데스크톱 Windows 애플리케이션용 API 인덱스

이 문서에서는 데스크톱 Windows 앱에서 사용할 수 있는 API에 대한 참조 설명서에 대한 링크를 제공합니다.

Win32(Windows API)

Win32 API(Windows API라고도 함)는 Windows 앱의 네이티브 플랫폼입니다. 이 API는 시스템 기능 및 하드웨어에 직접 액세스해야 하는 데스크톱 앱에 가장 적합합니다. Windows API는 모든 데스크톱 앱에서 사용할 수 있으며 일반적으로 32비트 및 64비트 Windows에서 동일한 함수가 지원됩니다.

- 기능 의한 Win32 API 참조
- 헤더 Win32 API 참조
- Win32 및 UWP 앱용 COM API
- Windows 우산 라이브러리
- Windows API 집합

WinRT(Windows 런타임)

WinRT는 데스크톱 앱을 포함하여 Windows 10 앱 및 게임을 위한 최고의 에지 플랫폼입니다. WinRT API는 정교한 UI, 스타일 사용자 지정 및 그래픽 집약적 시나리오가 필요한 네이티브 C++ 및 관리형 데스크톱 앱 모두에 적합합니다.

- WinRT API 참조
- 데스크톱 앱에서 호출할 수 있는 WinRT API

.그물

.NET 클래스 라이브러리는 WPF 및 Windows Forms 앱을 비롯한 관리형 데스크톱 앱에 대한 Windows 시스템 및 UI 기능에 대한 액세스를 제공합니다.

- .NET API

Last updated on 2025. 03. 13.

Windows API 인덱스

다음은 데스크톱 및 서버 애플리케이션에 대한 Windows API(애플리케이션 프로그래밍 인터페이스)에 대한 참조 콘텐츠 목록입니다.

Windows API를 사용하여 각 버전에 고유한 기능과 기능을 활용하면서 모든 버전의 Windows에서 성공적으로 실행되는 애플리케이션을 개발할 수 있습니다. (이전에 Win32 API라고 했습니다. Windows API라는 이름은 16비트 Windows의 루트와 64비트 Windows에서의 지원을 보다 정확하게 반영합니다.)

사용자 인터페이스

Windows UI API는 창을 만들고 사용하여 출력을 표시하고, 사용자 입력을 요청하며, 사용자와의 상호 작용을 지원하는 다른 작업을 수행합니다. 대부분의 애플리케이션은 하나 이상의 창을 만듭니다.

- 접근성
- DWM(데스크톱 창 관리자)
- 세계화 서비스
- 높은 DPI
- MUI(다국어 사용자 인터페이스)
- NLS(국가 언어 지원)
- 사용자 인터페이스 요소 :
  - 단추
  - 캐럿
  - 콤보 상자
  - 일반 대화 상자
  - 공용 컨트롤
  - 커서
  - 대화 상자
  - 컨트롤 편집
  - 헤더 컨트롤
  - 아이콘
  - 키보드 가속기
  - 목록 상자
  - List-View 컨트롤
  - 메뉴

    - 진행률 표시줄
    - 속성 시트
    - 서식 있는 편집 컨트롤
    - 스크롤 막대
    - 정적 컨트롤
    - 문자열
    - 도구 모음
    - 도구 설명
    - 트랙바
    - Tree-View 컨트롤

- windows 애니메이션 관리자

- Windows 리본 프레임워크

Windows 환경(Shell)

- windows 속성 시스템
- windows Shell
- windows Search
- 콘솔

사용자 입력 및 메시징

- 사용자 상호 작용
    - 직접 조작
    - 잉크 입력
    - 입력 피드백 구성
    - 상호 작용 컨텍스트
    - 포인터 디바이스 입력 스택
    - 포인터 입력 메시지 및 알림
    - 방사형 컨트롤러 입력
    - Text Services 프레임워크
    - 터치 히트 테스트
    - 터치 삽입

