variables aren't using default values.	Low	Maintainability
Desktop flow	desktopflow-limit-variable-name-length	Variable name exceeds the 25-character limit.	Low	Maintainability
Desktop flow	desktopflow-avoid-excessive-wait-actions	Misuse of wait actions detected, with more than 10 wait actions causing potential bottlenecks.	Low	Performance
Desktop flow	desktopflow-avoid-immense-wait-duration	Immense wait time detected, exceeding the 600-second limit for hardcoded wait actions.	Low	Performance

See also

Best practices and guidance for the Dataverse

Best practices and guidance for model-driven apps

Common issues and resolutions for Solution Checker

ⓘ **Note:** The author created this article with assistance from AI. Learn more

Common issues and resolutions for solution checker

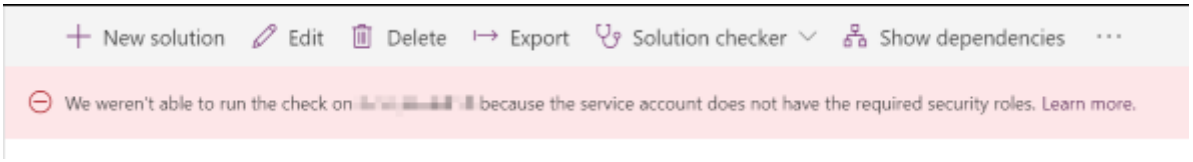
This article lists some common issues that you might encounter while using solution checker. Where applicable, workarounds are provided.

You're unable to use solution checker to run analysis or download results

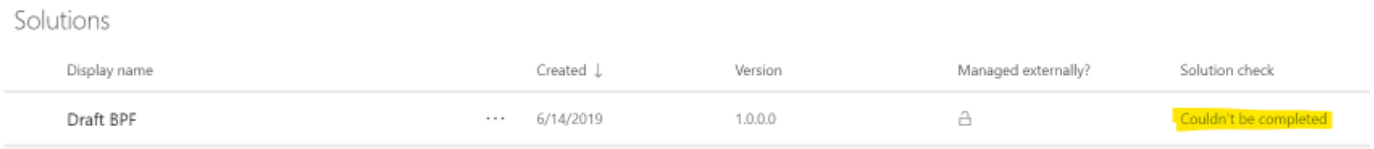
Shortly after submitting a solution checker request to run an analysis or download results the operation doesn't complete and an error message is displayed, such as:

> "*We weren't able to run the check on **[Solution Name]** solution. Try running it again.*"

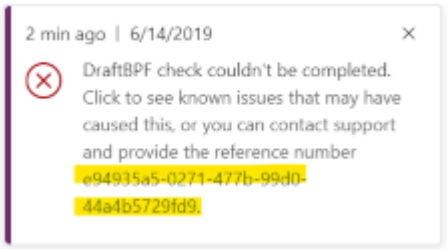
Whenever possible, solution checker attempts to return a specific error message with a link to details about the potential cause and resolution steps. Select **'Learn more'** for details.



Failures that occur during background processing of the analysis will fail with **'Couldn't be completed'** status and return an error message in the Power Apps portal as well as send email notification to the requestor.



Selecting the portal notification will link to this page of common issues for further troubleshooting. If one of the provided common issues doesn't resolve the problem, a reference number is also returned. Provide this reference number to Microsoft Support for further investigation.



Solution checker fails to check Default solution

Solution checker exports the solution from the Microsoft Power Platform environment for analysis. Exporting the Default solution isn't supported and the check may not complete successfully. More information: Solution Checker fails to export large solutions.

Solution checker fails due to unsupported version of Power Apps Checker

Solution checker is a feature enabled by the Power Apps Checker app. If you have installed a Power Apps Checker app version earlier than version **1.0.0.47**, solution checker runs may fail to complete successfully. You should upgrade your Power Apps Checker version from the Power Platform admin center.

However, if you have a Power Apps Checker version earlier than version **1.0.0.45** installed, we recommend that you delete the solution and install it again. Due to recent schema changes, upgrade of Power Apps Checker from versions earlier than version **1.0.0.45** may fail.

If you want to keep the past results from solution checker, export the results from a previous run or export all solution checker data using Export data to Excel to export the data from the following tables:

- Analysis Component
- Analysis Job
- Analysis Result
- Analysis Result Detail

How to uninstall Power Apps Checker

To uninstall the Power Apps Checker solution:

1. As a System Administrator or as a System Customizer, open the Power Apps portal by going to https://make.powerapps.com/environments.
2. Select **Solutions**.

3. Select **Power Apps Checker**, and then on the solutions toolbar select **Delete**.

How to install Power Apps Checker

To install Power Apps Checker back into your environment:

1. As a System Administrator or as a System Customizer, open up your Power Apps portal by going to https://make.powerapps.com/environments.
2. Select **Solutions**.
3. On the solution toolbar select **Solution checker**, and then select **Install**.

Solution checker can't access environments in Administration Mode

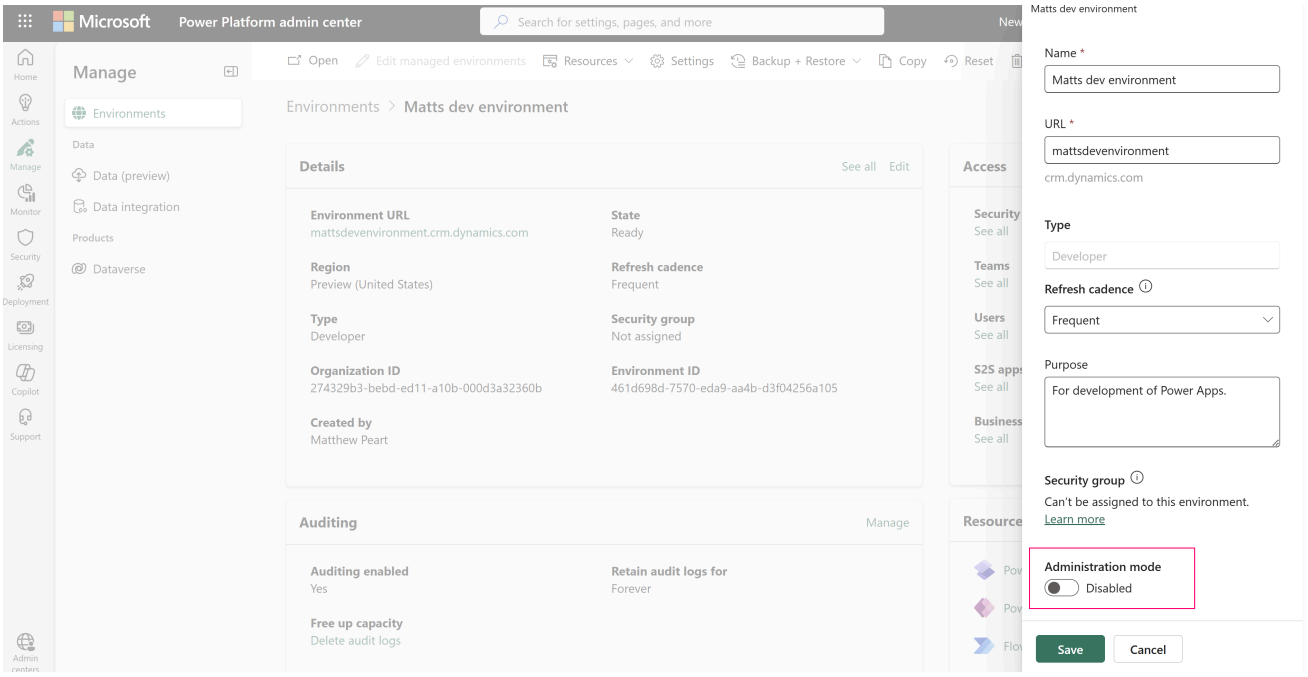
Environments that have been placed into Administration Mode purposely restrict access to only users with System Administrator and System Customizer roles. Because the Power Apps Checker application identity has neither of these roles assigned by default, it can't access organizations operating in this mode.

In order to use solution checker in this organization, Administration Mode must be disabled.

How to disable administration mode

To disable administration mode for an organization instance:

1. Sign in to the Power Platform admin center.
2. Open the environment that has issues running solution checker, and then select **Edit**.
3. Set **Administration mode** to **Disabled**, and then select **Save**.



4. Run solution checker again.

Solution checker fails due to missing security roles

The application user for solution checker requires the following security roles assigned in order to provide the necessary privileges to communicate with the Dataverse organization. If any of these roles aren't assigned to the user '**Power Apps Checker Application**', attempts to run analysis, download results, and run cancellation will fail. This occurs most often when customers have automation in place that removes security roles from unexpected users. The following security roles contain minimum required permissions:

- Export Customizations (Solution Checker)
- Service Writer
- Solution Checker

How to assign missing security roles

To assign missing security roles to the Power Apps Checker Application user:

1. In the Power Platform admin center, open your environment, then select **Users** > **app users list**.
2. Select the **Power Apps Checker Application** user from the list of users.
3. Select **Edit security roles** on the command bar.
4. Select checkboxes for the roles listed above, and then select **Save**.
5. Run solution checker again.

Solution checker fails due to restricted access mode

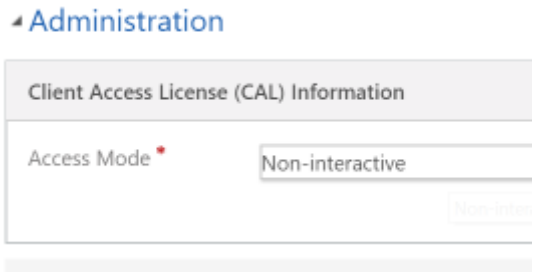
The application user for solution checker requires an access mode of **'Non-Interactive'** or **'Read-Write'** in order to communicate with the Dataverse organization. If the access mode has been changed to another value such as **'Administrative'**, then attempts to run analysis, download results, and run cancellation will fail.

To resolve this issue, you must update the **Power Apps Checker Application** user with 'Non-interactive' access mode.

How to update user access mode

To update the access mode for the Power Apps Checker user:

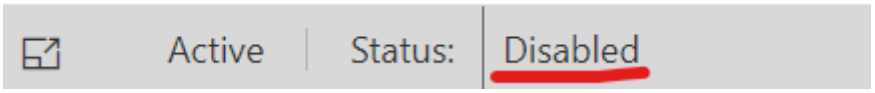
1. In Power Apps (make.powerapps.com) select you environment and then select **Settings** > **Advanced settings** > **Administration** > **Users** to open the legacy user settings.
2. Open the **Power Apps Checker Application** user from the list of users.
3. Scroll to the **Administration** > **Client Access License (CAL) Information** section of the form and select **'Non-interactive'** in the **Access Mode** drop-down control.



4. Save and close the user form.
5. Run solution checker again.

Solution Checker fails due to disabled application user

The Power Apps Checker application user in the Dataverse organization containing solutions to be analyzed must be enabled. If the application user becomes disabled, requests to analyze solutions in the same organization will fail. If receiving this error message, first verify that the Power Apps Checker application user is indeed disabled. Then follow the mitigation steps provided below.



How to activate the Power Apps Checker application user

1. In the Power Platform Admin center, open the environment and go to **Settings** > **User's + Permissions** > **Users**.
2. In the **Application Users** view, select the **Power Apps Checker Application** user.
3. On the command bar, select **Activate**
4. In the **Confirm User Activation** message, select **Activate**.

Solution checker remains in a Running state for more than 30 minutes

Solution checker may remain in a running state indefinitely if the analyzed solution contains plugin components developed using .NET Standard 2.0. Plugins using .NET Standard 2.0 aren't currently supported in solution checker and should be excluded from analysis. Exclude the plugin component by either removing it from the target solution, or by using the 'ExcludedFileNamePattern' parameter in the Power Apps Checker PowerShell module.

Common plugin conditions that cause solution checker to fail

When solution checker receives and processes analysis requests, it must call the Dataverse endpoint to retrieve/update relevant job data and export the selected solution(s). Each interaction made by the solution checker service with the Dataverse could potentially trigger one or more plugin steps that have been registered on message submitted in the request. These plugins may in turn introduce conditions that prevent the message from being handled as expected by the Dataverse and interrupt the ability of solution checker to process the requested analysis job. Similar situations can occur when downloading solution checker job results or canceling an in-progress analysis job.

Typical Dataverse operations requested by solution checker:

- Retrieve solution, `systemuser`, and organization table data
- Create, update, and retrieve analysis job, analysis component, and analysis result table data
- Export solutions

Plugin step registered to execute in context of an unlicensed user

When solution checker fails due to an "unlicensed user" exception, it's often caused by a triggered plugin step configured to execute in the context of a specific system user that is currently unlicensed. Ensure that any plugin steps that could be triggered by solution checker execute in context of a licensed user.

> **Important**
>
> It is highly recommended that plugin steps be configured to execute in context of the calling user rather than specific users which are subject to assigned license being revoked.

Plugin step performs operations that require privileges not granted to Power Apps Checker application user

When solution checker fails due to Dataverse denying access based on a missing privilege, it's often caused by a triggered plugin step that performs operations that require privileges not currently granted to the Power Apps Checker application user. Either reconfigure the plugin step to not execute on the operation invoked by solution checker or grant the Power Apps Checker application user the necessary privileges to execute the custom plugin step.

Plugin step unexpectedly interrupts execution by throwing InvalidPluginExecutionException

Solution checker may fail due to the errors "ISV code aborted the operation" or "An unexpected error occurred from ISV code". When this occurs, it indicates a plugin step was triggered that explicitly interrupted execution by throwing an InvalidPluginExcecutionException or that an error wasn't caught properly in plugin code. Either reconfigure the plugin step to not execute on the operation invoked by solution checker or adjust the plugin implementation not to interrupt execution based on the conditions presented by solution checker.

Common operations invoked by solution checker include Create and Update. These operations can occur on the following tables: Analysis Job, Analysis Component, or Analysis Result.

Solution checker fails due to disabled first-party application in Microsoft Entra ID

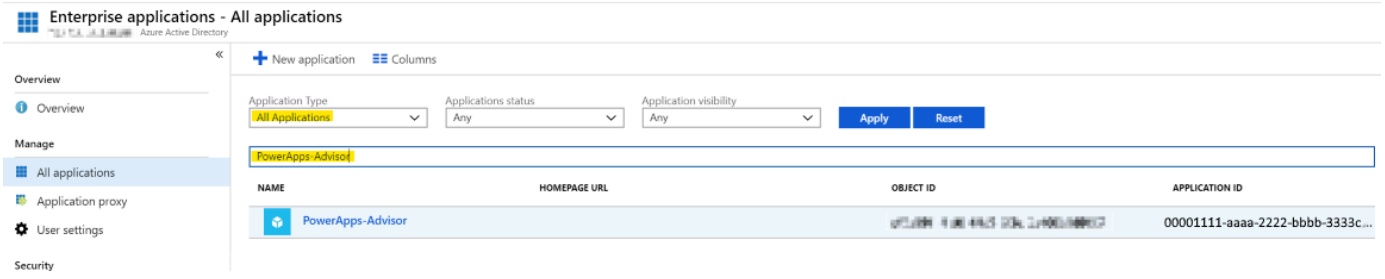
The first-party enterprise application identity used by solution checker (PowerApps-Advisor) shouldn't be disabled in Microsoft Entra ID. If disabled, the identity can't authenticate when requesting bearer tokens for Dataverse and other required resource providers on-behalf of the requesting user.

Follow the below steps to verify that the application identity hasn't been disabled in Microsoft Entra and if necessary enable the application.

How to verify and/or modify application enabled status

To verify and/or modify the enabled status of the PowerApps-Advisor enterprise application identity

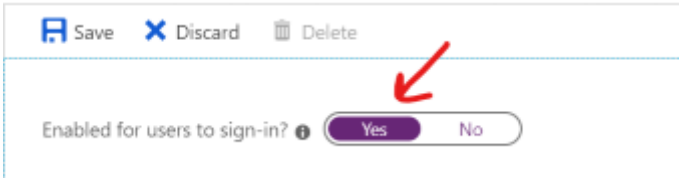
1. Access your tenant in the Microsoft Entra Portal.
2. Navigate to **Enterprise Applications**.
3. Select **All Application** and search for **'PowerApps-Advisor'**.



4. Select **'PowerApps-Advisor'** to view the app details.
5. Select **Properties**.
6. Check the state of **Enabled for users to sign in**. If **'No'**, then the application has been disabled.



7. Select the radio control to switch the value to **'Yes'**. This enables the application.



8. Select **Save**. The application is now enabled. You may need to wait a few minutes for change to propagate.
9. Run solution checker again.

> **Important**
>
> You must have administrator privileges in Microsoft Entra ID (Microsoft Entra ID) in order to edit enterprise applications.

Solution checker fails to export solutions with draft business process flow components

If a solution contains a business process flow component in draft state that has never been previously activated, then solution checker will fail to export the solution for analysis. This error isn't unique to solution checker and is caused by the business process flow having a dependency on a backing (custom) table component that doesn't get created until the business process flow is activated for the first time. This issue can also occur if a business process flow is activated from within solution explorer.

Reference KB Article #4337537: Invalid Export - Business Process table Missing for details about the issue and steps to resolve.

Solution checker fails to export solutions with model-driven app components

If a solution contains a model-driven app, solution checker might fail to export the solution for analysis. This error is caused by role-based security for sharing of apps. If the Power Apps Checker application user doesn't have appropriate access to model-driven apps, any solutions containing them will fail to export with solution checker.

To resolve this issue, grant the **Environment Maker** security role to the Power Apps Checker application user.

Solution checker fails to export patched solutions

If a solution has had a patch applied, solution checker will fail to export the solution for analysis. When a solution has had a patch applied, the original solution becomes locked and it can't be

changed or exported as long as there are dependent patches that exist in the organization that identify the solution as the parent solution.

To resolve this issue, clone the solution so that all patches related to the solution are rolled into the newly created solution. This unlocks the solution and allows the solution to be exported from the system. For more information, see Clone a Solution.

Solution checker won't analyze empty solutions

If solution checker exports a solution that contains no components to analyze, it terminates further processing and consider the run a failure. Ensure that the selected solution submitted for a solution checker analysis contains at least one component.

Solution checker fails to export large solutions

The primary scenario for failure to export a large solution from the environment involves a timeout exception on the export request. This occurs if the request exceeds 20 minutes. Large solutions, such as the Default Solution, may fail to get exported within this time frame, and the check won't complete successfully. If solution checker encounters a timeout during export, it retries three times before it fails to process the job, so it may take over an hour before you receive a failure notification.

The workaround is to create smaller solutions with fewer components to be analyzed. If the large file size of the solution is due to many plug-in assembly components, see guidance to Optimize custom assembly development.

> **Important**
>
> To minimize false positives, ensure you add dependent customizations. When you create a solution and add these components, include the following:
>
> - When you add plug-ins, include the SDK Message Processing Steps for the plug-in.
> - When you add table forms, include the JavaScript web resources attached to the form events.
> - When you add JavaScript web resources, include any dependent JavaScript web resources.
> - When you add HTML web resources, include any dependent scripts that are defined within the HTML web resource.

- When you add custom workflows, include the assembly used within the workflow.

Line number references for issues in HTML resources with embedded JavaScript aren't correct

When HTML web resources are processed within solution checker, the HTML web resource is processed separately than the JavaScript within the HTML web resource. Due to this, the line number of the violation found within `<script>` of the HTML web resource won't be correct.

Web-unsupported-syntax issue for web resources

Solution checker supports global variables for ECMAScript 2015 (ES6) and up to ECMAScript 2018 (ES9) syntax. When solution checker analyzes JavaScript using global variables later than ES6, or syntax later than ES9, a web-unsupported-syntax issue for the web resource is reported.

Multiple violations reported for plug-ins and workflow activities based on call scope

For plug-in and workflow activity rules where the issue is only relevant in the calling context, the solution checker tool starts its analysis at the `IPlugin` interface implementation and traverses a call graph to detect issues within the scope of that implementation. In some cases, many call paths may arrive at the same location where the issue is detected. Since the issue is relevant to the call scope, the tool may report based on that scope to provide a better picture of impact rather than on distinct locations. As a result, multiple issues may reference a single location that should be fixed.

app-formula-issues-high may show different number of errors than those shown in design mode

The solution checker lists the number of the errors found in the app, *including those from tests*. The app checker in design mode only shows errors from screen controls and components. If the error count shown in the app checker in design mode is smaller, open the Test Studio and search for the errors in the expressions used in tests.

Solution checker violations reported for code components

Solution checker reports multiple violations when run on the solution that has code components built in debug mode. One common error that occurs in debug mode is `web-avoid-eval`. Solution checker is targeted to validate code for production use, thus code components must be built using proper buildMode: `npm run build -- --buildMode production` or `msbuild /p:configuration=Release` depending on your building system. More information: Create and build a code component

If there are errors occurring in `bundle.js` introduced by code components, you can exclude this file from analysis. More information: Microsoft.PowerApps.Checker.PowerShell module and PAC CLI

See also

Best practices and guidance for the Dataverse

Best practices and guidance for model-driven apps

Last updated on 06/12/2023

`avoid-2011-api`

Article • 07/18/2022

Don't use the deprecated Microsoft Dynamics CRM 2011 object model. Instead, follow Dataverse Web API documentation.

`avoid-browser-specific-api`

Article • 07/18/2022

Don't use Internet Explorer legacy APIs or browser plug-ins.

Recommendation

Examples of APIs you shouldn't use include `XMLHTTP`, `XMLDOM`, and `DOMDocument`.

`avoid-crm2011-service-odata`

Article • 07/18/2022

Don't target the Microsoft Dynamics CRM 2011 OData 2.0 endpoint.

Recommendation

More information: Odata V2 Deprecation notice.

`avoid-crm2011-service-soap`

Article • 09/09/2023

Don't target the Microsoft Dynamics CRM 2011 SOAP services.

Recommendation

For details on the correct Dataverse services to use, go to Microsoft Dataverse SDK for .NET.

`avoid-dom-form`

Article • 02/14/2023

Don't use the Document Object Model (DOM) of model-driven apps directly. This isn't a supported approach in Dynamics 365 or Power Apps. Use the supported Client API object model instead.

Options

- By default, violations are only reported when certain `Xrm` or `Mscrm` calls are made in the same file, so that violations outside of Dataverse are not reported. The `requireXrm` option, which is `true` by default, can be set to `false` to report all violations regardless of `Xrm` or `Mscrm` usage
  - Example: `"@microsoft/power-apps/avoid-dom-form": ["error", { requireXrm: false }]`

`avoid-dom-form-event`

Article • 02/14/2023

Don't register or unregister handles to the Document Object Model (DOM) events of model-driven apps. This isn't a supported approach in Dynamics 365 or Power Apps. Use the supported Client-side event handlers instead.

Options

- By default, violations are only reported when certain `Xrm` or `Mscrm` calls are made in the same file, so that violations outside of Dataverse are not reported. The `requireXrm` option, which is `true` by default, can be set to `false` to report all violations regardless of `Xrm` or `Mscrm` usage
  - Example: `"@microsoft/power-apps/avoid-dom-form-event": ["error", { requireXrm: false }]`

`avoid-isactivitytype`

Article • 07/21/2022

Replace `Xrm.Utility.isActivityType` with `Xrm.Utility.getEntityMetadata` and don't use in ribbon rules.

Recommendation

For details on the correct Xrm.Utility functions go to the Xrm.Utility.getEntityMetadata documentation.

`avoid-loadtheme`

Article • 02/27/2023

`loadTheme` is a way to provide a theme in global (and only global) scope, which will affect your entire application. We recommend that you replace `loadTheme` with `ThemeProvider`. That way, your application consistently has one way of providing a theme.

Recommendation

For details on the correct `ThemeProvider` usage go to the ThemeProvider documentation.

`avoid-modals`

Article • 10/20/2022

Avoid using `showModalDialog()`.

Recommendation

Instead, go to the documentation for the XRM Navigation API.

`avoid-ui-refreshribbon`

Article • 07/18/2022

Don't use `ui.refreshRibbon` during evaluation of ribbon rules. This API triggers duplicated network calls from ribbon and degrades performance. Use promises and asynchronous patterns.

`avoid-unpub-api`

Article • 07/18/2022

Avoid using unpublished (private) functionality as it is not guaranteed to continue working and utilizing it may break your code in the future.

Recommendation

Avoid using unsupported namespaces, global objects, global functions, and CSS styles.

`avoid-window-top`

Article • 07/21/2022

Don't use `window.top` or `window.parent.parent`. Using either of these fields will likely result in cross-origin security errors when hosted outside of the primary web client. Develop an alternate approach. For more information, go to best practices: avoid-window-top.

`do-not-make-parent-assumption`

Article • 07/21/2022

Don't assume that the usage of `global.parent`, `window.parent`, `window.opener`, or `this.parent` from JavaScript web resources will result in the same context across versions or client types, for example mobile, tablet, and browser. Consider a different approach when attempting to interact with one of these contexts. For more information, go to best practices: avoid-window-top.

`use-appsidepane-api`

Article • 10/24/2023

Use Xrm.App.sidePanes.createPane instead of Xrm.Panels.loadPanel.

Recommendation

The Xrm.Panels.loadPanel API is being replaced with Xrm.App.sidePanes.createPane because the former only supports a single pane while the latter supports multiple panes. For more information, go to Use with Xrm.App.panels.loadPanel.

`use-async`

Article • 07/18/2022

Interact with HTTP and HTTPS resources asynchronously.

Recommendation

For more information, go to asynchronous HTTP best practices.

`use-cached-webresource`

Article • 07/21/2022

Don't use a relative or absolute URL that retrieves a web resource from the server rather than from the cache. Web resource URLs should always be relative to ensure caching support. Additionally, don't include `/WebResources/` within the URL path as it will refer to the caller's default organization.

`use-client-context`

Article • 07/18/2022

Don't use `Xrm.Page`. Although `Xrm.Page` is deprecated, `parent.Xrm.Page` will continue to work with HTML web resources embedded in forms as this is the only way to access the form context from the HTML web resource. For a more comprehensive list of appropriate replacements for `Xrm.Page` functionality, go to Client API Deprecations.

`use-getsecurityroleprivilegesinfo`

Article • 07/18/2022

Avoid `userSettings.securityRolePrivileges`. This returns only privilege GUIDs, which are difficult to work with. You might be making extra network requests to get further details about the privilege GUID and that degrades performance. Use userSettings.getSecurityRolePrivilegesInfo() instead, which gives you more details about each security role privilege.

`use-global-context`

Article • 07/18/2022

Don't use `Xrm.Page.context`. For access to the global context use `Xrm.Utility.getGlobalContext`.

Recommendation

For more detail on appropriate replacement APIs, go to Client API deprecations.

`use-grid-api`

Article • 07/21/2022

Don't use deprecated APIs in the grid object model. Replace these calls with the `data` property of the grid row object.

Recommendation

For more detail on the specific replacement calls, go to Client API deprecations.

`use-navigation-api`

Article • 07/18/2022

Use navigation API parameters instead of utility parameters.

Recommendation

For more information on deprecations and the associated replacements, go to Client API deprecations.

`use-offline`

Article • 07/18/2022

Don't use `Xrm.Mobile.offline`. Instead, use `Xrm.WebApi.offline.`

Recommendation

For a more comprehensive list of deprecations and replacements, go to Client API deprecations.

`use-org-setting`

Article • 07/18/2022

Use organization settings. Specifically:

- Calls to `Xrm.Page.context.getIsAutoSaveEnabled()` should be replaced with `Xrm.Utility.getGlobalContext().organizationSettings.isAutoSaveEnabled`
- Calls to `Xrm.Page.context.getOrgLcid()` should be replaced with `Xrm.Utility.getGlobalContext().organizationSettings.languageId`
- Calls to `Xrm.Page.context.getOrgUniqueName()` should be replaced with `Xrm.Utility.getGlobalContext().organizationSettings.uniqueName`

Recommendation

For more information, go to Client API deprecations.

`use-relative-uri`

Article • 07/18/2022

Don't use absolute Microsoft Dataverse endpoint URLs.

Recommendation

Don't use an absolute reference to a Microsoft Dataverse endpoint. Microsoft Dataverse endpoint URLs should be constructed using the appropriate client API function along with a relative path to the desired endpoint.

`use-utility-dialogs`

Article • 02/14/2023

Use Web API dialogs in form and ribbon commands.

Recommendation

Alerts should be replaced with `Xrm.Navigation.openAlertDialog`, while confirms should be replaced with `Xrm.Navigation.openConfirmDialog`. For more information, go to Xrm.Navigation.

Options

- By default, violations are only reported when certain `Xrm` or `Mscrm` calls are made in the same file, so that violations outside of Dataverse are not reported. The `requireXrm` option, which is `true` by default, can be set to `false` to report all violations regardless of `Xrm` or `Mscrm` usage
  - Example: `"@microsoft/power-apps/use-utility-dialogs": ["error", { requireXrm: false }]`

Catalog in Power Platform

Building from scratch every time by recreating branding, layouts, links, complex connectors and flows, and more is painful and error-prone. Organizations where developers and makers build and share customized and reusable components and templates get more value from Power Platform. Successful organizations adopt a *fusion teams* model where pro-developers, makers, and admins all work together to deliver the best solutions for their users, and derive the highest value possible from Power Platform.

> **ⓘ Important**
>
> - The catalog needs to be set up by an admin before you can use it, including permissions. More information Administer the catalog.
> - The catalog only works with Microsoft Dataverse environments. Environments without Dataverse aren't supported at this time.
> - You need a Managed Environment in order to submit solutions to the catalog. However, you can install catalog items from any environment. More information: Managed Environments overview

In any organization, there might be many components and templates distributed among many environments. The catalog in Power Platform enables developers and makers to:

- Crowd-source and find templates and components within their organization easily
- Find and install the latest and authoritative version of a component
- Get started with templates and components that provide immediate value

Components include things like:

- AI prompts
- Copilot app templates
- AI plugins
- Power Platform dataflows
- Custom connectors
- Power Apps component framework controls
- Power Automate flows
- Canvas apps
- Model-driven apps

Templates are items that represent an advanced starting point for components. Templates connect to their enterprise systems and resources and utilize an organization's themes.

Terminology

These terms are important to understand when using the catalog:

Expand table

Term	Description
Catalog Item	The unit of interaction with the catalog is called an item. - An item is what is being submitted to or installed from the catalog. - Typically, an item is a Dataverse solution or package deployer package. An item can contain a fully built application, a template for a Power App or flow, or a Power Platform code-first component such as a custom connector or Power Apps component framework (PCF) control.
Catalog Publisher	The owning entity of the application. For example, the publisher can be the human resources IT team or another line of business team. - A group of people in an organization that are responsible for managing its lifecycle. - Support and engineering contacts can be provided. - Note that the catalog publisher is different from the solution publisher.
Submission	The act of sharing an item involves submitting it to the catalog. This act creates a *submission request* in the system.
Submission Request	The result of a submission. This request can be approved or declined. When auto approval is configured for the catalog, no approval is required.
Discovery	Act of finding items within a catalog by authorized users.
Acquisition	Act of installing the item to a target environment by a developer.

Catalog basics

The catalog needs to be set up by an admin before you can use it, including privileges. More information: Administer the catalog.

You might choose to have multiple catalogs across environments for different regions, departments, or any group. Most organizations only need one catalog. A catalog admin can set up new catalogs in the Power Platform admin center.

You can submit unmanaged solutions to the catalog for other makers to install and use for themselves, as one of these two catalog item types:

- A **template**: A copy of your unmanaged solution that other makers can edit however they choose. Updates to the original unmanaged solution won't update templates, as templates are no longer "connected" to the solution it came from. If you want other

makers to have the ability to change the catalog components as they see fit, use a template.

> **Note**
>
> Currently, certain components aren't supported for use as templates. The supported components for templates are:
> - Table and its sub components like forms, saved queries, relationships, table maps.
> - Environment variables.
> - Canvas apps.
> - Modern workflows.
> - `AiModel` and its subcomponents like `AiConfig`.
> - Web resources.
> - Power Apps component framework components. This includes modern solution aware components created using the Power Apps component framework. For example custom API, `AiPlugin`, and `AIPlugin` operation.

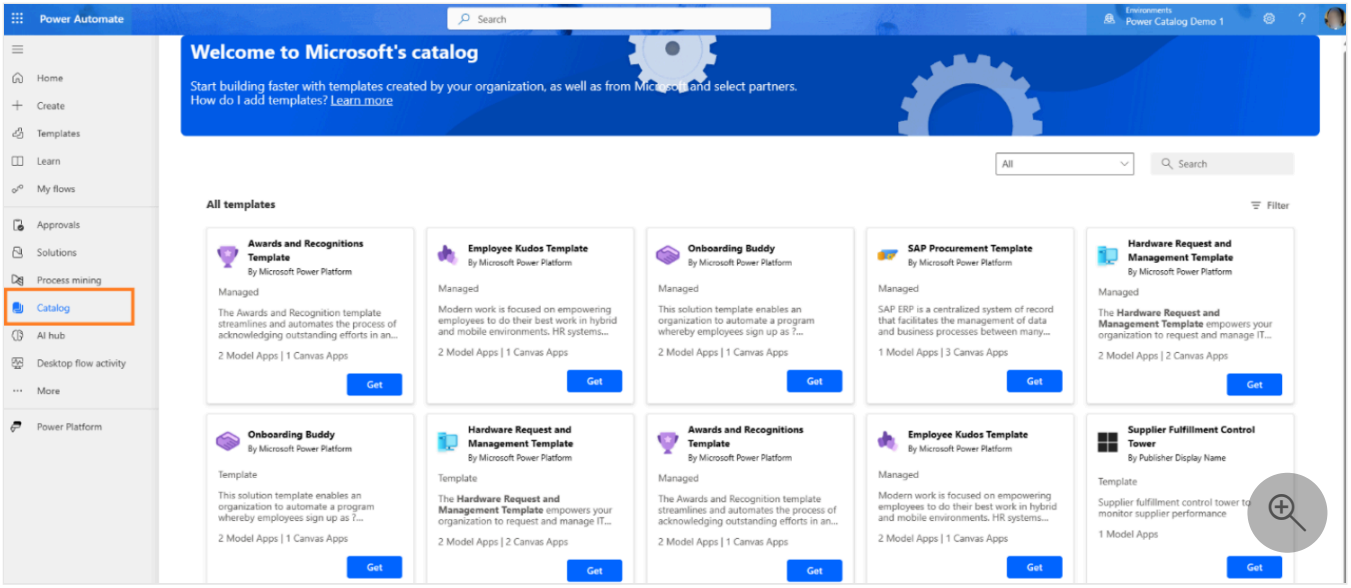
- A **managed item**: Managed items can be updated with more versions as the original solution is updated, but are generally restricted from editing. If you want makers to use your solution *as is* and you also want to keep copies updated with your changes in the future, use a managed item. Managed items are useful for a variety of scenarios for makers:
  - Use the assets of the managed item as is and make no changes.
  - Build custom functionality on top of the assets in the managed item without changing the original components.
  - Use the assets to assist with building your own solutions. For example, PCF components, custom connectors, and so on.

Admins can administer catalogs, manage access and security, approve catalog items, and more in the Catalog Manager, which is the dedicated workspace for admins of the catalog. The catalog itself is designed for makers of any level of experience to publish, discover, and acquire useful catalog items for themselves.

Discovery

The catalog area is available in Power Apps and Power Automate. On the catalog page are tiles representing all catalog items available to your organization. This includes items published by your organization, approved partners who have access to your environments, as well as many

published by Microsoft, such as AI prompts, Copilot app templates, dataflow templates, enterprise templates, PCF controls, plug-ins, and more.



On the page itself are the following controls:

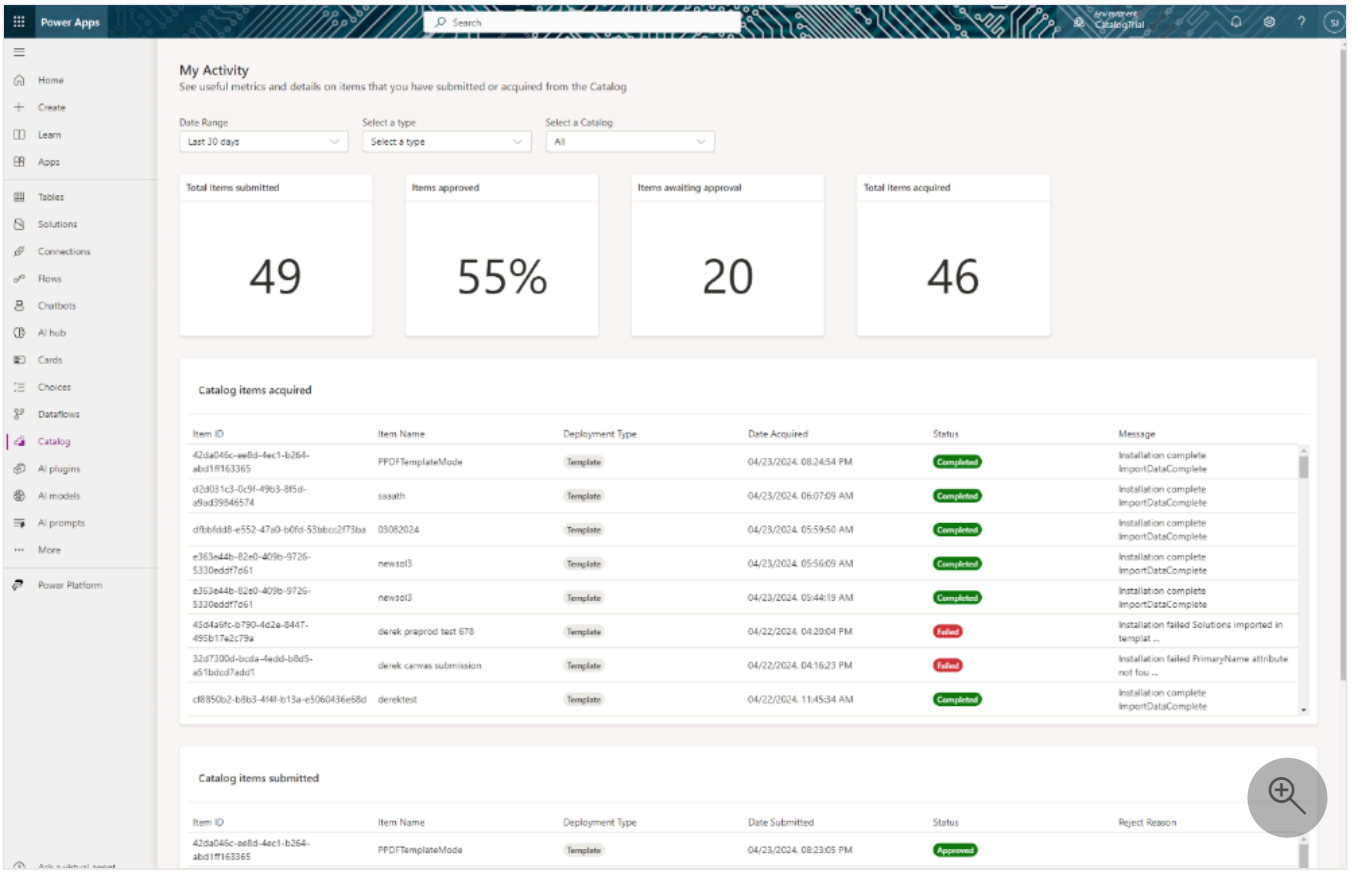
- Search box (searches the titles of available catalog items)
- Filters (filters for publisher, type, category, and more)
- Catalog selector (some organizations might choose to create multiple catalogs for regions or departments). Users can have access to one or more catalogs.

The list of Microsoft published catalog items that will be available in your catalog out of the box are:

- AI Prompts
- Copilot app templates
- Enterprise templates
- Power Platform dataflows
- PCF controls
- Plug-ins
- Retail cloud templates
- Copilot agents

My Activity

Linked in the banner of the catalog page is the **My Activity** page. This shows useful info on items you have submitted and acquired, including approval status.



Filters on top of the page allow you to look at different date ranges, types of catalog items, and across different catalogs (if you have multiple catalogs set up in your tenant).

- Metrics on this page:
  - **Total items submitted**: Sum of all items you submitted.
  - **Items approved**: Sum of all items you submitted that were approved by your admin.
  - **Items awaiting approval**: Sum of all items you submitted that your admin has not responded to yet.
  - **Items acquired**: Sum of all items you acquired from the catalog.

- Catalog items acquired table:
  - **Item ID**: Unique identifier for the acquired item.
  - **Item name**: Display name for the acquired item.
  - **Deployment type**: Whether item is a template or managed.
  - **Date acquired**: Date item was last acquired.
  - **Status**: Current status of the acquired item, such as completed or failed.
  - **Message**: Status message, can explain why an item failed.

- Catalog items submitted table:
  - **Item ID**: Unique identifier for the submitted item.
  - **Item name**: Display name for the submitted item.
  - **Deployment type**: Whether item is a template or managed.
  - **Date submitted**: Date item was last submitted.

- **Status**: Current status of the submitted item, such as approved, rejected, or pending approval.
- **Reject reason**: Why a submitted item was rejected by an admin.

Frequently asked questions (FAQ)

The following are frequently asked questions related to catalog in Power Platform.

Where do I provide feedback on catalog?

Ask questions and submit feedback at: github.com/microsoft/PowerPlatform-Catalog/discussions

What are the items in the catalog?

The catalog items are actually packages. A package contains one or more solutions and some metadata. The Power Apps experiences can submit individual solutions. The Power Platform CLI experience can be used to create a package containing multiple solutions.

What are solution templates? What does it mean to install an unmanaged solution?

The use of a solution as a *solution template* is a new scenario we're making available with the catalog. When a maker acquires a solution template, they'll get a new unmanaged copy of that solution. Multiple copies of that unmanaged solution can exist in the same environment and each has a unique identifier.

Catalog item install looks a lot like solution import. Are they the same thing?

A catalog item is a package that contains one or more solutions. Installation of a package containing a single solution with no other package configuration looks similar to a solution import.

Will the catalog support PCF components within solutions? What about other solution component types?

Yes, distribution of PCF controls is made easier by using a catalog. Any solution component can be contained in a solution in the catalog.

Can the templates have multiple owners so people know who to reach out to for more information?

Yes, ownership information is part of the metadata for a catalog item. The ownership can be a group with multiple people in it.

Is there a way to add documentation such as a user guide when submitting a template?

Yes, documentation links are part of the metadata for a catalog item.

Can catalogs be shared to security groups?

Yes, catalog permissions can be targeted at specific security groups.

Next steps

Learn how to administer the catalog.

Administer the catalog

Learn how to view, submit, and install catalog items.

Submit and install catalog items

Ask questions or contact the project team on GitHub.

github.com/microsoft/PowerPlatform-Catalog/discussions

Last updated on 02/04/2026

View, submit, and install catalog items

Article • 11/14/2024

In any organization, there might be many components and templates distributed among many environments. The catalog in Power Platform enables developers and makers to:

- Crowd-source and find templates and components within their organization easily
- Find and install the latest and authoritative version of a component
- Get started with templates and components that provide immediate value

> **Important**
>
> You need a Managed Environment in order to submit items to the catalog. You can however install catalog items from any environment. More information: Managed Environments overview

Before reading this article, you should:

- Learn about the catalog
- Learn how administrators setup and configure the catalog

> **Note**
>
> Developers can also use the Power Platform CLI, Dataverse SDK for .NET, and Dataverse Web API to perform the operations described here. Catalog in Power Platform for developers

Access controls

The catalog has a separate set of access controls from Power Platform. This means makers in a given environment don't automatically get permissions to publish and acquire items from the catalog. There are four access roles:

Expand table

Role	Enables user to:
Catalog Submitter	Submit items to the catalog

Role	Enables user to:
Catalog Consumer	Discover and install items from the catalog
Catalog Approver	Approve submissions to the catalog. Catalog approvers can be users from your central IT department or line of business that your organization wants to empower to participate in the approvals process.
Catalog Admin	Administer the catalog.

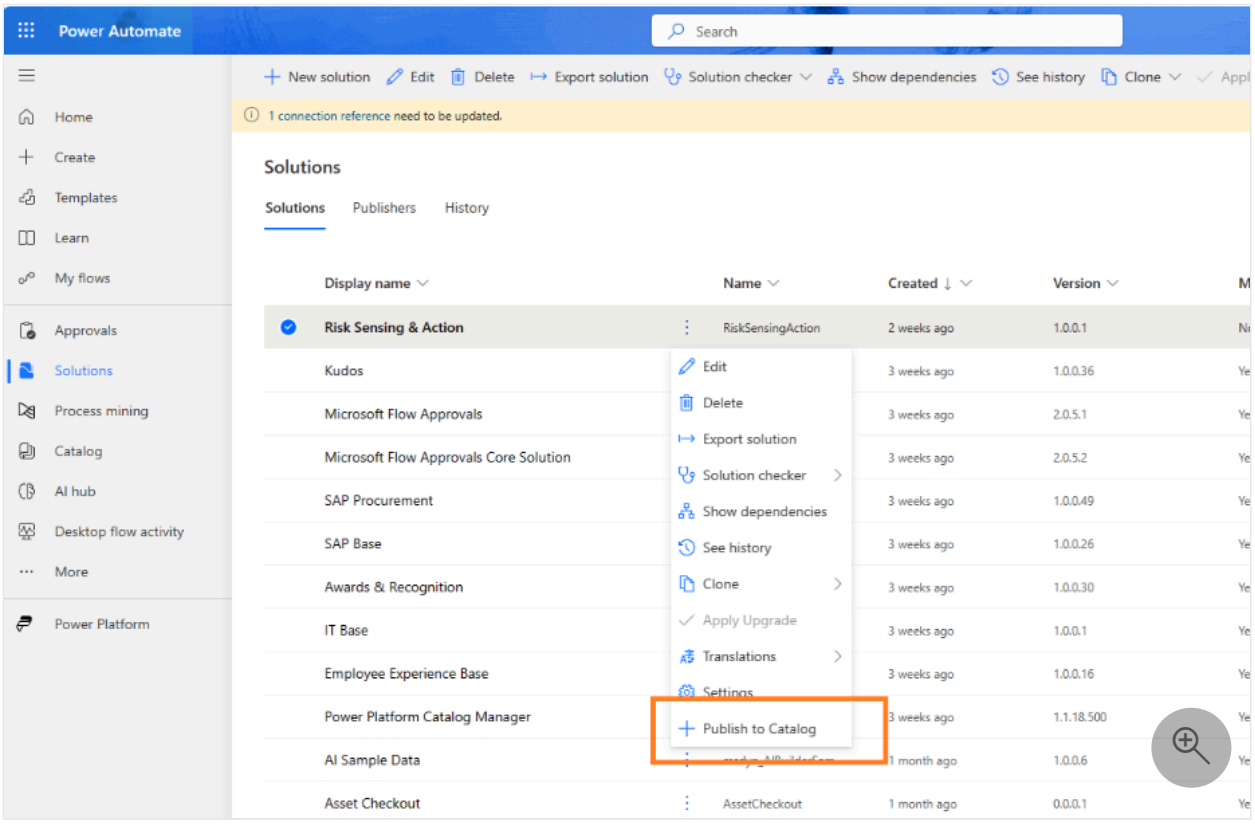
Any environment that has a catalog sees these roles inside of Power Platform admin center, and can be assigned to groups or individuals just like any other security roles.

Power Platform admins and system customizers already have full access to the catalog. However, don't assign these roles for generally managing catalog access, and instead use one of the roles in the previous table.

Submission

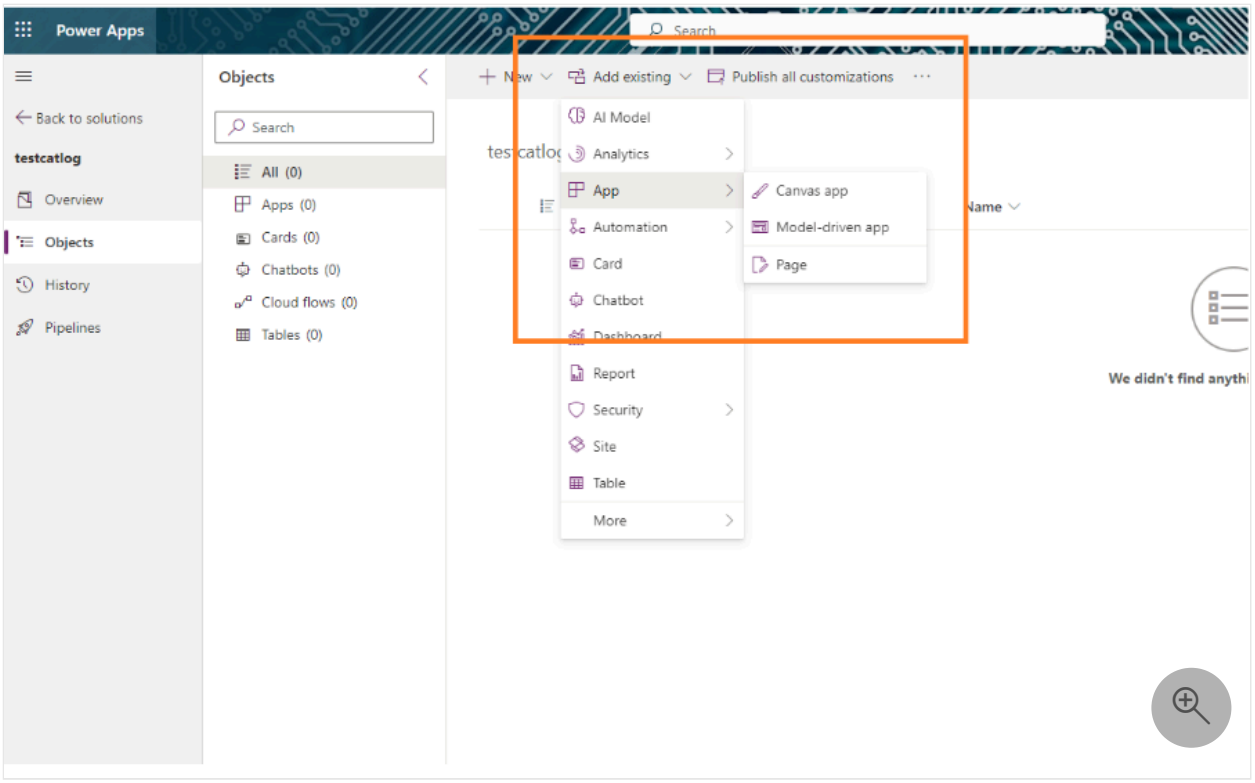
Unmanaged solutions in your environment can be published to the catalog. In your normal course of creating solutions, you might decide that what you're building would be useful as a reusable artifact for yourself or others in your organization.

Navigate to your solutions page, and any unmanaged solution has a new option for **Publish to Catalog** in the three vertical dots next to the solution display name. This option is disabled for anyone without the **Catalog Submitter** role.

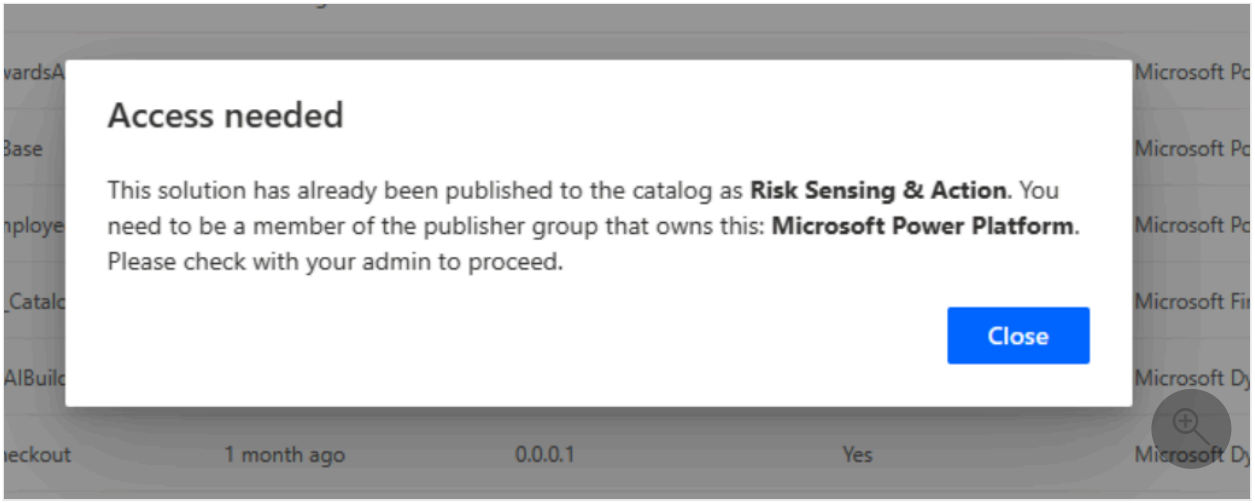


> **Note**
>
> This option is not enabled for managed solutions. You can't submit managed solutions into the catalog.

You can't submit an app, flow, or other component directly into the catalog. You must first ensure it exists in an unmanaged solution. You can add it to an unmanaged solution by selecting **New solution** on top of the **Solutions** area or by selecting an existing one from the list, and then selecting **Add existing** in the top menu in the solution itself, and adding your desired component (such as an app or a flow).



> **Note**
>
> Any maker with the **Catalog Submitter** role in a given environment will be able to publish. However, after a solution is published, a maker needs to be a part of the publishing group to re-publish that same solution again. You see this error if you don't have access, which you can request from your admin.



When you select **Publish to Catalog**, there's a wizard with four steps:

Step 1: Add your solution

1. **Select a catalog**: If you have multiple catalogs set up, you can select which catalog you would like to publish this item to.

2. Select **managed item** or **template** for your submission.

    - **Managed items** can't be edited by other makers, your environment shares a single copy, and that copy can be versioned with updates over time. Makers can then update their solutions with your updates.
    - A **template** is a standalone copy that can be edited, makers can have as many copies as they want, and won't automatically update solutions in environments when new updates are published.

3. **Select a primary component**: A solution might have many components in it. You might want makers to open a specific component for editing after they install it, such as opening Canvas Studio for your canvas app. Set the primary component to whichever you think is the "focus" of the solution, otherwise, set **Unspecified**, and installers go the solution itself.

Resubmitting items

If you're resubmitting an item, you get an alert message notifying you about the resubmission. You're prompted with a version field. The last digit of the version number is automatically incremented for you. You can modify the version number if you like, but don't set the version number to a lower number from this interface. Updating the version number in the catalog submission updates the version number in the original solution as well.

Keep these points in mind when resubmitting items:

- Resubmitting a **template** replaces the previous item in the catalog, and doesn't affect any solutions already using the previous catalog item. Makers who now install your resubmission get your newest version.
- Resubmitting **managed items** will update the catalog item with your most recent version, and the next time makers see your catalog item in an environment where that catalog item was already installed, they'll see an **Update** button, which installs the most recent changes to the managed item into that environment, and then all solutions using that managed item will be updated as well.
- You can't resubmit a managed item as a template, and vice versa.

Step 2: Catalog info

Enter the following information:

Expand table

Field	Instructions
Title	Rename the item if needed before it gets submitted to catalog. The default is the solution name. You can't edit title when submitted a managed item.
Description	Provide details on this catalog item. Makers read your description in the catalog gallery to find out more about it.
Business justification	By default, your admin must approve all catalog submissions before they're added to the catalog. This field provides a justification for your submission to your admin.
Works with	A flow that you're building might be intended to **work with** canvas apps and model apps, for example. Indicate here what your catalog item is intended to work with.
Business categories	Select up to five business categories that describe your catalog item. Makers use these categories to search for catalog items that are of interest to them.
Publisher	Select an existing publisher in this environment, or create a new publisher, to connect to this catalog submission. A maker can't resubmit a catalog item unless they're part of the publishing group that submitted the item the first time.
Author	This defaults to the current user, but you can change it if needed

Step 3: Solution info

Enter the following information:

Expand table

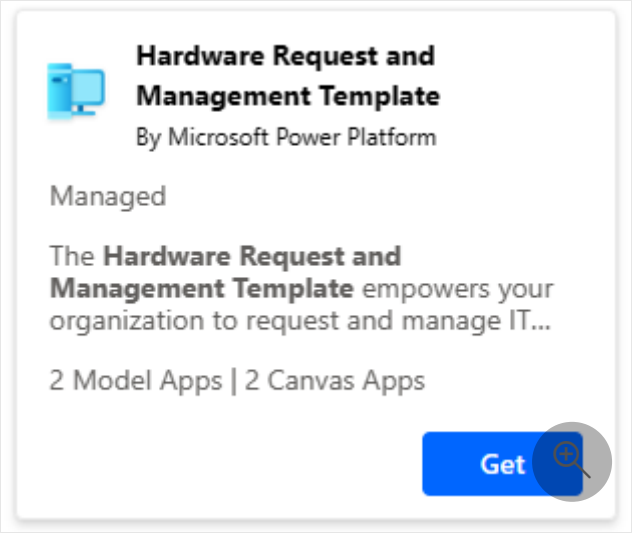
Field	Instructions
Solution icon	Attach an icon to your solution (216 x 216 pixel) to help identify it in the catalog gallery
template images	Attach some screenshots or other visuals that can help makers understand more details about this catalog item
Help link	Add the URL for any help documentation
Privacy policy link	Add the URL for any privacy policy documentation
Legal terms link	Add the URL for any legal terms documentation

Step 4: Review and finish

This last step provides a summary of all the info you provided. If everything looks good, select **Submit** to publish to the catalog. As previously mentioned, if your catalog requires an approver before submission can complete, you need to wait for approval. Contact your admin or approver if you need the process accelerated.

Acquisition

As makers browse the **Discover** page and find catalog items that are useful for them, they can acquire those artifacts and install a copy into their environment to start using or building with. Selecting **Get** on any template or managed item opens a details popup that contains information like description, links, included components, business categories, expandable screenshots, and more.



If the details meet what the maker is looking for, they can select **Get** on the details popup to start the acquisition wizard.

> ⓘ Note
>
> If the catalog item is a managed item, and has already been installed in the environment, you can't install the managed item again, as the environment shares that managed item. If the managed item has already been installed in an environment, and an update has been published to the managed item, you see an **Update** button instead of **Get**, which updates the shared managed solution in the environment. If the managed item was already updated, and no new updates have been published, the **Update** button is disabled until a new update is published.

The acquisition wizard has four steps:

Step 1: Check connections

This step checks the connections for the solution to make sure they're working for you in your environment. If there are any connection issues, you see a red status notification, and can resolve by selecting the ellipsis (...) next to the problematic connection to update it.

Step 2: Environment variables

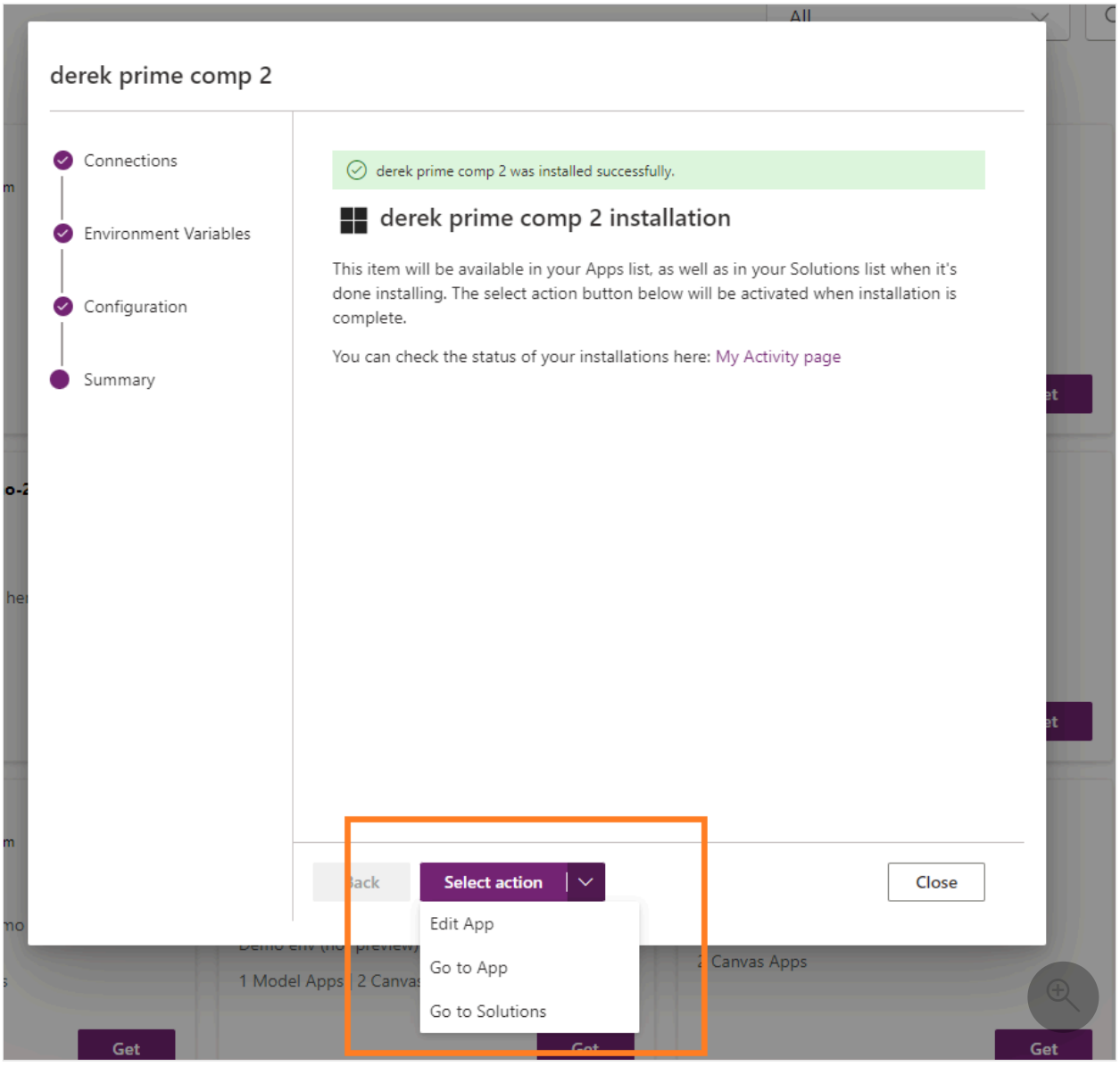
The publishing maker might decide some environment variables need to be specified before you can acquire the solution. Typically, these variables are set to some default value. If unsure of what to fill out, check with the author of the catalog item or in the details of the catalog listing, available on the catalog details. More information: Step 2: Catalog info

Step 3: Configuration

The publishing maker determines this step, if they want you to provide any additional information for using the solution. This might include things like, "*What days of the week do you run a report?*". If no additional information needed, this step is blank.

Step 4: Summary

When you reach the final step, the summary screen, you're now waiting for the catalog item to install into your environment. Depending on the complexity, this might take from less than a minute to several minutes. When complete, you're given several options in the dropdown, depending on what's included in the catalog item, and if the author set a primary component. The following is an example where a canvas app is the primary component of the catalog item:



In this example, you can:

- **Edit App**: Opens the app in Canvas Studio
- **Go to App**: Takes you to the app screen, if you aren't ready to edit yet
- **Go to Solutions**: Opens the solution file

Depending on what is contained in the catalog item package, different options appear.

After the catalog item is installed, you can find the catalog item:

- In the **Unmanaged solutions** list in the solutions page if the catalog item was a **template**
- In the **Managed solutions** list in the solutions page if the catalog item was a **managed item**

As previously mentioned, templates can be acquired as many times as you like, each as its own copy. The catalog appends a suffix to the unmanaged solution display name to differentiate the different copies. Managed items can be acquired only once into the

managed solutions list, and all makers in the environment share the same managed solution. This means you can't acquire a managed item back into the same environment it was published from and the **Update** button is disabled.

See also

Catalog in Power Platform)
Administer the catalog
Catalog in Power Platform for developers

Feedback

Was this page helpful? Yes No

Provide product feedback

Self-service data prep with dataflows

Article • 02/15/2022

As the volume of data continues to grow, so does the challenge of shaping that data into well-structured, actionable information. You want data that's ready for apps, AI workloads, or analytics so that you can quickly turn volumes of data into actionable insights. With self-service data prep in the Power Apps portal, you can transform and load data to Microsoft Dataverse or your organization's Azure Data Lake Storage Gen2 account with just a few clicks.

Dataflows were introduced to help organizations unify data from disparate sources and prepare it for consumption. You can easily create dataflows using familiar, self-service tools to ingest, transform, integrate, and enrich big data. When creating a dataflow, you will define data source connections, ETL (extract, transform, load) logic, and destination to load the resulting data to. Once created, you can configure a dataflow's refresh schedule to indicate how frequently it should run. In addition, the new model-driven calculation engine makes the process of data preparation more manageable, more deterministic, and less cumbersome for dataflow customers. With dataflows, tasks that once required a data IT organization to create and oversee (and many hours or days to complete) can now be handled with a few clicks by individuals who aren't even data scientists, such as app creators, business analysts and report creators.

Dataflows store data in tables. An table is a set of rows used to store data, similar to how a table stores data within a database. Customers can define custom table schema or leverage the Common Data Model's standard tables. The Common Data Model is a shared data language for business and analytical applications to use. The Common Data Model metadata system enables consistency of data and its meaning across applications and business processes such as Power Apps, Power BI, some Dynamics 365 apps (model-driven apps), and Azure, which store data in conformance with the Common Data Model. A dataflow's resulting tables can then be stored in either of the following:

- **Dataverse.** Lets you securely store and manage data that's used by business applications built using Power Apps and Power Automate.
- **Azure Data Lake Storage Gen2.** Lets you collaborate with people in your organization using Power BI, Azure Data, and AI services or custom-built line-of-business applications that read data from the lake. Dataflows that load data to an Azure Data Lake Storage Gen2 account store data in Common Data Model folders. **Common Data Model folders** contain schematized data and metadata in a standardized format to facilitate data exchange and to enable full interoperability

across services that produce or consume data stored in an organization's Azure Data Lake Storage account as the shared storage layer.

You can use dataflows to ingest data from a large and growing set of supported on-premises and cloud-based data sources including Excel, Azure SQL Database, SharePoint, Azure Data Explorer, Salesforce, Oracle database, and more.

After selecting the data source, you can use the Power Query low-code/no-code experience to transform the data and map it to standard tables in the Common Data Model or create custom tables. Advanced users can directly edit a dataflow's M-language to fully customize dataflows, similar to the Power Query experience that millions of Power BI Desktop and Excel users already know.

Once you've created and saved a dataflow, you will need to run it in the cloud. You can choose to trigger a dataflow to run manually or schedule the frequency for the Power Platform Dataflow service to run it for you. When a dataflow completes a run, its data is available to use. To get dataflow data loaded into Dataverse, the Common Data Service connector can be used in Power Apps, Power Automate, Excel, and all other applications that support the Dataverse connector. To get from dataflows stored in your organization's Azure Data Lake Storage Gen2 account, you can used the Power Platform Dataflow connector in Power BI Desktop or access the files directly in the lake.

How to use dataflows

The previous section provided background on dataflows technology. In this section, you get a tour of how dataflows can be used in an organization.

> ⓘ **Note**
>
> You must have a paid Power Apps plan to use dataflows, but you are not charged separately for using dataflows.

Load data to Dataverse

Dataflows can be used to populate tables in the Common Data Service that are then used in Power Apps applications. With a few clicks, you can integrate data from online and on-premises data sources.

Extend the Common Data Model for your business needs

For organizations that want to extend and build upon the Common Data Model, dataflows enable business intelligence professionals to customize the standard tables or create new ones. This self-service approach to customizing the data model can then be used with dataflows to build Power BI dashboards that are tailored to an organization.

Extend your capabilities with Azure Data and AI services

Power Platform dataflows can be configured to store dataflow data in your organization's Azure Data Lake Storage Gen2 account. When an environment is connected to your organization's data lake, data scientists and developers can leverage powerful Azure products such as Azure Machine Learning, Azure Databricks, Azure Data Factory, and more.

For more information about Azure Data Lake Storage Gen2 and dataflows integration, including how to create dataflows that reside in your organization's Azure Data Lake, see Connect Azure Data Lake Storage Gen2 for dataflow storage.

Summary of self-service data prep for big data in Power Apps

There are multiple scenarios and examples where dataflows can enable you to get better control—and faster insights—from your business data. Other people in your organization can leverage dataflows either via Dataverse, the Power Platform Dataflow connector in Power BI, or via direct access to Dataflow's **Common Data Service** folder in your organization's Azure Data Lake Storage Gen2 account. Using a standard data model (schema) defined by the Common Data Model, business applications can depend on a table's schema, and be abstracted from how the data was created or from which data source. When a dataflow completes a scheduled run, the data is ready for modeling and creation of apps, flows, or BI insights in a very short period... in what used to take months, or longer, to create.

The standardized format of the Common Data Model allows people in your organization to create apps that generate quick, easy, and automatic visuals and reports. Those include, but aren't limited to:

- Mapping your data from various sources to standard tables in the Common Data Model to unify data and leverage the known schema to drive out-of-the-box applications.
- Creating your own custom tables to unify data across your organization.

- Creating Power BI reports and dashboards that leverage dataflow data.
- Creating integration with Azure Data and AI services via your organization's Azure Data Lake Storage Gen2 account.

Next steps

This article provided an overview of self-service data prep in the Power Apps portal, and the ways you can use it. The following topics go into more detail about common usage scenarios for dataflows:

- Creating and using dataflows in Power Apps
- Add data to a table in Dataverse
- Connect Azure Data Lake Storage Gen2 for dataflow storage
- Using an on-premises data gateway in Power Platform dataflows

For more information about Power Query and scheduled refresh, you can read these articles:

- Query overview in Power BI Desktop
- Configuring scheduled refresh

For more information about the Common Data Model, you can read its overview article:

- Common Data Model - overview

Create and use dataflows in Power Apps

With advanced data preparation available in Power Apps, you can create a collection of data called a dataflow, which you can then use to connect with business data from various sources, clean the data, transform it, and then load it to Microsoft Dataverse or your organization's Azure Data Lake Gen2 storage account.

A dataflow is a collection of tables that are created and managed in environments in the Power Apps service. You can add and edit tables in your dataflow, as well as manage data refresh schedules, directly from the environment in which your dataflow was created.

Once you create a dataflow in the Power Apps portal, you can get data from it using the Dataverse connector or Power BI Desktop Dataflow connector, depending on which destination you chose when creating the dataflow.

There are three primary steps to using a dataflow:

1. Author the dataflow in the Power Apps portal. You select the destination to load the output data to, the source to get the data from, and the Power Query steps to transform the data using Microsoft tools that are designed to make doing so straightforward.

2. Schedule dataflow runs. This is the frequency in which the Power Platform Dataflow should refresh the data that your dataflow will load and transform.

3. Use the data you loaded to the destination storage. You can build apps, flows, Power BI reports, and dashboards or connect directly to the dataflow's Common Data Model folder in your organization's lake using Azure data services like Azure Data Factory, Azure Databricks, or any other service that supports the Common Data Model folder standard.

The following sections look at each of these steps so you can become familiar with the tools provided to complete each step.

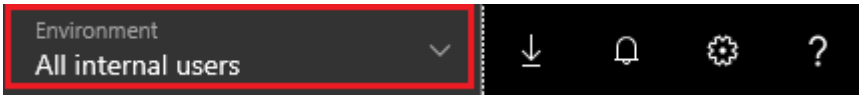
Create a dataflow

Dataflows are created in one environment. Therefore, you'll only be able to see and manage them from that environment. In addition, individuals who want to get data from your dataflow must have access to the environment in which you created it.

> **ⓘ Important**
>
> - Creating dataflows is currently not available with Power Apps Developer Plan licenses.

- The Firefox web browser currently isn't supported for the "Download refresh history file" action in Power Apps dataflows. More information: [You receive the error message "There was a problem refreshing the dataflow"]

1. Sign in to Power Apps, and verify which environment you're in, find the environment switcher near the right side of the command bar.

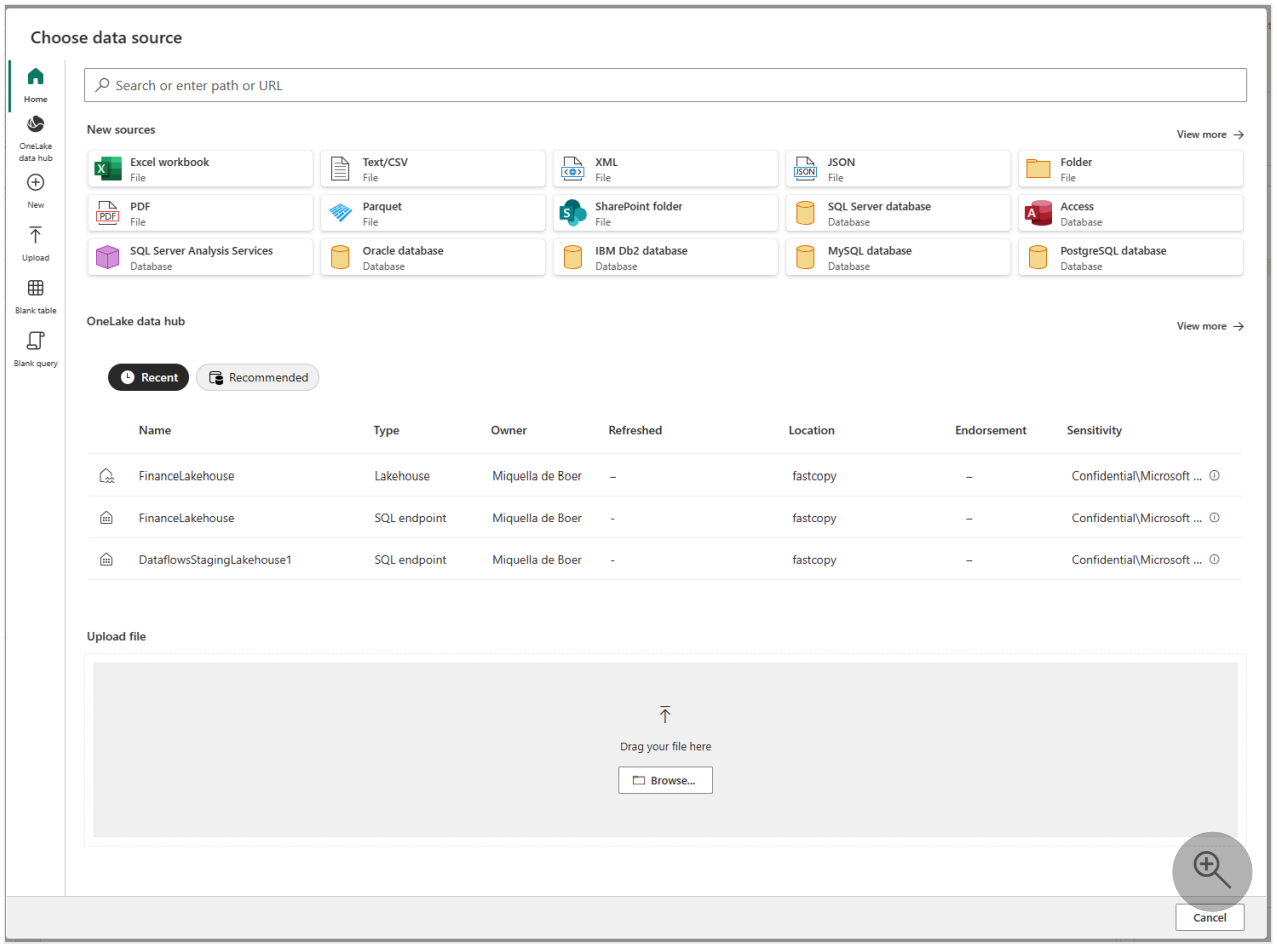


2. On the left navigation pane, select **Dataflows**. If the item isn't in the side panel pane, select ...More and then select the item you want.

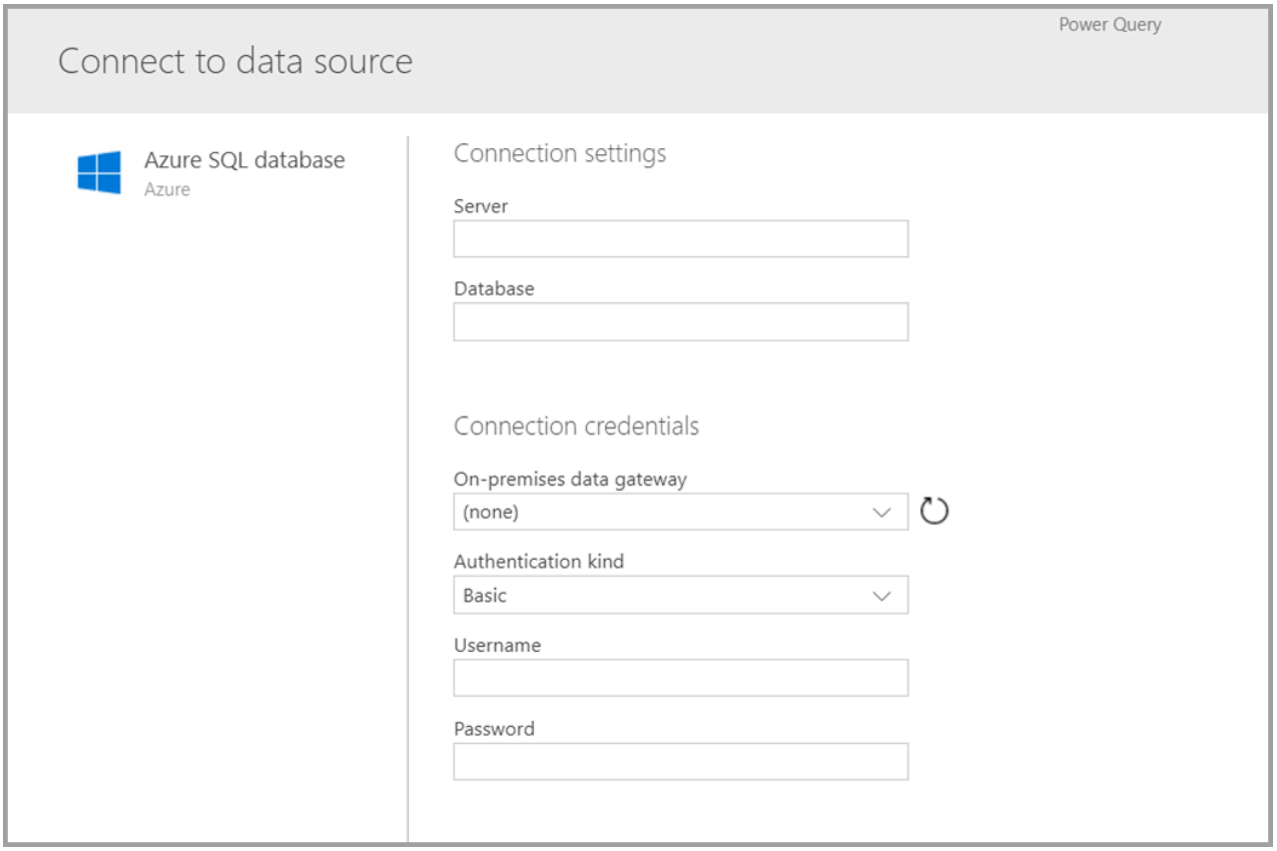
3. Select **New dataflow**. On the **New Dataflow** page, enter a **Name** for the dataflow. By default, dataflows store tables in Dataverse. Select **Analytical entities only** if you want tables to be stored in your organization's Azure Data Lake storage account. Select **Create**.

   > **Important**
   >
   > There's only one owner of any dataflow—the person who created it. Only the owner can edit the dataflow. Authorization and access to data created by the dataflow depend on the destination you loaded data to. Data loaded into Dataverse will be available via the Dataverse Connector and requires the person accessing the data to be authorized to Dataverse. Data loaded into your organization's Azure Data Lake Gen2 storage account is accessible via the Power Platform Dataflow connector and access to it requires membership within the environment it was created in.

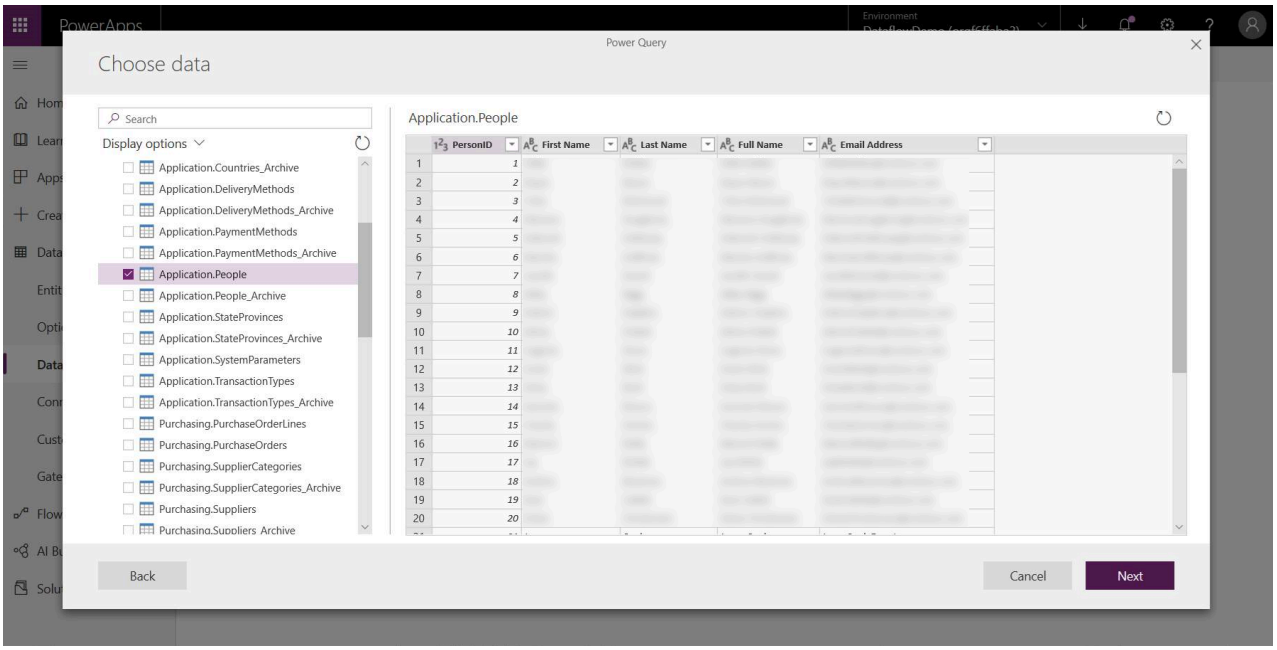
4. On the **Choose data source** page, select the data source where the tables are stored. The selection of data sources displayed allows you to create dataflow tables.



5. After you select a data source, you're prompted to provide the connection settings, including the account to use when connecting to the data source. Select **Next**.



6. Once connected, you select the data to use for your table. When you choose data and a source, the Power Platform Dataflow service will subsequently reconnect to the data source in order to keep the data in your dataflow refreshed, at the frequency you select later in the setup process.

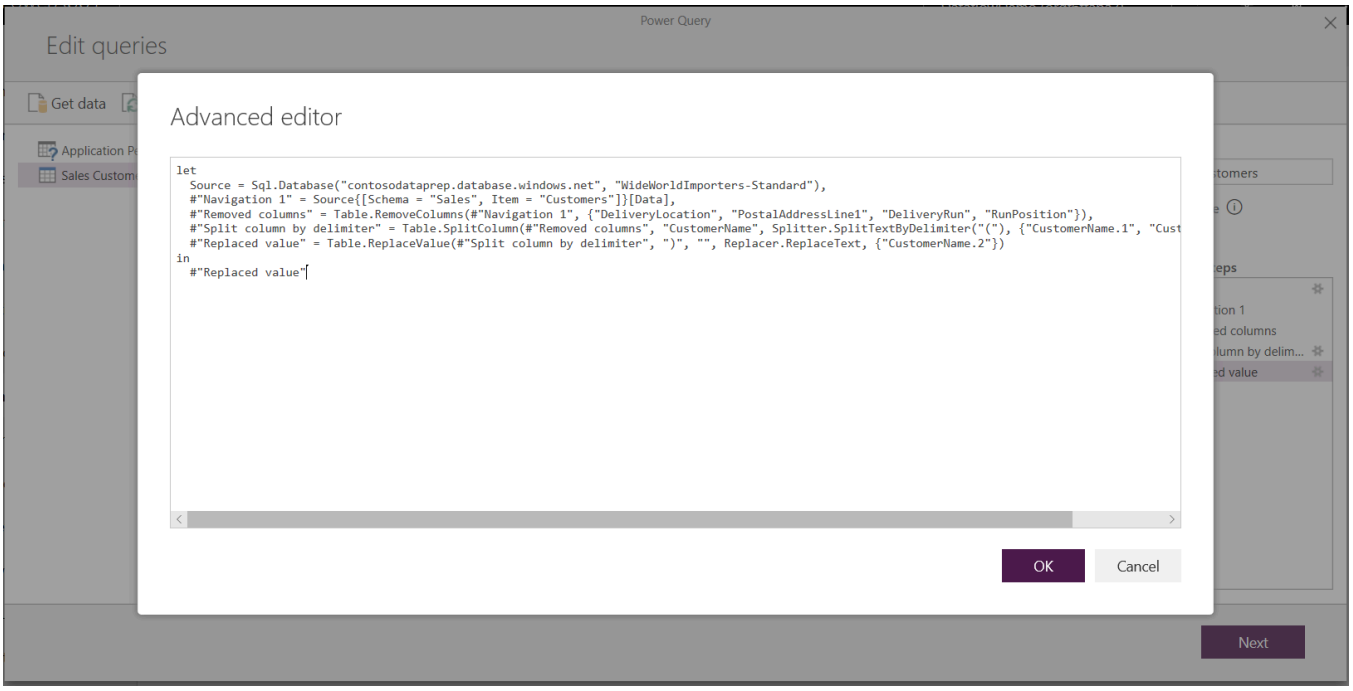


Now that you've selected the data to use in the table, you can use the dataflow editor to shape or transform that data into the format necessary for use in your dataflow.

Use the dataflow editor to shape or transform data

You can shape your data selection into a form that works best for your table using a Power Query editing experience, similar to the Power Query Editor in Power BI Desktop. To learn more about Power Query, see Query overview in Power BI Desktop.

If you want to see the code that Query Editor is creating with each step, or if you want to create your own shaping code, you can use the advanced editor.

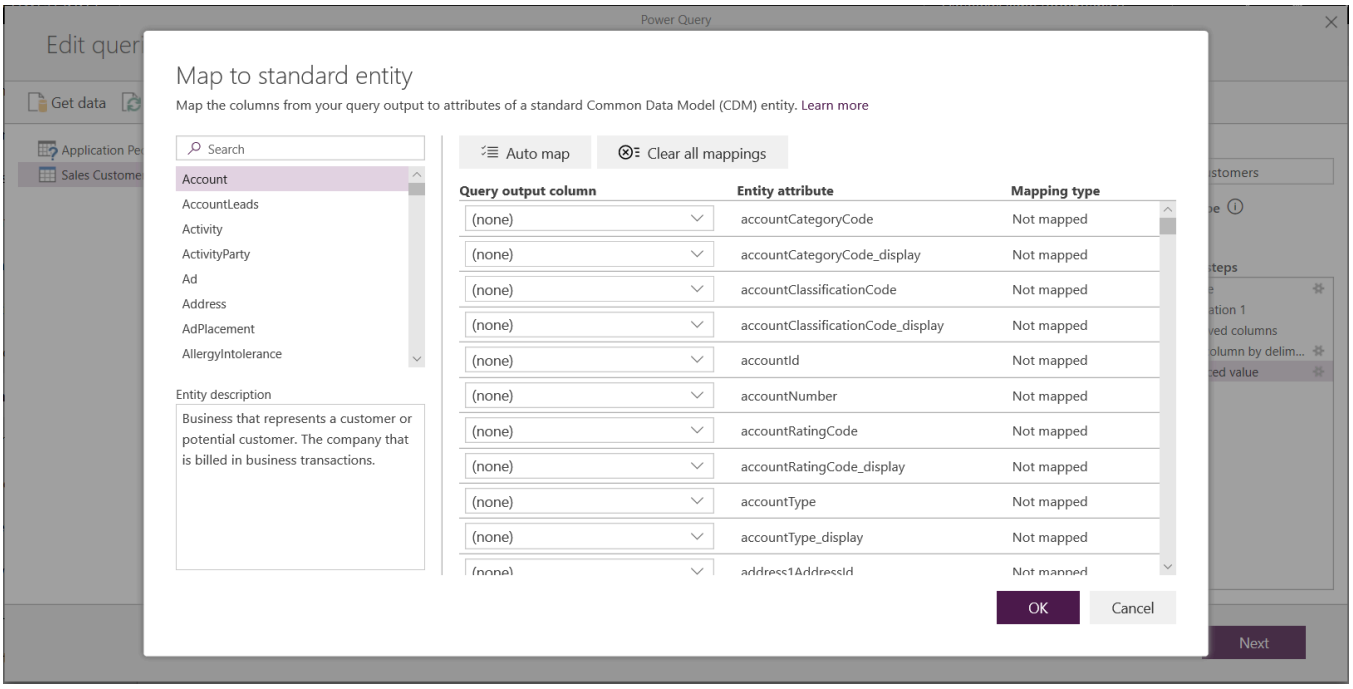


Dataflows and the Common Data Model

Dataflows tables include new tools to easily map your business data to the Common Data Model, enrich it with Microsoft and non-Microsoft data, and gain simplified access to machine learning. These new capabilities can be leveraged to provide intelligent and actionable insights into your business data. Once you've completed any transformations in the edit queries step described below, you can map columns from your data source tables to standard table columns as defined by the Common Data Model. Standard tables have a known schema defined by the Common Data Model.

For more information about this approach, and about the Common Data Model, see The Common Data Model.

To leverage the Common Data Model with your dataflow, select the **Map to Standard** transformation in the **Edit Queries** dialog. In the **Map tables** screen that appears, select the standard table that you want to map.



When you map a source column to a standard column, the following occurs:

1. The source column takes on the standard column name (the column is renamed if the names are different).

2. The source column gets the standard column data type.

To keep the Common Data Model standard table, all standard columns that aren't mapped get *Null* values.

All source columns that aren't mapped remain as is to ensure that the result of the mapping is a standard table with custom columns.

Once you've completed your selections and your table and its data settings are complete, you're ready for the next step, which is selecting the refresh frequency of your dataflow.

Set the refresh frequency

Once your tables have been defined, you should schedule the refresh frequency for each of your connected data sources.

Dataflows use a data refresh process to keep data up to date. In the **Power Platform Dataflow authoring tool**, you can choose to refresh your dataflow manually or automatically on a scheduled interval of your choice.

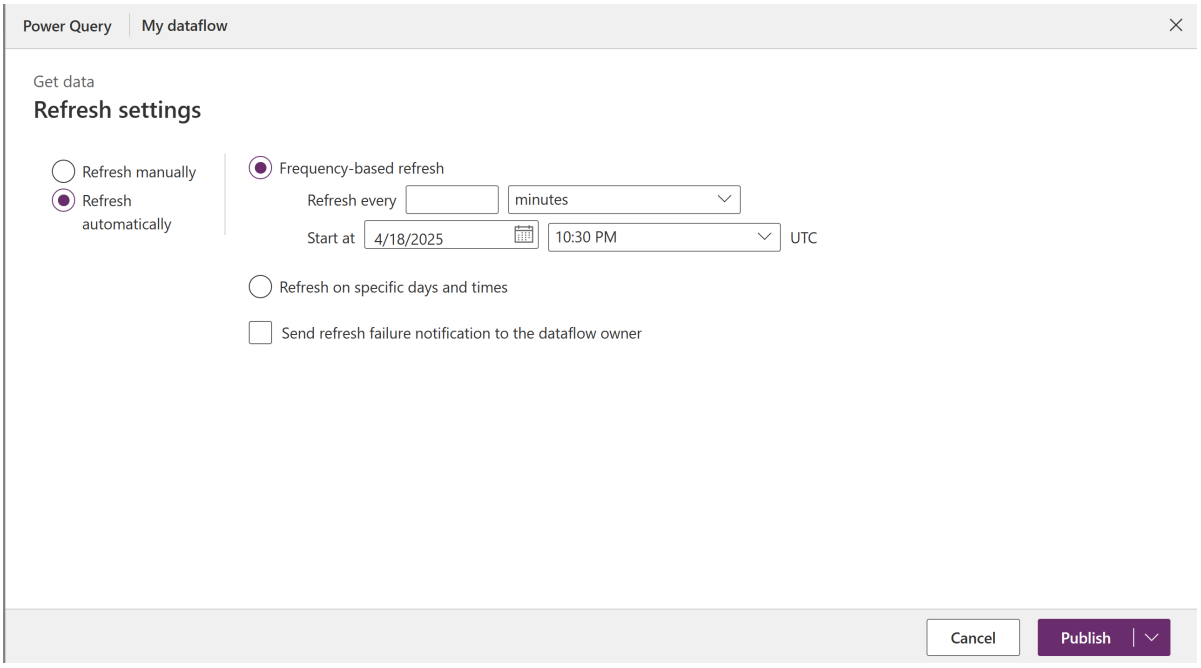
Schedule a refresh automatically

1. Select **Refresh automatically**.

2. Enter the dataflow frequency:

    - **Frequency-based refresh**. Set how often in 30-minute increments, start date, and time in UTC.

    - **Refresh on specific days and times**. Choose time zone, frequency (daily or weekly) and time of day in 30-minute increments.



3. Select **Publish.**

Some organizations might want to use their own storage for creation and management of dataflows. You can integrate dataflows with Azure Data Lake Storage Gen2 if you follow the requirements to set up the storage account properly. More information: Connect Azure Data Lake Storage Gen2 for dataflow storage

Troubleshooting data connections

There might be occasions when connecting to data sources for dataflows runs into issues. This section provides troubleshooting tips when issues occur.

- **Salesforce connector.** Using a trial account for Salesforce with dataflows results in a connection failure with no information provided. To resolve this, use a production Salesforce account or a developer account for testing.

- **SharePoint connector.** Make sure you supply the root address of the SharePoint site, without any subfolders or documents. For example, use a link similar to `https://microsoft.sharepoint.com/teams/ObjectModel`.

- **JSON File connector.** Currently you can connect to a JSON file using basic authentication only. For example, a URL similar to `https://XXXXX.blob.core.windows.net/path/file.json?sv=2019-01-01&si=something&sr=c&sig=123456abcdefg` is currently not supported.

- **Azure Synapse Analytics.** Dataflows don't currently support Microsoft Entra authentication for Azure Synapse Analytics. Use basic authentication for this scenario.

> ⓘ Note
>
> If you use data loss prevention (DLP) policies to block the **HTTP with Microsoft Entra (preauthorized)** connector then **SharePoint** and **OData** connectors will fail. The **HTTP with Microsoft Entra (preauthorized)** connector needs to be allowed in DLP policies for **SharePoint** and **OData** connectors to work.

Troubleshoot the error: Connection to Dataverse failed. Please check the link below on how to fix this issue

Users might receive an error message if the connection they're using for export requires a fix. In this case, the user receives an error message that states **Connection to Dataverse failed. Please check the link below on how to fix this issue**.

To resolve this issue:

1. In Power Apps (make.powerapps.com), select **Connections** from the left navigation pane. If the item isn't in the side panel pane, select ...More and then select the item you want.
2. Locate the **Microsoft Dataverse (legacy)** connection.
3. Select the **Fix connection** link in the **Status** column, and follow the instructions on your screen.

After the fix completes, retry the export.

You receive the error message "There was a problem refreshing the dataflow"

This error occurs when you try to refresh the dataflow while using the Firefox web browser. To work around this issue use a different web browser, such as Microsoft Edge or Google Chrome.

Next steps

The following articles are useful for further information and scenarios when using dataflows:

- Add data to a table in Dataverse
- Using an on-premises data gateway in Power Platform dataflows
- Connect Azure Data Lake Storage Gen2 for dataflow storage
- Use dataflows in solutions

For more information about the Common Data Model:

- Common Data Model - overview
- Learn more about the Common Data Model schema and tables on GitHub

---

Last updated on 01/09/2026

Get insights from your data

Article • 05/31/2024

Microsoft Dataverse is the intelligent data platform that powers Dynamics 365 apps and the Power Platform. All Dynamics 365 data is hosted in Dataverse. The data included data from finance and operations apps and customer engagement apps like Dynamics 365 Sales and Dynamics 365 Customer Service. Dynamics 365 applications can be extended with Power Platform tools like Power Apps, Power Automate, and Power Pages for low-code and pro-code extensions. Data from Power Platform apps is also hosted in Dataverse.

When you understand the data and its intent, Dataverse provides turnkey capabilities in a consistent manner for all your data. Dataverse isn't just a database or storage, it encompasses everything needed to build integrated applications, processes, and in turn provides insights from the data it hosts. Dataverse also provides a comprehensive governance framework for your data including granular role based security and audit and compliance. Because it's a low-code data platform, developers can spend more time focusing on business needs through the configuration of the system and less time building infrastructure.

There are several approaches to getting insights from your data hosted by Dataverse:

- Quick view is a data exploration feature built into Power Apps. When you build a low-code app, either by extending Dynamics 365 or with your own data, quick view is enabled by default. There's no need to create reports to inquire and explore data. You can save these reports as required and later extend them in Power BI Desktop - the authoring tool for Power BI.

- Power BI with Dataverse data. As a low-code maker or a business analyst, you can use Power BI with Dataverse data to create rich reports and embed these reports within pages and forms in Power Apps.

- Link to Fabric feature provides a no-copy, no-ETL, fully managed software as a service (SaaS) integration. The link to Microsoft Fabric feature enables data engineers and business analysts to use all the tools of Microsoft Fabric to combine Dataverse data with their own data.

- Azure Synapse Link enables continuous export of data to your own Azure data lake and enables IT admins and data integration specialists to create data pipelines.

Data export service, bring your own device (BYOD), and Export to Data Lake were features introduced in Dynamics 365 apps to export data for analytics and data

integration scenarios. These services enabled IT admins and specialists to export data into external databases or data lakes and build data integration pipelines. While we have improved these services over the years with updates, as part of unification of Dynamics 365 with the Power Platform, we rearchitected the same capabilities of these disparate services into simpler, unified experiences built into Power Apps (make.powerapps.com). When you transition to Fabric link or upgrading to Azure Synapse Link, these features provide you with an easy ramp to benefit from AI and Copilot investments in Dataverse and Fabric. Go to the transition from legacy integration services article for more information about how you can transition to Dataverse insights.

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Power BI with Microsoft Dataverse data

Article • 05/06/2022

Power BI is a comprehensive collection of services and tools that you use to visualize your business data. The following connector, templates, and features are available that make it easy to visualize and analyze Microsoft Dataverse data or the Dynamics 365 Sales and Dynamics 365 Customer Service apps data with Power BI.

- **Dataverse connector**: Dataverse allows you to connect directly to your data using Power BI Desktop to create reports and publish them to Power BI. From Power BI, reports can be used in dashboards, shared to other users and accessed cross platform on Power BI mobile apps. This connector is the most recent version and uses the tabular data stream (TDS) protocol. More information: Create a Power BI report
- **Common Data Service (legacy) connector**: This is the earlier version of the connector. Use this connector when the query results will be greater than 80 MB. This version also supports paging of the query results and building reports that use the image data type.
- **Dataflow templates**: Dataflows help organizations unify data from disparate sources and prepare it for consumption. You can easily create dataflows using familiar, self-service tools to ingest, transform, integrate, and enrich big data. Once you create a dataflow in Power Apps, you can get data from it using the Common Data Service connector or Power BI Desktop Dataflow connector. More information: Create and use dataflows in Power Apps
- **Power BI template apps**: Power BI template apps are integrated packages of pre-built Power BI dashboards and reports. Using Power BI template apps with Dynamics 365 Sales provides a convenient, powerful, and quick way to access and analyze your sales data. For more information, see these articles:
  - Sales analytics
  - Service analytics
- **Embed Power BI reports on a form**: You can use Power BI reports in Power Apps model-driven apps to bring rich reporting and analytics to your system forms and empower your users to accomplish more. More information: Embed a Power BI report in a model-driven system form
- **Embed Power BI reports on a system dashboard**: Use can add Power BI reports to the dashboards and include in the solution with Environment Variables for ALM. More information: Create or edit a Power BI embedded system dashboard

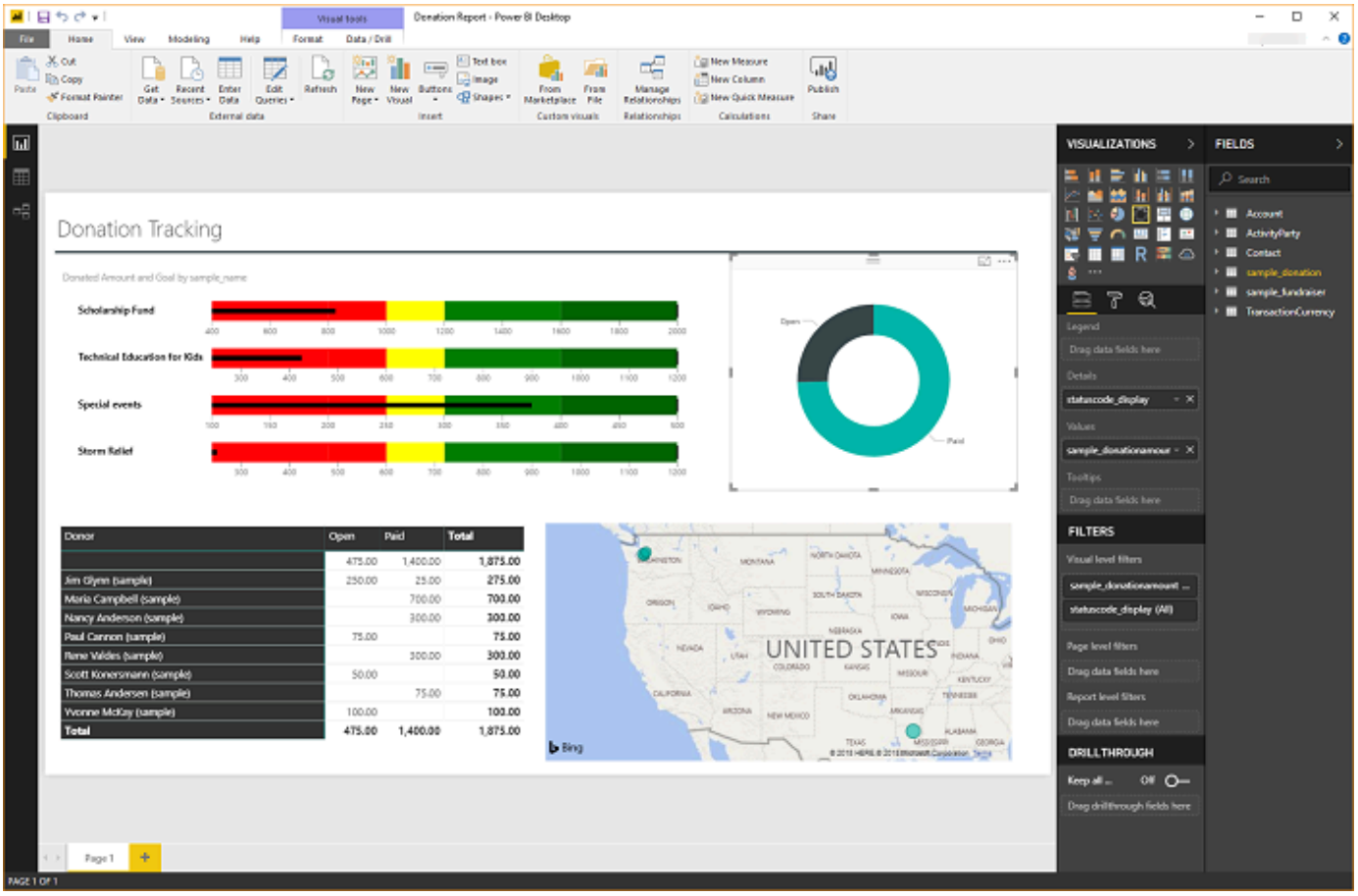
- **Tabular Data Stream (TDS) endpoint**: Use the TDS endpoint for Dataverse to view table data in Power BI Desktop. More information: View table data in Power BI Desktop

See also

Use Power BI

Create a Power BI report using data from Dataverse

Microsoft Dataverse allows you to connect directly to your data using Power BI Desktop to create reports and publish them to Power BI. From Power BI, reports can be used in dashboards, shared to other users, and accessed cross platform on Power BI mobile apps.



Prerequisites

To use Power BI with Dataverse, you need the following items:

- Download and install Power BI Desktop, which is a free application that runs on your local computer. You can download Power BI desktop here.
- A Power Platform environment with the following privileges:
  - To access data in a table, you must have read privileges to the table.
  - To modify a table in Power Apps (make.powerapps.com), you must have a security role that includes maker privileges, such as system customizer or environment maker.
- You must have the appropriate Power BI license to build and share Power BI reports.

- To use the **Dataverse** connector, the **Enable TDS endpoint** setting must be enabled in your environment. More information: Manage feature settings

> **Note**
>
> Most proxy servers don't handle the Tabular Data Stream (TDS) protocol data used by the Dataverse connector for Power BI.

Connect to Dataverse using a connector

Dataverse connector

1. Open **Power BI Desktop**. Select **Get data from other sources**.
2. In the **Get Data** list, select **Dataverse**, and then select **Connect**.
3. If you're prompted, select or enter your user credentials, and then select **Connect**.
4. The list of Power Platform environments with Dataverse appears. In the list of environments, expand the environment you want, select the tables you want, and then select **Load**.
5. Select from the following **Data Connectivity** mode options:

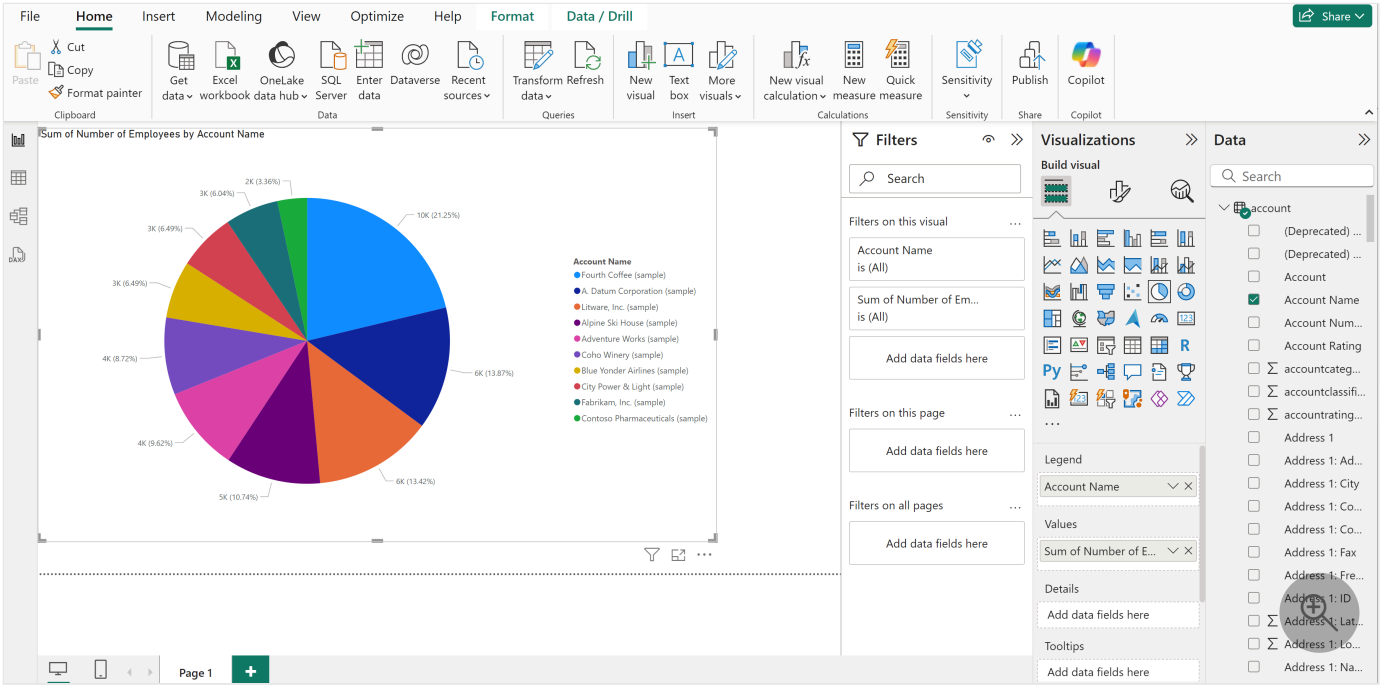
    - **Import**: We recommend that you import data to Power BI wherever possible. With this mode, data is cached in the Power BI service and imported on a scheduled interval.
    - **DirectQuery**: Connects directly to the data in Dataverse. Use this mode for real-time data retrieval. This mode can also more strictly enforce the Dataverse security model. More information: DirectQuery model guidance in Power BI Desktop.

6. Select **OK**. You might be prompted to sign in using the same credentials you use to connect to Power Apps and Dataverse. Select **Connect**.

> **Important**
>
> To use the Dataverse connector, TCP ports 1433 and/or 5558 need to be open to connect. If only port 5558 is enabled, you must append that port number to the environment URL, such as *yourenvironmentid.crm.dynamics.com,5558*.

Build reports using Dataverse tables

After loading the tables by using a connector, you can begin building reports, or repeat the previous steps to add additional tables. For example, in the **Columns** pane, select the **name** column and then select the **numberofemployees** column. In the **Visualizations** pane, select **Pie chart**. These selections add a new visualization to your report canvas.



Special column types

Choice columns

Choice columns are used in tables to provide a drop-down list of items to a user to make a single selection in apps and flows. When using the Dataverse connector, choice columns are presented as two columns to show both the unique value, and the display item value.

For example, if you had a choice column on your table called `approvalstatus`, you would see two columns in Power BI:

- `approvalstatus` - This column shows a unique integer value for each item in your choice. `approvalstatus` can help when you apply filters so the filters won't be impacted when you make future changes to the display name.
- `approvalstatusname` - This column shows the friendly display name of the item and is most commonly used when presenting the option in a table or chart.

Expand table

`approvalstatus`	`approvalstatusname`
1	Submitted
2	In Review
3	Approved
4	Rejected

Performance impact and choice name columns

When retrieving the label name for a choice column, Dataverse makes a join with the internal `stringmap` table (where localized labels are stored). This is executed for each label/name column. Note that, this join and doing filters against the label name column, rather than the value column, can significantly impact report query performance.

Choices columns

Choices are similar to choice columns with the difference being that users can select multiple items from the list. Choices aren't currently fully supported with the Dataverse connector. When you use the Dataverse connector with choices columns, you only receive the integer values, which are comma separated. The item label name columns aren't returned. For more information about the Dataverse data types not supported with the Dataverse connector, see Supported operations and data types.

Lookups

Lookup columns use a many-to-one (N:1) table relationship between the table you're working with and the target row type defined for the lookup. Lookups are presented in Power BI Desktop as two columns, `lookupid` and `lookupid-name`.

Navigating relationships

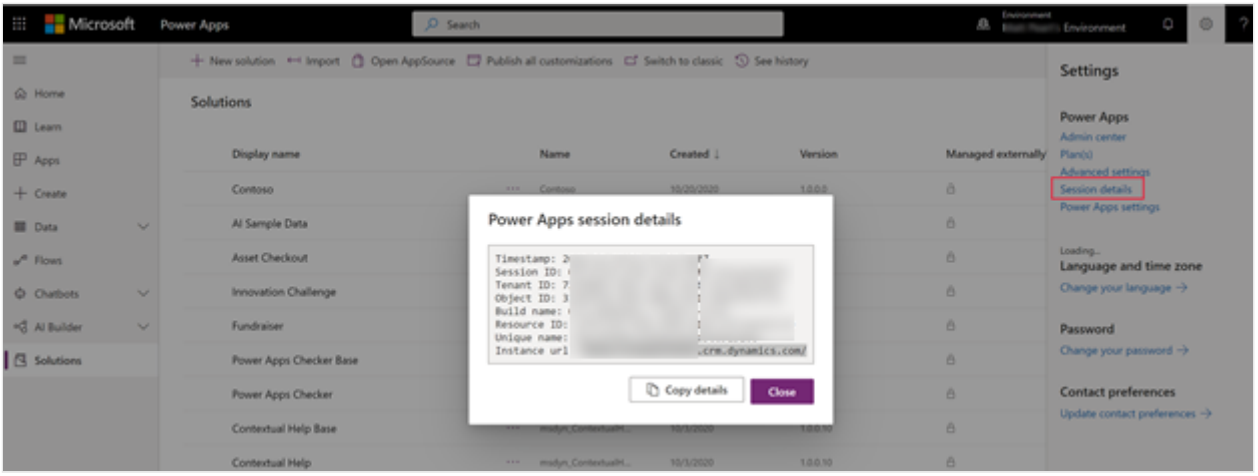
Relationships in Dataverse require you to create a relationship within Power BI desktop between the two tables using a GUID column, this is a system-generated unique identifier that ensures relationships are created for the create rows where ambiguity or duplication might exist with other columns. You can read more about managing relationships in Power BI desktop here.

While some relationships might be automatically created, you can still review and ensure the correct relationships are established when creating your report:

- The lookup column on the table contains the GUID of the row in the related table.
- The related table has a column in the format "[tableName]ID" that contains the GUID, for example `Accountid` or `MyCustomtableid`
- Using the Power BI desktop Manage Relationships feature, you would create a new relationship between your lookup column, and the ID column on the related table.

Find your environment URL

1. Open Power Apps, select the environment you're going to connect to, select **Settings** in the top-right corner, and then select **Session details**.



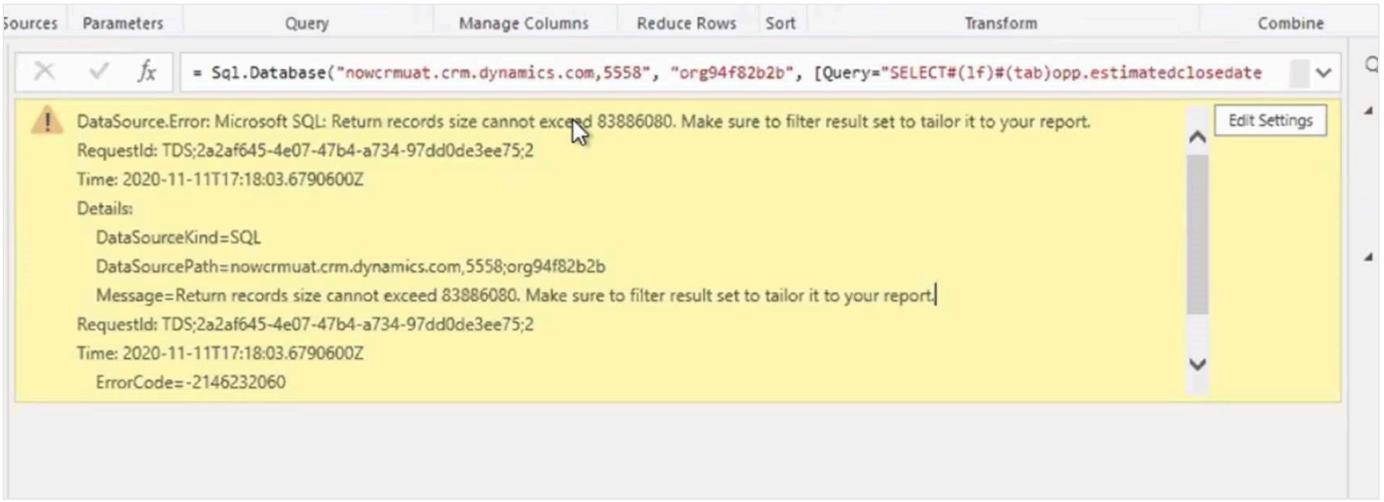
2. In the **Power Apps session details** dialog box, select **Copy Details**.

3. Paste the session details somewhere such as Notepad so that you can copy only the **Instance url**. The instance URL is the unique URL for your environment. The URL is in the format: `https://yourenvironmentid.crm.dynamics.com/`. Keep this somewhere handy so you can use it when creating your Power BI reports.

Troubleshooting

Error message: Return records size can't exceed 83886080

This error can occur with the Dataverse connector when you run or design a report with a result set greater than 80 MB. TDS has a result set size limit of 80 MB.



To work around this limit, optimize the query adding filters and dropping columns so that the query returns less data.

Workaround for very large number of lookups or choice columns

If the error message occurs in Power BI when you try to connect to a table with a very large number of lookups or choice columns, the following manual workaround might allow you to connect to the table. The account, contact, and opportunity table might encounter this issue when they're extensively customized with additional lookups or choice columns.

Manually connect to the table in a Power BI report:

1. In Power BI desktop with the report loaded, select **Transform Data** to load Power Query.
2. Select **New Source** > **Blank Query**.
3. **Name** your query.
4. Select **Advanced Editor** on the **Home** tab of Power BI Desktop.
5. Replace the query text with this query text.

```
let
    Source = CommonDataService.Database("<myenvironment.crmX>.dynamics.com"),
    dbo_contact = Source{[Schema="dbo",Item="contact"]}[Data],
    #"selectedcolumns" = Table.SelectColumns(dbo_contact,{"fullname",
"emailaddress1"})
in
    #"selectedcolumns"
```

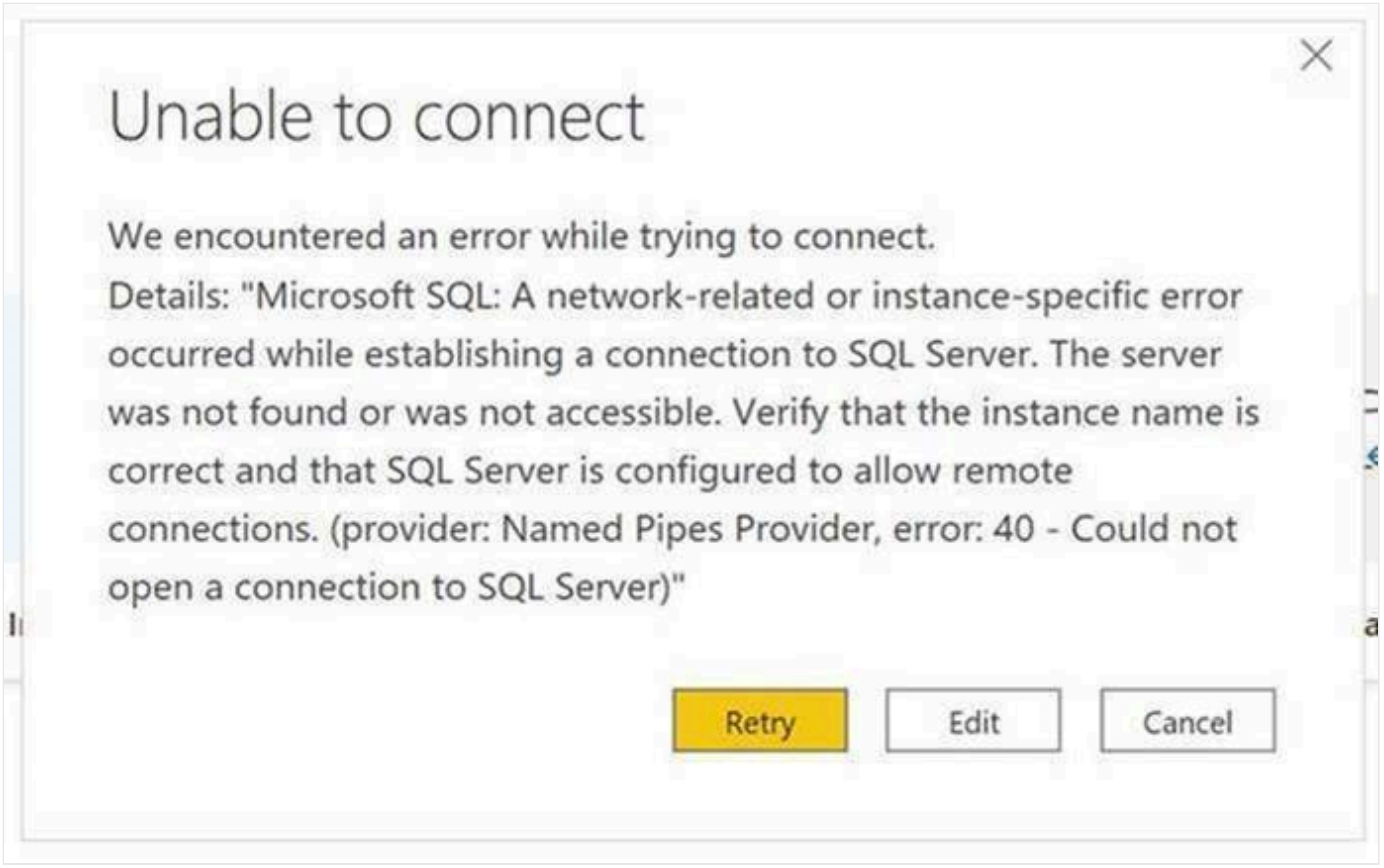
6. Replace *myenvironment.crmX* in the query text with your environment domain value, such as *contoso.crm4*.

7. Select **Done**.
8. Select **Choose columns** to add any additional needed columns.
9. Select **Close and Apply** to save model changes.
10. When prompted, select **Direct Query** for the new query.

The query can now be used in the report.

Error message: Unable to connect (provider Named Pipes Provider, error: 40 – Could not open a connection to SQL Server)

When this error message occurs, the connector fails to connect to the TDS endpoint. This can occur when the URL used with the connector includes *https://* and/or the ending */*.



Remove the https:// and ending forward slash so that the URL is in the form *orgname.crm.dynamics.com*.

Troubleshooting connection issues

For information about troubleshooting connection issues when using the TDS endpoint, see Troubleshooting connection problems.

See also

Use composite models in Power BI Desktop

View Dataverse for Teams table data in Power BI Desktop

Last updated on 04/27/2026

View table data in Power BI Desktop

Article • 08/08/2024

Use Power BI Desktop to view tables in Microsoft Dataverse. The table row data that you can access from your environment is read-only. Data access uses the Dataverse security model that is the same used to access table row data using a Power Apps app.

Prerequisites

- Enable the Tabular Data Stream (TDS) endpoint (on by default). More information: Manage feature settings.
- Power BI Desktop. Get it now

View table data

1. Sign into Power Apps, and then select the appropriate environment from the top-right corner.

2. On the left navigation pane select **Tables**. If the item isn't in the side panel pane, select ...More and then select the item you want.

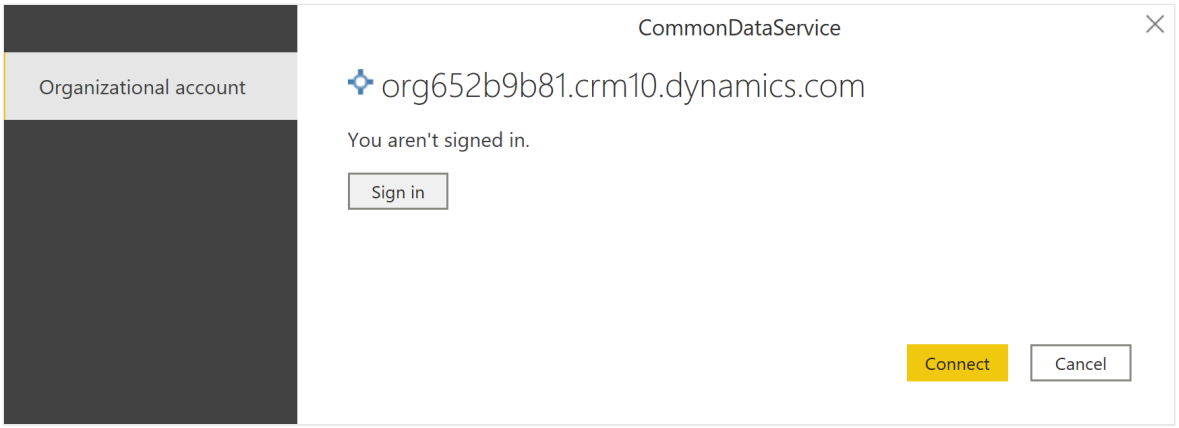
3. Select **Analyze** > **Analyze in Power BI** on the command bar.

   The pbids file for your environment is downloaded to your browser's default download folder.

   > **Note**
   >
   > To be able to download the file, you must have read access to the analysis component (msdyn_analysiscomponent) custom table privilege in Dataverse. Users with the environment maker security role have full access to this privilege.

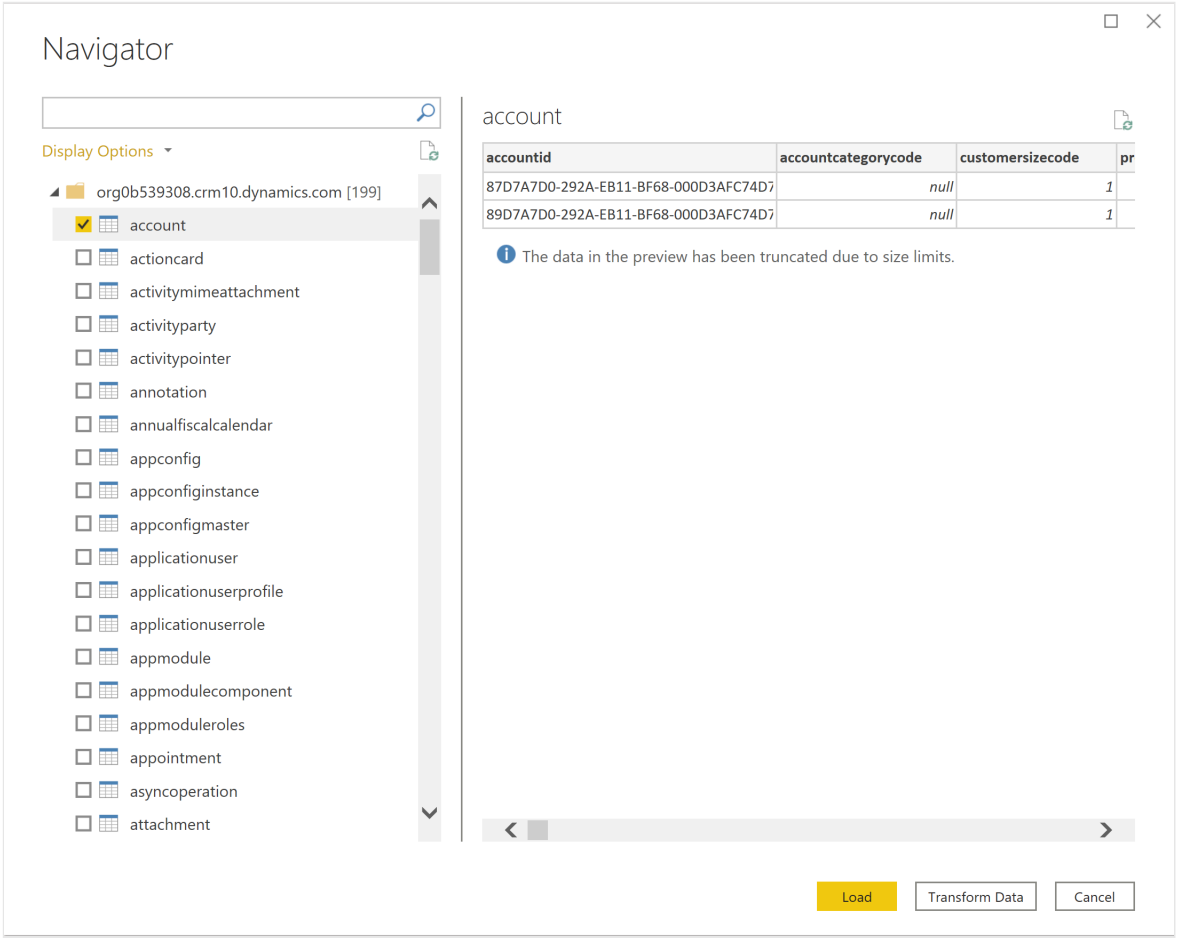
4. Open the .pbids file to access it in Power BI Desktop.

5. The pbids file is loaded in Power BI Desktop. In the dialog box, select **Organizational account**, select **Sign in**, and then in the browser window that appears select or enter your credentials.



6. In the dialog box in Power BI Desktop, select **Connect**.

   The environment appears in the Power BI Desktop **Navigator** window. Expand it to view the tables available to analyze. Select a table to preview its data.



7. After you're finished selecting the tables you want to analyze, select **Load** to build a report.

For more information about Power BI Desktop, see Get started with Power BI Desktop.

> **Note**
>
> SQL options, such as a T-SQL queries aren't supported.

> **Tip**
>
> The **Power BI modeling guidance for Power Platform** article provides implementation guidance and best practices on how to create a Power BI data model that connects to Dataverse:
>
> - Performance and security considerations when connecting Power BI to Dataverse.
> - Dataverse connector best practices.
> - Working with large datasets using Azure Synapse Link for Dataverse.
> - Considerations for DirectQuery.

See also

Use SQL to query data
Dataverse SQL
Power Query Dataverse connector

Feedback

Was this page helpful? Yes No

Provide product feedback

Transition from legacy data integration services

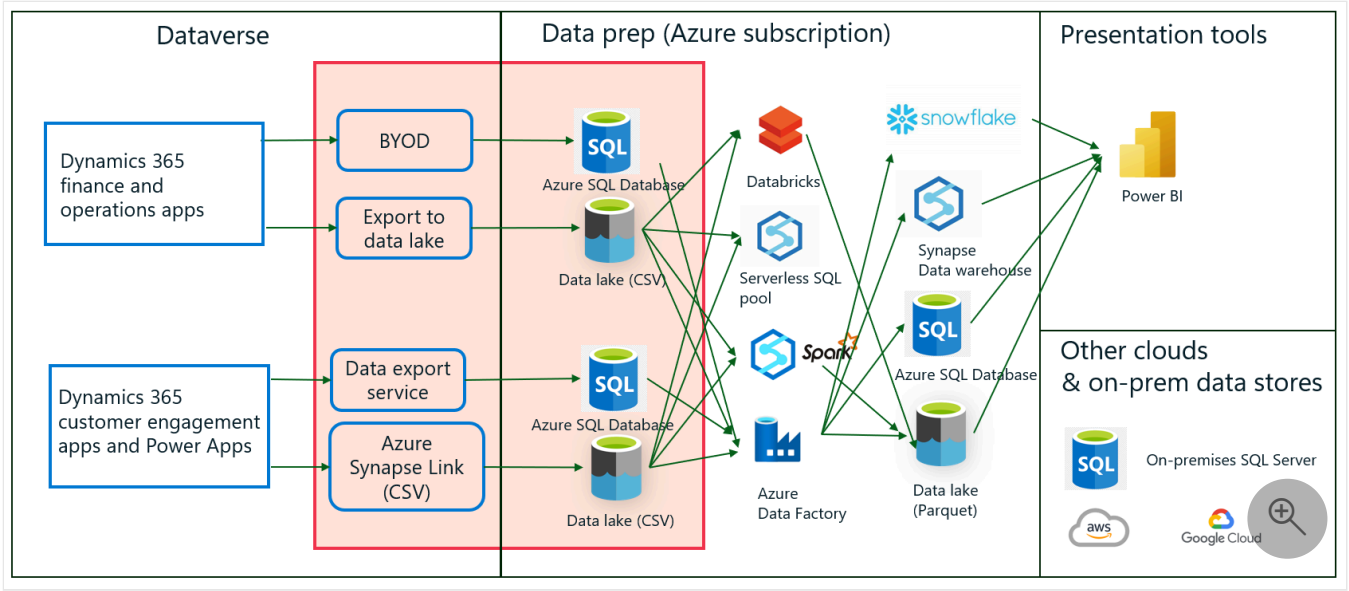
Data export service, bring your own database (BYOD), and Export to Data Lake were features introduced in Dynamics 365 apps to export data for analytics and data integration scenarios. These services enabled IT admins and specialists to export data into external databases or data lakes and build data integration pipelines. While we improved these services over the years with updates as part of unification of Dynamics 365 with the power platform, we rearchitected the same capabilities of these disparate services into simpler, unified experiences built into Power Apps (make.powerapps.com). With a transition to Microsoft Fabric link or upgrade to Azure Synapse Link, the rearchitected services provide you with an easy ramp to benefit from AI and Copilot investments in Microsoft Dataverse and Fabric.

If you're a customer using any of the previous generation services, this article provides guidance on upgrading to the new experiences, benefiting from innovations, as well as reducing end-to-end expenses and effort.

Based on preview customer surveys, we also compiled a high level cost and benefits estimate to help you with the transition. Links to more information and videos, links to join forums and weekly office hour sessions to engage with the product team, Microsoft specialists, and fellow users are also provided here. We strive to enhance these services with community participation.

Before transition

If you're a customer using legacy services BYOD, data export service, or Export to Data lake, you might have a data integration architecture similar to the one shown here. The highlighted box indicates the data pipelines your organization might have built to leverage the data exported from Dynamics 365 and Dataverse. You can use a selection of tools from Microsoft, as well as others, to copy and integrate Dynamics data with your own data. You can also transform and aggregate data by copying into multiple stores – shown in the data prep box of the diagram. You can use Power BI or another tool to visualize data and create actionable insights. You might have pipelines built to export data to an on-premises system and other clouds.



After transition

There are two data integration patterns enabled in Power Apps:

- Azure Synapse Link enables continuous export of data similar to BYOD, Export to Data Lake, or data export service. This option is enabled for IT admins and data integration specialists.
- Link to Fabric feature provides a no-copy, no-extract, transform, and load (ETL), fully managed software as a service (SaaS) integration.

These options are complementary. Here's a detailed comparison of the data integration services.

If you're a finance and operations apps customer currently using BYOD or Export to Data Lake features, by upgrading to Azure Synapse Link or Fabric link, you can benefit from a simplified data integration architecture resulting in reduced operational costs:

- Easy to configure and maintain via Power Apps (make.powerapps.com). Built-in integration with Azure Synapse and Fabric.
- Azure Synapse Link and Fabric link are fully managed services. Each requires minimal ongoing management overhead.
- New services offer the same data shapes as previous services. Your existing downstream integration pipelines can remain as-is.
- Minimal impact to operational workloads, you don’t need to manage workloads and schedule data exports.

Secure, end-to-end data integration pipelines:

- With Fabric link, your data doesn’t leave the Dataverse governance boundary while authorized Fabric users can securely access data that resides within Dataverse.

- Azure Synapse Link service enables you to restrict access to your storage accounts with firewalls while enabling Dataverse to export data with managed identities for Azure, a security feature built into Microsoft Entra.

If you're a Dynamics 365 customer engagement customer using data export service or classic Azure Synapse Link with data exports in CSV format, you can benefit from efficient reporting enabled with the industry standard Delta or Parquet data format:

- Built-in Delta Lake or Parquet conversion option reduces the need to build your own pipelines for analytics and operational reporting.
- Delta Lake or Parquet format enables faster, more responsive queries & reports and scales to larger datasets of any size.
- Data in lake is compressed to 1/3 to 1/8 the original size, resulting in smaller files that reduce data query and carrying costs.

Expand table

Link to Fabric	Azure Synapse Link
No copy, no ETL direct integration with Fabric.	Export data to your own storage account and integrate with Azure Synapse, Fabric, and other tools.
Data stays in Dataverse. Users get secure access in Microsoft Fabric.	Data stays in your own storage. You manage access to users.
All tables chosen by default.	System administrators can choose required tables.
Consumes additional Dataverse storage.	Consumes your own storage and other compute and integration tools.

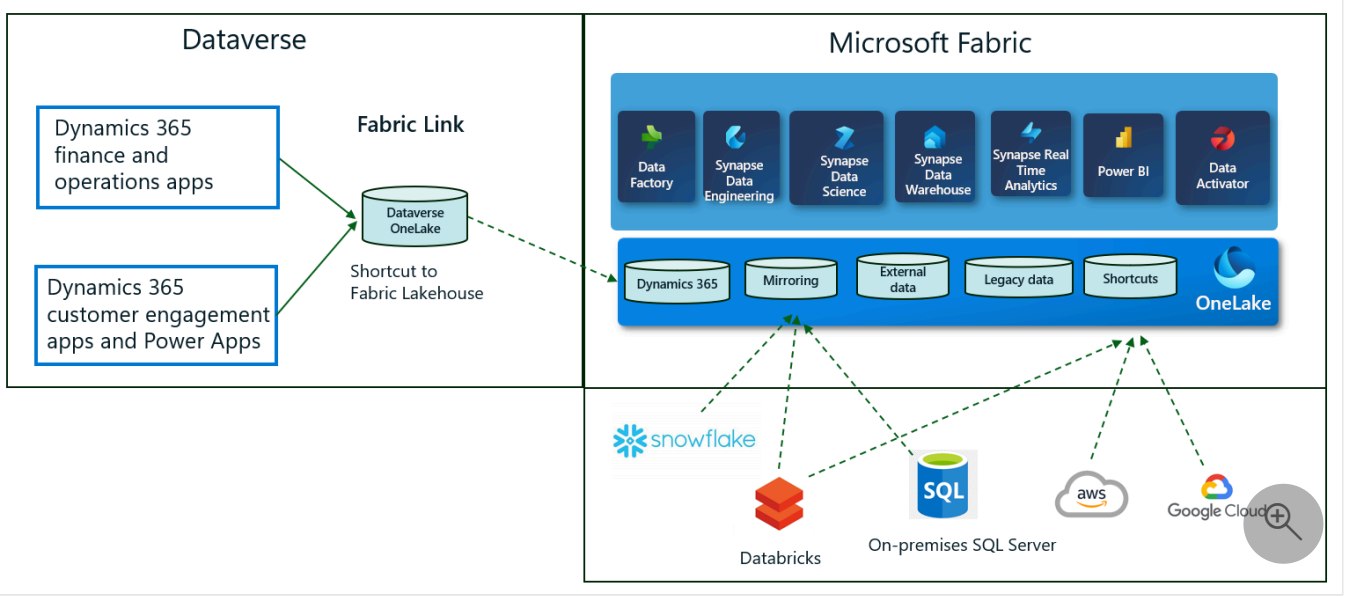
Which option should I use?

