Tell us about your PDF experience.

Bing Search API documentation

The Bing Search APIs let you build web-connected apps and services that find webpages, images, news and more without advertisements.

OVERVIEW
Compare the Bing Search APIs

HOW-TO...
Page through search results

REFERENCE
Use and display requirements

REFERENCE
Use and display requirements of Bing Searc...

REFERENCE
API pricing details

Get started

Get started quickly searching the web using one of the following quickstarts.

Bing Web Search

- Use C#
- Use Java
- Use JavaScript
- Use Python
- Web Search tutorial

See more >

Bing Custom Search API

- Create a custom search instance
- Use C#
- Use Java
- Use Python
- Create a single page web-app

See more >

Bing Image Search

- Use C#
- Use Java
- Use JavaScript
- Use Python
- Create a single page web-app

See more >

Bing Entity Search API

- Use C#

Bing News Search API

- Use C#

Bing Video Search API

- Use C#

- Use Java
- Use JavaScript
- Use Python
- Create a single page web-app

See more >

- Use Java
- Use JavaScript
- Use Python
- Create a single page web-app

See more >

- Use Java
- Use JavaScript
- Use Python
- Create a single page web-app

See more >

Bing Visual Search API

- Use C#
- Use Java
- Use JavaScript
- Use Python
- Visual Search tutorial

See more >

Bing Autosuggest API

- Use C#
- Use Go
- Use Java
- Use Node.js
- Use Python
- Create a single page web-app

See more >

Bing Spell Check API

- Use C#
- Use Java
- Use Node.js
- Use Python
- Create a single page web-app

See more >

Reference

REST API v7

- Bing Web Search
- Bing Custom Search
- Bing Image Search
- Bing Entity Search
- Bing News Search
- Bing Video Search
- Bing Visual Search
- Bing Autosuggest
- Bing Spell Check

.NET SDK

- Bing Web Search
- Bing Custom Search
- Bing Entity Search
- Bing Image Search
- Bing News Search
- Bing Video Search
- Bing Visual Search
- Bing Spell Check
- Bing Autosuggest

Java SDK

- Bing Web Search
- Bing Custom Search
- Bing Entity Search
- Bing Image Search
- Bing News Search
- Bing Video Search
- Bing Visual Search
- Bing Spell Check
- Bing Autosuggest

Python SDK

- Bing Web Search

Node.js SDK

- Bing Web Search

Go SDK

- Bing Web Search

Bing Custom Search
Bing Entity Search
Bing Image Search
Bing News Search
Bing Video Search
Bing Visual Search

Bing Custom Search
Bing Entity Search
Bing Image Search
Bing News Search
Bing Video Search
Bing Visual Search
Bing Spell Check
Bing Autosuggest

Bing Custom Search
Bing Entity Search
Bing Image Search
Bing News Search
Bing Video Search
Bing Spell Check
Bing Autosuggest

What is the Bing Web Search API?

Article • 04/05/2022

Bing Web Search API enables safe, ad-free, location-aware search results, surfacing relevant information from billions of web documents. Help your users find what they're looking for from the world-wide-web by harnessing Bing's ability to comb billions of webpages, images, videos, and news with a single API call.

Get started

To get started using the API, pick the subscription you want from Bing API Pricing. After getting your subscription key, you're all set to make your first call.

You can easily call the API by sending a native HTTP GET request or by using the Web Search SDK. For examples to help you get up and running quickly for either option, see Quickstarts.

Features

By default, the API returns and ranks whatever content is relevant to the user's search query. But if you want to have some control over what Bing returns, see the following features:

Feature	Description
Filter the answers that bing returns	Filter the response to include or exclude specific answers such as news or images, return webpages that Bing discovered within the last week, and more.
Page results	Page through multiple pages of webpage results.
Hit highlighting	Add highlighting characters to words and phrases in the results' titles and descriptions that identify the words or phrases from the user's search query.

Bing also provides API metrics, which you can use to inform your strategic decisions. Quickly retrieve statistics such as your top queries, call volume, market distribution, response code summary, and much more. For details, see Bing Web Statistics.

Search or search-like experience

Bing Web Search API may only be used as a result of a direct user query or search, or as a result of an action within an app or experience that logically can be interpreted as a user's search request. For illustration purposes, the following are some examples of acceptable search or search-like experiences:

- User enters a query directly into a search box in an app.
- User selects specific text or image and requests "more information" or "additional information".
- User asks a search bot about a particular topic.
- User dwells on a particular object or entity in a visual search type scenario.

If you are not sure if your experience can be considered a search-like experience, check with Microsoft.

Next steps

- Learn about other APIs in the family of Bing Search APIs.
- Learn about use and display requirements for Bing Web Search.
- Learn about calling the API.
- Learn about what's in the JSON response.
- Review Web Search API v7 reference documentation.

Bing family of search APIs

Article • 03/27/2023

Bing provides a family of search APIs that let your users comb billions of web documents and get back safe, ad-free, location-aware search results. Depending on the API, you can get back all relevant web results or only relevant news, images, or videos.

The following table provides a summary of all APIs in the family of Bing Search APIs. For pricing and subscription details, see Bing API Pricing.

Bing API	Description
Autosuggest	Improves your users' search box experience by providing a list of suggested queries with each character they type.
Custom search	Enables you to create tailored, ad-free search experiences for topics that your users care about. You specify the domains and webpages that Bing searches.
Entity search	Finds information about a well-known person, place, or thing. Bing identifies the most relevant entity based on your searched term, spanning multiple entity types such as famous people, places, movies, TV shows, video games, books, and even local businesses near you.
Image search	Scours the web for images and trending images. Results include thumbnails, full image URLs, publisher, image metadata, and more. You can also filter images by size, color, license, freshness, and more.
News search	Finds news articles, trending news, headline news, and today's top stories.
Spell check	Helps users identify and fix spelling, grammar, slang, names, homonyms, and brands. Bing spell-checker leverages machine learning and statistical machine translation to provide accurate and contextual corrections.
Video search	Scours the web for videos and trending videos. Results include creator, encoding format, video length, view count, and more. You can also filter videos by pricing, video length, freshness, and more.
Visual search	Provides insights about an image such as getting visually similar images, and related searches.
Web search	Enables safe, ad-free, location-aware search results, surfacing relevant information from billions of web documents. Results include webpages, images, videos, news, and more.

Getting API metrics

Bing provides API metrics such as your top queries, call volume, market distribution, response code summary, and many more, which you can use to inform your strategic decisions. For details, see Bing Web Statistics.

Next steps

- Learn about use and display requirements for Bing search results.
- Learn about pricing and subscription options.

Bing Web Search quickstarts

Article • 02/16/2022

Use these quickstarts to make your first Web Search API call in a matter of minutes.

Quickstarts using native HTTP GET requests

- C#
- Go
- Java
- JavaScript
- PHP
- Python
- Ruby

Quickstarts using the Bing client library

- C#
- Java
- JavaScript
- Python

Next steps

- For a more in depth web app example, see the Web Search tutorial.
- For other REST and SDK samples, see Samples.

Quickstart: Search the web using C# and Bing Web Search API

Article • 02/16/2022

Use this quickstart to make your first call to Bing Web Search API. This C# console application sends a search request to Bing and parses the response. Since it's a console application, it displays a text-based version of the response for illustrative purposes only. The source code for this sample is available on GitHub.

Grab your favorite .NET editor, JSON library, and Create Bing Search Service resource for Bing Web Search and let's get started.

Create a project and declare dependencies

Create a new project and declare the code's dependencies. This example uses Newtonsoft to parse the JSON response. Use Newtonsoft's NuGet package to install its libraries.

C#

```
using System;
using System.Net.Http.Headers;
using System.Net.Http;
using System.Threading.Tasks;
using System.Linq;
using System.Collections.Generic;
using Newtonsoft.Json;
```

Declare a namespace and class for your program

Add a namespace and class. This example uses `WebSearchQuickstart` for the namespace and `Program` for the class.

C#

```
namespace WebSearchQuickstart
{
    class Program
    {
        // The code in the following sections goes here.
```

```
    }
}
```

Define variables

Add a few variables to the `Program` class. For simplicity, this example hardcodes the subscription key, but make sure you're pulling it from secured storage instead.

C#

```
        // In production, make sure you're pulling the subscription key from
secured storage.

        private static string _subscriptionKey = "<your key goes here>";
        private static string _baseUri =
"https://api.bing.microsoft.com/v7.0/search";

        // The user's search string.

        private static string searchString = "coronavirus vaccine";

        // Bing uses the X-MSEdge-ClientID header to provide users with
consistent
        // behavior across Bing API calls. See the reference documentation
        // for usage.

        private static string _clientIdHeader = null;
```

Here are all the query parameters you can add to the base URI. The *q* parameter is required and you should always include the *mkt* parameter too. The rest are optional. For information about these parameters, see Query parameters.

C#

```
        private const string QUERY_PARAMETER = "?q=";  // Required
        private const string MKT_PARAMETER = "&mkt=";  // Strongly suggested
        private const string RESPONSE_FILTER_PARAMETER = "&responseFilter=";
        private const string COUNT_PARAMETER = "&count=";
        private const string OFFSET_PARAMETER = "&offset=";
        private const string FRESHNESS_PARAMETER = "&freshness=";
        private const string SAFE_SEARCH_PARAMETER = "&safeSearch=";
        private const string TEXT_DECORATIONS_PARAMETER =
"&textDecorations=";
        private const string TEXT_FORMAT_PARAMETER = "&textFormat=";
        private const string ANSWER_COUNT = "&answerCount=";
        private const string PROMOTE = "&promote=";
```

Declare the Main method

Our `Main()` method is pretty simple since we're going to implement the HTTP requests asynchronously.

C#

```
        static void Main()
        {
            RunAsync().Wait();
        }
```

Where all the work happens

The `RunAsync` method is where all the work happens. It builds the query string that's appended to the base URI, waits for the asynchronous HTTP request to return, deserializes the response, and either prints the search results or an error message.

This example uses dictionaries instead of objects to access the response data.

C#

```
        static async Task RunAsync()
        {
            try
            {
                // Remember to encode query parameters like q,
responseFilters, promote, etc.

                var queryString = QUERY_PARAMETER +
Uri.EscapeDataString(searchString);
                queryString += MKT_PARAMETER + "en-us";
                //queryString += RESPONSE_FILTER_PARAMETER +
Uri.EscapeDataString("webpages,news");
                queryString += TEXT_DECORATIONS_PARAMETER +
Boolean.TrueString;

                HttpResponseMessage response = await
MakeRequestAsync(queryString);

                _clientIdHeader = response.Headers.GetValues("X-MSEdge-
ClientID").FirstOrDefault();

                // This example uses dictionaries instead of objects to
access the response data.

                var contentString = await
response.Content.ReadAsStringAsync();
```

```
                Dictionary<string, object> searchResponse =
JsonConvert.DeserializeObject<Dictionary<string, object>>(contentString);

                if (response.IsSuccessStatusCode)
                {
                    PrintResponse(searchResponse);
                }
                else
                {
                    PrintErrors(response.Headers, searchResponse);
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }

            Console.WriteLine("\nPress ENTER to exit...");
            Console.ReadLine();
        }
```

The HTTP call

Here's the HTTP request. It's your basic HTTP GET request. Use whatever HTTP client works for you.

C#

```
        // Makes the request to the Web Search endpoint.

        static async Task<HttpResponseMessage> MakeRequestAsync(string
queryString)
        {
            var client = new HttpClient();

            // Request headers. The subscription key is the only required
header but you should
            // include User-Agent (especially for mobile), X-MSEdge-
ClientID, X-Search-Location
            // and X-MSEdge-ClientIP (especially for local aware queries).

            client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
_subscriptionKey);

            return (await client.GetAsync(_baseUri + queryString));
        }
```

That's all there is to sending a search request and getting back search results. To see what all the answers look like in the JSON response, see Handling the web search

response.

The rest of the sections walk you through one way of parsing the JSON response and displaying the search results. Be sure to read the use and display requirements to make sure you comply with all display requirements.

Using ranking to display the search results

If the request succeeds, the code calls the `PrintResponse` method to print the search results in the console window.

The example uses the RankingResponse answer to display the search results. The ranking determines which answers to display in the pole, mainline, and sidebar sections of the search results page. For information about using the RankingResponse answer, see Use ranking to display search results.

C#

```
        // Prints the JSON response data for pole, mainline, and sidebar.

        static void PrintResponse(Dictionary<string, object> response)
        {
            Console.WriteLine("The response contains the following
answers:\n");

            var ranking = response["rankingResponse"] as
Newtonsoft.Json.Linq.JToken;

            Newtonsoft.Json.Linq.JToken position;

            if ((position = ranking["pole"]) != null)
            {
                Console.WriteLine("Pole Position:\n");
                DisplayAnswersByRank(position["items"], response);
            }

            if ((position = ranking["mainline"]) != null)
            {
                Console.WriteLine("Mainline Position:\n");
                DisplayAnswersByRank(position["items"], response);
            }

            if ((position = ranking["sidebar"]) != null)
            {
                Console.WriteLine("Sidebar Position:\n");
                DisplayAnswersByRank(position["items"], response);
            }
        }
```

Display all results for an answer or a single result

Each item in the ranking tells you whether to display all results from the answer together or to display a single result from the answer.

If the item includes the `resultIndex` field, use the index value to display that single result from the answer. But if the item doesn't include `resultIndex`, you display all results from the answer. Typically, the ranking has you display individual webpages interspersed among the other answers, and has you group all images together. For answers that have many results like images and videos, you typically display a few of the images or videos and provide a link that users can click to see the rest.

C#

```
        // Displays each result based on ranking. Ranking contains the
results for
        // the pole, mainline, or sidebar section of the search results.

        static void DisplayAnswersByRank(Newtonsoft.Json.Linq.JToken items,
Dictionary<string, object> response)
        {
            foreach (Newtonsoft.Json.Linq.JToken item in items)
            {
                var answerType = (string)item["answerType"];
                Newtonsoft.Json.Linq.JToken index = -1;

                // If the ranking item doesn't include an index of the
result to
                // display, then display all the results for that answer.

                if ("WebPages" == answerType)
                {
                    if ((index = item["resultIndex"]) == null)
                    {

DisplayAllWebPages(((Newtonsoft.Json.Linq.JToken)response["webPages"])
["value"]);
                    }
                    else
                    {

DisplayWegPage(((Newtonsoft.Json.Linq.JToken)response["webPages"])
["value"].ElementAt((int)index));
                    }
                }
                else if ("Images" == answerType)
                {
                    if ((index = item["resultIndex"]) == null)
                    {

DisplayAllImages(((Newtonsoft.Json.Linq.JToken)response["images"])
```

```
["value"]);
                    }
                    else
                    {

DisplayImage(((Newtonsoft.Json.Linq.JToken)response["images"])
["value"].ElementAt((int)index));
                    }
                }
                else if ("Videos" == answerType)
                {
                    if ((index = item["resultIndex"]) == null)
                    {

DisplayAllVideos(((Newtonsoft.Json.Linq.JToken)response["videos"])
["value"]);
                    }
                    else
                    {

DisplayVideo(((Newtonsoft.Json.Linq.JToken)response["videos"])
["value"].ElementAt((int)index));
                    }
                }
                else if ("News" == answerType)
                {
                    if ((index = item["resultIndex"]) == null)
                    {

DisplayAllNews(((Newtonsoft.Json.Linq.JToken)response["news"])["value"]);
                    }
                    else
                    {

DisplayArticle(((Newtonsoft.Json.Linq.JToken)response["news"])
["value"].ElementAt((int)index));
                    }
                }
                else if ("RelatedSearches" == answerType)
                {
                    if ((index = item["resultIndex"]) == null)
                    {

DisplayAllRelatedSearches(((Newtonsoft.Json.Linq.JToken)response["relatedSea
rches"])["value"]);
                    }
                    else
                    {

DisplayRelatedSearch(((Newtonsoft.Json.Linq.JToken)response["relatedSearches
"])["value"].ElementAt((int)index));
                    }
                }
                else if ("Entities" == answerType)
                {
```

```
                    if ((index = item["resultIndex"]) == null)
                    {

DisplayAllEntities(((Newtonsoft.Json.Linq.JToken)response["entities"])
["value"]);
                    }
                    else
                    {

DisplayEntity(((Newtonsoft.Json.Linq.JToken)response["entities"])
["value"].ElementAt((int)index));
                    }
                }
                else if ("Places" == answerType)
                {
                    if ((index = item["resultIndex"]) == null)
                    {

DisplayAllPlaces(((Newtonsoft.Json.Linq.JToken)response["places"])
["value"]);
                    }
                    else
                    {

DisplayPlace(((Newtonsoft.Json.Linq.JToken)response["places"])
["value"].ElementAt((int)index));
                    }
                }
                else if ("Computation" == answerType)
                {

DisplayComputation((Newtonsoft.Json.Linq.JToken)response["computation"]);
                }
                else if ("Translations" == answerType)
                {

DisplayTranslations((Newtonsoft.Json.Linq.JToken)response["translations"]);
                }
                else if ("TimeZone" == answerType)
                {

DisplayTimeZone((Newtonsoft.Json.Linq.JToken)response["timeZone"]);
                }
                else
                {
                    Console.WriteLine("\nUnknown answer type: {0}\n",
answerType);
                }
            }
        }
```

Display answer results

This example accesses a few of the fields from each type of answer result and applies any contractual rules. You will likely use data from more of the fields than are shown in this example. For information about the fields that each answer result may include, see Response objects.

C#

```
        // Displays all webpages in the Webpages answer.
        static void DisplayAllWebPages(Newtonsoft.Json.Linq.JToken webpages)
        {
            foreach (Newtonsoft.Json.Linq.JToken webpage in webpages)
            {
                DisplayWegPage(webpage);
            }
        }

        // Displays a single webpage.
        static void DisplayWegPage(Newtonsoft.Json.Linq.JToken webpage)
        {
            string rule = null;

            // Some webpages require attribution. Checks if this page
requires
            // attribution and gets the list of attributions to apply.

            Dictionary<string, string> rulesByField = null;
            rulesByField = GetRulesByField(webpage["contractualRules"]);

            Console.WriteLine("\tWebpage\n");
            Console.WriteLine("\t\tName: " + webpage["name"]);
            Console.WriteLine("\t\tUrl: " + webpage["url"]);
            Console.WriteLine("\t\tDisplayUrl: " + webpage["displayUrl"]);
            Console.WriteLine("\t\tSnippet: " + webpage["snippet"]);

            // Apply attributions if they exist.

            if (null != rulesByField)
            {
                if (rulesByField.TryGetValue("snippet", out rule))
                {
                    Console.WriteLine("\t\t\tData from: " +
rulesByField["snippet"]);
                }
            }

            Console.WriteLine();

        }

        // Displays all images in the Images answer.

        static void DisplayAllImages(Newtonsoft.Json.Linq.JToken images)
        {
```

```
            foreach (Newtonsoft.Json.Linq.JToken image in images)
            {
                DisplayImage(image);
            }
        }

        // Displays a single image.

        static void DisplayImage(Newtonsoft.Json.Linq.JToken image)
        {
            Console.WriteLine("\tImage\n");
            Console.WriteLine("\t\tThumbnail: " + image["thumbnailUrl"]);
            Console.WriteLine();
        }

        // Displays all videos in the Videos answer.

        static void DisplayAllVideos(Newtonsoft.Json.Linq.JToken videos)
        {
            foreach (Newtonsoft.Json.Linq.JToken video in videos)
            {
                DisplayVideo(video);
            }
        }

        // Displays a single video.

        static void DisplayVideo(Newtonsoft.Json.Linq.JToken video)
        {
            Console.WriteLine("\tVideo\n");
            Console.WriteLine("\t\tEmbed HTML: " + video["embedHtml"]);
            Console.WriteLine();
        }

        // Displays all news articles in the News answer.

        static void DisplayAllNews(Newtonsoft.Json.Linq.JToken news)
        {
            foreach (Newtonsoft.Json.Linq.JToken article in news)
            {
                DisplayArticle(article);
            }
        }

        // Displays a single news article.

        static void DisplayArticle(Newtonsoft.Json.Linq.JToken article)
        {
            // News articles require attribution. Gets the list of
attributions to apply.

            Dictionary<string, string> rulesByField = null;
            rulesByField = GetRulesByField(article["contractualRules"]);

            Console.WriteLine("\tArticle\n");
```

```
            Console.WriteLine("\t\tName: " + article["name"]);
            Console.WriteLine("\t\tURL: " + article["url"]);
            Console.WriteLine("\t\tDescription: " + article["description"]);
            Console.WriteLine("\t\tArticle from: " +
rulesByField["global"]);
            Console.WriteLine();
        }

        // Displays all related search in the RelatedSearches answer.

        static void DisplayAllRelatedSearches(Newtonsoft.Json.Linq.JToken
searches)
        {
            foreach (Newtonsoft.Json.Linq.JToken search in searches)
            {
                DisplayRelatedSearch(search);
            }
        }

        // Displays a single related search query.

        static void DisplayRelatedSearch(Newtonsoft.Json.Linq.JToken search)
        {
            Console.WriteLine("\tRelatedSearch\n");
            Console.WriteLine("\t\tName: " + search["displayText"]);
            Console.WriteLine("\t\tURL: " + search["webSearchUrl"]);
            Console.WriteLine();
        }

        // Displays all entities in the Entities answer.

        static void DisplayAllEntities(Newtonsoft.Json.Linq.JToken entities)
        {
            foreach (Newtonsoft.Json.Linq.JToken entity in entities)
            {
                DisplayEntity(entity);
            }
        }

        // Displays a single entity.

        static void DisplayEntity(Newtonsoft.Json.Linq.JToken entity)
        {
            string rule = null;

            // Entities require attribution. Gets the list of attributions
to apply.

            Dictionary<string, string> rulesByField = null;
            rulesByField = GetRulesByField(entity["contractualRules"]);

            Console.WriteLine("\tEntity\n");
            Console.WriteLine("\t\tName: " + entity["name"]);

            if (entity["image"] != null)
```

```
            {
                Console.WriteLine("\t\tImage: " + entity["image"]
["thumbnail"]);

                if (rulesByField.TryGetValue("image", out rule))
                {
                    Console.WriteLine("\t\t\tImage from: " + rule);
                }
            }

            if (entity["description"] != null)
            {
                Console.WriteLine("\t\tDescription: " +
entity["description"]);

                if (rulesByField.TryGetValue("description", out rule))
                {
                    Console.WriteLine("\t\t\tData from: " +
rulesByField["description"]);
                }
            }
            else
            {
                // See if presentation info can shed light on what this
entity is.

                var hintCount = entity["entityPresentationInfo"]
["entityTypeHints"].Count();
                Console.WriteLine("\t\tEntity hint: " +
entity["entityPresentationInfo"]["entityTypeHints"][hintCount - 1]);
            }

            Console.WriteLine();
        }

        // Displays all places in the Places answer.

        static void DisplayAllPlaces(Newtonsoft.Json.Linq.JToken places)
        {
            foreach (Newtonsoft.Json.Linq.JToken place in places)
            {
                DisplayPlace(place);
            }
        }

        // Displays a single place.

        static void DisplayPlace(Newtonsoft.Json.Linq.JToken place)
        {
            Console.WriteLine("\tPlace\n");
            Console.WriteLine("\t\tName: " + place["name"]);
            Console.WriteLine("\t\tPhone: " + place["telephone"]);
            Console.WriteLine("\t\tWebsite: " + place["url"]);
            Console.WriteLine();
        }
```

```
        // Displays the Computation answer.

        static void DisplayComputation(Newtonsoft.Json.Linq.JToken
expression)
        {
            Console.WriteLine("\tComputation\n");
            Console.WriteLine("\t\t{0} is {1}", expression["expression"],
expression["value"]);
            Console.WriteLine();
        }

        // Displays the Translation answer.

        static void DisplayTranslations(Newtonsoft.Json.Linq.JToken
translation)
        {
            // Some webpages require attribution. Checks if this page
requires
            // attribution and gets the list of attributions to apply.

            Dictionary<string, string> rulesByField = null;
            rulesByField = GetRulesByField(translation["contractualRules"]);

            // The translatedLanguageName field contains a 2-character
language code,
            // so you might want to provide the means to print Spanish
instead of es.

            Console.WriteLine("\tTranslation\n");
            Console.WriteLine("\t\t\"{0}\" translates to \"{1}\" in {2}",
translation["originalText"], translation["translatedText"],
translation["translatedLanguageName"]);
            Console.WriteLine("\t\tTranslation by " +
rulesByField["global"]);
            Console.WriteLine();
        }

        // Displays the TimeZone answer. This answer has multiple formats,
so you need to figure
        // out which fields exist in order to format the answer.

        static void DisplayTimeZone(Newtonsoft.Json.Linq.JToken timeZone)
        {
            Console.WriteLine("\tTime zone\n");

            if (timeZone["primaryCityTime"] != null)
            {
                var time =
DateTime.Parse((string)timeZone["primaryCityTime"]["time"]);
                Console.WriteLine("\t\tThe time in {0} is {1}:",
timeZone["primaryCityTime"]["location"], time);

                if (timeZone["otherCityTimes"] != null)
                {
```

```
                    Console.WriteLine("\t\tThere are {0} other time zones",
timeZone["otherCityTimes"].Count());
                }
            }

            if (timeZone["date"] != null)
            {
                Console.WriteLine("\t\t" + timeZone["date"]);
            }

            if (timeZone["primaryResponse"] != null)
            {
                Console.WriteLine("\t\t" + timeZone["primaryResponse"]);
            }

            if (timeZone["timeZoneDifference"] != null)
            {
                Console.WriteLine("\t\t{0} {1}", timeZone["description"],
timeZone["timeZoneDifference"]["text"]);
            }

            if (timeZone["primaryTimeZone"] != null)
            {
                Console.WriteLine("\t\t" + timeZone["primaryTimeZone"]
["timeZoneName"]);
            }

            Console.WriteLine();
        }
```

Handling contractual rules

The `GetRulesByField` method builds a dictionary of the rules that the calling method accesses when it displays the result. If the rule applies to the result as a whole, the key is `global`. Otherwise, the key is the name of the field that the rule targets (see the `targetPropertyName` field).

C#

```
        // Checks if the result includes contractual rules and builds a
dictionary of
        // the rules.

        static Dictionary<string, string>
GetRulesByField(Newtonsoft.Json.Linq.JToken contractualRules)
        {
            if (null == contractualRules)
            {
                return null;
            }
```

```
            var rules = new Dictionary<string, string>();

            foreach (Newtonsoft.Json.Linq.JToken rule in contractualRules as
Newtonsoft.Json.Linq.JToken)
            {
                // Use the rule's type as the key.

                string key = null;
                string value = null;
                var index = ((string)rule["_type"]).LastIndexOf('/');
                var ruleType = ((string)rule["_type"]).Substring(index + 1);
                string attribution = null;

                if (ruleType == "LicenseAttribution")
                {
                    attribution = (string)rule["licenseNotice"];
                }
                else if (ruleType == "LinkAttribution")
                {
                    attribution = string.Format("{0}({1})",
(string)rule["text"], (string)rule["url"]);
                }
                else if (ruleType == "MediaAttribution")
                {
                    attribution = (string)rule["url"];
                }
                else if (ruleType == "TextAttribution")
                {
                    attribution = (string)rule["text"];
                }

                // If the rule targets specific data in the result; for
example, the
                // snippet field, use the target's name as the key. Multiple
rules
                // can apply to the same field.

                if ((key = (string) rule["targetPropertyName"]) != null)
                {
                    if (rules.TryGetValue(key, out value))
                    {
                        rules[key] = value + " | " + attribution;
                    }
                    else
                    {
                        rules.Add(key, attribution);
                    }
                }
                else
                {
                    // Otherwise, the rule applies to the result. Uses
'global' as the key
                    // value for this case.

                    key = "global";
```

```
                    if (rules.TryGetValue(key, out value))
                    {
                        rules[key] = value + " | " + attribution;
                    }
                    else
                    {
                        rules.Add(key, attribution);
                    }
                }
            }

            return rules;
        }
```

Handling errors

This section shows an option for handling errors that the service may return. For example, the service returns an error if your subscription key is not valid or is not valid for the specified endpoint. The service may also return an error if you specify a parameter value that's not valid.

C#

```
        // Print any errors that occur. Depending on which part of the
service is
        // throwing the error, the response may contain different error
formats.

        static void PrintErrors(HttpResponseHeaders headers,
Dictionary<String, object> response)
        {
            Console.WriteLine("The response contains the following
errors:\n");

            object value;

            if (response.TryGetValue("error", out value))  // typically 401,
403
            {
                PrintError(response["error"] as
Newtonsoft.Json.Linq.JToken);
            }
            else if (response.TryGetValue("errors", out value))
            {
                // Bing API error

                foreach (Newtonsoft.Json.Linq.JToken error in
response["errors"] as Newtonsoft.Json.Linq.JToken)
                {
```

```
                    PrintError(error);
                }

                // Included only when HTTP status code is 400; not included
with 401 or 403.

                IEnumerable<string> headerValues;
                if (headers.TryGetValues("BingAPIs-TraceId", out
headerValues))
                {
                    Console.WriteLine("\nTrace ID: " +
headerValues.FirstOrDefault());
                }
            }

        }

        static void PrintError(Newtonsoft.Json.Linq.JToken error)
        {
            string value = null;

            Console.WriteLine("Code: " + error["code"]);
            Console.WriteLine("Message: " + error["message"]);

            if ((value = (string)error["parameter"]) != null)
            {
                    Console.WriteLine("Parameter: " + value);
            }

            if ((value = (string)error["value"]) != null)
            {
                Console.WriteLine("Value: " + value);
            }
        }
```

Next steps

- For a more in depth web app example, see the Web Search tutorial.

Quickstart: Search the web using the Bing Web Search REST API and Go

Article • 09/27/2022

Use this quickstart to make your first call to the Bing Web Search API. This Go application sends a search request to the API, and shows the JSON response. Although this application is written in Go, the API is a RESTful Web service compatible with most programming languages.

The code examples in this quickstart require only core libraries; there are no external dependencies.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- Go binaries
- A subscription key

Create a project and import core libraries

Create a new Go project in your favorite IDE or editor. Then, import `net/http` for requests, `ioutil` to read the response, `time` and `encoding/json` to handle the JSON, and `fmt` to print the output.

Go

```
package main
import (
    "fmt"
    "net/http"
    "io/ioutil"
    "time"
    "encoding/json"
)
```

Create a struct to format the search results

The `BingAnswer` struct formats the data provided in the response.

Go

```
// This struct formats the answers provided by the Bing Web Search API.
type BingAnswer struct {
        Type         string `json:"_type"`
        QueryContext struct {
                OriginalQuery string `json:"originalQuery"`
        } `json:"queryContext"`
        WebPages struct {
                WebSearchURL          string `json:"webSearchUrl"`
                TotalEstimatedMatches int    `json:"totalEstimatedMatches"`
                Value                 []struct {
                        ID               string    `json:"id"`
                        Name             string    `json:"name"`
                        URL              string    `json:"url"`
                        IsFamilyFriendly bool      `json:"isFamilyFriendly"`
                        DisplayURL       string    `json:"displayUrl"`
                        Snippet          string    `json:"snippet"`
                        DateLastCrawled  time.Time `json:"dateLastCrawled"`
                        SearchTags       []struct {
                                Name    string `json:"name"`
                                Content string `json:"content"`
                        } `json:"searchTags,omitempty"`
                        About []struct {
                                Name string `json:"name"`
                        } `json:"about,omitempty"`
                } `json:"value"`
        } `json:"webPages"`
        RelatedSearches struct {
                ID    string `json:"id"`
                Value []struct {
                        Text         string `json:"text"`
                        DisplayText  string `json:"displayText"`
                        WebSearchURL string `json:"webSearchUrl"`
                } `json:"value"`
        } `json:"relatedSearches"`
        RankingResponse struct {
                Mainline struct {
                        Items []struct {
                                AnswerType  string `json:"answerType"`
                                ResultIndex int    `json:"resultIndex"`
                                Value       struct {
                                        ID string `json:"id"`
                                } `json:"value"`
                        } `json:"items"`
                } `json:"mainline"`
                Sidebar struct {
                        Items []struct {
                                AnswerType string `json:"answerType"`
                                Value      struct {
                                        ID string `json:"id"`
                                } `json:"value"`
                        } `json:"items"`
                } `json:"sidebar"`
```

```
        } `json:"rankingResponse"`
}
```

Declare the main function and define variables

This code declares the main function and sets the required variables:

2. Confirm that the endpoint is correct and replace the `token` value with a valid subscription key from your Azure account.
3. Optionally, customize the search query by replacing the value for `searchTerm`.

Go

```
// Declare the main function. This is required for all Go programs.
func main() {
// Replace the token string with a valid subscription key.
    const endpoint = "https://api.bing.microsoft.com/v7.0/search"
    token := "YOUR-ACCESS-KEY"
    searchTerm := "Microsoft Bing Search Services"

// The remaining code in this quickstart goes in the main function.

}
```

Construct a request

This code declares the HTTP request, inserts the header and payload, and instantiates the client.

Go

```
// Declare a new GET request.
req, err := http.NewRequest("GET", endpoint, nil)
if err != nil {
    panic(err)
}

// Add the payload to the request.
param := req.URL.Query()
param.Add("q", searchTerm)
req.URL.RawQuery = param.Encode()

// Insert the request header.
req.Header.Add("Ocp-Apim-Subscription-Key", token)
```

```
// Instantiate a client.
client := new(http.Client)
```

Make a request

Use this code to call the Bing Web Search API and close the connection after a response is returned.

Go

```
// Send the request to Bing.
resp, err := client.Do(req)
if err != nil {
    panic(err)
}

// Close the connection.
defer resp.Body.Close()
body, err := ioutil.ReadAll(resp.Body)
if err != nil {
    panic(err)
}
```

Handle the response

Use the struct we created previously to format the response and print the search results.