- 레거시 사용자 상호 작용
    - 터치 입력
    - 키보드 입력
    - 마우스 입력
    - 원시 입력

- Windows 및 메시지:
  - 메시지 및 메시지 큐
  - windows
  - 창 클래스
  - 창 프로시저
  - 타이머
  - 창 속성
  - 후크

데이터 액세스 및 스토리지

- BITS(Background Intelligent Transfer Service)
- 링크 바인딩
- 데이터 백업
  - 백업
  - 데이터 중복 제거
  - 볼륨 섀도 복사본
  - Windows Server Backup
- Data Exchange:
  - 클립보드
  - DDE(동적 데이터 교환)
  - DDEML(동적 데이터 교환 관리)
- 디렉터리 관리
- 디스크 관리
- DFS(분산 파일 시스템)
- 분산 파일 시스템 복제
- 확장 가능한 스토리지 엔진
- 파일 및 I/O(로컬 파일 시스템)
- iSCSI 검색 라이브러리 API
- 오프라인 파일
- 패키징
- 원격 차등 압축

- 트랜잭션 NTFS
- 볼륨 관리
- VHD(가상 하드 디스크)
- Windows Storage 관리
- Windows 데이터 액세스 구성 요소
  - Microsoft ODBC(Open Database Connectivity)
  - Microsoft OLE DB
  - Microsoft ADO(ActiveX Data Objects)

진단

진단 API를 사용하면 애플리케이션 또는 시스템 문제를 해결하고 성능을 모니터링할 수 있습니다.

- Application Recovery 및 다시 시작
- 디버깅
- 오류 처리
- 이벤트 로깅
- 이벤트 추적
- HCP(하드웨어 카운터 프로파일링)
- NDF(네트워크 진단 프레임워크)
- 네트워크 모니터
- 성능 카운터
- PLA(성능 로그 및 경고)
- 프로세스 스냅샷
- PSAPI(프로세스 상태)
- 구조적 예외 처리
- 시스템 모니터
- Wait Chain 순회
- WER(Windows 오류 보고)
- windows 이벤트 로그
- Windows 문제 해결 플랫폼

그래픽 및 멀티미디어

그래픽, 멀티미디어, 오디오 및 비디오 API를 사용하면 애플리케이션에서 서식이 지정된 텍스트, 그래픽, 오디오 및 비디오를 통합할 수 있습니다.

- 코어 오디오
- Direct2D
- directComposition
- DirectShow
- DirectWrite
- DirectX
- GDI(그래픽 디바이스 인터페이스)
- GDI+
- 미디어 스트리밍
- Microsoft Media Foundation
- microsoft TV 기술
- OpenGL
- Monitor Configuration
- 여러 디스플레이 모니터
- 그림 취득
- Windows 색 시스템
- WIC(Windows 이미징 구성 요소)
- windows Media Audio 및 Video Codec 및 DSP
- windows Media Center
- windows Media 형식
- windows Media Library Sharing Services
- windows Media Player
- Windows Media Services
- windows Movie Maker
- windows 멀티미디어

장치

- AllJoyn
- Communications 리소스
- 디바이스 액세스
- 디바이스 관리
- 향상된 스토리지
- 함수 검색
- 이미지 마스터링
- 위치
- PnP-X 연결 데이터베이스
- 인쇄
  - 인쇄 스풀러
  - 문서 패키지 인쇄

    - 인쇄 스키마 사양
    - 인쇄 티켓
    - XPS 인쇄
- 센서
- SENS(시스템 이벤트 알림 서비스)
- 도구 도움말
- UPnP
- 디바이스 웹 서비스
- WIA(Windows 이미지 획득)
- Windows Media Device Manager
- Windows 이식 가능 디바이스

시스템 서비스

System Services API를 통해 애플리케이션은 컴퓨터의 리소스와 메모리, 파일 시스템, 디바이스, 프로세스 및 스레드와 같은 기본 운영 체제의 기능에 액세스할 수 있습니다.