If your organization is already using Fabric or planning to transition in the coming months, we recommend using the Fabric link feature. You can continue to use the Azure Synapse Link service if your immediate focus is to upgrade from your current services.

Simplification with Fabric link

If you're already consuming data using Power BI, using a data warehouse, or using dataflows and notebooks to transform data, the link to Fabric feature provides immediate value. You can simplify your data integration architecture by removing the need to have your own storage account or Synapse services for Dataverse data. Instead of paying for Azure resources like storage and compute, you pay for the increase in Dataverse storage. Compute charges such as

near real-time data updates and management overhead is also factored into Dataverse storage. Fabric link option is like having a near real time read-only replica of your data optimized for insights.



Query this replica using T-SQL, Apache Spark, Python as well as other workloads in Fabric. You can also access this data using any tool that can consume T-SQL or can consume data from Azure Data Lake Storage.

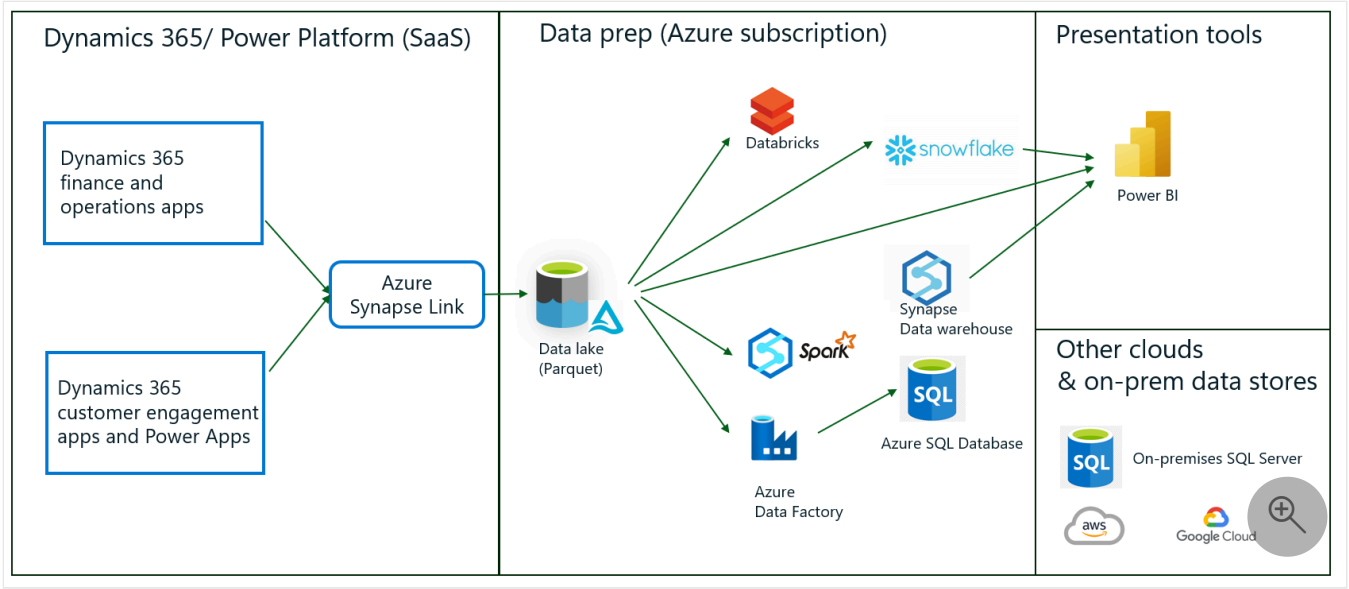
As a Dynamics or Power Apps customer, you get a Dataverse storage quota based on the number of licenses you purchased. The Fabric link feature uses this database quota. You can buy more storage add-ons if the data volume exceeds your quota.

You can continue to retain Azure platform as a service (PaaS) services, like Databricks and SQL Database, in your own subscription. Recently announced Fabric features like data mirroring and shortcuts can help you further simplify your data integration.

Examples of cost reductions achieved with the simplicity derived from Fabric integration feature are described here.

Upgrading to Azure Synapse Link

By upgrading to Azure Synapse Link and enabling Delta or Parquet conversion, you can eliminate Dataverse data prep pipelines in your solution. Azure Synapse Link service exports the same data shapes into your storage account in a more performant Delta or Parquet format. You can continue to use existing tools and Azure services like storage and Azure Synapse Analytics query with minimal disruptions to your production environments.

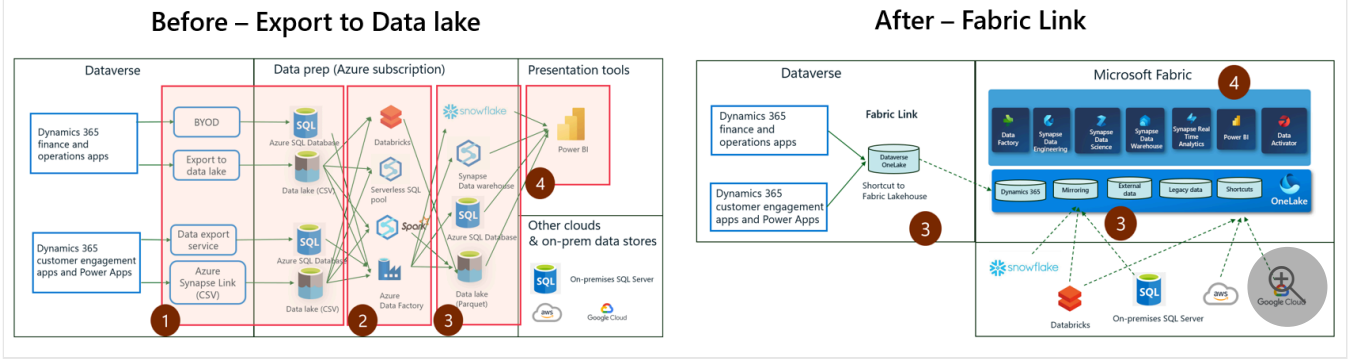


Understanding benefits and cost reductions

Simplicity achieved with Fabric link and Azure Synapse Link yields reductions in end-to-end costs. Consider the following examples that are based on preview customer experiences.

Example 1: Transition from BYOD or Export to Data Lake to Fabric link

Consider the case where you transition to Fabric link from Export to Data Lake.



As indicated in the before and after diagrams in this article, customer retired export to data lake service (1) as well as staging data stores (2) with Fabric link. For operational insights, (4), they consumed data in OneLake directly in Power BI. Some of the insights require data merge, transformation, and aggregation (3). Instead of using disparate Azure services, they standardized on the same tools built into Fabric.

Innovations in Dataverse and Fabric enable simplifications and cost reductions:

- Dataverse comes with a built-in OneLake store. Operational data from Dynamics 365 and Power Apps are replicated to built-in lake store near real-time (to avoid influence to

operational workloads) and linked securely to Fabric via shortcuts. There's no need to bring Azure storage and secure data that's exported out. Your data doesn't leave the Dataverse governance boundary and authorized users in Fabric can work with data using all Fabric workloads.

- Export to data lake and data export service exports data in CSV format. CSV files aren't suitable for direct consumption due to poor query performance as well as occasional read/write contention issues. The before transition solution uses Azure Data Factory to periodically ingest and convert raw data into an Azure SQL Database or an Azure data warehouse. This layer isn't needed in the after transition solution since Dataverse built-in OneLake data is maintained in Delta or Parquet format – the same open format that is native to Microsoft Fabric. Delta or Parquet format, along with optimizations in Fabric removes the need to maintain additional data stores for caching and improving query performance while eliminating read/ write contention. You can create Power BI reports using Direct Lake over data in OneLake without any additional data stages.
- While operational insights can be performed using the data already available in OneLake, you might have additional data from other systems. This data might need to be combined, reshaped, and aggregated with Dataverse data. Fabric provides an integrated environment, which provides best of breed tools like dataflows, Azure Data Factory, and Spark. Use these tools as opposed to configuring and provisioning tools standalone, where you adopt the tools of your choice. Integrated billing, source control, and security enable simpler management and governance.
- While you can continue to use Power BI service for reporting purposes, Fabric introduces Direct Lake mode reporting, which leverages the in-memory indexes built into Delta or Parquet format therefore removing the need to use Power BI import mode reports.

These innovations yield end-to-end cost savings in addition to the benefits discussed earlier. The tables here outline the line items of costs along with a comparison of before and after solutions. You can use this table as a guideline to estimate expected cost savings.

Expand table

Cost savings line item	Before cost - export to data lake	After cost - Fabric link
Set-up and configuration	Need to use multiple tools. - Pay for multiple software licenses and subscriptions. - System configuration efforts. - Effort to build and validate data pipelines. - Continued governance,	Simpler configuration experience in Power Platform. Purchase Fabric capacity and pay for use for all services. You only pay for what you use. In some cases, you're billed by the second. No ETL pipelines needed for Dataverse data.

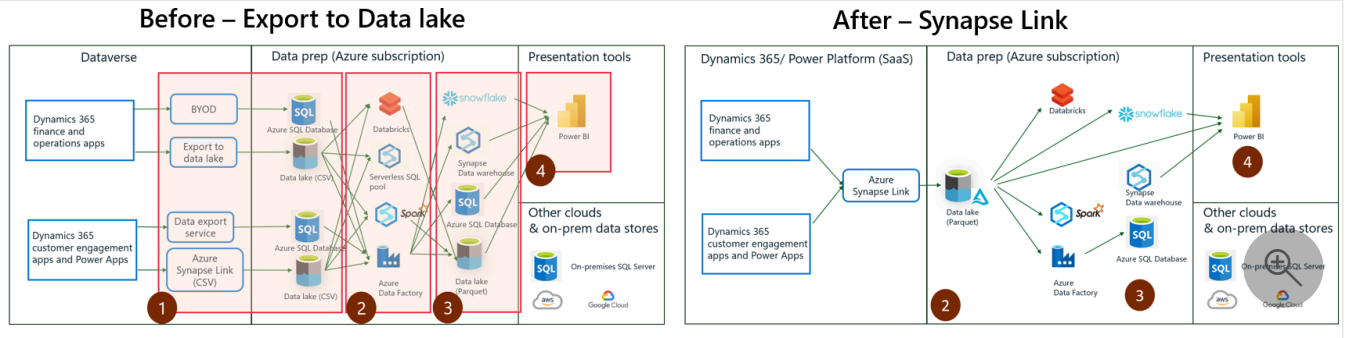
Cost savings line item	Before cost - export to data lake	After cost - Fabric link
	management, and monitoring. - Training of users.	
Data staging (1, 2)	Cost incurred for Azure services. - Azure storage cost including cost of IO. - Synapse Analytics (SQL serverless query). - Azure Data Factory jobs to copy data. - Staging data stores, such as Azure SQL Database.	Cost increase in Dataverse storage For example, if you sync 500 GB of data from Dynamics 365, Dataverse storage could increase by around 100 GB (assuming five to eight times data compression).
Operating Costs - data prep (3)	- Azure storage staging area. - Data pipelines. - Data ingestion into Azure SQL or data warehouse. - Development of maintenance data.	Spending shifts to Fabric where you pay for consumption with a shared capacity.
Operating costs – reporting (4) - Power BI datasets. - Reporting.	Synapse analytics (SQL query). Power BI capacity and storage for import mode reports.	As your data is compressed (for example 1/3 to 1/6 original size), your reporting and query costs reduce accordingly. New features, like Direct Lake reporting, reduce consumption of Power BI storage consumption.

> **Note**
>
> These estimates are provided to enable estimating the spend after transition. While these estimates are based on experience from preview customers, actual costs incurred in your environment as well as data compression can vary depending on volume and composition of data.

Example 2: Upgrade to Azure Synapse Link

Upgrading to Azure Synapse Link is an option to consider if you're not currently using Fabric – or not planning to transition in the coming months.

As indicated in the before and after diagrams here, customer retired Export to Data Lake service (1) as well as staging data stores (2) with Azure Synapse Link. Data stores used for data staging, such as intermediate Azure SQL Databases, are retired as Dataverse data is saved in Delta or Parquet format – a more performant and compressed data format. Downstream pipelines used for data merge, transformation, and aggregation, (3), as well as presentation tools (4), such as Power BI, remain unchanged.



Azure Synapse Link provides an easy upgrade path for customers looking to extend their existing investments with minimal changes. As we discuss here, innovations in Dataverse enable easy transition from finance and operations apps services like Export to Data Lake and BYOD.

1. Azure Synapse Link enables easy configuration of Azure storage and Synapse services within the Power Apps (make.powerapps.com). Several limitations of Export to Data Lake are removed:
   a. A uniform experience enables choosing data from all Dynamics 365 apps including both finance and operations and customer engagement apps. You can maintain a single service to export data from all Dynamics 365 and Power Apps.
   b. Table limitations present with Export to Data Lake go away. You can choose up to 1,000 tables for each Azure Synapse Link profile within a single storage account. You can create multiple profiles (export pipelines) each with up to 1,000 tables.
   c. Azure Synapse Link provides built-in support for using firewall restricted storage accounts to export data.
   d. No need to configure Azure resources. Azure Synapse Link provisions and configures Azure resources on your behalf.
   e. Azure Synapse Link exports data in the same format as Export to Data Lake or BYOD enabling you to retain existing pipelines.

2. Both Export to Data Lake and data export service export data in CSV format. CSV files aren't suited for direct consumption due to poor query performance as well as occasional read/write contention issues. Before transition uses Azure Data Factory to periodically ingest and convert raw data into a SQL Azure Database or an Azure data warehouse. This layer isn't needed in the 'after transition since Azure Synapse Link exports data in Delta or

Parquet, which removes the need to maintain additional data stores for caching and improving query performance while eliminating read/write contention.

3. Continue to use your existing data pipelines for combining, reshaping, and aggregating additional data from other systems.

4. Continue to use Power BI service for reporting purposes. Fabric introduces Direct Lake mode reporting that leverages the in-memory indexes built into Delta or Parquet format, which removes the need to use Power BI import mode reports.

These innovations yield end-to-end cost savings in addition to the benefits discussed earlier. The tables here outline the line items of costs along with a comparison of before and after solutions. Use the table as a guideline to estimate expected cost savings.

Expand table

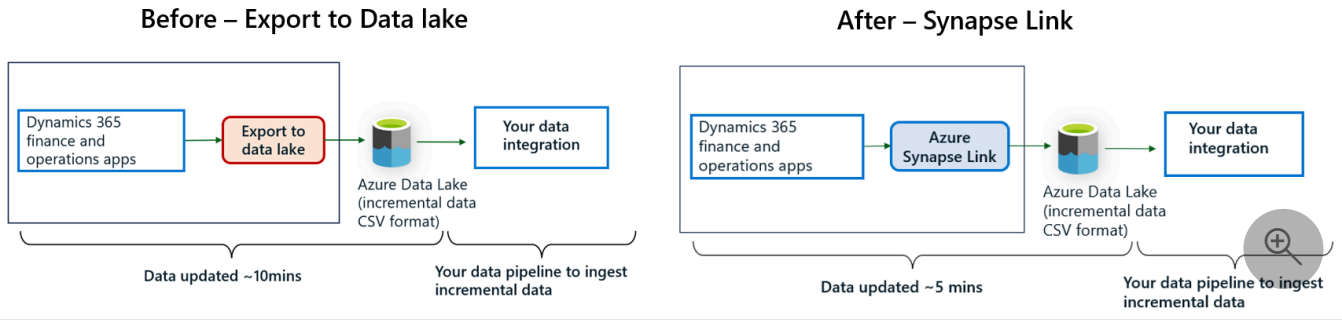
Cost savings line item	Before cost - Export to Data Lake	After cost - Azure Synapse Link
Set-up and configuration	Must use multiple tools. - Pay for multiple software licenses and subscriptions. - System configuration efforts. - Effort to build and validate data pipelines. - Continued governance, management, and monitoring. - Training of users.	Simpler configuration experience in Power Platform. No ETL pipelines needed for Dataverse data.
Data staging (1, 2)	Cost incurred for Azure services. - Azure storage cost including cost of IO. - Synapse Analytics (SQL serverless query). - Azure Data Factory jobs to copy data. - Staging data stores, such as Azure SQL Database.	Azure Synapse Link requires you to provide a Spark pool to convert data to Parquet format. Depending on the frequency of data sync, as well as the volume of data changes, Spark pool costs can vary. Small to medium data changes (per month): - $600 to $2,000 for hourly refresh. - $1,200 to $4,100 for 15-minute refresh. Medium to large data changes (per month): - $1,200 to $2,500 for hourly refresh. - $2,500 to $8,300 for 15-minute refresh.
Operating costs - data prep (3)	- Azure storage staging area. - Data pipelines. - Data ingestion into Azure SQL Database or data	Same costs as indicated above, however, you might not need to aggregate Dataverse data due to Parquet conversion.

Cost savings line item	Before cost - Export to Data Lake	After cost - Azure Synapse Link
	warehouse. - Development and maintenance data.	
Operating costs - reporting (4): - Power BI datasets. - Reporting.	Synapse Analytics (SQL query). Power BI capacity.	As your data is compressed, for example 1/3 to 1/6 original size, your reporting and query costs reduce accordingly.

> **Important**
>
> These estimates are provided to enable estimating the spend after transition. While these estimates are based on experience from preview customers, actual costs incurred in your environment as well as data compression can vary depending on volume and composition of data.

Example 3: Incrementally ingesting data to a data warehouse

If you're currently consuming incremental data from Export to Data Lake feature to populate a downstream data pipeline, you can continue to use the same pipeline. As shown here, Azure Synapse Link service can export incremental data changes in the same format as changes feed into Export to Data Lake.



Azure Synapse Link service provides several enhancements over Export to Data Lake for incremental data changes:

- Initial data load is included within change folders. This makes it easy for the same pipeline to consume both the initial load as well as incremental updates.
- Change data isn't deleted in case of a reinitialization of a table.
- System creates a time-stamped folder structure and metadata that helps you read the changes in chronological order. Change data once written, is never updated. This

approach is better suited to process changes using big data ingestion tools like Azure Data Factory.
- You can configure how often you want the change feeds updated in folders with Azure Synapse Link, as low as five minutes.

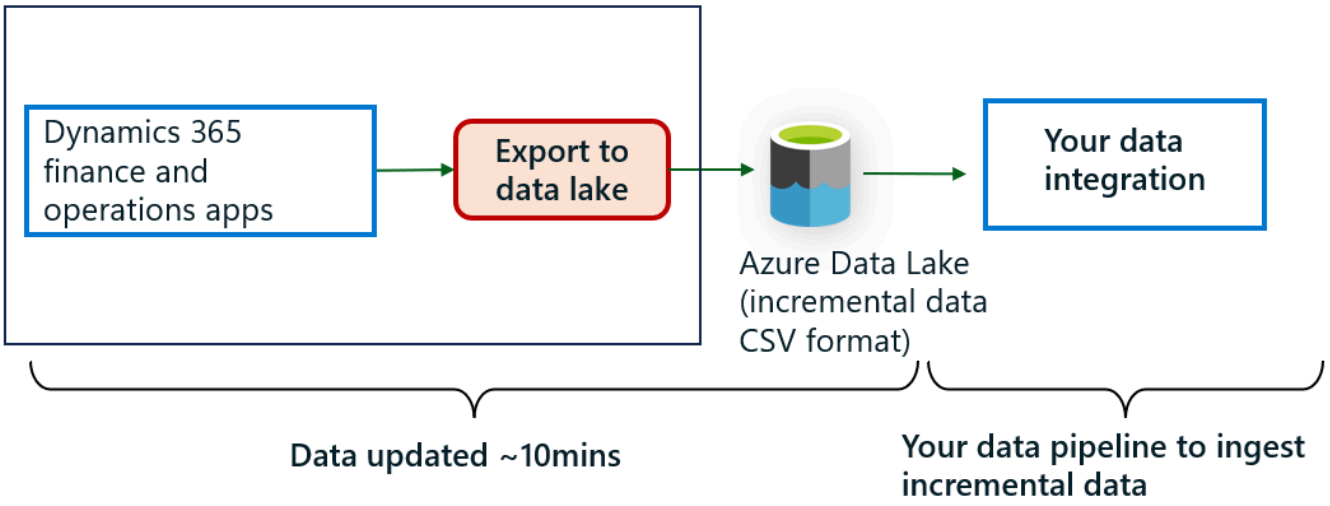
When you transition to Fabric or Azure Synapse Link, the change in cost is minimal for incremental data changes.

Understanding benefits – more real-time reporting

Fabric not only simplifies your data integration architecture, but also reduces the need to copy or replicate data. The underlying data format in Fabric, the industry standard Delta or Parquet format, can be directly consumed by Power BI, dataflows, notebooks, and other workloads eliminating the need for caching and staging areas.

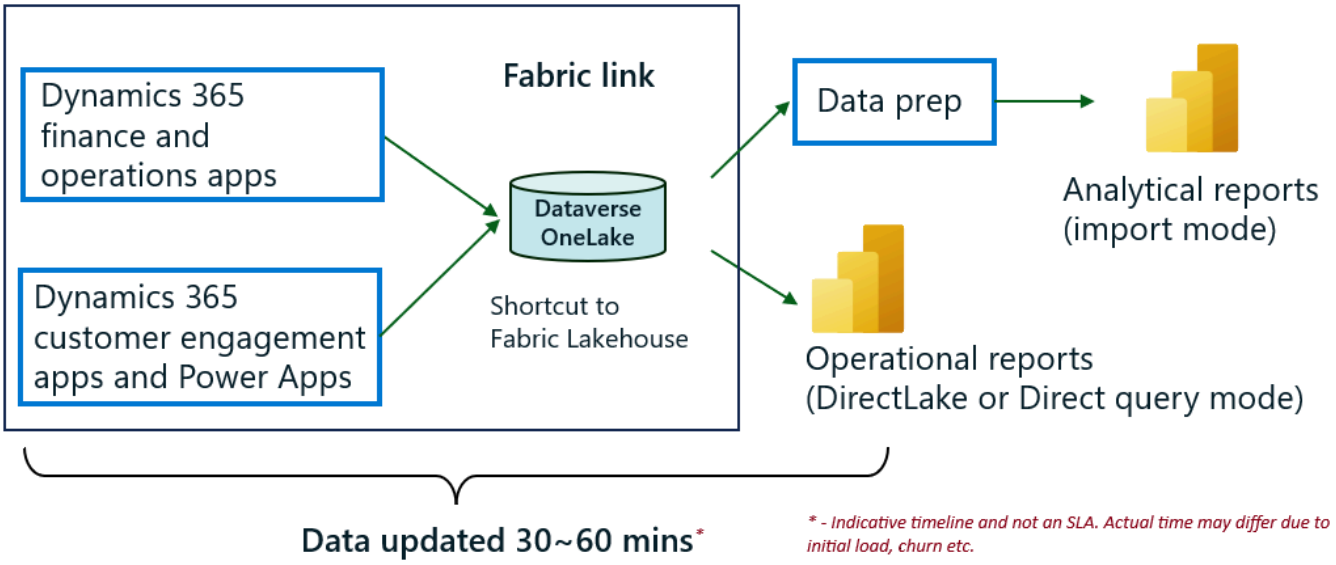
If you upgrade to Azure Synapse Link and continue to export data, your data is saved in Delta or Parquet format, which can reduce the steps in your own data pipelines for operational reporting.

Consider the time taken to refresh an operational report, such as inventory analysis or month-end financial analysis. These reports might require data aggregation of millions of rows of data from multiple tables in Dynamics 365 Finance. Using Export to Data Lake service, CSV data is exported within 10 minutes. This data might need to be imported into a Power BI report to provide better response times. Power BI refresh can be performed up to 24 times a day (every 30 minutes). Depending on how the report is designed, refresh can take several minutes to complete refresh. Using this approach, users can see data within 60 minutes of an update. These timelines are indicative. Actual sync time depends on factors such as the initial data load size, data churn rate, and the volume of change.



Operational reports that source data using Fabric link can leverage Direct Lake mode or DirectQuery mode in Fabric, which leverages the in-memory index built into Delta or Parquet

files. In these modes, you don't need to schedule refresh of Power BI reports as the report always shows latest Dataverse data updated in Fabric.



The Fabric link service updates data in Dataverse OneLake within the hour as of this point in time. Dataverse triggers data update jobs every 15 minutes and depending on the volume of data changes, you might see updated Parquet files within 15 to 45 minutes. These timelines are indicative. Actual sync time depends on several factors including the initial data load size, data churn rate, and the volume of changes. In situations with high transaction volumes, such as processes in Dynamics 365 finance and operations apps generating millions of records in a short time, or processes like the master planning feature included with Dynamics 365 Supply Chain Management that delete and re-create large volumes of records, Synapse Link must synchronize all changes including deletes. In these high-volume scenarios, data availability within the indicated timeframes can't be guaranteed.

If you're consuming incremental data feeds from Dynamics 365 with Export to Data Lake service for near-real time data integration scenarios (example 3). Upgrade to Azure Synapse Link enables you to run the same data pipelines.

Limitations and changes to behavior

You might find a few changes in table formats introduced that help avoid conflicts. More information: Known limitations.

Also verify that you have the latest updates, which address some of the known issues. To learn more about the upcoming roadmap and stay in touch with the product team, join the preview Viva Engage group.

Frequently asked questions

Questions? you can refer to the FAQ document published here. You can also join weekly office hrs, stay in touch with the product team and the customer forum by joining Viva Engage group.

See also

What is Azure Synapse Link for Dataverse?

Link your Dataverse environment to Microsoft Fabric and unlock deep insights

---

Last updated on 03/30/2026

Frequently asked questions when transitioning from legacy data integration services

Summarize this article for me

This article discusses the frequently asked questions with transition from legacy data integration services to Fabric link or Azure Synapse Link for Dataverse.

Data export service, bring your own database (BYOD), and export to data lake were features introduced in Dynamics 365 apps to export data for analytics and data integration scenarios. These services enabled IT admins and specialists to export data into external databases or data lakes and build data integration pipelines. While we improved these services over the years with updates, as part of unification of Dynamics 365 with the Power Platform, the same capabilities are rearchitected for these disparate services into simpler, unified experiences built into Power Apps (make.powerapps.com). The unified and improved service is called link to Azure Synapse.

Also available is a service named link to Microsoft Fabric where you can directly integrate Microsoft Dataverse, the data platform behind Dynamics 365 as well as Power Apps with Microsoft Fabric with no-copy, no-ETL required. Whether you link to Microsoft Fabric or upgrade to Azure Synapse Link, the rearchitected services provide you with an easy ramp to benefit from AI and Copilot investments in Dataverse and Fabric.

What is the difference between Link to Azure Synapse and link to Fabric?

Link to Azure Synapse and link to Fabric services are built into Dataverse, the platform powering Power Apps and all Dynamics 365 products.

Azure Synapse Link continuously exports data from Dynamics 365 and Power Apps to your own storage account, allowing IT professionals to build and manage data integration pipelines with Azure Synapse Analytics as well as other tools.

In contrast, link to Fabric offers a no-copy, no-ETL integration directly with Microsoft Fabric, keeping data within the Dataverse governance boundary for enhanced security. Fabric Link uses Dataverse managed storage, removing the need to configure and manage your own storage. When using link to Fabric, you might notice an increase in Dataverse storage consumption.

Both these services operate across PowerApps, Dynamics 365 customer engagement apps like Dynamics 365 Sales, Dynamics 365 Customer Service, as well as Dynamics 365 finance and operations apps.

Expand table

Link to Fabric	Azure Synapse Link
No copy, no extract, transform, load (ETL) direct integration with Fabric.	Export data to your own storage account and integrate with Azure Synapse, Fabric, and other tools.
Data stays in Dataverse. Users get secure access in Microsoft Fabric.	Data stays in your own storage. You manage access to users.
All tables chosen by default.	System administrators can choose required tables.
Consumes additional Dataverse storage.	Consumes your own storage and other compute and integration tools.

What option should I consider if I want to use Microsoft Fabric?

Fabric Link provides a seamless, no-copy integration with Dataverse and Microsoft Fabric, eliminating the need to manage complex data export processes. With link to Microsoft Fabric, your data stays securely within your Dataverse environment, while authorized users in Microsoft Fabric can access the data, making it easier for your team to leverage data for insights without the hassle of managing separate storage and integration solutions. This streamlined approach can lead to cost savings and more efficient data utilization, especially if you're already using Power BI. [Go to this article](/powerapps-docs/maker/data-platform/azure-synapse-link-transition-from-FnO.md#Simplification with Fabric link) for a comparison of total cost of ownership (TCO) between link to Microsoft Fabric and link to Azure Synapse.

Using the Fabric Link transition tool provided by Dynamics 365 FastTrack team, you can create a Fabric data warehouse with the same data shapes as export to data lake feature and minimize the effort to transition your existing reports and data pipelines.

What option should I consider if I want to keep the data export solution?

Upgrading from export to data lake to Azure Synapse Link enables you to preserve your investments with minimal changes. While Azure Synapse Link simplifies data integration by

continuously exporting data in a more efficient Delta Parquet format, it provides similar outputs as export to data lake service therefore reducing the effort to switch. You can use Power BI, Microsoft Fabric as well as many non-Microsoft tools that support Delta Parquet format to work with your Dynamics 365 data. You can also secure your storage account with firewalls and enable restricted access via managed identities, ensuring secure data access and compliance. Go to this article for a comparison of total cost of ownership (TCO) between link to Fabric and link to Azure Synapse.

Using the Synapse Link transition tool provided by Dynamics 365 FastTrack team you can create a Synapse workspace with similar data shapes as export to data lake feature and reduce the effort to transition your existing reports and data pipelines.

We don't use Power BI or Fabric with Dynamics 365 data, can I extract change data?

If you use Power BI or Microsoft Fabric to gain insights from your Dynamics 365 or Power Apps data, Fabric Link provides the most benefits while simplifying data integration efforts. However, if you're using non-Microsoft tools and/or use export to data lake change feeds to incrementally export data to your downstream data warehouse, you can use Synapse Link incremental data exports instead. More information

Using the The Data integration tool provided by Dynamics 365 FastTrack team you can create a pipeline to incrementally copy change data into an Azure SQL database, Synapse dedicated pool or an on-premises database.

Are there differences in columns/schema between Export to Data Lake and Fabric Link or Synapse Link?

Fabric Link and Synapse Link represent a unification of legacy services built into Dynamics 365. The same capabilities are rearchitected to transform these disparate services into simpler, unified experiences built into Power Apps (make.powerapps.com). In the process, to avoid conflicts, minor changes are made to the schema. The underlying data engine behind Fabric Link and Synapse Link is the same so the changes apply both to Fabric Link and Synapse Link options:

- Renaming SQL reserved words such as `Level_, Resource_`.
- ID field in export to data lake tables become `FnO_Id`.

- Deleted rows are included. For example, Synapse Link and Fabric Link perform what is referred to as a "soft delete." You can use the `isDelete` flag to identify deleted rows and remove them.
- TimeZoneID (TZID) fields are removed because these are legacy fields no longer used in finance and operations apps.
- Fields that contain binary data are removed.
- `nVarChar(max)` fields are included but data is truncated at 2,000 characters.

For more information, go to working with Finance and Operations data and metadata.

Using the Fabric Link transition tool provided by Dynamics 365 FastTrack team, you can create a Fabric data warehouse with the same data shapes as export to data lake feature and minimize the effort to transition your existing reports and data pipelines.

Using the Synapse Link transition tool provided by Dynamics 365 FastTrack team you can create a Synapse workspace with similar data shapes as export to data lake feature and reduce the effort to transition your existing reports and data pipelines.

Using the data integration tools provided by Dynamics 365 FastTrack team you can create a pipeline to incrementally copy change data into an Azure SQL database, Synapse dedicated pool or an on-premises database.

Why are all Dataverse tables auto selected with Fabric Link, why can’t I select only the Dataverse tables I need?

By selecting **Link to Microsoft Fabric**, the system adds all nonsystem Dataverse tables that have the **Track changes** property enabled. You can’t remove these tables as some of these tables might be used by Dynamics 365 as well as partner applications and removing them can cause the applications to fail.

Enabling link to Fabric feature (or a Dynamics 365 application that contains insights features) might result in an increase in Dataverse database storage consumption. You can view additional storage consumption as additional files in the environment storage capacity details view in the Power Platform admin center. For example, you might notice an additional file "Account-Analytics" if you selected the "Account" table for Fabric link. Also note that the chart only displays tables consuming highest storage. You can get a list of tables using the menu on top right of the chart.

Export to data lake service is working well for us. Why did you retire these services?

As part of the Dynamics 365 platform unification project, we consolidated several services that "exported data" in Dynamics 365 applications to a single service under the brand name Synapse Link for Dataverse. Using our learnings, many shortcomings of existing services were addressed in Synapse Link while maintaining same data formats – so that you can upgrade your investments with minimal disruptions. Fabric link service is a new, no-copy, no-ETL solution, which enables you to leverage innovations in Fabric without having to invest in data pipelines. It's a "read-replica" of your data for authorized users in Fabric.

Why can't I find some Tables and Entities from Finance and Operations in tables in Link to Fabric or Link to Azure Synapse

Azure Synapse Link or Fabric link enables tables where the "change tracking" property is enabled. Currently, change tracking can't be enabled for all finance and operations entities. The **Track changes** option is unavailable for entities created in finance and operations in the past for data migration. In some entities, enabling change tracking might fail with the error message **chosen entity doesn't pass the validation rules** or the **Track changes** checkbox is disabled for some entities.

For more information about entity validation rules and how you can fix them, go to Enable row version change tracking for data entities. You might need developer assistance to complete the steps. If the chosen entity is unavailable because of the change tracking limitation, choose the tables that comprise the data from that entity.

You can use the EntityUtil solution provided by the FastTrack team to create entity shapes using tables.

Is BYOD service retired? Is there a retirement date?

As part of the Dynamics 365 platform unification project, several services that exported data in Dynamics 365 applications are consolidated. BYOD is a service that exports entity data into a customers own Azure SQL database. While a retirement date for BYOD service hasn't been determined, we recommend that you transition to Synapse Link or Fabric link services. Fabric link service is a new, no-copy, no-ETL solution, which enables you to query your data with SQL similar to a "read-replica" of your data in Fabric.

Will you retire Synapse Link in the future?

As part of the platform unification project, we have consolidated several services that exported data in Dynamics 365 applications to a service under the brand name Synapse Link for Dataverse. Fabric link, a no-copy, no-ETL solution, which enables you to leverage innovations in Fabric without having to invest in data pipelines is also available. Link to Fabric is a "read-replica" of your data for authorized users in Fabric.

While many customers are excited to adopt Fabric, we also understand some customers want to continue to export data and build their own integration pipelines. As such, we want to enable you to export data well into the future.

Will export to data lake service stop since it's deprecated after November 1, 2024?

Export to data lake service is deprecated as of November 1, 2024 and support will be limited. You can reduce the risk by beginning the transition process now. We plan to decommission export to data lake service beginning March 25, 2025 as customers transition to Synapse Link and Fabric link services. We plan to start the decommissioning process with customers who have completed the transition as well as customers who aren't actively using the service. You'll be notified before the decommission process begins and have the option to ask for a past due extension via the link aka.ms/SynapseLinkPastDue.

My transition might run beyond deprecation date, is there a process to get an extension?

Export to data lake service is deprecated as of November 1, 2024, but will continue to operate under limited support until March 25, 2025. If you haven't done so already, you should begin the transition process now.

To help customers who might need more time to complete their transition, we introduced an in-product "past due extend" option. You can apply for a one-time "past due extension" by taking the survey the Azure Synapse Link area in Power Apps (make.powerapps.com) or the link aka.ms/SynapseLinkPastDue. This option is available only to customers who are currently validating either Synapse Link or Fabric Link services.

If a past due extension is approved, you receive a confirmation e-mail and the extended date is shown within the product. You can join the community and stay in touch via the forum or weekly office hours at aka.ms/SynapseLinkforDynamics.

I have completed the transition, how can I uninstall export to data lake feature?

If you aren't using this feature in your environment, ask your finance and operations apps administrator to uninstall this feature. To uninstall, visit the environment page in life cycle services (LCS), navigate to the **Environment Add-ins** section, and then select the **Uninstall** option in the **Export to Data lake** environment add-in.

I received an email about transitioning Azure Synapse Analytics workspaces to private links before 1 August 2026. What do I need to do?

As described in the email, the trusted services function that allows Azure Synapse Analytics to access Azure Storage accounts and Azure Key Vault using a managed identity and firewall exception will be retired on **1 August 2026**.

If you decide to transition your Synapse workspace to private links as described in the email, complete the following steps to ensure Azure Synapse Link for Dataverse continues to work as expected:

1. Create a Synapse workspace with a managed virtual network. A managed virtual network isolates your Synapse workspace and is required to use managed private endpoints. Follow the steps in Azure Synapse Analytics managed virtual network.

2. Create a managed private endpoint for your storage account (ADLS Gen 2). This replaces the trusted services firewall exception with a private, secure connection. Follow the steps in Create a managed private endpoint to your data source.

3. Enable these network settings:

    - On the **storage account**: enable **Allow Azure services and resources to access this storage account** (under **Networking** > **Firewalls and virtual networks**).
    - On the **Synapse workspace**: enable **Allow Azure services and resources to access this workspace** (under **Networking**).

    Both settings must be enabled for Azure Synapse Link to connect successfully.

After completing these steps, your Synapse workspace connects to the storage account through a private endpoint instead of the trusted services firewall exception.

Related articles

[What is Azure Synapse Link for Dataverse?]

[Link your Dataverse environment to Microsoft Fabric and unlock deep insights]

Last updated on 02/20/2026

Link your Dataverse environment to Microsoft Fabric and unlock deep insights

Microsoft Dataverse direct link to Microsoft Fabric enables organizations to extend their Power Apps and Dynamics 365 enterprise applications, and business processes into Fabric. The **Link to Microsoft Fabric** feature built into Power Apps makes all your Dynamics 365 and Power Apps data available in Microsoft OneLake, the built-in data lake for Microsoft Fabric.

- No need to export data, build extract, transform, load (ETL) pipelines, or use our partner integration tools.
- With shortcuts from Dataverse directly into OneLake, your data stays in Dataverse while authorized users get to work with data in Fabric.
- Link data from all Dynamics 365 apps, including Dynamics 365 Finance and Operations apps.
- Build Power Apps and automations to drive action from insights in OneLake.

Microsoft OneLake, a data lake built into Fabric, helps eliminate data silos. Combine data from your applications and devices web sites, mobile apps, sensors, and signals from your warehouse and factories with data from your business processes in Dynamics 365, such as sales, cases, inventory, and orders, to predict potential delays or shortages that affect keeping your promises to customers. Dataverse creates shortcuts to OneLake, which enables you to work with data without making multiple copies.

Dataverse also generates an enterprise-ready Microsoft Fabric Lakehouse and SQL endpoint and a Power BI dataset for your Power Apps and Dynamics 365 data. This makes it easier for data analysts, data engineers, and database admins to combine business data with data already present in OneLake using Spark, Python, or SQL. As data gets updated, changes are reflected in lakehouse automatically.

Low-code makers can build apps and automations to orchestrate business processes and react to insights found in Fabric. By adding those insights back to Dataverse as virtual tables connected to OneLake, makers build low-code apps with Power Apps, Power Pages, or Power Automate using the design tools already available. Using connectors to over 1,000 apps, makers create business processes that span Dynamics 365 as well as many other enterprise applications.

Watch this video to learn about accessing Dataverse data in Fabric:

https://learn-video.azurefd.net/vod/player?id=ddd288c6-e47a-4bc6-974b-4fe0f1a49362&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fazure-synapse-link-view-in-fabric

Link to Fabric from Power Apps

From the **Tables** area in Power Apps (make.powerapps.com), makers link to Fabric by selecting **Analyze** > **Link to Microsoft Fabric** on the command bar.

When you select the command for the first time, a wizard driven experience links your Dataverse environment to a Fabric workspace. A Fabric Lakehouse, SQL endpoint, a Power BI dataset are created and Dataverse tables are linked to the lakehouse via shortcuts.

Once the link is set up, you can grant other users access to the Fabric workspace so that they can work with Dataverse data using all Fabric workloads. The lakehouse, SQL endpoint, and Power BI dataset are updated with new data as changes occur in Dataverse. More information: Link to Microsoft Fabric

Direct access to your data in Microsoft OneLake

Link to Fabric creates a direct and secure Link between your data in Dataverse and a Fabric workspace. There's no need to provide a storage account or Synapse workspaces. When you link to Fabric from Power Apps, the system creates an optimized replica of your data in delta parquet format, the native format of Fabric and OneLake, using Dataverse storage such that your operational workloads aren't impacted. This replica is governed and secured by Dataverse and stays within the same region as your Dataverse environment while enabling Fabric workloads to operate on this data.

Admins can manage tables linked to OneLake from the **Azure Synapse Link for Dataverse** page. By opening the **Microsoft OneLake** link, admins view tables added by makers, add more tables, and migrate the link to other environments. Tables added to OneLake consume Dataverse storage and admins can see storage consumption in the Power Platform admin center.

> **Note**
>
> By selecting **Link to Microsoft Fabric**, the system adds all nonsystem Dataverse tables that have the **Track changes** property enabled. You can add more tables later.
>
> If you used this feature during public preview, you can continue to use the tables selected earlier. If you choose the **Link to Fabric** option in new environments (or unlink and relink existing environments), all tables are added.
>
> Enabling this feature results in an increase in **Dataverse database** storage consumption. You can view additional storage consumption in the environment storage capacity details view in Power Platform admin center. For an example, you notice an additional file

`Account-Analytics` if you selected the `Account` table for Fabric link. Also note that the chart only displays tables consuming the most storage. You can get a complete list of tables using the menu on the top right of the chart.

Comparing Link to Fabric with Azure Synapse Link for Dataverse

Azure Synapse Link for Dataverse enables IT admins to export data to their own storage and build data integration pipelines. Azure Synapse Link assists with provisioning and configuring Azure resources within an integrated experience.

Link to Fabric feature enables direct connectivity between your data in Dataverse with Microsoft Fabric without the need to bring your own storage and Synapse workspaces. Link to Fabric leverages storage built into Dataverse and removes the need to provision and manage your own storage.

This table provides a comparison between the options.

Expand table

Link to Fabric	Azure Synapse Link
No copy, no ETL direct integration with Microsoft Fabric.	Export data to your own storage account and integrate with Synapse, Microsoft Fabric, and other tools.
Data stays in Dataverse - users get secure access in Microsoft Fabric.	Data stays in your own storage. You manage access to users.
All tables chosen by default.	System administrators can choose required tables.
Consumes additional Dataverse storage.	Consumes your own storage as well as other compute and integration tools.

You can connect existing Azure Synapse Links with Microsoft Fabric and benefit from Fabric innovations like Power BI DirectLake mode reports and integrated Spark and data pipelines. You must create an Azure Synapse Link for Dataverse profile and enable the **Delta parquet conversion for Fabric link** option. This option isn't available for Azure Synapse Link for Dataverse profiles that use the CSV output format.

Next steps

Link to Microsoft Fabric

Last updated on 01/09/2026

Link to Microsoft Fabric

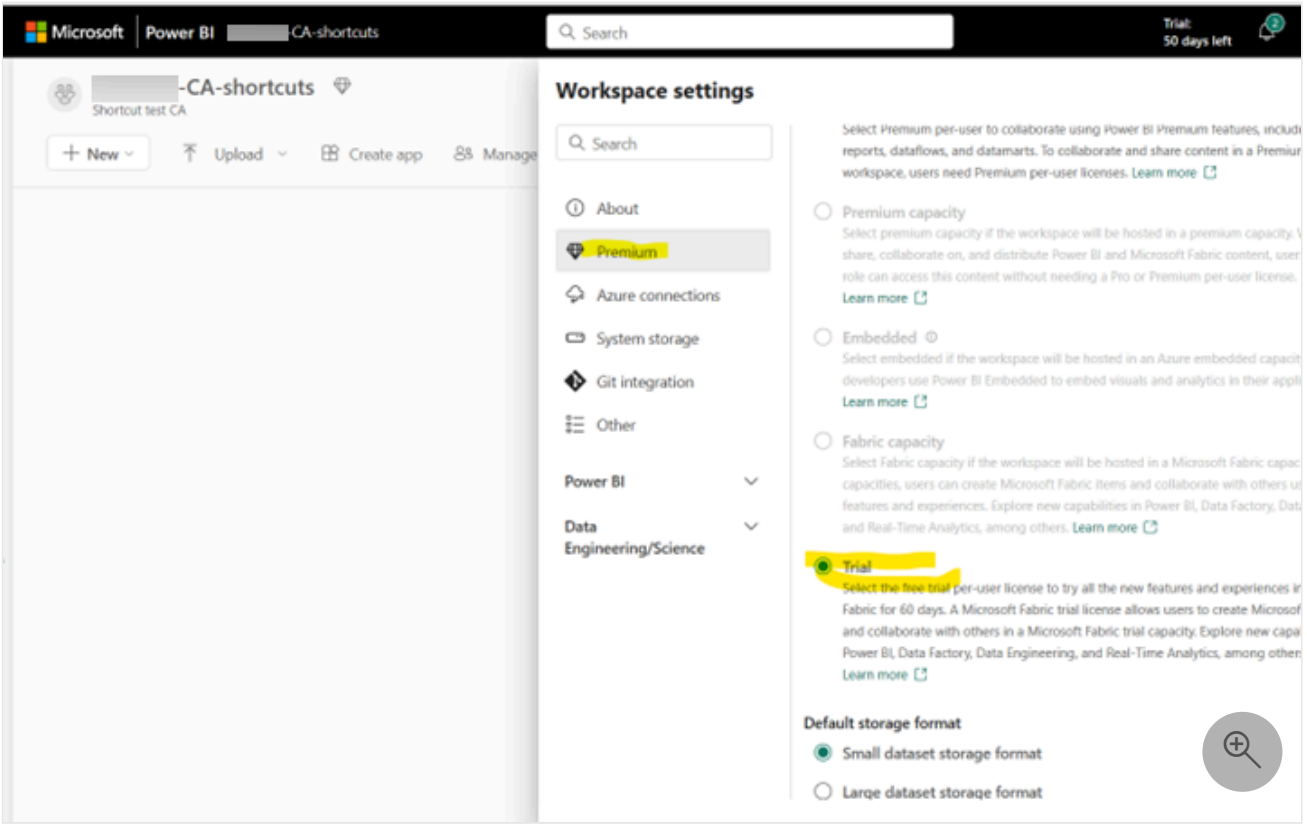
Microsoft Fabric integration with Microsoft Dataverse allows you to seamlessly link your Power Platform environment to Microsoft Fabric, enabling advanced data analysis and reporting capabilities. This guide provides step-by-step instructions on configuring your environment, linking it to Microsoft Fabric, and managing linked tables.

You can use an existing Dataverse environment or create a new developer environment if you want to try this feature. More information: Create a developer environment

Prerequisites

- You must have the System Administrator security role in the Power Platform environment to enable **Link to Fabric** or **Synapse Link**. This role is also required to add application users in your environment who need access to work with this feature by using the Power Platform admin center.
- You must be an administrator of the Power BI workspace.
- If you want the system to create a Power BI workspace, you need to have Power BI Capacity Administrator access to a capacity within the same region as the Dataverse environment.
- A Power BI premium license or Fabric capacity within the same Azure geographical region as your Dataverse environment is required. If you don't have Power BI premium license or Fabric capacity within the same geographical region, you can buy a capacity or sign up for a free Fabric trial capacity. More information: Fabric (preview) trial
- Your administrator needs to grant you access to create Fabric lakehouses and artifacts. You can find these settings in the Fabric admin portal. Go to **Tenant Settings** > **Microsoft Fabric** > **Users can create Fabric items**, **Tenant settings** > **Workspace settings** > **Create workspaces** as well as **Tenant settings** > **oneLake settings** > **Users can access data stored in OneLake with apps external to Fabric**.
- If you plan to use workspace identity authentication, you must be a workspace admin to create and manage a workspace identity. The workspace you're creating the identity for can't be a My Workspace. More information: Workspace identity in Fabric
- If you plan to use workspace identity authentication, the workspace identity must be onboarded as an application user in the Dataverse environment and granted the appropriate role (commonly System Administrator) so it can access Dataverse data on behalf of Fabric.

- You must have permissions in Fabric to manage connections via **Settings** > **Manage connections and gateways**.
- To confirm whether you have access to the required premium capacity, go to Power BI, open the workspace, and select **Workspace settings** > **Premium**. Make sure that **Trial** or **Premium capacity** is selected.



Create a link to Fabric

Link to Microsoft Fabric from the Power Apps **Tables** area: Select **Analyze** > **Link to Microsoft Fabric** on the command bar.

1. Sign into https://make.powerapps.com.
2. Select the environment you want, select **Tables** on the left navigation pane, and then select **Analyze** > **Link to Microsoft Fabric** on the command bar.
3. If you're linking to Fabric for the first time, a wizard appears. You can launch Fabric with the same option in subsequent runs.
4. The wizard starts with **Validate configuration**, which checks your prerequisites and Fabric subscription settings. If you don't have a Fabric capacity in the same geography or region as your Dataverse environment, the wizard notifies you to get a capacity in the required geography.

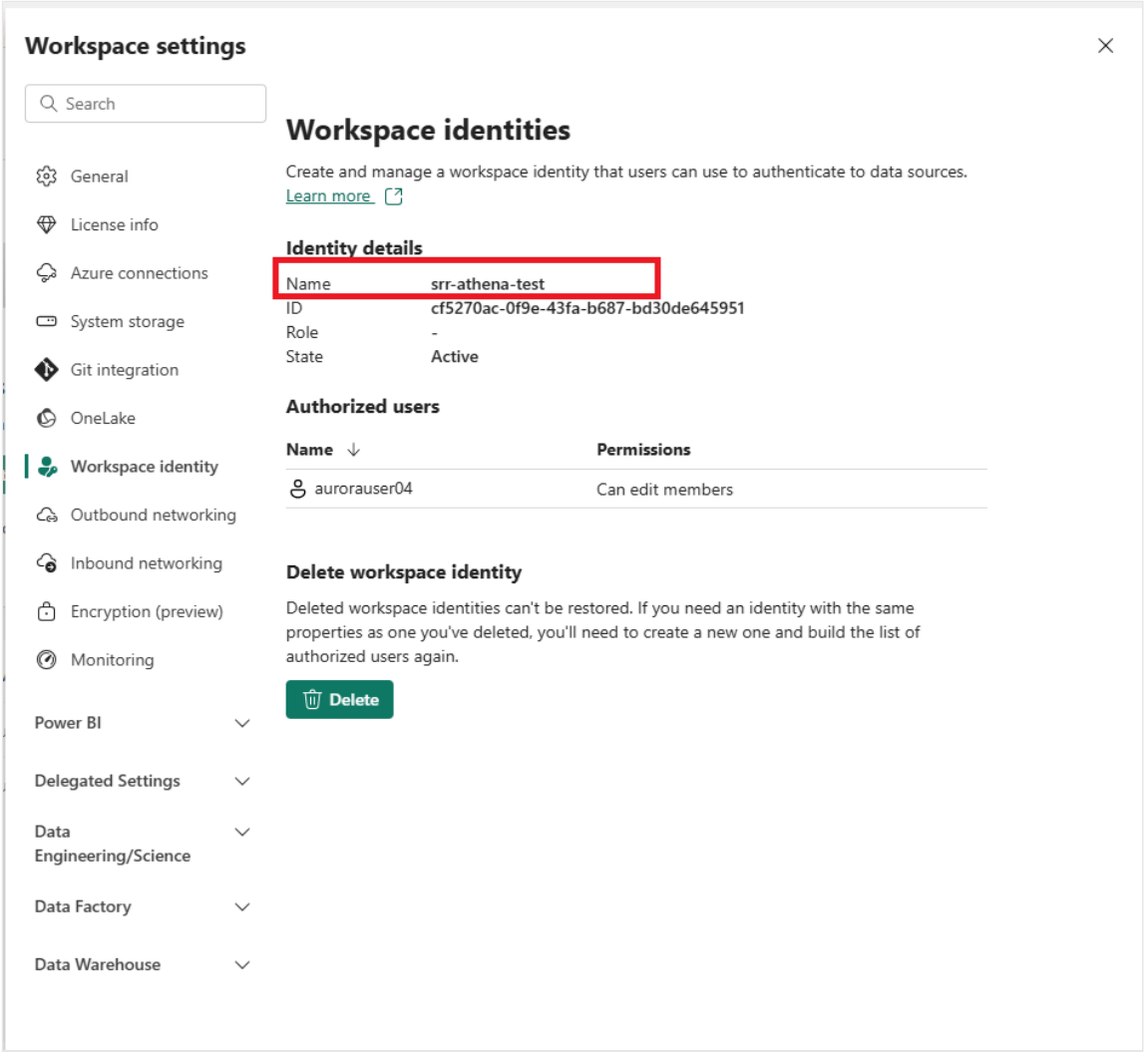
5. Next, choose the **Workspace** tab in the wizard. First, select a Fabric workspace. For that workspace, you also choose a connection for the first time. When creating a connection, you can select from three authentication options: **Organizational Account**, **Service Principal**, or **Workspace Identity**.

   - **Workspace Identity**:
     To use this option, you must first create a workspace identity in Fabric:

     a. In Fabric, open the target workspace.

     b. Go to **Workspace settings** > **Workspace Identity**.

     c. Select **+ Workspace Identity** to create it.
     - Note the name of the workspace identity (it matches the workspace name).



     d. For more information, go to Workspace identity in Fabric.

   - **Create an application user with worskpace identity**:
     After you create the identity, add it as an application user in Dataverse:
     - Go to **Power Platform admin center** > **Users (See all)** > **App users list**.

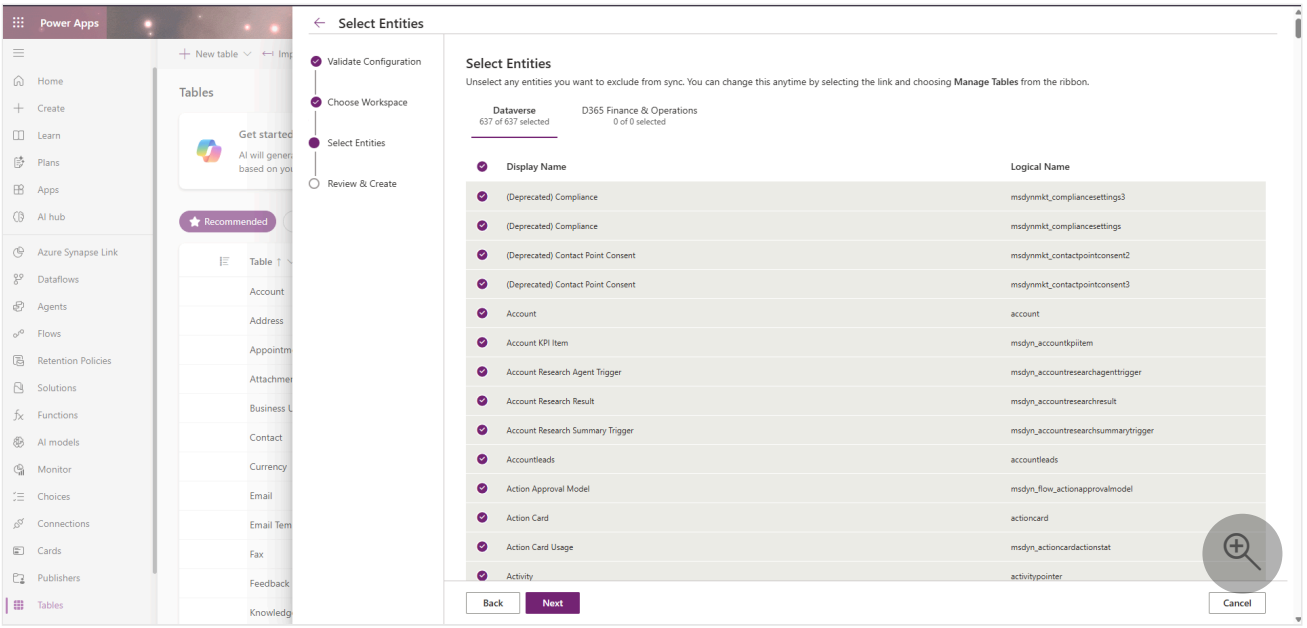
    - Select **+ New app user**, then **+ Add an app**, search for the workspace name you noted earlier(srr-athena-test in the above example), and add it.
    - Assign the correct **Business unit** and **System Administrator** role, then select **Create**. For detailed steps, go to the Create an application user in Dataverse step.

- **Service Principal**:
  To use this option, go to the create an application user with workspace identity previous step to create a service principal. Then, follow these steps:
  a. First, add the service principal as an application user in Dataverse following the same process described in the **Workspace Identity** section.
  b. After adding the application user, return to the **Link to Fabric** wizard.
  c. In the connection setup, provide the following details for your service principal:
    - **Tenant ID**: Your Azure tenant identifier.
    - **Client ID**: The application (client) ID of your service principal.
    - **Key**: The client secret or certificate for authentication.
  d. Save the connection to complete the setup.

- **Organizational Account**:
  Provide your credentials and save the connection. To change credentials later, select **Switch account** and provide new credentials.

6. You can expect to see shortcuts to all your tables within the selected workspace. If you don't see workspaces, ask the system to create a workspace. Go to Troubleshooting common issues if you don't see the desired workspace.

7. Next, in the **Select Entities** step, choose the tables you want to sync to Fabric. By default, all Dataverse tables where the **Track changes** property is enabled are selected. Clear any tables you don't want to sync. Only selected tables consume storage in Fabric, so you can optimize costs by excluding tables you don't need.



> **Note**
>
> You can change this selection anytime after setup by selecting the link and choosing **Manage Tables** from the ribbon. More information: Add or remove tables linked to Fabric

8. Review your selections in the **Review & Create** step. When you're done, select **Create** to create the workspace, create shortcuts, and to perform the initialization for the first time.

9. When complete, Fabric lakehouse opens in a separate browser tab.

> **Note**
>
> It might take up to 60 minutes to update data in OneLake including the conversion to delta parquet format. If you selected a table that contains a lot of data, the initial load time might take longer. When you open Fabric lakehouse, the links appear as **unidentified** until the initial sync is completed. More information: Troubleshooting common issues
>
> When the initial sync is complete, the system continuously refreshes updates in Dataverse in the lakehouse. It might take up to 60 minutes for the data to be refreshed especially during peak load periods.
>
> If you have more than 2,000 active Dataverse tables, Link to Fabric can fail with an error. Go to Troubleshooting common issues for help with resolving issues.
>
> The link was labeled as **Microsoft OneLake** and is now labeled **Link to Fabric**.

Manage link to Fabric

This section describes how to add or remove tables linked to Fabric, configure the link to use workspace identity, and share the data connection with other users.

Add or remove tables linked to Fabric

Admins can manage tables linked to Fabric from the **Azure Synapse Link for Dataverse** page. If this environment is linked to Fabric, you see a link called **Link to Fabric**.

Add tables to Fabric

1. Sign into Power Apps.

   > **Note**
   >
   > This feature is enabled by default on all environments. Power Platform admins can disable this feature in the Power Platform admin center. More information: Environment settings: Microsoft Fabric

2. Select **Azure Synapse Link** from the left navigation pane, and then select **Link to Fabric**.

3. Open Fabric by selecting **View in Microsoft Fabric**.

4. Add more table links to Fabric by selecting **Manage tables**.

5. When you add a table, the system performs an initial sync and indexes the data. When the initial sync is completed, a shortcut to OneLake is created. View the status of tables by selecting **Manage tables**. Use the **Refresh Fabric tables** option to add the newly enabled table in Fabric. You might need to review the report and downstream data flows to see that they aren't impacted by the change.

   > **Note**
   >
   > If your environment is linked to a Dynamics 365 finance and operations environment, the add tables option enables you to include tables from finance and operations apps. Learn more: Choose finance and operations data in Azure Synapse Link for Dataverse

6. When the sync status is **Active**, as data gets updated, your data changes are shown in reports created in Fabric.

7. If a new column is added to a table that's already part of the profile (also known as a metadata change), you can use the **Refresh Fabric tables** option, from the command bar, to update the change in Fabric. The update occurs after the next table data change is triggered. You might need to review the report and downstream data flows to confirm that they aren't impacted by the change.

8. You can also **Unlink**, which removes the Fabric link to your Dataverse environment. When unlinking, the Fabric lakehouse is also removed.

> **Note**
>
> If you've installed Dynamics 365 apps such as Customer Insights, the tables required for the app are also included in the **Link to Fabric** link.

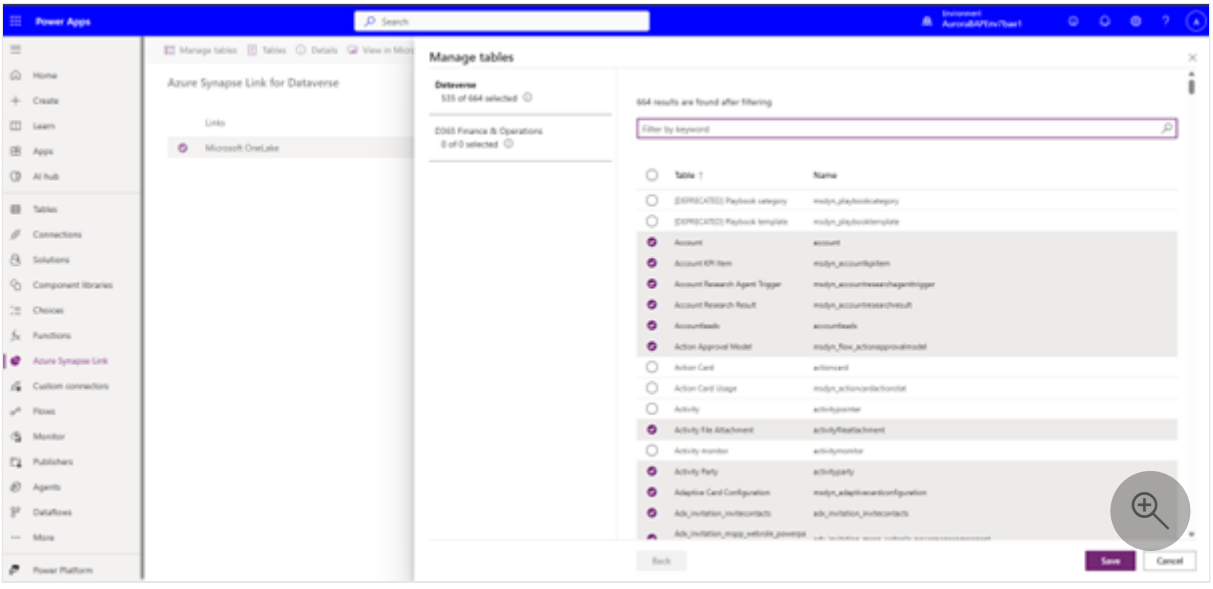
Remove tables from Fabric

You can stop syncing specific Dataverse and finance and operations apps tables to reduce storage costs and optimize your Fabric workspace. After accessing **Manage tables** (as described in step 4 in Manage link to Fabric), follow these steps to unlink tables:

1. The **Manage tables** pane shows all Dataverse and finance and operations apps tables currently linked to Fabric.

2. Tables that are already syncing are checked.

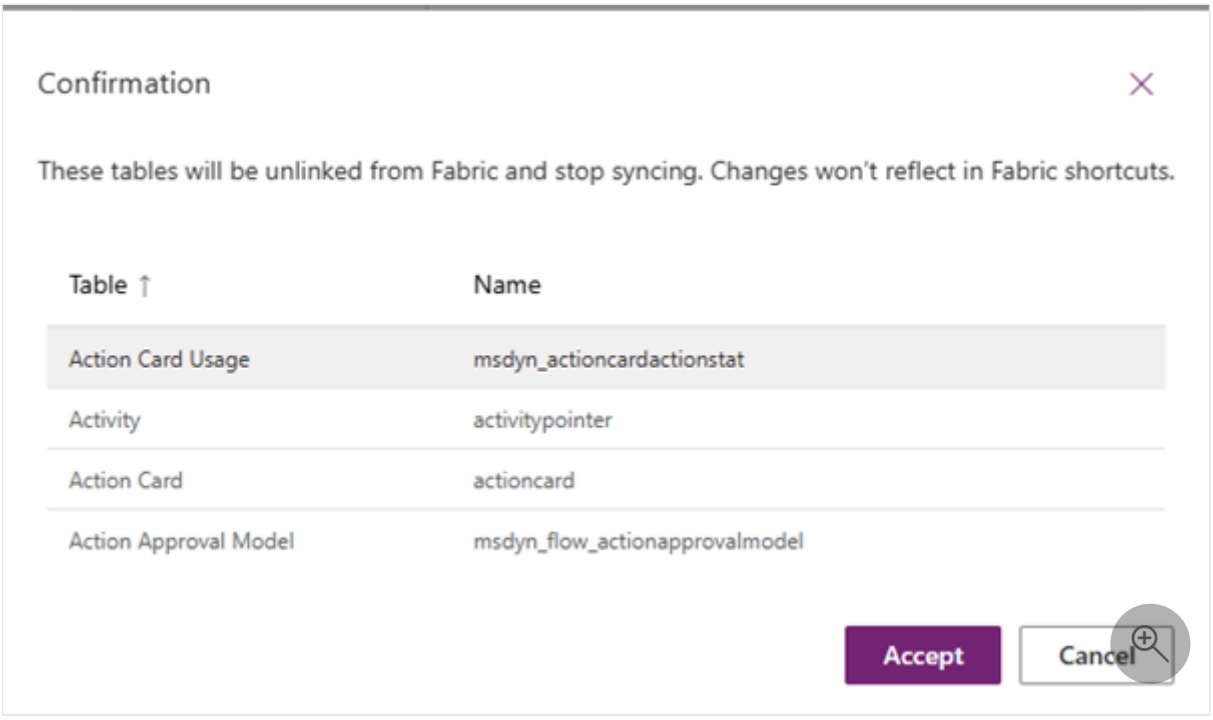
3. To stop syncing a table, clear the option to select it.



4. To keep syncing, leave it checked.

5. After making your selections, select **Save**.

6. A confirmation dialog appears listing the tables that stop syncing.



7. Review the list, and then select **Confirm**.

After confirmation:

- Shortcuts for the unselected tables are removed from the Fabric lakehouse.
- Selected tables are removed from the internal storage that are synchronized to Fabric via shortcuts. This ensures that you're paying only for the storage of tables that you selected.
- Synchronization stops immediately for those tables no longer selected.
- Remaining tables continue syncing without interruption.

> ⓘ **Important**
>
> - Some system tables and tables required by Microsoft add-ins can't be removed. More information: [Are there system tables that are automatically synchronized and can't be unlinked?]
> - Removing a table doesn't delete the table in Dataverse; it only removes the OneLake shortcut and stops data sync.
> - If you need to add tables later, repeat the same steps and check the tables you want to include. More information: [Manage link to Fabric].

Configure Fabric link to use workspace identity

If you linked your Dataverse environment to Fabric earlier using an organizational account, you can switch to workspace identity or service principal. Workspace identity is a Fabric managed service principal bound to a specific workspace; it removes the need to manage secrets and enables secure, keyless authentication for Fabric items that connect to Dataverse.

Things to verify before you begin

Before proceeding, ensure you meet the requirements described in the Prerequisites section, and specifically verify these three essential items:

- Workspace identity exists: The target Fabric workspace must have a workspace identity configured. Go to **Workspace settings** > **Workspace Identity**. If the identity doesn't exist, create it first. More information: Workspace identity in Fabric
- Dataverse application user: The same workspace identity must be onboarded as an application user in the Dataverse environment and granted the appropriate role (commonly System Administrator) so it can access Dataverse data on behalf of Fabric. This might already be in place if you followed the **Workspace Identity** steps in the Create a link to Fabric section previously.
- Rights to edit connections: You must have permissions in Fabric to manage connections via **Settings** > **Manage connections and gateways**.

What changes?

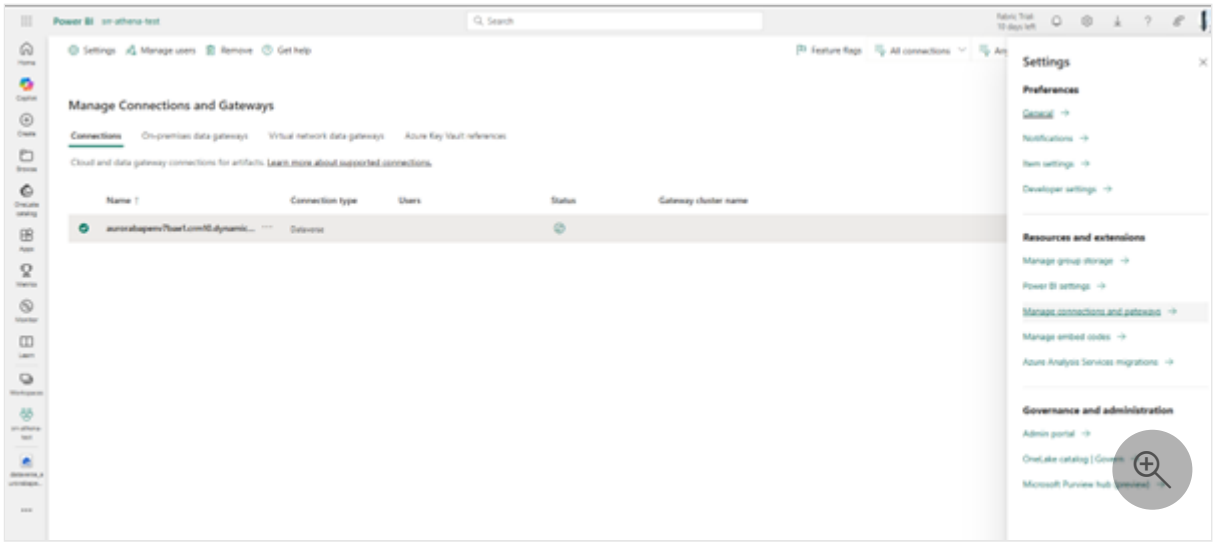
After you update the connection's authentication method to Workspace identity, all Fabric items that use that connection (for example, Lakehouses via OneLake shortcuts, Dataflows Gen2, pipelines, semantic models) authenticate with your workspace identity going forward.

Before you begin

Identify the connection name. When you used **Link to Microsoft Fabric**, the system created a Dataverse connection that typically includes (or matches) your environment name. You update this specific connection.

Switch to workspace identity authentication

1. **Open Fabric settings**
    - In Microsoft Fabric, select the gear icon (top right) and choose **Settings** > **Manage connections and gateways**.

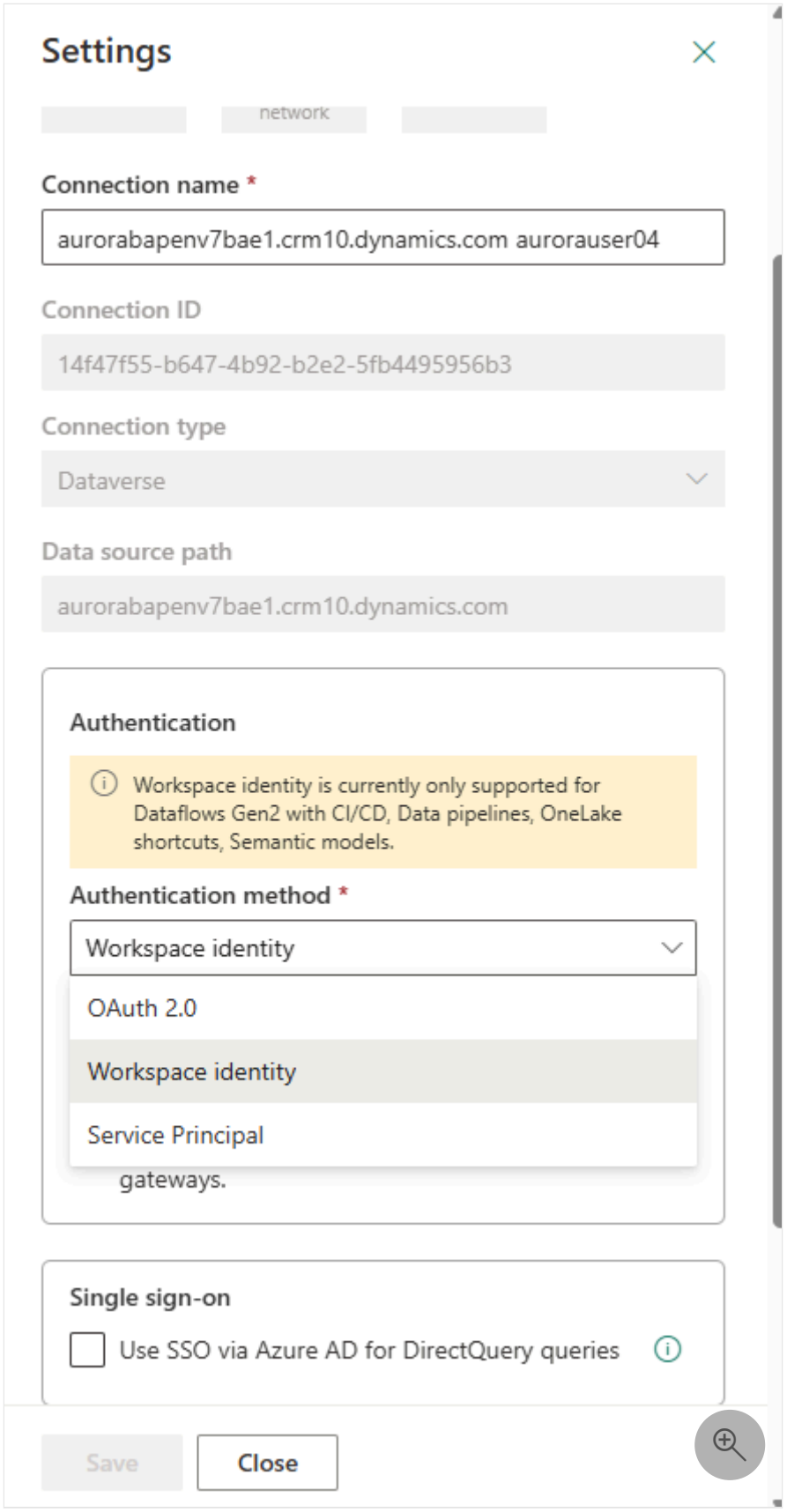


2. **Locate the Dataverse connection created by Link to Fabric**

    - In **Manage connections and gateways**, open the **Connections** tab (if not already selected).
    - Find the connection associated with your Dataverse environment. The **Link to Fabric** wizard typically creates this connection and names it using your environment's name so it's easy to recognize.

3. **Open the connection's management pane**

    - Select the ellipsis (⋮) next to the connection and choose **Settings** to open its settings.



4. **Switch the authentication method to Workspace identity**

- In the connection's **Settings** pane, go to **Authentication**.
- From the **Authentication method** dropdown, select **Workspace identity**.
- Select **Save** to apply the change.

> **Note**
>
> If your items run through a gateway, review your gateway and cloud connection policies as applicable. Fabric's manage connections and gateways experience governs connection usage, including gateway scenarios for shareable cloud connections.

Share the data connection with other users

The system creates a data connection between the Power Platform environment and Fabric workspace using the credentials of the user at the time of link creation. If you use the **Fabric link** option from the Power Apps **Tables** area, the system creates the connection and asks you to save it. If you use the **Synapse Link** option, you must create a data connection yourself before enabling the link.

The system uses this connection to enable Fabric users to connect to Dataverse - the data store behind the Power Platform environment. If you want to enable other users to add or remove tables to Fabric link, you need to share this data connection with other users.

1. Follow steps 1-2 from Switch to workspace identity authentication to access **Manage connections and gateways** and locate your Dataverse connection.
2. Once you select the correct data connection, select **...** > **Manage users**. Then you're shown users who have access to this connection.
3. Enter the name or email of other users who need access to data. When you select a user, specify either the **Owner** role or **Reader** role. You only need to provide reader role to enable them to consume data. The users you specify receive an e-mail confirming access to data.

You might need to grant access to other users to this workspace so that they can work with data. Depending on the need for data access, you might need to secure the data in this workspace before you share this data with others. You can secure the lakehouse as well as tables within the lakehouse using OneLake security. More information: OneLake security overview

You can only create user based connections at this time.

Link existing Azure Synapse Link for Dataverse links with Fabric

You can link your existing Azure Synapse Link for Dataverse profiles with Fabric from the **Azure Synapse Link for Dataverse** area. You need to select the **Enable Parquet/Delta lake** option to

enable the view in the Fabric feature for Azure Synapse Link for Dataverse profiles.

To enable an existing link, follow these steps:

1. Sign into Power Apps.
2. Select **Azure Synapse Link** from the left navigation.
3. Select an existing Azure Synapse Link for Dataverse profile, and then select **Link to Microsoft Fabric**.
4. You're prompted to choose a Power BI premium workspace to continue. A list of workspaces in the same region as your environment are displayed. If you don't see a workspace in the drop-down list, you might need to create one, and then return to this task. More information Link to Microsoft Fabric
5. Select **OK**. Validations are performed and the required artifacts are created in Fabric.
6. Select **View in Microsoft Fabric** open Fabric lakehouse.
7. You can add or remove tables using by selecting **Manage tables**. When you add a table, an initial sync is performed. When the initial sync is completed, select **Refresh Fabric tables** to refresh the Dataverse shortcut added to your Fabric lakehouse.

> **Note**
>
> - Select **Enable Parquet/Delta lake** to enable the view in Fabric.
> - Existing Azure Synapse Link for Dataverse profiles where the data is saved as CSV files can't be linked to Microsoft Fabric.
> - Currently, Azure Synapse Link profiles secured with managed identities, formerly Managed Service Identity (MSI), can't be linked to Microsoft Fabric.

Next steps

Work with Dataverse data and generate Power BI reports

Last updated on 04/27/2026

Work with Dataverse data and generate Power BI reports

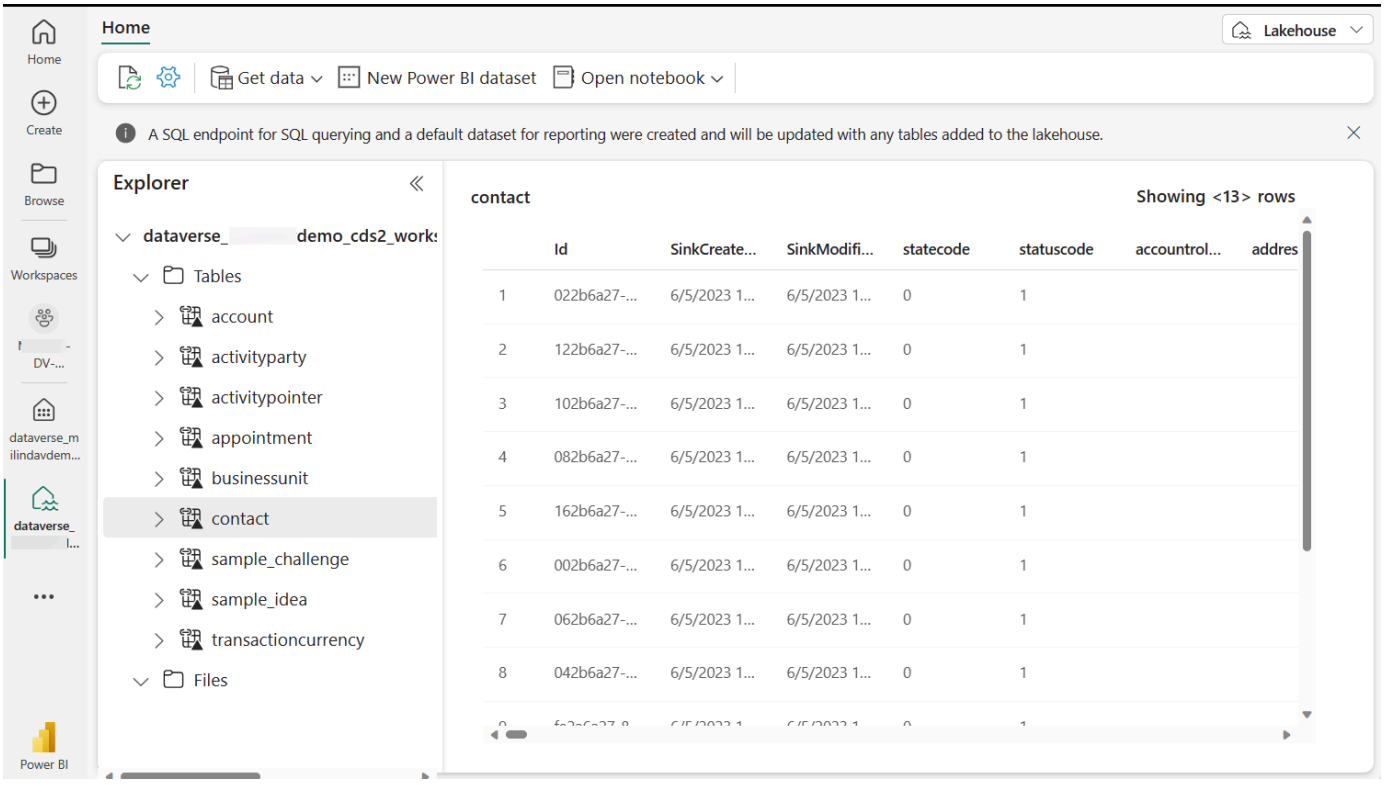
This section describes the different ways you can work with Microsoft Dataverse data in Microsoft Fabric and generate reports in Power BI.

You can view the Fabric lakehouse, SQL endpoint, and the default dataset generated by Dataverse in the Fabric workspace you chose earlier.

When you select **Link to Microsoft Fabric**, a Dataverse generated Fabric lakehouse opens. You can go to other Fabric features and work with Fabric and Power BI.

Explore the Dataverse generated Fabric lakehouse

The tables you selected are added to the Fabric lakehouse and displayed in Power BI as shown here. These tables are linked to your Power Platform environment using **Dataverse shortcuts**. As data changes in Dataverse, the Dataverse shortcuts in Fabric reflect the latest data.

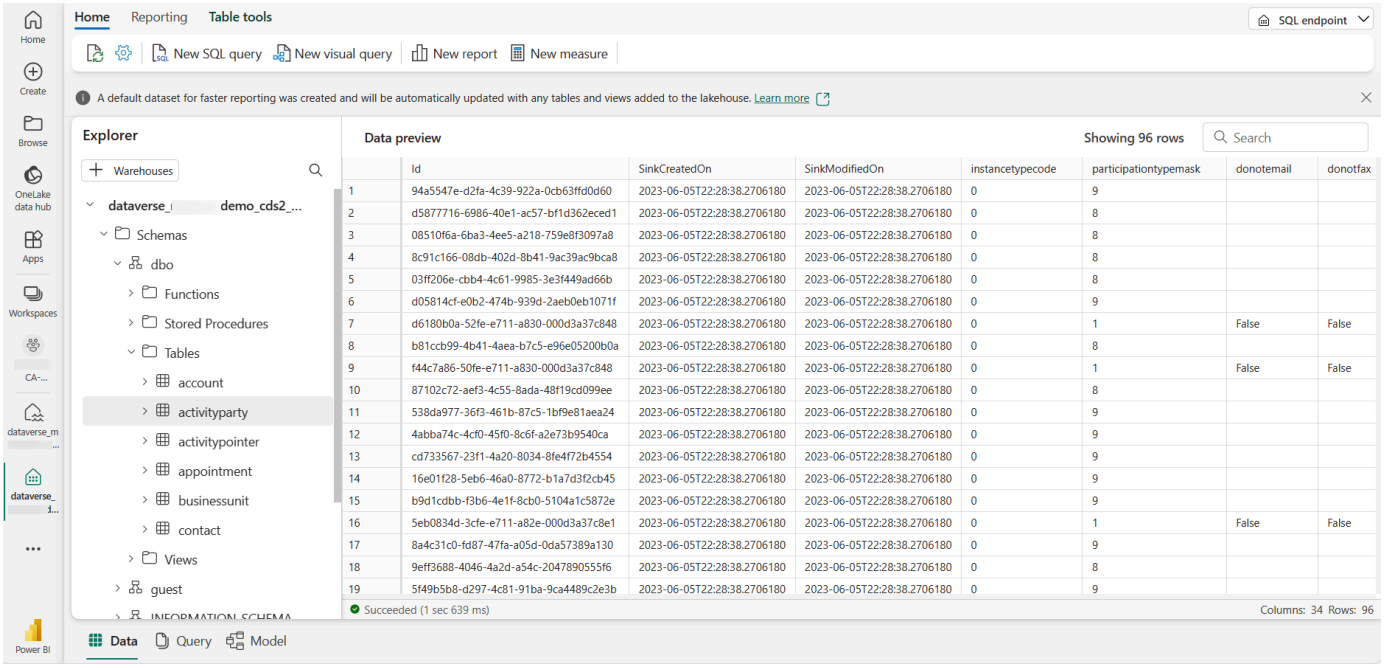


Note that Dataverse manages these shortcuts. You shouldn't delete or remove these shortcuts in Fabric. If you accidentally delete a link, you can go to the **Azure Synapse Link for Dataverse** area in Power Apps and select **Refresh Fabric links** to re-create the links.

Explore data with SQL endpoint

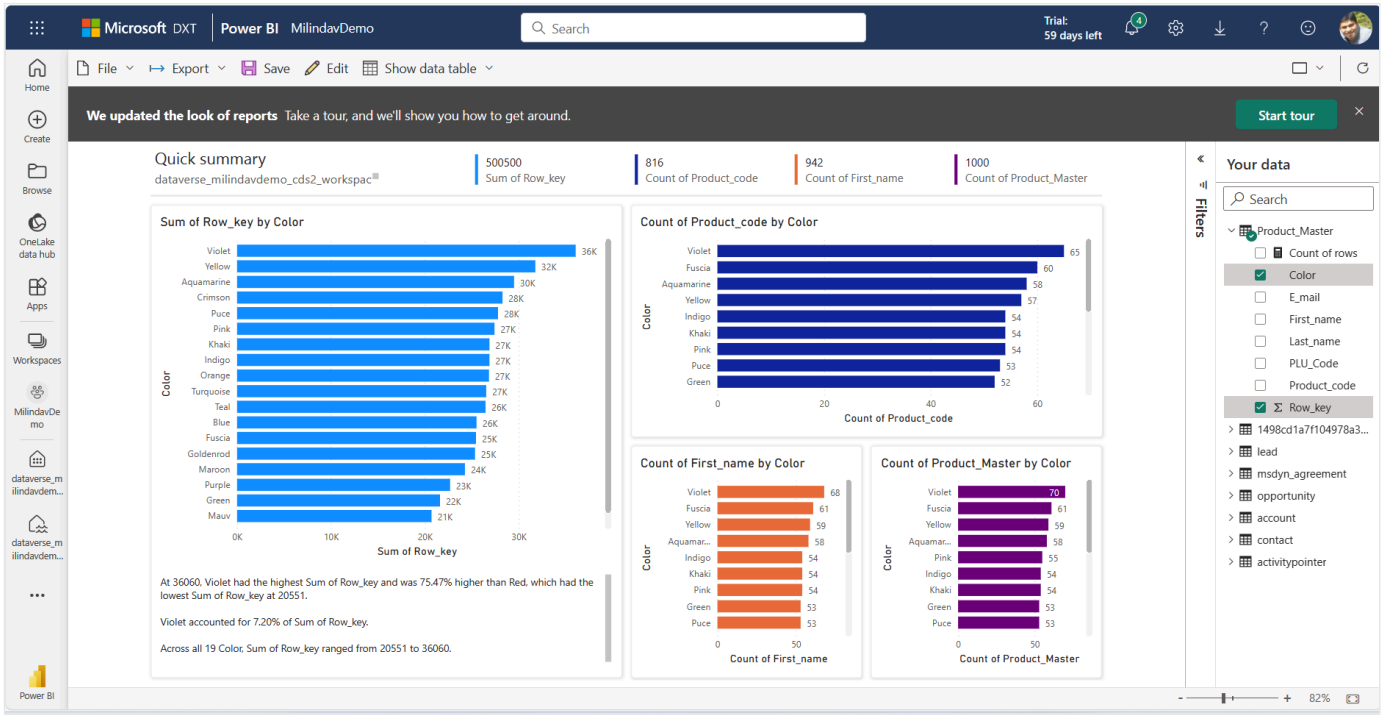
You can open SQL endpoint and query Dataverse data with SQL and generate views in Fabric.

In Power BI, select **SQL endpoint** from the top right context menu. The data is displayed in a SQL friendly experience where you can create SQL queries and views.



Autocreate a Power BI report

Choose the default dataset generated by Dataverse, and then select **Auto-create report**. A Power BI report with the data you have selected is created.



Related articles

Link your Dataverse environment to Microsoft Fabric and unlock deep insights

Last updated on 04/30/2026

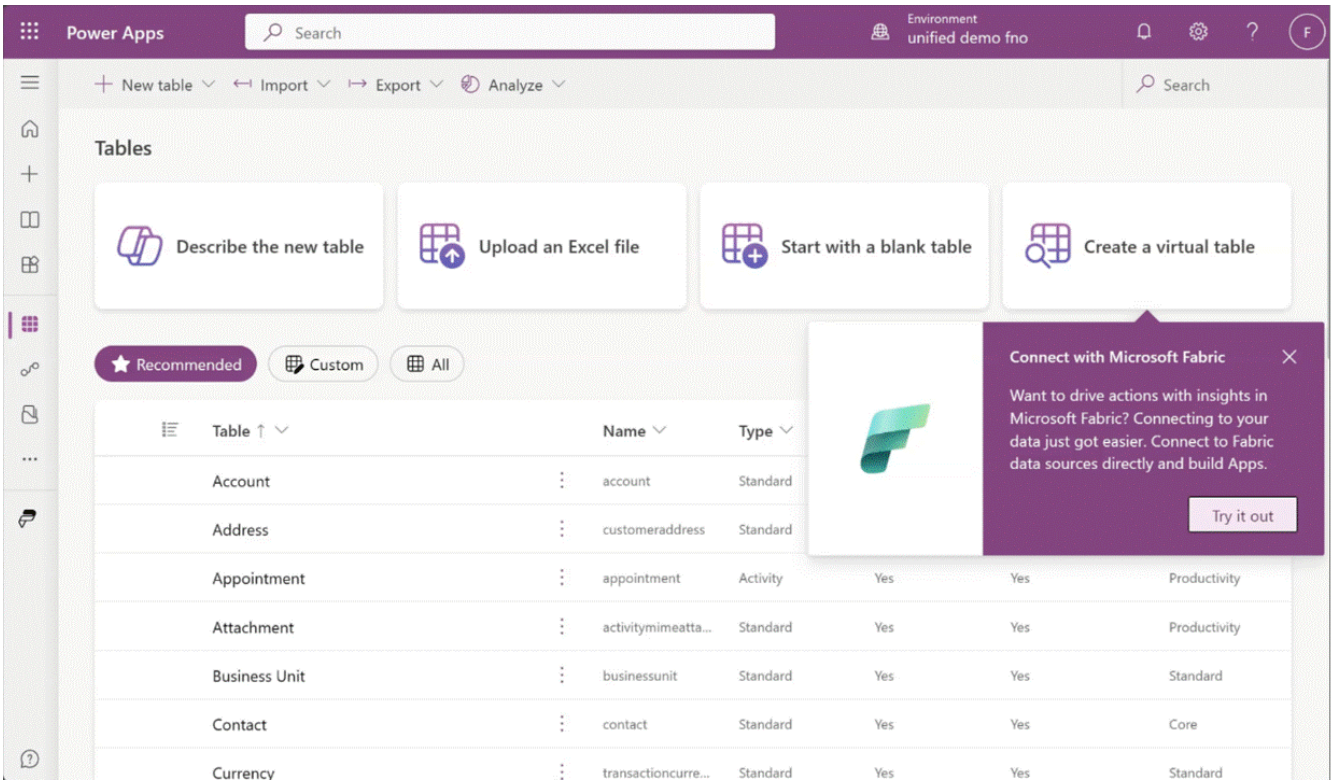
Build apps and automations, and drive action with insights from Microsoft Fabric

Link to Microsoft Fabric in Microsoft Dataverse enables extending your data and insights in Dynamics 365 and Power Apps in Fabric. Bring your own data into Fabric and combine, reshape, and aggregate data with data from Dataverse. Use Fabric tools such as SQL, Spark, and dataflows to work with your data within Fabric. For example:

- Combine financial data from Dynamics 365 with financial data from other systems to derive consolidated insights.
- Merge historical data ingested into Microsoft OneLake from legacy systems with current business data from Dynamics 365 and Dataverse.
- Combine weblogs and telemetry data from your website with product and order details from Dynamics 365.
- Apply machine learning to detect anomalies and exceptions within your data.

Insights aren't complete unless you can drive action and business processes. Bring insights you found in Fabric to build apps, drive business processes with Power Automate without data copy, no-ETL (extract, transform, load), or non-Microsoft integration tools.

With Dataverse virtual tables sourced with Fabric, your low-code app builders connect to data in OneLake and build Power Apps and drive business actions. Additionally, with Power Pages, low-code makers can build external facing websites and drive action from OneLake insights with partners, suppliers, and customers.



Prerequisites

- An existing Dataverse environment or you can create a new developer environment. More information: Create a developer environment
- Contributor or administrator access to a Fabric workspace with data.

If you don't have Power BI premium license or Fabric capacity, you can sign up for a free Fabric trial capacity.

If you don't have a workspace with data, create a workspace with sample data using the lakehouse tutorial.

Create a virtual table with Fabric data

Follow these steps to create a virtual table using Fabric data.

> **Note**
>
> This feature is enabled by default with all environments. Admins can disable this feature in the Power Platform admin center in the environment feature settings.

1. Sign into Power Apps, and then select **Tables** on the left navigation pane.

2. On the command bar, select **New table** > **Create a virtual table**.

3. Select **Microsoft Fabric** as the connection, and then select **Next**.
4. Select a **Microsoft Fabric workspace** from the available list of workspaces. All workspace where you have access are available in the list.
5. Choose a **Microsoft Fabric Lakehouse** from the drop-down list. All lakehouses and data warehouses within the workspace selected previously are available to choose.
6. On the **Data** page, select a table from the list of tables shown from the lakehouse. Select **Configure table and column names that will be used in Dataverse** if you want to rename any of the fields or tables in Dataverse or to specify additional properties available for you in Dataverse. Select **Next**.
7. If you selected the **Configure table and column names that will be used in Dataverse** checkbox, you're shown the table and column properties and you can modify them here. When you're done modifying them select **Next**.
8. Verify that the field chosen for **primary key field** is a unique field. You might need to verify the default field selected by the system as the primary key.

   > ⓘ **Note**
   >
   > While the table you selected from Fabric Lakehouse might not have a primary key defined, you need to select a field that is unique to continue. If you don't select a unique field, the table might not show all the records. While the wizard selects a field based on metadata available, the selection might not be accurate.

9. Review the information shown, and then select **Finish**.

A new Dataverse virtual table is created with data from Fabric.

Building apps and taking action with insights

You can build an app with the virtual table sourced with Fabric data in Power Apps (make.powerapps.com). From the **Home** page, select **Start with data** > **Select an existing table**, select the virtual table created earlier, and then select **Create app**. Provide a name for your app and build the app by following the wizard.

> ⓘ **Note**

Virtual tables created with data from Microsoft Fabric OneLake are read-only. Currently, you can't modify the data in Fabric OneLake with Power Apps.

See also

Create and edit virtual tables that contain data from an external data source

---

Last updated on 04/09/2026

Microsoft Fabric Link for Dataverse FAQ

This article provides information about frequently asked questions when you link Microsoft Dataverse to Microsoft Fabric.

General

What authentication methods are supported for Fabric link?

Fabric link supports three authentication methods when creating a connection:

- **Organizational Account**: Provide your credentials and save the connection.
- **Service Principal**: Provide Tenant ID, Client ID, and Key for authentication.
- **Workspace Identity**: Uses a Fabric-managed service principal bound to a specific workspace.

What happens when I unlink from Fabric?

When you unlink, the Fabric lakehouse is removed along with all shortcuts to your Dataverse tables.

Table support and limitations

Which Dataverse tables are selected by default when linking to Fabric?

During the initial **Link to Fabric** setup, the **Select Entities** step shows all Dataverse tables where the **Track changes** property is enabled. By default, all these tables are selected. You can clear any tables you don't want to sync before completing the wizard. Only selected tables are linked to Fabric and consume storage, so you can optimize costs by excluding tables you don't need. You can change your selection anytime after setup by using **Manage Tables**.

Which Dataverse tables aren't supported for Fabric link?

Any table that doesn't have change tracking enabled isn't supported. Additionally, certain Dataverse tables, such as postcomment, postregarding, postlike, post, and postrole, aren't available for customers to enable synchronization through Fabric link. In some cases, these tables might appear when Dataverse long term retention is enabled. If that happens, a subset or all records from these tables might be exported. This is expected behavior for long term retention scenarios.

> **Important**
>
> - These tables shouldn't be selected for synchronization.
> - Their presence doesn't indicate full support for incremental sync.

Can I add or remove specific tables from Fabric link?

Yes, you can manage which tables are linked:

- Add more table links to Fabric by selecting **Manage tables**.
- To stop syncing specific tables, clear them in the **Manage tables** pane and then select **Save**
- After confirmation, shortcuts for unselected tables are removed from the Fabric Lakehouse and sync stops immediately.

> **Important**
>
> Removing a table doesn't delete the table in Dataverse; it only removes the OneLake shortcut and stops data sync.

Are elastic tables supported for Fabric link?

Yes, elastic tables are supported with Fabric link. Ensure that change tracking is enabled for elastic tables in Dataverse for the sync to happen.

Are Finance and Operations apps tables supported?

If your environment is linked to a Dynamics 365 finance and operations environment, the add tables option enables you to include tables from finance and operations apps. Finance and operations apps tables aren't autoselected when creating the initial Fabric link.

Are there system tables that are automatically synchronized and can't be unlinked?

Yes, certain system tables are automatically synchronized in Link to Fabric and can't be unlinked by users. Examples of these system tables include:

- bot
- msdyn_copilotknowledgeinteraction
- desktopflowmodule
- flowmachine
- flowmachinegroup
- workflow

These tables are essential for system functionality and are always included in the synchronization process.

Microsoft provided add-ins and table selection

Fabric link table selection is also dependent on Microsoft provided add-ins such as Business Process Advisor (BPA) and Customer Insights - Journeys. These products use the same pipeline to export data from Dataverse to provide transformed insights.

The tables that you can't unselect depends on which add-on products you enabled. When you enable an add-in (for example, Customer Insights - Journeys), the specific tables required by that add-in are added to the entityanalyticsconfig table in Dataverse.

Any table that has **isenabledforadls** set to **true** in the entityanalyticsconfig table can't be selected from the Fabric link.

What happens when I add a new column to a linked table?

If a new column is added to a table that's already part of the profile (metadata change), you can use the **Refresh Fabric tables** option from the command bar to update the change in Fabric. The update occurs after the next table data change is triggered.

Data Synchronization

How long does initial data synchronization take?

Initial sync time depends on several factors including table size, number of updates and transactions happening in the environment, number of tables, and row count. The data is converted to delta parquet format during this process. When you open Fabric Lakehouse, the links appear as **unidentified** until the initial sync is completed.

How often is data refreshed in Fabric?

When the initial sync is complete, the system continuously refreshes updates in Dataverse in the lakehouse. Incremental updates typically take up to one hour to appear, though actual timing depends on table size, number of updates and transactions happening in the environment, number of tables, row count, and system load periods.

Why am I seeing differences in `SinkModifiedOn` and `ModifiedOn` columns?

This is expected behavior. `ModifiedOn` is the date and time a record was last changed in Dataverse, while `SinkModifiedOn` is the date and time the record was written to the Dataverse data lake from where Fabric shortcuts point to. `SinkModifiedOn` is typically later than `ModifiedOn`, and the difference depends on factors such as batch size, system load, and the number of records being processed.

> **Important**
>
> In the case of long-running transactions in finance and operations apps, a record might be synchronized multiple times to ensure no data loss. In this scenario, `SinkModifiedOn` reflects the last synchronization time, not the first. For example, if a record is synced 10 times within an hour, `SinkModifiedOn` shows the time of the 10th sync even though the data was available from the first sync itself. Use caution when building latency reports based on the difference between `SinkModifiedOn` and `ModifiedOn`, as it can give a misleading impression of actual data availability.

Limitations and Errors

What are the limits for Fabric Link?

If you have more than 2,000 active Dataverse tables, Link to Fabric can fail with an error.

Can I link existing Azure Synapse Link profiles to Fabric?

You can link your existing Azure Synapse Link for Dataverse profiles with Fabric if you select the **Enable Parquet/Delta lake** option. However:

- Existing profiles where the data is saved as CSV files can't be linked to Fabric.
- Azure Synapse Link profiles secured with managed identities currently can't be linked to Fabric.

See also

Link to Microsoft Fabric

Troubleshooting common issues with link to Fabric

Work with Dataverse data and generate Power BI reports

Troubleshooting common issues with link to Fabric

If you experience an error message when using link to Fabric with Microsoft Dataverse, here are suggestions and more information about how to resolve the issue.

Expand table

Error message	How to resolve
You must have Power BI premium or Fabric capacity in the same region {Region}. You can also get a Fabric trial. You won't receive this error after April 30, 2024. Instead, you're shown an error if you don't have capacity in the same geography.	You need a Power BI premium of a Fabric capacity in the same region as your Dataverse environment. Power BI premium per user capacity isn't sufficient. You can get a free trial capacity by visiting Fabric (preview) trial. More information: Prerequisites
Creation of Fabric workspace failed. You can try again. If this issue persists contact Microsoft customer support with the corelation ID.	You must be a Power BI Capacity Administrator or have contributor access to a capacity within the same geography as your Dataverse environment. Currently, the system supports these premium capacity SKUs described in the Prerequisites. Verify with your Power BI Tenant admin that you have permissions to create workspaces. You can find this setting in Power BI Admin portal under **Tenant settings** > **workspace settings** > **Create workspaces**. If the issue isn't resolved, contact Microsoft customer support with the provided reference ID
Creation of Fabric lakehouse failed. You can try again. If this issue persists contact Microsoft customer support with the corelation ID.	Verify with your Power BI Tenant admin that you have permissions to create OneLake shortcuts. You can find this setting in Power BI Admin portal under **Admin Portal** > **Tenant Settings** > **Microsoft Fabric** > **Users can create Fabric items**. More information: Prerequisites If the issue isn't resolved after several retries, you can contact Microsoft customer support with the provided reference ID.
We ran into an issue, Creaton of Fabric Lakehouse failed.	Verify with your Power BI tenant admin that you have permissions to create artifacts in Fabric. You can find this setting in Power BI admin portal under **Tenant settings** > **oneLake settings** > **Users can access data stored in OneLake with apps external to Fabric**.

Error message	How to resolve
	More information: Prerequisites If the issue isn't resolved after several retries, you can contact Microsoft customer support with the provided reference ID.
Your organization doesn't appear to have Microsoft Fabric. You can get a trial.	Contact your administrator or get a trial version of Microsoft Fabric.
You need to get a trial version of Fabric to get started.	You need a Power BI premium of a Fabric capacity. Power BI premium per user capacity isn't sufficient. You can get a free trial capacity by visiting: Fabric (preview) trial
You need to be a system administrator to link to Fabric.	You need the system administrator security role in Dataverse to perform this operation. More information: Security roles and privileges
Newly added fields in tables aren't reflected in Fabric Lakehouse.	When a new field is added to a Dataverse table, the newly added column isn't added to Fabric Linked workspaces automatically. To include newly added columns, select **Synapse Link** in Power Apps (make.powerapps.com), select the Microsoft OneLake or the Azure Synapse Link profile and select **Refresh Fabric tables**. This action refreshes the table metadata in Fabric Lakehouse.
Error message "Unauthorized. Access to target location https://[...].crm3.dynamics.com/ denied" when accessing Dataverse tables in Fabric Lakehouse.	This error message indicates that the user accessing tables in Fabric doesn't have the required permissions to access Dataverse tables. This error might be shown even if the same user has access to the tables in Dataverse. The Link to Fabric wizard in Power Apps creates a data connection at the time of creating the Link to Fabric. This data connection uses credentials of the user, it's possible that: - The password of the user who created the connection has changed or has expired. - The user account of the user who created the Fabric link is inactive. - Other users who need to access Dataverse tables in Fabric Lakehouse don't have access to the data connection. More information: Share the data connection with other users
Error message "A back end error occurred. The Fabric to Dataverse connection ID ... is not valid for this user. Please check that you have access to this connection, and that the connection is connected	This error message indicates that the user selecting the **Refresh Fabric tables** option in the Azure Synapse Link page doesn't have the required permissions. This error might be shown even when the user has access to the tables in Dataverse and is a system

Error message	How to resolve
to this organization, with URL ... You can try again, if this issue persists please contact support.	administrator. The link to Fabric wizard in Power Apps creates a data connection at the time of creating the link to Fabric. This data connection uses credentials of the user, it's possible that: - The password of the user who created the connection has changed or has expired. - The user account of the user who created the Fabric link is inactive. - The user performing the **Refresh Fabric tables** option doesn't have access to the data connection. More information: Share the data connection with other users
All tables from Dataverse show as **Unidentified** in Fabric Lakehouse.	You might see this error message while your data is being initialized for the first time. If this issue persists for more than several hours, go to **Synapse Link** in Power Apps (make.powerapps.com), choose the Microsoft OneLake or the Azure Synapse Link profile and select **Refresh Fabric Links**. This refreshes the table metadata in Fabric Lakehouse.
Tables from Dynamics 365 finance and operations apps are missing.	System auto selects nonsystem tables with **Change tracking** property set to **Yes** when creating a Fabric Link. To select more tables, open **Microsoft OneLake** profile in **Synapse Link** and select **Manage Tables**. If you have a finance and operations environment linked to this Power Platform environment, you can also select tables from finance and operations apps. Finance and operations apps tables aren't autoselected with Fabric Link. More information: Add Finance and Operations tables
There are {.. #tables} tables enabled for change tracking in your environment. Addition of more than 2,000 tables isn't supported.	Fabric link feature currently can't be enabled on environments with more than 2,000 change tracking enabled tables.

See also

Link your Dataverse environment to Microsoft Fabric and unlock deep insights

Link to Microsoft Fabric

Last updated on 04/27/2026

What is Azure Synapse Link for Dataverse?

Azure Synapse Link for Dataverse enables you to get near real-time insights over your data in Microsoft Dataverse. With a tight seamless integration between Dataverse and Azure Synapse Analytics, Azure Synapse Link enables you to run analytics, business intelligence and machine learning scenarios on your data.

> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed May 2021 and continues to export data to Azure Data Lake as well as Azure Synapse Analytics.

Using Azure Synapse Link, continuously export data from Dataverse to:

- Azure Synapse Analytics
- Azure Data Lake Storage Gen2

The Azure Synapse Link for Dataverse is a service designed for enterprise big data analytics by delivering scalable high availability with disaster recovery capabilities. Data is stored in the Common Data Model format, which provides semantic consistency across apps and deployments.



The Azure Synapse Link for Dataverse provides these features:

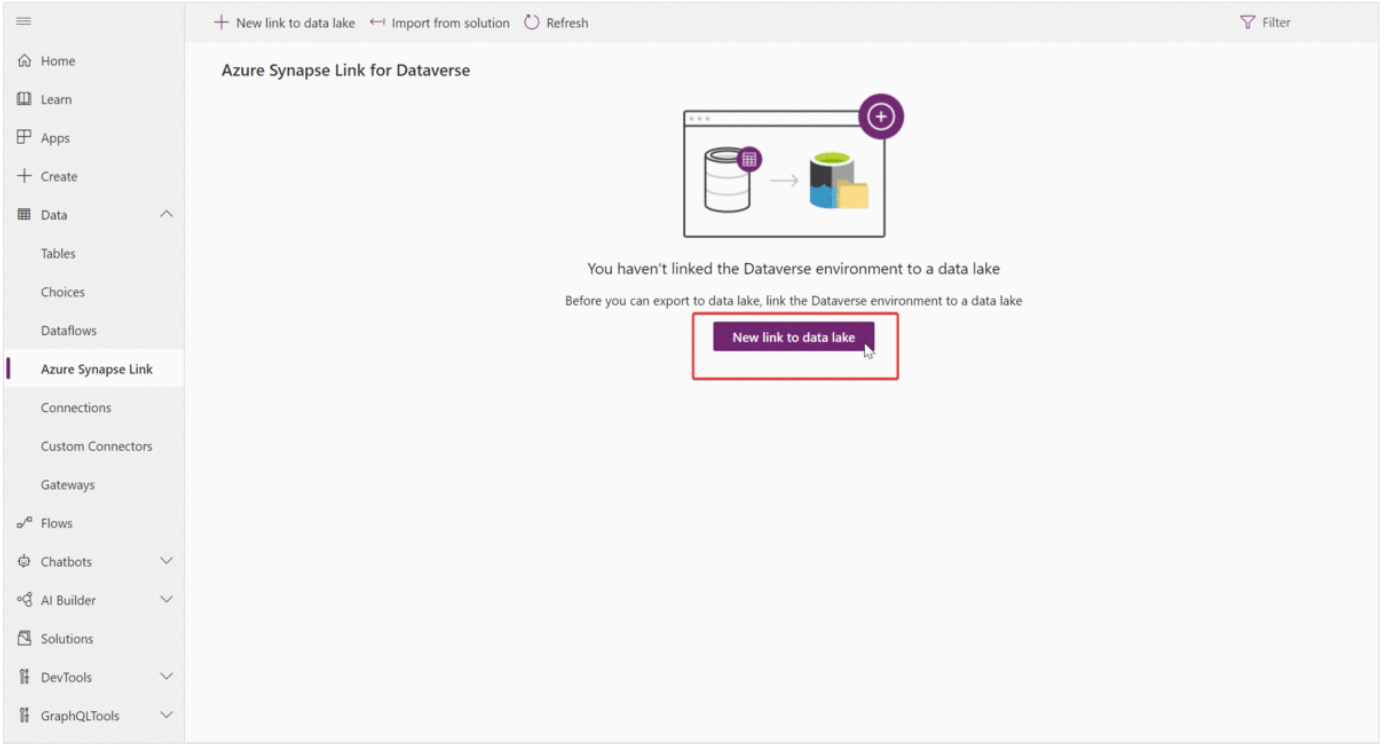
- Linking or unlinking the environment to Azure Synapse Analytics and/or Azure Data Lake Storage Gen2 in your Azure subscription.
- Continuous replication of tables to Azure Synapse Analytics and/or Azure Data Lake Storage Gen2 in your Azure subscription.
- Replication of both standard and custom tables as well as create, update, and delete (CUD) transactions.
- Fewer clicks to process, transform, and visualize your data in Azure Synapse Analytics.

- Serverless data lake exploration, data integration, and big data processing in Azure Synapse Analytics.
- Facilitated metadata discovery and interoperability between data producers and consumers such as Apache Spark, Power BI, Azure Data Factory, Azure Databricks, and Azure Machine Learning.

The Azure Synapse Link for Dataverse service supports initial and incremental writes for table data and metadata. Any data or metadata changes in Dataverse are automatically pushed to the Azure Synapse metastore and Azure Data Lake, depending on the configuration, without any additional action. This is a push, rather than pull, operation. Changes are pushed to the destination without you needing to set up refresh intervals.

Both standard and custom tables can be exported. Notice that the change tracking table attribute in Dataverse is used to keep the data synchronized in an efficient manner by detecting what data has changed since it was initially extracted or last synchronized.

All create, update, and delete operations are exported from Dataverse to the data lake. For example, when a user deletes an Account table row in Dataverse, the transaction is replicated in the destination location.



Rapidly analyze your Dataverse data in Azure Synapse Analytics

Azure Synapse Analytics is a limitless analytics service that brings together data integration, enterprise data warehousing, and big data analytics. It gives you the freedom to query data on your terms, using either serverless or dedicated resources—at scale. It also allows you to build analytics solutions on top of the Apache Spark engine. With Azure Synapse you can ingest, explore, prepare, manage, and serve data for immediate business intelligence and machine learning needs all from a single service.

Flexibly analyze your Dataverse data with Azure Data Lake Storage Gen2

Azure Data Lake is a storage solution that allows you to process data on demand, scale instantly, and only pay per job. It provides enterprise-grade security, auditing and support. Azure Data Lake is a great landing page for your Dataverse data before being utilized in another service or application. Each business is different and has different scenarios; Azure Data Lake gives you the flexibility to customize consumption and transition your data to other destinations.

See also

Configure Azure Synapse Link for Dataverse with your Azure Synapse Workspace

Configure Azure Synapse Link for Dataverse with Azure Data Lake

Blog: Announcing Azure Synapse Link for Dataverse

Blog: Exporting CDS data to Azure Data Lake

Last updated on 04/27/2026

Create an Azure Synapse Link for Dataverse with your Azure Synapse Workspace

Summarize this article for me

You can use the Azure Synapse Link to connect your Microsoft Dataverse data to Azure Synapse Analytics to explore your data and accelerate time to insight. This article shows you how to perform the following tasks:

1. Connect your Dataverse data to your Azure Synapse Analytics workspace with the Azure Synapse Link service.
2. Manage Dataverse tables included in the Azure Synapse Link.
3. Monitor your Azure Synapse Link.
4. Unlink your Azure Synapse Link.
5. Relink your Azure Synapse Link.
6. View your data in Azure Synapse Analytics.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

- Dataverse: You must have the Dataverse **system administrator** security role.
- The tables you want to export via Azure Synapse Link must have the **Track changes** property enabled. More information: Advanced options
- The Azure Synapse Workspace that you link with can't have data exfiltration protection enabled. More information: Create a workspace with data exfiltration protection enabled

Azure Data Lake Storage Gen2 requirements

You must have an Azure Data Lake Storage Gen2 account and the following roles:

- **Owner**
- **Storage Blob Data Contributor**
- **Storage Blob Data Owner**

Owner role requirement: The Owner role is required because adding the managed identity of the Azure Synapse Link service to the storage account is a privileged operation. This managed identity enables the service to perform synchronization from Dataverse to Synapse. Currently, the Owner role is required and custom roles with similar privileges aren't supported for this step.

Privileges for managed identity:

- **Owner and Storage Account Contributor:** Required only during first-time setup to create the filesystem in the storage account and are assigned on the storage account. After the setup, these privileges can be removed without impacting sync operations.
- **Storage Blob Data Contributor:** Required for normal blob operations during regular sync.
- **Storage Blob Data Owner:** Currently required for managing both data and access permissions in scenarios involving Azure Data Lake Storage Gen2.

Your storage account must enable **Hierarchical namespace** for both initial setup and delta sync.

Allow storage account key access is required only for the initial setup because the service uses Shared Key authorization to perform privileged operations such as creating the filesystem and establishing the link before switching to managed identity authentication. After setup, key-based access is no longer needed.

Your storage account must have permitted scope for copy operations set to **From any storage account**.

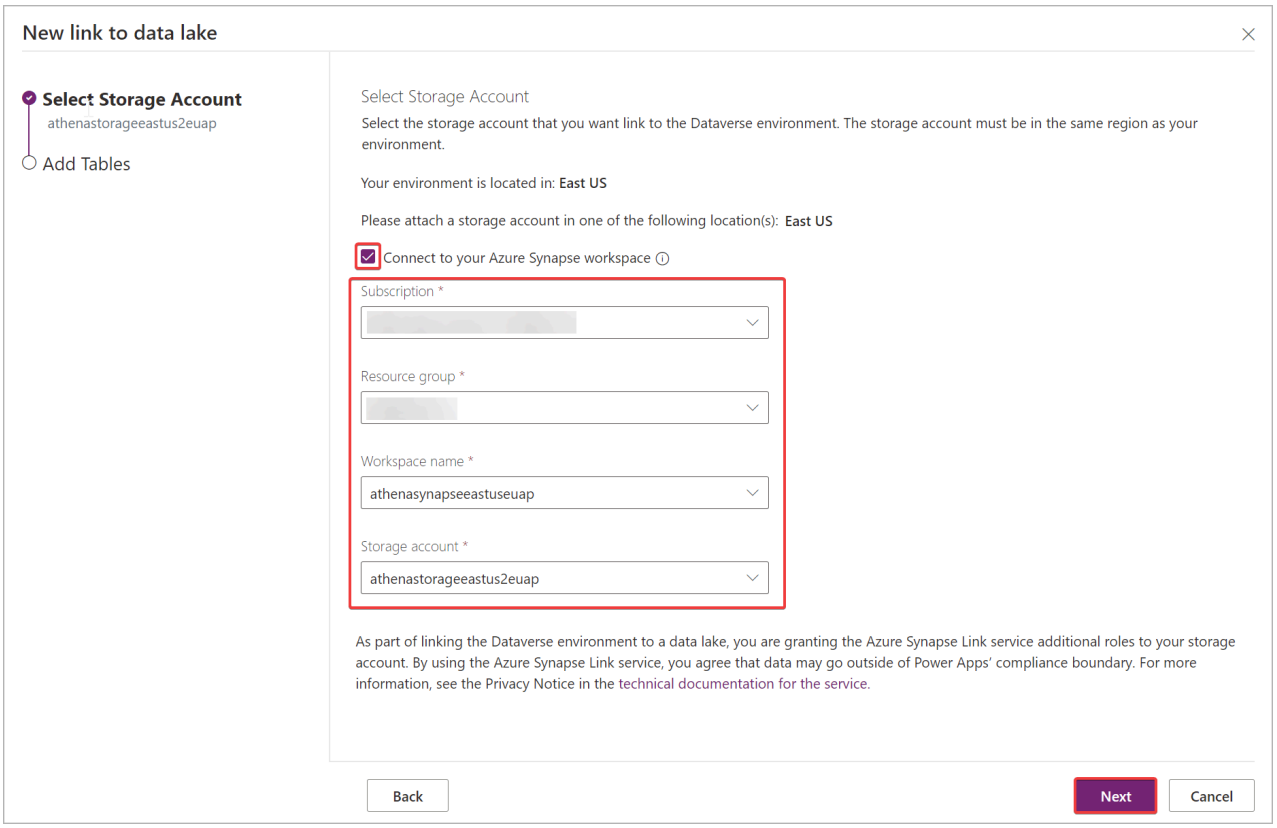
- Synapse workspace: You must have a Synapse workspace and the **Synapse Administrator** role access within the Synapse Studio. The Synapse workspace must be in the same region as your Azure Data Lake Storage Gen2 account. The storage account must be added as a linked service within the Synapse Studio. To create a Synapse workspace, go to Creating a Synapse workspace.

> ⓘ **Note**
>
> - The storage account and Synapse workspace must be created in the same Microsoft Entra tenant as your Power Apps tenant.
> - To set **Enabled from selected virtual networks and IP addresses** for linked storage account and workspace, you must create an Azure Synapse Link with managed identities. More information: Use managed identities for Azure with your Azure data lake storage (without managed identities set up, you must enable **public network access** for Azure resources for both initial setup and delta sync.)

- Synapse workspaces featuring managed private endpoints, data exfiltration protection, or managed virtual networks aren't supported.
- You must have **Reader** role access to the resource group with the storage account and Synapse workspace.
- When you add multiple users to the synapse workspace, they must have the **Synapse Administrator** role access within the Synapse Studio and the **Storage Blob Data Contributor** role on the Azure Data Lake Storage Gen2 account.
- The creation of Synapse Link profiles under a single DV environment is limited to a maximum of 10.

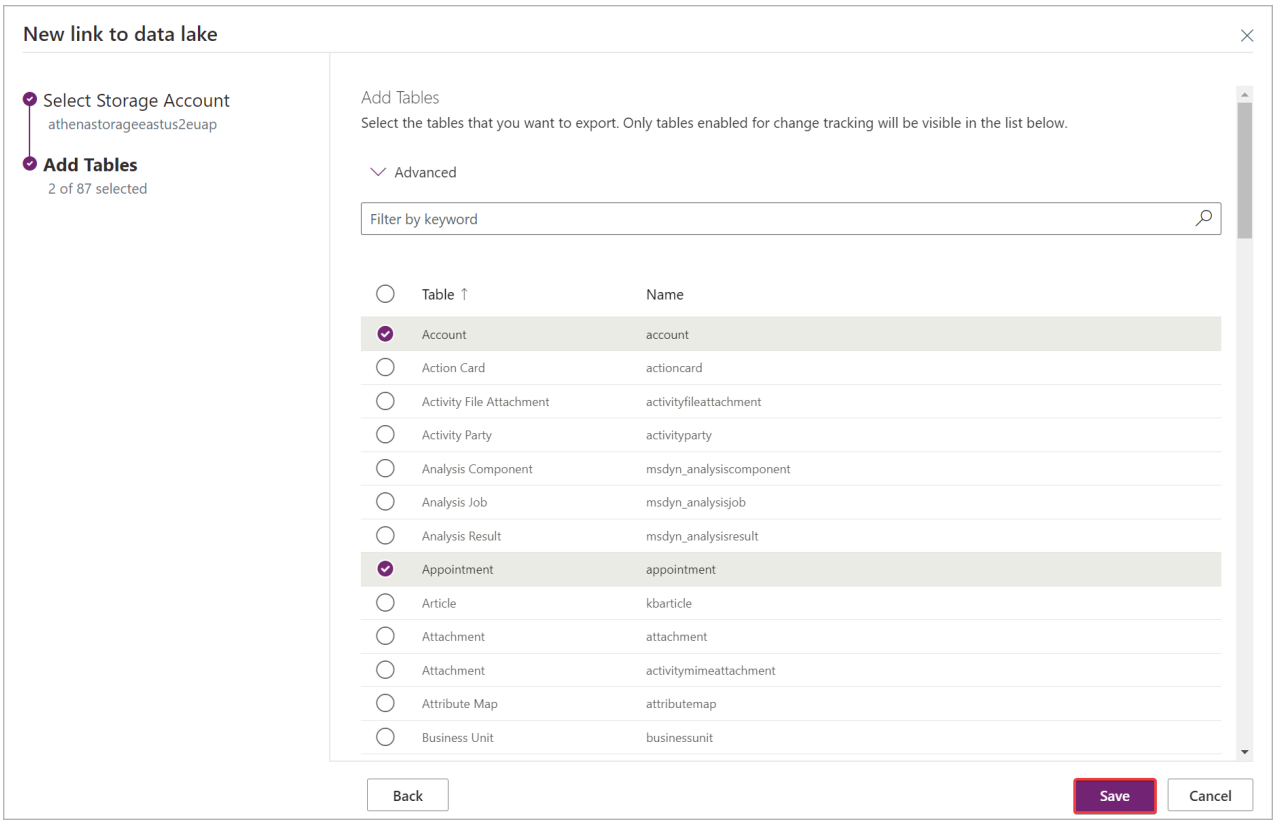
Connect Dataverse to Synapse workspace

1. Sign in to Power Apps and select your preferred environment.
2. On the left navigation pane, select **Azure Synapse Link**. If **Azure Synapse Link** isn't visible in the side pane, select **...More** and choose **Discover all**. Select **Azure Synapse Link** in the **Data Management** section.
3. On the command bar, select **+ New link**.
4. Select the **Connect to your Azure Synapse workspace** option.
5. Select the **Subscription**, **Resource group**, **Workspace name**, and **Storage account**. Ensure that the Synapse workspace and storage account meet the requirements specified in the Prerequisites section. Select **Next**.



> **Note**
>
> As part of linking the environment to a data lake, you grant the Azure Synapse Link service access to your storage account. Ensure that you followed the prerequisites of creating and configuring the Azure data lake storage account, and granting yourself an owner role on the storage account. Additionally, you grant the Power Platform Dataflows service access to your storage account. More information: Self-service data prep with dataflows.

6. Choose the tables you want to export either by selecting them one by one or by entering a comma separated list of tables in the search box, and then select **Save**. Only tables with the Track changes property enabled can be exported. More information: Advanced options.



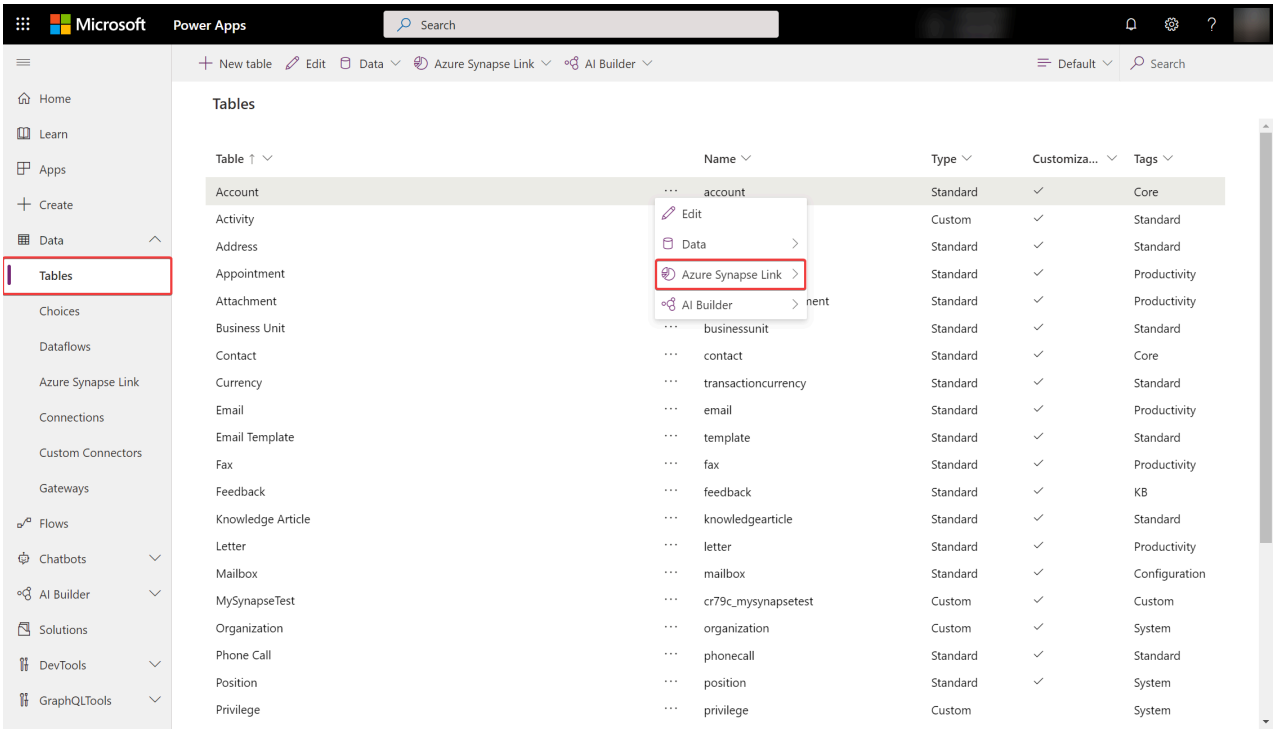
You can follow the steps above to create a link from one environment to multiple Azure Synapse Analytics workspaces and Azure data lakes in your Azure subscription by adding an Azure data lake as a linked service on a Synapse workspace. Similarly, you could create a link from multiple environments to the same Azure Synapse Analytics workspace and Azure data lake, all within the same tenant.

> **Note**
>
> The Azure Synapse Link for Dataverse service is seamlessly integrated into the Power Platform as an out-of-the-box feature. It meets the security and governance standards set for the Power Platform data storage and governance. More information: Data storage and governance
>
> The data exported by Azure Synapse Link service is encrypted at transit using Transport Layer Security(TLS) 1.2 or higher and encrypted at rest in Azure Data Lake Storage Gen2. Additionally, transient data in the blob storage is also encrypted at rest. Encryption in Azure Data Lake Storage Gen2 helps you protect your data, implement enterprise security policies, and meet regulatory compliance requirements. More information: Azure Data Encryption-at-Rest More information: Configure Azure Storage firewalls and virtual networks

Manage table data to the Synapse workspace

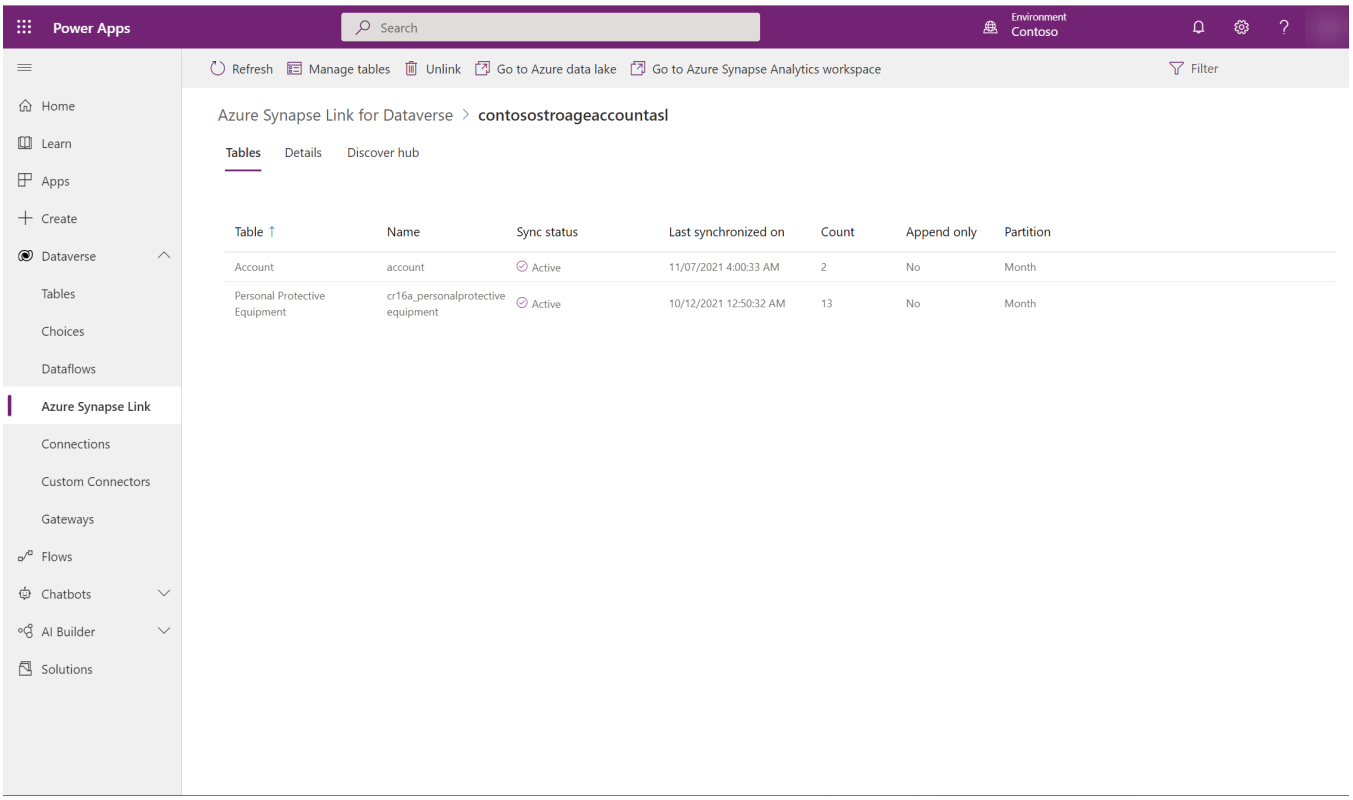
After you set up the Azure Synapse Link, you can manage the tables that are exported in one of two ways:

- In the Power Apps (make.powerapps.com) **Azure Synapse Link** area, select **Manage tables** on the command bar to add or remove one or more linked tables.
- On the Power Apps maker portal **Tables** area, select **...** next to a table, and then select the linked data lake where you want to export table data.



Monitor your Azure Synapse Link

After you set up the Azure Synapse Link, you can monitor the Azure Synapse Link on the **Tables** tab.



- There will be a list of tables that are a part of the selected Azure Synapse Link.
- There are different stages the sync status circulates through. **NotStarted** indicates that the table is waiting to be synced. Once the table initial sync is **Completed**, there's a post processing stage where incremental updates won't take place. This might take several hours depending on the size of your data. As the incremental updates start taking place, the date for the last sync will be regularly updated.
- The **Count** column shows the number rows written. When **Append only** is set to **No**, this is the total number of records. When **Append Only** is set to **Yes**, this is the total number of changes.
- The **Append only** and **Partition strategy** columns show the usage of different advanced configurations.

Unlinking an Azure Synapse Link

1. Select the desired Azure Synapse Link to unlink.
2. Select **Unlink** from the command bar.
3. To delete both the data lake file system as well as the Synapse Database, type the link name to confirm.
4. Select **Yes**, and allow a few minutes for everything to be unlinked and deleted.

> **Important**

The custom database objects you create within the Azure Synapse Link database are deleted once the unlink is completed.

You need to implement your own version control for your custom database objects scripts to ensure you have a backup.

Relinking an Azure Synapse Link

If you deleted the file system when unlinking, follow the steps above to relink the same Synapse workspace and data lake. If you didn't delete the file system when unlinking, you must clear the data to relink:

1. Navigate the Azure Synapse Analytics.
2. Select the **...** for the unlinked database and select **New notebook** > **Empty notebook**.
3. Attach the notebook to an Apache Spark pool by selecting a pool from the drop-down menu. If you don't have an Apache Spark pool, select **Manage pools** to create one.
4. Enter the following script, replace **<DATABASE_NAME>** with the name of the database to unlink, and run the notebook.

SQL

```
%%sql
DROP DATABASE <DATABASE_NAME> CASCADE
```

5. After running the notebook, refresh the database list from the left panel. If the database still exists, try right clicking on the database and selecting **Delete**.
6. Navigate to Power Apps, and relink the Synapse workspace and data lake.

Access near real-time data and read-only snapshot data

After creating an Azure Synapse Link, two versions of the table data will be synchronized in Azure Synapse Analytics and/or Azure Data Lake Storage Gen2 in your Azure subscription by default to ensure you can reliably consume updated data in the lake at any given time:

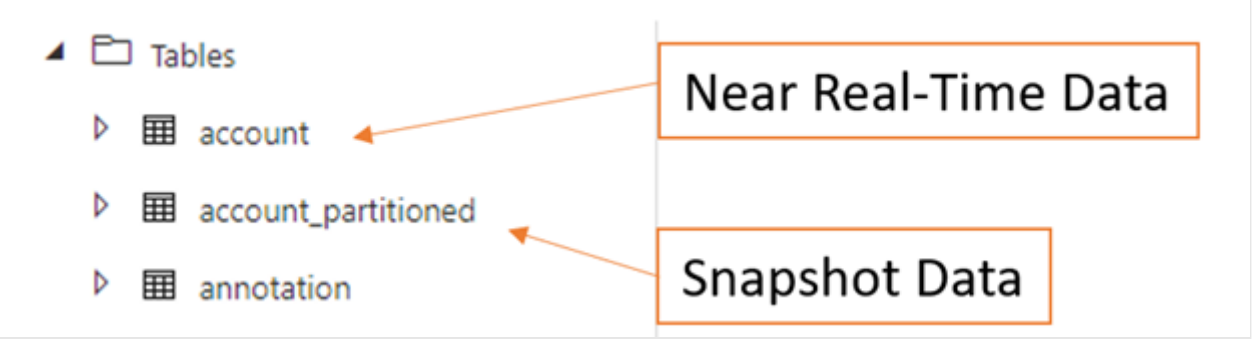
- Near real-time data: Provides a copy of data synchronized from Dataverse via Azure Synapse Link in an efficient manner by detecting what data has changed since it was initially extracted or last synchronized.

- Snapshot data: Provides a read-only copy of near real-time data that is updated at regular intervals (in this case every hour).