Go

```
// Create a new answer.
ans := new(BingAnswer)
err = json.Unmarshal(body, &ans)
if err != nil {
    fmt.Println(err)
}
// Iterate over search results and print the result name and URL.
for _, result := range ans.WebPages.Value {
    fmt.Println(result.Name, "||", result.URL)
}
```

Put it all together

The last step is to validate your code and run it. If you'd like to compare your code with ours, here's the complete program:

Go

```
package main
import (
    "fmt"
    "net/http"
    "io/ioutil"
    "time"
    "encoding/json"
)

// The is the struct for the data returned by Bing.
type BingAnswer struct {
        Type         string `json:"_type"`
        QueryContext struct {
                OriginalQuery string `json:"originalQuery"`
        } `json:"queryContext"`
        WebPages struct {
                WebSearchURL          string `json:"webSearchUrl"`
                TotalEstimatedMatches int    `json:"totalEstimatedMatches"`
                Value                 []struct {
                        ID               string    `json:"id"`
                        Name             string    `json:"name"`
                        URL              string    `json:"url"`
                        IsFamilyFriendly bool      `json:"isFamilyFriendly"`
                        DisplayURL       string    `json:"displayUrl"`
                        Snippet          string    `json:"snippet"`
                        DateLastCrawled  time.Time `json:"dateLastCrawled"`
                        SearchTags       []struct {
                                Name    string `json:"name"`
                                Content string `json:"content"`
                        } `json:"searchTags,omitempty"`
                        About []struct {
                                Name string `json:"name"`
                        } `json:"about,omitempty"`
                } `json:"value"`
        } `json:"webPages"`
        RelatedSearches struct {
                ID    string `json:"id"`
                Value []struct {
                        Text         string `json:"text"`
                        DisplayText  string `json:"displayText"`
                        WebSearchURL string `json:"webSearchUrl"`
                } `json:"value"`
        } `json:"relatedSearches"`
        RankingResponse struct {
                Mainline struct {
                        Items []struct {
                                AnswerType  string `json:"answerType"`
                                ResultIndex int    `json:"resultIndex"`
                                Value       struct {
                                        ID string `json:"id"`
                                } `json:"value"`
                        } `json:"items"`
```

```
                } `json:"mainline"`
                Sidebar struct {
                        Items []struct {
                                AnswerType string `json:"answerType"`
                                Value      struct {
                                        ID string `json:"id"`
                                } `json:"value"`
                        } `json:"items"`
                } `json:"sidebar"`
        } `json:"rankingResponse"`
}

// Replace the token string with a valid subscription key.
func main() {
    const endpoint = "https://api.bing.microsoft.com/v7.0/search"
    token := "YOUR-ACCESS-KEY"
    searchTerm := "Microsoft Cognitive Services"

    // Declare a new GET request.
    req, err := http.NewRequest("GET", endpoint, nil)
    if err != nil {
        panic(err)
    }

    // Add the payload to the request.
    param := req.URL.Query()
    param.Add("q", searchTerm)
    req.URL.RawQuery = param.Encode()

    // Insert the request header.
    req.Header.Add("Ocp-Apim-Subscription-Key", token)

    // Create a new client.
    client := new(http.Client)

    // Send the request to Bing.
    resp, err := client.Do(req)
    if err != nil {
        panic(err)
    }

    // Close the response.
    defer resp.Body.Close()
    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        panic(err)
    }

    // Create a new answer.
    ans := new(BingAnswer)
    err = json.Unmarshal(body, &ans)
    if err != nil {
         fmt.Println(err)
    }
```

```
    // Iterate over search results and print the result name and URL.
    for _, result := range ans.WebPages.Value {
        fmt.Println(result.Name, "||", result.URL)
    }

}
```

Example JSON response

Responses from the Bing Web Search API are returned as JSON. This sample response has been formatted by using the `BingAnswer` struct and shows the `result.Name` and `result.URL`.

Go

```
Microsoft Cognitive Services || https://www.microsoft.com/cognitive-services
Cognitive Services | Microsoft Azure ||
https://azure.microsoft.com/services/cognitive-services/
What is Microsoft Cognitive Services? | Microsoft Docs ||
https://learn.microsoft.com/azure/cognitive-services/Welcome
Microsoft Cognitive Toolkit || https://www.microsoft.com/en-us/cognitive-
toolkit/
Microsoft Customers || https://customers.microsoft.com/en-us/search?
sq=%22Microsoft%20Cognitive%20Services%22&ff=&p=0&so=story_publish_date%20de
sc
Microsoft Enterprise Services - Microsoft Enterprise ||
https://enterprise.microsoft.com/en-us/services/
Microsoft Cognitive Services ||
https://westus.dev.cognitive.microsoft.com/docs/services/563879b61984550e40c
bbe8d/operations/563879b61984550f30395236
Cognitive Services - msdn.microsoft.com ||
https://msdn.microsoft.com/magazine/mt742868.aspx
```

Next steps

Bing Web Search API single-page app tutorial

Quickstart: Use Java to search the web with the Bing Web Search REST API, an Azure cognitive service

Article • 02/16/2022

In this quickstart, you'll use a Java application to make your first call to the Bing Web Search API. This Java application sends a search request to the API, and shows the JSON response. Although this application is written in Java, the API is a RESTful Web service compatible with most programming languages.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- JDK 7 or 8
- Gson library
- A subscription key

Create a project and import dependencies

Create a new Java project in your favorite IDE or editor and import the following libraries. Gson is required to convert Java Objects into JSON.

Java

```
import java.net.*;
import java.util.*;
import java.io.*;
import javax.net.ssl.HttpsURLConnection;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
```

Declare Gson in the Maven POM file

If you're using Maven, declare Gson in POM.xml. Skip this step if you've installed Gson locally.

XML

```
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.5</version>
</dependency>
```

Declare the BingWebSearch class

Declare the `BingWebSearch` class. It includes most of the code we review in this quickstart, including the `main()` method.

Java

```
public class BingWebSearch {

// The code in the following sections goes here.

}
```

Define variables

The following code sets the `subscriptionKey`, `host`, `path`, and `searchTerm`. Add this code to the `BingWebSearch` class described in the previous section:

1. Replace the `subscriptionKey` value with a valid subscription key from your Azure account.
2. Optionally, customize the search query by replacing the value for `searchTerm`.

Java

```
// Enter a valid subscription key.
static String subscriptionKey = "enter key here";

/*
 * If you encounter unexpected authorization errors, double-check these
values
 * against the endpoint for your Bing Web search instance in your Azure
 * dashboard.
 */
static String host = "https://api.bing.microsoft.com";
static String path = "/v7.0/search";
static String searchTerm = "Microsoft Bing Search Services";
```

Construct a request

The `SearchWeb()` method, which is included in the `BingWebSearch` class, constructs the `url`, receives and parses the response, and extracts Bing-related HTTP headers.

Java

```
public static SearchResults SearchWeb (String searchQuery) throws Exception
{
    // Construct the URL.
    URL url = new URL(host + path + "?q=" +  URLEncoder.encode(searchQuery,
"UTF-8"));

    // Open the connection.
    HttpsURLConnection connection =
(HttpsURLConnection)url.openConnection();
    connection.setRequestProperty("Ocp-Apim-Subscription-Key",
subscriptionKey);

    // Receive the JSON response body.
    InputStream stream = connection.getInputStream();
    String response = new Scanner(stream).useDelimiter("\\A").next();

    // Construct the result object.
    SearchResults results = new SearchResults(new HashMap<String, String>(),
response);

    // Extract Bing-related HTTP headers.
    Map<String, List<String>> headers = connection.getHeaderFields();
    for (String header : headers.keySet()) {
        if (header == null) continue;      // may have null key
        if (header.startsWith("BingAPIs-") || header.startsWith("X-MSEdge-
")){
            results.relevantHeaders.put(header, headers.get(header).get(0));
        }
    }
    stream.close();
    return results;
}
```

Handle the response

Use Gson to parse and reserialize the response.

Java

```
public static String prettify(String json_text) {
    JsonParser parser = new JsonParser();
    JsonObject json = parser.parse(json_text).getAsJsonObject();
```

```
    Gson gson = new GsonBuilder().setPrettyPrinting().create();
    return gson.toJson(json);
}
```

Declare the main method

The `main()` method is required and is the first method invoked when you start the program. In this application, it includes code that validates the `subscriptionKey`, makes a request, and then prints the JSON response.

Java

```
public static void main (String[] args) {
    // Confirm the subscriptionKey is valid.
    if (subscriptionKey.length() != 32) {
        System.out.println("Invalid Bing Search API subscription key!");
        System.out.println("Please paste yours into the source code.");
        System.exit(1);
    }

    // Call the SearchWeb method and print the response.
    try {
        System.out.println("Searching the Web for: " + searchTerm);
        SearchResults result = SearchWeb(searchTerm);
        System.out.println("\nRelevant HTTP Headers:\n");
        for (String header : result.relevantHeaders.keySet())
        System.out.println(header + ": " +
result.relevantHeaders.get(header));
        System.out.println("\nJSON Response:\n");
        System.out.println(prettify(result.jsonResponse));
    }
    catch (Exception e) {
        e.printStackTrace(System.out);
        System.exit(1);
    }
}
```

Create a container class for search results

The `SearchResults` container class is defined outside of the `BingWebSearch` class. It includes relevant headers and JSON data for the response.

Java

```
class SearchResults{
    HashMap<String, String> relevantHeaders;
    String jsonResponse;
```

```
    SearchResults(HashMap<String, String> headers, String json) {
        relevantHeaders = headers;
        jsonResponse = json;
    }
}
```

Put it all together

The last step is to compile your code and run it. Use the following commands:

PowerShell

```
javac BingWebSearch.java -classpath ./gson-2.8.5.jar -encoding UTF-8
java -cp ./gson-2.8.5.jar BingWebSearch
```

If you'd like to compare your code with ours, sample code is available on GitHub.

Example JSON response

Responses from the Bing Web Search API are returned as JSON. This sample response has been truncated to show a single result.

JSON

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "Microsoft Cognitive Services"
  },
  "webPages": {
    "webSearchUrl": "https://www.bing.com/search?
q=Microsoft+cognitive+services",
    "totalEstimatedMatches": 22300000,
    "value": [
      {
        "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0",
        "name": "Microsoft Cognitive Services",
        "url": "https://www.microsoft.com/cognitive-services",
        "displayUrl": "https://www.microsoft.com/cognitive-services",
        "snippet": "Knock down barriers between you and your ideas. Enable
natural and contextual interaction with tools that augment users'
experiences via the power of machine-based AI. Plug them in and bring your
ideas to life.",
        "deepLinks": [
          {
            "name": "Face",
            "url": "https://azure.microsoft.com/services/cognitive-
services/face/",
```

```
            "snippet": "Add facial recognition to your applications to
detect, identify, and verify faces using a Face service from Microsoft
Azure. ... Cognitive Services; Face service;"
          },
          {
            "name": "Text Analytics",
            "url": "https://azure.microsoft.com/services/cognitive-
services/text-analytics/",
            "snippet": "Cognitive Services; Text Analytics API; Text
Analytics API . Detect sentiment, ... you agree that Microsoft may store it
and use it to improve Microsoft services, ..."
          },
          {
            "name": "Computer Vision API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/computer-vision/",
            "snippet": "Extract the data you need from images using optical
character recognition and image analytics with Computer Vision APIs from
Microsoft Azure."
          },
          {
            "name": "Emotion",
            "url": "https://www.microsoft.com/cognitive-services/en-
us/emotion-api",
            "snippet": "Cognitive Services Emotion API - microsoft.com"
          },
          {
            "name": "Bing Speech API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/speech/",
            "snippet": "Add speech recognition to your applications,
including text to speech, with a speech API from Microsoft Azure. ...
Cognitive Services; Bing Speech API;"
          },
          {
            "name": "Get Started for Free",
            "url": "https://azure.microsoft.com/services/cognitive-
services/",
            "snippet": "Add vision, speech, language, and knowledge
capabilities to your applications using intelligence APIs and SDKs from
Cognitive Services."
          }
        ]
      }
    ]
  },
  "relatedSearches": {
    "id": "https://api.cognitive.microsoft.com/api/v7/#RelatedSearches",
    "value": [
      {
        "text": "microsoft bot framework",
        "displayText": "microsoft bot framework",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+bot+framework"
      },
```

```
      {
        "text": "microsoft cognitive services youtube",
        "displayText": "microsoft cognitive services youtube",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+youtube"
      },
      {
        "text": "microsoft cognitive services search api",
        "displayText": "microsoft cognitive services search api",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+search+api"
      },
      {
        "text": "microsoft cognitive services news",
        "displayText": "microsoft cognitive services news",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+news"
      },
      {
        "text": "ms cognitive service",
        "displayText": "ms cognitive service",
        "webSearchUrl": "https://www.bing.com/search?q=ms+cognitive+service"
      },
      {
        "text": "microsoft cognitive services text analytics",
        "displayText": "microsoft cognitive services text analytics",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+text+analytics"
      },
      {
        "text": "microsoft cognitive services toolkit",
        "displayText": "microsoft cognitive services toolkit",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+toolkit"
      },
      {
        "text": "microsoft cognitive services api",
        "displayText": "microsoft cognitive services api",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+api"
      }
    ]
  },
  "rankingResponse": {
    "mainline": {
      "items": [
        {
          "answerType": "WebPages",
          "resultIndex": 0,
          "value": {
            "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0"
          }
        }
      ]
    },
```

```
    "sidebar": {
      "items": [
        {
          "answerType": "RelatedSearches",
          "value": {
            "id":
"https://api.cognitive.microsoft.com/api/v7/#RelatedSearches"
          }
        }
      ]
    }
  }
}
```


Next steps

Bing Web Search API single-page app tutorial

Quickstart: Search the web using the Bing Web Search REST API and Node.js

Article • 02/16/2022

Use this quickstart to make your first call to the Bing Web Search API. This Node.js application sends a search request to the API, and shows the JSON response. Although this application is written in JavaScript, the API is a RESTful Web service compatible with most programming languages.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- Node.js 6 or later
- A subscription key

Create a project and declare required modules

Create a new Node.js project in your favorite IDE or editor. Then, copy the following code snippet to your project in a file named search.js:

JavaScript

```
// Use this simple app to query the Bing Web Search API and get a JSON
response.
// Usage: node search.js "your query".
const https = require('https')
```

Set the subscription key

This code snippet uses the `AZURE_SUBSCRIPTION_KEY` environment variable to store your subscription key. Using an environment variable in this way is a good practice to prevent the accidental exposure of your keys when deploying code.

If you're unfamiliar with the use of environment variables, or you want to run this app as fast as possible, replace `process.env['AZURE_SUBSCRIPTION_KEY']` with your subscription key set as a string.

JavaScript

```
const SUBSCRIPTION_KEY = process.env['AZURE_SUBSCRIPTION_KEY']
if (!SUBSCRIPTION_KEY) {
  throw new Error('AZURE_SUBSCRIPTION_KEY is not set.')
}
```

Create a function to make the request

This function makes a secure GET request and saves the search query as a query parameter in the path.

1. Use `encodeURIComponent` to escape invalid characters. The subscription key is passed in a header.
2. The callback receives a response that subscribes to the `data` event to aggregate the JSON body, the `error` event to log any issues, and the `end` event to know when the message should be considered complete.
3. When the app is complete, it prints the relevant headers and message body. You can adjust the colors and set the depth to suit your preference. A depth of `1` gives a nice summary of the response.

JavaScript

```
function bingWebSearch(query) {
  https.get({
    hostname: 'api.bing.microsoft.com',
    path:     '/v7.0/search?q=' + encodeURIComponent(query),
    headers:  { 'Ocp-Apim-Subscription-Key': SUBSCRIPTION_KEY },
  }, res => {
    let body = ''
    res.on('data', part => body += part)
    res.on('end', () => {
      for (var header in res.headers) {
        if (header.startsWith("bingapis-") || header.startsWith("x-msedge-
")) {
          console.log(header + ": " + res.headers[header])
        }
      }
      console.log('\nJSON Response:\n')
      console.dir(JSON.parse(body), { colors: false, depth: null })
    })
    res.on('error', e => {
      console.log('Error: ' + e.message)
      throw e
    })
  })
}
```

Get the query

Let's look at the program's arguments to find the query. The first argument is the path to the node, the second is our filename, and the third is your query. If the query is absent, a default query of "Microsoft Cognitive Services" is used.

JavaScript

```
const query = process.argv[2] || 'Microsoft Bing Search Services'
```

Make a request and print the response

Now that everything is defined, let's call our function.

JavaScript

```
bingWebSearch(query)
```

Put it all together

The last step is to run your code with the command: `node search.js "<your query>"`.

If you'd like to compare your code with ours, here's the complete program:

JavaScript

```
const https = require('https')
const SUBSCRIPTION_KEY = process.env['AZURE_SUBSCRIPTION_KEY']
if (!SUBSCRIPTION_KEY) {
  throw new Error('Missing the AZURE_SUBSCRIPTION_KEY environment variable')
}
function bingWebSearch(query) {
  https.get({
    hostname: 'api.bing.microsoft.com',
    path:     '/v7.0/search?q=' + encodeURIComponent(query),
    headers:  { 'Ocp-Apim-Subscription-Key': SUBSCRIPTION_KEY },
  }, res => {
    let body = ''
    res.on('data', part => body += part)
    res.on('end', () => {
      for (var header in res.headers) {
        if (header.startsWith("bingapis-") || header.startsWith("x-msedge-
")) {
          console.log(header + ": " + res.headers[header])
        }
      }
```

```
      console.log('\nJSON Response:\n')
      console.dir(JSON.parse(body), { colors: false, depth: null })
    })
    res.on('error', e => {
      console.log('Error: ' + e.message)
      throw e
    })
  })
}
const query = process.argv[2] || 'Microsoft Bing Search Services'
bingWebSearch(query)
```

Example JSON response

Responses from the Bing Web Search API are returned as JSON. This sample response has been truncated to show a single result.

JSON


```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "Microsoft Cognitive Services"
  },
  "webPages": {
    "webSearchUrl": "https://www.bing.com/search?
q=Microsoft+cognitive+services",
    "totalEstimatedMatches": 22300000,
    "value": [
      {
        "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0",
        "name": "Microsoft Cognitive Services",
        "url": "https://www.microsoft.com/cognitive-services",
        "displayUrl": "https://www.microsoft.com/cognitive-services",
        "snippet": "Knock down barriers between you and your ideas. Enable
natural and contextual interaction with tools that augment users'
experiences via the power of machine-based AI. Plug them in and bring your
ideas to life.",
        "deepLinks": [
          {
            "name": "Face",
            "url": "https://azure.microsoft.com/services/cognitive-
services/face/",
            "snippet": "Add facial recognition to your applications to
detect, identify, and verify faces using a Face service from Microsoft
Azure. ... Cognitive Services; Face service;"
          },
          {
            "name": "Text Analytics",
            "url": "https://azure.microsoft.com/services/cognitive-
services/text-analytics/",
```

```
            "snippet": "Cognitive Services; Text Analytics API; Text
Analytics API . Detect sentiment, ... you agree that Microsoft may store it
and use it to improve Microsoft services, ..."
          },
          {
            "name": "Computer Vision API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/computer-vision/",
            "snippet": "Extract the data you need from images using optical
character recognition and image analytics with Computer Vision APIs from
Microsoft Azure."
          },
          {
            "name": "Emotion",
            "url": "https://www.microsoft.com/cognitive-services/emotion-
api",
            "snippet": "Cognitive Services Emotion API - microsoft.com"
          },
          {
            "name": "Bing Speech API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/speech/",
            "snippet": "Add speech recognition to your applications,
including text to speech, with a speech API from Microsoft Azure. ...
Cognitive Services; Bing Speech API;"
          },
          {
            "name": "Get Started for Free",
            "url": "https://azure.microsoft.com/services/cognitive-
services/",
            "snippet": "Add vision, speech, language, and knowledge
capabilities to your applications using intelligence APIs and SDKs from
Cognitive Services."
          }
        ]
      }
    ]
  },
  "relatedSearches": {
    "id": "https://api.cognitive.microsoft.com/api/v7/#RelatedSearches",
    "value": [
      {
        "text": "microsoft bot framework",
        "displayText": "microsoft bot framework",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+bot+framework"
      },
      {
        "text": "microsoft cognitive services youtube",
        "displayText": "microsoft cognitive services youtube",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+youtube"
      },
      {
        "text": "microsoft cognitive services search api",
```

```
        "displayText": "microsoft cognitive services search api",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+search+api"
      },
      {
        "text": "microsoft cognitive services news",
        "displayText": "microsoft cognitive services news",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+news"
      },
      {
        "text": "ms cognitive service",
        "displayText": "ms cognitive service",
        "webSearchUrl": "https://www.bing.com/search?q=ms+cognitive+service"
      },
      {
        "text": "microsoft cognitive services text analytics",
        "displayText": "microsoft cognitive services text analytics",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+text+analytics"
      },
      {
        "text": "microsoft cognitive services toolkit",
        "displayText": "microsoft cognitive services toolkit",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+toolkit"
      },
      {
        "text": "microsoft cognitive services api",
        "displayText": "microsoft cognitive services api",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+api"
      }
    ]
  },
  "rankingResponse": {
    "mainline": {
      "items": [
        {
          "answerType": "WebPages",
          "resultIndex": 0,
          "value": {
            "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0"
          }
        }
      ]
    },
    "sidebar": {
      "items": [
        {
          "answerType": "RelatedSearches",
          "value": {
            "id":
"https://api.cognitive.microsoft.com/api/v7/#RelatedSearches"
          }
```

```
        }
      ]
    }
  }
}
```

Next steps

Bing Web Search API single-page app tutorial

Quickstart: Use PHP to call the Bing Web Search API

Article • 02/16/2022

Use this quickstart to make your first call to the Bing Web Search API. This PHP application sends a search request to the API, and shows the response. Although this application is written in PHP, the API is a RESTful Web service compatible with most programming languages.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- PHP 5.6.x or later
- A subscription key

Enable secure HTTP support

Before we get started, locate php.ini and uncomment this line:

```
PHP

; extension=php_openssl.dll
```

Create a project and define variables

1. Create a new PHP project in your favorite IDE or editor. Add opening and closing tags: `<?php` and `?>`.
2. Confirm that the `$endpoint` value is correct and replace the `$accesskey` value with a valid subscription key from your Azure account.
3. Optionally, customize the search query by replacing the value for `$term`.

```
PHP

$accessKey = 'enter key here';
$endpoint = 'https://api.bing.microsoft.com/v7.0/search';
$term = 'Microsoft Bing Search Services';
```

Construct a request

This code declares a function called `BingWebSearch` that's used to construct requests to the Bing Web Search API. It takes three arguments: `$url`, `$key`, and `$query`.

PHP

```
function BingWebSearch ($url, $key, $query) {
    /* Prepare the HTTP request.
     * NOTE: Use the key 'http' even if you are making an HTTPS request.
     * See: http://php.net/manual/en/function.stream-context-create.php.
     */
    $headers = "Ocp-Apim-Subscription-Key: $key\r\n";
    $options = array ('http' => array (
                          'header' => $headers,
                           'method' => 'GET'));

    // Perform the request and get a JSON response.
    $context = stream_context_create($options);
    $result = file_get_contents($url . "?q=" . urlencode($query), false,
$context);

    // Extract Bing HTTP headers.
    $headers = array();
    foreach ($http_response_header as $k => $v) {
        $h = explode(":", $v, 2);
        if (isset($h[1]))
            if (preg_match("/^BingAPIs-/", $h[0]) || preg_match("/^X-
MSEdge-/", $h[0]))
                $headers[trim($h[0])] = trim($h[1]);
    }

    return array($headers, $result);
}
```

Make a request and print the response

This code validates the subscription key, makes a request, and prints the response.

PHP

```
// Validates the subscription key.
if (strlen($accessKey) == 32) {

    print "Searching the Web for: " . $term . "\n";
    // Makes the request.
    list($headers, $json) = BingWebSearch($endpoint, $accessKey, $term);

    print "\nRelevant Headers:\n\n";
```

```
    foreach ($headers as $k => $v) {
        print $k . ": " . $v . "\n";
    }
    // Prints JSON encoded response.
    print "\nJSON Response:\n\n";
    echo json_encode(json_decode($json), JSON_PRETTY_PRINT);

} else {

    print("Invalid Bing Search API subscription key!\n");
    print("Please paste yours into the source code.\n");

}
```

Put it all together

The last step is to validate your code and run it. If you'd like to compare your code with ours, here's the complete program:

PHP

```
<?php
$accessKey = 'enter key here';
$endpoint = 'https://api.bing.microsoft.com/v7.0/search';
$term = 'Microsoft Bing Search Services';

function BingWebSearch ($url, $key, $query) {
    $headers = "Ocp-Apim-Subscription-Key: $key\r\n";
    $options = array ('http' => array (
                          'header' => $headers,
                           'method' => 'GET'));
    $context = stream_context_create($options);
    $result = file_get_contents($url . "?q=" . urlencode($query), false,
$context);
    $headers = array();
    foreach ($http_response_header as $k => $v) {
        $h = explode(":", $v, 2);
        if (isset($h[1]))
            if (preg_match("/^BingAPIs-/", $h[0]) || preg_match("/^X-
MSEdge-/", $h[0]))
                $headers[trim($h[0])] = trim($h[1]);
    }
    return array($headers, $result);
}

if (strlen($accessKey) == 32) {
    print "Searching the Web for: " . $term . "\n";
    list($headers, $json) = BingWebSearch($endpoint, $accessKey, $term);
    print "\nRelevant Headers:\n\n";
    foreach ($headers as $k => $v) {
        print $k . ": " . $v . "\n";
```

```
    }
    print "\nJSON Response:\n\n";
    echo json_encode(json_decode($json), JSON_PRETTY_PRINT);

} else {
    print("Invalid Bing Search API subscription key!\n");
    print("Please paste yours into the source code.\n");
}
?>
```

Example JSON response

Responses from the Bing Web Search API are returned as JSON. This sample response has been truncated to show a single result.

JSON


```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "Microsoft Cognitive Services"
  },
  "webPages": {
    "webSearchUrl": "https://www.bing.com/search?
q=Microsoft+cognitive+services",
    "totalEstimatedMatches": 22300000,
    "value": [
      {
        "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0",
        "name": "Microsoft Cognitive Services",
        "url": "https://www.microsoft.com/cognitive-services",
        "displayUrl": "https://www.microsoft.com/cognitive-services",
        "snippet": "Knock down barriers between you and your ideas. Enable
natural and contextual interaction with tools that augment users'
experiences via the power of machine-based AI. Plug them in and bring your
ideas to life.",
        "deepLinks": [
          {
            "name": "Face",
            "url": "https://azure.microsoft.com/services/cognitive-
services/face/",
            "snippet": "Add facial recognition to your applications to
detect, identify, and verify faces using the Face service from Microsoft
Azure. ... Cognitive Services; Face service;"
          },
          {
            "name": "Text Analytics",
            "url": "https://azure.microsoft.com/services/cognitive-
services/text-analytics/",
            "snippet": "Cognitive Services; Text Analytics API; Text
Analytics API . Detect sentiment, ... you agree that Microsoft may store it
```

```
and use it to improve Microsoft services, ..."
          },
          {
            "name": "Computer Vision API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/computer-vision/",
            "snippet": "Extract the data you need from images using optical
character recognition and image analytics with Computer Vision APIs from
Microsoft Azure."
          },
          {
            "name": "Emotion",
            "url": "https://www.microsoft.com/cognitive-services/en-
us/emotion-api",
            "snippet": "Cognitive Services Emotion API - microsoft.com"
          },
          {
            "name": "Bing Speech API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/speech/",
            "snippet": "Add speech recognition to your applications,
including text to speech, with a speech API from Microsoft Azure. ...
Cognitive Services; Bing Speech API;"
          },
          {
            "name": "Get Started for Free",
            "url": "https://azure.microsoft.com/services/cognitive-
services/",
            "snippet": "Add vision, speech, language, and knowledge
capabilities to your applications using intelligence APIs and SDKs from
Cognitive Services."
          }
        ]
      }
    ]
  },
  "relatedSearches": {
    "id": "https://api.cognitive.microsoft.com/api/v7/#RelatedSearches",
    "value": [
      {
        "text": "microsoft bot framework",
        "displayText": "microsoft bot framework",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+bot+framework"
      },
      {
        "text": "microsoft cognitive services youtube",
        "displayText": "microsoft cognitive services youtube",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+youtube"
      },
      {
        "text": "microsoft cognitive services search api",
        "displayText": "microsoft cognitive services search api",
        "webSearchUrl": "https://www.bing.com/search?
```

```
q=microsoft+cognitive+services+search+api"
      },
      {
        "text": "microsoft cognitive services news",
        "displayText": "microsoft cognitive services news",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+news"
      },
      {
        "text": "ms cognitive service",
        "displayText": "ms cognitive service",
        "webSearchUrl": "https://www.bing.com/search?q=ms+cognitive+service"
      },
      {
        "text": "microsoft cognitive services text analytics",
        "displayText": "microsoft cognitive services text analytics",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+text+analytics"
      },
      {
        "text": "microsoft cognitive services toolkit",
        "displayText": "microsoft cognitive services toolkit",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+toolkit"
      },
      {
        "text": "microsoft cognitive services api",
        "displayText": "microsoft cognitive services api",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+api"
      }
    ]
  },
  "rankingResponse": {
    "mainline": {
      "items": [
        {
          "answerType": "WebPages",
          "resultIndex": 0,
          "value": {
            "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0"
          }
        }
      ]
    },
    "sidebar": {
      "items": [
        {
          "answerType": "RelatedSearches",
          "value": {
            "id":
"https://api.cognitive.microsoft.com/api/v7/#RelatedSearches"
          }
        }
      ]
```

```
    }
  }
}
```

Next steps

Bing Web Search API single-page app tutorial

Quickstart: Use Python to call the Bing Web Search API

Article • 10/28/2020

Use this quickstart to make your first call to the Bing Web Search API. This Python application sends a search request to the API, and shows the JSON response. Although this application is written in Python, the API is a RESTful Web service compatible with most programming languages.

This example is run as a Jupyter notebook on MyBinder. To run it, select the launch binder badge:

Prerequisites

- Python 2.x or 3.x

Define variables

1. Replace the `subscription_key` value with a valid subscription key from your Azure account.

   Python

   ```
   subscription_key = "YOUR_ACCESS_KEY"
   assert subscription_key
   ```

2. Declare the Bing Web Search API endpoint.

   Python

   ```
   search_url = "https://api.bing.microsoft.com/v7.0/search"
   ```

3. Optionally, customize the search query by replacing the value for `search_term`.

   Python

   ```
   search_term = "Microsoft Bing Search Services"
   ```

Make a request

This code uses the `requests` library to call the Bing Web Search API and return the results as a JSON object. The API key is passed in the `headers` dictionary, and the search term and query parameters are passed in the `params` dictionary.

For a complete list of options and parameters, see Bing Web Search API v7.

Python

```
import requests

headers = {"Ocp-Apim-Subscription-Key": subscription_key}
params = {"q": search_term, "textDecorations": True, "textFormat": "HTML"}
response = requests.get(search_url, headers=headers, params=params)
response.raise_for_status()
search_results = response.json()
```

Format and display the response

The `search_results` object includes the search results, and such metadata as related queries and pages. This code uses the `IPython.display` library to format and display the response in your browser.

Python

```
from IPython.display import HTML

rows = "\n".join(["""<tr>
                       <td><a href=\"{0}\">{1}</a></td>
                       <td>{2}</td>
                     </tr>""".format(v["url"], v["name"], v["snippet"])
                  for v in search_results["webPages"]["value"]])
HTML("<table>{0}</table>".format(rows))
```

Sample code on GitHub

To run this code locally, see the complete sample available on GitHub.

Next steps

Bing Web Search API single-page app tutorial

Quickstart: Use Ruby to call the Bing Web Search API

Article • 02/16/2022

Use this quickstart to make your first call to the Bing Web Search API. This Ruby application sends a search request to the API, and shows the JSON response. Although this application is written in Ruby, the API is a RESTful Web service compatible with most programming languages.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- Ruby 2.4 or later
- A subscription key

Create a project and declare required modules

Create a new Ruby project in your favorite IDE or editor. Then, require `net/https` for requests, `uri` for URI handling, and `json` to parse the response.

Ruby

```ruby
require 'net/https'
require 'uri'
require 'json'
```

Define variables

A few variables must be set before we can continue:

1. Confirm that the `uri` and `path` values are valid and replace the `accessKey` value with a subscription key from your Azure account.
2. Optionally, customize the search query by replacing the value for `term`.

Ruby

```ruby
accessKey = "YOUR_SUBSCRIPTION_KEY"
uri  = "https://api.bing.microsoft.com"
```

```
path = "/v7.0/search"
term = "Microsoft Bing Search Services"

if accessKey.length != 32 then
    puts "Invalid Bing Search API subscription key!"
    puts "Please paste yours into the source code."
    abort
end
```

Make a request

Use this code to make a request and handle the response:

Ruby

```
# Construct the endpoint uri.
uri = URI(uri + path + "?q=" + URI.escape(term))
puts "Searching the Web for: " + term

# Create the request.
request = Net::HTTP::Get.new(uri)
request['Ocp-Apim-Subscription-Key'] = accessKey

# Get the response.
response = Net::HTTP.start(uri.host, uri.port, :use_ssl => uri.scheme ==
'https') do |http|
    http.request(request)
end
```

Print the response

Validate the headers, format the response data as JSON, and print the results.

Ruby

```
puts "\nRelevant Headers:\n\n"
response.each_header do |key, value|
    # Header names are lower-cased.
    if key.start_with?("bingapis-") or key.start_with?("x-msedge-") then
        puts key + ": " + value
    end
end

puts "\nJSON Response:\n\n"
puts JSON::pretty_generate(JSON(response.body))
```

Put it all together

The last step is to validate your code and run it. If you'd like to compare your code with ours, here's the complete program:

Ruby

```ruby
require 'net/https'
require 'uri'
require 'json'

accessKey = "enter key here"
uri  = "https://api.bing.microsoft.com"
path = "/v7.0/search"
term = "Microsoft Bing Search Services"

if accessKey.length != 32 then
    puts "Invalid Bing Search API subscription key!"
    puts "Please paste yours into the source code."
    abort
end

uri = URI(uri + path + "?q=" + URI.escape(term))
puts "Searching the Web for: " + term

request = Net::HTTP::Get.new(uri)
request['Ocp-Apim-Subscription-Key'] = accessKey

response = Net::HTTP.start(uri.host, uri.port, :use_ssl => uri.scheme ==
'https') do |http|
    http.request(request)
end

puts "\nRelevant Headers:\n\n"
response.each_header do |key, value|
    if key.start_with?("bingapis-") or key.start_with?("x-msedge-") then
        puts key + ": " + value
    end
end

puts "\nJSON Response:\n\n"
puts JSON::pretty_generate(JSON(response.body))
```

Example JSON response

Responses from the Bing Web Search API are returned as JSON. This sample response has been truncated to show a single result.

JSON

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "Microsoft Cognitive Services"
  },
  "webPages": {
    "webSearchUrl": "https://www.bing.com/search?
q=Microsoft+cognitive+services",
    "totalEstimatedMatches": 22300000,
    "value": [
      {
        "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0",
        "name": "Microsoft Cognitive Services",
        "url": "https://www.microsoft.com/cognitive-services",
        "displayUrl": "https://www.microsoft.com/cognitive-services",
        "snippet": "Knock down barriers between you and your ideas. Enable
natural and contextual interaction with tools that augment users'
experiences via the power of machine-based AI. Plug them in and bring your
ideas to life.",
        "deepLinks": [
          {
            "name": "Face",
            "url": "https://azure.microsoft.com/services/cognitive-
services/face/",
            "snippet": "Add facial recognition to your applications to
detect, identify, and verify faces using the Face service from Microsoft
Azure. ... Cognitive Services; Face service;"
          },
          {
            "name": "Text Analytics",
            "url": "https://azure.microsoft.com/services/cognitive-
services/text-analytics/",
            "snippet": "Cognitive Services; Text Analytics API; Text
Analytics API . Detect sentiment, ... you agree that Microsoft may store it
and use it to improve Microsoft services, ..."
          },
          {
            "name": "Computer Vision API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/computer-vision/",
            "snippet": "Extract the data you need from images using optical
character recognition and image analytics with Computer Vision APIs from
Microsoft Azure."
          },
          {
            "name": "Emotion",
            "url": "https://www.microsoft.com/cognitive-services/en-
us/emotion-api",
            "snippet": "Cognitive Services Emotion API - microsoft.com"
          },
          {
            "name": "Bing Speech API",
            "url": "https://azure.microsoft.com/services/cognitive-
services/speech/",
            "snippet": "Add speech recognition to your applications,
```

```
including text to speech, with a speech API from Microsoft Azure. ...
Cognitive Services; Bing Speech API;"
          },
          {
            "name": "Get Started for Free",
            "url": "https://azure.microsoft.com/services/cognitive-
services/",
            "snippet": "Add vision, speech, language, and knowledge
capabilities to your applications using intelligence APIs and SDKs from
Cognitive Services."
          }
        ]
      }
    ]
  },
  "relatedSearches": {
    "id": "https://api.cognitive.microsoft.com/api/v7/#RelatedSearches",
    "value": [
      {
        "text": "microsoft bot framework",
        "displayText": "microsoft bot framework",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+bot+framework"
      },
      {
        "text": "microsoft cognitive services youtube",
        "displayText": "microsoft cognitive services youtube",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+youtube"
      },
      {
        "text": "microsoft cognitive services search api",
        "displayText": "microsoft cognitive services search api",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+search+api"
      },
      {
        "text": "microsoft cognitive services news",
        "displayText": "microsoft cognitive services news",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+news"
      },
      {
        "text": "ms cognitive service",
        "displayText": "ms cognitive service",
        "webSearchUrl": "https://www.bing.com/search?q=ms+cognitive+service"
      },
      {
        "text": "microsoft cognitive services text analytics",
        "displayText": "microsoft cognitive services text analytics",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+text+analytics"
      },
      {
        "text": "microsoft cognitive services toolkit",
```

```
        "displayText": "microsoft cognitive services toolkit",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+toolkit"
      },
      {
        "text": "microsoft cognitive services api",
        "displayText": "microsoft cognitive services api",
        "webSearchUrl": "https://www.bing.com/search?
q=microsoft+cognitive+services+api"
      }
    ]
  },
  "rankingResponse": {
    "mainline": {
      "items": [
        {
          "answerType": "WebPages",
          "resultIndex": 0,
          "value": {
            "id": "https://api.cognitive.microsoft.com/api/v7/#WebPages.0"
          }
        }
      ]
    },
    "sidebar": {
      "items": [
        {
          "answerType": "RelatedSearches",
          "value": {
            "id":
"https://api.cognitive.microsoft.com/api/v7/#RelatedSearches"
          }
        }
      ]
    }
  }
}
```


Next steps

Bing Web Search API single-page app tutorial

Quickstart: Use a Bing Web Search .NET client library

Article • 09/09/2022

The Bing Web Search client library makes it easy to integrate Bing Web Search into your C# application. In this quickstart, you'll learn how to instantiate a client, send a request, and print the response.