- 활동 코디네이터
- COM
- COM+
- 압축 API
- DTC(분산 트랜잭션 코디네이터)
- DLL(Dynamic-Link 라이브러리)
- HWREQCHK(하드웨어 요구 사항 평가기)
- 도움말 API
- Interprocess Communications:
    - Mailslots
    - 파이프
- KTM(커널 트랜잭션 관리자)
- 메모리 관리
- 작업 레코더
- 전원 관리
- 원격 데스크톱 서비스
- 프로세스
- Services
- 동기화
- 스레드
- Windows 데스크톱 공유
- Windows 시스템 정보
    - 핸들 및 개체
    - 레지스트리

    - 시간
    - 시간 공급자

보안 및 ID

보안 및 ID API를 사용하면 로그온 시 암호 인증, 모든 공유 가능한 시스템 개체에 대한 임의 보호, 권한 있는 액세스 제어, 권한 관리 및 보안 감사가 가능합니다.

- 인증
- 권한 부여
- 인증서 등록
- 암호화
- CNG(암호화 차세대)
- Directory Services
    - Active Directory Domain Services
    - ADSI(Active Directory Service Interfaces)
- EAP(확장 가능 인증 프로토콜)
- EAPHost(확장 가능 인증 프로토콜 호스트)
- MS-CHAP 암호 관리
- NAP(네트워크 액세스 보호)
- NPS(네트워크 정책 서버 확장)
- 자녀 보호
- 보안 WMI 공급자
- TBS(TPM 기본 서비스)
- Windows 생체 인식 프레임워크

애플리케이션 설치 및 서비스

- 게임 탐색기
- side-by-side 어셈블리
- 패키징, 배포 및 쿼리 API
- 개발자 라이선스
- 다시 시작 관리자
- windows Installer

시스템 관리자 및 관리

시스템 관리 인터페이스를 사용하면 애플리케이션 또는 시스템을 설치, 구성 및 서비스할 수 있습니다.

- 부팅 구성 데이터 WMI 공급자
- 장애 조치(failover) 클러스터
- FSRM(파일 서버 리소스 관리자)
- 그룹 정책
- MMC(Microsoft Management Console) 2.0
- NetShell
- 설정 관리 인프라
- 소프트웨어 인벤토리 로깅
- 소프트웨어 라이선싱
- 다시 시작 관리자
- 설정 관리 인프라
- 시스템 복원
- 시스템 종료
- 작업 스케줄러
- 사용자 액세스 로깅
- Windows Virtual PC
- Microsoft Virtual Server
- 네트워크 부하 분산 공급자
- windows Defender WMI v2
- windows Deployment Services
- Windows 정품 어드밴티지
- Windows 관리 인프라
- WMI(Windows Management Instrumentation)
- Windows 원격 관리
- Windows 리소스 보호
- Windows Server Update Services
- windows 시스템 평가 도구
- windows 업데이트 에이전트

네트워킹 및 인터넷

네트워킹 API를 사용하면 네트워크를 통해 애플리케이션 간에 통신할 수 있습니다. 디렉터리 및 네트워크 프린터와 같은 공유 리소스에 대한 액세스를 만들고 관리할 수도 있습니다.

- DNS(도메인 이름 시스템)
- DHCP(동적 호스트 구성 프로토콜)
- 팩스 서비스
- 연결 마법사
- HTTP 서버
- 인터넷 연결 공유 및 방화벽
- IP 도우미