> **Note**
>
> - Near real-time sync performance depends on several factors including the initial data load size, data churn rate, and the volume of changes. In situations with high transaction volumes, such as processes in Finance and Operations apps generating millions of records in a short time, or processes like the master planning feature included with Dynamics 365 Supply Chain Management that delete and re-create large volumes of records. Synapse Link must synchronize all changes including deletes. In these high-volume scenarios, data availability in near real-time can't be guaranteed.
> - To create read-only snapshot data, ensure that the **Permitted scope for copy operations** setting is configured to **From any storage account**. More information: Configure the permitted scope for copy operations

1. Select the desired Azure Synapse Link, and then select the **Go to Azure Synapse Analytics workspace** from the top panel.
2. Expand **Lake Databases** from the left panel, select **dataverse**-*environmentName*-*organizationUniqueName*, and then expand **Tables**.

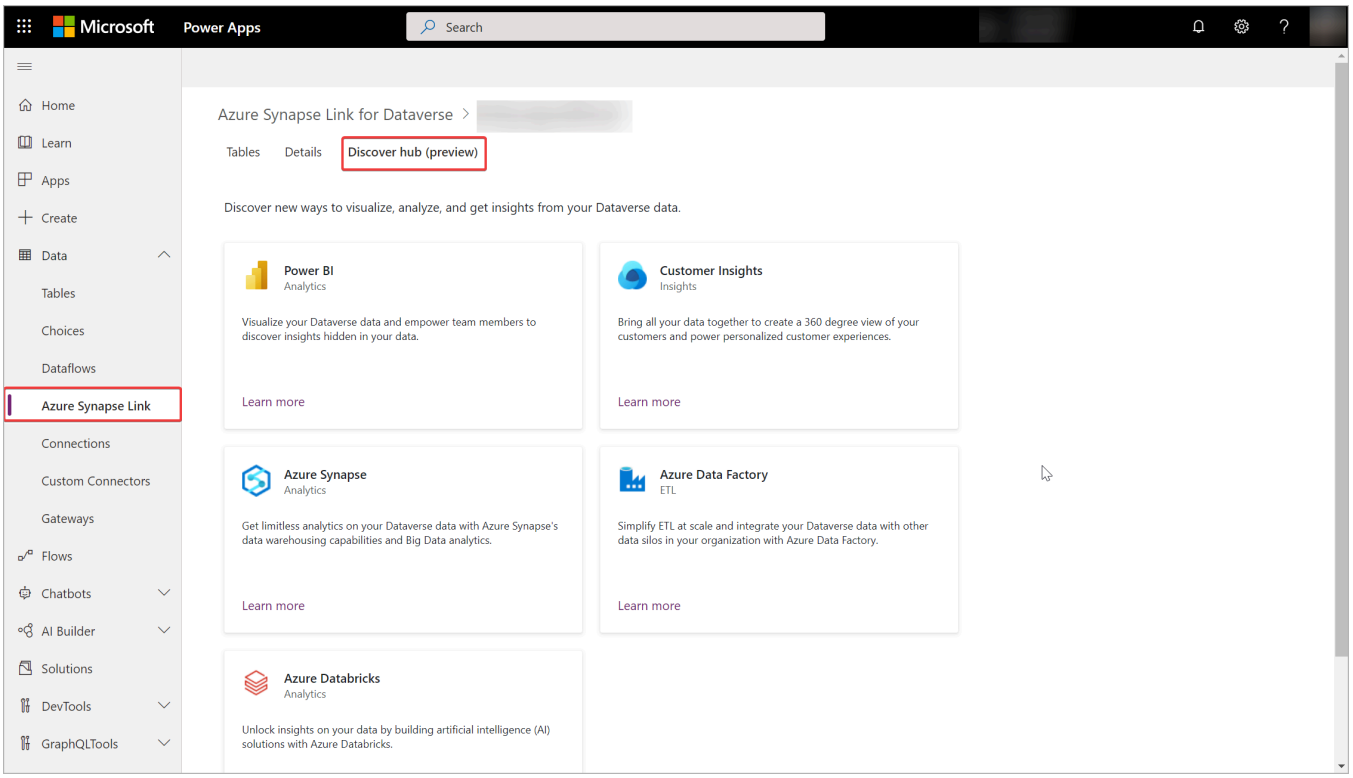
All **Near Real-Time Data Tables** are listed and available for analysis with the naming convention *DataverseTableName*. All **Snapshot Data Tables** are listed and available for analysis with the naming convention *DataverseTableName*_partitioned.



What's next?

After successfully using the Azure Synapse Link for Dataverse service, discover how you can analyze and consume your data with **Discover Hub**. To access **Discover Hub**, go to **Power Apps** > **Azure Synapse Link**. Select your linked service and then select the **Discover Hub** tab. Here you can find recommended tools and curated documentation to help you get the most value

out of your data.



See also

Configure Azure Synapse Link for Dataverse with your Azure Data Lake

Azure Synapse Link for Dataverse Advanced Configuration

Azure Synapse Link FAQ

Blog: Announcing Azure Synapse Link for Dataverse

Last updated on 02/18/2026

Create an Azure Synapse Link for Dataverse with Azure Data Lake

You can use the Azure Synapse Link to connect your Microsoft Dataverse data to Azure Data Lake Storage Gen2 to enable various analytics scenarios. This article shows you how to perform the following tasks:

1. Connect your Dataverse data to your Azure Data Lake Storage Gen2 account with the Azure Synapse Link service.
2. Manage Dataverse tables included in the Azure Synapse Link.
3. Monitor your Azure Synapse Link.
4. Unlink your Azure Synapse Link.
5. Relink your Azure Synapse Link.
6. View your data in Azure Data Lake and understand the file structure.

> ⓘ **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

- Azure Data Lake Storage Gen2: You must have an Azure Data Lake Storage Gen2 account and **Owner** and **Storage Blob Data Contributor** role access. Your storage account must enable **Hierarchical namespace** for both initial setup and delta sync. **Allow storage account key access** is required only for the initial setup. Your storage account must have Permitted scope for copy operations set to **From any storage account**.

> ⓘ **Note**
>
> - The storage account must be created in the same Microsoft Entra tenant as your Power Apps tenant.
> - To set **Enabled from selected virtual networks and IP addresses** for linked storage account to grant access from selected IP addresses, you must create an Azure Synapse

Link with managed identities.Use managed identities for Azure with your Azure data lake storage (without managed identities set up, you must enable public network access for Azure resources for both initial setup and delta sync.)

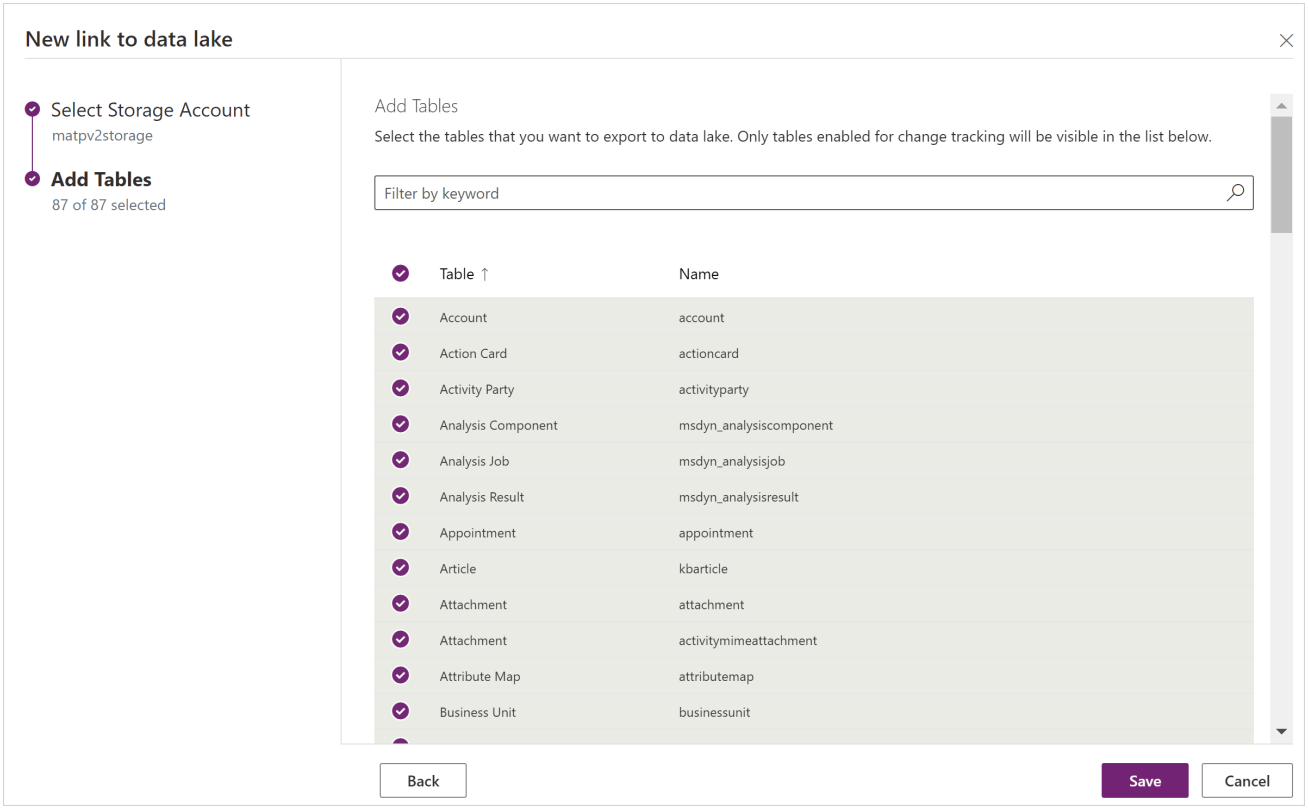
- We strongly recommend that you enable the soft delete feature on the storage account selected for this purpose. Enabling soft delete enables you to recover from accidental data deletes faster.
- You must have **Reader** role access to the resource group with the storage account.
- To link the environment to Azure Data Lake Storage Gen2, you must have the Dataverse system administrator security role.
- Only tables that have change tracking enabled can be exported.
- The creation of Azure Synapse Link profiles under a single Dataverse environment is limited to a maximum of 10.

Connect Dataverse to Azure Data Lake Storage Gen2

1. Sign in to Power Apps and select your preferred environment.
2. On the left navigation pane, select **Azure Synapse Link**. If **Azure Synapse Link** isn't visible in the side panel, select **...More** and select **Discover all**. **Azure Synapse Link** is in the **Data Management** section.
3. On the command bar, select **+ New link to data lake**.
4. Select the **Subscription**, **Resource group**, and **Storage account**. Ensure that storage account meets the requirements specified in the Prerequisites section. Select **Next**.

   > **Note**
   >
   > As part of linking the environment to a data lake, you grant the Azure Synapse Link service access to your storage account. Ensure that you followed the prerequisites of creating and configuring the Azure data lake storage account, and granting yourself an owner role on the storage account. Additionally, you grant the Power Platform Dataflows service access to your storage account. More information: Self-service data prep with dataflows.

5. Choose the tables you want to export either by selecting them one by one or by entering a comma separated list of tables in the search box, and then select **Save**. Only tables with the **Track changes** property can be exported. More information: Advanced options for tables.



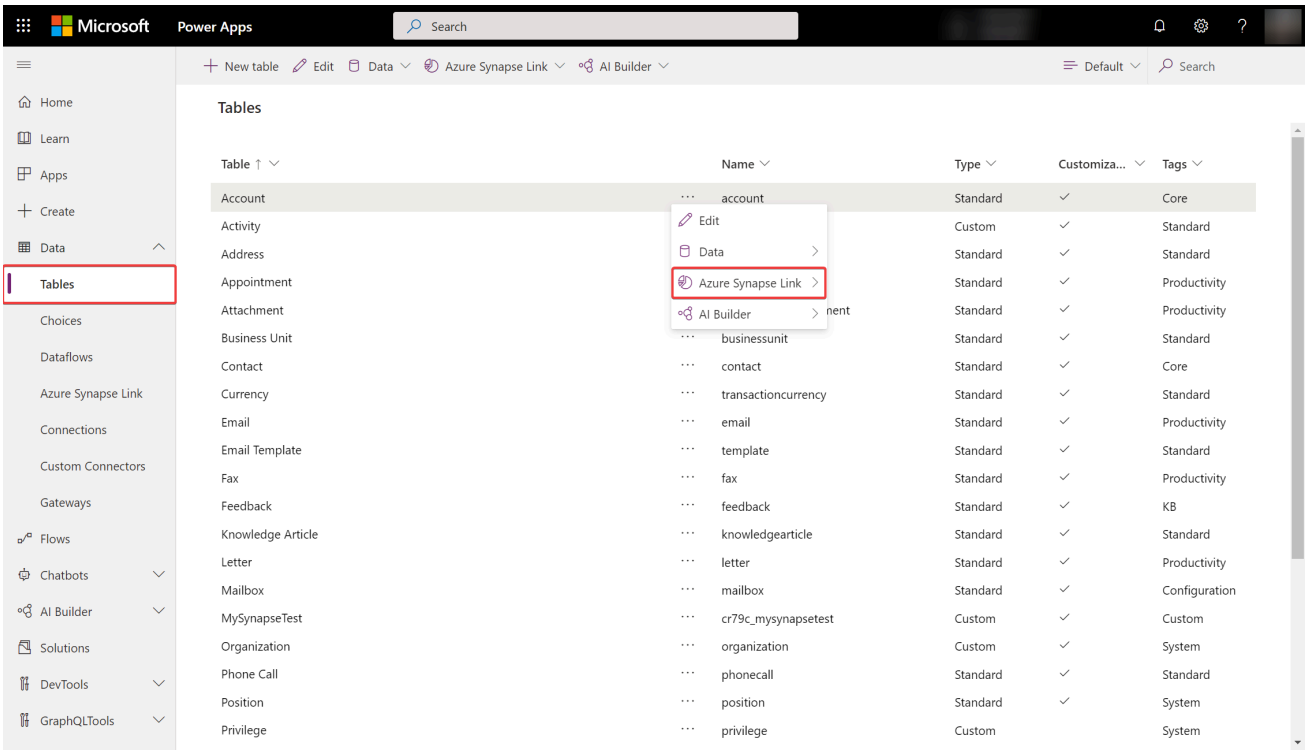
You can follow the steps in this article to create a link from one environment to multiple Azure data lakes in your Azure subscription. Similarly, you can create a link from multiple environments to the same Azure Data Lake, all within the same tenant.

> **Note**
>
> The Azure Synapse Link for Dataverse service is seamlessly integrated into the Power Platform as an out-of-the-box feature. It meets the security and governance standards set for the Power Platform data storage and governance. More information: Data storage and governance
>
> The data exported by Azure Synapse Link service is encrypted in transit by using Transport Layer Security (TLS) 1.2 or higher and encrypted at rest in Azure Data Lake Storage Gen2. Additionally, transient data in the blob storage is also encrypted at rest. Encryption in Azure Data Lake Storage Gen2 helps you protect your data, implement enterprise security policies, and meet regulatory compliance requirements. For more information, see Azure Data Encryption-at-Rest.

Manage table data to the data lake

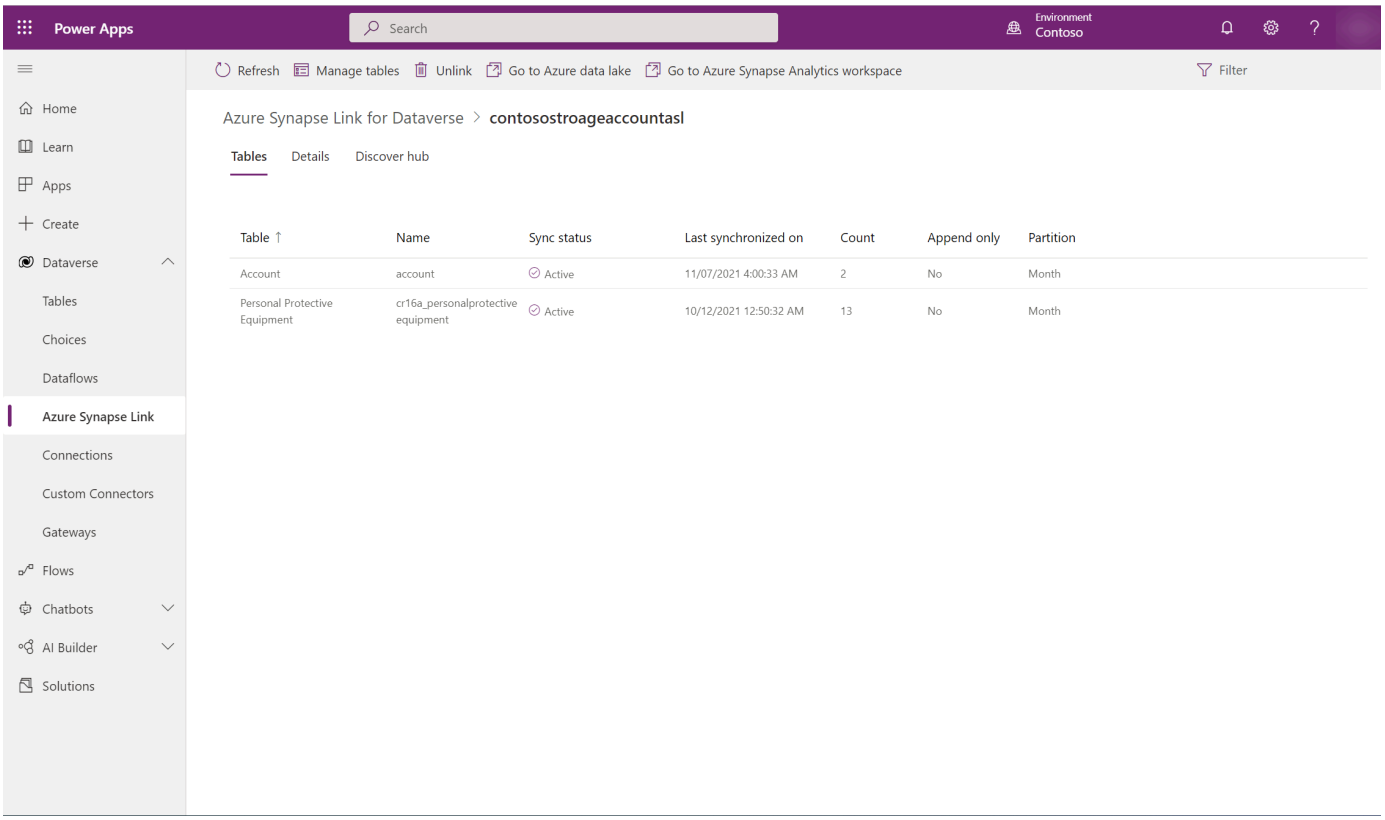
After you set up the Azure Synapse Link, you can manage the tables that are exported in one of two ways:

- On the Power Apps maker portal **Azure Synapse Link** area, select **Manage tables** on the command bar to add or remove one or more linked tables.
- On the Power Apps maker portal **Tables** area, select **...** next to a table, and then select the linked data lake where you want to export table data.



Monitor your Azure Synapse Link

After you set up the Azure Synapse Link, you can monitor the Azure Synapse Link under the **Tables** tab.



- There will be a list of tables that are a part of the selected Azure Synapse Link.
- There are different stages the sync status circulates through. **NotStarted** indicates that the table is waiting to be synced. Once the table initial sync is **Completed**, there's a post processing stage where incremental updates won't take place. This might take several hours depending on the size of your data. As the incremental updates start taking place, the date for the last sync will be regularly updated.
- The **Count** column shows the number rows written. When **Append only** is set to **No**, this is the total number of records. When **Append Only** is set to **Yes**, this is the total number of changes.
- The **Append only** and **Partition strategy** columns show the usage of different advanced configurations.

Unlinking an Azure Synapse Link

1. Select the Azure Synapse Link to unlink.

2. Select **Unlink data lake** from the command bar.

3. To delete both the data lake file system, select **Delete data lake file system**.

4. Select **Yes**, and allow a few minutes for everything to be unlinked and deleted.

Relinking an Azure Synapse Link

If you deleted the file system when unlinking, follow the preceding steps to relink the same data lake. If you didn't delete the file system when unlinking, you must clear the data to relink:

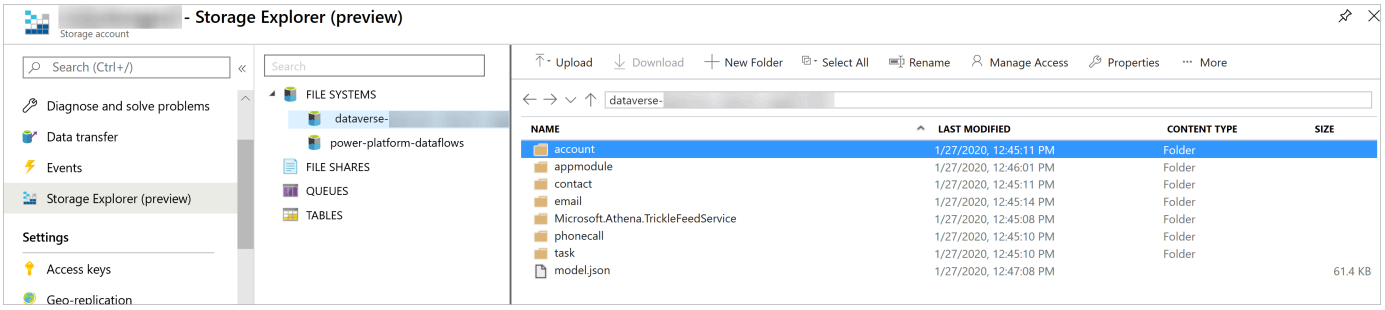
1. Go to the Azure Data Lake.
2. Delete the Dataverse container.
3. Go to Power Apps, and relink the data lake.

View your data in Azure Data Lake Storage Gen2

1. Select the desired Azure Synapse Link, and then select **Go to Azure data lake** from the top panel.
2. Expand **File Systems**, and then select dataverse-*environmentName*-*organizationUniqueName*.

The model.json file, along with its name and version, provides a list of tables that have been exported to the data lake. The model.json file also includes the initial sync status and sync completion time.

A folder that includes snapshot comma-delimited (CSV format) files is displayed for each table exported to the data lake.

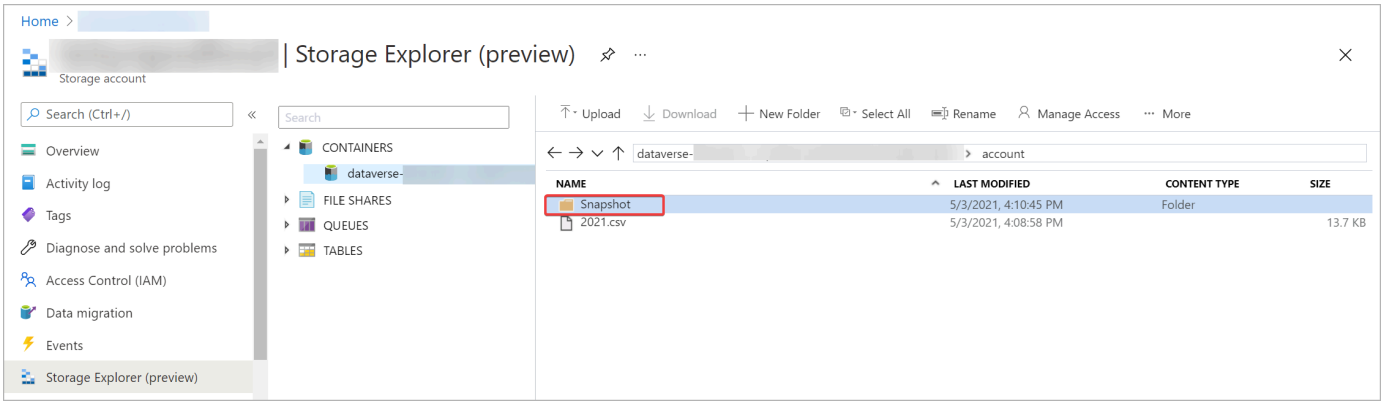


Link a Synapse workspace to an existing Azure Synapse Link profile with data lake only

1. In a web browser's address bar, append `?athena.updateLake=true` to the web address that ends with **exporttodatalake**.
2. Select an existing profile from the Azure Synapse Link area, and then select extended option.
3. Select **Link to Azure Synapse Analytics Workspace** and allow a few minutes for everything to be linked.

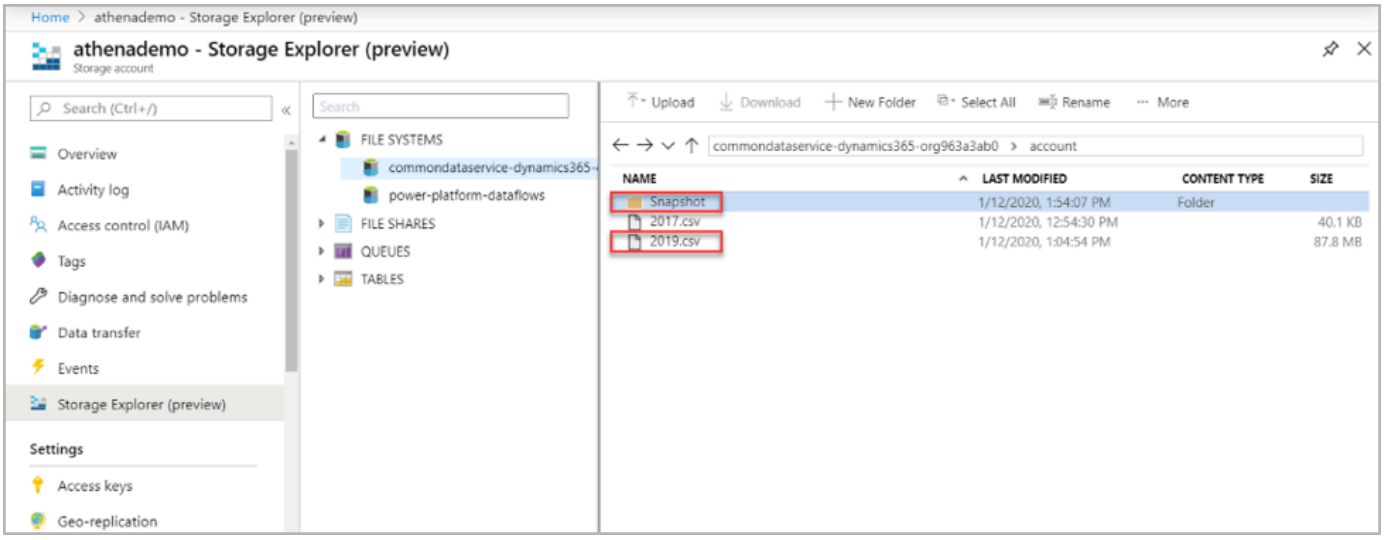
Continuous snapshot updates

Microsoft Dataverse data can continuously change through create, update, and delete transactions. Snapshots provide a read-only copy of data that's updated at regular intervals, in this case every hour. This ensures that at any given point, a data analytics consumer can reliably consume data in the lake.



When tables are added as part of the initial export, the table data is written to the table.csv files under the corresponding folders in the data lake. This is the T1 interval, where a snapshot read-only file named *table*-T1.csv—for example, Account-T1.csv or Contacts-T1.csv—is created. Additionally, the model.json file is updated to point to these snapshot files. Opening model.json, you can view the snapshot details.

Here's an example of an Account.csv partitioned file and snapshot folder in the data lake.

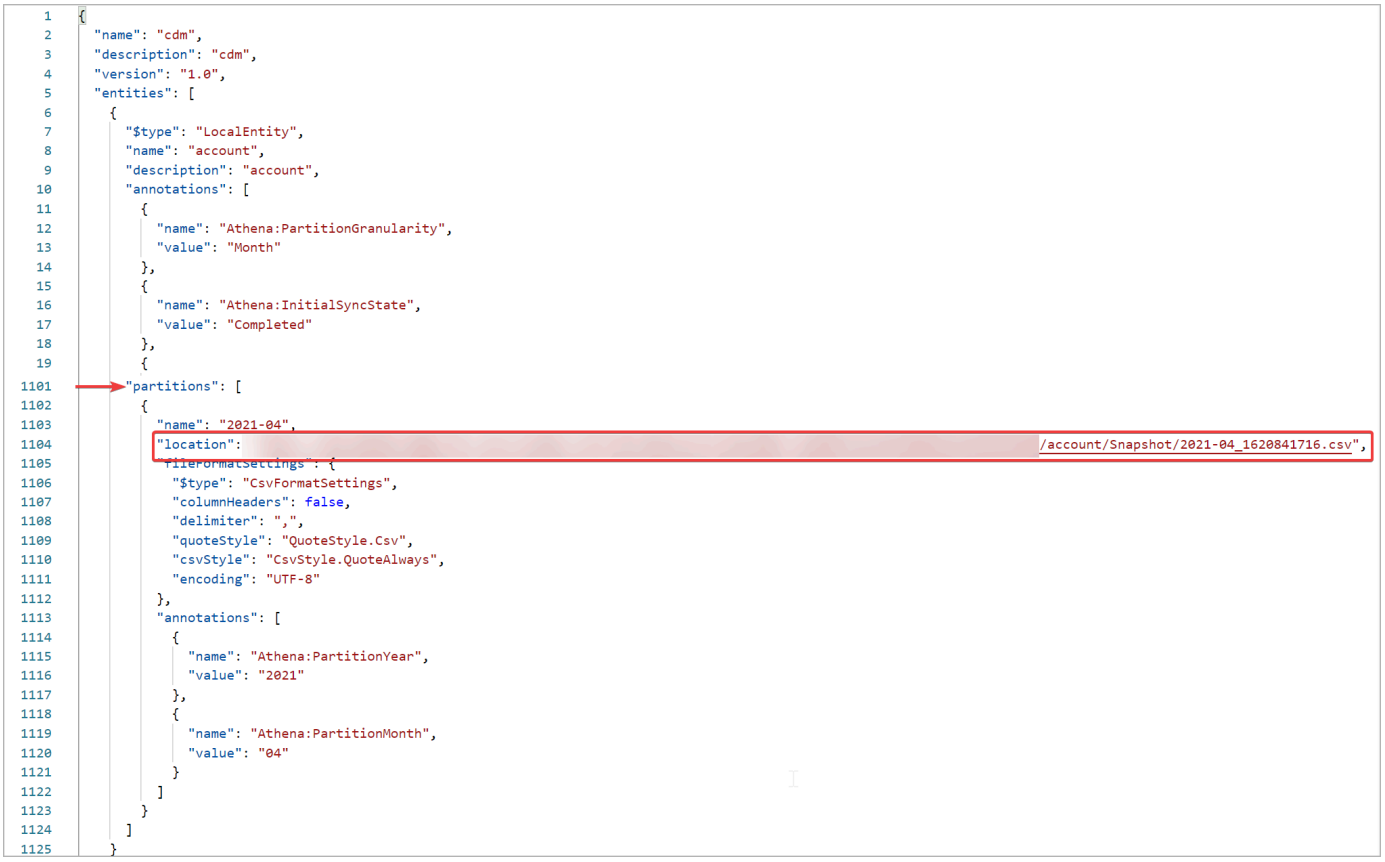


Changes in Dataverse are continuously pushed to the corresponding CSV files by using the trickle feed engine. This is the T2 interval, where another snapshot is taken. *table*-T2.csv—for example, Accounts-T2.csv or Contacts-T2.csv (assuming there are changes for the table) —and model.json are updated to the new snapshot files. Any new person who views snapshot data from T2 onward is directed to the newer snapshot files. This way, the original snapshot viewer can continue to

work on the older snapshot T1 files while newer viewers can read the latest updates. This is useful in scenarios that have longer-running downstream processes.

> **Note**
>
> - A new snapshot file is created only when there's a data update.
> - Only the latest five snapshot files are retained. Stagnant data is automatically removed from your Azure Data Lake Storage Gen 2 account.

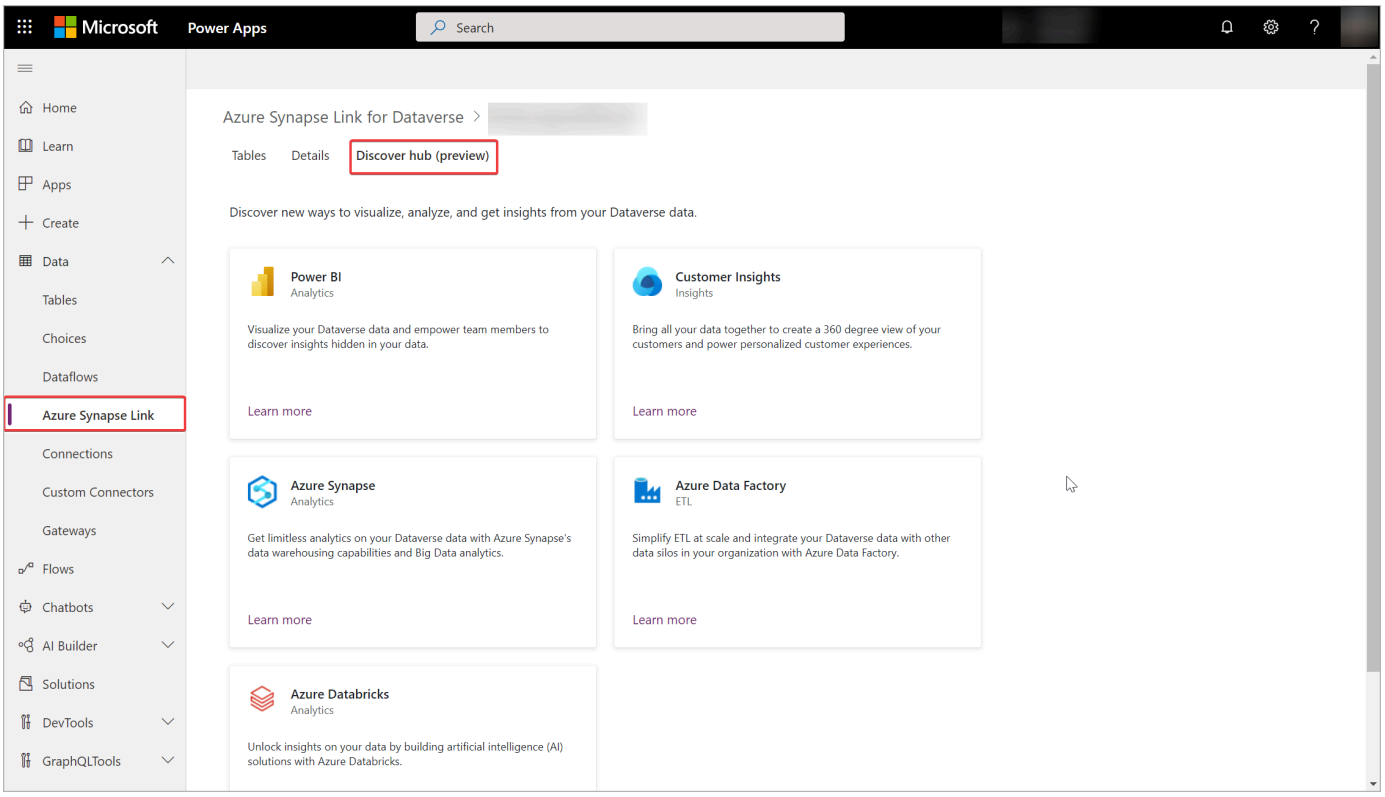
Here's an example of the model.json file, which always points to the latest time-stamped account snapshot file.



What's next?

After successfully using the Azure Synapse Link for Dataverse service, discover how you can analyze and consume your data with **Discover Hub**. To access **Discover Hub**, go to **Power Apps** > **Azure Synapse Link**. Select your linked service and then select the **Discover Hub** tab. Here you can find recommended tools and curated documentation to help you get the most value out of

your data.



See also

Analyze Dataverse data in the data lake with Power BI

Ingest Dataverse data in the data lake with Azure Data Factory

Azure Synapse Link for Dataverse Advanced Configuration

Azure Synapse Link FAQ

Last updated on 04/09/2026

Advanced configuration options in Azure Synapse Link

Azure Synapse Link offers multiple ways to write and read your data to fit various analytical scenarios. Depending on your analytical scenario, you can choose a specific configuration from the options below.

Expand table

Scenario	Applies to	Available configuration options
Operational reporting	Microsoft Dataverse tables, finance and operations tables, and entities.	Synapse Analytics with Delta lake option provides better query response times especially applicable for querying large amounts of data. More information: Synapse Link with Delta lake option
Operational reporting	Dataverse tables only.	Synapse Link with "In place update" configuration option provides CSV files in your data lake that are updated near-real time. This is a legacy option available for Dataverse tables. This option isn't supported for tables from finance and operations apps.
Data integration	Dataverse tables and finance and operations tables and entities.	"**Append only** option provides CSV files that contain incremental data. You can build pipelines that consume incremental data and populate downstream systems **User-specified data partition** feature enables choosing a custom data partitioning strategy specifically for Dataverse tables. Finance and operations table data are partitioned by the system based on appropriate partition strategy. This option isn't available for finance and operations apps.

> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as export to data lake. Microsoft renamed the service in May 2021. The service continues to export data to Azure Data Lake Storage as well as Azure Synapse Analytics. Starting Sept-2023, Azure Synapse Link also enables you to choose data from Dynamics 365 finance and operations applications. Not all integration patterns are supported with finance and operations apps. For guidance on

transitioning from export to data lake feature in finance and operations apps to Synapse Link, go to the Transition guide.

This article covers advanced configuration settings available for Dataverse tables. These options aren't available for finance and operations apps.

1. In-place updates vs. append-only writes.
2. User-specified data partitioning.

In-place updates vs. append-only writes

While writing Dataverse table data to the Azure data lake, based on the `createdOn` value, which is the date and time when the record was created, there are two different settings to choose from. They are, **In place update** and **Append only**.

The default setting (for tables where `createdOn` is available) is to do an in-place update or upsert (update or insert) of the incremental data in the destination. If the change is new and a corresponding row doesn't exist in the lake, in the case of a create, the destination files are scanned, and the changes are inserted into the corresponding file partition in the lake. If the change is an update and a row exists in the lake, the corresponding file in the lake is updated, rather than inserted, with the incremental data. In other words, the default setting for all CUD (create, update, delete) changes in Dataverse tables, where `createdOn` is available, is to do an in place update in the destination, in Azure data lake.

You can switch the default behavior of an in place update by using an optional setting called **Append only**. Rather than an **In place update**, in **Append only** mode, incremental data from Dataverse tables are appended to the corresponding file partition in the lake. This is a per table setting and available as a checkbox under **Advanced** > **Show advanced configuration settings**. For Dataverse tables with **Append only** turned on, all the CUD changes are incrementally appended to the corresponding destination files in the lake. When you choose this option, the partition strategy defaults to **Year** and when data is written to the data lake, it's partitioned by yearly basis. **Append only** is also the default setting for Dataverse tables that don't have `createdOn` value.

This table describes how rows are handled in the lake against CUD events for each of the data write options.

Expand table

Event	In place update	Append only
Create	The row is inserted in the partition file and is based on the `createdOn` value on the row.	The row is added to the end of the partition file and is based on the `createdOn` value of the record.
Update	If the row exists in the partition file, then it's replaced or updated with updated data. If it doesn't exist, it's inserted in the file.	The row, along with the updated version, is added to the end of the partition file.
Delete	If the row exists in the partition file, it's removed from the file.	The row is added to the end of the partition file with `IsDelete column = True`.

> ⓘ **Note**
>
> For Dataverse tables where **Append only** is enabled, deleting a row in the source doesn't delete or remove the row in the lake. Instead, the deleted row is appended as a new row in the lake and the `isDeleted` column is set to **True**.
>
> Dirty read (**ALLOW_INCONSISTENT_READS**) for serverless is enabled for append only mode. **ALLOW_INCONSISTENT_READS** means that user is able to read the files that can be constantly modified while the `SELECT` query is running. Results are consistent and equivalent to reading a snapshot of the file. (It isn't equivalent to database snapshot isolation because of the different snapshot generation time.)
>
> Not all CUD changes are captured in **append only**: The Synapse Link processes changes in data in groups or "batches" before publishing them to the data lake. As a result, if the user makes changes within a short time interval, not all CUD changes are captured in the data lake.

Here are some more details on when to use either of the options.

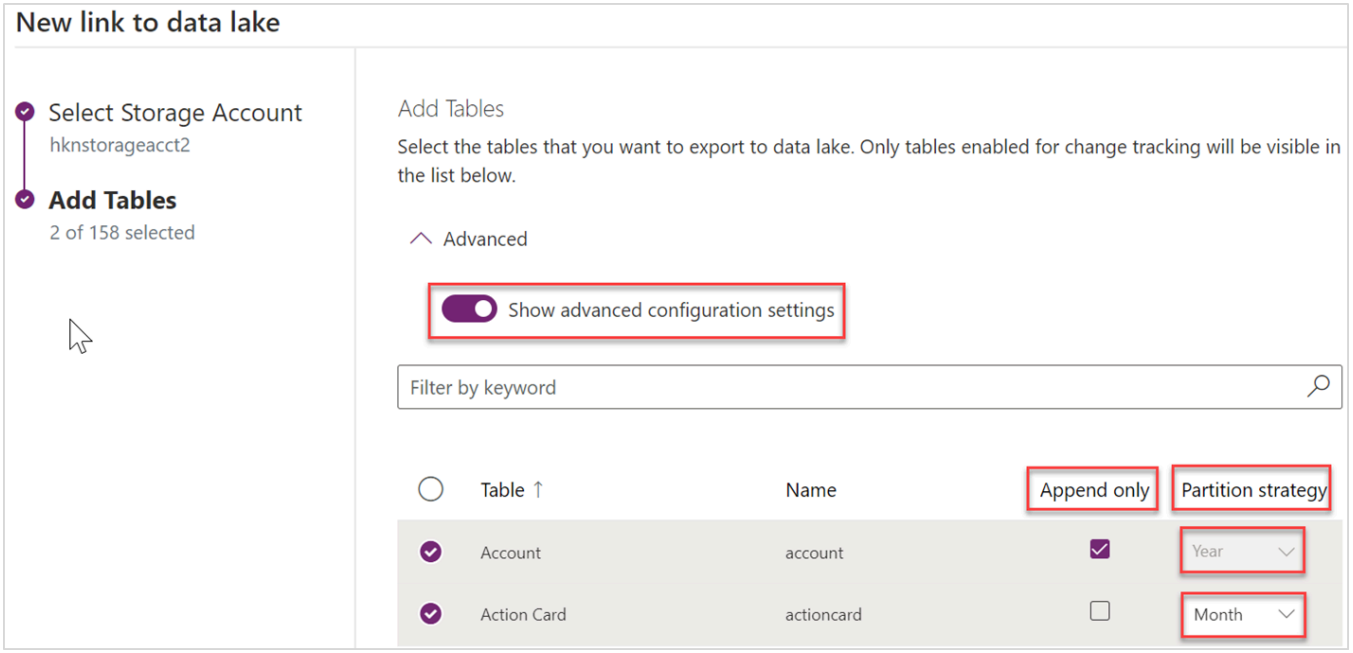
- **In place update** (legacy): This option is the default setting and recommended only if you want to connect directly to the data in lake and need the current state (not history or incremental changes). The file contains the full data set and can be utilized via Power BI or by copying the entire dataset for ETL (Extract, Transfer, Load) pipelines.

  > ⓘ **Important**
  >
  > **In place update** is a legacy mode and doesn't scale well with high data volumes or when changes are frequent. If your table has a large amount of data or experiences a

high rate of creates, updates, or deletes, use **Append only** instead to ensure reliable and performant data export.

- **Append only**: Select this option if you aren't directly connecting to data in the lake and want to incrementally copy data to another target using ETL pipelines. This option provides a history of changes to enable AI and ML scenarios. This is the recommended option for tables with large data volumes or frequent data changes.

You can toggle the **Show advanced configuration settings** under **Advanced** in Azure Synapse Link for Dataverse to customize your data partition strategy and select options to write to the Azure data lake.

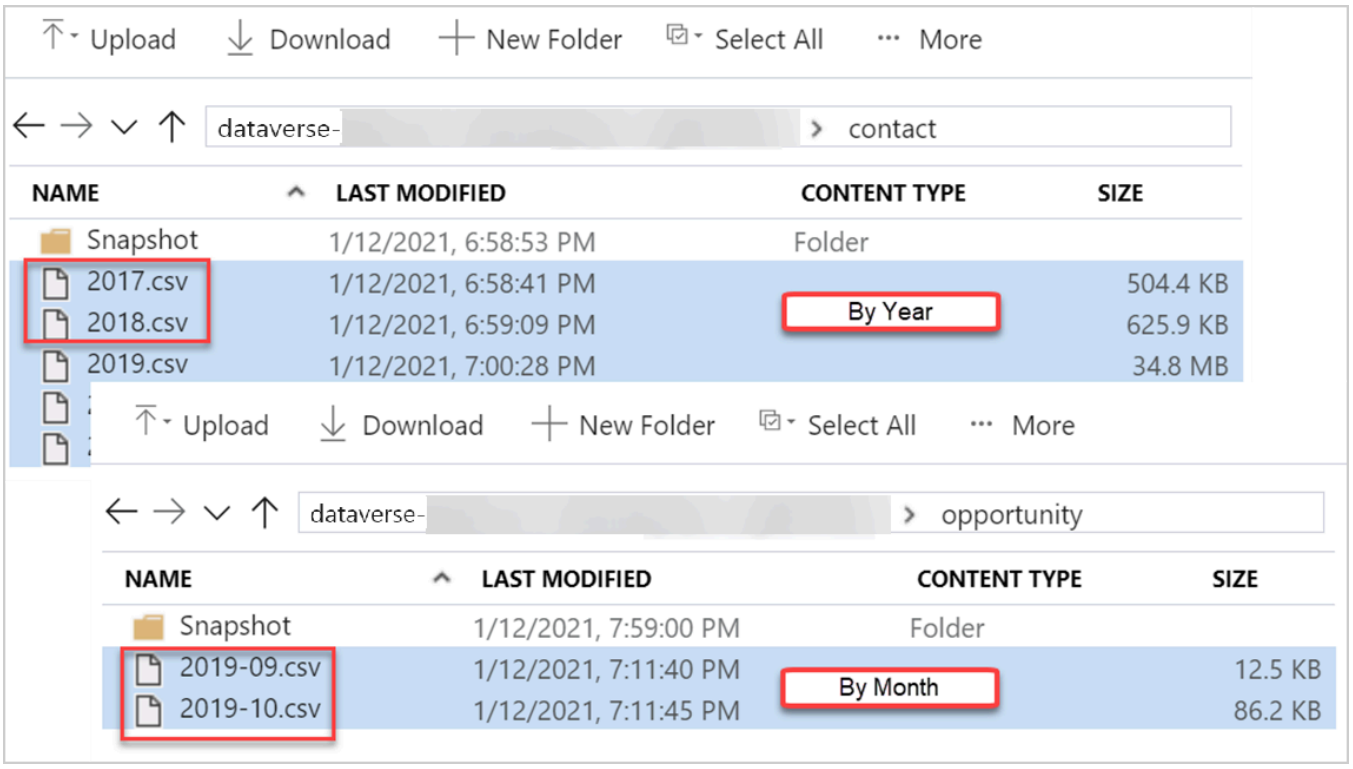


Data partitioning

When you write Dataverse table data to Azure Data Lake Storage by using Azure Synapse Link, the tables are partitioned in the lake based on the `createdOn` value on each row in the source. The default partition strategy is by month, so data is partitioned in Azure Data Lake on a monthly basis.

Based on the Dataverse table volume and data distribution, you can choose to partition your data by year. With this option, when Dataverse table data is written to the Azure data lake, it's partitioned on a yearly basis based on the `createdOn` value on each row in the source. For tables without the `createdOn` column, the rows of data are partitioned into a new file every 5,000,000 records. This is a per table setting and is available as a checkbox under **Advanced** > **Show advanced configuration settings**.

Here are examples of how data can be handled in the lake with a yearly or monthly partition strategy.



See also

Azure Synapse Link for Dataverse

Last updated on 04/22/2026

Use managed identities for Azure with your Azure data lake storage

Summarize this article for me

Azure Data Lake Storage provides a layered security model. This model enables you to secure and control the level of access to your storage accounts that your applications and enterprise environments demand, based on the type and subset of networks or resources used. When network rules are configured, only applications requesting data over the specified set of networks or through the specified set of Azure resources can access a storage account. You can limit access to your storage account to requests originating from specified IP addresses, IP ranges, subnets in an Azure Virtual Network (VNet), or resource instances of some Azure services.

Managed identities for Azure, formerly known as Managed Service Identity (MSI), help with the management of secrets. Microsoft Dataverse customers using Azure capabilities create a managed identity (part of enterprise policy creation) that can be used for one or more Dataverse environments. This managed identity that will be provisioned in your tenant is then used by Dataverse to access your Azure data lake.

With managed identities, access to your storage account is restricted to requests originating from the Dataverse environment associated with your tenant. When Dataverse connects to storage on behalf of you, it includes additional context information to prove that the request originates from a secure, trusted environment. This allows storage to grant Dataverse access to your storage account. Managed identities are used to sign the context information in order to establish trust. This adds application-level security in addition to the network and infrastructure security provided by Azure for connections between Azure services.

Before you start

- Azure CLI is required on your local machine. Download and install
- You need the following PowerShell modules. If you don't have them, open PowerShell and run these commands:
  - Azure Az PowerShell module: `Install-Module -Name Az`
  - Azure Az.Resources PowerShell module: `Install-Module -Name Az.Resources`
  - Power Platform admin PowerShell module: `Install-Module -Name Microsoft.PowerApps.Administration.PowerShell`
- Clone the PowerApps-Samples repository on GitHub to a location where you can run PowerShell commands: `git clone https://github.com/microsoft/PowerApps-Samples.git`.

Scripts are organized in subfolders under `powershell/managed-identities/Source`. Run each script from its specific subfolder—for example, `Source\Identity`.

- We recommend that you create a new storage container under the same Azure resource group to onboard this feature.

> **Important**
>
> Don't move scripts out of their folders. Scripts rely on relative paths and shared files in the repository structure.

Enable enterprise policy for the selected Azure subscription

> **Important**
>
> You must have **Azure subscription Owner** role access to complete this task. Obtain your Azure **Subscription ID** from the overview page for the Azure resource group.

1. Open Azure CLI with run as administrator and sign into your Azure subscription using the command: `az login` More information: Sign in with Azure CLI
2. (Optional) if you have multiple Azure subscriptions, make sure to run `Update-AzConfig -DefaultSubscriptionForLogin { Azure subscription id }` to update your default subscription.
3. In PowerShell, change to the `Source` folder within the repository you cloned as part of Before you start.
4. To enable the enterprise policy for the selected Azure subscription, run the PowerShell script **./SetupSubscriptionForPowerPlatform.ps1**.

    - Provide the Azure subscription ID.

Create enterprise policy

> **Important**
>
> You must have **Azure resource group Owner** role access to complete this task. Obtain your Azure **Subscription ID**, **Location**, and **Resource group** name, from the overview page for the Azure resource group.

1. In PowerShell, change to the `Source\Identity` subfolder and run the script to create the enterprise policy:

   PowerShell

   ```
   cd <path-to-repo>\powershell\managed-identities\Source\Identity
   .\CreateIdentityEnterprisePolicy.ps1
   ```

   - Provide the Azure subscription ID.
   - Provide the Azure resource group name.
   - Provide preferred enterprise policy name.
   - Provide the Azure resource group location.

2. Save the copy of the **ResourceId** after policy creation.

> **Note**
>
> The following are the valid **location** inputs supported for policy creation. Select the location that's most appropriate for you.

Locations available for enterprise policy

United States EUAP

United States

South Africa

UK

Australia

Korea

Japan

India

France

Europe

Asia

Norway

Germany

Switzerland

Canada

Brazil

UAE

Singapore

Grant reader access to the enterprise policy via Azure

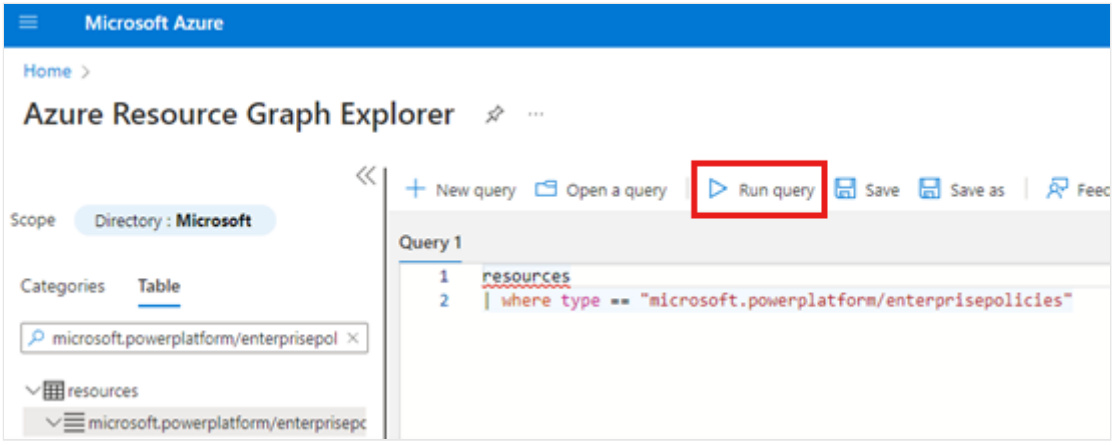
Dynamics 365 admins, and Power Platform admins can access the Power Platform admin center to assign environments to the enterprise policy. To access the enterprise policies, Azure Key vault admin membership is required to grant the **Reader role** to the Dynamics 365 or Power Platform admin. Once the reader role is granted, the Dynamics 365 or Power Platform admins will see the enterprise policies on the Power Platform admin center.

Only the Dynamics 365 and Power Platform admins who were granted the reader role to the enterprise policy can 'add environment' to the policy. Other Dynamics 365 and PowerPlatform admins might be able to view the enterprise policy, but they'll get an error when they try to add environment.

> **ⓘ Important**
>
> You must have - `Microsoft.Authorization/roleAssignments/write` permissions, such as User Access Administrator or Owner to complete this task.

1. Sign into the Azure portal.
2. Obtain the Dynamics 365 Power Platform admin user's **ObjectID**.
   a. Go to the **Users** area.
   b. Open the Dynamics 365 or Power Platform admin user.
   c. Under the overview page for the user, copy the **ObjectID**.
3. Obtain the enterprise policies ID:
   a. Go to the Azure **Resource Graph Explorer**.

b. Run this query: `resources | where type == 'microsoft.powerplatform/enterprisepolicies'`



c. Scroll to the right of the results page and select the **See details** link.

d. On the **Details** page, copy the ID.

4. Open Azure CLI and run the following command, replacing the `<objId>` with the user's **ObjectID** and the `<EP Resource Id>` with the enterprise policy ID.

   - `New-AzRoleAssignment -ObjectId <objId> -RoleDefinitionName Reader -Scope <EP Resource Id>`

Connect enterprise policy to Dataverse environment

> **Important**
>
> You must have the **Power Platform administrator** or **Dynamics 365 administrator** role to complete this task. You must have the **Reader** role for the enterprise policy to complete this task.

1. Obtain the Dataverse environment ID.
   a. Sign into the Power Platform admin center.
   b. Select **Manage** > **Environments**, and then open your environment.
   c. In the **Details** section, copy the **Environment ID**.
2. From the `Source\Identity` subfolder, run this PowerShell script: `./NewIdentity.ps1`

   - Provide the Dataverse environment ID.
   - Provide the **ResourceId**.

     StatusCode = **202** indicates the link was successfully created.

3. Sign into the Power Platform admin center.
4. Select **Manage** > **Environments**, and then open the environment you specified earlier.

5. In the **Recent operations** area, select **Full history** to validate the connection of the new identity.

Configure network access to the Azure Data Lake Storage Gen2

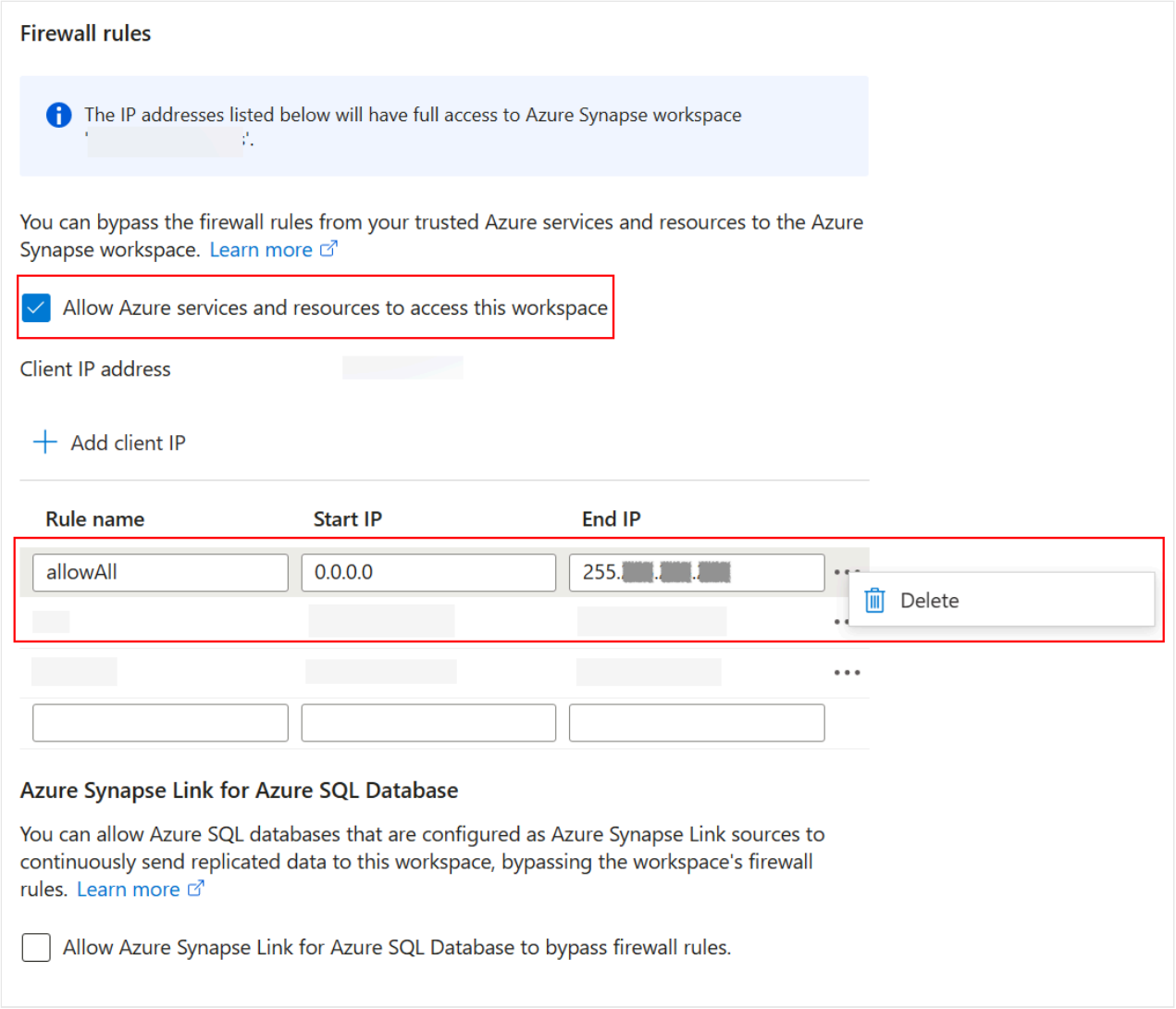
> **Important**
>
> You must have an Azure Data Lake Storage Gen2 **Owner** role to complete this task.

1. Go to the Azure portal.
2. Open the storage account connected to your Azure Synapse Link for Dataverse profile.
3. On the left navigation pane, select **Networking**. Then, on the **Firewalls and virtual networks** tab select the following settings:
   a. **Enabled from selected virtual networks and IP addresses**.
   b. Under **Resource instances**, select **Allow Azure services on the trusted services list to access this storage account**
4. Select **Save**.

Configure network access to the Azure Synapse Workspace

> **Important**
>
> You must have an Azure **Synapse administrator** role to complete this task.

1. Go to the Azure portal.
2. Open the Azure Synapse workspace connected to your Azure Synapse Link for Dataverse profile.
3. On the left navigation pane, select **Networking**.
4. Select **Allow Azure services and resources to access this workspace**.
5. If there are **IP firewall rules** created for all IP range, delete them to restrict public network access.



6. Add a new **IP firewall rule** based on the client IP address.
7. Select **Save** when done. More information: Azure Synapse Analytics IP firewall rules

Create a new Azure Synapse Link for Dataverse with managed identity

> **Important**
>
> Dataverse: You must have the Dataverse **system administrator** security role. Additionally, tables you want to export via Azure Synapse Link must have the **Track changes** property enabled. More information: Advanced options
>
> Azure Data Lake Storage Gen2: You must have an Azure Data Lake Storage Gen2 account and **Owner** and **Storage Blob Data Contributor** role access. Your storage account must enable **Hierarchical namespace** for both initial setup and delta sync. **Allow storage account key access** is required only for the initial setup.
>
> Synapse workspace: You must have a Synapse workspace and the **Synapse Administrator** role access within the Synapse Studio. The Synapse workspace must be in the same region

as your Azure Data Lake Storage Gen2 account. The storage account must be added as a linked service within the Synapse Studio. To create a Synapse workspace, go to Creating a Synapse workspace.

When you create the link, Azure Synapse Link for Dataverse gets details about the currently linked enterprise policy under the Dataverse environment then caches the identity client secret URL to connect to Azure.

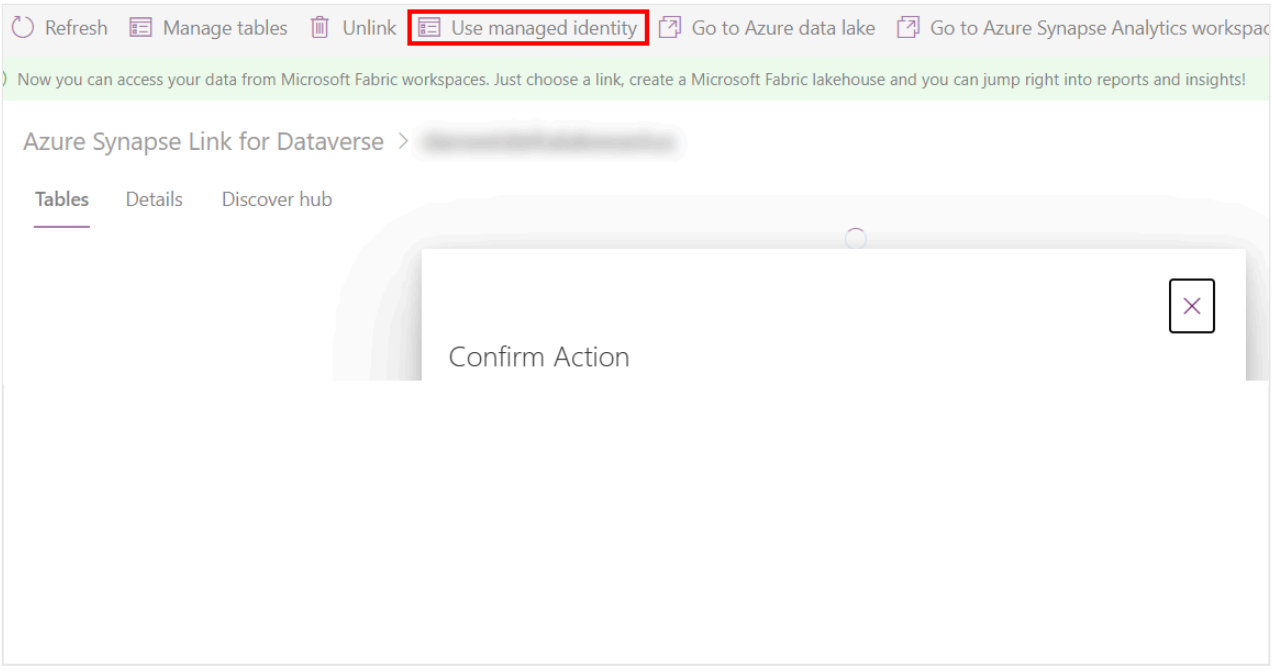
1. Sign into Power Apps and select your environment.
2. On the left navigation pane, select **Azure Synapse Link**, and then select **+ New link**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Fill in the appropriate fields, according to the intended setup. Select the **Subscription**, **Resource group**, and **Storage account**. To connect Dataverse to Synapse workspace, select the **Connect to your Azure Synapse workspace** option. For Delta Lake data conversion select a Spark pool.
4. Select **Select Enterprise Policy with Managed Service Identity**, and then select **Next**.
5. Add the tables you want to export, and then select **Save**.

Enable managed identity for an existing Azure Synapse Link profile

> **Note**
>
> To make the **Use managed identity** command available in Power Apps, you need to finish the above setup to connect the enterprise policy to your Dataverse environment. More information: Connect enterprise policy to Dataverse environment

1. Go to an existing Synapse Link profile from Power Apps (make.powerapps.com).

2. Select **Use managed identity**, and then confirm.



Troubleshooting

If you receive 403 errors during the link creation:

- Managed identities take extra time to grant transient permission during initial sync. Give it some time and try the operation again later.
- Make sure the linked storage doesn't have the existing Dataverse container(**dataverse-environmentName-organizationUniqueName**) from the same environment.
- You can identify the linked enterprise policy and `policyArmId` by running the PowerShell script `./GetIdentityEnterprisePolicyforEnvironment.ps1` from the `Source\Identity` subfolder, with the Azure **Subscription ID** and **Resource group** name.
- You can unlink the enterprise policy by running the PowerShell script `./RevertIdentity.ps1` from the `Source\Identity` subfolder, with the Dataverse environment ID and `policyArmId`.
- You can remove the enterprise policy by running the PowerShell script **.\RemoveIdentityEnterprisePolicy.ps1** from the `Source\Identity` subfolder, with `policyArmId`.

Known limitation

Only one enterprise policy can connect to the Dataverse environment simultaneously. If you need to create multiple Azure Synapse Link links with managed identity enabled, make sure all linked Azure resources are under the same resource group.

See also

What is Azure Synapse Link for Dataverse?

Last updated on 02/25/2026

Choose finance and operations data in Azure Synapse Link for Dataverse

Microsoft Azure Synapse Link for Dataverse lets you choose data from Dynamics 365 finance and operations apps. Use Azure Synapse Link to continuously export data from finance and operations apps into Azure Synapse Analytics and Azure Data Lake Storage Gen2.

Azure Synapse Link for Dataverse is a service that's designed for enterprise big data analytics. It provides scalable high availability together with disaster recovery capabilities. Data is stored in the Common Data Model format, which provides semantic consistency across apps and deployments.

Azure Synapse Link for Dataverse offers the following features that you can use with finance and operations data:

- You can choose both standard and custom finance and operations entities and tables.
- Continuous replication of entity and table data is supported. Create, update, and delete (CUD) transactions are also supported.
- You can link or unlink the environment to Azure Synapse Analytics and/or Data Lake Storage Gen2 in your Azure subscription. You don't have to go to the Azure portal or Microsoft Dynamics Lifecycle Services for system configuration.
- You can choose data and explore by using Azure Synapse. You don't have to run external tools to configure Synapse Analytics workspaces.
- All features of Azure Synapse Link for Dataverse are supported. These features include availability in all regions, saving as Parquet Delta files, and restricted storage accounts.
- The table limits in the Export to Data Lake service aren't applicable in Azure Synapse Link for Dataverse.
- By default, saving in Parquet Delta Lake format is enabled for finance and operations data, so that query response times are faster.

> **Note**
>
> This feature is generally available with finance and operations application versions shown in the following list. If you haven't yet applied these application versions, install the latest cumulative update to use this feature.
>
> - 10.0.38 (PU62) cumulative update 7.0.7120.59 or later
> - 10.0.37 (PU61) cumulative update 7.0.7068.109 or later.
> - 10.0.36 (PU60) cumulative update 7.0.7036.133 or later.

You might need to apply additional updates for recent fixes. More information: [Known limitations with finance and operations tables]

If you're planning to adopt the export to data lake feature in finance and operations apps, consider adopting Azure Synapse Link with finance and operations data support instead. Go to the software lifecycle announcements related to **export to data lake feature** for more details. For guidance and tools to upgrade from export to data lake to Azure Synapse Link go to **transition from legacy data generation services** as well as **TechTalk Series: Synapse Link for Dataverse: Transitioning from Export to Azure Data Lake to Synapse Link**

Prerequisites

- You must have a finance and operations sandbox (Tier-2) or higher environment. You can also use an environment provisioned with an ERP-based template.

  > **Note**
  >
  > With the availability of **Power Platform environment provisioned with ERP-based templates**, also known as *unified environments*, Microsoft offers limited support for cloud hosted environments (CHE) as of June 1, 2024. If you're using cloud hosted environments, consider moving to **Power Platform environment provisioned with ERP based templates**.

- You can't add finance and operations data to an existing storage account that's configured with Azure Synapse Link. You must have access to an Azure subscription so that you can create a new Synapse Link profile.

- Depending on how you plan to consume finance and operations data, there are additional prerequisites as shown here.

Expand table

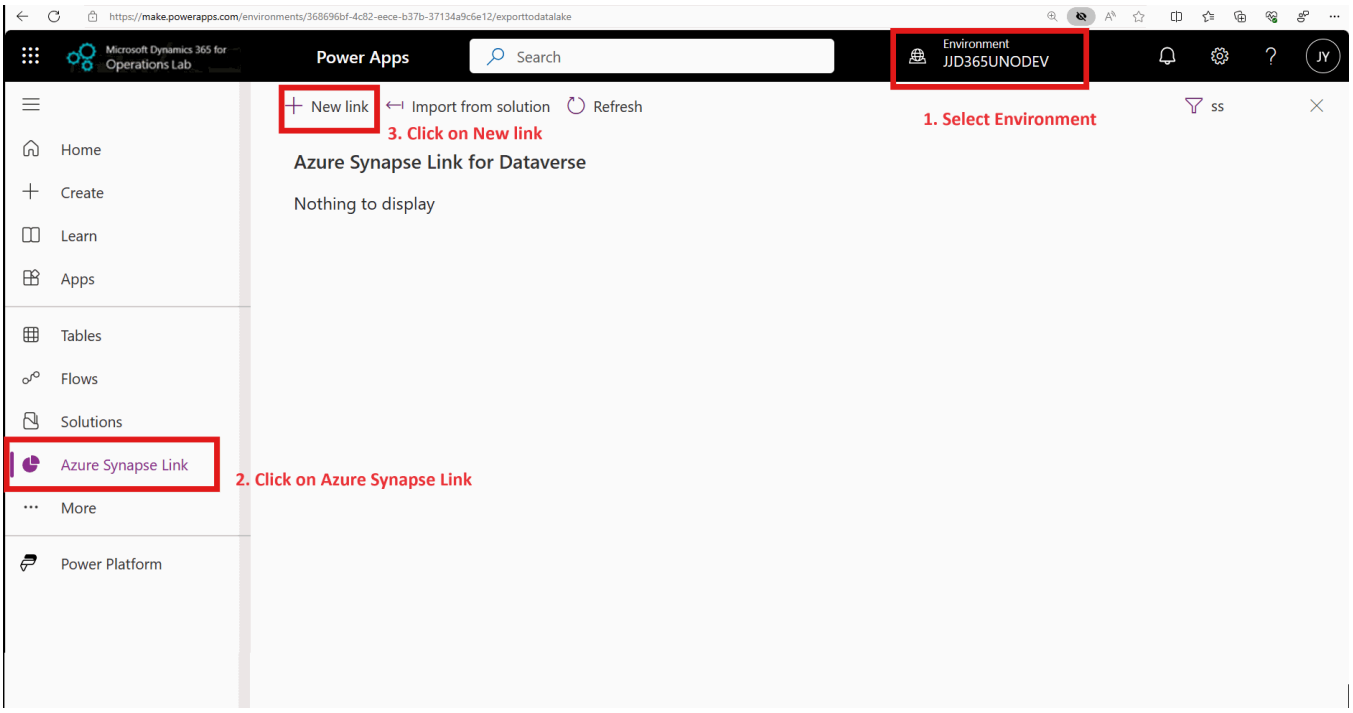
How you plan to consume data	Azure Synapse Link feature you use	Prerequisites and Azure resources needed
Access finance and operations tables via Microsoft Fabric No need to bring your own storage, Synapse	Link to Fabric	Microsoft Fabric workspace

How you plan to consume data	Azure Synapse Link feature you use	Prerequisites and Azure resources needed
workspace, or Spark pool because the system uses Dataverse storage and compute resources		
Access finance and operations tables via Synapse query Finance and operations tables are saved in delta parquet format enabling better read performance. You can't choose finance and operations tables to be saved in CSV format.	Go to Add finance and operations tables in Azure Synapse Link	Azure Data lake Azure Synapse workspace Azure Synapse Spark pool
Load incremental data changes into your own downstream data warehouse The system saves incremental changes into files in CSV format. No need to bring Synapse workspace or Spark pool because your data is saved in CSV format.	Go to Access incremental data changes from finance and operations Also go to Azure Synapse Link - incremental update)	Azure data lake

Add finance and operations tables in Azure Synapse Link

You can enable both finance and operations tables and finance and operations entities in Azure Synapse Link for Dataverse. This section is focused on finance and operations tables.

1. Sign in to Power Apps and select the environment you want.
2. On the left navigation pane, select **Azure Synapse Link**.
3. On the command bar of the **Synapse Link** page, select **+ New link to data lake**.
4. Select **Connect to your Azure Synapse Analytics workspace**, and then select the **Subscription**, **Resource group**, and **Workspace name**.
5. Select **Use Spark pool for processing**, and then select the precreated Spark pool and storage account.
6. Select **Next**.
7. Add the tables you want to export. You can choose finance and operations tables provided the prerequisites are met.
8. Select **Advanced**, select **Show advanced configuration settings** and enter the time interval, in minutes, for how often the incremental updates should be captured. The options **Append only** and **Partition** available at the table level are ignored for finance and operations tables because the system determines the partitioning strategy.
9. Select **Save**. Tables selected are initialized and ready for reporting.



Note

- Finance and operations apps tables are allowed only in Azure Synapse Link. Makers can't see them in the **Tables** area in Power Apps (make.powerapps.com).
- You don't have to define finance and operations apps tables as virtual tables, and you don't have to enable change tracking for each table.
- You can't add finance and operations tables into an existing Synapse Link profile that contained Dataverse tables. You need to create a new profile. Once you create a new profile, you can add tables from both finance and operations as well as Dataverse.
- To include finance and operations tables in Synapse Link, you must enable the Delta lake feature in your Synapse Link profile. Finance and operations table selection isn't visible if your Synapse Link profile isn't configured for Delta lake.
- Delta lake conversion time interval determines how often table data is updated in delta format. For near real time updates, enter 5, 15 or 60 minutes as the desired updated time interval. Choose 1440 for daily time interval if near real-time updates aren't required. Delta conversion consumes compute resources from the Spark pool you have provided in the configuration of the Synapse Link profile. The lower the time interval, the more compute resources are consumed and you can incur more cost. You can monitor the Spark pool cost in Azure portal to see the compute cost.

- The time interval chosen for Spark conversion might not reflect the actual data freshness you observe in the lake. By entering 15 minutes as the time interval, you request Delta conversion jobs to be triggered every 15 minutes in case there are data changes. Depending on the size of the Spark compute pool and the amount of incremental data available for processing, the actual data refresh observed in the lake could be more than 15 minutes. You can increase the size of the Spark pool to achieve better data refresh times. More information: Recommended spark pool configuration

- In the event that the system ran into an error during initial sync or updates, you receive an error and a link to trouble-shooting documents that can be used to diagnose and resolve the error.

Known limitations and changes to behavior

If you are transitioning from export to data lake feature in finance and operations, you might find a few changes to the behavior as described here. To learn more about the upcoming roadmap and stay in touch with the product team, join the preview Viva Engage group.

- **Don't see all tables?** Microsoft continues to enable all actively used finance and operations apps tables in Azure Synapse Link with application updates. If you have a previous version of finance and operations apps, not all required tables are enabled by default. You can enable more tables yourself by extending table properties and enabling the change tracking feature. For more information about how to enable change tracking, go to Enable row version change tracking for tables.
- **Don't see your custom tables?** You must enable change tracking for them. More information: Enable row version change tracking for tables. If you're using a cloud hosted environment (CHE), you must perform a database sync operation to reflect the changes.
- **Special fields** such as `TimeZoneID` (TZID), binary fields in finance and operations tables aren't enabled in Azure SynapseL Link.
- **Synapse Link retains deleted rows** from finance and operations tables. You can identify and filter out deleted rows using the `isDelete` field. Go to Working with data and metadata for more information.
- **Staging tables, temporary tables, and deprecated tables**, where names begin with `del_` in finance and operations apps, aren't allowed in Azure Synapse Link.
- The following tables, known as *kernel* tables in finance and operations apps, are supported by Fabric and Synapse Link. These tables are special, and you don't need to enable change tracking. Also, they're updated every 24 hours and not updated near-real time as the data doesn't change frequently: `DATAAREA`, `USERINFO`, `SECURITYROLE`,

`SECURITYUSERROLE`, `SQLDICTIONARY`, `PARTITIONS`, `SECURITYPRIVILEGE`, `TIMEZONESLIST`, `SECURITYDUTY`, `SECURITYSUBROLE`, `SECURITYUSERROLECONDITION`, `DATABASELOG`, `SECURITYROLERUNTIME`, `SECURITYROLEPRIVILEGEEXPLODEDGRAPH`, `SECURITYROLEDUTYEXPLODEDGRAPH`, `TIMEZONESRULESDATA`, `SECURITYROLEEXPLODEDGRAPH`, `USERDATAAREAFILTER`, `SYSINHERITANCERELATIONS`.

- Master company data sharing is a preview feature in finance and operations apps. Tables that participate in the master company data sharing feature are supported with Synapse Link or Fabric link features. Data exported from these tables are keyed by all company records, not only the master company. This is done to enable simpler reporting so you don't need to use master company data sharing logic to expand data. However, if you're transitioning from previous data export solutions, you might need to filter out data from non-master companies.
- Table inheritance and derived tables are concepts in finance and operations apps. When choosing a derived table from finance and operations apps, fields from the corresponding base table currently aren't included. For example, if you choose the `DirPerson` table, a table derived from `DirPartyTable` also known as the base table, exported data contains fields from the base table `DirPartyTable` You need to select the base table in addition to the derived table if you need access to these fields. You can use this FastTrack solution provided on GitHub. This solution creates views, which include columns from base tables.
- **Memo fields and long descriptions of type `nVarchar(Max)`** are included in Synapse Link. However, the field size is truncated to 2,000 characters.
- **ID fields from finance and operations tables are renamed to FnO_Id** to avoid field name conflicts with Dataverse tables.
- **Fields with SQL reserved words** are renamed by attaching a trailing character. Ex. `Level` becomes `Level_` and `Resource` becomes `Resource_`.

Recent fixes applicable to finance and operations tables

If you are on a previous version of finance and operations, you need to update to the latest version to apply fixes to several issues. These issues and fixes are mentioned here for reference purposes only. When you update to the latest version, all fixes are applied.

- **AOS Authorized fields**: If the table selected contains data columns that are secured via **AOS Authorization**, those columns are ignored and the exported data doesn't contain the column. For example in a custom table named *CustTable*, the column *TaxLicenseNum* has the metadata property **AOS Authorization** set to **Yes**. This column is ignored when *CustTable* data is exported with Azure Synapse Link.

  > ⓘ Note

Update your finance and operations environment to these versions or later to enable AOS authorized fields:

- PU 63:7.0.7198.105
- PU 62:7.0.7120.159

With this update, AOS authorization fields are added to tables:

- Incremental updates include this column.
- Modified records show these columns and value.
- Full refresh includes these fields and all values.

- **Extensible Data security**: When a finance and operations table added to Azure Synapse Link is secured via extensible data security policies, the system might not export data. This issue is fixed in the latest application update.

  > ⓘ Note
  >
  > Available updates to finance and operations tables with Azure Synapse Link for Dataverse:
  >
  > - Version 10.0.39 (PU63) cumulative update 10.0.1860.50
  > - Version 10.0.38 (PU62) cumulative update 10.0.1777.135
  > - Version 10.0.37 (PU61) cumulative update 10.0.1725.175
  >
  > You'll need to apply a quality build where the system applies a bypass for extensible data security policies for the Azure Synapse Link service.

- **Tables with valid time stamp behavior**: If there are finance and operations app tables that exhibit valid time stamp behavior, only the data rows that are currently valid are exported with Azure Synapse Link. For example, the **ExchangeRate** table contains both current and previous exchange rates. Only currently valid exchange rates are exported in Azure Synapse Link. This issue is fixed in the latest application update shown here.

  > ⓘ Note
  >
  > Available updates to finance and operations tables with Azure Synapse Link for Dataverse:
  >
  > - Version 10.0.40 (PU64) platform update 7.0.7120.179
  > - Version 10.0.39 (PU63) platform update 7.0.7198.143
  > - Version 10.0.38 (PU62) platform update 7.0.7279.58

With this update, expired data rows are added to tables. You need to perform a full refresh to include previous rows.

- **Fields of Array type**: If the table selected contains data columns that are of **Array** type, those columns are ignored and the exported data doesn't contain the column. For example, in a custom table named *WHSInventTable*, columns **FilterCode** and **FilterGroup** are of type array. These columns aren't exported with Azure Synapse Link. This issue is fixed in the latest application update shown here.

  > ⓘ Note
  >
  > Available updates to finance and operations tables with Azure Synapse Link for Dataverse:
  > - Version 10.0.41 (PU65) platform update 7.0.7367.153 or later
  > - Version 10.0.42 (PU66) platform update 7.0.7452.84 or later
  > - Version 10.0.43 (PU67) platform update 7.0.7521.153 or later
  >
  > With this update, Array type fields are added to tables. You need to perform a full refresh to include previous rows. Above updates also contain the fix for Array fields of Enumerated data types.

- Extended invoice length is a feature enabled with finance and operations version 10.0.40 and later.

  > ⓘ Note
  >
  > Extended vendor invoice number feature is enabled with updates to finance and operations tables with Azure Synapse Link for Dataverse:
  > - Version 10.0.41 (PU65) platform update 7.0.7367.134 or later
  > - Version 10.0.42 (PU66) platform update 7.0.7452.72 or later
  > - Version 10.0.43 (PU67) platform update 7.0.7521.0 or later

- **Deleted rows missing from derived tables** issue is addressed in updates mentioned below. When choosing a derived table from finance and operations apps. For example, if you choose the `DirPerson` table, a table derived from `DirPartyTable` also known as the base table, deleted markers from derived and base tables are missing in exported data.

  > ⓘ Note

Deleted rows missing from derived tables is enabled with updates to finance and operations tables with Azure Synapse Link for Dataverse:

- Version 10.0.40 (PU64) platform update 7.0.7279.199 or later
- Version 10.0.41 (PU65) platform update 7.0.7367.136 or later
- Version 10.0.42 (PU66) platform update 7.0.7452.75 or later

- **NULL values in finance and operations fields** aren't reflected as empty strings in exported data and might cause export failures. While NULL values are not expected to be present in finance and operations data, there might be NULL values inserted via data integration.

  > ⓘ **Note**
  >
  > Null value handling support for exported data for finance and operations tables is enabled with:
  >
  > - Version 10.0.41 (PU65) platform update 7.0.7367.149 or later
  > - Version 10.0.42 (PU66) platform update 7.0.7452.88 or later
  > - Version 10.0.43 (PU67) platform update 7.0.7497.0 or later

- Finance and operations apps tables added to an Azure Synapse Link profile might be removed when a back-up is restored in Dataverse. You can copy and paste a comma separated list of tables into the search box within the manage tables option to select a list of tables at once.

- When a finance and operations apps database is restored, tables added to an Azure Synapse Link profile need to be reinitialized. Before reinitializing finance and operations tables, you must also restore the Dataverse database. After restoring the database, you must add finance and operations tables into the profile. You can copy and paste a comma separated list of tables into the search box within the manage tables option to select a list of tables at once.

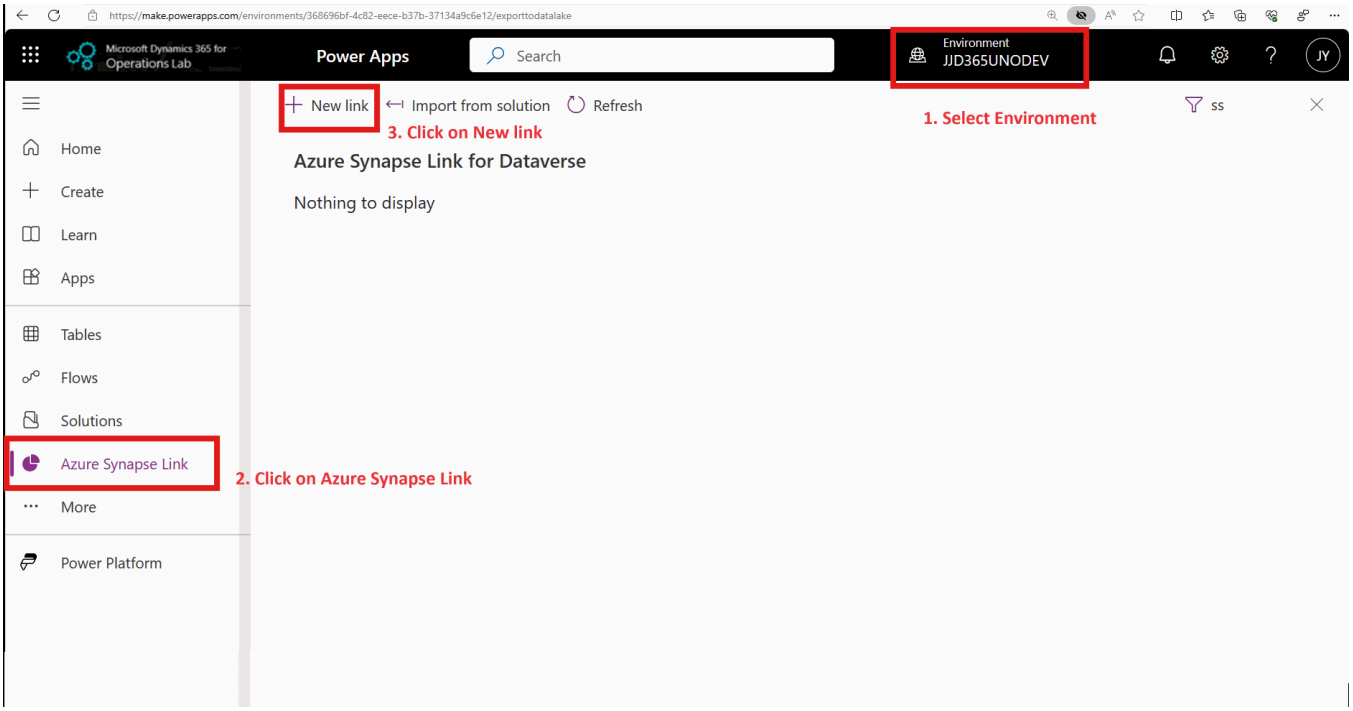
- Finance and operations apps tables included in an Azure Synapse Link profile can't be migrated to a different environment using the import and export profile feature in Azure Synapse Link.

Access incremental data changes from finance and operations

To load incremental data changes from finance and operations into your own downstream data warehouse, create an Azure Synapse Link profile that provides only incremental data. Azure Synapse Link provides an initial export of all data rows and then provides you with access to data that changed periodically. The data is provided in CSV files stored in time stamped folders and you can easily consume the data using Azure Data factory or other data tools. More information: Azure Synapse Link - incremental update

To create an Azure Synapse Link profile with incremental data:

1. Sign in to Power Apps and select the environment you want.
2. On the left navigation pane, select **Azure Synapse Link**.
3. On the **Azure Synapse Link for Dataverse** page, select **+ New link** on the command bar.
4. Select **Subscription**, **Resource group** and a **Storage account**. You don't need to provide a Synapse workspace or a Spark pool.
5. Select **Next**. The option to choose tables appears.
6. Select **Advanced**, select **Show advanced configuration settings**, and then enable the option **Enable incremental update folder structure**
7. In the **Time interval** field, choose the desired frequency for reading incremental data. Using this frequency, the system partitions data into time stamped folders such that you can read the data without being impacted by ongoing write operations.
8. Select the Dataverse tables you want. You can also select finance and operations tables. The options **Append only** and **Partition** available at a table level are ignored. Data files are always appended and data is partitioned yearly.
9. Select **Save**. Tables selected are initialized and you see incremental data in the storage account.



> **Note**
>
> If you're upgrading from the export to data lake feature, enabling the incremental data changes option provides similar change data as the Change feeds feature
>
> We recommend that you create separate Azure Synapse Link profiles for incremental data and tables for ease of management.
>
> When you choose tables and enable incremental data changes, the row count shown in the Azure Synapse Link details page for each table reflects the total number of changes, not the number of records in the table.
>
> Data rows that contain deleted records from Finance and Operations tables contain the `uniqueidentifier` (ID) field. They don't contain the body of the record. Your downstream data pipeline might need to look up the corresponding fields using the ID field.
>
> The finance and operations table limitations are also applicable to incremental data from tables. More information:Known limitations with finance and operations entities

Working with data and metadata

Enumerated fields are coded data fields in finance and operations apps. For example, the **AssetTrans** table contains a field called **TransType**, which is an **Enumerated** field. Table fields contain numeric codes like 110, 120, or 131, which represent detailed descriptions like "Depreciation," "Lease" or "Major repairs." You can access these detailed descriptions by using the **GlobalOptionsMetadata** table that's automatically exported when you choose a table that contains enumerated fields. Enumerated fields are also called choice labels or, formerly, option sets. More information: Choice labels

If there are metadata changes to finance and operations tables, for example, a new field is added to a table, and the data exported in Azure Synapse Link reflects the latest metadata inclusive of the change. More information: Azure Synapse Link FAQ. If you're using Azure Synapse Link to query the data, you see the updated metadata reflected in Azure Synapse Link. If you consume incremental data changes, you can locate updated metadata within the incremental data folder with the latest date stamp. More information: Incremental folder structure

You'll also notice additional metadata fields appended by the system for each data row. Explanation of these fields is shown here.

Expand table

Metadata field	What this field contains	Equivalent field or feature in export to data lake
`createdon`	Indicates the date and time the record was created	Derived from `Created Date` time field in each finance and operations records.
`Createdtransactionid`	Not used	Not used
`Id`	Record ID is used by the system to uniquely identify records	Contains the `RecID` field of the record from finance and operations. This value is converted from an integer to a GUID.
`IsDelete`	If True this record is deleted from Dataverse or finance and operations	In case of a delete at source, export to data lake deletes the row from the destination data lake immediately. Azure Synapse Link performs a "soft delete" for table data in Delta format such that you can identify deleted rows without consuming change feeds. If you want to read active records, you can filter records where `isDelete` is True. In case of table data in Delta files, soft deleted rows are purged after 28 days. `IsDelete` field is also available in incremental updates and can be used to identify rows that are deleted. The body of the records with `isDelete` true are empty. Use system fields such as **Id** field to determine values of other fields. For example, you can look up the latest row for incremental update where `isDelete` is False, with the latest version number and sync modified on date for a given ID.
`modifiedon`	Indicates the date and time the record was last modified	This field is populated from modified date time field in finance and operations tables.
`modifiedtransactionid`	Used internally	Because `modifiedtransactionid` is used internally, don't use this field.
`SinkCreatedOn`	Indicates the date the record was written to the data lake. If there's a CSV change data (incremental update), data, and time data was written to lake is shown. For table data in Delta format, indicates the date and time of Delta Lake conversion.	You can use this date similar to the data lake modified date time field in the export to data lake feature.

Metadata field	What this field contains	Equivalent field or feature in export to data lake
`SinkModifiedOn`	Indicates the date the record was modified. In case of tables in Delta format files as well as incremental CSV files, contains the same date time as `SinkCreatedOn`.	You can use this date similar to the data lake modified date time field in export to data lake feature.
`sysdatastatecode`	If 1, this record is archived using the long term data retention feature. If 0 this is a live record.	You can use this field to identify finance and operations records that have been archived (and deleted from live data). The same field is available for for CSV change data (incremental update).
`sysrowversion`	Version number maintained in each finance and operations apps record that determines whether changes have been made to data. This field is used by the system to determine incremental or delta changes to process.	`sysrowversion` is used internally. You can use this field as a *watermark* to determine the last version of record that was updated. This field might be empty for Dynamics 365 customer engagement apps tables.
`tableid`	Contains a unique ID of each table	Contains the table ID from finance and operations.
`versionnumber`	Used internally - contains the last version of the row that has been synced to the data lake.	Similar to the `sysrowversion` this field contains the last processed version for Dynamics customer engagement apps tables. In case of Dynamics finance and operations apps tables, this field contains the same value as `sysrowversion`.

Reacting to system events and generating audit reports

Azure Synapse Link generates events when data transfer activities are completed or an issue is encountered. These events are recorded in Dataverse tables. You can create automated actions and audit reports using these Dataverse tables. For example, if you have an automated data pipeline that continuously copies changed data rows from your data lake to another location, special actions might be required when a table is inactivated or reactivated, or when table metadata is updated. Creating events enables you to automate these tasks.

If you used business events generated by the export to data lake feature in finance and operations to react to system events, you can continue to get alerted via Azure Synapse Link service. Synapse Link service contains additional configuration data that you can use to generate audit reports and Power Apps to monitor specific data of interest. For more information about receiving and consuming events from Synapse Link go to this article.

This table summarizes events form export to data lake and discusses how you can improve with Synapse Link configuration data.

Expand table

Export to data lake event	Scenario	How you can use configuration data in Synapse Link
Initialize complete	When you select a table, the system makes an initial copy of the data. This event indicates that initialization for a selected table has been completed. If multiple tables are enabled in the environment, you receive multiple events as initialization is completed for each table.	You can monitor Dataverse tables **Synapse Link Profile Entity State** and **Synapse Link external table state**. **InitialSyncProcessCompletedTime** and **InitialSyncState** return the initial sync completion status, which includes both metadata and data. The initial sync state is updated as **Completed** once the initial sync completes. You can also get additional information such as record counts for each table. Synapse Link doesn't delete previous change data in CSV files (incremental update) when you reinitialize a table.
Metadata changed	This event indicates that table metadata has changed. For example, a new column has been added to the table, or, in rarer cases, a column has been deleted. You receive multiple events for each table, one for each metadata change. If you have downstream data pipelines that can't react to metadata changes (a situation that is known as schema drift), you can detect changes by using these events and modify downstream pipelines.	The **Initial Sync Metadata Created Time** field in the **Synapse Link Profile Entity State** table is updated when metadata changes are detected. Use this field to force metadata changes to downstream pipelines using metadata stored in the lake.
Deactivated	This event indicates that a previously active table has been	There's no "deactivate" option in Synapse Link. You can however, delete a table from a Synapse Link

Export to data lake event	Scenario	How you can use configuration data in Synapse Link
	deactivated. When a table is deactivated, data that is changed in finance and operations apps isn't updated in the data lake	profile. You can find deleted tables in the Dataverse table **Synapse Link Profile Entity State**.
Activated	This event indicates that a previously inactivated table has been reactivated. When a table is reactivated, the system reinitializes data in the data lake. You might later receive an **Initialize complete** event.	There's no deactivate or activate option in Synapse Link. You can delete a table and readd the table into a Synapse Link profile. A new record is added in the **Synapse Link Profile Entity State** table when a table is added. You can monitor the status of the table during initialize stage using the Dataverse tables **Synapse Link Profile Entity State** and **Synapse Link external table state**.
Error	This event indicates that the data sync process has paused due to an error.	**Synapse Link Profile Entity State** status field indicates the status of tables.
Lake switch	This event indicates that the system administrator has modified the destination of the data lake.	This event isn't applicable. Changing a data lake associated with a Synapse Link profile as well as changing configuration options isn't supported. However, Synapse Link supports multiple profiles. This means, an administrator can create multiple Synapse Link profiles with tables (the same table can be added to multiple profiles). Read the Dataverse table **Synapse Link profile** to find configuration data on all profiles including deleted profiles.

Enable finance and operations data entities in Azure Synapse Link

You can enable both finance and operations entities and finance and operations apps tables in Azure Synapse Link for Dataverse. This section is focused on finance and operations data entities.

The process of enabling finance and operations entities has the following steps. Each step is explained in the following subsections.

1. Enable finance and operations virtual entities in the Power Apps maker portal. This step lets you use finance and operations entities in Power Apps (make.powerapps.com) to build apps. You can also use them with Azure Synapse Link.

2. Enable row version change tracking for Entities. You must complete this step to enable Azure Synapse Link to use finance and operations entities.

After you complete both steps, you can select finance and operations entities in Azure Synapse Link under **Dataverse tables**. To create Azure Synapse Link for Dataverse in Delta Lake format, follow the steps in Export Dataverse data in Delta Lake format.

> **Note**
>
> Finance and operations entities start with the prefix **mserp_**.

Enable finance and operations virtual entities in Power Apps

You must enable finance and operations entities as virtual tables in Dataverse. Makers can then use the chosen finance and operations entities to build apps, and the entities can also be used with Azure Synapse Link.

To enable finance and operations entities, follow the steps in Enable Microsoft Dataverse virtual entities.

> **Tip**
>
> To validate Azure Synapse Link features, use a few of the sample entities from the following list. They appear under the **Dataverse tables** section in Azure Synapse Link.
>
> - **MainAccountBiEntity** – This entity contains a list of ledger accounts.
> - **ExchangeRateBiEntity** – This entity contains exchange rates in the system.
> - **InventTableBiEntity** – This entity contains a list of inventory items.

Enable change tracking for finance and operations entities

When you enable change tracking for finance and operations entities, they appear under Dataverse tables in Azure Synapse Link. Finance and operations entities start with the prefix **mserp_**.

To enable change tracking, follow these steps.

1. In Power Apps, select **Tables** on the left navigation pane, and then select the table you want.
2. Select **Properties** > **Advanced options**.

3. Select the **Track changes** option, and then select **Save**. If the option is unavailable, go to Known limitations with finance and operations entities).

Known limitations with finance and operations entities

Currently, there are several limitations with finance and operations entities and Azure Synapse Link. To learn more about the upcoming roadmap and stay in touch with product team, join the preview Viva Engage group aka.ms/SynapseLinkforDynamics.

- Enabling change tracking might fail with the error message "chosen entity doesn't pass the validation rules..." or the **Track changes** checkbox might be disabled for some tables that are virtual tables. Currently, change tracking can't be enabled for all finance and operations entities. The **Track changes** checkbox is unavailable for entities created in finance and operations in the past for data migration.

  > **Note**
  >
  > For a list of finance and operations entities that pass validation rules, run the **Data entity row version change tracking validation report** available in finance and operations apps at path **System administration/Setup/Row version change tracking/Data entity row version change tracking validation report.** This report shows entities that pass and fail validation rules. You can also use this report to enable row version change tracking for entities that pass data validation rules. To enable change tracking for an entity, choose **Row version Change Tracking** from the menu, and then choose **Enable change tracking**.
  >
  > For more information about entity validation rules and how you can fix them, go to Enable row version change tracking for data entities. You might need developer assistance to complete the steps.
  >
  > If the chosen entity is unavailable because of the change tracking limitation, you might be able to choose the tables that comprise the data from that entity. You can use EntityUtil solution provided by the FastTrack team to create entity shapes using tables.

- In case of a database restore operation in Dataverse, finance and operations entities enabled in Azure Synapse Link are removed. To re-enable entities, you need to re-enable corresponding virtual tables for all selected entities, re-enable change tracking, and reselect the tables in Azure Synapse Link.

Last updated on 01/09/2026

Azure Synapse Link for Dataverse troubleshooting guide

Article • 10/15/2024

When you open an Azure Synapse Link profile, you might see an error with a message and link to this article. In addition, users might also see **Error** status next to tables that are already in an Azure Synapse Link profile. Use this troubleshooting guide to diagnose and fix error conditions with Azure Synapse Link for Dataverse.

Expand table

Error message	Cause	Resolution
MSI-801: Azure Synapse Link is unable to access storage account secured via MSI. Data updates are paused.	You have enabled managed services identify (MSI) to the storage account selected for this Azure Synapse Link profile. Azure Synapse Link service doesn't have access to the storage account due to MSI configuration issues and the service has paused. Revisit and verify MSI configuration	Verify that the MSI policy isn't deleted. Validate that MSI is granted with the required access privileges. More information: MSI configuration Once the required access is granted, the service resumes.
ADLS-802: Azure Synapse Link is unable to access storage account. Data updates are paused.	Azure Synapse Link service can't access storage account and the service has paused. Permissions associated with storage account assigned to Azure Synapse Link profile might have changed or the storage account doesn't exist.	Verify that the storage account exists and that the Azure Synapse Link service has owner access to storage account. More information: Prerequisites Once required access is granted, the service resumes.
CT-803: Row version change tracking configuration key disabled in linked environment. Data updates in some tables may be paused.	Azure Synapse Link service requires the configuration key **enable row version change tracking** enabled in the linked finance and operations apps environment. This configuration key is disabled in the linked environment and the service has paused finance and operations apps tables from the linked environment. It's possible that a database restore operation caused the configuration to be reverted. Enable the configuration key and remove and readd finance and operations apps tables to resume.	Your administrator must reenable the finance and operations apps configuration key. More information: Add configurations in a finance and operations apps environment The service resumes when the configuration key is reenabled.

Error message	Cause	Resolution
	Impacted tables are initialized to reflect new data.	
DV-804: This environment is in admin mode and service is paused.	When a Power Platform environment is in administration mode all services including Azure Synapse Link service are temporarily paused. When **Administration mode** is **Disabled**, the services are resumed.	Verify with your environment administrator and ensure that the environment is operational. Also verify that the **Background operations** environment setting is **Enabled** after **Administration mode** is **Disabled**. Azure Synapse Link service resumes when the environment is operational.
SYN-805: Synapse workspace isn't accessible.	Azure Synapse Link service can't access the Synapse workspace associated with this profile. Synapse tables might not be created immediately when you resolve this issue. The system creates tables in the Synapse workspace when data gets updated in Dataverse.	In the Synapse workspace networking configuration, enable the option **Allow Azure services and resources to access this workspace**.
SYN-806: Default storage container in Synapse workspace isn't accessible.	Synapse workspace associated with this Azure Synapse Link profile doesn't contain a primary Azure Data Lake Storage Gen2 file system. The default storage container might be deleted or the default storage container isn't accessible. A default storage container is used by Synapse workspace as temporary storage in creating the file system. A default storage container must be provided when creating a Synapse workspace. It's possible that this storage container was deleted or no longer accessible.	Ask your administrator to launch Synapse workspace and verify the name of the primary Azure Data Lake Storage Gen2 file system. Also verify that an Azure Data Lake Storage Gen2 storage container exists with that name within the same subscription. If the storage container is missing, create a storage container with the same name.
SYN-809: Synapse workspace {name} has data exfiltration policies enabled. You need to choose another workspace.	Synapse Link profiles with Delta conversion option requires Synapse workpaces without data exfiltration policies. You have likely chosen a Synapse workspace with data exfiltration policies, which inteferes with Synapse Link compute services.	Ask your administrator to provide a Synapse workspace where data exfilteration policies aren't enabled. You might also consider using Fabric link as an alternative to Synapse Link.
SYN-810: The default storage	Synapse Link profiles with Delta conversion option requires Synapse	Ask your administrator to remove network restrictions associated

Error message	Cause	Resolution
container associated with Synapse workspace is network restricted.	workpaces with a default storage container. If the default storage container associated with the select Synapse workspace has network restrictions applied, Synapse Link compute services are unable to perform the required updates.	with the default storage container associated with the Synapse workspace chosen or choose another Synapse workspace.
SPARK-807: Spark compute pool {name} in {Synapse workspace} is no longer available. Data updates are paused.	Azure Synapse Link service requires a Spark pool in order to process data into delta format. The Spark pool provided with this Azure Synapse Link profile is deleted or no longer available. It's also possible that access to the Synapse workspace have been restricted. Once you resolve the issue, Azure Synapse Link service resumes data updates when the data changes.	Verify that the Spark pool with the chosen name exists. If you or someone deleted this Spark pool, create the Spark pool with the same name.
SPARK-808: Spark compute pool {name} in {Synapse workspace} is insufficient. One or more tables could not be updated.	Azure Synapse Link service uses the Spark pool provided in the configuration to convert files to Delta parquet format as and when data is updated. When a large batch of data is available for update, the size of the Spark pool provided in the Azure Synapse Link profile might not provide sufficient compute power. This can happen especially when you add large tables as the initialization process creates a large batch of data. This might also result in case you have a "pay as you go" subscriptions with limited capacity. The Spark pools within these types of subscriptions are limited in capacity.	Increase the number of cores in the assigned Spark pool. The system consumes the increased capacity in case of larger workloads and scale down as needed. More information: Recommended Spark pool sizes for Delta conversion
CT-809: Row version change tracking property disabled in table. Data updates are paused.	Azure Synapse Link service requires enabling the **Row version change tracking** metadata property. This property is disabled for the table and the table is no longer updated. Updating this metadata property in finance and operations requires developer intervention. It's possible	Re-enable the row version change tracking metadata property to resume this table. You might need an administrator or developer to re-enable the property using developer tools. More information: Enable row

Error message	Cause	Resolution
	that the property was disabled after the Azure Synapse Link profile was created.	version change tracking for tables for details.
CT-810: The **Allow Row version change tracking** property of this table is set to No.	Azure Synapse Link service requires enabling the **Allow row version change tracking** metadata property to be set to **Yes**. Updating this metadata property in finance and operations requires developer intervention. It's possible that the property was disabled after the Synapse Link profile was created.	Re-enable the row version change tracking metadata property to resume this table. You might need an administrator or developer to re-enable the property using developer tools. More information: Enable row version change tracking for tables for details.
FnO-810: Linked finance and operations environment is unavailable or paused.	Finance and operations apps environment linked to this Azure Synapse Link profile is either unavailable or in maintenance mode. Azure Synapse Link service is unable to contact the environment and the service has paused tables sourced from finance and operations apps. In the case of cloud-hosted environments, it's possible that the environment is in a paused state.	Verify with your administrator that the linked environment isn't paused and isn't in maintenance mode. Azure Synapse Link service resumes when the environment is in operational mode.
FnO-811: Linked finance and operations environment needs to be updated.	Finance and operations apps environment linked to this Azure Synapse Link profile needs to be updated to the latest version. If you're currently on finance and operations versions PU60, PU61, or PU62, you must update to the latest cumulative updates 10.0.36 (PU 60) - 7.0.7036.133 or later 10.0.37 (PU 61) - 7.0.7068.109 or later 10.0.38 (PU 62) - 7.0.7120.59 or later The required updates are included in cumulative update 10.0.39 (PU63)	Contact your finance and operations apps administrator and apply the latest version updates to your environment.
FnO-812: Need to create additional indexes in your finance and operations environment.	When you select tables from a finance and operations apps environment, the system creates additional indexes in the database to enable performant data initialization and updates. This error indicates that indexes are missing for tables. This might be due to several reasons.	If this is a CHE environment, contact the owner of the environment. If this is a sandbox or production environment, no action is needed. The system enables indexes on

Error message	Cause	Resolution
	if you're receiving this error from a Cloud Hosted Environment (CHE), you need to enable specific flights and indexes as outlined in documentation. If you're receiving this error from a sandbox or a production environment, index creation might be delayed due to heavy database operations.	completion of peak workloads in finance and operations.
FnO-813: There is an unexpected NULL value in the field <...>. Verify data in the table and correct values as needed.	This error is the result of a data issue found by the service when transferring data from finance and operations apps. The finance and operations service doesn't allow NULL values in data fields. Therefore a NULL value is unexpected by the service. A space character represents the absence of a value in finance and operations apps. However, during data migration, especially using external tools, a NULL value might have been entered into a data field.	You can examine the table using the finance and operation app user interface or using a tool to resolve the issue. The system resumes once the data issue has been resolved. You don't need to reinitialize the table.
The following app(s) were not present in your tenant:...: Azure Synapse Link cannot get all required app permission to create profile.	Missing App ID surfaced in UI error message. Here's the complete list of required app permissions: • Export to data lake: `7f15f9d9-cad0-44f1-bbba-d36650e07765` • Common Data Service - Azure Data Lake Storage: `546068c3-99b1-4890-8e93-c8aeadcfe56a` • Insights Apps Platform Prod: `99ff962b-6252-4b98-8478-0c65a3ea1925`	Ask your administrator to add missing permissions in your Azure tenant by following these steps: • Open PowerShell and connect to Azure with an authenticated account using the command: `Connect-AzAccount` • Add the principal by running the following command for the app ID: `New-AzADServicePrincipal -ApplicationId '7f15f9d9-cad0-44f1-bbba-d36650e07765'`

Feedback

Was this page helpful? Yes No

Provide product feedback

Receive Azure Synapse Link for Dataverse notifications in Power Apps

Article • 03/22/2024

The state of your Azure Synapse Link for Dataverse is stored in a Microsoft Dataverse table with real-time updates. You can customize a notification using Power Automate flows or Power BI to stay on top of the state of your Azure Synapse Link for Dataverse activity instead of manually checking the Azure Synapse Link for Dataverse profile page in Power Apps (make.powerapps.com).

The Azure Synapse Link for Dataverse profile and its synchronization status are stored in six system-generated read-only tables within Dataverse. These tables are designed to refresh and update in real time, ensuring that the information contained within them remains accurate and up to date at all times.

How notification works with Azure Synapse Link for Dataverse

To customize a notification, monitor the data changes in a Dataverse table related to the synchronization status of Azure Synapse Link. By tracking these changes, you can export the updated data for visualization using Power BI or set up an automated cloud flow that activates whenever there's a change in the Azure Synapse Link for Dataverse state. This is achieved by reading the row change from any of the Azure Synapse Link state tables.

Connector to use for a Power Automate flow

We recommend users have the appropriate Power Platform licensing to use Power Automate to customize the trigger function. Use the Dataverse connector, which triggers a flow when a row is added, modified, or deleted in the selected Dataverse table. More information: Trigger flows when a row is added, modified, or deleted - Power Automate

Several options are available for using Power Automate to send notifications. Here are some examples.

Expand table

Action	Connector reference
Send a Teams instant message	Microsoft Teams
Send a text message to mobile device	Azure Communication Services SMS
Send an email to one or more recipients	Mail Office 365 Outlook
Send a notification to Power Apps or Power BI	Power Apps Notification Power BI

Dataverse tables and columns used to track state

There are tables that store all Azure Synapse Link profile information. The two main tables listed below provide all sync state information for your Azure Synapse Link profile.

Expand table

Table name	Description	Table reference
Azure Synapse Link external table state	This table has data only if you have Delta Lake profile. This table represents external nonpartitioned tables state in connected Synapse workspace. One record per synced entity including metadata table like option set.	synapselinkexternaltablestate
Azure Synapse Link profile table state	This table represents the sync state of the Azure Synapse Link entity in Azure Data Lake storage. One record per synced entity excluding metadata table such as option set.	synapselinkprofileentitystate

Here are some useful columns for monitoring the health of your Azure Synapse Link in each table:

- `EntityName` and `SynapseWorkspaceName` can be used as the primary identification of the selected table. Synapse workspace name is the same as Azure Synapse Link profile name shown in Power Apps.

- `LastSynchronizedOn` returns the date and time when the latest round of the Delta Lake conversion was successfully completed for each table.
- `RecordCount` returns the total number of records in the Delta Lake profile, minus soft-delete records for each table.
- `TableState` is marked as **created** if the link to the data lake and Delta Lake conversion is active and error-free.

A soft-delete in the Azure Synapse Link external table state table is performed: `LastSyncState` and `TableState` are marked as deleted for removed tables.

Azure Synapse Link profile entity state tables:

- `EntityName` and profile can be used as the primary identification of the selected table. The profile is the same as the Azure Synapse Link profile name shown in Power Apps.
- `InitialSyncProcessCompletedTime` and `InitialSyncState` return the initial sync completion status, which includes both metadata and raw data. The initial sync state marks as **Completed** once the initial sync completes.
- `LakeRecordCount` returns the total records exported to Data Lake in CSV format.
- `LastSyncedDataTime` returns the date and time when the latest round of the data lake file updated or created successfully for each table.
- `SyncState` shows as **InProgress** if the link to data lake is active and error-free.

> **Note**
>
> `LakeRecordCount` is different from Dataverse record count for the following reasons:
>
> - Sync latency.
> - Append-only mode captures transactions and appends one additional row for each CUD operation.

The remaining four tables provide additional details for Azure Synapse Link setup information:

Expand table

Table name	Description	Table reference
Azure Synapse database	This table captures linked Azure Data Lake storage and Synapse workspace and setup metadata information (one record per Azure Synapse Link profile).	synapsedatabase

Table name	Description	Table reference
Azure Synapse Link profile	This table captures Azure Synapse Link profile information (one record per Azure Synapse Link profile). A soft-delete in this table is performed: `ProfileState` is marked as **deleted** for deleted profile.	synapselinkprofile
Azure Synapse Link profile entity	This table captures entity metadata within the connected Azure Synapse Link profile (one record per synced table).	synapselinkprofileentity
Azure Synapse Link schedule	This table captures Azure Synapse Link profile information for incremental folder update or delta lake conversion time interval (one record per Azure Synapse Link profile).	RecurrenceInterval

See also

What is Azure Synapse Link for Dataverse?

Feedback

Was this page helpful? Yes No

Provide product feedback

Azure Synapse Link for Dataverse FAQ

This article provides information on frequently asked questions about exporting Microsoft Dataverse table data to Azure Synapse Analytics and Azure Data Lake.

Can I manually perform tasks such as creating, updating, deleting, or setting autodelete policies for data files in the connected Azure storage?

Data files shouldn't be modified by a customer and no customer files should be placed in the data folders.

> ⓘ **Note**
>
> To drop stale and stagnant data in the data lake without breaking the Azure Synapse Link, consider using the feature Query and analyze the incremental updates

How can I access my table relationships?

To access many-to-many relationships, the relationship is available as a table to select from the **Add tables** page for a new link and from the **Manage tables** for a preexisting link.

> ⓘ **Note**
>
> All relationships data is in Append-only mode by default when written in CSV format.

How can I get estimated costs before adding Azure Synapse Link?

Azure Synapse Link is a free feature with Dataverse. Utilizing Azure Synapse Link for Dataverse doesn't incur additional charges under Dataverse. However, consider potential costs for the Azure service:

- Data storage in Azure Data Lake Storage Gen2: Azure Storage Data Lake Gen2 Pricing | Microsoft Azure
- Data consumption cost (such as Synapse Workspace): Pricing - Azure Synapse Analytics | Microsoft Azure For comprehensive Microsoft Cost Management information, go to: Plan to manage Azure costs - Microsoft Cost Management | Microsoft Learn

What happens when I add a column?

When you add a new column to a table in the source, it's also added at the end of the file in the destination in the corresponding file partition. While the rows that existed before the addition of the column are't shown in the new column, new or updated rows show the newly added column.

What happens when I delete a column?

When you delete a column from a table in the source, the column isn't dropped from the destination. Instead, the rows are no longer updated and are marked as null while preserving the previous rows.

What happens if I change the data type of a column?

Changing the data type of a column is a breaking change and you need to unlink and relink.

What happens when I delete a row?

Deleting a row is handled differently based on which data write options you choose:

- In-place update with CSV format: This is the default mode. When you delete a table row in this mode, the row is also deleted from the corresponding data partition in the Azure Data Lake. In other words, data is hard deleted from the destination.
- Append-only with CSV format and incremental folder update: In this mode, when a Dataverse table row is deleted, it isn't hard deleted from the destination. Instead, a row is added and set as `isDeleted=True` to the file in the corresponding data partition in Azure Data Lake.
- Export to Delta lake format: Azure Synapse Link performs a soft delete on data during the next delta synchronization cycle, followed by a hard delete after 30 days.

Why am I not seeing a column header in the exported file?

Azure Synapse Link follows the Common Data Model to make it possible for data and its meaning to be shared across applications and business processes such as Microsoft Power Apps, Power BI, Dynamics 365, and Azure. In each CDM folder, metadata like a column header is stored in the model.json file. More information: Common Data Model and Azure Data Lake Storage Gen2 | Microsoft Learn

Why does the Model.json file increase or change in length for the data types and doesn't keep what is defined in Dataverse?

Model.json keeps the database length for the column's size. Dataverse has a concept of database length for each column. If you create a column with a size of 200 and later reduce it to 100, Dataverse still allows your existing data to be present in Dataverse. It does that by keeping `DBLength` to 200 and `MaxLength` to 100. What you see in Model.json is `DBLength` and if you use that for downstream processes you'll never provision lesser space for your Dataverse columns.

> **Note**
>
> Memo fields are defined as `varchar(max)` with a default maximum length of 9999.

What date and time formats can be expected in exported Dataverse tables?

There are three date and time formats that can be expected in the exported Dataverse tables.

Expand table

Column Name	Format	Data Type	Example
SinkCreatedOn and **SinkModifiedOn**	M/d/yyyy H:mm:ss tt	datetime	6/28/2021 4:34:35 PM

Column Name	Format	Data Type	Example
CreatedOn	yyyy-MM-dd'T'HH:mm:ss.sssssssXXX	datetimeOffset	2018-05-25T16:21:09.0000000+00:00
All Other Columns	yyyy-MM-dd'T'HH:mm:ss'Z'	datetime	2021-06-25T16:21:12Z

> **Note**
>
> **CreatedOn** data type changed from `datetime` to `datetimeOffset` on 07/29/2022. To edit the data type format for a table created before the change, drop and readd the table.
>
> You can choose different column behaviors for a Date and Time column in Dataverse, which updates the data type format. More information: Behavior and format of the Date and Time column

Why am I seeing 1.csv or 1_001.csv file names instead of regular date time partitioned file names for some Dataverse tables?

This behavior is expected when you choose append-only export mode and have tables without a valid **CreatedOn** column. Blobs are organized into files such as 1.csv, 2.csv (employing custom partitioning due to the absence of a valid creation date). When any partition approaches 95% of the MaxBlockPerBlobLimit, the system automatically generates a new file—illustrated here as 1_001.csv.

When should I use a yearly or monthly partition strategy?

For Dataverse tables where data volume is high within a year, we recommend you use monthly partitions. Doing so results in smaller files and better performance. Additionally, if the rows in Dataverse tables are updated frequently, splitting into multiple smaller files helps improve performance in the case of in-place update scenarios. Delta Lake is only available with yearly partition due to its superior performance compared to CSV format.

What is append only mode and what is the difference between append only and in-place update mode?

In append only mode, incremental data from Dataverse tables are appended to the corresponding file partition in the lake. For more information: Advanced Configuration Options in Azure Synapse Link

When do I use append only mode for a historical view of changes?

Append only mode is the recommended option for writing Dataverse table data to the lake, especially when the data volumes are high within a partition with frequently changing data. Again, this is a commonly used and highly recommended option for enterprise customers. Additionally, you can choose to use this mode for scenarios where the intent is to incrementally review changes from Dataverse and process the changes for ETL, AI, and ML scenarios. Append only mode provides a history of changes, instead of the latest change or in place update, and enables several time series from AI scenarios, such as prediction or forecasting analytics based on historical values.

How do I retrieve the most up-to-date row of each record and exclude deleted rows when I export data in append only mode?

In append only mode, you should identify the latest version of record with the same ID using `VersionNumber` and `SinkModifiedOn` then apply `isDeleted=0` on the latest version.

Why do I see duplicated version numbers when I export data using append only mode?

For append only mode, if Azure Synapse Link for Dataverse doesn't get an acknowledgment from the Azure data lake that the data has been committed due to any reason such as network delays, Azure Synapse Link retries in those scenarios and commit the data again. The downstream consumption should be made resilient to this scenario by filtering data using `SinkModifiedOn`.

Why am I seeing differences in `SinkModifiedOn` and `ModifiedOn` columns?

This is expected behavior. `ModifiedOn` is the date and time a record was last changed in Dataverse, while `SinkModifiedOn` is the date and time the record was written to the data lake. `SinkModifiedOn` is typically later than `ModifiedOn`, and the difference depends on factors such as batch size, system load, and the number of records being processed.

> **Important**
>
> In the case of long-running transactions in finance and operations apps, a record might be synchronized multiple times to ensure no data loss. In this scenario, `SinkModifiedOn` reflects the last synchronization time, not the first. For example, if a record is synced 10 times within an hour, `SinkModifiedOn` shows the time of the 10th sync even though the data was available in the lake from the first sync itself. Use caution when building latency reports based on the difference between `SinkModifiedOn` and `ModifiedOn`, as it can give a misleading impression of actual data availability.

Which Dataverse tables aren't supported for export?

Any table that doesn't have change tracking enabled isn't supported in addition to following system tables:

- Attachment
- Calendar
- Calendarrule

Certain Dataverse tables, such as postcomment, postregarding, postlike, post, and postrole, aren't available for customers to enable synchronization through Azure Synapse Link.

In some cases, these tables might appear when Dataverse long term retention is enabled. If that happens, a subset or all records from these tables might be exported. This is expected behavior for long term retention scenarios.

> **Important**

- These tables shouldn't be selected for synchronization.
- Their presence doesn't indicate full support for incremental sync.

> **Note**
>
> You can add the audit table for export using Azure Synapse Link for Dataverse. However, the export of the audit table is only supported with Delta Lake profiles.

I'm using export to delta lake feature, can I stop the Apache Spark job or change the execution time?

The Delta Lake conversion job is triggered when there was a data change in the configured time interval. There's no option to stop or pause the Apache Spark pool. However, you can modify the time interval after the link creation under Manage tables > Advanced Time interval.

Does Azure Synapse Link support lookup columns?

Lookup columns are made up of an ID and a value. Lookup values only change on the root table. To better reflect the value of a lookup column, we recommend joining with the original root table to get the latest value.

Does Azure Synapse Link support calculated columns?

In Dataverse, calculated columns keep only the formula information and the real value depends on the base table column. So calculated columns are only supported when all columns are located within the same exported table.

Which Dataverse tables use append only mode by default?

All tables that don't have a createdOn field is synced using append only mode by default. This includes relationship tables and the ActivityParty table.

Why do I see the error message - Content of directory on path can't be listed?

- Dataverse data is stored in the connected storage container. You need the "Storage Blob Data Contributor" role in the linked storage account to perform read and query operations through Synapse Workspace.
- If you choose to export data with Delta Lake format, your CSV file is cleaned up after the Delta Lake conversion. You need to query data with **non_partitioned** tables through Synapse Workspace.

Why do I see the error message - can't bulk load because the file is incomplete or couldn't be read (CSV file only)?

Dataverse data can continuously change through creating, updating, and deleting transactions. This error is caused by the underlying file being changed when you read data from it. So, for tables with continuous changes, change your consumption pipeline to use snapshot data (partitioned tables) to consume. More information: Troubleshoot serverless SQL pool

How can I use Azure Synapse Link to archive critical data?

Azure Synapse Link for Dataverse is designed for analytics purposes. We recommend customers use long-term-retention for archive purposes. More information: Dataverse long term data retention overview

Why don't I see any data changes in the data lake when records have been deleted in Dataverse?

For any direct SQL call to remove a record, the Azure Synapse Link for Dataverse service doesn't trigger because BPO.Delete isn't being called. For a sample feature, go to How to clean up inherited access.

See also

Azure Synapse Link for Dataverse

Transport the Azure Synapse Link for Dataverse configuration across environments

Article • 02/22/2023

In Power Apps, solutions are used to transport apps and components from one environment to another, or to apply a set of customizations to existing apps. Making the Azure Synapse Link for Dataverse configurations solution-aware enables basic application lifecycle management (ALM) abilities such as distribution, copy, and backup and restore of the Azure Synapse Link for Dataverse configuration.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

This section describes the prerequisites necessary to transport the Azure Synapse Link for Dataverse configuration across environments.

- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse.
- **Azure Synapse Link for Dataverse Advanced Configuration Options enabled.** This guide assumes that you have already toggled the Show advanced configuration settings under Advanced when creating Azure Synapse Link for Dataverse Azure Synapse Link Advanced Configuration Settings.

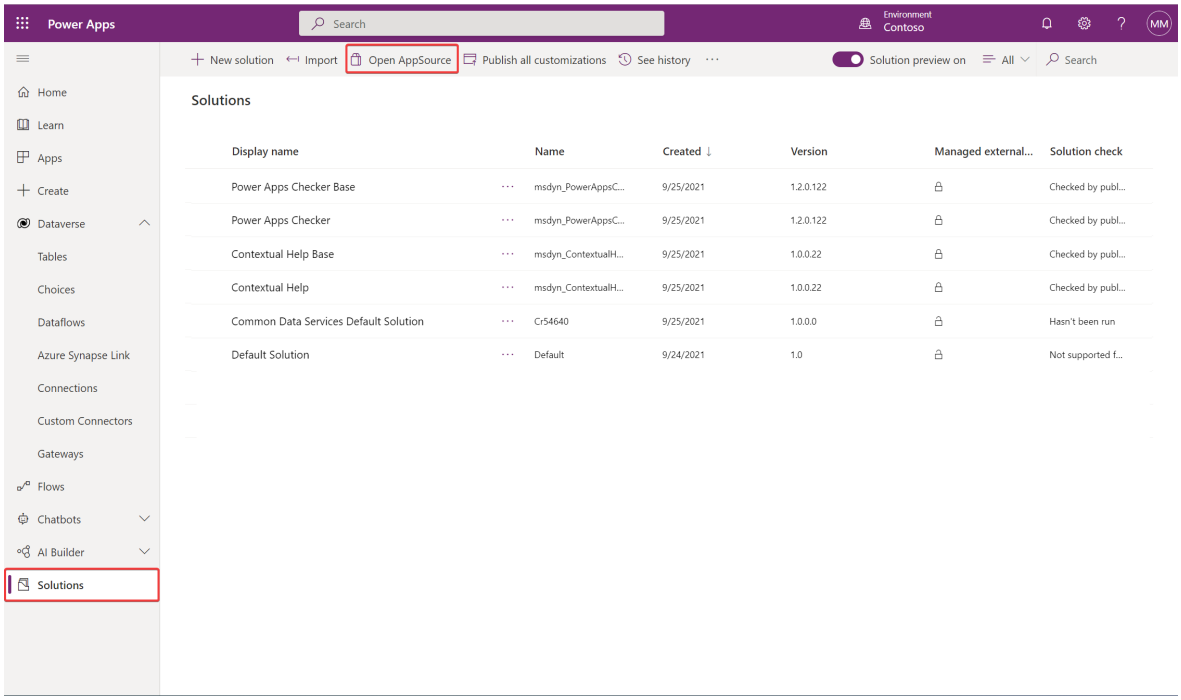
Install the Azure Synapse Link for Dataverse solution

> **Note**
>
> The Azure Synapse Link for Dataverse solution must be installed in both the source and the destination environments.

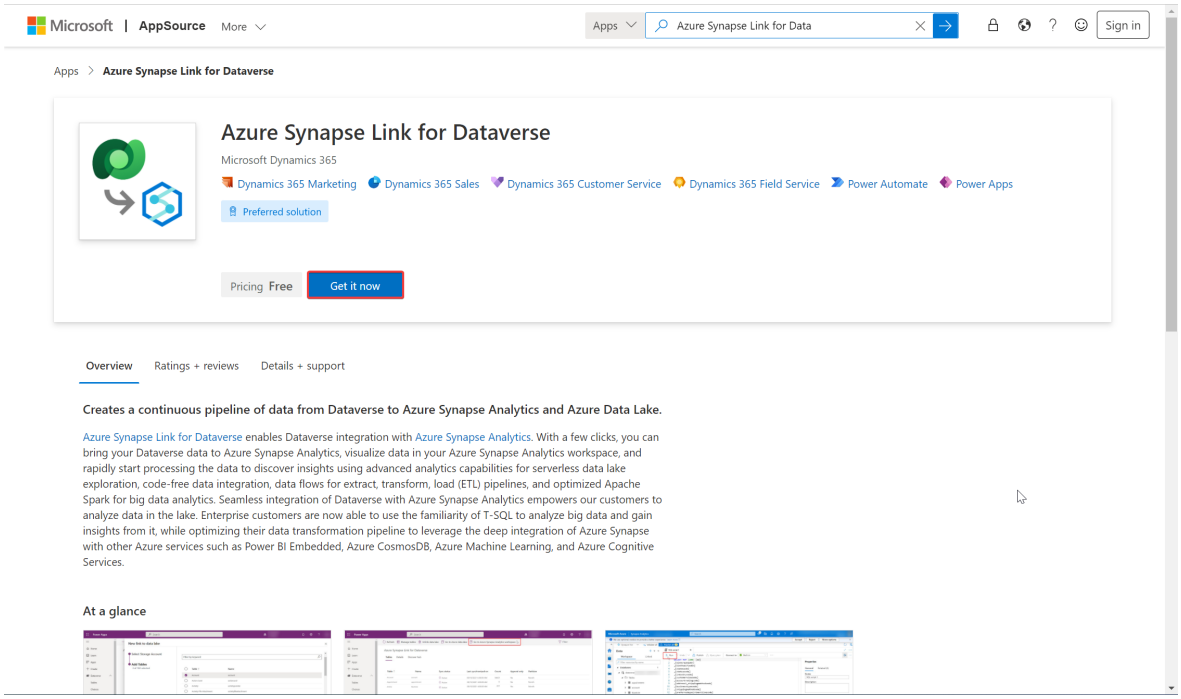
1. Sign into Power Apps, and then select the source environment for the Azure Synapse Link for Dataverse configuration.

2. On the left navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

3. Select **Open AppSource**.



4. Select for the solution named **Azure Synapse Link for Dataverse**, and then select **Get it now**.



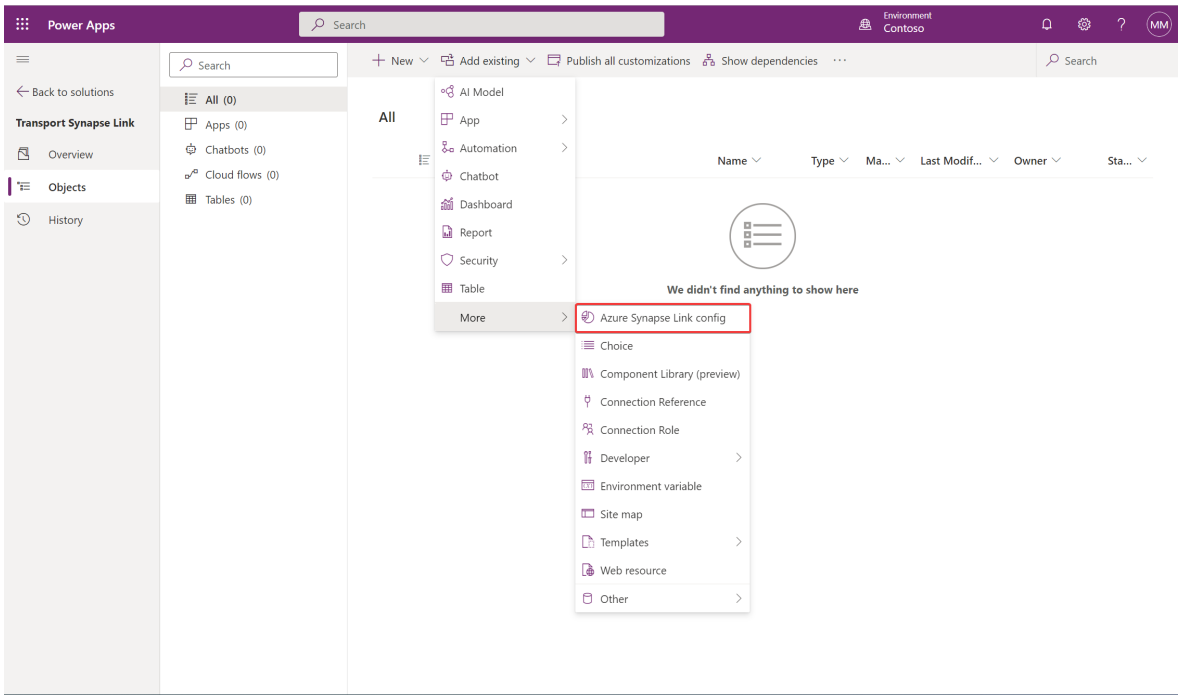
5. Repeat above steps in the destination environment. You need the **Azure Synapse Link for Dataverse** solution in both the source and destination environments.

Add an Azure Synapse Link configuration to a solution

1. Sign into Power Apps, select the source environment, and then on the left navigation pane, select **Solutions**. If the item isn't in the left navigation pane, select ...More and then select the item you want.

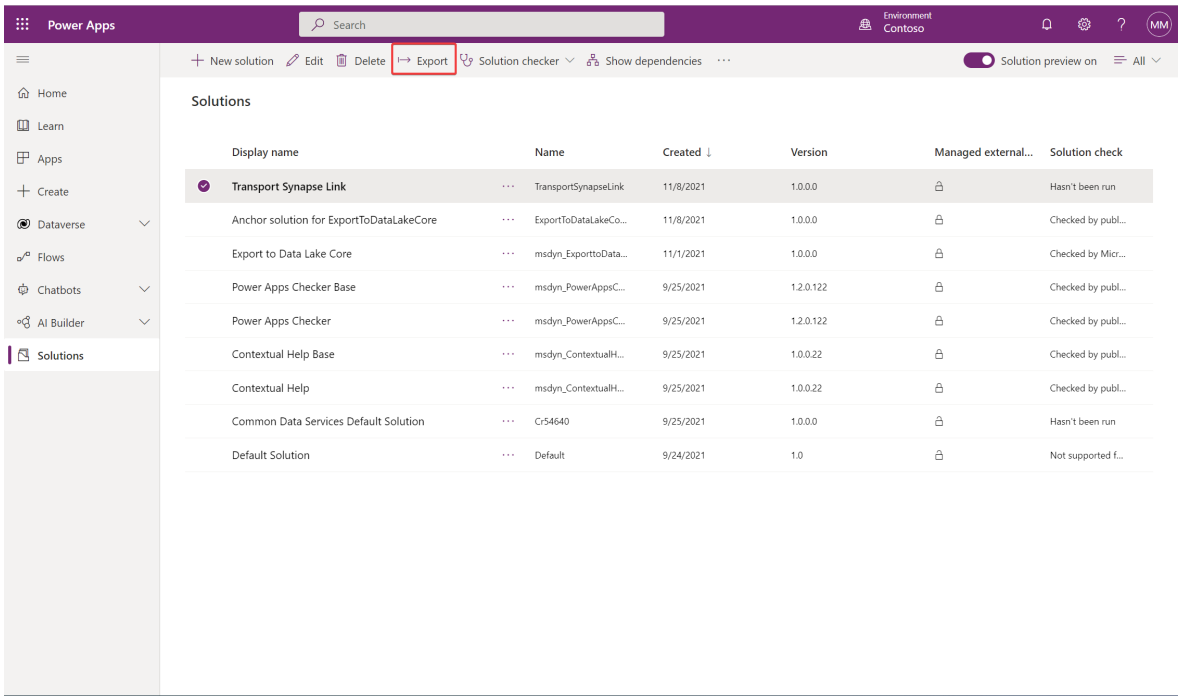
2. Select **New solution**. Provide a **Name**, **Publisher**, and **Version Number**.

3. Open the solution you created in the previous step, select **Add existing** > **More** > **Azure Synapse Link config**.



4. Select the link configurations that you want, and then select **Add**.

5. In the **Solutions** area, select the solution, and then on the command bar, select **Export**.

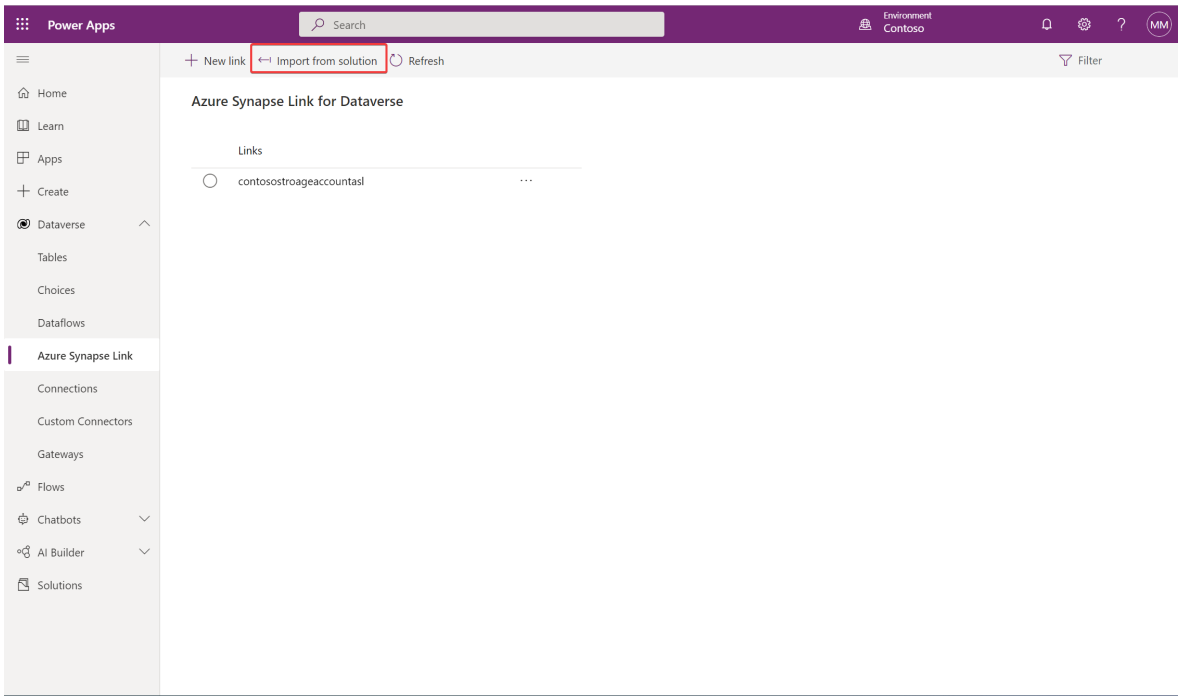


6. In the **Before you export** pane, select **Publish** to publish all changes before you export, and then select **Next**.