Want to see the code right now? Samples for the Bing Search client libraries for .NET are available on GitHub.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- Visual Studio or
- Visual Studio Code 2017
  - C# for Visual Studio Code
  - NuGet Package Manager
- .NET Core SDK

Create a project and install dependencies

> **Tip**
>
> Get the latest code as a Visual Studio solution from **GitHub**.

The first step is to create a new console project. If you need help with setting up a console project, see Hello World -- Your First Program (C# Programming Guide). To use the Bing Web Search SDK in your application, you'll need to install `Microsoft.Azure.CognitiveServices.Search.WebSearch` using the NuGet Package Manager.

The Web Search SDK package also installs:

- Microsoft.Rest.ClientRuntime
- Microsoft.Rest.ClientRuntime.Azure
- Newtonsoft.Json

Declare dependencies

Open your project in Visual Studio or Visual Studio Code and import these dependencies:

C#

```
using System;
using System.Collections.Generic;
using Microsoft.Azure.CognitiveServices.Search.WebSearch;
using Microsoft.Azure.CognitiveServices.Search.WebSearch.Models;
using System.Linq;
```

Create project scaffolding

When you created your new console project, a namespace and class for your application should have been created. Your program should look like this example:

C#

```
namespace WebSearchSDK
{
    class YOUR_PROGRAM
    {

        // The code in the following sections goes here.

    }
}
```

In the following sections, we'll build our sample application within this class.

Construct a request

This code constructs the search query.

C#

```
public static async void WebResults(WebSearchClient client)
{
    try
    {
        var webData = await client.Web.SearchAsync(query: "Yosemite National
Park");
        Console.WriteLine("Searching for \"Yosemite National Park\"");
```

```
        // Code for handling responses is provided in the next section...

    }
    catch (Exception ex)
    {
        Console.WriteLine("Encountered exception. " + ex.Message);
    }
}
```

Handle the response

Next, let's add some code to parse the response and print the results. The `Name` and `Url` for the first web page, image, news article, and video are printed if present in the response object.

C#

```
if (webData?.WebPages?.Value?.Count > 0)
{
    // find the first web page
    var firstWebPagesResult = webData.WebPages.Value.FirstOrDefault();

    if (firstWebPagesResult != null)
    {
        Console.WriteLine("Webpage Results # {0}",
webData.WebPages.Value.Count);
        Console.WriteLine("First web page name: {0} ",
firstWebPagesResult.Name);
        Console.WriteLine("First web page URL: {0} ",
firstWebPagesResult.Url);
    }
    else
    {
        Console.WriteLine("Didn't find any web pages...");
    }
}
else
{
    Console.WriteLine("Didn't find any web pages...");
}

/*
 * Images
 * If the search response contains images, the first result's name
 * and url are printed.
 */
if (webData?.Images?.Value?.Count > 0)
{
    // find the first image result
    var firstImageResult = webData.Images.Value.FirstOrDefault();
```

```
    if (firstImageResult != null)
    {
        Console.WriteLine("Image Results # {0}",
webData.Images.Value.Count);
        Console.WriteLine("First Image result name: {0} ",
firstImageResult.Name);
        Console.WriteLine("First Image result URL: {0} ",
firstImageResult.ContentUrl);
    }
    else
    {
        Console.WriteLine("Didn't find any images...");
    }
}
else
{
    Console.WriteLine("Didn't find any images...");
}

/*
 * News
 * If the search response contains news articles, the first result's name
 * and url are printed.
 */
if (webData?.News?.Value?.Count > 0)
{
    // find the first news result
    var firstNewsResult = webData.News.Value.FirstOrDefault();

    if (firstNewsResult != null)
    {
        Console.WriteLine("\r\nNews Results # {0}",
webData.News.Value.Count);
        Console.WriteLine("First news result name: {0} ",
firstNewsResult.Name);
        Console.WriteLine("First news result URL: {0} ",
firstNewsResult.Url);
    }
    else
    {
        Console.WriteLine("Didn't find any news articles...");
    }
}
else
{
    Console.WriteLine("Didn't find any news articles...");
}

/*
 * Videos
 * If the search response contains videos, the first result's name
 * and url are printed.
 */
if (webData?.Videos?.Value?.Count > 0)
{
```

```
    // find the first video result
    var firstVideoResult = webData.Videos.Value.FirstOrDefault();

    if (firstVideoResult != null)
    {
        Console.WriteLine("\r\nVideo Results # {0}",
webData.Videos.Value.Count);
        Console.WriteLine("First Video result name: {0} ",
firstVideoResult.Name);
        Console.WriteLine("First Video result URL: {0} ",
firstVideoResult.ContentUrl);
    }
    else
    {
        Console.WriteLine("Didn't find any videos...");
    }
}
else
{
    Console.WriteLine("Didn't find any videos...");
}
```

Declare the main method

In this application, the main method includes code that instantiates the client, validates the `subscriptionKey`, and calls `WebResults`. Make sure that you enter a valid subscription key for your Azure account before continuing.

C#

```
static void Main(string[] args)
{
    var client = new WebSearchClient(new
ApiKeyServiceClientCredentials("YOUR_SUBSCRIPTION_KEY"));

    WebResults(client);

    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
}
```

Run the application

Let's run the application!

Console

```
dotnet run
```

Define functions and filter results

Now that you've made your first call to the Bing Web Search API, let's look at a few functions that highlight SDK functionality for refining queries and filtering results. Each function can be added to your C# application created in the previous section.

Limit the number of results returned by Bing

This sample uses the `count` and `offset` parameters to limit the number of results returned for "Best restaurants in Seattle". The `Name` and `Url` for the first result are printed.

1. Add this code to your console project:

```C#
public static async void WebResultsWithCountAndOffset(WebSearchClient
client)
{
    try
    {
        var webData = await client.Web.SearchAsync(query: "Best
restaurants in Seattle", offset: 10, count: 20);
        Console.WriteLine("\r\nSearching for \" Best restaurants in
Seattle \"");

        if (webData?.WebPages?.Value?.Count > 0)
        {
            var firstWebPagesResult =
webData.WebPages.Value.FirstOrDefault();

            if (firstWebPagesResult != null)
            {
                Console.WriteLine("Web Results #{0}",
webData.WebPages.Value.Count);
                Console.WriteLine("First web page name: {0} ",
firstWebPagesResult.Name);
                Console.WriteLine("First web page URL: {0} ",
firstWebPagesResult.Url);
            }
            else
            {
                Console.WriteLine("Couldn't find first web result!");
            }
        }
        else
        {
```

```
            Console.WriteLine("Didn't see any Web data..");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Encountered exception. " + ex.Message);
    }
}
```

2. Add `WebResultsWithCountAndOffset` to `main`:

C#

```
static void Main(string[] args)
{
    var client = new WebSearchClient(new
ApiKeyServiceClientCredentials("YOUR_SUBSCRIPTION_KEY"));

    WebResults(client);
    // Search with count and offset...
    WebResultsWithCountAndOffset(client);

    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
}
```

3. Run the application.

Filter for news

This sample uses the `response_filter` parameter to filter search results. The search results returned are limited to news articles for "Microsoft". The `Name` and `Url` for the first result are printed.

1. Add this code to your console project:

C#

```
public static async void WebSearchWithResponseFilter(WebSearchClient
client)
{
    try
    {
        IList<string> responseFilterstrings = new List<string>() {
"news" };
        var webData = await client.Web.SearchAsync(query: "Microsoft",
responseFilter: responseFilterstrings);
        Console.WriteLine("\r\nSearching for \" Microsoft \" with
response filter \"news\"");
```

```
        if (webData?.News?.Value?.Count > 0)
        {
            var firstNewsResult = webData.News.Value.FirstOrDefault();

            if (firstNewsResult != null)
            {
                Console.WriteLine("News Results #{0}",
webData.News.Value.Count);
                Console.WriteLine("First news result name: {0} ",
firstNewsResult.Name);
                Console.WriteLine("First news result URL: {0} ",
firstNewsResult.Url);
            }
            else
            {
                Console.WriteLine("Couldn't find first News results!");
            }
        }
        else
        {
            Console.WriteLine("Didn't see any News data..");
        }

    }
    catch (Exception ex)
    {
        Console.WriteLine("Encountered exception. " + ex.Message);
    }
}
```

2. Add `WebResultsWithCountAndOffset` to `main`:

C#

```
static void Main(string[] args)
{
    var client = new WebSearchClient(new
ApiKeyServiceClientCredentials("YOUR_SUBSCRIPTION_KEY"));

    WebResults(client);
    // Search with count and offset...
    WebResultsWithCountAndOffset(client);
    // Search with news filter...
    WebSearchWithResponseFilter(client);

    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
}
```

3. Run the application.

Use safe search, answer count, and the promote filter

This sample uses the `answer_count`, `promote`, and `safe_search` parameters to filter search results for "Music Videos". The `Name` and `ContentUrl` for the first result are displayed.

1. Add this code to your console project:

C#

```
public static async void
WebSearchWithAnswerCountPromoteAndSafeSearch(WebSearchClient client)
{
    try
    {
        IList<string> promoteAnswertypeStrings = new List<string>() {
"videos" };
        var webData = await client.Web.SearchAsync(query: "Music
Videos", answerCount: 2, promote: promoteAnswertypeStrings, safeSearch:
SafeSearch.Strict);
        Console.WriteLine("\r\nSearching for \"Music Videos\"");

        if (webData?.Videos?.Value?.Count > 0)
        {
            var firstVideosResult =
webData.Videos.Value.FirstOrDefault();

            if (firstVideosResult != null)
            {
                Console.WriteLine("Video Results # {0}",
webData.Videos.Value.Count);
                Console.WriteLine("First Video result name: {0} ",
firstVideosResult.Name);
                Console.WriteLine("First Video result URL: {0} ",
firstVideosResult.ContentUrl);
            }
            else
            {
                Console.WriteLine("Couldn't find videos results!");
            }
        }
        else
        {
            Console.WriteLine("Didn't see any data..");
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Encountered exception. " + ex.Message);
    }
}
```

2. Add `WebResultsWithCountAndOffset` to `main`:

```
C#

static void Main(string[] args)
{
    var client = new WebSearchClient(new
ApiKeyServiceClientCredentials("YOUR_SUBSCRIPTION_KEY"));

    WebResults(client);
    // Search with count and offset...
    WebResultsWithCountAndOffset(client);
    // Search with news filter...
    WebSearchWithResponseFilter(client);
    // Search with answer count, promote, and safe search
    WebSearchWithAnswerCountPromoteAndSafeSearch(client);

    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
}
```

3. Run the application.

Clean up resources

When you're done with this project, make sure to remove your subscription key from the application's code.

Next steps

Cognitive Services .NET SDK samples

Quickstart: Use a Bing Web Search Java client library

Article • 10/28/2020

The Bing Web Search client library makes it easy to integrate Bing Web Search into your Java application. In this quickstart, you'll learn how to send a request, receive a JSON response, and filter and parse the results.

Want to see the code right now? Samples for the Bing Search client libraries for Java are available on GitHub.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- JDK 7 or 8
- Apache Maven or your favorite build automation tool
- A subscription key

Create a project and set up your POM file

Create a new Java project using Maven or your favorite build automation tool. Assuming that you're using Maven, add the following lines to your Project Object Model (POM) file. Replace all instances of `mainClass` with your application.

XML

```
<build>
    <plugins>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>1.4.0</version>
        <configuration>
          <!--Your comment
            Replace the mainClass with the path to your java application.
            It should begin with com and doesn't require the .java
extension.
            For example: com.bingwebsearch.app.BingWebSearchSample. This
maps to
            The following directory structure:
            src/main/java/com/bingwebsearch/app/BingWebSearchSample.java.
          -->
          <mainClass>com.path.to.your.app.APP_NAME</mainClass>
```

```
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.0</version>
        <configuration>
          <source>1.7</source>
          <target>1.7</target>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-assembly-plugin</artifactId>
        <executions>
          <execution>
            <phase>package</phase>
            <goals>
              <goal>attached</goal>
            </goals>
            <configuration>
              <descriptorRefs>
                <descriptorRef>jar-with-dependencies</descriptorRef>
              </descriptorRefs>
              <archive>
                <manifest>
                  <!--Your comment
                    Replace the mainClass with the path to your java
application.
                    For example:
com.bingwebsearch.app.BingWebSearchSample.java.
                    This maps to the following directory structure:

src/main/java/com/bingwebsearch/app/BingWebSearchSample.java.
                  -->
                  <mainClass>com.path.to.your.app.APP_NAME.java</mainClass>
                </manifest>
              </archive>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>com.microsoft.azure</groupId>
      <artifactId>azure</artifactId>
      <version>1.9.0</version>
    </dependency>
    <dependency>
      <groupId>commons-net</groupId>
      <artifactId>commons-net</artifactId>
      <version>3.3</version>
    </dependency>
    <dependency>
      <groupId>com.microsoft.azure.cognitiveservices</groupId>
```

```
      <artifactId>azure-cognitiveservices-websearch</artifactId>
      <version>1.0.1</version>
    </dependency>
  </dependencies>
```

Declare dependencies

Open your project in your favorite IDE or editor and import these dependencies:

Java

```
import
com.microsoft.azure.cognitiveservices.search.websearch.BingWebSearchAPI;
import
com.microsoft.azure.cognitiveservices.search.websearch.BingWebSearchManager;
import
com.microsoft.azure.cognitiveservices.search.websearch.models.ImageObject;
import
com.microsoft.azure.cognitiveservices.search.websearch.models.NewsArticle;
import
com.microsoft.azure.cognitiveservices.search.websearch.models.SearchResponse
;
import
com.microsoft.azure.cognitiveservices.search.websearch.models.VideoObject;
import
com.microsoft.azure.cognitiveservices.search.websearch.models.WebPage;
```

If you created the project with Maven, the package should already be declared. Otherwise, declare the package now. For example:

Java

```
package com.bingwebsearch.app
```

Declare the BingWebSearchSample class

Declare the `BingWebSearchSample` class. It will include most of our code including the `main` method.

Java

```
public class BingWebSearchSample {

// The code in the following sections goes here.

}
```

Construct a request

The `runSample` method, which lives in the `BingWebSearchSample` class, constructs the request. Copy this code into your application:

Java

```
public static boolean runSample(BingWebSearchAPI client) {
    /*
     * This function performs the search.
     *
     * @param client instance of the Bing Web Search API client
     * @return true if sample runs successfully
     */
    try {
        /*
         * Performs a search based on the .withQuery and prints the name and
         * url for the first web pages, image, news, and video result
         * included in the response.
         */
        System.out.println("Searched Web for \"Xbox\"");
        // Construct the request.
        SearchResponse webData = client.bingWebs().search()
            .withQuery("Xbox")
            .withMarket("en-us")
            .withCount(10)
            .execute();

// Code continues in the next section...
```

Handle the response

Next, let's add some code to parse the response and print the results. The `name` and `url` for the first web page, image, news article, and video are printed when included in the response object.

Java

```
/*
* WebPages
* If the search response has web pages, the first result's name
* and url are printed.
*/
if (webData != null && webData.webPages() != null &&
webData.webPages().value() != null &&
        webData.webPages().value().size() > 0) {
```

```
    // find the first web page
    WebPage firstWebPagesResult = webData.webPages().value().get(0);

    if (firstWebPagesResult != null) {
        System.out.println(String.format("Webpage Results#%d",
webData.webPages().value().size()));
        System.out.println(String.format("First web page name: %s ",
firstWebPagesResult.name()));
        System.out.println(String.format("First web page URL: %s ",
firstWebPagesResult.url()));
    } else {
        System.out.println("Couldn't find the first web result!");
    }
} else {
    System.out.println("Didn't find any web pages...");
}
/*
 * Images
 * If the search response has images, the first result's name
 * and url are printed.
 */
if (webData != null && webData.images() != null && webData.images().value()
!= null &&
        webData.images().value().size() > 0) {
    // find the first image result
    ImageObject firstImageResult = webData.images().value().get(0);

    if (firstImageResult != null) {
        System.out.println(String.format("Image Results#%d",
webData.images().value().size()));
        System.out.println(String.format("First Image result name: %s ",
firstImageResult.name()));
        System.out.println(String.format("First Image result URL: %s ",
firstImageResult.contentUrl()));
    } else {
        System.out.println("Couldn't find the first image result!");
    }
} else {
    System.out.println("Didn't find any images...");
}
/*
 * News
 * If the search response has news articles, the first result's name
 * and url are printed.
 */
if (webData != null && webData.news() != null && webData.news().value() !=
null &&
        webData.news().value().size() > 0) {
    // find the first news result
    NewsArticle firstNewsResult = webData.news().value().get(0);
    if (firstNewsResult != null) {
        System.out.println(String.format("News Results#%d",
webData.news().value().size()));
        System.out.println(String.format("First news result name: %s ",
firstNewsResult.name()));
```

```
        System.out.println(String.format("First news result URL: %s ",
firstNewsResult.url()));
    } else {
        System.out.println("Couldn't find the first news result!");
    }
} else {
    System.out.println("Didn't find any news articles...");
}

/*
 * Videos
 * If the search response has videos, the first result's name
 * and url are printed.
 */
if (webData != null && webData.videos() != null && webData.videos().value()
!= null &&
        webData.videos().value().size() > 0) {
    // find the first video result
    VideoObject firstVideoResult = webData.videos().value().get(0);

    if (firstVideoResult != null) {
        System.out.println(String.format("Video Results#%s",
webData.videos().value().size()));
        System.out.println(String.format("First Video result name: %s ",
firstVideoResult.name()));
        System.out.println(String.format("First Video result URL: %s ",
firstVideoResult.contentUrl()));
    } else {
        System.out.println("Couldn't find the first video result!");
    }
} else {
    System.out.println("Didn't find any videos...");
}
```

Declare the main method

In this application, the main method includes code that instantiates the client, validates the `subscriptionKey`, and calls `runSample`. Make sure that you enter a valid subscription key for your Azure account before continuing.

Java

```
public static void main(String[] args) {
    try {
        // Enter a valid subscription key for your account.
        final String subscriptionKey = "YOUR_SUBSCRIPTION_KEY";
        // Instantiate the client.
        BingWebSearchAPI client =
BingWebSearchManager.authenticate(subscriptionKey);
        // Make a call to the Bing Web Search API.
        runSample(client);
```

```
    } catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
    }
}
```

Run the program

The final step is to run your program!

Console

```
mvn compile exec:java
```

Clean up resources

When you're done with this project, make sure to remove your subscription key from the program's code.

Next steps

Cognitive Services Java SDK samples

Quickstart: Use a Bing Web Search JavaScript client library

Article • 02/16/2022

The Bing Web Search client library makes it easy to integrate Bing Web Search into your Node.js application. In this quickstart, you'll learn how to instantiate a client, send a request, and print the response.

Want to see the code right now? Samples for the Bing Search client libraries for JavaScript are available on GitHub.

Prerequisites

Here are a few things that you'll need before running this quickstart:

- Node.js 6 or later
- A subscription key

Set up your development environment

Let's start by setting up the development environment for our Node.js project.

1. Create a new directory for your project:

   Console

   ```
   mkdir YOUR_PROJECT
   ```

2. Create a new package file:

   Console

   ```
   cd YOUR_PROJECT
   npm init
   ```

3. Now, let's install some Azure modules and add them to the `package.json`:

   Console

   ```
   npm install --save azure-cognitiveservices-websearch
   npm install --save ms-rest-azure
   ```

Create a project and declare required modules

In the same directory as your `package.json`, create a new Node.js project using your favorite IDE or editor. For example: `sample.js`.

Next, copy this code into your project. It loads the modules installed in the previous section.

JavaScript

```
const CognitiveServicesCredentials = require('ms-rest-
azure').CognitiveServicesCredentials;
const WebSearchAPIClient = require('azure-cognitiveservices-websearch');
```

Instantiate the client

This code instantiates a client and using the `azure-cognitiveservices-websearch` module. Make sure that you enter a valid subscription key for your Azure account before continuing.

JavaScript

```
let credentials = new CognitiveServicesCredentials('YOUR-ACCESS-KEY');
let webSearchApiClient = new WebSearchAPIClient(credentials);
```

Make a request and print the results

Use the client to send a search query to Bing Web Search. If the response includes results for any of the items in the `properties` array, the `result.value` is printed to the console.

JavaScript

```
webSearchApiClient.web.search('seahawks').then((result) => {
    let properties = ["images", "webPages", "news", "videos"];
    for (let i = 0; i < properties.length; i++) {
        if (result[properties[i]]) {
            console.log(result[properties[i]].value);
        } else {
            console.log(`No ${properties[i]} data`);
        }
    }
```

```
}).catch((err) => {
    throw err;
})
```

Run the program

The final step is to run your program!

Clean up resources

When you're done with this project, make sure to remove your subscription key from the program's code.

Next steps

Cognitive Services Node.js SDK samples

Quickstart: Use a Bing Web Search Python client library

Article • 09/27/2022

The Bing Web Search client library makes it easy to integrate Bing Web Search into your Python application. In this quickstart, you'll learn how to send a request, receive a JSON response, and filter and parse the results.

Want to see the code right now? Samples for the Bing Search client libraries for Python are available on GitHub.

Prerequisites

The Bing Web Search SDK is compatible with Python 2.7, 3.3, 3.4, 3.5, and 3.6. We recommend using a virtual environment for this quickstart.

- Python 2.7, 3.3, 3.4, 3.5 or 3.6
- virtualenv for Python 2.7
- venv for Python 3.x

Create and configure your virtual environment

The instructions to set up and configure a virtual environment are different for Python 2.x and Python 3.x. Follow the steps below to create and initialize your virtual environment.

Python 2.x

Create a virtual environment with `virtualenv` for Python 2.7:

Console

```
virtualenv mytestenv
```

Activate your environment:

Console

```
cd mytestenv
source bin/activate
```

Install Bing Web Search SDK dependencies:

```
Console

python -m pip install azure-cognitiveservices-search-websearch
```

Python 3.x

Create a virtual environment with `venv` for Python 3.x:

```
Console

python -m venv mytestenv
```

Activate your environment:

```
Console

mytestenv\Scripts\activate.bat
```

Install Bing Web Search SDK dependencies:

```
Console

cd mytestenv
python -m pip install azure-cognitiveservices-search-websearch
```

Create a client and print your first results

Now that you've set up your virtual environment and installed dependencies, let's create a client. The client will handle requests to and responses from the Bing Web Search API.

If the response contains web pages, images, news, or videos, the first result for each is printed.

1. Create a new Python project using your favorite IDE or editor.
2. Copy this sample code into your project.

   ```
   Python
   ```

```
# Import required modules.
from azure.cognitiveservices.search.websearch import WebSearchClient
from azure.cognitiveservices.search.websearch.models import SafeSearch
from msrest.authentication import CognitiveServicesCredentials

# Replace with your subscription key.
subscription_key = "YOUR_SUBSCRIPTION_KEY"

# Instantiate the client and replace with your endpoint.
client = WebSearchClient(endpoint="YOUR_ENDPOINT",
credentials=CognitiveServicesCredentials(subscription_key))

# Make a request. Replace Yosemite if you'd like.
web_data = client.web.search(query="Yosemite")
print("\r\nSearched for Query# \" Yosemite \"")

'''
Web pages
If the search response contains web pages, the first result's name and
url
are printed.
'''
if hasattr(web_data.web_pages, 'value'):

    print("\r\nWebpage Results#
{}".format(len(web_data.web_pages.value)))

    first_web_page = web_data.web_pages.value[0]
    print("First web page name: {} ".format(first_web_page.name))
    print("First web page URL: {} ".format(first_web_page.url))

else:
    print("Didn't find any web pages...")

'''
Images
If the search response contains images, the first result's name and url
are printed.
'''
if hasattr(web_data.images, 'value'):

    print("\r\nImage Results#{}".format(len(web_data.images.value)))

    first_image = web_data.images.value[0]
    print("First Image name: {} ".format(first_image.name))
    print("First Image URL: {} ".format(first_image.url))

else:
    print("Didn't find any images...")

'''
News
If the search response contains news, the first result's name and url
are printed.
```

```
'''
if hasattr(web_data.news, 'value'):

    print("\r\nNews Results#{}".format(len(web_data.news.value)))

    first_news = web_data.news.value[0]
    print("First News name: {} ".format(first_news.name))
    print("First News URL: {} ".format(first_news.url))

else:
    print("Didn't find any news...")

'''
If the search response contains videos, the first result's name and url
are printed.
'''
if hasattr(web_data.videos, 'value'):

    print("\r\nVideos Results#{}".format(len(web_data.videos.value)))

    first_video = web_data.videos.value[0]
    print("First Videos name: {} ".format(first_video.name))
    print("First Videos URL: {} ".format(first_video.url))

else:
    print("Didn't find any videos...")
```

3. Replace `SUBSCRIPTION_KEY` with a valid subscription key.

4. Replace `YOUR_ENDPOINT` with your endpoint URL in portal and remove the "bing/v7.0" section from the endpoint.

5. Run the program. For example: `python your_program.py`.

Define functions and filter results

Now that you've made your first call to the Bing Web Search API, let's look at a few functions. The following sections highlight SDK functionality for refining queries and filtering results. Each function can be added to the Python program you created in the previous section.

Limit the number of results returned by Bing

This sample uses the `count` and `offset` parameters to limit the number of results returned using the SDK's search method. The `name` and `url` for the first result are printed.

1. Add this code to your Python project:

Python

```
 # Declare the function.
 def web_results_with_count_and_offset(subscription_key):
     client =
WebSearchAPI(CognitiveServicesCredentials(subscription_key))

     try:
         '''
         Set the query, offset, and count using the SDK's search
method. See:
         https://learn.microsoft.com/python/api/azure-
cognitiveservices-search-
websearch/azure.cognitiveservices.search.websearch.operations.weboperat
ions?view=azure-python.
         '''
         web_data = client.web.search(query="Best restaurants in
Seattle", offset=10, count=20)
         print("\r\nSearching for \"Best restaurants in Seattle\"")

         if web_data.web_pages.value:
             '''
             If web pages are available, print the # of responses, and
the first and second
             web pages returned.
             '''
             print("Webpage Results#
{}".format(len(web_data.web_pages.value)))

             first_web_page = web_data.web_pages.value[0]
             print("First web page name: {}
".format(first_web_page.name))
             print("First web page URL: {}
".format(first_web_page.url))

         else:
             print("Didn't find any web pages...")

     except Exception as err:
         print("Encountered exception. {}".format(err))
```

2. Run the program.

Filter for news and freshness

This sample uses the `response_filter` and `freshness` parameters to filter search results using the SDK's search method. The search results returned are limited to news articles

and pages that Bing has discovered within the last 24 hours. The `name` and `url` for the first result are printed.

1. Add this code to your Python project:

Python

```
# Declare the function.
def web_search_with_response_filter(subscription_key):
    client =
WebSearchAPI(CognitiveServicesCredentials(subscription_key))
    try:
        '''
        Set the query, response_filter, and freshness using the SDK's
search method. See:
        https://learn.microsoft.com/python/api/azure-cognitiveservices-
search-
websearch/azure.cognitiveservices.search.websearch.operations.weboperat
ions?view=azure-python.
        '''
        web_data = client.web.search(query="xbox",
            response_filter=["News"],
            freshness="Day")
        print("\r\nSearching for \"xbox\" with the response filter set
to \"News\" and freshness filter set to \"Day\".")

        '''
        If news articles are available, print the # of responses, and
the first and second
        articles returned.
        '''
        if web_data.news.value:

            print("# of news results:
{}".format(len(web_data.news.value)))

            first_web_page = web_data.news.value[0]
            print("First article name: {}
".format(first_web_page.name))
            print("First article URL: {} ".format(first_web_page.url))

            print("")

            second_web_page = web_data.news.value[1]
            print("\nSecond article name: {}
".format(second_web_page.name))
            print("Second article URL: {}
".format(second_web_page.url))

        else:
            print("Didn't find any news articles...")

    except Exception as err:
```

```
        print("Encountered exception. {}".format(err))

# Call the function.
web_search_with_response_filter(subscription_key)
```

2. Run the program.

Use safe search, answer count, and the promote filter

This sample uses the `answer_count`, `promote`, and `safe_search` parameters to filter search results using the SDK's search method. The `name` and `url` for the first result are displayed.

1. Add this code to your Python project:

Python

```
# Declare the function.
def
web_search_with_answer_count_promote_and_safe_search(subscription_key):

    client =
WebSearchAPI(CognitiveServicesCredentials(subscription_key))

    try:
        '''
        Set the query, answer_count, promote, and safe_search
parameters using the SDK's search method. See:
        https://learn.microsoft.com/python/api/azure-cognitiveservices-
search-
websearch/azure.cognitiveservices.search.websearch.operations.weboperat
ions?view=azure-python.
        '''
        web_data = client.web.search(
            query="Niagara Falls",
            answer_count=2,
            promote=["videos"],
            safe_search=SafeSearch.strict  # or directly "Strict"
        )
        print("\r\nSearching for \"Niagara Falls\"")

        '''
        If results are available, print the # of responses, and the
first result returned.
        '''
        if web_data.web_pages.value:

            print("Webpage Results#
{}".format(len(web_data.web_pages.value)))

            first_web_page = web_data.web_pages.value[0]
```

```
            print("First web page name: {}
".format(first_web_page.name))
            print("First web page URL: {} ".format(first_web_page.url))

        else:
            print("Didn't see any Web data..")

    except Exception as err:
        print("Encountered exception. {}".format(err))
```

2. Run the program.

Clean up resources

When you're done with this project, make sure to remove your subscription key from the program's code and to deactivate your virtual environment.

Next steps

Cognitive Services Python SDK samples

Tutorial: Create a single-page app using the Bing Web Search API

Article • 02/21/2024

This single-page app demonstrates how to retrieve, parse, and display search results from the Bing Web Search API. The tutorial uses boilerplate HTML and CSS, and focuses on the JavaScript code. HTML, CSS, and JS files are available on GitHub with quickstart instructions.

This sample app can:

- ✓ Call the Bing Web Search API with search options
- ✓ Display web, image, news, and video results
- ✓ Paginate results
- ✓ Manage subscription keys
- ✓ Handle errors

Prerequisites

Here are a few things that you'll need to run the app:

- An Azure subscription - Create one for free.
- Once you have your Azure subscription, create a Bing Search resource in the Azure portal to get your key and endpoint. After it deploys, click **Go to resource**.
- Node.js 8 or later

The first step is to clone the repository with the sample app's source code.

Console

```
git clone https://github.com/Azure-Samples/cognitive-services-REST-api-samples.git
```

Then run `npm install`. For this tutorial, Express.js is the only dependency.

Console

```
cd <path-to-repo>/cognitive-services-REST-api-samples/Tutorials/Bing-Web-Search
npm install
```

App components

The sample app we're building is made up of four parts:

- `bing-web-search.js` - Our Express.js app. It handles request/response logic and routing.
- `public/index.html` - The skeleton of our app; it defines how data is presented to the user.
- `public/css/styles.css` - Defines page styles, such as fonts, colors, text size.
- `public/js/scripts.js` - Contains the logic to make requests to the Bing Web Search API, manage subscription keys, handle and parse responses, and display results.

This tutorial focuses on `scripts.js` and the logic required to call the Bing Web Search API and handle the response.