- IPv6 인터넷 연결 방화벽
- 관리 정보 베이스
- MSMQ(메시지 큐)
- 멀티캐스트 주소 MADCAP(동적 클라이언트 할당 프로토콜)
- NAT(네트워크 주소 변환)
- NLM(네트워크 목록 관리자)
- 네트워크 관리
- 네트워크 공유 관리
- 피어 투 피어
- 서비스 품질(QOS)
- 원격 프로시저 호출
- 라우팅 및 RAS(원격 액세스 서비스)
- SNMP(Simple Network Management Protocol)
- SMB 관리
- TAPI(전화 통신 애플리케이션 프로그래밍 인터페이스)
- WebDAV
- WebSocket 프로토콜 구성 요소
- 무선 네트워킹:
  - Bluetooth
  - irDA
  - 모바일 광대역
  - 네이티브 Wifi
  - Windows Connect Now
  - Windows 연결 관리자
- Windows 필터링 플랫폼
- 고급 보안 사용하여 Windows 방화벽
- WinHTTP(Windows HTTP 서비스)
- WinINet(Windows Internet)
- WNet(Windows 네트워킹)
- Windows 네트워크 가상화
- Windows RSS 플랫폼
- Windows 소켓(Winsock)
- Windows Web Services
- XML HTTP 확장 요청

사용되지 않거나 레거시 API

다음은 오래되었거나 Windows 클라이언트 및 서버 운영 체제에서 대체되거나 사용되지 않는 기술 및 API입니다.

- directMusic

- DirectSound
- Microsoft UDDI SDK 이제 Microsoft BizTalk Server포함됩니다.
- 네트워크 DDE(동적 데이터 교환)
- 원격 설치 서비스: 대신 Windows 배포 서비스 사용합니다.
- VDS(Virtual Disk Service): 대신 Windows Storage 관리 사용합니다.
- 터미널 서비스: 원격 데스크톱 서비스사용합니다.
- Windows Media Rights Manager
- MAPI(Windows 메시징) : 대신 Office MAPI 사용합니다.
- Windows 가젯 플랫폼: 대신 UWP 앱을 만듭니다.
- Windows 사이드바: UWP 앱을 대신 만듭니다.
- Windows SideShow : 대체하지 않습니다.
- WPF 비트맵 효과

Last updated on 2025. 03. 13.

Windows API 세트

> **Important**
>
> 이 항목의 정보는 모든 버전의 Windows 10 이상에 적용됩니다. 여기서는 이러한 버전을 "Windows"라고 하며 필요한 경우 예외를 호출합니다.

모든 버전의 Windows는 핵심 *OS라고 하는* OS(운영 체제) 구성 요소의 공통 기반을 공유합니다(일부 컨텍스트에서는 이 공통 기반을 OneCore*라고도 함*). 핵심 OS 구성 요소에서 Win32 API는 API 집합*이라는* 기능 그룹으로 구성됩니다.

API 집합의 목적은 지정된 Win32 API가 구현되는 호스트 DLL과 API가 속한 기능 계약에서 아키텍처 분리를 제공하는 것입니다. API 집합이 구현과 계약 간에 제공하는 분리는 개발자에게 많은 엔지니어링 이점을 제공합니다. 특히 코드에서 API 집합을 사용하면 Windows 디바이스와의 호환성을 향상시킬 수 있습니다.

API 집합은 특히 다음 시나리오를 해결합니다.

- Win32 API의 전체 범위는 PC에서 지원되지만 HoloLens, Xbox 및 기타 장치와 같은 다른 Windows 장치에서는 Win32 API의 하위 집합만 사용할 수 있습니다. API 집합 이름은 지정된 디바이스에서 API를 사용할 수 있는지 여부를 명확하게 검색하는 쿼리 메커니즘을 제공합니다.
- 일부 Win32 API 구현은 여러 Windows 디바이스에서 서로 다른 이름을 가진 DLL에 존재합니다. API 가용성을 검색하고 API 로드를 지연할 때 DLL 이름 대신 API 집합 이름을 사용하면 API가 실제로 구현되는 위치에 관계없이 구현에 대한 올바른 경로를 제공합니다.

자세한 내용은 API 집합 로더 작업 및 API 집합 가용성 검색을 참조하세요.

API 집합 및 dll이 동일한가요?

아니요- API 집합 이름은 실제 파일의 `.dll`. 호출자로서 정보를 호스팅하는 모듈을 정확히 알 필요가 없는 구현 숨기기 기술입니다.