> **Note**
>
> Once the solution has been configured and exported, it is not updated automatically. Any updates you make to the configuration will require you to re-export and re-import the solution.

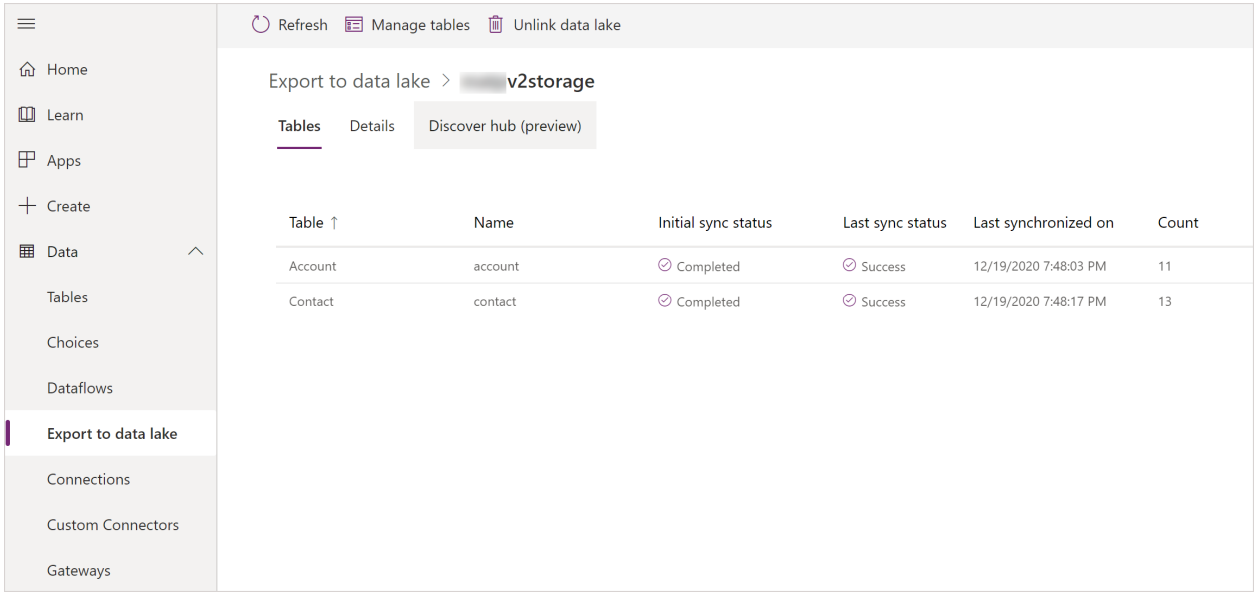
Import the solution that contains the Azure Synapse Link configuration

1. In the destination environment, import the solution under the **Solutions** tab.
2. In the destination environment, import the solution under the **Dataverse** > **Azure Synapse Link** tab.



Verify the Azure Synapse Link configuration

In the destination environment, verify that you can see the new Azure Synapse Link from the solution addition to the list of links.



See also

Azure Synapse Link for Dataverse

Manage your Azure Synapse Link during environment lifecycle events

Article • 02/15/2022

Azure Synapse Link for Dataverse provides a continuous pipeline of data from Dataverse to Azure Synapse Analytics and/or Azure Data Lake. Once the link is created it continuously writes all the initial data as well as any incremental data. When you make changes to your Power Platform environment, this can affect the state of your Azure Synapse Link for Dataverse. This article explains the behavior of the Azure Synapse Link for Dataverse and possible user action during environment lifecycle events.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Environment lifecycle events overview

Environment operation	Synapse link	Data	User action
Backup and restore	Stops syncing	Maintained	Use solutions to transport the link across environments
Copy	Source continues syncing	Source is maintained	Use solutions to transport the link across environments
Delete	Stops syncing	Maintained	None
Edit the properties	Error	Maintained	Unlink and relink
Move	Source stops syncing	Source is maintained	Use solutions to transport the link across environments
Recover	Continue syncing	Maintained	None
Reset	Removed	Maintained	Delete data before recreating the link

Backup and restore an environment

Azure Synapse Link for Dataverse profiles are not part of an environment backup or the subsequent restore. Once an environment is restored, use solutions to export and import the configuration to preserve the configuration of the Azure Synapse Link across environments.

Copy an environment

Azure Synapse Link for Dataverse profiles are not copied as part of the copy an environment process. To preserve the configuration of the Azure Synapse Link across environments, use solutions to export and import the configuration.

Delete an environment

When an environment is deleted, all Azure Synapse Link profiles are deleted and the data sync is stopped. All data written at the time of deletion is still available in your Azure Synapse Analytics workspace and/or Azure Data Lake.

Edit the properties of an environment

Azure Synapse Link for Dataverse profiles will fail if the environment name or URL environment properties are changed. You will need to unlink the Azure Synapse Link, optionally deleting the file system, and create a new link.

Move an environment

If an environment is migrated from one tenant to another, all Azure Synapse Link profiles are deleted and the data sync is stopped. All data written at the time of deletion is still available in your Azure Synapse Analytics workspace and/or Azure Data Lake. To preserve the configuration of the Azure Synapse Link across environments, use solutions to export and import the configuration.

Recover an environment

When recovering a deleted environment within the grace period, all Azure Synapse Link for Dataverse will be available and will continue syncing data with no user action as long as the data in your Azure Synapse Analytics workspace and/or Azure Data Lake has not been modified.

Reset an environment

If an environment is reset, all Azure Synapse Link for Dataverse profiles will be removed from the environment. All the corresponding data in Azure Synapse Analytics workspace and/or Azure Data Lake will still be available but will not be synced. You may need to clean up the data in the Azure Synapse Analytics workspace and/or Azure Data Lake and you'll need to re-create the Azure Synapse Link for Dataverse.

See also

Azure Synapse Link for Dataverse

Query Azure Synapse Link for Dataverse data with serverless SQL pool

Article • 02/15/2022

You can use the Azure Synapse Link to connect your Microsoft Dataverse data to Azure Synapse Analytics to explore your data and accelerate time to insight. This article shows you how to query your Dataverse data with built-in serverless SQL pool in your Azure Synapse Analytics workspace.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

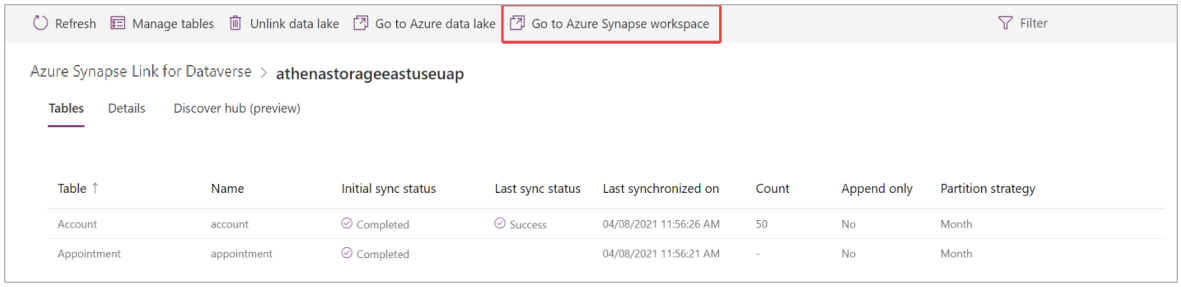
This section describes the prerequisites necessary to query your Dataverse data after using the Azure Synapse Link for Dataverse service.

- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.
- **Synapse administrator.** You must be granted the **Synapse Administrator** role access within Synapse studio.

Query your Dataverse data with serverless SQL pool

> **Note**
>
> Azure Synapse Link for Dataverse does not support the use of dedicated SQL pools at this time.

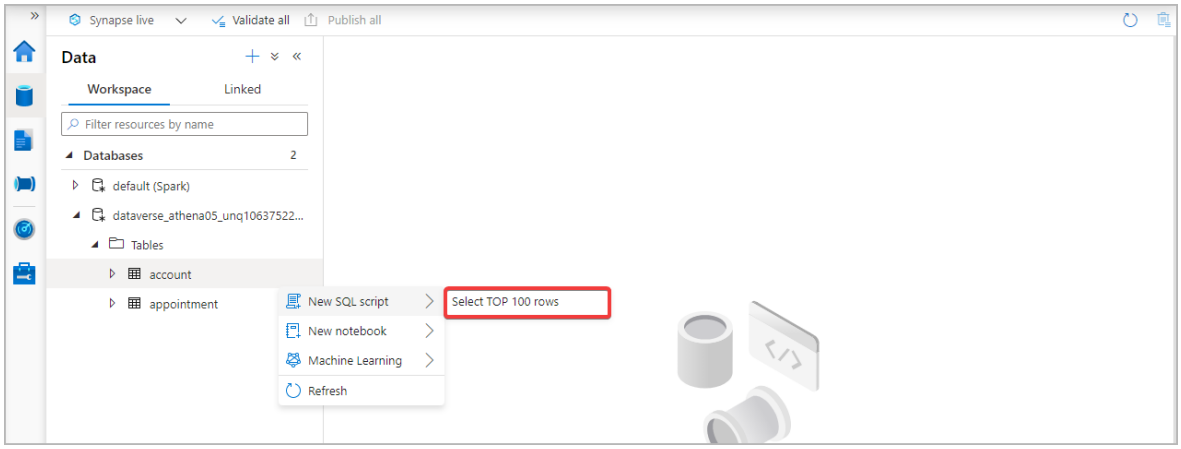
1. In Power Apps, select your desired Azure Synapse Link from the list, and then select **Go to Azure Synapse workspace**.



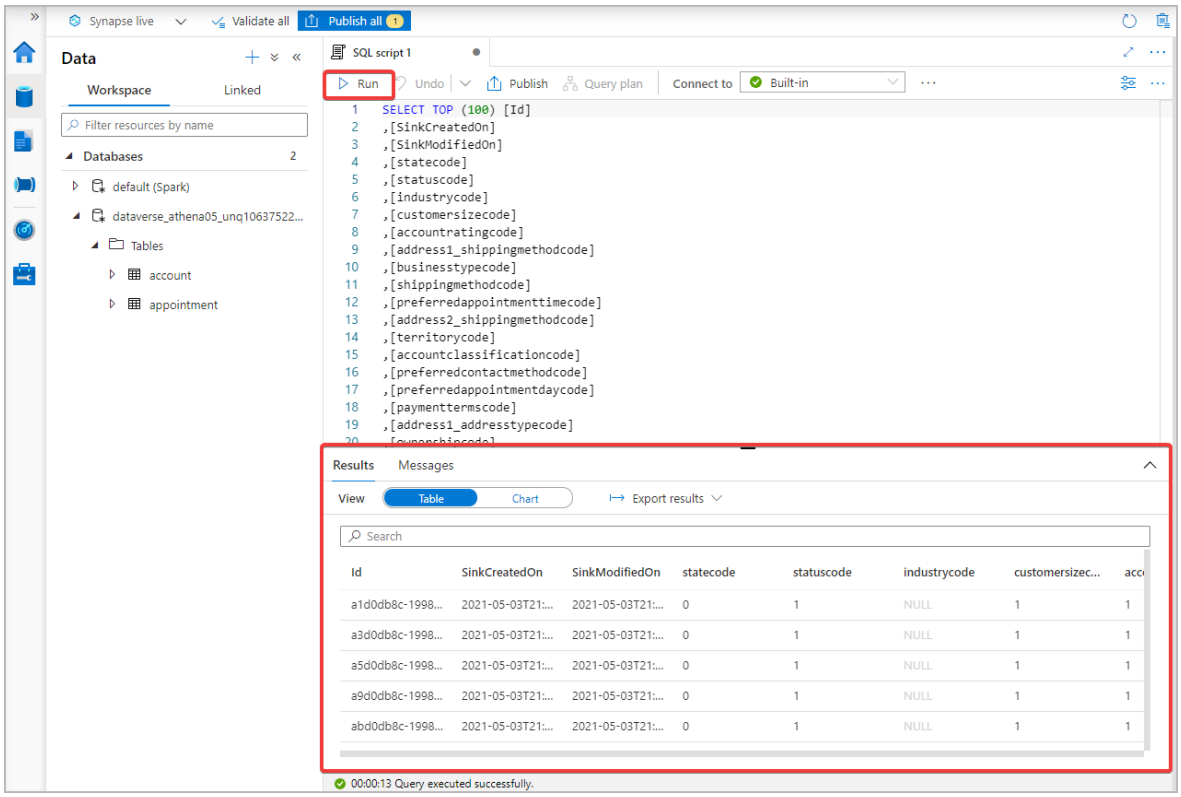
2. Expand **Lake database**, select your Dataverse container. Your exported tables are displayed under the **Tables** directory on the left sidebar.



3. Right-click the desired table and select **New SQL script** > **Select TOP 100 rows**.



4. Select **Run**. Your query results are displayed on the **Results** tab. Alternatively, you can edit the script to your needs.



Query multiple Dataverse databases with serverless SQL pool

> **Note**
>
> Querying multiple Dataverse databases requires that both Dataverse environments are in the same region.

1. Add another Azure Data Lake Storage Gen2 account as a Linked service to the same Azure Synapse Analytics workspace where the current link resides.
2. Follow the configuration steps to create a new Azure Synapse Link with the new Azure Synapse Analytics and Azure Data Lake combination.
3. Navigate to the shared Synapse workspace and expand **Databases**. Select one of the Dataverse containers. Your exported tables are displayed under the **Tables** directory on the left sidebar.
4. Right-click a table, and then select **New SQL script** > **Select TOP 100 rows**.
5. Edit the query to combine the two datasets. For instance, you can join the datasets based on a unique ID value.
6. Select **Run**. Your query results are displayed on the **Results** tab.

See also

Blog: Announcing Azure Synapse Link for Dataverse

Query and analyze the incremental updates with Azure Synapse Link for Dataverse

Article • 04/29/2025

Microsoft Dataverse data (including data from Dynamics 365 apps and finance and operations) can continuously change through create, update, and delete transactions. With the incremental update option, you can build incremental data pipelines that apply these changes to downstream systems and databases. Azure Synapse Link for Dataverse exports incremental data in time stamped folders that contain data changes within user-specified time intervals.

You can leverage incremental update feature for several scenarios:

- **Update a downstream datastore or a data warehouse**. You might need to apply changes from your Power Apps and Dynamics 365 data into a downstream datastore. Incremental update is a standard capability in most data transformation tools, such as Azure Data Factory. However, for the incremental update feature to work, you must identify the records that changed in source tables. The incremental update feature provides changed data as a set of files such that you don't need to detect changes by comparing before and after images of tables.

- **Analyze changes in large datasets**. If you need to analyze changes in large datasets, the incremental update feature provides a continuous stream of data in small batches such that you don't need to store all data. With this option you can drop stale and stagnant data to save data storage costs as well as track data changes relevant for a user-specified time period.

Azure Synapse Link for Dataverse also provides the option to export and maintain a replica of tables in your Azure Data Lake (Gen 2) storage. You can configure Azure Synapse Link to export incremental data in addition to exporting a replica of tables. Each configuration (known as a "Synapse Link profile") can export either tables or incremental data. While you can create multiple profiles, you can't configure both tables and incremental updates within the same profile.

> **Important**
>
> An initial time stamped folder is created when you enable this feature with a copy of your data. Subsequent timestamp and table folders are created only when there's a data update during the user-specified time interval.
>
> Once you create a Synapse Link profile with the incremental update feature, the configuration applies to all selected tables within the Synapse Link profile.

This feature can't be enabled with the option: **Connect to your Azure Synapse workspace**. For customers who require Azure Synapse analytics access, follow this instruction to setup the link: [Create an Azure Synapse Link for Dataverse with your Azure Synapse Workspace]

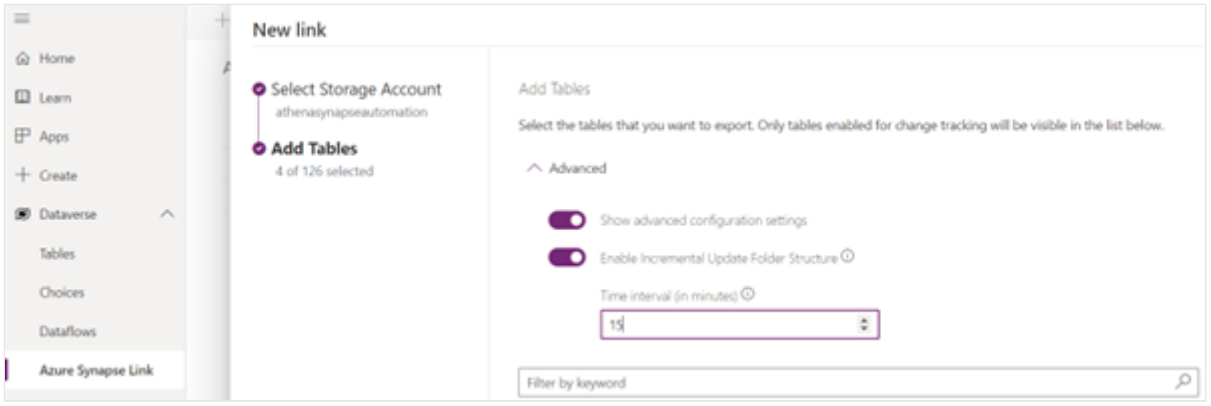
This feature is equivalent to the [Change feeds feature in export to data lake] built into Dynamics 365 finance and operations apps. Customers using the **Change feeds** feature have the option to enable a Synapse Link profile with change data without having to export table data.

Prerequisites

This guide assumes that you have already met the prerequisites to create an Azure Synapse Link. More information: Create an Azure Synapse Link for Dataverse with Azure Data Lake

Create a Synapse Link profile to export incremental data

1. Sign in to Power Apps and select your environment.
2. On the left navigation pane, select **Azure Synapse Link**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. On the command bar, select **+ New link**.
4. Select the **Subscription**, **Resource group**, and **Storage account**. Select **Next**.
5. Don't select the option **Connect to your Azure Synapse workspace**. If you choose this option, the incremental update feature is disabled.
6. Add the tables you want to export. If your Dataverse environment is linked to finance and operations apps, you can also select tables from finance and operations apps.
7. Select **Advanced**.
8. Turn on **Show advanced configuration settings** and **Enable Incremental Update Folder Structure**.
9. Enter the time interval (in minutes) for how often the incremental updates should be captured, and then select **Save**.



> **Note**
>
> The minimum time interval is 5 minutes. That means the incremental update folder is created every five minutes and contain the changes that occurred within the time interval. This setting is also configurable after the link creation via **Manage tables**. Maximum time interval is 1,140 minutes (or 24 hours).
>
> Ensure **Connect to your Azure Synapse workspace Azure Synapse workspace** isn't checked in the first page of setup.
>
> Incremental data in time stamped folders are stored as comma separated value text files (CSV files). You can't use the Delta conversion feature for incremental data and obtain incremental files in a Delta parquet format.

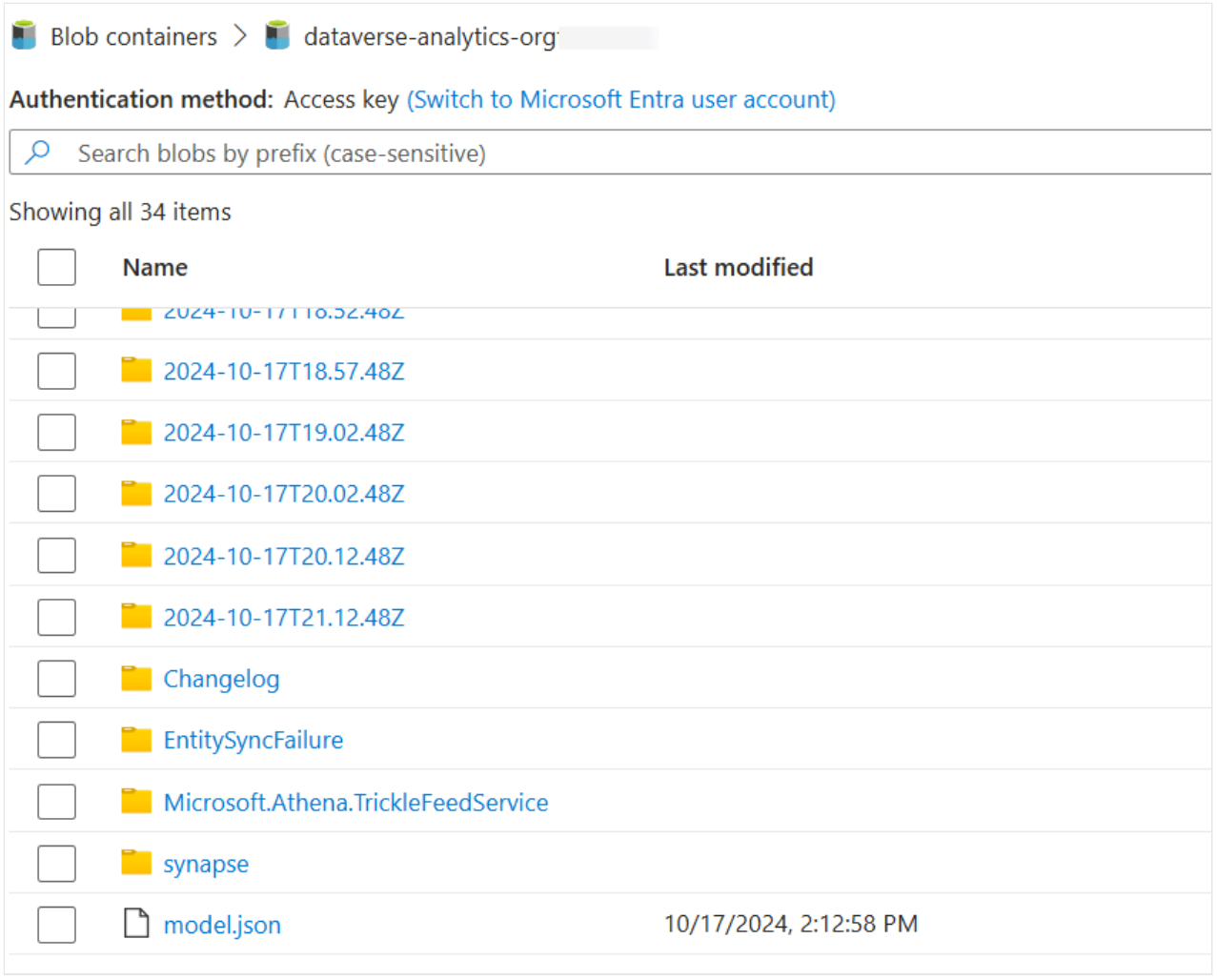
View incremental folder at Microsoft Azure Storage

When you create a Synapse Link profile with incremental data, the system makes an initial copy of all tables and stores it in the first incremental update folder. Once the initial copy is created, the system creates subsequent update folders with changed data. If there are no changes in any of the tables selected, you won't see incremental data folders.

To see incremental data folders in the storage account:

1. Select the desired Azure Synapse Link, and then select **Go to Azure data lake** on the command bar.
2. Select the **Containers** under **Data Storage**.
3. Select **dataverse**-*environmentName-organizationUniqueName*. Incremental updates folders are named by creation timestamp ("yyyy-MM-dd'T'HH:mm:ss.SSSz") in UTC. Notice that the time difference between time stamped folders is the time interval specified by you in advanced settings.

4. Within each time stamped folder, there are folders for each table. Not all tables chosen might have changed during the time interval and you only see folders corresponding to tables whose data changed.



> **Note**
>
> Due to the retry mechanism features, an extra empty timestamp folder might be created within the user-specified time interval.

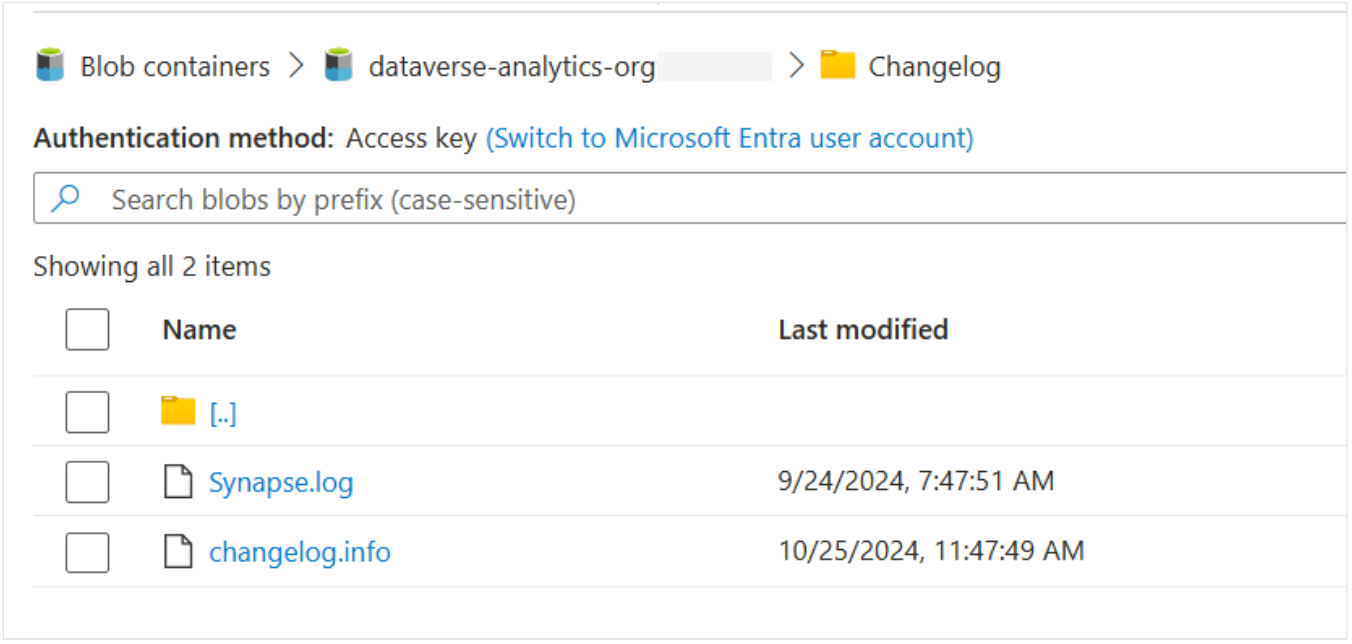
Consume incremental data

You can copy incremental data into an Azure SQL Database or a data warehouse using data integration tools such as Azure Data Factory or Azure Synapse Analytics pipelines. We provide a sample data pipeline that can be used for this purpose. For more information:Copy Dataverse data into Azure SQL.

If you're a Dynamics 365 finance and operations apps customer transitioning from the change feeds feature you can use Data integration sample tools provided in GitHub to update existing data pipelines used with the change feeds feature.

You can also build your own data pipeline to consume incremental data. However, you need to consider the following best practices when designing your own pipeline:

- **Consume data from previous time stamped folders only**: This way, you can avoid read-write conflicts with the Synapse Link service, which might be continuously updating data in the current folder. You can find the current folder by viewing the **Changelog/changelog.info** file. This file is a read-only file which contains a single row with the folder name that is currently updated. You shouldn't update this file as it can cause system instability.
- You can view the **model.json** file located within each time stamped folder to read metadata such as column names for the data contained in table folders. Notice that each model.json file in the folder located within time stamped folders contain metadata for all the tables, not just the tables contained within the time stamped folder.
- Avoid using other log files such as the Synapse.log file. This file is used for internal purposes and might not reflect accurate data.
- Consider deleting obsolete incremental folders from your Azure Data lake after you have finished processing. At present, Synapse Link maintains a lease on these files in Azure Storage to recover from any failures. The system might release the lease after some time. You should only delete incremental folders that are *older than 24 hours* to avoid any conflicts with system operation.
- You shouldn't modify or delete the "current folder" that is the folder contained in the **Changelog/changelog.info** file. If you change this file, the system pauses processing data.



See also

What is Azure Synapse Link for Dataverse?

Create a view of your Azure Synapse Link for Dataverse data

Summarize this article for me

Use the Azure Synapse Link to connect your Microsoft Dataverse data to Azure Synapse Analytics to explore your data and accelerate time to insight. This article shows you how to create a view of your Dataverse data in your Azure Synapse Analytics workspace.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

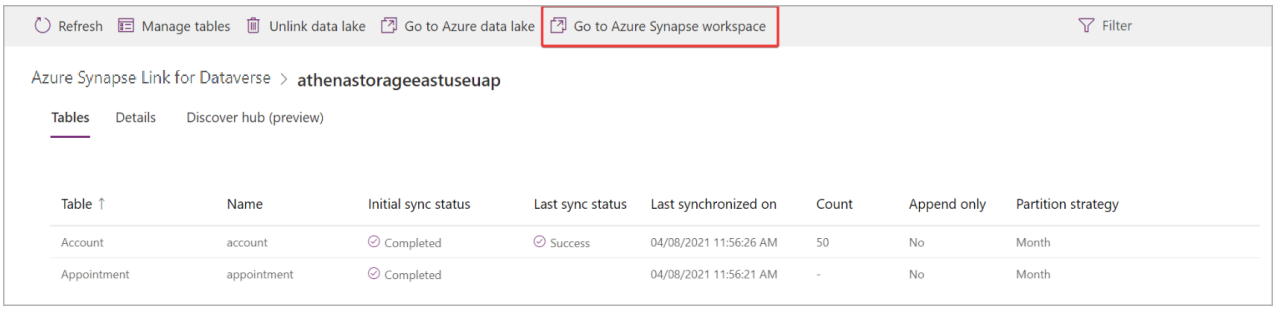
Prerequisites

This section describes the prerequisites necessary to create a view of your Dataverse data after using the Azure Synapse Link for Dataverse service.

- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.
- **Synapse administrator.** You must be granted **Synapse Administrator** role access within Synapse studio.

Create a view of your Dataverse data

1. In Power Apps, select your Azure Synapse Link from the list, and then select **Go to Azure Synapse workspace**.



2. Create User-defined schema within Synapse SQL

```SQL
CREATE SCHEMA [SCHEMA NAME]
GO
```

3. Run a create view script by specifying a **view name**, the **column names**, **Dataverse database name** under the new **schema**, and then exported **table name**.

```SQL
CREATE OR ALTER VIEW [SCHEMA NAME].[VIEW NAME]
AS
SELECT [COLUMN NAMES]
FROM [DATAVERSE DATABASE NAME].[dbo].[TABLE NAME]
```

Use a view of your Dataverse data

You can use views in your queries the same way you use views in SQL Server queries. The view is created and available under the **Views** folder in SQL Server Management Studio (SSMS).

> **Important**
>
> Custom database objects created by a customer within the Azure Synapse Link database are deleted once the unlink is completed.
>
> Customers will need to implement their own version control of their custom database objects scripts to ensure they have a backup.

See also

Blog: Announcing Azure Synapse Link for Dataverse

Last updated on 02/23/2026

Transform Azure Synapse Link for Dataverse data with Apache Spark

Article • 08/25/2022

You can use the Azure Synapse Link to connect your Microsoft Dataverse data to Azure Synapse Analytics to explore your data and accelerate time to insight. This article shows you how to transform your Dataverse data using the Apache Spark engine available in your Synapse workspace.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

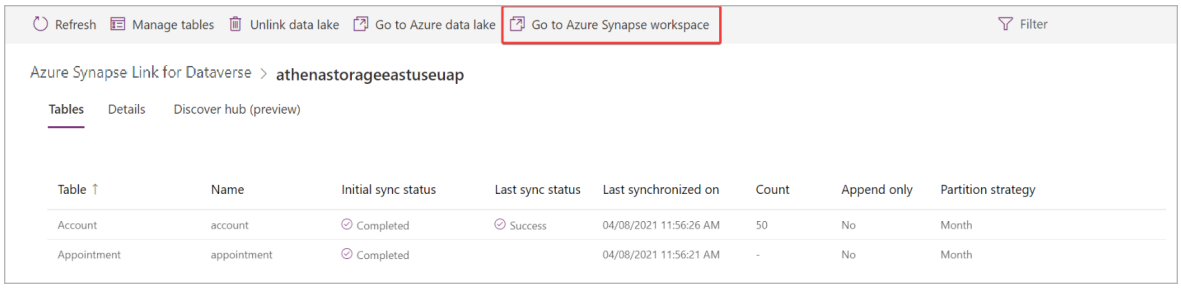
Prerequisites

This section describes the prerequisites necessary to transform Dataverse data with Apache Spark after using the Azure Synapse Link for Dataverse service.

- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.
- **Synapse administrator.** You must be granted the **Synapse Administrator** role access within Synapse studio.

Transform your data with an Apache Spark notebook

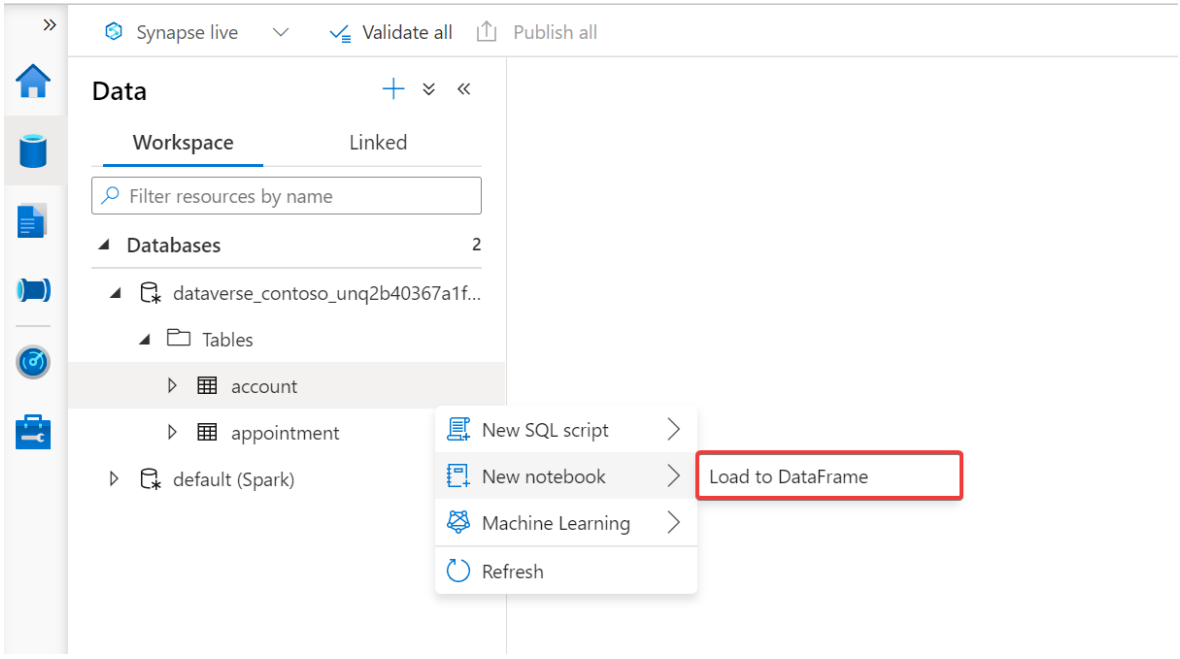
1. In Power Apps, select your desired Azure Synapse Link from the list, and then select **Go to Azure Synapse workspace**.



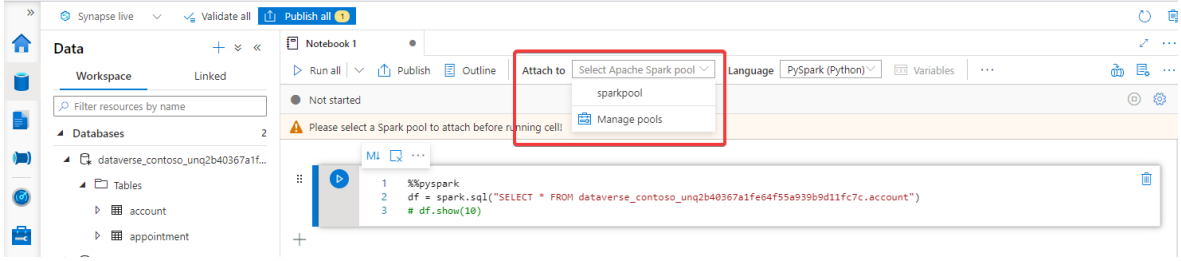
2. Expand **Databases**, select your Dataverse container. Your exported tables are displayed under the **Tables** directory on the left sidebar.



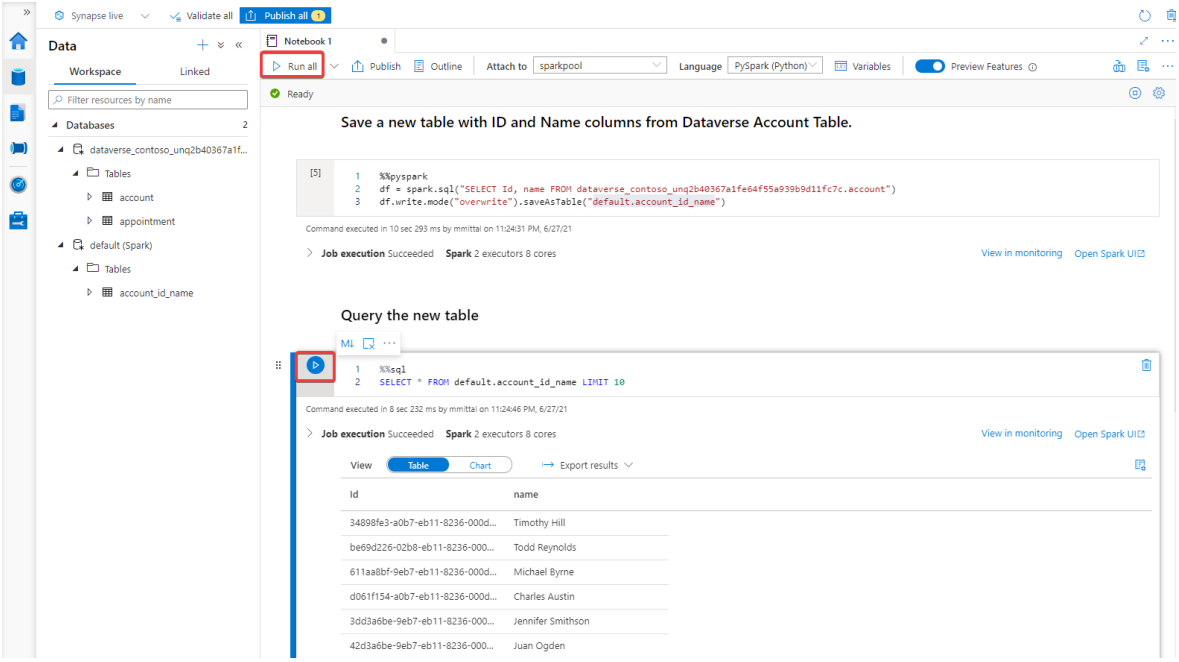
3. Right-click the desired table and select **New notebook** > **Load to DataFrame**.



4. Attach the notebook to an Apache Spark pool by selecting a pool from the drop-down menu. If you do not have an Apache Spark pool, select **Manage pools** to create one.



5. Add code cells to transform your data. Run individual cells by selecting the play button at the left of each cell or run all the cells in succession by selecting **Run all** from the top bar.



See also

Blog: Announcing Azure Synapse Link for Dataverse

What is Apache Spark in Azure HDInsight

Copy Dataverse data into Azure SQL

Article • 02/08/2023

Use the Azure Synapse Link to connect your Microsoft Dataverse data to Azure Synapse Analytics to explore your data and accelerate time to insight. This article shows you how to run Azure Synapse pipelines or Azure Data Factory to copy data from Azure Data Lake Storage Gen2 to an Azure SQL Database with incremental updates feature enabled in Azure Synapse Link.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics. This template is a code sample. We encourage you to use this template as guidance to test out the functionality of retrieving data from Azure Data Lake Storage Gen2 to Azure SQL Database using the pipeline provided.

Prerequisites

1. Azure Synapse Link for Dataverse. This guide assumes that you've already met the prerequisites to create an Azure Synapse Link with Azure Data Lake. More information: Prerequisites for an Azure Synapse Link for Dataverse with your Azure Data Lake
2. Create an Azure Synapse Workspace or Azure Data Factory under the same Microsoft Entra tenant as your Power Apps tenant.
3. Create an Azure Synapse Link for Dataverse with the **incremental folder update enabled** to set the time interval. More information: Query and analyze the incremental updates
4. Microsoft.EventGrid provider needs to be registered for trigger. More information: Azure portal. Note: If you are using this feature in Azure Synapse Analytics, ensure that your subscription is also registered with Data Factory resource provider, otherwise you'll get an error stating that the creation of an "Event Subscription" failed.
5. Create an Azure SQL database with the **Allow Azure services and resources to access this server** property enabled. More information: What should I know when setting up my Azure SQL Database (PaaS)?
6. Create and configure an Azure integration runtime. More information: Create Azure integration runtime - Azure Data Factory & Azure Synapse

> **Important**
>
> Using this template might incur additional costs. These costs are related to the usage of Azure Data Factory or Synapse workspace pipeline and are billed on a monthly basis. The cost of using pipelines mainly depends on the time interval for incremental update and the data volumes. To plan and manage the cost of using this feature, go to: Monitor costs at pipeline level with cost analysis
>
> It's important to take these additional costs into consideration when deciding to use this template as they are not optional and must be paid in order to continue using this feature.

Use the solution template

Synapse Analytics

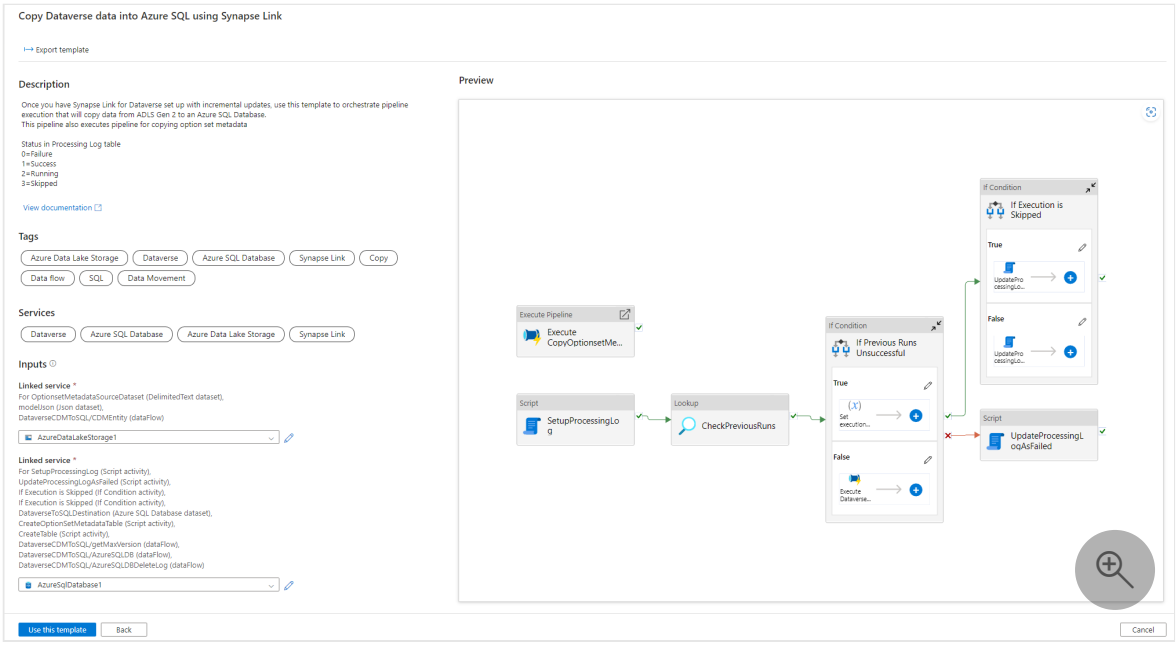
1. Go to the Azure portal and open Azure Synapse workspace.
2. Select **Integrate** > **Browse gallery**.
3. Select **Copy Dataverse data into Azure SQL using Synapse Link** from the integration gallery.

Configure the solution template

1. Create a linked service to Azure Data Lake Storage Gen2, which is connected to Dataverse using the appropriate authentication type. To do this, select **Test connection to validate the connectivity,** and then select **Create**.

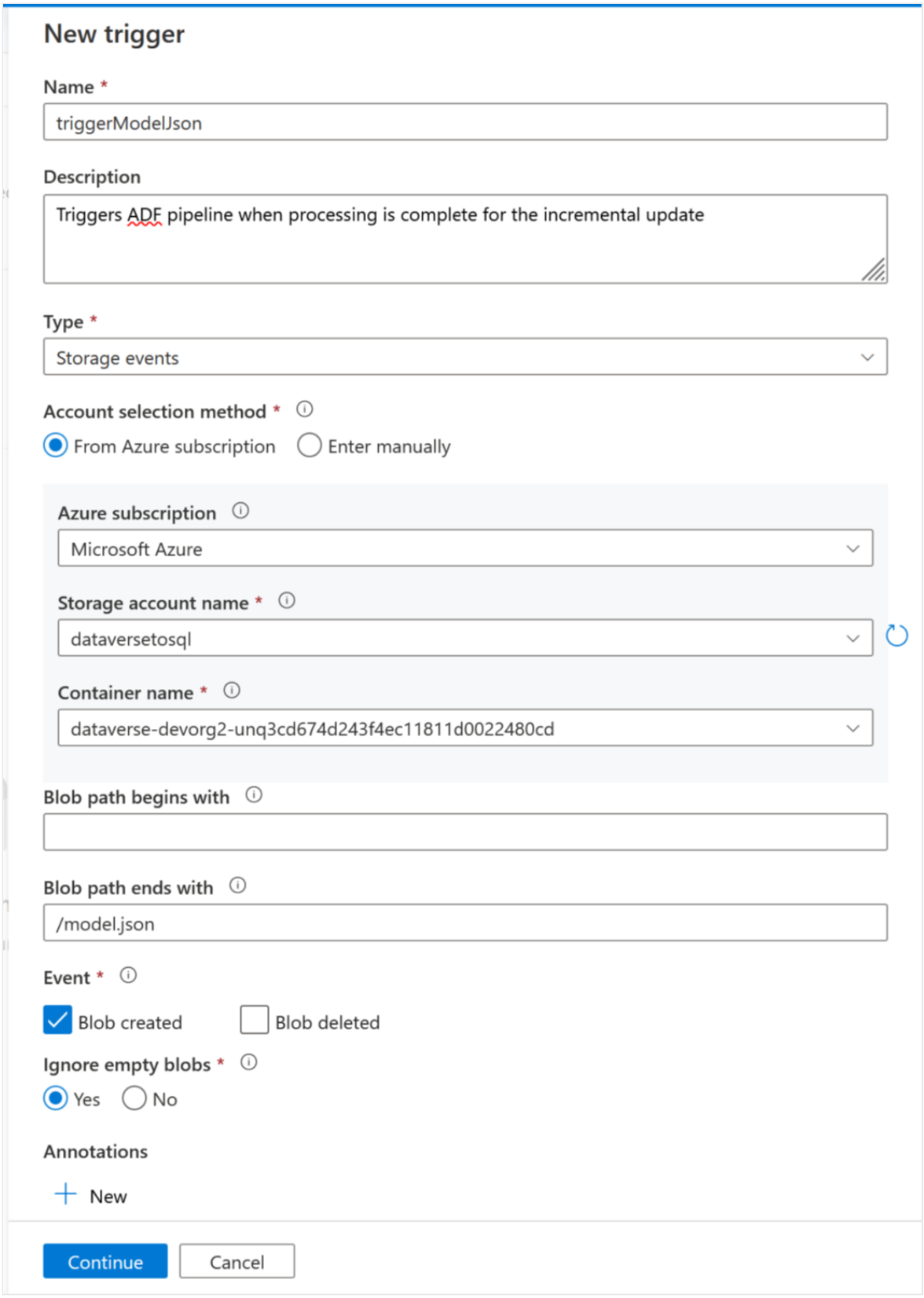
2. Similar to the previous steps, create a linked service to Azure SQL Database where Dataverse data will be synced.

3. Once **Inputs** are configured, select **Use this template**.



4. Now a trigger can be added to automate this pipeline, so that the pipeline can always process files when incremental updates are completed periodically. Go to **Manage** > **Trigger**, and create a trigger using the following properties:

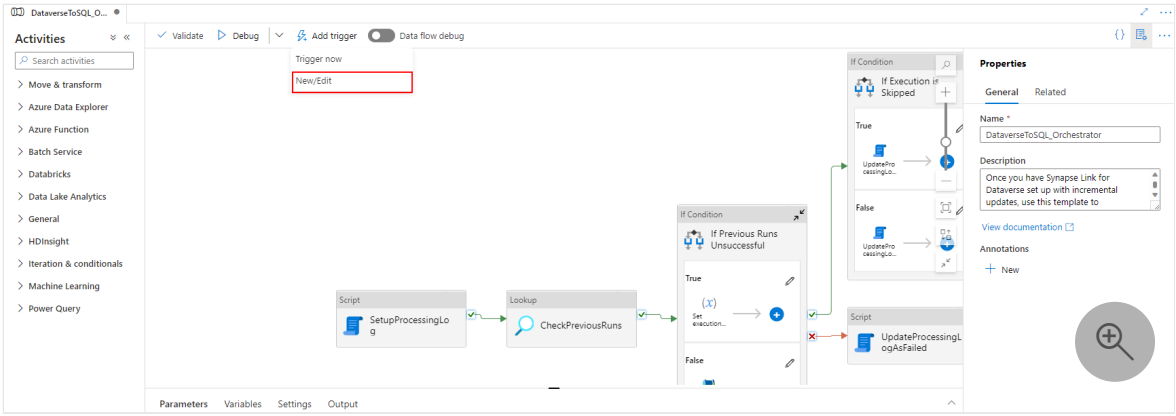
    - **Name**: Enter a name for the trigger, such as *triggerModelJson*.
    - **Type**: **Storage events**.
    - **Azure subscription**: Select the subscription that has Azure Data Lake Storage Gen2.
    - **Storage account name**: Select the storage that has Dataverse data.
    - **Container name**: Select the container created by Azure Synapse Link.
    - **Blob path ends with**: */model.json*
    - **Event**: **Blob created**.
    - **Ignore empty blobs**: **Yes**.
    - **Start trigger**: Enable **Start trigger on creation**.



5. Select **Continue** to proceed to the next screen.

6. On the next screen, the trigger validates the matching files. Select **OK** to create the trigger.

7. Associate the trigger with a pipeline. Go to the pipeline imported earlier, and then select **Add trigger** > **New/Edit**.



8. Select the trigger in the earlier step, and then select **Continue** to proceed to the next screen where the trigger validates the matching files.

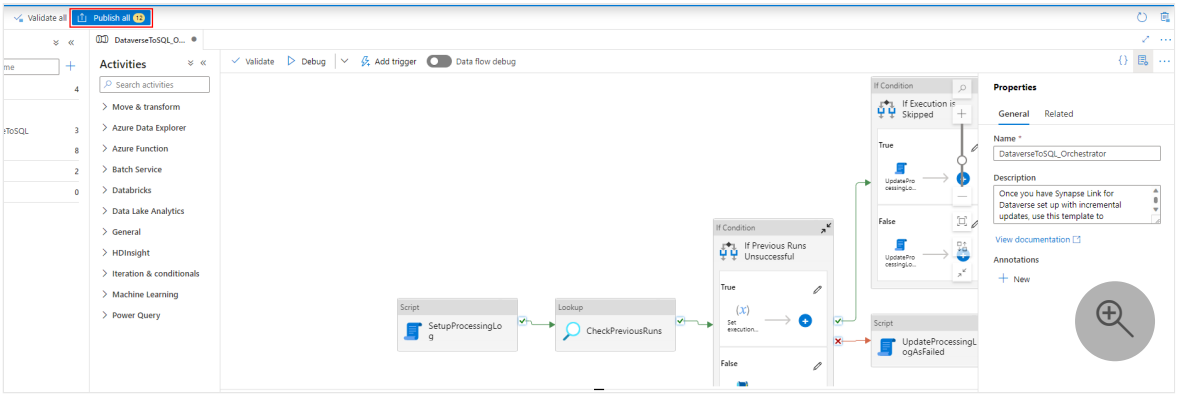
9. Select **Continue** to proceed to the next screen.

10. In the **Trigger Run Parameter** section, enter the below parameters, and then select **OK**.

    - **Container**: `@split(triggerBody().folderPath,'/')[0]`
    - **Folder**: `@split(triggerBody().folderPath,'/')[1]`

11. After associating the trigger with the pipeline, select **Validate all**.

12. Once validation succeeds, select **Publish All**.



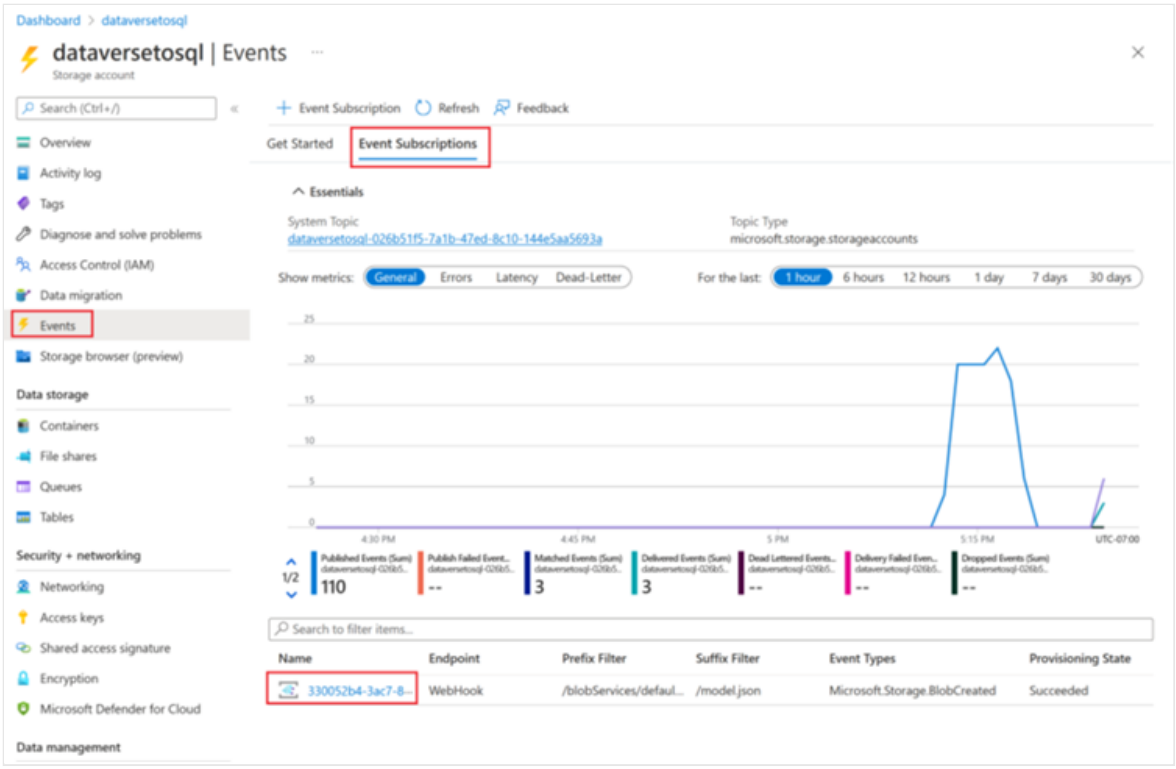
13. Select **Publish** to publish all the changes.

Add an event subscription filter

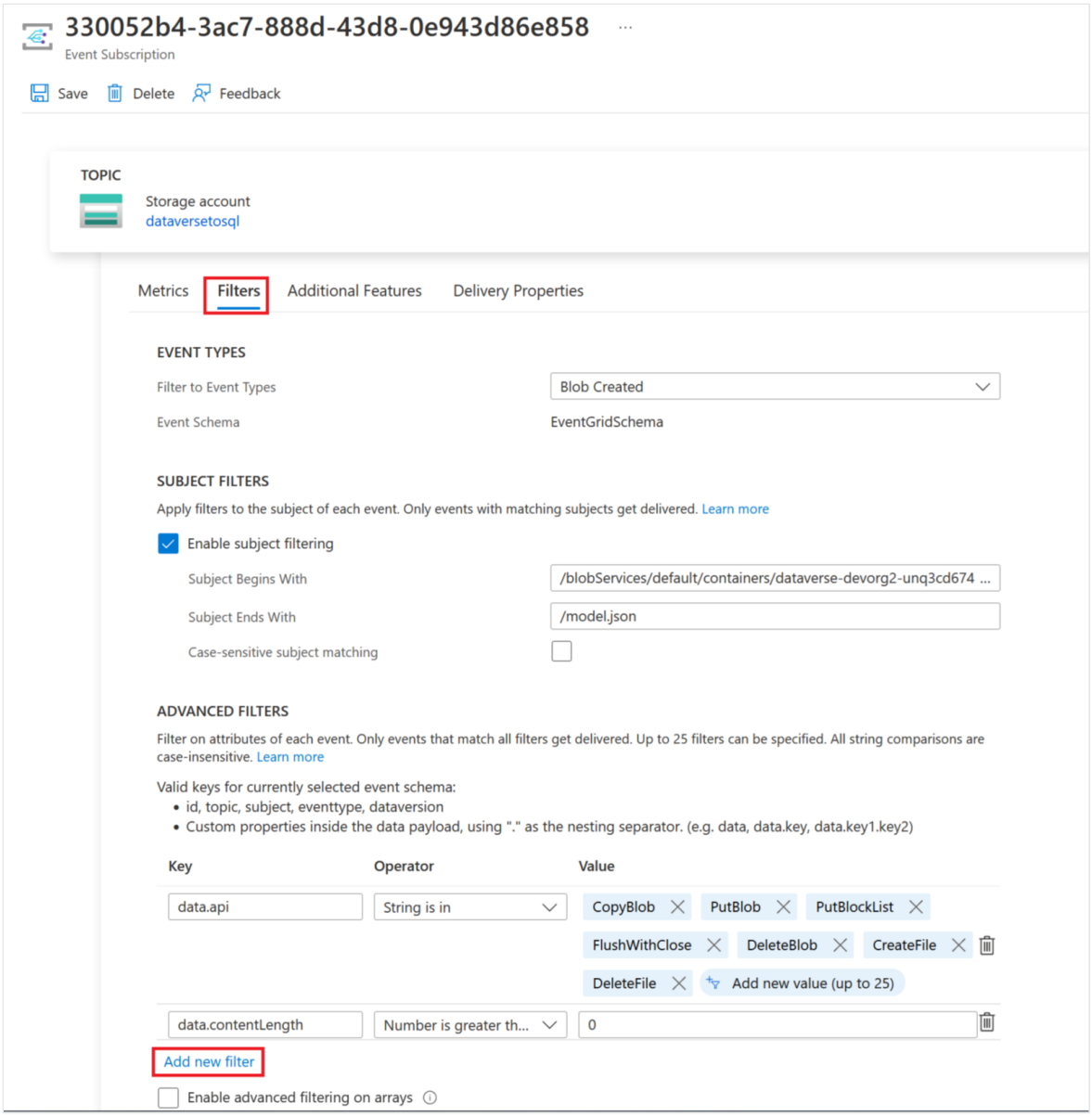
To ensure that the trigger fires only when model.json creation is complete, advanced filters need to be updated for the trigger's event subscription. An event is registered against the storage account the first time the trigger runs.

1. Once a trigger run completes, go to storage account > **Events** > **Event Subscriptions**.

2. Select the event that was registered for the model.json trigger.



3. Select the **Filters** tab, and then select **Add new filter**.

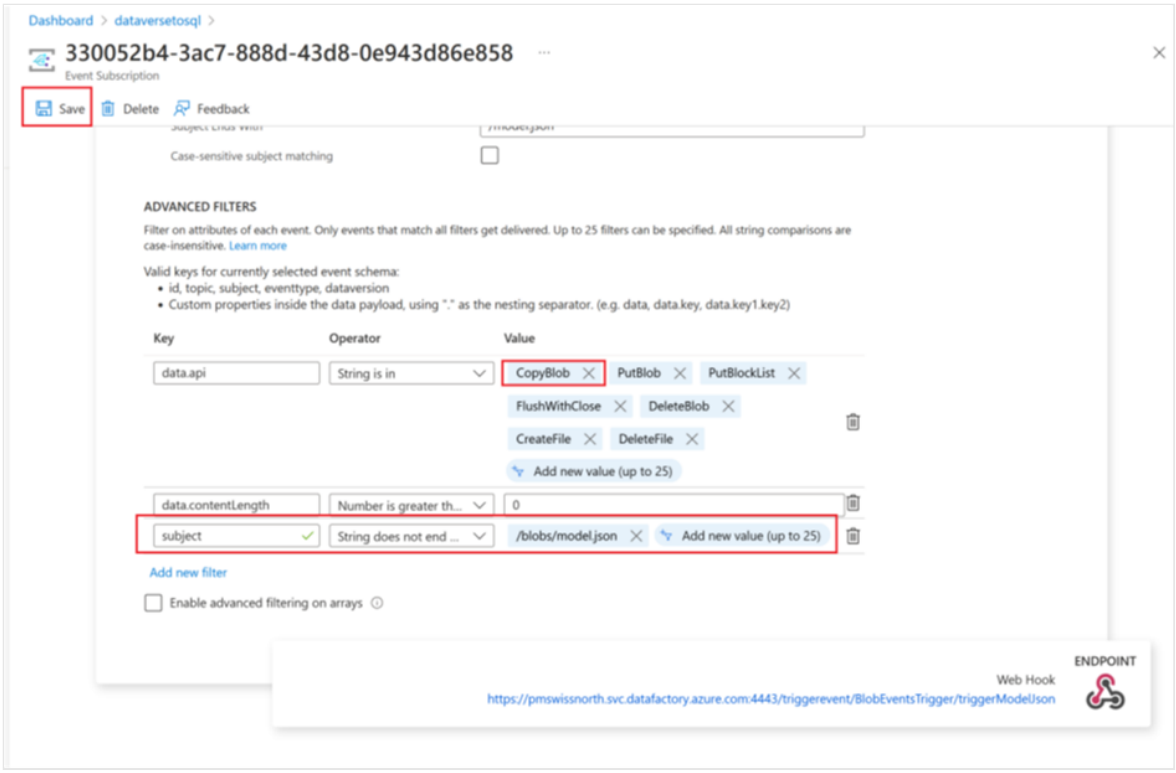


4. Create the filter:

   - **Key**: **subject**
   - **Operator**: **String does not end with**
   - **Value**: */blobs/model.json*

5. Remove the **CopyBlob** parameter from **data.api Value** array.

6. Select **Save** to deploy the additional filter.



See also

Blog: Announcing Azure Synapse Link for Dataverse

Visualize Azure Synapse Link for Dataverse data with Power BI

Article • 02/15/2022

You can use the Azure Synapse Link to connect your Microsoft Dataverse data to Azure Synapse Analytics to explore your data and accelerate time to insight. This article shows you how to generate a Power BI report by connecting to the serverless SQL endpoint from your Azure Synapse Analytics workspace.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

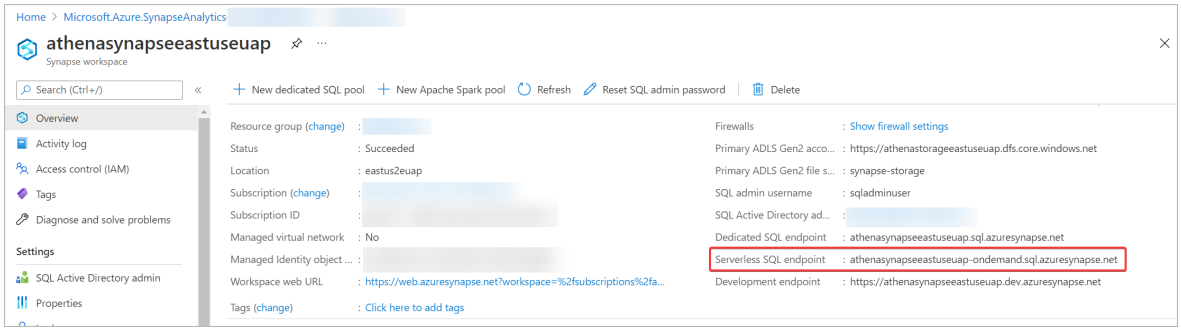
This section describes the prerequisites necessary to access Dataverse choices with Power BI after using the Azure Synapse Link for Dataverse service.

- **Power BI Desktop**. Get it now
- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.

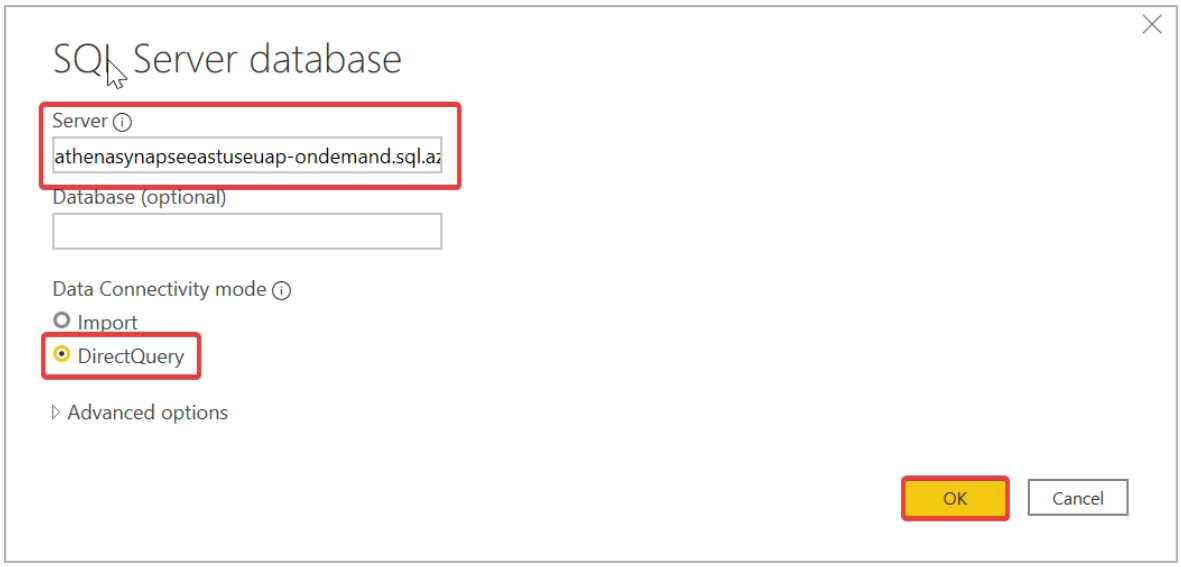
Connect to your Azure Synapse Analytics workspace to Power BI

1. Open Power BI Desktop.
2. Select **Get data** > **More...**.
3. Select **Azure** > **Azure Synapse Analytics (SQL DW)** > **Connect**.

4. Go to your Azure Synapse Analytics workspace and copy the **Serverless SQL endpoint**.

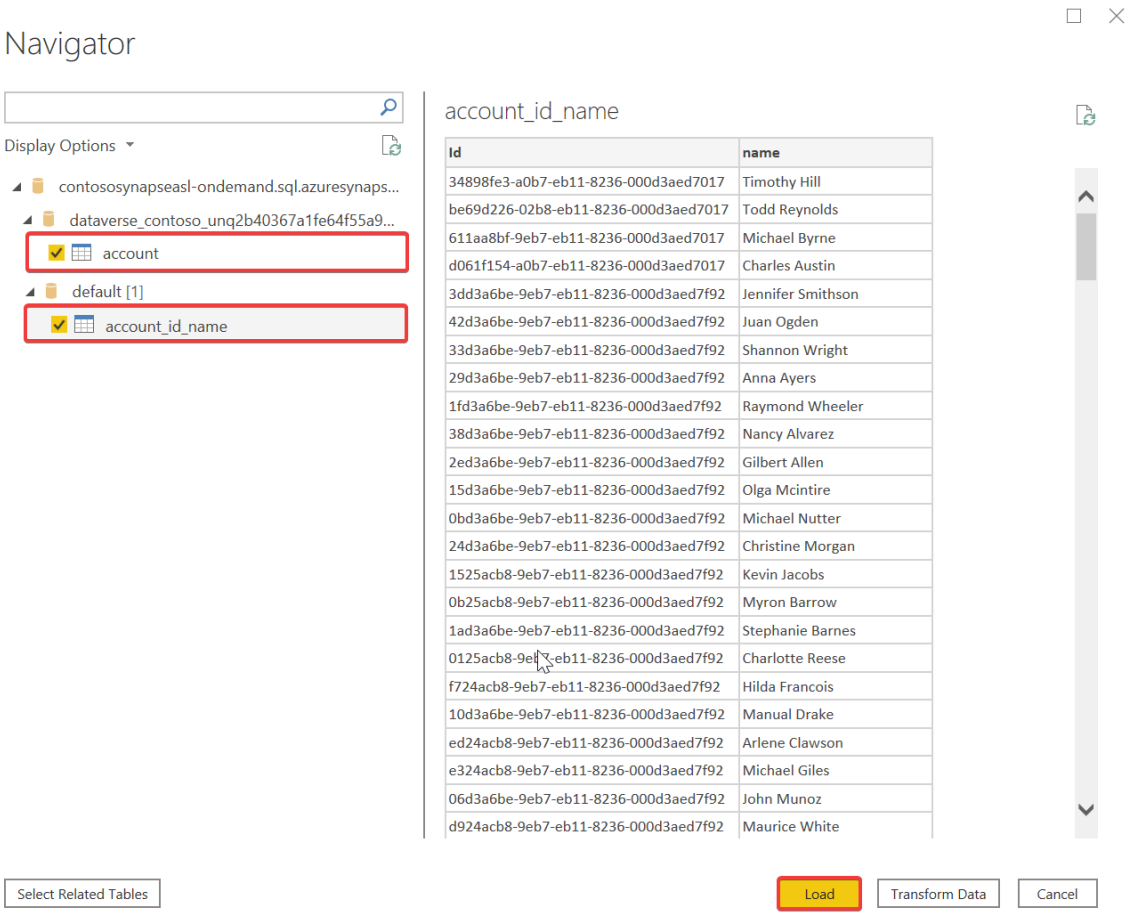


5. Paste the endpoint for the **Server**. Select **DirectQuery** for the **Data Connectivity mode**. Select **OK**.



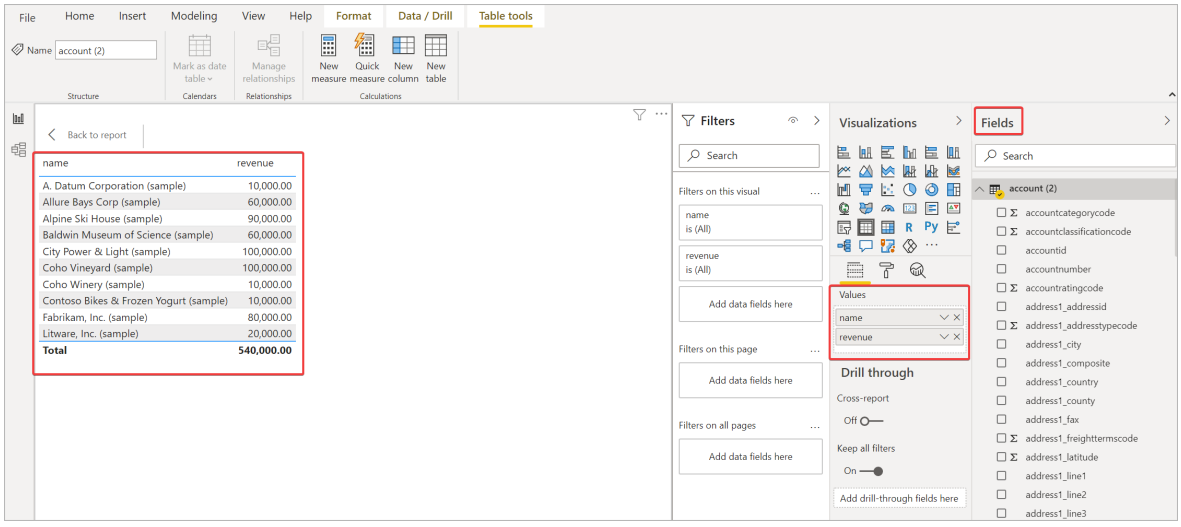
6. If prompted, sign in with your Microsoft Account.

7. Select the container and the preferred table from the Navigator, and then select **Load**. Optionally, expand *default* to access tables created with an Apache Spark notebook.



> **Note**
>
> If you receive a credentials error, go to **File** > **Options and settings** > **Data source settings** and clear the permissions for the serverless SQL endpoint. Repeat the above steps.

8. Select fields from the **Fields** pane to create a simple table or create other visualizations.



See also

Blog: Announcing Azure Synapse Link for Dataverse

Copy exported Dataverse data to dedicated SQL pool

Article • 02/15/2022

After creating a continuous pipeline of data from Dataverse to your Synapse workspace with Azure Synapse Link for Dataverse, you can copy the data to dedicated SQL pool with Synapse Pipelines.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

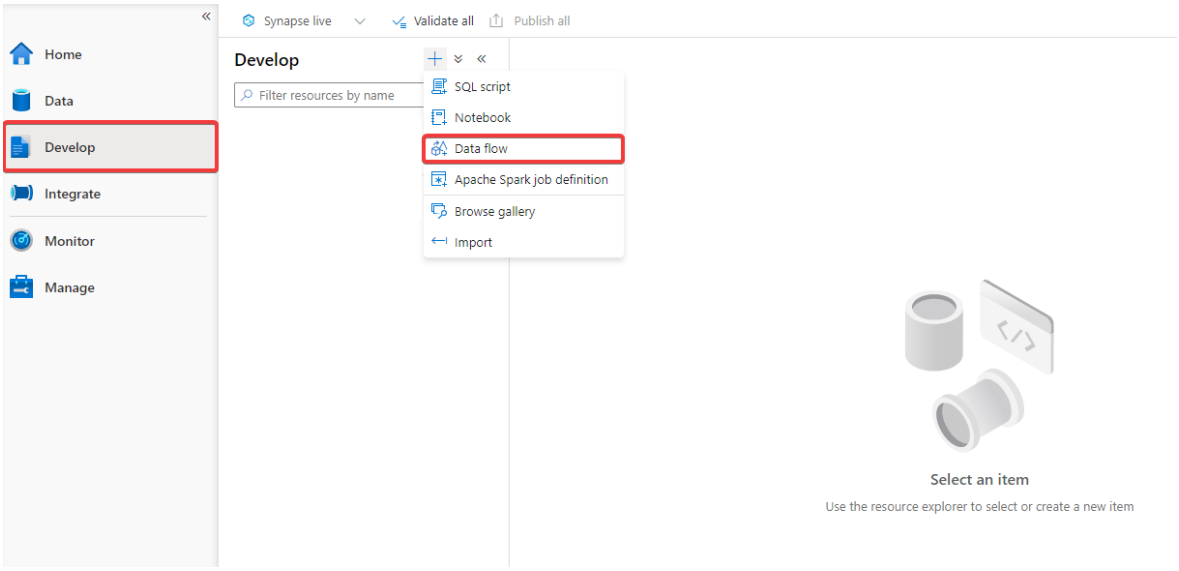
This section describes the prerequisites necessary to copy your Dataverse data to dedicated SQL pool after using the Azure Synapse Link for Dataverse service.

- **Azure Synapse Link for Dataverse.** This guide assumes that you've already exported Dataverse data by using Azure Synapse Link for Dataverse.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.
- **Synapse administrator.** You must be granted the **Synapse Administrator** role access within Synapse studio.
- **Dedicated SQL Pool.** This guide assumes that you've already created a dedicated SQL pool. You can create a dedicated SQL pool under the **Manage** tab in your Synapse workspace.

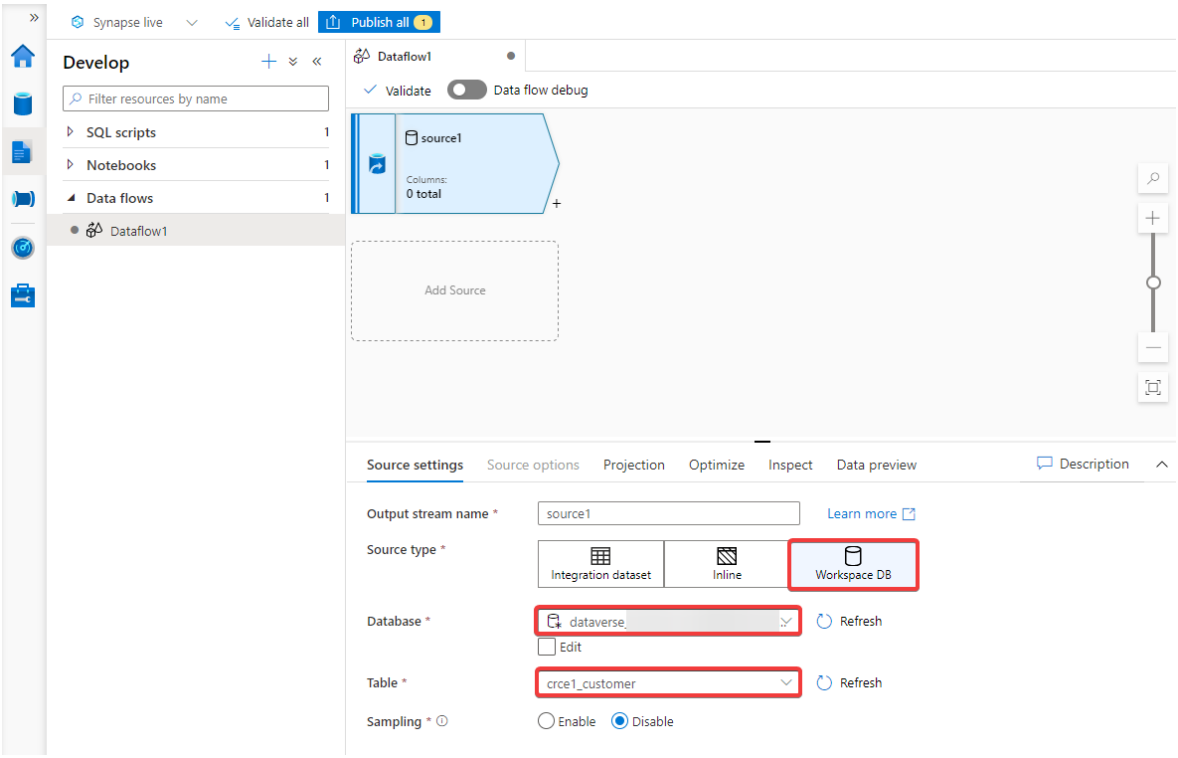
Copy the exported Dataverse data to dedicated SQL pool

1. Navigate to your Azure Synapse Analytics workspace.

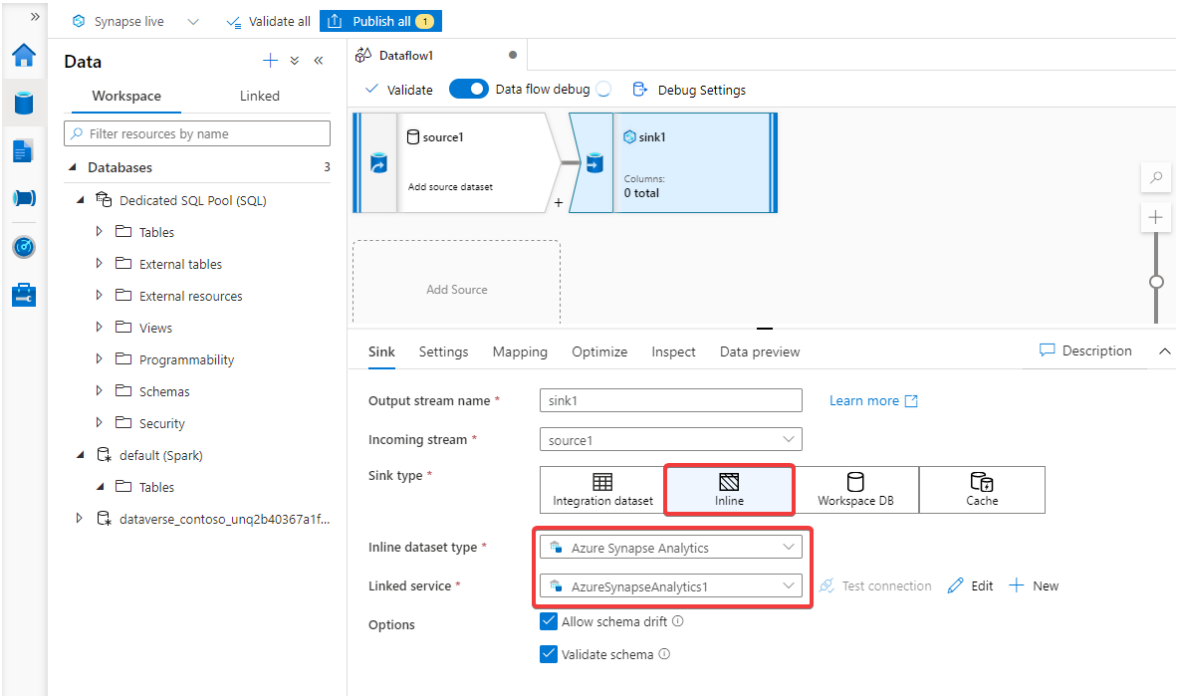
2. Open the **Develop** tab and create a new **Data flow**. Turn on **data flow debug**. This can take several minutes to complete.



3. Create a **Source** and specify the **Source type** as **Workspace DB** and select the **Database** that contains the Dataverse data and **Table** you want to copy.



4. Create a **Sink** and specify the **Sink type** as **Inline** and **Inline dataset type** as **Azure Synapse Analytics**. Select your dedicated SQL pool as the **Linked service**. If the dedicated SQL pool is not already a linked service, create a new connection.



5. Under the **Settings** tab, **Refresh** the **Schema Name**. Select *sys* and for the **Table name** select *all_columns*. Optionally, configure the other settings.

6. Navigate to the **Integrate** tab of the Synapse workspace and create a new **Pipeline**.

7. Expand the **Move & Transform** activity and drag and drop **Data flow** into the workspace.

8. Open the **Settings** and ensure that you have selected the data flow created in the previous steps.

9. Run your pipeline. Optionally, **Add trigger** to specify a time for the pipeline to run.

See also

Azure Synapse Link for Dataverse

What is dedicated SQL pool

Read the incremental updates of Dataverse data

Article • 02/15/2022

After creating a continuous pipeline of data from Microsoft Dataverse to your Synapse workspace with Azure Synapse Link for Dataverse, you can read the incremental updates of a specified time interval. Every Dataverse table exported with Azure Synapse Link for Dataverse contains a *SinkModifiedOn* column which can be used to get the incremental updates.

> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

This section describes the prerequisites necessary to read the incremental updates of your exported Dataverse data.

- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse and that all tables are syncing with **Append-only** mode.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.
- **Synapse administrator.** You must be granted the **Synapse Administrator** role access within Synapse studio.

Read the Incremental Updates of your Dataverse data

1. Navigate to your Azure Synapse Analytics workspace.
2. Select **Develop** from the left side panel, then select + > **SQL script**.

3. Paste the following SQL query and replace **CONTAINER_NAME** with the name of the container, **TABLE_NAME** with the name of the Dataverse table, and **TIMESTAMP_START** and **TIMESTAMP_END** with the time interval in UTC format (YYYY-MM-DDTHH:MM:SS).

```SQL
    SELECT *
    FROM [CONTAINER_NAME].[dbo].[TABLE_NAME]
    WHERE [SinkModifiedOn] >= TIMESTAMP_START AND [SinkModifiedOn] <=
TIMESTAMP_END
```

4. **Run** the query and a table containing the incremental updates to the Dataverse table from the specified time interval will be displayed.

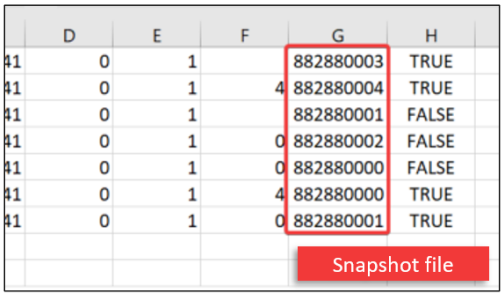
See also

Azure Synapse Link for Dataverse

Access Dataverse choices (option sets) with serverless SQL pool

Article • 02/15/2022

For columns that use Microsoft Dataverse Choices, choice values are written as an integer label and not a text label to maintain consistency during edits. The integer-to-text label mapping is stored in the *Microsoft.Athena.TrickleFeedService/table-EntityMetadata.json* file. This article covers how to access the integer-to-text label mapping using serverless SQL pool.





> **Note**
>
> Azure Synapse Link for Microsoft Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

This section describes the prerequisites necessary to consume Dataverse data with serverless SQL pool after using the Azure Synapse Link for Dataverse service.

- **Azure Synapse Link for Dataverse:** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse with and Azure Synapse Analytics workspace.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.

Consuming Dataverse choices with serverless SQL pool

1. Navigate to your Azure Synapse Analytics workspace.
2. Select **Develop** from the left side panel, then select + > **SQL script**.
3. Paste the following SQL query and replace <**STORAGE_ACCOUNT**> with the storage account name and <**CONTAINER_NAME**> with the name of the container.

SQL

```
    SELECT [EntityName], [OptionSetName], [Option], [IsUserLocalizedLabel],
[LocalizedLabelLanguageCode], [LocalizedLabel]
    FROM OPENROWSET (
        BULK
'https://<STORAGE_ACCOUNT>.dfs.core.windows.net/<CONTAINER_NAME>/Microsoft.A
thena.TrickleFeedService/*-EntityMetadata.json',
        FORMAT = 'csv',
        FIELDTERMINATOR ='0x0b',
        FIELDQUOTE = '0x0b',
        ROWTERMINATOR = '0x0b'
    ) WITH (doc nvarchar(max)) AS rows
    CROSS APPLY OPENJSON(doc, '$.OptionSetMetadata')
    WITH (
            [EntityName] nvarchar(4000) '$.EntityName',
            [OptionSetName] nvarchar(4000) '$.OptionSetName',
            [Option] int '$.Option',
            [IsUserLocalizedLabel] nvarchar(4000) '$.IsUserLocalizedLabel',
            [LocalizedLabelLanguageCode] int '$.LocalizedLabelLanguageCode',
            [LocalizedLabel] nvarchar(4000) '$.LocalizedLabel'
    )
```

4. **Run** the query. A table containing the Dataverse choices is displayed.
5. Join the Dataverse choices with your Dataverse table and store the view in a new database using a three-part naming convention - *[database-name].[schema-name].[table-name]*.

See also

Azure Synapse Link for Dataverse

Quickstart: Use serverless SQL pool

Export Dataverse data in Delta Lake format

Summarize this article for me

Use Azure Synapse Link for Dataverse to export your Microsoft Dataverse data in Delta Lake format. Delta Lake is the native format for Microsoft Fabric as well as many other tools like Azure Databricks. Exporting data in Delta lake format directly from Dataverse eliminates the need to have a separate Delta Lake conversion processes on your own and accelerates time to insight. This article provides information about this feature and shows you how to perform the following tasks:

- Explains Delta Lake and Parquet and why you should export data in this format.
- Export your Dataverse data to your Azure Synapse Analytics workspace in Delta Lake format with the Azure Synapse Link.
- Monitor your Azure Synapse Link and data conversion.
- View your data from Azure Data Lake Storage Gen2.
- View your data from Synapse Workspace.
- View your data in Microsoft Fabric.

What is Delta Lake?

Delta Lake is an open-source project that enables building a lakehouse architecture on top of data lakes. Delta Lake provides ACID (atomicity, consistency, isolation, and durability) transactions, scalable metadata handling, and unifies streaming and batch data processing on top of existing data lakes. Azure Synapse Analytics is compatible with Linux Foundation Delta Lake. The current version of Delta Lake included with Azure Synapse has language support for Scala, PySpark, and .NET. More information: What is Delta Lake?. You can also learn more from the Introduction to Delta Tables video.

Apache Parquet is the baseline format for Delta Lake, enabling you to leverage the efficient compression and encoding schemes that are native to the format. Parquet file format uses column-wise compression. It's efficient and saves storage space. Queries that fetch specific column values need not read the entire row data thus improving performance. Therefore, serverless SQL pool needs less time and fewer storage requests to read the data.

Why use Delta Lake?

- **Scalability**: Delta Lake is built on top of Open-source Apache license, which is designed to meet industry standards for handling large-scale data processing workloads.
- **Reliability**: Delta Lake provides ACID transactions, ensuring data consistency and reliability even in the face of failures or concurrent access.

- **Performance**: Delta Lake leverages the columnar storage format of Parquet, providing better compression and encoding techniques, which can lead to improved query performance compared to query CSV files.
- **Cost-effective**: The Delta Lake file format is a highly compressed data storage technology that offers significant potential storage savings for businesses. This format is specifically designed to optimize data processing and potentially reduce the total amount of data processed or running time required for on-demand computing.
- **Data protection compliance**: Delta Lake with Azure Synapse Link provides tools and features including soft-delete and hard-delete to comply various data privacy regulations, including General Data Protection Regulation (GDPR).

How Delta Lake works with Azure Synapse Link for Dataverse?

When setting up an Azure Synapse Link for Dataverse, you can enable the **export to Delta Lake** feature and connect with a Synapse workspace and Spark pool. Azure Synapse Link exports the selected Dataverse tables in CSV format at designated time intervals, processing them through a Delta Lake conversion Spark job. Upon the completion of this conversion process, CSV data is cleaned up for storage saving. Additionally, a series of maintenance jobs are scheduled to run on a daily basis, automatically performing compaction and vacuuming processes to merge and clean up data files to further optimize storage and improve query performance.

> ⓘ **Important**
>
> - If you're upgrading from CSV to Delta Lake with existing custom views, we recommend updating the script to replace all **partitioned** tables to **non_partitioned.** Do this by looking for instances of `_partitioned` and replace them with an empty string.
> - For the Dataverse configuration, append-only is enabled by default to export CSV data in `appendonly` mode. The Delta Lake table will have an in-place update structure because the Delta Lake conversion comes with a periodic merge process.
> - You need to provision a Spark pool (compute resources) in your own Azure subscription for Delta conversion. This Spark pool is used to perform periodic Delta conversions based on the time interval chosen by you.
> - There are no costs incurred with the creation of Spark pools. Charges are only incurred once a Spark job is executed on the target Spark pool and the Spark instance is instantiated on demand. These costs are related to the usage of Azure Synapse workspace Spark and are billed monthly. The cost of conducting Spark

computing mainly depends on the time interval for incremental update and the data volumes. More information: Azure Synapse Analytics pricing

- You need to create a Spark pool with the current version (go to the version table in In-place upgrade from previous versions). If you're using a previous Spark version, perform an in-place upgrade for your existing profiles. More information: In-place upgrade from previous versions

> **Note**
>
> The Azure Synapse Link status in Power Apps (make.powerapps.com) reflects the Delta Lake conversion state:
>
> - `Count` shows the number of records in the Delta Lake table.
> - `Last synchronized on` Datetime represents the last successful conversion timestamp.
> - `Sync status` is shown as **active** once the data sync and Delta Lake conversion completes, indicating that the data is ready for consumption.

Prerequisites

- Dataverse: You must have the Dataverse **system administrator** security role. Additionally, tables you want to export via Azure Synapse Link must have the **Track changes** property enabled. More information: Advanced options
- Azure Data Lake Storage Gen2: You must have an Azure Data Lake Storage Gen2 account and **Owner** and **Storage Blob Data Contributor** role access. Your storage account must enable **Hierarchical namespace** and **public network access** for both initial setup and delta sync. **Allow storage account key access** is required only for the initial setup.
- Synapse workspace: You must have a Synapse workspace and **Owner** role in access control(IAM) and the **Synapse Administrator** role access within the Synapse Studio. The Synapse workspace must be in the same region as your Azure Data Lake Storage Gen2 account. The storage account must be added as a linked service within the Synapse Studio. To create a Synapse workspace, go to Creating a Synapse workspace.
- An Apache Spark pool in the connected Azure Synapse workspace with the current Apache Spark version (go to version requirements) using this recommended Spark Pool configuration. For information about how to create a Spark Pool, go to Create new Apache Spark pool.
- The Microsoft Dynamics 365 minimum version requirement to use this feature is 9.2.22082. More information: Opt in to early access updates

Recommended Spark Pool configuration

This configuration can be considered a bootstrap step for average use cases.

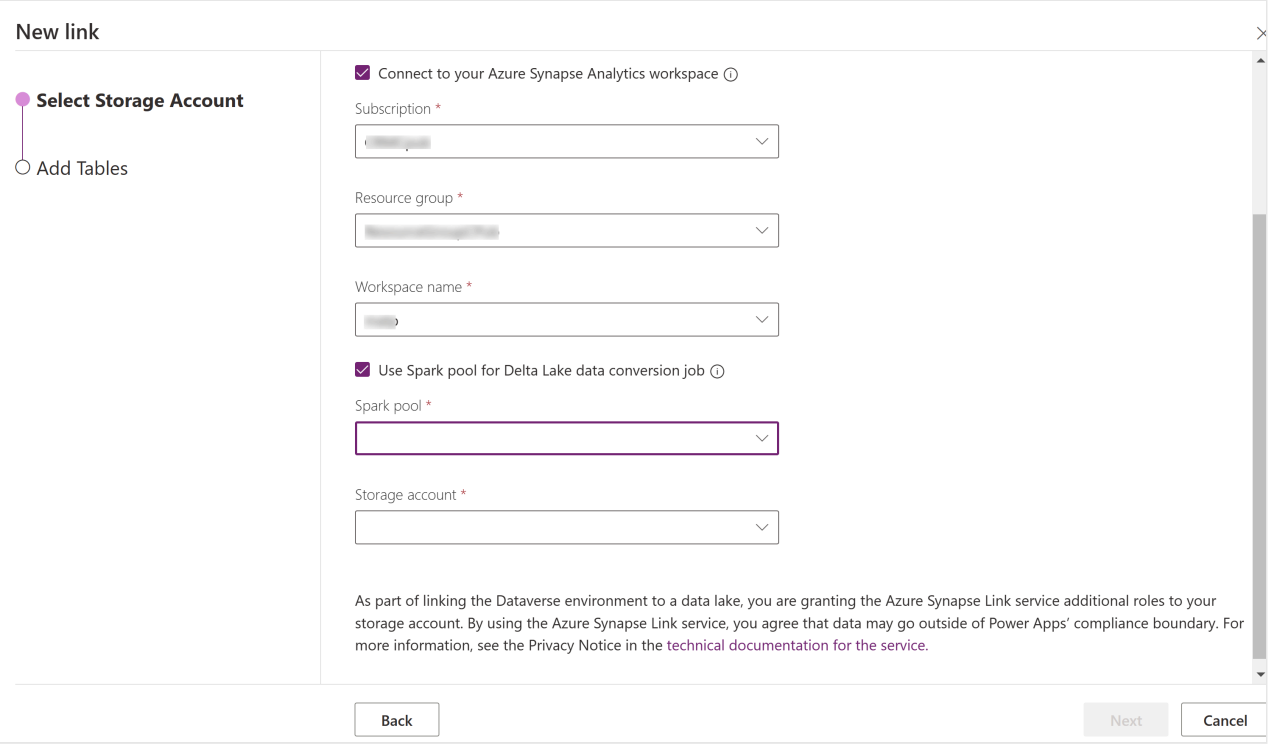
- Node size: small (4 vCores / 32 GB)
- Autoscale: Enabled
- Number of nodes: 3 to 10 (or 20 if needed. [1]More information below.)
- Automatic pausing: Enabled
- Number of minutes idle: 5
- Apache Spark: 3.5 (go to version requirements for latest)
- Dynamically allocate executors: Enabled
- Default number of executors: 1 to 9

> **Important**
>
> - Use the Spark pool exclusively for Delta Lake conversation operation with Synapse Link for Dataverse. For optimal reliability and performance, avoid running other Spark jobs using the same Spark pool.
> - You might need to increase the number of nodes of the Spark pool if you expect a large number of rows to be processed. If the size of the Spark pool is insufficient, Delta conversion jobs might fail
> - The same Spark pool is used by the system to run a nightly job that compacts Delta files in the lake between 11 PM and 6 AM local time. The system determines the night time to run this job based on the location of your Dataverse environment. You can't provide a specific time window. This option reduces the size of Delta files by merging files known as "compaction." In rare cases, this job might interfere with the incremental conversion job. You can increase the number of nodes to 20 in case you notice these failures.
> - You're only charged for the spark pool nodes actually utilized. Increasing the number of nodes might not result in higher charges.

Connect Dataverse to Synapse workspace and export data in Delta Lake format

1. Sign into Power Apps and select the environment you want.
2. On the left navigation pane, select **Azure Synapse Link**. If the item isn't in the side panel pane, select ...More and then select the item you want.

3. On the command bar, select **+ New link**
4. Select **Connect to your Azure Synapse Analytics workspace**, and then select the **Subscription**, **Resource group**, and **Workspace name**.
5. Select **Use Spark pool for processing**, and then select the precreated **Spark pool** and **Storage account**.



6. Select **Next**.
7. Add the tables you want to export, and then select **Advanced**.
8. Optionally, select **Show advanced configuration settings** and enter the time interval, in minutes, for how often the incremental updates should be captured.
9. Select **Save**.

Monitor your Azure Synapse Link and data conversion

1. Select the Azure Synapse Link you want, and then select **Go to Azure Synapse Analytics workspace** on the command bar.
2. Select **Monitor** > **Apache Spark applications**. More information: Use Synapse Studio to monitor your Apache Spark applications

View your data from Synapse workspace

1. Select the Azure Synapse Link you want, and then select **Go to Azure Synapse Analytics workspace** on the command bar.
2. Expand **Lake Databases** on the left pane, select **dataverse-***environmentNameorganizationUniqueName*, and then expand **Tables**. All Parquet tables are listed and available for analysis with the naming convention *DataverseTableName*. **(Non_partitioned Table)**.

> **Note**
>
> Don't use tables with the naming convention *_partitioned*. When you choose Delta parquet as the format, tables with the *_partition* naming convention are used as staging tables and removed after they're used by the system.

View your data from Azure Data Lake Storage Gen2

1. Select the Azure Synapse Link you want, and then select **Go to Azure data lake** on the command bar.
2. Select the **Containers** under **Data Storage**.
3. Select **dataverse- **environmentName-organizationUniqueName*. All parquet files are stored in the **deltalake** folder.

In-place upgrade from previous versions

Current versions and requirements

The following table shows the current and previous versions for Azure Synapse Link with Delta Lake. When you upgrade or create a new configuration, use the current versions listed below.

Expand table

Component	Current version	Previous version	Status	Reference link
Apache Spark	3.5	3.4	3.4 retired	Apache Spark 3.4 runtime
Delta Lake	3.0	2.4	-	Delta Lake compatibility

Upgrade from Apache Spark 3.4 to 3.5

In accordance with the Synapse runtime for Apache Spark lifecycle policy, Azure Synapse runtime for Apache Spark 3.4 is retired. After the end of the support date, the retired runtimes

are unavailable for new Spark pools and existing workflows with Spark 3.4 pools won't be executed and metadata temporarily remains in the Synapse workspace. More information: Azure Synapse Runtime for Apache Spark 3.4.

To ensure that your existing Synapse Link profiles continue to process data, upgrade your Synapse Link profiles using the in-place upgrade process here.

Prerequisites for upgrade

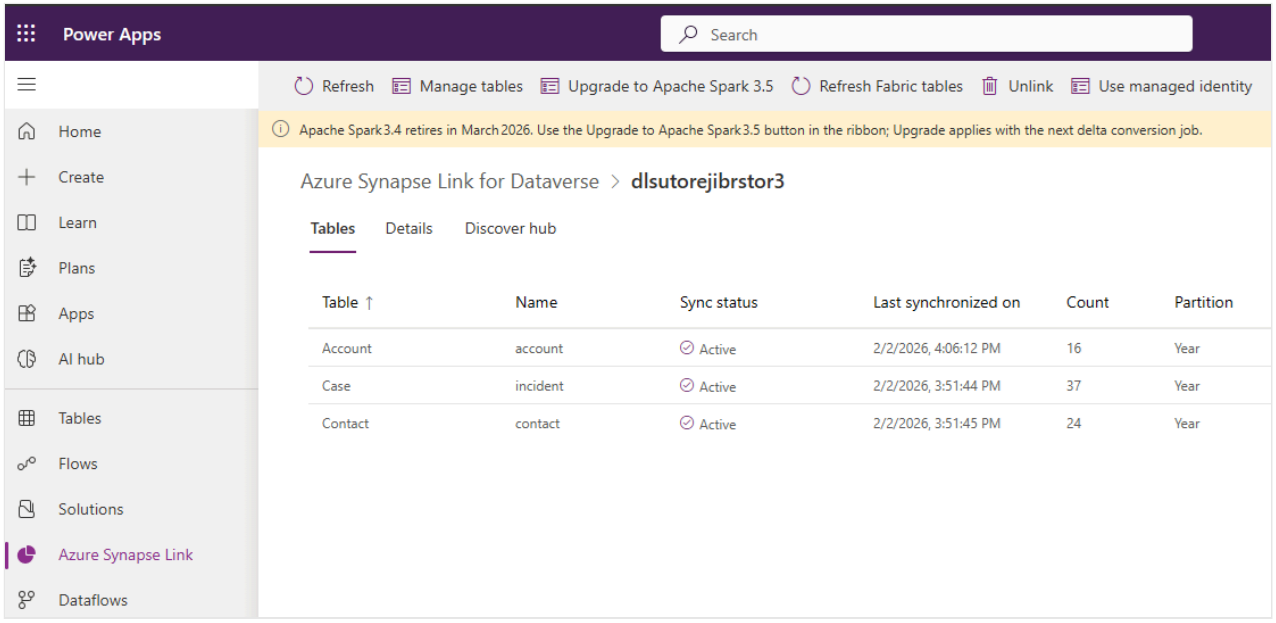
- You must have an existing Azure Synapse Link for Dataverse Delta lake profile running with a previous Apache Spark version.
- You must create a new Synapse Spark pool with the current Spark version (go to versions and requirements table), *using the same or higher nodes hardware configuration within the same Synapse workspace*. For information about how to create a Spark pool, go to Create new Apache Spark pool. This Spark pool should be created independent of your existing pool - *don't delete your current Spark pool or create the new pool with the same name*.

Upgrade steps

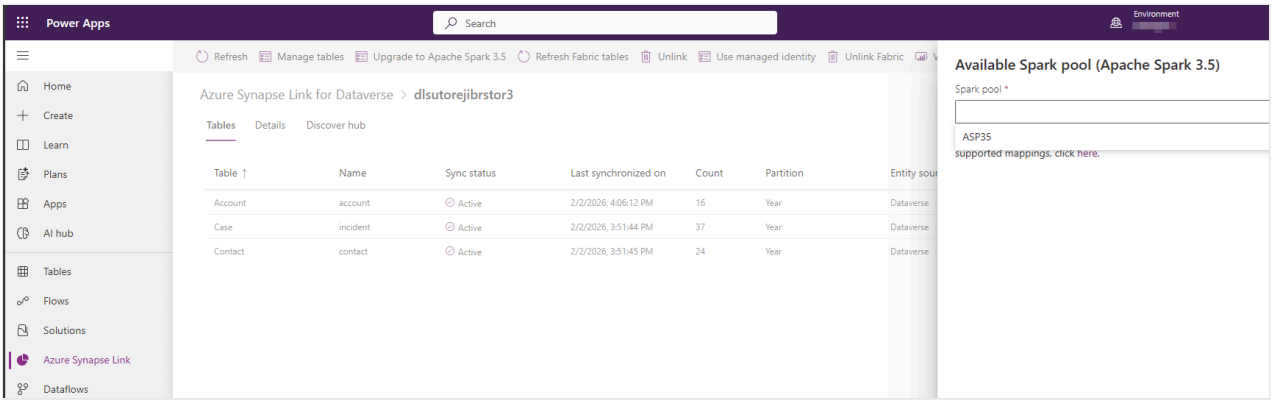
1. Sign in to Power Apps and select your preferred environment.

2. On the left navigation pane, select **Azure Synapse Link**. If the item isn't in the left navigation pane, select **...More** and then select the item you want.

3. If you're using a retired Spark version, an error message indicating that support has ended appears. Select the upgrade button in the ribbon to begin the upgrade process.



4. In the upgrade dialog, select the available Spark pool with the current version from the dropdown list.



5. Select **Update** to complete the upgrade.

> **Note**
>
> - The Spark pool upgrade occurs only when a new Delta Lake conversion Spark job is triggered. Ensure you have at least one data change after selecting **Update**.
> - After selecting **Update**, the upgrade process can take up to 48 hours to complete due to the cache. During this time, the old Spark pool continues to be used for Delta Lake conversion until the upgrade is fully applied in the backend. Don't delete the old Spark pool until you have confirmed that the new Spark pool is being used for Delta Lake conversion jobs. If the new Spark pool isn't used for Delta Lake conversion after two days, contact Microsoft to get support.

Related articles

What is Azure Synapse Link for Dataverse?

Last updated on 03/11/2026

Ingest exported Dataverse data with Azure Data Factory

Article • 04/13/2022

After exporting data from Microsoft Dataverse to Azure Data Lake Storage Gen2 with Azure Synapse Link for Dataverse, you can use Azure Data Factory to create dataflows, transform your data, and run analysis.

> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

This article shows you how to perform the following tasks:

1. Set the Data Lake Storage Gen2 storage account with the Dataverse data as a *source* in a Data Factory dataflow.
2. Transform the Dataverse data in Data Factory with a dataflow.
3. Set the Data Lake Storage Gen2 storage account with the Dataverse data as a *sink* in a Data Factory dataflow.
4. Run your dataflow by creating a pipeline.

Prerequisites

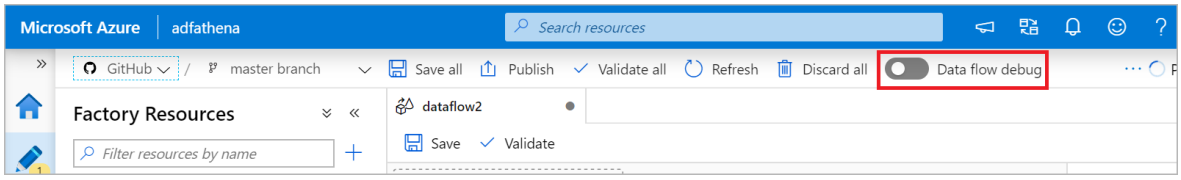
This section describes the prerequisites necessary to ingest exported Dataverse data with Data Factory.

- **Azure roles.** The user account that's used to sign in to Azure must be a member of the *contributor* or *owner* role, or an *administrator* of the Azure subscription. To view the permissions that you have in the subscription, go to the Azure portal, select your username in the upper-right corner, select ..., and then select **My permissions**. If you have access to multiple subscriptions, select the appropriate one. To create and manage child resources for Data Factory in the Azure portal—including datasets, linked services, pipelines, triggers, and integration runtimes—you must belong to the *Data Factory Contributor* role at the resource group level or above.

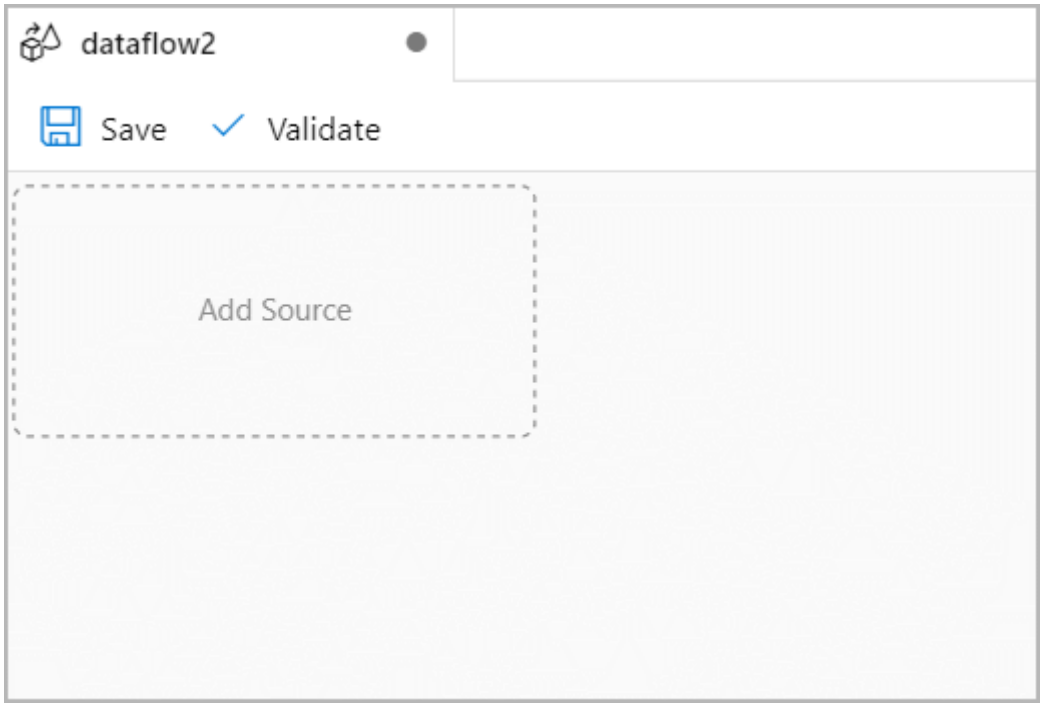
- **Azure Synapse Link for Dataverse.** This guide assumes that you've already exported Dataverse data by using Azure Synapse Link for Dataverse. In this example, the account table data is exported to the data lake.
- **Azure Data Factory.** This guide assumes that you've already created a data factory under the same subscription and resource group as the storage account containing the exported Dataverse data.

Set the Data Lake Storage Gen2 storage account as a source

1. Open Azure Data Factory and select the data factory that is on the same subscription and resource group as the storage account containing your exported Dataverse data. Then select **Create data flow** from the home page.
2. Turn on **Data flow debug** mode and select your preferred time to live. This might take up to 10 minutes, but you can proceed with the following steps.



3. Select **Add Source.**



4. Under **Source settings**, do the following:
    - **Output stream name**: Enter the name you want.

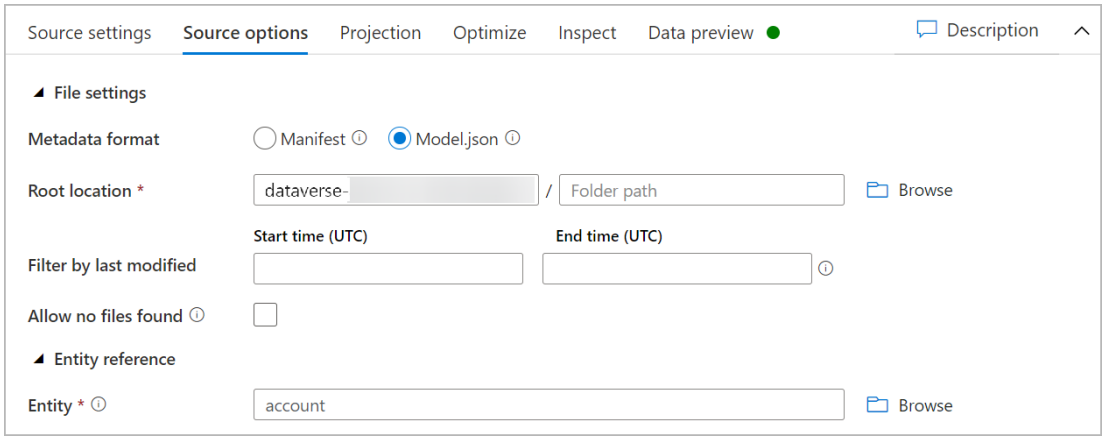
- **Source type**: Select **Inline**.
- **Inline dataset type**: Select **Common Data Model**.
- **Linked Service**: Select the storage account from the drop-down menu, and then link a new service by providing your subscription details and leaving all default configurations.
- **Sampling**: If you want to use all your data, select **Disable**.

5. Under **Source options**, do the following:

   - **Metadata format**: Select **Model.json**.

   - **Root location**: Enter the container name in the first box (**Container**) or **Browse** for the container name and select **OK**.

   - **Entity**: Enter the table name or **Browse** for the table.



6. Check the **Projection** tab to ensure that your schema has been imported successfully. If you do not see any columns, select **Schema options** and check the **Infer drifted column types** option. Configure the formatting options to match your data set then select **Apply**.

7. You can view your data in the **Data preview** tab to ensure the Source creation was complete and accurate.

Transform your Dataverse data

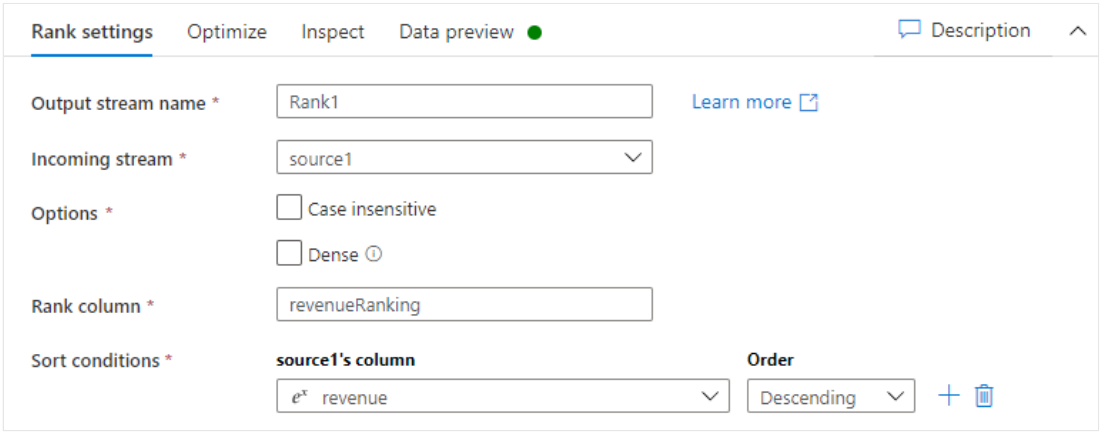
After setting the exported Dataverse data in the Azure Data Lake Storage Gen2 account as a source in the Data Factory dataflow, there are many possibilities for transforming your data. More information: Azure Data Factory

Follow these instructions to create a rank for the each row by the *revenue* field of the account table.

1. Select + in the lower-right corner of the previous transformation, and then search for and select **Rank**.

2. On the **Rank settings** tab, do the following:

    - **Output stream name**: Enter the name you want, such as *Rank1*.
    - **Incoming Stream**: Select the source name you want. In this case, the source name from the previous step.
    - **Options**: Leave the options unchecked.
    - **Rank column**: Enter the name of the rank column generated.
    - **Sort conditions**: Select the *revenue* column and sort by *Descending* order.



3. You can view your data in the **data preview** tab where you will find the new *revenueRank* column at the right-most position.

Set the Data Lake Storage Gen2 storage account as a sink

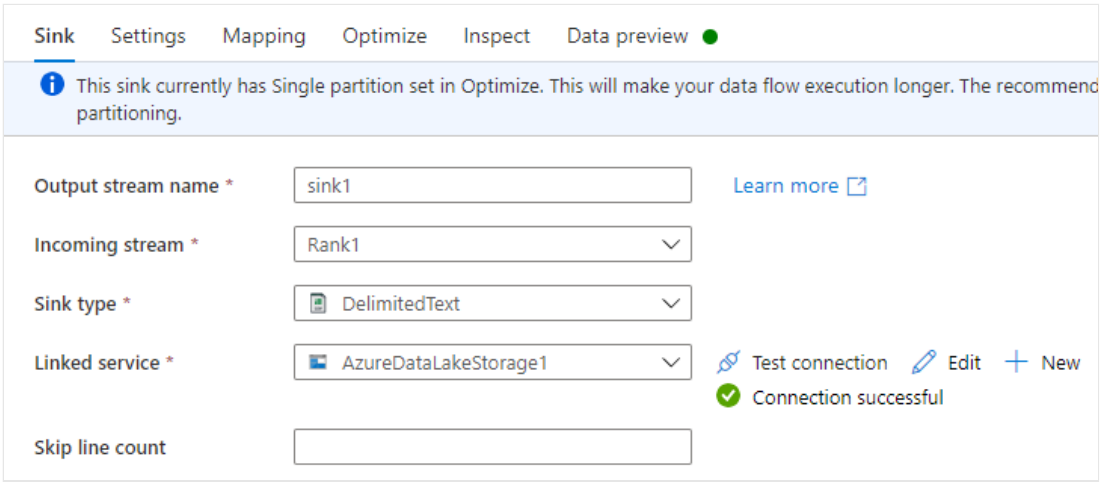
Ultimately, you must set a sink for your dataflow. Follow these instructions to place your transformed data as a delimited text file in the data lake.

1. Select + in the lower-right corner of the previous transformation, and then search for and select **Sink**.

2. On the **Sink** tab, do the following:

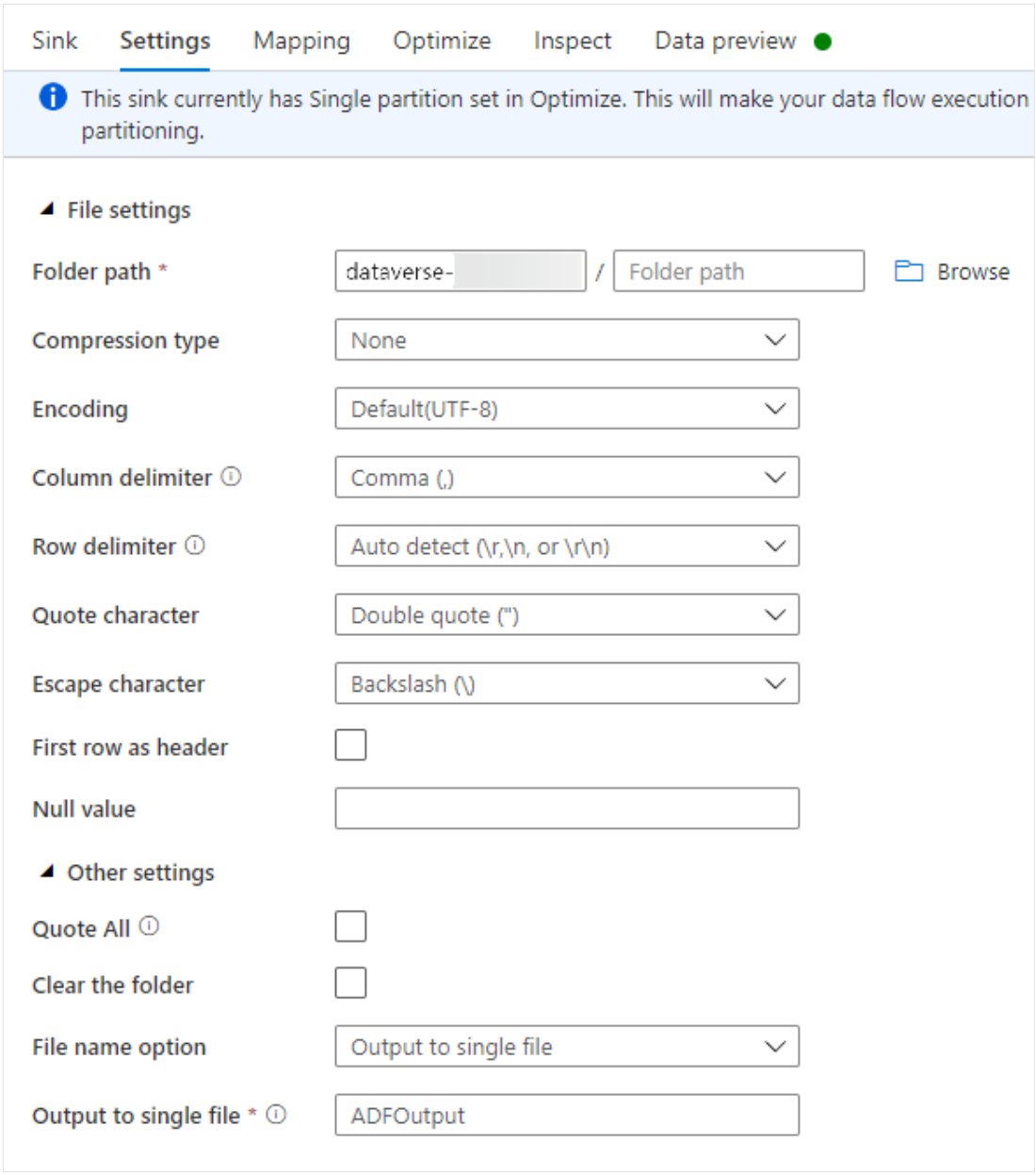
    - **Output stream name**: Enter the name you want, such as *Sink1*.
    - **Incoming stream**: Select the source name you want. In this case, the source name from the previous step.

- **Sink type**: Select **DelimitedText**.
- **Linked service**: Select your Data Lake Storage Gen2 storage container that has the data you exported by using the Azure Synapse Link for Dataverse service.



3. On the **Settings** tab, do the following:

- **Folder path**: Enter the container name in the first box (**File system**) or **Browse** for the container name and select **OK**.
- **File name option**: Select **output to single file**.
- **Output to single file**: Enter a file name, such as *ADFOutput*
- Leave all other default settings.

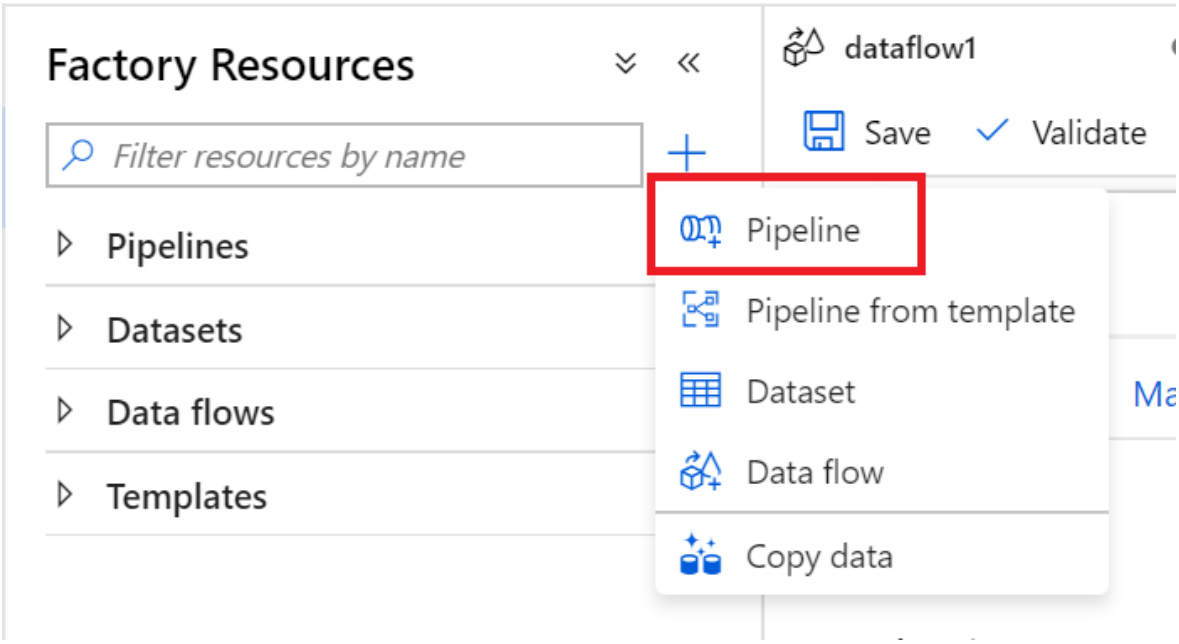


4. On the **Optimize** tab, set the **Partition option** to **Single partition**.

5. You can view your data in the **Data preview** tab.

Run your dataflow

1. In the left pane under **Factory Resources**, select +, and then select **Pipeline**.



2. Under **Activities**, select **Move & Transform**, and then drag **Data flow** to the workspace.

3. Select **Use existing data flow**, and then select the dataflow that you created in the previous steps.

4. Select **Debug** from the command bar.

5. Let the dataflow run until the bottom view shows that is has been completed. This might take a few minutes.

6. Go to the final destination storage container, and find the transformed table data file.

See also

Configure Azure Synapse Link for Dataverse with Azure Data Lake

Analyze Dataverse data in Azure Data Lake Storage Gen2 with Power BI

Copy Dataverse data from multiple data Lakes to a single data lake

Article • 02/15/2022

After exporting data from Microsoft Dataverse to Azure Data Lake Storage Gen2 with Azure Synapse Link for Dataverse, you can use Azure Data Factory to create a pipeline that copies data from multiple data lakes to a single data lake.

> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

This article shows you how to perform the following tasks:

1. Create a pipeline from an official Microsoft template.
2. Configure the template.
3. Run the pipeline.

Prerequisites

This section describes the prerequisites necessary to copy exported Dataverse data from multiple data lakes to a single data lake.

- **Azure roles.** The user account that's used to sign in to Azure must be a member of the *contributor* or *owner* role, or an *administrator* of the Azure subscription. To view the permissions that you have in the subscription, go to the Azure portal, select your username in the upper-right corner, select **...**, and then select **My permissions**. If you have access to multiple subscriptions, select the appropriate one. To create and manage child resources for Data Factory in the Azure portal—including datasets, linked services, pipelines, triggers, and integration runtimes—you must belong to the *Data Factory Contributor* role at the resource group level or above.
- **Azure Synapse Link for Dataverse.** This guide assumes that you've already exported Dataverse data to multiple data lakes by using Azure Synapse Link for Dataverse.

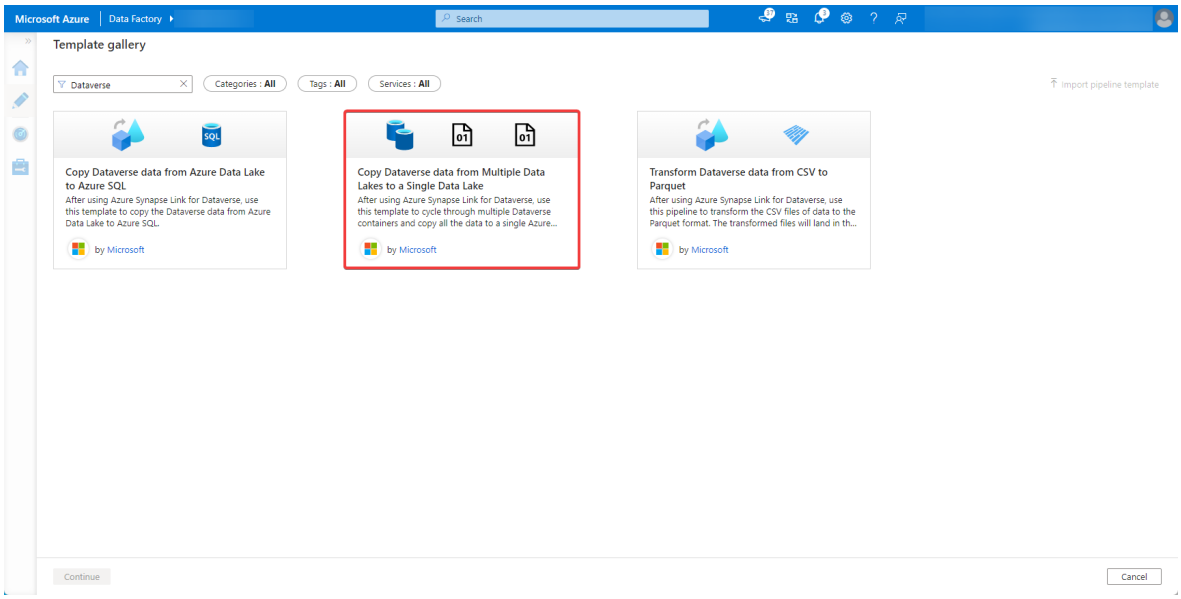
- **Azure Data Factory.** This guide assumes that you've already created a data factory under the same subscription and resource group as the storage account containing the exported Dataverse data.

Copy Dataverse data to a single data lake

1. Open Azure Data Factory and select the data factory that is on the same subscription and resource group as the storage accounts containing your exported Dataverse data. Then select **Author** from the left panel.

2. Select + > **Pipeline** > **Template gallery**.

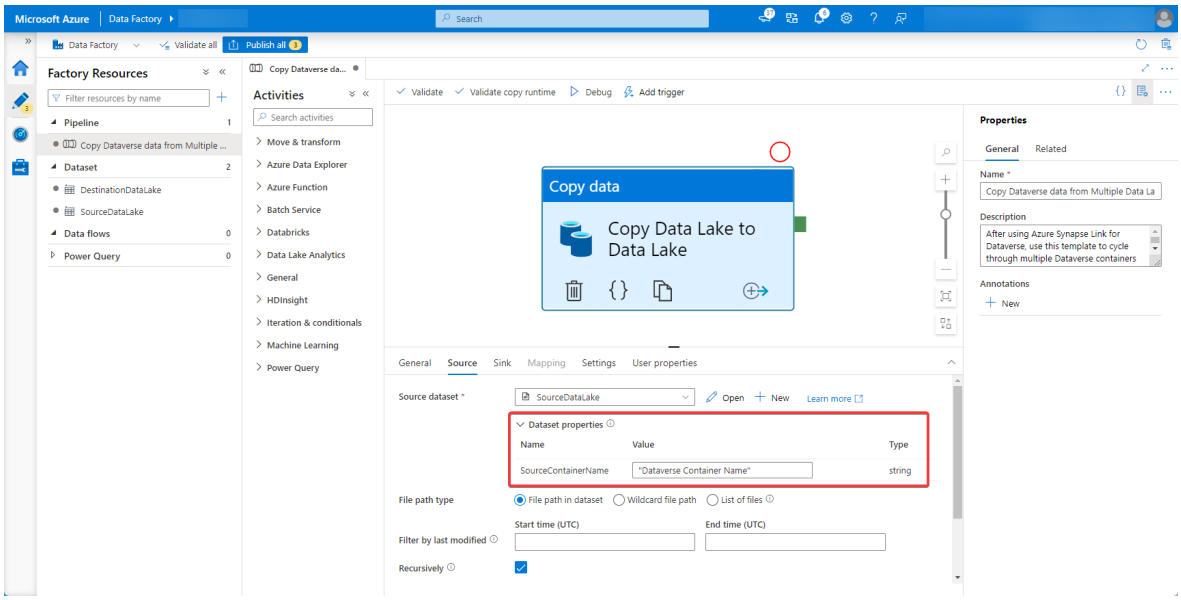
3. Search for and select the **Copy Dataverse data from Multiple Data Lakes to a Single Data Lake** template created by Microsoft.



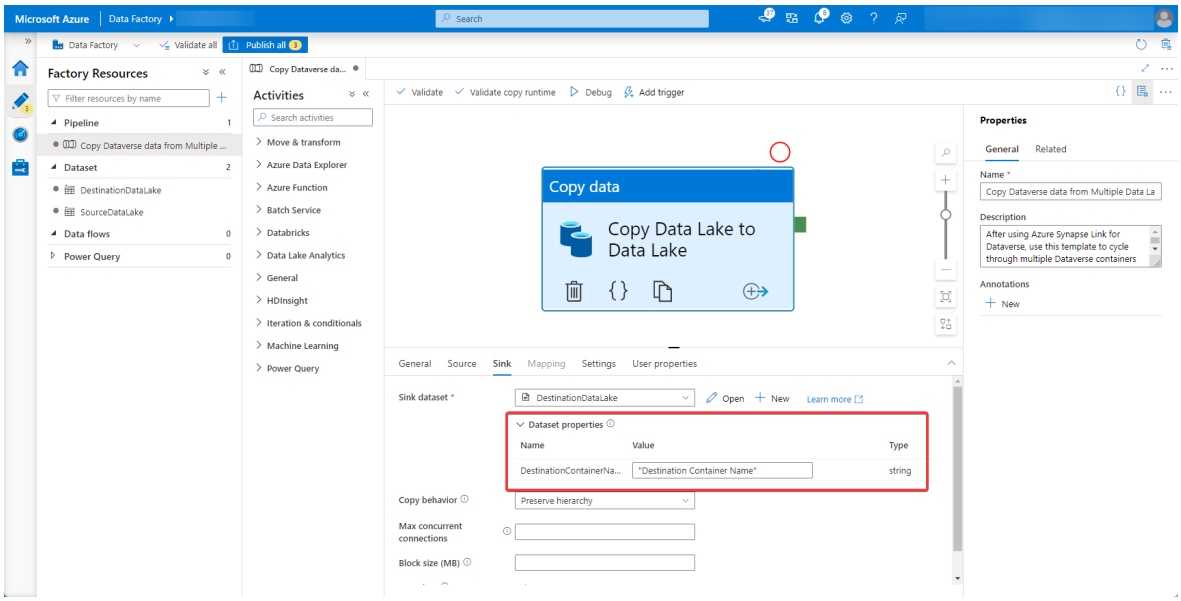
4. Select a **Linked service** for the *SourceDataLake* and the *DestinationDataLake*. Parameterize the *SourceDataLake* **Linked Service** to iterate through multiple data lakes. More information Parameterize linked services in Azure Data Factory and Azure Synapse Analytics.

5. Select **Use this template**.

6. Under the **Source** tab, replace the *SourceContainerName* with the source Dataverse data lake.



7. On the **Sink** tab, replace the *DestinationContainerName* with the destination container name for your copied Dataverse data.



8. To run the pipeline, select **Debug** from the top bar and optionally **Add trigger** to copy data on a schedule.

See also

Azure Synapse Link for Dataverse

Transform Dataverse data from CSV to Parquet with a pipeline template

Article • 02/15/2022

After exporting data from Microsoft Dataverse to Azure Data Lake Storage Gen2 with Azure Synapse Link for Dataverse, you can use Azure Data Factory to create a pipeline that copies data from the data lake to Azure SQL.

> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

This article shows you how to perform the following tasks:

1. Create a pipeline from a Microsoft template.
2. Configure the template.
3. Run the pipeline.

Prerequisites

This section describes the prerequisites necessary to transform Dataverse data to from CSV to Parquet.

- **Azure roles.** The user account that's used to sign in to Azure must be a member of the *contributor* or *owner* role, or an *administrator* of the Azure subscription. To view the permissions that you have in the subscription, go to the Azure portal, select your username in the upper-right corner, select **...**, and then select **My permissions**. If you have access to multiple subscriptions, select the appropriate one. To create and manage child resources for Data Factory in the Azure portal—including datasets, linked services, pipelines, triggers, and integration runtimes—you must belong to the *Data Factory Contributor* role at the resource group level or above.
- **Azure Synapse Link for Dataverse.** This guide assumes that you've already exported Dataverse data by using Azure Synapse Link for Dataverse. In this example, the account table data is exported to the data lake.