HTML form

The `index.html` includes a form that enables users to search and select search options. The `onsubmit` attribute fires when the form is submitted, calling the `bingWebSearch()` method defined in `scripts.js`. It takes three arguments:

- Search query
- Selected options
- Subscription key

HTML

```
<form name="bing" onsubmit="return bingWebSearch(this.query.value,
    bingSearchOptions(this), getSubscriptionKey())">
```

Query options

The HTML form includes options that map to query parameters in the Bing Web Search API v7 reference. This table provides a breakdown of how users can filter search results using the sample app:

Expand table

Parameter	Description
`query`	A text field to enter a query string.
`where`	A drop-down menu to select the market (location and language).
`what`	Checkboxes to promote specific result types. Promoting images, for example, increases the ranking of images in search results.
`when`	A drop-down menu that allows the user to limit the search results to today, this week, or this month.
`safe`	A checkbox to enable Bing SafeSearch, which filters out adult content.
`count`	Hidden field. The number of search results to return on each request. Change this value to display fewer or more results per page.
`offset`	Hidden field. The offset of the first search result in the request, which is used for paging. It's reset to `0` with each new request.

> **Note**
>
> The Bing Web Search API offers additional query parameters to help refine search results. This sample only uses a few. For a complete list of available parameters, see **Bing Web Search API v7** reference.

The `bingSearchOptions()` function converts these options to match the format required by the Bing Search API.

JavaScript

```
// Build query options from selections in the HTML form.
function bingSearchOptions(form) {

    var options = [];
    // Where option.
    options.push("mkt=" + form.where.value);
    // SafeSearch option.
    options.push("SafeSearch=" + (form.safe.checked ? "strict" :
"moderate"));
    // Freshness option.
    if (form.when.value.length) options.push("freshness=" +
form.when.value);
    var what = [];
    for (var i = 0; i < form.what.length; i++)
        if (form.what[i].checked) what.push(form.what[i].value);
    // Promote option.
    if (what.length) {
        options.push("promote=" + what.join(","));
```

```
            options.push("answerCount=9");
        }
        // Count option.
        options.push("count=" + form.count.value);
        // Offset option.
        options.push("offset=" + form.offset.value);
        // Hardcoded text decoration option.
        options.push("textDecorations=true");
        // Hardcoded text format option.
        options.push("textFormat=HTML");
        return options.join("&");
    }
```

`SafeSearch` can be set to `strict`, `moderate`, or `off`, with `moderate` being the default setting for Bing Web Search. This form uses a checkbox, which has two states: `strict` or `moderate`.

If any of the **Promote** check boxes are selected, the `answerCount` parameter is added to the query. `answerCount` is required when using the `promote` parameter. In this snippet, the value is set to `9` to return all available result types.

> ⓘ **Note**
>
> Promoting a result type doesn't *guarantee* that it will be included in the search results. Rather, promotion increases the ranking of those kinds of results relative to their usual ranking. To limit searches to particular kinds of results, use the `responseFilter` query parameter, or call a more specific endpoint such as Bing Image Search or Bing News Search.

The `textDecoration` and `textFormat` query parameters are hardcoded into the script, and cause the search term to be boldfaced in the search results. These parameters aren't required.

Manage subscription keys

To avoid hardcoding the Bing Search API subscription key, this sample app uses a browser's persistent storage to store the subscription key. If no subscription key is stored, the user is prompted to enter one. If the subscription key is rejected by the API, the user is prompted to re-enter a subscription key.

The `getSubscriptionKey()` function uses the `storeValue` and `retrieveValue` functions to store and retrieve a user's subscription key. These functions use the `localStorage` object, if supported, or cookies.

JavaScript

```
// Cookie names for stored data.
API_KEY_COOKIE   = "bing-search-api-key";
CLIENT_ID_COOKIE = "bing-search-client-id";

BING_ENDPOINT = "https://api.bing.microsoft.com/v7.0/search";

// See source code for storeValue and retrieveValue definitions.

// Get stored subscription key, or prompt if it isn't found.
function getSubscriptionKey() {
    var key = retrieveValue(API_KEY_COOKIE);
    while (key.length !== 32) {
        key = prompt("Enter Bing Search API subscription key:", "").trim();
    }
    // Always set the cookie in order to update the expiration date.
    storeValue(API_KEY_COOKIE, key);
    return key;
}
```

As we saw earlier, when the form is submitted, `onsubmit` fires, calling `bingWebSearch`. This function initializes and sends the request. `getSubscriptionKey` is called on each submission to authenticate the request.

Call Bing Web Search

Given the query, the options string, and the subscription key, the `BingWebSearch` function creates an `XMLHttpRequest` object to call the Bing Web Search endpoint.

JavaScript

```
// Perform a search constructed from the query, options, and subscription
key.
function bingWebSearch(query, options, key) {
    window.scrollTo(0, 0);
    if (!query.trim().length) return false;

    showDiv("noresults", "Working. Please wait.");
    hideDivs("pole", "mainline", "sidebar", "_json", "_http", "paging1",
"paging2", "error");

    var request = new XMLHttpRequest();
    var queryurl = BING_ENDPOINT + "?q=" + encodeURIComponent(query) + "&" +
options;

    // Initialize the request.
    try {
        request.open("GET", queryurl);
```

```
    }
    catch (e) {
        renderErrorMessage("Bad request (invalid URL)\n" + queryurl);
        return false;
    }

    // Add request headers.
    request.setRequestHeader("Ocp-Apim-Subscription-Key", key);
    request.setRequestHeader("Accept", "application/json");
    var clientid = retrieveValue(CLIENT_ID_COOKIE);
    if (clientid) request.setRequestHeader("X-MSEdge-ClientID", clientid);

    // Event handler for successful response.
    request.addEventListener("load", handleBingResponse);

    // Event handler for errors.
    request.addEventListener("error", function() {
        renderErrorMessage("Error completing request");
    });

    // Event handler for an aborted request.
    request.addEventListener("abort", function() {
        renderErrorMessage("Request aborted");
    });

    // Send the request.
    request.send();
    return false;
}
```

Following a successful request, the `load` event handler fires and calls the `handleBingResponse` function. `handleBingResponse` parses the result object, displays the results, and contains error logic for failed requests.

JavaScript

```
function handleBingResponse() {
    hideDivs("noresults");

    var json = this.responseText.trim();
    var jsobj = {};

    // Try to parse results object.
    try {
        if (json.length) jsobj = JSON.parse(json);
    } catch(e) {
        renderErrorMessage("Invalid JSON response");
        return;
    }

    // Show raw JSON and the HTTP request.
    showDiv("json", preFormat(JSON.stringify(jsobj, null, 2)));
```

```
    showDiv("http", preFormat("GET " + this.responseURL + "\n\nStatus: " +
this.status + " " +
        this.statusText + "\n" + this.getAllResponseHeaders()));

    // If the HTTP response is 200 OK, try to render the results.
    if (this.status === 200) {
        var clientid = this.getResponseHeader("X-MSEdge-ClientID");
        if (clientid) retrieveValue(CLIENT_ID_COOKIE, clientid);
        if (json.length) {
            if (jsobj._type === "SearchResponse" && "rankingResponse" in
jsobj) {
                renderSearchResults(jsobj);
            } else {
                renderErrorMessage("No search results in JSON response");
            }
        } else {
            renderErrorMessage("Empty response (are you sending too many
requests too quickly?)");
        }
    }

    // Any other HTTP response is considered an error.
    else {
        // 401 is unauthorized; force a re-prompt for the user's
subscription
        // key on the next request.
        if (this.status === 401) invalidateSubscriptionKey();

        // Some error responses don't have a top-level errors object, if
absent
        // create one.
        var errors = jsobj.errors || [jsobj];
        var errmsg = [];

        // Display the HTTP status code.
        errmsg.push("HTTP Status " + this.status + " " + this.statusText +
"\n");

        // Add all fields from all error responses.
        for (var i = 0; i < errors.length; i++) {
            if (i) errmsg.push("\n");
            for (var k in errors[i]) errmsg.push(k + ": " + errors[i][k]);
        }

        // Display Bing Trace ID if it isn't blocked by CORS.
        var traceid = this.getResponseHeader("BingAPIs-TraceId");
        if (traceid) errmsg.push("\nTrace ID " + traceid);

        // Display the error message.
        renderErrorMessage(errmsg.join("\n"));
    }
}
```

> **Important**
>
> A successful HTTP request *doesn't* mean that the search itself succeeded. If an error occurs in the search operation, the Bing Web Search API returns a non-200 HTTP status code and includes error information in the JSON response. If the request was rate-limited, the API returns an empty response.

Much of the code in both of the preceding functions is dedicated to error handling. Errors may occur at the following stages:

Expand table

Stage	Potential error(s)	Handled by
Building the request object	Invalid URL	`try` / `catch` block
Making the request	Network errors, aborted connections	`error` and `abort` event handlers
Performing the search	Invalid request, invalid JSON, rate limits	Tests in `load` event handler

Errors are handled by calling `renderErrorMessage()`. If the response passes all of the error tests, `renderSearchResults()` is called to display the search results.

Display search results

There are use and display requirements for results returned by the Bing Web Search API. Since a response may include various result types, it isn't enough to iterate through the top-level `WebPages` collection. Instead, the sample app uses `RankingResponse` to order the results to spec.

> **Note**
>
> If you only want a single result type, use the `responseFilter` query parameter, or consider using one of the other Bing Search endpoints, such as Bing Image Search.

Each response has a `RankingResponse` object that may include up to three collections: `pole`, `mainline`, and `sidebar`. `pole`, if present, is the most relevant search result and must be prominently displayed. `mainline` contains most of the search results, and is

displayed immediately after `pole`. `sidebar` includes auxiliary search results. If possible, these results should be displayed in the sidebar. If screen limits make a sidebar impractical, these results should appear after the `mainline` results.

Each `RankingResponse` includes a `RankingItem` array that specifies how results must be ordered. Our sample app uses the `answerType` and `resultIndex` parameters to identify the result.

> ⓘ **Note**
>
> There are additional ways to identify and rank results. For more information, see **Using ranking to display results**.

Let's take a look at the code:

JavaScript

```
// Render the search results from the JSON response.
function renderSearchResults(results) {

    // If spelling was corrected, update the search field.
    if (results.queryContext.alteredQuery)
        document.forms.bing.query.value = results.queryContext.alteredQuery;

    // Add Prev / Next links with result count.
    var pagingLinks = renderPagingLinks(results);
    showDiv("paging1", pagingLinks);
    showDiv("paging2", pagingLinks);

    // Render the results for each section.
    for (section in {pole: 0, mainline: 0, sidebar: 0}) {
        if (results.rankingResponse[section])
            showDiv(section, renderResultsItems(section, results));
    }
}
```

The `renderResultsItems()` function iterates through the items in each `RankingResponse` collection, maps each ranking result to a search result using the `answerType` and `resultIndex` values, and calls the appropriate rendering function to generate the HTML. If `resultIndex` isn't specified for an item, `renderResultsItems()` iterates through all results of that type and calls the rendering function for each item. The resulting HTML is inserted into the appropriate `<div>` element in `index.html`.

JavaScript

```
// Render search results from the RankingResponse object per rank response
and
// use and display requirements.
function renderResultsItems(section, results) {

    var items = results.rankingResponse[section].items;
    var html = [];
    for (var i = 0; i < items.length; i++) {
        var item = items[i];
        // Collection name has lowercase first letter while answerType has
uppercase
        // e.g. `WebPages` RankingResult type is in the `webPages` top-level
collection.
        var type = item.answerType[0].toLowerCase() +
item.answerType.slice(1);
        if (type in results && type in searchItemRenderers) {
            var render = searchItemRenderers[type];
            // This ranking item refers to ONE result of the specified type.
            if ("resultIndex" in item) {
                html.push(render(results[type].value[item.resultIndex],
section));
            // This ranking item refers to ALL results of the specified
type.
            } else {
                var len = results[type].value.length;
                for (var j = 0; j < len; j++) {
                    html.push(render(results[type].value[j], section, j,
len));
                }
            }
        }
    }
    return html.join("\n\n");
}
```

Review renderer functions

In our sample app, the `searchItemRenderers` object includes functions that generate HTML for each type of search result.

JavaScript

```
// Render functions for each result type.
searchItemRenderers = {
    webPages: function(item) { ... },
    news: function(item) { ... },
    images: function(item, section, index, count) { ... },
    videos: function(item, section, index, count) { ... },
```

```
    relatedSearches: function(item, section, index, count) { ... }
}
```

> **Important**
>
> The sample app has renderers for web pages, news, images, videos, and related searches. Your application will need renderers for any type of results it may receive, which could include computations, spelling suggestions, entities, time zones, and definitions.

Some of the rendering functions accept only the `item` parameter. Others accept additional parameters, which can be used to render items differently based on context. A renderer that doesn't use this information doesn't need to accept these parameters.

The context arguments are:

Expand table

Parameter	Description
`section`	The results section (`pole`, `mainline`, or `sidebar`) in which the item appears.
`index` `count`	Available when the `RankingResponse` item specifies that all results in a given collection are to be displayed; `undefined` otherwise. The index of the item within its collection and the total number of items in that collection. You can use this information to number the results, to generate different HTML for the first or last result, and so on.

In the sample app, both the `images` and `relatedSearches` renderers use the context arguments to customize the generated HTML. Let's take a closer look at the `images` renderer:

JavaScript

```
searchItemRenderers = {
    // Render image result with thumbnail.
    images: function(item, section, index, count) {
        var height = 60;
        var width = Math.round(height * item.thumbnail.width /
item.thumbnail.height);
        var html = [];
        if (section === "sidebar") {
            if (index) html.push("<br>");
        } else {
            if (!index) html.push("<p class='images'>");
        }
```

```
        html.push("<a href='" + item.hostPageUrl + "'>");
        var title = escape(item.name) + "\n" +
getHost(item.hostPageDisplayUrl);
        html.push("<img src='"+ item.thumbnailUrl + "&h=" + height + "&w=" +
width +
            "' height=" + height + " width=" + width + " title='" + title +
"' alt='" + title + "'>");
        html.push("</a>");
        return html.join("");
    },
    // Other renderers are omitted from this sample...
}
```

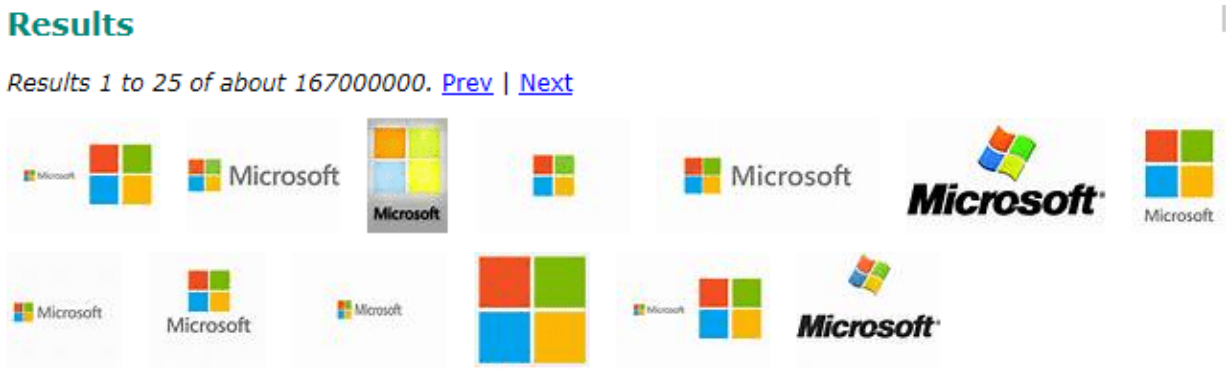
The image renderer:

- Calculates the image thumbnail size (width varies, while height is fixed at 60 pixels).
- Inserts the HTML that precedes the image result based on context.
- Builds the HTML `<a>` tag that links to the page that contains the image.
- Builds the HTML `<img>` tag to display the image thumbnail.

The image renderer uses the `section` and `index` variables to display results differently depending on where they appear. A line break (`<br>` tag) is inserted between image results in the sidebar, so that the sidebar displays a column of images. In other sections, the first image result `(index === 0)` is preceded by a `<p>` tag.

The thumbnail size is used in both the `<img>` tag and the `h` and `w` fields in the thumbnail's URL. The `title` and `alt` attributes (a textual description of the image) are constructed from the image's name and the hostname in the URL.

Here's an example of how images are displayed in the sample app:



Persist the client ID

Responses from the Bing search APIs may include a `X-MSEdge-ClientID` header that should be sent back to the API with each successive request. If more than one of the

Bing Search APIs is used by your app, make sure the same client ID is sent with each request across services.

Providing the `X-MSEdge-ClientID` header allows the Bing APIs to associate a user's searches. First, it allows the Bing search engine to apply past context to searches to find results that better satisfy the request. If a user has previously searched for terms related to sailing, for example, a later search for "knots" might preferentially return information about knots used in sailing. Second, Bing may randomly select users to experience new features before they are made widely available. Providing the same client ID with each request ensures that users who have been chosen to see a feature will always see it. Without the client ID, the user might see a feature appear and disappear, seemingly at random, in their search results.

Browser security policies, such as Cross-Origin Resource Sharing (CORS), may prevent the sample app from accessing the `X-MSEdge-ClientID` header. This limitation occurs when the search response has a different origin from the page that requested it. In a production environment, you should address this policy by hosting a server-side script that does the API call on the same domain as the Web page. Since the script has the same origin as the Web page, the `X-MSEdge-ClientID` header is then available to JavaScript.

> ⓘ **Note**
>
> In a production Web application, you should perform the request server-side anyway. Otherwise, your Bing Search API subscription key must be included in the web page, where it's available to anyone who views source. You are billed for all usage under your API subscription key, even requests made by unauthorized parties, so it is important not to expose your key.

For development purposes, you can make a request through a CORS proxy. The response from this type of proxy has an `Access-Control-Expose-Headers` header that identify headers and makes them available to JavaScript.

It's easy to install a CORS proxy to allow our sample app to access the client ID header. Run this command:

Console

```
npm install -g cors-proxy-server
```

Next, change the Bing Web Search endpoint in `script.js` to:

JavaScript

```
http://localhost:9090/https://api.bing.microsoft.com/v7.0/search
```

Start the CORS proxy with this command:

Console

```
cors-proxy-server
```

Leave the command window open while you use the sample app; closing the window stops the proxy. In the expandable HTTP Headers section below the search results, the `X-MSEdge-ClientID` header should be visible. Verify that it's the same for each request.

Next steps

Bing Web Search API v7 reference

Build a console app search client in C#

Article • 02/16/2022

This tutorial shows how to build a simple .NET Core console app that allows users to query the Bing Web Search API and display ranked results.

This tutorial shows how to:

- Make a simple query to the Bing Web Search API.
- Display query results in ranked order.

Prerequisites

To follow along with the tutorial, you need:

- An Azure subscription - Create one for free.
- Once you have your Azure subscription, create a Bing Search resource in the Azure portal to get your key and endpoint. After it deploys, click **Go to resource**.
- The Visual Studio IDE.

Create a new Console App project

In Visual Studio, create a project with `Ctrl` + `Shift` + `N`.

In the **New Project** dialog, click **Visual C#** > **Windows Classic Desktop** > **Console App (.NET Framework)**.

Name the application **MyConsoleSearchApp**, and then click **OK**.

Add the JSON.net NuGet package to the project

JSON.net allows you to work with the JSON responses returned by the API. Add its NuGet package to your project:

- In **Solution Explorer**, right-click on the project and select **Manage NuGet Packages**.
- On the **Browse** tab, search for `Newtonsoft.Json`. Select the latest version, and then click **Install**.
- Click the **OK** button on the **Review Changes** window.

- Close the Visual Studio tab titled **NuGet: MyConsoleSearchApp**.

Add a reference to System.Web

This tutorial relies on the `System.Web` assembly. Add a reference to this assembly to your project:

- In **Solution Explorer**, right-click on **References** and select **Add Reference**
- Select **Assemblies** > **Framework**, then scroll down and check **System.Web**.
- Select **OK**.

Add some necessary using statements

The code in this tutorial requires three additional using statements. Add these statements below the existing `using` statements at the top of **Program.cs**:

C#

```
using System.Web;
using System.Net.Http;
```

Ask the user for a query

In **Solution Explorer**, open **Program.cs**. Update the `Main()` method:

C#

```
static void Main()
{
    // Get the user's query
    Console.Write("Enter Bing query: ");
    string userQuery = Console.ReadLine();
    Console.WriteLine();

    // Run the query and display the results
    RunQueryAndDisplayResults(userQuery);

    // Prevent the console window from closing immediately
    Console.WriteLine("\nHit ENTER to exit...");
    Console.ReadLine();
}
```

This method:

- Asks the user for a query.
- Calls `RunQueryAndDisplayResults(userQuery)` to execute the query and display the results.
- Waits for user input in order to prevent the console window from immediately closing.

Search for query results using the Bing Web Search API

Next, add a method that queries the API and displays the results:

C#

```
static void RunQueryAndDisplayResults(string userQuery)
{
    try
    {
        // Create a query
        var client = new HttpClient();
        client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", "
<YOUR_SUBSCRIPTION_KEY_GOES_HERE>");
        var queryString = HttpUtility.ParseQueryString(string.Empty);
        queryString["q"] = userQuery;
        var query = "https://api.bing.microsoft.com/v7.0/search?" +
queryString;

        // Run the query
        HttpResponseMessage httpResponseMessage =
client.GetAsync(query).Result;

        // Deserialize the response content
        var responseContentString =
httpResponseMessage.Content.ReadAsStringAsync().Result;
        Newtonsoft.Json.Linq.JObject responseObjects =
Newtonsoft.Json.Linq.JObject.Parse(responseContentString);

        // Handle success and error codes
        if (httpResponseMessage.IsSuccessStatusCode)
        {
            DisplayAllRankedResults(responseObjects);
        }
        else
        {
            Console.WriteLine($"HTTP error status code:
{httpResponseMessage.StatusCode.ToString()}");
        }
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
```

```
    }
}
```

This method:

- Creates an `HttpClient` to query the Web Search API.
- Sets the `Ocp-Apim-Subscription-Key` HTTP header, which Bing uses to authenticate the request.
- Executes the request and uses JSON.net to deserialize the results.
- Calls `DisplayAllRankedResults(responseObjects)` to display all results in ranked order.

Make sure to set the value of `Ocp-Apim-Subscription-Key` to your subscription key.

Display ranked results

Before showing how to display the results in ranked order, take a look at a sample web search response:

JSON


```
{
    "_type" : "SearchResponse",
    "webPages" : {
        "webSearchUrl" : "https:\/\/www.bing.com\/cr?IG=70BE289346...",
        "totalEstimatedMatches" : 982000,
        "value" : [{
            "id" :
"https:\/\/api.cognitive.microsoft.com\/api\/v7\/#WebPages.0",
            "name" : "Contoso Sailing Club - Seattle",
            "url" : "https:\/\/www.bing.com\/cr?
IG=70BE289346ED4594874FE...",
            "displayUrl" : "https:\/\/contososailingsea...",
            "snippet" : "Come sail with Contoso in Seattle...",
            "dateLastCrawled" : "2017-04-07T02:25:00"
        },
        {
            "id" :
"https:\/\/api.cognitive.microsoft.com\/api\/7\/#WebPages.6",
            "name" : "Contoso Sailing Lessons - Official Site",
            "url" : "http:\/\/www.bing.com\/cr?IG=70BE289346ED4594874FE...",
            "displayUrl" : "https:\/\/www.constososailinglessonsseat...",
            "snippet" : "Contoso sailing lessons in Seattle...",
            "dateLastCrawled" : "2017-04-09T14:30:00"
        },

        ...
```

```
        ],
        "someResultsRemoved" : true
    },
    "relatedSearches" : {
        "id" :
"https:\/\/api.cognitive.microsoft.com\/api\/7\/#RelatedSearches",
        "value" : [{
            "text" : "sailing lessons",
            "displayText" : "sailing lessons",
            "webSearchUrl" : "https:\/\/www.bing.com\/cr?IG=70BE289346E..."
        }

        ...

        ]
    },
    "rankingResponse" : {
        "mainline" : {
            "items" : [{
                "answerType" : "WebPages",
                "resultIndex" : 0,
                "value" : {
                    "id" :
"https:\/\/api.cognitive.microsoft.com\/api\/v7\/#WebPages.0"
                }
            },
            {
                "answerType" : "WebPages",
                "resultIndex" : 1,
                "value" : {
                    "id" :
"https:\/\/api.cognitive.microsoft.com\/api\/v7\/#WebPages.1"
                }
            }

            ...

            ]
        },
        "sidebar" : {
            "items" : [{
                "answerType" : "RelatedSearches",
                "value" : {
                    "id" :
"https:\/\/api.cognitive.microsoft.com\/api\/v7\/#RelatedSearches"
                }
            }]
        }
    }
}
```


The rankingResponse describes the appropriate display order for search results. It includes one or more of the following, prioritized groups:

- `pole`: The search results to get the most visible treatment (for example, displayed above the mainline and sidebar).
- `mainline`: The search results to display in the mainline.
- `sidebar`: The search results to display in the sidebar. If there is no sidebar, display the results below the mainline.

The ranking response JSON may include one or more of the groups.

In **Program.cs**, add the following method to display results in properly ranked order:

C#

```
static void DisplayAllRankedResults(Newtonsoft.Json.Linq.JObject
responseObjects)
{
    string[] rankingGroups = new string[] { "pole", "mainline", "sidebar" };

    // Loop through the ranking groups in priority order
    foreach (string rankingName in rankingGroups)
    {
        Newtonsoft.Json.Linq.JToken rankingResponseItems =
responseObjects.SelectToken($"rankingResponse.{rankingName}.items");
        if (rankingResponseItems != null)
        {
            foreach (Newtonsoft.Json.Linq.JObject rankingResponseItem in
rankingResponseItems)
            {
                Newtonsoft.Json.Linq.JToken resultIndex;
                rankingResponseItem.TryGetValue("resultIndex", out
resultIndex);
                var answerType = rankingResponseItem.Value<string>
("answerType");
                switch (answerType)
                {
                    case "WebPages":
                        DisplaySpecificResults(resultIndex,
responseObjects.SelectToken("webPages.value"), "WebPage", "name", "url",
"displayUrl", "snippet");
                        break;
                    case "News":
                        DisplaySpecificResults(resultIndex,
responseObjects.SelectToken("news.value"), "News", "name", "url",
"description");
                        break;
                    case "Images":
                        DisplaySpecificResults(resultIndex,
responseObjects.SelectToken("images.value"), "Image", "thumbnailUrl");
                        break;
                    case "Videos":
                        DisplaySpecificResults(resultIndex,
responseObjects.SelectToken("videos.value"), "Video", "embedHtml");
                        break;
```

```
                    case "RelatedSearches":
                        DisplaySpecificResults(resultIndex,
responseObjects.SelectToken("relatedSearches.value"), "RelatedSearch",
"displayText", "webSearchUrl");
                        break;
                }
            }
        }
    }
}
```

This method:

- Loops over the `rankingResponse` groups that the response contains.
- Displays the items in each group by calling `DisplaySpecificResults(...)`.

In **Program.cs**, add the following two methods:

C#

```
static void DisplaySpecificResults(Newtonsoft.Json.Linq.JToken resultIndex,
Newtonsoft.Json.Linq.JToken items, string title, params string[] fields)
{
    if (resultIndex == null)
    {
        foreach (Newtonsoft.Json.Linq.JToken item in items)
        {
            DisplayItem(item, title, fields);
        }
    }
    else
    {
        DisplayItem(items.ElementAt((int)resultIndex), title, fields);
    }
}

static void DisplayItem(Newtonsoft.Json.Linq.JToken item, string title,
string[] fields)
{
    Console.WriteLine($"{title}: ");
    foreach( string field in fields )
    {
        Console.WriteLine($"- {field}: {item[field]}");
    }
    Console.WriteLine();
}
```

These methods work together to output the search results to the console.

Run the application

Run the application. The output should look similar to the following:

```
Enter Bing query: sailing lessons seattle

WebPage:
- name: Contoso Sailing Club - Seattle
- url: https://www.bing.com/cr?IG=70BE289346ED4594874FE...
- displayUrl: https://contososailingsea....
- snippet: Come sail with Contoso in Seattle...

WebPage:
- name: Contoso Sailing Lessons Seattle - Official Site
- url: http://www.bing.com/cr?IG=70BE289346ED4594874FE...
- displayUrl: https://www.constososailinglessonsseat...
- snippet: Contoso sailing lessons in Seattle...

...
```

Next steps

Read more about using ranking to display results.

Bing Web Search samples

Article • 10/28/2020

Use the Bing Web Search samples to learn how to add search capabilities to your application or service using a number of programming languages. The samples, prerequisites, and build instructions are provided on GitHub.

Samples using native HTTP GET requests

Here's a list of REST samples by language. The list is subject to change. For the current list, see GitHub.

Language	Sample
C#	Bing Web Search
Java	Bing Web Search
JavaScript	Bing Web Search
PHP	Bing Web Search
Python	Bing Web Search
Ruby	Bing Web Search
Go	Bing Web Search

Samples using the Bing client library

Here's a list of SDK samples by language. The list is subject to change. For the current list, see GitHub.

Language	Sample
C#	Bing Web Search
Java	Bing Web Search
Node.js	Bing Web Search
Python	Bing Web Search

Next steps

- For a more in depth web app example, see the Web Search tutorial.

Search the web

Article • 02/16/2022

Use Bing Web Search API to search billions of web documents for content that's relevant to the user's search string.

It's easy. If you have your subscription key, just send an HTTP GET request to the following endpoint:

```
https://api.bing.microsoft.com/v7.0/search
```

Here's a cURL example that shows you how to call the endpoint using your subscription key. Change the *q* query parameter to search for whatever you'd like.

curl

```
curl -H "Ocp-Apim-Subscription-Key: <yourkeygoeshere>"
https://api.bing.microsoft.com/v7.0/search?q=microsoft+devices
```

Request and response headers

Although that's all you need to do to search the web, Bing suggests you include a couple of other headers to provide a better search experience for your user. Those headers include:

- User-Agent — Lets Bing know whether needs a mobile or desktop experience.
- X-MSEdge-ClientID — Provides continuity of experience.
- X-MSEdge-ClientIP — Provides the user's location for location aware queries.
- X-Search-Location — Provides the user's location for location aware queries.

The more information you can provide Bing, the better the search experience will be for your users. To learn more about these headers, see Request headers.

Here's a cURL example that includes these headers.

curl

```
curl -H "Ocp-Apim-Subscription-Key: <yourkeygoeshere>" -H "X-MSEdge-
ClientID: 00B4230B74496E7A13CC2C1475056FF4" -H "X-MSEdge-ClientIP:
11.22.33.44" -H "X-Search-Location: lat:55;long:-111;re:22" -A "Mozilla/5.0
(X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
```

```
Chrome/29.0.1547.65 Safari/537.36"
https://api.bing.microsoft.com/v7.0/search?q=microsoft+devices
```

Bing returns a few headers you should capture.

- BingAPIs-TraceId — The ID that identifies the request in the log file.
- X-MSEdge-ClientID — The ID that you need to pass in subsequent request to provide continuity of experience.
- BingAPIs-Market — The market used by Bing for the request.

To learn more about these headers, see Response headers.

Here's a cURL call that returns the response headers. If you want to remove the response data so you can see only the headers, include the `-o nul` parameter.

curl

```
curl -D - -H "Ocp-Apim-Subscription-Key: <yourkeygoeshere>"
https://api.bing.microsoft.com/v7.0/search?q=microsoft+devices
```

Query parameters

The only query parameter that you must pass is the *q* parameter, which you set to the user's query string. You must URL-encode the user's query string and all query parameter values that you pass.

The API supports a number of query parameters that you can pass in your request. Here's a list of the ones you're most likely to pass.

- *count* and *offset* — Used to page webpage results. Read more
- *mkt* — Used to specify the market where the results come from, which is typically the market where the user is making the request from.
- *safeSearch* — Used to specify the user's safe search preference.
- *textDecorations* and *textFormat* — Used to turn on hit highlighting. Read more

To learn more about these parameters and other parameters that you may specify, see Query parameters.

Here's a cURL example that includes these query parameters.

curl

```
curl -H "Ocp-Apim-Subscription-Key: <yourkeygoeshere>"
https://api.bing.microsoft.com/v7.0/search?q=microsoft+devices&mkt=en-
```

```
us&safeSearch=moderate&textdecorations=true&textformat=raw&count=10&offset=0
```

For information about query parameters that you can use to filter the search results, see Filtering search results.

Next steps

- Learn about the response that Bing returns for the user's query.
- Learn how to get the next page of search results.
- Learn what happens if you don't stay within your queries per second (QPS) limit. Hint: your requests get throttled.
- Learn about the quickstarts and samples that are available to help you get up and running fast.

Filter the answers that Bing returns

Article • 02/16/2022

When you query the Web, Bing returns all relevant content that it finds. This could include webpages, images, news, videos, and more. But what if you're only interested in webpages and news; how can you tell Bing you're not interested in any other answers? You use the responseFilter query parameter.

You can specify the answers to include in the response or those that you want to exclude from the response, or not both. To include only the webpage and news answers, specify *responseFilter* as:

```
&responseFilter=webpages,news
```

Remember that you need to URL-encode all query parameters, so the parameter actually looks like:

```
&responseFilter=webpages%2Cnews
```

Bing includes the answers you request only if it finds relevant content that ranks high enough for the page of results you requested. For example, if you filter the response for images, videos, and news but Bing doesn't find relevant videos and news results that rank high enough for the first page, the response includes only images. But if you page through more results, they may include videos and news content.

To exclude specific answers from the response, prefix a minus sign (-) to the answer's name. For example, "-images."

In theory, you could use *responseFilter* to filter for a single answer like only images or news but you're strongly discouraged from doing so. Instead, you should use the answer-specific endpoint to get richer results and better performance. For example, to receive only images, send the request using Image Search API. The Image Search API offers filters that are not available to the Web Search API.

Getting results from a specific site

To get search results from a specific domain, use Bing's site: operator in the query string. The response may contain results from other sites depending on the number of relevant results found on the specified site.

```
q=sailing+dinghies+site%3Acontososailing.com
```

> **Note**
>
> Depending on the query, if you use the `site:` query operator, there is the chance that the response may contain adult content regardless of the **safeSearch** setting. You should use `site:` only if you are aware of the content on the site and your scenario supports the possibility of adult content.

Specifying the content's freshness

It's possible that Bing will return content that's older than 30 days. If you want to ensure the freshness of the content that Bing returns, use the freshness query parameter. Freshness refers to the date that Bing originally discovered the webpage, not when the publisher published the webpage. The Webpage object's `datePublished` field tells you when Bing originally discovered the page.