이 기술을 사용하면 다른 Windows 버전 및 버전에서 모듈을 리팩터링(분할, 통합, 이름 바꾸기 등)할 수 있습니다. 앱은 여전히 연결되고 런타임에 올바른 코드로 라우팅됩니다.

그렇다면 API 집합의 이름에 있는 `.dll` 이유는 무엇일까요? 그 이유는 DLL 로더*가* 구현되는 방식입니다. 로더는 DLL을 로드하거나 DLL에 대한 참조를 확인하는 OS의 일부입니다. 그리고 프런트 엔드에서 로더는 LoadLibrary에 전달된 문자열을 ".dll"로 종료해야 합니다. 그러나 프런트 엔드 후에 로더는 해당 접미사를 제거하고 결과 문자열을 사용하여 API 집합 데이터베이스를 쿼리할 수 있습니다.

LoadLibrary (및 지연 로드)는 API 집합 이름(".dll")으로 성공하지만 PC의 어느 곳에서도 해당 이름의 실제 파일이 반드시 있는 것은 아닙니다.

우산 라이브러리 연결

코드를 핵심 OS에서 지원되는 Win32 API로 쉽게 제한할 수 있도록 일련의 *우산 라이브러리*를 제공합니다. 예를 들어 명명 `OneCore.lib` 된 우산 라이브러리는 모든 Windows 디바이스에 공통적인 Win32 API의 하위 집합에 대한 내보내기를 제공합니다.

자세한 내용은 Windows 우산 라이브러리를 참조 하세요.

API 집합 계약 이름

API 집합은 라이브러리 로더에서 인식하는 이러한 표준 규칙을 따르는 강력한 계약 이름으로 식별됩니다.

- 이름은 문자열 **api 또는ext**-로 시작해야 합니다.
  - api로 시작하는 이름은 API의 버전 요구 사항을 충족하는 모든 Windows 버전에 존재하는 API를 나타냅니다.
  - ext로 시작하는 이름은 모든 Windows 버전에 존재하지 않을 수 있는 API를 나타냅니다.
- 이름은 n-n-n<>로끝나야 합니다. 여기서 **n**은 10진수로 구성됩니다.
- 이름의 본문은 영숫자 문자 또는 대시(-)일 수 있습니다.
- 이름은 대소문자를 구분하지 않습니다.

다음은 API 집합 계약 이름의 몇 가지 예입니다.

- **api-ms-win-core-ums-l1-1-0**
- **ext-ms-win-com-ole32-l1-1-5**
- **ext-ms-win-ntuser-window-l1-1-0**
- **ext-ms-win-ntuser-window-l1-1-1**

DLL 모듈 이름 대신 LoadLibrary 또는 P/Invoke와 같은 로더 작업의 컨텍스트에서 API 집합 이름을 사용하여 API가 현재 디바이스에서 실제로 구현되는 위치에 관계없이 구현에 대한 올바른 경로를 보장할 수 있습니다. 그러나 이렇게 하면 계약 이름의 끝에 문자열 **.dll** 추가해야 합니다. 이는 로더가 제대로 작동하기 위한 요구 사항이며 실제로 계약 이름의 일부로 간주되지 않습니다. 계약 이름은 이 컨텍스트에서 DLL 이름과 비슷하게 표시되지만 DLL 모듈 이름과 근본적으로 다르며 디스크의 파일을 직접 참조하지는 않습니다.

로더 작업에서 문자열 **.dll** 추가하는 경우를 제외하고 API 집합 계약 이름은 특정 계약 버전에 해당하는 변경할 수 없는 식별자로 간주되어야 합니다.

Win32 API에 대한 API 집합 식별

특정 Win32 API가 API 집합에 속하는지 여부를 식별하려면 API에 대한 참조 설명서의 요구 사항 테이블을 검토합니다. API가 API 집합에 속하는 경우 문서의 요구 사항 테이블에는 API 집합 이름과 API가 API 집합에 처음 도입된 Windows 버전이 나열됩니다. API 집합에 속하는 API의 예제는 다음 문서를 참조하세요.