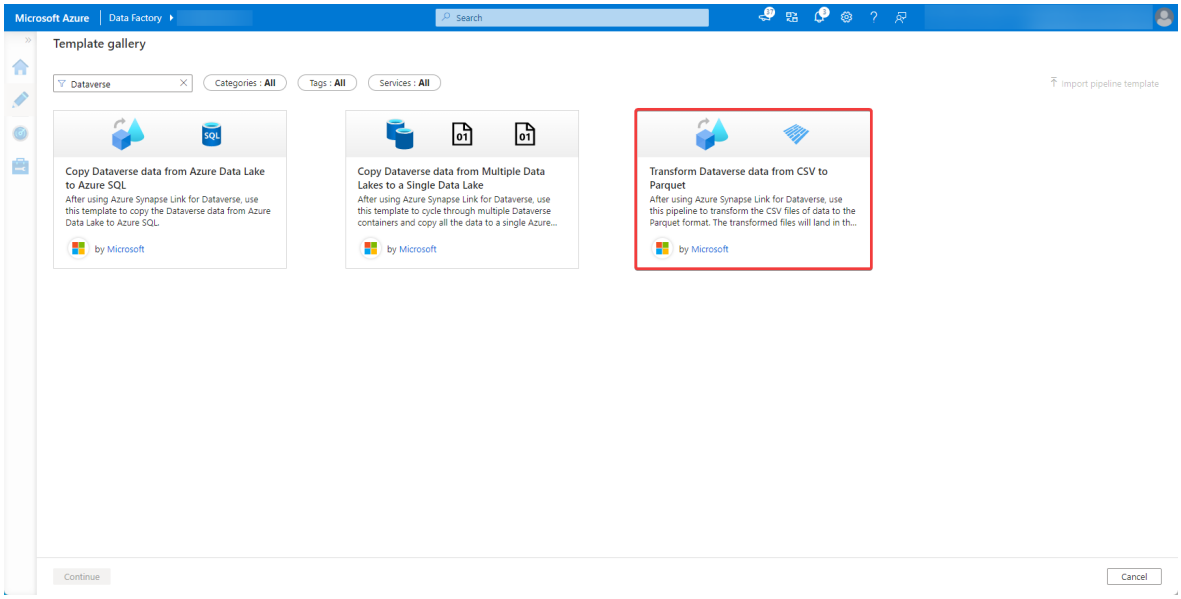
- **Azure Data Factory.** This guide assumes that you've already created a data factory under the same subscription and resource group as the storage account containing the exported Dataverse data.

Transform Dataverse data from CSV to Parquet with a Pipeline Template

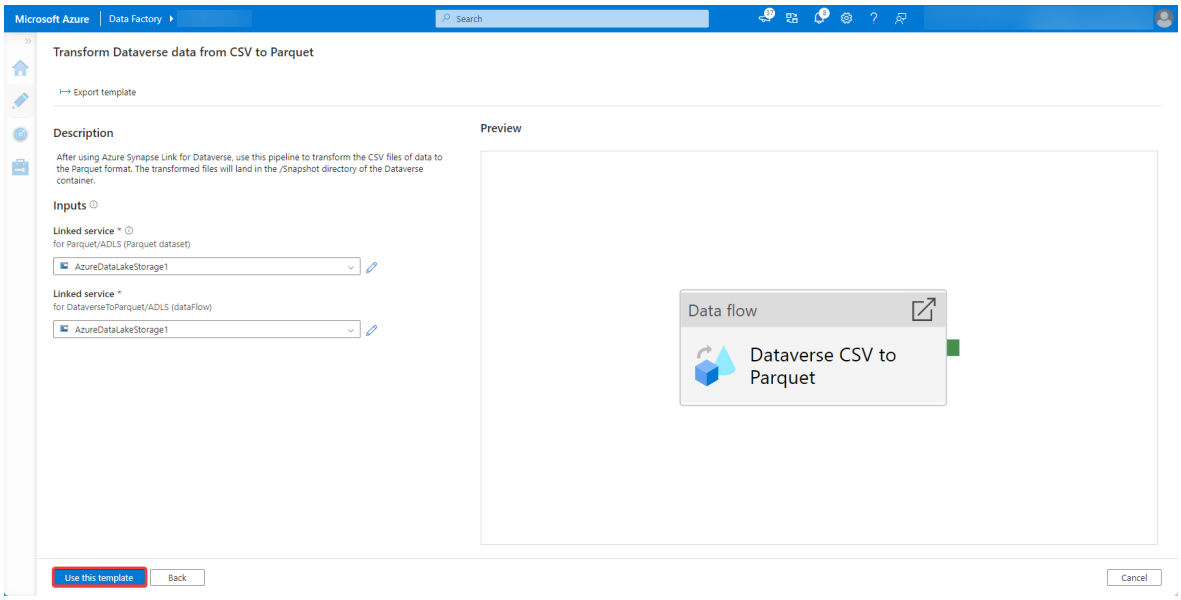
1. Open Azure Data Factory and select the data factory that is on the same subscription and resource group as the storage account containing your exported Dataverse data. Then select **Author** from the left panel.

2. Select + > **Pipeline** > **Template gallery**.

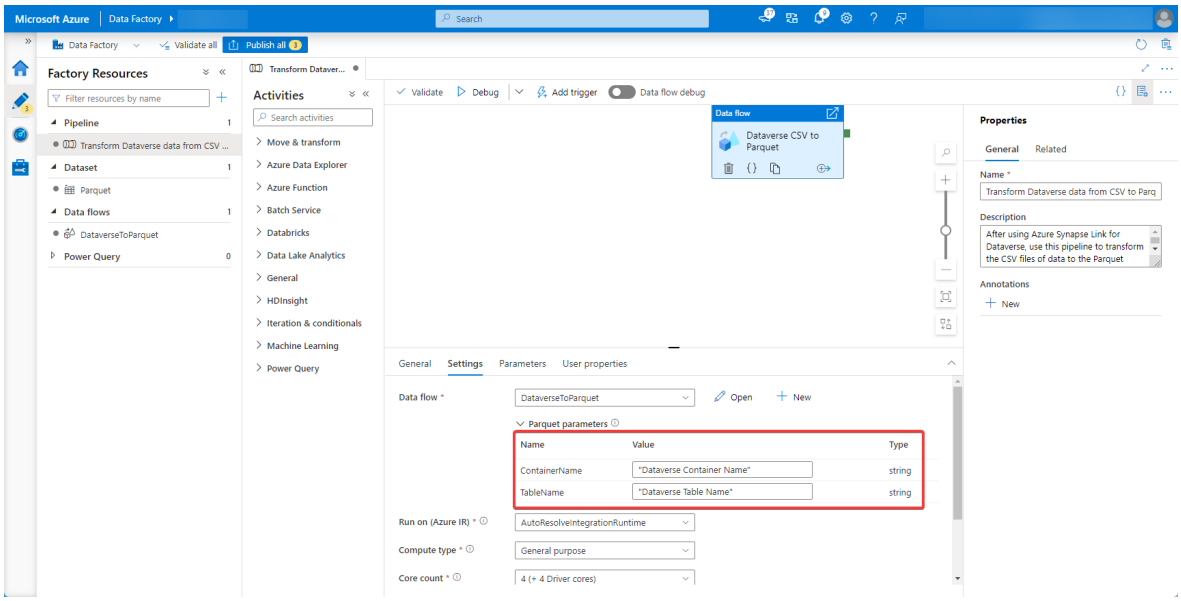
3. Search for and select the **Transform Dataverse data from CSV to Parquet** template created by Microsoft.



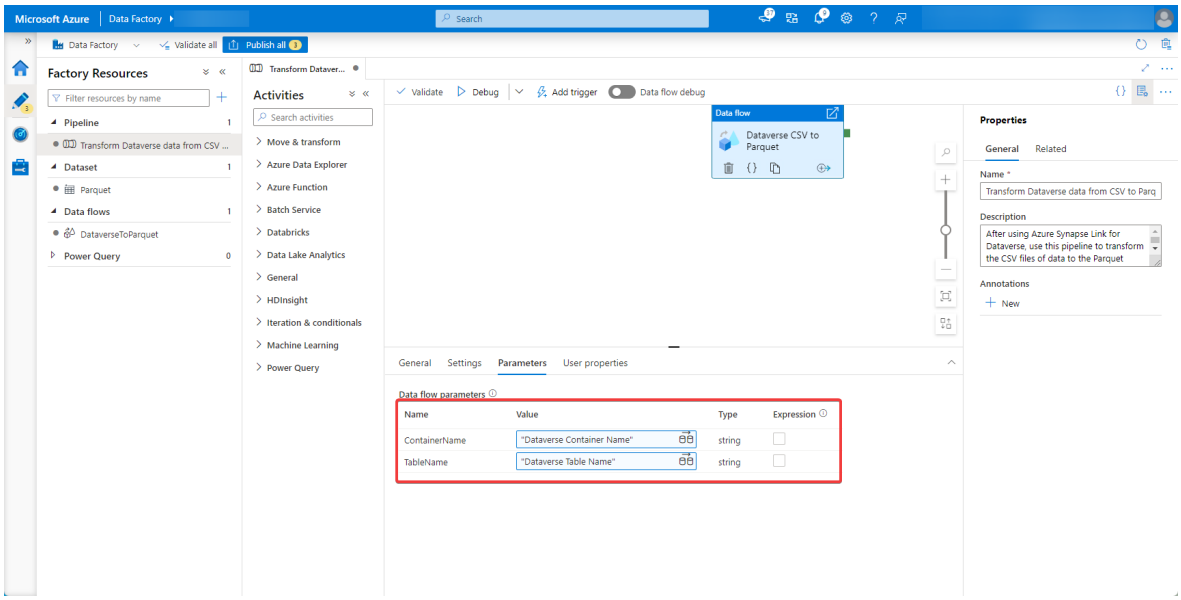
4. Input the Azure Data Lake Storage Gen2 account containing the exported Dataverse data for the first entry and the destination Azure Data Lake Storage Gen2 account where the parquet files will be created for the second entry. Select **Use this template**.



5. Select the data flow **Settings** tab and replace the values for *ContainerName* and *TableName*.



6. Select the data flow **Parameters** tab and replace the values for *ContainerName* and *TableName*.



7. Navigate to the pipeline and run it. Optionally, **Add trigger** to transform the data at specific time intervals.

See also

Azure Synapse Link for Dataverse

Analyze Microsoft Dataverse data in Azure Data Lake Storage Gen2 with Power BI

Article • 02/15/2022

After exporting data from Microsoft Dataverse to Azure Data Lake Storage Gen2 with Azure Synapse Link for Dataverse, you can use Power BI to create business reports and analytics. This can be useful for sales managers and sales associates to refine and build additional reports and dashboards in Power BI.

> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

This article shows you how to perform the following tasks:

1. Connect the Data Lake Storage Gen2 storage container containing the exported Dataverse data to Power BI.
2. Create a report in Power BI that graphs account owners and their respective total account revenue.

Prerequisites

This section describes the prerequisites necessary to consume Dataverse data with Power BI by using the Azure Synapse Link for Dataverse service.

- **Power BI Desktop.** Get it now
- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse. In this example, the account table data is exported to the data lake and will generate a report by using the account table columns.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.

Connect the Data Lake Storage Gen2 storage to Power BI Desktop

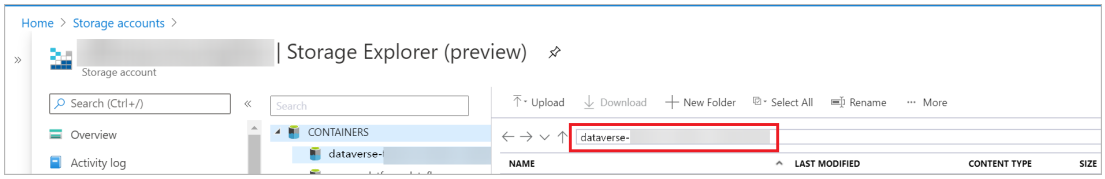
1. Open Power BI Desktop, and sign in.

2. Select **Get data** > **More**.

3. On the **Get Data** page, select **Azure** > **Azure Data Lake Storage Gen2** > **Connect.**

4. Enter the storage container name in the format `https://accountname.dfs.core.windows.net/containername/` and replace **accountname** and **containername** with your storage account's information.

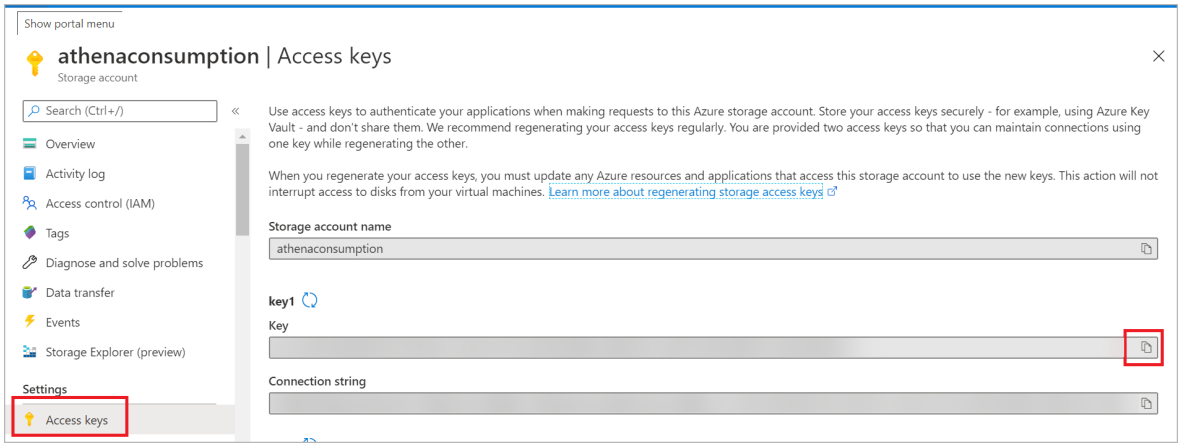
   - Find the storage account name by going to **Azure** > **Storage accounts**, and then select the Data Lake Storage Gen2 account that's populated with the exported Dataverse data.
   - Find the container name by going to **Storage Explorer (preview)** > **Containers** and select the name of the container with the exported Dataverse data.

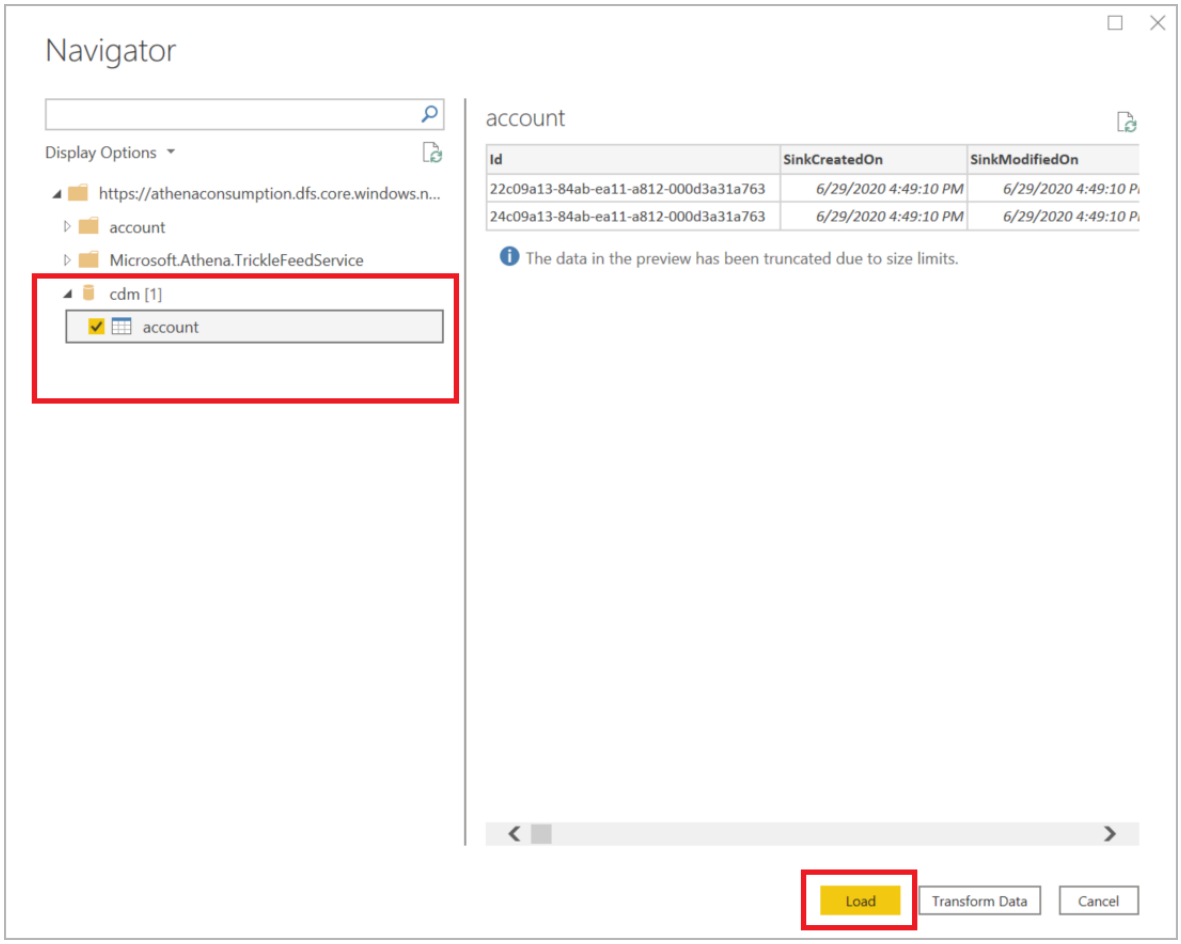
5. Select **CDM Folder View (Beta)**, and then select **OK**.

6. If prompted to sign in, you can sign in with your Microsoft Entra account or with the **Account key**. To sign in with an **Account key**, select the option from the left side bar.

7. Find the account key by going back to the storage account settings and selecting **Access keys** from the left pane. Copy the first key and paste it in Power BI. Select **OK**.

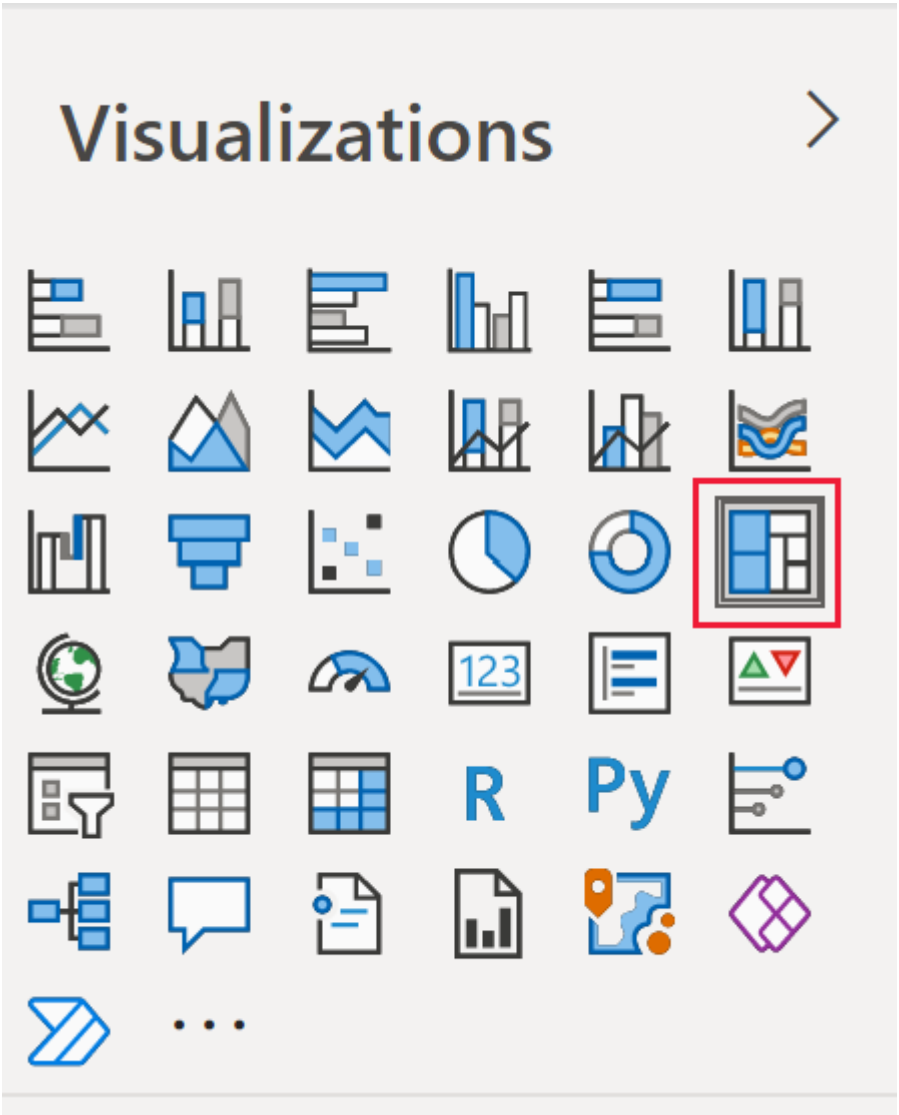
   

8. After successfully connecting to your storage account, expand the **cdm** directory and select the tables that you want to analyze. Then, select **Load**.

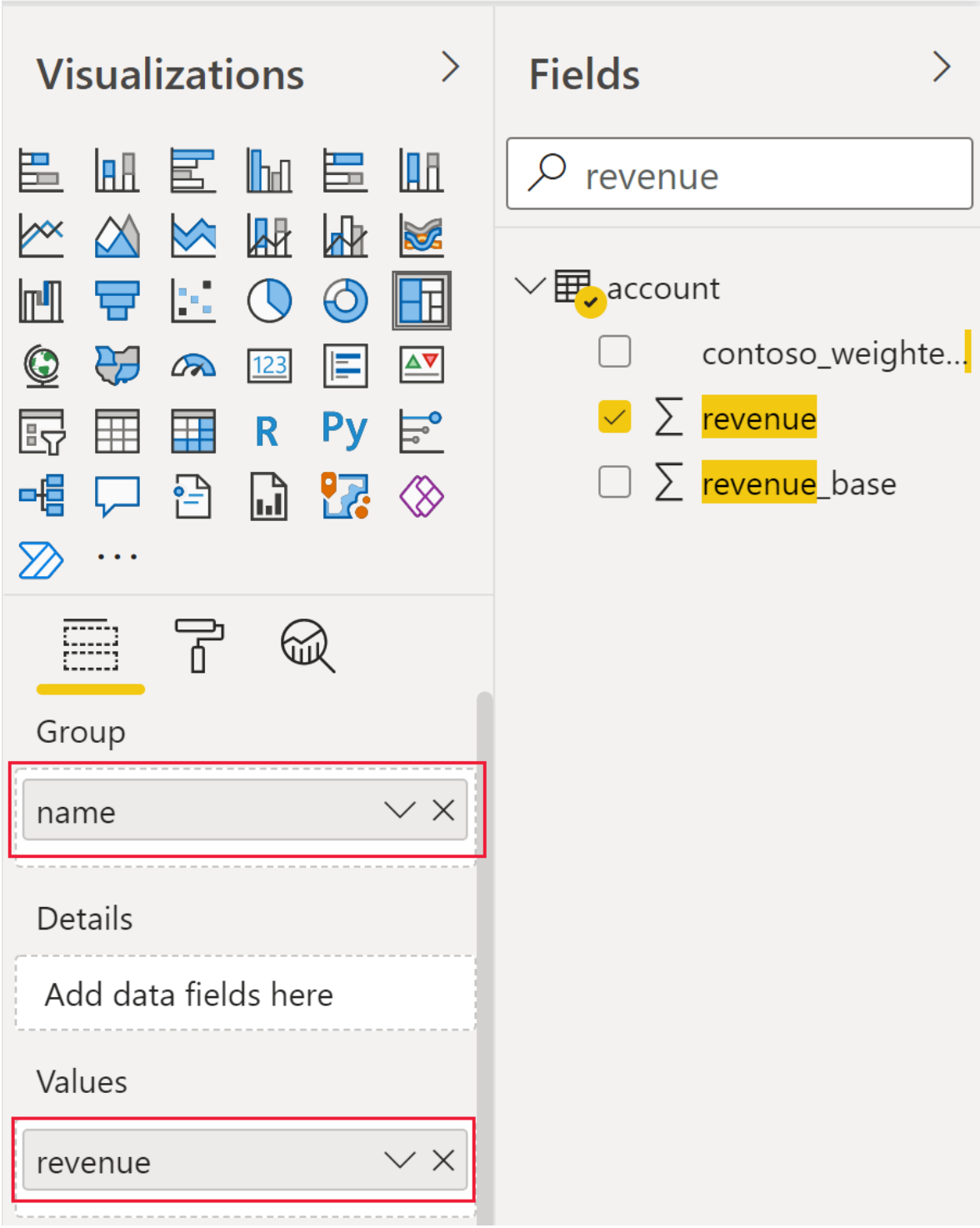


Create a Power BI report with account revenue by account name

1. Under the **Visualizations** menu, select **Treemap**.



2. On the **Columns** menu, search for and select the columns **name** and **revenue**. After you select them, the columns will appear on the **Visualizations** menu for the legend and values, respectively.



A tree map chart appears that separates each account name by color and indicates the total revenue for each account by size.



See also

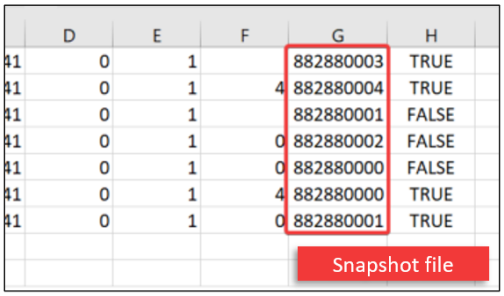
Configure Azure Synapse Link for Dataverse with Azure Data Lake

Ingest Dataverse data in Azure Data Lake Storage Gen2 with Azure Data Factory

Access Dataverse choices (option sets) with Azure Data Factory

Article • 02/15/2022

For columns that use Microsoft Dataverse choices, choice values are written as an integer label and not a text label to maintain consistency during edits. The integer-to-text label mapping is stored in the *Microsoft.Athena.TrickleFeedService/table-EntityMetadata.json* file. This article covers how to access the integer-to-text label mapping using Azure Data Factory.





> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

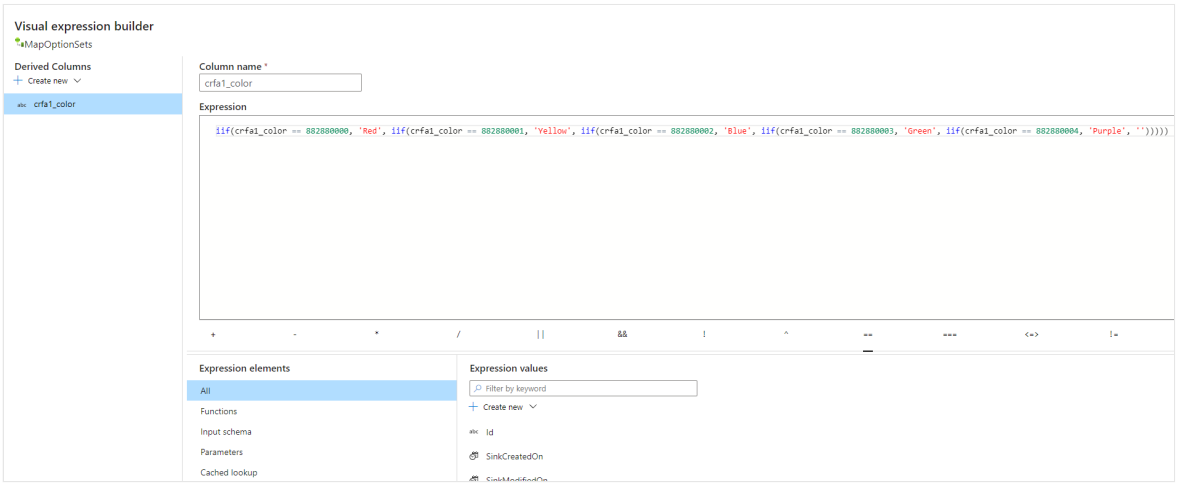
This section describes the prerequisites necessary to access Dataverse choices with Azure Data Factory after using the Azure Synapse Link for Dataverse service.

- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.

Consuming Dataverse choices with Azure Data Factory

To add a column containing the text label of the Dataverse choice using Azure Data Factory, complete the following steps:

1. Go to Azure Data Factory.
2. Create a new data flow and set the source as the Azure Data Lake Storage Gen2 with your Dataverse data.
3. Add a **Derived Column** transformation step and provide a new name for the column.
4. **Open expression builder** and build an expression with a series of `iff` functions to map the integer choice label to the text choice label that is found in the *Microsoft.Athena.TrickleFeedService/table-EntityMetadata.json* file.



5. Add a sink to the data flow and run the pipeline.

See also

Azure Synapse Link for Dataverse

Access choice labels directly from Azure Synapse Link for Dataverse

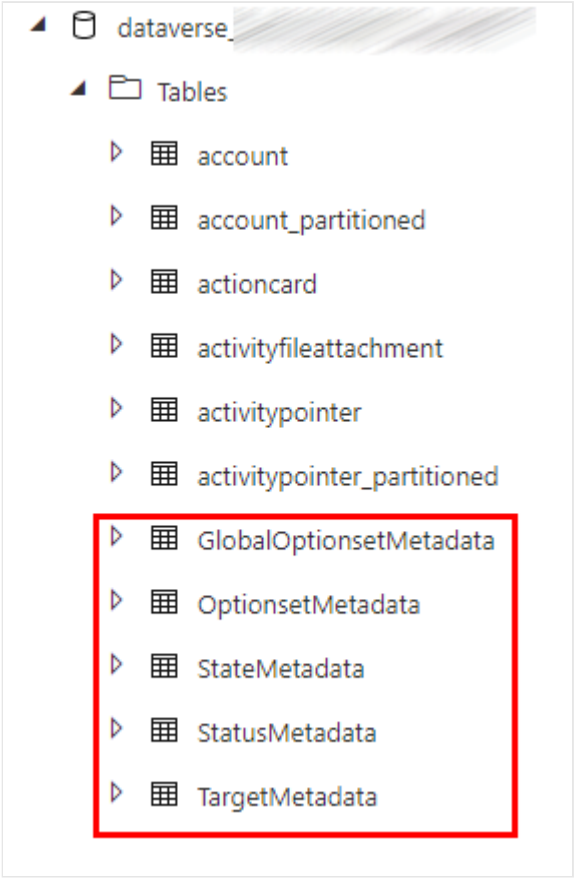
Article • 05/03/2022

Microsoft Dataverse provides rich metadata that can be used directly within Power Apps. A choice (picklist) is one of the most used types of columns that can be included in a table. It defines a set of options. When a choice is displayed in a form, it uses a drop-down list control. You can define a choice to use a set of options defined within itself (locally) or it can use a set of options defined elsewhere (globally), which can be used by other choice columns.

For more information about choice columns, go to Create and edit global choices overview.

After creating an Azure Synapse Link, the following five tables are created in a folder named **OptionsetMetadata** in Azure Data Lake Storage Gen2:

- OptionsetMetadata
- GlobalOptionsetMetadata
- StateMetadata
- StatusMetadata
- TargetMetadata



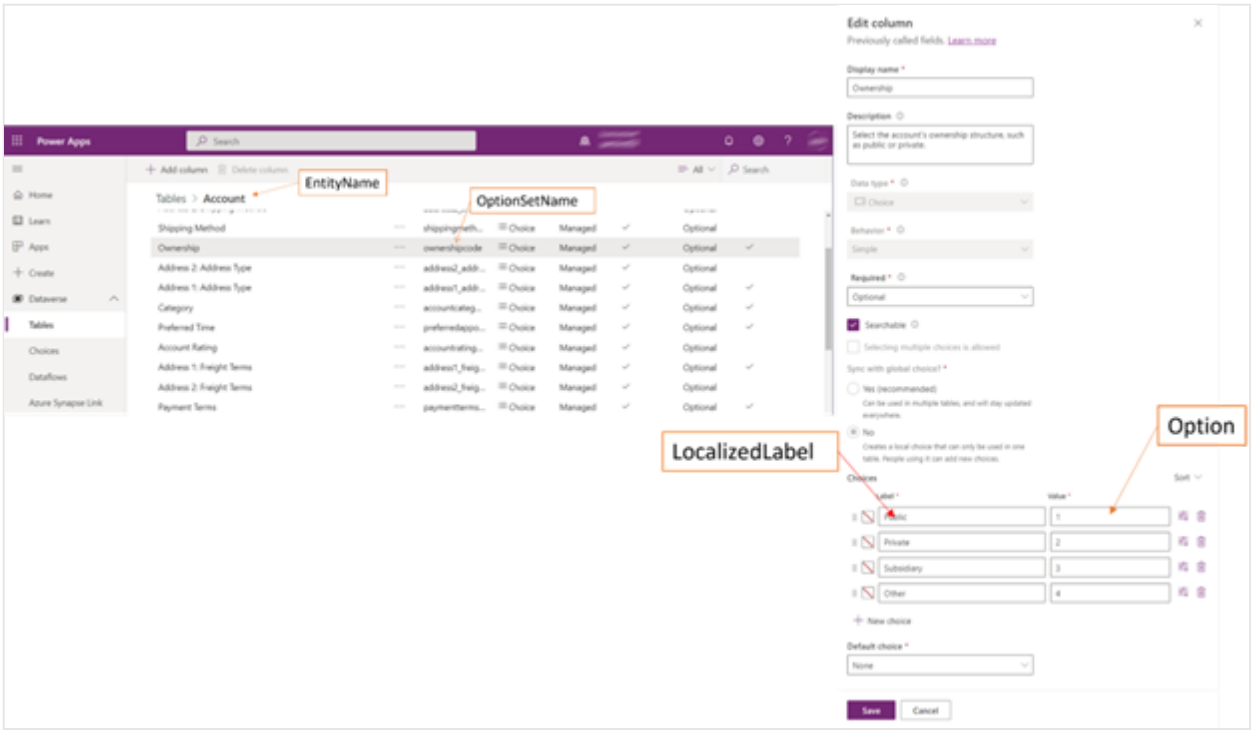
StateMetadata and **StatusMetadata** store the **State** and **Status** choice value to retrieve. **TargetMetadata** stores table relationships metadata to retrieve.

Column name	Data type	Sample value	Description
EntityName	String	account	Current Dataverse table name.
AttributeName	String	transactioncurrencyid	Current column name
ReferencedEntity	String	transactioncurrency	Related Dataverse table name
ReferencedAttribute	String	transactioncurrencyid	Related column name

More information: Table relationships overview.

OptionsetMetadata stores the local choices label metadata in the imported Dataverse tables. **GlobalOptionsetMetadata** stores the global choices label metadata and follows the same table schema plus one extra column, **GlobalOptionSetName**, a combination of table and choice name.

Column name	Data type	Sample value	Description
EntityName	String	account	Dataverse table name.
OptionSetName	String	ownershipcode	Column name.
Option	Bigint	1	User-specified numerical label when the choice item is created.
IsUserLocalizedLabel	Boolean	False	Return False by default.
LocalizedLabelLanguageCode	Bigint	1033	The language code of the choice label, such as 1033 for English (United States) or 1034 for Spanish (Spain).
LocalizedLabel	String	Public	User-specified text label when the choice item is created.
GlobalOptionSetName (GlobalOptionsetMetadata only)	String	socialprofile_community	a combination of table and choice name



In the Dataverse tables, the choice column contains a user-specified numerical value, which is the same as **Option** value in **OptionsetMetadata** table described above.

Prerequisite

Azure Synapse Link for Dataverse. This article assumes that you have already exported data from Dataverse by using Azure Synapse Link for Dataverse.

Access choice metadata

1. Select the desired Azure Synapse Link and select the **Go to Azure Synapse Analytics workspace** on the command bar.
2. Expand **Lake Databases** on the left pane, select **dataverse**-*environmentName*-*organizationUniqueName*, and then expand **Tables**. All the choice metadata listed is available for analysis.

To consume Dataverse choice columns with serverless SQL pool.

1. Right-click the database icon, then select **New SQL script** > **Empty script**.
2. Apply a join SQL script to join the choice metadata with your Dataverse table and store the view in a new database.

Example SQL script to join choice metadata

Replace **<DATABASE_NAME>**,**<COLUMN_NAME>** and **<TABLE_NAME>** with the name of the database, column and table to replace numerical choice value to meaningful text label

SQL

```
SELECT [LocalizedLabel] as [<COLUMN_NAME>]
FROM [<DATABASE_NAME>].[dbo].[<TABLE_NAME>_partitioned]
LEFT JOIN [<DATABASE_NAME>].[dbo].[OptionsetMetadata]
ON ([<DATABASE_NAME>].[dbo].[OptionsetMetadata].[Option] =
[<DATABASE_NAME>].[dbo].[<TABLE_NAME>_partitioned].[<COLUMN_NAME>] AND
[<DATABASE_NAME>].[dbo].[OptionsetMetadata].[OptionSetName] = <COLUMN_NAME>)
```

For more information about how to consume multiple option set values, you would have to use Using Common Table Expressions.

Access Dataverse choices with Power BI

Article • 02/15/2022

For columns that use Microsoft Dataverse choices, choice values are written as an integer label and not a text label to maintain consistency during edits. The integer-to-text label mapping is stored in the *Microsoft.Athena.TrickleFeedService/table-EntityMetadata.json* file. This article covers how to access the integer-to-text label mapping using Power BI.



> **Note**
>
> Azure Synapse Link for Dataverse was formerly known as Export to data lake. The service was renamed effective May 2021 and will continue to export data to Azure Data Lake as well as Azure Synapse Analytics.

Prerequisites

This section describes the prerequisites necessary to access Dataverse choices with Power BI after using the Azure Synapse Link for Dataverse service.

- **Power BI Desktop.** Get it now
- **Azure Synapse Link for Dataverse.** This guide assumes that you have already exported data from Dataverse by using the Azure Synapse Link for Dataverse.
- **Storage Account Access.** You must be granted one of the following roles for the storage account: Storage Blob Data Reader, Storage Blob Data Contributor, or Storage Blob Data Owner.

Consuming Dataverse choices with Power BI

1. Open Power BI Desktop.

2. Select **Get Data** > **Blank query** and then open the **Advanced Editor**.

3. Paste the following query and replace **<STORAGE>** with the storage account name, **<CONTAINER>** with the name of the container, and **<TABLE>** with the name of the Dataverse table.

PowerQueryM

```
  let
    Source =
AzureStorage.DataLake("https://<STORAGE>.dfs.core.windows.net/<CONTAINER>/Mi
crosoft.Athena.TrickleFeedService/<TABLE>-EntityMetadata.json"),
    #"https://<STORAGE> dfs core windows net/<CONTAINER>/Microsoft Athena
TrickleFeedService/_<TABLE>-EntityMetadata json" = Source{[#"Folder
Path"="https://<STORAGE>.dfs.core.windows.net/<CONTAINER>/Microsoft.Athena.T
rickleFeedService/",Name="<TABLE>-EntityMetadata.json"]}[Content],
    #"Imported JSON" = Json.Document(#"https://<STORAGE> dfs core windows
net/<CONTAINER>/Microsoft Athena TrickleFeedService/_<TABLE>-EntityMetadata
json",1252),
    OptionSetMetadata = #"Imported JSON"[OptionSetMetadata],
    #"Converted to Table" = Table.FromList(OptionSetMetadata,
Splitter.SplitByNothing(), null, null, ExtraValues.Error),
    #"Expanded Column1" = Table.ExpandRecordColumn(#"Converted to Table",
"Column1", {"EntityName", "OptionSetName", "Option", "IsUserLocalizedLabel",
"LocalizedLabelLanguageCode", "LocalizedLabel"}, {"Column1.EntityName",
"Column1.OptionSetName", "Column1.Option", "Column1.IsUserLocalizedLabel",
"Column1.LocalizedLabelLanguageCode", "Column1.LocalizedLabel"})
  in
    #"Expanded Column1"
```

This populates a dataset with the choices and various metadata for that choice that you can join with your Dataverse table data to display the text label for the choice.

See also

Azure Synapse Link for Dataverse

Connect to Dataverse with Model Context Protocol

The Model Context Protocol (MCP) is an open protocol that enables seamless integration between large language model (LLM) applications and external data sources and tools. Microsoft Dataverse can act as an MCP server, providing intelligent access to tables and records to various MCP clients like Copilot Studio agents, Visual Studio (VS) Code GitHub Copilot, GitHub Copilot CLI, Claude desktop, Claude Code, and many others. This integration standardizes and streamlines the interaction between AI models and Dataverse data, making it more efficient and effective for developers to apply Dataverse's rich data capabilities within their AI-driven applications.

To use Dataverse as an MCP server, you need to enable and configure the MCP server and allowed clients for your Power Platform environment. Once configured, you can connect to the Dataverse MCP server using different MCP clients. More information: Configure the Dataverse MCP server for an environment

Dataverse MCP server URL

The Dataverse MCP remote server URL follows this format:

`https://{dataverseOrgName}.crm.dynamics.com/api/mcp`

For example, if your Dataverse organization name is `contoso`, the MCP server URL is:

`https://contoso.crm.dynamics.com/api/mcp`

Connect to a Dataverse MCP server

There are multiple ways to connect to a Dataverse MCP server:

- Microsoft Copilot Studio. To learn how to connect to MCP through Dataverse MCP go to Connect to Dataverse with model context protocol in Microsoft Copilot Studio.
- Visual Studio GitHub Copilot and Copilot CLI. To learn how to connect to GitHub Copilot in Visual Studio Code or Copilot CLI go to Connect Dataverse MCP with GitHub Copilot in Visual Studio Code and Copilot CLI.
- Non-Microsoft clients such as Claude desktop and Claude Code. To learn how to connect through Dataverse MCP go to Connect Dataverse MCP with non-Microsoft clients.

List of tools

Once connected to the Dataverse MCP Server, you can choose from various tools in the Power Platform environment.

Expand table

Tool	Description
create_record	Inserts a new row into a Dataverse table and returns the GUID.
describe_table	Retrieves the T-SQL schema of a specified table.
list_tables	Lists all tables in the Dataverse environment.
read_query	Executes SELECT queries to fetch data from Dataverse.
update_record	Updates an existing row in a Dataverse table.
Create Table	Creates a new table with a specified schema.
Update Table	Modifies schema or metadata of an existing table.
Delete Table	Deletes a table from Dataverse.
Delete Record	Deletes a row from a Dataverse table.
Search	Searches through keywords over Dataverse for specific record.
Fetch	Retrieves full content of record in Dataverse using entity name and ID.

> **Note**
>
> Starting December 15, 2025 Dataverse MCP tools are charged when accessed by AI agents created outside of Microsoft Copilot Studio. If you have Dynamics 365 Premium licenses (such as Dynamics 365 Sales Premium, Finance Premium, Supply Chain Premium, and Customer Service Premium) or a Microsoft 365 Copilot User Subscription License (USL), you aren't charged for accessing Dynamics 365 data, even when that data is accessed from outside Microsoft Copilot Studio.

The Search tool is billed at the same Copilot Credit rate as Tenant graph grounding, while all other tools follow the Text and generative AI tools (basic) per 10 response Copilot credit rate. For information about Copilot billing, go to Billing rates and management.

Next steps

Configure the Dataverse MCP server for an environment

Last updated on 03/30/2026

Configure the Dataverse MCP server for an environment

This article provides detailed instructions about how to enable, manage, configure, and disable the Dataverse Model Context Protocol (MCP) server for environments within the Power Platform admin center. It's intended for Power Platform administrators handling managed environments and also covers prerequisites for enabling the server.

Prerequisites

- Power Platform administrator role in order to access Dataverse MCP server environment settings, enable allowed MCP clients, create or edit an environment group, and change connector policies.
- The steps described in this article require that the environment is a Managed Environment.
- By default, the Microsoft Copilot Studio client for Dataverse MCP is enabled for all environments. You must enable the additional clients in the Power Platform admin center before you can connect to the client.

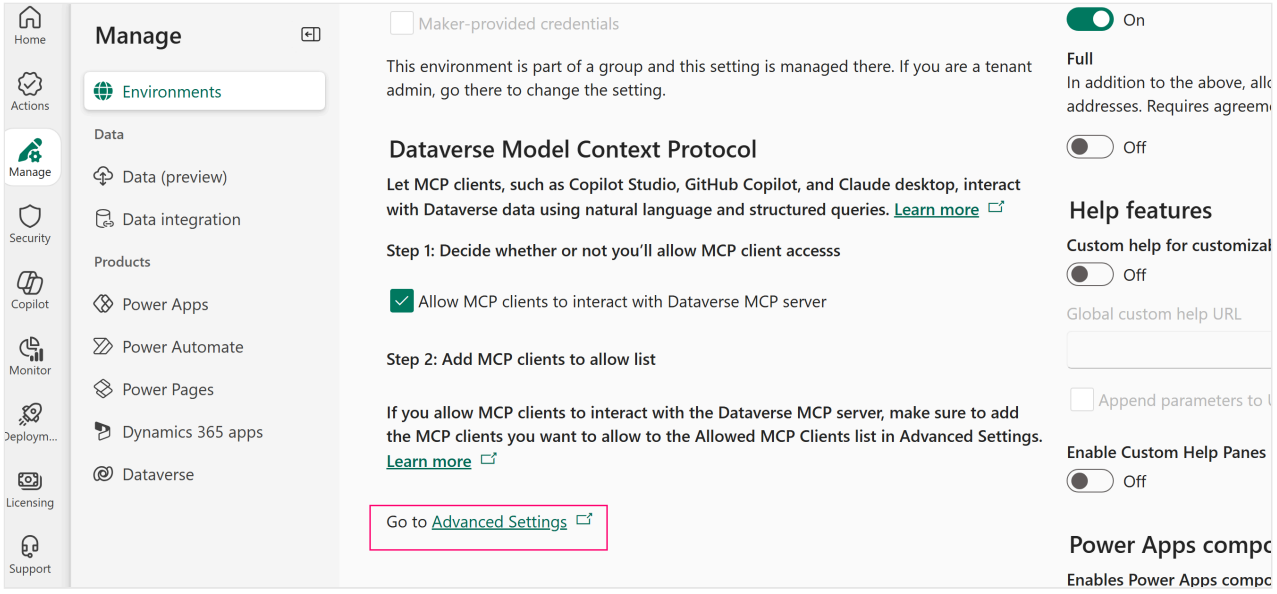
Configure and manage the Dataverse MCP server

By default, Dataverse MCP server is enabled for Copilot Studio. To enable non-Microsoft MCP clients, such as Visual Studio GitHub Copilot and Claude, follow these steps:

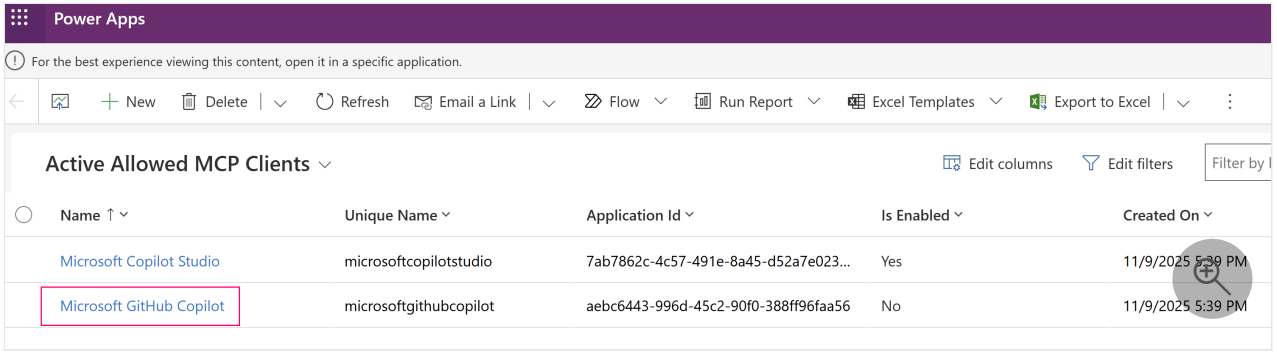
1. Go to Power Platform admin center. Select **Manage** > **Environments**.

2. Select the **Environment Name** where you want to turn on the Dataverse MCP server, and then select **Settings**. Under **Settings**, select **Product** > **Features**. Scroll down to locate **Dataverse Model Context Protocol** and make sure **Allow MCP clients to interact with Dataverse MCP server** is turned on.

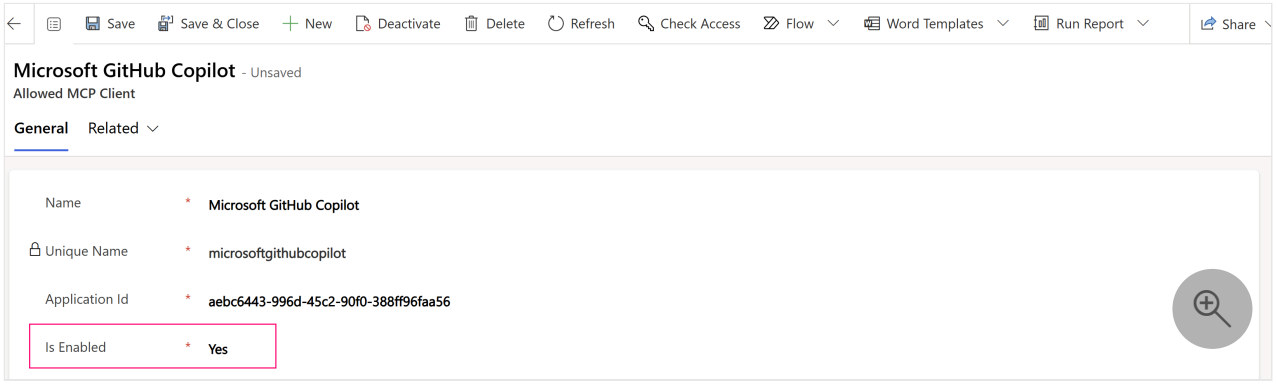
3. Select **Advanced Settings**.



4. The list of available clients is shown. Open the client record you want. In this example, the **Microsoft GitHub Copilot** client is enabled.



5. On the MCP client record, set **Is Enabled** to **Yes**.



6. Select **Save & Close**.

7. Repeat steps 4-7 to enable other clients as needed.

Disable the Dataverse MCP server for an environment

By default the **Allow MCP clients to interact with Dataverse MCP server** is turned on for Copilot Studio. Admins can disable MCP for Dataverse by clearing the setting.

> ⚠ **Warning**
>
> Disabling the Dataverse MCP Server stops all tools and agents that rely on it. Any ongoing development or AI integration testing using MCP is also interrupted.

Write effective instructions for a Dataverse MCP server agent

When you configure your agent in Copilot Studio or Visual Studio Code to use a Dataverse MCP server, clear and well-structured instructions are key to guiding how the agent operates. These instructions help the agent understand its role, what capabilities it has via the MCP server tools, and how to carry out workflows reliably and consistently.

Agent instructions are natural-language directives that tell your agent what it should do, how it should behave, and how to use the MCP tools available to it. They give important context so the agent can:

- Select and call the right MCP tools.
- Fill in tool inputs correctly.
- Decide when to use tools versus generating answers directly.
- Follow the desired tone or behavior patterns during conversations.

Principles for effective instructions

When writing instructions for a Dataverse MCP server agent, consider the following guidelines:

1. Define the agent's purpose clearly.

   State the agent's role in simple terms. For example: "Assist with customer data queries, updates, and record management using Dataverse MCP tools."

   Include tone or restrictions if applicable such as, "always respond in a professional tone" or "do not expose sensitive fields in responses."

2. Outline skills and tool actions.

   Tell the agent which high-level tasks it can perform with the MCP tools. For example:

- Retrieve or filter records using the read_query tool.
- Create or update records with `create_record` and `update_record`.
- Explore table schema with `describe_table`.

Phrase these as actionable instructions, such as "Use the read_query tool to fetch records when the user asks for data retrieval."

3. Add workflow guidance.

   For common processes (like looking up a record or updating a field), add step-by-step guidance. For example: "When updating a record, first identify it using a read_query call, then use update_record with the appropriate field values."

   Providing this context helps the agent orchestrate the right sequence of MCP tool calls.

Add instructions in Visual Studio Code or Copilot Studio

You can use a markdown structured block of instructions as a starting point for your own agent's instructions.

Add instructions in Visual Studio Code

1. Open Visual Studio Code.
2. Open the **Chat** pane.
3. Select **Settings** > **Chat Instructions**.
4. Select **New instruction file**.
5. Copy and paste the instruction content into the new file, such as the sample instructions.

Add instructions in Copilot Studio

1. Open Copilot Studio.
2. Select your agent.
3. Go to **System instructions** and paste your agent instructions, such as the sample instructions.

Sample agent instructions

Markdown

```
## Role
Act as an autonomous agent responsible for interacting with the Microsoft Dataverse
app.
```

```
## Objective
Your objective is to respond to tasks provided by the user. First execute each step
of the provided task workflow using your MCP tools. Check if you have achieved your
objective after each tool call. If you have not achieved your objective then
continue to execute the next step in the task workflow. Do not ask for user
intervention,if you have to run an operation externally in case dataverse doesnt
support that operation. you are allowed to run it without user confirmation. Also
do not ask confirmation for delete table or delete record operation, you can
delete.

## MCP tool selection instructions
 - Whenever you have to use logical table name, call the list_tables tools to get
that logical table name.
 - Whenever you have to use column/attribute name, call the describe_table tool to
get the column/attribute name.

## MCP tool usage instructions

- Before executing an MCP tool, always review the tool description and
restrictions.
- Always strictly follow the description of each MCP tool and perform actions
without any deviation from the tool description.
- Provide higher precedence to tool description over general knowledge.
- Always review the tool documentation and restrictions before running any query or
operation. Strictly validate each planned action against the tool's rules and
supported features before execution.
- For read_query tool, there are restrictions on SQL conditions. Always refer to
the tool description for supported and unsupported sql keywords before generating
the sql query and ensure only supported conditions/keywords are used.

## Reasoning instructions

- Think out loud and reason step by step.
- Before each tool call, plan and verify the action conforms to the tool
description.
- After each tool call, reflect on the result and determine the next step.
- If an exception, error, or warning is observed, communicate it clearly to the
user and retry based on the error message.
- When answering questions about data, DO NOT rely on general knowledge - always
use tools to retrieve accurate, current data.
- DO NOT stop reasoning until all tasks are complete or an unrecoverable error
occurs.
- Only ask clarifying questions if the task requirements are ambiguous.
```

Related articles

Advanced connector policies - Power Platform

Connect to Dataverse with model context protocol

Connect to Dataverse with model context protocol FAQ

Last updated on 04/01/2026

Connect to Dataverse with model context protocol in Microsoft Copilot Studio

This article explains how to set up and use the Microsoft Dataverse model context protocol (MCP) server with Microsoft Copilot Studio. By following the steps in this article, you can interact with Dataverse, asking natural language questions like *show me my contacts* and receive answers based on stored data.

Prerequisites

- The Microsoft Copilot Studio MCP client must be allowed in the environment. More information: Configure and manage the Dataverse MCP server for an environment

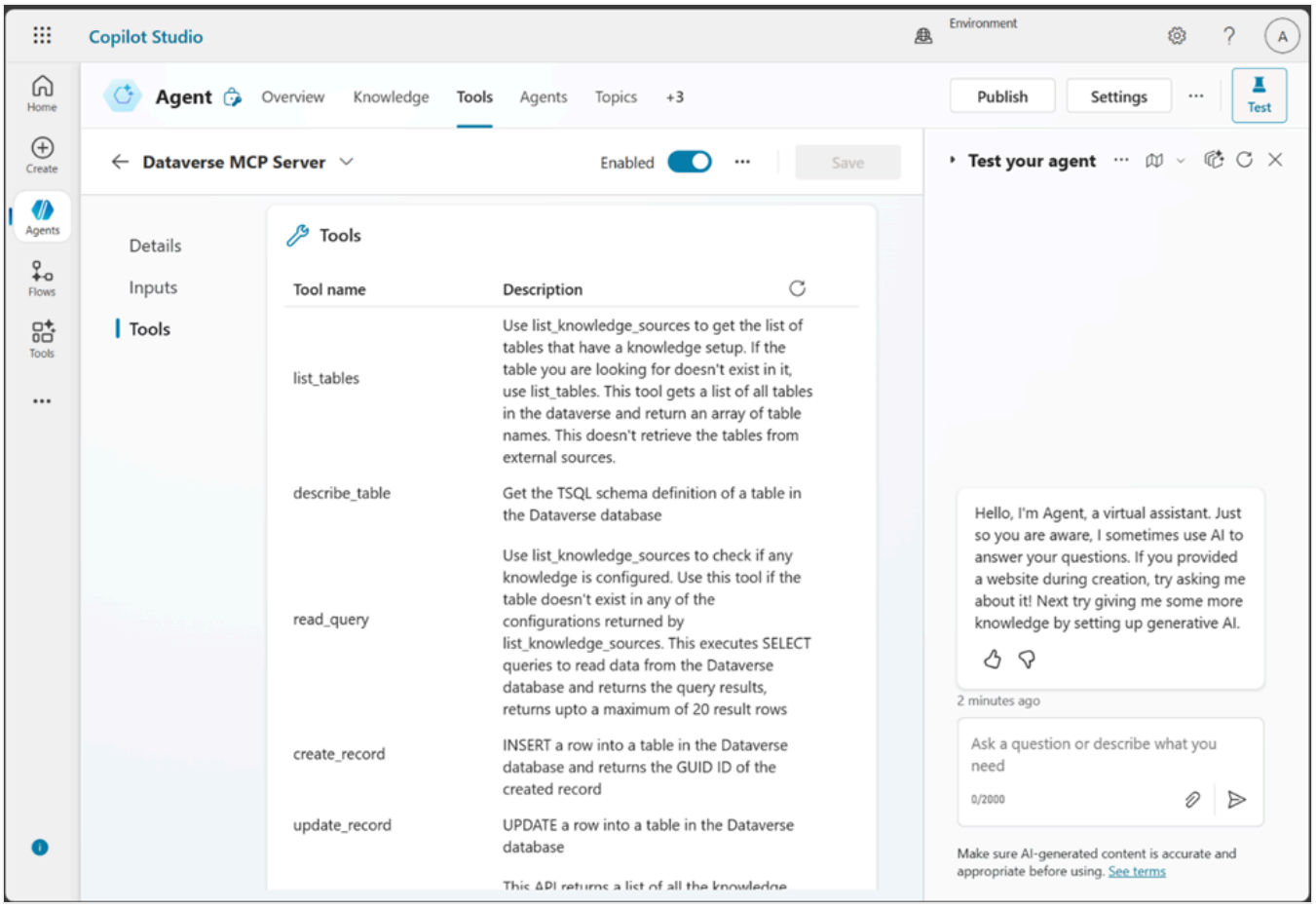
Connect to Dataverse using an MCP server in Microsoft Copilot Studio

1. Go to Power App and select your environment from the top right environment selector.
2. From the left navigation pane, select **Agents** > **Create new agent**. If the item isn't in the side panel pane, select ...More and then select the item you want.
3. Select **Create**.
4. Scroll down to the **Tools** section and select **+ Add tool**.
5. Select **Model Context Protocol**, and then select **Dataverse MCP Server**.
    - If there's no existing Dataverse connection, you're prompted to connect one.
6. Select **Add to agent**.

The individual tools available on this MCP server can be viewed and modified by selecting **...** > **Edit** next to the **Dataverse MCP Server** tool.

You can now interact with the Dataverse MCP Server tool in the **Test your agent** chat pane. Try commands like "list tables in Dataverse," "describe table account," or "how many accounts do I

have."



Related articles

Connect to Dataverse with Model Context Protocol

Last updated on 11/14/2025

Connect Dataverse MCP with GitHub Copilot in Visual Studio Code and Copilot CLI

This article explains how to set up and use the Microsoft Dataverse model context protocol (MCP) server with GitHub Copilot in Visual Studio Code and GitHub Copilot CLI.

GitHub Copilot in Visual Studio Code

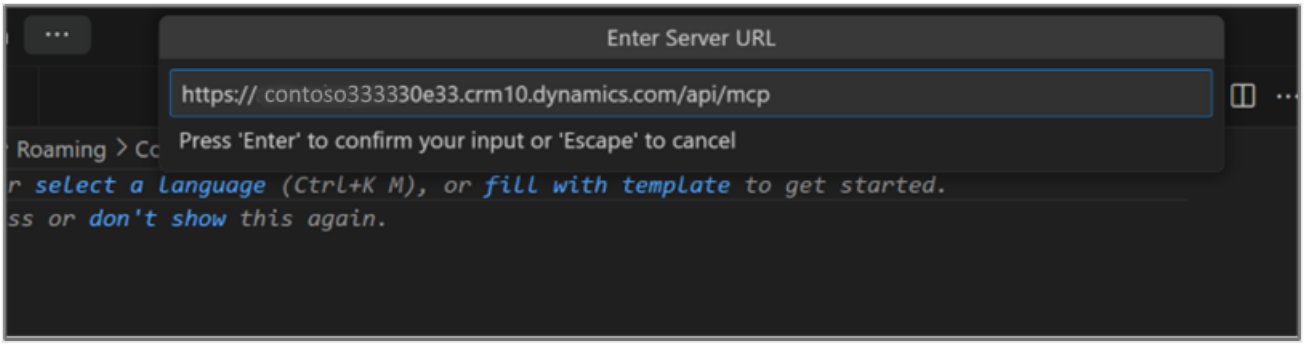
Prerequisites

- The **Microsoft GitHub Copilot** MCP client must be allowed in the environment. More information: Configure and manage the Dataverse MCP server for an environment
- Visual Studio Code installed with GitHub Copilot extension. More information: GitHub Copilot extension for Visual Studio Code

Steps to connect to Dataverse MCP server in Visual Studio Code

1. Open Visual Studio Code. Select **View** > **Command Palette** (Ctrl+Shift+P), type **MCP: Add Server**, and then select **HTTP or Server Sent Events**.

2. Paste your instance URL, such as `https://contoso.crm.dynamics.com/`, append */api/mcp* to it, and press Enter. You can get the instance URL at make.powerapps.com > **Settings** (gear icon) > **Session details** > **Instance url**.



   This step generates the MCP server configuration in Visual Studio Code.

3. Type a MCP server name or press Enter to accept the default name.

4. Choose **Global** or **workspace**.

5. Press **Ctrl+Alt+I** and ensure that agent mode is selected.



GitHub Copilot CLI

Prerequisites

- GitHub Copilot CLI installed. More information: GitHub Copilot CLI
- The **Microsoft GitHub Copilot** MCP client must be allowed in the environment. More information: Configure and manage the Dataverse MCP server for an environment

Option 1: Manually add the MCP server

You can configure the Dataverse MCP server in GitHub Copilot CLI by editing the MCP configuration file directly.

1. Open your MCP configuration file. For global configuration, edit `~/.copilot/mcp-config.json`. For project-scoped configuration, edit `.mcp/copilot/mcp.json` in your project directory.

2. Add the following JSON snippet. Replace `<your org URL>` with your Dataverse environment URL (for example, `https://contoso.crm.dynamics.com`).

   JSON

   ```
   {
     "mcpServers": {
       "DataverseMcp": {
         "type": "http",
         "url": "<your org URL>/api/mcp"
       }
     }
   }
   ```

3. Save the file and restart GitHub Copilot CLI for the changes to take effect.

Option 2: Use the Dataverse plugin from the Awesome Copilot marketplace

The Awesome Copilot marketplace provides a Dataverse plugin that includes an `mcp-configure` skill. This skill guides you through configuring the Dataverse MCP server interactively, including environment discovery and endpoint selection.

1. Add the Awesome Copilot marketplace to your Copilot CLI:

   Bash

   ```
   copilot plugin marketplace add github/awesome-copilot
   ```

2. Install the Dataverse plugin:

   Bash

   ```
   copilot plugin install dataverse@awesome-copilot
   ```

3. In a Copilot chat session, use the `/dataverse:mcp-configure` skill to configure the Dataverse MCP server. The skill walks you through selecting your environment and choosing between the generally available (`/api/mcp`) and preview (`/api/mcp_preview`) endpoints.

Related articles

Connect to Dataverse with Model Context Protocol

Last updated on 04/30/2026

Connect to Dataverse with model context protocol in non-Microsoft clients

Summarize this article for me

You can connect to Microsoft Dataverse using a non-Microsoft model context protocol (MCP) client, such as Claude desktop or Claude Code. There are two approaches for connecting non-Microsoft clients to a Dataverse MCP server:

- **Local proxy**: Use the `@microsoft/dataverse` npm package to run a local proxy that connects to the Dataverse MCP server on your behalf.
- **Remote endpoint**: Connect directly to the Dataverse MCP server remote endpoint (`/api/mcp`) by registering a custom Microsoft Entra app.

Prerequisites

- The Dataverse MCP server must be enabled for the environment. More information: Configure and manage the Dataverse MCP server for an environment
- For the local proxy approach: Node.js (version 18 or later) installed on your machine.
- For the remote endpoint approach: Access to register an application in Microsoft Entra ID.

Connect using the local proxy

The `@microsoft/dataverse` npm package provides a local proxy that handles authentication and communication with the Dataverse MCP server. This approach is recommended for most non-Microsoft MCP clients that can run local MCP servers.

Grant tenant admin consent

A tenant administrator must grant admin consent for the Dataverse CLI app before users can authenticate. Navigate to the following URL in a browser, replacing `{your-tenant-id}` with your Microsoft Entra tenant ID:

```
https://login.microsoftonline.com/{your-tenant-id}/adminconsent?client_id=0c412cc3-0dd6-449b-987f-05b053db9457
```

Sign in with a tenant administrator account and accept the permissions prompt. This step only needs to be completed once per tenant.

Enable the Dataverse CLI client in the Power Platform admin center

Before you can connect using the local proxy, the **Dataverse CLI** client must be enabled as an allowed MCP client in your environment.

1. Go to Power Platform admin center. Select **Manage** > **Environments**.
2. Select the environment where you want to enable the client, and then select **Settings**.
3. Under **Settings**, select **Product** > **Features**. Scroll down to locate **Dataverse Model Context Protocol** and select **Advanced Settings**.
4. Locate the **Dataverse CLI** client (app ID `0c412cc3-0dd6-449b-987f-05b053db9457`) and set **Is Enabled** to **Yes**.
5. Select **Save & Close**.

> **Note**
>
> If the **Dataverse CLI** entry doesn't appear in the list of available clients, you can add it manually. Create a new client entry with any name and specify the app ID `0c412cc3-0dd6-449b-987f-05b053db9457`, and then enable it.

Install the local proxy

You can install the `@microsoft/dataverse` package globally or run it directly with `npx`.

To install globally, run the following command in a terminal:

Bash

```
npm install -g @microsoft/dataverse
```

Alternatively, you can use `npx` to run the proxy without installing it globally:

Bash

```
npx @microsoft/dataverse mcp https://yourorg.crm.dynamics.com
```

> **Tip**
>
> To connect to the preview endpoint (`/api/mcp_preview`) instead of the generally available endpoint (`/api/mcp`), add the `--preview` parameter to the command. For example: `npx @microsoft/dataverse mcp https://yourorg.crm.dynamics.com --preview`. The preview

endpoint must be enabled in your environment. More information: Use preview tools and upcoming features in Dataverse MCP server

Configure the local proxy in Claude desktop

This section describes how to configure the Dataverse MCP server local proxy in Claude desktop. If you haven't already done so, download and install Claude desktop.

1. Open Claude desktop and go to **File** > **Settings** > **Developer**.
2. Select **Edit Config** to open the `claude_desktop_config.json` file.
3. Add the following JSON snippet to the file. Replace `<friendly name>` with a name you can easily remember (for example, *MyDataverseMCPServer*) and replace `<your org URL>` with your Dataverse environment URL (for example, `https://contoso.crm.dynamics.com`).

   JSON

   ```
   {
     "mcpServers": {
       "<friendly name>": {
         "command": "npx",
         "args": [
           "-y",
           "@microsoft/dataverse",
           "mcp",
           "<your org URL>"
         ]
       }
     }
   }
   ```

4. Save the file.

Verify the connection in Claude desktop

1. Exit Claude desktop by selecting **File** > **Exit**, and then reopen it to apply the changes.
2. Sign in with your credentials when prompted to authenticate to your Dataverse environment.
3. Select **Search and tools** to verify that the Dataverse MCP server and its tools are available. You should see the friendly name you configured (for example, *MyDataverseMCPServer*).
4. Select the MCP server name to view the list of tools that are supported by the server.

Tip

You can enable and disable individual tools for each MCP server registered with Claude desktop. This gives you control over which tools are available for use.

Configure the local proxy in Claude Code

This section describes how to configure the Dataverse MCP server local proxy in Claude Code. If you haven't already done so, download and install Claude Code.

Run the following command to add the Dataverse MCP server. Replace `https://yourorg.crm.dynamics.com` with your Dataverse environment URL.

Bash

```
claude mcp add dataverse -t stdio -- npx -y @microsoft/dataverse mcp
https://yourorg.crm.dynamics.com
```

Verify and interact with the connection in Claude Code

1. Restart Claude Code to apply the changes.
2. Sign in with your credentials when prompted to authenticate to your Dataverse environment.
3. Verify that the Dataverse MCP server and its tools are available. You should see the friendly name you configured (for example, *MyDataverseMCPServer*).
4. Select the MCP server name to view the list of tools that are supported by the server.

If you have data in the Dataverse environment, you can test the setup by asking *list tables in Dataverse*, *describe table account*, or *how many accounts do I have*. More information: Add and remove sample data

Tip

If you have other MCP servers registered with Claude Code, include *Dataverse* in your prompt to specify which MCP server to use.

Connect using the remote endpoint

You can connect non-Microsoft MCP clients directly to the Dataverse MCP server remote endpoint without using a local proxy. This approach requires you to register a custom application in Microsoft Entra ID and add its client ID to the allowed clients list in the Power Platform admin center.

Register a custom Microsoft Entra app

Register an application in Microsoft Entra ID to use for authentication when connecting to the Dataverse MCP server. For general information about app registration, go to Register an application with the Microsoft identity platform.

Follow these steps to register an app for use with the Dataverse MCP server:

1. Sign in to the Microsoft Entra admin center.
2. Go to **Identity** > **Applications** > **App registrations**, and then select **New registration**.
3. Enter a name for your application (for example, *Dataverse MCP Client*), configure the supported account types for your scenario, and then select **Register**.
4. On the **Overview** page, note the **Application (client) ID**. You need this value to configure the allowed client in the Power Platform admin center and to configure your MCP client.

Configure API permissions for the Dataverse MCP server

After you register the app, you must grant it permissions to access the Dataverse MCP server.

1. In the app registration, select **API permissions** from the left navigation pane.
2. Select **Add a permission**.
3. Select **Microsoft APIs**, and then select **Dynamics CRM**.
4. Select the **mcp.tools** permission, and then select **Add permissions**.

> **Note**
>
> The authentication flow used by the Entra app depends on the MCP client you're using. Refer to your MCP client's documentation for the supported authentication methods.

Add the custom app to the allowed clients list

After you register the Entra app, add its client ID to the list of allowed MCP clients for your environment.

1. Go to Power Platform admin center. Select **Manage** > **Environments**.
2. Select the environment where you want to allow the client, and then select **Settings**.
3. Under **Settings**, select **Product** > **Features**. Scroll down to locate **Dataverse Model Context Protocol** and select **Advanced Settings**.
4. Add a new client entry. Enter a name for the client and specify the **Application (client) ID** from your Entra app registration.
5. Set **Is Enabled** to **Yes**.

6. Select **Save & Close**.

Connect to the remote endpoint

Configure your MCP client to connect to the Dataverse MCP server at the following URL:

`https://<your org URL>/api/mcp`

For example: `https://contoso.crm.dynamics.com/api/mcp`

Use the **Application (client) ID** from your Entra app registration for authentication. Refer to your MCP client's documentation for specific configuration steps.

Related articles

Connect to Dataverse with Model Context Protocol

Last updated on 03/10/2026

Use preview tools and upcoming features in Dataverse MCP server

To help customers experiment with new capabilities and provide early feedback, the Microsoft Dataverse model context protocol (MCP) server includes preview tools that enable upcoming features before they're generally available (GA).

This article explains what preview tools are, what to expect when you use them, and how administrators can turn on the the preview features for Dataverse MCP server setting from the Power Platform admin center to access the latest Dataverse MCP server enhancements.

What are preview tools?

Preview tools are early versions of Dataverse MCP server capabilities released for customer evaluation. They allow makers, developers, and system integrators to:

- Try out new API endpoints and behaviors before GA.
- Validate integration scenarios.
- Provide feedback that shapes final product design.
- Prepare internal systems for upcoming releases.

Preview tools might not yet support full functionality, scale, or GA features.

Prerequisites

The environment must be enabled and configured for Dataverse MCP server. More information: Configure the Dataverse MCP server for an environment

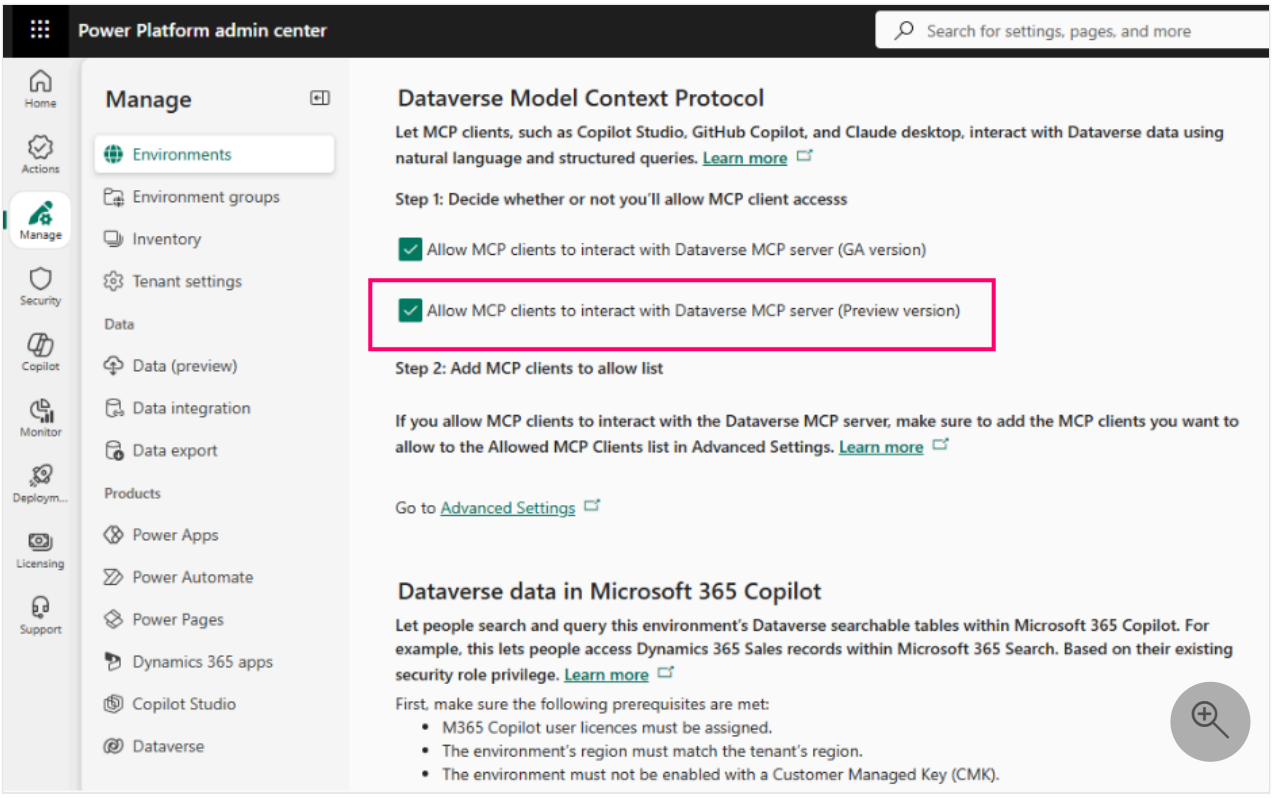
Enable preview features for Dataverse MCP server

Administrators can enable preview features using feature settings available in the Power Platform admin center. Once enabled, all users and copilots in the environment gain access to the new tools.

Follow these steps to enable preview capabilities:

1. Sign in to the Power Platform admin center
2. Select **Manage** > **Environments**.
3. Open the **Environment** where you want to turn on the Dataverse MCP server, and then select **Settings** on the command bar.
4. Expand **Product**, and then select **Features**.

5. Scroll down to locate **Dataverse Model Context Protocol** and enable **Allow MCP clients to interact with Dataverse MCP server (Preview version)**.



The environment might take a few minutes to apply the changes.

Connect to Dataverse MCP server (preview)

Once the environment setting for preview features are enabled, you can connect to the Dataverse MCP server preview endpoint.

Connect from Microsoft Copilot Studio

When adding a connector in Copilot Studio, select the connector named **Microsoft Dataverse MCP Server (Preview)**.

This connector is different from the standard **Microsoft Dataverse MCP Server** connector and is required specifically for preview features.

Connect using Claude, GitHub, or other non-Microsoft MCP clients

Use the preview endpoint `https://<orgUrl>/api/mcp_preview`.

Examples:

- `https://contoso.crm.dynamics.com/api/mcp_preview`
- `https://org123.crm4.dynamics.com/api/mcp_preview`

This endpoint makes available all preview tools and capabilities.

What happens after enabling preview for Dataverse MCP server

Once preview is enabled:

- MCP Server makes available the corresponding preview tools automatically.
- Agents and Copilots can begin calling these new endpoints immediately.
- You might notice additional system-generated logs or messages marked as **Preview**.
- Behavior might change as Microsoft iterates based on feedback.

If you encounter any issues, you can disable the **Allow MCP clients to interact with Dataverse MCP server (Preview version)** environment setting at any time.

About preview tools

Expand table

Topic	Details
Support	Preview tools aren't covered by Microsoft support agreements.
Breaking changes	Preview APIs might change without notice.
Performance	Features might not meet production-grade reliability.

Related articles

Configure the Dataverse MCP server for an environment

Last updated on 01/08/2026

Connect to Dataverse with model context protocol FAQ

Summarize this article for me

This article provides answers to frequently asked questions about using Microsoft Dataverse with a model context protocol (MCP) server.

I can't authenticate. What is the problem?

Verify that the Dataverse environment URL in your MCP client configuration is correct. Go to Power Apps, select the correct environment, and then select **Settings** (gear icon) > **Session details** to confirm your instance URL.

Also verify that the MCP client you're using is enabled in the Power Platform admin center. More information: Configure and manage the Dataverse MCP server for an environment

Which MCP tools are available and what do they do?

The Dataverse MCP server provides tools for common data operations such as querying records, creating and updating rows, describing table schemas, and listing tables. For the full list of tools and descriptions, go to Connect to Dataverse with Model Context Protocol.

Can I restrict which tables or records are accessible through the MCP server?

Yes. The Dataverse MCP server respects Dataverse security roles and row-level security. Users can only access tables and records that their security role permits. No additional MCP-specific access controls are needed beyond standard Dataverse security configuration.

Are there costs associated with using the Dataverse MCP server?

Starting December 15, 2025, Dataverse MCP tools are charged when accessed by AI agents created outside of Microsoft Copilot Studio. If you have qualifying Dynamics 365 Premium licenses or a Microsoft 365 Copilot User Subscription License (USL), you aren't charged for

accessing Dynamics 365 data. For information about billing rates, go to Connect to Dataverse with Model Context Protocol.

Can I use the Dataverse MCP server with multiple environments?

Yes. Each Dataverse environment can have its own MCP server configuration. You can connect to multiple environments by configuring separate MCP server entries in your client, each pointing to a different environment URL.

> **Tip**
>
> If you use Microsoft Copilot Studio, you can use the MCP onboarding wizard to connect your agent to Dataverse MCP servers across multiple environments. The wizard guides you through server configuration and authentication setup. More information: **Connect your agent to an existing MCP server**

What should I do if a Dataverse MCP tool returns an error?

If a tool returns an error, try rephrasing your prompt and submitting it again. Use more specific language to describe what you want to accomplish. If the error persists, verify that you have the appropriate Dataverse permissions for the operation you're attempting.

Why don't I see the Search tool in the Dataverse MCP server?

The Search tool is only available when Dataverse search is enabled for your environment. If Dataverse search isn't turned on, the Search tool doesn't appear in the list of available MCP tools. To enable Dataverse search, go to Configure Dataverse search for your environment.

Why do I see different tools on the /api/mcp and /api/mcp_preview endpoints?

The /api/mcp endpoint provides the generally available set of Dataverse MCP tools, while the /api/mcp_preview endpoint includes additional preview tools that are being evaluated before general availability. Preview tools might change or be removed without notice. To access the

preview tools, an administrator must enable the preview features setting in the Power Platform admin center. More information: Use preview tools and upcoming features in Dataverse MCP server

How do I enable debug logging for the local proxy?

If you experience issues with the local proxy (`@microsoft/dataverse`), you can enable debug logging to capture detailed output for troubleshooting. Add the `--log-level` and `--log-file` arguments to the proxy command:

Bash

```
npx @microsoft/dataverse mcp https://yourorg.crm.dynamics.com --log-level Debug --log-file
```

The log file is written to the system temporary directory. The default location depends on your operating system:

Expand table

Operating system	Log file location
Windows	`C:\Users\<username>\AppData\Local\Temp\`
Linux	`/tmp/`
macOS	The directory specified by the `$TMPDIR` environment variable, typically `/var/folders/.../T/`. Run `echo $TMPDIR` in a terminal to find the exact path.

You can share the output log file with Microsoft when opening a support incident to help diagnose the issue.

Last updated on 03/11/2026

What is Dataverse intelligence?

[This article is prerelease documentation and is subject to change.]

Microsoft Dataverse intelligence extends Work IQ to bring business data understanding to AI agents and Microsoft Copilot. Work IQ helps agents understand work artifacts like files, meetings, and messages. Dataverse intelligence builds on this foundation by enabling agents to understand and act on your business data.

With Dataverse intelligence, you can define reusable business context that agents use to understand what your data means, follow your organization's processes when taking actions, and read and update Dataverse records reliably. This business context is shared across agents, so you define it once and use it everywhere.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Prerequisites

- Microsoft 365 admin role (AI administrator, Global administrator) to access Microsoft 365 admin center Copilot settings. More information: Enable Microsoft 365 admin center Copilot Dataverse settings
- Power Platform administrator role to access Dataverse intelligence environment settings. More information: Enable Dataverse intelligence
- The environment where you use Dataverse intelligence must be a Managed Environment.
- The environment must be enabled and configured for Dataverse MCP server preview. Business skills are only available for use with the preview version of Dataverse MCP server. More information: Use preview tools and upcoming features in Dataverse MCP server

Enable Microsoft 365 admin center Copilot Dataverse settings (preview)

1. Go to Microsoft 365 admin center. Select **Copilot** > **Settings**.
2. Locate and select **Dataverse data available in Microsoft 365 Copilot**.

3. Select **All users**, or
4. Select **Specific groups** and enter the list of Entra security groups.
5. Select **Save** to save the setting changes.

Enable Dataverse intelligence (preview)

1. Go to Power Platform admin center. Select **Manage** > **Environments**.
2. Open the environment where you want to turn on the Dataverse MCP server, and then select **Settings** > **Product** > **Features**.
3. Scroll down to locate **Dataverse intelligence**.

    - Turn on **Allow data availability in M365 copilot**, and/or...
    - Turn on **Enable Dataverse intelligence (Work IQ) for agents and AI experiences**.

4. Make sure **Allow MCP clients to interact with Dataverse MCP server (Preview version)** is enabled. If it's not, enable it.
5. Select **Save** to save the setting changes.

Next steps

Business skills overview

Create and use business skills

Last updated on 03/30/2026

Business skills overview

Summarize this article for me

[This article is prerelease documentation and is subject to change.]

Business skills are natural-language instructions that capture how your organization gets work done. They represent your business processes, policies, and domain knowledge in a format that agents can understand and follow. Each skill describes how to complete a specific type of work—the steps involved, the information required, and the business rules that apply.

Agents discover and use business skills as needed at runtime to complete tasks according to your organization's processes. When multiple agents use the same skill, they follow the same process, ensuring consistent behavior across your organization.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

> **Important**
>
> Business skills are not executable code. They contain natural-language instructions that guide agent behavior, similar to how you might document a process for a new employee.

Key capabilities include:

- Reusability—Define a process once and use it across multiple agents and surfaces
- Consistency—Agents follow the same process definition, reducing variation in outcomes
- Maintainability—Update a skill in one place to change behavior across all agents that use it
- Governance—Control who can create, modify, and share business skills using Dataverse security. Each skill has an owner; sharing follows Dataverse RBAC patterns
- Solution-aware—A solution aware object that appears in Solution explorer and can be moved across environments

Skill structure

Business skills use a layered structure for efficient discovery and retrieval.

Expand table

Layer	Content	Purpose
Metadata	Name and description	Enables agents to quickly discover relevant skills
Instructions	Full skill body	Contains the complete process definition

Agents query metadata to find applicable skills, then retrieve full instructions only when needed.

Security and governance

Business skills follow Dataverse security patterns. Each skill has an owner who controls access through standard sharing mechanisms. Users these security roles can work with business skills.

Expand table

Security role	Privileges
Basic User	Can create skills (owned by them) Can use skills shared with them. Can't publish environment-wide.
Environment Maker	Can create and share skills (owned by them) Can manage solution deployment
System Administrator or System Customizer	Can create and share skills Can manage solution deployment

Next steps

Create and use business skills

Configure the Dataverse MCP server for an environment

Last updated on 03/10/2026

Create and use business skills

[This article is prerelease documentation and is subject to change.]

This article explains how to create and manage business skills that ground agents in your organization's processes. You'll learn how to enable the required features, author skill content, test skills with agents, and manage skills across environments.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and might have restricted functionality. These features are subject to supplemental terms of use, and are available before an official release so that customers can get early access and provide feedback.

Prerequisites

- The environment must be enabled and configured for Dataverse MCP server preview. Business skills are only available for use with the preview version of Dataverse MCP server.
- The environment where you create and use business skills must be a Managed Environment. More information: Use preview tools and upcoming features in Dataverse MCP server
- The environment must be enabled for Dataverse intelligence.

Open the business skills page

1. Go to Power Apps.
2. In the left navigation pane, select **More**, and then select **Business skills (preview)**.

> **Tip**
>
> Pin the **Business skills** page to the left navigation pane for quick access.

The **Business skills** page is the central hub for managing all business process knowledge in your environment. From here, you can create, edit, share, and organize the skills that agents use to follow your organization's processes.

Manage business skills

Create a business skill

1. Select **New business skill** on the command bar, or select the **Create a business skill** card.
2. Enter a name for your skill. A unique name is generated automatically with the preferred solution's prefix.
3. In the **Description** field, describe the skill's purpose and when it should be followed. Agents use this description to understand when to perform the skill.
4. In the **Instructions** section, write the step-by-step process in markdown format. Use the **Edit markdown** and **Preview** options to switch between editing and previewing your instructions.
5. Select **Save**.
6. After saving, you can add resource files in the **Resources** section. Select **Add Files** to attach reference materials such as policy documents, SOPs, templates, forms, or calculations (20 MB limit).

> **Note**
>
> All skills are active by default. Deactivate a skill

Upload a business skill

You can quickly create a business skill by uploading existing skills.

1. Select **Upload business skill** on the command bar, or select the **Upload a business skill** card.
2. Drag and drop a file into the upload area, or select **Select from device** to browse for a file.
3. Upload a `Skill.md` (markdown) or `.zip` file of your skill (30 MB limit).
4. The business skill gets uploaded and saved. You can edit the skill once uploaded.

Edit a business skill

1. Open a skill in the **Business skills** page by selecting its name in the list, or select a skill and then select **Edit** from the command bar or the three-dot menu (⋮).
2. Update the **Name**, **Description**, **Instructions**, or **Resources** as needed.
3. Select **Save**.

Create or update skills using the Dataverse MCP server (preview)

You can also create and update skills through an agent connected to the Dataverse MCP server (preview):

1. Add Dataverse MCP server (preview) as a tool to any agent in Microsoft Copilot Studio or connect to it from coding agents in Visual Studio Code or non-Microsoft clients.
2. Ask your agent to create a new skill or update an existing one.
3. Provide your business process information.
4. The business skill is saved in the environment you used to connect to the MCP server.

Share a business skill

Skills are user-owned by default. You can share skills with specific users or teams and control their level of access.

1. Select a skill, and then select **Share** from the command bar or the three-dot menu (⋮).
2. Search for and add the users or teams you want to share with.
3. For each user or team, choose a permission level:

    - **Viewer** - Can view the skill only.
    - **Co-owner** - Can view and edit the skill.

4. Select **Share**.

Select **Manage access** to review who has access, change permission levels, or remove access. Pending invitations that haven't been shared yet are shown on the **Pending invites** tab.

Change skill visibility

You can make a business skill visible to everyone in the organization or restrict it to just the owner and shared users.

1. Select a skill, and then select **Viewable by** from the command bar or the three-dot menu (⋮).
2. In the **Viewable by** dialog, select one of the following:

    - **Individual** - Only the owner and users the skill is shared with can see the skill. This is the default.
    - **Organization** - All users in the environment with at least basic user privileges can see the skill.

3. Select **Save**.

Deactivate a skill

To prevent a skill from being used by agents until you have finalized it:

1. Select a skill, and then select **Deactivate** from the command bar or the three-dot menu (⋮).

Deactivating a skill doesn't delete it, but prevents agents from discovering or using it. To reactivate, select the skill and then select **Activate**.

Delete a skill

To permanently remove a business skill:

1. Select a skill, and then select **Delete** from the command bar or the three-dot menu (⋮).

> ⊗ **Caution**
>
> Deleting a skill is permanent and can't be undone. Any agents that reference the deleted skill can no longer use it.

Use business skills

After you create business skills in your environment, AI agents can use them to understand and run business processes. Access skills through the Dataverse MCP server (preview).

- To start using business skills, connect to the Dataverse MCP server (preview) in Microsoft Copilot Studio or from agent mode in Visual Studio Code or non-Microsoft clients.
- Try asking your agent, "Show me all business skills in this environment." The agent retrieves a list of all the skills in your connected environment by using the Dataverse MCP server.
- Start testing by asking your agent a relevant scenario that matches your skill's intended use case. For example, if you created a skill for logging call transcripts into Dataverse, provide a sample transcript to your agent and ask it to log the transcript information in Dataverse.

  > 💡 **Tip**

> If your agent doesn't automatically fetch skill details, try adding "Using business skills" before your actual prompt. For example, "Using business skills log this transcript in Dataverse."

- Remember to add any other relevant tools, including MCP servers and connectors, that the agent needs access to for successfully executing your business processes defined in skills.
- Confirm your intended actions were successful and continue to iterate on the skill instructions based on results.

Add business skills to a solution

Business skills are solution-aware, which means you can package them into solutions for application lifecycle management (ALM). This allows you to move skills between environments as part of your deployment process.

To add a skill to a solution:

1. Sign in to Power Apps and select your environment.
2. Go to **Solutions** and open an existing solution or create a new one.
3. Select **Add existing** > **Business skill** from the top navigation bar.
4. Choose the skills you want to add to your solution, and then select **Add**.

Best practices for creating business skills

- Write clear, specific instructions. Vague instructions lead to inconsistent agent behavior.
- Use descriptive names and descriptions. Help agents discover the right skill for the task.
- Keep skills focused. One skill, one process. Create separate skills for distinct workflows.
- Test with real scenarios. Validate behavior before sharing broadly.
- A well-structured business skill should include:
  - Description:
    - What the skill does and when to use it.
    - Specific trigger phrases. For example, "Use when user asks about order status or mentions shipment tracking".
  - Step-by-step instructions:
    - Numbered steps in the process sequence.
    - What happens at each stage and how to validate before moving to the next step.
  - Examples:
    - Concrete examples showing expected inputs and outputs for common scenarios.
  - Troubleshooting:

    - Common errors with causes and solutions.
    - Recovery steps for failures.
  - Input and output specifications:
    - Required vs optional parameters with expected formats.
    - Output fields and success criteria.
    - Required tools for successful execution of skill.

Sample business skills for getting started

To help you get started, sample business skills are available: Business skills repo on GitHub

This open source GitHub repository also has a downloadable solution. Install the solution in your environment to explore working examples and use them as templates for your own skills. Refer to the ReadMe file located in the repository for more instructions on how you start using these sample skills.

Next steps

Business skills overview

Last updated on 04/20/2026

License requirements for tables

Article • 02/08/2023

App makers can use most of the tables available within Microsoft Dataverse (including custom tables and tables that are part of the Common Data Model) to create apps and flows for users who have a Power Apps or Power Automate license.

In some cases, tables contain complex business logic or are tied to customer engagement apps in Dynamics 365 (Dynamics 365 Sales, Customer Service, Field Service, Marketing, or Project Service Automation) that require app users to have a specific license. This topic provides licensing information for tables with complex business logic and tables that are tied to Dynamics 365 apps (termed as restricted tables).

Tables with complex business logic

Tables that include the following complex server-side logic require users of an app or flow that uses these tables to have a Power Apps or Power Automate license:

- Code plug-ins (for more information, see Plug-in development)
- Real-time workflows (for more information, see Workflow processes)

> **Note**
>
> Only workflows that are converted to a real-time workflow are considered real-time and synchronous. Workflows that are run in the background can still be used with the appropriate Power Apps plan and do not require additional licenses.

To know whether or not you added complex business logic to your tables, review the list of plug-in assemblies and workflows configured in your environment. For the list of tables which may contain server side logic after installing a Dynamics 365 app (such as Dynamics 365 Sales or Dynamics 365 Customer Service), see Complex tables requiring Power Apps or Power Automate licenses

Impacting license requirements when adding complex business logic

App makers can add code plug-ins and real-time workflows to tables within Dataverse, but doing so could change the license requirements for users of apps already deployed.

App makers should be cautious when adding complex business logic to a table and should first check which apps use the table and whether users of those apps have the appropriate licenses.

Restricted tables

Restricted tables are not standard tables within Dataverse, but are included in one of the customer engagement apps in Dynamics 365 (Dynamics 365 Sales, Customer Service, Field Service, Marketing, or Project Service Automation) or a third-party solution. For example, the knowledge article, goal, and entitlement tables.

> **Note**
>
> Apps and flows that use these tables require the app and flow user to be licensed appropriately-not the maker or developer of the app or flow.

tables that are tied to the functionality of Dynamics 365 apps (such as Dynamics 365 Sales or Dynamics 365 Customer Service) require app users to have the corresponding license for that application if they want to create, update, or delete rows within the tables. For a full list of restricted tables, see Restricted tables requiring Dynamics 365 licenses.

Licensing examples

Barb and Isaac are creating apps in Power Apps using Dataverse to store their data.

Table creation

- No user can create a new restricted table; Microsoft reserves the right to create and define them for Dynamics 365 apps (such as Dynamics 365 Sales or Dynamics 365 Customer Service)
- Users can create custom tables with Dynamics 365, Power Apps, or Power Automate license
- For existing restricted tables, a user can add rows with the appropriate Dynamics 365 app license

Create apps using Power Apps

- Barb and Isaac can create a canvas or model-driven app accessing restricted tables with a Dynamics 365 license
- Barb and Isaac can create a canvas or model driven app accessing custom tables with either Dynamics 365 or Power Apps license

Use apps

Barb wants to use two canvas apps:

- App 1 – uses the Appointment table along with a custom table that stores related information
- App 2 – uses the Appointment table along with the Work Order table, which is a restricted table

Isaac wants to use two model-driven apps:

- App 3 – uses the Appointment table along with a custom table that stores related information
- App 4 – uses the Appointment table along with the Work Order table, which is a restricted table

Barb and Isaac need the following licenses:

- Barb can use App 1 with a Dynamics 365 app license or a Power Apps license.
- Barb can use App 2 only with a Dynamics 365 app license because there is a restricted table accessed by the app.
- Isaac can use App 3 with a Dynamics 365 app license or a Power Apps license.
- Isaac can use App 4 only with a Dynamics 365 app license because there is a restricted table accessed by the app.

Create flows using Power Automate

Now, let's see what happens when Isaac adds a real-time workflow to the custom table that both Barb and Isaac are using in their apps.

- Isaac can create a workflow accessing restricted tables with a Dynamics 365 app license

- Barb and Isaac can create a workflow accessing custom tables with either Dynamics 365 app or Power Automate license

Use flows

- Barb or Isaac can run the workflow accessing restricted tables with a Dynamics 365 app license
- Barb or Isaac can run the workflow accessing custom tables with either Dynamics 365 app or Power Automate license

More about licensing

For more information about licensing, see Licensing overview.

For the latest information on licensing requirements for tables, see the Power Apps licensing guide.

Complex tables and licensing

Article • 02/15/2022

> **Important**
>
> *This topic is applicable only for older Power Apps Plan 1 and Plan 2 licenses.*
>
> Complex tables are applicable *only* for the older Power Apps Plan 1 and Plan 2 licenses, and not for the latest Power Apps per app and Power Apps per user plans.
>
> For the latest information on licensing requirements for tables, see the **Power Apps licensing guide** .

Tables that include the following complex server-side logic require users of an app or flow that uses these tables to have a Power Apps Plan 2 or Power Automate Plan 2 license:

- Code plug-ins. More information: Plug-in development
- Real-time workflows. More information: Workflow processes

> **Note**
>
> Only workflows that are converted to a real-time workflow are considered real-time and synchronous. Workflows that are run in the background can still be used with the appropriate Power Apps plan and do not require additional licenses.

To know whether or not you've added complex business logic to your tables, review the list of plug-in assemblies and workflows configured in your environment.

Complex tables installed with Dynamics 365 apps

The following table lists tables that contain complex server-side logic out-of-the-box as part of the installation of customer engagement apps in Dynamics 365 (Dynamics 365 Sales, Customer Service, Field Service, Marketing, and Project Service Automation). This list is intended as a guide. Depending on the Dynamics 365 apps and versions installed in your environment, the list of complex tables may vary.

> **Note**
>
> If you are using the Microsoft Dataverse and have not installed a Dynamics 365 app or third-party solution, your environment won't have tables containing complex server-side logic.

- Account
- Agreement
- Agreement Booking Date
- Agreement Booking Incident
- Agreement Booking Product
- Agreement Booking Service
- Agreement Booking Service Task
- Agreement Booking Setup
- Agreement Invoice Date
- Agreement Invoice Product
- Agreement Invoice Setup
- Agreement Sub-Status
- Bookable Resource
- Bookable Resource Booking
- Bookable Resource Booking Header
- Bookable Resource Category
- Bookable Resource Category Assn
- Bookable Resource Characteristic
- Bookable Resource Group
- Booking Alert
- Booking Alert Status
- Booking Status
- Characteristic
- Competency Requirement (Deprecated)
- Competitor
- Contact
- Customer Asset
- Delegation
- Expense
- Field Computation
- Field Service Price List Item
- Filter
- Follow
- Incident Type

- Incident Type Product
- Incident Type Service
- Incident Type Service Task
- Integration Job
- Integration Job Detail
- Inventory Adjustment
- Inventory Adjustment Product
- Inventory Transfer
- Invoice
- Invoice Frequency
- Invoice Line
- Invoice Line Detail
- Journal
- Journal Line
- Lead
- Note
- OData v4 Data Source
- Opportunity
- Opportunity Line
- Opportunity Line Detail
- Order
- Order Invoicing Product
- Order Invoicing Setup
- Order Line
- Payment
- Payment Detail
- Post Configuration
- Post Rule Configuration
- Postal Code
- Price List
- Price List Item
- Product
- Project
- Project Approval
- Project Contract Line Detail
- Project Contract Line Milestone
- Project Contract Line Resource Category
- Project Contract Line Transaction Category
- Project Parameter
- Project Stages

- Project Task Status User
- Project Team Member Sign-Up
- Purchase Order
- Purchase Order Bill
- Purchase Order Product
- Purchase Order Receipt
- Purchase Order Receipt Product
- Purchase Order Sub Status
- Queue Item
- Quote
- Quote Booking Incident
- Quote Booking Product
- Quote Booking Service
- Quote Booking Service Task
- Quote Booking Setup
- Quote Invoicing Product
- Quote Invoicing Setup
- Quote Line
- Quote Line Detail
- Quote Line Milestone
- Quote Line Resource Category
- Quote Line Transaction Category
- Quote Project Price List
- Rating Model
- Rating Value
- Requirement Characteristic
- Requirement Resource Category
- Requirement Resource Preference
- Requirement Status
- Resource Request
- Resource Requirement
- Resource Requirement Detail
- RMA
- RMA Product
- RMA Receipt
- RMA Receipt Product
- RMA Sub-Status
- Role competency requirement
- Role Price
- RTV

- RTV Product
- RTV Sub-Status
- Tax Code
- Tax Code Detail
- Time Entry
- Time Group
- Time Group Detail
- Time Off Request
- Transaction Category Price
- User
- View
- Wall View
- Warehouse
- Work Order
- Work Order Incident
- Work Order Product
- Work Order Service
- Work Order Service Task
- Work Order Sub-Status
- Work template

Restricted tables requiring Dynamics 365 licenses

App makers, who are building custom apps, can use all of the Microsoft Dataverse tables available within Common Data Model to create apps and flows for users who have any version of Power Apps and Power Automate license.

However, a smaller set of tables tied to Dynamics 365 apps (Dynamics 365 Sales, Customer Service, Field Service, Marketing, or Project Service Automation) require canvas and model-driven app users to have a license for the corresponding Dynamics 365 app if they need to create, update, or delete rows within the tables. These are referred to as restricted tables.

Tables might be restricted to a Dynamics 365 app license for the following reasons:

- The table is used to store and maintain product-specific configuration data that should typically be not used outside of the application.
- The table is accompanied by advanced logic that creates and maintains data in a specific way when used within a Dynamics 365 product.

If an app or flow only reads information from a table, a Dynamics 365 app license isn't required and an appropriate Power Apps or Power Automate license is all that's needed.

Restricted tables for create, update, and delete operations

The following table lists the restricted tables and the associated Dynamics 365 license requirements for users who create, update, or delete data stored within these tables.

> **Note**
>
> Dynamics 365 Customer Engagement Plan and Dynamics 365 Plan licenses are no longer available for purchase and is referenced here only for the benefit of existing customers who have bought these licenses in the past.

Expand table

Table	Logical name	License required
Actual	msdyn_actual	Dynamics 365 for Field Service **or** Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Agreement Business Process	msdyn_bpf_baa0a411a239410cb8bded8b5fdd88e3	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Booking journal	msdyn_bookingjournal	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Booking Setup Metadata	msdyn_bookingsetupmetadata	Dynamics 365 for Field Service **or** Dynamics 365 for Project Service Automation **or** Dynamics 365 for Customer Service Enterprise edition **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Booking timestamp	msdyn_bookingtimestamp	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Case[1]	incident	Dynamics 365 for Customer Service Professional edition **or** Dynamics 365 for

Table	Logical name	License required
		Customer Service Enterprise edition **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Case to Work Order Business Process	msdyn_bpf_989e9b1857e24af18787d5143b67523b	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Configuration	msdyn_configuration	Dynamics 365 for Field Service **or** Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Entitlement	entitlement	Dynamics 365 for Customer Service, Enterprise edition **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Estimate Line	msdyn_estimateline	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Estimate	msdyn_estimate	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Fact	msdyn_fact	Dynamics 365 for Project Service Automation **or** Dynamics 365

Table	Logical name	License required
		Customer Engagement plan **or** Dynamics 365 plan
Field service setting	msdyn_fieldservicesetting	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Field Service System Job	msdyn_fieldservicesystemjob	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Goal	goal	Dynamics 365 for Sales Professional, **or** Dynamics 365 for Sales, Enterprise edition, **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Inventory Journal	msdyn_inventoryjournal	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Invoice Process	msdyn_bpf_d8f9dc7f099f44db9d641dd81fbd470d	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Journey	journey	Dynamics 365 for Marketing **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan

Table	Logical name	License required
Knowledge article	`knowledgearticle`	Dynamics 365 for Customer Service, Enterprise edition **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Organizational Unit	msdyn_organizationalunit	Dynamics 365 for Field Service **or** Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Product Inventory	msdyn_productinventory	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Project Parameter	msdyn_projectparameter	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Project Stages	msdyn_bpf_665e73aa18c247d886bfc50499c73b82	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Project Task Dependency	msdyn_projecttaskdependency	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Project Task	msdyn_projecttask	Dynamics 365 for Project Service Automation

Table	Logical name	License required
		or Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Project Team Member	msdyn_projectteam	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Purchase Order Business Process	msdyn_bpf_2c5fe86acc8b414b8322ae571000c799	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Resource Assignment Detail (Deprecated)	msdyn_resourceassignmentdetail	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Resource Assignment	msdyn_resourceassignment	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Resource Restriction (Deprecated)	msdyn_workorderresourcerestriction	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Routing rule set	`routingrule`	Dynamics 365 for Customer Service, Enterprise edition **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan

Table	Logical name	License required
Schedule Board Setting	msdyn_scheduleboardsetting	Dynamics 365 for Field Service **or** Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Scheduling Parameter	msdyn_schedulingparameter	Dynamics 365 for Field Service **or** Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
SLA	`sla`	Dynamics 365 for Customer Service Professional edition **or** Dynamics 365 for Customer Service Enterprise edition **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
System User Scheduler Setting	msdyn_systemuserschedulersetting	Dynamics 365 for Field Service **or** Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Transaction Connection	msdyn_transactionconnection	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan

Table	Logical name	License required
Transaction Origin	msdyn_transactionorigin	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Transaction Type	msdyn_transactiontype	Dynamics 365 for Project Service Automation **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Unique Number	msdyn_uniquenumber	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Work Order[2]	msdyn_workorder	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan
Work Order Details Generation Queue (Deprecated)	msdyn_workorderdetailsgenerationqueue	Dynamics 365 for Field Service **or** Dynamics 365 Customer Engagement plan **or** Dynamics 365 plan

[1] *Case table actions permitted with only Power Apps, Power Automate, Power Pages or Microsoft Copilot Studio license:*
Power Apps, Power Automate, or Copilot Studio licensed users can ‘create’ cases; can ‘read’, ‘update’ and ‘delete’ self-created cases to enable scenarios including employee self-service and case creation on behalf of customers. Power Apps, Power Automate, or Copilot Studio licensed users can only perform ‘read’ operation on cases created by other users. They can't ‘update’, ‘resolve’, ‘route’, ‘close’,‘delete’,‘assign’,‘duplicate’, ‘merge’, ‘create child cases’ or perform any other custom operation on cases created by others. Users can't act as a customer service agent, can't manage cases.

[2] *Work Order table requires Dynamics 365 for Field Service license in most scenarios except for users self-reporting issues (for example, an employee reporting a facilities issue) in which case creating, reading, updating, and deleting work orders is permitted with the Power Apps or Power Pages license. Users can't complete/close work orders or act in a manner similar to that of performing or completing service.*

Licensing

For more information on Power Apps, Power Automate, Power Pages, Copilot Studio, and Dynamics 365 licenses, see Licensing overview page.

Last updated on 04/27/2026

SharePoint, OneNote, and OneDrive integration with Microsoft Dataverse

Article • 02/15/2022

Dataverse provides support for SharePoint, OneDrive, and OneNote integration. Integration with these services requires that you first enable SharePoint integration.

Microsoft 365 service	Description	More information
SharePoint	Lets app users manage common document types, such as Word, Excel, PowerPoint, OneNote, and create folders to save and manage those documents that are seamlessly stored in SharePoint from within Dataverse apps.	Manage your documents using SharePoint Set up SharePoint integration
OneDrive for Business	App users can create and manage private documents that can be accessed from within Dataverse apps.	Enable OneDrive for Business
OneNote	App users can use OneNote to take or review notes from within Dataverse rows.	Set up OneNote integration

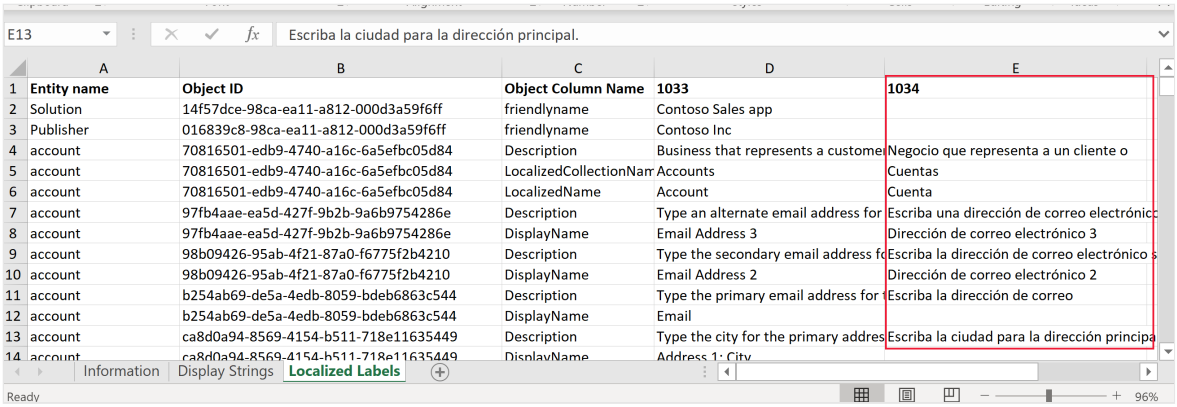
Translate customized table, form, and column text into other languages

Article • 03/13/2023

After you create customized table and column text in your unmanaged solution, you may want to translate it into other languages.

> **Important**
>
> When you export translations, the export translations feature exports translations for the table. So, that means even if the solution contains only a single form, labels for all the forms for the table will be exported. Make sure you only modify the form's labels when importing the translations back otherwise another component translation you modify will be added as a dependency to the solution.

1. Sign into Power Apps and select **Solutions** from the left navigation. If the item isn't in the left navigation pane, select ...More and then select the item you want.
2. Select the unmanaged solution you want, on the command bar select **...**, select **Translations**, and then select **Export translations**.
3. After the export completes the exported translations compressed (zip) file is downloaded to your browser's default download folder and contains the exported labels.
4. Extract the XML file from the compressed (.zip) file.
5. Open the CrmTranslations.xml file in Excel.
6. Select the sheet named **Localized Labels**.
7. Notice there is already a column with the base language code id, such as 1033 (English U.S.) Add a column with the language code id for every language you want to translate labels. For example, add a column for 1034 (Spanish traditional).
8. Add the translated text in the new column for the object names and object ids that you want.



	A	B	C	D	E
1	**Entity name**	**Object ID**	**Object Column Name**	**1033**	**1034**
2	Solution	14f57dce-98ca-ea11-a812-000d3a59f6ff	friendlyname	Contoso Sales app	
3	Publisher	016839c8-98ca-ea11-a812-000d3a59f6ff	friendlyname	Contoso Inc	
4	account	70816501-edb9-4740-a16c-6a5efbc05d84	Description	Business that represents a custome	Negocio que representa a un cliente o
5	account	70816501-edb9-4740-a16c-6a5efbc05d84	LocalizedCollectionNam	Accounts	Cuentas
6	account	70816501-edb9-4740-a16c-6a5efbc05d84	LocalizedName	Account	Cuenta
7	account	97fb4aae-ea5d-427f-9b2b-9a6b9754286e	Description	Type an alternate email address for	Escriba una dirección de correo electrónic
8	account	97fb4aae-ea5d-427f-9b2b-9a6b9754286e	DisplayName	Email Address 3	Dirección de correo electrónico 3
9	account	98b09426-95ab-4f21-87a0-f6775f2b4210	Description	Type the secondary email address fo	Escriba la dirección de correo electrónico s
10	account	98b09426-95ab-4f21-87a0-f6775f2b4210	DisplayName	Email Address 2	Dirección de correo electrónico 2
11	account	b254ab69-de5a-4edb-8059-bdeb6863c544	Description	Type the primary email address for t	Escriba la dirección de correo
12	account	b254ab69-de5a-4edb-8059-bdeb6863c544	DisplayName	Email	
13	account	ca8d0a94-8569-4154-b511-718e11635449	Description	Type the city for the primary addres	Escriba la ciudad para la dirección principa
14	account	ca8d0a94-8569-4154-b511-718e11635449	DisplayName	Address 1: City	

9. When you're finished adding your translations, save, and zip up the package so you can Import translated table and column text.

> **Note**
>
> Not all content can be translated using this tool. This includes labels for sitemap areas, groups, and subareas. To translate these labels, **use the More Titles property in the sitemap designer**.

Community tools

Easy Translator is a tool that XrmToolBox community developed for Power Apps. Use Easy Translator to export and import translations with contextual information.

> **Note**
>
> The community tools are not supported by Microsoft. If you have questions about the tool, please contact the publisher. More Information: **XrmToolBox**.

Next steps

Import translated table and column text

Import translated table, form, and column text back into an app

Article • 03/13/2023

If you have customized table or column text, such as column labels or drop-down list values, you can provide the users in your organization who are not working with the base language version of your environment with this customized text in their own languages. To do so, you export the text strings for all your customizations so that they can be translated into the languages you use in your organization.

After the translation, you need to import the translated text strings into your environment before users can take advantage of the changes.

> **Important**
>
> - The file that you import must be a compressed file that contains the CrmTranslations.xml and the [Content_Types].xml file at the root.
> - You can't import translated text that is over 500 characters long. If any of the items in your translation file are longer than 500 characters, the import process will fail. If the import process fails, review the line in the file that caused the failure, reduce the number of characters, and try to import again. Also note that after you import translated text, you must republish customizations.

1. Sign in to Power Apps at https://make.powerapps.com.
2. Select **Solutions**, and select the unmanaged solution from which to import the translated text. If the item isn't in the left navigation pane, select ...More and then select the item you want.
3. In the solution explorer, on the Actions toolbar, select **Translations**, and then select **Import translations**.
4. In the **Import Translated Text** dialog box, specify the file that contains the translated text, and then select **Import**.
5. When the import is complete, select **Close**.

> **Note**

Publishing customizations can interfere with normal system operation. We recommend you schedule publishing when it's least disruptive to users.

Community tools

Easy Translator is a tool that XrmToolBox community developed for Power Apps. Use Easy Translator to export and import translations with contextual information.

> **Note**
>
> The community tools are not supported by Microsoft. If you have questions about the tool, please contact the publisher. More Information: **XrmToolBox**.

Next steps

Export customized table and column text for translation

Privileges required for Dataverse customization

App users can personalize the system and even share some of their customizations with others. Only users with the correct privileges can apply changes for everyone.

> **Note**
>
> - This section assumes you know how to work with security roles. For more information about working with security roles, see Security roles and privileges.
> - For some feature and data access, a system customizer security role might be required. More information: A copied system customizer security role might not provide the same access.

System administrator and system customizer security roles

Anyone who customizes the system needs at least the system customizer security role associated with their account. This security role gives you the permission you need to customize in Microsoft Dataverse.

Expand table

System administrator	System customizer
Has full permission to customize the system and manage the environment.	Has permission to customize the system.
Can view all data in the system.	Can view all custom tables in the system but only view rows (records) in the account, contact, and activity tables that they create.

The difference between the system administrator and system customizer security roles is that a system administrator has read privileges on most records in the system and can see everything. Assign the system customizer role to someone who needs to perform customization tasks and has access to all the custom tables but only has access to the account, contact, and activity rows (records) they create. However, testing is an important part of customizing the system. If

system customizers can't see any data, they need to create rows (records) to test their customizations. By default, system customizers have full access to custom tables. If you want to have the same limitations that exist for system tables, you need to adjust the system customizer security role so that the access level is **User** rather than **Organization** for custom tables.

Add a security role to a user

For information about how a Power Platform admin can add a security role to a user, go to Add users to a security role

Delegate customization tasks

You might want to delegate some tasks to trusted people so that they can apply changes they need. Keep in mind that anyone can have multiple security roles associated with their user account and that privileges and access rights granted by security roles are based on the *least restrictive* level of permissions.

This means that you can give the system customizer security role to someone who already has another security role, perhaps a sales manager. This access lets them customize the system in addition to other privileges they already have. You don't need to edit the security role they already have, and you can remove the system customizer security role from the person's user account when you want.

Test customizations without customization privileges

Always test customizations by using a user account that doesn't have customization privileges. This approach ensures that people without the system administrator or system customizer security roles can use your customizations. To test effectively, you need access to two user accounts: one account with the system administrator security role and another account that has the security roles representing the people who use the customizations.

> ⓘ **Important**
>
> Don't attempt to remove your system administrator security role if you have only one user account. The system warns you if you try, but if you proceed, you might not be able to get it back. Most security roles don't allow editing of a user's security roles.

Next steps

Understand model-driven app components

Last updated on 04/08/2026

Microsoft Dataverse API limits overview

Article • 01/31/2023

Dataverse API limits help ensure service levels, availability, and quality. Dataverse API limits are part of the Power Platform Request limits and allocations. This article will introduce limits specifically for Dataverse applicable for Power Apps, Power Automate, and customer engagement apps (Dynamics 365 Sales, Dynamics 365 Customer Service, Dynamics 365 Field Service, Dynamics 365 Marketing, and Dynamics 365 Project Service Automation) connecting to Dataverse.

For information about limits for all areas within Power Platform, see Power Platform Request limits and allocations.

There are two categories of limits that apply for Dataverse: *Entitlement* and *Service protection* limits as summarized below.

Expand table

Service Protection limits	**Power Platform Request (API Entitlement) limits**
Immediately return 429 Too Many Requests when limits are exceeded.	Enforced for Power Automate flows but will have provisions for occasional overages on Dataverse, when enforced
Evaluated in a 5-minute sliding window.	Evaluated in a 24-hour period.
Applies to all external requests to Dataverse web services.	Applies to all compute operations and analytics originating from internal or external requests.
Developer is responsible for avoiding conditions that can cause errors and managing retry when they occur.	Administrator is responsible to manage assigned capacity. They can purchase capacity add-ons to increase limits.
Enforced today within the product.	- Will be enforced after preview report, available today for administrators to track and estimate usage, is made generally available. -Preview reports are available now.
More information: https://aka.ms/serviceprotectionlimits	More information: https://aka.ms/PlatformLimits

Entitlement limits

These limits represent the number of requests users are entitled to make each day. The allocated limit depends on the type of license assigned to each user.

If any user exceeds their request entitlement, the administrator would be notified and would be able to assign Power Apps and Power Automate request capacity to that user. Users won't be blocked from using apps for occasional and reasonable overages at this point of time.

For Dataverse, API requests include all data operations that interact with table rows where rows are created, retrieved, updated, or deleted (CRUD). Special operations such as *share* and *assign* are included because they're considered updates. These requests can be from any client or application and using any endpoint. These include, but aren't limited to, operations performed by plug-ins, async workflows, custom controls, and $batch (ExecuteMultiple) operations. There's a small set of system internal operations that are excluded, like sign in, sign out, and system metadata operations.

> **Important**
>
> Power Platform API request allocations include use of Power Automate, AI Builder, and Connector APIs. All requests through a connector that result in a Dataverse request will represent 1 Power Platform request.

For details on these entitlement limits, see Microsoft Power Platform requests allocations based on licenses.

For information about viewing and allocating capacity add-ons, see Capacity add-ons.

For information about purchasing individual capacity add-ons see the Power Apps and Power Automate Licensing Guide.

Service protection limits

To ensure consistent availability and performance for everyone we apply some limits to how APIs are used with Dataverse. Service protection API limits help ensure that users running applications can't interfere with each other based on resource constraints. The limits won't affect normal users of the platform. Only applications that perform a large number of API requests may be affected. The limits provide a level of protection from random and unexpected surges in request volumes that threaten the availability and performance characteristics of the Dataverse platform.

We limit the number of concurrent connections per user account, the number of API requests per connection, and the amount of execution time that can be used for each

connection. These metrics are evaluated within a five-minute sliding window. When one of these limits is exceeded, an exception will be thrown by the platform.

> **Note**
>
> Service protection limits apply to all external web service requests, not only the CRUD operations on tables counted against entitlement limits.
>
> Service protection API limits aren't applied against API calls made within workflows, custom workflow activities, or plug-in code. These operations are invoked internally.

Service protection limits are only encountered by applications that perform a high volume of data operations. We recommend that developers building applications that perform a high volume of data operations apply patterns to retry operations after a period of time when these exceptions are returned. Applying these patters will allow the application to respond to exceptions the service sends and reduce the total number of requests and achieve the highest possible throughput.

For information about the specific errors that can be returned and how developers can apply patterns to respond to these errors, see Service Protection API Limits.

See also

Administer Power Platform / Licensing and license management / Requests limits and allocations
Developer / Work with data using code / Service Protection API Limits

Migrate Microsoft Access data to Microsoft Dataverse

Article • 04/26/2023

Microsoft Access users can now choose to migrate their data into Dataverse or Dataverse for Teams to make use of Azure cloud security and Microsoft Power Platform functionality. Migrating Access data to Dataverse provides many new opportunities for interaction and management of data.

Watch this short video about migrating Access data to Dataverse.
https://learn-video.azurefd.net/vod/player?id=c151d1a0-3a30-4715-95cb-aab380611592&locale=en-us&embedUrl=%2Fpower-apps%2Fmaker%2Fdata-platform%2Fmigrate-access-to-dataverse

Migration is handled in Access using the export feature, and the migration tool that streamlines the process.

> **Note**
>
> Learn how to prepare MS Access for Migration

Once migrated, Access users can continue using their existing desktop client to manage their data. They're also able to use Power Platform to manage their data. Plus, they can:

- Create applications for desktop, laptop, phone, and tablet.
- Create automated processes based on the data and services in Dataverse or Dataverse for Teams.
- Create AI-driven virtual assistants based on the data.
- Perform deep analysis on the stored data using business intelligence.

Before you migrate, there are a few key factors to consider. Currently, Dataverse and Dataverse for Teams don't support every data type that is in Access. Additionally, some data types in Dataverse and Dataverse for Teams may have different data limits. More information: Data types and sizes for Access data migration to Dataverse

Dataverse and Dataverse for Teams offer differing features that you can use to best meet your needs. Dataverse for Teams is targeted at users who want to have Power Platform functionality within the Microsoft Teams application. Dataverse is included with licenses of Dynamics 365 or Power Apps. Organizations that have a Dataverse for Teams environment can choose later to upgrade to Dataverse.

Expand table

Dataverse	Dataverse for Teams
All available data types supported.	Relational data, image, and file data support, built on the core of Dataverse.
Dataverse enterprise-grade capabilities, including advanced Role-Based Access Control, highly extensible business logic, sophisticated reporting, and robust offline.	Use apps, chatbots, and automations backed by Dataverse embedded in Teams on any platform.
Existing premium connectors to enterprise systems such as SAP, Oracle, and so on.	Some data types are not supported or have minimal functionality (Currency, Date Only, Customer).
Standalone premium apps, bots, and automations outside of Teams.	Not all controls are available, such as bar code scanner, interface tools, and formulas.

Total storage capacity is also different. Dataverse and Dataverse for Teams have a heterogenous data storage platform, which allows for storing different types of data in ways that help improve performance and lower cost of maintenance.

Access

Dataverse[1]

Dataverse for Teams

2-GB storage on local machine

- 10-GB relational database
- 20-GB file and image blob storage
- 2-GB log storage in non-relational storage
- You can purchase additional capacity as needed in any of the store types
- 2-GB storage (up to 1 million records) per Team
- Images, files, and logs stored separately in blob storage (counts towards 2-GB limit)

[1] Single base license

For more information, go to How are Dataverse for Teams and Dataverse different?

Comparison of Access and Dataverse Terminology

Users integrating Access with Dataverse or Dataverse for Teams may find some differences in names for general features and functions. This section provides a guide to understand naming differences.

Unlike Access, Dataverse and Dataverse for Teams are data sources. The user interface layer will depend on what Power Platform feature you are using: Power Apps, Power Automate, Power BI, or Microsoft Copilot Studio. The lists below encompass naming differences and similarities at the database layer only.

Expand table

Access	Dataverse and Dataverse for Teams	Comments
Primary Key	Primary Key	The usage is the same but the key contents are different
Calculated Field	Calculated Column (property)	Dataverse does not have a calculated column data type but rather allows calculations to be created on individual data types like whole number, decimal, or text
Row	Row or record	Sometimes referred to as a record in Dataverse and Dataverse for Teams
Short Text	Text	Supported for migration
Long Text	Multiline Text	Supported for migration
Rich Text	Rich Text	Supported for migration
Number: Single	Float	Supported for migration
Number: Double	Float	Supported for migration
Large Number	Big Integer (BigInt)	Supported for migration, usable through Dataverse API, Power Apps UI does not currently support this
Number: Decimal	Decimal	Supported for migration
Yes/No	Yes/No	Supported for migration
Int	Whole Number	Supported for migration

Access	Dataverse and Dataverse for Teams	Comments
Multi-Select Options	Choices	Supported for migration
GUID	Unique Identifier	In Dataverse and Dataverse for Teams, this is currently only used as a key
Hyperlink	URL	Supported for migration
Unique Index	Alternate Key	Not supported for migration
Multi-Value Lookup	Choice	Supported for migration

Migrating Access databases into Dataverse solutions

To enable application lifecycle management (ALM), Dataverse uses solutions to allow for the management, and distribution of content including apps, tables, and more. A solution is a package of components such as tables, relationships, or other items that can be distributed across multiple Dataverse environments through solution export and import.

The Access migration tool allows users to choose if they wish to export their Access databases directly into a solution. Access databases in solutions are exported as an unmanaged solution. A solution must exist or be created in Dataverse prior to the migration to be used for the migration process. To learn more about solutions, go to: Solution Concepts

See also

Data types and sizes for Access data migration to Dataverse

Feedback

Was this page helpful? Yes No

Provide product feedback

Data types and sizes for Access data migration to Dataverse

Article • 01/26/2023

When you migrate from Microsoft Access to Microsoft Dataverse or Microsoft Dataverse for Teams, you should be aware of a few differences in the data types. These differences include supported types, data type names, and column capacity.

When you migrate, a validation will be executed to ensure:

- The table only includes supported data types.
- The column values in the rows being migrated don't exceed the size limits of Dataverse.

This validation is done to prevent data loss. If a table has columns that exceed the maximum column value in Dataverse, or the table contains data types not supported by Dataverse, the user will be alerted by the Access migration tool validator and will be provided additional information.

Users can choose to either cancel the migration completely, or to continue to migrate all supported content and keep the unsupported content in an Access table.

> **Note**
>
> Learn how to prepare MS Access for Migration

Access data types supported by Dataverse

In the following table, the data type mappings supported can assist you in planning your data migration.

Access data type	Dataverse data type	Can migrate?
Short Text	Text	Yes
Long Text	Multiline text	Yes
Autonumber	Autonumber	Yes
Date/Time	Date and Time	Yes
Currency	Currency	Yes
Number: Decimal	Decimal Number	Yes
Float	Number:Single, Number:Double	Yes[1]

Access data type	Dataverse data type	Can migrate?
Yes/No	Yes/No	Yes
Int	Whole Number	Yes
Lookup Wizard	Lookup	Yes
Multi-Value Lookups	Choice	Yes[2]
Hyperlink	URL	Yes
Attachment	File	Yes[3]

[1]Dataverse includes a float data type. However, it has lower limits than Access a min/max of +/- 100,000,000,000 or +/- 99,999,999,999.99999. A maximum of five decimal places is supported. Users will be informed of this limitation during migration and can opt out if they require greater than five decimal places.

[2]One column multi-value lookups only. Because of the difference in how Dataverse and Access identify these lookups, a manual process is needed in Access before migration. More information: Export multi-value lookup Access Fields to Dataverse choice columns

[3] Attachments can be migrated for single attachments per column at this time.

Access data types not supported for migration to Dataverse

- OLE Object
- Calculated[4]
- Rich Text

[4]When you migrate, the calculated field will migrate the last calculated value into a column. Users will need to configure new calculation columns in Dataverse. More information: Define calculated columns to automate calculations

Access and Dataverse data size comparison

You'll notice some Dataverse columns don't have the same size capacity as Access. As noted above, if a column contains data too large to be migrated, the migration tool alerts the user that the contents can't be migrated. This is to prevent data loss. This decision isn't based on the maximum possible size for the column, but rather the size of the actual data in each row.

Access/Dataverse data type	Access limit	Dataverse limit
Short Text/Text	255 characters	4,000 characters

Access/Dataverse data type	Access limit	Dataverse limit
Long Text/Multiline Text	1 GB	1,048,576 characters
Autonumber	2,147,483,647	4,000 characters
Date and Time	Standard date and time	Standard data and time
Currency[5]	Min/max -922,337,203,685,477/+922,337,203,685,477	Min/max -922,337,203,685,477/+922,337,203,685,477
Decimal Number	Min/max -10^28-1/+10^28-1 up to 28 decimals	Min/max -100,000,000,000/+100,000,000,000 up to 10 decimal places
Float (Number:Single)	-3.402823E38 to -1.401298E-45 for negative values and 1.401298E-45 to 3.402823E38 for positive values.	Min/max -100,000,000,000/+100,000,000,000 up to five decimal places
Float (Number:Double)	-1.79769313486231E308 to -4.94065645841247E-324 for negative values and 4.94065645841247E-324 to 1.79769313486231E308 for positive values.	Min/max -100,000,000,000/+100,000,000,000 up to five decimal places
Yes/No	Boolean	Boolean
Int/Whole Number	Min/max -2^31/+2^31	Min/max -2,147,483,647/+2,147,483,647
Lookup Wizard/ Lookup	Multiple column return	Single column return

[5]The migration tool assumes the currency coming from Access is the Dataverse base currency.

Calculated fields in Access will currently create a column for the content type in Dataverse that stores the calculated value. This could be text, whole number, decimal, and so on. Dataverse can be used to create calculated fields to enable calculations.

Export multi-value lookup Access Fields to Dataverse choice columns

Both Access and Dataverse offer users the ability to provide a list of multiple values a user can select from in a row. The ways these are implemented are different.

Access uses a multi-value lookup, which is a lookup that allows the user to enter any number of values, which will then be presented as a dropdown list for selection. Access has the ability to have more than one column of values for this function.

Dataverse uses choice columns, which are an enumerated list of values that each have a string label associated to them. Users locate and choose the values in a dropdown list using the label values that are stored in the background as an enum selection with a relationship to the table where the labels are stored.

Because of these differences, migrating multi-value lookup columns from Access into Dataverse presents some challenges. The following process must be followed to migrate:

1. Access can only start with a single column multi-value lookup.
2. Access users must add a new column to the multi-value lookup to act as the enum value expected by Dataverse.
3. At export, Dataverse will store this as two columns, which allows both Dataverse and Access to use the lookup appropriately in forms.

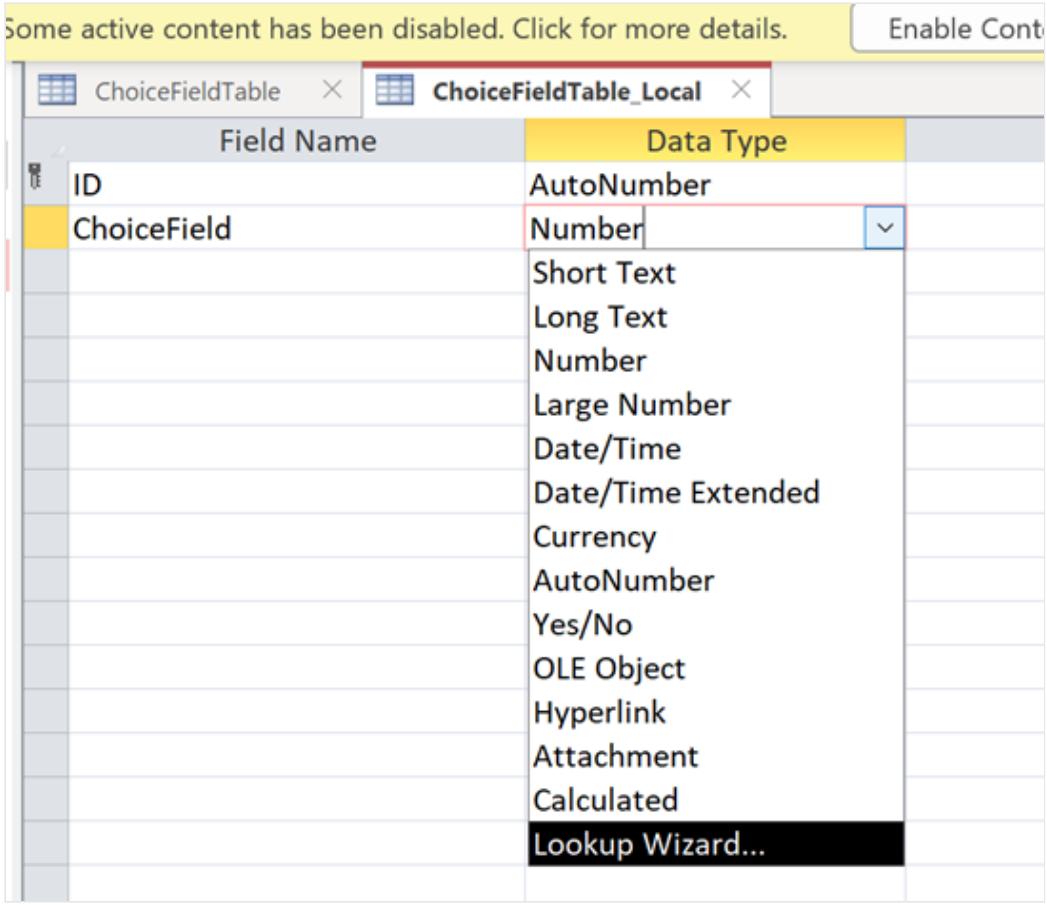
Creating a valid choice field for export to Dataverse

To successfully migrate a choice field from Access, the field must be created in a manner that is similar to the steps described here.

1. Create a new table in Access.

2. Add a **Number** field to the table. It must be a **Number** field to support export to Dataverse.

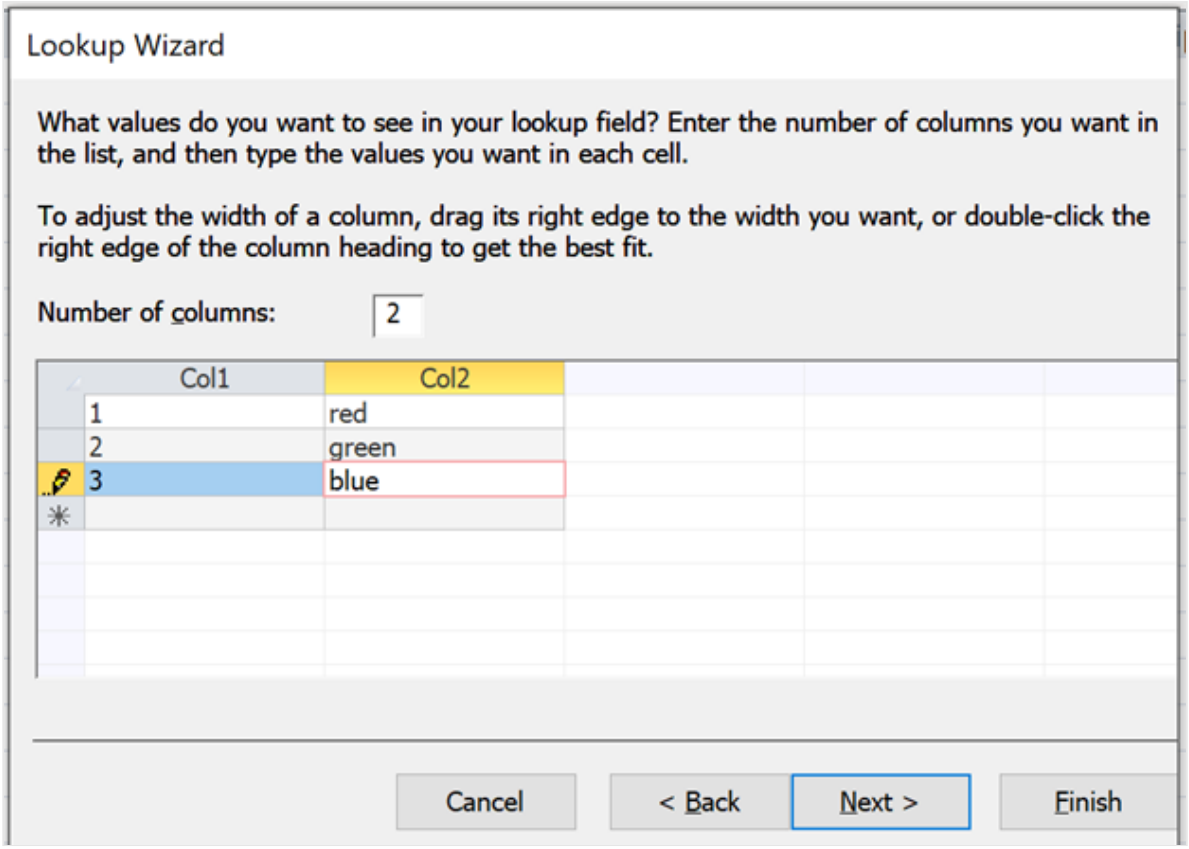
3. Go to **Design View** in Access. Select the new field, and then select **Lookup Wizard**.