You may set the parameter to one of the following time periods:

- Day — Return webpages that Bing discovered within the last 24 hours.
- Week — Return webpages that Bing discovered within the last 7 days.
- Month — Return webpages that Bing discovered within the last 30 days.

The following example filters the webpage results to those that Bing discovered in the last seven days:

```
freshness=week
```

You may also set this parameter to a custom date range in the form, `YYYY-MM-DD..YYYY-MM-DD`. For example:

```
freshness=2019-02-01..2019-05-30
```

To limit the results to a single date, set the *freshness* parameter to the specific date in the form, `YYYY-MM-DD`.

The results may include webpages that fall outside the specified period if the number of webpages that Bing finds that matches your criteria is less than the number of webpages you requested (or the default number that Bing returns).

Returning the top n answers

A response may include any number of answer types. To limit the number of answers in the response to the top two ranked answers, set the answerCount query parameter to 2. Bing chooses the answers based on ranking. For example, if Bing ranks the answers as webpages, images, videos, and relatedSearches, the response includes only webpages and images.

If the request includes the responseFilter and *answerCount* query parameters, both apply. For example, if you set *responseFilter* to webpages and news and *answerCount* to 2, the response contains only webpages since news is not ranked.

Promoting answers that are not ranked

To include answers that Bing would otherwise exclude because of ranking, use the promote query parameter. Set *promote* to a comma-delimited list of answer names that you want the response to include regardless of their ranking. For example, if the top ranked answers that Bing returns for a query are webpages, images, videos, and relatedSearches, the response would include those answers. But if you set answerCount to 2, Bing returns the top two ranked answers, which are webpages and images. If you want to ensure that Bing includes images and videos in the response, set *promote* to images and videos.

```
answerCount=2&promote=images%2Cvideos
```

Bing returns the top two answers, webpages and images, and promotes videos into the response.

If you set *promote* to news, the response doesn't include the news answer because it is not a ranked answer — you can promote only ranked answers.

The answers that you want to promote do not count against the *answerCount* limit. For example, if the ranked answers are news, images, and videos, and you set *answerCount* to 1 and *promote* to news, the response contains news and images. Or, if the ranked answers are videos, images, and news, the response contains videos and news.

You may use *promote* only if you specify the *answerCount* query parameter.

Limiting the number of webpages

By default, Web Search API returns 10 webpages. If you want to receive a different number of webpages, use the count query parameter. The following example shows how to get back only 5 webpages.

```
count=5
```

The *count* query parameter affects only the number of webpages that Bing returns; it does not affect the number of results that Bing includes for the other answers (for example, images or videos).

You can also use the *count* and *offset* query parameters together to page through all the webpages that match the user's intent. For details about paging results, see Paging results.

Filtering adult content

The safeSearch query parameter lets you filter out webpages, images, and videos with adult content. You may set this parameter to one of the following values:

- Off — Returns content with adult text and images but not adult videos.
- Moderate — Returns content with adult text but not adult images or videos.
- Strict — Does not return content with adult text, images, or videos.

The default is Moderate.

Next steps

- Learn about the response that Bing returns for the user's query.

- Learn how to highlight the user's search terms in the results that Bing returns.
- Learn about the quickstarts and samples that are available to help you get up and running fast.

Handling the web search response

Article • 04/05/2022

When you send a request to Web Search API, it returns a SearchResponse object in the response body. The object may include one or more of the following answer types:

JSON

```
{
    "_type": "SearchResponse",
    "queryContext": {...},
    "webPages": {...},
    "images": {...},
    "relatedSearches": {...},
    "videos": {...},
    "news": {...},
    "spellSuggestion": {...},
    "computation": {...},
    "timeZone": {...},
    "rankingResponse": {...},
    "entities": {...},
    "places": {...},
    "translations": {...},
}
```

Basically, Bing returns any answer it finds that's relevant to the user's query, which is typically a subset of the possible answers. But Bing always returns the following answers in each response:

JSON

```
{
    "_type": "SearchResponse",
    "queryContext": {...},
    "rankingResponse": {...}
}
```

But if an error occurs, the response body contains an ErrorResponse object. Bing returns an error response for all 400 level HTTP status codes. Read more

JSON

```
{
  "_type": "ErrorResponse",
  "errors": [
    {
      "code": "InvalidAuthorization",
```

```
      "subCode": "AuthorizationMissing",
      "message": "Authorization is required.",
      "moreDetails": "Subscription key is not recognized."
    }
  ]
}
```

The rest of this topic provides details about each of the answer types in the **SearchResponse** object.

> **Note**
>
> Because URL formats and parameters are subject to change without notice, use all URLs in Bing search results as-is. You should not take dependencies on the URL format or parameters except where noted.

Webpages answer

The webPages answer contains a list of links to webpages that Bing determined were relevant to the query. At a minimum, each webpage in the list includes the page's name, URL, display URL, a short description of the content, the date Bing found the content, and the ranking ID.

JSON

```
  "webPages": {
    "webSearchUrl": "https://www.bing.com/search?q=mt+rainier",
    "totalEstimatedMatches": 594000,
    "value": [
      {
        "id": "https://api.bing.microsoft.com/api/v7/#WebPages.0",
        "name": "Dinghy sailing",
        "url": "https://www.bing.com/cr?IG=3A43CA5...",
        "displayUrl": "https://en.contoso.com/wiki/Dinghy_sailing",
        "snippet": "Dinghy sailing is the activity of sailing small
boats...",
        "dateLastCrawled": "2017-04-05T16:25:00"
      },

      . . .
    ]
  }
```

Use `name` and `url` to create a hyperlink that takes the user to the webpage.

The webpage may include a few other fields that you should account for like deep links, a malware notification, and contractual rules.

Deep links

Deep links are related webpages that Bing found on the webpage's website. The **Webpage** object in this context includes only the name and URL fields and may include the snippet field.

JSON


```
        "deepLinks": [
          {
            "name": "Drug Product Database Online Query",
            "url": "https://health-products.canada.ca/dpd-bdpp/index-
eng.jsp"
          },
          {
            "name": "Access The Extracts",
            "url": "https://www.canada.ca/en/health-canada/services/drugs-
health-products/drug-products/drug-product-database/extracts.html"
          },
          {
            "name": "Product Monograph Brand Safety Updates",
            "url": "https://www.canada.ca/en/health-canada/services/drugs-
health-products/drug-products/drug-product-database/label-safety-assessment-
update/product-monograph-brand-safety-updates.html"
          }
        ],
```


Deep links can also have nested deep links.

JSON

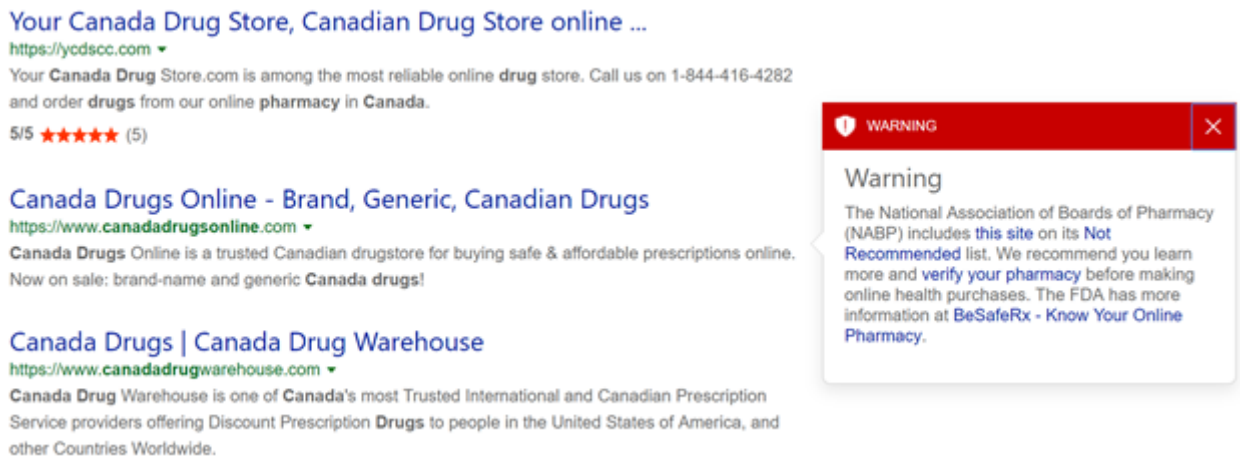

```
        "deepLinks": [
          {
            "name": "Webcams",
            "url":
"https://www.nps.gov/mora/learn/photosmultimedia/webcams.htm",
            "snippet": "See the view from the clouds! Camp Muir perches on
the side of Mount Rainier at an elevation of over 10,000 feet (3,000
meters). Updates every 5 minutes. NOTE: Down for repairs for the 2020
season. Thanks for your patience. View Webcam",
            "deepLinks": [
              {
                "name": "Road Status",
                "url": "https://www.nps.gov/mora/planyourvisit/road-
status.htm"
              },
              {
```

```
            "name": "Maps",
            "url": "https://www.nps.gov/mora/planyourvisit/maps.htm"
          },
          {
            "name": "Operating Hours & Seasons",
            "url": "https://www.nps.gov/mora/planyourvisit/hours.htm"
          },
          {
            "name": "Weather",
            "url": "https://www.nps.gov/mora/planyourvisit/weather.htm"
          }
        ]
      },
```

Malware notice

If Bing determines that a webpage may cause a potential issue for the user if they click the link, Bing provides a notice that you should display next to the webpage's link. Potential issues might be that the page contains malware, is a phishing site, or is not recommended for purchasing pharmaceuticals.

The following image shows how Bing might display the notice for the query, *canada drugs*. Bing displays the warning when the user hovers over the webpage in the search results page, tabs to it, or touches the webpage on a touch device.



Bing does not let the user click the link or deep links in the search results page. However, the warning does include a link that the user can click to navigate to the webpage if they choose to.

The following example shows what the notice looks like in the JSON response.

JSON

```
      "malware": {
        "malwareWarningType": "NABP",
```

```
            "warningExplanationUrl": "https://verifybeforeyoubuy.org/",
            "warningLetterUrl": "https://www.safe.pharmacy/not-recommended-
sites/",
            "beSafeRxUrl":
"https://www.fda.gov/Drugs/ResourcesForYou/Consumers/BuyingUsingMedicineSafe
ly/BuyingMedicinesOvertheInternet/BeSafeRxKnowYourOnlinePharmacy/default.htm
",
        },
```

If you follow Bing's approach of disabling the webpage's link in the search results, then you'd use the webpage's link in the notice's "visit anyway" link.

Contractual rules

If a webpage requires attribution, the **Webpage** object includes the `contractualRules` field. For webpages, you typically have to display the license under which the information is provided. The rule could target the webpage as a whole or a specific field of the **Webpage** object. Here's an example that targets the webpage's `snippet` text field.

JSON

```
        "contractualRules": [
          {
            "_type": "ContractualRules/LicenseAttribution",
            "targetPropertyName": "snippet",
            "targetPropertyIndex": 7,
            "mustBeCloseToContent": true,
            "license": {
              "name": "CC-BY-SA",
              "url": "http://creativecommons.org/licenses/by-sa/3.0/"
            },
            "licenseNotice": "Text under CC-BY-SA license"
          }
        ],
```

To learn more about attribution, see Data attribution.

Images answer

The images answer contains a list of images that Bing thought were relevant to the query. Each image in the list includes the image's URL, its size, its dimensions, and its encoding format. The **Image** object also includes a thumbnail URL and the thumbnail's dimensions.

JSON


```
  "images": {
    "id": "https://api.bing.microsoft.com/api/v7/#Images",
    "readLink": "https://api.bing.microsoft.com/api/v7/images/search?
q=dinghy+sailing&qpvt=dinghy+sailing",
    "webSearchUrl": "https://www.bing.com/images/search?
q=dinghy+sailing&qpvt=lady+gaga",
    "isFamilyFriendly": true,
    "value": [
      {
        "name": "Rich Passage Sailing Dinghy",
        "webSearchUrl": "https://www.bing.com/cr?IG=3A43CA5CA64...",
        "thumbnailUrl": "https://tse1.mm.bing.net/th?id=OIP....",
        "datePublished": "2011-10-29T11:26:00",
        "contentUrl": "http://upload.contoso.com/sailing/...",
        "hostPageUrl": "http://www.bing.com/cr?IG=3A43CA5CA6464....",
        "contentSize": "79239 B",
        "encodingFormat": "jpeg",
        "hostPageDisplayUrl": "http://en.contoso.com/wiki/File...",
        "width": 526,
        "height": 688,
        "thumbnail": {
            "width": 229,
            "height": 300
        },
        "insightsSourcesSummary": {
            "shoppingSourcesCount": 0,
            "recipeSourcesCount": 0
        }
      },

      . . .
    ]
  }
```


The `contentUrl` is the image's URL, the `hostPageUrl` is the URL of the webpage that includes the image, and `thumbnailUrl` is the URL to a thumbnail version of the image in `contentUrl`.

Depending on the user's device, you'd typically display a subset of the thumbnails, with an option for the user to see the remaining images.

You can also expand the thumbnail as the user hovers the cursor over it. Be sure to attribute the image if you expand it. For example, by extracting the host from `hostPageDisplayUrl` and displaying it below the image. For information about resizing the thumbnail, see Resizing and cropping thumbnails.

If the user clicks the thumbnail, use `webSearchUrl` to take the user to Bing's search results page for images, which contains a collage of the images.

For details about the image answer and images, see Image Search API.

Videos answer

The videosAnswer answer contains a list of videos that Bing thought were relevant to the query. Each video in the list includes the video's URL, its duration, its dimensions, and its encoding format. The **Video** object also includes thumbnail URL of the video and the thumbnail's dimensions.

JSON


```
  "videos": {
    "id": "https://api.bing.microsoft.com/api/v7/#Videos",
    "readLink": "https://api.bing.microsoft.com/api/v7/videos/search?
q=dinghy+sailing",
    "webSearchUrl": "https://www.bing.com/videos/search?q=dinghy+sailing",
    "isFamilyFriendly": true,
    "value": [
      {
        "name": "Sailing dinghy",
        "description": "Northwind Traders is a 12 foot gunter rigged...",
        "webSearchUrl": "https://www.bing.com/cr?IG=1CAE739681D84...",
        "thumbnailUrl": "https://tse2.mm.bing.net/th?id=OVP.wsKiL...",
        "datePublished": "2013-11-06T01:56:28",
        "publisher": [{
            "name": "Fabrikam"
        }],
        "contentUrl": "https://www.fabrikam.com/watch?v=MrVBWZpJjX",
        "hostPageUrl": "https://www.bing.com/cr?IG=1CAE739681D8400DB...",
        "encodingFormat": "mp4",
        "hostPageDisplayUrl": "https://www.fabrikam.com/watch?v=MrBWZpJjXo",
        "width": 1280,
        "height": 720,
        "duration": "PT3M47S",
        "motionThumbnailUrl": "https://tse2.mm.bing.net/th?id=OM.oa...",
        "embedHtml": "<iframe width=\"1280\" height=\"720\"
src=\"http://www....></iframe>",
        "allowHttpsEmbed": true,
        "viewCount": 19089,
        "thumbnail": {
            "width": 300,
            "height": 168
        },
        "allowMobileEmbed": true,
        "isSuperfresh": false
      },

      . . .
    ]
  }
```

Depending on the user's device, you'd typically display a subset of the videos and provide the user an option to view the remaining videos. You'd display a thumbnail of the video with the video's length, description (name), and attribution (publisher).

As the user hovers over the thumbnail you can use `motionThumbnailUrl` to play a thumbnail version of the video. Be sure to attribute the motion thumbnail when you display it.

If the user clicks the thumbnail, the following are the options for viewing the video:

- Use `hostPageUrl` to view the video on the host website (for example, YouTube).
- Use `webSearchUrl` to view the video in the Bing video browser.
- Use `embedHtml` to embed the video in your own experience.

For details about the video answer and videos, see Video Search API.

News answer

The news answer contains a list of news articles that Bing thought were relevant to the query. Each news article in the list includes the article's name, description, and URL to the article on the host's website. If the article contains an image, the object includes a thumbnail of the image.

JSON


```
  "news": {
    "id": "https://api.bing.microsoft.com/api/v7/#News",
    "readLink": "https://api.bing.microsoft.com/api/v7/news/search?
q=dinghy+sailing",
    "value": [
      {
        "contractualRules": [
          {
            "_type": "ContractualRules/TextAttribution",
            "text": "contoso.com"
          }
        ],
        "name": "WC Sailing Qualifies for America Trophy with...",
        "url": "http://www.bing.com/cr?IG=3445EEF15DAF4FFFBF7...",
        "image": {
          "contentUrl": "http://www.contoso.com/sports/sail...",
          "thumbnail": {
            "contentUrl": "https://www.bing.com/th?id=ON.1...",
            "width": 400,
            "height": 272
          }
        },
        "description": "The WC sailing team qualified for a...",
```

```
        "provider": [
          {
            "_type": "Organization",
            "name": "contoso.com",
            "image": {
              "thumbnail": {
                "contentUrl": "https://www.bing.com/th?
id=AR_85d7ddac409e7624f6b911ce58ac&pid=news"
              }
            }
          }
        ],
        "datePublished": "2017-04-16T21:56:00",
        "category": "Sports"
      },

      ...
    ]
  }
```

If Bing finds a video that's related to the article, it includes the video's URL in the **Video** object.

JSON

```
        "video": {
          "name": "America Trophy qualifiers",
          "motionThumbnailUrl": "https://wus-streaming-video-msn-
com.fabrikam.net/34572c32-1a0f-4576-...mp4",
          "thumbnail": {
            "width": 640,
            "height": 360
          }
        },
```

Depending on the user's device, you'd display a subset of the news articles with an option for the user to view the remaining articles. Use `name` and `url` to create a hyperlink that takes the user to the news article on the host's site. If the article includes an image, make the image clickable using `url`.

Be sure to use `contractualRules` to attribute the article. For information about attribution, see Data attribution.

For details about the news answer and news articles, see News Search API.

Related searches answer

The relatedSearches answer contains a list of the most popular related queries made by other users. Each query in the list includes a query string (`text`), a query string with hit highlighting characters (`displayText`), and a URL (`webSearchUrl`) to Bing's search results page for that query.

JSON

```
{
    "text": "dinghy racing teams",
    "displayText": "dinghy racing teams",
    "webSearchUrl": "https://www.bing.com/cr?IG=96C4CF214A0..."
}, ...
```

Use the `displayText` query string and the `webSearchUrl` URL to create a hyperlink that takes the user to the Bing search results page for the related query. You could also use the `text` query string in your own Web Search API query and display the results yourself.

For information about how to handle the highlighting markers in `displayText`, see Hit Highlighting.

Computation answer

If the user enters a mathematical expression or a unit conversion query, the response may contain a Computation answer. The `computation` answer contains the normalized expression and its result.

Unit conversions

A unit conversion query is a query that converts one unit to another. For example, *How many feet in 10 meters?* or *How many tablespoons in a 1/4 cup?*

The following shows the `computation` answer for *How many feet in 10 meters?*

JSON

```
  "computation": {
    "id": "https://www.bing.com/api/v7/#Computation",
    "expression": "10 meters",
    "value": "32.808399 feet"
  }
```

Mathematical expression

The following examples show different mathematical expressions and their corresponding `computation` answers.

Expression: (5+3)(10/2)+8

JSON

```
"computation": {
    "id": "https://www.bing.com/api/v7/#Computation",
    "expression": "((5+3)*(10/2))+8",
    "value": "48"
}
```

Expression: sqrt(4^2+8^2)

JSON

```
"computation": {
    "id": "https://www.bing.com/api/v7/#Computation",
    "expression": "sqrt((4^2)+(8^2))",
    "value": "8.94427191"
}
```

Expression: 30 6/8 - 18 8/16

JSON

```
"computation": {
    "id": "https://www.bing.com/api/v7/#WolframAlpha",
    "expression": "30 6/8-18 8/16",
    "value": "12.25"
}
```

Expression: 8^2+11^2-2811*cos(37)

JSON

```
"computation": {
        "id": "https://www.bing.com/api/v7/#Computation",
        "expression": "(8^2)+(11^2)-(2*8*11*cos(37))",
        "value": "44.4401502"
}
```

A mathematical expression may contain the following symbols:

Symbol	Description
+	Addition
-	Subtraction
/	Division
*	Multiplication
^	Power
!	Factorial
.	Decimal
()	Precedence grouping
[]	Function

A mathematical expression may contain the following constants:

Symbol	Description
Pi	3.14159...
Degree	Degree
i	Imaginary number
e	e, 2.71828...
GoldenRatio	Golden ratio, 1.61803...

A mathematical expression may contain the following functions:

Symbol	Description
Sort	Square root
Sin[x], Cos[x], Tan[x] Csc[x], Sec[x], Cot[x]	Trigonometric functions (with arguments in radians)
ArcSin[x], ArcCos[x], ArcTan[x] ArcCsc[x], ArcSec[x], ArcCot[x]	Inverse trigonometric functions (giving results in radians)
Exp[x], E^x	Exponential function
Log[x]	Natural logarithm

Symbol	Description
Sinh[x], Cosh[x], Tanh[x] Csch[x], Sech[x], Coth[x]	Hyperbolic functions
ArcSinh[x], ArcCosh[x], ArcTanh[x] ArcCsch[x], ArcSech[x], ArcCoth[x]	Inverse hyperbolic functions

Mathematical expressions that contain variables (for example, 4x+6=18, where x is the variable) are not supported.

TimeZone answer

If the user enters a time or date query, the response may contain a TimeZone answer. This answer supports implicit or explicit queries. An implicit query, such as *What time is it?*, returns the local time based on the user's location. An explicit query, such as *What time is it in Seattle?*, returns the local time for Seattle, WA.

Implicit queries, such as *What time zone am I in?*, require the user's location to provide accurate results. Although optional, you should always provide the user's location using the X-Search-Location and X-MSEdge-ClientIP headers. If you don't provide the user's location and Bing thinks the query would benefit from the user's location, it sets the QueryContext object's `askUserForLocation` field to **true**.

JSON

```
"queryContext": {
  "originalQuery": "what's the time",
  "askUserForLocation": true
},
```

The `timeZone` answer provides the name of the location, the current UTC date and time at the specified location, and the UTC offset. If the boundary of the location is within multiple time zones, the answer contains the current UTC date and time of all time zones within the boundary. For example, because Florida State falls within two time zones, the answer contains the local date and time of both time zones.

If the query requests the time of a state or country/region, Bing determines the primary city within the location's geographical boundary and returns it in the `primaryCityTime` field. If the boundary contains multiple time zones, the remaining time zones are returned in the `otherCityTimes` field.

The following examples show queries that return the `timeZone` answer.

Query: What time is it?

JSON


```
"timeZone": {
  "id": "https://www.bing.com/api/v7/#TimeZone",
  "primaryCityTime": {
    "location": "Redmond, Washington, United States",
    "time": "2015-10-27T08:38:12.1189231Z",
    "utcOffset": "UTC-7"
  }
}
```


Query: What time is it in the Pacific time zone?

JSON


```
"timeZone": {
  "id": "https://www.bing.com/api/v7/#TimeZone",
  "primaryCityTime": {
    "location": "Pacific Time Zone",
    "time": "2015-10-23T12:33:19.0728146Z",
    "utcOffset": "UTC-7"
  }
}
```


Query: Time in Florida?

JSON


```
"timeZone": {
  "id": "https://www.bing.com/api/v7/#TimeZone",
  "primaryCityTime": {
    "location": "Tallahassee, Florida, United States",
    "time": "2015-10-23T13:04:56.6774389Z",
    "utcOffset": "UTC-4"
  },
  "otherCityTimes": [
    {
      "location": "Pensacola, Florida, United States",
      "time": "2015-10-23T12:04:56.6664294Z",
      "utcOffset": "UTC-5"
    }
  ]
}
```


Query: What time is it in the U.S.

JSON

```
  "timeZone": {
    "id": "https://www.bing.com/api/v7/#TimeZone",
    "primaryCityTime": {
      "location": "Washington, D.C., United States",
      "time": "2015-10-23T15:27:59.8892745Z",
      "utcOffset": "UTC-4"
    },
    "otherCityTimes": [
      {
        "location": "Honolulu",
        "time": "2015-10-23T09:27:59.8892745Z",
        "utcOffset": "UTC-10"
      },
      {
        "location": "Anchorage",
        "time": "2015-10-23T11:27:59.8892745Z",
        "utcOffset": "UTC-8"
      },
      {
        "location": "Phoenix",
        "time": "2015-10-23T12:27:59.8892745Z",
        "utcOffset": "UTC-7"
      },
      {
        "location": "Los Angeles",
        "time": "2015-10-23T12:27:59.8942788Z",
        "utcOffset": "UTC-7"
      },
      {
        "location": "Denver",
        "time": "2015-10-23T13:27:59.8812681Z",
        "utcOffset": "UTC-6"
      },
      {
        "location": "Chicago",
        "time": "2015-10-23T14:27:59.8892745Z",
        "utcOffset": "UTC-5"
      }
    ]
  }
```


This answer also supports determining the difference between time zones or calculating dates such as 90 days from today.

Query: What's the date

JSON


```
  "timeZone": {
    "id": "https://<host>/api/v7/#TimeZone",
    "primaryResponse": "Wednesday, December 11, 2019",
```

```
    "description": "Date in Redmond, WA"
  },
```

Query: PST to EST

JSON

```
  "timeZone": {
    "id": "https://<host>/api/v7/#TimeZone",
    "description": "Pacific Standard Time is behind Eastern Standard Time
by",
    "timeZoneDifference": {
      "location1": {
        "location": "Pacific Standard Time",
        "time": "2019-12-11T11:54:14.5567693Z",
        "utcOffset": "UTC-8",
        "timeZoneName": "PST"
      },
      "location2": {
        "location": "Eastern Standard Time",
        "time": "2019-12-11T14:54:14.5567693Z",
        "utcOffset": "UTC-5",
        "timeZoneName": "EST"
      },
      "text": "3 hours"
    }
  },
```

Query: 90 days from today

JSON

```
  "timeZone": {
    "id": "https://<host>/api/v7/#TimeZone",
    "date": "Tuesday, March 10, 2020"
  },
```

Query: How may weeks in 2020

JSON

```
  "timeZone": {
    "id": "https://<host>/api/v7/#TimeZone",
    "primaryResponse": "52 weeks and 2 days",
    "description": "There are 52 weeks and 2 days from January 1, 2020 to
January 1, 2021"
  },
```

Query: What time zone am I in

JSON


```
  "timeZone": {
    "id": "https://<host>/api/v7/#TimeZone",
    "primaryTimeZone": {
      "location": "Redmond, WA",
      "time": "2019-12-11T11:56:13.6395905Z",
      "utcOffset": "UTC-8",
      "timeZoneName": "Pacific Standard Time"
    }
  },
```


SpellSuggestion answer

If Bing determines that the user may have intended to search for something different, the response includes a SpellSuggestions object. For example, if the user searches for *carlos pen*, Bing may determine that the user likely intended to search for Carlos Pena instead (based on past searches by others of *carlos pen*). The following shows an example spell response.

JSON


```
  "spellSuggestions": {
    "id": "https://www.bing.com/api/v7/#SpellSuggestions",
    "value": [
      {
        "text": "carlos pena",
        "displayText": "carlos pena"
      }
    ]
  },
```


QueryContext answer

Each response contains a QueryContext object that provides the context that Bing used for the request. At a minimum, the context contains the user's query string.

JSON

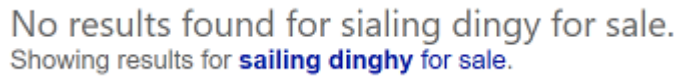

```
  "queryContext": {
    "originalQuery": "mt rainier"
  },
```

If the user's query string contains a spelling mistake, the context includes `alteredQuery` field, with contains the corrected spelling. Bing uses the `alteredQuery` query string in the request instead of the user's query string.

JSON

```
"queryContext": {
    "originalQuery": "sialing dingy for sale",
    "alteredQuery": "sailing dinghy for sale",
    "alterationOverrideQuery": "+sialing +dingy for sale"
}
```

The following example shows how Bing uses this information in the UX. If you provide the same feature and the user chooses to use their original query string, use the `alterationOverrideQuery` query string in the request. When encoded, the override string looks like, %2Bsialing+%2Bdingy+for+sale. This forces Bing to use the user's original string instead of the corrected string.



User location

Some queries are helped by knowing the user's location. For example, if the user asks for *today's weather* or *restaurants near me*, the context object includes the `askUserForLocation` field. If *true*, you should send a new query and include the X-MSEdge-ClientIP and X-Search-Location headers with the user's location.

JSON

```
  "queryContext": {
    "originalQuery": "today's weather",
    "askUserForLocation": true
  },
```

Adult intent

If the user's query has adult intent, the context includes the `adultIntent` field. If **true**, the user's safeSearch setting determines the content that Bing returns.

JSON

```
  "queryContext": {
    "originalQuery": "xxx movies",
    "adultIntent": true
  },
```

RankingResponse answer

Each Bing response contains a RankingResponse object that suggests the order in which you should display the Bing answers and the results within each answer. For information about how to use the **RankingResponse** object, see Using ranking to display search results.

The following example shows parts of the ranking response answer for brevity.

JSON

```
  "rankingResponse": {
    "mainline": {
      "items": [
        {
          "answerType": "Videos",
          "value": {
            "id": "https://<host>/api/v7/#Videos"
          }
        },
        {
          "answerType": "News",
          "value": {
            "id": "https://<host>/api/v7/#News"
          }
        },
        {
          "answerType": "WebPages",
          "resultIndex": 0,
          "value": {
            "id": "https://<host>/api/v7/#WebPages.0"
          }
        },

        . . .

        {
          "answerType": "RelatedSearches",
          "value": {
            "id": "https://<host>/api/v7/#RelatedSearches"
          }
        }
      ]
    },
```

```
    "sidebar": {
      "items": [
        {
          "answerType": "Images",
          "value": {
            "id": "https://<host>/api/v7/#Images"
          }
        },
        {
          "answerType": "Entities",
          "resultIndex": 0,
          "value": {
            "id": "https://<host>/api/v7/#Entities.0"
          }
        }
      ]
    }
  }
```

Translations answer

The Translations answer contains the translation of a word or phrase from one language to another. The context used in the translation comes from the query string and other signals. For example, in the query, *amigo in english*, Spanish is inferred from Amigo. If the query string doesn't explicitly specify the language to translate the text into (for example, if the query is *bon appetit*), Bing infers the language from the browser's language setting.

The `originalText` field contains the word or phrase to translate and `translatedText` contains the translated text. If the translation request cannot be satisfied within system-defined thresholds, the `translatedText` field is set to "...". If this occurs, you shouldn't display the answer.

JSON

```
  "translations": {
    "id": "https://<host/api/v7/#Translations",
    "contractualRules": [
      {
        "_type": "ContractualRules/LinkAttribution",
        "text": "Microsoft Translator",
        "url": "http://www.bing.com/translator/?
ref=TThis&text=hello&from=en&to=es"
      }
    ],
    "attributions": [
      {
```

```
        "providerDisplayName": "Microsoft Translator",
        "seeMoreUrl": "http://www.bing.com/translator/?
ref=TThis&text=hello&from=en&to=es"
      }
    ],
    "originalText": "hello",
    "translatedText": "Hola",
    "translatedLanguageName": "es",
    "inLanguage": "en"
  },
```

Be sure to use `contractualRules` to attribute the article. For information about attribution, see Data attribution.

Entities answer

The entities answer contains a list of entity objects that Bing thought were relevant to the query. Each entity in the list identifies a person, place, or thing. Bing returns well-known entities only. Well-known people may include singers, actors, athletes, models, and others. Places refers to well-known tourist attractions, organizations, and localities such as a cities, states, countries, and regions. Things cover everything else not covered by places and people, such as animals, foods, drinks, books, songs, movies, and more. For information about places such as restaurants, hotels, or other local businesses, see the Places answer.

Dominant entity versus disambiguation entities

The list of entities may contain a single dominant entity, multiple disambiguation entities, or both. The following example fragments show the different entity types. See the `entityScenario` field.

Dominant-only entity (query is Seattle)

JSON

```
  "entities": {
    "value": [
      {
        "name": "Seattle",
        "entityPresentationInfo": {
          "entityScenario": "DominantEntity",
          "entityTypeHints": [
            "City"
          ]
        },
      }
```

```
    ]
  },
```

Dominant and disambiguation entities (query is Mt Rainier)

JSON

```
  "entities": {
    "value": [
      {
        "id": "https://<host>/api/v7/#Entities.0",
        "name": "Mount Rainier",
        "entityPresentationInfo": {
          "entityScenario": "DominantEntity",
          "entityTypeHints": [
            "Place"
          ]
        },
      },
      {
        "id": "https://<host>/api/v7/#Entities.1",
        "name": "Mount Rainier National Park",
        "entityPresentationInfo": {
          "entityScenario": "DisambiguationItem"
        },
      }
    ]
  },
```

Disambiguation-only entities (query is Washington)

JSON

```
  "entities": {
    "value": [
      {
        "id": "https://<host>/api/v7/#Entities.0",
        "name": "Washington, D.C.",
        "entityPresentationInfo": {
          "entityScenario": "DisambiguationItem",
          "entityTypeHints": [
            "City"
          ]
        },
      },
      {
        "id": "https://<host>/api/v7/#Entities.1",
        "name": "The Washington Post",
        "entityPresentationInfo": {
          "entityScenario": "DisambiguationItem",
          "entityTypeHints": [
            "Organization"
```

```
          ]
        },
      },
      {
        "id": "https://<host>/api/v7/#Entities.2",
        "name": "Washington",
        "entityPresentationInfo": {
          "entityScenario": "DisambiguationItem",
          "entityTypeHints": [
            "State"
          ]
        },
      }
    ]
  },
```

Bing returns a dominant entity when there is no ambiguity as to which entity satisfies the request. If multiple entities could satisfy the request, the list contains more than one disambiguation entities. For example, if the request uses the generic title of a movie franchise, the list likely contains disambiguation entities. But, if the request specifies a specific title from the franchise, the list likely contains a single dominant entity.

The EntityPresentationInfo object contains information that tells you whether the entity is a dominant entity or a disambiguation entity (see the `entityScenario` field). The object may also include one or more hints (see the `entityTypeHints` field) that tell you the entity's type. The list of hints could contain a single hint such as Movie or a list of hints such as Place, LocalBusiness, Restaurant. Each successive hint in the array narrows the entity's type. But not all entities include type hints.

If the list contains one or more disambiguation entities (the `entityScenario` field is set to DisambiguationItem), consider displaying a list of entities and letting the user select the one they're interested in. The **Entity** object's `name` field contains the entity's name. Use the name along with the URL in the `url` field, if it exists, or the `webSearchUrl` field to create a hyperlink. The entity includes the `url` field only if Bing found a website or webpage for the entity. The URL in the `webSearchUrl` field takes the user to Bing's search result page for the entity.