- AllowSetForegroundWindow
- FindWindowsEx
- GetClassFile

이 섹션의 내용

- API 세트 로더 작업
- API 세트 가용성 검색
- Windows 우산 라이브러리

Last updated on 2025. 03. 13.

API 집합 로더 작업

> **Important**
>
> 이 항목의 정보는 모든 버전의 Windows 10 이상에 적용됩니다. 여기서는 이러한 버전을 "Windows"라고 하며 필요한 경우 예외를 호출합니다.

API 집합은 라이브러리 로더의 OS 지원을 사용하여 라이브러리 바인딩 프로세스에 모듈 네임스페이스 리디렉션을 효과적으로 도입할 있습니다. API 집합 계약 이름 라이브러리 로더가 API 집합의 적절한 구현을 포함하는 대상 호스트 이진 파일에 대한 참조의 런타임 리디렉션을 수행하는 데 사용됩니다.

로더가 런타임에 API 집합에 대한 종속성을 발견하면 로더는 이미지의 구성 데이터를 참조하여 API 집합에 대한 호스트 이진 파일을 식별합니다. 이 구성 데이터를 **API 집합 스키마**호출합니다. 스키마는 OS의 속성으로 어셈블되며, 지정된 디바이스에 포함된 이진 파일에 따라 API 집합과 이진 파일 간의 매핑이 다를 수 있습니다. 스키마를 사용하면 이진 호스트의 모듈 이름이 변경되었거나 다른 Windows 디바이스에서 완전히 리팩터링된 경우에도 단일 이진 파일의 가져온 함수를 다른 디바이스에서 올바르게 라우팅할 수 있습니다.

Windows는 API 집합을 사용하고 인터페이스하는 두 가지 표준 기술인 **직접 전달** 및 역방향 전달 지원합니다.

직접 전달

이 구성에서 사용하는 코드는 API 집합 모듈 이름을 직접 가져옵니다. 이 가져오기는 단일 작업에서 확인되며 오버헤드가 가장 적은 가장 효율적인 방법입니다. 개념적으로 이 해결 방법은 다음 예제와 같이 여러 Windows 디바이스의 서로 다른 이진 파일을 가리킬 수 있습니다.

가져온 API 집합: **api-feature1-l1-1-0.dll**

- Windows PC ->**feature1.dll**
- HoloLens ->**feature1_holo.dll**
- IoT ->**feature1_iot.dll**

매핑은 사용자 지정 스키마 데이터 리포지토리에 유지되므로 **.dll** 끝나는 API 집합 이름은 디스크의 파일을 직접 참조하지 않습니다. API 집합 이름의 **.dll** 부분은 로더에 필요한 규칙일 뿐입니다. API 집합 이름은 실제 DLL 파일의 별칭 또는 가상 이름과 비슷합니다. 이렇게 하면 Windows 디바이스의 전체 범위에서 이름을 이식할 수 있습니다.

역방향 전달

API 집합 이름은 디바이스에서 모듈에 안정적인 네임스페이스를 제공하지만 모든 이진 파일을 이 새 시스템으로 변환하는 것이 항상 실용적인 것은 아닙니다. 예를 들어 애플리케이션이 몇 년 동안 일반적으로 사용되었을 수 있으며 애플리케이션의 이진 파일을 다시 컴파일하는 것은 불가능할 수 있습니다. 또한 일부 애플리케이션은 특정 API 집합이 도입되기 전에 빌드된 시스템에서 계속 실행해야 할 수 있습니다.

이러한 수준의 호환성을 수용하기 위해 Win32 API 화면의 하위 집합을 포함하는 모든 Windows 디바이스에 전달자 시스템이 제공됩니다. 이러한 전달자는 Windows PC에 도입된 모듈 이름을 사용하고 API 집합 시스템을 활용하여 모든 Windows 디바이스에서 호환성을 제공합니다.

로더 작업은 다음과 같이 작동합니다.