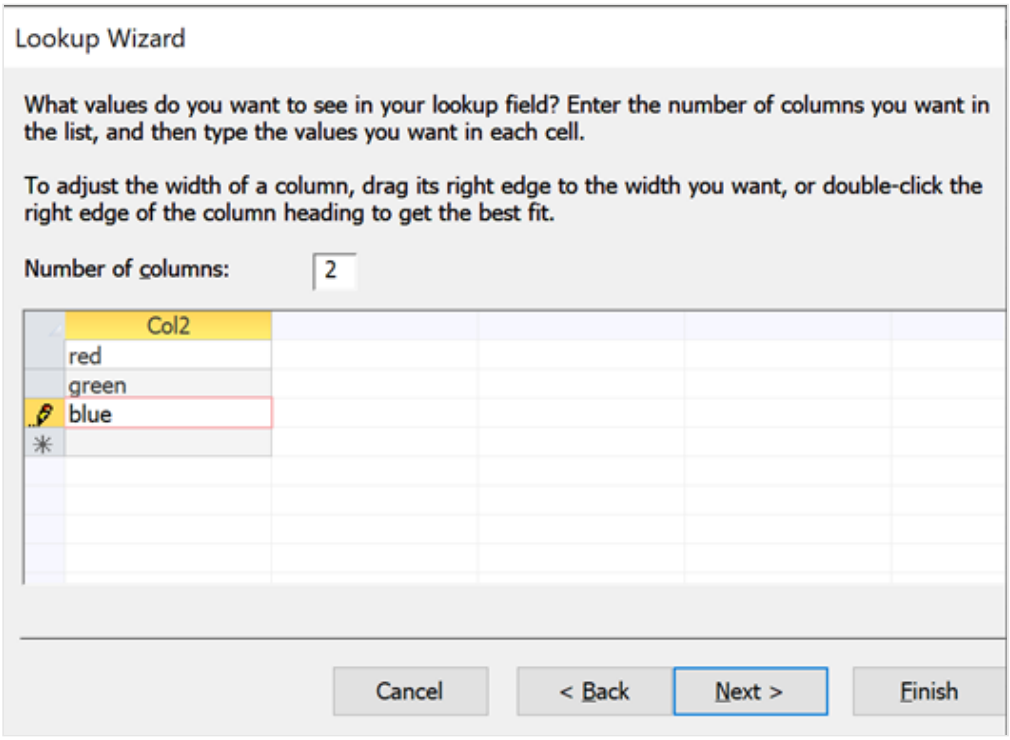
4. In the Lookup Wizard, select the **I will type in the values that I want** option, and then select **Next**.

5. In the next dialog, enter *2* for the number of columns and then select the field below **Col1**.

6. In **Col1** enter values for three rows by entering *1*, *2*, and *3*. In **Col2** enter values for three rows by entering *red*, *green*, and *blue*.



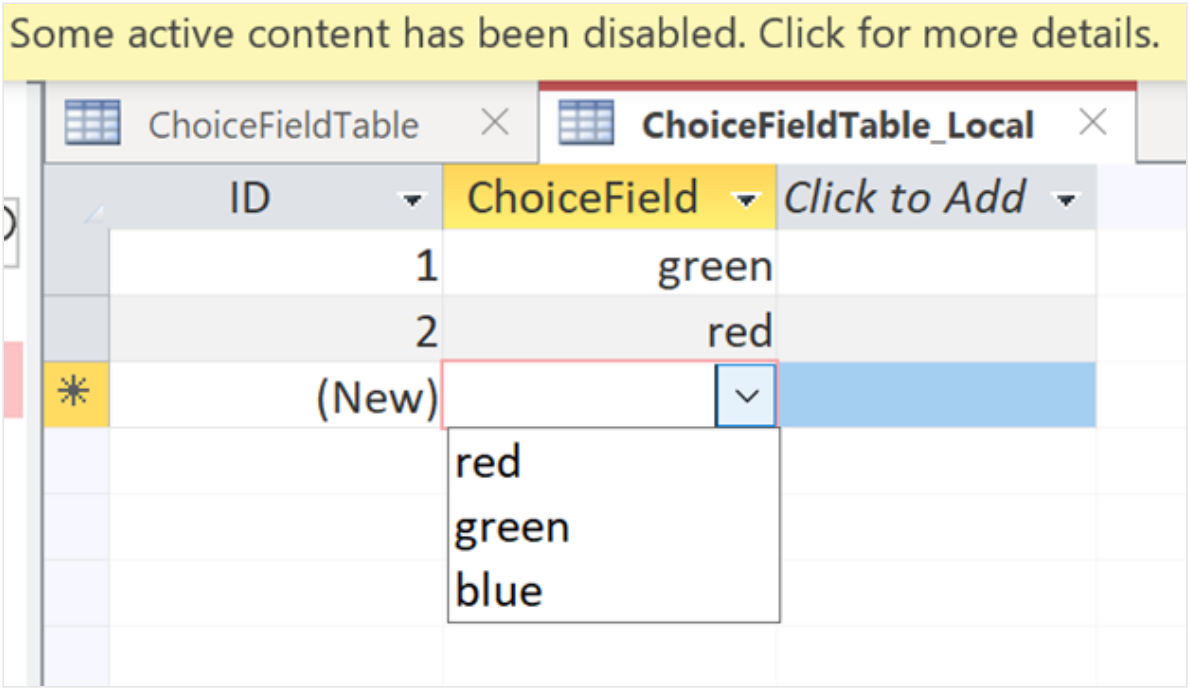
7. Select the separator between **Col1** and **Col2** and slide it to the left so that only **Col2** appears.



8. Select **Next** > **Next** > **Finish** to complete the Lookup Wizard.

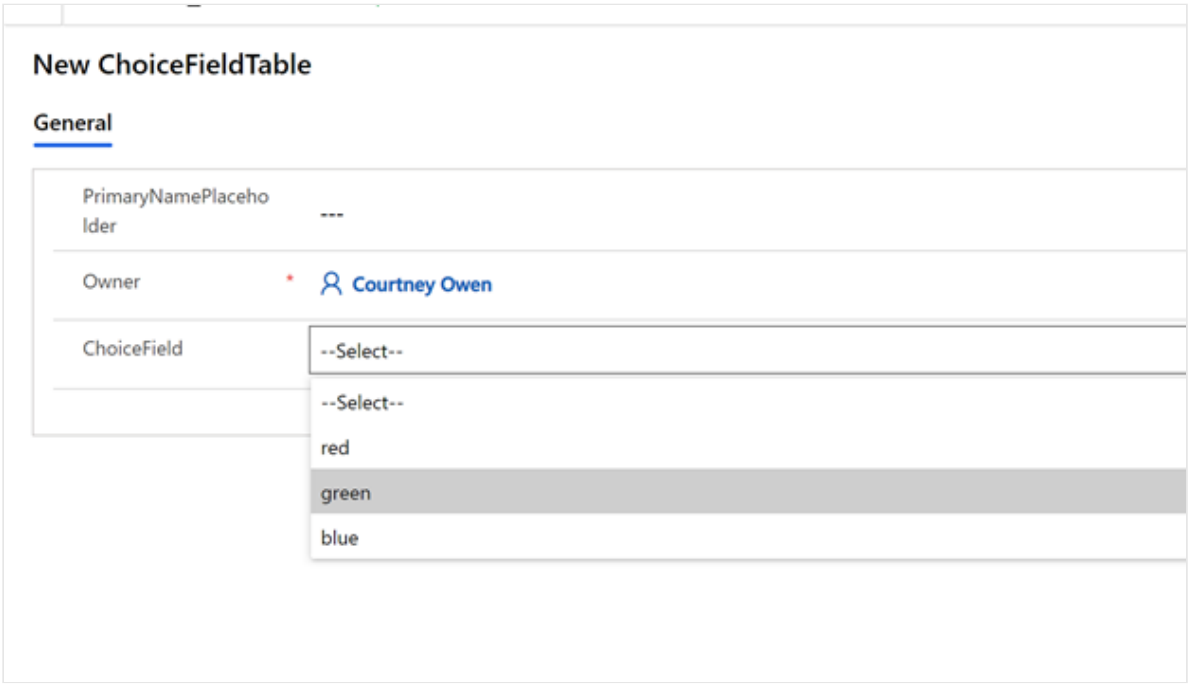
9. Save your changes and return to the **Datasheet View**.

10. Create a new record to validate the choice field works as expected. It should appear like this if it was configured properly.



11. Close the table and migrate it to Dataverse.

12. If everything worked properly, the choice column in Dataverse will appear like this when adding a new record.



Migrate Number:Single and Number:Double columns to Dataverse

Both Access and Dataverse include the ability to store floating point numbers. Access uses `Number:Single` and `Number:Double` for this. These data types are often used for any number column. Dataverse has a **Floating Point Number** data type, but it has some limitations with how it's implemented. Dataverse only allows a maximum of five decimal places with a minimum/maximum value of -/+ 100,000,000,000. When migrating, the validator informs you that by migrating this data as a float you'll lose some decimal precision. If that's acceptable, you can migrate the tables automatically. During the migration process any rows that exceed the min/max size limits won't be migrated and will be stored locally in a file in Access just as with any other out of range data type.

However, if you need to migrate but don't want to lose decimal precision, you can migrate Access `Number:Single` and `Number:Double` by changing the data type in Access. You can use the Access table designer and change the type of `Number:Single` and `Number:Double` columns to **Decimal**, which support up to 10 decimal places in Dataverse. Then any row that doesn't exceed the minimum/maximum decimal limits can be migrated.

See also

Migrate Microsoft Access data to Microsoft Dataverse

System tables and columns in Dataverse and Dataverse for Teams

Article • 05/11/2022

Both Dataverse and Dataverse for Teams have several tables and columns that need to be present for data types and features to work properly. These are part of the default organizations and cannot be removed. When migrating from Access to Dataverse, you'll see columns added to tables after the migration and additional linked tables added in Access to support the columns. This document provides an overview of the tables and columns that will be added.

> ⓘ Note
>
> Learn how to prepare MS Access for migration.

Tables

The following tables are included in environments of Dataverse and Dataverse for Teams and may appear as a linked table in Access after migration. These tables can't be deleted.

- **Currency**: Used for currency transactions.
- **Users**: Stores users of the environment that use the apps or data. Used in lookup columns that are created by default.
- **Teams**: Stores teams of users in the environment that use the apps or data. Used in lookup columns that are created by default.
- **Business Unit**: Stores business units users belong to in the environment that use the apps or data. Used in lookup columns that are created by default.

> ⓘ Note
>
> - The tables are present and usable but not displayed by default in Dataverse for Teams.
> - The Currency table is linked in Access only when a currency column is added to the table.

Columns

The following columns are included in all tables of Dataverse and Dataverse for Teams and may be visible in Access linked tables. These columns can't be deleted.

- **Primary Name**: A user-friendly text string used to identify rows. The name is chosen by the user. In Access you can choose which field you want to use during migration, or it will be picked for you. If an Access table has no text fields, a text field will be created named Primary Name Placeholder which will be used to fill this need. It will be set to "optional" and will accept null values.
- **Primary ID**: A unique identifier string used to identify rows. This is system generated and is the primary key. This column is present and usable but not displayed by default in Dataverse and Dataverse for Teams environments.
- **CreatedBy**: A lookup column to the Users table selecting the user who created the row.
- **CreatedBy (Delegate)**: A lookup column to the Users table selecting the user who created the row as a delegate for another user.
- **CreatedOn**: A date-time column that records the date and time a row was created, stored in UTC format.
- **Currency**: A lookup to the Currency column (logical name `Transactioncurrency`) to retrieve the details of a transaction currency for the purposes of calculating against the base currency. This column is visible in Access linked tables only when a currency column is added to the table.
- **Exchange Rate**: The exchange rate value retrieved from the transaction currency on the Currency table at the time the row was created. The column is visible in Access linked tables only when an exchange rate column is added to the table. This column is present and usable but not displayed by default in Dataverse, Dataverse for Teams environments, or Access linked tables.
- **Import Sequence Number**: The row's place in the sequence of imported records that added it to the table.
- **ModifiedBy**: A lookup column to the Users table selecting the user who modified the row.
- **ModifiedBy (Delegate)**: A lookup column to the Users table selecting the user who modified the row as a delegate for another user.
- **ModifiedOn**: A date-time column that records the date and time a row was modified, stored in UTC format.
- **Owner**: The unique OwnerID assigned to the user selected as the Owning User from the Users table. This column is present and usable but not displayed by default in Access linked tables.
- **Owning Business Unit**: A lookup to the Business Unit table to select the appropriate business unit that owns the row.

- **Owning Team**: A lookup to the Teams table to select the appropriate team that owns the row.
- **Owning User**: A lookup to the Users table to select the appropriate user that owns the row.
- **Record Created On**: Date and time that a row was migrated.
- **Status**: An option set that allows a user to pick the state of a row.
- **Status Reason**: An option set related to Status that allows a user to pick the status reason of a row based on the status selected.
- **Time Zone Rule Version**: Internal-only used to determine which set of time zone rules apply.
- **UTC Conversion Time Zone Code**: The time zone offset code that was in use at the time the row was created.
- **Version Number**: The version number of a row used to track previous changes if auditing is enabled.

See also

Migrate Microsoft Access data to Microsoft Dataverse

Data types and sizes for Access data migration to Dataverse

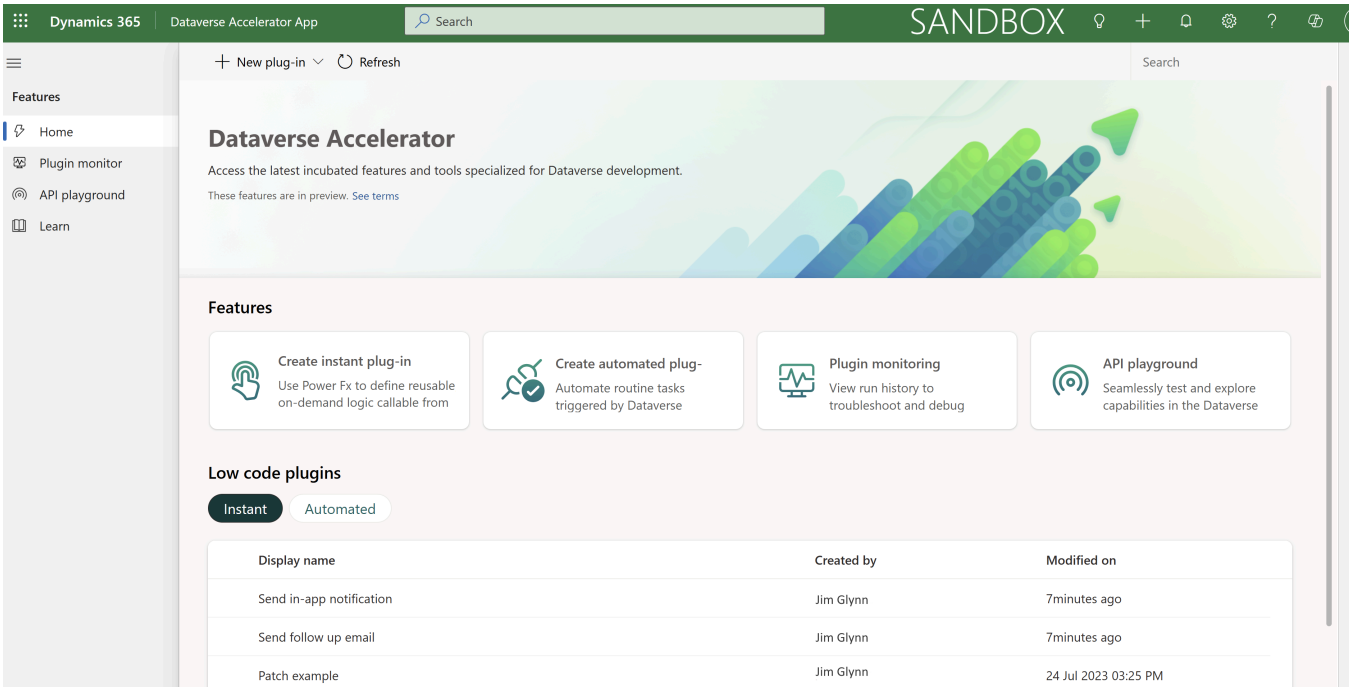
Get preview features early with the Dataverse accelerator (preview)

Summarize this article for me

[This topic is pre-release documentation and is subject to change.]

> **Important**
>
> The Dataverse accelerator, monitor plug-ins, and API playground features are deprioritized and aren't be delivered as features. For information about preview feature availability, go to Power Apps preview program.

The Microsoft Dataverse accelerator is a model-driven application that provides access to select preview features and tooling related to Dataverse development. The featured set of capabilities include early prototype experiences that represent backlog features scheduled to be in the native platform and tools for enriching Dataverse development for makers, delivered in a convenient and accessible web application built with low-code.



While preview features shouldn't be used in production instances, the accelerator offers the opportunity to proactively evaluate, share feedback, and prepare for integration. Using features available with the Dataverse accelerator helps you leverage cutting-edge capabilities to enhance productivity, optimize Dataverse processes, and maintain a competitive edge in Power Platform.

> **Important**

- This is a preview feature.
- Preview features aren't meant for production use and might have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

The Dataverse accelerator interface adheres to the same UX and accessibility standards as any Microsoft Power Platform experience. Furthermore, it serves as a demonstration of our platform's powerful low-code capability to ship enterprise-grade applications.

Features available with the Dataverse accelerator

Expand table

Feature	Description
Low-code plug-ins	Reusable, real-time workflows that execute a specific set of commands within Dataverse. Low-code plug-ins run server-side and are triggered by personalized event handlers, defined in Power Fx. For more information about how to use this feature, go to Low-code plug-ins.
Plug-in monitor	A modern interface to surface the existing plug-in trace log table in Dataverse environments, designed for developing and debugging Dataverse plug-ins and custom APIs. For more information about how to use this feature, go to Plug-in monitor.
API playground	A preauthenticated software testing tool that helps makers quickly and conveniently interact with the Dataverse Web API. For more information about how to use this feature, go to API playground.

Note

If all features aren't available in the Dataverse accelerator app, update the app. More information: update the Dataverse accelerator app.

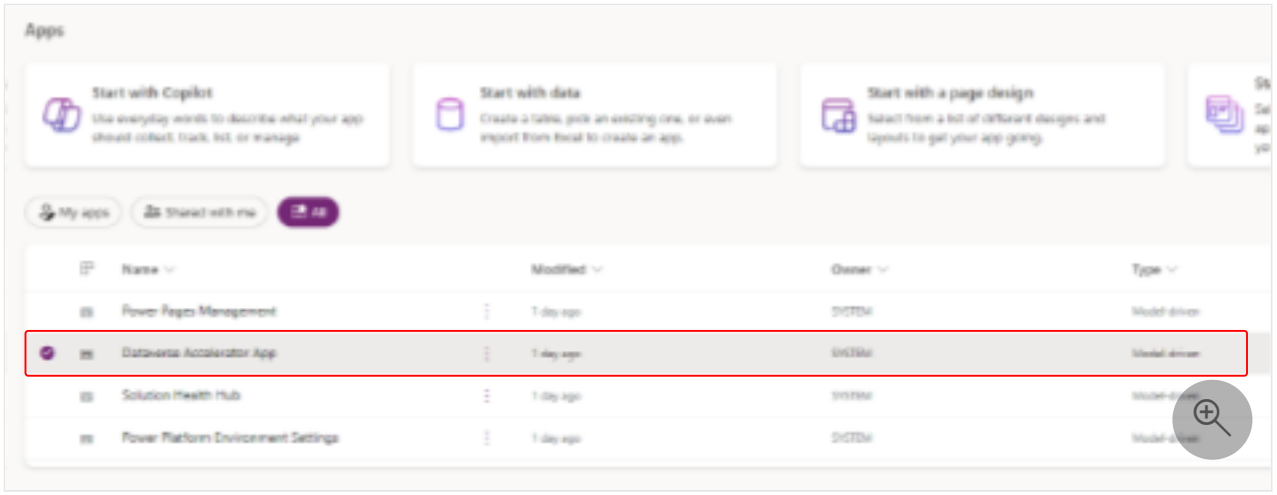
Play the Dataverse accelerator

Play the Dataverse accelerator app in any environment where it's installed.

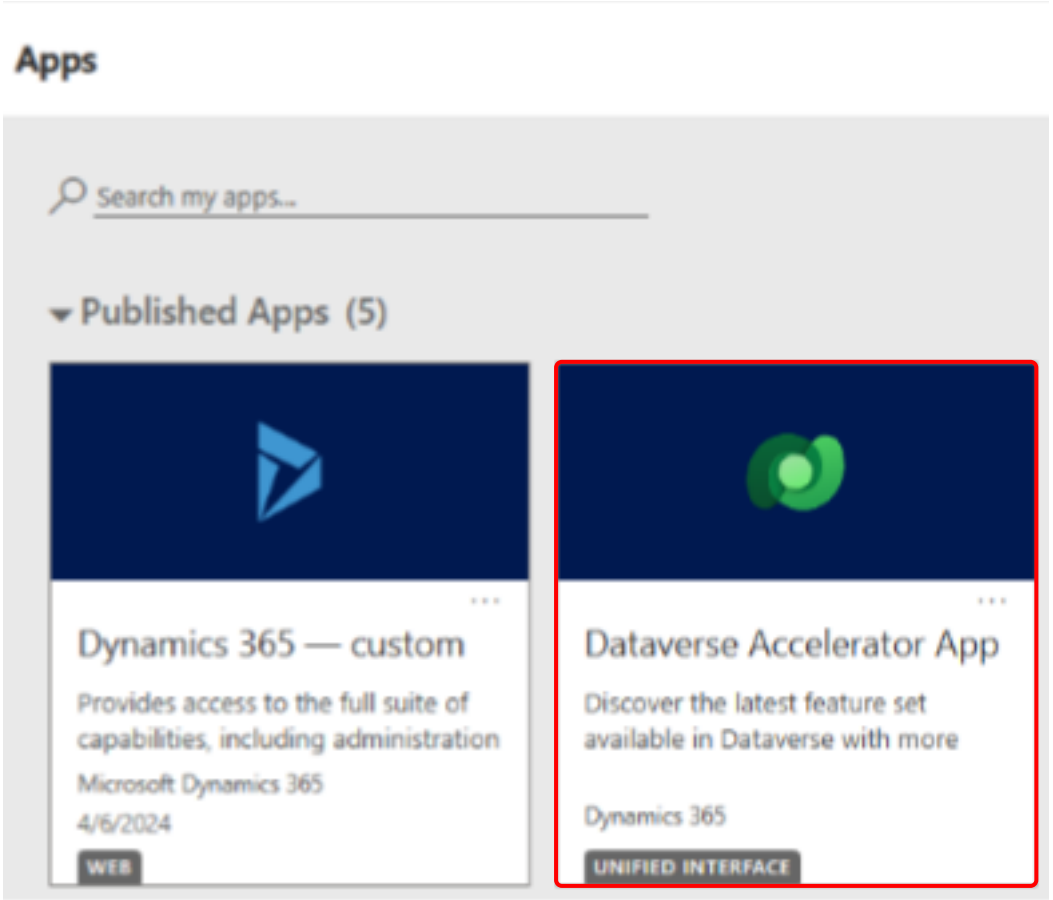
The app is automatically available in all new Microsoft Dataverse environments. If your environment doesn't already have it, you can install the Dataverse accelerator.

The app appears in two places:

1. In Power Apps, select **Apps**



2. On the Power Apps or Dynamics 365 home page.



When you play the Dataverse accelerator, features are displayed in the accelerator that you can try out.

Prerequisites to run the Dataverse accelerator

You must have the following privileges in your Dataverse environment:

- Security role: System customizer.

- App-level access to the Dataverse accelerator model driven app, such as system customizer or direct access from a security role.

Prerequisites for managing the app

You must have the following security role assignment and your Dataverse environment must be the same or later than the minimum version:

- Security role: You must have the system administrator security role in the environment.
- A Dataverse environment with minimum database (DB) version of `9.2.22122.00148`. Check the database version:
  - In the Power Platform admin center select **Manage** > **Environments**, and then select the environment you want. Select **Detailed view**, and under **Version** the version is displayed.
  - By using the RetrieveVersion function with your browser. Type `https://<your org>.<your region code>.dynamics.com/api/data/v9.2/RetrieveVersion` in your browser address bar and view the JSON returned.

Install or update the Dataverse accelerator

If the Dataverse accelerator isn't already installed or if you want to install the latest version, Power Platform admins should follow these steps:

1. In the Power Platform admin center, select **Manage** > **Environments**, select the environment you want, and then select **Resources** > **Dynamics 365 apps** on the command bar. If you're using the earlier version of the admin center, expand **Resources**, and then select **Dynamics 365 apps**.
2. Find **Dataverse Accelerator** in the list.
   - If **Update available** appears next to the **Dataverse Accelerator**, select the app and then select **Update**.
   - Otherwise, select **Install app**, select **Dataverse Accelerator** in the list, and then select **Next**.
3. Follow the instructions on your screen to install the app or apply the update.

Uninstall the Dataverse accelerator

You can delete the Dataverse accelerator app from an environment using Power Platform CLI.

> **Note**
>
> Deleting the Dataverse accelerator app doesn't delete all underlying capabilities presented in the app. The app provides a modern shell to the underlying platform API.

Using Power Platform CLI, connect to the target environment and execute the three solution delete commands here:

PowerShell

```
pac solution delete --solution-name msdyn_DataverseAcceleratorApp
pac solution delete --solution-name DataverseAccelerator
pac solution delete --solution-name DataverseAccelerator_Anchor
```

Frequently asked questions (FAQs)

What is Dataverse accelerator?

Dataverse accelerator is a toolkit designed to expedite the development process within the Dataverse environment. It provides capabilities designed to help makers get more out of Dataverse by providing modern experiences for new and existing Dataverse features.

Who can use Dataverse accelerator?

Members of the system customizer security role have access to the app and underlying features. A custom security role can also be assigned to a user by the environment's system admin with similar privileges if desired.

What are the key benefits of using Dataverse accelerator?

Dataverse accelerator offers experimental and preview features that are designed to enhance or accelerate development in certain areas of Dataverse capabilities. Although in preview, accessing these features earlier in development can help you stay competitive and know about features coming in the native platform.

Is Dataverse accelerator compatible with my existing Dataverse environment?

Yes, Dataverse accelerator is compatible with existing Power Platform environments that have a Dataverse database enabled. It can be installed into your environment to enhance development capabilities without disrupting ongoing operations.

How do I install Dataverse Accelerator?

Follow the install instructions of this article to install the Dataverse accelerator, which points to the documentation for installing any Dynamics 365 app in a Dataverse environment.

If I don't want the Dataverse accelerator installed in my environment, can I delete it?

Yes, you can remove any instance of the Dataverse accelerator from an environment. To remove the Dataverse accelerator from your environment, follow the delete instructions of this article.

Can I customize Dataverse accelerator components to fit my specific requirements?

The Dataverse Accelerator app provides an experience on top of existing capabilities available in Dataverse by default. Several settings related to the experiences provided in the app are surfaced, and are based on and limited to the context of the respective features. However, don't customize any of the managed solution components.

Is there a cost associated with using Dataverse accelerator?

The features available in the Dataverse accelerator are available at no additional cost for users with appropriate licenses for Power Platform and Dataverse. However, additional costs might apply for storage capacity utilized by some of the features.

Where can I find support and resources for Dataverse accelerator?

See the section on contacting help and support for instructions on getting support either with the features in the Dataverse accelerator, or if you encounter problems using the Dataverse accelerator application.

How often is Dataverse accelerator updated?

Dataverse accelerator is regularly updated to incorporate new features, improvements, and bug fixes. Major updates are announced based on the individual features, and are indicated in the feature details, which are available in the app itself, release notes, and blog posts.

Contacting help and support

For issues with the Dataverse accelerator solution installation or with features, such as low-code plug-ins errors, use the help + support experience and include the following information:

- Problem type: **Dataverse Web API and SDK**
- Problem subtype: **Accelerator kit for Dataverse**

Limitations and known issues

The Dataverse accelerator isn't available in GCC or GCC High environments.

Last updated on 02/23/2026

Monitor and troubleshoot Dataverse plug-in runs (preview)

Summarize this article for me

[This topic is pre-release documentation and is subject to change.]

> **Important**
>
> The Dataverse accelerator, monitor plug-ins, and API playground features are deprioritized and aren't be delivered as features. For information about preview feature availability, go to Power Apps preview program.

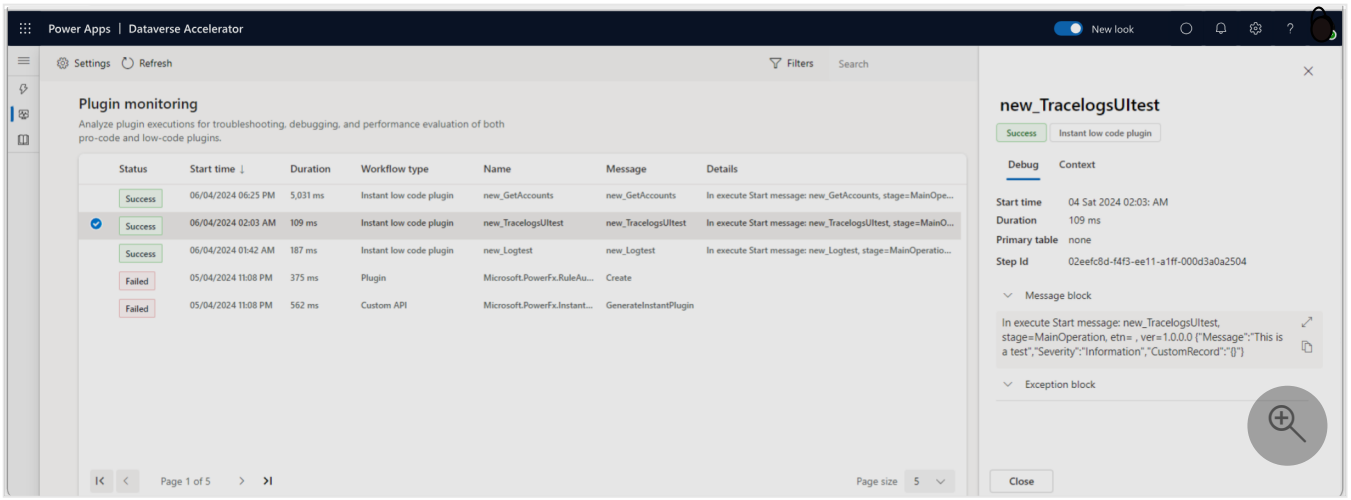
Use tracing to troubleshoot low-code and pro code plug-ins. Tracing helps makers by recording run-time information as an aid in diagnosing the cause of failures, or for general validation of certain states during development.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

The plug-in monitoring module of the Dataverse accelerator is an enhanced view of the existing tracing and logging capabilities available in Microsoft Dataverse for plug-ins. The interface helps improve the troubleshooting and debugging processes during development phase by collocating relevant details in a modern treatment.

All events surfaced in the app are read from the Trace table for 24 hours by the following workflow types:

- Plug-ins
- Custom APIs
- Instant and automated low-code plug-ins



Prerequisites

You must have the following security role assignment and access to the Dataverse accelerator app:

- Security role: Both system customizer and system administrator security roles have the necessary privileges to enable logs and use the app by default.
- Alternatively, you can create custom roles with specific access to each capability by configuring the following privileges:
  - To *read* logs, organization-level read privilege to the Trace table is required.
  - To *enable* logging in the environment, organization-level write privilege in the Organization table's `PluginTraceLogSetting` is required.

Key features

- Centralized log viewer: Access and view trace logs from Dataverse custom APIs, low-code plug-ins, and pro code plug-in executions in an environment from one central location.
- Filtering capabilities: Conveniently filter log history to quickly find logs relevant to your debugging needs.

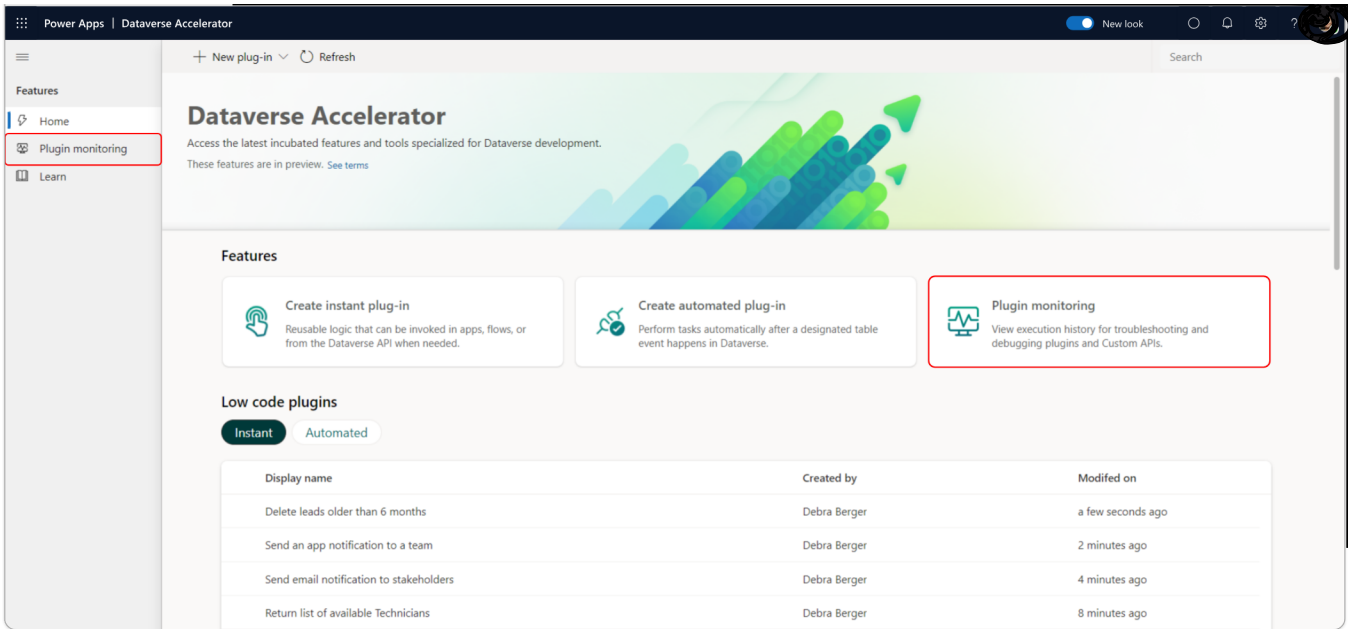
Installation

This feature is available in the Dataverse accelerator app. Go to the Dataverse accelerator article for installation instructions.

Try the plug-in monitoring feature

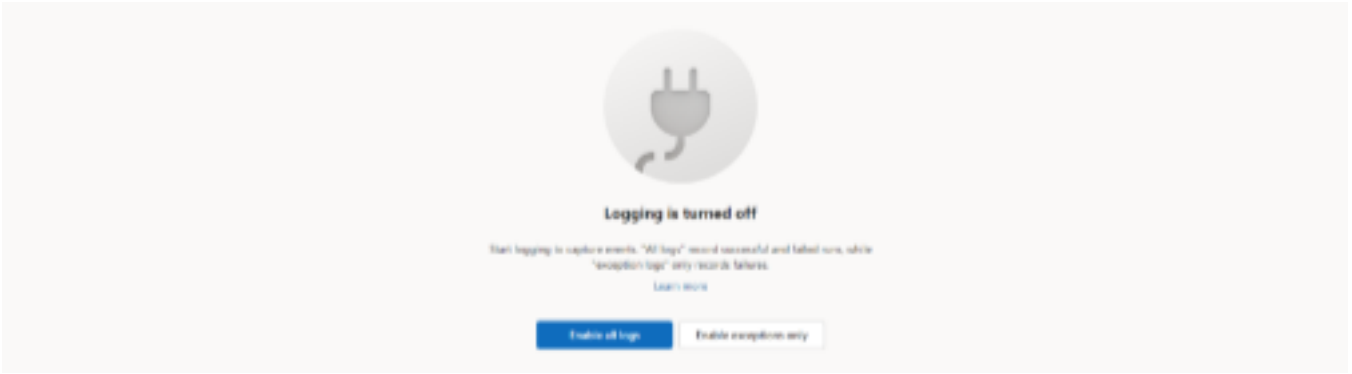
1. Play the Dataverse accelerator.

2. Go to the **Plugin monitoring** feature (on the left navigation pane or the feature card on the landing screen).

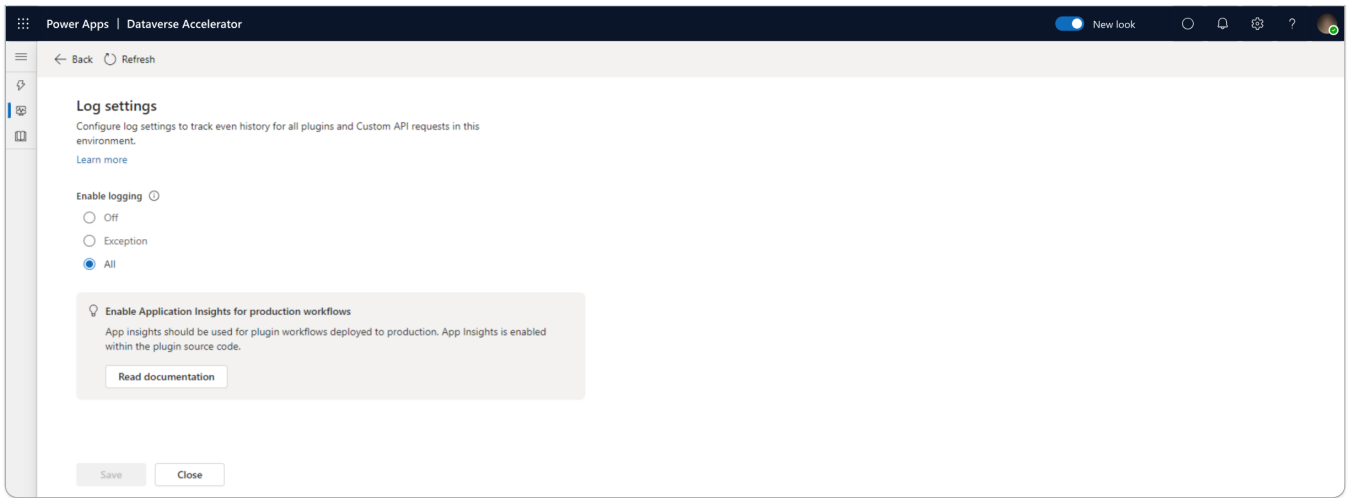


Enable log capturing

If log capturing isn't enabled in your environment, the landing page displays an off state with two options to enable logging: **All** or **Exceptions**. Choose the appropriate option based on your needs, go to the Logging and tracing article for more information.



You can change these settings later on the **Settings** screen if needed, which can be accessed on the command bar of the main screens.



> **Tip**
>
> Tracing and logging are ideal for debugging during plug-in development. For plug-in workflows in production, use Azure App Insights for monitoring.

View logs

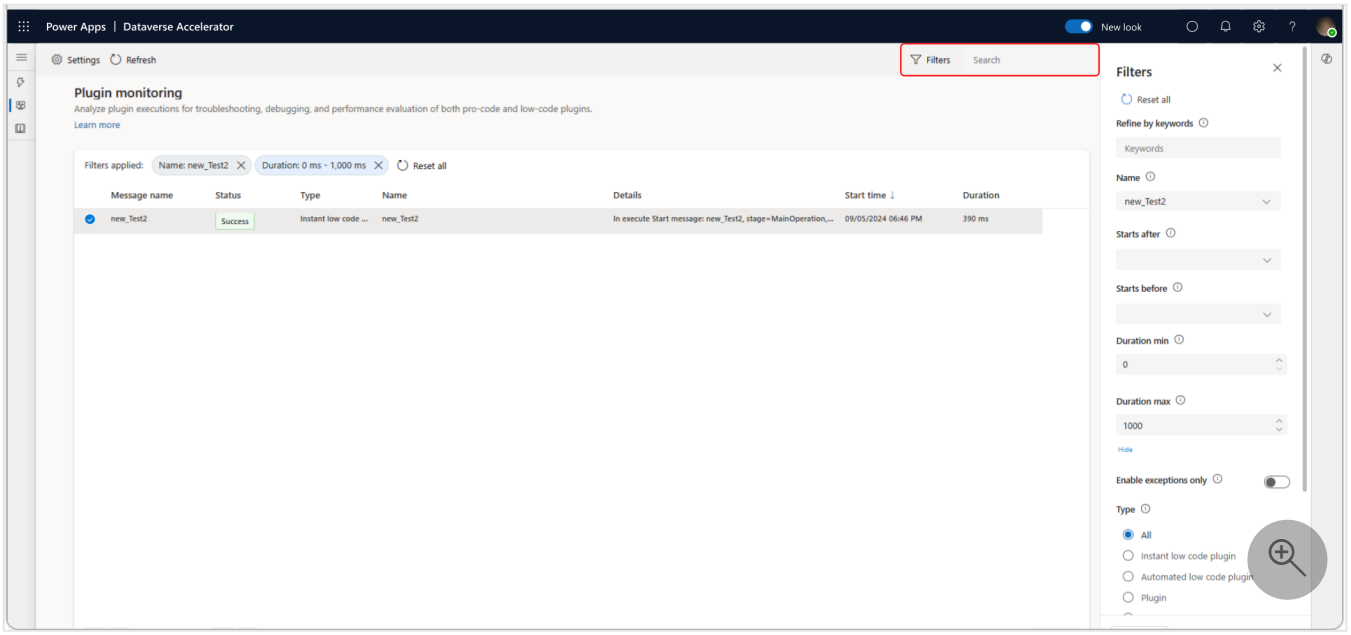
When logs are enabled, plug-in events start to show in the main list displayed on the screen.

> **Tip**
>
> If there are no logs showing, you can create and run a new low-code plug-in to start showing logs.

Filter logs

Utilize the filtering capabilities to quickly find logs relevant to the plug-ins you're working on.

Select **Filter** on the top right side of the command bar to open the filter pane. You can also quickly filter by keywords using the search bar.



Expand table

Filter	Description
Name	Select the logical name of the plug-in.
Starts after / before	The date time for when the plug-in was invoked.
Enable exception only	Shows runs with failure status.
Duration min / max	Minimum and maximum range of the plug-in duration.
Type	The classification of the plug-in workflow. Available types include: • Instant low-code plug-in • Automated low-code plug-in • Plug-ins • Custom API
Mode	Whether the plug-in is synchronous or asynchronous.

Read log details

Select a specific log entry to view detailed information about the plug-in execution. Log details include timestamp, plug-in name, execution status, output parameters, error messages, and more. Message block and exception block fields are displayed below the basic details in the pane, and are expanded if data exists in the respective columns.

Best practices

Examine log details

- Select a specific log entry to view detailed information about the plug-in execution.
- Log details might include timestamp, plug-in name, execution status, input/output parameters, error messages, and more.

Troubleshoot issues

- Use the information provided in the trace logs to troubleshoot any issues that arise during plug-in execution.
- Analyze error messages, input/output parameters, and execution details to identify the root cause of the problem.
- Use trace messages to provide helpful pointers. Writing to the trace service is available in the Dataverse SDK and in low-code plug-ins as the Trace function in Power Fx.

Take action

- Based on the insights gathered from trace logs, take appropriate action to address any performance issues or bugs identified.
- This might involve adjusting plug-in configurations, updating code logic, or seeking assistance from support resources.

Plan to test

- Make it a habit to plan test cases you can execute in trace logs to validate plug-in performance during development.
- Address any issues or anomalies promptly to maintain the overall stability and reliability of your Dataverse plug-ins.

Monitor plug-in performance

- Regularly monitor plug-in performance in production by reviewing trace logs registered with Application Insights.
- Keep an eye on execution status, error messages, and any anomalies that might indicate performance issues or bugs.

FAQs

What privileges are required to access the Trace Log Viewer?

You must have at least read privileges to the Trace table to view logs.

Who can enable log capturing within the environment?

System administrators with the appropriate security role can enable log capturing within their environment. Specifically, write access to the Organization table is needed to update the PluginTraceLogSetting column value.

Can I filter log history to find specific logs?

Yes, the trace log viewer provides filtering capabilities, allowing users to conveniently find logs relevant to their debugging needs.

Is the plug-in monitoring feature available in all environments?

The feature is delivered through the Dataverse accelerator and is available in all environments that have the app installed. The app is automatically installed in all new environments, but can also be installed in older environments by following the install instructions.

After I enabled logging, it still shows the "Logging is turned off" screen

If you have the correct privileges and are certain you have enabled logging, refresh the browser tab. You can validate whether logging is enabled using the legacy experience.

See also

Get preview features early with the Dataverse accelerator (preview)

Last updated on 02/26/2026

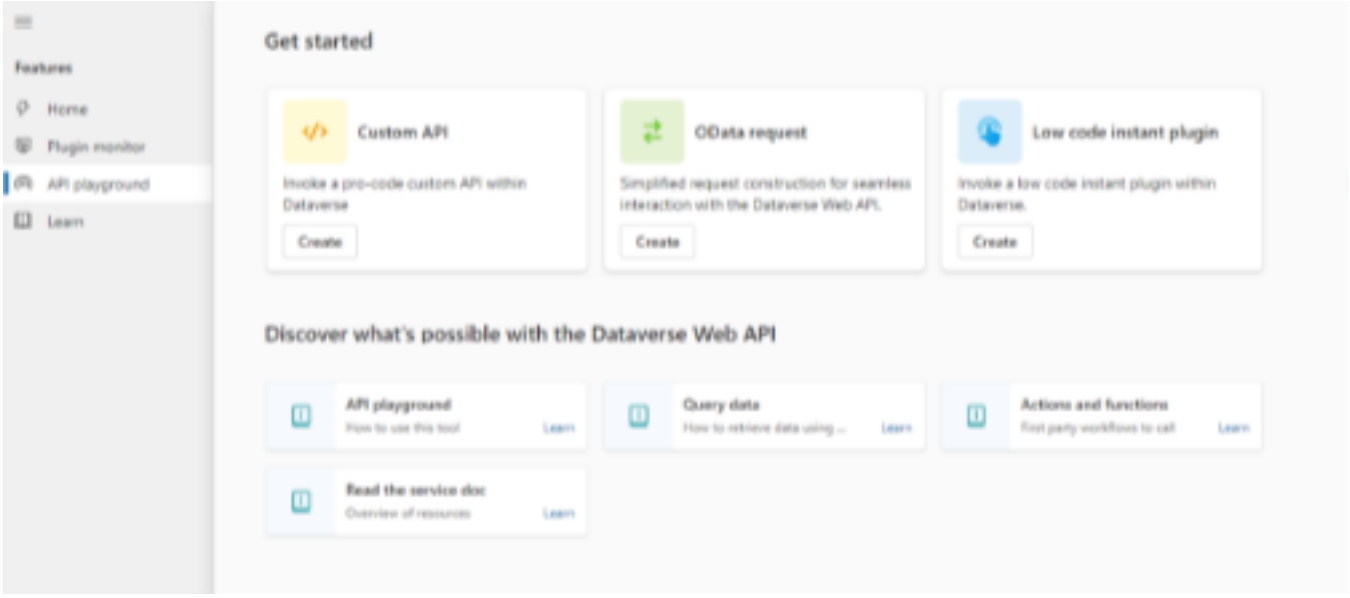
Explore and test Dataverse in the Web API playground (preview)

Summarize this article for me

[This topic is pre-release documentation and is subject to change.]

> **Important**
>
> The Dataverse accelerator, monitor plug-ins, and API playground features are deprioritized and aren't be delivered as features. For information about preview feature availability, go to Power Apps preview program.

The API playground is a preauthenticated software testing tool that helps makers quickly and conveniently interact with the Microsoft Dataverse Web API.



This web application allows developers to test and experiment with different API endpoints without the need for extensive setup or authentication processes. It also conveniently presents certain available actions and links to documentation for faster learning.

Use this tool to explore what's possible in the RESTful API, test user-defined workflows, and more.

> **Important**
>
> - This is a preview feature.
> - Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers

can get early access and provide feedback.

Key features

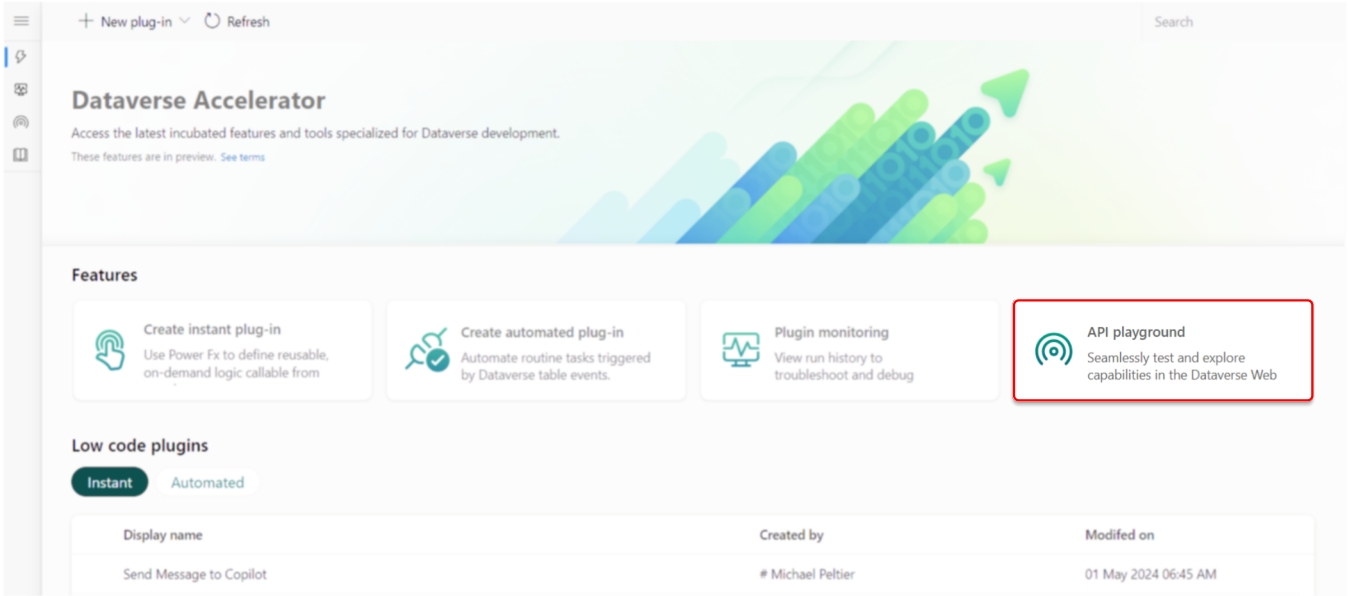
- Preauthenticated: The API playground comes preauthenticated, so developers can start testing and experimenting with API endpoints right away.
- Web-based: With the API playground, developers can quickly interact with the Dataverse Web API without the need for extensive setup like software downloads or non-Microsoft account setup. You can access the tool from the Dataverse accelerator, a Microsoft app available in all new Dataverse environments and can be installed in any environment.
- Convenient: Presents the various actions and capabilities available in the Dataverse Web API, making it easy to quickly learn what's possible and how to implement it.

Prerequisites

You must have access to Dataverse, and access to play the Dataverse accelerator app. Members of the system customizer security role have access to play the app.

Get started

To get started with the API playground, play the Dataverse accelerator, and then select the **API playground** page.



Make requests to the Dataverse Web API

On the landing screen, choose the request type to open the request editor by selecting **Create** on any of the top cards.

Three request types are presented:

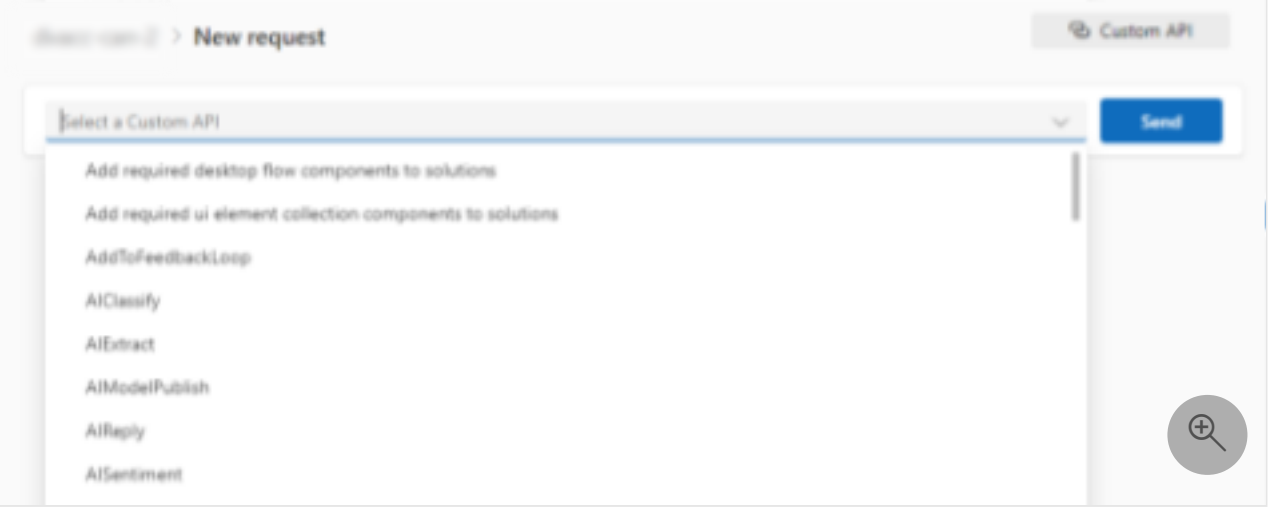
Expand table

Type	Description
Custom API	This includes any Dataverse Web API actions, functions from Microsoft, or any public user-defined custom APIs registered in the working environment.
Instant plug-in	Instant plug-ins are classified as any user-defined workflows registered as a custom API in the environment with a related FxExpression.
OData request	Allows more granular control over the request inputs to send OData requests.

Request a custom API or instant plug-in

These request types are designed to simplify calling plug-ins available in the working environment.

1. Select an option from the main dropdown list, which presents the available plug-ins in the current environment.



2. If parameters are available, they're displayed in the **Query params** table that appears. Provide values for required parameters.



> **Note**
>
> Checkboxes to the left of each parameter indicate whether they are included in the request. Required parameters have disabled checkboxes while optional parameters have editable ones.

3. Select **Send**.

4. Observe the response in the lower section of the screen.



> **Note**
>
> The interface provides a convenient listing of all available plug-ins in the environment. When selected, the associated parameters with data types are presented. Behind the scenes an OData call is still being made, but the interface simplifies the construction of the call.

Make an OData request

OData requests allow more options for request parameters.

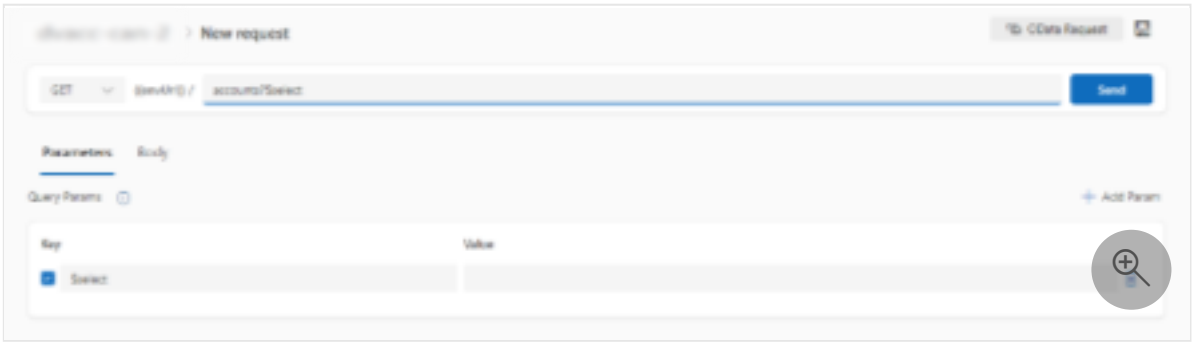
1. Select the **HTTP request method** in the first dropdown list.

2. Type in the *endpoint Url*. Only include the request URL after `[OrgUrl]/api/data/v9.2/`

3. Add *query parameters* using one of the two methods:

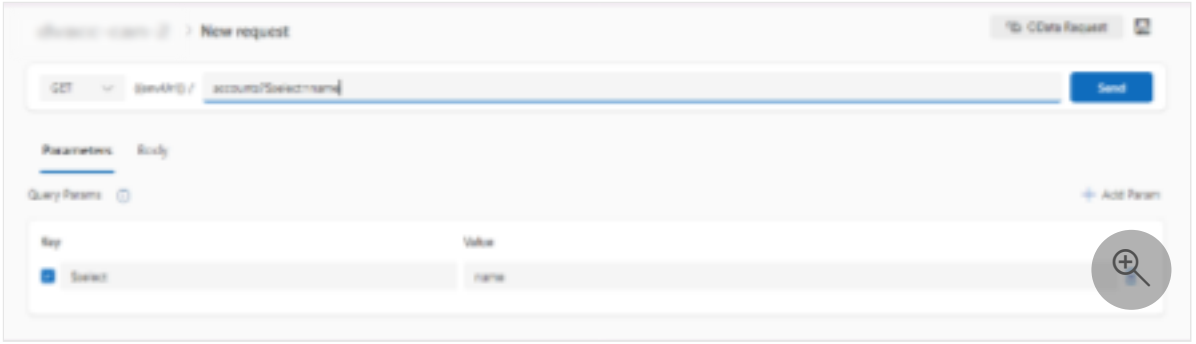
   a. In the URL.

      i. After the question mark (`?`) character in the URL, type in the query parameter key.



      ii. Type an equals (`=`) character after the key.

      iii. Type in the value.



      The keys and values will automatically populate the parameter table in the interface.

   b. Manually add query parameters by selecting **+ Add param**. This adds an empty parameter row to the table that you can type values into. The URL is updated dynamically as values are entered.

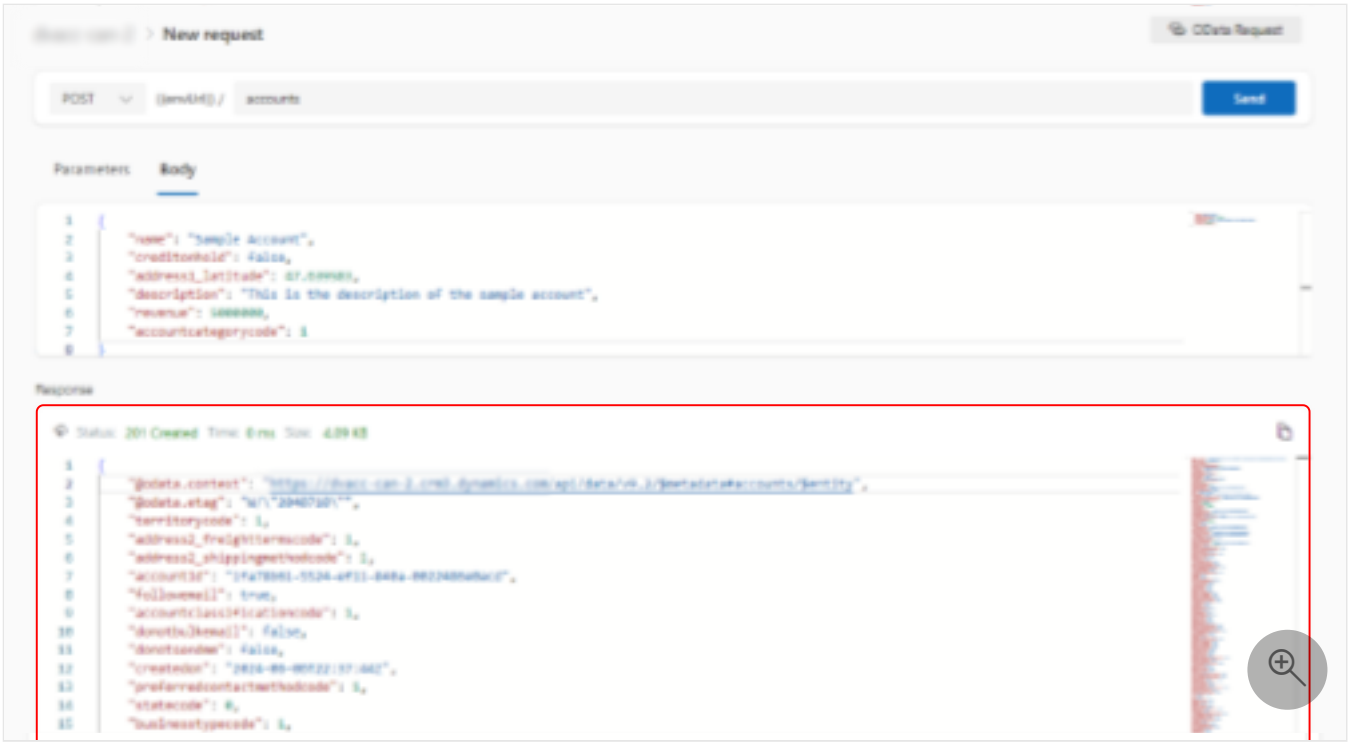
4. Optionally provide a body (if using all request methods except GET) by selecting the **Body** tab, and then enter the body value in the editor control.

5. Select **Send**.

6. Observe the response in the lower section.

View the response

After a request is sent, the response is displayed in the lower half of the screen.



The response status provides quick metrics on the request:

- Status message
- Duration (in milliseconds)
- Size (in kilobytes)

The response body is displayed below the response status. Select the copy icon to copy the response body to your clipboard.

FAQs

What privileges are required to access the API playground tool?

You must have access to the Dataverse accelerator app to use the tool.

Does the API playground tool only connect to the Dataverse Web API?

Yes, the API playground tool only connects to the Dataverse Web API. No other services are available to connect to in this interface.

How does the API playground tool authenticate to the Dataverse Web API?

The tool uses the authentication token of the logged in user that is required to play the Dataverse accelerator application.

Is the API playground feature available in all environments?

The feature is delivered through the Dataverse accelerator and is available in all environments that have the app installed. The app is automatically installed in all new environments, but can also be installed in older environments by following the install instructions.

See also

Get preview features early with the Dataverse accelerator (preview)

Last updated on 02/26/2026

Security concepts in Microsoft Dataverse

06/03/2025

One of the key features of Dataverse is its rich security model that can adapt to many business usage scenarios. This security model is only in play when there's a Dataverse database in the environment. As an administrator, you likely won't be building the entire security model yourself, but will often be involved in the process of managing users and making sure they have the proper configuration and troubleshooting security access related issues.

> **Tip**
>
> Check out the following video: Microsoft Dataverse – Security Concepts Shown In Demos.

Role-based security

Dataverse uses role-based security to group together a collection of privileges. These security roles can be associated directly to users, or they can be associated with Dataverse teams and business units. Users can then be associated with the team, and therefore all users associated with the team benefit from the role. A key concept of Dataverse security to understand is all privilege grants are accumulative with the greatest amount of access prevailing. If you gave broad organization level read access to all contact records, you can't go back and hide a single record.

Business units

> **Tip**
>
> Check out the following video: Modernize business units.

Business units work with security roles to determine the effective security that a user has. Business units are a security modeling building block that helps in managing users and the data they can access. Business units define a security boundary. Every Dataverse database has a single root business unit.

You can create child business units to help further segment your users and data. Every user assigned to an environment belongs to a business unit. While business units could be used to

model 1:1 a true organization hierarchy, more often they lean more towards just defined security boundaries to help achieve the security model needs.

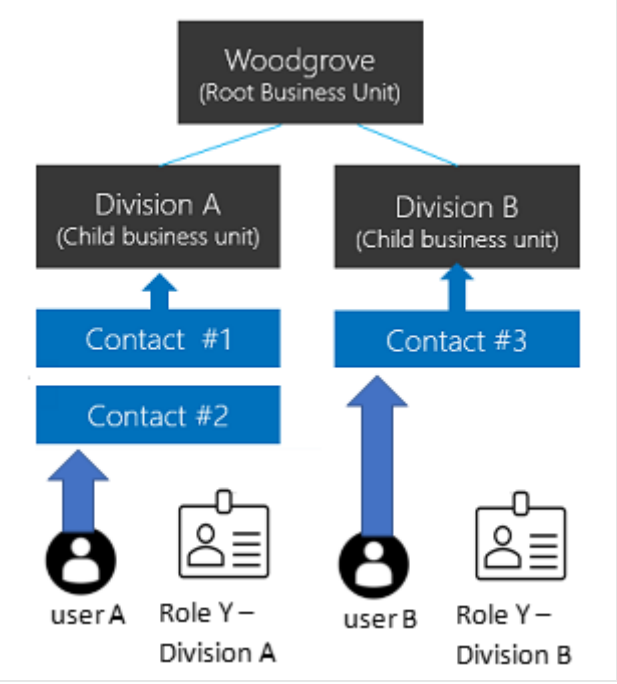
To better understand, let's look at the following example. We have three business units. Woodgrove is the root business unit and will always be at the top, that is unchangeable. We've created two other child business units A and B. Users in these business units have different access needs. When we associate a user with this environment, we can set the user to be in one of these three business units. Where the user is associated determines which business unit owns the records that user is the owner of. Having that association allows us to tailor a security role to allow the user to see all records in that business unit.

Hierarchical data access structure

Customers can use an organization structure where data and user are compartmentalized in a tree-like hierarchy.

When we associate a user with this environment, we can set the user to be in one of these three business units and assign a security role from the business unit to the user. The business unit the user is associated with determines which business unit owns the records when the user creates a record. By having that association it allows us to tailor a security role, which allows the user to see records in that business unit.

User A is associated with Division A and assigned a security role Y from Division A. This allows user A to access the Contact #1 and Contact #2 records. While user B in Division B can't access Division A's Contact records but can access Contact #3 record.

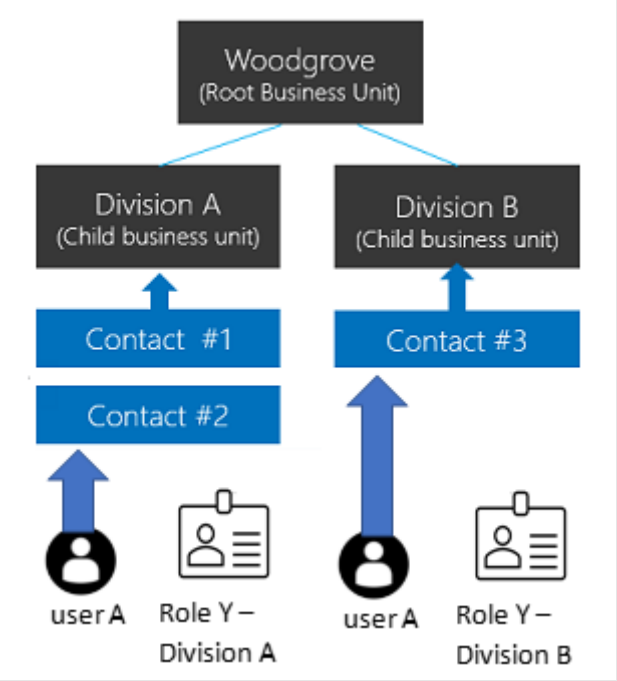


Matrix data access structure (Modernized Business Units)

Customers can use an organization structure where data is compartmentalized in a tree-like hierarchy, and users can work and access any business unit's data regardless of what the business unit the user is assigned to.

When we associate a user with this environment, we can set the user to be in one of these three business units. For each business unit that a user needs to access data, a security role from that business unit is assigned to the user. When the user creates a record, the user can set the business unit to own the record.

User A can be associated with any of the business units, including the root business unit. A security role Y from Division A is assigned to user A which gives the user access to Contact #1 and Contact #2 records. A security role Y from Division B is assigned to user A which gives the user access to Contact #3 record.



Enable the Matrix data access structure

> **Note**
>
> Before you enable this feature, you must publish all your customizations to enable all your new unpublished tables for the feature. If you find that you have unpublished tables that aren't working with this feature after you have turned it on, you can set the **RecomputeOwnershipAcrossBusinessUnits** setting using the OrgDBOrgSettings tool for Microsoft Dynamics CRM. Setting **RecomputeOwnershipAcrossBusinessUnits** to true allows the Owning Business Unit field to be set and updated.

New admin center

1. Sign in to the Power Platform admin center as an admin (Dynamics 365 admin or Microsoft Power Platform admin).
2. In the navigation pane, select **Manage**.
3. In the **Manage** pane, select **Environments**, and then choose the environment that you want to enable this feature for.
4. Select **Settings** > **Product** > **Features**.
5. Turn **On** the **Record ownership across business units** switch.
6. Select **Save**.

After this feature switch is turned on, you can select Business unit when you assign a security role to a user. This allows you to assign security role from different business units to a user. The user also requires a security role from the business unit that the user is assigned to with user settings privileges to run model-driven apps. You can refer to the Basic User security role to find how these user settings privileges are enabled.

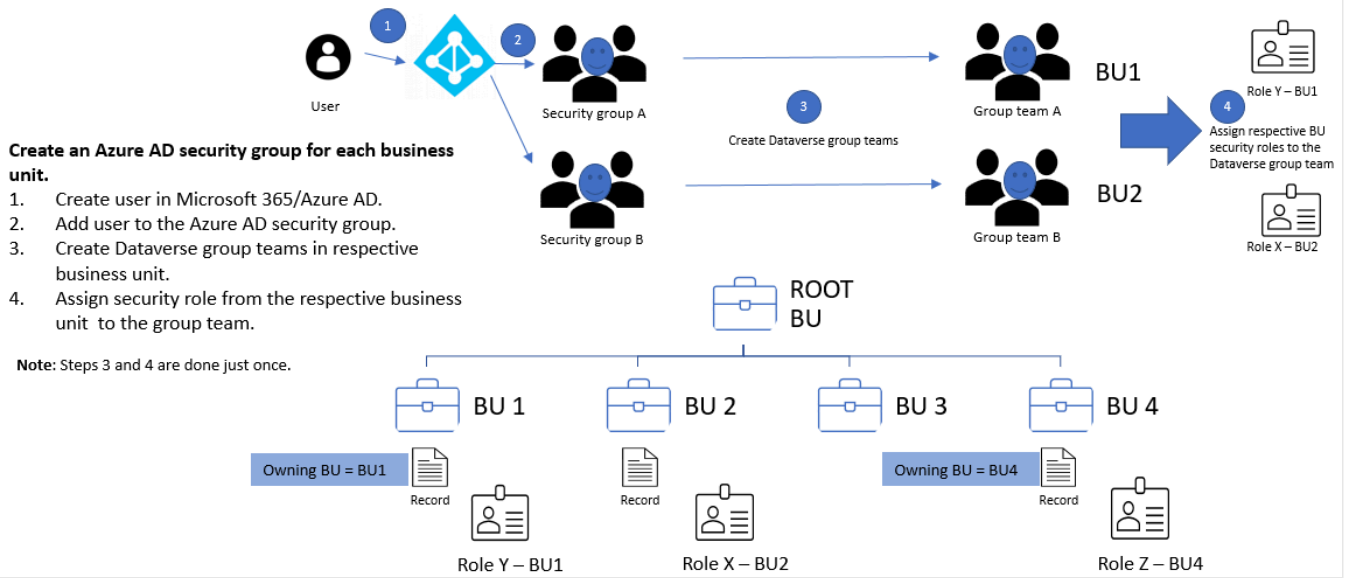
You can assign a user as record owner in any business unit without the need to assign a security role in the record's owning business unit as long as the user has a security role that has Read privilege to the record table. See Record Ownership in Modernized Business Units.

> **Note**
>
> This feature switch is stored in the **EnableOwnershipAcrossBusinessUnits** setting and can be set using the OrgDBOrgSettings tool for Microsoft Dynamics CRM.

Associate a business unit with a Microsoft Entra security group

You can use a Microsoft Entra security group to map your business unit for streamlining your user administration and role assignment.

Create a Microsoft Entra security group for each business unit and assign the respective business unit security role to each group team.



For each business unit, create a Microsoft Entra security group. Create a Dataverse group team for each Microsoft Entra security group. Assign the respective security role from the business unit to each Dataverse group team. The user in the above diagram will be created in the root business unit when the user accesses the environment. It's fine to have the user and the Dataverse group teams to be in the root business unit. They only have access to data in the business unit where the security role is assigned.

Add users into the respective Microsoft Entra security group to grant them access to the business unit. The users can immediately run the app and access its resources/data.

In the matrix data access, where users can work and access data from multiple business units, add the users to the Microsoft Entra security groups that mapped to those business units.

Owning Business Unit

Each record has an **Owning Business Unit** column, which determines which business unit owns the record. This column defaults to the user's business unit when the record is created and can't be changed except when the feature switch is turned ON.

> **Note**
>
> When you change which business unit owns a record, be sure to check out the following for cascade effects: Using SDK for .NET to configure cascading behavior.

You can manage whether you want to allow your user to set the Owning Business Unit column when the feature switch is ON. To set the Owning Business Unit column, you need to grant the user's security role the Business Unit table's **Append To** privilege with local level permission.

To allow your user to set this column, you can enable this column in the following:

1. Form - both the body and header.
2. View.
3. Column mappings. If you're using the AutoMapEntity, you can specify the column in your column mapping.

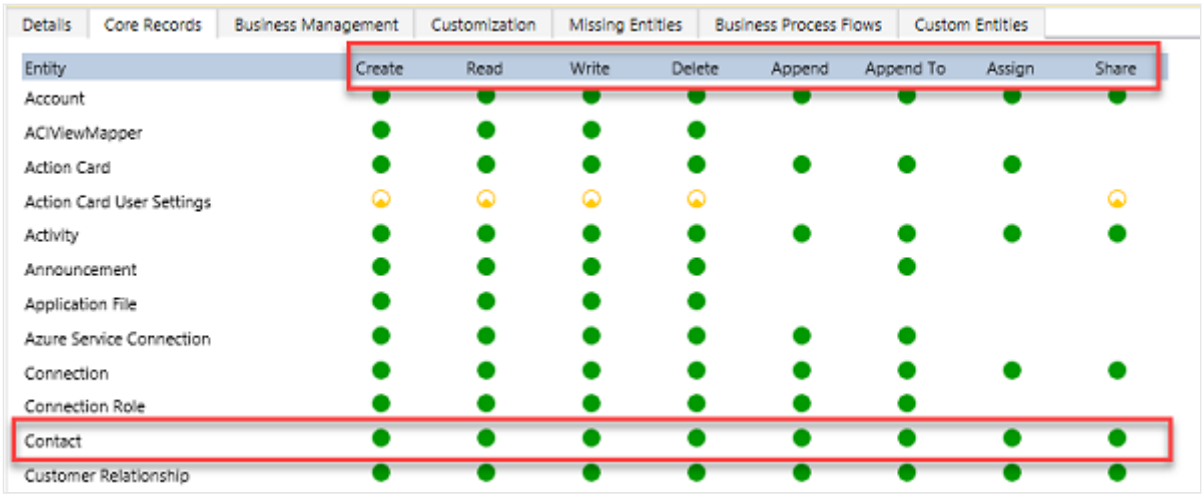
> **Note**
>
> If you have a job/process to sync data between environments and the **Owning Business Unit** is included as part of the schema, your job fails with a **Foreign KEY** constraint violation if the target environment doesn't have the same **Owning Business Unit** value.
>
> You can either remove the **Owning Business Unit** column from the source schema, or update the **Owning Business Unit** column value of the Source to any of the business units of the target.
>
> If you have a job/process to copy data from an environment to an external resource, for example Power BI, you'll need to select or deselect the **Owning Business Unit** column from your source. Select it if your resource can receive it otherwise deselect it.

Table/record ownership

Dataverse supports two types of record ownership. Organization owned, and User or Team owned. This is a choice that happens at the time the table is created and can't be changed. For security purposes, records that are organization owned, the only access level choices is either the user can do the operation or can't. For user and team owned records, the access level choices for most privileges are tiered Organization, Business Unit, Business Unit and Child Business Unit or only the user's own records. That means for read privilege on contact, I could set user owned, and the user would only see their own records.

To give another example, let's say User A is associated with Division A, and we give them Business Unit level Read access on Contact. They'd be able to see Contact #1 and #2 but not Contact #3.

When you configure or edit security role privileges, you're setting the access level for each option. The following is an example of the Security Role privilege editor.



In the above you can see the standard privilege types for each table Create, Read, Write, Delete, Append, Append To, Assign and Share. You can edit each of these individually. The visual display of each will match the key below as to what level of access you've granted.



In the above example, we have given organization level access to Contact which means that the user in Division A could see and update contacts owned by anyone. In fact, one of the most common administrative mistakes is getting frustrated with permissions and just over granting access. Very quickly a well-crafted security model starts looking like Swiss cheese (full of holes!).

Record Ownership in Modernized Business Units

In **Modernized Business units**, you can have users being owners of records across any business units. All the users need is a security role (any business unit) which has Read privilege to the record table. The users don't need to have a security role assigned in each business unit where the record resides.

If **Record ownership across business units** was enabled in your production environment during the Preview period, you need to perform the following to enable this record ownership across business unit:

1. Install the Organization Settings editor
2. Set the **RecomputeOwnershipAcrossBusinessUnits** organization settings to true. When this setting is set to true, the system is locked and can take up to 5 minutes to do the recomputation to enable the capability where users can now own records across business units without the need to have separate security role assigned from each business unit. This allows an owner of a record to assign their record to someone outside of the record's owning business unit.

3. Set **AlwaysMoveRecordToOwnerBusinessUnit** to false. This makes the record remain in the original owning business unit when the record ownership is changed.

For all non-production environments, you just need to set **AlwaysMoveRecordToOwnerBusinessUnit** to false to use this capability.

> ⓘ **Note**
>
> If you turn off either the **Record ownership across business units** feature or set the **RecomputeOwnershipAcrossBusinessUnits** setting to false using the OrgDBOrgSettings tool for Microsoft Dynamics CRM, you won't be able to set or update the Owning Business unit field, and all records where the Owning Business Unit field is different from the owner's business unit will be updated to the owner's business unit.

Teams (including group teams)

Teams are another important security building block. Teams are owned by a Business Unit. Every Business Unit has one default team that is automatically created when the Business Unit is created. The default team members are managed by Dataverse and always contain all users associated with that Business Unit. You can't manually add or remove members from the default team, they're dynamically adjusted by the system as new users are associated/disassociated with business units. There are two types of teams, owning teams and access teams.

- Owning Teams can own records, which give any team member direct access to that record. Users can be members of multiple teams. This allows it to be a powerful way of granting permissions to users in a broad way without micromanaging access at the individual user level.
- Access teams are discussed in the next section as part of record sharing.

Record sharing

Individual records can be shared on a one-by-one basis with another user. This is a powerful way of handling exceptions that don't fall into the record ownership or is a member of a business unit access model. It should be an exception, though, because it's a less performant way of controlling access. Sharing is tougher to troubleshoot because it's not a consistently implemented access control. Sharing can be done at both the user and team level. Sharing with a team is a more efficient way of sharing. A more advanced concept of sharing is with Access Teams, which provides auto-creation of a team and sharing of record access with the team is based on an Access Team Template (template of permissions) which is applied. Access teams

can also be used without the templates, with manually adding or removing its members. Access teams are more performant because they don't allow owning records by the team or having security roles assigned to the team. Users get access because the record is shared with the team and the user is a member.

Record-level security in Dataverse

You might be wondering – what determines access to a record? That sounds like a simple question but for any given user it's the combination of all their security roles, the business unit they're associated with, the teams they're members of and the records that are shared with them. The key thing to remember is all access is accumulative across all those concepts in the scope of a Dataverse database environment. These entitlements are only granted within a single database and are individually tracked in each Dataverse database. This all requires they have an appropriate license to access Dataverse.

Column-level security in Dataverse

Sometimes record-level control of access isn't adequate for some business scenarios. Dataverse has a column-level security feature to allow more granular control of security at the column level. Column-level security can be enabled on all custom columns and most system columns. Most system columns that include personal identifiable information (PII) are capable of being individually secured. Each column's metadata defines if that is an available option for the system column.

Column-level security is enabled on a column by column basis. Access is then managed by creating a Column Security Profile. The profile contains all column that have column-level security enabled and the access granted by that specific profile. Each column can be controlled within the profile for Create, Update and Read access. Column Security Profiles are then associated with a user or Teams to grant those privileges to the users to the records they already have access to. It's important to note that column-level security has nothing to do with record-level security. A user must already have access to the record for the Column Security Profile to grant them any access to the columns. Column-level security should be used as needed and not excessively as it can add overhead that is detrimental if over used.

Managing security across multiple environments

Security roles and Column Security Profiles can be packaged up and moved from one environment to the next using Dataverse solutions. Business Units and Teams must be created and managed in each environment along with the assignment of users to the necessary security components.

Configuring users environment security

Once roles, teams, and business units are created in an environment, it's time to assign the users their security configurations. First, when you create a user, you associate the user with a business unit. By default, this is the root business unit in the organization. They're also added to the default team of that business unit.

In addition, you would assign any security roles that user needs. You would also add them as members of any teams. Remember teams can also have security roles, so the effective rights of the user is the combination of directly assigned security roles combined with those of any teams they're members of. Security is always additive offering the least restrictive permission of any of their entitlements. The following is a good walkthrough of configuring environment security.

If you have used column-level security, you would need to associate the user or a team of the user to one of the Column Securities Profiles you created.

Security is a complex article and is best accomplished as a joint effort between the application makers and the team administering the users' permissions. Any major changes should be coordinated well in advance of deploying the changes into the environment.

See also

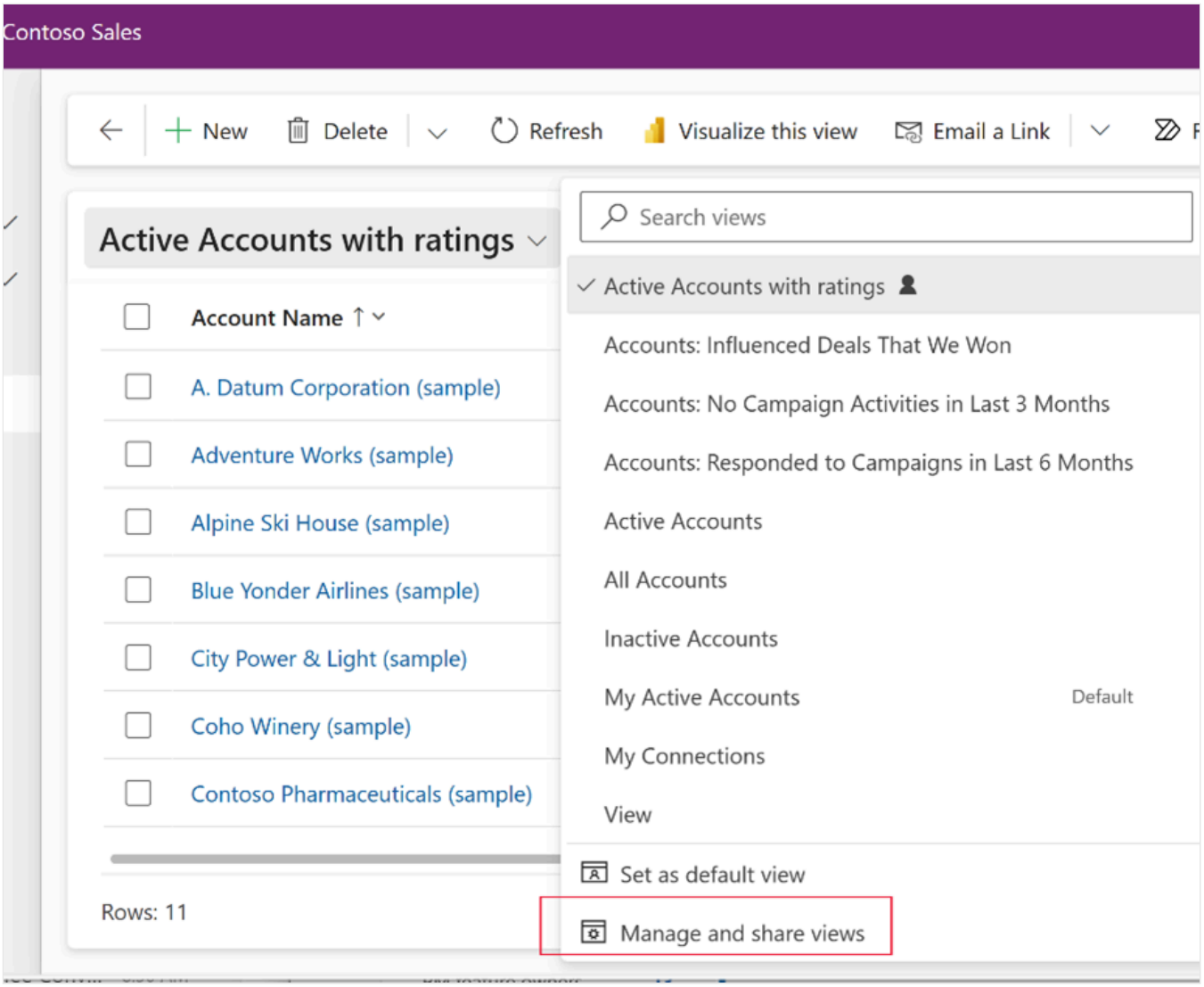
Configure environment security
Security roles and privileges

Manage access to public system views

Admins can manage the views that users can access in model-driven apps by using security roles. When a user plays a model-driven app, they only have access to the system views that apply to the security roles they're assigned.

System views are special views that model-driven apps depend on, which exist for system tables or are automatically created when you create custom tables. These views have specific purposes and some additional capabilities. By default, all system views are for **everyone**. When a Power Platform admin manages a view with a security role, only the users who are assigned with the selected security role are able to see the view in the view selector. The other system views aren't filtered from the view selector dropdown list. Users can still access all the views when they navigate to the **Manage and share views** option.

After a Power Platform admin turns on the manage table list views feature, users can set their own default view from the list of views that the admin manages and their own personal views from the **Manage and share views** option in a model-driven app.



Data access continues to be secured with security role privileges, which means only records that a user has access to are displayed by the views. Having access to a particular view doesn't mean you also have access to the data that's available in the view.

> **Note**
>
> This feature filters views that appear in the table list view selector. All system and associated views continue to display in subgrids and associated grids that have a view selector.

Prerequisites

- Confirm that the orgdbsettings `EnableRoleBasedSystemViews' property is set to **true** using the OrganizationSettingsEditor tool.
- System administrator security role membership in the Microsoft Dataverse environment.
- Turning on auditing is recommended, but not required.

System views in a Dataverse environment

System views are predefined views that exist for all system tables. These views can include Public, Quick Find, Advanced Find, Associated, or Lookup. When an admin or maker creates a view from Power Apps (make.powerapps.com), the view becomes a public view. You can manage public views with security roles.

Commonly used public view examples

You can manage the table list of views with your business users based on the users' security role assignment. Admins select the applicable system views and manage these views with security roles. When the user navigates to a table list form, they can access the system views that were managed with the security roles that are assigned to them. The selected system views with security role aren't visible to users who don't have the security role assigned.

Examples of public views:

Expand table

Table	Public view
Account	My Active Accounts
Account	Accounts: No Campaign Activities in Last 3 months
Lead	My Open Leads
Lead	All Leads
Case	All Cases
Case	Cases I follow
Case	My Cases
Case	My Resolved cases
Contact	All Contacts
Contact	Contact I follow
Contact	Inactive contacts

Turn on manage table list views

1. Make sure you enable the table list views feature in your environment. More information: Prerequisites
2. You manage public views with security roles. If you need to create new security roles, go to Security roles and privileges for more information about security roles.
3. Create a solution and add the tables and their corresponding views that you manage.
4. Export the solution.
5. Import the solution into your production environment.

> **Tip**
>
> Inform your users that their system views are now filtered based on their security role assignment.

Manage public views with security roles

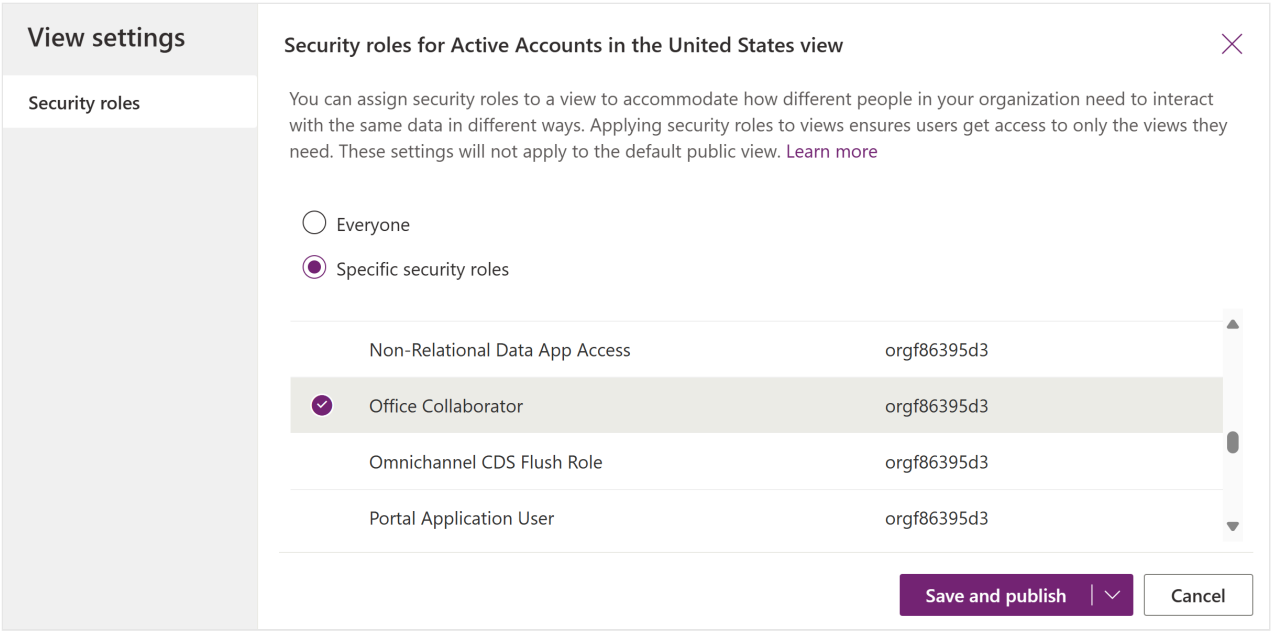
All system views can be managed with security roles. Once they're managed, users with the assigned security roles only view these views in the view selector.

> **Note**
>
> System views are denoted as **Public** view type in Power Apps (make.powerapps.com). Only public views can be managed. Default public views can't be assigned security roles for access.

Manage system views with security roles

1. Sign in to Power Apps.
2. On the left navigation pane, select **Solutions**, and then open the solution that has the table with the view you want to manage. If the item isn't in the side panel, select **...More** and then select the item you want.
3. Open a table, such as the account table, and then select the **Views** area.
4. Select the nondefault public view that you want, and then on the command bar, select **View settings**.
5. Select the **Specific security roles** option.
6. Select the security roles that you want to assign to the public view.

7. When you're done selecting the security roles, select **Save and Publish**.



> **Note**
>
> When you make changes to the 'EnableRoleBasedSystemViews' and **View settings**:
>
> 1. Setting the OrganizationSettingsEditor tool `EnableRoleBasedSystemViews` property to **true** is effective immediately.
> 2. Setting a view with security roles is effective immediately after you select **Save and publish**.
> 3. Changing a view setting from 'Specify security role' to 'Everyone' can take up to 24 hours to be effective or until the user signs out and back in.
>
> If you select multiple views and go to the view settings, only the first selected Views can be updated. You're required to select individual view to update the view settings.

> **Important**
>
> The security roles as listed under **View settings** are from the *root business unit*. Business unit level security roles are inherited from the root business unit. When you select the security roles from **View settings**, the system automatically applies filtering to users who are assigned the corresponding business unit level security roles.

Related articles

Create or edit a model-driven app view in Power Apps

ⓘ **Note:** The author created this article with assistance from AI. Learn more

---

Last updated on 04/06/2026

Microsoft Dataverse Developer Guide

Summarize this article for me

Microsoft Dataverse is the underlying data platform for Power Apps, providing core functionality such as server-side logic (plug-ins and workflows), business process flows, a sophisticated security model, and an extensible platform for developers. This guide helps developers understand how to extend Dataverse with code to build custom line-of-business apps for users, businesses, independent software vendors (ISVs), and systems integrators (SIs).

There are many ways developers can contribute to creating apps that use Dataverse. While you can build an application with code using Dataverse as your data source, most projects use either model-driven apps or canvas apps to generate the experience that people use.

Working with model-driven apps

Model-driven apps are built on Dataverse, and can only connect to a Dataverse environment. Dataverse stores all the data that defines a model-driven app.

Model-driven apps use Solutions to share customizations and extensions.

Model-driven apps also have many points where developers can write code to extend them. For information about what developers can do with model-driven apps, see Model-driven apps Developer Guide.

Some examples of model-driven apps available from Microsoft are Dynamics 365 Customer Service, Dynamics 365 Field Service, and Dynamics 365 Marketing.

Understand when to write code

Because Dataverse includes many capabilities for people to configure custom business logic without writing code, the most common scenarios for developers involve filling gaps where existing features don't provide the functionality you need. Fortunately, Dataverse provides many points for developers to extend the common functionality by using code.

As a developer, it's important to understand what you can do without writing code. Familiarize yourself with these capabilities. For more information, see What is Dataverse?

Content for on-premises deployments

Dataverse isn't available for on-premises deployments. Content in this guide doesn't include information about options that are only available for on-premises or internet facing

deployments (IFD). For information related to these options, see the Developer Guide for Dynamics 365 Customer Engagement (on-premises).

Get started

See also

Power Apps for developers
Model-driven apps Developer Guide

Last updated on 03/10/2026

Dataverse community tools

Article • 02/15/2022

View Layout Replicator and View Designer are a couple of the many tools available in a Power Platform community developed app named XrmToolBox. XrmToolBox is a Windows application that connects to Microsoft Dataverse.

> **Note**
>
> These tools are provided by XrmToolBox and are not supported by Microsoft. If you have questions pertaining to the tool, please contact the publisher. More information: **XrmToolBox**.

For more information about XrmToolBox and other Dataverse community tools, go to Developer tools.

Live monitor overview

Article • 11/19/2024

Live monitor is a tool that offers makers the ability to view a stream of events from a user's session to diagnose and troubleshoot problems. Makers of canvas apps can use Live monitor either to view events while building a new app in Power Apps Studio or to monitor published apps during runtime. Makers of model-driven apps can monitor page navigation, command executions, form-related issues, and other major actions to understand app behavior and make improvements.

Benefits

Live monitor can help you diagnose and troubleshoot problems faster, and build more reliable apps. It provides a deep view of your app by logging all the key activities that occur in the app as it runs. Live monitor also provides a better understanding of how the events and formulas contained in your app work, so you can improve performance and identify any errors or problems.

Debugging an app

The key to debugging a problem is to have a better understanding of what your app does, and how it does it. Sometimes it's difficult to isolate a problem by just looking at the app formulas, or even by reviewing runtime errors. Watching the events as they occur in your app can help you understand the order of events and how your app is performing, to spot errors and diagnose problems faster.

Some examples of problems and queries that you can uncover when using Live monitor are:

- **A high number of network calls**
  - Is the app fetching data too frequently?
  - Can you optimize the app to only fetch data when required?
  - Are timer controls firing too often?
  - Are too many events occurring when the app starts? Can fetching some of the data be delayed?

- **Retrieve data from the same data source**
  - Can you use different patterns such as caching data in collections or variables instead of fetching the same data multiple times?

- **The response data size**
  - Can you use query filters to reduce the amount of requested data?
- **The duration of the request**
  - Are connectors and plug-ins optimized?
  - Can you reduce the size of the response by using query filters?
- **Errors**
  - Have you configured the required permissions to run the app correctly?
  - Are your requests throttled by the platform?

Prerequisites

You'll need the following security role membership in the environment to use Live monitor.

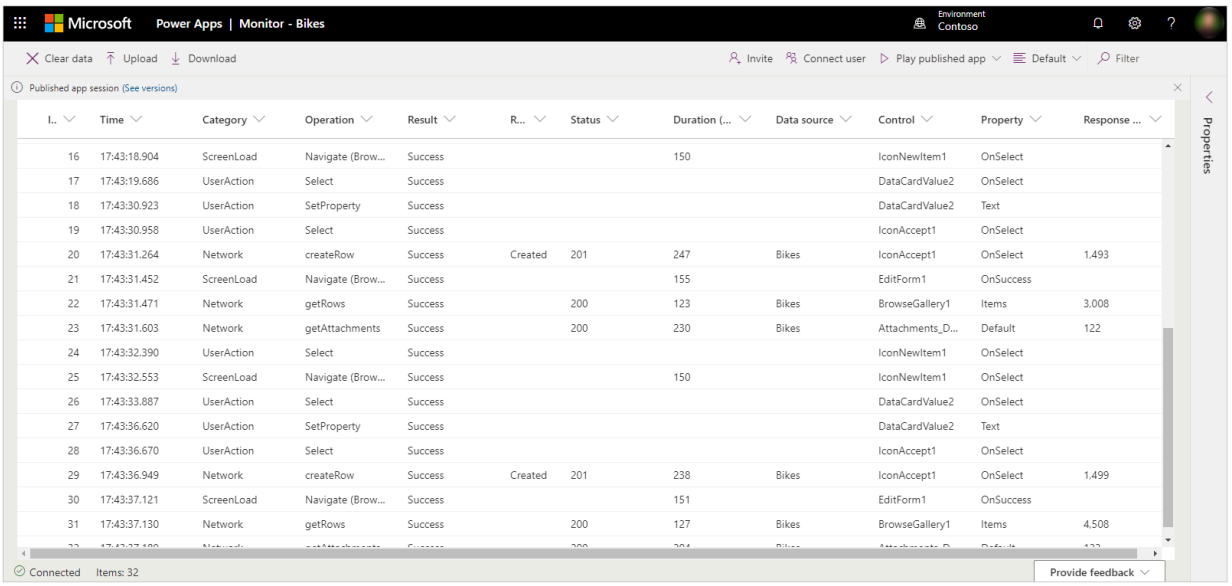
Expand table

App type	Required security role
Canvas app	Environment Admin or Environment Maker
Model-driven app	System Administrator or System Customizer for the Dataverse organization

More information: Configure user security to resources in an environment

Live monitor dashboard

You can review various properties for each event inside Live monitor. Depending on the event category, some of these properties might not contain data. The following image shows an example of the Live monitor dashboard for canvas apps.



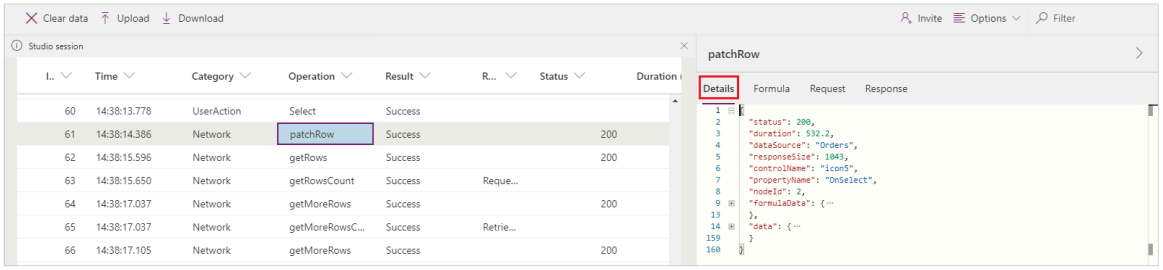
Expand table

Column name	Description
Id	The sequence number for the events.
Time	The time the event occurred.
Category	The type of event, such as *Network*.
Operation	The resulting internal operation name of the request inside the app. For example, *createRow* is the operation name from the **Patch** function.
Result	The text description of the status code. For example, a 429 status will appear as an "*Error*" in the result column. Row colors also help to identify any errors and warnings quickly.
Result Info	The detailed translation of error codes and results. For example, a 429 status code will appear as "Too many requests" in this column.
Status	The HTTP status code of a network request. For example, a 2XX code represents a successful request, whereas a 4XX code represents an error.
Duration	Duration can mostly be used to understand the performance of network calls in the app, but it also depends on the type of event. For example, for a network request, duration is the time taken for the request to be sent and a response to be received.
Data Source	The name of the data source accessed by the raised event operation (for example, the name of a Microsoft Dataverse table).
Control	The control name associated with this event
Property	The active control property of the raised event.

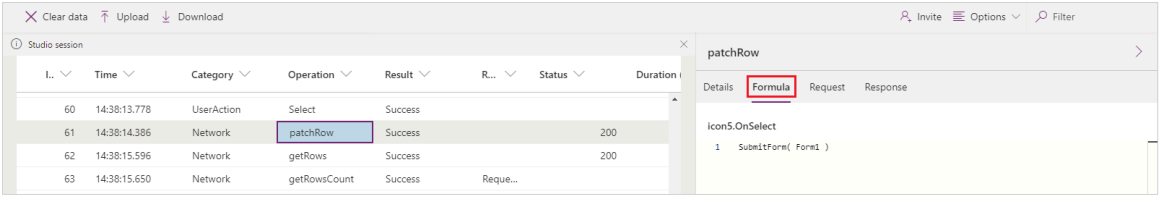
Column name	Description
Response size	For a network request event, represents the size of the response received from the sender to your app, in bytes.

When you select an event in the grid, a panel displays additional details about the event. The panel has four tabs:

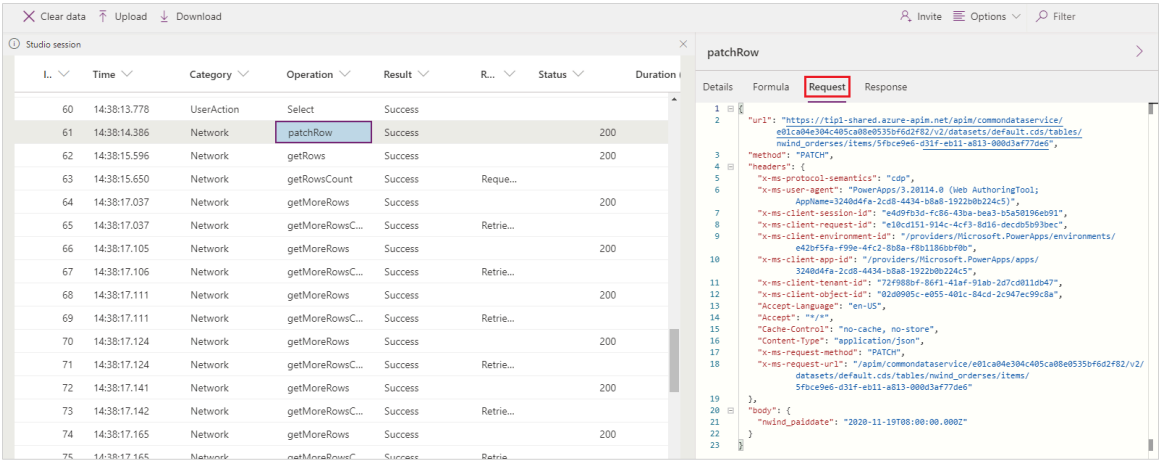
- **Details**: Shows a high-level overview of the event that you select. Some of the data might be collapsed in the tree view. You can expand and drill down to view content.



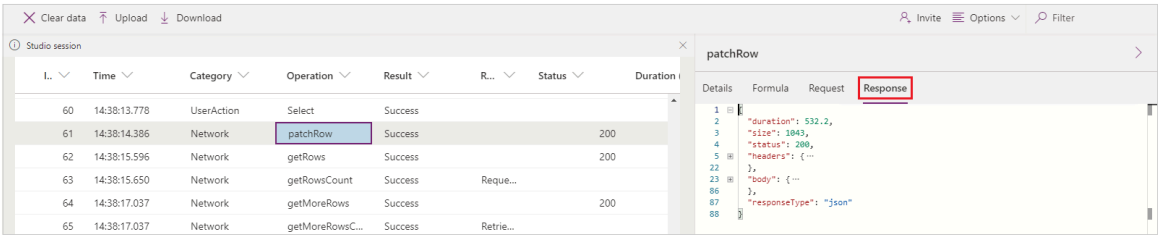
- **Formula**: Shows the related formula from your app for the selected event. The name of the control property triggering the event is displayed on top of the tab and inside the event table.



- **Request**: Shows the HTTP request that was sent.



- **Response**: Shows the HTTP response that was received. You can view the response in JSON format.



Unsupported scenarios for Live monitor

To learn about unsupported scenarios for Live monitor, go to Advanced monitoring - unsupported scenarios.

Next steps

- Monitor canvas apps using Monitor
- Monitor model-driven apps using Monitor

See also

Troubleshoot form issues in model-driven apps
Collaborative debugging with Live monitor
Advanced monitoring

Feedback

Was this page helpful? Yes No

Provide product feedback

Debugging canvas apps with Live monitor

Article • 11/19/2024

Live monitor is available by default for all canvas apps. Using Live monitor, you can trace events as they occur in a canvas app during the authoring experience in Power Apps Studio, or you can use Monitor to debug the published version of a canvas app. More information: Live monitor overview

Get started with Live monitor in canvas apps

To open Monitor when authoring an app

1. Sign in to Power Apps.
2. Create a new app, or edit an existing app.
3. On the left pane, select **Advanced tools**.
4. Select **Open live monitor**.

This action opens Live monitor in a new browser tab and connects it to your existing Power Apps Studio session.

A notification appears at the top to show the current monitoring session as a *Studio session*.

> **Tip**
>
> Live monitor has no impact on your app. You can use Live monitor with any app in a test environment or in production.

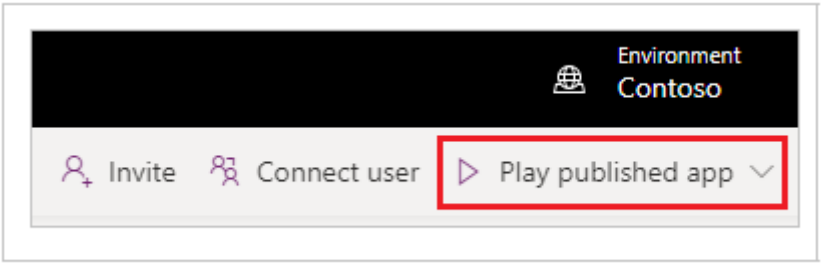
Open Live monitor for a published app

You can also use Live monitor to debug the published app in the web player.

To open Live monitor for a published app

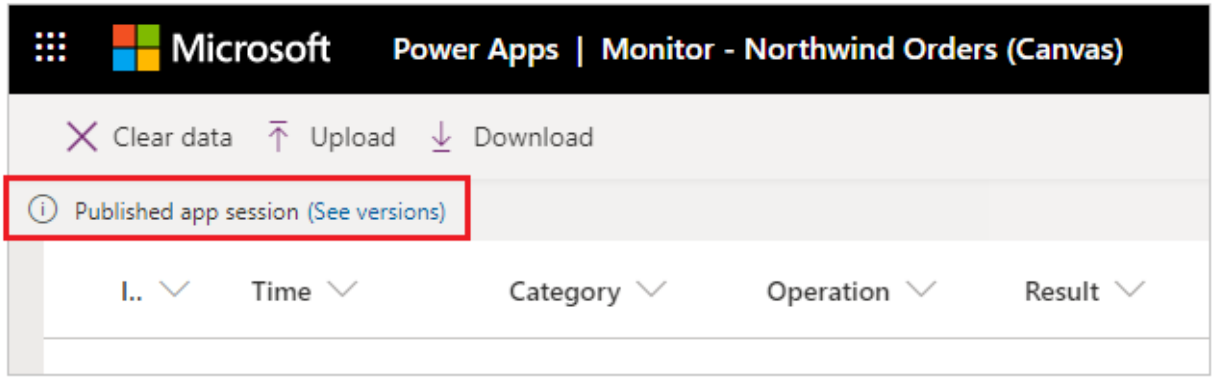
1. Sign in to Power Apps.
2. On the left pane, select **Apps**.

3. Select an app from the list.
4. Select the drop-down menu next to **Details** and then select, **Live monitor**.
5. Select **Play published app**.



This action opens the published app in a new browser tab and connects it to your current Live monitor session. You'll immediately see events in Live monitor when the app is loaded in the web player, and as you interact with the published app.

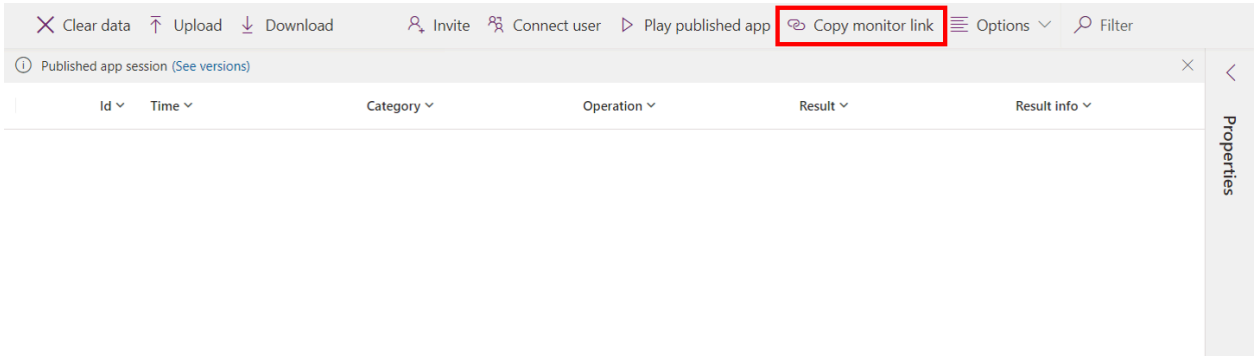
Live monitor also displays a notification that the monitoring session that's currently open is for the published version of the app.



For apps running on Power Apps mobile (preview)

Follow the above steps, but select **Copy monitor link** instead of **Play published app**. Use the copied link on your device to open a monitored session for the published app. Make sure the link is opened using Power Apps mobile, not a browser.

> **Note**
>
> **Copy monitor link** is available in https://make.preview.powerapps.com

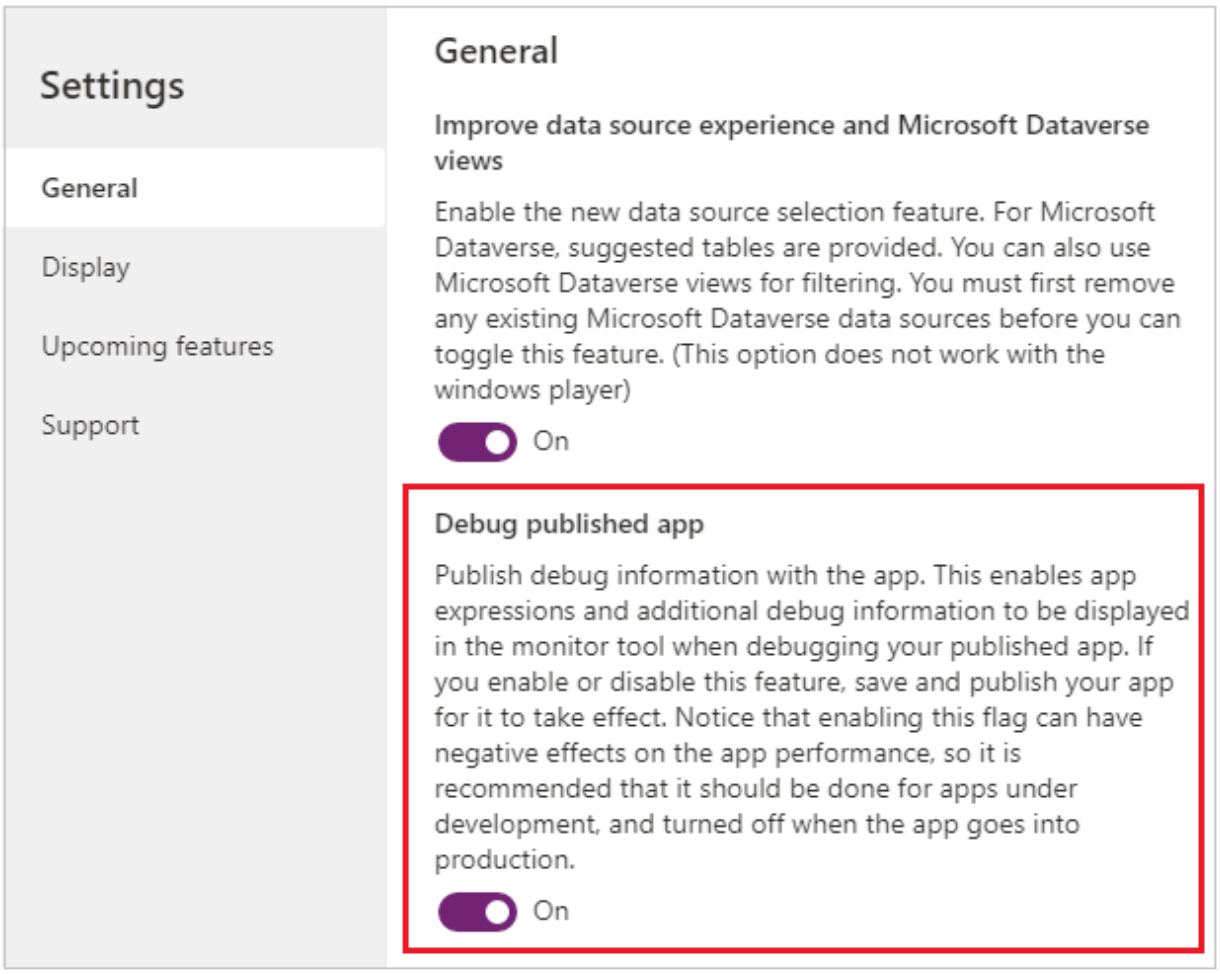


Setting: Debug published app

If you want to view the source expressions in Live Monitor for the published app, you need to turn on the setting to publish the expressions with the app. This setting is similar to generating a debug file in traditional development. Publishing source expressions with your app is optional. Even when this setting is off, you'll be able to see the events happening in your app, but you won't be able to map these events to specific expressions or formulas.

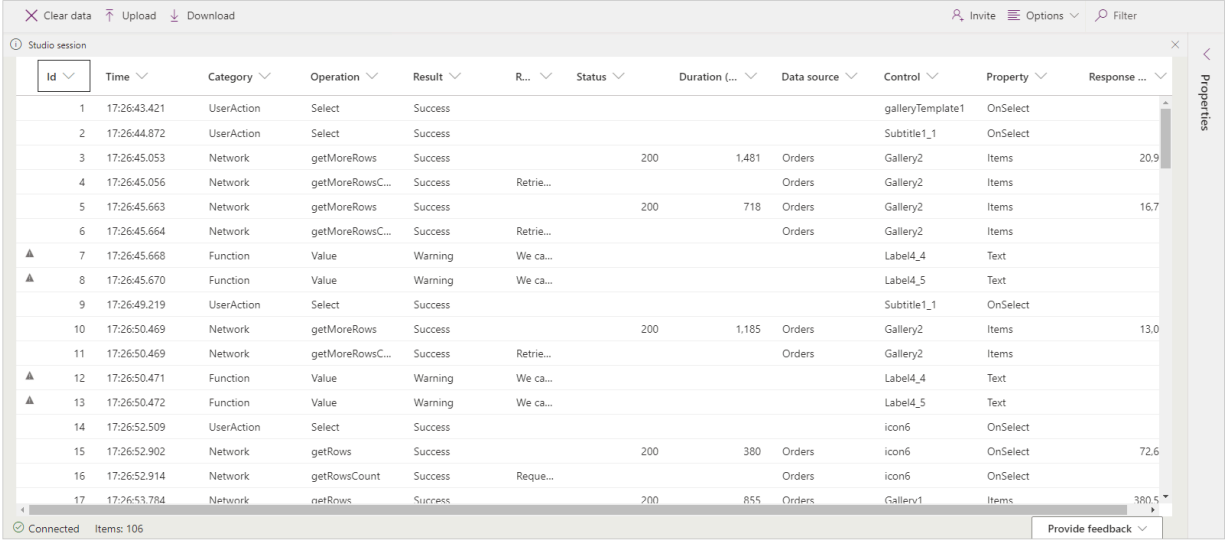
To enable this setting, go to **File** > **Settings**, and then turn on **Debug published app**.

> **Note**
>
> Enabling this setting has a detrimental impact on the performance of your app for all your users. To minimize the impact, disable this setting as soon as you no longer need to view source expressions when debugging your published app.



View events in Live monitor

To view events from your app, play the app in Power Apps Studio. Live monitor will then display the table of events as they're occurring, along with specific details.



Example: Using Live monitor with canvas apps

In this example, you'll use the Northwind Sample Data app included with the Northwind sample solution.

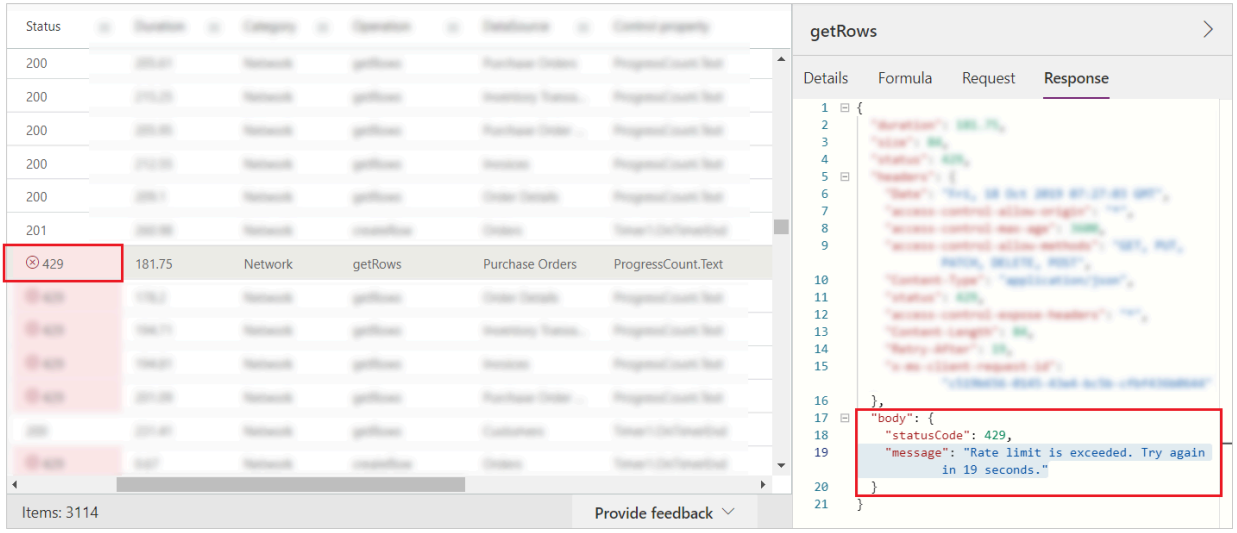
The *Northwind sample solution* is a canvas app that loads sample data into Microsoft Dataverse. You can also create a new app or use an existing app instead.

Background

Consider the scenario where an app has been deployed, and the initial version of the app experiences performance degradation. The app also intermittently generates errors with no clear pattern. Loading data in the app succeeds most of the time, but fails sometimes.

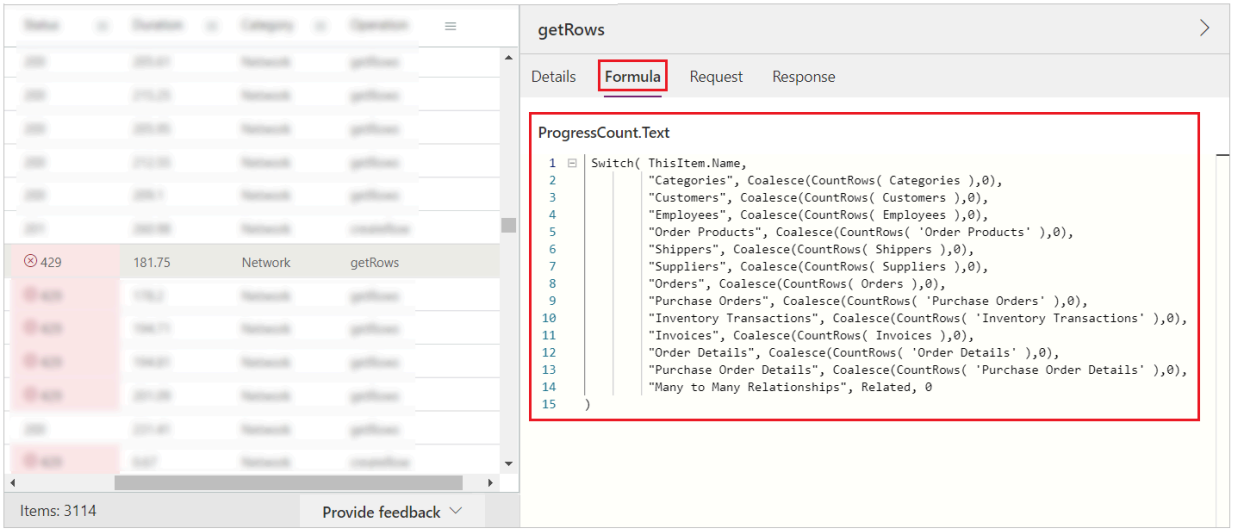
When you check Live monitor, you see data operations as expected. However, you also see several responses that have HTTP status code 429, indicating that there have been too many requests in a specific timeframe.

When you select such an event, you see the error as "Rate limit exceeded. Try again in XX seconds."



Analysis

The problem needs further analysis to understand why requests are getting throttled. In Live monitor, you see that for each **createRow** call, there are several **getRows** requests from the **ProgressCount.Text** property, each to a different entity. These entities aren't the entities the app is creating rows for. The **ProgressCount.Text** formula is seen in Live monitor, as shown in the following image.



For each record added, the formula is evaluated again and **CountRows** is called on several entities. This behavior results in **getRows** in the log, because **CountRows** isn't delegated for Dataverse. For each single request to add a record, you're potentially making 12 additional requests to count the rows in each entity.

These extra requests intermittently cause errors because the Dataverse platform is throttling the requests to the service. This also explains the overall performance problem.

Next steps

Collaborative debugging with Live monitor

See also

Advanced monitoring
Debugging model-driven apps with Monitor

Feedback

Was this page helpful? Yes No

Provide product feedback

Debug a model-driven app with Live monitor

Article • 11/19/2024

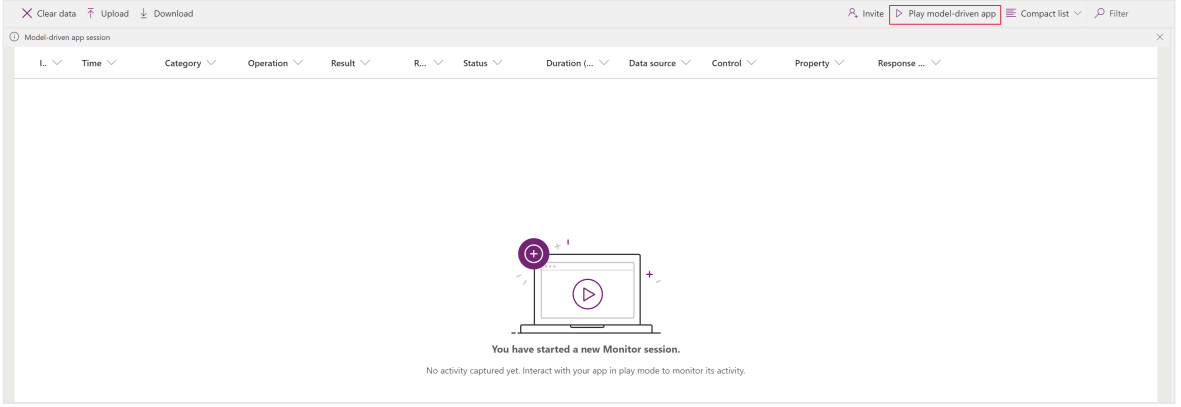
Live monitor can help you debug and diagnose problems and help you build faster, more reliable apps. Live monitor provides a deep view into how an app runs by providing a log of all key activities in your app as the app runs. You can also collaboratively debug a published model-driven app with a support engineer or debug issues of another user's session. For more information on how to collaboratively debug a published model-driven app, go to Collaborative troubleshooting using Live monitor.

> **Important**
>
> Live monitor for model-driven app requires the client version of your model-driven app to be 1.4.5973-2307.2 or later.

For more information about when you should monitor an app, see Monitor overview. To run Live monitor with a canvas app, see Debugging canvas apps with Live monitor.

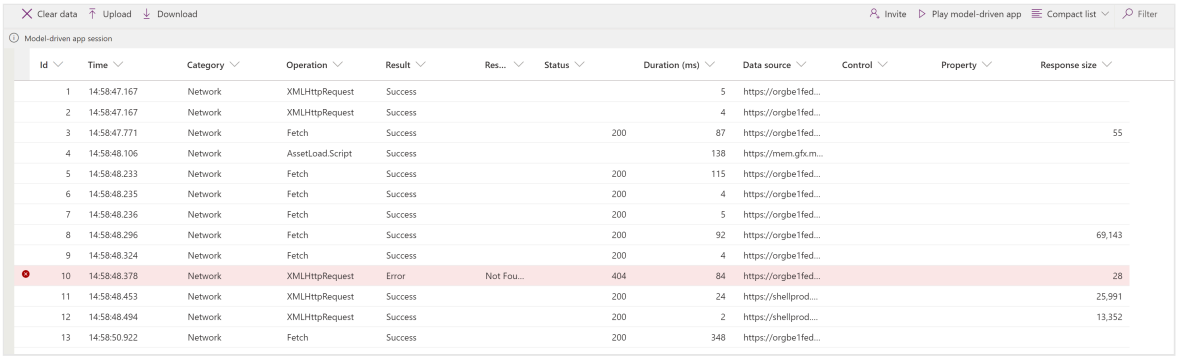
Use Live monitor to diagnose a model-driven app

1. Sign into Power Apps, and then select **Apps** from the left navigation pane.
2. Select the model-driven app that you want to monitor, and then select **Live monitor** on the command bar.
3. In the web page that opens, select **Play model-driven app** to open your app.



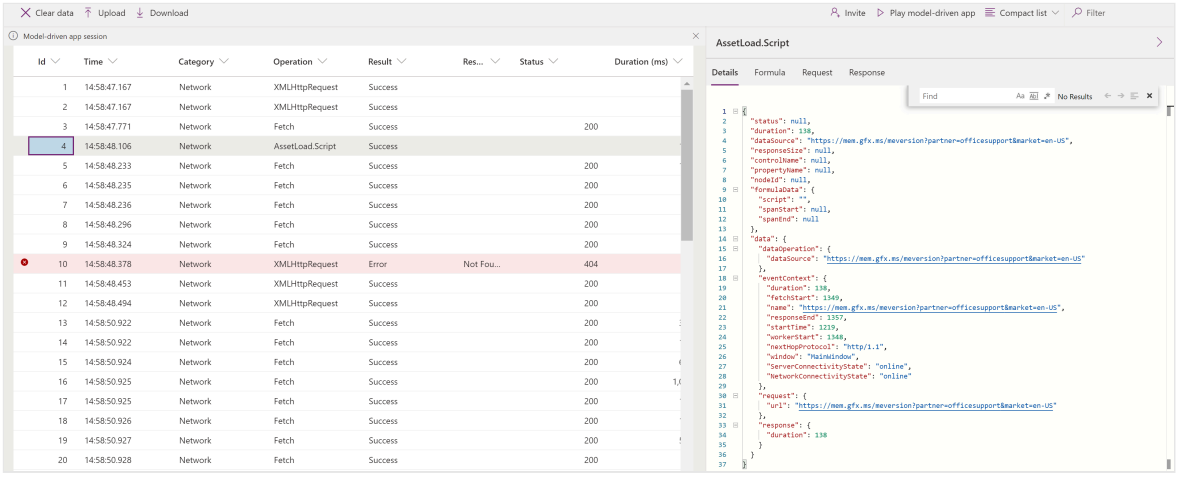
> **Note**
>
> Alternatively, you can run a model-driven app and add "&monitor=true" to the end of the URL in the browser. Then, select **Live monitor** on the command bar to start a monitoring session in a new tab.

4. After the app is opened from Live monitor, you'll see a **Join monitor debug session?** dialog box. This lets you know that any data from the app will be sent to the Live monitor owner. Select **Join**. Events begin to flow to the Live monitor session screen as they occur in the app.



Id	Time	Category	Operation	Result	Res...	Status	Duration (ms)	Data source	Control	Property	Response size
1	14:58:47.167	Network	XMLHttpRequest	Success			5	https://orgbe1fed...			
2	14:58:47.167	Network	XMLHttpRequest	Success			4	https://orgbe1fed...			
3	14:58:47.771	Network	Fetch	Success		200	87	https://orgbe1fed...			55
4	14:58:48.106	Network	AssetLoad.Script	Success			138	https://mem.gfx.m...			
5	14:58:48.233	Network	Fetch	Success		200	115	https://orgbe1fed...			
6	14:58:48.235	Network	Fetch	Success		200	4	https://orgbe1fed...			
7	14:58:48.236	Network	Fetch	Success		200	5	https://orgbe1fed...			
8	14:58:48.296	Network	Fetch	Success		200	92	https://orgbe1fed...			69,143
9	14:58:48.324	Network	Fetch	Success		200	4	https://orgbe1fed...			
10	14:58:48.378	Network	XMLHttpRequest	Error	Not Fou...	404	84	https://orgbe1fed...			28
11	14:58:48.453	Network	XMLHttpRequest	Success		200	24	https://shellprod...			25,991
12	14:58:48.494	Network	XMLHttpRequest	Success		200	2	https://shellprod...			13,352
13	14:58:50.922	Network	Fetch	Success		200	348	https://orgbe1fed...			

5. Select an event to display additional information in the right pane.



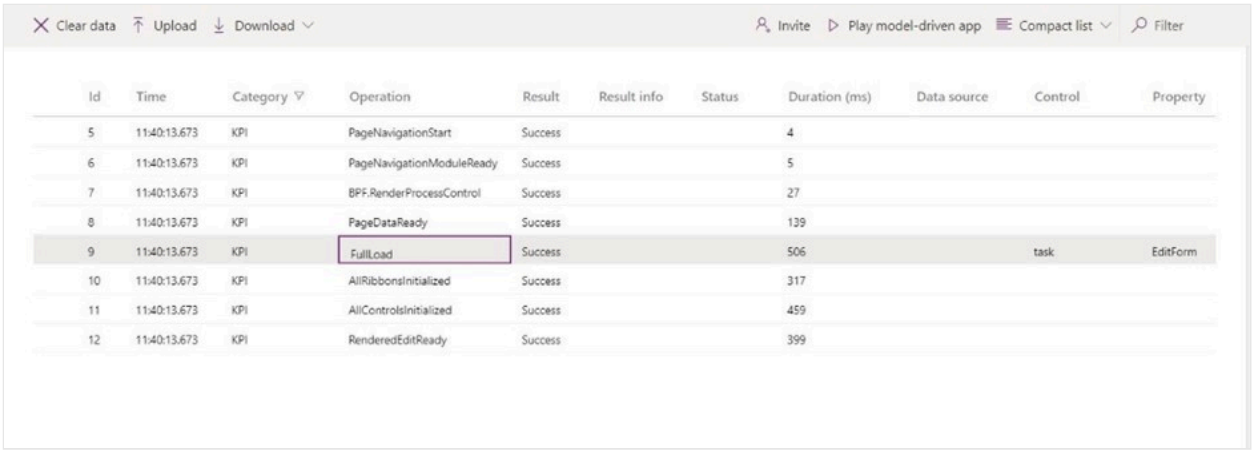
Events monitored

Page navigation, command executions, form saves, and other major actions will send key performance indicators and network events to Live monitor.

FullLoad

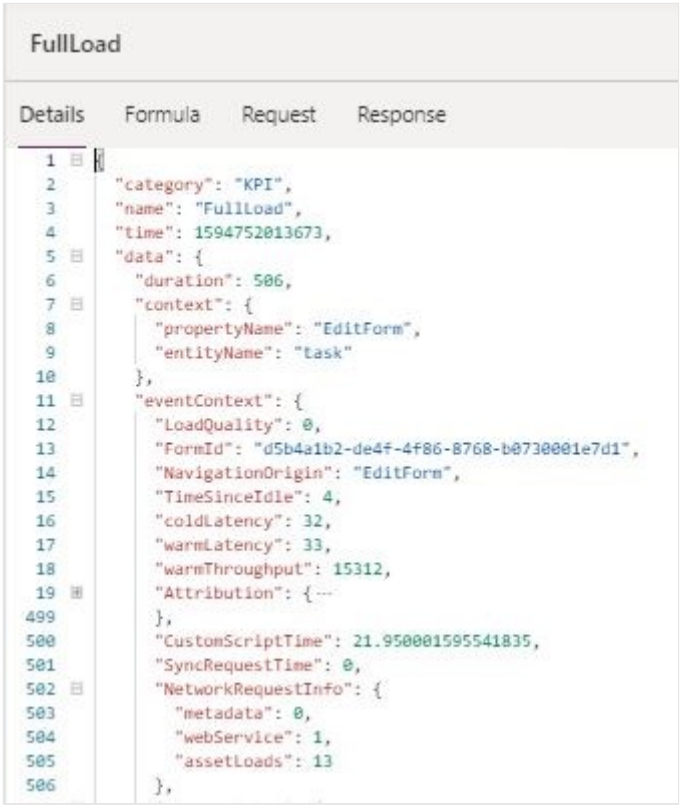
FullLoad signifies the complete load of a page navigation, such as an edit form load. This event waits for certain network requests to complete and all rendering to finish, so the

form can be ready before FullLoad completes.



Id	Time	Category	Operation	Result	Result info	Status	Duration (ms)	Data source	Control	Property
5	11:40:13.673	KPI	PageNavigationStart	Success			4			
6	11:40:13.673	KPI	PageNavigationModuleReady	Success			5			
7	11:40:13.673	KPI	BPF.RenderProcessControl	Success			27			
8	11:40:13.673	KPI	PageDataReady	Success			139			
9	11:40:13.673	KPI	FullLoad	Success			506		task	EditForm
10	11:40:13.673	KPI	AllRibbonsInitialized	Success			317			
11	11:40:13.673	KPI	AllControlsInitialized	Success			459			
12	11:40:13.673	KPI	RenderedEditReady	Success			399			

Select a FullLoad event to display the **Details** tab.