The following answer shows what the JSON response looks like for the query, *mt rainier*. Most entities include the entity's name, short description, contractual rules, and URL to Bing's search results page where the user can get more information about the entity. The optional fields that not all entities include are the `image`, `url`, and `entityTypeHints` fields.

JSON

```
{
  "_type": "SearchResponse",
  "queryContext": {
    "originalQuery": "mt rainier"
  },
  "entities": {
    "value": [
      {
        "id": "https://<host>/api/v7/#Entities.0",
        "contractualRules": [
          {
            "_type": "ContractualRules/LicenseAttribution",
            "targetPropertyName": "description",
            "mustBeCloseToContent": true,
            "license": {
              "name": "CC-BY-SA",
              "url": "http://creativecommons.org/licenses/by-sa/3.0/"
            },
            "licenseNotice": "Text under CC-BY-SA license"
          },
          {
            "_type": "ContractualRules/LinkAttribution",
            "targetPropertyName": "description",
            "mustBeCloseToContent": true,
            "text": "Wikipedia",
            "url": "http://en.wikipedia.org/wiki/Mount_Rainier"
          },
          {
            "_type": "ContractualRules/MediaAttribution",
            "targetPropertyName": "image",
            "mustBeCloseToContent": true,
            "url": "http://en.wikipedia.org/wiki/Mount_Rainier"
          }
        ],
        "webSearchUrl": "https://www.bing.com/entityexplore?
q=Mount+Rainier...",
        "name": "Mount Rainier",
        "image": {
          "name": "Mount Rainier",
          "thumbnailUrl": "https://www.bing.com/th?id=AMMS_65523b5b...",
          "provider": [
            {
              "_type": "Organization",
              "url": "http://en.wikipedia.org/wiki/Mount_Rainier"
            }
          ],
          "hostPageUrl":
"http://upload.wikimedia.org/commons/Mount_Rainier...",
          "width": 110,
          "height": 110,
          "sourceWidth": 474,
          "sourceHeight": 316
        },
        "description": "Mount Rainier, also known as Tahoma or Tacoma, is a
```

```
large active strato...",
        "entityPresentationInfo": {
          "entityScenario": "DominantEntity",
          "entityTypeHints": [
            "Place"
          ]
        },
        "bingId": "9ae3e6ca-81ea-6fa1-ffa0-42e1d7890906"
      },
      {
        "id": "https://<host>/api/v7/#Entities.1",
        "contractualRules": [
          {
            "_type": "ContractualRules/MediaAttribution",
            "targetPropertyName": "image",
            "mustBeCloseToContent": true,
            "url":
"http://en.wikipedia.org/wiki/Mount_Rainier_National_Park"
          }
        ],
        "webSearchUrl": "https://www.bing.com/entityexplore?
q=Mount+Rainier+Nat...",
        "name": "Mount Rainier National Park",
        "url": "https://www.nps.gov/mora/index.htm",
        "image": {
          "name": "Mount Rainier National Park",
          "thumbnailUrl": "https://www.bing.com/th?
id=AMMS_4bd2812676c04d54ef0e...",
          "provider": [
            {
              "_type": "Organization",
              "url":
"http://en.wikipedia.org/wiki/Mount_Rainier_National_Park"
            }
          ],
          "hostPageUrl":
"http://upload.wikimedia.org/Mount_Rainier_7437.JPG",
          "width": 72,
          "height": 72,
          "sourceWidth": 474,
          "sourceHeight": 355
        },
        "description": "Mount Rainier National Park is an American national
park located in southeast...",
        "entityPresentationInfo": {
          "entityScenario": "DisambiguationItem"
        },
        "bingId": "9a8a1f72-a577-9f45-e275-1d969576f069"
      }
    ]
  },
```


Entity attribution

Entities may include the `contractualRules` field, which contains one or more attributions that you must apply when you display the entity. Not all entities include rules. If the entity provides contractual rules, you must abide by them.

Entity information typically comes from third parties. You are responsible for ensuring that your use is appropriate; for example, by complying with any creative commons license your user experience relies on.

For information about applying attribution, see Data Attribution.

Places answer

The places answer contains a list of local business entity objects that Bing thought were relevant to the query. Bing returns this answer only when the query specifies the name of a local business or asks for a type of business. For example, *microsoft store* and *restaurants near me*. Each place in the list identifies a restaurant, hotel, or other local business.

> **Note**
>
> The Places answer supports only U.S. business locations.

> **Note**
>
> You, or a third party on your behalf, may not use, retain, store, cache, share, or distribute any data from the Places answer for the purpose of testing, developing, training, distributing or making available any non-Microsoft service or feature.

Local aware queries such as *restaurant near me* require the user's location to provide accurate results. Although optional, you should always provide the user's location using the X-Search-Location and X-MSEdge-ClientIP headers. The X-Search-Location header uses the user's geographical coordinates (latitude and longitude).

```
X-Search-Location: lat:47.806897;long:-122.221304;re:30
```

If you don't provide the user's location and Bing thinks the query would benefit from the user's location, it sets the `askUserForLocation` field of QueryContext to **true**.

JSON

```
  "queryContext": {
    "originalQuery": "italian restaurants near me",
    "askUserForLocation": true
  },
```

The EntityPresentationInfo object contains hints that identify the local entity's type. The list contains a list of hints such as Place, LocalBusiness, Restaurant. Each successive hint in the array narrows the entity's type.

JSON

```
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
```

The local entity includes the place's name, address, and telephone number. If the URL to the place's website is known, the entity includes it, too. When you display the entity information, use the URL in the `url` field to create link that takes the user to the business' website; otherwise, use the URL in `webSearchUrl` to take the user to Bing's search results page for the entity.

The following example shows what the JSON response looks like for the query, *coffee near me*.

JSON

```
  "places": {
    "value": [
      {
        "_type": "Restaurant",
        "id": "https://<host>/api/v7/#Places.0",
        "webSearchUrl": "https://www.bing.com/entityexplore?
q=Fourth+Coffee...",
        "name": "Fourth Coffee",
        "url": "http://www.fourthcoffee.com/",
        "entityPresentationInfo": {
          "entityScenario": "ListItem",
          "entityTypeHints": [
            "Place",
            "LocalBusiness",
            "Restaurant"
          ]
        },
```

```
        "address": {
          "addressLocality": "Bothell",
          "addressRegion": "WA",
          "postalCode": "98021",
          "addressCountry": "US",
          "neighborhood": "Bothell"
        },
        "telephone": "(425) 555-1234"
      },

      . . .

    ]
  },
```

Note that the address' `neighborhood` field may contain an empty string.

Note that the `_type` field identifies the local entity object's type. The above example shows the object's type as Restaurant. Others object types include Hotel and LocalBusiness.

Next steps

- Learn about use and display requirements for displaying Bing Web Search results.
- Learn about how to use the `RankingResponse` object to order the search results in your UX.
- Learn about the JSON objects found in the response.
- Learn about the hit highlighting characters found in the search results.
- Learn how to page webpage results.

Use ranking to display search results

Article • 10/28/2020

Each search response includes a RankingResponse answer tells you how to display the search results. The ranking response groups results by mainline content and sidebar content for a traditional search results page. If you don't display the results in a traditional mainline and sidebar format, you must provide the mainline content higher visibility than the sidebar content.

The following example fragment shows what the **RankingResponse** answer looks like in the JSON response for the query, *how to use saffron threads*.

JSON

```
"rankingResponse": {
  "mainline": {
    "items": [
      {
        "answerType": "Videos",
        "value": {
          "id": "https://<host>/api/v7.0/#Videos"
        }
      },
      {
        "answerType": "WebPages",
        "resultIndex": 0,
        "value": {
          "id": "https://<host>/api/v7.0/#WebPages.0"
        }
      },
      {
        "answerType": "WebPages",
        "resultIndex": 1,
        "value": {
          "id": "https://<host>/api/v7.0/#WebPages.1"
        }
      },

      . . .

      {
        "answerType": "WebPages",
        "resultIndex": 9,
        "value": {
          "id": "https://<host>/api/v7.0/#WebPages.9"
        }
      },
      {
        "answerType": "RelatedSearches",
        "value": {
```

```
            "id": "https://<host>/api/v7.0/#RelatedSearches"
          }
        }
      ]
    },
    "sidebar": {
      "items": [
        {
          "answerType": "Entities",
          "resultIndex": 0,
          "value": {
            "id": "<host>/api/v7.0/#Entities.0"
          }
        }
      ]
    }
  }
```

Within each group (mainline or sidebar), the Items array identifies the order that the content must appear in. Each item provides the following two ways to identify the result within an answer.

- `answerType` and `resultIndex`

  The `answerType` field identifies the answer (for example, the News answer) and the `resultIndex` field identifies a result within the answer (for example, a news article). The result index is zero based.

- `value`

  The `value` field contains an ID that matches the ID of either an answer or a result within the answer. Either the answer or the results contain the ID but not both. The fragment portion of the URI identifies the answer type and index. For example, `https://<host>/api/v7.0/#WebPages.9`, identifies the 10th webpage in the Webpages answer.

Because the videos ranking item doesn't include the `resultIndex` field and the `id` URI is missing the index value, you'd display all video results together.

JSON

```
        {
          "answerType": "Videos",
          "value": {
            "id": "https://<host>/api/v7.0/#Videos"
          }
        },
```

The following JSON shows the videos answer in the response. Notice that the Videos answer object includes the id field and the Video result objects don't. Either the answer will contain the id field or the results will, but not both.

JSON


```
  "videos": {
    "id": "https://<host>/api/v7.0/#Videos",
    "readLink": "https://<host>/api/v7.0/videos/search?
q=how+to+use+saffron+threads",
    "webSearchUrl": "https://www.bing.com/videos/search?
q=how+to+use+saffron+threads",
    "isFamilyFriendly": true,
    "value": [
      {
        "webSearchUrl": "https://www.bing.com/videos/search?
q=how%20to%20use%20sa...",
        "name": "How to use Saffron Threads",
        "description": "How to use Saffron Threads. Best way to use saffron
threads for maximum flavour and color.",
        "thumbnailUrl": "https://tse3.mm.bing.net/th?id=OVP.qj4aGA...",
        "datePublished": "2018-08-21T17:01:27.0000000",
        "publisher": [
          {
            "name": "Contoso"
          }
        ],
        "isAccessibleForFree": true,
        "contentUrl": "https://www.contoso.com/watch?v=Ai7LksfYDPs",
        "hostPageUrl": "https://www.contoso.com/watch?v=Ai7LksfYDPs",
        "encodingFormat": "mp4",
        "hostPageDisplayUrl": "https://www.contoso.com/watch?v=Ai7LksfYDPs",
        "width": 1280,
        "height": 720,
        "duration": "PT45S",
        "motionThumbnailUrl": "https://tse3.mm.bing.net/th?
id=OM1.F1163Ia21PXw5w_...",
        "embedHtml": "<iframe width=\"1280\" height=\"720\"
src=\"http://www.youtube.com/embed/Ai7LksfYDPs?autoplay=1\"
frameborder=\"0\" allowfullscreen></iframe>",
        "allowHttpsEmbed": true,
        "viewCount": 349,
        "thumbnail": {
          "width": 160,
          "height": 120
        },
        "allowMobileEmbed": true,
        "isSuperfresh": false
      },
```


However, because the Webpages answer doesn't include an `id` field, you'd display all webpages individually based on the ranking (each webpage includes an `id` field). Either

the answer will contain the `id` field or the results will, but not both.

JSON


```
  "webPages": {
    "webSearchUrl": "https://www.bing.com/search?
q=how+to+use+saffron+threads",
    "totalEstimatedMatches": 1020000,
    "value": [
      {
        "id": "https://<host>/api/v7.0/#WebPages.0",
        "name": "3 Ways to Prepare Saffron",
        "url": "https://www.fabrikam.com/Prepare-Saffron",
        "about": [
          {
            "name": "Saffron"
          }
        ],
        "isFamilyFriendly": true,
        "displayUrl": "https://www.fabrikam.com/Prepare-Saffron",
        "snippet": "Measure the saffron threads. Your recipe will usually
tell you how much saffron to use...",
        "dateLastCrawled": "2020-02-18T22:19:00.0000000Z",
        "language": "en",
        "isNavigational": false
      },
```


To use the ranking ID, simply match the ranking ID with the ID of an answer or one of its results. If you use the `answerType` and `resultIndex` fields, use `answerType` to identify the answer that contains the results to display. Then, use `resultIndex` to index through the answer's results to get the result to display.

Based on the ranking response example for the saffron query, you'd display the following search results in the mainline:

- All the videos (or several videos with a link to view the others)
- Webpages 0 through 9
- All the related searches

And you'd display the following search results in the sidebar:

- Entity 0

Next steps

- Learn about promoting unranked results.
- Learn about limiting the number of ranked answers in the response.

- See the ranking to tutorial to see how to display search results in C#.

Using decoration markers to highlight text

Article • 02/16/2022

Hit highlighting is when Bing highlights words or phrases from the user's search string that were found in search result strings. Bing uses either Unicode characters or HTML tags to mark the words or phrases in the webpage's name, display URL, and snippet text. Bing may mark other terms that Bing finds relevant.

By default, Bing doesn't highlight words or phrases in display strings. To enable hit highlighting, set the `textDecorations` query parameter in your request to **true**.

To specify whether you want Bing to use Unicode characters or HTML tags to mark the words or phrases, set the `textFormat` query parameter one off the following possible values.

- Raw — Uses Unicode characters to mark content that needs special formatting. The Unicode characters are in the range E000 through E019. For example, Bing uses E000 and E001 to mark the beginning and end of words or phrases for hit highlighting.
- HTML — Uses HTML tags to mark content that needs special formatting. For example, Bing uses <b> tags to mark the beginning and end of words or phrases for hit highlight.

The default is Raw.

Hit highlighting example

The following example shows a web result for `Sailing Dinghy`. Bing marked the beginning and end of the query term using the E000 and E001 Unicode characters.

```
{
     "name" : "Sailing dinghies, Sailing tenders - All boating and marine ...",
     "url" : "http:\/\/ www.contoso.com \/cr?IG=A56F7B1...",
     "displayUrl" : " www.contoso.com \/boat-manufacturer\/sailing-dinghy-1760.html",
     "snippet" : "Find all the manufacturers of sailing dinghies and contact...",
     "dateLastCrawled" : "2017-09-30T12:42:00"
},
```

Before displaying the result in your user interface, replace the Unicode characters with ones that are appropriate for your display format.

Additional text decorations

Bing can return several different text decorations. For example, a `Computation` answer can contain subscript markers for the query term `log(2)` in the `expression` field.

```
"computation": {
     "expression": "log10(2)",
     "value": "0.301029996"
},
```

If the request did not specify decorations, the `expression` field would contain `log10(2)`.

If `textDecorations` is **true**, Bing may include the following markers in the display strings of answers. If there is no equivalent HTML tag, the table cell is empty.

Unicode	HTML	Description
U+E000		Marks the beginning of the query term (hit highlighting).
U+E001		Marks the end of the query term.
U+E002		Marks the beginning of italicized content.
U+E003		Marks the end of italicized content.
U+E004	 	Marks a line break.
U+E005		Marks the beginning of a phone number.
U+E006		Marks the end of a phone number.
U+E007		Marks the beginning of an address.
U+E008		Marks the end of an address.
U+E009		Marks a non-breaking space.
U+E00C		Marks the beginning of bold content.
U+E00D		Marks the end of bold content.
U+E00E		Marks the beginning of content whose background should be lighter than its surrounding background.
U+E00F		Marks the end of content whose background should be lighter than its surrounding background.
U+E010		Marks the beginning of content whose background should be darker than its surrounding background.

Unicode	HTML	Description
U+E011		Marks the end of content whose background should be darker than its surrounding background.
U+E012	<del>	Marks the beginning of content that should be struck through.
U+E013	</del>	Marks the end of content that should be struck through.
U+E016	<sub>	Marks the beginning of subscript content.
U+E017	</sub>	Marks the end of subscript content.
U+E018	<sup>	Marks the beginning of superscript content.
U+E019	</sup>	Marks the end of superscript content.

Paging search results

Article • 10/28/2020

When you call any of the Bing APIs (for example, the Web Search API or Image Search API), the API returns a list of results. The list is a subset of the total number of results that may be relevant to the query. To get the estimated total number of available results, access the answer object's `totalEstimatedMatches` field.

The following example shows the `totalEstimatedMatches` field for News Search API.

JSON

```
{
  "_type": "News",
  "readLink": "https://<host>/api/v7.0/news/search?q=Sports",
  "queryContext": {
    "originalQuery": "Sports"
  },
  "totalEstimatedMatches": 118000,
  "value": [. . .]
}
```

The estimated number of matches **is only an estimate** and may likely change from request to request.

Paging through search results

To page through the results, use the *count* and *offset* query parameters.

The *count* parameter specifies the number of results to return in the response. The maximum number of results that you may request in the response is API specific (see Count values by API). For example, the maximum count value that you may specify for the Image Search API is 150.

The *offset* parameter specifies the number of results to skip. The offset is zero-based and should be less than (`totalEstimatedMatches` - *count*).

If your user interface presents 20 news articles per page, set *count* to 20 and *offset* to 0 to get the first page of results. For each subsequent page, increment *offset* by 20 (for example, 20, 40).

The following shows an example that requests 20 news articles beginning at offset 40.

```
https://<host>/api/v7.0/news/search?q=sailing&count=20&offset=40&mkt=en-us
```

Because each API sets a default value for *count*, you may specify only *offset*. For example, if the News Search API's default count is 20, you only need to include the *offset* query parameter.

```
https://<host>/api/v7.0/news/search?q=sailing&offset=40&mkt=en-us
```

Count values by API

The following table list the default and maximum count value per API.

API	Default	Maximum
Web Search	10	50
Image Search	35	150
News Search	10	100
Video Search	35	105

For the Image, News, and Video APIs, paging applies to only the general search endpoint. For example, you may not use paging with the trending endpoints.

Paging web search results

The Web Search API returns results that include webpages and may include other answers like images, videos, and news. When you page the search results, you are paging the webpage results and not the other answers. For example, if you set count to 15, Bing returns 15 webpage results, but may return 35 images and 4 news articles.

The answers that Bing returns from page to page is unknown. For example, Bing may include news on the first page but not the second page, or vise-versa.

Note that if you specify the *responseFilter* query parameter and do not include Webpages in the list of filters, you should not use the *count* and *offset* parameters.

Paging image and video results

Typically, if you page 30 images at a time, you set the *offset* query parameter to 0 on your first request, and increment *offset* by 30 on each subsequent request. However, some results in the subsequent response may be duplicates of the previous response. For example, the first two images in the response may be the same as the last two images from the previous response.

To eliminate duplicate results, set the *offset* query parameter to the value in the `nextOffset` field of the ImageAnswer object. The `nextOffset` value adjusts for duplicates.

JSON


```
{
  "_type": "Images",
  "readLink": "images/search?q=nurburgring",
  "webSearchUrl": "https://www.bing.com/images/search?
q=nurburgring&FORM=OIIARP",
  "queryContext": {. . .},
  "totalEstimatedMatches": 933,
  "pivotSuggestions": [. . .],
  "queryExpansion": [. . .],
  "relatedSearches": [. . .],
  "nextOffset": 65,
  "currentOffset": 0,
  "value": [. . .]
}
```


For example, if you want to page 30 images at a time, you'd set *count* to 30 and *offset* to 0 in your first request. In your next request, you'd set *count* to 30 and *offset* to the value of `nextOffset`. The value of `nextOffset` will be 30 if there are no duplicates or it may be 32 if there are 2 duplicates.

Use the same technique when paging videos.

Next steps

- Learn about using rank to display search results.

Resize and crop thumbnail images

Article • 02/15/2024

> **Note**
>
> Be sure when cropping and resizing thumbnail images that you're doing so in accordance with a search scenario that respects third party rights, as required by your Bing Search API **use and display requirements**.

Some answers include URLs to thumbnail images served by Bing. The following examples show several thumbnail URL formats that you might find in an answer.

curl

```
https://<host>/th?id=ON.A317B1375C1ADD5C646CB8635AE4E9&pid=News
https://<host>/th?id=OVP.VC36z4V1MoxzlwETyoaQHgFo&pid=Api
https://<host>/th?
id=AMMS_c1a785119b4d9fc14b6571a2a2f728&w=110&h=110&c=7&rs=1&qlt=80&cdv=1&pid
=16.1
```

You may resize and crop thumbnail images. To resize a thumbnail:

1. Remove all query parameters except the *id* and *pid* parameters.
2. Add only the *w* (width) or *h* (height) query parameter, but not both.
3. Set the *w* or *h* query parameter to the desired size, in pixels. Bing maintains the aspect ratio for you.

If you specify an image size that's larger than the thumbnail's original size, Bing adds white padding around the image as needed. For example, if the image's original size is 474x316 and you set *w* (width) to 500, Bing returns a 500x333 image. Bing pads the top and bottom edges with 8.5 pixels of white padding and the left and right edges with 13 pixels of white padding.

To prevent Bing from adding white padding if the requested size is greater than the thumbnail's original size, include the *p* query parameter. For example, if you include the *p* parameter in the above example, Bing returns a 474x316 image instead of a 500x333 image. Set the *p* parameter to 0 (zero).

curl

```
https://<host>/th?id=AMMS_92772df988...&w=500&p=0&pid=16.1
```

If you specify both the *w* and *h* query parameters, Bing maintains the thumbnail's aspect ratio and adds white padding as needed. For example, if the thumbnail's original size is 474x316 and you set the width and height parameters to 200x200 (&w=200&h=200), Bing returns an image that contains 33 pixels of white padding on the top and bottom. If you include the *p* query parameter, Bing returns an image that's 200x134.

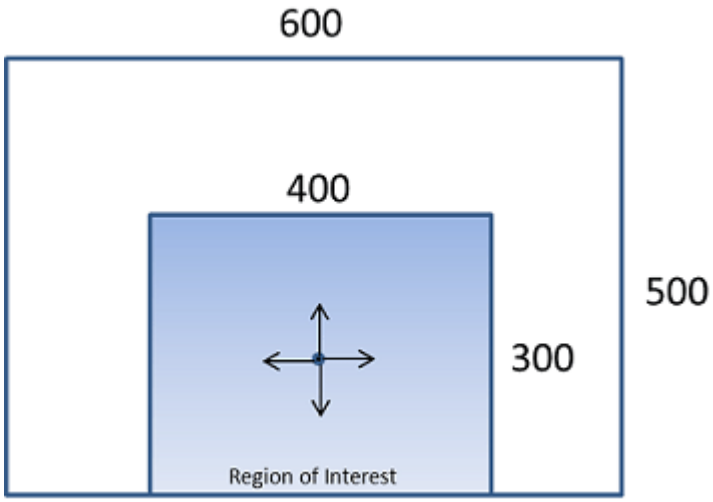
Cropping a thumbnail image

To crop an image, include the *c* (crop) query parameter. The following are the possible values that you may specify.

- `4` — Blind Ratio
- `7` — Smart Ratio

Requesting smart ratio cropping

If you request Smart Ratio cropping (c=7), Bing crops the image from the center of the image's region of interest outward, while maintaining the image's aspect ratio. The region of interest is the area of the image that Bing determines contains the most important part. The following shows an example region of interest.



If Bing cannot determine the image's region of interest, Bing uses Blind Ratio cropping.

Here's the original image used in the following examples.

Here's what it looks like if you resize the image to 200x200 using Smart Ratio cropping.

Here's what it looks like if you resize the image to 200x100 using Smart Ratio cropping.

Here's what it looks like if you resize the image to 100x200 using Smart Ratio cropping.

Requesting blind ratio cropping

If you request Blind Ratio cropping (c=4), Bing uses the following rules to crop the image.

- If (Original Image Width / Original Image Height) < (Requested Image Width / Requested Image Height), Bing measures the image from its top left corner and crops it at the bottom.
- If (Original Image Width / Original Image Height) > (Requested Image Width / Requested Image Height), Bing measures the image from the center and crops it to the left and right.

Here's the original image used in the following examples.

Here's what it looks like if you resize the image to 200x200 using Blind Ratio cropping.

Here's what it looks like if you resize the image to 200x100 using Blind Ratio cropping.

Here's what it looks like if you resize the image to 100x200 using Blind Ratio cropping.

Throttling requests

Article • 10/28/2020

The service and your subscription type determine the number of queries per second (QPS) that you can make. Make sure your application includes the logic necessary to stay within your quota. If the QPS limit is met or exceeded, the request fails and returns an HTTP 429 status code. The response includes the `Retry-After` header, which indicates how long you must wait before sending another request.

Denial-of-service versus throttling

Bing distinguishes between a denial-of-service (DoS) attack and a QPS violation. If Bing suspects a DoS attack, the request succeeds (HTTP status code is 200 OK) but the body of the response is empty.

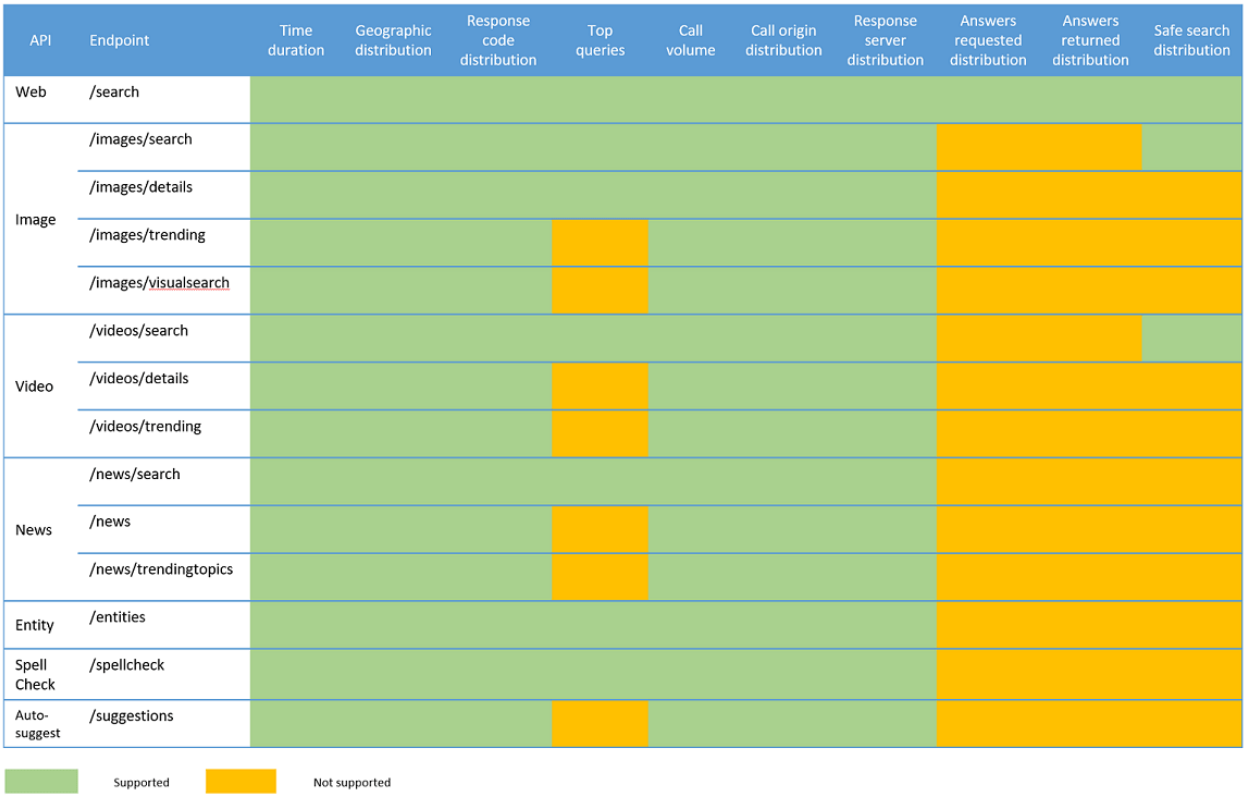
Add analytics to the Bing Search APIs

Article • 02/16/2022

Bing Statistics provides analytics for the Bing Search APIs. These analytics include call volume, top query strings, geographic distribution, and more. You can enable Bing Statistics in the Azure portal by navigating to your Azure resource and clicking **Enable Bing Statistics**.

> **Important**
>
> - Bing Statistics is not available with resources on the free `F0` pricing tier.
> - You may not use any data available via the Bing Statistics dashboard to create applications for distribution to third parties.
> - Enabling Bing Statistics increases your subscription rate slightly. See **pricing** for details.

The following image shows the available analytics for each Bing Search API endpoint.



Access your analytics

Bing updates analytics data every 24 hours and maintains up to 13 months' worth of history that you can access from the analytics dashboard. Make sure you're signed in

using the same Microsoft account (MSA) you used to sign up for Bing Statistics.

> **Note**
>
> - It may take up to 24 hours for metrics to surface on the dashboard. The dashboard shows the date and time the data was last updated.
> - Metrics are available from the time you enable the Bing Statistics Add-in.

Filter the data

By default, the charts and graphs display all metrics and data that you have access to. You can filter the data shown in the charts and graphs by selecting the resources, markets, endpoints, and reporting period you're interested in. You can change the following filters:

- **Resource ID**: The unique resource ID that identifies your Azure subscription. The list contains multiple IDs if you subscribe to more than one Bing Search API tier. By default, all resources are selected.
- **Markets**: The markets where the results come from. For example, en-us (English, United States). By default, all markets are selected. The `en-WW` market is the market that Bing uses if the call does not specify a market and Bing is unable to determine the user's market.
- **Endpoints**: The Bing Search API endpoints. The list contains all endpoints for which you have a paid subscription. By default, all endpoints are selected.
- **Time Frame**: The reporting period. You can specify:
  - **All**: Includes up to 13 months' worth of data.
  - **Past 24 hours**: Includes analytics from the last 24 hours.
  - **Past week**: Includes analytics from the previous 7 days.
  - **Past month**: Includes analytics from the previous 30 days.
  - **A custom date range**: Includes analytics from the specified date range, if available.

Charts and graphs

The dashboard shows charts and graphs of the metrics available for the selected endpoint. Not all metrics are available for all endpoints. The charts and graphs for each

endpoint are static (you may not select which charts and graphs to display). The dashboard shows only charts and graphs for which there's data.

The following are the metrics that the dashboard may include.

- **Call Volume**: Shows the number of calls made during the reporting period. If the reporting period is for a day, the chart shows the number of calls made per hour. Otherwise, the chart shows the number of calls made per day of the reporting period.

  > ⓘ **Note**
  >
  > The call volume may differ from billing reports, which generally includes only successful calls.

- **Top Queries**: Shows the top queries and the number of occurrences of each query during the reporting period. You can configure the number of queries shown. For example, you can show the top 25, 50, or 75 queries. Top Queries is not available for the following endpoints:
  - /images/trending
  - /images/details
  - /images/visualsearch
  - /videos/trending
  - /videos/details
  - /news
  - /news/trendingtopics
  - /suggestions

  > ⓘ **Note**
  >
  > Some query terms may be suppressed to remove confidential information such as emails, telephone numbers, SSN, etc.

- **Geographic Distribution**: The markets where search results originate. For example, `en-us` (English, United States). Bing uses the `mkt` query parameter to determine the market, if specified. Otherwise, Bing uses signals such as the caller's IP address to determine the market.

- **Response Code Distribution**: The HTTP status codes of all calls during the reporting period.

- **Call Origin Distribution**: The types of browsers used by the users. For example, Microsoft Edge, Chrome, Safari, and FireFox. Calls made from outside a browser (such as bots, Postman, or using curl from a console app) are grouped under Libraries. The origin is determined using the request's User-Agent header value. If the request doesn't include the User-Agent header, Bing tries to derive the origin from other signals.
- **Safe Search Distribution**: The distribution of safe search values. For example, off, moderate, or strict. The `safeSearch` query parameter contains the value, if specified. Otherwise, Bing defaults the value to moderate.
- **Answers Requested Distribution**: The Web Search API answers that you requested in the `responseFilter` query parameter.
- **Answers Returned Distribution**: The answers that Web Search API returned in the response.
- **Response Server Distribution**: The application server that served your API requests. The possible values are Bing.com (for traffic served from desktop and laptop devices) and Bing.com-mobile (for traffic served from mobile devices). The server is determined using the request's User-Agent header value. If the request doesn't include the User-Agent header, Bing tries to derive the server from other signals.

Upgrade from Bing Web Search API v5 to v7

Article • 02/16/2022

This upgrade guide identifies the changes between version 5 and version 7 of the Bing Web Search API. Use this guide to help you identify the parts of your application that you need to update to use version 7.

Breaking changes

Endpoints

- Changed the `cognitive` subdomain to `bing`.
- Changed the endpoint's version number from v5 to v7.
- Removed the `/bing` folder.

New search endpoint: `https://api.bing.microsoft.com/v7.0/search`

Error response objects and error codes

- All failed requests should now include an `ErrorResponse` object in the response body.
- Added the following fields to the `Error` object.
  - `subCode`—Partitions the error code into discrete buckets, if possible.
  - `moreDetails`—Additional information about the error described in the `message` field.
- Replaced the v5 error codes with the following possible `code` and `subCode` values.

Code	SubCode	Description
ServerError	UnexpectedError ResourceError NotImplemented	Bing returns ServerError whenever any of the subcode conditions occur. The response will include these errors if the HTTP status code is 500.

Code	SubCode	Description
InvalidRequest	ParameterMissing ParameterInvalidValue HttpNotAllowed Blocked	Bing returns InvalidRequest whenever any part of the request is not valid. For example, a required parameter is missing or a parameter value is not valid. If the error is ParameterMissing or ParameterInvalidValue, the HTTP status code is 400. If the error is HttpNotAllowed, the HTTP status code 410.
RateLimitExceeded		Bing returns RateLimitExceeded whenever you exceed your queries per second (QPS) or queries per month (QPM) quota. Bing returns HTTP status code 429 if you exceeded QPS and 403 if you exceeded QPM.
InvalidAuthorization	AuthorizationMissing AuthorizationRedundancy	Bing returns InvalidAuthorization when Bing cannot authenticate the caller. For example, the `Ocp-Apim-Subscription-Key` header is missing or the subscription key is not valid. Redundancy occurs if you specify more than one authentication method. If the error is InvalidAuthorization, the HTTP status code is 401.
InsufficientAuthorization	AuthorizationDisabled AuthorizationExpired	Bing returns InsufficientAuthorization when the caller does not have permissions to access the resource. This error can occur if the subscription key has been disabled or has expired. If the error is InsufficientAuthorization, the HTTP status code is 403.

- The following maps the previous error codes to the new codes. If you've taken a dependency on v5 error codes, update your code accordingly.