1. Windows PC 이외의 디바이스에서 로더에는 디바이스에 없는 레거시 Windows PC 모듈 이름 종속성이 표시됩니다.
2. 로더는 이 모듈에 대한 API 집합 전달자를 찾아 메모리에 로드합니다.
3. 전달자에는 호출되는 지정된 함수에 대한 API 집합에 대한 매핑이 있습니다.
4. 로더는 지정된 디바이스에 대한 적절한 호스트 이진 파일을 찾습니다.

개념적으로 매핑은 다음과 같습니다.

가져온 DLL: **feature1.dll**

- Windows PC ->**feature1.dll**
- HoloLens ->**feature1.dll** 전달자 ->**api-feature1-l1-1-0.dll** ->**feature1_holo.dll**
- IoT ->**feature1.dll** 전달자 ->**api-feature1-l1-1-0.dll** ->**feature1_iot.dll**

최종 결과는 직접 전달기능적으로 동일하지만 애플리케이션 호환성을 최대화하는 방식으로 수행합니다.

> **참고 항목**
>
> 역방향 전달은 Win32 API 표면의 하위 집합에 대해서만 적용 범위를 제공합니다. Windows 데스크톱 버전을 대상으로 하는 애플리케이션이 모든 Windows 디바이스에서 실행되는 것은 허용되지 않습니다.

Last updated on 2025. 03. 13.

API 집합 가용성 검색

경우에 따라 지정된 API 집합 계약 이름이 일부 Windows 디바이스의 빈 모듈 이름에 의도적으로 매핑될 수 있습니다. 그 이유는 다양하지만 일반적인 예는 리소스가 제한된 디바이스에 대해 구성할 때 시스템 리소스 측면에서 비용이 많이 드는 기능을 Windows OS에서 제거할 수 있기 때문입니다. 이렇게 하면 애플리케이션이 API 수준에서 선택적 기능을 정상적으로 처리해야 하는 문제가 발생합니다.

Win32 API를 사용할 수 있는지 여부를 테스트하는 기존의 방법은 LoadLibrary 사용하거나 GetProcAddress . 그러나 이는 Windows 10 이상에서 역방향 전달 지원으로 인해 API 집합을 테스트하기 위한 신뢰할 수 있는 수단이 아닙니다. 역방향 전달이 지정된 API에 적용되는 경우 **LoadLibrary** 또는 **GetProcAddress** 내부 구현이 제거된 경우에도 유효한 함수 포인터로 확인될 수 있습니다. 이 경우 함수 포인터는 단순히 오류를 반환하는 스텁 함수를 가리킵니다.

이 경우를 감지하기 위해 IsApiSetImplemented 함수를 사용하여 지정된 API 구현의 기본 가용성을 쿼리할 수 있습니다. 이 테스트는 이 함수를 호출하면 API의 기능 구현이 실행될 수 있음을 확인합니다.

다음 코드 예제에서는 **IsApiSetImplemented** 사용하여 WTSEnumerateSessions 함수를 호출하기 전에 현재 디바이스에서 사용할 수 있는지 여부를 확인하는 방법을 보여 줍니다.

c++

```
#include <windows.h>
#include <stdio.h>
#include <Wtsapi32.h>

int __cdecl wmain(int /* argc */, PCWSTR /* argv */ [])
{
    PWTS_SESSION_INFO pInfo = {};
    DWORD count = 0;

    if (!IsApiSetImplemented("ext-ms-win-session-wtsapi32-l1-1-0"))
    {
        wprintf(L"IsApiSetImplemented on ext-ms-win-session-wtsapi32-l1-1-0 returns
FALSE\n");
    }
    else
    {
        if (WTSEnumerateSessionsW(WTS_CURRENT_SERVER_HANDLE, 0, 1, &pInfo, &count))
        {
            wprintf(L"SessionCount = %d\n", count);

            for (ULONG i = 0; i < count; i++)
            {
                PWTS_SESSION_INFO pCurInfo = &pInfo[i];
                wprintf(L"    %s: ID = %d, state = %d\n", pCurInfo->pWinStationName,
                    pCurInfo->SessionId, pCurInfo->State);
            }
```

```
            WTSFreeMemory(pInfo);
        }
        else
        {
            wprintf(L"WTSEnumerateSessions failure : %x\n", GetLastError());
        }
    }

    return 0;
}
```

Last updated on 2025. 03. 13.