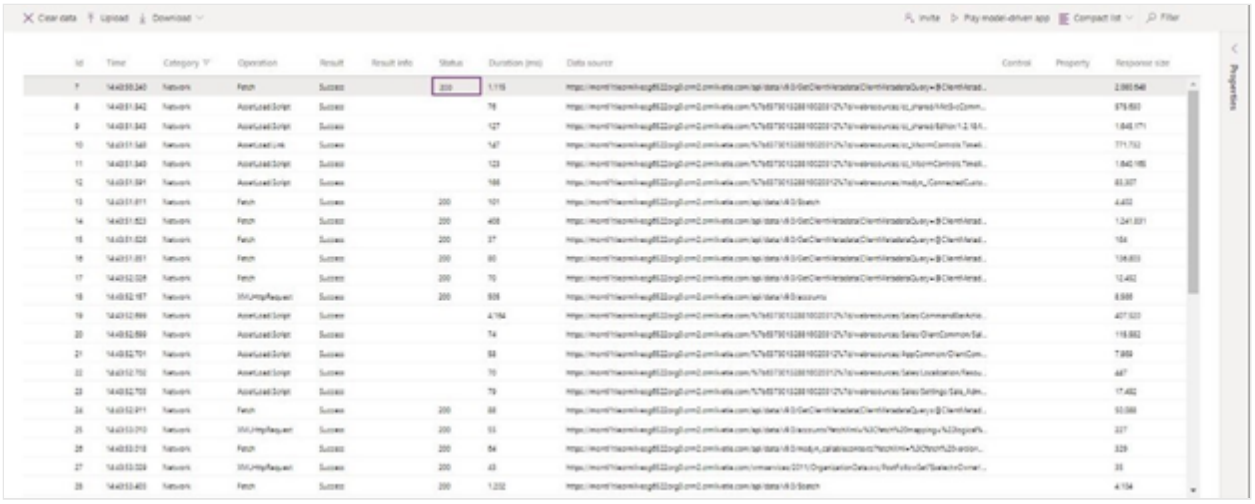



```
{
  "category": "KPI",
  "name": "FullLoad",
  "time": 1594752013673,
  "data": {
    "duration": 506,
    "context": {
      "propertyName": "EditForm",
      "entityName": "task"
    },
    "eventContext": {
      "LoadQuality": 0,
      "FormId": "d5b4a1b2-de4f-4f86-8768-b0730001e7d1",
      "NavigationOrigin": "EditForm",
      "TimeSinceIdle": 4,
      "coldLatency": 32,
      "warmLatency": 33,
      "warmThroughput": 15312,
      "Attribution": {···
      },
      "CustomScriptTime": 21.950001595541835,
      "SyncRequestTime": 0,
      "NetworkRequestInfo": {
        "metadata": 0,
        "webService": 1,
        "assetLoads": 13
      },
```

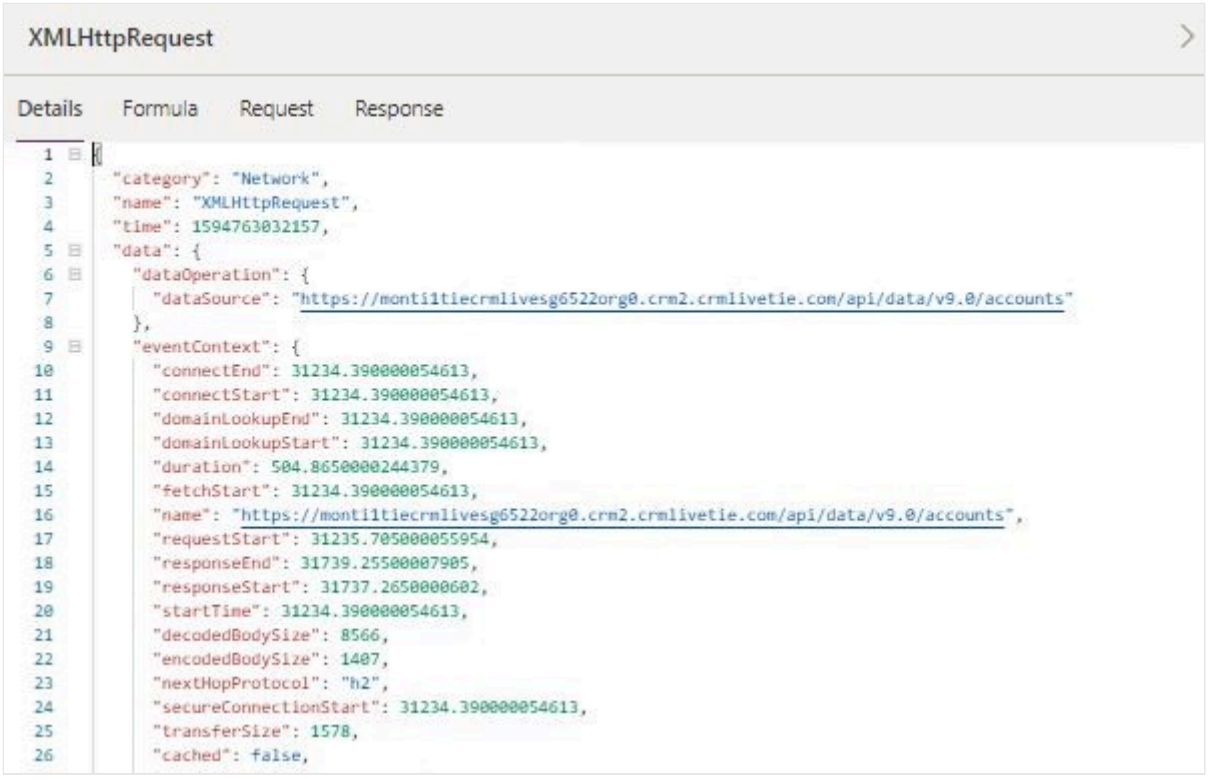
The FullLoad event captures many statistics about the page load. You can see the task edit form loaded in 506 ms, and selecting the row reveals information in the property pane. You can see details on `customScriptTime` (time spent execute custom JavaScript), `loadType` (0 = first-time loading page type, 1 = first time loading entity, 2 = first-time loading record, 3 = exact record has been visited), and `FormId` (form identifier for further diagnosis). Expanding **Attribution** gives a breakdown of custom JS execution time by type, publisher, solution, version, web resource, and method. This can help identify bottlenecks in form load time.

Network

The **Network** events reveal details about each request made from the app.



Select a Network event to display the **Details** tab.

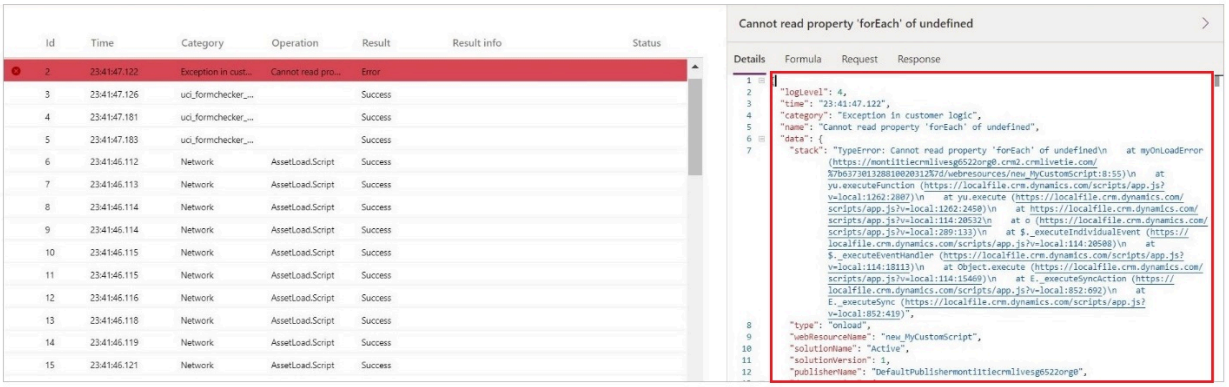


Earlier you learned about Live monitor and how to use it to model-driven apps. Let's look at a few scenarios where Live monitor can assist you in solving script errors, unexpected behavior, and slowdowns.

Custom script errors

Sometimes, a bug in the custom JS causes a script error or functionality problem when loading a page. While this usually produces a call stack in the dialog, it's hard to always know where it's coming from or decode the error. Live monitor receives events from the app with more details about the errors, so you can debug more quickly, and easily.

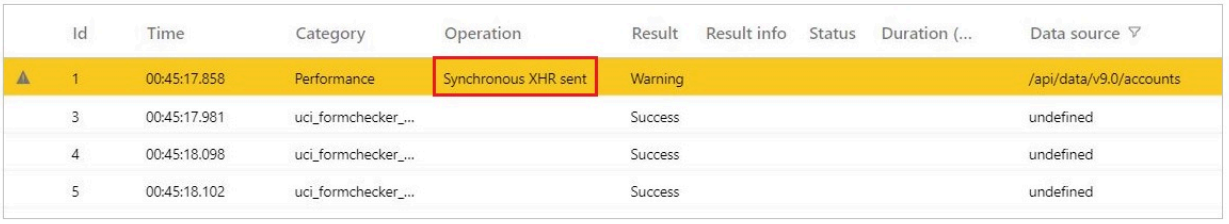
For example, imagine a user is experiencing a script error dialog on account form load. You can use Live monitor to get more details on the event. Once the scenario is reproduced, you can see the script error produces an error event highlighted in red. Selecting this row gives you not only the **call stack** but the **publisher name**, **solution name/version**, **web resource name**, and **type** (such as *onload*, *onchange*, *RuleEvaluation*, and *CustomControl*). In this example, it looks like a typo in the script.



Slow performance

Browser developer tools can help profile slow page loads, but there's much data to filter though and it's not clear what is important to look at. Live monitor solves this problem by showing relevant events that contribute to page load performance.

For example, imagine a user is experiencing slow account form loads, and the browser is constantly freezing up. In this case, after you reproduce the issue, you can see a performance warning telling you that a synchronous **XMLHttpRequest** was sent during the load which degraded performance.



See previous blog post for how to alleviate synchronous XMLHttpRequest (XHR) performance problems.

For every page load, the data platform sends all KPI for the loading sequence as well as network request details as mentioned earlier.

See also

Use Live monitor to troubleshoot model-driven app form behavior

Feedback

Was this page helpful? Yes No

Provide product feedback

Use Live monitor to troubleshoot model-driven app behavior

Article • 11/19/2024

Live monitor is a tool that can help app makers debug and diagnose problems, which help them build faster, more reliable apps. Live monitor provides a deep view into how an app runs by providing a log of all activities in the app as it runs.

To help you isolate certain model-driven app features, you can filter on specific categories, such as forms or the Copilot control.

Create a Live monitor session

Sign in to Power Apps, select **Solutions** on the left navigation pane, open the solution that contains the app you want to monitor, select the app, and then select **Live monitor** on the command bar.

On the Live monitor page, select **Play model-driven** app on the command bar. For more information about creating a Live monitor session, go to Use Live monitor to troubleshoot page behavior in model-driven apps.

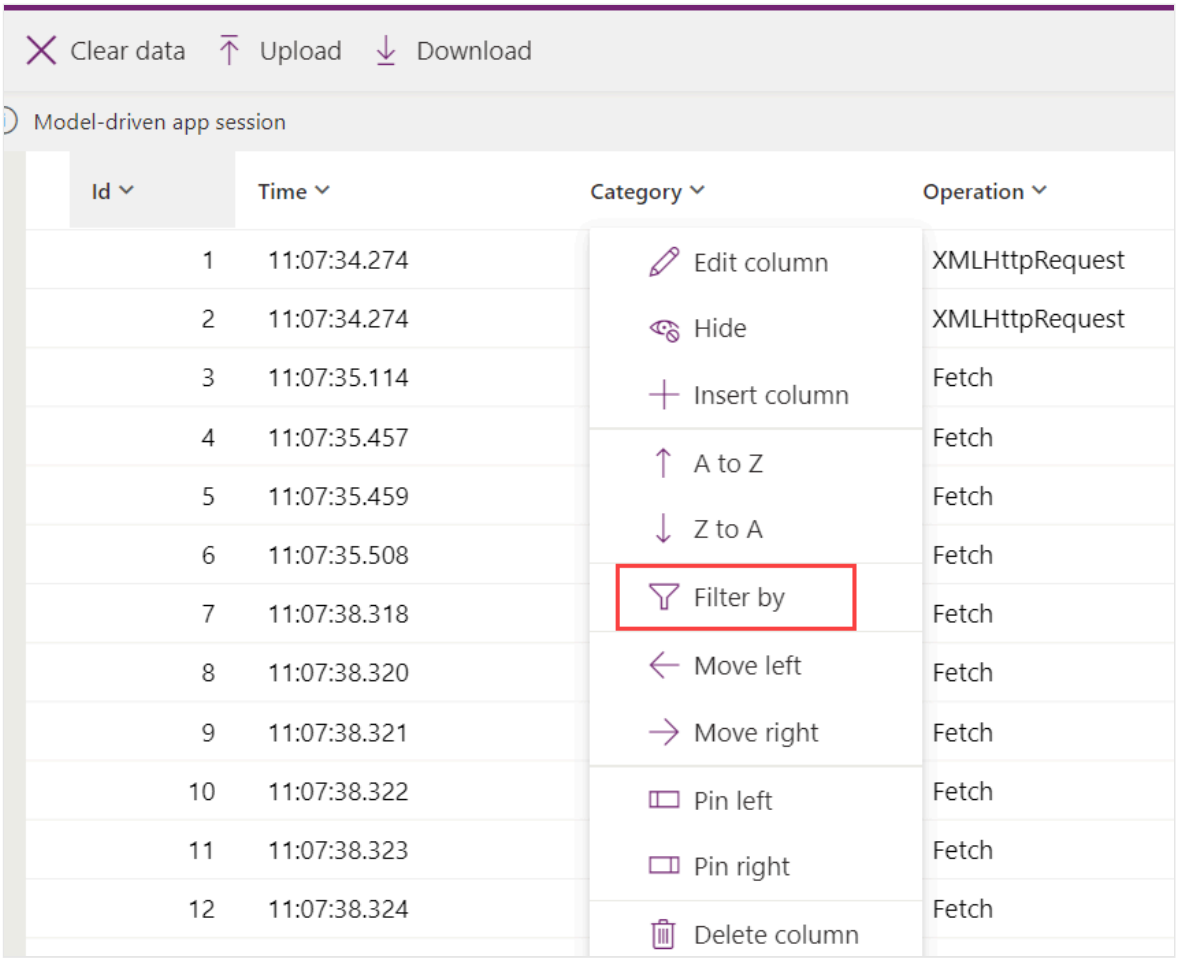
Filter Live monitor for form-related issues

Filtering on model-driven app form-related events in Live monitor can provide information about related tables, tables, controls, and components on a form in Live monitor as your app runs.

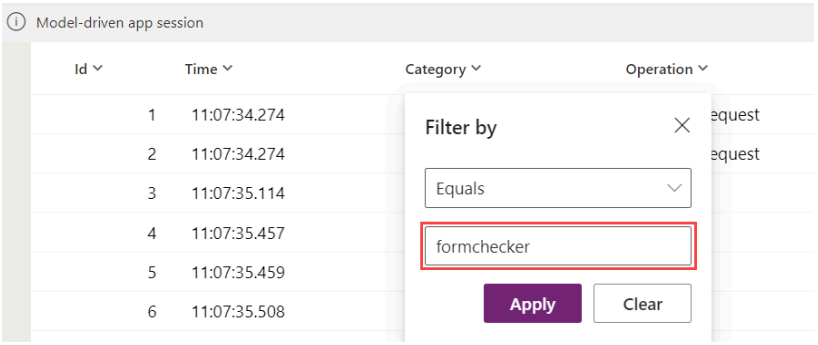
There are many situations where Live monitor can help makers understand why a form behaves a certain way. Many form issues are based on business rules, JavaScript, form events, or client API that admins and makers set. Live monitor can also help identify whether the issue experienced is designed out-of-the-box or is due to a customization. It provides details that can help answer the following questions:

- Why aren't rows showing in the related menu of a table?
- Why a control is disabled/enabled or visible/hidden
- Why is a row in a read-only state?

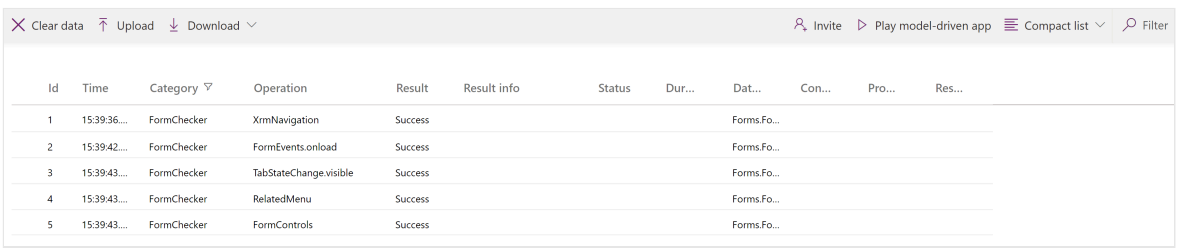
1. On the browser window running Live monitor, select the **Category** column, and then select **Filter by**.



2. Select **Equals** or **Contains** from the dropdown list, and then enter `formchecker` in the box. Select **Apply**.



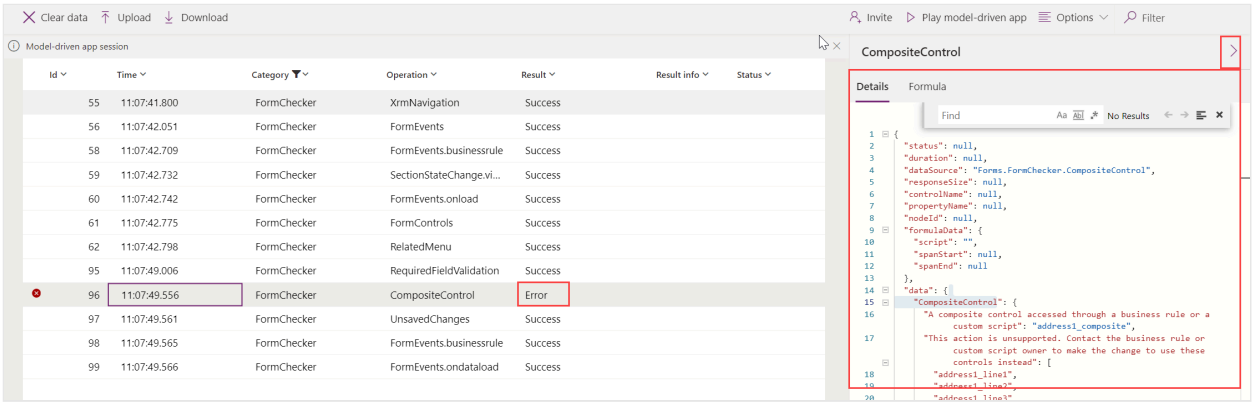
3. The categories are now filtered. The **Operation** column can be expanded to see the full name of the events that are tracked by selecting and holding the right side of the column and dragging to the right. As you use the app and open and use a form, Live monitor updates the list of events.



4. While the app is running in the monitored session, perform actions within the model-driven app consistent with normal use of the app. For example, open and change data using a table form.

Use Live monitor to understand form behavior

For each row with Live monitor, detailed information about the form event can be reviewed. For example, imagine you have a question about an error taking place within the form. You go to that form in the app and select the appropriate form component. Then return to the browser with Live monitor enabled and review the results either with or without filtering. In this case, there's an error on the composite control. By expanding areas of the **Details**, you can learn more about the event itself.



There are many types of events that are monitored, including the standard form events like `onload`, `onsave`, and `onclose`.

As you continue to use the app that's being monitored, Live monitor updates the information in the list of events. For forms, there are many different scenarios that you can troubleshoot and find additional information on the form, control, or table currently being worked on.

Form-checking areas and events

Areas for form monitoring include the following.

Expand table

App area	Description
Control state	Details about the state of the visible, enabled, and label source of a control when the form is loaded.
Related menu	Details about the state of related menu items. Examples: Why is a menu item not being displayed?

App area	Description
	Where does the menu item come from?
Tab / section / control state change	Details on who (via the callstack) has caused a form component—such as a tab, section, or control—to change the component's visibility and enabled state.
Navigation	Details about what's causing navigation or unexpected dialogs by tracing the callstack of these `Xrm.Navigation` client API methods: `openAlertDialog(), openConfirmDialog(), openDialog(), openErrorDialog(), navigateTo(), openForm(), openTaskFlow(), openUrl(), openWebResource()`
Unsupported customizations	Details about unsupported client API access before the form is ready. Examples: Accessing `parent.Xrm.Page` in iFrame before the form is fully loaded. Accessing `Xrm.Page` in a form web resource outside of form handler contexts using `window.setTimeout()` to periodically call the form client API. Accessing `Xrm.Page` in `updateView()` method of the Power Apps control framework control code.

Examples of the supported form-related events in Live monitor include:

- FormEvents.onsave
- XrmNavigation
- FormEvents.onload
- FormControls
- TabStateChange.visible
- RelatedMenu
- ControlStateChange.disabled
- ControlStateChange.visible
- SectionStateChange.visible
- UnsupportedClientApi

Filter Live monitor for Copilot control-related issues

1. On the browser window running Live monitor, select the **Category** column, and then select **Filter by**.
2. Select **Equals** or **Contains** from the dropdown list, and then enter `CopilotEvent` in the box. Select **Apply**.

Copilot control events

Events for Copilot control monitoring include the following.

Expand table

Event name	Description
AdaptiveCardActionData	Data sent to submit action for adaptive card
adaptiveCardContent	Content of adaptiveCard
AISummarize	This message is logged in case the summarize feature is off and logs the fcb status
AutoTriggerSummarizeHandler.isAutoTriggeredSummaryEnabled	Tells if auto trigger is enabled
ClientActionCalled	Logs the name of the client action and payload passed ot it
CopilotResponse	Response from the copilot
DVCopilotAction	isDVChatEnabled: if dv chat is enabled for the org or not
DynamicContextSentToBot	Dynamics context sent to bot
EnvironmentHostedBotSchemaName	Logs the environment bot schema name
ExecutingAction	Adaptive card action is getting executed
isSummaryEnabledForEntity	Tells if summarize is enabled for the entity
queryStructuredDataRequest	Request sent to query structure data
queryStructuredDataResponse	Response from query structure data
SessionContextSentToBot	Session context sent to bot
SummarizeResponse	Logs the response for the summarize
UserPrompt	Prompt entered by the User

Close a monitoring session

To close the monitoring session, close the browser tab where the monitored model-driven app is playing.

Next steps

For more information about how to troubleshoot issues with forms in a model-driven app, see Troubleshoot form issues in model-driven apps.

Learn about Live monitor as a Power Apps tool

Feedback

Was this page helpful? Yes No

Provide product feedback

Troubleshoot form issues in model-driven apps

10/10/2025

This article has information to help fix some of the common issues you might encounter while working with model-driven app forms.

> **Important**
>
> - The tools described in this article are designed for troubleshooting purposes; they aren't meant to be used in day-to-day production scenarios, even though you can use them for troubleshooting issues in production environments.
> - These troubleshooting tools only affect the current user session unless otherwise noted (for example, when a browser tab accesses the model-driven app). They don't change system customizations or affect any other users or sessions. After the current session is closed, the effect is no longer applied.
> - Most of the tools are available in all the production environments. Some of them mentioned in the article might not be deployed to your organization yet; new tools are added periodically.
> - Tools listed in this article are written in a scenario-driven way. You can use them independently to troubleshoot different types of issues.
> - Use URL parameters to disable various form components and View registered form event handlers and libraries in Monitor are critical and fundamental tools you'll frequently use to troubleshoot many scenarios.
> - For more information on how to use Monitor, see Use Monitor to troubleshoot model-driven app form behavior

Use URL parameters to disable various form components

When you're troubleshooting issues with forms, you need to use the URL parameters to disable components as you work to isolate the specific component that caused the issue. We recommend that you use the flags one at a time to narrow down the cause of the issue. You can use the following URL parameters:

- **DisableFormCommandbar**

Disables the command bar on the form. It only disables the command bar on form pages and not supports list (grid), dashboard, etc.

```
HTTP

https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000**&flags=DisableFormCommandbar=true
```

- **DisableFormHandlers**

  Disables all the form handlers. If you use the **DisableFormHandlers=true** flag, it disables the following event handlers: OnLoad, OnSave, business rule, OnChange, and TabStateChange.

  To learn more about obtaining event or library indices for granular controls, see View registered form event handlers and libraries in monitor.

  ```
  HTTP

  https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000**&flags=DisableFormHandlers=true
  ```

  - **&flags=DisableFormHandlers=*eventName***

    Disables the form handler by specifying the event name, for example, ****DisableFormHandlers=onload**.

    ```
    HTTP

    https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000**&flags=DisableFormHandlers=true
    ```

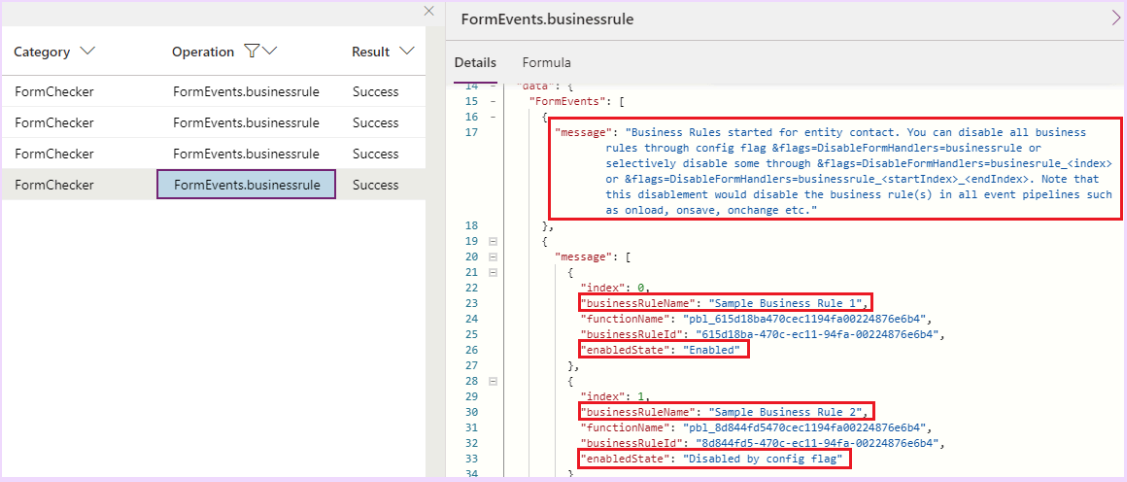
  - **&flags=DisableFormHandlers=*eventName_index***

    Disables the event handler at the specified index for any supported event name. For example, `DisableFormHandlers=true_0` disables all the event handlers at index 0. `DisableFormHandlers=onload_2` disables the OnLoad event handler at index 2.

  - **&flags=DisableFormHandlers=*eventName_startIndex_endIndex***

    Disables all the event handlers within the given range by specifying `startIndex` and `endIndex` values (both are included). For example, `DisableFormHandlers=true_0_2`

disables all the event handlers of index 0, 1, and 2. `DisableFormHandlers=onload_2_5` disables the OnLoad event handler of index 2, 3, 4, and 5. If you have more event handlers, you can use this approach to narrow down problematic handlers quickly.

> **Note**
>
> Business rules are authored in the business rule designer, compiled into the client-side script, and registered in multiple form events, such as `OnLoad`, `OnSave`, and `OnChange`. The way to disable business rules are similar to other form events. However, there are a few key differences:
>
> - When you use `DisableFormHandlers=true`, `businessrule`, `businessrule_*index*`, or `businessrule_*startIndex_endIndex*`, you're disabling all business rules in all the form events they're registered to.
> - For example, the following image shows instructions on refreshing business rules in the backend. You only need to do it once in your organization, and you can revert your changes after troubleshooting.
>
> 
>
>
> - After you perform the above action and refresh the form, you'll see different message with additional information, as shown in the following image:
>
> 
>

- **DisableFormLibraries**

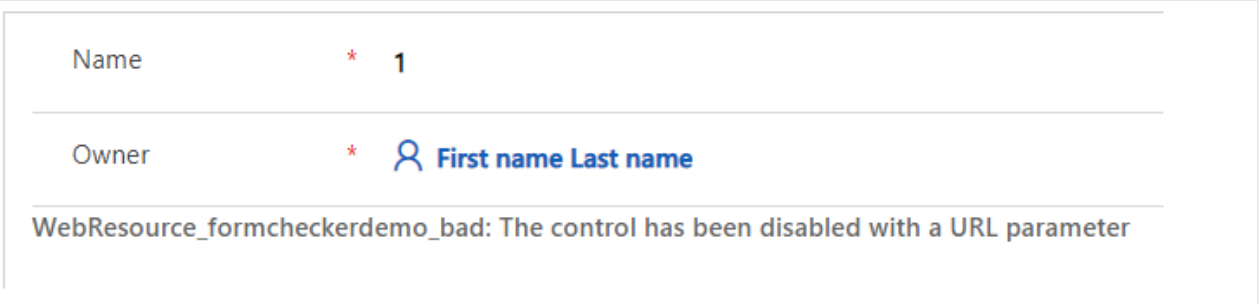
  Disables form libraries and prevents the libraries from being loaded. To learn more about obtaining event or library indices for granular controls, see View registered form event handlers and libraries in monitor . The usage is similar to `DisableFormHandlers`, except it doesn't take an event name as the value.

- **&flags=DisableFormLibraries=true**: Disable all the form libraries.
- **&flags=DisableFormLibraries=*index***: Disable form libraries at the specified index.
- **&flags=DisableFormLibraries=*startIndex_endIndex***: Disable form libraries in the range of startIndex and endIndex (both included).

- **DisableWebResourceControls**

  Disables all the web resource controls on the form.

  ```
  HTTP

  https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000**&flags=DisableWebResourceControls=true
  ```

- **DisableFormControl**

  Disables a form control. Specify the control name to disable the control. If you see that the issue goes away with **&flags=DisableWebResourceControls=true**, and there's more than one web resource control on the form, you can use this flag to further identify the control that's causing the issue.

  ```
  HTTP

  https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000**&flags=DisableFormControl=controlname
  ```

- **DisableBusinessProcessFlow**

  Disables all the business process flows on the form.

  ```
  HTTP

  https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000**&flags=DisableBusinessProcessFlow=true
  ```

- **navbar** This isn't a **flag** parameter; instead, use **navbar=off** in the URL.

You can also add multiple URL parameters separated with a comma (,).

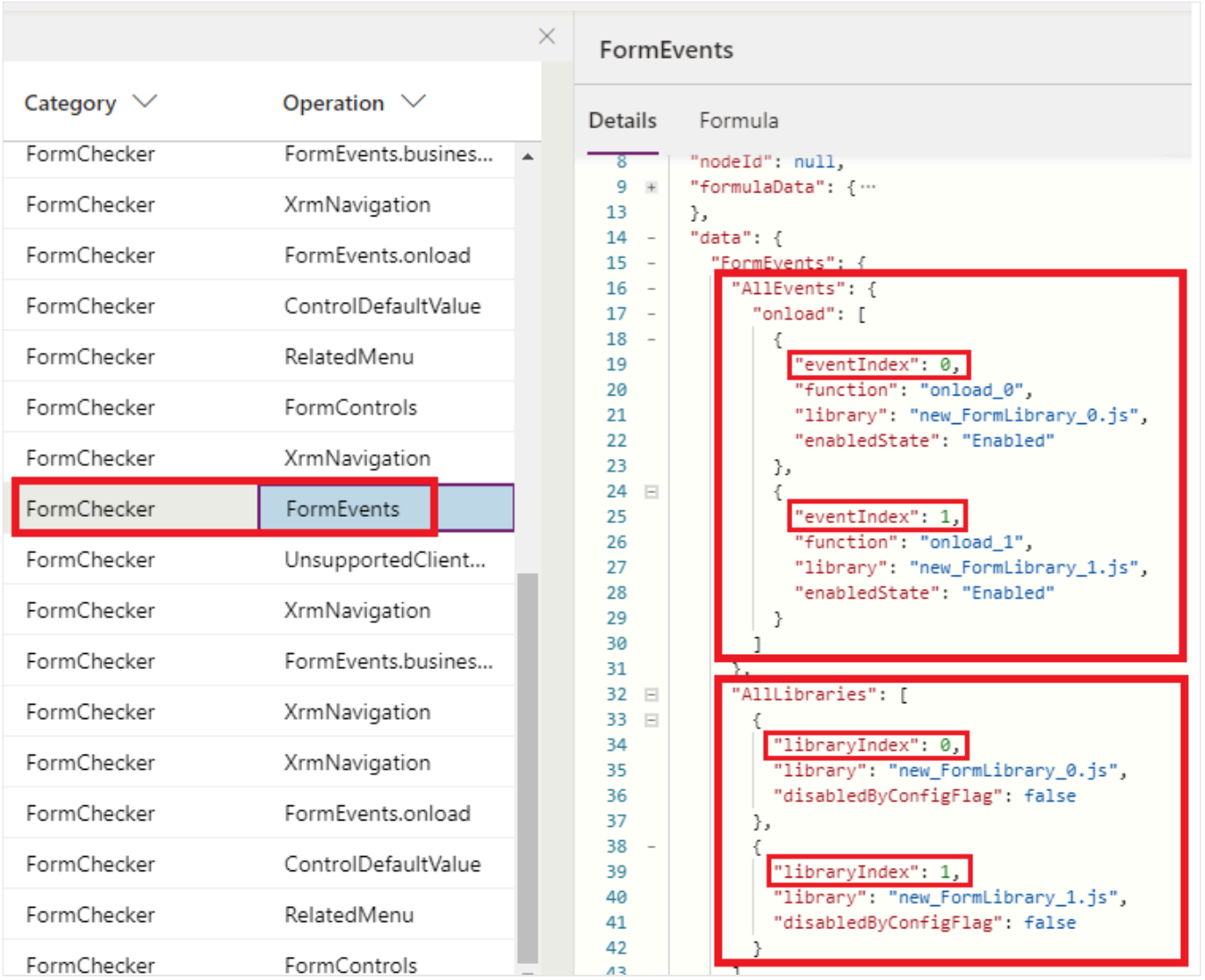
Http

```
https://myorg.crm.dynamics.crm/main.aspx?appid=00000000-0000-0000-0000-000000000000&pagetype=entityrecord&id=00000000-0000-0000-0000-000000000000**&flags=DisableFormHandlers=true,DisableWebResourceControls=true,DisableFormCommandbar=true,DisableBusinessProcessFlow=true&navbar=off
```

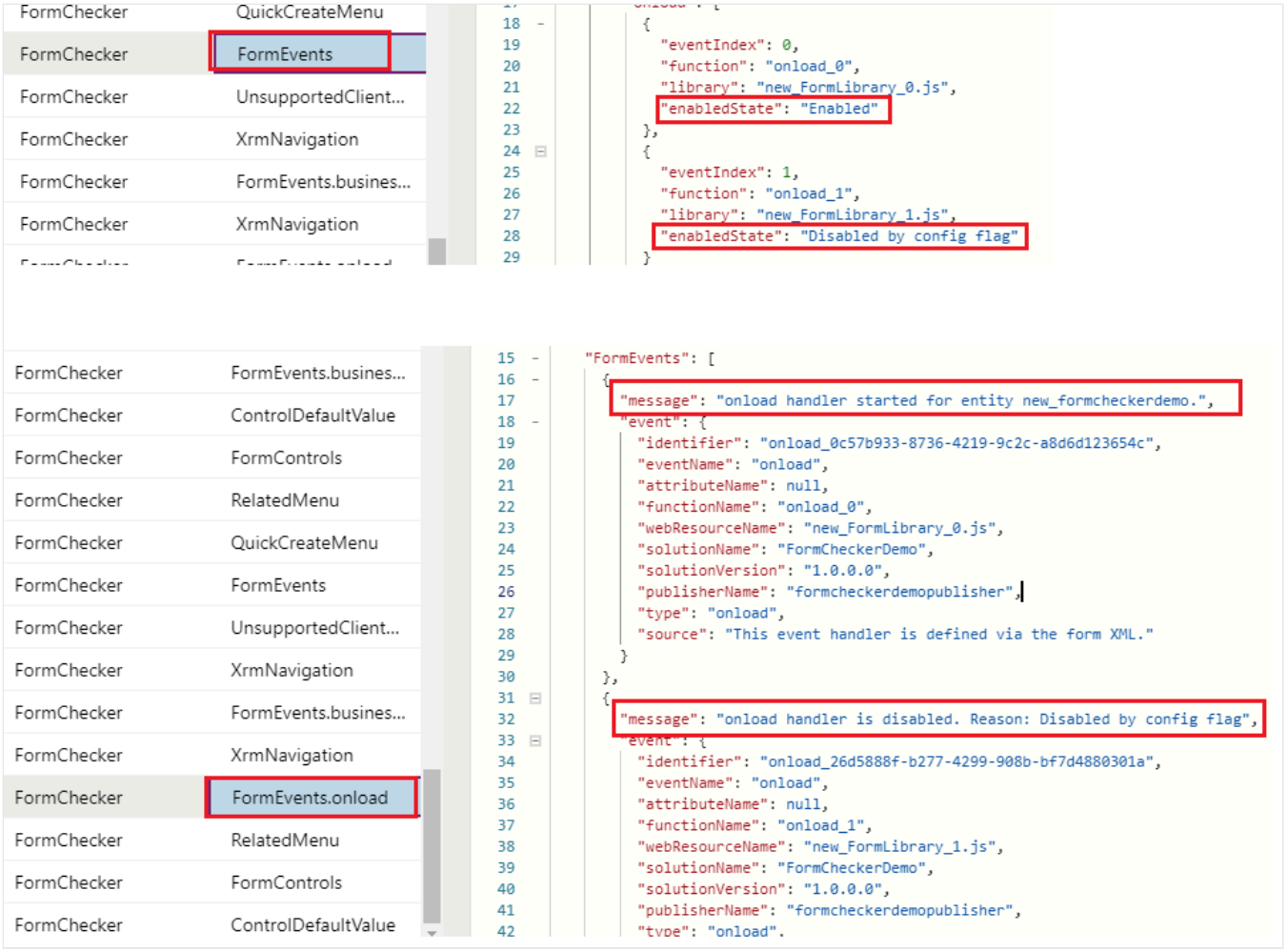
> ⓘ **Note**
>
> The difference between **DisableFormHandlers** and **DisableFormLibraries** are:
>
> - The **DisableFormHandlers** flag disables form handlers regardless of the containing form libraries. In contrast, the **DisableFormLibraries** flag disables the form libraries (web resources) regardless of the functions (event handlers) included in the libraries. Simply put, **DisableFormLibraries** makes sure the specified JavaScript web resource files aren't loaded.
> - The **DisableFormHandlers** flag doesn't prevent the containing form library from being loaded. Thus it doesn't stop the JavaScript code present in the library but not registered as an event handler from being executed. For example, if a form library `new_myscript.js` is written in the following way (not recommended practice):
> - You should start with **DisableFormHandlers** to see if the issue goes away, and if not, you can try **DisableFormLibraries**. Disabling any script always involves some risks of potentially breaking your form scenarios. However, the latter tend to have more side effects because of the disablement of the entire JavaScript files.
>
> ```
> // new_myscript.js
> alert("First alert that's not registered in any form events but will still be triggered when the form loads!");
>
> function myOnloadHandler () {
>     alert("Second alert that's registered in the form onload event!");
> }
> ```
>
> - Assuming the `myOnloadHandler` is registered as an `OnLoad` event handler, the `DisableFormHandlers=true` flag only prevents the second alert, whereas the `DisableFormLibraries=true` flag prevents both alerts.

View registered form event handlers and libraries in monitor

To view registered form event handles and libraries, you can view the `FormEvents` operation in Monitor.



You need the `eventIndex` and `libraryIndex` parameter values when using the **DisableFormHandlers** or **DisableFormLibraries** URL flags. After an event or library is disabled, you'll see the event enabled state in both **FormEvents** operation (an overall view of all registered event handlers of all events), and **FormEvents.*eventName*** operation (details logged when a specific event happens).



Unexpected behaviors when loading a form

Some common issues that can cause unexpected behavior when a model-driven app form is loaded are:

- Columns or controls don't have the values you expect.
- Controls aren't disabled or aren't enabled.
- Controls aren't shown or aren't hidden.

How to troubleshoot

There are multiple reasons why unexpected behaviors occur when a form opens. One of the most common is the OnLoad scripts that run synchronously or asynchronously to change the column or control behavior. To determine whether your script is causing the issue, you can disable the form handlers by appending **&flags=DisableFormHandlers=true** at the end of your app URL.

If the unexpected behavior stops occurring after you disabled the form handler, it's a strong indication that the specific form handler is causing this behavior. When you identify the script that's causing this behavior, follow up with the script owner to further troubleshoot this issue.

Saving in progress error message

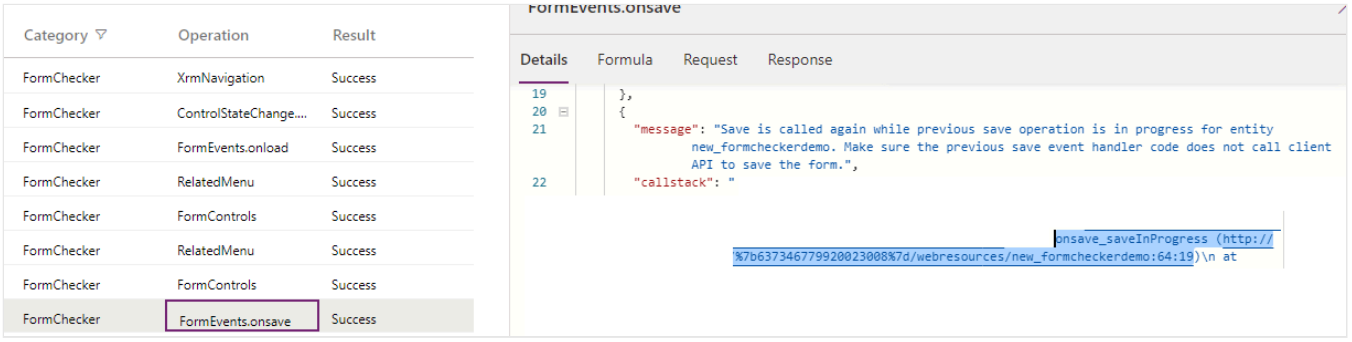
Sometimes when you save a form, you see a **Saving in Progress** error message.

This error occurs when the form OnSave event is triggered before the previous OnSave event completes. This behavior isn't supported, and the error appears by design because calling the `OnSave` event before the previous `OnSave` event is complete causes recursive save loops with unintended consequences.

A typical cause for this error is the script that calls the `save()` method in the OnSave event handler. Another possible cause might be concurrent `save()` calls in the `setTimeout()` method, which might cause the error to intermittently show up, depending on whether the prior `save()` call was completed before another `save()` call was made.

How to troubleshoot

In Monitor, the `FormEvents.onsave` operation provides all the details that are causing the error (this call stack is modified for demonstration purposes). The call stack tells what exact web resource, function, line, and row number causing this error. The form checker can't detect the error if the issue can't be reproduced.



Follow up with the script owner to further troubleshoot the issue.

Intermittent form errors

The most common cause of intermittent or random form errors is using unsupported client API methods. These errors have the following characteristics:

- They occur only for specific records, users, regions, or browsers, or only during periods when the network or service load is high.
- They rarely occur in support instances.
- They might occur once on a computer, and the same error might occur again after you clear the browser cache.

- formContext.getControl or formContext.getControl(arg).getAttribute() randomly returns null for a valid control or column.

There are many ways to write unsupported client API methods, and they all share a common pattern: they cause a race condition in the form load pipeline. Because they introduce a race condition, the issue only occurs when the custom script executes before the form is fully ready to be accessed via the client API. Many factors can cause a race condition:

- In the JavaScript web resource, code is put into a global scope executes immediately when the web resource file is loaded, without waiting for the form to be accessible. Make sure the code is executed inside a valid form handler, such as an OnLoad handler.

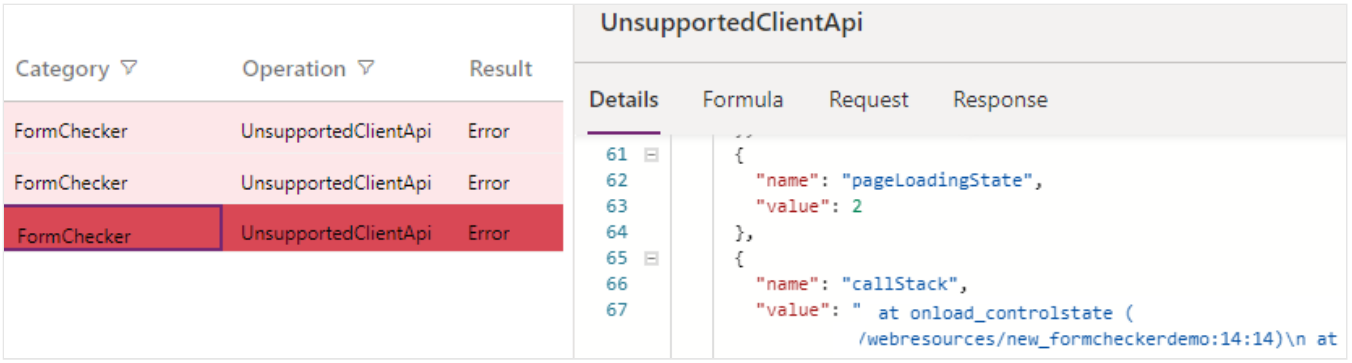
```
// Bad custom script example

// this is a bad place to invoke client API, this API will fail intermittently
// Move it into a form handler, such as onload
Xrm.Page.getAttribute("myAttribute").setDisabled(true);

function myOnloadHandler(executionContext) {
    // this is a recommended place to put your client API code in
}
```

- In the Power Apps component framework component script file, client API methods are accessed inside the init or updateView function. The `init()` and `updateView()` functions are executed immediately when the component is loaded, without waiting for the form to be readily accessible. You can't use unsupported Client API methods in Power Apps component framework components.

Client API is accessed inside a `window.setTimeout()` function in the web resource file. The page state is unpredictable when the `setTimeout()` method executes the wrapped function. Due to the nature of the timer function, when the execution occurs, the page might be in a transitional state (during page load or save) that's not readily accessible by the Client API.

How to troubleshoot

Using Monitor, you can access information that helps you determine when the unsupported client access occurred and when the access occurred at the wrong time due to a race condition. However, Form Checker doesn't report such unsupported client access when the unsupported code happens to execute at the right time that doesn't cause an issue.



> **Note**
>
> The call stack is modified for illustration purposes. The call stack shows details like web resource, function, and the line causing the error.

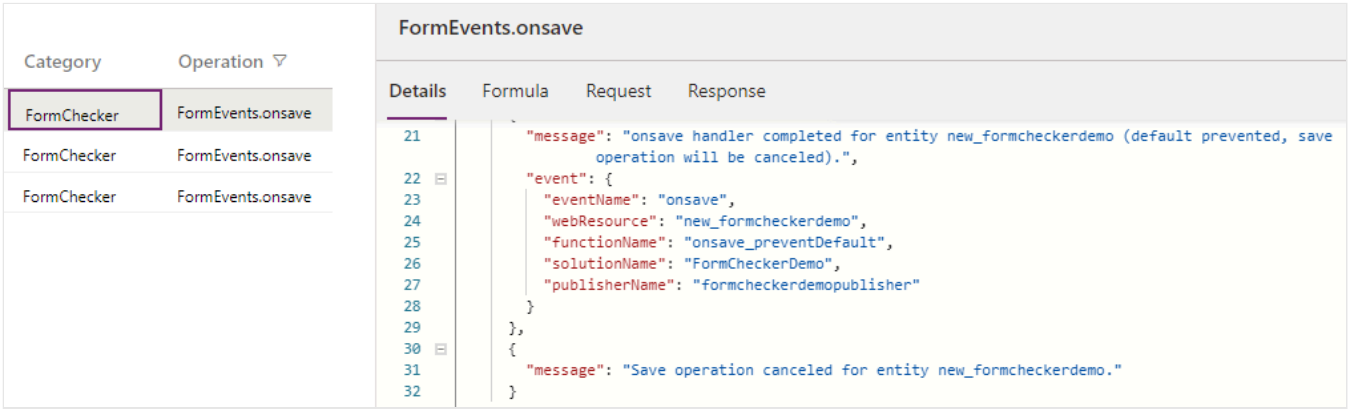
Follow up with the script owner to further troubleshoot the issue.

The form or record isn't saved when you try to save the form

A common cause is an OnSave event handler that calls the executionContext.getEventArgs().preventDefault() method to cancel the save operation.

How to troubleshoot

In Monitor, the `FormEvents.onsave` operation provides all the details of why the save event is canceled details than that are available from the form UI itself.



Follow up with the script owner to further troubleshoot the issue.

Form freezes, loads slowly, or throws unexplained errors

There are many possible reasons for a form to freeze, load slowly, or throw a "Web resource method doesn't exist" script error or an error that isn't a common script error. Some of the possible reasons include:

- Bad `OnLoad` scripts.
- Web resource controls.
- Ribbon button scripts and rules.
- Synchronous network requests.
- Custom plug-ins.
- Business process flow errors.

How to troubleshoot

Determine if the issue reproduces without involving forms. If it does, then there's a broader issue that should be investigated out of the form's context. Actual ownership of the problem depends on the particular details case by case.

- If you believe this issue only occurs on forms, see Use URL parameters to disable various form components to narrow down the component that's causing the issue.
- If you identify that certain form libraries/script files caused the issue, follow up with the owner who made these customizations to find out the root cause of the issue.
- If you identify that web resource controls cause the issue with the **DisableWebResourceControls** flag, then you can use the `DisableFormControl` flag to disable each one-by-one until the problem is longer reproduced. The last disabled control that doesn't reproduce the issue is the one that's causing the issue. Follow up with the owner of the control to further troubleshoot the issue.
- If you identify that the issue is caused by the command bar/ribbon with the **DisableFormCommandbar** flag, it means this isn't an issue with the form but an issue with the command bar. Use Command Checker to troubleshoot individual commands and identify which one is causing the issue.

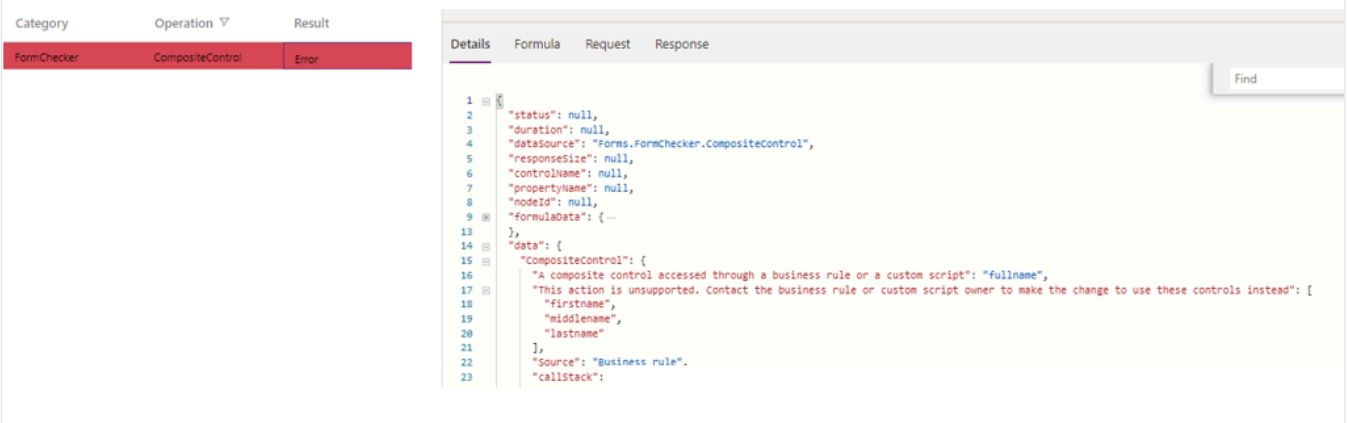
A business rule or custom script isn't working

This issue occurs if a business rule or custom script that used to work in the legacy web client stopped working in Unified Interface. One of the main reasons for this error to occur is when a business rule or script in Unified Interface references a control that isn't available in Unified Interface.

How to troubleshoot

One of the reasons that the business rule or script isn't working in Unified Interface is that the controls that are part of them don't exist in Unified Interface. Composite controls exist in the web client, but in Unified Interface composite control is broken down into parts and is stored differently. For example, if the column `fullname` is part of the business rule or custom script, columns `firstname`, `middlename`, or `lastname` should be used instead.

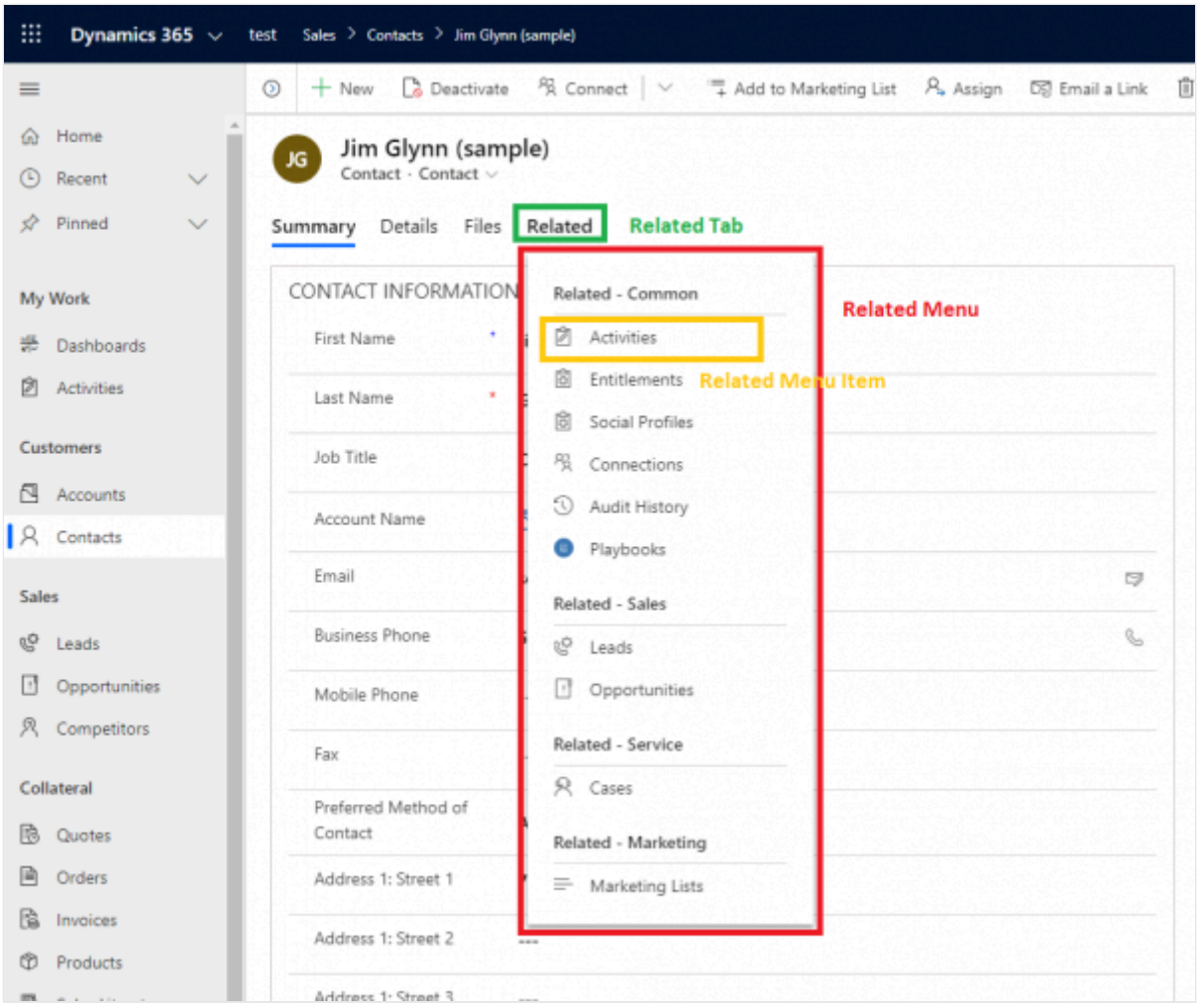
Once you launch form checker, you're able to see more details in the `CompositeControl` operation including the composite control that's causing the problem, the columns that can be used in the business rule or custom script instead and a full call stack (the call stack is modified for demonstration purposes).



Follow up with the corresponding owner of the business rule or custom script to change the control suggested by the form checker.

Related menu item doesn't appear in Related tab

Most forms have a **Related** tab. It opens the **Related menu** with **Related menu items**.



A **Related menu item** might not appear as expected.

How to troubleshoot

A related menu item might not appear because:

Relationship between the main and related table isn't configured correctly

There should be a one-to-many or many-to-many relationship between the main table and the related table. A form shows a row from the main table. The related table is the one that should appear in the **Related** menu of the form. If these relationships don't exist, the related menu item doesn't appear.

To verify, go to the Power Apps portal, select **Tables**, and select the table that has the relationships you want to view.

Dataverse created the relationship between the main and related table and isn't customizable

The **Related** menu doesn't show related tables from certain relationships created by Dataverse. These relationships are marked as noncustomizable.

The AssociatedMenuConfiguration.IsCustomizable property indicates whether the relationship can be customized. The easiest way to check is by querying the relationship using Web API to view the AssociatedMenuConfiguration complex type data.

Suppose you want to check whether the relationship between the Business Unit and Goal tables is customizable. The `SchemaName` of this relationship is business_unit_goal. Enter this URL in your browser:

HTTP

```
GET [Organization
URI]/api/data/v9.2/RelationshipDefinitions(SchemaName='business_unit_goal')/Micros
oft.Dynamics.CRM.OneToManyRelationshipMetadata?$select=AssociatedMenuConfiguration
```

You can also get the same data by querying the table definition:

HTTP

```
GET [Organization
URI]/api/data/v9.2/EntityDefinitions(LogicalName='businessunit')/OneToManyRelation
ships(SchemaName='business_unit_goal')/AssociatedMenuConfiguration
```

The response for either request might look like this:

JSON

```
{
    "@odata.context": "[Organization
URI]/api/data/v9.2/$metadata#RelationshipDefinitions/Microsoft.Dynamics.CRM.OneToM
anyRelationshipMetadata(AssociatedMenuConfiguration)/$entity",
    "MetadataId": "2124b4bd-f013-df11-a16e-00155d7aa40d",
    "AssociatedMenuConfiguration": {
        "Behavior": "UseCollectionName",
        "Group": "Details",
        "Order": null,
        "IsCustomizable": false,
        "Icon": null,
        "ViewId": "00000000-0000-0000-0000-000000000000",
        "AvailableOffline": true,
        "MenuId": null,
        "QueryApi": null,
        "Label": {
            "LocalizedLabels": [],
            "UserLocalizedLabel": null
        }
```

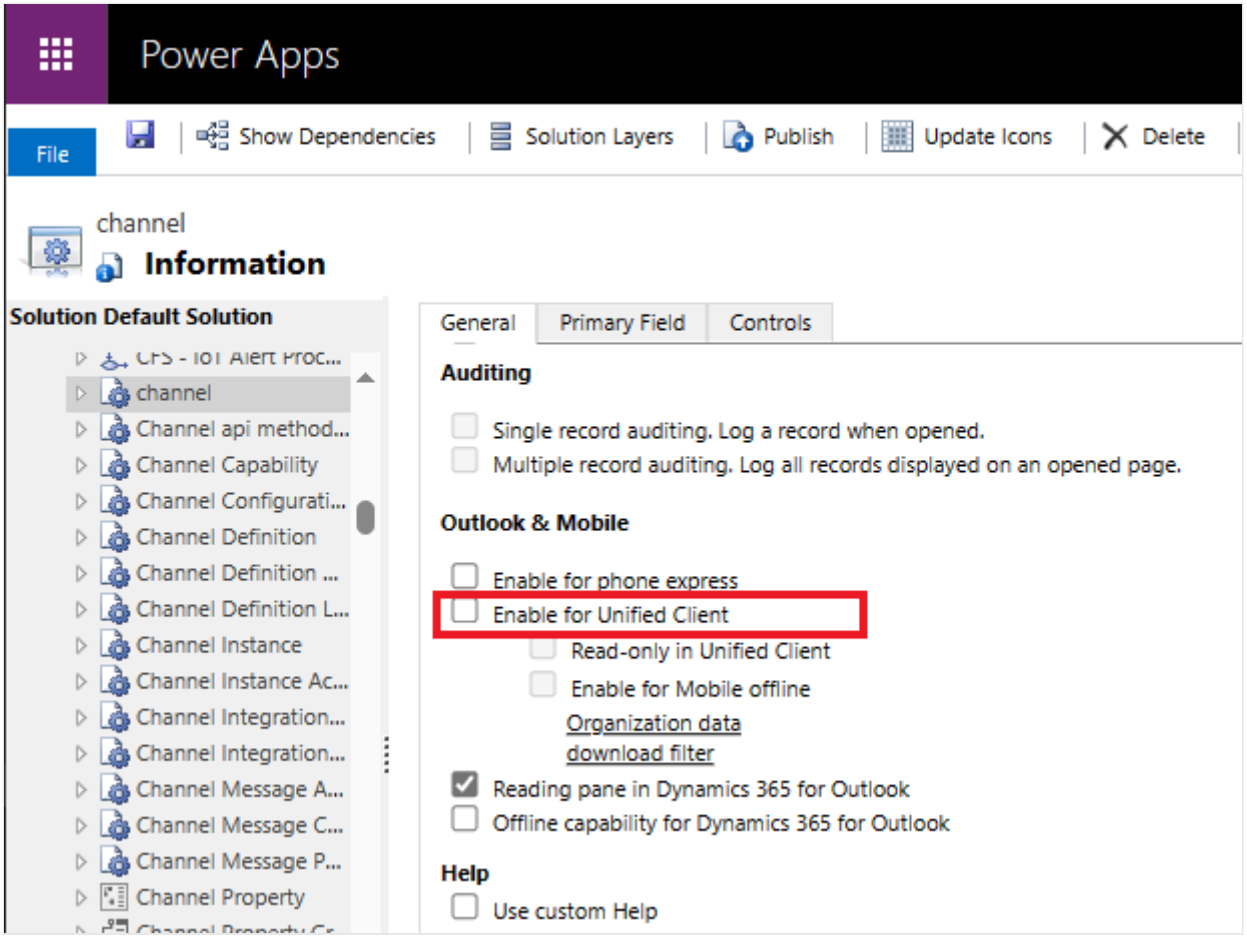
```
    }
}
```

Observe that `IsCustomizable` is `false`. Therefore, the relationship isn't customizable and **Goal** doesn't appear in the **Related** menu.

Related table isn't enabled for Unified Client

If the table was created in Web Client (deprecated since 2019), it might not appear because it's disabled for Unified Client.

To verify, go to Solution explorer and select the table (entity). Ensure that **Enable for Unified Client** is checked.



Tables created with the modern designer don't have this issue. They're always enabled for Unified Client.

> ⓘ **Note**
>
> Certain system tables can't be enabled for Unified Client. For example, **Process Session** can't be used in model-driven apps.

Audit History doesn't appear in Related tab

Audit History isn't in the Related menu.

How to troubleshoot

Audit history isn't supported in these cases:

- Tables that aren't enabled for auditing. Check the table's IsAuditEnabled property to confirm.
- System tables that don't support Audit History
- Mobile apps
- Offline mode
- Dynamics for Outlook

Unexpected related menu item appears in related tab

A **Related menu item** might appear when it shouldn't.

How to troubleshoot

A related menu item might appear because:

Related table has self-referencing many-to-many relationship with the main table

The system ignores form XML customizations for self-referencing many-to-many relationships. The system ignores these customizations because it isn't possible to indicate whether the customizations are for the primary table or the related table, which are both the same table in this case. Therefore, the system ignores these customizations.

If you modify the form XML to hide the related menu item, it still appears. Any form XML customizations for self-referencing relationships are ignored, like changing the order or the label of the related item.

Some system tables can't be hidden

For example, custom tables always show **Activities** related menu item. It isn't possible to hide it with the form designer or by modifying the form XML.

Related menu items aren't translated as expected

The text for related menu items aren't in the expected language.

How to troubleshoot

If some related menu items are displaying in a different language from the user's language, check whether the form XML is missing translated labels.

Check the form XML to see if there are labels defined for each language. For example, this form XML shows that the `navContacts` item only has a US English (1033) label: `Contacts`.

XML

```
<NavBarByRelationshipItem Id="navContacts" Area="Sales" Sequence="10064"
RelationshipName="contact_customer_accounts" Show="true">
  <Titles>
    <Title LCID="1033" Title="Contacts" />
  </Titles>
</NavBarByRelationshipItem>
```

To fix this issue, add the translated labels to the form XML. For example, this form XML shows that the `navContacts` item with both US English (1033) and German (1031) labels.

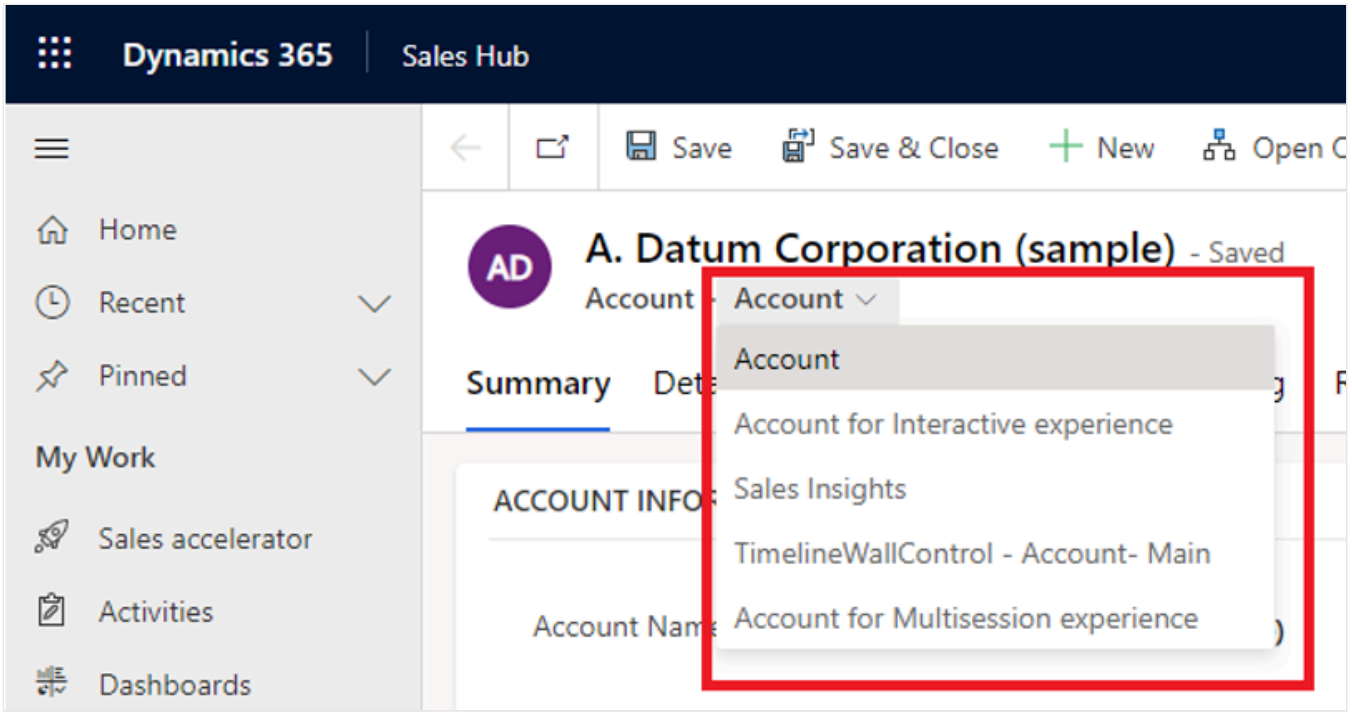
XML

```
<NavBarByRelationshipItem Id="navContacts" Area="Sales" Sequence="10064"
RelationshipName="contact_customer_accounts" Show="true">
  <Titles>
    <Title LCID="1033" Title="Contacts" />
    <Title LCID="1031" Title="Kontakte" />
  </Titles>
</NavBarByRelationshipItem>
```

If the text for the user's language isn't defined, the system uses the organization's base language. If that doesn't exist either, the system uses the US English text.

Why is a form showing or not showing in the form selector?

The form selector is a dropdown that allows users to switch between different forms for a particular table.



You need to understand the conditions that control whether the form is displayed.

How to troubleshoot

A form is available in the selector when all of these conditions are met:

1. User has permission to access the form.
2. Form is added to the app module.
3. Form isn't hidden with Client API.
4. For Dynamics Customer Service workspace forms, **ShowInFormSelector** is set to **True**.

If a form isn't showing in the form selector,

1. Check the affected user's security roles.
2. Check if the form is added to the app module.
3. Disable custom scripts.

Why an expected default form isn't shown by default?

When there are multiple forms for a table, the desired one isn't used as the default.

How to troubleshoot

The following criteria determine the first form shown to the user:

1. If a formId is provided when opening a form, then that form is shown. A formId can be provided through Client API functions like openForm or in a URL.
2. Otherwise, the most recent form that the user picked is shown. The system remembers the latest selection from the form selector.
3. If the user hasn't used the form selector before, the system uses the form order.

If the form isn't available to the user, the system continues to find an appropriate form to show.

A form is available to the user when:

1. User has permission to access the form.
2. Form is added to the app module.

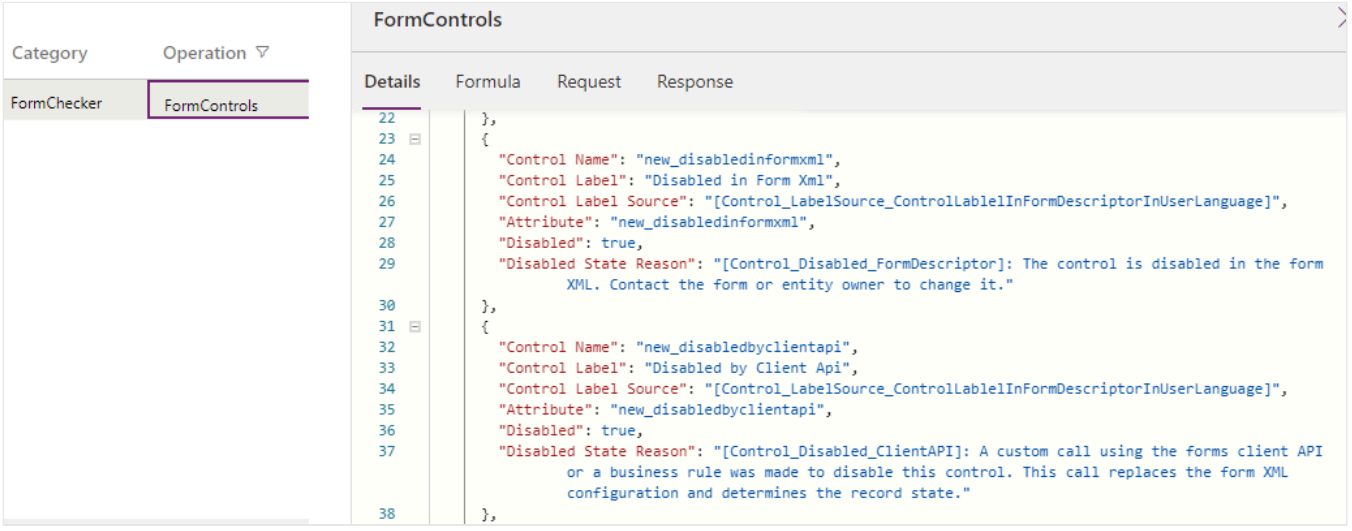
If no forms are available to the user, then the the fallback form is used.

Why a control is disabled/enabled or visible/hidden

There are many possible reasons why a control might be disabled or hidden when the form is loaded.

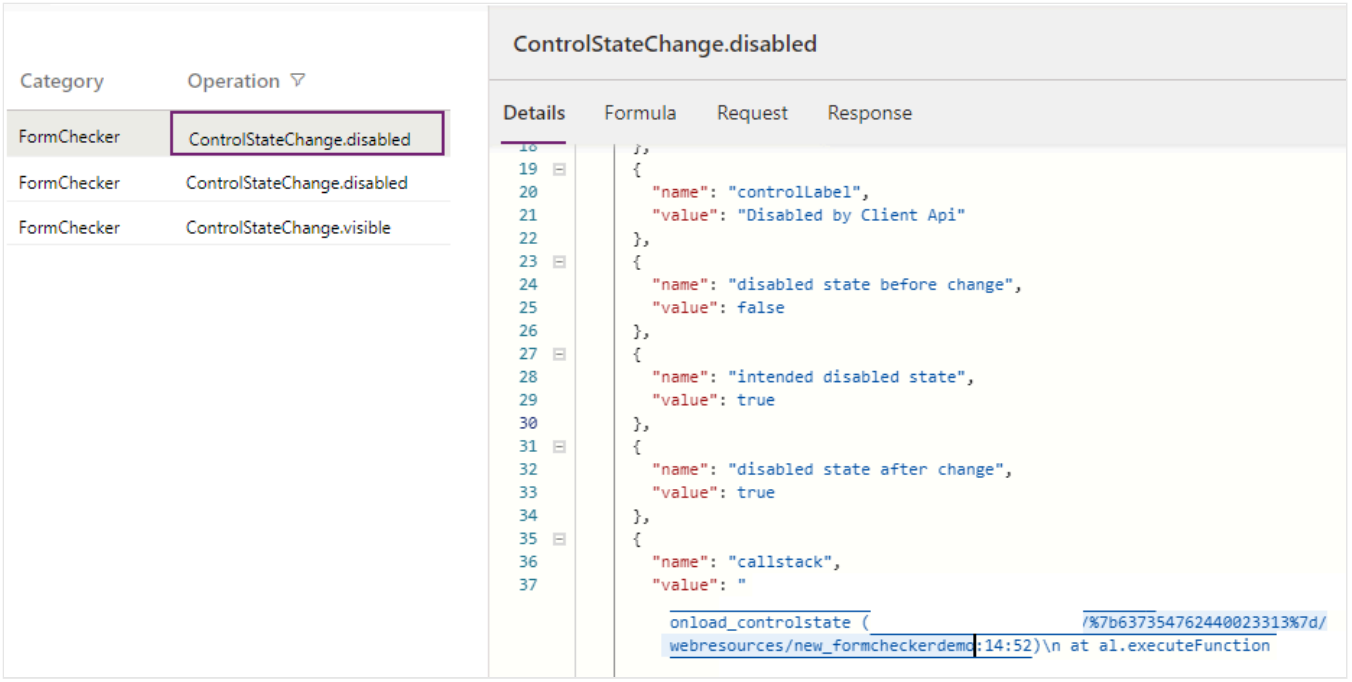
How to troubleshoot

You can use Monitor to view the `FormControls` operation that includes all the details about the initial control state when the form loads.



Another place to check is the `ControlStateChange.visible` or `ControlStateChange.disabled` operation that explains why the control disable or visible state is changed at any time on the form. This operation explains the control state before the change, intended state change that

might succeed, and the state after the change. Not all control state change attempts are successful. For a control disabled by form XML, you can enable it through client API in an `OnLoad` event handler. However, if control is disabled for security reasons, it's highly unlikely an attempt to enable it through client API would successfully change the state.



A control can be disabled by using the following list of rules in order. If a rule is met, then the following rules are ignored. If you want to change whether a control is disabled, you must change the input to the rule used for the result or to a rule earlier in the list.

- If the flags `DisableWebResourceControls=true` or `DisableFormControl=<control name>` are passed and the control is affected by these flags, the control is disabled.
- If the owning table is read-only in Unified Interface in table definitions, the control is disabled.
- If the table isn't available in offline mode, the control is disabled.
- If the current user doesn't have write permissions on the record, the control is disabled.
- If the column definition has `IsValidforCreate` set to false, the control is disabled.
- If the column definition has `IsValidforUpdate` set to false, the control is disabled.
- If the current user doesn't have `Assign to` privilege, the owner column is disabled.
- If the user doesn't have write permissions on the column defined by field-level security, the control is disabled.
- If a client API script enables or disables the control, the control disabled state honors that setting.
- If the control is disabled in the form designer, the control is disabled.
- If the user doesn't have `Append To` privilege for the lookup control's table, or `Append` privilege on the current record's table, the lookup control is disabled

Finally, if the control passes all the above checks, the record state determines whether the control is disabled. The control is enabled by default on active records and disabled on inactive records.

> **Note**
>
> The difference between `FormControls` and `ControlStateChange` is that the `FormControls` operation reflects the initial control state when the form is loaded, while the `ControlStateChange` operation reflects the state change at any time on the form, whether it's during form load, in OnChange or OnSave events after the form is loaded.

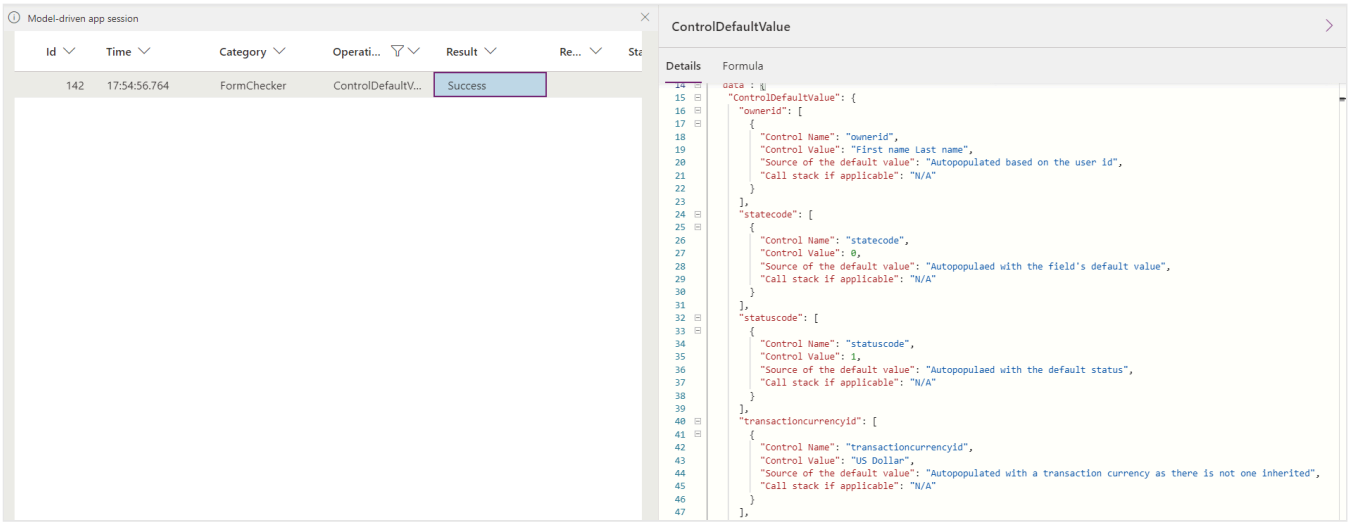
> **Important**
>
> A control's disabled and hidden state can change multiple times when a form is first loaded. To know the reason why a control is hidden or disabled, make sure to check the **last** operation logged in the monitor. For example, if there are no `ControlStateChange.visible/ControlStateChange.hidden` operations for the control being investigated, the value and reasoning will be in the `FormControls` operation. Otherwise, it's the value and reason in the **last** `ControlStateChange.visible/ControlStateChange.hidden` operation. You can order logs by timestamp to search for the last operation.

Why a control has a certain value on form load

A control might not have a specific value on form load as the user expected.

How to troubleshoot

There are many possible reasons why control can have a value when a form loads. The `ControlDefaultValue` operation in Monitor explains the source of the default values.



If multiple updates are happening to a control's value, an `Update Sequence` indicates the final value. For example, here's a control with a default value and then overridden with a value passed with a client API script. There's a call stack provided.



There are scenarios where columns are populated based on a relationship column mapping, in which case the event shows that.

```
"ControlDefaultValue": {
  "transactioncurrencyid": [
    {
      "Control Name": "transactioncurrencyid",
      "Control Value": "US Dollar",
      "Source of the default value": "Relationship field mapping",
      "Call stack if applicable": "N/A"
    }
  ],
  "address1_city": [
    {
      "Control Name": "address1_city",
      "Control Value": "Renton",
      "Source of the default value": "Relationship field mapping",
      "Call stack if applicable": "N/A"
    }
  ],
```


Verify where the value is coming from and take action based on the following table:

Expand table

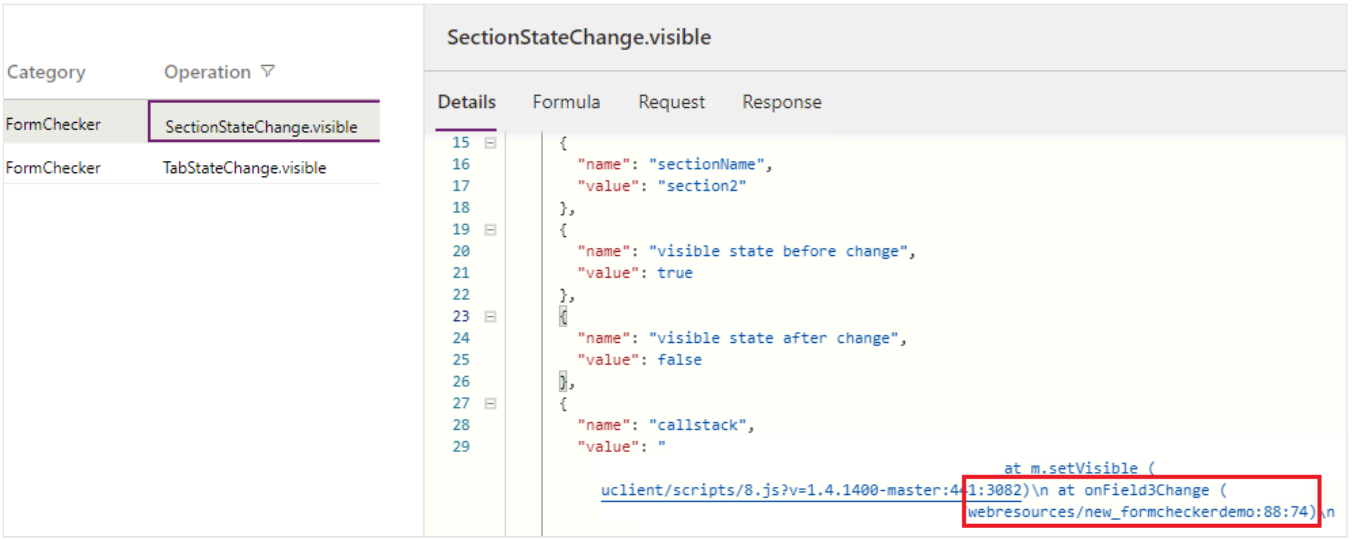
Source	How to fix
Client API script	Contact the script owner.
Default value	Check the control's configuration.
Relationship column mapping	Check the relationship configuration and update the column mapping.
Value passed by page input data passed via URL	Check the API that opens the specific form with the issue, it's passing the value.

Why a tab or section is visible or hidden

There are many possible reasons why a tab or section might be hidden or visible.

How to troubleshoot

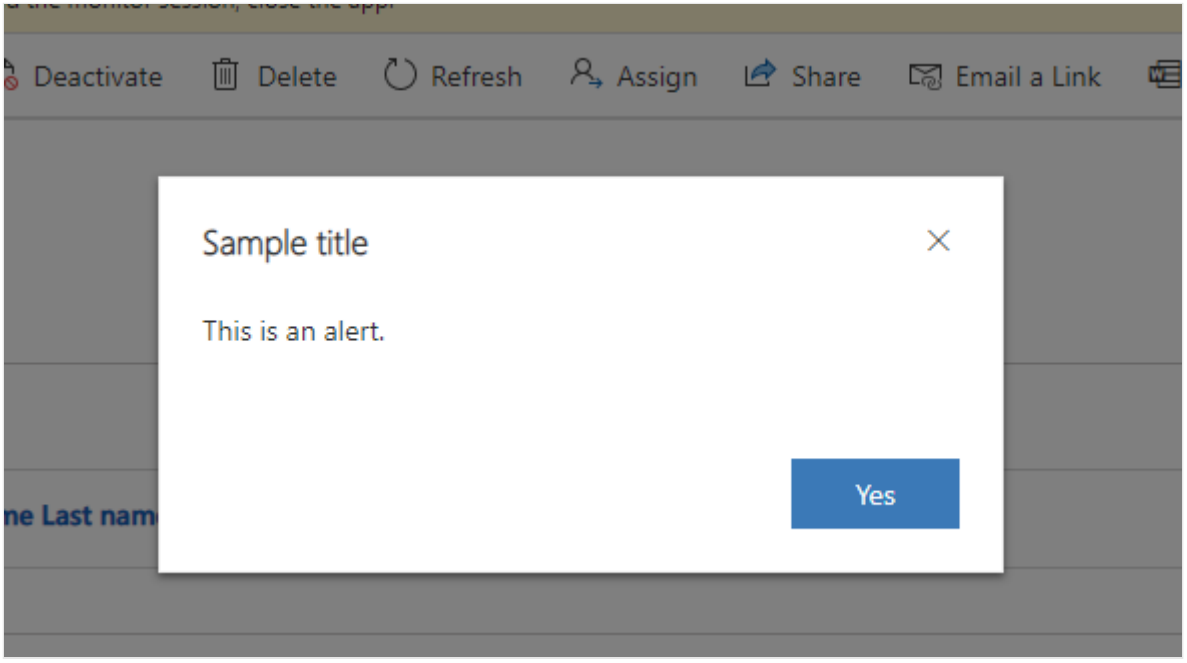
The `TabStateChange` or `SectionStateChange` operations in Monitor explain the visible state change, as shown in the following image. If a script causes it, then the call stack would reveal the web resource file, line number, and function name that caused this behavior.



Follow up according to the suggestion in the state reason or the owner of the web resource/business rules to change or fix the behavior.

Unexpected dialogs or navigation

There are many possible reasons why a dialog appears, or navigation happens unexpectedly. One of the common causes is the Xrm.Navigation API methods are called to open a record or a form by a custom script. For example, when you open a form, an alert appears, as shown in the following image.



How to troubleshoot

The `XrmNavigation` operation in Monitor contains call stack that helps you identify the script that's causing unexpected behavior.



Follow up with the owner of the web resource to change or fix the behavior.

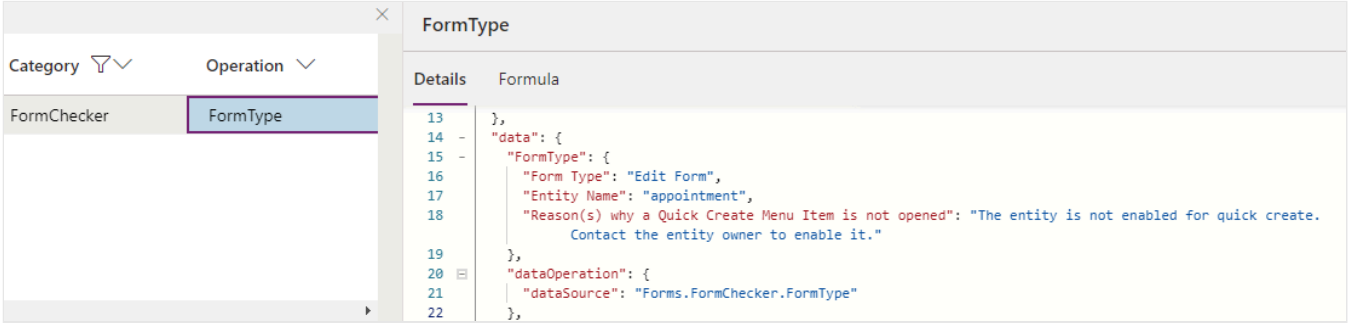
Opening another form instead of a quick create form

When opening a quick create form from a lookup or a grid, another form might open (edit or main form) instead of a quick create form. There are few reasons why this issue can happen:

- The main form dialog force flag is being set.
- Quick create form isn't available.

How to troubleshoot

You can use Monitor to view the `FormType` operation that includes all the reasons why a quick create form isn't opened.



You need to follow up with the table owner who has disabled quick create through table definitions (metadata).

Table doesn't appear in the quick create menu flyout

When opening the global quick create menu, not all tables are available. Some possible reasons:

- Table isn't enabled for quick create. Check if the table supports quick create and that it's enabled.
- Table isn't added to the app. Ensure all required components are added to an app.
- No quick create form is available for the table. Create a quick create form.
- Table is read-only.
- User doesn't have permission to access the quick create form. Check if the user has appropriate security roles.
- User doesn't have create privilege for the table.
- Table doesn't support create privilege.

How to troubleshoot

You can use Monitor to view the `QuickCreateMenu` operation, which includes all the tables and reasons why they're filtered from the quick create menu flyout.

See the following examples to understand the reasons for filtering. Based on the explanations, contact the responsible party or make changes accordingly.

```
"post": [
  "The entity is not enabled for quick create. Contact the entity owner to enable it.",
  "The entity is not enabled for Unified Client. Contact the entity owner to enable it.",
  "The entity's mobile client visibility cannot be modified. Contact the entity owner to fix it."
],
```

```
"new_a1": [
  "There is no quick create form available for the entity. Contact the entity owner to make sure there is a quick create form added."
]
```

```
"new_a6": [
  "The entity is not part of the app module. Contact the app owner to add the entity."
],
```

Unexpected unsaved changes message

When working on forms, you get the *unsaved changes* message on the form footer when you navigate from the current form or save the form without any changes.

How to troubleshoot

The *unsaved changes* error appears when you change the form and when the changes aren't saved. If you didn't make any changes manually, they could come from a JavaScript, plug-in, or

business rule. You can use Monitor to view the `UnsavedChanges` operation that helps find the source of the changes. You can filter by OperationType `UnsavedChanges`.

The `all attributes modified` section includes a quick summary of the columns causing the unsaved changes error and their values. The `unsaved changes` section shows what happened to the columns in detail. For every column, you can see a list of controls that could be causing a change. The value change is also displayed (previousValue, newValue), and a call stack.

The following screenshot shows the root cause of the issue. You can see that the change came from the `OnLoad` script.


```
"data": {
  "UnsavedChanges": [
    {
      "Reported from": "Form footer"
    },
    {
      "All attributes modified": {
        "Account": {
          "fax": 123,
          "name": "New account name"
        }
      }
    },
    {
      "Unsaved changes": {
        "Account": {
          "fax": [
            {
              "message": "Attribute with name 'fax' has been modified. The change could be coming from any of the below controls:",
              "controlNames": [
                "fax"
              ],
              "previousValue": null,
              "newValue": 123,
              "callStack":  at onload  /webresources/new_onload:3:33)  at Object.runFormOnLoadScripts
            }
          ],
          "name": [
            {
              "message": "Attribute with name 'name' has been modified. The change could be coming from any of the below controls:",
              "controlNames": [
                "name"
              ],
              "previousValue": null,
              "newValue": "Account name",
              "callStack": "The change was manually done by the user."
            },
            {
              "message": "Attribute with name 'name' has been modified. The change could be coming from any of the below controls:",
              "controlNames": [
                "name"
              ],
              "previousValue": "Account name",
              "newValue": "New account name",
              "callStack": "The change was manually done by the user."
            }
          ]
        }
      }
    }
  }
```


> **Note**
>
> If the user has manually made the changes on the form, a call stack won't be provided.

Verify where the change is coming from and if the behavior is expected or not. If a script causes the change, the original web resource can be traced back in the call stack. In most cases,

it's a script. Make a decision based on the web resource itself.

Business required column doesn't block saving

Business required columns is a usability feature that helps prevent users from saving a record with an empty value in that column. In model-driven apps and Power Pages, the following scenarios don't block saving a record when a required column has an empty value:

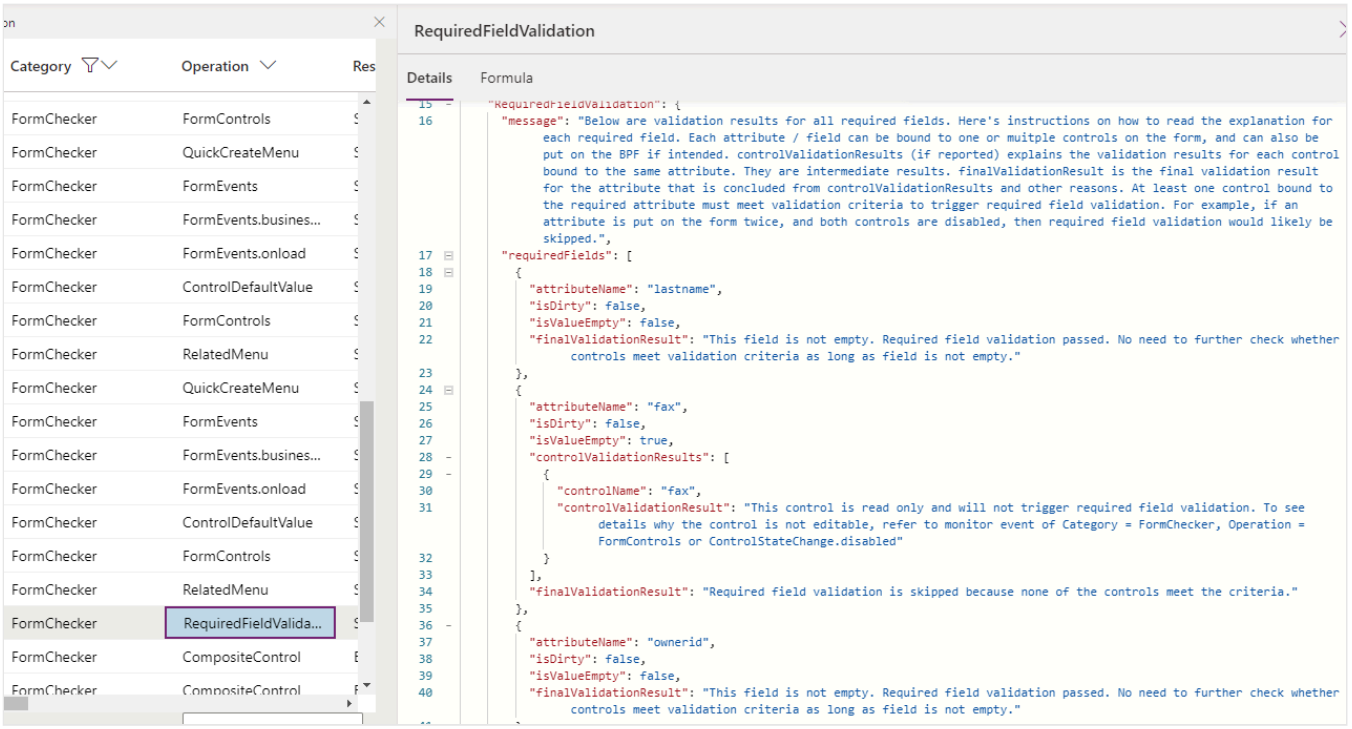
- The column is hidden from the form, either because of column properties or a client-side script using the control.setVisible Client API .
- The column is on a hidden form tab or section.
- A client-side script changes the column's required level using the setRequiredLevel Client API.
- The user isn't using model-driven apps or Power Pages to create the record. Client applications using Dataverse APIs aren't blocked from saving records when column AttributeMetadata.RequirementLevel is set to AttributeRequiredLevel.ApplicationRequired. Learn more about column requirement level

When you need to enforce data integrity, you should use entity business rules and other server-side validation instead, such as synchronous plug-ins.

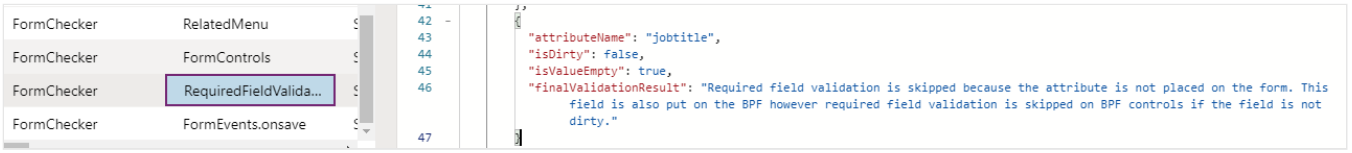
How to troubleshoot

The `RequiredFieldValidation` operation is logged when a save is attempted, regardless of whether save is successful or not. This operation explains why each business required column blocks or doesn't block the save operation.

The following image is an example of this operation. The message explains how to read the detailed reports of each required column. In this example, `fax` column is bound to one control, and the control of the same name is read-only. Therefore it won't trigger required column validation.



The following image is another example that `jobtitle` is a business required column on the business process flow but not on the form, and the column isn't modified. Thus it doesn't block the save operation even when it's empty.



Follow up

Most times, the behavior is by design, and the `RequiredFieldValidation` operation explains why this column behaves in a certain way in form save operation. If the required column validation is skipped on a column because the control is disabled or hidden, as the first example illustrated, see the form checker suggestions for further analysis.

This might lead to another troubleshooting scenario such as Why a control is disabled/enabled or visible/hidden.

Can't create a record because of insufficient permissions to a secured field, even though that field isn't in the form

This might happen when users create a record (row) from a different form. They get the error message **The user does not have create permissions to a secured field** even though they haven't entered a value for that field (column), or that field isn't on the form.

When table **A** has a lookup field to table **B**, creating a record **A** from record **B** automatically sets lookup fields on it to **B**, even if those fields aren't on the form.

For example:

1. Account table has a lookup field, `primarycontactid`, to the contact table.
2. User opens a contact form for **Robin Danielsen**.
3. User opens the lookup field, `parentaccountid`, on the form and selects the button to create a new account.
4. New account form opens, with the `primarycontactid` field automatically set to **Robin Danielsen**.

If the `primarycontactid` field is secured and the user doesn't have permissions to edit it, they get an error when they try to save the new account. They can clear the field before saving it. However, if that field isn't on the form, they can't clear it. A workaround is to create the account from the account page instead of from a contact form.

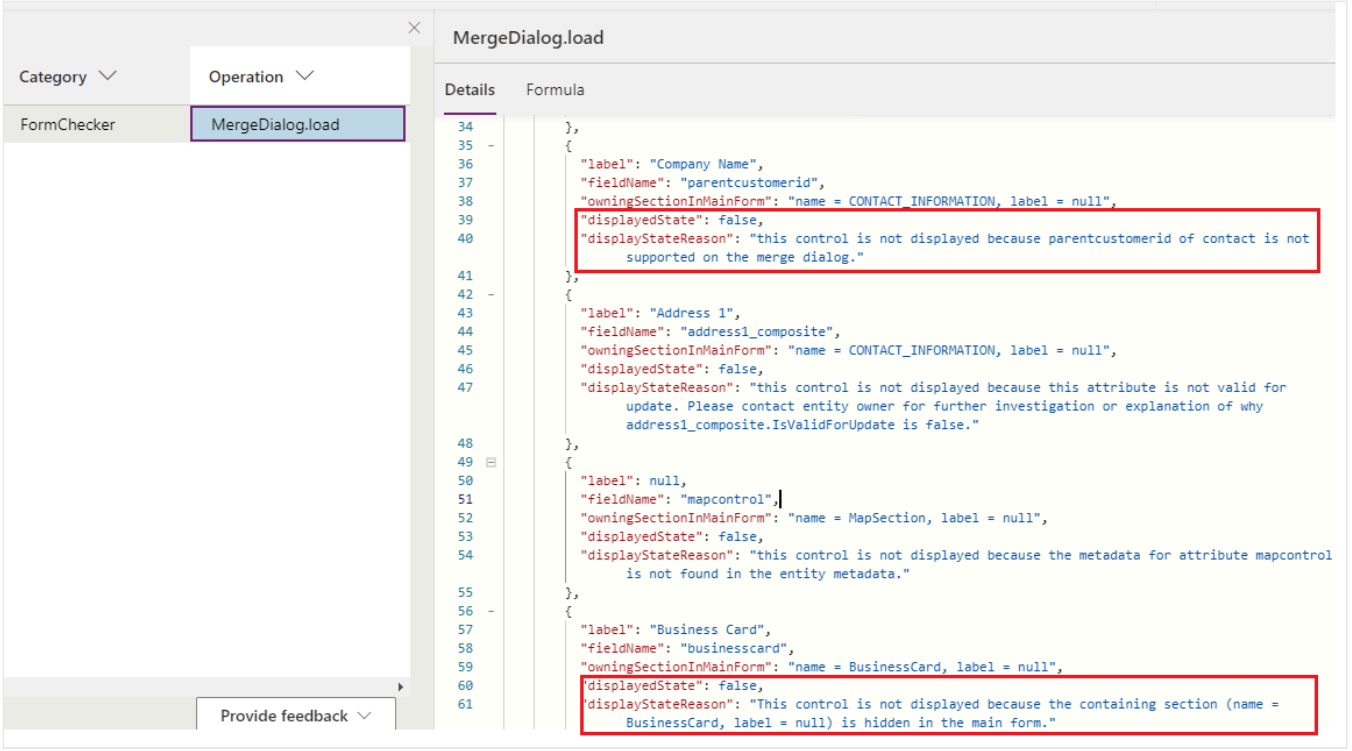
Some columns aren't displayed on the merge dialog

The merge dialog uses the default main form definition for the table and selectively renders most, but not all the columns in the dialog. This Form Checker operation explains why some of the columns aren't displayed on the merge dialog, even that they might be displayed on the main form.

How to troubleshoot

The following `MergeDialog.load` operation explains the reason why some columns aren't displayed.

In this example, `parentcustomerid` column on the contact form isn't supported in the merge dialog. The business card column isn't displayed because the containing section is hidden in the main form XML definition. Even though showing the owning section in the main form via client API is possible, the merge dialog honors the form XML configuration as event handlers aren't supported on the merge dialog.


Category
Operation
FormChecker
MergeDialog.load
Provide feedback
MergeDialog.load
Details
Formula
},
{
"label": "Company Name",
"fieldName": "parentcustomerid",
"owningSectionInMainForm": "name = CONTACT_INFORMATION, label = null",
"displayedState": false,
"displayStateReason": "this control is not displayed because parentcustomerid of contact is not supported on the merge dialog."
},
{
"label": "Address 1",
"fieldName": "address1_composite",
"owningSectionInMainForm": "name = CONTACT_INFORMATION, label = null",
"displayedState": false,
"displayStateReason": "this control is not displayed because this attribute is not valid for update. Please contact entity owner for further investigation or explanation of why address1_composite.IsValidForUpdate is false."
},
{
"label": null,
"fieldName": "mapcontrol",
"owningSectionInMainForm": "name = MapSection, label = null",
"displayedState": false,
"displayStateReason": "this control is not displayed because the metadata for attribute mapcontrol is not found in the entity metadata."
},
{
"label": "Business Card",
"fieldName": "businesscard",
"owningSectionInMainForm": "name = BusinessCard, label = null",
"displayedState": false,
"displayStateReason": "This control is not displayed because the containing section (name = BusinessCard, label = null) is hidden in the main form."

Collaborative troubleshooting using Live Monitor

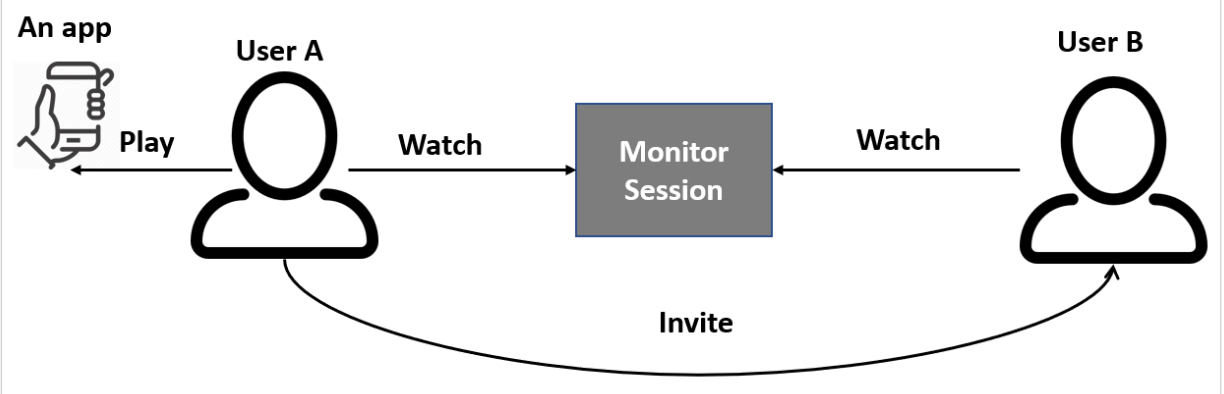
Article • 11/19/2024

Live monitor offers two features to enable collaborative troubleshooting and debugging of canvas and model-driven app problems in Power Apps:

- **Invite** enables makers to share a monitoring session by inviting other members within the same organization to the same Live monitor session. For example, a maker troubleshooting a performance problem with Power Apps can invite a member from the technical support team to concurrently view all the events flowing through the same troubleshooting Live monitor session.
- **Connect user** enables makers and support teams to share a link with users to open a published app and connect it to a Live monitor session so that support teams can watch the sequence of events generated by the user's interaction.

Invite user

With canvas apps and published model-driven apps, makers can share a real-time Live monitor session with others, such as support technicians, to help investigate problems. When you invite other users to your Live monitor session, you can quickly collaborate and debug an app together, without the need to share your screen. When you invite others to participate in a session, they see the exact same app events in their own browser without having to open the app or reproduce the specific scenario that you're debugging. You and other participants can browse, view, and inspect the app events independently, without stepping on each other or handing control back and forth to identify the problem.



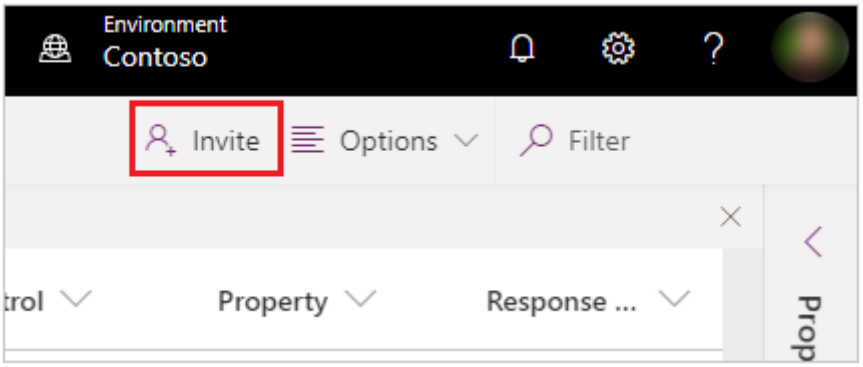
When using the **Invite** feature for a published app:

1. User A plays the published app from Power Apps.
2. User A invites User B to a Live monitor session.
3. User B joins the Live monitor session with the link that User A sent.
4. User A interacts with the app.
5. User B watches the events generated from User A's interaction with the app in their own Live monitor window.

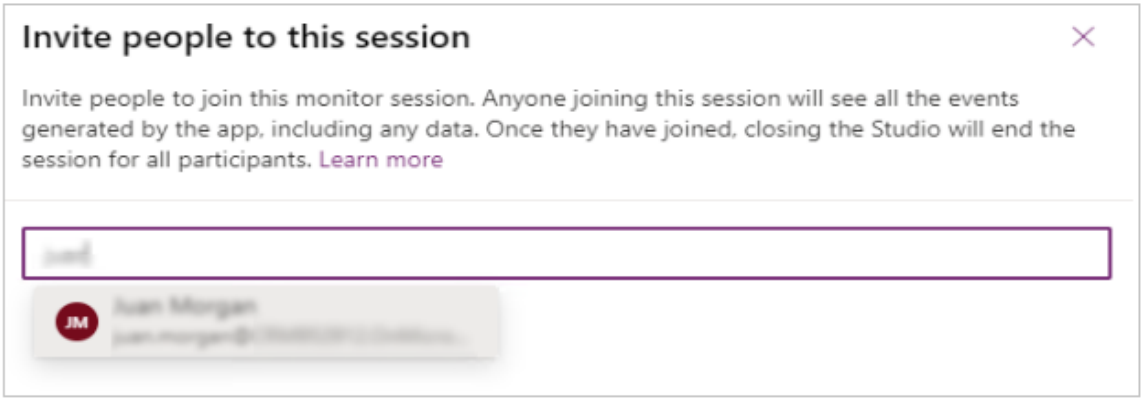
> **Note**
>
> You can invite any user in your organization to a Live monitor session. Users joining the session will see all the events generated by the app, including any data.

To invite a user to watch a monitoring session

1. Select **Invite** from the command bar.



2. Enter the Microsoft Entra user name or alias of the user whom you want to invite to the Live monitor session.

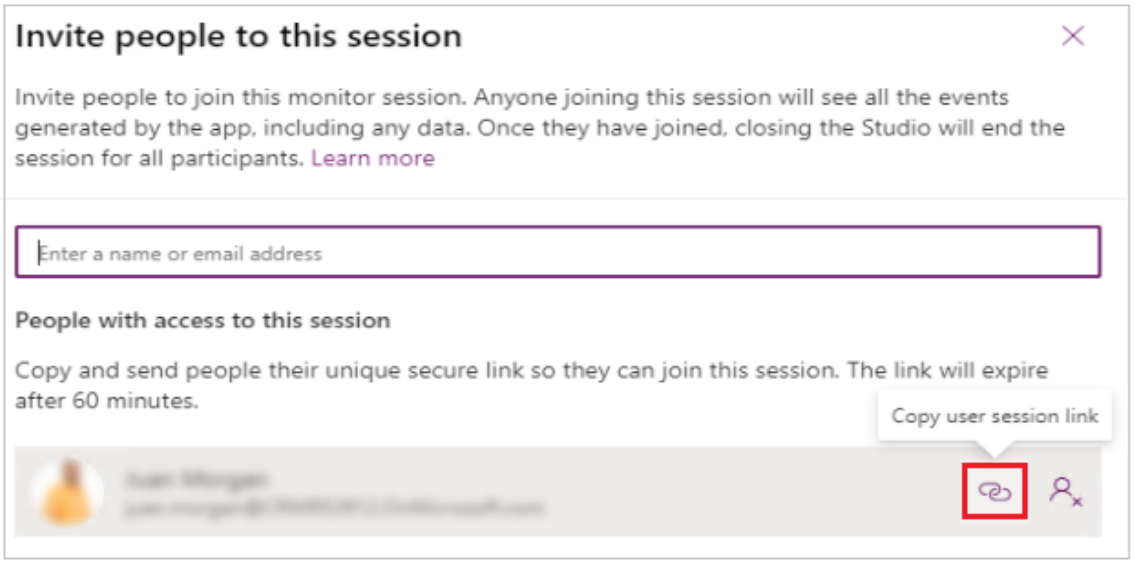


3. Select the user for whom you want to generate a link to the current Live monitor session.

> **Note**

The link is unique for each user. It can't be shared between users. The link expires after 60 minutes.

4. Select the link icon to copy the session link, and then send it to the user you've invited to the session.

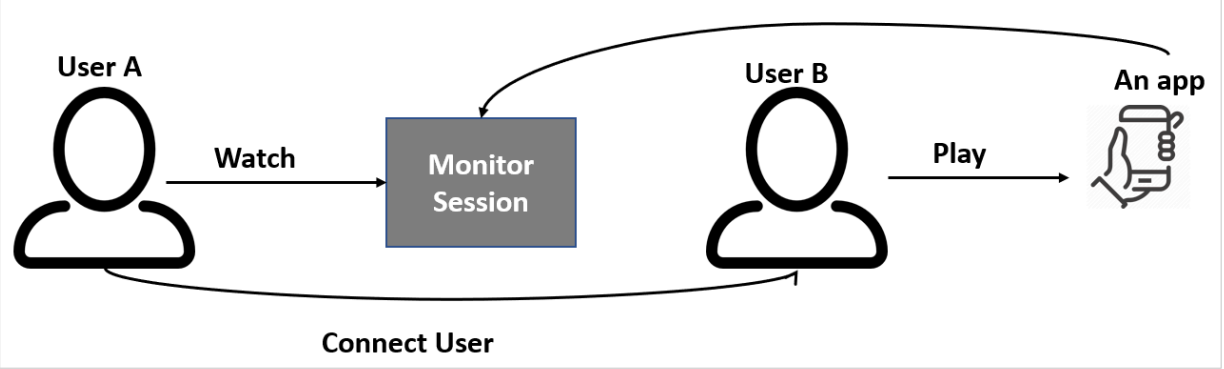


Repeat steps 2 through 4 for each user you want to invite.

The users who have received the link can use it to open the Live monitor and connect to your Live monitor session.

Connect user

With **Connect user**, a maker can enable a user to open a published canvas or model-driven app and connect by using a monitoring session. You can watch the sequence of events the user generates while interacting with the app, and help with problem investigation.



ⓘ Note

- With **Connect user**, makers and support teams share a link with users to open a published app and connect it to a Live monitor session, so support teams can watch the sequence of events generated by the user's interaction with the app.
- **Connect user** requires that the app be published with Power Apps version 3.20042 or later for canvas apps and client version 1.4.5973-2307.2 or later for model-driven apps. To make this feature available in earlier versions of a canvas app, publish the app again.

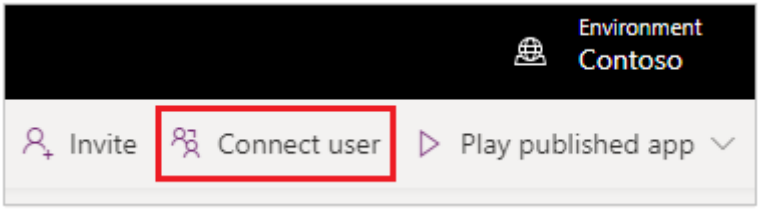
When using the **Connect user** feature for a published app:

1. User A chooses the app to monitor and opens Live monitor.
2. User A selects **Connect user**, and share the generated session link with User B.
3. User B plays the published app by selecting the link that was shared by User A.
4. User A watches the monitored session as User B interacts with the app.

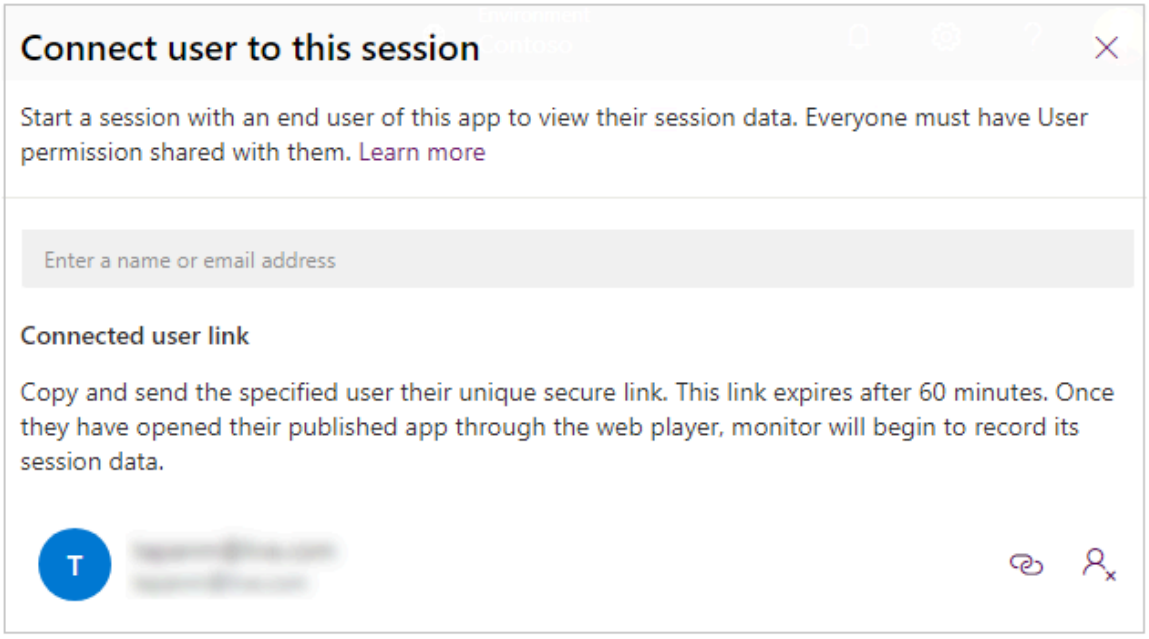
To connect a user to a monitoring session

1. Open Live monitor for the published app.

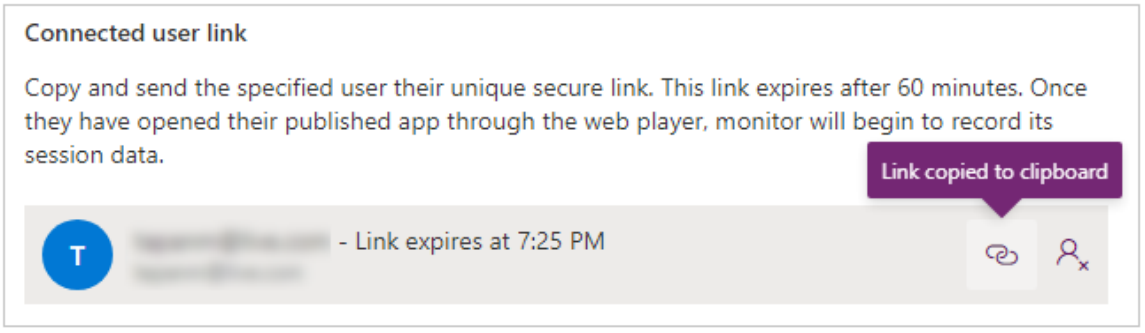
2. Select **Connect user**.



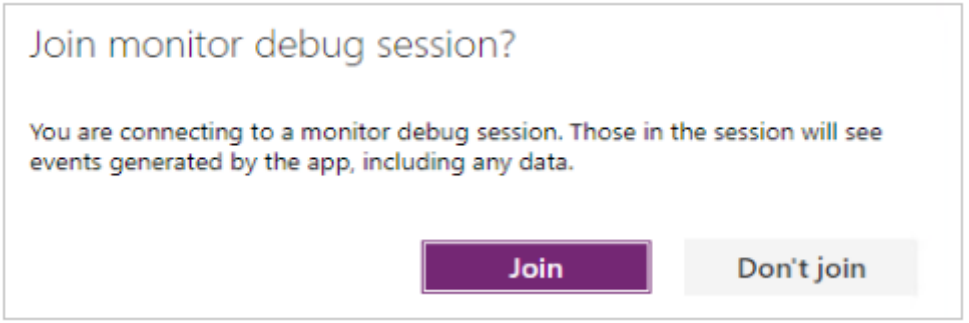
3. Search for and select the user who you want to connect to.



4. Copy the link, and share it with the user so they can play the app.



5. After you share the link, the user selects it to open the app and is notified that they're joining a monitoring session. The same link can be used to open the app in a monitored session on Power Apps mobile.



The user can now interact with the app, and Live monitor will show events for this interaction.

Next steps

Advanced monitoring

See also

Troubleshoot form issues in model-driven apps
Debugging canvas apps with Live monitor
Debugging model-driven apps with Live monitor

Feedback

Was this page helpful? Yes No

Provide product feedback

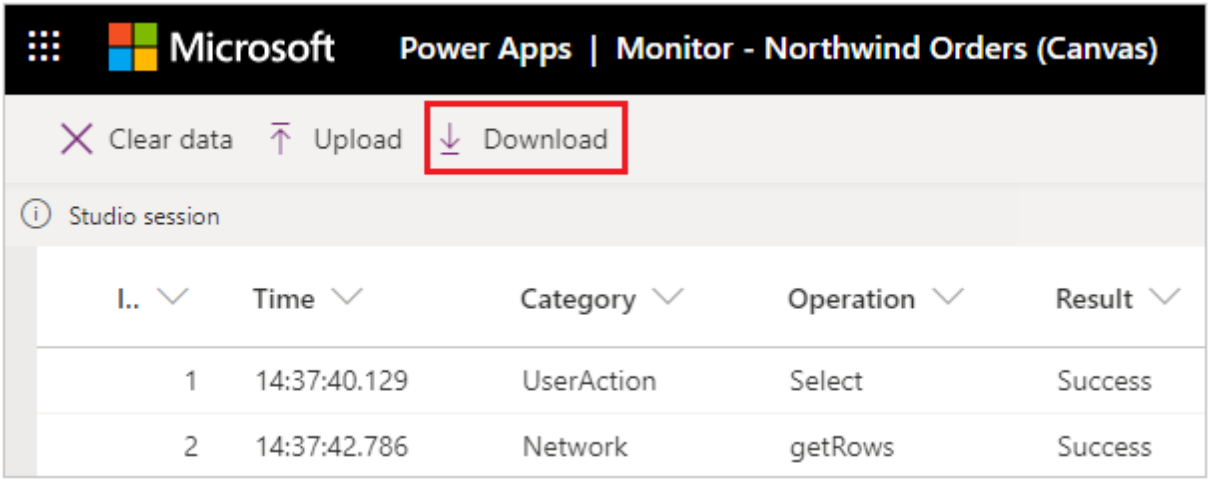
Advanced monitoring concepts

Article • 11/19/2024

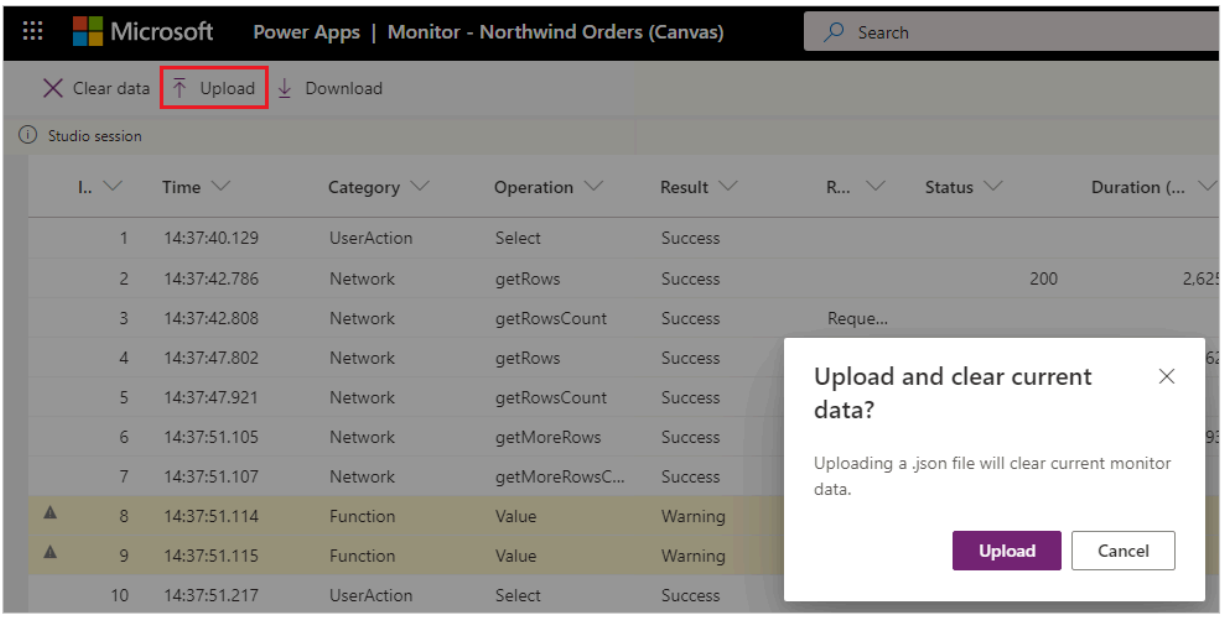
In this article, you'll learn about downloading and uploading trace files in Live monitor, events supported by Live monitor, and scenarios that Live monitor doesn't support.

Download and upload trace files

You can download the events listed in Supported events for offline analysis. Events can be downloaded in .json or .csv format, and you can share them with others. The .csv files can only be downloaded, not uploaded, but if you download the events in .json format, you can upload them later into Live monitor for analysis. You can also attach a trace file to support service requests, which can help speed up getting the solution to your problem.



Select **Upload** to load a .json trace file to Live monitor. This option removes any events currently displayed in the Live monitor table and replaces the details with events contained in the trace file.



Supported events

Live monitor supports several events for canvas apps and model-driven apps. The following table includes examples of the supported event categories and types.

Expand table

Canvas apps	Model-driven apps
• Data connectors • Network events (error status codes highlighted) • Screen load metrics • Cross-screen dependency warning • User actions such as *Navigate*, *Select*, and *SetProperty* • Custom Trace() • Delegated versus non-delegated queries • Verbose switch (internal telemetry) • Delegation • Function • Network • Parsing • Performance • Scenario • ScreenLoad • Telemetry • UserAction • Verbose	• Form events, such as load and save • Network • Page navigation • Command executions

Unsupported scenarios for Live monitor

Live monitor doesn't support the scenarios in the following table.

Expand table

Description	App type
Live monitor connected to a canvas app embedded in a model-driven app or custom page.	Canvas
Live monitor connected to a SharePoint custom form app.	Canvas
Live monitor connected to a Microsoft Teams embedded app. As an alternative, you can play the app in a web player for diagnostics purposes.	Canvas

See also

Debugging canvas apps with MonLive monitor
Debugging model-driven apps with Live monitor

Feedback

Was this page helpful? Yes No

Provide product feedback

Common issues and resolutions for Power Apps

Summarize this article for me

Summary

This article provides solutions to common problems that you might encounter when you create or use Power Apps. The article organizes problems by functional area, such as connectors, Power Automate integrations, Power Fx, and Power Apps Studio. Each section includes troubleshooting steps and links to related documentation.

Before you troubleshoot specific problems, review Power Apps troubleshooting strategies for guidance on how to identify the source of a problem. That article outlines key principles for debugging both functional and performance problems.

Common Power Apps issue areas

- Connectors and delegation
- Power Automate Integration
- Power Fx
- Regional performance issues
- Power Apps Studio and Forms
- Browser performance
- Power Apps for Windows

Connectors and delegation

For an overview of how tabular and action connectors work, see Connections.

For details about delegation, see Understanding delegation in a canvas app.

For a description of how to monitor the data being sent and returned, see Debugging canvas apps with Monitor.

- **You don't have the correct permissions to use a connection.**

  Users might encounter this message in two situations:

  - The application shares an implicit connection that isn't a secure implicit connection. To resolve the problem, share the connection with the end user, but consider that this

approach isn't recommended. The author should convert all connections to secure implicit connections.

  - The application uses a secure implicit connection. In this case, republishing the app might resolve the problem.

- **Automatic Next links for galleries and grids don't work for action-based connectors.**

  Action-based connectors don't support next links. Next links are properties on query results that allow a gallery or grid to automatically load the next set of results. For more information, see Overview of connectors for canvas apps.

- **Sharing a Canvas app using SharePoint connector**

  For more information about how to share an app for SharePoint, see Connect to SharePoint from a canvas app.

- **SQL data sources no longer add a `[dbo]` prefix to the data source name.**

  For more information about this change, see Connect to SQL Server from Power Apps.

- **Custom connectors and Microsoft Dataverse**

  You need to make adjustments if you built your app with Power Apps that:
  - Is version 2.0.540 or earlier.
  - Relies on a Dataverse database.
  - Uses at least one custom connector from a separate environment.

  First, deploy the custom connector to the same environment as the database. Then, update the app to use the newly deployed connector. Otherwise, a dialog notifies users that the API wasn't found. For more information, see Environments overview.

- **Column names with spaces**

  If you're using a list created using Microsoft Lists, a SharePoint library, or an Excel table in which a column name contains a space, use single quotes with the column name, for example `someList.'Color Tag'`.

- **Apps that connect to on-premises SharePoint**

  If you share an app that relies on connections that aren't automatically shared (for example, an on-premises SharePoint site), users who open the app in a browser see a dialog box with no text when they select or tap **Sign in**. To close the dialog box, select or tap the close (X) icon in the upper-right corner.

The dialog box doesn't appear if you open the app in Power Apps Studio or Power Apps mobile. For more information about shared connections, see Share app resources.

- **For apps that are created from data, only the first 500 records of a data source can be accessed.**

  Power Apps works with large data sources by delegating operations to the data source. For operations that can't be delegated, Power Apps shows a warning at authoring time and operates on only the first 500 records of the data source. For more information, see the Delegation overview.

- **Excel data must be formatted as a table.**

  For information about limitations when you use Excel as a data source, see Cloud-storage connections.

- **Microsoft Lists is supported but not SharePoint libraries, some types of list columns, or columns that support multiple values or selections.**

  For more information, see SharePoint Online.

- **Sign-in issues on certain Android mobile devices when using authenticator**

  On certain devices and in certain scenarios, you might experience sign-in failures when you use an authenticator. This problem happens because of a limitation that the OEM imposes. For more information, see ADALError: BROKER_AUTHENTICATOR_NOT_RESPONDING.

- **Apps cannot save null/blank values to data sources**

  If your app can't save null or blank values to data sources, the **Formula-level error management** feature might be disabled. To fix this problem, go to **Settings** > **Updates** > **Retired** and make sure the **Formula-level error management** feature is enabled.

Power Automate integration

- **Power Automate flows are orphaned in Power Apps.**

  Power Automate flows that you add by using an older version of the Power Apps panel might become orphaned and removed. To fix this problem, re-add the flows manually.

- **Power Apps custom pages (in a model-driven app) are out of synchronization with embedded Power Automate flow metadata.**

The metadata for a Power Automate flow might get out of synchronization with a model-driven app's custom page if you update the flow after embedding it. To update the metadata, follow these steps for each embedded flow:

1. Edit the custom pages that use the flow.
2. Open the Power Automate pane and refresh the flow.
3. Save and republish the custom page.

When you're done, follow these steps:

1. Edit the model-driven app.
2. Make a minor change to trigger the save option.
3. Save and publish the model-driven app.

Power Fx

For more information about Power Fx, see Microsoft Power Fx.

- **`Connection.Connected` returns the wrong value during OnStart in Power Apps for Windows.**

  While offline, the **Connection.Connected** formula might wrongly return **true** immediately after starting an app in Power Apps for Windows. As a workaround, use a Timer control to delay the execution of the logic depending on it.

- **Issues with Date-time**

  For more information about date and time problems, see:
  - Troubleshoot Canvas app date time issues
  - Troubleshoot Model driven app date time issues

Regional performance

- **Performance degradation when opening Power Apps Studio in China**

  Power Apps Studio might take more than 30 seconds to load in China. This problem doesn't affect tenants that 21Vianet hosts locally.

Power Apps Studio and forms

The Power Apps Studio hosts the app editing and publishing experience.

- **Problems with startup**

If you have trouble accessing or starting Power Apps, see Troubleshooting startup or sign-in issues for Power Apps.

- **Problems changing dimensions or orientation of SharePoint forms**

  If you have problems with the **Screen size + orientation** settings for custom SharePoint forms, use the **Custom** size to work around the problem:

  1. Reset the setting by selecting the **Small** size.
  2. Toggle **Orientation** to **Portrait** and then back to **Landscape**.
  3. Select **Custom** and enter a desired screen size. For reference, the preset values are:
     - "Width: 270, Height: 480" for the Small Portrait size.
     - "Width: 720, Height: 480" for the Small Landscape size.

- **Copying and pasting screens across apps**

  Copying and pasting screens across apps isn't currently supported. To work around this problem:

  1. Add a new screen to your target app.
  2. Copy the controls from the screen in your source app.
  3. Paste the controls into the screen of your target app.

- **Changing the layout of SharePoint forms**

  When you customize SharePoint forms in certain languages, if you try to change the layout from **Portrait** (default) to **Landscape**, the app might show multiple errors (yellow triangles in controls). To resolve these errors and retain the landscape layout, select **Undo**.

- **Changing a flow in a shared app**

  If you add a flow to an app and share the app, and then make changes to the flow, such as adding a service or modifying a connection, more steps are required. You must also first remove the flow from the shared app. Next, re-add the flow to the app. Finally, re-share the app.

- **Changing a "Title" field in a table**

  If you change the **Title** field for a table that other tables reference through one or more lookups, an error occurs when you try to save the change. To work around this problem:

  1. Remove any lookups to the table you want to change.
  2. Change the table's **Title** field.
  3. Re-create the lookups.

  For more information about lookups, see Create a relationship between tables.

- **When Power Apps generates an app from data, the field used for sorting and searching isn't automatically configured.**

  To configure the field, edit the Items formula for the gallery, as described in Filter and sort a gallery.

- **It can sometimes take a moment before a newly shared app can be used.**

  In some cases, a newly shared app isn't immediately available. Wait a few moments, and it should become available.

- **In the Form control, you can't change data by using a custom card.**

  The stock custom card is missing the Update property, which is required to write back changes. To work around this problem:

  1. Select the **Form** control, and insert a card by using the right-hand pane based on the field that you want the card to show.
  2. Unlock the card, as described in Understanding data cards.
  3. Remove or rearrange controls within the card as you see fit, just as you would with the custom card.

- **Card gallery is deprecated.**

  Existing apps that use this feature continue to run for now, but you can't add a card gallery. Replace card galleries with the new Edit form and Display form controls.

Browser performance

- **Browser running out of memory**

  If you're using the 32-bit version of Google Chrome or Microsoft Edge and the browser runs out of memory while you're using Power Apps, consider using the 64-bit version.

Power Apps for Windows

- **Power Apps mobile app for the Windows platform doesn't support the Dropbox connector.**

  A pop-up dialog shows the following message in this situation:

  > We can't connect to the service you need right now. Check your network connection or try again later.

When this issue occurs, consider using a web player on the Windows platform.

- **Microsoft Entra Conditional Access with the policy *Require device to be marked as compliant* doesn't work in Power Apps for Windows.**

  When the Conditional Access policy is set to *Require device to be marked as compliant* in Microsoft Entra ID, users receive the following sign-in error and can't access their Power Apps.

  > The application contains sensitive information and can only be accessed from devices or client applications that meet your enterprise management compliance policy.

  To work around the issue, they can use a browser.

Next steps

If your issue isn't listed in this article, you can search for more support resources, or contact Microsoft support. For more information, see Get Help + Support.

Last updated on 02/26/2026

Get session and app ID details

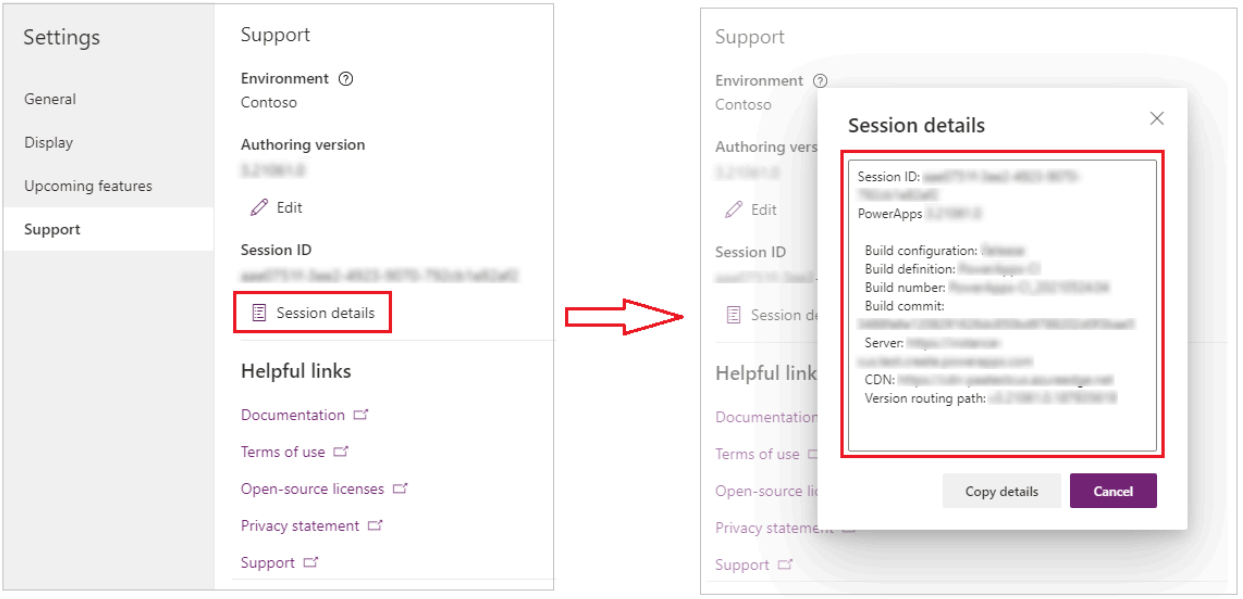
07/30/2025

If you find a problem with an app in Power Apps, you can help Microsoft troubleshoot the problem much more effectively with a session ID, an app ID, or both.

Get the session ID for a canvas app

When editing a canvas app

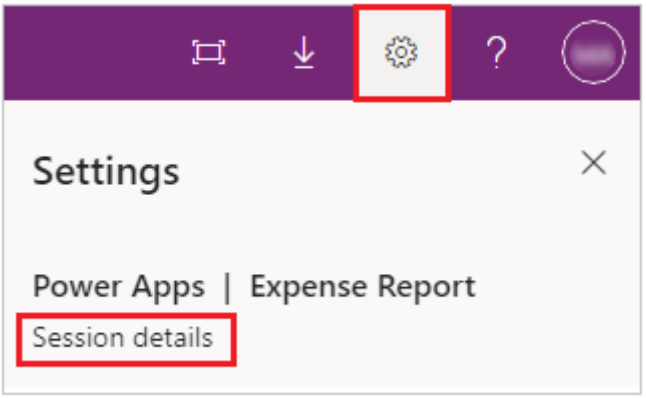
1. Select **Settings**.
2. Select **Support** > **Session details**.