Version 5 code	Version 7 code.subCode
RequestParameterMissing	InvalidRequest.ParameterMissing

Version 5 code	Version 7 code.subCode
RequestParameterInvalidValue	InvalidRequest.ParameterInvalidValue
ResourceAccessDenied	InsufficientAuthorization
ExceededVolume	RateLimitExceeded
ExceededQpsLimit	RateLimitExceeded
Disabled	InsufficientAuthorization.AuthorizationDisabled
UnexpectedError	ServerError.UnexpectedError
DataSourceErrors	ServerError.ResourceError
AuthorizationMissing	InvalidAuthorization.AuthorizationMissing
HttpNotAllowed	InvalidRequest.HttpNotAllowed
UserAgentMissing	InvalidRequest.ParameterMissing
NotImplemented	ServerError.NotImplemented
InvalidAuthorization	InvalidAuthorization
InvalidAuthorizationMethod	InvalidAuthorization
MultipleAuthorizationMethod	InvalidAuthorization.AuthorizationRedundancy
ExpiredAuthorizationToken	InsufficientAuthorization.AuthorizationExpired
InsufficientScope	InsufficientAuthorization
Blocked	InvalidRequest.Blocked

Non-breaking changes

Headers

- Added the optional Pragma request header. By default, Bing returns cached content, if available. To prevent Bing from returning cached content, set the Pragma header to no-cache (for example, Pragma: no-cache).

Query parameters

- Added the answerCount query parameter. Use this parameter to specify the number of answers that you want the response to include. The answers are chosen

based on ranking. For example, if you set this parameter to three (3), the response includes the top three ranked answers.

- Added the promote query parameter. Use this parameter along with `answerCount` to explicitly include one or more answer types, regardless of their ranking. For example, to promote videos and images into the response, you'd set promote to *videos, images*. The list of answers that you want to promote does not count against the `answerCount` limit. For example, if `answerCount` is 2 and `promote` is set to *videos, images*, the response might include webpages, news, videos, and images.

Object changes

- Added the `someResultsRemoved` field to the WebAnswer object. The field contains a Boolean value that indicates whether the response excluded some results from the web answer.

Web Search API v7 reference

Article • 10/28/2020

The Web Search API lets you send a search query to Bing and get back search results that include links to webpages, images, and more. This section provides technical details about the query parameters and headers that you use to request web search results and the JSON response objects that contain them. For examples that show how to make requests, see Bing Web Search overview.

For information about what you can do with the search results, see Use and display requirements.

> **Note**
>
> Because URL formats and parameters are subject to change without notice, use all URLs as-is. You should not take dependencies on the URL format or parameters except where noted.

Endpoints

To request web search results, send a GET request to:

```
https://api.bing.microsoft.com/v7.0/search
```

The request must use the HTTPS protocol.

> **Note**
>
> The maximum URL length is 2,048 characters. To ensure that your URL length does not exceed the limit, the maximum length of your query parameters should be less than 1,500 characters. If the URL exceeds 2,048 characters, the server returns 404 Not found.

Next steps

Check out the following programming elements you'll use when sending search requests and processing the responses:

- Headers
- Query parameters
- Response objects

Web Search API v7 response objects

Article • 04/24/2024

For a list of possible objects, see **In this article** in the right pane.

If the request succeeds, the top-level object in the response is the SearchResponse object. And if the request fails, the top-level object in the response is the ErrorResponse object.

The JSON objects in this section are specific to the web answer. For details about the JSON objects for other answer types that the search results may include, see the API-specific reference documentation. For example, if the search results contain the images and news answers, see the Image Search API reference and News Search API reference.

> **Note**
>
> Because URL formats and parameters are subject to change without notice, use all URLs as-is. You should not take dependencies on the URL format or parameters except where noted.

Attribution

Defines the publisher that the content is attributed to.

Expand table

Name	Value	Type
providerDisplayName	The publisher's name that you use to attribute the content to.	String
seeMoreUrl	The URL to the publisher's website. Use `providerDisplayName` and this URL to create a hyperlink that you display in the UX following the translation.	String

Computation

Defines an expression and its answer.

Expand table

Name	Value	Type
expression	The math or conversion expression. If the query contains a request to convert units of measure (for example, meters to feet), this field contains the *from* units and `value` contains the *to* units. If the query contains a mathematical expression such as 2+2, this field contains the expression and `value` contains the answer. Note that mathematical expressions may be normalized. For example, if the query was sqrt(4^2+8^2), the normalized expression may be sqrt((4^2)+(8^2)). If the user's query is a math question and the textDecorations query parameter is set to **true**, the expression string may include formatting markers. For example, if the user's query is *log(2)*, the normalized expression includes the subscript markers. For more information, see Hit highlighting.	String
value	The expression's answer.	String

Error

Defines the error that occurred.

Expand table

Name	Value	Type
code	The error code that identifies the category of error. For a list of possible codes, see Error codes.	String
message	A description of the error.	String
moreDetails	A description that provides additional information about the error.	String
parameter	The query parameter in the request that caused the error.	String
subCode	The error code that identifies the error. For example, if `code` is InvalidRequest, `subCode` may be ParameterInvalid or ParameterInvalidValue.	String
value	The query parameter's value that was not valid.	String

ErrorResponse

The top-level object that the response includes when the request fails.

Expand table

Name	Value	Type
_type	Type hint, which is set to ErrorResponse.	String
errors	A list of errors that describe the reasons why the request failed.	Error[]

Identifiable

Defines the identity of a resource.

Expand table

Name	Value	Type
id	An identifier.	String

Image

Defines an image.

> **Note**
>
> Because URL formats and parameters are subject to change without notice, all image URLs should be used as-is; you should not take dependencies on the URL format or parameters. The exception is those parameters and values discussed by Resize and crop thumbnail images.

Expand table

Name	Value	Type
height	The height of the source image, in pixels.	Unsigned Short
hostPageUrl	The URL of the webpage that includes the image. This URL and `contentUrl` may be the same URL.	String

Name	Value	Type
name	An optional text string that contains random information about the image.	String
provider	The source of the image. The array will contain a single item. You must attribute the image to the provider. For example, you may display the provider's name as the cursor hovers over the image or make the image a click-through link to the provider's website where the image is found. If the image object is part of a larger object, and the larger object contains a contractual rule that applies to this object, you must use the contractual rule for attribution instead of this field.	Organization[]
thumbnailUrl	The URL to a thumbnail of the image. For information about resizing the image, see Resize and crop thumbnail images.	String
width	The width of the source image, in pixels.	Unsigned Short

License

Defines the license under which you may use the content.

Expand table

Name	Value	Type
name	The name of the license.	String
url	A URL to the website that describes the license. Use `name` and `url` to create a hyperlink.	String

LicenseAttribution

Defines a contractual rule for license attribution.

Expand table

Name	Value	Type
_type	A type hint, which is set to LicenseAttribution.	String
license	The license under which the content may be used.	License

Name	Value	Type
licenseNotice	The license to display next to the targeted field. For example, "Text under CC-BY-SA license". Use the license's name and URL in the `license` field to create a hyperlink to the website that describes the details of the license. Then, replace the license name in the `licenseNotice` string (for example, CC-BY-SA) with the hyperlink you just created.	String
mustBeCloseToContent	A Boolean value that determines whether the contents of the rule must be placed in close proximity to the field that the rule applies to. If **true**, the contents must be placed in close proximity. If **false**, or this field does not exist, the contents may be placed at the caller's discretion.	Boolean
targetPropertyName	The name of the field that the rule applies to.	String

LinkAttribution

Defines a contractual rule for link attribution.

Expand table

Name	Value	Type
_type	A type hint, which is set to LinkAttribution.	String
mustBeCloseToContent	A Boolean value that determines whether the contents of the rule must be placed in close proximity to the field that the rule applies to. If **true**, the contents must be placed in close proximity. If **false**, or this field does not exist, the contents may be placed at the caller's discretion.	Boolean
targetPropertyName	The name of the field that the rule applies to. If a target is not specified, the attribution applies to the entity as a whole and should be displayed immediately following the entity presentation. If there are multiple text and link attribution rules that do not specify a target, you should concatenate them and display them using a "Data from: " label. For example, “Data from <provider name1> | <provider name2>".	String
text	The attribution text.	String

Name	Value	Type
url	The URL to the provider's website. Use `text` and URL to create of hyperlink.	String

Malware

Defines a notice that the webpage may cause a potential issue if the user clicks the `url` link.

Expand table

Name	Value	Type
beSafeRxUrl	A URL to a webpage where the user may get more information about safely buying prescription medicine online.	String
malwareWarningType	The type of malware notice. Possible values are: • NABP — Warns that the National Association of Boards of Pharmacy includes this pharmacy on its Not Recommended list. • Malware — Warns that the site may download malicious software that may harm the user's device. • MaliciousPageLink — Warns that the site may contain links that could download malicious software that may harm the user's device. • Phishing — Warns that the site could trick the user into disclosing financial, personal, or other sensitive information.	String
warningExplanationUrl	A URL to a webpage where the user can get an explanation of the issue. For NABP notices, users can use this link to verify a pharmacy.	String
warningLetterUrl	A URL to a webpage where the user can get more information about the notice. For NABP notices, users can use this link to see the list of online sites that the board doesn't recommend.	String

MediaAttribution

Defines a contractual rule for media attribution.

Expand table

Name	Value	Type
_type	A type hint, which is set to MediaAttribution.	String
mustBeCloseToContent	A Boolean value that determines whether the contents of the rule must be placed in close proximity to the field that the rule applies to. If **true**, the contents must be placed in close proximity. If **false**, or this field does not exist, the contents may be placed at the caller's discretion.	Boolean
targetPropertyName	The name of the field that the rule applies to.	String
url	The URL that you use to create of hyperlink of the media content. For example, if the target is an image, you would use the URL to make the image clickable.	String

MetaTag

Defines a webpage's metadata.

Expand table

Name	Value	Type
content	The metadata.	String
name	The name of the metadata.	String

Organization

Defines a publisher.

Note that a publisher may provide their name or their website or both.

Expand table

Name	Value	Type
name	The publisher's name.	String
url	The URL to the publisher's website. Note that the publisher may not provide a website.	String

Query

Defines a search query.

The SpellSuggestions object uses this object to suggest a query string that likely represents the user's intent. It's also used by RelatedSearchAnswer to return a related query that other users have made.

Expand table

Name	Value	Type
displayText	The display version of the query term. This version of the query term may contain special characters that highlight the search term found in the query string. The string contains the highlighting characters only if the query enabled hit highlighting (see the textDecorations query parameter). For details about hit highlighting, see Hit highlighting.	String
text	The query string. Use this string as the query term in a new search request.	String
webSearchUrl	The URL that takes the user to the Bing search results page for the query. Only related search results include this field.	String

QueryContext

Defines the query string that Bing used for the request.

Expand table

Name	Value	Type
adultIntent	A Boolean value that indicates whether the specified query has adult intent. The value is **true** if the query has adult intent. If **true**, and the request's safeSearch query parameter is set to Strict, the response contains only news results, if applicable.	Boolean
alterationOverrideQuery	The query string to use to force Bing to use the original string. For example, if the query string is *saling downwind*, the override query string is *+saling downwind*. Remember to encode the query string, which results in *%2Bsaling+downwind*. The object includes this field only if the original query string contains a spelling mistake.	String

Name	Value	Type
alteredQuery	The query string that Bing used to perform the query. Bing uses the altered query string if the original query string contained spelling mistakes. For example, if the query string is *saling downwind*, the altered query string is *sailing downwind*. The object includes this field only if the original query string contains a spelling mistake.	String
askUserForLocation	A Boolean value that indicates whether Bing requires the user's location to provide accurate results. If you specified the user's location by using the X-MSEdge-ClientIP and X-Search-Location headers, you can ignore this field. For location aware queries, such as "today's weather" or "restaurants near me" that need the user's location to provide accurate results, this field is set to **true**. For location aware queries that include the location (for example, "Seattle weather"), this field is set to **false**. This field is also set to **false** for queries that are not location aware, such as "best sellers."	Boolean
originalQuery	The query string as specified in the request.	String

RankingGroup

Defines a search results group, such as mainline.

Expand table

Name	Value	Type
items	A list of search result items to display in the group.	RankingItem[]

RankingItem

Defines a search result item to display. For more information about how to use the IDs, see Ranking results.

Expand table

Name	Value	Type
answerType	The answer that contains the item to display. For example, News. Use the type to find the answer in the SearchResponse object. The type is the name of a field in the `SearchResponse` object.	String
resultIndex	A zero-based index of the item in the answer. If the item does not include this field, display all items in the answer. For example, display all news articles in the News answer.	Integer
value	The ID that identifies either an answer to display or an item of an answer to display. If the ID identifies an answer, display all items of the answer.	Identifiable

RankingResponse

Defines where on the search results page content should be placed and in what order.

Expand table

Name	Value	Type
mainline	The search results to display in the mainline section of the search results page.	RankingGroup
pole	The search results that should be afforded the most visible treatment (for example, displayed above the mainline and sidebar).	RankingGroup
sidebar	The search results to display in the sidebar section of the search results page.	RankingGroup

RelatedSearchAnswer

Defines a list of related queries made by others.

Expand table

Name	Value	Type
id	An ID that uniquely identifies the related search answer. The object includes this field only if the Ranking answer specifies that you should display all related searches in a group. For more information about how to use the ID, see Ranking results.	String

Name	Value	Type
value	A list of related queries that were made by others.	Query[]

SearchResponse

The response's top-level object for search requests that succeed.

By default, the Search API includes all relevant answers unless:

- The query specifies the responseFilter query parameter to limit the answers it returns.
- One or more of the search components does not return results (for example, no news results are relevant to the query).
- The subscription key does not have access to the search component.

If the service suspects a denial of service attack, the request succeeds (HTTP status code is 200 OK), but the body of the response is empty.

Expand table

Name	Value	Type
_type	Type hint, which is set to SearchResponse.	String
computation	The answer to a math expression or unit conversion expression.	Computation
entities	A list of entities that are relevant to the search query.	EntityAnswer
images	A list of images that are relevant to the search query.	ImageAnswer
news	A list of news articles that are relevant to the search query.	NewsAnswer
places	A list of places that are relevant to the search query.	LocalEntityAnswer
queryContext	The query string that Bing used for the request.	QueryContext
rankingResponse	The order that Bing suggests that you display the search results in.	RankingResponse
relatedSearches	A list of related queries made by others.	RelatedSearchAnswer
spellSuggestions	The query string that likely represents the user's intent.	SpellSuggestions

Name	Value	Type
timeZone	The date and time of one or more geographic locations.	TimeZone
translations	The translation of a word or phrase in the query string to another language.	TranslationAnswer
videos	A list of videos that are relevant to the search query.	VideosAnswer
webPages	A list of webpages that are relevant to the search query.	WebAnswer

SpellSuggestions

Defines a suggested query string that likely represents the user's intent.

The search results include this response if Bing determines that the user may have intended to search for something different. For example, if the user searches for *alon brown*, Bing may determine that the user likely intended to search for *Alton Brown* instead (based on past searches by others of *alon brown*).

Expand table

Name	Value	Type
id	An ID that uniquely identifies the spelling suggestion answer. You use this field when you use ranking response to display the spelling suggestions. For more information about how to use the ID, see Ranking results.	String
value	A list of suggested query strings that may represent the user's intention. The list contains only one `Query` object.	Query[]

TextAttribution

Defines a contractual rule for plain text attribution.

Expand table

Name	Value	Type
_type	A type hint, which is set to TextAttribution.	String

Name	Value	Type
text	The attribution text. Text attribution applies to the entity as a whole and should be displayed immediately following the entity presentation. If there are multiple text or link attribution rules that do not specify a target, you should concatenate them and display them using a "Data from: " label.	String

TimeZone

Defines the date and time of one or more geographic locations.

Expand table

Name	Value	Type
date	A date in string form. For example, Thursday, June 5, 2019. The answer may include this field if the user's query asks Bing to compute a date. For example, *90 days from today*.	String
description	A description of the response. The answer may include this field if the query asks Bing how many days or weeks in a period (for example, *weeks in a year* or *days in a month*), or converts time to a different time zone (for example, *pst to est*).	String
otherCityTimes	A list of dates and times in a geographical location. The answer includes this field for queries like *US time zones* or *Arizona time zones*. The list is ordered by UTC offset.	TimeZoneInformation[]
primaryCityTime	The data and time, in UTC, of the geographic location specified in the query. If the query specified a specific geographic location (for example, a city), this object contains the name of the geographic location and the current date and time of the location. If the query specified a general geographic location, such as a state or country/region, this object contains the date and time of the primary city or state found within the specified state or country/region. If the location contains additional time zones, the `otherCityTimes` field contains the	TimeZoneInformation

Name	Value	Type
	date and time of cities or states located in the other time zones.	
primaryResponse	The primary data that satisfies the request. If the query string is *how many weeks in 2019*, this field contains, **52 weeks and 1 day**. Other query examples: *how many days in this month* and *what's the date*.	String
primaryTimeZone	The object contains the primary time zone in a geographical location. If a location contains more than one time zone, Bing determines which time zone is the primary time zone. The answer includes this field for queries like, time zone, time zones in Arizona, us time zones.	TimeZoneInformation
timeZoneDifference	The difference in time, in hours, between time zones. For example, there's a three hour difference between PST and EST.	TimeZoneDifference

TimeZoneDifference

Defines the difference in time, in hours, between time zone 1 and time zone 2.

Expand table

Name	Value	Type
location1	the date and time of the first time zone. For example, if the query is, *pst to est*, this field contains the date and time of the Pacific time zone.	TimeZoneInformation
location2	the date and time of the second time zone. For example, if the query is, *pst to est*, this field contains the date and time of the Eastern time zone.	TimeZoneInformation
text	A string that represents the difference in time between time zones.	String

TimeZoneInformation

Defines a date and time for a geographical location.

Expand table

Name	Value	Type
location	The type of the geographical location. For example, County; City; City, State; City, State, Country/Region; or Time Zone.	String
time	The UTC date and time specified in the form, YYYY-MM-DDThh;mm:ss.ssssssZ.	String
timeZoneName	The name of the time zone that the location is in. This string may be empty if the query is not related to time zones.	String
utcOffset	The offset from UTC. For example, UTC-7.	String

TranslationAnswer

Defines the translation of a word or phrase in the query string to another language.

Expand table

Name	Value	Type
attributions	A list of publishers that you must attribute the information to when you render the answer. You must display the names of all publishers in the list as the source of the data. Typically, you display the providers in a single line after the translation. For example, "Data from: <provider> | <provider> | ...", where <provider> is the name of the provider in providerDisplayName. **Note**: If the answer includes contractualRules, you must apply them instead of applying attributions from this field.	Attribution[]
contractualRules	A list of rules that you must adhere to if you display the answer. The following contractual rules may apply. • LinkAttribution For information about displaying contractual rules, see Data Attribution.	Object[]
id	An ID that uniquely identifies this answer. The RankingResponse answer uses the ID to indicate where in the rendered response you should display	String

Name	Value	Type
	this answer. For information about how to use this field, see How to use ranking to display search results.	
inLanguage	The language that the text was translated from. An ISO 639-1 two-letter language code identifies the language. For example, es for Spanish.	String
originalText	The text to translate.	String
translatedLanguageName	The language that the text was translated to. An ISO 639-1 two-letter language code identifies the language. For example, en for English.	String
translatedText	The translated text.	String

WebAnswer

Defines a list of relevant webpage links.

Expand table

Name	Value	Type
id	An ID that uniquely identifies the web answer. The object includes this field only if the Ranking answer suggests that you display all web results in a group. For more information about how to use the ID, see Ranking results.	String
someResultsRemoved	A Boolean value that indicates whether the response excluded some results from the answer. If Bing excluded some results, the value is **true**.	Boolean
totalEstimatedMatches	The estimated number of webpages that are relevant to the query. Use this number along with the count and offset query parameters to page the results.	Long
value	A list of webpages that are relevant to the query.	WebPage[]
webSearchUrl	The URL to the Bing search results for the requested webpages.	String

Webpage

Defines a webpage that is relevant to the query.

Expand table

Name	Value	Type
about	For internal use only.	Object[]
dateLastCrawled	The last time that Bing crawled the webpage. The date is in the form, YYYY-MM-DDTHH:MM:SS. For example, 2015-04-13T05:23:39.	String
datePublished	The time that webpage published. The date is in the form, YYYY-MM-DDTHH:MM:SS. Example: 2015-04-13T05:23:39.	String
datePublishedDisplayText	The display version of the datePublished.	String
contractualRules	A list of rules that you must adhere to if you display the answer. The following contractual rules may apply. • LicenseAttribution For information about displaying contractual rules, see Data Attribution.	Object[]
deepLinks	A list of links to related content that Bing found in the website that contains this webpage. The `Webpage` object in this context includes only the `name` and `url` fields and optionally the `snippet` field.	Webpage[]
displayUrl	The display URL of the webpage. The URL is meant for display purposes only and is not well formed.	String
id	An ID that uniquely identifies this webpage in the list of web results. The object includes this field only if the Ranking answer specifies that you mix the webpages with the other search results. Each webpage contains an ID that matches an ID in the Ranking answer. For more information, see Ranking results.	String
isFamilyFriendly	A Boolean value that indicates whether the webpage contains adult content. If the webpage doesn't contain adult content, `isFamilyFriendly` is set to **true**.	Boolean
isNavigational	A Boolean value that indicates whether the user's query is frequently used for navigation to different parts of the	Boolean

Name	Value	Type
	webpage's domain. Is **true** if users navigate from this page to other parts of the website.	
language	A two-letter language code that identifies the language used by the webpage. For example, the language code is *en* for English.	String
malware	A notice that Bing provides if it thinks the webpage may cause a potential issue if the user clicks the `url` link. You should display the notice with high visibility next to the webpage link.	Malware
name	The name of the webpage. Use this name along with `url` to create a hyperlink that when clicked takes the user to the webpage.	String
mentions	For internal use only.	Object
searchTags	A list of search tags that the webpage owner specified on the webpage. The API returns only indexed search tags. The `name` field of the `MetaTag` object contains the indexed search tag. Search tags begin with search.* (for example, search.assetId). The `content` field contains the tag's value.	MetaTag[]
snippet	A snippet of text from the webpage that describes its contents.	String
url	The URL to the webpage. Use this URL along with `name` to create a hyperlink that when clicked takes the user to the webpage.	String

Web Search API v7 query parameters

Article • 04/07/2023

The following are the query parameters that requests may include. The Required column indicates whether you must specify the parameter. You must URL encode the query parameter values.

Name	Value	Type	Required
answerCount	The number of answers that you want the response to include. The answers that Bing returns are based on ranking. For example, if Bing returns webpages, images, videos, and relatedSearches for a request and you set this parameter to two (2), the response includes webpages and images. If you included the `responseFilter` query parameter in the same request and set it to webpages and news, the response would include only webpages. For information about promoting a ranked answer into the response, see the promote query parameter.	Unsigned Integer	No

Name	Value	Type	Required
cc	A 2-character country code of the country where the results come from. For a list of possible values, see Market codes. If you set this parameter, you must also specify the Accept-Language header. Bing uses the first supported language it finds in the specified languages and combines it with the country code to determine the market to return results for. If the languages list does not include a supported language, Bing finds the closest language and market that supports the request. Or, Bing may use an aggregated or default market for the results. To know which market Bing used, get the BingAPIs-Market header in the response. Use this query parameter and the `Accept-Language` header only if you specify multiple languages. Otherwise, you should use the `mkt` and `setLang` query parameters. This parameter and the mkt query parameter are mutually exclusive — do not specify both.	String	No
count	The number of search results to return in the response. The default is 10 and the maximum value is 50. The actual number delivered may be less than requested. Use this parameter along with the `offset` parameter to page results. For more information, see Paging results. For example, if your user interface displays 10 search results per page, set `count` to 10 and `offset` to 0 to get the first page of results. For each subsequent page, increment `offset` by 10 (for example, 0, 10, 20). It is possible for multiple pages to include some overlap in results. This parameter affects only webpage results and has no impact on the number of results that Bing returns for other answers in the search results such as images or videos.	UnsignedShort	No

Name	Value	Type	Required
freshness	Filter search results by the following case-insensitive age values: • Day — Return webpages that Bing discovered within the last 24 hours. • Week — Return webpages that Bing discovered within the last 7 days. • Month — Return webpages that Bing discovered within the last 30 days. To get articles discovered by Bing during a specific timeframe, specify a date range in the form, YYYY-MM-DD..YYYY-MM-DD. For example, `&freshness=2019-02-01..2019-05-30`. To limit the results to a single date, set this parameter to a specific date. For example, `&freshness=2019-02-04`.	String	No
mkt	The market where the results come from. Typically, `mkt` is the country where the user is making the request from. However, it could be a different country if the user is not located in a country where Bing delivers results. The market must be in the form <language>-<country/region>. For example, en-US. The string is case insensitive. For a list of possible market values, see Market codes. **NOTE:** If known, you are encouraged to always specify the market. Specifying the market helps Bing route the request and return an appropriate and optimal response. If you specify a market that is not listed in Market codes, Bing uses a best fit market code based on an internal mapping that is subject to change. To know which market Bing used, get the BingAPIs-Market header in the response. This parameter and the cc query parameter are mutually exclusive — do not specify both.	String	No

Name	Value	Type	Required
offset	The zero-based offset that indicates the number of search results to skip before returning results. The default is 0. The offset should be less than (totalEstimatedMatches - `count`). Use this parameter along with the `count` parameter to page results. For example, if your user interface displays 10 search results per page, set `count` to 10 and `offset` to 0 to get the first page of results. For each subsequent page, increment `offset` by 10 (for example, 0, 10, 20). It is possible for multiple pages to include some overlap in results.	Unsigned Short	No

Name	Value	Type	Required
promote	A comma-delimited list of answers that you want the response to include regardless of their ranking. For example, if you set answerCount) to two (2) so Bing returns the top two ranked answers, but you also want the response to include news, you'd set `promote` to news. If the top ranked answers are webpages, images, videos, and relatedSearches, the response includes webpages and images because news is not a ranked answer. But if you set `promote` to video, Bing would promote the video answer into the response and return webpages, images, and videos. The answers that you want to promote do not count against the `answerCount` limit. For example, if the ranked answers are news, images, and videos, and you set `answerCount` to 1 and `promote` to news, the response contains news and images. Or, if the ranked answers are videos, images, and news, the response contains videos and news. The following are the possible values: • Computation • Entities • Images • News • RelatedSearches • SpellSuggestions • TimeZone • Videos • Webpages **NOTE:** Use only if you specify the answerCount parameter.	String	No

Name	Value	Type	Required
q	The user's search query term. The term may not be empty. The term may contain Bing Advanced Operators. For example, to limit results to a specific domain, use the **site:** operator (q=fishing+site:fishing.contoso.com). Note that the results may contain results from other sites depending on the number of relevant results found on the specified site.	String	Yes

Name	Value	Type	Required
responseFilter	A comma-delimited list of answers to include in the response. If you do not specify this parameter, the response includes all search answers for which there's relevant data. The following are the possible filter values: • Computation • Entities • Images • News • Places • RelatedSearches • SpellSuggestions • TimeZone • Translations • Videos • Webpages If you want to exclude specific types of content, such as images, from the response, you can exclude them by prefixing a hyphen (minus) to the `responseFilter` value. For example, `&responseFilter=-images`. Although you may use this filter to get a single answer, you should instead use the answer-specific endpoint in order to get richer results. For example, to receive only images, send the request to one of the Image Search API endpoints. The RelatedSearches and SpellSuggestions answers do not support a separate endpoint like the Image Search API does (only the Web Search API returns them). To include answers that would otherwise be excluded because of ranking, see the promote query parameter.	String	No

Name	Value	Type	Required
safeSearch	Used to filter webpages, images, and videos for adult content. The following are the possible filter values: • Off — Returns content with adult text and images but not adult videos. • Moderate — Returns webpages with adult text, but not adult images or videos. • Strict — Does not return adult text, images, or videos. The default is Moderate. **NOTE:** For video results, if `safeSearch` is set to Off, Bing ignores it and uses Moderate. **NOTE:** If the request comes from a market that Bing's adult policy requires that `safeSearch` be set to Strict, Bing ignores the `safeSearch` value and uses Strict. **NOTE:** If you use the `site:` query operator, there is a chance that the response may contain adult content regardless of what the `safeSearch` query parameter is set to. Use `site:` only if you are aware of the content on the site and your scenario supports the possibility of adult content.	String	No

Name	Value	Type	Required
setLang	The language to use for user interface strings. You may specify the language using either a 2-letter or 4-letter code. Using 4-letter codes is preferred. For a list of supported language codes, see Bing supported languages. Bing loads the localized strings if `setlang` contains a valid 2-letter neutral culture code (**fr**) or a valid 4-letter specific culture code (**fr-ca**). For example, for **fr-ca**, Bing loads the **fr** neutral culture code strings. If `setlang` is not valid (for example, **zh**) or Bing doesn't support the language (for example, **af**, **af-na**), Bing defaults to **en** (English). To specify the 2-letter code, set this parameter to an ISO 639-1 language code. To specify the 4-letter code, use the form <language>-<country/region> where <language> is an ISO 639-1 language code (neutral culture) and <country/region> is an ISO 3166 country/region (specific culture) code. For example, use **en-US** for United States English. Although optional, you should always specify the language. Typically, you set `setLang` to the same language specified by `mkt` unless the user wants the user interface strings displayed in a different language. This parameter and the Accept-Language header are mutually exclusive — do not specify both. A user interface string is a string that's used as a label in a user interface. There are few user interface strings in the JSON response objects. Also, any links to Bing.com properties in the response objects use the specified language.	String	No

Name	Value	Type	Required
textDecorations	A Boolean value that determines whether display strings in the results should contain decoration markers such as hit highlighting characters. If **true**, the strings may include markers. The default is **false**. To specify whether to use Unicode characters or HTML tags as the markers, see the textFormat query parameter. For information about hit highlighting, see Hit highlighting.	Boolean	No
textFormat	The type of markers to use for text decorations (see the `textDecorations` query parameter). The following are the possible values: • Raw — Use Unicode characters to mark content that needs special formatting. The Unicode characters are in the range E000 through E019. For example, Bing uses E000 and E001 to mark the beginning and end of query terms for hit highlighting. • HTML — Use HTML tags to mark content that needs special formatting. For example, use <b> tags to highlight query terms in display strings. The default is Raw. For a list of markers and information about processing strings with the embedded Unicode characters, see Hit highlighting. For display strings that contain escapable HTML characters such as <, >, and &, if `textFormat` is set to HTML, Bing escapes the characters as appropriate (for example, < is escaped to <).	String	No

Web Search API v7 headers

Article • 02/21/2024

The following are the headers that a request and response may include.

> **Note**
>
> Remember that the Terms of Use require compliance with all applicable laws, including regarding use of these headers. For example, in certain jurisdictions, such as Europe, there are requirements to obtain user consent before placing certain tracking devices on user devices.

Request headers

Expand table

Header	Required	Description
Accept	No	The default media type is application/json. To specify that the response use JSON-LD, set the Accept header to application/ld+json.
Accept-Language	No	A comma-delimited list of languages to use for user interface strings. The list is in decreasing order of preference. For more information, including expected format, see RFC2616. This header and the setLang query parameter are mutually exclusive — do not specify both. If you set this header, you must also specify the cc query parameter. To determine the market to return results for, Bing uses the first supported language it finds from the list and combines it with the `cc` parameter value. If the list does not include a supported language, Bing finds the closest language and market that supports the request or it uses an aggregated or default market for the results. To determine the market that Bing used, see the BingAPIs-Market header. Use this header and the `cc` query parameter only if you specify multiple languages. Otherwise, use the mkt and setLang query parameters. A user interface string is a string that's used as a label in a user interface. There are few user interface strings in the JSON response

Header	Required	Description
		objects. Any links to Bing.com properties in the response objects will apply the specified language.
Ocp-Apim-Subscription-Key	Yes	The subscription key that you received when you signed up for this service in Azure Portal while creating a Bing resource.
Pragma	No	By default, Bing returns cached content, if available. To prevent Bing from returning cached content, set the Pragma header to no-cache (for example, Pragma: no-cache).
User-Agent	No	The user agent originating the request. Bing uses the user agent to provide mobile users with an optimized experience. Although optional, you are encouraged to always specify this header. The user-agent should be the same string that any commonly used browser sends. For information about user agents, see RFC 2616. The following are examples of user-agent strings: • Android — Mozilla/5.0 (Linux; U; Android 2.3.5; en-us; SCH-I500 Build/GINGERBREAD) AppleWebKit/533.1 (KHTML; like Gecko) Version/4.0 Mobile Safari/533.1 • iPhone — Mozilla/5.0 (iPhone; CPU iPhone OS 6_1 like Mac OS X) AppleWebKit/536.26 (KHTML; like Gecko) Mobile/10B142 iPhone4;1 BingWeb/3.03.1428.20120423 • PC — Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; Touch; rv:11.0) like Gecko • iPad — Mozilla/5.0 (iPad; CPU OS 7_0 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) Version/7.0 Mobile/11A465 Safari/9537.53
X-MSEdge-ClientID	No	This header is used by both requests and responses. Bing uses this header to provide users with consistent behavior across Bing API calls. Bing often flights new features and improvements, and it uses the client ID as a key for assigning traffic on different flights. If you do not use the same client ID for a user across multiple requests, then Bing may assign the user to multiple conflicting flights. Being assigned to multiple conflicting flights can lead to an inconsistent user experience. For example, if the second request has a different flight assignment than the first, the experience may be unexpected. Also, Bing can use the client ID to tailor web results to that client ID's search history, providing a richer experience for the user.

Header	Required	Description
		Bing also uses this header to help improve result rankings by analyzing the activity generated by a client ID. The relevance improvements help with better quality of results delivered by Bing APIs and in turn enables higher click-through rates for the API consumer. **IMPORTANT:** Although optional, you should consider this header required. Persisting the client ID across multiple requests for the same end user and device combination enables: 1) the API consumer to receive a consistent user experience, and 2) higher click-through rates via better quality of results from the Bing APIs. The following are the basic usage rules that apply to this header: • Each user that uses your application on the device must have a unique, Bing-generated client ID. If you do not include this header in the request, Bing generates an ID and returns it in the X-MSEdge-ClientID response header. The only time that you should NOT include this header in a request is the first time the user uses your app on that device. • **ATTENTION:** You must ensure that this Client ID is not linkable to any authenticated user account information. • Use the client ID for each Bing API request that your app makes for this user on the device. • Persist the client ID. To persist the ID in a browser app, use a persistent HTTP cookie to ensure the ID is used across all sessions. Do not use a session cookie. For other apps such as mobile apps, use the device's persistent storage to persist the ID. The next time the user uses your app on that device, get the client ID that you persisted. **NOTE:** Bing responses may or may not include this header. If the response includes this header, capture the client ID and use it for all subsequent Bing requests for the user on that device. **NOTE:** If you include the X-MSEdge-ClientID, you must not include cookies in the request.