Windows 우산 라이브러리

> **Important**
>
> 이 항목의 정보는 모든 버전의 Windows 10 이상에 적용됩니다. 여기서는 이러한 버전을 "Windows"라고 하며 필요한 경우 예외를 호출합니다.

우산 라이브러리 Win32 API의 하위 집합을 내보내는 단일 정적 링크 라이브러리입니다. 예를 들어 **OneCore.lib**라는 우산 라이브러리는 모든 Windows 디바이스에 공통적인 Win32 API의 하위 집합에 대한 내보내기를 제공합니다.

우산 라이브러리의 API는 모듈 범위(모듈이 API 집합 또는 DLL)에 걸쳐 구현될 수 있습니다. 그러나 우산 라이브러리는 세부 정보를 추상화하여 운영 체제 버전에서 앱을 더 이식 가능하게 만듭니다. 데스크톱 앱 또는 드라이버에서 관심 있는 API 집합이 포함된 우산 라이브러리를 연결하기만 하면 됩니다.

테이블 확장

도서관	묘사
OneCore.lib	모든 Windows 10 디바이스 이상에 공통적인 Win32 API의 하위 집합에 대한 내보내기를 제공합니다. `OneCore.lib`(및 다른 라이브러리 없음)를 연결하여 해당 API에 액세스합니다. `OneCore.lib`연결하고 해당 라이브러리에서 Win32 API만 호출하면 데스크톱 앱 또는 드라이버가 모든 Windows 10 디바이스 이상에서 성공적으로 로드됩니다.
OneCore_apiset.lib	`OneCore.lib`동일한 범위를 제공하지만 API 집합 직접 전달사용합니다. 연결 `OneCore_apiset.lib` 대상으로 하는 SDK 버전과 관련된 Windows 버전 이상과만 호환됩니다.
OneCoreUap.lib	WinRT(Windows 런타임)를 지원하는 모든 Windows 10 디바이스 이상에 공통적인 Win32 API 하위 집합에 대한 내보내기를 제공합니다. `OneCoreUap.lib`(및 다른 라이브러리 없음)를 연결하여 해당 API에 액세스합니다. `OneCore.lib`연결하고 해당 라이브러리에서 Win32 API만 호출하는 경우 데스크톱 앱 또는 드라이버는 UWP를 지원하는 모든 Windows 10 디바이스 이상에서 성공적으로 로드됩니다.
OneCoreUAP_apiset.lib	`OneCoreUAP.lib`동일한 범위를 제공하지만 API 집합 직접 전달사용합니다. 연결 `OneCoreUAP_apiset.lib` 대상으로 하는 SDK 버전과 관련된 Windows 버전 이상과만 호환됩니다.

Last updated on 2025. 03. 13.

데스크톱 앱에서 호출할 수 있는 WinRT API

대부분의 WinRT(Windows 런타임) API 데스크톱(.NET 및 네이티브 C++) 앱에서 사용할 수 있습니다. 그러나 일부 WinRT 클래스는 UWP 앱에서만 사용하도록 설계되었으며 지원됩니다. 여기에는 CoreDispatcher, CoreWindow, ApplicationView및 일부 관련 클래스가 포함됩니다. 다른 WinRT 클래스는 특정 멤버를 제외한 데스크톱 앱에서 작동합니다.

자세한 내용은 데스크톱 앱 지원되지 않는 Windows 런타임 API참조하세요. 사용 가능한 경우 이 문서에서는 지원되지 않는 API와 동일한 기능을 달성하기 위한 대체 API를 제안합니다.

Last updated on 2025. 03. 13.