3. Select **Copy details** to copy all session to clipboard.

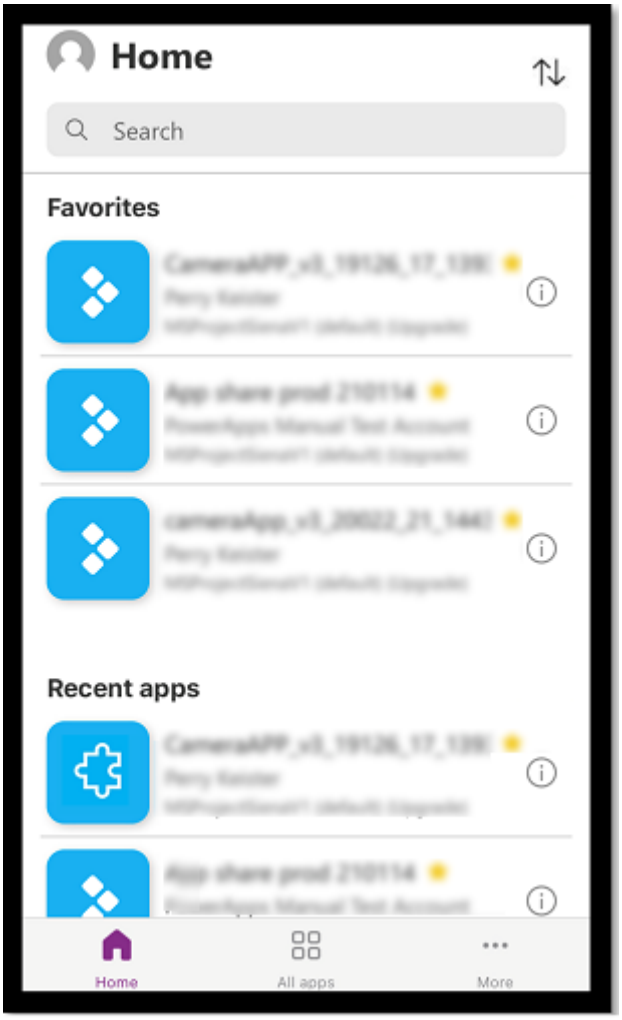
When running a canvas app in a browser

1. In the upper-right corner, select the **Settings** icon .
2. Select **Session details**.

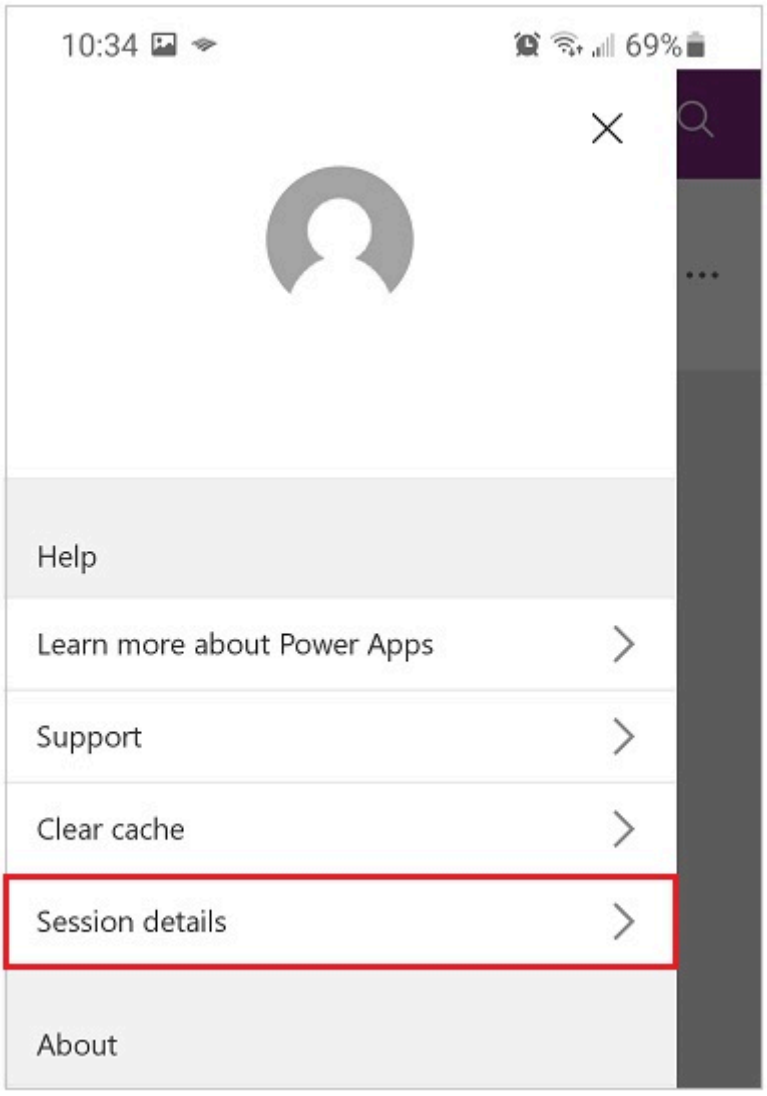


When running a canvas app on a phone or a tablet

1. In the upper-left corner, select your profile picture.



2. Select **Session details**.

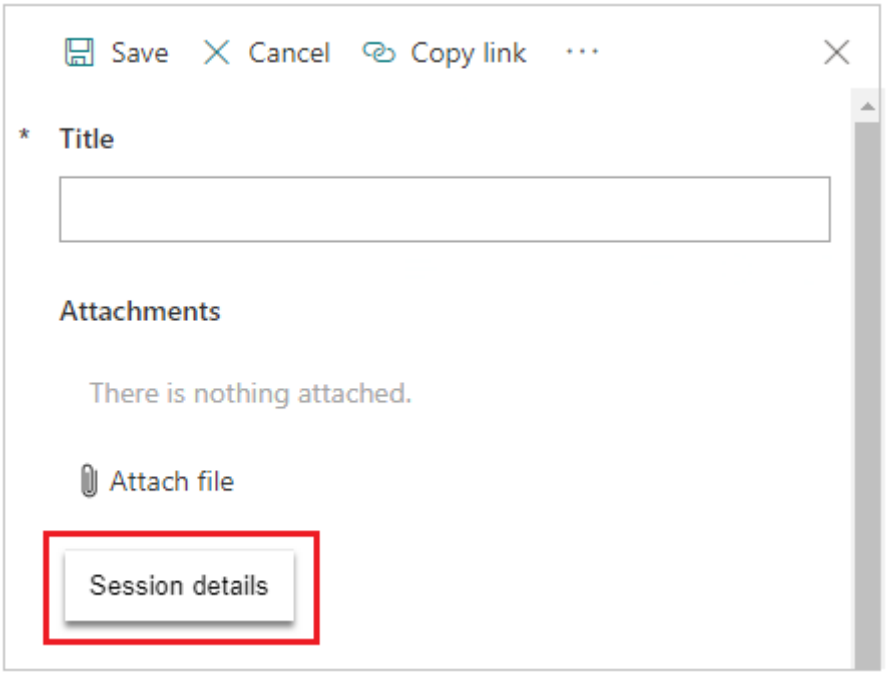


When running an embedded canvas app or form

1. Do one of these steps:

    - While holding down the Alt key, right-click the app or form.
    - Tap the app or form with two fingers for 1-2 seconds, and then release.

2. Select **Session details**.



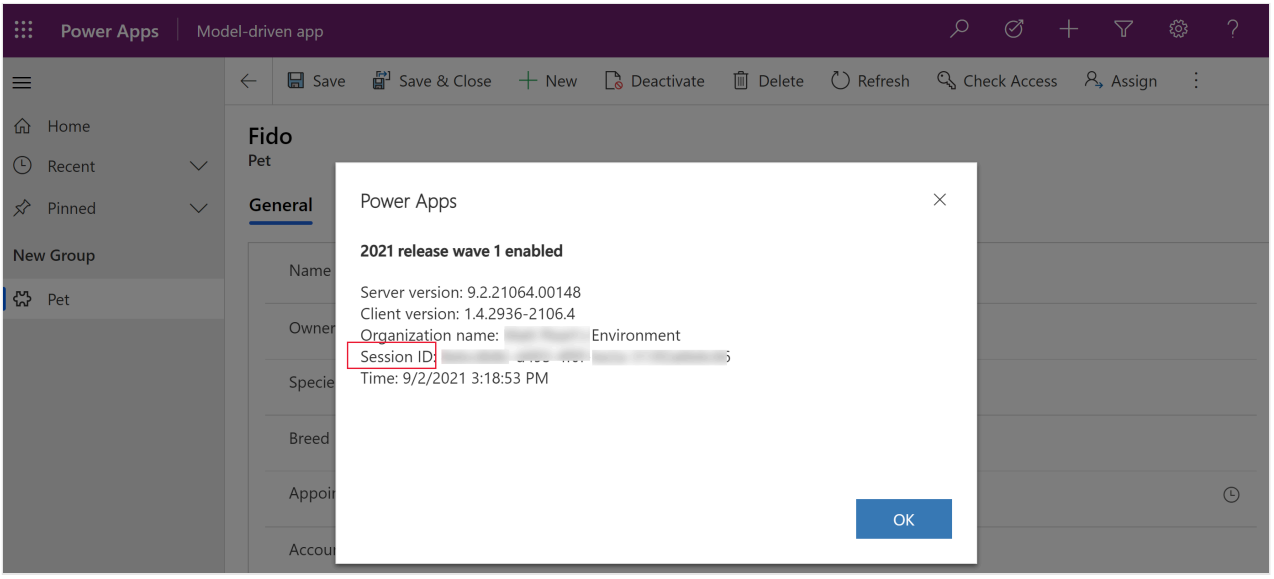
Get the session ID for a model-driven app

When editing a model-driven app

1. From the preview pane in app designer, on the command bar in the model-driven app displayed, select **Settings** (gear).
2. Select **About**.

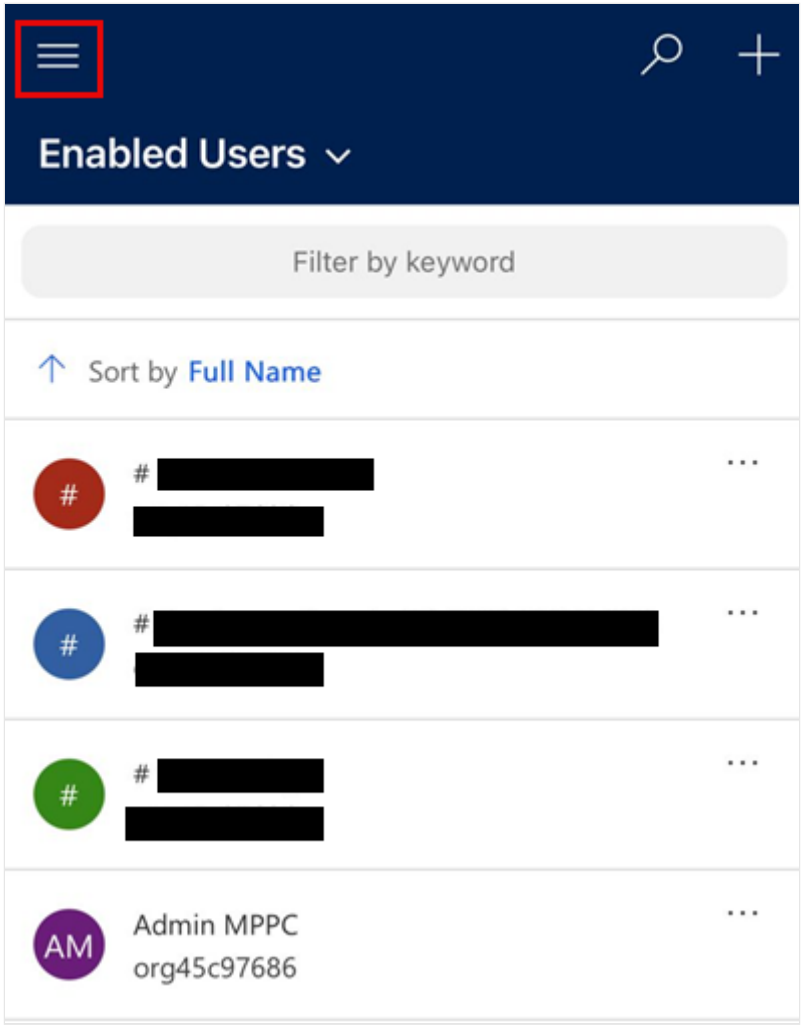
When running a model-driven app in a browser

1. On the command bar in a model-driven app, select **Settings** (gear).
2. Select **About**.

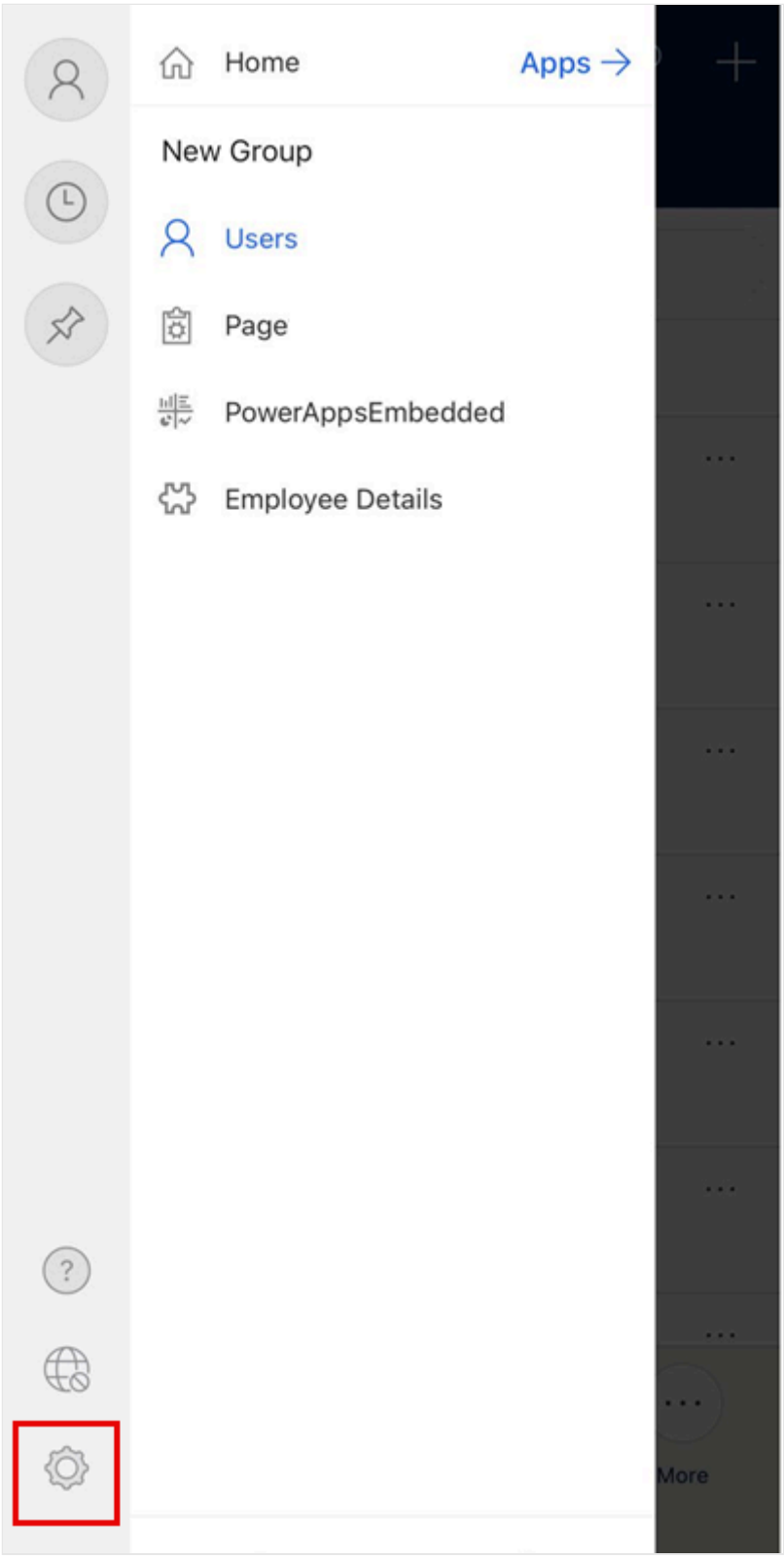


When running a model-driven app on a phone or a tablet

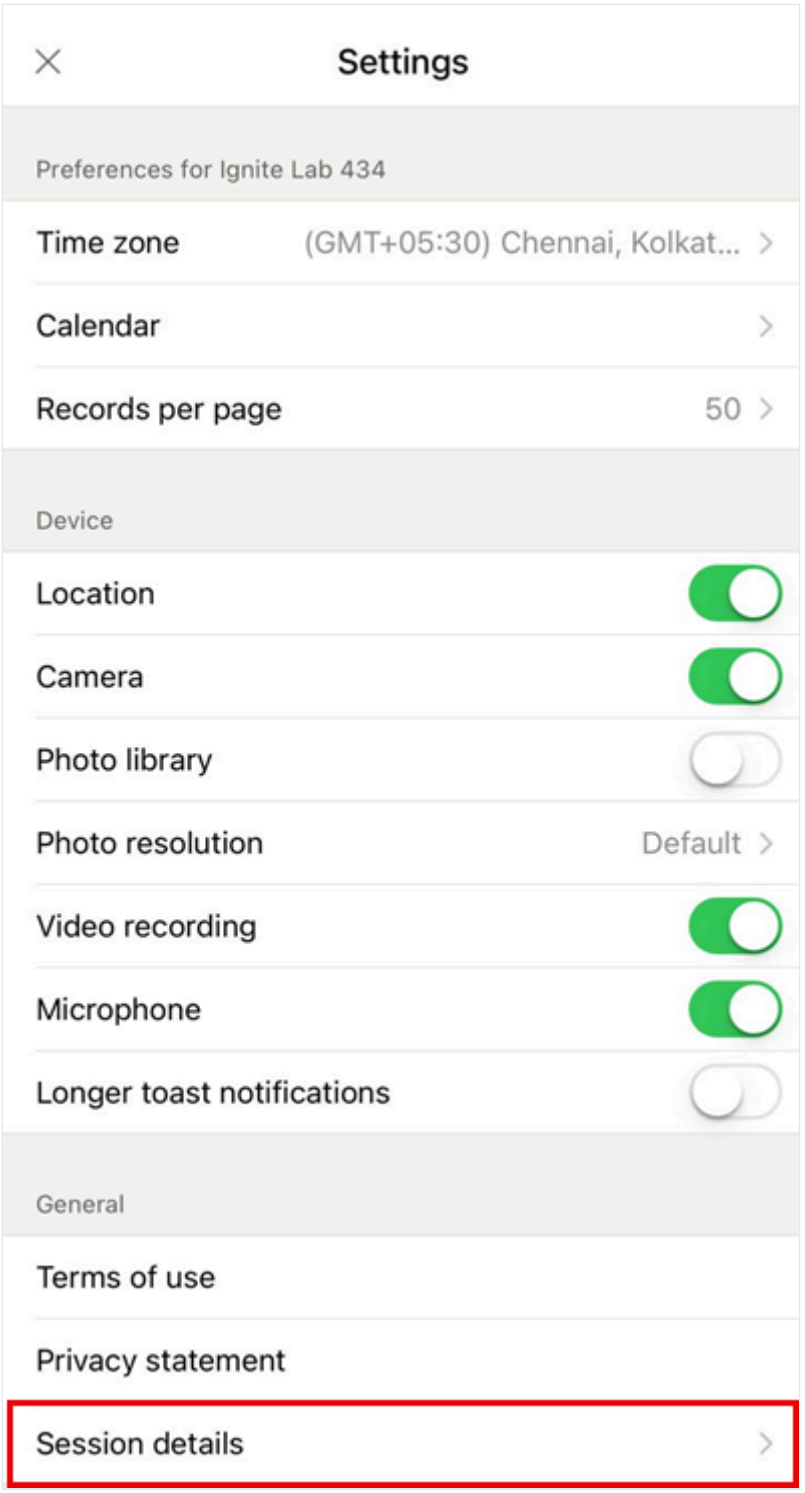
1. In the upper left corner, select the hamburger icon (the three lines).



2. Select the settings icon (gear) in the lower left corner of the screen.



3. Select **Session Details** at the bottom of the screen.



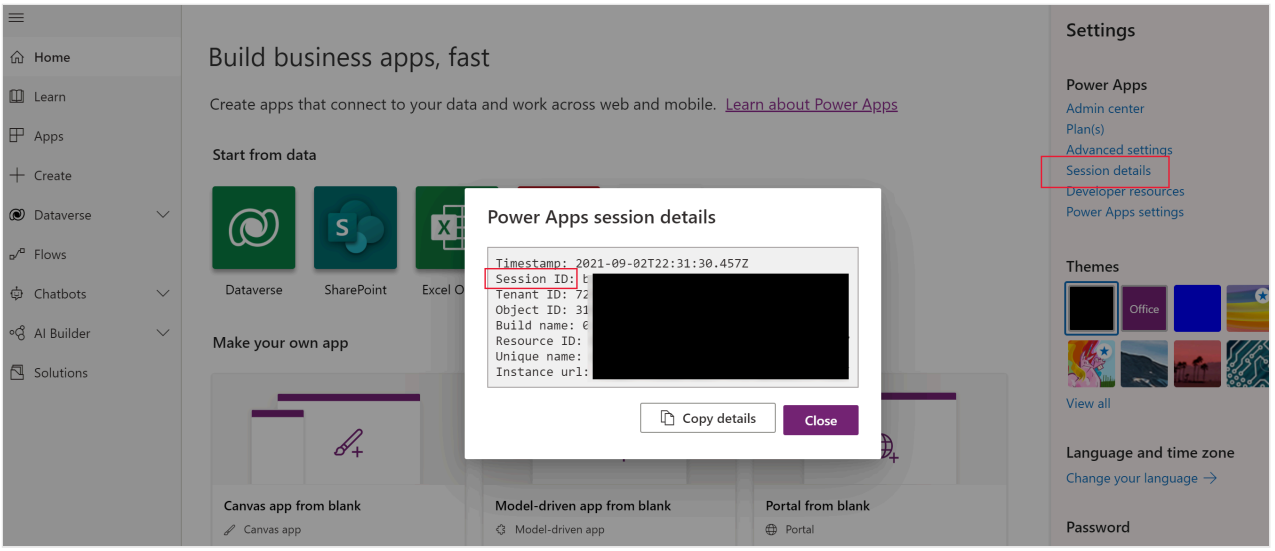
4. Review the session details.



Get the session ID for Power Apps (make.powerapps.com)

1. Sign into Power Apps (make.powerapps.com), and then, on the command bar, select **Settings** (gear).

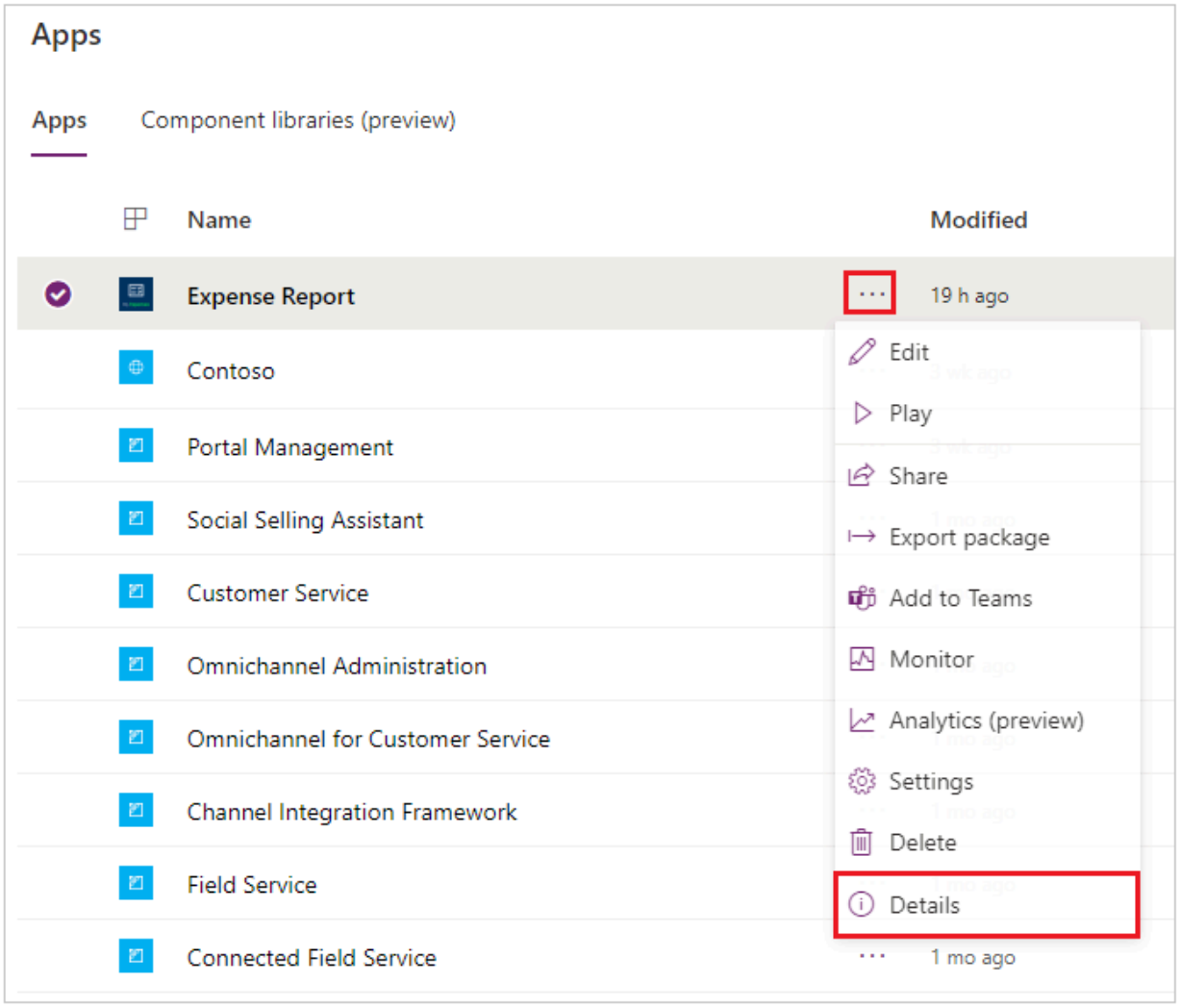
2. Select the **Session details**.



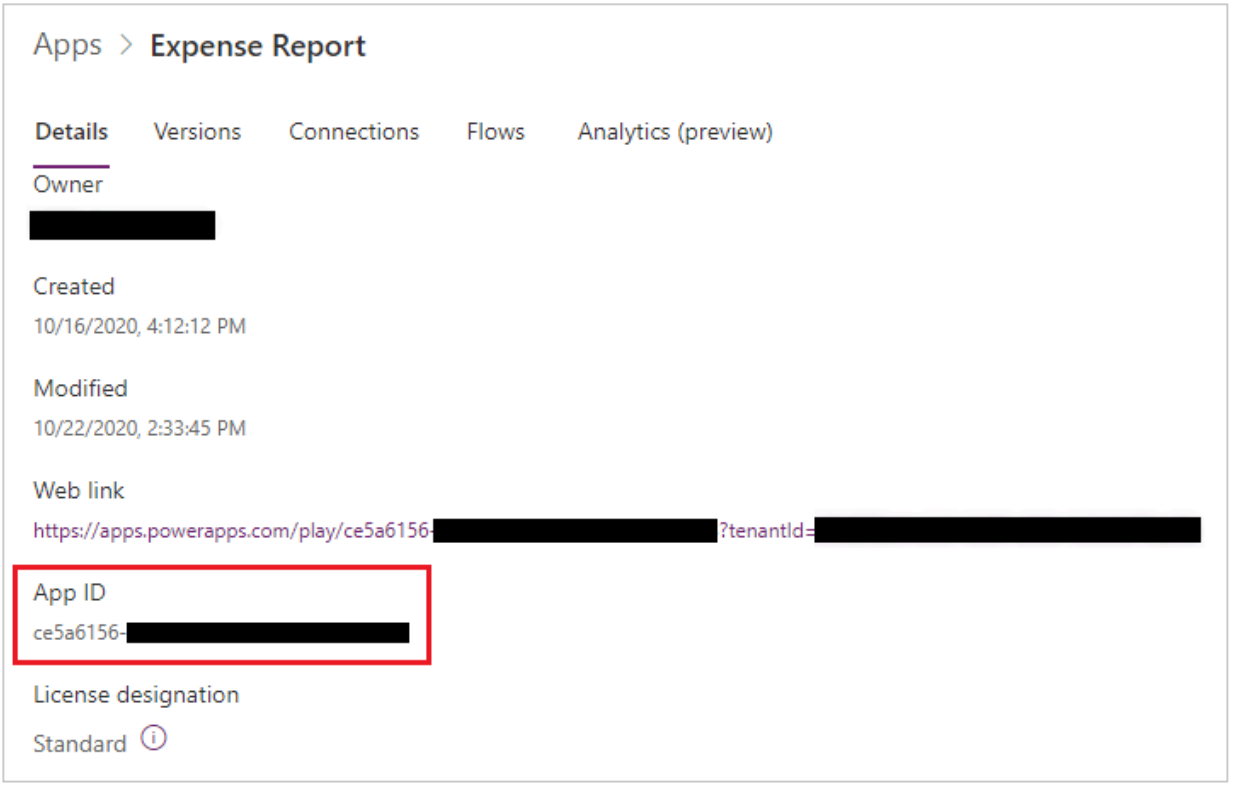
Get an app ID

Get the app ID for either a canvas or model-driven app:

1. Sign in to Power Apps.
2. Near the left edge, select **Apps**.
3. Select ··· (More Commands).
4. Select **Details**.



The app ID appears at the bottom of the **Details** pane for that app.



Create an effective support request

Article • 07/09/2024

If an app isn't behaving as expected, it could be an issue with your Power Apps configuration or a system outside of Power Apps.

Depending on the issue, you can get help from the following individuals or organizations:

Expand table

Type of issue	Who can help best
App	Creators of the app, administrators
Power Apps	Microsoft
External data sources and integrations	Users responsible for the external system

Differences between app bugs and Power Apps bugs

An *app* bug is unexpected behavior in a particular app. In comparison, a *Power Apps* bug is unexpected behavior in the system that creates, runs, or manages the app. An underlying Power Apps bug might cause an app bug, making it difficult to determine if the bug is an app or Power Apps bug.

Here are some distinguishing signs.

Expand table

App bug	Power Apps bug
Able to reproduce an issue in a specific app	Able to create an app that reproduces an issue
Requires access to *specific* data connections	*Any* data connection reproduces the issue, which might require a specific configuration of the data source
Affects only a single app in a single organization	Affects apps in multiple organizations or has the potential to affect them
Complicated app with components not relevant to the issue	Minimal repro app that shows an issue with a few components

App bug	Power Apps bug
Mentions custom features and code	Mentions specific Power Apps features
Microsoft documentation doesn't exist for the affected feature	Microsoft documentation exists for the affected feature and explains its expected behavior

Example app bugs

The following examples show app bugs and how users can describe them as steps to app owners, so owners can reproduce and resolve the bugs.

1. Canvas app doesn't show invoices.
   a. Sign in to your canvas app, such as Contoso.
   b. On your Home screen, select **My Invoices**.

      - Expected behavior: Invoices created by the user are displayed.
      - Actual behavior: No invoices are shown.

2. Model-driven app doesn't show invoices.
   a. Sign in to the Contoso app in your environment.
   b. On the **Dashboard** page, select **Invoices** on the site map.
   c. On the **Invoices** page, set the view to **My active invoices**. The list of invoices are displayed.

      - Expected behavior: Invoices created by the user are displayed.
      - Actual behavior: No invoices are shown.

Bugs can have great variance because features, user interfaces, and tables are specific to an app and organization, making the cause of bugs different for everyone. For example, **invoice** isn't a built-in table in Power Apps. There isn't a specialized feature in canvas apps for filtering records by a certain user, therefore the app maker must write Power Fx expressions or configure data connectors. Similarly, the maker must configure **Views** in model-driven apps to display desired records.

There isn't sufficient information in either of the examples to determine if there's a Power Apps issue. The creators of the app are best suited to investigate app issues.

Example Power Apps bugs

If app creators investigate and find what they classify as a Power Apps bug, they might report them similarly to the following examples.

1. **Canvas app Filter function returns no results when filter text contains asterisk character**
   a. A minimal repro app is attached to demonstrate the issue.
   b. This app contains a simple collection: `TestTable` with data `[{Name: "a"}, {Name: "*b"}]`.
   c. There are two **Table** controls in the app. Both controls are configured to show the **Name** column.
   d. **Table1** has **Items** set to `Filter(TestTable, Name="a")`.
   e. **Table2** has **Items** set to `Filter(TestTable, Name="*b")`.

      - Expected behavior: **Table1** shows the record `{Name: "a"}`, matching the **Filter** function's condition `Name="a"`. **Table2** shows the record `{Name: "*b"}`, matching the **Filter** function's condition `Name="*b"`.
      - Actual behavior: **Table1** shows expected record but **Table2** doesn't show any records.
      - Notes: Same behavior is observed with other dataset controls like **Gallery**, when **Items** are set to the same expressions.

2. **Model-driven app View returns no results when filter condition contains asterisk character**
   a. Create a **View** for any table.
   b. Remove all filters for the view.
   c. Add a filter for the primary column, matching text that starts with **b*.
   d. Add this view and the table to any model-driven app.
   e. Save and publish all changes.
   f. Run the app.
   g. Add a few records with different values for the primary column. Ensure one column starts with **b*.
   h. Go to the table's page.
   i. Change the view to the newly created view.

      - Expected behavior: Grid shows records with primary column values that start with **b*.
      - Actual behavior: No records shown.
      - Notes: When the filter condition doesn't use an asterisk, like *b*, the filter works as expected.

These Power Apps bugs describe issues with specific Power Apps features: the **Filter** function for the canvas app and View filters for the model-driven app.

A good support request provides enough information for anybody to reproduce the issue easily. The author might mention the result of their investigative actions or

troubleshooting they tried when attempting to isolate the issue. For example, the author might mention the issue only occurs in the Chrome browser but not in Firefox.

What makes a good Power Apps support request

A good support request frames an issue as a Power Apps bug, not an app bug. Follow these guidelines, where applicable, to help Microsoft understand and resolve the issue quickly.

Have a descriptive title

The title should mention:

1. A specific Power Apps feature
2. The unexpected behavior
3. The conditions that cause the bug to happen

Example of a bad title: "Gallery control is blank"

Example of a good title: "Filter function returns no results when filter text contains asterisk character"

Attach a simplified app, not the original app

For issues with running an app, provide a minimal repro app for canvas apps, or a vanilla repro solution for model-driven apps.

> **ⓘ Important**
>
> The goal of a good support request is to include enough information so that anyone can reproduce the issue on their own device.
>
> If you can show that the bug is a Power Apps bug instead of an app bug, with a minimal or vanilla repro app, Microsoft can resolve it swiftly. Without these repro apps, the resolution might be delayed or the support request might not be accepted.

Exceptions

A minimal or vanilla repro app isn't always feasible or needed for these issues:

- Data access and saves from a specific data source
- Specific user data, such as user permissions
- Licensing
- Offline operation
- General server issues

Regardless, simplify your app as much as possible and isolate the issue for canvas or model-driven apps.

Attach a network trace

For data and server issues, examining network communications between the client and server helps isolate the problematic layer. A record of network calls is known as a network trace.

You can either use Monitor or browser development tools to record a network trace.

Be sure to start recording just before reproducing the issue and end the recording right after you successfully reproduced it. This brief interval minimizes irrelevant information in the trace that might cause delays in Microsoft's handling of the support request.

Provide detailed steps to reproduce the issue

If you created a minimal or vanilla repro app, describe *how to reproduce the issue* in that app. Don't describe issues with the app where you originally found the problem.

Mention the observed behavior and the expected behavior.

Provide screenshots or videos

For user interface issues, screenshots or videos can more quickly clarify the steps to reproduce the bug, rather than a written description.

Link to official documentation

Include links to Microsoft documentation to clarify the affected feature and its expected behavior.

For issues with professional development features, provide a link to the documentation for the API function that doesn't work.

Simplify code samples

If the issue involves coding and other professional development features, simplify the code first. Usually, just a few lines of code are needed to demonstrate a Power Apps bug. Deploy the app in a fresh environment with no other customizations. Verify the issue occurs and attach a snippet of the problematic code.

Examples of professional development features in Power Apps:

- Power Apps components
- Custom form scripts in model-driven apps
- Power Fx or JavaScript commands in model-driven apps
- Web resources for model-driven apps

Mention specific configuration steps so that others can reproduce the issue in their own environment.

Provide system info

An issue might happen only on a specific version of Power Apps, a browser, or an environment. Provide the session ID so that Microsoft can use these details.

Add notes from your investigation

You might try different configurations to eliminate possible causes. Mention these configurations in the support request. This inclusion helps others understand the issue better and avoid repeating the same steps.

Submitting a Power Apps support request

Before submitting a support request, check if the bug is a known issue. If the bug isn't already known, you can create a support request to report the bug.

See also

- General Power Apps debugging strategies

Feedback

Was this page helpful? Yes No

Provide product feedback

Change authoring version for Power Apps Studio

07/08/2025

The default version of a Power Apps Studio session is determined by your environment's location and refresh cadence. This default version of Power Apps Studio changes when an update is released for your environment based on configured refresh cadence.

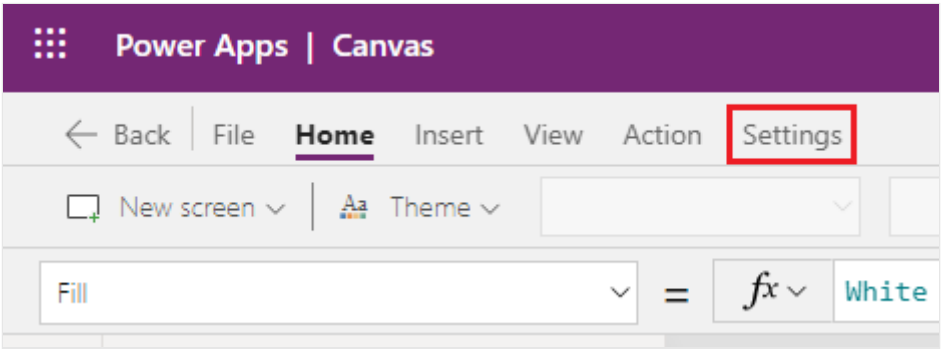
You can now change the authoring version of your current Power Apps Studio session from the version loaded by default, to an earlier or later version as available for your environment.

> **Important**
>
> Change the authoring version of a current Power Apps Studio session only when needed, such as to work around an issue as instructed by Microsoft support.

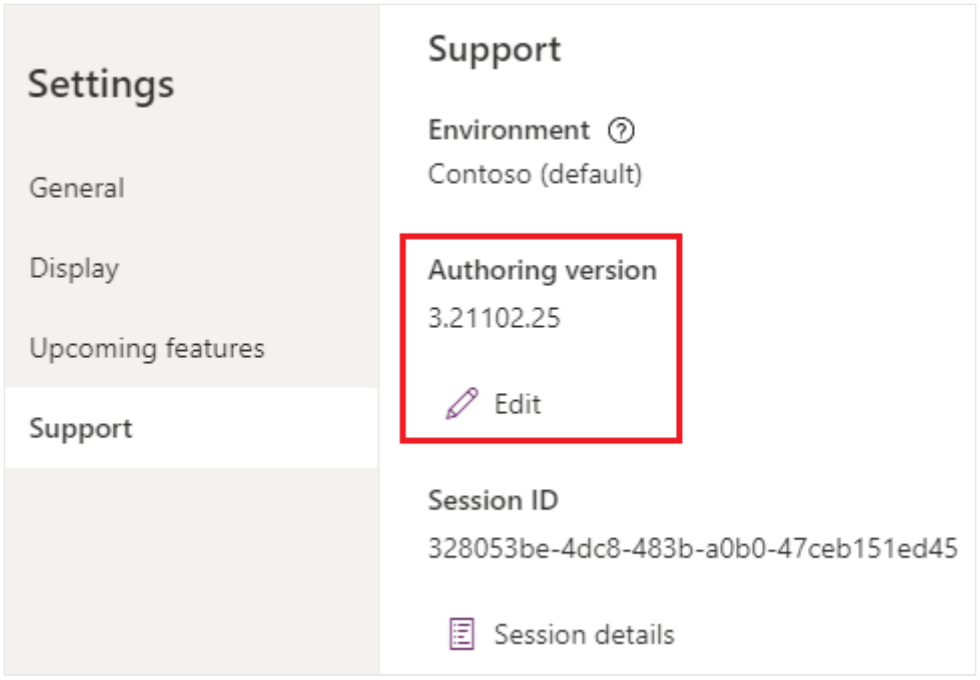
How you check and change the authoring version of your current Power Apps Studio depends on whether you have an app already open, or not.

Check Power Apps Studio version when an app is open

1. Open an existing or a new app in Power Apps Studio.
2. Select **Settings**.



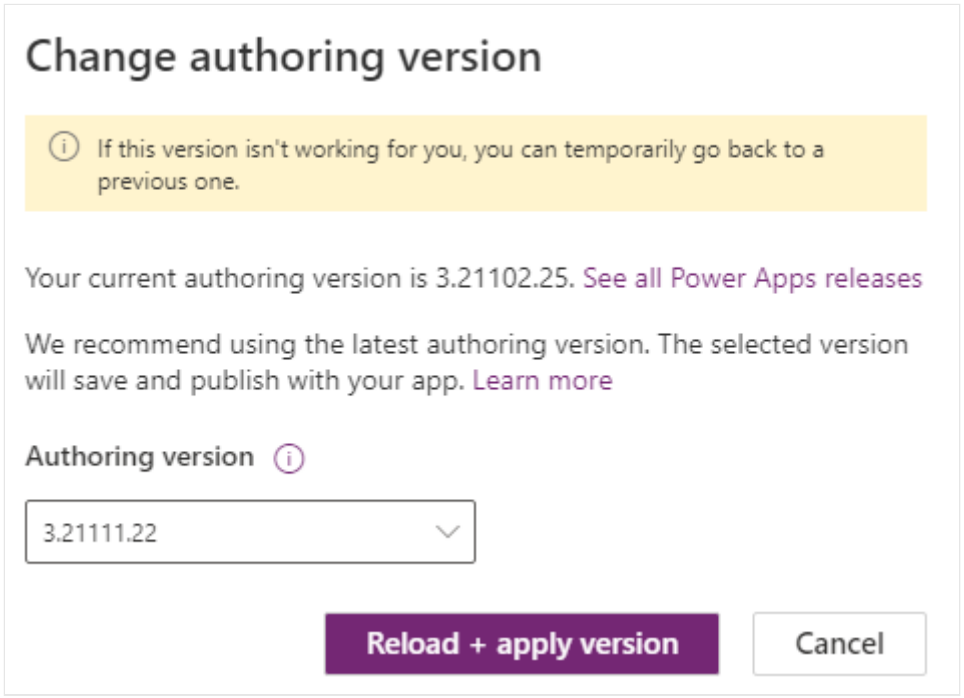
3. Select **Support** and review your existing Power Apps Studio version for the current session.



Change Power Apps Studio version

1. If you're instructed to change the version of your current Power Apps Studio session, follow the steps to see the current version.

2. From the list of available authoring versions, choose the version you want, and then select **Reload + apply version**.



Tip

The default version for your environment will have **(recommended)** next to the authoring version number.

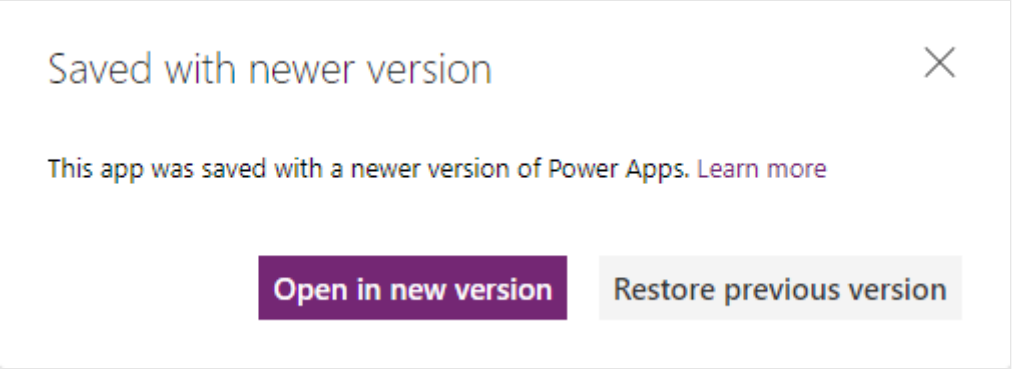
3. (Optional) Save the app when you're prompted to save the changes if you had an app already open that wasn't saved.
4. Select **Open** or **New** to work with the app inside Power Apps Studio using the version you selected.

You're now editing the app in the Power Apps Studio version you've selected. Since the authoring version change only persists for the current Power Apps Studio session, repeat the above steps when you want to work with the same app again.

> ⓘ **Note**
>
> You can't change the authoring version of Power Apps Studio when a you author custom page content or edit an existing custom page.

Known limitations and workarounds

- Selecting **Reload + apply version** only changes the current authoring session to the version you choose. This change doesn't persist for future sessions. New Power Apps Studio sessions always start with the default Power Apps Studio version for your environment.
- To update the app with the Power Apps Studio version you selected, you must save the app with the chosen Power Apps Studio version.
- If you save an app using a version newer than the default for your environment, you might not be able to open that app using the default version of Power Apps Studio. In that case, you'll see the following dialog:



- Select **Open in new version** to open your app for editing until your environment's default version gets updated to the app's current version.
- Select **Restore previous version** to see all versions for the app, and revert to an older version saved with the default Power Apps Studio version associated with your environment.

  > **Note**
  >
  > If you choose to restore the app to a previous version, you'll lose any changes made between your latest and previous app versions.

- You cannot change authoring versions if coauthoring is enabled.

See also

- Setting an environment refresh cadence
- Restore a canvas app to a previous version
- Get session and app ID details

Troubleshooting startup or sign-in issues for Power Apps

Summarize this article for me

This article helps you resolve some common issues or errors that might occur when starting up or signing in to Power Apps.

Common issues or errors

The following are some common issues or errors that might appear when you start up or sign in to Power Apps.

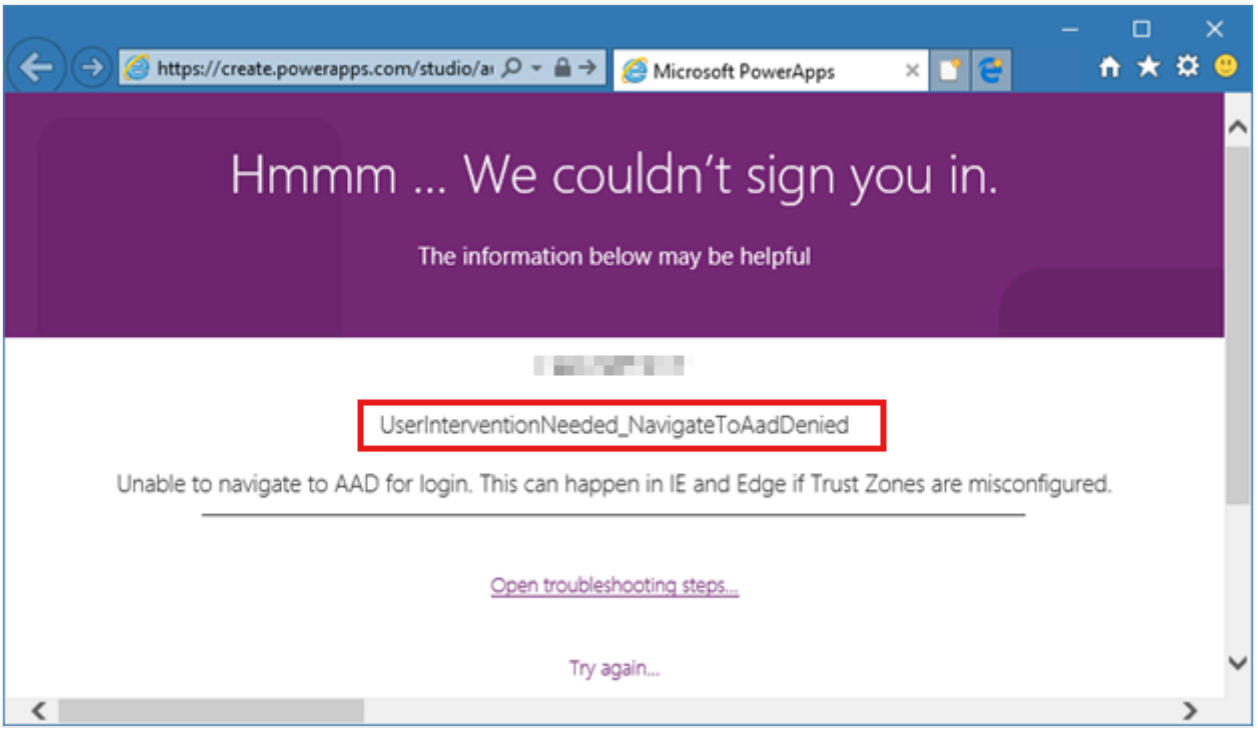
- You're prompted to sign in every time an app is embedded in another client such as SharePoint and Microsoft Teams. The Power Apps opening experience starts and halts until you sign in.
- Error message related to cookie settings.

  > Hmmm... Something went wrong.
  > thirdPartyCookiesBlocked
  > Please enable third party-cookies and site data in your browser settings. If you are using Chrome's Incognito mode, you can uncheck the 'Block third-party cookies' option on the Incognito landing page.
  > Try again

- "Sign in required" message when signing in to Power Apps, especially in InPrivate or incognito mode.

  > Sign in required
  > Please select sign in to continue.
  > Session ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  >
  > AADSTS50058: A silent sign-in request was sent but no user is signed in. The cookies used to represent the user's session were not sent in the request to Microsoft Entra ID. This can happen if the user is using Internet Explorer or Edge, and the web app sending the silent sign-in request is in different IE security zone than the Microsoft Entra endpoint (login.microsoftonline.com).
  > Trace ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  > Correlation ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
  > Timestamp: xxxx-xx-xx xx:xx:xxZ

- "Hmmm ... We couldn't sign you in" message.



- "WebAuthoring abnormal termination" message.

> WebAuthoring abnormal termination.
>
> Client date/time: <Client Time>Thh:mm:ss.sssZ
> Version: 2.0.602
> Session ID: xxxx-xxxxx-xxxxxxx--xxxxxxxx
> description: {"error":{"detail":{"exception":{}},"colno":0,"filename":"
> <`https://paaeuscdn.azureedge.net/v2.0.602.0/studio/openSource/modified/winjs/js/base.js?v=39de0f2edf1`>...",
> "lineno":0,"message":"Script error","initErrorEvent":"
> [function]","bubbles":false,"cancelBubble":false,"cancelable":false,"currentTarget":"
> [window]", "defaultPrevented":true,
> "eventPhase":2,"isTrusted":true,"srcElement":"[window]","target":"
> [window]","timeStamp":1490711965955,"type":"error","initEvent":"
> [function]","preventDefault":" [function]",
> "stopImmediatePropagation":"[function]","stopPropagation":"
> [function]","AT_TARGET":2,"BUBBLING_PHASE":3,"CAPTURING_PHASE":1},"errorLine":0,
> "errorCharacter":0,
> "errorUrl":"
> <`https://paaeuscdn.azureedge.net/v2.0.602.0/studio/openSource/modified/winjs/js/base.js?v=39de0f2edf1`>... error","setPromise":"[function]","exception":{}}
> stack: null

> errorNumber: 0
> errorMessage: Script error

- UserInterventionNeeded_CookiesBlocked
- UserInterventionNeeded_StorageBlocked
- UserInterventionNeeded_NavigateToAadTimeout
- UserInterventionNeeded_NavigateToAadDenied
- UserInterventionNeeded_StorageLost
- AadError

Resolution

Try the following steps to resolve the issue:

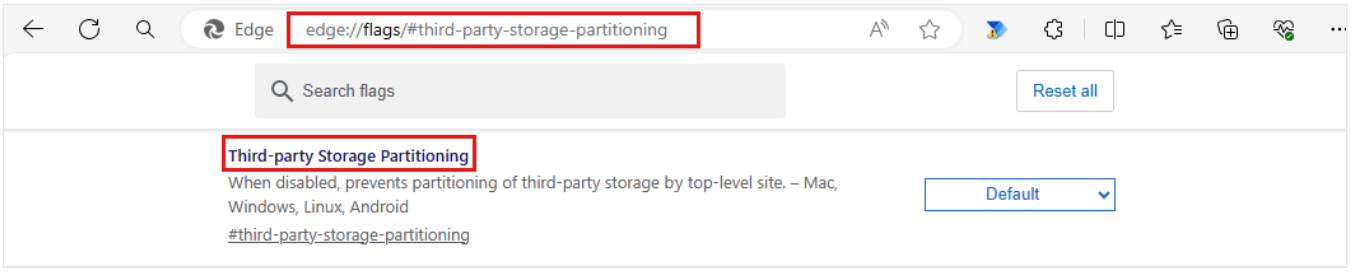
1. Enable third-party cookies and local data in your browser or app.
2. Clear your browser's cache and cookies and try again. Cached data can sometimes prevent you from signing in.
3. Try signing in with a different browser. For a list of supported browsers, see system requirements.
4. Check your network connection to make sure it's stable.
5. If you're getting Microsoft Entra errors, they're usually related to user authentication and authorization. The error page might contain additional information that can help diagnose and fix the problem. To resolve Microsoft Entra errors, you might need assistance from your IT department.
6. Check the "Third-party Storage Partitioning" setting in your browser to make sure it's disabled.

   > ⓘ **Note**
   >
   > Consider this resolution step only when you experience sign-in or sign-out issues under one of the following conditions:
   >
   > - You use a device that has multiple Microsoft Entra identity sign-ins to access different applications.

- You access multiple applications using Microsoft Entra across more than one tab. When you sign out of one tab, you observe that you aren't signed out of the second tab.
- You access Power Apps that are embedded via an iframe in a third-party website.
- You access a canvas app that's embedded in a model-driven app form.

Check the "Third-party Storage Partitioning" setting in Microsoft Edge or Google Chrome

- In Microsoft Edge, you can check the setting by navigating to `edge://flags/#third-party-storage-partitioning` using the address bar.
- In Google Chrome, you can check the setting by navigating to `chrome://flags/#third-party-storage-partitioning` using the address bar.



Enable storage of third-party cookies and local data in your browser or app

Power Apps stores some data locally, such as user identity and preferences, using your browser's capabilities. Problems occur if the browser blocks the storage of such local data, or third-party cookies set by Power Apps.

Most browsers allow settings to reflect the changes immediately. You might also need to close all the browser windows and reopen them instead.

To enable this setting for the Power Apps and Dynamics 365 mobile apps for iOS, you need to work through the iOS settings linked to the app rather than through the browser settings for iOS.

These instructions are shown on the following tabs.

Microsoft Edge

- Option 1: Enable storage of third-party cookies and local data for all sites

    1. Select **Settings** > **Cookies and site permissions**.
    2. Expand **Cookies and data stored**.
    3. Ensure the **Block third-party cookies** setting is disabled.
    4. If present, remove the following sites from the site-specific cookie configuration under **Block** and **Clear on exit**:
        - `https://create.powerapps.com`
        - `https://*.create.powerapps.com`
        - `https://make.*.powerapps.com`
        - `https://make.powerapps.com`
        - `https://login.microsoftonline.com`
        - `https://apps.*.powerapps.com`
        - `https://apps.powerapps.com`
        - (Only for sovereign clouds) US Government version URLs.

- Option 2: Create exceptions to allow the storage of third-party cookies and local data for Power Apps and associated services

    > **Note**
    >
    > The following steps require your Edge browser version to be 87 or above.

    1. Select **Settings** > **Cookies and site permissions**.
    2. Expand **Cookies and data stored**.
    3. Select **Add** under **Allow** and add:
        - `[*.]powerapps.com`
    4. Select **Clear browsing data on close**.
    5. Ensure **Cookies and other site data** is disabled. If you want to keep it enabled, select **Add** instead, and then add:
        - `[*.]powerapps.com`

Clear your browser cache

The browser cache is stored on your device's hard drive. When you visit a website, your browser downloads certain information that allows it to load faster when you revisit the same website in the future. Some Power Apps features use the browser cache to provide a faster user

experience. In some cases, you might want to clear your browser cache. Here are the instructions for different browsers:

- Microsoft Edge
- Google Chrome
- Safari on Mac

Next steps

If your issue isn't listed in this article, you can search for more support resources or contact Microsoft support. For more information, see Get Help + Support.

Last updated on 03/12/2026

App access checker for model-driven apps

07/22/2025

Use the app access checker tool to identify common issues found for a specific user that can occur when running a model-driven app.

Issues found by the app access checker include:

- If an app is visible to the user or not.
- The reason why an app is visible or not visible to the user.
- If the user has the required license.

How to use the app access checker

Power Platform admins and makers can use the app access checker to search apps based on user principal name (UPN) or email address.

Open the diagnostics page in your browser by typing https://*environmentURL*/WebResources/msdyn_AppAccessChecker.html, such as `https://contoso.crm.dynamics.com/WebResources/msdyn_AppAccessChecker.html`. Then enter the UPN or email address in the box and then select **Search**.

How to interpret the results

- **Visible**.
  - If the user doesn't have read privilege on the app module table, none of the apps are visible to the user, and **No** is displayed.
  - If the user has read and/or write privilege on the app module table, then all the apps are visible and **Yes** is displayed.
- **License**.
  - If the user has appropriate licensing to play the app, **Yes** is displayed.
  - Select the **details** link on either **Yes** or **No** result for each app to get information on how the user is or isn't able to access.
- **Security**.
  - If the user has create or write privilege on the app module table, **Yes** is displayed. If the user isn't associated to one or more security roles assigned to the app, check if the user is member of a team and whether the team is associated with that security role.

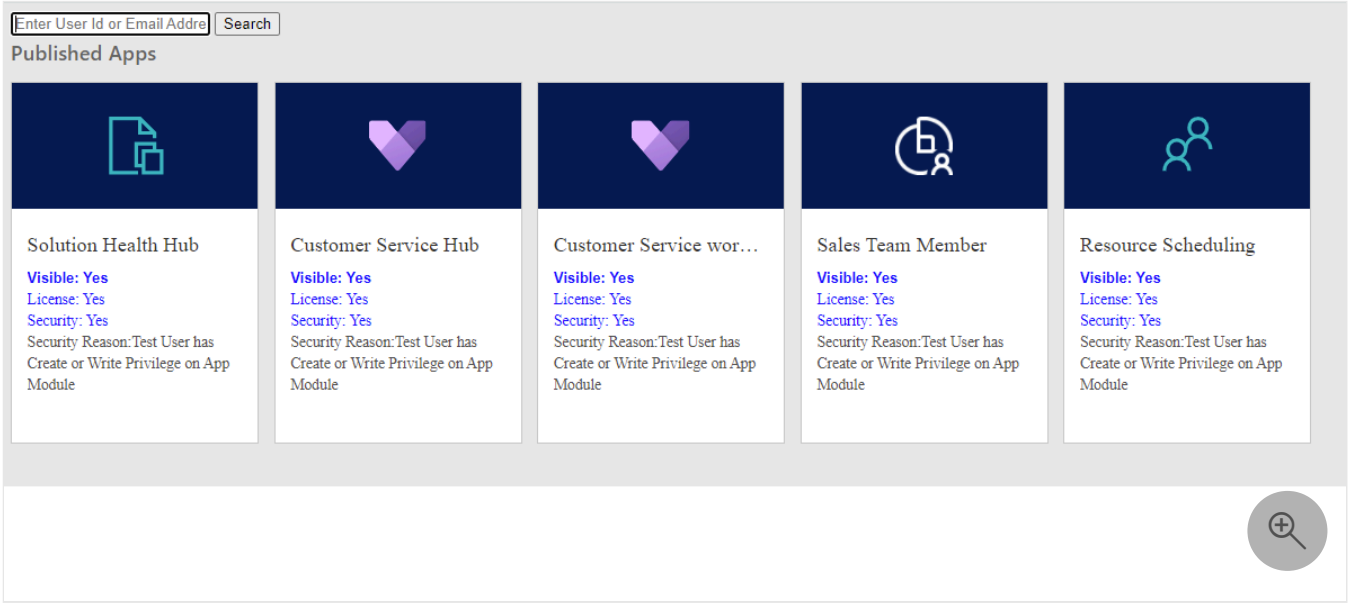
Licensing terminology and how model-driven apps are accessed

In order for an app to be visible to a user, both the results for **License** and **Security** must be **Yes**. If a user doesn't have an appropriate license, the error message might give information about what **service plans** could be used to give access to the app. Note that **Licenses** aren't the same as service plans. A **License** is what is assigned to users in Microsoft Entra and usually has multiple service plans. For a list of all licenses and what service plans are included, go to the licensing service plan reference. Referencing the Microsoft Dataverse licensing error message and the service plan reference list can help diagnose user licensing issues.

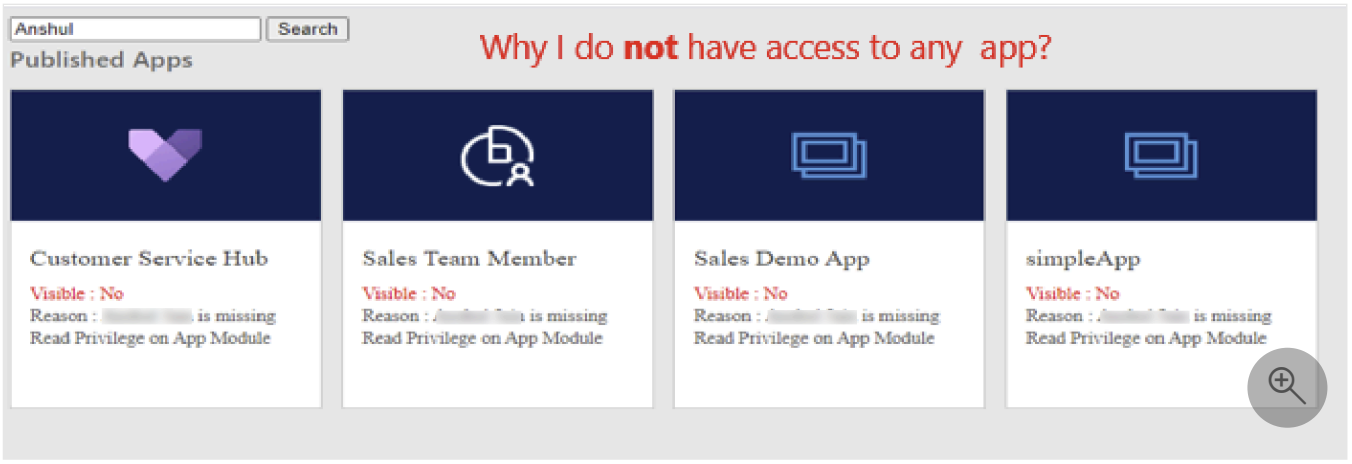
Also be aware that Dataverse caches the service plans that are assigned to users, so Microsoft Entra License assignments might not immediately reflect in Dataverse. To update the cached service plan assignments in Dataverse, have users sign out and sign back in. If the issue persists, it could be a result of Microsoft Entra caching, and the user should wait about 30 minutes, then sign out and sign back in again.

Example results

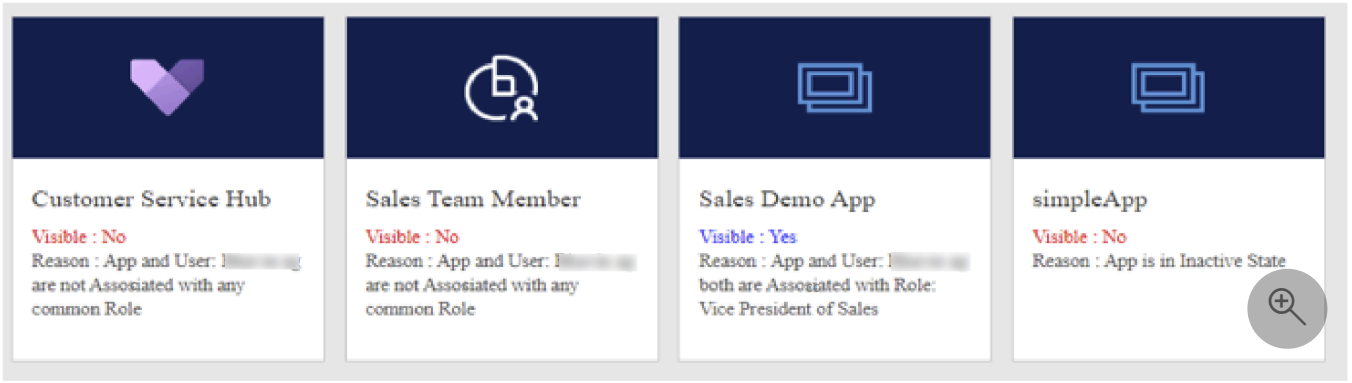
User has access to all visible apps in the environment.

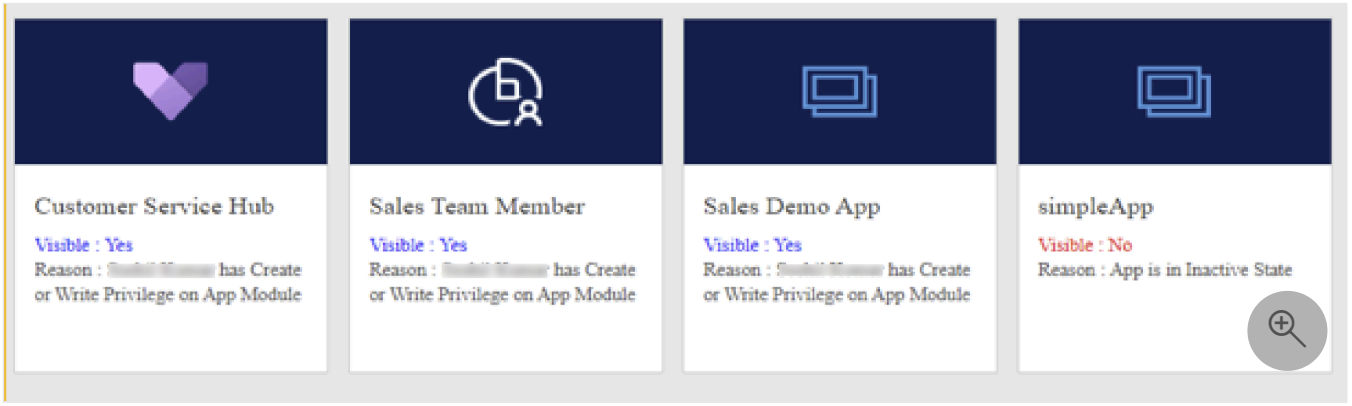


User doesn't have visibility or access to any apps in the environment because of missing security role membership associated with each app.

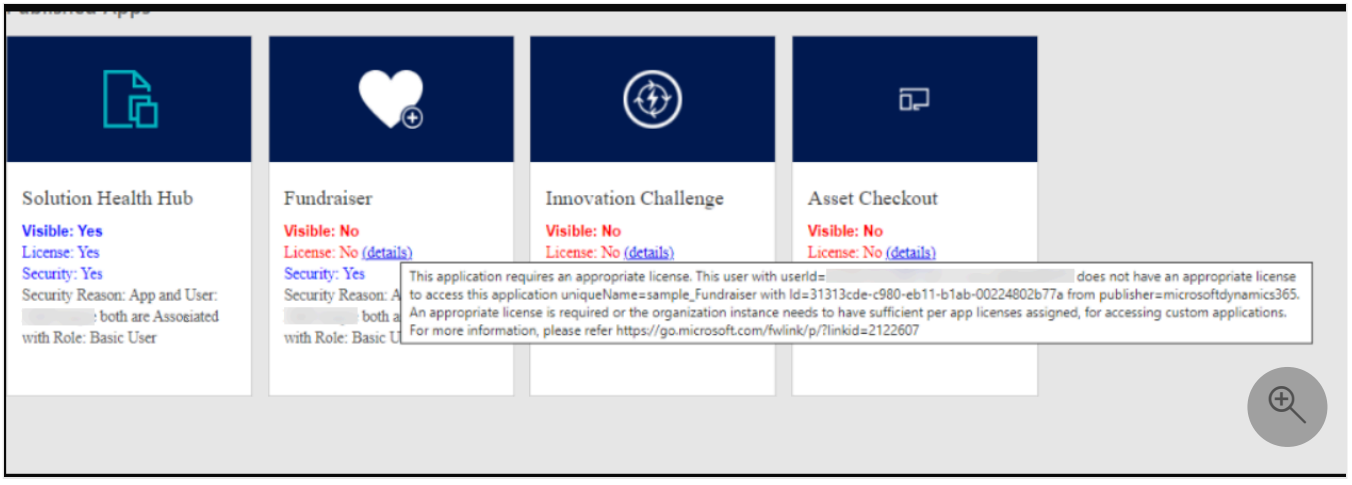


Two examples where the user has visibility to some apps but not others because of missing security role membership associated with the app.

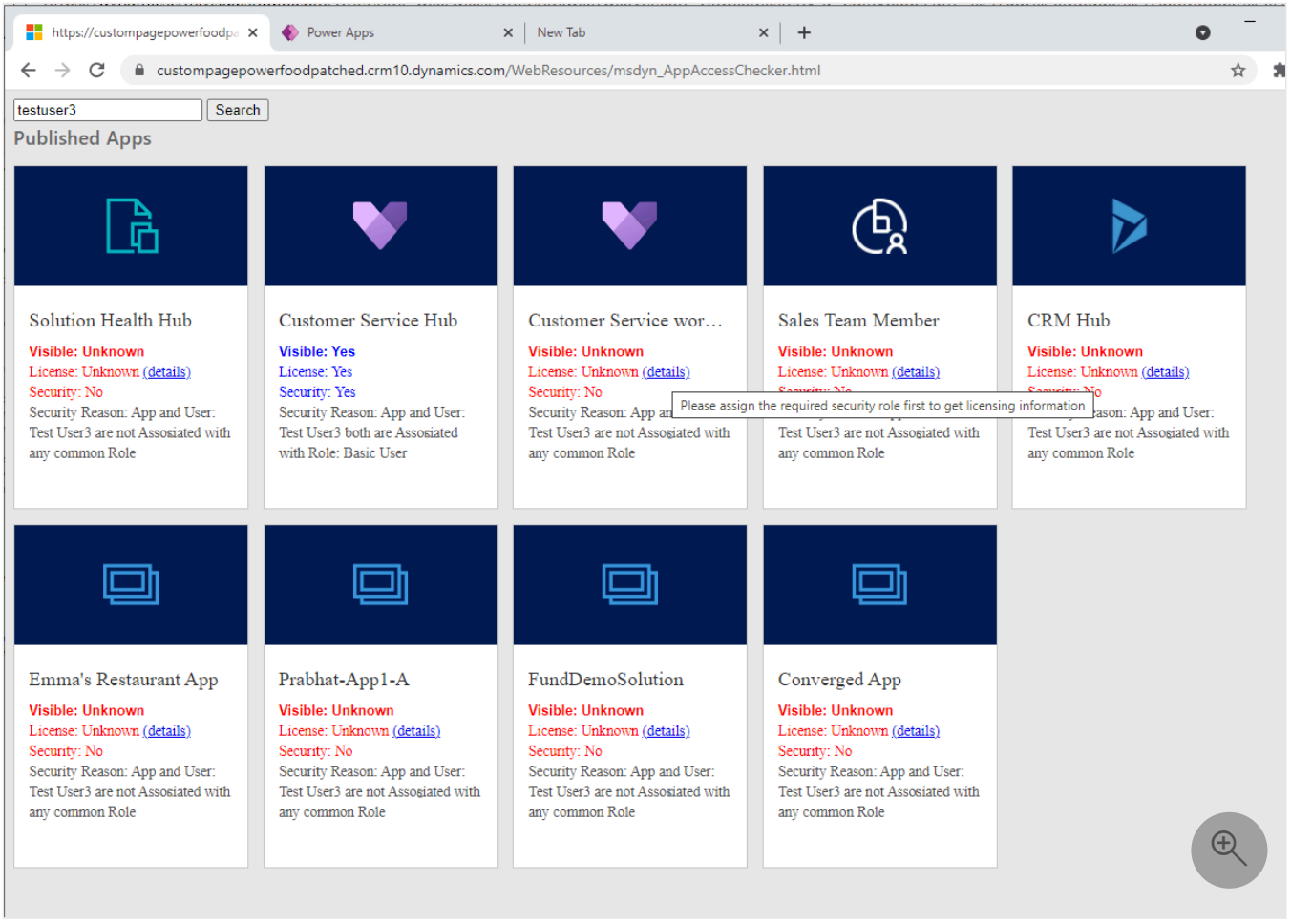




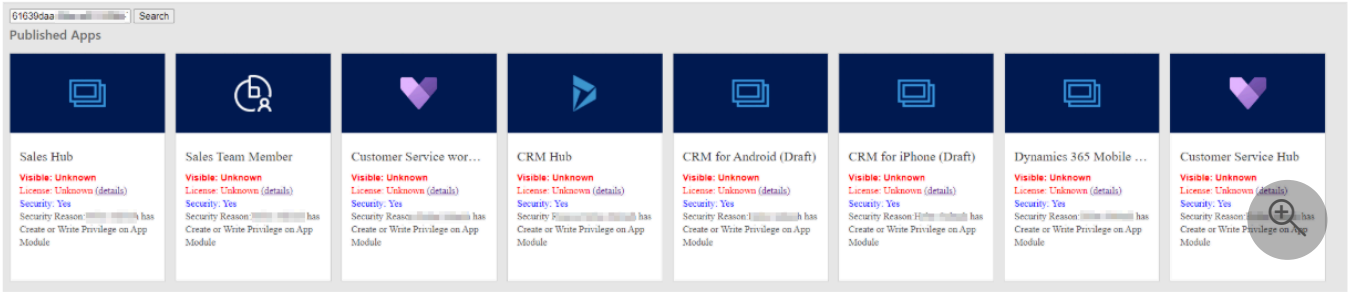
In this example, the user has the required security privilege for the app but doesn't have the required license to access some of the apps.



In this example, the user doesn't have the required security privilege, so the app access checker isn't able to show licensing details.



If **Security** is **Yes** and **License** is **No** or **Unknown** contact Help + support. More information: Get Help + support



Limitation

- App access checker doesn't show details for the Outlook App. This is because that information is only visible to Power Platform administrators.

Related articles

Share a model-driven app

Troubleshoot Power Query issues

When you use Power Query for Excel to create a custom table that contains data from external sources, you may receive the following error:

> "Your Microsoft Entra administrator has set a policy that prevents you from using this feature. Please contact your administrator, who can grant permissions for this feature on your behalf."

The error appears if Power Query can't access the organization's data in Power Apps or Microsoft Dataverse. This situation arises under two sets of circumstances:

- A Microsoft Entra tenant administrator has disallowed users' ability to consent to apps that access company data on their behalf.
- Using an unmanaged Active Directory tenant. An unmanaged tenant is a directory without a global administrator that was created to complete a self-service signup offer. To fix this scenario, users must first convert to a managed tenant and then follow one of the two solutions to this issue. The solutions are described in the next section.

To resolve this issue, the Microsoft Entra administrator must follow either of the procedures that are presented later in this article.

Allow users to consent to apps that access company data

This approach is perhaps easier than the next, but it allows for broader permissions.

1. In the Azure portal, open the **Microsoft Entra ID** pane, and then select **User settings**.
2. Next to **Users can consent to apps accessing company data on their behalf**, select **Yes**, and then select **Save**.

Allow Power Query to access company data

As an alternative, the tenant administrator can give consent to Power Query without modifying tenant-wide permissions.

1. Install Azure PowerShell.
2. Run the following PowerShell commands:

    - `Login-AzureRmAccount` (and sign in as the tenant admin)

- `New-AzureRmADServicePrincipal -ApplicationId 00001111-aaaa-2222-bbbb-3333cccc4444`

The advantage of this approach (versus the tenant-wide solution) is that this solution is very targeted. It provisions only the **Power Query** service principal, but no other permission changes are made to the tenant.

Update personal data

Users can update mashups and other information (such as query names and mashup metadata) through the Query Editor and through the **Options** dialog box that's accessible from the Query Editor.

In Power Apps, you access the Query Editor by doing the following:

1. Go to the **Data** pane, expand it, and then select **Tables**.
2. Select the ellipsis (...), and then select **Edit Queries**.
3. In the ribbon, select **Options**, and then select **Export Diagnostics**.

Delete personal data

Most data is deleted automatically within 30 days. For data and metadata around mashups, users must remove all their mashups through Power Apps. All of the associated data and metadata will be deleted within 30 days.

To remove mashups from Power Apps:

1. Remove the Data Integrator projects, which can be removed from the namesake tab.
2. Select the ellipsis (...), and then select **Delete**.

If you created a mashup through the "New tables from data (Technical Preview)" feature, you can remove it by doing the following:

1. Select the ellipsis (...), and then select **Edit queries**.
2. In the ribbon, select **Options**.
3. Select **Remove all queries**.
   After you confirm that you want to delete your queries, they are deleted.

Export personal data

To export personal data, users can do the following:

1. Open the Query Editor.
2. In the ribbon, select **Options**.
3. Select **Export Diagnostics**.

In Power Apps, you can access the Query Editor by doing the following:

1. Go to the **Data** pane, expand it, and then select **Tables**.
2. Select the ellipsis (...), and then select **Edit Queries**.
3. In the ribbon, select **Options**, and then select **Export Diagnostics**.

System-generated logs about user actions on the user interface (UI) can be accessed in the Azure portal.

Last updated on 04/02/2024

FAQ about using AI responsibly in Power Apps

Summarize this article for me

An AI system includes not only the technology, but also the people who use it, the people affected by it, and the environment in which it's deployed. Microsoft Responsible AI FAQs are intended to help you understand how AI technology works, the choices system owners and users can make that influence system performance and behavior, and the importance of thinking about the whole system, including the technology, the people, and the environment. You can use Responsible AI FAQs to better understand specific AI systems and features that Microsoft develops.

Responsible AI FAQs are part of a broader effort to put our AI principles into practice. To find out more, see Microsoft AI principles.

AI-driven features in this app

This app contains a growing list of AI-driven features. To learn about the capabilities and impact of specific features, select an article in the list.

- Overview of the new Power Apps vibe experience (preview)
- FAQ for Power Apps vibe experience (preview)
- Build apps through conversation
- FAQ about building apps through conversation
- Create app description for canvas apps with Copilot (preview)
- Create a model-driven app description with Copilot
- FAQ about generating app descriptions with Copilot
- Edit your app with Copilot (preview)
- FAQ about editing your app with Copilot
- Add the Copilot control to a canvas app (preview)
- FAQ about using the Copilot control in canvas apps
- Add Copilot to model-driven apps (preview)
- FAQ about using Copilot in model-driven apps
- Use Copilot in the rich text editor for email (preview)
- FAQ about using Copilot in the rich text editor
- Draft well-written, input text with Copilot
- FAQ for drafting well-written, input text with Copilot
- Use form filling assistance in model-driven apps
- FAQs about form filling assistance
- Use timeline highlights powered by generative AI

- FAQs about using Copilot in timeline highlights
- Create Power Fx formulas with Copilot
- FAQs about using Copilot with Power Fx
- Use field suggestions by Copilot
- FAQs for field suggestions by Copilot
- FAQ about filtering, sorting, and searching canvas galleries with Copilot
- FAQ about agent builder in canvas apps
- Build an agent to automate your business process (preview)
- FAQ for the Plan designer
- Overview of Plan designer
- FAQ for rename controls in canvas apps with Copilot
- Rename controls in canvas apps with Copilot (preview)
- FAQ for Agent APIs
- Bring intelligence into your component using Agent Xrm APIs (preview)
- Bring intelligence into your app using Agent Xrm APIs (preview)
- Use Agent Response component in the model-driven app forms (preview)

Last updated on 03/11/2026

FAQ for Power Apps vibe experience (preview)

Summarize this article for me

What is Power Apps vibe?

Power Apps vibe is an AI-powered app creation experience that helps makers quickly build functional business applications using natural language. Makers describe what they want to build, and vibe translates those descriptions into working app experiences, including data, screens, and logic. At a high level, Power Apps vibe takes maker prompts and optional context as input and produces a draft app that makers can review, refine, and publish.

What can Power Apps vibe do?

Power Apps vibe helps makers create apps faster by generating a fully functional app (including app structures, screens, and interactions) based on natural language descriptions. Makers can iteratively refine their apps by adjusting prompts, editing generated outputs, and previewing changes in real time. Power Apps vibe is designed to accelerate early app creation and reduce the effort required to get from an idea to a usable application.

What is the intended use of Power Apps vibe?

Power Apps vibe is intended to support makers who want to quickly prototype, explore, and build business applications without starting from scratch. It helps makers translate ideas into working apps, learn app-building concepts through iteration, and accelerate time to value.

How was Power Apps vibe evaluated? What metrics are used to measure performance?

Power Apps vibe is evaluated using a combination of qualitative and quantitative methods. Evaluation includes user studies with makers to assess the quality of generated outputs, usability of the experience, and how effectively Power Apps vibe supports app creation workflows. Feedback from these studies helps identify strengths, gaps, and opportunities for improvement.

In addition to qualitative feedback, telemetry data is monitored to understand how makers use Power Apps vibe in practice. This data includes metrics such as feature usage, completion and

success rates, and the ratio of positive to negative feedback signals. Before release, Power Apps vibe undergoes extensive functional testing to validate reliability and behavior across supported scenarios. For more information, see the Privacy Statement.

What are the limitations of Power Apps vibe? How can users minimize the impact of Power Apps vibe's limitations when using the system?

Like other AI-powered experiences, Power Apps vibe has known limitations. Generated apps might not always fully reflect a maker's intent. They might require extra refinement or might not be suitable for complex or highly customized production scenarios without further modification. Outputs are based on the information provided by the maker and the capabilities available in the platform.

Known limitations include:

- Non-English languages aren't supported.
- Supports image attachment up to 2 MB.
- You can't access or edit apps created outside of Power Apps vibe.
- If you export an app and edit it outside Power Apps vibe (for example, in VS Code), redeploying via PAC CLI creates a new app, and it disconnects from the original plan.
- Currently, you can only have one app per plan; multiple apps within a single plan aren't supported.
- You can't open preexisting plans in Power Apps vibe.
- Canvas and model-driven apps aren't supported; you can't recommend, open, or edit these app types in Power Apps vibe.
- Existing tables aren't automatically recommended during data model proposal; you can manually add existing tables to your plan.
- Editing existing Dataverse tables via chat isn't currently supported; you can make changes to these tables through the data editor manually.
- Direct code edits in Code view or Split view aren't supported; you can't modify code directly in these views.
- You can't directly create proposed technologies (apart from code apps) from the plan; you must add additional technologies manually.

Makers can minimize the impact of these limitations by:

- Providing clear, specific descriptions of their app requirements.
- Iterating on prompts and reviewing generated outputs carefully.
- Testing app behavior and data interactions before publishing.
- Treating generated apps as a starting point rather than a finished solution.

Makers are ultimately responsible for validating that their apps meet their functional, security, and compliance requirements.

What operational factors and settings allow for effective and responsible use of Power Apps vibe?

Effective and responsible use of Power Apps vibe depends on how makers configure and interact with the system. Makers should:

- Clearly describe business scenarios, users, and tasks when prompting
- Review generated content before saving or sharing it
- Test apps in appropriate environments prior to broader use
- Follow organizational governance, data, and compliance policies

How do I provide feedback for Power Apps vibe?

- Select the thumbs up or thumbs down in the chat pane to provide feedback on a specific action.
- Select the **Give feedback** icon in the command bar to provide general feedback, report an issue, or share suggestions.

See also

Create apps, data, and plans together using vibe (preview)

Last updated on 03/11/2026

FAQ for Power Apps in agents

These frequently asked questions (FAQs) describe the AI impact of Power Apps in agents for model-driven apps.

What is Power Apps in agents?

Power Apps in agents enables users to interact with data from Power Apps model-driven apps directly in Microsoft 365 Copilot through natural language. When makers deploy a declarative agent for a model-driven app, users can query records, view data in grids, create and edit records, and receive AI-generated summaries, all through a Copilot conversation, without switching to the app.

The agent connects to the model-driven app's underlying Dataverse tables and uses the same data exploration capabilities that power the Grid view experience in model-driven apps.

What are the system's capabilities?

The system can:

- Query and filter records across tables configured in the model-driven app using natural language
- Display results as an interactive grid, consistent with the Grid view in the app
- Show individual record details using a Form, consistent with the Form experience in model-driven apps
- Create and update records using the Form widget in Copilot
- Hand off to the full model-driven app for actions that require the complete app experience

What is the system's intended use?

Power Apps in agents serves two audiences:

- **For makers**: Extend the reach of an existing Power Apps model-driven app by making its data accessible to users directly in Microsoft 365 Copilot. Deploy a declarative agent for your app. No additional app development is required beyond the initial setup.

- **For end users**: Help business users access and act on their organization's data without leaving Microsoft 365 Copilot, reducing context-switching and enabling faster decisions in the flow of work. It's not intended to replace the full model-driven app experience, but to make common data tasks – querying, reviewing, and updating records – accessible

through Microsoft 365 Copilot. When more complex actions are needed, it also serves as a handoff point directly to the model-driven app.

How was Power Apps in agents evaluated? What metrics are used to measure performance?

The system was evaluated across a range of query types, table configurations, record volumes, and user scenarios. Evaluation focused on query accuracy, widget render fidelity, and the correctness of information presented in them.

If you encounter problems, submit feedback through Microsoft 365 Copilot. Microsoft uses your feedback to improve products and services. For more information, see Data, privacy, and security for Azure OpenAI Service.

What are the limitations of Power Apps in agents? How can users minimize the impact of these limitations?

- **Query accuracy**: Data retrieval results might not always match the user's intent on the first attempt. Iterate by using natural language to refine results, such as by guiding Copilot on which column or field you meant in your original ask.
- **Unsupported field types in forms**: The form fill experience doesn't support certain field types. For those fields, open the full model-driven app to complete or edit the record.
- **Non-English languages are unsupported**: Power Apps in agents currently supports English-language queries and responses only. If you interact in other languages, you might receive inaccurate or unexpected results.

What operational factors and settings allow for effective and responsible use of the system?

Data retrieval is powered by the data exploration agent. Power Apps in agents leverages the Data Exploration Agent, the same underlying AI capability that powers the Grid view experience in Power Apps model-driven apps. As a result, the quality and accuracy of data retrieval is benchmarked against and consistent with the data exploration experience users already have in model-driven apps.

Form filling is powered by form fill assistance. Power Apps in agents leverages Form Fill Assistance, the same underlying AI capability that powers field suggestions in Power Apps model-driven app forms. As a result, the quality and accuracy of prefilled values is benchmarked against and consistent with the form-filling experience users already have in model-driven apps. Users should review prefilled values before saving.

How do I provide feedback?

Use the thumbs up or thumbs down controls in Microsoft 365 Copilot to provide feedback on individual responses. Microsoft uses this feedback to improve products and services.

Related information

- Set up App in Copilot
- Find data in a view with AI
- Use Copilot's form fill assistance feature in model-driven apps
- Use grid filters and views in model-driven apps
- FAQ for Copilot chat in model-driven apps
- Responsible AI FAQs for Microsoft Power Platform
- Data, privacy, and security for Azure OpenAI Service

Last updated on 04/02/2026

FAQ for building apps and tables through conversation

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of build apps and tables through conversation feature.

What is build apps and tables through conversation?

Build apps and tables through conversation is a feature in the Power Apps that allows you to describe the app or table that you want to build, and AI will design it for you. To start, you use natural language to tell the AI assistant what kind of information you want to collect, track, or show. The assistant will generate a Dataverse table and use it to build your canvas app. The AI assistant is available from the Power Apps home page.

What are the system's capabilities?

This capability is powered by Azure OpenAI Service, which uses powerful language models. These language models generate new text in English that looks and sounds like text written by humans.

Using Azure OpenAI service as a foundation, build apps through conversation considers your app description to propose relevant data tables on which to build your app. As part of data generation, the system recommends relevant columns, sample row data, and data types that you can customize. The sample data is for illustration only and is fictitious. No real association is intended or inferred.

What is the system's intended use?

Build apps and tables through conversation is intended to simplify your app creation process by automatically generating a data table on which your app will be built. AI-generated content may have mistakes. Make sure it's accurate and appropriate before using it. For more information, read the full terms.

How was build apps and tables through conversation evaluated? What metrics are used to

measure performance?

Build apps and tables through conversation underwent substantial testing before the feature was released. If you encounter issues with the content being generated, please submit feedback. Your feedback will be used to improve Microsoft products and services. IT admins for your organization will be able to view and manage your feedback data.

For more information, see: Data, privacy, and security for Azure OpenAI Service

What are the limitations of build apps and tables through conversation? How can users minimize the impact of the build apps through conversation limitations when using the system?

This capability is in the process of rolling out, and may not be available in your region yet.

This capability may be subject to usage limits or capacity throttling.

The following are requirements to access this feature:

- Your account must have English (United States), French, German, Spanish, Dutch, Danish, Italian, Japanese, or Portuguese (Brazil) as the browser language.
- Have a Microsoft Dataverse database in your environment.
- AI Builder must be enabled for your environment to use the AI models or controls leveraging AI models:

    1. Sign in to the Power Platform admin center.
    2. Go to **Environments** > [select an environment] > **Settings** > **Features**.
    3. On the **Features** settings page, under **AI Builder**, enable or disable **AI Builder preview models**.

For more information, see Copilot in Power Apps overview

What operational factors and settings allow for effective and responsible use of the system?

Follow these procedures to make the most of the feature:

- Use everyday words to describe what your app should collect, track, list, or manage such as the following:
  - “Collect RSVPs for the team birthday party”
  - “Track sales leads”
- Customize the data table with prompts such as:
  - “Add a column for …”
  - “Remove the … row”
  - “Change the data type for the … column to… ”
- When in doubt, you can ask the AI assistant for help by typing, “Give me suggestions”

See also

- Build apps through conversation

FAQ for generate app descriptions with Copilot

Article • 05/28/2025

These frequently asked questions (FAQ) describe the AI impact of Power Apps generate app descriptions with Copilot feature.

What is generate app descriptions with Copilot?

With the AI-generated app descriptions feature, Makers can use AI to generate descriptions for their canvas and model-driven apps. Makers can generate a description from app **Settings** or when you publish the app. By using AI to generate descriptions, Makers can focus on building their apps and allow AI to assist in summarizing the purpose of the apps for end-users.

What are the system's capabilities?

It can take the contents of your app and provide a brief description of the app for makers to share with their end-users.

What is the system's intended use?

As an AI assistant to makers, it provides a brief description of your app that end-users can see.

How was generate app descriptions with Copilot evaluated? What metrics are used to measure performance?

We evaluate the feature both qualitatively and quantitatively. To assess the quality of the feature, we're conducting user studies with makers to gather their feedback on their experiences, thoughts about the feature's quality, and suggestions for improvement. Additionally, we're monitoring telemetry data to track the number of makers who tried the feature, the success rate of the feature, and the ratio of positive to negative feedback. Before releasing the Copilot feature in preview, we conducted extensive testing to ensure its functionality. If you encounter any issues with the content generated, provide feedback. Your feedback is used to enhance Microsoft's products and services. Your organization's IT admins have access to your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of generate app descriptions with Copilot? How can users minimize the impact of the generate app descriptions with Copilot limitations when using the system?

- To use this capability, you must be in a Managed Environment.
- This capability is powered by Azure OpenAI Service.
- This capability is in process of rolling out, and might not be available in your region yet.
- This capability can be subject to usage limits or capacity throttling.

For canvas apps (preview)

- Preview features aren't meant for production use and might have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.
- For more information, go to preview terms.
- Your environment must be in the United States region.
- This feature doesn't support non-English language browsers.

What operational factors and settings allow for effective and responsible use of the system?

If you get this error, **Something happened and we weren't able to generate a description for you**, this might be due to capacity limits. We recommend you give the system some time before trying again.

See also

- Create a Canvas App description with Copilot (preview)
- Create a model-driven app description with Copilot

FAQ for edit your app with Copilot

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of Power Apps' for edit your app with Copilot feature.

What is edit your app with Copilot?

With edit your app with Copilot feature, makers can delegate some tasks of app development to an AI assistant by telling Copilot their requirements. Makers can expect actions such as add, edit, or style controls, work with containers, and bulk edits. This allows makers to concentrate on designing the app while maintaining oversight of all actions taken by Copilot.

What are the system's capabilities?

The following system capabilities supported:

- Add a screen using pre-existing templates.
- Add and modify properties of a single control, limited to classic controls only.
- Update multiple controls at once.
- Work with containers.
- Provide default suggestions for users to help in typing.
- Use templatized formulas such as Navigate() to create buttons that perform specific actions, like navigating to the Home screen, and SubmitForm() for creating a Submit button for Form2.

The following controls are supported:

- Screen
- Container
- Gallery
- Form
- Button
- Label
- TextInput

What is the system's intended use?

As an AI assistant that helps people find information, Copilot provides the following list of UI actions for makers that are included in the preview release scope:

- Add a screen using existing screen templates
- Add and update properties of a single classic control
- Bulk updates controls on a single screen
- Work with containers
- Provide default suggestions to guide users on what to type in
- Offer the following templatized formulas to build your app:
  - Navigate(): For example, "Create a button for Home that navigates to the Home Screen"

How was edit your app with Copilot evaluated? What metrics are used to measure performance?

We evaluate the feature both qualitatively and quantitatively. To assess the quality of the feature, we're conducting user studies with makers to gather their feedback on their experiences, thoughts about the feature's quality, and suggestions for improvement. Additionally, we're monitoring telemetry data to track the number of makers who tried the feature, the success rate of the feature, and the ratio of positive to negative feedback. Before releasing the Copilot feature in preview, we conducted extensive testing to ensure its functionality. If you encounter any issues with the content generated, provide feedback. Your feedback is used to enhance Microsoft's products and services. Your organization's IT admins have access to your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of edit your app with Copilot? How can users minimize the impact of edit your app with Copilot limitations when using the system?

- To use this capability, you must have a Microsoft Dataverse database in your environment.
- Preview features aren't meant for production use and might have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.
- For more information, see preview terms.
- This capability is powered by Azure OpenAI Service.
- This capability is in process of rolling out, and might not be available in your region yet.
- This capability can be subject to usage limits or capacity throttling.
- Your environment must be in the United States region.
- This feature doesn't support non-English language input.

- This feature only supports limited controls and actions as listed in the intended use section. Microsoft is actively working on expanding the scope of this feature and support more actions incrementally.

What operational factors and settings allow for effective and responsible use of the system?

Here are some tips to help you get the most out of this feature:

- Use simple language to explain the changes you want to make to your app, such as "add a new screen" or "insert 2 buttons."
- If the results you get aren't what you expected, try being more precise. For example, you could say "add a new button to the selected container," "change the color of the selected button to red", or "display Screen 2 when Button1 is selected."
- You can also try using the default suggestions provided in the Copilot panel and then customize them to suit your needs. And if you still can't get the desired results, send us your feedback.

See also

- Edit your app with Copilot (preview)

FAQ for Copilot chat in model-driven apps

05/28/2025

This FAQ describes the AI impact of Power Apps model-driven apps Copilot chat feature.

What is Copilot chat in model-driven apps?

Copilot chat for model-driven apps in Power Apps is a feature for app users to get insights about the data in their apps through conversation in natural language. Copilot chat helps app users boost their productivity through AI-powered insights and intuitive app navigation. This feature is available throughout the model-driven app.

What are the system's capabilities?

It can answer questions about the configured tables in an app, and aids in navigation through queries such as "Take me to *page name*."

What is the system's intended use?

As an AI assistant to end users, it provides answers to questions about the configured data in the app, and helps with app navigation.

How was Copilot chat in model-driven apps evaluated? What metrics are used to measure performance?

Copilot chat in model-driven apps underwent substantial testing before the feature was released in preview. If you encounter issues while using Copilot chat in model-driven apps, submit your feedback. Your feedback is used to improve Microsoft products and services. For more information, go to: Data, privacy, and security for Azure OpenAI Service

What are the limitations of Copilot chat in model-driven apps? How can users minimize the impact of the Copilot chat in model-driven apps limitations when using the system?

To use this capability your environment must be in the US region.

Copilot chat in model-driven apps is a preview feature. Preview features aren't meant for production use and may have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.

Copilot chat is not supported with the Power Apps mobile app.

This capability may be subject to usage limits or capacity throttling.

What operational factors and settings allow for effective and responsible use of the system?

Follow these procedures to make the most of the feature:

- Use everyday words to describe your question about the data:
  - "How many high priority tickets were created last month?"
  - "How many tickets were closed in July?"
- Clearly identify which page you want to navigate to:
  - "Take me to *page name*, " such as "Take me to Accounts."

Related information

Add Copilot chat to model-driven apps (preview)
Use Copilot chat in model-driven apps
Responsible AI FAQs for Power Apps

FAQ for Copilot control in canvas apps

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of Copilot control in canvas apps.

What is Copilot control in canvas apps?

Copilot control in canvas apps is a feature for app users to get insights about their data through conversation in natural language. Copilot control helps app users boost their productivity through AI-powered insights.

What are the system's capabilities?

It can answer questions about the data configured for the Copilot control in a canvas app.

What is the system's intended use?

As an AI assistant to end users, it provides answers to questions about the data configured for the Copilot control in a canvas app.

How was Copilot control in canvas apps evaluated? What metrics are used to measure performance?

Copilot control in canvas apps underwent substantial testing before the feature was released in preview. If you encounter issues while using Copilot control in canvas apps, please submit feedback. Your feedback will be used to improve Microsoft products and services. For more information, read the Privacy Statement.

What are the limitations of Copilot control in canvas apps? How can users minimize the impact of the Copilot control in canvas apps limitations when using the system?

To use this capability your environment must be in the US region. Preview features aren't meant for production use and may have restricted functionality. These features are available

before an official release so that customers can get early access and provide feedback. This capability may be subject to usage limits or capacity throttling. Currently, the Copilot control can only answer questions for smaller datasets. The only exception to this is when the Copilot control is configured with Dataverse.

What operational factors and settings allow for effective and responsible use of the system?

Follow these steps to make the most of the feature:

As a maker, you can:

- Configure the Copilot control to the appropriate data source, keeping in mind its limitation about smaller datasets except when configured with Dataverse.
- Edit the Copilot control properties to clarify what the control has been connected to and what it can answer questions about

As an app user, use everyday words to describe your question about the data such as:

- How many high priority tickets were created last month?
- How many tickets were closed in July?

See also

- Add Copilot control to a canvas app (preview)

FAQ about using Copilot in the rich text editor

Article • 05/28/2025

Here are some common questions and answers about using Copilot in the rich text editor.

What is Copilot in the rich text editor?

Copilot is a feature that uses AI to suggest content ideas based on your input. It helps you draft better content faster.

How does Copilot work?

Copilot uses advanced language models to generate content suggestions that match your tone, style, and purpose. You can choose from different types of content, such as emails, replies, or follow-ups. You can also adjust the tone and length of the content to suit your customers' preferences.

What can I use Copilot for?

You can use Copilot for various scenarios where you need to communicate with your customers effectively. For example, you can use Copilot to:

- Quickly create emails to follow up, reply to inquiries, and move a deal forward.
- Write engaging and personalized messages that resonate with your customers.
- Save time and energy by using ready-made content suggestions.

How do I use Copilot?

Using Copilot is easy and intuitive. Here are the steps to use Copilot in the rich text editor:

- Start typing your content in the rich text editor. You can use a word, a phrase, or a sentence as your input.
- Wait for Copilot to generate content suggestions based on your input. You can see the suggestions in a pop-up window next to your cursor.
- Choose the suggestion that best fits your needs. You can select on it to insert it into your content, or you can copy and paste it.
- Review and edit the content as needed. You can change the wording, add details, or delete parts of the content. Make sure the content is accurate, relevant, and appropriate

for your customers.
- Send the content to your customers when you're satisfied.

How did you evaluate Copilot?

We tested and evaluated Copilot on various datasets and scenarios to ensure its quality and performance. We used the following metrics to measure Copilot's performance:

- Accuracy: How well the content suggestions match the input and the context
- Relevance: How well the content suggestions address the customer's needs and expectations
- Diversity: How well the content suggestions offer different options and perspectives
- Readability: How easy the content suggestions are to read and understand

What are the limitations of Copilot in the rich text editor?

Copilot is a powerful and helpful feature, but it isn't perfect. It has some limitations that you should be aware of and respect. These limitations include:

- Language: Copilot only supports the English language for now. If you need to write in another language, you can't use Copilot.
- Suggestions: Copilot only provides suggestions, not final content. You're responsible for reviewing and editing the content before you send it to your customers. You shouldn't rely on Copilot to write your content for you.
- Ethics: Copilot is designed to help you create ethical and responsible content. However, Copilot may not be able to detect or prevent offensive, harmful, or misleading content. You should always use your own judgment and follow your company's policies and values when using Copilot.

Are there any settings I should change for effective and responsible use of Copilot in the rich text editor?

- Copilot in the rich text editor doesn't provide a way to change its behavior.
- Follow these suggestions to make the most of Copilot in the rich text editor:
    - Choose the right prompt to generate content and choose the right tone and length for the content. You can enter a group of keywords or a full sentence, but keep all the keywords on the same topic.

- Always review and edit Copilot's suggestions before sending them to your customer.

See also

Use Copilot in the rich text editor for email

FAQ for Excel to table and app

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of the Excel to table and app feature.

What is Excel to table and app?

Excel to table and app allows makers to bring their own Excel file or SharePoint list and convert that to a Dataverse table and an app along with it. The feature accepts an Excel file or SharePoint list and produces a Dataverse table as the output. When a maker uses the feature from the home page or the **Apps** area, it can also generate an app based on the Dataverse table. When Copilot is enabled in the current environment, it leverages Azure OpenAI Service to generate the best-fitting table and column metadata based on the uploaded Excel file or SharePoint list.

What can Excel to table and app do?

Excel to table and app converts the first range of tabular data from the first tab of the uploaded Excel file or SharePoint list to a Dataverse table by generating column metadata and ingesting data to the created table. When Copilot capability is enabled in this environment, Copilot assists with the table creation by generating table names, descriptions, along with column names and data types based on the uploaded Excel data or SharePoint list.

What is/are Excel to table and app's intended use(s)?

Excel to table and app is intended to assist makers to migrate their existing data from Excel into Power Platform. It ensures new or less experienced makers can move away from their existing Excel file based system or SharePoint lists to Dataverse tables and be put in the position of having a working app connected to that data so they can understand how apps in Power Apps are built.

How was Excel to table and app evaluated? What metrics are used to measure performance?

We have set up reports and dashboards to track the usage, net promoter score (NPS), and engagement of the feature. We have also set up monitors to track the success rate of table

creation, data ingestion, and AI inference. Top errors that occur to the users are logged as part of our monitors to help us understand what's causing the issues.

What are the limitations of Excel to table and app? How can users minimize the impact of Excel to table and app's limitations when using the system?

- When Copilot is enabled for the environment, Excel to table and app currently produces metadata output in U.S. English, Czech, Danish, German, Spanish, Finnish, French, Italian, Dutch, Norwegian, Polish, Portuguese, Swedish, or Turkish. Users can manually update their metadata before creating their tables and apps to minimize the impact.
- The Copilot capability of Excel to table and app is currently supported in limited regions. Users located in regions that don't currently support this Copilot feature are still able to use the upload and table and app creation feature without Copilot assistance.
- The autogenerated metadata of Excel to table and app might not be 100% accurate. Users can minimize the impact by reviewing the metadata before table and app creation.

What operational factors and settings allow for effective and responsible use of Excel to table and app?

- Users can configure the metadata settings of the table including display name, plural name, description, primary column, and schema name. This affects the metadata settings of the table when the table is created.
- Users can configure the metadata settings of the columns of the table including display name, data type, format, whether the column is required, schema name, and some other settings specific to each selected data type, such as choices and default choice when the datatype is choices.
- Users can configure whether the table should use first row of data as column headers. When this is turned on, the first row of the uploaded Excel file or SharePoint list is used as the header of the table.
- Users can configure the ownership of the table. Ownership depends on whether the data of the table is owned by specific users, teams, or the entire organization.

- The Excel file uploaded by users is stored in Dataverse and isn't deleted automatically after table creation. If an Excel file contains personal data and a user wants to delete it, they must manually delete it from the File Upload table. Open the **File Upload** table in Power Apps (make.powerapps.com), select **Edit** in the **File Upload columns and data** area, and then delete the Excel file from the table.

How can I use this feature in a Power Platform environment that is created outside of the United States or Switzerland?

To do this, turn on data movement across regions. More information: Enable copilots and generative AI features outside United States and Switzerland

See also

Create with external data

FAQ for SharePoint List to table and app

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of the SharePoint List to table and app feature.

What is SharePoint List to table and app?

SharePoint List to table and app allows makers to bring their own SharePoint List and convert that to a Dataverse table and an app along with it. The feature accepts a SharePoint List and produces a Dataverse table as the output. When a maker uses the feature from the home page or the Apps area, it can also generate an app based on the Dataverse table. When Copilot is enabled in the current environment, it leverages Azure OpenAI Service to generate the best-fitting table and column metadata based on the selected SharePoint List.

What can SharePoint List to table and app do?

SharePoint List to table and app converts the selected SharePoint List to a Dataverse table by generating column metadata and ingesting data from the SharePoint List to the created Dataverse table. When Copilot capability is enabled in this environment, Copilot assists with the table creation by generating column descriptions and data types based on the selected SharePoint List. Copilot will not read all of the customer data to generate the metadata. Copilot will not change any of the customer data from the original source.

What is/are SharePoint List to table and app's intended use(s)?

SharePoint List to table and app is intended to assist makers to migrate their existing data from SharePoint List into Power Platform. It ensures new or less experienced makers can move away from their existing SharePoint based system to Dataverse tables and be put in the position of having a working app connected to that data so they can understand how apps in Power Apps are built.

How was SharePoint List to table and app evaluated? What metrics are used to measure performance?

We have set up reports and dashboards to track the usage, net promoter score (NPS), and engagement of the feature. We have also set up monitors to track the success rate of table creation, data ingestion, and AI inference. Top errors that occur to the users are logged as part of our monitors to help us understand what's causing the issues.

What are the limitations of SharePoint List to table and app? How can users minimize the impact of SharePoint List to table and app's limitations when using the system?

When Copilot is enabled for the environment, SharePoint List to table and app will produce metadata output according to user's language setting if it's configured to be U.S. English, Czech, Danish, German, Spanish, Finnish, French, Italian, Dutch, Norwegian, Polish, Portuguese, Swedish, or Turkish. If user is not using one of the supported languages, Copilot will produce metadata in U.S. English by default. User can manually update their metadata before creating their tables and apps to minimize the impact.

The Copilot capability of SharePoint List to table and app is currently supported in limited regions. Users located in regions that don't currently support this Copilot feature are still able to use the SharePoint List to table and app feature without Copilot assistance.

The autogenerated metadata of SharePoint List to table and app might not be 100% accurate. Users can minimize the impact by reviewing the metadata before table and app creation.

What operational factors and settings allow for effective and responsible use of SharePoint List to table and app?

Users can configure the metadata settings of the table including display name, plural name, description, primary column, and schema name. This affects the metadata settings of the table when the table is created.

Users can configure the metadata settings of the columns of the table including display name, data type, format, whether the column is required, schema name, and some other settings specific to each selected data type, such as choices and default choice when the datatype is choices.

Users can configure the ownership of the table. Ownership depends on whether the data of the table is owned by specific users, teams, or the entire organization.

How can I use this feature in a Power Platform environment that is created outside of the United States or Switzerland?

To do this, turn on data movement across regions. More information: Enable copilots and generative AI features outside United States and Switzerland

See also

Create with external data

FAQ about form fill assistance

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of form fill assistance in model-driven apps.

What is form fill assistance?

Form fill assistance provides users AI-generated optional suggestions for fields in a form based on the data they frequently use, or based on text or image they have copied and used smart paste (via button or keyboard shortcut) to paste into the form, or based on the file contents the user provides through the files capability. These suggestions are entirely optional, and don't get saved until the user explicitly accepts them.

What can form fill assistance do?

Based on user's recent usage of the app, or based on text or image they have copied and used smart paste (via button or keyboard shortcut) to paste into the form, or based on the file contents the user provides through the files capability, optional suggestions for empty form fields are shown. The user can choose to accept, reject, or ignore these suggestions. Ignoring these optional suggestions is equivalent to rejecting the suggestions. Suggestions also include citations for reference by users.

What is the intended use of form fill assistance?

Help users fill forms more efficiently by providing optional suggestions based on the user's recent app usage or based on text or image they have copied and used smart paste (via button or keyboard shortcut) to paste into the form, or based on the file contents the user provides through the files capability. Users always need to choose to accept these suggestions in order to save them, otherwise these suggestions automatically get discarded.

How was form fill assistance evaluated? What metrics are used to measure performance?

In our evaluations, we look at how good the suggestions are based on how often they're accepted in the form. We also look at how often suggestions are generated. Moreover, the suggestions are evaluated for satisfying Microsoft's responsible AI principles.

What are the limitations of form fill assistance? How can users minimize the impact of form fill assistance's limitations when using the system?

The suggestions are optional and AI-generated. Users should ensure these suggestions are accurate and appropriate before accepting them.

What operational factors and settings allow for effective and responsible use of form fill assistance?

Users are always required to review suggestions and explicitly accept the suggestions that are accurate and appropriate.

How do I provide feedback?

To provide feedback about this feature, users can provide feedback by providing a compliment, reporting a problem, or making a suggestion.

Related information

Use form filling assistance in model-driven apps

Responsible AI FAQ for Copilot in the timeline

05/28/2025

These frequently asked questions (FAQ) describe the AI impact in the timeline highlights feature.

What is Copilot in the timeline?

Copilot is a feature that uses AI to generate highlights based on timeline activities.

How does Copilot work?

Copilot uses advanced language models to generate content suggestions that refer to the activities in the timeline and generate highlights. Timeline highlights show details from activities such as notes, emails, tasks, appointments, phone calls, and conversations.

What can I use Copilot for?

You can use Copilot for various scenarios where you need to communicate with your customers effectively. For example, you can use Copilot to:

- Summarize activities in the timeline.
- Shorthand catchup on activities in the timeline for entities, such as Case, Account, Contact, Lead, Opportunity, and other custom entities where the timeline is available.
- Save time and energy by using ready-made activity highlights.

How do I use Copilot?

Using Copilot is easy and intuitive.

To use Copilot in the timeline:

1. Select the chevron in the **Timeline highlights** section, and then wait for Copilot to generate highlights based on activities in the timeline.
2. Do either of the following actions:
   a. Edit the highlights before reusing them: Select **Copy**, and then paste the highlights into a note and edit them as needed. You can change the wording, add details, or

delete parts of the content. Make sure the content is accurate, relevant, and appropriate for your customers.

b. Reuse the highlights as-is: Select **Copy**, and then paste the highlights in notes or wherever you want to share them.

3. Send the content to your collaborators.

How did you evaluate Copilot?

We tested and evaluated Copilot on various datasets and scenarios to ensure its quality and performance.

We used the following metrics to measure Copilot's performance:

- **Accuracy**: How well the generated highlights match the input and the context
- **Relevance**: How well the generated highlights address the customer's needs and expectations
- **Diversity**: How well the generated highlights offer different options and perspectives
- **Readability**: How easy the generated highlights are to read and understand

What are the limitations of Copilot in the timeline?

Copilot is a powerful and helpful feature, but it isn't perfect. It has some limitations that you should be aware of and respect. These limitations include:

- **Language**: Many languages are supported in Copilot scenarios. While the quality is expected to be highest in English, the quality in the other languages is expected to improve over time.

  The following languages are available for Copilot in the timeline:
  - Danish
  - Dutch
  - English
  - French
  - German
  - Italian
  - Japanese
  - Portuguese (Brazil)
  - Spanish
  - Thai

    - Simplified Chinese
    - Czech
    - Finnish
    - Greek
    - Korean
    - Norwegian (Bokmål)
    - Polish
    - Russian
    - Swedish
    - Turkish
    - Vietnamese

- **Suggestions**: Copilot only provides suggestions, not final content. You're responsible for reviewing and editing the content before you send it to your customers. You shouldn't rely on Copilot to write your content for you.

- **Ethics**: Copilot is designed to help you create ethical and responsible content. However, Copilot may not be able to detect or prevent offensive, harmful, or misleading content. You should always use your own judgment and follow your company's policies and values when using Copilot.

Are there any settings I should change for effective and responsible use of Copilot in the timeline?

Copilot in the timeline doesn't provide a way to change its behavior. Timeline highlights are generated based on the activities in timeline such as notes, emails, tasks, phone calls, appointments, and conversations.

Always review and edit Copilot's suggestions before sending them to your collaborators.

See also

Use Timeline highlights powered by generative AI

FAQ about using Copilot with Power Fx

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of using Copilot to write or explain Power Fx formulas.

What is Copilot for Power Fx?

Power Apps Studio has several features to help makers understand and write Power Fx. Makers can use the formula bar to explain code, or use Power Fx comments in the formula bar to describe a formula and receive a suggested Power Fx expression. Additionally, the inline action bar on supported controls provides the ability to describe a formula and receive a suggested formula and a suggested property to apply the formula to.

What are the system's capabilities?

- Write a comment in the formula bar that describes a formula to receive a suggested Power Fx expression.
- Receive an explanation for the current formula displayed in the formula bar.
- Open Copilot from a supported control's inline action bar to describe a formula and receive a suggested formula and suggested property to apply it to.

What is the system's intended use?

- Generate a single Power Fx expression from an English description.
- Generate an English description of the formula shown in the formula bar.

What languages are supported?

Because Power Fx can be used across many different features, each generally available (GA) feature may have a different set of supported languages the meet Microsoft's responsible AI standards. For details on language support for specific features see, Create Power Fx formulas with Copilot

How was Power Fx with Copilot evaluated? What metrics are used to measure performance?

We qualitatively test the model's Power Fx formula generation and descriptions against an internally built set of Power Fx formulas and descriptions. Additionally, we're monitoring telemetry data to track the number of makers who tried the feature, the success rate of the feature, and the ratio of positive to negative feedback. Before releasing the Copilot feature in preview, we conducted extensive testing to ensure its functionality. If you encounter any issues with the content generated, provide feedback. Your feedback is used to enhance Microsoft's products and services. Your organization's IT admins have access to your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of Power Fx with Copilot? How can users minimize the impact of Power Fx Copilot limitations when using the system?

- Named Formulas and User Defined Functions can't be generated with Copilot at this time.
- Variables have to be in use (declared) already for Copilot to use them, it doesn't introduce new variables.
- Preview features aren't meant for production use and might restrict functionality. These features are available before an official release so that customers can get early access and provide feedback.
- More information available in preview terms.
- This capability is powered by Azure OpenAI Service.
- This capability is in process of rolling out, and might not be available in your region yet.
- This capability can be subject to usage limits or capacity throttling.
- Your environment must be in the United States region.
- This feature doesn't support non-English language input.

What operational factors and settings allow for effective and responsible use of the system?

Here are some tips to help you get the most out of this feature:

- Use simple language to explain the formula you're looking for, such as "if myvariable is true, return green" or "filter mylist on IDs that are larger than 5".
- If the results you get aren't what you expected, try being more precise with exact names of variables, controls and field names, or try rephrasing the question.
- If you still can't get the desired results, send us your feedback.

FAQs for field suggestions by Copilot

05/28/2025

These frequently asked questions (FAQs) describe the AI impact of field suggestions by Copilot in Power Apps.

What is field suggestions by Copilot?

In Power Apps Studio, when a maker connects a gallery or form to a data table, Power Apps automatically selects certain system-generated key columns, such as ID, created on, or modified on. These columns might not be relevant to the app users. Moreover, makers have to manually select each field, which can be a tedious process for large data tables. With field suggestions, Copilot can recommend relevant columns to the gallery or form when the developer binds it to data using the table's metadata. This feature can save makers time in field configuration, thereby accelerating the app building process.

What are the system's capabilities?

At the time, a maker first binds to or changes data source of a gallery or form to a data table, Copilot will automatically suggest up to 10 of the most relevant fields. A maker must approve the suggestions for it to take effect. Makers can adjust the results easily as they change fields today.

Supported controls:

- Gallery
- Form (classic)
- Form (modern)
- Table (modern)
- Supported views and forms (model-driven apps)
- Quick view form
- Quick create form
- New view

Supported data sources:

- Dataverse
- SQL
- SharePoint list

What is the system's intended use?

The system is intended to help makers save time in app editing by preloading and selecting the most relevant fields when data source change happens.

How was field suggestions by Copilot evaluated? What metrics are used to measure performance?

We evaluate the feature both qualitatively and quantitatively. To assess the quality of the feature, we're conducting user studies with makers to gather their feedback on their experiences, thoughts about the feature's quality, and suggestions for improvement. Additionally, we're monitoring telemetry data to track the number of makers who tried the feature, the success rate of the feature, and the ratio of positive to negative feedback. Before releasing the Copilot feature, we conducted extensive testing to ensure its functionality. If you encounter any issues with the content generated, provide feedback. Your feedback is used to enhance Microsoft's products and services. Your organization's IT admins have access to your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of field suggestions by Copilot? How can users minimize the impact of field suggestions by Copilot limitations when using the system?

- Copilot will only use the table's metadata to make recommendations, it doesn't allow maker to provide natural language input to specify what columns they want. If you think this is critical, send us feedback.
- This capability is powered by Azure OpenAI Service.
- This capability is in the process of rolling out and might not be available in your region yet.
- This capability may be subject to usage limits or capacity throttling.
- Your environment must be in the United States region.
- This feature doesn't support non-English language input.
- This feature only supports limited controls and actions as listed in the intended use section. - Microsoft is actively working on expanding the scope of this feature and support more actions incrementally.

What operational factors and settings allow for effective and responsible use of the system?

Copilot will only read your table's metadata (so table name, description, column name, description, column's data type) to make recommendations and select fields. To yield the best quality, it's strongly recommended that when you create your data source, use meaningful names and descriptions for both the table and the columns.

See also

Use field suggestions by Copilot

Column suggestions by Copilot

FAQ on filtering, sorting, and searching Power Apps galleries with Copilot

05/25/2025

These frequently asked questions (FAQ) describe the AI effect of filtering, sorting, and searching Power Apps canvas galleries with Copilot.

What is filtering, sorting, and searching with Copilot?

Filtering, sorting, and searching with Copilot is a feature for users of Power Apps who want to filter their galleries using natural language. It surfaces in galleries of canvas apps that use SharePoint as their backend. There are plans to expand to other data sources in the future.

What are the system’s capabilities?

It takes the user's input and uses advanced language models to filter, sort, and search the gallery. The user can then add more filters, view and remove current filters, or accept to persist the filtered view.

What is the system’s intended use?

As an AI assistant to users, it filters the gallery based on the user's input.

How was filtering, sorting, and searching with Copilot evaluated? What metrics are used to measure performance?

We evaluate the feature both qualitatively and quantitatively. To assess the quality of the feature, we're conducting studies with users to gather feedback on their experiences, thoughts about the feature's quality, and suggestions for improvement. We're also monitoring customer data to track the number of users who tried the feature, the success rate of the feature, and the ratio of positive feedback to negative feedback.

Before releasing the feature in preview, we conducted extensive testing to ensure its functionality. If you encounter any issues with your experience, provide feedback. Your feedback is used to enhance Microsoft's products and services. Your organization's IT admins

have access to your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of filtering, sorting, and searching with Copilot? How can users minimize the effect of the filtering, sorting, and searching with Copilot limitations when using the system?

- To use this capability, you must be a premium user.
- Preview features aren't meant for production use and might have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.
- For more information, go to preview terms.
- This capability is powered by Azure OpenAI Service.
- This capability might be subject to usage limits or capacity throttling.
- Your environment must be in the United States region or the **Move data across regions** checkbox must be selected in Power Platform admin center.
- This feature doesn't support non-English language input.

What operational factors and settings allow for effective and responsible use of the system?

If you get this error, **Sorry, Copilot can't help right now. Try again later.**, this might be due to capacity limits. We recommend that you give the system some time before trying again.

FAQ for agent builder in canvas apps

08/14/2025

These frequently asked questions (FAQ) describe the AI impact of agent builder in canvas apps.

What is agent builder?

Agent builder lets users create copilot agents to automate their processes. The goal of agent builder is to help users create copilot agents using their existing applications. Agent builder takes your app metadata and the desired agent goal to create a step-by-step process the user currently takes to complete the process they're looking to automate. This process is then combined with deterministically extracted skills from the app to create a Copilot Studio copilot on the user's behalf with instructions, actions, triggers, and knowledge. Once created, the user can edit, test, and publish the agent through Microsoft Copilot Studio.

What can agent builder do?

Agent builder can help a user create an agent. Agent builder can generate suggestions for which processes within an app we can automate. Using the suggestion and/or the sentence the user provided for what the desired goal for the agent is agent builder provides a process for how the user currently completes the task. Agent builder also extracts actions, triggers, and knowledge from the app metadata. Using the process instructions generated from agent builder and the triggers, knowledge, and actions extracted from agent builder the user can edit, test, and publish the agent in Microsoft Copilot Studio.

What is the intended use of agent builder?

The intended use of agent builder is to create agents to automate processes with a focus on data entry. The system allows the user to automate the process of data through generating an agent that takes on the steps the user would take to accomplish the task. The system wasn't designed, and we don't recommend customers to create agents for medical, legal, or hiring purposes. Agent builder was also not designed to be used to create an agent that has nothing to do with the app or generate an agent from an app that has no content.

How was agent builder evaluated? What metrics are used to measure performance?

We qualitatively tested the model's agent generation capabilities by testing against an internal set of benchmark applications and expected high quality agent outputs. Additionally, we have set up reports and dashboards to track the usage, net promoter score (NPS), and engagement of the feature. We have also set up monitors to track the success rate of agent generation, app understanding, and AI inference. Top errors that occur to the users are logged as part of our monitors to help us understand what's causing the issues. If you encounter any issues with the content generated, provide feedback. Your feedback is used to enhance Microsoft's products and services. Your organization's IT admins have access to your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of agent builder? How can users minimize the impact of agent builder's limitations when using the system?

- This feature is only currently available for Power Apps canvas applications.
- This capability is powered by Azure OpenAI Service.
- This capability is in the process of rolling out and might not be available in your region yet.
- This capability can be subject to usage limits or capacity throttling.

What operational factors and settings allow for effective and responsible use of agent builder?

Follow these procedures to make the most out of the feature:

- Use everyday words to describe the desired goal for agent:
  - "Submit completed claim forms to the database for processing"
  - "Generate reports for claims filed within a specific date range"
- Review all suggestions for accuracy and appropriateness before proceeding to the next step.

Authenticating actions after publishing

If your agent is missing authentication to perform an action or is configured to request user authentication, it sends a message to the user asking for credentials. If an agent's flow is

interrupted because it can't receive information or an action failed, it can't continue the session. If you want your agent to run autonomously, each action must be configured with working maker authentication that doesn't require user input. You can also instruct your agent to not request credentials from users.

Because triggers use maker authentication, be aware of what data potential users can access through a published agent that has triggers. See the Data protection with triggers section for more information.

Related information

Build an agent to automate your business process

FAQ for plans in Power Apps

Summarize this article for me

What is plans?

Plans in Power Apps is an AI-powered experience that builds an end-to-end business solution in minutes. Describe your business problem in natural language and include images, like a screenshot of a legacy app or a process diagram. Plans outlines user roles and requirements, and generates a complete Power Platform solution with objects like Dataverse tables, Power Apps (canvas and model-driven apps), Power Automate cloud flows, Power Pages sites, Power BI, and Microsoft Copilot Studio agents. You can iterate and refine your business requirements in your plan to get a more precise solution that fits your needs. In the latest preview features, makers will be able to see a fully functioning code app generated by AI that will directly address the business problem and saving makers more time.

What can plans do?

Plans builds an end-to-end business solution based on a description of a business problem and can include images, like a legacy app or process diagram. A complete plan outlines user roles and requirements, and includes an end-to-end Power Platform solution with objects like Dataverse tables, Power Apps (canvas and model-driven apps), Power Automate cloud flows, Power Pages, Power BI, and Microsoft Copilot Studio agents. It can also generate a code app based on the business requirements and data objects using the latest preview features.

What is plans intended use?

Plans lets makers describe their business problem and quickly get a comprehensive plan and a working code app. The plan includes user stories, data schema, and apps, flows, and agents to build. This approach saves time and helps users learn and integrate different Power Platform products.

The plan also serves as documentation that users can refer to later.

How was plans evaluated? What metrics are used to measure performance?

We evaluate the feature both qualitatively and quantitatively. To check quality, we conduct user studies with makers to get feedback on their experiences, thoughts about the feature's quality,

and suggestions for improvement. Also, monitor telemetry data to track the number of makers who try the feature, the feature's success rate, and the ratio of positive to negative feedback. Before releasing the feature, run extensive testing to check its functionality. If you find any issues with the generated content, provide feedback. Microsoft uses your feedback to improve products and services. Your organization's IT admins can access your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of plans? How can users minimize the impact of system or product name's limitations when using the system?

- To use this capability, you need a Microsoft Dataverse database in your environment.
- This capability is powered by Azure OpenAI Service.
- This capability can be subject to usage limits or capacity throttling.
- This feature lets you use a maximum input size of 4,000 tokens, including both text and images (about 3,000 words if your input only has text).
- ThePower Apps vibe experience is onlyavailable in English.

What operational factors and settings allow for effective and responsible use of plans?

- Use these tips to get the most out of this feature:
- Clearly describe the business problems end to end, including the types of user involved and the tasks they need to finish.
- If you upload an image, explain what the image shows in the text and explicitly ask plan designer to use it. For example, say "Use the ERD attached."
- If the results aren't what you expect, keep iterating the plan by selecting the specific part you want to edit and typing your requirement.
- Try the default suggestions in the banner, then customize them to suit your needs. If you still can't get the results you want, send us your feedback.

How do I provide feedback for plans?

- Select the thumbs up or thumbs down in the plan document to provide feedback on a specific section.
- Select the **Give feedback** icon in the command bar to provide general feedback for your experience usings plans.

See also

Overview of plans

Last updated on 03/04/2026

FAQ for rename controls in canvas apps with Copilot

05/25/2025

These frequently asked questions (FAQ) describe the AI impact of Power Apps rename controls in canvas apps with Copilot feature.

What is rename Canvas controls with Copilot?

With rename canvas controls with Copilot feature, makers can delegate renaming Canvas controls during app development to an AI assistant. When makers rename a control manually, they can expect Copilot to show up with renaming suggestions for other controls with valid property value. Makers can preview the suggestions before applying, ensuring full control over Copilot's suggestions. This feature's goal is to save makers time and effort otherwise spent manually renaming controls one by one, ensuring app maintainability and easy reference of controls in logic.

What are the system’s capabilities?

The following system capabilities are supported:

- Provide control rename suggestions based on primary property of the controls and naming convention/pattern used by maker during manual rename.
- Enable makers to apply all or selected rename suggestions.
- Apply rename to multiple controls at once, per the suggestions shown in preview.
- Copilot nudges the maker when they manually rename a control, and there are other controls of the same type under the same screen with valid, nondefault value such as **Button** or **Text** in **Text** property.
- Text property can be a Text literal or a formula returning Text type.
- Any formulas referencing the control name is automatically updated to use the rename.

The following controls are supported:

- Button
- Label

Support for more controls is coming soon.

What is the system’s intended use?

As an AI assistant that helps people rename Canvas controls, Copilot provides the following list of UI actions for makers that are included in the preview release scope:

- Click on Copilot icon when it appears near Tree View after a manual rename action.
- Copilo only appears if there are other controls of the same type under the same screen with valid, nondefault value such as **Button** or **Text** in **Text** property.
- Preview control renaming suggestions provided by Copilot.
- Apply one or more rename suggestions at once.

How was rename Canvas controls with Copilot evaluated? What metrics are used to measure performance?

We evaluate the feature both qualitatively and quantitatively. To assess the quality of the feature, we're conducting user studies with makers to gather their feedback on their experiences, thoughts about the feature's quality, and suggestions for improvement. Additionally, we're monitoring telemetry data to track the number of makers who tried the feature and the success rate and acceptance rate of the feature. Before releasing the Copilot feature in preview, we conducted extensive testing to ensure its functionality. If you encounter any issues with the content generated, provide feedback. Your feedback is used to enhance Microsoft's products and services. Your organization's IT admins have access to your feedback data for management purposes. For more information, read the Privacy Statement.

What are the limitations of rename Canvas controls with Copilot? How can users minimize the impact of rename Canvas controls with Copilot limitations when using the system?

- Preview features aren't meant for production use and might have restricted functionality. These features are available before an official release so that customers can get early access and provide feedback.
- For more information, see preview terms.
- This capability is powered by Azure OpenAI Service.
- This capability is in process of rolling out, and might not be available in your region yet.
- This capability can be subject to usage limits or capacity throttling.
- Your environment must be in the United States region.
- This feature doesn't support non-English language input.

- This feature only supports limited controls and actions as listed in the intended use section. Microsoft is actively working on expanding the scope of this feature and support more controls incrementally.

What operational factors and settings allow for effective and responsible use of the system?

Here are some tips to help you get the most out of this feature:

- Use this feature after Text property of controls has been set to ensure quality results from Copilot. For example, a button with default Text property value "Button" or invalid Text property value If() won't be considered for rename. And if you still can't get the desired results, send us your feedback.
- Use standard and relevant name and naming pattern when manually renaming a control so Copilot can reference it while suggesting renames. Avoid use of special characters and space. For example, renaming a button to "collectResponse" will provide better renaming suggestions for other buttons than "collect_1"
- Ensure New Analysis Engine app setting is enabled in your app. Go to Settings, click Updates, New, and then turn on New analysis engine toggle to true.

See also

Rename controls in canvas apps with Copilot (preview)

FAQ about Copilot visualizations on a view (preview)

05/28/2025

These frequently asked questions (FAQ) describe the AI impact of in visualizing data in a view.

What are Copilot visualizations on a view?

Copilot-generated visualization provides users with an option to view AI-generated chart for the rows in a view in a model-driven app. Users can select the Visualize button on the view page to see an AI-generated chart that aggregates the records in the view to show trends and distributions.

What can Copilot visualizations do?

Copilot-generated visualizations are generated based on the columns in the view. Users can change the chart or select areas of the chart to drill into the visualization generated by Copilot to do data analysis.

What is the intended use of Copilot visualizations?

Copilot-generated visualizations are intended for users to visualize the rows of data in a view as a chart. Users can apply filters or change chart type to personalize the chart, and interact with the chart and the tabular data side by side to explore the data in the app.

How were Copilot visualizations evaluated? What metrics are used to measure performance?

We continuously evaluated and tested Copilot visualization with different datasets and scenarios to measure quality and performance. We evaluated the accuracy and relevance of the chart generated for a view based on the type of data visualized. The type of charts generated were also evaluated for satisfying Microsoft's responsible AI principles.

What are the limitations of Copilot visualizations? How can users minimize the impact of the limitations when using the system?

Copilot visualizations are not shown to users by default. Users need to select the Visualize option on the page to see AI-generated chart. Copilot visualizations present users with a suggested chart for the data that users can review and edit. In addition, this capability is only available for environments with English as the base language.

What operational factors and settings allow for effective and responsible use of Copilot charts?

Users need to select the Visualize option on the page to see AI-generated chart. Copilot visualizations present users with a suggested chart for the data that users can review and edit.

How do I provide feedback on Copilot charts?

To provide feedback about this feature, users can select the thumbs up or thumbs down button and optionally, type in comments to describe their feedback and submit their response.

Related information

Visualize data in a view with Copilot

FAQ for Agent APIs and Agent Response component

06/18/2025

These frequently asked questions (FAQ) describe the AI impact of Agent APIs feature in model-driven apps.

What is Agent APIs and Agent Response component?

Microsoft Power Apps has introduced APIs and code components designed to interface with the topics created within Microsoft Copilot Studio (MCS) from model-driven apps. These APIs are available as XRM and PCF. The code component is available in the Form Designer to be added to forms.

What are capabilities of the Agent APIs and Agent Response component?

The new APIs can either call a specific topic in MCS or send a prompt that MCS orchestrates. The response from both APIs is a JSON structure the caller can use within their model driven app. The APIs are available for use in the Xrm events of model driven apps and also use within custom PCF control.

The Agent Response component provides a simplified component to call the Agent API with a topic for MCS. The component renders the results of the MCS response in the form.

What is the intended use of the Agent APIs and Agent Response component?

The APIs are intended to allow customization of model driven apps to include application calls to MCS and allow the results to be integrated into the app. It provides a simplified process to call MCS.

How was Agent APIs and Agent Response component evaluated? What metrics are used to

measure performance?

In our evaluations, we look at how accurately the response is from MCS. MCS responses are evaluated for satisfying Microsoft's responsible AI principles.

What are the limitations of Agent APIs and Agent Response component? How can users minimize the impact of the Agent APIs limitations when using the system?

The APIs and component are optional for makers to use and makers need to consider how best to use the APIs and component within their customizations.

What operational factors and settings allow for effective and responsible use of the feature?

Makers are required to ensure that the MCS topics are returning accurate and appropriate responses. Makers using the API are responsible to handle the response as part of their customization.

In addition, makers are strongly encouraged to indicate to users where AI has been used in the user experience. This is like how Microsoft shows "AI-generated content may be incorrect".

See also

- Bring intelligence into your component using Agent Xrm APIs (preview)
- Bring intelligence into your app using Agent Xrm APIs (preview)
- Use Agent Response component in the model-driven app forms (preview)

FAQ about generative pages in model-driven apps

These frequently asked questions (FAQ) describe the AI impact of generative pages in model-driven apps.

What are generative pages?

Generative pages simplify the app design process for model-driven app makers through AI code generation. By describing their needs in natural language, makers can create fully structured pages in their apps. The system processes these requirements and intelligently generates React code that covers both the frontend user experience and business logic, ensuring modern, consumer-grade UI experiences for enterprise applications. Makers must publish these pages before they're available to users in apps.

What can you do with generative pages?

With generative pages, makers can create and refine app pages in real time through an interactive, conversational experience. They can adjust elements, layout, and functionality to perfectly match their vision. This capability helps ensure designs adhere to best practices and deliver high-quality user experiences, making the app development process more efficient and intuitive. When makers are satisfied with the generated page, they can publish them so they're available when playing the app.

What is the intended use of generative pages?

The intended use of generative pages is to streamline the app design process for model-driven app makers. By leveraging AI to generate React code based on natural language descriptions, makers can quickly create pages that meet their specific requirements. This tool is designed to enhance productivity and ensure that app designs are both functional and visually appealing.

How were generative pages evaluated? What metrics are used to measure performance?

In our evaluations, we measure how often the agent can generate syntactically correct code, and we also test how good the code generation is in terms of UX design and meeting functional requirements. Moreover, the agent is evaluated for satisfying Microsoft’s responsible AI principles.

Are the images from the include images option AI-generated?

No. When the **Include images** option is enabled, the resulting images aren't AI-generated. They're sourced from a library of curated stock images. The option only decides whether images are needed based on the prompt and then selects the most relevant images from this library.

What are the limitations of generative pages? How can users minimize the impact of generative page limitations when using the system?

The current limitations of generative pages include:

- Pages can only connect to Dataverse tables and perform CRUD operations.
- Generative pages are only available in US environments.
- The code generated by generative pages isn't guaranteed to be production-ready or compliant with your organization's standards.
  - While the agent makes a best-effort attempt to produce high-quality code with accessibility and security best practices in mind, you're responsible for validating the code output to ensure it aligns with your organizational policies and compliance requirements.

Makers can minimize the impact of these limitations by testing any generated code before publishing it to users.

What operational factors and settings allow for effective and responsible use of generative pages?

Effective and responsible use of generative pages involves setting clear guidelines for app design requirements and ensuring that makers provide comprehensive descriptions. Regular training on best practices for using the tool and monitoring its performance can help maintain high standards. Additionally, incorporating user feedback and continuously improving the system based on real-world usage can enhance its effectiveness.

Makers are always required to publish generative pages before they're made available to users. Extensive testing of any code before making it available to users in an app is always encouraged.

Related articles

- Describe a page using natural language
- Legal terms

Last updated on 01/08/2026

FAQ about Power Apps MCP server invoke_data_entry tool

These frequently asked questions (FAQ) describe the AI impact of Power Apps MCP Server invoke_data_entry tool in Power Apps.

What is the invoke_data_entry tool in Power Apps MCP?

Create your own AI agent that utilizes the Power Apps MCP Server's invoke_data_entry tool to create records in Dataverse. This tool accepts various formats of unstructured input, such as PDF or Excel files, and creates draft records in Microsoft Dataverse based on the inputs. You can review and edit these draft records using the agent feed, and save to Dataverse once ready.

What can the invoke_data_entry tool do?

The invoke_data_entry tool can accept unstructured inputs, such as PDF or Excel files, parse them, and identify how to fill in Dataverse forms from these unstructured inputs in order to create a record. These are presented as suggestions for users to review in agent feed, where users can choose to save them to Dataverse. At this time, no records get automatically created in Dataverse.

What is the invoke_data_entry tool's intended use(s)?

The data entry agent (form fill) tool can take an unstructured input, such as PDF or Excel files, and create a Dataverse record based on its contents. This is a manual process today, but with the tool, customers can create an agent to automatically create a draft record. This draft can be reviewed in the agent feed, and once the user chooses to save, the record gets written to Dataverse.

How was the invoke_data_entry tool evaluated? What metrics are used to measure performance?

In our evaluations, we look at how good the suggestions are based on the provided input files, and how often they're accepted in the form to be saved in the record in Dataverse. Moreover, the suggestions are evaluated for satisfying Microsoft's responsible AI principles.

What are the limitations of the invoke_data_entry tool? How can users minimize the impact of invoke_data_entry tool's limitations when using the system?

At this time, the invoke_data_entry tool only supports a limited number of unstructured input formats, and only creates records in Dataverse. The tool isn't supported for use with an unsupported input format or for creating records in a non-Dataverse data source.

What operational factors and settings allow for effective and responsible use of the invoke_data_entry tool?

The invoke_data_entry tool is effective when used with the supported file formats provided as inputs, and is used to create records in Dataverse. Agent feed must be used by users to review invoke_data_entry tool's draft records and saving these to Dataverse once ready.

How do I provide feedback on the invoke_data_entry tool?

Users can provide feedback on the invoke_data_entry tool while reviewing the draft records in the agent feed.

Related articles

Add agents to your model-driven app (preview)

Work with Power Apps MCP server

Last updated on 02/11/2026