Header	Required	Description
X-MSEdge-ClientIP	No	The IPv4 or IPv6 address of the client device. The IP address is used to discover the user's location. Bing uses the location information to determine safe search behavior. **NOTE:** Although optional, you are encouraged to always specify this header and the X-Search-Location header. Do not obfuscate the address (for example, by changing the last octet to 0). Obfuscating the address results in the location not being anywhere near the device's actual location, which may result in Bing serving erroneous results.
X-Search-Location	No	A semicolon-delimited list of key/value pairs that describe the client's geographical location. Bing uses the location information to determine safe search behavior and to return relevant local content. Specify the key/value pair as <key>:<value>. The following are the keys that you use to specify the user's location. • lat — Required. The latitude of the client's location, in degrees. The latitude must be greater than or equal to -90.0 and less than or equal to +90.0. Negative values indicate southern latitudes and positive values indicate northern latitudes. • long — Required. The longitude of the client's location, in degrees. The longitude must be greater than or equal to -180.0 and less than or equal to +180.0. Negative values indicate western longitudes and positive values indicate eastern longitudes. • re — Required. The radius, in meters, which specifies the horizontal accuracy of the coordinates. Pass the value returned by the device's location service. Typical values might be 22m for GPS/Wi-Fi, 380m for cell tower triangulation, and 18,000m for reverse IP lookup. • ts — Optional. The UTC UNIX timestamp of when the client was at the location. (The UNIX timestamp is the number of seconds since January 1, 1970.) • head — Optional. The client's relative heading or direction of travel. Specify the direction of travel as degrees from 0 through 360, counting clockwise relative to true north. Specify this key only if the `sp` key is nonzero. • sp — Optional. The horizontal velocity (speed), in meters per second, that the client device is traveling.

Header	Required	Description
		• alt — Optional. The altitude of the client device, in meters. • are — Optional. The radius, in meters, that specifies the vertical accuracy of the coordinates. Specify this key only if you specify the `alt` key. • disp — Optional. The user's geographic location in the form, disp:<City, State>. For example, disp:Seattle, Washington. This is the display text version of the user's location that you specified using the lat/long keys. Remember to URL encode the location. For example, Seattle%2C%20Washington. **NOTE:** The order that you specify the parameters is import. For example, if you specify the disp parameter, the order must be: lat, long, re, and disp. **NOTE:** Bing ignores this header if the query string includes a location. For example, if this header is set to San Francisco, but the query is *restaurants seattle*, Bing returns restaurants located in Seattle, Washington. **NOTE:** Although many of the keys are optional, the more information that you provide, the more accurate the location results are. **NOTE:** Although optional, you are encouraged to always specify the user's geographical location. Providing the location is especially important if the client's IP address does not accurately reflect the user's physical location (for example, if the client uses VPN). For optimal results, you should include this header and the X-Search-ClientIP header, but at a minimum, you should include this header.

Response headers

Expand table

Header	Description
BingAPIs-Market	The market used by the request. The form is <languageCode>-<countryCode>. For example, en-US. This value may be different from the value you specify in the request's mkt query parameter if that market value is not listed in Market codes. The same is true if you specify values for cc and Accept-Language that can't be reconciled.
BingAPIs-TraceId	The ID of the log entry that contains the details of the request. When an error occurs, capture this ID. If you are not able to determine and resolve the issue, include this ID

Header	Description
	along with the other information that you provide the Support team.
Retry-After	The response includes this header if you exceed the number of queries allowed per second (QPS) or per month (QPM). The header contains the number of seconds that you must wait before sending another request.
X-MSEdge-ClientID	This header is used by both requests and responses. Bing uses this header to provide users with consistent behavior across Bing API calls. Bing often flights new features and improvements, and it uses the client ID as a key for assigning traffic on different flights. If you do not use the same client ID for a user across multiple requests, then Bing may assign the user to multiple conflicting flights. Being assigned to multiple conflicting flights can lead to an inconsistent user experience. For example, if the second request has a different flight assignment than the first, the experience may be unexpected. Also, Bing can use the client ID to tailor web results to that client ID's search history, providing a richer experience for the user. Bing also uses this header to help improve result rankings by analyzing the activity generated by a client ID. The relevance improvements help with better quality of results delivered by Bing APIs and in turn enables higher click-through rates for the API consumer. **IMPORTANT:** Although optional, you should consider this header required. Persisting the client ID across multiple requests for the same end user and device combination enables: 1) the API consumer to receive a consistent user experience, and 2) higher click-through rates via better quality of results from the Bing APIs. The following are the basic usage rules that apply to this header. • Each user that uses your application on the device must have a unique, Bing generated client ID. If you do not include this header in the request, Bing generates an ID and returns it in the X-MSEdge-ClientID response header. The only time that you should NOT include this header in a request is the first time the user uses your app on that device. • **ATTENTION:** You must ensure that this Client ID is not linkable to any authenticated user account information. • Use the client ID for each Bing API request that your app makes for this user on the device. • Persist the client ID. To persist the ID in a browser app, use a persistent HTTP cookie to ensure the ID is used across all sessions. Do not use a session cookie. For other apps such as mobile apps, use the device's persistent storage to persist the ID.

Header	Description
	The next time the user uses your app on that device, get the client ID that you persisted. **NOTE:** Bing responses may or may not include this header. If the response includes this header, capture the client ID and use it for all subsequent Bing requests for the user on that device. **NOTE:** If you include the X-MSEdge-ClientID, you must not include cookies in the request.

Accessing headers in JavaScript

If you're using JavaScript, your browser's built-in security features (CORS) might prevent you from accessing the response header values. To handle this case, host a server-side script on the same domain as the Web page that uses the Bing Web Search API. This script should make API calls upon request from the Web page JavaScript and pass all results, including headers, back to the client. Since the two resources (page and script) share an origin, the headers are accessible to the JavaScript on the Web page.

This approach also protects your API key from exposure to the public, since only the server-side script needs it. The script can use another method to make sure the request is authorized.

Market and language codes used by Bing Web Search API

Article • 01/29/2024

The following table lists the market code values that you set the mkt query parameter to. Bing returns content for only these markets. The list is subject to change.

For a list of country codes that you may set the cc query parameter to, see Country codes.

Expand table

Country/Region	Language	Market code
Argentina	Spanish	es-AR
Australia	English	en-AU
Austria	German	de-AT
Belgium	Dutch	nl-BE
Belgium	French	fr-BE
Brazil	Portuguese	pt-BR
Canada	English	en-CA
Canada	French	fr-CA
Chile	Spanish	es-CL
Denmark	Danish	da-DK
Finland	Finnish	fi-FI
France	French	fr-FR
Germany	German	de-DE
Hong Kong SAR	Traditional Chinese	zh-HK
India	English	en-IN
Indonesia	English	en-ID
Italy	Italian	it-IT

Country/Region	Language	Market code
Japan	Japanese	ja-JP
Korea	Korean	ko-KR
Malaysia	English	en-MY
Mexico	Spanish	es-MX
Netherlands	Dutch	nl-NL
New Zealand	English	en-NZ
Norway	Norwegian	no-NO
People's republic of China	Chinese	zh-CN
Poland	Polish	pl-PL
Republic of the Philippines	English	en-PH
Russia	Russian	ru-RU
South Africa	English	en-ZA
Spain	Spanish	es-ES
Sweden	Swedish	sv-SE
Switzerland	French	fr-CH
Switzerland	German	de-CH
Taiwan	Traditional Chinese	zh-TW
Türkiye	Turkish	tr-TR
United Kingdom	English	en-GB
United States	English	en-US
United States	Spanish	es-US

Country codes

The following table lists the country codes that you may set the cc query parameter to. The list is subject to change.

Expand table

Country/Region	Country Code
Argentina	AR
Australia	AU
Austria	AT
Belgium	BE
Brazil	BR
Canada	CA
Chile	CL
Denmark	DK
Finland	FI
France	FR
Germany	DE
Hong Kong SAR	HK
India	IN
Indonesia	ID
Italy	IT
Japan	JP
Korea	KR
Malaysia	MY
Mexico	MX
Netherlands	NL
New Zealand	NZ
Norway	NO
People's Republic of China	CN
Poland	PL
Portugal	PT
Republic of the Philippines	PH

Country/Region	Country Code
Russia	RU
Saudi Arabia	SA
South Africa	ZA
Spain	ES
Sweden	SE
Switzerland	CH
Taiwan	TW
Türkiye	TR
United Kingdom	GB
United States	US

Bing supported language codes

The following table lists the Bing supported languages that you may set the setLang query parameter to. The list is subject to change.

Expand table

Supported Languages	Language Code
Arabic	ar
Basque	eu
Bengali	bn
Bulgarian	bg
Catalan	ca
Chinese (Simplified)	zh-hans
Chinese (Traditional)	zh-hant
Croatian	hr
Czech	cs
Danish	da

Supported Languages	Language Code
Dutch	nl
English	en
English-United Kingdom	en-gb
Estonian	et
Finnish	fi
French	fr
Galician	gl
German	de
Gujarati	gu
Hebrew	he
Hindi	hi
Hungarian	hu
Icelandic	is
Italian	it
Japanese	jp
Kannada	kn
Korean	ko
Latvian	lv
Lithuanian	lt
Malay	ms
Malayalam	ml
Marathi	mr
Norwegian (Bokmål)	nb
Polish	pl
Portuguese (Brazil)	pt-br
Portuguese (Portugal)	pt-pt

Supported Languages	Language Code
Punjabi	pa
Romanian	ro
Russian	ru
Serbian (Cyrylic)	sr
Slovak	sk
Slovenian	sl
Spanish	es
Swedish	sv
Tamil	ta
Telugu	te
Thai	th
Turkish	tr
Ukrainian	uk
Vietnamese	vi

HTTP status codes that Bing Web Search API may return

Article • 06/07/2023

> **Note**
>
> If you are using the Bing APIs, bring your own LLM S15 instance and expect to exceed the 1M queries per day quota, you should consider implementing monitoring to detect when you are nearing your limit so you can automate switching to the next tier, S16.

The following are the possible HTTP status codes that a request may return.

Status code	Description
200	Success.
400	One of the query parameters is missing or not valid.
401	The subscription key is missing or is not valid.
403	The user is authenticated (for example, they used a valid subscription key) but they don't have permission to the requested resource. Bing may also return this status if the caller exceeded their queries per day or queries per month quota.
410	The request used HTTP instead of the HTTPS protocol. HTTPS is the only supported protocol.
429	The caller exceeded their queries per second quota.
500	Unexpected server error.

If the request fails, the response contains an ErrorResponse object, which contains a list of Error objects that describe what caused the error. If the error is related to a parameter, the `parameter` field identifies the parameter that caused the issue. And if the error is related to a parameter value, the `value` field identifies the value that is not valid.

JSON

```
{
  "_type": "ErrorResponse",
```

```
  "errors": [
    {
      "code": "InvalidRequest",
      "subCode": "ParameterMissing",
      "message": "Required parameter is missing.",
      "parameter": "q"
    }
  ]
}

{
  "_type": "ErrorResponse",
  "errors": [
    {
      "code": "InvalidAuthorization",
      "subCode": "AuthorizationMissing",
      "message": "Authorization is required.",
      "moreDetails": "Subscription key is not recognized."
    }
  ]
}
```

Error codes

The following are the possible error code and sub-error code values.

Error code	SubCode	Description
ServerError	UnexpectedError ResourceError NotImplemented	HTTP status code is 500.
InvalidRequest	ParameterMissing ParameterInvalidValue HttpNotAllowed Blocked	Bing returns InvalidRequest whenever any part of the request is not valid. For example, a required parameter is missing or a parameter value is not valid. If the error is ParameterMissing or ParameterInvalidValue, the HTTP status code is 400. If you use the HTTP protocol instead of HTTPS, Bing returns HttpNotAllowed, and the HTTP status code is 410.

Error code	SubCode	Description
RateLimitExceeded	No sub-codes	Bing returns RateLimitExceeded whenever you exceed your queries per second (QPS) or queries per month (QPM) quota. If you exceed QPS, Bing returns HTTP status code 429, and if you exceed QPM, Bing returns 403.
InvalidAuthorization	AuthorizationMissing AuthorizationRedundancy	Bing returns InvalidAuthorization when Bing cannot authenticate the caller. For example, the Ocp-Apim-Subscription-Key header is missing or the subscription key is not valid. Redundancy occurs if you specify more than one authentication method. If the error is InvalidAuthorization, the HTTP status code is 401.
InsufficientAuthorization	AuthorizationDisabled AuthorizationExpired	Bing returns InsufficientAuthorization when the caller does not have permissions to access the resource. This can occur if the subscription key has been disabled or has expired. If the error is InsufficientAuthorization, the HTTP status code is 403.

Entity types used by Bing Web Search API

Article • 10/28/2020

This section contains the list entity hints that the EntityPresentationInfo object's `entityTypeHint` field can be set to. The hints are grouped by category of entities.

Base entity types

- Generic
- Person
- Place
- Media
- Organization

Place base type entity hints

- Attraction
- City
- Continent
- Country
- Hotel
- House
- LocalBusiness
- Locality
- MinorRegion
- Neighborhood
- Other
- PointOfInterest
- PostalCode
- RadioStation
- Region
- Restaurant
- State
- StreetAddress
- SubRegion
- TouristAttraction
- Travel

Media base type entity hints

- Book
- Movie
- TelevisionSeason
- TelevisionShow
- VideoGame

Event-related entity hints

- Event

Profession-related entity hints

- Actor
- Artist
- Attorney

Education-related entity hints

- CollegeOrUniversity
- School
- Speciality

Other entity hints

- Animal
- Car
- Drug
- Food
- Product
- SportsTeam

Create Bing Search resource through Azure Marketplace

Article • 02/15/2024

Here are the steps to create a Bing Search Service resource through Azure Marketplace and get your key.

1. Go to Azure Portal and sign in with your Microsoft account. If you don't have a Microsoft account, click **Create one!**.
2. From the portal, type *Bing* in the search box.
3. Under **Marketplace** in the search results, select the Bing service you're interested in (for example, **Bing Search** or **Bing Custom Search**).
4. If you have a free trial or pay account, skip to Create your Bing resource.
5. On the **Create a free account** splash screen, click **Start free**.
6. Next, you have the option of continuing with the free trial (click **Start free** again) or paying for an Azure subscription (click **Or buy now**). You can always start with the free trial and pay for a subscription later.

Free trial option

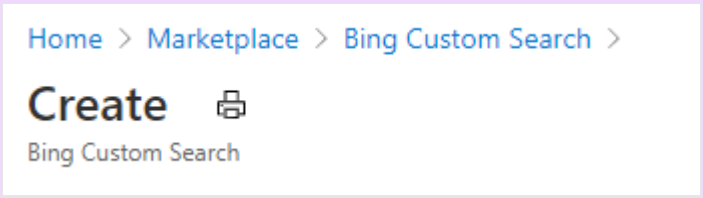
If you clicked **Start free**, simply follow the sign up process.

1. Provide your name and phone number. The process includes this step only if your Microsoft account profile doesn't include your name and phone number.
2. Next, verify your identify and phone number. Enter a phone number, if it's not already set. Choose to verify the phone number by using a text verification code or by receiving a phone call.
3. Add your credit card information and click **Next**. Don't worry, you won't be charged during the trial period. Read the **No automatic charges** section in the right pane.
4. Click the check box if you agree to the subscription agreement, offer details, and privacy statement.
5. Finally, click **Sign up**.

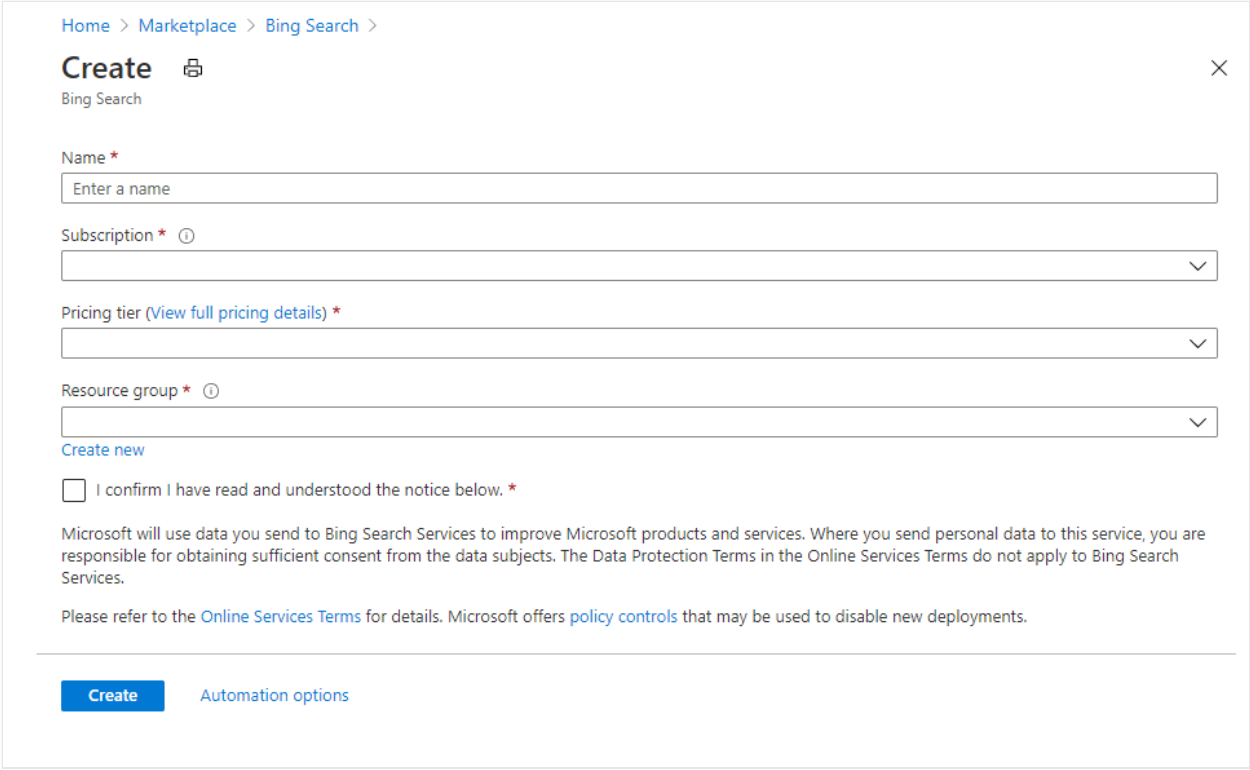
You should be redirected back to Azure Portal where you can create a resource and get your key. If Azure wasn't able to redirect you, go to Azure Portal and sign in with your Microsoft account. Back in the portal, type *Bing* in the search box. Under **Marketplace** in the search results, select the Bing service you're interested in (for example, **Bing Search** or **Bing Custom Search**).

Create your Bing resource

The following steps walk you through creating a Bing Search resource:

> **Note**
>
> While creating Bing Custom Search resource ensure **Bing Custom Search** is selected as top level service and not "Bing Search".
>
> 
>

1. Enter a resource name. Names may contain alphanumeric characters and dashes (-) only.
2. The **Subscription** field could be set to **Free Trial** or select your appropriate subscription.
3. In the **Pricing tier** dropdown, select **Free F1** package. The other packages are for the pay model. To view package options and pricing for the pay model, click **View full pricing details**.
4. If you have an existing resource group that you want to add this resource to, select it from the **Resource group** dropdown list. Otherwise, click **Create new** to create a resource group.
5. Select a location from the **Resource group location** dropdown. The location is where the metadata associated with your account resides and has no impact on runtime availability.
6. Click the check the box to indicate that you have read and understood the notice.
7. Click **Create**. This starts the deployment process, which can take several minutes.
8. When the deployment process completes, click **Go to resource**.
9. To get your subscription key to use in API calls, click **Keys and Endpoint** in the left pane.



Next steps

- Learn about calling the Bing Web Search API.
- Learn about the quickstarts and samples that are available to help you get up and running fast.
- Review Web Search API v7 reference documentation.

Bing Search API use and display requirements

Article • 01/29/2024

> **Note**
>
> The Use and Display requirements on this page apply to the Bing Search APIs. For Use and Display Requirements specific to using the Bing Search APIs, with your LLM, refer here.

These use and display requirements apply to any implementation of the content and associated information from the following Bing Search APIs, including relationships, metadata, and other signals.

- Bing Custom Search
- Bing Entity Search
- Bing Image Search
- Bing News Search
- Bing Video Search
- Bing Visual Search
- Bing Web Search
- Bing Spell Check
- Bing Autosuggest

Definitions

Expand table

Term	Description
Answer	A category of results returned in a response. For example, a response from the Bing Web Search API can include answers in the categories of webpage results, image, video, and news.
Response	Any and all answers and associated data received in response to a single call to a Search API.
Result	An item of information in an answer. For example, the set of data connected with a single news article is a result in a news answer.

Term	Description
Search APIs	Collectively, the Bing Custom Search, Entity Search, Image Search, News Search, Video Search, Visual Search, and Web Search APIs.

Bing Spell Check and Bing Autosuggest API restrictions

Do not:

- Copy, store, or cache any data you receive from the Bing Spell Check or Bing Autosuggest APIs.
- Use data you receive from Bing Spell Check or Bing Autosuggest APIs as part of any machine learning or similar algorithmic activity. Do not use this data to train, evaluate, or improve new or existing services that you or third parties might offer.
- Display data received from the Bing Spell Check or Bing Autosuggest APIs on the same page as content from any general web search engine, large language models or advertising network.

Bing Search APIs

> ⓘ **Note**
>
> The requirements in this section apply to only the Search APIs, which does not include Bing Spell Check or Bing Autosuggest.

Internet search experience requirements

All data returned in responses may only be used in internet search experiences. An internet search experience means the content displayed:

- Is relevant and responsive to the end user's direct query, or other indication of their search interest and intent (for example, a user-indicated search query).
- Helps users find and navigate to the response's data sources. For example, providing clickable links from hyperlinks in the response.
- Includes multiple results for the user to select from.
- Are in a placement that enables users to search.
- Includes a visible indication that the content is an internet search result. For example, a statement that the content is "from the web".

- Includes any other appropriate measures to ensure your Bing Search API data does not violate any applicable laws or rights of, or duties or obligations owed by you to, third parties. Consult your legal advisors to determine what measures may be appropriate.

The only exception to these internet search experience requirements is for URL discovery, as described later in this article.

Restrictions

Do not:

- Copy, store, or cache any data from responses (except retention to the extent permitted by continuity of service).
- Use data received from the Search APIs as part of any machine learning or similar algorithmic activity. Do not use this data to train, evaluate, or improve new or existing services that you or third parties might offer.
- Display data received from the Search APIs on the same page as content from any general web search engine, large language models or advertising network.
- Modify the results content (other than to reformat them in a way that does not violate any other requirement), unless required by law or agreed to by Microsoft.
- Omit attribution information and URLs associated with results content.
- Reorder, including by omission, the results displayed in an answer when an order or ranking is provided, unless required by law or agreed to by Microsoft.

  > ⓘ **Note**
  >
  > This requirement does not apply to reordering implemented through the portal for the Bing Custom Search API.

- Display content that was not included within any part of a response in a way that would lead a user to believe that content is part of the response.
- Use Responses for websites where you are restricted by the website from using such content, including but not limited to, where your crawler has been blocked via robots.txt.
- Display advertising that is not provided by Microsoft on any page that displays any part of a response.

- Display any advertising on pages with responses:
  - From the Bing Image, News Search, Video Search, or Visual Search APIs, or
  - That are filtered or limited primarily (or solely) to image, news and/or video or visual search results.

Notices and branding

Do:

- Prominently include a functional hyperlink to the Microsoft Privacy Statement, near each point in the user experience (UX) that offers a user the ability to input a search query. Label the hyperlink **Microsoft Privacy Statement**.
- Prominently display Bing branding, consistent with the Bing Trademark Usage Guidelines, near each point in the UX that offers a user the ability to input a search query. Such branding must clearly state to the user that Microsoft is powering the internet search experience.
- Attribute each response (or portion of a response) displayed from the Bing Web Search, Entity Search, Image Search, News Search, Video Search, and Visual Search APIs to Microsoft, unless Microsoft specifies otherwise in writing for your use. This is described in Bing Trademark Usage Guidelines.

Do not:

- Attribute responses (or portions of responses) displayed from the Bing Custom Search API to Microsoft, unless Microsoft specifies otherwise in writing for your particular use.

Transferring responses

If you enable a user to transfer a response from a Search API to another user, such as through a messaging app or social media posting, the following apply:

- Transferred responses must:
  - Consist of content that is unmodified from the content of the responses displayed to the transferring user. Formatting changes are permissible.
  - Not include any data in metadata form.
  - Display language indicating that a Bing Web, Image, News, Video, or Visual API response was obtained through an internet search experience powered by Bing. For example, you can display language such as "Powered by Bing" or "Learn more about this image on Bing," or you can use the Bing logo.
  - Display language indicating that a Bing Custom Search API response was obtained through an internet search experience. For example, you can display

language such as "Learn more about this search result.”
    - Prominently display the full query used to generate the response.
    - Include a prominent link or similar attribution to the underlying source of the response, either directly or through the search engine (bing.com, m.bing.com, or your custom search service, as applicable).
- You may not automate the transfer of responses. A transfer must be initiated by a user action clearly evidencing an intent to transfer a response.
- You may only enable a user to transfer responses that were displayed in response to the transferring user's query.

Continuity of service

Do not copy, store, or cache any data from Search API responses. However, to enable continuity of service access and data rendering, you may retain results solely under the following conditions:

Device

You may enable a user to retain results on a device for the lesser of (i) 24 hours from the time of the query, or (ii) until a user submits another query for updated results, provided that retained results may be used only:

- To enable the user to access results previously returned to that user on that device (for example, in case of service interruption).
- To store results returned for your proactive query personalized in anticipation of the user's needs, based on that user's signals (for example, in case of anticipated service interruption).

Server

You may retain results specific to a single user securely on a server you control, and display the retained results only:

- To enable the user to access a historical report of results previously returned to that user in your solution. The results may not be (i) retained for more than 21 days from the time of the end user's initial query, and (ii) displayed in response to a user's new or repeated query.
- To store results returned for your proactive query personalized in anticipation of the user's needs, based on that user's signals. You can store these results for the lesser of (i) 24 hours from the time of the query, or (ii) until a user submits another query for updated results.

Whenever retained, results for a specific user cannot be commingled with results for another user. That is, the results of each user must be retained and delivered separately.

General

For all presentation of retained results:

- Include a clear, visible notice of the time the query was sent.
- Present the user with a button or similar means to re-query and obtain updated results.
- Retain the Bing branding in the presentation of the results.
- Delete (and refresh with a new query if needed) the stored results within the timeframe specified.

Non-display URL discovery

You may only use search responses in a non-internet search experience for the sole purpose of discovering URLs of sources of information responsive to a query from your user or customer. You may copy such URLs in a report or similar response you provide:

- Only to that user or customer, in response to that query.
- Only if it includes significant additional valuable content, relevant to the query.

The previous sections of Search APIs use and display requirements do not apply to this non-display URL discovery use, except for the following:

- Do not cache, copy, or store any data or content from, or derived from, the search response, other than the limited URL copying described previously.
- Ensure your use of data (including the URLs) received from the Search APIs does not violate any applicable laws or rights of, or duties or obligations owed by you to, third parties.
- Do not use the data (including the URLs) received from the Search APIs as part of any search index or machine learning or similar algorithmic activity. Do not use this data to create train, evaluate, or improve services that you or third parties might offer.

GDPR compliance

With respect to any personal data subject to the European Union General Data Protection Regulation (GDPR) and that is processed in connection with calls to the Search APIs, Bing Spell Check API, or Bing Autosuggest API, you understand that you

and Microsoft are independent data controllers under the GDPR. You are independently responsible for your compliance with the GDPR.

Use and Display requirements of Bing Search APIs, with your LLM

Article • 01/29/2024

> **Note**
>
> The Use and Display requirements on this page apply to the Bing Search APIs, with your LLM. For Use and Display Requirements specific to using the Bing Search APIs, refer **here**.

These Bing Search APIs, with your LLM use and display requirements apply to any implementation of the content and associated information from the following Bing Search APIs, including relationships, metadata, and other signals. Specific terms apply to use of Bing Search APIs, with your LLM used in connection with large language models (LLMs), see "LLM use and display requirements" below.

- Bing Entity Search
- Bing Image Search
- Bing News Search
- Bing Video Search
- Bing Web Search
- Bing Spell Check
- Bing Autosuggest

Note use of Bing Custom Search and Bing Visual Search is not included in Bing Search APIs, with your LLM.

Definitions

Expand table

Term	Description
Answer	A category of results returned in a response. For example, a response from the Bing Web Search API can include answers in the categories of webpage results, image, video, and news.
Grounding	The process of allowing an LLM model to temporarily access and use the Web Results to formulate or augment an LLM response for a single query and user, but not for the purposes of training the LLM model with such data for future use.

Term	Description
LLM(s)	Large language model(s).
Response	Any and all answers and associated data received in response to a single call to a Search API.
Result	An item of information in an answer. For example, the set of data connected with a single news article is a result in a news answer.
Search APIs	Collectively, Entity Search, Image Search, News Search, Video Search, and Web Search APIs.
Web Results	The title, URL and snippet for the top ten webpage results returned from the Bing Web Search API.

Bing Spell Check and Bing Autosuggest API restrictions

Do not:

- Copy, store, or cache any data you receive from the Bing Spell Check or Bing Autosuggest APIs.
- Use data you receive from the Bing Spell Check or Bing Autosuggest APIs as part of any machine learning or similar algorithmic activity. Do not use this data to train, evaluate, or improve new or existing services that you or third parties might offer.
- Display data received from the Bing Spell Check or Bing Autosuggest APIs on the same page as content from any general web search engine, LLMs (except as permitted below) or advertising network.
- Attribute to Microsoft responses (or parts of responses) displayed from the Bing Spell Check or Bing Autosuggest APIs, unless Microsoft specifies otherwise in writing for your particular use.

Bing Search APIs

Note:

The requirements in this section apply to only the Search APIs, which does not include Bing Spell Check or Bing Autosuggest. This section applies to use and display of Responses from the Search APIs.

When using in connection with LLMs specific additional terms apply, see "LLM use and display requirements" below.

Internet search experience requirements

All data returned in Responses may only be used in internet search experiences. An internet search experience means the content displayed:

- Is relevant and responsive to the end user's direct query, or other indication of their search interest and intent (for example, a user-indicated search query).
- Helps users find and navigate to the response's data sources. For example, providing clickable links from hyperlinks in the response.
- Includes multiple results for the user to select from.
- Are in a placement that enables users to search.
- Includes a visible indication that the content is an internet search result. For example, a statement that the content is "from the web".
- Includes any other appropriate measures to ensure your Bing Search API data does not violate any applicable laws or rights of, or duties or obligations owed by you to, third parties. Consult your legal advisors to determine what measures may be appropriate.

Restrictions

Do not:

- Copy, store, or cache any data from Responses.
- Use data received from the Search APIs as part of any machine learning or similar algorithmic activity. Do not use this data to train, evaluate, or improve new or existing services that you or third parties might offer.
- Display data received from the Search APIs on the same page as content from any general web search engine, LLMs (except as permitted below) or advertising network.
- Modify the results content (other than to reformat them in a way that does not violate any other requirement), unless required by law or agreed to by Microsoft.
- Omit attribution information and URLs associated with results content.
- Reorder, including by omission, the results displayed in an answer when an order or ranking is provided, unless required by law or agreed to by Microsoft.
- Attribute to Microsoft Responses (or parts of Responses) displayed from the Search APIs, unless Microsoft specifies otherwise in writing for your particular use.
- Display content that was not included within any part of a Response in a way that would lead a user to believe that content is part of the Response.

- Use Responses for websites where you are restricted by the website from using such content, including but not limited to, where your crawler has been blocked via robots.txt.
- Display advertising that is not provided by Microsoft on any page that displays any part of a Response.
- Display any advertising on pages with Responses:
  - From the Bing Image, News Search, or Video Search APIs, or
  - That are filtered or limited primarily (or solely) to image, news and/or video results.

Privacy Notice

Do:

- Prominently include a functional hyperlink to the Microsoft Privacy Statement, near each point in the user experience (UX) that offers a user the ability to input a search query. Label the hyperlink **Microsoft Privacy Statement**.

LLM use and display requirements for Search APIs, Bing Spell Check API and Bing Autosuggest API

When using Search APIs, Bing Spell Check API, or Bing Autosuggest API in connection with LLMs, you must follow the guidelines below:

You may only:

- Display responses received from the Search APIs, Bing Spell Check API and Bing Autosuggest API on the same webpage as content from an LLM, provided the Search APIs, Bing Spell Check API and Bing Autosuggest API responses are clearly separated from the LLM content (e.g. the LLM content should not be inserted in between the Search APIs Results); and/or
- Use Web Results for Grounding an LLM.

Do not:

- Use any responses (including, without limitation, web, images, videos, news, entity data, auto suggest, spell etc.) to train, evaluate or improve any LLM, including your proprietary LLM;
- Use Responses for websites where you are restricted by the website from using such content, including but not limited to, where your crawler has been blocked

via robots.txt;
- Display advertising that is not provided by Microsoft; or
- Attribute the Web Results for Grounding your LLM to Microsoft, unless Microsoft specifies otherwise in writing for your particular use.

LLM Attribution:

- You must provide source attribution and a link back to each source that is used for the content being displayed from the LLM, when using Web Results for Grounding an LLM. The source attribution must be near the displayed LLM content.

GDPR compliance

With respect to any personal data subject to the European Union General Data Protection Regulation (GDPR) and that is processed in connection with calls to the Search APIs, Bing Spell Check API, or Bing Autosuggest API, you understand that you and Microsoft are independent data controllers under the GDPR. You are independently responsible for your compliance with the GDPR.

Release notes for Web Search API

Article • 09/27/2022

See the following sections for information about changes that were included with each release.

October 30, 2020

Initial release of this version of Bing Web Search API. This API replaces the same API hosted by Azure Cognitive Services, which is being phased out.

New to Bing Search?

To subscribe to a package that includes this API, see Create Bing Search Service resource. Next, get familiar with the Web Search API documentation, and be sure to check out the quickstarts to get up and running quickly.

Current Cognitive Services user?

Things current Cognitive Services users must do prior to their subscription ending:

1. Update your application to use your Bing Search Services subscription key. Current Cognitive Services users that want to maintain continuity of service should sign up here for a Bing Search Services subscription before their Azure Cognitive Services subscription ends.

2. Update your app to use the new endpoints. For the new endpoints, see Web Search API reference.

If you use Local Business Search API, it's not available in Bing Search Services.